Feedback-Based Gameplay Metrics and Gameplay Performance Segmentation:

An audio-visual approach for assessing player experience.

A thesis
submitted in fulfilment
of the requirements for the degree
of
Doctor of Philosophy
at
The University of Waikato
by

RAPHAËL MARCZAK

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

2014

Abstract

Gameplay metrics is a method and approach that is growing in popularity amongst the *game studies* research community for its capacity to assess players' engagement with game systems. Yet, little has been done, to date, to quantify players' responses to feedback employed by games that conveys information to players, i.e., their audio-visual streams. The present thesis introduces a novel approach to *player experience* assessment - termed *feedback-based gameplay metrics* - which seeks to gather gameplay metrics from the audio-visual feedback streams presented to the player during play. So far, gameplay metrics - quantitative data about a game state and the player's interaction with the game system - are directly logged via the game's source code. The need to utilise source code restricts the range of games that researchers can analyse. By using computer science algorithms for audio-visual processing, yet to be employed for processing gameplay footage, the present thesis seeks to extract similar metrics through the audio-visual streams, thus circumventing the need for access to, whilst also proposing a method that focuses on describing the way gameplay information is broadcast to the player during play.

In order to operationalise feedback-based gameplay metrics, the present thesis introduces the concept of *gameplay performance segmentation* which describes how coherent segments of play can be identified and extracted from lengthy game play sessions. Moreover, in order to both contextualise the method for processing metrics and provide a conceptual framework for analysing the results of a feedback-based gameplay metric segmentation, a *multi-layered architecture* based

on five gameplay concepts (system, game world instance, spatial-temporal, degree of freedom and interaction) is also introduced.

Finally, based on data gathered from game play sessions with participants, the present thesis discusses the validity of feedback-based gameplay metrics, gameplay performance segmentation and the multi-layered architecture. A software system has also been specifically developed to produce gameplay summaries based on feedback-based gameplay metrics, and examples of summaries (based on several games) are presented and analysed. The present thesis also demonstrates that feedback-based gameplay metrics can be conjointly analysed with other forms of data (such as biometry) in order to build a more complete picture of game play experience. Feedback based game-play metrics constitutes a post-processing approach that allows the researcher or analyst to explore the data however they wish and as many times as they wish. The method is also able to process any audio-visual file, and can therefore process material from a range of audio-visual sources.

This novel methodology brings together game studies and computer sciences by extending the range of games that can now be researched but also to provide a viable solution accounting for the exact way players experience games.

Acknowledgements

Throughout this thesis, except in the present acknowledgements section, I have avoided writing in the first person. More than a stylistic choice, it has been a conscious decision to acknowledge the fact that this thesis is not the work of a single person, but has been the result of a strong collaboration process. I know many PhD candidates that would say that writing their thesis has been one of the tougher experiences of their student lives. It may be. But I have been lucky to be supported by wonderful people who took the pressure from my shoulders, and made this their pressure. This thesis, and my whole PhD experience, would definitely not have been the same without the help of a number of extraordinary people, and they deserve to be properly thanked for their contributions.

First and foremost, I would like to thank **Gareth Schott**, my chief supervisor at the University of Waikato, for his constant help and support, which went beyond the thesis writing time frame. Not only did he provide me with constant and detailed feedback about the different chapters of the present thesis; but I will not forget the time he spent on my behalf while I was not even a PhD candidate, reviewing my application and helping me to eventually be part of his research project. He also showed a great open-mindedness to the fact that English is not my mother tongue, and that my background is from computer science. That probably made the chapters I handed him very opaque at first, but he always tried to understand the point I was making in order to provide feedback accordingly. I cannot imagine the time it took him to review each chapter of this thesis, and for that, I want to deeply thank him. I also would like to acknowledge his deep contribution to the multi-layered model presented in Chapter 4, as he helped a lot in formalising how hierarchical and transversal analysis can work together inside a single architecture.

I would like to thank **Bevin Yeatman**, who also supervised my thesis writing at the University of Waikato. He had been added to my supervisory panel when I started to feel lost and uncertain about the direction to take, not only in my thesis, but also in my future life. He has been here every time I was in need of a strong motivating atmosphere. He provided valuable feedback for each of the chapters constituting this thesis, trying to have an external look at the work, while also offering me (spiritual) food when my brain needed a rest, or at least an alternative focus. He has been caring for my mental health during the whole thesis, forcing me to focus on my thesis writing when the time was right, and to get away from my thesis when the time was opportune. The life and organisational lessons he taught me during these three years are as valuable as the academic lessons I learned. I owe him so much; I hope I will be able to return the favour one day.

I would like to acknowledge the **University of Waikato International Scholarship** that funded my education, and allowed me to focus full-time on my thesis. This is a comfort I never took for granted, so thank you very much. I would also like to thank the **Conseil Régional d'Aquitaine** and **Pôle Emploi International**, and especially **Catherine Galharret**, **Isabelle Pons** and **Christine Dupont**, who helped me to obtain the **Cap Mobilité** fund, which paid my expenses to move worry-free to New Zealand. I would like to thank **The Royal**

Society of New Zealand, Marsden Grant for funding the equipment used during the game sessions, as well as registrations and travel expenses that allowed me to attend several core conferences.

I would like to thank **Pierre Hanna** and **Jean-Luc Rouas** from the Laboratoire Bordelais de Recherche en Informatique (LaBRI, University of Bordeaux, France), who provided the sound processing software systems and algorithms that are used in the present thesis (cross-correlation, similarity matrix, speech detection and Harmonic Pitch Class Profiles methods) in order to be able to test as quickly as possible the validity of sound processing applied to gameplay footage. I would also like to thank **Mathias Robine**.

I would like to thank the **Laboratoire Bordelais de Recherche en Informatique (LaBRI)** for having invited me twice in order to work directly at the laboratory with Pierre Hanna and Jean-Luc Rouas, and also for having paid my travel expenses. I would also like to thank the LaBRI for having authorised me to use their video-conference system the day/night of my thesis defence.

I would like to thank **Lennart Nacke**, who supervised me from the University of Ontario. His feedback toward the computer science contributions to this thesis has been valuable. He also recommended that I study the game ontology project and the gameplay segmentation concept, which are an integral part of the present thesis.

I would like to thank my external examiners, **Doctor Brigid Costello** and **Associate Professor Georgios N. Yannakakis**. I would like to especially acknowledge their time and effort in reviewing my thesis. The comments and feedback they sent were really detailed and therefore extremely helpful in reviewing the work for producing the final version. My supervisors have constantly helped me take a step back, and they helped me take a step further.

I would like to thank **Jasper van Vught** who has been an office mate during the three years of my study at the University of Waikato, and who not only helped me settle in from the day I arrived in New Zealand, but also constantly provided me with relevant game studies literature to read in order to improve my knowledge of the discipline. He is notably the one that advised me to read Computers as Theatre by Brenda Laurel, and this has been the inspiration for my using the word "performance" in the present thesis to describe the segmentation process. I also would like to thank him for his regular input on my work and my English-writing skills.

I would like to thank **Fiona Martin** who meticulously proofread the first half of this thesis. Working with her during the last weeks of my thesis writing has been a real pleasure, as she has been really thoughtful and careful, at this stressful time, so that I did not feel overwhelmed by her feedback.

I would like to thank everyone who contributed to the organisation of the game sessions, and who participated in the data collection. Thank you to **Jenny Baker** and to all the participants and parents who were willing to be part of the research project. Thank you to **Jasper van Vught** and **Gareth Schott** who have been with

me during the gameplay sessions, and conscientiously recorded and stored the different gameplay footage which has then been processed in order to validate the approach presented in the present thesis. Thank you to **Leanne Neshausen** who designed the method presented in Chapter 7, which correlates feedback-based gameplay metrics with biometric measurements. Thank you to **Luke Jacobs** who helped setting up the computers. Thank you to **Aaron Hawthorn** and **Euan Kilgour** for having set up the computers, and for having solved, sometime in less than a minute (the time to send an email and receive an immediate answer), all the computer-related issues that could have prevented the sessions taking place. Thank you to **Bill Rogers** for having lent us a gaming machine, and providing feedback about the methodology.

I would like to thank everyone that helped me with administrative tasks, so that I was able to focus on my research work: **Athena Chambers**, **Candice Duke** and **Carolyn Henson**. I would also like to thank everyone from the **Screen and Media Studies Department**, who helped me every time I needed support.

There are also a lot of people who, even without being directly involved in the thesis writing process, helped this thesis to exist. I am thinking of course of **Athena Chambers**, who cared about me as a parent would do, and solved all my everyday life worries, while being a close confidant. **Ben Lenzner**, mon frère, who has always been there when I needed to clear my mind**. Fiona Jackson**, and her daughters **Isabella** and **Gabby**, for always having been so welcoming and caring about me. **Kasha Latimer** and **Omar Alagna**, who showed me what the true definition of "home away from home" is. **Jerome Arfi** who welcomed me the day I arrived in New Zealand, and shared his experience as a French foreigner in order to ease my integration. **Mylène Delarue**, with who I now share amazing memories of New Zealand. **Fanny Jorand**, whom frankness and sense of humour helped me to grow up and taught me how to stand for what I believe is important. **Maxime Rouast** and **Stephanie Smet** who helped me settle in, and regularly took me on trips around New Zealand to take my mind off things. And **Ashley Hooper**, who opened my mind to so many topics that I would need a full second thesis to exhaustively list them.

I would also like to thank **Shraddha Borawake**, **Jordan Browne**, **Baltazar Campos**, **Chris Clark**, **Sue Clearwater**, **Cathy Coleborne**, **Anthea Fester**, **Craig Hight**, **Kirsty Horrell**, **Willemijn Krijnen**, **Allan Lenzner**, **Alex Pelham-Waerea**, **Alistair Swale**, and **Daniel Trainor**. You are all amazing people, and I am lucky to have met you.

The full experience of writing a thesis is not limited to the present time frame. It also encompasses the past experiences, the people I met who made me who I am today.

I first would like to thank **Myriam Desainte-Catherine**, who was my first ever supervisor in the academic world, and who sent me the position offer that eventually led to this thesis. She showed me what it is to work in the academic world, and her kindness, open-mindedness and ethics ended up convincing me I needed to be part of this world too. I would also like to thank **Pascal Baltazar** and

everyone involved in the VIRAGE project. I would also like to thank **Pierre Casteran** who has always been keen to review my different applications. I would also like to thank the **Studios de Création et de Recherche en Informatique et Musique Electroacoustique** (**SCRIME**), and everyone I met there, for the help and support I constantly received when I was still working in Bordeaux. Special thanks to **Annick Mersier**, **Joseph Larralde** and **Gyorgy Kurtag**.

I would like to thank **Mireille Garreau** and **Antoine Simon**, from the Laboratoire Traitement du Signal et de l'Image, who also contributed to the growth of my interest in the academic world. I would also like to thank everyone at **BeTomorrow**, and especially **Jean-Dominique Lauwereins**, **Laurent Alvaro**, **Alexandre Ribeiro**, **Sylvie Clin** and **Thomas Cassany**, for having trusted me and initiated me to the game development world.

I would like to thank my former colleagues and friends at LaBRI, and especially those who inhabited the Bureau 327: **Antoine Allombert**, **Florent Berthaut**, **Jérôme Charton**, **Allyx Fontaine**, **Noemie-Fleur Sandillon-Rezer**, **Luc Vercellin**. Your friendship and fascinating research work made these years at LaBRI some of the best in my life. I would also like to thank **Damien Cassou** and **Jenny Grenier** for having always been there for me when I needed support. I would like to thank **Laurent Garnier**, **Thomas Rocher** and **Noel Gillet**.

I would also like to recognize the strong impact that some of my best teachers had on my life, providing strong life-lessons too: **Françoise Lacavalerie**, **Mr. Billault**, **Mr. Marchand**, **Delphine Reyss**, **Denis Lapoire**, **Jacques Olivier Lachaud**, **Géraud Senizergues**, **François Pellegrini**, **Kenneth Beirne**, **Aymeric Vincent**, **Christian Eloy**, **Laurent Soulié**, **Guillaume Martial** and **Sylvain Marchand**.

Finally, I would like to thank all my friends and family who tried to understand and respect my choice, and who never tried to prevent me from doing this “crazy” jump into the void. I am coming back better and stronger, and this is all thanks to you, because you let that happen. Despite the thousands of miles between us, the love you sent me, day after day, was received straight into my heart, and kept me standing up even when I felt down. Thank you so very much to my friends **Aurélie Chabrier**, **Emilie Chabrier**, **Agnes Chabrier**, **Philippe Chabrier**, **Christophe Garcia**, **Marie-Camille Bernard**, **Martine Bernard**, **Audrey Mouneyres**, **Sébastien Claret**, **Jean Pinaud**, **Stéphane Dolique**, **Chantal Lubespère**, **Claude Lubespère**, **Sylvie Gervais**, **Melanie Laborde**, **Helene Rouanet**, **Bruno Valèze**, **Yoann Yonnet**, **Michael Pierre**, **Thibaud Hervé**, **Ruddiane Hervé**, **Elisabeth Boribon**, **Pierre-Thomas Lorette**, **Laurie Mazinq**, **Nicole Lay-Mazinq**, **Odile Dubouilh**, **Nicole Nivet**, **Veronique Colombel**, **Jean-Yves Colombel**, **Laetitia Arnaud-Alexandre**, **Axelle Garcia**, **Delphine**, **Liliane**, **Marcel Martinez**, **Catherine Garcia, Gerard Le Paih**, **Serge Démory**; and my family **Concetta**, **Michel**, **Anne-Lise**, **Dominique**, **Liliane** as well as my grandparents.

Special thanks to **David S. Armstrong** and **Stuart Hofstetter**, for giving me the key to amazing new worlds. Special thanks to **Yann Leroux**. Special thanks to **Alexandre Altain**, **Aurore Reichert**, **Jean Pascal Boffo**, **Olivier Libaux**, **Yoann Lemoine**, **Jérôme Patard**, **Stephen Fry**, **Jim Sterling**, **Yann Barthès**, **Beth Gibbons**, **David Massard** and **Romain Galland**. Special thanks to **Henri Pull**.

My thanks to **you,** the reader, who may make my thesis live in future research projects.

For those I am leaving, I will miss you. For those I am about to be reunited with, I missed you.

To all of you, with all my love: thank you,

This thesis is dedicated to the memory of Kaleb.

Contents

List of Figures

List of Abbreviations

BBFC	British Board of Film Classification
ESRB	Entertainment Software Rating Board
FPS	First Person Shooter
GSR	Galvanic Skin Response
HCI	Human Computer Interaction
HUD	Head Up Display
NPC	Non Player Character
OFLC	Office of Film and Literature Classification
OCR	Optical Character Recognition
OSC	Open Sound Control
PEGI	Pan European Game Information
RPG	Role Playing Game
SDK	Software Development Kit
TRUE	Tracking Real-Time User Experience
XP	Experience

Chapter 1

Introduction

In order to enhance the visibility of and publicity around their game, the publishers of the text-based adventure game *Dracula* (CRL Group, 1986) hoped that the British Board of Film Classification (BBFC) would impose an age-restriction on a videogame. The idea behind this ploy was to create a *forbidden fruit* effect as a means of guaranteeing a raise in sales. While the game makers were successful in achieving a R15 rating from the BBFC, this seemingly harmless marketing ploy had deep implications. *Dracula* became the very first game to be ever rated, by a process that was not originally designed to classify videogames. Given that the BBFC was not authorised to judge the textual content (as they do not classify literature), the decision was made on the basis of the different depictions of gruesome events depicted in the game via still images (cf. Figure 1).

Figure 1. *Dracula Adventure Game*: death screen

This decision served to legitimate the idea that film and videogames share enough similarities to be classified using similar assessment criteria. As a result, the video and sound proponents of depictions of violence in videogames would go on to

become core elements of the assessment of what the *experience* of playing a game entails.

The idea that videogames and films are similar enough to be equivalently classified has persisted for decades. The BBFC, while initially assessing film content, was the administration in charge of videogame classification in the United Kingdom (UK) until 2012. Yet, similar systems still exist in countries like New Zealand with the Office of Film and Literature Classification (OFLC) and Australia, with the Australian Classification Board (ACB). The *New Zealand Classification Act 1993* (New Zealand OFLC, 1993) for instance stipulates that videogames are to be considered similar to films on the basis that both share "moving images content" (1993, p. 14). The assumption is that audio-visual information is equally perceived by the end-user, regardless of the modality involved in the act of playing a videogame.

In 2001, the year identified by Espen Aarseth as "Year One of Computer Game Studies" (Aarseth, 2001), the need for a new discipline, termed *game studies*, and dedicated to the specificities of the videogame media, was debated. Aarseth not only acknowledges that "games are not a kind of cinema or literature" (2001) but also that "colonising attempts from [cinema and literature] have already happened, and no doubt will happen again" (2001). The above example of *Dracula* being classified by an office originally focused on film rating is an example of these colonising attempts, implying that movies and videogames are similar enough to be considered as equal. However, Aarseth also warns about the dangers of ignoring the specificities of different media, as he believes - referencing the theory

of *media-blindness* by Liv Hausken (Hausken, 2004) - that "a failure to see the specific media differences leads to a "media neutral" media theory that is anything but neutral" (2001).

Trying to understand the specific experiences that constitute playing a videogame – as opposed to more traditional interactive software or linear media – has been a core topic of interest for the *game studies* research community for more than a decade now. By defining a videogame as "both [an] object and process [… that] must be played" (Aarseth, 2001), and by considering that "playing is [then] integral, not coincidental" (2001), the game studies discipline has begun to appreciate the necessity of assessing the play, by measuring *player experience* and focusing on the rationale underlying player interactions and reactions. A wide range of approaches, both qualitative and quantitative, are now being applied to a more exhaustive assessment of what player experience entails (Canossa, Drachen, & Rau Møller Sørensen, 2011; IJsselsteijn et al., 2008; Kim et al., 2008). In 2009, Lennart Nacke proposed a *player experience model* (Nacke, 2009), focusing on establishing the foundations for formalising a wide, diverse, and complex assessment of what play entails. Understanding player experience can be conceptualised as trying to understand the relationship between numerous factors likely to influence moments of play. These factors include *temporality* (before, during and after play); the *game systems*; the input and activities of the *player(s)*; and the *context* of play. One quantitative method in particular has been promoted heavily for the empirical investigation of gameplay. This method is transversal, covering the *game system*, the *player* and the *context* of play. Termed *gameplay metrics*, this method functions by directly logging data from the game, that is by

extracting information representative of both the videogame state and player interactions. Gameplay metrics has been defined as

> measures of the player behaviour, e.g. navigation, item- and ability use, jumping, trading, running and whatever else players actually do inside the virtual environment. [Several] types of information can be logged whenever a player does something – or is exposed to something – in a game: What is happening? Where is it happening? At what time is it happening? In addition, when multiple objects (e.g., players) interact: to whom is it happening? (Drachen, Seif El-Nasr, & Canossa, 2013)

The variety of gameplay elements covered by the gameplay metrics method has become more appealing to the game studies community, as it potentially offers insights into playability (Nacke, Drachen, et al., 2009), and into the assessment of the compatibility between player interaction and the designers intentions (Kim et al., 2008). Gameplay metrics has also been used for more player-centric design solutions, like the determination of play personas (Drachen & Canossa, 2008) being the identification of different behaviours through the analysis of *gameplay metrics* that can, together, provide insight about a general play-style.

Several limitations currently restrict the use and application of gameplay metrics. The first limitation is implied by the method of gathering. In fact gameplay metrics demands that information is drawn directly from the game system, necessitating that the videogame source code is made available, or contains a

built-in option (such as built-in metric system, or a modding system (Sasse, 2008)). Furthermore, the current scope of implementation is, in general, restricted to the sole purpose of improving game design. This provides a rationale for having access to a game source code, and requires an agreement between the game's developers and the researchers (open source games are an exception (Pedersen, Togelius, & Yannakakis, 2010)). Design and implementation of *gameplay metrics* therefore coincides with, or seeks to influence the agenda of developers over academic agenda.

The second limitation relates to the scope of gatherable information. Given that gameplay metrics refers to the state of the game system and of the player's interaction with that system, it does not currently account for the actual form in which the game system is presented to the player, that is, via audio and video information and feedback.[1] Indeed, it would appear that gameplay metrics are currently both too broad and too restrictive at the same time. Too broad, because they embrace and include numerous gameplay elements, that function and operate out-of-sight for the player. For instance, variables can be recorded because they exist in the game source code, but from a player's perspective they are not yet visible; for example, the number of enemies that reside ahead in an undiscovered room. They are also too restrictive, because audio and video streams also contain elements that impact upon the player's experience, and these are not addressed by

[1] The haptic feedback, more scarcely used, is not covered in the present thesis, but is worth becoming the topic of a specific research project.

gathering variables at the source code level. This is, for instance, the case of sound and video parameters such as sound frequencies or colour distributions.

This thesis reflects on the paradox between the potential offered by gameplay metrics for understanding player experience, and its current level of inaccessibility, which makes it a difficult method to apply insofar as it restricts the kind of games that can be studied. In doing so, this thesis seeks to fulfil the potential of game metrics, by translating the method from a non-representative account of the exact way in which gameplay information is conveyed to the player, to a quantification of gameplay elements as they are actually experienced by a player through the main feedback streams.

The main objective of the present thesis is to study the possibilities for rethinking and adapting gameplay metrics, circumventing its current data-gathering limitations, by focusing precisely on what the player is presented with during play as communication and feedback. Adaptation will take the form of a novel method and process labelled *feedback-based gameplay metrics*. The present thesis core idea is to appropriate, adapt and apply algorithms employed in the processing of video and audio streams. Such algorithms have yet to be applied to videogame analysis. The main strength of feedback-based gameplay metrics is their ability to highlight, from the audio-visual feedback streams, moments of play that convey a specific experience of play. The process of identifying these moments of gameplay in relation to gameplay understanding is also discussed in the current thesis, a process defined as *gameplay performance segmentation* (cf. Chapter 3).

The aim of this research and its contribution to knowledge is in bringing together different disciplinary approaches, and allowing one academic endeavour to benefit from the method of another. More specifically, the present thesis is trying to establish two connections: one between feedback-based gameplay metrics and classification systems, and one between feedback-based gameplay metrics and player experience assessment. The expectation is that a connection can then be, by association, established between classification systems and player experience.

However, it must be acknowledged that the aim of the present thesis does not include an assessment or evaluation of current classification systems, as this would require that consideration be given to the manner in which legislation and policy frame the experience of play from a legal perspective. Nor is the present thesis attempting to interject directly into debates on the impact of games on those who play them. The main objective of the thesis is to try to create a means whereby theories that account for the structure and experience of play may be extended beyond the academy and into public debates surrounding games. It seeks to provide a tool and means for game analysts grappling with the complexities of games as objects and as interactive experiences.

It must also be acknowledged that while the present thesis presents audio-visual processing algorithms and produces a software system for gathering feedback-based gameplay metrics, this work is submitted as a PhD completed in the School of Arts at the University of Waikato, and aims to contribute to the humanities-instigated field of game studies. At the same time, the rationale for a PhD with a strong methodological focus stems from a need to be able to communicate game

studies approaches within a social science research context via a solution that draws on techniques developed within the computer sciences. While this thesis does apply computer science thinking and methods, its structure will differ from PhD theses submitted under computer science. For instance, to ease the reading, the validity of each algorithm presented in Chapter 5 and 6 is discussed with the algorithm presentation, instead of dedicating a full specific chapter to a discussion of validity.

The following section presents the context of the thesis by describing current classification systems, as their study has been the basis of the rationale for new quantitative data. By highlighting the differences between assessment of player experience in videogame classification systems, and the game studies discipline, the introduction then illustrates how the new approach of feedback-based gameplay metrics can be used to establish a connection between the different ways of assessing videogame experience, before finally presenting the thesis structure and outline.

1.1 Classification systems designed for videogame assessment.

If the *Dracula* example presented above illustrates how classification systems originally designed for film rating have been applied in rating of videogames, the present section describes classification systems that have been created specifically for videogames. The current section shows that, while being actually closer to the experience of game as some ludic aspect are taken into account for the rating; such classification systems still rely mainly on the audio-visual depiction of elements of gameplay, which are closer to a watcher's experience than a player's experience.

In 1992, the release of *Mortal Kombat* (Midway Games, 1992) initiated strong public debates about the presence of violence in videogames. The opportunities offered by this fighting game – including engaging in torture and killing opponents that are injured at the end of fights (termed *finish him* or *fatality* moves, see Figure 2) – confronted societies with the increasing realism strived for by developers, and which involved players in violent acts. The strong debates surrounding the content of *Mortal Kombat* pushed the introduction and creation of specific classification systems, designed to inform the consumer and regulate the sales of videogames. The Entertainment Software Rating Board (ESRB), founded in 1995 in USA, and the Syndicat des Editeurs de Logiciels de Loisirs (SELL) in France, are good examples of regulation bodies created to monitor and control the

distribution of, and access to game content. SELL has since been absorbed[2] with other European classification systems into the Pan European Game Information (PEGI) organisation, established across Europe in 2004.

Figure 2. ***Mortal Kombat*****: fatality move**

Both the ESRB and the PEGI systems were the first classification systems that attempted to consider videogames as an entity distinct from other media, including particular elements that reflect a player's perspective. For instance, both systems acknowledge that the ludic aspect of the games needs to be taken into account. Under the PEGI classification system, an age-rating is for instance lower if the use of a drug penalises the player rather than awarding a bonus. However, audio and video depictions continue to override any ludic aspect of the game in arriving at a final decision. For instance in *Mortal Kombat*, initiating a *fatality* move is a very complex interaction (a sequence of buttons must be pushed in the

[2] The SELL still exists, but no longer for Classification purposes.

correct order, within a narrow time frame). The question can be raised whether an objectionable content that is easily avoidable should be classified similarly as an objectionable content that cannot be ignored. But the existence of at least one objectionable sequence, regardless of the actions leading to it, justifies a high age-rating under the ESRB and PEGI systems.

Furthermore, the way videogames are classified is still based mainly on an assessment of the experience of someone watching the game, rather than the experience of someone playing the game, and carries the assumption that players engage with the screen in the same way as film viewers do. The ESRB system classifies videogames based upon a DVD of game sequences sent by the publisher, in which the publisher selects moments of play believed to be objectionable. The classification is then based on a document similar to a movie trailer. The PEGI system classifies videogames by utilising a survey completed by the publisher itself, on which one single "yes" answer to any age-related question overrides any lower age decision (a single R18 "yes" answer will override all other R16 questions for instance). The survey ("PEGI Questionnaire," n.d.) mostly relies on audio-visual depictions, as illustrated in Figure 3. The word *depiction* appears regularly and the criteria rely both on the notion of what is visible, and on what form the visible content appears, for example "large amounts of blood or gore" vs. "disappear in a puff of smoke".

Depictions of gross violence, which includes torture, dismemberment, sadism and horrific depictions of death or injury towards human-like or animal-like characters. **Gross violence** will mean **depictions** of decapitation, dismemberment or torture and other horrific methods of bringing death, severe pain or injury to the recipient. This will usually be associated with **large amounts** of **blood or gore**. The emphasis is on the horrific nature of the violence. The violence will not be treated as gross violence if the recipients die or are injured in an **unrealistic** manner. If they instantly **disappear in a puff of smoke** or are killed/injured and then come back to life **or appear uninjured** this will not be treated as gross violence. The characters must **look like** humans or animals. If a character **looks like** a human it should be treated as human even if it is unrealistic.
Depictions of apparently motiveless killing or serious injury to multiple numbers of innocent human-like characters This is where groups of **human-like** characters are killed or injured at random for no apparent reason and deals with themes such as the killing of pedestrians in the street, shoppers in a shopping arcade and children in a school. The characters must **look like** humans or animals. If a character **looks like** a human it should be treated as human even if it is **unrealistic**.
Glamorisation of the use of illegal drugs The **depictions** will **show** that the user of the drugs is able to achieve success (win the game, get the girl, kill the enemy, commit the crime) after the use of illegal drugs. The drugs concerned should be real and be illegal (not fantasy or legal drugs).
Depictions of sexual activity with visible genital organs Sexual activity means all aspects of human sexual intercourse, masturbation and sexual foreplay (homosexual or lesbian activity included) where a male or female sexual organ is **visible**. The **depiction** of 'boobs and bottoms' or pubic hair only will not be treated as **visible** sexual organs.

Figure 3. PEGI system, selection of assessment criteria with emphasis on the audio-visual attributes

The New Zealand OFLC system is closer to an actual play experience, as a professional player is invited to play the game while classification officers assess it. However, even if the New Zealand system involves a player, the decision is still based on a viewing experience of gameplay, rather than a game-playing one.

The predominance of audio and video content as the main criteria for producing an age-rating decision can be illustrated using two descriptions of the same game: one by the ESRB committee,[3] and the other one by the online review magazine IGN[4]. The best-selling role-playing game *Skyrim* (Bethesda Softworks, 2011) was reviewed by the ESRB in a 250-word summary[5] as being a game containing sequences:

> highlighted by slow-motion effect, particularly for decapitation… large blood-splatter effects … some environments … are stained with blood or body parts … [possibility of killing] non-adversary characters [that] scream in pain amid splashes of blood or fire … dialogs [with] references to sexual material … alcohol [that] can be purchased and consumed by player's character throughout the game … [and possibility of engaging] in a drinking contest with another character, which eventually results in slurred speech. (ESRB, 2011)

Three quarters of the summary focuses on depictions, giving the reader a sense of a violence-focused game. Only the two introduction sentences counterbalance the overall feeling of violence: "[Skyrim] is a fantasy role-playing game in which players assume the role of Dovakin, a prophesied figure with the power to combat

---

[3] ESRB has been chosen because the decisions are transparent and all available online.
[4] IGN has been chosen because it is widely used in the videogame players' community.
[5] http://www.esrb.org/ratings/synopsis.jsp?Certificate=31575

dragons in the fictional world of Skyrim. As players traverse through mountainous ‘open-world’ environments, they complete missions and quests that impact the eventual fate of their character” (ESRB, 2011).

The IGN review, on the other hand, hardly mentions any of the concerns highlighted by the ESRB summary. Only three sentences, in over more than 3000 words acknowledge the moments of concern for the ESRB system, but the impact of the violence seems diminished. The reviewer recognises the possibility of “blacking out after randomly entering into a drinking contest, luring innocents to their deaths at demonic shrines or complying with the demands of a cannibal haunting a morgue”, but describes them as, “peculiar events you may encounter in Skyrim's non-essential content”. The decapitation point is actually seen as part of a set of strategic possibilities, including the use of “points [that] can be stored until [being] absolutely sure about the best bonuses to unlock, which could be anything from a chance to decapitate enemies with a one-handed weapon swing, to zoom ability while using the bow and arrow”. The reviewer continues, “significant bonuses embedded into every perk tree, unlocking new abilities is always an exciting process” (Onyett, 2011, p. 2)

Slow motion effects are seen as a way to facilitate combat. For example, the game has “useful perks like a slow motion effect that triggers every time you block during an enemy’s power attack [and that] gives depth to the close-range fighting, so while the system may be simple at first, it improves if you [are] willing to invest” (Onyett, 2011, p. 2).

However most of the review is about strategy (skills, player's attributes, powers, interface), or story ("rich fictional legacy"). And even when the graphics or sounds are mentioned, the aesthetic aspect is the principal point:

> The visuals have also been dramatically improved over the last Elder Scrolls game. The sense of adventure and discovery is strong enough in Skyrim given how many cool items and quests there are to find, but the addition of beautiful scenery makes the inclination to obsessively scour Skyrim's landscapes irresistible. Mountains shrouded in mist ring every tract of open field, forest and marsh, and if you're willing to walk, you can eventually climb their slopes and view the rest of the world from above. Waterfalls tumble from high cliffs and split off into smaller tributaries as they wind through the rocks below, flowing across terrain that feels realistically varied. You'll see foggy mornings and crystal clear days, take in polychromatic aurorae streaked ribbon-like across night skies, rainstorms and near blinding blizzards, making it easy to drop what you're doing and survey a scene just to appreciate its beauty. (Onyett, 2011, p. 3)

The feeling given by each review is quite different, with the IGN review assessing the game for its relaxed and beautiful atmosphere, in comparison with ESRB, which focuses on potential objectionable contents, even if they are not core in the game. The IGN review seems to be closer to an actual player experience, exploring the ludic aspect of the different gameplay elements, and mentioning the

space for its immersive properties. On the other hand, the ESRB review is more of a watcher's experience, commenting on the aspect considered as ludic by the IGN review (for instance the slow motion as a strategic element) as being mainly an aestheticisation of violence.

1.2 Game studies: distinguishing spectator experience and player experience.

The core distinction illustrated in Section 1.1 is between a movie-spectator experience and a videogame-player experience. Such a distinction is also occasionally illuminated in the disciplinary differences that make up the game studies community and the differences in approach evident between the Social Sciences and the Humanities. For instance, if Gordon Calleja's (2011) work on player involvement and motivation mentions involvement dimensions that may be applicable to film, such as spatial or affective involvement; other dimensions reinforce game-like qualities such as kinaesthetic involvement (the ability to efficiently control an avatar) or ludic (the ability to build a strategy to win the game). Other research linking the flow theory of Csikszentmihalyi (1990) to videogames (Nacke & Lindley, 2008), which can be summarised as the motivation mechanism produced by a well-thought difficulty and learning curve, also requires an interaction component, absent from a movie experience, in order to be explained.

It could be argued, as suggested by the New Zealand Classification Act, that because videogames contain moving images and sound content, then at least part of the experience can be assessed from a watcher's perspective. No matter how much the level of interaction, involvement or motivation factors may differ,

assessing a videogame based on the audio-visual content, while not being exhaustive (classifying a film only based on the sound content would not be sufficient), is at least a first step to achieve an evaluation of the game that reflects the experience it offers.

However, the idea that interaction is just another layer alongside audio and video content has been challenged within the game studies community (Eskelinen, 2001; Juul, 2003). Interactivity can for instance be seen as a catalyst for modifying the way audio and video is perceived and experienced by a player, as there are numerous audio-visual elements that exist only to inform the player about such game elements, as avatar health or points earned (Fagerholt & Lorentzon, 2009; Ruch, 2010). These elements are likely to attract the player's attention from away the diegetic world, especially for highly strategic players who are motivated to complete a game. A spectator of the very same game would not necessarily notice or value these informational resources as having any value. However, such informational resources might dictate player behaviour. Schott (2009), in a study on player's perceptions of violence in videogames, reports how a player in one of his studies explained that killing an enemy in a first person shooter game was a way to avoid losing the game. The player justified their motivation by a "survival instinct" in terms of screen life, rather than influenced by the aesthetic of the violent content. An over-the-shoulder spectator, like a classification officer, cannot easily feel the survival instinct element, and can only rely on the audio-visual depiction, thereby understanding the experience as a gore-generating action.

1.3 Two separate worlds

So far, there has been no real convergence between advisory organisations and scholars of game studies, particularly regarding their respective understanding of *player* experience, as schematised in Figure 4.

One area which has influenced both the academic world and classification systems is the psychological experiential research focused on violence in videogames. This research is divided between the active user perspective and the active media perspective, and has been assessed and summarised in the UK government-commissioned Byron Report (Byron, 2008). This report sparked the UK government's move from the BBFC to the PEGI system in 2012. In the other direction, several academic works – such as research by Hasan et al. (2013) – tried to establish a link between violence in videogames, and violence in the real world, defining a violent videogame as one rated R18 by the PEGI system. This made the classification criteria an official way of defining violence. So far however, *game studies* have not been involved in such debates.

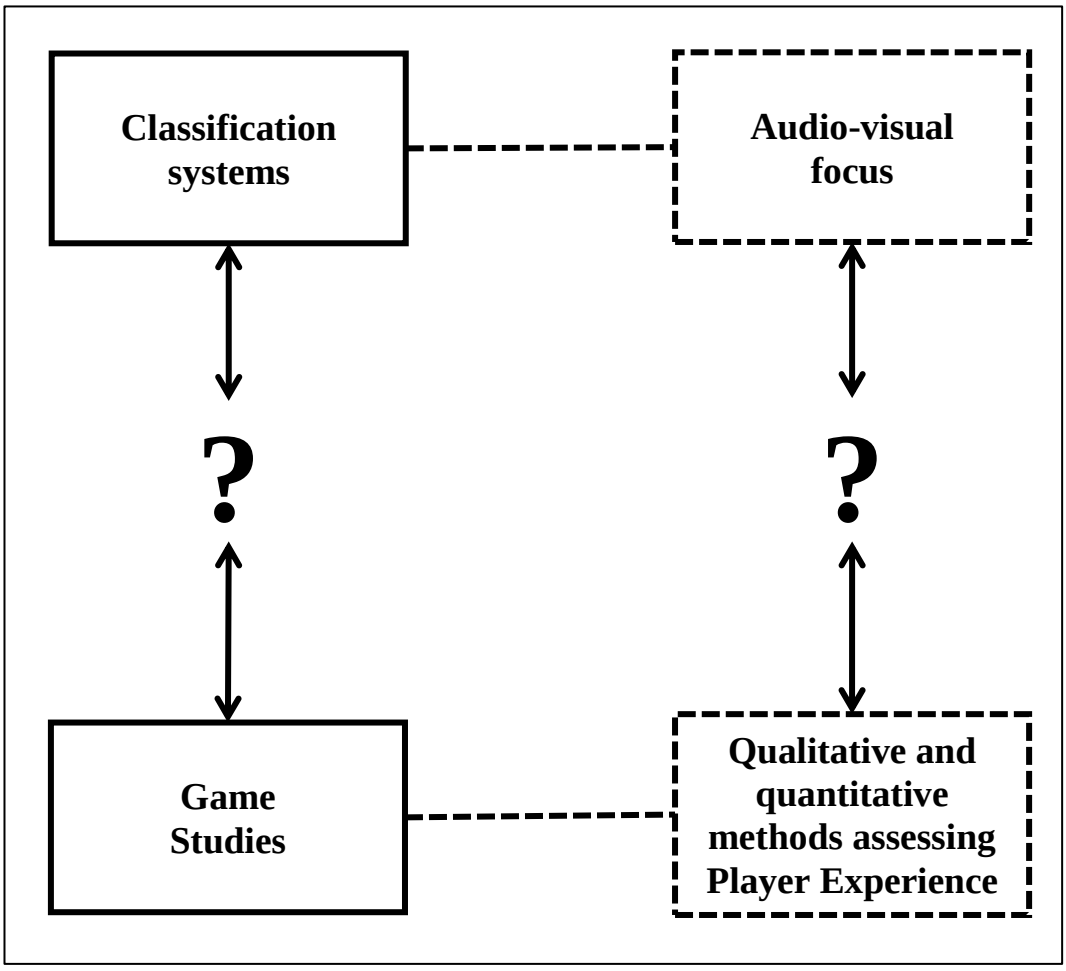


Figure 4. Two distinct considerations of videogames

1.4 Computer Science: the missing bridge

To provide a bridge between game studies and classification systems, the present thesis turns to computer science approaches that offer sound – and image – processing techniques that have the potential to be applied to media-segmentation (cf. Figure 5).

Indeed, *game studies* research has explored and utilised numerous methods, both qualitative (IJsselsteijn et al., 2008) and quantitative (Drachen et al., 2013), in order to establish how to assess player experience. Research typically takes into account three main elements of player experience: the player, the game system and the context (Nacke, 2009). However, while research has been conducted to demonstrate that the sound and video content of games has been studied as impacting player experience as an interactive medium (Grimshaw, Lindley, & Nacke, 2008), no method has yet been proposed to automatically process and quantify the audio-visual content that drives and reflects *play experience* through feedback streams presented to the player while he/she activates the play. The lack of such a method could be explained by the fact that much of the research focused on player experiences is concerned about finessing design principles. From a videogame classification perspective, the lack of assessment of the role of audio-visual content as it relates to the rules system prevents a connection with game studies theorisation.

In the computer sciences, an important and well-established strand of research seeks to automatically process video and audio streams in order to analyse content. Lengthy audio-visual documents are divided by cutting a media into

smaller segments that are topically-coherent, and labelling each segment with descriptive words for analysis and retrieval purposes (latter defined as *indexing*, see Chapter 3). Due to the large amount of diegetic and symbolic gameplay information enclosed in the audio and video streams of videogame play; this thesis will demonstrate over the course of the following chapters how an audio-visual recording of a gameplay footage can be broken down and analysed via the extraction of audio-visual metrics (called feedback-based gameplay metrics). This process will serve as basis for an automatic structural deconstruction of the gameplay footage. This will be achieved by applying, revising and adapting audio-visual algorithms currently in use in Computer Science (Gonzales & Woods, 2007; Huang & Liao, 2001; Pratt, 1978; Roads, 1996; Santos & Kim, 2006). This will hopefully allow an experiential reconstruction and understanding of player experience based only on audio and video content. These findings aim to realize a first step in reconciling the two perspectives.

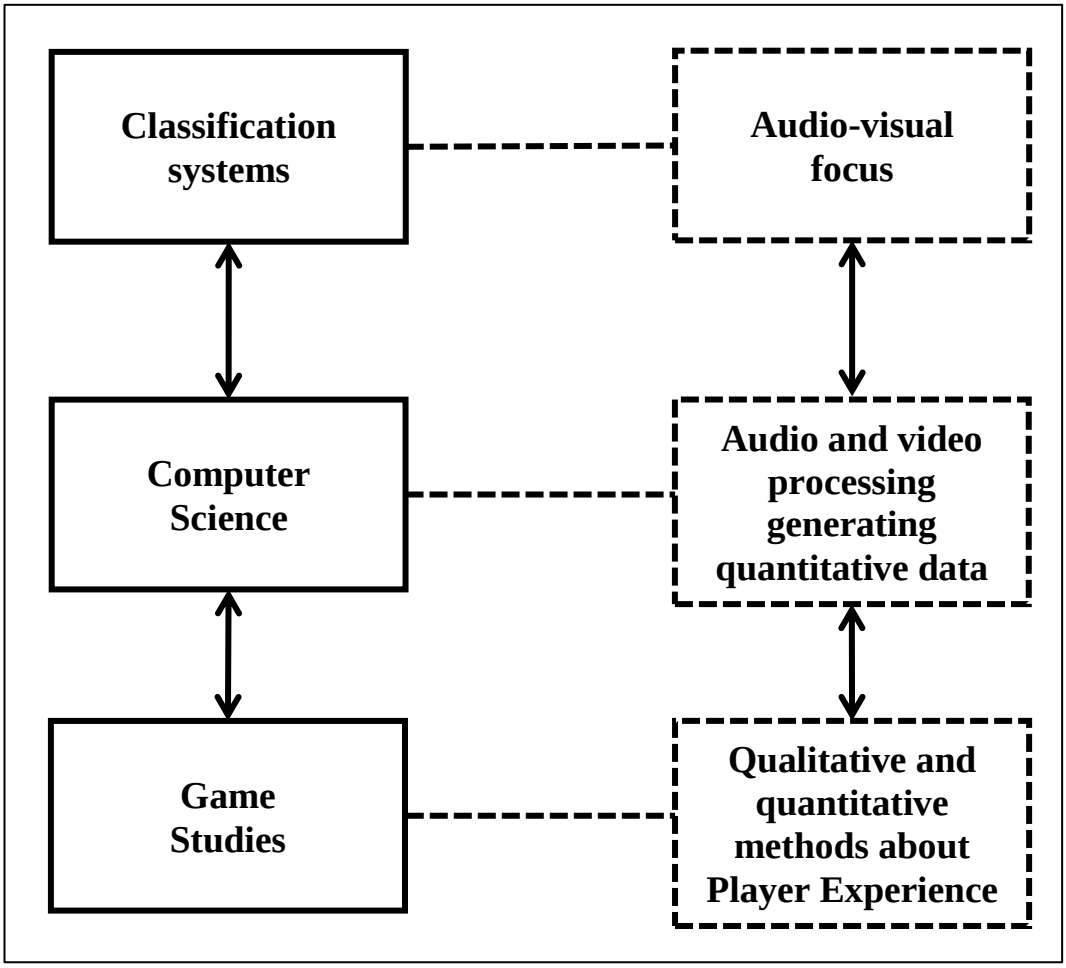


Figure 5. Computer Science as a bridge between Classification systems and Game Studies

1.5 Thesis outline

After presenting a rationale for developing a new method based on the audio-visual processing of videogame feedbacks, Chapter 1 describes the structure of the thesis. For each chapter, a short summary is provided, along with a diagram that serves to recapitulate the key points addressed in each chapter.

1.5.1 *Chapter 2 - Gameplay metrics (context)*

Review
- Player experience model
- Gameplay metrics
- Audio-visual elements in videogame

Definition
- Feedback-based gameplay metrics

Application
- Description of game sessions

Chapter 2 is dedicated to a review of the existing literature that deals with *gameplay metrics*, starting with the definition and assessment of *player experience*. The emphasis is put on the *player experience model* proposed by Nacke (2009), which includes an exhaustive consideration of the current qualitative and quantitative methods currently in use in the *game studies* community. The player experience model highlights three main factors influencing player experience: the player, the game system and the context. Chapter 2 then illustrates that gameplay metrics – time-stamped quantitative data about player interaction automatically logged by the game system – represents the only method that covers player, game system and context factors (as in the model proposed by Nacke), making gameplay metrics a method of choice for a wide consideration of player experience. The diversity of gatherable information makes gameplay metrics a method adaptable for assessing the usability of a game in

conjunction with the strategies of the game designer (Kim et al., 2008); understanding style of play, play persona and behaviour (Drachen & Canossa, 2008); or even adapting and modifying a game in real-time in order to model ways of playing and improve the *fun* feeling (Pedersen et al., 2010). The wide range of player experience assessment covered by gameplay metrics suggests that this is a method that could also be applied to quantifying audio-visual contents, and could be used to assess the game classification criteria.

However, Chapter 2 also introduces some key limitations of gameplay metrics. First, they have been mostly used for game design improvement purposes, and their validity has yet to be proved in a more generic context of use. Second, they rely on source code logging, meaning that the game source code must be made available (or previously adapted) for researchers to work with such metrics. Even if *gameplay metrics* can sometimes register events that also have a graphical representation, the required use of source code for logging implies that some out-of-sight information can also be logged (for instance the position of an enemy, regardless if the enemy is on screen or outside the player avatar view). *Gameplay metrics* is then a solution distinct for the actual audio-visual output of a game, and separate from how the player actually experiences the game feedbacks.

Chapter 2 then considers research about the way game information is broadcast to the player through audio and video streams, by highlighting how elements can be presented to the player so that they can correctly interact with the game system (Fagerholt & Lorentzon, 2009) and also be informed by abstract representations of all the senses the player cannot endure as not being directly present into the game

world (like avatar health for instance) (Ruch, 2010). Through three layers: symbolic, diegetic and audio-visual parameters, Chapter 2 illustrates how audio-visual information can be automatically processed to provide gameplay metrics-like outcomes, tied to the audio-visual representation of the game as directly broadcast to the player.

These metrics, called *feedback-based gameplay metrics* (Marczak, Schott, Hanna, & Rouas, 2013) are defined as being quantitative data about player interaction with the game system, extracted from the audio-visual output streams. Feedback-based gameplay metrics can be about audio-visual validation of player interaction (symbolic), an avatar status (symbolic), a change of space (diegetic) or even a global colour or sound frequency of a sequence (audio-visual parameters).

Chapter 2 finally presents data on game sessions that were organised at the University of Waikato as part of the current project, in which teenage participants played R16 games (under the New Zealand classification). During these sessions, several modalities of data have been recorded (biometry, eye tracking, gameplay footage) and synchronised, in order to be able to assess the validity of a method which gathers metrics from the audio-visual footage, while also considering the usefulness of feedback-based gameplay metrics in multi-modality research (cf. Chapter 7).

1.5.2 *Chapter 3 – Gameplay performance segmentation (context and model)*

Review
- Gameplay segmentation
- Performance notion
- Game Ontology Project
- Segmentation and indexing in the general context of document retrieval
- Closed vs. open system indexing
- Free-text vs. controlled-vocabulary indexing
- Topic homogeneity and breaks for segmentation
- Audio and video segmentation and indexing

Definition
- Gameplay performance segmentation

Application
- Examples of segmentation and indexing approaches adapted for videogame

Once the idea that gameplay-metrics can be automatically gathered from the audio-visual streams in the form of feedback-based gameplay metrics has been presented in Chapter 2, it becomes possible to study how feedback-based gameplay metrics can be applied to the understanding of player experience through the analysis of recorded gameplay footage.

Chapter 3 focuses on the notion of *gameplay performance segmentation*, which seeks to identify moments of play that convey coherent properties and can then be labelled using structural and experiential labels. These moments of play can be identified through the use of the feedback streams, using feedback-based gameplay metrics. This process is called *gameplay performance segmentation*.

The process is called gameplay performance segmentation in order to distinguish it from the *gameplay segmentation* model already defined by Zagal et al. (2008) as part of the Game Ontology Project (Zagal, Mateas, Fernández-Vara, Hochhalter, & Lichti, 2005). This model differs from the one presented in Chapter

3, although it shares certain similarities. Indeed, *gameplay* is a term that can be understood in two connected, but different, ways: *gameplay* as the game mechanisms and structure of the videogame; or *gameplay* as an actual instance of play, the moment of play when the player is interacting with the game system. In order to resolve the ambiguity, Chapter 3 defines gameplay performance as the result generated by an interaction between a player and the game system (using the term *performance* defined in Laurel's works (1993)). The gameplay performance segmentation notion becomes then a concept about segmenting and indexing the result of a game *as played*, relative to a player, as opposed to gameplay segmentation, that is a more absolute description of a game structure and mechanisms.

Chapter 3 then focuses on introducing the notion of *segmentation*, which is *the* process of dividing documents into coherent segments, and the notion of *indexing*, the process of locating and labelling the whole or part of a document, in the general context of document retrieval. Chapter 3 proposes to present the historical debates and standard notions of segmentation and indexing in order to better understand what these concepts entail, and then assess how they can be efficiently adapted for gameplay analysis. One core debate is for instance about distinguishing between open and closed system indexing (Klement, 2002) differentiating whether a document should be considered as a whole and described as such (open system), or should be considered as a collection of sub-segments and sub-topics, each of them needing to be independently described (closed system). Another crucial debate is about distinguishing between *controlled-vocabulary indexing*, meaning that the segment labels are based on a set of words

pre-elicited by the researcher, or *free-text indexing*, meaning that the labels are based on the direct document content, and are using words present in the document (Fidel, 1991). This distinction is important as it determines the way feedback streams can be processed: by locating an element of interest for the analyst (*controlled-vocabulary*, further developed in Chapter 5) or by letting the content of the stream lead the segmentation (*free-text*, further developed in Section 6.2).

Similarly for segmentation, different methods to determine the coherence of a segment are debated in Chapter 3. These methods range from detecting topic changes, to assessing the homogeneity of a segment, through detecting one element in a segment that would give sense to the whole segment by resolving all the references and removing any ambiguity.

Each of the core notions presented in Chapter 3 is illustrated by game examples extracted from the game sessions described in Chapter 2, in order to illustrate how each complex notion of segmentation and indexing is applicable to videogame analysis and player experience assessment. Core research about audio-visual segmentation and indexing from computer science is also presented in order to highlight the link between feedback-based gameplay metrics and gameplay performance segmentation, and to demonstrate that automatisation is possible, as further developed in Chapter 5.

1.5.3 *Chapter 4 – A Multi-layered Architecture for Deconstructing and Reconstruct Gameplay (model)*

Review
- Multi-layered approach in gameplay segmentation
- Hierarchy in gameplay segmentation

Model
- 5-layered segmentation model
  - game system
  - game world instance
  - spatial-temporal
  - degree of freedom
  - interaction
- Structural hierarchical deconstruction
- Experiential transversal reconstruction

Application
- Full gameplay segmentation process using *Bioshock 2* as a case study

If isolated videogame-based examples are provided for the concepts of segmentation and indexing in Chapter 3, it is necessary to demonstrate how, in practice, gameplay performance segmentation would be effectuated to extract player experience from recorder gameplay footage. Chapter 4 is designed to build a model of segmentation and indexing adapted for game studies, and to highlight its application, using the game *Bioshock 2* (2K Games, 2010) as a case study.

Based on the observation that a gameplay performance segmentation model based on only one gameplay consideration would be an incomplete process, Chapter 4 seeks to demonstrate that using a *multi-layered model* can account more efficiently for the diversity of experiences of play. Five layers constitute the model: the game system layer, the game world instance layer, the spatial temporal layer, the degree of freedom layer, and the interaction layer. Chapter 4 demonstrates that the five layers can be used hierarchically for deconstructing a performance (each top-layer contextualizing the segmentation of each sub-layer),

and can then be used transversally during a reconstruction process based on player experience analysis.

Finally, based on examples derived from the game Bioshock 2, Chapter 4 illustrates how gameplay performance segmentation is executed, from a game session to the production of a gameplay performance summary.

1.5.4 *Chapter 5 – Automatic gameplay performance deconstruction (method)*

Definition
- Controlled-vocabulary automatic segmentation

Method
- Static logo detection
- Moving logo detection
- Colour ratio
- Scene change detection
- Sound correlation
- Validity, genericity and discussion

Application
- Examples of feedback-based gameplay metric results

Once the multi-layered model of gameplay performance segmentation has been introduced in Chapter 4, it becomes possible to think about methods for automatizing the deconstruction process via audio-visual analysis. Automatizing gameplay performance segmentation allows consideration of large proportions of performances, as well as detailed summaries of players' interactions with games to be automatically produced.

Chapter 5 is the central methodology chapter of the present thesis. In Chapter 5, algorithms – originated and adapted from computer science and the audio-visual segmentation research area – are used to automatically process the deconstruction part of the gameplay segmentation model and offer quantitative data resulting

from audio and video consideration of the game. The specificity of the algorithms presented in Chapter 5 is that they rely on *controlled-vocabulary* indexing, meaning that the researcher needs to pre-determine the gameplay concepts of interest (death, menu, fighting sequences, and so on) and also provides information about their audio-visual representation (logo, sound, colour, etc.). The different algorithms are also linked to each layer introduced in Chapter 4.

Chapter 5 presents five algorithms – adapted from existing methods in TV-stream, movie and radio-stream processing – that can detect audio-visual content, such as: static logo (Santos & Kim, 2006) (e.g., menu, interaction feedback); moving logo (Pratt, 1978, p. 553) (including enemy location, moving information panel); colour of an area (Gonzales & Woods, 2007, Chapter 2) (such as life bar, stamina bar); scene change (Huang & Liao, 2001) (e.g., menu page change, etc.); and sound similarities (Roads, 1996, p. 509; Yarlagadda, 2010, Chapter 2) (death, warning, etc.). For each method, a complete description is provided, along with the algorithm written in pseudo-code, examples of use and a discussion of validity.

1.5.5 *Chapter 6 – Interpreting play (method and results)*

Application
- Full gameplay performance summaries
- Experiential reconstruction based on
    - Contextualisation
    - Temporal frame of reference
    - Semantic network
    - Loop
    - Comparison

Method
- Free-text algorithms
    - Music recognition
    - Speech detection
    - Similarity matrices

Chapter 5 presents algorithms that deconstruct a performance based on the five layers presented in Chapter 4. Chapter 6 then displays and comments on summaries resulting from performance deconstruction works, and illustrates how the reconstruction process, based on the very same layers, using four games; *Bioshock 2* (2K Games, 2010), *Battlefield 3* (Electronic Arts, 2011), *Dead Island* (Deep Silver, 2011), and *Max Payne 3* (Rockstar Games, 2012). Four games are used in order to assess the genericity of the methods. Numerous gameplay performance summaries are presented in Chapter 6, along with annotations representing different modes of transversal reconstruction: contextualisation, temporal frame of references, semantic network, loop and comparison.

If the algorithms presented in Chapter 5 are essentially controlled-vocabulary (that is, they depend upon concepts pre-determined by the analyst), Chapter 6 also presents other results derived from a consideration of the direct game content. These methods and results, defined as *free-text*, are mostly useful for comparing performances, identifying key similar moments between two participants, or

repetition from within a single performance. Free-text methods are not the central point of the present thesis, but are presented with their results, as they can improve an understanding of player experience. Free-text approaches would, however, require further research.

1.5.6 *Chapter 7 – Implementing Gameplay Performance Segmentation alongside Other Measures (application)*

Application/Software System
- Feedback-based gameplay metrics software system
- Synchronised presentation
- Use of slow motion in Max Payne 3
- Biometry and feedback-based gameplay metrics correlation

Once gameplay performance segmentation (Chapters 3 and 4) has been presented in terms of controlled-vocabulary deconstruction (Chapter 5), experiential reconstruction and free-text comparison (Chapter 6), it is necessary to replace the results into the larger context of player experience research in the game studies community. Indeed, Chapters 5 and 6 demonstrated that, alone, the gameplay performance segmentation model (Chapters 3 and 4) based on feedback-based gameplay metrics (Chapter 2), can provide insight into player experience. However, numerous research works about player experience are based on multi-modality analysis, to offer a more exhaustive account of player experience. Chapter 7 illustrates how gameplay performance segmentation results can be included in multi-modality projects to offer a more exhaustive understanding of player experience.

Chapter 7 first presents the software system developed during the research (in terms of structure, upgradability and usability) in order to be able to assess the validity of different algorithms presented in Chapter 5 and 6. Then, Chapter 7 illustrates how the gameplay performance segmentation results can be used conjointly with other modalities, using three applications. The first one is a synchronised presentation, displaying into a single video the gameplay footage, feedback-based gameplay metrics, biometric measures and player facial capture, constructed in order to help the analyst identify key moments of play, and to extract meaningful questions to ask the participants during qualitative interviews. The second one is about analysing conjointly the use of the controller buttons with the detection of slow motion moment in the game *Max Payne 3*, in order to assess how the players engage with these supposedly violent sequences. The third one is about studying the eventual correlation between biometric and gameplay events, by first identifying key sequences of play that triggered a strong body reaction, and then assessing whether these moments are linked to moments identified by the feedback-based gameplay metrics.

1.5.7 *Chapter 8 - Conclusion*

The concluding chapter recapitulates the different contributions to knowledge, being the conceptual model consisting of the gameplay performance segmentation notion and the multi-layered model, as well as the methodology that seeks to automatically process audio-visual streams of gameplay footage through the use of feedback-based gameplay metrics. Both the conceptual model and the methodology are replaced in the context of the research trigger, being the need to assess player experience in relation with videogame classification, and the thesis rationale, arising from the way that gameplay metrics, while being efficient for

measuring the core constituents of a player performance, are not applicable to any off-the-shelf games. By describing the different functions that are offered by both the conceptual model and the novel methodology, the conclusion evaluates the impact of these contributions in game studies, computer science and in assessing classification system criteria. Finally, the conclusion illustrates that the conceptual model and the methodology have been designed to be generic enough, thus allowing analysts to tailor them in order to match the focus of future research into player experience.

Chapter 2

Gameplay metrics

Chapter 2 is dedicated to a review the literature about the use of *gameplay metrics* as a quantitative method to assess and understand *player experience*. By first discussing what the concept of player experience entails, and then acknowledging the strength of gameplay metrics as a method able to account for players' engagement with game systems, Chapter 2 presents the current uses of gameplay metrics in the *game studies* community, before discussing their limitations, which notably restrain the number of games that gameplay metrics can study. Chapter 2 then proposes a method derived from gameplay metrics, termed *feedback-based gameplay metrics*, and using the audio-visual feedback streams as a vector of gameplay information. This constitutes an alternative solution addressing several of the core limitations of the gameplay metrics approach.

Attempting to understand the specific experience that constitutes playing a videogame is a challenging task. For more than a decade now, the game studies community has proposed models and methods to attempt to pinpoint and assess the core aspects of *play*. Numerous approaches have been designed that not only account for the manner in which players actually engage with a game system (the *what*), but also for the rationale behind their actions and interactions (the *why*). A large range of works variously address how players engage in games. These works include more theoretically-oriented approaches, such as Gordon Calleja's (2011) work on player involvement which speculates as to what constitutes the main factors explaining why players continue to engage with game systems. The

literature also includes more methodologically-oriented approaches, such as studies that accurately trace and log the different interactions between the player and a game system (Drachen et al., 2013; Kim et al., 2008). Some approaches blend theory and method, such as the analysis of how flow theory (Csikszentmihaly, 1990) might be translated to game systems (Nacke & Lindley, 2008).

Within the wider context of game studies, the different works that seek to understand the nature of player engagement with game systems generally consider that the aim of their work is to address *play*. However, the concept of *player experience* remains vague, with no real consensus or unified definition achieved to date. One possibility for defining player experience might begin by recalling the origin of player experience as a research concept. Derived from *user experience* research as a component of Human-Computer Interaction (HCI), the focus has been on how a user interacts with a computer system. User experience is a term that is specified by the International Organization for Standardization (ISO) system (ISO, 2008) and represents the "perceptions and responses resulting from the use and/or anticipated use of a product, system or service" (2008, sec. 2). The ISO system also acknowledges the existence of three time frames of engagement that need to be taken into account while assessing user experience, the "before, during, and after" (2008, sec. 2), as well as the existence of three main factors believed to influence experience, "the interactive system, the user's state and the context of use" (2008, sec. 2).

It is not uncommon to mistake user experience with *usability*. However, distinguishing between the two terms is a way of understanding what user experience, and therefore what player experience, entails. The difference is indeed an essential one, as usability pertains not to the manner in which a user experiences a system, but in assessing the potential for a system to be used in accordance with the designers' aims, regardless of the users' "emotions, beliefs, preferences, perceptions" (ISO, 2008, sec. 2). Marc Hassenzahl (2013) provides a useful example to help distinguish between usability and user experience. A buzzing alarm clock and an alarm clock, recreating the light atmosphere of a sunrise both need to fulfil the goal of waking up a user at a given time. Assessing usability is more about ensuring that the user is awake at a specified time. However, the experience of being suddenly jolted awake by a loud noise is different from being gradually awakened by a light that imitates a natural sunrise.

The distinction between *usability* and *user experience* is similar to the difference between *player experience* and *playability* (Nacke, Drachen, et al., 2009). While playability assesses whether the use of a game conforms to its intended use, player experience deals more with players' emotions and reactions, and the rationale behind their actions.

In 2009, Nacke adapts the user experience definition in a *player experience model* (2009)[6], having reviewed existing literature on player experience. In doing so, Nacke classified the different methods that seek to assess player experience, mainly on *the player* (*user* in the *user experience* definition), the *game system* (*interactive system* in the user experience definition) and the *context of play* (*context of use* in the user experience definition). Similar to the user experience definition, Nacke also acknowledges that player experience is not only constituted by the *actual* moment of play, but also by the *previous* experiences (*before*) and the *consequences* of the play (after).

For the game system, Nacke presents different methods for correcting and improving the game during the development phase (for example, bug-tracking or beta-testing). He also refers to gameplay metrics, which represent quantitative data about the game state and the player's interaction with the game system. Such data is time-stamped, and directly logged by the game system itself (in general, it is included in the game source code, or in a modding system).

[6] The Artificial Intelligence (AI) community is also interested in the understanding of player experience. Mostly found under the denomination of *player modeling* (Yannakakis, Spronck, Loiacono, & André, 2013) (see also Section 2.1.5), this strand of research focusses on understanding players in order to create a computer and mathematical model that can be used within the game system to improve the experience of play. However, player modeling sits outside the scope of the present thesis, as player modeling is mainly a contribution taking place during the game development process. The present thesis on the other hand, as well as Nacke (2009) contribution, seeks to bring computer sciences outcomes into the game studies community, and consider the game after the release date as a finished product. That being said, player modeling research can be useful to consider for future work, notably as a way to automatize the interpretation process, performed manually in the present thesis.

To assess the player's contribution, Nacke focuses on presenting psycho-physiological methods, which represent numeric data about body signals from the player, such as heart rate, and galvanic skin response (moisture level of the skin). However, Nacke mentions again the key role of gameplay metrics for assessing the player component of the player experience model, as gameplay metrics not only inform the researcher about the player's actual actions, but they can also be used in studies about player behaviour and play style (Drachen & Canossa, 2008).

To understand the context, Nacke proposes a focus on external elements that can influence the play experience. This could be the presence or absence of an observer, or player interactions with other players in collaborative play (Schott & Kambouri, 2003). Most of the methods assessing context are typically qualitative, but once again, gameplay metrics are acknowledged as being able to quantify part of the context component of the player experience model. This is the case, for instance, of gameplay metrics linked to the detection of all the multiplayer interactions in a game (McEwan, Gutwin, Mandryk, & Nacke, 2012).

It becomes evident, then, that gameplay metrics represents a versatile data form that is able to cover and assess the core components of player experience, including the player, the game system and the context. Chapter 2 focuses on the literature about gameplay metrics, and describes how gameplay metrics have been used so far in the game studies community (Section 2.1). Chapter 2 also questions the limitations of gameplay metrics, and debates whether their scope of applications can be widened by adapting them. Through a consideration of the variety of information presented to the player through the two main audio-visual

feedback streams, Chapter 2 finally demonstrates that a derived method can account for the complexity of player experience, while also proposing a new way to quantify the content of the audio-visual streams (Section 2.2). This post-processing method, termed feedback-based gameplay metrics, can not only be used to process any off-the-shelf game, regardless of the source code level of access, but can also be considered by analysts interested in quantifying the exact way in which information is conveyed to the player (such as, for instance, assessing videogame classification criteria, as introduced in Chapter 1). Finally, Chapter 2 also describes the game sessions that have been organised at the University of Waikato, in order to test the validity of this new method using concrete examples (Section 2.3).

2.1 Gameplay Metrics

As outlined in the introductory part of the present chapter, *gameplay metrics* represents the only method to date – which is acknowledged in the *player experience model* – (Nacke, 2009) of linking the *game*, the *player* and the *context* components of the model. However, it would be misleading to consider gameplay metrics an exhaustive method that covers all possible questions arising from player experiences with digital games. For instance, for the context component of Nacke's player experience model, gameplay metrics are able to account for player interaction, but unable to account for the social background or the culture in which the videogame is being played. That being said, the present section demonstrates that, when used to assess the link between the player and the game system, gameplay metrics represent a method of choice that has already been used extensively in order to solve numerous player experience-related problems. These issues include assessing the relevance of game design choices (Section 2.1.4), adapting game content to improve the player experience of the game (Section 2.1.5), and understanding players' behaviour through the automatic determination of play style and play persona (Section 2.1.6). Gameplay metrics are also considered ready to be applied to a wider range of problem-solving areas for further research projects (Nacke, Drachen, et al., 2009). The present section therefore focuses on presenting gameplay metrics more extensively, to assess its advantages and limitations.

2.1.1 *Origin*

Trying to assess the usability and readability of user interfaces is a key topic in Computer-Human Interaction (CHI) research. Automatically logged data,

resulting from user interactions with a computer system, has been used for decades for assessing the consistency between the intended use of a software system, and its actual use by an end-user. In 1983, IBM (Neal & Simons, 1984) presented a program called Playback, which records the different keyboard actions (key strokes) performed by a user while interacting with determined software systems. Several special keys were also added, allowing the user to ask for help or to signal the accomplishment of a task. After the recording session, these data are analysed as a summary of the system's usability, but is also replayed so that observers can see and comment (step-by-step) on the previously performed actions, and identify the moments and elements of software design that could be improved in later versions of the software, to facilitate case of use. In 1993, Microsoft (Hoiem & Sullivan, 1994) proposed a system called Observer. Starting from the observation that key stroke recordings are too raw, not specific enough and redundant in videotape filming when used for playback features, they developed a system the main feature of which was an *Event Logging* method. Instead of logging the user interaction by recording the keyboard and mouse input, this method – for the first time – recorded information directly enclosed in the software system, such as menu selections, the movement of windows on screen, or dialogue box actions. The produced summary was then synchronised with a video recording by using a system of matching time codes, for visualisation and analysis.

Gameplay metrics – being quantitative data about the game state and the player interaction with the game system – represent an adaptation of these early methods for the study of games. Further examples of these early HCI examples can be

found in Hilbert & Redmiles (2000). They present different issues from the study of user interface, which can also be encountered during gameplay metrics analysis: (1) data searching and synchronisation; (2) data transformation; (3) summary statistics; (4) sequence comparison; (5) sequence characterisation; and (6) visualisation.

2.1.2 *Gameplay metrics terminology*

Because gameplay metrics are adaptations of the HCI methods described above for the study of videogames; and because, while being a growing topic of interest, academic publications about gameplay metrics are still scarce, very little has been done to try to refine the terminology and definitions around the gameplay metrics concept. While gameplay metrics is a regularly used term, several variations can be found within the literature, for example *user-interaction metrics* (Drachen, 2008), *gameplay features* (Pedersen et al., 2010), *entertainment metrics* (Togelius, Nardi, & Lucas, 2007), and the general HCI term, *user initiated events* (Kim et al., 2008). However, Drachen et al. (2013) propose a full terminology to define the different metrics linked to videogames, whether the metrics measure the user interaction with the game system, or provide more general information about games (e.g., development process or marketing). The present section is essentially based on Drachen et al. (2013) works.

According to Drachen et al. (2013), the terminology of gameplay metrics represents a subset of *user metrics*, itself being a subset of *game metrics*. Game metrics is an umbrella term defined as "a quantitative measure of one or more attributes of one or more objects that operate in the context of games" (Drachen et al., 2013). This definition, then, considers expanded quantitative measures in

regard to games, such as the number of sold videogame units, or even the completion time of tasks during the development process. Game metrics is then subdivided into three categories, in accordance with Mellon (2009): *performance metrics*, relative to the technical performance of the game (for example, frame rate, stability of a server); *process metrics*, relative to development process (which includes iteration rate and failure rate); and *user metrics*, which is relative to the users, considered either as customers or players.

User metrics has different levels of applicability (*generic*, *genre specific* or *game specific*), and can be again subdivided into three categories: *customer metrics*, relative to the player when considered as a customer (e.g., game downloaded, virtual items bought in-game, demographic information); *community metrics*, relative to players' interactions with each other, either in-game (which can then be also considered as *gameplay metrics*), or out-game (e.g., forum activity, live conversation); and *gameplay metrics*, relative to player behaviour inside the game (e.g., running, shooting, current health level) (Drachen et al., 2013).

Gameplay metrics is essential to assessing player experience and is defined as

> measures of player behaviour, e.g. navigation, item – and ability use, jumping, trading, running and whatever else players actually do inside the virtual environment. [Several] types of information can be logged whenever a player does something – or is exposed to something – in a game: What is happening? Where is it happening? At what time is it happening? In addition, when

> multiple objects (e.g., players) interact: to whom is it happening? (Drachen et al., 2013).

Gameplay metrics is the most commonly used sub-set of game metrics in the game study community, as they give an analysable summary of player interaction and behaviour within a game system. In contrast, the other *game metrics* are mostly meaningful for the game development company, in terms of efficiency assessment or for marketing purposes.

2.1.3 *Gathering gameplay metrics*

With the exception of keystrokes, which can be recorded using any type of key logger software system, gathering gameplay metrics requires co-operation from the game developers, by: (1) offering access to the game source code, through agreements, or including a metric system inside the game (Drachen & Canossa, 2009b; Kim et al., 2008); (2) allowing the game to be modded by providing a modding system. (Sasse, 2008); or (3) by releasing an open source version of their game (Pedersen et al., 2010). Under these conditions it becomes possible to select and record the different gameplay metrics of interest for summarising the player's interactions with the game system.

Gameplay metrics can either be continuous and recorded at a specific sample rate (e.g., life level, avatar position, car speed), or discrete/triggered and recorded when the event occurs (e.g., weapon change, interaction with NPC, damage inflicted) (Drachen & Canossa, 2008). Gameplay metrics, when they can be gathered, are valuable for game research because they can be logged in a detailed, precise and unobtrusive manner (Canossa & Drachen, 2009a). However, the

gathering method, based on the game source code, makes these valuable data restrained to a narrow number of game systems. The following sections provide examples of how gameplay metrics are used for assessing player experience, and also discuss some of their core limitations.

2.1.4 *Game design improvement*

Because gameplay metrics require a source code access or adaptation in order to be identified and logged, the most common applications of this quantitative data is found within game development, where the game source code is available, in addition to regular software and game testing methods (e.g., unit-testing, bug-tracking, beta-testing) (Nacke, 2009, p. 48). Designed to test the technical quality of videogames, game companies start to consider the usefulness of gameplay metrics as a design improvement tool, even if the other categories of game metrics, linked to business and development areas, are still the most highly prioritised (Drachen et al., 2013). The different systems presented in the present section use gameplay metrics as quantitative data that can enhance the successive game testing stages in order to improve design by finding weak spots in game design or by ensuring that the game is well balanced.

Microsoft has, for instance, created a tool called Tracking Real-Time User Experience (TRUE) (Kim et al., 2008), used for improving the design of dozens of internally-developed games during the production, beta test and even post-release phases. Based on gameplay metrics analysis (called *User Initiated Events* in this publication), but also attitudinal data, Microsoft has available a system capable of answering questions about design efficiency. In the game *Halo 2* (Microsoft Game Studios, 2004), TRUE has been used to detect unintended

increases in difficulty, consecutive to level design modifications. The metrics recorded during a play session focused on player deaths, death times, who killed them and how. By analysing the number of deaths per missions, it was possible to highlight how one particular mission was proving highly difficult for players. Furthermore, by analysing which enemies were killing most of the players, it became clear that the Artificial Intelligence (AI) on one single enemy needed to be adapted, as it was allowing the deaths of most of the players. The designers were then able to identify this enemy, and study its behaviour in order to correct the level of difficulty. Another session was then organised, with new participants engaging with the corrected design, to ensure the validity of the modifications.

For the game *Shadowrun*, TRUE was used for a different purpose: to evaluate whether or not a game was well-balanced regarding the character class selection. Over a period of 30 days (open beta testing), and using telemetric data from 10,000 players, the designers were able to notice that, during the first week, most players decided to play with *human* characters and not really with *elf* ones. But after ten days, most of the players decided to choose the *elf* class. The designers then discovered that this class was more likely to win, even though the desired goal of the designers was to provide well-balanced character classes, each with strengths and weaknesses to make them reasonably equal in terms of efficiency.

Other systems should be mentioned too, such as the one made by *Bioware* to study the game *Mass Effect* (Electronic Arts, 2008), based on the use of *time spent reports* (including the player's time spent in character creation, levelling up, inventory, journal, map screen, combat mode, exploration mode, dialogue mode,

store mode, prerendered cinematic, cut scenes, required minigames, and optional minigames) (Derosa, 2007). Time spent reports have been used for a variety of purposes. In one study, they were used to gain a clearer picture of RPG experiences (Derosa, 2007). In another one, they were used as post-released analysis of a free online strategy game (Gagné, Seif El-Nasr, & Shaw, 2012) (e.g., level information, movement information, death information, score information) in order to assess various questions, notably about the motivation behind free-playing (e.g., "at what point do players stop playing the game?"). *Time spent reports* were also used to analyse Massively Multiplayer Online games (Mellon, 2009) (e.g., popular levels, objects most frequently purchased, time to level up) in order to understand player behaviour and adapt the game for making it more fun.

Finally, some research focuses on the spatial behaviour in-game, combining the use of gameplay metrics for design improvement with visualisation ideas aimed to simplify the analysis process (Drachen & Canossa, 2009a, 2009b, 2011). Using the in-house EIDOS metrics suite system, the gameplay metrics (including, deaths, causes of death, player role at death, and navigation) are recorded, along with the precise coordinate positions of the avatar at the time of the metric log. Then, by using a second software system called *ArcGIS* – which offers a visualisation feature – the position information is linked on the matching level map, pinpointing the data and colourising them in relation to the metric type. The system was used on three games, for different purposes. In *Fragile Alliance* (Eidos Interactive, 2007), it was used to plot the location of player deaths and causes, in order to test the intended design of the game level. For *Kane & Lynch* (Eidos Interactive, 2007), it was used to plot navigation, health, crouch and cover,

and speed data, in order to assess what happens if the player is deviating from the "perfect path" as imagined by the game designers. In *Tomb Raider: Underworld* (Eidos Interactive, 2008) it plots the location of player deaths and causes in order to adapt the levels of challenge.

The use of gameplay metrics, as discussed in the current section, is clearly directed toward evaluation of the game system. If the players are part of the process, they are only involved to validate, or invalidate, pre-determined design decisions. Their experiences of play do not impact the game system beyond assessment of the designers' decisions. This is not about how the player engages with the game system, but whether or not the player engages in the intended manner. This use of gameplay metrics is actually closer to the notion of playability than the notion of player experience.

2.1.5 *Player modeling and game content adaptation*

The different approaches presented above serve mainly for validating and correcting game designers' decisions while creating levels in games. However, gameplay metrics can also be used in the context of Artificial Intelligence (AI) for *player modeling*: in order to create human-like behaviour (Thurau, Bauckhage, & Sagerer, 2003, 2004); to adapt enemy behaviour to player behaviour (Booth, 2009; Houlette, 2004); to support automatic generation of personalised content for videogames (Pedersen et al., 2010; Shaker, Yannakakis, & Togelius, 2010; Togelius et al., 2007; Yannakakis & Hallam, 2009); or, in a more exploratory approach, to try to identify different types of players based on several behavioural traits (Drachen, Canossa, & Yannakakis, 2009). This last example is discussed more thoroughly in Section 2.1.6.

As illustrated by the variety of applications briefly described above, player modeling is a broad concept. A discussion of this term, the lack of taxonomy attached to it, and its different applications, may be read in Smith et al. (2011). However, when linked to gameplay metrics, player modeling tends to follow an analogous approach, which can be defined as the creation of a "predictive model of player actions resulting from machine learning" (Smith et al., 2011). For that purpose, mathematical models based on the Artificial Neural Network domain are commonly used and fed with gameplay metrics resulting from one or several players' engagement with the game.

The research of Thurau et al. (2003, 2004) is directed toward analysing player behaviour, in order to create a model of AI aimed at replicating human-like behaviours. The game *Quake II* (Activision, 1997) has been used as a test-bed because of the free availability of its source code, and as numerous records (called "demos") of gameplay can be easily found in the player community. These records contain precious *gameplay metrics* information. Movement metrics (for example the player's position, view angle and velocity) – combined with more strategically oriented metrics (e.g., the player's inventory, current armour, health value) from a large data set of demos – are used to determine the identically encountered situations that will lead to similar actions or movement patterns. Then, the model can be applied to the enemies in order to adapt their movement behaviours; rather than being limited to the shortest path, they may be given a more strategically appealing one (e.g., remaining in darker areas).

Opponent behaviours can be adapted in real time, in relation to player actions. Indeed, Valve (Booth, 2009) created a system for the game *Left 4 Dead* (Valve Corporation, 2008), in which the "*emotional intensity*" of the player avatar is analysed with the help of several gameplay metrics, representing the damage sustained by the player's avatar, the degree to which the avatar is incapacitated, whether the avatar has been pushed off a ledge, and the distance between the avatar and a dying enemy. In conjunction with this "emotional intensity", a population of enemy is procedurally generated, in order to create and maintain a *dramatic game pacing*. Dramatic game pacing is defined as "Algorithmically adjusting game pacing on the fly to maximize drama (player excitement/intensity)" (Booth, 2009, p. 78). The concept of emotional intensity, and the measurement of game pacing, can also be found in studies that are linked with game design improvement (see Section 2.1.4); for instance, measuring cooperative gameplay pacing in massively multiplayer online games (Ashton & Verbrugge, 2011). As an alternative approach to the one used in *Left 4 Dead*, Houlette (2004) proposes to adapt directly the internal AI of the opponents instead of the number and pace of enemy strikes, to create a personalized and challenging behaviour.

Tailoring the players' experience can go beyond adapting the AI of opponents, with the re-design of whole levels to match the player style. In order to assist game designers in the time-consuming task of level design (Pedersen et al., 2010; Togelius et al., 2007), or to offer players unlimited amount of automatically generated levels matching their play style (Shaker et al., 2010; Togelius et al., 2007), researchers have been able to analyse how players actually perform in

several basic levels, by recording gameplay metrics during play sessions, and then automatically generating (by adapting the game source code) new levels tailored to the player style.

One example of game adaptation has been based on a car racing game (Togelius et al., 2007). The researchers first designed a test track with a topography representative of various racing challenges (e.g., straight lines, smooth curves, sharp curves). The track also contained several invisible waypoints, which served to ensure that the player drives on the entire track before crossing the finish line. During play, for each crossed waypoint, two data sets were recorded: the speed of the car, and the car position. By combining this data with the matching section of the test track, it became possible to identify which parts were easier or harder for the player. The summary was then processed and analysed to automatically design personalised tracks meant to enhance the fun of play.

Finally, in another example using *Infinite Mario Bros* (Markus Persson, 2006), an open source clone version of *Super Mario Bros* (Nintendo, 1985) was specially adapted (Pedersen et al., 2010; Shaker et al., 2010; Yannakakis, 2012). The system was designed to analyse a player's actions throughout one level, in order to generate the content of the next level (e.g., number, width and position of gaps, left-to-right-direction or multi-direction level) in line with the results. The gathered metrics were classified for their capacity to represent three emotional states: (1) fun, e.g., time spent while the avatar had super-powers (bigger Mario), time spent running, number of enemy "killed" (unleashed shell); (2) frustration, e.g., total number of cannons fired in the level, number of times the avatar lost his

super-powers, number of jumps executed; and (3) challenge, e.g., time needed to complete the level, number of collected items, number of times the player fires. By comparing the actions performed during the first level with previously generated computational models built upon the same game (Pedersen et al., 2010), a new level is generated which is meant to increase the player's positive experience (of fun and challenge).

This use of gameplay metrics is interesting because, unlike the methods presented in Section 2.1.4, the data is not passively gathered merely for the purpose of design analysis, without any direct impact on the player. In the present section, the gathering and analysis of *gameplay metrics* has an active consequence, as the player can then immediately profit by progressing to levels designed to fit his own play-style. The player is not considered as a beta-tester, but as a decisive part of the design process, whose own experience while playing may be enhanced.

2.1.6 *Play-persona*

Another major contribution to game studies provided by gameplay metrics analysis, is the identification of the *play persona* through player behaviour, as defined by Canossa and Drachen (Canossa & Drachen, 2009a; Drachen & Canossa, 2008; Drachen et al., 2009). The premise underlying the concept has its origin in player modeling approaches (Section 2.1.5) as the mathematical models are able to account for different styles of play in terms of how they have been trained. In research based on the game *Tomb Raider: Underworld*, *gameplay metrics* – including causes of death, total number of deaths, completion times, and use of helps – have been gathered from a large population of players and then used as input for the mathematical models, in order to visualise and reveal several

distinct groups from the player population. The findings created four classes of players: (1) Veterans, a well-performing group; (2) Solvers, who die often, take time to complete the game, and do not ask for hints; (3) Pacifists, who are killed by active opponents; and (4) Runners, who die often but complete the game quickly. Based on this research, and using Alan Cooper *persona* modeling theory (A. Cooper, Reimann, & Cronin, 2007), Canossa and Drachen extended the determination of group of players to the notion of play-persona. Play-persona is defined as "clusters of preferential interaction (what) and navigation (where) attitudes, temporally expressed (when), that coalesce around different kinds of inscribed affordances in the artefacts provided by game designers" (Canossa & Drachen, 2009b).

The game *Hitman: Blood Money* (IO Interaction, 2007) has been used to illustrate the concept of play-persona (Drachen & Canossa, 2008). After defining the different player character elements (mechanics, physical behaviour and movement, location, psyche, goals, association, category, background, appearance) that can be built upon a subset of avatar features (e.g., powers, abilities, skills, physical properties for the mechanics element), they propose to determine which gameplay metrics can be linked with each feature. For that purpose, they first classify the different gatherable and relevant metrics into four categories. The first category is *navigation metrics*, including speed, direction, position, camera view. The second category is *interaction metrics*, which are subdivided between *objects vs. game world interaction* (e.g., picking item) and *entities vs. Non Player Character (NPC) interaction*, the latter also subdivided between *combat interaction* (e.g., firing a ranged weapon) and *passive interaction*

(e.g., talking). The third category is *narrative metrics*, including dialogue choice, or quest completion time. The fourth category is *interface metrics* (e.g., time in menu, navigation when levelling).

These four categories are also classified according to whether they are associated with the player characters (*character-bound metrics*), or related to the other NPC or the game world in general (*event metrics*). Then, the researchers create a link between these gameplay metrics and the different character elements. For instance, they link *mechanics* with "frequency and place of player character death, time spent holding different weapon types, frequency of acquiring and donning disguises", or *location* with "X,Y,Z coordinates tracked as a function of time, location of the character in the context of other characters or objects, [or] time spent in different types of security-labelled areas" (Drachen & Canossa, 2008). Finally, they can determine a *play-persona*, constructed upon the recognition of *play-style* and *play-mode*. *Play-mode* "refers to the behaviour of a player with respect to one or a few discrete metrics, within the same overall group or type of metrics" (Drachen & Canossa, 2008). The term *play-style* is used to designate "a set of composite play modes" that can be built upon several metric groups, such as *navigation metrics* plus *interaction metrics* (Drachen & Canossa, 2008). A *play-persona* can then be determined "when a player uses one or more play-styles consistently throughout the game play session" (Drachen & Canossa, 2008). The player can use several play-styles while adopting a play-persona. One example the researchers give is the *silent assassin* persona, which matches the use of the *stealth* and *brain* play-styles. The *stealth* play-style is recognisable by the detection of several play-modes (e.g., crouching, sneaking, close combat, using

shadows), which are linked to specific gameplay metrics (e.g., time spent walking in shadows and in closets; time spent sneaking, frequency of silent/close combat moves). It is interesting to note that this hierarchical characterisation into *play-mode*, *play-style*, and *play-persona* can also be linked with the hierarchical player modeling proposed by Houlette (2004), in the context of Artificial Intelligence adaptation and player modeling.

A use of *play-persona* for the study of games is also proposed by Canossa & Drachen (2009a). They propose to assess whether a linear game, such as *Tomb Raider: Underworld*, can still offer freedom in the form of the *play-personas* a player can choose to adopt, and then compare this degree of freedom with the one intended by the game designer. They offer definitions for two different elements of play-personas. The first one is *metaphors of player* which can be summarised as the designer's point of view and expectations; the second is *lenses for analysing player behaviour* which refers mainly to the actual analysis and detection of play-personas (Drachen & Canossa, 2008). Then, based on the player modeling research related to the same game (Drachen et al., 2009), the researchers can detect the similarities and differences between the designer point of view and the player's behaviour, and also evaluate the degree of freedom given to players to express their own play-personas.

The play-persona contribution is a major one because, while still closely bound to game design purposes, it focuses more on the player than the game system. The gameplay metrics, even if they are uniquely gathered from the game system, can be representative of meaningful information about the actual player experience.

The players are an integral part of the player experience assessment, as the mathematical models are trained using previous play-styles, meaning that the players lead the determination of personas.

2.1.7 *Discussion and limitations*

As highlighted above, gameplay metrics are insightful, and include the game system, the player and the context parts of the player experience model (Nacke, 2009), through the numerous methods and applications described in the present section. In addressing core questions linked to player experience – and because they are easy to combine with other quantitative data – gameplay metrics represent the preferred method for assessing player experience. For these reasons, the method is gaining increased interest, both by academics and in the world of game development.

However, the range of questions that may be answered by gameplay metrics in terms of player engagement with a game system is still debated in the game studies community. The introduction to Chapter 2 describes the importance of assessing not only the *what*, but also the *why* behind player interaction. That being said, the ability of gameplay metrics to answer the *why* question, thus addressing the rationale behind player engagement, has been challenged. During a panel presentation, Nacke et al. (2009) stated that the *why* question is not covered by *gameplay metrics*. For instance the question "What level do players like?" concerned, according to Gagne (2012), "with players' feelings, and such as cannot be answered with telemetry alone". This is indeed true for some aspects of the *why* question, when only one metric is analysed. But correctly correlated metrics can actually provide information about the reason underlying an action, and even

highlight a particular emotion. For instance, detecting a moment when the player changes the difficulty level from medium to easy following a series of deaths is a strong indicator of the player's frustration. Furthermore, gameplay metrics can also provide information about the different interaction possibilities offered to a player at any given time. Therefore, the *why* can also be extracted from gameplay metrics, as an answer related to the degree of possibilities offered by the games. Gameplay metrics can actually be seen as a method answering all of the essential questions about player experience: (1) *what* (what is actually happening?); (2) *where* (the location of the avatar in the game space); (3) *when* (referring to both the metric timestamp and the time within the game space); (4) *how* (that is, the player interaction), (5) *why* (the rationale behind an action); and (6) *who* (refers to the chosen avatar).

However, Section 2.1 has pinpointed several core limitations which restrain the scope of use of gameplay metrics to the improvement of game design. To begin with, the method of gathering (that is, through the game source code or modding) limits the type of games that can be analysed to the ones authorised by game companies, or to open-source games. More than a problem of unwillingness to collaborate – as more and more videogame companies are opened to academic collaborations – this is a problem of feasibility: some companies simply do not have time for such collaboration, or they cannot legally share their source code. The strong need for a collaboration in order to be able to gather gameplay metrics explains why, so far, most of the research about gameplay metrics has been about addressing the needs of videogame company.

Furthermore, it is sometime required for player experience to be assessed from the exact *player's* point of view; that is, in regard with what the player directly experiences through the videogame feedback streams (mostly pertaining to the video and sound streams). Then gameplay metrics both suffer from being not restrictive enough and too restrictive. Gameplay metrics are not restrictive enough in the sense that they can grab hidden-to-the-player information (what Drachen et al. call *system metrics* (2013)). This is the case with out-of-sight enemy positions, or any kind of avatar state in which player is unaware of. An example from *Battlefield 3* illustrates this point. The player does not have a life bar, and can die if he/she is hit too many times in a row, and can then recover by waiting; in this case the avatar life still exists as a metric as to how close the player is to death, but the player does not know this until he/she is nearly out of lives. The information exists in the game source code as the player activates the game, but is not a metric that can be considered as being part of the actual experience of the player. And *gameplay metrics* are too restrictive too, because the different feedback streams contain information that is not easily loggable through the sole source code processing. This is the case when assessing the sound or video parameters as they are broadcasted to the player, such as an overall frame colour or the sound frequency; i.e., the exact way information is communicated to the player.

2.2 Feedback-based gameplay metrics

The limitations presented above constrain the use of gameplay metrics, while gameplay metrics seems to be the method of choice in establishing an overall understanding of player experience. The key aim of Section 2.2 is to determine whether or not there is a way to circumvent the source-code/modding limitation of

gameplay metrics that would then open the use of gameplay metrics beyond game design improvement purposes. To that end, Section 2.2 is evaluating the content of the audio-visual feedback streams presented to the player during play, in order to assess these streams as an alternative vector of gameplay metrics, which would circumvent the need for a source code access and then open the use of gameplay metrics to any off-the-shelf games. To do that, several questions need to be carefully addressed:

1- Is it possible to find gameplay-metric-like information which is directly broadcasted to the player streams (i.e. sound and video streams), and what would be the level of detail of such information?

2- If yes, is it possible to automatically extract the information from the different streams, and quantify it in a time-stamped gameplay-metric-like fashion?

3- Finally, what would be the contribution of such a method in regard to the currently existing methodologies?

2.2.1 *Audio and video streams as vectors of gameplay information*

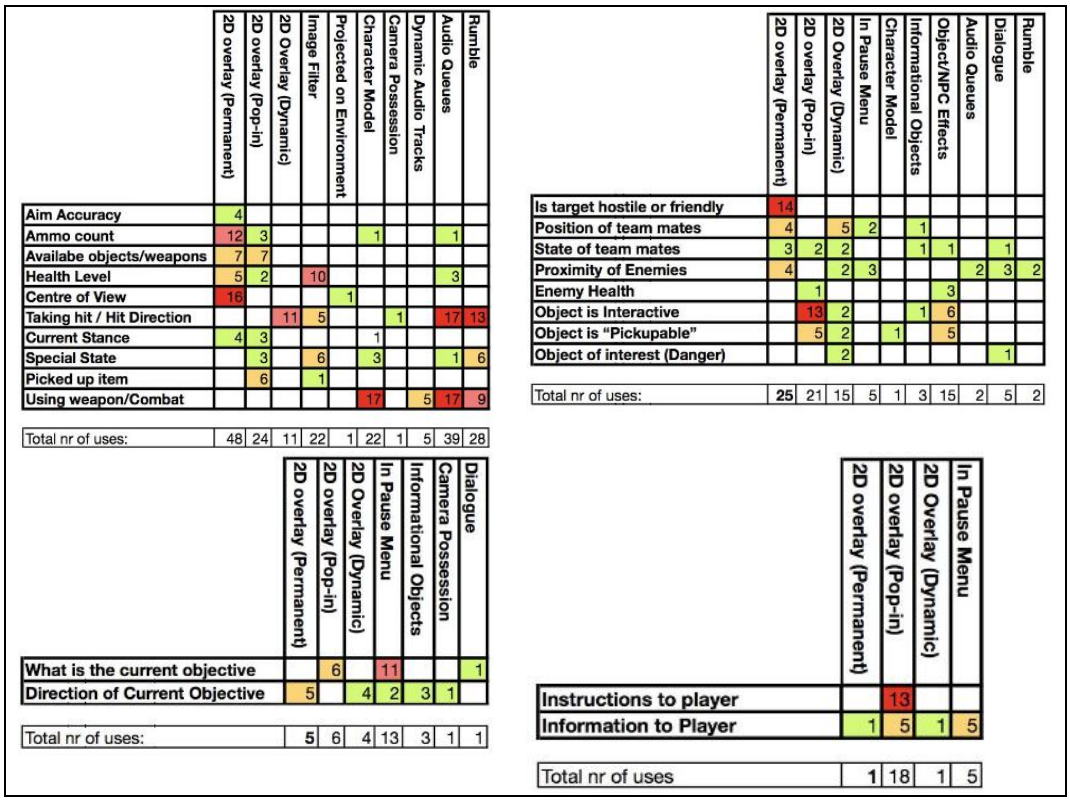
During play, the player is presented with two main feedback streams that inform him/her in real time about the game state: the video stream – through the computer monitor or the television set – and the audio stream through the speakers or headphones.[7] The audio-visual streams inform the players about the game-world, the different gameplay entities, and the current game state; in order

[7] Another feedback stream can also be the haptic feedback through the controller, but is more scarcely used, and will not be covered in the current thesis.

to allow them to interact efficiently; that is, the player can make the choice appropriate to achieving a desired goal. Several research works are focusing on analysing the gameplay elements communicated to the player through the audio-visual streams. For instance, numerous studies on Head Up Display (HUD) highlight the fact that the player (because he/she is not physically present in the game world), needs to be constantly informed about non audio-visual elements in regard to the avatar's physical state, such as loss of health and stamina drop (Ruch, 2010). Moreover, such symbolic information must be displayed if the player is to be sufficiently informed about the game; such elements include goal description, enemy health, or points scored (Fagerholt & Lorentzon, 2009). Fagerholt (2009), reviews 19 First Person Shooter (FPS) games and classifies the different symbolic and game-related elements enclosed inside the sound and video streams. The graph summarising his research (see Figure 6) highlights the importance of the audio-visual streams in conveying gameplay elements to the player. For instance, the visual stream features a 2D overlay for core information such as ammo count or centre of view; while the sound gives the player information about the fight or the health level.

Other research works about sound and video in videogames also point out the importance of diegetic material; that is elements of the game world, directed toward the player as well as other protagonists in the game (Grimshaw et al., 2008; Jørgensen, 2006). Diegetic material might include, for instance a gun shot, an item on the floor, a room design or a different character with whom the avatar can interact. Although they are not direct gameplay elements by being clearly represented as such, diegetic material is still linked to the player experience and

also represents gameplay elements (a gunshot is indicative of a fighting interaction that may also lead to points being earned, NPC interaction may lead to a new quest, and so on).



	2D overlay (Permanent)	2D overlay (Pop-in)	2D Overlay (Dynamic)	Image Filter	Projected on Environment	Character Model	Camera Possession	Dynamic Audio Tracks	Audio Queues	Rumble
Aim Accuracy	4									
Ammo count	12	3				1			1	
Availabe objects/weapons	7	7								
Health Level	5	2		10					3	
Centre of View	16				1					
Taking hit / Hit Direction			11	5			1		17	13
Current Stance	4	3				1				
Special State		3		6		3			1	6
Picked up item		6		1						
Using weapon/Combat						17		5	17	9
Total nr of uses:	48	24	11	22	1	22	1	5	39	28

	2D overlay (Permanent)	2D overlay (Pop-in)	2D Overlay (Dynamic)	In Pause Menu	Character Model	Informational Objects	Object/NPC Effects	Audio Queues	Dialogue	Rumble
Is target hostile or friendly	14									
Position of team mates	4		5	2		1				
State of team mates	3	2	2			1	1		1	
Proximity of Enemies	4		2	3				2	3	2
Enemy Health		1					3			
Object is Interactive		13	2			1	6			
Object is "Pickupable"		5	2		1		5			
Object of interest (Danger)			2						1	
Total nr of uses:	25	21	15	5	1	3	15	2	5	2

	2D overlay (Permanent)	2D overlay (Pop-in)	2D Overlay (Dynamic)	In Pause Menu	Informational Objects	Camera Possession	Dialogue
What is the current objective		6		11			1
Direction of Current Objective	5		4	2	3	1	
Total nr of uses:	5	6	4	13	3	1	1

	2D overlay (Permanent)	2D overlay (Pop-in)	2D Overlay (Dynamic)	In Pause Menu
Instructions to player		13		
Information to Player	1	5	1	5
Total nr of uses	1	18	1	5

Figure 6. Erik Fagerholt's summary of HUD elements using 19 games (2009)

Figure 6 is a set of tables generated by Erik Fagerholt (2009), summarising his research into the audio-visual content of HUD in FPS games.

The research works presented above not only validate the idea that gameplay elements are usually broadcasted to the player via audio-visual streams, but also that two levels of detail can be considered (cf. Figure 7). The first is a symbolic level, mostly containing non-diegetic gameplay-related elements. The second is a game world level, which contains mostly diegetic game world-related elements, and which can also be linked with gameplay elements via an abstraction process.

Figure 7. Symbolic and diegetic levels of feedback streams

Figure 7 illustrates the two main levels of analysis of the audio-visual feedback streams: a symbolic level, mostly containing non-diegetic gameplay-related elements, and a game world level, mostly containing diegetic game world-related elements, that can be also linked with gameplay elements via an abstraction process.

The question may be raised as to whether the two levels are the only ones needing to be considered when studying audio-visual streams. Actually, another level can be added, much more abstract than the two symbolic and game-world levels. This level focuses on how the streams are treated by computers as data structure. Indeed, video and audio streams may also be analysed in regard to their parameters, such as the frequency of a sound or the main colour of a video frame. Such an abstraction, which seems far away from gameplay consideration, can actually be insightful when assessing player experience. A black frame can be indicative of a cut in the game progression, while a red frame is likely to contain gore elements; a high pitched sound frequency can represent a warning signal. Furthermore, some research works link colours with emotion – both in the general context (Plutchik, 2001) and the videogame context (Joosten, Van Lankveld, & Spronck, 2010) – and colours are part of the parameter level. A new level of analysis can then be added (see Figure 8) when audio-visual streams are considered as a vector of gameplay elements.

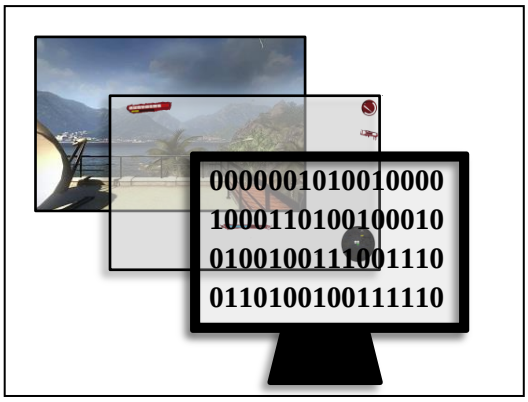


Figure 8. Symbolic, diegetic and audio-visual parameters levels of feedback streams

Figure 8 illustrates the existence of a level of analysis when considering audio-visual feedback streams. This level focuses on the audio-visual parameters, extracting information such as video main colour or sound frequency.

2.2.2 *Automatic extraction of audio-visual contents*

If it is clear that audio-visual streams are a vector of gameplay elements, the remaining question to address regards the possibility of gameplay elements being automatically extracted and turned into quantifiable and time-stamped data, in a process similar to that of gameplay metrics. To achieve this, automatic video and audio analysis methods must be considered. They represent a growing topic of interest in the computer science community about sound and image processing. In terms of symbolic video detection, numerous algorithms have been proposed for movie, TV-stream and camera footage, which detect symbolic information such as the presence or absence of TV-Channel logos (Santos & Kim, 2006); text recognition (Slik, Jongebloed, & van Staalduinen, 2013), and number recognition (Ye, Huang, Jiang, Liu, & Gao, 2006). In terms of diegetic detection, numerous algorithms exist to identify objects or characters in video streams (Bay, Ess, Tuytelaars, & van Gool, 2008). In terms of video parameters, algorithms also exist to recognise scene changes based on video colour distribution (Huang & Liao, 2001). Similarly, audio algorithms exist and are employed for radio streams and music tracks, such as in the recognition of sounds via cross-correlation (Roads, 1996, p. 509; Yarlagadda, 2010, Chapter 2), recognition of music via musical representation and genetic-like comparison algorithms (Robine, Hanna, Ferraro, & Allali, 2007), and the detection of speech sequences (Rabiner, 1989).

Finally, algorithms also exist to automatically compare video and audio streams, and to highlight repetitions and similarities (Martin, Hanna, Robine, & Ferraro, 2011). All of these algorithms, while proven highly effective and accurate, have yet to be employed for videogame footage analysis. For now, the area of

application is limited to more traditional media. Videogame footage, because of the diversity of audio-visual elements it conveys, would actually profit from the adaptation of these methods for player experience assessment. The audio and video analysis, and extraction of gameplay elements, could then provide discrete data – such as a presence or absence of a logo on screen – that is indicative of a performed action; or it could provide continuous data, such as the amount of one colour in a specific area, that can represent the filling level of a health bar.

2.2.3 *Feedback-based gameplay metrics*

Both Section 2.2.1 and Section 2.2.2 demonstrate not only that audio-visual streams convey extensive core gameplay information (symbolic, diegetic, and audio-visual parameters) and directly broadcast to the player, but also that such information can be detected by using existing audio and video processing algorithms which are currently limited to movie, TV-stream, camera footage, radio stream and music track processing. Section 2.2.1 and Section 2.2.2 outcomes lead to the possibility of considering a new player experience method based on gameplay metrics methodology.

The full method of processing audio and video streams (also known as feedback streams) in order to obtain time-stamped gameplay metrics-like information is called *feedback-based gameplay metrics* and is the most signification contribution in the present thesis.

Such a method differs from the other quantitative data currently in use, due to crucial fact that feedback-based gameplay metrics rely not less and not more on what is actually presented to the player. The game system is not viewed as a

black-box, but through the window of what the game is authorising the player to see about itself. Furthermore, focusing on feedback streams circumvents the source code availability restriction, as a recording of the gameplay footage, (accessible for any game), is enough. This opens the range of analysable videogames by the game studies community, and also broadens the area of analysis, so far mainly limited to the interests of game companies. Furthermore, a consideration of gameplay footage offers a post-processing method, which means that the metrics can be extracted again later if the researcher eventually needs to consider different metrics. Feedback-based gameplay metrics seeks to add a new dimension to the player experience assessment, both by proposing a new vector of analysis (focusing on the feedback streams) and also representing the exact way in which game information is conveyed to the player.

2.3 Game sessions

In order to test the validity of the feedback-based gameplay metrics approach in terms of feasibility, applicability and genericity, several game sessions were organised at the University of Waikato. During these sessions, participants – aged between 14 and 16 – played R16 games[8] for 45 minutes, over five weeks (each weekly session was a direct continuation of the previous one, playing the same game). It is important to note that the choice of using R16 games has not impacted on the methodology work presented in the current thesis, but the decision was made so that the outcomes of the sessions may be further used in current and

[8] Consent was obtained from the participants, their parents, and from the chief censor of New Zealand.

forthcoming works about player experience, *violence* in videogames, and *classification systems*. That being said, in the context of the current thesis, the games were chosen for their diversity in terms of sonic and graphical representation (first person, third person, realistic, fantasy, and so on) in order to assess, at the same time, the genericity of the feedback-based gameplay metrics approach. *Battlefield 3* (Electronic Arts, 2011) is a First Person Shooter game, with a realistic war background, and with a Head Up Display (HUD) full of dynamic (moving) information. *Bioshock 2* (2K Games, 2010) is a First Person Shooter game, in a fantasy world and atmosphere, and with a HUD full of static information. *Max Payne 3* (Rockstar Games, 2012) is a Third Person Shooter game, in a realistic world, and with a HUD full of transparent information. Finally, *Dead Island* (Deep Silver, 2011) (which is a R18 game, so it was used as a test game with an adult participant) is a First Person Shooter game from the zombie genre, with a HUD full of symbolic information based on a caricatured use of colours (the blood is *very* red for instance).

During the sessions, considerable amounts of data have been concurrently recorded and synchronised (more details about the synchronisation process are included in Chapter 7), from biometry (GSR in the present thesis) to gameplay footage, including keystrokes and player facial expressions. The feedback-based gameplay metrics are extracted from the different gameplay footages, and the methods of extraction are presented in Chapter 5. The other recorded data are useful to illustrate multi-modality player experience assessment, using feedback-based gameplay metrics in conjunction with other methods, both quantitative and qualitative, as developed in Chapter 7.

2.4 Conclusion

Chapter 2 highlights the rationale behind the need for a new quantitative method linked to player experience, and of deriving gameplay metrics from the audio and video streams. For that, Chapter 2 started by introducing the concept of user experience and its adaptation to player experience via the player experience model proposed by Lennart Nacke (2009).

Chapter 2 then focused on the fact that gameplay metrics is the only method to date to include the game system, the player and the context components of the player experience model. The gameplay metrics approach was explored, from its origin to its current applications. As powerful as it seems, gameplay metrics also suffers from strong limitations, in terms of gathering constraints which limit the number of games the game studies community can consider – and in terms of the core distinction between gameplay metrics and the actual feedback streams presented to the player. In one sense, gameplay metrics is not restrictive enough (able to represent information invisible to the player); in another sense, it is too restrictive (excluding some graphical and sound parameters that are important for assessing the player experience). This, plus the demonstration that gameplay metrics can be extracted from audio-visual feedback streams via three levels of details (symbolic, diegetic, and audio-visual parameters) using computer science algorithms, justified the need for a new method, introduced in Chapter 2 as feedback-based gameplay metrics. Finally, the chapter also presented studies conducted at the University of Waikato that were useful in assessing and validating the innovative feedback-based gameplay metrics approach.

Now that feedback-based gameplay metrics has been introduced and the rationale for using it has been justified, it is crucial to focus on how an analysis of feedback-based gameplay metrics can actually provide a key understanding of player experience. Chapter 3 will introduce the notion of gameplay performance segmentation, which aims to identify moments of play – coherent both in terms of game structure and player experience – through defining the concepts of segmentation and indexing as they are already applied in game studies, in computer sciences, and in the standard discipline of document retrieval.

Chapter 3

Gameplay performance segmentation

In Chapter 2, the potential of *gameplay metrics* as a comprehensive quantification of *player experience* was presented. Although gameplay metrics constitutes a well-established method that has been presented to the game studies community as a useful approach for the study of player experience, the review of literature stresses that the use of gameplay metrics has been predominantly limited to the study and improvement of game-design decisions (see especially Drachen et al. (2013), Kim et al. (2008) or Derosa (2007)). While assessing the relevance of game-design is an important validation process for the game industry, actors also interested in other game analysis issues – such as assessing the player experience of an off-the-shelf game – would profit from the opportunity to possess a quantitative method which assesses player experience in a broader context. This is notably the case for game study researchers, who are interested in understanding the general experience of play (see for instance the involvement model of Gordon Calleja (2011)), or even legal issues, such as videogame classification systems (cf. Chapter 1) where age-rating decisions, in general, occur after the development phase.

In order for a broader community to profit from the outcomes of quantitative research articulating player experience, Chapter 2 also demonstrated the need to rethink how a gameplay metrics approach could be opened up for use at any stage of the life cycle of a videogame. A novel notion, termed *feedback-based gameplay metrics* – differing from traditional gameplay metrics in that it is based

on automatic video and audio analysis of gameplay footage – was introduced. Feedback-based gameplay metrics aims to constitute an alternative solution to not only circumventing the limitations imposed by current gameplay metric approaches in terms of data gathering, but also allowing post-processing analysis of game sessions, while focussing on the exact way games convey information to the player through their main feedback streams.

The following chapters, in their turn, will clarify how feedback-based gameplay metrics can actually be employed and analysed for player experience assessment and understanding. Indeed, the main strength of feedback-based gameplay metrics is their ability to highlight, from the main feedback streams, moments of play that convey a specific player experience aspect, translated from the study of gameplay evolution over time. For instance, studying a health bar progression over time can distinguish several game experiences: health drop is likely to indicate a fighting sequence and an action-engaging sequence; an increase in health is likely to be indicative of a strategic sequence in which the player is trying to cure his avatar; a stable health is likely to indicate a more exploratory sequence; and a zero value is indicative of the avatar death, in general followed by a "retry" sequence. Each of these moments, coherent in terms of gameplay understanding, is defined as a *segment* in the present thesis.

The term *segment* has been consciously chosen because of its direct relationship with *segmentation*, a concept widely researched in the academic world, and established in a large number of disciplines, from librarian studies to computer science, and including game studies. Segmentation is defined as "[the process of]

dividing a lengthy document into topically coherent sections" (Reynar, 1998, p. 3), meaning that the structure of a segmented document becomes apparent for analytical purposes; the concept is closely related to indexing. Indeed, once a segment boundary has been determined, it needs to be labelled in order to be further understood by an analyst. This is the purpose of indexing: to find an insightful way of describing a segment of a document. *Indexing* is defined in the present thesis as the process of locating and describing all or part of a document. Segmentation and indexing are complementary and need to be performed together. Assessing the boundaries of a segment means having an idea of the descriptors of interest (for instance "fighting sequence", "exploratory sequence", "death screen", and so on), and labelling a segment means knowing the segment boundaries.

However, the nature of the document considered in the present thesis for segmentation and indexing differs from the one used by Zagal et al. in their *gameplay segmentation* definition (Zagal et al., 2008). For Zagal et al. (2008), the document being segmented is the game itself, through its gameplay elements. Each game possesses one segmentation, describing its mechanisms, space and time management. The identified segments are then chunks of gameplay proposed to the player to evolve in and interact with, regardless of any players' activations of the game. However, in the present thesis, the feedback-based gameplay metrics are representative, less of the game as an absolute entity, and more as a specific series of activations of the game performed by a specific player. That means that in the present thesis, gameplay has to be considered as the "game as play", a relative entity. Each game is then able to have several segmentation outcomes,

depending upon who is playing, and how. There is then a strong need to differentiate both *gameplay* documents in order to also differentiate both types of *segmentation*. In this chapter, the term *performance* – offered by Laurel (1993) – is used to specify that the gameplay document is the game as played, not an absolute consideration of game, and then acknowledges that *feedback-based gameplay metrics* are realising *gameplay performance segmentation* as opposed to *gameplay segmentation*.

The main goal of Chapter 3 is to consider the application of segmentation and indexing processes for gameplay performance analysis, as a solution for correctly identifying gameplay segments that are indicative of a specific experience of play. Applied to gameplay performance, the notions of segmentation and indexing not only provide a structure for conceptualising the different phases and media experience offered by games which could then be operationalised as feedback-based gameplay metrics; but also at a deeper level again determine the different player experience accounts that might arise from a gameplay-metrics analysis.

In order to fulfil this objective, Section 3.1 first introduces the notion of gameplay segmentation as proposed by Zagal et al. (2008), emphasising its usefulness and current applications. Then, the concept of gameplay performance segmentation is presented in Section 3.2, using the definition of performance proposed by Laurel (1993) in her attempt to link computers with theatre, and then compared with the gameplay segmentation concept of Zagal et al. The two concepts will be acknowledged as being complementary rather than distinct, as a good knowledge of the game mechanisms, effected through gameplay segmentation, is mandatory

to identify feedback-based gameplay metrics of interest and then to perform relevant gameplay performance segmentation. Then, indexing and segmentation, – as they have been originally defined, debated and specified in the librarian disciplines – are presented in Sections 3.3 to 3.5, in order to identify the standard notions attached to indexing and segmentation that need to be adapted for gameplay performance. Finally, as feedback-based gameplay metrics are based on audio and video analysis, Section 3.6 presents the existing takes on audio-visual segmentation and indexing in the academic world; and Section 3.7 outlines computer science approaches currently used to deconstruct multimedia documents in a process of automatic indexing and segmentation.

3.1 Gameplay segmentation

Chapter 3 focuses on the concept of *segmentation* as applied to the analysis of *feedback-based gameplay metrics* outcomes in order to identify gameplay segments that are coherent in terms of *player experience*. *Segmentation*, as applied to gameplay, is not a novel approach, and has been already proposed by Zagal et al. (2008). In Section 3.1, the concept of *gameplay segmentation* is presented in order to understand its main focus and rationale and justify, for the needs of the present thesis, the definition of a novel approach called *gameplay performance segmentation* (see Section 3.2).

Gameplay segmentation, as defined by Zagal et al., represents "the manner in which a game is broken down into smaller elements of gameplay" (2008, p. 176). The document of interest is the game, and the segmentation process seeks to explore the structure of gameplay and analyse the role of "design elements" (2008, p. 177). So, the idea of gameplay segmentation is to extract, from a game document, the different gameplay segments that organise the game mechanisms, for game analysis and games comparison purposes. Zagal et al. propose three different segmentation approaches: temporal segmentation, spatial segmentation and challenge segmentation.

By temporal segmentation, Zagal et al. (2008) refer to gameplay mechanisms based on time, such as *temporal coordination* and *temporal resource*. Temporal coordination "regulates the actions of a player in a game [and determines] how these actions occur over time" (2008, p. 179). Temporal segmentation is then, for instance, useful to describe how multiplayer games manage the different player

moments of play, such as “turn taking” games, “rounds” games, “interleaved” games (Zagal et al., 2008, p. 180). Temporal resource deals with the gameplay elements where “time is treated as a resource” (Zagal et al., 2008, p. 180), meaning play mechanisms where time is part of the rules. This would include for instance, the needed time to fulfil a goal, or games based on sub-periods, as in sports games.

Spatial segmentation results “from the division of the game world into different spaces that also partition gameplay” (Zagal et al., 2008, p. 182). The main segment provided by Zagal et al. is the level, defined as “a recognizable subspace of the game world” (2008, p. 182), which is recognisable by “a discontinuity in gameplay and space” (2008, p. 183). To assess discontinuity, several gameplay mechanisms can be considered, such as the impossibility of killing off screen enemies, or areas where avatar health is regenerated. If the level is mainly a discrete entity, Zagal et al. also propose more continuous spatial gameplay mechanisms, called “spatial checkpoints”, which “divide a space into sub-locations that follow one another continuously” (2008, p. 184). A spatial checkpoint has an impact on the gameplay by, for instance, indicating a respawn point.

Finally, the challenge segmentation refers to the point at which “the player [has to] resolve a series of discrete self-contained challenges that need to be solved before moving forward in the game” (Zagal et al., 2008, p. 187). These include, “waves”; which is a group of enemies that the player has to deal with, with pauses between waves; “puzzle” is a reflection-based challenge that needs to be solved

before moving forward in the game, “boss challenge” is a difficult challenge to overcome, representative of a milestone or ending of the game; and “bonus stage” is a moment when the players know they cannot lose but can only improve their score.

As already stated in the introductory section of this chapter, segmentation concept tied to a second concept, called indexing, that efficiently labels the different identified segments. The gameplay segmentation approach also shows some strong connections with indexing. Indeed, when proposing segmentation approaches, Zagal et al. (2008) also define a set of terms aiming to describe the different segments of gameplay, that is level, waves, boss, turn-based, checkpoint, time limit and so on. Furthermore, gameplay segmentation is acknowledged as part of the *Game Ontology Project* (Zagal et al., 2008, p. 175) which represents an organised list of words that seeks to propose a “framework for describing, analysing and studying games” (Zagal et al., 2005, p. 1), with the rationale that “designers lack a unified vocabulary for describing existing games” (Zagal et al., 2005, p. 1) and that designers would valorise an ontology that can describe gameplay elements. Figure 9 illustrates the determination of indexing terms through the game ontology. Figure 9 focuses on an excerpt from the game ontology about interface vocabulary, as this subset of words is particularly relevant in the context of the present thesis, centred to audio-visual feedbacks of games. Game interface is defined in the Game Ontology Project as “[providing] the means by which the player experiences the game and takes action within the game” (Zagal et al., 2005, p. 5). Figure 9 for instance shows the two main groups of feedback-based gameplay metrics, one of which is “confirmation” feedback

(describing a player action), and the second one "warning" feedback (describing the game state), as well as different generally observed components of the Head Up Display (HUD).

- Interface
  - Input
    - Input Devices
    - Input Method
  - Presentation
    - Cardinality Of Gameworld
      - 1-Dimensional Gameworld
      - 2-Dimensional Gameworld
      - 3-Dimensional Gameworld
    - Presentation Hardware
      - Audio Display Hardware
      - Haptic Display
      - Visual Display Hardware
    - Presentation Software
      - Feedback
        - Confirmation
        - Warning
      - Point of View
        - First-Person Point of View
        - Second-Person Point of View
        - Third-Person Point of View
      - Sensory Output
        - Aural Output
        - Tactile Output
        - Visual Output
          - Camera-based World View
            - …
          - Head Up Display
            - Buttonpress Indicator
            - Control Bindings Display
            - Health Indicator
            - Lives Indicator
            - Map Display
            - Next Piece Indicator
            - Player Configurable Buttons/Keys
            - Points Indicator
            - Radar Display
            - Special Weapon Indicator
            - Time Indicator
            - Vehicular Instrumentation

Figure 9. Excerpt from the Game Ontology, focusing on the interface

Figure 9 is an excerpt of the Game Ontology from the Game Ontology Project, which is linked to the gameplay segmentation notion. Here, the Interface sub-tree is considered, as it contains numerous elements that can be seen as quantifiable through the use of feedback-based gameplay metric.

Gameplay segmentation is linked to the Game Ontology Project, as each of the three segmentation approaches introduces gameplay terminology that can update the ontology. The idea of building a vocabulary to describe a document, or a segment of a document, is definitely in line with the concept of indexing, even if the term indexing is not mentioned in the two reference articles defining gameplay segmentation (Zagal et al., 2008) and the Game Ontology Project (Zagal et al., 2005). The presence of an indexing process through the Game Ontology Project illustrates how the two concepts of segmentation and indexing are interdependent.

Gameplay segmentation is a valuable approach to partition (segmentation) and label (indexing) the different gameplay elements, constituting a specific game and governing its mechanisms. However, the way a game is activated by a player has no influence over the segmentation result. This is why the present thesis, based on feedback-stream resulting from player activation of the game, proposes to define a novel approach, called gameplay performance segmentation.

3.2 Gameplay performance segmentation

As highlighted in Section 3.1, *Gameplay segmentation* considers the game as being the document to be segmented. That means that, depending on the chosen *gameplay segmentation* approach (temporal, spatial or challenge), a game will have one and only one *segmentation* result. Gameplay segmentation is a deconstruction of a game into chunks of gameplay, allowing an insightful structural description of the game content and mechanisms. However, assessing the *player experience* through *feedback-based gameplay metrics*, as presented in Chapter 2, forces to go beyond the game as an absolute entity, as the player experience is tied to the understanding of the different game activation choices made by the player. Feedback-based gameplay metrics cannot be used for gameplay segmentation, but their analysis can lead to a novel segmentation of gameplay, considering, not the *game* alone as the document of interest, but the *game as played* as the document to segment. Segmentation derived from feedback-based gameplay metrics means that a game can produce an infinity of possible segmentations in terms of the way the game has been activated by the player. As this is the case, another terminology must be found to avoid ambiguity between the two segmentation approaches, one considering the game as an absolute entity, the other considering the game as relative to a player.

To differentiate the *gameplay segmentation* approach from the one needed in the present thesis, Chapter 3 proposes to use the term of *performance*, as developed by Laurel (1993), whose research compares computer programs with theatre plays. Laurel explains that interface design and theatre have some commonalities, as they both “deal with the representation of action” (Laurel, 1993, p. 14).

Performance, from a theatre point of view, refers to the exact moment in which a play is acted out, and the audience can, indirectly, influence the performance, by laughing, clapping, and so on, while forgetting about the technical aspects. Laurel then imagines what it would be like if the audience was invited on stage, and could influence the play, to engage in more interaction possibilities than just clapping or laughing. From this perspective, a performance can be translated to the videogame world, perceived as a play (the game) that can be influenced by the audience (the player), who is unaware of the technical aspects (the players interact regarding the feedback streams, not directly the game source code).

The term performance as proposed by Laurel becomes a strong one to describe the idea of a "game as played". It embodies the ideas of activation (audience reaction), the idea of representation (the play) and the idea of feedback (how the play is perceived as not being technical). Furthermore, the degree of freedom allowed by a game is also included in the definition of performance. A play can be highly scripted, translatable to linear games; guided, translatable to non-linear games; or improvisation-based, translatable to sand-box games. That is why, in the present thesis, segmentation based on feedback-based gameplay metrics, being the representation of player activations and the game state, established through feedback streams and not the source code, is defined as *gameplay performance segmentation*.

Gameplay performance segmentation is the determination of segments of play that are coherent in terms of player experience and/or game state, from a specific performance of play. This could be for instance the moments when the player is

low in health, or the moments when the player is inside the help menu, or the moment the game triggers a cut-scene etc. Gameplay performance segmentation is not about the detection of the existence of cut-scenes, or game menus, or health. Gameplay performance segmentation is about detecting when these gameplay elements occurred during a performance, and what they can inform in terms of player experience.

It is however important to specify that gameplay performance segmentation does not override gameplay segmentation. Indeed, feedback-based gameplay metrics, constituting the main quantitative data of interest in the present thesis, are unable to identify gameplay elements or mechanisms. However, gameplay elements and mechanisms need to be identified beforehand, as feedback-based gameplay metrics are a description of the way these elements evolve over time. Gameplay performance segmentation must then be seen as a continuation of gameplay performance. Once the game is analysed using gameplay segmentation, it becomes possible to focus on the evolution of each determined segment using a gameplay performance segmentation approach. Gameplay performance segmentation will be more thoroughly described, illustrated and validated in Chapter 4, proposing a multi-layered segmentation structure, Chapter 5, describing algorithms to automatize the segmentation, and Chapter 6, presenting gameplay performance summaries based on actual game session (presented in Section 2.3).

3.3 Document indexing and segmentation: an overview

Before describing further, in the following chapters of the present thesis, the use of gameplay performance segmentation and feedback-based gameplay metrics for player experience understanding, Chapter 3 needs to consider the historical debates arising from the *indexing* and *segmentation* standard disciplines, in order to demonstrate that going back into the standard specifications of indexing and segmentation is a core process that not only strengthen the definition of gameplay performance segmentation, but also bring several notions of interest that enhance the whole gameplay performance segmentation approach. For instance for *indexing*, *open-system* and *closed-system indexing* differentiates whether the *indexing process* is performed for a whole document or for units inside a document; *metadata* represents document properties for *open-system indexing*; *free-text* and *controlled-vocabulary* notions represent two different approaches useful for *closed-system indexing*, and differentiates whether they are mostly content-based or concept-based. Word relationships from within a *controlled-vocabulary* are discussed, with emphasis put on *synonymy*, *hierarchy* and *association*. Similarly for *segmentation*, the reflexions about segment size (i.e., including enough information but not too much, and assessing the coherence of the content), topic shift (i.e., detecting when a segment ends or begins because of specific content cues), reference resolving (i.e., understanding a segment through resolving most of the unresolved references) or semantic network (i.e., understanding how different segments are linked with each other), can profit to the gameplay performance segmentation approach when correctly translated.

Applied to gameplay performance analysis, indexing and segmentation represent two important processes that can help in summarizing a performance of play, in order to help the analysis process by highlighting both the gameplay structure and the understanding of the player experience during this specific gameplay performance. Indexing applied to gameplay performance is the determination of sets of words that can label different gameplay perspectives, e.g. a spatial perspective (e.g., cut-scene, menu, in-game), a degree of interaction perspective (e.g. fully interactive, semi-interactive, no interaction), or an interaction perspective (e.g. fight sequence, loot sequence, story understanding sequence). Segmentation applied to gameplay is the determination of where to place the boundaries (beginning and end time stamps) for a coherent segment, having decided on a particular indexing perspective, e.g., where a cut-scene (indexing) starts and ends (segmentation), where a semi-interactive sequence (indexing) starts and ends (segmentation), where a fighting sequence (indexing) starts and ends (segmentation) etc.

If the following sections introduce the two concepts of document indexing and segmentation from a standard perspective, applicable to a wide range of media, gameplay performance examples are provided to illustrate how each introduced notion also carries strong connections with gameplay performance analysis.

3.4 Document indexing

In the standard librarian discipline, *indexing* is actually an ambiguous term that assumes several characterizations depending whether it refers to the indexer's perspective or the user's perspective, or whether it is applied for a full document or a unit inside a document. For the need of the present thesis, the definition of

indexing is determined as: a process of locating and describing all or parts of a document.

Although generic, this definition still respects the core notion of indexing applied in more traditional disciplines, while opening the way for broader research, like gameplay performance analysis and the inclusion of player experience assessment when the considered document is a gameplay performance. For instance, indexing a gameplay performance can be seen as the process of describing a gameplay performance in its entirety, e.g., from the name of the player or the title of the game down to labelling parts of a performance (time stamped) that are more dedicated to action, strategic thinking; or, in a more spatial way, parts of a performance that are cut-scenes, menu browsing, or in-game moments. The definition proposed above has been built using several removals of ambiguities originating from different perspectives. The present section introduces the academic reasoning that led to this definition, which also revealed further concepts of interest for the research.

3.4.1 *Closed-system vs. open system indexing*

Even if document indexing is a well-established discipline constituted by approaches designed for abstracting purposes since antiquity (Witty, 1973) and the beginning of printing (Wellisch, 1986), as well as for cataloguing purposes since the 19th century (Diakoff, 2004); finding a comprehensive definition of *indexing* is a complicated task. In fact, two different points of view, supported by two different groups of practitioners, share the same term (Klement, 2002). On the one hand the first group of practitioners, roughly librarians (for physical libraries) or database indexers (for the digital ones), try to address the question of specific

document finding within a significant collection of documents. The term *index* then designates a product, detached from any indexed documents (Diakoff, 2004), that is "designed to facilitate finding one or more documents that contain relevant information" (Klement, 2002, p. 2). On the other hand the second group of practitioners, usually described as back-of-the-book indexers, are interested in improving information finding inside a document. The term *index* then characterizes a separate product directly included in the indexed document (Diakoff, 2004) that "assists people in finding a unit or units of relevant information within a document" (Klement, 2002, p. 2).

In order to allow the two groups of practitioners to still employ the term *indexing*, but without ambiguity, Klement (2002) recommends to distinguish the two different approaches by specifying whether they are *open-system indexing* (entire document) or *closed-system indexing* (unit inside a document). Applied to gameplay indexing, open-system indexing can be seen as identifying the context of a gameplay performance, e.g., name of the player, title of the game, date of play, duration; while closed-system indexing can be seen as labelling the content of a gameplay performance, structurally, e.g., cut-scene, menu, in-game, or experientially, e.g., action, story, strategy.

3.4.2 *User or indexer perspective*

If the *closed* vs. *open system* distinction is essential ("vital" as suggested by Klement (2002)), it offers a first overview of what *indexing* means, but it remains difficult to fully understand what the process of indexing actually entails. However, by focusing more on what unifies closed and open systems instead of what separates them, the notion of "information finding" seems to be a key aim

and common motivation. Klement (2002) acknowledges this specificity when she references Milstead's (1984) attempt to distinguish indexing from cataloguing; an attempt that can be directly applied for unifying closed and open system indexing: "They both provide subject access to the content of documents; […] they are different in degree, not in kind" (Milstead, 1984). Milstead's definition is valuable for two main reasons; first this is a definition that offers an initial unified step to understanding indexing and its purposes; second, this is a definition that also introduces the notion of degree that represents the basics of the related segmentation (see Section 3.5).

Although Milstead's definition can be considered as nearly comprehensive, there is another ambiguity that requires resolving. The literature on indexing indiscriminately explains the process of indexing using two related yet distinct verbs: "find" and "describe". Even Klement (2002), in her paper that distinguishes closed and open-system indexing, mixes the use of these two verbs when presenting the indexing process. "Find" and "describe" are related, as they represent two ends of the same process; "find" emphasises the user perspective and specifies the needs addressed by indexing, while "describe" highlights the indexer perspective and defines the action performed to achieve an indexing process. But they are distinct because addressing the problem of indexing through the eyes of a user as "finding" limits the approach of indexing for a unique aim, discarding for instance the use of index for document analysis purposes; while addressing the problem of indexing through the indexer's perspective as "describing" actually opens the use of index for a wider range of possibilities, like document analysis. Replacing the term "finding" with "locating" resolves the

ambiguity, as locating reunites both users' (finding) and indexers' (identifying) perspectives.

For the purpose of the present study, the term indexing then refers to the process of locating and describing all or part of a document.

3.4.3 *Metadata*

Now that indexing has been defined, it becomes possible to focus more on detailing the process of *indexing* by presenting different approaches currently in use that describe document content; approaches that are still debated (Fidel, 1991; McCutcheon, 2009). *Metadata* (current section) is about describing a document in regard to other document (*open-system indexing*), while *free-text* and *controlled-vocabulary* (see Section 3.4.4) are about describing the document itself (*close-system indexing*), and differ regarding if the document description is content-based or concept-based.

The first approach is based on the use of *metadata*. Metadata, in the context of indexing, represents properties and administrative information about objects (Garshol, 2004). Metadata can for instance refer to a document title, author, date or language (Garshol, 2004). Because of the nature of metadata, Garshol also specifies that "it says very little about the subject of an object" (2004, p. 379). Thus, metadata is a method of choice for open-system indexing, but is hardly adaptable for close-system indexing. Applied for the purpose of gameplay performance analysis, metadata represents the set of properties providing general

information about the context of each play session recording, e.g., player's name or ID, game title, date of play, location of play, computer/console used etc.

3.4.4 *Free-text vs Controlled-vocabulary indexing*

For *close-system indexing,* i.e., indexing inside a document, two indexing processes are commonly applied, and diverge according to whether the set of words used for describing the document is primarily content-based or concept-based.

The content-based approach, designated as *free-text indexing,* is the most straightforward one to introduce. The main idea behind free-text indexing is to consider using words that are already employed in the document to index (Fidel, 1991). The indexer selects a set of terms from within the document that carry a significant meaning, and locate them inside the document (for instance by using page numbers). The free-text method is content-based, as the content of the document represents the starting point of the process.

For gameplay indexing, the use of a free-text process can be seen as the process of extracting content exactly as it appears in the game, without further interpretation. In the game *Bioshock 2* (2K Games, 2010) for instance, numerous textual elements appear on the screen every time a player is looking at an object, a wall inscription or a propaganda poster. Locating the instances when a text appears on the screen, and labelling these instances using the text itself is *free-text indexing*. On Figure 10, illustrating a sequence in *Bioshock 2* where the player is looking at a propaganda poster, the index would be the displayed text "Re Juvena Skin-Tightening Tonic Ad" to label the sequence.

Figure 10. Textual information appearing on the screen in the game *Bioshock 2*.

Figure 10 serves as an example to illustrate the difference between Free-text and Controlled-vocabulary indexing. When a player is looking at a propaganda poster, the title of the poster is displayed on the screen. Free-text indexing will use the exact displayed text, to label the sequence. Different propaganda posters will then be labelled differently. However, a controlled-vocabulary indexing will be linked to a concept, like for instance "exploration", indicative that during this sequence, the player is exploring the game world. Then, the different posters will be considered as necessitating the same label, as part of the same experience.

The concept-based approach, designated as *controlled-vocabulary indexing*, can be perceived as taking a reverse approach from free-text indexing. Instead of considering the document first and building an index constituted of words as they appear in the document, the words constituting a controlled-vocabulary index are pre-selected by the indexer, and then compared with each document unit. Two complementary definitions of controlled-vocabulary indexing, proposed by Hedden (2008) and Leise (2008), help to build a clear picture of what this process entails. Hedden (2008) gives a structural description, and defines controlled-vocabulary as "a restricted list of words or terms used for indexing […]; controlled because only terms from the list may be used for the subject area covered by the controlled-vocabulary" (Hedden, 2008, p. 33). Leise (2008) suggests a more process-oriented description, and defines controlled-vocabulary

as a “list of term and term relationship designed to (1) collect similar information, (2) assist content authors in consistently tagging content and (3) enable users to find the information they need by translating their language into the language of the information store” (Leise, 2008, p. 121).

The controlled-vocabulary approach applied to gameplay indexing can be seen as the definition beforehand of a concept of interest. For instance, in the example provided in Figure 10, instead of using the free-text label “Re Juvena Skin-Tightening Tonic Ad”, the same moment can be labelled as “exploration”, considering that the sequence is about the player exploring the game world. Then, each poster looked at will be labelled using the same term, as being part of the same concept. The chosen controlled-vocabulary influences the way a gameplay element is labelled. Referring to Figure 10, a controlled-vocabulary about space will consider this moment as being an in-game one, without distinguishing it from a fighting sequence for instance (permissive controlled-vocabulary); while a controlled-vocabulary about interaction, will consider the sequence as being a looking sequence, or a controlled-vocabulary about experience will consider the sequence as being an exploration sequence (precise controlled-vocabulary).

A noteworthy observation about free-text indexing and controlled-vocabulary indexing is that what differentiates them can also be related to two different intentions for gameplay analysis. In fact, free-text indexing explains a lot about the content at a specific moment, but little about a desired concept (such as player experience for instance); whereas controlled-vocabulary indexing, with a well-chosen set of words, can indicate a lot about a concept, but less about specific

game contents. Concretely, that means that free-text indexing applied to gameplay analysis is a powerful tool for detecting matching moments from within a performance, and also as a comparison of other players' performances. In the *Bioshock 2* example described above, gathering the list of the propaganda poster titles looked at by each player allows for direct comparison (e.g., which posters have been seen by all the players? which ones are generally ignored? at what time stamp is each poster seen?). Controlled-vocabulary indexing, on the other hand, is much more an analysis tool in line with a concept. Obviously, it would still be possible to compare several performances, but then these comparisons will be concept-driven, not content-driven, and would have another degree of precision. Two different cut-scenes in *Bioshock 2* would be labelled "semi-interactive" (they can be skipped) in the case of a degree of freedom concept, and thus considered similar; whereas a free-text approach would label them as distinct, offering the possibility to match them more precisely between two performances.

The distinction is particularly central, as the algorithms and approaches presented later in the thesis are discriminated using the content-based vs. concept-based distinction. Concept-based, or controlled-vocabulary, indexing constitutes the foundation of the multi-layered structure that will be presented in Chapter 4, and also defines the different algorithms seeking to gather feedback-based gameplay metrics in Chapter 5; while content-based, or free-text, indexing constitutes an alternative approach for comparing performances, as illustrated in Section 6.2.

3.4.5 *Synonymy, Hierarchy and Association in controlled-vocabulary*

In the case of controlled-vocabulary indexing, the indexing result relies on a concept characterised by a chosen list of words, constraining indexers to study

and understand the relationships that document words share with other document words, and also with controlled-vocabulary selected terms. Three types of relationships are commonly presented when controlled-vocabulary indexing is considered: synonymy, hierarchy and association (Leise, 2008).

Synonymy relationships link two words that have meanings so similar that they may be considered as near-identical. In the context of gameplay, in the game *Dead Island* (Deep Silver, 2011) for instance, the player can create homemade weapons. The creation of homemade weapons can be variously categorised as, for example, "weapon-crafting", "weapon creation", "weapon personalisation", while actually representing exactly the same action.

Hierarchy represents a relationship whereby "one term is broader in meaning than its child term" (Leise, 2008, p. 22). In *Dead Island*, "Pistols", "Rifles", and "Shotguns" can be considered as the children of "Firearms"; whereas "Axes", "Hammers", and "Sticks" are all "Melee Weapons"; both groups are the children of the larger category "Weapons". The Game Ontology presented in Section 3.1 and illustrated in Figure 9 is another example of a hierarchy of *controlled-vocabulary* terms.

Finally, the third type of relationship, termed *association*, is a relationship exhibiting any kind of association between two terms. For instance in *Dead Island*, "Weapon making", "Fighting", "Weapon repairing", and "Weapon dropping" are related as they are all connected to weapons, even if "Weapon making" "Weapon repairing" and "Weapon dropping" carry a more strategic

notion, and “Fighting” a more action-driven notion. Figure 11 illustrates another *association* example, using the game *Max Payne 3* (Rockstar Game, 2012). When the player is in melee-combat mode, the player can slow-down the time; or control a kill-cam; or regain some life during when hurt during a last man standing section. The “execute”, “slow-down”, “kill-cam” and “last standing man” sections are clearly different in terms of offered interactions, but they have in common their use of the slow motion effect. They are then highly associated and would benefit from being linked during the indexing process.

It is interesting to note that the vocabulary relationships can also be seen as the determination of a semantic network, a concept also tied to *segmentation*, as further developed in Section 3.5.3. Furthermore, the definition of a *controlled-vocabulary* and its elements of synonymy, hierarchy and association regarding a chosen concept can be seen as a first step toward gameplay analysis, which will be covered in Chapter 4.

Figure 11. Different game sequences in *Max Payne 3*, illustrating the "association relationship" through the use of slow-motion.

Figure 11 illustrates how the association relationship can also be used to link sequences that are different in terms of interactions, but share a gameplay common point. Here, the player can execute an enemy (top left corner), consciously slow-down the time (top right corner), control a kill-cam (bottom left corner) or try to be offered a second chance during a last man standing sequence (bottom right corner). The actions and outcomes of each sequence are different, but they are sharing the use of slow-motion, and can then be considered as associated in regard to this particular gameplay element.

As already mentioned in present chapter, the notion of indexing is interdependent of the notion of segmentation (that Milstead (1984) introduced as indexing degree). The next section is then dedicated to the segmentation process, as defined and used in the standard disciplines.

3.5 Document segmentation

Milstead (1984), in her attempt to reconcile different indexing points of view, points out that the actions of indexing a full document, or indexing units from within a document, are similar; what differentiates them is the degree of precision. By saying so, Milstead (1984) initiates a first step toward the notion of segmentation, and toward the description of what *segmentation* entails. Indeed, while Milstead's definition (1984) acknowledges the existence of different degrees of precision in the indexing process, segmentation seeks to address the precision – sometimes called granularity – in an optimal way. More precisely, starting from the assumption that most documents contain, in general, more than one subject, segmentation is defined as the process of "dividing lengthy documents in topically coherent sections" (Reynar, 1998, p. 3). The quality of being "topically coherent" – sometimes also called homogeneity – is the principal reason that illustrates why indexing and segmentation may be difficult to separate and distinguish. In fact, detecting topically coherent sections implies that a topic, or concept, has been determined beforehand, with a corresponding descriptive vocabulary. The idea of a pre-determined concept and matching vocabulary constitutes the foundation of the controlled-vocabulary indexing process introduced in Section 3.4.4. To summarise, segmentation is inconceivable without controlled-vocabulary indexing in mind, and indexing is challenging without determined boundaries of the segment to label.

The idea of granularity can be illustrated using the weapon example provided in Section 3.4.5, in the game *Dead Island*. A weapon can be crafted from elements found in the game world, equipped, and used. A coarse-grained granularity would

consider these different actions as being part of the same segment (they might be, for instance, all labelled “weapon-related action”); whereas a fine-grained granularity would segment these three different action accordingly (“craft of weapon”, “weapon equipped” or “use of weapon”).

If finding a unified definition of indexing necessitated the resolution of several ambiguities (see Section 3.4), the definition of segmentation suggests a better consensus. This may be explained using Klement’s (2002) claim that the lack of proper attention to distinguishing open-system and closed-system indexing is because “much of indexing theory concentrates on the product (the index), rather than the process of creating the index” (Klement, 2002, p. 23). However, if indexing is linked to a final product, the index, segmentation is much more a process than a product, which may explain, according to Klement, why it is easier to define. The discussion in the present section is then mainly dedicated to the different approaches that have been standardly used for accomplishing a correct segmentation process. The segmentation process can be divided into two principal subgroups of focus (Reynar, 1998): determining the homogeneity of segments (meaning detection), and determining and locating structuring cues (structure detection).

The first group relates to the study of meaning, and seeks to detect meaning-breaks inside the document. The second one focuses on the detection of specific elements that self-sufficiently carry structuring properties (e.g., connector words). For gameplay segmentation, determining the homogeneity of segments means studying the content of a segment of gameplay, and assessing its homogeneity.

For instance, sections containing a HUD, in general indicate fully interactive sequences. Determining structuring cues will be more focused on very specific elements indicative of a change; for instance, a black screen, a death screen or the disappearance of the very moment the HUD disappears off the screen, indicative of breaks in the performance.

3.5.1 *Homogeneity: segment sizes and unresolved references*

As underpinned by the given definition of segmentation, one of the main challenges of segmentation is to determine topically-coherent segments. That definition implies that a segment should be short enough to only encompass one topic, but also large enough to get a sense of what the topic is. Estimating the correct balance between small enough and large enough is the core facet to consider for obtaining the most accurate possible segmentation. For instance in *Bioshock 2*, the game is cut into different levels. Using a spatial point of view, level separations can be small enough; but in terms of player experience, a full level would be too large because different experiences (i.e., game story elements, looting, fighting, healing) can constitute a level. On the other hand, segmenting a conversation moment with a Non-Player Character (NPC) would be large enough as a story-oriented moment; while segmenting each sentence of the conversation may be too small.

Other factors can also sometimes tip the scale differently between small enough and large enough, like the intended main goal of the *segmentation*. Reynar (1998) points out that, when segmentation is performed with a summarising goal in mind, segments need to be large enough to resolve most of the potential undefined references, such as pronominal references, so that the summary can be self-

sufficient. The notion of unresolved references is an essential one, because it emphasises the need to understand a document's content. Applied to gameplay analysis, resolving references can be understood in two different ways. First, resolving references can be seen as the process of identifying the meaning of some elements that are presented to the player in a schematised way. For instance, in *Bioshock 2*, the life-bar is nothing more than a bar displayed on screen and, without context, can be representative of different information, e.g., stamina, remaining bullets, experience level. But the context can resolve the ambiguity. Figure 12 illustrates how game design conventions can lead the player to resolve references. In *Bioshock 2* and *Max Payne 3*, the life bars are filled with red colour (indicative of blood), which is quickly recognised by a player as being related to health. The evolution of the bar regarding the rules is also a way, for a player, to resolve the bar usage. Indeed, the player may notice that the life bar diminishes in size every time he/she is hit, and then the player can understand the bar purpose for the rest of the performance.

Life bar in *Bioshock 2*

Life bar in *Max Payne 3*

Figure 12. Life bars in *Bioshock 2* and *Max Payne 3*.

Figure 12 illustrates how a bar representation, in Bioshock 2 and Max Payne 3, can be easily understood as being a life bar by its use of the red colour.

Resolving references can also be seen as determining one element, in a segment, that gives the full segment its meaning. In *Max Payne 3* for instance, the sequences without HUD can indicate a bullet cam sequence, a kill-cam sequence, a last-man-standing sequence or an intra-mission (called chapters) cut-scene. However, Figure 13 illustrates how the appearance of the word "chapter" during a segment resolves the ambiguity of the full segment, and identifies it as being an intra-mission cut-scene.

Figure 13. Intra-chapters cut-scene in *Max Payne 3*.

Figure 13 illustrates how a specific element in a segment (the appearance of the word "chapter" in Max Payne 3) can resolve the ambiguity of the full segment (then being an intra-chapters cut-scene).

3.5.2 *Homogeneity: coherence vs. cohesion*

In the definition proposed by Reynar (1998), the notion of *coherence* represents a pivotal point governing the segmentation process. However, while central, *coherence* is not an easy term to define, especially as *coherence* is directly linked to the concept of *cohesion*, with which it is often confused (Reynar, 1998). The extensive debate about cohesion and coherence is a very specialised one (Halliday & Hasan, 1976; Tangkiengsirisin, 2012), and requires rigorous linguistic knowledge that is beyond the scope of the present thesis. However, the distinction between coherence and cohesion is meaningful for the segmentation process and can easily be appreciated. Two sentences are cohesive if some elements in one sentence need the second sentence to be interpreted (Reynar, 1998). Halliday and Hassan (Halliday & Hasan, 1976) differentiate several types of cohesive relations: references, substitution, ellipsis, conjunction and lexical cohesion. The cohesion is mainly based on semantic structure, but "does not concern what a text means" (Halliday & Hasan, 1976). Two sentences are coherent if they are cohesive and if the meaning is homogeneous.

Because cohesive and coherent are difficult to define, but easy to appreciate (Reynar, 1998); two examples are given. "John is in a plane. He likes dogs" are cohesive, because "he" and "John" are related; but the two sentences are not coherent. "John is in a plane. He enjoys flying" is cohesive and coherent.

The ideas of cohesion and coherence are interesting when applied to gameplay analysis. Figure 14 illustrates how, during the first level of *Max Payne* 3, the player is involved in a shooting sequence taking place in a building. Then, he/she

can run toward a roof exit. The camera angle changes and a cut-scene begins with the avatar walking on the roof. Then the avatar jumps, and the player gains some control again. In terms of the story, this section of gameplay may be seen as *coherent*, but in terms of the level of interaction, it is only *cohesive*; this is because the style of play shifts from total interaction to no-interaction, and then to limited interaction, even though the character and the location remain the same. Still in *Max Payne 3*, Figure 14 also demonstrates that, at the end of Chapter 3, a cut-scene begins and shifts to a flashback introducing Chapter 4. Here, the story is *cohesive* because the time and space suddenly change, but the level of interaction is *coherent* because this remains the same cut-scene.

Story coherent but degree of interaction not coherent.

Story cohesive (same character) but not coherent (shift in space and time), but degree of interaction coherent.

Figure 14. Illustration of the cohesion vs. coherence distinction applied to gameplay analysis.

Figure 14 illustrates how the notions of cohesion and coherence can be understood when applied to gameplay analysis. In the top example, the avatar is inside a building, going toward the exit; then he is on the roof, and he finally jumps. The game story is perfectly coherent, but the gameplay is not. If the sequence inside the building is fully-interactive, the roof sequence is a non-interactive cut-scene, and the jump sequence is a limited-interaction sequence (the player can only shoot). In the bottom example, the story is only cohesive (focused on the same character), but not coherent (the character is in a helicopter, and then suddenly in a pub, several years ago). However, the three displayed sequences are part of the same cut-scene, so the gameplay is coherent, as the player is offered the exact same degree of interaction.

The two examples in Figure 14 illustrate that coherence and cohesion also depend on the concept of interest, and so establish a link with the discussion initiated in Section 3.4.4 about how a controlled-vocabulary choice can also influence the precision of the labelling process. This is the main rationale of Chapter 4, which proposes several layers of analysis, each layer carrying its own coherence definition, for instance a "spatial-temporal" coherence, "degree of freedom" coherence, or "interaction" coherence.

3.5.3 *Homogeneity: semantic networks*

The notions of *segment size* and *references* presented in Section 3.5.1 address the problem of segmentation in a linear way: a segment is a set of consecutive words, sentences or paragraphs. The notions of coherence and cohesions pave the way to a less linear view, as two sentences can be related even if they are not consecutive. Indeed, the notion of texture (Halliday & Hasan, 1976), mainly based on cohesion and coherence is, in general, applied to a full text. The concept of the *semantic network* (Skorochod'ko, 1971) goes a little further and formalises the different possible semantic relationships between segments in a full document, whether they are consecutive or not. Applied to the study of gameplay, this notion of semantic network can identify and link similar segments of play; for instance, every player death, whenever it occurs, can be linked and considered as semantically identical to the other player deaths. It is possible to go further than that. For instance, in *Max Payne 3*, different gameplay mechanisms use slow motion effects (kill-cam, bullet-cam, last-man-standing). These can then be seen as part of the same "slow motion" semantic network. In *Battlefield 3*, being out of the game world boundaries will trigger a countdown, leading to a mission failed once the countdown reaches zero. Both the actual failure, and the countdown, can

be seen as part of a “mission failure” semantic network. Finally, in *Dead Island*, the possibility offered to the player to gather objects in order to craft weapons, the chance to upgrade weapons, and the use of the weapons, can be seen as part of the same semantic network, even if the player’s motivation is seen as different (gathering, crafting and upgrading is more strategic, and weapon use in more action-oriented).

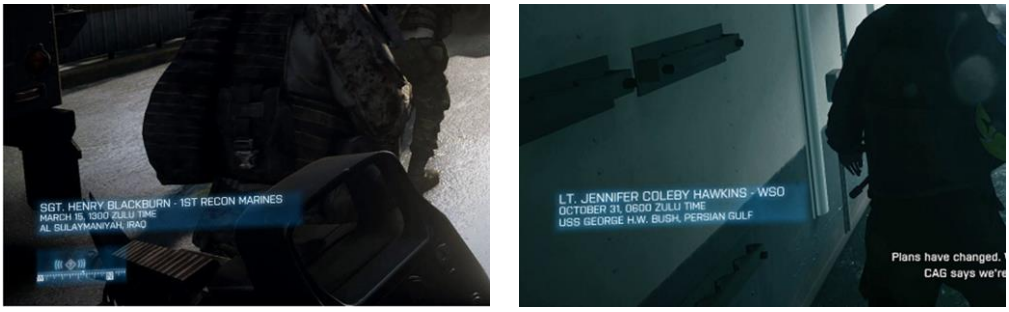
3.5.4 *Structuring cues: topic boundaries*

In the above sections, the determination of segments is based on the document meaning, trying to understand a segment, its unresolved references, its coherence, and its link with other semantically related segments. The following sections are about focusing on the document’s structural properties that can determine segment breaks.

The first presented idea about *topic boundaries* can be seen as the frontier between homogeneity and structuring cues. On the one hand, topic boundaries deals with the idea of detecting breaks in the document homogeneity; and on the other hand, topic boundaries rely on cues used to detect a topic change. Reynar (Reynar, 1998) discusses the two different points of view of Grimes (Grimes, 1976) and Nakhimovsky (Nakhimovsky, 1988), who propose to consider specific cue aspects to determine *topic boundaries*. Grimes (Grimes, 1976) focuses on changes in scene, participant orientation, chronology or theme; Nakhimovsky (Nakhimovsky, 1988) focuses on topic shifts, such as shifts in space and time, discontinuities of figure and ground, and changes in narrative perspective. In the game *Battlefield 3* for instance, during the main campaign, the player controls different avatars (participant orientation), and can see them from a third person

perspective during the cut-scenes, and from first-person perspective during in-game moments (changes in narrative perspective), during flashbacks or present time (shift in time, chronology), in different cities (change in scene, shift in space), and is using different vehicles (change of theme). Figure 15 illustrates the participant orientation shift, and the narrative perspective shift in the game *Battlefield 3*.

Change in narrative perspective (first person vs. third person view)



Change in participant orientation (Sgt. Henry Blackburn vs. Lt. Jennifer Coleby Hawkins)

Figure 15. Topic shifts in the game *Battlefield 3*.

Figure 15 illustrates two topic shifts in the game Battlefield 3, being a strong indicator of a change of segment during a segmentation process. In the top example, the narrative perspective changes from first person to third person view. On the bottom example, the participant orientation changes, as the player is controlling with different avatars (Sgt. Henry Blackburn and Lt. Jennifer Coleby Hawkins).

3.5.5 *Structuring cues: cue elements and pre-existing structure*

Sometimes, documents exist with a pre-existing partition already in place (Reynar, 1998). For a thesis document, this is for instance Title – Abstract – Acknowledgements – Contents – Chapter 1 – … – Chapter n – Conclusion. The intrinsic nature of a thesis conditions its structure. For a videogame, a *pre-existing structure* is in general Splash Screen – Title Screen – Menu – Introduction Cut-Scene – Level 1 – … – Level n – Ending Cut-Scene. If a *pre-existing structure* is in general not specific enough to be considered a final segmentation, it offers a relevant first step to initiate the segmentation process.

Another similar approach is to base the segmentation, not on pre-existing structures, but on cue elements that are known to carry structuring properties. For instance, connector words such as ‘however’, ‘although’, ‘since’, and ‘then’; are likely to indicate the beginning of a new segment for textual documents. Applied to gameplay analysis, *cue elements* can be loading screens, cut-scenes, game-over screens, and so on. Figure 16 illustrates different cue elements in *Battlefield 3*. Indeed, each mission beginning or retry is identified by a specific splash screen, each death is accompanied by the same sound and screen, and each Quick Time Event (QTE) has the same key-to-press information displayed on screen. These cues can be used through feedback-based gameplay metrics, to determine the nature of a segment.

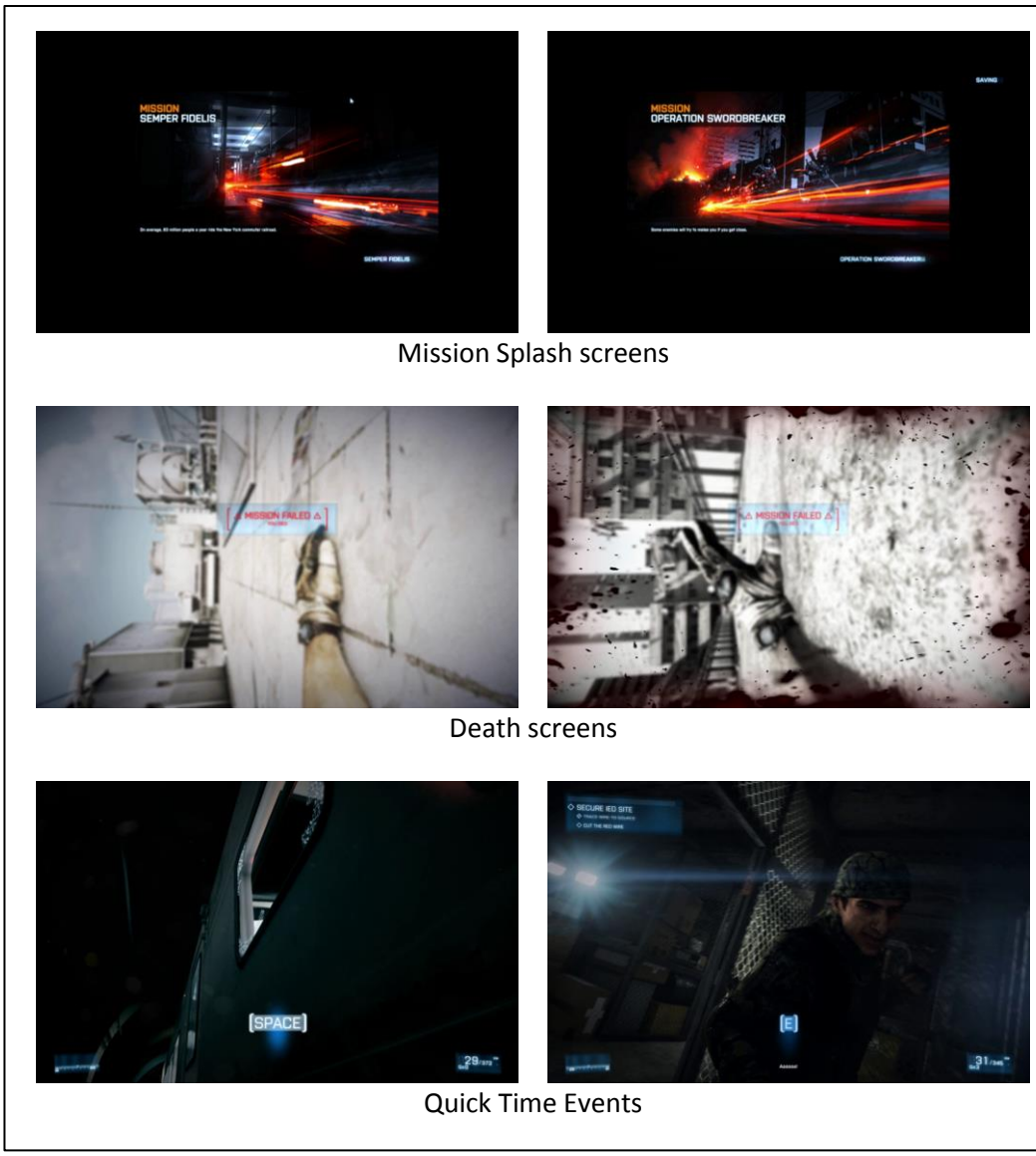


Figure 16. Cue elements in *Battlefield 3*

Figure 16 illustrates the use of cue elements to help determine segments. In Battlefield 3, each mission beginning or retry is indicated by a specific mission screen (top); each death sequence is recognisable by a specific sound and visual atmosphere (middle); and each QTE is recognisable by the presence of "key to press" information on screen (bottom).

The present section introduced the concept of segmentation. Because segmentation is directly related to the idea of topic-coherence, its strong interconnection with close-system indexing was illustrated. Some core concepts, adaptable for gameplay analysis, have also been presented, including unresolved references, cohesion, coherence, semantic networks, topic boundaries, cue elements and pre-existing structures. These notions, and their applicability to gameplay analysis, will be further detailed in Chapter 4.

3.6 Audio-visual indexing and segmentation

In Section 3.4 and Section 3.5, the notions of indexing and segmentation were presented, along with important linked properties (e.g., closed and open system indexing, controlled-vocabularies, unresolved references, coherence and cohesion). Most of the terms used in Section 3.4 and Section 3.5 are about textual documents. However, an important strand of research, which can be connected to the feedback-based gameplay metrics approach, also seeks to address indexing and segmentation for various media documents, including TV-stream, music, and movies. A first instinctive approach would be to transcribe any textual or speech contents of a multimedia document, and apply the usual indexing and segmentation methods to the transcription (Reynar, 1998). The transcription idea is an interesting one, but suffers two major limitations. First, a transcription process needs to be performed beforehand. Then, core audio-visual elements are usually not transcribed (e.g., black frames, silence, change of camera angle) and are discarded even if they carry a strong meaning.

When considering audio-visual material for indexing and segmentation, the core approaches seek to go further than the transcription idea, and use indications

directly enclosed into the considered media stream, whether they are textual (e.g., text, speech) or not. The present section presents several examples of indexing and segmentation focused on music, moving image and television or radio streams. If the elements to consider are quite apparent for textual documents, audio-visual ones usually need a layer of abstraction, which means that the stream content must be translated into interpretable elements. Here, the emphasis is on the different elements of abstraction associated with the process of *indexing* and *segmentation*. More information about automatic processing methods can be found in Section 3.7.

3.6.1 *Music*

One highly influential work for the purpose of indexing and segmentation – applied to music – is the "Traité des Objets Musicaux" by Schaeffer (1966). Schaeffer proposes a full vocabulary for describing musical elements in an abstract way, i.e., applicable to a wide range of music genres, including electroacoustic music. Schaeffer's work, which may be seen as the establishment of a controlled-vocabulary for musical indexing, has been adapted for *segmentation* purposes. Indeed, the derived concept of "Temporal Semiotic Units" (Frey et al., 2009) represents segments of music "that convey a meaning through their dynamic organization in time" (Frey et al., 2009, p. 248) and which are classified into categories, including 'Enthusiastic', 'Falling', 'Floating', 'Stationary', and 'In Suspension'. Another way of considering music as a stream is to use parameters from the audio signal itself. For instance, the study of the pitch homogeneity of musical segments can help to identify repetitions (Martin et al., 2011) and, therefore, global musical structures can be utilised for description purposes (Chai, 2006). The process of identifying repetition using musical

segments can be, for instance, applied for gameplay analysis, by focussing on the soundtrack. Similar soundtrack elements are likely to represent similar sections in different gameplay performances. Chapter 5 and Section 6.2 will present several instances in which sound similarity techniques have been used to study *gameplay performance*.

3.6.2 *Television/Radio Stream*

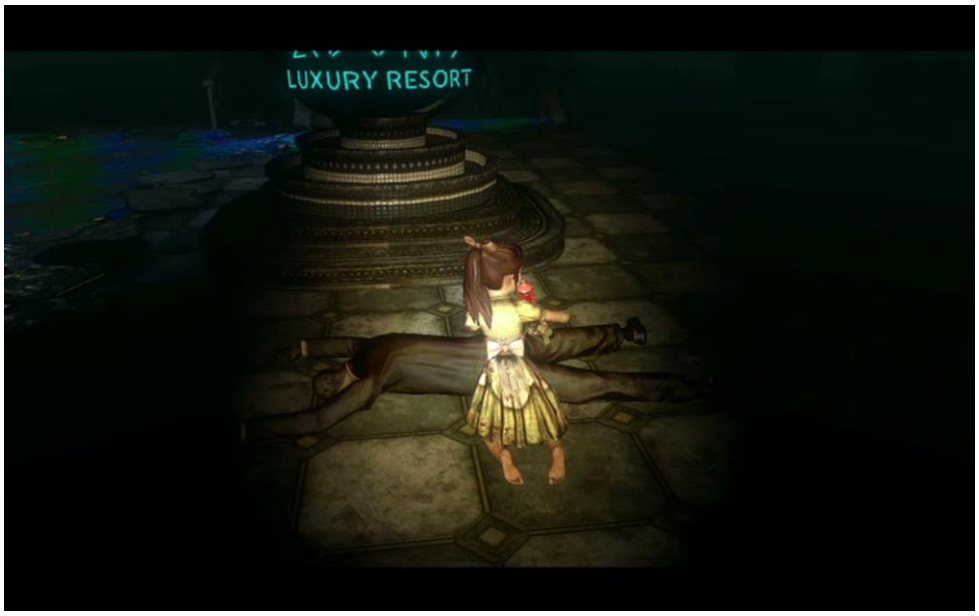
Television and radio broadcasts are rarely uniform, and are commonly constituted of various types of programs, e.g., news, shows, music/movie, commercials etc. Although television and radio programs are easily distinguishable by a human being, some objective clues can be determined to differentiate and recognise each program type. Segmenting a television or radio stream regarding the main program class can be seen as trying to assess the boundaries of a pre-existing broadcast structure (see Section 3.5.5). Several research studies have been conducted to highlight objective elements separating two different programs in the Television/Radio streams, such as speech vs. music considerations (Saunders, 1996); channel logo detection (Santos & Kim, 2006), (e.g., advertisement (no logo) vs. show (logo) vs. live transmission (different logo)); or black frames and audio decrease detection structuring a sequence of advertisements (Sadlier, Marlow, O'Connor, & Murphy, 2002).

In *Bioshock 2*, for instance, a narrative component of the game is the opportunity for the player to collect and listen to audio tapes, describing the past of the city in which the player is evolving. Each audio tape contains a character speaking. When speech is detected, then the gameplay performance can be, like a radio

broadcast, structurally segmented. Section 6.2 will demonstrate the use of speech detection on gameplay performance.

Similarly, Figure 17 illustrates how the presence or absence of a logo on screen can be assimilated to the presence or absence of a HUD during the play. In *Bioshock 2*, this can be used to discriminate a fully interactive segment (HUD on screen) from a cut-scene (no HUD). In the top image, the HUD is present, indicative of a full interaction sequence, during which the player can fight and use power. In the bottom image, the HUD is absent, indicative of a sequence where the player is no longer required to fight (the life bar and power bar become meaningless). The bottom image is indeed a cut-scene. The two sequences happen close to each other (the same action is taking place in front of the player), indicative of the exact segment break moment between in-game and cut-scene. The logo detection method applied to gameplay elements is the basis of the static and moving information detection algorithms that are presented in Chapter 5.

Just before the cut-scene, HUD still on screen.



Cut scene begins, no HUD.

Figure 17. Logo presence/absence detection, indicative of different gameplay segments.

Figure 17 illustrates, using the game Bioshock 2, how the use of a logo detection method can discriminate two segments different in terms of degree of interaction. On the top image, the HUD is present, indicative of a full interaction sequence. On the bottom image, the HUD is absent, indicative of a cut-scene in which the player is not required to fight. As the two sequences appear to be close to each other, Figure 17 illustrates the exact moment when the game changes from one gameplay mode to the other.

3.6.3 *The moving image*

In terms of the moving image, three levels of granularity are in general proposed for *segmentation* and *indexing*: shot, scene (or sequence), and chapter (Chasanis, Kalogeratos, & Likas, 2009; Davenport, Smith, & Pincever, 1991; Hanjalic, Lagendijk, & Biemond, 1999). A shot is "the smallest addressable unit having the finest level of descriptive granularity" and "consists of one or more frames generated and recorded contiguously and representing a continuous action in time and space" (Davenport et al., 1991, p. 68). A scene is composed of several shots that "take place in the same physical location or […] that describe an action or event" (Chasanis et al., 2009). Finally, a chapter is a "group of scenes describing a sub-theme in the movie" (Chasanis et al., 2009). The three levels of granularity are fully hierarchical, as a chapter is a group of scenes, which are themselves a group of shots. Different elements are generally taken into consideration for describing a shot, a scene or a chapter. A shot can for instance be described using four parameters, i.e., perspective, camera and sound recorder, content and context (Davenport et al., 1991). The visual content determines shot boundaries. This is the concept of Logical Story Units (Hanjalic et al., 1999), whereby a change in scenery, background, people, face or dresses in the video involves the end of the current shot and the beginning of the next one. In *Max Payne 3* for instance, it is possible to consider the determination of shot boundaries as an indicator of change from in-game to cut-scene, and as an indicator of a sequence change from within a cut-scene.

3.7 A computer science approach: automatizing indexing and segmentation processes

In Section 3.4, Klement's (2002) understanding of the ambiguity underpinning the term indexing suggests that indexing is more often described as a product rather than a process. It would then seem straightforward to deduce that computer science approaches to indexing and segmentation would not suffer any degree of ambiguity, as computer science research relies mainly on describing a process that accomplishes a specific task. Unfortunately, the same degree of ambiguity as the one described in Section 3.4.1 exists in computer science. In fact, indexing and segmentation are two core areas of research which seek to address *information retrieval* challenges, and no unified definition of information retrieval exists. Belkin (1992) references two definitions of information retrieval: "An information retrieval system is considered to have the function of leading the user to those documents that will best enable him/her to satisfy his/her need for information") (Robertson, 1981, p. 10) and "The goal of an information [retrieval] system is for the user to obtain information from the knowledge resource which helps her/him in problem management" (Belkin, 1984, p. 112).

As in Section 3.4.1, the open-system vs. closed-system distinction emerges because the first definition pertains only to an open-system, while the second definition can be applied to both open- and closed-systems, provided that the "knowledge resource" may refer to a single document. The more inclusive second definition, then, is the one selected for the needs of the present argument.

The main contribution of the computer sciences to the notions of indexing and segmentation is to propose methods for automatizing indexing and segmentation processes for a wide typology of documents (e.g., textual, audio, visual). The main rationale is to allow relevant information to emerge from important multimedia databases, sometimes even growing in size rapidly.[9] The open vs. closed system distinction still exists because computer sciences have to deal with finding a document in a massive database, but they must also be able to describe the content of a document for other purposes such as structural determination and description of music or video. For gameplay performance analysis (a performance as an audio-visual record of gameplay footage), automatizing open-system indexing, closed-system indexing and segmentation would allow for automatic summarisation of gameplay performances, permitting researchers to pinpoint moments of interest and study correlations between players.

It is important to acknowledge that, while computer science approaches for automatizing indexing and segmentation of documents have been commonly applied to text, movie, music, television and radio stream documents, they have yet to be applied to gameplay footage analysis. Automatic deconstruction of gameplay sessions into structurally and experientially coherent segments, via *feedback-based gameplay metrics* data, represents an approach that could improve the task of gameplay analysis as it exploits visual and audio information already enclosed in gameplay footage recordings, but also makes gameplay footage

[9] For instance on the Internet, the video website Youtube receives 100 hours of new videos every minute (2013 statistics given on Youtube website).

valuable as test-cases for the different computer science approaches to *indexing* and *segmentation*.

3.7.1 *Open system*

Automatizing open system indexing for gameplay analysis is not the main concern of the present research project. While the administrative metadata about a session are meaningful (e.g., name of the player, title of the game, date, duration), they can easily be manually entered by a user, and the performance analysis is more important for player experience understanding than for session retrieval. However, an automatic open-system indexing system can still be useful in the case of recorded gameplay performances that have not been previously tagged such as, for instance, performances gathered from the Internet. Most of the standard video file formats already enclose most of the useful information (e.g., date, duration). The game title can be automatically found by comparing the video with other already tagged videos and studying the similarities. Unfortunately, some *Metadata*, such as the player's name, cannot be automatically detected. For the purpose of gameplay analysis, *closed-system* consideration of gameplay performances are much more meaningful, and are presented in Section 3.7.2 and Section 3.7.3.

3.7.2 *Coherence and breaks detection*

The core closed-system approach to automatic indexing and segmentation concerns the determination of rules, or parameters, which are supposed to remain consistent in order to assure the coherence of a segment, or the detection of elements that are known to carry a segment break meaning. For instance, for video scene change detection, a parameter to consider for segment coherence is

often a homogeneity in the colour distribution, or in the number of edges detected, from one frame to another (Adhikari, Gargote, Digge, & Hogade, 2008). Automatic logo detection, based on edges detection, is also a way of assessing the homogeneity of a segment (Santos & Kim, 2006). In terms of a break, a black frame (Sadlier et al., 2002), a silence, or an advertisement jingle are examples of breaks that can be taken into account. Finally, a more content-based approach is also regularly used to assess the coherence of a segment, is the automatic detection of people, objects or settings in a video (Snoek & Worring, 2005). Detection coherence and breaks detection can be seen as elements of: closed-system indexing (see Section 3.4.1), controlled-vocabulary (see Section 3.4.4) in that a set of processing rules has to be defined, and linked to the segmentation notions of coherence (see Section 3.5.2), and structuring cues (see Section 3.5.4 and Section 3.5.5).

Applied to gameplay footage analysis, coherence and break detections would be used to determine a set of audio-visual elements that insure a structurally or player experience based coherence (e.g. absence of HUD during the cut-scene, presence of HUD during in-game moment, presence of a specific icon during conversation, same colour distribution during an extended period, menu title detection), or that are known to break a coherence (e.g. death screen, loading screen, sudden drop in health). This can be done using feedback-based gameplay metrics outcomes, and will be further developed in Chapter 5, which discusses audio-visual algorithms for automatic gameplay footage analysis, and in Chapter 6, which focuses on analysing gameplay performance summaries produced using these algorithms.

3.7.3 *Similarity detection*

Another important strand of research about information retrieval is the automatic detection of similar sections from within a document (for instance musical structure based on repetition detection (Hanna, Robine, & Ferraro, 2008)), or between several documents (for instance, automatic plagiarism detection (Robine et al., 2007)). The main idea is to consider a unit of a document, i.e. words for textual documents (Choi, 2000) or frames for musical (Hanna et al., 2008) and video (M. L. Cooper & Foote, 2001) documents; and to evaluate the similarity of these units with other units. The similarity is a quantitative measure that studies the commonality and differences of two elements, and is assumed to be maximal when the two elements are identical (Lin, 1998). Because similarity estimation relies directly on the content of a document, this research can be seen as being relevant to: closed-system indexing (see Section 3.4.1), free-text (see Section 3.4.4) and mostly link to the notion of semantic network for the segmentation process (see Section 3.5.3).

Applied to gameplay analysis, similarity detection can be a valuable process for detecting repetitions from within a performance, e.g., die-retry segments, similar death screen or sound or similar cut-scene. Section 6.2 will illustrate further the idea of the similarity applied to gameplay analysis, using similarity matrices.

3.8 Conclusion

The main contribution of Chapter 3 was to exhibit how the two core concepts of *segmentation* and *indexing* can be translated and adapted for the purpose of analysing *gameplay* and *player experience.* As *gameplay segmentation* is an approach already formalised by Zagal et al. (2008), Section 3.1 discussed the rationale of this approach, and concluded that, while valuable for identifying gameplay elements constituting a game and governing its mechanisms, a *gameplay segmentation* outcome is independent of the way a game is activated by players. Gameplay segmentation is useful in identifying game elements likely to be influenced by a player (Section 3.2). However, there has been a need for a method for studying the evolution of these elements when activated by players, and the proposed gameplay performance segmentation is a novel approach. Gameplay performance segmentation focuses on segmenting, rather than the game itself, so it can be linked to the feedback-based gameplay metrics quantitative data presented in Chapter 2.

In order to strengthen the gameplay performance segmentation approach, Chapter 3 also acknowledged the fact that the two interrelated concepts of segmentation and indexing are well-established, and tied to numerous standard specifications. By considering the historical debates around segmentation and indexing, and by proposing several gameplay examples, Section 3.4 and Section 3.5 illustrated that the standard segmentation and indexing specifications can actually be translated into a gameplay performance document. The presentation of these specifications, translated for gameplay performance, will enhance the understanding of the subsequent thesis chapters. The notions of controlled-vocabulary and close-

system indexing are central in the multi-layered structure of gameplay performance segmentation presented in Chapter 4; the notions of controlled-vocabulary and structuring cues help toward an understanding of the algorithms presented in Chapter 5; the notions of semantic network and unresolved reference are important for player experience analysis, as demonstrated later in Chapter 6; and the notion of free-text is necessary to understand the comparison methods presented in Section 6.2.

Finally, Chapter 3 presented, in Section 3.6, several audio-visual segmentation and indexing approaches and, in Section 3.7, the possibility of automatizing these approaches using computer science contributions; thus the two concepts of segmentation and indexing are connected with the concept of feedback-based gameplay metrics as presented in Chapter 2. Linking gameplay performance segmentation and feedback-based gameplay metrics paves the way for an automatic gameplay performance segmentation method, and this will be further described in Chapter 5 (algorithms) and Chapter 6 (results).

Chapter 4

A Multi-layered Architecture for Deconstructing and Reconstructing Gameplay

In Chapter 3, the concept of a *gameplay performance segmentation* was presented as an extension of the *gameplay segmentation* concept defined by Zagal et al. (2008). Gameplay performance segmentation focuses on a play performance reflecting the relationship between a player and the game. Segmenting based on a performance requires a method that can capture and process play in terms of its constituent parts. Gameplay metrics have been identified as a method that has the potential to capture player engagement with and within a game system.

While several isolated examples of gameplay performance segmentation have been explored in Chapter 3, an adequate model of gameplay performance segmentation has yet to be established. As part of the scholarly interest in notion of *segmentation* and *indexing* covered in librarian disciplines (see Sections 3.3 and 3.4), it was established that a gameplay performance segmentation would produce different outcomes based upon the different conceptual choices guiding analysis. In Section 3.4.5, the introduction of *controlled-vocabulary* indexing illustrated how similar moments of a performance can be labelled, or indexed, differently depending on the concept of interest. For instance, actions or activities within games may be labelled *fighting* as they present visual representation of conflict, but the conditions under which fighting appears in a game can be qualified in terms of whether it occurs in-game, indicating player activity, or within the context of a cut-scene and therefore outside the player's control.

Using an analysis of *Bioshock 2* as an example, this chapter provides an account of what a gameplay performance segmentation can bring to an understanding of player experience when achieved via a multi-layered deconstruction and reconstruction process. The rationale for the selection of *Bioshock 2* relates to its complexity as a game world, which offers players a large spectrum of different play experiences. For instance, it is possible for a player to be action-focused and ignore the numerous game story descriptors scattered throughout the environment or become immersed in an understanding of the game world, through narrative devices such as audio tapes, propaganda posters, or even menus. Once the understanding of the multi-layered architecture is illustrated using the game Bioshock 2, Chapter 5 and Chapter 6 will include a wider range of games to assess the universality and applied value of the framework guiding feedback-based gameplay metrics.

4.1 Multi-layered deconstruction and reconstruction of gameplay performance

The idea of using a multi-layered structure to capture the structure of a game is not new (Cousins, 2004; Neubauer, 2006). However, work by Cousins (2004) and Neubauer (2006) has been dedicated to achieving *gameplay segmentation* in the way Zagal et al. (2008) conceive of it. While structural analysis of games acknowledges the need for interactions between a player and the game system, the researchers do not ultimately segment how a game is activated during play. Much of this work is dedicated to identify its stable constituents (whether utilised or not).

The multi-layered architecture applied in this thesis has been informed by previous research into segmentation, but differs in both its usefulness and definition, due to its emphasis upon performance and how that is achieved via an automatized approach to data processing. The multi-layered architecture for processing or conceptual framework for interpretation is comprised of five layers:

1- A *game system* layer
2- A *game world instances* layer
3- A *spatial-temporal* layer
4- A *degree of freedom* layer
5- An *interactions* layer.

Hierarchically, at the very top of an account of the events of a game play session, the *game system* layer comprises of all the game mechanisms and gameplay elements contained within a piece of game software. The second layer down determines whether a game has been initiated, creating a *game world instance*. That is what happens after the user presses a play or load menu item. The idea of distinguishing a game world instance in a layer is to acknowledge how a player's performance occurs both within the fictional world of the game, but also in the menus and settings. For example, a change to difficulty levels, when selected by a player can be understood differently, depending on whether a game world instance has already been created. That is, player motivation or reason for altering the difficulty level once play has been instigated is more likely to be a result of challenges encountered. Prior to the instigation of play, the same action from a player may suggest confidence levels and previous experiences as a player. The

spatial-temporal layer accounts for segments of play spent in particular game locations. If the spatial component can refer to space traversed and player progress, it can also account for movement out of the game world, back to menu. The *degree of freedom* layer provides information about the restrictions or freedom offered to the player at any given time to perform actions. For instance, during in-game cut-scenes, the avatar might remain in the same fictional space, but the player is prevented from acting independently for the duration of a clip. Finally, the *interactions* layer describes the *performance* in terms of the actions a player actively performs in response to a scenario or choice.

The main focus of this chapter is the introduction of a five-layer framework for processing play as a performance and interpreting the role of a game component when analysing player experience. The architecture can be used *hierarchically* – for a structural deconstruction of performances – and *transversally*, for an experiential reconstruction of the performance. Both steps allow for an interpretation of player experience. "Hierarchy" is a term that is derived from its definition in computer science and the object-oriented programming paradigm (Bruegge & Dutoit, 2010; Stroustrup, 1997). The idea of *hierarchy* for the multi-layered structure is to acknowledge a parent-child relationship between the layers, in such a way that a *child* (at the bottom) inherits some of the *parent* (or top) properties, allowing each *child* segmentation process to be guided by the *parent*. For the five layers constituting the architecture, the hierarchical ordering from top to bottom is determined as: the *game system* layer, the *game world instances* layer, the *spatial-temporal* layer, the *degree of freedom* layer, and the *interactions* layer. The interactions depend on a given degree of freedom, a degree of freedom

that is only present in specific spaces, themselves depending on the existence, or not, of a game world instance, conditioned by the whole game mechanisms and rules of the game system. Figure 18 illustrates the hierarchical use of the five-layer architecture, emphasizing the deconstruction process (the deeper the layer position, the more deconstructed is the performance).

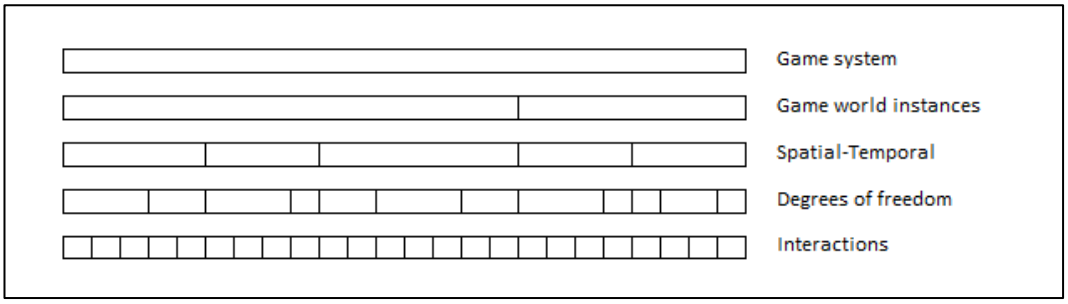


Figure 18. Five-layer architecture used hierarchically

Figure 18 illustrates the hierarchical use of the five-layer architecture introduced in Chapter 4. Each sub-layer segments further the above one, while inheriting several of its properties. The deeper the layer position, the more deconstructed is the segmentation.

In a *transversal* application, each segment can be understood and related to any of the layers. If a player is inside a help menu (spatially), directly after fight sequences (an interaction) that has resulted in several on-screen deaths (altered degree of freedom), then the use of a "help menu" is best interpreted in terms of what came before and also what follows in terms of whether the player eventually overcomes the situation. A transversal consideration of the layers is, then, a way to acknowledge that, at a given time, the player experience encompasses previous experiences and is determined by future decisions. Figure 19 illustrates the transversal use of the five-layer architecture.

Considering the segments transversally also means that there is also potential for understanding segments in terms of what they are not. For instance, when the player is inside a menu, they are in a different space from the game world. Each layer, along with the deconstruction and reconstruction processes, will be

described further below, and illustrated using the game Bioshock 2 as a case study.

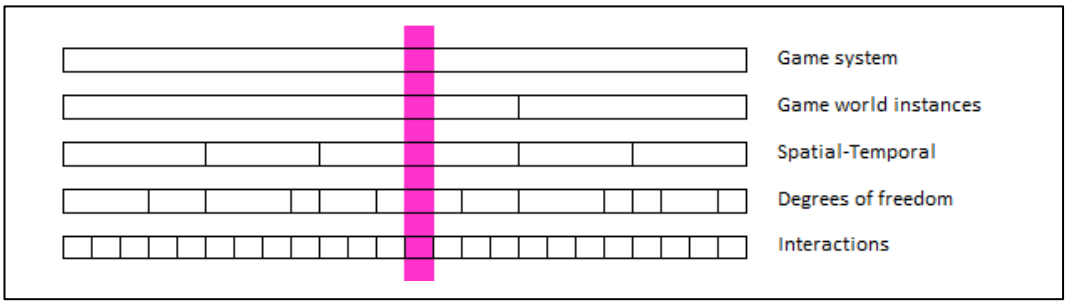


Figure 19. Five-layer architecture, used transversally

Figure 19 illustrates the transversal use of the five-layer architecture presented in Chapter 4. The interaction highlighted by the pink bar can be interpreted by concurrently analyzing any of the other layers.

4.1.1 *Bioshock 2: overview of the game*

Before presenting the five layers that guide a *gameplay performance segmentation*, the game *Bioshock 2* will first be introduced, in order to better understand the different examples related to *Bioshock 2* in the subsequent section.

Bioshock 2 is a First Person Shooter (FPS) body-horror game that is set in the underwater city of Rapture. The player assumes the role of Subject Delta, a Big Daddy prototype. Big Daddies represent genetically modified and enhanced human beings that are locked into Victorian deep-sea diving suits. They are supposed to follow very specific rules, notably protecting other genetically modified characters called Little Sisters. During the opening cut-scene, Subject Delta is shot by Sophia Lamb, the main antagonist of the game, only to be resuscitated ten years later. The player can then, in terms of the narrative, try to discover the reasons behind Subject Delta's death and resuscitation.

The city of Rapture has a long and complex history, incorporating a set of laws and social rules that are an integral part of the game. Rapture was founded by Andrew Ryan after World War II, in an attempt to create an idealistic city cut off from the outside world, free from religion or government control. Art and science are not undermined by any kind of overarching or preordained morality, allowing unconventional research practices to flourish in the area of genetic modification. Indeed, in Rapture, inhabitants can employ stem cells, called ADAM, in order to enhance both body and mind. Most of Rapture's inhabitants, labelled Splicers, have become addicted to ADAM, and have become mentally and physically impaired. Splicers also represent the main adversaries in Bioshock 2. Unstable and aggressive, they attack the player. As hosts for ADAM, the Little Sisters gather and store this precious commodity. It is also why they need to be accompanied by a Big Daddy, who protects them from ADAM-addicted Splicers.

The above description of the Bioshock universe is only a very brief overview of its narrative depth and complexity. Each component described above (e.g., Rapture as a city, Big Daddy, Little Sister, ADAM) can be explored in-depth by players, due to the level of detail given to the world and background story. The impressive attention to detail makes *Bioshock 2* a game that offers a range of experiences, depending on whether the player is interested in the back story, or in the action-oriented nature of the gameplay (or both). Numerous clues exist regarding the pre-history of *Rapture*, its political system and its inhabitants, via different vectors of information. Figure 17 illustrates, for example, how propaganda posters and graffiti litter the spaces of *Rapture*. These artefacts play an important role in informing the players about the world they inhabit. One

interesting feature of the propaganda posters is their intended audience. They serve to address both the inhabitants and the player, making them a clever, seamless and diegetic way to convey information to the player. The conveyed information is seemingly narrative-driven – indicating how the different characters and entities co-exist in the world of *Rapture* – but it is also ludic, indicating how the player might interact efficiently with the characters and entities that they encounter.

In Figure 20, the Little Sister instruction posters (Figure 20a) exist as part of the world of Rapture. Their purpose is to teach the Little Sisters that they should always be accompanied by a Big Daddy (left example) and to gather ADAM from corpses (right example). This is the story of Rapture. But these posters serve a dual purpose, as they explain to the players how they are supposed to relate with Little Sisters, and what their function is, in terms of gameplay. This forms one means of ludic gameplay instruction. In the image on the right hand side of Figure 20a, the "ZZZZ" tag also suggests that the Little Sisters are still innocent. The poster deliberately misleads the viewer, suggesting that Little Sisters are harvesting ADAM from sleeping people, when in fact the victims are dead. This informs the player about the nature of the game as a complex web of manipulation, ideals and propaganda, but also serves as a ludic guide in its revelation that the player will be required to adopt behaviours indicative of being protective (fighting in order to protect).

(a) Little Sisters Instruction Posters

(b) ADAM Advertisements

(c) Wall Inscription

Figure 20. Propaganda in *Bioshock 2*

Figure 20 shows different posters and inscriptions that can be found on the walls of Rapture, used both as an in-game world communication (from inhabitants to inhabitants) and as a game-to-player communication (to inform the players about the role of elements and characters they may encounter). (a) shows the description of the role and personality of Little Sisters and Big Daddies, (b) informs the players about ADAM, and (c) depicts political messages (or slogan).

The ADAM advertisements (Figure 20b) sell the power of ADAM to the inhabitants of Rapture as well as the player. At the same time, the posters fail to declare the addictive nature of ADAM. Finally, several wall inscriptions (Figure 20c) indicate the political unrest within Rapture. The graffiti "Lamb is watching" serves as a more direct warning the inhabitants (and the player), as to the power possessed by the antagonist Sophia Lamb.

Other vectors of information are also used to inform the player. Figure 21 shows the audio diaries that can be collected by the player. The content of the audio diaries are clearly directed to the player. The sound of the inhabitants' voices creates a much more intimate feeling for the player, connecting play in the present to past events of Rapture. The audio diary presented in Figure 21 contains references to the role and impact of genetic modification ("It's your mind that's atrophying") and informs players about how Splicers came into existence. Finally, Figure 22 illustrates the help menu, which players may use to understand the usefulness and origin of Big Daddies (e.g., "those metal diving suits were genetically altered to become the Little Sisters' mindless, lumbering protectors").

In Figure 20 and Figure 21, the information on Rapture is disseminated in a passive way. It is made available for the players to discover as they wander around, allowing them to either ignore or miss it. In contrast, Figure 22 indicates a more active way of engaging with the story, as the players presumably decide to enter the menu to find the information they need. The amount of time spent in such menu spaces would indicate player intent.

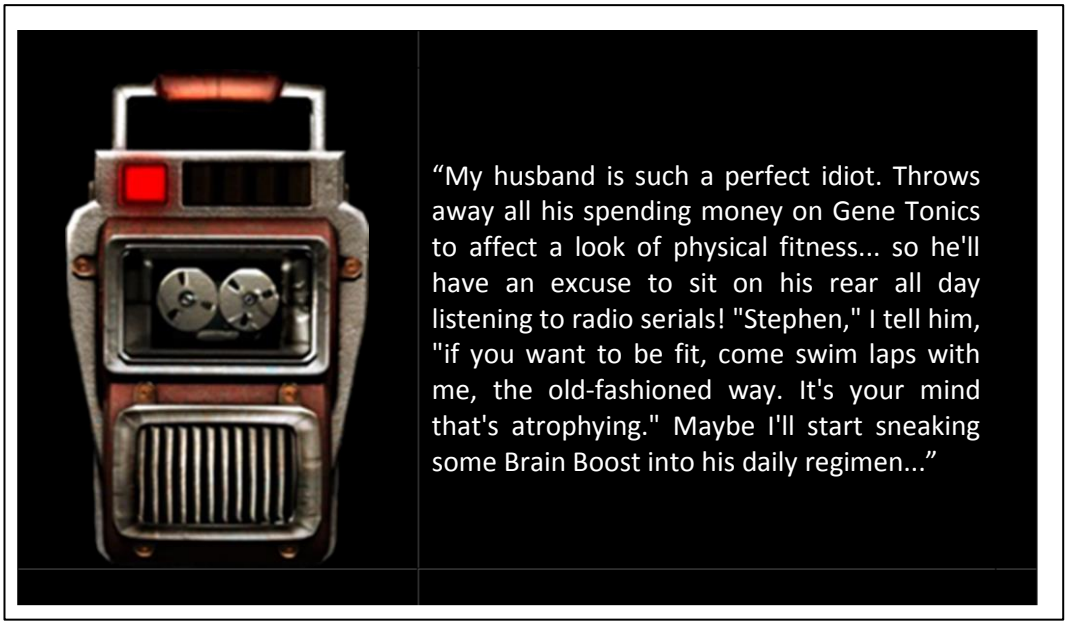


Figure 21. Audio Diary example in *Bioshock 2*, about ADAM

Figure 21 illustrates the use of the audio diary in Bioshock 2. Here, the players are, in an allusive way, informed about the danger of the overuse of ADAM, atrophying the mind. These diaries not only provide a sound way of conveying information, but also create a more intimate feeling for the players, as if they are listening to an actual inhabitant’s voice.

Figure 22. Help menu example in *Bioshock 2*, about Big Daddies

Figure 22 illustrates the use of the help menu to inform the players about the diverse elements and characters of Rapture. Here, the players can learn about the history and current role of Big Daddies. Contrary to the examples in Figures 17 and 18, the players are supposed to actively request such information, as it is only present in the help menu, and not directly in the game world.

On the other hand, players can also choose to ignore the different story elements, and decide instead to experience the game mostly from its action-oriented perspective. Figure 23 shows that the action component of *Bioshock 2* is also core to the game. The enemies are various (another *Big Daddy* in the top image, and a *Splicer* in the bottom one), and the player's weapon is visible. The HUD elements are directed toward fighting sequences; health, remaining power, power in use, ammunition and the enemy's remaining health.

Because *Bioshock 2* offers a large spectrum of different *play experiences*, it has been selected as a relevant game to illustrate how a *gameplay performance segmentation* can be used in conjunction with a multi-layered deconstruction and reconstruction structure. Chapter 4 can now detail each of the different layers. The presentation of the layers in the following sections is organised using the hierarchical structure, from the top root layer (i.e., without a parent) to the bottom one (i.e., without children).

Figure 23. Fight in *Bioshock 2*

Figure 23 shows how Bioshock 2 has also clearly been designed as an action game. Various enemies have been designed (Big Daddy on the top, Splicer on the bottom), as well as different weapons. Furthermore, numerous HUD elements only exist for an action-oriented reason these include the health bar, the power bar, power in use, ammunition, and the enemy's remaining health.

4.1.2 *Layer 1: Game System*

The *game system* layer should be seen as the starting point of any segmentation process, as it represents the first step in identifying the gameplay elements likely to be used as cues to guide the segmentation of the gameplay performance. The game system layer is actually the layer that connect to the nature of gameplay segmentation conducted by Zagal et al. (2008). It constitutes a pre-analysis step, prior to a full gameplay performance segmentation.

Some elements of the Game Ontology Project (Zagal et al., 2005) are particularly relevant to the approach presented in the current chapter. Figure 24 illustrates terms that constitute a Game Ontology, which can be applied to the indexing of segments during a gameplay performance. For example, drilling down from "Interface" through "Presentation ➔ Presentation Software ➔ Sensory Output ➔ Visual Output" to "Health Indicator" contains gameplay elements that can cue a segmentation of *Bioshock 2*. The *Bioshock 2* HUD contains a life-bar that can be used to segment the interaction, based on the principle that the life-bar decreases, say as a result of sustaining damage, increases via health regeneration or remains stable during exploration for example. Similarly, deconstructing from "Interface" through "Presentation ➔ Presentation Software ➔ Feedback" to "Warning" is useful to detect the end of a mission and the beginning of a new one in *Bioshock 2*, as it is indicated through a panel appearing on screen warning the player about the next step to be accomplished.

- **Interface**
  - **Presentation**
    - Cardinality Of Gameworld
      - 1-Dimensional Gameworld
      - 2-Dimensional Gameworld
      - 3-Dimensional Gameworld
    - Presentation Hardware
      - Audio Display Hardware
      - Haptic Display
      - Visual Display Hardware
    - **Presentation Software**
      - **Feedback**
        - **Confirmation**
        - **Warning**
      - Point of View
        - First-Person Point of View
        - Second-Person Point of View
        - Third-Person Point of View
      - **Sensory Output**
        - **Aural Output**
        - Tactile Output
        - **Visual Output**
          - Camera-based World View
            - …
          - **Head Up Display**
            - **Buttonpress Indicator**
            - Control Bindings Display
            - **Health Indicator**
            - **Lives Indicator**
            - **Map Display**
            - Next Piece Indicator
            - Player Configurable Buttons/Keys
            - **Points Indicator**
            - Radar Display
            - **Special Weapon Indicator**
            - **Time Indicator**
            - Vehicular Instrumentation

Figure 24. Feedback-based examples in the Game Ontology

Figure 24 is an updated version of Figure 9, and illustrates, in bold, the different elements of the Game Ontology Project that can be identified and used as cues for further gameplay performance segmentation. For instance, Heath Indication, under Interface/Presentation/Presentation Software/ Sensory Output/Visual Output, can be linked to the life bar in the Bioshock 2 HUD, and can be used to segment the interaction layer, whether the life-bar is decreasing (difficult fight), increasing (health regeneration action), or stable (exploration or controlled-fight).

Highlighting gameplay elements is precisely what the gameplay segmentation and Game Ontology Project are designed to achieve. In practice, this is accomplished by playing the game, but it can also be based on previously accomplished gameplay segmentation of the same game, or even by watching gameplay performance videos, or chronicling the play of others. Obviously, this part of the process of identifying game elements can be time-consuming, but is necessary, especially in order to list the gameplay entities of interest for the gameplay segmentation. The work of a single player-analyst can then be validated or tested against the actions of a wider sample. Identification of the game elements only needs to be done once per game.

4.1.3 *Layer 2: Game world instances*

The second-order layer, labelled *game world instances*, is actually when a segmentation of gameplay performance is initiated. Because of its top level position, the game world instances layer remains general and does not yet account for all the player's specific interactions. However, the distinction of game world instances layer remains essential, as it offers a better understanding of the experience underlying some additional core player actions.

The notion of an *instance*, as used in the current thesis, is derived from a computer sciences definition of instance, as specified by the Object Oriented paradigm (Bruegge & Dutoit, 2010, p. 37). Two core notions represent the foundation of the paradigm: the notion of class, and the notion of instance (or object). A class is the description of an object, like a pattern, in terms of properties (elements defining the object) and behaviour (how to interact with the object). A class is then a full definition of an object, but it represents the idea of an object, rather than an actual object. An instance of a class is an actual object created using the class pattern.

In the majority of games, during the launch process the player is presented with a series of splash screens or legal notices. This occurs before they can interact with a main menu in which they are given the opportunity to adjust different parameters, such as controls, sound, game difficulty, and gameplay mechanisms prior to activating play. The structure of *Bioshock 2* does not differ from this. Before the player takes the decision to press play, or load game, there is actually no existing instance of the game world from the perspective of player experience.

The class of the game world only exists because it has been previously created by game designers. It has not yet been instantiated for the player to progress within and act upon. When the player decides to click on play or load, then a game world instance is created and the player is free to modify the instance created for them with their actions inside the game world. Should other players play the same game on another machine, this would not impact on the current play. That is, each player possesses and experiences their own instance of the game world, even if the game world class is identical.[10] When no game world instance exists, it is possible to consider the game world as being only a source code entity, the exact form as conceived by designers, in an untouched state. But once the game world is instantiated, the game world is altered by the player's actions, and is then moulded by their interactions.

It is possible to relate the concept of game instance to the concept of the *magic circle*, formalised for all types of games by Huizinga (1955) who observed play-elements within cultures, and described how the "ritual" of play (1955, p. 8) occurs in "consecrated spots" (1955, p. 10) like "the arena, […] the card table, […] the screen, the tennis court" (1955, p. 10). The magic circle has subsequently been applied to videogames by Salen and Zimmerman (2004), who state that:

> The magic circle of a game is where the game takes place. To play a game means entering into the magic circle, or perhaps creating one as a game begins. (2004, p. 95)

[10] This is different for multiplayer games, and notably massively multiplayer online (MMO) games, but MMOs are beyond the scope of the current thesis.

The question can be raised as to whether the magic circle would actually include elements like the main menu or splash screen, which the definition of instance tends to exclude. However, the definition of magic circle is large enough to encompass the use of an instance in the present thesis, given that it refers to the actual moment when a game world instance is created for the player.

There remains a question about the usefulness of a layer that distinguishes between an instantiated and non-yet-instantiated game world. Three major benefits can be derived from considering the existence (or not) of a game world instance during a performance. Firstly it is meaningful, given the aim of the present thesis to point out that some elements of the game or player experience only occur when no game world instance exists, while some others can exist only when a game world has been instantiated. For instance, this is the case with a *legal notice* screen, *splash* screen or *main menu,* which typically appear before the main gameplay begins or is about to begin. From a processing perspective, which seeks to foreground key moments during play, it would be unnecessary to initiate any detection of in-game actions (e.g., fighting, running, looking at environmental elements), during activities that sit outside the game world.

Secondly, and more crucial for the present research project, is the manner in which the player experience assessment is different whether or not the player is conscious of the existence of a personal game world instance. To illustrate this, should a player change key bindings, it is possible to assume the intention that underpins this customisation action. Should this occur before the existence of any

instance, the customisation process can be seen as prompt by a general experience. The player is probably changing the key bindings to fit their favoured control style that they have employed in similar games. Should it occur during an instance, the customisation might be a response to the instance (such as a specific event that forces the player to adapt the instance). That is, it is possible to arrive at a different interpretation, based on the timing and context of a particular customisation.

When accomplished before the creation of a game world instance, a customisation can be understood in the following way as 1) a desire to understand the upcoming game world and see the different possibilities on offer within the menu, as 2) a prior knowledge of the genre, allowing the player to adapt the upcoming game world to suit their previously established playing style, or as 3) a perceived knowledge of a game within the franchise. Once a game world instance exists, the player is then more likely to be adapting or adjusting to the playing conditions of the game world. In *Bioshock 2*, for example, the player can select (and modify) the game difficulty level. Before any game world instance exists, the player is able to make an arbitrary choice as to how difficult they wish the game to be. The choice is more likely to be based on evaluation of experience with other First Person Shooter games, including prior games in the *Bioshock* series. To guide the player in this particular choice, game designers provide advice based on experience with the FPS genre. Figure 25 illustrates how the game makers categorise *easy*, *medium* and *hard*. The choice of difficulty, made without any prior instance of the game world, is principally an anticipatory action.

Figure 25. Difficulty selection menu in *Bioshock 2*

Figure 25 shows how game designers use previous play experiences to guide players during initial choices, when players cannot rely on any game world instance: "Easy - you're new to shooters; Medium - you've played other shooters; Hard – you've played a lot of shooters.

Finally, the concept of *instance* is essential because of the manner in which it raises the question as to what constitutes the duration of a *game world instance*. From a computer science perspective, the existence of an instance would be connected to its actual existence within computer memory. While the computer science perspective makes sense, its value in terms of a player experience perspective – as to what an instance of gameplay represents – is limited. Indeed, should an instance be defined by memory, it would mean that each loading screen – that is, moments when different game world sections are loaded into memory – would suggest a new instance, as it erases the previous one. This would result in a game world instance being determined by how the game implements its loading management system. It is reasonable to question this from the perspective of the player, as the appearance of a loading screen between two consecutive moments of play is not necessarily perceived as a "reset" of the game world. What the player has accomplished before is still valid after the loading. For the player, the game world remains the same, even if the experience is interrupted by several loading screens. Loading may represent a familiar game-like device, comparable to the snake in snakes and ladders, yet still part of the consecutive section of the play. In terms of the *magic circle*, the player remains inside the *magic circle*.

This discussion translates methodologically into the decision that a loading screen between two sequences of play need not be considered an instance break. However, a distinction can be made for other types of loading screens, which can actually overwrite the existing game world instance. This, for example, can be found in the case of a consciously and player-triggered *load*, which enables the

player to return to a previous state in the gameplay. In this case, the instance is deleted.

The possibility of using the saving/loading system to erase and replace an instance can form a player strategy. If a player is about to die, they know reloading will only cause them to lose a small portion of their progress. One participant in the *Bioshock 2* sessions (see Section 2.3) regularly used the loading system to execute a perfect performance. Every time they instigated an action which they could have done better, they loaded the game at a prior point in order to over-write their imperfect performance. This example also justifies the need for a separation of an existing vs a non-existing game world instance. When no instance exists, loading a previously saved game represents the desire to continue the previous play, a loading executed during the play represents a wish to roll back the game world to a previous state, thus indicating the desire for a retry.

"Death screens" present a challenge as to whether they constitute an instance that is overwritten or not. This applies to *Bioshock 2*, in which there is actually no "reset" after a death. Instead, a respawn has been designed as part of the game world and narrative. Subject Delta just reappears in the closest Vita Chamber. In Rapture, Vita Chambers are devices that resuscitate inhabitants dying next to them. Vita Chambers are part of the history of Rapture and, when used, give to the player a small amount of energy. They can also be used indefinitely. When the player dies, he/she remains inside the *magic circle*. There is a sense of continuity, despite the player's death. Indeed game entities, like enemies, continue to exist during the respawn process, responding to the screen death by moving toward the

Vita Chamber, and anticipating the emergence of the player's avatar. On the other hand, in games such as *Battlefield 3* and *Max Payne 3*, a death constitutes a retry and repeat of the same narrative until a solution is found. The game then restarts at the last save point. Death represents a moment that ends an instance, requiring the creation of a new one. In terms of the *magic circle*, this can be seen as exiting it and re-entering it again.

To summarise, a game world instance is created when a player chooses to start a new game or to load a previous one. In terms of the magic circle, this represents the moment when the player enters the magic circle. An instance ends when the player quits the game (goes back to main menu, or back to the system), exiting the magic circle and play session. An instance also ends when a new instance overwrites the previous one, breaking the continuity of the game world. The notion of whether or not the game world instance exists improves the understanding of player actions in terms of player experience, for example illustrating anticipation, adaptation, strategy, performance and consequences. Instance also provides a context for the sub-layers of the hierarchical model, as some segments can be defined only by whether instance exists (e.g., in a cut scene, or in-game), or not (e.g., splash screen, main menu).

4.1.4 *Layer 3: Spatial and Temporal*

While the game world instance layer separates the gameplay performance by considering whether a game world instance actually exists or not, this remains a general and coarse-grained segmentation, useful to initiate a first step of gameplay performance segmentation. Each segment contains a diversity of spaces and times that need to be distinguished in the context of finer-grained layer of segmentation.

When Zagal et al. use the term "spatial segmentation" (2008), they refer to spatial changes from within the game world, that divide the game world "into different spaces that also partition gameplay" (2008, p. 182). Their definition implies, for instance, that a checkpoint – indicative of a spatial respawn point after a death – would be considered part of the spatial segmentation even if, during the play, no discontinuity was perceived by the player. The application of *spatial* dimension to a segmentation, as presented in the current thesis, is potentially more radical, as it represents spatial-temporal consistency between two consecutive moments. That is, it refers to strong changes of experientially distinct spaces, such as from the game world to a game menu or loading screens. This is different from assessing the spatial changes from within the game world, such as a change of rooms, or from the exterior to the interior, etc. The consequence of this definition of space employed in this thesis is that when it is in total continuity with the game actions an in-game cut-scene will be understood as a change of degree of freedom, rather than a change of space.

Figure 26 shows a still, taken from an in-game cut scene in *Bioshock 2*, which represents when the avatar acquires its first power. In this moment, player agency

and interactivity is at its most limited, as indicated by the absence of HUD and the "letterbox" screen dimension effect. However, full interactivity is soon given back to the player, as indicated by the presence of HUD. The space remains the same, yet during the cut-scene, the player is being invited to pay more attention to the story, as well as to its ludic consequences. For a change of space to induce a new segment, a strong change of space is required, such as movement from the in-game space to the help menu, or two in-game spaces separated by a clear loading screen. Figure 27 illustrates when a player has consciously chosen to enter the help menu, in order to examine a map that might improve navigation. In this moment, the design totally changes (no more first person view, no weapon, no HUD, but a brown panel instead), and the diegetic game world sounds are no longer heard. The two spaces are clearly distinct here. The rationale behind the detection of a change of space is twofold. Firstly, a space prescribes what interactions are possible. Secondly, a space is also indicative of a specific experience in a game. For instance, inside the help menu, the player will act in a more strategically-minded manner, improving his/her knowledge of the game in order to facilitate success.

Figure 26. Injection in-game cut scene in *Bioshock 2*

Figure 26 shows the injection cut scene in Bioshock 2, when the avatar acquires its first power. The top left corner image is the player going to grab the new power. At this point, the player is in full control of the avatar. The top right image shows the avatar taking the power in order to inject it into itself. Then, the player cannot interact nor move (as also indicated by the absence of HUD and the letterbox effect). When the player gets back the full control (in the bottom image), he/she also has a new gameplay element: electro bolt power. The Figure 26 example shows that, in the very same space, different degrees of freedom can exist, justifying the need for a sub-layer segmentation, in terms of the degree of freedom offered to the player.

Figure 27. Player entering the help menu in *Bioshock 2*

Figure 27 illustrates an actual space change as defined in the present thesis. On Figure 26, the player is still in the same space, as no clear break seems to remove the avatar from one place to another. Even if the controls are different, on Figure 26 the avatar remains in the same space. However, in Figure 27, a clear change of space is highlighted. There is a change of perspective, no HUD, brown panel, game world in pause, interface with buttons (back button for example in the bottom), and the sound of the game world is muted. when the player enters the help menu to read a map.

Obviously, different spaces are identifiable in any game and *Bioshock 2* is no exception. In *Bioshock 2,* the following spaces can be identified: legal disclaimers, splash screens, main menu, loading screen, main cut-scene, help menu, pause menu, vending machine interface, and actual game world. During the Legal Information Screens, the player is presented with information related to the creation of the game, the copyright-holders, and the conditions of use. These sequences graphically reflect their formal nature. Here, the player is considered more as a consumer rather than a player, and is not yet part of the game experience. The player's only possible action is to skip these sequences. During splash screens, the aim is still to credit the developers, publishers and creators. Some splash screens sometimes also contain elements from the game, building the player's anticipatory feelings. The player's only possible action is again to 'skip'.

In the main menu, the player is offered a greater range of interaction possibilities, as he/she can configure the game in line with his/her needs, prior knowledge of the game, or of the game genre. The player is invited to prepare him/herself before entering the game world. During loading screens, the player is offered a limited amount of actions as he/she can browse advice. During the cut-scene, the player is presented with a video sequence, with actions taking place in the game world. During such sequences, the backstory of the game begins to unfold and context is presented to the player in terms of location, time period, and characters. The player's sole authorised action is to skip.

Once the game is activated, the player can move, look at the environment, interact with objects, communicate with NPCs, listen to audio-diaries or fight against

enemies. During some play sequences, part of the interaction is removed, notably when the player is underwater and cannot fight, or during short in-game cut-scenes, during which the avatar's gaze forces the player to pay attention to a specific element in the game world. The diversity of degree of interaction within in-game segments is part of the justification for the final two layers.

In the help menu, the player can obtain information about the game world he/she is progressing in, and about the different actions expected from him/her in the shape of the aims list. During this time, the authorised interaction is a menu navigational one. In the vending machine menu, the player is invited to (strategically) reflect about what items he/she would need to accomplish his/her mission. The player can make more or less effective choices, also depending on the amount of money he/she possesses. The interaction is limited to buying and selling items, and to reading their descriptions. In the pause menu, the player can configure the game in a way similar to that of the main menu segment, but he/she is also offered information about the money he/she owns, the ADAM he/she owns, and he/she has the option to save the game.

The spatial and temporal layer produces a good sense of what the player is currently experiencing, because each space carries a meaning in terms of what the player can be invited to do, or not to do, at a meta-level. The layers, up until this point define, coherent sections of play for analysis and situate play. The subsequent layers account for player behaviours in the game world itself.

4.1.5 *Layer 4: Degree of freedom*

Now that the continuity or disruption of spaces has been ascertained, it becomes possible to focus on the different *degree of freedom* offered to the player during a gameplay performance. The intention is to segment a play performance in relation to the degree of changes that relate to what actions the player is able to perform, or whether the player is invited by the game to execute a specific interaction. In order to be able to detect specific interactions, it is important to know what kind of interactions are actually authorised by the game at a specific instant. This knowledge not only improves automatic processing – as there is no need to assess the execution of an action if the context clearly defines the action as unavailable – but it also enables the action to be contextualised. Moreover, when a set of authorised actions is known, it is possible to identify when a player fails to use a specific action, and try to assess the rationale behind this choice. Degrees of freedom are essential in order to assess the motivation of the player and the nature of their actions.

Zagal et al. (2008) use the term "temporal segmentation" to describe "who plays, when", or to "time limits or periods of gameplay" (p.179). They also distinguish a second form of "temporal segmentation", that refers to when "the player is allotted a specific amount of time to complete a certain task, fulfil a goal, or simply do the best he or she can" (Zagal et al., 2008, p. 180) . This clearly describes a very specific gameplay mechanism under which the player is pressured to accomplish a task. If the "time pressure" aspect is removed from the definition of Zagal et al., and replaced by the notion of the time frame in which a certain task, or goal, can be fulfilled, the concept of "time segmentation" can

actually be integrated into the definition of the degree of freedom layer, meaning the moment in which specific interactions are allowed. “Challenge segmentation” (Zagal et al., 2008) also sits within the degree of freedom layer if, instead of understanding a challenge as composed of “puzzles, boss challenges and waves” as proposed by Zagal et al. (2008, p. 187), challenge is understood as a “time frame in which the player can execute different interactions”.

It is noteworthy that in the degree of freedom layer, segments can actually overlap. Indeed, in *Bioshock 2*, the looting-possibilities exists concurrently with fighting-possibilities (the player can still fight or move, even if invited by the game system to loot because they are looking at a corpse). This is not problematic as long as each interaction in the following interactions layer can be linked to a degree of freedom layer, as it represents player response to the possibilities provided within a game space.

Examples of the different degrees of freedom in the in-game space can be drawn from *Bioshock 2*. No restriction on behaviours, beyond those defined by the environment in which the player finds him/herself is considered *full-interactivity*. That is the player can loot, fight, move, listen to audio-tapes, etc. *Semi-interactive* or *exploratory* refers, in *Bioshock 2*, to situations in which the player can move freely and explore, but not engage in fights. One scenario in which the degree of freedom is completely *restricted* is during in-game cut-scenes, wherein players can only pause the game, but cannot move. While operating within full-interactivity, the player is invited to accomplish specific actions, such as looting a corpse (a panel appears with instructions that the players can either follow or

ignore), or fight a wave of enemies (a timer then appears to cue the player fight). Each death in *Bioshock 2* can also be categorised as a change of *degree of freedom*, as the player loses all agency in that moment. Similarly, in the pause menu, each sub-menu represents a different *degree of freedom*. The player can only save in the save sub-menu, load in the load sub-menu, change the difficulty in the gameplay settings sub-menu, and so forth.

4.1.6 *Layer 5: Interactions*

Finally, the last layer notes the actual player actions and *interactions* occurring under the degrees of freedom offered to the player by the game. It becomes possible, then, in *Bioshock 2*, to detect: 1) if the player grabs object (this is indicated by detecting the presence of the looting panel, which would constitute a detection within the *degree of freedom* layer), 2) if the player is listening to an audio-tape, 3) if a new quest is made available to the player, 4) if the player changes menu pages, 5) if the player is engaged in a fighting sequence, 6) if the player is using his weapon or power, 7) if the player is changing the game difficulty, and 8) if the player is loading or saving the game. Interactions can thus be either active (that is, the player is currently performing the action), or passive (that is, the player is informed of an on-going action within the game world). This could mean a tape that is played in the background, or a game state like a pop-up panel that gives a new quest to the player.

Now that the layers have been presented, it is important to focus on the way they will be used to deconstruct and reconstruct a performance of play.

4.2 A Full Gameplay Performance Segmentation

Section 4.2 introduces a *gameplay performance segmentation* as a step-by-step process, moving from a recorded play session (as described in Chapter 2), to a summary of the gameplay performance, including both a structural deconstruction of the performance, and an assessment of the player experience. Section 4.2 describes each step in order to better understand the context in which the gameplay performance segmentation takes place. Prior to a description of what ultimately constitutes the research process of the present thesis, Figure 28 schematises and describes the full gameplay performance segmentation process, starting with a game session that includes various data and actors (including, player, game, biometry, controller, and so on). The initial focus is on the performance elements (the player and the game), and then on the game system itself (the generator of gameplay metrics). The performance is then seen as embedded inside the recorded footage (a vector of feedback-based gameplay metrics). The footage can be automatically deconstructed in terms of coherent gameplay sequences, and reconstructed in terms of player experience. Finally, a summary of the performance is produced, that includes both the structural deconstruction and the experiential reconstruction.

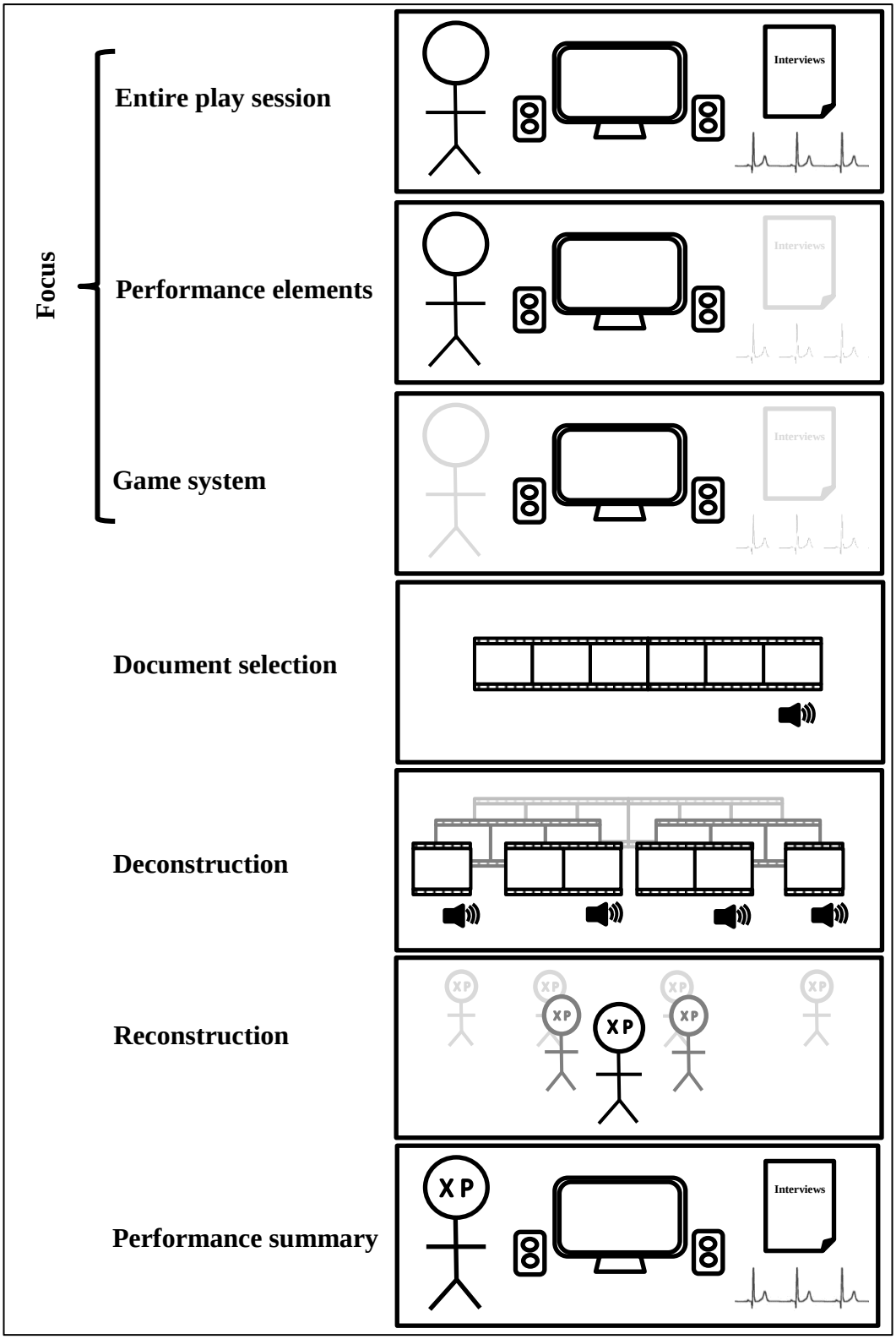
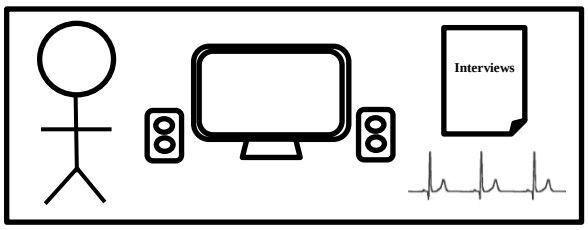


Figure 28 Step-by-step approach contextualising the gameplay performance segmentation

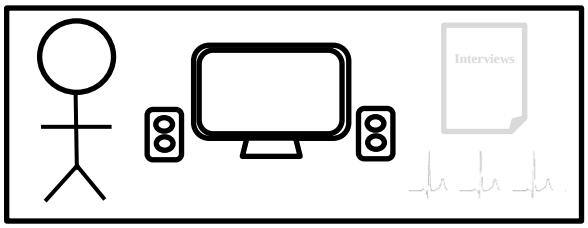
Figure 28 schematizes and contextualises the full gameplay performance segmentation process. Starting from a game session, the focus is on the performance elements (player and game), then the game (gameplay metrics). The game is then seen through the recorded footage document (feedback-based gameplay metrics). The footage can be deconstructed in terms of coherent gameplay sequences, and reconstructed in terms of player experience. Finally, the game session can be considered as updated via the outcomes in the performance summary.

4.2.1 *Initial focus*

The goal is to identify the specific elements that relate to the conceptual model presented in Section 4.1.



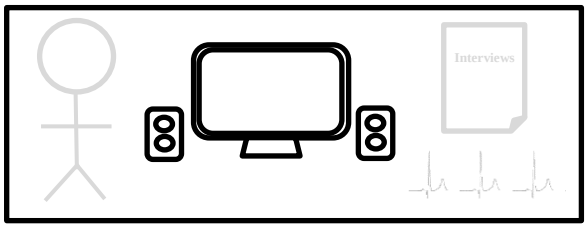
While a single game play session can produce a range of data from different measures[11], it is first necessary to determine salient informational sources that can later be combined with other metrics. The initial *focus* is on attaining a set of *performance* elements that define the experience of the player with the game system, via inputs.



The *focus* then shifts to the game system. As mentioned by Nacke (2009) and presented in Chapter 2, gameplay metrics is quantitative data produced by the game system that is useful for the understanding of the interaction between the game system and the player.

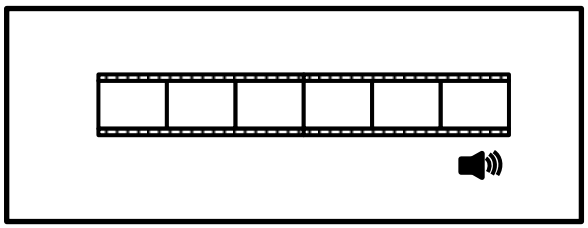
[11] During play sessions, the audio-visual feedback and communication of the game and the player's performance within it was captured using the video capture software FRAPS. However, during sessions additional measures were also taken (biometry, interviews, eye tracking). The focus of this chapter however, is on processing of the data file containing the game play footage.

The player is not discarded, but is “seen through the eyes” of the game system in terms of how he/she interacts with it. Once the focus is on the game system, it is necessary to determine the way in which the interactions between the player and the game system will be analysed.



4.2.2 *Document selection*

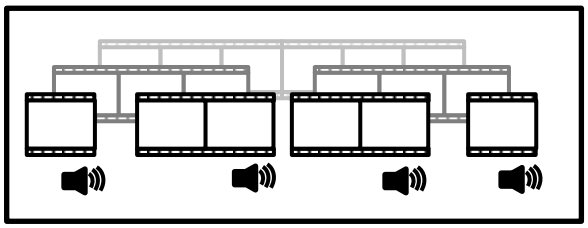
The next step is to elicit a document, representative of a gameplay performance, which can be segmented and indexed (see Chapter 3). Typically for a gameplay metrics approach, the equivalent of a document would be the game-source code (or mod) that directly logs in real-time the different metrics, saved or sent (in the case of telemetry) for further processing (Drachen et al., 2013). In the present thesis, the feedback-based gameplay metrics exploits the audio-visual feedback broadcast to the player.



This is captured (via the screen recorder software FRAPS (Beepa, 2013)) as play unfolds, with no processing. Only once a play session is completed is the audio-visual file processed. Once a video file has been produced, it is also possible to initiate an open-system indexing based on metadata describing the game session. Appendix A provides more details about this possibility

4.2.3 *Performance deconstruction*

Because the documents that are being segmented and indexed are audio-visual recordings, the gameplay performance deconstruction process presented in the current section is tied specifically to the deconstruction of a video file that holds a record of the performance.



In the following paragraphs, the description of the deconstruction process – using *Bioshock 2* – will refer to audio-visual cues, even though a deconstruction process can also be initiated using other forms of data that represent a performance.

Game system

As described in Section 4.1.2, the root layer in the deconstruction process is the full *game system* layer. In terms of deconstruction and video file processing, the *game system layer* represents an untouched and unsegmented video, in its original state, and in the case of audio-visual segmentation, it links gameplay entities with their audio-visual properties.

For the deconstruction process, the game system layer is then the entry point, the untouched video file; this should be used to identify all the useful gameplay mechanisms and elements, as well as the matching audio-visual cues linked to gameplay entities. The *game system* layer is where the different labels used to index the segment in each sub-layer are determined. The pathway outlined below represents a gameplay segmentation as distinct from a gameplay performance

segmentation. This is the structure that pre-exists the player's activation of the game representing the constituents that a player will come to encounter during a game world instance.

In the case of *Bioshock 2*, the terms have been determined as follows: 1) for the *game world instances* layer: no instance or instance, 2) for the *spatial-temporal* layer, main menu, loading screen, in-game, help menu, vending machine or pause menu, 3) for the *degree of freedom* layer, full-interactivity, exploration, in-game cut-scene, cut-scene, loot available, load/save screen, settings screen, enemy wave or death, 4) for the *interaction* layer, enemy hit, health, power, object collected, radio communication, audio diary, goal pop-up, difficulty changed to easy or difficulty changed to hard.

- No instance
    - Legal information
    - Splash screen
    - Main menu
- Instance
    - Loading screen
    - Cut-scene
    - Help Menu
    - Pause menu
        - Load
        - Save
        - Settings
            - Change to easy
            - Change to medium
    - Vending machine
    - In-game
        - Full interactivity
            - Radio communication
            - Tape listened to
            - Health
            - Power
            - Goal pop-up
            - Loot available
                - Object collected
            - Enemy wave
        - Exploration
            - Goal pop-up
            - Radio communication
        - In-game cut-scene
        - Death

Note: many more terms can be added (such as a change in the control bindings) but for the clarity of the explanation, only a sub-set has been extracted.

Game World Instances

The deconstruction based on the *game world instance* layer focuses on the detection of cues that acknowledge the creation (*start game, load*), or destruction (*quit*) of a game world instance. In *Bioshock 2*, a new instance will be determined by a load menu or a new game menu, immediately followed by a loading screen (indicative of the player validation). The conclusion of an instance is indicated by a *quit menu* immediately followed by a loading screen. Figure 29 and Figure 30 illustrate the sequence of screens that indicate a new instance, or the end of an instance, in *Bioshock* 2. The loading screen is a core visual cue for processing, although it is necessary for it to be preceded by a new game screen or a load game screen. If not, the loading screen operates as a pause between two different levels.

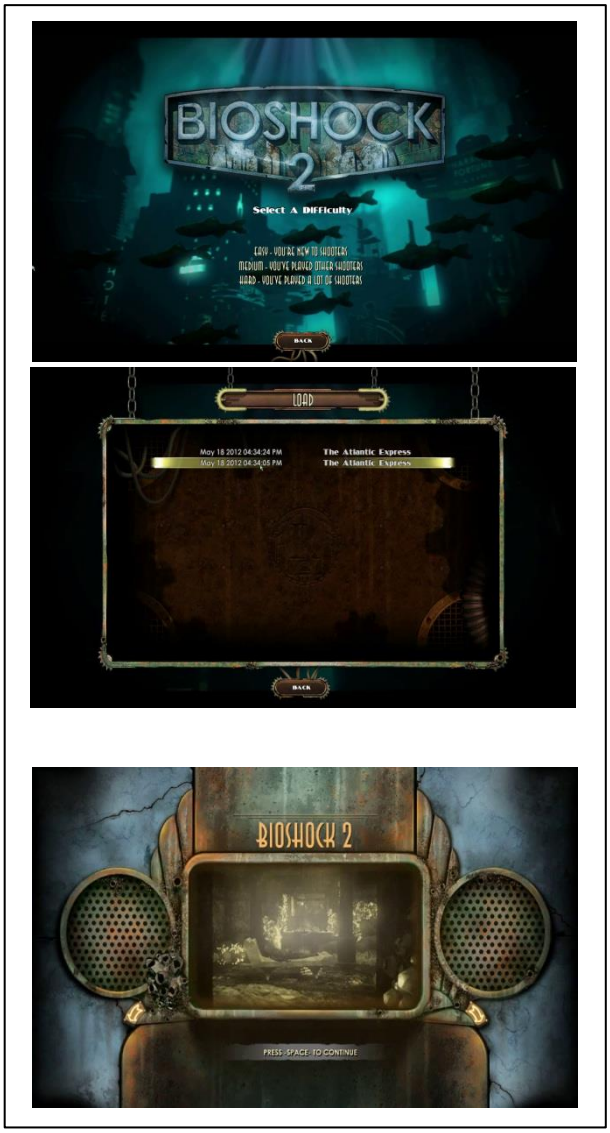


Figure 29. New game and load menu followed by a loading screen

Figure 29 displays the creation of a new game world instance in Bioshock 2. When the players are in a "new game" or "load game" screen, and then inside a "loading waiting screen" (indicative of the players validation of "new game" or "load game"), then a new game world instance is created for the player to evolve in.

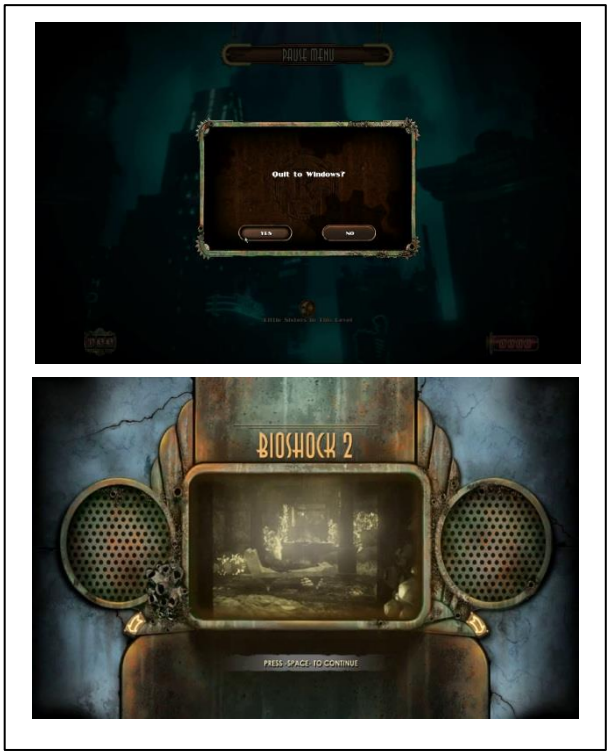


Figure 30. Quit game followed by a loading screen

Figure 30 displays the end of a game world instance in Bioshock 2. When the players are in a "quit game" and then inside a "loading waiting screen" (indicative of the players validation of "quit game"), then the game world instance is erased (until the player decides to load it again later).

Spatial-Temporal layer

The concept of time and space in its simplest form serves to assess the homogeneity of a performance segment. While some spaces are clearly defined by their distinct contribution to gameplay (for example, a help menu is a space outside the game's virtual world, with a clear gameplay goal of providing useful information about the game rules and the back story), other spaces within the in-game experience are audio-visually coherent, but cannot be distinguished by their distinct gameplay contribution as they contain a large variety of gameplay experiences. Figure 31 illustrates the variety of experiences offered by the in-game space of *Bioshock 2*. Whilst in the game space, players can, most of the time; utilise the interaction freedom afforded to them by the system (that is, they can walk, fight, observe, or go into the menu). This full interaction state is indicated by the presence of the HUD. This is distinct from moments in which players have their agency curbed, which are indicated by the complete absence of HUD on screen, or as in exploration sequences, distinguished by a different HUD (when a player is able to walk, observe or go into menu, but not to fight). The space is the same, so a chain of these different interaction possibilities constitutes the same segment at the spatial-temporal level. It is in the degree of freedom layer, that further distinctions are made that contextualise play.

Figure 31. Different degree of freedom during in-game segment

Figure 31 illustrates the variety of gameplay possibilities that can be offered from within the same spatial-temporal segment. During in-game, the players can do most of the actions (fighting, moving, observing, going into menu), indicated by the full HUD presence. During a cut-scene, the players have a very restricted set of actions (only going into menu), indicated by no-HUD. During exploration, the player can perform a limited set of actions (moving, observing, going into menu, but not fighting), as indicated by a limited HUD.

Zagal et al. (2008), define the coherence of space by identifying “transitional screens” (p.183) that describe movement between distinctive game spaces. An in-game segment is easier to detect when it occurs between two strong loading screen breaks, or when a cut-scene divides two spaces. But the scope of a segment that indicates a coherent space can be extended by referring to a game’s inner coherence. Figure 32 and Figure 33 illustrate a spatial coherence between two consecutive moments. In Figure 32, the two frames appear to be distinct spaces that might be processed as separate spaces. The rationale for separating them appears strong, as the HUD is present in one and absent in the second one. The two images also display different game moments, as there is the player climbing stairs in the one, and there is a Little Sister gathering ADAM in the other. However, Figure 33 illustrates that, by considering intermediate frames, the two frames that appeared as distinct spaces in Figure 32 can actually be considered as being part of the very same space segment. The player is climbing the stairs in order to go to the next room, which contains the Little Sister gathering ADAM. The change here actually describes a different degree of freedom, as agency is removed from the player. Likewise, sound or music can also be used to indicate spatial coherence, persisting across a sequence of movement, or a break that introduces a sudden change in the sonic atmosphere of the game. The assessment of coherence, across two consecutive moments, follows the notion of coherence established in Section 3.5.2.

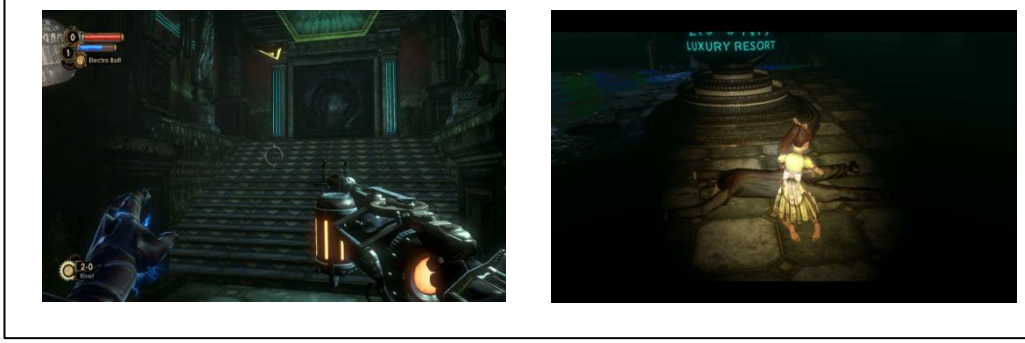


Figure 32. Two frames in *Bioshock 2* appearing visually non-coherent.

Figure 32 shows two frames in Bioshock 2, which appear visually non-coherent. The HUD is present in the first frame, and absent in the second one. The second one has a letterbox effect. The displayed topics are different, too: the player is climbing stairs in the first one, and a Little Sister is gathering ADAM from a corpse in the second one.

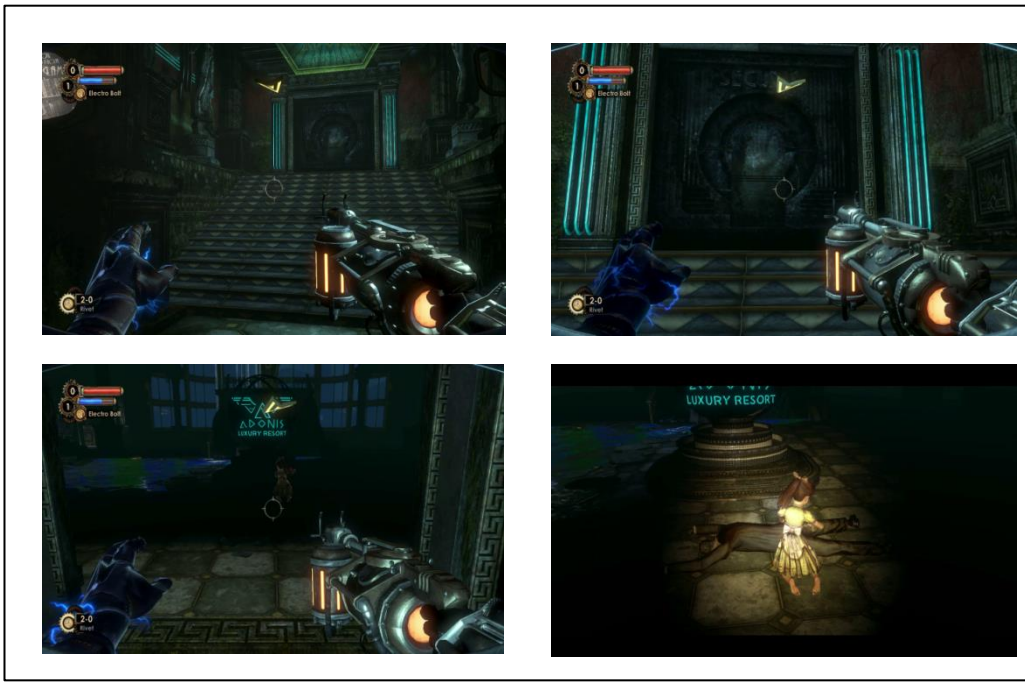


Figure 33. The two same frames of Figure 32 appearing more coherent with the addition of intermediate frames.

Figure 33 illustrates how the two apparently distinct frames of Figure 32 are actually coherent in terms of space. The player is climbing the stairs in order to reach a new room, in which a Little Sister is gathering ADAM. The absence of HUD in the last frame is then more indicative of a change of degree of freedom (the player can no longer move) than a change of space.

It is important to note that some spatial-temporal segments can only be found under a specific situation, which helps to determine a distinct spatial-temporal segment. For example, the main menu is only detectable before a new game or load (no game world instance). Loading screen, in-game, help menu, pause menu and vending machine, on the contrary, can only exist when a game world instance has been created.

In *Bioshock 2*, a similarity exists between the main menu and the pause menu. By identifying the parent layer, it is possible to distinguish between them. For example, the player knows that exiting the *pause menu* will bring them back into the game world, as an instance already exists. But during the *main menu*, an instance is yet to be created, so the player knows that exiting the main menu will actually end the engagement with the game software. Furthermore, information relating to player progression, such as amassed money and *ADAM*, only appears in the pause menu, as it represents a game world state that cannot exist without the creation of a game world instance.

Assessing spatial coherence is complex when only the gameplay mechanisms are considered. The use of a video file as a performance document paves the way for an audio-visual assessment of space and time, one that directly reflects a player's progress and experience. However, assessing spatial consistency via audio-visual representations is not a straightforward process. Figure 34, for example, shows how the main menu in Bioshock 2 is actually composed of several design features, matching the player's choice of item. In Figure 34 for instance, the graphic option menu item could be considered as another segment based upon audio-visual

coherence, due to the large brown panel that is superimposed. However, several other audio-visual properties make the process of separating the graphic option menu item difficult to justify. First, the music and sound atmosphere remains coherent, as if nothing has changed on screen. Secondly, the background remains consistent and active, even if mainly overlaid by a brown panel.

Figure 34. Different visual designs in the main menu

Figure 34 displays the diversity of graphic designs in the same main menu space. However, the musical atmosphere remains constant (changing menu does not change or stop the music), and the background is still present.

It is tempting to consider the main menu and the pause menu identical, as both share a number of audio-visual similarities. Some elements do distinguish them, however. For instance, as already discussed above, the presence and inclusion of information on resources possessed by the player would not occur in the main menu, as no game instance exists at that point. Here, audio-visual coherence alone is not enough, and gameplay information is required to assess where this information occurs in a player's progress.

Degree of freedom layer

The *degree of freedom layer* is the determination of a segment of play wherein the interaction possibilities of the player are determined by the system. In general, the player is made aware of such a change of possibilities by the presence of gameplay cues. In *Bioshock 2*, the *in-game* segment (spatial-temporal layer) can be further segmented as: 1) fully interactive: the HUD is on screen, and the player can fight, 2) exploratory, showing an altered HUD on screen, or the player situated underwater, breathing heavily, and unable to fight, 3) in-game cut-scene, indicated by a letterboxing effect, or player's inability to control the avatar, 4) death; the screen turns blue and an agonising sound is emitted by the avatar, 5) enemy wave, in which a timer appears on screen, showing how long the wave will last and 6) looting, in which the player is presented with a panel inviting him/her to loot a corpse. Another example is the load, save or gameplay setting sub-menus, constituting different degrees of interaction within the pause menu segment.

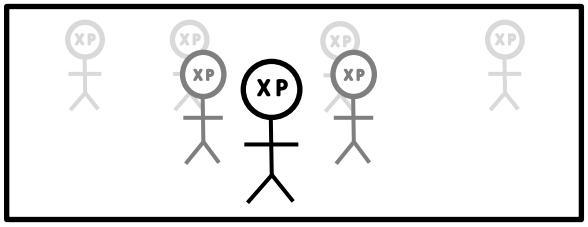
It is important to note that, in contrast to the above presented layer, the degree of freedom layer can actually have overlapping segments. This is the case, for example, of the looting cues, as a looting panel appears on screen, but this interaction can co-exist with a player being fully interactive or exploratory.

Interaction layer

In *Bioshock* 2, segments in the *interaction* layer represent different forms of feedback relating to player interactions (e.g., object looted, an audio tape played), cues from the game world (e.g., new goal or radio communication), or avatar state (e.g., health, power). In terms of hierarchy, it is interesting to note that health and power information, for example, only exist when the player is in a full interaction mode, and so when the HUD is on screen.

4.2.4 *Player experience reconstruction*

Having illustrated how the different layers introduced in Section 4.1 can be used hierarchically – in order to deconstruct a gameplay performance – and summarise the performance in terms of player state and interaction, this section demonstrates how the same multi-layered structure can be used transversally to reconstruct a gameplay performance and summarise the performance in terms of player experience.



The layers are now used for their conceptual properties, guiding the understanding of player experience. The analysis of player experience is an indexing process (closed-system), as it involves extracting a sequence of a performance and describing it in terms of player experience. Furthermore, as the description, or labelling, is performed by the indexer using an analytic vocabulary. Based on a transversal consideration of the five layers, the performances with *Bioshock* 2 are reconstructed in terms of contextualisation, temporal frame of references, semantic network, loop and comparison. Chapter 5 will go on to outline the automatic audio-visual processing techniques that generate accounts of player experience. However, further examples in which the feedback-based gameplay metrics system was tested with participants will be presented in Chapter 6.

Contextualisation

One way of using the five layers transversally is by the detection of interval boundaries in the deconstructed performance, creating a *context* for understanding the segments that appear inside the interval. Notions of context and interval are

linked to the idea of topic boundaries (Section 3.5.4), referring to the determination of changes that can elicit a different experience within play. In one play performance with *Bioshock 2*, one participant died ten times, over a period of twenty minutes. Each death occurred during an enemy wave. Soon after, the participant decided to change the difficulty level from medium to easy. The change in difficulty level is identifiable by the activation of the pause menu (space), the settings page (degree of freedom), and a change from medium to easy (interaction). The modification goes beyond the space, degree of freedom and interaction layers, as it impacts on the game system layer. The game rules are adapted to make the progression easier for the player. The current example illustrates the difference between a top-down hierarchical application of the layers, and a transversal one. From a hierarchical perspective, the game system layer is seen as an inclusive layer, whereas a transversal and conceptual perspective displays how one layer (such as interaction) can actually impact another one (such as the game system). The change of difficulty means that the player, probably frustrated by the repetition of death sequences, decided to find a way to overcome the challenge, an action that occurs outside the game-world, yet impacts play within the game world. Interestingly enough, after overcoming the challenge, the player reversed the difficulty re-selecting medium again, thus completing a sequence of events. During this interval in which the difficulty was easy, the player showed less interest in fighting, preferring to loot extensively, acquiring money and using it in vending machines to obtain several power up items. When the player encountered the strong adversary that had defeated him/her on previous attempts, he/she lost a lot of health, but was able to regain

full health in the middle of the battle, as he/she had used the money to buy first aid kits that permitted him/her to complete the challenge.

During their first performance with *Bioshock 2*, the same participant encountered their first fighting sequence (identifiable by hitting an enemy and suffering a small drop of health). Then, an option to loot a corpse was taken up. This event highlights a key moment for the player, as it shapes the play from this moment onward. This is *transversal*, as the moment the player was invited to loot for the first time constituted a change in the player's degree of freedom. While the player was not prevented from looting beforehand, the game did not present the player with any corpses to loot until this point. The player was not prevented from looting, nor was information withheld, but behavioural opportunities were presented alongside the presence of objects. That is, the corpses possessed a ludic function as depositories or sources of wealth accumulation.

Utilising a Temporal Frame of Reference

The layers can also be used transversally to connect different segments together, in order to improve understanding of experience. A reconstruction based on *temporal frame of references* can be linked to the notion of the unresolved reference presented in Section 3.5.1. The idea is to find a segment where an index can be used to make the understanding of other segments more explicit. This transversal reconstruction uses three temporal relationships, those that are 1) *preceding*, when a prior segment can explain the meaning of what occurs later, 2) *co-occurring*, when co-occurring segments aid the experiential interpretation of

both, and 3) *subsequent*, when a segment can explain the meaning of a previously encountered segment.

The use of *co-occurring* segments can be illustrated with *Bioshock* 2. During the same sequence in which the player died continuously over a twenty-minute period, goal pop-up segments can be analysed in conjunction with death segments. The rationale for doing so is that, in the case of an enemy wave, the goal is reinforced again after each death. Thus, each death segment co-occurs with a goal pop-up panel indicating that the player has to retry the sequence. The occurrence of a goal pop-up also occurs following the successful completion of the enemy wave. This specific communication device can then be understood as validating success and progress as it must be the next goal. The co-occurrence of death with a pop-up panel signifies the player's need to retry, while the absence of any death prior to a pop-up panel indicates new directive.

Based on the *preceding segment*, the same player was confronted with an intense fighting sequence, identifiable by two strong health drops and the use of power. In the minutes following the fighting sequence, the player went into the help menu space. It is reasonable to speculate that the fighting sequence triggered some thoughts in the player to better understand the gameplay mechanisms of the game and the game world.

Finally, an example of *subsequent segment* explaining a sequence of events can be drawn from the understanding of why a participant previously entered *key binding* menu. By studying the first actions made by the player as soon as he/she exits the

key binding menu, it is possible to know, in retrospect, what was the reason he/she initially enters the menu.

Analysis using Semantic Networks

Another manner in which transversal analysis of performance can be achieved is through the conjoint study of two elements that can be considered as being parts of the same *semantic network* (see Section 3.5.3). For example, health and power can be considered as being part of the same semantic network, as both are known to evolve principally during, or because of, fight sequences. By examining how one evolves in relation to the other, the analyst can learn much about a player experience. For example, either the player uses power without losing health (indicative of a perfect fighting sequence), or loses health without the use of power (indicative of a strategy that involves trying to save it for later).

Distinguishing Loop Sequences

Linked to a semantic network reconstruction focused on similarities, the detection of a *loop* seeks to identify sequences that are (nearly) identical. For instance, if in *Bioshock 2* the player is presented with the same in-game cut-scene or listens to the same audio-tape several times. As mentioned before, when players die in *Bioshock 2*, they respawn at the last *Vita Chamber* encountered, which provides spatial checkpoint or marker. After each death, players then need to retrace their steps to get back to the place where they had previously died. This represents action within a space that has already been encountered by the player, and can therefore be considered a loop. It can be assessed according to whether or not the player decides to engage differently. For example, a player may choose to loot

more the second time, in order to prepare themselves better for an upcoming fight that previously led to his/her death.

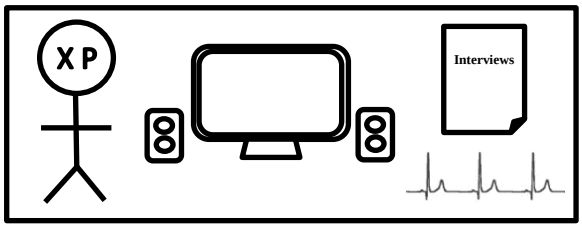
Comparison Based Reconstruction

The last reconstruction approach can be seen as cross-transversal, as it is about comparing segments from one performance, with other segments occurring in different performances. Three types of comparisons can be executed: *between-sessions*, *between-players*, and *between-games*. *Between-sessions* refers to the same player, the same game, but different sessions. In the research example of *Bioshock 2*, one participant mainly spent the first session (the first 45 minutes of play) in the help menu space, but by the fourth session (after three hours of play), this behaviour has almost disappeared from their gameplay performance, indicating the possession and mastery of better skill sets.

Between-players refers to the same game, the identical session number, but different players. In *Bioshock 2*, different performances can be compared, to distinguish between the players who are more interested in the story and the ones who are more interested in the action.

Lastly, the *between-games* comparison refers to the same player but different games. One participant was very story-oriented in the game *Bioshock 2* (indicated by the constant use of the help menu), but paid no attention to the story elements in *Max Payne 3* (indicated by the finding of clues). Such a comparison indicates how the same player can engage differently, depending on the story or the game genre.

4.2.5 *Performance summary*



Once a performance has been successfully deconstructed and reconstructed, producing a summary of the performance, it is possible for further analysis and correlations to be extracted using other data sources, such as psychophysiology measures (Mirza-Babaei, Nacke, Gregory, Collins, & Fitzpatrick, 2013), interviews, or visualisation approaches (Marczak, Vught, Schott, & Nacke, 2012).

4.2.6 *Summarisation*

To demonstrate the outcome of a gameplay performance segmentation for a 45-minute session with the game *Bioshock 2*, performed by the same participant (anonymised as Participant P.), a summary is presented in its original form in Figure 35. In Chapter 6, more detailed and annotated summaries are presented in an interpretation of play. Figure 35 is presented to complete the account of the research process.

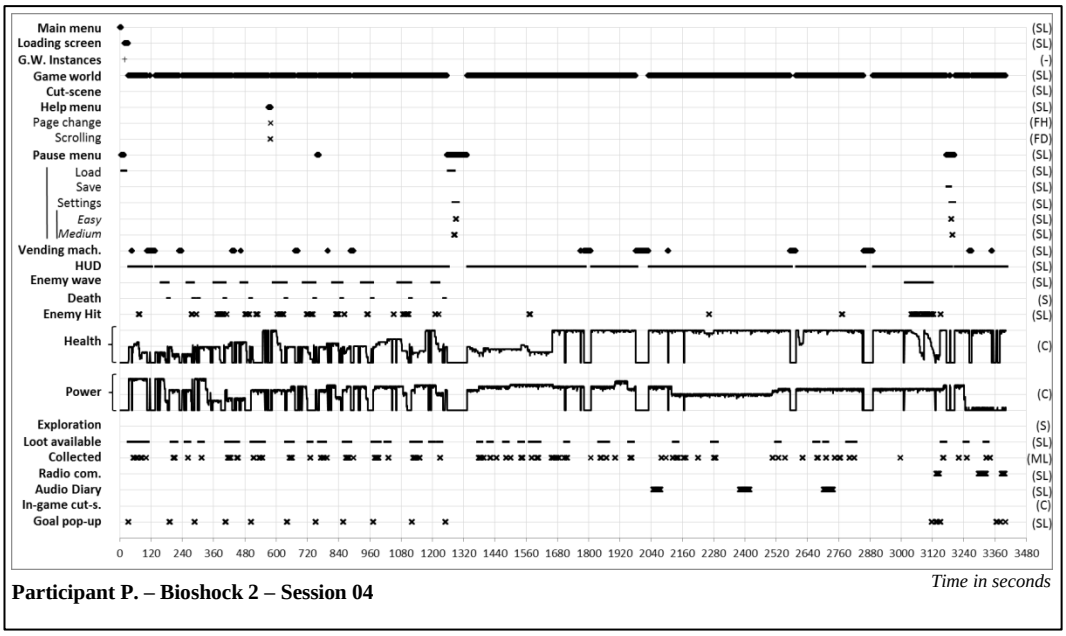


Figure 35. Fourth session of Partipipant P. with the game *Bioshock 2*

Figure 35 illustrates a performance summary using feedback-based gameplay metrics and the gameplay performance segmentation concept. The processing will be more thoroughly described in Chapter 5, and the different summaries will be more detailed in Chapter 6. However, in the current chapter, the summary is presented to provide an idea of what a summary looks like.

4.3 Conclusion

Chapter 4 has discussed the process associated with applying a multi-layered performance model. It addressed how different approaches to controlled-vocabulary processing can impact upon what constitutes segment coherence, illustrating how they need to be processed in separate layers. Placed in the overall context of an assessment of player experience (from the gameplay sessions, to the final analysis), Chapter 4 has demonstrated how the multi-layered model is intended to guide both the deconstruction and reconstruction of a recorded gameplay performance, highlighting the way each layer may be used (hierarchically for a deconstruction, and transversally a the reconstruction).

The five layers presented in this chapter address several basic questions concerning player experience. The interaction layer addresses the *what* and *how* behind player actions. The *degree of freedom* layer answers the *why* in the sense that it provides a context in which the action takes place, and how play is cued. The *spatial* and *temporal* layer asks the *when* and *where*. Finally, during the reconstruction, the introduced approaches for interpretation (contextualisation, temporal frame of reference, semantic network, loop and comparison) are also linked to the *why* of play, offering contexts for an increased understanding of the player experience.

It is now the task of the thesis to demonstrate that feedback-based gameplay metrics can be generated automatically, capable of segmenting and summarising a gameplay performance, while also demonstrating that such an approach can be applied to games other than *Bioshock 2*. Indeed, one of the main contributions of

the present thesis is how it proposes a methodology that seeks to process hours of gameplay footage, making automatisation a mandatory component of the processing. The role of the next chapter is to present the algorithms employed to automatically deconstruct gameplay performance.

Chapter 5

Automatic gameplay performance deconstruction

After recalling the current existing models around player experience assessment, Chapter 2 presented the concept of feedback-based gameplay metrics, which produces quantitative data reflecting player interaction with different or individual game systems. Such data is directly extracted by processing into audio-visual streams presented to the player during game play.

Chapter 3 was then dedicated to the introduction and definition of a gameplay performance segmentation process. While similar to gameplay segmentation, as defined by Zagal et al. (2008), in the manner that considers gameplay as an absolute content-based description of a videogame (regardless of a specific player activation); gameplay performance segmentation understands gameplay more as a relative entity, representative of the way a game unfolds from a specific set of interactions performed by a specific player. Chapter 3 argued that player experience is a part of the gameplay performance segmentation process, as the version of the game system available for segmentation is generated by a player's gameplay session. Chapter 3 also introduced the concepts of segmentation and indexing in the general context of document retrieval, with the underlining idea that feedback-based gameplay metrics can be used for both segmenting and indexing gameplay documents.

Chapter 4 then demonstrated that the gameplay performance segmentation process is a useful novel methodology to assess player experience and was achieved by

using a multi-layered approach, first hierarchically to (structurally) deconstruct a gameplay performance; and transversally to (experientially) reconstruct the same performance. The challenge of this approach is its automatisation, requiring a strong rationale for the way gameplay performance is processed. So far, no attempt has been made to automatize gameplay performance segmentation. Typically, in order to operationalize different aspects of player experience, several data sources need to be combined. Any gameplay session can produce a large amount of audio-visual material that, while allowing hand-coding process, would still require hours of work to log and summarize gameplay information. Not only hand-coding gameplay elements from audio-visual recording would be highly time consuming (for the *Bioshock 2* (2K Games, 2010) case study example provided in Chapter 4, nearly 25 elements are taken into account), but also continuous data, such as health level, are impossible to manually process with precision.

Chapter 5 is devoted to demonstrate that audio and video processing algorithms, as developed in computer science for media segmentation and indexing, can be applied on gameplay footage to automatize the complex and time-demanding task of gameplay performance segmentation. The hierarchical structure of the deconstruction process lends itself well for contextualizing each step of the algorithms, in order to avoid false alarm[12] detection and have a finer processing.

[12] False Alarm represents an algorithm result stating element detection where the element is actually not present.

For instance, there is no need to detect a player shooting interaction if the game space has been detected beforehand as being an option menu. Chapter 5 focuses on methods that automatize the segmentation process. This necessitates a pre-analysis of the game, a gameplay segmentation as defined by Zagal et al. (2008), as an initial step to elicit and understand the different gameplay elements that describe efficiently a gameplay performance in order to understand the player experience. The pre-analysis process, where the researcher consciously selects the different gameplay elements of interest, and identifies their audio or video representation, makes the methods presented in Chapter 5 part of what is defined in the present thesis as a *controlled-vocabulary gameplay performance segmentation*.

The algorithms presented in Chapter 5 are about detecting static information (Section 5.3), detecting moving information (Section 5.4), assessing a bar progression (Section 5.5), detecting visual breaks (Section 5.6) and detection of specific sounds (Section 5.7). For each algorithm, a discussion about the validity and some processing examples are provided. Chapter 6 will then illustrate the use of the presented algorithms for gameplay performance segmentation to an experiential reconstruction of the gameplay performance using the videogames introduced in Chapter 2 during the game sessions. But first, Chapter 5 describes in more detail what automatic audio-visual controlled vocabulary segmentation actually entails (Section 5), and recalls some standard concepts of image, video and audio representation of data in computer science (Section 5.2), in order to understand better the different presented algorithms.

5.1 Automatic controlled-vocabulary segmentation

As introduced above, the main objective of Chapter 5 is to present several methods seeking to automatize the demanding, yet insightful, process of gameplay performance segmentation. By automatizing the time-consuming task of audio-visual analysis, the likeliness of producing an exhaustive summary of the gameplay performance improves.

5.1.1 *A computer science contribution*

Chapter 5 introduces several algorithms, already used for audio and video analysis of multimedia documents, which have been proposed by the computer science community as part of the sound and image processing research area. Indeed, automatic segmentation and indexing of multimedia content is a growing topic of interest in computer science, especially in the current context where the amounts of available multimedia documents are growing exponentially. The large amount of data produced every day profits from mechanisms seeking to automatically sub-divide them in cohesive segments, using an automatic tagging system (Benois-Pineau, Precioso, & Cord, 2012).

Methods like automatic scene change detection (Adhikari et al., 2008), logo detection in TV-Stream (Santos & Kim, 2006) or detection of specific sounds in larger sound streams (Roads, 1996, p. 509; Yarlagadda, 2010, Chapter 2) are popular methods that have yet to be applied to videogame footage analysis. However, the very nature of gameplay footage makes them a perfect candidate for current audio-visual processing algorithms, as numerous symbolic information (both audio and video), directed to the player to make them aware of gameplay

elements needed for correctly interacting with the game system, exist and are ready to be processed.

5.1.2 *A controlled-vocabulary approach*

Before getting to the heart of the matter, it is important to focus on the *controlled-vocabulary* aspect of the methods presented in the current Chapter; as opposed to *free-text* methods that will be presented in Section 6.2. The controlled-vocabulary algorithms presented in the following sections require the analyst to first have a pre-understanding of the videogame mechanism, which could be achieved through an initial gameplay segmentation process as proposed by Zagal et al. (see Section 4.1.2). An example for instance is the static information algorithm presented in Section 5.3. Based on a logo-detection algorithm, the result is not about the morphological properties of the logo (such as colour, size, shape, potential text content), but about the gameplay meaning of its presence or absence (such as an HUD on-screen meaning in-game, and no HUD meaning cut-scene). Similarly for the sound detection algorithm presented in Section 5.7, the result is not about the sound itself (frequency, amplitude, tonality, harmonic) but about the gameplay meaning of the sound being heard (such as death sound, or object correctly looted). The analyst is then required to effectuate a pre-analysis of the game (once per game), and elicit the different elements of interest, while linking them with matching gameplay concepts and audio-visual representations. As highlighted in Chapter 4, the pre-analysis process can be based both on the *Game Ontology Project* (Zagal et al., 2005) outcomes, and the gameplay segmentation process (Zagal et al., 2008). The need for a pre-analysis step can also be illustrated by the nature of inputs required by the algorithms in Chapter 5, indicative of the pre-analysis outcome (which part of the screen, which element, etc.); as opposed to

the algorithms that will be overviewed in Section 6.2, requiring no (or few) inputs from the analyst[13].

5.2 Computer science representations of image, video and sound

The multidisciplinary nature of the present thesis, mixing social sciences and computer science, involve containing chapters discipline-specific, and that may be confusing for a reader not used to the discipline standards. This is the case of Chapter 5 that is presenting automatic audio and video processing algorithms, giving the chapter a strong computer science angle. The aim of Section 5.2 is to recall some notions about image, video and sound data representation and analysis, accepted as standard in computer science. This section is useful for a non-computer science expert to follow the different approaches and algorithms presented throughout Chapter 5, and is directed toward a general understanding of the main data structures that are used to efficiently represent and process audio-visual data.

Section 5.2 focuses on presenting two basic data structures, *vector* and *matrix*, that are standard for representing image, video and audio data types. Data structures are structures designed to ease the representation, analysis and manipulation of complex data types, like image, video and sound. Figure 36 illustrates a *vector* and a *matrix*. The vector structure is a one dimensional array of

[13] Section 6.2 will introduce the meaning of free-text automatic methods, which are methods requiring no, or very little, prior-knowledge of the videogame. In other words, that means that the gameplay elements are directly gathered for the game without an analyst abstraction. Free-text methods will be mainly used for comparison purposes.

contiguous values, each of them being accessible through an index (square number). In the context of the present thesis, a vector is used as a temporal representation, with the first index pointing to the beginning of a media and the last index pointing to the end of that media. This for instance is the case for sound representation (see Section 5.2.3). In Figure 36, an example of vector of n contiguous values is shown. The value at index 0 is S_0 (or vec(0) = S_0), the value at index 3 is S_3 (or vec(3) = S_3), etc. until n-1 (as the indices start at 0, the last position is n-1). The matrix structure is a two dimensional array of values, each value being accessible using two indices: a column index and a row index. In the context of the present thesis, a matrix is used as a spatial representation, each pair of indices pointing to a planar location. This is for instance the case of an image representation (see Section 5.2.1). In Figure 36, a matrix of n columns and m rows is displayed. The value at indices 0 and 0 is $P_{0,0}$ (or mat(0,0) = $P_{0,0}$), the value at indices 3 and 2 is $P_{3,2}$ (or mat(3,2) = $P_{3,2}$) etc. until n-1 and m-1. Finally a video, being both a temporal media and a spatial one (sequence of images) can be seen as a vector (temporal) in which each element is a matrix (spatial) (see Section 5.2.2). For sound, each value is called a *sample*. For image, each value is called a *pixel*. For video, each temporal position is called a *frame*.

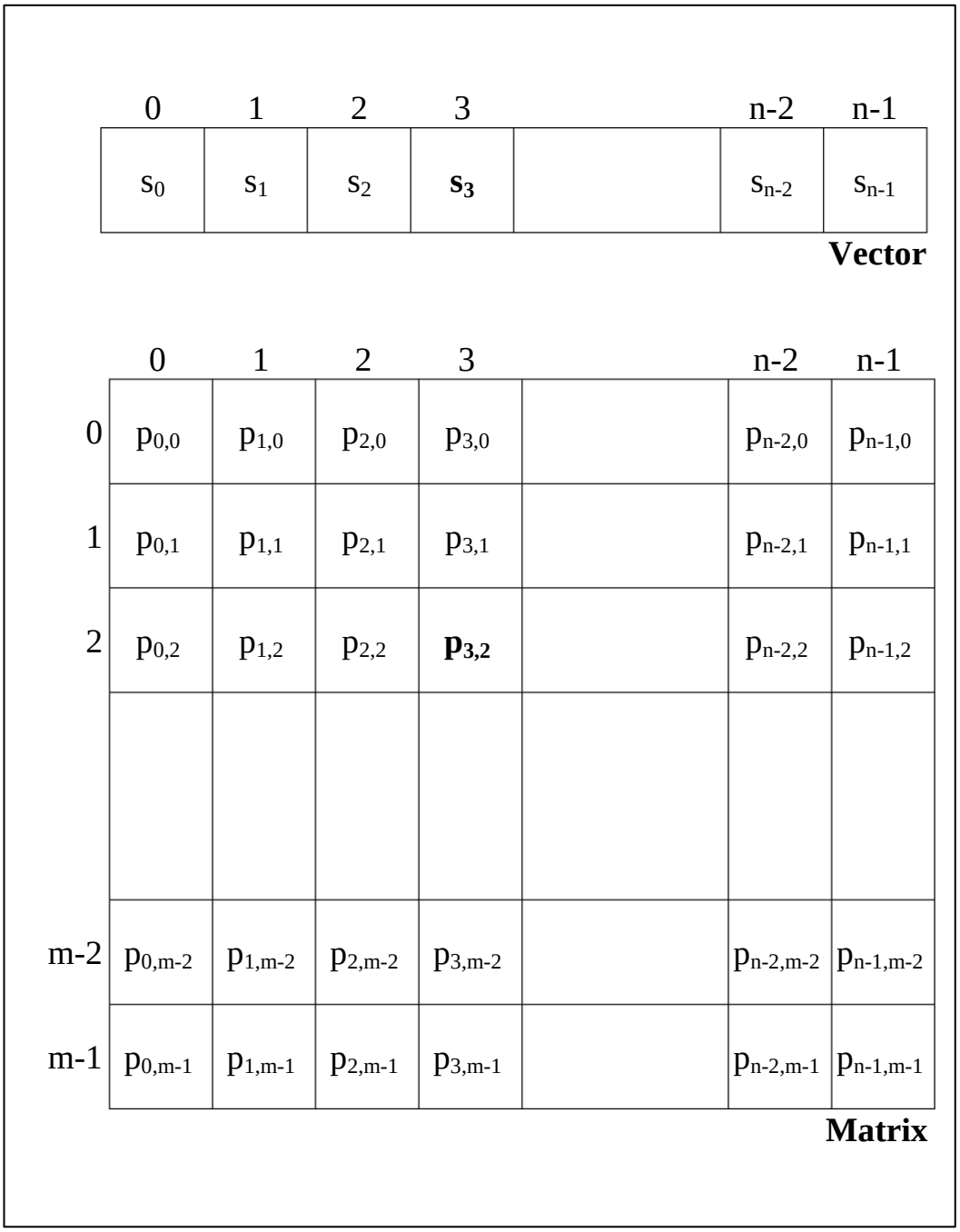


Figure 36. Vector and Matrix data structures

Figure 36 displays two diagrams representing the main data structures used for image and sound processing. The top one, a vector, is a list of contiguous values, which can be accessed through a position index. For example, the value at position 3 is s_3. The bottom one, a matrix, is a two dimensional array of values, accessible using two indices (column and row positions). For example, the values at indices 3 and 2 is $p_{3,2}$.

5.2.1 *Image representation*

In computer science, an image can be seen as a matrix of pixels, a pixel representing a colour value at a specific location inside the matrix (Gonzales & Woods, 2007, Chapter 2). A physical image needs to be cut into several columns and rows, in order to be represented in computer memory as a matrix. This process is called *discretisation* (change from continuous representation to discrete representation). Figure 37 illustrates the discretisation process using six columns and five rows, giving a 30-pixel image. In Figure 37, the pixel at position (4,1) represents the sun.

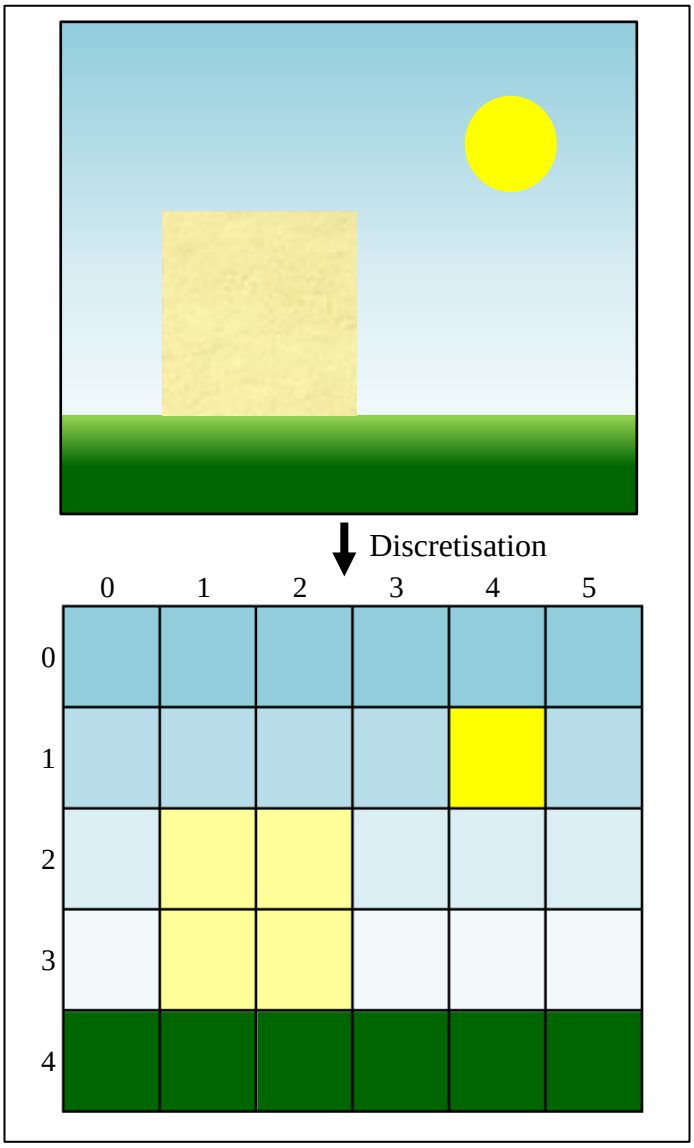


Figure 37 illustrates the discretisation process, converting a physical image into a matrix representation, by cutting it into several columns and rows. By using six columns and five rows, the image, once discretised, contains 30 pixels. Using the matrix structure, the pixel at position (4, 1) represents the sun.

Figure 37. Image discretisation

There is no immediate link between the physical dimensions of an image (width and height in cm), and the digital dimensions (number of columns and rows). Then, more or less pixels can be used for a same physical length in order to improve the quality (more pixels), or to lower the storage size (less pixels). However, the core point to understand for the following sections is that an image is a *matrix* of pixels, with the pixel indexed (0, 0) representing the upper left corner of the image, and the pixel indexed (nbofcolumns - 1, nbofrows - 1) representing the bottom right corner, and every other index representing the pixels in between. All the pixels of an image can then be accessed using all index values in between.

The way colour information is represented by each pixel can also differ depending on the purpose of an image processing algorithm. Giving a pixel a value is a process called *quantisation*. For greyscale images, each pixel will have only one value, in general ranging from 0 (black) to 255 (white), with the values in between representing the different shades of grey. Figure 38 illustrates the greyscale quantisation process using the image example of Figure 37

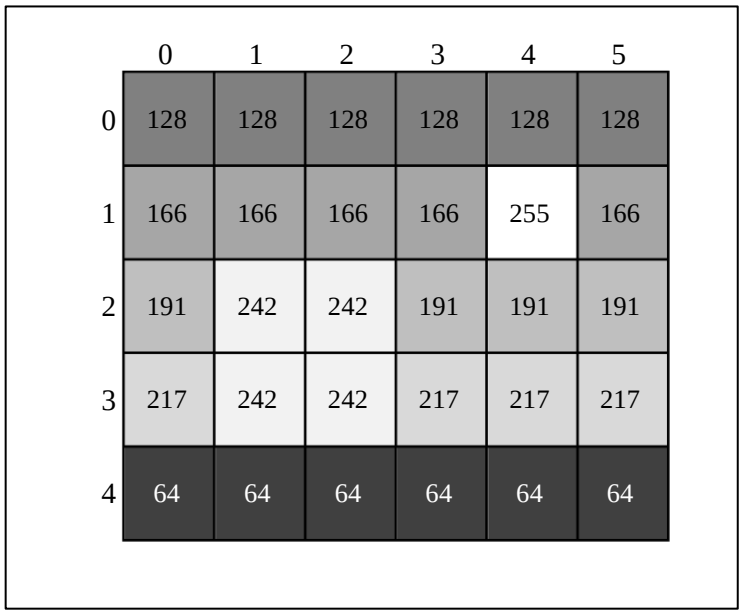
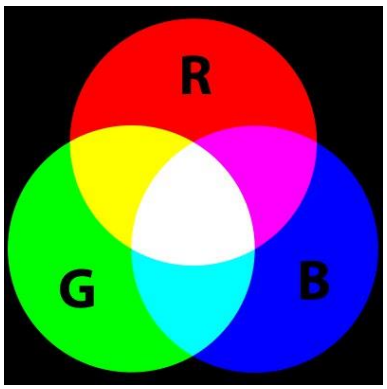


Figure 38. Greyscale quantisation

Figure 38 illustrates the quantisation process, using greyscale colour information. Each pixel has one value, ranging from 0 (black) to 255 (white). On Figure 38, the sun is white, so with a value of 255.

For colour images however, a pixel colour will generally be represented using three values[14]. The most commonly used colour quantisation is the RGB representation, which specifies a ratio of the primary colours Red, Green and Blue (between 0 and 255) in order to create all the possible colours as an additive model. For instance, (255, 0, 0) represents Red, (0, 255, 0) represents Green, (0, 0, 255) represents Blue, (255, 255, 0) represents Yellow, (255, 255, 255) represents White, etc. Different shades can be represented by using more or less of each component ((128, 0 ,0) would be a darker Red). Figure 39 illustrates the use of the RGB representation for quantizing the "house" example of Figure 37.



	0	1	2	3	4	5
0	146 205 220	146 205 220	146 205 220	146 205 220	146 205 220	146 205 220
1	182 221 232	182 221 232	182 221 232	182 221 232	255 255 0	182 221 232
2	218 238 243	255 255 153	255 255 153	218 238 243	218 238 243	218 238 243
3	242 249 251	255 255 153	255 255 153	242 249 251	242 249 251	242 249 251
4	0 102 0	0 102 0	0 102 0	0 102 0	0 102 0	0 102 0

Figure 39. RGB quantisation

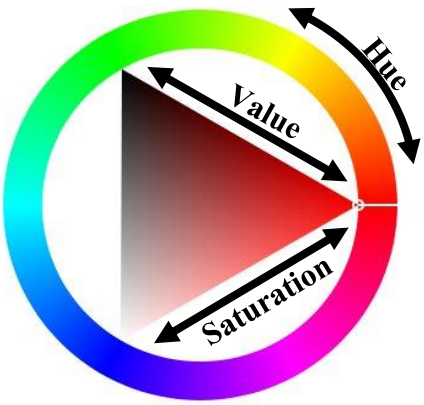
Figure 39 illustrates a RGG quantisation of the "house" example of Figure 37. Each pixel has three values, which range from 0 to 255, and that represent a ratio of the three primary colours for additive model (Red, Green, Blue). The sun is a perfect yellow, so the pixel color values must represent a full mix of Red and Green (as illustrated on the left diagram). The pixel representing the sun is then given the values (255, 255, 0).*

** The left diagram is a public domain figure, originally created by Mike Horvath and updated by Jacobolus*

[14] Sometimes four, when transparency (alpha channel) is also considered.

Another colour representation, called HSV, is sometimes preferred to the RGB one[15] (this is notably the case for the algorithm presented in Section 5.5). The HSV representation, instead of being based on Red, Green and Blue values, is based on a Hue, Saturation and Value information. The key contribution to the HSV representation in regard to the RGB one is that it is closer to the human perception of colour (notably due to the cyclic nature of the Hue component), meaning that two close pixel values are more likely to look the same as two distant pixel values. In Figure 40 for instance, it is noteworthy that all the blue-looking pixels have exactly the same hue (96), and the yellow-looking pixels also have the same hue (30). In the RGB representation in Figure 39, however, the blue-looking pixels have more variations. In the case of the algorithm presented in Section 5.5, which seeks to identify the presence or absence of a specific colour on screen, using HSV allows, for instance, to look for pixels that *look like* red more easily than with the RGB system.

[15] There are actually more different pixel colour representations, but only RGB and HSV are useful in the context of the current thesis.



	0	1	2	3	4	5
0	96 86 219	96 86 219	96 86 219	96 86 219	96 86 219	96 86 219
1	96 56 232	96 56 232	96 56 232	96 56 232	30 255 255	96 56 232
2	96 25 242	30 102 255	30 102 255	96 25 242	96 25 242	96 25 242
3	96 10 250	30 102 255	30 102 255	96 10 250	96 10 250	96 10 250
4	60 255 102	60 255 102	60 255 102	60 255 102	60 255 102	60 255 102

Figure 40. HSV quantisation

Figure 40 illustrates a HSV quantisation using the "house" example of Figure 37. HSV (Hue, Saturation, Value) is closer to the human perception of colours than RGB, especially due to the cyclic nature of the Hue (left diagram). All the blue looking pixels have exactly the same Hue value (96), as well as all the yellow looking pixels (30). However, in the RGB version of the same image (Figure 39), the blue looking pixels show a greater variation in their values.*

**the left diagram is based on a screen capture of the HSV representation inside the GIMP software system (Spencer Kimball & Peter Mattis & the GIMP Development Team, V2.8.2, 2012)*

5.2.2 *Video representation*

A video can be seen as a sequence of images, temporally organized (or a vector of images, or even a vector of matrices in the most abstracted definition). A number of pixels defining a physical area depend on the desired quality during the discretisation process; and there is no standard for how many images are needed to represent one second of actual video. 25 to 30 images are in general accepted as a good number, but this could vary dependent on the desired usage of the video (15 images per second can be enough for a surveillance camera, 60 images per second can be better for a videogame). Each image is called a *frame*, and each frame can be accessed providing the frame number. The number of images, or frames, per second is in general summarized with the acronym FPS (Frames Per Second). In a 30 FPS video for example, the frame at index 90 represents the frame initiating the third second of the video. Once accessed, a frame can be processed as a regular image, as presented in Section 5.2.1. Figure 41 illustrates a sunset video, using four frames. The "moon" will be accessed by first considering the Frame at index 3, and then the pixel at position (4,0).

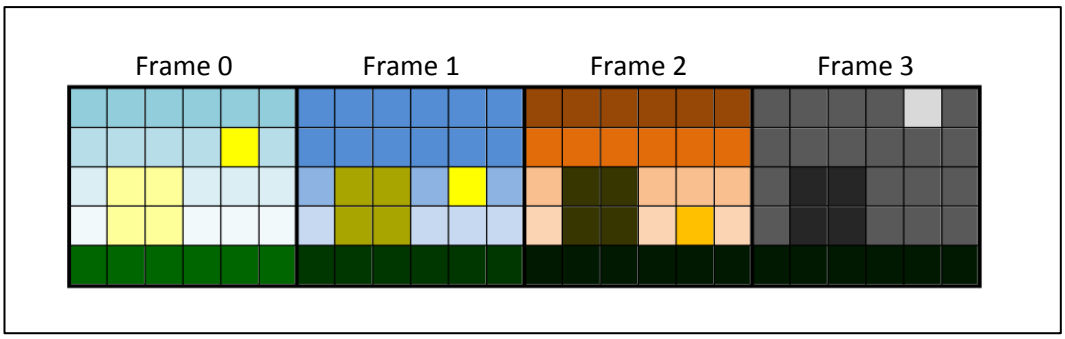


Figure 41. Video representation

Figure 41 illustrates a video representation of a sunset, as a vector of matrices. The vector contents can be accessed through a frame number (or vector index), and represent a full image (each vector value is a full matrix). Once the frame is selected, the pixels can be accessed as usual. To access the moon, the image must first be returned through the frame number (frame 3). Then the pixel can be accessed, at position (4,0).

In the following sections describing the different video-based algorithms, a *frame step* is regularly used as an input. Frame step is a way to speed up the video processing, or sometimes even improving it (see Section 5.3), by not considering every single frame, but only one frame over several. For example, a frame step of 2 would mean that only one frame over two are considered; a frame step of 10 would mean that only one frame over ten are considered, etc.

5.2.3 *Sound representation*

For audio media, the data structure is a discretised representation of the sound wave, in the form of a one dimension vector of values (values quantified between -1 and 1 for a normalized signal) (Roads, 1996, Chapter 1). As above, the number of values, or *samples*, is linked to the desired quality. For a CD, the *sample rate* of 44100 means that there are 44100 values to represent one second of sound, and these are used to describe the sound wave during this second. In Section 5.7, a sound will be seen as a list of values, temporally organized, that can be accessed using the *sample* number value. Figure 42 illustrates the sound representation, from discretisation to quantisation, based on vector structure.

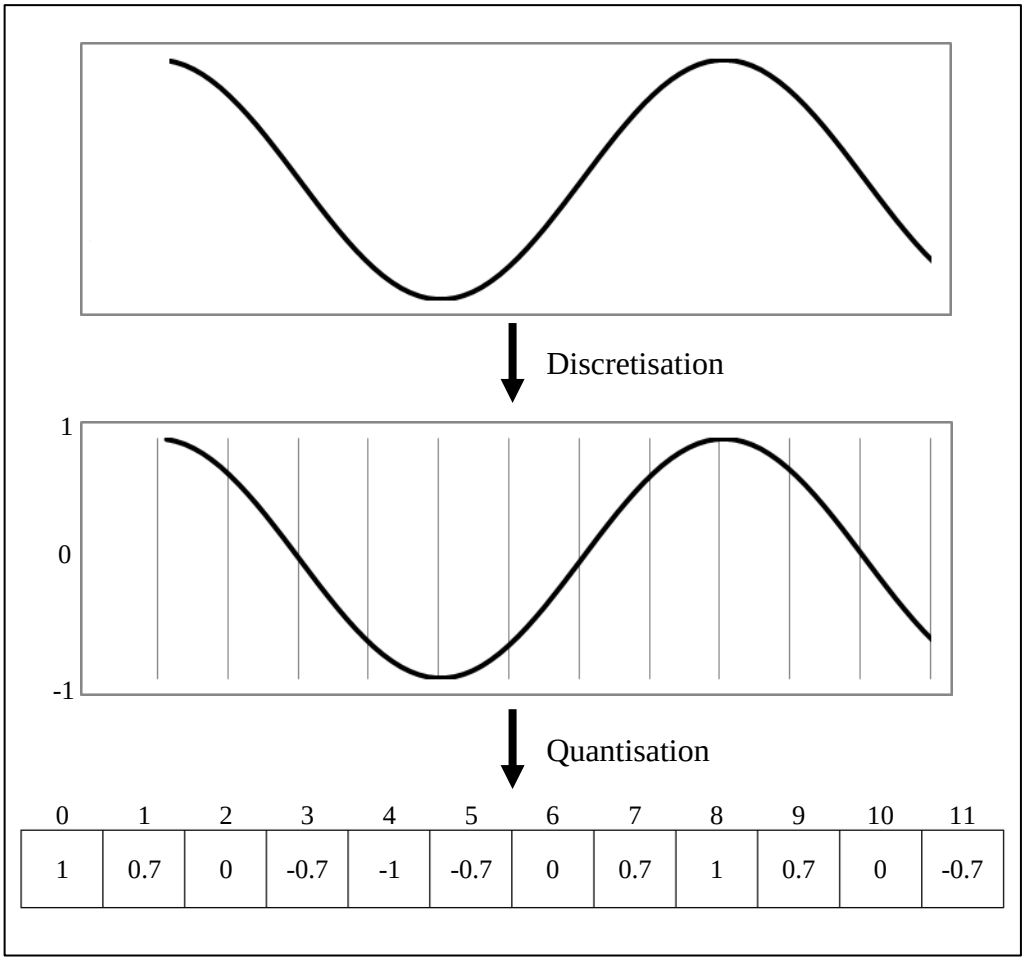


Figure 42. Sound representation

Figure 42 illustrates the discretisation (determining the number of values representing one second of sound) and the quantisation (determining the value between -1 and 1 in order to represent the actual sound wave at a given position) for the sound media, using a vector representation. A sound is then a list of values, temporally organized, that can be accessed using a sample number index.

5.2.4 *The mask image notion*

The notion of *mask image* needs to also be discussed, as all the video-based algorithms presented below need a mask image as one of their inputs. Providing a mask image is a way of specifying a sub-area of interest in an image or a video. To guide the video-based algorithms, it is mandatory to make explicit what part of the screen should be considered and what part of the screen needs to be discarded. A mask image is a binary image (containing only to colour, black and white) specifying the area to consider. The positions of the white pixels in the mask image correspond to the area to take into account into the considered image, and the positions of the black pixels match the area to discard. A mask image should be seen as a tracing paper that is superimposed on each of the video frames during the processing: white letting the bottom frame area visible, and black covering what is underneath. Figure 43 illustrates with one example the usefulness of the mask image. The mask image in Figure 43 specifies the "house" area. The mask image is a useful input because it informs both the position of the element to consider, but also its size and shape.

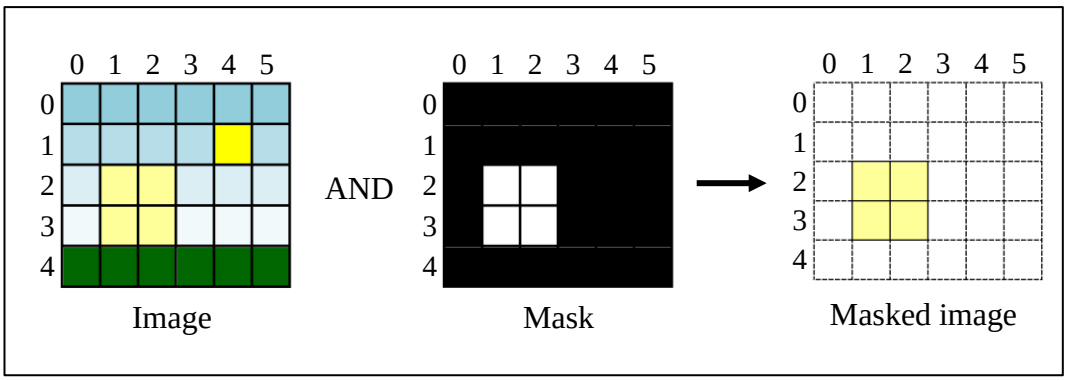


Figure 43. Mask image

Figure 43 illustrates the mask image notion, which will be used as input in the following video-based algorithms of Chapter 5, in order to specify the area of interest on the screen. A black pixel represents a position to discard, while a white pixel represents a position to consider. In Figure 43, the mask image serves to specify the "house" area.

5.2.5 *The process of algorithm development*

Given that Section 5.1 and Section 5.2 discussed the audio-visual processing of gameplay footage in a general manner, the following sections are dedicated to the presentation of algorithms that were developed specifically to meet the research aim of the current thesis.

The development of each algorithm used in the present research has been an on-going iterative and adaptive process on-going throughout the duration of the research. *On-going*, because the algorithms have been constantly interconnected with the different research needs, which are governed by the variations between game content and have required a more universal conceptual framework (see the multi-layered structure presented in Chapter 4) that accommodate a large range of games while also attending to the specific design features of different titles. *Iterative*, as every time a form of audio-visual feedback is identified as representative of player experience; a new algorithm has to be implemented. *Adaptive*, as when an algorithm produces results that are not accurate enough to be analysed, it has to be updated and re-trialled.

To outline this process, the next paragraph recapitulates the different incremental steps that contributed to the development of the algorithms presented in Chapter 5:

1) First, because of the variety of static symbolic information on screen (Fagerholt & Lorentzon, 2009) (on-going), the *static logo detection* algorithm has been developed (iterative). This algorithm accounts for a broad recognition of gameplay elements such as spaces (logo indicative of

a location), degree of freedom given to the player (logo indicative of a possible set of interactions), or action feedback (logo indicative of the correct accomplishment of an action). The algorithm was initially based on colour matching, but appeared to be unable to process semi-transparent information, a feature included in the game *Max Payne 3* (Rockstar Games, 2012) being tested. The algorithm was modified so that it was based on edge matching (adaptive), to address the issue of processing semi-transparent information.

2) Then, it became apparent that the detection of a static logo was too limited, as several symbolic elements move on screen (on-going). This is for instance the case of enemy location information appearing on-screen (as in *Battlefield 3* (Electronic Arts, 2011)), or on a map (as in *Dead Island* (Deep Silver, 2011)). A new algorithm was required to achieve detection of a *moving logo* (iterative). Several solutions have been developed, but showed unsatisfying results (such as *cross-correlation* between a logo image and the video frames). The solution eventually implemented was an adaptation of the one presented by Santos et al. (2006), that use *cross-correlation* over edge representations of the logo and the frames (adaptive).

3) The need to process continuous data has emerged next (on-going), as numerous games contain information constantly present on-screen and providing continuous updates of the avatar's condition (for instance health, power or stamina). A review of different games, such as *Max Payne 3*, *Dead Island*, and *Bioshock 2* (2K Games, 2010), shows that most of these continuous data are represented using a *bar representation*. An algorithm

to process *bar representation* was developed (iterative). First based on RGB colour representation, several tests show that an HSV colour representation greatly improves the accuracy of the result (adaptive).

4) By studying the behaviour of player whilst inside game menus (like the help menu of *Bioshock 2*), the need for a solution identifying a scene break has then emerged to detect the moments when the player change menu pages (on-going). A *scene change detection* algorithm was then developed (iterative). Because several issues can be addressed by studying scene change, two scene change detection methods (direct comparison and histogram comparison) have finally been proposed instead of only one (adaptive).
5) Finally, acknowledging the fact that the sound stream also conveys core gameplay information (on-going), like for instance during death screen in *Battlefield* 3, *Max Payne 3* and *Bioshock* 2; a *sound cross-correlation* algorithm has been developed (iterative).

The process can continue beyond the five algorithms presented in the current chapter. For instance, if the moving information algorithm can detect moving objects on-screen, it is unable to process any changing size information (for example a logo that grows bigger when the player is closer to an object). A solution can be found in adapting the SURF algorithm (Bay et al., 2008) for the purpose of gameplay analysis.

The present thesis only presents the final state of the algorithm. However, this iterative and adaptive process should be acknowledged as having occurred for each of the presented algorithm. To present the algorithm, the following sections

are based on the same structure: 1) introduction, which replaces the algorithm inside the general context of use, and recalls its origin; 2) approach description, which gives an overview of its role for gameplay analysis; 3) step by step process, which outlines in detail the different procedures of the algorithm; 4) pseudo-code algorithm, which presents the algorithms using a syntax that can easily be translated into programming languages; 5) results and validity discussion, which evaluates the approach efficiency, and acknowledges some limitations.

The following sections will also contain links to the pseudo-code version of each algorithm. In the present thesis, the notation [Acronym Lines] will be used to reference the matching lines in the pseudo-code version of the algorithm. For instance, [SL 10-20] represent the Static Logo pseudo-code, from line 10 to line 20. Algorithm inputs can also be referenced using the notation [input number]. For instance, [input 3] means that the element corresponds to the algorithm input number 3. In the present thesis, the algorithm acronyms are SL (Static Logo, Section 5.3), ML (Moving Logo, Section 5.4), C (Colour Ratio, Section 5.5), F (Frame comparison, Section 5.6) and S (Sound cross-correlation, Section 5.7).

5.3 Static information detection

The first algorithm presented in Chapter 5 for automatizing gameplay performance segmentation process is a straightforward one, yet accurate and generic enough to be applicable at any layer of the gameplay performance segmentation model (cf. Chapter 4). Derived from a video processing method seeking to detect TV-Channel logos for automatic segmentation and indexing of TV-Streams (Santos & Kim, 2006), the following approach represents a direct adaptation to the videogame world, where static graphical information is widely

used (HUD, menu title, buttons, etc.) (Fagerholt & Lorentzon, 2009), and can be seen as identical to logos in terms of their parameters and representation. Any type of static on-screen information can be treated as a logo. For gameplay analysis, these logos can be indicative of a wide spectrum of gameplay elements, such as spaces, interaction possibilities offered to the player, or action feedback.

The method proposed by Santos et al. focuses on assessing the presence or absence of a TV-Channel logo and recognizing which version of the logo is actually presented to the watcher (for instance, a TV-Channel logo can slightly change during live retransmission as opposed to pre-recorded show). Identifying logos and assessing their presence is an insightful tool for a TV-stream. Indeed, a TV-Channel logo informs the nature of the broadcast content, making a TV-channel logo analysis a good tool for segmentation (separating different programs) and indexing (describing their content). For example, the absence of a logo in general represents an advertisement segment, while a change of logo may indicate a change of the stream content (from live to pre-recorded sequence for instance). But more than the nature of the TV-stream content, detecting a logo is also linked to the notion of space, as channel logo recognition is a way to discriminate between different channels. The power of a logo detection method highly increases when applied to the videogame media, where static graphical information is numerous and various (Fagerholt & Lorentzon, 2009).

The strength of the method proposed by Santos et al. (2006) is that it allows for an inclusive definition of logo. The method can indeed be used to detect a regular logo, but also any static text or static portion of screen (typically, for videogame,

any non-diegetic information that is superimposed over the game world). Furthermore, the method is also based on edge recognition, instead of colour matching. Therefore, the method is robust to semi-transparent logos. Figure 44 illustrates the significance of considering edges instead of colours for logo detection using an example from the game *Max Payne 3* (Rockstar Games, 2012). In the case of *Max Payne 3* the game employs a semi-transparent logo that updates the player on the status of avatar health. The transparency of the logo combined with player movement permits colours from the bottom layer (game world) to alter the colour of that logo. In this instance edge detection, focusing on logo shape and discarding the colour information, is preferable as a means of recognising the logo.

Figure 44. Semi-transparent HUD in Max Payne 3

Figure 44 displays the HUD in the game Max Payne 3, at three different moments, in order to illustrate how colour detection methods would be unable to recognize semi-transparent logos: the first one is mainly white, the second one is mainly red, and the third one is mainly purple.

Applied to a videogame, the wide range of elements that can be detected by such a method makes an adaptation of the Santos et al. algorithm a powerful tool that can not only be used to automatize gameplay performance segmentation algorithm, but also be applicable to any of the layers presented in Chapter 4. Due to the nature of the videogame, the player needs not only to be aware of the game space and the state of their avatar by being presented with representation of senses they cannot feel as not being directly inside the game world (Ruch, 2010), but the player also needs to be aware of the authorized actions and receive feedback to

validate their accomplishment. These gameplay elements can be broadcast through audio or video cues. The logo-detection-based algorithm presented in Section 5.3 gives an accurate solution for the video cues. Detecting a logo in the videogame context can then inform about the space, as some spaces, like menu for instance, have a very specific design with static parts that can be considered as the logo. A logo can also indicate new interactivity possibilities, for example by a skip icon when it is allowed to skip a cut-scene, or by an instruction text appearing when a body can be examined. Finally, a logo can inform about actual interactions, as some interactions are associated with graphical feedbacks, for instance a change of weapon, a tape being listened to, item collected.

5.3.1 *Approach overview*

The main idea behind the logo detection process is to be able to detect static graphical elements on screen, and more specifically to assess whether or not a chosen element is indeed enclosed inside the current video frame. In the current section the exact location of the graphical element is considered as known and is part of the input needed from the analyst[16]. A first straightforward solution would be to directly calculate the sum of differences between the pixel values of a logo, and the supposed location in the frame. This solution, colour dependent, has been used as a first solution (Marczak et al., 2012), but has later proven to be unable to detect semi-transparent logos. An edge-based method, robust to colour changes,

[16] Section 5.4 will deal with a moving or location-unknown logo.

like the one proposed by Santos et al. is however robust to semi-transparent information, and allows considering a greater variety of videogames.

5.3.2 *Step by step description*

To conduct logo detection, the algorithm requires the following inputs:

1. The *gameplay footage* in the form of a video file (sound is not mandatory)
2. A *reference image*, sourced from a screen capture containing the desired logo of interest that matches the size of the video frames.
3. A *mask image*, indicating the area in which the logo is present and should be found.
4. An *error tolerance value*, specifying how much error is tolerated during the logo detection process between the reference image and the screen image.
5. A *time tolerance value*, specifying the minimum period of time that the logo should remain on screen (to discard short false-detection).
6. A *frame step*, that reduces the processing by not requiring each individual frame to be processed, but instead every 2^{nd} or 10^{th} frame.

Because the algorithm is based on edge comparison, the first step is to transform the input reference image [input 2], from full colour image to its binary edge representation (each edge is in white (1), everything else is black (0)), through a

greyscale version[17]. For that, the reference *canny edge detection* method (Canny, 1986) is used [SL 1-3]. Then, the image mask [input 3] is employed to specify the logo position, and discard the pixels outside the area of interest [SL 6]. Figure 45 and Figure 46 illustrate these first steps. The static information detection is used to detect the presence or absence of the HUD in the game *Max Payne 3*, indicative of a full-interactivity gameplay (when HUD on screen) or a reduce-interactivity section, like cut scene (when HUD is absent). In Figure 45, the reference image containing the HUD (lower-right corner) is provided by the analyst, and converted first into its greyscale representation, and then to an edge representation. As displays in Figure 45, the edge image represents the shapes outline of the objects present in the frame. In Figure 46, the edged version of the reference image is masked using the provided mask[18] to only select the HUD area and discard the rest of the frame.

[17] Indeed, the edges are similar in a greyscale conversion of an image, but each pixel is easier to process as it has only one value (see Section 5.2.1) instead of three in the case of colour pixels
[18] Using the Boolean AND operator (Boole, 1848).

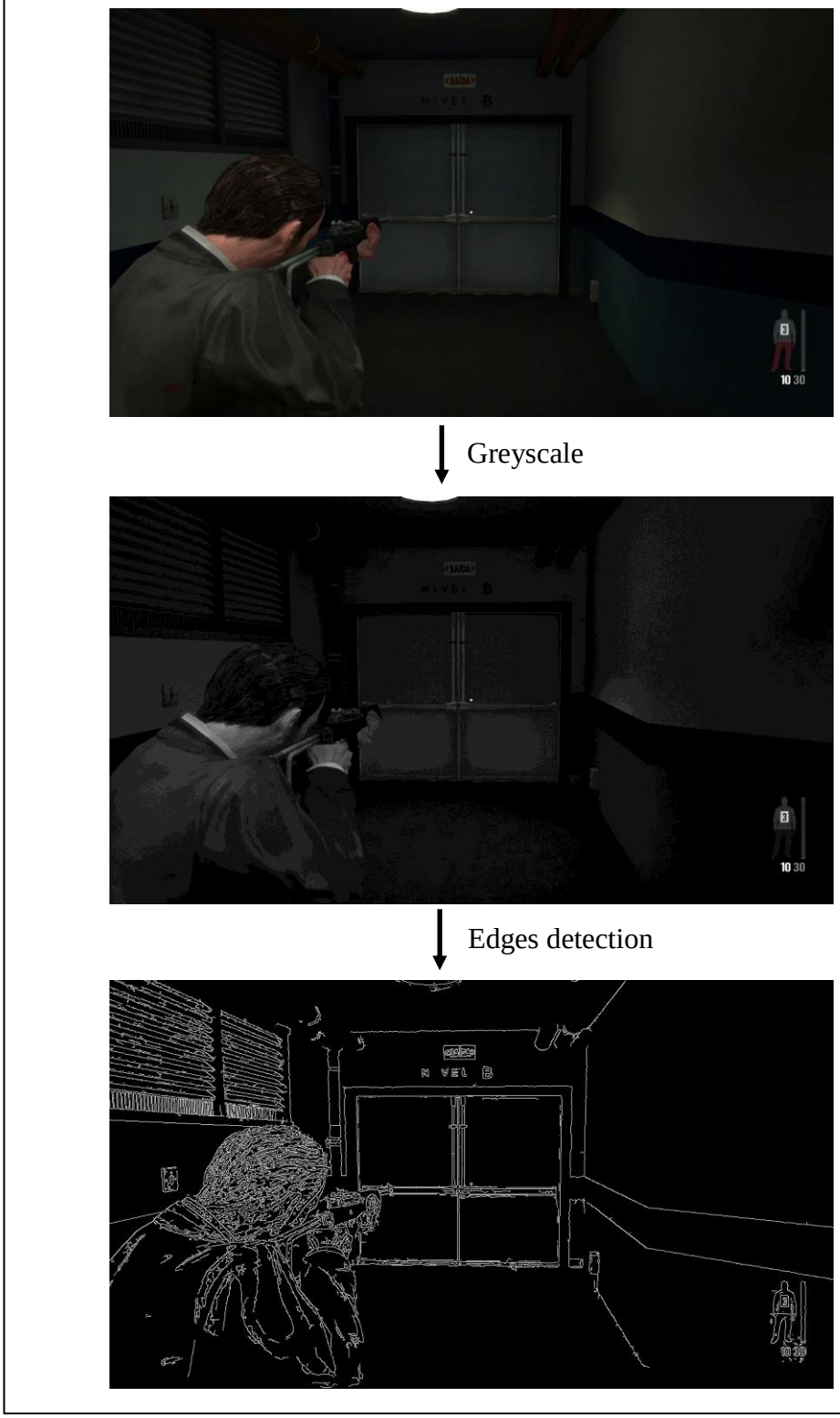


Figure 45. Reference image to edge representation

Figure 45 illustrates the process of converting the reference image into its edge representation, in order to obtain the shape information about the logo of interest. First, the reference image is converted into greyscale to simplify the edge detection process, and then the canny method is applied to extract the shape outlines. The HUD health logo (bottom right corner of the frame) appears clearly, even if initially semi-transparent.

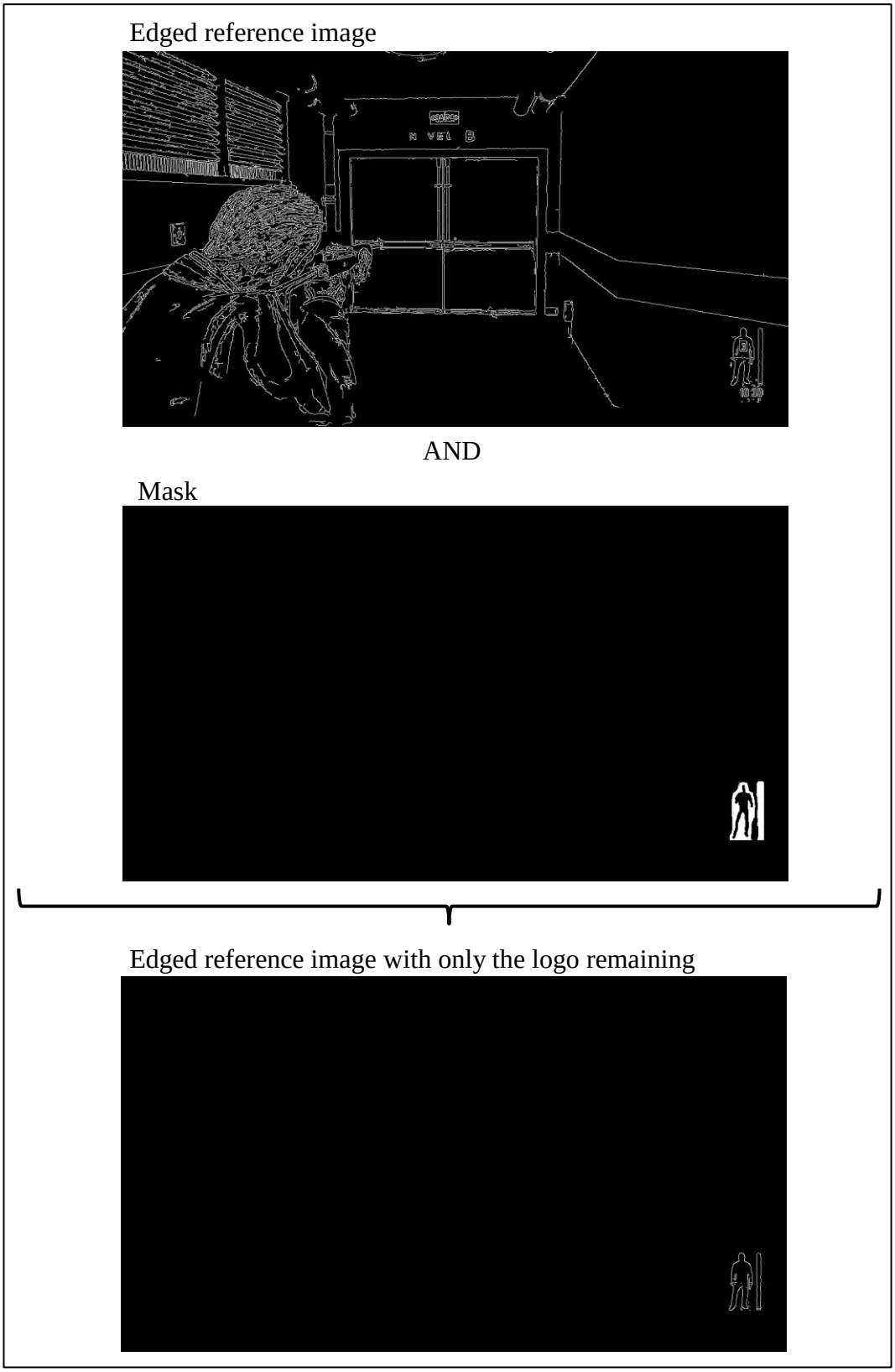


Figure 46. Masking process

Figure 46 illustrates the use of the mask image. By using the Boolean AND operation between the mask image and the edge version of the reference image, only the logo of interest remain visible.

Once the edge version of the logo is created, an edge image is generated by using the same canny edge detection method [SL 10-13; 22-23][19] for each video frame [input 1]. The frame and the reference image can then be compared using a method for computing *relative distance*. For each white pixel (1) in the reference image, the distance between the pixel value and the matching pixel (i.e. pixel at the same position) value in the frame image is computed, resulting in a 0 if the two pixels are identical (so both white), or 1 if the two pixels fail to match. The distances are summed and then divided by the total numbers of white pixels (1) in the reference image [SL 25-28]. The closest the result is to 0, the stronger the two images match, and the logo has been detected successfully. The relative distance can be summarized as:

$$rNorm = \frac{\sum_{x=0,y=0}^{x=w,y=h} (edgedReference(x,y)\ AND\ |edgedAveraged(x,y) - edgedReference\ (x,y)|)}{\sum_{x=0,y=0}^{x=w,y=h} edgedReference(x,y)}$$

Figure 47 illustrates the process by using a diagram schematizing a logo reference image, a mask image, and two video frames; one containing the logo, the other not. The reference image is first masked (as in the example of Figure 46) and then compared with the two frame examples. In the first one, two pixels that are white in the reference image are black in the frame (over a total of nine white pixels in the reference image) giving a relative distance of 0.22. In the second example, six pixels are changing, giving a bigger relative distance of 0.60. Figure 48 shows the relative difference step using the game *Max Payne 3*, giving an error of 0.2.

[19] The lines 14-21 will be further explained later in the paragraph about time-averaging

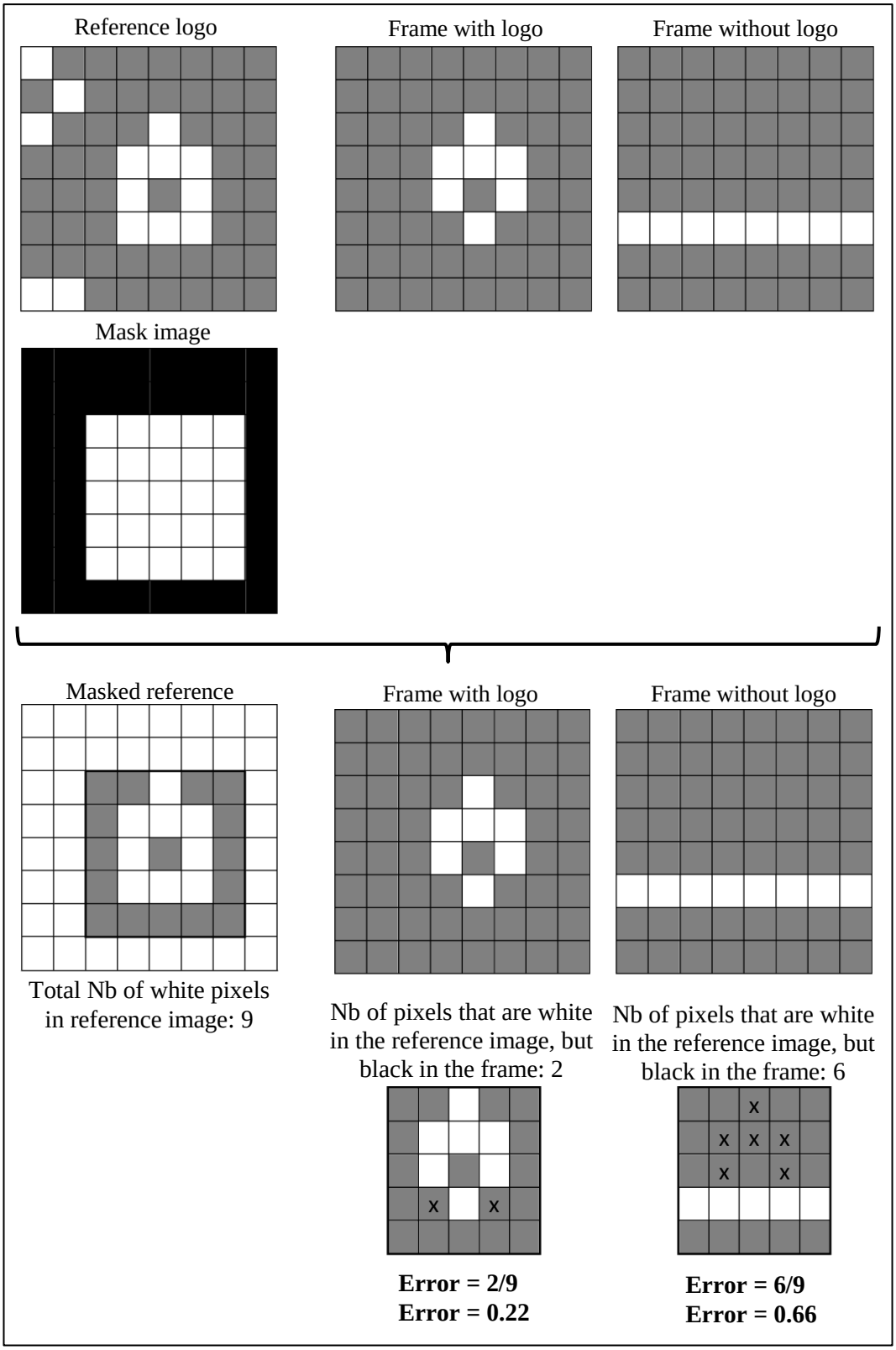


Figure 47. Logo detection steps

Figure 47 illustrates the logo detection algorithm using diagrams schematizing a logo reference image, a mask image, and two video frames. After first being masked, the reference logo image is compared to the two frames using a relative distance measure. For the frame with logo, two pixels that are white in the reference image are black in the frame; and six pixels for the frame without logo. The error distance is 0.22 for the frame with logo, and 0.66 for the one without (the closest to 0, the best the detection).

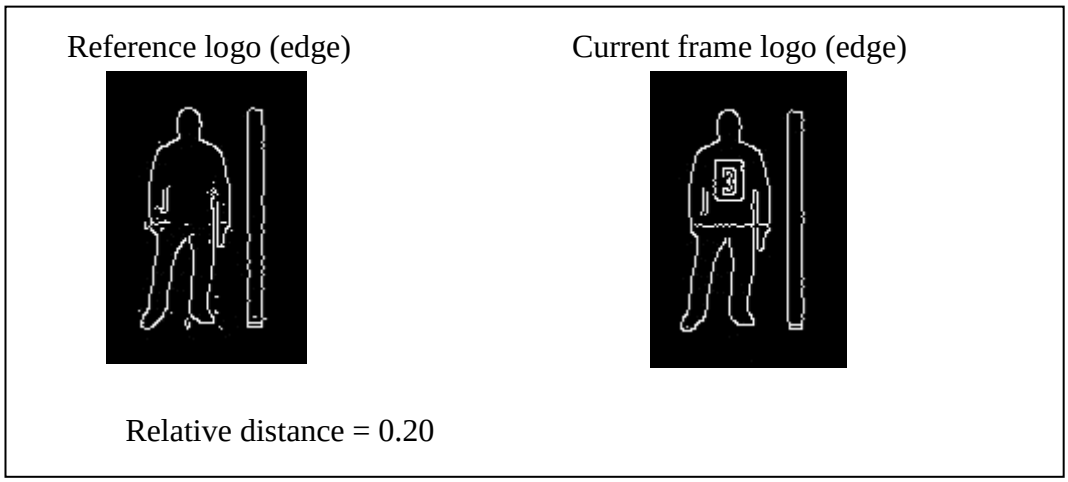


Figure 48. Relative distance result

Figure 48 illustrates the relative distance measure using the health logo detection of Max Payne 3. The number of white pixels in the reference logo that turn into black pixels, divided by the total number of white pixels in the reference image gives a score of 0.20.

The relative distance is actually representative of an error value; with 0 indicating no error and 1 no match. Values between 0 and 1 represent a "difference" ratio. This is where the error tolerance value [input 4] is employed. When the relative distance result is below the error tolerance value, the detection is accepted. If the value is above, the detection is rejected [SL 31]. In Figure 47, for instance, using a tolerance value of 0.3 will accept the first example and reject the second. In Figure 48, a tolerance value of 0.3 will accept the HUD as being displayed.

Once all the frames have been processed, the algorithm returns a series of binary values, 0 for no logo detected or 1 for logo detected; time-stamped using the time position of the frame [SL 30-34]. Figure 49 shows the tolerance value applied to a series of results once all the video frames have been processed. Every time the curve goes under the error tolerance value, then the detection is accepted (1). All the other values are rejected (0).

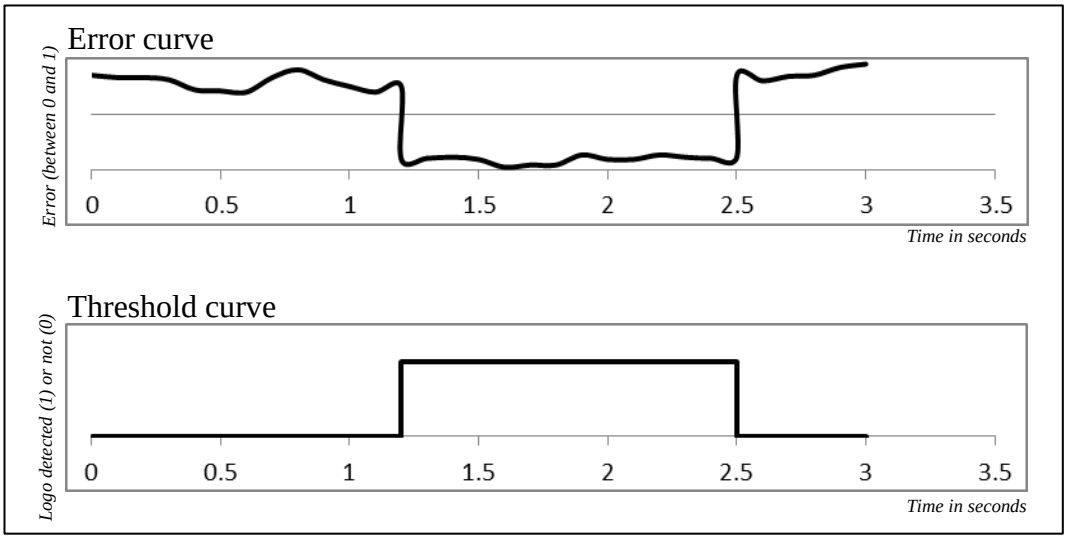


Figure 49. Use of threshold value

Figure 49 illustrates the use of a threshold value over a curve. All the values above the threshold value are set to 1, and all the values below the threshold value are set to 0. Using a threshold value makes the result easier to interpret, by distinguishing the detections from the non-detections.

To improve the edge detection of static information, Dos Santos et al. propose an intermediate step, called time-averaged image. The idea of time-averaging is crucial and drastically improves the accuracy of the method [SL 15-20]. Dos Santos et al. define time-averaging (2006, p. 3) as

$$averaged_t(x,y) \leftarrow \frac{averaged_{t-1}(x,y) + frame_t(x,y)}{2}$$

(with x, y being each pixel positions)

Time-averaging means that each frame becomes a mix of the current frame and the previous ones, with each previous frame having less and less weight as they become older. The implication of time-averaging is that everything that is moving in the video is likely to be blurred (not generating any edges), while everything static is reinforced (generating strong edges). In the case of static information detection, the time-averaging step highly improves the accuracy of the detection as the static information becomes clearer, while everything else is blurred. It is interesting to note that a high frame-step is likely to give better time-averaging results, as the compared frames will have more motion between them, meaning that more non-static elements will be blurred. Figure 50 illustrates time-averaging using the *Max Payne 3* example. In the upper example, no time averaging has been executed. The edge result is highly noisy. In the bottom example, time averaging has been executed, making the frame blurrier, but the edge image less noisy, with the HUD appearing more salient.

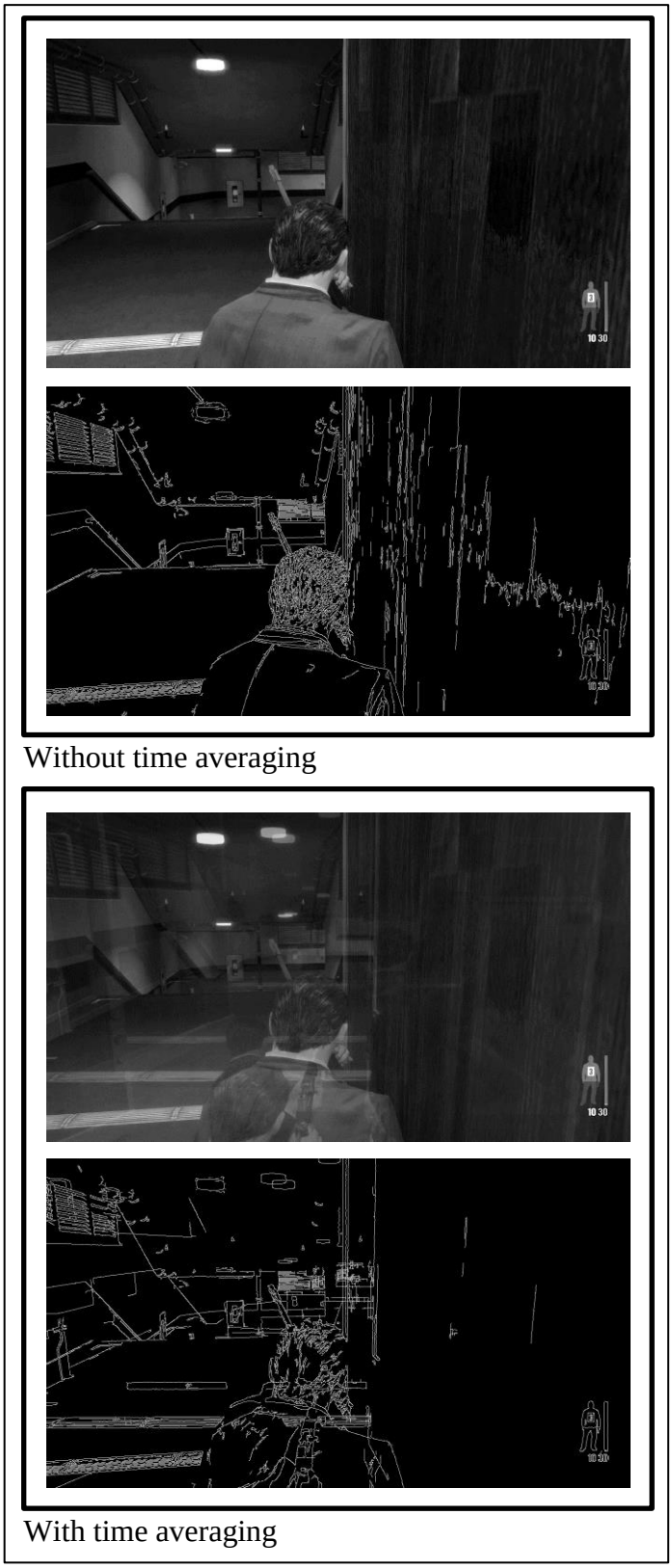


Figure 50. Time averaging

Figure 50 illustrates the time averaging step as proposed by Santos et al. Without time averaging (top example), the frame is clearer, meaning that all the edges are detected. With time averaging (bottom example), the frame is blurrier, except for the static information. Then, fewer edges are detected, and the remaining ones represent the information that remains still. Time averaging highly improves the detection of static objects.

Finally, once the algorithm has produced the binary detected/non-detected result, it is possible to specify a time tolerance value [input 5], in order to deal with false alarm or missed-detection that can occur during short periods of time (a couple of frames). The idea is to consider that logo detection during a period of time shorter than the time tolerance is a false alarm (a HUD appearing only for half a second is unlikely) and should be ignored; and that a "gap" in the detection occurring during a period of time shorter than the time tolerance is a short missed-detection (a HUD disappearing for less than half a second is also unlikely). For that, the two operations *closing* and *opening* derived from morphology operators (Gonzales & Woods, 2007, Chapter 9; Jain, 1986, pp. 387–389) and adapted for one dimension are used [SL 36-38]. Figure 51 illustrates the use of the morphology operators on an example curve; the *closing* operator filling the short gaps, and the *opening* operator erasing the short peaks.

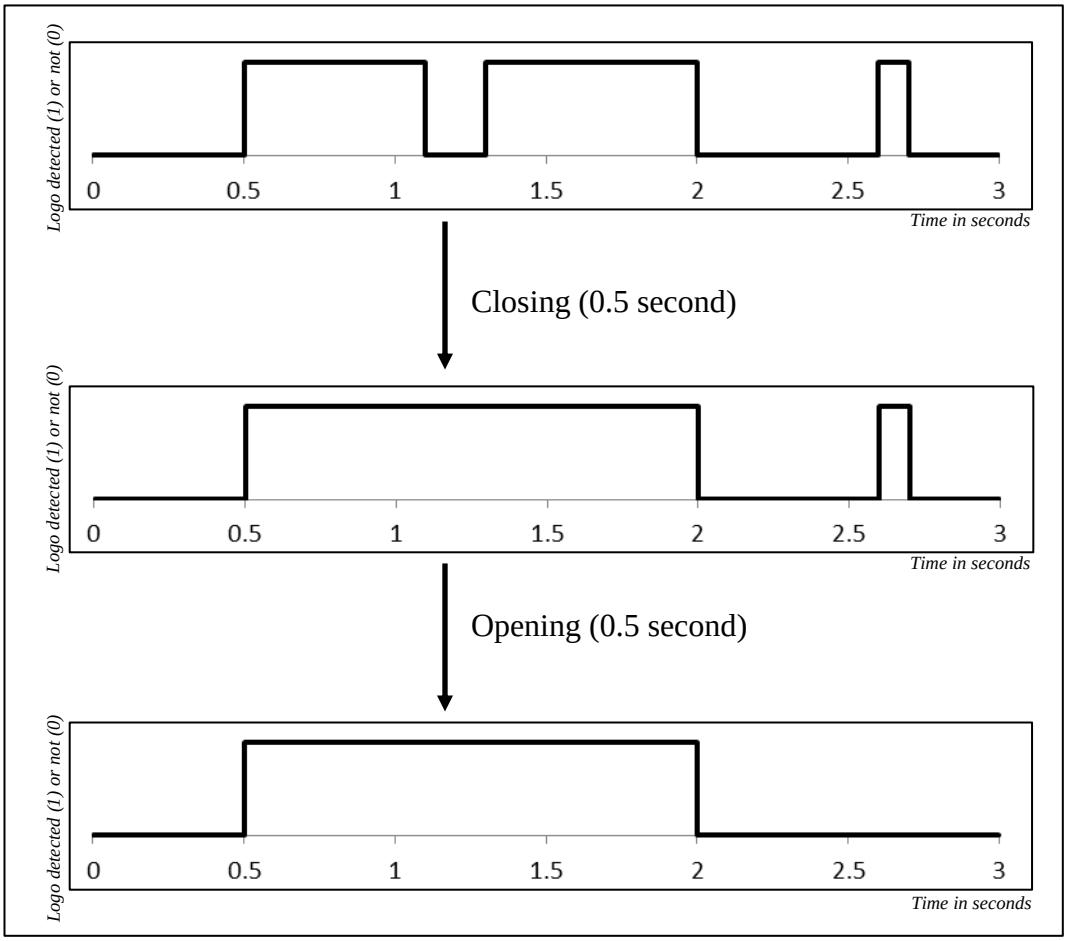


Figure 51. "Morphology" operator on vector

Figure 51 illustrates the use of morphology operators to reduce false-alarm and missed detection. The closing operation removes the short no-detections (for instance, it is unlikely to have a logo disappearing for only 0.5 second), and the opening operation removes the short detections (for instance, it is unlikely to have a logo only appearing for 0.5 second).

5.3.3 *Static Logo (SL) and Vector Morphology (M) algorithms*

Static logo detection (SL)

Inputs

gameFootage: video /* *gameplay footage video* */
locationMask: binary image /* *exact area where the information should be found* */
referenceImage: image /* *screenshot (size of a gameplay video frame) containing the information to be found* */
errorTolerance: float /* *percentage (between 0 and 1) of tolerated error* */
timeTolerance: float /* *time under which a sequence of detections is considered as false alarm and under which a sequence of non-detections is considered as a missed-detection* */
frameStep: integer /* *video frame increment* */

Output

result: vector of {timestamp: float, detection: boolean} /* *detection or not, per timestamp* */

Variables

n: integer /* *number of frames in the game footage* */
i: integer /* *loop variable* */
h, w: integer /* *frame dimensions* */
x, y: integer /* *pixel coordinates* */
currentFrame: image /* *frame currently processed* */
timeAveragedImage: image /* *time averaged frame* */
edgedReference: binary image /* *edge version of the reference image* */
edgedMask: binary image /* *edge version of the reference image, masked by the location mask* */
edgedAveraged: binary image /* *edge version of the time averaged frame* */
rNorm: integer /* *relative distance between the edgedReference and the edgedAveraged pixels, masked using the edged mask* */

Algorithm

```
1   /* Creation of the edge version of the reference image containing the logo to detect */
2   referenceImage ← greyscaleConversionOf(referenceImage)
3   edgedReference ← edgeDetection(referenceImage, METHOD_CANNY)
4
5   /* Masked version of the edgedReference */
6   edgedMask ← edgedReference AND locationMask
7
8   n ← nbFrames(gameFootage)
9
10  /* Sequentially grab the frames of interest from the gameplay footage */
11  for each position i between 0 and (n − 1), using a step of frameStep
12    currentFrame ← gameFootage(i)
13    currentFrame ← greyscaleConversionOf(currentFrame)
14
15    /* Time Averaging, to blur moving areas, and enhance static areas */
16    if (i = 0)
17      timeAveragedImage ← currentFrame
18    else
19      for each pixels (x, y) in currentFrame
20        timeAveragedImage(x, y) ← (timeAveragedImage(x,y) + currentFrame(x,y)) / 2
```

21

22 /* Edge version of the timeAveragedImage */

23 $edgedAveraged \leftarrow edgeDetection(timeAveragedImage, METHOD\_CANNY)$

24

25 $(w, h) \leftarrow getSize(edgedTimeAveraged)$

26

27 /* Relative norm between the edgedAveraged image, and the edgedReference one,
meaning the count of non-edge pixels supposed to be part of an edge,
masked by edgedMask,
divided by the total number of edge pixels in the mask */

28 $rNorm \leftarrow \frac{\sum_{x=0,y=0}^{x=w,y=h} (edgedReference(x,y)\ AND\ |edgedAveraged(x,y) - edgedReference\ (x,y)|)}{\sum_{x=0,y=0}^{x=w,y=h} edgedReference(x,y)}$

29

30 /* RelativeNorm is an error percentage. If relativeNorm is below the errorTolerance,
then a detection is accepted. If not, the detection is rejected. */

31 $\boldsymbol{if}\ (rNorm <= errorTolerance)$

32 $\quad result.push\left(\frac{i}{getFrameRate(gameFootage)}, 1\right)$

33 $\boldsymbol{else}$

34 $\quad result.push\left(\frac{i}{getFrameRate(gameFootage)}, 0\right)$

35

36 /* Morphology-like operator, on vector, to remove too short sequences of detections,
and too short sequences of non-detections */

37 $result \leftarrow vectorMorphology(result, timeTolerance, MORPHO\_CLOSE)$

38 $result \leftarrow vectorMorphology(result, timeTolerance, MORPHO\_OPEN)$

Vector Morphology

Inputs

```
/* vector on which the morphology will be applied */
inputVector: vector of {timestamp: float, detection: boolean}
morphTime: float /* as the morphology is time-based, morphologyTime indicate
                    power of the morphology to be applied */
morphType: enum {MORPH_DILATE, MORPH_ERODE, MORPH_OPEN, MORPH_CLOSE}
/* Type of morphology to be applied
MORPH_DILATE for dilatation
MORPH_ERODE for erosion
MORPH_OPEN for opening
MORPH_CLOSE for closing */
```

Output

```
/* inputVector once the morphology is applied */
result: vector of {timestamp: float, detection: boolean}
```

Variables

```
n: integer /* vector size */
i, j: integer /* loop variables */
beginTime: float /* first timestamp */
endTime: float /* last timestamp */
currentTime: float /* current considered timestamp */
morphValue: boolean /* value to expend, regarding the morphology,
                       1 for dilatation, 0 for erosion */
```

Algorithm

```
1   if (morphType = MORPH_OPEN)
2     /* Opening is an erosion followed by a dilatation */
3     result ← vectorMorphology(result, timeTolerance, MORPH_ERODE)
4     result ← vectorMorphology(result, timeTolerance, MORPH_DILATE)
5   else if (morphType = MORPH_CLOSE)
6     /* Closing is a dilatation followed by an erosion */
7     result ← vectorMorphology(result, timeTolerance, MORPH_DILATE)
8     result ← vectorMorphology(result, timeTolerance, MORPH_ERODE)
9   else
10    result ← copyOf(inputVector)
11    n ← sizeOf(inputVector)
12
13    /* beginTime and endTime are useful to avoid exceeding the vector size */
14    beginTime ← inputVector(0).timestamp
15    endTime ← inputVector(n − 1).timestamp
16
17    if (morphType = MORPH_DILATE)
18      morphValue ← 1 /* In the case of dilation, the value to expand is 1 */
19    else if (morphType = MORPH_ERODE)
20      morphValue ← 0 /* In the case of erosion, the value to expand is 0 */
21
22    /* Morphology time is considered as half before the current timestamp,
         and half after */
23    morphologyTime ← morphologyTime / 2.0
24
25    for each position i between 0 and (n − 1)
26      if (inputVector(i).detection = morphValue)
```

```
27          currentTime ← inputVector(i).timestamp
28
29          /* Expand to the past */
30          if ((currentTime − morphTime) ≥ beginTime)
31              j ← i − 1
32              while ((currentTime − morphTime) < inputVector(j).timestamp)
33                  resultVector(j).detection ← morphValue
34                  j ← j − 1
35
36          /* Expand to the future */
37          if ((currentTime + morphTime) ≤ endTime)
38              j ← i + 1
39              while ((currentTime + morphTime) > inputVector(j).timestamp)
40                  resultVector(j).detection ← morphValue
41                  j ← j + 1
```

5.3.4 *Result*

The following illustrations outline different results using the *static information detection* algorithm. Each result shown is linked to a different segmentation layer. Figure 48 illustrates the detection of the main menu of *Bioshock 2*, through the assessment of the title logo presence. This demonstrates how the static information detection approach can indicate the instigation of the game world instances segmentation layer (when the player leaves the main menu, they then instantiate a new game world).

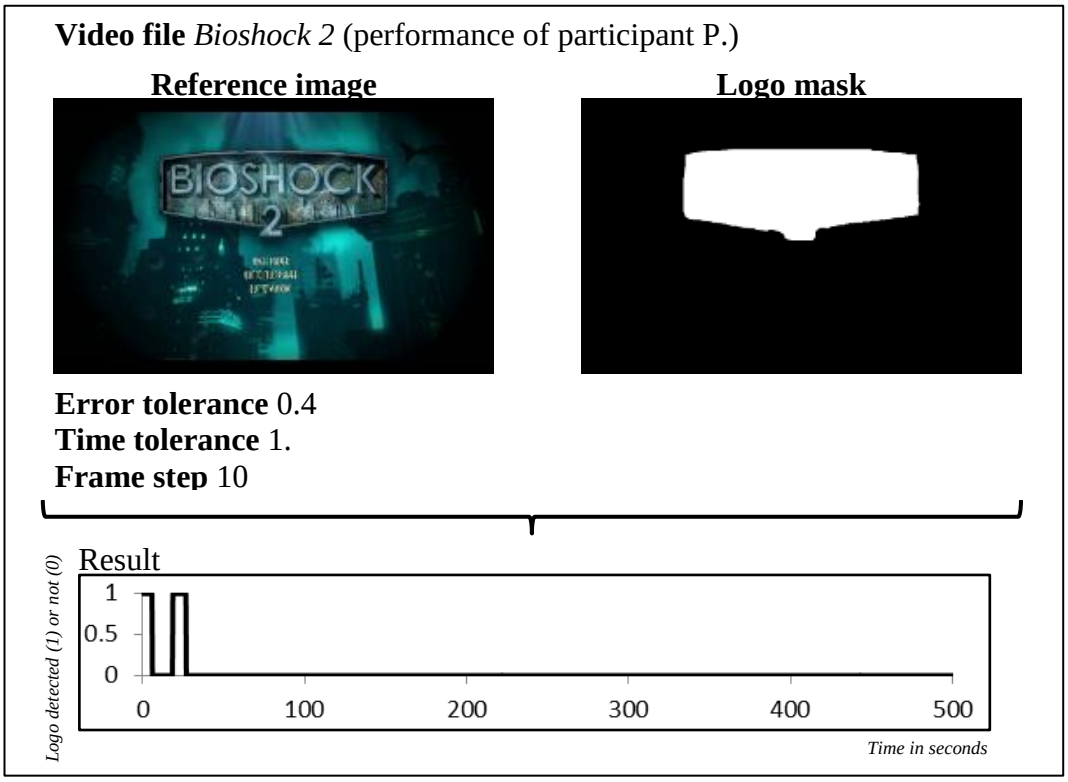


Figure 52. Main menu detection in *Bioshock 2*

Figure 52 illustrates the detection of the main menu of Bioshock 2 using the main "BIOSHOCK 2" logo appearing in the background. When the player exits the main menu, they are then inside a new instance of game world created for them. The result displayed in Figure 52 can be seen as being part of the game world instance segmentation layer.

Figure 53 illustrates the detection of mission screens in *Battlefield 3*, using the word "MISSION" text as a reference image. A mission-loading screen appears between each mission, signifying progression and also immediately after screen-death, reloading and sending the player back for a re-try. This result is linked to the spatial temporal segmentation layer.

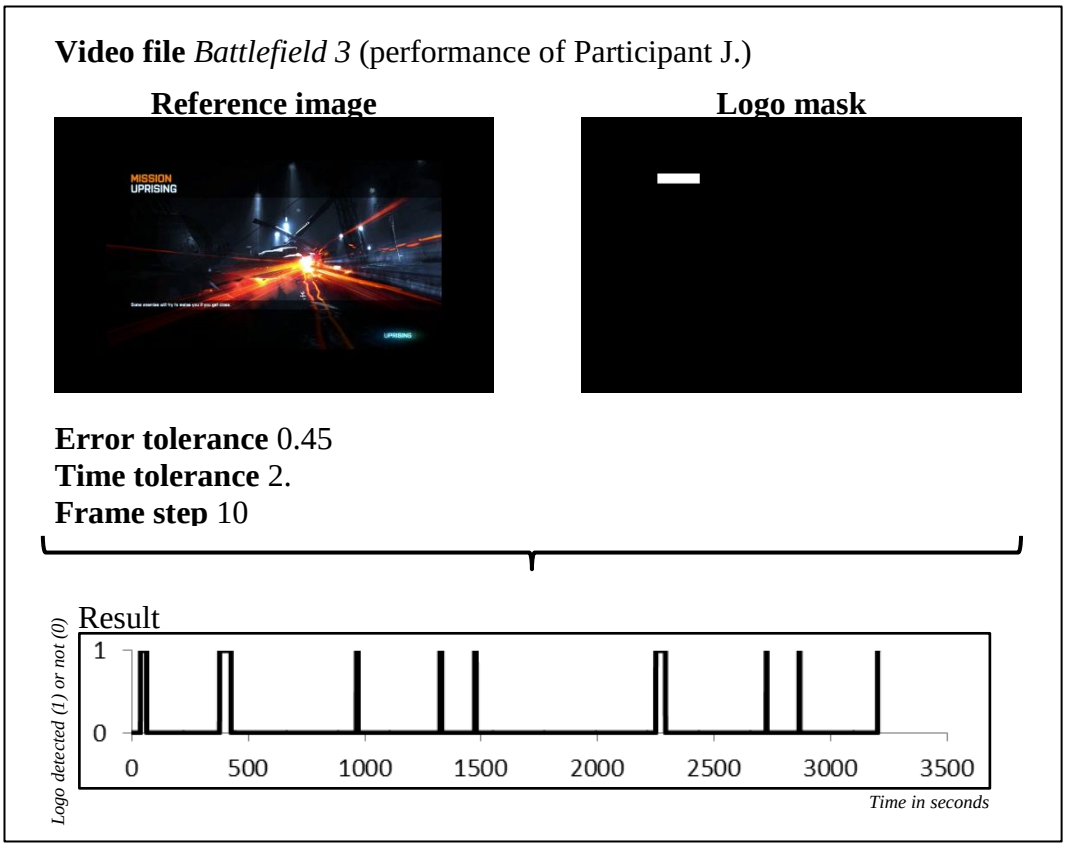


Figure 53. Mission screen detection in *Battlefield 3*

Figure 53 illustrates the detection of mission screen in Battlefield 3, through the detection of the "MISSION" text on the upper left corner of the screen. A mission screen appears between two missions, and after each death. The result displayed in Figure 53 can be seen as linked to the spatial temporal segmentation layer.

Figure 54 illustrates the result of the example throughout Section 5.3, seeking to detect the presence or absence of the HUD in *Max Payne 3* (through the life bar detection). In *Max Payne 3*, the HUD information alone is not enough to delimitate a space (the in-game segment can contain moments with, or without, HUD). However, the HUD disappears each time some agency is removed from the player (cut-scene or slow-motion effect). The HUD on-screen is indicative of a full-interactivity segment. The result in Figure 54 is then linked to the degree of freedom segmentation layer.

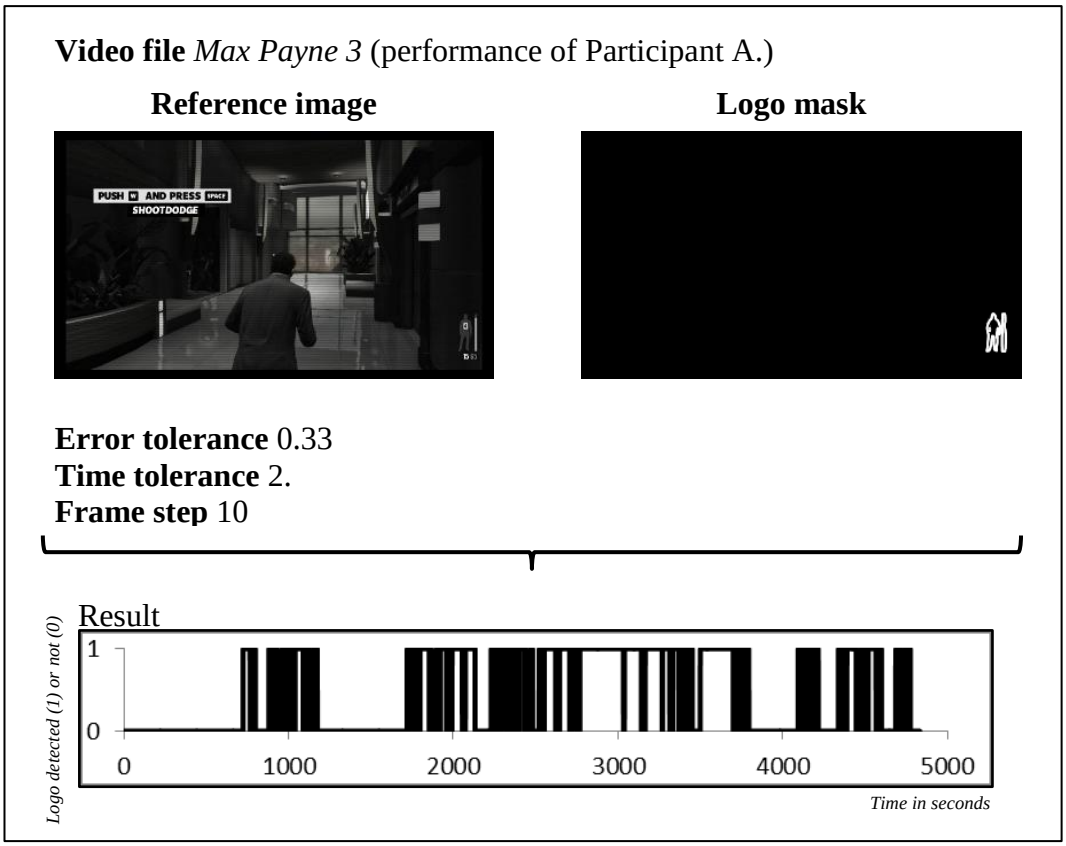


Figure 54. HUD Detection in *Max Payne 3*

Figure 54 displays the HUD detection result on the example used throughout Section 5.3. The presence of HUD is indicative of a full-interactivity segment, while the absence of HUD is indicative of sequences where part of the player degree of interaction is reduced. The result in Figure 54 can be seen as linked to the degree of freedom segmentation layer.

Finally, Figure 55 illustrates the detection of sequences in the game Dead Island when the player restores avatar health via the application of a first aid kit. Activating this process results in a first aid icon appearing in the upper-right corner of the screen. This represents feedback of player interaction, seeking to restore the avatar health before further fighting sequences. The result in Figure 55 is linked to the interaction segmentation layer.

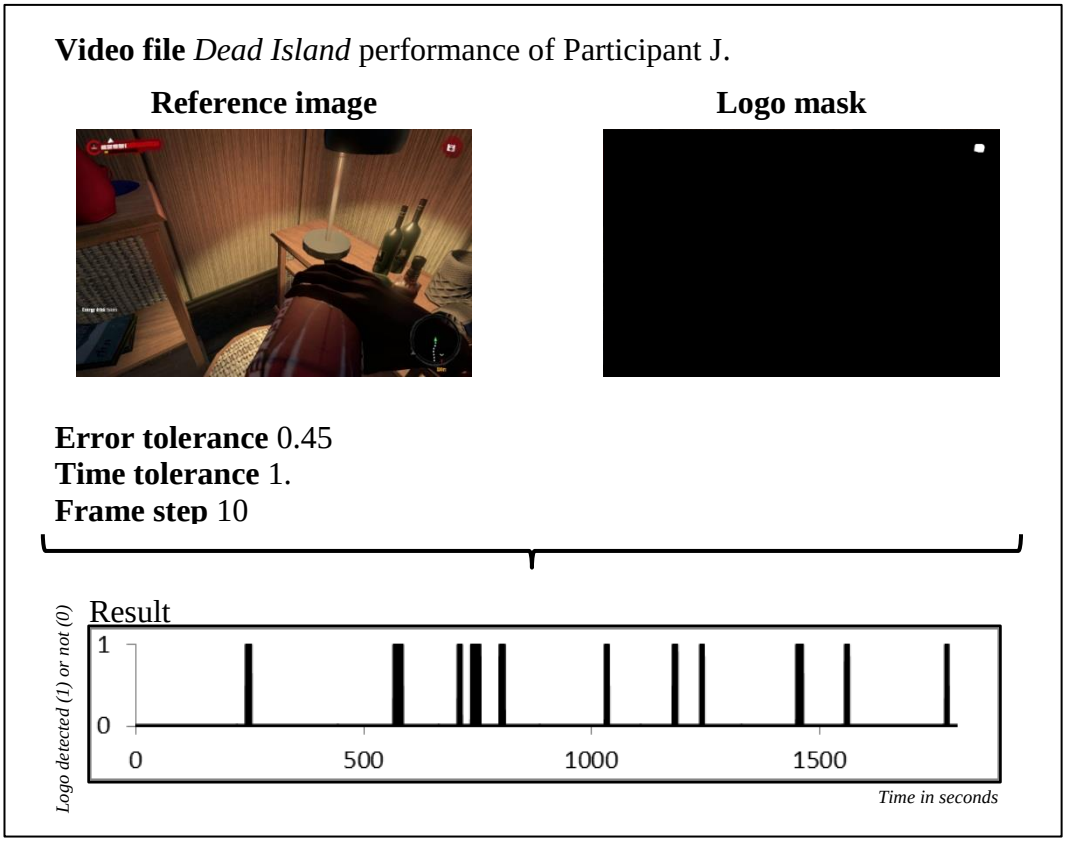


Figure 55. First aid kit used in *Dead Island*

Figure 55 illustrates the detection of the first aid used logo in Dead Island, indicative of sequences when the player decided to cure their avatar in order to be ready for the upcoming fights. The result in Figure 54 can be seen as linked to the interaction segmentation layer.

5.3.5 *Validity*

Dos Santos et al. state that their algorithm achieves a 100% correct recognition rate when applied to TV-Stream processing (Santos & Kim, 2006). Part of the explanation for such an accurate result is to be found in the fact that a TV-Channel logo remains on screen for an extended period of time, improving the detection, due to the image-averaging step. For gameplay performance segmentation processing, some elements are similar to the TV-Channel logo, in the sense that they remain on-screen for an extended period of time. This is, for instance, the case for the HUD main elements or menu titles. However some static information can appear during less than a couple of seconds (feedback icons, possibility to do an action in a reduce period of time), impacting on the processing accuracy. In *Bioshock 2* for example, the algorithm has been used to detect the looting panel, indicative of the player trying to collect items from a corpse. When the algorithm result is compared with the hand-coded version, a couple of detections have been missed. Examination of the footage revealed that these instances have been missed both because of the short period of time during when the player watches the corpse (therefore lowering the time where the looting logo is on screen), and because the logo, yellowish, can sometime appear over a yellow background (making the edge recognition less accurate). That means that the time tolerance value can impact the accuracy.

However, such detection failures are negligible, and do not compromise significantly the interpretation process, especially considering that static information usually remains on screen for an extended period of time, the time for the player to efficiently notice it.

5.4 Moving information detection

Section 5.3 presented a highly accurate algorithm to detect static information on screen, based on work initially developed for TV-Channel logo detection. The presence or absence of static graphical information, like HUD elements for instance, represents gameplay elements enhancing the understanding of player experience from gameplay performance recordings. However, there is a strong limitation attached to the method presented in Section 5.3: the considered elements need to be static, or at least appearing at a regular position. This limitation unfortunately prevents the detection of core gameplay elements like moving enemy position, directions to the next goal or map annotations. Section 5.4 proposes a solution, based on updating the algorithm presented in Section 5.3 by adding several steps to first locate the best potential location of the information, before comparing it using the static information algorithm.

5.4.1 *Approach overview*

When an element of interest is not static or fails to appear in the same screen position, it is necessary to execute an algorithm capable of dealing with the localization and detection of objects that change position on screen. Here, a strand of research termed *image registration* can be employed (Pratt, 1978, Chapter 19). This research executes algorithms that are efficient in identifying commonalities between two images thus highlighting how they relate to each other. Thus, a smaller object can be located with a larger image when its position is not guaranteed (i.e., to identify the position of a given logo in an image susceptible of containing the same logo). One method employed within this strand of research is *cross-correlation* method (Pratt, 1978, p. 553), also mentioned by Santos et al.

(2006). For Santos et al. the aim of using *cross correlation* is to avoid having to specify the believed position of TV-Channel logos, and then be able to directly use a logo database. In the work of Santos et al., the detected logo coordinates information is useless in terms of interpretation. However, for gameplay performance segmentation and player experience analysis, the coordinates of a displayed gameplay element can inform the analyst a lot, especially if the gameplay element also carries a sense of space. The approach presented in Section 5.4 is about adding cross-correlation to the algorithm presented in Section 5.3, in order to improve graphical gameplay element detection by also considering the on-screen position information as a meaningful result. The position information is indeed insightful for gameplay element. For instance, when a logo represents an enemy location, its coordinates indicate the enemy position regarding the avatar attention. If the logo is detected in the centre of the screen, it is then possible to conclude that the player is aiming at the enemy.

5.4.2 *Step by step description*

To execute detection of a non-static object, the algorithm employed requires the following inputs

1. The *gameplay footage* in the form of a video file (sound is not mandatory)
2. A *logo image*, representing the object that will be cross-correlated.
3. An *image mask* that can be used to restrict the search area for an object in cases where the object only occupies a smaller area on screen. If no mask image is specified then the whole screen is scanned.
4. Once the scanned image locates likely matches, then an *error tolerance* value will be employed to determine whether the object identified is a correct match.

5. A *time tolerance* value, specifying the minimum period of time that the object should remain on-screen.
6. A *frame step* that reduces the processing process by not requiring each individual frame to be processed, but instead every 2nd or 10th frame.

The algorithm functions similarly to the static one presented in Section 5.3, with the addition of several processing functions that account for the challenges of having to locate an object (or several objects) that may change position over time. These are used to register how to find the logo position and how to deal with the multiple logo possibility (for instance, a logo representing an enemy position is likely to appear several times on screen if several enemies are present simultaneously, and must then all be detected). The first step, as in Section 5.3, is to transform the logo image [input 2] into an edge representation using the Canny Edge Detection method (Canny, 1986). The reference image, unlike the static detection, is not a full reference frame as the position is unknown now, but represents only the logo (with smaller dimensions than the frame size). . Once the object image has been converted into a binary edge representation it is shifted sequentially through every possible position over the video frame (beginning in the upper left corner of the frame and ending in the bottom right corner[20]). Figure 56 illustrates, with diagrams representing a reference image and a frame, how the shift is performed, step by step.

---

[20] minus the logo dimensions

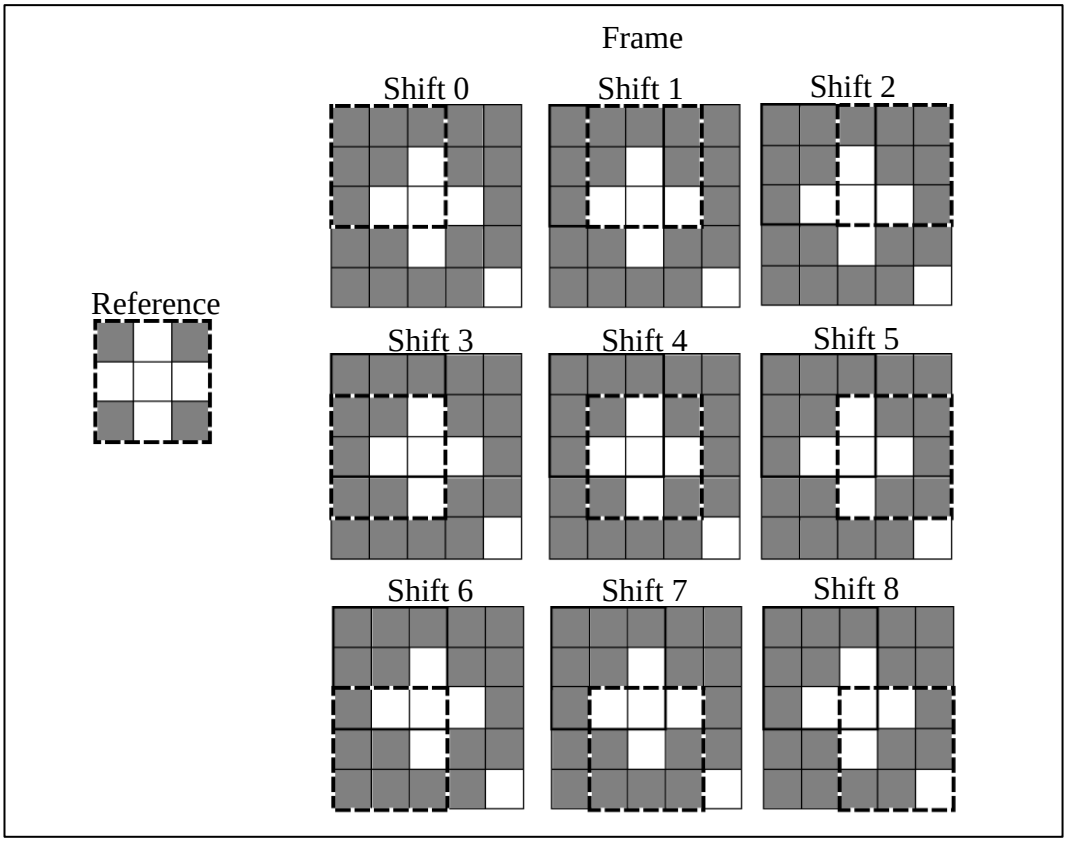


Figure 56. Logo shift

Figure 56 illustrates the process of shifting the logo over the frame; starting from the top-left corner, and ending to the bottom-right corner. For each shift position, a correlation score is calculated. In the above example, shift 4 is the one that will generate the highest correlation score.

For each shift position, a correlation value is computed using the normalized cross-correlation method (Fitzpatrick, Hill, & Maurer, Jr., 2000, p. 489), and stored with the matching coordinates values. A correlation matrix [ML 20-28] is used to store the correlation results, and can be displayed as an image in which each pixel value represents the result of the correlation process[21]. The biggest value (inside the image mask area), i.e. the whitest pixel in the correlation image, represents the position where the logo is the more likely to be [ML 36]. The normalized cross correlation formula is

$$ccorrImage(xc, yc) = \frac{\sum_{xl,yl}\big(edgedLogo(xl,\ yl).edgedFrame(xc + xl,\ yc + yl)\big)}{\sqrt{\sum_{xl,yl} edgedLogo(xl,\ yl)^2.\sum_{xl,yl} edgedFrame(xc + xl,\ yc + yl)^2}}$$

(with xl and yl representing the pixel positions inside the reference logo image)

[21] the pixel position matching the position in the frame image where the upper left corner of the reference logo should be translated to

Figure 57 illustrates the usage of a correlation image as a cross-correlation processing result. The reference logo is sequentially shifted, as in Figure 56, and each shifting position gives a correlation score that is stored into the correlation image (the pixel position in the correlation image represents where the upper left corner of the reference image should be translated in order to obtain this correlation score). In Figure 57 two perfect correlations are detected, at positions (1, 1) and (3, 4), where the pixels are the whitest. The correlation image dimensions are smaller than the frame, because each value represents the upper-left corner of the reference image, meaning that there are positions on the right and on the bottom where the logo cannot be translated without overcoming the frame boundaries.

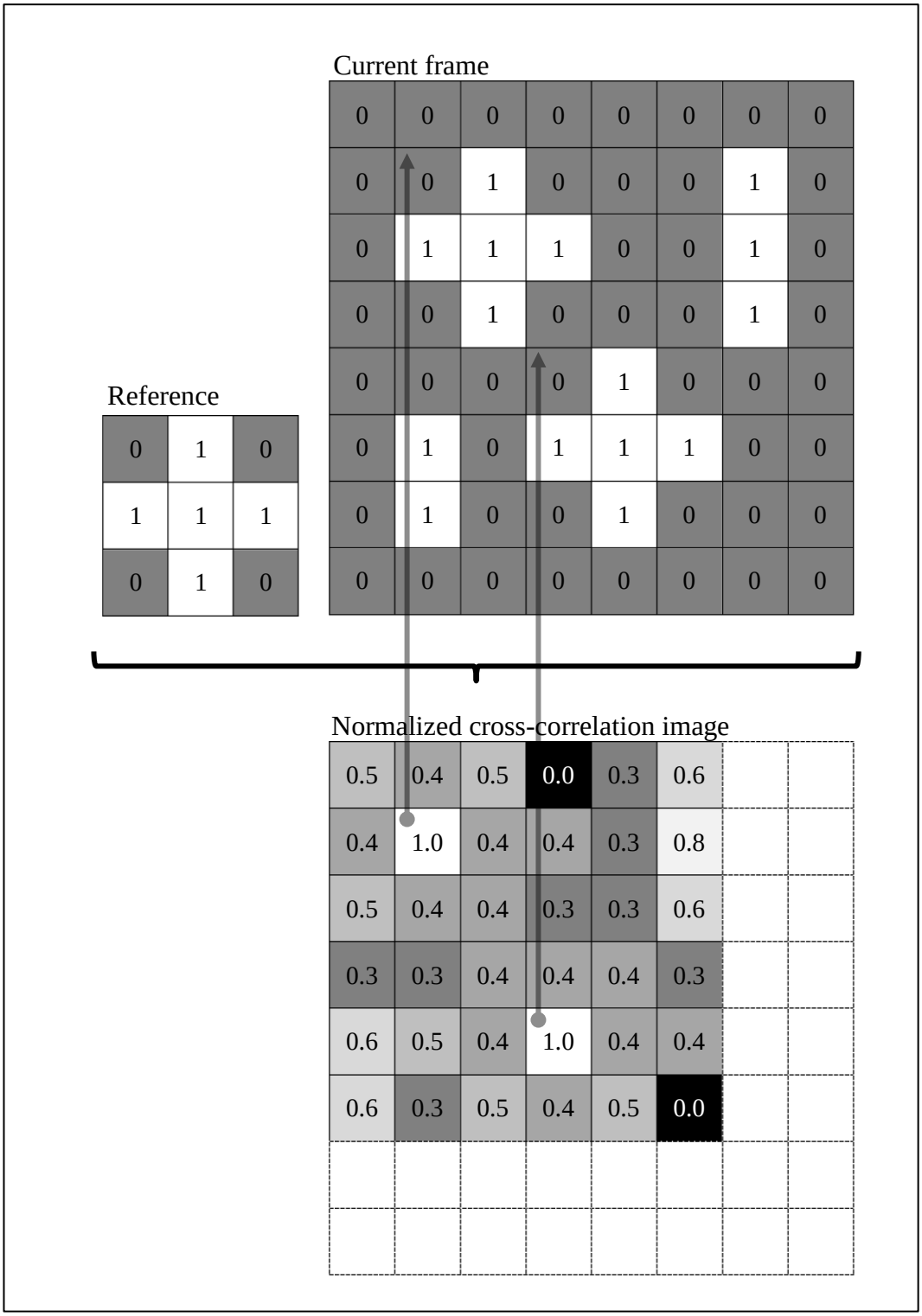


Figure 57. Cross-correlation matrix

Figure 57 displays a diagram representing a cross-correlation matrix. The cross-correlation matrix stores the different scores after each shift. A correlation matrix can also be displayed as an image; the whitest the pixel, the best the correlation score. Then, identifying the whitest pixels informs about the logo positions.

Figure 58 and Figure 59 illustrate the same process, but utilizing data from the game *Battlefield 3*. In this game, the player is sometimes presented with logos identifying out-of-sight enemy positions. These logos can be detected, and inform the analyst about the nature of the space (the player is in an environment where enemies can shot at him from a distance), about the number of enemies (number of logos) and about the player reaction (i.e., if the logo is in the centre of the screen, then the player is typically trying to aim at the enemy). Figure 58 illustrates the initial algorithm steps, from the video frame (a) to the edge version (b) and then to the correlation image (c), producing white pixels where the correlation score is high. Figure 59 is a bigger version of the correlation image in Figure 58, to identify more easily the different scores. This figure (rotated counter-clockwise), shows two perfect correlations in the upper-right corner

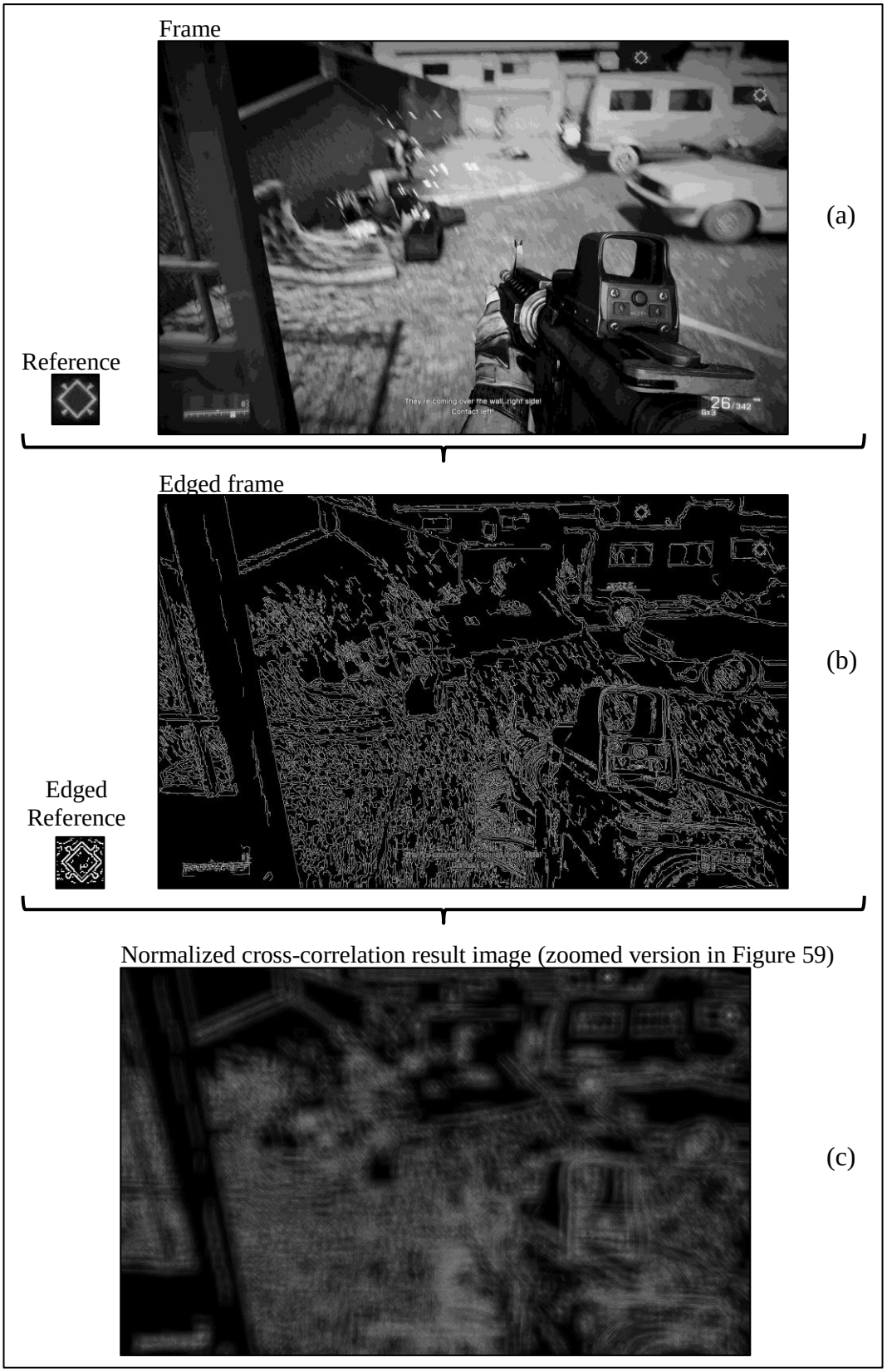


Figure 58. Cross Correlation image

Figure 58 illustrates the making of a cross-correlation image, using the enemy position logo detection in Battlefield 3.(a) represents the current frame and the logo reference image. Then, (b) illustrates the edge detection process. Finally, (c) shows the cross-correlation result, after the reference logo has been shifted over all the possible positions. Figure 59 displays a bigger version of the correlation image.

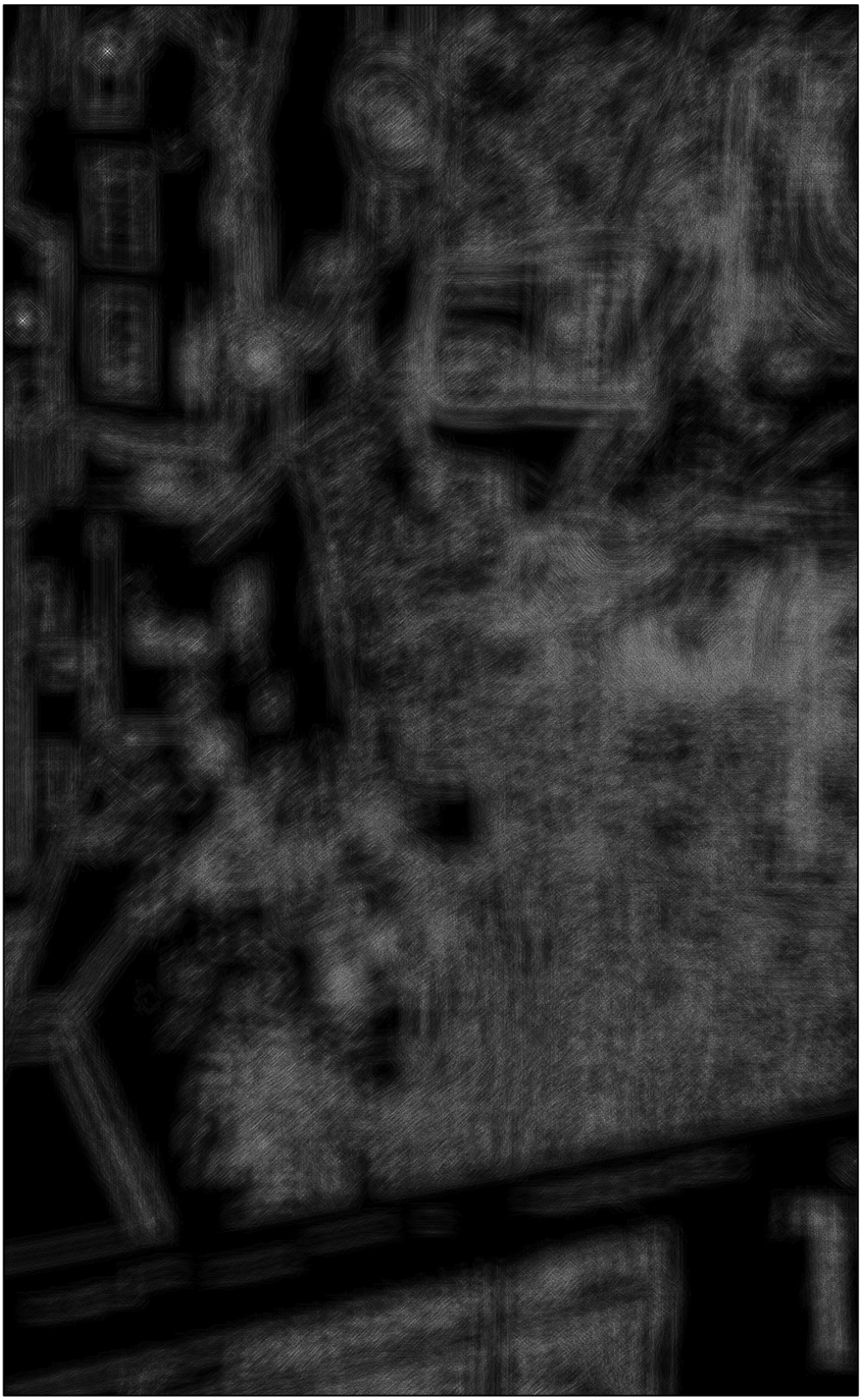


Figure 59. Cross Correlation image (Figure 58), zoom

Figure 59 is a zoomed version (rotated anti-clockwise) of the correlation image of Figure 58. Two bright pixels can be easily observed in the top-right corner of the image, matching the positions of two logos actually present in the current frame.

In order to obtain accurate results, two questions need to be addressed. The first one is about assessing if a version of the logo is actually present in the frame. Indeed, the correlation method will always give a highest score, but that does not mean that a logo is actually present; the score only represents the most likely position. To answer this question, the comparison step of the static information algorithm is used [ML 38-46].

The second question is about assessing if the logo is present in more than one version. If the relative distance test returns *false*, then there is no logo on screen. But if the test returns *true*, then at least one logo has been detected. However, the uniqueness is not guaranteed, as several similar logos can be simultaneously on screen. This is actually the case in the *Battlefield 3* example used in the current section. For addressing this question, the previous highest value in the correlation image is set to 0[22] to prevent the position to be reconsidered a second time, and the process is executed again until the logo detection algorithm return *false* [ML 45-56]. The result is a list of the frame time stamps, with the number of detected logos, and the coordinate position for each of them [ML 49; 58-59].

Figure 60, being the continuation of Figure 57, illustrates how the two questions are addressed by the algorithm. At step 1, the best correlation value is at position (1, 1). The matching area in the frame is extracted, and compared with the logo using the relative distance test. The error is 0 (perfect matching), so the logo

[22] Actually, several pixels around should also been set to 0 to fully avoid detecting again the already detected logo

finding continues, after discarding the detected logo by nullifying the best correlation pixel and its neighbours. Step 2 is similar, with the position (3, 4). At step 3 however, the best correlation is at (5, 1), and the comparison gives an error of 0.4, above the 0.25 tolerance value. The detection process stops, and give as a result two detections, of positions (1, 1) and (3, 4).

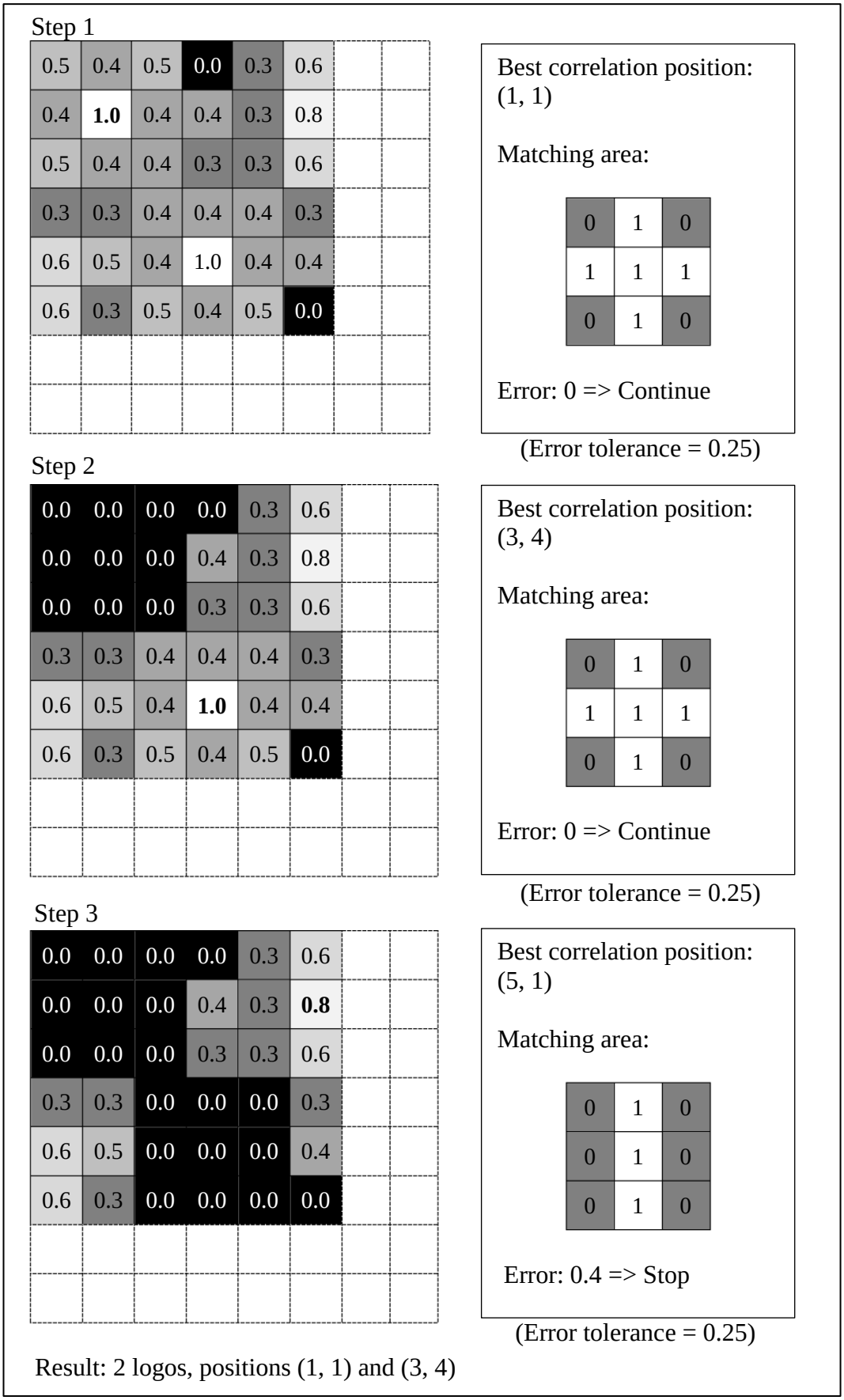


Figure 60. Multi logo detection

Figure 60 will be described in the next page

Figure 60 is the continuation of Figure 57, and illustrates the detection of potential multiple version of the same logo in the frame. At step 1, the best correlation score (in bold) is at position (1, 1). The matching area (same size as the reference image) is then extracted, and compared using the relative distance test. The result is 0, so the process is repeated, after first nullifying the position already processed. Step 2 is similar, with (3, 4). At step 3, the error is 0.4, above the error tolerance. The process stops here.

Figure 61 achieves the same result, but with the *Battlefield 3* as the example. The first whitest pixel is considered, and the matching area in the frame is compared with the logo. The error is 0.3, bellow the tolerance of 0.4, so the algorithm continues, after first discarding the best correlation position, as it has already been identified once. At step three, the detected area does not match the logo. The process stops, after finding the two enemy logos, at position (1585, 121) and (1267, 21). In Figure 61, the white arrows represent the currently processed position, and the red arrows represent the previous best score being discarded in order to not be processed a second time.

Three remarks should be made. First, there is no time averaging this time, as the logos are moving, and would then be blurred by such a step. Then, a time tolerance value can also be used [input 5] as in Section 5.3 to both discard detections that are too short, and the too short non-detection. Finally, the mask image [input 3] has not been presented in the present section to avoid confusing an already complex section. However, goal of the mask here is to select an area in which the logos can be found (like for instance delimiting a map area where position icons are likely to be). Then, the mask is applied to the correlation image, discarding (nullifying) the results that are outside the mask area [ML 28-29].

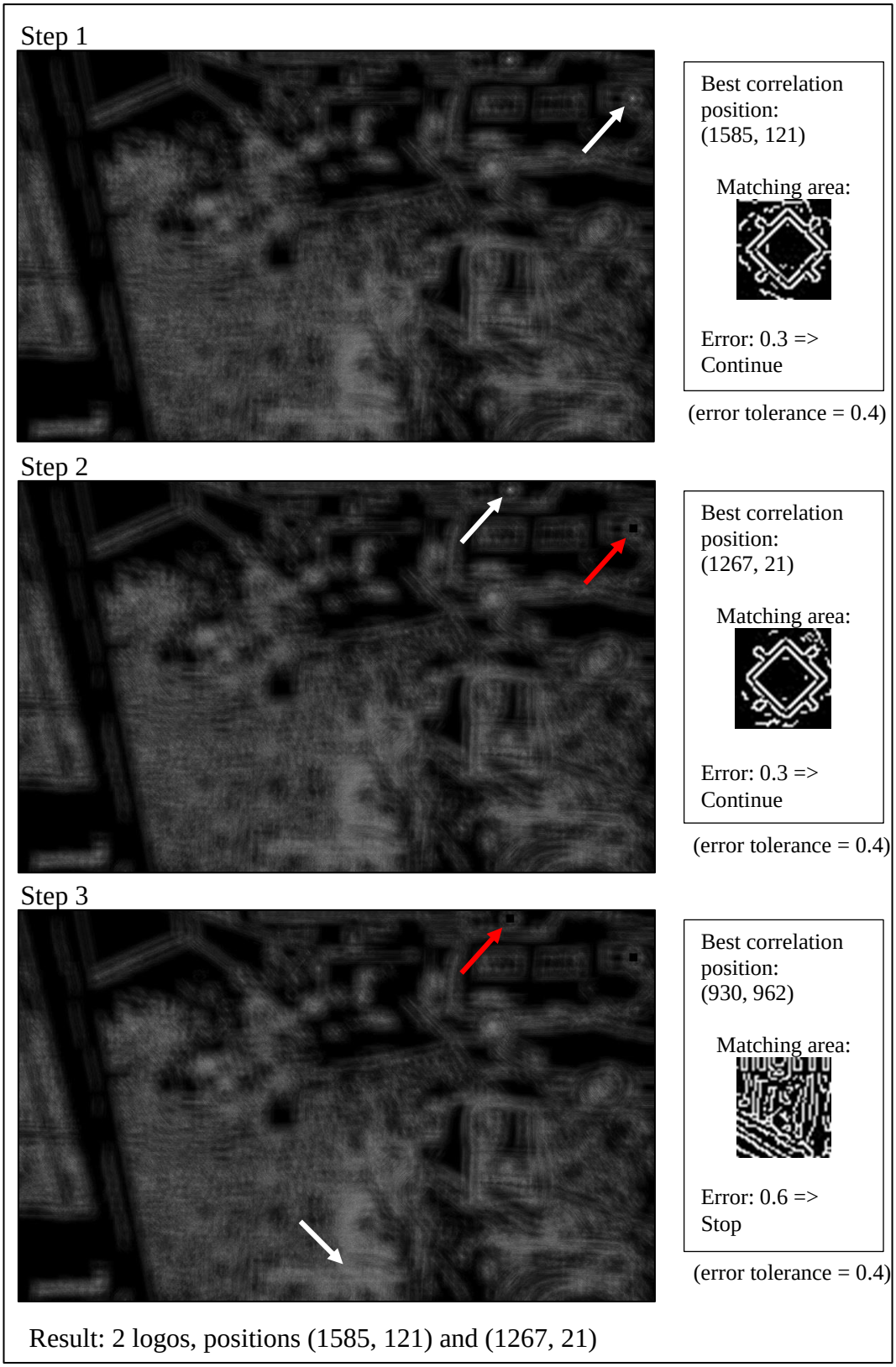


Figure 61. Multi logo detection on gameplay example

Figure 61 illustrates the multi-logo detection, using a real gameplay example. The white arrows represent the detected logo location, and the red ones represent the previous ones, then nullified before the next step.

5.4.3 *Moving Logo (ML) Algorithm*

Moving logo detection (ML)

Inputs

gameFootage: video /* *gameplay footage video* */
maskArea: binary image /* *area in which the logo is likely to be, in one or several versions* */
logoImage: image /* *moving information to find, in the form of a logo, smaller than a gameplay video frame* */
errorTolerance: float /* *percentage (between 0 and 1) of tolerated error to consider the information has being found* */
timeTolerance: float /* *time under which a sequence of detections is considered as false alarm and under which a sequence of non-detections is considered as a missed-detection* */
frameStep: integer /* *video frame increment* */

Output

/* *number of detected logos, and their positions, per timestamp* */
result: vector of {timestamp: float, nbDetections: integer, locations: vector of points}

Variables

n: integer /* *number of frames in the game footage* */
i: integer /* *loop variable* */
wl, hl: integer /* *logo image dimensions* */
wf, hf: integer /* *frame dimensions* */
wc, hc: integer /* *cross-correlation image dimensions* */
xl, yl: integer /* *logo pixel coordinates* */
xc, yc: integer /* *cross-correlation image pixel coordinates* */
currentFrame: image /* *frame currently processed* */
edgedLogo: binary image /* *edged version of the logo* */
edgedFrame: binary image /* *edged version of the current frame* */
ccorrImage: image /* *cross-correlation image* */
continueLogoSearch: boolean /* flag indicating if the logo search must continue or stop */
nbLogo: integer /* total number of logos detected in the current frame */
bestCorPos: point /* *current best correlation position* */
edgedReference: binary image /* *reconstituted edged reference image for relative distance* */
edgedMask: binary image /* *reconstituted edged mask for relative distance* */
foundLogoPositions: vector of points /* *list of all the detected logo positions in the current frame* */

Algorithm

1 $(wl, hl) \leftarrow getSize(logoImage)$
2 $(wf, hf) \leftarrow getSize\big(gameFootage(0)\big)$
3
4 /* *Edge version of the logo image* */
5 $logoImage \leftarrow greyscaleConversionOf(logoImage)$
6 $edgedLogo \leftarrow edgeDetection(logoImage, METHOD\_CANNY)$
7
8 /* *The size of the cross-correlation image result is (wf-wl, hf-fl), as each pixel represents the correlation value when moving the top left corner of the logo to this pixel location. For size consistency, the mask area image should be cropped accordingly* */
9 $maskArea \leftarrow cropImage(0, 0,\ wf - wl, hf - hl)$
10

```
n ← nbFrames(gameFootage)

/* Sequentially grabs the frames of interest from the gameplay footage */
for each position i between 0 and (n − 1), using a step of frameStep
  currentFrame ← gameFootage(i)
  currentFrame ← greyscaleConversionOf(currentFrame)
  /* Edge version of the current frame (no time averaging unlike the static version
     because the considered information is moving this time) */
  edgedFrame ← edgeDetection(currentFrame, METHOD_CANNY)

  /* Creation of the normalized cross-correlation image
     (the whitest a pixel in this image, the highest the correlation score) */
  ccorrImage ← createEmptyImage(wf − wl, hf − hl)
  (wc, hc) ← getSize(ccorrImage)

  for each pixels (xc, yc) in ccorrImage
    ccorrImage(xc, yc) = Σ_{xl=0,yl=0}^{xl=wc,yl=hc} (edgedLogo(xl,yl).edgedFrame(xc+xl,yc+yl))
                         / sqrt( Σ_{xl=0,yl=0}^{xl=wc,yl=hc} edgedLogo(xl,yl)² . Σ_{xl=0,yl=0}^{xl=wc,yl=hc} edgedFrame(xc+xl,yc+yl)² )

  /* Pixels outside the provided area are discarded */
  ccorrImage ← ccorrImage AND maskArea

  nbLogo ← 0
  continueLogoSearch ← true

  /* Unlike the static version, a logo can appear several time */
  while (continueLogoSearch = true)
    /* Position of the best correlation, i.e. pixel coordinates of where to translate
       the upper left corner of the logo in order to have the best similarity score */
    bestCorrPos ← maximumPixelLocation(ccorrImage)

    /* Creation of a reference image, like the one used in the static version,
       by creating an empty (black) image, and copying the logo to the bestCorrPos */
    edgedReference ← createEmptyImage(wf, hf)
    edgedReference ← copy(edgedLogo, bestCorrPos)

    /* Relative norm between the edgedFrame, and the edgedReference one,
       meaning the count of non-edge pixels supposed to be part of an edge,
       masked by edgedMask,
       divided by the total number of edge pixels in the mask */
    rNorm ← Σ_{x=0,y=0}^{x=w,y=h} (edgedReference(x,y) AND |edgedFrame(x,y) − edgedReference(x,y)|)
            / Σ_{x=0,y=0}^{x=w,y=h} edgedReference(x,y)

    /* RelativeNorm is an error percentage.
       If relativeNorm is below the errorTolerance, then a detection is accepted */
    if (rNorm <= errorTolerance)
      /* Because the information is moving, the location information
         is also meaningful, and needs to be stored */
      foundLogoPositions.push(bestCorrPos)
      nbLogo ← nbLogo + 1

      /* In order to consider the next best correlation score,
         the current best need to be discarded. For that, a black (0) rectangle
         area is created around the current best position
```

```
                (CANCEL_AREA_SIZE is arbitrary) */
53          drawFilledRectangle(ccorrImage,
                                bestCorrPos.x − CANCEL_AREA_SIZE/2,
                                bestCorrPos.y − CANCEL_AREA_SIZE/2,
                                CANCEL_AREA_SIZE,
                                CANCEL_AREA_SIZE,
                                BLACK_COLOR)
54      else
55          /* Logo discarded, the search must stop */
56          continueLogoSearch ← false
57
58    /* Add to the result list the current timestamp in second,
         the number of detected logos,
         and the list of logos positions */
59    result.push(i/getFrameRate(gameFootage), nbLogo, foundLogoPositions)
60
61  /* Morphology-like operator, on vector, to remove too short sequences of detections,
         and too short sequences of non-detections */
62  result ← vectorMorphology(result, timeTolerance, MORPHO_CLOSE)
63  result ← vectorMorphology(result, timeTolerance, MORPHO_OPEN)
```

5.4.4 *Result*

Figure 62 displays the result of the example discussed in Section 5.4 (the mask is all white, because the entire screen is considered). The logos represent enemy positions that can appear anywhere on the screen. The enemies (snipers) are distant enough to be out of sight of the player, but still able to shoot him. Detecting the logo can elucidate the moments when the player is aimed at by distant enemies, and even their number (one or two around 1000 seconds). The lower graph in Figure 62 shows the usefulness of the logo position information, indicating how the player has then responded to this information by adjusting their position, thus centring the logo position on-screen presumably to take aim and fire at the enemy There are 1050 pixels in height, meaning that a coordinate value around 500 represents the centre of the screen.

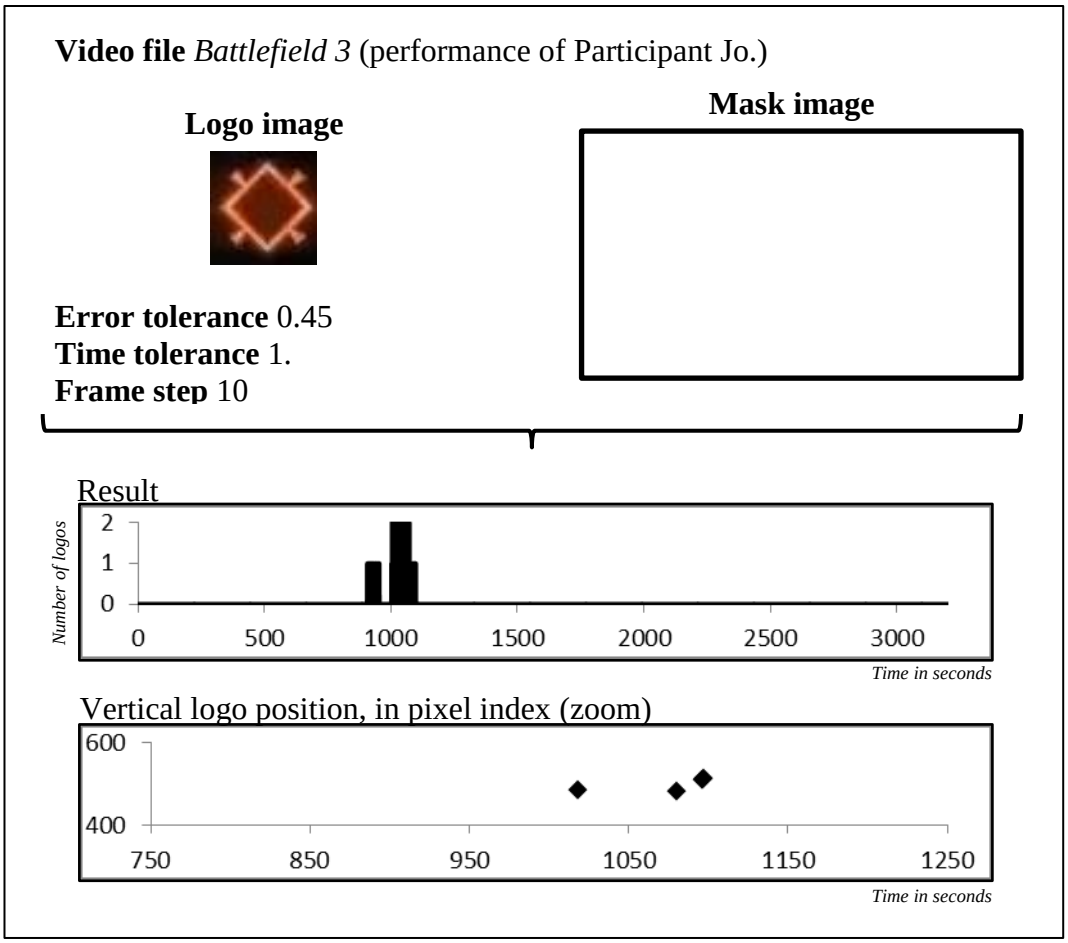


Figure 62. Enemy locations in *Battlefield 3*

Figure 62 displays the result of the Battlefield 3 example used throughout Section 5.4. The mask image is fully white, as the full screen is considered. The result graph displays the actual logos detection, with two distant enemies aiming to the player at around 1000 seconds. The bottom graph displays instances when the player is actually aiming to one of the enemies (a vertical position of 500 pixels in a frame of 1050 vertical pixels represent the centre of the screen)

Figure 63 illustrates a similar process with the game *Dead Island*. By detecting skull logos in the map (Figure 64 displays what the HUD looks like in *Dead Island* in order to understand the role of the mask image in Figure 63), it is possible to count the number of enemies surrounding the avatar during enemy waves (skull logos are only present during enemy waves). Moreover, the closest the icon to the centre of the map (representing the avatar position), the closest the enemy is to the player. The lower graph of Figure 63 shows the moments when at least one logo has been found close to the centre, indicating moments when the player is under attack.

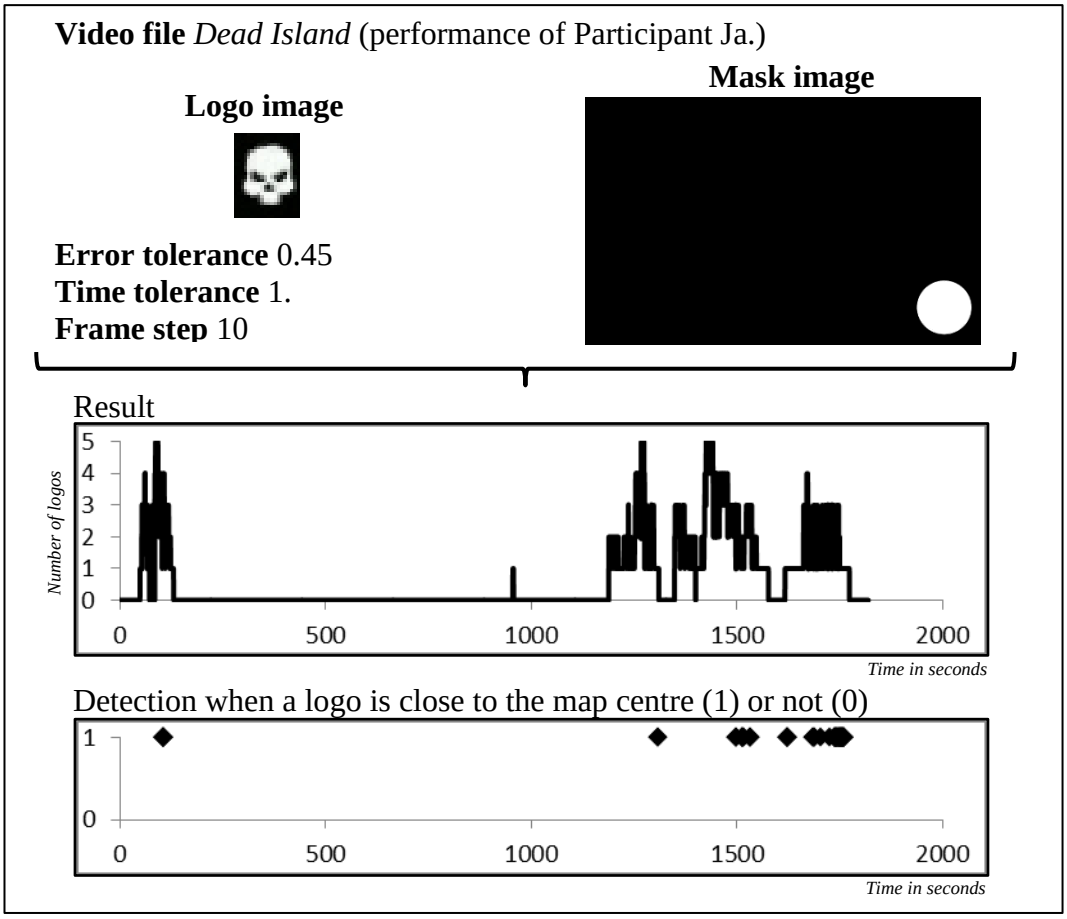


Figure 63. Detection of enemy logo in *Dead Island*

Figure 63 illustrates the moving logo detection with the game Dead Island. By detecting the presence of skull logos on the HUD map, it becomes possible to assess the number of enemies surrounding the player (up to five at around 1250 and 1400 seconds), and also to assess when the player is under attack, i.e. when a logo is close to the map centre.

Figure 64. HUD in *Dead Island*

Figure 64 displays a frame of the game Dead Island, in order to illustrate the presence of a map in the bottom-right corner of the screen for the mask used in Figure 63.

5.4.5 *Validity*

The accuracy of the moving information algorithm is much more sensitive to the logo morphological properties than the static information one. Indeed, a small logo, or a logo with a simple shape (like a square) is likely to trigger numerous false alarms. In *Battlefield 3* for instance, a “friend following” icon is represented by using a small square, and is then likely to be confounded with many other square shaped background elements. Using a mask is a good solution to limit the potential for false alarms. However, as illustrated in Figures 58 and 59, when the logo is big enough, or based on a complex shape (like a skull shape), the detection is highly accurate. Chapter 6, by presenting full gameplay performance segmentation results on several games, will show that the algorithm can still be used for a vast domain of applications.

Another limitation is the fact that the logo cannot change size. Algorithms like SURF (Bay et al., 2008) can address this limitation, but need to be studied further.

5.5 Bar progression assessment

The detection algorithms employed on graphical information presented thus far essentially produce discrete values, that is, being able to say whether a logo is detected or not, and how many logos are detected and where. The information presented to the researcher utilizing these methods is therefore limited to information that confirms an action or presents the player with information that can cue behaviours (e.g. looting). However, there are other forms of information that are constantly present on-screen (whilst in play) that provide continuous updates on player's standing in the game, for example, health, power or stamina bars. *Bar progression assessment* addresses these information sources that present continuous data that is the value can increase or decrease between minimum and maximum values. Such information allows the player to adapt their strategy appropriately in response to the current standing of their avatar. In the game *Dead Island,* for instance, low health reading during a fight may trigger a desire in the player to run away, however, if stamina levels that are required for running are low this may not be possible[23].

5.5.1 *Approach overview*

In order to execute an assessment of continuous values, the approach present in the current section, and also called colour ratio algorithm, is a method to summarize and visualize the constant movement of a bar progression throughout play. Such a result can then be analysed to pinpoint in which moments key

[23] Some example of use can also be seen in *Bioshock 2* (2K Games, 2010) (health level and power level) and *Max Payne 3* (Rockstar Games, 2012) (health level and bullet time power).

attributes related to avatar performance, such as health or stamina, change: dropping moments (loss of health, power use), raising moments (health kit found, ammunition found) or stable moments (no-fighting sequence, no-running); therefore segmenting the gameplay on the interaction level, based on avatar state.

Bar progression activity serves as a key signifier of the intensity of action as increases or decreases tend to occur in active moments, that stand in contrast to more passive moments during game play where bar progression does not fluctuate so drastically. Such information is nearly impossible to extract accurately when done manually, due to its continuous nature[24]. Automatically processing a bar representation is then crucial both for its meaning in terms of player experience, and also in terms of the impossibility to perform such a process manually.

The colour ratio algorithm functions to study each pixel in a given area (utilizing a mask image to indicate bar location and shape), comparing its pixel colour to the one constituting the fill colour of the bar. The ratio between matching pixels and the maximum number of pixels in the area is then calculated, with 1 representative of all the pixels matching one of the colours being searched, and 0 in the case of no match. Any intermediate value corresponds to the different levels of filling. For this algorithm, the colour domain representation (see Section 5.2) is the HSV one. As discussed earlier, the strength of the HSV system is that the Hue value matches the human perception of colour; meaning that two close Hue values are

[24] Except in some special cases when the bar can be discretely divided, like in *Dead Island* where the life bar is actually ten consecutive boxes.

more likely to look similar than two distant Hue values (see Figure 40). The rationale behind the use of the HSV colour domain is that it is able to compensate for levels of transparency when a HUD overlap the game world (a red colour, even transparent, will still have a red hue.

5.5.2 *Step by step description*

The required inputs to execute a colour ratio algorithm for a bar progression analysis are:

1. The *gameplay footage* in the form of a video file (sound is not mandatory)
2. A *mask image* indicating the bar position, or at least a meaningful portion of it.
3. A *reference colour image* containing the colour (or colours) utilised in the filling colours of the bar to be processed.
4. The *tolerance values* for Hue, Saturation and Value of the HSV colour domain; that controls how much a pixel value can divert from the colour of interest and still be considered a match.
5. A *frame step* that reduces the processing process by not requiring each individual frame to be processed, but instead every 2nd or 10th frame.

The first step of the algorithm is to process the reference colour image [input 3], representing the list of colours constituting the bar filling, and extracts from each pixel its HSV values [C 3-6]. Then, the number of white pixels in the mask [input 2] is calculated to represent the number of pixels indicative of a full bar [C 16; 34-36]. Finally, for each video frame, the area defined by the mask is considered. Each pixel included in the bar is first converted into the HSV colour domain, before comparing those to the reference colours [C 20-32]. If the differences are

lower than the tolerance values stipulated [input 4], then the pixel is considered as matching[25] (the pixel needs to match at least one reference colour to be accepted). The filling of the bar is calculating using $\frac{nb\ of\ matching\ pixels}{nb\ of\ white\ pixels\ in\ the\ mask}$ [C 40-41]. A result of 1 is return for a full bar representation, 0 for an empty bar, with intermediate results between 0 and 1 indicating how full or empty the bar is at any given point. The result return by the algorithm is the list all the computed ratios, with the matching frame timestamp.

Figure 65 is illustrating the major steps using diagrams schematizing two frames containing a health bar. Two frame examples are given: one with a full bar, and one with a half-filled bar. (a) represents the first stage, before the frame are masked using the provided mask image [input 2]. (b) represents the results of the masking stage, discarding the pixels that are not constituting the bar. The bar is now ready to be analysed using the reference colours [input 3], and the tolerance values [input 4]. (c) represents the actual comparison between each pixel from the bar area and each of the reference colours.

If all the differences between the H, S and V values of the current bar pixel and the reference colours are below the H, S and V tolerance values, the pixel is considered as part of the filling. Figure 65 (c) illustrates that it is sufficient for the

[25] For the comparison step, the cyclic nature of the Hue (see Section 5.2.1) must be taken into account. Indeed, if the Hue is represented using values between 0 and 179 (which is the case for instance in OpenCV (Intel Corporation, Willow Garage & Itseez, 1999)); 179 and 0 are both red pixels; meaning that a reference pixel with 178, a frame pixel with 5 and a tolerance of 10 must consider the pixel as detected.

current bar pixels to match one of the reference colours to have a match. On the left example, the first two pixels match the “blue” reference, and the three last match the “green” reference, making the bar a full bar (100%). However, on the right example, only the two first pixels are matching one of the two reference colours (the “blue” one), the three last ones having their values too far from any of the reference colours. Only two pixels are matching on a total of five (40%).

Figure 65 illustrates, using diagrams, the different steps of the bar progression algorithm. (a) represents two frames, one with a bar fully filled, the second with a bar nearly empty, before they are masked. (b) represents the masking process, focusing on the area of interest, and discarding the rest of the frame. (c) represents the comparison process between each reference colour, and the bar pixels. When all the differences between the H, S and V component of the pixels, and one reference colour are below the three matching tolerance values, then the pixel is considered as part of the filling. (c) shows that it is necessary to have a match with only one of the reference colour to have the pixel accepted. The left example displays a detection of 100%, and the right one displays a detection of 40%.

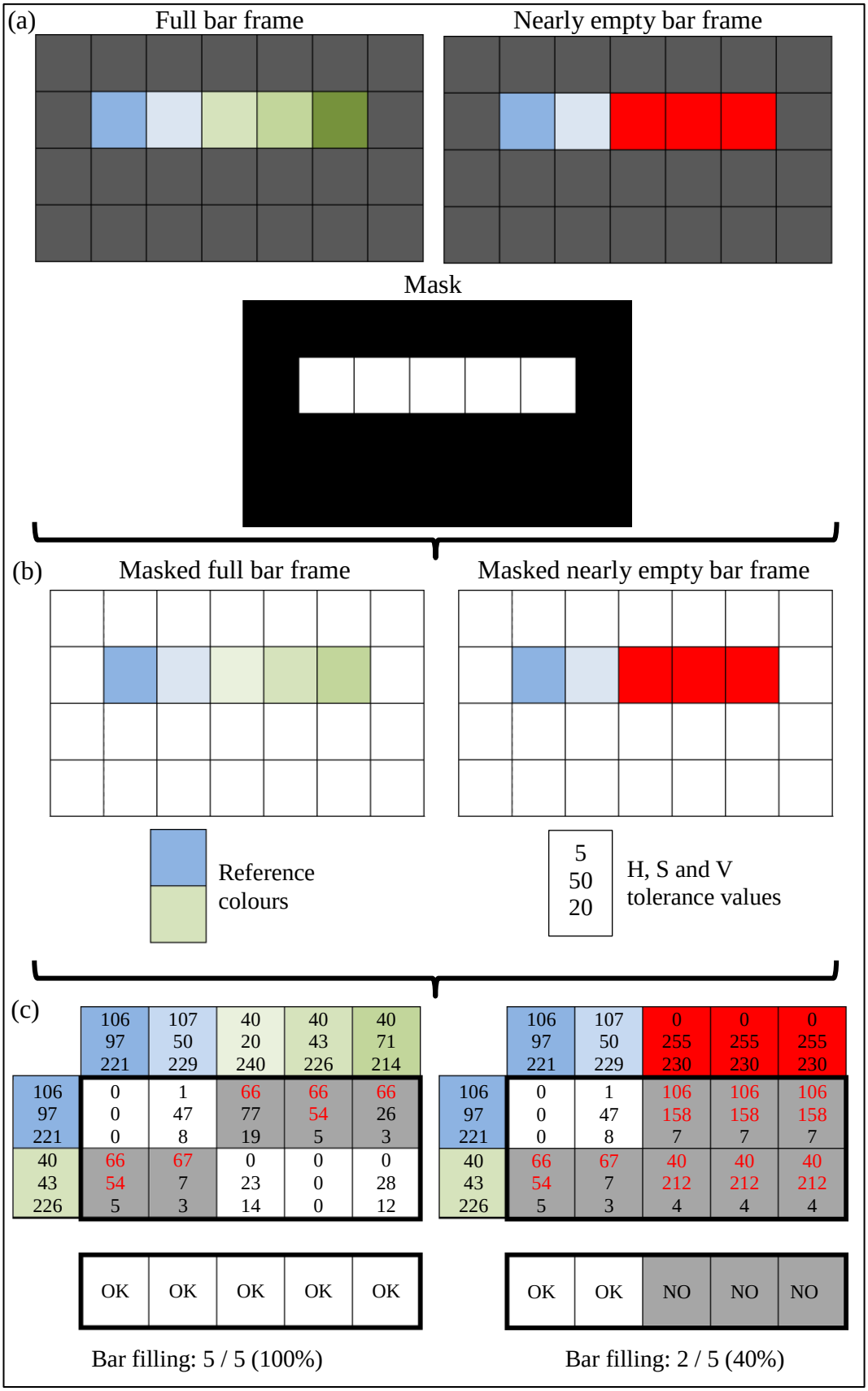


Figure 65. Bar progression algorithm step by step

Figure 65 is described on the previous page

Figure 66 illustrates the application of the colour ratio algorithm to an actual gameplay frame, taken from the game *Dead Island*. First, the health bar is extracted by masking the current frame. Then each bar pixel is compared with each reference colour, using the tolerance values. When the ratio between the matched pixels and the total number of bar pixels in computed, the frame reveals that 65% of the bar is filled.

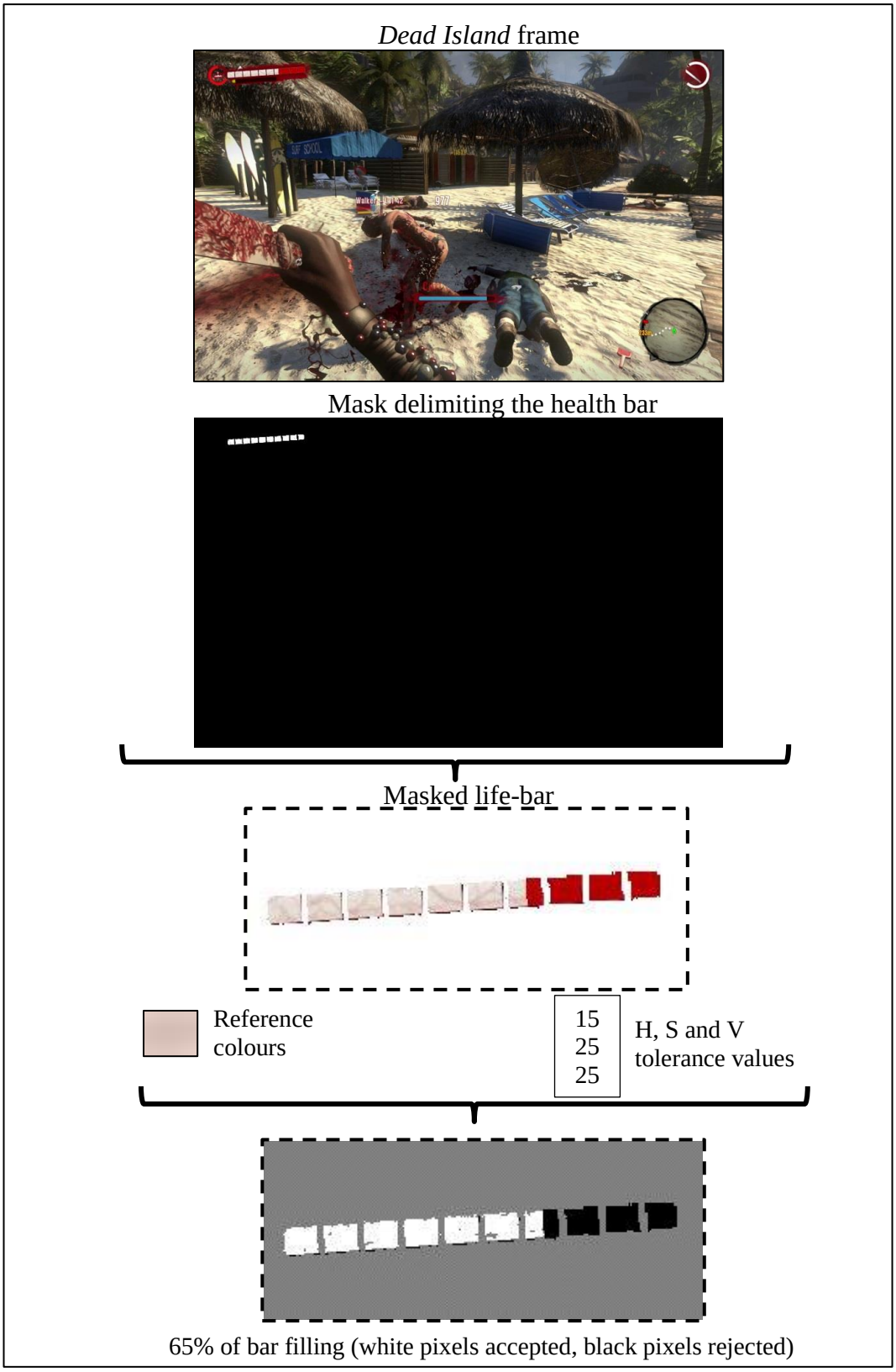


Figure 66. Life bar progression assessment in *Dead Island*

Figure 66 illustrates the different steps of assessing the life bar progression in the game Dead Island. First, the mask image [input 2] is used to focus on the area of interest, and discard the rest of the frame. Then, each bar pixel is compared with each reference colour, using the tolerance values. Finally, the bottom image shows the accepted pixels (in white), and the rejected ones (in black), giving a filling of 65%.

5.5.3 *Colour Ratio (C) Algorithm*

Colour Ratio (C)

Inputs

gameFootage: video /* *gameplay footage video* */
barLocationMask: binary image /* *area of the screen where the bar is located* */
colourReferenceImage: image /* *bar-filling colour* */
hueTolerance, satTolerance, valTolerance: integer /* *tolerance values from which each hsv values can divert and still be accepted* */
frameStep: integer /* *video frame increment* */

Output

result: vector of {timestamp: float, value: float} /* *bar filling ratio per timestamp* */

Variables

n: integer /* *number of frames in the game footage* */
i: integer /* *loop variable* */
x, y: integer /* *pixel coordinates* */
colourToFind: vector of HSV colors /* *vector of all the colors constituting the bar-filling* */
currentFrame: image /* *frame currently processed* */
pixelMatched: bool /* *true if the currently processed pixel matches one of the bar-filling colors* */
hueDistance, satDistance, valDistance: integer /* *current hsv distances between the currently process pixel hsv color, and one of the colors constituting the bar-filling* */
currentPixel: hsv pixel /* *pixel currently processed* */
nbMatchingPixels: integer /* *number of pixels matching one of the bar-filling colors* */
nbWhitePixelsInMask: integer /* *maximum number of pixels in the bar area* */

Algorithm

```
1   n ← nbFrames(gameFootage)
2
3   /* Grabs all the colors from the colorReferenceImage, in hsv color domain */
4   colourReferenceImage ← hsvConversionOf(colourReferenceImage)
5   for each pixels (x, y) in colourReferenceImage
6     colourToFind.push(colorReferenceImage(x, y))
7
8   /* Sequentially grabs the frames of interest from the gameplay footage */
9   for each position i between 0 and (n − 1), using a step of frameStep
10    currentFrame ← gameFootage(i)
11    currentFrame ← hsvConversionOf(currentFrame)
12    nbMatchingPixels ← 0
13    nbWhitePixelsInMask ← 0
14
15    /* Considers all the pixel locations that constitute the bar area in barLocationMask */
16    for each pixels (x, y) = 1 in barLocationMask
17      currentPixel ← currentFrame(x, y)
18      pixelMatched ← false
19
20      /* Compares the current pixel color with all the colors constituting the bar-filling */
21      for each index j between 0 and size(colourToFind)
22        if (pixelMatched = false) /* if true: pixel already accepted as matching */
```

23 $hueDistance \leftarrow |currentPixel.hue - colourToFind(j).hue|$
24 $satDistance \leftarrow |currentPixel.sat - colourToFind(j).sat|$
25 $valDistance \leftarrow |currentPixel.val - colourToFind(j).val|$
26
27 /* Hue is cyclic, so the distance should be a modulo one (MAX_HUE being
28 the maximal hue value possible, 180 for instance in OpenCV) */
29 $hueDistance \leftarrow min(hueDistance, MAX\_HUE - hueDistance)$
30
31 /* The current pixel color is considered as part of the bar-filling
when the different distances are inferior with
the corresponding tolerances */
32 $pixelMatched \leftarrow (hueDistance \leq hueTolerance)$
$AND\ (satDistance \leq satTolerance)$
$AND\ (valDistance \leq valTolerance)$
33
34 /* Counts the maximum number of pixels constituting the bar */
35 $nbWhitePixelsInMask \leftarrow nbWhitePixelsInMask + 1$
36 $\boldsymbol{if}\ (pixelMatched = true)$
37 /* Counts the number of actual color matching pixels */
38 $nbMatchingPixels \leftarrow nbMatchingPixels + 1$
39
40 /* Adds the current frame timestamp, in second, and the ratio value,
representing the filling of the bar */
41 $result.push\left(\frac{i}{getFrameRate(gameFootage)}, \frac{nbMatchingPixels}{nbWhitePixelsInMask}\right)$

5.5.4 *Result*

Figure 67 extends the example provided in Section 5.5.1, to include the variation and changes in the health bar over a period of 30 minutes. As discussed, there are a number of moments in which health drops, rises dramatically amongst more stable moments. It is interesting to note that the output from the colour ratio algorithm can be combined with the findings shown for static logo detection (illustrated in Figure 55), that displays the presence or absence of the health kit logo (indicating its use), therefore confirming whether health restoration is the result of a player acquiring a health kit.

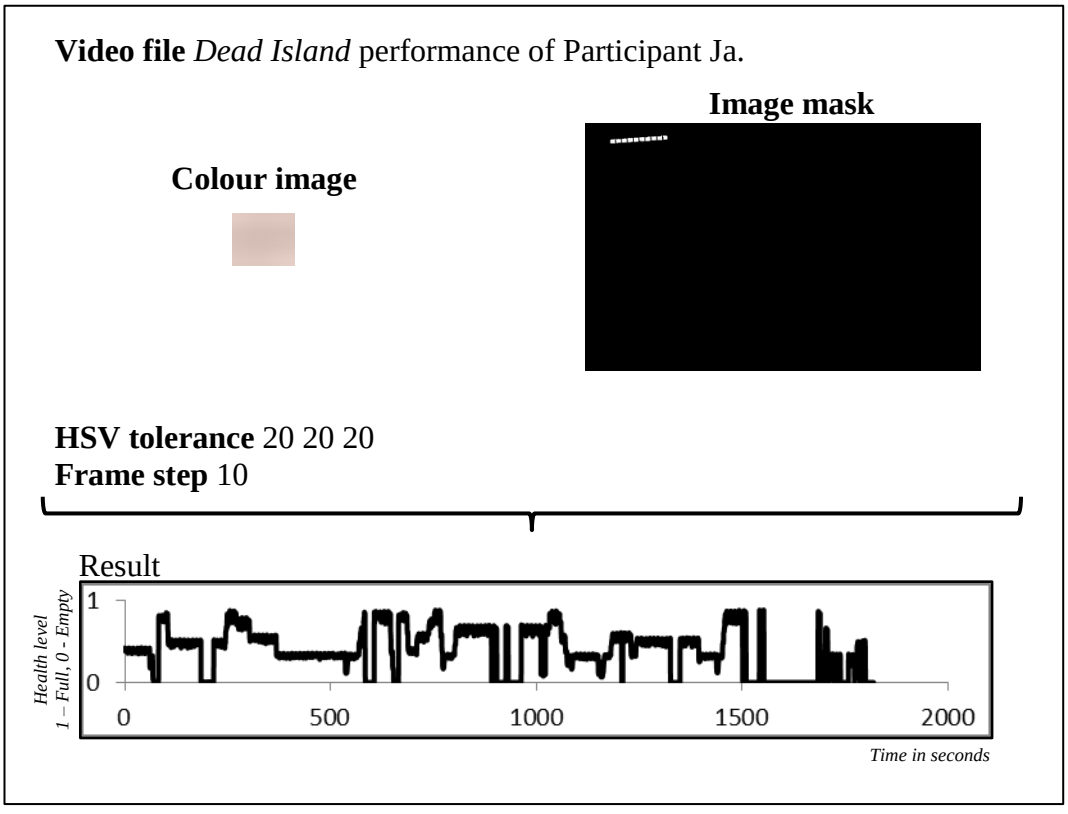


Figure 67. Health processed in *Dead Island* through life bar analysis

Figure 67 illustrates the example provided in Section 5.5. The health evolution can be studied through its dropping (fighting), stationary (no fight or mastered fights) and rising moments (health regenerating). The rising moments can be combined with the findings shown for static logo detection (illustrated in Figure 55), showing that the rising moments indeed match the use of first aid kits.

Figure 68 presents another possibility to use the bar progression method. In *Battlefield 3*, the HUD is always slightly in movement, making it undetectable using the static information detection algorithm, and also changes over time in terms of content display, making it undetectable using the moving information detection algorithm either. However, the bottom right panel of the HUD is always displayed in bluish colours, as displayed by Figure 69. It is then possible to study the colour of the HUD area in terms of blue. If the area is mainly blue, then the HUD is on screen.

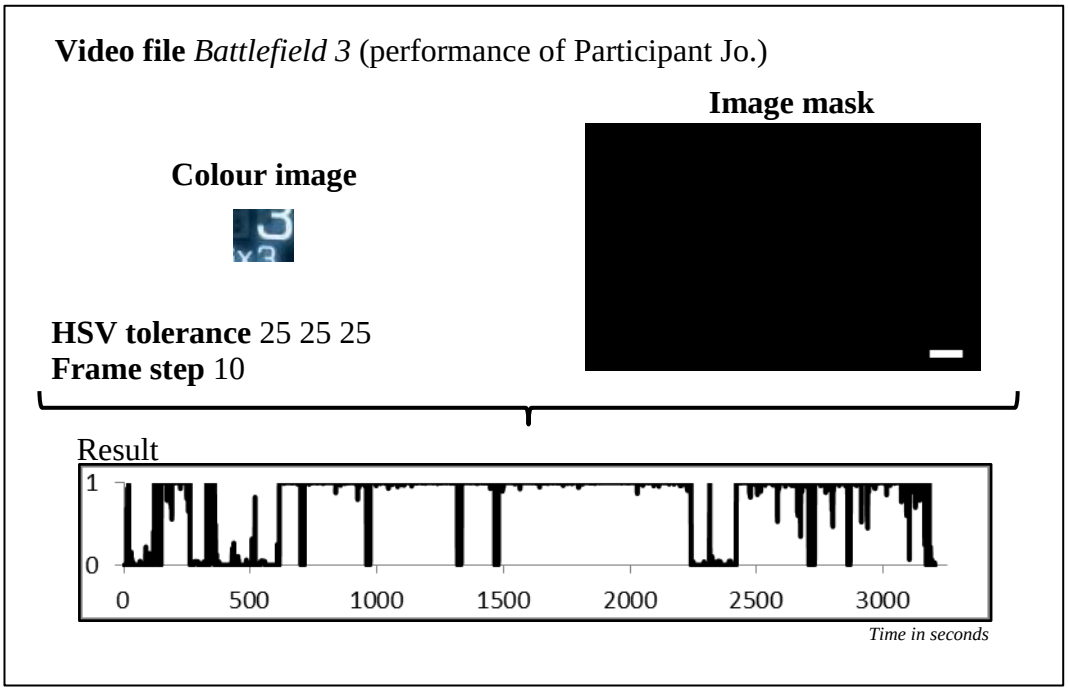


Figure 68. HUD detection in *Battlefield 3*

Figure 68 illustrates another use of the colour ratio algorithm. In Battlefield 3, the HUD is constantly slightly moving, and its contents change regarding the weapon carried by the avatar. None of the logo detection method can then work properly. However, the HUD is always in the bottom right corner of the screen, and using bluish colours. The HUD can be detected using the colour ratio algorithm. The result, noisy, can be improved using averaging and thresholding, as illustrated in Figure 69. The cut-scenes are then easy to detect, as they are represented by long segments without HUD: around 100 seconds, 500 seconds, and 2300 seconds.

Figure 69. HUD in *Battlefield 3*

Figure 69 is a frame from the game Battlefield 3, displaying the HUD bluish design.

As illustrated in Figure 68 though, the result can be noisy (explainable by some blue frames also present in cut-scene or by a short missed-detection). Figure 70 shows how noisy results based on the bar progression algorithm can eventually be improved, by averaging the curve (each value being the mean of the 5 previous ones, and the 5 following ones), and then thresholding it.

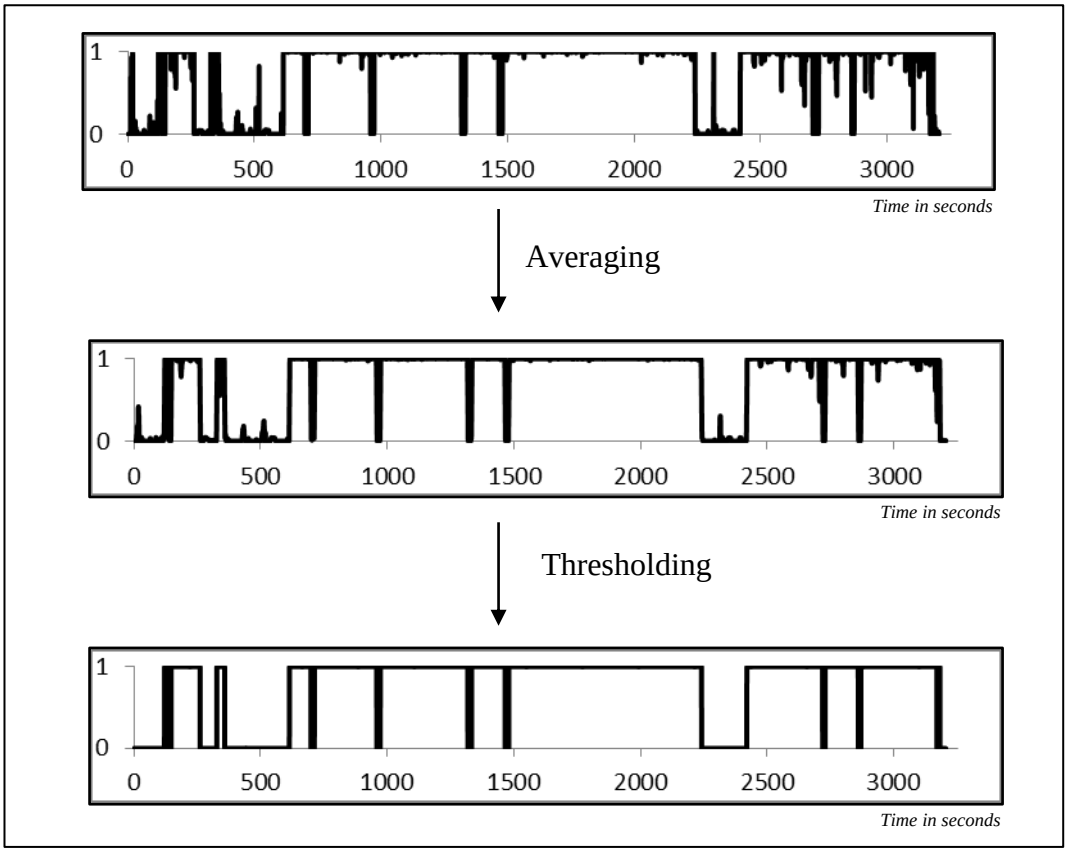


Figure 70. Noisy result improvement through averaging and thresholding

Figure 70 illustrates the use of averaging and thresholding to improve a noisy colour ratio result. In the example of Battlefield 3, the HUD is constantly moving, and several cut-scenes also contain blue pixels inside the area of interest, thus creating short drops or short peaks. Averaging (meaning that each value becomes the mean of the five previous and five next ones) reduces the importance of isolated peaks and drops. Then, thresholding (meaning that all values over the threshold value become 1, and all values below become 0) then remove the isolated peaks, and fill the isolated drops.

The *bar progression* approach is a very versatile one. It can then be used for a large range of application. For instance, in *Bioshock 2*, it is also used to detect letter-boxing effects during the cut-scene (by assessing the presence of black area on the top and bottom of the frame).

5.5.5 *Validity*

If, for the information detection algorithms, the level of transparency has no, or few, impacts on the accuracy of the result, because the colour information can be removed and replaced by an edge representation; the colour ratio algorithm is unable to completely discard the colour information, making the level of transparency a variable impacting on the accuracy of the result. Indeed, if the results for the games *Dead Island* and *Bioshock 2* are accurate and give a good representation of the bar evolution over time, *Max Payne 3* does not profit from the same amount of precision, due to the choice of the graphical design making the HUD highly influenced by the background colour. For example, with a red background, it is impossible to process the health of *Max Payne*. That being said, the information is still usable on two levels. First, most of the time, the information is still accurate enough to be interpreted in conjunction with other values, and can be improved using the averaging and thresholding operation illustrated in Figure 70. Secondly, it also informs on the way information is broadcasted to the player. In *Dead Island* or *Bioshock 2*, the player can, at any time, have a quick and exact update about his avatar state, with a HUD covering part of the game world representation; while in *Max Payne 3*, the choice has been made to favour the game world representation, at the expense of the possibility to always gain accurate and immediate avatar status information. The player will encounter moments when his desire to have an accurate avatar state information will not be fulfilled.

Finally, in order to have accurate results, bar representation assessments can be coupled with other data, like for instance the detection of the presence or absence

of the HUD on screen (Section 5.3 and Section 5.4). Indeed, some of the 0 values can be misinterpreted as an empty bar, while it could also be the total absence of the bar on-screen. Coupling the results with logo detection considerably limits these potential misinterpretations. This is a powerful use of the hierarchy structure of the layers introduced in Chapter 4, the health bar being a sub-layer (interaction) of the HUD (degree of freedom).

5.6 Visual breaks detection

In automatic video processing and indexing, a major area of interest is the automatic detection of shot or scene boundaries from within large video files. Indeed, as on a DVD where a movie is cut into sub-chapters in order to improve navigation and help the watcher find the sequence of interest (with a well-chosen title), a video file can be subdivided to improve its content description and localization. Numerous research works are conducted in order to identify relevant parameters indicative of a clear break in a video (Shukla & Sharma, 2011). These methods define a scene, or shot, based on the idea of continuity. For example, two consecutive frames of the same scene are likely to share commonalities, while two highly different consecutive frames are likely to be indicative of a break.

The idea of continuity can be applied to gameplay performance segmentation, by considering whole, or part, of the screen and then studying the amount of changes between two consecutive frames. An example of that, which will be developed in the following sections, is the detection of page change in a game menu. In *Bioshock 2*, the player can enter inside a help menu any time they need more information about the game world. The help menu space can be detected via the static information detection presented in Section 5.3. Then, to detect also the moment when the player decides to change page (to go to another topic), it is possible to evaluate how much the text area evolves. This is indicative of, regarding the chosen break detection method, a topic change, or a scrolling action in the same topic (page scroll up-down for instance).

5.6.1 *Approach overview*

As discussed above, the *visual break detection* approach is separated into two different methods of detecting breaks, one being less permissive than the other. The idea is to select an area of the video, and compare the amount of change between a frame and the previous one. In the example provided in the current section, “frame t-1” represents the frame occurring just before “frame t”.

The first method, *direct distance comparison*, is the less permissive one, and compares each pixel from the previous frame with the matching pixel (same position) in the current frame. The greater the difference between the pixels, the more likely a break is detected. That means that even the smallest change (like a small camera movement or a text scroll) will generate a break.

The second method, more permissive, and actually used for movie automatic segmentation because of its robustness to camera movement, is the *colour histogram comparison* method. The pixels in the considered area are summarized into colour histograms, representing the colour distribution of the area (i.e., how many pixels are mainly red, blue, green, white, etc.). Then, the histograms are compared instead of comparing the area actual content. If two consecutive frames look the same in terms of colour, a break will not be detected. If the colour distribution changes drastically, then a break is detected. That means that a small change, like a text scroll for instance, will not be detected, but a consequent change, like going from a text area to a topic selection menu, will generate a break.

5.6.2 *Step by step description*

The required inputs for the visual break detection processing are (for both the direction comparison and histogram methods)

1. The *gameplay footage* in the form of a video file (sound is not mandatory)
2. An *image mask* indicating the screen area in which a break needs to be detected.
3. The *method* to use: direct comparison or histogram comparison
4. A frame step that reduces the processing by not requiring each individual frame to be processed, but instead every 2nd or 10th frame.

The first step of the algorithm is to convert the colour frame into a greyscale one [F 5-6]. The rationale behind such a conversion is to simplify the analysis and classification of pixels (notably for the histogram method), as greyscale pixels only have one value (in general between 0 (black) and 255 (white)), as opposed to colour pixels, which have three values to consider. The proposed algorithm works on any kind of video, including greyscale ones. Changing from colour to greyscale does not have a strong impact on the result, and simplifies the comparison step while improving the clarity of the description. It is, however, possible to think of a colour-based version of it by separating the three colour channels. Then, except for the first frame that does not have any predecessor [F 8], the algorithm compares the area covered by the mask image [input 2] (see the masking mechanism in Figure 71) in both the current frame and the previous one, and gives a score based on the degree of differences. The higher the score, the bigger the differences are, and the more likely a scene break is present.

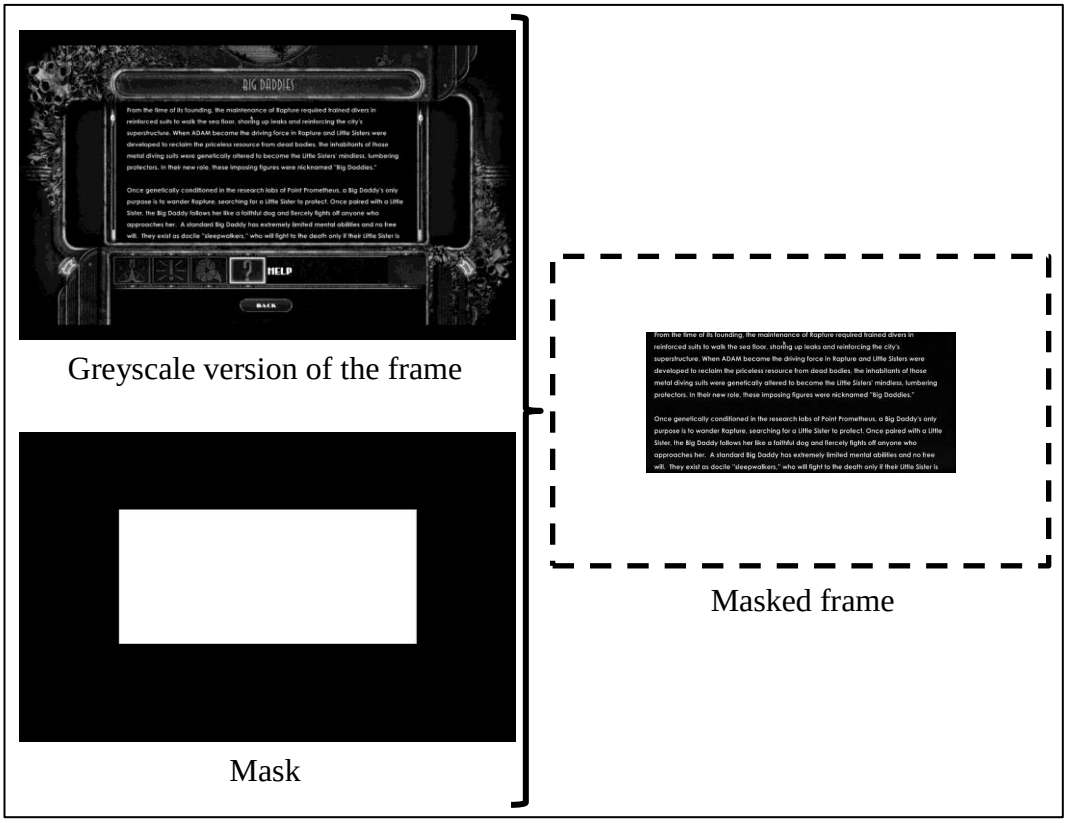


Figure 71. Masking mechanism

Figure 71 illustrates the masking process, in order to focus only on the area of interest. In the help menu of Bioshock 2, the area of interest is the text box.

Two methods can be chosen by the analyst [input 3]. The more straightforward one, but also the less permissive, is utilizing a *difference comparison* between each pixel in the current frame (t) and the one at the same position in the previous one (t-1), using a regular distance formula. The score for the current frame is then the sum of all the distances between all the considered pixels [F 9-16].

$$distance = \sum_{x,y} |frame_t(x,y) - frame_{t-1}(x,y)|$$

The second method is more permissive (allowing two frames to be slightly different without detecting a break), and is actually used more widely for video segmentation as opposed to the method described above (this latter detecting too many breaks to be useful for movie segmentation, as even the slightest camera movement will generate a break detection). The idea is to abstract an image into a colour distribution histogram, which considers all the pixels enclosed in the area defined by the mask image, and counts how many of them are the same colour (i.e. the same value between 0 and 255 for greyscale images). The result is a histogram representing the colour scheme of a frame. Figure 72 illustrates the conversion of the masked frame into its colour distribution histogram. The frame is mainly black so the highest values in the histogram are around 0.

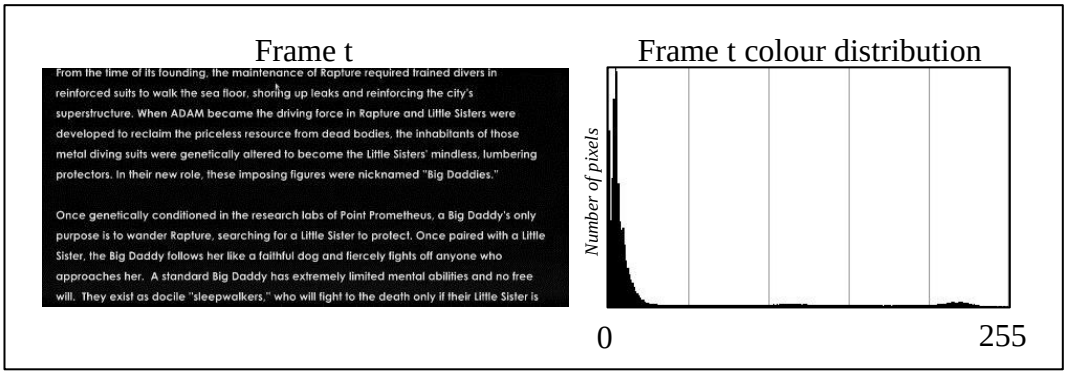


Figure 72. Example of colour distribution histogram

Figure 72 illustrates the use of colour distribution histogram. Because the frame mainly black, the histogram has its highest values around 0.

Instead of comparing the pixels directly from the frame, the comparison is executed between the *colour-distribution histograms* (current and previous frame histograms), using the Chi Squared test (Huang & Liao, 2001, p. 713) [F 15-23]

$$chiSquared = \sum_{j=0}^{nbColors-1} \frac{\big(currentHist(j) - previousHist(j)\big)^2}{\max\big(currentHist(j), previousHist(j)\big)}$$

With this method, only a major change in the colour distribution would trigger break detection.

Figure 73 illustrates the differences between the two methods by using simplified frames. Example A in Figure 73 shows for instance that, by shifting a line downward, 8 pixels change values (and are then included into the differences sum), triggering a break detection using the *difference* method. However, the number of 1 pixels and 0 pixels are exactly the same. The chi squared test over the *colour distribution histograms* will not trigger break detection. The *colour distribution* method accounts for movements as part of the same scene. However, in Example B, not only pixels are moving, but the colour distribution is changing too. Then, both methods detect a break.

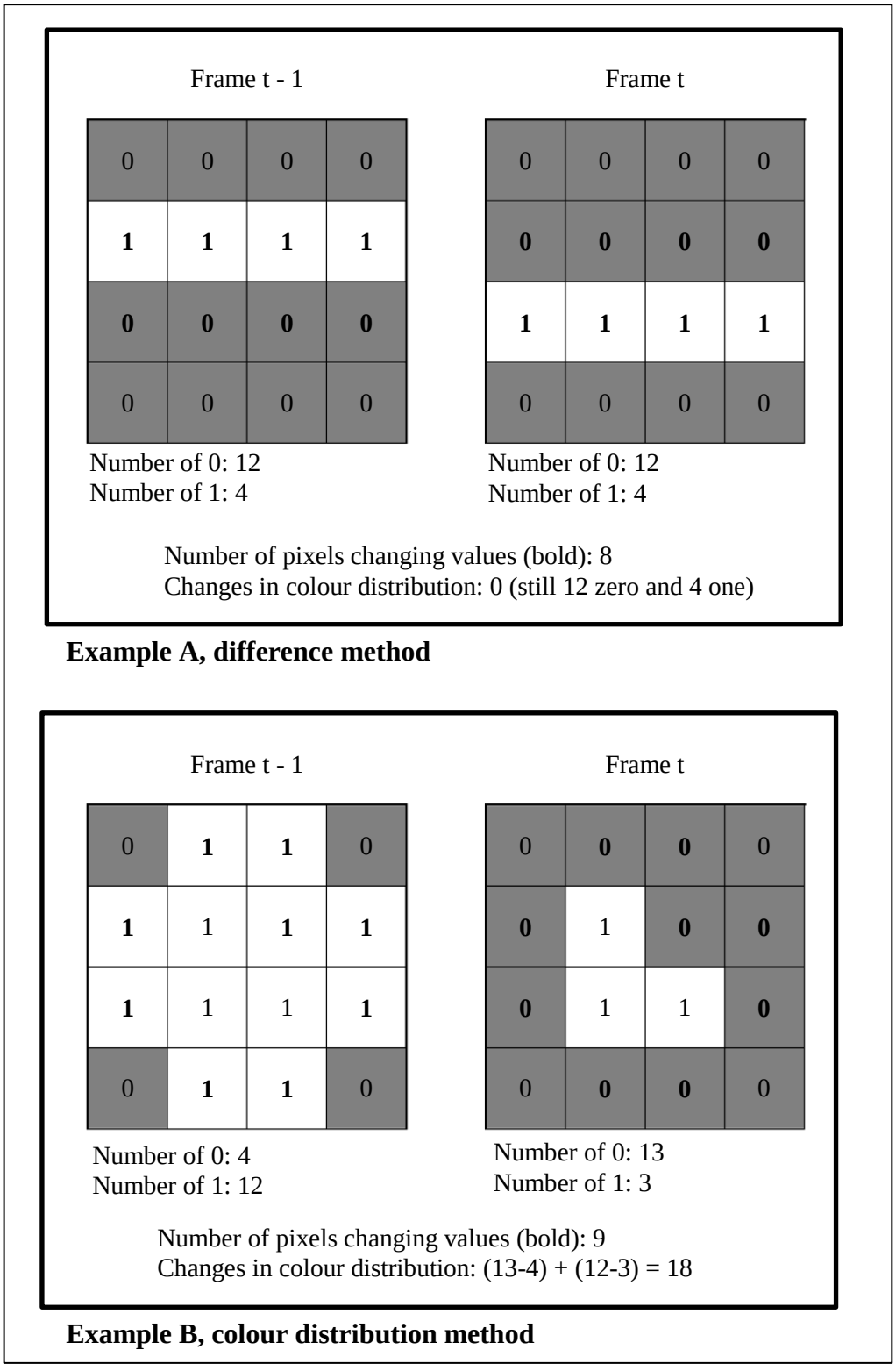


Figure 73. Difference in break detection methods

Figure 73 displays several diagrams to explain the two methods discussed in Section 5.6. The top example (A) displays two consecutive frames, representing a white line being shifted downward. The number of black and white pixels remains stable, meaning that the colour distribution is unchanged. The histogram method will not detect any change, while the difference method will acknowledge the fact that several pixels change their values (in bold). (B) displays two consecutive frames, with a strong change in the colour distribution: 9 black pixels more, 9 white pixels less. This time, both methods will detect a change.

Figure 74 and Figure 75 illustrate the two methods using a sequence from *Bioshock 2*. In the help menu, the player can read information, change the main topic, or scroll if the text is long. In Figure 74, the player is scrolling. The *difference* method does not detect any break while the player is reading, but will trigger a break as soon as the player decides to scroll. However, because the text is roughly the same (just a couple of lines are different), the histograms in Figure 74 look similar, and the *colour distribution* method will not trigger a break. However, in Figure 75, the player is going from the topic selection panel, containing a list of buttons, to the topic screen, containing text. Not only will the difference method detect a break, but the histograms are also different (see red circle in Figure 75), and will also trigger a break. Then, it becomes possible, while the player is inside the help menu, to know when they decide to change topic (break detected by both methods), or when they are interacting with the text, to scroll down or up (break detected by only the difference method).

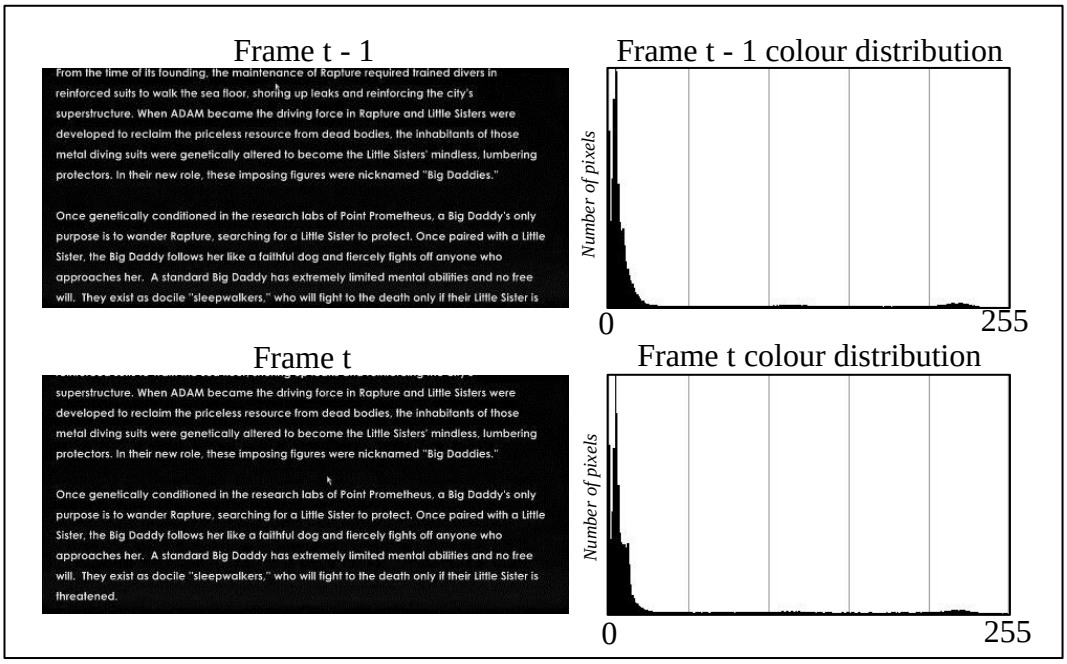


Figure 74. Text scroll and influence over the histogram

Figure 74 illustrates the weak influence of text scroll over the colour distribution histogram. Indeed, the colour scheme remains similar (mainly black in frame t-1 and frame t). The histogram method will not detect any break.

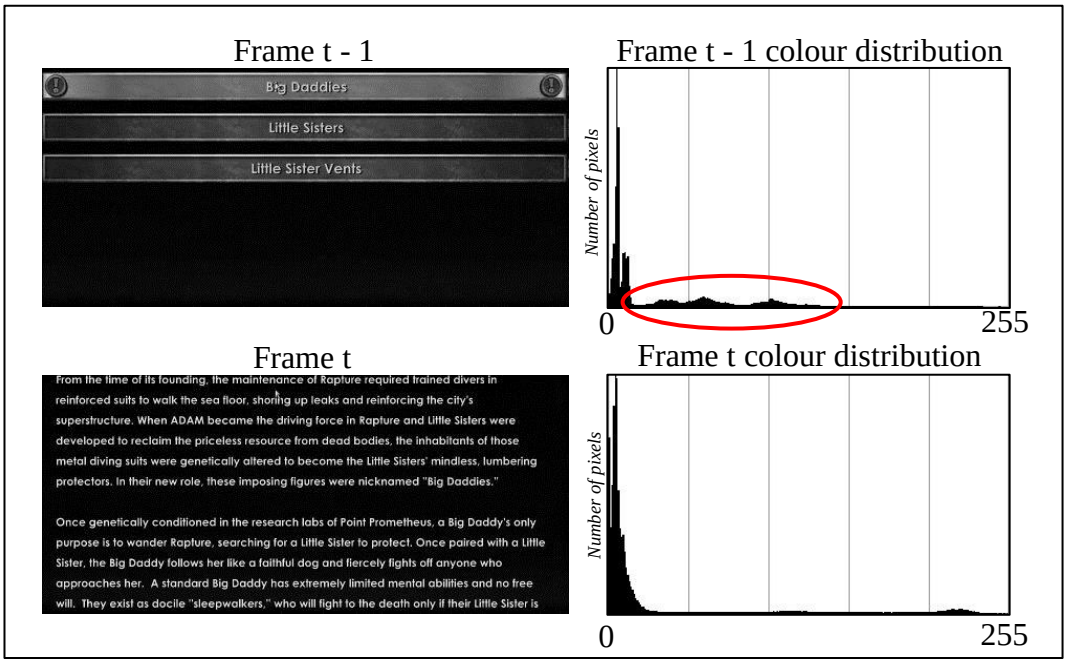


Figure 75. Menu change and influence over the histogram

Figure 75 illustrates the strong influence of menu page change over the colour distribution histogram. Because the colour scheme drastically changes (from greyish in frame t-1 to mainly black in frame t), the histogram also changes (cf. red circle). The histogram method will then detect a break.

At the end of the algorithm, the results for both algorithms are then normalized (divided by the highest score, so that the results range between 0 and 1), and returned to the analyst with the matching time-stamps. [F 25-26]

5.6.3 *Frame comparison (F) algorithm*

Frame comparison (F)

Inputs

```
gameFootage: video  /* gameplay footage video */
maskImage: binary image /* area in which the scene change is considered */
/* desired scene change method */
method: enum {DIRECT_METHOD, HISTOGRAM_METHOD}
frameStep: integer   /* video frame increment */
```

Output

```
/* amount of detected changes per timestamp */
result: vector of {timestamp: float, value: float}
```

Variables

```
n: integer /* number of frames in the game footage */
difference, chiSquared: float /* differences between two consecutive frames,
                                 regarding the chosen method */
i, j: integer /* loop variables */
x, y: integer /* pixel coordinates */
currentFrame: image /* frame currently processed */
previousFrame: image /* previous frame, to assess the changes with the current one */
currentHist: histogram (256 bins) /* color distribution of the current frame */
previousHist: histogram (256 bins) /* color distribution of the previous frame */
```

Algorithm

```
1   n ← nbFrames(gameFootage)
2
3   /* Sequentially grabs the frames of interest from the gameplay footage */
4   for each position i between 0 and (n − 1), using a step of frameStep
5     currentFrame ← gameFootage(i)
6     currentFrame ← greyscaleConversionOf(currentFrame)
7
8     if (i > 0)  /* if i = 0, no previous frame exists to do the comparison with */
9       if (method = DIRECT_METHOD) /* Pixel to pixel comparison method */
10        difference ← 0
11        for each pixels (x, y) = 1 in maskImage
12          /* Sum of all the distances between the same position pixels */
13          diff ← diff + |currentFrame(x, y) − previousFrame(x, y)|
14
15        /* Adds the current frame timestamp, in second, and the difference value,
               to the result vector */
16        result.push( i / getFrameRate(gameFootage) , difference )
17
18      else if (method = HISTOGRAM_METHOD) /* Color distribution method */
19        currentHist ← getColorDistribution(currentFrame, maskImage)
```

```
20          previousHist ← getColorDistribution(previousFrame, maskImage)
21          m ← nbBins(currentHistogram)
22          /* ChiSquared test */
23          chiSquared = ∑_{j=0}^{m−1} (currentHist(j) − previousHist(j))^2 / max(currentHist(j), previousHist(j))
24
25          /* Adds the current frame timestamp, in second, and the ChiSquared test value,
                to the result vector */
26          result.push( i / getFrameRate(gameFootage), chiSquared )
27
28      /* The currentFrame is now the previous one for the next processing */
29      previousFrame ← currentFrame
30
31  /* Put all the result values between 0 and 1 */
32  normalize(result)
```

5.6.4 *Result*

Figure 76 illustrates the result when applying scene change detection to a full performance. The first remark is about the impossibility to use the raw result, as during a play, two consecutive frames are always likely to have strong changes between them (a). However, Figure 76 displays a good example of the usefulness of the hierarchy in the five layers proposed in Chapter 4. Indeed, what is interesting is the amount of change between frames, but only while the player is inside the help menu space. The detection of the help menu segment (spatial temporal) can then help to focus on useful sub-layer elements (degree of freedom or interaction), which can be analysed. This is the case for instance of the sequence between 1672 and 1941 seconds (b). The result becomes interpretable. Each peak in the histogram result indicates a change in the menu page, while the difference result peaks indicate when the player is scrolling (c).

5.6.5 *Validity*

As illustrated above, the results of *scene change* algorithms are difficult to analyse when non-contextualized by any other top-layer metric, meaning that the algorithms should only be used over a segment that is known to have static visual sequences. The algorithms work then efficiently and highlight breaks with strong accuracy.

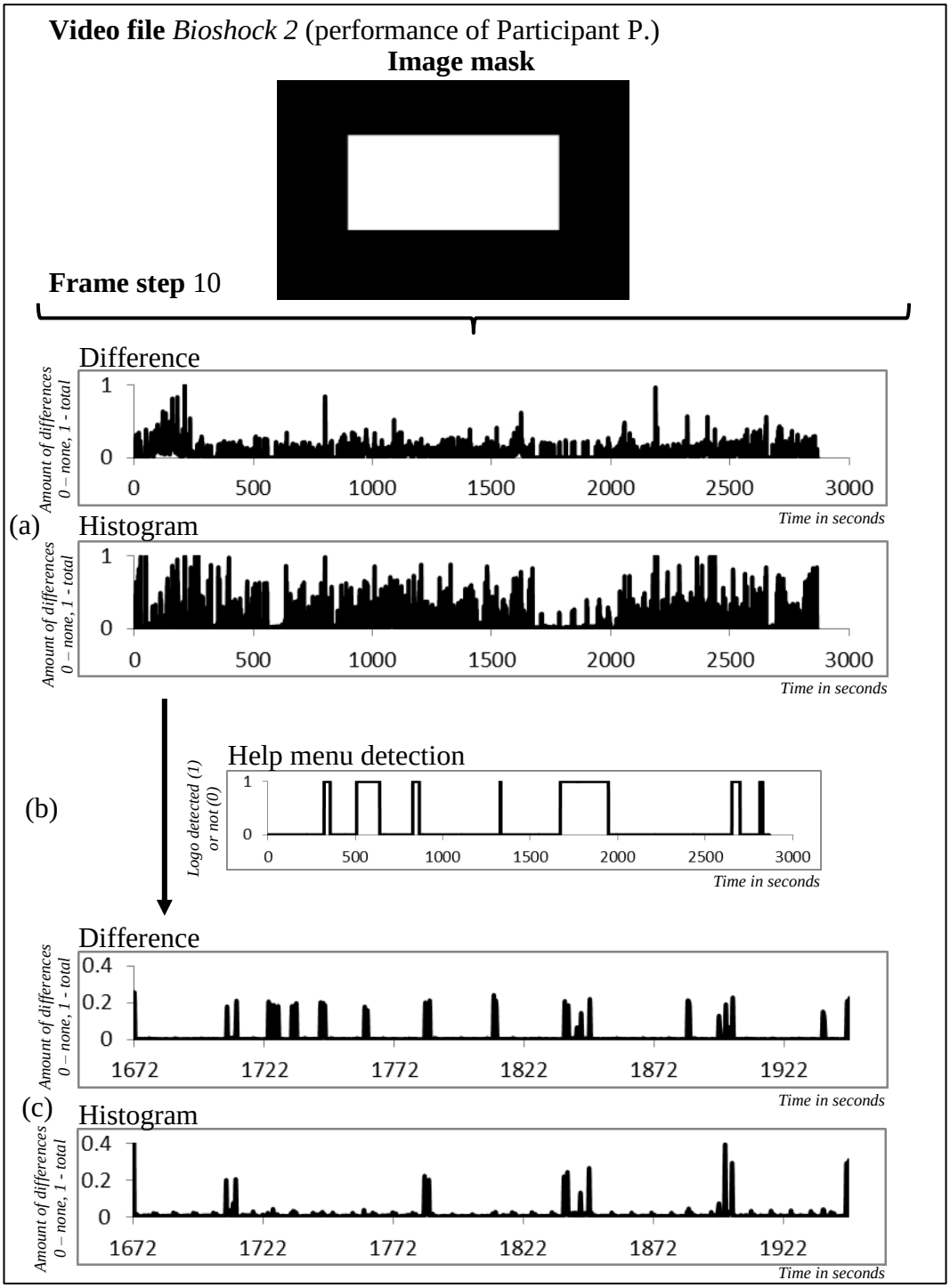


Figure 76. Detection of menu change and text scroll in *Bioshock 2*

Figure 76 displays the result of the example used throughout Section 5.6, and also illustrates the usefulness of hierarchically considering the layers presented in Chapter 4. (a) displays the raw result, impossible to analyse due to the noise (calculating the amount of differences between two consecutive frame generates a lot of noise because of the constant movement on the screen during the play). However, (b) illustrates that, by using the help menu detection, it is possible to focus on segments during which the study of frame differences make sense, for instance between 1672 and 1941 seconds. Then, (c) represents the metrics that can be understood. The difference method produces a peak each time the screen contains changes, therefore indicating when the player is interacting (by scrolling a text for instance). The histogram method produces a peak only when the colour scheme of a frame suddenly changes, therefore indicating when the player changes page in the menu.

5.7 Sound detection

The four previous algorithms are video-based. They process information enclosed in the video stream by analysing each frame, in order to automatize part of the gameplay performance segmentation process. The following algorithm, based on *sound correlation* (Roads, 1996, p. 509; Yarlagadda, 2010, Chapter 2), focuses on audio information enclosed in the audio-stream of the gameplay performance footage, thus illustrating the value of sound and how it can be processed. Indeed, the sonic atmosphere of videogames also carries a considerable amount of information. Furthermore, most of the graphical elements are also paired with an audio feedback. For example, a loss of health is accompanied with the avatar screaming in *Bioshock 2*; the use of slow motion is accompanied by a specific low-pitch sound in *Max Payne 3*; entering a new menu is accompanied by a specific menu sound atmosphere in *Dead Island*.

Moreover, some sounds extend beyond the graphic information directly presented to the player as objects in the diegetic world of the game can generate sounds even when out of the player's sight. They may not be easy to visually represent, but easily recognizable by the sound atmosphere. This is, for instance, the case of the avatar death in *Battlefield 3*, *Max Payne 3* and *Bioshock 2*; or the underwater sequence in *Bioshock 2*, where the HUD disappears from the screen, but the player is still able to move in the game world, while prevented to shoot. Such a sequence is detectable using the heavy breathing sound made by the avatar while trying to survive underwater.

Most of the player actions are accompanied by a matching sound, to validate them and give an instant confirmation feedback. This could be for instance the sound of the cash register when the player grabs money in *Bioshock* 2. That is why the audio stream should not be ignored when considering feedback based gameplay metrics, and Section 5.7 is dedicated to highlight the usefulness and the feasibility of automatic audio processing methods applied to automatic gameplay performance segmentation.

5.7.1 *Approach overview*

Similar to cross-correlation for image processing (see Section 5.4), a sound cross-correlation approach is employed to identify specific sounds within a multi-layered soundtrack (Roads, 1996, p. 509; Yarlagadda, 2010, Chapter 2). A similarity score is calculated to indicate when a comparison is located for a particular sound within a larger soundtrack. The higher the score, the more likely the two segments are a match.

5.7.2 *Step by step description*

The required inputs to execute a cross correlation for sound processing are:

1. The *gameplay footage* in the form of a sound file.
2. A *reference sound file* representing the sound to find. This can be extracted either from footage taken from the game, selecting a clear articulation of the sound desired, or alternatively, the game can be played with the musical atmosphere muted in order to be able to extract pure sound examples.
3. A *threshold value*, which is used in order to select the highest scores.

The algorithm works by scanning the reference sound [input 2] over the soundtrack [input 1], and comparing their amplitudes. Let n be the size of the reference sound and m the size of the whole soundtrack. The first n samples from the whole soundtrack are considered, and compared with the reference file using the normalized cross-correlation formula. The closest to 1, the best a similarity is detected. Then, and until having covered the whole soundtrack, the same process is repeated by shifting the considered samples from the soundtrack by one sample (n samples from the 2nd sample, n samples from the 3rd sample etc. until the m-n sample)

[S 5-6].

$$normCorr(i) = \frac{\sum_{j=0}^{n-1}\left(fullSoundTrack(i+j).referenceSound(j)\right)}{\sqrt{\sum_{j=0}^{n-1} fullSoundTrack(i+j)^2 . \sum_{j=0}^{n-1} referenceSound(j)^2}}$$

(i varying between 0 and m-n, and representing the reference sound shifting over the soundtrack).

Figure 77 illustrates the process using a randomly filled 8-element vector, schematizing a soundtrack, and a 4-element vector schematizing a reference sound. Figure 77 shows how is working the shifting, computing the correlation score for each shift. Shift 3 displays a perfect correlation score.

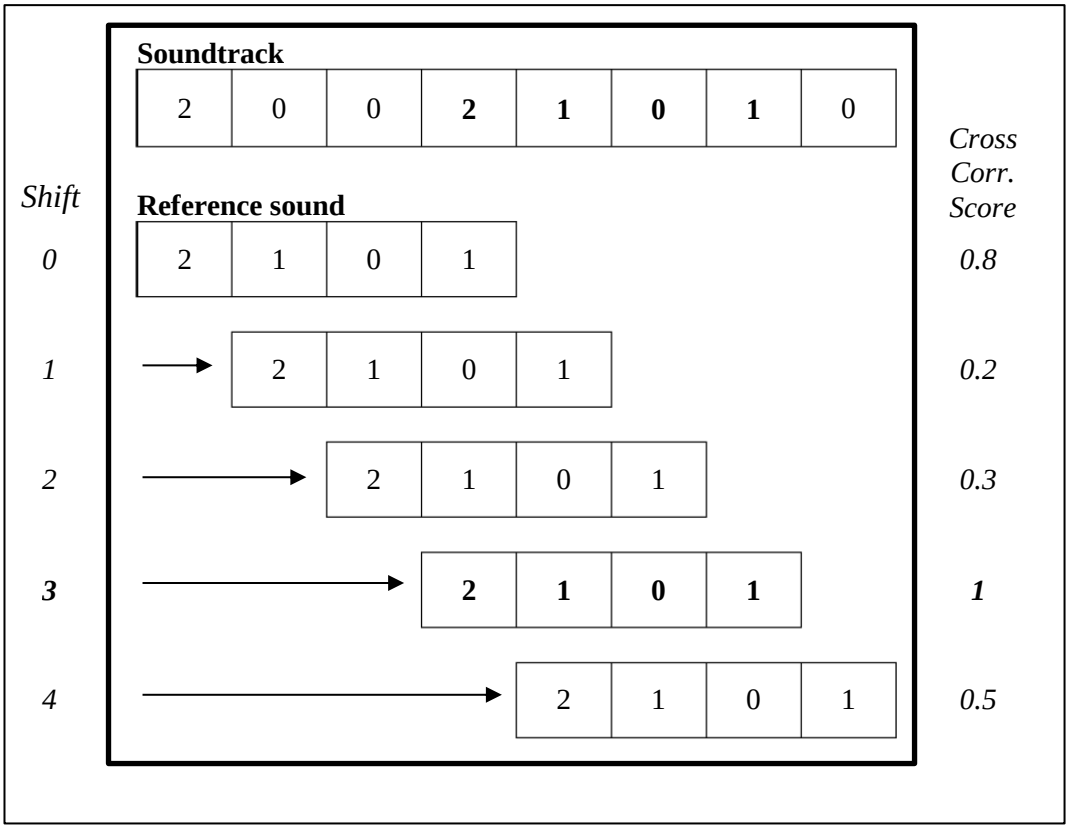


Figure 77. Sound shift and cross correlation

Figure 77 illustrates the sound cross-correlation process using diagrams. The soundtrack is represented using a vector of height integers, and the reference sound is represented using a vector of four integers. A correlation score is calculated between the element of the reference sound, and the matching ones in the soundtrack ([2 0 0 2] vs. [2 1 0 1] for shift 0, [0 0 2 1] vs. [2 1 0 1] for shift 1 etc.). Then the reference sound is shifted, and the process is repeated until the reference sound reach the end of the soundtrack. In Figure 77, the best-correlation (a perfect one) is achieved by shifting the reference sound by three samples.

The correlation result once the shifting is complete is a curve showing the similarity between the reference sound and every segment of the same size; the closest to 1, the best the similarity. Figure 78 illustrates the result of the process discussed above using a 35-second extract from the game *Battlefield 3*, in which the sound that accompanies screen-death is searched for. The figure shows that the correlations curve reaches its maximum at around 15 seconds, which is actually a death sequence. The time stamps matching the highest values (above a provided threshold), meaning the beginnings of the matching moments, are then considered as match (on Figure 78, only one time stamp will be logged, with the

value 15). The threshold value [input 3] is used to finally identify the accepted similarities [S 8-11].

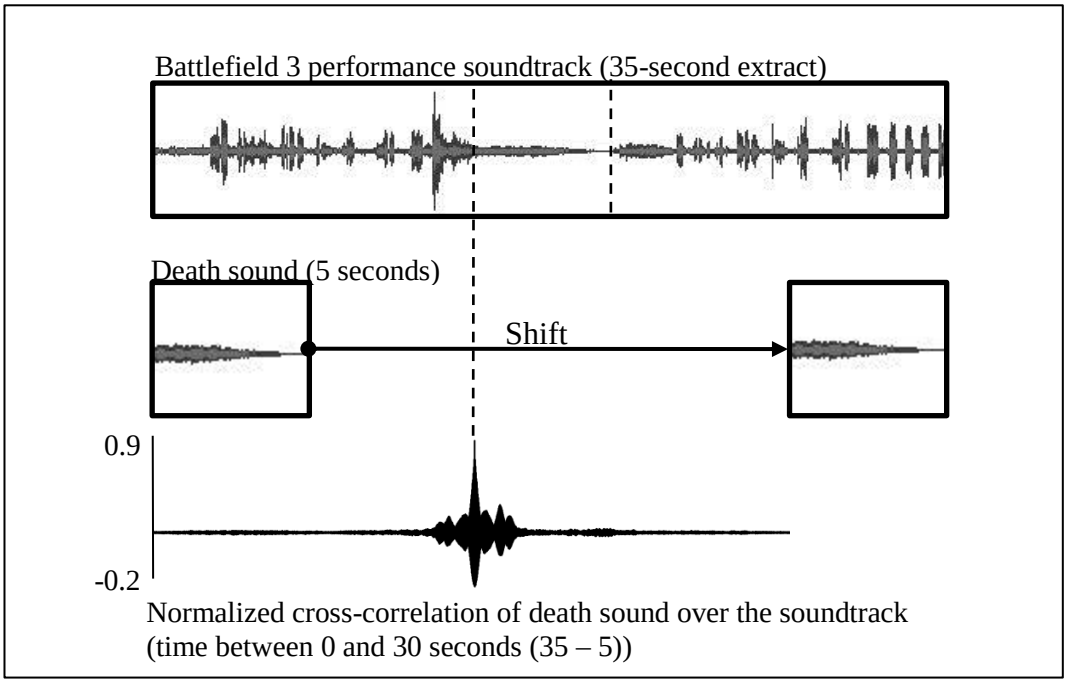


Figure 78. Sound cross-correlation using real sounds

Figure 78 illustrates the sound cross-correlation approach. The top image represents the waveform of a gameplay footage soundtrack (35 seconds). The middle image represents the death sound (5 seconds) that is shifted over the full soundtrack. A correlation score is calculated for each shift offset. The bottom graph represents the correlation scores for all the offset positions. The curve maximum represents the best correlation score, meaning that the 5-second (duration of the death sound) portion of the soundtrack starting at the detected offset is similar to the death sound.

5.7.3 *Sound detection (S) Algorithm*

Sound detection (S)
Inputs
referenceSound: mono sound */* sound file to find */*
fullSoundtrack: mono sound */* game soundtrack */*
threshold: float */* correlation value above which a similarity is detected */*

Output
result: vector of floats */* timestamps of each detection */*

Variables
normCorr: vector of floats */* result of the normalized cross correlation */*
m, n: integer */* sounds size */*
i, j: integer */* loop variables */*

Algorithm

$m \leftarrow nbSamples(fullSoundTrack)$
$n \leftarrow nbSamples(referenceSound)$

/ Sound normalized cross correlation */*
$\boldsymbol{for\ each\ position}\ i\ \boldsymbol{between}\ 0\ \boldsymbol{and}\ (m-n)$
$$normCorr(i) = \frac{\sum_{j=0}^{n-1}(fullSoundTrack(i+j).referenceSound(j))}{\sqrt{\sum_{j=0}^{n-1} fullSoundTrack(i+j)^2 . \sum_{j=0}^{n-1} referenceSound(j)^2}}$$

/ Determination of the detected similarities timestamps */*
$\boldsymbol{for\ each\ index}\ i\ \boldsymbol{between}\ 0\ \boldsymbol{and}\ size(normCorr)$
$\boldsymbol{if}\ (normCorr(i) \geq threshold)$
$$result.push\left(\frac{i}{getSampleRate(fullSoundTrack)}\right)$$

5.7.4 *Result*

Figure 79 illustrates the result of the process discussed throughout Section 5.7, across a full session of gameplay. In this result, the player dies five times, which also informs about the challenging sequences in *Battlefield 3*. Figure 80 is another example, using *Bioshock 2* this time, and illustrating the detection of a very complex 20-minute segment of play, with eleven deaths.

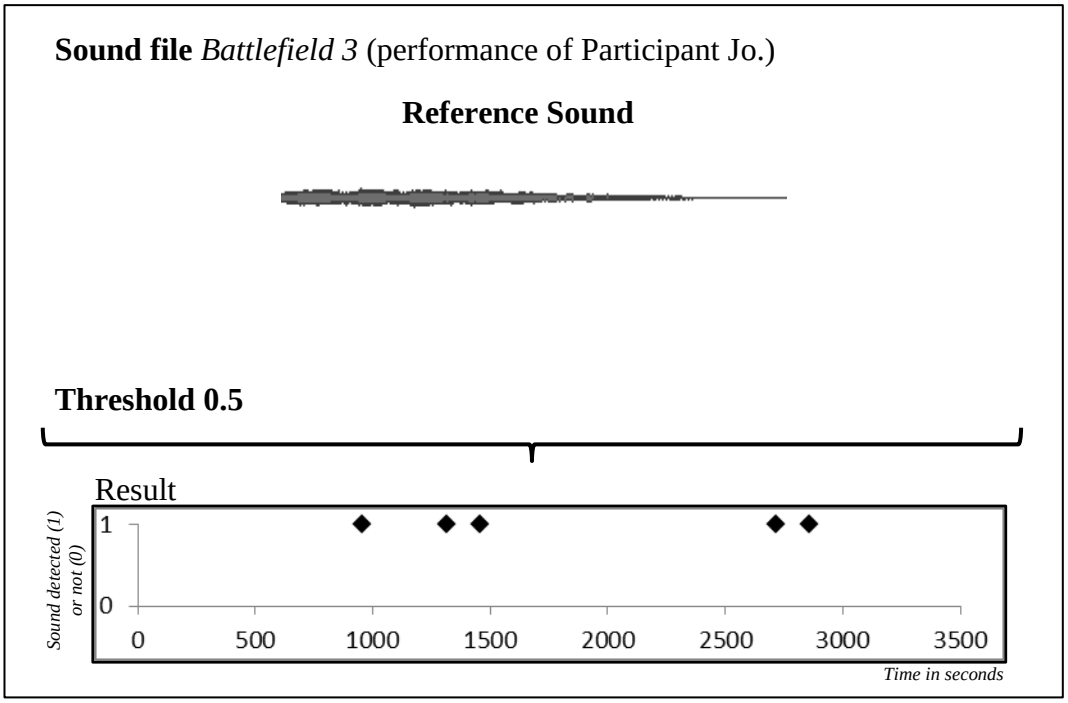


Figure 79. Death detection using sound in *Battlefield 3*

Figure 79 displays the result of the example used throughout Section 5.7. In this result, the player dies five times, and two challenging sequences can be identified around 1300 seconds and around 2700 seconds.

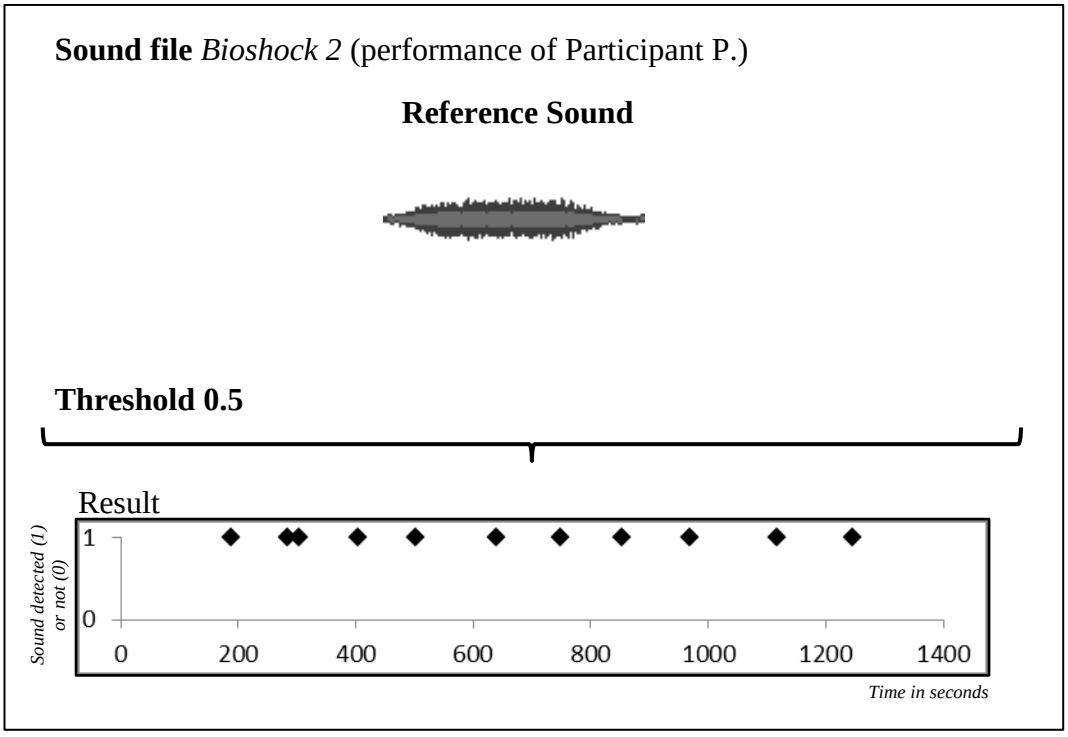


Figure 80. Death detection using sound in *Bioshock 2*

Figure 80 illustrates the use of the sound cross-correlation to detect death sequence in Bioshock 2, and displays a challenging sequence containing eleven deaths in twenty minutes.

5.7.5 *Validity*

The reliability of the sound cross-correlation algorithm is highly dependent on the size of the sound being detected. Screen deaths in Figure 79 and Figure 80 have been detected with 100% accuracy in large part due to the fact that sound lasts two seconds. However, Figure 81 shows the detection of an exploratory session in Bioshock 2, through the heavy breathing sound made by the avatar. A single uptake of breath only lasts one second, making the detection subject to incorrect identification. The correct breathing sound is identified between 2177 and 2418 seconds. However, as highlighted by Figure 81, some incorrect detection is present at 1352, 1490, 1617 and 2789 seconds. Incorrect detection is explained by the fact that a moaning sound that accompanies an enemy hit sounds like a breathing sound. It is possible to discard all the incorrect detections by considering the multi-layered structure hierarchically by combining the cross-correlation with HUD detection. Indeed, an exploratory segment in Bioshock 2 does not include HUD.

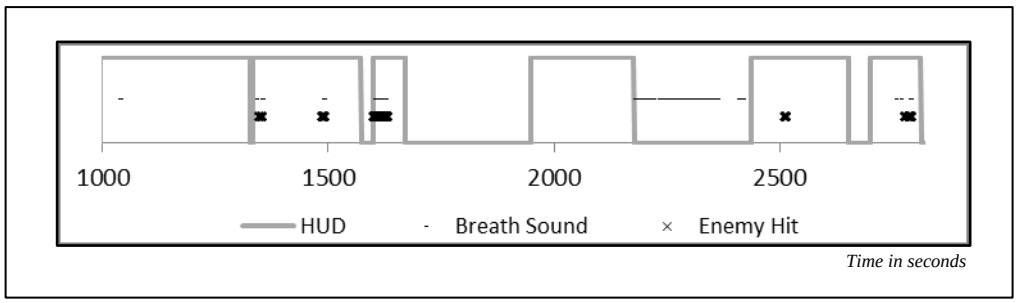


Figure 81. Underwater detection through breathing sound in *Bioshock 2*

Figure 81 illustrates how the sound cross-correlation is subject to false-alarms when the reference sound is short. In Bioshock 2, an exploratory segment, underwater, can be identified through the detection of the avatar heavy breathing sound. The correct exploratory segment in the above example occurs between 2177 and 2418 seconds. However, numerous false alarms are detected at 1352, 1490, 1617 and 2789 seconds. In fact, when the avatar is hit by an opponent, he is making a moaning sound that is similar to a breathing sound. That being said, Figure 81 also illustrates that it is possible to discard most of the false alarms by using other data to contextualize the detection. It is not only possible to discard the detections co-occurring with an enemy hit, but it is also possible to use the HUD information. Indeed, an exploratory segment in Bioshock 2 does not include HUD.

One limitation associated with this process is its reliance on the clarity of the sound. For most of the underwater sections, the breathing sound is simple to recognize. Yet towards the end of the section (around 2400 seconds in our example), a strong musical motif that contributes to the atmosphere of the game is introduced making the detection more complex. Nevertheless, the detection produces enough accuracy to correctly label the segment of play.

5.8 Conclusion

Chapter 5 has been dedicated to the presentation of methods derived from the audio and video segmentation area of the computer science disciplines, and applied for gameplay performance segmentation and, more precisely, the deconstruction process based on the five layers presented in Chapter 4. The algorithms presented in Chapter 5 are considered as being controlled-vocabulary, because they rely on gameplay concepts pre-determined by the analyst. Each algorithm has been presented, using detailed description, step by step examples, and commentary about the results and their validity. In order to assess the actual play experience, however, full gameplay performances have yet to be fully segmented and deconstructed using the algorithms presented in Chapter 5, in conjunction with the multi-layered model presented in Chapter 4. Chapter 6 will be dedicated to present full summaries of gameplay performances of several games, produced using the algorithms of Chapter 5, in order to provide a relevant experiential reconstruction. Chapter 6 will also provide an overview of what *free-text* algorithms can produce for the understanding of player experience.

Chapter 6

Interpreting Play

Chapter 5 was dedicated to presenting audio and video processing algorithms, appropriated from computer science research into audio-visual segmentation and indexing of multimedia documents. Such algorithms have been employed to gather feedback-based gameplay metrics from recorded gameplay footage, and accomplish a deconstruction of gameplay performance. A five-layered conceptual framework for segmentation is used to structure both data gathering and analysis. Each layer, as described in Chapter 4, represents a specific gameplay dimension (i.e., game system, game world instance, spatial-temporal, degree of freedom and interaction), making the nature of deconstruction conducted on each layer a *controlled-vocabulary* one, in the sense that the terms used for labelling the segments are pre-determined by the layer nature. Chapter 6 demonstrates the validity of the algorithms presented in Chapter 5. This is achieved using game sessions conducted with participants who provided the study with hours of gameplay footage from which it was necessary to discern the nature of the play experience (cf. Section 2.3). Chapter 6 focuses on illustrating how the algorithms presented in Chapter 5 can fully deconstruct a gameplay performance and provide full performance summaries, ready for player experience interpretation.

Additionally, Chapter 6 introduces alternative types of algorithms that may also be applied (given a different research agenda), not based on predetermined gameplay concepts, but based on the direct audio-visual content of the games. These algorithms, termed *free-text* algorithm (distinct from *controlled-*

vocabulary, see Section 3.4.4 and Section 3.4.5), can be used for the purpose of automatic performance comparison purposes. Section 6.2 is dedicated to the presentation of free-text approaches.

6.1 Experiential reconstruction of a deconstructed gameplay performance

Section 6.1 is dedicated to the presentation of gameplay performance summaries (representing participants gameplay, see Section 2.3) derived from the information achieved from feedback-based gameplay metrics, employing the algorithms presented in Chapter 5. Sections 6.1.2 to 6.1.6 show how a transversal reconstruction process can begin when performance summaries are available. The five modes (contextualisation, temporal frame of reference, semantic network, loop, comparison) of reconstruction presented in Section 4.2.4 will be employed in discussion of the four different games (*Bioshock 2* (2K Games, 2010), *Dead Island* (Deep Silver, 2011), *Battlefield 3* (Electronic Arts, 2011) and *Max Payne 3* (Rockstar Games, 2012)) played by participants (n = 10 per game, except *Dead Island* n = 1, as *Dead Island* was used as a pilot study to test the organisation of the gameplay sessions). The rationale behind using several games and players is to illustrate that the algorithms can be applied to the diversity found amongst videogames, and also account for that the diversity of approaches player can employ in their enjoyment and experience of games. However, before outlining a reconstruction process, it is important to fully present the data gathering process. This includes: 1) how data was selected, 2) how metrics affiliation to one of the five layers was decided (see Section 4.1) and 3) how the summaries were produced.

6.1.1 *Gameplay performance segmentation summaries*

All the figures included in sections 6.1.2 to 6.1.6 represent annotated versions of deconstructed performance summaries. The original versions of the summaries as well as an explanation of the used legend is accessible in Appendix B. Section 6.1.1 comments on the summary making process, and highlights issues that arose during the deconstruction phase.

The data gathering process

The different figures included in Appendix B and throughout Section 6.1 are summary of performances from footage recorded during the thesis field work (presented in Section 2.3). The footage has been used as the document representing the performance (see Chapter 4), and feedback-based gameplay metrics have been extracted from this audio-visual document using the five algorithms presented in Chapter 5. To actually process the video recordings, a software system has been developed, and used to extract the metrics of interest. The developed software system also constitutes a contribution of the present thesis, and will be described more exhaustively in Chapter 7.

To give relevant inputs to the software system, a pre-analysis stage of each game is necessary (as discussed in Chapter 4), in order to identify key logos of interest, or sounds representative of specific gameplay actions. Once the inputs have been determined, different algorithms are then executed, generating curves as outputs (converted into “timestamp – value” format), each curve representing one specific feedback-based gameplay metric. Finally, the different metrics are assembled into a single graph, in order to study their co-evolution and characterised the nature of

players' gameplay experience. For creating the summary graph, a spreadsheet application (Microsoft Excel ®) has been used. In order to support a reconstruction, the figures included in this chapter not only display the combined metrics for the same performance, but they also indicate the metrics layer of affiliation (see Figure 139) and the algorithm used to process them (see Figure 140).

Data selection and layer affiliations

As mentioned in the above paragraphs, two manual phases are required in order to generate the performance summaries presented in Section 6.1. Because the inputs selection and the metric assembly phase are manual, these phases are dependent on several subjective choices determined by the analyst.

In terms of inputs selection, a comment regarding the diversity of games with regards to their audio-visual contents is that the amount of symbolic information broadcast to the player varies from one game to the other. Both *Bioshock 2* and *Dead Island* rely heavily on symbolic feedback to communicate with the player, which can be exploited to generate exhaustive summary of performances. In contrast, *Battlefield 3* and *Max Payne 3* have less symbolic elements, generating then smaller summaries. However, as the following sections will demonstrate, it is still possible to experientially reconstruct a performance even when the amount of metrics is reduced.

The second comment is about the terms used to describe the metrics. The summaries presented in the present section are based on a *controlled-vocabulary*

method of segmentation, meaning that the terms used to describe the segment of play need to be pre-determined by the analyst. In the following figures, most of the terms are self-explanatory, or will be explained when used in the following sections. For instance, the use of "cut-scene" is different from "in game cut-scene" with the former representing a total change of space, while the latter is a cut-scene in continuity with the game action, in general more representative of a change of degree of freedom (the player cannot move) than a change of game space. In the *Bioshock 2* summaries, "enemy hit" represents a visual confirmation, during close-combat, that an enemy has been hit by the player. "XP to spend", in the *Dead Island* summary, represents a visual warning asking for the player to spend the XP points they gained during previous fighting sequences. "Friendly fire", in the *Battlefield 3* summaries, represent moments when the player accidently shot a teammate.

A third comment can be made about the choice of algorithms. In general, choosing an algorithm is straightforward, as it relies on the way information is broadcast to the player. But some metrics, identified as (-) on the summaries, are constructed not by direct processing of audio-visual footage, but by studying other metrics conjointly. This is for instance the case of "in game" in *Max Payne 3* summaries, that is actually build upon the idea that "in game" are the segments that remain when "intro cut-scene", "main menu" and "cut-scene" are not detected.

Finally, a comment should be made about the phase when a metric is affiliated to a layer. Affiliating a metric to a layer is in general straightforward, but sometimes,

a choice was required where the metric could represent one or more layers. This is the case with "death" in the presented summaries. "Death-screen" has been attached to the degree of freedom layer, as in general, the players remain in the same space, but have their control removed or reduced. However, a death sequence can also contain specific elements, like strong colour changes that can be considered as belonging to the spatial-temporal layer. In *Max Payne 3* for instance, when the player dies, a screen-shot of the last action is taken, and replayed in a "comic-strip" effect. The "comic-strip" screen can be seen as another game space. In the current thesis, the choice has been made to connect death-screen with the degree of freedom layer, but the spatial-temporal layer could have been a solution too. Another example is the "cut-scene (chapter)" label that can be found in the *Max Payne 3* summaries. The metric is actually the combination of two metrics. In *Max Payne 3*, a cut-scene can actually be identified by the presence on-screen, for a couple of seconds, of the word "Chapter", indicative of the beginning of a new chapter in the game. Detecting the word "chapter" gives to the full segment the "cut-scene" label. The full segment relates to the spatial-temporal layer, but the on-screen display of the word "chapter" is a warning, belonging to the interaction layer. The segment is a spatial-temporal one, but its detection relied on an interaction. On the summaries, both metrics are represented on the "cut-scene (chapter)" line: cut-scene as the line, chapter as the cross (see for instance Figure 86).

Summary selection

The final discussion in the present section relates to the choice of the summaries that illustrate the reconstruction process. The main rationale of the present thesis

is to present a new method for gathering gameplay metrics, and link this method to a gameplay performance segmentation process. That means that, from the fifty hours of recorded footage, a sub-set of performances has to be selected, in order to present and discuss the method. Analysing the fifty hours of footage would mean going to another level of detail, going far beyond the methodological aspect of the present thesis. However, the presented summaries have not been randomly selected[26]. For the game *Bioshock 2*, two summaries are taken from the same participant, at two different points in the game, allowing for a reconstruction that takes into account the progression and development of a player's playing style. For the game *Battlefield 3*, two summaries are taken from two different participants, at the same point in the game, allowing for an experiential reconstruction that compares different play styles and behaviours. Finally, for the game *Max Payne 3*, two summaries are shown from two different participants. However, one of the summaries is taken from the same participant who also played the game *Bioshock 2*, allowing for an experiential reconstruction comparing how a player engages differently with different game conditions.

Now that the rationale for presenting the summaries used in Section 6.1 has been presented, the sections 6.1.2 to 6.1.6 can focus on the actual reconstruction process, and experiential understanding of the summaries. Five modes of reconstruction are presented: contextualisation, temporal frame of reference, semantic network, loop and comparisons. The reconstruction process is defined as

[26] Apart for the *Dead Island* performance, as *Dead Island* has been used as a pilot study, and only one performance has been recorded

transversal (see Chapter 4) as the metrics can be combined, regardless of their layer affiliation, to provide better insight about players experience. The remaining subsections of Section 6.1 are devoted to presenting the summary results, in order to illustrate the validity of the algorithms presented in Chapter 5, and also to demonstrate how the reconstruction phase presented in Chapter 4 can be built upon these summaries.

6.1.2 *Contextual interpretation of metrics*

The first transversal reconstruction presented here addresses *contextual* interpretation of metrics (introduced in Section 4.2.4), that is sequences in which a specific gameplay metric offers context necessary for understanding several other metrics. C*ontext* can be 1) *temporary*, being an interval delimited by a beginning and an end, inside which a change occurs that contributes to an understanding of other gameplay elements; 2) *perennial*, being a specific point after which the particular change is then persistent for the rest of the performance; or 3) *hierarchical*, referring to an interval during which a particular subset of metrics are known to be operational. These three contextual approaches are outlined in greater detail below.

In the following figures, the annotations should be interpreted as follows: 1) each vertical line represents one of the context boundaries(s) (two boundaries for the temporary and hierarchical contexts, and one for the perennial context); 2) the circled metric represents the metric that determines the boundary; 3) the arrow represents the context interval.

Perennial context

In Figure 82, the game *Bioshock 2* is used as an example to illustrate a reconstruction based on *perennial context*. The first session of Participant P. displayed in Figure 82 represents the very first time the player interacts with the game. The first minutes of the game are then dedicated to learning the controls, via a tutorial. The player begins the game with a basic set of authorized actions, but as the game progresses they are progressively allowed to use more and more

gameplay features. At around 720 seconds in Figure 82, it can be observed that the player encounters their first enemy (indicated by an "enemy hit" and a "loss of health" interaction). After this point, the game mechanism (at the game system layer) can be seen as enriched with several new gameplay possibilities that the player can, from now on and until the end of the game, execute. The loot option becomes available, by its introduction and first detection at 720 seconds (degree of freedom). The player can then collect items, and the player's health can also be impacted (adding the gameplay possibility of "game over"). In terms of story, this moment also represents the very first time the player encounters their main enemies.

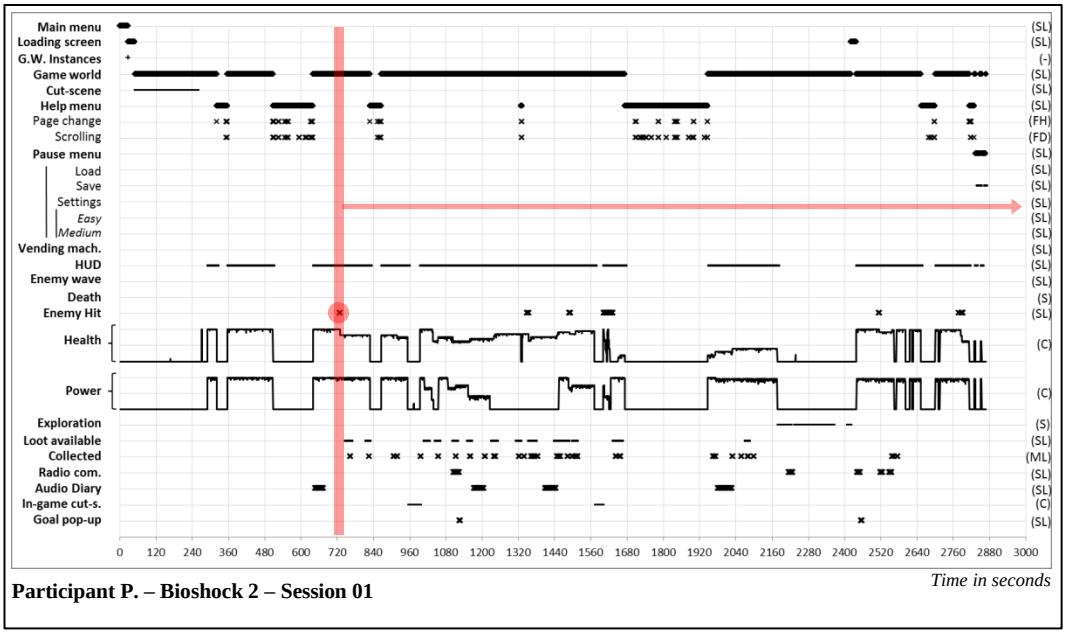


Figure 82. Perennial context: after the first encountered enemy (*Bioshock 2*)

Figure 82 illustrates a perennial change of context, using a performance from the game Bioshock 2. The first session of Participant P. is mainly based on tutorial sequences, meaning that the player is progressively given more and more skills. At around 720 seconds, the player encounters their first fight, being a perennial change in terms of game system. After the first fight, "corpse looting" and "element collecting" become available for the rest of the game; while the player also knows, in terms of story, who are the main enemies in the game.

Another remark on Figure 82, concerns activity at the 1000 seconds mark, which illustrates the first “power drop”. This is indicative of the approximate moment when the player gets a “power usage” tutorial, which can also be seen as a perennial change. Again, from this point onward, the player has been shown how to use power adding it to the repertoire of behaviours available to the player. Perennial changes are important to detect, when possible, as they explain player’s choices.

Temporary context

Figure 83 and Figure 84 display an example of what is described as *temporary context*, using the fourth session of Participant P. This illustrates the notion of a change that is able to explain specific player behaviours, until the change is reversed. Figure 83 displays intervals in which the player encounters "enemy waves". During these periods, the game system introduce a boss-like challenge (Zagal et al., 2008), meaning that the player is presented with more and tougher enemies. These intervals are interesting to evaluate for player's reactions to sudden health drops and challenging enemy confrontations. In between 150 seconds and 1200 seconds, the waves are very short, ending with the player's screen death. However, before entering the next wave, the player collects as many items dropped by the previously killed enemies as they can. This can be interpreted as a desperate response as the player tries to locate the few items that can provide some help, but without any real success. However later, between 3000 seconds and 3120 seconds, a longer enemy wave can be detected. Yet the player is still alive at the end of the interval, and a rise of health shows that the player is finally able to cure themselves to overcome the challenge. The player, probably more ready for the fight, is in a much more "controlled" behaviour. By identifying the context of play, it is possible to assess how a player engages not only with specific instances but also similar challenges over the course of the performance.

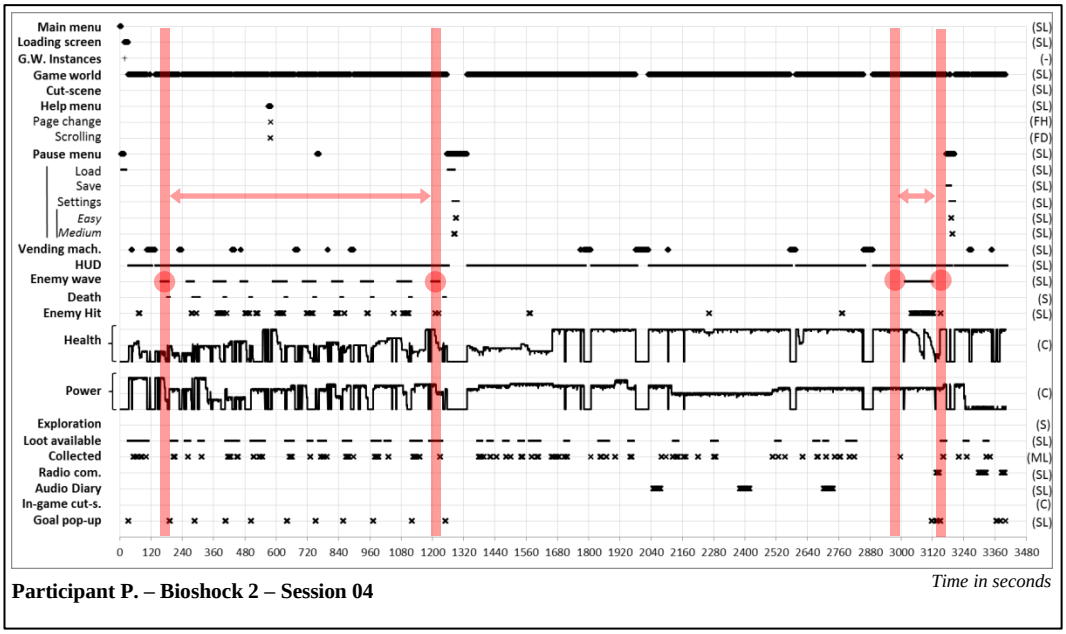


Figure 83. Temporary context: enemy waves context (*Bioshock 2*)

Figure 83 illustrates a temporary change of context (implying the detection of an interval, where the change of context starts, and when it ends). Here, the context deals with "enemy waves" intervals, during which the player is confronted with a difficult "boss-fight" challenges. The first interval, between 150 and 1200 seconds shows a chaotic behavior from the player (unsuccessful actions resulting in deaths, health drops, items collection as much as possible before fights, etc.). The second interval, from 3000 seconds to 3120 seconds show a more in-control behavior (successful actions resulting to the avatar still being alive, possibility to cure as illustrated by the rise of heath in the middle of the interval, etc.). Detecting these intervals allows to compare more accurately players' behavior throughout a performance.

Figure 84 illustrates another temporary contextual change. At 1300 seconds, by interacting on the difficulty level from within the pause menu space and the settings degree of freedom. Then, the player is consciously impacting the game system layer, as the whole rules are adapted to make the game less challenging. At 3200 seconds, the difficulty is put back to its original state.

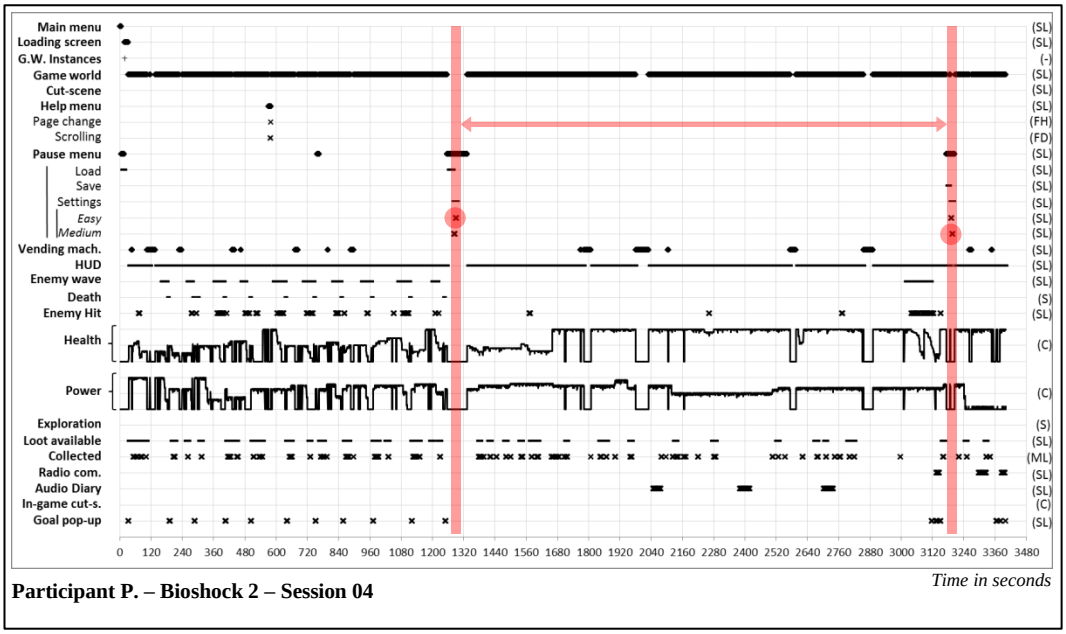


Figure 84. Temporary context: easy difficulty context (*Bioshock 2*)

Figure 84 illustrates another temporary change of context with the game Bioshock 2. Between 1300 and 3200 seconds, the player changes the difficulty from medium to easy (medium to easy interaction detected at 1300 seconds, and easy to medium interaction detected at 3240 seconds). This change of context, impacting the game system by making the enemies less challenging to fight, informs about the player's change of behavior as they are less threatened by enemies. The player is more into a preparatory state of mind, collecting much more items than actually fighting. Then, at 3000 seconds, the player fights again inside an enemy wave (see Figure 83), and succeed, due to both the collection, and the easy difficulty context.

During the "easy" interval, less health drops are indicated, as well as fewer enemies being hit. At the same time, more items than usual are collected; so the player is able to collect and fight under the easier condition (as determined by the game system). The enemy wave interval highlighted in Figure 83 (between 3000 seconds and 3120 seconds) occurs during the "easy" interval, probably also explaining part of the player's success to overcome it. During the "easy" interval, prior to the final "wave", the player takes time, profiting from the easier condition

or execution of the game rule system, allowing the player to replenish their inventory (item collection), explaining the rise in health during the “wave”. The player is equipped with enough items to cure themselves when hit or damaged. However, once the “wave” has been passed successfully, the player decides to return the difficulty level to “medium”.

Hierarchical context

The three remaining examples in the present section are taken from the games *Battlefield 3* and *Max Payne 3*. They illustrate how a context can also be used to highlight elements that require the attention of the analyst. “Hierarchy” is used purposely here to evoke the hierarchical use of layers employed during the deconstruction phase. In this process, some metrics are used to guide the processing of other metrics. The hierarchy can then also be used for reconstruction purposes. In Figure 85 for instance, the notion of aiming using a viewfinder in *Battlefield 3* makes sense only when there is a distant enemy to aim for, typically employed when at the degree of freedom layer, the game indicates the presence of distant enemies.

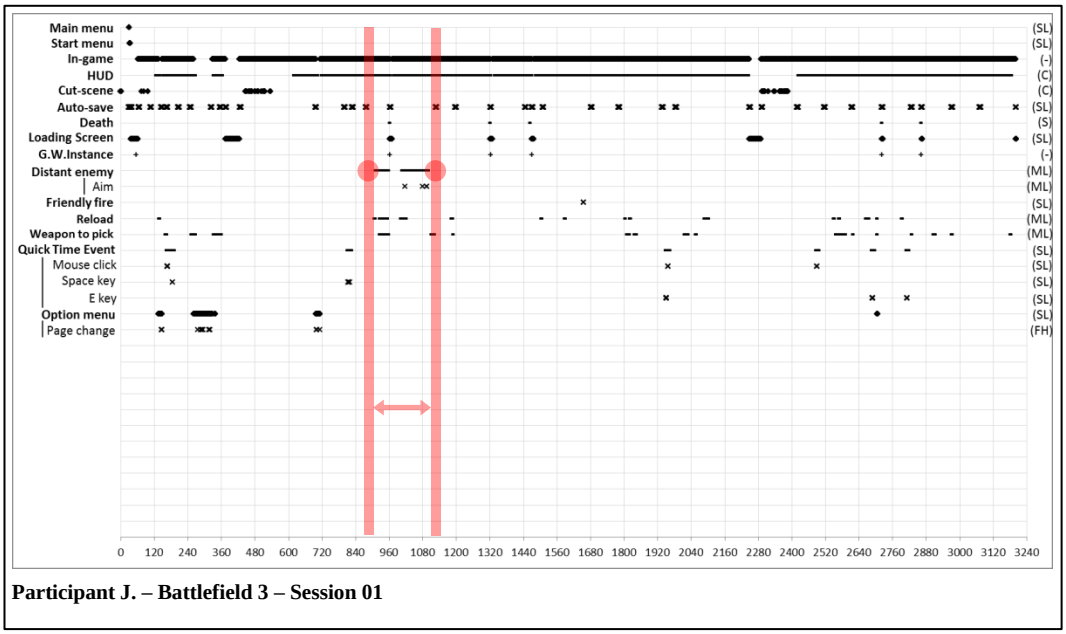


Figure 85. Hierarchical context: aim while distant enemy (*Battlefield 3*)

Figure 85 illustrates the notion of “hierarchical context” using the game Battlefield 3. In this example, the notion of “Aim” only makes sense when “Distant enemies” are on screen. So the hierarchy is used here to guide the analyst, by focusing on what is “possible”, and discarding what is “impossible” during the interval.

The hierarchy can also be used in the game *Max Payne 3* in Figure 86, where a "skipping" interaction needs to be contextualized by the detection of a cut-scene space (as skipping is an action tied to the cut-scene).

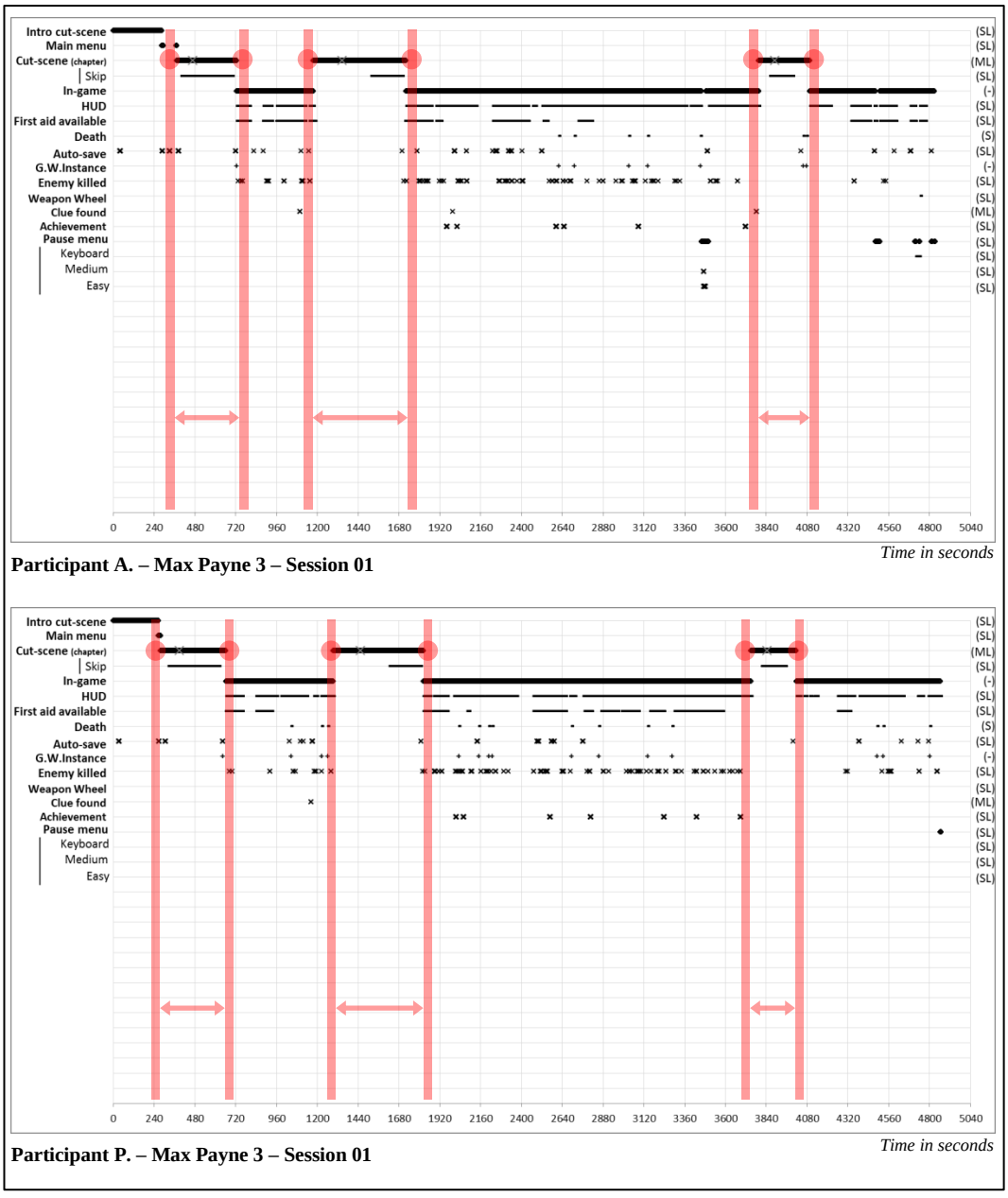


Figure 86. Hierarchical context: skip during the cut-scene (*Max Payne 3*)

Figure 86 illustrates the hierarchical context using the game Max Payne 3. Here, the interval is used to clarify the sequences where the "skip" interaction makes sense, i.e. during the cut-scene. "Skip" is not a possible action outside the "cut-scene" space.

In Figure 87, this is the opposite of the example displayed in Figure 86. The intervals displayed in Figure 87 reflect an "in-game" context. During this context "skipping" is a meaningless interaction (because outside the cut-scene space), but most of the other elements (HUD, health, first aid, death, etc.), meaningless during the cut-scene, can be this time considered by the analyst.

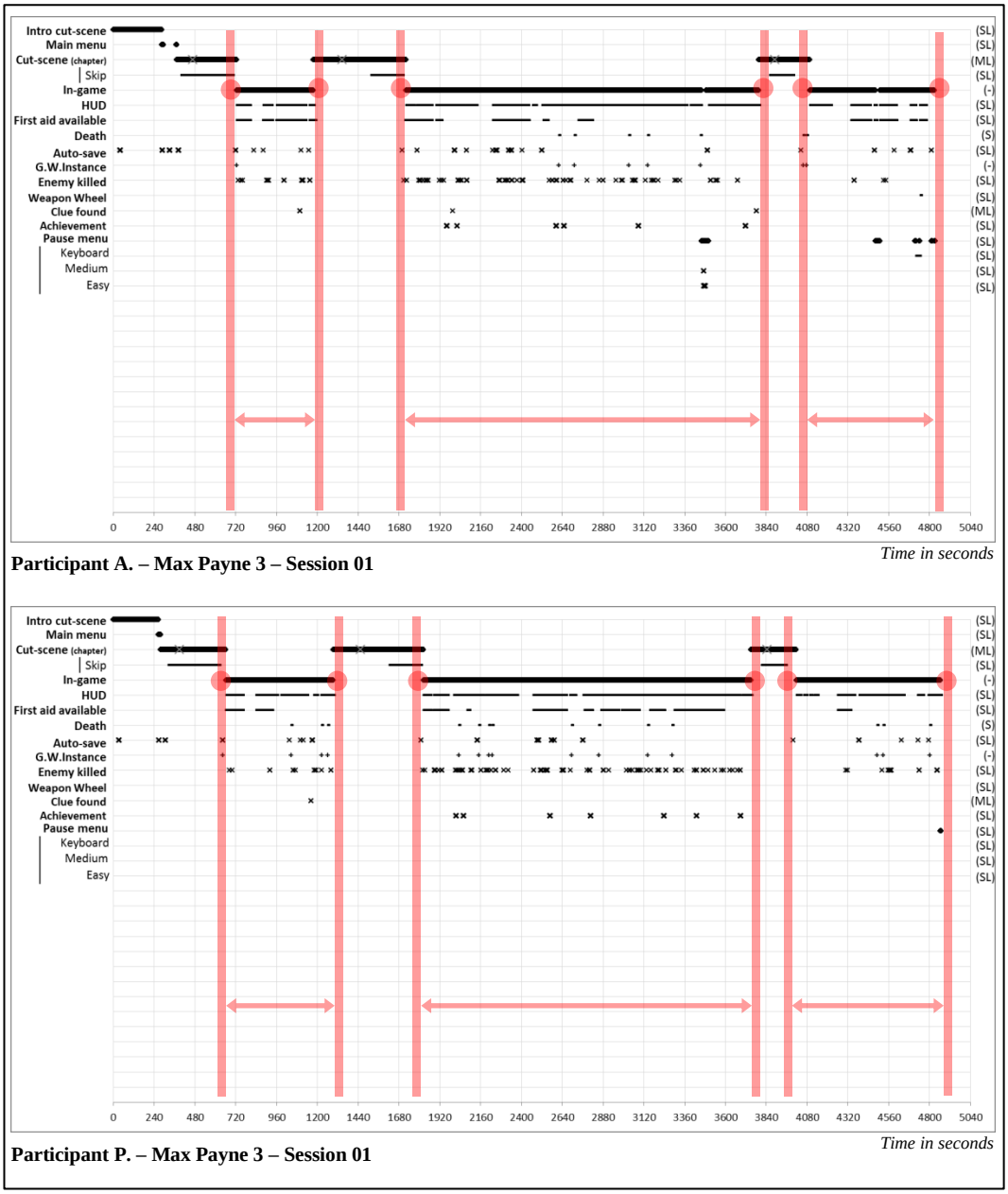


Figure 87. Hierarchical context: large interactivity during in-game (*Max Payne 3*)

Figure 87 is linked to Figure 86 in the sense that it illustrates the opposite, i.e. the sequences where "skip" is irrelevant, but where all the other interactions are possible. Figure 87 shows how in-game interval can help focusing on the relevant actions.

A *contextual* reconstruction is useful for identifying moments that possess a transversal quality. A context detection is able to highlight change for the rest of the performance (perennial) or for a substantial period of time (temporary), as well as guiding the metrics of interest for the analyst by discriminating the ones that are relevant to consider from the ones that should be temporary discarded (hierarchical). The next presented transversal reconstruction mode, focussed on temporal frame of reference, is about identifying a group of metrics that require the analysis of other metrics, precedent, co-occurring, or subsequent, to be fully understood.

6.1.3 *Temporal frame of reference*

The second transversal reconstruction deals with how play is interpreted based on information preceding, concurrent or subsequent to the section under consideration, in attempt to better understand a gameplay sequence, or a group of sequences, by analysing prior metrics (*preceding*), co-occurring metrics (*concurrent*), or metrics appearing later (*subsequent*). By using the four games from the current thesis field work (see Section 2.3), the present section provides examples, illustrating *preceding*, *concurrent* and *subsequent* analysis of groups of metrics.

In the following figures, the annotation should be interpreted as follows: 1) each circled metrics represent a metric (or group of metrics) of interest; 2) an arrow between two group of metrics represents a temporal link between them (going right for preceding, going left for subsequent, remaining at the same time frame for concurrent). Sometimes, for readability purposes, a figure does not contain any arrow. That means that the circled metrics need to be interpreted concurrently (they are too close in the graph to properly draw an arrow).

Preceding events

The first example is linked to the *Bioshock 2* example presented in Section 6.1.2 in which a change of difficulty instigated by the player is connected to the outcomes shown in Figure 83 and Figure 84. Figure 88 illustrates that the player's decision to change the difficulty of the game system is preceded by ten screen deaths that occur in rapid succession. After the change of difficulty level (at around 1300 seconds), no further screen death are noted. So, at this juncture of the game, the player assumedly is frustrated by the consecutive screen deaths that they decided to change the difficulty level instead of trying another strategy. From a processing perspective, the player's decision to change the difficulty is not random, in the sense that the player could have randomly decided to change the difficulty with no apparent reason. Instead, preceding metrics showing the ten deaths justify, the player's motivation behind the particular difficulty change occurring at 1300 seconds[27].

[27] Using preceding metrics, the second difficulty change, occurring at around 3200 seconds can also be explained, as the change is preceded by the successful "enemy wave", showing that the player finally overcomes the challenge, and is ready to be confronted again with the medium difficulty.

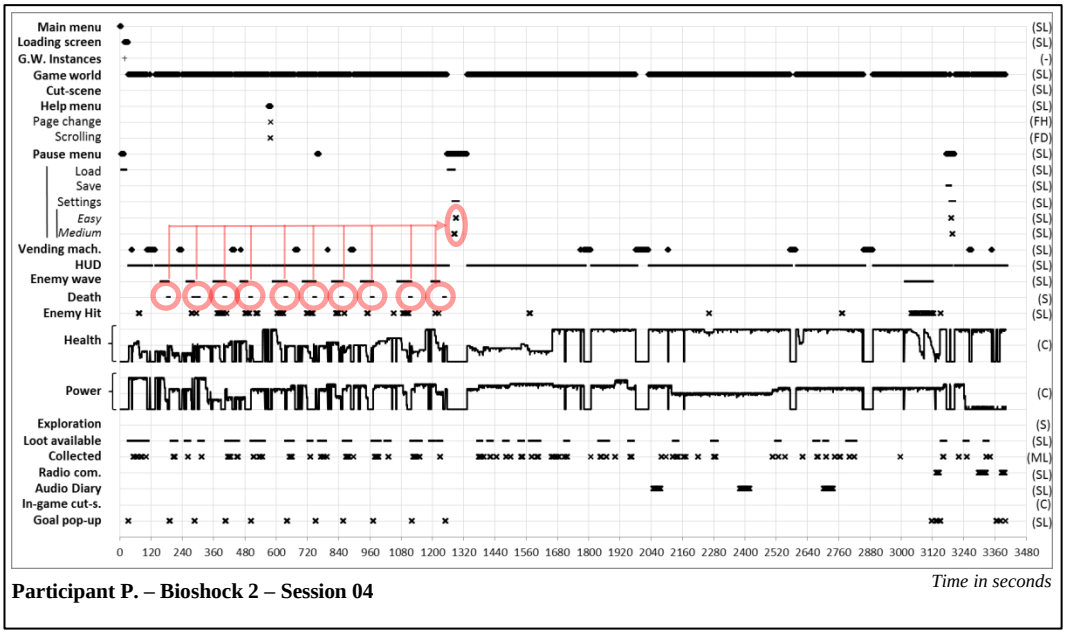


Figure 88. Preceding metrics: consecutive deaths leading to difficulty change (*Bioshock 2*)

Figure 88 illustrates the notion of "preceding metrics" using the game Bioshock 2. During the performance, the player changes the difficulty from medium to easy (at around 1300 seconds, see also Figure 84). This change of difficulty can be explained by considering the ten previous, and consecutive, deaths. The change of difficulty can then be understood as response to a frustration feeling, originating from the long, and difficult, sequence that the player just encountered.

Figure 89 illustrates the same process with the game *Max Payne 3*. After the player's first five consecutive deaths, they again take the decision to change the difficulty before re-attempting to play the section. This leads the player to succeed and complete the game level.

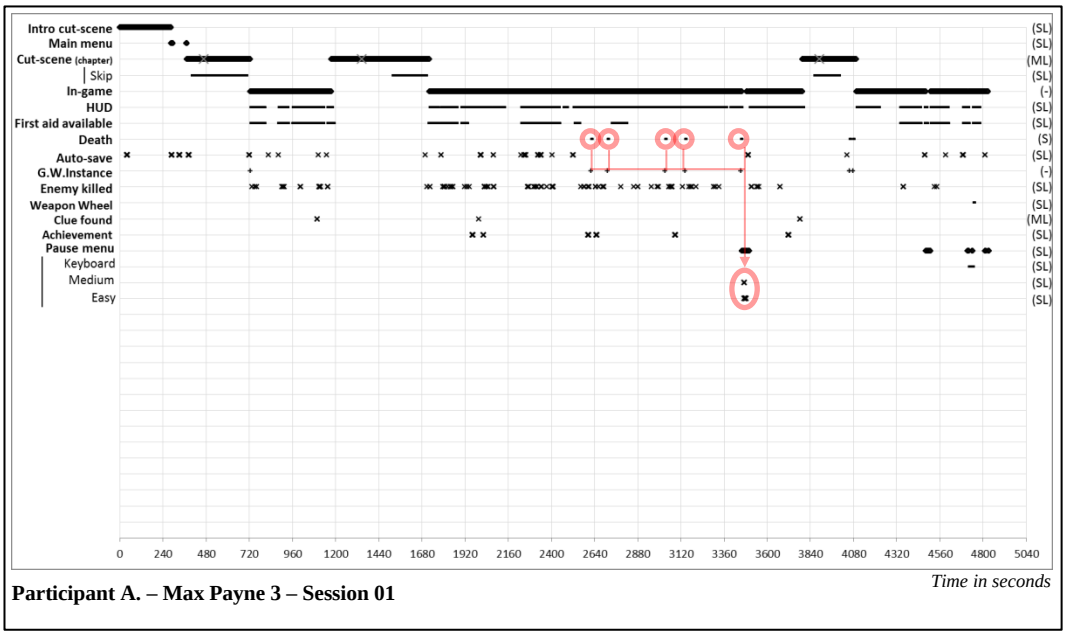


Figure 89. Preceding metrics: consecutive deaths leading to difficulty change (*Max Payne 3*)

Figure 89, similar to Figure 88 but with the game Max Payne 3, illustrates how a change of difficulty can be explained by the detection of a prior sequence of consecutive deaths. Here, the player encounters their first difficult sequence (no death has been detected before 2600 seconds), but has been unable to overcome it without changing the difficulty.

Another example of using preceding metrics from *Bioshock 2*, is taken from the first session of Participant P. Figure 90 displays that the player is inside the help menu space between around 1680 seconds and 1920 seconds. Within the help menu space, no real action occurs. The player is only presented with text they can read, that will inform them about *Bioshock 2*'s gameplay mechanisms and the game world of Rapture.

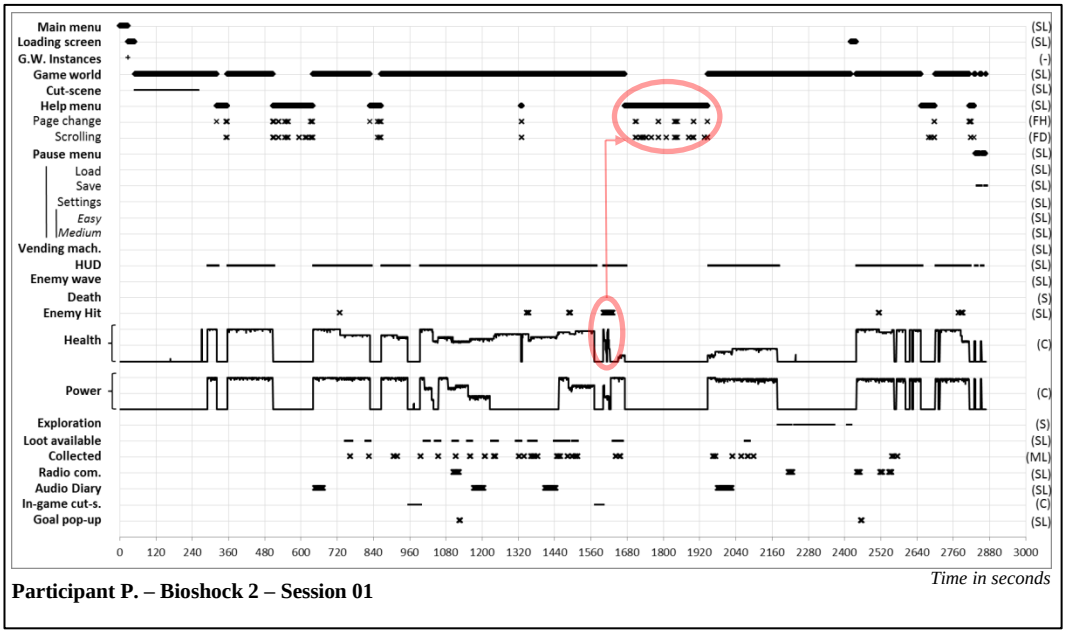


Figure 90. Preceding metrics: difficult session leading to help menu (*Bioshock 2*)

Figure 90 displays another example of "preceding metrics" usage, using the game Bioshock 2. Between around 1680 and 1920 seconds, the player spends four minutes inside the help menu space. This is not a common behavior, as the help menu space is pretty empty in terms of actions; as the player is only presented with text they can scroll and read. By analyzing the past, it is possible to understand why the player choses this particular moment to spend four minutes inside the help menu. Indeed, at around 1600 seconds, the player encounters a very complex fight, that almost kills the avatar. The player, probably insecure about their current skill, decides then to improve their knowledge of the game mechanisms and the game world.

In this instance, it is sensible to wonder why the player choses this moment to spend a significant amount of time (approximatively four minutes) inside the "help menu" space. The rationale for staying in the "help menu" is again understood by considering the precedent metrics. Figure 90 shows that, at around 1600 seconds, the player encounters their first demanding fighting sequence, with

two important and consecutive losses of life (despite the use of first aid kit to replenish health level). With this information, the player's actions points to the more likely explanation of the player requiring further knowledge of the game mechanisms and the game world (to understand the enemies they were fighting against). It is possible to suppose that the player probably felt insecure after being confronted by their first intense fight, and needed to reinsure themselves, by updating their knowledge of the game.

A fourth example is finally presented, in order to highlight the application of the reconstruction on different games. Figure 91 illustrates another use of preceding metrics, using the experience (XP) mechanism in the game *Dead Island*. In this zombie FPS game, the player gains XP points every time they kill a zombie, or successfully accomplish a given mission. Once the player has been awarded enough XP points to upgrade their avatar abilities, the game regularly prompts an on-screen message to invite the player to spend the XP points ("XP to spend") in Figure 91, at 100, 300, 500, 650 and 950 seconds. The player in this instance ignores the first four prompts, but at 950 seconds one, s/he decides to follow the prompt, and enters the XP menu space. The question can be raised as to why, at this specific moment, the player decides to follow the prompt, having ignored it four times before.

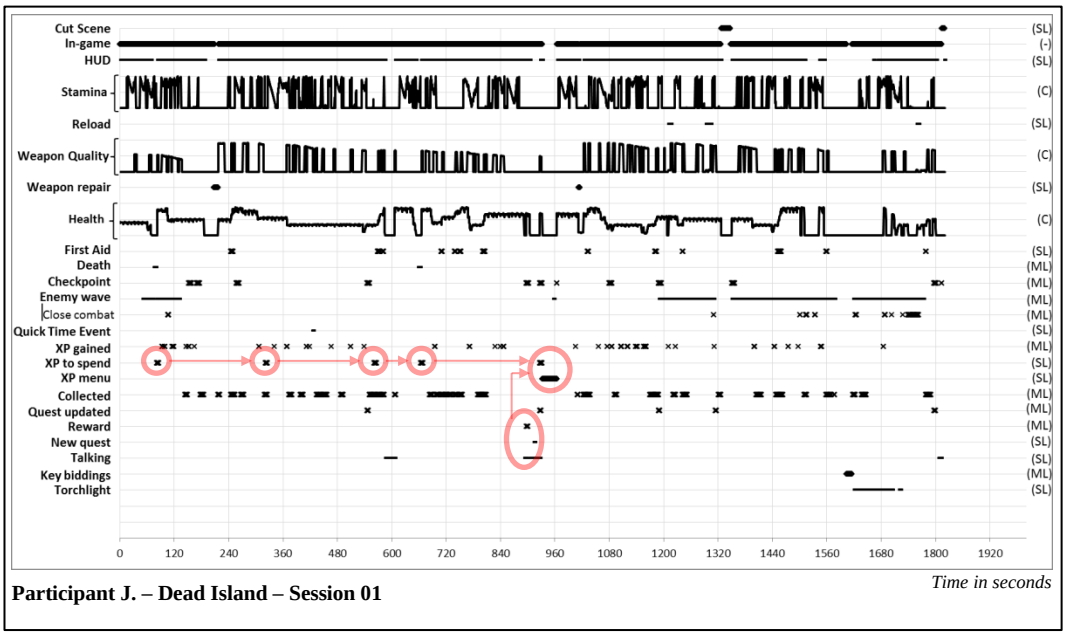


Figure 91. Preceding metrics: new quest leading to the use of XP points (*Dead Island*)

Figure 91 provides another "preceding metrics" example, using the game Dead Island. At around 950 seconds, the player enters the XP menu, meaning that the player is in a space where they can spend "XP points" to improve the avatar abilities. However, a warning panel, inviting the player to spend their "XP points" has been displayed at around 100, 300, 500 and 650 seconds, and the player ignored them. The reason why the player decides to follow the warning panel at 950 seconds can be explained by considering other past metrics. Indeed, at around 900 seconds, the player finishes a quest (reward), and accepts a new one (new quest). So, at this particular moment, the warning panel makes sense for the player, as they need to prepare for the new quest to come.

By examining preceding metrics that account for experiences leading up to the use of XP, it is possible to see that, at around 900 seconds in Figure 91, the player has just completed a mission (indicated by the “reward” interaction), and is presented with a new mission (indicated by the “new quest” panel degree of freedom). At this very moment, the player is in a preparatory state of mind, as they successfully accomplished a mission, and need to be prepared for the next one. That is likely the reason why the player finally takes time to upgrade their avatar for the upcoming challenges. Before, the player was in the heat of the action, and probably did not even notice the advice panel.

Concurrent events

This section addresses how gameplay is assessed via co-occurring metrics. Figure 92 represents when a player in *Bioshock 2* is presented with a "new goal", which presents itself on a pop-up panel. New goals also appear when a player dies serving to remind the player of the current aim of play. Pop-up panels indicating the player's prescribed goal have two meanings when presented for the first time (new game sequence) or repeated (reinforced as a consequence of failure). By examining the pop-up goal alongside the co-occurrence of death, it becomes possible to discriminate the actual meaning of goal pop-up panels. The first instance (30 seconds), and last two (3120 and 3360 seconds) are indicative of the player progressing in the game (as they do not co-occur with death), while all the other instances (200, 300, 400, 500, 650, 750, 840, 960, 1080 and 1300 seconds) are indicative of a need to repeat sequences of play.

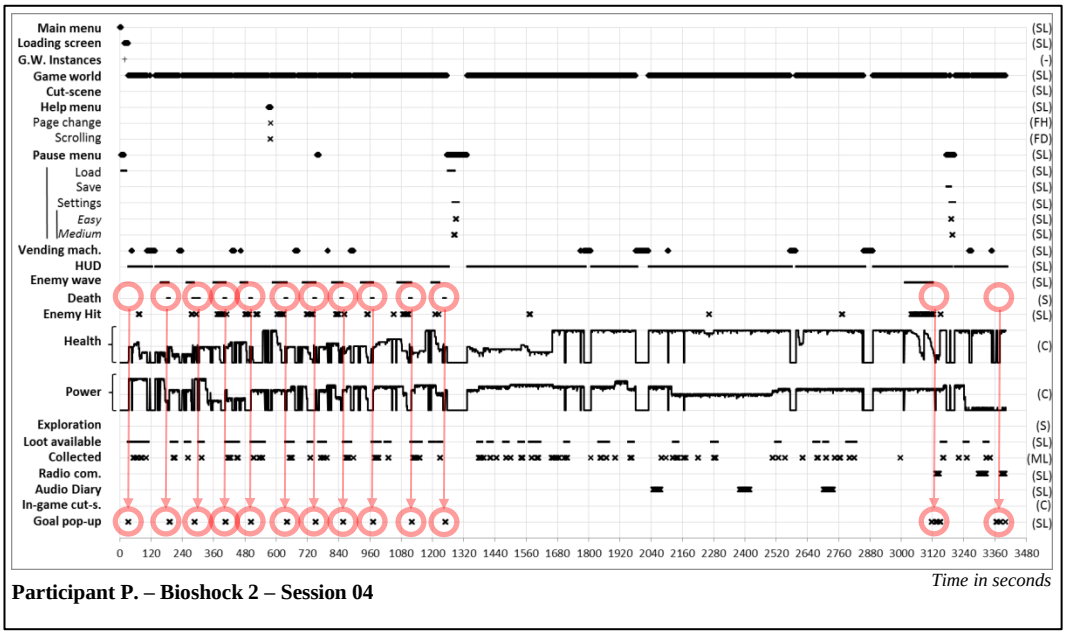


Figure 92. Concurrent metrics: goal pop-ups discriminated via conjoint analysis of death (*Bioshock 2*)

Figure 92 illustrates the use of "concurrent metrics" with the game Bioshock 2. The idea is to explain the role of the goal pop-up by looking for potential co-occurring death. If no death is co-occurring, then the goal pop-up is an indicator of the player moving forward in the game (first and two last annotations, from left to right). If a death is co-occurring, then the goal pop-up is a reminder, indicative of the player doing the same action again (all the other annotations).

The second example uses the game *Dead Island*. In this game, when a player talks to a non-player character (NPC), it can either be for narrative progression or explanation or for ludic reasons (e.g. the NPC proposes a new quest). By studying the co-occurrence of a "new quest" panel with a NPC interaction, it is possible to elucidate the nature of the interaction. In Figure 93, the encounter at 600 seconds with a NPC is a narrative related interaction (as no "new quest" panel is detected), but at 900 seconds, the interaction is more ludic oriented (indicated by the presence of a "new quest" panel).

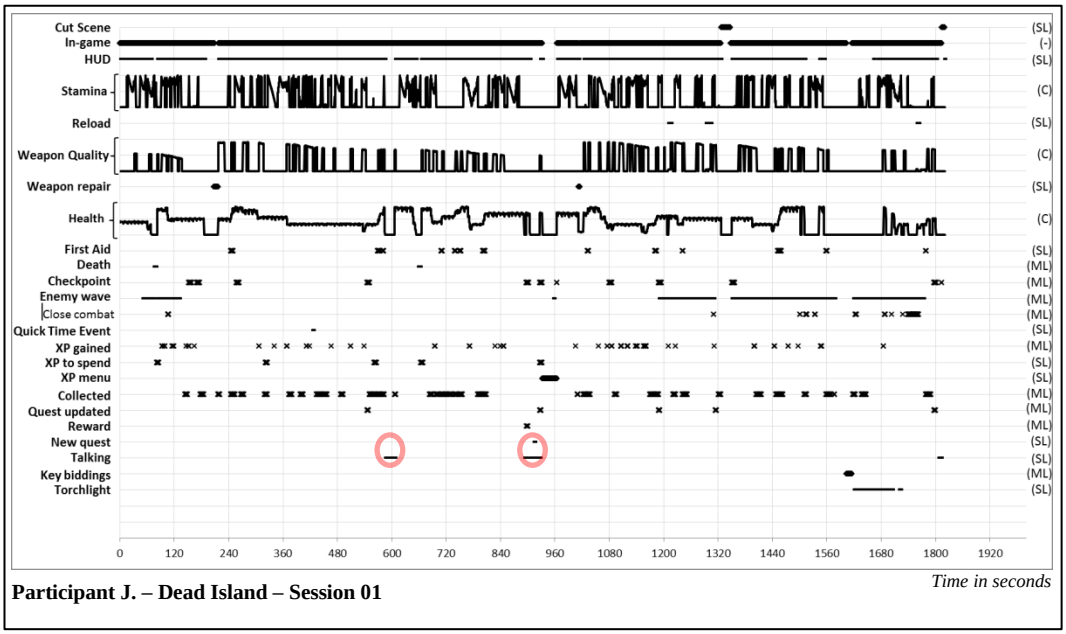


Figure 93. Concurrent metrics: talk discriminated via conjoint analysis of new quest panels (*Dead Island*)

Figure 93 is another example of "concurrent metrics", using the game Dead Island this time. While talking to a NPC, the player can be presented with a mainly story oriented speech (the NPC is speaking about their own life), or a mainly ludic oriented one (the NPC is proposing a new quest). To discriminate between both possibilities, the detection of a new quest panel, co-occurring with a talking detection, is indicative of the ludic orientation (around 900 seconds), while no quest panel is indicative of a story orientation (around 600 seconds).

Finally, Figure 94 shows how in *Battlefield 3*, a loading screen can appear during two different moments during play. The first when the player dies (loading during the respawn process), and the second when a mission is successfully accomplished (intra-missions loading). Similar to the *Bioshock 2* example, it is possible to discriminate the meaning of the loading screen by studying the co-occurrence of death. The appearance of the loading screen at 500, 360, 2200 and 3200 seconds are indicative of new missions, as no death is detected simultaneously. All the other appearances (960, 1320, 1440, 2700 and 2800 seconds) are indicative of the need to "repeat" a section of play.

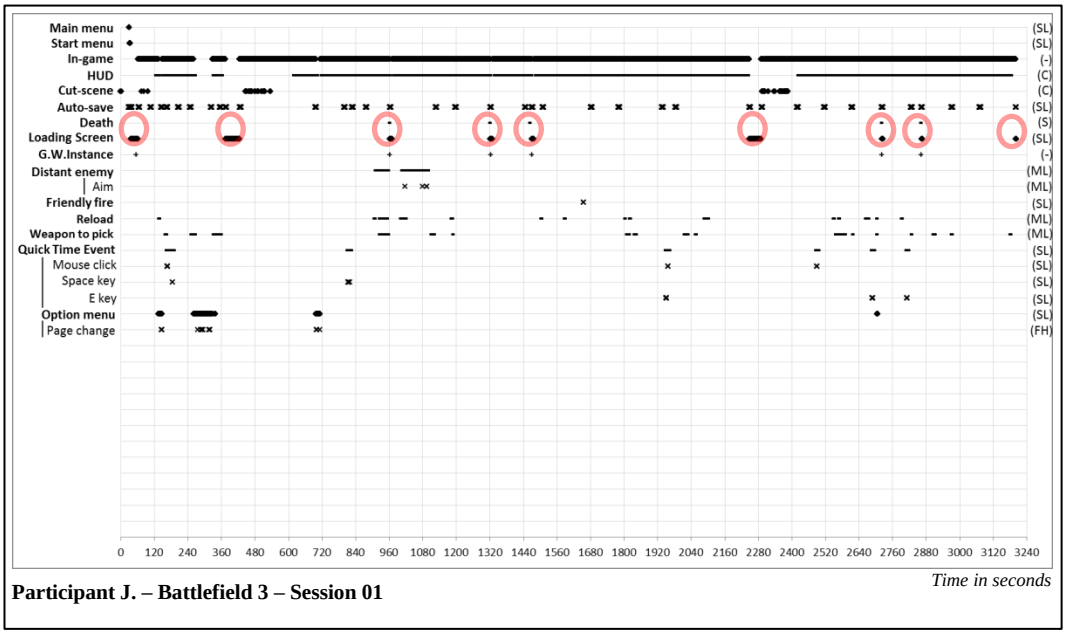


Figure 94. Concurrent metrics: loading screen discriminated via conjoint analysis of death (*Battlefield 3*)

Figure 94 example, using Battlefield 3, is similar to the Bioshock 2 one (Figure 92). The idea is to discriminate between loading screen indicative of the player moving forward into the game (inter-missions), and redoing the same sequence (game over) by studying the existence of a co-occurring death. The first, second and sixth annotations (from left to right) are indication of the player successfully moving forward (no death), while the other ones are indicative of failures (death).

Subsequent events

This section describes metrics that will retroactively justify and explain a sequence of interest. Figure 95 and Figure 96 deal with the use of “key bindings” menu by the player within two different games, *Dead Island* and *Max Payne 3* respectively. When a player enters a “key bindings” menu, it is either to change key mapping, or to learn the current controls. By observing the first action to be accomplished once the player exits a “key bindings” menu, it is possible to retroactively estimate why the player entered the menu in the first place. In *Dead Island*, the player never uses the flashlight during the full performance, except immediately after they exit the “key bindings” menu (see Figure 95). It then becomes obvious that the player has not used the flashlight before because they were unaware of which key is used to activate the flashlight. The reason for entering the “key bindings” menu is then clarified using subsequent metrics.

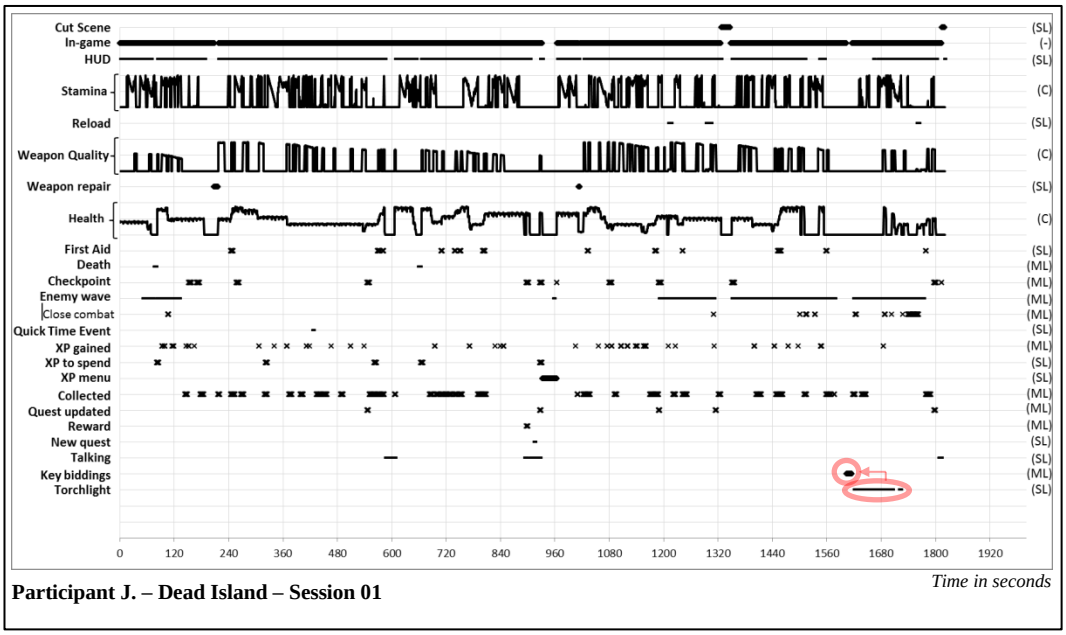


Figure 95. Subsequent metrics: use of torchlight explains the key biddings (*Dead Island)*

Figure 95 illustrated the use of "subsequent metrics" with the game Dead Island. In the summarized performance, the player never uses the torchlight, until they enter the key biddings menu. By analyzing the player first action once exiting the key biddings menu, it is possible to retrospectively understand why they enter the key biddings first. The player then entered the key biddings space to find a way to switch on the torchlight

Working on the same principle, Figure 96 illustrates the use of the weapon wheel in *Max Payne 3*. The participant does not change weapon until after they have been to the "keyboard" menu. This suggests that the player entered the "keyboard menu" in order to learn how to use the weapon wheel.

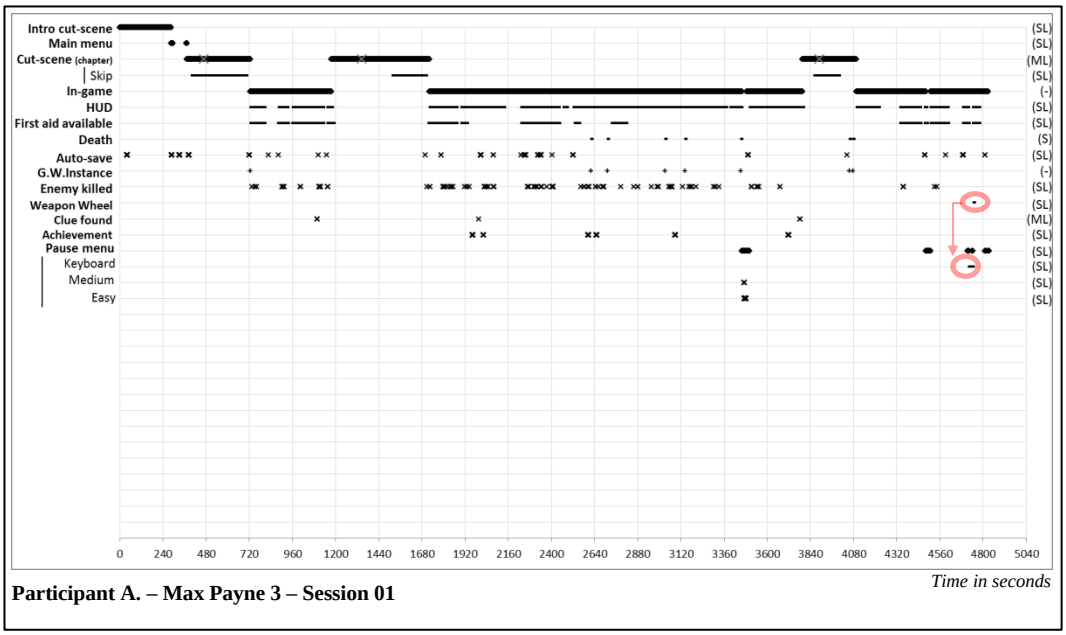


Figure 96. Subsequent metrics: the use of weapon wheel explains the keyboard menu (*Max Payne 3*)

Figure 96, based on the game Max Payne 3, is similar to the example of Figure 95, but about the possibility to change weapon through the use of weapon wheel. The player never used the weapon wheel until they enter the keyboard menu of Max Payne 3, then retrospectively justifying why the player previously entered the keyboard menu.

By considering the three temporal frames of reference, it becomes possible to interpret player action as presented above. The following section now focuses on a reconstruction based on semantic network, which describes how a set of metrics that are known to constantly influence one another actually evolves.

6.1.4 *Semantic Network*

A *semantic network* reconstruction deals with the identification of gameplay elements that are known to influence one another. The evolution of one is likely to directly impact the evolution of another. The role of semantic networks is to focus on intertwined elements in order to see how the player experiences their conjoint evolution, and how the player modifies their behaviour in response. The difference with the temporal frame of reference reconstruction is that semantic network is about identifying elements that are obviously connected in terms of gameplay purpose. For instance, fight and loss of health, can easily be seen as part of the same semantic network, as they constantly influence each other; whereas fight and help menu, while connected in Figure 90 (a difficult fighting session leads the player to go into the help menu), are less likely to be part of the same semantic network, as a fight does not necessarily mean the player will seek help inside the help menu.

In the following figures, the identified semantic networks are displayed over a red square background.

For *Bioshock 2*, a "fighting" semantic network can be identified (see Figure 97), and includes from the game *Bioshock 2* the HUD (degree of freedom allowing fight), enemy waves (degree of freedom indicative of a tougher fighting sequence), death (unfortunate fighting outcome), enemy hit (interaction representing a close combat where the player successfully hits an enemy), health and power (interaction representing the remaining energies of the player, in terms of life and weapon power). During the first session of Participant P. (top

summary), it is interesting to notice that the player uses all their power for the first time ever at 1000 seconds (the former zero point represent moment when the player is in the help menu, thus removing the power bar from the screen). The player tries to kill an opponent using power, fails, and then tries using a weapon this time. But more interesting is the graph summarizing the fourth session (bottom summary), demonstrating a potential matching between the player behaviour, and the game designer's intentions. Indeed, the player only dies during enemy waves. During these waves, the player is confronted with a lot of close combat ("enemy hit"), consequently the health/power levels are somewhat chaotic, illustrating a sequence in which the player is fighting for survival. But during a successful wave (3000 seconds), the health drops more incrementally, and no power is used, showing a more controlled weapon oriented fight (able to heal themselves in the middle of the fight). This would suggest that the player, from session 1 to 4, learned how to adapt their strategy, which is probably the expectation of the game designers when designing the enemy wave sequences.

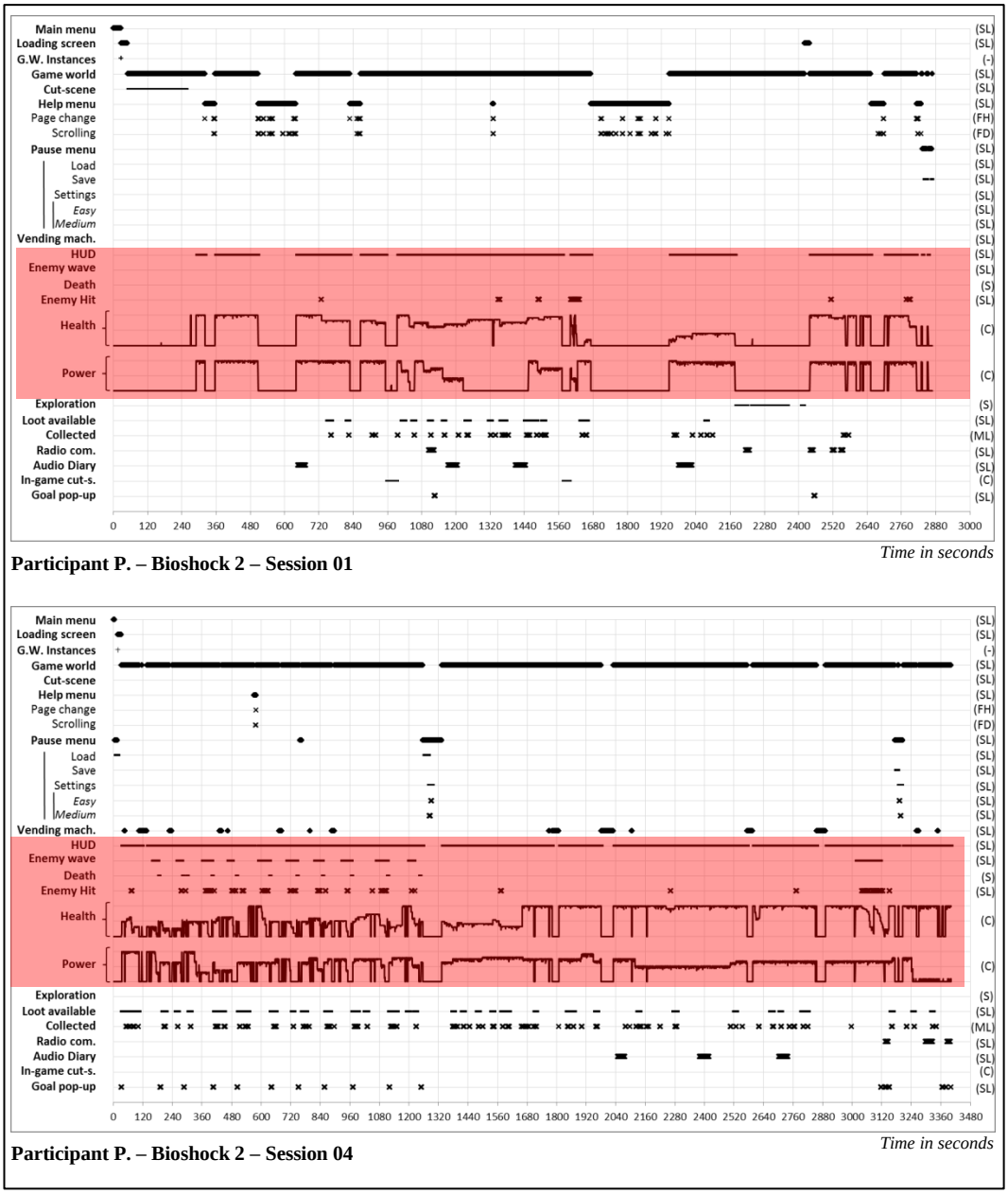


Figure 97. Semantic network: fighting (*Bioshock 2*)

Figure 97 illustrates semantic network with the game Bioshock 2. The highlighted gameplay elements are all part of the "fighting" network, as they all influence each another. The HUD indicates the possibility for a fight. The enemy wave indicates a strong fighting sequence. The death indicates an unfortunate fight ending. The enemy hit indicates a close combat. The health evolves regarding the player successful or unsuccessful fight, and the power can be used as a weapon.

Figure 98 illustrates *another semantic network*, which can be labelled "upgrading", and is about the gameplay mechanisms of collecting items, from loot or directly on the ground, in order to later spend them for upgrade, or into the vending machines in the case of money collection.

Figure 98. Semantic network: upgrading (*Bioshock 2*)

Figure 98 illustrates another semantic network in Bioshock 2, called "ugrading". The vending machine space usefulness is connected to the player having enough money to buy items, through loot and collection. The amount of available money can be seen as being part of the game system layer, and impact the behavior of the player (without money, the player is less likely to enter a vending machine space)

The “upgrading” semantic network can be seen as impacting the game system layer, as looted money in *Bioshock 2* becomes part of the player inventory, creating or preventing gameplay responses that require finances. During Participant P.’s first session (top summary), the player ignores the vending machines and only loots occasionally. The game system mechanism of buying items to facilitate survival and progression is not the player’s highest priority at this juncture of the game. However, by the fourth session (bottom summary), the collection process is a continuous and more routine action. Vending machines are often detected, indicative of a more strategic state of mind. This will be further developed in Section 6.1.6 about comparisons (notably in Figure 108).

Figure 99 presents another semantic network using the game *Dead Island* as an example. In this game, the player is invited to craft their own weapons, which quality depends on the used materials. The current qualities of weapons are highly tied to the detection of "weapon repair" spaces, as the player is likely to repair the weapon each time the quality is below a safety limit. Figure 99 illustrates a "weapon solidity" semantic network.

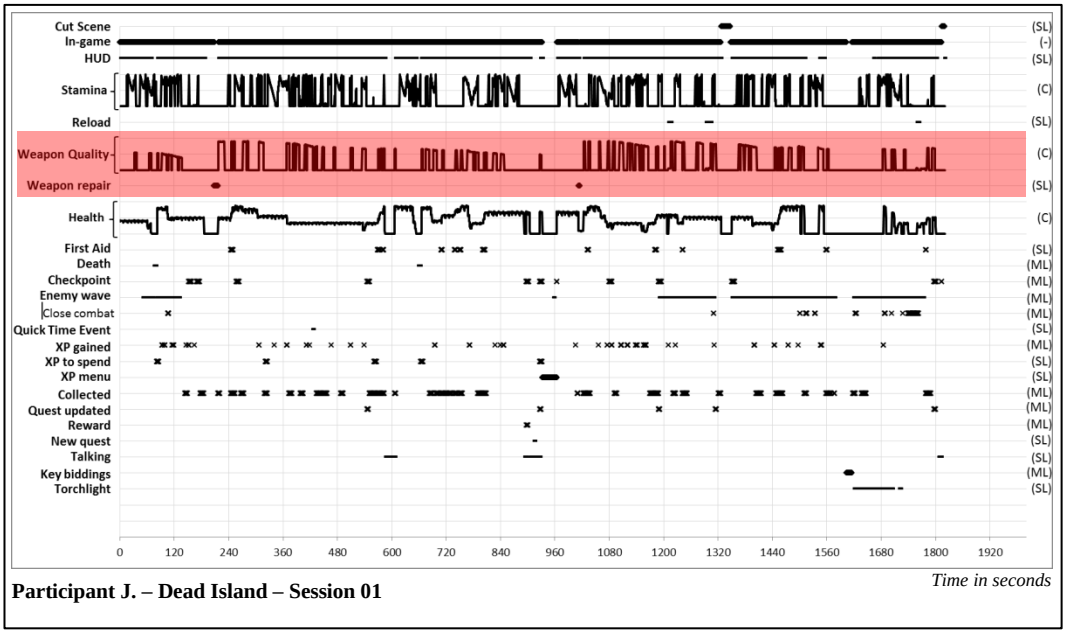


Figure 99. Semantic network: weapon solidity (*Dead Island*)

Figure 99 illustrates the use of semantic network in the game Dead Island. Here, the semantic network, called "weapon solidity" is linking the weapon quality with the weapon repair menu space. Indeed, the more likely the weapon is to break (quality goes down), the more likely the player is to be inside a weapon repair menu. The opposite is also true. Once the weapon repair menu is exited, the weapon quality is better, so the less likely another weapon repair space will be found in next future.

Figure 100 is also taken from the game *Dead Island*, and is related to a "fighting" semantic network (as Figure 97 with the game *Bioshock 2*). Low on health means imminent death, first aid means possibility to increase health, and stamina, connected to the player's ability to run, means that the player can, or cannot, escape a close combat. All these factors can influence the player's behaviour. For instance not having any first aid kit, and an empty stamina, could lead to a "risk it all" behaviour.

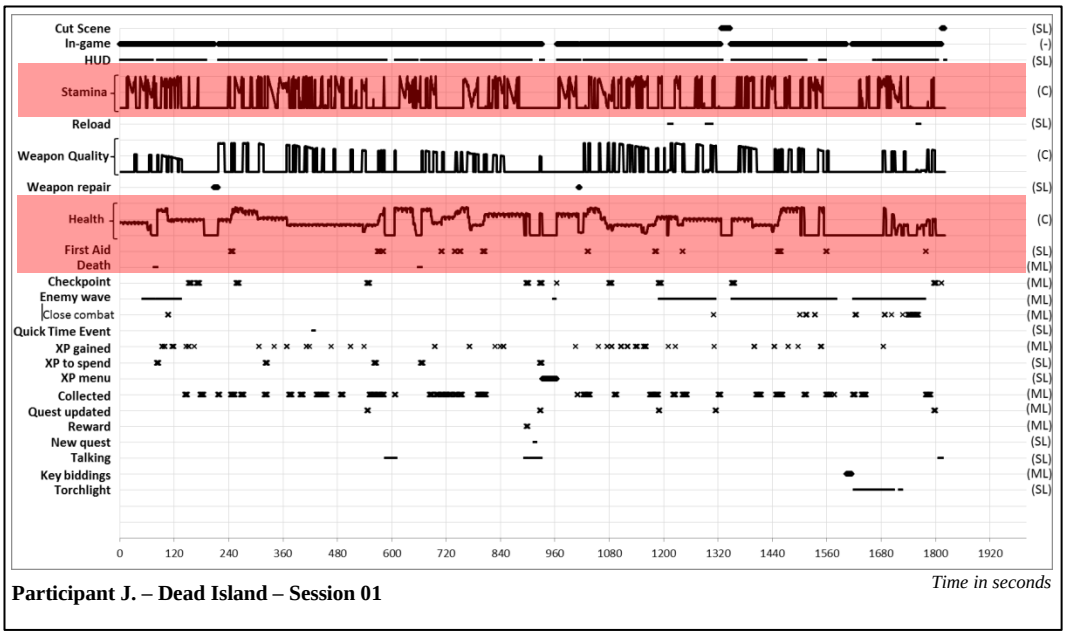


Figure 100. Semantic network: fighting (*Dead Island*)

Figure 100 is another example of "fighting" semantic network (as Figure 96), but this time with the game Dead Island. Death is an unfortunate outcome of a fight. First aid impacts the health by giving back some life. The health is indicative of successful or unsuccessful fight, and the stamina is indicative of how the player can engage with a fight. Without stamina, the player cannot run away, and is forced to fight. With stamina, the player can run away, and try to heal their avatar.

Remark: XP are mainly gained when an enemy is successfully killed, and enemy wave, close combat and Quick Time Event are indicative of other fight sequences. These elements could be considered as part to the semantic network too.

Figure 101 illustrates the "XP" semantic network in *Dead Island*, as already overviewed in Section 6.1.3 and Figure 91: the more XP the player gains, the more likely the "XP to spend" panel will appear, and the more likely the player will then go into the XP menu to spend the gained XP points (happening at around 960 seconds).

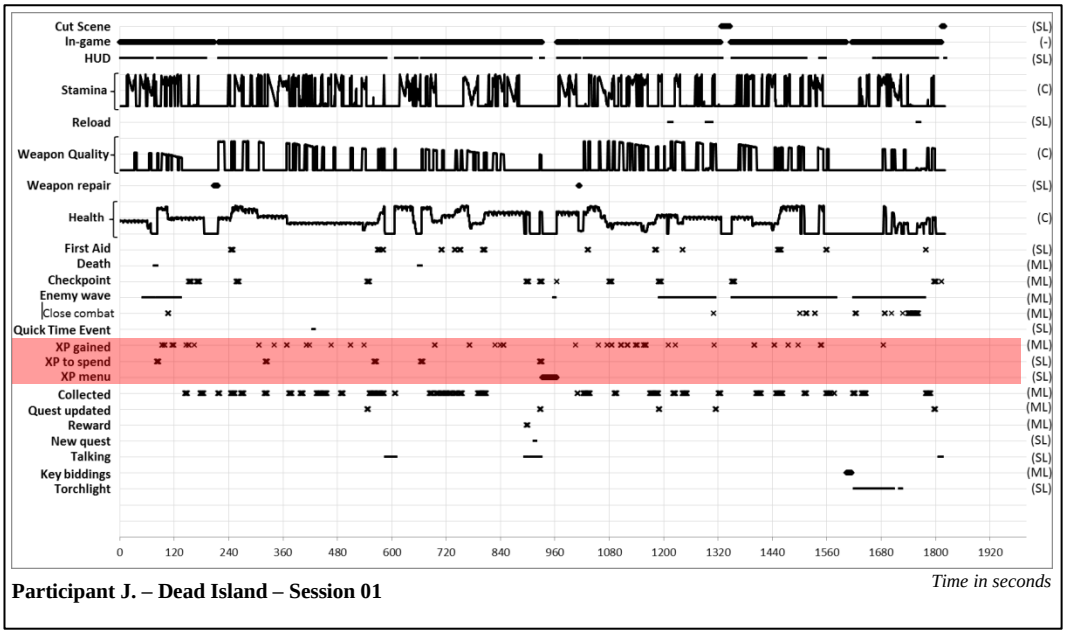


Figure 101. Semantic network: XP (*Dead Island*)

Figure 101 illustrates the "XP" semantic network with the game Dead Island, and can be connected to Figure 90 in terms of XP points usage. The player can gain XP by fighting, the panel warning the player that they have XP to spend only appears when the player has enough XP points, and the XP menu space is only useful when enough XP points are spendable. The number of XP points can be seen as part of the game system layer, and impacts, through the warning panels, the behavior of the player, constantly pushing them to go inside the XP menu space.

Figure 102 illustrated the semantic network reconstruction with *Battlefield 3*, about the "use of weapon", including: when the player is aiming at a distant enemy, when they unfortunately shot a teammate, when they can pick a new weapon, or when they are required to reload their weapon.

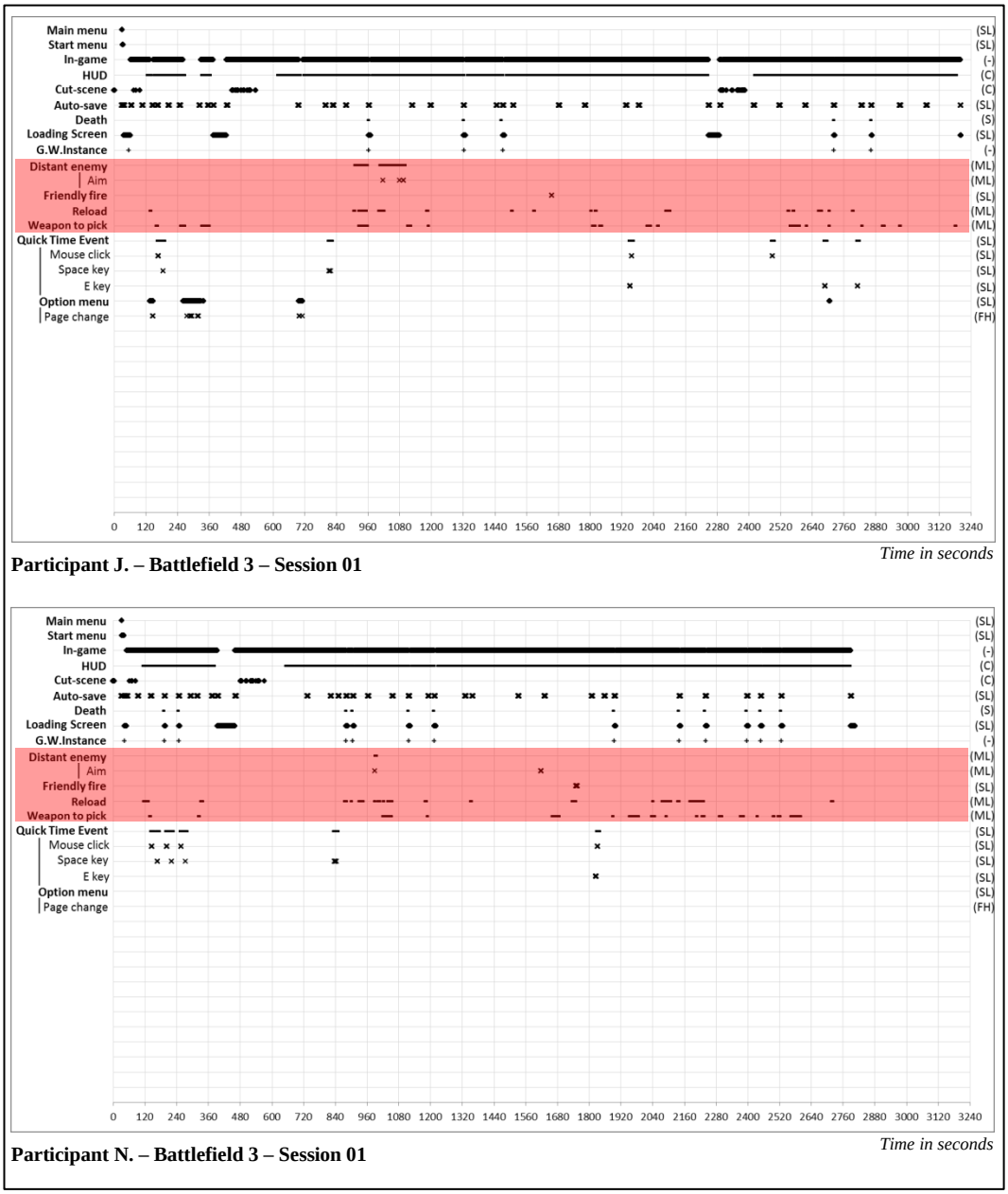


Figure 102. Semantic network: use of weapon (*Battlefield 3*)

Figure 102 illustrates the "use of weapon" semantic network in Battlefield 3. The different linked elements share a weapon common point. "Weapon to pick" represents a game warning that a weapon is near to the player, ready to be collected. "Reload" is a warning pushing the player to reload their weapon. "Friendly fire" is a bad use of the weapon. "Distant enemy" and "aim" are two metrics connected to the player using correctly their weapon, during specific "distant enemy" sequences.

Finally, Figure 103 illustrated semantic network in the game *Max Payne 3*. The game system proposes an interesting mechanism: the player, even if they lost all their health, will not die if they have a first aid kit available ("painkiller" in *Max Payne 3*). Instead, the player is presented with a "last man standing" sequence, in slow motion, allowing the player to kill the enemy that shot the last bullet. If successful, the painkiller is automatically used, and the player is resuscitated. "First aid available" and "Death" are then strongly interconnected, as death is unlikely when a first aid kit is available, but likely when no first aid kit is into the player's possession. The player is probably also less careful when they are aware that a painkiller can automatically trigger a "last man standing" sequence in case of heavy fire. This semantic network can be names "life-sustaining" network.

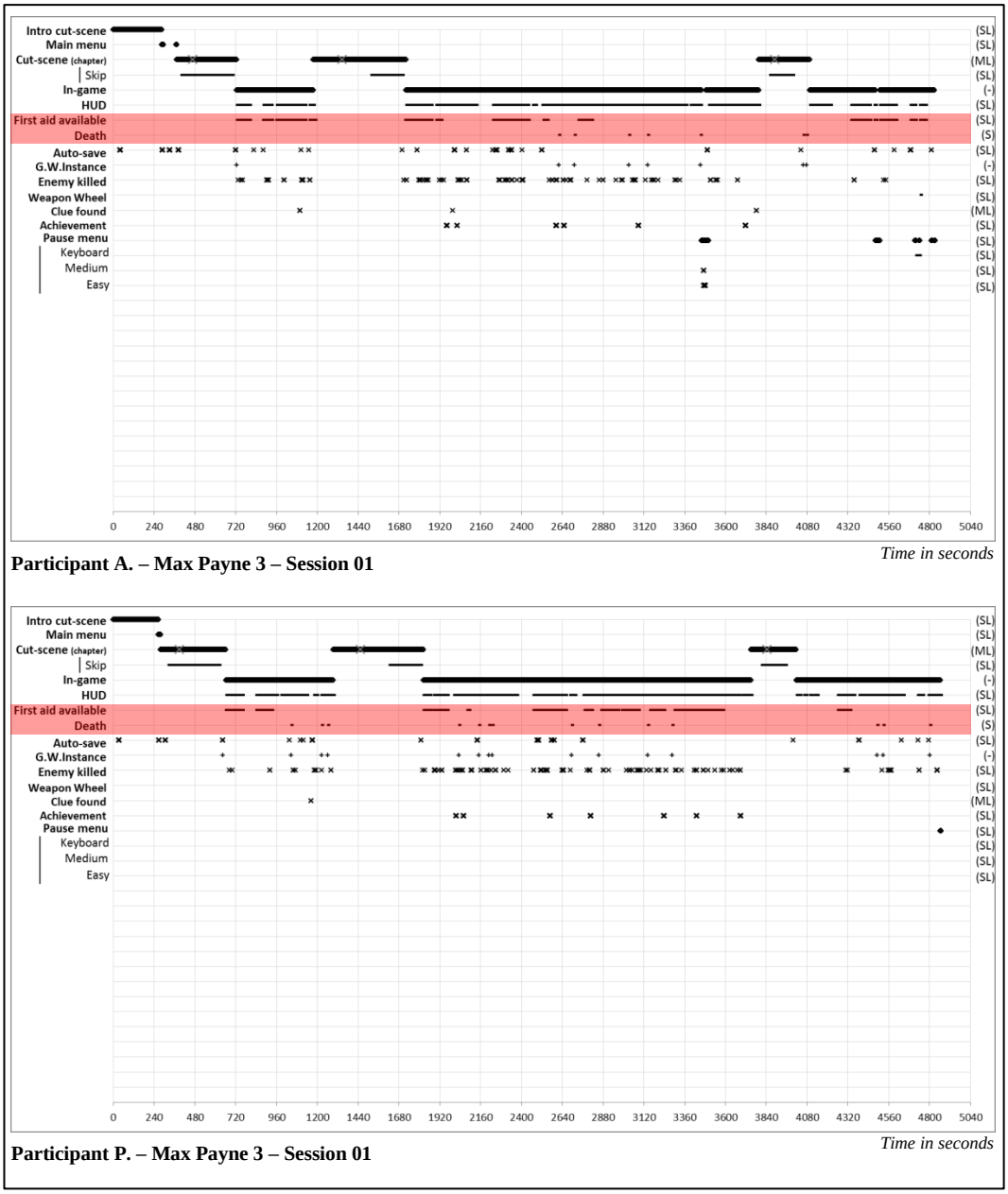


Figure 103. Semantic network: life-sustaining (*Max Payne 3*)

Figure 103 illustrates the "life-sustaining" semantic network in Max Payne 3. In Max Payne 3, as long as the player has a first aid kit with them (called painkiller in Max Payne 3), the game system will not let the player dies, or at least not directly. If a painkiller is available, the game will go slow motion, and the player can kill the opponent. If successful, the painkiller is used. That means that it is highly unlikely for a death to occur when a painkiller is available. Death and First aid available are then linked into the same semantic network.

Semantic networks are useful to identify and regroup metrics that belong to a same gameplay concept, so that the way each metric influences the others can be studied, and the performance can be reconstructed by considering their actual co-evolution. After the *semantic network*, the next transversal reconstruction mode is

about the detection of *loops*, which are similar consecutive sequences that can happen during a gameplay performance.

6.1.5 *Loop*

Section 6.1.5 illustrates the notion of *loop* as applied to a transversal reconstruction of gameplay performance. Loops, as described in Section 4.2.4, are consecutive sequences that are (nearly) identical, in general because the player needs to retry and improve an already encountered gameplay section. Figure 104, Figure 105, Figure 106 and Figure 107 are examples of the usage of loop as a transversal analysis of the games *Bioshock 2*, *Dead Island*, *Battlefield 3* and *Max Payne 3*. In the following examples, loops are always triggered by the death of the avatar, as death is the most encountered mechanism generating loops during the different sessions (death is always linked with retry). However, other interactions, like a loading action accomplished by the player are also accepted as triggering *loops*.

Different types of loop are presented in the current section: 1) loops without a clearly defined checkpoint (implicit), and not creating a new game world instance; 2) loops with a clearly defined checkpoint (explicit), and not creating a new game world instance; and 3) loops with a clearly defined checkpoint (explicit), and creating a new game world instance.

The annotations in the following figures are not straightforward, and should be explained beforehand: 1) a plain line represents a checkpoint; 2) a dashed line represents the loop trigger (deaths for instance), 3) an arrow represents the checkpoint a related to the loop trigger: and finally 4) each small circle represents the actual metric that justifies the line (checkpoint or loop trigger).

Loop one: implicit checkpoint, no new game world instance

The first game used to illustrate loop reconstruction is *Bioshock 2*, and this is represented in Figure 104. In *Bioshock 2*, every time the player dies, they are resuscitated into the nearest encountered regeneration chamber, called a Vita chamber. The respawn process is not a "load and retry" mechanism, as the avatar is regenerated while all the other characters in the game continue to go about their business. The implication of that is that a death does not generate a new game world instance in *Bioshock 2*. Death is part of the game story, and the player continues to evolve in a world where their avatar has already been killed, possibly even several times. In *Bioshock 2*, death should be seen more as a teleportation rather than game over. The most challenging part for detecting a loop in *Bioshock 2* is the absence of any clear clue to inform the player about their respawn point. The player can pass by a Vita Chamber unawares, and this vita chamber can still be the one used for respawn. In terms of experience, that means that the player can feel lost after a respawn and they need to orientate themselves. However, Figure 104 itself does display a possible loop analysis because the series of deaths occurring during the session all happen at the very beginning of the session. Furthermore, as the deaths are close to each other, it is reasonable to assume that they are all related to the same checkpoint, and it can be assumed this is the beginning of the session. In Figure 104, it is possible to see that the ten detected deaths trigger a similar loop, making the first minutes of the session a very repetitive and probably frustrating sequence. The repetition can be confirmed by analysing what happens in between two consecutive deaths (in this case between two dashed lines). Participant P. is using the same strategy of looting, collecting items, buying items in vending machines when possible, and then engaging with

the enemy wave. Between two consecutive dashed lines, the same pattern can be highlighted. The repetitive pattern probably arises from a lack of other strategic ideas, and the frustration is confirmed by observing the change in difficulty level at 1300 seconds (see also Figure 88). The loop transversal analysis is then a good solution to assess and visualize whether a player adapts their strategy, or not, when confronted to a similar challenge.

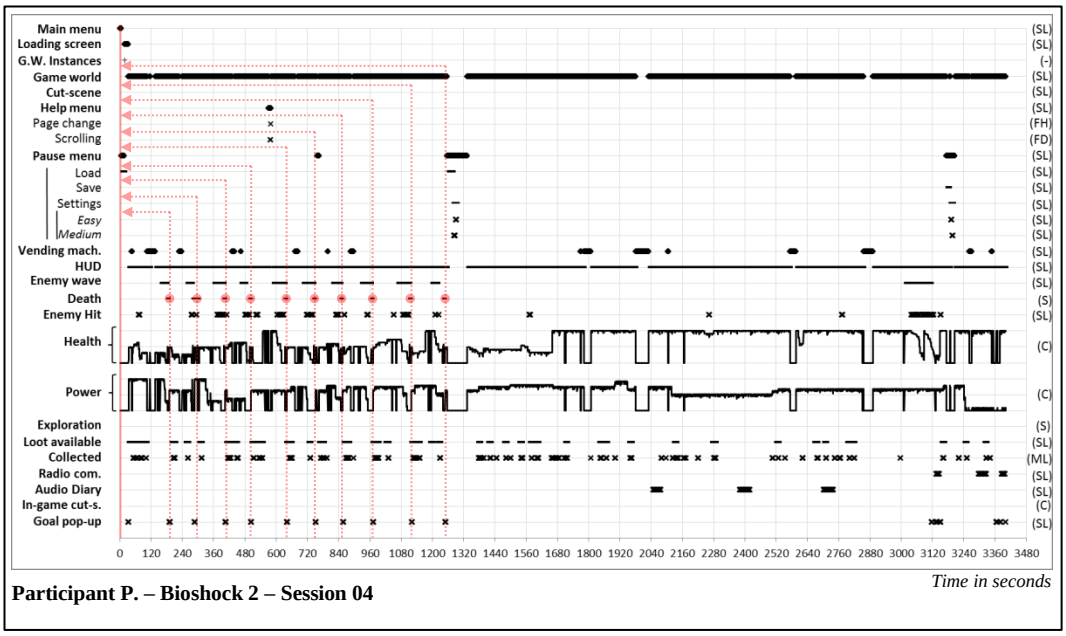


Figure 104. Loop: implicit checkpoint, no new game world instance (*Bioshock 2*)

Figure 104 illustrates the use of loop, with the game Bioshock 2. The loops in Bioshock 2 do not create a new game world instance (the player is teleported to the nearest encountered respawn point, called the vita chamber), and the checkpoint is not explicit (the player can walk passed a vita chamber without noticing it, while activating it for respawn). In terms of experience, that means that a player can be lost when they die, as they first have to understand where they are. In terms of reconstruction, an implicit checkpoint is also complex as it is difficult to really know where a loop should be connected. But the performance represented in Figure 104 is solves the issue as the deaths and generating loops, occur at the very beginning of the performance, therefore indicative that the loop entry point is probably the beginning of the session.

Loop two: explicit checkpoint, no new game world instance

The second example, based on Figure 105, represents a session from the game *Dead Island*. In *Dead Island*, the game mechanisms make the player aware of the exact respawn checkpoint, by displaying the word “checkpoint” on the left side of

the screen when these are encountered. In terms of experience, the player is clearly aware of the location where they will be teleported, meaning that the player can adapt their strategy accordingly. The displayed information, which can be extracted through the feedback-based gameplay metrics, allows for a better determination of *loops*, indicating the exact spatial point of respawn.

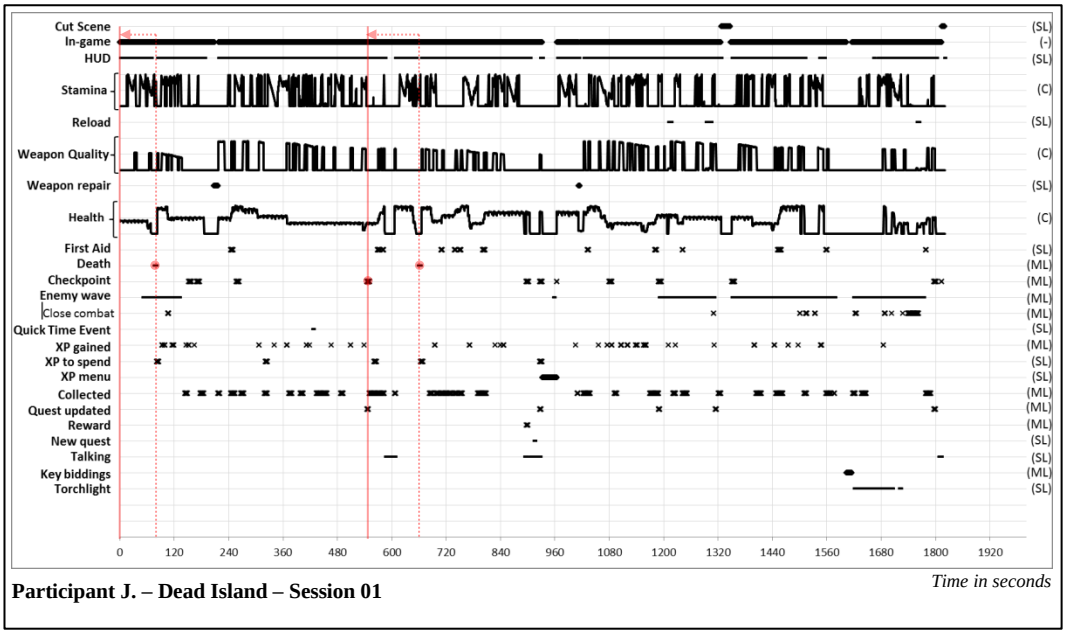


Figure 105. Loop: explicit checkpoint, no new game world instance (*Dead Island*)

Figure 105 illustrates, using the game Dead Island, another loop mechanism. As in Bioshock 2 (see Figure 104), dying does not create a new game world instance. The player is teleported to the last encountered checkpoint. However this time, the checkpoints are clearly indicated to the player in the form of a text message. The loops are easier to identify as the metrics are clear in terms of matching time stamps. The fact that no game world instance is recreated means that the player is presented with a "similar" sequence, but not identical (the already killed enemies are still dead for instance), explaining why the player in general only dies once.

However, as in *Bioshock 2*, dying does not indicate a reloading of the game, but only the avatar being teleported to the last checkpoint, and everything else is the same. This mechanism explains why loops have only one repetition, as the second attempt is far easier. So, *Dead Island* illustrates another way of managing loops: a continuous game world (no reload creating a new game world instance), but an explicit checkpoint. The *Bioshock 2* and *Dead Island* examples create imperfect loops, as each repetition is similar, but not identical.

Loop three: explicit checkpoint, new game world instance

Figure 106 and Figure 107 summarize sessions from the games *Battlefield 3* and *Max Payne 3* respectively, and in these cases a death stops the current progression of the player, and creates a reload of the game to the last encountered checkpoint. This causes a full rollback, both in space and time. All actions that have happened in between are erased and the game world instance is reset to a previous state. The combination of a new game world instance and an explicit checkpoint creates the loops that are the most identical, as the player restarts at the same point, and has to overcome exactly the same challenges. Furthermore, because of the repetitiveness of such *loops* (in *Max Payne 3* for instance, every intermediate cut-scene will be played again with each retry), the loops are a good indication of how a specific gameplay sequence may or may not be more likely to be remembered by a player.

In Figure 106, using the game *Battlefield 3*, it is interesting to analyse how the two participants (J. on top, N. on bottom) engage with the beginning of the game.

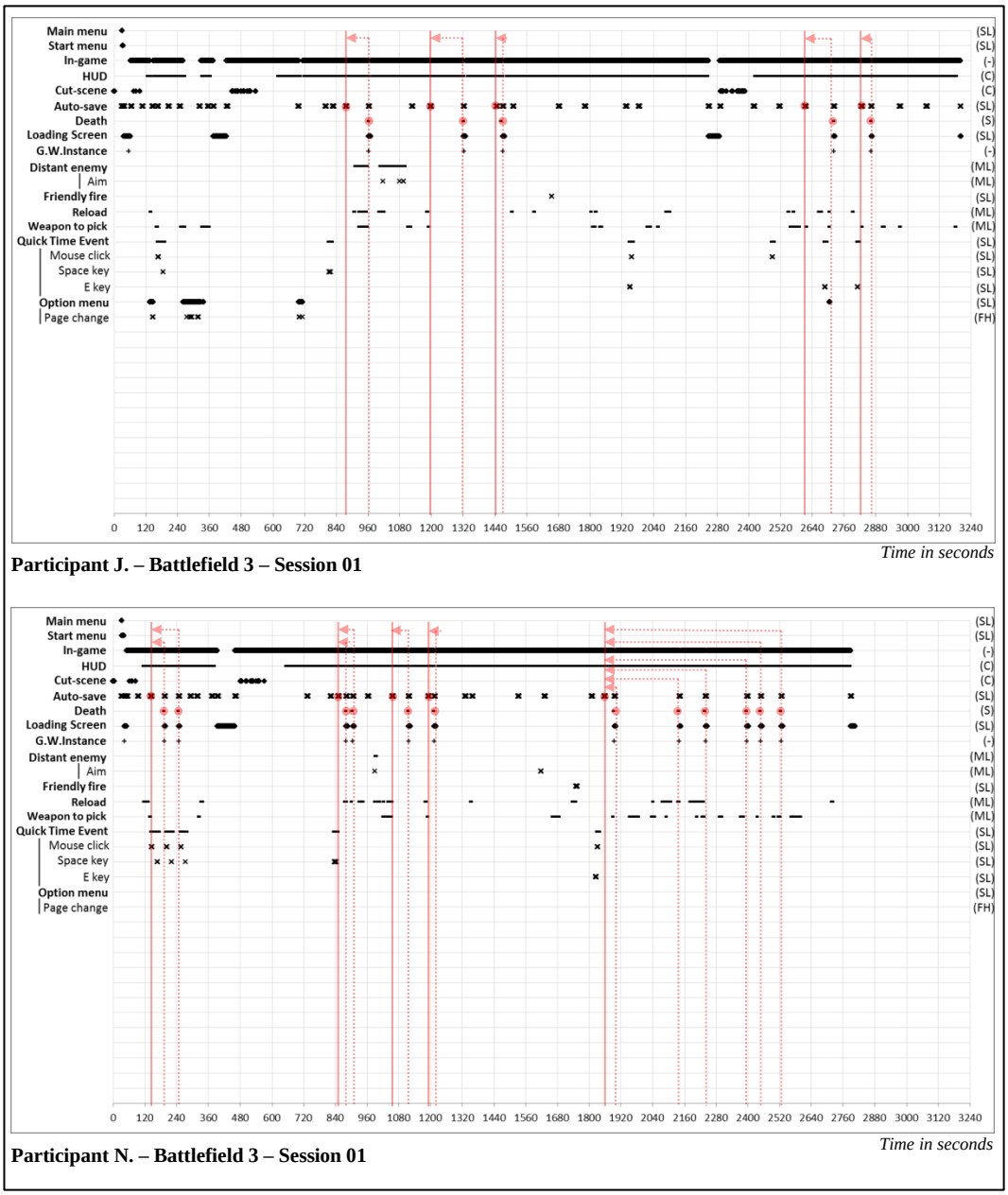


Figure 106. Loop: explicit checkpoint, new game world instance (*Battlefield 3*)

Figure 106 illustrates, using the game Battlefield 3, another type of loop. This time, the loops create new game instances, as the game is stopped, and the last "saving" point is used to recreate the game. This is a shift, not only in space, but also in time. The loops are then clearly more identical than the other types of loop. Participant J. (top summary) dies regularly, but only redoes the sequence once, probably surprised by a death, but is able to quickly find a successful strategy. Participant N. (bottom summary) has more trouble to overcome the situation and has to deal with more repetitions. Analysis of the loops between 1850 and 2520 seconds shows, however, that the player is adapting his/her strategy, as no "reload" warnings are detected in the latest loops, showing a better ammunition management.

When Participant J. dies, the situation is overcome the following time. Participant N., on the other hand, seems to struggle more, especially during the series of deaths between 1850 seconds and 2520 seconds (six deaths). However, the analysis of each loop is informative about Participant N.'s adaptation of behaviour. At 2160 seconds and 2200 seconds, Participant N. runs out of ammunition (the game is asking them to reload, as detected by the "reload" metric), while during the following deaths, the player tries to have enough bullets available. This indicates that the player changes their strategy and develops a solution

Another comment can be made concerning Figure 106, by comparing both Participant J. and Participant N. The sequence between 1920 seconds and 2280 seconds for Participant J., and the one between 1800 seconds and 2760 seconds for Participant N. are the same. This can be assessed by the detection of the very same Quick Time Event pattern at 1920 seconds (Participant J.) and 1800 seconds (Participant N.), and the detection of the following cut-scene at 2280 seconds (Participant J.) and 2760 seconds (Participant N.) (the use of QTE for comparing sequences will be more thoroughly discussed in Section 6.1.6 about comparisons). Participant J. does not die once, while Participant N. dies six times, meaning that Participant N. encountered the same sequence seven times. In terms of remembering the game, Participant N. will probably remember this sequence more than Participant J., who just passes through it without any trouble.

Max Payne 3 works with a similar "load" and "retry" function. In terms of comparison, a quick look over the two sessions in Figure 107 is enough to realise

that Participant P. (bottom summary) is having more trouble than Participant A. (top summary).

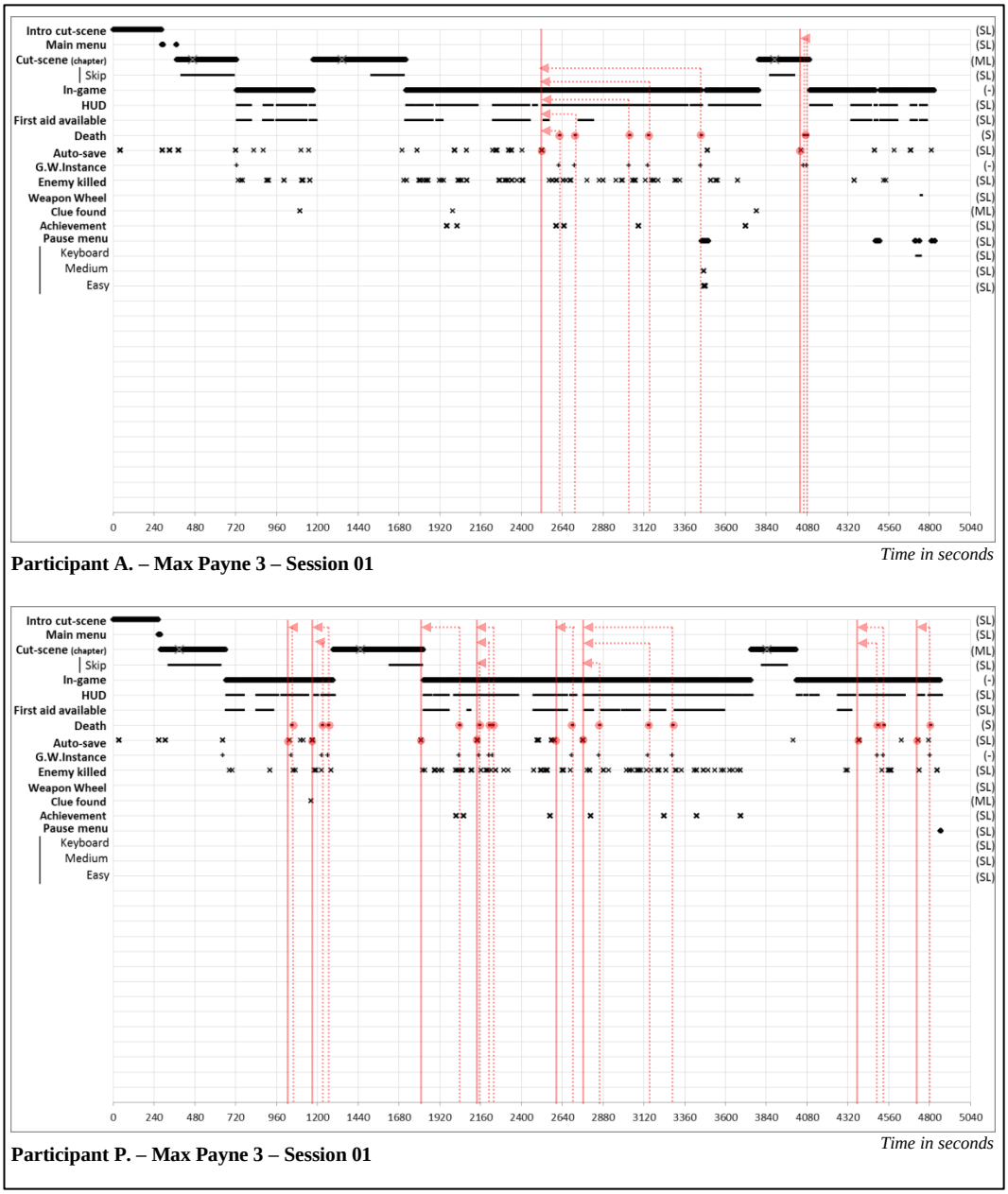


Figure 107. Loop: explicit checkpoint, new game world instance (*Max Payne 3*)

Figure 107, based on Max Payne 3, is similar to the one in Figure 106. But the comparison is different. Participant A. (top summary) only encounters one difficult sequence but remains stuck for a long time (five loops), while Participant P. (bottom summary) is having more trouble throughout the performance, but has in general a maximum of two or three consecutive loops. An interesting comment can be made about Max Payne 3. Although a new game world instance is created, some elements, like achievement, are remembered in the game system layer as correctly achieved. So the shift in "time and space" is only for the game world, not for the entire game system.

However, while encountering the last section of chapter 2 (in *Max Payne 3*, a level is a called “chapter”), at around 2500 seconds, Participant A. dies five time, against only three times for Participant P. (around 2700 seconds). Furthermore, to overcome the situation, Participant A. eventually changes the level of difficulty (at around 3500 seconds, see also Figure 89). This shows that, in terms of an engagement with the game, Participant P. is less careful, and dies quite regularly, but can find a solution after repetitive sequences; while Participant A., although more careful, is unable to change strategy when a challenge forces them to redo the same sequence. Finally, an important transversal comment should be made about *Max Payne 3*. Indeed, the “load” and “retry” mechanism seems to erase everything that happened in between. However, at the game system level, some elements are still preserved, including “achievements” that once unlocked, remain unlocked.

Loops are meaningful to detect, as they help in studying the ways participants are modifying, or not, their behaviours when presented with similar contents over the time of a performance; and also comparing several player engagements with similar contents. About comparisons, the final section is about transversal comparison, meaning the possibility to compare two sessions of the same player, or different players, and using the same game, or different games. The comparison approach has actually been used several times so far, as understanding player experience always profits from comparing different sessions.

6.1.6 *Comparisons*

In previous sections, the different transversal reconstructions mainly occurred intra-sessionally, meaning that a gameplay element is related to another one in the same session (same player, same game, and same session). It is however important, in order to better understand player experience, to also be able to compare different sessions together, in a cross-transversal way. Cross-transversal means that, unlike the previous transversal modes, a gameplay element is compared beyond the current session, crossing the session boundaries in order to be linked with another gameplay element in another session. Three cross-transversal modes can be highlighted: 1) *between-sessions* (same player, two different sessions), emphasizing a player evolution while engaging with the same game at two different moments; 2) *between-players* (same game, two different players), emphasizing two different ways of engaging with a same game regarding player styles; 3) *between-games* (same player, different games), emphasizing how a same player can react differently in regard to different games mechanisms, or genre.

In the following figures, unlike previously, the annotations go beyond the boundaries of a session, and should be interpreted as follows: 1) a group of metrics of interest is represented over a red square background, and 2) the related groups of metrics, in two different sessions, are connected with a double arrow.

Between-sessions

Figure 108 illustrates how cross-transversal comparisons can be made using the same game, the same participant, but with two different sessions of play. Figure 108 is based on the game *Bioshock 2*, with participant P., using the first session (0 to 45 minutes of play, top summary) and the fourth one (135 to 180 minutes of play, bottom summary), in order to highlight the change in play experience that can occur when the player knowledge of the game mechanisms increases. Figure 108 shows how participant P. shifts their play style, when confronted with difficult sequences and changes to a much more strategic style of play.

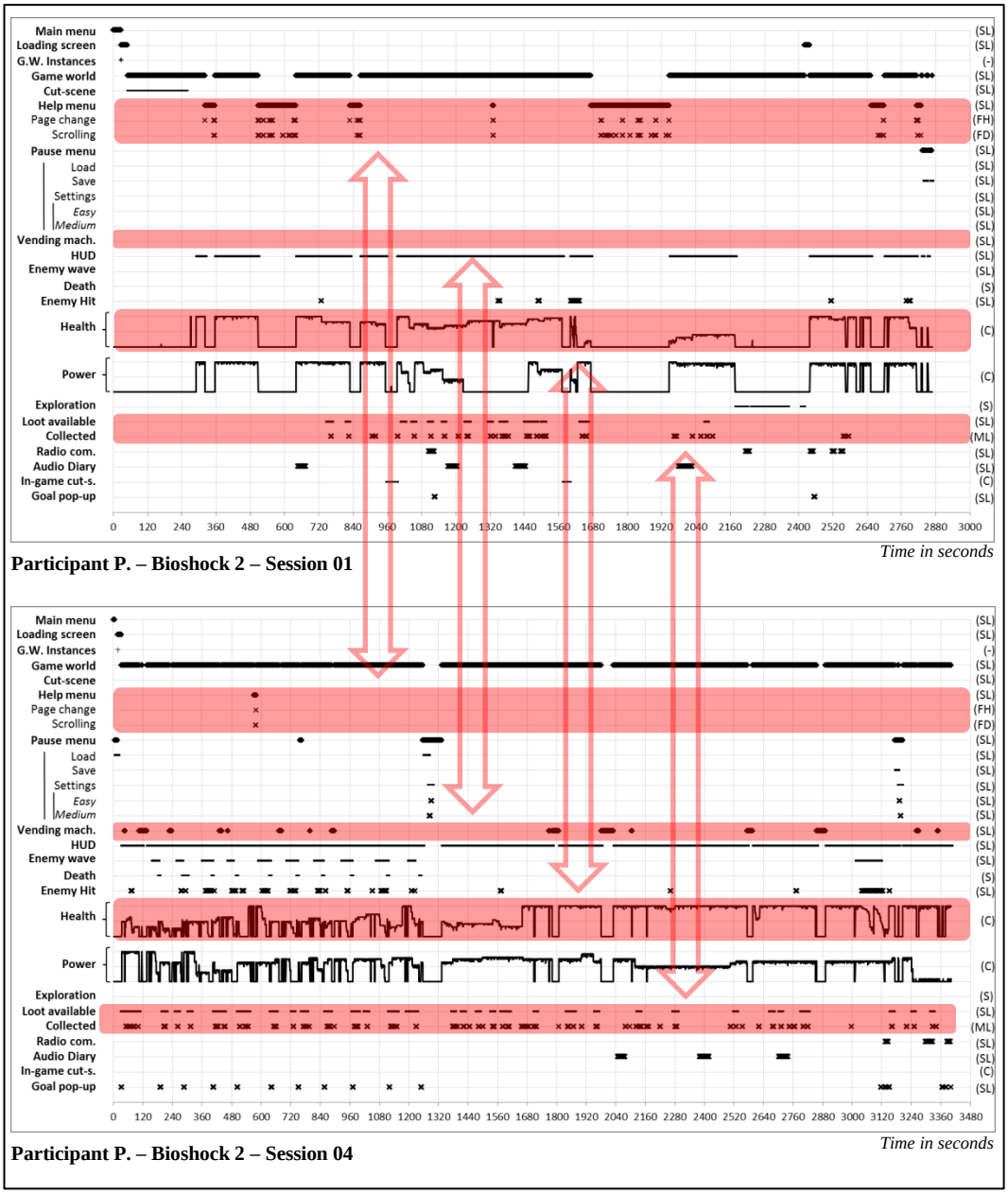


Figure 108. Between-sessions comparison: from help to upgrade (*Bioshock 2*)

Figure 108 illustrates the between-sessions comparison (same player, same game, different sessions), using the game Bioshock 2. In this example, the shift from a more "rules understanding" strategy (top summary, showing how the player uses mainly the help menu after difficult sequences) to a more "loot and upgrade" strategy (bottom summary, showing how the vending machines, ignored during Session 1 (top summary), are more thoroughly used during Session 4, and how the help menu is not used as often).

During the first session, when participant P. encounters a challenging sequence (loss of health detected around 1600 seconds), their next step was to go inside the help menu, to understand the game mechanisms (as already presented on Figure 90). However, after the first fighting sequence (around 720 seconds, see also Figure 82), participant P. started to loot extensively, which means that they could also have decided to use the result of their looting to buy upgrades and becomes stronger in order to prepare themselves for eventual following challenging sequences. But no “vending machine” space sequences have been detected during the first session. Participant P. did not use any of the looting collection during the first 45 minutes of play. On the other hand, during the fourth session, almost no time has been spent inside the help menu (only several seconds around 600 seconds). But numerous “vending machine” space sequences are detected. After several deaths (also highlighted on Figure 104), participant P. decides to go for a looting sequence (starting around 1320 seconds) and uses the vending machine several time. At around 3000 seconds, participant P. engages another time with a challenging sequence, and is able to cure themselves during the fight (as highlighted by health raises after sudden drops), meaning that the participant prepared themselves for the fight. This time, unlike the first session, the player uses the looting results for strategic purposes (i.e., in order to preparing himself to the fight). This can be understood as a shift from a beginner engagement with the game, assuming that having difficulties to overcome a sequence may come from an insufficient knowledge of the game world; to a more skilled engagement, making the player aware that they must find a strategic solution to the problem, as their knowledge of the game world is now sufficient.

Between-players

The second cross-transversal comparison is between players, playing the same game. The *comparison between-players* can improve player experience understanding on two main levels: temporal, by highlighting when the players cross the same checkpoint; and experiential, by illustrating how the players can engage differently, or similarly, with the provided game mechanisms. Figure 109 and Figure 110 illustrate the temporal comparison, while Figure 112, Figure 111 and Figure 113 illustrate the experiential comparison.

In *Battlefield 3*, several quick time event (QTE) sequences are encountered by the players. The QTE sequences appear in the same order to all the players, and cannot be avoided. Furthermore, the keys the players are supposed to press in order to succeed the QTE are different from one QTE to another. That means that, by identifying the displayed key information during QTE sequences, it is possible to also identify similar moments in different sessions. On Figure 109, participant J. encounters five different QTE sequences: mouse click + space (1), space (2), mouse click + E (3), mouse click (4), E (5). Participant N. only encounters the first three QTE sequences, so it is possible to conclude that Participant N. went less far into the game in comparison to participant J. But more interesting, it is possible to identify the exact moments when each participant encounters the same content by studying the QTE patterns. The first QTE sequence appears three times for participant N., indicative of a loop, which is actually in phase with the two deaths happening almost at the same time.

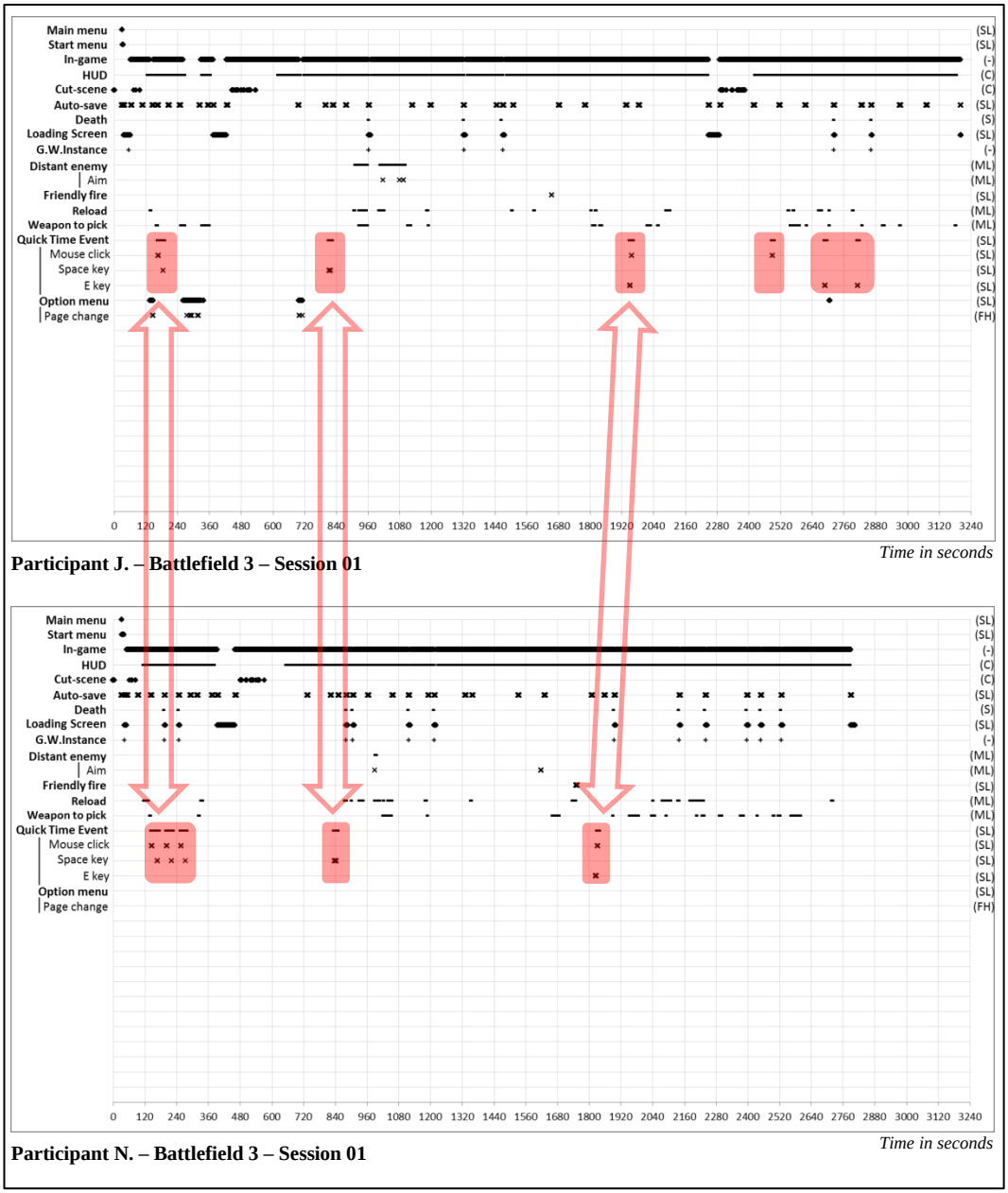


Figure 109. Between-player comparison: similar Quick Time Event sequences (*Battlefield 3*)

Figure 109 illustrates a between-players comparison (same game, same session, but different players), based on the game Battlefield 3. Here, the pattern of the Quick Time Events is a cue to identify similar sequences in between the sessions. The interest is double; first, to identify identical moments, and second to identify different ways of behaving during these moments. Participant N. for instance (bottom summary), unlike Participant J. (top summary), repeats three times the first QTE, indicative of a more unsuccessful way of dealing with the QTE mechanism.

Figure 110 is more straightforward, as it illustrates that the detection of the word "CHAPTER" on the screen in *Max Payne 3*, appearing each time a new chapter begins, can also be used to identify similar key moments in different sessions. Participants A. and P. seem to play the game at a very similar pace, as they encounter the same key moments at roughly the same time-stamp, even if Participant P. (bottom summary) dies more often (see Figure 107).

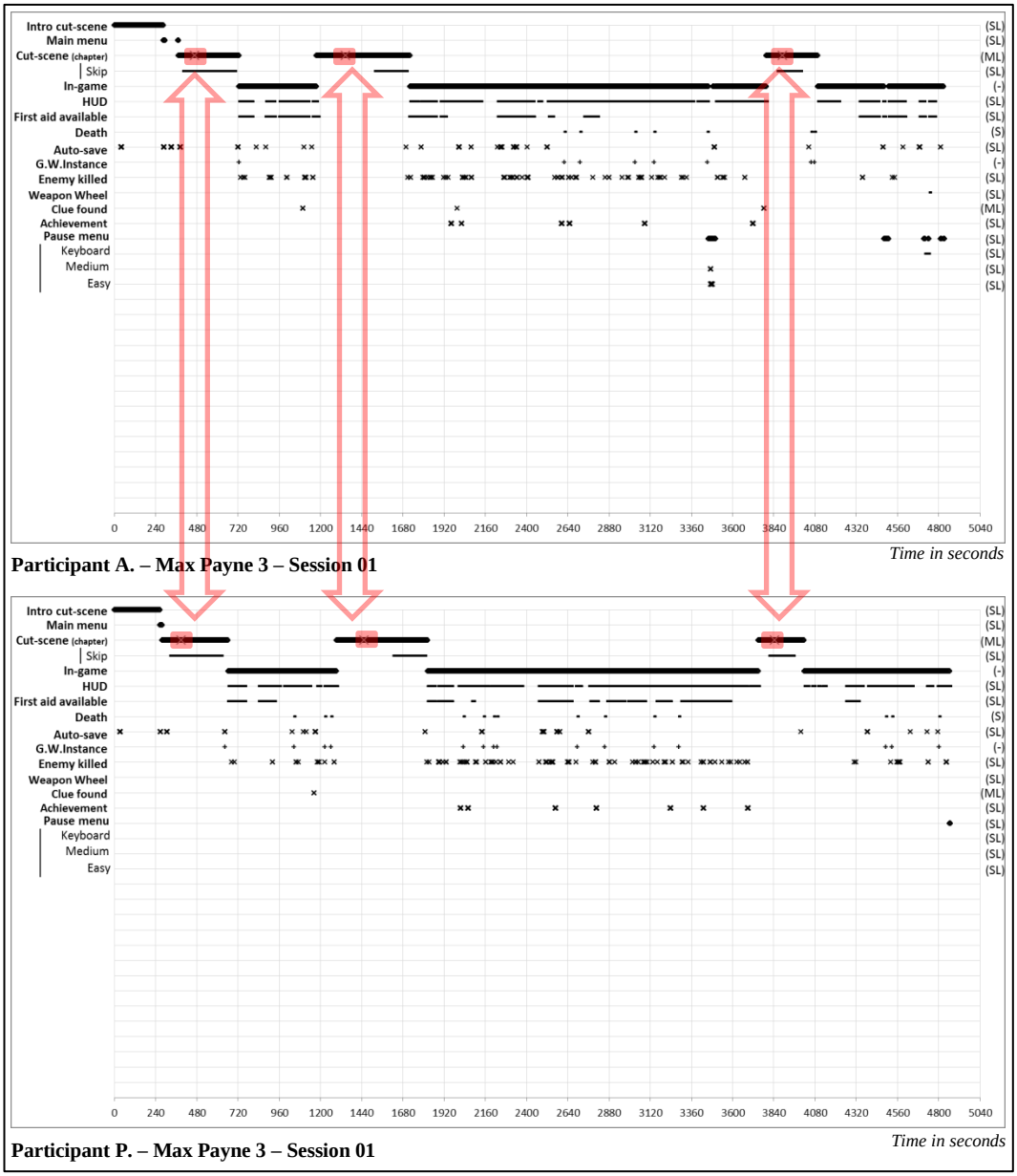


Figure 110. Between-players comparison: similar chapter beginnings (*Max Payne 3*)

Figure 110, based on Max Payne 3, is similar to the one presented in Figure 109. It is possible to detect the word "CHAPTER" displayed on the screen, indicative of the beginning of a new level. Although Participant P. dies more often, both participants evolve through the game at a similar pace, as the different words "CHAPTER" are detected at similar time stamps.

Figure 111 illustrates two similar ways of playing the first session of *Max Payne 3*. Both participants did not interact with the weapon wheel (except participant A. at the end of the session), and so the question can be asked if it is because both of them like playing with the last collected weapon, or if there is another reason that needs to be considered. However, as already illustrated by Figure 96, participant A. interacts with the weapon wheel for the first time just after being in the "keyboard menu". The comparison between the two participants can here be used for design improvement purpose, because it seems that none of the participants have been aware of the existing mechanism to switch between weapons, even if participant A., probably from previous experience with similar games, takes the conscious decision of browsing the keyboard menu, in order to look for such a possibility.

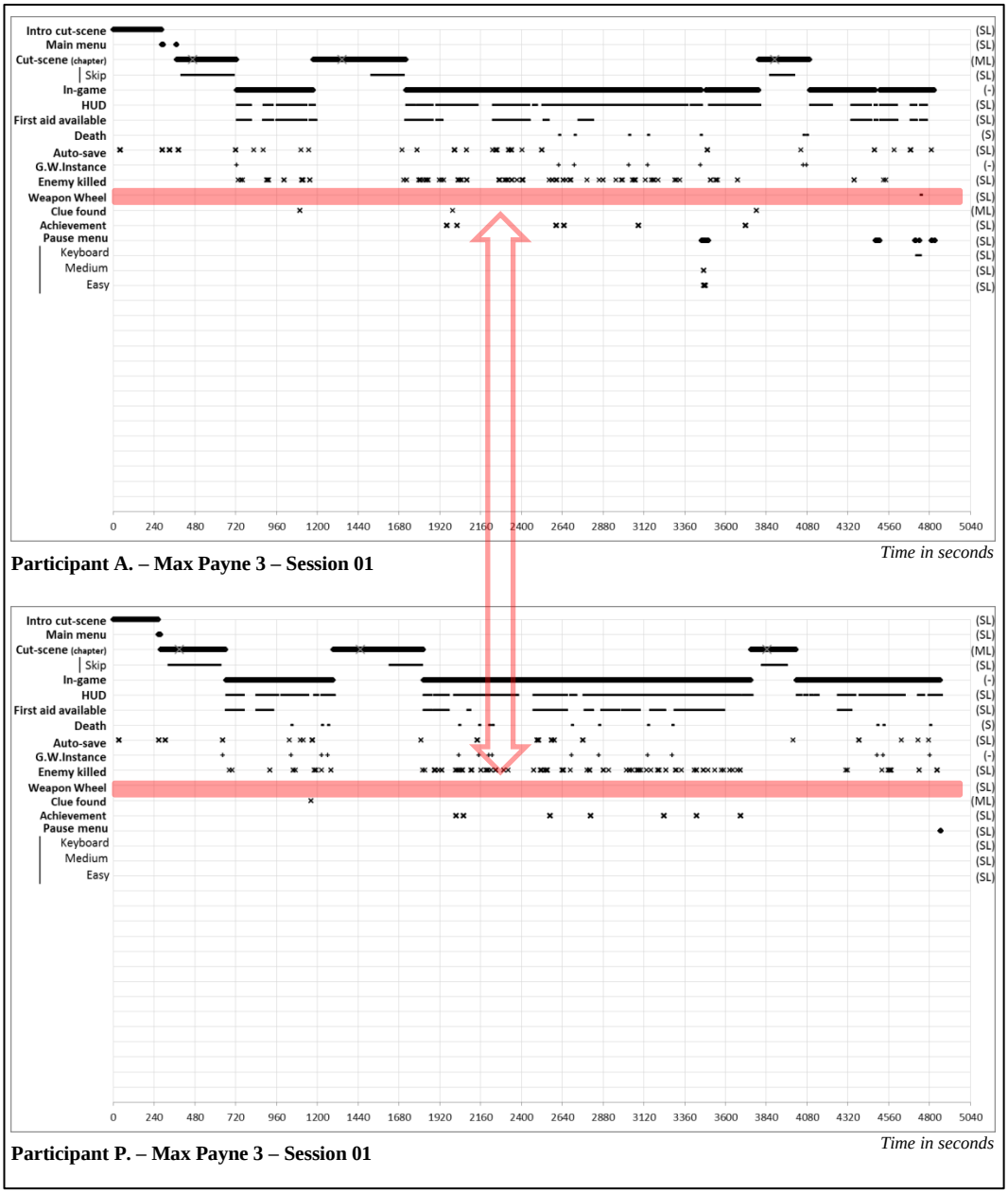


Figure 111. Between-players comparison: use of weapon wheel (*Max Payne 3*)

Figure 111 illustrates another between-players comparison, using the game Max Payne 3, but this time to emphasize a similar behavior. Both of them decide not to use the weapon wheel, except Participant A. (top summary), at the very end of the performance, after entering the keyboard menu (see Figure 96). This comparison informs about the game design, as the avoidance of the weapon wheel option is likely to be related to the players not even knowing about this gameplay mechanism.

Figure 112 shows two different ways of engaging with the game *Battlefield 3*. Participant N. never uses the option menu, and plays with the default settings proposed by the game, while participant J., as soon as the game starts, goes straight into the option menu. Participant J. seems to be a player that likes to customise the game in order to suit his/her own way of playing.

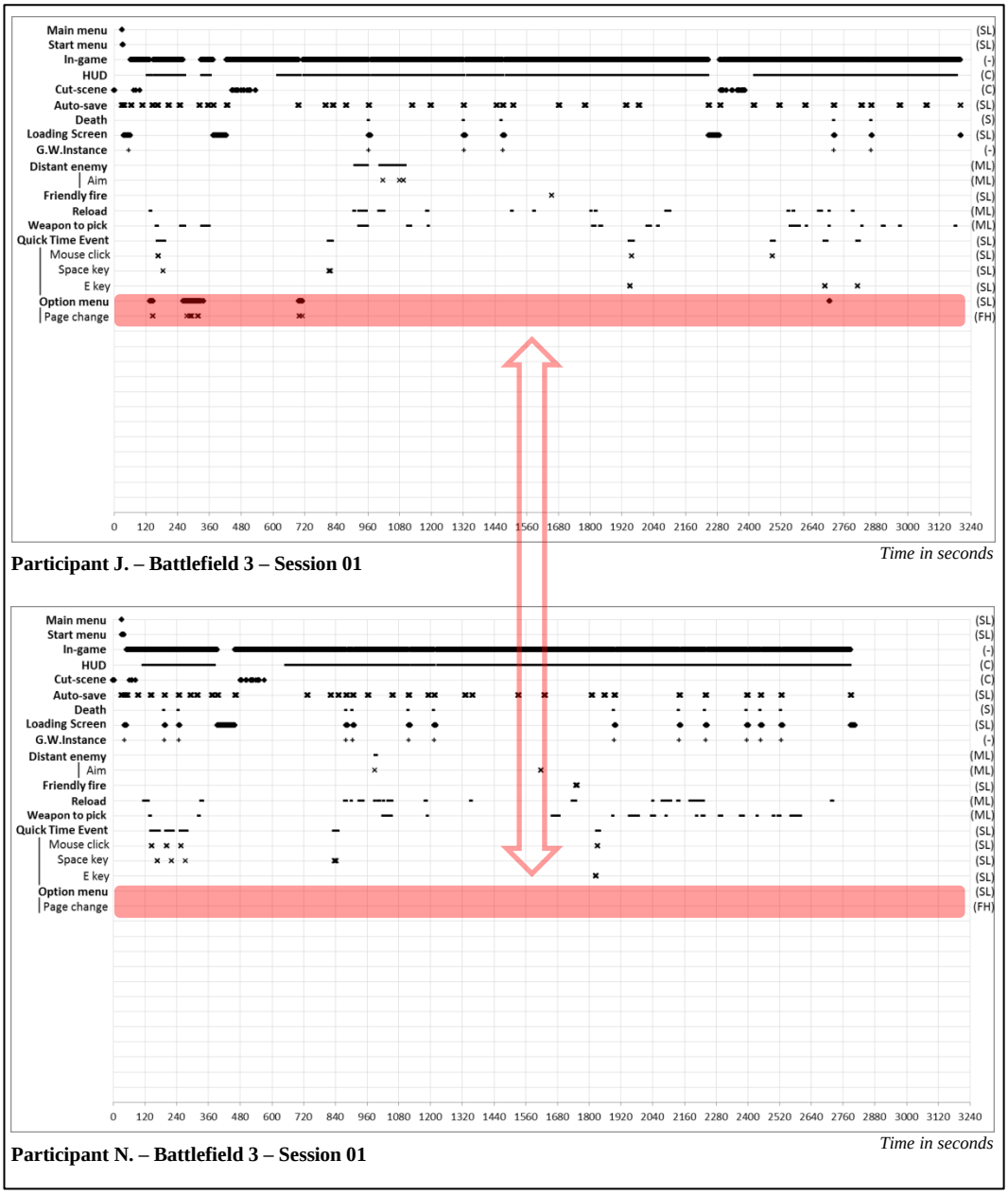


Figure 112. Between-players comparison: use of option menu (*Battlefield 3*)

Figure 112 illustrates, using the game Battlefield 3, two different ways of engaging with the default settings of a game. Participant N. (bottom summary) never goes to the option menu, showing that they accept the default settings, while Participant A. (top summary) is seen into the option menu at the performance beginning, showing their need to adapt the game to their own style of play.

Finally, Figure 113 illustrates two different ways of engaging with the story of *Max Payne 3*. In *Max Payne 3*, the main character protects a girl without knowing much about her past, and the reason for her protection. However, several clues at different places in the game world inform the player about the girl. Except for the first clue that is mandatory to find, because it is part of the tutorial; the players can easily miss the other clues if they are not paying attention. Participant A. found three clues in total during the first session, while participant P. only found one, the tutorial one. Participant A. then shows a greater interest in unfolding the story, and exploiting the game environment, while participant P. does not show a particular interest in looking for these clues and is probably not engaging as much with the story of the game.

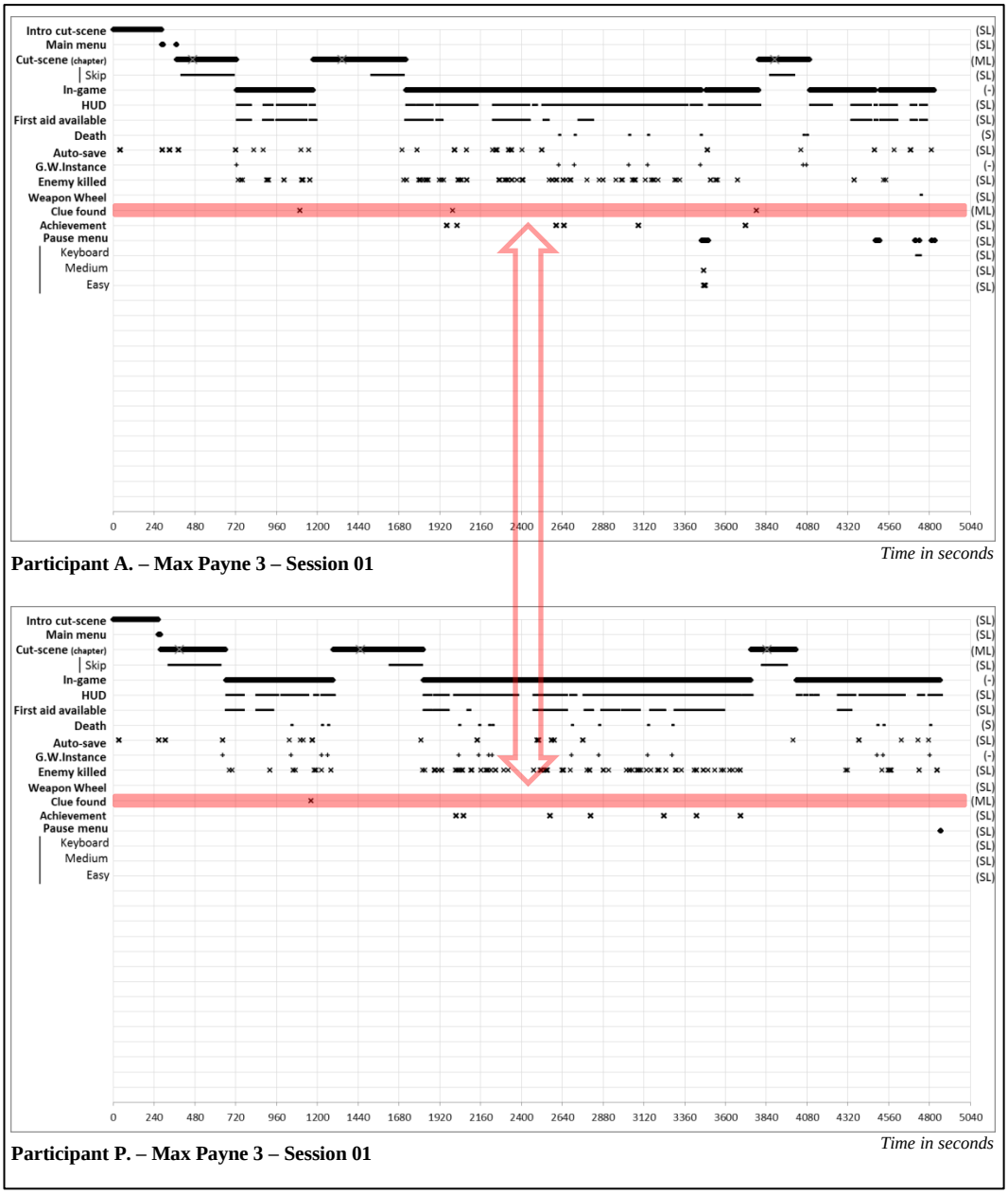


Figure 113. Between-players comparison: different number of clues found (*Max Payne 3*)

Figure 113 illustrates, using the game Max Payne 3, how two players can engage differently with the story elements of the game. In Max Payne 3, several clues can be found inside the game world, to inform the player about the underlying storyline. These clues have to be consciously looked for in order to be found. Participant A. (top summary) found three clues, while Participant P. (bottom summary) only found one. Participant P. seems to show less interest into the game narrative than Participant A.

Between-games

The last cross-transversal comparison considers the same player, but now playing two different games. Comparing two different games is more challenging, because unlike the previously presented cross-transversal comparisons, it is not possible to compare identical metrics, as metrics are highly game-dependent. It is then necessary to first classify the different metrics and establish their significance. In Figure 114, the idea is to compare how participant P. engages, in terms of story and interest in the game world, with the games *Bioshock 2* and *Max Payne 3*. In *Bioshock 2*, a good indicator of the interest of a player for the game world is the amount of time spent in the help menu; while in *Max Payne 3*, a good indicator is the number of clues found. As illustrated by Figure 113, participant P. only finds one clue during the first session, while participant A. finds three clues, and this suggests that Participant P. seems to be ignoring most of the story. However, Figure 114 illustrates how, during the first session of *Bioshock 2* (top summary), participant P. spends almost ten minutes in the help menu, and this implies that participant P. is engaging with the game story. This comparison is interesting, because both games have a strong background story, but they are also distinct. It could then be interesting to interview participant P. on this specific comparison, and ask what elements of the story in *Bioshock 2* were engaging, and what elements of *Max Payne 3* were of less interest.

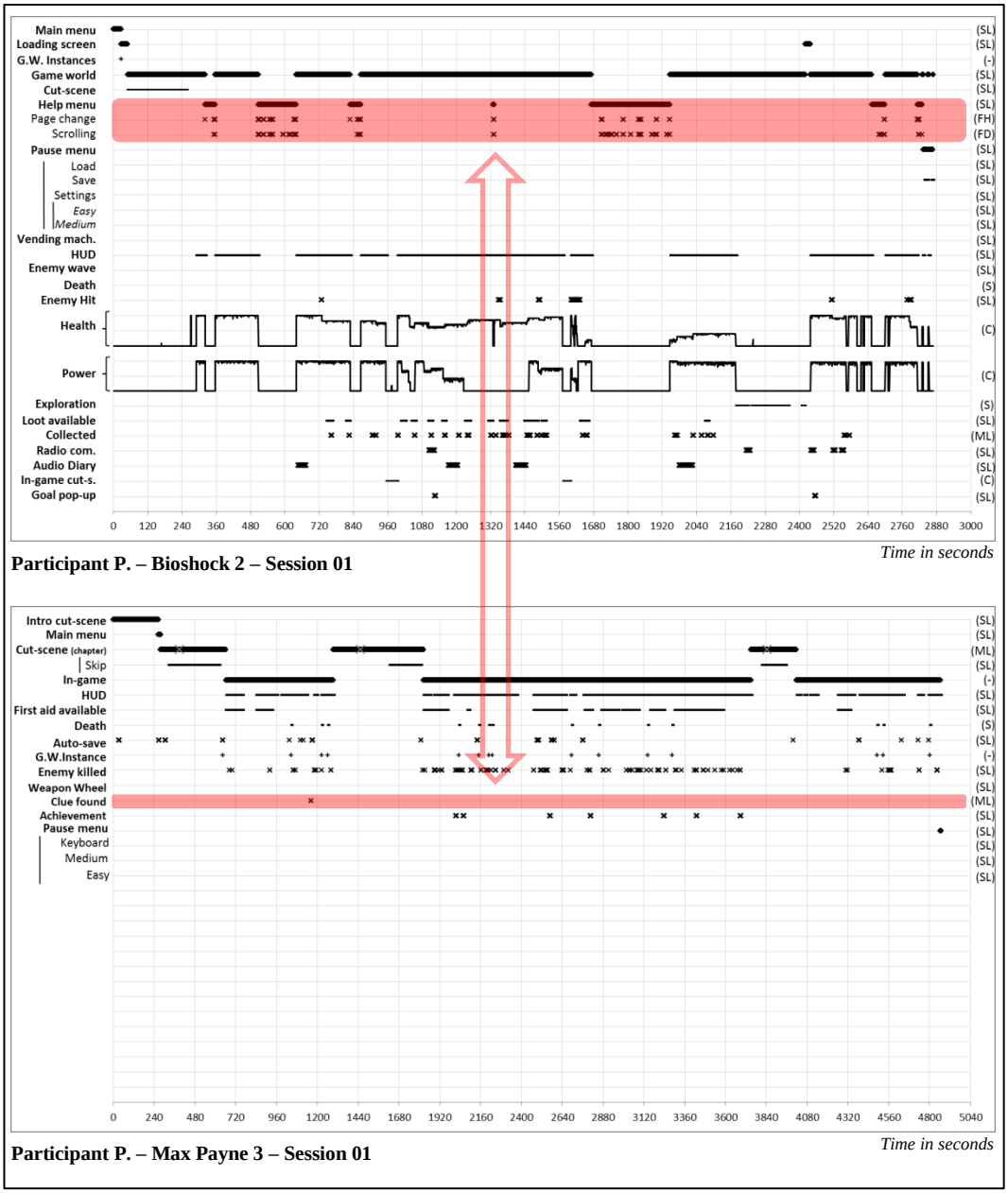


Figure 114. Between-games comparison: different interest in game story (*Bioshock 2 and Max Payne 3*)

Figure 114 illustrates the cross-games comparison (same player, different game), using the games Bioshock 2 (top summary) and Max Payne 3 (bottom summary). In Bioshock 2, the player shows a great interest in the story, as indicated by the regular use of the help menu. However, in Max Payne 3, the same player only comes across one clue. Participant P. is then not always a story-oriented player. The player shows different interest in the game narrative regarding the style of the world and games they play.

Cross-transversal comparison is a powerful reconstruction mode, as it allows the understanding of the player experience to go beyond the boundaries of a single performance. This means that a player evolution can be assessed through several consecutive sessions, and the player style can be compared with other players, as well as assessing the influence of the game genre over player engagement.

6.2 Free-text approaches

Section 6.1 illustrated the reconstruction process of a gameplay performance segmentation based on the multilayer architecture presented in Chapter 4, and using the algorithms presented in Chapter 5. The deconstruction and reconstruction processes are based on gameplay elements and concepts pre-determined by the analyst and the multilayer structure. These are automatic *controlled-vocabulary* approaches to gameplay performance segmentation; as a vocabulary of interest is pre-determined before the segmentation occurs. In Chapter 4, Chapter 5 and Section 6.1, the approaches and algorithms are based on gameplay concepts that are generic enough to be applicable to a vast majority of games, like loading screens, deaths, looting, menus, cut-scenes, health, stamina/power etc. The analyst determines, via the algorithm inputs, the concept of interest and the corresponding audio or video representation in the feedback streams, so that feedback-based gameplay metrics outputs can be used for gameplay performance segmentation. The rationale behind focusing on controlled-vocabulary approaches is that, in order to build a multi-layer segmentation process, it is essential to have a first analysis of the game that can elicit concepts able to lead to the deconstruction process, and can facilitate a performance reconstruction in terms of *player experience*.

However, if controlled-vocabulary approaches constitute the core of the present thesis, it is important to also acknowledge the existence of other types of approaches, defined as *free-text* (see Section 3.4.4), in the sense that the segmentation is not lead by a pre-determined vocabulary of interest, but by the game content itself. For instance, in the game *Bioshock 2*, the player can select

several topics of interest in the help menu. Detecting the help menu space, as illustrated in Section 6.1, is a controlled-vocabulary approach. Indeed, nowhere in the game is the term “help menu” written, but the notion of “help menu” makes sense in terms of gameplay analysis. However, detecting the moments when the player is reading about a specific topic, like for instance “Big Daddy” is a free-text approach, as “Big Daddy” is a term exclusive to the game *Bioshock 2*, and is not applicable to any other game outside this series.

The present thesis does not focus on free-text approaches, as they can be seen as parallel to the gameplay performance segmentation model, but are not essential for both the deconstruction and reconstruction phases. However, free-text approaches have an interest for a better understanding of player experience, and notably for comparing performances, between players (detection of similar moments) or with the same player (detection of repetitions). That is why the current section presents several free-text approaches, and has two goals to fulfil: showing that audio and video analysis of gameplay footage is not limited to the sole algorithms presented in Chapter 5; and also enhancing the understanding of what *controlled-vocabulary* entails, by also defining and illustrating the term free-text.

The present section first shows how the controlled-vocabulary methods presented throughout Chapter 5 can be adapted for free-text purposes. Then, this section presents different free-text solutions. The results given by these methods are driven only by the game content itself and not by any pre-selected gameplay concepts. The algorithms presented below have been developed in the context of

the present thesis to assess what free-text method can bring to an understanding of player experience. However, a choice has been made to focus on controlled-vocabulary approaches as they fit better with gameplay performance segmentation. That is why the following methods have not been as thoroughly explored and not fully validated. This section then should be seen as an exploratory possibility.

6.2.1 *Adaptation of Controlled-vocabulary approaches*

The algorithms presented in Chapter 5 have been utilised as controlled-vocabulary approaches. Indeed, their strength is their capacity to assess the evolution of a pre-determined gameplay concept in regard to a change in the audio-visual information. The controlled-vocabulary algorithms work efficiently because the audio-visual representations are consistent, meaning that their presence, absence, or evolution always carries a meaning throughout the whole performance and these are easy to determine. For instance, a death sound will be the same in *Battlefield 3* from the beginning to the ending; a help menu design is the same in *Bioshock 2* every time the player enters the menu; the life-bar representation stays consistent in *Dead Island*; and the HUD presence or absence in *Max Payne 3* carries the same indication. Providing the correct inputs (in the form of logos, sounds, masks etc.) is a task that needs to be accomplished once per game. However, applying the controlled vocabulary algorithms for free-text purposes, is a much more complex operation. For instance, in *Bioshock 2*, using a *controlled-vocabulary* approach would be to assess if the player is inside a help menu; while a *free-text* approach would be to assess the content of the help menu. Using the "static information" algorithm presented in Section 5.3 for detecting the current topic of interest is possible, but the topics tree is massive and it would become

time consuming to cover all the possibilities using the controlled-vocabulary algorithm[28]. Figure 115 illustrates different free-text elements that can be detected inside the help menu of *Bioshock 2*: the current topic title (WHAT IS ADAM?), the actual topic text, or the next selected topic of interest (The power of ADAM).

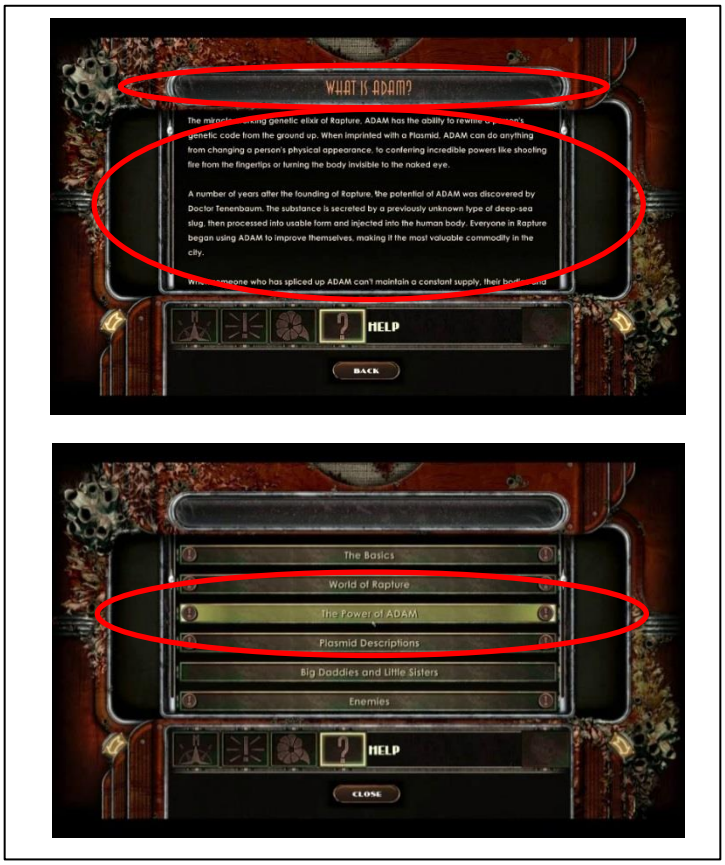


Figure 115. Free-text elements in the help menu of *Bioshock 2*

Figure 115 illustrates that several elements inside the help menu in Bioshock 2 can be considered as free-text information. The title WHAT IS ADAM?, the topic content, or the next topic of interest, ready to be selected by the player, the power of ADAM.

However, when the amount of free-text elements are reasonable to use as inputs, controlled-vocabulary algorithms can still be used as backup solutions. For

[28] A better idea can be for instance to automatically extract the text content using the Optical Character Recognition method. OCR extraction has been tried during the thesis project using the Tesseract-OCR library; unfortunately, the results, sometimes usable, were mainly random. It would be an interesting idea to push forward, but it is for now outside the context of the present thesis.

instance, in *Bioshock 2*, several narrative sequences, appearing linearly, are present in all the performances. Four of them have been selected, and presented in Figure 116: 1) the player is given their first power through an injection cut-scene; 2) they encounter the first boss fight with the "Big Sister"; 3) they are sent underwater by the "Big Sister" during a second encounter; and finally, 4) the player is presented with an external underwater view over the city of Rapture. Each of these four key moments is recognizable by a very specific sound and musical atmosphere.

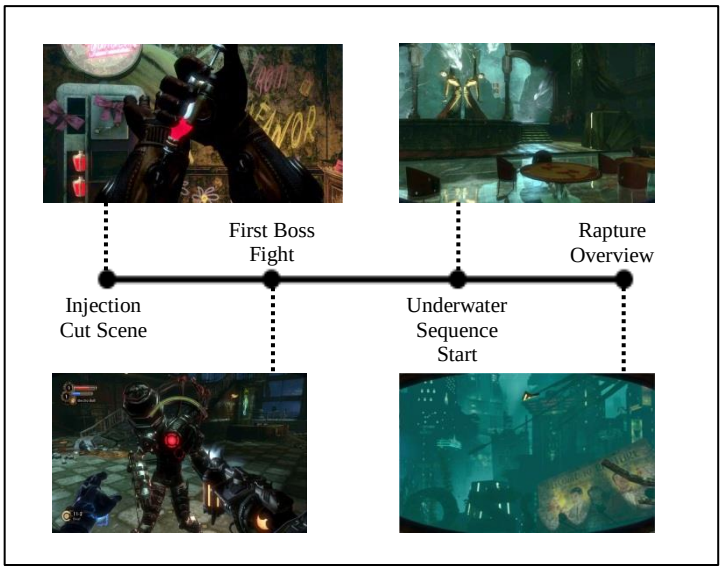


Figure 116. Key narrative moments in *Bioshock 2*, easily recognizable by their soundtrack

Figure 116 shows four key narrative moments in the game Bioshock 2, which can be easily recognized by studying the soundtrack. The player is given their first power through an injection cut-scene; they encounter the first boss fight with the "Big Sister"; they are sent underwater by the "Big Sister" during a second encounter; and finally, the player is presented with an external underwater view over the city of Rapture

It is then possible to use the sound-correlation algorithm presented in Section 5.7 to establish the temporal profile of these key moments, and compare these results between players. Figure 117 shows the result of the detection of these key moments using nine participants taken from the thesis field work.

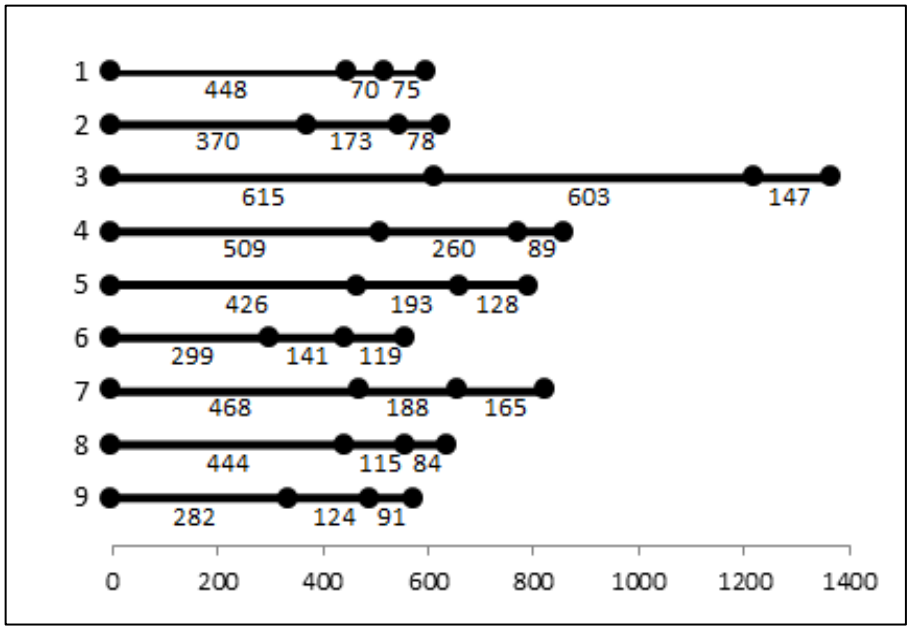


Figure 117. Key moments detection results, synchronised with the first key detection

Figure 117 shows the four key moments (Figure 116) detected using the sound cross-correlation algorithm. Because these moments are narrative sequences that only exist in Bioshock 2, they are considered as free-text. It becomes easy to compare different play-styles when the first key moment of each participant is synchronised. Participant 3, for instance, takes time between two key moments, probably interested to explore the game world, while Participant 1 seems to rush between two key points, probably more action-oriented.

By synchronising the first key moment of each of the nine participants, it is possible to study the time spent by each of them in between any two key moments, and therefore compare play-styles. For instance, it is interesting to compare the time spent by the different participants after the "Big Sister" fight (second moment), when "she" runs away; to find "her" again (third moment). Some participants rushed after "her" (participant 1, during 70 seconds), while others take their time (participant 3, during 600 seconds). Participant 3 is actually Participant P. in the summaries presented in Section 6.1, and has already been seen as reading about the story (in help menu), and looting the different corpses to

prepare themselves. More interestingly, between key moments three and four, a story sequence that happens underwater, where a “Big Daddy” protects a “Little Sister”, can be watched or ignored by the player. Participant 3, who has already shown interest in the story (long time in the help menu) obviously watched it (during 147 seconds), while Participant 1 obviously ignored it (half the time, during 75 seconds).

In terms of validity, each sequence is long (several seconds), so the cross-correlation method gives a 100% correct detection, when compared with hand coded results.

This *Bioshock 2* example shows that the controlled-vocabulary approaches presented in Chapter 5 can be applied for free-text purposes, and offers information that can be added on top of the deconstruction process, by describing the space in a more diegetic way. However, the complexity of each game in terms of story, or specific elements, makes the adaptation of the controlled-vocabulary algorithms to free-text indexing difficult.

6.2.2 *Frame hue/saturation*

After showing that adapting controlled-vocabulary approaches for free-text purposes is possible, though complicated, the question can be asked about what would be relevant for a free-text approach. A possibility would be focus on the audio-visual parameters level of the analysis of feedback streams (see Section 2.2.1) which produces audio-visual segmentation based on audio and video parameters. This would work, regardless the game, as the operating elements are audio-visually based, requiring no prior-knowledge of the game. The game content, through the audio and video feedback streams, guides the segmentation, without any necessity for a pre-analysis of the game content. Section 6.2.2 illustrates the use of the audio-visual parameters level by processing the video stream parameters, and this can give useful results depending on the game graphical design choices.

The graphical parameter used in the present section is the colour, and more specifically each of the H, S and V channels from the HSV representation (see Section 5.2.1). The analyst is not required to provide any input (other than the gameplay footage), and the hue or saturation of each pixel is classified, using five classes for hue (yellow, green, blue, purple and red) and two for saturation (full colour, black and white). The more represented class is then detected, frame by frame. The analyst does not need to provide any screen area or specify any colour of interest. The game, through the visual feedback, is the one leading the segmentation.

Two examples are provided in Section 6.2.2. The first one uses the game *Dead Island*, which is based on a simple colour scheme: the blood is very red; the menu very green; the island has a very blue and yellow look, etc. Assessing the principal hue of each frame gives insight into what is actually displayed by the game. For instance, each red frame can actually be seen as being part of the same semantic network, as they are all linked to blood. Figure 118 illustrates the use of hue to detect moments of interest in the game, without any pre-determined inputs given by the analyst.

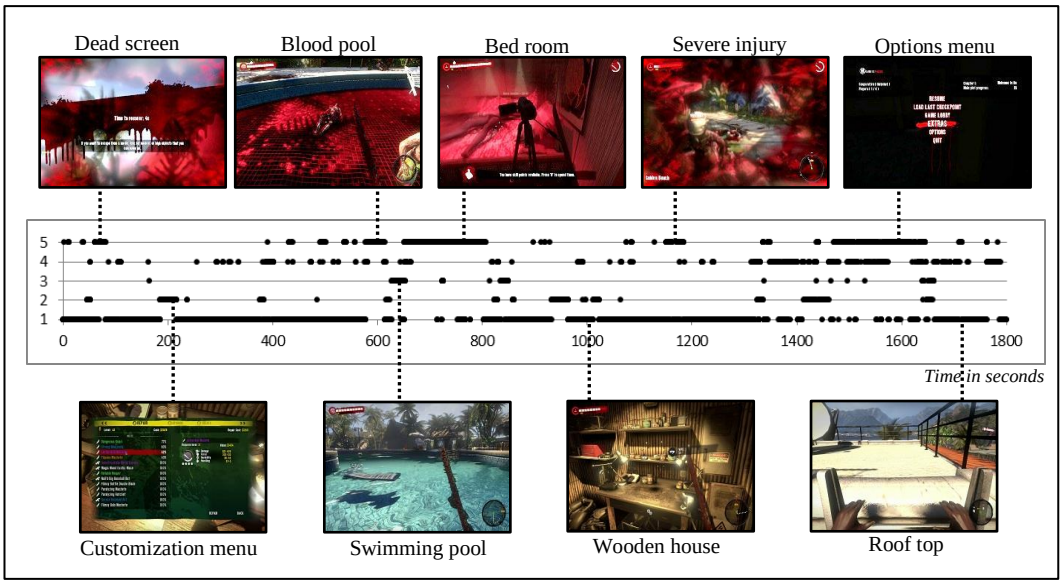


Figure 118. Hue recognition using the game *Dead Island*

Figure 118 illustrates the use of hue recognition with the game Dead Island. The detection of hue allows for a free-text analysis for the performance. The interesting moments are the ones linked with the red hue, which gives insight about the gloomy nature of the sequence: death screen, severe injury, blood pool, blood-looking menu, bed room with an attached zombie on the bed. The y-axis represents the detected hue: 1 yellow, 2 green, 3 blue, 4 purple, 5 red

The red frames can be seen as matching some of the videogame classification moments of concern (see Chapter 1). Then, automatically detecting them can be a solution to quantify them and include them in a *player experience* assessment.

The second example focuses on the game *Max Payne 3*. In *Max Payne 3*, desaturation is used to represent several key moments of the game. Figure 119 illustrates that these de-saturated moments are always linked to moments when then player's degree of freedom is altered. For example: 1) during tutorials, the player is forced to do a specific action (T in Figure 119), 2) in the cut-scene, there is no interaction, except to skip (C on Figure 119), 3) in bullet cam, the camera is slow downed (B on Figure 119), 4) death, there is no interaction except retry or quit (D on Figure 119) and 5) in the last man standing moment, the player is forced to shoot, and cannot move (L on Figure 119). The de-saturated moments can also be seen as being part of the same semantic network, which concerns limitation of agency.

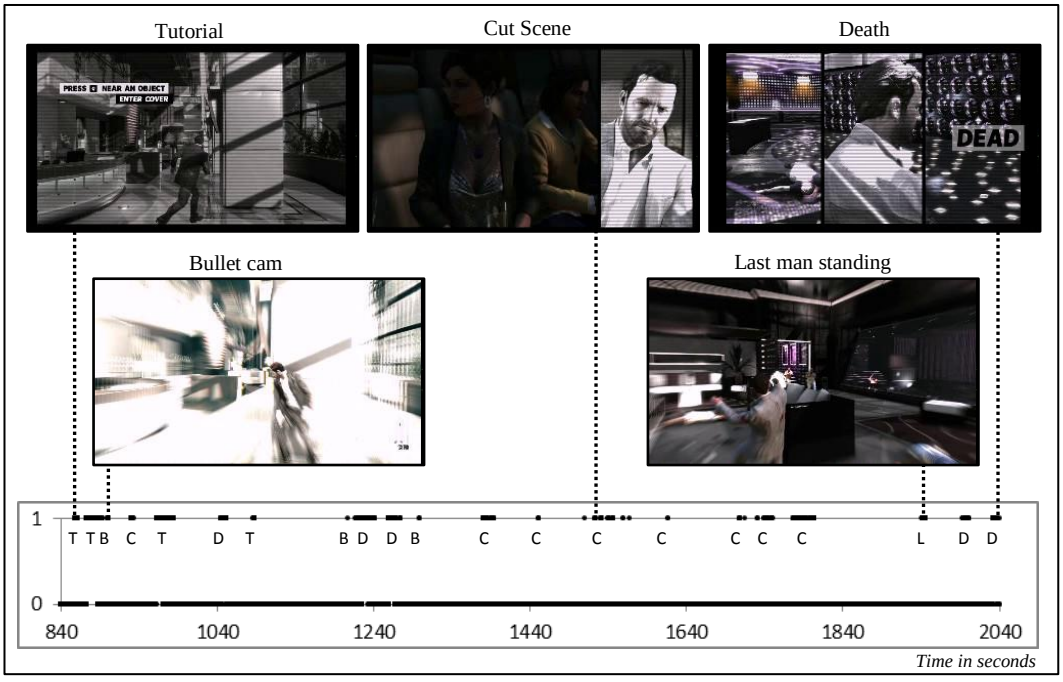


Figure 119. Saturation recognition using the game *Max Payne 3*

Figure 119 illustrates the use of saturation recognition which allows for a free-text analysis of the performance. However, with Max Payne 3, de-saturated sequences can be linked with an alteration of the degree of freedom. So this free-text method can, in this special case, be linked to one of the controlled-vocabulary layers. T is of tutorial panels. C is of cut-scenes. D is for deaths. B is for bullet-cams. L is for last man standing. The y-axis represents the saturation level: 0 for full-colour, 1 for black and white.

6.2.3 *Speech detection*

Another free-text method, based on the audio stream, identifies speech segments. The method utilises Hidden Markov Models (HMM), based on classical speech processing features, the Perceptual Linear Prediction coefficients (Rabiner, 1989). The idea is to train the models with generic sound streams, labelled as being speech, music, other etc. The game soundtrack can then be segmented and labelled using the Viterbi algorithm (Viterbi, 1971). The models have been trained using radio broadcasts, instead of videogame streams, in order to not make the method game specific. Figure 120 illustrates the speech detection using three participants playing *Bioshock 2* (the grey areas represent the speech detection). The participant numbering is the same as on Figure 117, so the key moments detected in Section 6.2.1 can be added to the graph.

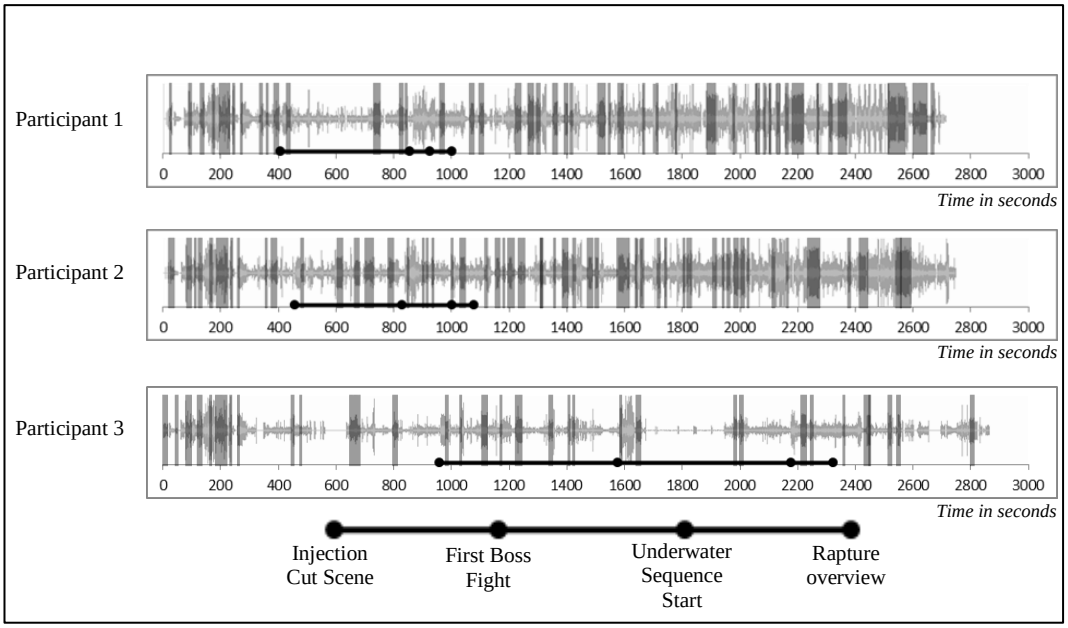


Figure 120. Speech detection in the game *Bioshock 2*

Figure 120 shows a speech detection result using the game Bioshock 2. The grey areas on the graph represent a detection of speech. As the performances are the same as the one in Figure 117, the four key moments are also added. Between the injection sequence and the overview about Rapture, Participant 1 only encountered four segments of speech, while Participant 3 encountered thirteen. Counting the moments of speech can validate inferences about a certain play style: Participant 1 has a more action driven play style, ignoring most of the available audio cues and Participant 3 seems to take time to listen to most of the narrative containing audio tapes.

As highlighted in Section 6.2.3 and Figure 120, Participant 3 took ten minutes to go from the first "Big Sister" encounter to the second encounter. Figure 120 shows no speech detection between 1650 seconds and 1950 seconds. This matches the fact that Participant 3 is in the help menu which contains no background audio. But more telling, counting the speech moments between the injection sequence and the overview about Rapture reveals that Participant 1 only encountered four segments of speech, while Participant 3 encountered thirteen. Participant 1 also gets to the Rapture overview when Participant 3 just finished watching the injection sequence. It seems that counting the moments of speech can validate inferences about certain play styles: Participant 1 has a more action driven play style, ignoring most of the available audio cues and Participant 3 seems to take time to listen to most of the story delivered on the audio tapes.

In terms of validity, the speech detection algorithm has a low false alarm rate and a good precision level, meaning that a detected sequence is highly likely to match the hand-coded version. However, the missing detection rate (amount of speech moments existing in the hand-coded version, but not detected) is around 50%. This could be explained because of the high level of noise accompanying the speech. In general, however, the missing detection occurred from within a correctly detected segment, so the number of segments is still valid. The algorithm needs to be improved to be used more thoroughly, but it is a useful example highlighting how the sound stream can be used for free-text purposes.

6.2.4 *Similarity matrixes*

The last free-text method to be discussed considers the game content in an abstract way but it is also useful as a method to understand player experience understanding. The approach is based on the concept of *similarity matrices*, which is an approach currently widely used for automatic document segmentation and indexing (M. L. Cooper & Foote, 2001; Foote & Cooper, 2003; Hanna et al., 2008). A similarity matrix is a visualisation of the degree of similarity that exists between two distinct documents or inside the same document (when it is then called self-similarity matrix). The process is to analyse the document as sub-units (words for text, frames from video, time-segments for audio), compare each of these sub-units, give a similarity score using a comparison formula, and finally construct a matrix showing all the comparison results, unit to unit. The matrix can be represented in a form of an image, the darker the pixel, the closer the similarity. For textual documents, the similarity can be determined if two compared units are part of the same lexical field. For video, the similarity can be through colour distribution (like in Section 5.6). For sound, it can be through cross-correlation (like in Section 5.7).

In the present section, the proposed similarity method is audio-based, and relies on the concept of *Harmonic Pitch Class Profile* (Fujishima, 1999), sometimes also called *chroma*. A chroma can be seen as the frequency distribution of a portion of music in terms of the twelve usual semitones of the equal tempered scale. The contiguous chromas of the soundtracks can then be compared between them, using a simple distance metric. The chroma method, based on frequency distribution, is far faster than any cross-correlation method. Figure 121 shows a

similarity matrix, generated with the chroma method, based on two different performances of the beginning of *Max Payne 3*, one by Participant P, the other by Participant A. The Participant A. soundtrack is on the horizontal axis and Participant P on the vertical one. Each pixel represents the comparison between the matching time in Participant A.'s performance and Participant P.'s performance. Each square mark represents five minutes, so the two performances are one hour and twenty minutes each (truncated for readability purposes). The main patterns to look for in a similarity matrix are diagonal lines. Indeed, each diagonal represents a contiguous sequence in one performance that is perfectly matching another contiguous sequence in the second performance. *Max Payne 3* uses numerous sound cues. Background music is omnipresent, and Max regularly speaks to the player, by thinking aloud. Each cut-scene, but also each game chapter, have their own specific sound atmosphere and can then be compared using the chroma approach.

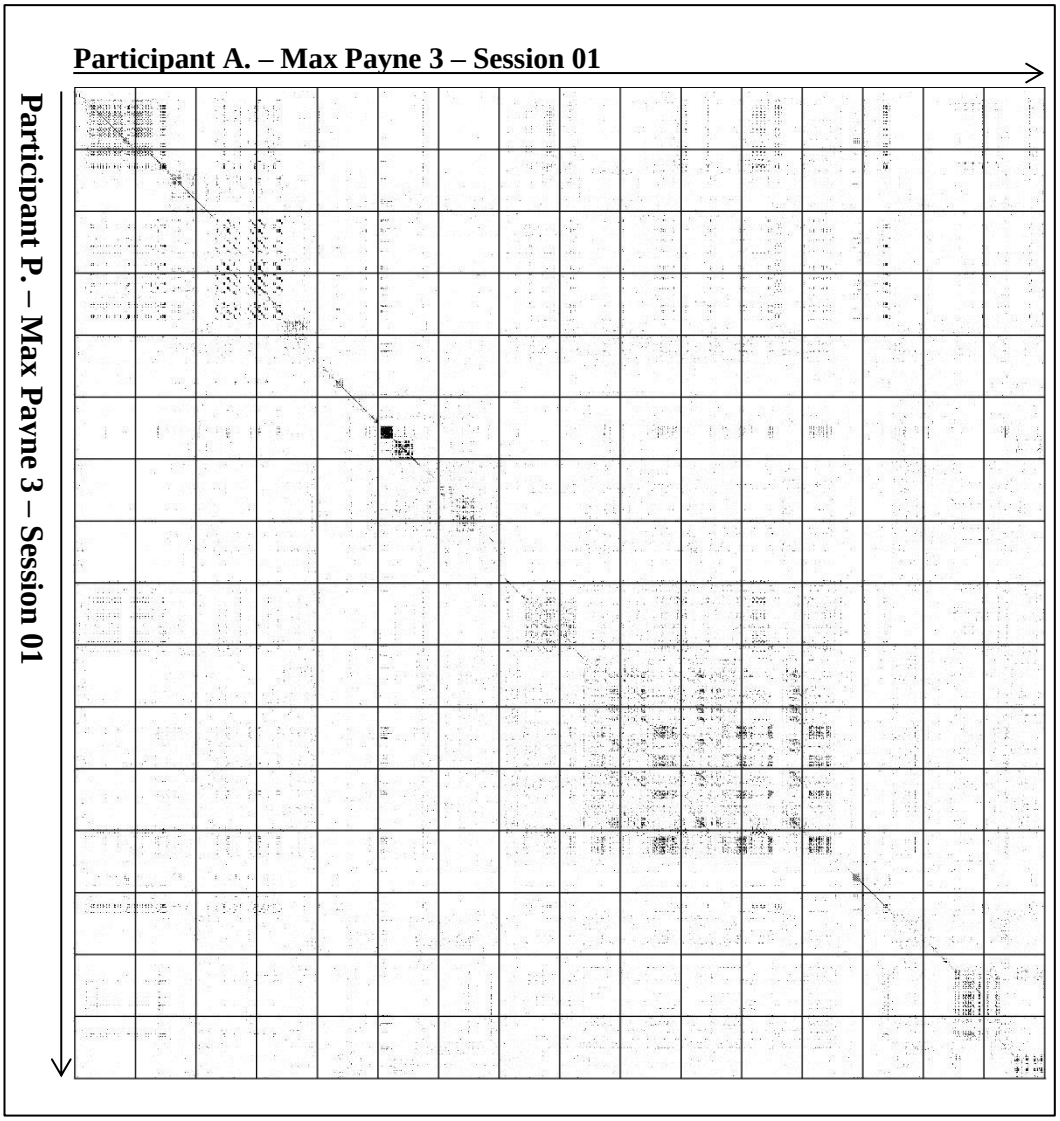


Figure 121. Similarity matrix using two performances of the game *Max Payne 3*

Figure 121 is a similarity matrix, generated by sound comparison, and uses two performances of the game Max Payne 3. Each pixel represents the comparison between the matching time in Participant A.'s performance (read on the horizontal axis position), and Participant P.'s performance (read on the vertical axis position). Each square mark represents five minutes. The main patterns to look for in the similarity matrix are diagonal lines. Each diagonal represents a contiguous sequence in one performance that matches another contiguous sequence in the second performance.

Figure 122 represents an annotated version of Figure 121, for an easier discussion and analysis. (1) represents the introduction cut-scene, that crossfades directly into the main menu (the cut-scene continues in background, looping on Max drinking and smoking, sat at a table). (2) represents the first game chapter introduction. There is a break in the line between (1) and (2) because Participant A spends one minute more in the menu, configuring the game. (3) represents the cut-scene between chapter 1 and chapter 2, that Participant A achieves after twenty minutes of play, and Participant P after twenty-two minutes. The square represents a moment in the cut-scene when the music is looped, making all the units in the area similar. (4) represents the hardest sequence of chapter 2, so both participants A. and P. died a lot there. As the music is played in a loop, and starts again after each death, this explains the square shape. Finally, (5) and (6) represent the cut-scene between chapter 2 and 3, which becomes semi-interactive half-way. The player can die during the semi-interactive sequence (they can just move, but not shoot). Participant A. dies once, which explains the break between (5) and (6). (6) ends when the player gets back the full interactivity, and participants A. and P. are playing differently from this point onwards, and this, therefore, ends the diagonal.

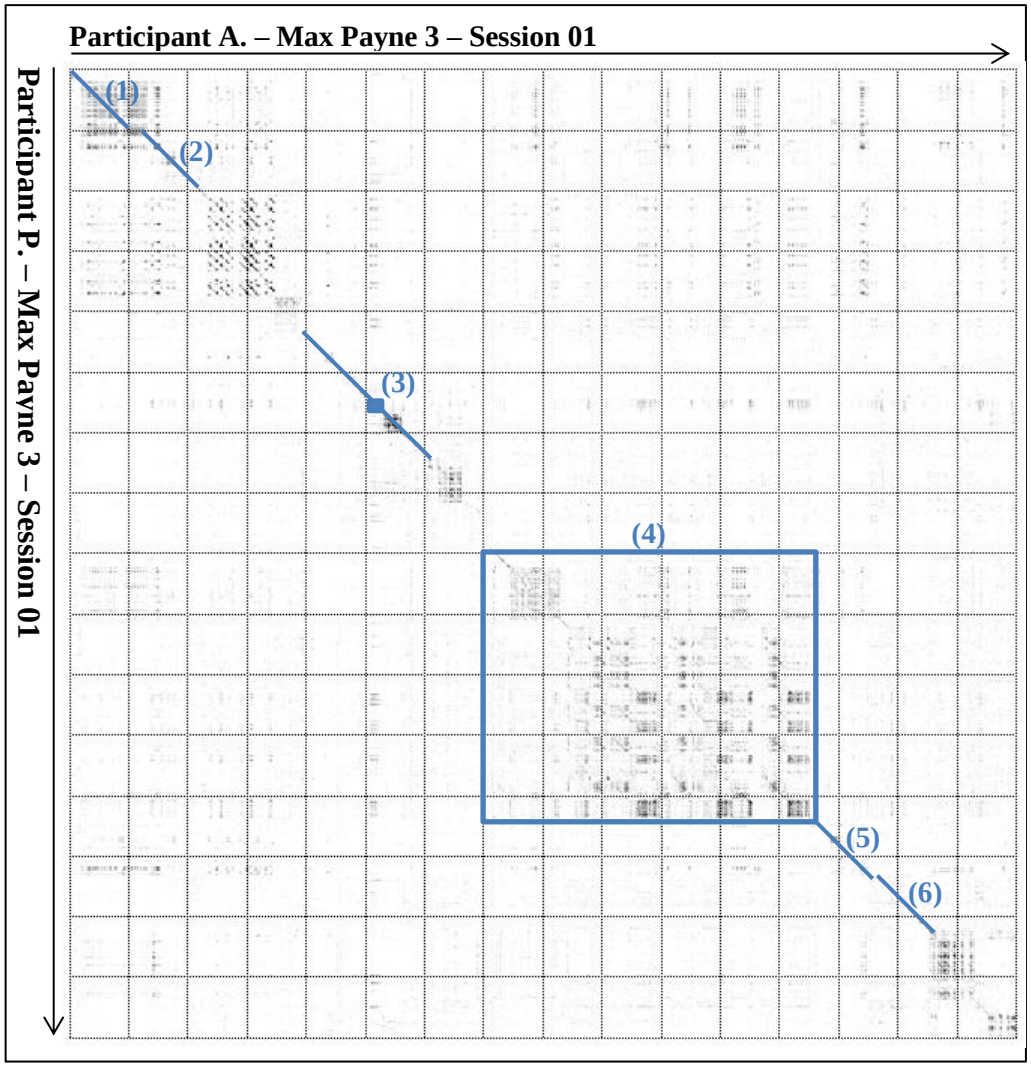


Figure 122. Annotated similarity matrix

Figure 122 is an annotated version of Figure 121, emphasizing several diagonals of interest. (1) represents the introduction cut-scene, (2) represents the first game chapter introduction. The line between (1) and (2) breaks because Participant A spends one minute more in the menu, configuring the game. (3) represents the cut-scene between chapter 1 and chapter 2. The square represents a moment in the cut-scene when the music is looped. (4) represents the hardest sequence of the performance. The music is played in loop, and starts again after each death, and so this explains the square shape. Finally, (5) and (6) represent the cut-scene between chapter 2 and 3, that becomes semi-interactive in the middle. The player can die during the semi-interactive sequence. Participant A. dies once, which explains the break between (5) and (6).

Self-similarity

Another way of using a similarity matrix is to compare a performance with itself, giving a self-similarity matrix highlighting repetitions, or loops. Figure 123 shows a self-similarity matrix of Participant N. diagramming a performance of *Battlefield 3*. The game uses musical atmosphere less than *Max Payne 3*, except during the death sequences and loading screens. In these cases, the diagonals are likely to represent death sequences[29]. Not only are the deaths easy to locate through the localisation of diagonals, but also the difficult sequences are clearly highlighted. The bottom right corner for instance is filled with diagonals and actually represents one of the hardest sequences of the game.

[29] As this is a self-similarity matrix, the main diagonal is obvious, because it represents a unit compared with itself, and the main diagonal is also a symmetry axis, as a unit a time stamp t_1 horizontal, compared with t_2 vertical is the same as a unit a time stamp t_2 horizontal, compared with t_1 vertical.

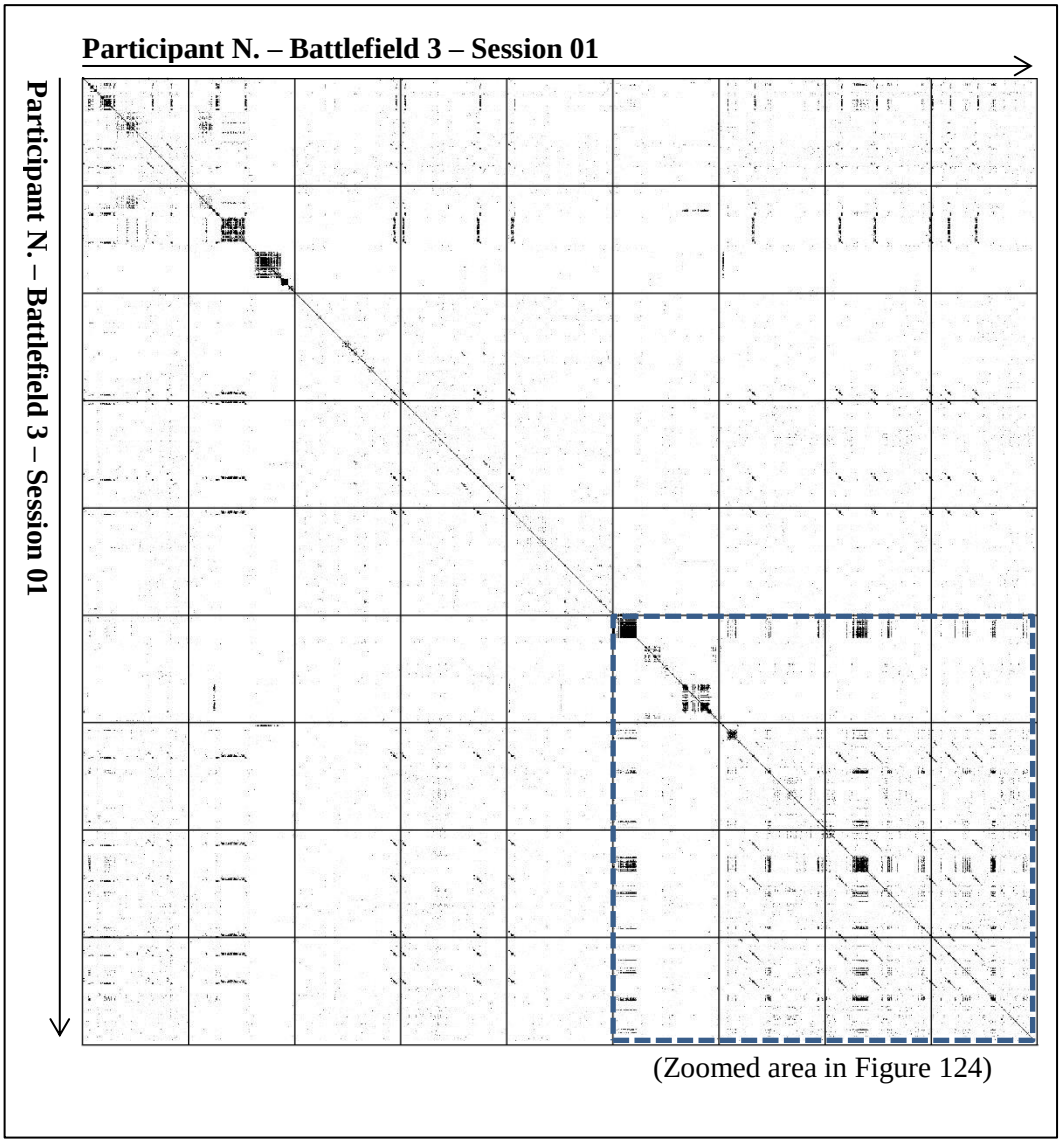


Figure 123. Self-similarity matrix example, using the game *Battlefield 3*

Figure 123 illustrates how a self-similarity matrix can be used to automatically detect repetitions from within the same performance (each square mark represents five minutes). In Battlefield 3, the death sequences have a very distinct soundtrack. So the diagonals in the above matrix are likely to represent deaths. The more diagonals in an area, the more difficult the sequence may be.

Figure 124 focuses over the bottom right section of the matrix in Figure 123, to also highlight another powerful use of a self-similarity matrix.

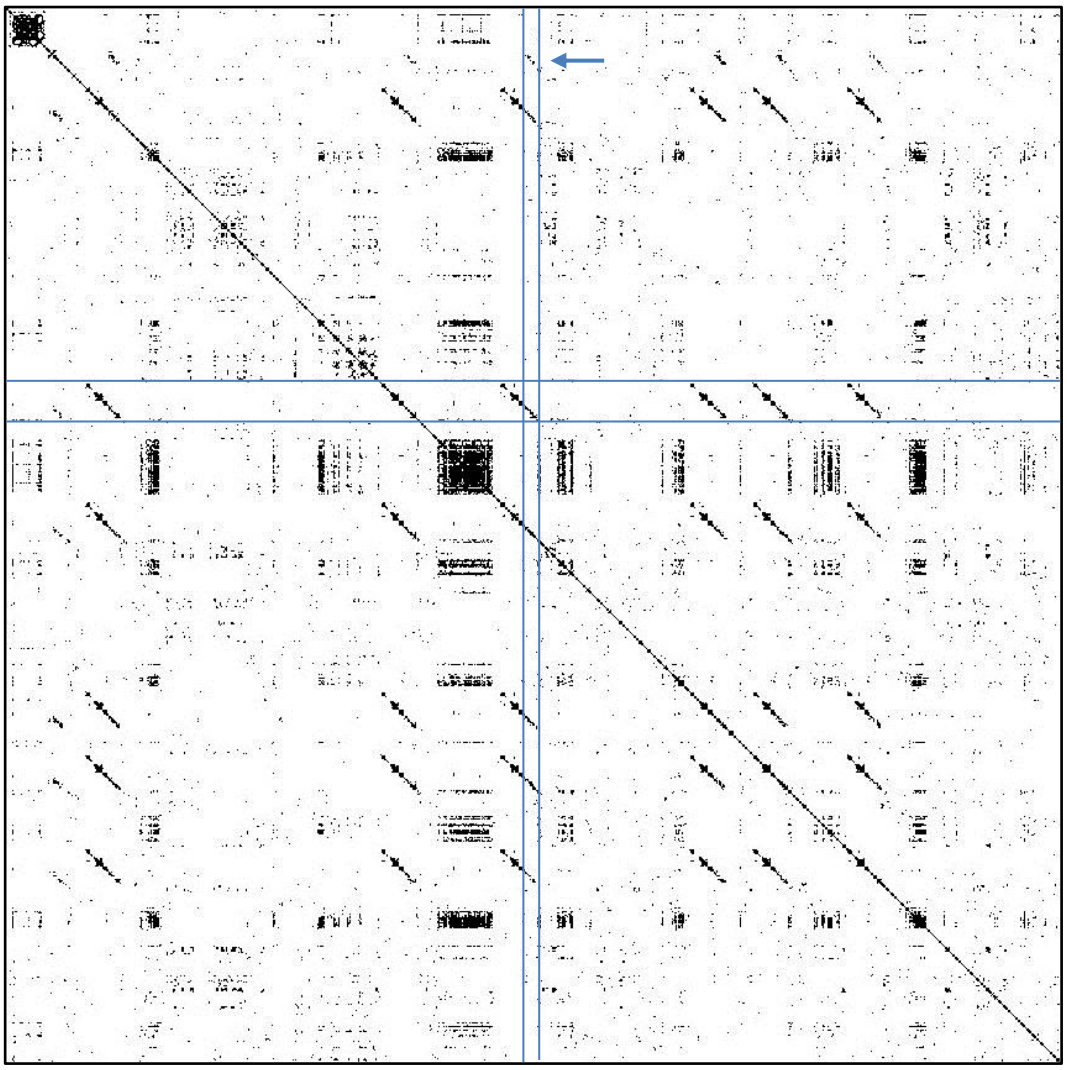


Figure 124. Self-similarity matrix zoom, and death sequence detection

Figure 124 is focuses on the bottom right corner of Figure 123. The zoom allows easy detection of the death segments: six deaths at time 1890, 2134, 2233, 2390, 2443 and 2521 seconds. However, because a diagonal is actually the sum of three sequences: death, loading screen, and beginning of the retry sequence (radio message), the smaller diagonal, pointed by the arrow at 1863 seconds represents the first time the sequence is encountered (radio message) but without any prior death. Here, the free-text method can detect moments, insightful for loop detection, which are not gatherable through controlled-vocabulary methods.

The horizontal line annotations represent where it is possible to locate the different deaths during the bridge sessions (six deaths at time 1890, 2134, 2233, 2390, 2443 and 2521 seconds). But actually, Figure 125 illustrates that a death sound represents more than the death screen. When dying, a full sequence is

repeated: the death screen, the loading screen (with a specific background music) and, importantly, the beginning of the bridge sequence, with the same incoming radio message.

Death

Positions in the matrix diagonals

Loading screen

Radio communication

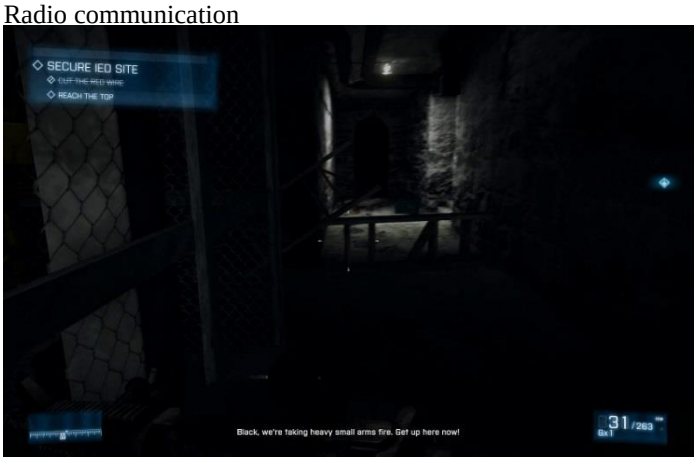


Figure 125. The three sequences constituting a death diagonal in the self-similarity matrix

Figure 125, based on the game Battlefield 3, illustrates the three sequences that constitute a death diagonal in Figure 124, the death screen, the loading screen, and the radio communication beginning the sequence; along with a representation of the diagonal part that matches each sequence.

Knowing that, the top diagonal in Figure 124, that is shorter and matches the end of all the death diagonals in this section, represents actually the first instance of the radio message, without any prior death. It is then possible to locate the first time the player entered the difficult section, around 1863 seconds.

The chroma approach is highly accurate and meaningful to compare player experience, and the detection of similar performance sequences. As two similar sound sequence can differ visually, it would be an interesting idea to also try similarity matrix using colour distribution.

6.3 Conclusion

Chapter 6 has been dedicated to demonstrate that play can be interpreted from gameplay performance summaries generated using feedback-based gameplay metrics processing. If Chapter 5 contains several examples of player experience interpretation, each result is interpreted independently, as the main purpose of Chapter 5 is to validate the different introduced algorithms. Chapter 6, however, uses fully deconstructed performances, in order to illustrate how the five layers presented in Chapter 4 can be used transversally for understanding player experience in a more exhaustive and correlated way. Chapter 6 provides examples, from four different games, to the different mode of transversal reconstruction first described in Chapter 4: contextual, temporal frame of reference, semantic network, loop and comparison. The comparison interpretation is also interesting as it can be seen as cross-transversal, meaning that the metrics of interest for assessing the player experience can be located in two different performances. It is interesting to note that, if the summary presented in Chapter 6 have been generated using feedback-based gameplay metrics, in order to

demonstrate the strength of the methods described in Chapter 5; the different transversal analysis can be applied to summary generated using other metric modality (like more traditional gameplay metric for example). The contribution of Chapter 6 is then not limited to strengthening the feedback-based gameplay metrics approach, but the transversal analysis can also be adapted and used for assessing player experience based on research works already existing in the game studies community.

Then, Chapter 6 introduced a parallel approach to interpret play[30]. Based on the direct content of the game instead of pre-determined concepts of interest, free-text processing of audio-visual content offers an alternative to controlled-vocabulary approaches for the purpose of performance comparison. The strength of free-text methods is that they do not require any input from the analyst as they are based directly on the audio-visual parameters of the streams, and can then be directly applied to gameplay footage, without a pre-analysis of the footage. They are however limited by the abstract nature of the outputs. If the results can be easily used for comparison purposes (such as using similarity matrices), a full understanding of the player experience requires a post-analysis of the footage. However, free-text approaches can be further studies, as algorithms like the ones extracting textual information from audio-visual content (Slik et al., 2013) can give strong insight about the content of a specific key moment in the game.

[30] For future work, it would be interesting to explore the automatic detection of patterns in game data, based on genetic algorithms, which would allow for some automisation of interpretation. The work of Hector P. Martinez (2011) into frequent sequence mining would be of value in this context.

Now that different solutions to interpret play based on feedback-based gameplay metrics have been presented, Chapter 7 will describe the software system developed and used to automatically gather them from gameplay footage, and present different applications of feedback-based gameplay metrics when conjointly interpreted with other modalities, such as biometry or key strokes.

Chapter 7

Implementing Gameplay Performance Segmentation alongside Other Measures

Chapter 5 presented audio-visual algorithms that automatize the process of achieving a segmented *gameplay performance segmentation*, a process that is guided by the multi-layered architecture presented in Chapter 4. Each layer focuses on a specific concept (controlled-vocabulary) that represents a specific aspect of gameplay. This is done in order to delineate (segmentation) and describe (indexing) segments of play – coherent in terms of experience – that constitute sections that may also be recalled by the player as a memorable section of play. Chapter 6 provided several summaries taken from several *gameplay performances* that illustrated how a *gameplay performance segmentation* and *player experience* can be analysed conjointly during a reconstruction process. Chapter 6 also introduced a wider range of algorithms, which enable the method to be applied in a *free-text* approach. The different results, in Chapter 5 and Chapter 6, have been produced using a software system developed throughout the duration of the research. So far, however, neither the software system nor its range of applications have been presented. This is the aim of Chapter 7, as it seeks to provide examples of the applied value of the method outlined in this thesis.

This chapter focuses on presenting the software system developed to gather *feedback-based gameplay metrics* and assess the validity of the gameplay performance segmentation approach. The description provided in this chapter also includes the software code structure and documentation. The intention is to

release the software as an open-source contribution to the game studies community. Chapter 7 also illustrates how the outcomes of a gameplay performance segmentation can be applied beyond a mono-modal consideration of performances (i.e., where only feedback-based gameplay metrics are considered on their own or in isolation to describe player experience), and used in conjunction with other modalities that provide measures of player experience. The modalities of interest discussed here include, but are not limited to, biometry and keystroke. Moreover, the applications of the method and software produced in this research are used to address the notion of objectionable content as defined by videogame classification process (see Chapter 1). Indeed, applications of the method has been used to highlight how players react to sequences of concern, such as slow motion in *Max Payne 3* (Rockstar Games, 2012), or fighting sequences in *Dead Island* (Deep Silver, 2011). The potential of *biometry* as a modality capable of guiding segmentation has been explored with *Battlefield 3* (Electronic Arts, 2011).

Three examples of a multi-modal application of the method are presented in the current chapter. In order to utilise the method in a multi-modal assessment of player experience, a visualisation tool that synchronises outputs within a single video file has first been developed. It has been possible to produce a video that contains the performance in several forms; comprising of the captured footage, several feedback-based gameplay metrics outputs, biometric data and the player's facial and gesture expressions. Section 7.2 describes how the visualisation tool has been used to help determining relevant questions for interview sessions with the participants (cf. Section 2.3).

The second example addresses slow motion sequences in the game *Max Payne 3*, along with the player engagement and reaction to slow motion effects. Section 7.3 illustrates how feedback-based gameplay metrics can be used to segment *Max Payne 3* performances for their use of slow-motions, and then analyse these conjointly with the player's input during these moments to assess the degree of the player's interaction and engagement with these sequences.

Finally, instead of segmenting a performance beginning with feedback-based gameplay metrics method, before comparing the results with other measures, the approach presented in Section 7.4 initiates an analysis based upon biometric data, highlighting segments that trigger excitement in a player. Segments identified through biometry are assessed to study if they can be correlated and validated by feedback-based gameplay metrics. One rationale, beyond allowing the player to dictate what is significant and analysed, is to delve into a gameplay performance without needing to view the gameplay footage representing a play session. That is, examine play in its quantified form. Such an approach also aims to add objectivity to what has so far required player-analyst to engage with games of interest.

Before presenting multi-modal applications, the core software system that allows feedback-based gameplay metrics and gameplay performance segmentation concepts to be empirically validated throughout the project is presented. The software system is used for all multi-modal applications presented in this chapter.

7.1 Feedback-based gameplay metrics software system

In parallel with the conceptualisation of feedback-based gameplay metrics and gameplay performance segmentation, a software system had to be developed in order to empirically assess the validity of the method presented in the present thesis. The software system constitutes part of the contribution to knowledge of this PhD thesis. Section 7.1 describes the structure of the software system source code that will be released as an open-source software, as well as the rationale that influenced core development decisions and its usefulness.

The software system processes videos representative of gameplay performance[31]. Section 7.1 provides a short description of the software system, both in terms of its goals and technical features, before the chapter moves on to present the source code structure, its usage, and the possibility to adapt it for further considering new audio-visual elements in the future.

[31] The sound algorithms have been developed in collaboration with the LaBRI (Laboratoire Bordeaux de Recherche en Informatique, University of Bordeaux, France), using a sound library that is yet to be published. For this reason, Section 7.1 focuses exclusively on the video analysis, which has been developed exclusively at the University of Waikato. However, the source code structure and usage can be easily translated to sound analysis.

7.1.1 *Software system presentation*

The software system presented here has been incrementally developed, tested, adapted and upgraded in order to empirically validate feedback-based gameplay metrics method and gameplay performance segmentation concept. The software system was developed using *C++ programming language* (Stroustrup, 1997), on the *Microsoft Windows 7* (64 bits) operating system (Microsoft, 2009), using the *gcc 4.6.1 compiler of MinGW* (Colin Peters & MinGW Project, 1998) and *Eclipse CDT Indigo* (Eclipse Foundation, 2011) as an Integrated Development Environment. For video processing, the open-source cross-platform library *OpenCV* (v 2.3.1) (Intel Corporation, Willow Garage & Itseez, 2011) is employed to load, save and process gameplay footage. The rationale behind using *MinGW*, *Eclipse*, and *OpenCV*, is to simplify future ports on other operating systems, as all three are cross-platform.

The main purpose of the software system is to automatically extract feedback-based gameplay metrics from video streams. The two main objectives pursued during the software development stage were: (1) to have the source code structured in such a way that it is possible to add new video processing algorithms easily without modifying the core part of the source code (such as a plugin system); (2) to be able to configure and execute the desired algorithms through a configuration file, without needing to recompile the source code each time there is an algorithm input change. Both goals are essential, because of the numerous existing audio and video processing algorithms. Feedback-based gameplay metrics gathering must not be limited to only a handful of algorithms, but should be able to accommodate further algorithms and new ideas for processing.

Furthermore, the software system is intended to be used by analysts that do not necessarily hold a background in computer science (and should not be forced to use the opaque mechanism – for a non-computer scientist – of compilation), or modify the source code every time they need to process a new footage or use an algorithm with different inputs. In the source code, it is important to specify that the word filter is used to represent an implementation of an audio-visual *algorithm*. *Filter* designates the implementation, *algorithm* represents the process.

7.1.2 *Source code structure*

Figure 126 displays the source code architecture using a UML class diagram (Bruegge & Dutoit, 2010, Chapter 2). The source code can be separated into four main sections that are described in more detail in the following paragraphs: a) *file management*, which deals with parsing the algorithm configuration file and logging the feedback-based gameplay metric outcomes*;* b) *media management*, which deals with loading a video file, and extracting the video frames; c) *media filter systems*, which processes the different frame extracted by the *media management* part in order to compute the metrics; and d) the *core engine*, which links a, b and c together, while providing a user-friendly library, that hides most of the code complexity for future programmers.

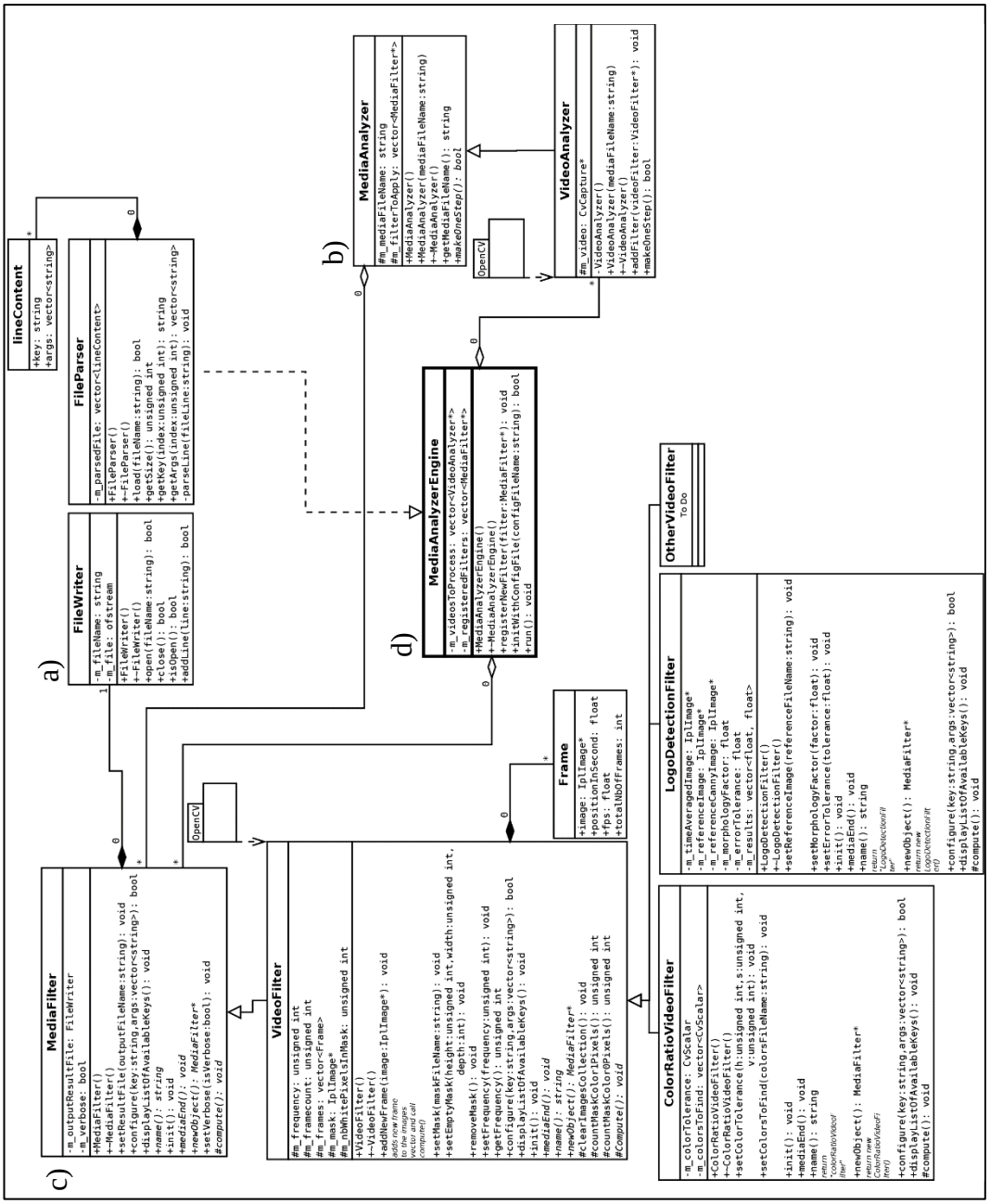
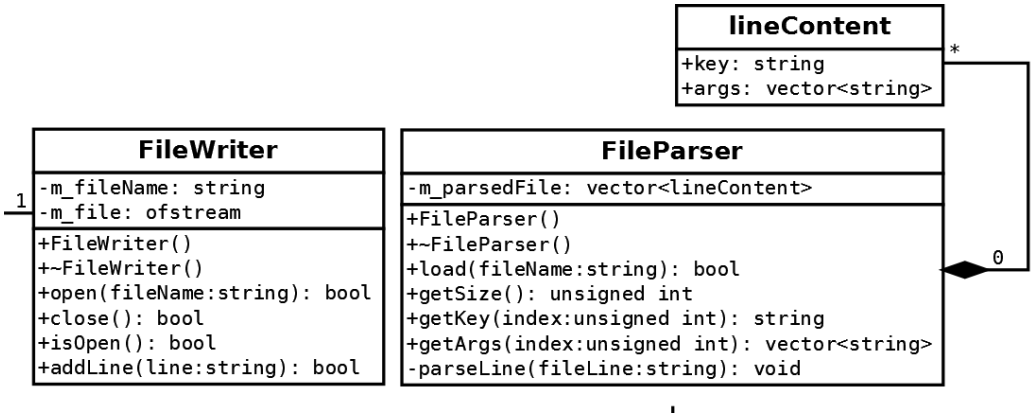


Figure 126. Feedback-based gameplay metric software system - UML class diagram

Figure 126 displays the UML class diagram of the feedback-based gameplay metrics software system. Four main sections can be distinguished: (a) file management, which parses the algorithm configuration file and logs the feedback-based gameplay metrics outcomes; (b) media management, which loads a video file and extract the frames of interest: (c) media filter system, which processes the different frames and computes the metrics; (d) core engine, which links the above parts together, and proposes a user-friendly library hiding most of the source code complexity of future users.

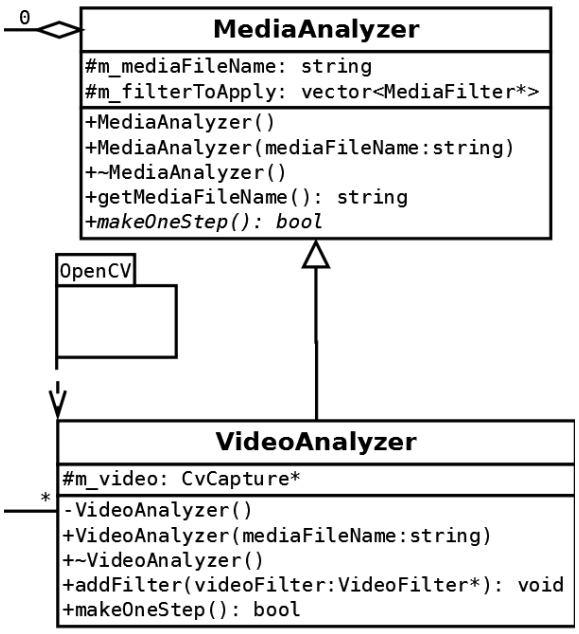
File management



The *file management* section of the source code deals with the management of text files, whether it is for loading and parsing the algorithm configuration file[32] or for logging the feedback-based gameplay metric results into a text file. For parsing, each line of the configuration file a key plus a list of arguments. For instance, "TOLERANCE 10 20 30" for the HSV tolerance values of the bar progression method (presented in Section5.5) is understood as: the algorithm input TOLERANCE that possesses the three values 10, 20 and 30. For the logged file, the format is in general a tab of two columns: one for the frame time position, and another for the computed value of the metric.

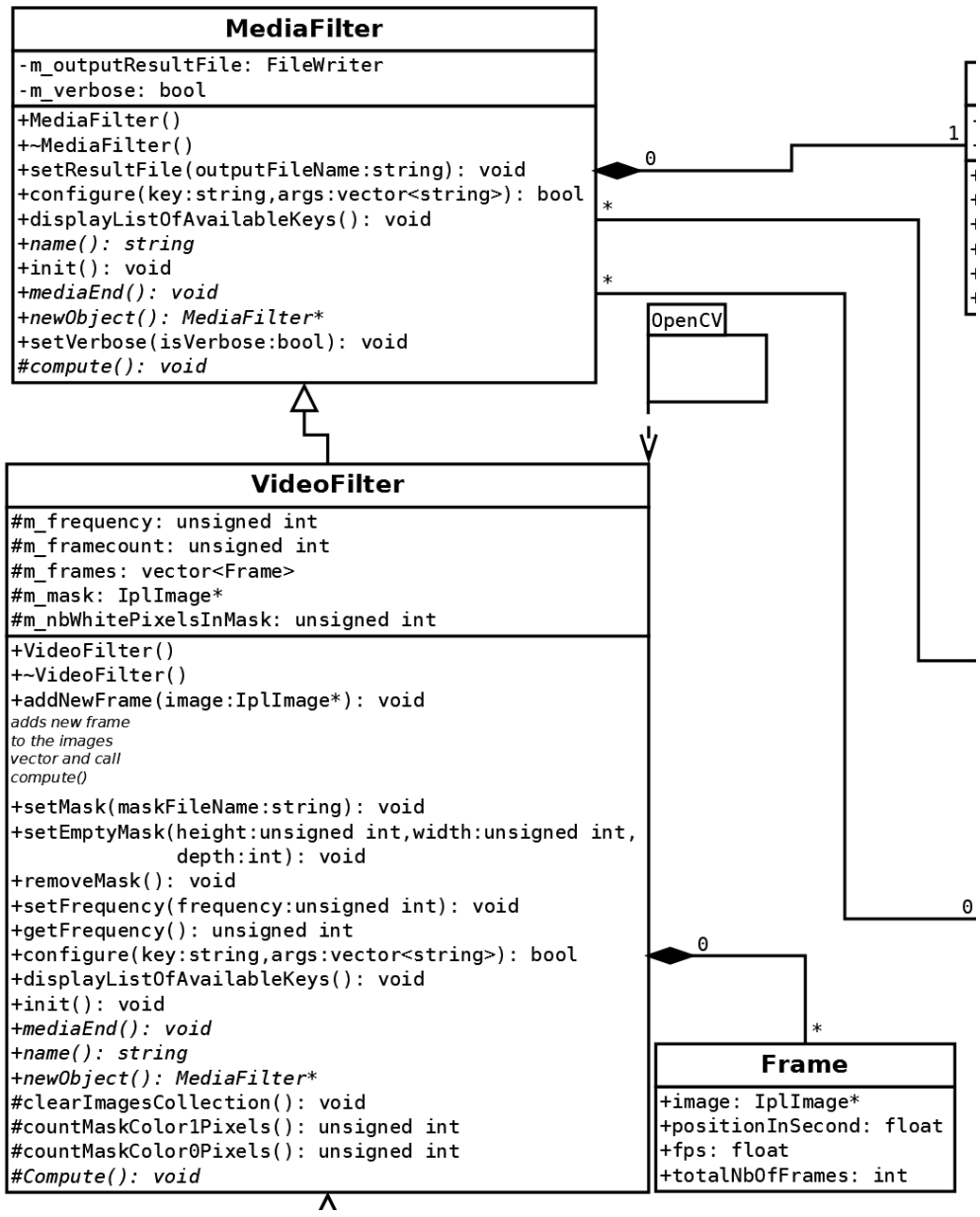
[32] The nature of the configuration file is more thoroughly described in Section 7.1.3.

Media management



The *media management* section of the source code handles everything related to video navigation. This refers to loading a video file and sequentially extracting each frame. Every times a frame is supposed to be processed by a video filter (such as static information, moving information, bar progression, or scene change detection, as discussed in Section 5.3 to Section 5.6 respectively), the media management system informs the matching filter that a new frame is available to be processed (*addnewframe* function of the filter), and also warns each filter when the end of the video file has been reached (*mediaend* function of the filter). More about the communication between *media management* and the filters is discussed in the presentation of the *core engine*.

Media filter system



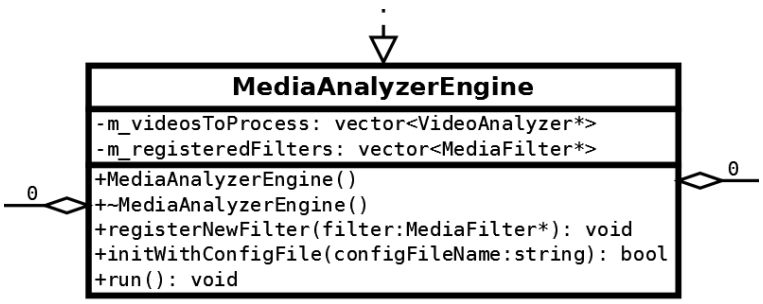
The *media filter system* is a core part of the source code, in the sense that it not only represents the actual audio-visual processing, but also because it is the only section of the source code a programmer may amend in order to add new functionalities (the *file management* and *media management* parts being essentially for simplifying and hiding most of the usual input/output functions). The way a user is supposed to utilise the *media filter* to add new video algorithms

is described more thoroughly in Section 7.1.4 that deals with the creation of new filters.

The *media filter system* relies heavily on the object oriented notion of inheritance (Bruegge & Dutoit, 2010, p. 38; Stroustrup, 1997). The video filter class describes the general mechanism of any video filter, being: having a name (to be recognized through the configuration file), being configurable using a key and list of arguments mechanism (to be configurable using the configuration file), having a mask image system to specify the area of the video to consider (see Section 5.2.4), and knowing its result file name. Each time the *media management* part is calling the *addnewframe* function of a video filter, the video filter will call its own *compute* function, which contains the method to process the video frame. When the *addnewframe* function is called for the first time, the *init* function of the filter is called first, in case some variables need to be initialized first. The *compute* function is also called only if the frame step (see Section 5.2.2) is matching. This represents the global mechanism of a video filter.

Each filter must inherit the video filter class, and specify: 1) what the *compute* and *init* functions should do in the specific case of the actual filter, 2) what is the name of the filter, 3) how it could be configured using a key and arguments format. Filter creation will be discussed in Section 7.1.4.

Core engine



The *core engine* deals with registering the existing filters, initiating the parsing of the configuration file, and then running video processing. Each time a registered filter name is identified in the configuration file, a new instance of the filter is created, and passed to the *media analyzer* object within the *media management* system. This is how *media management* functions to know which filter it is supposed to pass the frames to. In the case where several videos need to be processed, the *core engines* creates as many *media analyzers* as there are videos, dispatching each of them the matching video filters they need to call. It is important to recall that a user should not modify the *core engine*, nor the *media management* or the *file management* parts. These parts exist to automatize and hide most of the complex mechanisms. Ideally, only the *filter* part should be upgraded.

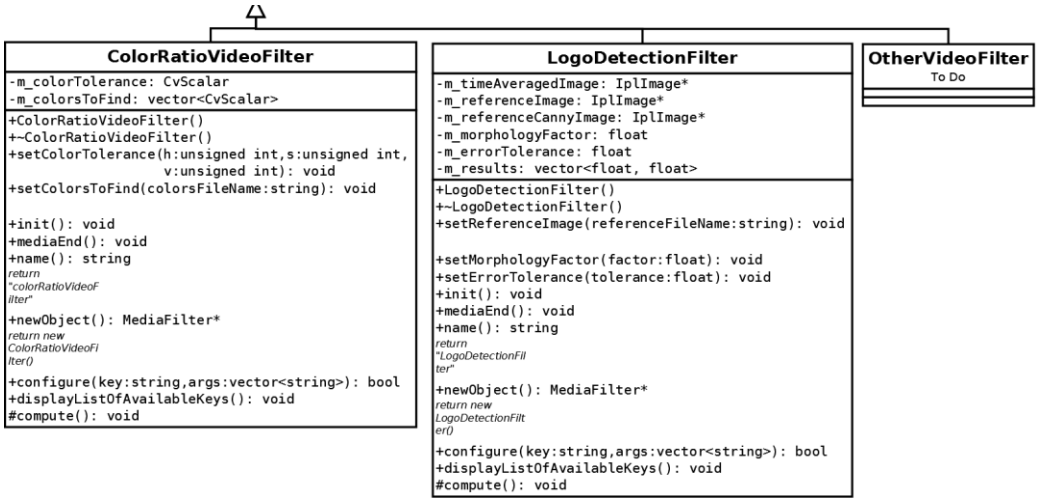
7.1.3 *Using the software*

Before seeing how a programmer can add new video processing functionalities; it is important to present how the program can be used by an analyst through a configuration file. The role of a configuration file is to allow the analyst to make several choices in how they desire to process the gameplay footage without the need to modify or recompile the source code. As presented above, each filter possesses its own name, and its own way to be configured using a "key arguments" format. A configuration file looks like:

```
# A line starting with a # is a comment
# Video to process
VIDEO ..\Battlefield3\J***01\footage.mp4

# Static information detection
FILTER LogoDetectionFilter
RESULTFILE ..\Battlefield3\Results\J01_missionScreen.txt
MASKFILE ..\Battlefield3\Inputs\Images\missionScreen_mask.png
REFERENCEFILE ..\Battlefield3\Inputs\Images\missionScreen.png
TIMETOLERANCE 2
ERRORTOLERANCE 0.33
FREQUENCY 10
VERBOSE 1

# Moving information detection
FILTER MovingLogoDetectionFilter
RESULTFILE ..\Battlefield3\Results\J01_enemyLocation.txt
REFERENCEFILE ..\Battlefield3\Inputs\Images\enemyLocation_logo.png
TIMETOLERANCE 1.
ERRORTOLERANCE 0.44
FREQUENCY 10
VERBOSE 1

# Bar progression
FILTER colorRatioVideoFilter
RESULTFILE ..\Battlefield3\Results\J01_HUD.txt
MASKFILE ..\Battlefield3\Inputs\Images\HUD_mask.png
COLORFILE ..\Battlefield3\Inputs\Images\HUD_color.png
TOLERANCE 25 25 25
FREQUENCY 10
VERBOSE 1
```

A VIDEO key identified by the *file management* section of the code is then interpreted by the *core engine* as acknowledging the existence of a new video to process, thus necessitating the creation of a specific *video analyzer* object (one per video). A FILTER key is understood as necessitating a new filter instance to be added to the last created *video analyzer*. Each remaining key is understood as being related to the last created filter, and the pairs <key arguments> are then passed to the filter for its own parameterization.[33] Once the configuration file has been parsed, the *core engine* sequentially launches each created *video analyzers* (going to the next one when the video is over). The filters are then executed in parallel for each video, but the different video files are sequentially processed.

7.1.4 *Creating a new filter*



In line with the architecture presented above, creating a new filter only requires the programmer to inherit the abstract *video filter* class, and override: 1) the *compute* function to describe how to process the frames; 2) the *init* function that

[33] If the key is not understood by the filter, then the filter display the list of available keys in order to guide the analyst in correcting the configuration file.

sets up the filter before the first *compute* call; 3) the *mediaend* function that describes what should be done once the video is over (such as improving the results, or correctly managing computer memory); 4) the *initialize* function that processes the keys and arguments for configuring the filter; 5) the *name* function that returns the name of the filter (for being recognized using the configuration file); 6) and the *createnewinstance* function that returns an instance of the filter, which is used by the *core engine* to pass a new instance of the filter to the *media analyzer* objects, when necessary. Once the new filter has been created, the remaining action required is to let the *core engine* know about the existence of the newly created filter, through the registration functionality. The main source code file is then very short and easy to upgrade:

```
int main( int argc, const char** argv )
{
   MediaAnalyzerEngine mediaEngine;

   mediaEngine.registerNewFilter(new ColorRatioVideoFilter());
   mediaEngine.registerNewFilter(new LogoDetectionFilter());
   mediaEngine.registerNewFilter(new MovingLogoDetectionFilter());
   mediaEngine.registerNewFilter(new HueRangeByHystogramFilter());
   mediaEngine.registerNewFilter(new SceneChangeDetectionFilter());

   mediaEngine.initWithConfigFile("config.txt");
   mediaEngine.run();

   return EXIT_SUCCESS;
}
```

7.2 Synchronised video presentation

Now that the software system has been presented from a structural perspective (Section 7.1.2), usage perspective (Section 7.1.3) and an upgrade perspective (Section 7.1.4), the following sections focus on different applications of the software, beyond the mono-modal usage already presented in Chapter 5 and Chapter 6, within multi-modal applications.

The first multi-modal application to be presented in Section 7.2 is the use of *synchronised video presentation*, which displays to the analyst different data types, synchronised all together. Using a synchronised video presentation is useful in order to not only identify key moments that can arise from other modalities as well as feedback-based gameplay metrics, but also to study the eventual correlation between different modalities. It should be acknowledged that synchronised video presentation can be seen as a video version of the *biometric storyboard* concept developed by Mirza-Babaei (Mirza-Babaei et al., 2013).

7.2.1 *Synchronising data*

Before creating a synchronised video presentation, it is important to also have data available that is already synchronised; that is, data which time stamps match (or with a negligible delay). During game sessions, presented in Section 2.3, it has been necessary to develop and plan for a system that can record all the different data sets simultaneously. Section 7.2.1 provides an overview of the synchronisation recording system, which can also be seen as a contribution to the research

The different data modalities recorded during the game sessions run to test feedback-based gameplay metrics also included: gameplay footage captured using FRAPS (Beepa, 2013) (for feedback-based gameplay metrics), player facial expression using a High Definition webcam, controller inputs using GlovePie (Carl Kenner, 2010) and biometry using the WildDivine device. In order to limit lag issues, the data were recorded using two different computers: the *game machine*, recording the gameplay footage and the controller inputs (keystrokes and mouse); and the *control machine* containing the biometric recording (Galvanic Skin Response (GSR) and Heart Rate (HR)) plus the webcam. In order to have synchronised the two computers, the scripting software GlovePie was installed on each machine. GlovePie is a software system that is generally used to translate a user input into another input (for instance, turning a mouse click into a F11 key press). Furthermore, GlovePie can also log any controller input (like keystrokes and mouse events), and communicate on the network using the Open Sound Control protocol (OSC) (Wright, Freed, & Momeni, 2003). The OSC protocol is based on the use of UDP messages in the form of URL, such as "/record/start" for example. Using the control machine, once the webcam application and the GSR recorder – developed specifically for the project – are launched, the analyst can press a specific key ("scroll lock" in this case) that initiates the webcam recording, starts the GSR recording, and sends a "/record/start" OSC message to the game machine. Once the game machine receives the "/record/start" message through GlovePie, it initiates the recording of key stroke and emulates a key press that matches the FRAPS key sequences for screen recording. By pressing the same key on the control machine, the process is reversed: GSR and the webcam stop, and a "/record/stop" OSC message is sent

back to the game machine. Because both machines are on the same network, the latency is less than half a second, which is enough for considering the different modalities as synchronised. Figure 127 illustrates the communication between the two machines.

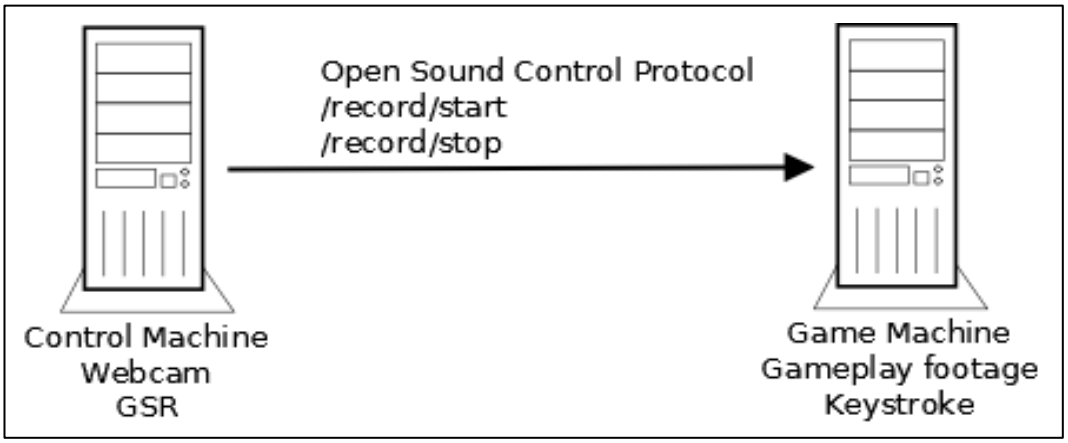


Figure 127. Communication between two machines for synchronisation purposes

Figure 127 illustrates the synchronisation process between the control machine and the game machine. When the researcher press a specific key on the control machine, the webcam and GSR recordings start, and an OSC message "/record/start" is sent to the game machine. The game machine then starts the recording of the gameplay footage, and mouse/keyboard inputs. Another key on the control machine stops the process. The data then starts and ends at the same time, simplifying the synchronisation.

7.2.2 *Synchronised presentation filter*

A specific presentation filter has been developed, able not only to transform curve files (feedback-based gameplay metrics, or GSR curve) into video files (each frame representing the curve with a moving cursor matching the time position of the frame), but also to combine different sub-videos into one. It is then possible to have in the same video the gameplay footage, the player facial expression, several feedback-based gameplay metrics and biometric information displayed together. Because each curve also has a visual cursor matching the video time with the curve time, it is possible to easily fast-forward to a moment of interest as indicated by any modality, and assess what is happening on the gameplay footage at that moment. Figure 128 is an example of synchronised video presentation with

the game *Dead Island*, where a correlation can be found between a GSR spike, health drop and red hue (feedback-based gameplay metrics). The game visual content is immediately accessible in order to ease the interpretation process.

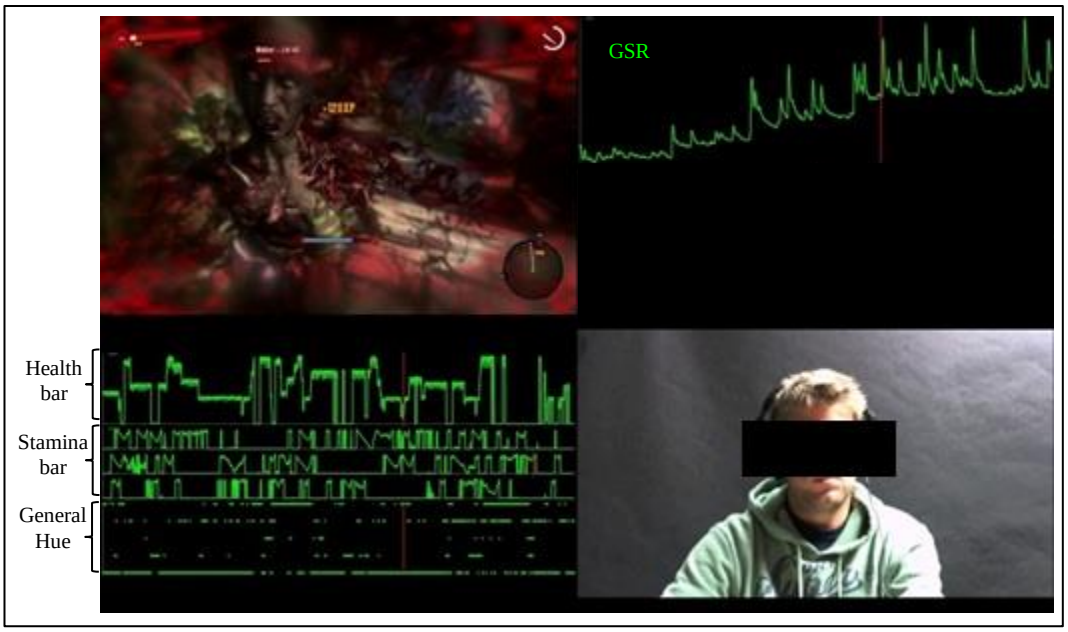


Figure 128. Example of synchronised video presentation using the game *Dead Island*

Figure 128 displays a frame extracted from a synchronised video presentation. By jumping to a sequence identified as mainly red by the Hue metric (bottom left corner), it is possible to asses that the player (bottom right corner) is fighting a zombie (top left corner), and is having a strong bodily reaction at the same time (top right corner).

7.2.3 *Tool for qualitative interviews*

At the end of each series of gameplay sessions (in general during the fifth week, see Section 2.3), an interview was conducted with each participant, during which they were presented with portions of their gameplay footage, and asked to reflect upon them. The synchronised video presentation was used to select the relevant sequences of gameplay before discussing them with players. The synchronised video presentation has proved useful for identifying moments where nothing specific appears to happen during play, but still triggers a strong emotion from the player. For instance, in a specific sequence of *Battlefield 3*, one of the participants finds themselves in an empty hallway, in which nothing particular happens in

terms of action. However at the same time, the player appears really irritated (web cam footage). This sequence has been then selected and presented to the player during the interview session. In this moment, the participant explained that he/she was really agitated because a bug in the game removed his/her favourite weapon. This was first noticed whilst in this corridor. The use of synchronised video presentation has enable identification of key moments for the player through access to a broad context of information that covers a single session. More detail can be found in Marczak et al. (Marczak et al., 2013, 2012), publications that outline the use of synchronised video presentation to contextualise the different produced feedback-based gameplay metrics.

7.3 Controller Input

Another modality that can be used conjointly with feedback-based gameplay metrics to study the player engagement with a game system is the study of the controller inputs. In this instance, *Max Payne 3* has been used as a case study for how the study of mouse clicks can improve understanding of player experience during specific sequences.

Max Payne 3 is a game that mainly relies on the use of 'slow motion' effects.[34] Indeed, several gameplay sequences use slow motion effects; some directly triggered by the player, termed 'bullet-time' and 'shoot dodge' (in which the player needs to press specific keys to activate slow motion); and other instance that are triggered by the game itself, like 'execution sequences' or 'bullet cam'. Figure 129 displays a bullet cam sequence, which is triggered by the game, which occurs when the player has successfully killed a wave of opponents. At this point, the bullet from the last shot is followed by a camera, from the avatar's gun to the impact point on the body.

[34] It is interesting to also note that slow motion effect are also mentioned as potentially harmful in terms of violent contents by classification systems, like the Australian one (Australian OFLC, 2005, p. 5)

Figure 129. Bullet cam in *Max Payne 3*

Figure 129 displays a bullet cam sequence from the game Max Payne 3. After an enemy wave, the bullet from the last shot is followed by a camera, from the avatar's gun to the body impact point. During these sequences, the player can perform two actions with their mouse: slowing down the camera (right click) and shooting more bullets (left click).

During the bullet cam sequence, the player is only allowed to perform two actions (or a change in degree of freedom, see Section 4.1.5): slow down the camera (by holding the right mouse button) or shoot several extra bullets (by clicking on the left mouse button). Regardless of the choices made by the player during a bullet cam sequence, the opponent is always eventually shot. What differentiates the sequences is that it is mostly aesthetic, as slowing down or shooting more bullets mainly have a visual impact. The only impact on future gameplay difference occurs if the player decides to shoot several bullets, they then get fewer bullets once the game resumes. During interviews, some participants commented that during the interview sessions that this was a reason not to use the left click during bullet cam sequences, while others, knowing that bullet cam only appears after successful completion of an enemy wave, explained that they could freely use their ammunition.

The left and right mouse buttons are then core player inputs to analyse in order to assess how the participants decide to engage with the bullet cam sequences. The process requires identifying bullet cam sequences through gameplay performance segmentation (the HUD disappears during bullet cam), and then use the synchronised input recording (see Section 7.2.1) to see how the player engages with the sequences.[35] Figure 130 presents three different bullet cam sequences for four participants. They occurred 1) during the game tutorial (players' first encounter with bullet cam, which comes with instructions and on-screen prompts); 2) after the last enemy of game chapter one has been killed; and 3) after the last enemy of game chapter two has been killed. The black curve shown in Figure 130 represents the HUD presence (1) or absence (0), and the red crosses represent the use of left click (shoot), while the blue line represents the use of right click (slow down the camera). It is important to note that, when the HUD is on-screen, left click is for shooting, and right click is for aiming.

Figure 130 illustrates that participants 1 and 2 follow the tutorial to the letter, doing nothing more, and nothing less, than what the game tutorial ask them to do: shoot, slow down the camera, and shoot again. However, participants 3 and 4 use the opportunity to fire a number of bullets. Participants 1 and 2 then avoid shooting several bullets during the two other bullet cam sequences (no red cross

[35] It is important to acknowledge that the use of slow motion could be seen as a *feedback-based gameplay metric* by studying the amount of motions between two consecutive frames, but the present thesis does not cover the motion approach. Similarly, the shot can be detected through sound analysis, but as the shot sound lasts less than a second, the cross-correlation algorithm presented in Section 5.7 needs to be improved in order to avoid false alarms. Using the mouse clicks detection is then, for now, a much more accurate solution.

while the HUD is off-screen). Participant 1 seems to use the slowdown effect after killing the last enemy of game in chapter two, but this is obviously because they were aiming before (right click) the sequence begins and not releasing the button. Participant 3 is shown to always use the opportunity to empty their gun on an enemy during the bullet cam sequences, while participant 4's use diminishes as the game proceeds. Further interpretation of the relevance of this information when considering the player engagement with violence can be found in Schott et al. (Schott, Marczak, Mäyrä, & Vught, 2013).

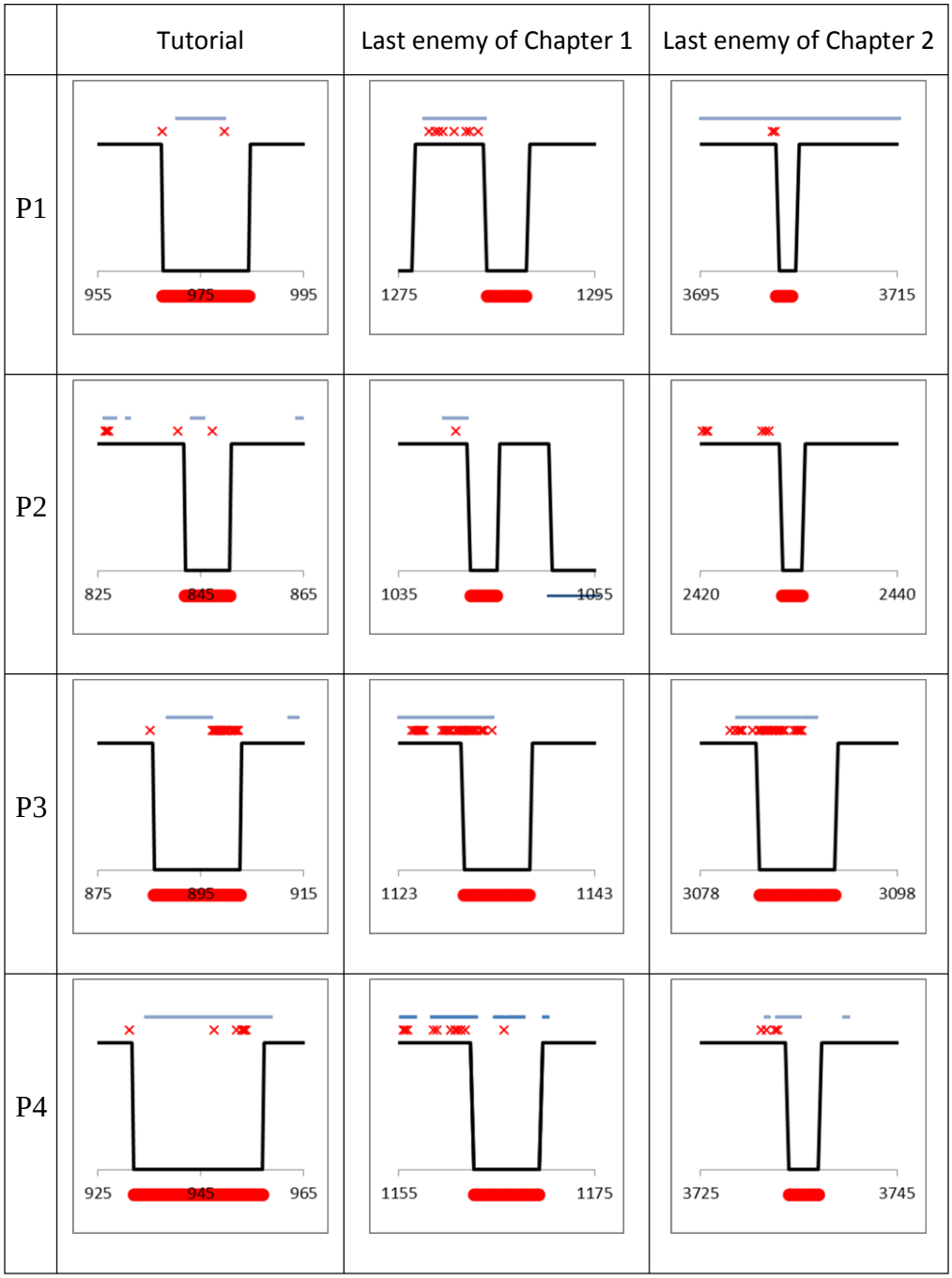


Figure 130. Bullet Cam Activation and Player Engagement

Figure 130 presents different bullet cam sequences with four participants: 1) during the game tutorial; 2) after the last enemy of the game chapter one has been killed; 3) and after the last enemy of the game chapter two has been killed. The black curve represents the HUD presence (1) or absence (0), the red crosses represent the use of left click (shoot), and the blue line represents the use of right click (slow down the camera). The x-axis represents the time in seconds.

Except during the tutorial, because the game prompts them to do so, participants 1 and 2 does not use the interaction possibilities offered by the game during the bullet cam sequences. Participant 3, however, always empties their ammunition during bullet cam sequences. Finally, participant 4 seems to use less and less the slowdown and multi-shot effect as they progress in the game.

7.4 Unresolved biometric spikes: Examining the coverage of feedback-based gameplay metrics

The last example of an application presented in this chapter is related to the discussion presented in Section 7.2. It seeks to identify the reason for spikes in Galvanic Skin Response that seem unrelated to any feedback-based gameplay metrics collected during the same moment. Here instead of using a synchronised video presentation showing simultaneous GSR, feedback-based gameplay metrics, and gameplay footage to explain player experience, the idea was to begin with GSR readings and their relationship with metrics, avoiding the gameplay footage.

The principal goal was to identify key moments for the player as indicated through the detection of GSR spikes. The absolute GSR results from the game session were first translated to a relative curve by considering the differences between a value and its previous one. The positive values are then summed using a six seconds window, thus highlighting the GSR spikes.

The idea that GSR spikes can be correlated with segments of performance was first tested using the game *Dead Island*, where GSR relative curves were placed in parallel with feedback-based gameplay metrics representing drop in the avatar health. Figure 131 displays this result, and highlights the potential of a conjoint analysis for providing insight into gameplay events that cause GSR spikes.

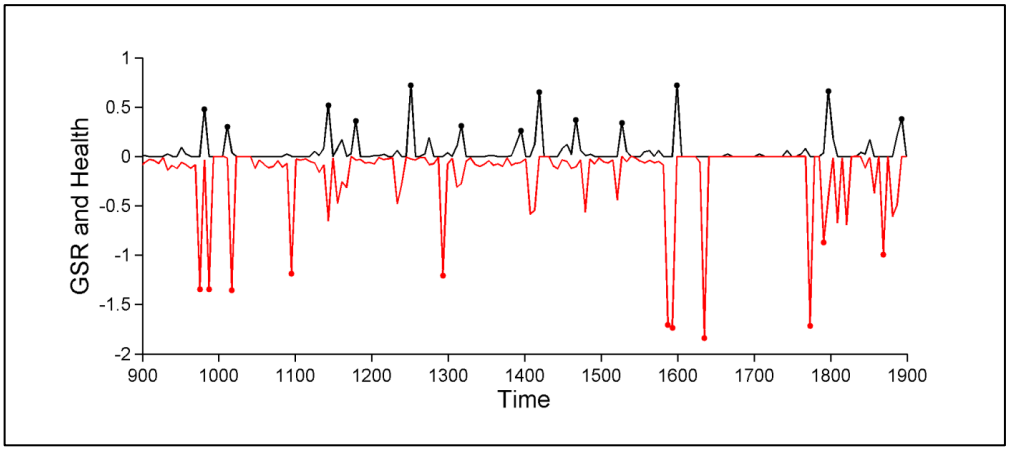


Figure 131. GSR (relative) against health drop (relative) on *Dead Island* performance

Figure 131 illustrates the conjoint analysis of GSR (black curve), and health drop (red curve). It is interesting to note that numerous GSR spikes can be related with strong health drop, probably indicating a player frustration. The x-axis represents the time in seconds.

7.4.1 *Storyboard visualization filter*

A special video filter (see Section 7.1.4) was developed to automatically generate a storyboard using several frames from a specified time stamp. This resulted in ten frames from five seconds before the time stamp to five second after the time stamp being produced automatically. Figure 132 displays an example of storyboard, using the game *Battlefield 3*. The idea was to generate storyboards for each of the major GSR spikes, so that the analyst could review what is happening without the video context. The analyst is then made aware of what is happening during the narrow time frame surrounding the spike. The GSR spike generated storyboard in Figure 132 can be understood as a death sequence.

Figure 132. Storyboard example using a performance from *Battlefield 3*

Figure 132 displays an example of storyboard generated (using the storyboard visualization filter) around a GSR spike with the game Battlefield 3. This GSR spike can be seen as being related to a death sequence.

7.4.2 *A GSR/Feedback-based gameplay metric loop*

The different generated storyboards were reviewed and assessed to evaluate whether they can be related to a specific gameplay concept (death, loading screen, quick time events etc.). For instance, Figure 132 clearly displays a death sequence that can also be processed using feedback-based gameplay metrics based on the sound stream (as presented in Section 5.7). The matching metric is generated, and all the correlated GSR spikes with the metric are discarded. This then left some unresolved spikes. By examining the storyboards again in order to try to identify a gameplay concept, the loop is repeated until no new gameplay concept can be identified. The remaining storyboards are considered as totally unresolved spikes. Figure 133 is a diagram schematizing the loop full process.

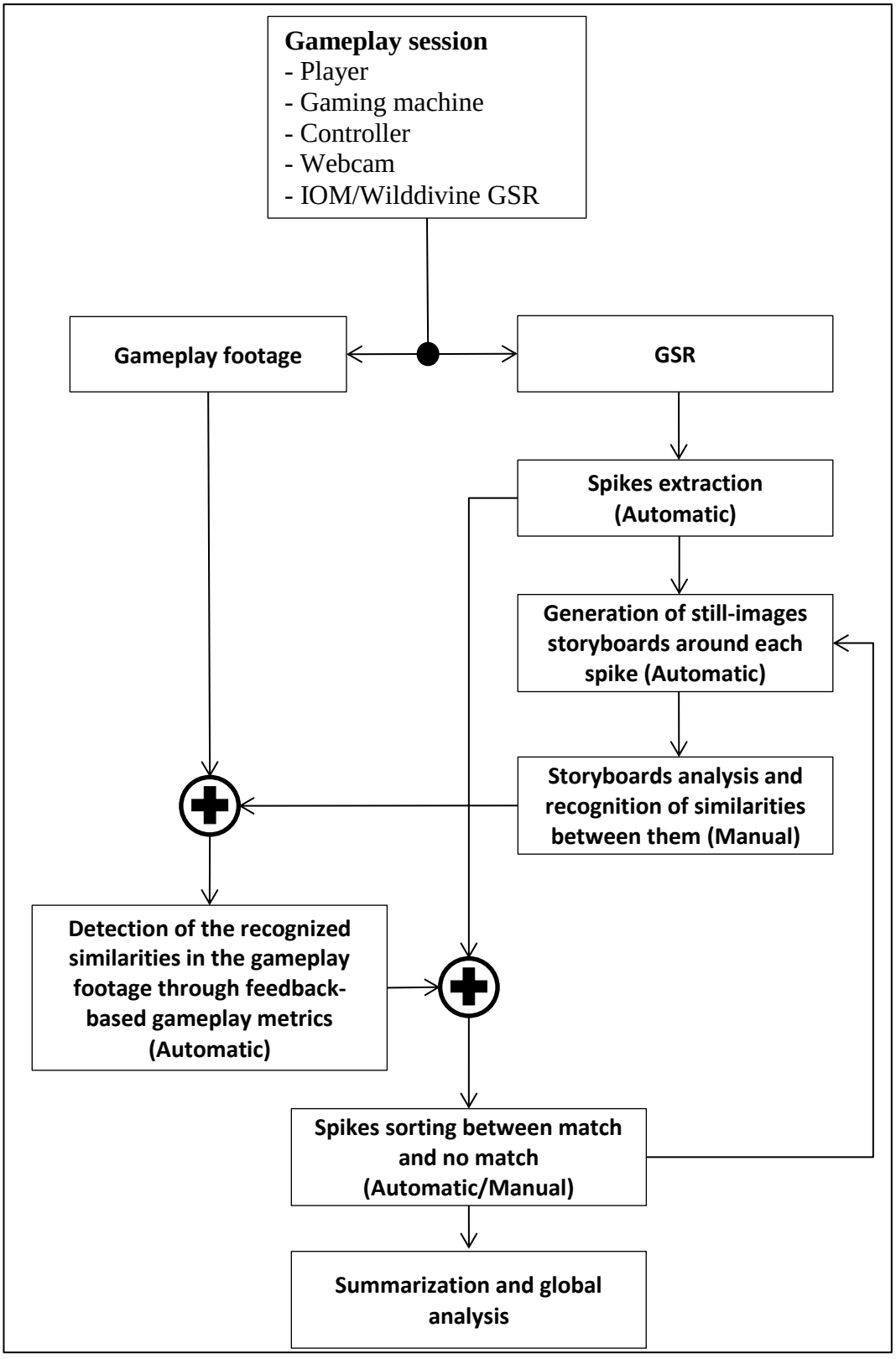


Figure 133. GSR/Feedback-based gameplay metric loop

Figure 133 is a diagram illustrating the GSR/Feedback-based gameplay metric loop. The GSR is considered, and the key moments (spikes) are selected. Then, storyboards are generated for each key moments, and analyzed to see if gameplay similarities can regroup some of them. The matching feedback-based gameplay metric is generated, and all the correlated spikes are discarded. The unresolved spikes are then studied again, etc. The process is repeated until no gameplay concept can be found by only looking at the storyboards.

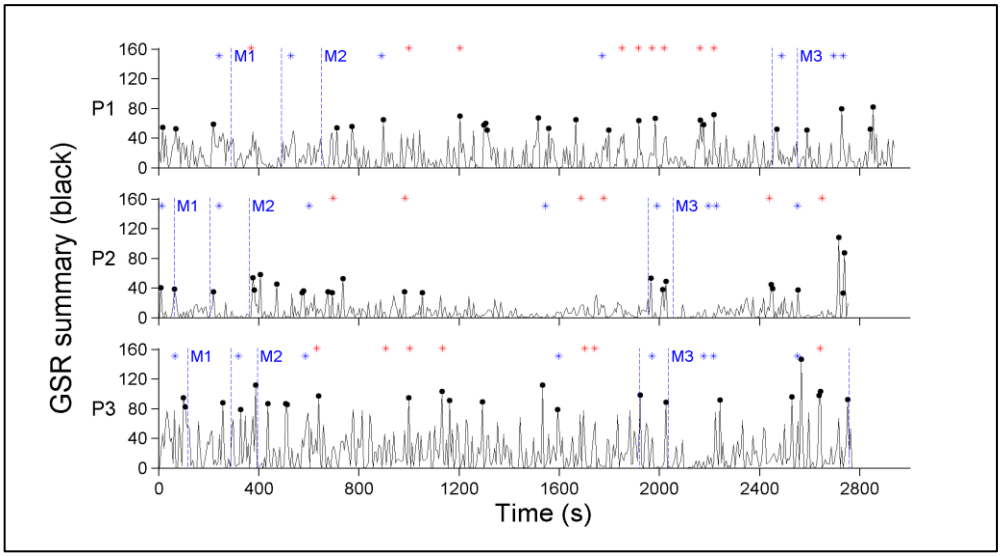


Figure 134. GSR spikes against key moments using three performances of *Battlefield 3*

Figure 134 displays a result using the game Battlefield 3, and three participant. The black dots represent the main GSR spikes. The blue annotations represent key moments in Battlefield 3, detected by using video processing (M1, M2, M3 are the beginning of each mission, the asterisks preceding M1, M2 or M3 is the cut-scene introducing the mission, and the remaining ones represent Quick Time Event). The red crosses represent deaths, detected by using sound processing. Once the spikes matching these events are discarded, the unresolved ones can be assessed through the analysis of their storyboard. The x-axis represents the time in seconds.

Figure 134 displays results for the game *Battlefield 3*. The black line represents the relative GSR curves generated using the method presented in Section 7.4.1. The black dots represent the major spikes. The red crosses represent death sequences. The blue annotations represent key moment of *Battlefield 3* identified using feedback-based gameplay metrics. The deaths and key moments have been decided of interest after a first loop of storyboards review. M1, M2 and M3 represent the beginning of new mission; the asterisk just before each M1, M2 and M3 is the beginning of the mission introduction cut-scene; the asterisks in between represent quick time events. Once the spikes matching these events are discarded, it was possible to have a closer look at the unresolved GSR spikes through analysing the remaining storyboards. A more complete presentation of the interpretation of this process can be found in Schott et al. (2014) which addresses moments of player excitement that fail to match preconceived ideas as to the

pleasures of playing violent war based games. A presentation of the algorithm used to detect the GSR spikes can be found in the same paper.

An example of an unresolved GSR spikes is illustrated in Figure 135. Here, there is nothing on screen that seems to be indicative of an obvious gameplay explanation for a GSR spike. However, the study of the storyboard revealed teammates in *Battlefield 3* were building a bridge by using a wooden plank. The player then crosses the roof and progresses in the game. The GSR spike can then be connected with player excitement in this unusual and quite unique moment that may not get picked up in a manual observation of gameplay. Several unresolved spikes were also linked with in-game mission briefing and anticipatory moments that are not easily identifiable without GSR.

Figure 135. Unresolved GSR spikes: plank/bridge sequence

Figure 135 displays an unresolved storyboard after the loop process, in the game Battlefield 3. On this storyboard, the teammates are building a bridge, for helping the player finally progressing further in the game world. The GSR spike can be connected to player excitement to finally being offered to explore the rest of the mission map.

7.5 Conclusion

This chapter has presented the development of the software system (Section 7.1) underlying the method, and described exploratory applications of feedback-based gameplay metrics within a multi-modal approach to measurement (Sections 7.2 to 7.4). In this sense, it is hoped that Chapter 7 has established a connection between the novel gameplay metric system based on feedback streams (Chapter 2), the

concept of gameplay performance segmentation (Chapters 3 to 6), and eventual future application in explanation of player experience. It is interesting however to note that the multi-modal measures presented in Chapter 7 also offer a connection with the videogame classification systems, as presented in Chapter 1. The synchronised video presentation is a tool that is able to provide immediate feedback to a classification officer on not just screen content, but also other modalities of interest, in consideration of the impact on the player and 'dominant effect" of the game. As shown in Schott et al. (2014), the pleasure of the experience might not always be influenced by element of the game connected to its core narrative (i.e., the war). Also, the study of player's inputs (in conjunction with slow motion sequences) detected using feedback-based gameplay metrics is a tool that informs about how a player actually engages with objectionable visual content.

All the three applications are still exploratory and far from being able to make any definitive conclusions or critiques of about existing classification approaches, but the existence of the software system presented in Chapter 7 makes it possible for better evidence to be generated on player experiences with games that attract the interest of game classification under their assessment criteria.

Chapter 8

Conclusion

The research presented in this thesis was triggered by a need to be able to assess *player experiences* with games. While this need is justifiable as a general scholarly interest, the thesis was developed in response to a specific call from the New Zealand classification office, who wish to see research that:

> Explore[s] the extent to which public's perception of causal links between game playing and various social ills … [might be] moderated or even undermined by [knowledge of] how players actually respond to and negotiate their way through the content and characteristic of the medium (New Zealand OFLC, 2009).

This need led to a focus on both *player performance*, as an account of the manner in which players engage with a game system, and player experience, that addresses how players experience the performance. The notion of *gameplay performance* operationalised in the current thesis endeavours to reflect the way that games are actually analysed in accordance with Aarseth definition of games, as "both an object and process that must be played, [as] playing is integral, not coincidental" (Aarseth, 2001)

While *gameplay metrics* have been proven to be an efficient method for measuring the core constituents of a play performance (the player, the game and the context), the rationale for the present thesis arose from the way that this

method is not applicable to any off-the-shelf games. Furthermore, the need to assess player experience rather than *playability* moves the focus of the present thesis beyond the sole assessment of design decisions and principles (the main focus of existing applications of gameplay metrics). A method was required to address the life of game once it has been released to the public.

Prior to investigating measurement of player performance, it was necessary to structure both processing and analysis. To achieve this, a conceptual model was developed, based on a literature review that covered core research topics linked to the understanding of player experience, but also extended into linguistics in order to identify how game footage might be best segmented and its content indexed. Computer science research literature was also surveyed to explore the potential value of existing 'processing' algorithms.

A multi-layered model was then produced, incorporating input from these three areas. By acknowledging the need for several *controlled-vocabularies* (linguistic) to guide performance *segmentation* (linguistic) and by identifying coherent segments of play in terms of experience (game studies), the multi-layered model possesses a dualistic function as a step-by-step guide for automatic processing (computer science) and a means of interpreting gameplay (game studies). This conceptual framework presents researchers, interested in exploring player experience in the future, with a structure that is able to contextualise the analytical focus of their research. Five layers compose this model: game system layer, game world instances layer, spatial and temporal layer, degree of freedom layer and interactions layer. These layers are first used hierarchically to deconstruct a

performance, contextualising at each level the automatic processing of each sub-layer, in order to then be used transversally after the deconstruction, summarizing a performance of play, and allowing for a better understanding of the experience of play. As a hierarchy, the conceptual framework is able to account for game systems as holistic entities, whilst also enabling the researcher to drill down to specific moments in which the player activated the game. It is therefore possible to map any research focus transversally, in terms of its proximity and relationship to both macro and other micro components accounted for by the framework. The validity of deconstruction and reconstruction has been demonstrated by examining several performances based on different games.

As a new variant of an existing methodology, f*eedback-based gameplay metrics* represents a rethink of how *gameplay metrics* can be achieved. For the first time, audio-visual processing algorithms originating from *computer science* have been applied in analysis of play performance in the context of game studies research (see discussion of the applications of the method later in this chapter). This approach serves to quantify elements of gameplay as it is actually experienced and observed by players through two main feedback streams of moving-image media. This methodology represents a highly significant contribution to knowledge in terms of how it is achieved, that is, through automatic processing of the audio-visual content of videogames, it is possible to process any off-the-shelf game, whilst being able to focus on the actual player experience rather than playability. The post-processing aspect of feedback-based gameplay metrics also means that the possibilities of analysis remain open to researchers, who can now go back and re-examine a performance from different angles when new needs are identified.

Moreover, this methodology focusses on capturing and summarizing the exact moment of the play. The methodology can then be used to assist more qualitative methods, such as interviews, which are mainly based on the player's own memories and interpretation of the performance (when the interview occurs after the play), or can be disruptive of the play (when the interview is conduction during the play). By using the gameplay performance summary produced through feedback-based gameplay metrics and gameplay performance segmentation, the researcher can now describe the actual moment of play, thus guiding research, whether quantitative or qualitative.

By quantifying audio-visual content, the method provides empirical basis for academic arguments widening the impact of *game studies*, which has so far been a predominantly theoretical accounts provided by player analysts. The methodology also holds potential for future research within the *computer sciences*, as the context of this research provides a test-bed for the validation of other existing audio-visual processing methods and algorithms. To assist future researchers in using this method, the different audio and video algorithms presented in this thesis will be soon published as an open-source contribution. A user manual guiding the utilisation of the software system, as well as a more user-friendly graphical interface, are core future research works to consider in order to ease the application of the method in the game studies community.

This thesis establishes several connections between game studies, computer science and the classification systems (cf. Figure 136). In terms of end users, by bringing *computer science* to the study of gameplay footage, the methodology

becomes usable within the game studies community, by not only widening the range of games on which gameplay metrics can be gathered (i.e., considering any off-the-shelf games), but also allowing any form of gameplay footage, even those produced from previous studies, to be post-processed in order to generate new quantitative data about the player's interactions with game systems. Post-processing has so far been absent from standard gameplay metrics (that are recorded in real-time).

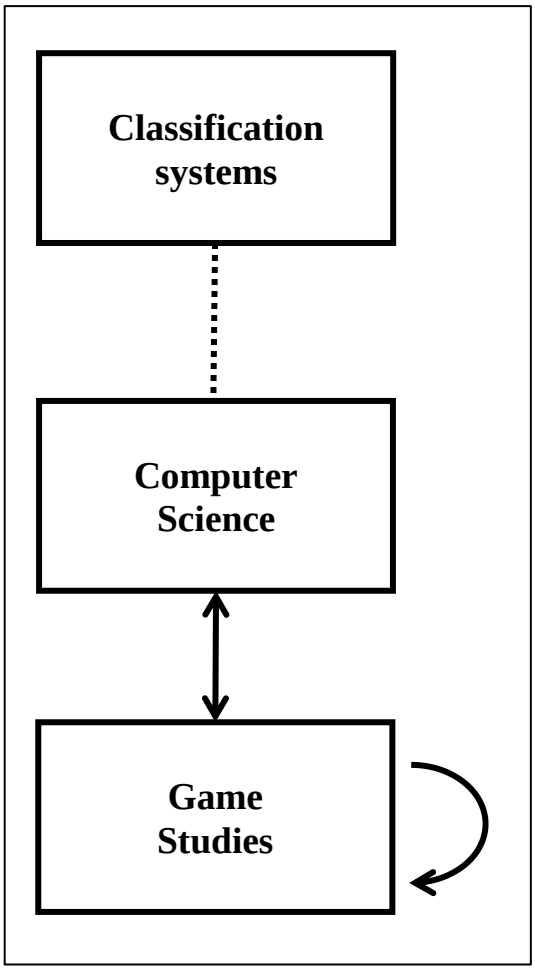


Figure 136. Connections built by the present thesis

The strength of the conceptual model presented in this thesis also resides in that even though it has been successfully used to validate the feedback-based gameplay metrics methodology as a core contribution of the thesis, the gameplay

performance segmentation and multi-layered model can also be employed with other types of metric, notably the more standardly used gameplay metrics.

The study of gameplay footage is still underdeveloped within the field of computer science. There are some notable exceptions, such as the study of visual attention in games (Peters & Itti, 2008), which estimates the attentional focus of the player when watching the screen, by analysing the most salient content colour, shape and movement. It is hoped that this thesis demonstrates to computer science researchers, interested in automatic audio-visual processing, that they can now profit from using gameplay footage to gather useful data on human-computer interaction. The amount of information, symbolic and diegetic, obtained from audio-visual footage also makes gameplay footage a rich and useful document to consider for strengthening the validity of current, and future, audio-visual processing algorithms.

The present thesis demonstrates that computer science and game studies can be combined effectively together to assess player experience. By doing so, this thesis has been able to detect key moments that are considered objectionable by game classification criteria, such as close-combat fighting (by detecting the loss of health using the bar progression algorithm), slow motion (by detecting the absence of Head Up Display using the static information detection algorithm) or blood present on screen (by detecting red frames using the hue identification algorithm). As assessing videogame classification systems requires more than a single approach (Schott, Vught, & Marczak, 2012), the present thesis illustrates how multi-modal approaches can be applied successfully to the study of player

experience with objectionable content. This approach produced synchronised video outputs that presented the end-user with a visualisation of data (Marczak et al., 2013, 2012), that allowed an assessment of player's actual engagement with specific contentious sequences of play (Schott et al., 2013), and explore causes of player excitement during play (Schott et al., 2014).

The thesis also paves the way for future contributions and connections. The source code is ready for the addition of new audio-visual processing methods. The methods described in this thesis are generic enough to account for a large variety of gameplay concepts. However, it could be appropriate, in the near future, to develop algorithms more oriented to the detection of videogame classification criteria, such as violent content in video (de Souza, Chávez, do Valle, & De A Araujo, 2010) or sound streams (Giannakopoulos, Kosmopoulos, Aristidou, & Theodoridis, 2006). This can be easily added to the current set of algorithms by implementing new filters in the existing source code. The source code, with its filter system, can be considered as a platform, which can be continuously updated to match the various needs of different researchers. It is also expected that the conceptual model will be adapted to the needs of future research work. To accommodate change in game technology, the method can also be adapted for multiple screens (useful for game consoles with several screens, or for virtual reality systems). However, the best solution for being adaptable to any future technology might be to focus more and more on sound. Indeed, regardless eventual strong changes in game technology (for instance the use of holograms), the sound stream broadcast to the player is likely to remain the easiest to capture. The haptic feedback could also later be included into the model, through the use

of sensors applied on the game controllers. The five layers described in this thesis correspond to different levels of granularity used for segmenting a gameplay performance. But this model is ready to be amended in terms of granularity detail. It is possible to foresee the removal or addition of new layers to match analysts' needs. Thus the conceptual model, as well as the methodology, are both generic and flexible enough to be tailored to match the focus of future research into player experience.

References

Aarseth, E. (2001). Computer Game Studies, Year One. *Game Studies*, *1*(1).

Adhikari, P., Gargote, N., Digge, J., & Hogade, B. G. (2008). Abrupt Scene Change Detection. *World Academy of Science, Engineering and Technology*, *18*, 711–716.

Australian OFLC. (2005). *Guidelines for the Classification of Films and Computer Games 2005*.

Bay, H., Ess, A., Tuytelaars, T., & van Gool, L. (2008). SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding (CVIU)*, *110*(3), 346 – 359.

Belkin, N. J. (1984). Cognitive Models and Information Transfer. *Social Science Information Studies*, *4*(2-3), 111–129.

Belkin, N. J., & Croft, W. B. (1992). Information Filterting and Information Retrieval: two sides of the same coin? *Communications of the ACM - Special Issue on Information Filtering*, 29–38.

Benois-Pineau, J., Precioso, F., & Cord, M. (2012). *Visual Indexing and Retrieval*. Springer.

Boole, G. (1848). The Calculus of Logic. *Cambridge and Dublin Mathematical Journal*, *3*, 183–98.

Booth, M. (2009). *The AI Systems of Left 4 Dead*. Presented at the Artificial Intelligence and Interactive Digital Entertainment Conference, Stanford. Retrieved from http://www.valvesoftware.com/publications/2009/ai_systems_of_l4d_mike_booth.pdf

Bruegge, B., & Dutoit, A. H. (2010). *Object-Oriented Software Engineering* (Third Edition.). Prentice Hall.

Byron, T. (2008). *Safer Children in a Digital World: the Report of the Byron Review*.

Calleja, G. (2011). *In-Game: From Immersion to Incorporation*. Cambridge, MA: The MIT Press.

Canny, J. (1986). A Computational Approach to Edge Detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *PAMI-8*(6), 679–698.

Canossa, A., & Drachen, A. (2009a). Patterns of Play: Play-Personas in User-Centred Game Development. In *Breaking New Ground: Innovation in Games, Play, Practice and Theory*. London: Brunel University. Retrieved from http://www.digra.org/dl/display_html?chid=09287.49165.pdf

Canossa, A., & Drachen, A. (2009b). Play-Personas: Behaviours and Belief Systems in User-Centred Game Design. In *Human-Computer Interaction – INTERACT 2009* (pp. 510–223). Uppsala, Sweden. doi:10.1007/978-3-642-03658-3_55

Canossa, A., Drachen, A., & Rau Møller Sørensen, J. (2011). Arrrgghh!!!: Blending Quantitative and Qualitative Methods to Detect Player Frustration. In *Proceedings of the 6th International Conference on Foundations of Digital Games* (pp. 61–68).

Chai, W. (2006). Semantic Segmentation and Summarization of Music. *IEEE Transactions on Audio, Speech, and Language Processing*.

Chasanis, V., Kalogeratos, A., & Likas, A. (2009). Movie Segmentation into Scenes and Chapters Using Locally Weighted Bag of Visual Words. In *Proceedings of the ACM International Conference on Image and Video Retrieval*.

Choi, F. Y. Y. (2000). Advances in Domain Independent Linear Text Segmentation. In *NAACL 2000 Proceedings of the 1sr North American chapter of the Association for Computational Linguistics Conference* (pp. 26–33).

Cooper, A., Reimann, R., & Cronin, D. (2007). *About Face 3: The Essentials of Interaction Design*. Wiley.

Cooper, M. L., & Foote, J. T. (2001). Scene boundary detection via video self-similarity analysis (Vol. 3, pp. 378–381). Presented at the Image Processing, 2001. Proceedings. 2001 International Conference on.

Cousins, B. (2004). Elementary Game Design. *Develop Magazine*, (October 2004), 51–54.

Csikszentmihaly, M. (1990). *Flow: The Psychology of Optimal Experience*. Harper Perennial.

Davenport, G., Smith, T. A., & Pincever, N. (1991). Cinematic Primitives for Multimedia. *Computer Graphics and Applications, IEEE*, *11*(4), 67–74.

Derosa, P. (2007). Tracking Player Feedback To Improve Game Design. Retrieved from http://www.gamasutra.com/view/feature/129969/tracking_player_feedback_to_.php

De Souza, F. D. M., Chávez, G. C., do Valle, E. A., & De A Araujo, A. (2010). Violence Detection in Video Using Spatio-Temporal Features. In *Proceedings of 2010 Conference on SIBGRAPI* (pp. 224 – 230).

Diakoff, H. (2004). Database indexing: yesterday and today. *The Indexer*.

Drachen, A. (2008). Crafting User Experience via Game Metrics Analysis. Presented at the NORDICHI 2008, Lund, Sweden. Retrieved from http://www.cs.uta.fi/~ux-emotion/submissions/Tychsen.pdf

Drachen, A., & Canossa, A. (2008). Defining Personas in Games Using Metrics (pp. 73–80). Presented at the Future Play, New York, USA. doi:10.1145/1496984.1496997

Drachen, A., & Canossa, A. (2009a). Analyzing spatial user behavior in computer games using geographic information systems. In *Everyday Life in the Ubiquitous Era* (pp. 182–189). New York, NY, USA. doi:10.1145/1621841.1621875

Drachen, A., & Canossa, A. (2009b). Towards gameplay analysis via gameplay metrics. In *Everyday Life in the Ubiquitous Era* (pp. 202–209). New York, NY, USA. doi:10.1145/1621841.1621878

Drachen, A., & Canossa, A. (2011). Evaluating motion: spatial user behaviour in virtual environments. *International Journal of Arts and Technology*, *4*(3), 294–314. doi:10.1504/IJART.2011.041483

Drachen, A., Canossa, A., & Yannakakis, G. N. (2009). Player Modeling Using Self-Organization in Tomb Raider: Underworld. Presented at the Computational Intelligence and Games. CIG 2009. IEEE Symposium on. doi:10.1109/CIG.2009.5286500

Drachen, A., Seif El-Nasr, M., & Canossa, A. (2013). Game Analytics - The Basics. In *Game Analytics*. Retrieved from http://andersdrachen.files.wordpress.com/2012/08/thebasics.pdf

Eskelinen, M. (2001). The Gaming Situation. *Game Studies*, *1*(1). Retrieved from http://www.gamestudies.org/0101/eskelinen/

ESRB. (2011). The Elder Scrolls V: Skyrim. Retrieved from http://www.esrb.org/ratings/synopsis.jsp?Certificate=31575&Title=The%20Elder%20Scrolls%20V%3A%20Skyrim

Fagerholt, E., & Lorentzon, M. (2009). *Beyond the HUD - User Interfaces for Increased Player Immersion in FPS Games* (Master's Thesis). Göteborg : Chalmers tekniska högskola. Retrieved from http://publications.lib.chalmers.se/records/fulltext/111921.pdf

Fidel, R. (1991). Searchers' Selection of Search Keys: II. Controlled Vocabulary or Free-Text Searching. *Journal of the American Society for Information Science*.

Fitzpatrick, J. M., Hill, D. L. G., & Maurer, Jr., C. R. (2000). *Handbook of Medical Imaging*. SPIE PRESS.

Foote, J. T., & Cooper, M. L. (2003). Media Segmentation using Self-Similarity Decomposition. In *Proceedings SPIE Storage and Retrieval for Multimedia Databases* (pp. 67–75).

Frey, A., Marie, C., Prod'Homme, L., Timsit-Berthier, M., Schön, D., & Besson, M. (2009). Temporal Semiotic Units as Minimal Meaningful Units in Music? An Electrophysiological Approach. *Music Perception, 26*(3), 247–256.

Fujishima, T. (1999). Realtime Chord Recognition of Musical Sound: a system using Common Lisp Music. In *Proceedings - ICMC 1999* (pp. 464–467).

Gagné, A. R., Seif El-Nasr, M., & Shaw, C. D. (2012). Analysis of telemetry data from a real-time strategy game: A case study. *Computers in Entertainment (CIE) - Theoretical and Practical Computer Applications in Entertainment*, *10*(3). doi:10.1145/2381876.2381878

Garshol, L. M. (2004). Metadata? Thesauri? Taxonomies? Topic Maps! Making Sense of it all. *Journal of Information Science*.

Giannakopoulos, T., Kosmopoulos, D., Aristidou, A., & Theodoridis, S. (2006). Violence Content Classification Using Audio Features. In *Proceedings of Helenic Conference on AI 2006* (pp. 502–507).

Gonzales, R. C., & Woods, R. E. (2007). *Digital Image Processing* (3rd ed.). Prentice Hall.

Grimes, J. E. (1976). *The Thread of Discourse*. The Hague: Mouton.

Grimshaw, M., Lindley, C. A., & Nacke, L. E. (2008). Sound and Immersion in the First-Person Shooter: Mixed Measurement of the Player's Sonic Experience. In *Proceedings - 2008 Audio Mostly*.

Halliday, M. A. K., & Hasan, R. (1976). *Cohesion in English*. Longman.

Hanjalic, A., Lagendijk, R. L., & Biemond, J. (1999). Automatically Segmenting Movies into Logical Story Units. *Visual Information and Information Systems, Lecture Notes in Computer Science*, *1614*, 229–236.

Hanna, P., Robine, M., & Ferraro, P. (2008). Visualisation of Musical Structure by Applying Improved Editing Algorithms. In *Proceedings of the 2008 International Computer Music Conference*.

Hasan, Y., Bègue, L., Scharkow, M., & Bushman, B. J. (2013). The more you play, the more aggressive you become: A long-term experimental study of cumulative violent video game effects on hostile expectations and aggressive behavior. *Journal of Experimental Social Psychology, 49*.

Hassenzahl, M. (2013). User Experience and Experience Design. In M. Soegaard & R. F. Dam (Eds.), *Encyclopedia of Human-Computer Interaction* (2nd ed.). The Interaction Design Foundation.

Hausken, L. (2004). Textual Theory and Blind Spots in Media Studies. In M.-L. Ryan (Ed.), *Narrative across Media: The Languages of Storytelling*. University of Nebraska Press.

Hedden, H. (2008). Controlled vocabularies, thesauri, and taxonomies. *The Indexer*, *26*, 33–34.

Hilbert, D. M., & Redmiles, D. F. (2000). Extracting usability information from user interface events. *ACM Computing Surveys (CSUR)*, *32*(4), 384–421. doi:10.1145/371578.371593

Hoiem, D. E., & Sullivan, K. D. (1994). Designing and using integrated data collection and analysis tools: challenges and considerations. *Behaviour & Information Technology*, *13*(1-2), 160–170. doi:10.1080/01449299408914595

Houlette, R. (2004). Player Modeling for Adaptive Games. *AI Game Programming Wisdom II*, 557–566.

Huang, C.-L., & Liao, B.-Y. (2001). A robust scene-change detection method for video segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, *11*(12), 1281 – 1288.

Huizinga, J. (1955). *Homo Ludens: A study of the Play Element in Culture*. Boston, MA: The Beacon Press.

IJsselsteijn, W. A., Hoogen, W. M. van den, Klimmt, C., Kort, Y. A. W. de, Lindley, C. A., Mathiak, K., … Vorderer, P. (2008). Measuring the Experience of Digital Game Enjoyment. In *Proceedings of Measuring Behavior 2008*.

ISO. (2008). ISO 9241-210: Ergonomics of human-system interaction.

Jain, A. K. (1986). *Fundamentals of Digital Image Processing*. Prentice-Hall.

Joosten, E., Van Lankveld, G., & Spronck, P. (2010). Colors and Emotions in Video Games. In *Proceedings of GAME-ON 2010* (pp. 61–65).

Jørgensen, K. (2006). On the Functional Aspects of Computer Game Audio. In *Proceedings - Audio Mostly Conference*.

Juul, J. (2003). The Game, the Player, the World: Looking for a Heart of Gameness. In *Level Up: Digital Games Research Conference Proceedings* (pp. 30–45). Utrecht: Utrecht University. Retrieved from http://www.jesperjuul.net/text/gameplayerworld/

Kim, J. H., Gunn, D. V., Schuh, E., Philips, B., Pagulayan, R. J., & Wixon, D. (2008). Tracking real-time user experience (TRUE): a comprehensive instrumentation solution for complex systems (pp. 443–452). Presented at the SIGCHI Conference on Human Factors in Computing Systems, New York, USA. doi:10.1145/1357054.1357126

Klement, S. (2002). Open-system versus closed-system indexing, A vital distinction. *The Indexer*.

Laurel, B. (1993). *Computers as Theatre*. Addison-Wesley Professional.

Leise, F. (2008). Controlled vocabularies: an introduction. *The Indexer*.

Lin, D. (1998). An Information-Theoretic Definition of Similarity. In *Proceedings of the 15th International Conference on Machine Learning* (pp. 296–304).

Marczak, R., Schott, G., Hanna, P., & Rouas, J.-L. (2013). Feedback-Based Gameplay Metrics. In *Proceedings of FDG 2013*.

Marczak, R., Vught, J. V., Schott, G., & Nacke, L. E. (2012). Feedback-based gameplay metrics: measuring player experience via automatic visual analysis. In *Playing the System*. Auckland, New Zealand. doi:10.1145/2336727.2336733

Martin, B., Hanna, P., Robine, M., & Ferraro, P. (2011). Indexing Musical Pieces Using Their Major Repetition. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries* (pp. 153–156). Ottawa, Canada.

Martinez, H. P., & Yannakakis, G. N. (2011). Mining Multimodal Sequential Patterns: A Case Study on Affect Detection. In *Proceedings of the 13th International Conference in Multimodal Interaction*. Alicante.

McCutcheon, S. (2009). Keyword vs controlled vocabulary searching: the one with the most tools wins. *The Indexer*.

McEwan, G., Gutwin, C., Mandryk, R. L., & Nacke, L. E. (2012). Analysing Social Metrics in an Online Game Site. In *In GRAND 2012*.

Mellon, L. (2009). *Applying Metrics-driven Development to MMO Costs and Risks* (Tech. Rep.). Versant Corporation. Retrieved from http://maggotranch.com/MMO_Metrics.pdf

Milstead, J. L. (1984). *Subject access systems : alternatives in design*. Orlando : Academic Press.

Mirza-Babaei, P., Nacke, L. E., Gregory, J., Collins, N., & Fitzpatrick, G. (2013). How Does It Play Better? Exploring User Testing and Biometric Storyboards in Games User Research. In *Proceedings of CHI 2013*. Paris, France.

Nacke, L. E. (2009). *Affective Ludology: Scientific Measurement of User Experience in Interactive Entertainment* (PhD Thesis). Blekinge Institute of Technology, Karlskrona, Sweden. Retrieved from http://hci.usask.ca/publications/view.php?id=178

Nacke, L. E., Ambinder, M., Canossa, A., Mandryk, R. L., & Stach, T. (2009). *Game Metrics and Biometrics: the Future of Player Experience Research*. Presented at the GDC 2009 - Future Play.

Nacke, L. E., Drachen, A., Kuikkaniemi, K., Niesenhaus, J., Korhonen, H. J., Hoogen, W. M. van den, … Kort, Y. A. W. de. (2009). *Playability and Player Experience Research*. London, UK: DiGRA. Retrieved from

http://www.bth.se/fou/forskinfo.nsf/17e96a0dab8ab6a1c1257457004d59ab/e0a8cdd8cfc0c7e6c125762c005557c0!OpenDocument

Nacke, L. E., & Lindley, C. A. (2008). Flow and Immersion in First-Person Shooters: Measuring the Player's Gameplay Experience. In *Proceedings of the 2008 Conference on Future Play: Research, Play, Share* (pp. 81–88).

Nakhimovsky, A. (1988). Aspect, Aspectual Class, and the Temporal Structure of Narrative. *Computational Linguistic*, *14*(2), 29–43.

Neal, A. S., & Simons, R. M. (1984). Playback: a method for evaluating the usability of software and its documentation. *IBM Systems Journal*, *23*(1), 82 – 96. doi:10.1147/sj.231.0082

Neubauer, S. (2006). *The Gameplay Video Segmentation Method* (Diploma Thesis). Stuttgart Media University.

New Zealand OFLC. (1993). *Films, Videos, and Publications Classification Act 1993*.

New Zealand OFLC. (2009). *Public Perception of a Violent Videogame: X-Men Origins: Wolverine*.

Onyett, C. (2011). The Elder Scrolls V: Skyrim Review; Say goodbye to real life. *IGN*. Retrieved from http://au.ign.com/articles/2011/11/10/the-elder-scrolls-v-skyrim-review?page=1

Pedersen, C., Togelius, J., & Yannakakis, G. N. (2010). Modeling Player Experience for Content Creation. *Computational Intelligence and AI in Games, IEEE Transactions on*, *2*(1), 54 – 67. doi:10.1109/TCIAIG.2010.2043950

PEGI Questionnaire. (n.d.). Retrieved from http://www.pegi.info/media/pdf/235.pdf

Peters, R. J., & Itti, L. (2008). Applying computational tools to predict gaze direction in interactive visual environments. *ACM Transactions on Applied Perception (TAP)*, *5*(2). doi:10.1145/1279920.1279923

Plutchik, R. (2001). The Nature of Emotions. *American Scientist*, *89*, 344 – 350.

Pratt, W. K. (1978). *Digital Image Processing*. Wiley-Interscience Publication.

Rabiner, L. (1989). A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, *77*(2), 257 – 286.

Reynar, J. C. (1998). *Topic Segmentation: Algorithm and Applications.* (PhD Thesis). University of Pennsylvania.

Roads, C. (1996). *The Computer Music Tutorial*. MIT Press.

Robertson, S. E. (1981). The Methodology of Information Retrieval Experiment. *Information Retrieval Experiment*, 9–31.

Robine, M., Hanna, P., Ferraro, P., & Allali, J. (2007). Adaptation of String Matching Algorithms for Identifcation of Near-Duplicate Music Documents. In *Proceedings of the 30th Annual International SIGIR Conference*.

Ruch, A. W. (2010). Videogame Interface: Artefacts and Tropes. In *Videogame Cultures and the Future of Interactive Entertainment Global Conference*. Oxford, UK.

Sadlier, D. A., Marlow, S., O'Connor, N. E., & Murphy, N. (2002). Automatic TV Advertisement Detection from MPEG Bitstream. *Pattern Recognition Society*, *35*(12), 2–15.

Salen, K., & Zimmerman, E. (2004). *Rules of Play: Game Design Fundamentals*. MIT Press.

Santos, A. R., & Kim, H. Y. (2006). Real-Time Opaque and Semi-Transparent TV Logos Detection. In *Proceedings of the 5th International Information and Telecommunication Technologies Symposium (I2TS)*.

Sasse, D. (2008). *A Framework for Psychophysiological Data Acquisition in Digital Games* (Master's Thesis). Otto-von-Guericke-University Magdeburg. Retrieved from http://www.gamecareerguide.com/thesis/080520_sasse.pdf

Saunders, J. (1996). Real-Time Discrimination of Broadcast Speech/Music. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing* (pp. 993–996). Atlanta, GA, USA.

Schaeffer, P. (1966). *Traité des Objets Musicaux [Treatise on Musical Objects]*. Le Seuil.

Schott, G. (2009). "I Like the Idea of Killing But Not the Idea of Cruelty": How New Zealand Youth Negotiate the Pleasures of Simulated Violence. In *Proceedings of DiGRA 2009*. London.

Schott, G., & Kambouri, M. (2003). Moving between the Spectral and Material Plane: Interactivity in Social Play with Computer Games. *Convergence*, *9*(3), 41–55.

Schott, G., Marczak, R., Mäyrä, F., & Vught, J. V. (2013). DeFragging Regulation: From putative effects to "researched" accounts of player experience. In *Proceedings of DiGRA 2013*.

Schott, G., Marczak, R., & Neshausen, L. (2014). <Exploring> the <Cause> of Game (Derived) <Arousal>: What Biometric Accounts of Player Experience Revealed. In *Proceedings of DiGRA 2014*.

Schott, G., Vught, J. V., & Marczak, R. (2012). The “Dominant Effect” of Games: Content vs. Medium. *CoLab: Journal of Creative Technologies*, (3).

Shaker, N., Yannakakis, G. N., & Togelius, J. (2010). Towards Automatic Personalized Content Generation for Platform Games. Presented at the Artificial Intelligence and Interactive Digital Entertainment (AIIDE’10).

Shukla, D., & Sharma, M. (2011). Scene Change Detection Algorithms & Techniques: A Survey. *International Journal of Computer Science and Information Security*, *9*(12), 73–77.

Skorochod’ko, E. F. (1971). Adaptive Method of Automatic Abstracting and Indexing. In *Proceedings of the IFIP Congress* (Vol. 2, pp. 1179–1182).

Slik, M., Jongebloed, H., & van Staalduinen, M. (2013). *Video Based OCR: a Case Study of Real Time In-screen Subtitle Recognition* (Technical Review No. Q3 2013).

Smith, A. M., Lewis, C., Hullett, K., Smith, G., & Sullivan, A. (2011). *An Inclusive Taxonomy of Player Modeling* (Tech. Rep. No. UCSC-SOE-11-13). Retrieved from http://www.soe.ucsc.edu/research/technical-reports/ucsc-soe-11-13

Snoek, C. G. M., & Worring, M. (2005). Multimodal Video Indexing: A Review of the State-of-the-art. *Multimedia Tools and Applications*, *25*(1), 5–35.

Stroustrup, B. (1997). *The C++ Programming Language* (3rd ed.). Addison-Wesley Professional.

Tangkiengsirisin, S. (2012). Cohesion and Coherence In Text. *Thammasat University Journal*, *31*(2).

Thurau, C., Bauckhage, C., & Sagerer, G. (2003). Combining Self Organizing Maps and Multilayer Perceptrons to Learn Bot-Behavior for a Commercial Computer Game. Presented at the GAME-ON.

Thurau, C., Bauckhage, C., & Sagerer, G. (2004). Learning Human-Like Movement Behavior for Computer Games. Presented at the 8th Int. Conf. on the Simulation of Adaptive Behavior (SAB’04).

Togelius, J., Nardi, R. D., & Lucas, S. M. (2007). Towards Automatic Personalised Content Creation for Racing Games. Presented at the Computational Intelligence and Games. CIG 2007. IEEE Symposium on. doi:10.1109/CIG.2007.368106

Viterbi, A. J. (1971). Convolutional Codes and Their Performance in Communication Systems. *IEEE Transactions on Communications Technology*, *19*(5), 751–772.

Wellisch, H. H. (1986). The oldest printed indexes. *The Indexer*.

Witty, F. J. (1973). The Beginnings of Indexing and Abstracting: Some Notes towards a History of Indexing and Abstracting in Antiquity and the Middle Ages. *The Indexer*.

Wright, M., Freed, A., & Momeni, A. (2003). Open Sound Control: State of the Art 2003. In *Proceedings of NIME 2003* (pp. 153–159).

Yannakakis, G. N. (2012). Game AI revisited. In *Proceedings of the 9th conference on Computing Frontiers* (pp. 285–292).

Yannakakis, G. N., & Hallam, J. (2009). Real-time Game Adaptation for Optimizing Player Satisfaction. *Computational Intelligence and AI in Games, IEEE Transactions on, 1*(2), 121 – 133.

Yannakakis, G. N., Spronck, P., Loiacono, D., & André, E. (2013). Player Modeling. In *Artificial and Computational Intelligence in Games* (Vol. 6, pp. 45–59).

Yarlagadda, R. K. R. (2010). *Analog and Digital Signals and Systems*. Springer.

Ye, Q., Huang, Q., Jiang, S., Liu, Y., & Gao, W. (2006). Jersey number detection in sports video for athlete identification. In *Proc. SPIE 5960, Visual Communications and Image Processing*.

Zagal, J. P., Fernandez-Vara, C., & Mateas, M. (2008). Rounds, Levels, and Waves: The Early Evolution of Gameplay Segmentation. *Games and Culture, 3*(2), 175–198.

Zagal, J. P., Mateas, M., Fernández-Vara, C., Hochhalter, B., & Lichti, N. (2005). Towards an Ontological Language for Game Analysis. In *Proceedings of the 2005 Digital Games Research Association Conference (DiGRA)*. Retrieved from http://www.digra.org/dl/db/06276.09313.pdf

Appendix A

Open system indexing and Metadata

Once a gameplay performance is captured into a video file, it is also possible to initiate open-system indexing (see Section 3.4.1), which refers to labelling the document itself, rather than its content. The existence of a video-file permits a metadata tagging process (see Section 3.4.3), in order to describe the video file as part of a session context. Figure 137 shows the different metadata that can be used as video descriptors. These include: 1) administrative information, such as player name, age, gender, game title, genre, publisher, date, classification, 2) technical information, such as video file duration, frame rate, frame size, video format, audio format, 3) spatiotemporal information, such as session location, starting date, duration, and 4) reference information, such as psychophysiology measures, facial expression capture, keystrokes, interviews, previous session. Figure 138 illustrates how the metadata can be filled using information about a session.

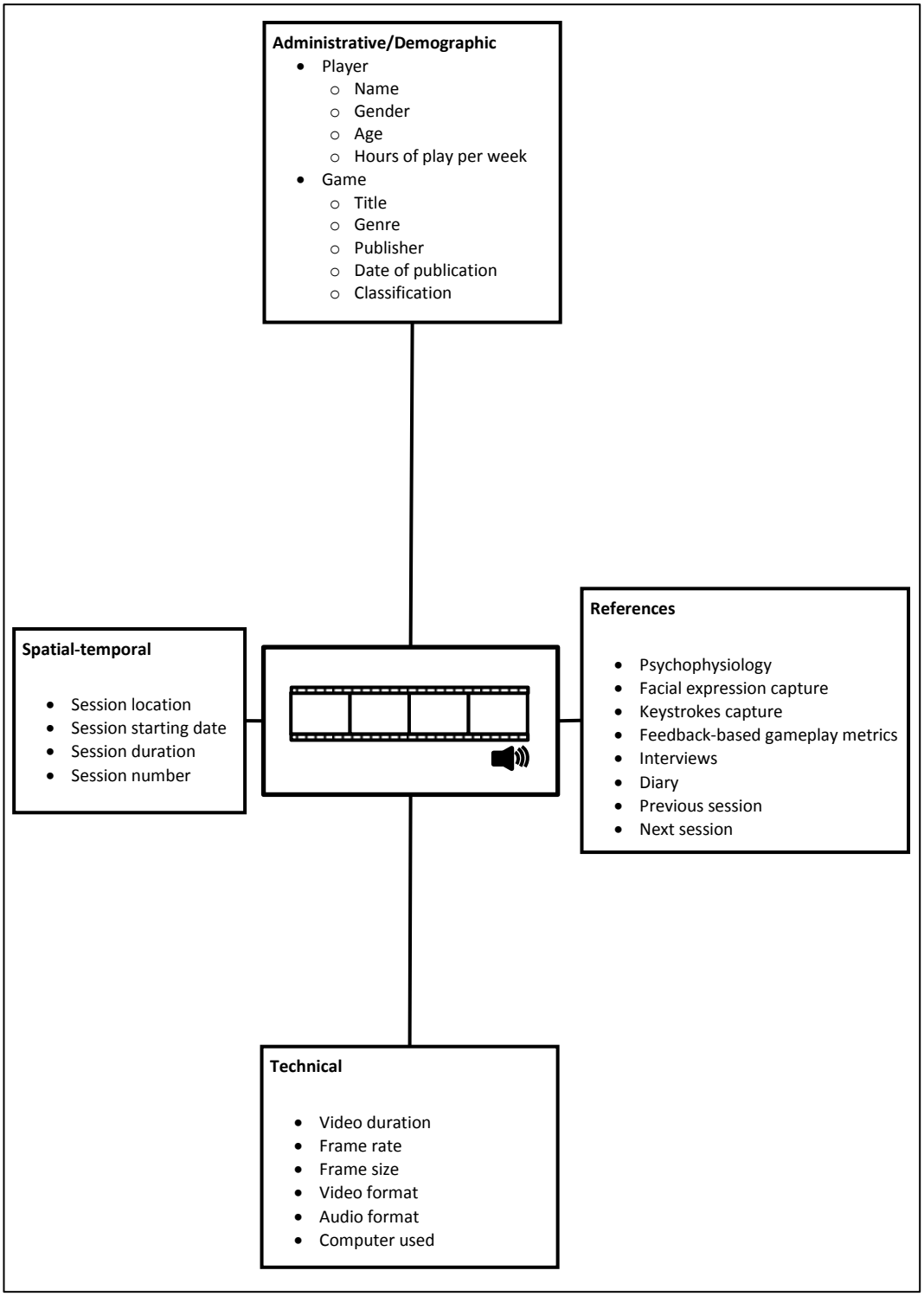


Figure 137. Metadata linked to the video file

Figure 137 illustrates how gameplay session metadata, representing administrative information, as well as information lost during the focus phase, can be useful to replace the performance in the session context. The figure also illustrates the existence of connected reference data for further correlation. Four metadata types are highlighted: Administrative/Demographic, Reference, Technical and Spatial-temporal. The metadata tagging of the video file represents an open-system indexing of the performance.

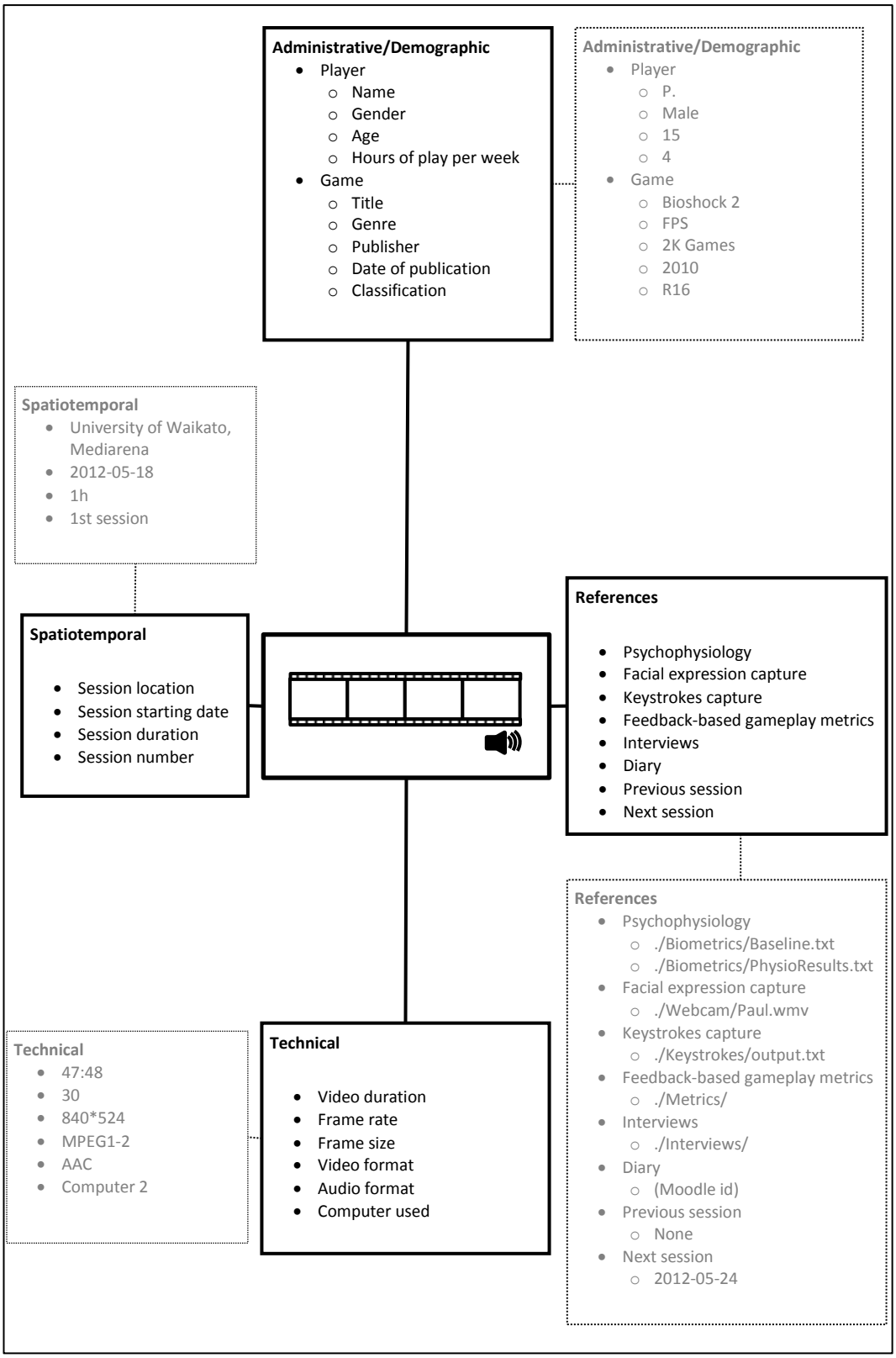


Figure 138. Metadata example with the first session of participant P.

Figure 138 provides an example of the use of metadata open-system indexing tagging, as first illustrated in Figure 137. The four type of metadata are filled using a session with the participant P.

Appendix B

Segmented performances

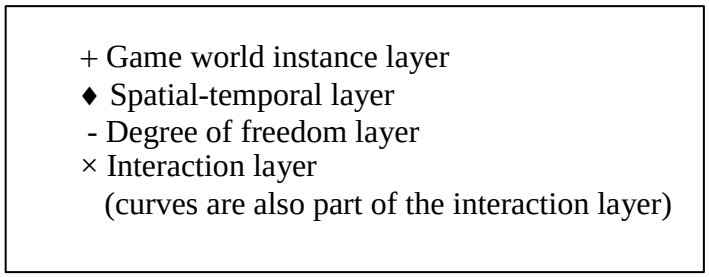


Figure 139. Layer legend

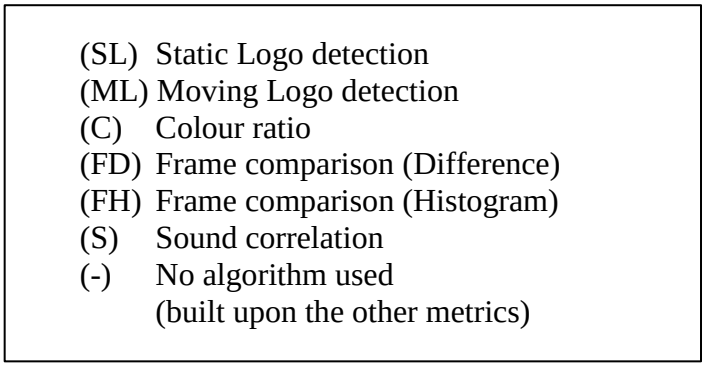


Figure 140. Algorithm legend

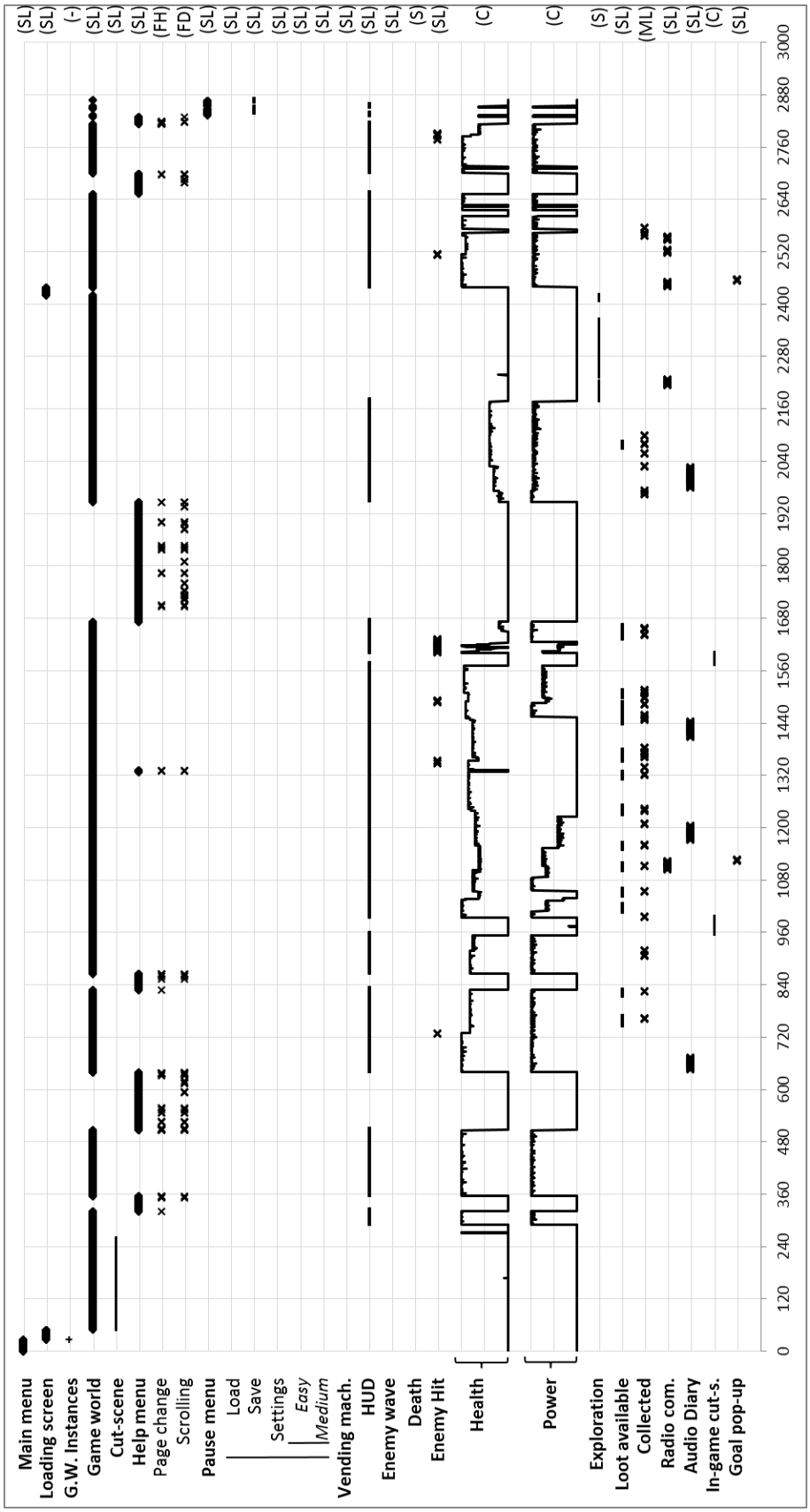


Figure 141. ***Bioshock 2*** **– Participant P. – Session 01**

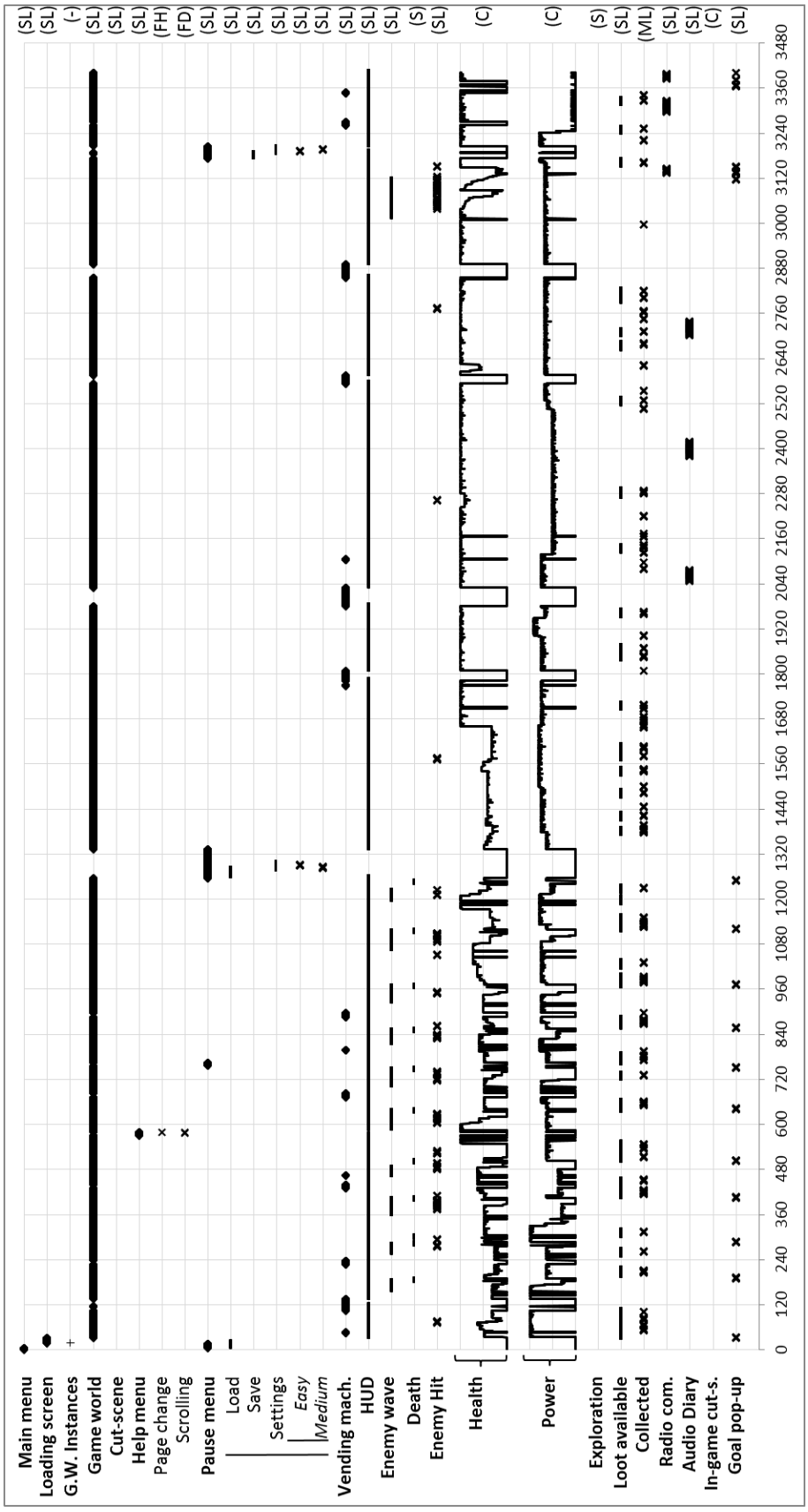


Figure 142. *Bioshock 2* – Participant P. – Session 04

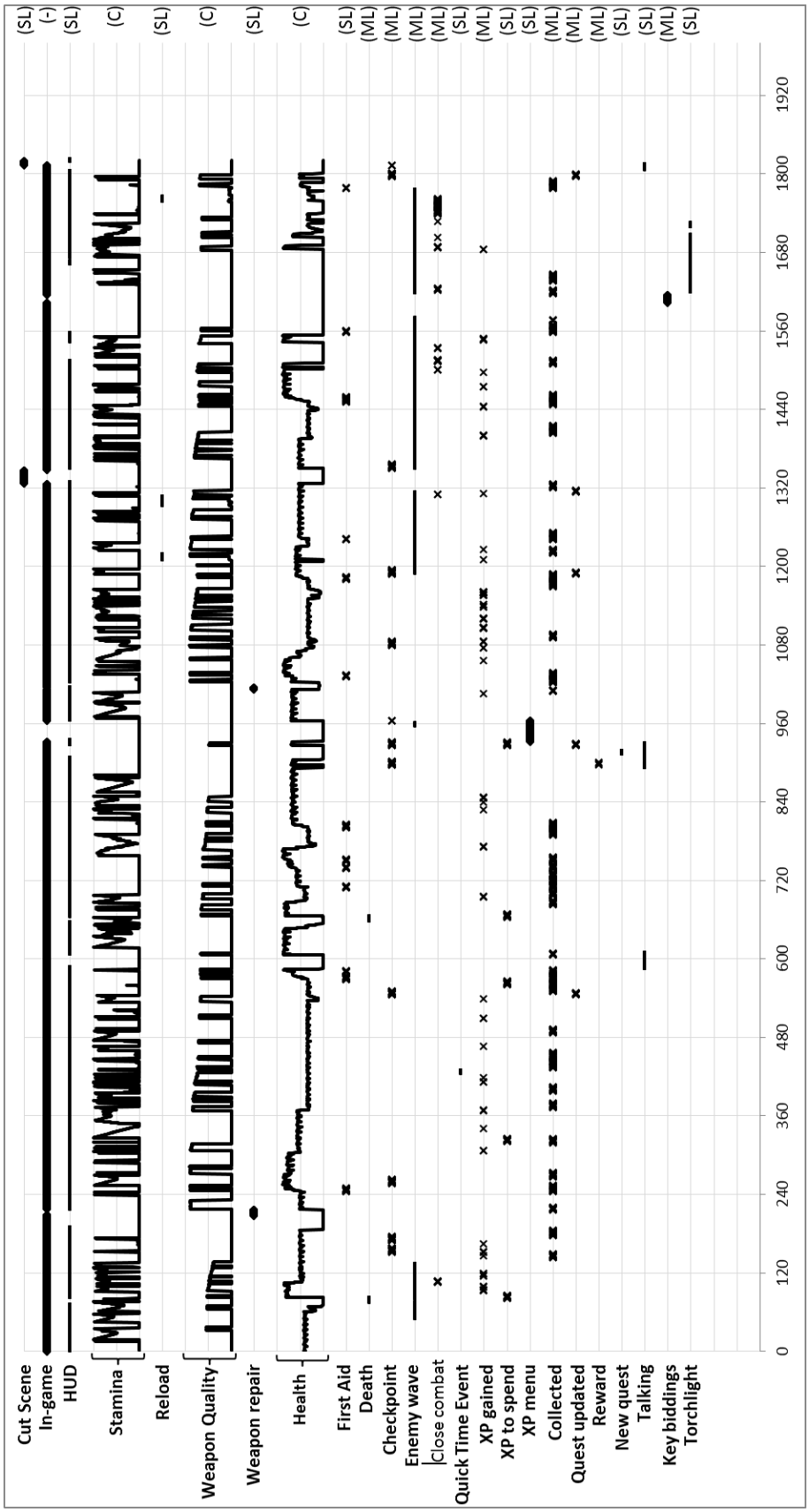


Figure 143. ***Dead Island*** **– Participant Ja. – Session 01**

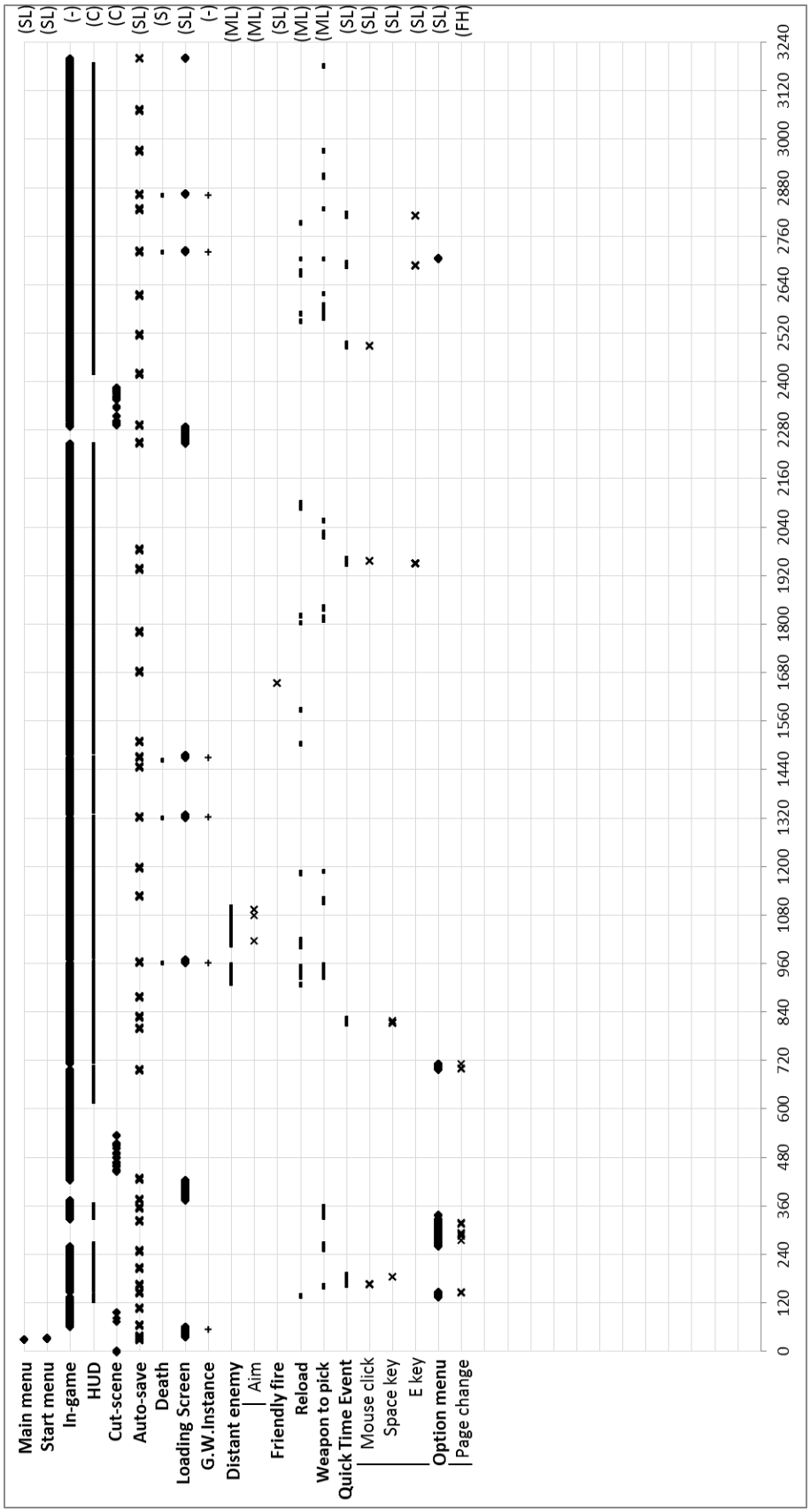


Figure 144. ***Battlefield 3*** **– Participant Jo. – Session 01**

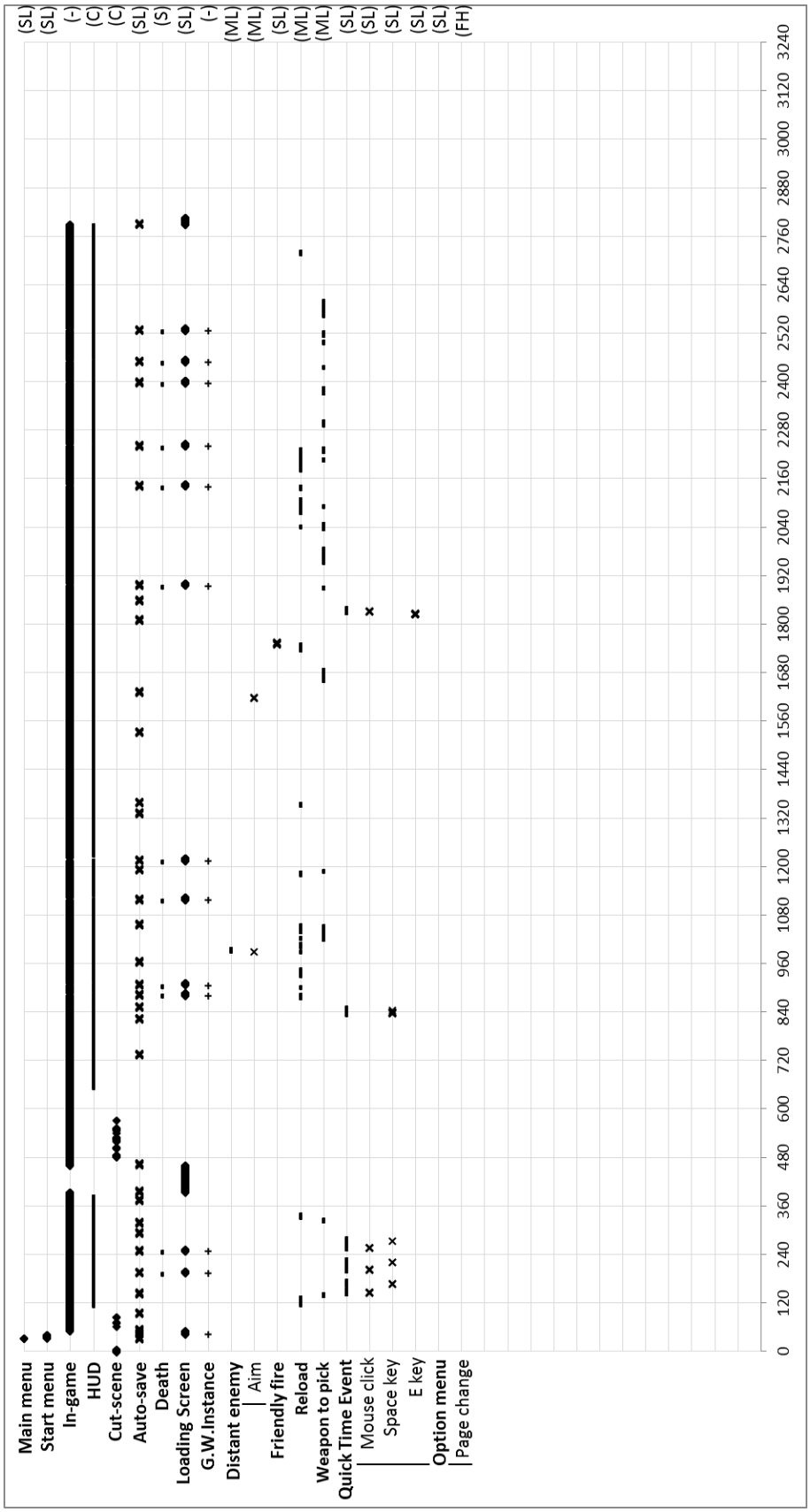


Figure 145. ***Battlefield 3*** **– Participant N. – Session 01**

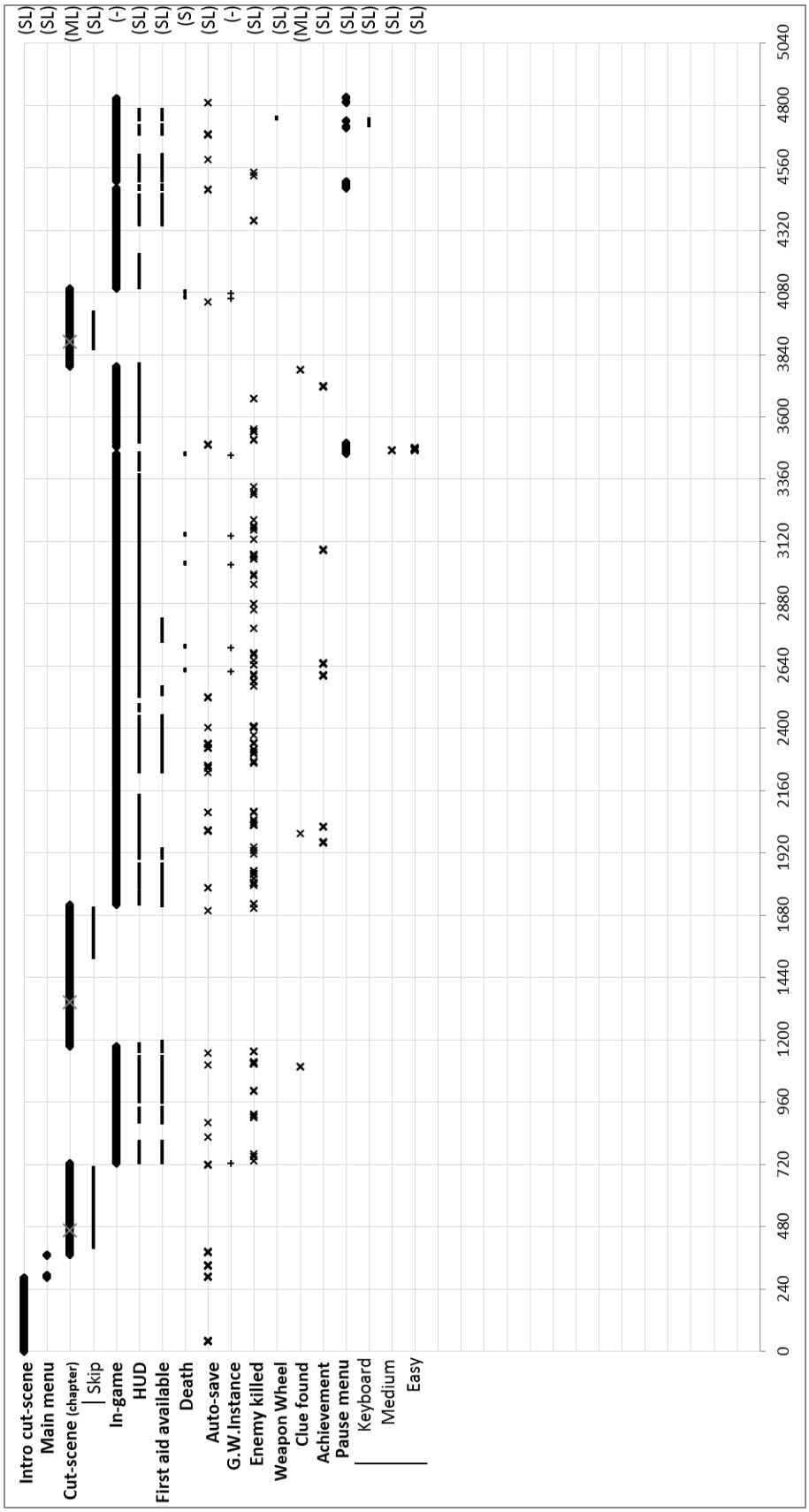


Figure 146. ***Max Payne 3*** **– Participant A. – Session 01**

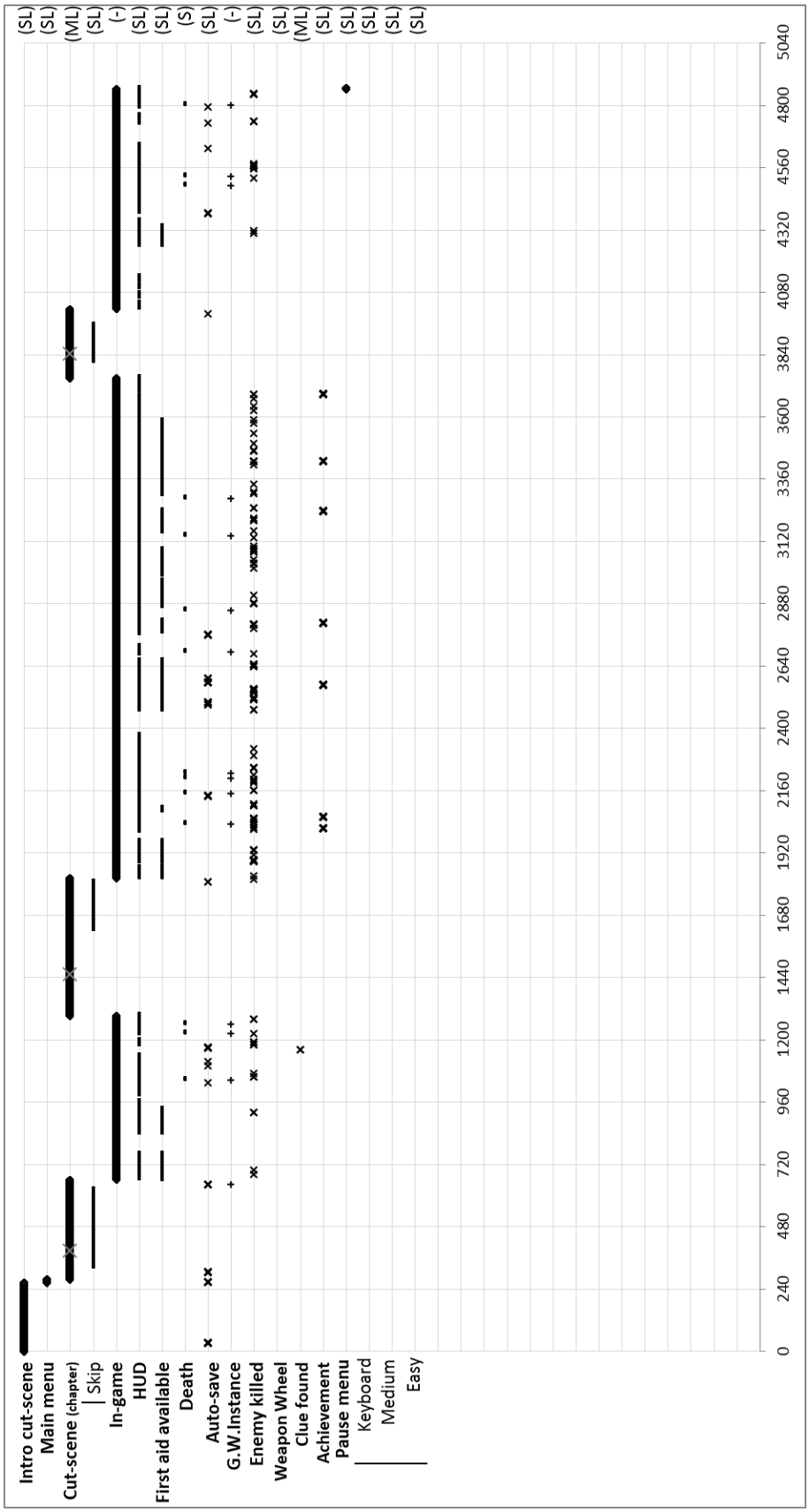


Figure 147. *Max Payne 3* – Participant P. – Session 01

Appendix C

Published Papers during the PhD

Post-Processing Gameplay Metrics for Gameplay Performance Segmentation based on Audio-Visual Analysis
Raphaël Marczak, Gareth Schott, Pierre Hanna

IEEE Transactions on Computational Intelligence and AI in Games, special issue on Game Data Mining, 2015 (in press)

This paper introduces a new variant of game-play metrics that further develops a set of processes that expand user-centred game testing practices capable of quantifying user experiences. The key goal of the method presented here is to widen the appeal and application of game metrics within research relevant to, and representative of the wider field of game studies. In doing so, we acknowledge that the interests of this research community is often focused on player experience and performance with a broad range of off-the-shelf games that have already been released to the public. In order to be able to include any PC game system within research (or audio-video stream, e.g. YouTube walkthroughs) our approach comprises of a post-processing method for analysing player performance. Through exploiting the audio-visual streams that are transmitted to the player, this approach processes content activated and generated during play and is therefore representative of individual player's encounters with specific games.

Audio visual analysis of player experience: Feedback-based gameplay metrics
Raphaël Marczak and Gareth Schott

Game Research Methods, 2015 (in press), Book Chapter.

This chapter introduces readers to an innovative method designed specifically to meet game researchers' need to be able to analyse and explore players' experiences with any off-the-shelf PC game. We introduce how insights into players' experiences are achieved from automatically processing significant elements of the audio-visual feedback (i.e., moving image and sound) produced by the game.

From Automatic Sound Analysis of Gameplay Footage [Echos] to the Understanding of Player Experience [Ethos]: an Interdisciplinary Approach to Feedback-Based Gameplay Metrics
Raphaël Marczak, Pierre Hanna, Jean-Luc Rouas, Jasper Van Vught, Gareth Schott

SMC-ICMC 2014 (Joint International Computer Music Conference and Sound & Music Computing conference), Greece

In line with the ICMC|SMC|2014 conference theme "from digital echos to virtual ethos", and the conference interdisciplinary main objective; the present paper seeks to demonstrate that the sound feedback stream produced by videogames when activated by players (echos) can be automatically analyzed in order to study how sound can, not only, describe a gameplay perfor-mance, but also help to understand player experience and emotions (ethos). To do so, the present paper illustrates how sound processing algorithms can be applied in the game studies discipline in order to assess and understand better how players engage with videogames. The present paper proposes to adapt the Feedback-based Gameplay Metrics method, successfully applied to the analysis of gameplay footage video stream, to the sound stream, via the automatic detection of musical sequences and speech segment.

Exploring the Cause of Game Derived Arousal: What biometric accounts of player experience revealed
Gareth Schott, Raphaël Marczak, Leanne Neshausen

DiGRA 2014 (International Digital Games Research Association Conference), Salt Lake City, USA

In the context of a three-year research study into game violence, designed to query the strong association between policy-oriented effects research and responsive regulation measures, a mixed methodology was employed to examine player experience with 'violent' texts (as introduced in Schott et al., 2013a). Guided by the supposition that 'explor[ing] the extent to which the public's perception of causal links between game playing and various social ills' might be 'moderated or even undermined by [knowledge of] how players actually respond to and negotiate their way through the content and characteristics of the medium' (OFLC, 2009, p. 24), our study contains a number of data or 'entry' points. The aim is to characterize the multi-dimensional nature of players' experiences. This paper addresses the outcome of utilizing one measure in particular, biometric measures (GSR), as a guide for determining what aspects of Battlefield 3 (Electronic Arts) should be examined in accounts of player experiences. Our method of applying biometric data is outlined and what it was able to reveal in terms of the occurrence and cause of arousal for players is discussed. The paper reflects on what a broader and textually neutral method of accessing game-play experiences in the context of a 'violent' game reveals about play. A key outcome of taking this approach to detecting what aspects of a game had the most impact on players, is how GSR led us away from content that is more commonly highlighted and prioritized in the classification of games like Battlefield 3 - as an engagement with 'violence'.

Feedback-based gameplay metrics
Raphaël Marczak, Gareth Schott, Pierre Hanna and Jean-Luc Rouas

FDG 2013 (International Foundations of Digital Games Conference), Chania, Creete, Greece

The application of gameplay metrics to empirically express a player's engagement with the game system has become more appealing to a broader range of researchers beyond the computer sciences. Within game studies, the appropriation and use of gameplay metrics not only further shifts these methods beyond formalized user testing (e.g. with the aim of product improvement) but creates a demand for a more universal approach to game metric extraction that can be applied to a wider range of games and account for different social science research agendas. Set in the context of a study into player experiences with participants aged 14-16 years of age, a method of articulating and identifying key moments of game-play pertinent to an analysis of experience with simulated violence was required. This paper describes a pragmatic solution to the need to study a range of games that were determined not by the researchers but the study's participants. The solution exploits and examines the audio-video feedback presented to the player as part of their engagement with a game system. This paper broadly outlines this mode of metric extraction and its application to research that seeks to understand the nature of engagement and player motivations across a mixed methodology approach.

DeFragging Regulation: From Putative Effects to Researched Accounts of Player Experience
Gareth Schott, Raphaël Marczak, Frans Mäyrä and Jasper Van Vught

DiGRA 2013 (International Digital Games Research Association Conference), Atlanta, USA

In line with the conference theme for 2013, this paper introduces a research project that is seeking to 'defragment' research dealing with player experiences. Located at an intersection between humanities, social sciences and computer sciences, our research aims to achieve greater receptiveness for accounts of games that emphasise "the relationship between the structure of a game and the way people engage with that system" (Waern, 2012, p.1) in the context of game regulation. Working specifically within the context of the New Zealand classification system, which possesses a legally enforceable age-restriction system, the project seeks to strengthen regulators capacity to utilize Section 3(4) of the current Classification Act and support the employment of concepts such as 'dominant effect', 'merit' and 'purpose' when classifying games (OFLC, 2012). Extending an established appreciation within game studies for the way games produce polysemic performances and readings, this paper draws on our mixed methods approach in an exploration of the nature of a players' experience with Max Payne 3 (Rockstar Vancouver). In doing so, we illustrate the different dynamics at play in its expression and use of violence - dynamics that fail to achieve expression when games are considered more generally within political and social realms.

Not Knowing: Locating Player Experience Between Ideal and Active Play
Gareth Schott, Jasper van Vught and Raphaël Marczak

Playing with Virtuality (2013). Theories and Methods of Computer Game Studies. Book Chapter

No abstract.

Understanding player experience finding a usable model for game classification
Jasper van Vught, Gareth Schott and Raphaël Marczak

IE 2012 (Australasian Interactive Entertainment Conference), Auckland, New Zealand

Digital games receive an age restriction classification rating based on their depiction of harmful content and its presumed impact on players. While classification processes serve as predictors of the subsequent interactions between player and game text they remain largely inferential and an exercise in caution. Confounded by the medium's interactive nature, we argue that classification processes would benefit from research that provides empirical accounts of the interactive experience of games. This paper presents findings taken from a research project with the aim of operationalizing over a decade of Game Studies theorization on the distinct quality of games. The intention is to produce an empirically validated model of media 'usage,' capable of informing regulation processes and the classification of games (within a New Zealand context). Here we draw on findings achieved from one component of our mixed methodology research design [37] - A structured diary method that was employed to allow game players to chronicle different elements of their gameplay experience with a single text as they progressed through it. The findings serve to highlight the applied value of Game Studies' theory and its capacity to account for the 'actual' experience of play and the ways game texts are activated under the agency of players once they enter everyday life and culture.

Feedback-Based Gameplay Metrics: Measuring Player Experience via Automatic Visual Analysis
Raphaël Marczak, Jasper van Vught, Gareth Schott and Lennart Nacke

IE 2012 (Australasian Interactive Entertainment Conference), Auckland, New Zealand

Using gameplay metrics to articulate player interaction within game systems has received increased interest in game studies. The value of gameplay metrics comes from a desire to empirically validate over a decade of theorization of player experience and knowledge of games as ludic systems. Taking gameplay metrics beyond formalized user testing (i.e. with the aim of improving a product) allows researchers the freedom of examining any commercially available game without the need to have access to the game's source code. This paper offers a new methodology to obtain data on player behaviour, achieved through analysing video and audio streams. Game interface features are being analysed automatically, which are indicative of player behaviour and gameplay events. This paper outlines the development of this methodology and its application to research that seeks to understand the nature of engagement and player motivations.

The "Dominant Effect" of Games: Content vs. Medium
Gareth Schott, Jasper van Vught and Raphaël Marczak

Creative Technologies, Third Edition 2012. Journal.

Digital games receive an age-restriction rating based on the depiction of harmful content and its possible impact on players. Following on from film, the relationship between media content and its psychological impact on audiences is assumed to be further heightened by the interactive nature of the medium as it makes players responsible for constructing the moving-image on screen. While classification processes continue to serve as an exercise in caution, there remains little evaluation of a particular rating decision's accuracy via any subsequent examination of the interactions between player and game text. This paper argues for the benefits of researched accounts detailing the interactive experience of games for its capacity to challenge public understanding of the medium. In doing so, the paper will introduce a research design that is currently being employed to achieve an understanding of player experiences. The intention is to produce an empirically validated model of media 'usage,' capable of accounting for the 'actual' experience of play and the ways game texts are activated under the agency of players once they enter everyday life and culture.

Extracting Game-play Metric Data from Audio/Video processing: a Practical Solution for Game Studies Research
Raphaël Marczak, Gareth Schott and Jasper van Vught

CHINZ 2012 (New Zealand Human-Computer Interaction Conference), Dunedin, New Zealand

Today videogame classification works to a set of guidelines initially designed for, and more conducive to linear mediums (film, television and literature). As a result, digital games receive an age restriction rating based on both their depiction of harmful content and its prospective impact on players. Not accounting for the medium's interactive qualities means that envisaging the player's experience and the nature and impact of interactions between players and game texts remains a largely inferential practice and an exercise in caution. Given the medium's interactive nature, we argue that classification processes would be supported by research that provides empirical accounts of the interactive experience of games. In order to take into account the unique experiential properties of games, a mixed methodological approach located at an intersection between humanities, social sciences and computer science is being employed in a large-scale study of player experiences. This paper presents one of the earliest challenges for this study: the capacity to gather game-play metric data from game texts that are selected by participants as an ongoing process throughout the course of the study. With no real advance knowledge of game choices, access to source code or game developers, a pragmatic solution was required to capture player interactions with game texts. This paper presents the beginnings of a method of gathering game-play metrics through utilizing audio and video processing and its required synchronization with other forms of data output.

Age-Restriction: Re-examining the interactive experience of 'harmful' game content
Jasper van Vught, Gareth Schott and Raphaël Marczak

Nordic DiGRA 2012 (International Digital Games Research Association Conference), Tampere, Finland

Similar to the classification rating of films, screen depictions of violence within digital games are issued with an age restriction rating. Such approaches still fail to adequately incorporate players' experience of the screen, confounded by the medium's interactive nature, in their assessments. The current failure to account for, or describe subsequent interactions between player and game text leaves the classification process largely inferential. This paper presents a framework that forms the basis for an empirical assessment of the interactive experience of games. In it, we aim to account for the processes and outcomes of play and the extent to which play relates to the design of the game text. By operationalizing game studies' extensive theorization of the distinct quality of games, a new model of media 'usage' is sought to enhance regulation processes and better inform the public's perception of games (specifically within New Zealand). In this paper we draw specifically on data produced from one part of a mixed methodology research design. A structured diary method was employed to allow game players to chronicle different elements of their gameplay experience with a single text as they progressed through it. By demonstrating the applied value of game studies' contribution to knowledge, the research project aims to contribute to a new paradigm that is capable of accounting for the 'actual' experience of play and the ways game texts are activated under the agency of players once they enter everyday life and culture.