

<http://researchcommons.waikato.ac.nz/>

Research Commons at the University of Waikato

Copyright Statement:

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

The thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognise the author's right to be identified as the author of the thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from the thesis.



THE UNIVERSITY OF

WAIKATO

Te Whare Wānanga o Waikato

Exploration of Pervasive Games in Relation to Mobile Technologies

Paul Morris

This thesis is submitted in partial fulfillment of the requirements for the
Degree of Master of Science at the University of Waikato.

February 2013

© 2013 Paul Morris

Abstract

The project is an exploration of Pervasive Games in relation to mobile technologies, with the intention of developing a pervasive game engine. Pervasive Games are interactive games where the participants drive the game play by playing the game in both the real world and a virtual environment. This is an area of gaming that has rapidly evolved over the last few years.

The initial research involved establishing several key elements common to existing pervasive applications, defining real world / virtual world considerations for game play (both positive and negative) and identifying the technical requirements needed to implement play elements on a mobile device. After comparing several platforms the Windows 7 platform was selected for development purposes. The requirements for establishing a working development platform (with delivery mechanism) was investigated and a working environment set-up.

A pervasive games engine was then developed in the format of 67 code stubs (coding solutions) that allow the implementation of solutions to gaming elements required in the development of pervasive applications.

Two new helper classes were in addition developed containing solutions to topics related to run-time data storage (StorageUtils.cs) and generic gaming tasks (GameCode.cs). A pervasive game was implemented to test a cross section of functionality in the engine. The basic principle behind the game was to overlay various layers video, backgrounds, sprite and text, to build up an immersive pervasive environment with a player in the centre of the game imagery, game domain and real world. The intention of the game was to see how the pervasive game experience could be reflected in the game mechanics and pervasive interaction, while utilising the engine functionality.

Contents

Chapter 1:	Introduction	1
Chapter 2:	Background Info: What Is Play?	3
Chapter 3:	Pervasive Games	7
Chapter 4:	The Platform.....	13
Chapter 5:	Setting Up the System	17
Chapter 6:	Game Engine	29
Chapter 7:	Development of a Game.....	117
Chapter 8:	Test Game	119
Chapter 9:	User Personas	127
Chapter 10:	Usability Requirements	141
Chapter 11:	Conclusions	145
Appendix A:	Isolated Storage Testing	155
Appendix B:	Pervasive Game Engine	167
Appendix C:	Pervasive Game Engine Setup and Configuration	171
Appendix D:	Code Index for the Zombie Gold Application	173
Appendix E:	NIST/IUSR – CISU-R.....	175
Appendix F:	Ethics Approval and Statement.....	199
Appendix G:	Usability Evaluation Data Template	201
Appendix H:	Participant Task Scenarios for Zombie Gold	203
Appendix I:	System Usability Scale (SUS).....	209
Appendix J:	Pervasive Experience Factor	211
Appendix K:	Preliminary Benchmark Usability Data	213
Appendix L:	Preliminary Benchmark SUS Levels.....	215
Appendix M:	Preliminary Benchmark Pervasive Experience Factor.....	217
Appendix N:	Usability Data.....	219
Appendix O:	SUS Levels	223
Appendix P:	Pervasive Experience Factor	225
Appendix Q:	Windows 7 Phone Features	229

Chapter 1: Introduction

Definition: Pervasive Games are interactive games where the participants drive the game play by playing the game in both the real world and a virtual environment (modern concept circa. 2001).

In the Pervasive Game Model games can be played in different time frames as well as incorporating non-linear elements. A game may last several days with the players' normal life (eating, sleeping, working, and talking with non-players) being taken into consideration and add to the game experience.

Conventional games tend to setup a rigorously defined sphere of influence (board, pitch, PC) that is static and isolates the player from their environment, in a pervasive game the game world can go wherever you go, into the virtual or real world. Players can interact with their environment creating new game play based on how they perceive their current game location. An example of a Pervasive Game is a treasure hunt that takes place in both the real world and virtual world. A blurring of the boundaries between the real and virtual domains are achieved by hiding clues and quest items from players in the real world, net or any virtual environment.

This document records my exploration into the genre of Pervasive Games.

The intention is to research the topic, find a platform to develop a Pervasive Game Engine (functionality useful to pervasive game development), design and implement a test application, and then use a recognised methodology to test the API's.

- | | |
|------------|--|
| Chapter 2: | Background Info: What Is Play?
Briefly looks at what is play by identifying some of the historical definitions of play (magic circle) and then extends these definitions to include the Pervasive Game Model. |
| Chapter 3: | Pervasive Games
Breaks down the components of a pervasive game. |
| Chapter 4: | The Platform
Looks at my reasons for selecting the Windows 7 platform. |
| Chapter 5: | Setting Up the System
Breaks down the process of registering for a developer licence, setting up the Integrated Development Environment, identifying relevant elements of the Framework and recognising any Design Guidelines that need to be complied with. |
| Chapter 6: | Game Engine
Identifies specific programming elements (technical areas) that need to be understood and addressed from a coding point of view to implement Pervasive Games on a mobile device. Solutions are provided in the form of explanations of the theory associated with the topic and implemented solutions (coded examples). |
| Chapter 7: | Development of a Game
Explanation of what the game sets out to test. |
| Chapter 8: | Test Game
Description of the game developed to test the coding research undertaken to implement Pervasive Games on a mobile device. |
| Chapter 9: | User Personas
Description of the intended demographic for the game, a breakdown of the scenarios of use the players will expect of the game and participant task scenarios. |

- Chapter 20: Usability Requirements
Usability requirements and testing in the form of a NIST/IUSR Common Industrial Specification for Usability Requirements document.
There is some repetition of data in this chapter, this is to comply with the format and requirements of the NIST/IUSR – CISU-R documentation.
- Chapter 12: Conclusions
Reflection on the project process.

Chapter 2: Background Info: What Is Play?

The concept of play seems to be intrinsic to our basic nature; we are introduced to it in childhood and gravitate to it during the remainder of our lives. The literature on this topic is unending and the question seems to have always been a focus of some debate.

Modern sources tend to gravitate initially to Johan Huizinga who conducted philosophical and anthropological work in the 1930s and defined conventional play as:

“Something happening outside of ordinary life, where play is seen as ritual activity, governed by rules that are separate from everyday reality” [Johan Huizinga 1930’s]

“ Play is free activity standing quite consciously outside ‘ordinary’ life as being ‘not serious’, but at the same time absorbing the players intensely and utterly. It is an activity connected with no material interest, and no profit can be gained by it. It proceeds within its own proper boundaries of time and space according to fixed rules and in an orderly manner. It promotes the formation of social groupings, which tend to surround themselves with secrecy and to stress their difference from the common world by disguise or other means.” [Johan Huizinga 1938]

The implication is that, we need to buy into the activity and must be prepared to suspend the accepted normal social rule system/s. The act of playing must also be conducted within a physical boundary. This boundary is required to allow the participant/s to engage in the activity safely and be compliant with the game rules.

Huizinga later revised the concept of a game boundary further by defining the ‘Magic Circle’, the special domain within which the game exists.

“All play moves and has its being within a play-ground marked off beforehand either materially or ideally, deliberately or as a matter of course. Just as there is no formal difference between play and ritual, so the ‘consecrated spot’ cannot be formally distinguished from the play-ground. The arena, the card-table, the magic circle, the temple, the stage, the screen, the tennis court, the court of justice, etc., are all in form and function play-grounds, i.e., forbidden spots, isolated, hedged round, hallowed, within which special rules obtain. All are temporary worlds within the ordinary world, dedicated to the performance of an act apart.” [Johan Huizinga 1938]

This is the first reference to the modern concept of pervasive games. Huizinga recognised (in the 1930’s) that games can build communities, secret of players and potentially overlap into the real world by shifting the game boundaries.

The distinction between allowed real world activity and allowed game activity is picked up again in the 50’s by Gregory Bateson. Bateson identifies the difference between the game and real worlds as interpretations of metacommunication .

“The difference is in metacommunication. Implicit metacommunication frames ordinary actions and playful actions differently. Even though a boxing punch is a punch, it is viewed differently in the boxing ring than a punch on a street” [Gregory Bateson 1955].

This ascribes consequence to how people see the participants of a game and is the mechanism that gives players licence to make actions that are not necessarily normally acceptable behaviour.

Roger Cailliois expanded on this concept by setting some boundaries to Bateson’s definition (later in the 50’s) and also noted the requirement for some mechanism to be in place that made the game distinct from real world activities. He called this mechanism the isolating contractual barrier (Magic Circle) and argued that this barrier must be in place to reinforce the games status in the real world by setting up a special meaning within the game contract.

“Playful activities can be classified on an axis ranging from free play, paidia, to formal play, ludus. Paideic activities include very informal playful activities, such as children’s play, make-believe, riding a rollercoaster, pretend play, and mimicry, whereas ludic activities are well defined and somewhat formal forms of play such as chess or basketball.” [Roger Caillois 1958]

“The function of the isolating contractual barrier is to forbid the players from bringing external motivations and personal histories into the world of game and to forbid taking game events into the realm of ordinary life. While all human activities are equally real, the events taking place within the contract are given special social meanings”. [Roger Caillois 1958]

With the modern phenomenon of global communication (mobile phones and online games in particular) rapidly, there has been a renewed interest in the concept of a Magic Circle. Katie Salen and Eric Zimmerman (2004) revisited the idea of games and play being separate from everyday life and adapted the concept of the Magic Circle from Huizinga’s work. They defined games as:

“autotelic activities since their goal (telos) is self-contained”

In their interpretation of the game construct they see the Magic Circle much more metaphorically, as a conceptual boundary that can be generated by the game not restricting the game:

“shorthand for the idea of a special place in time and space created by a game”

The implication being that it can exist independently in different states and on different platforms and concurrently in different time frames. World of WarCraft is probably the perfect example of this, 12 million users totally absorbed in a fantasy world online, in forums and in real-life.

Greg Lastowka (2007) probably deserves the final word on this topic where he states:

“Conflicts staged within the magic circle are artificial in some sense. When boxers fight in the boxing ring, their conflict is artificial. Though the punches, the pain, the damage, and possibly even the motivation are real, the fight is given an artificial form negotiated by rules. Within the magic circle, different rules apply; lying, backstabbing, betrayal, and limited violence may be acceptable, whereas in ordinary life the same actions would result in serious repercussions”.

So what is this magic circle and why is it important to the Pervasive Game model.

The Magic Circle

The *Magic Circle* (discussed a lot in the literature) is the boundary between game and real world a perfect example is sumo wrestling where all play/rules/rituals happen within the game magic circle.

The Magic Circle is the game boundary that separates the ordinary from ludic [ludic, of or relating to play, playful, play as an act of self-definition] and real from playful.

Recognised places like sports pitches, board games, games location areas where it is acceptable to play a game with the accepted game rules. Normal rules can be suspended and game rules inserted.



[www.traveladventure.org the exclusive proprietor of the image]

Take a moment to think how strange boxing is as a system. Men are allowed to avoid the law and ritualistically beat each other to death so long as they adhere to the game rules.

In the Pervasive Game model the Magic Circle is not a barrier but seen as a membrane. The intention is to traverse this boundary (membrane) between the game and real world in a seamless way. A Pervasive Game will have elements of the game in both the game and real world and aim to blur the distinction between both worlds.

In the perfect Pervasive Game example the player's life would be the game board with the elements of the game accessible across any medium.

“to let game and life bleed together so that game becomes heavy with the reality of life, and life becomes charged with the meaning of game” [unknown quote, I lost the link].

I can bring this into perspective with a small anecdote; I asked a group of students the question, what is your favourite film? This led onto some discussion about how they prefer to play online games (World of Warcraft / Call of Duty); you are in the film not just watching it. It became apparent as we talked that they knew more about each other's avatars and had more shared experience with their avatars than actually real life contact. They were also more animated when they were discussing their avatars (exploits) among themselves than their actual real life activities. Where is the boundary here, very confused in the case of WOW, and as a consequence 12 million paying users involved in this series of extensively designed Magic Circles.

Chapter 3: Pervasive Games

Pervasive Games and Alternate Reality Games (ARGs) are interactive games where the participants drive the game play. The key element in a Pervasive Game is the elimination (if possible) or distorting of the Magic Circle. This is achieved by adding features to a game that expand the contractual Magic Circle of play spatially, temporally and socially.

Spatial Game Elements

Conventional games setup a rigorously defined sphere of influence (board, pitch, PC) that is static and isolates the player from their environment, in the pervasive model the game world can go wherever you go, into the virtual or real world. Players can interact with their environment creating new game play based on how they perceive their current game location.

Temporal Game Elements

Standard games tend to have a fixed time frame; in the Pervasive Game Model a blurring of this fixed game duration is required. Games can be played in different time frames as well as incorporating non-linear elements. A game may last several days with the players' normal life (eating, sleeping, working, and talking with non-players) being taken into consideration and add to the game experience. The game moves from the focus of attention to periphery and back again, or is put on hold until some interaction is required. This has particular relevance to mobile phone games (see tombstoning) where the player may interrupt the game for a variety of reasons (phone call) over extended periods.

Social Game Elements

Traditional games put people into one of two classes, those playing and those not playing. This is distorted in the context of pervasive games. Because the game moves with the player, non-participants in the game can become confused when encountering game players and this element can be used as a game mechanism.

There is a sliding scale of reaction from non-players when confronted by players in a game. This ranges from:

- Being intrigued by the game and joining in to some extent.
- Not actually noticing anything has just happened.
- Thinking they have just been confronted by a crazy person.

There are several warnings in the literature relating to the reaction of non-players encountering a game event; the following quotes express the general mood:

“Due to the lack of voluntariness, the unaware participants are not players. They are not shielded by the protective frame of playfulness” [Suits 1990]

“Unlike aware game players, unaware participants do not limit their actions according to any game rules. Lacking the illusory attitude, an unaware police officer encountering Killer game players would take them down with real violence” [Suits 1990]

The goal of a pervasive game is therefore to provide a seamless experience that moves from the game world to the real world and back again. To make a game experience that seems real without the game mechanism impeding the player interactions or goals.

A common phrase referred to in the genre is *“Pervasive games, like vampires, can only enter your lives if you let them in”*.

Design Considerations for Pervasive Game Development

Traditional Game Play

Traditional games and sports have a very clearly defined set of rules. The game play tends to be in the exploration and exploitation of the game strategies that are allowed by the rule set.

Pervasive Game Play

Pervasive games are more unpredictable than traditional games. Anything and everything can influence the state of a pervasive game.

Game moves do not (and should not) exist in the traditional sense, game actions and real life actions overlap during the course of play. This makes the potential for affordances unlimited, any object encountered can be important to the game, even if accidentally encountered or designed into the game (even if the players recognise the object or not). This has major implications on game play, especially when players confuse real world items as game elements when they are not. This type of ambiguity should be removed (or resolved) whenever possible. Classifying items or setting boundaries on objects and locations are helpful strategies here.

Emergent Game Play

A positive side to affordances is emergent game play. Emergent game play is triggered by affordances and unpredicted perceptions of environments combining to create surprising coincidences and occurrences that have not been planned (by the developer or player) in the game. Emergent game play is noted in the literature as the best part of any pervasive game. When players encounter an emergent game play experience (which has not been planned) it leads to the most intense and fun game experience.

Apophenia

Apophenia [apophenia: the experience of seeing meaningful patterns or connections in random or meaningless data, unmotivated seeing of connections, Klaus Conrad 1958] is regarded as the cause of emergent game play, the experience of mistakenly ascribing meanings to coincidental occurrences which are unrelated or accidental (seeing a face in the landscape of Mars). Apophenic events cannot be designed into a game directly, because they need to be unpredicted and accidental. So an element of ambiguity should be designed into the game that will create the potential for genuine apophenia. Contextual restraints can then be imposed on the design to resolve any ambiguity and bring the game back on track.

Example of Emergent Game Play

I once saw an example of emergent game play while running a game club for a group of students. Several of them were logged onto World of Warcraft and we also had the stereo on in the room pumping out music. One of the students discovered that he could get his character in the game to strip down to its underwear and actually dance (in time to the music in the room) in the game. So he sent a message out to the rest of the students to meet up with his character at a location, a clearing in a wood near a temple. When they got there they all striped down and started to make their characters dance (looked very funny). Within about 10 minutes a circle of hundreds of characters from other players logged on had surrounded them (looked even odder), then they all striped down and started dancing as well (looked surreal). For the next couple of hours the conventional game came to a standstill and they made up their own events, cross country runs, mini Olympics, tag and follow-the-leader with a couple of hundred characters racing around the environment (great fun).

Communal Play

Pervasive games tend to exhibit an element of communal play to a lesser or greater extent depending on the genre. This can have both positive and negative effects, due to a hive mind coming into play to solve problems collectively (encrypted data on websites hacked / clues solved by force of numbers). Designing for this collective approach to the gaming experience can extend the potential of individual players in all areas (creative interaction, fun, problem solving and game emersion). On the negative side game elements tend to be solved quickly and players may find themselves out of sync with other players in the game.

In all of the current literature on the topic, Communal Interaction is noted as a focus for intense play and as the best part of the pervasive game style. Consideration should be given to the fact that, as individuals we are inundated with an endless array of social networks, from local to world-wide and it is therefore inevitable that players will call on these resources at some point in (if not all) the game play.

Real and Virtual World Considerations

Incorporate real and virtual world elements into the game play is required to bridge the gap between game and reality. This dictates that the game will reside on some form of mobile device or in some format that is accessible to all the players in a virtual and real state. The Web, LAN, existing virtual environments or online game platforms are all viable alternative platforms.

The focus in this study will be on using a mobile device as the delivery mechanism.

The following strategies for setting up the game domain/s and play elements complement the use of mobile technology:

- Use of a real world environment as a setting to stage a game, selecting locations with intrinsic meaning or value for the game theme.
- Use of a real world environment to set the context for game elements, landmarks, historical locations, places / items of interest. Solving a puzzle found in the real world to progress to next part of the game in the virtual world (a player may find a locked chest in some location and then need to explore the web or converse with an online community to crack the password). Inversely solving a puzzle found in a virtual world to progress to next the part of the game in the real world.
- Exploit the aesthetics of the environment (a church yard at night) to set the mood.
- Stage live events using actors or teachers to have scripted interactions with players.
- Reporting live events (concerts) or the locations of people (celebrities or players).
- Run game events (treasure hunts etc.) across real and virtual environments.
- Allow players to affect and change the game outcomes / narrative via a feedback mechanism. A game character may re-use new information in the game that was previously gathered from player feedback or player in game choices (prediction game).
- Provision for players and outsiders (ranging from passive spectators to full player participants) to receive varying amounts of information relating to the game, depending on their status within the game.
- It is important that players are able to interact in the style of the game.
- Using the habits of the communication generation (when something is solved everyone knows the solutions) to an advantage, setting the puzzle complexity to community level.
- Using (referencing) existing cyberspace environments and existing game worlds (WOW etc.).
- Generate game-related discussions and role-play on the Web and message boards. Allow group decisions (votes) based on discussions to affect the outcome of the game.
- Use existing media where possible, newspapers, TV, radio and Web (create a fake Web site as a game environment to explore) as a delivery mechanism.

The technical requirements needed to implement these play elements on a mobile phone device are:

- A device with the required specification (smart phone) to allow game events (treasure hunts etc.) to be run across real and virtual environments. The device must also have the facility of retrieving stored game data from a storage media located on the device (isolated storage API's) or a server.
- The availability of a game / graphics API (XNA) on the platform.
- Positioning technology using mapping API's and GPS, Bluetooth and Wi Fi are also options.
- Notification of live events via push notifications.
- Notification of game events (treasure hunts etc.) across real and virtual environments via toast messaging.
- Availability of text messaging services.
- Availability of email services.
- Availability notification services.
- The facility to generate game-related discussions and role-play among players and bystanders on the phone, via text, email and phone call services.
- The facility to generate and save on the device (isolated storage API's) pre-recorded phone messages via voice, wave file, text, post and email services.
- Availability of web browser services.
- Availability of radio receiver services.
- Random number generator accessible via a maths library.

Pervasive Game Genres

Several genres of game are suited to the pervasive game model. The best fit tends to be treasure hunts, assignation games, larps, alternate reality games, smart street sports, performance events, adventure games and reality games.

The applications (test exercises) developed in this study will be based on aspects of the Treasure Hunt genre. The rational being, that this is the most accessible genre for creating the game scenario (the creative element), while also allowing all the technical requirements to be addressed (developed) to implement the required play elements on a mobile device.

Treasure Hunts

Treasure Hunts are basically variations on the theme of finding a series of objects in a game space. The real world, net or any virtual environment is used and the game progresses to the next phase when a clue is solved, item found or task is completed. Prizes range from nothing (kudos of winning) to very large cash awards (Mogi a Japanese game, has massive cash prizes).

Some variations on the theme are:

- *Paper Trails*: follow the clues the find the runner.
- *Letterboxing*: hunt for clues and get a logbook stamped.
- *Geocaching*: find an object at a specific GPS location.
- *Scavenger Hunts*: find a series of items. Some are on a massive scale (In the Amazing Race players compete around the world).
- *Meaning Quests*: gain some insight about yourself (situation) or the future by progressing through certain tasks (ceremonies). Predictors, dating and mystical applications are an example of this style.

Assassination Games

You play an assassin (of some kind), by stalking a target, gathering information and striking. A simple rule set allows many possibilities, played in the day-to-day environment, making everyday life a game. Clues are added to move the level of game play up a notch, the games then take on the characteristics of mysteries (who-done-it).

Pervasive Larps

Pervasive Larps are theatre style games, character-based make-believe, pretend play in the format of live-action role-playing games. They are very popular in Scandinavia, Northern Europe and North America.

Variants of the style are:

- *Historical re-enactments*: Usually of battles, where apposing groups dress in historical costume and re-enact battles (one of the largest is Medieval Week in Gotland where hundreds of players wearing historical costumes play). Sometime sporting and other events are re-enacted.
- *Role-Play*: Based on Dungeons and Dragons, Lord of the Rings and other fantasy books (rule systems). Players dress up as and role-play their characters in elaborately crafted scenarios at dedicated events. Masquerades and Erotic role-play also fall under this category, the context of the event in this case dictating the type of costume, expected scenario, level and type of role-play.
- *Table-Top Role-Playing*: Precursor to Role-Play again based on Dungeons and Dragons, Lord of the Rings and other fantasy books (rule systems). Players role-play their characters (miniature models) in elaborately crafted scenarios (run by a Games Master) sat around a table with maps and other game elements used to aid the imagination.

Alternate Reality Games

Alternate Reality Games are very complex puzzle-based games that require individual and collaborative play by large self-organized player communities (usually internet-based). There is an emphasis on a secretive (conspiratorial) game model to the extent that the first task is often to actually discover the game. A good example of the genre is the *'The Beast'* an Artificial intelligence theme game started as an advertisement campaign for the film *'A.I.'*. A fake company called *'Belladerma'* (sold artificial companions) was set up on the web. Players explored the world created on line and also became involved in real world events (publicity stunts) where actors were used.

Smart Street Sports

Smart Street Sports are based on tactical thinking (heavily reliant on technology) combined with physical challenges (played in the urban environment). They are basically technology-enhanced (cell phone, GPS, Bluetooth) versions of children's games or sports. Very quick to set up as little preparation is required and they are highly re-playable.

Playful Public Performances

Playful Public Performances are very visual performances in urban centres, such as street parties, scavenger hunts, public concerts or political theatre. There is a danger of them becoming uncontrollable and taking on a political or subversive nature (hijacked by groups with an agenda) due to large groups of nonparticipants being left with a confused impression of the event.

Urban Adventure Games

Urban Adventure Games use interactive fiction as a guide for exploring the environment by using stories and puzzles from a script. Very similar to a treasure hunt but the story is the main driving force. The goal is to traverse the scenario. Several works of fiction (Sci-Fi or Horror) have been re-edited to fit a game context and played out. Re-enactments of digital adventure games are also popular.

Reality Games

Reality Games are full-scale pervasive events based in the urban environment. Extensive preparation is put into these events and playing styles from all the other genres are utilised. Locations are completely transformed to fit the game theme (small gothic villages / post-apocalyptic landscape), specialist equipment is designed (surveillance, specialist game items, artefacts) and the scenarios used are very complex. These events run over several days and are very popular in Scandinavia.

Negative Design Considerations

Due to game play taking place in the real world and potentially affecting non-players certain guidelines need to be established for a Pervasive Game design. This will vary depending on the game scenario but the following considerations need to be recognised:

- Moral questions or issues may be raised relating to the players and non-players interactions. Non-players should not be compromised or made to feel uncomfortable in any way. This is dependent on the game script, but setting a vampire larp in a catholic church or a terrorist Urban Adventure on an army base may raise issues from the non-players point of view.
- There is a real danger of non-players misunderstanding the game. This will range from them feeling uncomfortable with the situation to calling the police. Therefore dissuade players from involving non-players. Wearing something recognisable as a game costume (badge) or explaining to people what is happening (when necessary) will help.
- Non-players can become obstacles in both a game and real sense. So players need to be informed to be aware, stay alert and avoid bashing into people, things and others' property.
- Game locations need extensive checking for hidden dangers (at different times of day) before sending players to them. A quiet street at 10:00AM in the morning may be a mugger's paradise at 2:00AM at night. Map locations need to be thoroughly checked, an open space on a map may turn out to be a 200 metre drop off a cliff face.
- Remember that game locations will become an invisible theatre to non-participants, full of unpredictable outcomes; The TV program Candid Camera is a good example, generating unpredictable outcomes in a public place is fraught with issues.

Chapter 4: The Platform

Selecting a Mobile Phone Platform

There are several choices out there Android; iPhone (iOS), BlackBerry, Symbian and Windows 7.5 (Mango), all have the required specification to host a pervasive game.

The criteria for selecting a platform in this study are slightly blurred. The conventional view would be selecting the platform with the best spec and the most users. I need to add in some other considerations as I expect to be dealing with mobile phone development for some time to come. Longevity, accessibility of the development environment, set-up costs, familiarity with the technology, commercial potential and recognising that the device (and operating system) will change considerably over the next few years.

Development Environments

All mobile phone platforms have an associated development environment used to develop, test and release applications. The following chart notes the development features of each of the market leaders. The sections highlighted in bold are areas I am currently familiar with, so they would require less time to integrate into this project.

Platform	Language	Debuggers	Emulator	IDE	Cross-platform	Delivery	Cost
Android	C++	Eclipse, also supports a standalone debugging monitor	Yes	Eclipse, Project Kenai an Android plugin for the NetBeans IDE	Android only, Dalvik virtual machine only supports Android	Android application package file (apk)	Free
Black Berry	Java	Debugger included as part of the IDE	Yes	Eclipse	BlackBerry only, due to the use of the RIM Device Java Library	Compiled source code (cod) and ActiveX Layout File (alx)	Free
iPhone iOS	Objective C	Debugger included as part of the Xcode IDE	Included as part of the Xcode IDE and iPhone Software Development Kit.	Xcode	iPhone, iPad, iPod Touch	Through the App Store, applications need to go through the review process and gain approval by Apple	Intel-based Mac's have free tools, Simulator and test bed. To deploy onto a but device requires registration as a developer, a fee and a developer registration key
Symbian	C++	Yes	Yes free	Designed to be cross platform	Compiled for each target platform	Symbian installation file (sis)	Some basic free tools available is a commercial package
Windows Phone	C# & VB.Net	Yes	Free emulator include in the Integrated Development Environment	Visual Studio 2010 (Express C# , C++ , VB.NET , Web, Phone, Silverlight, XNA and Blend free)	Windows Phone 7.0 and 7.1 (7.5 Mango)	Over-the-air deployment (OTA) and Silverlight application package (XAP) files	Tools, compilers, resources and simulator testing are free. Installing on a device needs a \$100 fee for a developer licence and registration key.

All Platforms met the requirements stated earlier. The deciding factor for me was my current familiarity with the Microsoft IDE and C# development (an additional positive was also the availability of XNA game libraries and Xbox). All of which point to the Windows 7 platform.

Future Trends

Currently the market leader is Android with Symbian and Apple following.

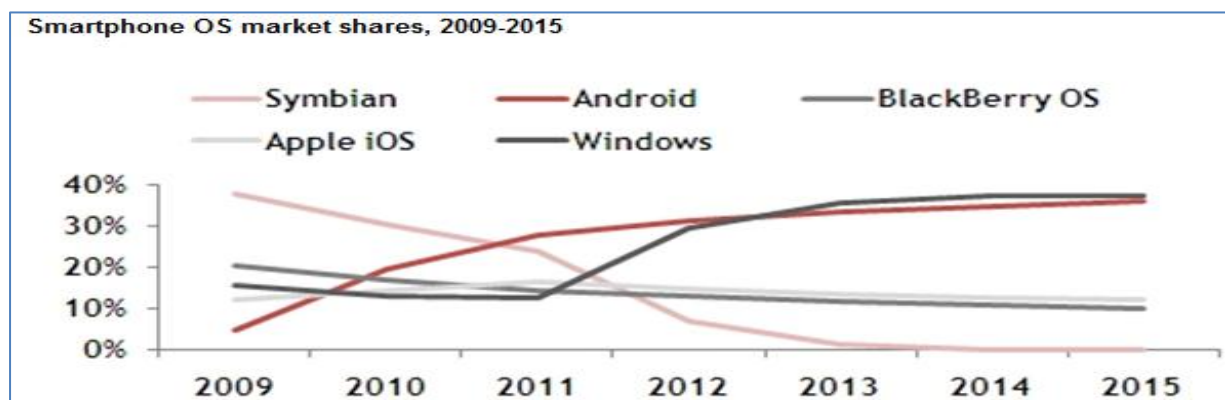
Nokia's decision to move away from Symbian to Windows 7 technology and incentives from Microsoft (benefits for developers, free tools and training) are starting to see Microsoft make an impression on the market.

OS	2010	2011	2012	2015
Symbian	111,577	89,930	32,666	661
Market Share (%)	37.6	19.2	5.2	0.1
Android	67,225	179,873	310,088	539,318
Market Share (%)	22.7	38.5	49.2	48.8
Research In Motion	47,452	62,600	79,335	122,864
Market Share (%)	16.0	13.4	12.6	11.1
iOS	46,598	90,560	118,848	189,924
Market Share (%)	15.7	19.4	18.9	17.2
Microsoft	12,378	26,346	68,156	215,998
Market Share (%)	4.2	5.6	10.8	19.5
Other Operating Systems	11,417.4	18,392.3	21,383.7	36,133.9
Market Share (%)	3.8	3.9	3.4	3.3
Total Market	296,647	467,701	630,476	1,104,898

[Chart from Gartner Research 2011]

This points to a potential for future Windows 7 development as iPhone has 270,000 Apps on the market, Android has 100,000 Apps and to date Windows 7 only 10,000 Apps. From a developers viewpoint this is an opportunity, as there is scope to enter an emerging market.

This seems to be reflected in a recent article from Business Week which points out that there have been *"1.5 million downloads for the development kits. 36,000 registered developers and 11,500 currently released applications. Microsoft is in a massive recruitment drive for developers offering cash incentives, free training, and free software, with 1,200 new developers registering per week."*



[Pyramid Research 2011 Global Smart Phone Research]

Several research organisations (International Data Corporation, Gartner Research and Pyramid Research) have recently released reports predicting a 20-30% capture of the market for Microsoft by 2015, placing them second largest in the world. Please note this is a future projection.

Entry Level Platform

So based on my research I have decided to go for the Microsoft Smartphone Phone (Nokia Lumia 800 top range device, HTC Trophy and Nokia Lumia 710 mid-range devices) and Microsoft 2010 (C#) Integrated Development Environment.

As all the current phones on the market must comply with the following base Windows 7 smart phone features this will be my device entry level specification for the applications (test exercises) developed in this study:

- Phone:
 - Display, 480x800 QVGA / 320x480 HVGA
 - Capacitive Touch, 4 or more contact points
 - Sensors, GPS, accelerometer, compass and light
 - Camera, 5 mega pixels or more
 - Hardware buttons, Start, Search and Back
 - Multimedia, common detailed spec, Codec acceleration
 - Graphics Processing Unit (GPU) DirectX 9 acceleration
 - CPU ARMv7, Cortex/Scorpion or better
- Capabilities:
 - Input, Touch and hardware buttons
 - Media, Digital capture / playback and media library access
 - .Net, Superset of Silverlight 3.0, Windows Commercial Foundation, Simple Object Access Protocol (SOAP) services and Representational State Transfer (REST) web services
 - Phone Access, Integrated access to the phone UI, Sensors and picker for contact / photos
 - Integrated with Cloud services, App deployments and updates, notifications, location and Xbox LIVE



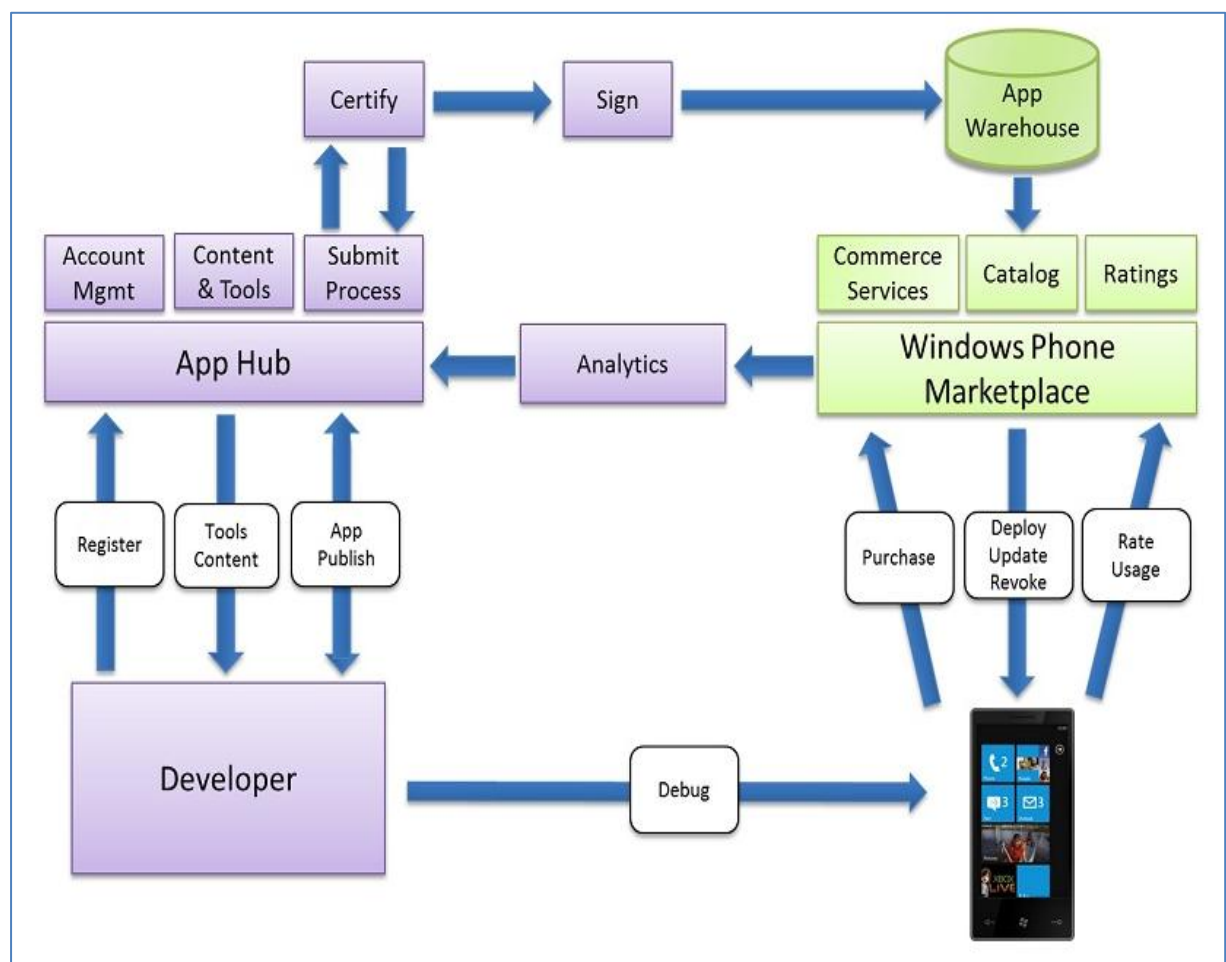
Chapter 5: Setting Up the System

To develop applications on the Windows platform, there are a few registration and licencing requirements that need to be set-up. It is possible to develop applications just using an emulator but they cannot be tested (deployed) on a device until the registration process is completed.

Applications also need to pass approval (compliance) and there are several design constrains that need to be complied with (see Design Guidelines for Windows Phone) before Microsoft will release the application onto the market place.

The process of becoming a registered developer and releasing applications revolves around the App Hub, Microsoft's developer portal (with test deployment via Zune Microsoft's portal to the Marketplace) and incorporates the following steps:

1. Download the libraries, compilers and tools
2. Register as a developer



[Development Lifecycle: Image from Microsoft MSDN: <http://msdn.microsoft.com>]

3. Once you are a registered developer
 - Developing an application (test via Zune)
 - Submitting for certification
 - Deploy to the market place and hopefully receive revenue

Downloading the Libraries, Compilers and Tools

The Microsoft development environment, libraries and tools are available as a massive free download (6 to 10 Gig depending what you download) from the Microsoft Download Centre and App Hub

[<http://www.microsoft.com/download/en/default.aspx>].

For Windows Phone development the Windows Phone SDK is needed, this download contains the following:

- Microsoft Visual Studio 2010 Express for Windows Phone
- Windows Phone Emulator
- Windows Phone SDK 7.1 Assemblies
- Silverlight 4 SDK and DRT
- Windows Phone SDK 7.1 Extensions for XNA Game Studio 4.0
- Microsoft Expression Blend SDK for Windows Phone 7
- Microsoft Expression Blend SDK for Windows Phone OS 7.1
- WCF Data Services Client for Window Phone
- Microsoft Advertising SDK for Windows Phone

It is also advisable to download the following free development environments, libraries and tools this allows projects to be developed on multiple platforms (PC, Web and client / server):

- Zune (for testing and deployment)
- Bing Maps Silverlight Control
- Microsoft SQL Server 2008 and Tools
- Microsoft Visual Web Developer 2010 Express
- Microsoft Visual C# 2010 Express
- Microsoft Visual C++ 2010 Express
- Microsoft Visual Basic 2010 Express
- Windows Azure SDK v1.4
- Windows Azure Tools for Visual Studio
- WCF Services for Visual Studio
- Windows Live

The Registration Process

The App Hub [<http://create.msdn.com/en-US/>] is Microsoft's developer portal for Windows Phone and Xbox360 development.



To register as a developer a Windows Live ID (a new ID can be set up on the Windows Live site [<http://login.live.com/>]), contact details and credit card details are required. Registration costs US\$100 per year and is done via a wizard when you log in with your Windows Live ID. After entering your account type (Phone / Xbox), details, profile and payment details the confirmation process takes place.

You receive an email within 3 days (it was actually straight away) with confirmation of payment, a developer account, and a link to activate the account. Once the account is activated you have your own company area on the web, with application data and payment / earnings details. You can also unlock 3 phones and load up to 10 developer applications per phone, for development testing and deployment (via Zune and the IDE).

The unlock process is fairly straightforward 'The Windows Phone Developers Registration' wizard (`C:\Program Files (x86)\Microsoft SDKs\Windows Phone\v7.1\Tools\Phone Registration\PhoneReg.exe`) is used to unlock and register the phone. For the wizard to complete the registration process a phone needs to be connected, Zune must be running and there needs to be a connection to the web.



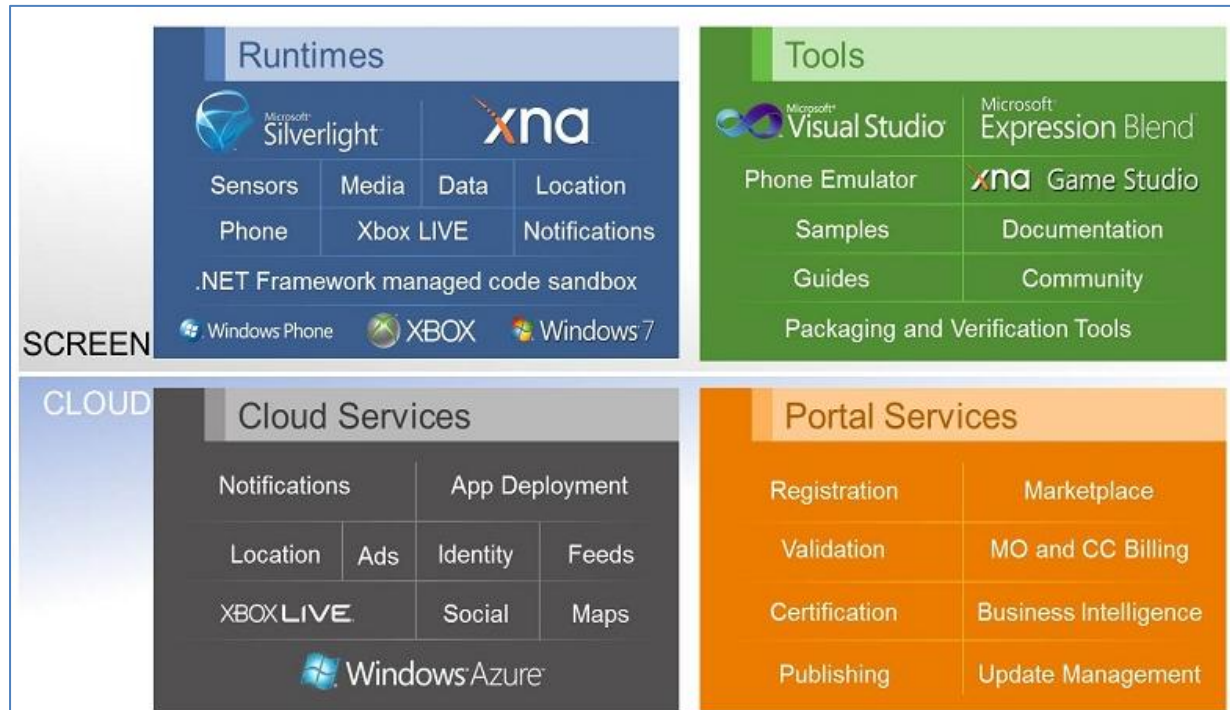
To receive payment for applications, Microsoft take 30% off the top, then you are required to supply a local tax Identification Number and a W-8BEN form (US Tax form) for Microsoft to process payment as a royalty payment.

This does raise a couple of points though. Even though the platform resources are free 36,000 developers are currently paying US\$3,600,000 into Microsoft every year. Also for every \$1 earned, 30 cents go to Microsoft directly, 12.25 cents go to VAT, and 14 cents go to the US government, which leaves 43.75 cents before the NZ government get their hand on it. That said if you sold a million even 10 cents in the dollar would be fine.

As a registered developer I also received the Mango update 7 months before it was released. I was informed via a phone update message to connect to Zune. The installation process was seamless and took about 20 minutes, very nice feature, makes it feel like you are getting a new phone. This had a slight effect on the project time frames as all the libraries, compilers and tools needed to be re-installed and my test applications rebuilding for the new upgrade (taking several days). The phone also needed re-registering as a development phone and the developer applications needed re-deploying.

Platform Architecture

The architecture of the Windows Phone Application Platform comprises four components Runtimes, Cloud Services Tools and Portal Services.



[Platform Architecture: Image from MSDN <http://msdn.microsoft.com>]

Reflecting back on my requirements for developing Pervasive Games this application platform more that covers my criteria. There are also several technologies that it would be interesting to explore in the Cloud and Runtimes (particularly all the Runtimes and the Maps, Location, Feeds and Notifications services).

Portal Services

Portal services allow developer access to Microsoft's App Hub and Marketplace, to register and develop applications.

The following services are provided:

- **Registration and Validation:** Provides registration facilities via the App Hub for setting up developer accounts downloading tools and reregistering development phones.
- **Certification, Publishing, and Update Management:** Access to tools for uploading applications and applying for certification. There are also tracking and notification facilities to monitor progress of a submission.
- **Windows Phone Marketplace and Billing:** The main sales area for applications allowing cost and marketing criteria to be setup.
- **Business Intelligence:** Provides analysis tool providing feedback on the status of application sales and customer responses.

Portal Services	
Registration	Marketplace
Validation	MO and CC Billing
Certification	Business Intelligence
Publishing	Update Management

Tools

The following are the development environments and tools that need to be set up in order to develop Windows Phone applications:



- Visual Studio 2010 Express: IDE for developing Windows Phone applications using Silverlight, XNA, HTML and C#. Express components comes as separate IDE's (it is the free version) so it is best to use Microsoft Visual C# 2010 Express for Windows 7 and Xbox applications. Use Microsoft Visual Web Developer 2010 Express for web servers and Azure servers. Use Microsoft Visual Studio 2010 Express for Windows Phone for the phone component, XNA and locating all the other projects for a final build. This causes less problems linking in services and resources especially for multi-platform project types.
- Expression Blend: Used to design Silverlight (XAML) based pages for the UI of the phone. Nice touch in this platform as this allows a real graphics designer to design the pages, and then a developer can add the functionality in the IDE (C#).
- Windows Phone Emulator: Very nice test emulator to run and debug applications is accessible via Visual Studio and Expression Blend. Be wary of changing the configuration file of this beast, I had to reinstall 3 Gig of software when I inadvertently changed the registration ID and it reconfigured the whole Phone IDE on the next build.
- XNA Game Studio: Extends Visual Studio tools to support the XNA Framework. Used to develop games on the Phone, Xbox 360, Zune HD and Windows 7 platforms. Very flexible API, I will be looking at the 2D functionality (would like to delve into the 3D if time permits).
- Samples, Documentation, Guides and Community: An extensive set of resources to aid development, accessible via help search, the App Hub and MSDN. Some excellent free resources, Microsoft seems to be taking great care to get their training material to a high standard. There is some very interesting videos from the development teams about the framework on the App Hub [<http://create.msdn.com/en-US/>].

Cloud Services

Entry point for web based services and the sharing of data across the Cloud. Using Cloud services save battery live on the device as they are not heavily dependent battery usage.



The following Cloud services are fully supported for phone development:

- Notifications: API's that allow Push Notifications (Raw, Toast and Tile). Very useful API set here allows notifications to groups of registered users.
- Location Cloud Service: Provides location position lookup via Wi-Fi, cellular (device), and GPS data.
- Identity, Feeds, Social, and Maps Services: Web services located in the cloud offering consumer identification, social network interaction, data feeds and mapping features. To use Bing mapping services a developer registration [<https://www.bingmapsportal.com/>] is required, it is free and once registered you receive a unique Bing Maps registration key. You are not charged for the service but your usage is monitored, usage data can be viewed on the account centre [<https://www.bingmapsportal.com/>]. The Bing Maps Silverlight control also needs to be downloaded to add map objects to phone applications.
- Windows Azure: Access to Microsoft's internet-scale Azure cloud computing and services platform (hosted on Microsoft data centres). Beware.... to use this facility seriously, it costs big time, and they have your card details when you register as a developer.

- Microsoft Advertising SDK for Windows Phone: Access to Microsoft Advertising API's, potential here to release applications for free and receive revenue from advertising streams (targeted advertisement). Currently only active in North America, according to the App Hub due to go global in about a year.

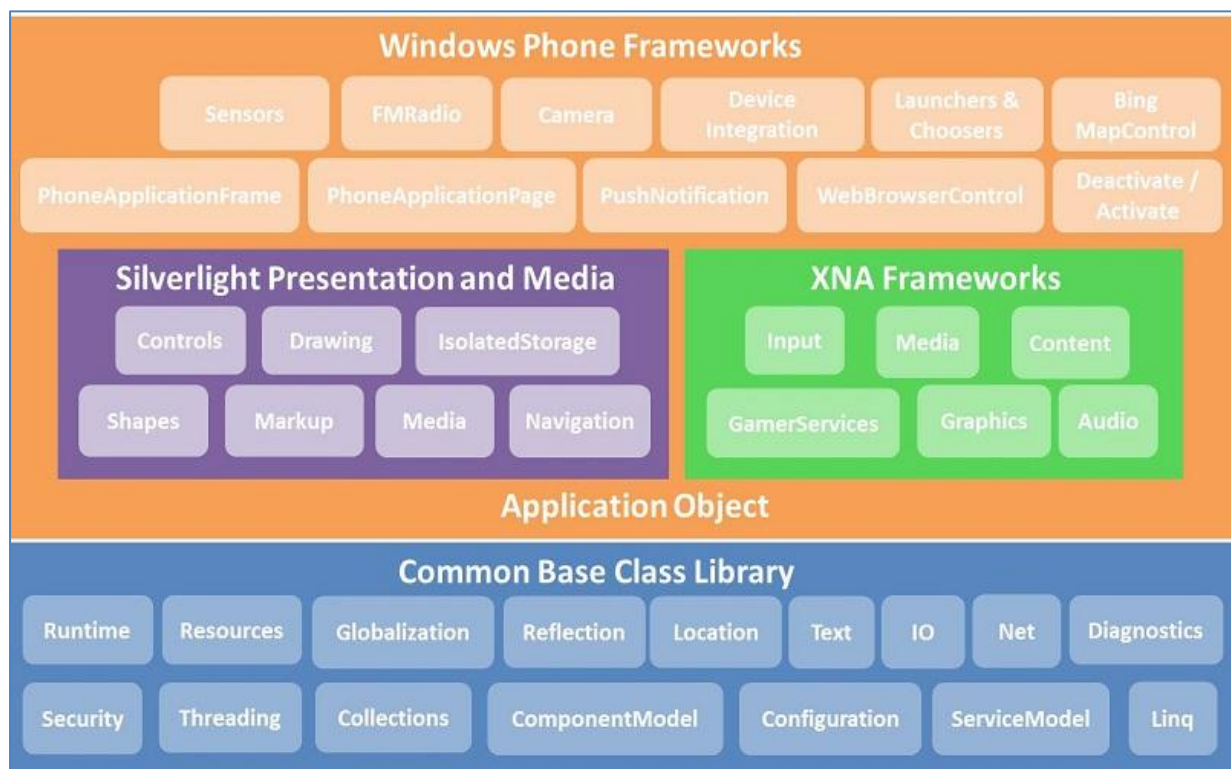
Runtimes

The following runtime frameworks are the base frameworks for Phone development:

- Silverlight: Used to create Rich Internet Application style User Interfaces. A phone application is setup as a series of UI pages written in XAML (.xaml file) then the backend functionality is added in C# (.xaml.cs file).
- XNA Framework: Dedicated set of managed game API's for Windows Phone, Xbox 360, Zune HD, and Windows 7. Supports 2D and 3D API's for games development.
- Sensors: API support for built in sensors multi-touch input, microphone, accelerometer, compass, and gyroscope (not supported on all devices) sensors.
- Media: Managed API's for graphics, animation, and media elements.
- Data: Accessing Isolated Storage, on device 'application specific storage'.
- Location: Device location finding and mapping services via an API set. Location services work in conjunction with the Location Cloud Services.



Important Note: Implemented in the new upgrade Mango (7.1 builds), applications can be made as both Silverlight and XNA, allowing Silverlight objects on the page that runs in an XNA game loop (great feature from a development point of view).



[Runtime Frameworks: Image from MSDN <http://msdn.microsoft.com>]

Problems and Issues

The whole process of setting up the environment is fairly straight forward although time consuming.

The separate components work well together and all interaction with Microsoft has been seamless, very quick response times and free (apart from the US\$100 registration fee).

Having a centralised location for developer account details tends to be a bit spooky. If you show an interest in one area or update the environment Microsoft will contact you with other relevant updated and services, they seem to be pushing development on the platform actively. So updating my development environment can trigger my phone to tell me about a developer update for the OS downloadable via Zune.

My main stumbling block is non-technical and relates to taxation documentation in the US.

On a personal note Microsoft having my credit card details is a bit of a worry, especially with the amount of (free) software I have been downloading from their site.

Design Guidelines for Windows Phone

Before an application can be released onto the market, it must pass certification. This involves Microsoft checking a submission for compliance against their design guidelines, technical requirements and policies.



The main concerns of the certification process are:

- The UI design is Metro (Microsoft's name for a typography-based design language) oriented allowing consistency of user experience across applications and platforms.
- Applications are bug free and reliable. Applications must show a splash (SplashScreenImage.jpg) screen within 5 seconds of launch and not take any longer than 10 seconds to fully load (an important consideration for Microsoft this one).
- All resources must be used efficiently and de-allocated (also services closed down) on exit.
- Phone functionality must not be compromised in any way, all configuration changes and device data access needs to be approved by the user.
- Applications must be free from malicious software and intent. They must not be offensive (ethically, morally, religiously or explicit sexually) in any way or breach existing laws.

The documentation is extensive and needs checking prior to each release as various sections relate to the type and nature of the application. The following is a list of the compliance documentation for reference:

- Application Policies [<http://msdn.microsoft.com>]
- Content Policies [<http://msdn.microsoft.com>]
- Application Submission Requirements [<http://msdn.microsoft.com>]
- Technical Certification Requirements [<http://msdn.microsoft.com>]
- Additional Requirements for Specific Application Types [<http://msdn.microsoft.com>]

Metro Design

Metro is Microsoft's name for a typography-based [Typography: the technique of make language visible by arranging type in order] design language used for Phone interfaces, Zune Marketplace and Xbox dashboard. It will also be a feature of Windows 8 and future releases of Windows Live and the Xbox 360. The principle is a clean metropolitan (globally recognisable images/icons) design and a consistent user experience across platforms.

To implement applications that conform to a Metro designed the following UI elements (Graphics, Visual Indicators and Notifications) need to be correctly implemented.



[Metro Design: Image from MSDN <http://msdn.microsoft.com>]

Start Tiles

Applications should always offer a start tile. Tiles need to be an easily recognizable visual shortcut for an application, located in the phone's Start view. Tiles need to give some information about the application as well as making the application brand visible.

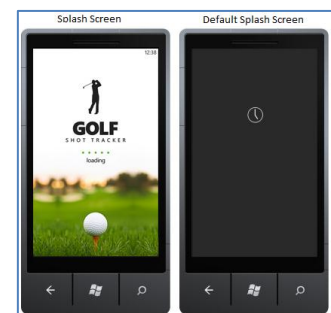
Tile images must be 173 pixels x 173 pixels 256 dpi (JPEG or PNG) format. Note the Start Tile image is also scaled down (63 x 63) for the image on the application list.



Splash Screens

Applications must show a splash (SplashScreenImage.jpg) screen within 5 seconds of launch and not take any longer than 10 seconds to fully load, otherwise the application will terminate.

The splash screen must be a graphical advertisement of the application. The recommendation is not use the default splash screen for an application.



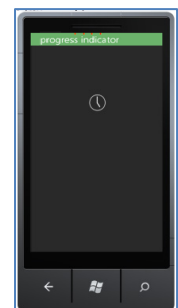
Visual Indicators

The use of visual indicators (Progress Indicator, Scroll Indicator and Scroll Viewer) must remain consistent with the Metro design principle.

Progress Indicators are displayed in the status bar allowing display across applications and must be reserved for system operations, they are not user actionable.

Scroll Indicators are used as an indication about the page size and are not user actionable.

Scroll Viewers are used to indicate that navigation to hidden content is possible and are not user actionable.



Status Bar

A Status bar is used to display automatically updated system-level status information in a reserved portion of the application workspace at the top of the phone display. It can be hidden, but users must be allowed access to it if they want to see it. The system clock is always visible by default and when a user taps the Status Bar all other active indicators slide into view. They are displayed for about 8 seconds.

The following indicators are displayed left to right if active:

- Signal strength
- Data connection
- Call forwarding
- Roaming
- Wireless network signal strength
- Bluetooth status
- Ringer mode
- Input status
- Battery power level
- System clock

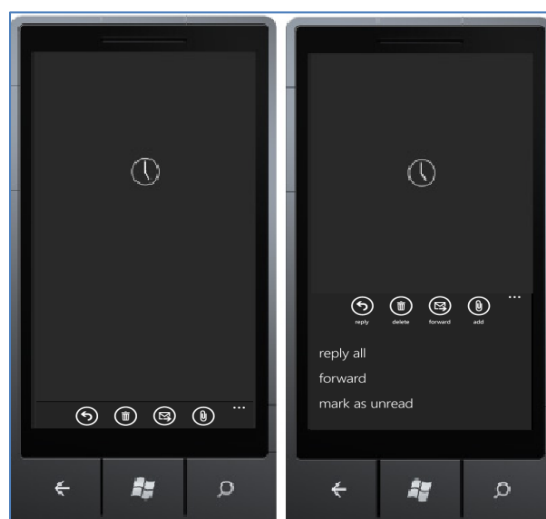


Application Bar

The Application Bar allows up to four application tasks to be displayed as icon buttons with associated text hints. An optional context menu (Application Bar Menu) can be set up (maximum of five entries) so that when a user taps the visual indicator of sequential dots (.....) or flicks up the Application bar, it is activated. The Application bar menu will remain on screen until the user performs an action and is deactivated by tapping on the dots (or outside of the menu area or selecting a menu item).

The recommendation is that the application bar should be used in preference to a developer designed menu, to stay consistent with a Metro design. As a bonus (for using) the Application bar has built in animation and rotation support.

Opacity is provided but the settings (0 to 1) affect the application display area (left image). A setting of 1 resizes the display area (right image) to a screen size not covered by the Application Bar (72 pixels). A setting of less than 1 sets the display area to the size of the screen with the Application Bar overlaid on the bottom of it.



Application Bar Icons

Application Bar Icons must be clear, understandable and represent real-world metaphors consistent with the Metro design. Icons can be displayed as enabled or disabled, must be 48 x 48 pixels and drawn as a white foreground on a transparent background (using an alpha channel).

There is a basic standard set provided in the `C:\Program Files\Microsoft SDKs\Windows Phone\v7.0\Icons` folder.

A restriction on using the Application Bar is that an Application Bar Icon must not be used as a Back button, in all cases the hardware back Button must be used for backwards navigation.

Application Bar Icons are rotated when the phone is turned so this rotation needs to be taken into account when designing icons.



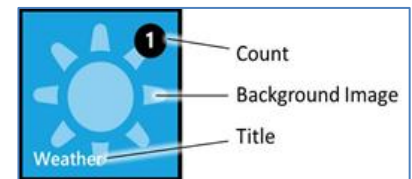
[Standard Icon Set: From MSDN <http://msdn.microsoft.com>]

Notifications

Cloud services can be used to send a push notification to a phone. Push notifications are in the format of Tile, Toast, or Raw notification.

Tile Notifications

The Start Tile of an application can receive a tile notification. When received the tile is updated with a new count, title and background image. This allows scope for a graphic representation of a change of state of an application, when the application is not in use.



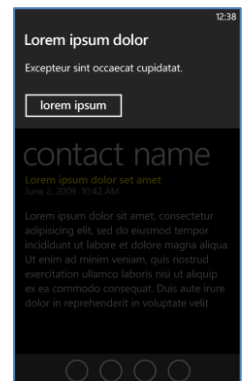
Toast Notifications

A toast notification can be sent in the format of two text strings, displayed at the top of the screen. Toast notifications should be used to request some action from the user, there is a facility so that when pressed a link to a page in the application can be activated (very useful option). A toast message will only stay visible for 10 second and no record of the message is kept on the phone. By default toasts are turned off on a device so the user must be asked for permission to active this service.



Raw Notifications

A Raw notification can be sent to an active application in the format of a text string. Raw notifications should be used for in-application information, data or status updates or notifications that require the user to take some action.



On Network Outage

When the device loses contact with the network the application must be sufficiently robust to facilitate the following recommendations:

- An appropriate warning should be displayed.
- Navigation away from the application should still be possible as a base level. Functional navigation within the application is also recommended.
- Data storage and retrieval process should be accessible locally on the device.
- Information about connection problems or related problems must be indicated to the user where possible.

Include an End-User License Agreement

An end end-user license agreement (EULA) should be laid out for the user when they first launch (purchase from the market place) the application.

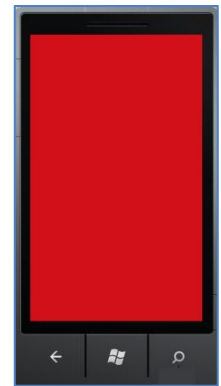
This is a set of guidelines for the use of the application and users rights as the developer stipulates. The application can only be used when the user agrees to the EULA. The following quote is taken from the Microsoft licencing section of the App Hub and outlines the requirements of the EULA *“The EULA statement should discuss behaviours and uses of your app that you (the developer) endorse, including specific terms about content, license, installation, activation, validation, Internet-based services, or the use of certain information. It's also a good place to inform users about your plans to update or offer upgrades to the app, or your use of software formats and standards. If your app is associated with a business, your EULA should also indicate where your business is registered, and whether you offer any kind of warranty.”*

Chapter 6: Game Engine

This section focuses on my investigation into issues (and coding solutions) relating specifically to the development of a pervasive games engine (67 test programs) on the Windows 7 platform.

The engine will utilise existing API's and where necessary functionality will be developed.

There is an assumed familiarity with basic C#, .Net and Windows based application development principles, as a discussion of these topics are not in the scope of this project.



All development was implemented using the Windows Phone SDK and associated tools outlined in the previous section.

The basic configuration of builds and libraries required to implement the coding examples noted in this section are:

- Windows 7.1 Phone development:
 - Microsoft Visual Studio 2010 Express for Windows Phone, with the Windows Phone SDK 7.1, Silverlight 4 SDK and Windows Phone SDK 7.1 Extensions for XNA Game Studio 4.0 assemblies and libraries.
 - Microsoft Expression Blend SDK for Windows Phone OS 7.1
 - Bing Maps Silverlight Control
- Multi-platform developed (client / server):
 - Microsoft Visual Web Developer 2010 Express (server interface)
 - Microsoft Visual C# 2010 Express (multi project builds)
- Testing and deployment
 - Windows Phone Emulator (testing)
 - Zune (device deployment and testing)
 - Active Link to the web (device activation)

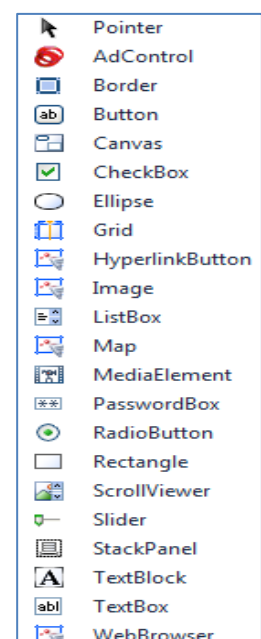
Basic Silverlight Phone Applications

Windows 7 development follows a task-centric philosophy, events are triggered by user interaction, the operating system or services, and then the associated event code is implemented. The whole process is required to be a seamless interactive experience for the user.

Silverlight (development add in tool) is used to create UI pages written in XAML (.xaml file).

There is a set of standard controls (and downloadable extended controls) with a Metro feel that can be dragged onto a Phone page to build up applications.

XAML code is generated automatically when a control is added (or you can write the XAML code directly), associated events can be attached to an object and C# is used to implement the functionality (.xaml.cs file).

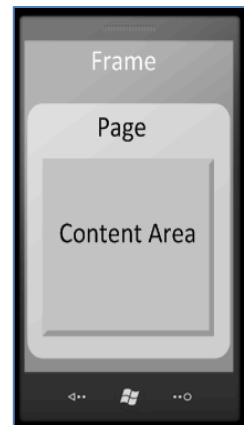


Page Format

A basic Page is made up of a Frame, Page, Content Area and hardware Back button.

The Frame control is the top level container (PhoneApplicationFrame class) which contains the Page Control, System Tray and Application Bar.

The Page (PhoneApplicationPage derived class) fills the whole content region of the frame. Inside the Page is a title and Content Area, where application specific content is placed.

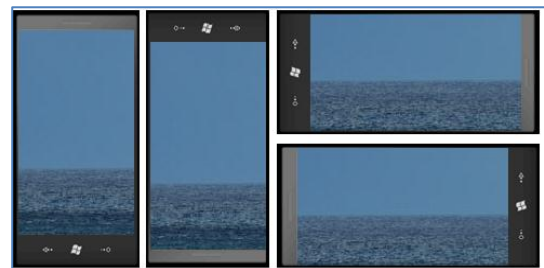


Page Orientation

A serious design consideration is that Windows 7 Phone applications should be orientation aware.

Applications may be required to run in any of the following orientations:

- Portrait Up (480 * 800).
- Portrait Down (480 * 800).
- Landscape Left (800 * 480).
- Landscape Right (800 * 480).



The following XAML syntax is used to set page orientation:

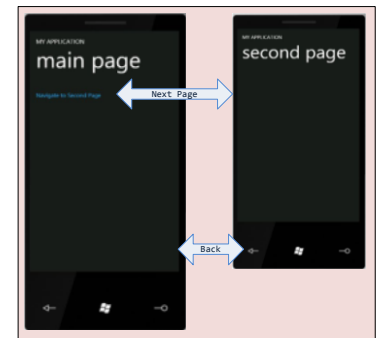
```
SupportedOrientations="PortraitOrLandscape" Orientation="Landscape"
```

Code Listings: The following code listings provide complete coding solutions to the Page Format section along with some basic getting started code.

Basic Silverlight Applications	Notes
001 HelloPhone	A basic Hello World application using Blend to design the interface.
002 SplashScreen	Displays a Splashscreen for 3 seconds.
003 DamageCalc	A basic tax calculator application using the Text Change Event.
004 OrientationSample	An orientation aware page example.
005 OrientationGrid	An orientation aware page example using a grid format.

Page Navigation

Navigation between pages on Windows 7 phones is a page-based navigation model similar to a web page model. Each page has its own URI (Uniform Resource Identifier) and navigation takes place with a call to the `NavigationService.Navigate` method.



The following syntax is used for Uri navigation:

```
NavigationService.Navigate( new Uri("/SecondPage.xaml", UriKind.RelativeOrAbsolute)
```

The hardware Back button is (the recommended option to be) used to navigate back a step through the navigation stack or a call to `NavigationService.GoBack()` method can be used.

Code Listings: The following code listings provide complete coding solutions to the Page Navigation section.

Basic Silverlight Applications	Notes
006 Navigation	Page navigation, back button override and navigation example.
007 NavigationAndControls	Page navigation and objects, images, music, video and web link example.

Panorama

Where possible a panorama page presentation structure is recommended (Metro style). The format allows the user to flick across an extended first page of the application section by section. Setting up a Panorama application is relatively simple as it is one of the project type option presented by the IDE when the project is being created. Page navigation principles are the same as a normal page setup it is just the organisation of the links into sections that is different.



The following syntax is used to set up a Panorama page:

```
<!--Panorama item one-->
<controls:PanoramaItem Header="first item">
  <!--Double line list with text wrapping-->
  <ListBox Margin="0,0,-12,0" ItemsSource="{Binding Items}">
    <ListBox.ItemTemplate>
      <DataTemplate>
        <StackPanel Margin="0,0,0,17" Width="432" Height="78">
          <TextBlock Text="{Binding LineOne}" TextWrapping="Wrap"
            Style="{StaticResource PhoneTextExtraLargeStyle}"/>
          <TextBlock Text="{Binding LineTwo}" TextWrapping="Wrap" Margin="12,-
            6,12,0" Style="{StaticResource PhoneTextSubtleStyle}"/>
        </StackPanel>
      </DataTemplate>
    </ListBox.ItemTemplate>
  </ListBox>
</controls:PanoramaItem>
```

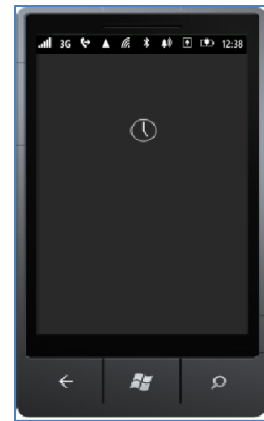
Code Listings: The following code listing provides a complete coding solution to the Panorama section.

Basic Silverlight Applications	Notes
008 PanoramaApplication	Basic Panorama page setup example.

System Tray

The System Tray is system owned and non-programmable, however there is control over its display status.

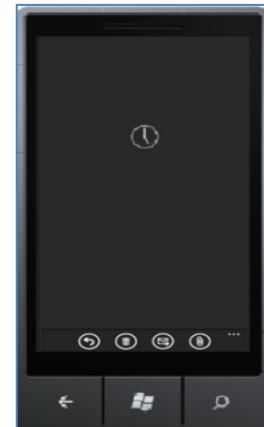
An application can allow the System Tray to be shown or hidden by setting the `Microsoft.Phone.Shell.SystemTray.IsVisible` attribute to `true` or `false`.



Application Bar

Every page comes with a default Application bar set up (commented out) so it is relatively simple to use the code as a template. There are some implementation issues that are not documented, the Build Action on icons needs to be set as Content and icons will not display (no exception generated) if the `IconUri` path is incorrect or the format is wrong (48 x 48 pixels, drawn as a white foreground on a transparent background using an alpha channel).

The Application Bar should be the main navigation and option menu on a page so the requirement is to keep things simple, not too many options on a page.



The following syntax is used to set up an Application Bar:

```
<!--Sample XAML code showing usage of ApplicationBar-->
<phone:PhoneApplicationPage.ApplicationBar>
  <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True" Opacity="0.5">
    <shell:ApplicationBarIconButton IconUri="/Images/appbar.expand.rest.png"
      Text="Close Bar" Click="Close_Bar_Click"/>
    <shell:ApplicationBar.MenuItems>
      <shell:ApplicationBarMenuItem Text="Increase Opacity"
        Click="MenuItem_Click_Increase_Opacity"/>
    </shell:ApplicationBar.MenuItems>
  </shell:ApplicationBar>
</phone:PhoneApplicationPage.ApplicationBar>
```

The following syntax is used for the associated events in this example:

```
private void Close_Bar_Click(object sender, EventArgs e){ }

private void MenuItem_Click_Increase_Opacity(object sender, EventArgs e){ }
```

Code Listings: The following code listings provide complete coding solutions to the System Tray and Application Bar sections.

Basic Silverlight Applications	Notes
009 ApplicationBarBlend	Basic Application Bar set up using Blend.
010 ApplicationBarXAML	Application Bar (XAML code) with close, opacity, menu and next options.
011 ApplicationBarSampleC#	Application Bar (C# code) with close, opacity, menu and next options.
012 ControlsTrayAppBar	Page navigation and Application Bar example. Example of the System Tray, objects, images, music and video.

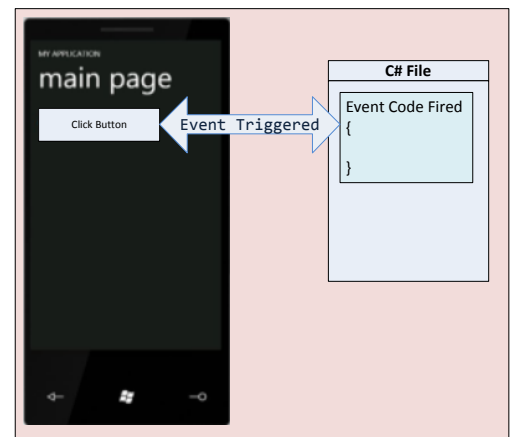
Events

Events contained within a class are subscribed to when specific code is written that is called when the event is raised.

Events can be raised in numerous ways, it may be when a user clicks a button, an object on a page is updated or you define your own event.

There are several formats that can be used when setting up an event.

I will look at these in some detail because this has caused me a little confusion while getting to grips with this environment, as every reference source and tutorial I have found tends to use a different format (with no explanation as to why).

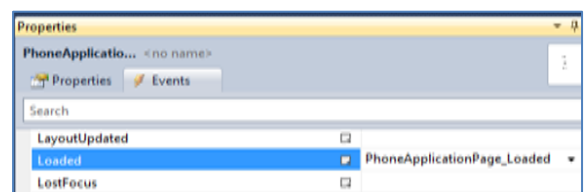


The following are for my reference and will be used as templates for future development.

Subscribe to events using the class properties window and events tab

This method utilises the auto generate facilities of the IDE.

Select an event, in this case the `Loaded` event, then associate an event to it by selecting the event list and adding a name for the event (clicking will generate a default name).

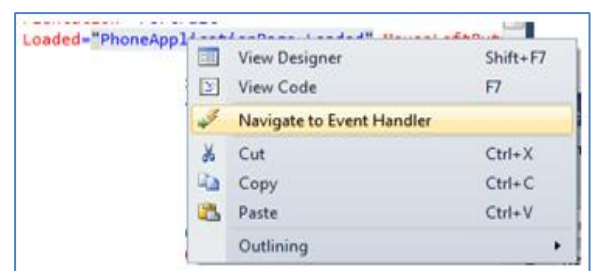


This will auto generate the event to the XAML description of the object.

The following syntax is used to define the event:

```
Loaded="PhoneApplicationPage_Loaded"
```

To auto generate the C# event handler code in the associated .xaml.cs file, double click the event name in the list or right click the XAML entry and select `Navigate to Event Handler` option to see the event code.



The following syntax is used for the associated event handler code in this example:

```
private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e) { }
```


Subscribe to events programmatically

An event can be set up in code (C#) by the following process.

In the constructor of the class attach a new event to the required event, by using the addition assignment operator (+=).

The following syntax is used to define the event:

```
this.MouseLeftButtonDown += new MouseButtonEventHandler(MainPage_MouseLButtonDown);
```

Add the event method

The following syntax is used for the associated event handler code in this example:

```
private void MainPage_MouseLButtonDown(object sender, MouseButtonEventArgs e) { }
```

Note: Visual Studio 2010 will render the default event handler by pressing the Tab key twice after += in the event subscription. This will auto generate both the definition and the event method stub. The parameter types of MouseButtonEventHandler (the event) can be found by laying the mouse over the event name (before making the event code block).

Backwards compatibility issues

- The following syntax is the C# 1.0 compliant version:

```
this.MouseLeftButtonDown += new MouseButtonEventHandler(MainPage_MouseLButtonDown);
```

- The following syntax can also be used for C# 2.0 and onwards:

```
this.MouseLeftButtonDown += MainPage_MouseLButtonDown;
```

Subscribe to events programmatically using => a lambda expression

An event handler can be added using a lambda expression =>

The following syntax is used when utilising the lambda expression:

```
this.MouseLeftButtonDown += (s,e) =>
{
    textBox4.Text = ((MouseButtonEventArgs)e).ToString();
};
```

Note: The types of the inputs (Object and MouseEventArgs) are inferred by the compiler and do not have to be explicitly given in the lambda input parameters.

Subscribe to events by using an anonymous method

This format can be used when the event does not need to be unsubscribed from later.

Anonymous methods are useful for reducing the coding overhead in instantiating delegates, because they eliminate the need to create a separate method. This may be of interest later in this project if I need to optimise any events.

The following syntax is used to subscribe to an event by using an anonymous method:

```
this.MouseLeftButtonDown += delegate(object sender, MouseButtonEventArgs e)
{
    //textBox5.Text = ((MouseButtonEventArgs)e).ToString();
    textBox5.Text = sender.ToString();
};
```

Unsubscribe from an event

To unsubscribe from an event the subtraction assignment operator -= is used.

The following syntax is used to unsubscribe from an event:

```
this.MouseLeftButtonDown -= MainPage_MouseLeftButtonDown;
```

Subscribe to events by using the EventHandler<TEventArgs> delegate

Events can be created specifically for various classes using the generic EventHandler delegate.

The <> denotes the type of Event Arguments that contain the event data.

If the event has no event data the generic EventArgs event type parameter can be used.

MS IDE Tip: The correct event data will be auto generated by using the tab - tab option after adding the += in the code.



```
map1.ViewChangeOnFrame +=  
new EventHandler<MapEventArgs>(map1_ViewChangeOnFrame); (Press TAB to insert)
```

The following syntax is used to subscribe to events by using the EventHandler delegate:

```
this.BackKeyPress += new  
EventHandler<System.ComponentModel.CancelEventArgs>(MainPage_BackKeyPress);  
  
private void MainPage_BackKeyPress(object sender, System.ComponentModel.CancelEventArgs e) { }
```

Note: the same event data type System.ComponentModel.CancelEventArgs has to be used in both.

Adding levels of complexity to subscribing to events

The following is an example of some geocode functionality that is a little more complex syntactically; notice the embedded path to the type data. It is here for my reference as a template for future coding solutions.

```
GeoCoordinateWatcher w;  
w = new GeoCoordinateWatcher(GeoPositionAccuracy.High);  
  
w.StatusChanged += new EventHandler<GeoPositionStatusChangedEventArgs>(watcher_statusChanged);  
  
w.PositionChanged +=  
new EventHandler<GeoPositionChangedEventArgs<GeoCoordinate>>(watcher_PositionChanged);
```

Memory and Resource Management with Events

In the Windows 7 environment it is very important that events are released when moving from one page to other pages and then back again. Otherwise multiple events will be generated overriding each other and considerably tying up resources by overwriting memory.

It is also a prerequisite for the application compliance process that all resources are freed when not required further.

The following process can be used as a template for allocating and freeing (via the hardware back button) event resources:

```
// Object representing
private Microphone microphone = Microphone.Default;

// Timer to simulate the XNA Framework game loop
DispatcherTimer dt = new DispatcherTimer();

public RecordAlarmPage()
{
    dt.Tick += new EventHandler(dt_Tick);

    // Event handler for getting audio data when the buffer is full
    microphone.BufferReady += new EventHandler<EventArgs>(microphone_BufferReady);

    // Back key pressed use to switch off the events when leaving the page
    this.BackKeyPress +=
        new EventHandler<System.ComponentModel.CancelEventArgs>(RecordAlarmPage_BackKeyPress);
}

void microphone_BufferReady(object sender, EventArgs e) { }

void dt_Tick(object sender, EventArgs e) { }

// Back key pressed use to switch off the events when leaving the page
void RecordAlarmPage_BackKeyPress(object sender, System.ComponentModel.CancelEventArgs e)
{
    dt.Tick -= new EventHandler(dt_Tick);

    microphone.BufferReady -= new EventHandler<EventArgs>(microphone_BufferReady);
}
```

Code Listings: The following code listing provides a complete coding solution to the Events section.

Basic Silverlight Applications	Notes
013 Events	Template for setting up events in various formats.

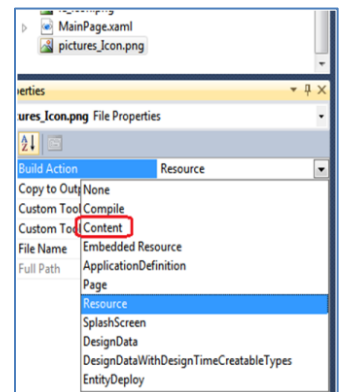
Problems and Resolutions

Resource Files

For something that seems fairly straightforward, the reality is, including resource files (image, audio, and video files) into a Windows 7 project is slightly tricky.

The problem is the type of build action and location string of the file must be correct for the application to build the resource correctly.

The issue stems from fact that the IDE automatically allocates a build option for a file that is generated by the environment (defaults correctly), but when a developer adds resources to the build it is difficult to set the correct option. This is because there is very little documentation on what type of build action is best for a specific resource type (and platform), the fact that the resource can be linked to the application in several way (properties, XAML, C# and direct links) adds to the issue.



So what are the Build Action options?

Resource files can be deployed as individual files, in a package (XAP file) or retrieved on demand (referenced from its original location). They can also be deployed as embedded files, in the assembly application package (exe), an external library package (.dll) or retrieved on demand (referenced from its original location).

The build options available for Silverlight resource files are:

- **Resource:** Embeds the file in a project assembly (recommendation is use with application and library projects) and deploys the resource assembly inside or outside the build package.
- **Content:** Includes the file in an application package without embedding it in a project assembly (recommendation is use for resources shared by multiple assemblies in the package).
- **None:** Does not include the resource file in the application package or assembly (recommendation is use for resources that are retrieved on demand). On-demand resources must be deployed at the same server location as the application package.

XNA, .NET, Web and Cloud services also offer additional options as well:

- **Compile resource option:** The resource is compiled into the build output (dll or exe), normally used for code files.
- **Embedded resource option:** The resource is compiled into the assembly (dll or exe).
- **Site of origin resource option:** The resource is referenced from its original location.

There are also a couple of overrides to the list:

- The Embedded Resource option must not be used if a project has a Silverlight component.
- The Resource option must be used for multi-platform projects.

So how are they linked in as resource URI formats?

A resource is referenced in C#, XAML or as a property using a URI (Uniform Resource Identifier), and the format is specific to the how the file is referenced.

The URI format can be referenced as relative or absolute:

- An absolute URI is used to retrieve resources externally from the application package domain. The URI denotes the exact location of a resource as a link to the resource file on a specific domain, for example `http://www.paul.com/resources/picture.png`.
- A relative URIs denotes the location of a resource file relative to the application package root (application folder) or the referencing XAML (page).
 - A leading slash is used to reference an application root folder.
For example `/resources/picture.png`.
 - XAML is references without using a leading slash or if the resource is located in a folder outside the XAML files folder it can be referenced relative to the application root.
For example `../resources/picture.png`.
 - Resource files that are embedded in other assemblies (in the same project) can also be referenced by XAML, using the following format
`/assemblyShortName;component/resourceLocation`.

What is my solution to this problem?

To make life easier for future development and in an attempt to solve this ambiguity I will update the following Resource Options Quick Reference Table with the correct resource configurations as I come into contact with each standard resource type.

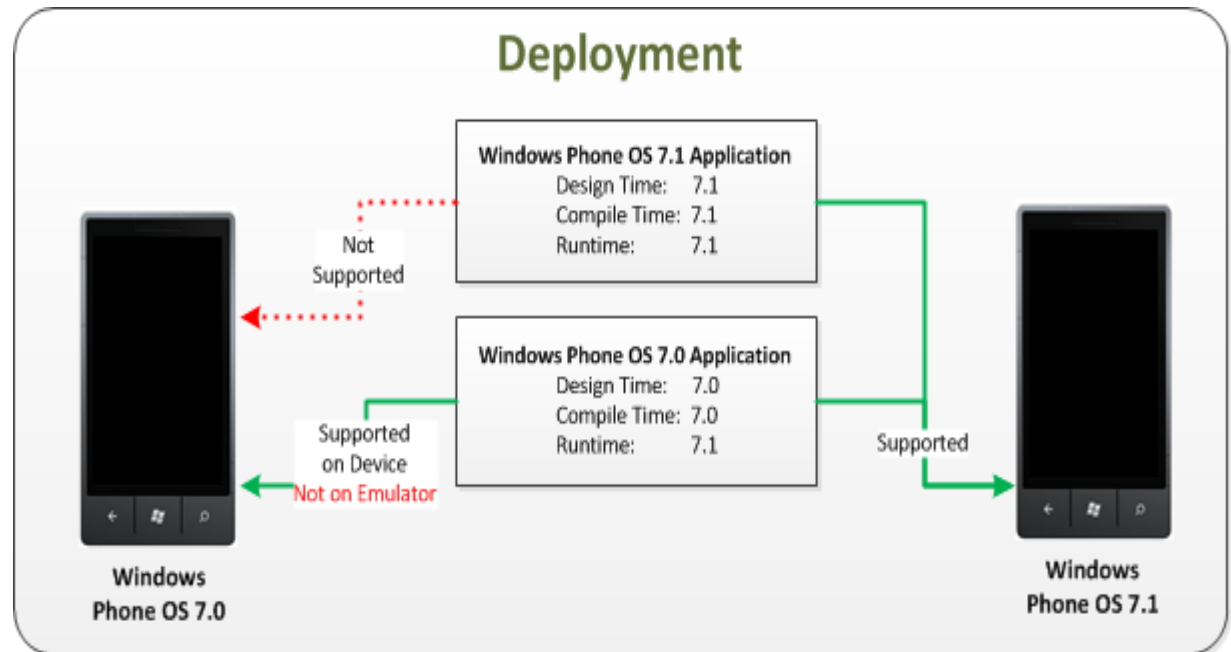
Resource Options Quick Reference Table

File Type	Type	Build Option	URI Link Type
Application icons	.png	content	relative
Application backgrounds	.png	content	relative
Splashscreens	.jpg	content	None, defaults to the SplashScreenImage.jpg file
Image object image	.jpg	Resource	Properties link relative
File link sound	.mp3	content	Properties link relative
File link video	.wmv	content	Properties link relative
Panorama background	.jpg	Resource	ImageBrush ImageSource link relative
Initialization file XAML format	.dat	Resource	relative
Bitmap Image object image	.png	content	RelativeOrAbsolute
XNA 2D texture	.png	compile	Content.Load<Texture2D>("image")
Push Pin Background Image	.jpg	Resource	ImageBrush ImageSource link
Push Pin Image source	.jpg	Resource	BitmapImage source RelativeOrAbsolute
Video from web link	.wmv	@""	Web link RelativeOrAbsolute
Video in map	.wmv	content	/folder/name.wmv RelativeOrAbsolute
Application Tile Image	.png	content	Relative
Application Tile Image	.png	@""	Direct Web link
Secondary Tile Image	.png	content	Relative
Sound effect from a stream	.wav	content	TitleContainer.OpenStream("Folder/File.wav")

Upgrade of the Operating System

During the first stage of this project I was working on the Windows 7.0 OS (both phone and emulator) this was subsequently upgraded to Windows 7.1 OS.

The following diagram depicts the compatibility problem that this upgrade enforced.



[New Deployment: Image from MSDN <http://msdn.microsoft.com>]

Obviously because 7.1 applications are not supported on the 7.0 OS development environment and because the phone device had been upgraded all the 7.0 examples needed converting.

The literature on the upgrade also warns that the event calling order of events may have changed, but there is no specific example as to how or which events this effects. So far I have had no compatibility issues in this area.

Currently I am working on the Windows 7.5 OS Mango (both phone and emulator), the upgrade caused no problems and utilises the 7.1 build in the compiler.

Error Checking and Messaging

There are circumstances when an application needs to report on or log its current status. This may be to a user when there is a required action needed, as a warning, an error message / log or for some system / device related reason (network failure).

From a developers point of view a method of diagnostic information retrieval and custom error tracing is required. There are several options available to provide this feedback; the following seem to be the most accessible mechanisms (based on testing various coding options), Error Pages, Message Boxes, Logging methods and Trace methods.

Messages and Debugging

For simple messages and debugging data a call to a text block and the `debug.WriteLine` method will directed output to the phone screen and debugging output window.

Note: The `System.Diagnostics` class needs to be included into the build.

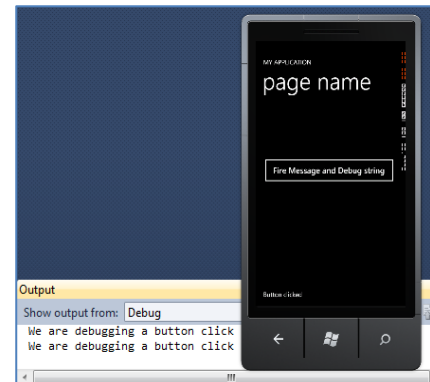
The following syntax is used for simple messaging:

```
using System.Diagnostics;

private void UpdateStatus(string message)
{
    txtStatus.Text = message;
}
private void Trace(string message)
{
    #if DEBUG
        Debug.WriteLine(message);
    #endif
}
```

Code Listings: The following code listing provides a complete coding solution for this section.

Error Checking & Messaging	Notes
014 Messages	Simple debugging and messages example.



Error Pages

It is possible to utilise the exception handling mechanism built into C#, by displaying to a Page (on the phone) any data retrieved from unhandled exceptions.

The following syntax is used to retrieve exception data:

```
private void Application_UnhandledException(object sender,
                                           ApplicationUnhandledExceptionEventArgs e)
{
    e.Handled = true;
    ErrorPage.Exception = e.ExceptionObject;
    (RootVisual as Microsoft.Phone.Controls.PhoneApplicationFrame).Source = new
    Uri("/ErrorPage.xaml", UriKind.Relative);
}
```

Code Listings: The following code listing provides a complete coding solution to the Error Pages section.

Error Checking & Messaging	Notes
015 ErrorPage	The example directs unhandled exception data to a display page.

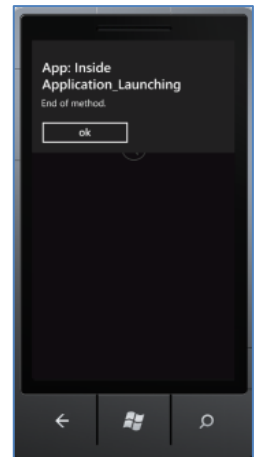


Message Boxes

A Message Box can be overlaid on a Page to force user interaction or report on a situation. A text message, title bar caption, and response buttons can be displayed.

The following syntax is used to set up a message box:

```
MessageBox.Show("Message", "Title", MessageBoxButton.OK);
```

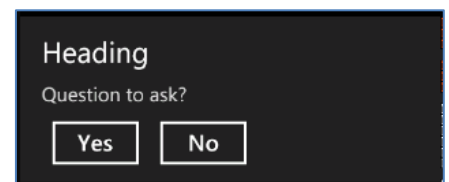


Code Listings: The following code listing provides a complete coding solution to the Message Box section.

Error Checking & Messaging	Notes
016 DebugTest	Shows the use of MessageBox and Logger Class debugging functionality.

Custom Message Boxes

Silverlight only supports the OK and OKCancel options, but there are helper files (Generic.xaml, NotificationBoxCommand.cs and NotificationBox.cs) supplied by Microsoft (free distribution and modification licence) that can be modified to produce a custom Message Box with more functionality.



Note: When a Custom Message Box is called it overrides the page displays and will stay resident until an option is clicked.

The following syntax is used to setup a custom message box:

```
using ProjectName.Controls;

// Note the () => { } format is used as a close mechanism (close / do not
// call a method)
NotificationBox.Show( "Heading",
    "Question to ask?",
    new NotificationBoxCommand("Yes", () => { NavigationService.Navigate(new
        Uri("/Page1.xaml", UriKind.Relative)); }),
    new NotificationBoxCommand("No", () => { }));
```



Code Listings: The following code listing provides a complete coding solution to the Custom Message Boxes section.

Error Checking & Messaging	Notes
017 CustomMessageBox	An example of setting up a Custom Message Box utilising the Generic.xaml, NotificationBoxCommand.cs and NotificationBox.cs utility files.

Logger Class

There is a relatively new release (7.1 OS) for Phone development that provides logging capabilities. State data and messages can be sent to a log file and the output window of the debugger.

There was something in the press recently about iPhones tracing all interaction on a phone. This is probably the Microsoft version, all interaction within an application can be logged, the restriction currently for the developer is, it is only your applications you can interrogate and process data from.

There is very little documentation on the functionality apart from where some of the methods are used in a piece of example code on circular navigation problems (source the App Hub developer forum).

When the logger is used a log file is generated (data is also directed to the output window in the debugger) that displays the call to each method in the following format:

- Start of file: Log Creation Time = the time
- "WP: App - .ctor" are text markers
- "WP: App - .ctor" the Windows Phone prefix
- "WP: App - .ctor" the name of the class file the method is in
- "WP: App - .ctor" the name of the method currently executing

```
Log Creation Time = 9:27:39 PM
**** WP: App - .ctor
**** WP: App - Application_Launching
**** WP: MainPage - .ctor
**** WP: MainPage - OnNavigatedTo
**** WP: MainPage - PhoneApplicationPage_Loaded
```

```
**** WP: MainPage - .ctor - message - 11/16/2011 11:21:56 PM [894]
```

The message can be extended by adding a message string and time stamp to the log entry:

- "WP: MainPage - .ctor - message - 11/16/2011 11:21:56 PM [894]"
message to display
- "WP: MainPage - .ctor - message - 11/16/2011 11:21:56 PM [894]"
date / time and milliseconds of event

To use the logger, the WindowsPhoneLogger.dll needs to be downloaded [create.msdn.com] and added to a project as a reference.

The following syntax is used to display, add a log entry and add an extended log entry:

```
txbLog.Text = WindowsPhoneRecipes.Logger.Instance.Log.ToString();

WindowsPhoneRecipes.Logger.Instance.AddLine();

WindowsPhoneRecipes.Logger.Instance.AddLine("- message - " + DateTime.Now.ToString() +
                                             " [" + DateTime.Now.Millisecond.ToString() + "]");
```

Code Listings: The following code listing provides a complete coding solution to the Logger Class section.

Error Checking & Messaging	Notes
016 DebugTest	Shows the use of MessageBox and Logger Class debugging functionality.

Data Storage

There are two issues here that are unique to the phone platform, run-time data storage / retrieval and external data storage / retrieval.

Run-Time Data Storage

There are several options (of increasing complexity) suggested in the literature for dealing with run-time application data processing.

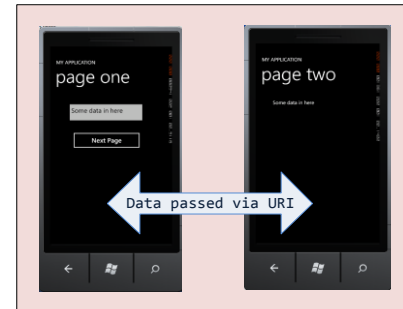
The problem in essence is saving and retrieving page data when navigating between pages.

A quick fix for passing a string between pages can be implemented by appending a message to the next page URI then retrieving the string on the next page via a query string option.

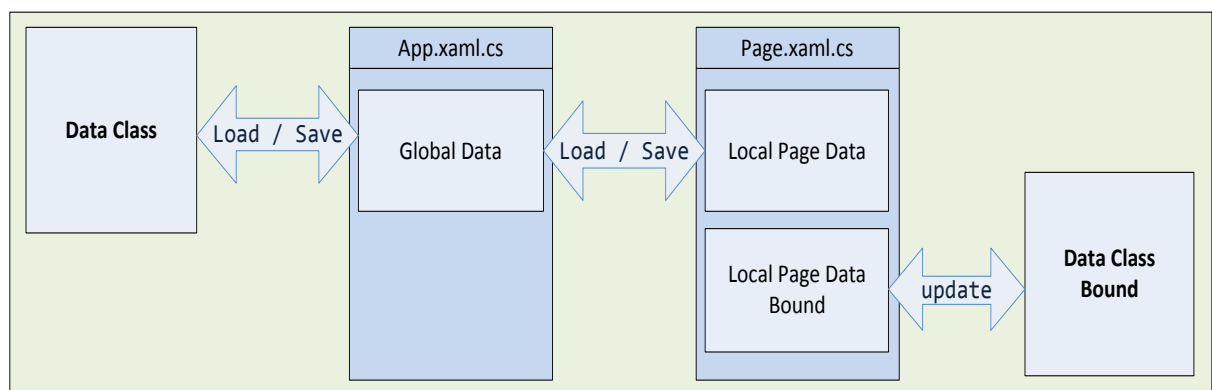
The following syntax is used to pass data between pages:

```
NavigationService.Navigate(new Uri("/PageTwo.xaml?msg=" + "Some  
message", UriKind.Relative));
```

```
NavigationContext.QueryString.TryGetValue("msg", out msg)
```



The simplest and most efficient solution though, is to design a class for the required data, declare it globally (in the App.xaml.cs file) and utilise it locally (Page.xaml.cs file).



Data binding is also supported on Silverlight objects, a UI objects property can be bound to a property of a class.

Data binding works in the normal .Net way, an event can update a field via a property and vice-a-versa.

OneTime (set once), OneWay (object updates the UI) and TwoWay (changes both ways) options are supported.

Code Listings: The following code listings provide complete coding solutions to the Run-Time Data storage section.

Data Storage	Notes
018 PassingDataBetweenPages	Example of URI and Query options used to pass a string between pages.
019 PassingDataGlobally	Example of a global class used to pass data between pages.
020 DataBinding	A simple example of data binding.

External Data Storage

There is an intrinsic problem associated with data processing in mobile devices and it stems from the fact that the device is a phone. On a Windows 7 phone a developer does not know what will happen to an application while running as it may be interrupted or closed down. Even so there is a requirement of providing a seamless user experience and continuity between the closed down and restart states of the application. In the following section I will look at the issues raised by tombstoning (application deactivation) and suggest a solution to the problem by the development of a set of methods that facilitates data storage operations.

Isolated Storage

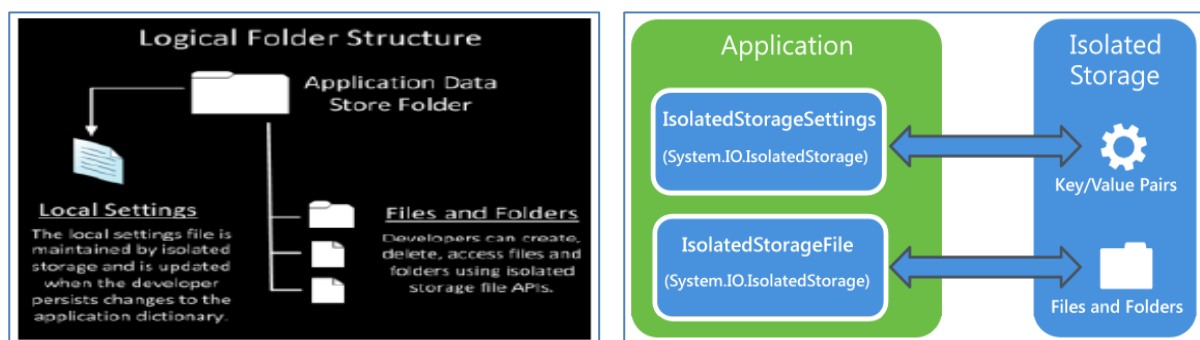
The actual process of moving data between an application and a storage area on a device is fairly straightforward, Isolated Storage is the mechanism used. In the Windows Phone environment all I/O actions are restricted to Isolated Storage. Each application has access only to its own designated storage area; the idea behind this is for improved security and reducing the chance of unauthorized access corrupting data. This is a positive feature from a developer's perspective as it stops other application interfering with stored data (no chance of overwriting data). There is no upper limit to the amount of data that can be stored per application, so data storage needs to be managed correctly (if not managed correctly it may prove difficult, not to be greedy by hogging all the available storage space on the phone).

There are several recommendations that need noting in respect to Isolated Storage:

- When updating an Application the Isolated Storage folder will not be modified, therefore all required data needs to be migrated correctly.
- Warning, when an application is uninstalled all data within the Isolated Storage store (including the folder), will be deleted by the OS.
- There is no quota for storage space per application so less is best; users will be notified by the OS when the Phone has only 10% storage space remaining.
- Clear all temporary data files in isolated storage, when no longer needed.
- When applicable always give options to allow user generated data to be deleted.

Isolated Storage data can be stored in two formats:

- Files and folders: Files and folders can be stored in a directory structure by using the `IsolatedStorageFile` class. Data can also be serialized in the normal way and processed (Note: The XML Serialization `System.Xml.Serialization` namespace does not use the `Serializable` attribute, as it only serializes public read / write properties).
- Settings Dictionary: Data can be stored as key/value pairs in a dictionary structure used for saving an applications settings by using the `IsolatedStorageSettings` class. Each entry is stored as named pairs of KEY <> Value where the value can be any object type.



[Isolated Storage: Images from MSDN <http://msdn.microsoft.com>]

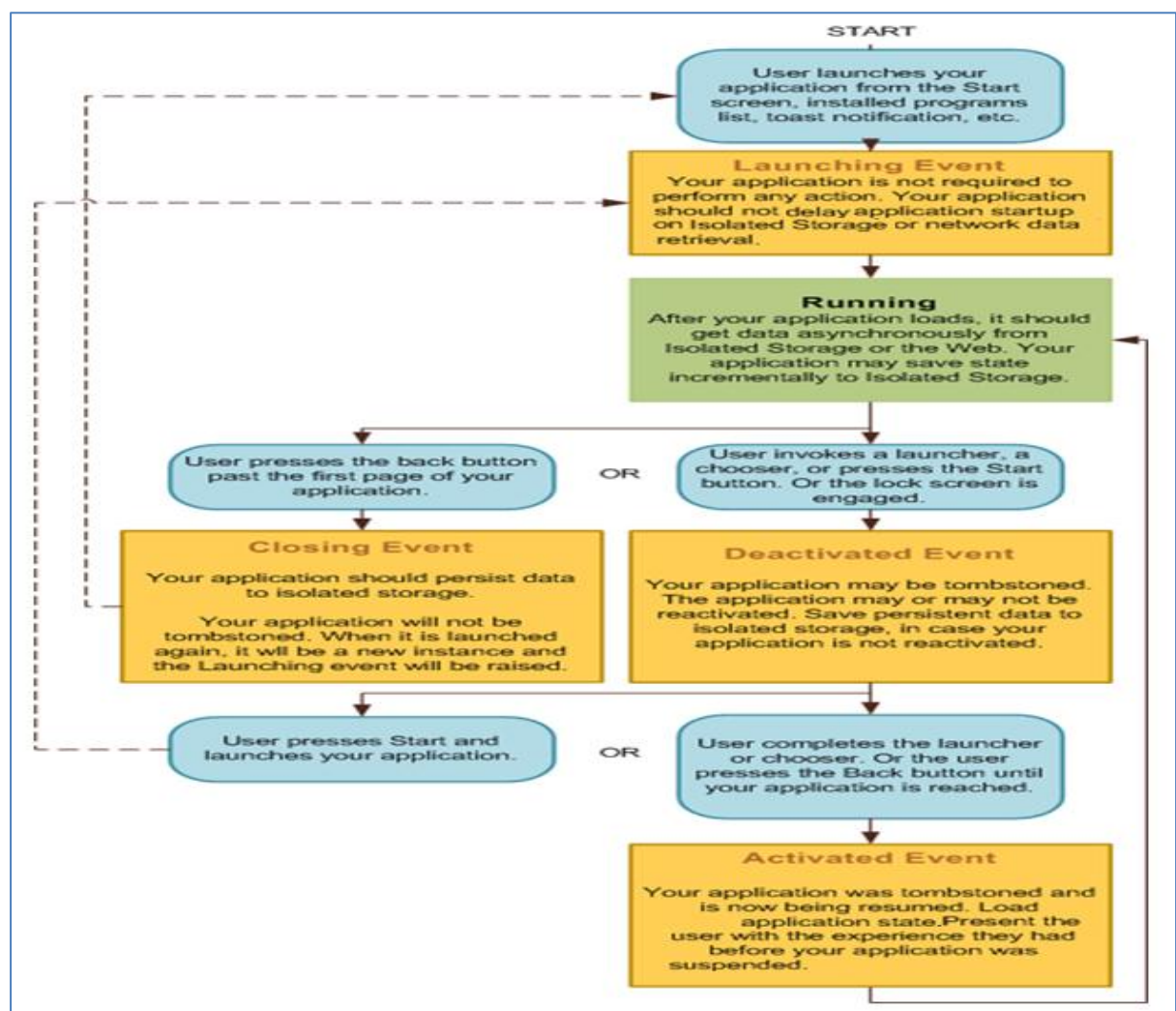
There is also third option that can be used, the `PhoneApplicationService.Current.State` class allows data to be saved as key/value pairs in a dictionary structure used for saving an applications current state. Again each entry is stored as named pairs of KEY <> Value where the value can be any object type.

The Application Life Cycle Problem

So what is the problem, as I noted previously the problem stems from the fact that the device the application will be running on is a mobile phone. Anything may happen while the application is running, as the primary purpose of the device is obviously to make phone calls. A quick look at the Application Life Cycle for mobile applications will illustrate the issues.

The Windows 7 Application Lifecycle is built around a world of interrupts, in the form of phone calls, services, reminders, text messages, application switching, low battery locks and lock screen events.

The following diagram depicts the Basic Application Lifecycle:



[Windows Phone Application Life Cycle: Updated from MSDN <http://msdn.microsoft.com>]

When an application is interrupted it may be closed down (tombstoned) or not depending on how it was interrupted. This means that the current state of an applications data may or may not need saving / updating again depending on the type of interrupt taking place. User interaction or OS actions can put an application in one of six states Launching, Running, Closing, Deactivated, Tombstoned or Activated.

Life Cycle Events

The key to controlling tombstoning problems are the life cycle events `Application_Launching`, `Application_Activated`, `Application_Deactivated` and `Application_Closing` which are located in the `App.xaml.cs` file. These events are triggered in the following ways when specific conditions are applied to the application state by the user or OS.

Application_Launching `private void Application_Launching(object sender, LaunchingEventArgs e){}`

This event executes when the application is launched from cold from the menu App List. If the application is loaded for the very first time initialization configuration data needs to be accessed here (from a configuration data file or a default data object). This event does not execute if the application is reactivated.

The following needs to be implemented in this event:

- If it is the first time the application is loaded then set-up the Isolated Storage Folder and File with default settings, otherwise load the data from Isolated Storage.
- If it is the first time the application is loaded then set-up the Settings Dictionary entries with default settings otherwise load the data from the Settings Dictionary as Key / Value pairs.

Application_Activated `private void Application_Activated(object sender, ActivatedEventArgs e){}`

This event executes when the application is activated (brought to foreground) or if the application is reactivated after the application was tombstoned. The data retrieved must reinstate the application to the same state it was in before. This event does not execute when the application is first launched.

The following needs to be implemented in this event:

- Load the Transient State Data from the Current State Dictionary as Key / Value pairs, if the key is not found then load from default file.
- As backup load the data from the Isolated Storage folder and file.
- As backup load the data from Isolated Storage Settings Dictionary as Key / Value pairs.

Application_Deactivated `private void Application_Deactivated(object, DeactivatedEventArgs){}`

This event executes when the application is deactivated (sent to background), is interrupted by an interrupt or the start key is pressed. It is recommended that data should be save here because the application may not return. This event does not execute when the application is closing.

The following needs to be implemented in this event:

- Save the Transient State Data to the Current State Dictionary as Key / Value pairs.
- As backup save the data to the Isolated Storage folder and file.
- As backup save the data to Isolated Storage Settings Dictionary as Key / Value pairs.

Application_Closing `private void Application_Closing(object sender, ClosingEventArgs e){}`

This event executes when the application is closing (Back Key hit), that is it only activates when the user hits the Back key (closes correctly) from the home page of the application. This event does not execute when the application is deactivated.

The following needs to be implemented in this event:

- Save the data to the Isolated Storage folder and file.
- Save the data to Isolated Storage Settings Dictionary as Key / Value pairs.

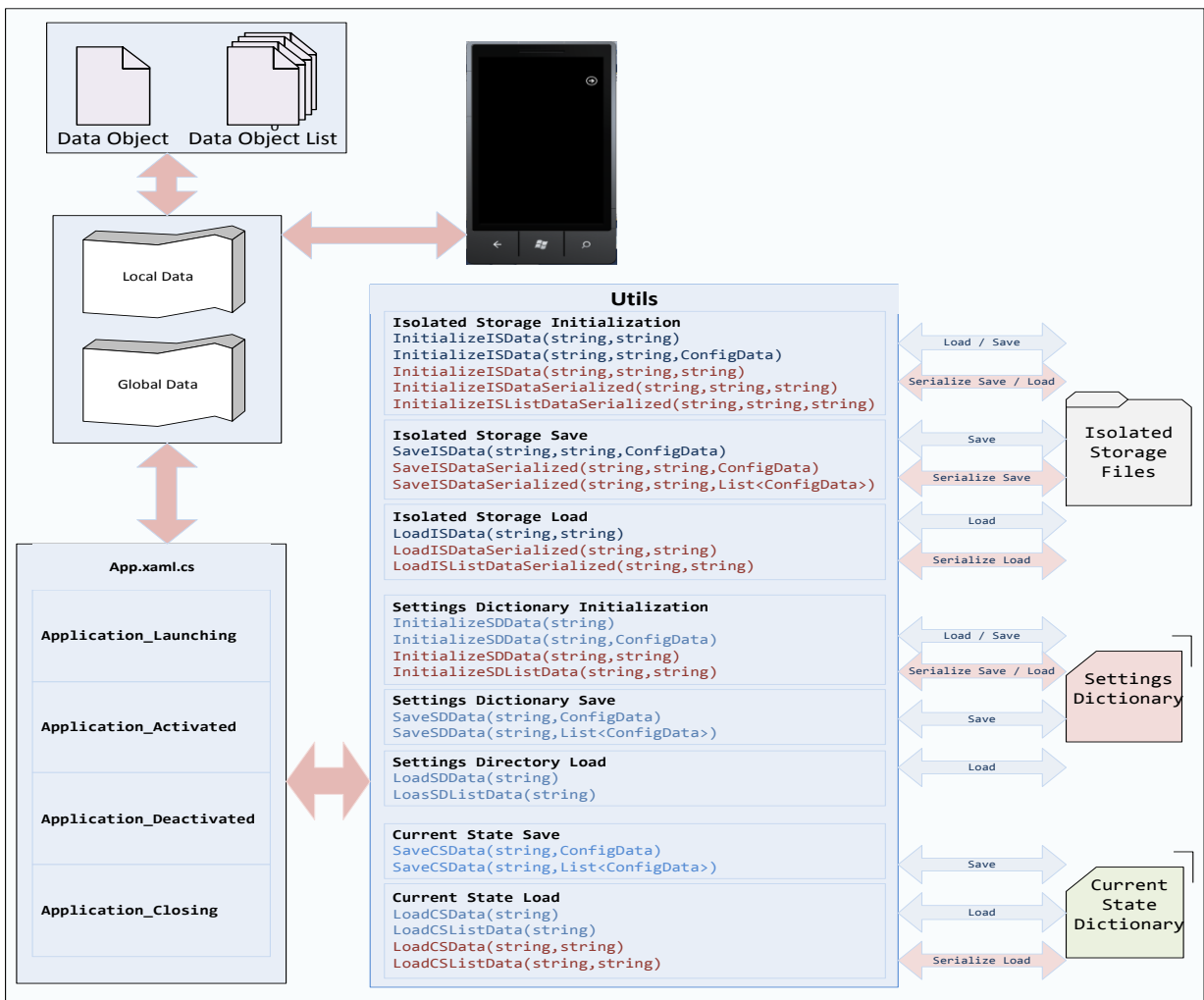
Compliance

There are also several Microsoft compliance requirements for interrupted service and data management to be taken into account. To comply with certification the following must apply:

- Provide a seamless user experience
 - Requirement to reinstate an application in the same state it was left in, this is a similar process to minimising a window on the desktop in conventional windows applications.
 - Avoid very slow read and write data operations (data can be serialized), 10 second rule for application start and close.
- Maintain long battery life.
- Only one 3rd party application can run at one time on a device.
- Storage usage must be kept to the minimum.
- Application data must be managed correctly.
- When saving and loading data there is a 10 second rule. Therefore if there is lots of data to transfer then processing must be done concurrently with application usage, not in the Life Cycle Events.

What I propose as a solution to the problems associated with data processing is to develop a set of methods that facilitate the fast saving and retrieval of data (various formats) to / from the Isolated Storage file directory, the Isolated Storage Settings Directory and the Current State Directory.

The following diagram shows how the methods can be used in the context of application development.



The following table lists the developed methods (located in the StorageUtils.cs file) for the Windows Phone Application Life Cycle Events and notes their functionality:

Initialize Isolated Storage Methods	Notes
InitializeISData()	If it is the first time the application is run then the default configuration is set via code and then the Isolated Storage file is set up line-by-line. Otherwise the data from Isolated Storage is loaded line-by-line.
InitializeISData()	If it is the first time the application is run then the default configuration is set via the input object and then the Isolated Storage file is set up line-by-line. Otherwise the data from Isolated Storage is loaded line-by-line.
InitializeISData()	If it is the first time the application is run then the default configuration is set via a serialized file (Build action: Resource) and then the Isolated Storage file is set up line-by-line. Otherwise the data from Isolated Storage is loaded line-by-line.
InitializeISDataSerialized()	If it is the first time the application is run then the default configuration is set via a serialized file (Build action: Resource) and then the Isolated Storage file is set up as serialized data. Otherwise the data from Isolated Storage is loaded as serialized data.
List<ConfigData> InitializeISListDataSerialized()	If it is the first time the application is run then the default configuration is set via a serialized file (Build action: Resource) and then the Isolated Storage file is set up as serialized data. Otherwise the data from Isolated Storage is loaded as a serialized list of data objects.

Save Isolated Storage Methods	Notes
SaveISData()	Saves the application data to Isolated Storage (virtual store) the data is saved field by field.
SaveISDataSerialized()	Saves the serialized application data to Isolated Storage (virtual store).
SaveISListDataSerialized()	Saves a list of serialized application data to Isolated Storage (virtual store).

Load Isolated Storage Methods	Notes
LoadISData()	Loads the application data from Isolated Storage (virtual store).
LoadISDataSerialized()	Loads the serialized application data from Isolated Storage (virtual store).
LoadISListDataSerialized()	Loads a list of serialized application data from Isolated Storage (virtual store).

Initialize Isolated Storage Settings Methods	Notes
InitializeSDData()	If the Isolated Storage Settings Dictionary Key is found the application data object is loaded from the Isolated Storage Settings Dictionary. Otherwise if it is the first time the application is run the defaults are set in the ConfigData object (hard coded) and the Isolated Storage Settings Dictionary object is saved.
InitializeSDData()	If the Isolated Storage Settings Dictionary Key is found the application data object is loaded from the Isolated Storage Settings Dictionary. Otherwise if it is the first time the application is run the defaults are set in the ConfigData object (via the input parameter) and the Isolated Storage Settings Dictionary object saved.
InitializeSDData()	If the Isolated Storage Settings Dictionary Key is found the application data object is loaded from the Isolated Storage Settings Dictionary. Otherwise if it is the first time the application is run the defaults are set in the ConfigData object (via a serialized file) and the Isolated Storage Settings Dictionary object saved.
List<ConfigData> InitializeSDListData()	If the Isolated Storage Settings Dictionary Key is found the application data object list is loaded from the Isolated Storage Settings Dictionary. Otherwise if it is the first time the application is run the defaults are set in the ConfigData object (via a serialized file containing a list of objects) and the Isolated Storage Settings Dictionary object list saved.

Save Settings Dictionary Methods	Notes
SaveSDData()	Saves the application data to the Isolated Storage Settings Dictionary.
SaveSDData()	Saves the application list data to the Isolated Storage Settings Dictionary.

Load Settings Directory Methods	Notes
LoadSDData(string)	Loads the application data from the Isolated Storage Settings Dictionary.
LoasSDListData(string)	Loads the application data List from the Isolated Storage Settings Dictionary.

Save Current State Methods	Notes
SaveCSData()	Saves the work in progress (transit) application data to the Current State Dictionary.
SaveCSData()	Saves the work in progress (transit) application list data to the Current State Dictionary.

Load Current State Methods	Notes
LoadCSData()	Loads the application data from the Isolated Storage Current State Dictionary.
LoadCSData()	If the Key is found, loads the application data from the Isolated Storage Current State Dictionary. Otherwise the defaults are loaded from file.
LoadCSListData()	Loads the application list data from the Isolated Storage Current State Dictionary.
LoadCSListData()	If the Key is found, loads the application list data from the Isolated Storage Current State. Otherwise the defaults are loaded from file.

Code Listings: The following code listings provide complete coding solutions to the Life Cycle Events section.

Data Storage	Notes
021 ApplLifecycle debug test	A debugging test bed for Application Lifecycle event firing order.
022 Utils Template	Code stub for the data processing methods in the StorageUtils.cs class.

Testing

The methods were tested for speed using different data ranges and installation / run states. Three phones were used and several test-runs undertaken. The results were consistent across all phones with very little deviation (none or +/- one millisecond). The sample data in [Appendix A](#) represents the mean values for the tests.

Each test was carried out 3 times (using 10 / 40 / 208 / 832 / 1816 / 18016 / 18160 / 1801600 byte data blocks):

Test 1: The first time the application was loaded onto the device.

Test 2: The application was re-opened from the Menu bar.

Test 3: The phone was switched off and on and then the application was re-opened from the Menu bar a second time.

Review of Results

All the methods were proven to comply with the recommended requirements previously set out for speed of data transfer and user experience. The methods are accessible by including the StorageUtils.cs file into a Windows 7.1 Phone project build (See [Appendix A: Isolated Storage Testing](#) for a full break down of the testing results).

Code Listings: The following code listings provide access to the Speed Tests used in this Testing section.



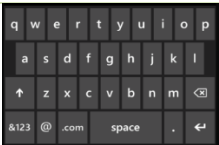


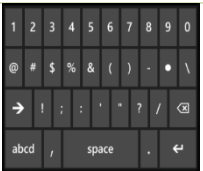

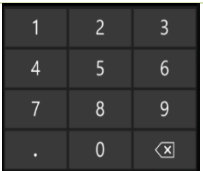
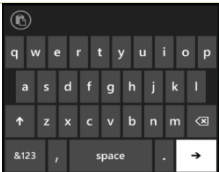
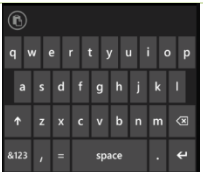
Data Storage	Notes
023 Speed Test 1	Speed tests for data blocks 80 bits and 320 bits.
024 Speed Test 2	Speed tests for data blocks 1,664 bits and 6,656 bits.
025 Speed Test 3	Speed tests for data blocks 14,528 bits and 145,280 bits.
026 Speed Test 4	Speed tests for data blocks 144,128 bits and 14,412,800 bits.

User Interaction with the Phone

User interaction via a mobile phone has several unique possibilities for the development of Pervasive Games. All input routes utilise relatively new technologies, from basic input via a touch screen (keyboard and touch input) to the use of built in devices / sensors (accelerometer, compass, gyroscope and microphone) and OS / Cloud services (launchers and choosers).

Keyboard

The keyboard is accessed by touching an input object (textbox) on the UI. There are 10 keyboard configurations available as the following table shows.

Chat and Text. Includes auto-complete and an emoticon menu with two pages of emoticons.		The default keyboard.	
EmailNameOrAddress, EmailSmtAddress, EmailUserName. Includes the @ and .com keys. Press and hold the .com key to display additional options (.org .com .edu .net).		Url. Includes the .com key. Press and hold the .com key to display additional options (.com .org .edu .net). Press and hold the period key to display additional options (- + & " : . /).	
NameOrPhoneNumber Press the 123 key to change to the telephone number keyboard. Press and hold the period key to display additional options (- _ , .).		AddressStreet*, Time, PostalCode, CurrencyAmountAndSymbol, CurrencyChinese and, PostalAddress. Contains numbers and symbols.	
TelephoneAreaCode, TelephoneLocalNumber and TelephoneNumber. Mimics the telephone keypad. Press and hold the period key to display additional options (, () X .). Press and hold the 0 key to enter +.		CurrencyAmount, DateDay, DateMonth, DateYear, Digits, Number, NumberFullWidth, NumericPassword, TimeHour and TimeMinorSec. Contains numbers and a decimal point. Press and hold the period key to display additional options (, . -).	
Maps and Search. Includes auto-complete.		Formula. Includes auto-complete. Press the &123 key to change to the number and symbol keyboard, optimized for entering numeric formulas.	

[Keyboard Configuration: Data from MSDN <http://msdn.microsoft.com>]

To implement a keyboard the following syntax is used, first XAML and then XAML / C#:

```
<TextBox Name="txtPhoneNumber" Height="71" Width="460" > <TextBox.InputScope>
    <InputScope> <InputScopeName NameValue="Number" /> </InputScope>
</TextBox.InputScope> </TextBox>
<TextBox Name="txtPhoneNumber" Height="71" Width="460" />
InputScope scope = new InputScope();
InputScopeName name = new InputScopeName();
name.NameValue = InputScopeNameValue.Number;
scope.Names.Add(name);
txtPhoneNumber.InputScope = scope;
```

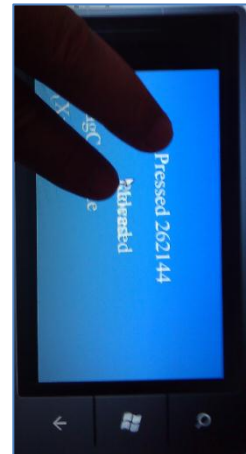
Code Listings: The following code listing provides a complete coding solution to the Keyboard section.

User Interaction	Notes
027 Keyboards	Displays the basic Keyboard options to screen.

Touch Input

The primary mode of UI interaction on a mobile phone is via the touch screen. There are two options for developers to tap into, the Silverlight frameworks and XNA framework.

Standard screen functionality allows for up to four input sources (touch events), each of which can be uniquely identified over its runtime.



Silverlight Touch Input

Silverlight offers a basic set of gesture responses, Tap, DoubleTap and Hold, which are accessible via the following events:

- ManipulationStarted: Triggered when finger is placed on the screen.
- ManipulationDelta: Fires repeatedly when a finger is moving about the screen.
- ManipulationCompleted: Triggered when a finger is taken off the screen.

The following syntax is used to capture basic touch events in Silverlight:

```
this.ManipulationDelta += this.PhoneApplicationPage_ManipulationDelta;  
void PhoneApplicationPage_ManipulationDelta(object sender, ManipulationDeltaEventArgs e) {}
```

Code Listings: The following code listing provides a complete coding solution to this section.

User Interaction	Notes
028 Touch	Touch example, reports touch positions via the ManipulationDelta event.

XNA Touch Input

A more flexible approach is via the XNA framework (I will look at the actual XNA framework in more detail later) which has a built in gesture system. The `TouchPanel` class offers access to multi-touch features, such as device capability, touch state of the touch panel and gesture type recognition. Accessing the touch panel's capabilities is done via the `TouchPanel.GetCapabilities` method. The method returns the touch panel state in a structure (`TouchPanelCapabilities`) which can be queried to ascertain if a touch panel is connected and if so how many touch points it supports.

The following syntax is used to check if a touch panel is connected and retrieve the maximum number of supported touch points on the touch panel:

```
TouchPanelCapabilities TouchCapabilities = TouchPanel.GetCapabilities();  
if (TouchCapabilities.IsConnected) int temp = TouchCapabilities.MaximumTouchCount;
```

The `TouchPanel.GetState` method returns the state of the touch panel in a `TouchCollection` structure. The data in this structure describes each `TouchLocation` in the current `TouchCollection` as location and state data for each of the four touch points. It is possible to query the touch point location, pressed, released and moved states.

The following syntax is used to read the state of the touch panel:

```
TouchCollection touchState = TouchPanel.GetState();  
foreach(TouchLocation touch in touchState)  
{  
    Point touchPoint = new Point((int)touch.Position.X,(int)touch.Position.Y);  
    if (touch.State == TouchLocationState.Pressed) { }  
    if (touch.State == TouchLocationState.Released) { }  
    if (touch.State == TouchLocationState.Moved) { }  
}
```

It is possible by enabling each gesture type required using the `TouchPanel.EnabledGestures` method. Gestures can then be queried in a gesture queue using the `TouchPanel.IsGestureAvailable` flag to determine if there are any more gestures in the queue to process. False represents no more gestures in the queue to process, if there are gestures to retrieve then the `TouchPanel.ReadGesture` method is used to retrieve the data.

The following syntax is used to process a gesture queue:

```
TouchPanel.EnabledGestures = GestureType.DoubleTap | GestureType.DragComplete |
GestureType.Flick | GestureType.FreeDrag | GestureType.Hold |
GestureType.HorizontalDrag | GestureType.None | GestureType.Pinch |
GestureType.PinchComplete | GestureType.Tap | GestureType.VerticalDrag;

while (TouchPanel.IsGestureAvailable)
{
    GestureSample gs = TouchPanel.ReadGesture();
    textOutput4 = gs.GestureType.ToString();
    textOutput5 = gs.Position.ToString();

    switch (gs.GestureType)
    {
        case GestureType.DoubleTap: // some code; break;
        case GestureType.DragComplete: // some code; break;
        case GestureType.Flick: // some code; break;
        case GestureType.FreeDrag: // some code; break;
        case GestureType.Hold: // some code; break;
        case GestureType.HorizontalDrag // some code; break;
        case GestureType.None: // some code; break;
        case GestureType.Pinch: // some code; break;
        case GestureType.PinchComplete: // some code; break;
        case GestureType.Tap: // some code; break;
        case GestureType.VerticalDrag: // some code; break;
    }
}
```

The following table notes all the touch gestures supported by the XNA framework:

Type of Gesture	Notes
Tap	Touch and release the screen.
DoubleTap	Two taps on the screen in succession.
Hold	Touch the screen and hold in place for a short time.
FreeDrag	Touch the screen and move (slide) in any direction.
VerticalDrag	Touch the screen and move (slide) up or down.
HorizontalDrag	Touch the screen and move (slide) left or right.
DragComplete	Notes the end of a FreeDrag, VerticalDrag, or HorizontalDrag gesture type.
Flick	Drag across the screen and lift up in a flick motion without stopping.
Pinch	Press on the screen with two fingers and make an open or close pinch gesture.
PinchComplete	Notes the end of a Pinch gesture.
None	Represents no gesture.

[XNA Touch Gestures: Data from MSDN <http://msdn.microsoft.com>]

Code Listings: The following code listings provide complete coding solutions to this section.

User Interaction	Notes
029 Touch Rectangles	Basic XNA touch, reports pressed, moved, released and position types.
030 Touch Enumeration	XNA touch input, reports all gesture and position states.

The Guide

Touch input can also be used to active Guide functionality, allowing text input via a keyboard option. The Guide class belongs to the `Microsoft.Xna.Framework.GamerServices` namespace and allows access to the Guide Text Entry System.

I have included this for my reference, as it could be useful in some gaming contexts where it is difficult (or not possible) to use a Silverlight object for text input.

The following syntax is used to set up a keyboard and an AsyncCallback method to process input text using the Guide:

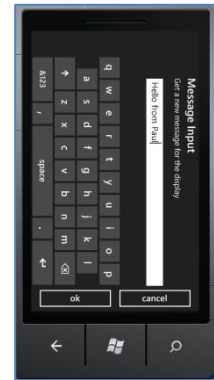
```
object stateObject;

void keyboardDone(IAsyncResult result)
{
    string guideString = Guide.EndShowKeyboardInput(result);
    if (guideString != null) message = guideString;
}

protected override void Update(GameTime gameTime)
{
    TouchCollection touches = TouchPanel.GetState();
    if (touches.Count > 0)
    {
        if (!Guide.IsVisible) Guide.BeginShowKeyboardInput( PlayerIndex.One, title, text, message,
                                                            new AsyncCallback(keyboardDone), stateObject);
    }
}
```

Code Listings: The following code listing provides a complete coding solution to the Guide section.

User Interaction	Notes
031 GuideInput	Activates the keyboard via touch input using the Guide mechanism.



Sound Input

Voice recognition is publicized as a major feature on the Windows 7 OS and it is supported on both desktops and phones. The problem is that currently the voice recognition API's are not accessible for 3rd party application development on Microsoft Phone devices. This means there is no documentation, no libs and the IDE has no support for including other voice recognition packages (.Net / Cloud / Web services) into the Windows 7 Phone build.



Searching through the discussion forums and developer sites [[Microsoft TellMe developer forums: http://www.microsoft.com/en-us/Tellme/developers/default.aspx](http://www.microsoft.com/en-us/Tellme/developers/default.aspx)][[App Hub Community forums: http://forums.create.msdn.com/forums/](http://forums.create.msdn.com/forums/)][[MSDN forum: http://create.msdn.com/en-US/](http://create.msdn.com/en-US/)] flags this topic as a real bone of contention with developers and corporates. What makes this decision from Microsoft controversial is it allows Microsoft applications and OS operations access to the Bing audio speech search options (texting, phoning, searching, location finding etc.) but nothing else. The potential here is to make 3rd party applications look amateurish (less polished than Microsoft applications), a subtle difference but commercially very important from both the financial and branding (of product range or company) standpoints. This is also slightly disappointing for the prospective of this project as voice recognition / activation would have made an interesting addition to the pervasive gaming experience. There is light at the end of the tunnel though, according to the team working on the voice activation API's [[App Hub discussion forums: http://forums.create.msdn.com/](http://forums.create.msdn.com/)] Windows 8 may have limited support for mobile voice activation API's. None-the-less I will investigate what is currently available to developers, which is processing audio data via the built in microphone.

Microphone

The built in Microphone can be accessed in both Silverlight and XNA applications via the `Microsoft.Xna.Framework.Audio.Microphone` class.

The process is to enable the microphone, collect one second of audio data at a time into a byte buffer then save the audio string in a `MemoryStream`.

Playback is then possible using the data in the `MemoryStream`.

There are several classes required in the build, `System.IO`, `System.Windows.Threading`, `Microsoft.Xna`, `Microsoft.Xna.Framework` and `Microsoft.Xna.Framework.Audio`.



The following syntax is used to set up and retrieve data from the Microphone:

- Set Global variables:

```
Microphone microphone = Microphone.Default;  
byte[] buffer;  
MemoryStream stream = new MemoryStream();  
SoundEffect sound;
```
- Set up the microphone event handler:

```
microphone.BufferReady += new EventHandler<EventArgs>(microphone_BufferReady);
```
- Copy data from the microphone to a buffer and then to a stream:

```
void microphone_BufferReady(object sender, EventArgs e)  
{  
    microphone.GetData(buffer);  
    stream.Write(buffer, 0, buffer.Length);  
}
```
- Start the recording process:

```
microphone.BufferDuration = TimeSpan.FromMilliseconds(1000);  
buffer = new byte[microphone.GetSampleSizeInBytes(microphone.BufferDuration)];  
microphone.Start();
```
- Stop the recording process:

```
if (microphone.State == MicrophoneState.Started)  
{  
    microphone.Stop();  
}
```
- Play the recorded data:

```
sound = new SoundEffect(stream.ToArray(), microphone.SampleRate, AudioChannels.Mono);  
sound.Play();
```

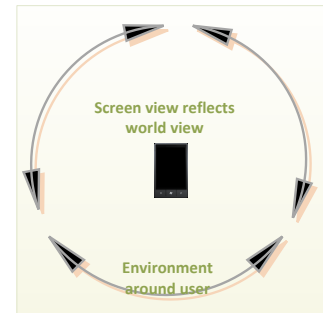
Code Listings: The following code listing provides a complete coding solution to the Microphone section.

User Interaction	Notes
032 MicrophoneTest	Simple Microphone application that records and plays microphone input.

Camera

The built in camera's still and video data capture features can be used to add content to pervasive games. Capturing still or video images via the camera is supported in both Silverlight and XNA applications.

With a bit of tweaking of the camera API's it is also possible to redirect the camera data to a screen buffer. This allows the phone to display the external environment as a backdrop to an applications interface display, opening the door to some interesting pervasive application development (I will look at this area in some detailed later when I discuss combining several phone features into an application).



Retrieving Images for a Background Display

The process required to access the camera is fairly straightforward, create a canvas to display the data, initialise the camera device, setup any camera events required, then switch the camera on, process the events and finally switch the camera off.

There are several classes required in the build to implement the full camera options depending on what features are required, `System.ComponentModel`, `System.Threading`, `System.IO`, `System.IO.IsolatedStorage`, `Microsoft.Phone.Shell`, `System.Windows.Navigation` and `Microsoft.Xna.Framework.Media`.

The base class required to implement standard camera functionality is `Microsoft.Devices`.

The following syntax is used to set up and retrieve data from the camera:

- Camera viewfinder set in XAML as a canvas
`<Canvas.Background> <VideoBrush x:Name="viewfinderBrush" /> </Canvas.Background>`
- Variable declarations
`PhotoCamera cam;`
- Check if the device is available, set up the camera events and set the video source

```
if ((PhotoCamera.IsCameraTypeSupported(CameraType.Primary) == true) ||  
    (PhotoCamera.IsCameraTypeSupported(CameraType.FrontFacing) == true))  
{  
    cam = new Microsoft.Devices.PhotoCamera(CameraType.Primary);  
    cam.Initialized += new  
        EventHandler<Microsoft.Devices.CameraOperationCompletedEventArgs>(cam_Initialized);  
    // Set up the following events as required  
    cam.CaptureCompleted += new  
        EventHandler<CameraOperationCompletedEventArgs>(cam_CaptureCompleted);  
    cam.CaptureImageAvailable += new  
        EventHandler<Microsoft.Devices.ContentReadyEventArgs>(cam_CaptureImageAvailable);  
    cam.CaptureThumbnailAvailable += new  
        EventHandler<ContentReadyEventArgs>(cam_CaptureThumbnailAvailable);  
    viewfinderBrush.SetSource(cam);  
}
```
- Updates the UI if the initialization succeeded

```
void cam_Initialized(object sender, Microsoft.Devices.CameraOperationCompletedEventArgs e)  
{  
    if(e.Succeeded) this.Dispatcher.BeginInvoke(delegate(){txtDebug.Text="Camera initialized";});  
}
```

- Change the output relative to the orientation

```
protected override void OnOrientationChanged(OrientationChangedEventArgs e)
{
    if (cam != null)
    {
        int landscapeRightRotation = 180;
        if (cam.CameraType == CameraType.FrontFacing) landscapeRightRotation = -180;
        if (e.Orientation == PageOrientation.LandscapeRight)
        {
            viewfinderBrush.RelativeTransform = new CompositeTransform()
            { CenterX = 0.5, CenterY = 0.5, Rotation = landscapeRightRotation };
        }
        else
        {
            viewfinderBrush.RelativeTransform = new CompositeTransform()
            { CenterX = 0.5, CenterY = 0.5, Rotation = 0 };
        }
    }
    base.OnOrientationChanged(e);
}
```
- Switch off the camera and close down the events

```
if (cam != null)
{
    cam.Dispose();
    cam.Initialized -= cam_Initialized;
    cam.CaptureCompleted -= cam_CaptureCompleted;
    cam.CaptureImageAvailable -= cam_CaptureImageAvailable;
    cam.CaptureThumbnailAvailable -= cam_CaptureThumbnailAvailable;
}
```

Code Listings: The following code listings provide complete coding solutions to the Camera section.

User Interaction	Notes
033 CameraGrab	Example sets up the camera data as a background (Silverlight version).
034 CameraGrabXNA	Example sets up the camera data as a background (XNA version).
035 BasicCamera	Basic camera application (an updated MS example).
036 BasicVideo	Basic camera video application (an updated MS example).

Vibration

As a feedback mechanism, vibration has been used very successfully on game consoles, particularly to let users know something different is happening in (or is about to happen in) a game. The Phone has a vibrated feature accessible via the `VibrateController` class and although it is limited it can be used to good effect. Vibrations can be set from zero to 5 seconds (only) using the `Start()` method.



The following syntax is used to set up a one second vibration on a device:

```
using Microsoft.Devices;
VibrateController vibrate = VibrateController.Default;
vibrate.Start(TimeSpan.FromMilliseconds(1000));
```

Code Listings: The following code listing provides a complete coding solution to the Vibrate section.

User Interaction	Notes
037 Vibrate	Example of vibrating the phone for one second.

Sensors

This is an interesting area for pervasive game development as smart phones offer access to several sensors which provide feedback on the orientation and motion of the device.

This allows the devices physical positioning in the real world to be used as an input mechanism for pervasive or virtual reality applications.

Data can be processed from the Accelerometer, Compass and Gyroscope individually, or combined together and accessed via the Combined Motion API's (allowing combinations of sensor data to be set up).

There are restrictions on using the built in sensors and the following should be taken into consideration at the design stage of developing an application:

- Not all sensors may be supported by the device, therefore the required sensor must be tested for and an alternative input source provided if not there (required to comply with registration).
- If a device is not present a warning must be output and an agreement by the user is also required that they understand they do not have the required device (required to comply with registration).
- Sensors are not accessible when the screen lock is in place.
- Sensors are not accessible if the application is running as a background agent (application code that runs in the background even when the application is not the active application).

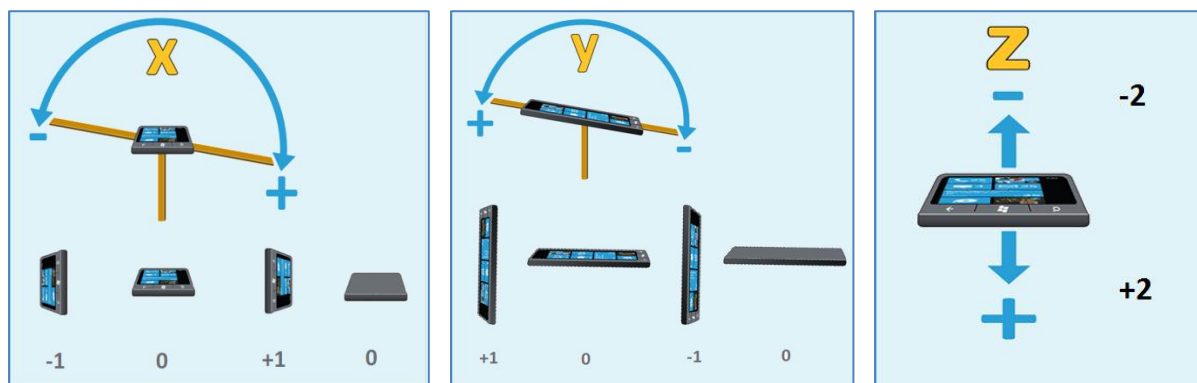
The base class for sensor readings is the `Microsoft.Devices.Sensors.SensorBase<TSensorReading>` class.

Accelerometer

The accelerometer measures the movement of the device as a gravitational acceleration force on the X, Y and Z axis.

The following table shows the gravitational values returned by the accelerometer.

X axis	Y axis	Z axis
Tilted left on its edge -1.	Tilted up standing on its edge -1.	With no movement face up -1
Centre face up 0	Centre face up 0	With no movement face down -1
Tilted right on its edge +1.	Tilted back standing on its edge +1.	With no movement any edge 0
Centre face down 0	Centre face down 0	



When designing the UI, allowing the user to actually see the screen is an important consideration, the following table shows useable ranges on the X, Y and Z axis.

X axis	Y axis	Z axis
-0.3 to +0.3 Can see the screen clearly	-0.15 to -0.55 Can see the screen clearly	Fast movement on the device can reach from -2 to +2.
-0.5 to -0.3 and +0.3 to +0.5 Can just see the screen	0.0 to -0.15 and -0.55 to -0.85 Can just see the screen	
-1.0 to -0.5 and +0.5 to +1.0 Cannot see the screen	+0.3 to 0.0 and -0.85 to -0.95 Cannot see the screen	

All Windows 7 phones will have an accelerometer as standard so this is the primary sensor for gathering rotational data. Note that the API's changed considerably with no backward compatibility (why) from the 7.0 to 7.1 OS.

There are a couple of classes required in the build to access the accelerometer and they also need adding to the reference list, `Microsoft.Devices.Sensors` and `Microsoft.Xna.Framework`.

The following syntax is used to set up and retrieve data from the accelerometer:

- Access the accelerometer
`Accelerometer accelerometer;`
- Check to see if the device on which the application is running supports the accelerometer sensor
`if (!Accelerometer.IsSupported) // Accelerometer not supported code`
- Instantiate the accelerometer sensor
`if (accelerometer == null)`

```

{
    accelerometer = new Accelerometer();
    accelerometer.TimeBetweenUpdates = TimeSpan.FromMilliseconds(20);
    accelerometer.CurrentValueChanged += new EventHandler
    <SensorReadingEventArgs<AccelerometerReading>>(accelerometer_CurrentValChanged);
    try
    {
        accelerometer.Start();
    }
    catch (InvalidOperationException ex)
    {
        // error starting accelerometer
    }
}
else
{
    try
    {
        accelerometer.Stop();
        accelerometer = null;
    }
    catch (AccelerometerFailedException ex)
    {
        // error stopping accelerometer;
    }
}

```

- Implement the CurrentValueChanged event handler.

```
void accelerometer_CurrentValChanged(object
sender, SensorReadingEventArgs<AccelerometerReading> e)
{
    Dispatcher.BeginInvoke(() => UpdateUI(e.SensorReading));
}
```
- Implement the UpdateUI method that will show the accelerometer data to the user.

```
private void UpdateUI(AccelerometerReading accelerometerReading)
{
    Vector3 acceleration = accelerometerReading.Acceleration;
    // acceleration.X.ToString()
    // acceleration.Y.ToString()
    // acceleration.Z.ToString()
}
```

Code Listings: The following code listings provide complete coding solutions to the Accelerometer section.

User Interaction	Notes
038 AccelerometerTest	Basic test of the Accelerometer, returns X, Y and Z data.
039 AccelerometerApp	Simple level and seismograph example based on Accelerometer data.
040 AccelerometerXNA	Accelerometer XNA sprite movement test, five movement types are tested, changeable via touching the screen.
041 MapMotion	Quick test to link Accelerometer motion on map movement (unfinished but tests the Accelerometer).
042 SensorTest	Returns data from the Accelerometer, Compass and Gyroscope sensors.

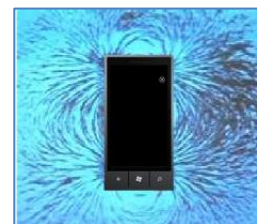
Compass

The compass sensor can be used to gather data on the devices relative rotational position to the Earth's magnetic (and true) North Pole.

Raw magnetometer readings can also be accessed that define the magnetic forces surrounding the device on the X, Y, and Z axis.

The magnetometer is fairly sensitive and has the potential for some interesting uses.

There is an interesting piece of research being conducted by Deutsche Telekom [<http://www.telekom.com>] on 3D interaction (MagiTact). The research is focused on the use of magnets and magnetometers to track changes in magnetic fields with the intention of identifying hand gestures.



The magnetometer built into the HTC phone I have been testing on seems sensitive enough to recognise a hand passing over the phones surface. Thinking in a pervasive model, it would be possible to use value changes on each axis as a trigger to add (or remove) game elements, say searching a room for items (ghosts / butterflies / treasure) and adding elements relative to magnetic fluctuations in the room.

There are a couple of classes required in the build to access the compass and they also need adding to the reference list, `Microsoft.Devices.Sensors`, `Microsoft.Xna.Framework` and `System.Windows.Threading`.

The following syntax is used to set up and retrieve data from the compass and magnetometer:

- Access the compass
Compass compass;
- Check to see if the device on which the application is running supports the compass sensor.
if (!Compass.IsSupported) // Not supported code
- If the Compass object is null, then create a new instance.
if (compass == null)
{
 compass = new Compass();
 compass.TimeBetweenUpdates = TimeSpan.FromMilliseconds(20);
 compass.CurrentValueChanged += new
 EventHandler<SensorReadingEventArgs<CompassReading>>(compass_CurrentValueChanged);
}

try
{
 compass.Start();
}
catch (InvalidOperationException)
{
 // Unable to start compass
}

• Catch new compass data at the frequency specified with TimeBetweenUpdates.
void compass_CurrentValueChanged(object sender, SensorReadingEventArgs<CompassReading> e)
{
 Dispatcher.BeginInvoke(() => UpdateUI(e.SensorReading));
}

• Process the compass data.
private void UpdateUI(CompassReading reading)
{
 // reading.MagneticHeading.ToString("0.00");
 // reading.TrueHeading.ToString("0.00");

 // reading.MagnetometerReading.X.ToString("0.00");
 // reading.MagnetometerReading.Y.ToString("0.00");
 // reading.MagnetometerReading.Z.ToString("0.00");
}

• Stop the compass.
if (compass != null) compass.Stop();

• Calibrate the compass, the system will determine when calibration is required and take action
all that is required is to recognise that the process is happening.
compass.Calibrate += new EventHandler<CalibrationEventArgs>(compass_Calibrate);

void compass_Calibrate(object sender, CalibrationEventArgs e)
{
 Dispatcher.BeginInvoke(() => { calibrationStackPanel.Visibility = Visibility.Visible; });
 calibrating = true;
}

Code Listings: The following code listing provides a complete coding solution to the Compass section.

User Interaction	Notes
042 SensorTest	Returns data from the Accelerometer, Compass and Gyroscope sensors.

Gyroscope

Some of the higher end smart phones are equipped with a gyroscope sensor, but this is not a standard feature on all devices, so the sensor needs to be checked for if it is a required feature in an application.

The gyroscope sensor returns data on the rotational velocity of the device around the centre points of the X, Y and Z axis.



The value of rotation velocity is returned in radians per second, where no movement on an axes returns 0, the value increases as the rotation gets faster ($2 * \pi$ radians is a full rotation).



There are a couple of classes required in the build to access the gyroscope and they also need adding to the reference list, `Microsoft.Devices.Sensors` and `Microsoft.Xna.Framework`.

The following syntax is used to set up and retrieve data from the gyroscope:

- Access the gyroscope
`Gyroscope gyroscope;`
- Check to see if the device on which the application is running supports the gyroscope sensor.
`if (!Gyroscope.IsSupported) // Sensor not supported code`
- If the Gyroscope object is null, then create a new instance.

```
if (gyroscope == null)
{
    gyroscope = new Gyroscope();
    gyroscope.TimeBetweenUpdates = TimeSpan.FromMilliseconds(20);
    gyroscope.CurrentValueChanged += new
        EventHandler<SensorReadingEventArgs<GyroscopeReading>>(gyroscope_CurrentValueChanged);
}
try
{
    gyroscope.Start();
}
catch (InvalidOperationException)
{
    // Unable to start gyroscope
}
```
- Catch new gyroscope data at the frequency specified with `TimeBetweenUpdates`.

```
void gyroscope_CurrentValueChanged(object sender, SensorReadingEventArgs<GyroscopeReading> e)
{
    Dispatcher.BeginInvoke(() => UpdateUI(e.SensorReading));
}
```

- Process the gyroscope data.

```
private void UpdateUI(GyroscopeReading gyroscopeReading)
{
    Vector3 rotationRate = gyroscopeReading.RotationRate;
    // rotationRate.X.ToString("0.00")
    // rotationRate.Y.ToString("0.00")
    // rotationRate.Z.ToString("0.00")
}
```
- Stop data acquisition from the gyroscope.

```
if (gyroscope != null) gyroscope.Stop();
```

Code Listings: The following code listing provides a complete coding solution to the Gyroscope section.

User Interaction	Notes
042 SensorTest	Returns data from the Accelerometer, Compass and Gyroscope sensors.

Problems with Sensors

There are several inherent problems with the current sensor setup on mobile devices.

The accelerometer is subject to gravitational error, the gyroscope tends to drift and the compass only gives an approximate positional reference. Calculations based on real-time sensor data can also cause errors.

One major issue that is still waiting resolution from Microsoft is the True North / Southern Hemisphere data bug. Data returned for southern hemisphere calculations is capped at a value of 225, which has the effect of clipping off any degree readings over this value. The magnetic and true north headings returned from the API are basically wrong when calibrated in the southern hemisphere.

The consensus on the developer forums seems to be that there is a casting issue in the API calculations that affects the return data. The bug is in the calculation process that involves processing the magnetometer reading data in micro teslas (float), the conversion calculation (16 bit integer) and the returned degrees (byte max 255).

The issues have been raised with Microsoft via Nokia engineers but currently there is no fix. Seems to be like the rugby, all the focus (money) is on the northern hemisphere markets. The only current workaround for southern hemisphere development is to use the Location Services and Bing Maps API's to return location positional data. [Thanks to: <http://answers.microsoft.com/en-us/winphone/forum/wp7-wpfeatures/faulty-compass-on-windows-phones-will-there-be-a/d3b63a7e-7fc2-427f-b2d2-9e5c24fae040>: JChurcher's Forum Post: This one caused me to question my code until I discovered others had a similar problem on the NoKia devices].

In response to some of these issues associated with inaccurate sensor data Microsoft have revised sensor operations and released a new Motion API (7.1 onwards).

Motion

The Motion class is a very useful feature in the new mango (7.1) release that allows the orientation and motion of the device to be calculated more accurately than previously.

The Motion class offers an interface that sits on top of some complex low-level sensor operations (calculations) that return the attitude (yaw, pitch, and roll) and rotational / linear acceleration of the device based on the effects of gravity and the movement of the phone by the user.

This is currently the definitive API for pervasive game applications as two different levels of accuracy are offered automatically based on the sensor configuration of the device.

When the motion API is used in an application, a check is automatically made for any supported sensors on the device, Normal Mode activates if the accelerometer and compass are present, then data can be processed from these devices to calculate relative positions. Enhanced Mode activates if all 3 sensors are present (accelerometer, compass and gyroscope) giving a greater accuracy by using data collectively from the accelerometer, compass and gyroscope.

There are a couple of classes required in the build to access the motion API's and they also need adding to the reference list, `Microsoft.Devices.Sensors` and `Microsoft.Xna.Framework`.

The following syntax is used to set up and retrieve data from the motion API's:

- Access the motion operations
`Motion motion;`
- Check to see if the Motion API is supported on the device.

```
if (!Motion.IsSupported)
{
    return;
}
```
- If the Motion object is null, initialize it and add a CurrentValueChanged event

```
if (motion == null)
{
    motion = new Motion();
    motion.TimeBetweenUpdates = TimeSpan.FromMilliseconds(20);
    motion.CurrentValueChanged += new
        EventHandler<SensorReadingEventArgs<MotionReading>>(motion_CurrentValueChanged);
}
```
- Start the Motion API.

```
try
{
    motion.Start();
}
catch (Exception ex)
{
    // Unable to start the Motion API
}
```
- Catch new motion data at the frequency specified with TimeBetweenUpdates.

```
void motion_CurrentValueChanged(object sender, SensorReadingEventArgs<MotionReading> e)
{
    Dispatcher.BeginInvoke(() => CurrentValueChanged(e.SensorReading));
}
```

- Process the motion data.

```
private void CurrentValueChanged(MotionReading e)
{
    if (motion.IsDataValid)
    {
        // MathHelper.ToDegrees(e.Attitude.Yaw).ToString("0")
        // MathHelper.ToDegrees(e.Attitude.Pitch).ToString("0")
        // MathHelper.ToDegrees(e.Attitude.Roll).ToString("0")

        // e.DeviceAcceleration.X.ToString("0.00")
        // e.DeviceAcceleration.Y.ToString("0.00")
        // e.DeviceAcceleration.Z.ToString("0.00")
    }
}
```
- Stop acquisition of data from the sensors.

```
if (motion != null) motion.Stop();
```

Motion Data

Data is returned from the motion API's as values of Attitude, Device Acceleration, Device Rotation Rate, Gravity and a Time Stamp.

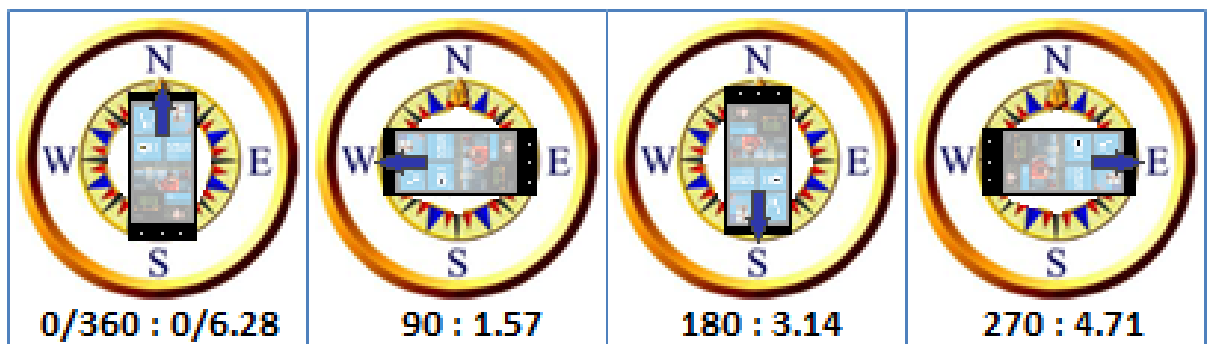
Attitude

The Attitude values return the Yaw, Pitch and Roll of the device in radians. Radians can be converted into degrees with the aid of math helper method `MathHelper.ToDegrees(e.Attitude.Yaw)`

Yaw

The Yaw is calculated as a (compass type) rotation around the centre of the phone (rotating around the central Z axis), starting from True North and moving anticlockwise, where $2 * \text{Pi}$ (6.285714285714286) radians is a full rotation.

Using True North as the start allowing a range of 0 to 360 degrees to be used that reflects a compass directional setting.



Yaw movement has the following rotation pattern:

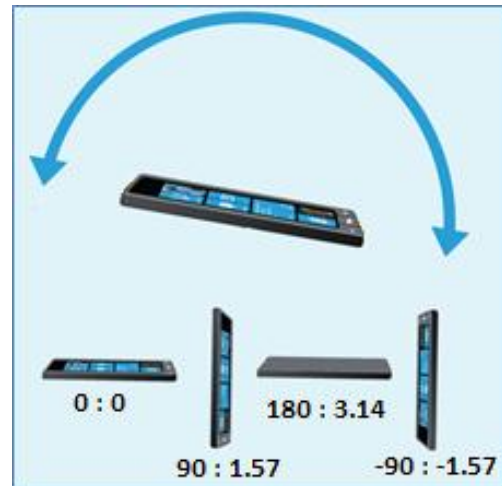
- Device is facing True North up 0 (6.285714285714286) radians or 0 (360) degrees.
- Device rotated left and is facing West 1.571428571428571 radians or 90 degrees.
- Device rotated left is facing South 3.142857142857143 radians or 180 degrees.
- Device rotated left is facing East 4.714285714285714 radians or 270 degrees.

Pitch

The Pitch is calculated as a rotation around the central X axis of $2 * \pi$ (0 to 1.571428571428571) radians registered in both the positive and negative directions.

Pitch movement has the following rotation pattern:

- Device face up 0 radians or 0 degrees.
- Device rotated forwards and is standing up facing user 1.571428571428571 radians or 90 degrees.
- Device rotated forwards and is face down 3.142857142857143 radians or 180 degrees.
- Device rotated forwards and is upside down facing away from the user -1.571428571428571 radians or -90 degrees.

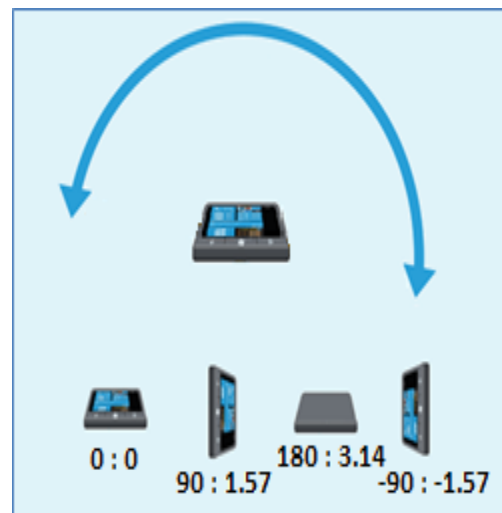


Roll

The Roll is calculated as a rotation around the central Y axis of $2 * \pi$ (0 to 1.571428571428571) radians registered in both the positive and negative directions.

Roll movement has the following rotation pattern:

- Device is face up 0 radians or 0 degrees.
- Device rotated right and is standing on right edge 1.571428571428571 radians or 90 degrees.
- Device rotated right and is face down 3.142857142857143 radians or 180 degrees.
- Device rotated right and is standing on left edge -1.571428571428571 radians or -90 degrees.



When designing the UI, allowing the user to actually see the screen is an important consideration, the following table shows useable ranges on the Yaw, Pitch and Roll in both degrees and radians.

Yaw	Pitch	Roll
Will always see the screen, but the orientation may need setting in code to make some UI intuitive	+16 to +57 +0.28 to +0.99 Can see the screen clearly	-18 to +18 -0.30 to +0.30 Can see the screen clearly
	0 to +16 and +57 to +79 0.0 to +0.28 and +0.99 to +1.38 Can just see the screen	-32 to -18 and +18 to +32 -0.55 to -0.30 and +0.30 to +0.55 Can just see the screen
	-20 to 0.0 and +79 to +99 -0.35 to 0.0 and +1.38 to +1.73 Cannot see the screen	-90 to -32 and +32 to +90 -1.57 to -0.55 and +0.55 to +1.57 Cannot see the screen

Device Acceleration

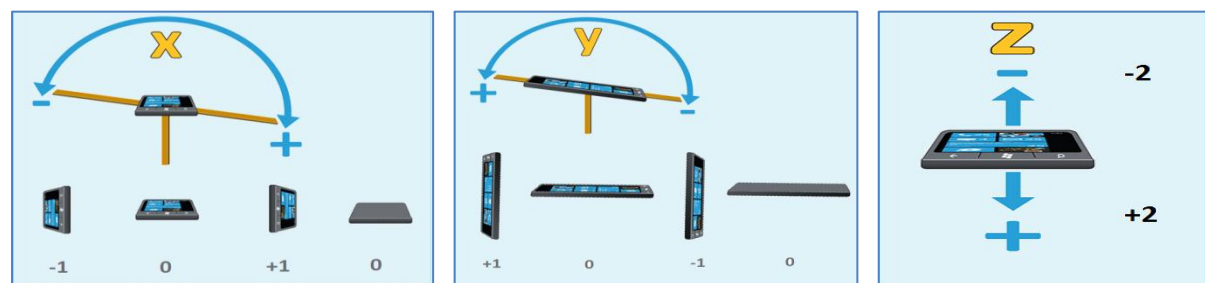
The Device Acceleration values return the linear acceleration of the device on the X, Y and Z axis in gravitational units. These are the non-gravitational acceleration values generated only while the device is moving the range is -2 to +2 depending on the speed of movement.

Device Rotation Rate

The Device Rotation Rate values return the rotational velocity of the device on the X, Y and Z axis. The value of rotation velocity is returned in radians per second, where no movement on an axes returns 0, the value increases as the rotation gets faster ($2 * \pi$ radians is a full rotation). See the Gyroscope section above for a range of output.

Gravity

The Gravity values return the gravity vector associated with the devices current position.



The following table shows the gravitational values returned.

X axis	Y axis	Z axis
Tilted left on its edge -1.	Tilted up standing on its edge -1.	With no movement face up -1
Centre face up 0	Centre face up 0	With no movement face down -1
Tilted right on its edge +1.	Tilted back standing on its edge +1.	With no movement any edge 0
Centre face down 0	Centre face down 0	

See the Accelerometer section above for a range of output.

Time Stamp

The Time Stamp value returns the time that the motion reading was taken from the sensors. Several time and data formats are available to help processing sensor data by time stamping sensor readings.

Code Listings: The following code listings provide complete coding solutions to the Motion section.

User Interaction	Notes
043 BasicMotion	Get motion data from the sensors and outputs the raw data to screen.
044 MotionTest	Gets motion data, Attitude (Yaw, Pitch and Roll) and Acceleration (X,Y,Z).
045 AR_Test	Basic augmented reality movement test based on the Motion API data.

Launchers and Choosers

The Windows Phone model requires an application to be sandboxed, that is only one 3rd party application can run at a time and an application is only allowed access to its own data and storage area.

To allow the integration of phone specific features and data transfer between specified applications Microsoft have provided a set of API's called Launchers and Choosers.

Launchers and Choosers can be called in an application to execute prebuilt applications (tasks) that provide access to useful phone features.

The only difference between a Launcher and Chooser is that a Launcher does not return any data to the calling application on completion of the task and a Chooser returns some kind of data to the calling application on completion of the task.



When a Launcher or Chooser (task) is used the following must be considered:

- When a task is called in an application. The `OnNavigationFrom` event fires in the current page and the `Deactivated` event fires in the application. This forces control to pass to the task.
- While a task is running the calling application may or may not be tombstoned, therefore configuration data and application state data needs to be managed.
- When the task finishes, the `Activate` event in the calling application is fired and the `OnNavigateTo` event fires in the calling page. Note on returning to the calling application its state may be a new instance of the application or not.

The following tables list supported Launchers and Choosers:

Phone Task Type	Return Type	Features
PhoneCallTask	Launcher	Make a phone call
PhoneNumberChooserTask	Chooser	Return a phone number from the contacts application
SavePhoneNumberTask	Chooser	Save a phone number to the contacts application

Text Task Type	Return Type	Features
SmsComposeTask	Launcher	Send text messages

Email Task Type	Return Type	Features
EmailComposeTask	Launcher	Send email
EmailAddressChooserTask	Chooser	Return an email address from the contacts application
SaveEmailAddressTask	Chooser	Save email address to the contacts application

Web Task Type	Return Type	Features
SearchTask	Launcher	Launch the web searcher
WebBrowserTask	Launcher	Launch the web browser

Maps Task Type	Return Type	Features
BingMapsTask	Launcher	Bing Map centred at current or specified location
BingMapsDirectionsTask	Launcher	Bing Map directions, start and end gives street route

Media Task Type	Return Type	Features
CameraCaptureTask	Chooser	Take pictures using the camera
PhotoChooserTask	Chooser	Select a photo from the chooser
MediaPlayerLauncher	Launcher	Launches the media player
SaveRingtoneTask	Chooser	Save a ring tone to the ringtone list

Marketplace Task Type	Return Type	Features
MarketplaceHubTask	Launcher	Launces the Marketplace Hub
MarketplaceSearchTask	Launcher	Displayed the search results for a Marketplace item
MarketplaceDetailTask	Launcher	Displays the details page of a Marketplace item
MarketplaceReviewTask	Launcher	Displays the review page for a selected Marketplace App

Contacts & Invites Task Type	Return Type	Features
SaveContactTask	Chooser	Save contacts to the contacts application
AddressChooserTask	Chooser	Return an address from the contacts application
ShareLinkTask	Launcher	Setup links on social networks
ShareStatusTask	Launcher	Share status messages on social networks
GameInviteTask	Chooser	Invite players to multiplayer games

Utilities Task Type	Return Type	Features
ConnectionSettingsTask	Launcher	Edit connection settings

The procedure for implementing all Launchers follows a similar format.

The `Microsoft.Phone.Tasks` class is required to be added to the build.

The following syntax is used to implement Launchers; the `PhoneCallTask` will be used as an example:

- Access the phone call task to enable users to make a phone call
`PhoneCallTask phoneCallTask = new PhoneCallTask();`
- Get the details from the user
`phoneCallTask.PhoneNumber = NumberTextBox.Text;`
`phoneCallTask.DisplayName = CallerTextBox.Text;`
- Activate the phone call task
`phoneCallTask.Show();`

The procedure for implementing all Choosers follows a similar format.

The `Microsoft.Phone.Tasks` class is required to be added to the build.

The following syntax is used to implement Choosers; the `CameraCaptureTask` will be used as an example:

- Initializes the task object and assign the Completed handler in the page constructor.

```
cameraCaptureTask = new CameraCaptureTask();  
cameraCaptureTask.Completed += new EventHandler<PhotoResult>(cameraCaptureTask_Completed);
```
- Launch the task.

```
try  
{  
    cameraCaptureTask.Show();  
}  
catch (System.InvalidOperationException ex)  
{  
    MessageBox.Show("An error occurred. " + ex.Message);  
}
```
- Capture the return data

```
void cameraCaptureTask_Completed(object sender, PhotoResult e)  
{  
    if (e.TaskResult == TaskResult.OK)  
    {  
        MessageBox.Show(e.ChosenPhoto.Length.ToString());  
        System.Windows.Media.Imaging.BitmapImage bmp = new  
        System.Windows.Media.Imaging.BitmapImage();  
        bmp.SetSource(e.ChosenPhoto);  
        myImage.Source = bmp;  
    }  
}
```

Code Listings: The following code listing provides a complete coding solution to the Launchers and Choosers section.

User Interaction	Notes
046 LaunchersChoosers	<p>Implements the following Launchers and Choosers API's:</p> <ul style="list-style-type: none">• Phone Tasks, make a phone call, return and save a phone number.• Text Task, send a text message.• Email Tasks, send an email, return and save an email address.• Web Tasks, launch a browser or web search.• Map Tasks, find location and directions.• Media Tasks, take pictures, select a photo, launch media player and save ringtones.• Marketplace Tasks, launch marketplace hub and display results, details and review pages.• Contacts Tasks, save / return contacts, setup links, share status messages and invite players.• Utility Tasks, edit connection settings.

Radio

The Windows 7 Phone has a built in FM radio receiver as standard and its functionality can be accessed via the `Microsoft.Devices.Radio` class. Earphones need to be attached to the device to use the FM receiver as the earphone cable is used as an aerial.

There are several options supported by the namespace that offer some interesting possibilities of integrating commercial broadcasts into a game:

- `CurrentRegion`, sets the frequency region information for the radio.
- `Frequency`, tunes the FM radio to a specified frequency.
- `Instance`, returns an instance of the FM radio.
- `PowerMode`, sets the `RadioPowerMode` of the device, setting this property turns the device radio on or off.
- `SignalStrength`, retrieves the received signal strength indicator (RSSI) value for the currently-tuned frequency.
- `RadioPowerMode`, is an enumeration that contains the possible power modes of the radio, the current `RadioPowerMode` values are `On` and `Off`.
- `RadioRegion`, is an enumeration that contains the set of radio frequency regions available for radio tuning, the current values are `Europe`, `Japan`, and `UnitedStates`.



The FM receiver tends to be slightly volatile on initialization and the following design considerations need to be taken into account when accessing the FM receiver:

- It can take up to three seconds for the first `FMRadio` method call to return after the phone boots up.
- After the FM Radio is first initialized, if the phone is running in an active state, the radio methods will normally return within 100 ms (but may not).
- The API warns against setting up the FM Radio or synchronizing the Silverlight UI thread while the application is running.
- It is advised to delay sending further commands to the FM Radio until at least one second after the initial call to enable the FM Radio.

The `Microsoft.Devices.Radio` class is required to be added to the build.

The following syntax is used to access and tune the FM receiver:

- Set the radio region.
`App.myRadio.CurrentRegion = RadioRegion.Europe;`
- Set the frequency.
`App.myRadio.Frequency = 94.2;`
- Turn the radio on.
`App.myRadio.PowerMode = RadioPowerMode.On;`
- Turn the radio OFF.
`App.myRadio.PowerMode = RadioPowerMode.Off;`

Code Listings: The following code listing provides a complete coding solution to the Radio section.

User Interaction	Notes
047 Radio	Basic Radio application (needs earphones to run) search, tune, region, play and stop (still work in progress).

Maps and Locations

Location finding and Mapping allows the exploration of real world locations to be included into the Pervasive Game Model. As this has the potential to be a fairly important feature of any pervasive games developed I will look at the implementation requirements of mapping in some detail.

Bing Maps are the API set offered on the Windows 7 Platform and before the API's can be used effectively registration is required.



Bing Map Registration

A Bing Map Key is required from the Bing Map Account Centre [<https://www.bingmapsportal.com>] to activate any map object included into a project.



A Windows Live ID is required for sign in to the site, so one needs setting up prior to registration. After logging in the registration process is fairly straight forward, just answer the questions and click the 'Create Key' button, then a key is generated. There is no fee for this service but all transactions are logged and data on usage of all mapping services are retained. A detailed categorised report on all transactions registered to a Bing Map Key can be accessed via the Bing Map Account Centre for free.

Although mapping services are reported to be free in all the documentation, when I initially researched this topic I found a restriction on usage of over 100,000 hits per day for a commercially released application. At this level of usage there may be an incurred fee, unfortunately I have lost the link and have been unable to find it again so this is unsubstantiated although worth mentioning. That said if I had an application out there with 100,000 plus paying users using the application every day, a small fee would be the least of my worries.

Once acquired the registration key needs to be included into the map XAML, this removes all restriction on usage and the registration overlay displayed on the map object.

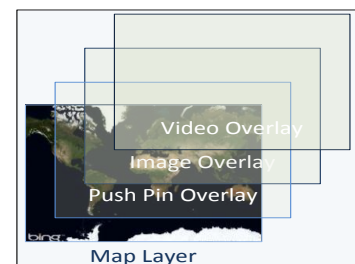
The key needs to be included into the map XAML in the following format:

```
CredentialsProvider="AgegP7PK9uptxnifnMh1JH0gCXRZ3W_AAAAAAA_F5YdqduhAK6REqAAAAAAAAAAAA"
```

Bing Maps

Bing maps can be displayed in several modes (road and aerial being the most common) and there is an overlay feature that can be used to display Push Pins, images and videos.

In this section I will focus on the implementation of the following very usefully Bing Maps features:



- Basic Options, setting the mode, zoom, scale, language and display options.
- Locking Maps, locking the movement and levels of two maps together.
- Finding a Location, finding locations based on GeoCoordinates.
- Basic Collision, basic collision based on GeoCoordinates and a defined search rectangle.
- Collision, validated collision based on the World Map longitude and latitude boundaries of the map and adding overlays to test the collision.
- Push Pins, adding and removing push pins, push pin events, push pin images, images and videos anchored to locations and displayed on overlays.

There are a couple of classes required in the build to access the Mapping API's `System.Globalization`, `System.Device.Location` and `Microsoft.Phone.Controls.Maps`.

The following syntax is used to set up Basic Map options:

- A map can be set up as XAML.

```
<my:Map Height="251" HorizontalAlignment="Left" Margin="0,440,0,0" Name="map1"
CredentialsProvider=" AgegP7PK9uptxnifnMh1JH0gCXRZ3W_AAAAAAA_F5YdqduhAK6REqAAAAAAAAAAAA "
VerticalAlignment="Top" Width="456" />
```
- Setting the road mode

```
map1.Mode = new RoadMode();
```
- Setting the arial mode

```
map1.Mode = new AerialMode();
```
- Setting the arial mode with labels

```
map1.Mode = new AerialMode(true);
```
- Adding a zoom option to a map

```
map1.ZoomBarVisibility = Visibility.Visible;
```
- Adding a scale option to a map

```
map1.ScaleVisibility = Visibility.Visible;
```
- Making a Map visible

```
Map1.Visibility = Visibility.Visible;
```
- Zooming in (out is --)

```
double zoom = map1.ZoomLevel;
map1.ZoomLevel = ++zoom;
```
- Setting the language via a list entry

```
ListBoxItem tempListBox = ((sender as ListBox).SelectedItem as ListBoxItem);
String tempString = tempListBox.Content.ToString();
CultureInfo tempCulture = new CultureInfo(tempString);
if (tempCulture != null && map1 != null) map1.Mode.Culture = tempCulture;
```

The following syntax is used to Lock Maps together via XAML:

- Lock the movement and zoom on two maps.

```
<my:Map Name="map1" Center="{Binding Center, ElementName=map2, Mode=TwoWay}"
ZoomLevel="{Binding ZoomLevel, ElementName=map2, Mode=TwoWay}" AnimationLevel="UserInput" >
</my:Map>
<my:Map Name="map2" AnimationLevel="UserInput"></my:Map>
```

The following syntax is used to display a map location based on GeoCoordinates:

- Variables

```
double latitude = 0;
double longitude = 0;
double zoomLevel = 1;
```
- Get values from text boxes, alternative way using TryParse

```
double.TryParse(LatTextBox.Text, NumberStyles.Float,CultureInfo.InvariantCulture, out
latitude);
double.TryParse(LongTextBox.Text, NumberStyles.Float,CultureInfo.InvariantCulture, out
longitude);
double.TryParse(ZoomTextBox.Text, NumberStyles.Float,CultureInfo.InvariantCulture, out
zoomLevel);
```
- Initializes instance of the GeoCoordinate class from latitude and longitude data and show map.

```
map1.SetView(new GeoCoordinate(latitude, longitude), zoomLevel);
```

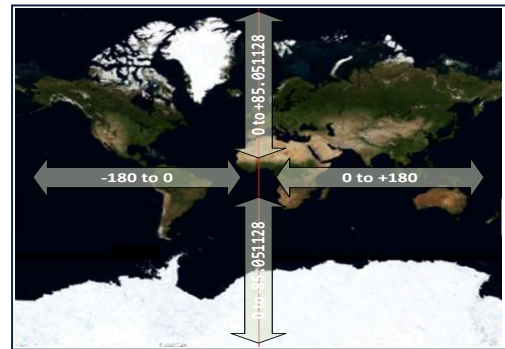

The following syntax is used for basic collision based on GeoCoordinates and a defined search rectangle:

- Make your own event handler for change of frame

```
map1.ViewChangeOnFrame += new
EventHandler<Microsoft.Phone.Controls.Maps.MapEventArgs>(MyMap_ViewChangeOnFrame);
```
- Event handler code to check on collision

```
void MyMap_ViewChangeOnFrame(object sender, Microsoft.Phone.Controls.Maps.MapEventArgs e)
{
    Map map = sender as Map;
    if (map != null)
    {
        if ( ((map.Center.Latitude > -37.691381) && (map.Center.Latitude < -37.691238)) &&
            ((map.Center.Longitude > 176.260488) && (map.Center.Longitude < 176.260734)))
        {
            SearchTextBox.Text = "You found it";
        }
        else
        {
            SearchTextBox.Text = "Keep Looking";
        }
    }
}
```

The following syntax is used for validated collision based on the World Map longitude and latitude boundaries of the map:



- Set up boundary definitions

```
private const double LATITUDE_NORTH_LIMIT = 85.051128;
private const double LATITUDE_SOUTH_LIMIT = -85.051128;
private const double LONGITUDE_EAST_LIMIT = 180;
private const double LONGITUDE_WEST_LIMIT = -180;
```
- Event handler to update the page changes

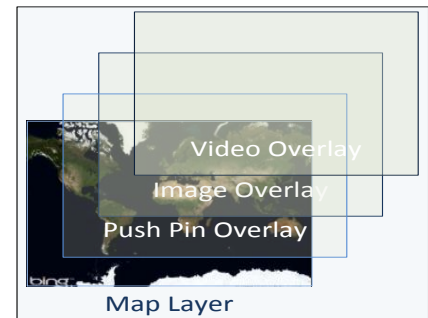
```
map1.ViewChangeOnFrame += new
EventHandler<Microsoft.Phone.Controls.Maps.MapEventArgs>(MyMap_ViewChangeOnFrame);
```

```
void MyMap_ViewChangeOnFrame(object sender, Microsoft.Phone.Controls.Maps.MapEventArgs e)
{
    Map map = sender as Map;
    GeoCoordinate locationToFind = new GeoCoordinate(double.Parse(LatitudeTextBox.Text,
        NumberStyles.Float, CultureInfo.InvariantCulture),
        double.Parse(LongitudeTextBox.Text, NumberStyles.Float,
        CultureInfo.InvariantCulture));
    double boxHeight = double.Parse(HeightTextBox.Text, NumberStyles.Float,
        CultureInfo.InvariantCulture);
    double boxWidth = double.Parse(WidthTextBox.Text, NumberStyles.Float,
        CultureInfo.InvariantCulture);
    if (map != null)
    {
        if (LocationCollision(map.Center, locationToFind, boxHeight, boxWidth) == true)
        {
            SearchTextBox.Text = "You found it";
        }
        else SearchTextBox.Text = "Keep Looking";
    }
}
```
- Method that implements the collision, see listing 048 Maps, option Collision With World Validation for implementation.

```
private bool LocationCollision(GeoCoordinate currentLocation, GeoCoordinate locationToFind,
    double boxHeight, double boxWidth)
```

The following syntax is used for adding and removing push pins elements (push pin events, push pin images, images and videos) anchored to locations and displayed on overlays.

There are a couple more classes required in the build to access the Mapping API's and image options `System.Globalization`, `System.Device.Location`, `Microsoft.Phone.Controls.Maps`, `System.Windows.Media.Imaging` and `System.Windows.Resources`.



- Define a layer for the map and position.

```
private MapLayer myLayer = null;
private GeoCoordinate mapCenter;
```
- Set up the layer.

```
myLayer = new MapLayer();
map1.Children.Add(myLayer);
```
- Add a Push Pin to the layer

```
mapCenter = map1.Center;
Pushpin pin1 = new Pushpin();
pin1.Location = mapCenter;
myLayer.AddChild(pin1, mapCenter);
```
- Clear the layer (pushpins)

```
myLayer.Children.Clear();
```
- Add an image to the layer

```
Image image = new Image();
image.Source = new BitmapImage(new Uri("/Maps;component/test.jpg",
                                     UriKind.RelativeOrAbsolute));
image.HorizontalAlignment = HorizontalAlignment.Stretch;
image.VerticalAlignment = VerticalAlignment.Stretch;
image.ImageFailed += MediaFailed;
LocationRect locationRect = new LocationRect(-37.6912708, 176.2606689, -37.6912208,
                                              176.2607189);

myLayer.AddChild(image, locationRect);
GeoCoordinate locationToFind = new GeoCoordinate(-37.6912708, 176.2606689693);
map1.SetView(locationToFind, 21);
```
- Set an image up as a pushpin

```
mapCenter = map1.Center;
Image pinImage = new Image();
pinImage.Source = new System.Windows.Media.Imaging.BitmapImage(new
    Uri("/Maps;component/Pushpin.png", UriKind.RelativeOrAbsolute));
pinImage.Stretch = System.Windows.Media.Stretch.None;
PositionOrigin position = PositionOrigin.Center;
myLayer.AddChild(pinImage, mapCenter, position);
```
- Adding a push pin and an associated event.

```
Pushpin pinEvn = new Pushpin();
pinEvn.Location = new GeoCoordinate(-37.69127, 176.26066);
pinEvn.Content = "1";
pinEvn.MouseLeftButtonUp += new MouseButtonEventHandler(pinEvn_MouseLeftButtonUp);
myLayer.AddChild(pinEvn, pinEvn.Location);
map1.SetView(pinEvn.Location, 21);

void pinEvn_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)
{
    myLayer.Children.Clear();
}
```

- Add a video to a video layer (only one video playing per map is supported)

```

MapLayer videoLayer;
videoLayer = new MapLayer();
map1.Children.Add(videoLayer);
videoLayer.Children.Clear();
MediaElement video = new MediaElement();
video.Source = new Uri("/Media/video1.wmv", UriKind.RelativeOrAbsolute);
video.Width = 250;
video.Height = 200;
video.Position = TimeSpan.FromMilliseconds(0);
PositionOrigin position = PositionOrigin.Center;
videoLayer.AddChild(video, map1.Center, position);

```
- Switch off the video by clearing the video layer.

```

videoLayer.Children.Clear();

```

Location

Location services provide data from several sources GPS, Wi-Fi and cellular data via an event driven (managed code) interface. This location data is a key factor in developing location-aware games, in laymen's terms this means we can track a device (and its user) in the real world.

There are three layers to Location services data retrieval:

- The bottom layer (hardware) represents all communication with the hardware on the device, the GPS receiver (GPS satellites / cell phone towers), Wi-Fi and cellular radio. All device hardware returns location data but with varying degrees of accuracy and power usage.
- The middle layer (code) is the native code layer, device data is retrieved here (and correlated) depending on the hardware available on the device (at the time) and the requirement (accuracy / power consumption) of the application. Data is also retrieved via a Microsoft Web Service that returns location data from an on line database and the location setting set on the device.
- The top layer (API) represents a manage interface that offers functionality to switch on / off the services, set the level of accuracy and retrieve location data as it is received from the hardware via the native code layer.



Location data is a tricky area as there are several parameters affecting accuracy and also compliance for registration, as a rule of thumb the following applies:

- Use a low-accuracy setting where possible. Less accurate location data will be retrieved via the Wi-Fi and cellular radio, but this setting will also use less battery power.
- Using a high-accuracy setting will retrieve data from all sources including the GPS receiver, Wi-Fi and cellular radio, but will eat up the battery.
- Turn off the location services when not in use in an application.
- There is a compliance requirement to notify the user that device location data is being solicited and only access location services if the user agrees.
- Location services in high-accuracy mode (when using GPS connections) can take 15 to 120 seconds to establish a correct initial location reading (if the service has never been used on the device), so initialization strategies need to be considered. The application needs to warn the user of this fact while initializing. The low-accuracy setting does not wait for the GPS to return so tends to be quicker as an initial start-up service.
- Applications need to handle lack of data gracefully, data is retrieved from several sources so this is rare, but it may happen, the `StatusChanged` event can be polled to check if data is available.

Movement Threshold

The last consideration for retrieving location data is the movement threshold of the device, when to trigger new location data relative to the device movement.

The `GeoCoordinateWatcher` class exposes the `MovementThreshold` property which can be set to define the minimum change in position before a `PositionChanged` event is fired.

The following settings are recommended:

- Zero meters will cause the event to fire every second, and is useful for navigational applications.
- 20 meters will filter out any device sensitivity noise and only report positional changes.

The following syntax is used for processing location data.

There are a couple more classes required in the build to access the Mapping API's and image options `System.Globalization`, `System.Device.Location` and `Microsoft.Phone.Controls.Maps`.

- Define Location Service
`GeoCoordinateWatcher watcher;`
- Set Location Service to low-accuracy
`StartLocationService(GeoPositionAccuracy.Default);`
- Set Location Service to high-accuracy
`StartLocationService(GeoPositionAccuracy.High);`
- Start up the location data acquisition

```
private void StartLocationService(GeoPositionAccuracy accuracy)
{
    watcher = new GeoCoordinateWatcher(accuracy);
    watcher.MovementThreshold = 20;

    watcher.StatusChanged += new
        EventHandler<GeoPositionStatusChangedEventArgs>(watcher_StatusChanged);

    watcher.PositionChanged += new
        EventHandler<GeoPositionChangedEventArgs<GeoCoordinate>>(watcher_PositionChanged);

    watcher.Start();
}
```
- Handler for the StatusChanged event.

```
void watcher_StatusChanged(object sender, GeoPositionStatusChangedEventArgs e)
{
    Deployment.Current.Dispatcher.BeginInvoke(() => MyStatusChanged(e));
}
```
- Handler for the PositionChanged event.

```
void watcher_PositionChanged(object sender, GeoPositionChangedEventArgs<GeoCoordinate> e)
{
    Deployment.Current.Dispatcher.BeginInvoke(() => MyPositionChanged(e));
}
```

- Break down the StatusChanged data

```
void MyStatusChanged(GeoPositionStatusChangedEventArgs e)
{
    switch (e.Status)
    {
        case GeoPositionStatus.Disabled:        // Some code
                                                break;

        case GeoPositionStatus.Initializing:     // Some code
                                                break;

        case GeoPositionStatus.NoData:           // Some code
                                                break;

        case GeoPositionStatus.Ready:            // Some code
                                                break;
    }
}
```
- Process the PositionChanged event handler data

```
void MyPositionChanged(GeoPositionChangedEventArgs<GeoCoordinate> e)
{
    GeoCoordinate ourLocation = new GeoCoordinate( e.Position.Location.Latitude,
                                                    e.Position.Location.Longitude);

    map1.SetView(ourLocation, map1.ZoomLevel);
}
```
- Stop the Location services

```
if (watcher != null)
{
    watcher.Stop();
}
```

Code Listings: The following code listing provides a complete coding solution to the Maps and Locations sections.

User Interaction	Notes
048 Maps	<p>Implements the following Bing Maps and Location Services API's:</p> <ul style="list-style-type: none"> • Basic Options, sets mode, zoom, scale, language and display options on maps. • Lock Maps, locks the movement of maps together. • Find Location, finds locations based on GeoCoordinates. • Basic Collision, basic collision based on GeoCoordinates and a defined search rectangle. • Collision, validated collision based on the geometry of the map, adds testing overlays. • Push Pins, adding and removing to locations push pins (events / images), images and videos. • Track Phone, displays the location of the phone on a map using Location and Map services.

Tiles and Notifications

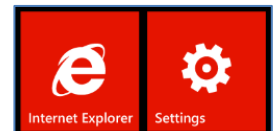
The new Mango release has a very nice feature for Pervasive Game development. Messages can be sent to a device and displayed as text or images, the mechanism used is via the Application Tile display and the Notification services API's.

This has some interesting uses, as once a device is registered messages can be sent to it (and multiple registered devices) and the received text message/s can be used in the application logic.



Tiles

Tiles are used in the Start Page of a Windows 7 Phone as a graphical representation of an application (like an icon). When a tile is clicked the application will execute and open at a specific page.



The interesting thing about Tiles is that the graphics and text on a tile can be updated in real time. This opens the door to some interesting real time messaging implementations by displaying information to the user from the application or external sources via the tile display mechanism.

There are currently two types of tile supported, application tiles and secondary tiles. Both types allow graphics and text to be displayed via a front and back tile mode.

An application tile is created by pinning an application to the start menu (press and hold the application in the app list until the pin menu option appears) and tends to be used for starting an application.

A secondary tile is usually implemented via a user selection (or service) in an application and executed programmatically, secondary tiles tend to be used for displaying information (like a weather forecast update).

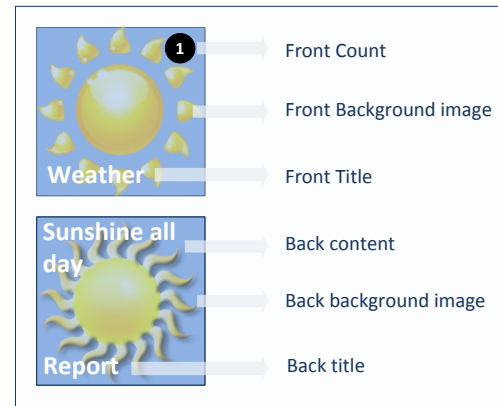
The following table defines the tile type characteristics:

Tile Type	Tile Activation	Tile Deactivation	Tile Update
Application	Created by user when the application name in the Application List is tapped and held, then the pin to start option in the menu is selected.	The tile can be removed from the display by: <ul style="list-style-type: none">Unpinning the Tile from the Start Menu.Uninstalling the application.Programmatically by calling the Delete() event. All of these do not delete the tile; it is always active so that the tile can be updated programmatically (so its current state is available for future display).	<ul style="list-style-type: none">ShellTile APIs (first item in the ActiveTile collection to be updated).Push Notifications.ShellTileSchedule APIs.
Secondary	Created by user input to an application.	The tile can be removed from the display by: <ul style="list-style-type: none">User unpinning the Tile from the Start Menu.User uninstalling the application.Programmatically by calling the Delete() event.	<ul style="list-style-type: none">ShellTile APIs (first item in the ActiveTile collection to be updated).Push Notifications.ShellTileSchedule APIs.

Mango (OS 7.1) supports tiles with two sides that can flip over when displayed. If data is only supplied for the front of a tile then the tile will not flip over when displayed. The format is the same for both application tiles and secondary tiles.

The following are the supplied configuration options:

- Front Count (Badge), integer counter 1 to 99, zero or not set sets the counter display off.
- Front Background Image, image representing the application, the default image on initialization is the Background.png. For the secondary tile the images resources must be created using local resources.
- Front Title, title text string (15 characters) for the front of the tile.
- Back Content, content text string (40 characters) for the back of the tile.
- Back Background image, image representing the application. For secondary tile the images resources must be created using local resources.
- Back Title, title text string (15 characters) for the back of the tile.



There are several recommendations for the use of tile images that need to be considered to pass compliance standards:

- Only .jpg or .png (and transparent) file types are supported.
- Because of the variability of networks and file size performance issues only local image resources should be used.
- Tiles size should be kept to 173 x 173 pixels, otherwise automatic image stretching will caused reduced performance.
- There is no https supported for remote image access and all image downloads are limited to 80 KB, with the restraint that remote image downloads will time out after 30 seconds if the download is not completed.

The following syntax is used to process Application and Secondary Tiles.

The `Microsoft.Phone.Shell` class is required in the build.

- To set an Application Tile, the Application Tile is always the first Tile, even if it is not pinned to Start.

```
ShellTile ApplicationTileToFind = ShellTile.ActiveTiles.First();
```
- Check if the Application Tile can be found, then set the Tile properties and update the Tile

```
if (ApplicationTileToFind != null)
{
    StandardTileData NewApplicationTileData = new StandardTileData
    {
        Title = "Testing Title",
        BackgroundImage = new Uri("Red.png", UriKind.Relative),
        Count = 1,
        BackTitle = "Testing Back Title",
        BackBackgroundImage = new Uri("Pink.png", UriKind.Relative),
        BackContent = "This is the back"
    };
    ApplicationTileToFind.Update(NewApplicationTileData);
}
```

- To set a Secondary Tile, first look to see if the tile already exists and if so, don't try to create again.

```
ShellTile TileToFind = ShellTile.ActiveTiles.FirstOrDefault(x =>
    x.NavigationUri.ToString().Contains("DefaultTitle=FromTile"));
```
- Check if the Secondary Tile can be found, if not then then set the Tile properties and create the Tile.

```
if (TileToFind == null)
{
    StandardTileData NewTileData = new StandardTileData
    {
        BackgroundImage = new Uri("Blue.jpg", UriKind.Relative),
        Title = "Secondary Tile",
        Count = 2,
        BackTitle = "Back of Tile",
        BackContent = "Welcome to the back of the Tile",
        BackBackgroundImage = new Uri("Background.png", UriKind.Relative)
    };

    ShellTile.Create(new Uri("/MainPage.xaml?DefaultTitle=FromTile", UriKind.Relative),
        NewTileData);
}
```

Application and Secondary Tiles front backgrounds can be updated on a schedule using the `ShellTileSchedule` class.

A one-shot or repeating schedule (duration and repeat count can be set) can be setup and will continue to run even if the application is not activated.

The documentation advises that the Application Tile schedule settings need to be stored and the scheduler re-started each time the application starts, as schedule errors have the potential to disable the scheduler.

The following syntax is used to process the Tile Scheduler.

- Set up the Scheduler class

```
ShellTileSchedule SampleTileSchedule = new ShellTileSchedule();
```
- Set up the Scheduler parameters and start

```
SampleTileSchedule.Interval = UpdateInterval.EveryHour;
SampleTileSchedule.Recurrence = UpdateRecurrence.Interval;
SampleTileSchedule.RemoteImageUri = new
Uri(@"http://www.weather.gov/forecasts/graphical/images/conus/MaxT1_conus.png");
SampleTileSchedule.Start();
```
- Stop the Scheduler

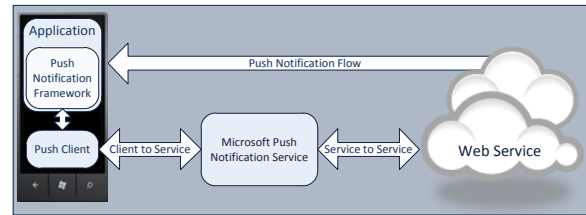
```
SampleTileSchedule.Stop();
```

Code Listings: The following code listing provides a complete coding solution to the Tiles section.

User Interaction	Notes
049 ChangingTiles	Process Application and Secondary Tiles, also sets up a Tile Scheduler.

Push Notifications

Push Notification Services allow developers to send data between a mobile device and a Web Service, this is a fantastic feature of the Mango OS.



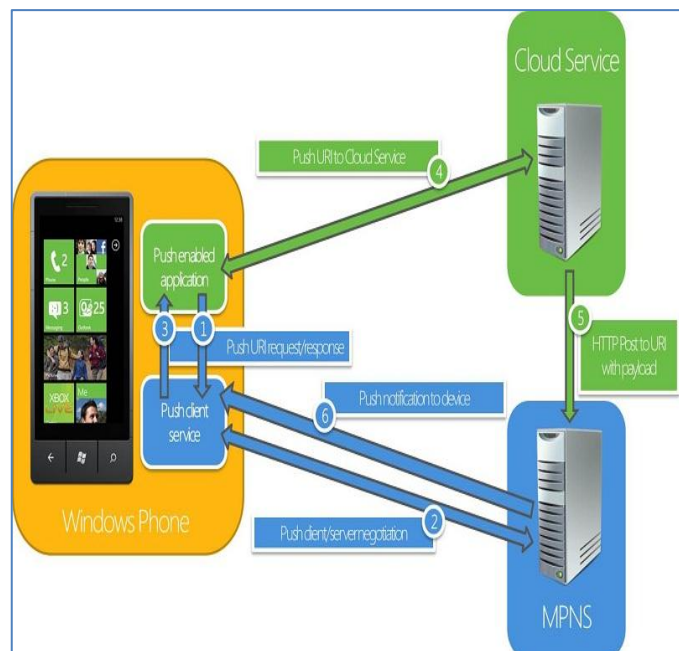
Although setting up a web server is currently outside the scope of this project, as there is potential here for some very interesting pervasive game interactions, I will look at this area in some detail by including some example code simply set up on a localhost server (the device will connect to a local network via a Wi-Fi modem or cable when registration is required).

It is assumed that there is a familiarity with multi-project builds and client server project configurations in the Microsoft Integrated Development Environment when dealing with the code examples in this section of the project.

As the Web Server applications are built in Microsoft Visual Studio Web Developer 2010, the Phone Client applications are built in Microsoft Visual Studio for Windows Phone 2010 and the final example is a multi-project build (see code listings 50 to 56).

The diagram steps through the process of sending a Push Notification from a Web server to a Phone application:

1. A push enabled Phone Application (client) requests a push notification URI from the Push Client Service.
2. The Push Client Service connects to the Microsoft Push Notification Service (MPNS) and a notification URI is returned.
3. The notification URI is passed to the Client Application on the phone.
4. The Client Application on the phone sends a URI to Cloud Services.
5. A Push notification is sent using the URI to Microsoft Push Notification Service from the Web Service server (cloud connection).
6. Microsoft Push Notification Service sends the push notification to the application on the phone (client).



[Push Notifications: Diagram from MSDN <http://msdn.microsoft.com>]

There are several restrictions regarding the use of Push Notifications, the following list notes the main performance and compliance issues:

- Only one push notification channel per application is supported, on error an `InvalidOperationException(PushNotificationChannelQuotaExceeded)` exception is thrown.
- Only 30 push notification channels per device are supported, on error an `InvalidOperationException(PushNotificationChannelQuotaExceeded)` exception is thrown.
- Personal user information should not be included in the data transmitted.
- Real world connection states are variable and a defensive coding strategy is recommended to deal with corrupt transmission data and any loss of service.
- Data packets (including configuration data) are only 4KB (header 1KB and 3KB payload), excesses are clipped and a Web Exception status `WebExceptionStatus.ProtocolError` exception is thrown.
- There is no end-to-end confirmation of a Push Notification delivery (web service to the device) although there is confirmation from the Microsoft Push Notification Service to the web service. So it is worth noting that the final stage of the service transmission is where problems may need to be dealt with, that is the data stream from the Microsoft Push Notification Service to the application on the device.

There are three ways a Push Notification can be sent from a Web Service to an Application, these are as a Toast message, Tile and as Raw data.

Toast Notifications

Toast Notifications are displayed on the top of the screen and stay visible for 10 seconds. When they are tapped the application is loaded and the required application page is displayed. Data from the Toast can also be captured inside the application.

The Toast Notification structure is as follows:

- Title, defined as the Text1 property in the XML schema and contains a boldface string.
- Content, defined as the Text2 property in the XML schema and contains a string that is displayed after the title.
- Parameter, defined as the Param property in the XML schema, contains the page to launch in the application, when the Toast is clicked.



The maximum amount of text that can be transmitted is:

- If only the Title string is used 40.
- If only the Content string is used 47.
- With and even split between Title and Content strings 41.

The following syntax is used to send and receive Toast Notifications.

There are a couple more classes required in the build to implement Push services:

- On the Server side `System.Net`, `System.IO` and `System.Text`.
- On the Client side `Microsoft.Phone.Notification` and `System.Text`.

On the Server side

- The code needs to be in a try / catch block because of the network unpredictability

```
try
{
```
- Get the URI that the Microsoft Push Notification Service returns to the push client when creating a notification channel.

```
string subscriptionUri = "http://sn1.notify.live.net/throttledthirdparty/01.00/
    AAG8wQEey7jsRbCsvT49JIIAAgAAAAADAQAAAAQUZm520jIzOEQ2NDJDRkI5MEVFMEQ";
HttpWebRequest sendNotificationRequest = (HttpWebRequest)WebRequest.Create(subscriptionUri);
```
- Create an HTTPWebRequest that posts the toast notification to the Microsoft Push Notification Service.

```
sendNotificationRequest.Method = "POST";
```
- Create the toast message.

```
string toastMessage = "<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
    "<wp:Notification xmlns:wp=\"WPNotification\">" +
    "<wp:Toast>" +
    "<wp:Text1>" + "My title" + "</wp:Text1>" +
    "<wp:Text2>" + "My Content" + "</wp:Text2>" +
    "<wp:Param>/Page2.xaml?NavigatedFrom=Toast Notification</wp:Param>" +
    "</wp:Toast> " +
    "</wp:Notification>";
```
- Set the notification payload, send and get response.

```
byte[] notificationMessage = Encoding.Default.GetBytes(toastMessage);
sendNotificationRequest.ContentLength = notificationMessage.Length;
sendNotificationRequest.ContentType = "text/xml";
sendNotificationRequest.Headers.Add("X-WindowsPhone-Target", "toast");
sendNotificationRequest.Headers.Add("X-NotificationClass", "2");
using (Stream requestStream = sendNotificationRequest.GetRequestStream())
{
    requestStream.Write(notificationMessage, 0, notificationMessage.Length);
}
HttpWebResponse response = (HttpWebResponse)sendNotificationRequest.GetResponse();
string notificationStatus = response.Headers["X-NotificationStatus"];
string notificationChannelStatus = response.Headers["X-SubscriptionStatus"];
string deviceConnectionStatus = response.Headers["X-DeviceConnectionStatus"];
```
- Display the response from the Microsoft Push Notification Service.

```
TextBox.Text = notificationStatus + " | " + deviceConnectionStatus + " | " +
    notificationChannelStatus;
```
- Catch any errors with the transmission

```
}
catch (Exception ex) { // "Exception caught sending update: " + ex.ToString(); }
```

On the Client side

- Initialize the push channel that is created or found.

```
HttpNotificationChannel pushChannel;
string channelName = "ToastSampleChannel";
pushChannel = HttpNotificationChannel.Find(channelName);
if (pushChannel == null)
{
    pushChannel = new HttpNotificationChannel(channelName);
    pushChannel.ErrorOccurred += new
        EventHandler<NotificationChannelEventArgs>(PushChannel_ErrorOccurred);
    pushChannel.Open();
    pushChannel.BindToShellToast();
}
```

```

else
{
    pushChannel.ErrorOccurred += new
        EventHandler<NotificationChannelErrorEventArgs>(PushChannel_ErrorOccurred);
}

• Process any error handling.
void PushChannel_ErrorOccurred(object sender, NotificationChannelErrorEventArgs e)
{
    Dispatcher.BeginInvoke(() =>
        MessageBox.Show(String.Format("A push notification {0} error occurred. {1} ({2}) {3}",
            e.ErrorType, e.Message, e.ErrorCode, e.ErrorAdditionalData)) );
}

```

Code Listings: The following code listings provide complete coding solutions to the Toast Notifications section.

User Interaction	Notes
050 SendToast	Server side application that sends a Toast Notification.
051 ToastNotificationClient	Client side application that receives Toast Notifications, when the Toast is clicked it will go to specified page in the application.
056 MessagingGame	Work in progress (MS example), currently supported: <ul style="list-style-type: none"> GameServer, registers the phone and sends and logs weather Toast, Tile and Raw messages. GameClient, registers with the server and receives updated weather Toast, Tile and Raw messages.

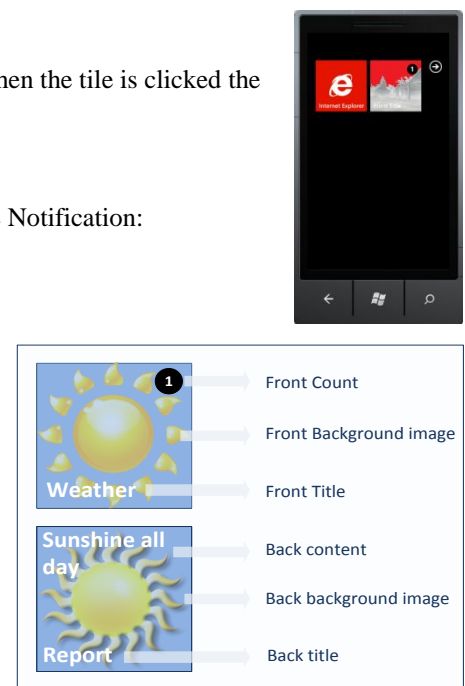
Tile Notifications

Tile Notifications can be used to send data to the Application Tile. When the tile is clicked the application will load and display the required page.

The following are configuration options that can be updated via a Tile Notification:

- Front count (Badge) numbers between 0 to 99.
- Front background image, must be stored locally.
- Front title text string maximum 15 characters long.
- Back content text string maximum 40 characters long.
- Back background image must be stored locally.
- Back title text string maximum 15 characters long.

All the normal recommendations (as previously stated) apply to Tiles that have been updated.



The following syntax is used to send and receive Tile Notifications.

There are a couple more classes required in the build to implement Push services:

- On the Server side `System.Net`, `System.IO` and `System.Text`.
- On the Client side `Microsoft.Phone.Notification`.

On the Server side the structure of the code is very similar to the Toast server, it is only the construction of the actual message that is different so that is all I will note here.

- Create the Tile message.

```
string tileMessage = "<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
    "<wp:Notification xmlns:wp=\"WPNotification\">" +
    "<wp:Tile>" +
    "<wp:BackgroundImage>" + "Blue.jpg" + "</wp:BackgroundImage>" +
    "<wp:Count>" + "1" + "</wp:Count>" +
    "<wp:Title>" + "My Title" + "</wp:Title>" +
    "<wp:BackBackgroundImage>" + "Green.jpg" + "</wp:BackBackgroundImage>" +
    "<wp:BackTitle>" + "My Title" + "</wp:BackTitle>" +
    "<wp:BackContent>" + "My Content" + "</wp:BackContent>" +
    "</wp:Tile> " +
    "</wp:Notification>";

byte[] notificationMessage = Encoding.Default.GetBytes(tileMessage);
sendNotificationRequest.ContentLength = notificationMessage.Length;
sendNotificationRequest.ContentType = "text/xml";
sendNotificationRequest.Headers.Add("X-WindowsPhone-Target", "token");
sendNotificationRequest.Headers.Add("X-NotificationClass", "1");
```

On the Client side the structure of the code is very similar to the Toast client, it is only the channel name and the binding type that is different so that is all I will note here.

- The name of our push channel.

```
string channelName = "TileSampleChannel";
```
- Bind the channel for Tile events.

```
pushChannel.BindToShellTile();
```

Code Listings: The following code listings provide complete coding solutions to the Tile Notifications section.

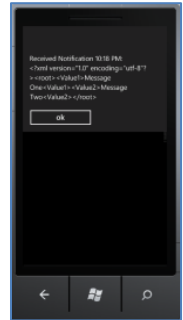
User Interaction	Notes
052 SendTile	Server side application that sends a Tile Notification.
053 TileNotificationClient	Client side application that receives Tile Notifications, when the Tile is clicked it will go to specified page in the application.
056 MessagingGame	Work in progress (MS example), currently supported: <ul style="list-style-type: none">• GameServer, registers the phone and sends and logs weather Toast, Tile and Raw messages.• GameClient, registers with the server and receives updated weather Toast, Tile and Raw messages.

Raw Notifications

Raw Notifications can be used to send data to an application when it is running. If the application is not running then the data is discarded at the Microsoft Push Notification Service layer and not sent to the device application. There is a cap of 4KB on the data but only about 3KB is useable payload data.

The Raw Notification structure is as follows:

- Data string one, defined as the Value1 property in the XML schema and contains a data string.
- Data string two, defined as the Value2 property in the XML schema and contains a data string.



The maximum amount of text that can be transmitted is about 3,345 characters (maximum) split between the two data strings.

The following syntax is used to send and receive Raw Notifications.

There are a couple more classes required in the build to implement Push services:

- On the Server side `System.Net`, `System.IO` and `System.Text`.
- On the Client side `Microsoft.Phone.Notification`.

On the Server side the structure of the code is very similar to the Toast server, it is only the construction of the actual message that is different so that is all I will note here.

- Create the raw message.

```
string rawMessage = "<?xml version=\"1.0\" encoding=\"utf-8\"?>" +
    "<root>" +
    "<Value1>" + "First lot of data" + "<Value1>" +
    "<Value2>" + "Second lot of data" + "<Value2>" +
    "</root>";

byte[] notificationMessage = Encoding.Default.GetBytes(rawMessage);
sendNotificationRequest.ContentLength = notificationMessage.Length;
sendNotificationRequest.ContentType = "text/xml";
sendNotificationRequest.Headers.Add("X-NotificationClass", "3");
```

On the Client side the structure of the code is very similar to the Toast client, so I will only highlight the differences here. Note there is no binding type for Raw.

- In the initialize the push channel that is created or found section there are the following additions.

```
string channelName = "RawSampleChannel";

if (pushChannel == null)
{
    pushChannel.HttpNotificationReceived += new
        EventHandler<HttpNotificationEventArgs>(PushChannel_HttpNotificationReceived);
}
else
{
    pushChannel.HttpNotificationReceived += new
        EventHandler<HttpNotificationEventArgs>(PushChannel_HttpNotificationReceived);
}
```

- Process the raw notification.

```
void PushChannel_HttpNotificationReceived(object sender, HttpNotificationEventArgs e)
{
    string message;
    using (System.IO.StreamReader reader = new System.IO.StreamReader(e.Notification.Body))
    {
        message = reader.ReadToEnd();
    }
    Dispatcher.BeginInvoke(() =>
        MessageBox.Show(String.Format("Received Notification {0}:\n{1}",
            DateTime.Now.ToShortTimeString(), message)) );
}
```

Code Listings: The following code listings provide complete coding solutions to the Raw Notifications section.

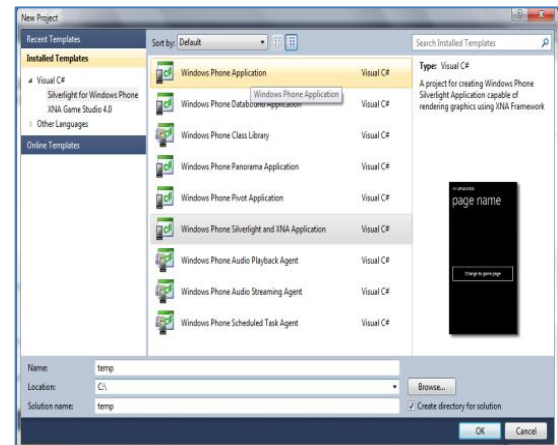
User Interaction	Notes
054 SendRaw	Server side application that sends a Raw Notification.
055 RawNotificationClient	Client side application that receives Raw Notifications when it is running.
056 MessagingGame	Work in progress (MS example), currently supported: <ul style="list-style-type: none"> • GameServer, registers the phone and sends and logs weather Toast, Tile and Raw messages. • GameClient, registers with the server and receives updated weather Toast, Tile and Raw messages.

Application Types

When developing applications in the Microsoft environment there is a large set of project templates that can be used to set up project types quickly.

The development done during this project (67 coded applications) will predominately focus on the following three project types:

- Windows Phone Application (Silverlight).
- Windows Phone Game (4.0) (XNA games).
- Windows Phone Silverlight and XNA Application (Silverlight and XNA games).



Each project type can be accessed as a 'Project Type' from the 'New Projects' menu in the Integrated Development Environment and offers a pre-made entry point for developing a particular type of application.

The following table notes the Microsoft recommendations for selecting a project type based on application requirements:

Use Silverlight if...	Use the XNA Framework if...	Combine the XNA Framework and Silverlight if...
You want a XAML-based, event-driven application framework.	You want a high-performance game framework.	You want a XAML-based, event-driven application integrated with a high-performance game framework.
You want rapid creation of a Rich Internet Application-style user interface.	You want rapid creation of multi-screen 2D and 3D games.	You want to create a multi-screen 2D or 3D game that incorporates a Rich Internet Application-style user interface.
You want to use Windows Phone controls.	You want to manage art assets such as models, meshes, sprites, textures, effects, terrains, or animations in the XNA Content Pipeline.	You want to use Windows Phone controls along with art assets such as models, meshes, sprites, textures, effects, terrains, or animations in the XNA Content Pipeline.
You want to embed video inside your application.		You want to use Silverlight text rendering, including international text, instead of sprite fonts.
You want to use an HTML web browser control.		You want the page navigation model of a Silverlight application combined with the rich graphics power of an XNA Framework application.

[Build Type Recommendations: Data from MSDN <http://msdn.microsoft.com>]

Windows Phone Application (Silverlight)

Windows Phone Applications are the build for standard phone applications and use Silverlight objects for the UI interactions.

When a user interacts with an object an event is fired and the associated event code is executed.

A Page has a standard set of events that allow actions to be performed at various stages of a page life cycle.

The following events are derived from two classes the `PhoneApplicationPage` class and the `Page` class (`System.Windows.Controls`) they define the basic flow of control through the life cycle of a phone page:

- The Constructor is used to initialize the page coding logic, it is the first thing to be accessed on start up.

```
public MainPage()
{
}
}
```

- The `OnNavigatedTo (Page)` event is next to fire. It tends to be used in preference to the `Loaded` event because it is called once each time the page is activated. It is recommended (in the documentation) to use this event for code that needs to execute with each page activation.

```
protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
}
```

- The `Loaded (PhoneApplicationPage)` event is fired when an element is added to the visual tree (display objects) this makes the event slightly unpredictable as it may be raised more than once when the page activates (or not at all depending on the build type). The event usually fires last in the load up sequence.

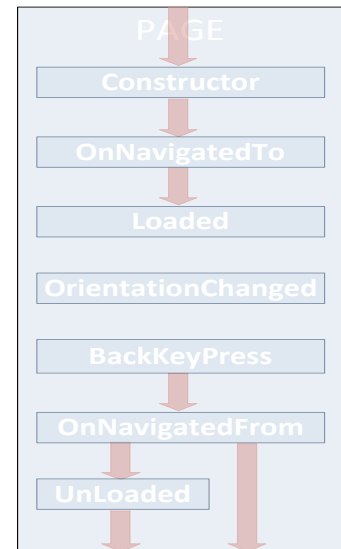
```
private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
{
}
}
```

- The `OrentationChanged (PhoneApplicationPage)` event fires when the value in the `Orientation` property is changed. Orientation aware application code is located here.

```
private void PhoneApplicationPage_OrientationChanged(object sender,
    OrientationChangedEventArgs e)
{
}
}
```

- The `BackKeyPress (PhoneApplicationPage)` event fires when the hardware 'Back' button (arrow on device front/bottom/left case) is pressed. If page exit is via a 'Back' button it will be the first event to fire on exit from the page.

```
private void PhoneApplicationPage_BackKeyPress(object sender,
    System.ComponentModel.CancelEventArgs e)
{
}
}
```



- The OnNavigatingFrom (Page) event is used to implement code that needs to execute just before a page becomes inactive. The event fires after the BackKeyPress event if a back key is pressed or first in the page deactivation flow for all other page exit paths.

```
protected override void OnNavigatingFrom(NavigatingCancelEventArgs e)
{
    base.OnNavigatingFrom(e);
}
```

- The UnLoaded (PhoneApplicationPage) event is called last in the page deactivation flow. There are problems associated with using this event as it is unpredictable. Sometimes it may never be called (depending in the shutdown state of the application), this is because the UnLoaded event will not be fired if any object in a Silverlight application shuts down (for any reason).

```
private void PhoneApplicationPage_Unloaded(object sender, RoutedEventArgs e)
{
}
```

Code Listings: The following code listing provides a complete coding solution to the Windows Phone Application (Silverlight) section.

Application Types	Notes
057 WindowsPhoneApps	<p>The example implements the following Windows Phone Application (Silverlight) options:</p> <ul style="list-style-type: none"> • App Type - Basic App Events, basic Windows Phone Application showing Page and PhoneApplicationPage events sequence. • App Type - Basic Game Events, depicts the basic flow of control through the life cycle of a phone game page, utilising a game loop system. • App Type - Basic Game Events, uses a different system to depict the basic flow of control through the life cycle of a phone game page, utilising a game update system. • Sound – Basic Sound in an application, uses the basic sound API's to play sound in a normal Application Framework. • Sound – XNA Sound in Game loop, uses the XNA sound API's to play sound in a Game Loop normal Application Framework. There are different methods of processing a sound test stream, SoundEffect, SoundEffectInstance, StreamResourceInfo, 3D sound and AudioEmitter / AudioListener are used.

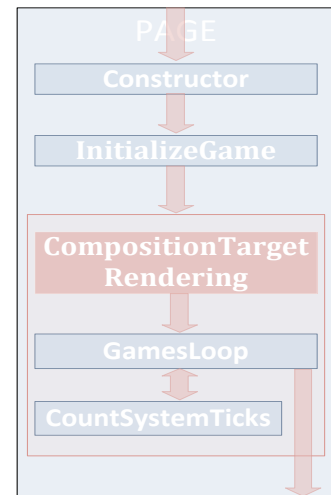
Silverlight Game Setup

The 'Windows Phone Application' build is an ideal platform for including Casual Game elements into the development of Pervasive Games.

Casual Games are simple games that can utilise Silverlight objects as game assets (puzzles, turn base, word games, picture games etc.).

The main requirement for using Silverlight as a game platform is the ability to run a games loop. A games loop is a method that is controlled by some form of timer allowing game code (and in game elements) to fire at a consistent speed.

The following events are derived from the `System.Windows.Media` class (along with the methods defined here) they depict the basic flow of control through the life cycle of a phone game page, utilising a game loop system:



- The Constructor is used to initialize the page coding logic and to call the InitializeGame method, it is the first thing to be accessed on start up.

```
public MainPage()
{
    InitializeComponent();
    InitializeGame();
}
```

- The InitializeGame method is the initialization method for the game (it is only called once). All game initialization code is set here and the main game event CompositionTarget_Rendering is defined.

```
private void InitializeGame()
{
    CompositionTarget.Rendering += new EventHandler(CompositionTarget_Rendering);
}
```

- The CompositionTarget_Rendering event is the main game event, it is called every time the Silverlight rendering process renders a frame the GamesLoop is called here.

```
private void CompositionTarget_Rendering(object sender, EventArgs e)
{
    GameLoop();
}
```

- The GamesLoop method plays each time the frame change, it runs at the fastest possible speed (frame change), then timers are set to slow down the game elements.

```
private void GameLoop()
{
    if (CountSystemTicks(interval, ref currentWaitTime)) { // Game code here }
}
```

- The CountSystemTicks method returns true if the specified interval has passed.

```
private Boolean CountSystemTicks(int frameRate, ref int lastTick)
{
    int currentTick = Environment.TickCount;
    if ((currentTick - lastTick) > frameRate)
    {
        lastTick = Environment.TickCount;
        return true;
    }
    else return false;
}
```

Code Listings: The following code listings provide complete coding solutions to the Silverlight Game Setup section.

Application Types	Notes
057 WindowsPhoneApps	<p>The example implements the following Windows Phone Application (Silverlight) options:</p> <ul style="list-style-type: none"> • App Type - Basic App Events, basic Windows Phone Application showing Page and PhoneApplicationPage events sequence. • App Type - Basic Game Events, depicts the basic flow of control through the life cycle of a phone game page, utilising a game loop system. • App Type - Basic Game Events, uses a different system to depict the basic flow of control through the life cycle of a phone game page, utilising a game update system. • Sound – Basic Sound in an application, uses the basic sound API's to play sound in a normal Application Framework. • Sound – XNA Sound in Game loop, uses the XNA sound API's to play sound in a Game Loop normal Application Framework. There are different methods of processing a sound test stream, SoundEffect, SoundEffectInstance, StreamResourceInfo, 3D sound and AudioEmitter / AudioListener are used.
058 AlarmClock	<p>Basic alarm clock (test for the sound API's and platform) with the following options:</p> <ul style="list-style-type: none"> • Page 1 displays an alarm clock. • Page 2 saves / loads any recorded sound for the alarm and sets the alarm time. • Page 3 records the sound and plays it back. <p>Very quick test of the sound and microphone API's using the Games loop and Silverlight Application frameworks. Work in progress.</p>

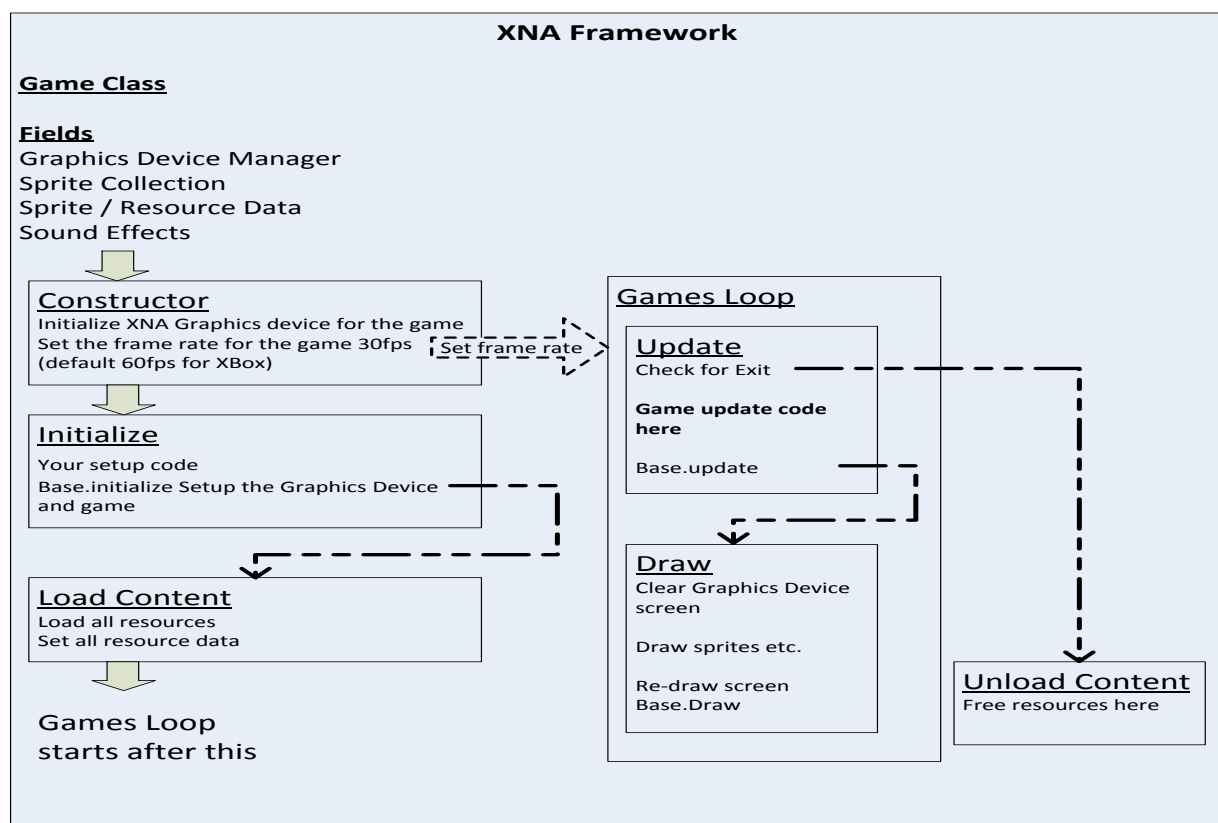
XNA Game Setup

The XNA Games Studio (4.0) > Windows Phone Game (4.0) build is the entry point into the XNA framework allowing game development on Windows Phone, PC and Xbox360.

There is a dedicated set of managed game API's that supports 2D and 3D games development, a very useful set of API's these.

Although the 3D elements of the framework are extremely interesting an investigation into this area is not in the scope of this project so I will focus on the 2D elements here (still great fun). It is assumed that there is a familiarity with the basic components of a 2D game engine, sprites (movement / collision), scrolling backgrounds, text, sound, overlays and game states, as the implementation of these concepts will be the focus of this section.

The framework (`Microsoft.Xna.Framework.Game`) is fairly straight forward (load resources, update the game domain and draw the game domain as quickly as possible) as the following diagram shows:



- The Graphics Device Manager (graphics device) and any resource (sprites / sounds) needed are set as fields.

```
GraphicsDeviceManager graphics;  
SpriteBatch spriteBatch;
```

- The Constructor is called first and is used to initialize the graphics device, set the frame rate to 30fps (default for Windows 7 Phones) for the games loop and sets up any events required.

```
public Game1()  
{  
    graphics = new GraphicsDeviceManager(this);  
  
    TargetElapsedTime = TimeSpan.FromTicks(333333);  
}
```

- The Initialize method is then called and performs any initialization needed before the game runs. It is used to load any non-graphic related content and initialize any components by a call to the base.Initialize method (it enumerates through them).

```
protected override void Initialize()
{
    // Add initialization logic here
    base.Initialize();
}
```

- The LoadContent method is called next and is used to load all of the game content and initialize any sprite batches (sets of sprite images). On exit from this method the game loop will begin its loop cycle.

```
protected override void LoadContent()
{
    spriteBatch = new SpriteBatch(GraphicsDevice);

    // Use this.Content to load game content here.
}
```

- The Games Loop is the guts of the game, it is where the game is constantly updated and drawn to the screen (the Update method is called then the Draw method is called). It will run at the frame rate set in the constructor until some in game exit condition is reached.

- The Update method contains and updates the game logic / calculations. World updates, collision checking, processing input and audio play are implemented here.

```
protected override void Update(GameTime gameTime)
{
    if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
        this.Exit();

    // Update logic here

    base.Update(gameTime);
}
```

- The Draw method is called to draw the required game images (sprites / backgrounds) to the screen.

```
protected override void Draw(GameTime gameTime)
{
    GraphicsDevice.Clear(Color.CornflowerBlue);

    // Drawing code here

    base.Draw(gameTime);
}
```

- The UnloadContent method is called when some in game exit condition is reached. All clean up logic and code need to be implemented here.

```
protected override void UnloadContent()
{
    // Unload any non ContentManager content here.
}
```

Code Listings: The following code listing provides a complete coding solution to the XNA Game Setup section.

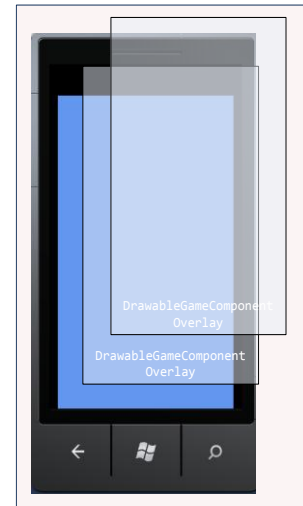
Application Types	Notes
059 XNAGameSetup	Basic template for the XNA game framework.

XNA Game Setup Overlays

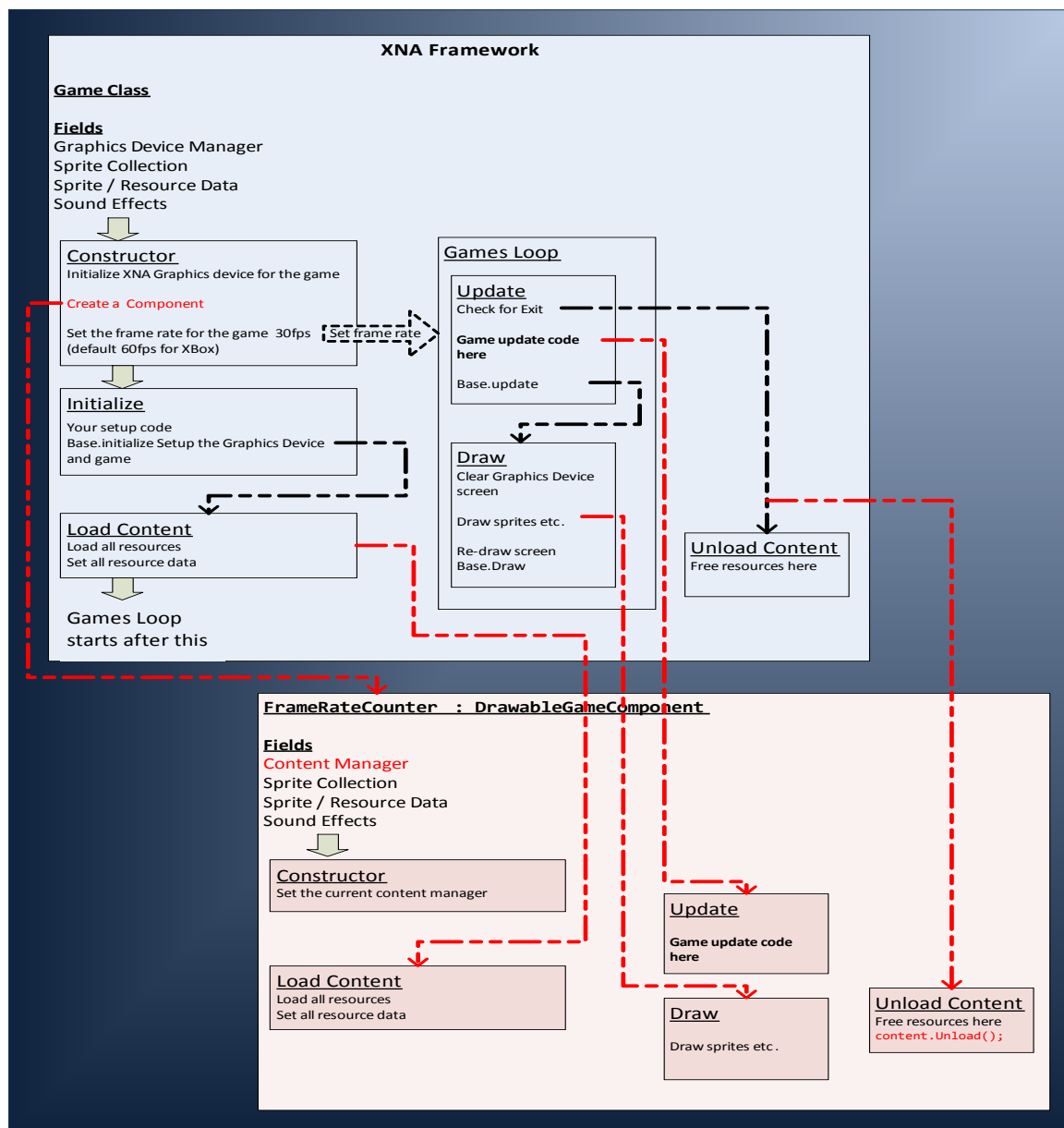
There is a feature of the of the XNX framework that allows game components to be added in a modular way. One of the interesting uses of this mechanism is adding drawing layers (useful for menus, depth effects or to keep the game logic modular / simple) via the use of the `DrawableGameComponent` class.

To use an overlay, add a new Game Component to the build. Then edit the code so that the new class is derived from the `DrawableGameComponent` class and the following methods are overridden (`GameComponent.Initialize` optional)

`GameComponent.LoadContent`, `GameComponent.Update`, `DrawableGameComponent.Draw` and `DrawableGameComponent.UnloadContent`. Game logic and rendering code can be added to each new layer and the layer will be updated at the frame rate set in the game constructor. A new `DrawableGameComponent` needs to be registered in the game constructor via a call to the `Game.Components.Add()` method. All components set up as overlays are displayed in the ascending order they are set up in.



Adding overlays to the framework structure is fairly straight forward as the following diagram shows:



- The Game Constructor needs to register a new DrawableGameComponent via a call to the Game.Components.Add() method.

```
public Game1()
{
    graphics = new GraphicsDeviceManager(this);
    Components.Add(new FrameRateCounter(this));
    TargetElapsedTime = TimeSpan.FromTicks(333333);
}
```

- The class is derived from the DrawableGameComponent class.

```
public class FrameRateCounter : DrawableGameComponent {}
```

- The Content Manager (runtime component that manages objects) and any resource (sprites / sounds) needed are set as fields.

```
ContentManager content;
SpriteBatch spriteBatch;
```

- The Constructor is called first and is used to set the current content manager.

```
public FrameRateCounter(Game game) : base(game)
{
    content = game.Content;
}
```

- The LoadContent method is called after the Game. LoadContent method fires and is used to load all of the overlay content and initialize any sprite batches (sets of sprite images).

```
protected override void LoadContent()
{
    spriteBatch = new SpriteBatch(GraphicsDevice);
}
```

- The Update method is called after the initial part of the Game.Update method is called then the base.Update methods (called in the Game.Update method) completes the update. The overlay game logic / calculations (world updates, collision checking, processing input and audio play) are implemented here.

```
public override void Update(GameTime gameTime) { // Do any game coding here }
```

- The Draw method is called after the initial part of the Game.Draw method is called then the base.Draw methods (called in the Game.Draw method) completes the draw cycle. The Draw method is called to draw the required game images (sprites / backgrounds) for the overlay to the screen.

```
public override void Draw(GameTime gameTime) { // Draw the sprites and/or text }
```

- The UnloadContent method is called when some in game exit condition is reached. All clean up logic and code need to be implemented here. Each overlay is unloaded in reverse order (by calling the content.Unload method) it was created, with the game Unload firing last in the sequence.

```
protected override void UnloadContent()
{
    content.Unload();
}
```

Code Listings: The following code listing provides a complete coding solution to the XNA Game Setup Overlays section.

Application Types	Notes
060 XNAGameSetupOverlays	Basic template for the XNA game framework with an overlay added.

Silverlight and XNA Application

The final build option is the Silverlight and XNA Application project type (Silverlight for Windows Phone > Windows Phone Silverlight and XNA Application or XNA Game studio (4.0) > Windows Phone Silverlight and XNA Application). This is a new feature of the 7.1 OS release which offers a main page (Silverlight) navigating to a game page with both Silverlight and XNA content, that is a Silverlight page that can render XNA content.

The build is a combination of the following three project types:

- Main Application Project, which contains the Silverlight (MainPage) and Silverlight / XNA (GamePage) content. The App.xaml/.cs offers access to a GameTimer, ContentManager and SharedGraphicsDeviceManager, which allows Silverlight and XNA to run in the same application.
- XNA Library Project, which is the bridge for the Silverlight and XNA build systems.
- XNA Library Content Project, used as the repository for the XNA resources.

The framework is fairly straight forward as the following diagram shows, a Silverlight main page is used as an application start page and then the game page (Silverlight / XNA) is accessed via a start button:



- The ContentManager, GameTimer and any resource (sprites / sounds) needed are set as fields.

```
ContentManager contentManager;  
GameTimer timer;  
SpriteBatch spriteBatch;
```

- The Constructor is called first and is used to set the content manager of the page to the content manager of the application and also set a game timer (30fps).

```
public GamePage()
{
    InitializeComponent();
    contentManager = (Application.Current as App).Content;
    timer = new GameTimer();
    timer.UpdateInterval = TimeSpan.FromTicks(333333);
    timer.Update += OnUpdate;
    timer.Draw += OnDraw;
}
```

- The onNavigatedTo method is then called (and each time the page is navigated to). The sharing mode of the graphics device is set to true to enable XNA rendering, a new SpriteBatch is created so textures can be drawn, game content is loaded and the game timer is started.

```
protected override void OnNavigatedTo(NavigationEventArgs e)
{
    SharedGraphicsDeviceManager.Current.GraphicsDevice.SetSharingMode(true);
    spriteBatch = new SpriteBatch(SharedGraphicsDeviceManager.Current.GraphicsDevice);
    // this.content used to load game content here
    timer.Start();
    base.OnNavigatedTo(e);
}
```

- The Games Loop in this framework consists of the OnUpdate and OnDraw methods, it is the game engine where the game is constantly updated and drawn to the screen (the OnUpdate method is called then the OnDraw method is called). It will run at the frame rate set in the constructor until some in game exit condition is reached.

- The OnUpdate method (handles the Timer.Update event) contains and updates the game logic / calculations. World updates, collision checking, processing input and audio play are implemented here.

```
private void OnUpdate(object sender, GameTimerEventArgs e) { // Add update logic here }
```

- The OnDraw method (handles the Timer.Draw event) is called to draw the required game images (sprites / backgrounds) to the screen.

```
private void OnDraw(object sender, GameTimerEventArgs e)
{
    SharedGraphicsDeviceManager.Current.GraphicsDevice.Clear(Color.CornflowerBlue);
    // Add drawing code here.
}
```

- The onNavigatedFrom method is called each time the page is navigated from. The sharing mode is set to false turning off XNA rendering, the timer is stopped and all clean up logic and code also needs to be implemented here.

```
protected override void OnNavigatedFrom(NavigationEventArgs e)
{
    timer.Stop();
    SharedGraphicsDeviceManager.Current.GraphicsDevice.SetSharingMode(false);
    base.OnNavigatedFrom(e);
}
```

Code Listings: The following code listing provides a complete coding solution to the Silverlight and XNA Application section.

Application Types	Notes
061 SilverlightXNAApplication	Basic template for the Silverlight and XNA Application framework.

Problems with the Silverlight and XNA Application Framework

This is a relatively new feature of the IDE (7.1 release) so it will invariably have some teething problems.

The build is solid and the framework works very well, but there is an irritating warning that is generated when every application is built *“Warning 1: The project ‘ProjectNameLib’ cannot be referenced. The referenced project is targeted to a different framework family (.NETFramework)”*.

On face value it looks quite serious as ProjectNameLib is one of the build projects, but after some extensive investigation (Microsoft do not seem to have heard of the problem) it seems to be a throw away warning endemic to the current release. It does not stop the build or affect any executable in any way so it can be ignored.

The explanation for this issue is a point of discussion on several developers’ forums with the following source giving the most succinct explanation *“A Silverlight and XNA Application include a Silverlight project, an XNA Game Studio Windows Phone Game Library project and an XNA Game Studio content project. The Silverlight project is the parent project, and it includes a project-to-project reference to the XNA game library project. The warning occurs because Visual Studio detects that the XNA project does not target the same .NET Framework profile as the Silverlight project. However, on Windows Phone, there is only one .NET Framework profile, regardless of what the target framework settings are in the project files, so the warning does not indicate any possible functional problems when you build and run your application.”* [Thanks to : <http://blogs.msdn.com/b/astebner/> : Aaron Stebner's WebLog : for the solution to this one].

Common XNA Game Elements

There are several useful XNA game elements that can be used to blur the distinction between real world and virtual world game play in the pervasive game model. Sprites, backgrounds (static or scrolling), text, sensor input, gesture input, sound and overlays, are the main ingredients in the pervasive mix. Some of these topics have already been looked at in previous sections, so I will focus on manipulating images via the format of sprites and backgrounds in this section.

Sprites

Sprites form the basis of most 2D games, sprites (rectangular images) are blitted to the screen and moved about to produce the effect of responsive in game visual elements. XNA supports several mechanisms to manipulate sprites as 2D bitmaps that are rendered directly to a target location (graphics buffer then) on screen at an X / Y coordinate.

Sprites have the following characteristics:

- Textures, Texture2D bitmaps are used for sprites, the default is to draw the whole texture. If only a portion of the texture is required a bounding rectangle can be passed via the sourceRectangle parameter to the `spriteBatch.Draw` method. There is a maximum texture size of 2048 bits per texture, so this needs to be considered when dealing with large background images and animation arrays sheets.

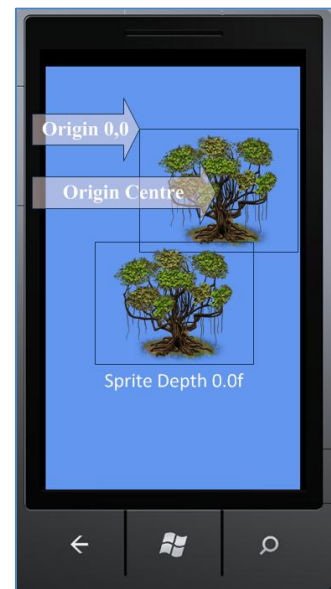
```
Rectangle spritRect;  
spritRect = new Rectangle( 0, 0, 16, 16);  
spriteBatch.Draw(SpriteTexture, screenpos, spritRect,  
                Color.White, RotationAngle,  
                origin, 1.0f,  
                SpriteEffects.None, 0.0f);
```

- Sprite Origin, is the blit position of the sprite, the default is the top / left (0,0). The centre or any other position in the sprite can be set as a Vector2 coordinate pair which needs to be passed as a (origin) parameter to the `spriteBatch.Draw` method.

```
private Vector2 origin;  
origin.X = SpriteTexture.Width / 2;  
origin.Y = SpriteTexture.Height / 2;  
spriteBatch.Draw(SpriteTexture, screenpos,  
                null, Color.White,  
                RotationAngle, origin,  
                1.0f, SpriteEffects.None, 0.0f);
```

- Sprite Depth, is the blit order of the sprite, when set to 0.0f sprites are drawn on top, the draw order descends down until a setting of 1.0f. Sprites with the same depth value or none will be drawn in the sequence they are called in the sprite batch sequence. The depth value needs to be passed as a (layerDepth) parameter to the `spriteBatch.Draw` method.

```
spriteBatch.Draw(SpriteTexture, screenpos,  
                null, Color.White,  
                RotationAngle, origin,  
                1.0f, SpriteEffects.None, 0.0f);
```



Drawing a Sprite

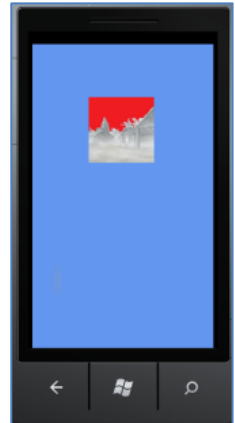
The process of drawing a sprite is, define a `SpriteBatch` object, `Texture2D` object and `Vector2` object, load a bitmap texture and then display it at a location via the sprite batch mechanism.

The following syntax describes the process of drawing a sprite to the screen:

- Create a `SpriteBatch` object, `Texture2D` object and `Vector2` object. The bitmap data for a sprite is defined in a `Texture2D` object, transparency or semi-transparency can be set on the alpha channel. The position of the sprite is defined as a 2D vector.
`SpriteBatch spriteBatch;`
`Texture2D spriteTexture1;`
`Vector2 spritePosition1;`
- Initialize the `SpriteBatch` objects, load the `Texture2D` texture and set the sprite start position.

```
protected override void LoadContent()
{
    spriteBatch = new SpriteBatch(GraphicsDevice);
    spriteTexture1 = Content.Load<Texture2D>("SpriteImage");
    spritePosition1.X = 0;
    spritePosition1.Y = 0;
}
```
- Begin a sprite batch operation, add a sprite to a batch to be rendered and flush the sprite batch (restored the device state to how it was before `Begin` was called).

```
protected override void Draw(GameTime gameTime)
{
    GraphicsDevice.Clear(Color.CornflowerBlue);
    spriteBatch.Begin();
    spriteBatch.Draw(spriteTexture1, spritePosition1, Color.White);
    spriteBatch.End();
    base.Draw(gameTime);
}
```



Drawing a Transparent Sprite

Transparent background pixels in a bitmap can be blitted in the correct format (transparent) by using the correct setting in the `spriteBatch` operation (default or `BlendState.AlphaBlend` set).

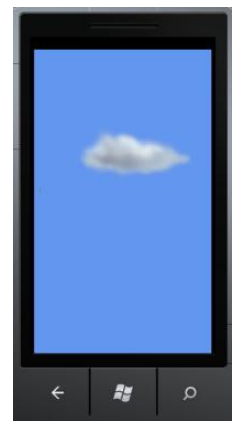
More complex effects can also be achieved by using `blendState` and colour tint options in the `spriteBatch` process.

The following syntax describes the process of drawing a sprite with a transparent background to the screen (both versions produce the same result):

- Draw a sprite with a transparent background.

```
protected override void Draw(GameTime gameTime)
{
    spriteBatch.Begin();
    spriteBatch.Draw(spriteTexture, spritePosition, Color.White);
    spriteBatch.End();
}
```
- Draw a sprite with a transparent background.

```
protected override void Draw(GameTime gameTime)
{
    spriteBatch.Begin(SpriteSortMode.BackToFront, BlendState.AlphaBlend);
    spriteBatch.Draw(spriteTexture, spritePosition, Color.White);
    spriteBatch.End();
}
```

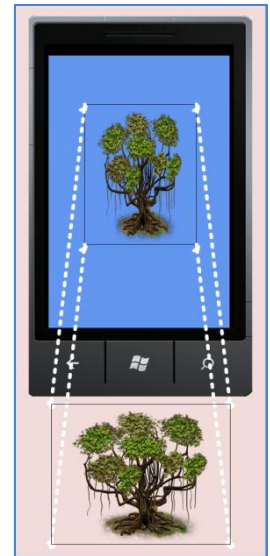


Scaling a Sprite

Scaling can be used to change the size of the sprite on screen in relation to its texture size (as the diagram shows).

In a gaming context scaling a sprite can be used to give an illusion of depth to a 2D domain by adding a fake sense of perspective. This is achieved by increasing the size of the sprite the further out of the perspective of the domain it moves (towards the user) and decreasing the size of the sprite as it moves into the perspective of the domain (recedes into the background).

There are three modes of scaling offered, uniform (floating-point number multiplying the X and Y axes), none-uniform (Vector2 used as the multiplier) and a sourceRectangle / destinationRectangle rectangle (rectangular areas to scale to and from). Values are passed to the `spriteBatch.Draw` method via the scale (float or Vector2) or sourceRectangle / destinationRectangle parameters depending on which scaling method is required.



The following syntax describes the process of scaling a sprite (to 12th the screen dimensions) using the sourceRectangle / destinationRectangle rectangle method (probably the most flexible):

- Create a scaling factor rectangle.
`Rectangle scalingFactorRectangle;`
- Define the scaling rectangle.

```
protected override void LoadContent()
{
    scalingFactorRectangle = new Rectangle((int)(spritePosition3.X),
                                           (int)(spritePosition3.Y),
                                           GraphicsDevice.Viewport.Width / 12,
                                           GraphicsDevice.Viewport.Height / 12);
}
```
- Begin a sprite batch operation, add a sprite to a batch to be rendered and flush the sprite batch (restored the device state to how it was before Begin was called).

```
protected override void Draw(GameTime gameTime)
{
    spriteBatch.Begin(SpriteSortMode.BackToFront, BlendState.AlphaBlend);
    spriteBatch.Draw(spriteTexture, scalingFactorRectangle, Color.White);
    spriteBatch.End();
}
```

The following syntax is used for passing a parameter to the `spriteBatch.Draw` method when scaling a sprite using the uniform and none-uniform methods:

- Call using the uniform method with a float scaling value.

```
spriteBatch.Draw(spriteTexture, screenpos,
                  spritRect, Color.White,
                  RotationAngle, origin,
                  1.0f, SpriteEffects.None, 0.0f);
```
- Call using none-uniform method with a Vector2 scaling value.

```
spriteBatch.Draw(spriteTexture, screenpos,
                  spritRect, Color.White,
                  RotationAngle, origin,
                  scailVector, SpriteEffects.None, 0.0f);
```

Rotating a Sprite

A sprite can be rotated around its origin to give the appearance of turning or spinning.

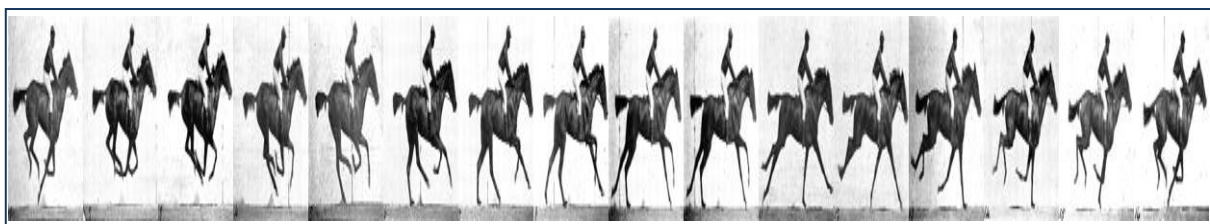
The following syntax describes the process of rotating a sprite around its centre:

- Create an origin Vector2 and a rotation angle (float).
`Vector2 origin;`
`float rotationAngle;`
- Calculate the rotation as the centre of the sprite and angle to rotate.
`protected override void LoadContent()`
{
 `origin.X = spriteTexture.Width / 2;`
 `origin.Y = spriteTexture.Height / 2;`
 `rotationAngle = 0;`
}
- Calculate the new rotation position based on the time between Update calls.
`protected override void Update(GameTime gameTime)`
{
 `float elapsed = (float)gameTime.ElapsedGameTime.TotalSeconds;`
 `rotationAngle += elapsed;`
 `float circle = MathHelper.Pi * 2;`
 `rotationAngle = rotationAngle % circle;`
}
- Begin a sprite batch operation, add a sprite to a batch to be rendered and flush the sprite batch (restored the device state to how it was before Begin was called).
`protected override void Draw(GameTime gameTime)`
{
 `spriteBatch.Begin(SpriteSortMode.BackToFront, BlendState.Opaque);`
 `spriteBatch.Draw(spriteTexture, spritePosition, null, Color.White, rotationAngle, origin,`
 `1.0f, SpriteEffects.None, 0f);`
 `spriteBatch.End();`
}



Animated Sprites

The basic process of animating a sprite is to display successive images to the screen at a speed that fools the eye into believing that the image is moving, in the XNA framework this involves looping through a series of cells in a bitmap texture.



[Galloping horse, animated using photos by Eadweard Muybridge]

A frame rate of 24fps to 30fps is required to produce a seamless animation, the default speed of the XNA framework is 30 frames per second, so no problem there.

Consideration also needs to be given to the number of cells in the sequence and the distance travelled (or not) across the screen between each call to a cell. Getting the animated sequences correct seems to be a bit of a black art (probably best left to the more creative types) as the actual animation image will dictate these variables reliant on the type, size and speed of the image.

The most intuitive description I have of the process is from my daughter who is currently doing animation in her art class *“A bullet moves faster (and will have less cells) than a flower swaying in the wind, characters need to appear to keep in step when they are walking, running or doing something and an image of say a bird should fly in relation to the animation cells of wings flapping”* [Nadia Morris, art student, 2012].

The recommendations in the XNA literature note the following key factors to consider when attempting to produce a realistic sprite animation sequence:

- Use the highest resolution possible for the images.
- The quality of the graphics needs to be consistent across all the visual elements of the game.
- A valid sequence of images is required for every direction that the sprite image will move in, with a sufficient number of animation cells in the sequence to allow any action to be viewed seamlessly.
- A realistic transition between states is required, the sprite image should appear to smoothly change between each action or movement in the animation sequence. The use of Key Frames (transition points in the animation sequence) is recommended.

The following syntax describes the process of drawing an animated sprite to the screen:

- Create a rectangle for the animation cell location.

```
Rectangle sourceRectangle;
int numberOfCells = 10;
```
- Set up the animation cell size and initial location in the source rectangle.

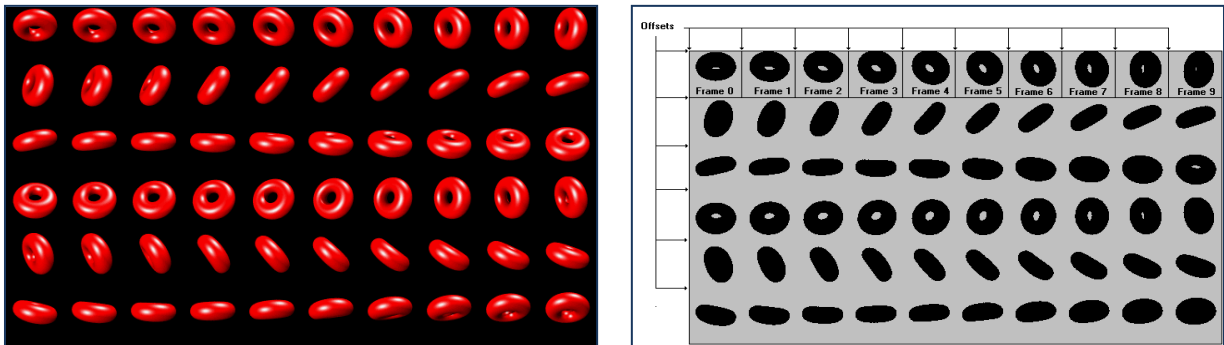
```
protected override void LoadContent()
{
    sourceRectangle.Width = spriteTexture.Bounds.Width / numberOfCells;
    sourceRectangle.Height = spriteTexture.Bounds.Height;
    sourceRectangle.X = 0;
    sourceRectangle.Y = 0;
}
```
- Calculate cell location and increment through the cells, reset when the end of the cell is reached.

```
protected override void Update(GameTime gameTime)
{
    sourceRectangle.X = sourceRectangle.X + sourceRectangle.Width;
    if (sourceRectangle.X >= (sourceRectangle.Width * numberOfCells))
    {
        sourceRectangle.X = 0;
    }
}
```
- Begin a sprite batch operation, add a sprite to a batch to be rendered and flush the sprite batch (restored the device state to how it was before Begin was called).

```
protected override void Draw(GameTime gameTime)
{
    spriteBatch.Begin(SpriteSortMode.BackToFront, BlendState.AlphaBlend);
    spriteBatch.Draw(spriteTexture, spritePosition,
        sourceRectangle, Microsoft.Xna.Framework.Color.White,
        0f, Vector2.Zero, 1.0f,
        SpriteEffects.None, 0f);
    spriteBatch.End()
}
```


Complex Animations

More complex animation sequences can be achieved by using more animation cells, this requires some organization of the cells into a standard format.



The tradition method used, is to store each sequence of animations sequentially in a file (in the order they will be use). This allows an offset to be calculated for each animation frame in the file and then an algorithm can be implemented to iterate through each frame in the animation sequence.

Images can also be set up in rows that represent a specific action (or movement sequence) in a particular direction. Key Frames (bridging frames) can also be defined, which represent the neutral or transition states between animation sequences.

The following syntax describes the process of drawing an animated sprite to the screen using a cell array:

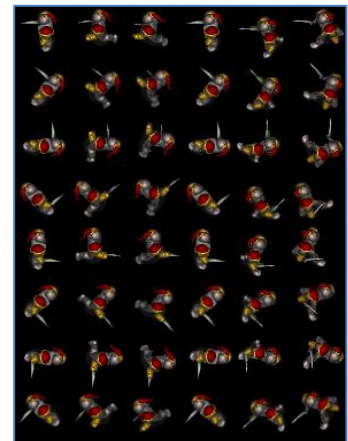
- Set the number of cells in the sequence and the number of sequences.

```
int numberOfCells = 6;
int numberOfSequences = 8;
```
- Set up the animation cell size and initial source location rectangle.

```
protected override void LoadContent()
{
    sourceRectangle.Width = spriteTexture.Bounds.Width / numberOfCells;
    sourceRectangle.Height = spriteTexture.Bounds.Height / numberOfSequences;
    sourceRectangle.X = 0;
    sourceRectangle.Y = 0;
}
```
- Calculate cell location and increment through the cells, reset when the end is reached.

```
protected override void Update(GameTime gameTime)
{
    sourceRectangle.X = sourceRectangle.X + sourceRectangle.Width;
    if (sourceRectangle.X >= (sourceRectangle.Width * numberOfCells))
    {
        sourceRectangle.X = 0;
        sourceRectangle.Y = sourceRectangle.Y + sourceRectangle.Height;
    }
    if (sourceRectangle.Y >= (sourceRectangle.Height * numberOfSequences)) sourceRectangle.Y=0;
}
```
- Begin a sprite batch operation, add sprite to batch to be rendered and flush the sprite batch

```
protected override void Draw(GameTime gameTime)
{
    spriteBatch.Begin(SpriteSortMode.BackToFront, BlendState.AlphaBlend);
    spriteBatch.Draw(spriteTexture, spritePosition, sourceRectangle,
        Microsoft.Xna.Framework.Color.White, 0f, Vector2.Zero, 1.0f, SpriteEffects.None, 0f);
    spriteBatch.End();
}
```



Sprite Font

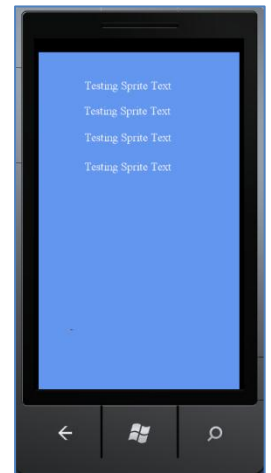
The spriteBatch mechanism can also be used to display text (sprite font).

The DrawString method is used to draw text onto the screen at a designated location. Colour, rotation, origin, and scaling are all supported when displaying Sprite Font.

To use the Draw string method a SpriteFont needs to be imported into a project. This is done via the 'New Items' list and selecting a SpriteFont.

When naming the font file it is recommended to use a 'friendly name' to identify the font as a recognised type (Times New Roman, Courier New etc.) for later ease of use.

The Sprite Font XML file is generated automatically and it is then just a matter of changing the values in the tags, to set up the font type.



The following table is a reference of the Sprite Font XML Schema:

Tag name	Content type	Content description
<FontName>	string	The name of the font to be imported. This is not the name of a font file, but rather the friendly name that identifies the font once it is installed on your computer.
<Size>	float	The point size of the font to be imported.
<Spacing>	float	The number of pixels to add between each character when the string is rendered.
<UseKerning>	Boolean	Specifies whether to use kerning information when rendering the font. Default value is true.
<Style>	Regular, Bold, Italic, or Bold Italic	The style of the font to be imported.
<DefaultCharacter>	char	The Unicode character to substitute any time an attempt is made to render characters that are not in the font. Specifying this element is optional.
<CharacterRegions>	One or more <CharacterRegion> tags	One or more numerical ranges indicating which subset of Unicode characters to import.
<CharacterRegion>	One <Start> and one <End> tag	The beginning and ending of a region of Unicode characters.
<Start>	char	The first Unicode character to include in a <CharacterRegion>.
<End>	char	The last Unicode character to include in a <CharacterRegion>.

[Sprite Font XML Schema Reference: Data from MSDN <http://msdn.microsoft.com>]

Not all font types are currently supported as Sprite Fonts for Windows 7 phones, but a default standard set will be available for all devices.

The following table notes the default Sprite Fonts supported for XNA on Windows 7 Phones:

Font Name	Languages	Regular	Bold	Italic	OS 7.0	OS 7.1
Arial	Latin, Cyrillic, and Greek-based languages	yes	yes	yes	yes	yes
Arial Black	Latin, Cyrillic, and Greek-based languages	yes	no	no	yes	yes
Calibri	Latin, Cyrillic, and Greek-based languages	yes	yes	no	yes	yes
Comic Sans MS	Latin, Cyrillic, and Greek-based languages	yes	yes	no	yes	yes
Georgia	Latin, Cyrillic, and Greek-based languages	yes	yes	yes	yes	yes
Lucida Sans Unicode	Latin, Cyrillic, and Greek-based languages	yes	no	no	yes	yes
Malgun Gothic	Korean	yes	no	no	yes	yes
Meiryo UI	Japanese	yes	no	no	yes	yes
Microsoft YaHei	Chinese (Traditional), Chinese (Simplified)	yes	no	no	yes	yes
Segoe UI	Latin, Cyrillic, and Greek-based languages	yes	yes	no	yes	yes
Segoe UI Symbol	Supports various Unicode symbols	yes	no	no	no	yes
SimSun	Chinese (Simplified) only Chinese devices	yes	no	no	no	yes
Tahoma	Latin, Cyrillic, and Greek-based languages	yes	yes	no	yes	yes
Times New Roman	Latin, Cyrillic, and Greek-based languages	yes	yes	yes	yes	yes
Trebuchet MS	Latin, Cyrillic, and Greek-based languages	yes	yes	yes	yes	yes
Verdana	Latin, Cyrillic, and Greek-based languages	yes	yes	yes	yes	yes

[Supported Sprite Font: Data from MSDN <http://msdn.microsoft.com>]

It is then just a matter of using the font it a similar way to a Texture2D.

The following syntax describes the process of drawing a Sprite Font to the screen:

- In the XML Sprite Font file update the FontName Tag to the type of font required.
`<FontName>Times New Roman</FontName>`
- Create a SpriteFont object and Vector2 object.

```
SpriteFont FontXNA;
Vector2 fontPosition;
string outputText;
```
- Load the Sprite Font and set the text start position.

```
protected override void LoadContent()
{
    FontXNA = Content.Load<SpriteFont>("Times New Roman");
    fontPosition.X = 0;
    fontPosition.Y = 0;
    outputText = "Testing Sprite Text";
}
```

- Begin a sprite batch operation, add a sprite to a batch to be rendered and flush the sprite batch (restored the device state to how it was before Begin was called).
protected override void Draw(GameTime gameTime)
{
 spriteBatch.Begin();
 spriteBatch.DrawString(FontXNA, outputText, fontPosition, Color.White);
 spriteBatch.End();
}

Backgrounds

Backgrounds in XNA are basically the first sprite to be displayed in the spriteBatch cycle. This allows the background to be used to clear the screen each time it is refreshed.

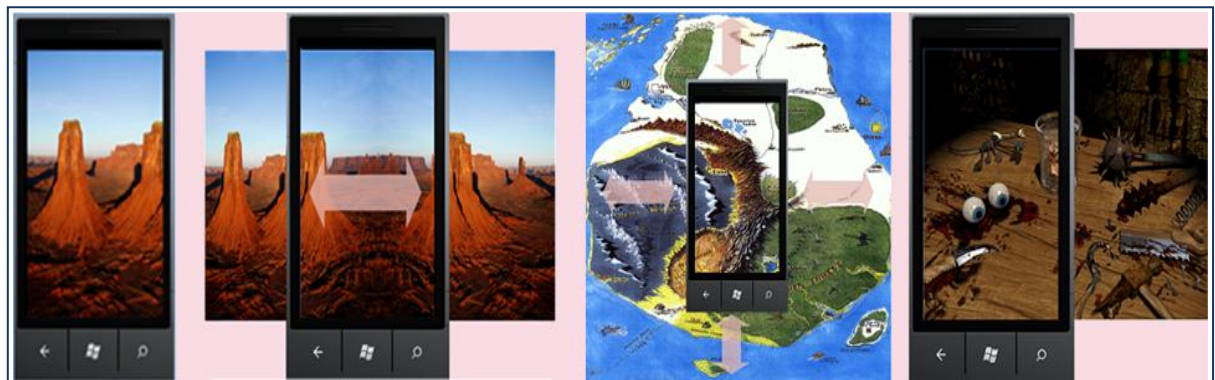
Depending on the type of background and the orientation of the device the size of a background can vary from the screen size (remembering orientation variations) to a maximum size of 2048 * 2048 pixels.

The following restrictions limit the use of backgrounds and relate to the configuration of the Texture2D:

- The maximum texture size allowed is 2048 * 2048 pixels.
- The maximum texture aspect ratio allowed is 2048.
- If mipmapped texture are used the size of the texture must be in a power of two.

The configuration (coding logic) of a background depends on the type of game being developed, the orientation setup for the game play and if the background is static or moving (scrolling).

Different graphical perspectives are also used to focus the game play to top-down, face-on or isometric views.



The technique used is to create a large Texture2D and selectively blit parts of it to the screen.

The following syntax describes the process of drawing a basic scrolling background to the screen:

- Create a Texture2D object and blitting rectangles.
Texture2D backgroundTexture;
Rectangle backgroundRectangle;
Rectangle backgroundSourceRect;
Rectangle clipScreen;
Rectangle clipSource;

- Load the Texture2D texture and set the background start positions.

```
protected override void LoadContent()
{
    backgroundTexture = Content.Load<Texture2D>("backgroundImage");
    backgroundRectangle.X = 0;
    backgroundRectangle.Y = 0;
    backgroundRectangle.Width = GraphicsDevice.Viewport.Width;
    backgroundRectangle.Height = GraphicsDevice.Viewport.Height;
    backgroundSourceRect.X = 0;
    backgroundSourceRect.Y = 0;
    backgroundSourceRect.Width = GraphicsDevice.Viewport.Width;
    backgroundSourceRect.Height = GraphicsDevice.Viewport.Height;
    clipScreen.X = 0;
    clipScreen.Y = 0;
    clipScreen.Width = GraphicsDevice.Viewport.Width;
    clipScreen.Height = GraphicsDevice.Viewport.Height;
    clipSource.X = 0;
    clipSource.Y = 0;
    clipSource.Width = GraphicsDevice.Viewport.Width;
    clipSource.Height = GraphicsDevice.Viewport.Height;
}
```

- Calculate the current position in the scroll, reset when the end is reached.

```
protected override void Update(GameTime gameTime)
{
    backgroundSourceRect.X = backgroundSourceRect.X + 4;
    if (backgroundSourceRect.X >= backgroundTexture.Width) backgroundSourceRect.X = 0;
}
```

- Begin a sprite batch operation, with two versions depending if the blit is over the texture.

```
protected override void Draw(GameTime gameTime)
{
    if ((backgroundSourceRect.X + backgroundSourceRect.Width) > backgroundTexture.Width)
    {
        int StartGapWidth = ((backgroundSourceRect.X + backgroundSourceRect.Width) -
                             backgroundTexture.Width);
        int endGapWidth = backgroundSourceRect.Width - StartGapWidth;

        clipScreen.X = 0;
        clipScreen.Width = endGapWidth;
        clipSource.X = backgroundSourceRect.X;
        clipSource.Width = endGapWidth;
        spriteBatch.Begin(SpriteSortMode.BackToFront, BlendState.AlphaBlend);
        spriteBatch.Draw(backgroundTexture, clipScreen, clipSource, Color.White);
        spriteBatch.End();

        clipScreen.X = endGapWidth;
        clipScreen.Width = StartGapWidth;
        clipSource.X = 0;
        clipSource.Width = StartGapWidth;
        spriteBatch.Begin(SpriteSortMode.BackToFront, BlendState.AlphaBlend);
        spriteBatch.Draw(backgroundTexture, clipScreen, clipSource, Color.White);
        spriteBatch.End();
    }
    else
    {
        spriteBatch.Begin(SpriteSortMode.BackToFront, BlendState.AlphaBlend);
        spriteBatch.Draw(backgroundTexture, backgroundRectangle, backgroundSourceRect,
            Color.White);
        spriteBatch.End();
    }
}
```

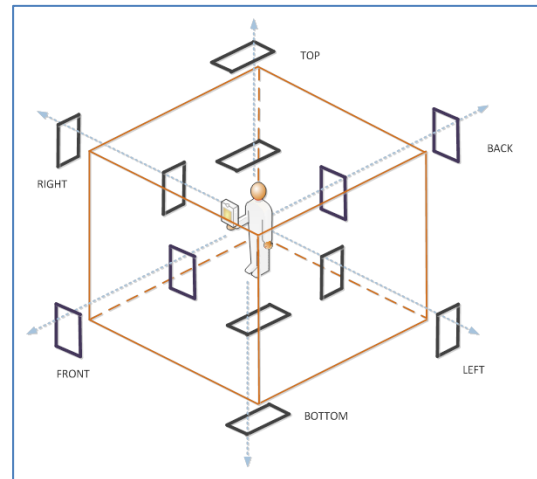
Code Listings: The following code listings provide complete coding solutions to the Common XNA Game Elements section.

Common XNA Game Elements	Notes
062 SpriteTest	<p>The test application implements the following sprite operations:</p> <p>Sprite 1: Draw opaque sprite test, a sprite is displayed to the screen.</p> <p>Sprite 2: Draw transparent sprite test, a sprite is displayed to the screen and the background transparency is displayed correctly.</p> <p>Sprite 3: Draw scaling sprite test, a sprite is displayed to the screen with its size set by a scaling rectangle.</p> <p>Sprite 4: Draw rotating sprite test, a sprite is displayed to the screen and rotated around its centre (sprite origin centre).</p> <p>Sprite 5: Draw animated sprite test, two sprites are displayed animated to the screen one transparent and the other opaque.</p> <p>Sprite 6: Draw complex animated sprite test, five sprites are displayed animated to the screen in different formats (from cell array textures).</p> <p>Sprite Font 7: Draw Sprite Font test, a Times New Roman Sprite Font is displayed to the screen.</p> <p>Sprite Background 8: Draw background test, a scrolling background is displayed to the screen from a texture 1979 * 792 (stretches).</p>
063 XNATest1	<p>Quick test to blit two sprites and move them, the sprites collide against the screen and each other, text is also displayed moving and as an overlay layer.</p>
064 XNATest2	<p>The test application displays a sprite to the screen moving relative to the accelerometer, the movement types are:</p> <p>Movement 1: Slides and with some acceleration speed increase (0.3f) and slows down by a friction (0.99f) factor.</p> <p>Movement 2: Slides and with some acceleration, the movement is set up to be natural when the device is tilted on a Y-axis setting of 0.3f.</p> <p>Movement 3: Exponential movement travels further exponentially.</p> <p>Movement 4: Exponential movement, travels further exponentially, the movement is set up to be natural when the device is tilted on a Y-axis setting of 0.3f.</p> <p>Movement 5: Only moves when tilted, remains static at optimal viewing angles.</p> <p>Flick 6: Select the sprite and then flick, movement based on time and distance between flick gestures.</p>
065 Piano	<p>Sound test, basic piano application, quick XNA sound on click test.</p>

A Pervasive System

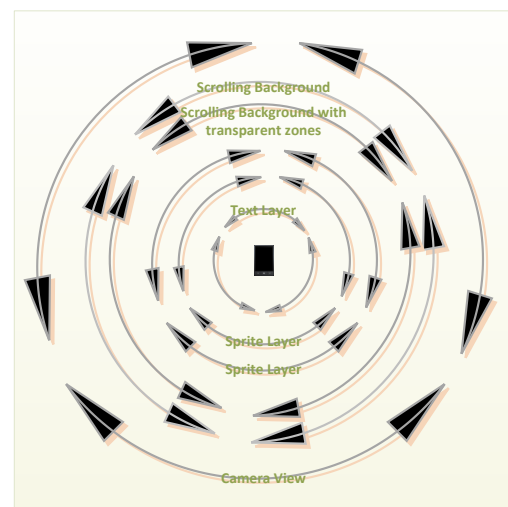
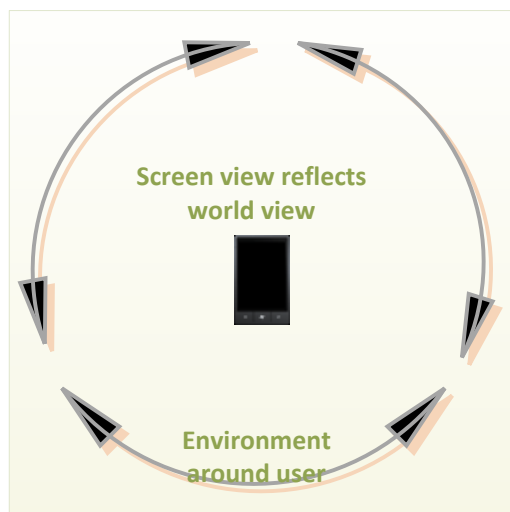
Sprites and text overlays are best utilized in the Pervasive Game model as assets that provide the bridge between real world and virtual world elements, allowing the development of a variety of pervasive applications.

The idea is to place a user (with a phone) in the centre of a game domain that contains aspects of in-game interaction on the phone and references to real world items and environments that the user can actually see and interact with (to varying degrees).



Reference points in the real world can be locations, items or coordinates, accessed via the camera or location functionality. Micro control is supplied via the device sensors and the screen touch mechanism.

The basic principle behind the system is to overlay various layers video, backgrounds, sprite and text, to build up complex game domains with the user in the centre of both game imagery and environment. The effect on game play is to emerge the user in an engaging series of concentric circles of real world and virtual world content.



Code Listings: The following code listing provides a complete coding solution to the A Pervasive System section.

Common XNA Game Elements	Notes
066 ARTest	<p>Pervasive system test, very quick test (no loop / arrays / lists used) to test the base layering processes, built up of the following layers:</p> <p>Layer 1; Camera input projected to screen.</p> <p>Layer 2: Scrolling background, movement based on compass position.</p> <p>Layer 3: Animated sprites with gesture collision sprites positioned top, bottom, left, right, front and back relative to the user in the centre.</p> <p>Layer 4: Text output to the screen when collision on sprite.</p>

Problems and Solutions

While researching this area and working on some test examples it became apparent that there are several generic tasks that needed to be implemented when using common game elements. The movement of images and backgrounds, collision on images and boundaries, selecting animation cells, responding to input and general in-game-logic tasks were a few that sprang to mind. My solution for the implementation of these generic tasks was the coding of a Game Utilities (GameCode) class (methods are overloaded where required for flexibility).

The following table lists the methods developed (and will be added to during the course of this project):

GameCode.Methods	Overloads	Description
CountSystemTicks()	none	Timer returns true if the interval has passed.
CountPositiveSystemTicks()	none	Timer returns true if the interval has passed. Yields a nonnegative number that cycles between zero and MaxValue once every 24.9 days.
grabCellSheet()	4	Grab cells from animation sheet by incrementing through each row in the sheet. Takes a sprite source rectangle and increments through the animation sheet positions, returns the modified rectangle. Overloads with and without timer.
grabCellRow()	4	Grab cells from animation sheet by incrementing through a defined row in the sheet. Takes a sprite source rectangle and increments through the animation sheet, at a specified row, the row is defined by the parameter cellRow. Overloads with and without timer.
grabCell()	2	Grab a cell from an animation sheet at a defined position in the sheet. Returns the modified rectangle.
rowRelativeMovemet()	none	Calculates the animation row (in the animation sheet) relative to the sprite movement direction. The direction can be set to or away from a point depending on the value in direction parameter. Returns the row position for the animation sequence. It is assumed that the animation sheet will start with the sequences facing West.
rotateSprite()	2	Calculates the rotation angle used to rotates a sprite. Returns the modified rotation angle.
scaleSprite()	2	Scales a sprite depending on its position on the Y axis. The horizon and screen are the movement boundaries, with the minScaleFactor and maxScaleFactor the size the sprite can change between them. The new sailing factor is returned if sprite is in the allowed movement range (Y axis).
moveRelativePoint()	4	Moves a sprite towards or away from a screen touch (or any Vector2 position). Returns the coordinates to move to.
moveInScreen()	3	Move a sprite inside the screen (or rectangle). Bounce of walls if hit at the same angle as the collision. Returns the current spritePosition and spriteSpeed. Overloads to change the animation direction based on movement direction.

GameCode.Methods	Overloads	Description
collisionBox()	none	Check for a basic collision using bounding boxes. Returns true if hit or false if not hit.
Collision()	3	<p>Has three overloads:</p> <ul style="list-style-type: none"> • Check for a basic collision using rectangles. Returns true if hit or false if not hit. • Check for a collision using rectangles based on the direction of the collision. Checks on four directions (each edge). Returns the modified spritPosition rectangle. • Check for a collision using rectangles based on the direction of the collision. Checks on four directions (each edge), then looks at three positions on each edge, the centre region and both edge regions. Returns the modified spritPosition rectangle. There is a check on the corner size for an edge, if corner depth is bigger than half the edge length then the corner size is set to half the length of the edge.
TranslateScreenPointToVector3()	none	Converts a screen touch point to a 3D vector. Calculations use the motion sensor data to calibrate the position.
TranslateVector3ToScreenPoint()	none	Converts 3D vector coordinates into 2D screen coordinates. Based on the devices location motion readings. Calculations use the motion sensor data to calibrate the position.
CompassDirection()	none	Calculates the angle between vectors from (x1, y1) to (x2, y2) Cartesian Coordinates in degrees.
CompassDirectionRads()	none	Calculates the angle between vectors from (x1, y1) to (x2, y2) Cartesian Coordinates in Radians.

Code Listings: The following code listing provides the implementation of the GameCode Methods test bed.

Common XNA Game Elements	Notes
067 UtilsTest	Tests for the methods currently listed in the GameCode.Methods table.

Summary

The coding solutions for the implementation of pervasive games defined in this chapter constitute a Pervasive Games Engine that address the requirements previously set out in Chapter 3.

The following table lists the current engine functionality, that is, a coding entry point to some of the accessible features, setup and configuration options of the platform:

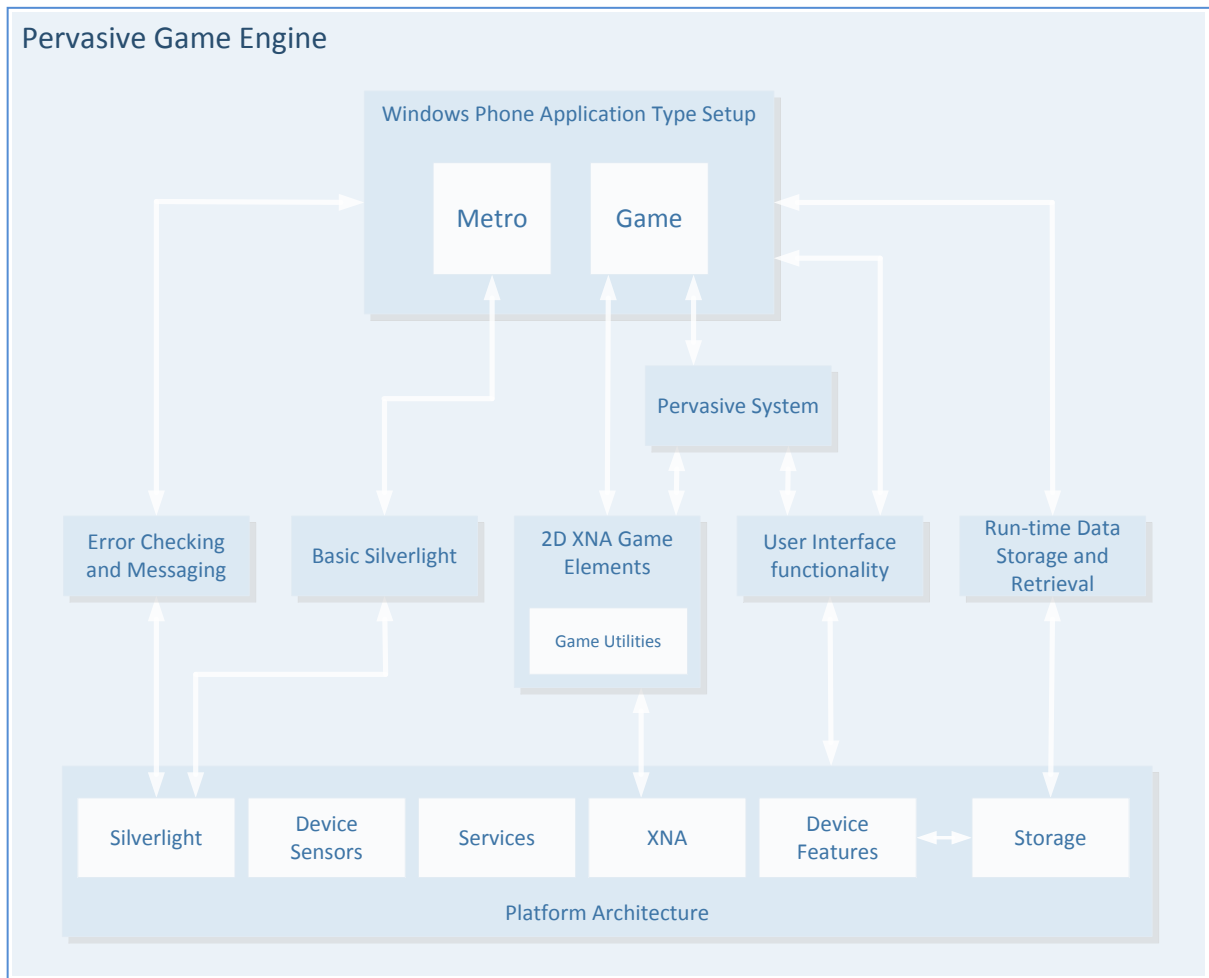
Basic Silverlight Functions <ul style="list-style-type: none"> • Basic page setup • Page orientation • Page navigation • Panorama page setup • System tray • Application bar menu • Event processing • Resources (images, sounds etc.) 	User Interaction <ul style="list-style-type: none"> • Keyboard • Silverlight touch screen • XNA touch screen with gesture support • XNA guide • Microphone • Camera • Vibration • Sensors <ul style="list-style-type: none"> • Accelerometer • Compass (magnetic sensor) • Gyroscope • Motion • Launchers and Choosers <ul style="list-style-type: none"> • Phone, text and email • web, maps and media • Market, contacts and utilities • Radio • Maps <ul style="list-style-type: none"> • Basic Options • Locking Maps • Location finding • Basic and complex collision on locations • Push Pins (images and videos) • Location (movement threshold) • Tiles • Notification services <ul style="list-style-type: none"> • Toast • Tile • Raw
Error Checking and Messaging	
Run-time Data Storage and retrieval <ul style="list-style-type: none"> • Internal data processing • External data storage and retrieval 	
Windows Phone Application Type Setup <ul style="list-style-type: none"> • Silverlight • Silverlight game • XNA games • XNA games with overlays • Silverlight and XNA games 	
2D XNA Game Elements <ul style="list-style-type: none"> • Sprites • Sprite font • Background (scrolling) • Pervasive System • Game Utilities 	

The engine currently comprises 67 coding examples that can be used as an implementation for the topics discussed in this section.

The examples are loosely grouped by topic and represent useful functionality for specific types of interaction.

See [Appendix B: Pervasive Game Engine](#), for a full breakdown of the functionality contained in the Pervasive Game Engine example applications.

See [Appendix C: Pervasive Game Engine Setup and Configuration](#), for any non-standard configuration requirements for the Pervasive Game Engine applications.



The Pervasive Game Engine diagram above represents the usable elements of the engine in relation to the Platform Architecture and Windows Build Types.

The Platform Architecture supplies the base functionality allowing access to Silverlight, XNA, services, device sensors and device features. The engine sits on top of this and provides a quick starting point to develop pervasive applications of varying degrees of complexity.

The development of Silverlight (Metro), XNA and combined Silverlight / XNA pervasive applications is possible, with the selection of functionality dependent on the features required in the application.

Error checking, data storage and a pre-made pervasive system are also available for the quick development of common tasks.

The inclusion of 2D XNA Game Elements can be extended by the use of the GameCode class which provides solutions to several generic gaming tasks (movement of images and backgrounds, collision on images and boundaries and selecting animation cells).

Chapter 7: Development of a Game

In an attempt to prove the effectiveness of the engine from a development view point a pervasive game has been developed to test as much of the engine as possible

The intention of the game is to use a sufficient amount of functionality previously defined in this project on Pervasive Games to develop a working pervasive application.

The current theme of the test game is *Zombie Gold*, but the theme could have been anything and relates to the selection of graphic and sound resources. The current settings for game play are also arbitrary in that they can also be staged in a variety of scenarios.

The following scenarios are used for profiling and testing purposes so that the player's requirements and interaction with game elements can be evaluated in the format of a NIST/IUSR Common Industrial Specification for Usability Requirements document (see Section 9):

- To pass time for passengers on a journey, where the player is a passenger with some time to pass.
- As an exploration game (gold hunt) while at safe locations such as home / parks / recreation areas.
- As a lazy boy aid, to get a game player off the couch and walking about.

The game is also used to test how a pervasive game experience is reflected in the chosen game mechanics and interaction for the player.

The following elements were identified in Chapter 3 of this document as being characteristics of a good pervasive system. They form the focus of the test criteria which is questionnaire based and used to evaluate the effectiveness of the test game (see Appendix J for questionnaire):

- **Pervasive Game Play:** Does the real world influence the state of the game play allowing game actions and real life actions to overlap during the course of play?
- **Spatial Game Elements:** Does the game world go wherever the player goes, into the virtual or real world? Can a player interact with the environment to create new game play based on how they perceive their current game location?
- **Temporal Game Elements:** Is there a blurring of the fixed game duration? Is the game implemented so that it can be played in different time frames and can move from the focus of attention to periphery and back again?
- **Social Game Elements:** Is a game experience provided that draws both the player and non-players into the game, blurring the distinction between those playing and those not playing? Does the engagement of the player draws outsiders into the experience by setting the game play in both real and virtual worlds?
- **Affordances:** Does any accidentally encountered real world situation affect the mood of the game?
- **Ambiguity:** Are important in game elements defining clearly?
- **Emergent Game Play:** Is there scope in the game to allow for surprising coincidences and occurrences (generated by apophenia) in the game that result from a mix of real world and game feedback, to set a mood?
- **Communal Play:** To what extent are play tips accessible in the game?

- Real and Virtual World Considerations: Several of these are implementable in this test genre:
 - Is a real world environment used as a setting to stage a game?
 - Is a real world environment used to set the context for game elements?
 - Are the aesthetics of the environment exploited to set the mood?
 - Is there provision for players and outsiders to receive varying amounts of information relating to the game?
 - Is the player able to interact in the style of the game?
 - Is there movement between a 3D (real world) environment and a 2D display (virtual world) in the game?
 - Are pervasive device features that allow data from environmental conditions used as triggers for game assets?
- Negative Design Considerations: Is consideration given to the following when game play takes place in the real world and potentially affecting non-players:
 - Are measures implemented so that non-Player are not compromised or made to feel uncomfortable in any way?
 - Are measures implemented so that non-players do not misunderstand the game?
 - Are measures implemented so that players stay alert and avoid bashing into people, things and others property?
 - Are measures implemented so that game locations are seen as safe?

The following chapters explain the operation / functionality of the test game (chapter 8), define the intended user tasks (chapter 9) and explain the test process used (chapter 9).

Chapter 8: Test Game: Zombie Gold

Introduction Zombie Gold is a pervasive game developed to test a cross section of functionality previously researched in this project on Pervasive Games.

A Windows 7 Phone is utilised as the delivery mechanism and several main features of the game engine are used in the application, in an attempt to give a believable as possible pervasive experience.

The following table list the elements included in this test from the functionality researched for the game engine:

Basic Silverlight Functions <ul style="list-style-type: none"> • Basic page setup • Page orientation • Page navigation • Panorama page setup • System tray • Application bar menu • Event processing • Resources (images, sounds etc.) 	User Interaction <ul style="list-style-type: none"> • Keyboard • Silverlight touch screen • XNA touch screen with gesture support • XNA guide • Microphone • Camera • Vibration • Sensors <ul style="list-style-type: none"> • Accelerometer • Compass (magnetic sensor) • Gyroscope • Motion • Launchers and Choosers <ul style="list-style-type: none"> • Phone, text and email • web, maps and media • Market, contacts and utilities • Radio • Maps <ul style="list-style-type: none"> • Basic Options • Locking Maps • Location finding • Basic and complex collision on locations • Push Pins (images and videos) • Location (movement threshold) • Tiles • Notification services
Error Checking and Messaging	
Run-time Data Storage and retrieval <ul style="list-style-type: none"> • Internal data processing • External data storage and retrieval 	
Windows Phone Application Type Setup <ul style="list-style-type: none"> • Silverlight • Silverlight game • XNA games • XNA games with overlays • Silverlight and XNA games 	
2D XNA Game Elements <ul style="list-style-type: none"> • Sprites • Sprite font • Background (scrolling) • Pervasive System • Game Utilities 	

Target Group

The game potentially has two usage scenarios:

- To pass time for passengers on a journey, where the player is a passenger with some time to kill. Movement vertically in the game is triggered by the player and horizontally by the direction of the car.
- As an exploration game (gold hunt) while at safe locations such as home / parks / recreation areas.

The theme is interchangeable and the in-game graphics / sounds are implemented to allow theme changes easily (just by changing the graphic and sound files).

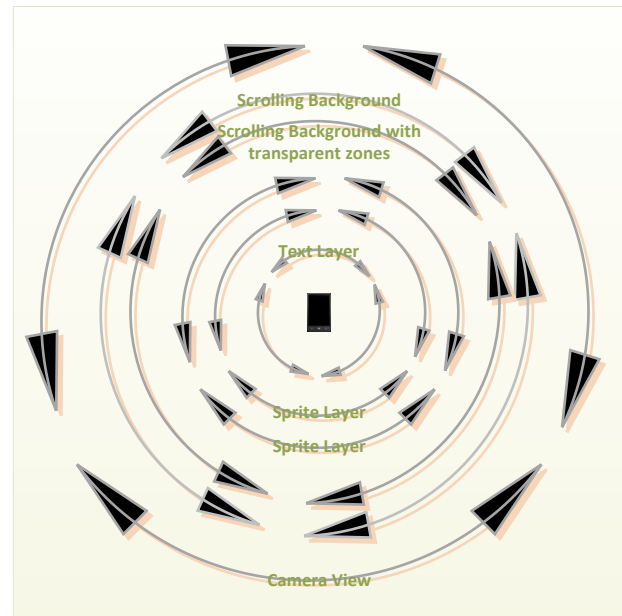
The Game

The game is a combination of treasure hunt (touch screen) and combat (flick screen).

A player searches locations with the phone to collect location type gold. The more locations a player searches (or the more a location is searched) the better chance a player has of collecting all the gold types associated with that location.

If a tilt is generated or the game time expires, then ghosts or monsters attack.

The game is built around a pervasive system that places a player (with a phone) in the centre of the game domain.



The basic principle behind the system is to overlay various layers video, backgrounds, sprite and text, to build up an immersive environment with a player in the centre of both game imagery and game world.

Micro control is supplied via the device sensors and the screen touch mechanism.

Game Modes

The game has two main modes Search Mode (find the treasure) and Attack Mode (kill the monsters).

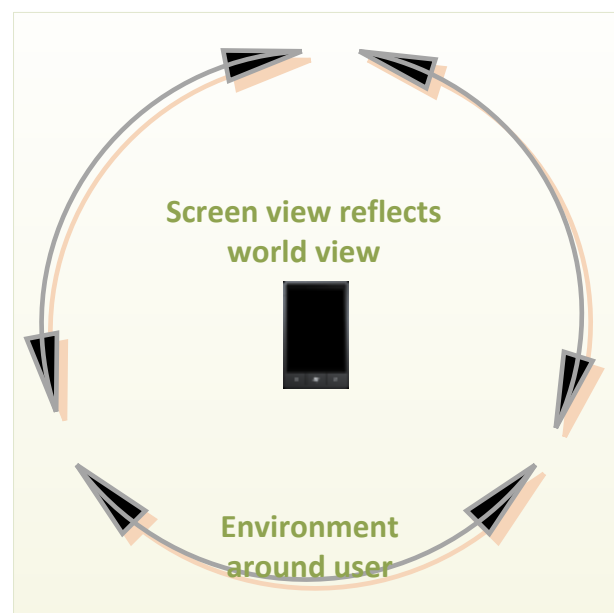
Search Mode

Search mode is accessed when a new game starts or the device is moved to a new location after killing monsters in Attack mode.

In Search Mode a player searches a location for treasure using the phone display and exploring the current location (house or car). Several types of gold need to be found and these are displayed on screen (on the search overlay) for the player.

Gold is retrieved by tapping a gold sprite on the screen; the points for the gold type are then added to the game score.

Gold is generated relative to the player's location and is displayed at a randomly generated coordinate inside a 3D sphere projected out from the device.



When all gold is collected at the current location, the player needs to move (change location) to find the next gold sprite (or wait for the timer to fire).

If the device is moved too sharply (X, Y or Z axis) then a tilt is generated, forcing the Gold sprites to fade, the search timer to decrease, Ghosts to be generated and a screen crack effect to be displayed.

Accelerometer and motion data is used to measure movement thresholds which are displayed on screen as movement tilt bars.

Ghosts may be activated depending on how sensitive a player is with the device or how quickly the device is moved (tilting the device). Ghosts hinder a player from collecting gold by chasing to treasure locations and blowing them up (and themselves). Ghosts can be activated by generating a tilt.

All ghosts track the same gold sprites and the first one to reach it explodes taking out both themselves and the gold sprite when they collide.

When the timer bar expires this means that the player has run out of time and the game moves to Attack Mode where the player is attacked by monsters.

Attack Mode

Attack mode is all about fighting monsters in a pervasive environment. When all the monsters are destroyed the game moves back to Search Mode.

Monsters are destroyed with an on screen slingshot or flick action. Several types of monsters need to be killed and these are displayed on screen (on the attack overlay) for the player.

There are three types of Zombie monsters, crawlers, floaters and flyers.

They attack in formation in the following ways:

- Crawlers grow in size towards the player until they get too big and explode causing damage (lose game points).
- Floaters move towards the player in formation, if they reach the centre of the screen (centre of the 3D domain) game points are lost.
- Flyers, fly above the player in formation, they drop / dive bomb at intervals, if they home in on the player (reach the centre of the 3D domain) game points are lost.

When the player is hit game points are subtracted and a splat effect is displayed on screen.

Combat is configurable in the following three ways:

- Catapult, this give a sling shot action based on a catapult action.
- Slingshot, this give a pullback sling shot action.
- Flick, this give a tap / forwards / backwards flick action.

Game Rules

The game is configured up to level 20 and the following game rules are implemented.

To increase a level in the game the following is required:

- In Search Mode catch all the gold of each gold type (displayed on the Search Mode game overlay) before the game timer runs out.
- In Attack Mode kill all the monsters of each monster type (displayed on the Attack Mode game overlay) before the game points reach zero.

Restarting a level occurs:

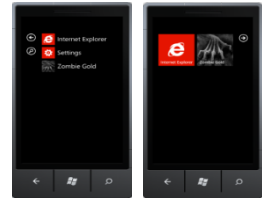
- When the game timer runs out in Search Mode and then Attack mode activates.
- In Attack Mode all monsters of each type (displayed on the Attack Mode game overlay) are killed before the game points reach zero.

Decreasing a level occurs:

- When the game timer runs out in Search Mode and then Attack mode activates.
- In Attack Mode all monsters of each type (displayed on the Attack Mode game overlay) are not killed before the game points reach zero.

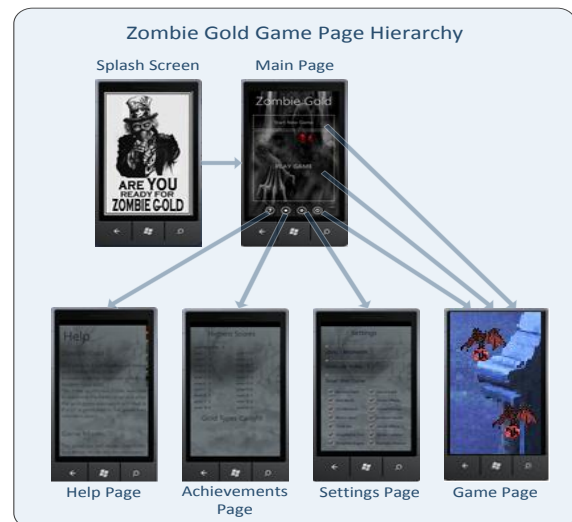
Game Screens

The application can be activated from the phone applications list or by setting up a tile in the phone main menu.



The application has the following screens that are accessible via the in game application bar menu and back button.

- Splash Page
- Main Page
- Game Page
- Settings Page
- Achievements Page
- Help Page



Splash Page

The Splash Page activates when the application first loads.

The page remains active while the sound files are streamed and the game global data is loaded from the Isolated Storage Directory.

Isolated Storage is set up so that any interruptions to game play ‘tombstoning’ or loss of focus are compensated for by saving and retrieval of the current game data.



Main Page

The Main page is the applications main navigation point.

The “New Game” and “Play Game” options are available via buttons on the page.

The application bar menu allows access to the Play Game Option, Settings, Achievements and Help pages.



Game Page

The Game page implements the main game functionality, sound, graphics, camera, location, motion, vibration, sprites and pervasive system.

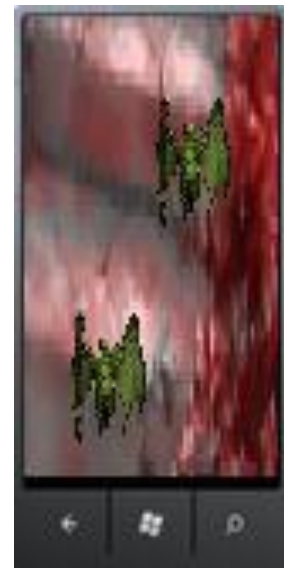
In Search mode the following game layers are processed based on sensor, location, motion and touch data:

- Camera input, displays the real world to screen.
- Gold and ghost sprites, the sprites are defined in a 3D domain (relative to device orientation) and projected onto a 2D screen.
- Search world overlay, scrolling overlay with movement based on compass position and device orientation.
- Tilt detector and timer display, process positional data of the device.
- Game score output, game points processed and displayed.
- Search treasure overlay, calculates the treasure types remaining to catch.
- Screen crack effect, activates graphics, sound and vibration effects when the device is tilted.



In Attack mode the following game layers are processed based on sensor, location, motion and touch data:

- Camera input, displays the real world to screen.
- Attack background, scrolling game background with movement based on compass position and device orientation.
- Zombie monsters 3 types, the sprites are defined in a 3D domain (relative to device orientation) and projected onto a 2D screen.
- Weapon and projectiles, the sprites are defined in a 3D domain (relative to device orientation) and projected onto a 2D screen. The weapon can be configured as Catapult, Slingshot or Flick. The dynamics of the projectile flight path are calculated dependent on the weapon configuration. Collision on the projectiles can be configured as none, explode and bounce.
- Game score output, game points processed and displayed.
- Monster attack overlay, calculates the monster types remaining to catch.
- Screen splats effects, activates graphics and sound effects when the player is hit by a monster.

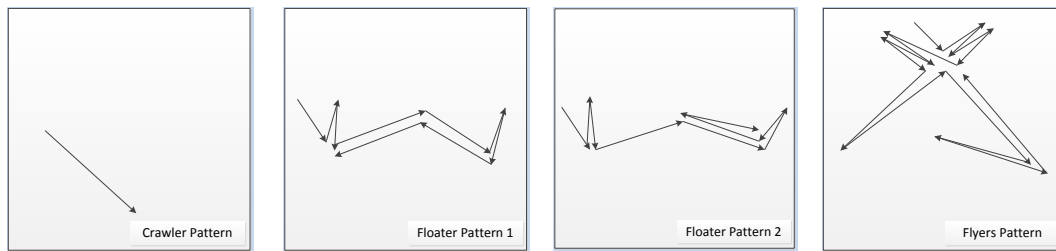


Attack Patterns

The attack pattern for each of the monster attacks is defined as a very basic scripted “AI” movement pattern.

The monster sprites are moved in 3D space then projected onto a 2D screen surface, if a player is facing the correct direction (position calculated based on the device orientation and location).

The flight paths are repeated until the monster is destroyed.



Crawlers are restricted to a position under the horizon.

Floater cycle slightly above and below the horizon in two randomly generated patterns.

Flyers rotate above a player, then cycle slightly above and below the horizon, returning to rotate above a player.

Backing Track

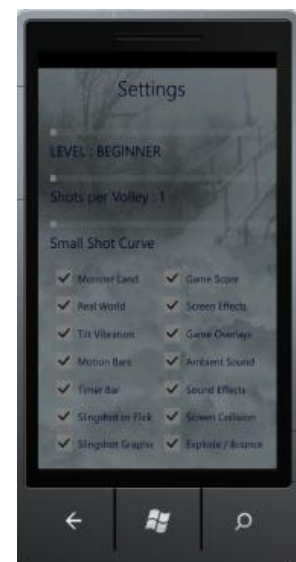
The backing track is generated on a simple random number pattern with a 1 in 5000 chance that a sound will fire each game cycle.

Settings Page

The Settings page offers several options to tailor the game to a player's preference.

The following game parameters and options are configurable:

- Level Slider, sets and displays the current game level (0 – 20). The slider and game level is reset to zero when the New Game button on the Main page is pressed.
- Number of Shots Slider, set the number of shots that can be repeated in attack mode (1 - 12)
- Shot Curve Slider, curve factor added to shot trajectory (0 - 2).
- Monster Land, displays or hides the scrolling game overlay and background.
- Real World, displays or hides the camera input.
- Tilt Vibration, switches the vibration on a tilt on or off;
- Motion Bars, displays or hides the device motion readings display bars.
- Timer Bar, displays or hides the game timer display bar.
- Slingshot or Flick, sets the attack mode to sling shot or flick action.



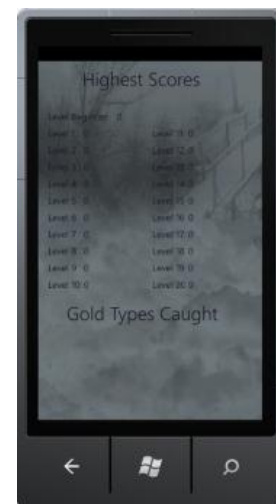
- Slingshot Graphic, displays or hides sling shot animation.
- Game Score, displays or hides the game score display.
- Screen Effects, adds or removes the screen crack (Search mode) and screen splat (Attack mode) effects to the game play.
- Game Overlays, displays or hides the gold type overlay (Search mode) and monster type overlay (Attack mode).
- Ambient Sound, switches the background ambient sound on or off.
- Sound Effects, switches in game sound effects on or off.
- Screen Collision, switches the screen edge collision with the projectiles on or off.
- Explode / Bounce, if the Screen Collision is switched on then this option allows the projectile to explode on screen collision or bounce of the screen on collision.

Note: If the Level Slider is set manually by the player the game will reset to the start of the level.

Achievements Page

The Achievements page displays the current high scores per level.

The Gold types caught from all games played are also displayed as fade in graphics (fade in based on the number caught so far).



Help Page

The Help page provides help on game related topics displayed in a scrolling stack panel.

Graphics and Sound Files

The level of creativity / detail in the graphic and sound arrangements may be lacking, as all the game assets (sound / image) have been downloaded from licence free sites. Some editing has been attempted to make them consistent with the theme but the intention for future development would be to engage an artist.



See [Appendix D: Code Index for the Zombie Gold Application](#), for an explanation of the game files.

Chapter 9: User Personas

This section defines the intended demographic for the game by breaking down several scenarios of use the players will expect of the game and identifies the main participant task scenarios that are required to use the application successfully.

The user base for the game is the current gaming community which represents a cross-section of the whole population. Experience levels are assumed to vary considerably, ranging from beginners with a limited experience of mobile games through to expert users who are very familiar with the interaction style of gaming applications.

The main interaction in the game revolves around users browsing the application pages and playing a pervasive game in two modes, search mode (treasure hunt) and attack mode (combat).

Several scenarios are explored that help to identify the main interaction required by a player as depicted in the Use Case Overview below.



Scenarios of Use

Scenario 1: Learn how to play the game

Joe lives north of Auckland and commutes into the City centre during the week (by bus) where he works as an accountant for a large City Firm.

He has a long bus commute to work each day and tends to spend the time playing on his phone during the journey.

Today on his way to work he downloaded a copy of Zombie Gold from the Marketplace and has had his phone on most of the day next to his laptop checking out the graphics in the game.

Although eager to play the game he frustratingly was too busy all day at work to use the application.


At 5:00pm he heads for home and get his usual bus. When he is settled in his seat Joe gets his phone out, switches it on, clicks on the game icon in the application list and watches the Zombie Gold splash screen as the game loads.

When the main screen appears he is unsure where to go so he expands the menu bar to see several navigation options offered by the game.

Joe selects the question mark icon that is defined as the help option and waits for the Help page to load up.

Joe touches the phones screen when the page loads and realises that the help text is moving when he moves his thumb across the screen (nice feature he thinks). After playing with the scrolling display Joe realises that he can read the text by scrolling left / right and up / down.

On his journey home Joe reads through the help options to learn how to play the game.

When he is finished reading he presses the start menu icon  on the phone and exits the game as his stop is approaching.

As a veteran gamer Joe always wants to explore the help options first to learning how to configure and play a game properly.

Learn how to play the game.

- Start the application.
- Find information on how to play the game.
- Find information on how to configure all the game options.
- Exit the game.

Scenario 2: Play a new game

Tina is a housewife who lives on a house in Hamilton. She is 25 years old, married with two small children Bill and Ben.

She is very busy with the children around all day and has little time to get out during the week as her husband also works long hours. The weekends tends to be spent in Tauranga with her parents as this give the family a break at the weekends and a chance for the kids to see their grandparents.

On Friday evening at 8:00pm the family set out for Tauranga. Usually the journey is a nightmare keeping the kids amused as it is over an hour with toilet stops for the children, but recently Tina downloaded a free copy of the Zombie Gold application onto her and her husband's phones. The game was configured for the easiest game options when downloaded and ready to play.

Once the kids are strapped in the back of the car, they head off for Tauranga.

Tina gets the phones out, switches them on, clicks on the game icon in the application list of each phone and watches the Zombie Gold splash screens as the game loads. When the main screens appear she presses the Start New Game Button on each of the phones and carefully passes the phones to the kids, they get grumpy if she tilts the game before they start playing.


The children can see the search gold icons on top of the screen and know they need to catch all of them to progress to the attack mode (where they can shoot monsters).

As they play they remind each other to look around / up /down for gold and to click on them to catch them.

After a while playing Ben looks at the score on the timer bar of Bill's game and sees Bill is winning. This causes him to search faster but in the process he tilts the game (Bill laughs as he likes the glass cracking effect when it appears in his brothers game), generating several ghosts in the process. Ben's timer is also very low so he desperately needs to catch gold quickly. Bill tells him to keep pressing the screen to keep the ghosts away from the gold and doing so Ben has a large gang of ghosts under his control.

The kids have been having so much fun they have not noticed that the car is pulling into granddads drive.

Tina takes the phones off the boys and reads the final scores, Ben is the winner this time and this causes a minor scuffle in the back seat. Dad shouts and the boys scramble out of the car.

Tina then presses the start menu icon  on each phone and exits the games, the boys are already banging on granddads door.

Play a new game	<ul style="list-style-type: none">• Start the application.• Start a new game.• Play in Search Mode until capturing all the gold.• Exit the game.
-----------------	---

Scenario 3: Setting the game options

Henry is a male employee of the Bay of Plenty Polytechnic, he is 41 years old, married and has one child. Recently Henry ripped his Achilles tendon and found that using a mobile phone game (Zombie Gold) of his son's helped to alleviate the pain.

The combination of turning and holding the phone out / down / above his head, while not focusing on the pain in his foot seemed to give him some mobility again.

The only problem with the game is, his son has the game on a difficult setting which is too fast for Henry to cope with.

Early Sunday morning Henry decides that a quick play in the garden will help get his foot going for the day.

Henry grabs the phone from the desk in his son's room and switches it on. He clicks on the game icon in the application list and watches the Zombie Gold splash screens as he hobbles out into the garden.

Not having setup the game before (his son did it initially) when the main screen appears he is not sure what to do. The menu bar seems to be the best place to start (he thinks), so he expands the menu bar by touching the ... option to see several navigation options offered by the game.


Henry selects the settings menu list option and waits for the Settings page to load up. When he has previously used the game he found the search mode fine, but the attack mode was too difficult to shoot the monsters before he was killed (too stressful).

Henry sits down on a chair in the garden and looks at the options on the setting screen, he still wants all the graphics and sounds (they distract him in a positive way) but decides to make the attack options easier to use.


He selects the flick attack (easy shot action) by touching the Slingshot or Flick selection box until the box is un-ticked, he notices that this also greys out the graphic for the sling shot option which makes sense he thinks.

He then selects the screen collision (more bouncing the easier to hit) option by touching the Screen Collision selection box until the box is ticked, this enables the Explode / Bounce option. He selects the bounce option by touching the Explode / Bounce selection box until the box is un-ticked.

Finally he increases the Shot Curve slider to 2 (add spin) and the Number of Shots slider to 12 (repeated shots).

Henry gets up and starts to walk a little as his leg is getting stiff. He navigates back to the main page by pressing the back button  on the phone. Before he starts playing he wonders if he needs to manually save the new settings or are they automatically saved, so he selects the settings menu list option again and waits for the Settings page to load up. When it opens he sees his new settings are still there so he realises that they have been saved automatically.

Just then his wife calls him from the kitchen, the cat has brought a bird into the house, come quickly and help, great he thinks that will sort his foot out.

Henry presses the start menu icon  on the phone and exits the games, then hobbles for the house.

Setting the game options

- Start the application.
- Set the game option to easy, maximum shots, maximum curve, flick shot and collision bounce.
- Save the current game.
- Exit the game.

Scenario 4: Continue a game

Sid is a male Software Test Engineer living and working in Auckland. He is 26 years old, single and has a high disposable income.

Sid spends a lot of time testing phone applications so he has access to and an interest in all the new applications on the market. Zombie Gold is his current favourite.

This weekend his two smaller sisters (twins) Holly and Molly have come up for Hamilton to Auckland with his parents, who want to do a bit of shopping.

It is Saturday evening and for a treat Sid has set up a scary game for his little sisters. He has put up Halloween decorations around his flat and the girls room (the spare bedroom) is lit with coloured Christmas decorations.

Sid gets a couple of phones from his office desk, they are loaded with Zombie Gold from the last time the twins were over. The games are configured for the easiest game options. Once the twins are sat on the couch in the living room Sid switches them on, clicks on the game icon in the application list of each phone and passes them to his sisters. The girls watch the splash screen and giggle (it looks spooky) as the game loads.


They both press the Play game button and when the game loads exchange phones. They remember the score and number of monsters they need to shoot (shown on the attack overlay) from the last time they played and realise they have the wrong phones.

Both Holly and Molly are experts and leap around the room shooting monsters as the games were both last played in Attack Mode.

As they play the dark rooms in the flat merge with the spooky monsters in the game and it looks like the phone is a magic window that lets them see otherwise invisible monsters in the room.

They both have different playing styles. Holly uses a flick shot and shoots all the crawling monsters first, then the floating ones and finally the flying ones, by spiralling around as she moves around the flat. Molly shoots by keeping her finger in one place for rapid fire and searches for any monster by its size (biggest first), while running frantically about then stopping to fire. Some of the background sounds make the girls jump as they play randomly and this make Sid laugh as the girls try not to show they were spooked.

While the girls are playing Sid chats with his parents and puts the clean washing and food away his parents have brought for him.

10:00pm is approaching and it's time for bed. Sid takes the phones off the girls and reads the final scores, Holly is the winner this time, but the score will be saved automatically so they both know the game is not over. Sid presses the start menu icon  on each phone and exits the games, as the girls are shepherded to brush their teeth.

Sid's parents are happy, with the game and shopping today they know the girls will sleep tonight.

Continue a game

- Start the application.
- Continue a game.
- Play in Attack Mode until the level is completed.
- Save the current game.
- Exit the game.


Scenario 5: Explore the game pages



Lucas lives in Hamilton where he studies Mathematics. He is 30 years old, single and has a relatively low income.

Recently at lunch time he has been playing a new game called Zombie Gold, a pervasive game at several locations around the town. Passers-by have started to notice (and become interested in) the strange antics of Lucas and his friends as they lunge about to catch gold or shoot monsters. Lucas is bit of a techie and is determined to check out everything about the game mechanics (how the thing works).

This lunch time Lucas is alone in the park and decides to investigate the game. He wants to see how the basic mechanics of the game works by removing some of the graphics and effects that distract him.

Lucas takes his phone out and switches it on. He clicks on the game icon in the application list and watches the Zombie Gold splash screens as he eats one of his sandwiches.

He knows the main game page menu fairly well and selects the achievement icon (star) by touching it and waits for the Achievements page to load up. Henry likes the fade in boxes of the gold types, but really just wanted to boost his ego by looking at the massive score he has for level 15. Getting bored he navigates back to the main page by pressing the back button  on the phone.

From the main page he selects the settings icon (cogwheel) by touching it and waits for the Settings page to load up. Lucas leans back on the grass, checks he is still on level 15 and starts to switch off the following options Monster Land, Real World, Tilt Vibrate, Motion Bars, Timer Bars, Game Score, Screen Effects, Game Overlays, Ambient Sound, and Sound Effects. He thinks to himself that the moving sprites in the game will just be placed on the screen randomly and not relate to the movement of the device. He navigates back to the main page by pressing the back button  on the phone. Pressing the Play Game button the game starts in attack mode the setting saved from the last time he played the game and he is surprised to see how realistic the physics on the shot curve are implemented. As monsters are generated it also becomes apparent the game sprites move relative to the devices location and orientation. The conversions from 3D to 2D space are very accurate, Lucas practically likes the fact that you only see what you are facing (a very nice touch). Being engrossed in trying to find any flaws in the game Lucas realises that the game has moved to search mode. Excellent, the ghosts actually track the gold in a 3D space. He then picks up on a bit of insider game play (from watching the ghost movement) if he allows the ghost to start to chase the gold they will show him in what direction to turn. He glances at his watch, time to get back to class he thinks. Lucas presses the start menu icon  on the phone and exits the games, then heads for his statistics lecture.

Explore the game pages


- Start the application.
- Check the current score.
- Check the current achievements.
- Check the current level.
- Set all the game layers off
- Exit the game.

Scenario 6: Configure and play the game on a difficult setting

Tom is a male student living and studying in Dunedin. He is 19 years old, single and spends a lot of time avoiding work, snowboarding and texting his friends. Recently Tom and his friends have taken to playing a pervasive game (Zombie Gold) in the oddest locations they can think of.


The challenge is based on a player succeeding in completing a level at a strange location before leaving the location (or be moved on) with the game setting set to the hardest difficulty possible. Tom is the current leader at Level 9 and today is going for a level 10 attempt on the spinning Tea Cup ride at the local kids fair. There is a 5 ride limit set by the students for a successful attempt to be recognised.

Tom and his friends get ready for the official attempt by paying for 5 rides (\$3 per ride). While in the queue he takes his phone out and switches it on. He clicks on the game tile he has pinned to the phones main menu and watches the Zombie Gold splash screen as he moves towards the pay booth. When the main page loads he selects the settings icon (cogwheel) by touching it and waits for the Settings page to load up. His friends cluster around to check that he sets the correct options for the attempt to be valid. Tom checks his level setting is still on level 10 and starts to check that the following options are ticked Monster Land, Real World, Tilt Vibrate, Motion Bars, Timer Bars, Game Score, Screen Effects, Game Overlays, Ambient Sound, Sound Effects. He is reminded by his friends that the following also need setting Shots per Volley 1, Small Shot Curve 0, Slingshot or Flick ticked (Slingshot), Slingshot Graphic ticked, Screen Collision ticked and Explode / Bounce ticked (Explode). Tom makes the settings changes and pays for his tickets.

The gang take their seats in a pink tea cup, waiting for the ride to start Tom navigates back to the main page by pressing the back button  on the phone. As the ride starts Tom touches the Play Game button on the main page and the game begins. Tom hasn't thought about the fact that his three friends are not only in the cup with him but will be spinning the cup with the wheel in the centre of the seats. He can catch the gold but the tilt is impossible to control and the game is swamped with ghosts. As the first ride ends he has tilted the game into attack mode and now needs to kill all the monsters just to restart the level.

Luckily the ride stops for quite a while for the other cups to fill up and in attack mode the monsters generate 3 low numbers. By the time the ride starts again he is back to search mode with 3 hysterically laughing friends next to him. The next two rides see Tom clearing search mode successfully and re-entering attack mode with a chance to level up.

That's when disaster strikes Tom has forgotten to charge his phone overnight, the screen goes blank and the battery critical message pops. End of game, end of attempt, Toms mates are in hysterics, Tom moves to legendary status.

When Tom gets home he charges the phone and has a quick check to see if the game is still current at the point it stopped. He clicks on the game menu tile and when the main page loads touches the Play Game button. The game loads and is still in the exact state it was on the ride. Tom quickly presses the start menu icon  on the phone and exits the games.

The texts begin to fly and the debate begins, can the attempt be continued..... or is it a restart.....

Configure and play the game on a difficult setting	<ul style="list-style-type: none">• Start the application.• Set the permutations of the game option, minimum shots, minimum curve, sling shot with graphic, collision explode and level slider 10.• Continue a game.• Play the game rule paths (Search Mode and Attack Mode) for a level dependent on the player's success or failure.• Exit the game.
--	--

Participant Task Scenarios for Zombie Gold

The following tasks are based on those initially identified in the previous Scenarios of Use and are used for profiling and testing purposes so that the player's requirements and interaction with game elements can be evaluated in the format of a NIST/IUSR Common Industrial Specification for Usability Requirements document (see Chapter 10).

Scenario 1: Learn how to play the game (in car)

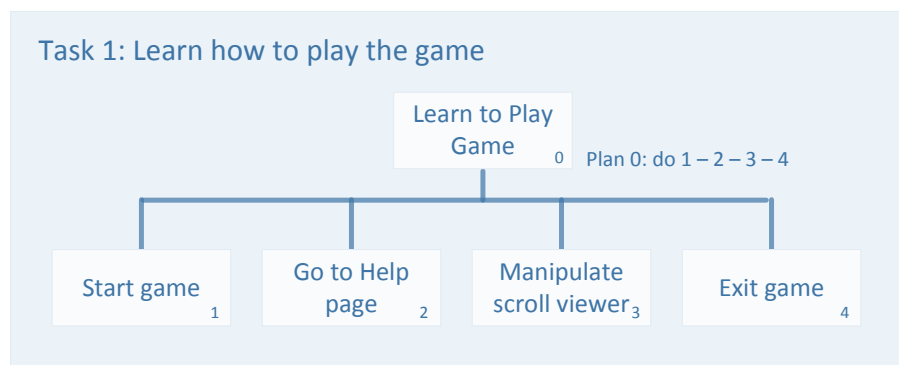
You are an accountant called Joe who lives and works in Auckland.

*This morning on your way to work you downloaded a copy of **Zombie Gold** from the Marketplace and have been waiting all day to play the game.*

As a veteran gamer you always want to explore the help options first to learn how to configure and play a game properly. On your way home on the bus you decide to learn about the game.

Task 1: Learn how to play the game

- 1.1 Find the **game icon** and open the game.
- 1.2 Find and select the **Help navigation icon** on the Main screen.
- 1.3 Find out how to **read the Help text** on the Help page.
- 1.4 Read the section on **Game Rules** in the help text.
- 1.5 Read the section on **Settings Page** in the help text.
- 1.6 Exit the game using the **Start Menu** icon on the phone.



Scenario 2: Play a new game (in car)

You are a young housewife called Tina who lives with your husband and two children on batch in Hamilton. You very busy with the children around all day and have little time to get out during the week.

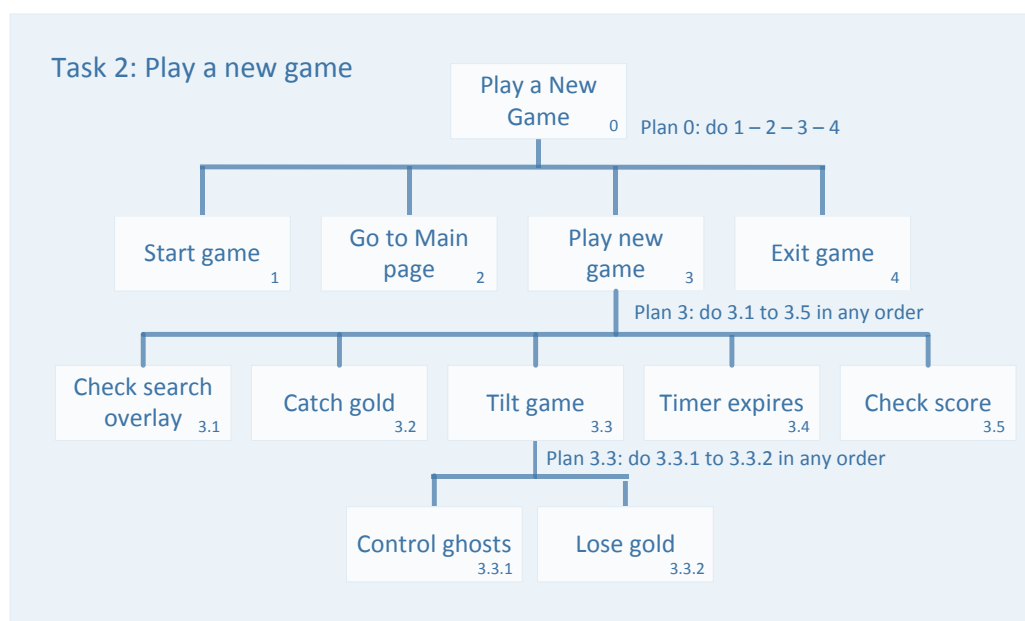
Your weekends tend to be spent in Tauranga with your parents as this give the family a break at the weekends and a chance for the kids to see their grandparents.

On Friday evening at 8:00pm the family set out for Tauranga. Usually the journey is a nightmare keeping the kids amused as it is over an hour with toilet stops for the children, but you have downloaded a free copy of the Zombie Gold application onto a couple of phones.

The game was configured for the easiest game options when downloaded and is ready to play. Once the kids are strapped in the back of the car, you get the phones out for the children to play with and head off for Tauranga.

Task 2: Play a new game

- 2.1 Find the **game icon** and open the game.
- 2.2 Find and select the **Start New Game button** on the Main screen.
- 2.3 Find the **search gold icons on the top** of the game screen and **note how many there are** and **the values on each one**.
- 2.4 Play the game looking around / up /down for gold until you catch one.
- 2.5 Tilt the game by moving the game too quickly.
- 2.6 Control a ghost by tapping the screen (move the ghost to centre of the screen).
- 2.7 Allow a ghost to catch a gold sprite (stop tapping the screen) follow it on the screen until it explodes.
- 2.8 Play the game until search mode is completed successfully or the game timer expires.
- 2.9 Exit the game using the **Start Menu** icon on the phone.



Scenario 3: Setting the game options

Your name is Henry and you are a male employee of the Bay of Plenty Polytechnic, you are 41 years old, married and have one child.

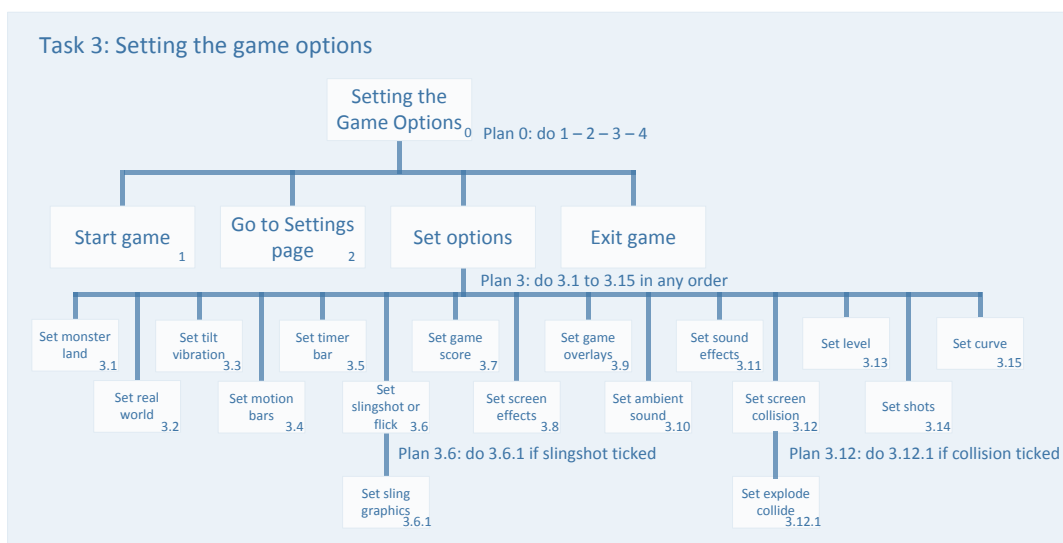
Recently you ripped your Achilles tendon and have found that using a mobile phone game (Zombie Gold) of your son's has helped to alleviate the pain. The only problem with the game is that your son has the game on a difficult setting which is too fast for you to cope with.

Early Sunday morning you decide that a quick play in the garden will help get your foot going for the day.

You sit down on a chair in the garden with your phone and decide to reset the game option to an easier mode before you play.

Task 3: Setting the game options

- 3.1 Find the **game icon** and open the game.
- 3.2 Expand the **menu bar** on the Main screen by touching the ... option
- 3.3 Select the **settings menu list option** to load the Settings page.
- 3.4 Set the following options on the **Settings** page to make the game easier to play:
 - Slingshot or Flick option to un-ticked.
 - Screen Collision option to ticked (this enables the Explode / Bounce option).
 - Explode / Bounce option to un-ticked.
 - Increase the Shot Curve slider to 2.
 - Increase the Number of Shots slider to 12.
- 3.5 Navigate back to the **Main** page by pressing the **Back button** icon on the phone.
- 3.6 Select the **settings menu list option** on the Main page menu bar to load the Settings page again, to see if the setting you previously changed have remained changed (saved).
- 3.7 Exit the game using the **Start Menu** icon on the phone.



Scenario 4: Continue a game

You are Software Test Engineer called Sid living and working in Auckland.

You spend a lot of time testing phone applications so have access to and an interest in all the new applications on the market. Zombie Gold is your current favourite.

This weekend your two smaller sisters (twins) Holly and Molly have come up for Hamilton to Auckland with your parents.

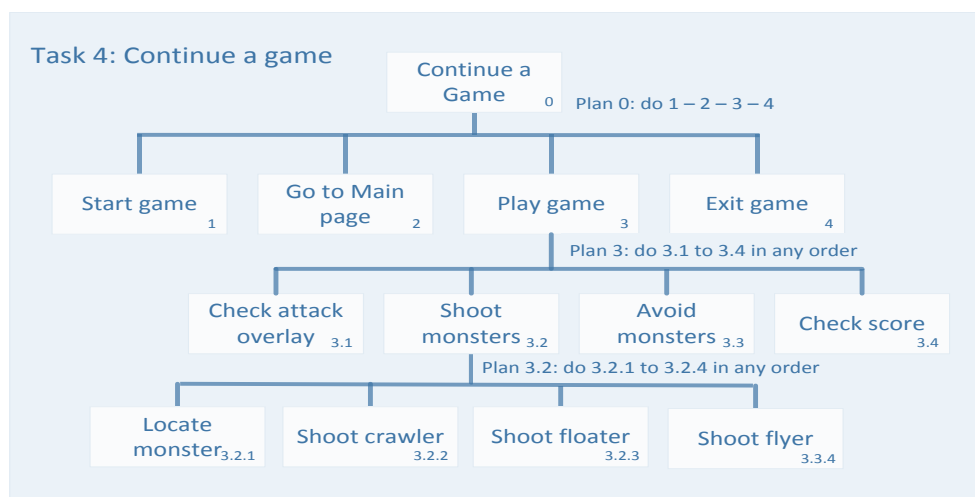
It is Saturday evening and for a treat you have set up a scary game for your little sisters, by putting up Halloween decorations and coloured Christmas lights around your flat.

You get a couple of phones out that are loaded with Zombie Gold from the last time the twins were over. The games are configured for the easiest game options.

Once the twins are sat on the couch in the living room you start the games and hand them the phones.

Task 4: Continue a game

- 4.1 Find the **game icon** and open the game.
- 4.2 Check that the splash screen appears while the game loads (record how long it is visible).
- 4.3 Find and select the **Play Game button** on the Main screen.
- 4.4 Find the current **Score on the bottom** of the game screen and **note how many points you have**.
- 4.5 Find the **attack monster icons on the top** of the game screen and **note how many there are and the values on each one**.
- 4.6 Listen until a random **sound** is generated for the background noise in the game.
- 4.7 Play the game looking around / up /down shooting monsters until you **shoot one of each of the three monster types**.
- 4.8 Bounce a shot of the screen edge using a **pullback flick shot**.
- 4.9 Let one of the monsters hit you to generate a screen splat.
- 4.10 Play the game until attack mode is completed successfully or you run out of points.
- 4.11 Exit the game using the **Start Menu** icon on the phone.



Scenario 5: Explore the game pages

You are a student called Lucas who studies Mathematics in Hamilton.

Recently at lunch time you have been playing a new pervasive game at several locations around the town. You are a bit of a techie and are determined to check out everything about the game mechanics (how the thing works).

This lunch time you are alone in the park and decide to investigate the game. You want to see how the basic mechanics of the game work by removing some of the graphics and effects that distract you during normal game play.

You take your phone out and switch it on as you start munching your sandwiches.

Task 5: Explore the game pages

- 5.1 Find the **game icon** and open the game.
- 5.2 Find and select the **Achievements navigation icon** on the Main screen.
- 5.3 Find the current score for **Level Beginner** on the Achievements page, note the score.
- 5.4 Navigate back to the **Main** page by pressing the **Back button** icon on the phone.
- 5.5 Find and select the **Settings navigation icon** on the Main screen.
- 5.6 Set the following options on the **Settings** page to make the game easier to play:
 - Decrease the **Level slider** to **Beginner**.
 - **Monster Land** option to **un-ticked**.
 - **Real World** option to **un-ticked**.
 - **Tilt Vibrate** option to **un-ticked**.
 - **Motion Bars** option to **un-ticked**.
 - **Timer Bars** option to **un-ticked**.
 - **Game Score** option to **un-ticked**.
 - **Screen Effects** option to **un-ticked**.
 - **Game Overlays** option to **un-ticked**.
 - **Ambient Sound** option to **un-ticked**.
 - **Sound Effects** option to **un-ticked**.
- 5.7 Navigate back to the **Main** page by pressing the **Back button** icon on the phone.
- 5.8 Find and select the **Play Game button** on the Main screen.
- 5.9 When the game page loads check that all the options are off, no effect layers will be displayed.
- 5.10 Exit the game using the **Start Menu** icon on the phone.



Scenario 6: Configure and play the game on a difficult setting

You are a student called Tom living and studying in Dunedin. Recently you and your friends have taken to playing a pervasive game in the oddest locations you can think of.

The challenge is based on a player succeeding in completing a level at a strange location (or situation) before leaving the location with the game setting set to the hardest difficulty possible. You are the current leader at Level 9 and today you are going for a level 10 attempt. The challenge is to play a level while a friend (another test volunteer) spins a revolving office chair you will be sat on while playing the game.

You take out your phone and switch it on while sitting in the chair.

Task 6: Configure and play the game on a difficult setting

- 6.1 Find the **game tile** in the **phone main menu** and open the game.
- 6.2 Find and select the **Settings navigation icon** on the Main screen.
- 6.3 Set the following options on the **Settings** page to make the game harder to play:
 - Set the **Level slider** to **10**.
 - **Monster Land** option to **ticked**.
 - **Real World** option to **ticked**.
 - **Tilt Vibrate** option to **ticked**.
 - **Motion Bars** option to **ticked**.
 - **Timer Bars** option to **ticked**.
 - **Game Score** option to **ticked**.
 - **Screen Effects** option to **ticked**.
 - **Game Overlays** option to **ticked**.
 - **Ambient Sound** option to **ticked**.
 - **Sound Effects** option to **ticked**.

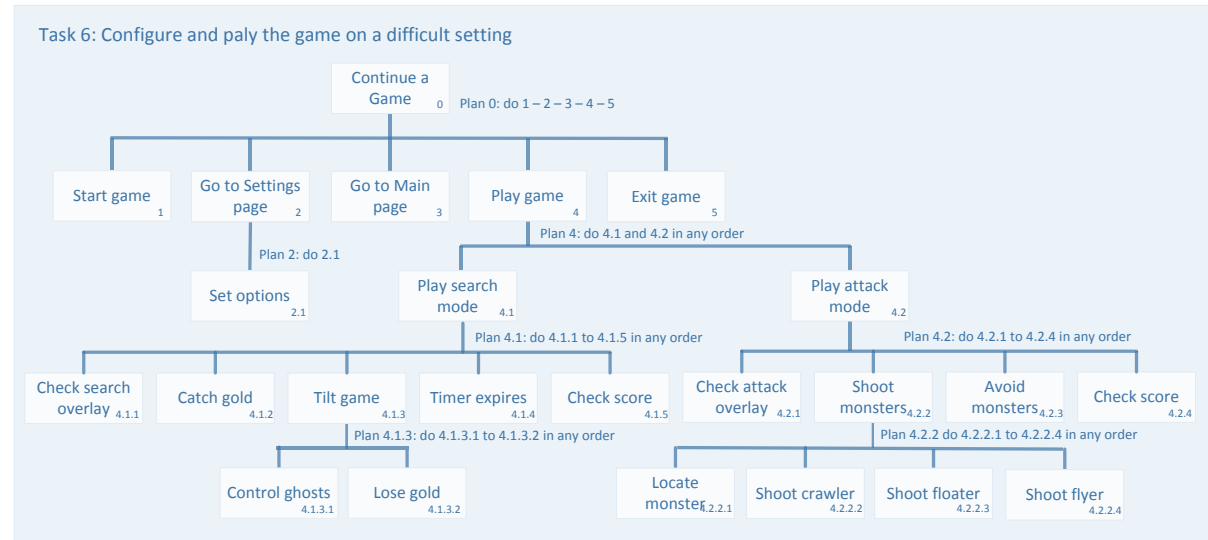
You friend (other test volunteer) reminds you that you also need to set the attack options to difficult, as you are going for a level 10 attempt at the record.

- 6.4 Set the following options on the **Settings** page to make the game harder to play:
 - Shots per Volley slider to 1.
 - Small Shot Curve slider to 0.
 - Slingshot of Flick option to **ticked** (Slingshot).
 - Slingshot Graphic option to **ticked**.
 - Screen Collision option to **ticked**.
 - Explode / Bounce option to **ticked** (Explode).
- 6.5 Navigate back to the **Main** page by pressing the **Back button** icon on the phone.

You friend (other test volunteer) begins to spin the chair and move you around the room.

- 6.6 Find and select the **Play Game button** on the Main screen.

- 6.7 Play a **game rule path** for the game level, both the search and attack modes, the game length will be dependent on your success (progress a level) or failure (go back a level) playing the level.
- 6.8 When you have completed the level play, exit the game using the **Start Menu** icon on the phone.



Chapter 10: Usability Requirements

The application was tested using NIST/IUSR – CISU-R Common Industrial Specification for Usability – Requirements format (see [appendix E: NIST/IUSR – CISU-R](#) for the full usability testing report).

Test Criteria

The performance criteria for the application was measured against the ability of appropriate users to complete goal tasks without any assistance or training, apart from the help and information supplied by the application. On completion of the tasks, users needed to be least as satisfied with the game, as they were with any existing games (with similar functionality) they had used. Levels for effectiveness, efficiency, satisfaction and pervasive experience were established in relation to the applications usability and the gaming experience provided.

Test Results

The following results were obtained from the testing process.

Effectiveness

All core tasks of the game, tasks users were required to fully perform for the game to play successfully, were rated at 100% effectiveness. Based on the data from the tests this clearly validates the game play mechanism.

Efficiency

Based on the test data for efficiency, the indication was that the correct in game balance had been achieved, to provide challenging game play at increasingly harder levels, without the game becoming too tedious for the player.

Satisfaction

A very high satisfaction level (97.5%) was returned on completion of the tests. This was an unexpectedly high result and for the development team it made all the hard work put into the implementation of an easy to use and intuitive application worthwhile. Data and feedback from the tests indicated that the participants had found the application intuitive, easy-to-use, fun (they all wanted to know the release date) and surprisingly uncomplicated for a mobile phone game.

User Feedback

Overall game play received very positive feedback:

- All test participants were satisfied with the content and play duration.
- The test participants thought the game was fantastic and were fully engrossed.
- Search mode features noted very positively included, the tilt effect generally liked and ghost generation.
- One of the test subjects fully immersed themselves in the mood of the game by making the lab dark and became absorbed in the spookier aspects of the game.
- Attack mode features noted very positively included, shot action (better than angry birds), attack formations, bouncing and swerve on the shots.

- The auto save feature was generally liked. With feedback on the lines of, means you can take a rest anytime you want and the game saves, whatever state you put the phone in the application was in the same state when you returned to it.
- All the test participants liked the style of interaction and partially the menu system, settings layout and scroll text in the help display.

During the testing a couple of suggestions were addressed to the development team for further inclusion:

- A preference for the help text as a sound file activated when clicked.
- Pictures and diagrams in the help topics should be used.
- More graphics sounds and attack types in the next version please.

Pervasive Experience Factor

The results from the tests returned a PEF level of 86%, indicating a fully immersive pervasive experience for the players was achieved. This was also substantiated by test participant feedback that had been recorded during the tests.

Several points scored practically highly on the PEF scale with all test participants agreeing strongly with the following:

- The game world goes wherever the player goes.
- The game can be played when the player wants.
- A real world environment is used as a setting to stage a game.
- The game links 3D (real world) and 2D (virtual world) environments.
- Features of the phone are used as triggers in the game.

This indicated that the intention to create a pervasive game that bridged the conventional game boundaries of real world and virtual world had been realised. It also proved that two of the main pervasive game elements, spatial and temporal game play strategies had been implemented successfully in the application.

Feedback gathered during discussions with participants as part of the testing process focused exclusively on the pervasive experience provided by the game and several aspects of the experience stood out for the participants, again collaborating the proposition that the game had been implemented successfully.

The in-game movement between real and virtual domains figured as the main point of interest for the test participants eliciting the following feedback:

- When you move the phone around it's like you are seeing things that are invisible in the room.
- You need to play the game all around the house, things then seem to pop out of the background.
- When you switch the lights off it makes the place really spooky.
- The music and graphics shock you sometimes, you are in a location and it all fits together, strange.
- You sort of become a child again; it's like catching butterflies in the garden with a net.
- The game jumps between outside and inside the game, you get lost in it at times.
- As you move the world moves and things pop up.
- When you are catching gold and shooting monsters you need to move the phone around your body and move about the room, it is fun.
- It was awesome; the game had stuff flying around all over the place.
- There is this tilt thing that if you move the phone too quickly, the glass breaks and it buzzes and stuff.
- When you are playing you start to think of weird places you could play to get people going.

Final Result

One of the stated goals of the application was as a pervasive game developed to test a cross section of functionality previously researched in this project on Pervasive Games. Here the application was successful based on the usability data provided from the tests.

The application successfully utilised a cross section of the functionality from the prior research in this project and provided an entertaining game in the pervasive style as proof.

The main proof is the amount of feedback from the users saying the same thing **THE GAME IS FUN**.

To date everyone who has played the game has enjoyed the experience, to the extent that one of the test phones is currently touring around the south island on loan (to amuse the children while driving) and there is already a group planning to use the game at odd locations (and set up a league), an idea taken from the testing.

See [appendix E: NIST/IUSR – CISU-R](#) for the full usability testing report.

Chapter 11: Conclusions

Pervasive Game Play

I started this project with the intention of looking at an area of gaming that has rapidly evolved over the last few years, pervasive game play. It is a phenomenon that is generating some major development in the gaming industry particularly in the areas of mobile platforms (pads and phones), new peripherals (for Wii, Xbox Connect etc.), the move to touch screen input and centrally controlled entertainment environments (Windows 8 / Windows Live).

In an attempt to find a starting point for my research on pervasive game play I initially looked at what exactly is play. This proved to be fascinating reading (it would make interesting project in itself). 'What is play' seems to be one of those questions on the lines of 'What is art', and has been discussed constantly in the literature, with no definitive conclusion (lots of passion, fewer firm conclusions).

The concept of pervasive play on the other hand proved to be a relatively modern concept evolving around the 1930s and amending itself into the 50's. The main premise being the classification of a game domain (Magic Circle) then for the pervasive experience to be effective the transition of game play between this virtual world and the real world non-game domain.

It was also clear from the literature that the advent of mobile technology and external device peripherals around 2000 (and onwards) has combined with a subtle shift towards a pervasive view of game play to create a new perception (and expectation) of game play for the 'text' generation.

This initial research produced a very positive result for me. It established a direction for my research to follow, by generating the next research question, 'What exactly are the physical requirements that constitute a Pervasive Game Experience'.

What constitutes a Pervasive Game Experience?

Answer: Pervasive Games are interactive game where the participants drive the game play. The key element in a Pervasive Game is the elimination (if possible) or distorting of the Magic Circle. This is achieved by adding features to a game that expand the contractual Magic Circle of play spatially, temporally and socially.

Unexpectedly finding the answer to this question proved difficult as there are very few definitive texts on pervasive gaming and the literature is relatively scarce. The majority of my research therefore focused on current research on the topic in the form of University papers, articles and also breaking down the key elements in the game play of existing applications (invaluable later in developing a coding solution).

Although laborious, this phase of my research allowed me to establish several key elements common to existing pervasive applications, define some real / virtual world consideration for game play (positive and negative), identify the technical requirements needed to implement play elements on a mobile device and note the game types best suited to the genre.

Overall the research produces some interesting results, in that I was able to establish the boundaries of this project for myself, the focal point being to develop a Pervasive Game Engine (functionality useful in the development of pervasive games and demonstrate / explore an example of pervasive games using this engine).

It also provided me with access to a considerable amount of resources and a clearer idea of what I wanted to achieve, understanding the mechanics involved in the development of Pervasive Games.

Selecting a Mobile Phone Platform

Selecting a platform was probably the easiest part of the project for me. Although I compared several platforms (Android, iPhone (iOS), BlackBerry, Symbian and Windows 7.5) and looked at future market trends from the onset I was slightly enamoured with the Windows 7 platform and the amount of resource Microsoft had put at the disposal of developers.

So based on my preference (from my research) I decided to go for the Windows 7 platform, Microsoft Smartphone Phones (Nokia Lumia 800 top range device, HTC Trophy and Nokia Lumia 710 mid-range devices) and the Microsoft 2010 (C#) Integrated Development Environment. This is where I put my funding for the project in the purchase of a developers licence and three Windows 7 smart phones (not cheap).

As a consequence of the project being over two years (part time) and the funding only available to me at the onset, it placed a couple of financial restricts on the project for me. Basically it locked me into the Windows 7 hardware / OS updates for the duration of the project. As a consequence I limited the scope of this project to utilising the Windows 7 phone and development platform. All the concepts, code (is portable) and test mechanisms are valid for the new Windows 8 hardware / OS (and future releases) but my test and distribution mechanisms for this project are Windows 7 compliant.

Coding Requirements

The next phase of my project consisted of two main tasks, investigate and get a development platform (with delivery mechanism) up and running, then looking at the theoretical elements required to develop pervasive games and implement a pervasive game engine.

I consider several setups for the engine from API libraries to a more complete development environment, but decided that as the key elements for a pervasive game interaction had been identified clearly in the initial research phase I would stick to that format and provide an implementation for each element.

This produced an engine in the format of 67 code stubs (applications) that implement solutions to elements required in the development of a pervasive application. They are intended to be used as quick access points for working code solutions that can be easily combined to form a completed application.

i.e. If you need to get the camera, touch screen and sprites working in an application just make a project and copy the code in.

Because of the nature of the isolated storage and sprite operations I also implemented two helper classes containing this functionality.

Setting up the System

This is what attracted me to the Windows platform in the initial research phase. The whole process of setting up the environment proved to be fairly straightforward although time consuming. The separate components worked well together and all interaction with Microsoft has been seamless. A massive plus was a centralise location for developers, the resources were excellent (long hours spent watching on line tutorials).

The main chore was wading through the certification and compliance documentation from Microsoft. This is a big one for Microsoft and in hindsight there was more documentation on the implementation / release / compliance standards and requirements than I found on the design considerations of pervasive applications.

This is a point worth noting from my research on existing applications. The development (get it working with the new technology) tended to take precedence over design considerations (methodology). I felt it reflects the appearance over the last few years of new devices with extended functionality every few months (if not weeks).

In my opinion (and it was reinforced in the nature of conversations on the developer forums) the old 8-bit culture is emerging again among developers, of get it in and get it working. Hopefully it will lead to some innovative games like those prevalent in the early 90's.

My exploration of Metro design compliance proved to be interesting. It is a slightly restrictive concept but the restrictions and the availability of free resources for the graphics takes the pressure off the none-creative developer. Surprisingly the Metro design allows some really quick development of very professional looking interfaces. Metro seems not to be popular among my colleagues but after becoming familiar with the system as part of this project, I am now a fan.

There is a facility (Blend) to extend the appearance of the graphics (graphic designer tool) but I felt its inclusion was outside the scope of this project.

Windows 7 Coding Issues

To make the project viable I felt that the game engine should clarify the theory behind a topic then present a basic code solution. This was my reason for including code snippets as I felt they were the main point of the exercise. As a developer you tend to need some clarification on how it works and an example on which to base your implementation.

The idea was to research the basic operations and then focus on specific pervasive features. Windows 7 development being a task-centric philosophy, looking at the basic application format seemed a sensible starting point and it addressed the basic requirements for the Metro style previously researched.

Some topics (events, resource files build options) are included for my later ease of use and represent information that game developers will require even though they are not strictly game API components. This will be an on-going addition to the engine in the future as and when I encounter other topics for which I need a quick reference.

Implementing normal application debugging methods (error checking and messaging) required some research as the device plays a major role in displaying data, so the normal strategies needed to be phone specific.

I then encountered a real show stopper (in the course of my research), external data storage. There is a recognised intrinsic problem associated with data processing in mobile devices and it stems from the fact that the device is a phone. On a Windows 7 phone a developer does not know what will happen to an application while running as it may be interrupted or closed down. Even so there is a requirement of providing a seamless user experience and continuity between the closed down and restart states of the application (for compliance). Researching the problem I felt that I could come up with a set of functionality to alleviate some of the basic issues. I have therefore included a set of methods (StorageUtils.cs) into the project that facilitate the fast saving and retrieval of data (various formats) to / from the Isolated Storage file directory, the Isolated Storage Settings Directory and the Current State Directory. The class was tested and proven to comply with the Microsoft recommended requirements set out for speed of data transfer and user experience.

The whole area of basic user interaction with the device was next on my research list, as user interaction via a mobile phone offers several unique possibilities for the development of Pervasive Games. All input routes utilise relatively new technologies, from basic input via a touch screen (keyboard and touch input) to the use of built in devices / sensors (accelerometer, compass, gyroscope and microphone) and OS / Cloud services (tile notification, launchers and choosers). A fascinating area to look at which resulted in the development of some really useful pervasive orientated code. Looking into location finding and mapping (exploration of real world locations) proved to be a fairly important feature of pervasive games, so I looked at the implementation requirements of mapping in some detail, from the Bing Maps registration process through to the development of several useful features (basic options, finding locations, collision and adding pushpins).

In the Microsoft environment there is a large set of project templates that can be used to set up project types quickly. The development done during this project focused on the following three templates Windows Phone Application, Windows Phone Game and Windows Phone Silverlight and XNA Application. I found the Phone Silverlight and XNA Application template the most flexible as the name implies it allows both Silverlight objects and XNA elements to be implemented in the same application. The example game presented in this project was developed using this basic template.

While researching XNA content it became apparent that several conventional gaming elements could be used to blur the distinction between real world and virtual world game play in the pervasive game model. Sprites, backgrounds (static or scrolling), text, sensor input, gesture input, sound and overlays, were the main components identified.

It was also clear that several generic tasks needed to be implemented (the movement of images and backgrounds, collision on images and boundaries, selecting animation cells, responding to input and general in-game-logic tasks) for the pervasive engine to be effective. In response to this I developed a solution to these generic tasks, a utilities (GameCode.cs) class made up of several overloaded (where required) methods.

The result of this protracted research and development phase was the creation of a pervasive game engine in the form of coding solutions. It is not the definitive list but a good starting point (reference) to be added to.

The following table lists the current engine functionality:

Basic Silverlight Functions <ul style="list-style-type: none"> • Basic page setup • Page orientation • Page navigation • Panorama page setup • System tray • Application bar menu • Event processing • Resources (images, sounds etc.) 	User Interaction <ul style="list-style-type: none"> • Keyboard • Silverlight touch screen • XNA touch screen with gesture support • XNA guide • Microphone • Camera • Vibration • Sensors <ul style="list-style-type: none"> • Accelerometer • Compass (magnetic sensor) • Gyroscope • Motion • Launchers and Choosers <ul style="list-style-type: none"> • Phone, text and email • web, maps and media • Market, contacts and utilities • Radio • Maps <ul style="list-style-type: none"> • Basic Options • Locking Maps • Location finding • Basic and complex collision on locations • Push Pins (images and videos) • Location (movement threshold) • Tiles • Notification services <ul style="list-style-type: none"> • Toast • Tile • Raw
Error Checking and Messaging	
Run-time Data Storage and retrieval <ul style="list-style-type: none"> • Internal data processing • External data storage and retrieval 	
Windows Phone Application Type Setup <ul style="list-style-type: none"> • Silverlight • Silverlight game • XNA games • XNA games with overlays • Silverlight and XNA games 	
2D XNA Game Elements <ul style="list-style-type: none"> • Sprites • Sprite font • Background (scrolling) • Pervasive system • Game Utilities 	

Zombie Gold

To test the engines effectiveness Zombie Gold was conceived, a pervasive game developed to test a cross section of functionality previously researched in this project. The intention being to generate a pervasive experience that could be tested using recognised usability standards.

The following table lists the elements included in this test application from the functionality researched for the game engine:

Basic Silverlight Functions <ul style="list-style-type: none"> • Basic page setup • Page orientation • Page navigation • Panorama page setup • System tray • Application bar menu • Event processing • Resources (images, sounds etc.) 	User Interaction <ul style="list-style-type: none"> • Keyboard • Silverlight touch screen • XNA touch screen with gesture support • XNA guide • Microphone • Camera • Vibration • Sensors <ul style="list-style-type: none"> • Accelerometer • Compass (magnetic sensor) • Gyroscope • Motion • Launchers and Choosers <ul style="list-style-type: none"> • Phone, text and email • web, maps and media • Market, contacts and utilities • Radio • Maps <ul style="list-style-type: none"> • Basic Options • Locking Maps • Location finding <ul style="list-style-type: none"> • Basic and complex collision on locations • Push Pins (images and videos) • Location (movement threshold) • Tiles • Notification services
Error Checking and Messaging	
Run-time Data Storage and retrieval <ul style="list-style-type: none"> • Internal data processing • External data storage and retrieval 	
Windows Phone Application Type Setup <ul style="list-style-type: none"> • Silverlight • Silverlight game • XNA games • XNA games with overlays • Silverlight and XNA games 	
2D XNA Game Elements <ul style="list-style-type: none"> • Sprites • Sprite font • Background (scrolling) • Pervasive system • Game Utilities 	

The game combined treasure hunt (touch screen) and combat (flick screen). A player searches locations with their phone to collect (location type) gold. If a tilt has been generated or the game time expires, then ghosts or monsters attack. The basic principle behind the system was to overlay various layers video, backgrounds, sprite and text, to build up an immersive environment with a player in the centre of both game imagery and game world.

Basing the game in a virtual 3D domain reliant on constant device and location data caused endless problems that would not occur in a conventional game. Sensor and transmission distortion, the calibration from 3D to 2D plains, collision in 3D domain and 2D screen coordinates, sprite generation and the depth perception of moving objects, were all the cause of the number of mathematic books referenced in this project. Developing the game also took far longer than anticipated but based on user feedback, it seems to have the makings of a finished product.

Intention of the Game

The theme of the game was Zombie Gold but that is interchangeable with any other required theme/s and is not the main feature of the game.

The game was intended for testing purposes. To test how a pervasive game experience could be reflected in the game mechanics and interaction for a player, so that a player's requirements and interaction with game elements could be evaluated in the format of a NIST/IUSR Common Industrial Specification for Usability Requirements document.

User Profiling

Standard profiling techniques were used to define the intended demographic for the game. By breaking down several scenarios of use for the players and identifying the main participant task scenarios that would be required to use the application successfully.

As the main interaction in the game revolved around users browsing the application pages and playing a pervasive game. Several task scenarios were developed to identify the main interaction required by a player, these were then used for profiling and testing purposes.

Usability Testing

Usability testing was conducted in the format of a NIST/IUSR Common Industrial Specification for Usability Requirements process which conformed to Level 1, Level 2 and Level 3 of the CISU-R.

The application was evaluated in a usability test against the following criteria Effectiveness, Efficiency, Satisfaction and Pervasive Experience Factor.

In compliance with Level 2 of the CISU-R all preliminary target values were substantiated by a benchmark usability test conducted before the final testing of the application.

The usability levels resulting from the tests all exceed the benchmark values and there was a considerable amount of positive feedback recorded from the test participants.

Effectiveness ratings of 100%, 100%, 95.5%, 100%, 97.5% and 100% all exceed the required levels.

Efficiency data indicated that the correct balance had been achieved, to provide challenging game play at increasingly harder levels, without the game becoming too tedious for a player.

A satisfaction level of 97.5% indicated a very high level of user satisfaction with the application.

A Pervasive Experience Factor level of 86 indicated a fully immersive pervasive experience for the players was achieved. This was also substantiated by test participant feedback that was recorded during the tests.

The usability testing proved to be conclusive and vindicate the project as the stated goals of the Zombie Gold application was as a pervasive game developed to test a cross section of functionality researched in this project on Pervasive Games.

Here the application was not only successful, but based on the usability data provided from the tests, the pervasive engine was also proved effective.

Conclusion

Where to take this in the future?

The engine is a growing entity so it will be continually updated by the addition of more functionality as-and-when more features become available on the platform.

The inclusion of a 3D graphic engine would be an intriguing addition as would be the addition of voice recognition software if the API's become available in the future.

Cross platform development is an option, as now the foundation for the engine is established the ability to develop games on other mobile platforms would be an advantage.

The game has had positive feedback from everyone who has played it and the development of the application was a very interesting process.

So the obvious road to take, is to use the engine to develop Pervasive Games.

References

- Audio File open source library sound bible: Retrieved from <http://soundbible.com>: Repository of licence free sound files.
- Bateson Gregory (1955). A theory of play and fantasy. *Psychiatric Research Reports*, 2 (39), 39-51.
- Bing Maps Account: <https://www.bingmapsportal.com/>: The Bing Maps account centre allows access to account and usage data.
- Bing Maps Registration: https://www.bingmapsportal.com: Link to register as a developer for Bing maps.
- Bing Maps Registration Help: <http://msdn.microsoft.com>: Help on the Bing Maps registration process.
- Bjork S. and Holopainen J.: *Patterns in game design*: Charles River Media.
- Britannica Encyclopaedia: Muybridge Eadweard: British photographer conducted photographic studies of motion: Retrieved from <http://www.britannica.com>.
- Broll Wolfgang, Ohlenburg Jan, Lindt Irma, Herbst Iris and Braun Anne-Kathrin: *Meeting Technology Challenges of Pervasive Augmented Reality Games: Collaborative Virtual and Augmented Environments* Department Fraunhofer FIT: Retrieved from <http://academic.research.microsoft.com>.
- Brooke, J. (1996). *SUS: A “quick and dirty” usability scale*. *Usability Evaluation in Industry*. UK: Taylor and Francis.
- Business Week: Retrieved from <http://www.businessweek.com>.
- Caillouis, R. (1958). *Man, play and games*. Chicago: Free Press.
- Capra Mauricio, Radenkovic Milena, Benford Steve, Oppermann Leif, Drozd Adam and Flintham Martin: *The Multimedia Challenges Raised by Pervasive Games: The University of Nottingham – Mixed Reality Laboratory*: Retrieved from <http://www.academia.edu>
- Common Industry Format (CIF) for usability test reports: Retrieved from <http://www.techstreet.com>
- Cruel-2-B-Kind: Retrieved from <http://www.crulegame.com>.
- Dix, Finley, Abowd and Beale. *Human-computer interaction*: Prentice Hall.
- Federoff, M.: *Improving Games with User Testing: Getting Better Data Earlier*: In *Game Developer’s Magazine*: Retrieved from <http://www.gamasutra.com>.
- Gartner Research: Gartner, Inc.: Retrieved from <http://www.gartner.com>: Gartner, Inc. one of the world's leading information technology research and advisory companies.
- Falk Jennica and Davenport Glorianna: *Live Role-Playing Games: Implications for Pervasive Gaming Play*: Interactive Institute, Göteborg, Sweden & Media Lab Europe, Dublin, Ireland: Retrieved from <http://citeseerx.ist.psu.edu>.
- Hargreaves S.: Retrieved from <http://www.shawnhargreaves.com>: Extensive blog space on gaming related topics.
- Huizinga, J. (1938). *Homo ludens: Versuch einer bestimmung des spielements der kultur*. Published originally in 1944. Ref. English translation (1955): *Homo ludens: A study of play element in culture*. Boston, MA: Beacon Press.
- Humans vs. Zombies (rules): Retrieved from <http://humansvszombies.org>.
- Insectopia: Retrieved from <http://iperg.sics.se>.
- International Data Corporation (IDC): <http://www.idc.com>: Premier global provider of market intelligence, advisory services.
- Jegers Kalle: *Pervasive Game Flow: Understanding Player enjoyment in Pervasive Gaming*: Dept of Informatics, Umeå University, 90751 Umeå, Sweden: Retrieved from <http://umu.diva-portal.org>.

- Kampmann Walther Bo, Associate Professor, Ph.D: Atomic Actions – Molecular Experience: Theory of Pervasive Gaming: Centre for Media Studies University of Southern Denmark: Retrieved from <http://dblp.uni-trier.de>.
- Killer (Killer on the iPhone): Retrieved from <http://killer-game.appspot.com>.
- Killer, commercial game: Retrieved from <http://www.sjgames.com>.
- Koster R.: A Theory of Fun for game Design: Paraglyph Press, 2005
- Lastowka, G. (2007). Rules of play. Ref. rough draft dated at October 16, 2007, retrieved from <http://terranova.blogs.com>: Also presented in AoIR 8.0 conference.in Vancouver, October 2007.
- Mac World: Retrieved from <http://www.macworld.co.uk>.
- Magerkurth Carsten, Cheok Adrian David, Mandryk Regan L.and Nilsen Trod: Pervasive Games: Bringing Computer Entertainment Back to the Real World: Ambiente Germany, Nanyang Technological University Singapore, Simon Fraser University, Vancouver and Human Interface Technology Laboratory Christchurch: Retrieved from <http://academic.research.microsoft.com>.
- Microsoft App Hub: <http://create.msdn.com>: Microsoft's developer portal Windows Phone and Xbox360 development.
- Microsoft Certification Requirements:
 - Application Policies: Retrieved from <http://msdn.microsoft.com>.
 - Additional Requirements for Specific Application Types: Retrieved from <http://msdn.microsoft.com>.
 - Application Submission Requirements: Retrieved from <http://msdn.microsoft.com>.
 - Content Policies: Retrieved from <http://msdn.microsoft.com>.
 - Technical Certification Requirements: Retrieved from <http://msdn.microsoft.com>.
- Microsoft Design Resources for Windows Phone: Retrieved from <http://msdn.microsoft.com>: Resources to aid the design of Metro style Applications.
- Microsoft Download Centre: Retrieved from <http://www.microsoft.com>: Microsoft site to download compilers.
- Microsoft MSDN Library: Retrieved from <http://msdn.microsoft.com>: Microsoft Help.
- Microsoft Windows Live: Retrieved from <http://login.live.com>: Windows Live Hotmail, Messenger, Xbox LIVE and other Microsoft services.
- Mogi: Retrieved from <http://www.geocaching.com>.
- Montola, Stenros and Waern: Pervasive Games: Theory and Design: IGDA.
- Murach Joel: Murach's C# 2010: Murach.
- NISTIR 7432, Common Industry Specification for Usability – Requirements, Annex F (Informative) An example of a Level 1 to 3 requirements document. Retrieved from <http://zing.ncsl.nist.gov>.
- Nokia: www.nokia.com.
- Preece, Rogers and Sharp: Interaction Design: Beyond human-computer interaction: Wiley.
- Pyramid Research: <http://www.pyramidresearch.com>: Pyramid Research an independent researcher for communications, media and technology industries.
- Rafael Ballagas¹ and Steffen P. Walz: REXplorer: Using player-centred iterative design techniques for pervasive game development: Media Computing Group, RWTH Aachen University: Retrieved from <http://www.mics.org>.
- Salen, K., & Zimmerman, E. (2004). The rules of play: Game design fundamentals. Cambridge: MIT Press.
- Salen, K., & Zimmerman, E. (2006). The game design reader: A rules of play anthology. Cambridge: MIT Press.

- Shelby Logan's Run "The Game": Retrieved from <http://www.shelbylogansrun.com>.
- Street Wars: Retrieved from <http://www.streetwars.net>.
- Stroud K. A.: Engineering Mathematics: Macmillan.
- Suits, B. (1990). Construction of a definition. In K. Salen & E. Zimmerman (Eds.), (2006): The game design reader: A rules of play anthology. Cambridge: MIT Press.
- System Usability Scale (SUS), © Digital Equipment Corporation, 1986.
- Thomas and Finney: Calculus and Analytic Geometry: Addison Wesley.
- Van Verth James M. and Bishop Lars M.: Essential Mathematics for Games and Interactive Applications: Elsevier.

Definitions:

- Anonymous method: Anonymous methods enable you to omit the parameter list. This means that an anonymous method can be converted to delegates with a variety of signatures. This is not possible with lambda expressions. [Definition Microsoft MSDN: <http://msdn.microsoft.com>]
- HVGA: Half size Video Graphics Array
- QVGA: Quarter Video Graphics Array, display use in mobile devices.
- URI: Uniform Resource Identifier, a string of characters used to identify a resource.
- UI: User Interface, the page displayed on the device allowing user interaction with the application.
- XNA: Xbox New Architecture or XNA is Not an Acronym depending on your point of view, Microsoft's game libraries for the Xbox and Phone platforms.

Appendix A: Isolated Storage Testing

Each test was carried out 3 times

1. The first time the app was loaded onto the device
2. Application was re-opened from the Menu bar
3. Phone was switched off and on then the application was re-opened from the Menu bar a second time

The following (usable) application data sizes (base data not packaged) were used:

80 bits / 10 bytes, 1,664 bits / 208 bytes, 14,528 bits / 1,816 bytes and 144,128 bits / 18016 bytes

320 bits / 40 bytes, 6,656 bits / 832 bytes, 145,280 bits / 18,160 bytes and 14,412,800 bits / 1,801,600 bytes

Test Data

Method	None / Benchmark					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Launching	1.1.a	0	963	977	14	Benchmark test for system speed, initializes the application.
	1.1.b	0	414	416	2	Benchmark test for system speed
	1.1.c	0	39	41	2	Benchmark test for system speed
Test Application 2 Application_Launching	1.2.a	0	245	261	16	Benchmark test for system speed, initializes the application.
	1.2.b	0	733	736	3	Benchmark test for system speed
	1.2.c	0	895	897	2	Benchmark test for system speed
Test Application 3 Application_Launching	1.3.a	0	837	853	16	Benchmark test for system speed, initializes the application.
	1.3.b	0	854	856	2	Benchmark test for system speed
	1.3.c	0	49	51	2	Benchmark test for system speed
Test Application 4 Application_Launching	1.4.a	0	513	527	14	Benchmark test for system speed, initializes the application.
	1.4.b	0	156	158	2	Benchmark test for system speed
	1.4.c	0	540	542	2	Benchmark test for system speed

Method	public static ConfigData InitializeISData(string ISFolder, string ISFile)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Launching	2.1.a	80	310	398	88	Needs to build the IS File
	2.1.b	80	955	976	21	
	2.1.c	80	397	410	13	
Test Application 2 Application_Launching	2.2.a	1664	42	134	92	Needs to build the IS File
	2.2.b	1664	170	186	16	
	2.2.c	1664	674	689	15	
Test Application 3 Application_Launching	2.3.a	14528	641	736	95	Needs to build the IS File
	2.3.b	14528	47	65	18	
	2.3.c	14528	202	219	17	
Test Application 4 Application_Launching	2.4.a	144128	120	257	137	Needs to build the IS File
	2.4.b	144128	915	929	14	
	2.4.c	144128	408	420	12	

Method	public static ConfigData InitializeISData(string ISFolder, string ISFile, ConfigData defaultData)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Launching	3.1.a	80	185	330	145	Needs to build the IS File
	3.1.b	80	754	767	13	
	3.1.c	80	455	469	14	
Test Application 2 Application_Launching	3.2.a	1664	119	205	86	Needs to build the IS File
	3.2.b	1664	704	718	14	
	3.2.c	1664	583	595	12	
Test Application 3 Application_Launching	3.3.a	14528	141	232	91	Needs to build the IS File
	3.3.b	14528	499	513	14	
	3.3.c	14528	623	637	14	
Test Application 4 Application_Launching	3.4.a	144128	0	94	94	Needs to build the IS File
	3.4.b	144128	201	218	17	
	3.4.c	144128	959	973	14	

Method	public static ConfigData InitializeISData(string ISFolder, string ISFile, string fileLink)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Launching	4.1.a	80	690	770	80	Needs to build the IS File
	4.1.b	80	30	43	13	
	4.1.c	80	848	860	12	
Test Application 2 Application_Launching	4.2.a	1664	123	366	243	Needs to build the IS File
	4.2.b	1664	918	940	22	
	4.2.c	1664	796	814	18	
Test Application 3 Application_Launching	4.3.a	14528	700	759	59	Needs to build the IS File
	4.3.b	14528	360	376	16	
	4.3.c	14528	481	496	15	
Test Application 4 Application_Launching	4.4.a	144128	518	778	260	Needs to build the IS File
	4.4.b	144128	158	175	17	
	4.4.c	144128	980	996	16	

Method	public static ConfigData InitializeISDataSerialized(string ISFolder, string ISFile, string fileLink)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Launching	5.1.a	80	41	320	279	Needs to build the IS File
	5.1.b	80	254	275	21	
	5.1.c	80	481	504	23	
Test Application 2 Application_Launching	5.2.a	1664	174	458	284	Needs to build the IS File
	5.2.b	1664	341	365	24	
	5.2.c	1664	462	483	21	
Test Application 3 Application_Launching	5.3.a	14528	347	634	287	Needs to build the IS File
	5.3.b	14528	624	656	32	
	5.3.c	14528	256	282	26	
Test Application 4 Application_Launching	5.4.a	144128	456	768	312	Needs to build the IS File
	5.4.b	144128	606	637	31	
	5.4.c	144128	775	806	31	

Method	public static List<ConfigData> InitializeISListDataSerialized(string ISFolder, string ISFile, string fileLink)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Launching	6.1.a	320	154	459	305	Needs to build the IS File
	6.1.b	320	156	201	45	
	6.1.c	320	332	367	35	
Test Application 2 Application_Launching	6.2.a	6656	38	339	301	Needs to build the IS File
	6.2.b	6656	960	991	31	
	6.2.c	6656	356	384	28	
Test Application 3 Application_Launching	6.3.a	145280	53	413	360	Needs to build the IS File
	6.3.b	145280	162	228	66	
	6.3.c	145280	727	790	63	
Test Application 4 Application_Launching	6.4.a	14412800	1	1223	1222	Needs to build the IS File
	6.4.b	14412800	621	1254	633	
	6.4.c	14412800	906	1536	630	

Method	public static ConfigData InitializeSDDData(string settingsKey)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Launching	7.1.a	80	163	596	433	Needs to build the IS File
	7.1.b	80	563	602	39	
	7.1.c	80	188	222	34	
Test Application 2 Application_Launching	7.2.a	1664	517	945	428	Needs to build the IS File
	7.2.b	1664	199	234	35	
	7.2.c	1664	441	480	39	
Test Application 3 Application_Launching	7.3.a	14528	981	1419	438	Needs to build the IS File
	7.3.b	14528	596	638	42	
	7.3.c	14528	106	144	38	
Test Application 4 Application_Launching	7.4.a	144128	226	665	439	Needs to build the IS File
	7.4.b	144128	527	571	44	
	7.4.c	144128	760	802	42	

Method	public static ConfigData InitializeSDDData(string settingsKey, ConfigData defaultData)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Launching	8.1.a	80	300	730	430	Needs to build the IS File
	8.1.b	80	928	965	37	
	8.1.c	80	924	959	35	
Test Application 2 Application_Launching	8.2.a	1664	456	888	432	Needs to build the IS File
	8.2.b	1664	296	332	36	
	8.2.c	1664	171	205	34	
Test Application 3 Application_Launching	8.3.a	14528	409	845	436	Needs to build the IS File
	8.3.b	14528	813	856	43	
	8.3.c	14528	161	203	42	
Test Application 4 Application_Launching	8.4.a	144128	443	874	431	Needs to build the IS File
	8.4.b	144128	450	493	43	
	8.4.c	144128	643	686	43	

Method	public static ConfigData InitializeSDData(string settingsKey, string fileLink)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Launching	9.1.a	80	686	1240	554	Needs to build the IS File
	9.1.b	80	85	123	38	
	9.1.c	80	218	254	36	
Test Application 2 Application_Launching	9.2.a	1664	188	760	572	Needs to build the IS File
	9.2.b	1664	206	243	37	
	9.2.c	1664	208	243	35	
Test Application 3 Application_Launching	9.3.a	14528	799	1379	580	Needs to build the IS File
	9.3.b	14528	709	749	40	
	9.3.c	14528	835	875	40	
Test Application 4 Application_Launching	9.4.a	144128	405	996	591	Needs to build the IS File
	9.4.b	144128	460	503	43	
	9.4.c	144128	484	527	43	

Method	public static List<ConfigData> InitializeSDListData(string settingsKey, string fileLink)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Launching	10.1.a	320	286	877	591	Needs to build the IS File
	10.1.b	320	756	803	47	
	10.1.c	320	475	519	44	
Test Application 2 Application_Launching	10.2.a	6656	766	1366	600	Needs to build the IS File
	10.2.b	6656	185	237	52	
	10.2.c	6656	792	836	44	
Test Application 3 Application_Launching	10.3.a	145280	24	648	624	Needs to build the IS File
	10.3.b	145280	759	881	122	
	10.3.c	145280	929	1028	99	
Test Application 4 Application_Launching	10.4.a	14412800	562	2137	1575	Needs to build the IS File
	10.4.b	14412800	633	1265	632	
	10.4.c	14412800	2	618	616	

Method	None / Benchmark					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Activated	11.1.a	0	573	579	6	Benchmark test for system speed
	11.1.b	0	972	973	1	Benchmark test for system speed
	11.1.c	0	619	620	1	Benchmark test for system speed
Test Application 2 Application_Activated	11.2.a	0	279	283	4	Benchmark test for system speed
	11.2.b	0	648	649	1	Benchmark test for system speed
	11.2.c	0	130	131	1	Benchmark test for system speed
Test Application 3 Application_Activated	11.3.a	0	623	630	7	Benchmark test for system speed
	11.3.b	0	158	159	1	Benchmark test for system speed
	11.3.c	0	575	576	1	Benchmark test for system speed
Test Application 4 Application_Activated	11.4.a	0	18	25	7	Benchmark test for system speed
	11.4.b	0	618	619	1	Benchmark test for system speed
	11.4.c	0	74	75	1	Benchmark test for system speed

Method	public static ConfigData LoadCSDData(string stateKey)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Activated	12.1.a	80	245	260	15	
	12.1.b	80	902	904	2	
	12.1.c	80	827	828	1	
Test Application 2 Application_Activated	12.2.a	1664	324	333	9	
	12.2.b	1664	248	250	2	
	12.2.c	1664	122	123	1	
Test Application 3 Application_Activated	12.3.a	14528	710	717	7	
	12.3.b	14528	416	417	1	
	12.3.c	14528	378	380	2	
Test Application 4 Application_Activated	12.4.a	144128	60	68	8	
	12.4.b	144128	308	312	4	
	12.4.c	144128	191	192	1	

Method	public static ConfigData LoadCSDData(string stateKey, string fileLink)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Activated	13.1.a	80	672	784	112	Needs to load defaults
	13.1.b	80	299	308	9	
	13.1.c	80	991	1000	9	
Test Application 2 Application_Activated	13.2.a	1664	274	387	113	Needs to load defaults
	13.2.b	1664	106	116	10	
	13.2.c	1664	337	347	10	
Test Application 3 Application_Activated	13.3.a	14528	885	1015	130	Needs to load defaults
	13.3.b	14528	37	55	18	
	13.3.c	14528	826	845	19	
Test Application 4 Application_Activated	13.4.a	144128	909	1070	161	Needs to load defaults
	13.4.b	144128	980	998	18	
	13.4.c	144128	74	92	18	

Method	public static List<ConfigData> LoadCSListData(string stateKey)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Activated	14.1.a	320	931	938	7	
	14.1.b	320	31	32	1	
	14.1.c	320	810	811	1	
Test Application 2 Application_Activated	14.2.a	6656	82	90	8	
	14.2.b	6656	496	497	1	
	14.2.c	6656	416	417	1	
Test Application 3 Application_Activated	14.3.a	145280	191	198	7	
	14.3.b	145280	806	807	1	
	14.3.c	145280	584	585	1	
Test Application 4 Application_Activated	14.4.a	14412800	627	635	8	
	14.4.b	14412800	547	548	1	
	14.4.c	14412800	529	530	1	

Method	public static List<ConfigData> LoadCSListData(string stateKey, string fileLink)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Activated	15.1.a	320	474	612	138	Needs to load defaults
	15.1.b	320	54	55	1	
	15.1.c	320	757	758	1	
Test Application 2 Application_Activated	15.2.a	6656	238	384	146	Needs to load defaults
	15.2.b	6656	396	399	3	
	15.2.c	6656	62	64	2	
Test Application 3 Application_Activated	15.3.a	145280	636	845	209	Needs to load defaults
	15.3.b	145280	375	377	2	
	15.3.c	145280	988	990	2	
Test Application 4 Application_Activated	15.4.a	14412800	611	1373	762	Needs to load defaults
	15.4.b	14412800	597	599	2	
	15.4.c	14412800	970	974	4	

Method	public static ConfigData LoadISData(string ISFolder, string ISFile)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Activated	16.1.a	80	211	230	19	
	16.1.b	80	674	687	13	
	16.1.c	80	199	207	8	
Test Application 2 Application_Activated	16.2.a	1664	854	873	19	
	16.2.b	1664	735	748	13	
	16.2.c	1664	474	483	9	
Test Application 3 Application_Activated	16.3.a	14528	675	694	19	
	16.3.b	14528	25	38	13	
	16.3.c	14528	107	118	11	
Test Application 4 Application_Activated	16.4.a	144128	309	330	21	
	16.4.b	144128	296	310	14	
	16.4.c	144128	123	142	19	

Method	public static ConfigData LoadISDataSerialized(string ISFolder, string ISFile)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Activated	17.1.a	80	263	353	90	
	17.1.b	80	391	407	16	
	17.1.c	80	193	207	14	
Test Application 2 Application_Activated	17.2.a	1664	255	307	52	
	17.2.b	1664	990	1006	16	
	17.2.c	1664	166	181	15	
Test Application 3 Application_Activated	17.3.a	14528	251	311	60	
	17.3.b	14528	670	693	23	
	17.3.c	14528	474	497	23	
Test Application 4 Application_Activated	17.4.a	144128	737	803	66	
	17.4.b	144128	306	330	24	
	17.4.c	144128	444	468	24	

Method	public static List<ConfigData> LoadISListDataSerialized(string ISFolder, string ISFile)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Activated	18.1.a	320	210	240	30	
	18.1.b	320	743	759	16	
	18.1.c	320	488	505	17	
Test Application 2 Application_Activated	18.2.a	6656	783	807	24	
	18.2.b	6656	582	599	17	
	18.2.c	6656	59	76	17	
Test Application 3 Application_Activated	18.3.a	145280	371	429	58	
	18.3.b	145280	53	139	86	
	18.3.c	145280	942	1031	89	
Test Application 4 Application_Activated	18.4.a	14412800	349	1015	666	
	18.4.b	14412800	942	1582	640	
	18.4.c	14412800	194	830	636	

Method	public static ConfigData LoadSDDData(string settingsKey)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Activated	19.1.a	80	519	524	5	
	19.1.b	80	123	124	1	
	19.1.c	80	499	500	1	
Test Application 2 Application_Activated	19.2.a	1664	981	987	6	
	19.2.b	1664	554	555	1	
	19.2.c	1664	345	346	1	
Test Application 3 Application_Activated	19.3.a	14528	944	951	7	
	19.3.b	14528	533	534	1	
	19.3.c	14528	815	816	1	
Test Application 4 Application_Activated	19.4.a	144128	689	697	8	
	19.4.b	144128	365	366	1	
	19.4.c	144128	969	970	1	

Method	public static List<ConfigData> LoasSDListData(string settingsKey)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Activated	20.1.a	320	720	726	6	
	20.1.b	320	696	702	6	
	20.1.c	320	785	786	1	
Test Application 2 Application_Activated	20.2.a	6656	304	312	8	
	20.2.b	6656	965	966	1	
	20.2.c	6656	840	841	1	
Test Application 3 Application_Activated	20.3.a	145280	205	212	7	
	20.3.b	145280	851	852	1	
	20.3.c	145280	614	615	1	
Test Application 4 Application_Activated	20.4.a	14412800	373	379	6	
	20.4.b	14412800	795	796	1	
	20.4.c	14412800	328	329	1	

Method	None / Benchmark					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Deactivated	21.1.a	0	417	424	7	Benchmark test for system speed
	21.1.b	0	300	301	1	Benchmark test for system speed
	21.1.c	0	424	425	1	Benchmark test for system speed
Test Application 2 Application_Deactivated	21.2.a	0	758	768	10	Benchmark test for system speed
	21.2.b	0	346	347	1	Benchmark test for system speed
	21.2.c	0	149	150	1	Benchmark test for system speed
Test Application 3 Application_Deactivated	21.3.a	0	161	166	5	Benchmark test for system speed
	21.3.b	0	507	508	1	Benchmark test for system speed
	21.3.c	0	33	34	1	Benchmark test for system speed
Test Application 4 Application_Deactivated	21.4.a	0	498	507	9	Benchmark test for system speed
	21.4.b	0	11	12	1	Benchmark test for system speed
	21.4.c	0	887	888	1	Benchmark test for system speed

Method	public static void SaveCSDData(string stateKey, ConfigData CSObject)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Deactivated	22.1.a	80	731	744	13	
	22.1.b	80	448	450	2	
	22.1.c	80	761	762	1	
Test Application 2 Application_Deactivated	22.2.a	1664	170	179	9	
	22.2.b	1664	396	398	2	
	22.2.c	1664	327	328	1	
Test Application 3 Application_Deactivated	22.3.a	14528	163	176	13	
	22.3.b	14528	993	995	2	
	22.3.c	14528	833	835	2	
Test Application 4 Application_Deactivated	22.4.a	144128	448	458	10	
	22.4.b	144128	574	575	1	
	22.4.c	144128	187	188	1	

Method	public static void SaveCSDData(string stateKey, List<ConfigData> CSObjectList)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Deactivated	23.1.a	320	72	82	10	
	23.1.b	320	15	16	1	
	23.1.c	320	888	889	1	
Test Application 2 Application_Deactivated	23.2.a	6656	472	482	10	
	23.2.b	6656	824	825	1	
	23.2.c	6656	294	295	1	
Test Application 3 Application_Deactivated	23.3.a	145280	670	678	8	
	23.3.b	145280	977	978	1	
	23.3.c	145280	631	632	1	
Test Application 4 Application_Deactivated	23.4.a	14412800	174	181	7	
	23.4.b	14412800	776	777	1	
	23.4.c	14412800	263	264	1	

Method	public static void SaveISData(string ISFolder, string ISFile, ConfigData ISObject)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Deactivated	24.1.a	80	711	754	43	
	24.1.b	80	531	565	34	
	24.1.c	80	278	327	49	
Test Application 2 Application_Deactivated	24.2.a	1664	120	141	21	
	24.2.b	1664	646	660	14	
	24.2.c	1664	701	713	12	
Test Application 3 Application_Deactivated	24.3.a	14528	276	295	19	
	24.3.b	14528	247	258	11	
	24.3.c	14528	313	326	13	
Test Application 4 Application_Deactivated	24.4.a	144128	909	933	24	
	24.4.b	144128	501	514	13	
	24.4.c	144128	136	153	17	

Method	public static void SaveISDataSerialized(string ISFolder, string ISFile, ConfigData ISObject)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Deactivated	25.1.a	80	649	884	235	
	25.1.b	80	490	511	21	
	25.1.c	80	405	420	15	
Test Application 2 Application_Deactivated	25.2.a	1664	743	936	193	
	25.2.b	1664	902	916	14	
	25.2.c	1664	442	464	22	
Test Application 3 Application_Deactivated	25.3.a	14528	666	893	227	
	25.3.b	14528	783	810	27	
	25.3.c	14528	585	607	22	
Test Application 4 Application_Deactivated	25.4.a	144128	968	1174	206	
	25.4.b	144128	695	728	33	
	25.4.c	144128	228	257	29	

Method	public static void SaveISDataSerialized(string ISFolder, string ISFile, List<ConfigData> ISObjectList)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Deactivated	26.1.a	320	775	811	36	
	26.1.b	320	877	891	14	
	26.1.c	320	644	661	17	
Test Application 2 Application_Deactivated	26.2.a	6656	449	483	34	
	26.2.b	6656	587	601	14	
	26.2.c	6656	70	85	15	
Test Application 3 Application_Deactivated	26.3.a	145280	945	999	54	
	26.3.b	145280	164	243	79	
	26.3.c	145280	220	303	83	
Test Application 4 Application_Deactivated	26.4.a	14412800	523	999	476	
	26.4.b	14412800	857	1563	706	
	26.4.c	14412800	406	1093	687	

Method	public static void SaveSDDData(string settingsKey, ConfigData SDOject)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Deactivated	27.1.a	80	444	511	67	
	27.1.b	80	561	584	23	
	27.1.c	80	733	755	22	
Test Application 2 Application_Deactivated	27.2.a	1664	175	187	12	
	27.2.b	1664	900	922	22	
	27.2.c	1664	179	205	26	
Test Application 3 Application_Deactivated	27.3.a	14528	991	1011	20	
	27.3.b	14528	696	726	30	
	27.3.c	14528	959	1015	56	
Test Application 4 Application_Deactivated	27.4.a	144128	932	968	36	
	27.4.b	144128	293	339	46	
	27.4.c	144128	839	909	70	

Method	public static void SaveSDData(string settingsKey, List<ConfigData> SDOBJECTList)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Deactivated	28.1.a	320	684	725	41	
	28.1.b	320	768	799	31	
	28.1.c	320	296	335	39	
Test Application 2 Application_Deactivated	28.2.a	6656	472	586	114	
	28.2.b	6656	104	302	198	
	28.2.c	6656	323	397	74	
Test Application 3 Application_Deactivated	28.3.a	145280	464	721	257	
	28.3.b	145280	805	1118	313	
	28.3.c	145280	862	1194	332	
Test Application 4 Application_Deactivated	28.4.a	14412800	831	1755	924	
	28.4.b	14412800	466	1410	944	
	28.4.c	14412800	888	1807	919	

Method	None / Benchmark					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Application 1 Application_Closing	29.1.a	0	444	451	7	Benchmark test for system speed
	29.1.b	0	299	300	1	Benchmark test for system speed
	29.1.c	0	788	789	1	Benchmark test for system speed
Test Application 2 Application_Closing	29.2.a	0	111	121	10	Benchmark test for system speed
	29.2.b	0	87	88	1	Benchmark test for system speed
	29.2.c	0	204	205	1	Benchmark test for system speed
Test Application 3 Application_Closing	29.3.a	0	778	783	5	Benchmark test for system speed
	29.3.b	0	141	142	1	Benchmark test for system speed
	29.3.c	0	768	769	1	Benchmark test for system speed
Test Application 4 Application_Closing	29.4.a	0	334	343	9	Benchmark test for system speed
	29.4.b	0	307	308	1	Benchmark test for system speed
	29.4.c	0	814	815	1	Benchmark test for system speed

Method	public static void SaveISData(string ISFolder, string ISFile, ConfigData ISOBJECT)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Applicaion 1 Application_ Closing	30.1.a	80	500	543	43	
	30.1.b	80	234	268	34	
	30.1.c	80	77	126	49	
Test Application 2 Application_Closing	30.2.a	1664	222	243	21	
	30.2.b	1664	511	525	14	
	30.2.c	1664	3	15	12	
Test Application 3 Application_Closing	30.3.a	14528	456	475	19	
	30.3.b	14528	243	254	11	
	30.3.c	14528	990	1003	13	
Test Application 4 Application_Closing	30.4.a	144128	806	830	24	
	30.4.b	144128	657	670	13	
	30.4.c	144128	275	288	13	

Method	public static void SaveISDataSerialized(string ISFolder, string ISFile, ConfigData ISObject)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Applicaion 1 Application_ Closing	31.1.a	80	233	468	235	Needs to load defaults
	31.1.b	80	122	143	21	
	31.1.c	80	44	59	15	
Test Application 2 Application_Closing	31.2.a	1664	337	530	193	Needs to load defaults
	31.2.b	1664	255	269	14	
	31.2.c	1664	77	99	22	
Test Application 3 Application_Closing	31.3.a	14528	743	970	227	Needs to load defaults
	31.3.b	14528	601	628	27	
	31.3.c	14528	775	797	22	
Test Application 4 Application_Closing	31.4.a	144128	588	794	206	Needs to load defaults
	31.4.b	144128	198	231	33	
	31.4.c	144128	201	230	29	

Method	public static void SaveISDataSerialized(string ISFolder, string ISFile, List<ConfigData> ISObjectList)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Applicaion 1 Application_ Closing	32.1.a	320	36	72	36	Needs to load defaults
	32.1.b	320	610	624	14	
	32.1.c	320	1	18	17	
Test Application 2 Application_Closing	32.2.a	6656	764	798	34	Needs to load defaults
	32.2.b	6656	555	569	14	
	32.2.c	6656	30	45	15	
Test Application 3 Application_Closing	32.3.a	145280	66	120	54	Needs to load defaults
	32.3.b	145280	365	444	79	
	32.3.c	145280	109	212	103	
Test Application 4 Application_Closing	32.4.a	14412800	51	527	476	Needs to load defaults
	32.4.b	14412800	11	717	706	
	32.4.c	14412800	205	892	687	

Method	public static void SaveSDDData(string settingsKey, ConfigData SDOObject)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Applicaion 1 Application_ Closing	33.1.a	80	401	428	27	
	33.1.b	80	277	300	23	
	33.1.c	80	444	466	22	
Test Application 2 Application_Closing	33.2.a	1664	568	580	12	
	33.2.b	1664	113	135	22	
	33.2.c	1664	902	928	26	
Test Application 3 Application_Closing	33.3.a	14528	479	499	20	
	33.3.b	14528	579	609	30	
	33.3.c	14528	358	504	146	
Test Application 4 Application_Closing	33.4.a	144128	853	889	36	
	33.4.b	144128	178	224	46	
	33.4.c	144128	151	221	70	

Method	public static void SaveSDDData(string settingsKey, List<ConfigData> SDOBJECTList)					
Application / Event	Test No	Data Size in bits	Start count	End count	Speed	Notes
Test Applicaion 1 Application_ Closing	34.1.a	320	644	685	41	
	34.1.b	320	142	173	31	
	34.1.c	320	141	180	39	
Test Application 2 Application_Closing	34.2.a	6656	7	121	114	
	34.2.b	6656	200	398	198	
	34.2.c	6656	635	709	74	
Test Application 3 Application_Closing	34.3.a	145280	261	518	257	
	34.3.b	145280	376	689	313	
	34.3.c	145280	19	351	332	
Test Application 4 Application_Closing	34.4.a	14412800	5	929	924	
	34.4.b	14412800	364	1308	944	
	34.4.c	14412800	407	1326	919	

Appendix B: Pervasive Game Engine

The following applications are code samples developed while researching the Pervasive Game Engine.

If the directories are changed (folder locations) and they are opened in the IDE, they will need to be rebuilt to set the project dependencies.

Please note these applications are not edited (or extensively commented) and were used to demonstrate elements of XAML and C# code engine:

Basic Silverlight Applications	Notes
001 HelloPhone	Basic Hello World and using Blend to design the interface.
002 SplashScreen	Displays a Splashscreen for 3 seconds
003 DamageCalc	Basic tax calculator using the Text Change Event
004 OrientationSample	Orientation aware page example
005 OrientationGrid	Orientation aware page example using grid format
006 Navigation	Page navigation, back button override and back button navigation
007 NavigationAndControls	Page navigation and objects, images, music, video and web link
008 PanoramaApplication	Basic Panorama page setup
009 ApplicationBarBlend	Basic Application Bar set up via Blend
010 ApplicationBarXAML	Application Bar set up via XAML with several options, close, opacity, menu and next
011 ApplicationBarSampleC#	Application Bar set up via C# with several options, close, opacity, menu and next
012 ControlsTrayAppBar	Page navigation and Application Bar, System Tray, objects, images, music and video
013 Events	Template for setting up events in various formats

Error Checking & Messaging	Notes
014 Messages	Simple debugging and messages
015 ErrorPage	Directs unhandled exception data to a display page
016 DebugTest	Shows the use of MessageBox and Logger Class functionality for debugging
017 CustomMessageBox	Custom Message Box utilising the Generic.xaml, NotificationBoxCommand.cs and NotificationBox.cs utility files

Data Storage	Notes
018 PassingDataBetweenPages	URI and Query option used to pass a string between pages
019 PassingDataGlobally	Global class used to pass data between pages
020 DataBinding	A simple example of data binding
021 AppLifecycle debug test	Debug test for Application Lifecycle event firing order
022 Utils Template	Code stub for using the data processing methods StorageUtils.cs
023 Speed Test 1	Speed tests for data blocks 80 bits (10 bytes) and 320 bits (40 bytes)
024 Speed Test 2	Speed tests for data blocks 1,664 bits (208 bytes) and 6,656 bits (832 bytes)
025 Speed Test 3	Speed tests for data blocks 14,528 bits (1,816 bytes) and 145,280 bits (18,160 bytes)
026 Speed Test 4	Speed tests for data blocks 144,128 bits (18016 bytes) and 14,412,800 bits (1,801,600 bytes)

User Interaction with the Phone	Notes
027 Keyboards	Displays the basic Keyboard options
028 Touch	Basic Silverlight touch input, reports the position of the touch in the ManipulationDelta event
029 Touch Rectangles	Basic XNA touch, reports pressed, moved, released and position. Exit on rectangle collision.
030 Touch Enumeration	XNA touch input, reports all gesture types also, pressed, moved, released and position states.
031 GuidelineInput	Activates the keyboard via touch input using the Guide text input mechanism.
032 MicrophoneTest	Simple Microphone application records and plays microphone input. No save feature.
033 CameraGrab	Sets up the camera data as a background Silverlight version.

User Interaction with the Phone	Notes
034 CameraGrabXNA	Sets up the camera data as a background XNA version.
035 BasicCamera	Basic camera application (updated MS example).
036 BasicVideo	Basic camera video application (updated MS example).
037 Vibrate	Vibrates the phone for one second.
038 AccelerometerTest	Basic test of the Accelerometer, returns X, Y and Z data.
039 AccelerometerApp	Simple level and seismograph app based on Accelerometer data.
040 AccelerometerXNA	Accelerometer XNA movement, five movement types changeable via touching the screen.
041 MapMotion	Fast test to put Accelerometer motion on a map (needs more work).
042 SensorTest	Gets data from the Accelerometer, Compass and Gyroscope sensors.
043 BasicMotion	Get motion data from the sensors and outputs the raw data.
044 MotionTest	Gets data via the Motion API's, Attitude (Yaw, Pitch and Roll) and Acceleration X, Y and Z.
045 AR_Test	Basic Augmented Reality test bed based on the Motion API's.
046 LaunchersChoosers	Implements the following Launchers and Choosers API's: <ul style="list-style-type: none"> • Phone Tasks, make a phone call, return and save a phone number. • Text Task, send a text message. • Email Tasks, send an email, return and save an email address. • Web Tasks, launch a browser or web search. • Map Tasks, find location and directions. • Media Tasks, take pictures, select a photo, launch media player and save ringtones. • Marketplace Tasks, launch marketplace hub and display results, details and review pages. • Contacts Tasks, save / return contacts, setup links, share status messages and invite players. • Utility Tasks, edit connection settings.
047 Radio	Basic Radio application (need earphones), search, tune, region, play and stop (work in progress).
048 Maps	Implements the following Bing Maps and Location Services API's: <ul style="list-style-type: none"> • Basic Options, sets mode, zoom, scale, language and display options on maps. • Lock Maps, locks them movement of maps together. • Find Location, finds locations based on GeoCoordinates. • Basic Collision, basic collision based on GeoCoordinates and a defined search rectangle. • Collision, validated collision based on the geometry of the map, adds testing overlays. • Push Pins, adding and removing to locations push pins (events / images), images and videos. • Track Phone, displays the location of the phone on a map using Location and Map services.
049 ChangingTiles	Process Application and Secondary Tiles, also sets up a Tile Scheduler.
050 SendToast	Server side application that sends a Toast Notification, the ChannelUri needs to be hard coded in the Uri input box (this can be found by running the 051 ToastNotificationClient application) the current one is "http://sn1.notify.live.net/throttledthirdparty/01.00/AAE1G7k6-MHNR7E1kTnLWIFBAGAAAAADBgAAAAQUZm52OjUwQzQwOUM5NjUzQ0QxQUY"
051 ToastNotificationClient	Client side application that receives Toast Notifications, when the Toast is clicked it will go to specified page in the application. Toast data is also displayed in the application. Note: To be allocated a ChannelUri (push notification channel) and use the service the phone need to be topped up (in credit) so data can be downloaded and the service registered.
052 SendTile	Server side application that sends a Tile Notification, the ChannelUri needs to be hard coded in the Uri input box (this can be found by running the 053 TileNotificationClient application) the current one is http://sn1.notify.live.net/throttledthirdparty/01.00/AAE1G7k6-MHNR7E1kTnLWIFBAGAAAAADBgAAAAQUZm52OjUwQzQwOUM5NjUzQ0QxQUY Use Red.jpg for the Front Background Image and Blue.jpg for the Back Background Image.
053 TileNotificationClient	Client side application that receives Tile Notifications, when the Tile is clicked it will go to specified page in the application. Tile data is updated on both the front and back of the tile. Note: To be allocated a ChannelUri (push notification channel) and use the service the phone need to be topped up (in credit) so data can be downloaded and the service registered.
054 SendRaw	Server side application that sends a Raw Notification, the ChannelUri needs to be hard coded in the Uri input box (this can be found by running the 055 RawNotificationClient application) the current one is http://sn1.notify.live.net/throttledthirdparty/01.00/AAE1G7k6-MHNR7E1kTnLWIFBAGAAAAADBgAAAAQUZm52OjUwQzQwOUM5NjUzQ0QxQUY

User Interaction with the Phone	Notes
055 RawNotificationClient	Client side application that receives Raw Notifications when it is running. Note: To be allocated a ChannelUri (push notification channel) and use the service the phone need to be topped up (in credit) so data can be downloaded and the service registered.
056 MessagingGame	Work in progress (see Set-up table for set-up requirements), currently supported: <ul style="list-style-type: none"> • GameServer, registers the phone and sends and logs weather Toast, Tile and Raw messages. • GameClient, registers with the server and receives updated weather Toast, Tile and Raw messages. Look at the registration process when you have time it is very clumsy and expensive in both resources and connectivity, the Messaging service works well though.

Application Types	Notes
057 WindowsPhoneApps	<p>Implements the following Windows Phone Application (Silverlight) options:</p> <ul style="list-style-type: none"> • App Type - Basic App Events, basic Windows Phone Application showing Page and PhoneApplicationPage events sequence. • App Type - Basic Game Events, depicts the basic flow of control through the life cycle of a phone game page, utilising a game loop system. • App Type - Basic Game Events, uses a different system to depict the basic flow of control through the life cycle of a phone game page, utilising a game update system. • Sound – Basic Sound in an App, uses the basic sound API's to play sound in a normal Application Framework. • Sound – XNA Sound in Game loop, uses the XNA sound API's to play sound in a Game Loop normal Application Framework. There are different methods of processing a sound tested, stream, SoundEffect, SoundEffectInstance, StreamResourceInfo, 3D sound and AudioEmitter / AudioListener are used.
058 AlarmClock	<p>Basic alarm clock with the following options:</p> <ul style="list-style-type: none"> • Page 1 displays an alarm clock. • Page 2 saves / loads any recorded sound for the alarm and sets the alarm time. • Page 3 records the sound and plays it back. <p>Very quick test of the sound and microphone API's using the Games loop and Silverlight Application frameworks. Work in progress, some bugs in the logic and a hacked version of Isolated Storage just to get it going (sound buffer appending on IS would need looking at). The main functionality is correct sound in / out via microphone and speakers which is what I wanted to test, the rest is just a bit of fun that needs some tidying up (when I have time).</p>
059 XNAGameSetup	Basic template for the XNA game framework, "XNA Games Studio (4.0) > Windows Phone Game (4.0)" build.
060 XNAGameSetupOverlays	Basic template for the XNA game framework with an overlay. "XNA Games Studio (4.0) > Windows Phone Game (4.0)" build set up then a Game Component added and edited into a derived class from the DrawableGameComponent class. Also with the GameComponent.LoadContent, GameComponent.Update, DrawableGameComponent.Draw and DrawableGameComponent.UnloadContent methods overridden.
061 SilverlightXNAApplication	Basic template for the Silverlight and XNA Application framework "Silverlight for Windows Phone > Windows Phone Silverlight and XNA Application or XNA Game studio (4.0) > Windows Phone Silverlight and XNA Application" build.

Common XNA Game Elements	Notes
062 SpriteTest	<p>Sprite 1: Draw opaque sprite test: A sprite is displayed to the screen.</p> <p>Sprite 2: Draw transparent sprite test: A sprite is displayed to the screen and the background transparency is displayed correctly.</p> <p>Sprite 3: Draw scaling sprite test: A sprite is displayed to the screen with its size set by a scaling rectangle.</p> <p>Sprite 4: Draw rotating sprite test: A sprite is displayed to the screen and rotated around its centre (sprite origin centre).</p> <p>Sprite 5: Draw animated sprite test: Two sprites are displayed animated to the screen one transparent and the other opaque.</p> <p>Sprite 6: Draw complex animated sprite test: Five sprites are displayed animated to the screen in different formats (from cell array textures).</p> <p>Sprite Font 7: Draw Sprite Font test: Displayed a Times New Roman Sprite Font to the screen.</p> <p>Sprite Background 8: Draw background test: Displayed a scrolling background to the screen from a texture 1979 * 792 (stretches).</p>
063 XNATest1	Quick test to blit two sprites and move them. Sprites collide against the screen and each other. Text is also displayed moving and as an overlay layer.
064 XNATest2	<p>Displays a sprite to the screen moving relative to the accelerometer:</p> <p>Movement 1: Slides and with some acceleration speed increase (0.3f) and slows down by a friction (0.99f) factor.</p> <p>Movement 2: Slides and with some acceleration, the movement is set up to be natural when the device is tilted on a Y-axis setting of 0.3f.</p> <p>Movement 3: Exponential movement travels further exponentially.</p> <p>Movement 4: Exponential movement, travels further exponentially, the movement is set up to be natural when the device is tilted on a Y-axis setting of 0.3f.</p> <p>Movement 5: Only moves when tilted, remains static at optimal viewing angles.</p> <p>Flick 6: Select the sprite and then flick, movement based on time and distance between flick gestures.</p>
065 Piano	Sound test, basic piano application, quick XNA sound on click test.

Common XNA Game Elements	Notes
066 ARTTest	<p>Pervasive system test, built up on the following layers:</p> <p>Layer 1: Camera input projected to screen.</p> <p>Layer 2: Scrolling background, movement based on compass position projected to screen.</p> <p>Layer 3: Animated sprites with gesture collision on (Sprite 1 also has pinch collision). Six sprites positioned top, bottom, left, right, front and back relative to the user in the centre.</p> <p>Layer 4: text output to the screen when collision on sprite.</p> <p>This is a very quick test no loop / arrays / lists used just to test the base layering processes.</p>
067 UtilsTest	Tests for the methods currently listed in the GameCode.Methods table.

Appendix C: Pervasive Game Engine Setup and Configuration

If any of the applications require setting up in a nonstandard way (as opposed to just load in the compiler and running) or configuring with certain parameters (etc.) these requirement have been noted in the following table.

Application	Set-up and Configuration Requirements
056 MessagingGame	<p>Is a Microsoft example but can be modified to test Post Messaging (work in progress)</p> <p>Application Setup:</p> <ul style="list-style-type: none">• To build the application, open the IDE as administrator, to allow access to the server side files. Right click the Microsoft Visual Studio 2010 for Windows Phone and run as administrator. Then find and open the Game.sln project file.• Rebuild as a multiple start up project and run on the emulator (use <code>http://computer name:8000/GameRegirstatorService</code>). Client and Server applications will load and the server will register the client. The interface is set up as a weather system messaging application sending and logging Post Notifications.• To run on the phone emulator (use <code>http://computer name:8000/GameRegirstatorService</code>) register using a cable connection and then keep the server running to send Post Messages.• To get limited local network coverage, link in the phone on a local network via the same Wi-Fi connection and the following needs to be setup:<ul style="list-style-type: none">○ In the GameServer\ app.config file <endpoint address=<code>http://localhost:8000/GameRegirstatorService</code>, localhost needs changing to the computer name or IP address.○ In the GameClient\ MainPage.xaml.cs file string <code>baseUri = "http://localhost:8000/GameRegirstatorService/Register?uri={0}"</code>;; localhost needs changing to the computer name or IP address.• Port 80 needs opening on the server side and the device needs to be set to the same Wi-Fi connection as the server box.

Appendix D: Code Index for the Zombie Gold Application

The DVD contains the following project files:

- **Pervasive Games.docx**, an electronic version of this document.
- **Test Exercises Folder**, the code samples developed while researching the Pervasive Game Engine.
- **Game Folder**, the test game used in this project. To open the game click the Game.sln file and the game will load in the IDE. All the following source code (.cs) files and classes are included in the build:
 - **App.xaml.cs**, application entry point and global file
 - **Bars.cs**, Bars class implements the animated vibration sensor bars in the game
 - **configData.dat**, contains the initial game settings
 - **ConfigData.cs** Config Data class contains the global data for the application
 - **GameCode.cs**, Game Code game utilities library
 - **GameSprites.cs**, Game Sprites class implements the basic in game sprite objects
 - **Ghost.cs**, Ghost class implements the Ghost sprites in the game
 - **Gold.cs**, Gold class implements the gold sprites in the game
 - **Monster.cs**, Monster class implements the monster array sprites in the game
 - **Overlay.cs**, Overlay class implements the search and attack overlays in the game
 - **Pages:**
 - **AchievementPage.cs**, Achievement Page code
 - **GamePage.xaml.cs**, main game XNA page for the application
 - **GameSettingsPage.xaml.cs**, Game Settings Page code
 - **HelpPage.xaml.cs**, Help Page code
 - **MainPage.xaml**, application Main Page code
 - **ScreenEffect.cs**, Screen Effect class implements the overlay screen sprites
 - **Scroller.cs**, Scroller class implementation for scrolling overlays and backgrounds
 - **Shot.cs**, Shot class implements the shot sprites and dynamics in the game
 - **SlingShot.cs**, Sling Shot class implements the sling shot sprites and dynamics in the game
 - **StorageUtils.cs**, Storage Utils is a game data storage utilities library
 - **TimerBar.cs**, Timer Bar class implements the game timer bar in the game

Note: All applications included in this project need to be rebuilt in the IDE to set the project dependencies.

Appendix E: NIST/IUSR – CISU-R

Common Industrial Specification for Usability – Requirements

Usability Requirements for Zombie Gold Windows 7 Phone Application

The following document specifies the usability requirements for the Zombie Gold application.

Testing conforms to Level 1, Level 2 and Level 3 of the NIST/IUSR – CISU-R, Common Industrial Specification for Usability – Requirements.

NIST/IUSR – CISU-R

Common Industrial Specification for Usability – Requirements

Usability Requirements for Zombie Gold Windows 7 Phone Application

Level 1, 2 and 3 Compliance

Date produced: 1st February 2013

Requirements produced by: Paul Morris.

Reviewers: Paul Morris.

Created under the Common Industry Specification for Usability – Requirements (CISU-R) v0.90

Address inquiries to: Paul Morris, Zombie Gold

Phone: 07 5749142

Email: paul.morris@boppoly.ac.nz

Address: Tauranga Campus

Executive Summary

This document specifies the usability requirements for the initial release of the Zombie Gold application. The prototype is delivered in a format that facilitates a review of the application by the elicitation of user input.

This document conforms to Level 1, Level 2 and Level 3 of the CISU-R.

The main function of the Zombie Gold application is defined as a pervasive game developed to test a cross section of functionality previously researched in this project on Pervasive Games. The main interaction in the game revolves around users browsing the application pages and playing a pervasive game in two modes, search mode treasure hunt (touch screen) and attack mode combat (flick screen).

The user base for the application is defined as the current gaming community which represents a cross-section of the whole population. Initially the Microsoft Development Environment linked to Zune is the delivery mechanism with the intention of using the Microsoft Marketplace as the delivery portal for downloading the application if compliance is approved post testing. Experience levels among the users vary considerably, but there is an assumed base level of competence with Windows 7 Phones, allowing users to be able to explore the application and conduct basic operations.

The application is accessed via an installation on a Windows 7 Phone. The application is a pervasive game with play experience designed to be undertaken in potentially any environment, but it is expected that the home environment, passengers on a journey and safe locations such as home / parks / recreation areas will be the predominant game settings. A Microsoft Smartphone (Windows 7 Phone) with the application installed is required as are the Microsoft 2010 (C#) Integrated Development Environment for Windows Phone and Zune for installation purposes. A Nokia Lumia 800 (top range device), HTC Trophy and Nokia Lumia 710 (mid-range devices) phones are utilised for testing purposes.

Requirement scenarios (scenarios of use) have been developed for the following main tasks:

- Learn how to play the game.
- Play a new game.
- Setting the game options.
- Continue a game.
- Explore the game pages.
- Configure and play the game on a difficult setting.

The product prototype has been evaluated in a usability test against the following criteria:

- Effectiveness, the effectiveness of the application is defined as the 'unassisted task completion rate' (priority 1). This is measured as the following percentage, unassisted task completion rate = (total number of unassisted tasks completed / total number of tasks attempted) * 100.
- Efficiency, the efficiency of the application is measured as the 'average time to complete each task', this is defined as the average number of seconds it takes to complete each task (priority 1).
- Satisfaction, a satisfaction rating for the system has been obtained by using a standard SUS questionnaire (priority 1).
- Pervasive Experience Factor, as the application is also used to test how a pervasive game experience is reflected in the game mechanic and interaction for the player.
A questionnaire has been developed to give an effectiveness rating from the perspective of a pervasive experience. The questionnaire has been designed using elements previously identified in this project (Chapter 3) as being crucial to the production of an effective pervasive application (priority 1).

The testing comprises interviews with users to illicit their requirements and research of the current market.

Initial Benchmark Usability Testing

In compliance with Level 2 of the CISU-R all preliminary target values are included in this document and will be substantiated by a benchmark usability test conducted before the final testing of the application.

Usability Testing

Table 1 lists the six main tasks used to test the usability and performance of the application. The usability levels for effectiveness and efficiency are presented for each task along with an overall satisfaction rating and pervasive experience factor.

Task	Effectiveness An unassisted task completion rate of at least the following is required	Efficiency Under the following maximum user time on task is required or between boundaries	Satisfaction A rating of the same or above is required	Pervasive Experience Factor A rating of the same or above is required
Learn how to play the game	100.00%	00:03:56		
Play a new game	100.00%	00:11:01		
Setting the game options	95.50%	00:01:51		
Continue a game	100.00%	00:05:41		
Explore the game pages	97.50%	00:02:10		
Configure and play the game on a difficult setting	100.00%	01:00:50		
The application overall			97.5	86

Table 1: Usability criteria values

The usability levels resulting from the tests all exceed the benchmark values and there has been a considerable amount of positive feedback recorded from the test participants.

Effectiveness

The benchmark unassisted task completion rate fixed at 100% for tasks 1, 2, 4 and 6 have been achieved by all test subjects.

The unassisted completion rate for tasks 3 and 5 had been defined during benchmark testing as 90%. The test results of 95.5% and 97.5% both exceed the required level.

Efficiency

The target values for efficiency set during the benchmark testing for tasks 1 (4:54), 3 (2:02), and 5 (2:49) represented the maximum amount of time a user should take to complete each task. The results from the tests all came in significantly under these benchmark values at task 1 (3:56), task 3 (1:51), task 5 (02:10) indicating the test participants interacted with the application in a very efficient manner.

The benchmark values for tasks 2 (0:07:00 – 0:14:00), 4 (0:02:30 - 0:08:00) and 6 (0:30:00 – 1:20:00) represented target ranges (in-game play time) for completing level tasks. The test data indicated that the correct balance had been achieved, to provide challenging game play at increasingly harder levels, without the game becoming too tedious for the player as all the values (tasks 2 11:01), (task 4 5:41), (task 6 1:00:50) returned from the tests had been recorded as mid-range values.

Satisfaction

A benchmark satisfaction level of 55% had been set pre-testing. A satisfaction level of 97.5% had been returned on completion of the tests indicating a very high level of user satisfaction with the application.

Pervasive Experience Factor

The benchmark pervasive experience factor level of 55% had been set prior to testing with the expectation of the game showing pervasive characteristics at this level. A fuller immersion into the pervasive experience was expected for the player at a PEF level of around the seventies.

The results from the tests returned a PEF level of 86, indicating a fully immersive pervasive experience for the players was achieved. This was also substantiated by test participant feedback that had been recorded during the tests.

Conclusion

One of the stated goals of the Zombie Gold application is as a pervasive game developed to test a cross section of functionality previously researched in this project on Pervasive Games.

Here the application has been successful based on the usability data provided from the tests. The application successfully utilises a cross section of the functionality from the prior research in this project and provides an entertaining game in the pervasive style as proof.

See [appendix G](#) for an example of the Usability Evaluation Data Template.

See [appendix H](#) for an example of the Participant Task Scenarios.

See [appendix I](#) for an example of the SUS questionnaire.

See [appendix J](#) for an example of the Pervasive Experience Factor questionnaire.

See [appendix K](#) for the Preliminary Benchmark Usability Data.

See [appendix L](#) for the Preliminary Benchmark SUS levels.

See [appendix M](#) for the Preliminary Benchmark Pervasive Experience Factor levels.

See [appendix N](#) for the Usability Data.

See [appendix O](#) for the SUS levels.

See [appendix P](#) for the Pervasive Experience Factor levels.

Product Details

Product Description

Name and Version:	Zombie Gold, release 1.0.
Parts for which requirements provided:	Game software.
Intended user groups:	The current gaming community which represents a cross-section of the whole population.
Training required:	None for the intended user groups.
Work supported by product:	Gaming with several types of gaming interaction offered to the users.
Usage environment:	Safe locations and transit as a passenger.

Product Objectives

The objective of the Zombie Gold application is as a pervasive game developed to test a cross section of functionality previously researched in this project on Pervasive Games.

The main interaction in the game revolves around users browsing the application pages and playing a pervasive game in two modes, search mode treasure hunt (touch screen) and attack mode combat (flick screen).

Context of Use and Scenarios

Users

The Gaming Community

The user base for the Zombie Gold game is the current gaming community which represents a cross-section of the whole population.

Experience levels among the users vary considerably, ranging from beginners with a limited experience of mobile games through to expert users who are very familiar with the interaction style of gaming applications. The level of education among the user group also varies considerably as the demographic represents the whole adult population not a specific professional / occupational sub-category. Even though the experience and educational levels of the user group may be low, users are assumed to have a base level of competence, the ability to explore the game pages and also conduct basic operation on a Windows 7 Phone.

The key characteristics of the intended user group are the enjoyment of playing mobile games along with an interest in new and innovative entertainment applications.

Typical users of the application are:

- *Joe lives north of Auckland and commutes into the City centre during the week (by bus) where he works as an accountant for a large City Firm. He is 54 years old, married with two children and has a moderate disposable income. Joe spends a lot of time on his phone during the day and likes to keep a few new applications on his phone to keep himself amused on his journey to work (he feels it brightens up his day).*
- *Tina is a housewife who lives on a batch in Hamilton. She is 25 years old, married with two small children and has a small disposable income. Tina parents live in Tauranga which she visits every weekend with the family. The drive is over an hour and the problem is in entertaining the kids over a boring journey that she takes twice a week. Tina downloaded the Zombie Gold application a couple of months ago. The ride now has become a treat for the children as they are engrossed in the game, which stops them getting cranky.*
- *Henry is a male employee of the Bay of Plenty Polytechnic, he is 41 years old, married and has one child. Recently he ripped his Achilles tendon. After playing the game (by chance) for a while Henry realised that using the game helped him get some movement in his foot again.*
- *Sid is a male Software Test Engineer living and working in Auckland. He is 26 years old, single and has a high disposable income. Sid spends a lot of time testing phone applications as part of his working day. He also enjoys visiting the on-line gaming sites, XSports events and searching for new applications in his spare time.*
- *Lucas lives in Hamilton where he studies Mathematics. He is 30 years old, single and has a relatively low income. Recently at lunch time he has been playing a new pervasive game at various locations around town.*
- *Tom is a male student living and studying in Dunedin. He is 19 years old, single and spends a lot of time avoiding working, snowboarding and texting his friends. Tom and his friends have discovered a new location based game (Zombie Gold) that they can play while texting each other to fill the gaps between texts.*

Users with Disabilities

Users with disabilities (motor, hearing or vision) are able to access the game with the appropriate assistive devices (e.g. touch screens, earphones or screen magnifiers).

Goals and Tasks

The goal of the Zombie Gold application is to provide a fun and seamless gaming experience to its users. The main function of the Zombie Gold application is as a pervasive game developed to test a cross section of functionality previously researched in this project on Pervasive Games.

The main interaction in the game revolves around users browsing the application pages and playing a pervasive game in two modes, search mode a treasure hunt (touch screen) and attack mode a combat shooter (flick screen).

The following six tasks are required to interact successfully with the application:

- Learn how to play the game.
- Play a new game.
- Setting the game options.
- Continue a game.
- Explore the game pages.
- Configure and play the game on a difficult setting.

Technical Environment (equipment)

The Zombie Gold application requires installation on a Windows 7 Phone. A Microsoft Smartphone (Windows 7 Phone) with the application installed is required as are the Microsoft (C#) Integrated Development Environment for Windows Phone and Zune for installation purposes.

A Nokia Lumia 800 (top range device), HTC Trophy and Nokia Lumia 710 (mid-range devices) phones have been utilised for testing purposes.

The base specification required for the phones is:

- Display, 480x800 QVGA / 320x480 HVGA
- Capacitive Touch, 4 or more contact points
- Sensors, GPS, accelerometer, compass and light
- Camera, 5 mega pixels or more
- Hardware buttons, Start, Search and Back
- Multimedia, common detailed spec, Codec acceleration
- Graphics Processing Unit (GPU) DirectX 9 acceleration
- CPU ARMv7, Cortex/Scorpion or better
- Media, Digital capture / playback and media library access
- .Net, Superset of Silverlight 3.0, Windows Commercial Foundation, Simple Object Access Protocol (SOAP) services and Representational State Transfer (REST) web services
- Phone Access, Integrated access to the phone UI, Sensors and picker for contact / photos
- Integrated with Cloud services, App deployments and updates, notifications, location and Xbox LIVE

Physical and Social Environment

The physical environment is any safe locations a user can use their mobile phone.

For the purpose of these tests the following settings have been identified.

- As a passenger on a journey in private (car) or public (bus) transport.
- At safe locations such as home, parks, recreation areas, locations around town.
- At a location used for exercise, garden, gym and recreation areas.

The game is accessible in most social environment. Users predominantly interact with the game on their own, but the game can also be viewed and watched by groups of users. Users are expected to be involved in other day-to-day tasks while they access the game. Therefore there is an expectation that the game is potentially open all day as a background application.

Scenarios of Use for the Most Important Goals

Scenario 1: Learn how to play the game

Joe lives north of Auckland and commutes into the City centre during the week (by bus) where he works as an accountant for a large City Firm. He has a long bus commute to work each day and tends to spend the time playing on his phone during the journey.

Today on his way to work he downloaded a copy of Zombie Gold from the Marketplace and has had his phone on most of the day next to his laptop checking out the graphics in the game. Although eager to play the game he frustratingly was too busy all day at work to use the application.

At 5:00pm he heads for home and get his usual bus. When he is settled in his seat Joe gets his phone out, switches it on, clicks on the game icon in the application list and watches the Zombie Gold splash screen as the game loads. When the main screen appears he is unsure where to go so he expands the menu bar to see several navigation options offered by the game. Joe selects the question mark icon that is defined as the help option and waits for the Help page to load up. Joe touches the phones screen when the page loads and realises that the help text is moving when he moves his thumb across the screen (nice feature he thinks). After playing with the scrolling display Joe realises that he can read the text by scrolling left / right and up / down. On his journey home Joe reads through the help options to learn how to play the game. When he is finished reading he presses the start menu icon on the phone and exits the game as his stop is approaching.

As a veteran gamer Joe always wants to explore the help options first to learning how to configure and play a game properly.

Scenario 2: Play a new game

Tina is a housewife who lives on a batch in Hamilton. She is 25 years old, married with two small children Bill and Ben. She is very busy with the children around all day and has little time to get out during the week as her husband also works long hours. The weekends tends to be spent in Tauranga with her parents as this gives the family a break at the weekends and a chance for the kids to see their grandparents.

On Friday evening at 8:00pm the family set out for Tauranga. Usually the journey is a nightmare keeping the kids amused as it is over an hour with toilet stops for the children, but recently Tina downloaded a free copy of the Zombie Gold application onto her and her husband's phones. The game was configured for the easiest game options when downloaded and ready to play. Once the kids are strapped in the back of the car, they head off for Tauranga. Tina gets the phones out, switches them on, clicks on the game icon in the application list of each phone and watches the Zombie Gold splash screens as the game loads. When the main screens appear she presses the Start New Game Button on each of the phones and carefully passes the phones to the kids, they get grumpy if she tilts the game before they start playing. The children can see the search gold icons on top of the screen and know they need to catch all of them to progress to the attack mode (where they can shoot monsters). As they play they remind each other to look around / up / down for gold and to click on them to catch them. After a while playing Ben looks at the score on the timer bar of Bill's game and sees Bill is winning. This causes him to search faster but in the process he tilts the game (Bill laughs as he likes the glass cracking effect when it appears in his brothers game), generating several ghosts in the process. Ben's timer is also very low so he desperately needs to catch gold quickly. Bill tells him to keep pressing the screen to keep the ghosts away from the gold and doing so Ben has a large gang of ghosts under his control.

The kids have been having so much fun they have not noticed that the car is pulling into granddads drive. Tina takes the phones off the boys and reads the final scores, Ben is the winner this time and this causes a minor scuffle in the back seat. Dad shouts and the boys scramble out of the car. Tina then presses the start menu icon on each phone and exits the games, the boys are already banging on granddads door.

Scenario 3: Setting the game options

Henry is a male employee of the Bay of Plenty Polytechnic, he is 41 years old, married and has one child. Recently Henry ripped his Achilles tendon and found that using a mobile phone game (Zombie Gold) of his son's helped to alleviate the pain. The combination of turning and holding the phone out / down / above his head, while not focusing on the pain in his foot seemed to give him some mobility again. The only problem with the game is his son has the game on a difficult setting which is too fast for Henry to cope with.

Early Sunday morning Henry decides that a quick play in the garden will help get his foot going for the day. Henry grabs the phone from the desk in his son's room and switches it on. He clicks on the game icon in the application list and watches the Zombie Gold splash screens as he hobbles out into the garden. Not having setup the game before (his son did it initially) when the main screen appears he is not sure what to do. The menu bar seems to be the best place to start (he thinks), so he expands the menu bar by touching the ... option to see several navigation options offered by the game. Henry selects the settings menu list option and waits for the Settings page to load up. When he has previously use the game he found the search mode fine, but the attack mode was too difficult to shoot the monsters before he was killed (too stressful). Henry sits down on a chair in the garden and looks at the options on the setting screen, he still wants all the graphics and sounds (they distract him in a positive way) but decides to make the attack options easier to use. He selects the flick attack (easy shot action) by touching the Slingshot or Flick selection box until the box is un-ticked, he notices that this also greys out the graphic for the sling shot option which makes sense he thinks. He then selects the screen collision (more bouncing the easier to hit) option by touching the Screen Collision selection box until the box is ticked, this enables the Explode / Bounce option. He selects the bounce option by touching the Explode / Bounce selection box until the box is un-ticked. Finally he increases the Shot Curve slider to 2 (add spin) and the Number of Shots slider to 12 (repeated shots).

Henry gets up and starts to walk a little as his leg is getting stiff. He navigates back to the main page by pressing the back button on the phone. Before he starts playing he wonders if he needs to manually save the new settings or are they automatically saved, so he selects the settings menu list option again and waits for the Settings page to load up. When it opens he sees his new settings are still there so he realises that they have been saved automatically. Just then his wife calls him from the kitchen, the cat has brought a bird into the house, come quickly and help, great he thinks that will sort his foot out. Henry presses the start menu icon on the phone and exits the games, then hobbles for the house.

Scenario 4: Continue a game

Sid is a male Software Test Engineer living and working in Auckland. He is 26 years old, single and has a high disposable income. Sid spends a lot of time testing phone applications so he has access to and an interest in all the new applications on the market. Zombie Gold is his current favourite. This weekend his two smaller sisters (twins) Holly and Molly have come up for Hamilton to Auckland with his parents, who want to do a bit of shopping.

It is Saturday evening and for a treat Sid has set up a scary game for his little sisters. He has put up Halloween decorations around his flat and the girls room (the spare bedroom) is lit with coloured Christmas decorations. Sid get a couple of phones from his office desk, they are loaded with Zombie Gold from the last time the twins were over. The games are configured for the easiest game options. Once the twins are sat on the couch in the living room Sid switches them on, clicks on the game icon in the application list of each phone

and passes them to his sisters. The girls watch the splash screen and giggle (it looks spooky) as the game loads. They both press the Paly game button and when the game loads exchange phones. They remember the score and number of monsters they need to shoot (shown on the attack overlay) from the last time they played and realise they have the wrong phones. Both Holly and Molly are experts and leap around the room shooting monsters as the games were both last played in Attack Mode. As they play the dark rooms in the flat merge with the spooky monsters in the game and it looks like the phone is a magic window that lets them see otherwise invisible monsters in the room. They both have different playing styles. Holly uses a flick shot and shoots all the crawling monsters first, then the floating ones and finally the flying ones, by spiralling around as she moves around the flat. Molly shoots by keeping her finger in one place for rapid fire and searches for any monster by its size (biggest first), while running frantically about then stopping to fire. Some of the background sounds make the girls jump as they play randomly and this make Sid laugh as the girls try not to show they were spooked. While the girls are playing Sid chats with his parents and puts the clean washing and food away his parents have brought for him.

10:00pm is approaching and it's time for bed. Sid takes the phones off the girls and reads the final scores, Holly is the winner this time, but the score will be saved automatically so they both know the game is not over. Sid presses the start menu icon on each phone and exits the games, as the girls are shepherded to brush their teeth. Sid's parents are happy, with the game and shopping today they know the girls will sleep tonight.

Scenario 5: Explore the game pages

Lucas lives in Hamilton where he studies Mathematics. He is 30 years old, single and has a relatively low income. Recently at lunch time he has been playing a new pervasive game at several locations around the town. Passers-by have started to notice (and become interested in) the strange antics of Lucas and his friends as they lunge about to catch gold or shoot monsters. Lucas is bit of a techie and is determined to check out everything about the game mechanics (how the thing works). This lunch time Lucas is alone in the park and decides to investigate the game. He wants to see how the basic mechanics of the game works by removing some of the graphics and effects that distract him.

Lucas takes his phone out and switches it on. He clicks on the game icon in the application list and watches the Zombie Gold splash screens as he eats one of his sandwiches. He knows the main game page menu fairly well and selects the achievement icon (star) by touching it and waits for the Achievements page to load up. Henry likes the fade in boxes of the gold types, but really just wanted to boost his ego by looking at the massive score he has for level 15. Getting bored he navigates back to the main page by pressing the back button on the phone. From the main page he selects the settings icon (cogwheel) by touching it and waits for the Settings page to load up. Lucas leans back on the grass, checks he is still on level 15 and starts to switch off the following options Monster Land, Real World, Tilt Vibrate, Motion Bars, Timer Bars, Game Score, Screen Effects, Game Overlays, Ambient Sound, and Sound Effects. He thinks to himself that the moving sprites in the game will just be placed on the screen randomly and not relate to the movement of the device. He navigates back to the main page by pressing the back button on the phone. Pressing the Play Game button the game starts in attack mode the setting saved from the last time he played the game and he is surprised to see how realistic the physics on the shot curve are implemented. As monsters are generated it also becomes apparent the game sprites move relative to the devices location and orientation. The conversions from 3D to 2D space are very accurate, Lucas practically likes the fact that you only see what you are facing (a very nice touch). Being engrossed in trying to find any flaws in the game Lucas realises that the game has moved to search mode. Excellent, the ghosts actually track the gold in a 3D space. He then picks up on a bit of insider game play (from watching the ghost movement) if he allows the ghost to start to chase the gold they will show him in what direction to turn. He glances at his watch, time to get back to class he thinks. Lucas presses the start menu icon on the phone and exits the games, then heads for his statistics lecture.

Scenario 6: Configure and play the game on a difficult setting

Tom is a male student living and studying in Dunedin. He is 19 years old, single and spends a lot of time avoiding work, snowboarding and texting his friends. Recently Tom and his friends have taken to playing a pervasive game in the oddest locations they can think of. The challenge is based on a player succeeding in completing a level at a strange location before leaving the location (or be moved on) with the game setting set to the hardest difficulty possible. Tom is the current leader at Level 9 and today is going for a level 10 attempt on the spinning Tea Cup ride at the local kids fair, there is a 5 ride limit set by the students for a successful attempt to be recognised.

Tom and his friends get ready for the official attempt by paying for 5 rides (\$3 per ride). While in the queue he takes his phone out and switches it on. He clicks on the game tile he has pinned to the phones main menu and watches the Zombie Gold splash screen as he moves towards the pay booth. When the main page loads he selects the settings icon (cogwheel) by touching it and waits for the Settings page to load up. His friends cluster around to check that he sets the correct options for the attempt to be valid. Tom checks his level setting is still on level 10 and starts to check that the following options are ticked Monster Land, Real World, Tilt Vibrate, Motion Bars, Timer Bars, Game Score, Screen Effects, Game Overlays, Ambient Sound, Sound Effects. He is reminded by his friends that the following also need setting Shots per Volley 1, Small Shot Curve 0, Slingshot of Flick ticked (Slingshot), Slingshot Graphic ticked, Screen Collision ticked and Explode / Bounce ticked (Explode). Tom makes the settings changes and pays for his tickets. The gang take their seats in a pink tea cup and waiting for the ride to start Tom navigates back to the main page by pressing the back button on the phone. As the ride starts Tom touches the Play Game button on the main page and the game begins. Tom hasn't thought about the fact that his three friends are not only in the cup with him but will be spinning the cup with the wheel in the centre of the seats. He can

catch the gold but the tilt is impossible to control and the game is swamped with ghosts. As the first ride ends he has tilted the game into attack mode and now needs to kill all the monsters just to restart the level. Luckily the ride stops for quite a while for the other cups to fill up and in attack mode the monsters generate 3 low numbers. By the time the ride starts again he is back to search mode with 3 hysterically laughing friends next to him. The next two rides see Tom clearing search mode successfully and re-entering attack mode with a chance to level up.

That's when disaster strikes Tom has forgotten to charge his phone overnight, the screen goes blank and the battery critical message pops. End of game, end of attempt, Toms mates are in hysterics, Tom moves to legendary status. When Tom gets home he charges the phone and has a quick check to see if the game is still current at the point it stopped. He clicks on the game menu tile and when the main page loads touches the Paly Game button. The game loads and is still in the exact state it was on the ride. Tom quickly presses the start menu icon on the phone and exits the games.

The texts begin to fly and the debate begins, can the attempt be continued..... or is it a restart.....

Usability Performance and Satisfaction Criteria

Usability Goals

The goal of the Zombie Gold application is to provide a fun and seamless gaming experience to its users. The main function of the Zombie Gold application is as a pervasive game developed to test a cross section of functionality previously researched in this project on Pervasive Games.

The main interaction in the game revolves around users browsing the application pages and playing a pervasive game in two modes, search mode a treasure hunt (touch screen) and attack mode a combat shooter (flick screen).

The following six tasks are required to interact successful with the application:

- Learn how to play the game.
- Play a new game.
- Setting the game options.
- Continue a game.
- Explore the game pages.
- Configure and play the game on a difficult setting.

The following six tasks are required to achieve this goal:

- Learn how to play the game. This task has the highest priority because without learning how to play the game users will not be able to play the game to its fullest potential.
- Play a new game. This task has the highest priority because playing the game is the primary function of the application.
- Setting the game options. This task has the highest priority because without configuring the game correctly users will not be able to tailor the game to their requirements.
- Continue a game. This task has the highest priority because playing the game is the primary function of the application.
- Explore the game pages. This task has the highest priority because it allows users to the check on game elements that add extra value to the game play.
- Configure and play the game on a difficult setting. This task has the highest priority because it allows users to keep a long term interest in the game by setting themselves harder tasks to complete.

All six of the above tasks are classified as having a high level of priority, because the omission of any one of these actions would exclude a recognised user type (new user, user and advanced user) within the targeted demographic from the game.

Test Criteria

The performance criteria for the Zombie Gold application is measured against the ability of the appropriate users to complete all six (goal) tasks without any assistance or training, apart from the help and information supplied by the application.

On completion of the tasks, users must be at least as satisfied with the game, as they are with any existing games (with similar functionality) they have used.

Effectiveness, the effectiveness of the application is defined as the ‘unassisted task completion rate’. This is the percentage of users that complete all tasks unassisted, other than by using the in-game help. This is measured as the following percentage, unassisted task completion rate = (total number of unassisted tasks completed / total number of tasks attempted) * 100. The effectiveness of the application is the first priority, as any failed tasks will make the game unusable to users.

Efficiency, the efficiency of the application is measured as the ‘average time to complete each task’, this is defined as the average number of seconds it takes to complete each task. This is a first priority as there is a requirement for efficient interaction in the game, despite the fact that exploration and extended periods spent playing the game are key factors in the enjoyment of the application.

Satisfaction, a satisfaction rating for the system has been obtained by using a standard SUS questionnaire after users completed the six tasks. This is also a first priority because users will be unwilling to continue using the application if there are high levels of dissatisfaction.

Pervasive Experience Factor, an effectiveness rating from the perspective of a pervasive experience has been obtained by using a tailor made questionnaire after users completed the six tasks. A positive pervasive experience of the application is the first priority, as one of the primary purposes of the testing is to prove the effectiveness of the functionality developed in the project.

See [appendix G](#) for an example of the Usability Evaluation Data Template.

See [appendix H](#) for an example of the Participant Task Scenarios.

See [appendix I](#) for an example of the SUS questionnaire.

See [appendix J](#) for an example of the Pervasive Experience Factor questionnaire.

Initial Benchmark Usability Testing

The benchmark for user effectiveness, efficiency, satisfaction and pervasive experience factor ratings focused on interviews with users to illicit their requirements, quantitative and qualitative usability testing and research of the current gaming market.

Table 2 lists the six main tasks used to test the usability and performance of the Zombie Gold application. The tasks have been designed to simulate a valid set of user interactions with the application by utilising all of the games functionality. The preliminary benchmark usability levels for effectiveness and efficiency have been presented for each task along with an overall satisfaction rating and pervasive experience factor.

The preliminary benchmark usability levels have been set by the observation and discussion with selected test participants from the defined demographic. Test participants undertook equivalent tasks on current games in the marketplace and the data from these tasks has been used to set the preliminary benchmark

usability levels. Where tasks were unique to the Zombie Gold application and not repeatable, values have been defined based on discussions between the development team and the test participants.

The unassisted task completion rate has been fixed at 100% for tasks 1, 2, 4 and 6 as all users of the application will be required to fully perform these basic tasks in the game. If the effectiveness rating of any of these four tasks falls below this percentage then the application becomes unusable. This value (100%) also represented the mean value ascertained from the benchmark participant testing.

The unassisted completion rates for tasks 3 and 5 defined during benchmark testing were 86.50% and 97.50%. Based on discussions between the development team and the test participants these percentages have been reset to 90%, as it is felt that a significant amount of the application functionality is accessible to the users at this level. It has also been noted by the test participants that due to the nature of the application there is a very easy learning curve once the game had been played a couple of times. In fact the ease of use once learned was mentioned by all test participants. Although 90% effectiveness rating has been defined as being an acceptable level of achievement for tasks 3 and 5, the development team felt that 100% effectiveness should be attainable post testing, by reviewing the test feedback on the effectiveness of the application.

The target values for efficiency represent the maximum amount of time a user should take to complete a task. These values have been set during the benchmark testing and represent the mean time for each task. These values are an indication of the maximum user time on task and represent a benchmark for the internal development team for future development.

Tasks 2, 4 and 6 show target ranges that represent in-game level completion times (level tasks and play paths). These ranges have been defined to allow challenging game play at increasingly harder levels without the gaming experience becoming too tedious for the player.

The benchmark satisfaction level of 55% has been set by distributing a standard SUS questionnaire after users completed the six tasks.

The benchmark pervasive experience factor level of 55% has been set by distributing a standard tailor-made PEF questionnaire after users completed the six tasks.

Task	Effectiveness An unassisted task completion rate of at least the following is required	Efficiency Under the following maximum user time on task is required or between boundaries	Satisfaction A rating of the same or above is required	Pervasive Experience Factor A rating of the same or above is required
Learn how to play the game	100%	0:04:54		
Play a new game	100%	0:07:00 – 0:14:00		
Setting the game options	90%	0:02:02		
Continue a game	100%	0:02:30 - 0:08:00		
Explore the game pages	90%	0:02:49		
Configure and play the game on a difficult setting	100%	0:30:00 – 1:20:00		
The application overall			55	55

Table 2: Initial benchmark target criteria values

See [appendix K](#) for the Preliminary Benchmark Usability Data.

See [appendix L](#) for the Preliminary Benchmark SUS levels.

See [appendix M](#) for the Preliminary Benchmark Pervasive Experience Factor levels.

Usability Test Requirements Format

This section defines the method used to test if the usability requirements for the application have been met.

The format used reflects the CIF (ISO/IEC 25062) so that the results of all future testing can be documented using the CIF.

Users

During the tests conducted ten test participants have been used, all from the required demographic. This exceeds the minimum CIF recommendation of one user taken from the defined demographic.

Goals

Refer to the Usability Performance section in the body of this document for the test Goals.

Test facility

The CIF recommendation advocates that testing should be conducted in an environment that represents as closely as possible the physical and social environment in which the user will be using the application.

To facilitate this, usability labs have been setup as a home environment and as a car, with testing conducted on a several Window 7 phones. A test administrator monitored the test participants and record their feedback and actions while undertaking the test.

Participant's Mobile Platform

To facilitate testing on several platforms three Windows 7 phones have been used for testing purposes a Nokia Lumia 800 (top range device), HTC Trophy and Nokia Lumia 710 (mid-range devices). This exceeds the minimum CIF recommendation of the participant's mobile device should meet or exceed the equipment requirements defined in the Technical Environment (equipment) section of this document.

Experimental Design

Tasks have been given to participants in order, tasks one through to task six, as all tasks are dependent on all of the tasks being successfully completed. The following seven dependent measures have been utilised in the testing:

- **Time to complete task**, the total time it takes a participant to complete a task.
- **Unassisted task completion rate**, the percentage of the task completed without assistance, other than by using the in-game help.
- **Assisted task completion rate**, the percentage of the task completed but requiring assistance from the test administrator during a task.
- **Number of assists**, the number of times a participant received help from the test administrator during a task.
- **Number of errors**, the number of errors a participant made during a task.
- **SU questionnaire**, a SU satisfaction measure calculated from a SUS satisfaction questionnaire given to a participant at the end of the six tasks.
- **PE questionnaire**, a PE satisfaction measure calculated from a Pervasive Experience Factor questionnaire given to a participant at the end of the six tasks.

See [appendix G](#) for an example of the Usability Evaluation Data Template.

See [appendix H](#) for an example of the Participant Task Scenarios.

See [appendix I](#) for an example of the SUS questionnaire.

See [appendix J](#) for an example of the Pervasive Experience Factor questionnaire.

Procedure

Testing of the Zombie Gold application has been conducted in form of 5 testing sessions over a 5 day period. The following is a record of the testing procedure utilised during the testing process:

- Each session had been scheduled for a morning start (10:00am) and split into two parts home and car, each session required two participants.
- Each session had been set up in usability labs designed to look and feel like a home environment and the passenger seat of a car.
- Several Windows 7 phones had been loaded with the application and made available.
- A test administrator monitored the participants and also recorded the participant's feedback and actions while undertaking each task.
- At the start of the test the participants had been told they were participating in a usability evaluation of the Zombie Gold application and that the procedures were to test the system not their individual performance or capabilities.
- At the start of the tests it had been explained to the participants that they will be following some scenarios (asked to play, search and change setting in the game) and that their actions will be timed, monitored and then recorded on data sheets by the test administrator present.
- Test participants worked through the scenarios in the usability labs:
 - Home, activities staged around the house and gardens.
 - Car, activities staged in the passenger seat of a car driving around Tauranga.
- During the testing the test administrator timed, observed and recorded usability evaluation data as each participant progressed through the task scenarios.
- During the testing the test administrator asked the test participants to think aloud and answer questions.
- During the testing the test administrator recorded feedback.
- At the end of the test the participants filled in subjective questionnaires.

Concluding the testing sessions participants have been thanked for volunteering and undertaking the tests. A barbeque had also been organised in the form of a thank you.

Participant general instructions

Pre-test it had been explained to the participants that there are no time limits to the tasks and that any questions can be directed to the test administrator (Paul Morris). Each participant had been given a copy of the tasks which had also been read aloud by the test administrator during the test. Participants had been encouraged to think aloud throughout the session and answer any questions directed to them during the test. The participants had been assured that all their feedback is relevant and also instructed to work through the tasks as efficiently as possible.

Participant task instructions

Task 1: Learn how to play the game

You are an accountant called Joe who lives and works in Auckland. This morning on your way to work you downloaded a copy of Zombie Gold from the Marketplace and have been waiting all day to play the game. As a veteran gamer you always want to explore the help options first to learn how to configure and play a game properly. On your way home on the bus you decide to learn about the game.

- 1.1 Find the **game icon** and open the game.
- 1.2 Find and select the **Help navigation icon** on the Main screen.
- 1.3 Find out how to **read the Help text** on the Help page.
- 1.4 Read the section on **Game Rules** in the help text.
- 1.5 Read the section on **Settings Page** in the help text.
- 1.6 Exit the game using the **Start Menu** icon on the phone.

Task 2: Play a new game

You are a young housewife called Tina who lives with your husband and two children on batch in Hamilton. You are very busy with the children around all day and have little time to get out during the week. Your weekends tend to be spent in Tauranga with your parents as this gives the family a break at the weekends and a chance for the kids to see their grandparents. On Friday evening at 8:00pm the family set out for Tauranga. Usually the journey is a nightmare keeping the kids amused as it is over an hour with toilet stops for the children, but you have downloaded a free copy of the Zombie Gold application onto a couple of phones. The game was configured for the easiest game options when downloaded and is ready to play. Once the kids are strapped in the back of the car, you get the phones out for the children to play with and head off for Tauranga.

- 2.1 Find the **game icon** and open the game.
- 2.2 Find and select the **Start New Game button** on the Main screen.
- 2.3 Find the **search gold icons on the top** of the game screen and **note how many there are and the values on each one**.
- 2.4 Play the game looking around / up /down for gold until you catch one.
- 2.5 Tilt the game by moving the game too quickly.
- 2.6 Control a ghost by tapping the screen (move the ghost to centre of the screen).
- 2.7 Allow a ghost to catch a gold sprite (stop tapping the screen) follow it on the screen until it explodes.
- 2.8 Play the game until search mode is completed successfully or the game timer expires.
- 2.9 Exit the game using the **Start Menu** icon on the phone.

Task 3: Setting the game options

Your name is Henry and you are a male employee of the Bay of Plenty Polytechnic, you are 41 years old, married and have one child. Recently you ripped your Achilles tendon and have found that using a mobile phone game (Zombie Gold) of your son's has helped to alleviate the pain. The only problem with the game is that your son has the game on a difficult setting which is too fast for you to cope with. Early Sunday morning you decide that a quick play in the garden will help get your foot going for the day. You sit down on a chair in the garden with your phone and decide to reset the game option to an easier mode before you play.

- 3.1 Find the **game icon** and open the game.
- 3.2 Expand the **menu bar** on the Main screen by touching the ... option
- 3.3 Select the **settings menu list option** to load the Settings page.
- 3.4 Set the following options on the **Settings** page to make the game easier to play:
 - Slingshot or Flick option to un-ticked.
 - Screen Collision option to ticked (this enables the Explode / Bounce option).
 - Explode / Bounce option to un-ticked.
 - Increase the Shot Curve slider to 2.
 - Increase the Number of Shots slider to 12.
- 3.5 Navigate back to the **Main** page by pressing the **Back button** icon on the phone.
- 3.6 Select the **settings menu list option** on the Main page menu bar to load the Settings page again, to see if the setting you previously changed have remained changed (saved).
- 3.7 Exit the game using the **Start Menu** icon on the phone.

Task 4: Continue a game

You are Software Test Engineer called Sid living and working in Auckland. You spend a lot of time testing phone applications so have access to and an interest in all the new applications on the market. Zombie Gold is your current favourite.

This weekend your two smaller sisters (twins) Holly and Molly have come up for Hamilton to Auckland with your parents. It is Saturday evening and for a treat you have set up a scary game for your little sisters, by putting up Halloween decorations and coloured Christmas lights around your flat.

You get a couple of phones out that are loaded with Zombie Gold from the last time the twins were over. The games are configured for the easiest game options. Once the twins are sat on the couch in the living room you start the games and hand them the phones.

- 4.1 Find the **game icon** and open the game.
- 4.2 Check that the splash screen appears while the game loads (record how long it is visible).
- 4.3 Find and select the **Play Game button** on the Main screen.
- 4.4 Find the current **Score on the bottom** of the game screen and **note how many points you have**.
- 4.5 Find the **attack monster icons on the top** of the game screen and **note how many there are** and **the values on each one**.
- 4.6 Listen until a random **sound** is generated for the background noise in the game.
- 4.7 Play the game looking around / up /down shooting monsters until you **shoot one of each of the three monster types**.
- 4.8 Bounce a shot of the screen edge using a **pullback flick shot**.
- 4.9 Let one of the monsters hit you to generate a screen splat.
- 4.10 Play the game until attack mode is completed successfully or you run out of points.
- 4.11 Exit the game using the **Start Menu** icon on the phone.

Task 5: Explore the game pages

You are a student called Lucas who studies Mathematics in Hamilton.

Recently at lunch time you have been playing a new pervasive game at several locations around the town.

You are a bit of a techie and are determined to check out everything about the game mechanics (how the thing works).

This lunch time you are alone in the park and decide to investigate the game. You want to see how the basic mechanics of the game work by removing some of the graphics and effects that distract you during normal game play.

You take your phone out and switch it on as you start munching your sandwiches.

- 5.1 Find the **game icon** and open the game.
- 5.2 Find and select the **Achievements navigation icon** on the Main screen.
- 5.3 Find the current score for **Level Beginner** on the Achievements page, note the score.
- 5.4 Navigate back to the **Main** page by pressing the **Back button** icon on the phone.
- 5.5 Find and select the **Settings navigation icon** on the Main screen.
- 5.6 Set the following options on the **Settings** page to make the game easier to play:
 - Decrease the **Level slider** to **Beginner**.
 - **Monster Land** option to **un- ticked**.
 - **Real World** option to **un- ticked**.
 - **Tilt Vibrate** option to **un- ticked**.
 - **Motion Bars** option to **un- ticked**.
 - **Timer Bars** option to **un- ticked**.
 - **Game Score** option to **un- ticked**.
 - **Screen Effects** option to **un- ticked**.
 - **Game Overlays** option to **un- ticked**.
 - **Ambient Sound** option to **un- ticked**.
 - **Sound Effects** option to **un-ticked**.
- 5.7 Navigate back to the **Main** page by pressing the **Back button** icon on the phone.
- 5.8 Find and select the **Play Game button** on the Main screen.
- 5.9 When the game page loads check that all the options are off, no effect layers will be displayed.
- 5.10 Exit the game using the **Start Menu** icon on the phone.

Task 6: Configure and play the game on a difficult setting

You are a student called Tom living and studying in Dunedin. Recently you and your friends have taken to playing a pervasive game in the oddest locations you can think of.

The challenge is based on a player succeeding in completing a level at a strange location (or situation) before leaving the location with the game setting set to the hardest difficulty possible.

You are the current leader at Level 9 and today you are going for a level 10 attempt. The challenge is to play a level while a friend (another test volunteer) spins a revolving office chair you will be sat on while playing the game.

You take out your phone and switch it on while sitting in the chair.

- 6.1 Find the **game tile** in the **phone main menu** and open the game.
- 6.2 Find and select the **Settings navigation icon** on the Main screen.
- 6.3 Set the following options on the **Settings** page to make the game harder to play:

- Set the **Level slider** to **10**.
- **Monster Land** option to **ticked**.
- **Real World** option to **ticked**.
- **Tilt Vibrate** option to **ticked**.
- **Motion Bars** option to **ticked**.
- **Timer Bars** option to **ticked**.
- **Game Score** option to **ticked**.
- **Screen Effects** option to **ticked**.
- **Game Overlays** option to **ticked**.
- **Ambient Sound** option to **ticked**.
- **Sound Effects** option to **ticked**.

You friend (other test volunteer) reminds you that you also need to set the attack options to difficult, as you are going for a level 10 attempt at the record.

- 6.4 Set the following options on the **Settings** page to make the game harder to play:
 - Shots per Volley slider to 1.
 - Small Shot Curve slider to 0.
 - Slingshot of Flick option to **ticked** (Slingshot).
 - Slingshot Graphic option to **ticked**.
 - Screen Collision option to **ticked**.
 - Explode / Bounce option to **ticked** (Explode).
- 6.5 Navigate back to the **Main** page by pressing the **Back button** icon on the phone.

You friend (other test volunteer) begins to spin the chair and move you around the room.

- 6.6 Find and select the **Play Game button** on the Main screen.
- 6.7 Play a **game rule path** for the game level, both the search and attack modes, the game length will be dependent on your success (progress a level) or failure (go back a level) playing the level.
- 6.8 When you have completed the level play, exit the game using the **Start Menu** icon on the phone.

Usability metrics

Effectiveness

Completion rate, both unassisted and assisted has been considered along with the number of errors and assists.

Completion Rate

Unassisted completion rate is defined as the percentage of a task completed correctly by the participant without any assistance from the test administrator.

Assisted completion rate is defined as the percentage of a task completed correctly by the participant with the test administrator's intervention.

Assists

An assist is defined as any verbal help given by the test administrator to guide the participant to the next step in completing a task. The participants have been given unlimited attempts at completing a task, assistance has only been given when the participant asks the test administrator, and then there has been no set limit on assists.

Errors

An error is defined as any action taken by the participant not required in the sequence of actions in the process of completing a task, including omissions.

Efficiency

Time on task has been considered, the task completion rate efficiency also includes the number of references to the in-game help the user makes.

Task Time

Task time is defined as the total amount of time it takes the participant to complete a task.

Completion Rate Efficiency

The completion rate efficiency is defined as, $\text{completion rate efficiency} = (\text{unassisted completion rate} / \text{mean task time}) * 100$.

Satisfaction

At the end of the session the test participants completed a subjective post-test questionnaire designed specifically (based on the SUS questionnaire) to highlight issues relating to the inherent usefulness, ease of use, clarity, functionality, appearance and enjoyment rating of the Zombie Gold application. The participants rated questions in the questionnaire on a 5-point semantic differential scale, strongly disagree/don't know/strongly agree.

Pervasive Experience Factor

At the end of the session the test participants completed a subjective post-test questionnaire designed specifically to highlight issues relating to the pervasive experience provided by the Zombie Gold application. The participants rated questions in the questionnaire on a 5-point semantic differential scale, strongly disagree/don't know/strongly agree and with extended answers to open question.

Usability Testing Results

Table 3 lists the six main tasks used to test the usability and performance of the Zombie Gold application. The tasks have been designed to simulate a valid set of user interactions with the application by utilising all of the games functionality. The usability levels for effectiveness and efficiency have been presented for each task along with an overall satisfaction rating and pervasive experience factor.

The initial benchmark usability target values seen in Table 2 have been used to as a base measure for the testing of the application. Ten test subjects assisted with the testing of the application and the mean data from the test is presented in Table 3.

Task	Effectiveness An unassisted task completion rate of at least the following is required	Efficiency Under the following maximum user time on task is required or between boundaries	Satisfaction A rating of the same or above is required	Pervasive Experience Factor A rating of the same or above is required
Learn how to play the game	100.00%	00:03:56		
Play a new game	100.00%	00:11:01		
Setting the game options	95.50%	00:01:51		
Continue a game	100.00%	00:05:41		
Explore the game pages	97.50%	00:02:10		
Configure and play the game on a difficult setting	100.00%	01:00:50		
The application overall			97.5	86

Table 3: Usability criteria values

See [appendix N](#) for the Usability Data.

See [appendix O](#) for the SUS levels.

See [appendix P](#) for the Pervasive Experience Factor levels.

The usability levels resulting from the tests all exceed the benchmark values and there has been a considerable amount of positive feedback recorded from the test participants.

Effectiveness

The benchmark unassisted task completion rate fixed at 100% for tasks 1, 2, 4 and 6 have been achieved by all test subjects. These have been highlighted as core tasks of the game which all users are required to fully perform for the game to play successfully.

Based on the data from the tests this is the case, as an unassisted task completion rate of 100% clearly validates the game play mechanism.

The unassisted completion rate for tasks 3 and 5 had been defined during benchmark testing as 90%. The test results of 95.5% and 97.5% both exceed the required level.

The test administrator felt (post testing) that these results would raise to 100% after extended use of the application as the learning curve was very low. It had been noted during testing that the test participants were fully proficient with the system by the end of the test process and that the test participants all felt the application was very easy to use once learned.

During the testing it had been noted by the test administrator that all issues with system navigation involved the initial recognition by test participants that the ... icon represented expanding the menu. This being a convention of the metro environment identified an initial unfamiliarity with the Window 7 phones operation. Once this had been explained none of the test participants had any other navigation issues.

Efficiency

The target values for efficiency set during the benchmark testing for tasks 1 (4:54), 3 (2:02), and 5 (2:49) represented the maximum amount of time a user should take to complete each task. The results from the tests all came in significantly under these benchmark values at task 1 (3:56), task 3 (1:51), task 5 (02:10) indicating the test participants interacted with the application in a very efficient manner. Higher than the expectation of the internal development team.

The benchmark values for tasks 2 (0:07:00 – 0:14:00), 4 (0:02:30 - 0:08:00) and 6 (0:30:00 – 1:20:00) represented target ranges (in-game play time) for completing level tasks. Pre-testing the internal development team had hoped the final test data would sit mid-range. This would indicate that the correct balance had been achieved, to provide challenging game play at increasingly harder levels, without the game becoming too tedious for the player. The test data indicated that this was actually the case as all the values (tasks 2 11:01), (task 4 5:41), (task 6 1:00:50) had been recorded as mid-range values.

Satisfaction

A benchmark satisfaction level of 55% had been set pre-testing. A satisfaction level of 97.5% had been returned on completion of the tests. This was an unexpectedly high result and for the development team it made all the hard work put into the implementation of an easy to use and intuitive application worthwhile.

Data and feedback from the tests indicated that the participants had found the application intuitive, easy-to-use, fun (they all wanted to know the release date) and surprisingly uncomplicated for a mobile phone game.

User Feedback

Overall game play received very positive feedback, in fact, two of the test participants demanded to play the game in their own configuration (they were fully engaged in the game play) and one test participant still has one of the test phones (to let his children use on their two week holiday driving around the South Island).

The following feedback from the test participants reinforces the proposition that Zombie Gold has been a successfully implemented application which conforms to the intended user's usability requirements:

- All test participants were satisfied with the content and play duration.
- The test participants who undertook the tests in the car thought the game was fantastic and were fully engrossed.
- Search mode features noted very positively included, the tilt effect generally liked, ghost generation and one of the home test subjects made the room dark and became absorbed in the spookier sound effects.
- Attack mode features noted very positively included, shot action (better than angry birds), attack formations, bouncing and swerve on the shots.
- The auto save feature was generally liked. With feedback on the lines of, means you can take a rest anytime you want and the game saves, whatever state they put the phone in the application was in the same state when you returned to it.
- All the test participants liked the style of interaction and partially the menu system, settings layout and scroll text in the help display.

During the testing a couple of suggestions were addressed to the development team for further inclusion:

- One participant indicated a preference for the help text as a sound file activated when clicked.
- Pictures and diagrams in the help topics should be used.
- More graphics sounds and attack types in the next version please.

Pervasive Experience Factor

The benchmark pervasive experience factor level of 55% had been set prior to testing with the expectation of the game showing pervasive characteristics at this level. A fuller immersion into the pervasive experience was expected for the player at a PEF level of around the seventies.

The results from the tests returned a PEF level of 86, indicating a fully immersive pervasive experience for the players was achieved. This was also substantiated by test participant feedback that had been recorded during the tests.

Several points scored practically highly on the PEF scale with all test participants agreeing strongly with the following:

- The game world goes wherever the player goes.
- The game can be played when the player wants.
- A real world environment is used as a setting to stage a game.
- The game links 3D (real world) and 2D (virtual world) environments.
- Features of the phone are used as triggers in the game.

This indicates that the intention to create a pervasive game that bridges the conventional game boundaries of real world and virtual world has been realised. It also indicates two of the main Pervasive game elements, spatial and temporal game play strategies have been implemented successfully in the Zombie Gold application.

Feedback gathered during discussions with participants as part of the testing process focused exclusively on the pervasive experience provided by the game and several aspects of the experience stood out for the participants, again collaborating the proposition that the game has been implemented successfully.

The in-game movement between real and virtual domains figured as the main point of interest for the test participants eliciting the following feedback:

- When you move the phone around it's like you are seeing things that are invisible in the room.
- You need to play the game all around the house, things then seem to pop out of the background.
- When you switch the lights of it makes the place really spooky.
- The music and graphics shock you sometimes, you are in a location and it all fits together, strange.
- You sort of become a child again; it's like catching butterflies in the garden with a net.
- The game jumps between outside and inside the game, you get lost in it at times.
- As you move the world moves and things pop up.
- When you are catching gold and shooting monsters you need to move the phone around your body and move about the room, it is fun.
- In the car it was awesome; the game had stuff flying around all over the place as we drove about.
- There is this tilt thing that if you move the phone too quickly, the glass breaks and it buzzes and stuff.
- When you are playing you start to think of weird places you could play to get people going.

Conclusion

One of the stated goals of the Zombie Gold application is as a pervasive game developed to test a cross section of functionality previously researched in this project on Pervasive Games. Here the application has been successful based on the usability data provided from the tests. The application successfully utilises a cross section of the functionality from the prior research in this project and provides an entertaining game in the pervasive style as proof.

Appendix F: Ethics Approval and Statement


Ray Scott



This is to confirm that Paul Morris a tutor on the Diploma in Applied Computing and was able to undertake testing on mobile applications here in the Labs of the Computing department of the Polytechnic.

The testing was allowed in the labs as it did not breach any regulations and was in compliance with polytechnic procedures, ethics and recommendations.

Yours Ray Scott


Group Leader (IT)
Bay of Plenty Polytechnic

The following statement was presented to all participants who undertook the testing process.

Ethical Statement

Privacy, confidentiality and anonymity of participant data

No personal data will be solicited (or is required) from participants involved in the testing.

It is preferred that participants remain anonymous and during the test process (and documentation) will be referred to as Participant 1 to 10.

All notes taken during the testing relate exclusively to the usability of the application and the feedback expressed by participants on the use of application features and pervasive game elements.

This document is to be kept by the participant along with any data that is deemed personal on completion of the testing.

Informed consent and limitation of deception

Participants are only allowed to volunteer after being fully informed of the testing process and all testing documentation, procedures and scenarios are presented (by Paul Morris at this meeting) prior to the consent of the participants.

Minimisation of risk

Testing will be conducted in usability labs set up in safe locations.

The testing process will also be monitored by a test administrator (Paul Morris) whose first concern is the safety and comfort of participants both physically and emotionally.

Addressing the risk of exploration and the respect of property rights

All data from the testing relate exclusively to the usability of the application Zombie Gold (a pervasive game). The data once correlated will be presented in a project by Paul Morris (Exploration of Pervasive Games in Relation to Mobile Technologies) a copy of which will be sent to you on request.

Social and cultural sensitivity

All data from the testing relate exclusively to the usability of the application Zombie Gold (a pervasive game). The demographic for the application is the gaming community as a whole and does not focus on (or exclude) a selective demographic. The testing process, data elicitation and application interaction relate to commercially acceptable processes currently validated for testing application software NIST/IUSR – CISU-R Common Industrial Specification for Usability – Requirements format.

Appendix G: Usability Evaluation Data Template

Template used to record data for a usability evaluation on the Zombie Gold game.

Usability Evaluation Data for Zombie Gold Application: Participant

Task	Time to complete task	Percentage of task completed unassisted	Percentage of task completed assisted	Number of assists	Number of errors
Task 1: Learn how to play the game					
Task 2: Play a new game					
Task 3: Setting the game options					
Task 4: Continue a game					
Task 5: Explore the game pages					
Task 6: Configure and play the game on a difficult setting					

Notes for task 1	
------------------	--

Notes for task 2	
------------------	--

Notes for task 3	
------------------	--

Notes for task 4	
------------------	--

Notes for task 5	
------------------	--

Notes for task 6	
------------------	--

Appendix H: Participant Task Scenarios for Zombie Gold

The following tasks are used for profiling and testing purposes so that the player's requirements and interaction with game elements can be evaluated. The tasks are read to the participants by a test administrator who monitors the users and record their feedback and actions while undertaking each task.

Task 1: Learn how to play the game

You are an accountant called Joe who lives and works in Auckland.

This morning on your way to work you downloaded a copy of Zombie Gold from the Marketplace and have been waiting all day to play the game.

As a veteran gamer you always want to explore the help options first to learning how to configure and play a game properly. On your way home on the bus you decide to learn about the game.

- 1.1 Find the **game icon** and open the game.
- 1.2 Find and select the **Help navigation icon** on the Main screen.
- 1.3 Find out how to **read the Help text** on the Help page.
- 1.4 Read the section on **Game Rules** in the help text.
- 1.5 Read the section on **Settings Page** in the help text.
- 1.6 Exit the game using the **Start Menu** icon on the phone.

Task 2: Play a new game

You are a young housewife called Tina who lives with your husband and two children on batch in Hamilton. You are very busy with the children around all day and have little time to get out during the week.

Your weekends tend to be spent in Tauranga with your parents as this give the family a break at the weekends and a chance for the kids to see their grandparents.

On Friday evening at 8:00pm the family set out for Tauranga. Usually the journey is a nightmare keeping the kids amused as it is over an hour with toilet stops for the children, but you have downloaded a free copy of the Zombie Gold application onto a couple of phones.

The game was configured for the easiest game options when downloaded and is ready to play. Once the kids are strapped in the back of the car, you get the phones out for the children to play with and head off for Tauranga.

- 2.1 Find the **game icon** and open the game.
- 2.2 Find and select the **Start New Game button** on the Main screen.
- 2.3 Find the **search gold icons on the top** of the game screen and **note how many there are** and **the values on each one**.
- 2.4 Play the game looking around / up /down for gold until you catch one.
- 2.5 Tilt the game by moving the game too quickly.
- 2.6 Control a ghost by tapping the screen (move the ghost to centre of the screen).
- 2.7 Allow a ghost to catch a gold sprite (stop tapping the screen) follow it on the screen until it explodes.
- 2.8 Play the game until search mode is completed successfully or the game timer expires.
- 2.9 Exit the game using the **Start Menu** icon on the phone.

Task 3: Setting the game options

Your name is Henry and you are a male employee of the Bay of Plenty Polytechnic, you are 41 years old, married and have one child.

Recently you ripped your Achilles tendon and have found that using a mobile phone game (Zombie Gold) of your son's has helped to alleviate the pain. The only problem with the game is that your son has the game on a difficult setting which is too fast for you to cope with.

Early Sunday morning you decide that a quick play in the garden will help get your foot going for the day.

You sit down on a chair in the garden with your phone and decide to reset the game option to an easier mode before you play.

- 3.1 Find the **game icon** and open the game.
- 3.2 Expand the **menu bar** on the Main screen by touching the ... option
- 3.3 Select the **settings menu list option** to load the Settings page.
- 3.4 Set the following options on the **Settings** page to make the game easier to play:
 - Slingshot or Flick option to un-ticked.
 - Screen Collision option to ticked (this enables the Explode / Bounce option).
 - Explode / Bounce option to un-ticked.
 - Increase the Shot Curve slider to 2.
 - Increase the Number of Shots slider to 12.
- 3.5 Navigate back to the **Main** page by pressing the **Back button** icon on the phone.
- 3.6 Select the **settings menu list option** on the Main page menu bar to load the Settings page again, to see if the setting you previously changed have remained changed (saved).
- 3.7 Exit the game using the **Start Menu** icon on the phone.

Task 4: Continue a game

You are Software Test Engineer called Sid living and working in Auckland.

You spend a lot of time testing phone applications so have access to and an interest in all the new applications on the market. Zombie Gold is your current favourite.

This weekend your two smaller sisters (twins) Holly and Molly have come up for Hamilton to Auckland with your parents.

It is Saturday evening and for a treat you have set up a scary game for your little sisters, by putting up Halloween decorations and coloured Christmas lights around your flat.

You get a couple of phones out that are loaded with Zombie Gold from the last time the twins were over. The games are configured for the easiest game options.

Once the twins are sat on the couch in the living room you start the games and hand them the phones.

- 4.1 Find the **game icon** and open the game.
- 4.2 Check that the splash screen appears while the game loads (record how long it is visible).
- 4.3 Find and select the **Play Game button** on the Main screen.
- 4.4 Find the current **Score on the bottom** of the game screen and **note how many points you have**.
- 4.5 Find the **attack monster icons on the top** of the game screen and **note how many there are and the values on each one**.
- 4.6 Listen until a random **sound** is generated for the background noise in the game.
- 4.7 Play the game looking around / up /down shooting monsters until you **shoot one of each of the three monster types**.
- 4.8 Bounce a shot of the screen edge using a **pullback flick shot**.
- 4.9 Let one of the monsters hit you to generate a screen splat.
- 4.10 Play the game until attack mode is completed successfully or you run out of points.
- 4.11 Exit the game using the **Start Menu** icon on the phone.

Task 5: Explore the game pages

You are a student called Lucas who studies Mathematics in Hamilton.

Recently at lunch time you have been playing a new pervasive game at several locations around the town.

You are a bit of a techie and are determined to check out everything about the game mechanics (how the thing works).

This lunch time you are alone in the park and decide to investigate the game. You want to see how the basic mechanics of the game work by removing some of the graphics and effects that distract you during normal game play.

You take your phone out and switch it on as you start munching your sandwiches.

- 5.1 Find the **game icon** and open the game.
- 5.2 Find and select the **Achievements navigation icon** on the Main screen.
- 5.3 Find the current score for **Level Beginner** on the Achievements page, note the score.
- 5.4 Navigate back to the **Main** page by pressing the **Back button** icon on the phone.
- 5.5 Find and select the **Settings navigation icon** on the Main screen.
- 5.6 Set the following options on the **Settings** page to make the game easier to play:
 - Decrease the **Level slider** to **Beginner**.
 - **Monster Land** option to **un- ticked**.
 - **Real World** option to **un- ticked**.
 - **Tilt Vibrate** option to **un- ticked**.
 - **Motion Bars** option to **un- ticked**.
 - **Timer Bars** option to **un- ticked**.
 - **Game Score** option to **un- ticked**.
 - **Screen Effects** option to **un- ticked**.
 - **Game Overlays** option to **un- ticked**.
 - **Ambient Sound** option to **un- ticked**.
 - **Sound Effects** option to **un-ticked**.
- 5.7 Navigate back to the **Main** page by pressing the **Back button** icon on the phone.
- 5.8 Find and select the **Play Game button** on the Main screen.
- 5.9 When the game page loads check that all the options are off, no effect layers will be displayed.
- 5.10 Exit the game using the **Start Menu** icon on the phone.

Task 6: Configure and play the game on a difficult setting

You are a student called Tom living and studying in Dunedin. Recently you and your friends have taken to playing a pervasive game in the oddest locations you can think of.

The challenge is based on a player succeeding in completing a level at a strange location (or situation) before leaving the location with the game setting set to the hardest difficulty possible. You are the current leader at Level 9 and today you are going for a level 10 attempt. The challenge is to play a level while a friend (another test volunteer) spins a revolving office chair you will be sat on while playing the game.

You take out your phone and switch it on while sitting in the chair.

6.1 Find the **game tile** in the **phone main menu** and open the game.

6.2 Find and select the **Settings navigation icon** on the Main screen.

6.3 Set the following options on the **Settings** page to make the game harder to play:

- Set the **Level slider** to **10**.
- **Monster Land** option to **ticked**.
- **Real World** option to **ticked**.
- **Tilt Vibrate** option to **ticked**.
- **Motion Bars** option to **ticked**.
- **Timer Bars** option to **ticked**.
- **Game Score** option to **ticked**.
- **Screen Effects** option to **ticked**.
- **Game Overlays** option to **ticked**.
- **Ambient Sound** option to **ticked**.
- **Sound Effects** option to **ticked**.

You friend (other test volunteer) reminds you that you also need to set the attack options to difficult, as you are going for a level 10 attempt at the record.

6.4 Set the following options on the **Settings** page to make the game harder to play:

- Shots per Volley slider to 1.
- Small Shot Curve slider to 0.
- Slingshot of Flick option to **ticked** (Slingshot).
- Slingshot Graphic option to **ticked**.
- Screen Collision option to **ticked**.
- Explode / Bounce option to **ticked** (Explode).

6.5 Navigate back to the **Main** page by pressing the **Back button** icon on the phone.

You friend (other test volunteer) begins to spin the chair and move you around the room.

6.6 Find and select the **Play Game button** on the Main screen.

6.7 Play a **game rule path** for the game level, both the search and attack modes, the game length will be dependent on your success (progress a level) or failure (go back a level) playing the level.

6.8 When you have completed the level play, exit the game using the **Start Menu** icon on the phone.

Appendix I: System Usability Scale (SUS)

Please answer the questions by ticking the appropriate box on the right	strongly disagree	disagree	not sure	agree	strongly agree
1. I think that I would like to use this system frequently					
2. I found the system unnecessarily complex					
3. I thought the system was easy to use					
4. I think that I would need the support of a technical person to be able to use this system					
5. I found the various functions in this system were well integrated					
6. I thought there was too much inconsistency in this system					
7. I would imagine that most people would learn to use this system very quickly					
8. I found the system very cumbersome to use					
9. I felt very confident using the system					
10. I needed to learn a lot of things before I could get going with this system					
Rating	1	2	3	4	5

© Digital Equipment Corporation, 1986.

Scoring the SUS scale

SU scores have a range of 0 to 100 and are calculated by summing the score contributions from each item (each item's score contribution will range from 0 to 4) in the following way:

For items 1, 3, 5, 7 and 9 the score contribution is the scale position minus 1.

For items 2, 4, 6, 8 and 10, the contribution is 5 minus the scale position.

Finally multiply the sum of the scores by 2.5 to obtain the overall value of SU.

Appendix J: Pervasive Experience Factor

Please answer the questions by ticking the appropriate box on the right	strongly disagree	disagree	not sure	agree	strongly agree
The real world influences the state of the game play					
The game world goes wherever the player goes					
The game can be played when the player wants					
The game can move from the focus of attention and back again					
The game experience draws the player and non-players into the game					
Situations encountered in the world affect the mood of the game					
All important in game elements are clearly defined					
The mix of real world and game feedback change the mood of the game					
Game play tips are accessible in the game					
A real world environment is used as a setting to stage a game					
The aesthetics of the environment are used to set the mood of the game					
Information on the game is available for both players and non-players					
The player is able to interact in the style of the game					
The game links 3D (real world) and 2D (virtual world) environments					
Features of the phone are used as triggers in the game					

Pervasive Game Play: Do you feel any real life actions overlap into the game play during the course of game?

Spatial Game Elements: How does the game world go wherever the player goes and interacts with the environment?

Temporal Game Elements: How can the game be played in different time frames does it move to the background and back?

Social Game Elements: How does the game experience draws both the player and non-players into the game?

None-Players: What was the reaction of none-players to the game, noticed nothing, intrigued, thinking you were mad?

None-Players: What's in the game to help none-players feel comfortable, understand, alert and safe around game players?

Appendix K: Preliminary Benchmark Usability Data

The preliminary benchmark usability levels have been set by the observation and discussion with selected players from the defined demographic. Players undertook equivalent tasks on current games in the marketplace and the data from these tasks was used set the preliminary benchmark usability levels. Where tasks were unique to the Zombie Gold game and not repeatable, values were defined based on discussions between the development team and the players.

Preliminary Benchmark Usability Evaluation Data: Participant 1

Task	Time to complete task	Percentage of task completed unassisted	Percentage of task completed assisted	Number of assists	Number of errors
Task 1: Learn how to play the game	00:05:01	100.00%	0.00%	0	0
Task 2: Play a new game	00:11:56	100.00%	0.00%	0	1
Task 3: Setting the game options	00:02:10	82.00%	18.00%	2	0
Task 4: Continue a game	00:05:41	100.00%	0.00%	0	2
Task 5: Explore the game pages	00:02:43	95.00%	5.00%	1	1
Task 6: Configure and play the game on a difficult setting	00:57:46	100.00%	0.00%	0	0

Preliminary Benchmark Usability Evaluation Data: Participant 2

Task	Time to complete task	Percentage of task completed unassisted	Percentage of task completed assisted	Number of assists	Number of errors
Task 1: Learn how to play the game	00:04:48	100.00%	0.00%	0	0
Task 2: Play a new game	00:08:25	100.00%	0.00%	0	0
Task 3: Setting the game options	00:01:54	91.00%	9.00%	1	0
Task 4: Continue a game	00:03:06	100.00%	0.00%	0	1
Task 5: Explore the game pages	00:02:55	100.00%	0.00%	0	1
Task 6: Configure and play the game on a difficult setting	00:38:09	100.00%	0.00%	0	2

Preliminary Benchmark Mean Usability Evaluation Data

Task	Time to complete task	Percentage of task completed unassisted	Percentage of task completed assisted	Number of assists	Number of errors
Task 1: Learn how to play the game	0:04:54	100.00%	0.00%	0	0
Task 2: Play a new game	0:10:10	100.00%	0.00%	0	0.5
Task 3: Setting the game options	0:02:02	86.50%	13.50%	1.5	0
Task 4: Continue a game	0:04:23	100.00%	0.00%	0	1.5
Task 5: Explore the game pages	0:02:49	97.50%	2.50%	0.5	1
Task 6: Configure and play the game on a difficult setting	0:47:57	100.00%	0.00%	0	1

Appendix L: Preliminary Benchmark SUS Levels

The preliminary benchmark usability levels have been set by the observation and discussion with selected players from the defined demographic. Players undertook equivalent tasks on current games in the marketplace and the data from these tasks was used to set the preliminary benchmark usability levels. Where tasks were unique to the *Zombie Gold* game and not repeatable, values were defined based on discussions between the development team and the players.

SU scores have a range of 0 to 100 and are calculated by summing the score contributions from each item (each item's score contribution will range from 0 to 4) in the following way:

- For items 1, 3, 5, 7 and 9 the score contribution is the scale position minus 1.
- For items 2, 4, 6, 8 and 10, the contribution is 5 minus the scale position.
- Finally multiply the sum of the scores by 2.5 to obtain the overall value of SU.

The following SU scores were ascertained:

- The SU scores for participant 1 was 57.5
- The SU scores for participant 2 was 52.5
- The mean SU scores was 55

Benchmark System Usability Scales	Participant 1	Participant 2
I think that I would like to use this system frequently	4	4
I found the system unnecessarily complex	3	2
I thought the system was easy to use	3	3
I think that I would need the support of a technical person to be able to use this system	4	4
I found the various functions in this system were well integrated	3	3
I thought there was too much inconsistency in this system	3	4
I would imagine that most people would learn to use this system very quickly	5	4
I found the system very cumbersome to use	2	3
I felt very confident using the system	3	3
I needed to learn a lot of things before I could get going with this system	3	3
Total	57.5	52.5

Appendix M: Preliminary Benchmark Pervasive Experience Factor

The preliminary benchmark usability levels have been set by the observation and discussion with selected players from the defined demographic. Players undertook equivalent tasks on current games in the marketplace and the data from these tasks was used to set the preliminary benchmark usability levels. Where tasks were unique to the Zombie Gold game and not repeatable, values were defined based on discussions between the development team and the players.

PSF scores have a range of 0 to 100 and are calculated by summing the score contributions from each item (each item's score contribution will range from 0 to 4) in the following way:

- Scale position minus 1.
- Finally the sum of the scores is divided by 0.6 to obtain the overall value of PSF.

The following PSF scores were ascertained:

- The PSF scores for participant 1 was 53.33333
- The PSF scores for participant 2 was 56.66667
- The mean PSF scores was 55

Benchmark Pervasive Experience Factor	Participant 1	Participant 2
The real world influences the state of the game play	3	3
The game world goes wherever the player goes	4	4
The game can be played when the player wants	3	3
The game can move from the focus of attention and back again	3	3
The game experience draws the player and non-players into the game	2	2
Situations encountered in the world affect the mood of the game	3	4
All important in game elements are clearly defined	4	4
The mix of real world and game feedback change the mood of the game	3	4
Game play tips are accessible in the game	3	2
A real world environment is used as a setting to stage a game	3	4
The aesthetics of the environment are used to set the mood of the game	3	3
Information on the game is available for both players and non-players	2	2
The player is able to interact in the style of the game	3	3
The game links 3D (real world) and 2D (virtual world) environments	4	4
Features of the phone are used as triggers in the game	4	4
Total	53.33333	56.66667

Appendix N: Usability Data

The usability levels have been set by the observation and discussion with selected players from the defined demographic.

Usability Evaluation Data: Ten Participants

Tasks	Time to complete task				
	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5
Task 1	0:03:50	0:03:04	0:04:51	0:04:04	0:02:59
Task 2	0:11:32	0:10:09	0:13:40	0:14:37	0:07:46
Task 3	0:02:01	0:01:55	0:02:00	0:02:05	0:01:27
Task 4	0:05:37	0:06:22	0:07:41	0:08:12	0:04:35
Task 5	0:02:12	0:01:40	0:02:01	0:02:40	0:02:12
Task 6	0:49:39	1:02:11	1:15:23	1:19:52	1:02:08

Tasks	Time to complete task				
	Subject 6	Subject 7	Subject 8	Subject 9	Subject 10
Task 1	0:04:00	0:03:50	0:04:27	0:04:20	0:04:00
Task 2	0:11:53	0:11:02	0:10:10	0:07:56	0:11:26
Task 3	0:02:01	0:02:09	0:01:51	0:01:09	0:01:52
Task 4	0:06:11	0:03:38	0:02:39	0:04:39	0:07:16
Task 5	0:01:58	0:02:06	0:02:19	0:02:19	0:02:22
Task 6	1:00:44	1:02:19	0:43:42	0:47:08	1:05:17

Tasks	Percentage of task completed unassisted				
	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5
Task 1	100.00%	100.00%	100.00%	100.00%	100.00%
Task 2	100.00%	100.00%	100.00%	100.00%	100.00%
Task 3	82.00%	91.00%	100.00%	100.00%	82.00%
Task 4	100.00%	100.00%	100.00%	100.00%	100.00%
Task 5	90.00%	100.00%	100.00%	100.00%	90.00%
Task 6	100.00%	100.00%	100.00%	100.00%	100.00%

Tasks	Percentage of task completed unassisted				
	Subject 6	Subject 7	Subject 8	Subject 9	Subject 10
Task 1	100.00%	100.00%	100.00%	100.00%	100.00%
Task 2	100.00%	100.00%	100.00%	100.00%	100.00%
Task 3	100.00%	100.00%	100.00%	100.00%	100.00%
Task 4	100.00%	100.00%	100.00%	100.00%	100.00%
Task 5	100.00%	95.00%	100.00%	100.00%	100.00%
Task 6	100.00%	100.00%	100.00%	100.00%	100.00%

Tasks	Percentage of task completed unassisted				
	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5
Task 1	100.00%	100.00%	100.00%	100.00%	100.00%
Task 2	100.00%	100.00%	100.00%	100.00%	100.00%
Task 3	82.00%	91.00%	100.00%	100.00%	82.00%
Task 4	100.00%	100.00%	100.00%	100.00%	100.00%
Task 5	90.00%	100.00%	100.00%	100.00%	90.00%
Task 6	100.00%	100.00%	100.00%	100.00%	100.00%

Tasks	Percentage of task completed unassisted				
	Subject 6	Subject 7	Subject 8	Subject 9	Subject 10
Task 1	100.00%	100.00%	100.00%	100.00%	100.00%
Task 2	100.00%	100.00%	100.00%	100.00%	100.00%
Task 3	100.00%	100.00%	100.00%	100.00%	100.00%
Task 4	100.00%	100.00%	100.00%	100.00%	100.00%
Task 5	100.00%	95.00%	100.00%	100.00%	100.00%
Task 6	100.00%	100.00%	100.00%	100.00%	100.00%

Tasks	Number of assists									
	S 1	S 2	S 3	S 4	S 5	S 6	S 7	S 8	S 9	S 10
Task 1	0	0	0	0	0	0	0	0	0	0
Task 2	0	0	0	0	0	0	0	0	0	0
Task 3	2	1	0	0	2	0	0	0	0	0
Task 4	0	0	0	0	0	0	0	0	0	0
Task 5	2	0	0	0	2	0	1	0	0	0
Task 6	0	0	0	0	0	0	0	0	0	0

Tasks	Number of errors									
	S 1	S 2	S 3	S 4	S 5	S 6	S 7	S 8	S 9	S 10
Task 1	0	0	0	0	0	0	0	0	0	0
Task 2	1	0	0	0	1	0	0	0	0	0
Task 3	1	1	0	0	1	0	0	0	0	0
Task 4	1	0	0	0	1	0	0	0	0	0
Task 5	1	0	0	0	0	0	1	0	0	0
Task 6	1	0	0	0	2	0	0	0	1	0

Mean Usability Evaluation Data

Task	Time to complete task	Percentage of task completed unassisted	Percentage of task completed assisted	Number of assists	Number of errors
Task 1: Learn how to play the game	00:03:56	100.00%	0.00%	0	0
Task 2: Play a new game	00:11:01	100.00%	0.00%	0	0.2
Task 3: Setting the game options	00:01:51	95.50%	4.50%	0.5	0.3
Task 4: Continue a game	00:05:41	100.00%	0.00%	0	0.2
Task 5: Explore the game pages	00:02:10	97.50%	2.50%	0.5	0.2
Task 6: Configure and play the game on a difficult setting	01:00:50	100.00%	0.00%	0	0.4

Notes Taken During Usability Testing

Task 1	<p>All subjects liked the style of interaction and partially liked the menu system and scroll text in the help display.</p> <p>One subject would have liked the text as a sound file activated when clicked.</p> <p>One subject would have liked some pictures and diagrams in the help topics.</p>
Task 2	<p>All subjects were content with the content and play duration.</p> <p>The test subjects who undertook the test in the car thought the game was fantastic as the gold generated very quickly.</p> <p>One of the home test subjects made the room dark and really got into the spookier sound effects.</p> <p>The tilt effect initially caused some concern to the test subjects, but as they played the game and got use to it, comments were very positive. Great, I like the ghost generation feature. More effects for the screen the cracking is nasty.</p> <p>The main errors concerned the subjects pressing the phone buttons (back, menu or search) instead of the screen, this was dependent on the quality of the phone, by chance the cheaper phones with a mechanical button press did not have this issue, it was the more expensive touch screen activated phone buttons.</p>
Task 3	<p>All subjects liked the style of interaction and particularly liked the menu system and setting layout display.</p> <p>There were several requests for help with this task from subject, all with the same issue. The initial recognition that the ... icon represented expanding the menu. This is a convention for the metro environment and identified an initial unfamiliarity with the phones operation. Once this was explained none of the subject had any other issues.</p> <p>All of the subjects noted they liked the options becoming active or not depending on the setting selected.</p>
Task 4	<p>Great feedback in attack mode from the subjects:</p> <p>Major shot action, better than angry birds. Love the attacks in formations. Bouncing and swerve on the shots is killer. More monsters and more shot types and this is a winner. Best prototype I have played, get the graphics, sounds and more attack types in the final version please. Can you put this on my phone please?</p> <p>Again the main errors concerned the subjects pressing the phone buttons (back, menu or search) instead of the screen. There was a lot of positive feedback when the subjects realised that whatever state they put the phone in the application was in the same state when they returned to it (by whatever route).</p> <p>Examples were:</p> <p>Sorry killed it, just a minute, cool it's back to how it was, nice feature.</p> <p>Dumb pressed the back button again, its lucky this reload properly.</p> <p>Like the auto save feature, means you can take a rest anytime you want.</p>
Task 5	<p>All subjects liked the style of interaction and particularly liked the menu system and setting layout display.</p> <p>There were several errors due to the subject becoming familiar with the system and pressing icons and options just for the fun of it and then quickly selecting the correct one/s. The feedback for the subjects being I just wanted to see what would happen. This was seen as a positive process as the subjects were at this point in the test a lot more interested in the application (fully engaging with it) than the test process.</p> <p>The assists were in the form of reminders to go back and select the correct options after the subjects had been playing extensively with the application options. All the subject continued to set options and see what affect it had on game play long after the test was called to halt. This was seen by the test administrators as a very positive element in the evaluation of the user satisfaction with the application.</p>
Task 6	<p>Very positive feedback, in fact one of the test subject has one of the test phones to let his children use on their two week holiday driving around the South Island.</p> <p>There were a couple of errors due to two of the subject demanding to play the game in their own configuration (after they set the test configuration), again this was seen as positive feedback as at this point in the test process these subjects just wanted to play with the application, especially as they pointed out, it's the last test on a difficult setting and we get to spin around, cool, and I need to beat his score.</p>

Appendix O: SUS Levels

The usability levels have been set by the observation and discussion with selected players from the defined demographic.

SU scores have a range of 0 to 100 and are calculated by summing the score contributions from each item (each item's score contribution will range from 0 to 4) in the following way:

- For items 1, 3, 5, 7 and 9 the score contribution is the scale position minus 1.
- For items 2, 4, 6, 8 and 10, the contribution is 5 minus the scale position.
- Finally multiply the sum of the scores by 2.5 to obtain the overall value of SU.

The following SU scores were ascertained:

- The SU scores for participant 1 was 87.5
- The SU scores for participant 2 was 97.5
- The SU scores for participant 3 was 100
- The SU scores for participant 4 was 100
- The SU scores for participant 5 was 95
- The SU scores for participant 6 was 100
- The SU scores for participant 7 was 97.5
- The SU scores for participant 8 was 100
- The SU scores for participant 9 was 97.5
- The SU scores for participant 10 was 100
- **The mean SU scores was 97.5**

System Usability Scales

Questions	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
I think that I would like to use this system frequently	4	4	5	5	5	5	4	5	5	5
I found the system unnecessarily complex	1	1	1	1	1	1	1	1	1	1
I thought the system was easy to use	4	5	5	5	5	5	5	5	5	5
I think that I would need the support of a technical person to be able to use this system	1	1	1	1	1	1	1	1	1	1
I found the various functions in this system were well integrated	4	5	5	5	4	5	5	5	4	5
I thought there was too much inconsistency in this system	1	1	1	1	1	1	1	1	1	1
I would imagine that most people would learn to use this system very quickly	5	5	5	5	5	5	5	5	5	5
I found the system very cumbersome to use	1	1	1	1	1	1	1	1	1	1
I felt very confident using the system	3	5	5	5	4	5	5	5	5	5
I needed to learn a lot of things before I could get going with this system	1	1	1	1	1	1	1	1	1	1
Total	87.5	97.5	100	100	95	100	97.5	100	97.5	100

Appendix P: Pervasive Experience Factor

The usability levels have been set by the observation and discussion with selected players from the defined demographic. PSF scores have a range of 0 to 100 and are calculated by summing the score contributions from each item (each item's score contribution will range from 0 to 4) in the following way. Scale position minus 1. Finally the sum of the scores is divided by 0.6 to obtain the overall value of PSF.

The following PSF scores were ascertained:

- The PSF scores for participant 1 was 78.3
- The PSF scores for participant 2 was 81.6
- The PSF scores for participant 3 was 86.6
- The PSF scores for participant 4 was 90
- The PSF scores for participant 5 was 90
- The PSF scores for participant 6 was 80
- The PSF scores for participant 7 was 83.3
- The PSF scores for participant 8 was 90
- The PSF scores for participant 9 was 90
- The PSF scores for participant 0 was 90
- **The mean PSF scores was 86**

Questions	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
The real world influences the state of the game play	4	3	5	5	4	3	5	5	5	4
The game world goes wherever the player goes	5	5	5	5	5	5	5	5	5	5
The game can be played when the player wants	5	5	5	5	5	5	5	5	5	5
The game can move from the focus of attention and back again	4	4	4	5	4	4	5	5	5	5
The game experience draws the player and non-players into the game	3	3	3	2	4	4	3	3	3	4
Situations encountered in the world affect the mood of the game	3	4	4	5	5	3	4	5	5	5
All important in game elements are clearly defined	5	5	5	5	5	5	4	5	5	5
The mix of real world and game feedback change the mood of the game	3	5	4	5	5	3	3	4	5	5
Game play tips are accessible in the game	4	4	4	5	5	5	4	4	5	4
A real world environment is used as a setting to stage a game	5	5	5	5	5	5	5	5	5	5
The aesthetics of the environment are used to set the mood of the game	4	3	4	4	4	3	4	4	4	4
Information on the game is available for both players and non-players	3	3	4	3	3	4	3	4	3	3
The player is able to interact in the style of the game	4	5	5	5	5	4	5	5	4	5
The game links 3D (real world) and 2D (virtual world) environments	5	5	5	5	5	5	5	5	5	5
Features of the phone are used as triggers in the game	5	5	5	5	5	5	5	5	5	5
Total	78.3	81.6	86.6	90	90	80	83.3	90	90	90

The following qualitative data has been gathered (in the form of comments) during discussions with participants as part of the testing process. These discussions focused on the pervasive experience provided by the game.

Pervasive Game Play: Do you feel any real life actions overlap into the game play during the course of game?

- Yes, when you move the phone around it's like you are seeing things that are invisible in the room, like a Ghostbuster's monster finder.
- The more you move the better chance you have of finding gold.
- Yes, you need to play the game all around the house, things then seem to pop out of the background.
- When you switch the lights of it makes the place really spooky, cool.
- Is this the first game like this, I have never played a game that works on phone and in the real world.
- Yes and it is sort of weird, who made it.
- The music and graphics shock you sometimes, you are in a location and it all fits together, strange, other times you get caught up in the speed thing shooting monsters.
- Yes, you sort of become a child again, it's like catching butterflies in the garden with a net. My children will love this.
- Yes the whole game, it jumps between outside and inside the game, you get lost in it at times.
- I am not sure what the question really means, but the game was fun.

Spatial Game Elements: How does the game world go wherever the player goes and interacts with the environment?

- The whole game does this, that's what the game is about, isn't it.
- As you move the world moves and things pop up.
- When you are catching gold and shooting monsters you need to move the phone around your body and move about the room, it is fun.
- Yes same as the other question, you need to play the game all around the house, things then seem to pop out of the background.
- Yes, in the car it was awesome, the game had stuff flying around all over the place as we drove about.
- There is this tilt thing that if you move the phone too quickly, the glass breaks and it buzzes and stuff.
- I think the game is based on this, it's like you are playing in the world that part of the phone. It's like the phone is changing things in the world.

Temporal Game Elements: How can the game be played in different time frames does it move to the background and back?

- Things get saved automatically.
- When you switch it off the setting are still there when you start again.
- If you stuff it up, other applications load up on the phone, but you can go back to the game and it is still the same as before.
- You can press the back button on the phone, do things then go back to the game.
- Same, you can press the menu or search buttons on the phone, do other stuff then go back to the game.
- You can text, then play, then text, that's a good feature.

Social Game Elements: How does the game experience draws both the player and non-players into the game?

- Not sure we tested it in a car, so we just played.
- Don't know.
- Seeing the other people playing the game makes you curious.
- Makes you want to have a go when you see people chasing around.
- Curiosity, you think what the hell are they doing.
- When you are playing you start to think of way to challenge people.
- When you are playing you start to think of weird places you could play to get people going.
- I don't think it does it is a single player game.
- You could give it to a gang of people and let them loose around town that would be funny.
- It does make people not playing look at you.

None-Players: What was the reaction of none-players to the game, noticed nothing, intrigued, thinking you were mad?







- They noticed nothing we tested it in a car.
- Most seemed intrigued and curious.
- Everyone who could see us was looking at what was going on.

None-Players: What's in the game to help none-players feel comfortable, understand, alert and safe around game players?

- Not sure.
- Don't know.
- Nothing I think.
- The game is on the phone so they see you are playing.
- You are moving with the phone so you are fairly careful what you are doing.
- You see things around you and through the phone so that's OK and you are fairly easy to spot by people who are not playing.

Appendix Q Windows 7 Phone Features

The following is a list of features supported by the Windows 7 Phone platform.

<ul style="list-style-type: none"> • Microsoft's partnership with Nokia (and Acer, Fujitsu, and ZTE) and the financial / technical backing they put behind their products. 	
<ul style="list-style-type: none"> • Some surprising features of the new Mango 7.5 OS (built around a task-centric philosophy) that point to future innovations for the under development Windows Phone Apollo. 	
<ul style="list-style-type: none"> • Friendly interface using tiles and push notifications that looks good, with easy access to all email accounts. 	
<ul style="list-style-type: none"> • Concept of the productive phone services for business etc. with optimised services (hardware / software). • The Hub strategy used for music, business, people etc. 	
<ul style="list-style-type: none"> • Access to business and entertainment applications easily accessible via the App Marketplace, seen as the best place to bring new device to the market not necessarily in the same format as today. • Quality Apps currently only 10,000 so lots of scope for development. • Try before you buy, where you can download Apps for a trial before buying, again interesting for developers. 	
<ul style="list-style-type: none"> • Full Xbox support on the phone, games (XNA) and profile. 	

- Connection to Cloud services.



- The Mesh, bringing together all aspects of games, entertainment, work and communication across all platforms.

