Towards Formalisation of 3D User Interaction: Making Spatial Interaction Verifiable

Jessica Turner
jessica.turner@waikato.ac.nz
University of Waikato
Tauranga, Bay of Plenty, New Zealand

Benjamin Weyers
weyers@uni-trier.de
Trier University
Trier, Germany

Abstract

Interactive systems which incorporate 3D interaction techniques, such as virtual or augmented reality, lack appropriate methods for formalisation to verify and validate their properties. In safety-critical contexts, such as virtual environments used for training users in medical settings, ensuring systems behave as expected is crucial to enabling positive user experiences. In this paper we propose a formalisation that will allow us to express 3D virtual environments, end users and their interactions. This work represents a first step towards creating formalisation methods for virtual reality applications, enabling model checking techniques to ensure systems behave as expected, leading to improved reliability, safety and better user experience in virtual environments.

CCS Concepts

• **Human-centered computing** → *Human computer interaction (HCI)*; **Virtual reality**; • **Software and its engineering** → *Formal methods.*

Keywords

Virtual Reality, Formal Methods, Interactive Systems, 3D environments

ACM Reference Format:
Jessica Turner and Benjamin Weyers. 2025. Towards Formalisation of 3D User Interaction: Making Spatial Interaction Verifiable. In *Companion Proceedings of the 17th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS Companion '25), June 23–27, 2025, Trier, Germany.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3731406.3734976

1 Introduction

Interactive applications which facilitate 3D interaction techniques, such as virtual or augmented reality, require specific engineering methods due to their spatial and multi-modal characteristics. Formal methods have been used in the past to support the engineering of interactive systems [20]. Such methods are primarily used in safety-critical contexts, such as medical applications [19] or aviation [1]. In addition, formal methods have also been used more widely, in adaptability for persuasive systems [12, 13] or user-driven adaption of interaction logic [5]. The use of formal methods enables formal validation and verification of software systems [20].

However, most of the work in this area focuses on the design and engineering of classic 2D user interfaces. When it comes to 3D interaction, such as virtual reality applications, work in this domain is much more sparse. Early work on 3D user interfaces presents taxonomies describing 3D interaction techniques, such as work by Bowman and Hodges [4]. This work does not specify interaction on an operation level but on a task level, which structures the interaction techniques. Similarly, work presented by Navarre et al. [16] or Ying [21] take a more process-related perspective using Petri nets to describe interactions. With this work at hand, we aim to build a bridge between process and task orientated modelling.

In addition, we keep the user as part of the overall model to capture their movements and gestures to describe how their position in the overall environment affects interaction. This is motivated by the observation that 3D virtual environments differ from standard 2D interfaces as they not only include sequences of interactions but also aspects such as the user's physical movement, position of their limbs, and how the (virtual) world responds to these interactions. As these actions are unique to the 3D environment they are not considered in the majority of existing formalisation approaches. However, these aspects are highly relevant in the case of 3D interaction as the movement, position and gaze of a user in a 3D or virtual environment is crucial for successful interactions.

To fill this gap, we propose a formalisation for 3D user interfaces that will allow us to express 3D virtual environments, end users and interactions. This work is the first step towards a framework to enable formal modelling, validation and verification of 3D user interaction and spatial interfaces. The rest of this paper is structured as follows: first we present a formalisation framework capturing the unique aspects of 3D virtual environments; this is followed by a practical use case to formalise a typical virtual environment within The Lab by Valve[1]. Next we present limitations and discussion for further extensions to the formalisation before finishing with future work and concluding remarks.

2 Formalising Virtual Environments and 3D User Interfaces

Being aware 3D user interfaces are also used in non-VR related applications contexts, in this work we will focus on the formalisation of interaction techniques aimed at manipulating a virtual environment and the user being immersed in it. The approach outlined here may be also extendable towards 3D interaction techniques for a physical environment.

[1] See https://store.steampowered.com/app/450390/The_Lab/

EICS Companion '25, June 23–27, 2025, Trier, Germany

ACM ISBN 979-8-4007-1866-3/2025/06
https://doi.org/10.1145/3731406.3734976

With the focus on virtual reality-related applications, we first outline a formalisation of a scene graph as a data structure representing a virtual environment followed by a description of the user; this is followed by a formalisation of the actual interaction technique and sequence.

2.1 Scene Graph

A scene graph is conceptualised as a hierarchical data structure, structuring objects of different type and their spatial interrelations within a three-dimensional environment [10]. This structure mirrors a tree-like configuration, wherein each node distinctly represents either a singular object or an aggregation of objects (later referred to as group node). The connecting edges within this data structure serve to illustrate the spatial relationships between these nodes. Such a hierarchical arrangement is instrumental in delineating the spatial organisation and interaction dynamics of objects in a 3D scene, providing a systematic approach to managing complex spatial data as well as offering the basis for rendering pipelines. It further reflects the integral input data structure for a rendering, thus visual representation of the scene to be visualised through a display to the user.

The inclusion of a scene graph for a formalisation of 3D interaction is crucial, as it represents the complex spatial relationships and properties of objects within 3D environments with which a user interacts and may be also applicable for physical scenes in future work. This structure is crucial for efficient rendering, interaction design, and overall management of 3D scenes. However, the following formalisation neglects the concept of a "camera" in the scene for rendering purposes and focuses on the formalisation of a user as an integral concept. The level of abstraction presented in the formalisation is open enough to integrate or to relate the concept of a camera object into the scene graph at a later stage.

A *scene graph* $\mathcal{S}$ is an acyclic and directed graph (tree), which can be formally defined as a tuple $S = (N, E, P, U, n_r, \psi, \mu)$, such that

- N is a finite set of nodes, where each node $n_i \in N$ represents an object (elements of N_O), a group of objects (elements of N_G), or a user (elements of N_U) in a geometrical 3D scene, such that

$$N = N_O \cup N_G \cup N_U, \text{ where } N_O \cap N_G \cap N_U = \emptyset. \tag{1}$$

- All nodes in N are organised as acyclic graph, specifically a *tree*, such that E is a set of directed edges with

$$E \subseteq \{(n, m) \in N \times N \mid m \neq n \wedge (n \in N_O \vee n \in N_U) \Rightarrow m \in N_G\}. \tag{2}$$

  The root of the tree is given by $n_r \in N_G$, such that $(m, n_r) \notin E$. Node n is the parent of m, thus, n_r has no parent node.
- Let P be a finite set of geometrical transformations (as homogeneous coordinates), such that

$$P \subseteq \mathcal{R}^{4\times4}. \tag{3}$$

- ψ is a function that maps each node of N to a geometrical transformation in P, such that

$$\psi : N \ni n \rightarrow p \in P. \tag{4}$$

- Let U be a finite set of tuples of user-specific parameters.

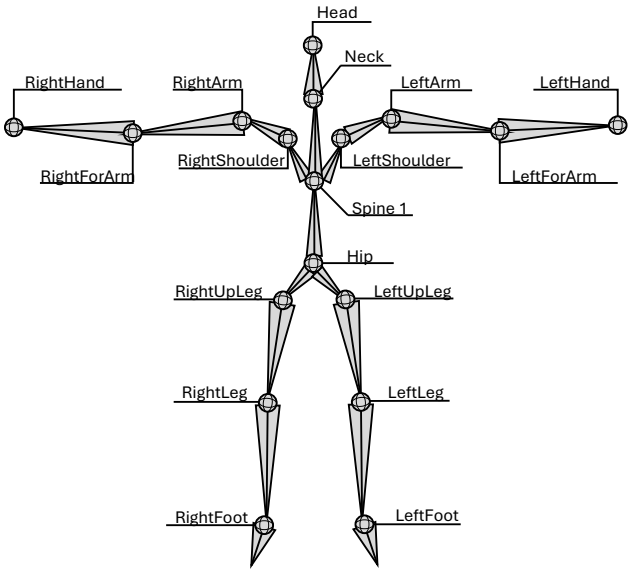

Figure 1: Standard rig for a human avatar which may also represent available sensory information generated by tracking systems for following and digitalizing the human user's movement.

- μ is a function that maps each user node to a tuple of interaction parameters, such that

$$\mu : N_U \ni n \rightarrow u \in U. \tag{5}$$

Each element of U represents parameters needed for formalising interaction operations and techniques. This definition of parameters is flexible and may be extended for future work in the area, depending on the types of environments modelled. For this work, we decided to consider the elements of each tuple representing positional and orientation information of the user's limbs and head considering a standard rig as outlined in Figure 1. In this regard, we consider rigs as an adequate representation of potential sensor and tracking input attached to a user to allow full body tracking system and mapping the user within a 3D scene. In addition to the rig, we also consider a "platform", which may be equivalent to the origin of the user node's local coordinate system in a scene graph. Thus, the platform can be defined as a vector $u_0 \in \mathcal{R}^4$ indicating a global position of the user (and the rig in relation to it) within the world coordinate system, thus the basis of the scene graph. This definition simplifies future work as we can assume that transformation affecting u_0 are bidirectionally mapped to each other. Thus if an interaction operation manipulates the position of the user, this might affect u_0 specifically and its representation in the scene graph.

As a consequence, all $u \in U$ have the following structure. Let r being a rig point, where

$$r = (p, d), \text{where } p, d \in \mathcal{R}^4. \tag{6}$$

The value p represents the position (vector) of r where d represents its direction or normal vector. Depending on the type of interaction

operation, avatar design or tracking system used to measure the user's position, a variation of rigging points may be used. This choice is deliberate, as different virtual environments and user configurations will require a variety of rig points. Therefore, we do not explicitly define this in our above definition and instead let this be an aspect for formalisation of interaction operations. In addition, in most systems rigs for avatars are hierarchically organised as it is the case for scene graphs. Thus, we assume that the resulting position and orientation of rig points are similarly organised as nodes in the scene graph. This makes sense as the position of the hand depends on the position and orientation of the lower arm, which depends on the position and orientation of the upper arm and so forth. For simple VR setups where only the hands (via tracked controllers) and the position/orientation of the head (through a tracked head set) are known, we would design this as a rig containing only three independent rig points mapped to the information coming from the tracking system. Overall, a rig-based representation of a user can be defined as set R of all rig points r_i of a given rig.

In addition to the positional and orientational information of the user's limbs, the user is potentially equipped with (3D) interaction devices. These devices may be comprised of a position, orientation, as well as a set of discrete and continuous input and output channels. Thus, an interaction device id may be defined as follows.

$$id = (p, d, C_d, C_c) \tag{7}$$

Where p and d are defined similarly to rig points. C_d defines a set of discrete and C_c a set of continuous input channels. The primary difference between a discrete channel and a continuous channel is that a discrete channel either has the *true* or *false* state where a continuous channel holds a number value, potentially normalised between 0 and 1. All interaction devices are comprised into a set ID of interaction devices.

Thus, u is defined (for this work) as

$$u = (u_0, R, ID). \tag{8}$$

2.2 Interaction Sequences

In 3D user interactions, there are 3 key interaction techniques: selection and manipulation, in which users select objects to interact with and manipulate in terms of moving positions, changing size and so on; travel, in terms of users moving throughout the 3D environment to different positions; and lastly system control, in which commands are issued to the system to ask it to perform a particular function or change interaction mode and system states. In [11] the authors provide an overview of the techniques that can be found in existing virtual environments. In this work, we will focus on selection and locomotion techniques as highlighted in the use case outlined below.

For each VE, I is the set of all possible interactions in that environment. It can be composed of all types of interaction techniques, methods and scene manipulations, which are related to the user's behaviour. In general, we define two functions pre and $post$, which outline the pre- and post-conditions of each interaction describing under which circumstances (pre-condition) the interaction is applicable and what the outcome of the interaction is (post-condition). This approach follows the general idea of specification languages such as Z [8], which enables the verification of systems without an actual implementation. A more detailed definition of pre and $post$ will be given below.

For any node n in a scene graph $\mathcal{S}$ each node (including the user who is also modelled as node) has a set of interactions n_I such that

$$n_I \subseteq I \tag{9}$$

Within a VE a user may wish to complete a task which requires them to interact with multiple objects. Therefore, we must know the interactions available to the user, the possible interactions for each node they wish to interact with and the sequence of steps required to complete the task. In this regard, task models may be derived by applying task modelling methods such as those outlined in [2].

Let T be the set of all possible user tasks in a VE. Each task t is represented by an interaction sequence such that

$$t \in T \tag{10}$$

Each task t is represented for a given scene graph $\mathcal{S}$ as a tuple as follows

$$t = (n_u, O, T_I, \delta), \tag{11}$$

where:

- n_u represents a user node, the person who wants to complete the task such that $n_u \in N_u$.
- O represents the set of object nodes in the task such that $O \subseteq N_o \cup N_g$.
- T_I is the set of interactions for the objects in O such that $T_I \subseteq I$.
- δ is a set of triples which defines an interaction sequence. That is, given objects $o, o' \in O$, interaction $i \in T_i$, we can denote each triple as (o, i, o'). Object o' is reachable from object o with interaction i if i is a valid interaction for this node, that is a user can interact with each object in a sequence, such that:

$$\delta = \{(o_1, i_1, o_2), (o_2, i_2, o_3), ..., (o_{n-1}, i_n, o_n)\} \tag{12}$$

Building upon the previous outlined formalisation, we can now specify the definition of pre and $post$ for each interaction i_n, a given triple (o_j, i_n, o_k) and a user n_u as follows, where pre_{i_n} tests o_j and the user n_u whether the interaction i_n is currently applicable and $post_{i_n}$ describes the effect i_n has on o_j eventually resulting in o_k.

$$pre_{i_n} : (o_j, n_u) \rightarrow \{true, false\} \tag{13}$$

$$post_{i_n} : (o_j, n_u) \rightarrow o_k \tag{14}$$

3 Case Study: The Lab

To demonstrate the applicability of the above formalisation we applied this to an existing virtual reality application. The Lab is an interactive system developed by Valve and freely available on the Steam platform for users to download[2]. The premise of the application is that it offers a variety of unique ways for users to interact in a virtual environment. The application includes 8 key sub-games including: slingshot, longbow, xortex, postcards, human medical

[2]See https://store.steampowered.com/app/450390/The_Lab/

scan, solar system, robot repair, and secret shop. For the purposes of this case study we focus on some of the unique interactions for virtual environments for travel and selection that best illustrate the use of the above formalism.

3.1 Interactions

For users to make selections in the game there are a variety of ways they can interact. Each user has two controllers (depending on the headset used), one for both their left and right hands. Users may interact by: raycasting, in which a button is selected by using a laser beam emitted from their controller and the user confirms their selection; direct touch, hand tracking allows users to "press" buttons with their virtual hands; and proximity-based activation, where a user moves close to or hovers their controller over an interactive element.

In addition to their selection options, users may travel throughout the environment using a teleportation system. In The Lab environment a visual arc or path appears to allow users to determine where they will land. The arc can be manipulated directly by the user and may be adjusted in real time. When a user confirms their selection they are moved instantly to that location.

Lastly, the game includes an interaction through the use of orbs to teleport to different areas within the virtual environment. Players are presented with a glowing orb which can be targeted for teleportation. The user must pick up the orb and hold the orb to their head in order to activate the teleportation. This will allow them to move to a different area seemlessly using a cinematic cut from the scene to black and into the new scene again, similar to a loading screen in computer games. The orbs serve as a destination marker, guiding the player to where they may want to go in the environment. This interaction enables quick travel in large virtual environments, almost similar to hyperlinks in a web application.

3.2 Task: Navigating into a New Environment

As the orb interaction has been strongly promoted by The Lab, we will use this as the focus point for our formalisation. The task we wish to model is when a user approaches an orb and selects it to move into a new environment. This involves the user first travelling to the orbs location, picking up the orb, then moving the object towards their head to trigger the travel function. The user knows the task has been successfully completed when the transition screen appears and the new area finishes loading. We can informally describe this interaction sequence as follows:

(1) User travels to area near orb.
(2) User selects and holds the orb.
(3) User moves the orb near head position.
(4) User is teleported to new area in the scene graph.

Next we describe modelling the virtual environment using this task to define the scope of our models.

3.3 Modelling the Virtual Environment

For this task the lab environment can be defined as a Scene Graph $S_{Lab} = (N_{Lab}, E_{Lab}, P_{Lab}, U_{Lab}, n_{r_{Lab}}, \psi_{Lab}, \mu_{Lab})$ such that:

- $N_{Lab} = \{n_r, n_u, n_orb\}$ where $n_u \in N_{U_{Lab}}$, $n_{orb} \in N_{O_{Lab}}$, and $root \in N_{G_{Lab}}$.
- $E_{Lab} = \{(n_r, n_U), (n_r, n_{orb})\}$
- $P_{Lab} = \{p_{n_u}, p_{orb}\}$ such that $p_{n_u} = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}$ and $p_{orb} = \begin{bmatrix}100 & 0 & 0 & 1\end{bmatrix}$.
- $\psi_{Lab} = \{n_u \rightarrow p_{n_U}, n_{orb} \rightarrow p_{orb}\}$
- $U_{Lab} = \{u_{lab} = (u_0, R, I))\}$ where $R = \{r_{LeftHand}, r_{RightHand}, r_{Head}\}, ID = \{id_{Left} = (p, d, C_d, C_c), id_{Right} = (p, d, C_d, C_c)$
- $\mu_{Lab} = (n_u, u_{lab})$

In this formalisation we focus on only describing the virtual environment that is related directly to this task for the user. Of note in our above definition is the way in which we have specialised user parameters for this task. In this case we have defined 3 key rigging points for our end user, these are the left and right hands ($r_{LeftHand}$ and $r_{RightHand}$) as well as the head (r_{Head}). You may also refer to Fig. 1 for identification of the corresponding rig points. These rig points are important as the user will need to pick up the orb in the virtual environment and move it within the vicinity of their head rigging point to trigger the travel interaction. Similarly, the interaction devices for this task are also modelled with a device for both the left and right hands (id_{Left} and id_{Right}).

We define our travel task as defined in section 3.2 above as $t_{orb} = (n_u, O_{orb}, T_{orb}, \delta_{orb})$ as follows:

- n_u is simply our end user in the scene graph S_{Lab},
- $O_{orb} = \{n_{orb}\}$ where n_{orb} represents the orb the user will interact with to travel to a different area in the virtual environment,
- T_{orb} = {select, nearby, travel}, where each of these operations are defined formally in a specification,
- $\delta_{orb} = \{(n_u, travel, n_{orb}), (n_{orb}, nearby, n_u), (n_u, select, n_{orb}), (n_{orb}, travel, n_u), (n_u, travel, n_u)\}$

This definition identifies the relevant user, interaction and objects for this task and the associated interaction sequence. Note that for each of the interactions in T_{orb}, namely travel, nearby and select, we do not specify these in this definition. We anticipate that a Z specification or similar can be used to describe these behaviours, building on similar approaches by Bowen and Reeves [3, 9]. For space concerns we omit these definitions here as we focus on a preliminary demonstration of the formalisation and its potential.

For each triple in δ_{orb} we must define a set of corresponding pre and post conditions as follows:

(1) $\{p_{n_u} = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}\}(n_u, travel, n_{orb})\{p_{n_u} = p_{orb} \pm x\}$ describes how the user's position changes to within a distance of x to the orb.
(2) $\{p_{orb} = \begin{bmatrix}100 & 0 & 0 & 1\end{bmatrix}\}(n_{orb}, nearby, n_u)\{p_{orb} = p_{n_u} \pm threshold\}$ describes how the orb must be within a specified threshold to the user to enable interaction.
(3) $\{n_{orb}select = false\}(n_u, select, n_{orb})\{n_{orb}select = true, (p_{orb} = r_{righthand} \vee p_{orb} = r_{lefthand})\}$ describes that on the selection interaction the select value for the orb moves from false to true. The orb's position also changes to the user's right or left hand position in the rig.
(4) $\{p_{orb} = r_{righthand} \vee p_{orb} = r_{lefthand}\}(n_{orb}, travel, n_u)$ $\{p_{orb} = r_{head} \pm threshold\}$ describes how the orb should be in either the right or left hand of the user and that the user has travelled to a location within the threshold for the head rigging point of the user.

(5) $\{n_{orb}active = true\}(n_u, travel, n_u)\{p_{n_u} = \begin{bmatrix} x & y & z & 1 \end{bmatrix}\}$ describes how the position of the user changes to the new location within the new area in the virtual environment.

Next we describe our insights on verification and validation, using the above formalisation to demonstrate directions for future work.

3.4 Insights on Verification & Validation

Each of the above pre and post conditions for our interaction sequence gives us the information required to confirm the correct interaction has taken place and that the virtual environment has been modified accordingly. This enables us to define conditions for a model checking process as with other specification languages [7]. Let us consider each of the steps 1-5 for the model above. Firstly, on step 1 of the sequence we are able to confirm that the user's position within the scene graph has been modified. This is important as the travel interaction should enable the user to move freely within the virtual environment. In this case the movement is tied to the position of the orb as we wish to enable interaction with this object. If the position is not near the orb we know that the task has failed as the user must be within the vicinity of the orb in order to enable interaction. In this regard, one may also consider to integrating concepts such as interaction to enable the repetition of the task and interaction to reach a certain condition. This may be derived from task modelling approaches such as CTT [18] or Hamsters [17].

For the user to be within reach of the orb we must consider what reachability means within a 3D environment. Unlike in 2D environments where a button may be "enabled" or "disabled" to determine if it is reachable, interactions in the 3D world rely on more complex conditions. On step 2 of the interaction the user must determine if they are "nearby" the orb, the pre and post conditions define what this means in the context of this virtual environment. That is, given the position of the orb at a specified location as stated in the pre-condition, we know the orb is reachable by the user if its location is within a specified threshold of the user's position. If this condition is true, the orb should be reachable by the user, if false the task fails as the user cannot select the orb.

On step 3 of the sequence the pre and post conditions define whether or not the orb has been selected. Note that if the select function fails here it is because the orb was not reachable by the user as per our previous step. Once the orb is selected the user is able to move the orb, resulting in a change in the orb's position. In step 4 we see that the orb should travel to a location near the user's head to trigger the travel movement of the user to a new area in the virtual environment. This highlights the importance of location within the 3D environment and how interactions are closely tied to user's positions and movements. While the user is holding the orb they may change its position continuously, resulting in a virtually infinite number of possible locations. Model checking this condition will allow us to check boundaries of movement to ensure travel only occurs when the appropriate conditions have been met. This is important as unexpected travel within the environment could lead to disorientation and even motion sickness for end users [14].

Lastly step 5 defines the reachability within the scene for the end user, that is that the user is able to move to another location within the virtual environment by interacting with the orb. Navigation within 3D environments is crucial to ensure that users can move between different points quickly and efficiently [6]. Ensuring correctness of this behaviour is necessary to enable a positive user experience and provide the user with freedom to explore the virtual world.

4 Discussion

In the above section we describe each interaction at a high level of abstraction, focusing on the connections between the end user and various objects in the 3D environment, leaving more complex functionality to be expressed in a specification. It may be that there are important details expressed in the specification that could be explored in the user interactions to enable further verification and validation techniques. For example, we may wish to explore more than position parameters but also responses to interactions, such as the orb glowing when the user picks up this object. Expansion of the formalism will allow us to include these types of parameters and by laying the groundwork in the user and object parameters, this will enable easy extension of the definitions.

In section 3.4 we described how the pre and post conditions could be used for model checking to ensure that the user is able to complete the task as expected. However, using and expanding this formalism will enable the exploration of further properties such as deadlock and livelock. Deadlock occurs when no further action can be taken by the user, resulting in being stuck in the same state, while livelock occurs when a system continues to execute, sometimes in the same loop, but no progress is made. Each of these properties need to be considered in the virtual environment. For example, we can envision deadlock in a virtual environment occurring when a user is unable to leave a particular area within the scene. Conversely, livelock would occur when the user is unable to leave a particular part of the scene and unable to make progress towards their task. Model checking the system for each of these properties would ensure correctness of the behaviours and result in a more positive user experience, reducing the frustration users may encounter if these scenarios occurred.

In this paper we have focused on virtual reality applications and designed the formalisation from this perspective. As a result, we made a deliberate choice to use rigging points to describe the end user. However, in other 3D environments, such as augmented reality applications, these rigging points may not be necessary depending on the types of interactions used. In the same way the formalisation should be extended to enable more complex interactions and consequently model checking properties, we may also simplify and adapt the formalisation to cater for these variations in the virtual environment. This is important as often AR and VR applications may be created in tandem for training and other safety critical scenarios to enable more users to have access to the software (see [15] for some examples).

When compared with formalisations for interactive systems in 2D environments, most formalisms hide the user completely, in terms of not modelling the exact mouse clicks or positions of the mouse on the screen. Such information does not add anything to the model for verification and validation purposes, and is therefore abstracted. However, as we have explored in our case study above,

this is crucial in a 3D environment as these have significant impacts on how users interact. In order to avoid modelling users themselves, we have chosen to represent them as rigging points. This is a limitation of our modelling approach as this may not capture all of the nuances of interactions and interaction devices, especially as the formalism is expanded. However, by following the taxonomy of interactions as described in LaViola Jr et al. [11] we have provided a basis for this abstraction level which enables a starting point for expansion.

Furthermore, in a virtual environment we may wish to investigate how different objects interact not only with the end user, but also with each other. This is very different to standard 2D interfaces, as all widgets are often laid out on a screen and visible to the user. For example, if a user can see a ball but is is partly occluded by another object, we may wish to check how these two objects interact. Model checking sequences between these objects and also with the end user will allow us to eliminate potential issues to improve the user experience as objects behave as intended. That is, we are able to verify that the ball is either reachable or not despite being occluded without relying solely on user testing.

5 Concluding Remarks and Future Work

In this paper we present a formalisation of 3D virtual environments that allows us to express end users and their interactions. This formalisation allows us to express a virtual environment and associated scene graphs in a tree-structure, where each interactive element and user is considered as a node within the directed graph. Users are expressed as rigging points to capture movements and interactions while interactive elements are represented as objects or groups of objects. For simplicity we considered two types of interactions, selection and travel, and formalised related interaction sequences. Lastly, we used The Lab by Valve as a case study to demonstrate the applicability of the formalisation to a "real-world" virtual reality application.

This formalisation represents a starting point for modelling virtual environments and provides the basis for incorporating formal methods techniques to enable verification and validation of such systems. The benefits of this to virtual environments is immense, enabling measures to ensure consistency and correctness and improve the user experience. In addition, as virtual applications are increasingly used for training individuals in safety-critical contexts (e.g. medical, aviation, or safety simulations), verifying the environment's behaviour is crucial to avoid real-world consequences. Formal verification and validation of these systems will enhance trustworthiness and ensure systems behave as expected.

References

[1] Elodie Bouzekri, Alexandre Canny, Camille Fayollas, Célia Martinie, Philippe Palanque, Eric Barboni, Yannick Deleris, and Christine Gris. 2019. Engineering issues related to the development of a recommender system in a critical context: Application to interactive cockpits. *International Journal of Human-Computer Studies* 121 (2019), 122–141.

[2] Judy Bowen, Anke Dittmar, and Benjamin Weyers. 2021. Task modelling for interactive system design: A survey of historical trends, gaps and future needs. *Proceedings of the ACM on Human-Computer Interaction* 5, EICS (2021), 1–22.

[3] Judy Bowen and Steve Reeves. 2017. Generating obligations, assertions and tests from UI models. *Proceedings of the ACM on Human-Computer Interaction* 1, EICS (2017), 1–18.

[4] Doug A Bowman and Larry F Hodges. 1999. Formalizing the design, evaluation, and application of interaction techniques for immersive virtual environments. *Journal of Visual Languages & Computing* 10, 1 (1999), 37–53.

[5] Dina Burkolter, Benjamin Weyers, Annette Kluge, and Wolfram Luther. 2014. Customization of user interfaces to reduce errors and enhance user acceptance. *Applied ergonomics* 45, 2 (2014), 346–353.

[6] Heni Cherni, Natacha Métayer, and Nicolas Souliman. 2020. Literature review of locomotion techniques in virtual reality. *International Journal of Virtual Reality* (2020).

[7] John Hatcliff, Gary T Leavens, K Rustan M Leino, Peter Müller, and Matthew Parkinson. 2012. Behavioral interface specification languages. *ACM Computing Surveys (CSUR)* 44, 3 (2012), 1–58.

[8] Jonathan Jacky. 1997. *The way of Z: practical programming with formal methods*. Cambridge University Press.

[9] Sapna Jaidka, Steve Reeves, and Judy Bowen. 2017. Modelling safety-critical devices: Coloured Petri Nets and Z. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. 51–56.

[10] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. 2015. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3668–3678.

[11] Joseph J LaViola Jr, Ernst Kruijff, Ryan P McMahan, Doug Bowman, and Ivan P Poupyrev. 2017. *3D user interfaces: theory and practice*. Addison-Wesley Professional.

[12] Yuen C Law, Wilken Wehrt, Sabine Sonnentag, and Benjamin Weyers. 2017. Generation of information systems from process models to support intentional forgetting of work habits. In *Proceedings of the acm sigchi symposium on engineering interactive computing systems*. 27–32.

[13] Yuen C Law, Wilken Wehrt, Sabine Sonnentag, and Benjamin Weyers. 2023. Obtaining semi-formal models from qualitative data: From interviews into bpmn models in user-centered design processes. *International Journal of Human–Computer Interaction* 39, 3 (2023), 476–493.

[14] Alexander KC Leung and Kam Lun Hon. 2019. Motion sickness: an overview. *Drugs in context* 8 (2019).

[15] Xiao Li, Wen Yi, Hung-Lin Chi, Xiangyu Wang, and Albert PC Chan. 2018. A critical review of virtual and augmented reality (VR/AR) applications in construction safety. *Automation in construction* 86 (2018), 150–162.

[16] David Navarre, Philippe Palanque, Rémi Bastide, Amelie Schyn, Marco Winckler, Luciana P Nedel, and Carla MDS Freitas. 2005. A formal description of multimodal interaction techniques for immersive virtual reality applications. In *Human-Computer Interaction-INTERACT 2005: IFIP TC13 International Conference, Rome, Italy, September 12-16, 2005. Proceedings 10*. Springer, 170–183.

[17] Philippe Palanque and Célia Martinie. 2016. Designing and assessing interactive systems using task models. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 976–979.

[18] Fabio Paternò. 2004. ConcurTaskTrees: an engineered notation for task models. *The handbook of task analysis for human-computer interaction* (2004), 483–503.

[19] Jessica Turner, Judy Bowen, and Steve Reeves. 2020. Model-based testing of interactive systems using interaction sequences. *Proceedings of the ACM on Human-Computer Interaction* 4, EICS (2020), 1–37.

[20] Benjamin Weyers, Judy Bowen, Alan Dix, and Philippe Palanque. 2017. The handbook of formal methods in human-computer interaction.

[21] Jianghui Ying. 2006. An approach to Petri net based formal modeling of user interactions from X3D content. In *Proceedings of the eleventh international conference on 3D web technology*. 153–157.