UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

# Computer Science Honours
# Final Paper
# 2015

Title: A Virtual Reality Interface for Previsualization

Author: Joshua Ramsbottom

Project Abbreviation: PREVIS

Supervisor: A/Prof James Gain

| **Category** | **Min** | **Max** | **Chosen** |
|---|---|---|---|
| Requirement Analysis and Design | 0 | 20 | 15 |
| Theoretical Analysis | 0 | 25 | 0 |
| Experiment Design and Execution | 0 | 20 | 10 |
| System Development and Implementation | 0 | 15 | 15 |
| Results, Findings and Conclusion | 10 | 20 | 10 |
| Aim Formulation and Background Work | 10 | 15 | 10 |
| Quality of Paper Writing and Presentation | 10 | | 10 |
| Adherence to Project Proposal and Quality of Deliverables | 10 | | 10 |
| Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*) | 0 | 10 | 0 |
| **Total marks** | | **80** | **80** |

# A Virtual Reality Interface for Previsualization

Joshua Ramsbottom
University of Cape Town
joshuaramsbottom@gmail.com

## ABSTRACT

Previsualization is a pre-production process in filmmaking that allows directors to plan shots without incurring unnecessary costs. The increasing power of consumer hardware as well as the increasing popularity of free 3D modelling and animation tools are making filmmaking more accessible. Current previsualization software is based on 3D modelling software and usually employs a traditional WIMP paradigm that restricts users to 2D input and output.

We present a Virtual Reality (VR) based interface for the task of previsualization with the aim of accessibility and usability using commodity hardware. The system places users in a virtual scene and allows them to manipulate objects to create a timeline of key frames by capturing snapshots of the scene. The system uses a head-mounted display to place the user in a scene as the camera, allowing them to make fine adjustments using head movement. We conducted a small-scale qualitative evaluation and found that while there were multiple issues with our interface, it also shows promise in some aspects of previsualization.

## CCS Concepts

• **Human-centered computing** ➞ **Interaction paradigms** ➞ **Virtual reality**

## Keywords

Virtual reality; 3D user interfaces; Previsualization; Head-mounted display

## 1. INTRODUCTION

Previsualization (previs) is process that takes place during the pre-production phase in filmmaking. It is used by directors to visualize various aspects of scenes without the costs of fully producing them. This includes camera placement and movement, and the movement of objects during the course of a shot. More recently, 3D graphics software has been used to aid this process with low-fidelity 3D models. Many programs currently used for previs were originally designed for 3D modelling. These packages often require the skills of a trained animator, whose expertise could be better used in other processes. Traditional WIMP software used for previs also constrains the user to both 2D input devices (mouse) and a 2D interaction window (monitor). This can become problematic for performing tasks which are 3D in nature [10].



**Figure 1. Screenshot of current previs software: *FrameForge* [22].**

With the increasing power of consumer desktop hardware, as well as the rise in popularity of free 3D modelling and animation software, the tools and resources for filmmaking are becoming more accessible to amateur filmmakers and hobbyists. Nitsche [14] argues that the interactive aspects of games combined with storytelling make game engines suitable for previs, and that the ability of virtual reality to allow users to play a character in a film offers new possibilities for previs. In fact there are a number of game engines that support film production (or *machinima*). This means that digital filmmaking is becoming more accessible to people without formal training [14].

We propose a new 3D user interface for previs that aims to allow users with no training to create previsualizations intuitively and efficiently. The interface uses a virtual reality (VR) head-mounted display (HMD) to place the user in the scene. This allows the user to better understand the 3D spatial relationships between objects in the scene. The user interacts with the system using a combination of head movements and a gaming console controller. The controller provides a familiar and simple control scheme. Our system allows users to select objects using their gaze, and then perform 3D translations and rotations on them by changing modes. The user acts as the camera in order to capture key frames as snapshots of their current view. This allows the user to make fine adjustments to the camera shots using head tracking.

There is currently little development of new ways of previs. Our system is novel in that users are given a unique perspective from within the scene using virtual reality. While the existing body of research on VR and 3D user interfaces does apply, other design choices still need to be explored. This applies especially to the choice of input device. Our system uses an Xbox360 controller as an input device, both because of its popularity, as well as its endorsement by Oculus.

Our system focuses on low-fidelity previs, which involves low-resolution models often without textures or rigging. Our system aims at producing a series of still renderings captured by the user.

This is in contrast to high-fidelity previs which normally allows users to plan camera and object movements and render them in real-time. High-fidelity previs also enables basic animation of objects to show gestures and simple body language. This allows more detailed plans to be made.

We made use of an iterative development process which included feedback from pilot users as well as expert heuristic evaluation. Prototypes were used to quickly obtain feedback on design decisions. Finally, a small-scale qualitative evaluation was undertaken to determine the effectiveness of this interface in terms of usability.

The remainder of this paper is structured as follows: Section 2 discusses related work pertaining to both virtual reality and 3D interaction metaphors; Section 3 details the interface design; Section 4 covers the implementation and hardware used; Section 5 describes the experimental methodology as well as the results of a qualitative evaluation; and lastly, Section 6 provides a conclusion as well as possibilities for future work.

# 2. RELATED WORK

While there is little previous work that specifically addresses previs, there is a large body of literature regarding VR and 3D user interfaces for 3D tasks similar to previs.

## 2.1 Virtual Reality

There are a number of examples where a VR-based system was found to improve users' 3D spatial understanding and 3D scene creation. Both of these aspects are important for previs where users are required to: (a) manipulate objects to create a scene and, (b) capture renderings where the relative positions of objects are important for capturing the desired camera shot.

Butterworth et al. [2] developed a 3D modeling tool using an HMD and found that placing users in the virtual scene made it easier to understand the spatial relationships between objects. Users were also able to make fine adjustments to their viewpoints in a natural way due to head tracking by the HMD. This is important for a VR previs application because the user's viewpoint is used to display the current frame for capture. Enabling users to make fine adjustments intuitively is advantageous for this purpose.

Wang and Lindeman [20] presented a level editing system for virtual environments that used a partially occlusive HMD. Users would look forward to see the environment inside the HMD, and look down to see a tablet which gave an overview of the environment as well as a means to interact with the system. The overview gave users another perspective of the environment and the 2D input on the tablet made tasks such as menu navigation and data entry easier. Level editing is similar to previs in that it also involves transforming objects to place them in the desired manner. This is important because of how VR enables users to be placed in the scene, as a better understanding the spatial relationships between objects is equally important for previs. The system presented in [20] also indicates the importance of providing an overview of the environment to users, enabling them to keep track of all the objects in the scene while focusing only on some of them.

Two CAVE-based systems were developed by Hughes et al. [7] and Ponto et al. [16] for creating 3D scenes. Both of these systems allowed users to manipulate 3D objects in order to create a scene. Hughes et al. [7] found that users were able to recreate complex architectural models of entire buildings in under 15 minutes, but it was also found that users required previous knowledge of 3D modelling techniques to use the system effectively. Ponto et al. [16] found that users were able to create complex scenes in under 10 minutes. Both of these systems have similar requirements to previs, where users are required to create 3D scenes by manipulating objects. One difference is that previs also involves camerawork which requires users to move around the scene, unlike in [7] and [16] where a static viewpoint is used. It must be noted that the short time taken to recreate scenes in [7] could be attributed to participants' training in architecture.

## 2.2 3D Interaction Metaphors

User interaction with a 3D interface is classified using metaphors. These metaphors describe how a virtual environment (VE) will respond to user interaction by likening it to a real-world physical response. Jankowski [8] provides a taxonomy of general tasks performed in 3D virtual environments from which these metaphors can be derived. Bowman et al. [1] define three main interaction techniques: manipulation, selection, and travel (similar to navigation).

### *2.2.1 Navigation*

Navigation involves the user changing viewpoint within a VE. This encompasses both general movement with no particular goal, and targeted movement with regard to some specific point of interest. Targeted movement may also involve a set of specific orientations and positions. Ware and Osborne [21] define three metaphors for 3D navigation, namely Eyeball-in-Hand, Scene-in-Hand, and Flying Vehicle Control. The user's hand movements directly transfer to viewpoint and scene movements, respectively, in the 'in-hand' metaphors, and the user's interaction moves a virtual vehicle through the scene in the last metaphor. It was found that in the Eyeball-in-Hand metaphor, users consciously calculate their movements and that this can become disorienting. The Scene-in-Hand metaphor is useful for manipulating objects and changing the viewpoint. This is useful for previs because users will spend most of their time manipulating objects and changing the viewpoint. The Flying Vehicle Control metaphor is most effective at reducing simulator sickness, but prevents the use of head movement for navigation. This is because the user's head movements translate to movements within the 'vehicle' and don't affect its movement.

### *2.2.2 Selection*

Selection is the process that occurs when a user designates which object in a VE will be selected for navigation or manipulation. In 3D VEs a common metaphor for selection is *pointing*. Most 3D pointing techniques fall under two metaphors: 'ray-based' and 'virtual hand' [17, 18]. The main distinguishing characteristic of virtual hand techniques is that they require depth information in order to check for the intersection between the hand and cursor. In contrast, Dang [4] describes a third 'spotlight' metaphor. This is similar to ray-based techniques but a conical projection volume is used to select objects instead of a ray. The conical shape increases the selection area as the distance from the user increases. This mitigates the loss of accuracy associated with objects which are further away [11]. Lubos et al. [11] proposes two guidelines for 3D selection in HMD VR environments. Firstly, 3D selection tasks that require fine movement should be restricted to objects that are close to the eyes. In previs, selection tasks will have to be performed when an object is far away from the desired camera position and so this guideline must be considered. Secondly, an elliptical shaped projection should be used to increase the selection space and reduce errors along the view direction. A plot

of the error points in [11] showed an elliptical shape, meaning that using an elliptical selection shape should reduce errors. The second guideline only refers to the case where spherical objects must be selected, and it is unclear whether this will apply to other objects used for previs.

### *2.2.3 Manipulation*

Manipulation refers to the methods used when changing an object's position, orientation, and scale. In 3D user interfaces manipulation techniques fall under three metaphors. *Using a Manipulator* – virtual handles are attached to an object and displayed to the user and these handles are used to manipulate the object; *Automatic Viewing Control* – the position of the virtual camera is used to augment manipulation; *Constrained Manipulation* – physical aspects of both the world and the objects are used to constrain and simplify manipulation [8]. Using a Manipulator in VR-based applications may increase the difficulty of manipulation because the virtual handles can be seen as smaller objects which also have to be selected. This would add to the number of tasks that must be performed in order to manipulate objects. Automatic Viewing Control has a new meaning for VR-based previs due to the ability of VR to place the user in the scene as the camera. This means that the user's movements as the camera could be used to augment manipulation. Constrained Manipulation is important for previs because objects should not be able to move in unexpected ways (e.g. through a wall or below the floor). It must be noted that special cases may occur where the environment used for previs does not obey natural laws (e.g. objects are able to float through things).

A number of hybrid approaches to 3D manipulation have been proposed, where a combination of 3D and 2D input is used. Gallo et al. [5] develop an interface for manipulating medical scans where a *Wiimote* is used to capture 3D gestures and discrete button input. This interface exploits the advantages of 3D and 2D input using modes where each mode is used for a different type of manipulation. Users are able to perform the currently selected manipulation by waving the *Wiimote* in space and change between modes by pushing buttons on it. In our interface the joysticks on the controller are used to provide pseudo-3D input while buttons perform discrete actions including changing modes. Wang and Lindeman [20] describe a virtual level-editing interface using a non-occlusive HMD and a tablet. The HMD gives users a first-person experience and captures head motion. The HMD also allows users to look down and interact with a tablet on their lap. The tablet provides an overview of the environment and allows 2D touch input. Alternative approaches to displaying an overview for our system are required due to the occluding HMD used. This is less important for low-fidelity previs because all of the objects required for a particular key frame will already be in view for that frame. Mine at al. [12] create another hybrid interface that uses a handheld 3D-printed shell to house a smartphone and a microcontroller. The smartphone captures 2D touch input and the microcontroller is used to capture 3D gestures. This system shares similarities with both [5] and [20]. The input device allows for gesture based 3D input similar to the *Wiimote* as well as more complex 2D input via touch. These hybrid approaches leverage the strengths of both 3D and 2D input, which in this case are 3D manipulation and system control respectively.

### *2.2.4 System control*

System control refers to interactions between the user and the system which are not represented in the VE, such as changing the mode of interaction. Actions such as menu navigation and numeric data entry are difficult to perform using 3D input methods due to the lack of 2D pointing precision compared to a traditional mouse. Data entry is also better suited to a device with discrete buttons for each possible type of input [12, 20]. This is important for previs because certain aspects such as timeline editing are done using discrete actions. It must also be noted that in VR applications head tracking can also be used for menu navigation where the user's gaze selects menu options. For this to be possible the menu must be displayed as a part of the environment.

## 3. INTERFACE

The output of low-fidelity previs is a series of key frames, which is referred to as a 'timeline'. These key frames show important camera shots over the course of a scene in a film, similar to storyboarding. The 3D nature of digital previs and the constraints of a game engine must be considered when designing a previs application in order to enable users to generate a timeline. Most importantly, the user must be able to edit a timeline. This involves the ability to create, view, overwrite, and delete key frames as well as a means to select them for these actions. Capturing a desired key frame involves two aspects: manipulating objects into the desired position and orientation, and moving the camera into the desired position. This means that the system must allow users to perform 3D translations and manipulations on objects, and move the camera while being able to see its viewpoint.

These aspects of previs are all considered in our interface. Firstly, the user is able to view and edit the timeline by shifting back and forth to select key frames and perform actions on them. The timeline is always viewable because it is constantly referred to by users. Secondly, the user is able to move around in the scene as a first-person camera. This allows the viewpoint to be changed so that the scene objects can be viewed in a different way, either for framing the objects as the camera or planning a camera's movement. Thirdly, the user is able to select objects and manipulate them into various positions and orientations to plan an object's movement during a shot. Object and camera movement are derived by interpolating between their positions on consecutive key frames. This core functionality was determined during a preliminary HCI task analysis to be fundamental to the low-fidelity previs process.

Ideally this interface should be kept as simple as possible so that extensive training is not required. This would allow a director with minimal training to create a timeline and more accurately communicate the vision for a scene to others involved in the production process.

Our system uses a modal interface with two modes. The functionality of the system is separated into: (a) navigation, timeline editing, object selection; (b) object manipulation via 3D translation and rotation. This maintains the simplicity of the controls, since the user is not able to look at the input device while using the system. This is similar to Gallo et al.'s [5] *Wiimote* interface where users changed between modes using buttons on the device.

## 3.1 Navigation Mode

This mode includes both the navigational and timeline aspects of previs. In this mode users are able to move around the scene using a combination of joystick movements and head tracking. This allows the user to move along all three axes. The axes are framed relative to the direction the user is facing in the scene so that the user can make fine adjustments to the direction of movement using head movement. The direction the user is facing is framed as the z-axis or the 'forward' direction, and the x-axis is framed

perpendicularly to this. The y-axis is kept constant as directly up or down. This scheme allows for both coarse and fine adjustments to movement, which is important for navigation in a VE [19]. The user makes coarse adjustments by moving the avatar within the environment with joystick movements on the controller, while fine adjustments are made using head tracking.

The Oculus best practices are followed with regards to movement. Movement speed is kept at a pace similar to walking and any acceleration effects on the user's avatar are kept short. Forward movement is considered more natural than backwards or sideways movement and while this movement is possible in our system, forward movement is encouraged by allowing the user to change direction using head adjustments. In any case where the user's avatar must be moved automatically (e.g. the user resets the position to where the selected key frame captured) the screen first fades to black before the movement is carried out. This was done in order to mitigate the effects of *simulator sickness* [15] and provide a more natural user experience.

Timeline navigation and editing are also possible in navigation mode. Our system uses a heads-up display (HUD) to show users the timeline along the top edge of their view. While the Oculus best practices recommend that HUD elements rather be displayed in the environment as objects, this might cause issues with previs because the user often references the timeline. The user may become fatigued from keeping track of the timeline's location in the environment. The timeline is represented as a strip of small images along the top of the display. Each image represents a key frame that the user has captured. The small size of the timeline images allows the user to see enough detail without occluding the objects in the scene. The currently selected gap or key frame is always kept centered in the display, allowing users to easily see their current selection. A red border around the key frame indicates that it is currently selected.

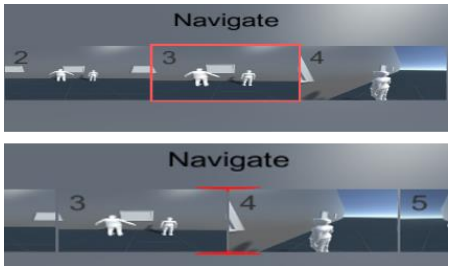


**Figure 2. Timeline screenshots showing both a keyframe and a gap selected.**

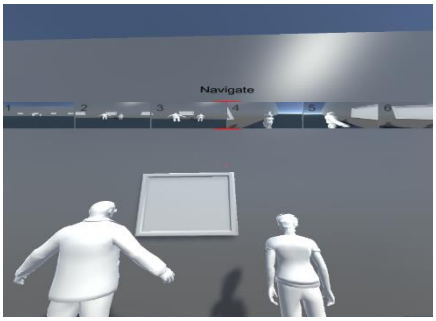


**Figure 3. Screenshot showing the position of the timeline on the HUD.**

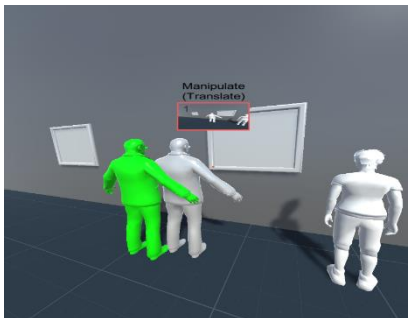


**Figure 4. Screenshot showing a 'ghost' object, indicating that the current key frame is being edited.**

Users are able to move back and forth along the timeline one key frame at a time, each object in the scene is automatically updated to its position and orientation when the selected key frame was captured. Jumping to a specified key frame was not implemented as numeric entry would be difficult given the chosen input methods. Users are also able to select the spaces between key frames in order to insert a new key frame. When this occurs the scene objects keep their current positions and orientations until another key frame is selected. If a key frame is currently selected, it may be deleted or edited. The user is prompted before key frame deletion. As soon as an object is moved the system identifies the currently selected key frame as being 'edited'. A slightly transparent ghost image of the object in its original position and orientation is shown to indicate that editing is active, this only applies to objects which have been moved from their original positions and/or orientations. Our system also allows the user to reset the viewpoint back to the configuration when the key frame was originally captured. This enables the user to make fine adjustments to the camera positioning of a key frame. When this occurs the display first fades to black in order to prevent disorientation.

In order to manipulate an object, that object must first be selected. In our system this is done using a ray-based technique that allows the user to select an object by directing his or her gaze at the object in the environment. This technique is used because it has been shown that 3D selection tasks in sparse environments are easier to perform using ray-based techniques than virtual hand techniques [1, 6]. Visual feedback is provided by highlighting the currently targeted object in red, allowing users to quickly and accurately decide if the right object has been selected [17, 23].

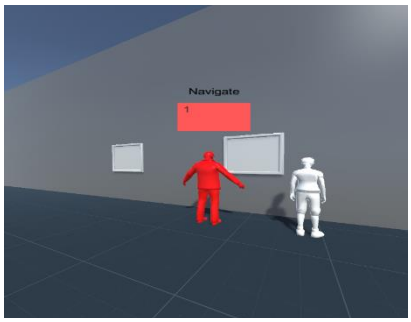


**Figure 5. Screenshot showing an object (highlighted red) that is currently selected.**

A challenge associated with this method of 3D selection is that it becomes difficult to select smaller objects which are far away from the user [11]. In order to mitigate this problem a small reticle is drawn in the center of the user's gaze to display the current target of the ray.

## 3.2 Manipulation Mode

In order to manipulate an object it must first be selected. The user may then choose to enter manipulation mode. The selection is then locked and the user may look away without deselecting the object. The object is highlighted in green and the mode icon in the HUD changes to manipulation mode. These multiple sources of feedback clearly indicate to the user that a new interaction mode is active. This mode consists of two sub-modes: translation and rotation. Translation and rotation were separated into sub-modes so that the movement controls could be reused. In order to make the interaction more predictable the mapping of axes to controls is kept the same between all modes and sub-modes. While in manipulation mode, users may change freely between the two sub-modes by pressing a button on the controller.

Initially the system is in the translation sub-mode. This allows the user to move the selected scene object along all three axes. These axes are kept relative to the direction the user is facing, which should allow for more natural object translations. This allows users to predict the effect of object manipulations using their frame of reference in the environment. It also allows users to partially direct object movement by moving their head towards the desired position. For example, users are able to move an object upwards by looking up while moving an object forwards. In order to prevent users from moving an object out of the scene, translations are restricted. For example, objects cannot be moved below the floor of the environment. The rotation sub-mode is similar to the translation sub-mode, allowing users to rotate the object about the three axes using the same controls. One notable difference is that in this sub-mode the rotation axes are in the selected object's local space. This was done to make rotations more predictable. It was found that rotations are disorienting if the axes are relative to the user's gaze direction.

While in manipulation mode, the system allows users to reset the selected object's position and orientation back to the configuration of the original key frame independently. Once the user switches back into navigation mode the selected object's new position and orientation are locked and the user is free to move around the environment again. Once the user switches back into navigation mode the selected object's new position and orientation are locked and the user is free to move around the environment again. If the user switches back into manipulation mode the system is automatically switched to the previously selected sub-mode.

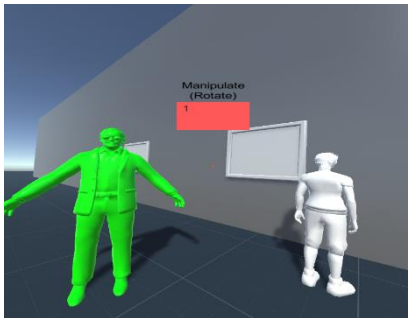


**Figure 6. Screenshot showing an object (highlighted green) that has been successfully selected and manipulated.**

# 4. IMPLEMENTATION

## 4.1 Platform & Hardware

The Unity3D game test engine was chosen as the development platform for the following reasons: (a) it has first-party support for the Oculus Rift VR HMD, (b) it is freely available to use, (c) it allows for rapid prototyping, and (d) it readily supports an Xbox360 game controller as an input device. Unity3D also allows for a modular codebase by separating different functionality into scripts, which are written in C#. In our system each feature is separated into its own script. The Unity3D engine is able to display directly to the HMD without first requiring an executable build file, which accelerates the development cycle. This also allows the system to be tested without the HMD when necessary. Oculus provides default avatar controllers for Unity3D, which incorporate the Oculus best practices, such as minimal acceleration and slow movement speed. Lastly, 3D models from popular 3D modelling software such as Blender can be easily imported into the Unity3D engine.

The Oculus Rift Development Kit 2 (DK2) was used as a HMD for our system. At the time of writing, this is the most recent iteration of the popular VR headset and has a higher resolution display as well as a higher supported frame rate than previous models. These are both factors that can cause simulator sickness if too low. It is also best to support a readily-available commercial VR product as one of the aims of our system is a low barrier to entry. The Oculus Rift supports head tracking, which is required for selecting scene objects as well as user and object movement.

An Xbox360 game controller was used as the input device. Due to its widespread adoption, it might already be familiar to potential users and they would require less training. This game controller is also widely available, which lowers the requirements for our system. In 2016, the commercial version of the Oculus Rift will be packaged with an Xbox game controller, which means that anyone who has purchased one will be able to use our system. Due to the HMD occluding views of the real world users will be unable to look at the input device and may be unable to determine what is required to perform a desired action. To overcome this, users are able to look down while wearing the HMD and a diagram of a controller with the relevant mode's control mapping will be displayed. This provides scaffolding for novice users.

## 4.2 Control Scheme

The Xbox game controller was also chosen because it has a smaller number of controls, but nevertheless includes both discrete buttons and joysticks. We reserve the joysticks for axial movement in all modes of the system, where the left joystick controls both the X- and Z-axes and the right joystick controls the Y-axis. In navigation mode this controls user movement within the scene, with the right joystick allowing for 'flying' movement. This is required to make aerial camera shots possible. In manipulation mode, the joysticks either translate or rotate (depending on the selected sub-mode) the selected object using the same axial mapping. The joysticks allow finer control of movement as they vary magnitude depending on how far the joystick is displaced from its origin. They also allow movement in any direction, including diagonally. The buttons on the controller are used for discrete actions: changing of modes and timeline editing.

The direction of the user's gaze from the HMD's head tracking is used to augment both user and object movement. This is done by determining the three axes of movement relative to the user's gaze

direction, where the Z-axis is matched to the user's forward-facing direction.

## 4.3 Timeline Management

There is a set of information that must be associated with each key frame on the timeline. For each scene object the position and orientation in each key frame is stored. The same applies to the user's position and orientation. This information, along with a reference to the associated key frame, is stored in a set of lists where a separate list is used for each attribute. Key frame images are also stored locally on disk. Each key frame is numbered according to its position so that key frame information can be accessed directly using the key frame's index. Users are able to select both key frames and the spaces between them on the timeline. The spaces allow key frames to be inserted between other key frames. In terms of indexing, the spaces are treated as 'half' indices. For example, the space at position 4.5 is the space between key frames 4 and 5.

Navigation along the timeline can be done in 'half' or 'full' movements. This means that to move back and forth between key frames one pair of buttons is used, but to select the spaces between key frames for insertion another pair is used. This not only supports ordinary timeline navigation for 'playback' purposes to view a consecutive series of key frames, but also enables the spaces between key frames to be selected when required.

## 4.4 Object Selection

In order to use head tracking for object selection, a ray is drawn from the point between the user's 'eyes' in the direction of their gaze. A check is then performed to determine if this ray intersects any objects in the scene. This is performed once per frame render. This method is inefficient for large scenes containing many objects, but for previs purposes it is satisfactory as densely populated scenes are unlikely. If the collision check passes for a certain object it is given a red highlight to indicate this to the user.

A problem arises when the ray collides with multiple objects along the gaze direction. The user may be unable to select an object that is partially occluded. In this case the object that was first hit by the ray is selected. This allows users to select any object that can be seen from their current position by directing their gaze into the viewable part/s of the object from outside the object. This allows the selection to be reset by looking away. This approach was taken as opposed to something similar to a 'depth cursor' as defined in [6]. While a depth cursor allows for more control over selection when there are multiple candidates, an occluded object may be difficult to see when it is highlighted. Another problem with the depth cursor is that due to limitations of the Unity3D engine, the cursor itself could also become occluded by objects.

A further problem occurs when an object far away from the user has to be repeatedly selected. This is because the target for selection becomes smaller due to perspective as the distance between the user and the object increases. To help mitigate this issue, a reticle is drawn in the center of the user's gaze. This allows the user to determine the exact target of his/her gaze within the VE.

# 5. EXPERIMENTS AND RESULTS

During the development process, three expert HCI heuristic evaluations were performed. Each expert was asked to use the system and explore its functionality without any particular goal in mind. The experts identified issues according to Nielsen's ten usability heuristics [13] and dictated them while using the system. Each issue was given a priority ranking of low, medium, or high. One of the issues flagged with a high priority involved the use of joysticks for any axial input. Initially in manipulation mode there were no sub-modes, instead users could perform translations using the joysticks and rotations using buttons. It was found that users had less control over fine rotations when using buttons as opposed to joysticks. Another issue that arose was regarding the selection of objects which were far away from the user. If the user wanted to make fine adjustments to an object that was far away the repeated selection task became tiring. This lead to the implementation of the reticle to assist in object selection. Another high priority issue was that of timeline control. Initially users were only able to overwrite a key frame by deleting it before inserting a new key frame. Users were also only able to capture key frames by inserting them after the selected key frame. It was found that more control was required to make timeline editing more efficient, and this lead to the 'half' index system being implemented. These evaluations formed part of a development iteration where the issues were resolved before the final qualitative evaluation.

Due to the explorative nature of this project, as well as time constraints, a qualitative approach was taken to evaluate the interface's effectiveness. This was focused on the effectiveness of the interface for the previs task. A total of three participants took part in the evaluation. No participants had any prior experience with 3D modelling, animation, or the Unity3D editor. Participants were asked to replicate an example timeline as quickly and accurately as possible using our system. While performing the task participants were encouraged to 'think out loud' while their voices were recorded using a smartphone. Once the participant had either finished the task or 40 minutes had elapsed they were then required to answer four open-ended questions, as well as a System Usability Scale (SUS) [9].

The recordings were transcribed and then analyzed together with the question responses to determine any possible themes. This was done by searching for meaningful words or phrases which were commonly used by participants when describing their experience of the system. All of the participants first spent time going through the control mappings that appear when the user looks down. While they did this they repeated the labels for each mapping out loud, and some tried each action to determine what it does. One participant commented: "It is difficult to remember all these things", referring to the control scheme. The original rationale for displaying the control scheme when the user looks down was that the user might be inclined to look down at the physical controller in hand. However, the control scheme image could be constantly displayed in one corner of the HUD with the ability to be toggled on or off by the user. Displaying the labels for each action clearly while making the image smaller on the display could be challenging.

The system's reaction to the 'reset position' action was unexpected by one participant, who exclaimed and said "I thought I just deleted everything!", when the camera began to fade to black. While fading the user's vision to black is a common technique used to prevent simulator sickness, the participant's reaction could be attributed to their inexperience with VR. Two of the participants initially expressed feelings of confusion with regards to their task of replicating a timeline. While the task was explained beforehand, participants were unsure if the characters in the scene needed to be moved before capturing the first key frame. Characters should be placed in an initial position that clearly

shows that manipulation is required in order to replicate the first key frame.

One of the aspects where participants expressed the most difficulty was camera/viewpoint movement. Two participants said they found it "difficult" to rotate the camera view by only using head movement. These participants said that they expected the right joystick to rotate or pan the camera. One participant said: "I kept moving the joystick in a circular motion expecting the camera to rotate and pan in that direction but it didn't." Another participant cited his experience from video games as the reason for this expectation. This relates to the Scene-in-Hand metaphor described by Ware and Osborne [21] where the user's hand movements translate to scene movements. In the case of our system the user's hand movements on the joysticks translate to scene movement in navigation mode. By using head movement as the only means to change the user's forward direction the user's control in terms of the Scene-in-Hand metaphor is reduced. This is because users are unable to change the forward direction using hand movements.

Participants were also unsure whether to move the camera or the character when capturing close up shots of characters. This may be partially attributed to the previous issue as well as the lack of background objects in the sample timeline. Background objects can be used as a reference to aid in determining whether or not an object needs to be manipulated.

All participants had trouble understanding the purpose of the 'ghost' objects that appear when the currently selected key frame is edited. This concept had to be explained to each participant during the task. This may have been caused by an issue with the sample models used that were unable to display transparency. Instead, a white opaque version of the model was displayed as the ghost. This also caused issues when participants attempted to make small adjustments to objects because of the occlusion created by the ghost.

Participants' feelings towards object manipulation were mixed. Two participants answered that they found manipulating objects easy. One participant attributed this to previous experience using the Xbox360 controller. All participants answered that they found translations an easy aspect of the system. This supports previous research suggesting that placing the user in a scene using VR allows for a better understanding of 3D spatial relationships between objects [2, 20]. Participants also mentioned difficulty specifically with rotating objects: one participant described some rotations as being unexpected given the movements being made on the joysticks, another participant mentioned that the joystick was too sensitive when performing rotations. The unexpected rotations could be attributed to certain models having different origins around which rotations are calculated. This should be normalized when a model is imported into the scene. Another approach to mitigating this problem is constraining rotations similarly to how translations are constrained. This would prevent objects from being rotated through the floor of the scene.

No participants reported any issues with object selection. This could be expected given that a ray-based technique was used in a sparsely populated scene [4]. One participant answered that "selecting characters was easy as there was a crosshair on the body of the characters". This confirms the possibility that displaying a reticle to show the user's gaze target may aid in object selection.

While our sample size is not big enough to produce significant results from a quantitative evaluation tool, the SUS was administered as an extra measure of evaluation. The SUS uses a series of Likert scales that give scores from 0-4. These scores are then combined to produce a final SUS score from 0-100. This final score is not interpreted as a percentage, instead it represents a "composite measure of the overall usability of the system being studied" [9]. The three scores from our evaluation were 47.5, 62.5, and 70. It has been shown that a score higher than 68 is considered above average, and conversely, below average if below 68 [9]. According to this, our system scored below average overall. However, it is difficult to interpret these scores without normalizing them.

# 6. CONCLUSION

This paper has presented a VR, HMD-based system for previsualization. The system uses commercial, readily available hardware to enable people to create previsualizations. Users are placed in a virtual scene to help them better understand the 3D spatial relationships between objects while navigating around the scene. Our system allows users to create a timeline of key frames by capturing renderings of their current perspective, effectively making the viewpoint and camera synonymous. Users are able to insert and delete key frames as well as edit key frames by manipulating objects in a separate mode. In a small-scale qualitative evaluation it was found that participants had multiple issues with our interface, mostly regarding the control scheme, camera control, and 3D rotation. Participants reported positively on the 3D translation and object selection aspects of our system. Overall it is clear that previs poses new challenges for a VR application and that the immersive nature of VR shows potential for a novel, narrative way of creating previsualizations by placing the user in the scene. Users are able to explore the scene from the perspectives of the characters in it. However, alternative interaction techniques must be considered in addressing the issues shown in our evaluation.

One possible avenue for future work would be to explore alternative input devices. While the game controller has some advantages, users are required to memorize a control scheme which adds a learning element to the system. Previous work indicates that a hybrid approach using a combination of 2D touch input and 3D gesture-based input holds promise. It would also be useful to undertake a more extensive quantitative study in order to better ascertain the potential of this system. The features of our system could also be extended to explore high-fidelity previs. This includes planning and rendering camera and object movements, as well as the ability to do basic animations.

# 7. ACKNOWLEDGMENTS

Our thanks to Bryan Davies, and Hendranus Vermeulen for their expertise and analysis. Our thanks to our supervisor, James Gain, without your guidance and motivation this project would not have been possible. Finally, thanks to our participants for participating in our evaluation.

# 8. REFERENCES

[1] Bowman, D.A. et al. 1999. Testbed Evaluation of Virtual Environment Interaction Techniques. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (London, UK, 1999), 26–33.

[2] Butterworth, J. et al. 1992. 3DM: A Three Dimensional Modeler Using a Head-mounted Display. *Proceedings of the 1992 Symposium on Interactive 3D Graphics* (San Diego, CA, USA, 1992), 135–138.

[3] Chittaro, L. and Burigat, S. 2004. 3D Location-pointing As a Navigation Aid in Virtual Environments. *Proceedings of the Working Conference on Advanced Visual Interfaces* (Gallipoli, Italy, 2004), 267–274.

[4] Dang, N.-T. 2007. A Survey and Classification of 3D Pointing Techniques. *2007 IEEE International Conference on Research, Innovation and Vision for the Future* (Hanoi, Vietnam, 2007), 71–80.

[5] Gallo, L. et al. 2008. Toward a Natural Interface to Virtual Medical Imaging Environments. *Proceedings of the Working Conference on Advanced Visual Interfaces* (Napoli, Italy, 2008), 429–432.

[6] Grossman, T. and Balakrishnan, R. 2006. The Design and Evaluation of Selection Techniques for 3D Volumetric Displays. *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology* (Montreux, Switzerland, 2006), 3–12.

[7] Hughes, C.E. et al. 2013. CaveCAD: Architectural design in the CAVE. *2013 IEEE Symposium on 3D User Interfaces (3DUI)* (Orlando, FL, USA, 2013), 193–194.

[8] Jankowski, J. 2011. A Taskonomy of 3D Web Use. *Proceedings of the 16th International Conference on 3D Web Technology* (Paris, France, 2011), 93–100.

[9] Jordan, P.W. SUS: a 'quick and dirty' usability scale. in Jordan, P.W. Thomas B. ed. *Usability Evaluation In Industry*, CRC Press, 1996, 189-194.

[10] Laundry, B. et al. 2010. Interaction with 3D Models on Large Displays Using 3D Input Techniques. *Proceedings of the 11th International Conference of the NZ Chapter of the ACM Special Interest Group on Human-Computer Interaction* (Auckland, New Zealand, 2010), 49–56.

[11] Lubos, P. et al. 2014. Analysis of direct selection in head-mounted display environments. *2014 IEEE Symposium on 3D User Interfaces (3DUI)* (Minneapolis, MN, USA, 2014), 11–18.

[12] Mine, M. et al. 2014. Making VR Work: Building a Real-world Immersive Modeling Application in the Virtual World. *Proceedings of the 2Nd ACM Symposium on Spatial User Interaction* (Honolulu, HI, USA, 2014), 80–89.

[13] Nielsen, J. and Mack, R.L. *Usability Inspection Methods*. John Wiley & Sons, New York, NY, USA, 1994.

[14] Nitsche, M. 2008. Experiments in the Use of Game Technology for Pre-visualization. *Proceedings of the 2008 Conference on Future Play: Research, Play, Share* (Toronto, Canada, 2008), 160–165.

[15] Oculus Best Practices. Retrieved November 8, 2015, from Developer Center, Oculus VR, LLC: https://developer.oculus.com/documentation/intro-vr/latest/concepts/book-bp/.

[16] Ponto, K. et al. 2013. SculptUp: A rapid, immersive 3D modeling environment. *2013 IEEE Symposium on 3D User Interfaces (3DUI)* (Orlando, FL, USA, 2013), 199–200.

[17] Stuerzlinger, W. and Teather, R.J. 2014. Considerations for Targets in 3D Pointing Experiments. *Proceedings of HCI Korea* (South Korea, 2014), 162–168.

[18] Teather, R.J. and Stuerzlinger, W. 2014. Visual Aids in 3D Point Selection Experiments. *Proceedings of the 2Nd ACM Symposium on Spatial User Interaction* (Honolulu, HI, USA, 2014), 127–136.

[19] Tollmar, K. et al. 2004. Navigating in Virtual Environments Using a Vision-based Interface. *Proceedings of the Third Nordic Conference on Human-computer Interaction* (Tampere, Finland, 2004), 113–120.

[20] Wang, J. and Lindeman, R. 2014. Coordinated 3D Interaction in Tablet- and HMD-based Hybrid Virtual Environments. *Proceedings of the 2Nd ACM Symposium on Spatial User Interaction* (Honolulu, HI, USA, 2014), 70–79.

[21] Ware, C. and Osborne, S. 1990. Exploration and Virtual Camera Control in Virtual Three Dimensional Environments. *Proceedings of the 1990 Symposium on Interactive 3D Graphics* (New York, NY, USA, 1990), 175–183.

[22] Wikiversity: The main page of FrameForge 3D Studio, 2015. Retrieved November 9, 2015, from Wikiversity, Lesson Page: Introduction to FlameForge: https://upload.wikimedia.org/wikiversity/en/thumb/b/b4/FrameForge_SBTDS_Screen_Shot.png/360px-FrameForge_SBTDS_Screen_Shot.png

[23] Zhang, Y. et al. 2005. The Use of Visual and Auditory Feedback for Assembly Task Performance in a Virtual Environment. *Proceedings of the 21st Spring Conference on Computer Graphics* (Budmerice, Slovakia, 2005), 59–66.