StoryTIME: Development of a Tangible Interface for Storytelling

by

Michael Gharbharan

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Science

in

Computer Science

The Faculty of Science

University of Ontario Institute of Technology

August 2018

Contents

List of Figures

List of Tables

Abstract

The production of 3D digital animated sequences is an iterative process where ideas are scrutinized by members of the production team until they are finalized. Production teams are comprised of many individuals with varying levels of expertise in a broad spectrum of disciplines such as screen writing, animating and directing. It is imperative that everyone involved understands the development tasks for the production. To facilitate the communication of ideas, the prototyping process typically begins with the development of a storyboard that is then turned into a 3D previsualization. This process requires specialized training to be effective. This thesis presents StoryTIME, a system designed to simplify the storyboarding and digitalization process by leveraging tangible interfaces and projected augmented reality. The presented system works by capturing and recording the motion of physical props such as small toys and figurines which are then applied to a virtual counterpart. We discuss the design and development of the system followed by a user study that investigates its effectiveness with respect to traditional animation prototyping systems.

Chapter 1

Introduction

Creating computer generated animations is an extensive process requiring expertise in a broad spectrum of disciplines. The process begins with the inception of an idea and ends with a playable clip bringing the idea to life [1, 9]. The disciplines needed to achieve this range from non-technical roles such as screen-writing to very technical roles such as rigging and animating. It is crucial that individuals involved with the production are able to quickly and accurately express their ideas among each other. The premise of this work is to examine traditional techniques used to prototype ideas in the early production stage of computer animation projects and propose a system which aims to promote collaboration, reduce iteration time and simplify communication. The proposed system is intended for use by members of the animation's production team and will try to make tasks commonly reserved for animators accessible to everyone.

1.1 Production Pipeline for 3D Animation

The production pipeline for 3D animation is similar to the pipeline of traditional film production [9]. A visual diagram of a typical production pipeline can be seen in Figure 1.1. While this figure shows the steps a typical production will follow, it is important

to remember these steps are more of a guideline and can vary between productions. Figure 1.1 shows the pipeline we will focus on for the context of this thesis. The pipeline starts in pre-production where the planning, designing and research of the 3D production takes place [1]. This thesis focuses on the pre-production stage. In this stage, production team members need to convey ideas among each other. To help facilitate this, visual representations of the script are produced. The first visual depiction is usually in the form of a storyboard (see Figure 1.1). A storyboard is a 2D creation of the script which resembles a comic book and consists of shot-by-shot sketches of the plot. Depending on the nature of the production, the next step is to develop an animatic or a 3D pre-visualization or sometimes both. An animatic is essentially an animated version of the storyboard with primitive audio cues and animations. A pre-visualization by definition is similar to an animatic with the key difference being that it is in 3D. Pre-visualizations are especially useful when making 3D productions as they allow developers (animators, screenwriters, directors etc.) to map out the movement and staging of objects and characters in a scene.

The production and post-production stages of the pipeline are where the actual production ready assets are developed with maximum fidelity (discussed in Chapter 2.1). While these stages are crucial in production, they are not the focus of this thesis and will not be discussed in further depth. This thesis is focused on the development of pre-visualizations for 3D animated sequences.

1.1.1 Pre-Visualizations

Development of an animated clip is an iterative process where ideas are criticized and rethought until finalized. Creating high fidelity clips for each iteration can be very time consuming and costly. Instead it is common to create a pre-visualization (pre-vis) which is a low fidelity prototype that virtually recreates the layout of the set. The pre-visualizations allow developers to identify problems with the staging

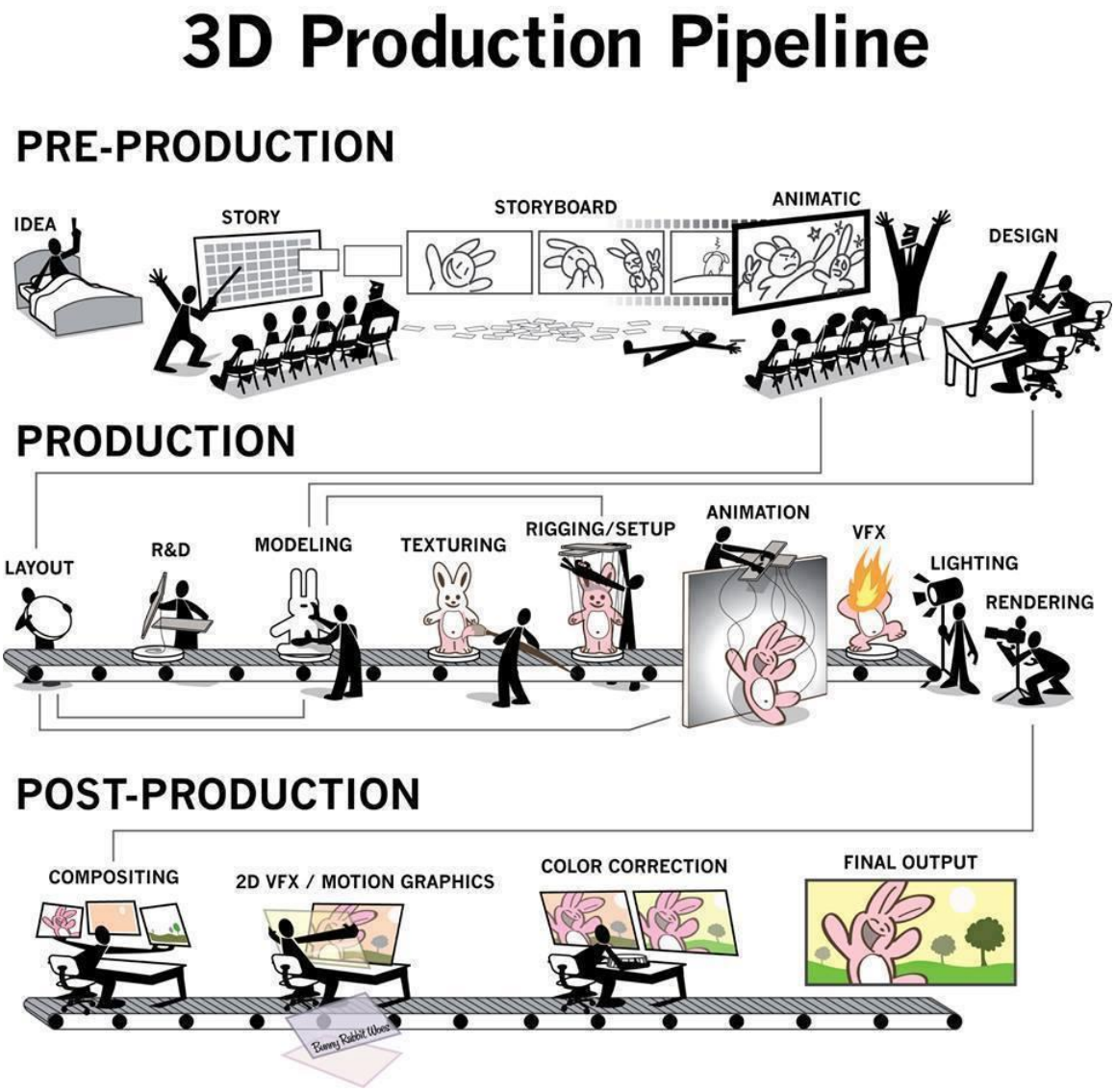


Figure 1.1: A graphical breakdown of the 3D animation pipeline from idea to final product [1]

and motion within the scene prior to the creation of high fidelity assets [2, 1]. It is important to establish staging and actions as early as possible to prevent issues which may arise further in development, such as spatial limitations which hinder an actor's capability to perform desired actions within the set.

An example of a pre-vis can seen in Figure 1.2. In this example, the production team was tasked with laying out a complicated scene in the movie "Rise of the Planet of the Apes"[10]. As this scene relies heavily on visual effects (VFX), which can

take considerable amounts of time and money to produce, it is imperative that they work out the details of the sequence prior to performing final renders. Since this scene consists of shots filmed in the real world, in addition to virtual augmentations, the pre-vis also includes depictions for the placement of cameras and other filming equipment as seen in the "Witness_Cam shot" in Figure 1.2. In this shot, a virtual camera crane is displayed to determine the logistics of filming in the physical area to minimize the potential for conflicts such as insufficient space to place the various camera rigs[1]. When creating a purely virtual production the development of the pre-vis can become simplified as spatial limitations of the real world do not need to be necessarily considered.

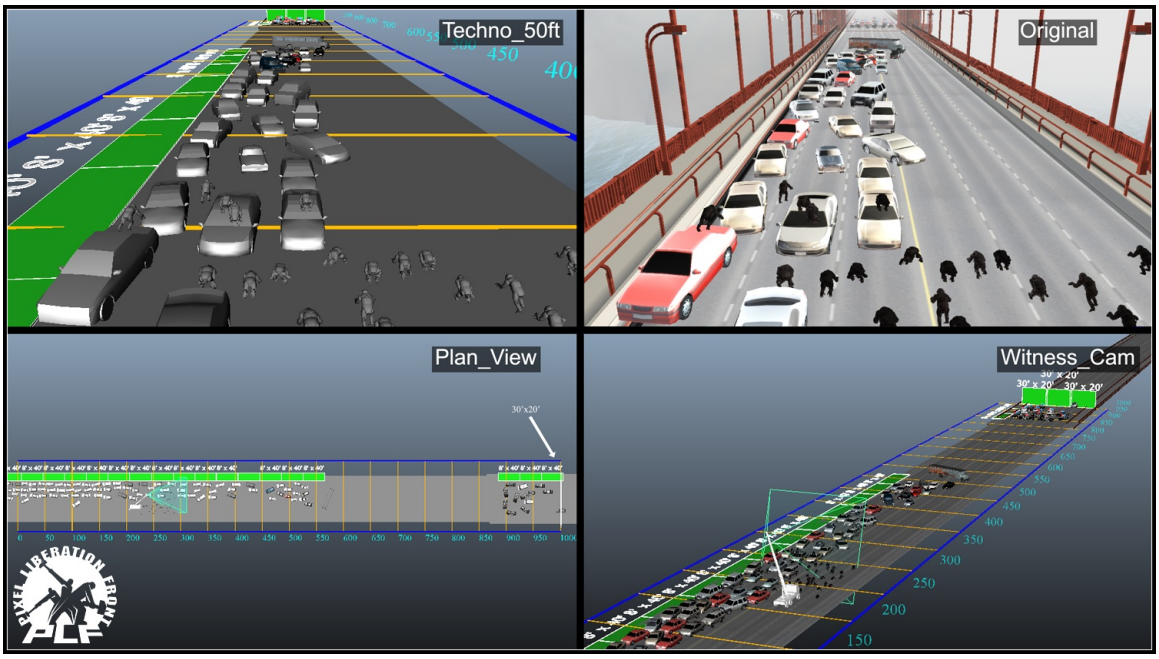


Figure 1.2: An example of a pre-vis used in the film "Rise of the Planet of the Apes". Top left: Technical render from the camera point of view visualizing scene depth. Top right: Render of a frame in the pre-vis. Bottom left: Top-down view of the scene. Bottom right: Virtual representation of what the physical set will look like.[2]

[1]Camera rigs include dolly tracks and cranes.

Issues Creating Pre-Visualizations

Translating an idea into a playable virtual clip is non-trivial due to the nature of the technical software used to develop them. Large productions will often have entire departments dedicated to this task. Methods to create pre-visualizations will be discussed in Chapter 2, however the typical approach is to key frame the locomotion of objects using a keyboard and mouse interface. While this is effective, it relies on the animator's interpretation of the vision as described by the director. Miscommunication and misinterpretation can result in a flawed prototype of the scene which in turn can lead to more development iterations. Reduction in the number of iterations and the time it takes to produce an iteration is beneficial as it can accelerate development time. The StoryTIME system which will be presented in this thesis was created to improve iteration time in the development of pre-visualizations in collaborative environments.

1.2 Tangible Interactive Media Environment

The concept of the Tangible Interactive Media Environment (TIME) was presented in 2015 by Buckstein [11] with the development of PlayTIME. The idea was to bridge the gap between physical and digital prototyping techniques. Physical techniques include developing paper prototypes such as sketches in a notebook. PlayTIME specifically investigated the scenario of a game designer placing objects such as power-ups and enemy spawn points within a virtual environment. Traditionally these tasks are performed using a keyboard and mouse interface. PlayTIME replaced the keyboard and mouse with a set of Augmented Reality (AR) markers, each of which performed a specific task. For example, if the designer wanted to spawn a "enemy spider" to attack the player, they would place the AR marker representing the enemy spider in

the *interaction area*[2] and press a button to create an instance of the object in the virtual world. The interaction area is an area on a physical table which is mapped to locations in the virtual world.

The underlying principle of TIME is that all aspects of game development can be done with some form of a tangible interface. The end goal is to have a suite of systems which make development tasks more accessible by reducing the reliance on technical software, as shown in Figure 1.3. This can be achieved by enforcing the use of common language (i.e. minimal reliance of technical terms) and creating an interface where users do not need to undergo extensive training to use the software but can instead learn by interacting with the system naturally. Conceptually this can be applied to any discipline but the current focus is on digital media.

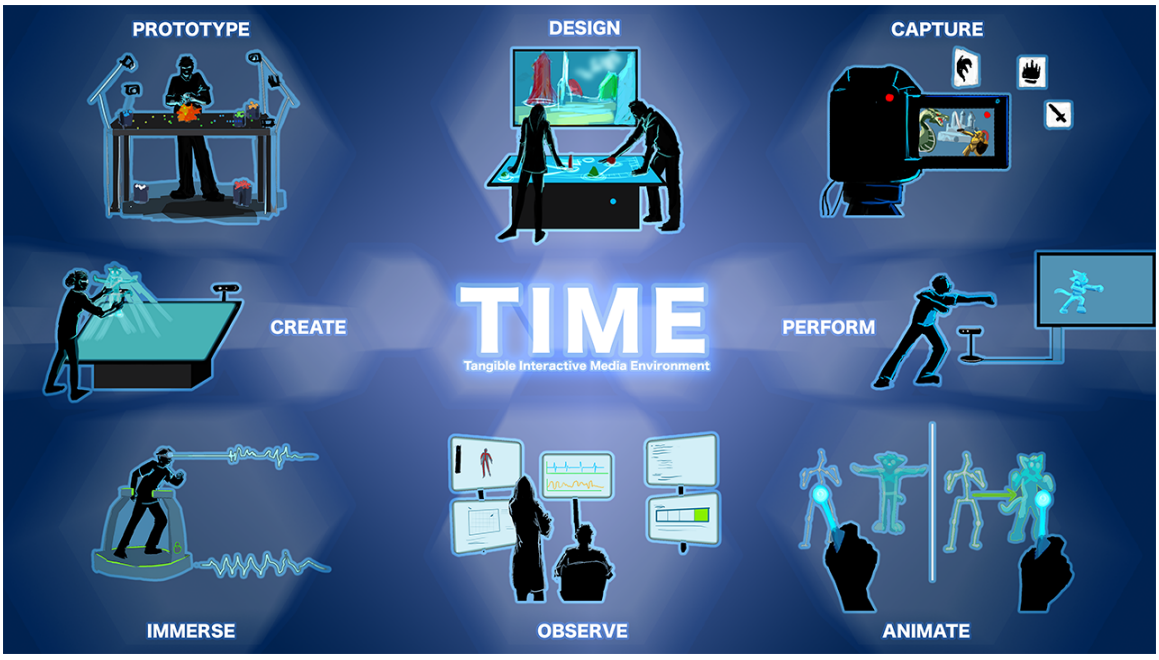


Figure 1.3: The concept of the TIME suite of development stations. Each station would provide a tangible interface for specific development tasks.

[2]Interaction area refers to a region in the real world where the system is able detect and digitize physical objects.

1.2.1 StoryTIME

StoryTIME is a system within the TIME suite intended for story creation. The system was created to address the previously discussed issues with iteration time, miscommunication and collaborative prototype development. This is achieved by leveraging advances in Augmented Reality and Tangible User Interfaces to create a novel interface for prototyping the staging and locomotion of objects in a virtual scene. The core idea is to capture the motion of physical objects and apply them onto a corresponding virtual object. Physical objects are in the form of small figurines which are tracked with AR markers. The physical objects and their surrounding area are augmented using spatial augmented reality. The system is throughly described in Chapter 3.

The rest of this chapter introduces the essential technologies that compose the StoryTIME system.

1.3 Tangible User Interfaces

The entire TIME ideology revolves around tangible user interfaces. Traditionally, the interaction with a computer is done with a keyboard and mouse and an interface built with windows, icons, menus and pointers (WIMP). WIMP interfaces have existed since the inception of graphical user interfaces [12]. A tangible user interface (TUI) is one that allows for the control or manipulation of a virtual object through the use of a physical handle [13, 14]. In the case of StoryTIME, instead of using a mouse to position a virtual object, users grasp onto a physical prop such as a small toy or figurine and the software will interpret the spatial manipulations on the prop and apply changes to the virtual objects. Related applications and work in the area of tangible user interfaces will be presented in Chapter 2.

Visibility of system status is one of the 10 usability heuristics for user interfaces outlined by Nielsen [15]. This heuristic is concerned with keeping the user informed

with what is currently happening within the system. Tangible user interfaces commonly leverage *augmented reality* as a means to display system status.

1.4 Augmented Reality

Augmented reality (AR) overlays digital information in a real-world environment [16]. There are three main technologies utilized to display the digital augmentations: LCD/LED displays (i.e computer monitors, TVs, mobile phones etc.), head mounted displays (HMD) and projectors [17].

PokemonGO! [3] is a popular game which makes use of AR. A screenshot of the game can be seen in Figure 1.4. In this screenshot, a digital character is overlaid on top of an image of the real world captured by the camera on a mobile device. This display medium is very common making it a lucrative option for many AR developers. The limitation to AR in this form is that in order to see the virtual augmentations you must look at the display device.

Figure 1.4: Screenshot from PokemonGo!, a game that uses augmented reality. In this image a virtual character is overlaid on top of an image of the real world. [3]

Mixed Reality is a form of AR which aims to circumvent this limitation by displaying the virtual augmentations directly in the real world [18]. These augmentations can be displayed using head mounted displays such as the Microsoft Hololens [4] or the Google Glass [19]. Figure 1.5 depicts a user wearing a Hololens and interacting with virtual augmentations. Since the display is mounted to the user it is impossible for the augmentations to leave their field of view. Some limitations of this method of displaying virtual augmentations is that it encumbers the user which could lead to fatigue and depending on the design of the system could require each user to have a HMD device.

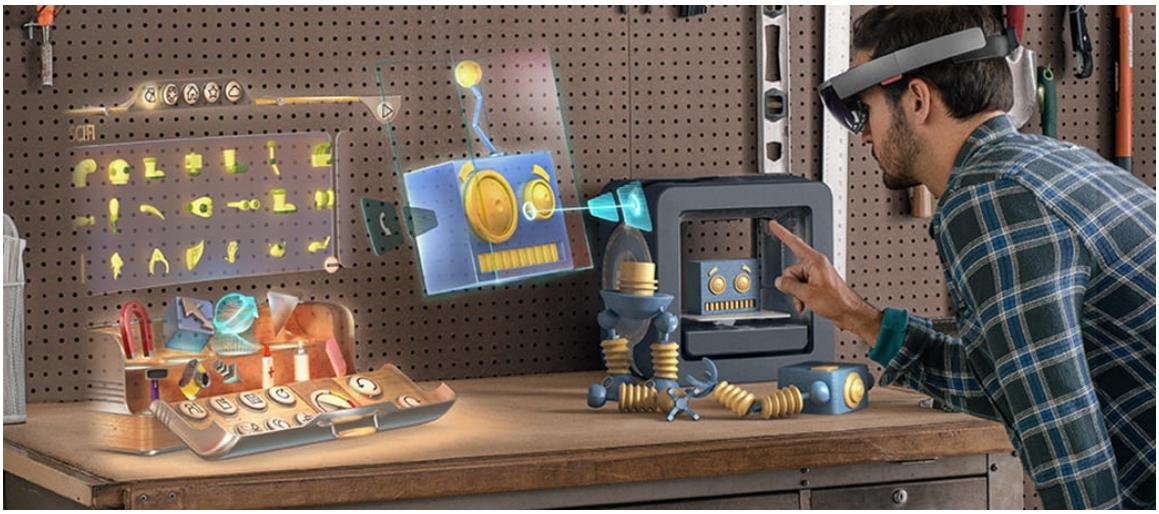


Figure 1.5: Visualization of the augmentations displayed using a Hololens. The user is wearing the head mounted display and is interacting with virtual augmentations. [4]

Projectors are another way to display virtual augmentations in the real world. This is sometimes referred to as Spatial Augmented Reality (SAR). ShaderLamps [20] demonstrates using projectors to augment the lighting conditions on physical models. Figure 1.6 illustrates this, on the left a model of the Taj Mahal is presented on the right we see the same model with textures and artificial lighting projected onto it.

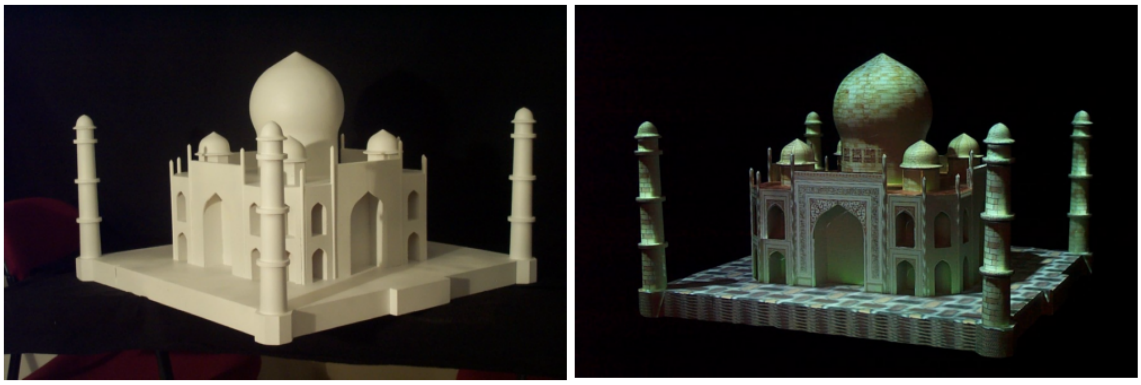


Figure 1.6: Demonstration of Spatial Augmented Reality the model on the left is augmented using projectors to visualize texture and variable lighting conditions. [5]

Each of these display technologies can be used to bring a tangible user interface to life. StoryTIME utilizes spatial augmented reality by projecting the digital augmentations onto and around the physical props users interact with. The primary justification for this design decision is that it allows users to see the augmentations without being limited by the viewing angles of a LCD/LED display and it also allows them to remain unencumbered as they do not have to wear any HMDs. Projection based AR is limited by factors that comprise the visibility of the augmentations such as ambient light, obstructions and projector coverage. Ambient light refers to light present in the environment that does not originate from the projector. If there a lot of ambient light then the images displayed by the projector will appear washed out and faded, this can happen in a bright room or outside on a sunny day. To overcome this projectors are typically used in darker environments. Obstructions between the projector lens and display surface cast shadows that prevent the augmentations from being visible. A single projector can only display a single image plane, this means it is only capable of covering an object from a single view point, this is the projector coverage issue. Shadows and projector coverage limitations can be mitigated by using multiple projectors as demonstrated in RoomAlive [21]. Related work which leverage spatial augmented reality and tangible user interfaces will be presented in Chapter 2.

1.5 Research Contributions

The contributions of this work include the design, development and evaluation of StoryTIME, a tangible interface for the creation of pre-visualizations. Evaluation was done in the form of a user study. This work aims to answer the following questions:

- Does prototyping with tangibles improve iteration time when developing pre-visualizations?
- How does prototyping pre-visualizations with tangibles compare with traditional keyboard and mouse based tools in terms of usability and user experience?
- How does displaying augmentations on a computer monitor compare with projected augmentation in terms of user preference and performance?

1.6 Thesis Structure

Chapter 2 will discuss related work in the area of pre-visualization development.

Chapter 3 will provide a detailed breakdown of the StoryTIME system. This chapter will cover implementation details and justifications for the design decisions made during development.

Chapter 4 presents the evaluation of StoryTIME. In this chapter the evaluation methodology is described and the results are presented and discussed.

Chapter 5 concludes the thesis with a summary and discussion of the study results and future work.

Chapter 2

Related Work

The purpose of this chapter is to establish a background in prototyping techniques used during the production of *animated sequences*. An animated sequence in the context of this work refers to a series of related events showing the progression of a scene from start to finish. The chapter begins with an overview of exiting prototyping techniques and is followed by a series of underlying technologies that support AR and tangible user interfaces (TUI). The goal of this chapter is to provide rationale into the design decisions that went into creating StoryTIME.

2.1 Prototyping Animated Sequences

The definition of a prototype varies between industries, the general idea is that they are a "work in progress" or "proof of concept". McElroy defines a prototype as a manifestation of an idea into a format that communicates the idea to others or is tested with users, with the intention to improve that idea over time [22]. This is the definition that will be used in the context of this thesis.

Prototype Fidelity

Determining the level of *fidelity* for a prototype is an essential decision that needs to be made early in development. Fidelity refers to how feature complete the prototype

is [22, 23]. There are three common levels of fidelity used when developing prototypes, low, medium and high.

Low fidelity prototypes are the first step in the development of a product as they are generally quick to develop and have low development costs. A paper prototype is an example of a low fidelity prototype. Paper prototypes are non-functional mock-ups of the end product. If prototyping a graphical user interface (GUI) one could draw the UI elements on a pieces of paper and quickly iterate by simply re-drawing elements based on design decisions. In the case of computer animation, paper prototypes are commonly in the form of a storyboard which presents the rough idea of how a scene will play out. While paper prototypes are not functional, they allow quick iterations at a low cost.

Medium fidelity prototypes often build upon their low fidelity counterparts. This is where functionality starts being added in, in the case of a GUI, this can be where a digital mock-up with primitive functionality could be produced for the intention of early testing. Early testing allows developers to determine if an idea is technically feasible or if any major revisions need to be made. Medium fidelity prototypes typically have longer development times than low fidelity prototypes as they require some level of functional implementation.

High fidelity prototypes are a level above medium fidelity prototypes, these prototypes should have all major issues ironed out and the system should be usable. In the case of animation these prototypes should be ready for final renders.

Storyboards & Pre-Visualizations

Development of an animated sequence often begins with a *storyboard* [24, 25]. Development of a storyboard occurs after the core details of the plot have been developed. Storyboarding allows creators to workout the visual elements that best suit the story [24]. Visual elements refer to the objects in the scene and the twelve principles of animation. The principles of animation were created by Disney as a means to provide

guidelines for animators to follow when creating animations [1]. While each of the twelve principles are important, developing a single TUI to accommodate all of them would be a difficult challenge. However, creating a series of TUIs designed for specific tasks, as discussed in the TIME framework presented in Chapter 1 should be a more attainable goal. StoryTIME focuses on the *staging*, *timing* principles of animation. These principles are described below:

- **Staging:** how the objects in the scene are laid out and how they fill the screen.
- **Timing:** how long an action takes to complete.

Storyboards are a form of prototyping and as such have varying levels of fidelity. Figure 2.1 (top) shows an early storyboard. This board is made up of sticky notes and rough sketches. Early storyboards in this from are referred to as *thumbnails* [26, 24, 25, 6]. Figure 2.1 (bottom) shows the same scene but with more detailed drawings. Comparing these two figures, we can see that the staging and framing are nearly identical. The big difference between the two revisions is the additional detail added in the storyboard. These additional details give a better sense of the mood and environment. The reasoning for using low fidelity prototypes is to reduce the number of high fidelity iterations needed as developing higher fidelity renditions take longer to develop. Furthermore, staging issues can be resolved without the need for high fidelity drawings.

Pre-visualizations (pre-vis), as described in Chapter 1.1.1 are higher fidelity prototypes in the form of a playable clip that depicts the essential elements shown in the storyboard. The traditional approach to creating a pre-vis is by creating a key-framed animation. A key-framed animation is one where the animator poses an object at *key* moments and relies on interpolation to calculate the intermediate frames. There are many software suits available with key-framing functionality such as Autodesk Maya, Blender, Unity and Unreal Engine.

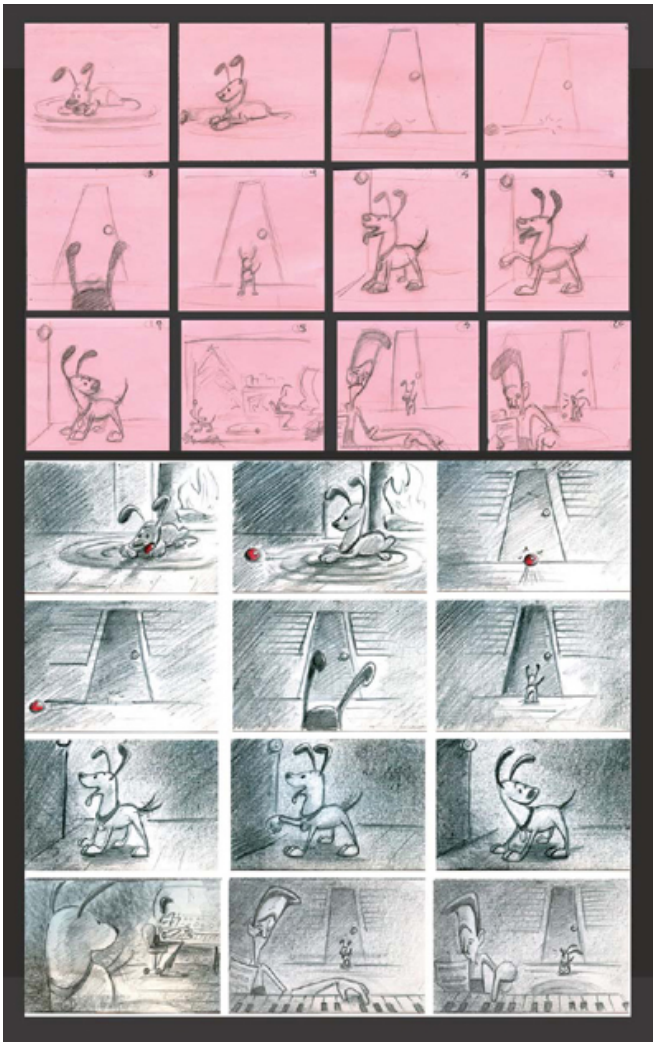
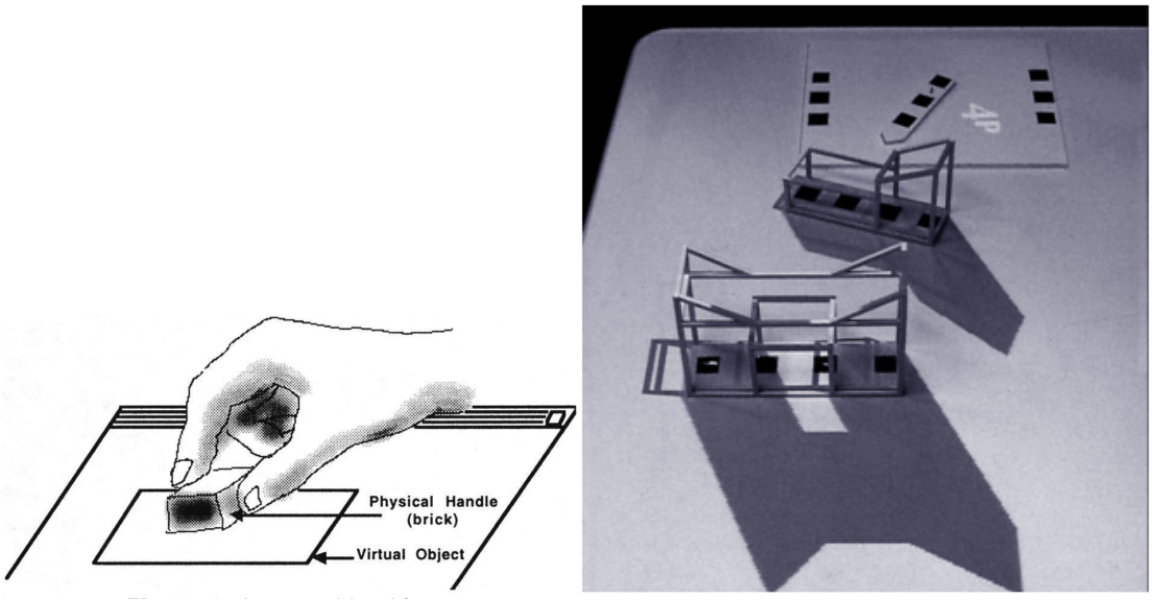


Figure 2.1: Example of thumbnail board (top) and a revised storyboard (bottom).[6]

2.1.1 Tangible User Interfaces for Prototyping

The first form of a Tangible User Interface (TUI) was presented in the *Bricks* system shown in Figure 2.2a by Fitzmaurice [13]. The idea was to replace the mouse with a set of blocks to interact with a UI. Imagine a slider on a touch screen, but instead of using your finger to interact with it, you placed a block on it and slid the block. This introduced the idea of having multiple blocks to interact with multiple UI elements simultaneously, something that a cursor based interaction method would not be able to achieve. Tangible Bits [14] generalized the Bricks concept by disconnecting it

from the WIMP[1] style interface by allowing the physical object itself be the UI. Urp, shown in Figure 2.2b is an urban planning tool where each object was individually recognizable by the system and allowed users to prototype the positioning of buildings and wind flow by manipulating the physical objects.



(a) The Bricks user interface [13] (b) Urp, a TUI for urban planning [14]

Figure 2.2: The pioneering works in the field of Tangible User Interfaces

Enabling users to interact with a system using natural movements is one of the biggest affordances of a TUI. A study on the impact of TUIs on designer's spatial cognition was conducted by Kim and Maher [27]. They investigated the effects of epistemic and pragmatic actions on a user's ability to complete designing tasks. Epistemic refers to the use of exploratory motor actions to incrementally move towards an end goal. Pragmatic actions rely on mentally forming the end goal and performing minimal motor movements to get there. TUIs generally favor epistemic actions while traditional graphical user interfaces (GUI) favor pragmatic actions. The study asked participants to design the layout of office environments using a TUI and a GUI. Their intention was to explore the change in the participant's spatial cognition between the

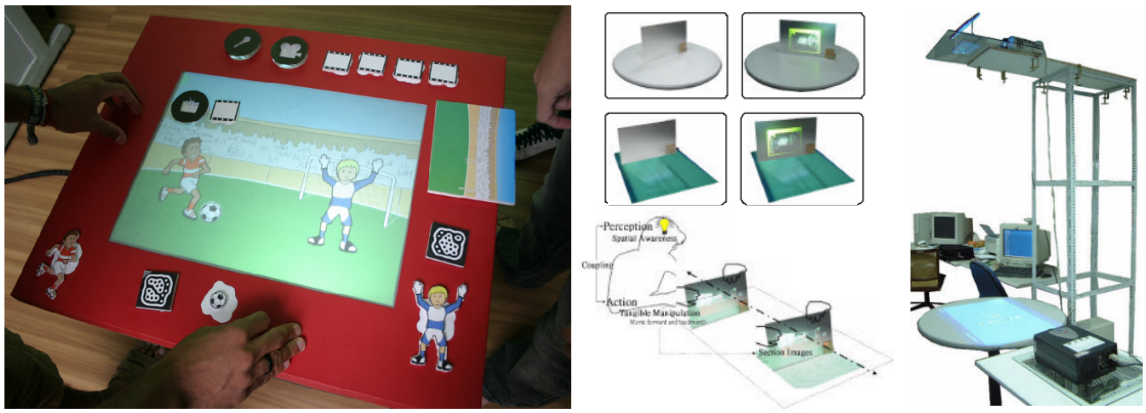
[1] windows, icons, menus and pointers

two types of interface. They found that when using the TUI participants began the session by randomly placing the physical props in the interaction area and then began the process of deciding where the items should be positioned. They also found that with the TUI participants repositioned objects more frequently than with the GUI. This finding suggests that the claim of exploratory development is more prevalent with a TUI.

Physlights [28] is a TUI for positioning lights in a virtual scene. They conducted a user study comparing their Physlights TUI with the established Maya GUI. Participants were given a scene and were asked to recreate the lighting conditions using the two interfaces. They found that participants completed the task significantly faster using the TUI.

Reactoon [29] shares many of the same intrinsic philosophies as StoryTIME. They identify that storytelling is an essential activity in day to day life, and that there is a barrier to creating virtual recreations of their story. Reactoon specifically investigates creating an interface for children ages 5 to 9 years to create animations in 2D for educational purposes. The system can be seen in Figure 2.3a, it uses a table top design with a camera beneath the display surface. Placing the camera behind the display surface eliminates any potential issues introduced as a result of marker occlusion. The props are designed as two sided "disks", on one side is an image of the tool, as seen in the top row of Figure 2.3a and on the other side is the AR marker, as seen at the bottom of Figure 2.3a. When the prop is placed on the surface marker side down, the camera is able to detect and identify the prop and the actions being performed. The drawback to this approach is that the detection is limited to 2D tracking.

iNAVIGATOR [30] is a TUI designed for the purpose of visualizing 3D environments. The objective of this system was to provide an intuitive alternative to visualizing 3D objects. This is a different application area from the previous work discussed in the prototyping area but it is important to acknowledge other innovative

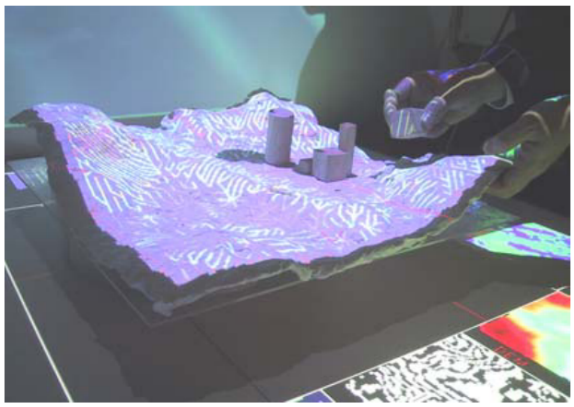


(a) The Reactoon TUI [29] (b) The iNAVIGATOR system [30]

Figure 2.3: TUI examples

TUIs. The iNAVIGATOR system, illustrated in Figure 2.3b, allows users to observe cross sections of 3D objects. The system has two display surfaces, a table and an orthogonal display plane which can be slid across the table. The system has two projectors, one for each display surface. As the user slides the display surface across the table, the projector will update the projected image as if the the physical display surface was taking a cut into the virtual object.

Clay has also been shown to be an effective TUI [31, 32]. The Illuminating Clay system shown in Figure 2.4a demonstrates visualizing changes in elevation by projecting augmentations directly onto a mock-up of the terrain.



(a) The Illuminating Clay TUI [31]

2.2 Augmented Reality Technology

The definition of augmented reality (AR) is to alter and enhance a user's visual field with additional context sensitive information pertaining to the task they are performing by overlaying virtual content in a real world environment. This is the definition used by Caudell [7] when he coined the term in 1992, while investigating the challenge of aircraft manufacturing at Boeing [33]. Aircrafts are a very complicated assortment of millions of parts, many of which need to be assembled by hand due to the level of dexterity needed. During the manufacturing process, workers must refer to a manual which provides them with step by step assembly instructions. Manuals commonly come in the form of printed documentation. It is often a challenge for workers to align the documented diagrams with their workspace. This can lead to longer assembly times and even improper construction. To remedy this situation, Caudell presented a head mounted display to overlay virtual assembly instructions directly in the user's field of view. The intention of this system was to improve the efficiency and quality of the worker's performance. A diagram of this setup can be seen in Figure 2.4.

While the case of a HMD based AR solution for manufacturing may not seem very related to prototyping animated sequences, the underlying technology is inherently the same. Head mounted displays are just one form of displaying virtual augmentations and will be discussed further among other display technologies in section 2.2.2. The key challenges with implementing an AR solution can be broken down into three categories: object detection, object tracking and augmentation visualization.

2.2.1 Object Identification and Tracking

Object identification and tracking are two essential pillars to the foundation of any AR application [34, 35, 36]. Recall that the goal of AR is to annotate physical objects by overlaying them with virtual imagery. Before this can be done, the application

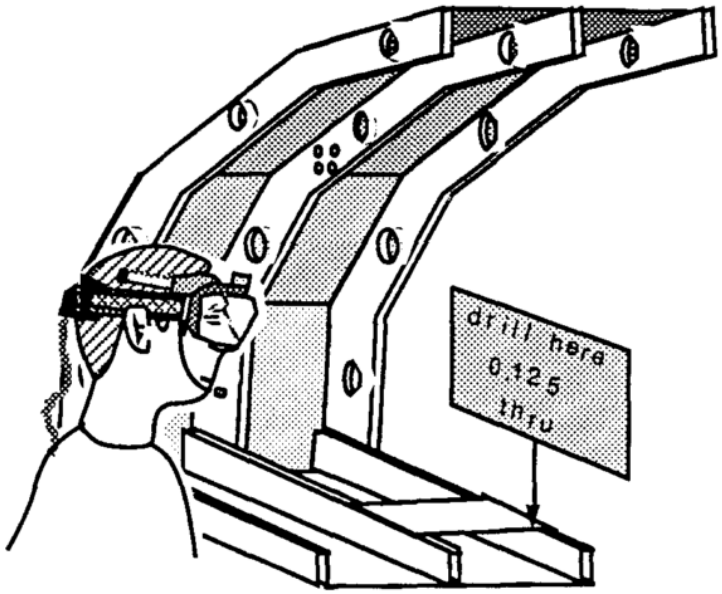


Figure 2.4: Depiction of a HMD used to provide a worker with assembly instructions. The worker is on the left, the "goggles" in-front of their face is a display. The "drill here" instruction on the right is an example of a virtual instructional overlay. This overlay is visible to the user wearing the HMD. [7]

must determine where the object is with respect to the viewer, this is the object tracking problem. There are many approaches to solving this problem including but not limited to optical, mechanical [37] and electromagnetic [38] solutions.

Optical Tracking

Optical tracking refers to methods which rely on an imaging device such as a digital camera to obtain data for the tracking algorithm [39]. This is also referred to as vision based tracking. Vision based tracking is very common as the only hardware required is a digital camera. The underlying idea behind vision based tracking algorithms is to align sets of points. For example, Figure 2.5e shows an *AR marker*, Figure 2.5a shows the AR marker as it is observed by a camera. The objective of the tracking algorithm is to map points from the reference marker (Figure 2.5e) to points on the observed marker (2.5d). These points of interest are referred to as *features*. Once these point correspondences are established, the *pose estimation* process can begin. Pose

estimation refers to the problem of calculating a transformation that can transform points between their respective coordinate systems. The process is summarized in Figure 2.6. AR markers such as the one in Figure 2.5e are referred to as *fiducial*. These markers consist of patterns which are easily detectable, in the case of Figure 2.5e, the marker is surrounded by a thick border. The detection algorithm is then programmed to explicitly search for squares in the input image, as shown in Figure 2.5d.

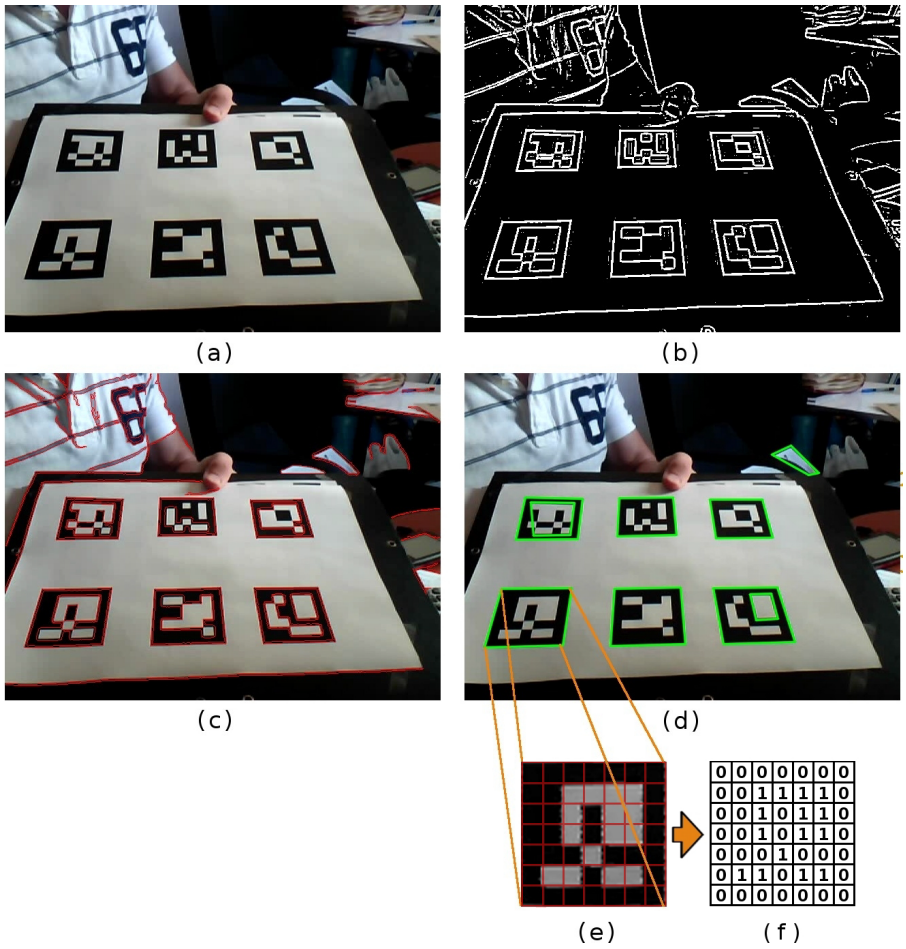


Figure 2.5: Summary of the tracking process for fiducial markers, (a) Input from camera, (b) Result from binarization (c) Result of contour detection, (d) Result of square fitting, (e) Example of reference marker, (f) Binary representation of marker [8].

A drawback to fiducial markers is that they typically have a distinct and unnatural look to them. The alternate approach to fiducial markers are *natural markers*.

Algorithms using natural markers work similarly to fiducial based solutions in the sense that they still look for features, however these features are not in the form of explicitly defined markers.

Figure 2.6: The steps to image registration. The first step is to detect features in the image. The second step is to match those features to known features on the object. The third step is to solve the spatial transformation that maps the detected features to the known features.

2.2.2 Displaying Augmentations

Once an AR application knows where to display an augmentation, the question becomes how to display the augmentation. As mentioned in the introduction, LCD/LED displays (i.e computer monitors, TVs, mobile phones etc.), head mounted displays (HMD) and projectors are the three major technologies utilized in displaying virtual augmentations. The rest of this section discusses each of them in more depth.

Traditional Displays

LED/LCD displays are perhaps the most popular display medium utilized by AR applications to date. These displays are everywhere and with rising popularity of high-end AR capable smart phones, this trend shows no sign of slowing down. Example of these displays include TVs, mobile phones and computer monitors. Sport broadcasts often use AR in this form to illustrate play breakdowns.

Head Mounted Displays

Head mounted displays (HMD) are displays which are affixed to the user's head [37, 7]. Commercial examples of HMDs include the Hololens by Microsoft [4], Google Glass [19] and the Oculus Rift [40]. It is important to realize the distinction between AR

and virtual reality (VR). VR applications are ones which completely substitute the user's world with a virtual one. With AR applications the user remains within their environment. The Oculus Rift is commonly used in VR applications but can be used for AR applications when used in conjunction with a camera to provide imagery of the working environment.

Ivan Sutherland invented the first head mounted display (HMD) in 1966 [37]. The concept of the HMD has not changed much since its inception. The concept is to place a see-through display in front of the user's eyes. The design is to allow users to see their environment but with the ability to show additional information. The case of a HMD for manufacturing was discussed earlier in the chapter. HMDs can be also used as a means for immersive telecommunication. Video chat applications such as Skype [41] allow users to see a live video stream of the remote parties they are communicating with. Holoportation [42] uses a Hololens as the display medium and leverages AR and 3D reconstruction to virtually insert the remote parties directly in the physical environment of the wearer. This concept is referred to as "physical co-presence". A user study was conducted to evaluate the Holoportation system. The findings illustrate several improvements to traditional video calls for collaborative tasks. One specific example examined was the perception of natural movements such as pointing, leaning and gazing. Participants reported that when the remote party was superimposed in the physical environment, they were able to clearly identify what the remote party was looking and pointing at. These cues are often lost with traditional video calls. Overall the Holoportation system seems like a promising technology for remote collaboration.

Projector Based Displays

Spatial augmented reality (SAR) refers to a branch of AR applications which utilize projectors to display virtual augmentations directly on their physical counterparts [43, 20, 44]. Illumiroom [44] used projection mapping to augment the visual ap-

pearance of the area surrounding a TV with the intention of increasing immersion. Roomalive [21] expanded on the work of Illumiroom by using a series of projector-camera pairs to augment an entire room. They demonstrate the ability to project *spatially aware* interactive applications such as games at the room scale. Spatially aware refers to content that is able to react to the environment they are in, for example, they demonstrate a physics simulation of rain drops falling onto the surfaces in the room with accurate interactions with respect to the angle of the surfaces. Projection mapping can also be used to bring static objects to life, as shown in Display objects [43] where projected augmentations are used to demonstrate an interactive GUI on objects constructed from foam. SideBySide [45] demonstrates a multi-projector system where users of the system hold small projectors in their hands and point at a surface they want to display information onto.

Chapter 3

StoryTIME

Chapter 2 discussed related literature and traditional techniques for prototyping animated sequences. The prototyping process typically begins with a paper prototyping technique such as storyboarding to establish a snapshot for the staging and layout of a scene. The next step is to create a 3D animated mock-up of the scene known as a pre-visualization. On an abstract level this can be thought of as a transformation between the real and virtual worlds. This process has been shown to be effective in countless animated productions but an argument can be made that it is more laborious and reliant on technical ability than necessary. StoryTIME was created to address this to lower barriers to story prototyping.

StoryTIME is a **T**angible **I**nteractive **M**edia **Environment** for **story**telling. The goal of the system is to allow for rapid iteration during the development of animated sequences by leveraging augmented reality (AR) and tangible user interfaces (TUI). StoryTIME is a system where users of all disciplines can create prototypes in a playful environment, eliminating the need to learn technical software. A playful interface is one that encourages users to engage in social and physical interaction [46]. The idea is to digitize the motion of physical objects and apply them onto their digital counterparts. Imagine a system where users could create prototypes of their animations by

simply "playing" with toys. This would make prototype creation accessible to users of all ages and skill sets. In order to do this, we must have a way for the computer to determine which toy is being manipulated (object identification), where the toy is located in the physical world (object localization) and be able to record the toy's motion (object tracking). This is the goal of StoryTIME.

StoryTIME can be summarized as a system which allows users to create computer generated animated sequences by interacting with physical props such as small toys and figurines. In order to do this the system must have the following technical capabilities at a minimum.

- Low latency position and orientation tracking for several objects at a time
- Recording the position and orientation of tracked objects
- Playing back recordings
- Providing the user with visual feedback in real time

3.1 System Requirements

The technical generalized requirements listed above were refined into the system specific features listed below:

- **Object tracking and identification:** The ability to track and record multiple physical objects at the same time is required as scenes are often comprised of multiple objects. Objects are tracked with 6 degrees of freedom. One of the key design decisions was to enable users to capture the motion of any object in the scene at any time. In order to do this we must be able to identify objects and determine their *pose* in the real world. Pose refers to the position and orientation of an object. The current implementation of StoryTIME supports tracking of up to eight physical objects at a time. The system can be scaled

to allow for the tracking of more objects, however, with the current hardware setup (detailed in Appendix A) performance begins to degrade below the 30 frame per second (FPS) target when eight objects are tracked simultaneously.

- **Virtual reconstruction of physical world:** A fundamental goal of a tangible prototyping system is to bridge the gap between the physical and virtual worlds [14]. With StoryTIME interactions are performed using physical props, these interactions are processed and a corresponding virtual reconstruction is produced and presented to the user in real-time.

- **Augmentation of the physical world:** Virtual augmentation is used to bring the physical props to life by altering their visual appearance and giving them functional capabilities as described in Chapter 3.17. Projection mapping was chosen as the display medium used to provide spatial feedback to the users of the system by displaying system status directly in the working environment.

- **Serialization:** Saving the recorded animations and virtual world reconstructions to a persistent storage device (i.e. a hard drive) is essential in allowing creators to have a redistributable asset containing their work.

3.2 System Setup

The current implementation of StoryTIME does not use any platform specific code and has been tested on a HP Sprout. The Sprout (Figure 3.15) was chosen because it had all of the required equipment conveniently integrated as a single unit and was available for use in the GAMERLab at UOIT. An added benefit to having everything integrated as a single unit is not having to worry about recalibration in the event of a sensor being bumped. Having said that, StoryTIME is not restricted to the Sprout platform, the system only relies on a depth sensing camera and a projector. The only thing that must be considered is the operational range of the camera and

projector. For example, early prototypes of StoryTIME used a Microsoft Kinect V2, but these setups were unsuitable because the Kinect's operational range of 60cm to 4.5m was designed for room scale applications and StoryTIME was designed for table top applications and needed depth data in the 30cm to 80cm range, which is the operational range of the F200 camera on the HP Sprout. Full specifications for the HP Sprout can be found in Appendix A.

3.3 System Architecture

As discussed in Chapter 2, there are many enabling technologies that can be used to achieve the requirements and capabilities listed above. One of the challenges developing a tangible system is determining which technologies to use. The underlying software architecture driving StoryTIME was developed with a modular design in mind to allow for experimentation with different technologies. Each of the essential components of the system can be replaced without having influence on other components. The essential components of the system are summarized in Table 3.1. StoryTIME is currently implemented as a C++ application, however the functionality can be exported as a DLL. This allows integration in arbitrary frameworks and game engines such as Unity3D or Unreal Engine. StoryTIME uses fiducial AR markers for tracking and projection as the display medium. A high level overview of the system architecture can be seen in Figure 3.1.

StoryTIME has two primary threads, the *main thread* and the *image acquisition thread*. The main thread is responsible for managing the state of the application, performing the object tracking and rendering tasks. The image acquisition only has one task which is to continuously pull data from the cameras. The flow diagram for the image acquisition thread can be found in Figure 3.5. The general steps in the StoryTIME pipeline are summarized in the following list:

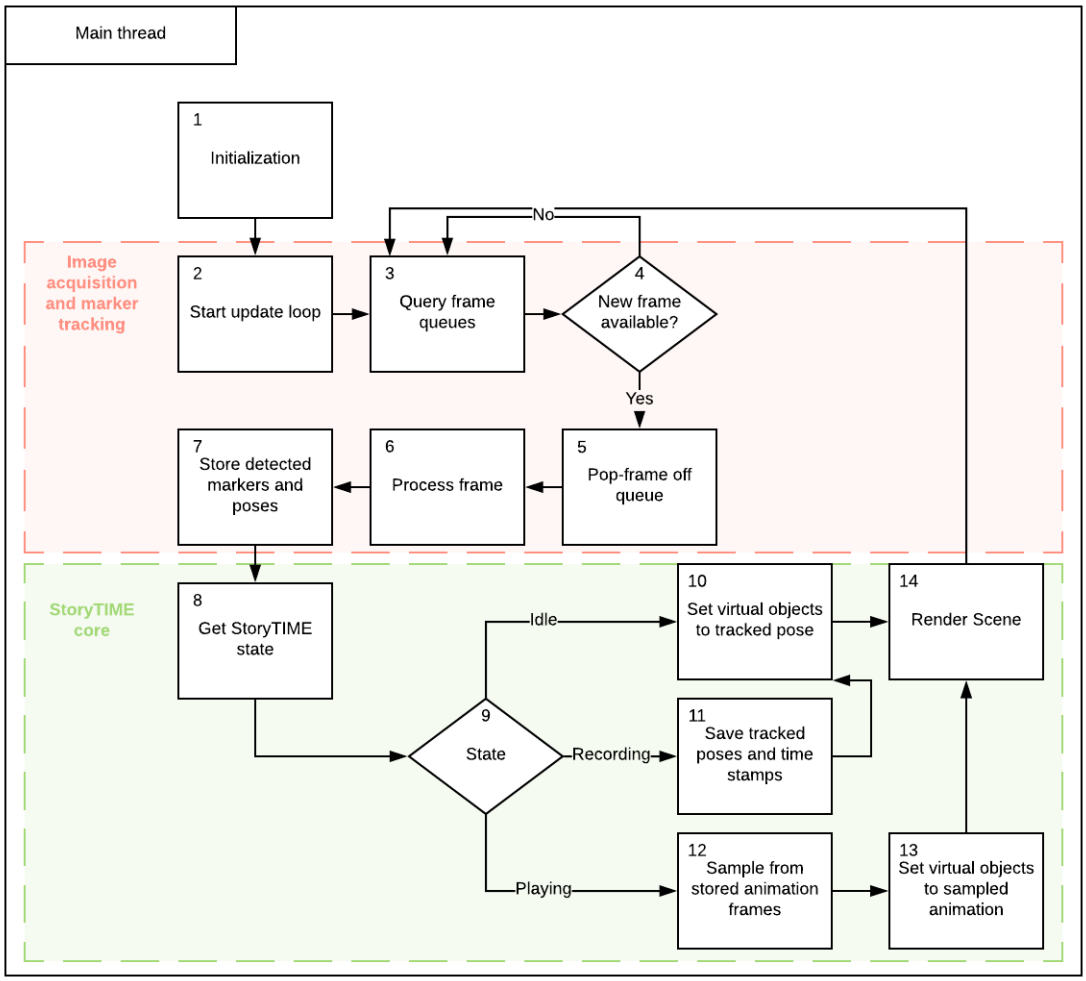


Figure 3.1: High level diagram of the data flow and system architecture of StoryTIME.

1. **System initialization:** This occurs once and only once when the system is first run and is where resources are allocated and configured. This is discussed in Section 3.3.1.

2. **Image acquisition:** This happens every frame (every 33 milliseconds), this is where images are obtained from the camera. This is discussed in Section 3.3.2.

3. **Image processing:** After an image is acquired, it needs to be processed. This is where the object identification and tracking happens, the input image is

processed to detect AR markers and determine the object's 3D pose. This is discussed in Section 3.3.3.

4. **StoryTIME Core:** Once all of the markers have been identified and tracked, the StoryTIME core uses them as input. Depending on the state of the system, it will either record animations, play animations back or remain idle.

Table 3.1: StoryTIME core interfaces

Interface Name	**Description**
CameraSensorBase	Interacts directly with camera SDK to obtain data from camera, discussed in Section 3.3.2
ObjectTrackerBase	Interacts directly with the object tracking library to obtain marker identifiers and poses, discussed in Section 3.3.3
ObjectTrackerConfigBase	Configuration data for the object tracker, discussed in Section 3.3.3
TrackedObjectBase	Contains per-object pose and identification data, discussed in Section 3.3.3
AppBase	Runs application loop and responds to messages from operating system (OS)
StoryTIME	The main StoryTIME class that manages the state of the system.
SingletonBase	Easy way to make any class a singleton type

The remainder of this section will discuss each of the processes in the system architecture flow chart (Figure 3.1) in depth.

3.3.1 Initialization

Initialization of system resources if the first step which happens immediately when the StoryTIME executable is run (box 1 in Figure 3.1). A detailed visualization of the initialization process can be seen in Figure 3.2. The initialization phase is responsible for starting the camera, object tracker, renderer and internal StoryTIME systems. These are the core components of the system as listed in Table 3.1, if any of these components fail to start, the application is terminated.

The first component to initialize is the camera. Recall that the user's input to StoryTIME is in the form of spatial manipulations to physical objects, these manipulations are observed by the camera. If the camera fails to initialize, the application immediately exits as StoryTIME will be unable to receive the inputs. Further details regarding the image acquisition process are discussed in Subsection 3.3.2.

The second step in the initialization process is to setup the object tracker (box 4 in Figure 3.2). StoryTIME relies on the Chilitags [47] library that process input images to detect, identify and extract the 3D pose of AR markers. Several steps are performed in this stage, they are summarized in the following list:

- **Configure AR markers:** inform the object tracker which markers to look for and provide a description of the physical dimensions (width and height).
- **Specify camera calibration properties:** in order to calculate 3D pose, the tracker must be informed of the projection and lens distortion parameters which map 3D points to 2D pixels. This calibration is provided by the RealSense API.
- **Configure the Kalman filter:** used to smooth noisy tracking results [47]

The third step is to initialize the StoryTIME core which refers to the various managers (Table 3.1) that are responsible for managing the state of the system. These managers are responsible for maintaining the animation takes and the mapping of physical to virtual objects.

The fourth and final step is to setup the renderer. This process involves loading and preparing assets such as shaders, 3D models and textures for rendering.

3.3.2 Image Acquisition

Image acquisition refers to the process of obtaining data from the cameras. StoryTIME has an interface named CameraSensorBase which is responsible for encapsulating the image data and functions needed to pull data from a specific camera, a UML diagram

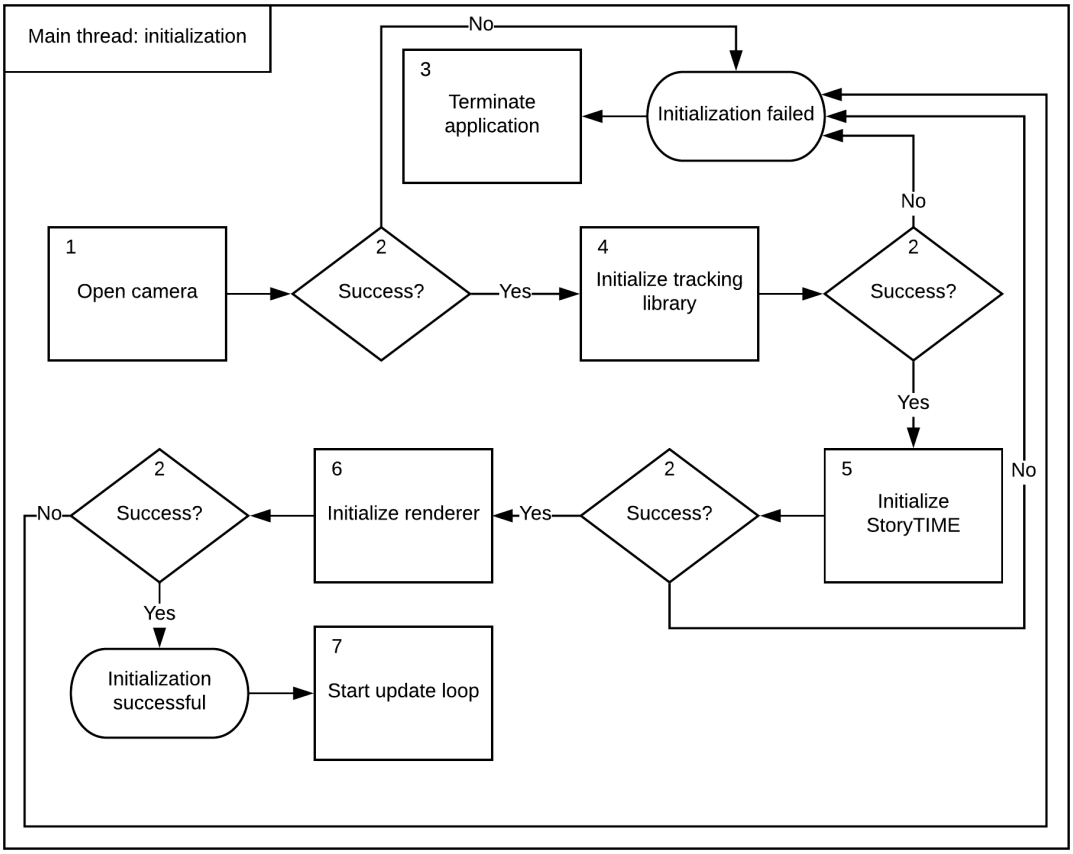


Figure 3.2: Initialization process, this is a detailed view of box 1 from Figure 3.1.

of this class is shown in Figure 3.3. This abstract class does not do any work directly but rather instead provides a common interface for all cameras to follow. The use of a camera specific API is often required for interacting with the camera's hardware. Figure 3.3 shows the three camera APIs implemented during the development of StoryTIME.

StoryTIME uses an Intel RealSense F200 camera. To obtain data from this camera, the RealSense SDK by Intel must be used. There are two ways of pulling data from the camera, *polling* and *queuing* [48].

Polling works by first checking if a new frame is available, if a frame is available, it gets returned and is fed through the processing pipeline (Figure 3.1, box 6). The problem with this method of image acquisition is the potential for frames to be missed,

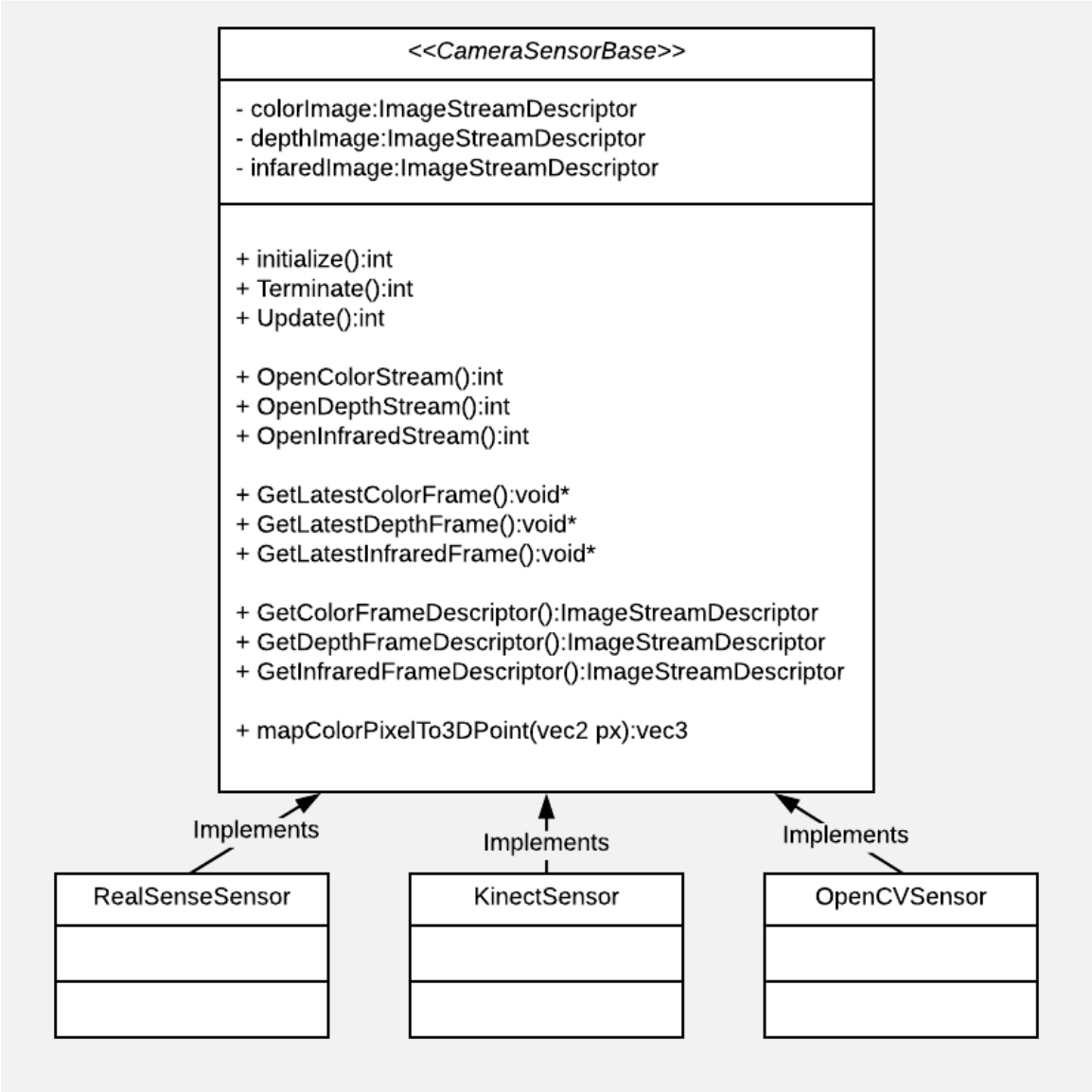


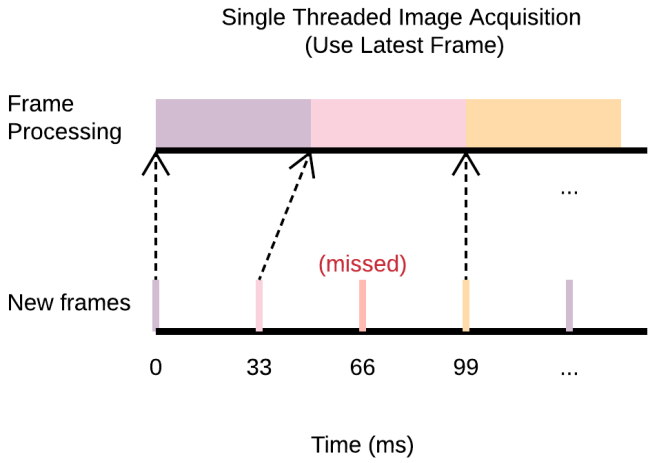
Figure 3.3: CameraSensorBase is the interface that abstracts the image acquisiton process for various camera APIs. In this diagram we see three camera APIs that were used during the development of StoryTIME. Each of these APIs implement the CameraSensorBase interface.

this is illustrated in Figure 3.4a. In this diagram we see that the frame at time 66ms is not processed because the application is always grabbing the most recent frame. In many applications this may not be an issue, but since StoryTIME is performing motion capture it is crucial that every frame gets processed to ensure the motion of the objects is faithfully captured.

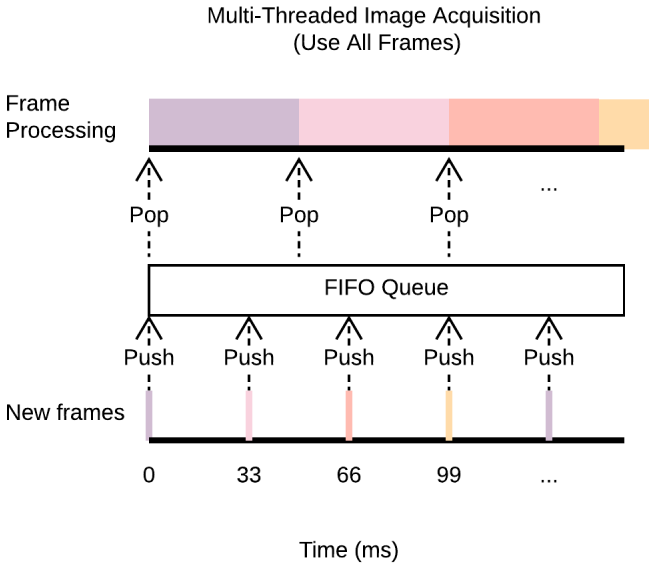
The queuing approach to image acquisition works by creating a thread dedicated to pulling data from the camera, a diagram of the data flow can be seen in Figure 3.5. When a new frame is received, it gets pushed onto a *frame queue*, this is illustrated in Figure 3.4b. A queue is a first in first out (FIFO) data structure which means data is evicted from the queue in the order it was entered. The frame queue stores all incoming image frames, when the application finishes processing a given frame, instead of grabbing the most recent frame, it grabs the next frame in line from the frame queue. This introduces some potential latency, but ensures that every frame gets processed which in turn results in more accurate motion capture. StoryTIME makes use of the queuing approach to image acquisition.

3.3.3 Object Tracking

After an image has been obtained, the next step is to process the image. This involves detecting markers in the image and obtaining their 3D pose. The current implementation of StoryTIME is capable of tracking the position and orientation of up to eight objects in real time. This limit is imposed to maintain stable tracking performance and can be increased by using a more powerful computer. It is crucial for StoryTIME to operate in *real time* to maintain the feeling of responsiveness and to perform reliably. A real time system is one that can respond to a user's inputs with imperceptible latency. The target refresh rate was 33 milliseconds (ms), or 30 frames per-second (fps). This means the time between a user's input and system's response should happen within 33 ms. The lower the refresh rate, the lower the latency. The



(a) Single threaded image acquisition. Note that only the most recent frame is processed, as a result frames can be missed and not be processed.



(b) Multi threaded image acquisition. A dedicated thread is responsible for reading all data from the camera and storing frames in a queue.

Figure 3.4: Overview of the polling and queuing image acquisition methodologies.

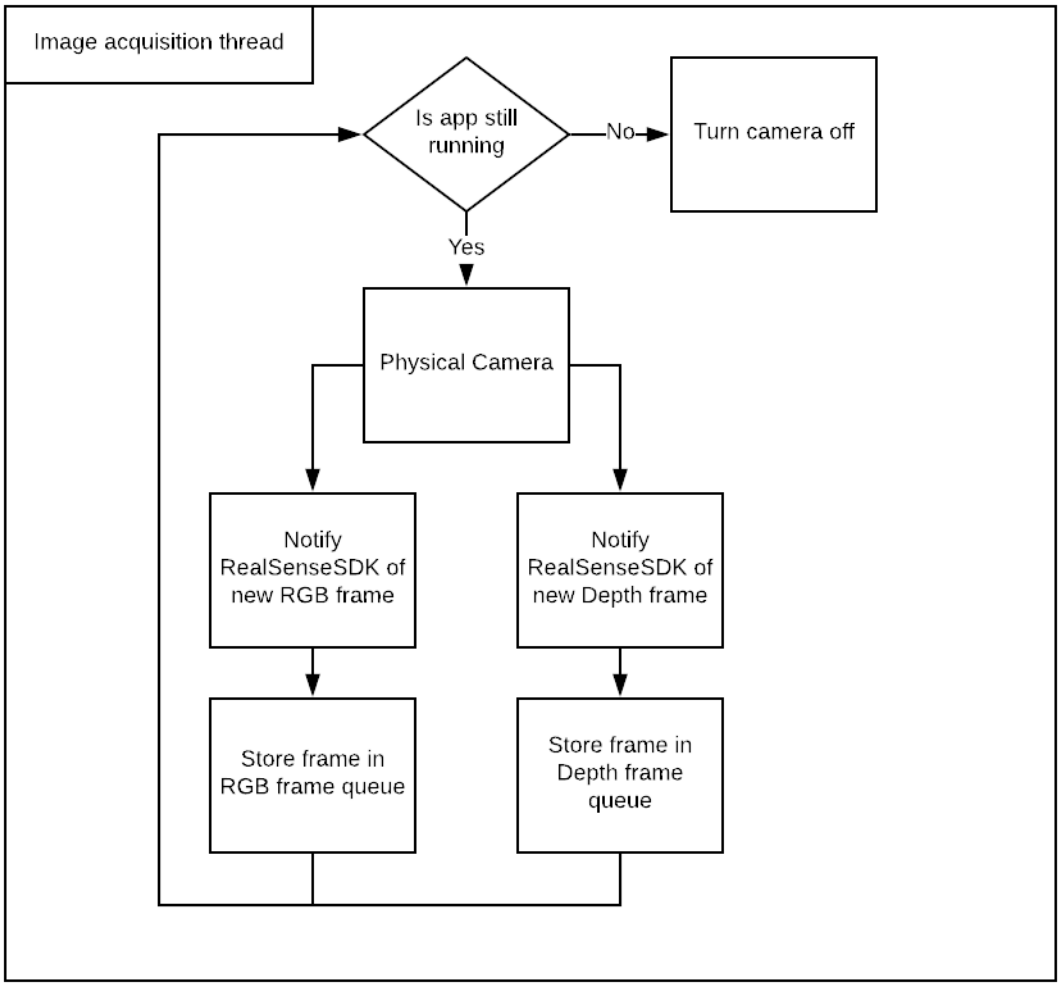


Figure 3.5: High level diagram of the image acquisition process.

F200 camera used provides images at a rate of 30 (fps). Ideally all of the work in the system should happen in the 33 ms between acquiring frames (Boxes 5 to 14 in Figure 3.1). This is the best case scenario as it means we are able to process inputs as fast as we are receiving them. Inputs to the system are in the form of spatial manipulations to physical objects that are observed by a camera. The system must be able to analyze these camera frames, detect the object manipulations and provide visual feedback to the user within a 33 ms time frame to maintain the target of 30 frames per second. When working with motion capture it is desired to have the fastest frame rate possible. More measurements enable more accurate and smoother recordings.

One major challenge when developing a tangible user interface is adequately balancing tracking robustness and responsiveness of the system. Tracking robustness refers to the accuracy of the tracking results with the presence of errors in the input data such as sensor noise and lens artifacts such as bloom. System responsiveness refers to the speed in which input is processed and turned into output. It is important to balance these to maintain system usability. We could spend a lot of processing time refining the results of the object tracker to obtain a really good pose estimation however the additional processing would increase the time takes it takes to give feedback to the user.

There are a plethora of object tracking libraries available such as ARToolKit [49], Vuforia [50] and reacTIVision [51], each of which with their own pros and cons. It was apparent that experimentation would be required to determine which tracking library would be most suitable for StoryTIME. As discussed, the entire StoryTIME system revolves around the ability to track physical objects, we needed a way to encapsulate the tracking functionality so that changing the tracking technology did not require changes to the image acquisition or animation recording systems. Similarly to the abstract CameraSensorBase class created for image acquisition (Section 3.3.2), the ObjectTrackerBase and TrackedObjectBase classes were created to abstract the

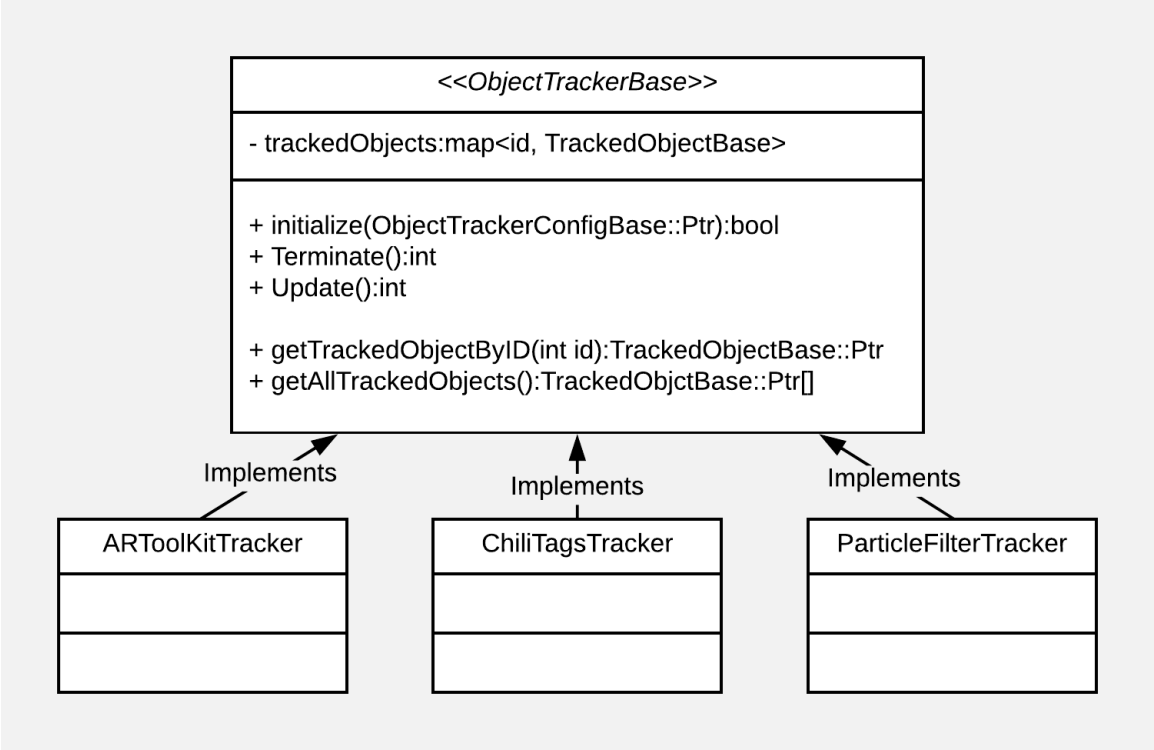


Figure 3.6: UML diagram for the ObjectTrackerBase interface that abstracts the object tracking process for various object tracking technologies. All object trackers ultimately return two things: a unique identifier for the object and the objects 3D pose.

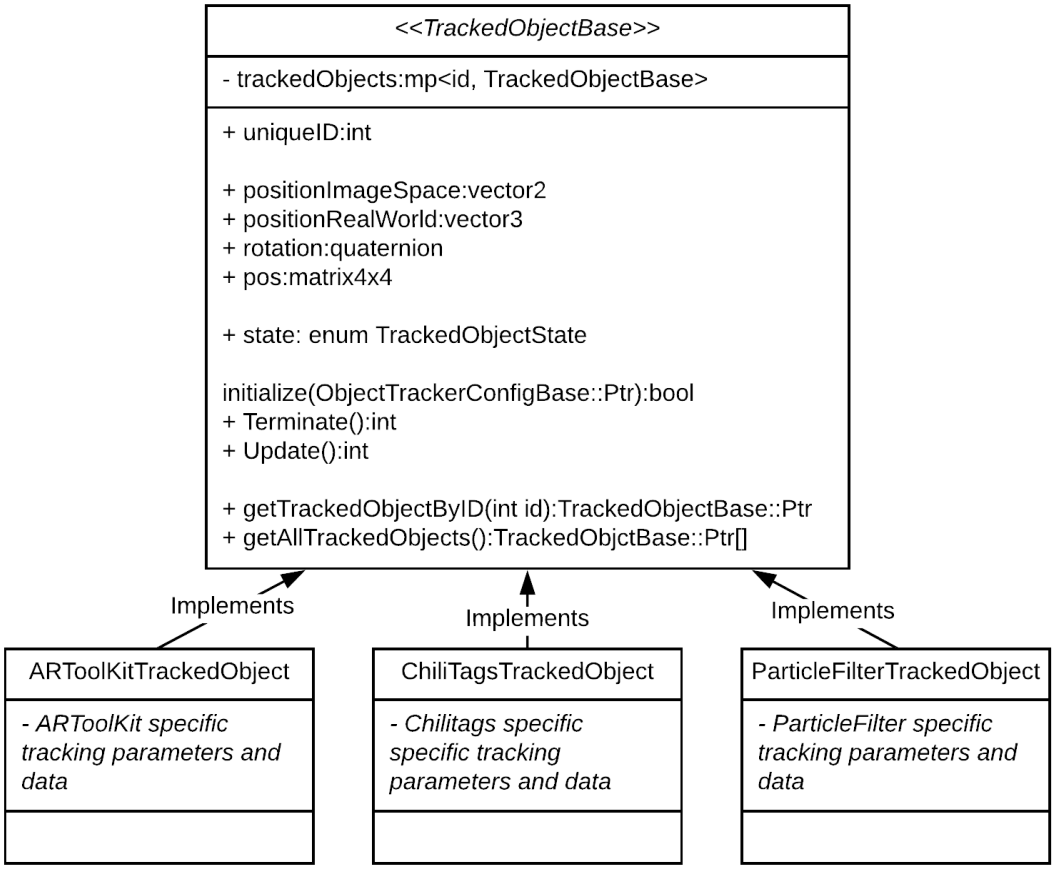


Figure 3.7: UML diagram for the TrackedObjectBase base class that provides a common interface to obtain object poses regardless of the underlying tracking technology.

object tracking process. UML diagrams for the aforementioned interfaces can be seen in Figures 3.6 and 3.7.

The ObjectTrackerBase class (Figure 3.6) is to be implemented by any tracking libraries which are to be experimented with. This class provides a common interface for the application to update and obtain the results of the object tracking process. The TrackedObjectBase class provides a common interface which is used to encapsulate the results of the object tracker. The intuition behind this design stems from the realization that all object trackers return two things, an identifier for the object and the pose of the object. Each object tracked by the object tracker will have an instance of the TrackedObjectBase class.

The following subsections discuss two tracking methodologies that were explored during the development of StoryTIME, each of the mentioned tracking technologies implement the object tracking base classes.

Prototype v1.0: Markerless Tracking

Markerless tracking technologies were explored in early prototypes of StoryTIME. The initial objective during development was to track objects without the need for any tracking aids such as AR markers by using natural features present in the object [52]. An early experimentation example can be seen in Figure 3.8. This example illustrates the tracking of an object, in this case a chair, using nothing but the features of the chair itself. The blue points show the result of the tracking, as we can see the tracking result aligns with the chair quite accurately. The biggest issue with this approach was computation time, the performance for the single object was about 2 frames per second. This impacted StoryTIME's ability to deliver its requirements and required development of an alternate solution.

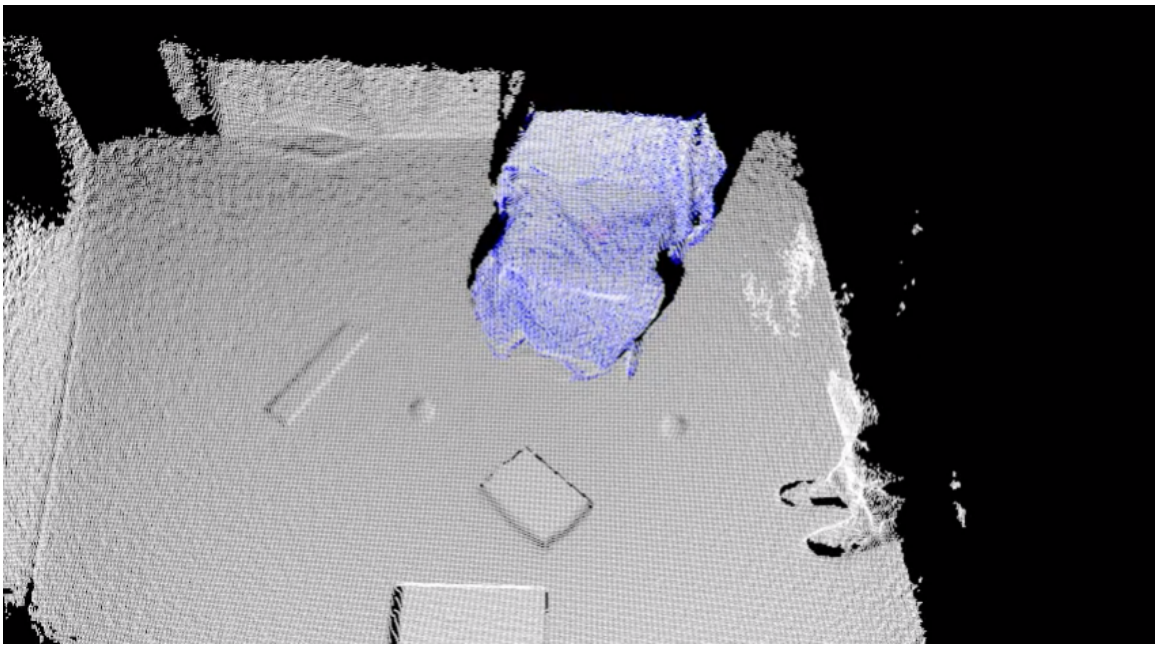
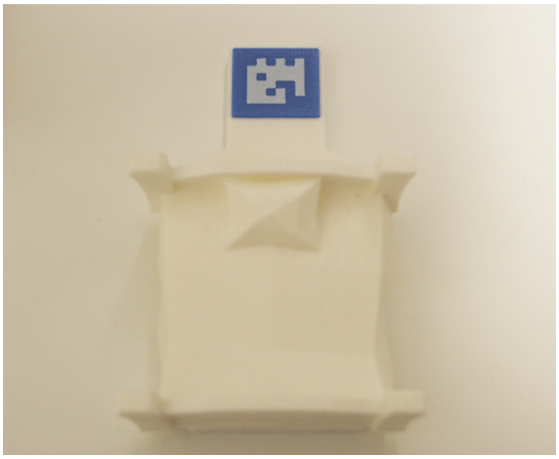


Figure 3.8: Early experiment with particle filter based tracking using natural features. This image demonstrates the alignment of a virtual chair (blue points) with its physical counterpart (grey points).

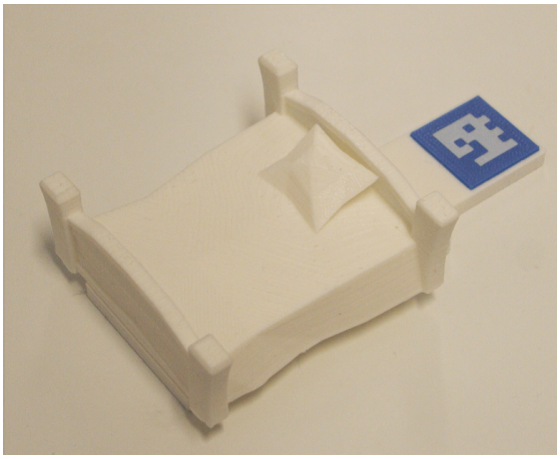
Prototype v2.0: Marker Based Tracking

The markerless tracking approach resulted in accurate pose estimation but unsatisfactory responsiveness. To overcome this issue we resorted to marker based tracking. Fiduciary AR markers were affixed to the objects that were to be tracked, as shown in Figure 3.9. The benefit to using AR markers is that they are significantly faster than using markerless approaches since the features are simple and well defined. These markers were 3D printed using matte materials. The markers seen in Figure 3.9 were 3D printed instead of being printed on paper to overcome the issues of cosmetic damage to the marker and reflectance. If an AR marker is cosmetically damaged, its trackability is compromised since the mapping between observed points and reference points is broken. As a result, the returned transformation will not represent the actual pose of the marker. Tracking algorithms are very sensitive to bent markers, especially if they assume the points on the marker are planar. Since it is anticipated that the props will be subject to daily use, it is crucial to ensure they are reasonably

durable. Marker reflectance issues happen when a marker is shiny. When a marker is shiny, it can appear incorrect to the camera due to specular highlights that occur as the object is rotated which results in the camera observing unexpected colours. The 3D printed markers are made of matte plastic, which is more durable than paper and therefore less susceptible to cosmetic damage and reflectance issues.



(a) Bed prop, top down view



(b) Bed prop, alternate view

Figure 3.9: Example of a trackable StoryTIME prop. An AR marker is affixed to the object.

Marker visibility must also be considered when designing an interface based on AR markers. In order for a marker to be detected and tracked, it must be visible to the camera. The use of a *multi-marker* configuration can help overcome this limitation. A multi-marker configuration refers to using multiple markers instead of one. Each marker in the configuration will be aware of their position *relative* to the origin of the multi-marker. The idea is that by detecting one marker, we can infer the poses of the other markers. An example of a multi-marker can be seen in Figure 1.9 where six AR markers are laid out in a cube formation. The rationale for the cube formation is that regardless of the marker's pose, at least one marker will be visible to the camera. The cube marker prototype was developed for use with the ARToolKit tracking library. ARToolKit had extremely fast tracking performance and was able to track 64 markers in less than 4 ms on the HP Sprout 3.2. Unfortunately the tracking results were not

very accurate and marker detection was erroneous due to ARToolKit's inability to compensate for noisy images, partial occlusion of markers and changes in luminance. ARToolKit was essentially the polar opposite of the markerless tracking approach previously discussed. Chilitags is another AR tracking library which like ARToolKit, relies on fiduciary AR markers. The difference being that Chilitags is robust to the aforementioned limitations of ARToolKit. StoryTIME ultimately ended up using Chilitags, with a single marker configuration.

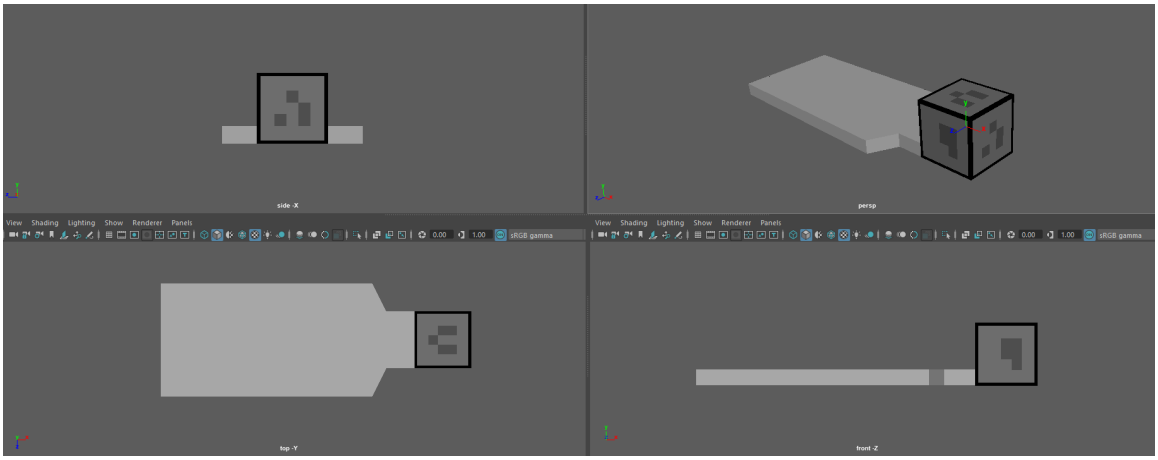


Figure 3.10: Early prototype of the AR marker cube.

3.3.4 Calibration

One of the design goals for StoryTIME was to bring the physical objects to life through the use of virtual augmentations. As discussed in Chapter 2, there are many options for displaying virtual augmentations. StoryTIME used projection mapping to provide the user with visual feedback. By using a projector we are able to project augmentations directly onto the physical props, as shown in Figure 3.13. This allows users to see the augmentations while remaining unencumbered and without needing to take their attention off of the props they are interacting with. To achieve this we must be able to "pre-warp" the projected images so that they appear correct when displayed on the irregular geometry of the physical object. One way to think about this is to consider that both a camera and a projector can be modeled by a pinhole

projection. A diagram of a pinhole projection for a projector camera setup can be seen in Figure 3.11. With a pinhole projection, a camera forms an image with light rays that pass through an infinitesimal point (the pinhole) that intersect with the imaging plane [53]. A projector essentially works in the opposite way, instead of light rays passing through the pinhole and hitting the image plane, the light rays emit from the plane and pass through the pinhole, eventually hitting the projection surface.

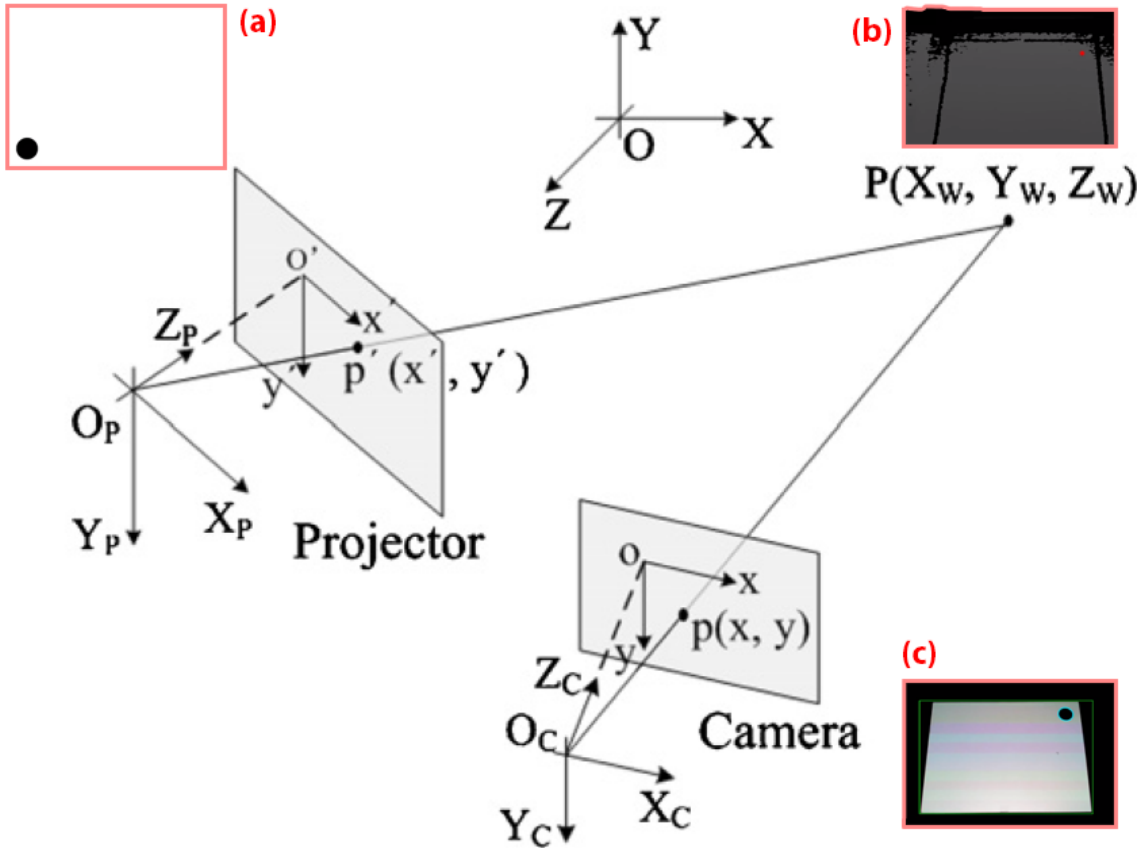


Figure 3.11: Pinhole projection for a camera and projector. If a point in the world can be observed by a camera and a projector, we can determine the extrinsic transformation between the two coordinate systems. (a), (b) and (c) are described in Figure 3.12

.

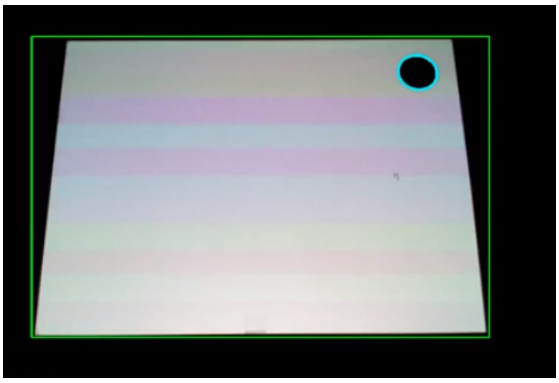
To accurately generate images we need to know the projection parameters of the projector which are obtained by performing a projector camera calibration. The projection parameters that turn 3D points into 2D pixels can be calculated by having

a set of 3D points and their corresponding 2D projections. The procedure to perform the calibration is summarized below [53]:

1. Project a point (Figure 3.12a).
2. RGB camera observes point (Figure 3.12b)
3. Detect 2D coordinate P_{RGB} of point in RGB image
4. Map P_{RGB} to a pixel on the depth image P_{Depth} (Figure 3.12c)
5. Sample depth value from depth image and calculate 3D point relative to depth camera
6. Store point correspondence in an array

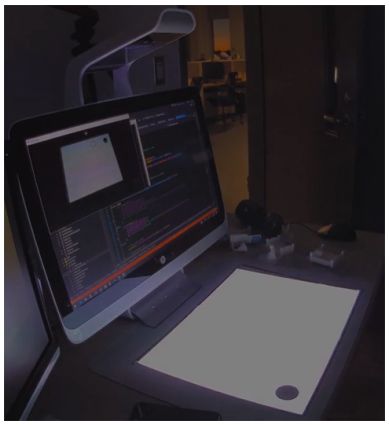
Once a set of point correspondences are obtained, we pass them to OpenCV's calibrate camera function which performs the DLT algorithm to obtain the extrinsic and intrinsic parameters for the projector [54]. Extrinsic parameters refer to the pose of the projector, it answers the question of "what orientation was the projector in when these points were observed". Intrinsic parameters define the mapping from 3D to 2D, they answer the question of "how do these 3D points get turned into 2D pixels" [53]. These parameters are used heavily in the Transformation Pipeline described in Chapter 3.4. Figures 3.11 and 3.12 illustrate the calibration procedure. In Figure 3.12, we see the various images used by the calibration procedure. Figure 3.12a shows a dot projected by the projector. This dot is observed by the RGB camera as shown in Figure 3.12b. The RGB image is processed to find the 2D coordinate of the dot in the frame, this 2D coordinate is then used to read from the depth image shown in Figure 3.12c. With the depth value we are able to reconstruct a full 3D coordinate for the projected dot. This process is repeated several times with dots in different locations. A minimum nine point correspondences are needed for the DLT algorithm.

(a) The image projected by the projector. The dot has a known 2D position



(b) The projected dot as viewed by the RGB sensor.

(c) The projected dot as viewed by the depth sensor. Note the red mark is just for visualization.

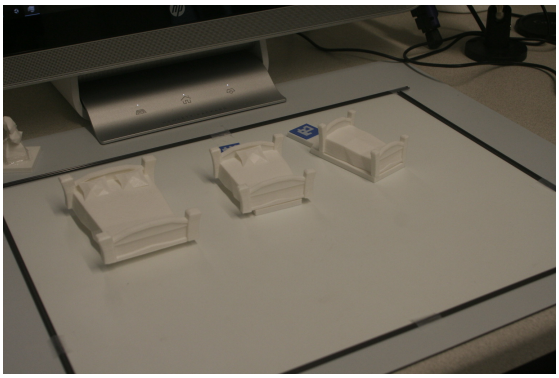


(d) Visualization of the real-world setup

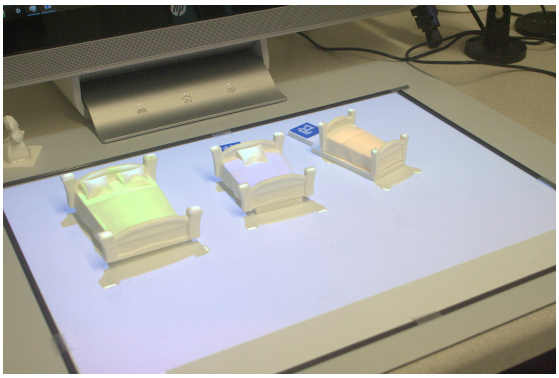
Figure 3.12: Demonstration of the different images used to perform calibration

Figure 3.11 shows where the images in Figure 3.12 are used in respect to the pinhole projection model. Lens distortion parameters are also computed using the same point

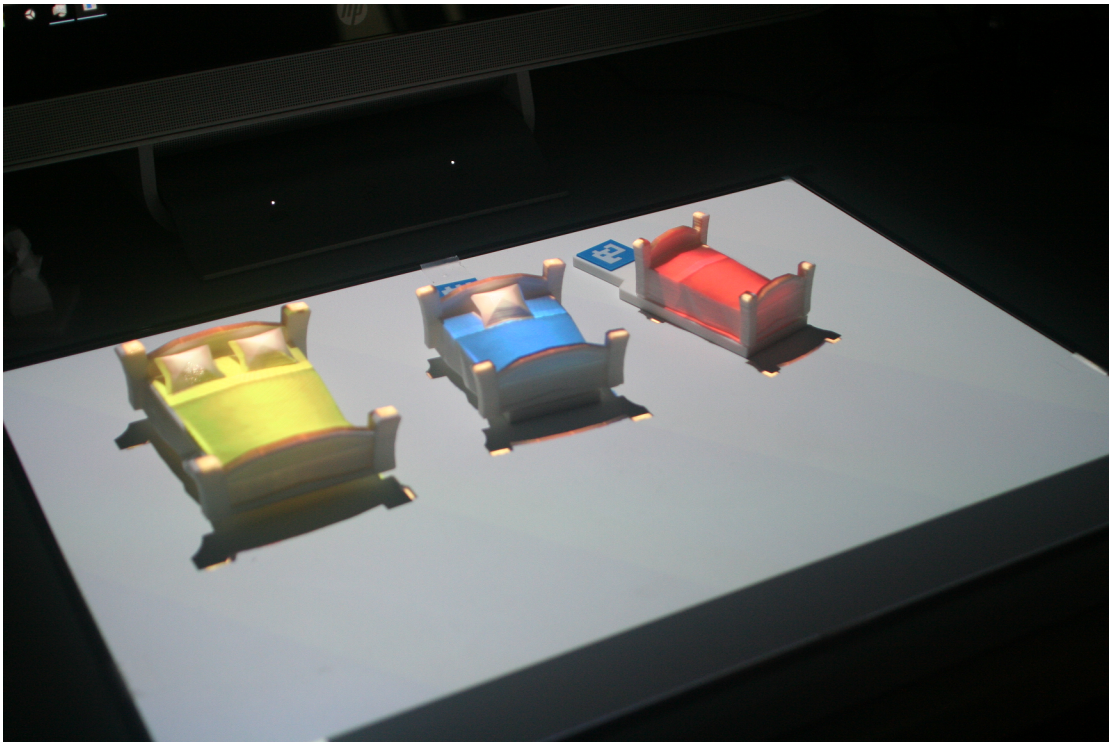
correspondences, these distortion parameters allow us to take lens distortion artifacts into account.



(a) Physical props in their natural state, projections turned off.



(b) Physical props with projections turned on in an environment with high ambient light. Notice how the projections are washed out.



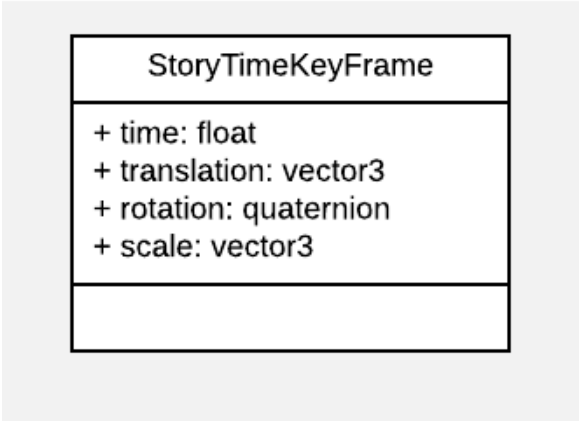
(c) Physical props with projections turned on in an environment with low ambient light. Notice how the projections are significantly more vibrant.

Figure 3.13: Demonstration of the spatial augmentations in StoryTIME.

3.3.5 Recording and Playback

Recording object movement can be performed once the pose is determined by the object tracker. To inform the system of their intent to begin recording, the user must click the "record" button shown in Figure 3.17a box 4. Once this button is clicked, for every subsequent frame in which an object's pose is tracked, a *StoryTimeKeyFrame* object is created and is appended to an animation look up table, this is the process shown in Figure 3.1 box 11. A UML diagram for the StoryTIMEKeyFrame class can be seen in Figure 3.14a. The animation look up table is stored as a *dynamic array*. An array is a data structure where data is stored contiguously in memory. A visualization of the animation table can be seen in Figure 3.14b.

When the user clicks the play button, a playback timer is started. This timer is used to sample from the animation table and performs a linear interpolation to get the position of the virtual object.



(a) UML diagram for the StoryTIMEKeyframe class. Each frame of an animation is represented by an instance of this object.

Time (seconds)	Position X	Position Y	Position Z	Quaternion X	Quaternion Y	Quaternion Z	Quaternion W
0.032	-0.00232311	-0.0870005	0.586959	-0.606338	0.78515	-0.102809	-0.0729717
0.034	-0.00232311	-0.0870005	0.586959	-0.606338	0.78515	-0.102809	-0.0729717
0.037	-0.00232311	-0.0870005	0.586959	-0.606338	0.78515	-0.102809	-0.0729717
0.041	-0.00232311	-0.0870005	0.586959	-0.606338	0.78515	-0.102809	-0.0729717
0.044	-0.00231322	-0.0869696	0.587071	-0.606491	0.784978	-0.102965	-0.0733203
0.071	-0.00231322	-0.0869696	0.587071	-0.606491	0.784978	-0.102965	-0.0733203
0.078	-0.00231322	-0.0869696	0.587071	-0.606491	0.784978	-0.102965	-0.0733203
0.084	-0.00229708	-0.0870262	0.58689	-0.606684	0.784913	-0.102743	-0.0727319
0.113	-0.00229708	-0.0870262	0.58689	-0.606684	0.784913	-0.102743	-0.0727319
0.12	-0.00233869	-0.086998	0.587022	-0.606423	0.785102	-0.102888	-0.0726605
0.151	-0.00233139	-0.0870073	0.58683	-0.606455	0.785085	-0.102843	-0.0726501
0.189	-0.0023038	-0.0870061	0.586814	-0.606657	0.784918	-0.102899	-0.0726838

(b)

Figure 3.14: Example of the recorded animation data.

3.4 The Transformation Pipeline

Aligning the Virtual and Physical Worlds

The prior sections established the necessary pieces for implementing StoryTIME. This section will assemble the pieces and present the full mathematical pipeline.

In order to augment a physical object, as shown in Figure 3.13 we must have the projector project a virtual render of the physical object in such a way that the virtual render lands right on top of the physical object. To do this we must have a virtual representation of the physical object in the form of a triangulated mesh. In the case of StoryTIME, we used a 3D printer to create a physical object from the virtual mesh. The process of performing the augmentation requires a series of

spatial transformations. Spatial transformations in this work are represented by the T symbol. A spatial transformation is represented using a 4x4 transformation matrix as shown in Equation (3.1). The 4x4 matrix consists of a rotation-scale, denoted by rs and translation, denoted by t in Equation (3.1). Figure 3.15 shows the various coordinate spaces used in the StoryTIME transformation pipeline. As an example of a spatial transformation, consider a point that is expressed relative to coordinate system "A", if we wanted to express the point terms of coordinate system "B", we can "change the basis" of the point by applying the transformation that aligns coordinate system B to coordinate system A.

$$\begin{bmatrix} rs_{00} & rs_{01} & rs_{02} & t_x \\ rs_{10} & rs_{11} & rs_{12} & t_y \\ rs_{20} & rs_{21} & rs_{22} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

The series of transformations to align a virtual object with its physical counterpart is shown in Equation (3.2). Note that StoryTIME follows the column major matrix notation and the transformation presented in Equation 3.2 is to be read from right to left.

$$V_{projector} = T_{projector} \cdot T_{depth} \cdot T_{marker} \cdot T_{object} \cdot V_{local} \tag{3.2}$$

where:

V_{local}: local position of the vertex, represented as a 4D vector

T_{object}: offset from marker to physical object, represented as a 4x4 matrix

T_{marker}: pose of the tracked marker as returned by the object tracker each frame

T_{depth}: extrinsic transform from 3D RGB camera space to 3D depth camera space

$T_{projector}$: extrinsic transform from 3D depth camera space to projector space

$V_{projector}$: vertex in 3D projector space

Figure 3.15: Visualization of the different coordinate spaces in the transformation pipeline. **A** is the colour camera, this space is referred to as "RGB camera space". **B** is the depth camera, this space is referred to as "Depth camera space". **C** is the projector, this space is referred to as "Projector space". **D** is the physical object, this is referred to as "object space". **E** is the AR marker, this is referred to as "marker space".

The object tracker (discussed in Section 3.3.3) returns the 4x4 transformation T_{marker} which is depicted as **E** in Figure 3.15, notice that there is an offset between the tracked pose **E** and the actual physical object **D**. The first step of Equation (3.2) is to shift the origin of the virtual mesh to account for this offset. This is referred to as the *marker to object* transformation T_{object}. Then we apply the transformation obtained by the object tracker T_{marker}. This gives us a 3D pose relative to the RGB

camera (**A** in Figure 3.15). Since the projector calibration (discussed in Section 3.3.4) used points in depth camera space, we need to apply the extrinsic transform between the RGB and depth cameras T_{depth}. Once in depth space, we can apply the depth to projector transform $T_{projector}$ to obtain a 3D coordinate relative to the projector. Now that we have a 3D point in projector space, we need to project it to a 2D pixel coordinate on the projector's image plane, the process is performed in a vertex shader and is described below.

Forming the Image

The first step is to turn the 3D point $V_{projector}$ into a 2D coordinate on the projector's image plane. This is done by performing *perspective division* which means dividing the x and y coordinates of the 3D point by the z component, as shown in Equation (3.3).

$$\begin{aligned} x &= \frac{V_{projector.x}}{V_{projector.z}} \\ y &= \frac{V_{projector.y}}{V_{projector.z}} \end{aligned} \tag{3.3}$$

where:

x: x coordinate on the projector's image plane

y: y coordinate on the projector's image plane

$V_{projector}$: a 3D point in projector space

Next we need to apply the projector's lens distortion. We use OpenCV's distortion model, which is comprised of radial and tangential distortion as shown in Equation (3.4).

$$
\begin{aligned}
RadialDistortion &= 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 \\
d_x &= x \cdot RadialDistortion + p_1 2xy + p_2 r^2 + 2x^2 \\
d_y &= y \cdot RadialDistortion + p_2 2xy + p_1 r^2 + 2y^2
\end{aligned}
\tag{3.4}
$$

where:

$RadialDistortion$: the total amount of radial distortion

x: x coordinate on the projector's image plane

y: y coordinate on the projector's image plane

r: radial distance from center of image, calculated as $x^2 + y^2$

d_x: distorted x coordinate on the projector's image plane

d_y: distorted y coordinate on the projector's image plane

k_1, k_2, k_3: radial distortion coefficients

p_1, p_2: tangential distortion coefficients

At this point we have distorted the points on the image plane to account for lens distortion. The next step is to calculate actual pixel coordinates. This is done by scaling the plane by the focal lengths and applying the principal point offset as shown in Equation (3.5).

$$
P_{xy} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} d_x \\ d_y \\ 1 \end{bmatrix}
\tag{3.5}
$$

where:

P_{xy}: pixel coordinate on projector image, note that the z coordinate is discarded

f_x: focal length on x axis

f_y: focal length on y axis

s: skew coefficient, in the calibration used by StoryTIME this is equal to 0

d_x: distorted x coordinate on the projector's image plane

d_y: distorted y coordinate on the projector's image plane

We now have projected the 3D point in projector space to a pixel on the projector's image. However in-order to integrate this equation into the rasterization pipeline, we must transform the pixel coordinates into *normalized device coordinates* (NDC). NDC is a space defined by the graphics API that dictates which points end up on the screen, if a point is outside of this range, the graphics API will *clip* it. StoryTIME uses the OpenGL graphics API which defines NDC as a unit cube ([-1, 1] on each axis) where the origin of the cube is the center of the screen. The above transformations are occurring in the first stage of the graphics pipeline which is the vertex shader [55]. The responsibility of the vertex shader is to output points in such a way that ensures the rest of the pipeline can calculate pixel coordinates. Since pixel coordinates we have at this point are well beyond the [-1, 1] range, all of the points will get discarded and we will end up with a blank screen. The equation to transform a pixel coordinate into NDC is shown in Equation (3.6). We first divide the pixel coordinate by the image's width and height which will put points on the image into the [0, 1] range, then we shift into the [-1, 1] range.

$$\begin{aligned} NDC_x &= \frac{P_x}{w} * 2 - 1 \\ NDC_y &= \frac{P_y}{h} * 2 - 1 \end{aligned} \tag{3.6}$$

where:

NDC_x, NDC_y: x and y NDC coordinates for projected points

w: image width

h: image height

The final step is to calculate the z coordinate for the depth buffer. This coordinate is necessary to ensure that depth testing can be properly performed. Without a valid depth buffer we will have rendering artifacts since the rasterization pipeline used to render the 3D meshes will not be able to determine which triangles should be in-front of others. Equation (3.7) shows the transformation. This equation comes from the traditional perspective matrix used in computer graphics. Note that this equation embeds the transformation into NDC.

$$NDC_z = V_{projector.z} \cdot \frac{f+n}{f-n} + V_{projector.z} \cdot \frac{-2f \cdot n}{f-n} \tag{3.7}$$

where:

NDC_z: z NDC coordinate for projected points

f: far clip plane

n: near clip plane

Summary

That was the procedure StoryTIME uses to perform projection mapping. We transform the mesh into projector space, then we project the mesh onto the projector's image plane and finally we apply the NDC transform to adhere to the graphics API specification.

3.5 User Interface

The goal of StoryTIME is to allow for the prototyping of animated sequences by manipulating physical objects. These manipulations are digitized and turned into

a rendered 3D objects. To demonstrate the capability of StoryTIME, we opted to create a series of props pertaining to the story of Goldilocks and the Three Bears, the props are shown in Figure 3.16. This story was chosen because it was simple, familiar and depicted everything the system could do. We specifically wanted to recreate the scene where the main character, Goldilocks, enters a room with three beds and she inspects each of them one by one before deciding which one she wants to rest on. The following points are a breakdown of the tasks involved with creating this scene:

- **Staging:** how the beds laid out in the scene. Where Goldilocks enters the scene from how does she navigate around the room.
- **Framing:** how the observer is viewing the scene.
- **Multiple possibilities:** there is no one way of staging and framing this scene.

Figure 3.16 shows the six props created to demonstrate the capabilities of StoryTIME. Each of these objects can be tracked and recorded simultaneously in real-time. Objects 1, 2, and 3 are the large, medium and small beds. The user has the ability to place these beds where ever they please by simply moving the physical prop however they choose. Object 4 is the camera, moving this object moves the virtual camera. This allows users to pose the camera to achieve the framing they desire. Object 5 is Goldilocks. Object 6 represents a point light in the scene, moving this object allows users to change the shading in the room.

Figure 3.17 shows the system in action. Figures 3.17a and 3.17b show the scene with two different layouts. The following list summarizes the characteristics of the UI:

Figure 3.17a

1. The new take button allows users to create a new animation of their scene, this will save their current layout and recordings and gives them a new blank slate which they can use to create a new take on the scene.

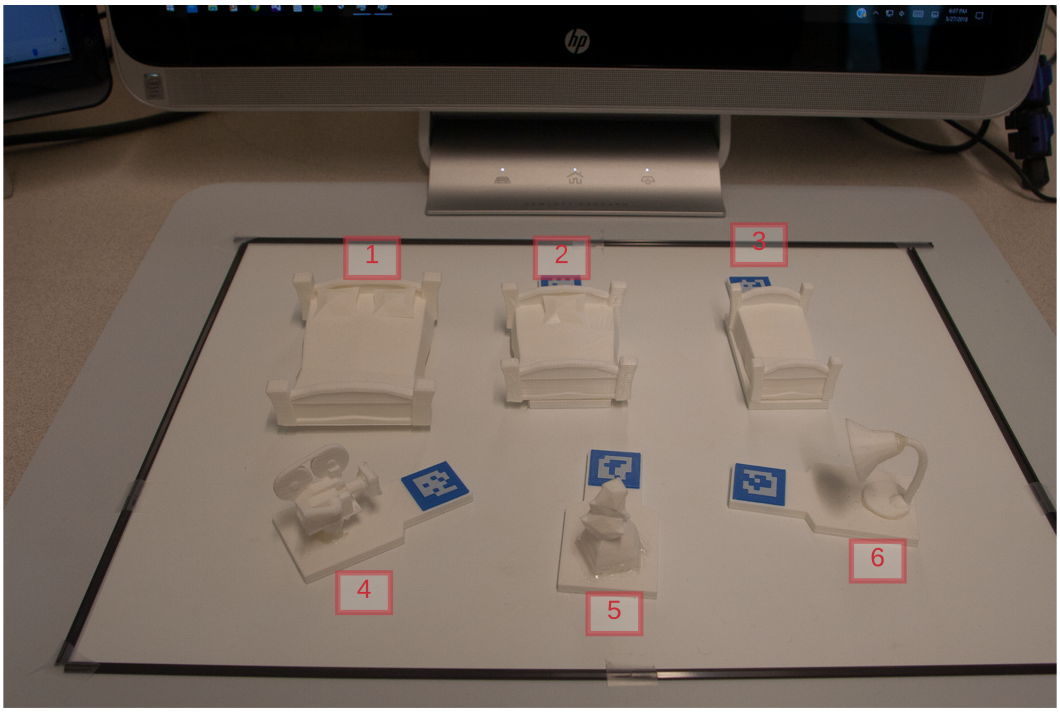


Figure 3.16: The physical objects used in the StoryTIME tangible user interface. Objects 1,2 and 3 are beds. Object 4 is a special object representing a camera. Object 5 is Goldilocks. Object 6 is a light.

2. A previously recorded take, currently greyed out because it is not the active take. If the user were to click on this thumbnail, it would become the active take and its scene would be restored on the canvas (5).

3. The currently active take that the user is manipulating.

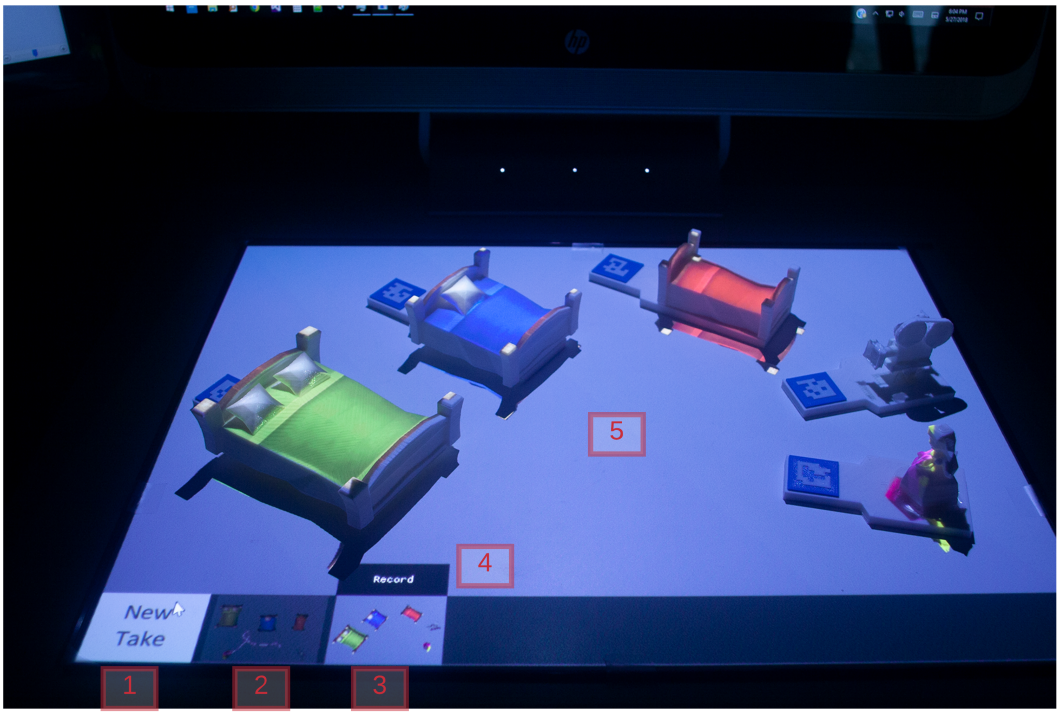
4. The record button, once the user clicks this button the system transitions to the recording state and all actions applied on the physical props will be recorded.

5. The interaction canvas, this is the entire gray rectangle in the image. All objects in this are are trackable by the system. The various props are the tangible objects the user interacts with.

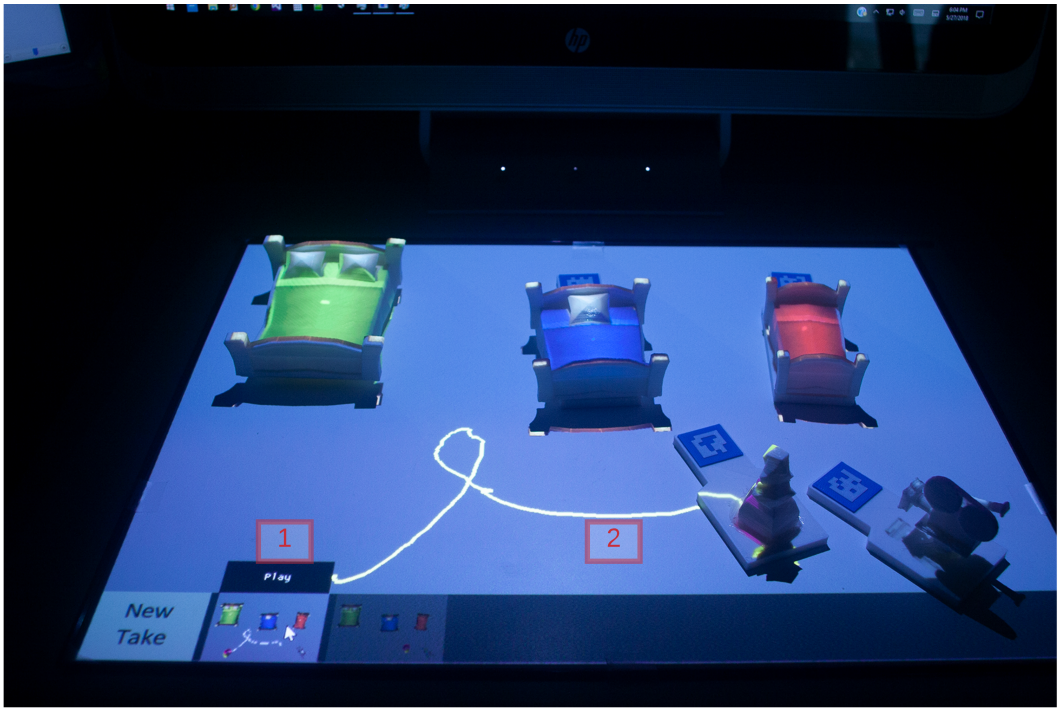
Figure 3.17b

1. The play button, once an animation has been recorded, the user will have the ability to play it back by clicking on this play button.

2. The yellow curve represents the recorded path for Goldilocks. When the user clicks on the play button (1), a virtual render of Goldilocks will be drawn that follows the path exactly as recorded by the user.



(a)



(b)

Figure 3.17: Demonstration of the UI elements in StoryTIME.

3.6 Summary

In this section we broke down the essential elements that went into creating StoryTIME. We discussed the challenges involved with image acquisition, object tracking and calibration and how we overcame them. We also discussed how all of those components were put together to form a coherent system that allows users to prototype animated sequences using physical objects. The next chapter presents an evaluation which aims to determine how StoryTIME compares to established prototyping software.

Chapter 4

Evaluation of StoryTIME

Chapter 3 discussed the underlying technical elements that went into building StoryTIME. This chapter evaluates StoryTIME as a tool for prototyping animated sequences. A preliminary user study was conducted asking participants to prototype an animated sequence using both StoryTIME and the Unity game engine. Unity is an established engine and was chosen as the ground line comparison. The user study is essentially a comparison between a new interface (StoryTIME) and a traditional interface (Unity).

4.1 Research Questions and Hypotheses

The user study captured metrics for user experience, preference, performance and system usability. These metrics were captured to answer the following questions:

> **Does prototyping with StoryTIME improve iteration time when developing pre-visualizations?** This question aims to address the user's performance characteristics in terms of *completion time* and *failure rate*. Completion time refers to the amount of time it took participants to complete the predetermined tasks. Failure rate measures whether or not participants were able to complete the given tasks. The tasks performed by the users are described in Section 4.3.

Hypothesis 1 (H1): *iteration time will be faster when using StoryTIME compared to Unity.*

How does prototyping pre-visualizations with tangibles compare with traditional keyboard and mouse based tools in terms of usability and user experience? This question aims to measure *user experience* and *system usability*. There are many metrics that can be measured when discussing user experience, this study focused on the user's mood. To measure the user's mood the Positive and Negative Affect Scale (PANAS) [56] discussed in Section 4.4.2, was used. System usability refers to the user's ability to complete the given tasks and was measured with the System Usability Scale (SUS) [57], discussed in Section 4.4.3.

Hypothesis 2 (H2): *usability and experience will be either the same or better with the tangible interface when compared with the traditional interface.*

How does displaying augmentations on a computer monitor compare with projected augmentation in terms of user preference and performance? This question explores the user's *preference* and *performance* when using projected virtual augmentations compared to virtual augmentations displayed on a computer monitor. To measure preference, the post activity questionnaire discussed in Section 4.4.4 was used. The metrics used to capture performance are: time to completion and accuracy.

Hypothesis 3 (H3): *users will prefer and perform better with projected augmentations.*

4.2 Experimental Setup

The study was conducted in the Games and Media Entertainment Research Lab (GAMERLab) at the University of Ontario Institute of Technology (UOIT). The

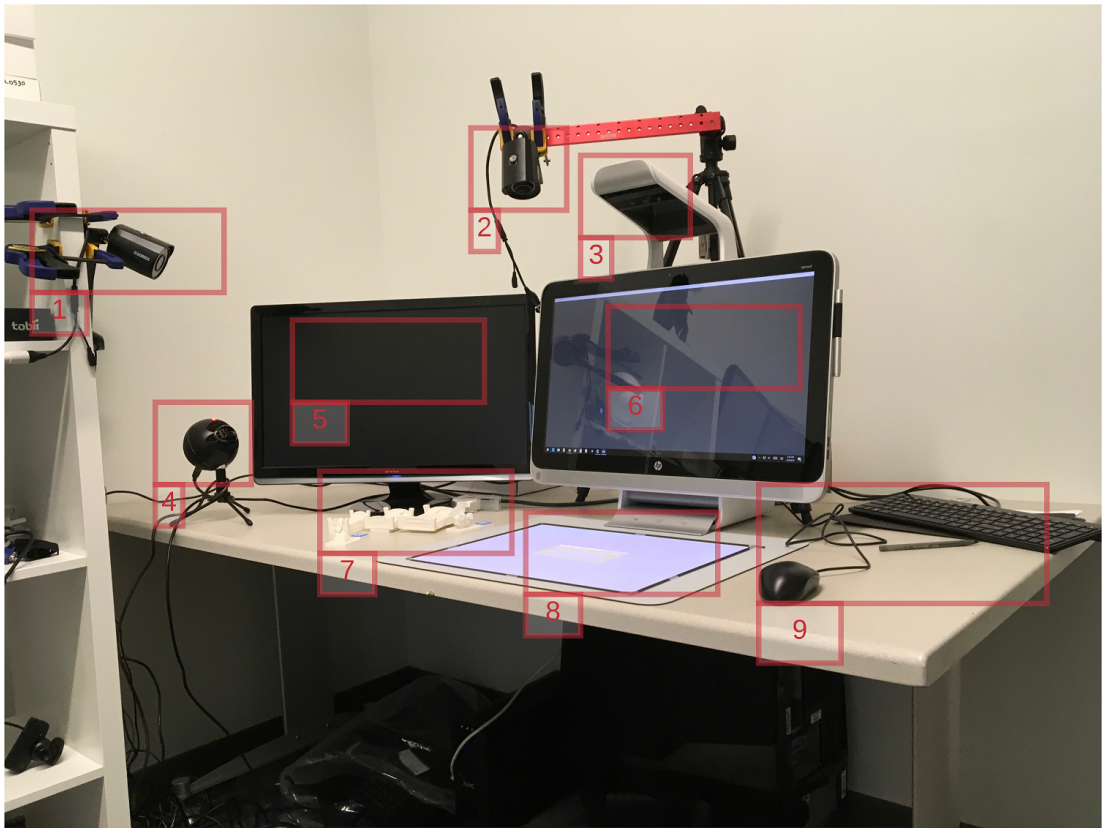


Figure 4.1: Experimental setup used for the StoryTIME user study. (1) and (2) are cameras used to record the session. (3) is the projector camera setup on the HP Sprout. (4) Microphone used to record audio. (5) Secondary display used during the A/B test. (6) The primary display. (7) The physical props used by participants in the StoryTIME condition. (8) The StoryTIME interaction area. (9) Keyboard and mouse used with in the Unity condition.

software for the experiment was run on an HP Sprout (Appendix A). An image of the experimental setup can be seen in Figure 4.1. The study was run in a controlled environment, only the participant and researcher were in the room during the study. The numbered items in Figure 4.1 are described below:

1. Camera used to observe the primary display.

2. Camera used to observe the interaction area while users were interacting with StoryTIME.

3. The projector camera setup on the Hp Sprout. This includes the RGB colour and Depth cameras used to track the AR markers on the props. The projector is used to project augmentations into the interaction area (8).

4. Microphone used to record the session.

5. Secondary display used to show directives for the A/B test described in Section 4.3.3.

6. The primary display, during the StoryTIME condition, this is where the real-time virtual representation of the physical scene is shown (discussed in Section 3.5). During the Unity condition, this is where the Unity interface is shown.

7. The physical props used by the StoryTIME system (discussed in Section 3.5). These are the props participants manipulate to prototype their animated sequences when using StoryTIME. These props have no purpose in the Unity condition.

8. This is the interaction area where StoryTIME is able to track, detect and augment the physical props (7).

9. Keyboard and mouse used in the Unity condition.

4.3 Methodology

The study was approved by the Research Ethics Board (REB) at the University of Ontario Institute of Technology (REB# 14438).

The study followed a structured flow which was kept constant for all participants, a visual depiction of procedure can be seen in Figure 4.2. The study is broken into three phases which are summarized below:

Phase One: participants are briefed on the tasks they will be completing and are asked to complete a consent form, demographics survey and PANAS questionnaire.

Phase Two: participants complete the main task of prototyping an animation sequence using StoryTIME and Unity.

Phase Three: participants complete an A/B test alternating between projected virtual augmentations and augmentations displayed on a monitor.

Participants were able to withdraw at anytime without penalty and any data collected would be promptly destroyed. The three phases in the study are described in detail below:

4.3.1 Phase 1: Introduction

The first phase of the study can be seen in the green section in Figure 4.2. Upon entry, participants were welcomed and briefed on the tasks they would be performing. The following script was used for each participant:

> *This study is an exploration into user interfaces for creating animated sequences. You will be re-creating a scene from the story of Goldilocks and the Three Bears. In this scene Goldilocks inspects three beds before deciding which one she wants to take a nap in. You will be creating this scene twice using two different interfaces. First you will be using Unity which has a traditional key-framing and then you will use StoryTIME, a system which captures the motion of physical props*.*
>
> **The order of the last sentence was swapped depending on the order the conditions were used.*

After the participants were briefed, they were asked to sign a consent form shown in Appendix B. If the participant agreed to continue they were asked to complete the

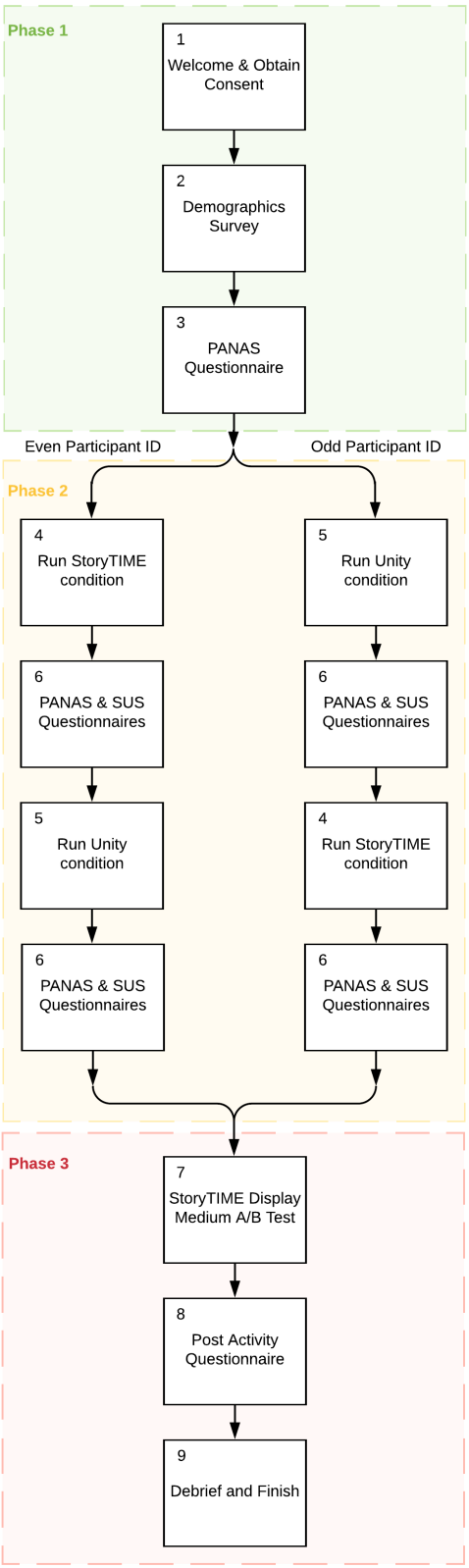


Figure 4.2: Study procedure

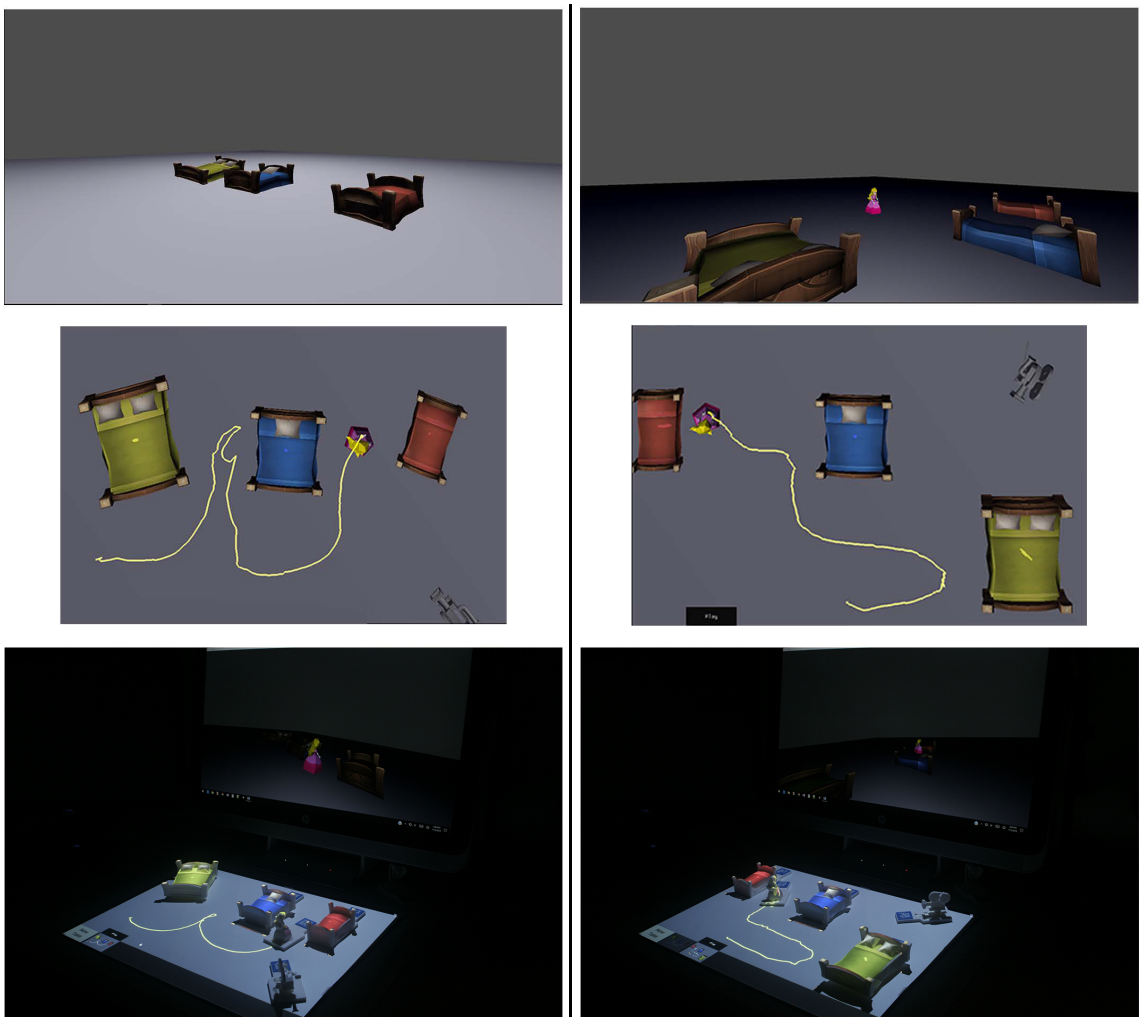
PANAS for the first time. The PANAS was administered three times in total for each participant, once at the very beginning, and then after each successive condition. The reasoning being that we wanted to see if there were any changes in the participant's mood from before the conditions and after.

4.3.2 Phase 2: Prototyping an Animated Sequence

Phase two is where the main experiment occurs, in this stage participants are asked to position objects in a scene and animate a character within it. They are then asked to create a second iteration of the scene which changes the layout and the character animation. The two scenes can be seen in Figures 4.3a and 4.3b.

Since participants were asked to create the same scene in both StoryTIME and Unity, the possibility of a learning bias is introduced. For example, if we always performed the Unity condition first, and the StoryTIME condition second, this may give StoryTIME an unfair advantage, since participants are already familiar with the tasks they will be asked to perform. To overcome this, the condition order is alternated between participants. This is done by having the participants with an even *participant number* perform the StoryTIME condition first, and performing the Unity condition second, this is the opposite case for participants with an odd number. Participant number refers to a unique number assigned to each participant. This number starts at 0, for the first participant and increments by 1 for each subsequent participant.

Upon completion of each condition, participants are asked to complete the PANAS and SUS surveys. When both conditions are complete, we begin the third and final phase.

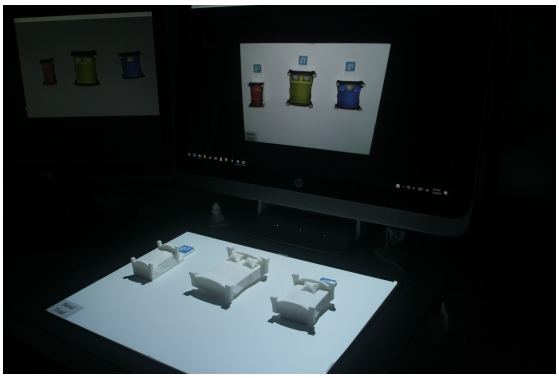


(a) (b)

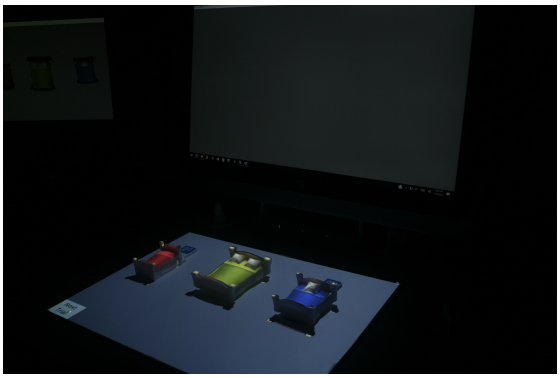
Figure 4.3: The two scenes participants were asked to re-create. The path for the animated character is shown in yellow. The top image shows the virtual recreation of the scene from the point of view of the virtual camera. The middle image shows a top-down layout of the scene. The bottom image shows an example of how the scenes are created using the tangible props.

4.3.3 Phase 3: Alpha Beta Test and Debriefing

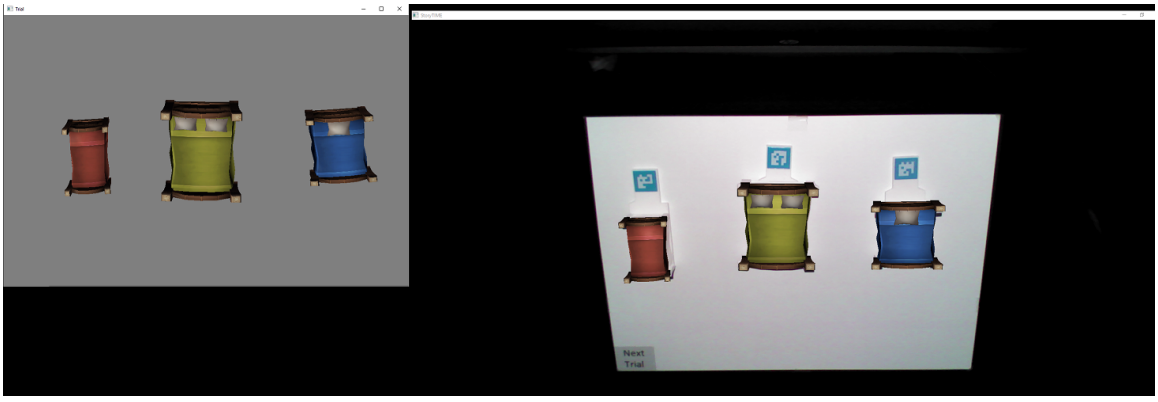
Upon completion of the animation tasks, an A/B test was conducted to determine if projected AR had any benefits in terms of performance, accuracy and preference to augmentations displayed on a monitor. Participants were given a layout of a scene and were asked to re-create it using just StoryTIME, as shown in the left side of Figure 4.4c. The first condition projected the augmentations directly onto the physical props, ask shown in Figure 4.4b and the second condition displayed the augmentations on a monitor, as shown in Figure 4.4a. Participants were asked to complete 10 trials, each trial consisted of a different layout and alternated between the two display mediums.



(a) Augmentations displayed on monitor



(b) Projected augmentations



(c) Left: layout participants were asked to recreate. Right: augmented camera image

Figure 4.4: Demonstration of the A/B conditions.

4.4 Results

4.4.1 Demographics

The demographics questionnaire (Appendix B) asked participants questions about their experience creating virtual environments and computer animation. A total of 10 participants were run in the preliminary user study. Educational background for participants included 2 undergraduate students, 6 MSc students and 2 PhDs. Each of the 10 participants were able to complete the requested tasks described in Section 4.3. The age of the sample population ranged from 19 - 38 years old (M = 25.4, SD = 5.816).

Participants were asked several questions about their experience developing virtual environments and animated sequences. First they were asked about the development tools they used. Unity3D was the most popular tool used among the participants, followed by Maya. On average participants spend 5 - 10 hours per-week developing virtual environments. Participants were also asked about prototyping techniques they used when developing virtual environments using a five point Likert scale where 1 = never and 5 = always. The most common prototyping techniques were paper prototyping (M = 4.40, SD = 0.516) and creating digital mockups (M = 3.5, SD = 0.850). Participants rarely utilized tangible prototyping tools (M = 1.4, SD = 0.843). A summary of the responses can be seen in Figure 4.5.

4.4.2 The Positive and Negative Affect Scale (PANAS)

The Positive and Negative Affect Scale (PANAS) [56] is a questionnaire which measures the mood and emotions of the participant at the time of administration. Positive affect refers to positive emotions such as being interested or excited. Negative affect refers to negative emotions such as being upset or irritated. The questionnaire and the raw results gathered can be found in Appendix B.3. PANAS consists of 20 ques-

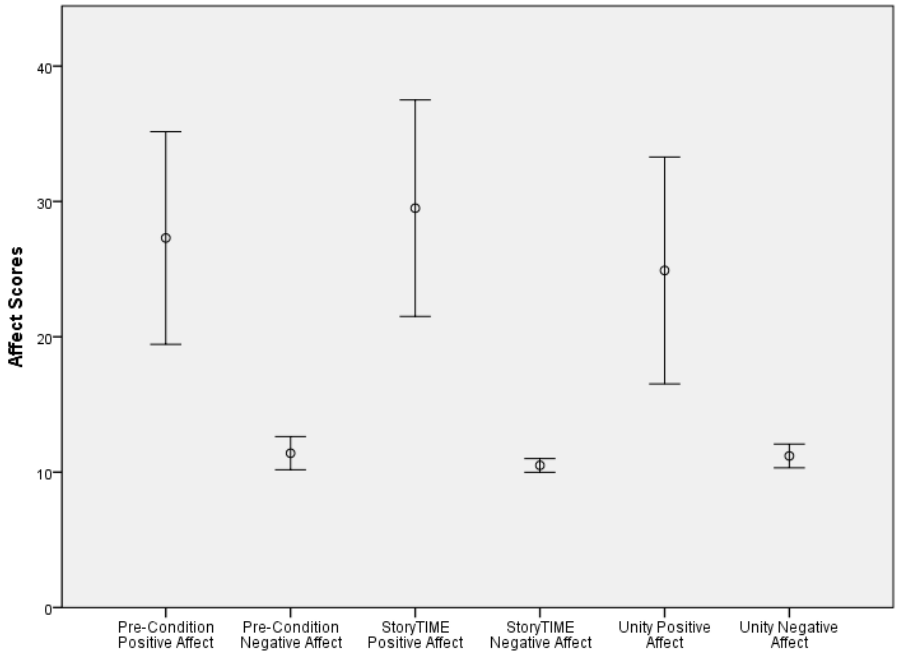
Frequency Using Various Prototyping Techniques

	N	Minimum	Maximum	Mean	Std. Deviation
Write Ideas on Paper	10	4	5	4.40	.516
Create Physical Mockup	10	1	4	2.10	.994
Create Digital Mockup	10	2	5	3.50	.850
Modify Existing Sample	10	2	3	2.70	.483
Tangible Prototyping Tool	10	1	3	1.70	.823
VR Prototyping Tool	10	1	3	1.40	.843
Valid N (listwise)	10				

Figure 4.5: Summary of the frequency using specific prototyping tools when creating a virtual environment. Results are from a Likert scale where 1 = Never and 5 = always.

tions, 10 of which measure positive affect and 10 which measure negative affect. Each question is in the form of a five point Likert scale. To calculate the scores for positive affect, we add up the measures for each of the 10 questions corresponding to positive affect. Calculating negative affect works the same way but the questions for negative affect are used instead. This gives us scores between 10 and 50 for both positive and negative affect where a higher score represents a higher level of affect. The questionnaire is administered three times during the duration of the experiment, prior to performing either of the conditions (Figure 4.2 box 3) and immediately following each condition (Figure 4.2 box 6). This gives us three snapshots of the user's mood scale which allows us to measure any changes in their emotion after using the two interfaces.

Prior to running any prototyping tasks, the average positive affect score was 27.30 (SD = 10.985) and the average negative affect score was 11.40 (SD = 1.713). After running the StoryTIME condition the average positive affect score was 29.50 (SD = 11.174) and the negative affect score was 10.50 (SD = 0.707). After running the Unity condition the positive affect score was 24.9 (SD = 11.174) and the negative affect score was 11.20 (SD = 1.229). Figure 4.6 summarizes the results from the



(a)

Descriptive Statistics - PANAS

	N	Minimum	Maximum	Mean	Std. Deviation
Initial Positive Affect	10	12	50	27.30	10.985
Initial Negative Affect	10	10	15	11.40	1.713
StoryTIME Positive Affect	10	14	50	29.50	11.178
StoryTIME Negative Affect	10	10	12	10.50	.707
Unity Positive Affect	10	12	50	24.90	11.714
Unity Negative Affect	10	10	13	11.20	1.229
Valid N (listwise)	10				

(b)

Figure 4.6: PANAS descriptive statistics.

PANAS questionnaire. A paired sample t-test was performed to determine if there was any significance between the positive and negative affects for the two conditions. There was no significant difference between the negative affect score for Unity (M = 11.20, SD = 1.229) and StoryTIME (M = 10.50, SD = 0.707); t(9) = -1.481, p = 0.173. There was a significant difference between the positive affect score for Unity (M = 24.9, SD = 11.714) and StoryTIME (M = 29.50, SD = 11.179); t(9) = 2.379, p = 0.041.

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Pre-Condition Positive Affect	27.30	10	10.985	3.474
	StoryTIME Positive Affect	29.50	10	11.178	3.535
Pair 2	Pre-Condition Negative Affect	11.40	10	1.713	.542
	StoryTIME Negative Affect	10.50	10	.707	.224
Pair 3	Pre-Condition Positive Affect	27.30	10	10.985	3.474
	Unity Positive Affect	24.90	10	11.714	3.704
Pair 4	Pre-Condition Negative Affect	11.40	10	1.713	.542
	Unity Negative Affect	11.20	10	1.229	.389
Pair 5	StoryTIME Positive Affect	29.50	10	11.178	3.535
	Unity Positive Affect	24.90	10	11.714	3.704
Pair 6	StoryTIME Negative Affect	10.50	10	.707	.224
	Unity Negative Affect	11.20	10	1.229	.389

Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	Pre-Condition Positive Affect & StoryTIME Positive Affect	10	.905	.000
Pair 2	Pre-Condition Negative Affect & StoryTIME Negative Affect	10	-.275	.441
Pair 3	Pre-Condition Positive Affect & Unity Positive Affect	10	.889	.001
Pair 4	Pre-Condition Negative Affect & Unity Negative Affect	10	-.253	.480
Pair 5	StoryTIME Positive Affect & Unity Positive Affect	10	.858	.001
Pair 6	StoryTIME Negative Affect & Unity Negative Affect	10	-.128	.725

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	Pre-Condition Positive Affect - StoryTIME Positive Affect	-2.200	4.826	1.526	-5.652	1.252	-1.442	9	.183
Pair 2	Pre-Condition Negative Affect - StoryTIME Negative Affect	.900	2.025	.640	-.548	2.348	1.406	9	.193
Pair 3	Pre-Condition Positive Affect - Unity Positive Affect	2.400	5.400	1.707	-1.463	6.263	1.406	9	.193
Pair 4	Pre-Condition Negative Affect - Unity Negative Affect	.200	2.348	.742	-1.479	1.879	.269	9	.794
Pair 5	StoryTIME Positive Affect - Unity Positive Affect	4.600	6.114	1.933	.226	8.974	2.379	9	.041
Pair 6	StoryTIME Negative Affect - Unity Negative Affect	-.700	1.494	.473	-1.769	.369	-1.481	9	.173

Figure 4.7: PANAS paired-sample t-Test result

4.4.3 The System Usability Scale (SUS)

The System Usability Scale (SUS) [57] is a reliable questionnaire used for measuring system usability. The SUS is comprised of 10 questions in the form of a five point Likert scale, where a response of 1 corresponds to "strongly disagree" and a response of 5 corresponds to "strongly agree". The questionnaire was administered twice as shown in Figure 4.2 box 6. To obtain the final SUS score for a system, we take the average of all participant SUS scores. The SUS score for Unity was 58 (SD = 17.83) and the SUS score for StoryTIME was 79.75 (SD = 13.36), this suggests StoryTIME has better usability traits than Unity. A summary of the SUS responses can be seen in Figure 4.8. Items 1,3,5,7 and 9 measure positive traits of the system, a higher response value for these questions are in the favor of the system being evaluated. Items 2,4,6,8 and 10 measure negative traits of the system, a higher response value for these questions are not in the favor of the system being evaluated.

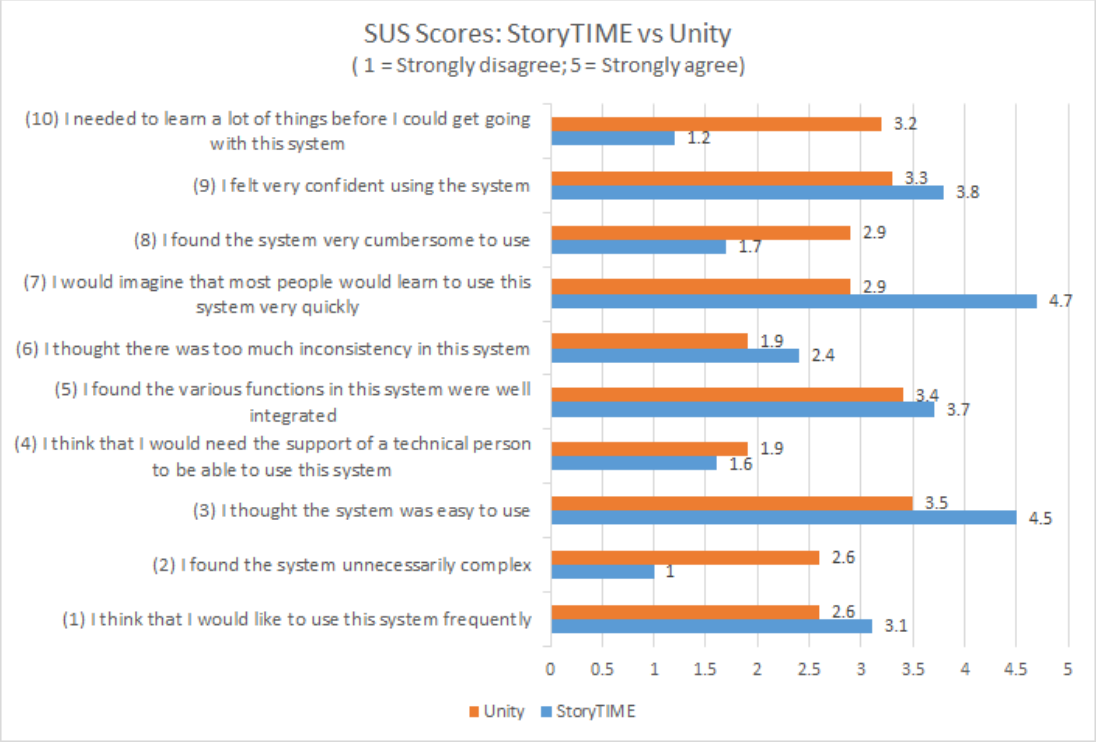


Figure 4.8: Summary of SUS scores

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	SUS Score StoryTIME	79.750	10	13.3568	4.2238
	SUS Score Unity	58.000	10	17.8263	5.6372

Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	SUS Score StoryTIME & SUS Score Unity	10	.079	.828

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	SUS Score StoryTIME - SUS Score Unity	21.7500	21.4103	6.7705	6.4340	37.0660	3.212	9	.011

Figure 4.9: Paired samples T-test for SUS scores

A paired sample T-test was performed to determine if there was any statistical significance between the Unity and StoryTIME SUS scores. The T-test was used because the study followed a within subject design. The results of the T-test are summarized in Figure 4.9. There was a significant difference between the SUS scores for Unity (M = 58, SD = 17.83) and StoryTIME (M = 79.750, SD = 13.36); $t(9) = 3.212$, $p = 0.011$.

4.4.4 Post Activity Questionnaire

The post activity questionnaire inquired about the participant's preference and personal perception of the their ability to complete the tasks. The responses are summarized in Figure 4.10. Participants reported that the tracking markers used on the physical props got in the way (M = 4.1, SD = 2.13). This was expected as this is one of the biggest drawbacks to fiduciary AR markers. Participants reported that they would see themselves using a TUI to create prototypes (M = 5.2, SD = 0.79). Interestingly participants were generally neutral when asked about their preferences using StoryTIME with projections enabled. They reported that the system felt faster when projections were disabled (M = 3.2, SD = 2.25) and they mentioned that prototyping felt more accurate when projections were disabled (M = 3.7, SD = 1.89). Both of

these results are likely attributed to tracking limitations. Slowness could be a result of latency induced by the Kalman filter to reduce jitter.

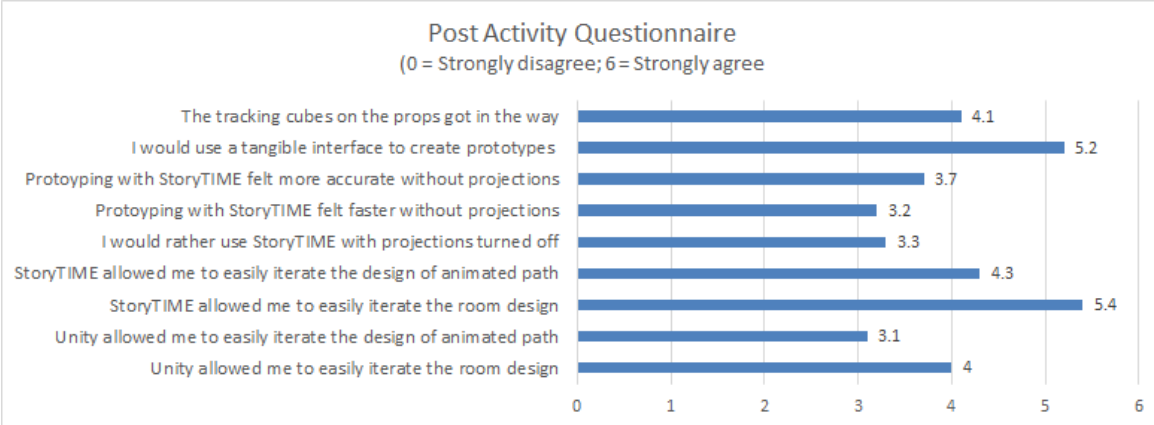


Figure 4.10: Summary of post activity questionnaire responses.

4.5 Discussion

PANAS showed that there was no negative impact to the participant's emotions after running either conditions. This is a good thing as we did not want to have a negative impact on the participant's emotional state. This is always a concern when introducing a new system as no UI designer wants to build a system that is frustrating to use. There was significance found between the positive affect scores for Unity and StoryTIME which indicates that participants enjoyed using the StoryTIME system over unity. These results suggest that StoryTIME is on the right track in terms of user experience.

The SUS scores indicate that StoryTIME ($M = 79.75$, $SD = 13.36$) has better usability characteristics compared to Unity ($M = 58$, $SD = 17.83$). The results from the paired-sample t-test shows that there is a significant difference between the two systems ($p = 0.011$, $p < 0.05$). Consider that modern WIMP based interfaces such as Unity are the product of decades of refinement and iteration. The fact that StoryTIME is able to demonstrate comparable usability scores to an established sys-

tem validates the potential for TUIs based on natural interactions. Below we discuss each of the items in the SUS questionnaire.

- **Q1 - I think that I would like to use this system frequently:** Participants reported that they would like to use both of these systems frequently. Unity received an average score of 2.6 (SD = 1.35) and StoryTIME received an average score of 3.1 (SD = 1.29). This response is in compliance with the demographics questionnaire as all participants work in related disciplines. The higher StoryTIME score could be attributed to user preference.

- **Q2 - I found the system unnecessarily complex:** Participants reported that they found StoryTIME (M = 1, SD = 0) to be less complex than Unity (M = 2.6, SD = 1.17). All 10 participants unanimously agreed that StoryTIME was not complicated to use, this is a great result as it shows that StoryTIME succeeded in its goal to simplify the process of prototyping animated sequences.

- **Q3 - I thought the system was easy to use:** Participants reported that they found both Unity (M = 3.5, SD = 1.27) and StoryTIME (M = 4.5, SD = 0.71) easy to use. The Unity score is likely influenced by prior experience, it would be interesting to see how a participant with no Unity experience would have scored this question.

- **Q4 - I think that I would need the support of a technical person to be able to use this system:** Participants reported that they do not feel that they would need technical support to use either StoryTIME (M = 1.6, SD = 0.97) or Unity (M = 1.9, SD = 1.2). This affirms StoryTIME's commitment to ease of use as this score indicates the participants were confident in using the system. The Unity score could be attributed to the participants' prior experience Unity.

- **Q5 - I found the various functions in this system were well integrated:** Participants found that features in both Unity (M = 3.4, SD = 0.97)

and StoryTIME (M = 3.7, SD = 0.95). Unity has a much larger feature set than StoryTIME, features in this context would pertain to the specific tasks performed in this study which were to placing and animating virtual objects.

- **Q6 - I thought there was too much inconsistency in this system:** Participants reported that they found StoryTIME (M = 2.4, SD = 1.43) to be more inconsistent than Unity (M = 1.9, SD = 0.99). This is likely due errors in the object tracking, as jitter is quite noticeable when you are trying to capture motion.

- **Q7 - I would imagine that most people would learn to use this system very quickly:** Participants reported that new users of both StoryTIME (M = 4.7, SD = 0.48) and Unity (M = 2.9, SD = 1.2) would be able to figure out how to use the systems quickly.

- **Q8 - I found the system very cumbersome to use:** Participants reported that Unity (M = 2.9, SD = 1.45) was more cumbersome to use than StoryTIME (M = 1.7, SD = 1.06). This response is likely due to the difficultly in editing previously placed key-frames in Unity.

- **Q9 - I felt very confident using the system:** Participants felt confident using both Unity (M = 3.3, SD = 0.95) and StoryTIME (M = 3.8, SD = 1.23). It was expected that they would feel confident using Unity, as per their prior experience but to see StoryTIME receive a higher score was unexpected. This may be attributed to the low technical forethought needed to use StoryTIME.

- **Q10 - I needed to learn a lot of things before I could get going with this system:** Participants reported that they did not have to learn much to be productive with StoryTIME (M = 1.2, SD = 0.42). With Unity (M = 3.2, SD = 1.4) they reported that they needed to learn quite a bit. The phrasing of this

question may have lead to some ambiguity, as it does not account for previous experience with the system.

Participants responded neutrally when asked about their preference between projected augmentations and traditional augmentations displayed on a monitor. This was an unexpected result as it was anticipated that the projected AR would be the display medium of choice. There are two possible explanations for this: tracking error and latency. There is always some **tracking error** when the augmentations are projected onto the physical objects, the projections never line up absolutely perfectly. The best we can do is minimize this error to ensure that the misalignment is at a minimum. Errors are introduced at every step in the transformation pipeline described in Section 3.4, the projector-camera calibration, object tracker and marker to object transformations are all calculated using some form of a linear equation solver. As this error accumulates, the projections become more offset from the physical object. For the projector-camera calibration and marker to object transformations, these errors can be minimized quite a bit as these components are calculated offline and do not change on a frame-by-frame basis. The object tracker however is very susceptible to errors and there is not much that can be done about it. Jitter is an artifact where the tracked position "jumps" around between frames. Jitter can result in the user doubting the accuracy of the system as there is a disconnect between the physical and virtual objects. When the purpose of the system is to position objects and record their motion, this is a big problem. Jitter can be minimized by using techniques such as a Kalman filter which work by taking a running average of previous frames but this comes at the cost of introducing latency into the system. **Latency** is the delay between the user performing an action and the system providing some kind of reaction. When participants move the physical props, and there is a delay between the update in the projection, this can make the system feel slow. It can be argued

that when projections are displayed on a monitor, the effects of jitter and latency are less noticeable.

Chapter 5

Conclusions

5.1 Contributions

The original goal of this work was to address the challenges in developing prototypes for animated sequences. This thesis presented the development and evaluation of StoryTIME, a system which leverages tangible user interfaces and augmented reality with the intention of simplifying the process of prototyping animated sequences by replacing traditional key-framing tasks with natural movements. We have found that prototyping with tangibles is as effective as using a traditional keyboard and mouse system in terms of usability with the additional benefit of improved user experience. Iteration times were generally faster when using the TUI. In terms of user preference between traditional AR displayed on a monitor versus projected we found that users were generally neutral.

Below we present the answers to the research questions introduced in the first chapter:

- **Does prototyping with tangibles improve iteration time when developing pre-visualizations?** We found that prototyping with the TUI was generally faster, however it was noted that faster did not necessarily mean bet-

ter. For example, participants experienced with animating in Unity spent longer tweaking their animations by manipulating the animation curves. On one hand we can say that with StoryTIME it would not be necessary to manually tweak animations to implement "ease-in-ease-out" style transitions, as the user would just record the motion as they see fit. On the other hand we could say that the precision of using Unity to fine tune curves would allow users to create motions they would not be able to replicate with a physical prop.

- **How does prototyping pre-visualizations with tangibles compare with traditional keyboard and mouse based tools in terms of usability and user experience?** We found that there was a significant difference between the usability of StoryTIME and Unity and that participants favored StoryTIME for its simplicity and ease of use. This makes a case for the concept of the TIME suite of TUIs (introduced in Chapter 1), for domain specific tasks.

- **How does displaying augmentations on a computer monitor compare with projected augmentation in terms of user preference and performance?** We found that participants were neutral with their preference for the display medium for the virtual augmentations.

5.2 Limitations and Future Work

The biggest limitation in the current implementation of StoryTIME is in the tracking technology. The system currently uses an optic based solution which relies on AR markers for tracking and detection. This method was chosen because it provides fast and reliable tracking however there are drawbacks. Most notability is the impact of the AR markers on the usability of this system, as noted by participant responses discussed in Chapter 4. The issue is that the markers got in the way while users were trying to create their scenes, resulting in users having to alter their anticipated move-

ments to account for the AR marker. Future iterations of StoryTIME should have a deeper focus on tracking technology. As mentioned in Chapter 3, the architecture of StoryTIME is designed such that the implementation of individual components, such as the object tracker, could be replaced without having any impact on other components in the system. The current implementation of StoryTIME relies solely on optic based tracking, a potential area to explore is tracking solutions which utilize integrated sensors such as accelerometers and gyroscopes in addition to vision based tracking. This would require the development of a small integrated circuit to minimize any potential obstruction. It is also a possibility that advances in real-time point cloud tracking suggest that one day the need for AR markers and other tracking aids could be completely removed.

The next step for StoryTIME would be to add additional features to the system, such as the ability of speech recording for audio. We could also investigate alternate use cases for the technology developed. Currently we only investigate the case of story board development, but other potential uses could be military or tactical planning, entertainment or post-traumatic stress therapy. The StoryTIME technology could essentially be used for any application which involves placing objects and moving them around.

References

[1] A. Beane, *3D Animation Essentials*. Indianapolis, IN: Wiley, 2012.

[2] I. Failes, "The rise of the previs of Planet of the Apes," 2011. www.fxguide.com/featured/rise-of-the-previs/, Retrieved January 31 2018.

[3] "Pokemongo!." [Computer Software], 2016. https://www.pokemongo.com/, Retrieved January 31 2018.

[4] "Microsoft Hololens." [Computer hardware], 2018. www.microsoft.com/en-us/hololens.

[5] R. Raskar, M. Cutts, G. Welch, and W. Stuerzlinger, "Efficient image generation for multiprojector and multisurface displays," *Rendering techniques' 98: proceedings of the Eurographics Workshop in Vienna, Austria, June 29-July 1, 1998*, p. 139, 1998.

[6] N. Beiman, *Prepare to Board! Creating Story and Characters for Animation Features and Shorts*. Focal Press, second ed., 2012.

[7] T. Caudell and D. Mizell, "Augmented reality: an application of heads-up display technology to manual manufacturing processes," *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, no. February, pp. 659–669, 1992.

[8] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[9] R. Parent, *Computer Animation: Algorithms and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 3 ed., 2012.

[10] R. Wyatt, "Rise of the Planet of the Apes." [Movie] 20th Century Fox Home Entertainment, 2011.

[11] D. S. Buckstein, "Playing is creating with PlayTIME: introducing and evaluating a tangible UI-based interactive scenario prototyping system," 2015.

[12] O. Shaer, "Tangible User Interfaces: Past, Present, and Future Directions," *Foundations and Trends® in Human–Computer Interaction*, vol. 3, no. 1-2, pp. 1–137, 2009.

[13] G. Fitzmaurice, H. Ishii, and W. Buxton, "Bricks: laying the foundations for graspable user interfaces," *SIGCHI Conference on Human Factors in Computing Systems*, p. 442–449, 1995.

[14] H. Ishii, "Tangible bits," *Proceedings of the 8th international conference on Intelligent user interfaces - IUI '03*, p. 3, 2003.

[15] J. Nielsen, "10 Usability Heuristics for User Interface Design," 1995.

[16] G. Kipper and J. Rampolla, *Augmented Reality: An Emerging Technologies Guide to AR*. 2012.

[17] J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani, and M. Ivkovic, "Augmented reality technologies, systems and applications," *Multimedia Tools and Applications*, vol. 51, no. 1, pp. 341–377, 2011.

[18] E. Costanza, A. Kunz, and M. Fjeld, "Mixed reality: A survey," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5440 LNCS, pp. 47–68, 2009.

[19] "Google Glass." [Computer hardware], 2018. https://x.company/glass/partners/.

[20] R. Raskar, K. Low, and G. Welch, "Shader lamps: Animating real objects with image-based illumination," *University of North Carolina at Chapel Hill*, 2000.

[21] B. Jones, L. Shapira, R. Sodhi, M. Murdock, R. Mehra, H. Benko, A. Wilson, E. Ofek, B. MacIntyre, and N. Raghuvanshi, "RoomAlive," *Proceedings of the 27th annual ACM symposium on User interface software and technology - UIST '14*, pp. 637–644, 2014.

[22] K. McElroy, *Prototyping for Designers: Developing the Best Digital and Physical Products*. O'Reilly Media Inc., 2017.

[23] J. Arnowitz, M. Arent, and N. Berger, *Effective Prototyping For Software Makers*. Interactive Technologies, Elsevier Science, 2007.

[24] L. Blazer, *Animated Storytelling: Simple Steps For Creating Animation and Motion Graphics*. Peachpit Press, 2015.

[25] E. Ghertner, *Layout and Composition for Animation*. Focal Press/Elsevier, 2012.

[26] A. Jew, *Professional Storyboarding*. CRC Press, 2013.

[27] M. J. Kim and M. L. Maher, "The Impact of Tangible User Interfaces on Designers' Spatial Cognition," *Human-Computer Interaction*, vol. 23, pp. 101–137, 2008.

[28] K. Lim and B. Binotti, "PhysLights : a Tangible User Interface for CG Lighting," 2014.

[29] A. Alves, R. Lopes, P. Matos, L. Velho, and D. Silva, "Reactoon: Storytelling in a tangible environment," *2010 Third IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning*, pp. 161–165, 2010.

[30] Y.-P. Ma, C.-H. Lee, and T. Jeng, "iNavigator: a Spatially-Aware Tangible Interface for Interactive 3D Visualization," *Proceedings of the 8th International Conference on Computer-Aided Architectural Design Research in Asia*, pp. 963–974, 2003.

[31] H. Ishii, C. Ratti, B. Piper, Y. Wang, A. Biderman, and E. Ben-Joseph, "Bringing Clay and Sand into Digital Design - Continuous Tangible user Interfaces," *BT Technology Journal*, vol. 22, no. 4, pp. 287–299, 2004.

[32] M. Reed, "Prototyping digital clay as an active material," *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction TEI 09*, p. 339, 2009.

[33] "Boeing," 2016. www.boeing.com.

[34] S. You, U. Neumann, and R. Azuma, "Hybrid inertial and vision tracking for augmented reality registration," *Proceedings IEEE Virtual Reality (Cat. No. 99CB36316)*, pp. 260–267, 1999.

[35] R. Azuma and R. Azuma, "A survey of augmented reality," *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 355–385, 1997.

[36] M. V. Wyawahare, P. M. Patil, and H. K. Abhyankar, "Image Registration Techniques : An overview," *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 2, no. 3, pp. 11–28, 2009.

[37] I. E. Sutherland, "A head-mounted three dimensional display," *Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS '68 (Fall, part I)*, p. 757, 1968.

[38] N. Villar, D. Cletheroe, G. Saul, C. Holz, O. Salandin, T. Regan, M. Sra, H.-S. Yeo, W. Field, and H. Zhang, "Project Zanzibar: A Portable and Flexible Tangible Interaction Platform," 2018.

[39] D. Schmalstieg and T. Höllerer, "Augmented reality: Principles and Practice," in *Proceedings - IEEE Virtual Reality*, 2017.

[40] Oculus VR, "Oculus Rift." [Computer hardware]. oculusvr.com.

[41] Skype, "Skype for Business," *Skype for Business*, no. August, p. 1, 2016.

[42] S. Fanello, S. O.-e. C. Rhemann, M. Dou, V. Tankovich, C. Loop, and P. Chou, "Holoportation : Virtual 3D Teleportation in Real-time," *Chi*, pp. 741–754, 2016.

[43] E. Akaoka, T. Ginn, R. Vertegaal, and O. N. Kl, "DisplayObjects : Prototyping Functional Physical Interfaces on 3D Styrofoam , Paper or Cardboard Models," pp. 49–56.

[44] B. R. Jones, H. Benko, E. Ofek, and A. D. Wilson, "IllumiRoom: Peripheral Projected Illusions for Interactive Experiences," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*, p. 869, 2013.

[45] K. D. D. Willis, I. Poupyrev, S. E. Hudson, and M. Mahler, "SideBySide : Ad-hoc Multi-user Interaction with Handheld Projectors," *In Proceedings of UIST '11*, pp. 431–440, 2011.

[46] A. Nijholt, *Playful User Interfaces: Interfaces that Invite Social and Physical Interaction*. Gaming Media and Social Effects, Springer Singapore, 2014.

[47] Q. Bonnard, S. Lemaignan, G. Zufferey, A. Mazzei, S. Cuendet, N. Li, A. Ozgur, and P. Dillenbourg, "Chilitags: Robust Fiducial Markers for Augmented Reality," 2013.

[48] "Intel realsense." [Computer Software], 2018. https://realsense.intel.com/, Last accessed January 31 2018.

[49] ARToolkit, "ARToolKit Documentation," 2015.

[50] Qualcomm Technologies, Inc., "Vuforia Augmented Reality SDK." [Computer software], 2015.

[51] M. Kaltenbrunner and R. Bencina, "reacTIVision: a computer-vision framework for table-based tangible interaction," *Proceedings of the 1st international conference on Tangible and embedded interaction*, pp. 69–74, 2007.

[52] U. Neumann and S. You, "Natural feature tracking for augmented reality," *IEEE Transactions on Multimedia*, vol. 1, no. 1, pp. 53–64, 1999.

[53] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. New York, NY, USA: Cambridge University Press, 2 ed., 2003.

[54] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[55] T. Akenine-Moller, E. Haines, and N. Hoffman, *Real-Time Rendering*. Natick, MA, USA: A. K. Peters, Ltd., 3rd ed., 2008.

[56] D. Watson, L. A. Clark, and A. Tellegen, "Development and Validation of Brief Measures of Positive and Negative Affect: The PANAS Scales," *Journal of Personality and Social Psychology*, vol. 54, no. 6, pp. 1063–1070, 1988.

[57] J. Brooke, "SUS - A quick and dirty usability scale," pp. 1–8, 1986.

Appendix A

HP Sprout Hardware Details

- 20" (diagonal) touch enabled surface (referred to as the interaction area in this work)
- Intel Core i7-4790S CPU
- NVIDIA GeForce GT 745A GPU with 2GB of VRAM
- 23" touch compatible display with a resolution of 1920x1080
- 1024x768 projector
- Intel F200 RealSense camera
- 8GB of RAM
- 1TB hard drive

Appendix B

Surveys, Questionnaires and Ethics

This appendix contains all information pertaining to the surveys and questionnaires administered as well as the ethics approval for the user study conducted as a part of the evaluation of StoryTIME discussed in Chapter 4.

B.1 Consent Form

The consent form was administered to participants immediately upon welcoming them into the room and briefing them on the purpose of the study and the tasks they will be completing.

Title of Research Study: Evaluation of StoryTIME

You are invited to participate in a research study entitled Evaluation of StoryTIME. This study has been reviewed the University of Ontario Institute of Technology Research Ethics Board **[REB # 14438]** and originally approved on **6/6/2017.**
Please read this consent form carefully, and feel free to ask the Researcher any questions that you might have about the study. If you have any questions about your rights as a participant in this study, please contact the Ethics and Compliance Coordinator at 905 721 8668 ext. 3693 or researchethics@uoit.ca.

Researcher(s):
Michael Gharbharan, Faculty of Business and IT, Michael.Gharbharan@uoit.ca
Dr. Andrew Hogue, Faculty of Business and IT (Ext. 3698), Andrew.Hogue@uoit.ca

Purpose and Procedure:
The purpose of this study is to evaluate StoryTIME (a Tangible Interactive Media Environment for Storytelling). StoryTIME is a system that combines augmented reality with tangible interfaces that allows users to rapidly prototype story sequences using physical objects.

This study is split into two phases: (1) You will be given a paper prototype of an animated sequence and will be asked to create a digital prototype using a traditional tool (Unity3D) and an experimental tool (StoryTIME). (2) You will be asked to perform prototyping tasks using StoryTIME with digital projections enabled and disabled.

This study will take approximately 30 to 60 minutes. Upon completion of the prototyping tasks, you will be asked to complete a questionnaire to gather your feedback on the tools used. The data collected is anonymized and will be used in a statistical analysis to evaluate the StoryTIME system. The results of the statistical analysis may be published in journal or conference proceedings and may be used in future studies as secondary data.

Potential Benefits:
You will not benefit directly from participating in this study.

Potential Risk or Discomforts:
There are no risks involved with this project. You may withdraw without consequence if you feel any discomfort, see the "Right to Withdraw" section below for details on withdrawal.

Storage of Data and Confidentiality:
All data collected during this experiment is anonymized, at no point will any personally identifying data be collected. Data will be collected in the form of questionnaires and screen recording during the prototyping tasks. Collected data will be stored in a password protected cloud storage account which will only be accessible by Michael Gharbharan.

Right to Withdraw:
Your participation is voluntary, and you can answer only those questions that you are comfortable with. The information that is shared will be held in strict confidence and discussed only with the research team. You may withdraw at any time without needing to specify a reason. There are no consequences for withdrawing. If you withdraw from the research project at any time, any data collected will be destroyed immediately.

Compensation:
There will be no compensation for participation.

Debriefing and Dissemination of Results:
You have the option to be informed of the results of the research. If you are interested in the results of the study, you may leave your email address below. Results will be emailed to you once they are compiled (usually within 2 - 3 months).

Participant Concerns and Reporting:
If you have any questions concerning the research study or experience any discomfort related to the study, please contact the researcher Michael Gharbharan at michael.gharbharan@uoit.ca.
Any questions regarding your rights as a participant, complaints or adverse events may be addressed to Research Ethics Board through the Research Ethics Coordinator – researchethics@uoit.ca or 905.721.8668 x. 3693.

By consenting, you do not waive any rights to legal recourse in the event of research-related harm.

Consent to Participate:

1. I have read the consent form and understand the study being described;
2. I have had an opportunity to ask questions and my questions have been answered. I am free to ask questions about the study in the future;
3. I freely consent to participate in the research study, understanding that I may discontinue participation at any time without penalty. A copy of this Consent Form has been made available to me.

_____ _____
(Name of Participant) (Date)

_____ _____
(Signature of Participant)/ (Signature of Researcher)

If you would like to be informed of the results of the study, please print an email address below. **This is optional.**

_____
(Email Address)

B.2 Demographics Survey

Demographics

* Required

1. **Enter Unique Participant ID** *

_____

2. **Task order** *

Mark only one oval.

- ◯ Unity First
- ◯ StoryTIME first

Demographics

The purpose of this demographics questionnaire is to contextualize the data obtained during this session.

3. **Please indicate your gender**

Mark only one oval.

- ◯ Male
- ◯ Female
- ◯ Prefer not to say
- ◯ Other: _____

4. **Please indicate your age**

_____

5. **Have you ever created any kind of virtual environment? (This includes everything from using an in-game level editor to using a professional tool such as Maya)** *

Mark only one oval.

- ◯ Yes
- ◯ No

Demographics

6. **How frequently do you use following tools to create a virtual environment**

Mark only one oval per row.

	Never	Rarely (once every few months)	Sometimes (a few times a month)	Very Often (at least once a week)	Always (almost daily)
Unity3D	◯	◯	◯	◯	◯
Unreal Engine	◯	◯	◯	◯	◯
Maya	◯	◯	◯	◯	◯
Blender	◯	◯	◯	◯	◯

7. **How many hours per week do you spend developing and or modifying virtual environments in an average week?**

Mark only one oval.

- ◯ Less than 5
- ◯ 5 - 10
- ◯ 11 - 15
- ◯ 16 - 20
- ◯ 21 - 25
- ◯ 26 - 30
- ◯ More than 30

8. **Which best describes your experience with creating virtual environments**

Mark only one oval.

- ◯ I have never created a virtual environment
- ◯ I have followed a few tutorials
- ◯ I have created virtual environments for 1-2 projects
- ◯ I have created virtual environments for 3-5 projects
- ◯ I have created virtual environments for too many projects to count

9. **How frequently have you use the following prototyping techniques when creating a virtual environment**
Mark only one oval per row.

	Never	Rarely	Sometimes	Very Often	Always
Write ideas on paper (i.e. a list of things that are needed in the environment)	◯	◯	◯	◯	◯
Create a physical mock-up (i.e. a diorama)	◯	◯	◯	◯	◯
Create a digital mock-up using primitive shapes	◯	◯	◯	◯	◯
Modify an existing environment (i.e. building off of a sample scene)	◯	◯	◯	◯	◯
Use a tangible prototyping tool	◯	◯	◯	◯	◯
Use a VR level editor (i.e. Unreal Engine in VR)	◯	◯	◯	◯	◯

10. **If you have used any prototyping techniques which were not listed above please describe them here**

_____

_____

_____

_____

_____

Demographics

11. **Have you ever created any kind of key framed animated sequence?** *
Mark only one oval.

◯ Yes *Skip to question 12.*

◯ No *Stop filling out this form.*

Demographics

12. **How frequently do you use following tools to create a key framed animated sequence**

Mark only one oval per row.

	Never	Rarely (once every few months)	Sometimes (a few times a month)	Very Often (at least once a week)	Always (almost daily)
Unity3D	◯	◯	◯	◯	◯
Unreal Engine	◯	◯	◯	◯	◯
Maya	◯	◯	◯	◯	◯
Blender	◯	◯	◯	◯	◯

13. **How many hours per week do you spend developing and or modifying key framed animated sequences in an average week?**

Mark only one oval.

- ◯ Less than 5
- ◯ 5 - 10
- ◯ 11 - 15
- ◯ 16 - 20
- ◯ 21 - 25
- ◯ 26 - 30
- ◯ More than 30

14. **Which best describes your experience with creating key framed animated sequences**

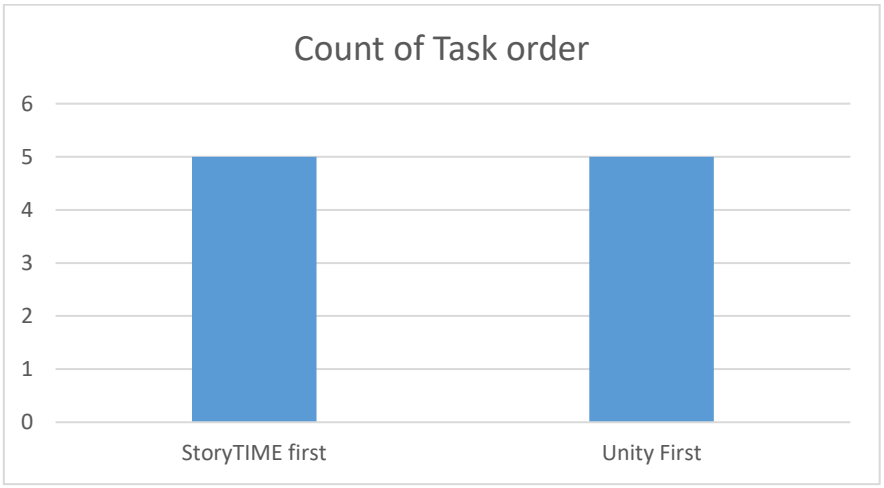
Mark only one oval.

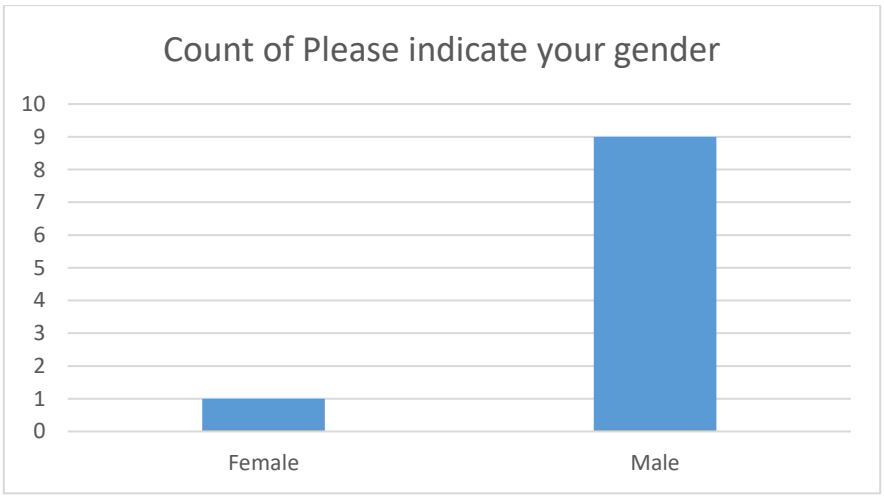
- ◯ I have never created a virtual environment
- ◯ I have followed a few tutorials
- ◯ I have created virtual environments for 1-2 projects
- ◯ I have created virtual environments for 3-5 projects
- ◯ I have created virtual environments for too many projects to count

15. **How frequently have you used the following prototyping techniques when creating a key framed animated sequence?**
Mark only one oval per row.

	Never	Rarely	Sometimes	Very Often	Always
Write ideas on paper (i.e. a list of events which need to happen)					
Create a physical mock-up (i.e. a diorama)					
Create a digital mock-up using primitive shapes					
Modify an existing key framed sequence (i.e. building off of a sample scene)					
Use a tangible prototyping tool					
Use a VR level editor (i.e. Unreal Engine in VR)					

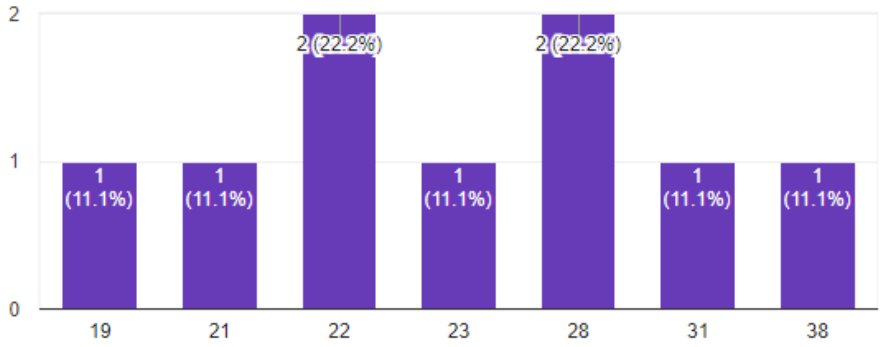
16. **If you have used any prototyping techniques which were not listed above please describe them here**


Count of Task order
6
5
4
3
2
1
0
StoryTIME first
Unity First


Count of Please indicate your gender
10
9
8
7
6
5
4
3
2
1
0
Female
Male

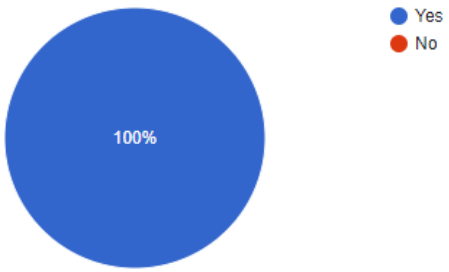
Please indicate your age

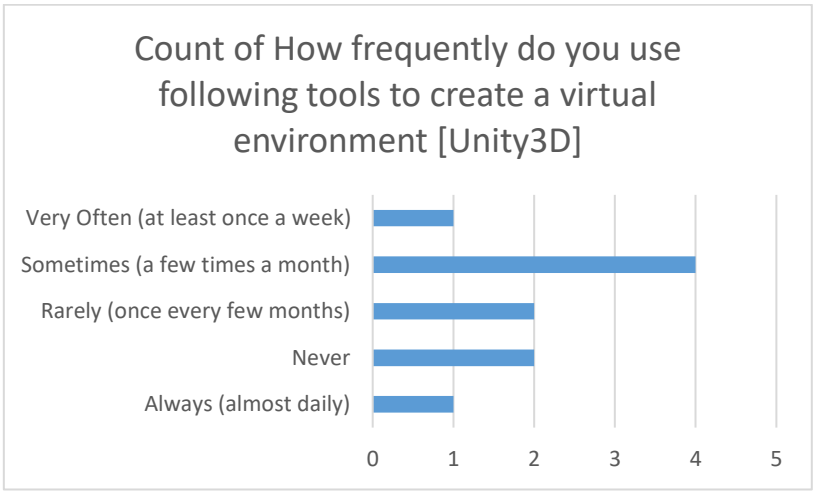
9 responses

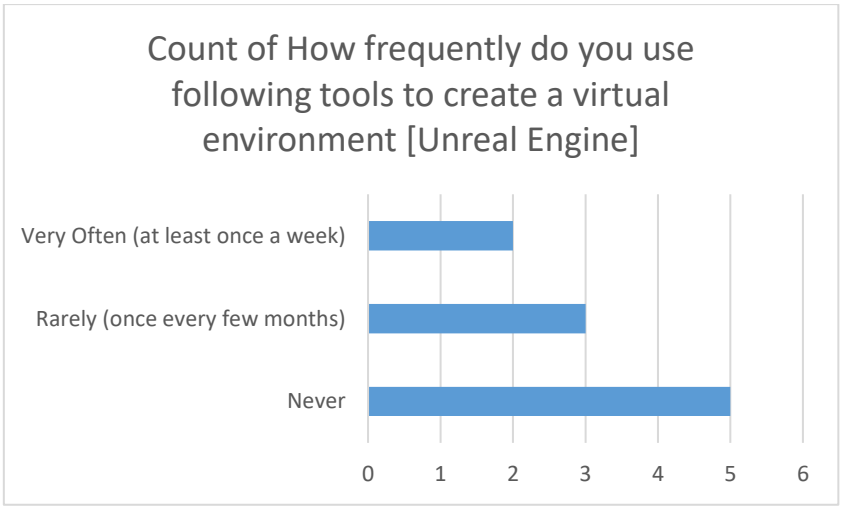


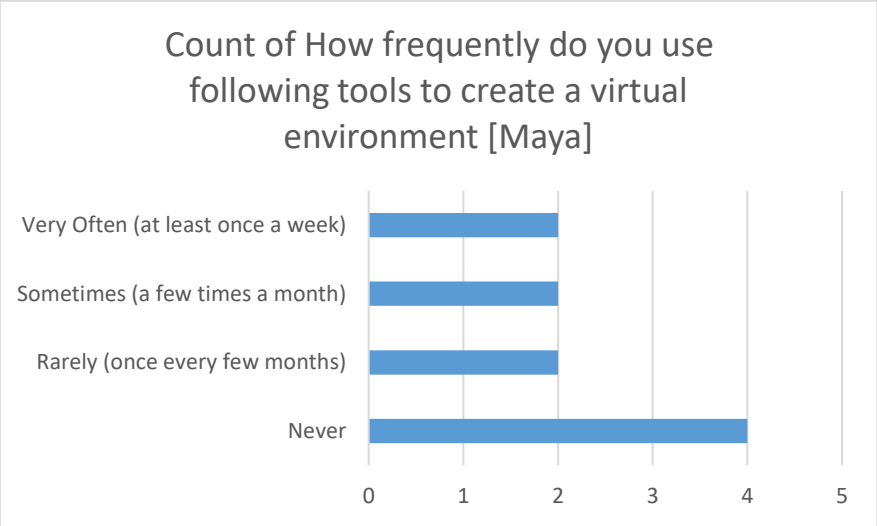
Have you ever created any kind of virtual environment? (This includes everything from using an in-game level editor to using a professional tool such as Maya)

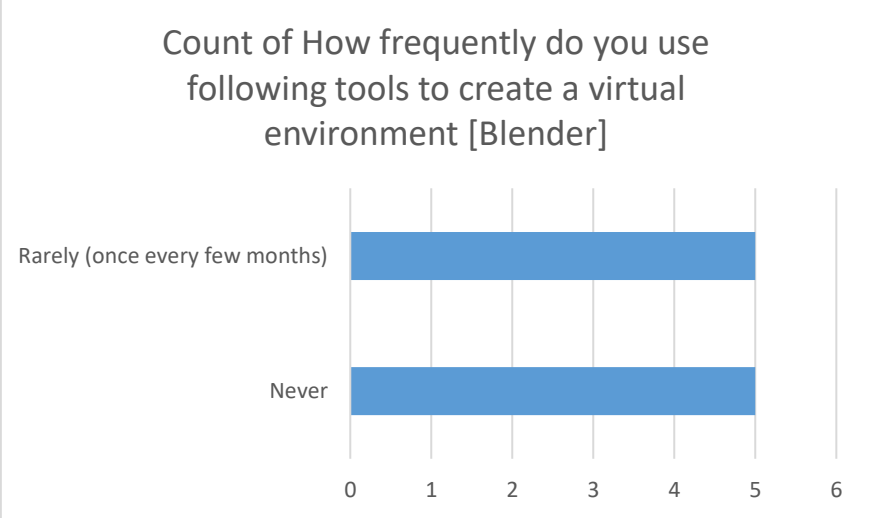
10 responses

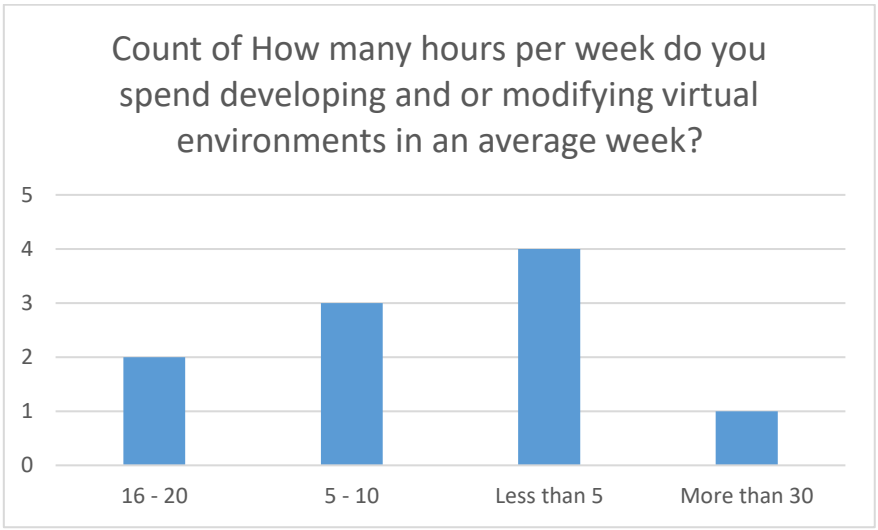


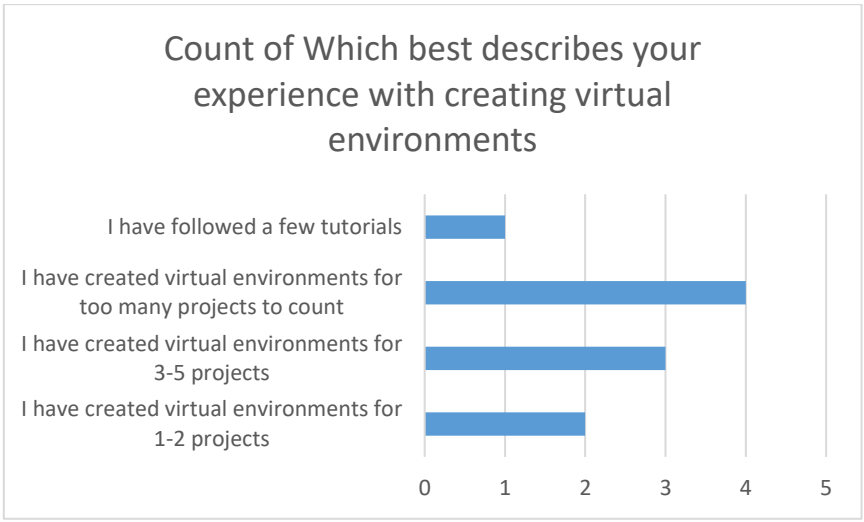

Count of How frequently do you use following tools to create a virtual environment [Unity3D]
Very Often (at least once a week)
Sometimes (a few times a month)
Rarely (once every few months)
Never
Always (almost daily)
0
1
2
3
4
5

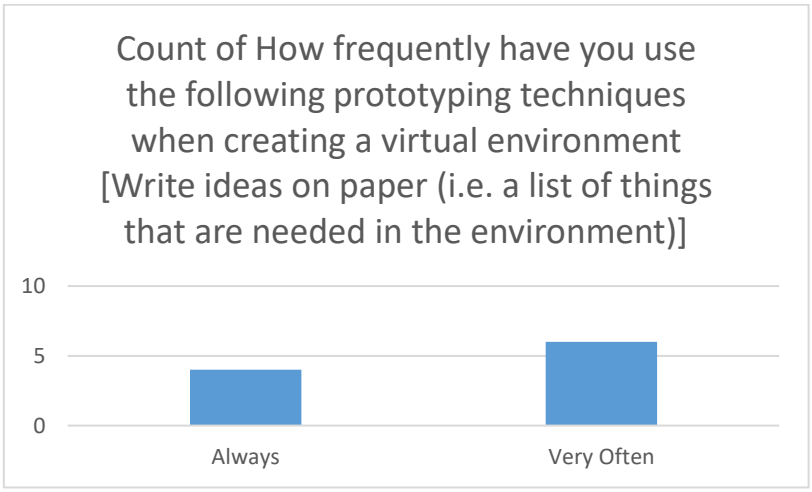

Count of How frequently do you use following tools to create a virtual environment [Unreal Engine]
Very Often (at least once a week)
Rarely (once every few months)
Never
0
1
2
3
4
5
6

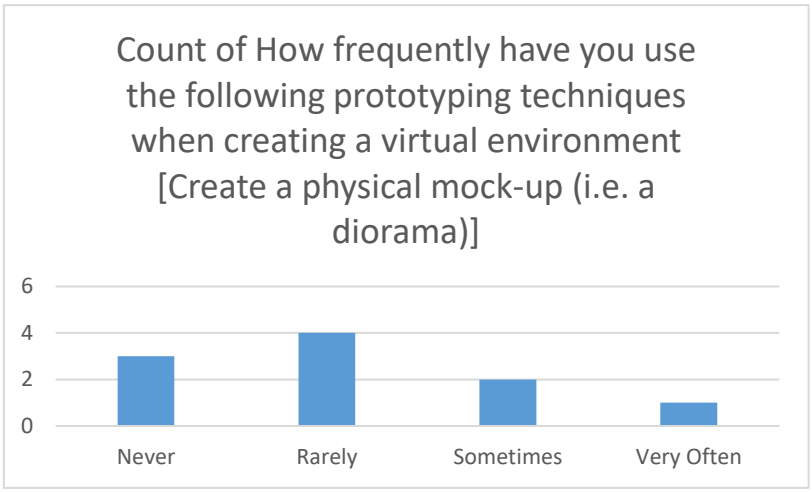

Count of How frequently do you use following tools to create a virtual environment [Maya]
Very Often (at least once a week)
Sometimes (a few times a month)
Rarely (once every few months)
Never
0
1
2
3
4
5

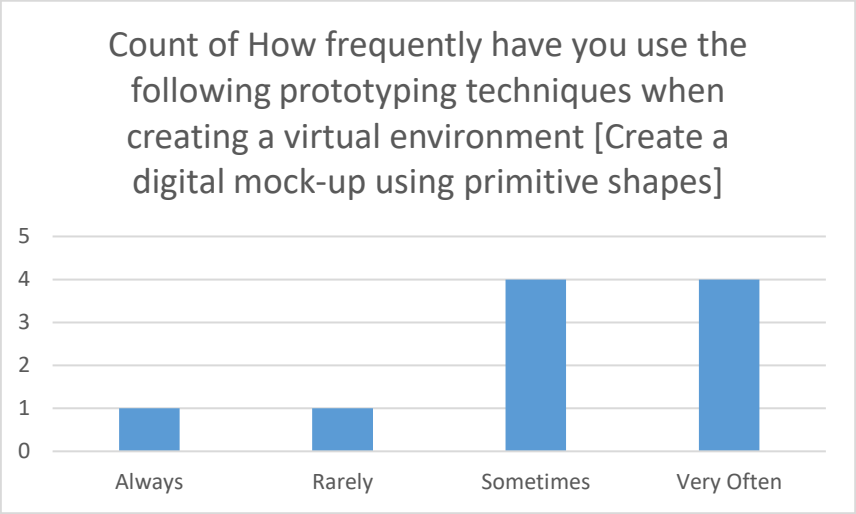

Count of How frequently do you use following tools to create a virtual environment [Blender]
Rarely (once every few months)
Never
0
1
2
3
4
5
6

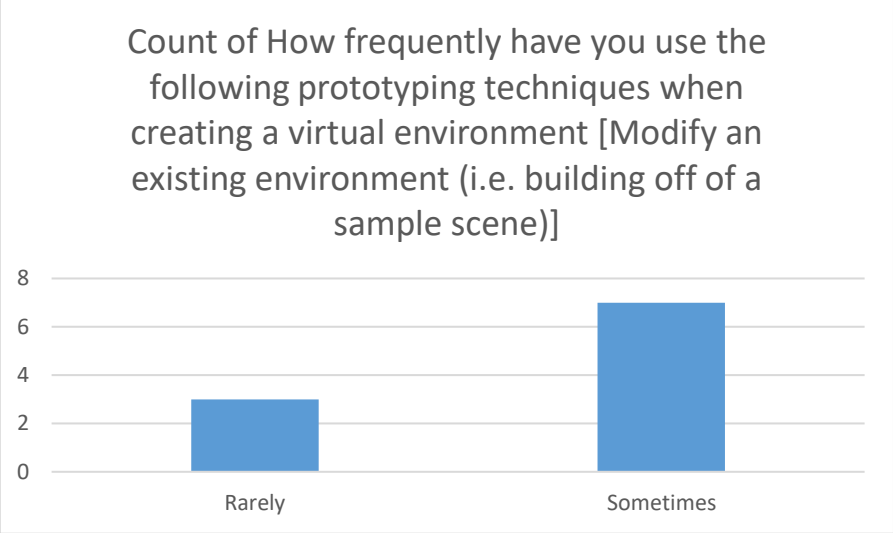

Count of How many hours per week do you spend developing and or modifying virtual environments in an average week?
5
4
3
2
1
0
16 - 20
5 - 10
Less than 5
More than 30

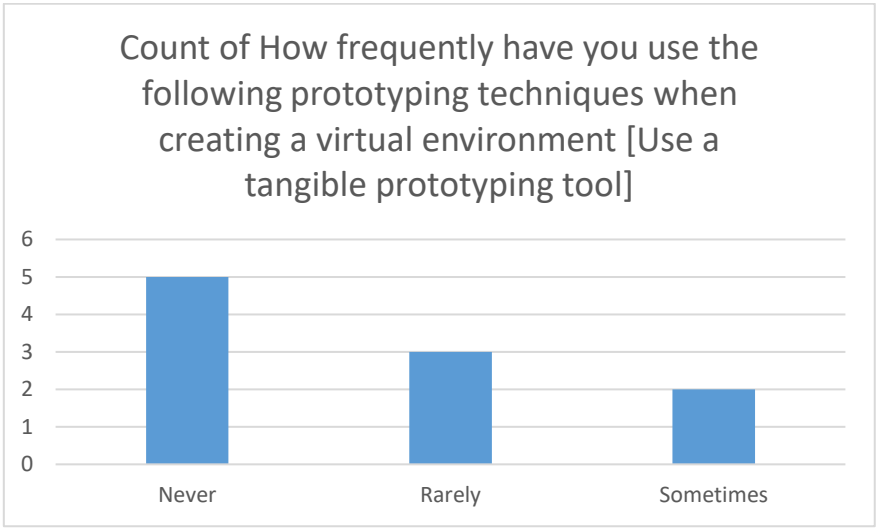

Count of Which best describes your experience with creating virtual environments
I have followed a few tutorials
I have created virtual environments for too many projects to count
I have created virtual environments for 3-5 projects
I have created virtual environments for 1-2 projects
0
1
2
3
4
5

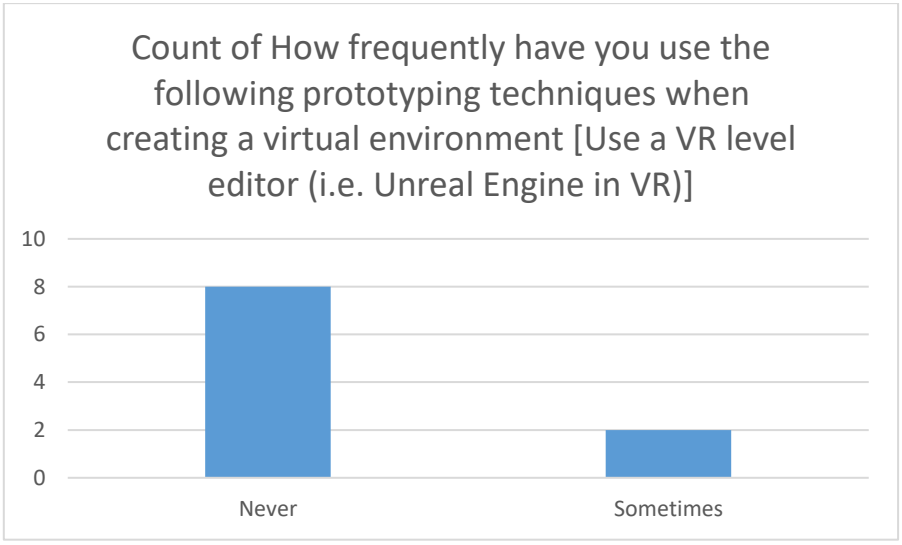

Count of How frequently have you use the following prototyping techniques when creating a virtual environment [Write ideas on paper (i.e. a list of things that are needed in the environment)]
10
5
0
Always
Very Often


Count of How frequently have you use the following prototyping techniques when creating a virtual environment [Create a physical mock-up (i.e. a diorama)]
6
4
2
0
Never
Rarely
Sometimes
Very Often


Count of How frequently have you use the following prototyping techniques when creating a virtual environment [Create a digital mock-up using primitive shapes]
5
4
3
2
1
0
Always
Rarely
Sometimes
Very Often


Count of How frequently have you use the following prototyping techniques when creating a virtual environment [Modify an existing environment (i.e. building off of a sample scene)]
8
6
4
2
0
Rarely
Sometimes


Count of How frequently have you use the following prototyping techniques when creating a virtual environment [Use a tangible prototyping tool]
6
5
4
3
2
1
0
Never
Rarely
Sometimes


Count of How frequently have you use the following prototyping techniques when creating a virtual environment [Use a VR level editor (i.e. Unreal Engine in VR)]
10
8
6
4
2
0
Never
Sometimes

B.3 Positive and Negative Affect Schedule

PANAS

* Required

1. **Enter Unique Participant ID**

2. **When is this survey being administered?** *

   Mark only one oval.

   - ◯ Prior to any prototyping task
   - ◯ After using Unity
   - ◯ After using StoryTIME

Positive and Negative Affect Scale

Please indicate to what extent you feel this way right now for each of the items below

3. *

   Mark only one oval per row.

	Very slightly or not at all	A little	Moderately	Quite a bit	Extremely
Interested	◯	◯	◯	◯	◯
Distressed	◯	◯	◯	◯	◯
Excited	◯	◯	◯	◯	◯
Upset	◯	◯	◯	◯	◯
Strong	◯	◯	◯	◯	◯
Guilty	◯	◯	◯	◯	◯
Scared	◯	◯	◯	◯	◯
Hostile	◯	◯	◯	◯	◯
Enthusiastic	◯	◯	◯	◯	◯
Proud	◯	◯	◯	◯	◯
Irritable	◯	◯	◯	◯	◯
Alert	◯	◯	◯	◯	◯
Ashamed	◯	◯	◯	◯	◯
Inspired	◯	◯	◯	◯	◯
Nervous	◯	◯	◯	◯	◯
Determined	◯	◯	◯	◯	◯
Attentive	◯	◯	◯	◯	◯
Jittery	◯	◯	◯	◯	◯
Active	◯	◯	◯	◯	◯
Afraid	◯	◯	◯	◯	◯

B.4 System Usability Scale

System Usability Scale

* Required

1. **Enter Unique Participant ID** *

_____

2. **Prototyping system** *

Mark only one oval.

- ◯ Unity
- ◯ StoryTIME (No Projection)
- ◯ StoryTIME (With Projection)

3. **I think that I would like to use this system frequently** *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly Agree

4. **I found the system unnecessarily complex** *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly Agree

5. **I thought the system was easy to use** *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly Agree

6. **I think that I would need the support of a technical person to be able to use this system** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

7. **I found the various functions in this system were well integrated** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

8. **I thought there was too much inconsistency in this system** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

9. **I would imagine that most people would learn to use this system very quickly** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

10. **I found the system very cumbersome to use** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

11. **I felt very confident using the system** *
Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

12. **I needed to learn a lot of things before I could get going with this system** *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	⬭	⬭	⬭	⬭	⬭	Strongly agree

B.5 Post Activity Questionnaire

Post Activity Questionnaire

* Required

1. **Enter Unique Participant ID**

_____

2. **Please answer the following questions regarding the tasks performed** *

Mark only one oval per row.

	Strongly Disagree	Disagree	Moderately Disagree	Neutral	Moderately Agree	Agree	Strongly Agree
Unity allowed me to easily iterate the room design	⬭	⬭	⬭	⬭	⬭	⬭	⬭
Unity allowed me to easily iterate the design of animated path	⬭	⬭	⬭	⬭	⬭	⬭	⬭
StoryTIME allowed me to easily iterate the room design	⬭	⬭	⬭	⬭	⬭	⬭	⬭
StoryTIME allowed me to easily iterate the design of animated path	⬭	⬭	⬭	⬭	⬭	⬭	⬭
I would rather use StoryTIME with projections turned off	⬭	⬭	⬭	⬭	⬭	⬭	⬭
Protoyping with StoryTIME felt faster without projections	⬭	⬭	⬭	⬭	⬭	⬭	⬭
Protoyping with StoryTIME felt more accurate without projections	⬭	⬭	⬭	⬭	⬭	⬭	⬭
I would use a tangible interface to create prototypes	⬭	⬭	⬭	⬭	⬭	⬭	⬭
The tracking cubes on the props got in the way	⬭	⬭	⬭	⬭	⬭	⬭	⬭

3. **Please comment or compare your thoughts on the tools used**

_____

_____

_____

_____

_____

4. **Any general comments about the tasks performed?**