Playing Is Creating with PlayTIME: Introducing and Evaluating a Tangible UI-Based Interactive Scenario Prototyping System

by

Daniel Stephen Buckstein

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

February 2015

Contents

List of Figures

List of Tables

Abstract

With traditional game prototyping activities, physical and digital game prototyping tasks are commonly separate, often requiring iteration between the two and different personnel to complete physical and digital tasks. In this thesis, we present PlayTIME: a Tangible Interactive Media Environment for Game-Play, as a means to narrow the gap between digital and non-digital game design techniques. The system is designed to allow game designers to focus on physical prototyping while the computer digitalizes real-world actions into a playable digital game. The current PlayTIME implementation uses Tangible User Interfaces to facilitate specific functionalities in Unity, a widely-used game engine. We hypothesized that using PlayTIME would improve the game design experience for users. To test our hypotheses, we ran a user study to evaluate the usability, creativity support and enjoyment of PlayTIME against the usability, creativity support and enjoyment of Unity alone. We found that PlayTIME had a significant effect on usability, but both qualitatively and quantitatively did not show results better than Unity alone. We found that PlayTIME had an insignificant effect on creativity support. Finally, we found that users enjoyed PlayTIME significantly more than Unity, citing that it is novel and makes design feel more like play.

Keywords: tangible user interface; tangible interaction; video game; game prototyping; game design; game editor; simulation; simulation design; interactive scenario; collaborative design; augmented reality

Acknowledgements

This is basically the only non-scientific part of this thesis, so please allow me to be briefly personal and preface the document with a shout-out to those who helped me through my MSc experience.

First, I'd like to thank the academy for having me here. I am, of course, referring to the University of Ontario Institute of Technology (UOIT), where I spent four years in the undergraduate Game Development and Entrepreneurship program before pursuing a Master's degree. I have had the pleasure of watching the school grow over the past six and a half years, but more specifically I was able to spend time with many wonderful people in the Faculty of Business and Information Technology (FBIT), namely those associated with the Game Development and Entrepreneurship program. The support I have had is just perfect.

The next group of people I would like to thank belong to the Games and Media Entertainment Research (GAMER) Lab, where I had the privilege of working since its conception while I was still an undergrad. I would like to name a few people for their contributions: a thousand thanks to Michael Gharbharan for working on PlayTIME directly and doing an amazing job at it; thanks to Harrison Andrews for helping me with some of the earlier development; thanks to Saad Khattak for being the best programmer and teaching me a great portion of what I know; thanks to Sedona Parnham for providing the artistic talent (she knows too well that we programmers simply cannot art); thanks to everyone at Gamer House for letting me stay over while I ran my study; thanks to Ken Finney for giving me my first job at his company; thanks to Dr. Bill Kapralos and Dr. Lennart Nacke for all the reference letters I've bugged you about; and many thanks to Jessica Clarke for being a supportive academic advisor... and the best graduate program coordinator ever!

Maximum thanks and a standing ovation go out to my thesis supervisor, Dr. Andrew Hogue. We first met when I was new at UOIT and I couldn't wait to show off some stuff I had been working on. Dr. Hogue taught me many things about programming for games, and he has always been around to help when needed. I have always been awestruck and utterly overwhelmed by the ambitions and innovative ideas of Dr. Hogue; we shall coin this collection of awesomeness "*The Hogueverse.*" He also tolerated my too-many-revisions per-chapter, and thousands of questions, for which I am extremely grateful. Through the hilarious times, and the not-so-hilarious times, Dr. Hogue has always treated me well and we managed to pull through as a team.

I would also like to thank the various sources of funding throughout my MSc: NSERC, SSHRC and PVLabs. I was also awarded the Ontario Graduate Scholarship last year which was a great help.

Last and most importantly, I would like to thank my family. My parents have had the biggest impact on my education for putting me through school and for telling me that I could do whatever I wanted to do in life. They were always there for support and offered assistance in the first few years of post-secondary life, while I transitioned to living away from home and becoming independent... okay, *mostly* independent. I'll also give a shout out to my brother for not pestering me any more. He'll never admit it, but he secretly missed me while I lived away from home. While living in Oshawa, it was sometimes difficult to make the trip home to visit everyone, grandparents included, so I hope everyone can appreciate the time I've invested in what I hope is a creative future.

Happy reading!

- Daniel S. Buckstein

Chapter 1

Introduction

1.1 Overview

Creating compelling interactive scenarios is a fundamentally difficult problem for content creators (i.e. video game developers, virtual reality simulation designers, film producers, etc.). Traditionally, designers of such products use developer tools, for example the game engine *Unity 3D* [1], to develop games and prototype new ideas. Another common technique for designing games involves paper prototyping, during which designers implement and simulate their scenarios using non-digital media and physical objects, thus creating a primitive conceptual proof of how the digital version should behave. Although these methods allow designers to collaborate and produces immediate non-digital results, there is still a gap between the prototype and a working digital product that will take a considerable amount of effort. Design is a creative and iterative process requiring an environment to facilitate designers' needs to prototype their ideas quickly and flexibly.

The problem: current methods of digital game prototyping, using computer-based, traditionally independent keyboard and mouse workspaces, are not *naturally* conducive to a collaborative atmosphere. Furthermore, iterative design can be very time consuming since both digital and non-digital prototyping methods must often be revisited before a final decision is made. The traditional iterative design techniques leave a large gap between initial idea formulation stages and the digitalization process. By adding *playfulness* to the act of *creation* and design, we hypothesize that a tangible, hands-on approach to interactive scenario design that autonomously digitalizes real-world activities can create meaningful design experiences. Such a solution would encourage collaboration and move us towards creatively *narrowing the gap* between digital and non-digital, solving issues that exist in the traditional computer-based and paper prototyping-based design work flows bringing digital products into fruition earlier in the development cycle. As a means to solve this problem, we introduce ***PlayTIME**: a **T**angible **I**nteractive **M**edia **E**nvironment for Game-**Play***. *PlayTIME* focuses on the design of games and other interactive media. *TIME* on its own is a conceptual framework for collaborative production tools across multimedia disciplines, with current plans for games and film.

Instead of using traditional computer-based development, TIME software will consist of a variety of stations that use *Tangible User Interfaces* (TUIs), also known as *graspable user interfaces* [2]. These are physical objects that a user can manipulate. The computer recognizes simple attributes of tangible interfaces such as their positions or orientations, or more complex attributes such as gestures and actions performed with them. Being physical objects, TUIs afford to be carried between work stations for designers to interact with in different stages of creation, enabling simple sharing of information and collaboration. It also facilitates hands-on, playful design and creation. Such systems would support the digitalization of physical objects and gestures, thereby helping merge both digital and non-digital design tasks.

1.2 Objectives

The overall objective of PlayTIME is to integrate a set of development tools into a multidisciplinary framework that can be used by designers to rapidly and collaboratively prototype their ideas, by sharing the resources and skills involved with developing a simulation or game.

Augmented Reality (AR) [3] [4] is an emerging technology in game development and is becoming more popular for driving gameplay that interfaces with the real world, as it can be used to overlay the real world with a simulated environment while special patterns are tracked relative to the viewer. Since fiduciary AR markers are a popular technology used for tracking position and orientations, *and* for storing information, the current version of PlayTIME uses fiduciary AR markers to represent information about the game assets and resources.

In this thesis we present the current software prototype of the PlayTIME system. The software prototype demonstrates the use of tangible AR as a tool for designing and prototyping video game scenarios. The ultimate goal of this thesis is to determine if PlayTIME is on the right track using these technologies. We present a user study in which content designers from a variety of creative backgrounds evaluate PlayTIME, an extension of an existing game engine editor, *Unity 3D*, with the use of the editor alone. The system is used to determine whether the use of tangible interfaces, tracked by the computer using AR tags, has a positive impact on scenario design. We evaluate the system by having users complete a design task with it. The effectiveness of the system will be determined by its ability to aid designers in the rapid prototyping of interactive scenarios, while demonstrating the desired results in the game environment.

My[1] primary contributions described in this thesis are (1) the design of the TIME and PlayTIME frameworks; (2) design and implementation of the current implementation of PlayTIME, whose interactions are digitalized, supporting collaborative physical prototyping while simultaneously producing a digital product; and, most importantly, (3) designing and conducting a user study that will ultimately guide future development of the PlayTIME system.

In this thesis I present PlayTIME as a starting point in narrowing the physical-digital design gaps, and how it performs against an existing prototyping tool.

We have formulated a set of hypotheses around the user study, and there we use a variety of statistical tests to discover evidence that supports or rejects the hypotheses. We are hoping that the use of PlayTIME has an effect on the following main areas:

1. **Usability:**
   Hypothesis: The use of PlayTIME will have a significant effect on the *performance* of developers completing the assigned task.
   Hypothesis: This effect will be in PlayTIME's favour.

   The *Computer System Usability Questionnaire* (CSUQ) [5] [6], or the *Post-Study System Usability Questionnaire* (PSSUQ) [5] [7] is used to qualitatively determine if the system is usable. Screen capture software is also used to record the actions performed during the activities, helping us pinpoint the exact areas of struggle. We track the users' actions to see if they are making sense of the system. The study provides us with feedback on how we may be able to improve the system's design using tangible AR interfaces. We ask some questions while analysing the data to help us evaluate the performance of the system: *Does the use of AR-based tangible objects provide designers with a conductive and fluid prototyping experience? Can it improve the quality of their designs? Does it compare positively with the traditional approaches?*

[1] "*My*" contributions are explicitly stated here, but since PlayTIME is a collection of many components under construction by several people, it is only fair to acknowledge the collaborative effort put towards developing PlayTIME. Therefore, the use of "*we*" or "*our*" throughout the thesis refers to the inclusion of others who are involved with developing PlayTIME and related projects, including virtual sculpting, FilmTIME (see Chapter 4 for details).

2. **Creativity:**
   Hypothesis: The use of PlayTIME will have a significant effect on users' *creative output.*
   Hypothesis: This effect will be in PlayTIME's favour.

   The *Creativity Support Index* (CSI) [8] [9] [10] questionnaire is used to rate a set of creativity-related metrics. The CSI indicates whether PlayTIME provides users with a conductive development environment. We also score the output of each activity to determine how participants deviate from the instructions they are given, which is a potential indicator of creativity.

3. **Enjoyment and fun:**
   Hypothesis: The use of PlayTIME will have a significant effect on users' *emotions* and will positively affect users' *enjoyment* of the activity.
   Hypothesis: This effect will be in PlayTIME's favour.

   The study makes use of the *Positive and Negative Affect Schedule* (PANAS) [11], an emotion-rating questionnaire, to measure positive and negative emotions before and after each part of the study. This helps us directly understand users' emotional change through the activity with and without PlayTIME, thereby directly revealing its emotional impact. Feedback at the end of the study directly asks about enjoyment, which we can relate back to their positive and negative emotions. Enjoyment is also discussed as part of the CSI questionnaire.

1.3 Thesis Outline

In Chapter 2, we discuss related works in game design, prototyping, tangible user interfaces, augmented reality and human-computer interaction (HCI).

Chapter 3 presents an evaluation of an existing and uncommon development tool: the mapmaker feature built into the game *TimeSplitters: Future Perfect* [12]. The study was conducted to gain insight on how people with some expertise in a variety of game development disciplines use an unfamiliar scenario-building system to complete a simple game design task.

Chapter 4 introduces the conceptual design for a new collaborative game production environment called *PlayTIME.* The conceptual framework is discussed, including the roles of the people who would use the system. The current implementation of PlayTIME, a plugin for an existing editor, is also discussed. The system uses AR-based tangible interfaces as control.

Chapters 5 and 6 present the key study in which a diverse group of scenario-driven developers evaluate the current implementation of PlayTIME by using it to complete a level design task given an existing project. The study discusses our findings about the interfaces used and identifies the actual uses for each system, complete with time distributions for each feature.

Chapter 7 discusses the limitations of the current system and future directions.

1.4 Summary

In this thesis, we introduce PlayTIME, a conceptual framework for a tangible-based prototyping work flow, and its current implementation. We then present a user study that evaluates the current implementation. From this study we hope to find evidence that PlayTIME is usable in its current state, supports creativity, and is fun. We conclude with guidelines for future development based on our findings from the study.

Chapter 2

Related Works

2.1 Introduction

This chapter contains background research in pertinent areas: game design, game prototyping, tangible user interfaces, augmented reality, and some principles of human-computer interaction. The focus of this chapter is game prototyping. We explain traditional game prototyping techniques and identify how these are supported by the different areas. This information is important to this thesis since its focus is game and simulation scenario prototyping using tangible user interfaces.

2.2 The Game Design Process

In *Rules of Play* by Salen and Zimmerman [13], a *game* is defined as "a system in which players engage in an artificial conflict, defined by rules, that results in a quantifiable outcome." Salen and Zimmerman define a *game designer* as a particular kind of designer who designs *game play*: the "rules and structures that result in an experience for players." *Iterative design* is defined as "a method in which design decisions are made based on the experience of playing a game while it is in development." In essence, the practice involves a steady refinement of concepts and ideas by implementing an idea, testing it out, learning from the successes and failures of each attempt, making relevant modifications and repeating the process. Iterative design revolves around prototyping and testing, which are fundamental to the core of this thesis.

Salen and Zimmerman focus on the central theme of *meaningful play.* The authors provide two definitions of meaningful play. The first definition is:

> *"Meaningful play in a game emerges from the relationship between player action and system outcome; it is the process by which a player takes action within the designed system of a game and the system responds to the*

action. The meaning of an action in a game resides in the relationship between action and outcome." (p.34)

This *descriptive* definition states that there should be a significant relationship between the actions taken by players in a game and the outcome within the game environment. The second definition for meaningful play is:

"Meaningful play occurs when the relationships between actions and outcomes in a game are both discernible and integrated into the larger context of the game. Creating meaningful play is the goal of successful game design." (p.34)

This *evaluative* definition identifies the aforementioned actions and outcomes that are perceptible to players and integrated into the larger context of the game itself; the actions and outcomes must be involved with *the game*, a complex system.

The Art of Game Design [14] provides close-up examinations ("lenses") of one hundred attributes of a game and associates these with the principles of game design. Schell provides an overview of game design, and the process of creating a game beginning with what makes a designer and finishing with the business side of games.

Salen and Zimmerman's approach bridges game design theory with actual practice. In contrast, Schell leans towards the practical end of the spectrum, providing readers with many tips and tricks to enhance their design process. Schell paints a picture of just how complicated the game design process is, seen in Figure 2.1.

Challenges for Game Designers [15] is a book of non-digital design exercises that describes a hands-on approach to game design. The reader is provided with tips and practices for developing different genres of games, and is challenged to implement various board games and card games to practice the design activities described. *Challenges for Game Designers* focuses on non-digital exercises to explore game design and game prototyping without immediately thinking about going digital. This trains designers to be able to iterate with their designs more efficiently before writing any code. This is a key contribution to game design because it thoroughly explains how non-digital games serve as feasible prototypes for digital games later on (see section 2.3 for details on digital and non-digital prototypes).

Fullerton [16] [17] et al [18] describe the concept of *play-centric design*, which is a design process that consistently involves the player in the design process and focuses on the player experience. This concept is an improvement on iterative design and involves prototyping and testing with actual players. In [18] and [19], play-centric design is applied to real game projects in development.

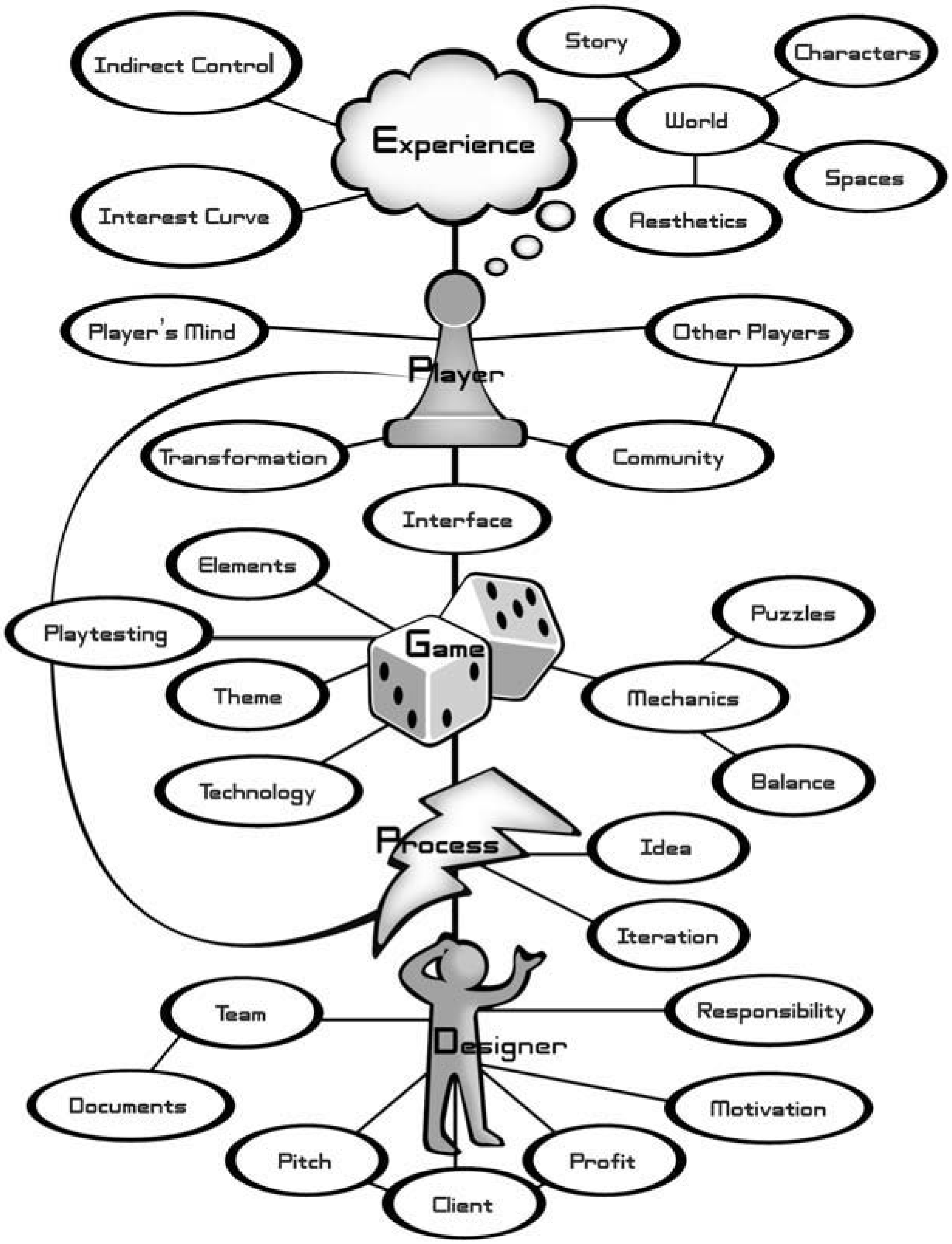


Figure 2.1: The complete game design process from [14] (p.463). Image copyright ©Jesse Schell.

2.3 Prototyping

A *prototype* can be defined as "a limited representation of a design that allows users to interact with it and to explore its suitability" [20]. In the context of game design, a prototype can be some representation of the final game that has limited functionality, but is still usable and reflects the essence of the final game. Prototyping enables content designers to visualize and simulate what they want the product to do. There are different methods of prototyping, but there are ultimately two general categories of prototypes: *low-fidelity* and *high-fidelity* prototypes. In [21], the fidelity of a prototype is defined as "a measure of how authentic or realistic a prototype appears to the user when it is compared to the actual service." Low-fidelity prototypes are not meant to look like the final product or accurately represent the final product. They are preliminary, exploratory designs that use a variety of different materials. Low-fidelity prototypes are distinguishable from the final game, whereas high-fidelity prototypes are more functional and behave much like the game is expected to behave. Low-fidelity prototypes are commonly referred to as paper prototypes (see section 2.3.1).

A high-fidelity prototype is generally computer-based, interactive and includes some implementation of the system design. Thus, a high-fidelity prototype may often be referred to as a software prototype. There are also methods that are considered low-fidelity but still assume a software prototype is used. For example, with *Wizard of Oz prototyping* [20] [22] [23], a user interacts with the system as they normally would; however, all of the responses are controlled by the "wizard," an unseen human operator controlling all of the system outcomes according to the user's input. Thus, it is possible to simulate an interface or system with a computer but without an actual working prototype.

Type	Advantages	Disadvantages
Low-fidelity	Lower development cost Evaluate multiple design concepts Useful communication device Address screen layout issues Useful for identifying market requirements Proof-of-concept	Limited error checking Poorly-detailed specification for code Facilitator-driven Limited utility after requirements established Limited usefulness for usability tests Navigational and flow limitations
High-fidelity	Complete functionality Fully interactive User-driven Clearly defines navigational scheme Use for exploration and testing Look and feel of the final product Serves as a living specification Marketing and sales tool	More expensive to develop Time-consuming to create Inefficient for proof-of-concept designs Not effective for gathering requirements

Table 2.1: Advantages and disadvantages of low- and high-fidelity prototypes [24].

In [24], Rudd et al present considerations for using both low- and high-fidelity prototypes, and even goes as far as summarizing the advantages and disadvantages of each approach (Table 2.1).

In [25], Rettig identifies several issues with high-fidelity prototyping:

- High-fidelity prototypes take too long to build
- Evaluators get caught up in specific details rather than the high concept or mechanics of the system
- Developers resist changes; instead of focusing on a functional software prototype, they would be less hesitant to modify paper and pen mock-ups
- High-fidelity prototypes can set unrealistic expectations
- Bugs in software prototypes can halt production; with low-fidelity it's just paper

High-fidelity prototypes are more useful for selling an idea or showing something more closely representative of the final product. One of the major benefits of low-fidelity prototyping is that it is less costly than high-fidelity prototyping [21] [24] [26] [27] [28].

There has been some discussion on whether low-fidelity prototyping is, in fact, more appropriate on paper or if a computer should be used. A user study of two different systems comparing both paper- and computer-based low-fidelity prototypes [29] concluded that the results are similar (the same quantity and quality of criticisms), but users prefer to use a computer. In [26], a different experiment was conducted to compare two different versions of the same interface design: one version on paper and one on the computer. The purpose of the study was to identify usability problems using the two implementations. Using heuristic evaluation, the study concluded that the paper prototype was better for identifying minor issues such as inconsistencies, whereas the computer version was better for identifying major problems as it closely represented the experience of the final product. In [30], another study compares low- and high-fidelity prototypes for multi-touch, multi-user interfaces on tabletop surfaces.

Much of our knowledge in game prototyping is derived from principles, such as those described above, in user interface design and human-computer interaction (section 2.4). In [15], Brathwaite and Schreiber define a prototype in the context of game design:

> *"A prototype is a playable early version of the game or part of the game constructed by the designer to assist in understanding and enhancing the player experience. It may be done with software ('digital prototype') or*

with physical materials as a tabletop game ('physical prototype' or 'paper prototype')." (p.12)

Their challenges would serve as feasible low-fidelity prototypes for video games. An interactive demo using cubes and spheres in the place of high-polygon, animated models would be high-fidelity because it is playable on a computer and not made of paper and other materials.

A study is discussed in [31] and [32] during which 27 people with game design experience were interviewed to gain a better understanding of the role of prototyping in real design environments. Common responses showed that prototyping is an effective communication tool that allows designers to validate their ideas and goals.

In the game design context, low- and high-fidelity prototypes ultimately align with *physical* and *digital* prototypes respectively.

2.3.1 Physical Prototypes

A low-fidelity prototype for games may be referred to as a *physical* or *paper prototype*. With physical and paper prototyping, the designer uses everyday materials, such as paper, index cards and pens to sketch and simulate the flow of how a game works. Even acting out scenarios is considered physical prototyping and can be helpful [16]. Early prototypes are not meant to look pretty [13], hence they are prepared using primitive materials.

In [33], Snyder provides a widely-accepted definition for paper prototyping in the context of usability testing:

> *"Paper prototyping is a variation of usability testing where representative users perform realistic tasks by interacting with a paper version of the interface that is manipulated by a person 'playing computer,' who doesn't explain how the interface is intended to work."* (p.4)

Snyder's definition can be applied to game design since it explains that a paper prototype is a physical, paper-based imitation of a system that may eventually become digital. Paper prototyping is a common technique used to map out interactive scenarios such as games and simulations, in the areas of level design, game play design and user interface design. Paper prototyping is very common in human-computer interaction (see section 2.4), from which many principles of game design are derived. Designers decide on how a player will interact with the pieces, how the pieces interact with each other, what the pieces do, how these things interact with the environment, etc. Paper is useful because it is easily manipulated and can be drawn on and folded.

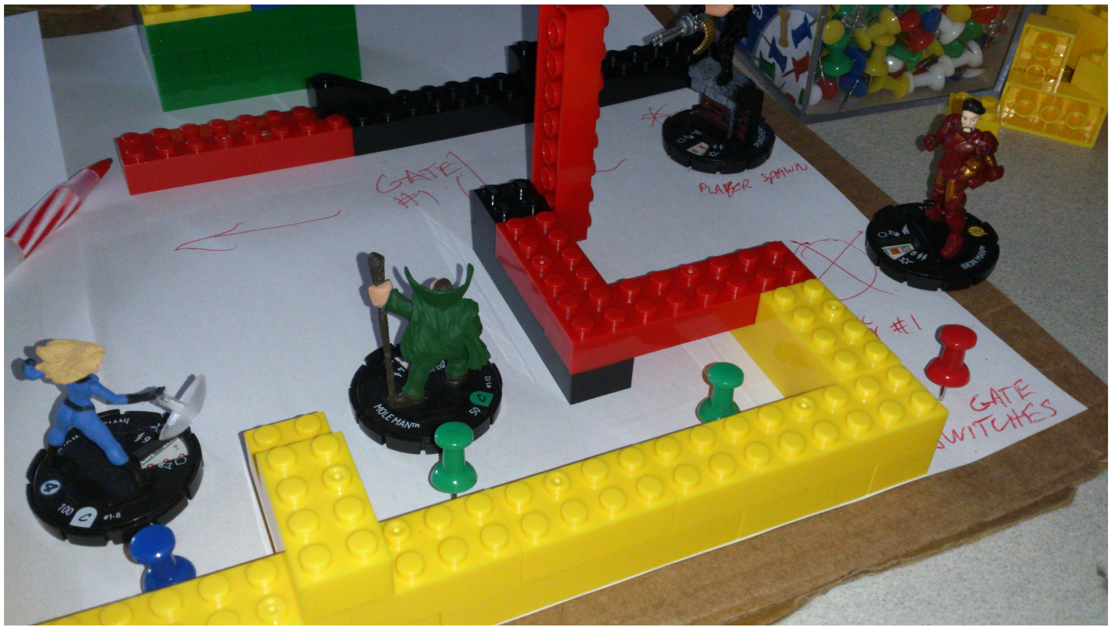


Figure 2.2: This image shows an example of a physical game prototype. The level uses paper for the floor, which affords to be written on. Lego bricks were also used to build walls.

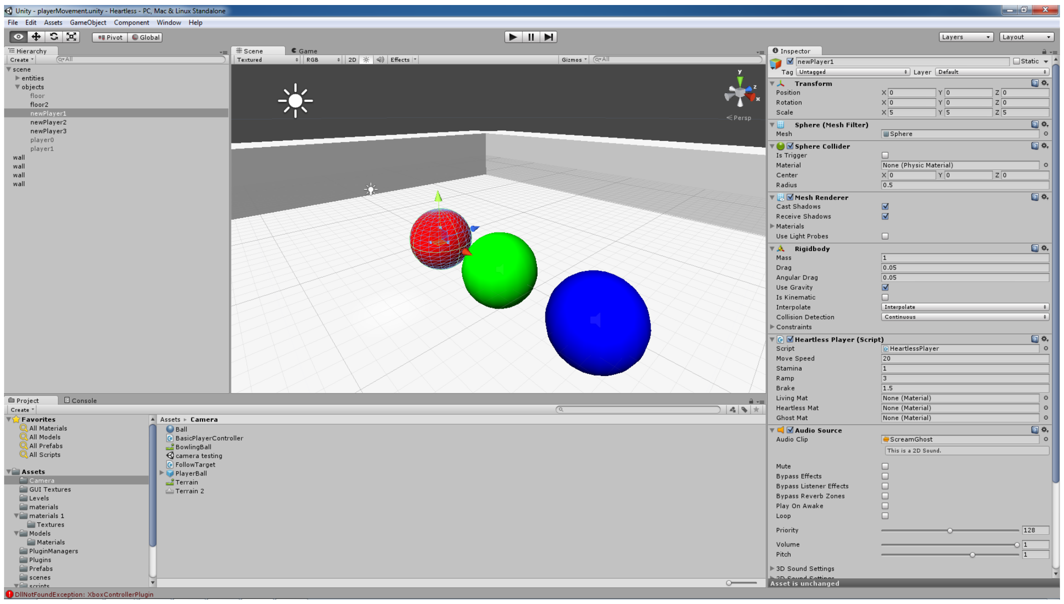
To increase the fidelity of the prototype slightly, designers will typically use physical everyday objects to determine and simulate behaviours of game objects. Designers often prepare hand-drawn maps of a game environment and use anything they can find to represent game pieces. Lego bricks can be used to build obstacles, plastic mini army soldiers can represent characters, small toys can reflect other interactive objects. From miscellaneous toys to office supplies, designers can collect mostly anything to develop a low-fidelity, physical prototype of the game. Physical objects offer freedom in the earlier stages of design so designers can figure out how things will behave. Figure 2.2 shows an example of a physical prototype.

The focus of [16] is play-centric design, described above, which emphasizes the need for paper prototyping. The low-fidelity nature allows designers to focus on the game play and mechanics instead of technology.

Paper prototyping is also involved with the process of iterative design. Salen and Zimmerman report that paper-based games (whether prototypes or board games) are more involved with iterative design than digital prototypes [13].

2.3.2 Digital Prototypes

Since its first release in 2005, *Unity 3D* [1] (referred to throughout this thesis as just 'Unity,') has become one of if not the most popular game engine for developers of



```
newPlayer1                         Static
Tag Untagged           Layer Default
Transform
  Position        X 0   Y 0   Z 0
  Rotation        X 0   Y 0   Z 0
  Scale           X 5   Y 5   Z 5
Sphere (Mesh Filter)
  Mesh            Sphere
Sphere Collider
  Is Trigger      ☐
  Material        None (Physic Material)
  Center          X 0   Y 0   Z 0
  Radius          0.5
Mesh Renderer
  Cast Shadows    ✓
  Receive Shadows ✓
  Materials
  Use Light Probes ☐
Rigidbody
  Mass            1
  Drag            0.05
  Angular Drag    0.05
  Use Gravity     ✓
  Is Kinematic    ☐
  Interpolate     Interpolate
  Collision Detection Continuous
  Constraints
Heartless Player (Script)
  Script          HeartlessPlayer
  Move Speed      20
  Stamina         1
  Ramp            3
  Brake           1.5
  Living Mat      None (Material)
  Heartless Mat   None (Material)
  Ghost Mat       None (Material)
Audio Source
  Audio Clip      ScreamGhost
                  This is a 2D Sound.
  Mute            ☐
  Bypass Effects  ☐
  Bypass Listener Effects ☐
  Bypass Reverb Zones ☐
  Play On Awake   ☐
  Loop            ☐
  Priority        128
  Volume          1
  Pitch           1
  3D Sound Settings
Asset is unchanged
```

Figure 2.3: This image of Unity shows the editor's layout. The scene is shown in the centre, with all of the selected object's editable properties on the right panel. Assets are shown in the bottom panel.

today. It is most popular within the independent developer community for its low cost and expansive tool set. Unity provides its users with a simple interface that allows everything to be clicked-and-dragged into place, with editable text fields used to change values. Furthermore, one of its strongest features is the ability to preview the current state of the game in-development at any time. Similar to modifying real objects and reflecting their behaviours in an engine, designers can move things around in Unity easily and immediately check how these changes affect the game's playability. Figure 2.3 shows Unity's layout with primitive models used in a digital prototype.

Other game engines with integrated editors exist, such as *Unreal Engine* [34], *CryEngine* [35] and *GameMaker: Studio* [36], each supporting similar features and interfaces. Editors allow designers to visualize their creations while they create, and preview the game in-development to see that it works. Developers interested in using and extending existing technologies and providing users with additional functionalities (see Chapter 4) have created tools for prototyping within popular game engines [37]. Other digital prototyping tools, such as Designscape [32], also exist.

Game Jams (one- or two-day game development sessions) have recently become a popular practice for students and developers to work in teams and produce a working game. The Global Game Jam [38] has been held annually since 2009 and encourages participants world-wide to rapidly prototype games according to a theme. Musil et al [39] explain that game jams can be practical in producing working software prototypes since the rules and overall structure of a game jam prove beneficial in other software development areas.

2.3.3 Summary of Prototypes

To summarize prototyping, both non-digital, or physical, and digital prototypes have their benefits and drawbacks. Game engines and tools are difficult to develop, and developing them takes a long time. Even gameplay programming can be difficult: developers must learn whatever technology they want to use to produce their game. Code bugs can also be difficult and time-consuming to track down and fix. Physical prototyping is ultimately beneficial because physical objects *do what we want them to do*: we understand how tangible things will behave when we move them, and this is supported by our spatial awareness and reasoning. Physical prototypes often use cheap and accessible materials that we can easily manipulate in space, allowing game designers to quickly and easily express and experiment with their ideas. One downside of non-digital prototyping is that physical objects are not autonomous and do not have *any* aspect of spatial or artificial intelligence, which is where digitalization helps.

In the context of this thesis, we are trying to narrow the gap between digital and non-digital by reaping the benefits of non-digital while maintaining feasible prototypes, thereby narrowing the gap between the two. The gap exists because the technology and processes that we are used to in digital prototyping *constrain the creative process*, whereas with non-digital tools *anything can be modified*, promoting creative freedom and playfulness.

2.4 Human-Computer Interaction

Many of the principles used in game design stem from one of the largest fields in computer science: *human-computer interaction* (HCI). The Association for Computing Machinery (ACM) defines HCI as "a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them" [40]. HCI research focuses on the science and psychology of how humans interact with computers.

A significant part of HCI research is in the domain of *user interfaces*. A user interface (UI) is a program or system that allows a user to control, manipulate or interact with the computer (see section 2.5 for more on different kinds of interfaces). A lot of UI research looks at the creation and evaluation of interfaces. Nielsen focuses on *heuristic evaluations* of interfaces [41] [42] [43] [44]. In essence, heuristic evaluation is defined as having a person or group of people identify the strengths and weaknesses of an interface according to a set of established principles for interface design. Nielsen and Molich have shown that people are not able to identify errors as well individually; heuristic evaluation works much better when working in groups [41]. The original nine principles for heuristic evaluation ("heuristics") are introduced in [45]:

- Simple and natural dialogue
- Speak the user's language
- Minimize user memory load
- Be consistent
- Provide feedback
- Provide clearly marked exits
- Provide shortcuts
- Good error messages
- Prevent errors

This simplification provides a practical contrast to the hundreds of guidelines originally found in [46]. These nine heuristics were later revised into the ubiquitous *ten usability heuristics* for interface design that are still heavily referenced today [47]. These heuristics, summarized below, are important and widely accepted principles for the design of new interfaces and systems:

Visibility of system status
Inform users what is going on at all times.

Match between system and real world
Use terms that users can understand, instead of highly technical jargon; information appears to be natural and logical.

User control and freedom
Users make mistakes in their selections; the system should offer a clearly-labelled quick way to undo mistakes.

Consistency and standards
Use consistent language and terminologies.

Error prevention
Instead of having users worry about fixing errors, prevent errors from occurring in the first place by removing system components that are prone to errors.

Recognition rather than recall

Reduce users' memory load by displaying their options and actions; they should not have to memorize all of the functionalities of the system, but rather be able to recognize them when seen. Recognition is easy, recall is hard [48]. This is a frequently-addressed issue in UI research.

Flexibility and efficiency of use

Speed up the interactions.

Aesthetic and minimalist design

Focus on including relevant information only; irrelevant information can block out the important stuff and create clutter.

Help users recognize, diagnose, and recover from errors

If an error should occur, it should be clear and descriptive, and offer a solution to fix or revert the problem.

Help and documentation

Provide references for users that can descriptively and clearly guide them with their tasks.

There is another usability-related topic that will be discussed briefly in Chapter 5. Fitts's law [49] describes the time needed to reach an on-screen target, such as a button, text box or scroll bar, as a function of the distance to and size of the target. In [50], MacKenzie discusses Fitts' law in the context of HCI research and design and suggest ways to extend the theorem. In [51], MacKenzie and Buxton further discuss extending Fitts' law by applying it to two-dimensional tasks, including diagonal target approaches.

Usability heuristics are the most important HCI factor in this thesis because ultimately they help identify drawbacks of PlayTIME. PlayTIME itself is relevant to HCI because, as the concept should be able to help us achieve design goals in different media domains, it may be used for exploring user interfaces for websites and other systems. Furthermore, we are directly discussing the use of *tangible user interfaces* as a means of allowing game designers to interact with the computer.

2.5 Tangible & Natural User Interfaces

A *tangible user interface* (TUI) uses physical objects to "represent and control computational abstractions" [52]. In other words, a tangible interface is a physical object whose interactions are mapped to a virtual object; when manipulated, it allows the user to work with the system. The most familiar example of a TUI would be the mouse, whose position on a desk or surface maps to the position our operating system's cursor on the screen. It is a hand-held pointer that offers direct manipulation of the cursor. The study described in Chapter 5 makes use of the mouse. Another straightforward example of a TUI would be the pen used on a drawing tablet. The tablet itself is a metaphor for the drawing surface, and the user holds a pen that behaves as a pointer on the surface; when the pen contacts the tablet it behaves as a mouse click. This is incredibly useful for artists using programs such as Photoshop. Using the tablet pen in real life represents a symbolic interaction with a virtual paintbrush. Sometimes the tablet is sensitive to pressure, which adds an entirely new dynamic to the interaction.

Different tangible interfaces make more sense in different scenarios. For example: both the mouse and a drawing tablet could be used to create a picture in Photoshop or Microsoft Paint, but it would be more appropriate to use the tablet since it better reflects the act of drawing.

In [2], tangible interfaces are introduced as *graspable user interfaces.* The authors use "bricks" as the model tangible object, which directly maps to some virtual object on a display. Graspable UI takes advantage of two-handed interactions, our spatial awareness and caching, and simultaneous control and manipulation of position and orientation. To evaluate the performance of bricks as graspable UI, the authors conducted simple user studies that allowed them to observe *how* people interact with these brick objects, allowing them to develop a "vocabulary" of actions that are afforded by using bricks. These same actions were mapped to a physical representation of the virtual objects that the graspables would manipulate: a stretchable square made of foam core. Next, mock-ups using Lego bricks and prototyping software helped visualize what graspable interactions would look like. A prototype of the bricks and display were prepared to test interactions with graspable UI, and finally a commercial application was modified to enable the use of bricks and testing of graspable UI "in the real world."

The concept of graspable UI was later developed into Ishii and Ullmer's *tangible bits* [53]. The core goals of tangible bits are to (1) transform surfaces in a real (architectural) space into active interfaces between the physical and virtual worlds; (2) "couple bits and atoms" by mapping graspable objects to the digital information that pertains to them; and (3) use ambient media, such as light and sound, as background interfaces for human periphery. Essentially, the world itself can be effectively evolved into an interface with the use of tangibles.

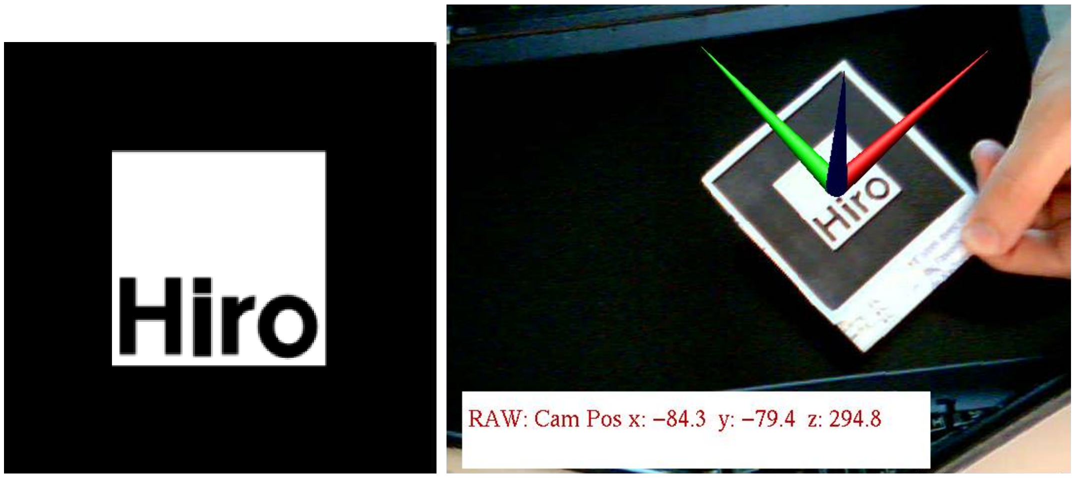
In [54], the authors expand on tangible bits and explicitly compare TUI representations against graphical representations. They also present a variety of "genres" of TUI applications.

Tangible interfaces have also been shown to enhance collaboration and foster collaborative activities, which is particularly important for this thesis [55].

A *Natural User Interface* (NUI) can be defined as "a user interface designed to reuse existing skills for interacting appropriately with content" [56]. In essence, we as humans are naturally skilled with our bodies, arms, hands, fingers, etc. Natural and gestural interfaces use our physical features and movements to interact with the system and its content. Sometimes there are no buttons, in the case of Microsoft's *Kinect* [57] [58], which uses hand gestures to simulate pressing virtual buttons, foot gestures to simulate kicking a ball, and full-body actions to perform other tasks. Devices like the *Leap Motion* [59] focus only on hands and fingers. Sony's *PlayStation Move* [60] and Nintendo's *Wiimote* [61] are hand-held motion controllers whose spatial positions and orientations are tracked by the system; therefore our hands' motions in space are symbolic to the system.

2.6 Augmented Reality

Augmented Reality (AR) is a technology that superimposes virtual media (video and/or audio) in a computer program or simulation on to some video feed of real world and real objects. In [62], it is stated that AR *supplements* reality instead of completely replacing it, as virtual environments do. Typically, AR applications run a video feed from a webcam or built-in camera (in the case of mobile devices). Using software such as *ARToolKit* [63], each frame of the video feed is processed and scanned for recognized patterns. ARToolKit tracks the thick black borders on *fiducial markers* (Figure 2.4). If a pattern on a marker is recognized, its position and orientation are calculated relative to the camera and returned to the application for use in graphics, physics, or anything else that may require 3D transformation data. If a device has a gyroscope, this can also be used to determine the orientation of the camera. Optimization algorithms, such as a Kalman filter [64], can be used to estimate the change in a virtual object's transformation over time and return the optimal pose. Other tracking methods for AR include Parallel Tracking and Mapping (PTAM) [65], which does not use markers and instead uses feature recognition in a sequence of images (video) to determine where the viewer is located in the world.



```
RAW: Cam Pos x: −84.3  y: −79.4  z: 294.8
```

Figure 2.4: An example of a fiducial marker, by itself on the left and in-use on the right. The pattern is recognized by ARToolKit and a transformation is computed based on the pose of the marker as it appears in the frame. A virtual object is then superimposed on top of the image.

2.7 Tangible AR

AR has been widely recognized for its usefulness in collaborative environments [66] [67] [68] [69] [70] [71] [72] [73] [74]. The benefit of collaboration is probably why AR software is a popular tool for developing TUIs; this is an important theme throughout this thesis. Many researchers use AR as a mapping tool for tangible interfaces.

Mark Billinghurst introduced the MagicBook, which uses markers on a real book with printed text to overlay virtual scenes on the pages [66] [67]. In [75], students prototyped tangible interfaces for a variety of applications by attaching fiducial markers to movable parts; the markers were tracked and their orientations were used to determine system states. In [76], the authors discuss designing tangible interfaces using AR, and identify special actions that users can perform with AR markers, such as shaking. The article bases a lot of its content on a demo called "Shared Space" [3], with which the authors conducted a user study. The users were pleased with the presented system, but there were shortcomings with tracking and in the event of marker occlusion.

Tangible tiles are described in [77] as hexagons with tiny infra-red markers attached to the corners. The tiles are used to augment the board game Settlers of Catan, which uses hexagonal tiles. Tangible tiles are a TUI that are used much like AR markers, and suffer the same drawbacks, such as occlusion. They can be tracked over time, and their movements and interactions with other tiles are symbolic to the system.

There have been many tangible tabletop games built using AR technology. In [78], a game called *Art of Defense* is introduced. The game uses hexagonal AR markers to represent the locations of virtual objects. The authors found that the tangible layout allowed players to use spatial reasoning and communication.

A digital prototyping tool kit known as *ToyVision* is introduced in [79]. ToyVision uses fiduciary markers to track a variety of tangible objects that, when interacting with each other, establish game play. The tool kit is used to prototype a game in [80]. This is extremely valuable to PlayTIME since one of the goals is to be able to have object interactions represent symbolic in-game logic.

Kaltenbrunner et al [81] introduce their system, called *reactTIVision*, a framework designed for the construction of tabletop TUIs. The tangible parts are amoeba-shaped AR markers.

Osmo [82] harnesses both tangible interfaces and more AR technology on iOS devices to provide children with hands-on, playful, tabletop-based exercises in spatial awareness and problem solving. The three main uses for Osmo are: "*Tangram*," a digitalization of the classic dissection puzzle, with which users can build and explore simple objects and solve puzzles in-application; "*Newton*," which identifies lines and other objects on the table and converts interprets them as virtual physical objects; and "*Words*," with which users show physical tiles with letters on them to describe a photograph. Osmo is extremely relevant because it demonstrates how tangible objects, in general as opposed to AR-based, can be used to represent information that is interpreted by a digital application. It also demonstrates tangibles' abilities to be used in spatial and creative tasks.

castAR [83] is a new virtual reality and augmented reality technology that uses glasses with micro-projectors to create a personal virtual reality experience. The technology has demonstrated the use of RFID technology to register virtual meanings to physical objects, thereby utilizing TUIs to play and create games.

Tangible user interfaces and augmented reality together are relevant to PlayTIME since fiduciary patterns were chosen as the computer's method of recognizing tangible objects and allowing us, as designers, to interact with the computer in ways that are more symbolic to being a designer.

2.8 Summary

In this chapter, we provided a brief overview of game design and prototyping techniques both within and outside of gaming contexts. We explain that low-fidelity and physical prototypes are useful for being low-cost and very easily manipulable, whereas high-fidelity and digital prototypes are built for higher-quality, marketable versions of the product. We also discuss some of the technologies that can be used to perform physical prototyping tasks. Tangible interfaces work well for prototyping since they can be picked up and moved around. Positions and orientations of objects can be tracked using AR software. Using AR to define a tangible interface system has been shown to work well for gaming and prototyping activities. Prototyping is relevant to this thesis since the system being developed aims to digitalize the process of physical prototyping.

Chapter 3

Designer Actions Study

3.1 Introduction & Motivation

Prior to developing PlayTIME and related interactive tools, we conducted a preliminary user study to understand the actions developers take when using a constrained and minimal development environment. The study invited users, new to a design tool, to provide feedback about the system, for example: what they liked, what they disliked, which features they found themselves using the most and what was used the least. They were assigned a level design task to complete within a time limit, during which they planned out a level to build for an existing game and used the features of the system to construct it.

The study was exploratory in nature, so there was no initial hypothesis. The expected outcomes included feedback from each participant on their experience using the provided interface, as well as a measurement of the time participants spent performing specific tasks and using different features of the system during the study. This study was important because the survey results, performance analysis and feedback would ultimately serve as guidelines for the design of a new interactive scenario development tool using a tangible interface. The results indicate what actions designers want to do while developing a level and how long they perform these actions. The study was ultimately intended to help guide the development of a new prototyping system by giving us solid understanding of the needs of potential users of the system. The study was designed to tell us what features are most important for a design interface, what features will be used and the time it takes to use them.

The design tool used for this experiment was the mapmaker feature in *TimeSplitters: Future Perfect*, described in detail in the next section (3.1.1). It was selected for its extensive selection of objects with predefined behaviours that can be placed into the world with the press of a button. Furthermore, the mapmaker supports interactive preview modes, which users could use to view their level during development.

3.1.1 TimeSplitters: Future Perfect

TimeSplitters: Future Perfect ("TSFP") is a first-person shooter game that involves time travel [12]. The game was developed by Free Radical Design and published by Electronic Arts in 2005. For this study, the selected tool is the mapmaker feature built-into the game. The mapmaker gives users the creative freedom to build a story level with objectives and logic that features their favourite weapons, items and characters from the game's main story. The editor also facilitates other game modes that are typically found in first-person games, such as team death match, capture the flag, domination, last man standing, and many more. Furthermore, users can preview their level in the preferred game mode at any time.

The TSFP mapmaker has two preview tools: an in-game interactive play-through, and an in-editor 3D view of the level. The 3D view is instantly accessible by pressing a single button, and exited just as easily. This view displays a texture-less layout of the map, showing how tiles are laid out in the world. The user can make changes to the tiles and immediately see the 3D representation of the map. However, it is unfortunately not interactive and does not display the objects and items in the world; it only shows how the tiles are arranged, and this view cannot be rotated or scaled. Furthermore, since the editor is multi-floored, higher floors may occlude lower ones, which makes viewing difficult in some cases. For this, the user must start an interactive preview. Figure 3.1 shows an example of the 3D view. This feature could be considered a lower-fidelity representation of the level due to its limitations.

The interactive preview is a compiled, playable representation of the map. It is the actual output of the level design, a demo of the game itself: TimeSplitters, taking place within the level created in the editor, as seen in Figure 3.2. With this feature, the user can preview exactly how their level will behave when played by others. Upon starting the interactive preview, the game is compiled into a playable version of the level. The level is then loaded and then the user temporarily exits the editor and plays through the game. This preview can be quit at any time and the user will be returned to the editor. This feature provides users with a higher-fidelity look at the level. See section 3.4.3 for more information about the participants' use of the preview features.

Aside from building the level, there is also a logic editor that can be used to make things happen within the game. The user can create and select 'triggers' (events

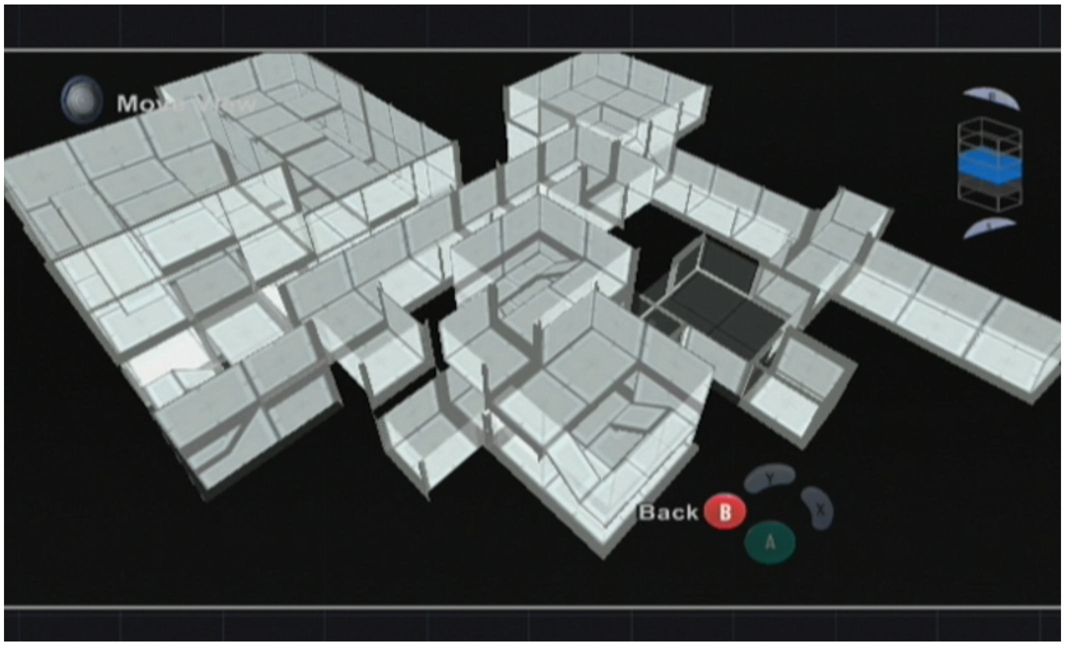


Figure 3.1: The 3D preview in the mapmaker. It only displays the tile arrangement; none of the items or interactive parts of the level can be seen.

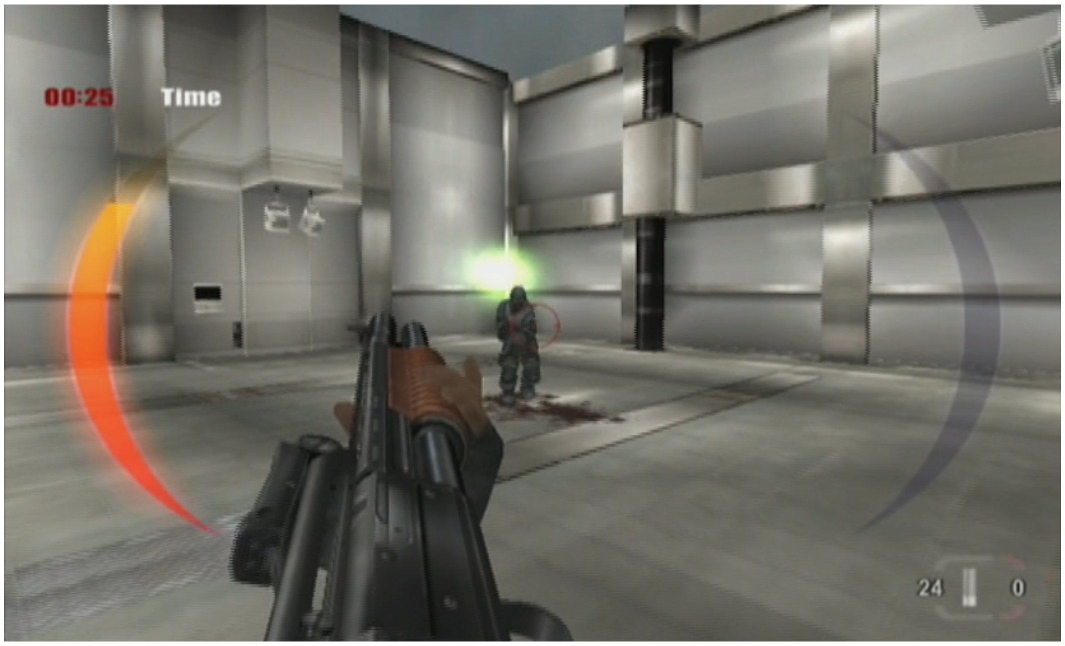


Figure 3.2: In-game screen-shot of TSFP. The player is caught reloading while in a fire fight with an enemy character!

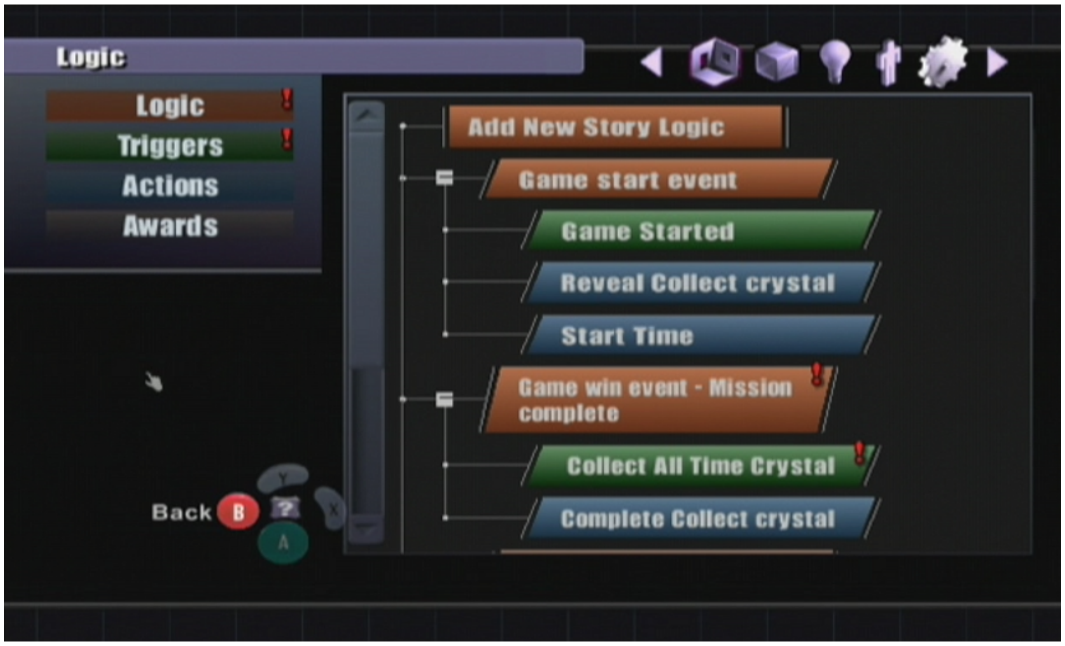


Figure 3.3: The logic editor's tree layout. In this image the logic for the very beginning of the game and the win-condition are shown. Blue boxes are triggers and green boxes are actions. The orange boxes represent events with linked triggers and actions.

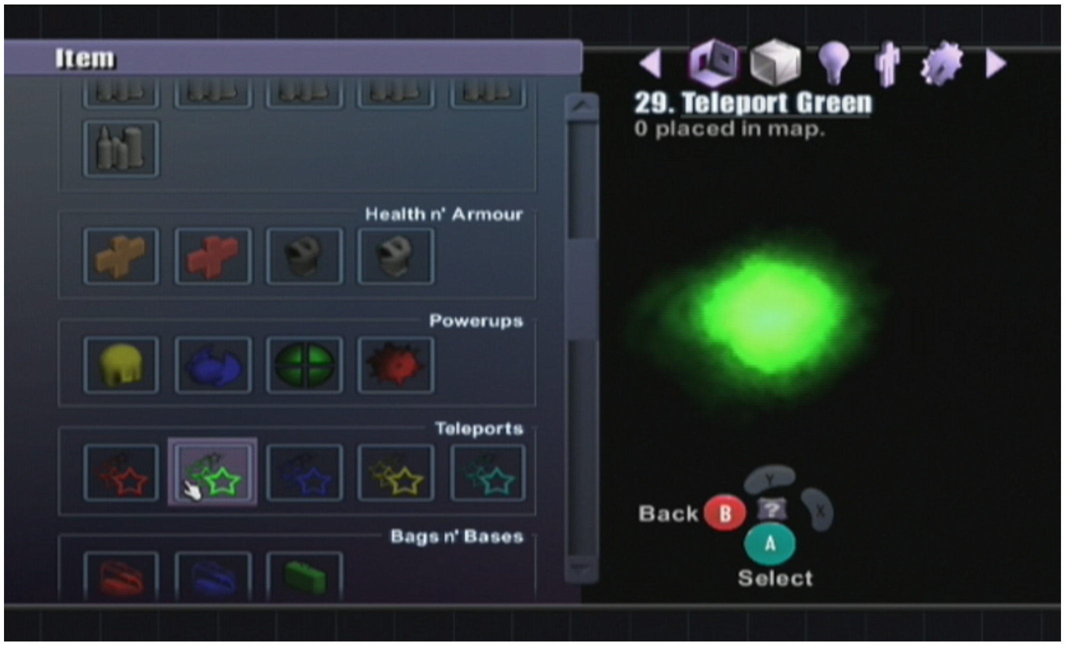


Figure 3.4: The items menu in the mapmaker. The user can scroll through a variety of items, represented by icons, and see what the object looks like in the world. Here a green teleport is selected in the menu on the left and displayed on the right.

that cause something to happen) and 'actions' (events that happen as a result of something else) to prepare simple and complex game events. In essence, this is a graphical interface to simulate programming or scripting. The main menu for logic editing is a collapsible tree that displays events and their related triggers and actions. In the map itself, all objects associated with logic operations are marked with a blue dot. Broken logic items are clearly marked with a red exclamation mark; the user receives a warning if a preview is started with unfixed logic issues. Figure 3.3 shows the editor's logic tree.

The editor provides users with a constrained grid-based environment in which they can place tiles, items, enemies and quickly connect objects together to create simple logic operations. There are many objects that have pre-determined behaviours that only need to be placed in the map by the user, all categorized and arranged in scrollable menus (Figure 3.4). This system was ultimately chosen as it facilitates basic map editing and creation abilities in a simple interface.

3.2 Method

3.2.1 Session Overview

The goal of the study was to observe what actions would be performed by game designers when provided with an unfamiliar game creation tool. To facilitate this, they were asked to complete a task using TSFP's mapmaker feature; they were allowed full creative freedom, constrained only by the content of the mapmaker itself.

Participants began the study by reading and signing an informed consent form which provided them with a description of the study (REB file 13-022; see Appendix A.1). They then filled out the demographics questionnaire in which they described their experience and expertise in different disciplines of game and simulation design, and experience using a variety of game development-related tools. Participants were then introduced to the development tool used for the experiment. They were allowed to try the editor for a few minutes before the experiment began. They were provided with a written description of a task they were required to complete in the editor with only a few constraints:

- The game genre is first-person shooter
- The player will only have five minutes to play through the level
- The level must have a puzzle mechanic

- The player must collect crystals; you must place one or more crystals within the map
- You have doors, keys, switches and other objects to help you build a compelling puzzle mission

The participants were provided with an image of the control scheme for the editor. They were also given a list of the editor's features which they were allowed and not allowed to use to complete their task. A notebook turned to a blank page was left beside these materials; they were not instructed to use the notebook but it was there if they wanted to plan out their levels.

A simple template map was provided to all participants in the TSFP mapmaker. The starting layout had a few tiles in the grid, a starting point, one enemy, an example switch-controlled door, one crystal, and incorporated logic for the winning condition (collectibles picked up) and the losing condition (in-game time limit exceeds five minutes). They were instructed to build from the starting layout, adding to it, removing from it and changing it however they felt necessary to complete the task. See Appendix A.1 for the complete task description and all of the supplementary materials provided to participants for this study.

Participants were allotted 45 minutes to build the level and were informed if they violated the constraints they were given. They were also allowed to ask for help as needed. Throughout the study, each participant's progress was recorded via screen capture software, and the participants themselves were recorded on video. At the end of the session, each participant filled out a survey in which they described their experience, rated the features that they used and discussed their favourite and least favourite features and moments of the experiment.

3.2.2 Video Analysis

After the study, determining the actions performed and time spent by participants required analysis and coding of the video recordings. For each participant, the first video analysed was the recording of the participant. Each time they picked up the notebook to either make a drawing or reference it, the action was recorded with the duration of the drawing or reference.

Next, the same process was used on the screen capture video for a set of pre-determined key actions. Every time a participant started doing a new task, the time of this switch was recorded, thus giving us the total duration and frequency of the actions. For a list of the key actions monitored in the video analysis and their statistics, see section 3.3.3.

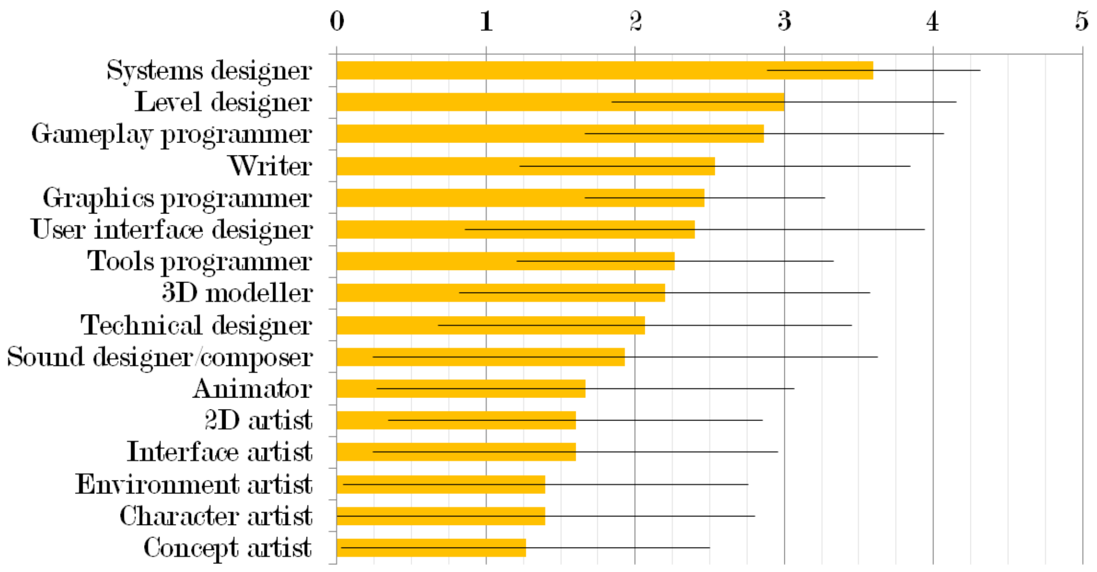


Figure 3.5: A bar graph of self-reported proficiencies in a variety of game development disciplines. The data represents the mean 'expertise' in each of the listed fields. The error bars represent one standard deviation *in either direction* from the mean. This was consistently selected for error bars throughout the thesis to reflect the overall variance or dispersion of the data. Larger error bars indicate widely-spread or inconsistent data, whereas small error bars reflect agreement between samples.

3.3 Results

3.3.1 Demographics

Fifteen participants were selected to take part in the experiment. The only requirement for participation was that the potential participant must have had some experience in any kind of game development, with strengths in any of the common game development disciplines (programming, animation, design, etc.). Most of the participants were studying game development. The average age of participants was 24 years old. The average game development experience was approximately 4.5-6.5 years; this accounts for any time associated with developing games, including studying. Only one of the fifteen participants had previously used mapmaker.

Participants were asked about their expertise in a variety of common roles in game and simulation development. Figure 3.5 displays the average self-reported expertise in various roles, with error bars to represent one standard deviation. Expertise was selected using a Likert scale with the following ranks:

0=Not applicable; 1=Beginner; 2=Competent; 3=Intermediate; 4=Advanced; 5=Expert

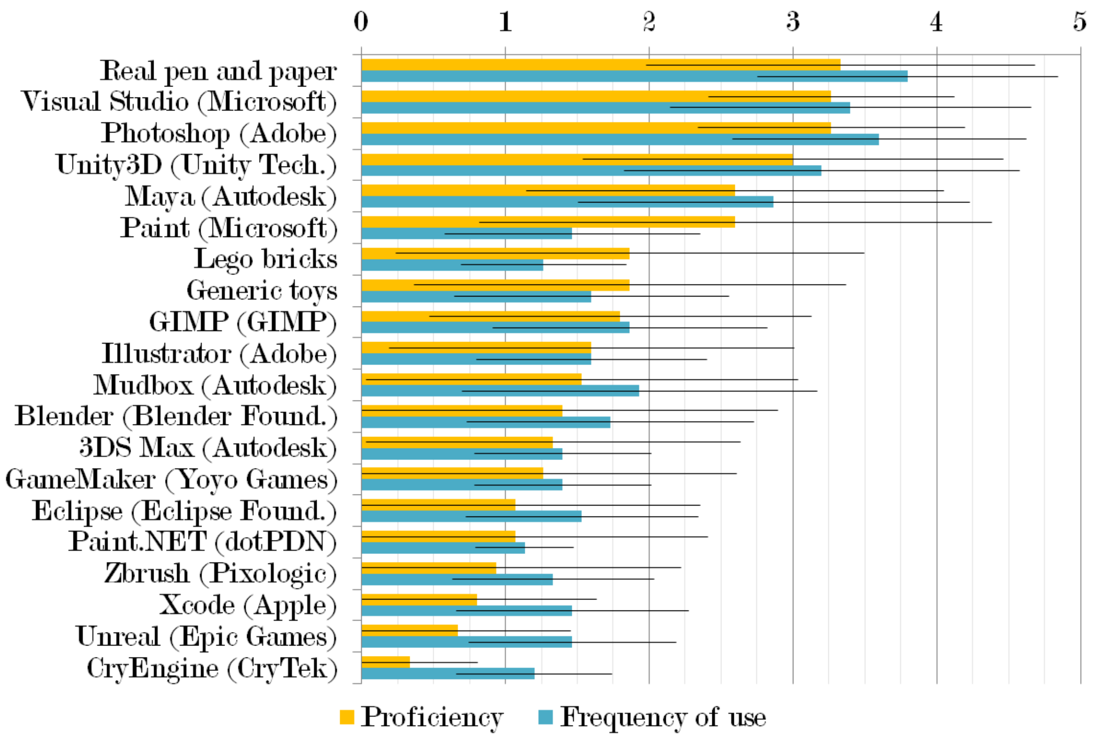


Figure 3.6: A bar graph of self-reported proficiency using a variety of game development tools. The orange bars represent the mean proficiency using each tool, and the blue bars represent the mean frequency of use with each tool. The error bars represent one standard deviation from the respective mean.

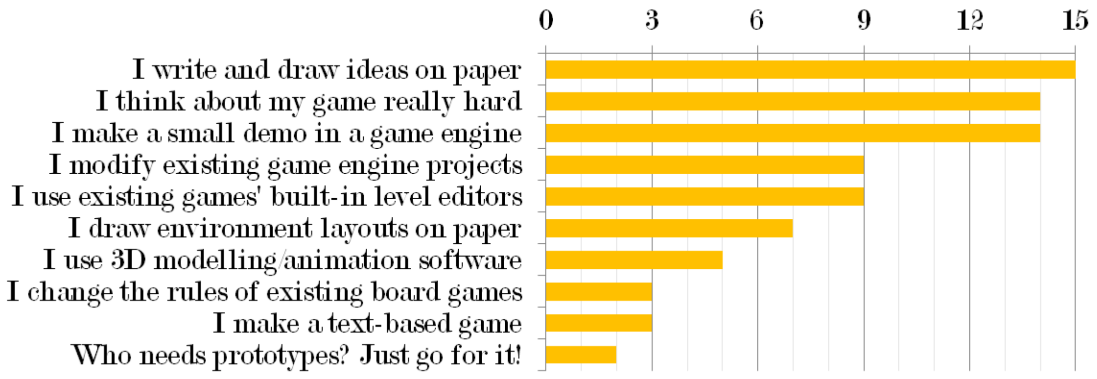


Figure 3.7: Here are some of the tactics that may be used while designing a level. The graph shows how many people use each of the listed tactics during their individual design process.

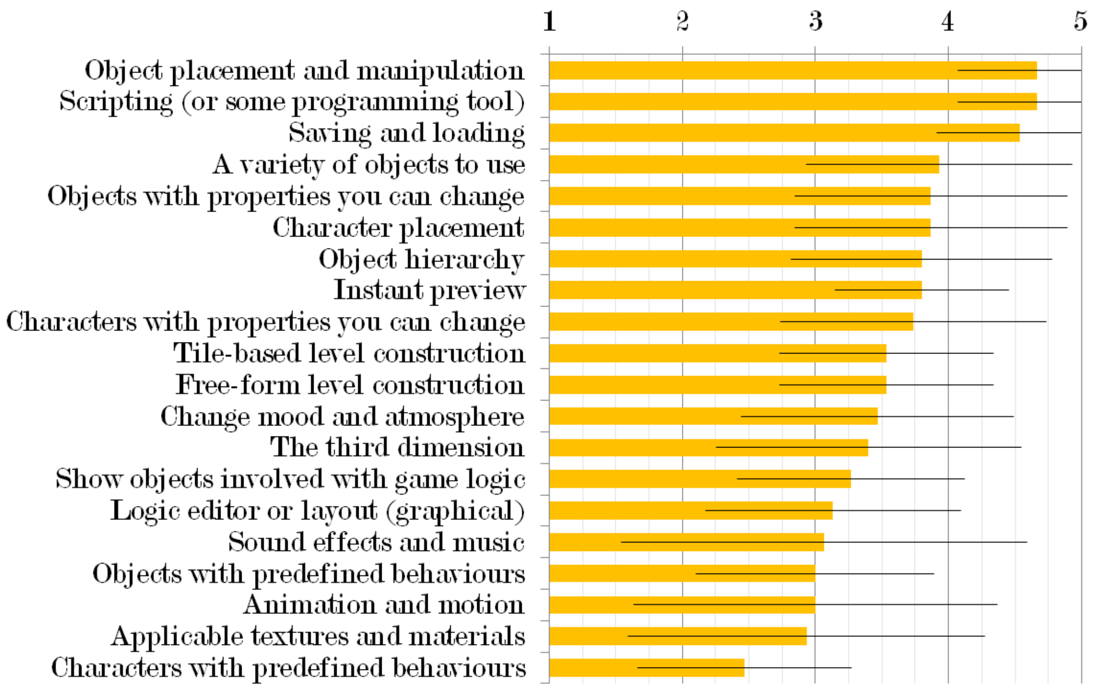


Figure 3.8: Here is a list of features that one might find in a popular game development editor. The bars in this graph represent the mean importance of each feature to the participants. The error bars represent one standard deviation from the mean.

Next, participants described their expertise using popular tools for game development. Figure 3.6 shows the participants' self-reported proficiency using a variety of development tools, as well as how often they use each tool on average. Proficiency was measured using the same scale as above, whereas frequency was measured with a different Likert scale:

0=Not applicable; 1=Never; 2=Rarely; 3=Occasionally; 4=Often; 5=Always

Note that the participants were informed that their rating of each tool is in the domain of game development and not just general use. Most participants preferred using Microsoft Visual Studio as a programming development environment and Unity 3D was the preferred game engine.

Participants were then asked to choose from a list of design tactics and practices they used frequently while designing games. A list of these practices can be found in Figure 3.7 with the number of participants that do each practice.

The participants were provided with a variety of features they may find in a popular game engine. They were asked to rank the importance of these features. The average ratings can be found in Figure 3.8 with the following scale:

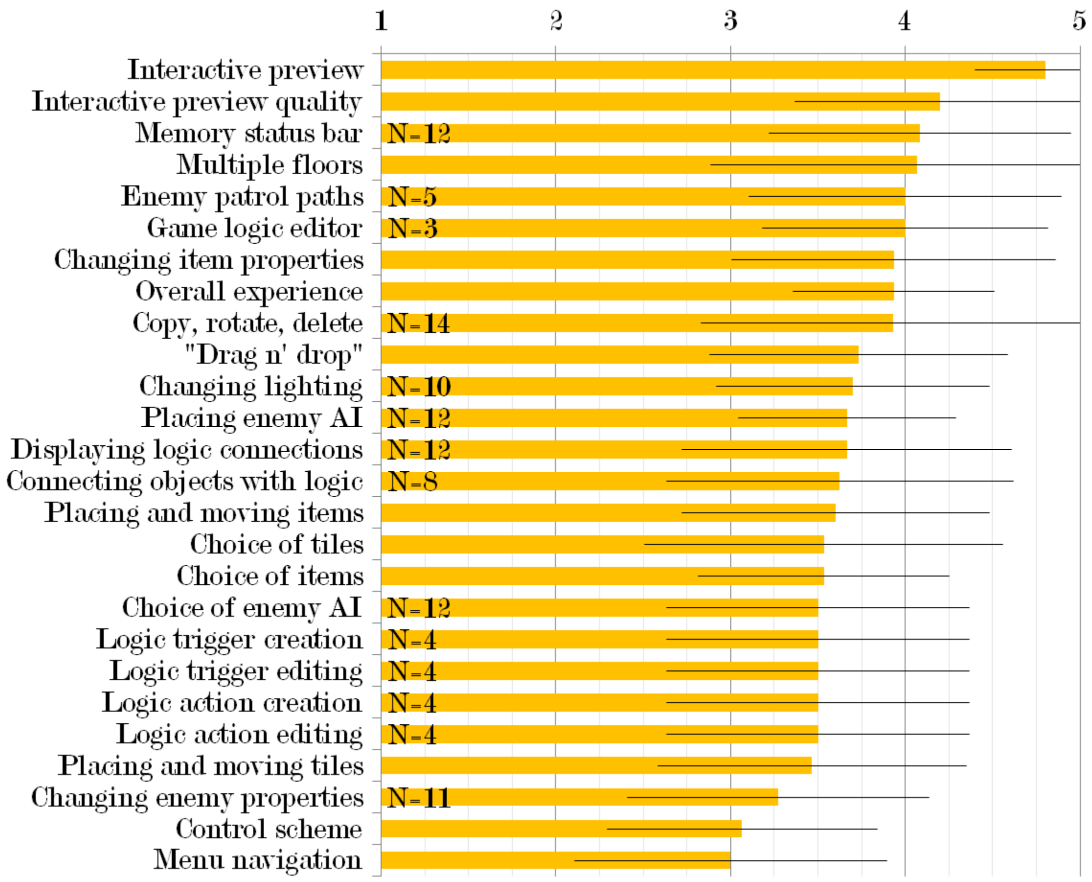


Figure 3.9: The ratings of a variety of features used in the TSFP mapmaker. The bars represent the mean rating for each feature. The error bars represent one standard deviation from the mean. Additionally, the response count (N) is displayed for the features that did not receive a valid response from all participants.

1=Not important; 2=Slightly important; 3=Moderately important; 4=Very important; 5=Essential

Finally, the participants were asked to describe their design processes. Their detailed responses can be found in Appendix B.2.

3.3.2 Post-Study Questionnaire

The post-study questionnaire asked participants to rate their experiences with the different features within the TSFP mapmaker. The scale used for each feature is:

1=Hate; 2=Dislike; 3=Neutral; 4=Like; 5=Love

The results from this part of the post-questionnaire can be found in Figure 3.9. In this bar graph, each feature is listed with the number of valid ratings out of the fifteen participants (N). The bars represent the mean rating for each feature. The error bars represent one standard deviation in either direction of the mean for each

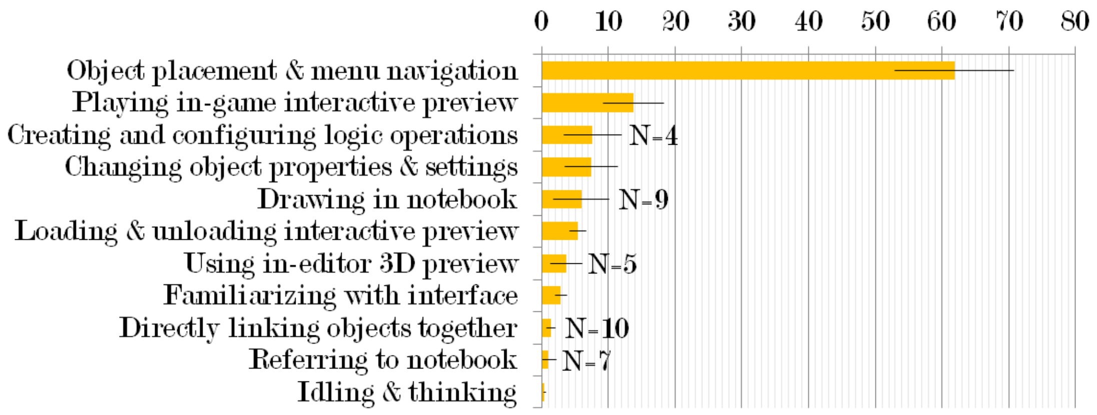


Figure 3.10: Percentage of time spent performing different design tasks, both within and outside of the editor. The bars represent the mean rating for each feature. The error bars represent one standard deviation from the mean. Additionally shown is the number of participants who used the feature.

feature. A response was deemed valid if the participant claimed to have used the feature (did not answer 'Did Not Use' when filling out the questionnaire), and this claim was confirmed in the evaluation of the participant's screen capture and video recordings. A rating was excluded if the participant said they did not use the feature or the videos did not show them using the feature at any point during the session.

3.3.3 Screen Capture & Video Recordings

Each participant was recorded on video, and their design session was recorded using screen capture software. Analysis of the videos (section 3.2.2) gave a clear indication of how each participant spent the time allotted to them to complete their task. The average percentage of time spent doing different activities can be seen in Figure 3.10, shown in further detail in Table 3.1. Table 3.2 shows the average event frequencies and durations. The list of activities includes actions observed in-editor, such as placing objects, and those observed of the participants themselves, such as drawing in the notebook that was provided to them (see section 3.4.6 for details about the notebook).

Feature	N	M_t	$M_{\%}$
Object placement & menu navigation	15	34m 20s	61.87
Playing in-game interactive preview	15	7m 25s	13.73
Creating and configuring logic operations	4	4m 20s	7.61
Changing object properties & settings	15	4m 03s	7.42
Drawing in notebook	9	3m 17s	5.97
Loading & unloading interactive preview	15	3m 02s	5.42
Using in-editor 3D preview	5	2m 01s	3.69
Familiarizing with interface	15	1m 34s	2.85
Directly linking objects together	10	47.7s	1.45
Referring to notebook	7	28.3s	0.95
Idling & thinking	15	15.0s	0.45

Table 3.1: This table summarizes how participants in the designer actions study spent their time using different features and doing different tasks. The table shows the number of participants, out of fifteen, who did each of the listed tasks (N), as well as the mean time spent on these tasks per-session in minutes and seconds (M_t) and as a percentage of the total time per-session ($M_{\%}$).

Feature	N	M_d	M_o
Object placement & menu navigation	15	1m 35s	21
Creating and configuring logic operations	4	1m 34s	2
Playing in-game interactive preview	15	1m 28s	5
Drawing in notebook	9	59.3s	3
Changing object properties & settings	15	18.9s	12
Directly linking objects together	10	17.0s	2
Using in-editor 3D preview	5	9.6s	12
Referring to notebook	7	6.4s	4

Table 3.2: This table shows the frequency of events occurring during the study. The columns are the number of participants who did each of the listed tasks (N), the mean duration of the events in minutes and seconds (M_d), and the average number of times the events occurred (M_o).

3.4 Discussion

3.4.1 Participants

As seen in Figure 3.5, the role of Systems Designer had the highest levels of reported competence, with the smallest deviation (M=3.6, SD=0.71). The systems designer is the person responsible for coming up with and prototyping game mechanics and rules. The role of Level Designer had the second-highest average competence (M=3.0, SD=1.15).

Of the 15 participants, 14 were students and likely had not spent time doing professional development in any of the creative disciplines mentioned in the questionnaire. It would have been useful to be able to evaluate the reported competence values in the context of professional game development.

3.4.2 Development Process

Each participant would start by playing through the template map they were given using the interactive preview, and then learning how to use and navigate the editor. Users would consistently scroll through the different menus and see what items and tools they had at their disposal. After coming up with initial ideas, participants spent time scrolling through the tile menu and building the map, or the shape of the traversable game world. Level types varied between complex multi-floor mazes and large open spaces with walls as obstacles.

Once they had the basic map of their level done, participants moved on to placing items, enemies and other interactive objects throughout the world. Although the instructions said one or more crystals could be placed, participants often used just the one that was provided in the template map, and it was placed in some area of the map shortly after the study began. Some examples of level layouts created by different participants can be found in Figure 3.11.

Table 3.1 shows that an average of 34 minutes, 20 seconds, or 61.87% of participants' time was spent editing tiles, placing objects and navigating menus, making this most-used action. It was difficult to distinguish between these tasks since participants flipped between viewing menus and placing objects very rapidly. Table 3.2 shows that each time a participant began selecting and placing objects, this lasted for approximately 1 minute 35 seconds and happened an average of 21 times.

We can conclude that since object selection and placement has the highest of all time-related statistics, it is the most time-consuming feature. It should therefore be

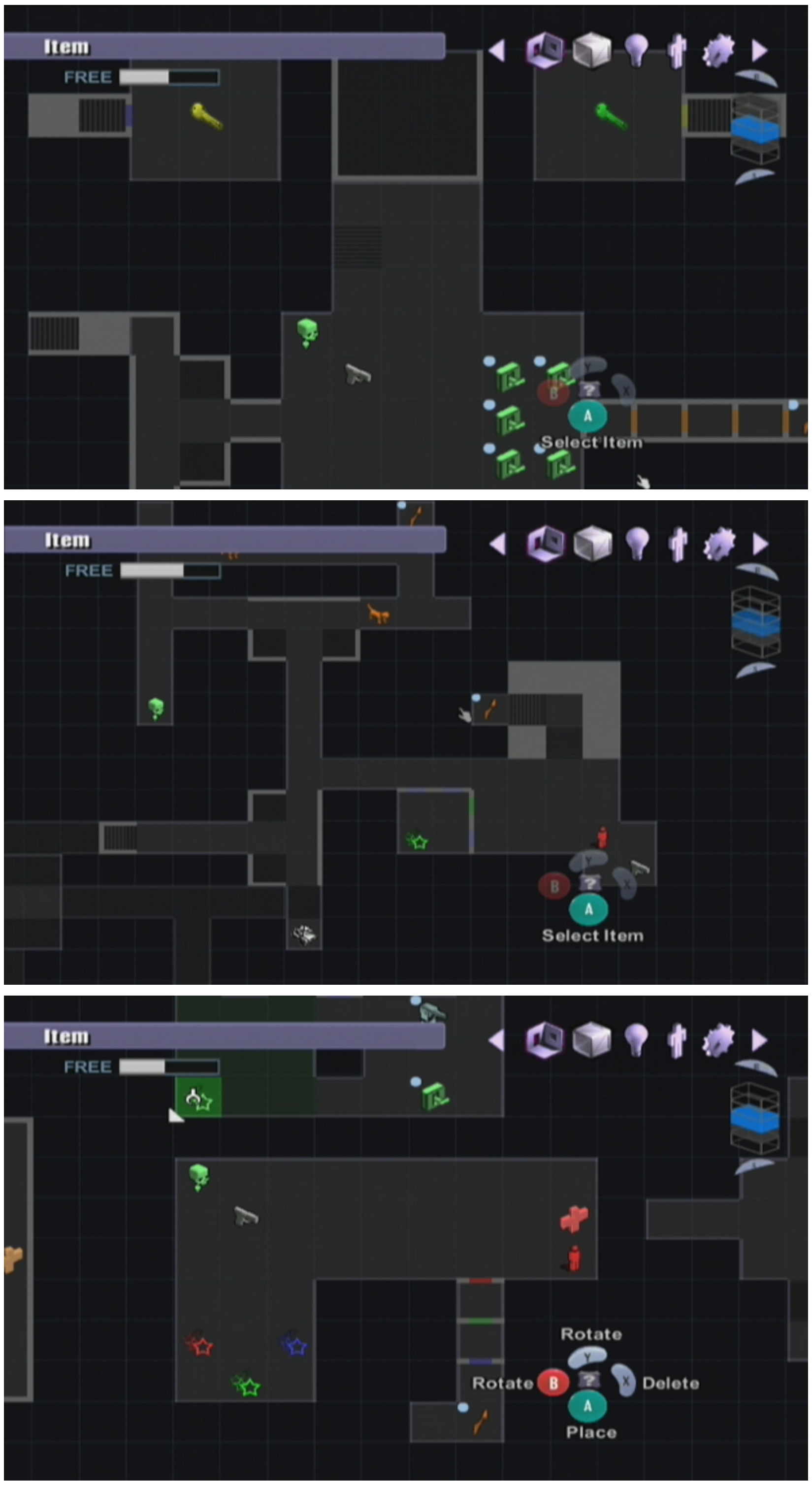


Figure 3.11: Some different map designs, showing tile layouts and placement of interactive objects. The crystals (required) and other objects connected to game logic are marked with a blue dot.

considered a high-priority feature of design interfaces, and it is important for such interfaces to save time doing this process.

3.4.3 Previewing

All participants used the in-game preview and discussed its usefulness. One of the most important takeaways is that the interactive preview feature had the highest ratings in the post-study questionnaire (M=4.8, SD=0.4). The second-highest ratings were for the quality of the interactive preview (M=4.2, SD=0.83) (Figure 3.9). In Table 3.1, we see that the feature was also the second most time-consuming, with an average of 7 minutes 35 seconds per session, or 13.73% of the total time spent in the game itself. One downside reported by most participants is the load time for the in-game preview. TSFP reloads the entire level each time a preview is started, so the user must wait for approximately sixteen seconds before being able to see the game they have created. Furthermore, upon ending the preview it takes another fifteen seconds to unload the map and return the editor. For the participants who previewed many times, this wait time accumulated and became frustrating. The average time per session spent waiting to switch to and from preview mode was 3 minutes 2 seconds, which amounted to an average of 5.42% of the allotted time. This time accounts for the time it took for the participant to initiate the preview through the main menu, the preview's load time and the unload time. If we add the time spent in-game and the time spent loading and unloading, then the average total time spent using the interactive preview feature was 10 minutes 37 seconds, or 19.15% of the total time. These measurements show that participants made excellent use of the interactive preview. They also collectively agree that it was an invaluable feature of the editor. The time measurements are consistent with the participants' written responses, which described the previewing ability as extremely helpful.

The duration of the session as a whole was extended by the time previewing the levels to accommodate for the long and accumulating preview load time. It was recorded how many times a participant used each of the previews and for how long each preview lasted. As seen in Table 3.2, participants used the interactive preview an average of 5 times per session, with an average preview duration of 1 minute 28 seconds. Generally, this was enough time for participants to navigate through the level to the part they wanted to test, see if it worked and then return to editing.

Those who used the 3D view did so much more frequently but, due to its limited functionality, and being extremely low-fidelity compared to the interactive mode, they did not use it for long. Table 3.2 shows that the 3D view was used an average of 12 times per session, with each use lasting 9.6 seconds. Only five participants actually used the feature at any point in the session. Most of the people who did not use the 3D view reported the interactive load time as an annoyance and frequently used the interactive preview just to move around their maps without any objects, or to see how the tiles were laid out. The hindrance of loading times could have been prevented if they had used the 3D view since it is mainly useful for seeing how the tiles are laid out. It is possible that they simply did not pay close attention to the control layout reference sheet that all participants were provided with, which labels the 3D view mapping, and therefore did not realize the feature existed. Alternatively, maybe these participants would rather be more immersed in their creation and the 3D view would not be sufficient; however this came at the cost of repeatedly loading and unloading the level.

Based on the feedback regarding previews, and given that it was the most time-consuming feature behind the actual construction of the level, we can conclude that some sort of preview tool is absolutely necessary for PlayTIME. This will be an important guideline for the development of a new design tool because users like the ability to make sure that what they are doing is correct and that it will work as expected. Without this ability, users would be left only to assume that their levels work and will not be able to find out until actually fully building and packaging the level. In Chapter 4, we discuss the planned features of PlayTIME, including the networked preview feature.

The need for previewing is reflected in many other software products for many different artistic industries. In gaming, engine editors such as Unity have a single button that compiles the current scene so that the designer can interact with it. Professional film software allows editors to move video and audio clips around in a graphical timeline with the ability to playback the scene in another window. Digital artists using Photoshop can preview their work by simply looking at the canvas. A preview generally lets someone know whether development is on the right track or if changes must be made.

3.4.4 Logic Operations

The third most time-consuming feature was the logic editor. In Table 3.1, we see that it was used an average of 4 minutes 20 seconds per session, or 7.61% of the total time. Table 3.2 shows that it was only used an average of twice per-session for approximately 1 minute 34 seconds. Only four participants used the feature, but it took up a lot of time because of its complexity. The small number of users and large amount of time spent using the logic editor suggests that it was too complicated to use in a limited time.

TSFP has a simplified version of this feature: the ability to directly link objects together. For example, a switch can be placed, and the object controlled by the switch can be selected through its properties menu without touching the logic editor. The tables show that ten participants used direct linking for an average of 47.7 seconds per session, or 1.45% of the total time, an average of twice per session with each use lasting 17 seconds. One of the drawbacks of direct linking is that it only links one input with one output, or one trigger with one action. For example, a switch can be pressed and it will open a door, *or* close another door, but not both. For more complex operations involving multiple triggers and actions, participants turned to the logic editor.

3.4.5 Questionnaires & Feedback

Another trend that can be observed is the consistency of feedback about the logic components of the editor. In fact, the results for all participants who used the logic features were almost identical. The users that did use logic generally agreed that it was easily understood. The feedback suggests that logic should be kept simple and be predefined for each item.

The next highest set of ratings we can see is the selection, placing and changing of items and their properties. When a user places an item in the world, it is important to be able to change the object's behaviours. In TSFP mapmaker, all items provided have predefined functionality and uses in the world. In Unity, this concept is called a 'Prefab' (prefabricated object); the idea is that the item is set-up and ready for use even before it is placed in the world. A user has the freedom to change the way an item behaves by opening its property editor and making adjustments to the provided values and options. This is a common feature in game editors such as TSFP mapmaker and Unity as it allows users to reuse the same assets without having to reconfigure their behaviours every time one is to be placed. Assets should be prepared with a few key behaviours so that designers using such a tool can create objects with some variance; not too many such that game objects become too complicated but not too few such that they are too similar. When designing a level editor, it is important to consider the fundamental idea of what each object is and what it will be used for.

Another feature that participants provided generally positive feedback for was the ability to edit their maps on multiple floors instead of just one planar level, with the ability to see through the tiles to the floor below. One of the constraints of real

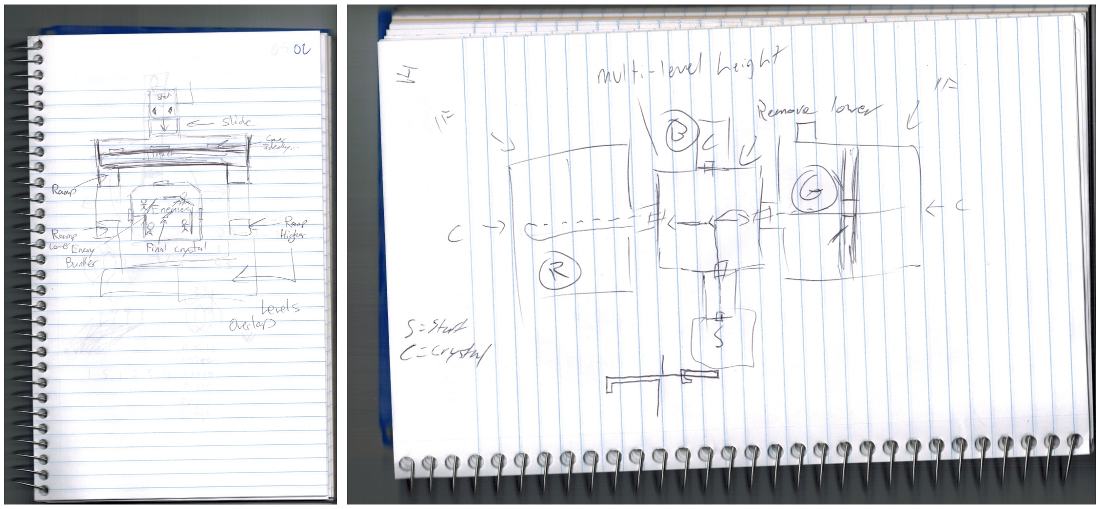


Figure 3.12: Pages in the notebook depicting puzzle designs by two different participants.

paper prototyping is that designers are restricted to a table or desk, which is a single surface. Potential solutions to multiple-floor problems would require a 3D map or multiple floor maps, which would take up space and may cause issues when working on different pieces of the map at the same time. A prototyping framework should have a 3D map or some preview of space in general so designers can see how things affect each other in space.

Overall the participants reported that they had a good experience working with the TSFP mapmaker and provided considerable feedback and constructive criticism. For more detailed questionnaire responses, see Appendix B.3.

3.4.6 Notebook

One piece of data collected was in a notebook that was added to the study to allow preliminary paper sketches before using the editor (Figure 3.12). Since part of the study was evaluating the actions performed by designers and the choices they make, a notebook turned to a blank page was left on a table in front of the participant. They were not instructed to use the notebook and it was not mentioned unless the participant asked if they were allowed to use it. Pen-and-paper sketches reflect a common practice in level design, as discussed in section 2.3; we wanted to see how many people would use this technique to assist with the design of their maps.

Watching the video recordings showed which participants used the notebook and when. Nine of the participants used a page to draw out their maps and take notes on their puzzle ideas before creating the level in the editor. As seen in Table 3.1, the average time spent drawing in the notebook was 3 minutes 17 seconds, or 5.97% of the total time. Typically the notebook was used at the very beginning of the session and once more later on. Out of the nine people who used the notebook, only seven actually referred back to it while building the level. Participants referred to their page an average of four times after the initial drawing. The time spent referring to the notebook averaged 28 seconds per session, or 0.95% of the total time.

Some of the final designs incorporated elements of the levels originally planned in the notebook. Most of the sketches were incomplete or very limited, so it was difficult to tell if the final levels accurately matched intended designs. It is difficult to conclude whether participants found the notebook useful since, unfortunately, there was no question regarding the notebook in the post-study questionnaire. However, after each session some verbal questions were asked concerning the notebook. Most participants reported that they found it somewhat useful; in the end they improvised most of their designs in-editor. When the people who did not use the notebook were asked why not, they stated they either did not notice it or were unsure if they were allowed to use it.

3.5 Summary

By running the designer actions experiment, we have learned about what experienced designers like and dislike about unfamiliar prototyping software. More importantly for PlayTIME, the study has shown which actions taken by designers are most prominent in a basic level-building exercise. TSFP's mapmaker provided the participants with an unfamiliar, constrained design tool set whose features will be reflected in PlayTIME.

The experience has provided us with feedback, ratings and time measurements about features that will help guide the development of PlayTIME. We also learned about which features will be substantial and those that will be helpful. The three most important features used by all participants in this study were object placement, previewing and changing object properties.

We learned that users collectively spent the most time selecting objects from menus and placing them in the world: as evidenced by the performance statistics, an average of 61.87% of their time was spent picking objects from a menu of pre-built assets and placing them in the scene. Therefore object placement is the most important requirement for level design in PlayTIME; this is how the interactive world is actually constructed. Users spent an average of 19.7% of their time previewing their level; this includes the long preview loading times. It was also rated highest, on average, when asked about feature preferences. Therefore the ability to preview with PlayTIME will be necessary. There are some planned methods of previewing for PlayTIME to satisfy this requirement, but whatever system that PlayTIME is paired with, whether it is Unity or another editor-based engine, must have a way to preview efficiently. Third, users collectively spent 7.42% of their time modifying the parameters of pre-defined object behaviours. Therefore objects in PlayTIME must have simple properties and behaviours that can be changed to interact with the game environment in different ways and enable creative ways to use these objects.

From running the study we gained a perspective on how users experience an unfamiliar game development tool. Our findings were used to guide the development of PlayTIME, described in Chapter 4.

Chapter 4

PlayTIME

4.1 Introduction

This chapter introduces **PlayTIME**: a **T**angible **I**nteractive **M**edia **E**nvironment for Game-**Play**. It is a conceptual development environment and work flow for designing interactive content.

The goal of PlayTIME is to utilize tangible user interfaces to complete game design tasks that would otherwise be done using traditional computer-based methods, while simultaneously fostering collaboration and teamwork. The system aims to merge forms of physical design and prototyping with digital design and prototyping.

In this chapter, we briefly discuss traditional scenario design techniques, including both physical and digital design spaces. We then discuss the conceptual framework including PlayTIME, its proposed features and components, and the current design and implementation of PlayTIME.

4.2 Traditional Scenario Design

4.2.1 Physical Prototypes

In Chapter 2, we introduced physical prototyping and paper prototyping as common methods of preliminary game design. The process of building a paper prototype involves collecting everyday materials and using them to map out a scenario. The scenario is guided by the rules of the game. This design process is extremely flexible because the objects can be easily picked up, handled and replaced to suit the needs of the designer. This allows people responsible for designing game play to test mechanics in different situations. Paper prototypes are non-digital versions of the final digital game that may be played like board games.

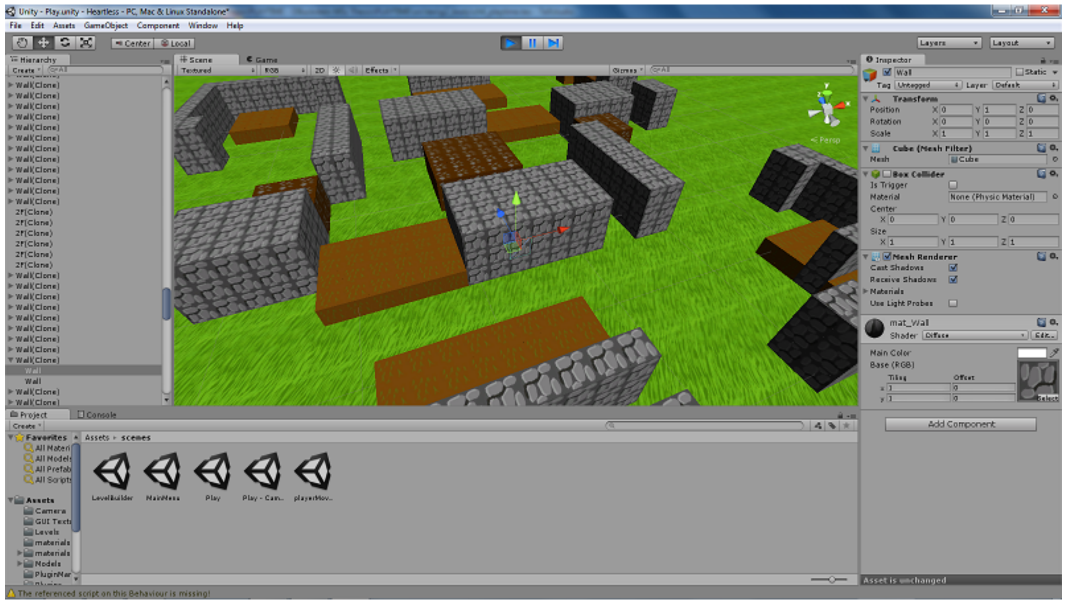


Figure 4.1: An example of a digital prototype being constructed in Unity. This view shows a maze of stone and wood blocks.

With this comparison, non-digital game design often becomes a simulation of how the game is to be played; the act of physically *playing* with everyday tangible objects and *testing* game ideas with these objects effectively simulates *creating* and *designing* the game. Playing and creating primitive forms of interactive content are similar processes, so it is possible to treat the process of design more like play.

After refining some of the rules for a game, responsibilities are handed off to developers who specializes in digital prototyping to build a digital representation of the game. Paper prototyping is often revisited to develop new pieces of the game and its rules.

4.2.2 Digital Prototypes

Most of the available software in the domain of digital game development provides the user with a graphical interface with many adjustable fields. Editors are often built on an integrated game engine that gives both high-level and low-level access to game play elements such as physics, animation and graphics. The graphical editor allows designers to simply click-and-drag objects into place without actually editing any code. Some of the most popular game engines with editors that behave in this fashion are Unity [1], Unreal [34] and CryEngine [35]. This is only a small sample of the many digital game development tools that are used in the industry today.

Figure 4.1 shows an example of a digital prototype being constructed in Unity's editor. In this example, we can see that a basic world layout exists, but it is difficult to describe what the game rules are. Object behaviours are controlled by scripts and simply moving them in the editor does not have any effect on the game play. Editors such as Unity rely on mouse-based actions, such as clicking and dragging to bring digital assets from storage into the game environment. They also provide text fields for users to type values into.

As we learned from the designer actions study described in Chapter 3, the ability to preview is a most desirable feature in a digital game editor. One of the problems with today's editors is the inability to simultaneously build a level and preview it. For example, Unity allows users to hit a play button and this will start a preview. However, to further develop the scene, users must stop the preview, continue working, and start a new preview when the scene is ready. Unity allows work to be done while the preview is running in a separate window, but changes made in this state are not saved as part of the actual scene and will be reverted upon ending the preview. This can lead to extra work since designers must write down experimental values or find an editor extension to save the changes made while in preview mode. The reversion of the scene state can also be very frustrating to new users who are not aware that stopping the preview undoes all changes. To summarize, it can be difficult to build and run a digital prototype. In contrast, developing and manipulating a paper prototype can represent actual game play and testing.

4.2.3 Bridging the Gap

There is a clear divide between physical and digital game prototyping. In traditional design practices, the act of creating a paper prototype creates a version of the game that is entirely separate from the final product. Paper prototyping creates a tangible version of the game that designers can use to refine mechanisms and rules, but these designs still need to be handed off to programmers to create the digital version. Even with strategies like iterative design, both physical and digital prototypes must be refined and revisited separately. The system discussed in this chapter aims to bridge the gap between physical and digital prototyping by automating the digitalization of physical prototyping. This way, designers can practice the earlier stages of design and this carries directly over to a working digital version of the game.

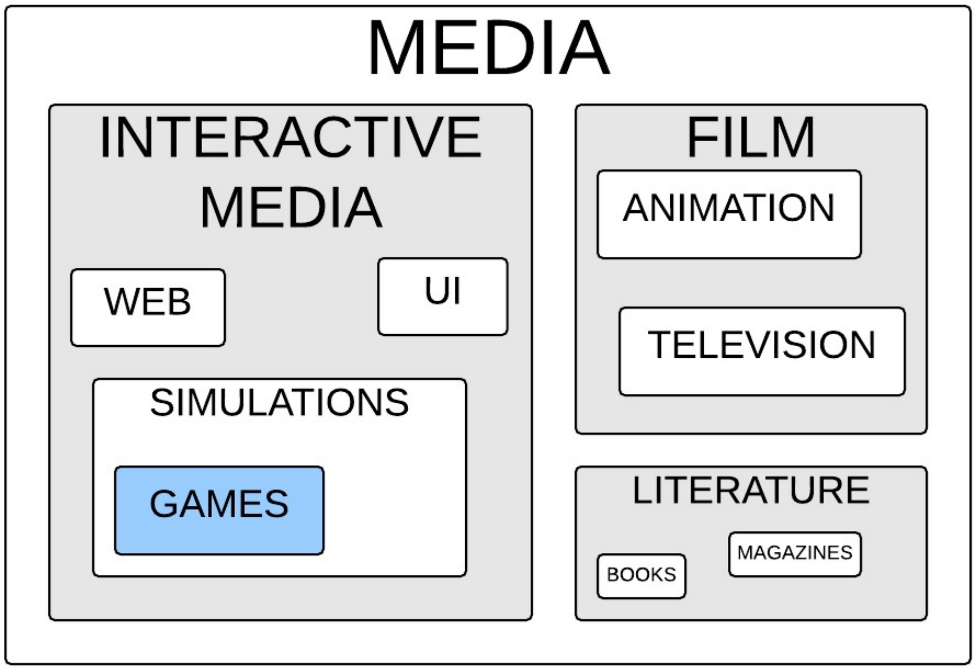


Figure 4.2: An overview of different types of media.

4.3 Tangible Interactive Media Environment

The system proposed in this thesis ultimately digitalizes the practices of physical prototyping of interactive content and media. In other words, a designer physically creates their environment while their actions are interpreted by a digital editor that creates the digital version of their design. Since physical interactions can be digitally interpreted using tangible user interfaces, we call this type of system a **Tangible Interactive Media Environment**, or **TIME**.

TIME is a conceptual development framework that can be applied to different domains of media. The motivation behind TIME is to be able to bring visions of digital media to life. Figure 4.2 shows examples of different types of media. Tangible interaction could be useful in the creation of interactive media spanning simulations and games, web design, user interfaces, and more. The framework can also help directors and producers of non-interactive domains, such as film and animation, realize their visions. This may be extremely helpful in the cinematography department, where planning out camera movements relative to the scene plays a big part in production. Literature is also shown in the diagram. Although books and magazines are physical media, the TIME framework may also be useful for authors to represent their ideas in real life and get past the hurdle of writer's block.

4.4 PlayTIME

The core theme of this thesis is based on an implementation of TIME in the domain of game design and prototyping. Here we introduce **PlayTIME**: a class of TIME systems used specifically for developing gameplay. The goal of the system is to allow game designers and interactive scenario creators to utilize tangible objects to build a digital version of their vision. The different components of PlayTIME will be responsible for digitalizing physical actions such those taken while building and playing a paper prototype, and natural body interactions. Stations will be specialized in different game design activities, and will determine relevant meanings of actions and tangible objects.

One of the visions of PlayTIME is to foster the collaborative nature of game design by allowing people of different disciplines to work on their respective tasks while sharing all assets in a common project. Assets created at different stations will be usable, manipulable and replaceable at different stages in the development pipeline. Furthermore, stations that use assets from others should not be hindered by an incomplete resource. Another goal of PlayTIME is to support stand-in objects and facilitate on-demand updates between stations.

For example, a character can be created by the modeller, passed on to the animator to be given an idle stance, and finally added to a scene by the scenario designer. A simple cube or cylinder can be used in the character's place while it is being initially created. The proposed stations and the development pipeline are described in detail in section 4.4.1.

One of the problems with today's digital development work flow is that people spend a lot of time in front of computer, using a mouse and keyboard. Tasks are often delegated from the initial physical designers to team members who specialize in building digital prototypes. When complete, PlayTIME should be a valuable tool that unites every stage of design with actual development, from initial paper prototyping, play testing and iteration to a playable digital product. The primary vision is to work towards a collaborative work flow that allows us to treat the process of developing a game more like experimental play. With this mindset, ***playing <u>is</u> creating***. Game designers, and later creators of other media, will benefit from the collaborative and productive environment described in the following sections.

4.4.1 System Overview

The concept of PlayTIME is a collaborative game design environment made up of *stations* that facilitate different aspects of game creation. We aim to provide developers with exciting hands-on and natural ways to apply their skills towards a project in an integrated development environment.

The stations of PlayTIME are grouped into two general categories: asset development and design. Asset developers, or artists, are responsible for putting together the raw resources used in production. Traditionally, creating resources such as models, animations and textures is a very time-consuming process using a mouse and keyboard. Existing TUIs, such as tablets, help artists simulate painting digitally. One of the goals of PlayTIME is to use more technologies across game development disciplines to ease the process of asset creation.

Designers use the available assets provided by the artists to create and experiment with game mechanics, and build the game's world and interactions. Essentially, the designers work together to arrange the raw resources and build the actual game.

Tangible user interfaces (TUIs) and natural user interfaces (NUIs), discussed in Chapter 2, are used to allow *physical* interactions with a system. TUI-based systems use *physical objects* to interface with a system or application, and NUI-based systems interpret *movements and gestures* that are symbolic to the system. With PlayTIME, the plan is to leverage tangible and natural technologies at each station, straying away from the traditional mouse and keyboard and providing developers with a conductive and comfortable environment.

4.4.2 Development Pipeline

Figure 4.3 provides a graphical view of the relationships of the PlayTIME stations within the *development pipeline*. This is a map of how data flows throughout the entire PlayTIME system, from initial resource creation to the final output built using a game engine.

At the heart of the entire operation is the PlayTIME server. The server keeps track of all the stations' outputs and listens for data requests. This is the main centre for collaboration; all digital communications between stations happen through the server. This way, all stations have access to whatever data they need, whenever they need it. Details on the communication protocol between stations can be found in section 4.5.1.

Development with PlayTIME begins with the asset developers (section 4.4.3), who create the raw art and resources to be used in the game. The resources are sent to

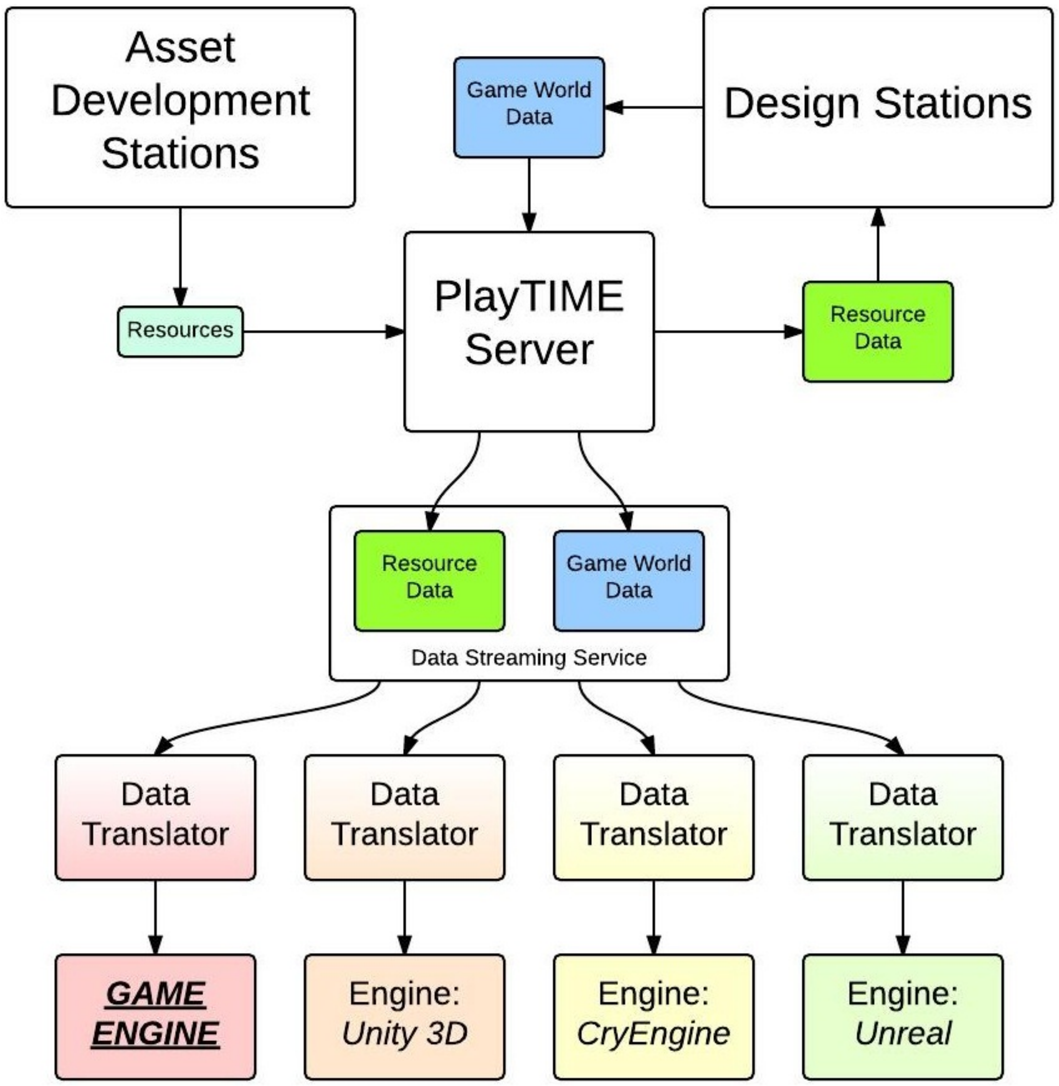


Figure 4.3: An overview of the PlayTIME development pipeline. This system diagram shows the groups of developers involved with game production and how assets go from initial creation to being used in the final product built with a game engine.

the server, where they are stored as-is or converted to a data format that can be read by other stations. The data is made available to design stations (section 4.4.4), where it is used to construct pieces of the game world. The game world, scenarios, logic, behaviours, etc. are packed up and sent back to the server. At this point, we can see that the server knows all about the game, its environment and its resources to form an organized, collaborative and shareable game development project.

Parallel to development, PlayTIME is also designed to have real-time streaming of the data into existing game engine interfaces. While assets are being created and added to scenes, the game is being assembled into its final form. This enables

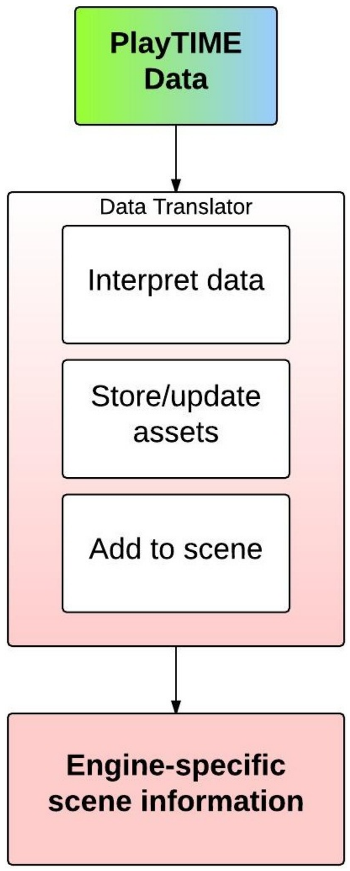


Figure 4.4: A close-up of the data translation process, bringing PlayTIME asset and scene data into a game engine.

developers to preview the game as its parts are being worked on. The streaming service is responsible for frequently asking the server if anything new has happened: asset revisions, scene changes and behaviour modifications are all in the streamer's interests. If the streamer detects a relevant update, or if the server says it is time for an update, then it will pass data to connected engine-based PlayTIME projects via data translators designed for each engine.

Unity, for example, allows users to create DLLs with external code, and a script to access these external functionalities; this is called a plugin. A data translator will generally be an extension or plugin for an existing game engine. Figure 4.4 shows some of the steps in the data translation process, which happens per engine.

The incoming data is received by a function designed to decode the information and figure out what it is. It's job is to answer questions such as, "What piece of data did I just receive? Is it a model? Is it a texture? Is it a scene update?" Once the data is understood, it needs to figure out what to do with it. If a scene change happens, for example, if an object was placed in the map, then the data will be an

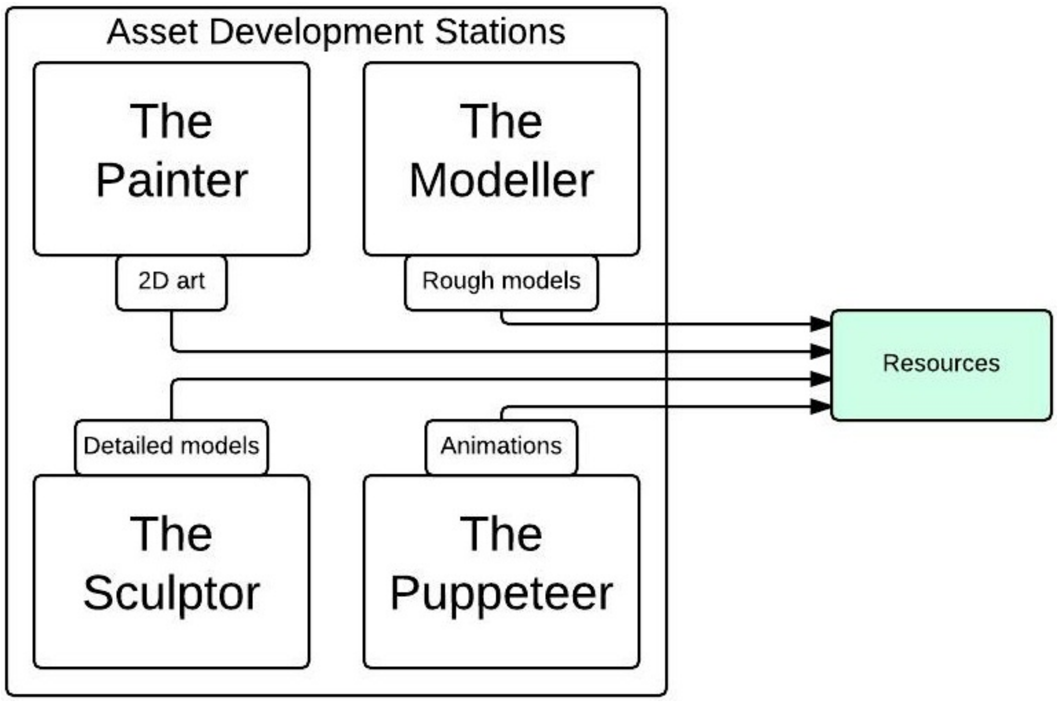


Figure 4.5: A close-up of the asset development stations of PlayTIME.

encoded description of what happened. An example of this in English could be: "A barrel was added at position (320, 240), rotated 90 degrees." After interpreting this, the plugin is responsible for telling the engine to add a barrel model or its stand-in to the scene. If the barrel model is updated, then the already-placed barrel will change its appearance as the updated asset is received. An updated asset, such as the barrel model, can simply be saved on the computer in a format that the engine understands and is currently using to represent the object in the scene, if it exists.

The result of these processes is a scene file or set of scene files that are editable and preview-able within a game engine.

4.4.3 Asset Development Stations

The following roles are responsible for developing the raw content to be used in the final product, which is a collaborative effort between all the roles. These team members are the artists. See Figure 4.5 for a graphical overview of these stations.

The Painter

PlayTIME's Painters will be responsible for drawing 2D textures to be applied to models in the game world. A Painter may also be responsible for drawing environments and landscapes. Digital artists often use a tablet because it digitally simulates drawing on paper with a pen.

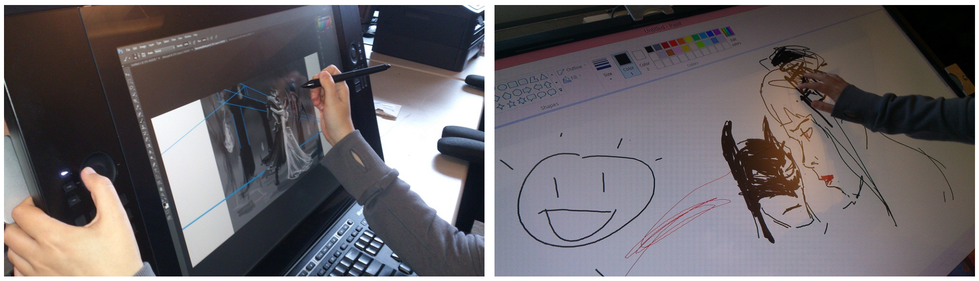


Figure 4.6: The Painter uses various input devices to draw pictures. On the left is a Cintiq tablet with art in Photoshop, and on the right is a touch table with some scribbles in MS Paint.

Figure 4.6 shows a Painter using a 24-inch Wacom Cintiq pen tablet [84], which provides the digital artist with a large, adjustable drawing space. Painters may also use a touch table, a NUI that uses finger presses and movements as inputs. Tablets like the Cintiq are effective TUIs since they offer a clear pen-and-paper metaphor, and they often have buttons to change the properties and dynamics of virtual paintbrush. A table with multiple touch inputs also allows multiple people to paint simultaneously. A model of the Cintiq exists that allows both pen and finger touches to be used.

The Modeller

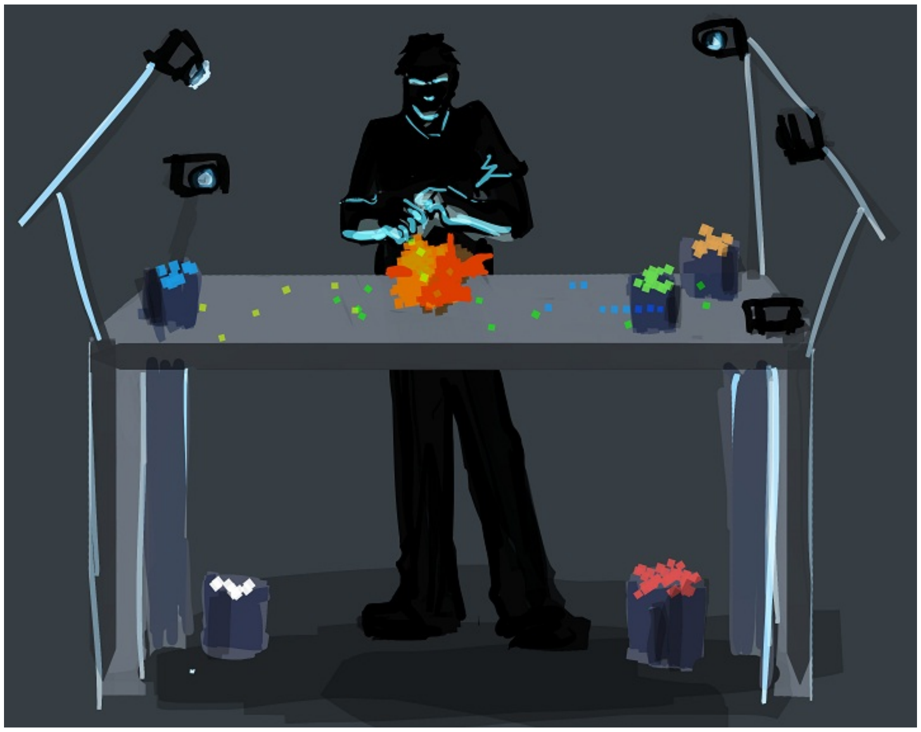


Figure 4.7: The Modeller uses Lego bricks to construct a primitive 3D model.

The Modeller will use tangible toys and objects, such as Lego bricks, to construct primitive models of objects to be used in the game. Figure 4.7 is a concept sketch of the Modeller using Lego bricks. When ready, the rough models can be digitalized using a 3D scanner and passed forward to the Sculptor to touch it up.

Tangible toys like Lego bricks afford to be assembled freely, taken apart and replaced quickly. They are inexpensive, numerous and easily acquired. Furthermore, having a wide variety of bricks means the size and complexity of a model will not be an issue.

The Sculptor

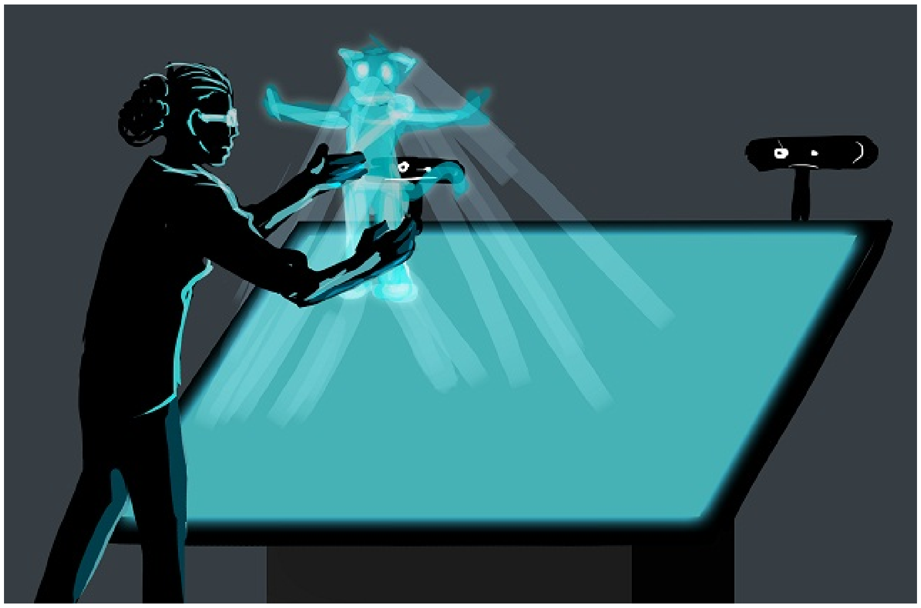


Figure 4.8: The Sculptor virtually refines a model using hand gestures tracked by Kinects.

When the Modeller completes a rough design for an object to be placed in the world, it is time to make the model look pretty. The Sculptor's responsibilities include smoothing out the rugged models and applying fine details. The gestures tracked by devices like the Kinect and Leap Motion allow the Sculptor to simulate clay sculpting. Projects linked to PlayTIME [85] [86] [87]discuss algorithms that make virtual sculpting possible. The works emphasize the use of VR technologies, such as the Oculus Rift [88], and NUIs, such as the Leap Motion and touch screens, to provide users of the system with a virtual stereoscopic 3D view of the model.

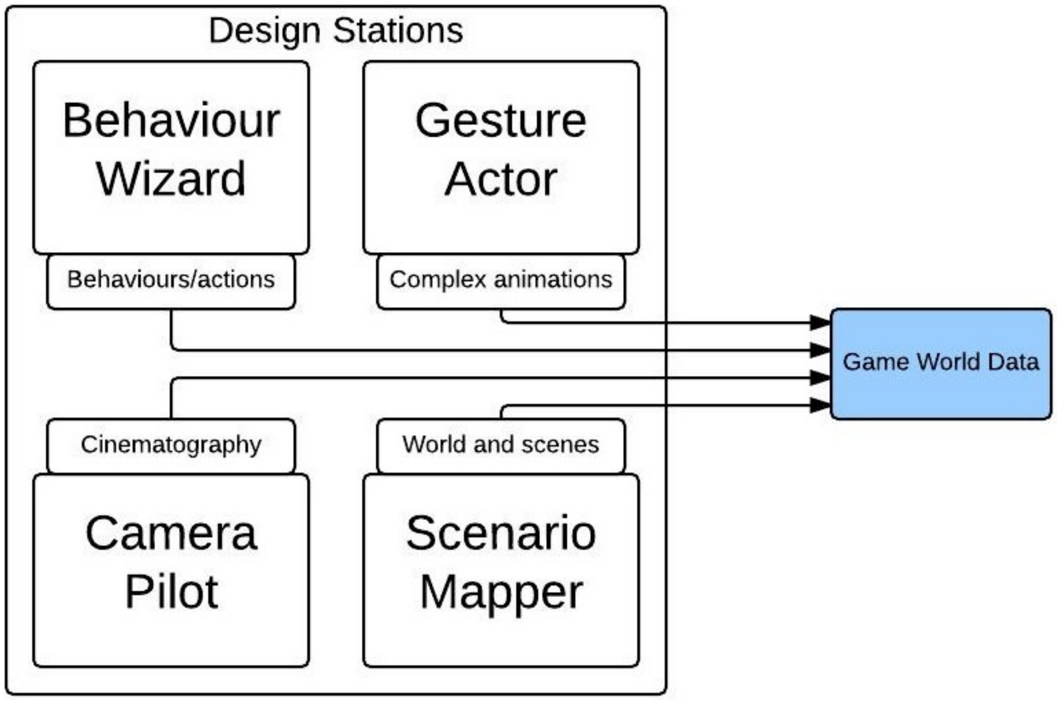


Figure 4.9: A close-up of the design stations of PlayTIME.

The Puppeteer

Just as a real puppeteer may tie strings to their fingers and use their digits to control the movements of a marionette, PlayTIME's Puppeteers will do their work in a similar fashion. Devices like the Leap Motion are capable of tracking hand and finger movements; this makes simulating string pulls possible. The Puppeteer will use these gestural inputs to manipulate a character's skeletal structure, designing animations for the characters in-game.

4.4.4 Design Stations

The following team members are responsible for assembling the game using the resources described above. They focus on the rules of the game, the layout of the world and behaviours of objects within. See Figure 4.9 for a graphical overview of these stations.

The Behaviour Wizard

Objects in any game development pipeline must be open to new behaviours, otherwise it would be difficult to explore new game mechanisms and create new designs. Typically, game engines allow some form of programming or scripting for a coder to apply new, custom behaviours and properties to an object. The role of PlayTIME's

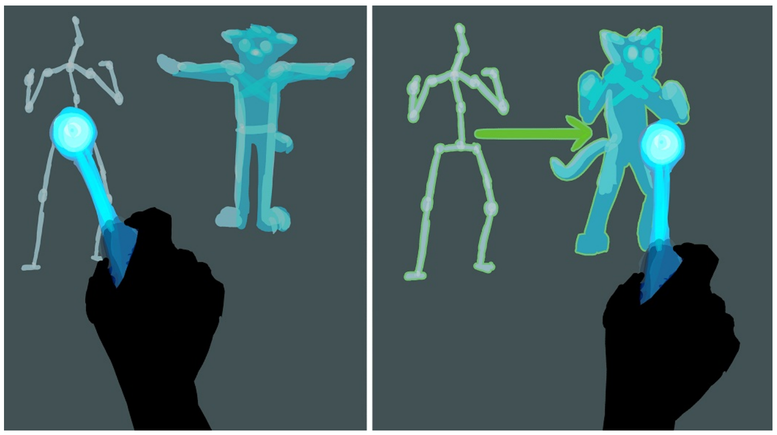


Figure 4.10: Behaviours and object linking can be applied by waving a 'wand' between objects.

Behaviour Wizard is analogous to the game play programmer. This team member's job is to create new behaviours by performing symbolic actions using tangible objects. For example, if a character is to follow a path, the Wizard can *show* the system how this works by holding a marker and tracing out the path that the character is to follow; the character will then replicate this path in-game. The Wizard can also determine how objects in the world are to interact with other objects by moving a marker between the objects to link them; each object will have a set of interaction options which can be selected when linking with another object.

One way of implementing behaviour linkage would be to use a custom 'wand' controller with either a fiduciary marker, or existing hardware like the PlayStation Move. Figure 4.10 depicts a wand being used to link a skeletal animation with a mesh; this adds an animation behaviour to the model so that it can be more lifelike in-game. In the left image, the user points the wand towards a posed or animated skeleton to select it, with an unbound character waiting in its default pose (T-pose). On the right, the user has gestured the selected skeleton towards the character, thus binding the mesh ('skin') to the skeleton ('bones'); the character can now assume any pose or animation related to the skeleton. The Wizard metaphor comes from the gesture of waving a wand between objects, and how linking objects with PlayTIME will be so easy it will seem like magic!

The Gesture Actor

While the Puppeteer is responsible for some animations, making complex movements can be a full-body task. The Gesture Actor is responsible for playing out motions and gestures that may be seen in a game, such as fighting moves or walking and running.

Figure 4.11: A Kinect is used to capture full-body gestures to create animation sequences.

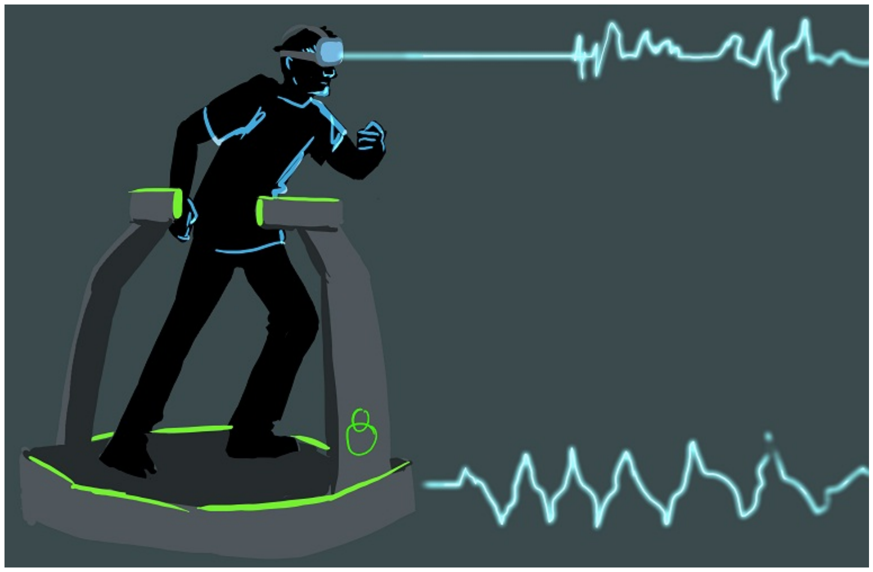


Figure 4.12: The Virtuix Omni treadmill can be used to get the feel of first-person movements within the world.

Figure 4.11 shows an actor recording a punch using a Kinect, while a virtual character simulates the same move on the display. The actor may also tune player movements within the world using the *Virtuix Omni* [89], a treadmill that tracks the position of the user's feet, while seeing the world in a first-person perspective using the *Oculus Rift* [88] (Figure 4.12).

The Camera Pilot

The Camera Pilot is the team cinematographer. This person is responsible for setting up camera movements within the scene for in-game cutscenes and cinematics. Markerless AR technologies like PTAM [65] can be used to keep track of the camera's movements in a real environment, while simultaneously showing a digitalized path within the virtual world. The Pilot is similar to the Wizard, but the role specializes in the film aspects of game development.

The Scenario Mapper

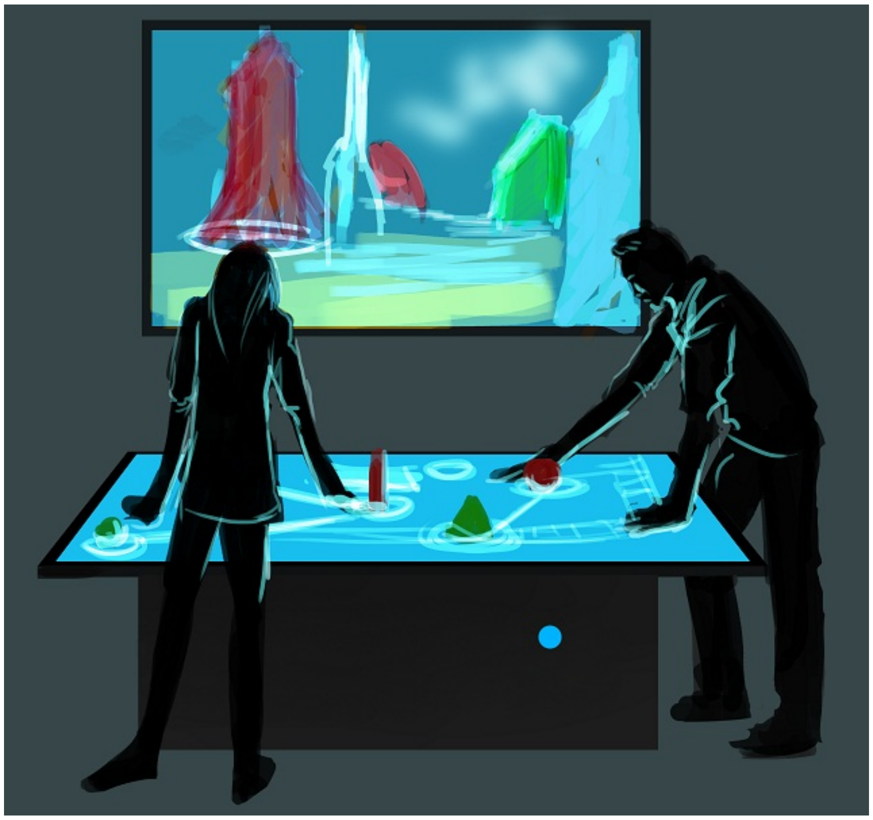


Figure 4.13: Scenario Mappers lay out objects in a game world using a touch table. An in-game preview of the world is displayed on a screen.

The Scenario Mapper is responsible for building the game world. The tasks performed by this role are analogous to the entire physical prototyping process; this station is the focus of this thesis, and the basis for the development of other stations.

Using a flat tabletop surface with touch input, people in this role take existing models or stand-ins and place them in the game world. The digital assets are represented by tangible objects with AR markers attached. These tangible objects are moved around to suit the needs of the game mechanisms and rules of the game. This is where the true meaning of 'PlayTIME' lies: while the designers play around with the physical objects, the object states are digitalized and the game is created. This station is also geared towards collaborative work, where multiple designers can share the tabletop space and develop the world together. Figure 4.13 depicts the vision for this station: two designers stand over a table display and discuss the placement of objects in the world. While developing, an fully playable in-game preview is displayed in the background. One goal of PlayTIME is to have the preview of the game update in real-time as the scene is developed.

AR markers are useful for tracking the position and rotation of the object, and they can also be used to encode information, such as what the object actually is. For example, the marker representing a basic model can be placed and moved around. If the model is edited, the representation of the model in the scene and in the preview can be automatically updated to the newer version without halting development.

Another responsibility of the Scenario Mapper is to apply behaviours and components to objects within the world. An example of a behaviour application is once a path has been traced, it can be passed to multiple characters who should follow the same path. Components include things like textures and animations; these can be taken from storage once the respective artists create the assets and applied directly to models using the wand interface discussed in *The Behaviour Wizard*.

One of PlayTIME's long term goals is to be able to simply place a marker next to an asset, alerting the server that this marker now holds the data for that asset. The marker can then be transferred to any station and use freely.

Scenario Mappers in the Current Implementation

The current implementation of PlayTIME is limited to a *prototype of the Scenario Mapper station*. As discussed below, we have prepared a specific scenario based on an existing project; this skips the entire resource development pipeline and brings us right to the final assembly, allowing us to focus on one role.

A Scenario Mapper working on the existing project uses AR markers with predefined meanings related to the game elements: one marker represents the player token, which is necessary for players down the line to be able to control an avatar in-game; two other markers represent different enemies that can be placed; another marker assigns a specific behaviour to enemies; and another marker modifies a specific parameter of the behaviour.

The current functionalities provided to Scenario Mappers facilitate the primary needs identified in Chapter 3: placing objects, previewing, and modifying behaviours.

4.5 Implementation Details

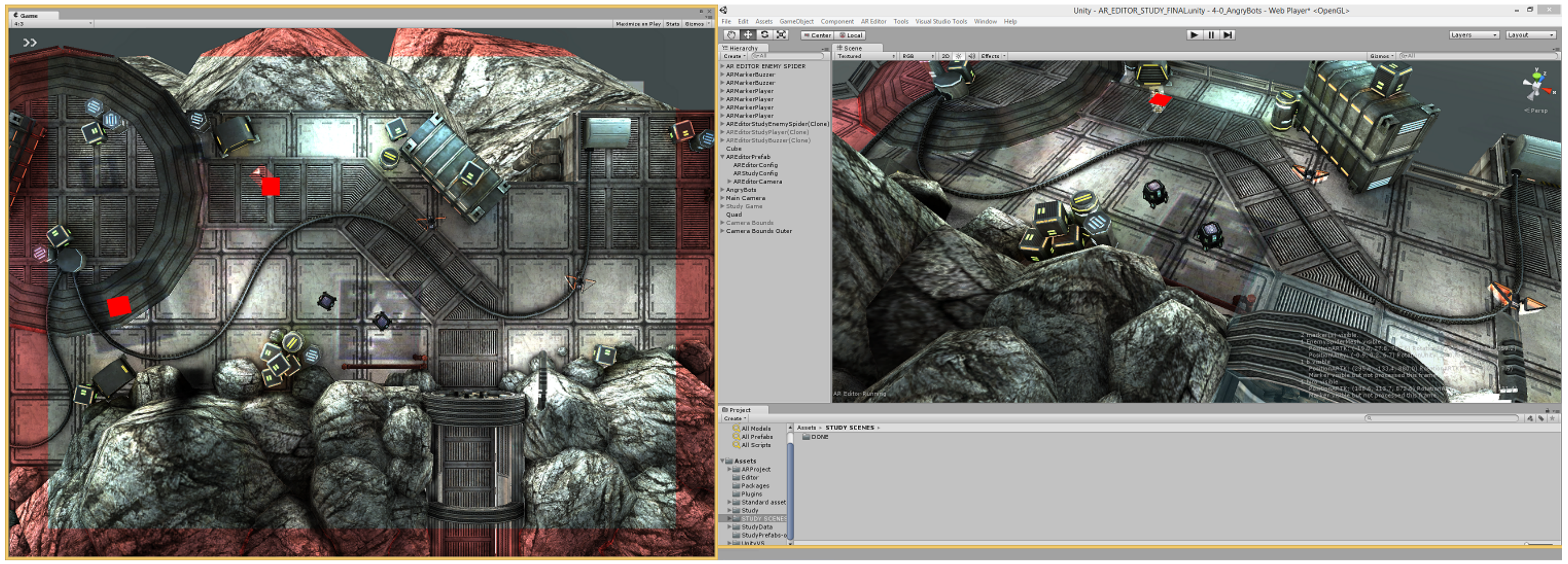
The main focus of the current implementation is the scenario design station, operated by Scenario Mappers, which will make the most use of tangible objects to position objects in the world and manipulate their behaviours. The current implementation also explores parts of the behaviour-building and cinematography stations.

We implemented a prototype data translator (plugin) for Unity that encapsulates ARToolKit [63] for marker detection. The plugin operates through an editor-extension script: Unity allows special scripts to override the way its editor behaves. This allows AR to be operated in real-time without having to run the game. AR markers are directly assigned to affect existing assets within the Angry Bots project. The markers are printed out and attached to wooden blocks to make them easy to handle. Moving markers in the real world causes their assigned objects to be manipulated in the digital version, in the editor. Thus, we have created a tangible user interface for AR-controlled Unity interactions.

Figure 4.14 shows the plugin being used within the Angry Bots project in Unity. The top image shows the editable scene within Unity with all of the adjustable parameters such as position, rotation and scale: the properties affected by moving tangible AR markers. The bottom window shows a preview of the game running simultaneously. When the play button is pressed, editing mode stops and the game can be played. The plugin also projects a semi-transparent image the AR-tracking camera's view on to the scene. This is intended to help the use see how their physical actions will affect the scene. The transparent red border shows the user the limits of the area their camera can see.

The scenario used for the current implementation is a modified version of a Unity sample project called *Angry Bots* [90], a fully-operational project freely provided by the developers of Unity. This project gives users a detailed exploration grounds to learn about Unity objects and scenario design within the editor. It is a package complete with an expansive level in a game with fully textured models, working behaviours and a scene layout. Having these pre-compiled assets makes testing different interactions within Unity possible. We have modified and reduced the scope of the original Angry Bots project allowing future users, the Scenario Mappers, to focus on a small portion of the game world.

For the purposes of this thesis and the experiment in Chapter 5, we have implemented only a few key features of Unity using the new AR-based TUI. A selection of



(a) The preview window used to display the PlayTIME view. A(b) The editor view, which follows the active marker, in this case the spider transparent overlay of the camera's current frame shows users howplacement marker. The selection marker is also visible.
the real world aligns with the virtual world. The red border indicates
the area that is beyond the AR detection camera's vision.

Figure 4.14: A dual view of the current implementation of PlayTIME's tangible interaction with AR in Unity. The user sees the PlayTIME view on the left and Unity's editor on the right. Here, a spider is being placed at the centre of each view. The red border shows PlayTIME users the area within the camera's field of view, and a transparent overlay of reality shows users how their actions coincide with the virtual world in Unity's editor.

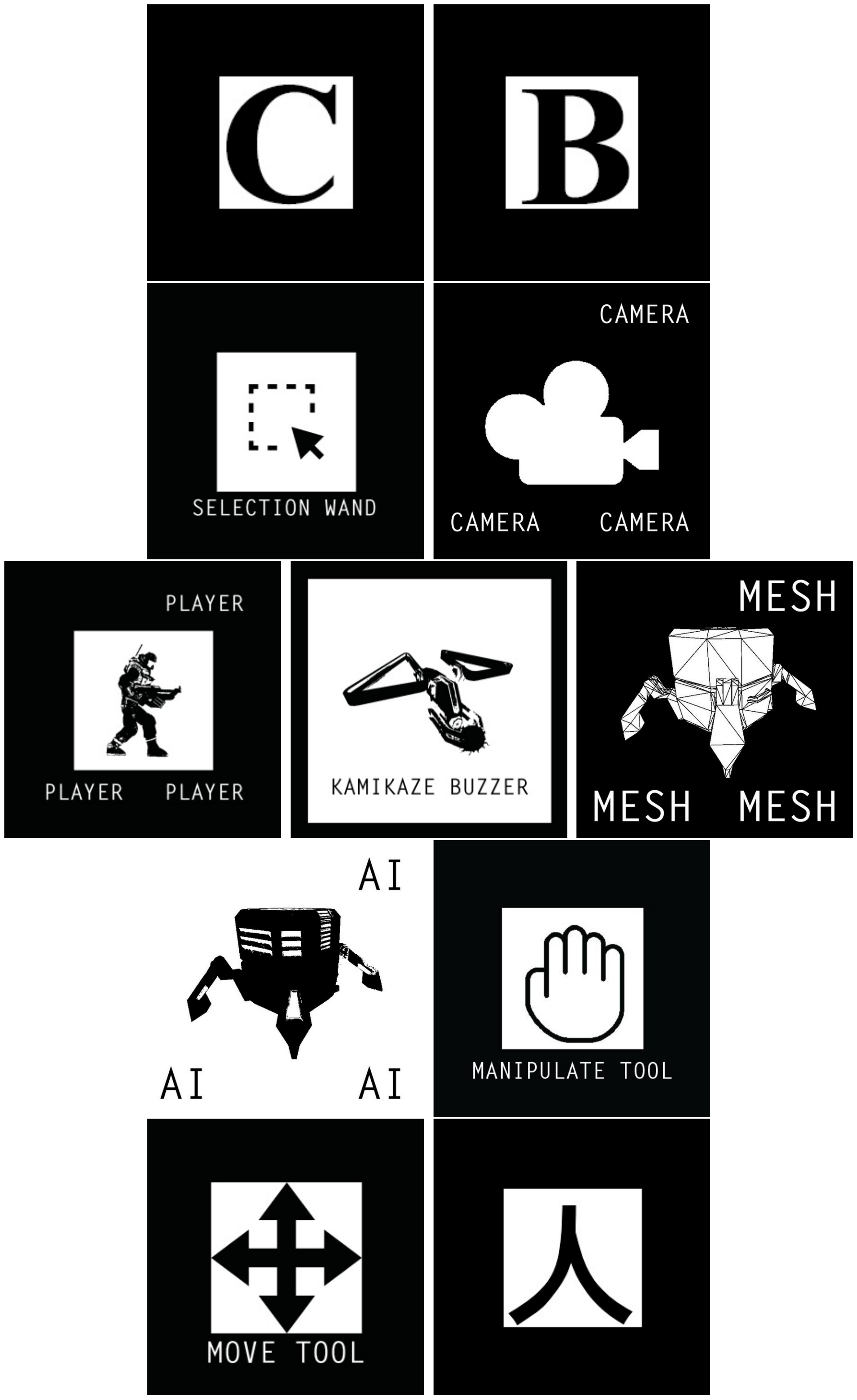


Figure 4.15: A collage of the markers used in the current implementation. From top to bottom: buttons to simulate mouse clicks and key presses ("Confirm" and "Back"); selection and camera pan; the object placement markers (player spawn, flying buzzer enemy and exploding spider enemy); AI behaviour application and modification for spiders; move selected object, and camera AR calibration.

core game elements can be placed and manipulated within the game world. Here is a list of the current functionalities of the Unity PlayTIME extension:

Select/deselect object The user can 'point' to an object using a marker to select it for manipulation.

Move selection With an object selected, the user can change its position and orientation.

Press buttons Some AR markers behave as buttons: waving your hand over them simulates a click of the mouse or key press.

Move editor camera The view of the scene in the editor can be changed by moving a marker that controls the camera.

Technical calibration The user may need to fix the AR tracking if the camera is bumped. This feature uses an AR marker to correct the tracking.

Player placement The user can place a starting position for the player. This is the location in the world where the player will begin the game.

Enemy placement Angry Bots has three types of enemies; we allow two of the simpler ones to be placed in the world without any behaviours.

Apply enemy behaviour An enemy behaviour is attached to an enemy object using a gesture similar to waving the Behaviour Wizard's wand.

Modify enemy behaviour To test the ability to change a behaviour ('code') once it has been applied to an object, one of the enemies' attack radius can be modified by sliding a marker closer to or farther from the enemy.

The markers used for these actions can be seen in Figure 4.15.

Since the purpose of this thesis is to evaluate the feasibility of tangible interaction for PlayTIME, the current implementation supports tangible placement of objects in a digital environment. The focus is on the *scenario design* portion of the overall project, described above, which we decided would be simple enough to appropriately determine the feasibility and usefulness of TUIs for the overall system.

4.5.1 Unused Features

Tile-Based Editor

A tile-based level editor was started to allow designers to assemble a rough layout of the game world using tangible AR blocks to represent tiles. This grid-based system was designed after TSFP's Mapmaker (described in Chapter 3) and would, similar

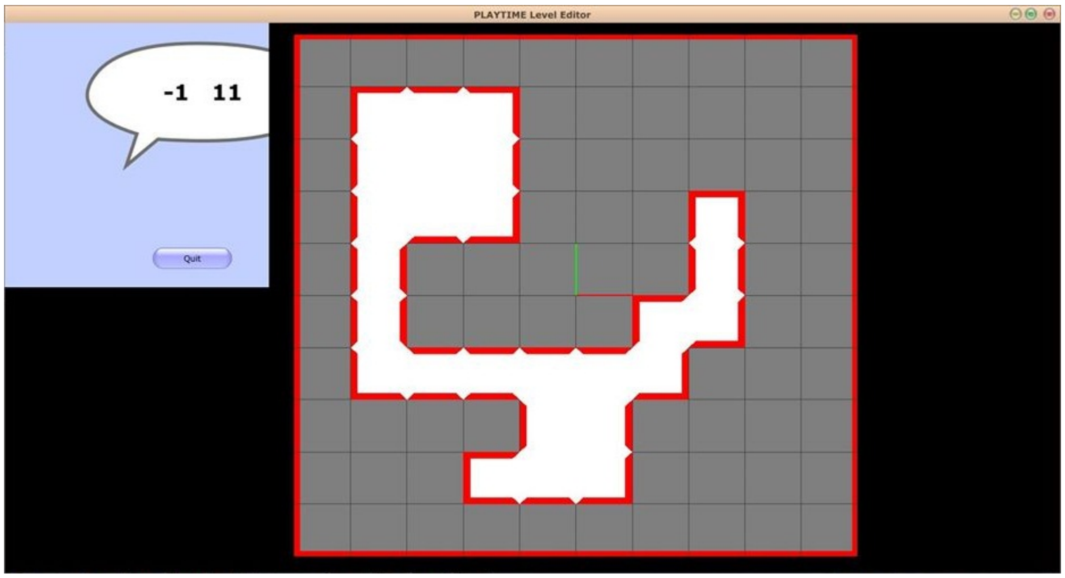


Figure 4.16: An example of the tile editor. Placed tiles are white, with automatically-generated walls in red.

to TSFP, feature a small, pre-constructed set of tiles and interactive objects. The design was experimental and ultimately put aside in favour of Angry Bots, a game world that is already stable and comes with a variety of features. Figure 4.16 shows an early implementation of the editor. The tiles placed are white, and the walls at the edge of the map would be automatically generated and marked with red. The system was also designed to facilitate some of the data streaming functionalities discussed in section 4.4.2. The goal was to have the level data sent at-will to a custom, primitive game framework designed to render models and support first-person exploration of the level. This was intended to be considered a low-fidelity software prototype since it would lack detail and behaviours, but still allow the designer to put together a rough level and experiment with object placement.

PlayTIME's Preview Tool

Built in-tandem with the tile editor was a custom preview tool, designed to accommodate a real-time updating version of the level designed in the tile editor. The vision for this tool was intended to be live display seen in Figure 4.13 (Scenario Mapper), so scenario creators could see their scene in action as they create it. Figure 4.17 shows the first-person preview, a low-fidelity representation of a portion of a level built with the tile editor. The grid is projected in transparent yellow, walls are represented with green rectangles, and stand-in objects are in the distance.

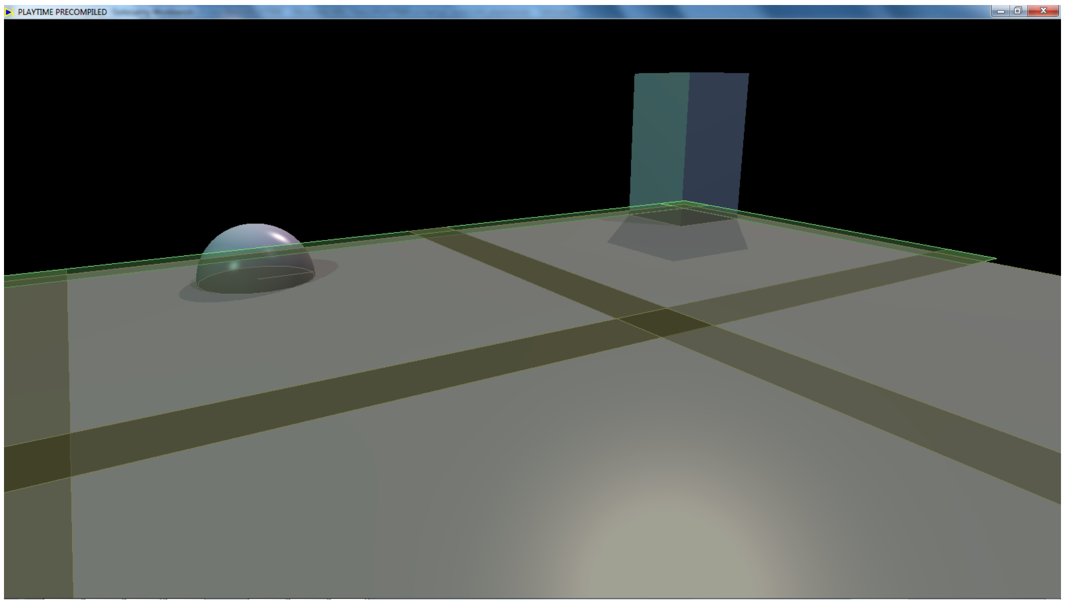


Figure 4.17: The low-fidelity digital prototype of the world created using the tile editor.

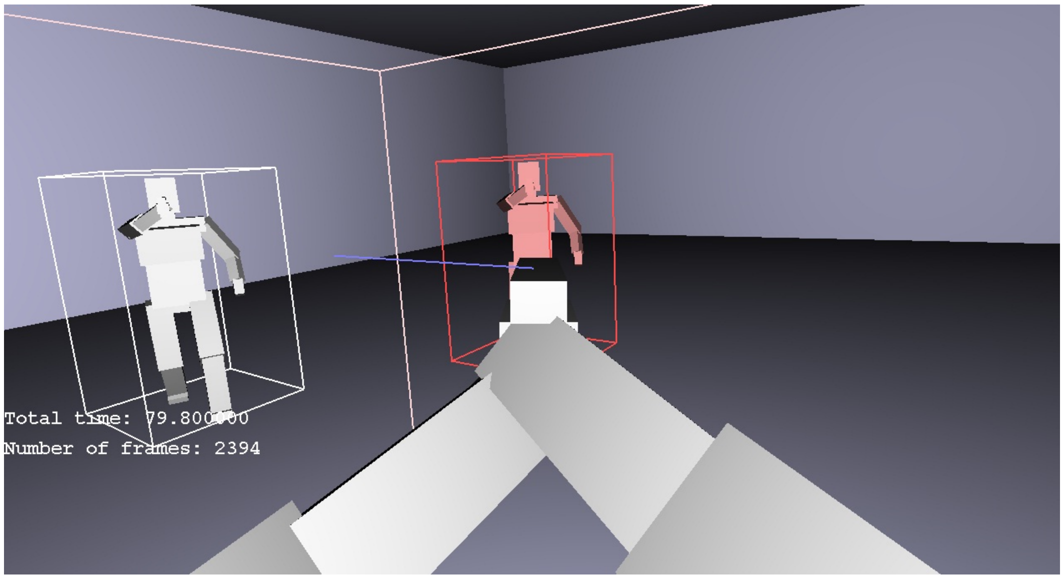


Figure 4.18: Another game prototype developed to test behaviours in PlayTIME. The game operated as a first-person shooter.

A different version of the preview was developed to test behaviours. In Figure 4.18, a working first-person shooter prototype can be seen with a working pistol object and enemies tracking and shooting at the player. These features were built as prototypes for specific features that will eventually be a part of PlayTIME. They hold potential for future development of PlayTIME.

Communication Protocol

The networking and communication protocol between the server and stations was designed using *RakNet* [91]. This design is not used in the current PlayTIME prototype as there are not enough stations developed to pose a need for the protocol. However, this design has been planned and experimented with for future development of PlayTIME and its stations.

The managers were designed to always behave the same way, which introduces and justifies the need for *listeners*. A listener's job is to receive and interpret commands from a manager, and route them to the correct receiver; the manager always directs messages to the currently-attached listener. Therefore there were different versions of each listener, each sharing the same basic functionalities, to accommodate local or remote messaging. Depending on the station each machine was intended to run, a different version of the PlayTIME station was compiled to delegate tasks cleanly and efficiently.

The server manager is responsible for receiving the data from the stations and storing it on its machine. This program is essentially the brain of the entire operation as its job is to know about everything going on in PlayTIME at all times: the resources and assets, the scenes, the stations currently in operation, and which AR tags currently being used.

Station managers are responsible for overseeing the operations at each station. The few use-cases that were started using this system were programmed to perform specific tasks appropriate to the station. One case was a prototype of the scenario station, which used AR tracking to position primitive objects.

The client manager behaved more as a middleman between the server and the station. One client interface existed per each section, and was either built into the station's code as part of the local machine (Figure 4.19), or run on a separate computer for dedicated AR processing (Figure 4.20). The separate-machine layout was designed to allow an extra machine to be strictly committed to detecting AR tags, processing its meaning and sending the result to the station manager to be utilized for station activities.

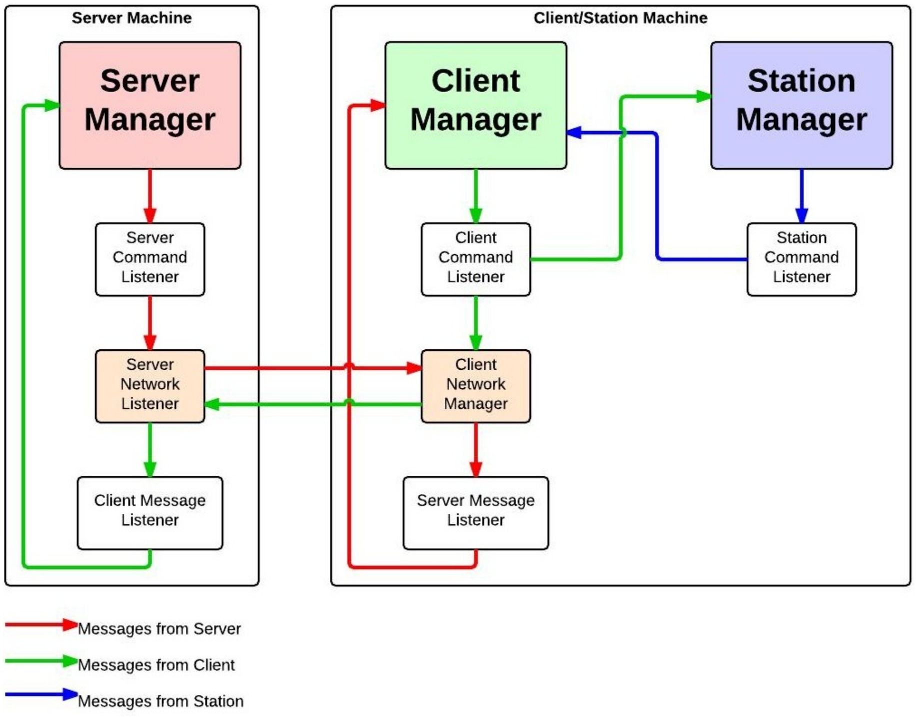


Figure 4.19: The original server-station communication model. The station and client 'middleman' run in the same program on the same computer.

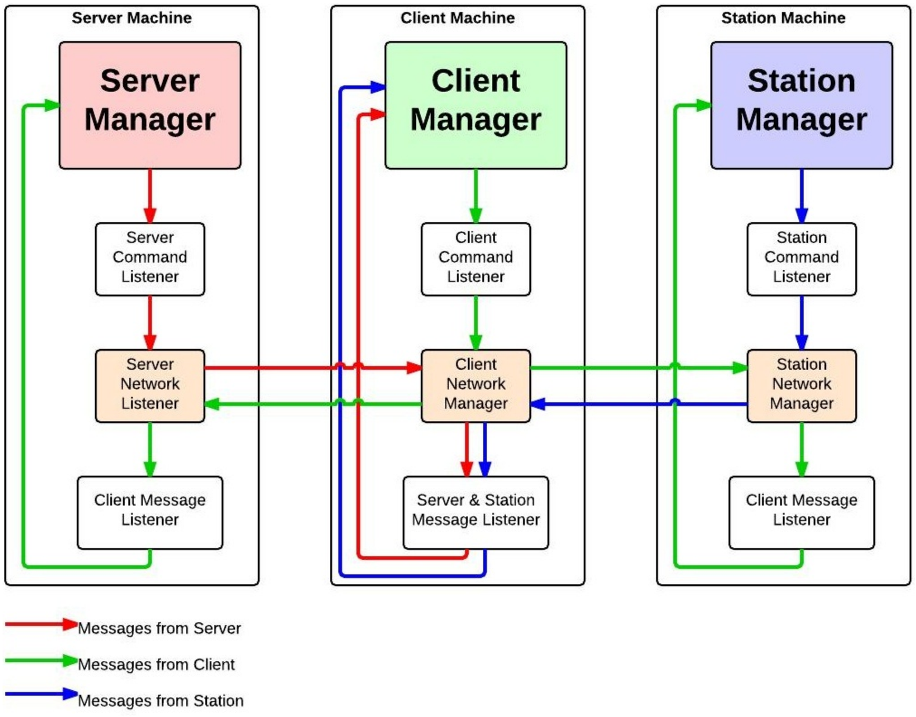


Figure 4.20: The alternative communication model. The station and client run on separate machines, allowing the client code to perform dedicated AR detection.

4.6 Summary

In this chapter we introduced PlayTIME, a conceptual framework for designing games and interactive scenarios. We discussed the structure of the proposed system and its components. An overview of the current implementation is provided as it will be used to evaluate the effectiveness of tangible interfaces.

PlayTIME is currently in a primitive state capable of evaluating the feasibility of tangible interfaces for some of the design stations. Some features exist for other purposes, but have been put aside as they are not directly involved with the evaluative study. Many of the features for the asset development stations have yet to be developed and are being worked on in parallel research projects. The current system prototype will play an important role in decision-making for the rest of the system later on.

In Chapters 5 and 6, we present a usability study evaluating the current implementation of PlayTIME, namely the Scenario Mapper station, as it is used to complete a level design task. We identify usability issues, as well as the creative potential of the system. We also investigate whether provides users with an enjoyable prototyping experience.

Chapter 5

User Evaluation of PlayTIME: Study Design

5.1 Introduction & Motivation

In this chapter, we use the current implementation of PlayTIME to determine if tangible interaction provides suitable interface for a game design environment. Participants took on the role of Level Designer to build a prototype level using the Angry Bots project in Unity. The outcomes of this study will ultimately provide guidelines for the development of new tangible interfaces to bridge the gap with the traditional computer, keyboard and mouse-based interfaces for game development and prototyping.

This study evaluates users' experiences of preparing an interactive scenario using two different systems: (1) PlayTIME integrated into the Angry Bots project, and (2) Unity's editor as-is, with a specific set of features allowed to align with the capabilities of PlayTIME. For this study, the systems were paired with a physical interface: PlayTIME was controlled using "paddles" with AR markers on them (the markers are described in section 4.5), and Unity was controlled using only the mouse. We will take a close look at different sets of data that will help us understand aspects of the systems used.

From this study, we hope to learn whether the current version of PlayTIME is a suitable starting point for future implementations to be used for different stations of the TIME framework. In this chapter, we discuss everything learned by inviting unfamiliar users to complete a design task using the current PlayTIME implementation. We will also discuss what parts of the system must be changed for future implementations.

5.1.1 Important Definitions

"Conditions"

The activities themselves may be referred to as the "*Conditions.*" This is the *primary independent variable* of the study. The study had a *within-subjects* design; each participant ran both conditions. The two conditions, or *levels*, are referred to as the "activities," named after one system paired with one tangible interface: *PlayTIME with AR paddles* (referred to as condition P) and *Unity with the mouse* (referred to as condition U). We are most concerned with *the effect of the condition on the results.* The main confounding factor we are investigating for each condition is the *system*, so either *PlayTIME* or *Unity.*

"Order"

Since this was a within-subject study, the results for each condition were balanced by alternating the order in which the activities were presented to the participants. The "*Activity Order*" is the *secondary independent variable*, analysed separately from the conditions. Those who completed the activity with PlayTIME first are referred to as "*P-first*" or "*PlayTIME-first,*" and those who started with Unity are referred to as "*U-first*" or "*Unity-first.*" The *levels* are P-first or U-first, and exactly half of the participants were assigned to each activity order.

When referring to specific participants, their unique participant ID number will be used, which includes their number in sequence and their starting activity. For example: *Participant 01-P* was participant number 1 and started with PlayTIME; *Participant 02-U* was participant number 2 and started with Unity.

"Groups"

The study had a total of *50 participants.* The first 30 participants will be referred to as "*Group 1.*" Participants in Group 1 were the first population to complete the study, but their results are not associated with survey data.[1] Because of this, a second, smaller set of participants were recruited to provide us with a set of complete data. The last 20 participants will be referred to as "*Group 2*" or the "*surveys-only*" group. Any combination of the two groups will be referred to as "*both groups.*"

In the interest of avoiding the assumption that the surveys from the 20 participants in Group 2 are representative of all 50 participants, we provide *two* sets of results in

[1] On October 31, 2014 we discovered that the server hosting the survey collection software had been mysteriously corrupted and was rendered inaccessible to anyone. The survey results from Group 1 were truly irrecoverable. Remember to back up your data, kids!

Chapter 6: the distinct population that had complete data (Group 2 or surveys-only), and the super-population of all participants, *only where applicable.*

5.1.2 Assigned Tasks

The activity was to be completed by each participant using two systems (the conditions), with the same data collected for each. The activity was completed using *PlayTIME* as an extension of Unity (condition P) using AR paddles for control, and the second system was *Unity* on its own (condition U), using the mouse for control without any keyboard. Half of the participants were first introduced to PlayTIME, and the other half were first introduced to Unity; the order swap was done to balance the data between the conditions.

The task was broken down into a small set of sub-tasks, which were the same for each condition:

1. Place the ***Player*** object as close to the marker as possible (***X*** in the map, ***red square*** in Unity)
2. Place ***2 Buzzers each*** in zones 1, 2 and 3 (see map)
3. Place ***1 Spider each*** in zones 1, 2 and 3
4. Place ***7 spiders*** anywhere in zone 4
5. Place ***2 spiders*** anywhere in zone 5
6. Ensure ***all*** the spiders have ***AI behaviours attached***
7. Change the attack radius of at least ***4 spiders anywhere***

Figure 5.1 shows the detailed map, labelling the specific zones and names of the areas.

5.1.3 Hypotheses

By conducting the activities using each system, we hope to find differences between the two across a broad set of metrics. Here we revisit the hypotheses discussed in Chapter 1:

1. On ***usability***: The use of PlayTIME will have a significant effect on the *performance* of the users, and this effect will be in PlayTIME's favour.

2. On ***creativity***: The use of PlayTIME will have a significant effect on the users' *creative output.*, and this effect will be in PlayTIME's favour.

3. On ***enjoyment and fun***: The use of PlayTIME will have a significant effect on the users' *emotions* and will positively affect users' *enjoyment* of the activity, and this effect will be in PlayTIME's favour.

The metrics used to find evidence of the hypotheses are outlined in section 6.2.

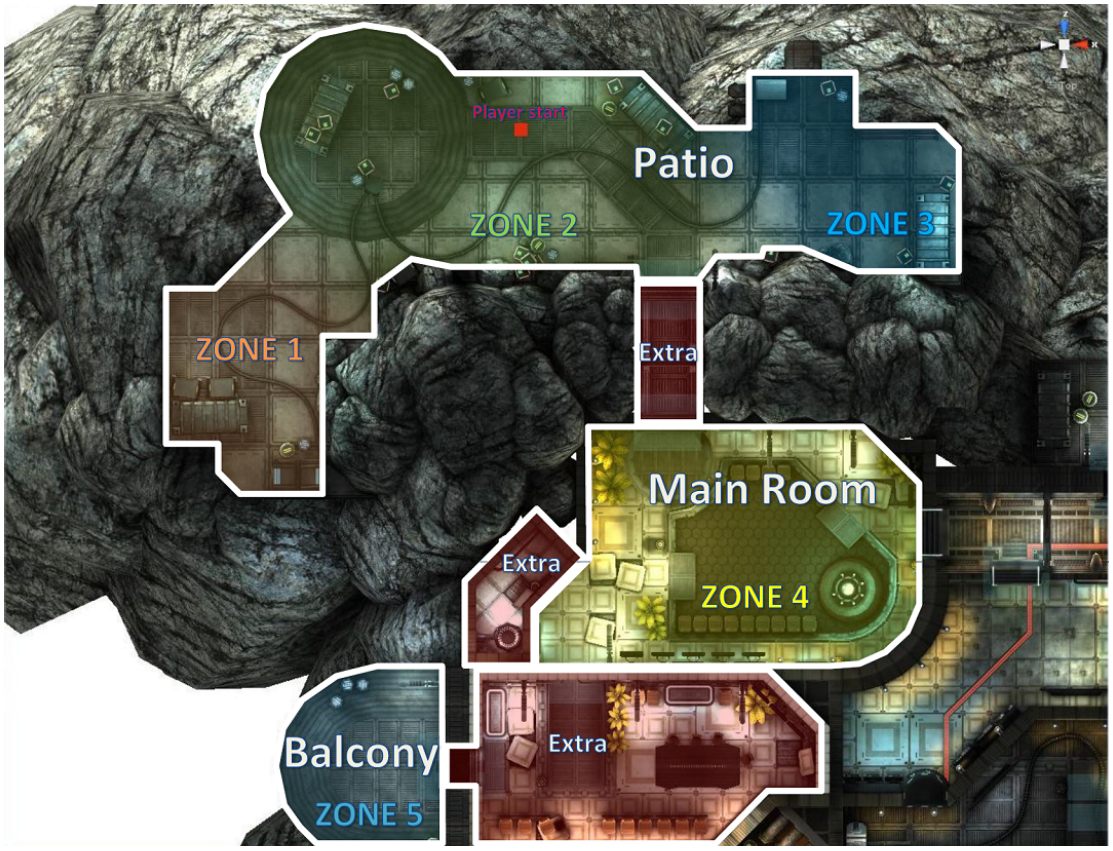


Figure 5.1: A map of the level provided to participants. Zones 1, 2 and 3 are referred to as "the patio." Zone 4 is "the main room" and zone 5 is "the balcony." The areas not labelled by zones are "extra" rooms, and users were permitted to place enemies in these locations.

5.2 Method

5.2.1 Session Overview

Participants began the study by reading and signing an informed consent form which provided them with a description of the study (REB file 14-014; see Appendix C.1). They then filled out the demographics questionnaire in which they described their experience and expertise in different disciplines of game and simulation design, and experience using a variety of game development-related tools.

Next, they were briefed on their role and task in the study: they were acting as Scenario Mappers, or level designers in general terms, and they would be required to build a simple level prototype in an existing game using two systems and interfaces. Both conditions required participants to complete the same activity, consisting of a small set of tasks which they could do in any order and preview as necessary.

All participants were given a step-by-step walk-through of their first system (condition), and provided with printed instructions in case they needed reference during the activity. Participants were not informed of the time limit unless they explicitly asked. They would be allowed 20 minutes to complete the activity, excluding preview time, with a warning at 15 minutes. The timer was paused while previews were run so the measured time included only what was spent building the level.

After being briefed on the system, participants were allowed five minutes to become acquainted with the system (if needed). Each participant then completed the activity starting with a default Unity scene.

Upon completing the activity to the satisfaction of the participant, or upon reaching the time limit, video capture was stopped, the current Unity scene was saved, and a pre-loaded online survey was administered. Each participant completed the post-condition questionnaire, and then was briefed on the second condition: those who started with PlayTIME were introduced to the Unity editor in the same fashion, and vice-versa. The activity was then repeated: trying out the new system, completing the assigned task, and filling out an online survey on the second system. After the second survey, a third and final survey was administered. This questionnaire looked at the ease of use and preference of both systems at the same time, and allowed participants to provide free-form feedback.

The average session time, from sitting down to look at the informed consent form, to completing the final survey, was 87 minutes. All of the materials provided and used during the experiment can be found in Appendix C.1.

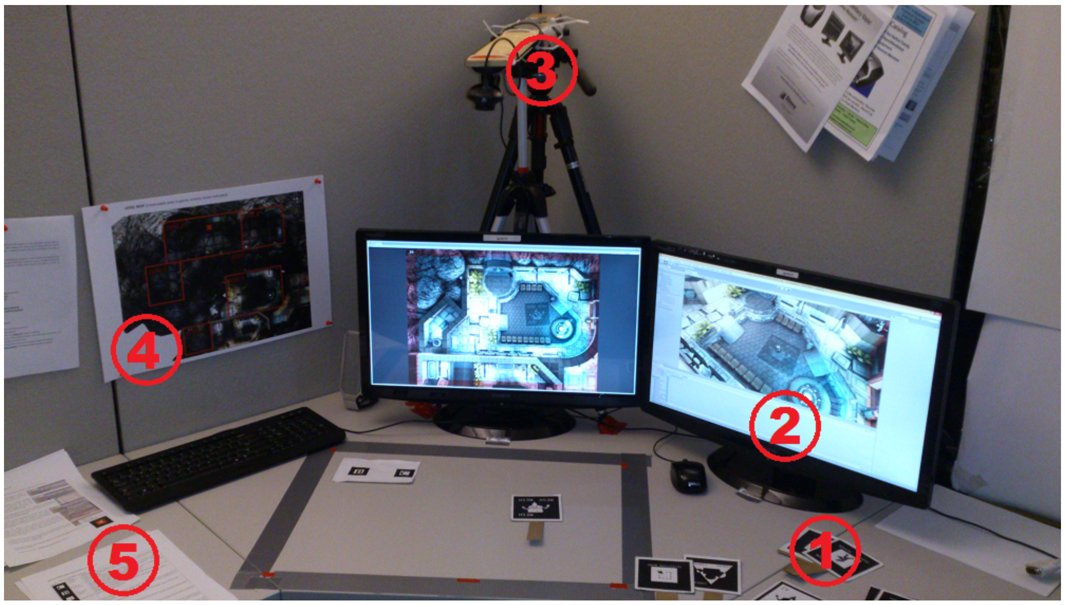


Figure 5.2: An overview of the study setup showing the different components. Pictured here are five different components of the setup: (1) the workspace complete with AR paddles; (2) the dual-monitor setup used to display the activity systems; (3) the setup used to hold the camera in place; (4) a map of the scenario to be worked on; and (5) the notes that were provided to the participants (observer section not pictured).

5.2.2 Experimental Setup

Figure 5.2 shows an overview of the different parts of the experimental setup. Participants were repeatedly informed that they were not allowed to change any part of the setup, at any time, for any reason. This was to keep a uniform setup for all participants.

The numbered items in the above image are described below.

1. Work area

Figure 5.3 shows the paddles that were used to operate PlayTIME. Each paddle consisted of an AR marker which would be recognized by the system to perform a specific task, and a handle for manipulation. As discussed later in this chapter, the handles proved to be invaluable due to the sensitivity of the markers themselves. The mouse can be seen towards the top-centre of the screen; this was the method of control for the Unity activity. The workspace was outlined with duct tape. This was an indication as to what the camera could see during the PlayTIME activity; anything outside this area would not be visible to the PlayTIME system.

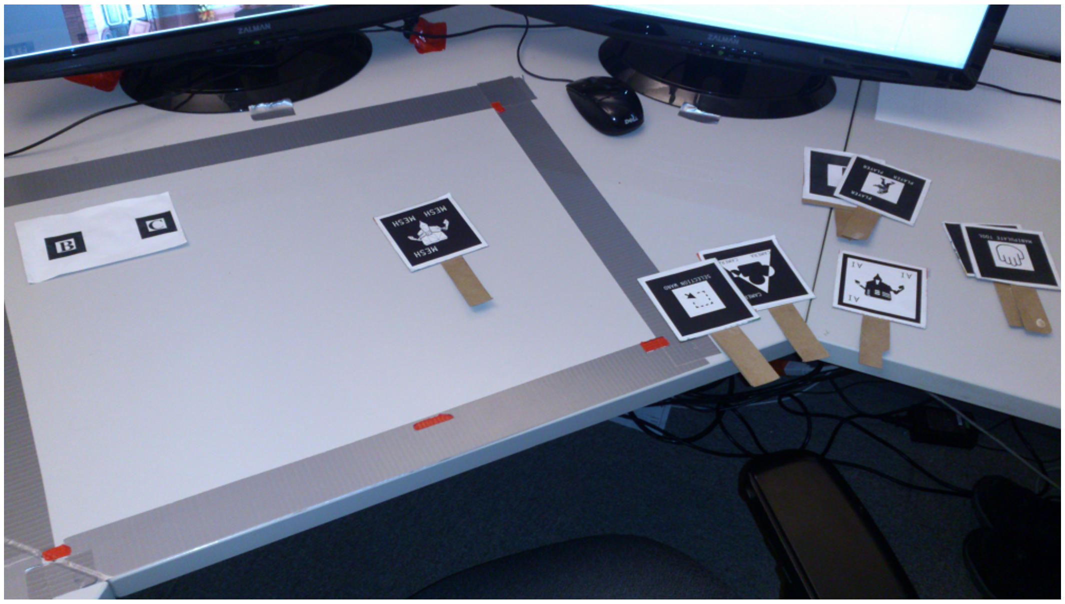


Figure 5.3: A look at the space participants had to work (labelled as (1) in Figure 5.2). Here we see the AR paddles used to control PlayTIME. The mouse is near the top. The duct tape represents the area visible to the camera.

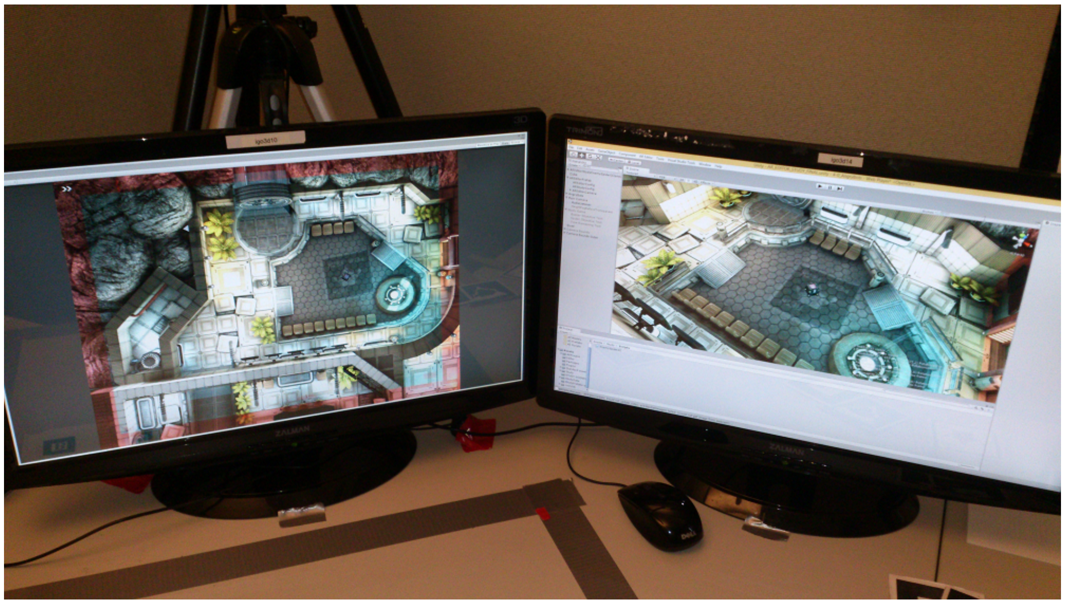


Figure 5.4: The dual monitor display setup for the study (labelled as (2) in Figure 5.2). Here, PlayTIME is enabled and is operated on the left monitor, while the right monitor provides an alternate view of the active marker within the scene.

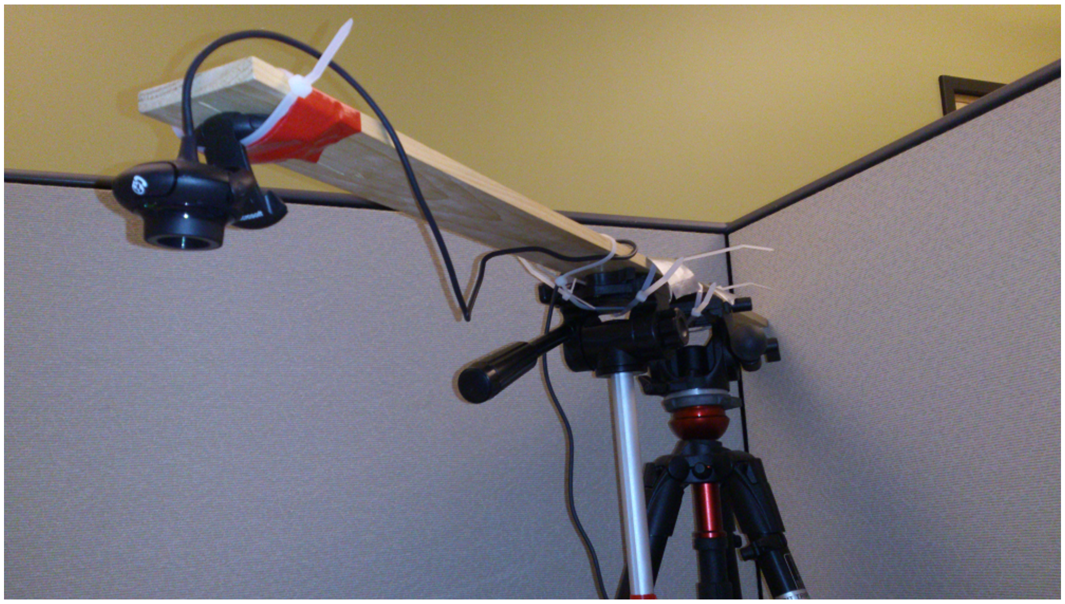


Figure 5.5: The camera rig used to help PlayTIME track AR markers (labelled as (3) in Figure 5.2). The camera was suspended approximately 60cm above the centre of the workspace using a wood plank attached to a pair of tripods.

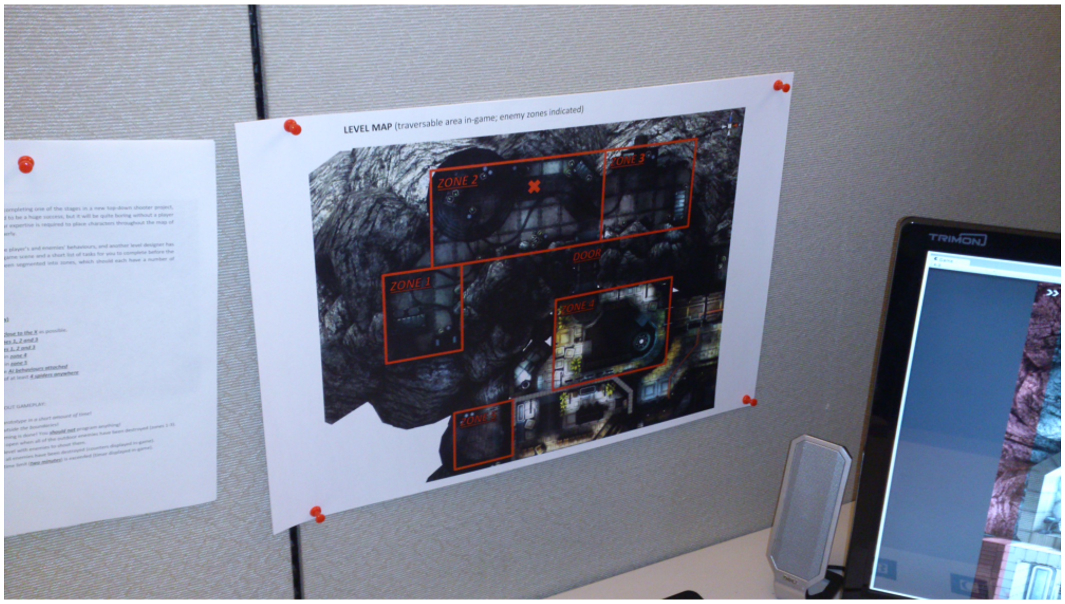


Figure 5.6: A view of the level map on the wall (labelled as (4) in Figure 5.2). The map showed the "zones" that enemies were to be placed in and was less detailed than the map seen in Figure 5.1. Extra areas in the map were identified to participants.

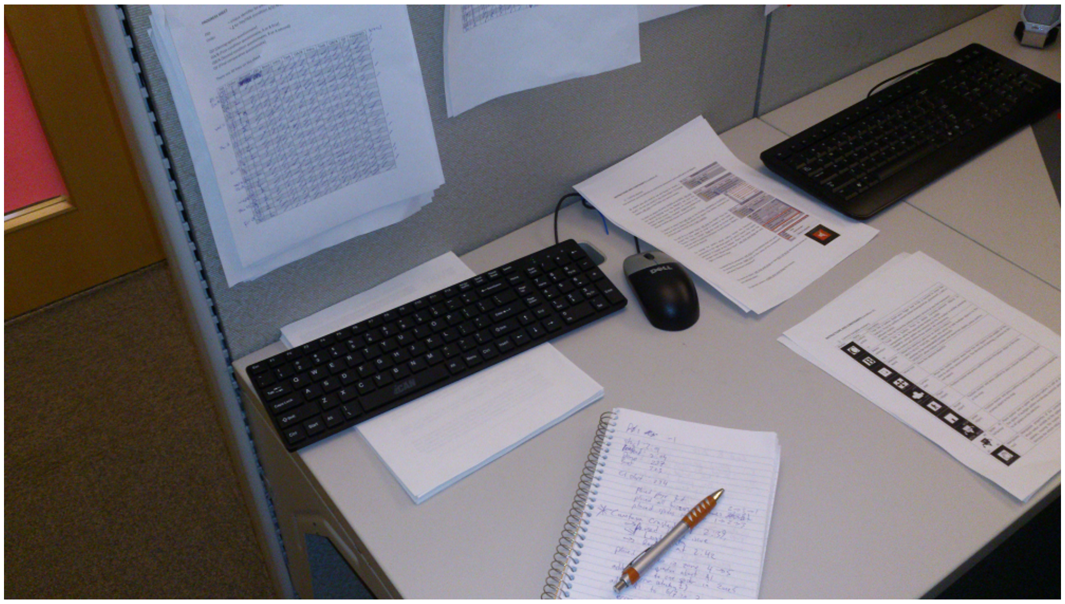


Figure 5.7: The reference sheets (labelled as (5) in Figure 5.2) and observer control setup. This image shows one set of reference sheets for each activity, but only the one pertaining to the current condition was made available. The observer setup allowed fluent control over the activity, primarily to avoid having to switch seats while toggling between surveys and activities. This also came in handy when system-related bugs emerged.

2. Monitors

Figure 5.4 shows the display setup: participants were provided with two side-by-side monitors. The left monitor was dedicated to the PlayTIME system and game previews. During the PlayTIME and AR activity (cond. P), the left monitor showed a top-down view of a portion of the level; users could change the current view using the camera pan marker (described later).

When a preview was started, the PlayTIME system was suspended and the left monitor switched to the in-game view, allowing users to play through the level they had been working on. When a preview was terminated, the left view would switch back to PlayTIME and the system would resume operation. Each participant was informed that the left view would be the primary view for PlayTIME.

The right monitor always showed Unity's editor. PlayTIME used this to follow any marker that was currently visible, providing users with a secondary view of what they were working on. The right monitor was used primarily for the Unity and mouse activity (cond. U), and during this condition the left monitor was inactive during development time and used only for previews.

3. Camera

Figure 5.5 shows the rig that was created to fix the camera at approximately 60cm above the work area: a wooden plank was zip-tied to a pair of tripods, and the camera was attached to the plank and adjusted to its permanent position. Participants were informed that they were not allowed to touch the camera as it would throw off calibration for future sessions. The camera stayed in roughly the same position for all sessions, giving all participants the same view of the desk, and allowing PlayTIME to operate without needing to recalibrate.

4. Map

Figure 5.6 shows the map of the level. The map was pinned to the wall of the experiment area and remained in the same place for all sessions. The map outlined different areas of the level (zones) where participants were to place a number of enemies. A detailed map can be seen in Figure 5.1; this differs from the map that participants were given, which only showed the zones, but did not give the areas names. The player start position was also marked more clearly with a large X.

5. Reference sheets and observer controls

Towards the right of Figure 5.7, the information sheets that participants were allowed to reference are shown; these sheets described the features of the current system and how to use them. Only the reference pages for the current condition were shown. These pages were the subject of some of the post-condition survey questions discussed later, pertaining to information.

Located just behind the participant, the observer controls were convenient for changing activities and surveys without having the participant move. This also provided a quick way for the experimenter to fix any issues that could possibly occur. Also, the experimenter had a notebook that was used to take note of any interesting happenings or quotes by the participant.

5.3 Demographics

Here we present and discuss the information collected about the participants.

5.3.1 Overview

For the first run of the study, participants were recruited from a population of game development students, specializing in a variety of disciplines. The target demographic was later changed to include people from a wider population: we included students with backgrounds in the creative domains of film, animation and games, focusing

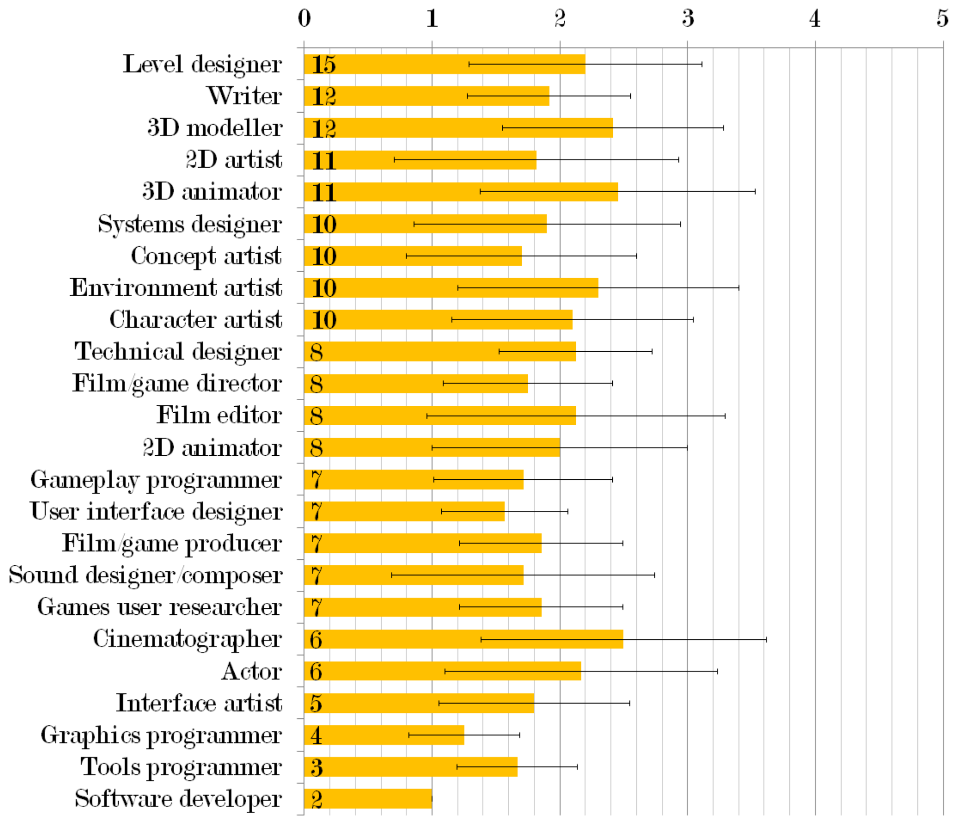


Figure 5.8: The average proficiency of the participants in a variety of creative roles. They have been sorted by the number of responses per role. The error bars represent one standard deviation from the mean.

on people who considered themselves novice or competent in their field. This section presents information about the participants from Group 2. The average age of the participants was 21.1 years old (SD=2.3 years), with an average of 2.6 years of experience (SD=1.7 years) working in their field.

Similar to the demographics questionnaire in Chapter 3, participants were asked to rate their proficiency in different roles (Figure 5.8), with new types of creative roles added for this demographic.

To describe expertise, participants picked a value from a five-point scale, with descriptions provided directly in the survey (see Appendix C for complete surveys):

0=No answer (assigned to blank responses); 1=Beginner; 2=Competent; 3=Intermediate; 4=Advanced; and 5=Expert

Participants also picked their proficiency and frequency of use for a variety of creative tools (Figure 5.9). This used the same expertise scale as above and a different five-point scale to describe frequency of use:

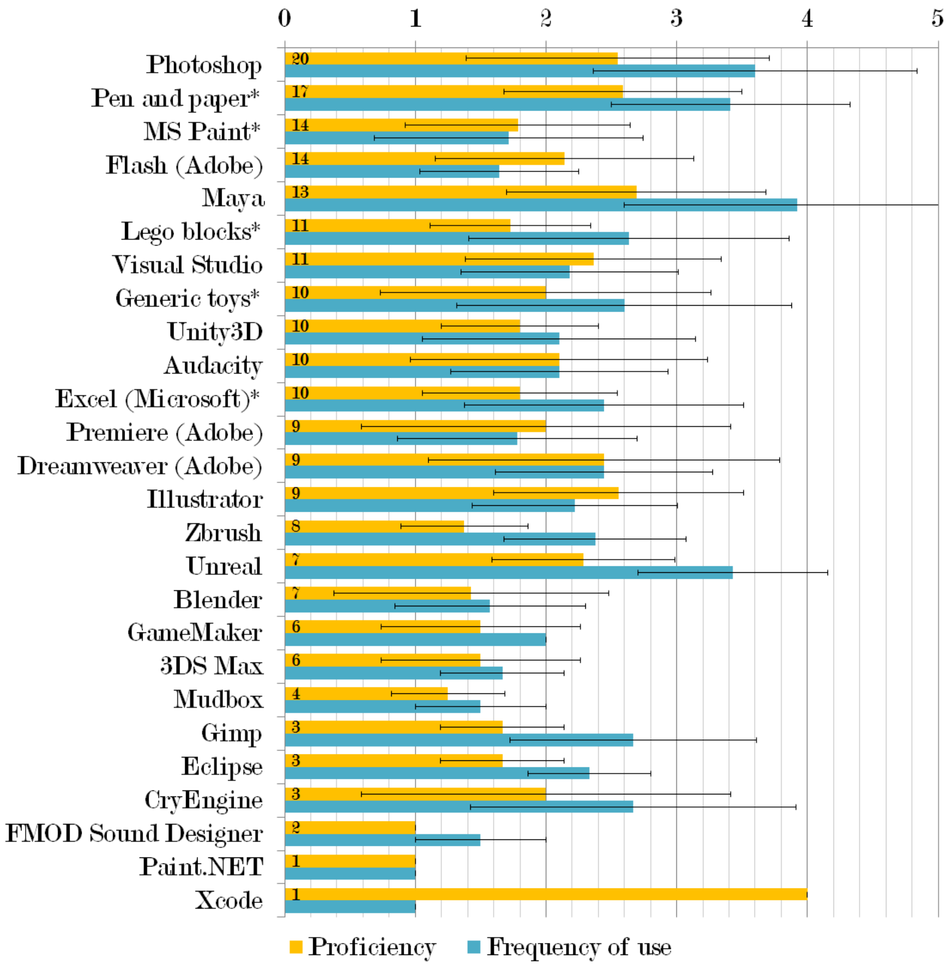


Figure 5.9: The average proficiency and frequency of use using a variety of creative tools. They have been sorted by the number of responses per tool. The error bars represent one standard deviation from the mean. Tools marked with an asterisk were subject to a common participant confusion, explained in the text.

0=No answer (assigned to blanks); 1=Never; 2=Rarely; 3=Occasionally; 4= Often; and 5=Always

Since all tools were to be rated only in the context of the creative domains practised by the participants, the tools marked with an asterisk might have confused some people; for example, we may all use pens and paper all the time, but how often do we use this as a tool specifically for game development or animation? This was clarified for all participants, but the numbers for these tools may be skewed towards general use instead of focusing on scenario prototyping in multimedia.

Participants were then asked to identify techniques used when building prototypes or scenarios for their line of work, whether it be games, film or animation (Figure 5.10). Finally, participants gave their opinions on the usefulness of different

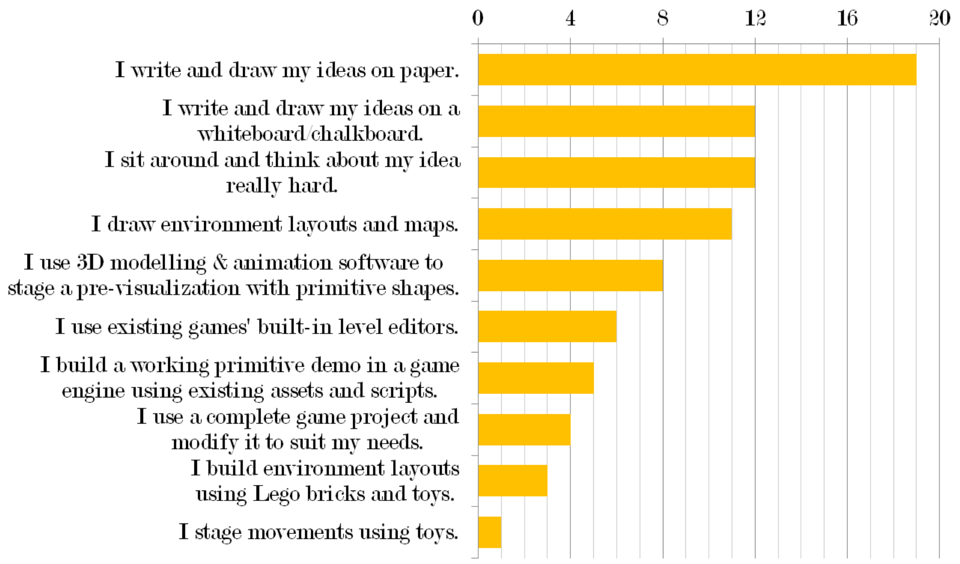


Figure 5.10: Common scenario design and prototyping techniques. This was a yes-or-no question; the chart displays the number of participants who answered yes to each metric, ranked by order.

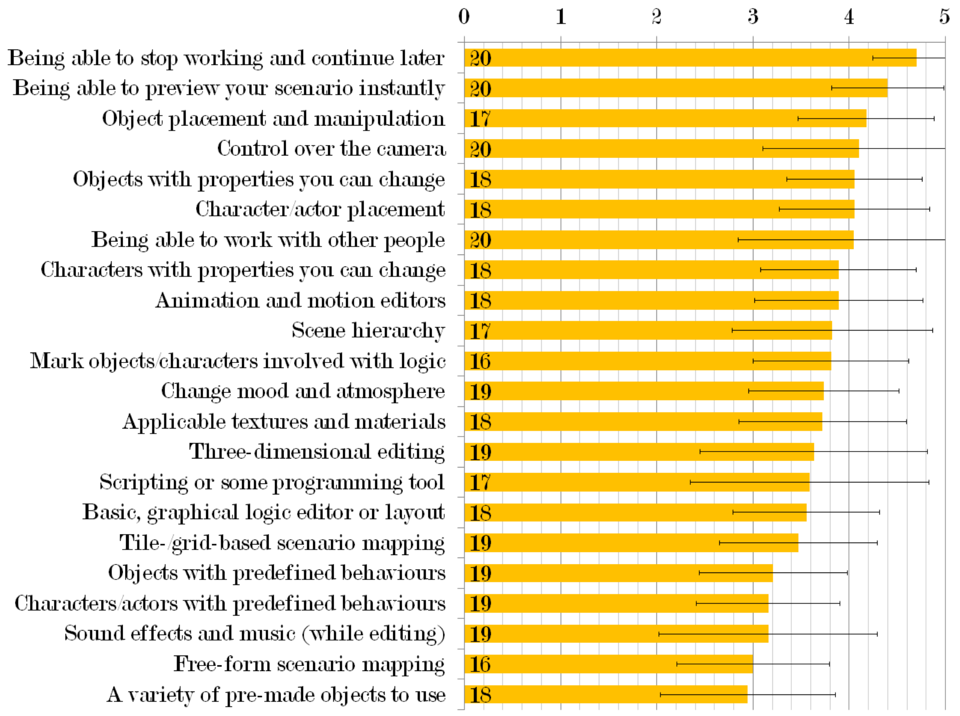


Figure 5.11: The average ratings of a variety of features, sorted by average importance rating, with the number of responses for each at the base of each bar. These features are commonly found in creative software.

features commonly found in creative, multimedia-related software (Figure 5.11). The importance of the features were rated using a five-point scale:

0=No answer (assigned to blanks); 1="I could do without it"; 2=Slightly important; 3=Moderately important; 4=Very important; 5=Essential

Some of these features would later be seen by the participants during the Unity activity, and most of the features are highly accessible for scenario design and prototyping tasks in a variety of software.

5.3.2 Expertise

Most participants claimed to have experience as level designers (N=15, M=2.2, SD= 0.909); these would have come from the game development population. The artists are clustered towards the top since the study was catered towards more artistic domains. On contrast, few people from this group had experience in the programming disciplines.[2]

All participants had some experience with Photoshop, which supports the high presence of artists (M=3.6, SD=1.241). Pen and paper and MS Paint were also rated highly since they can be used for art, and Maya was highly-rated by the people specializing in 3D art and design.

The game designers preferred game engines with editors, such as Unity (N=10, M=2.6, SD=1.281) and Unreal (N=7, M=3.43, SD=0.728). During condition B using Unity, participants who already had moderate experience with the editor were upset with the significant pruning of the features they were allowed to use. For example: participants 39-U and 43-U had already known that all it took to rotate the camera was a single right click-and-drag, and proceeded to do it even though they were instructed not to; the Unity editor camera was positioned in a way that matched PlayTIME's camera, which could not be rotated.

Another example: participant 41-U had attached AI to one spider and proceeded to duplicate the object several times to avoid having to repeat the placement *and* AI attachment steps. Unity's quick duplication affordance was known to the participant, who had ignored its restriction, and favourable over the timely process of navigating through folders and placing assets directly; this resulted in 76 seconds of observer intervention time to correct the duplicated objects and have the participant re-do these actions by following the rules. The first few duplications had gone unnoticed

[2]The surveys from Group 1 would have shown the opposite: that most people were programmers and few were artists. Of the 50 people who participated, there was an even distribution of skills.

by the observer, who had been writing down notes, but the actions were reverted and repeated correctly.

For future studies like this one, the target demographic should *strictly* include people who are *significantly inexperienced with any of the tools used in the study.* This will prevent any noticeable deviation in experience and result in a more uniform performance across participants. Furthermore, with this practise, participants should not have any expectations entering the study and would therefore remain patient and curious, following a design process uniform with other participants, instead of trying to exploit the system or wanting to use features that they are told not to. Vast design tools, such as Unity, should be left open to discovery, instead of having someone spoil the surprise and say that certain things are not allowed.

In their feedback, some of the participants suggested that PlayTIME might be better suited towards a younger demographic:

> *"...PlayTIME feels more like a toy; I can see kids using it then sending levels to their friends or building full games using pre-made objects."*
> - Participant 48-M

> *"...PlayTIME was just cool. I was like a kid discovering a toy for the first time."*
> - Participant 41-M

For evaluating a tool specifically built on the concept of playing, building and creating, perhaps it would be worthwhile to target children or adolescents as the demographic. We know that children love to play and explore, so they would be suitable for play-based research, especially if the assigned task required them to be openly creative. If someone should evaluate future implementations of PlayTIME or other TIME tools, the requirements should be simple yet imaginative such that the final outcome could be reached easily by a child.

5.3.3 Techniques

Of the 20 participants, 19 found that writing or drawing on paper would be useful for prototyping. Although whiteboards placed second (N=12), it is likely that paper was more agreeable since it is more accessible. We are surrounded by pens and paper in our daily lives, whereas we may not have direct access to a whiteboard if we are at home or away from work. Thinking was the third most popular technique (N=12), and drawing maps was the fourth (N=11).

Activities such as writing, drawing, sketching and "sitting around and thinking" are common tasks that we do in our daily lives to help us plan our activities and figure out what we are trying to accomplish. Therefore these are not necessarily design techniques that require a certain degree of expertise in a given discipline. The remaining techniques in the questionnaire were practised by less than half of the participants. Perhaps the high presence of non-game development personnel means that the importance of prototyping techniques was undervalued by the general population. The animators would more likely focus on the final production instead of a pre-visualization or perhaps they simply do not see the two as similar: a pre-visualization is as important for a film or animation as a low-fidelity prototype is important to a game.

5.3.4 Features

Similar to the TimeSplitters study in Chapter 3, having the ability to preview was extremely important and this was generally agreed upon by all participants (M=4.4, SD=0.583). Having learned from the TSFP Mapmaker's extremely slow preview load and unload times, Unity was a good choice to ensure a quick transition to and from the in-game preview. The custom level editor briefly described in Chapter 4 also learned from TSFP, and would have also accommodated fast and dynamic preview times, but that feature was not attainable for this study.

Previewing was trumped only by the importance of saving your work (M=4.7, SD=0.458). This metric received a relatively even distribution of Very Important and Essential ratings (4 and 5) across all disciplines, which demonstrates the universal need for saving, and possibly obsessively making backups. Although the saving responses from 3D artists blended in with the other disciplines, it is important to note that one of the most popular 3D modelling and animation tools, Autodesk Maya, is highly prone to crashes loss of work. This happens so frequently that it has become part of pop culture in the animation and game development communities.[3]

Object placement and manipulation has the third-highest ratings (N=17, M= 4.176, SD=0.706). This is important to this study since object placement formed the basis of the activity in the study: creating a fun and winnable level in the Angry Bots world. Spiders and buzzers were required to "win," with behaviour attachment only present to make the scenario more interesting; it would not have been fun if the enemies could not attack the player, thereby adding challenge to the game. The features pertaining to "objects with properties" (N=18, M=4.056, SD=0.705) and

[3]http://cdn.meme.am/instances/55741538.jpg

"characters with properties" (N=18, M=3.889, SD=0.809) were also up there; this relates to the AI behaviours that could be manipulated to give each object or character in the level its own effect within the world.

The fourth most important feature was having control over the camera (N=20, M=4.1, SD=0.995), which also proved to be necessary during the experiment. Overall, the features were generally highly-rated, but it is interesting to see that the features that participants found most important were directly involved with the study. The exact uses of the features during the study are discussed further in section 6.5.5.

5.4 Summary

In this chapter we introduced the study used to evaluate PlayTIME. We discussed the experimental design and setup, and the participants. The target audience for the study consisted of people in creative domains, such as film, animation and games. We had a variety of expertise across all participants, but most of the experience leaned towards visual arts and animation. The results of the study are presented in Chapter 6.

Chapter 6

User Evaluation of PlayTIME: Results & Implications

6.1 Overview

In Chapter 5 we introduced the user study evaluating the current implementation of PlayTIME. The study aims to find significant differences between PlayTIME, as an extension of Unity, and Unity on its own in three areas: *usability*, *creativity*, and *enjoyment*. We presented an overview of the study and its procedure, and discussed the users that participated. Here we introduce the data analysis procedures and then we present the results of the study. Finally, we discuss the implications of the results.

6.2 Data Analysis Methods

Many measurements were made over the course of the study. Surveys were used to answer questions pertaining to system usability, creativity, and feelings or affect. A screen capture video of each session was used to identify the many uses of each system and time measurements of the different uses and tasks completed. The final scenes from each activity were used to explore the participants' creativity. Participant feedback was examined to look for common preferences for the systems.

The main question was whether the system used had an effect on the results. Therefore the main independent variable was one of two conditions: (1) PlayTIME controlled using the AR paddles, or (2) Unity's editor controlled using the mouse. A secondary question was whether the order each participants used the systems made a difference. Therefore it was important to separate the data and do secondary analyses on the same metrics within the P-first and U-first groups.

The appropriate sample size, mean, median and standard deviation were calculated for each data set. The raw data were compared between the different groups to detect significant differences.

6.2.1 Significance Testing

For significance testing, three flexible methods for non-parametric data were used: the *Kruskal-Wallis one-way analysis of variance by ranks* [92] (referred to as the "KW" test) was used to analyse the effects of both the condition and the activity order; the *Wilcoxon Signed-Rank test* [93] ("WSR") the effect of the conditions only (cond. P data against cond. U data); and the *Mann-Whitney-Wilcoxon rank-sum test* [93] [94] ("MWW"), also called the *Mann-Whitney U test*, was used to analyse the effect of the activity order (P-first data against U-first data).

A p-value was collected for each metric, denoting the significance of the effects, telling us how much the data from each set differed from the other sets. For all p-values, the significance level was $\alpha = 0.05$. Therefore, a p-value less than the significance level ($p < \alpha$) was considered *statistically significant*, meaning the condition or order had a strong impact on the tested data set, and the data was noticeably different between groups. All significance testing was done using *R* [95], which has existing functions for KW, WSR and MWW.

The Wilcoxon Signed-Rank test and the Mann-Whitney-Wilcoxon test often had trouble computing precise p-values when data from separate groups were tied, and the signed rank test alone required an equal number of samples from each of the groups being compared. Furthermore, the Wilcoxon Signed-Rank test assumes that data are paired; therefore it was only useful for comparing data from the activity conditions where participants provided one full set of results for each condition. The Mann-Whitney-Wilcoxon test assumes that data are not paired; therefore it was useful for comparing equivalent data of the P-first and U-first groups. The strongest test was the Kruskal-Wallis test, which ran consistently without errors. Therefore all p-values from Kruskal-Wallis tests take priority; the others were used for extra support.

6.2.2 Organization of Data

For the condition tests, each data set was analysed in *three parts*. First, the complete sets of data, balanced by the alternating order, were compared for significance ("all"). Second, the data sets for each condition within the P-first group were compared. Finally, the data sets for each condition within the U-first group were compared. This *within-groups* split allowed analysis with and without the need for balancing.

Each population tells us explicitly about the effect of the conditions, and the split groups also implicitly tell us about the effect of order.

For the order tests, the data was analysed in *two parts*. First, the PlayTIME data from P-first was compared against the Unity data from P-first. Second, the PlayTIME data from U-first was compared against the Unity data from U-first. The data *was* tested as a combination of PlayTIME and Unity data (i.e. all P-first vs. all U-first), however *nearly all* of the tests returned *highly insignificant results*, since the data sets coming from each condition are very different; it simply does not make sense to combine different data.[1] Therefore the unique data sets from each condition were kept isolated, and this explicitly tells us about the effect that order had on the results from each condition independently.

6.2.3 Survey Collection & Analysis

Surveys were built and hosted on SurveyMonkey.com [96]. The four survey web pages were loaded and minimized before each session to avoid wasting time during the session.

All post-condition questionnaires used Likert scales [97] to directly rate the interfaces presented during the study. For usability of the systems and satisfaction using the systems, we used the *Computer System Usability Questionnaire* (CSUQ) [5] [6], which may also be referred to as the *Post-Study System Usability Questionnaire* (PSSUQ) [5] [7]; the difference is simply a slight change of wording. This is a nineteen-question survey that provides a general assessment of a system's usability. CSUQ/PSSUQ are far more detailed than the three-question After-Scenario Questionnaire (ASQ) [98] [99], which summarizes user satisfaction in three questions. CSUQ/PSSUQ improves on the System Usability Scale (SUS) questionnaire [100] because it avoids errors by caused by confusion between affirmative and negative responses [101] [102]. It also provides more categories of metrics: system usefulness, information quality, interface quality and overall satisfaction; SUS only provides metrics in usability and learning ability [103]. Other popular surveys exist, such as the Questionnaire for User Interface Satisfaction (QUIS) [104], but the questions are not suited for this experiment.

Usability is one of the most important and frequently sought-after attributes of systems designed in both the domains of HCI and game design. It was critical to select a qualitative measurement that would quickly and simply provide an overview

[1] The combined-data tests that returned significant results were on the same metrics for which *both* of the other order tests returned *highly significant* results. The p-values for combined data tests have since been deleted.

of PlayTIME's usability. CSUQ was selected over other questionnaires because it provides users with a chance to evaluate the system as a whole, and since its strictly-affirmative wording makes it easier to understand than other questionnaires, namely the SUS [101] [102].

We used the Creativity Support Index (CSI) [8] [9] [10] questionnaire to determine how participants felt about the systems' abilities to support creativity and expression, immersion, enjoyment, exploration, producing desired results and collaboration. As explained in [8]:

> *"Each agreement statement is answered on a scale of 'Highly Disagree' (1) to 'Highly Agree' (10). In deployment, the factor names are not shown, and the participant does not see the statements grouped by factor."* (p.21:6)

The CSI result is a single score ranging from 0 to 100, which is analogous to a percentage grade. Two minor errors were made with our execution of the CSI questionnaire. Randomization was disabled, so the responses stayed grouped by factor, but the factor names were not shown. Also, our responses were limited to the same seven-point Likert scale used for the other questionnaires, ranging from Strongly Disagree (1) to Strongly Agree (7). This error was corrected by applying a simple linear function to all data points:

$$x' = 1.5x - 0.5$$

where x is a data point in the range [1,7], transformed by the function into x' within the range [1,10], used to compute the final score. This fix resulted in having a final score correctly ranging from 0 to 100 instead of a faulty score ranging from 0 to 70.

Since PlayTIME is designed to support collaboration, creative expression, and enjoyability, it was important to find a qualitative measurement that could explain how well PlayTIME supported these areas. CSI was selected since it includes these key factors in its score calculation. The CSI is a relatively new metric in systems research, useful for summarizing how well a system supports a variety of creativity-related attributes in a single score.

To measure how participants felt while completing the assigned tasks, we used the Positive and Negative Affect Schedule (PANAS) questionnaire [11], a widely accepted measure of positive and negative emotions. In this experiment, the questionnaire specified that participants were to answer based on how they felt momentarily at the end of each condition, producing two separate scores between 10 and 50: *positive affect* and *negative affect*. The former provides a measure of strong *positive* emotions (e.g. excited, interested), and the latter measures *negative* emotions (e.g. distressed, hostile). A high positive score indicates a state of well-being at the time of the test, and a high negative score indicates the opposite. A PANAS questionnaire was added to the end of the demographic survey so that we could see how the emotions changed *through* each condition.

It was important to evaluate affect since it would be useful to describe the enjoyment of the systems. Since one of the hopes for PlayTIME is to have it add playfulness to game design, making the process feel less like work and more like play, it was important to determine how participants were feeling throughout the experiment. PANAS was selected because of its popularity across various scientific fields. Its reliability and usefulness are validated in [11] and [105].

In the post-study questionnaire, a Single Ease Question (SEQ) [106] [102] asked participants to rate the ease of use for *each feature* used and tasks performed under both conditions; the individual SEQ responses were averaged into a single ease of use score per-system. Next, participants directly selected their preference between the two *interfaces* for completing specific tasks. The preference scores were given a weight of +1 if the AR paddles were picked and -1 if the mouse was picked, and the average for all rated features determined whether each participant leaned more towards the AR paddles or the mouse as the preferred input.

6.2.4 Video Capture & Analysis

During each session, the participant's activities were captured using the screen capture software *Camtasia Studio* [107]. For analysis, we measured and annotated the time spent performing a wide variety of tasks using each system, and the distribution of time spent using each feature in the experiment. We will explain how much time users spent completing tasks correctly and incorrectly, and the time spent doing nothing at all or recovering from system errors.

The performance annotations were done by hand, with each change of action measured within one fifth of a second. Actions and their start time were recorded in a Microsoft Excel spreadsheet with the start time. The end time of each action was programmed to copy the start time of the next, thus producing a duration for every action (similar to the statistics described in Chapter 3). Special cases observed in the PlayTIME recordings were modified by hand (see section. Later, each duration was logged in a master spreadsheet that had a list of all participants along the X axis and all observed actions along the Y axis. Durations were summed to produce complete time measurements for all observed actions for all participants.

To achieve an accurate representation of participant actions using each system, it was important to note the many possible correct and incorrect uses to which they were exposed. Existing video annotation software may have helped, but despite the time it took to translate the videos by hand, we were able to identify 130 actions that occurred while using PlayTIME, and 80 actions that occurred while using Unity, across all participants.

6.2.5 Scene Analysis

The Unity scenes created by participants allow us to tell how far they deviated from the instructions they were given. The scenes were analysed because they may contribute to the creativity aspect of their development. The Manhattan distance [108] was used to measure the deviation of each scene from the exercise that the participants were assigned. Different components of the scene were assigned weights based on importance: enemy placements were assigned a weight of 1, AI attachment and manipulation were assigned a weight of 0.5 (since the ability to have AI was dependent on the number of spiders present), and the player token was assigned a weight of 0 since the instructions restricted its placement. The formula used to score each scene was a weighted Manhattan distance, or a measure of how much the final outcome deviated from the assigned task:

$$d = \sum_{i=1}^{n} w_i |p_i - x_i|$$

where i is a factor in the task list, w_i is the weight of each factor, p_i is the participant's count for each factor, and x_i is the expected count for each factor in the task description. The formula describes the absolute number of differences between a participant's data and the expected values, with a weight factor included to denote the importance of potential deviations. A participant who followed the task description exactly received a score of zero. If the participant deviated from the instructions by adding one extra buzzer enemy, their score would be 1. Removing an enemy from one area and placing it elsewhere would yield a score of 2. Modifying the AI of one more spider than what was asked would add 0.5, etc. This analysis provides a measurable value to describe creativity, since a higher score may imply that the participant wanted to complete the task in a way that was not already determined for them. This metric contributed to the analysis of *creativity* since it ultimately reflects the placement choices made by participants, measuring how closely they decided to follow the task. This metric is discussed further in sections 6.3.5, 6.4.2 and 6.5.6.

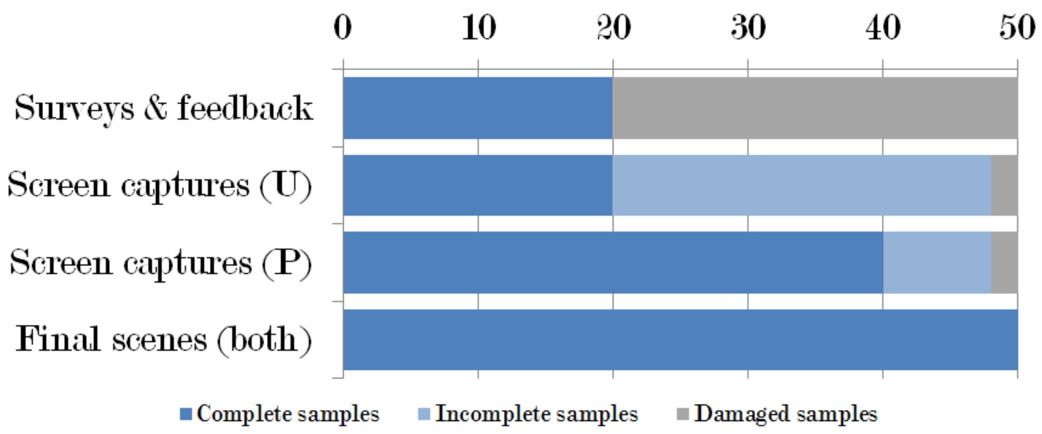


Figure 6.1: A graphical overview of the data collected during the study.

6.3 Results

6.3.1 Summary of Data Collected

There were 20 participants with a *complete* set of data throughout the study. This section presents those 20 complete samples from Group 2 alone. Section 6.4 presents the performance and scene data from all participants, including all 20 samples from Group 2 and the portions collected from Group 1.

All core data collected during the study is represented graphically in Figure 6.1. Additional complementary figures may be found in Appendix E.

Surveys & Feedback

A set of questionnaires administered after using each system gave users the chance to rate their experiences. The surveys include a pre-study demographics questionnaire, a set of scientific questionnaires after each condition (the same for each condition), and a post-study ratings questionnaire. The survey data presented in this chapter comes from Group 2 only, for a total of 20 samples. The surveys from Group 1 were corrupted and therefore unusable.

At the end of the post-study questionnaire each participant was given the opportunity to provide comments and feedback based on what they liked and did not like about the systems, their preference, and whether their first activity helped their second. The surveys and notable comments provided by participants are discussed throughout section 6.5 where appropriate.

Videos

The screen recordings afforded us a detailed performance analysis, and an overview of the distribution of activity time for each participant. For the Unity activity, only the 20 videos from Group 2 were examined because they were associated with the survey data. All PlayTIME videos form Group 2 were analysed along with 20 from Group 1, for a total of 40 samples. The Group 2 results are presented in section 6.3.4, and the results from all participants are presented in section 6.4.1.

Scenes

The Unity scenes saved after each activity show the final results of both conditions for all participants and may yield important information pertaining to creativity. All of the scenes were analysed for differences, yielding a total of 50 samples for each condition. The scenes from Group 2 are presented in section 6.3.5, and the results from all participants are presented in section 6.4.2.

6.3.2 Post-Condition Questionnaires

PANAS Questionnaire

The Positive and Negative Affect Schedule (PANAS) [11] questionnaire, administered before the study and after each activity, describes the conditions' effects on the participants' emotions. During the questionnaire, a set of 20 emotions (10 positive and 10 negative) are shown to the participant, who is asked to rate their momentary experience with each emotion using a five-point Likert scale:

1=Very slightly or not at all; 2=A little; 3=Moderately; 4=Quite a bit; and 5=Extremely

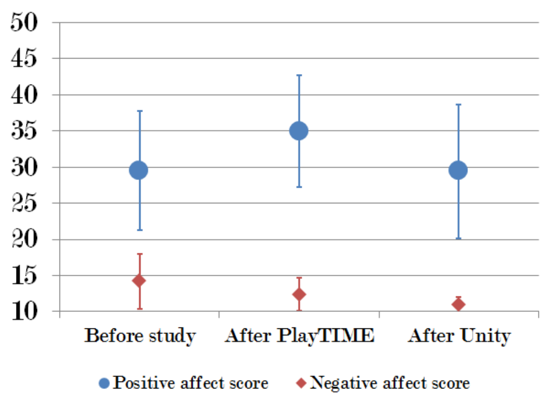
The positive emotion ratings are added up for a positive affect score ranging from 10 to 50, and the same is done for the negative emotion ratings.

Figure 6.3 shows the average overall PANAS scores and changes at each time of measurement. Table 6.1 shows all of the p-values retrieved from statistical testing of the PANAS data.

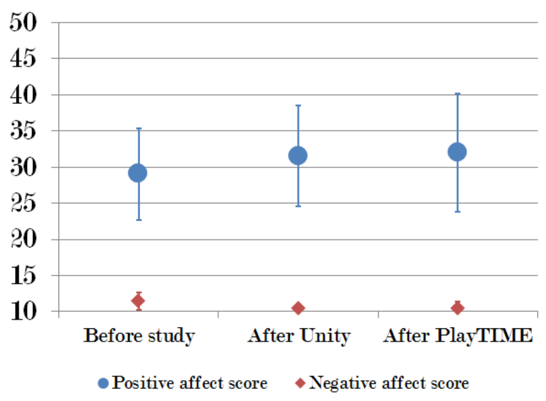
The average affect scores for the P-first and U-first groups can be seen as progressions over the entire study in Figure 6.2.

CSUQ

The Computer System Usability Questionnaire (CSUQ) [5] [6] was administered after each activity using a seven-point Likert scale for each question:

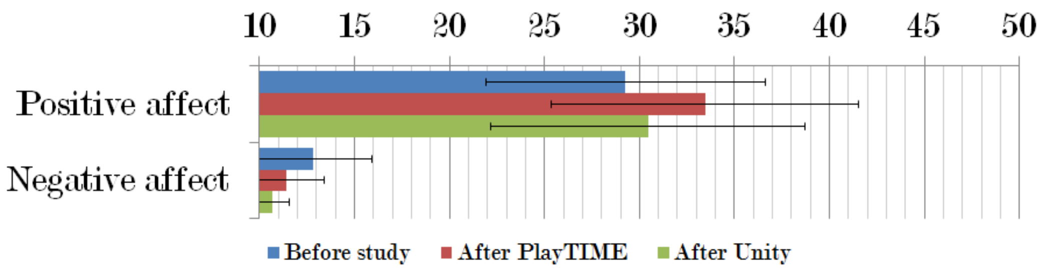


(a) PlayTIME-first

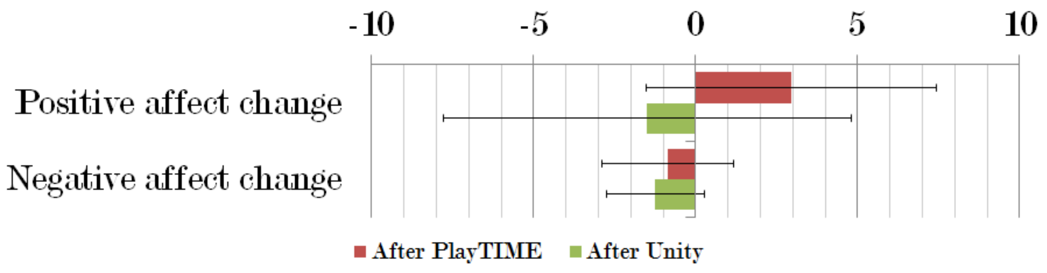


(b) Unity-first

Figure 6.2: The average PANAS scores, ranging from 10 to 50, as a progression over the duration of the study for the PlayTIME-first and Unity-first participants. The error bars represent one standard deviation from the mean.



(a) Pre- and post-condition averages

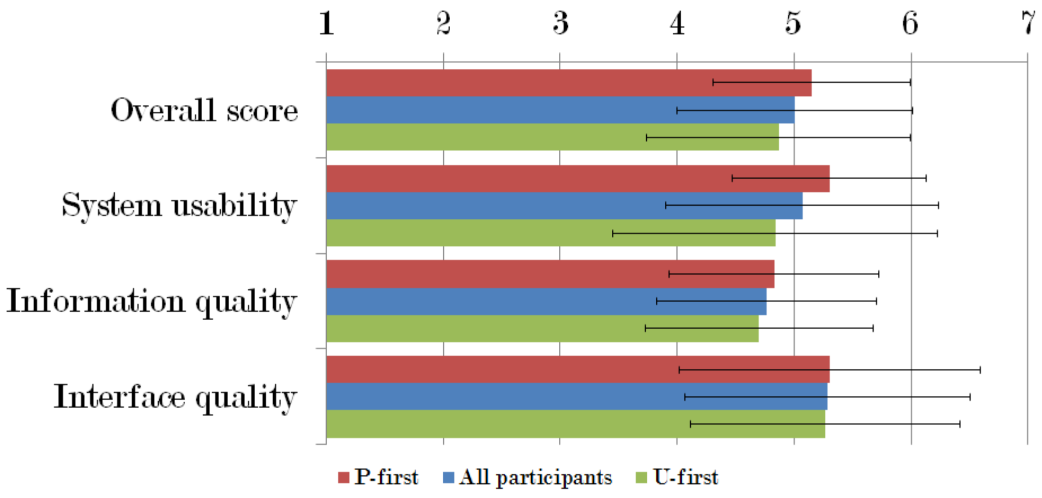


(b) Average change through conditions

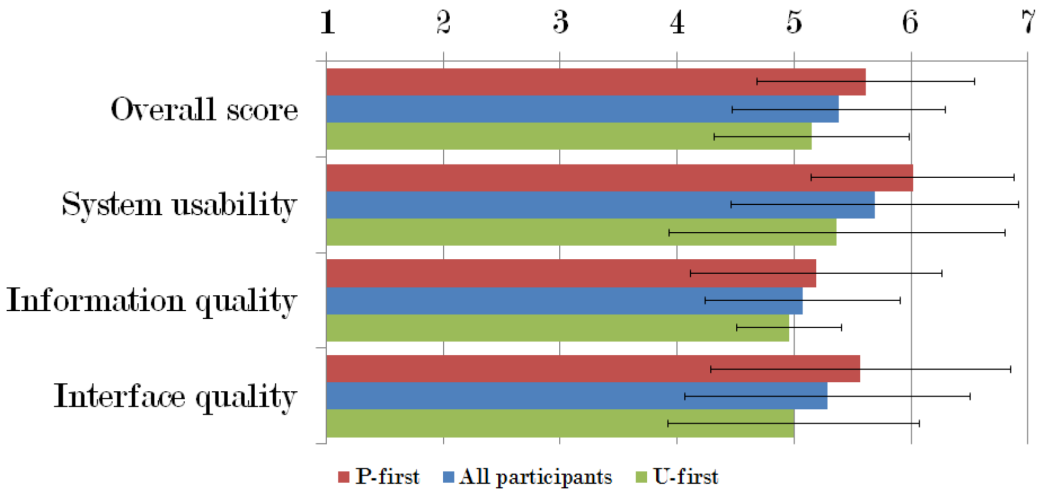
Figure 6.3: The average PANAS scores for all participants between both groups. The error bars represent one standard deviation from the mean.

Kruskal-Wallis test: ***PlayTIME vs. Unity***			
	All	P-first	U-first
Positive affect change through condition	**0.02532**	**0.0005718**	0.2864
Positive affect after condition	0.2907	0.2559	0.8792
Negative affect change through condition	0.3621	0.5296	**0.02289**
Negative affect after condition	0.2808	0.1205	0.9626
Kruskal-Wallis test: ***P-first vs. U-first***			
		PlayTIME	Unity
Positive affect change through condition		**0.01238**	**0.004003**
Positive affect after condition		0.5447	0.5191
Negative affect change through condition		**0.04701**	0.9681
Negative affect after condition		**0.01674**	0.155
Wilcoxon Signed-Rank test: ***PlayTIME vs. Unity***			
	All	P-first	U-first
Positive affect change through condition	**0.04362**	**0.005889**	0.5738
Positive affect after condition	**0.03465**	**0.02826**	0.8314
Negative affect change through condition	0.5254	0.4403	**0.04983**
Negative affect after condition	**0.04143**	0.05791	0.8501
Mann-Whitney-Wilcoxon test: ***P-first vs. U-first***			
		PlayTIME	Unity
Positive affect change through condition		**0.01377**	**0.00451**
Positive affect after condition		0.5702	0.544
Negative affect change through condition		*0.05159*	1
Negative affect after condition		**0.01864**	0.1675

Table 6.1: This table shows the statistical p-values from the PANAS questionnaire data. Values less than 0.05 are considered statistically significant and are boldfaced. Values less than 0.01 are considered extremely significant and are shown in red. Notable values greater than 0.05 are italicized.



(a) CSUQ/PSSUQ average scores for the PlayTIME activity



(b) CSUQ/PSSUQ average scores for the Unity activity

Figure 6.4: The average CSUQ/PSSUQ scores for all participants between both groups. The error bars represent one standard deviation from the mean.

Kruskal-Wallis test: *PlayTIME vs. Unity*			
	All	P-first	U-first
Overall CSUQ score	0.208	0.1615	0.7052
System usability	**0.03021**	*0.05327*	0.2714
Information quality	0.4472	0.3245	0.8492
Interface quality	1	0.5689	0.5946
Kruskal-Wallis test: *P-first vs. U-first*			
		PlayTIME	Unity
Overall CSUQ score		0.6227	0.2114
System usability		0.4952	0.3245
Information quality		0.7616	0.3823
Interface quality		0.9093	0.1822
Wilcoxon Signed-Rank test: *PlayTIME vs. Unity*			
	All	P-first	U-first
Overall CSUQ score	**0.04572**	**0.01367**	0.386
System usability	**0.009305**	**0.005666**	0.3135
Information quality	0.06072	0.102	0.4002
Interface quality	0.9055	0.6224	0.37
Mann-Whitney-Wilcoxon test: *P-first vs. U-first*			
		PlayTIME	Unity
Overall CSUQ score		0.6497	0.2256
System usability		0.5194	0.3434
Information quality		0.7906	0.4034
Interface quality		0.9395	0.195

Table 6.2: The p-values for the CSUQ results. Statistically significant values are boldfaced, and those less than 0.01 are shown in red. Notable values greater than 0.05 are italicized.

1=Strongly disagree; 2=Disagree; 3=Slightly disagree; 4=Neither agree nor disagree; 5=Slightly agree; 6=Agree; and 7=Strongly agree

The questions were presented as follows:

Q1 (Sys1): Overall, I am satisfied with how easy it is to use this system.

Q2 (Sys2): It is simple to use this system.

Q3 (Sys3): I can effectively complete my work (the assigned tasks and scenarios) using this system.

Q4 (Sys4): I am able to complete my work quickly using this system.

Q5 (Sys5): I am able to efficiently complete my work using this system.

Q6 (Sys6): I feel comfortable using this system.

Q7 (Sys7): It was easy to learn to use this system.

Q8 (Sys8): I believe I became productive quickly using this system.

Q9 (Info1): The system gives error messages that clearly tell me how to fix problems.

Q10 (Info2): Whenever I make a mistake using the system, I recover easily and quickly.

Q11 (Info3): The information (on-screen messages, documentation) provided with this system is clear.

Q12 (Info4): It is easy to find the information I needed.

Q13 (Info5): The information provided for the system is easy to understand.

Q14 (Info6): The information is effective in helping me complete the tasks and scenarios.

Q15 (Info7): The organization of information on the system screens is clear.

Q16 (Inter1): The interface of this system is pleasant.

Q17 (Inter2): I like using the interface of this system.

Q18 (Inter3): This system has all the functions and capabilities I expect it to have.

Q19 (Overall19): Overall, I am satisfied with this system.

The *Sys* responses were averaged to get the system usability score, the *Info* responses gave the information quality score, and the *Inter* responses gave the interface quality score. The average of all responses gave the overall score, giving four CSUQ scores per participant. Figure 6.4 shows the average scores for both conditions, for all groups of participants. The per-question breakdown of the CSUQ results can be found in Figures E.1a for PlayTIME and E.1b for Unity. Table 6.2 shows the CSUQ significance test results.

CSI

For the Creativity Support Index (CSI) [8] [9] [10] questionnaire, participants rated their agreement with 12 statements using the same Likert scale as above. The questions were presented as follows, without showing the factor names:

Q1 (Collaboration 1): The system would allow other people to work with me easily.

Q2 (Collaboration 2): It would be really easy to share ideas and designs with other people using this system.

Q3 (Enjoyment 1): I would be happy to use this system on a regular basis.

Q4 (Enjoyment 2): I enjoyed using the system.

Q5 (Exploration 1): It was easy for me to explore many different ideas, options, designs, or outcomes, using this system.

Q6 (Exploration 2): The system was helpful in allowing me to track different ideas, outcomes, or possibilities.

Q7 (Expression 1): I was able to be very creative while doing the activity inside this system.

Q8 (Expression 2): The system allowed me to be very expressive.

Q9 (Immersion 1): My attention was fully tuned to the activity, and I forgot about the system that I was using.

Q10 (Immersion 2): I became so absorbed in the activity that I forgot about the system that I was using.

Q11 (Results worth effort 1): I was satisfied with what I got out of the system.

Q12 (Results worth effort 2): What I was able to produce was worth the effort I had to exert to produce it.

Each response was adjusted to be within the correct range, and weighted by comparing their preferences between the factors in the questionnaire, resulting in a score from 0 to 100. The factors corresponding to the above questions are:

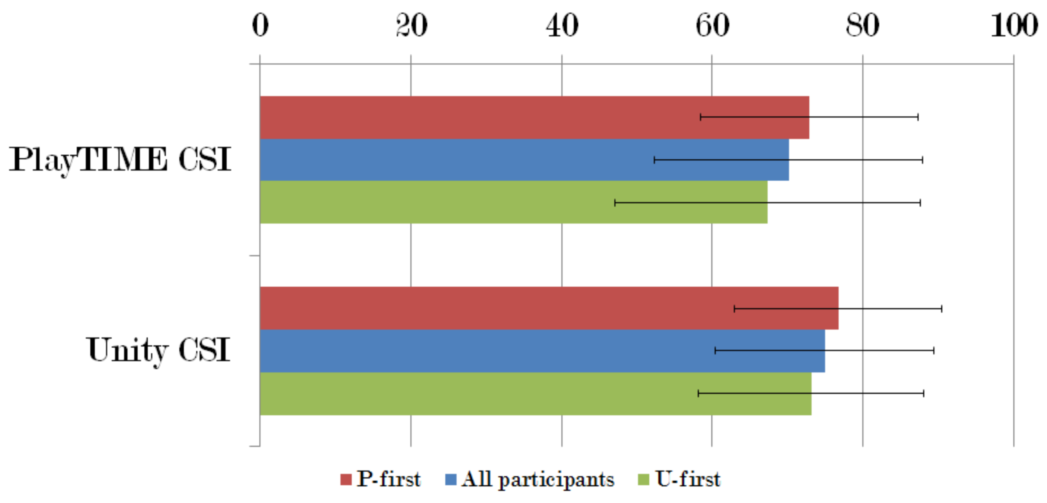


Figure 6.5: The average CSI scores for all participants between both groups, for both activities. The error bars represent one standard deviation from the mean.

Kruskal-Wallis test: *PlayTIME vs. Unity*			
	All	P-first	U-first
CSI score	0.4092	0.5964	0.5204
Kruskal-Wallis test: *P-first vs. U-first*			
		PlayTIME	Unity
CSI score		0.5964	0.3845
Wilcoxon Signed-Rank test: *PlayTIME vs. Unity*			
	All	P-first	U-first
CSI score	0.1977	0.3223	0.5073
Mann-Whitney-Wilcoxon test: *P-first vs. U-first*			
		PlayTIME	Unity
CSI score		0.6229	0.4055

Table 6.3: The p-values for the CSI questionnaire results.

Collaboration: The system would support working with other people.

Enjoyment: People enjoyed using the system.

Exploration: The system allowed the discovery of new possibilities.

Expressiveness: The system supports creativity and self-expression.

Immersion: The system supports concentration and immersion in the work.

Results worth effort: People got what they wanted by using the system; it supports time well-spent.

A CSI score of 0 indicates a system that is not conductive at all, and a score of 100 means that the system supports one or more of the questionnaire's factors extremely well (discussed further in section 6.5.6). Figure 6.5 shows the average CSI scores for both conditions, for all groups of participants, with the per-question breakdown in Figures E.2a for the PlayTIME activity and E.2b for the Unity activity. Table 6.3 shows the p-values from significance testing.

6.3.3 Post-Study Questionnaire

For the post-study questionnaire, participants were asked to rate each of the features they used during the experiment using a single ease question (SEQ) presented as a seven-point Likert scale for each question:

0=No answer (blanks); 1=Very difficult; 2=Difficult; 3=Slightly difficult; 4= Neither easy nor difficult; 5=Slightly easy; 6=Easy; and 7=Very easy

Figure 6.6 shows the average ease of use rating for both activities, with Figure E.3 showing the per-question breakdown. Figure E.4 shows the average preference ratings, with the average ratings for each question in Figure E.5. Table 6.4 shows the p-values of all the significance tests for the post-study questionnaire.

6.3.4 Performance & Time

The most data was collected while watching the recordings of participants completing the activities. This data is also where the most statistically significant differences occurred. A total of 130 actions were observed across all participants during the PlayTIME activity, and 80 during the Unity activity.

Table 6.5 shows the average time spent throughout the full activities. Figure 6.7a graphically illustrates the breakdown of the activity into three categories of tasks: completing the assigned task ("construction time"); appearing not to be doing anything at all ("activity idle time;" this is discussed further in section 6.5.5); or stopping editing for a couple of minutes to test the level in its current state ("previewing

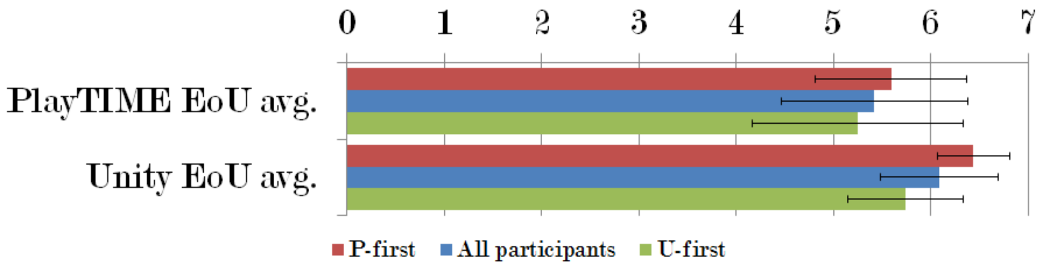


Figure 6.6: The average overall ease of use (EoU) ratings for both activities. The error bars represent one standard deviation from the mean.

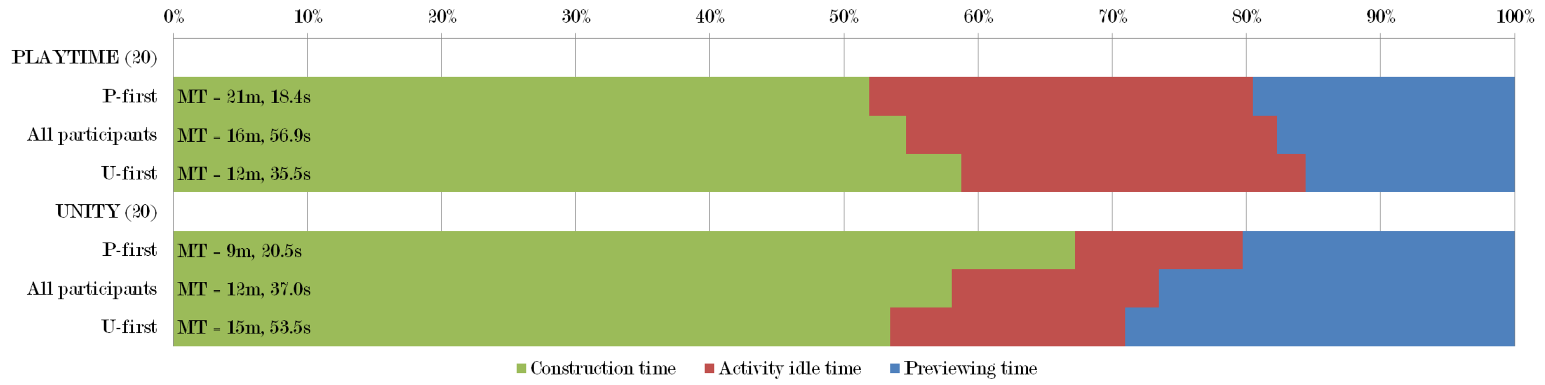
Kruskal-Wallis test: ***PlayTIME vs. Unity***			
	All	P-first	U-first
Ease of use	**0.0222**	**0.01902**	0.2723
Kruskal-Wallis test: ***P-first vs. U-first***			
	All	PlayTIME	Unity
Ease of use		0.4723	**0.01108**
Preference	0.6228		
Wilcoxon Signed-Rank test: ***PlayTIME vs. Unity***			
	All	P-first	U-first
Ease of use	**0.01173**	**0.005859**	0.3223
Mann-Whitney-Wilcoxon test: ***P-first vs. U-first***			
	All	PlayTIME	Unity
Ease of use		0.496	**0.01235**
Preference	0.6498		

Table 6.4: The p-values for the post-study questionnaire results. Statistically significant values are boldfaced, and those less than 0.01 are shown in red.

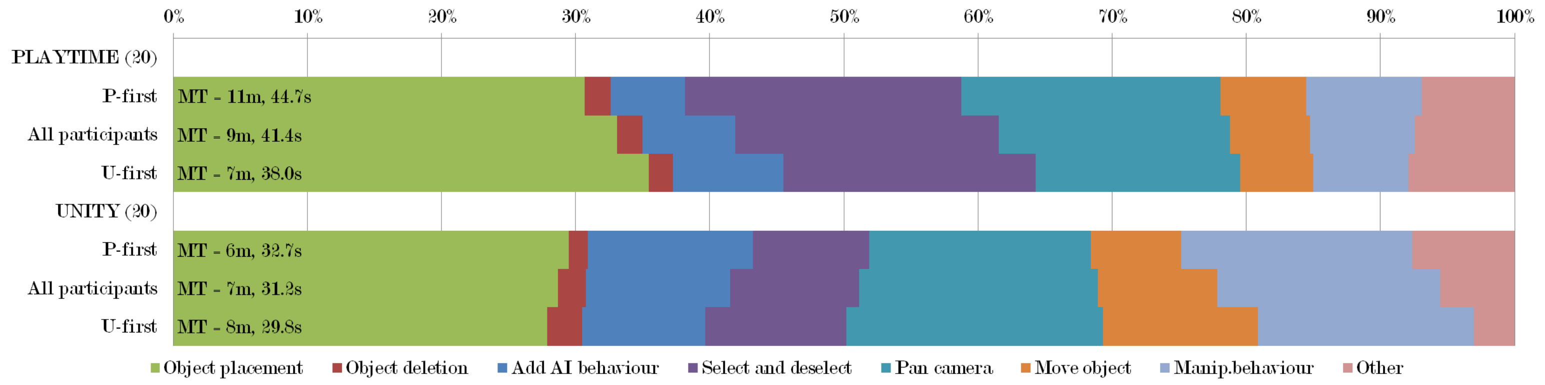
Average Times: Overall Activity						
	PlayTIME (N=20)			Unity (N=20)		
Attribute	All	P-first	U-first	All	P-first	U-first
Activity time	16:57	21:18	12:35	12:37	9:20	15:53
Construction time	9:41	11:44	7:38	7:31	6:32	8:29
Activity idle	4:54	6:28	3:20	2:00	1:13	2:47
Previewing time	3:08	4:25	2:01	3:26	1:58	4:36
Preview count	2	2	1	2	1	3
Preview users	15	7	8	18	8	10

Average Times: Level Construction						
	PlayTIME (N=20)			Unity (N=20)		
Construction task	All	P-first	U-first	All	P-first	U-first
Object placement	3:46	4:14	3:19	2:03	1:51	2:16
Object deletion	:14	:16	:10	:13	:08	:17
Add AI behaviour	:46	:45	:47	:42	:44	:41
Sel. and desel.	2:19	2:53	1:45	:47	:38	:56
Pan camera	2:03	2:38	1:28	1:24	1:07	1:40
Object movement	:43	:52	:32	:47	:31	1:08
Manip. AI	:55	1:11	:39	1:13	1:05	1:21
Deletion users	15	10	5	7	3	4
Movement users	11	6	5	18	10	8
Correct usage time	8:45	10:38	6:52	6:29	5:37	7:21
User error time	:27	:32	:22	:25	:17	:33

Table 6.5: A list of the average times spent doing different tasks during the activities. The values are approximated in minutes:seconds. The table also shows how many participants used features, if not everyone.



(a) The average time distributions for the the full activity duration. Participants were either building (construction), idling or previewing. The mean total activity time for each group is printed at the base of each bar.



(b) The average distribution of construction time doing specific tasks that can be equally compared between the two activities. The mean construction time is printed at the base of each bar.

Figure 6.7: The average time distributions for the activities. The bars represent the average percentage of time spent. Shown here are the averages for both activities, for all participant groups.

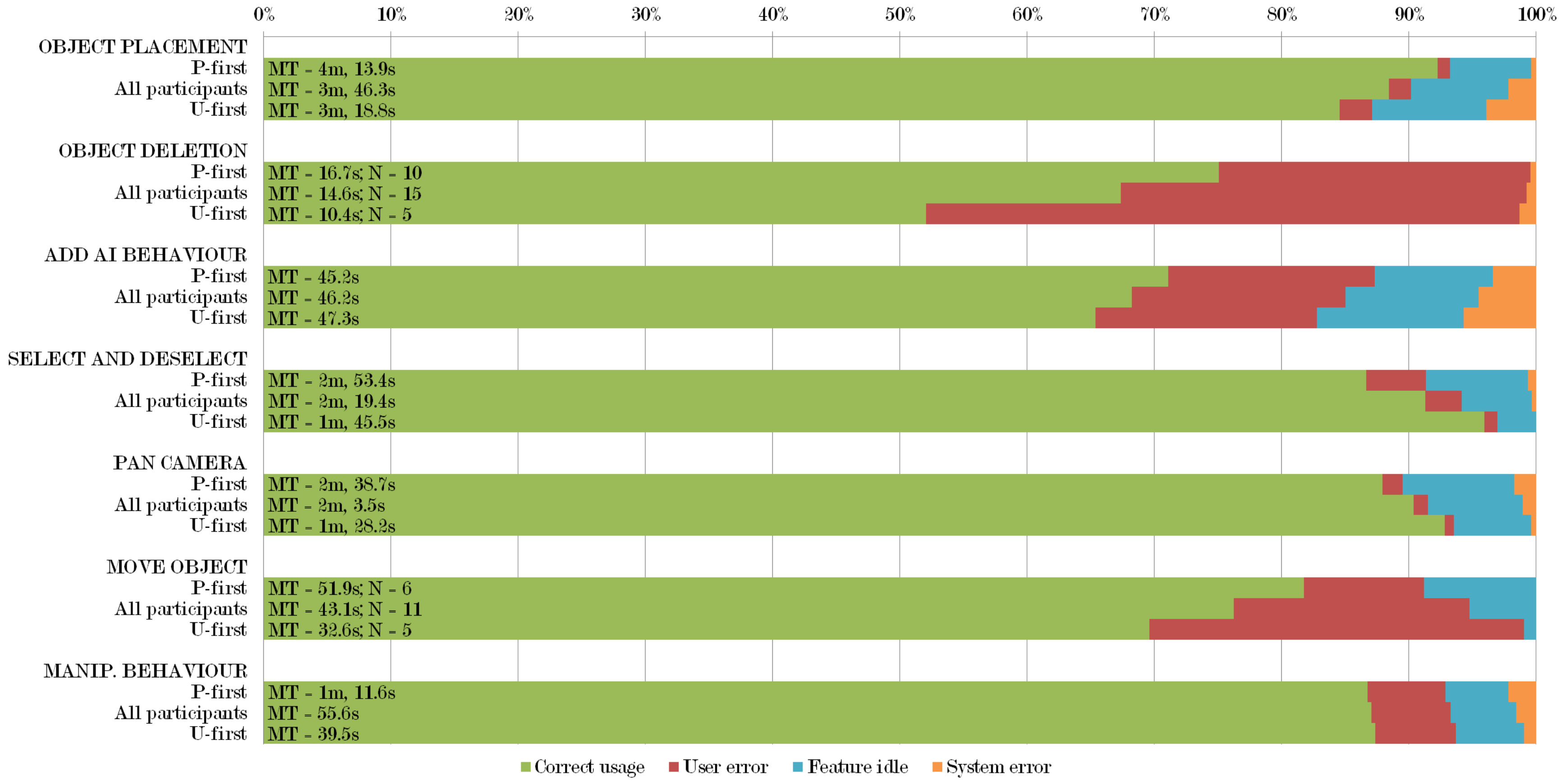


Figure 6.8: Average time distributions using the different features of PlayTIME. The number of participants from each group who used the feature is printed at the base of each bar.

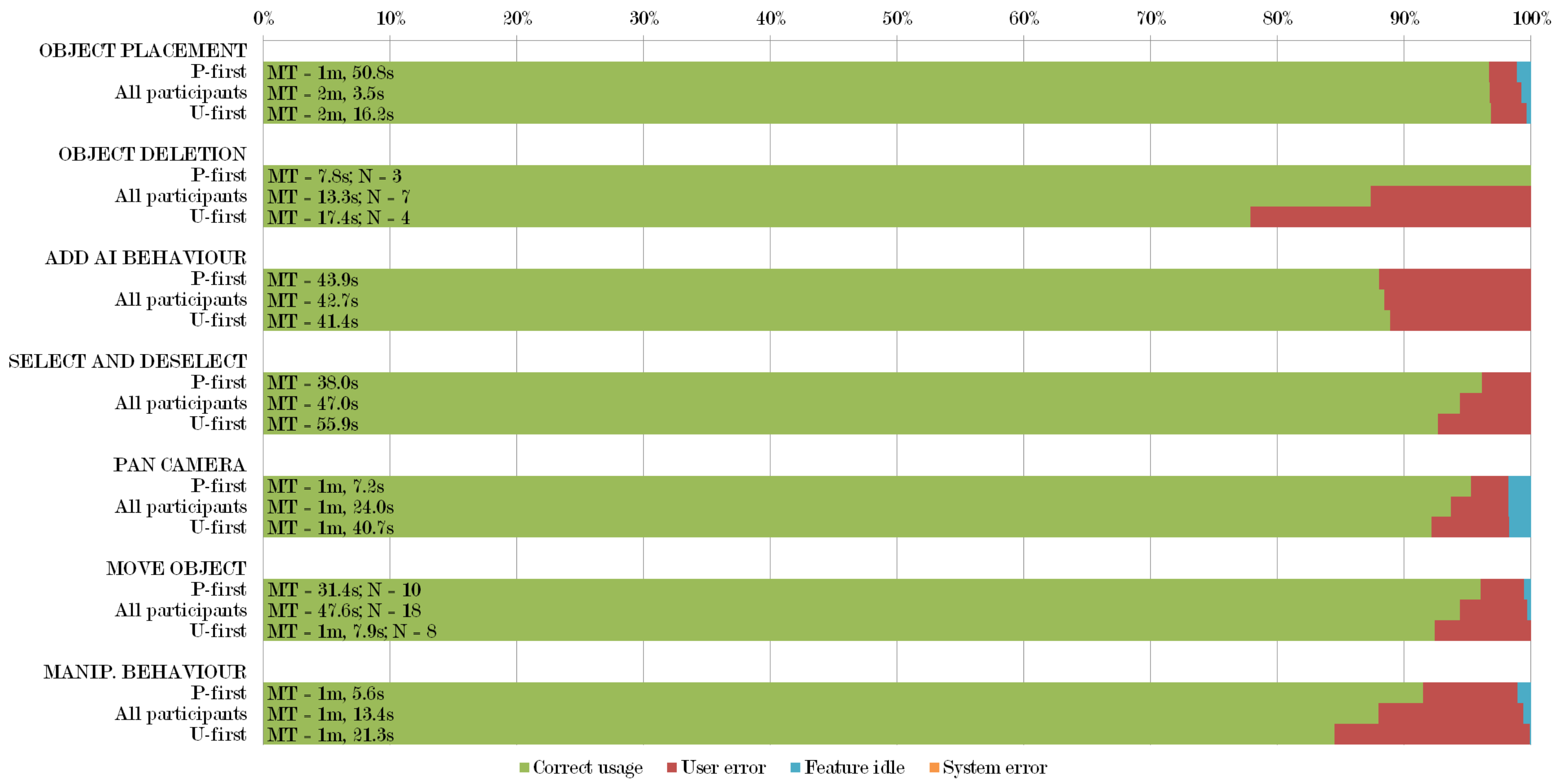


Figure 6.9: Average feature time distributions for the Unity activity.

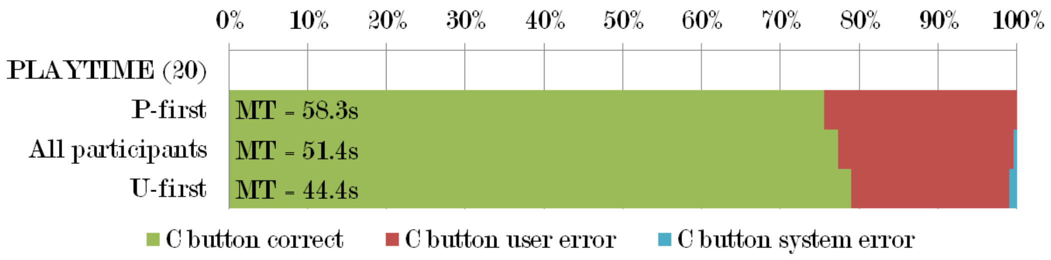


Figure 6.10: Usage of PlayTIME's C marker, or placement button. The average total time using the C button is printed at the base of each bar.

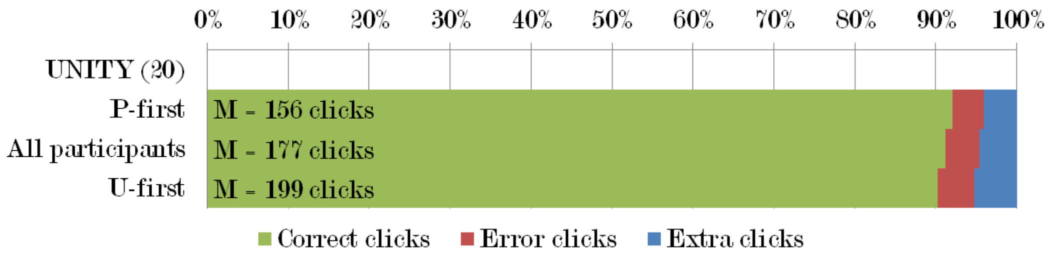


Figure 6.11: The average number of mouse clicks for each group of participants. The total clicks are printed at the base of each bar.

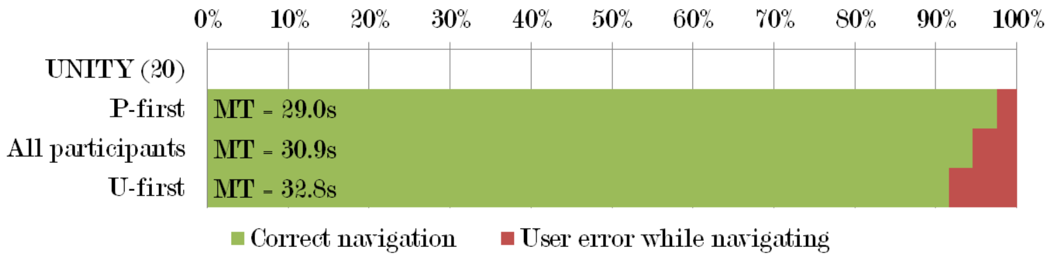


Figure 6.12: The average time distributions for navigating through Unity's assets folders during the Unity activity. The average total times are printed at the base of each bar.

Kruskal-Wallis test: *PlayTIME vs. Unity*			
	All	P-first	U-first
Total activity time	**0.02149**	**0.0001571**	0.2265
Construction time	**0.02149**	**0.00194**	0.4963
Activity idle time	**3.49E-05**	**0.0002122**	0.06964
Previewing time	0.3565	0.1478	**0.006482**
Total correct usage, % of constr. time	**0.0001699**	**0.006502**	**0.01261**
Total user error, % of constr. time	0.2914	0.7055	*0.0821*
Kruskal-Wallis test: *P-first vs. U-first*			
		PlayTIME	Unity
Total activity time		**0.0001571**	**0.005159**
Construction time		**0.0001571**	0.2568
Activity idle time		**0.01017**	**0.008151**
Previewing time		0.1704	**0.004058**
Total correct usage, % of constr. time		0.4963	0.8206
Total user error, % of constr. time		0.8206	0.0821
Wilcoxon Signed-Rank test: *PlayTIME vs. Unity*			
	All	P-first	U-first
Total activity time	*0.06958*	**0.001953**	0.1934
Construction time	**0.04844**	**0.001953**	0.4316
Activity idle time	**0.0003223**	**0.001953**	0.08398
Previewing time	0.3838	0.1073	**0.01367**
Total correct usage, % of constr. time	**0.0004826**	**0.001953**	*0.06445*
Total user error, % of constr. time	0.08255	1	**0.01953**
Mann-Whitney-Wilcoxon test: *P-first vs. U-first*			
		PlayTIME	Unity
Total activity time		**1.08E-05**	**0.003886**
Construction time		**1.08E-05**	0.2799
Activity idle time		**0.008931**	**0.006841**
Previewing time		0.1826	**0.004571**
Total correct usage, % of constr. time		0.5288	0.8534
Total user error, % of constr. time		0.8534	0.08921

Table 6.6: The p-values for performance. Statistically significant values are boldfaced, values less than 0.01 are shown in red, and other notable values are italicized.

Kruskal-Wallis test: ***PlayTIME vs. Unity***			
	All	P-first	U-first
Total object placement time	**1.92E-05**	**0.0005041**	**0.02334**
Total object deletion time	**0.01528**	**0.001542**	0.9341
Total object configuration time	0.675	0.9397	0.7054
Total selection and deselection time	**6.26E-06**	**0.0003791**	**0.006502**
Total time navigating game world	0.09351	**0.004072**	0.2567
Total object movement time	0.08501	0.704	0.06379
Total property tuning time	0.1231	0.9397	0.06964
Object placement %	**0.0009665**	0.0963	**0.004072**
Object deletion %	**0.02238**	**0.002005**	0.804
Object configuration %	**0.02655**	**0.001499**	0.8798
Select and deselect %	**8.52E-07**	**0.0006697**	**0.0003811**
World navigation %	0.7455	0.1124	0.1736
Object movement %	**0.02329**	0.2247	0.06379
Property tuning %	**0.0006537**	**0.02334**	**0.01556**
Kruskal-Wallis test: ***P-first vs. U-first***			
		PlayTIME	Unity
Total object placement time		0.0821	0.427
Total object deletion time		**0.007679**	0.4247
Total object configuration time		0.6501	0.65
Total selection and deselection time		**0.02837**	0.1508
Total time navigating game world		**0.01014**	0.1306
Total object movement time		0.3416	0.4488
Total property tuning time		0.104	0.7624
Object placement %		**0.04937**	0.7055
Object deletion %		**0.03973**	0.4247
Object configuration %		**0.04937**	0.06964
Select and deselect %		0.8206	0.4057
World navigation %		0.0821	0.2265
Object movement %		0.6344	0.4961
Property tuning %		0.4963	1

Table 6.7: The p-values for the comparable features tracked through performance analysis. Statistically significant values are bold-faced, values less than 0.01 are shown in red, and other notable values are italicized.

time"). Table 6.6 shows the significance values comparing the time distributions for each activity.

The preview and activity idle times were subtracted from the total activity time, giving us the *construction time* metric, during which the features of each system were used to complete the activity. Figure 6.7b illustrates the average distribution of feature usage during the construction time. It is important to note here that while using PlayTIME only it was possible for users to have multiple features active simultaneously; therefore the bars represent *normalized* time distributions for the purpose of showing each feature's usage compared against the others. There were 7 features or tasks that participants did to build the level. The "other" category cannot be compared since this included features that did not have a close equivalent in the other system. Table 6.7 shows the significance values comparing the use of these features or tasks for each activity.

The overall time spent using each feature was broken down further into *how* it was used. Features were either used correctly by the participant, used incorrectly (mistakes or user error); visible but not being used ("feature idle;" differs from activity idle), or misinterpreted by the system, thereby causing extra errors that were not the user's fault ("system error"). There were two types of system errors: the markers simply did not respond due to failed AR detection, or the marker was partially occluded by the user. It would not be fair to count the latter as a user error because generally it was no more than the very tip of their finger that was blocking a tiny part of the marker; this is a limitation of the AR technology since it was apparently sensitive. Figure 6.8 shows the features' average usage distributions for the PlayTIME activity, and Figure 6.9 shows the distributions for the Unity activity.

The features that were not directly and confidently comparable were categorized as *other*. For PlayTIME, this included the use of the C marker, used to place objects. Figure 6.10 shows the average usage of the C marker. For Unity, clicking was a frequent action that did not have a direct PlayTIME counterpart. The average distribution of correct, incorrect and extra or unnecessary clicks can be seen in Figure 6.11. Finally, Unity's editor required folder navigation to place things in the scene. The average distribution of navigation time can be seen in Figure 6.12. The "other" features are discussed in section 6.5.5.

6.3.5 Scenes

The final outcome of each activity, a Unity scene file, was used to compare each participant's activity outcomes with the expected activity outcome, or the instructions

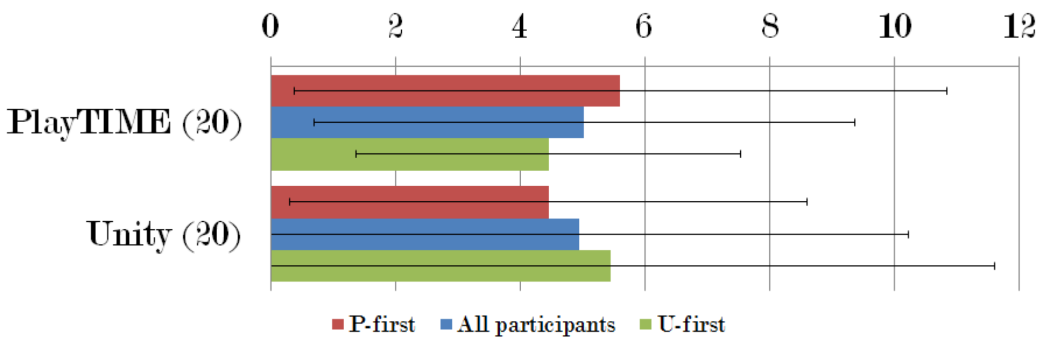


Figure 6.13: The average Manhattan scores for both activities, for each participant group. The total sample size is 20.

Enemy Placement (Group 2)						
	PlayTIME			*Unity*		
Placement attribute	P-first count	U-first count	Total count	P-first count	U-first count	Total count
Exact match with activity description	1	1	2	0	2	2
Level was *winnable*	9	10	19	10	10	20
Extra buzzers on patio	1	2	3	0	0	0
Fewer buzzers on patio	0	0	0	0	0	0
Buzzers in main room	1	1	2	0	3	3
Buzzers on balcony	1	0	1	1	2	3
Buzzers in extra rooms	2	1	3	1	3	4
Extra spiders on patio	0	1	1	1	0	1
Fewer spiders on patio	1	0	1	0	0	0
Extra spiders in main room	2	0	2	1	0	1
Fewer spiders in main room	3	2	5	3	2	5
Extra spiders on balcony	0	0	0	0	0	0
Fewer spiders on balcony	1	0	1	0	0	0
Spiders in extra rooms	3	3	6	3	3	6

Table 6.8: This table shows some of the different ways users in the surveys-only group placed objects throughout the scene. The values in the table represent the number of participants who had placed enemies in the specified location.

Kruskal-Wallis test: *PlayTIME vs. Unity*			
	All	P-first	U-first
Weighted Manhattan score	0.5758	0.7315	0.7599
Kruskal-Wallis test: *P-first vs. U-first*			
		PlayTIME	Unity
Weighted Manhattan score		0.9696	0.9089
Wilcoxon Signed-Rank test: *PlayTIME vs. Unity*			
	All	P-first	U-first
Weighted Manhattan score	0.876	1	0.7349
Mann-Whitney-Wilcoxon test: *P-first vs. U-first*			
		PlayTIME	Unity
Weighted Manhattan score		1	0.9392

Table 6.9: The p-values for the scene Manhattan scores.

they were given; it was ultimately decided that this would serve as an indication of creativity.

Figure 6.13 shows the average Manhattan scores computed for each activity. The results of the significance tests can be found in Table 6.9. The scores were calculated *specifically* based on the participants' deviations from the task description. This means that the only factor involved was the *areas on the map*; the *specific* hiding places of enemies were not accounted for when computing the scores. Table 6.8 shows how participants placed their enemies in the level, with the main areas contributing to the Manhattan scores. A few notable examples of specific placements are also shown and contribute to creativity despite not being considered in the scores. All of the results are discussed in section 6.5.6.

6.4 Extended Results

This section presents the data collected from all participants from both Group 1 and Group 2 and compares these results with those from only Group 2. The data sets that are different from those described above are the videos and the scenes. A total of 60 videos were analysed for performance statistics: 20 Unity activity recordings from Group 2 alone, and 40 PlayTIME activity recordings from all of Group 2 and some of Group 1. Additionally, a total of 100 Unity scenes were reviewed for indications of creativity: 50 PlayTIME scenes and 50 mouse scenes.

6.4.1 Extended Performance & Time

A total of 40 videos were analysed for PlayTIME and 20 were analysed for Unity. This section presents the results for the larger PlayTIME sample-size, or the whole population; the mouse results are the same as those in section 6.3.4.

The exact time averages can be found in Table 6.10. Figure 6.14a shows the average time spent through the full activities, broken down again into construction time, activity idle time and preview time. Table 6.11 shows the significance values comparing the time distributions for each activity for this larger PlayTIME population compared with the same mouse population.

Figure 6.14b shows the feature usages for this population compared with the data analysed for the Unity activity. Table 6.12 shows the significance values comparing the use of these features.

Figure 6.15 shows the features' average usage distributions for the larger PlayTIME population, and Figure 6.16 shows the average usage of the C marker.

20 vs. 40 Participants

Here we compare the direct effect of *population size* on the *effects of condition and order*. We used the KW test and the MWW test to compare the p-values from the smaller population of 20, within Group 2 alone, with the p-values from the larger population of 50, accounting for all performance data from both groups. This was a last minute decision to see if changing the sample size to acquire the extended results had a direct impact on the significant test outcomes.

The tests agree that comparing data with different population sizes had an *insignificant* effect on the outcome of the other tests. This means that the results were relatively similar when comparing the 20 PlayTIME data sets with the 20 Unity sets, and when comparing the 40 PlayTIME data sets with the 20 Unity data sets. Figure 6.17 clearly illustrates this by comparing the time distributions from Group 2 alone with the distributions from both Group 1 and Group 2. The p-values for the effect of changing the population size on performance analysis are shown in Table E.1.

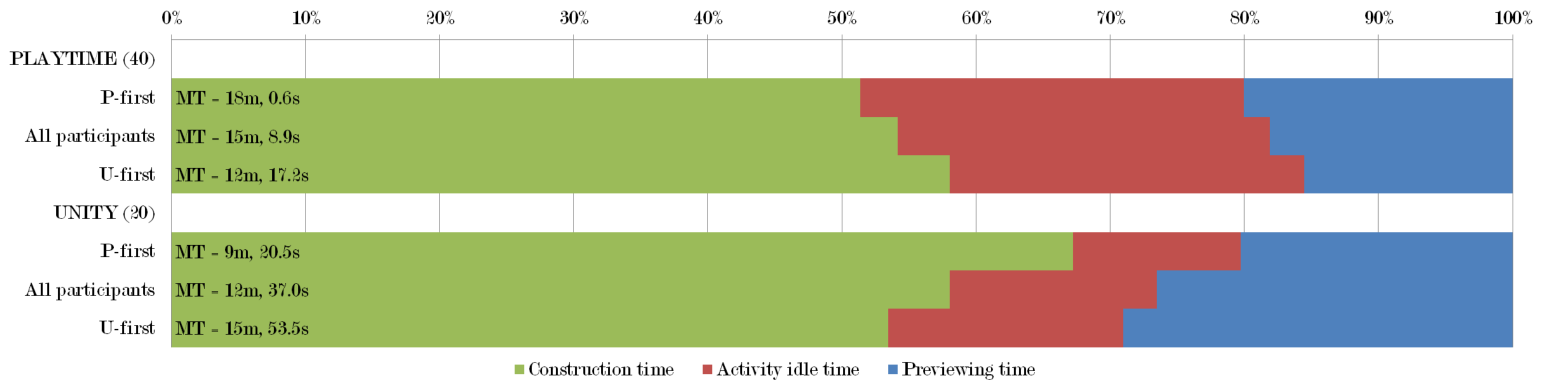
6.4.2 Extended Scenes

For the extended scene data, the scenes from *all participants* of both Group 1 and Group 2 were examined, and a Manhattan score computed for each, yielding 50 samples for each activity. Figure 6.18 shows the average Manhattan scores computed for each activity across the whole population. The results of the significance tests can be found in Table 6.14. Table 6.13 shows the enemy placements for this population.

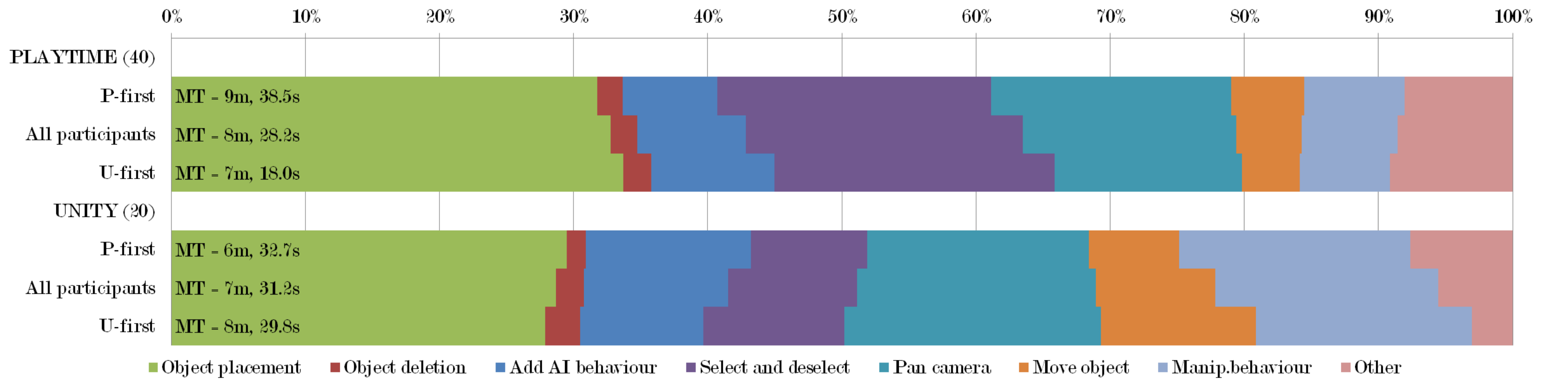
Average Times: Overall Activity						
	PlayTIME (N=40)			Unity (N=20)		
Attribute	All	P-first	U-first	All	P-first	U-first
Activity time	15:09	18:00	12:17	12:37	9:20	15:53
Construction time	8:28	9:38	7:18	7:31	6:32	8:29
Activity idle	4:20	5:21	3:19	2:00	1:13	2:47
Previewing time	2:49	3:45	1:57	3:26	1:58	4:36
Preview count	2	2	1	2	1	3
Preview users	33	16	17	18	8	10

Average Times: Level Construction						
	PlayTIME (N=40)			Unity (N=20)		
Construction task	All	P-first	U-first	All	P-first	U-first
Object placement	3:15	3:33	2:58	2:03	1:51	2:16
Object deletion	:13	:15	:10	:13	:08	:17
Add AI behaviour	:46	:43	:48	:42	:44	:41
Sel. and desel.	2:07	2:22	1:53	:47	:38	:56
Pan camera	1:37	2:01	1:13	1:24	1:07	1:40
Object movement	:33	:40	:25	:47	:31	1:08
Manip. AI	:44	:52	:38	1:13	1:05	1:21
Deletion users	29	15	14	7	3	4
Movement users	21	11	10	18	10	8
Correct usage time	7:41	8:44	6:37	6:29	5:37	7:21
User error time	:27	:29	:25	:25	:17	:33

Table 6.10: A list of the average times spent doing different tasks during the activities for the extended population. The values are approximated in minutes:seconds.



(a) The average time distributions for the the full activity duration.



(b) The average distribution of construction time doing specific tasks that can be equally compared between the two activities for the same populations.

Figure 6.14: The extended average time distributions, comparing the 40 analysed samples for the PlayTIME activity with the 20 from the Unity activity. The bars represent the average percentage of time spent.

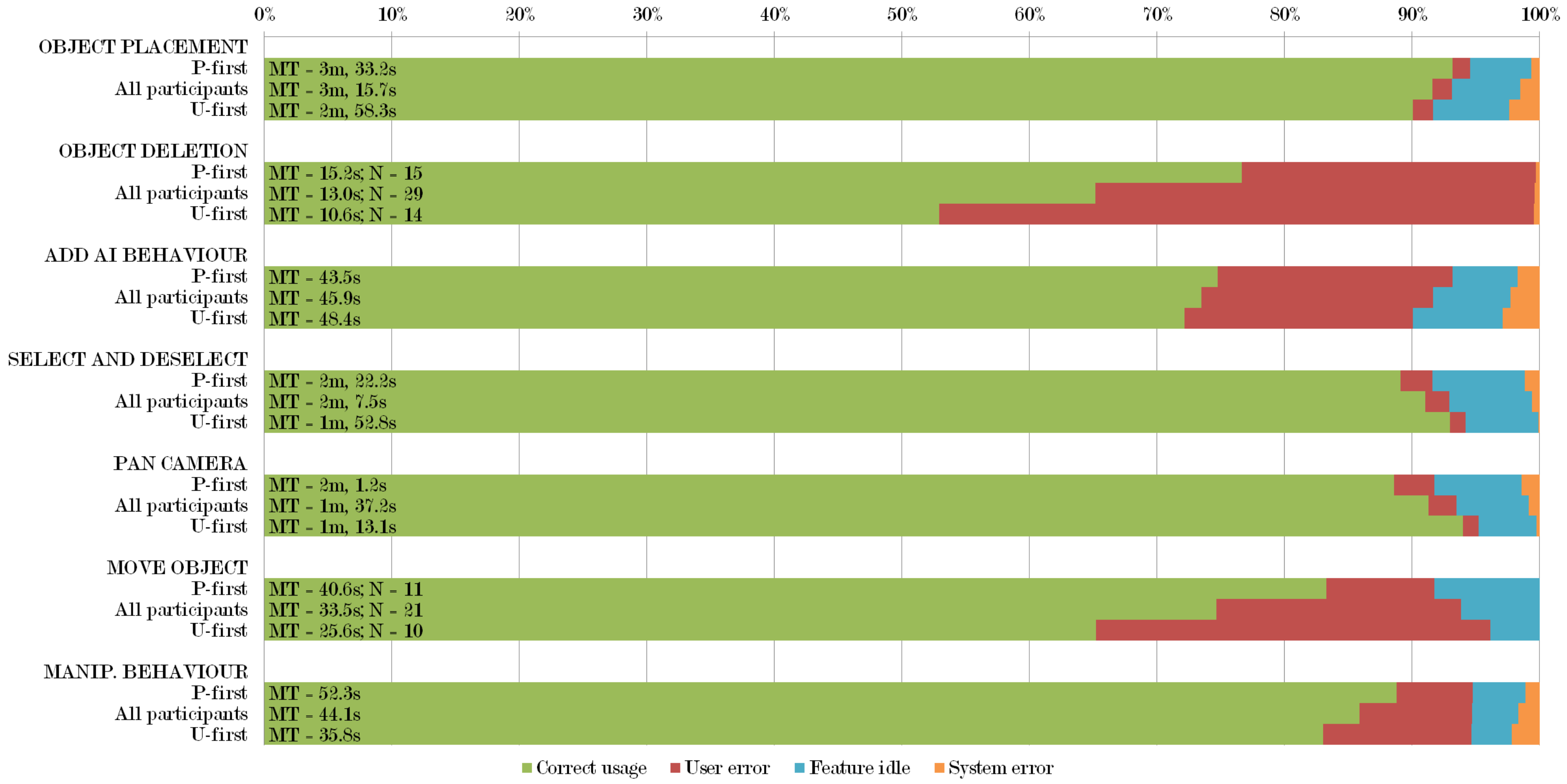


Figure 6.15: Feature time distributions for the extended PlayTIME performance samples (40). The total number of users for each feature is printed at the base of each bar.

Kruskal-Wallis test:
PlayTIME vs. Unity

	All	P-first	U-first
Total activity time	*0.05573*	**0.0001082**	0.1236
Construction time	0.1725	**0.01553**	0.3116
Activity idle time	**4.02E-06**	**1.62E-05**	*0.05852*
Previewing time	0.3301	**0.03401**	**0.001314**
Total correct usage, % of constr. time	**5.96E-05**	**0.004868**	**0.005578**
Total user error, % of constr. time	0.4148	0.7581	0.1869

Kruskal-Wallis test:
P-first vs. U-first

		PlayTIME	Unity
Total activity time		**4.42E-05**	**0.005159**
Construction time		**0.006294**	0.2568
Activity idle time		**0.00117**	**0.008151**
Previewing time		**0.01079**	**0.004058**
Total correct usage, % of constr. time		0.5518	0.8206
Total user error, % of constr. time		0.4989	0.0821

Mann-Whitney-Wilcoxon test:
P-first vs. U-first

		PlayTIME	Unity
Total activity time		**1.34E-05**	**0.003886**
Construction time		**0.005618**	0.2799
Activity idle time		**0.0008358**	**0.006841**
Previewing time		**0.01122**	**0.004571**
Total correct usage, % of constr. time		0.5648	0.8534
Total user error, % of constr. time		0.5117	0.08921

Table 6.11: The p-values for extended performance analysis. Statistically significant values are bold-faced, values less than 0.01 are shown in red, and other notable values are italicized.

Kruskal-Wallis test: ***PlayTIME vs. Unity***			
	All	P-first	U-first
Total object placement time	**4.56E-05**	**0.0005091**	**0.02783**
Total object deletion time	**0.01228**	**0.01163**	0.3396
Total object configuration time	0.5674	0.895	0.895
Total selection and deselection time	**8.48E-07**	**0.0001292**	**0.002775**
Total time navigating game world	0.4902	**0.01375**	*0.0585*
Total object movement time	**0.007796**	0.1808	**0.02301**
Total property tuning time	**0.001305**	0.07838	**0.007276**
Object placement %	**0.000202**	**0.03111**	**0.003689**
Object deletion %	**0.01345**	**0.015**	0.2557
Object configuration %	0.1282	**0.003689**	0.5674
Select and deselect %	**4.50E-09**	**4.29E-05**	**2.41E-05**
World navigation %	0.9625	0.1236	0.09457
Object movement %	**0.0008567**	**0.04024**	**0.0109**
Property tuning %	**4.34E-06**	**0.0006002**	**0.002775**
Kruskal-Wallis test: ***P-first vs. U-first***			
		PlayTIME	Unity
Total object placement time		0.2036	0.427
Total object deletion time		0.2624	0.4247
Total object configuration time		0.8817	0.65
Total selection and deselection time		0.1298	0.1508
Total time navigating game world		**0.001063**	0.1306
Total object movement time		0.4396	0.4488
Total property tuning time		0.194	0.7624
Object placement %		0.1298	0.7055
Object deletion %		0.6818	0.4247
Object configuration %		0.1105	0.06964
Select and deselect %		0.6263	0.4057
World navigation %		**0.01383**	0.2265
Object movement %		0.647	0.4961
Property tuning %		0.8077	1

Table 6.12: The p-values for the extended comparable features tracked through performance analysis. Statistically significant values are bold-faced, values less than 0.01 are shown in red, and other notable values are italicized.

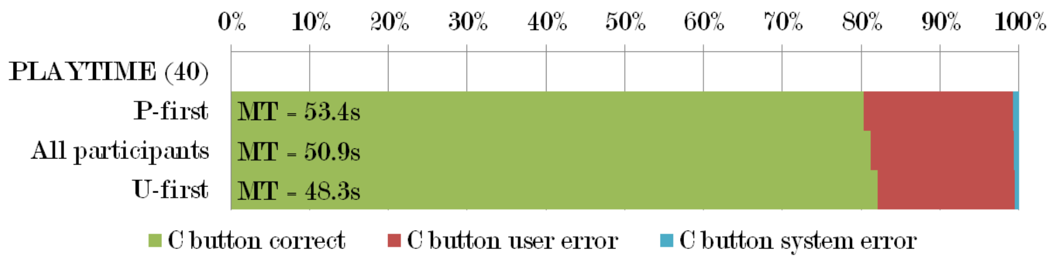
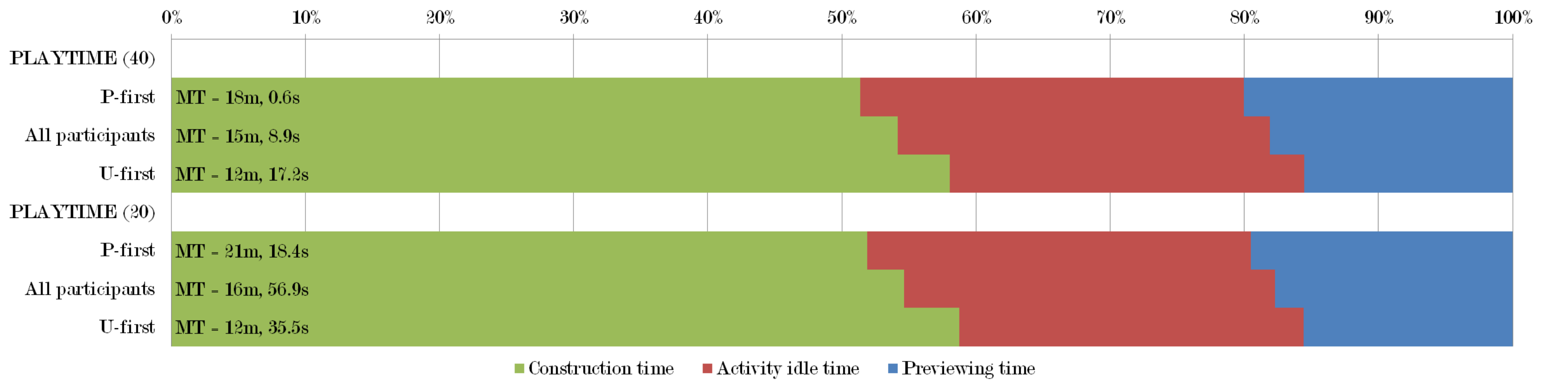


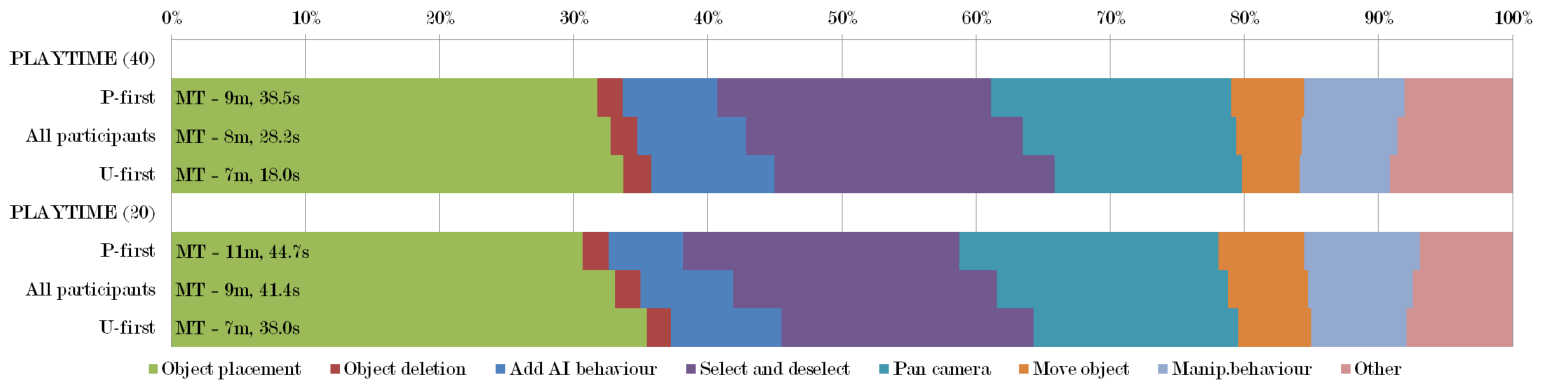
Figure 6.16: Average usage of PlayTIME's C button for the extended PlayTIME performance samples (40). The average time spent using the C button is printed at the base of each bar.

Enemy Placement (full population)						
	PlayTIME			*Unity*		
Placement attribute	P-first count	U-first count	Total count	P-first count	U-first count	Total count
Exact match with activity description	5	5	10	5	9	14
Level was *winnable*	23	25	48	25	25	50
Extra buzzers on patio	1	3	4	0	0	0
Fewer buzzers on patio	1	0	1	0	0	0
Buzzers in main room	2	1	3	1	4	5
Buzzers on balcony	1	0	1	1	3	4
Buzzers in extra rooms	2	1	3	1	3	4
Extra spiders on patio	0	1	1	1	0	1
Fewer spiders on patio	1	0	1	0	0	0
Extra spiders in main room	2	2	4	1	0	1
Fewer spiders in main room	4	4	8	4	3	7
Extra spiders on balcony	0	0	0	0	0	0
Fewer spiders on balcony	1	0	1	0	0	0
Spiders in extra rooms	4	5	9	4	4	8

Table 6.13: This table shows some of the different ways users in the full population, Groups 1 and 2, placed objects throughout the scene.



(a) Comparing the average time distributions for the two PlayTIME activity populations (40 vs. 20).



(b) Comparing the average construction time distributions for the PlayTIME activity populations.

Figure 6.17: Comparing all 40 analysed samples for the PlayTIME activity with only the 20 that were associated with surveys.

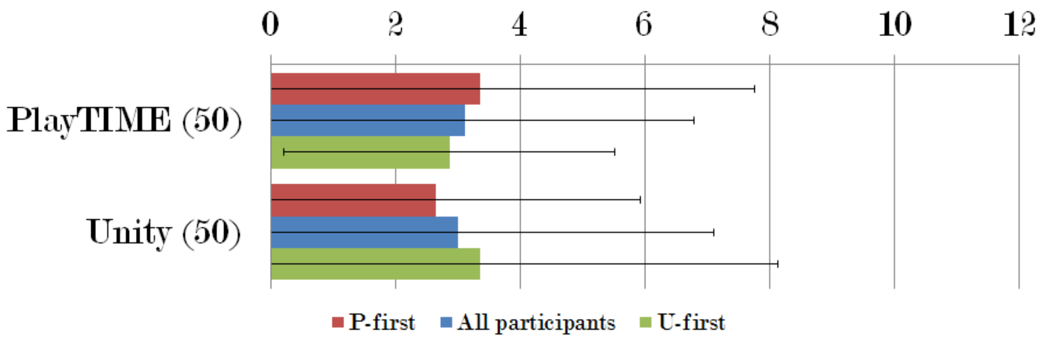


Figure 6.18: The average extended Manhattan scores for both activities, for each participant group. The total sample size is 50.

Kruskal-Wallis test: ***PlayTIME vs. Unity***			
	All	P-first	U-first
Weighted Manhattan score	0.4689	0.8068	0.5681
Kruskal-Wallis test: ***P-first vs. U-first***			
		PlayTIME	Unity
Weighted Manhattan score		0.703	0.9216
Wilcoxon Signed-Rank test: ***PlayTIME vs. Unity***			
	All	P-first	U-first
Weighted Manhattan score	0.8166	0.6767	0.856
Mann-Whitney-Wilcoxon test: ***P-first vs. U-first***			
		PlayTIME	Unity
Weighted Manhattan score		0.7103	0.9295

Table 6.14: The p-values for the extended scene Manhattan scores.

20 vs. 50 Participants

Changing the population size from 20 while analysing the scenes from Group 2, to 50 while analysing the scenes from both Group 1 and Group 2 had a significant effect on the condition and order's effects. This effect is discussed in section 6.5.6. The exact p-values representing the effect of this change are shown in Table E.2.

6.5 Discussion

6.5.1 PANAS & Emotions

Before the study, the average positive affect score was 29.25 (SD=7.354) and the average negative affect score was 12.85 (SD=3.103). Through the PlayTIME activity, the average overall positive affect change was +2.95 points (score M=33.45, SD=8.084), and the average overall negative affect change was -0.85 points (score M=11.45, SD=1.962). Through the Unity activity, the average overall positive affect change was -1.5 points (score M=30.45, SD=8.273), and the average overall negative affect change was -1.25 points (score M=10.7, SD=0.9).

Positive Affect

In Figure 6.2a, we see that the average positive affect change was +5.4 points through the PlayTIME activity (score M=34.9, SD=7.739) and -5.5 points through the Unity activity (score M=29.4, SD=9.276). Figure 6.2b shows that for the Unity-first group, the average positive affect change was +2.5 points through the Unity activity (score M=31.5, SD=6.975) and +0.5 points through the PlayTIME activity (score M=32, SD=8.161).

We can see that the *order* of the activities had a statistically significant effect on the positive affect change *through* both activities, which means the activities themselves had a different impact on positive feelings depending on the activity order.

The KW p-value testing order through the PlayTIME activity is 0.01238; this is agreed upon by the MWW p-value of 0.01377. This significance is visualized by the trend lines in Figure 6.2, which clearly show the differences between the emotions of the participants who did the PlayTIME activity first and those who did the Unity activity first. Both groups had an increase in positive affect through the PlayTIME activity, but the P-first group shows a much steeper slope. Testing order through the Unity activity, the KW p-value of 0.004003 and the MWW p-value of 0.00451 both indicate extreme significance. The trend lines show that the P-first group had a *negative* change in positive affect, but the U-first group had a *positive* change through the Unity activity. The complete change of direction through this activity would explain the extreme significance.

Looking at the positive affect change through both conditions, we notice that the activities themselves had a significant impact on the positive affect changes. The steep and always-positive changes through the PlayTIME activity are quite different from the negative and flat slopes through the Unity activity. The KW p-value for all 20 participants is 0.02532, and for the P-first group only the p-value is 0.0005718. However, the test did not find a significant difference in the positive affect change between conditions for the U-first group.

Despite the significance of the change in positive affect through the activities, the raw positive affect *scores* are not significantly impacted by the order or by the activities. The WSR test found that changing the condition had a significant effect on the positive affect scores, but the KW test does not agree (Table 6.1). These tests were double checked using *R*.

Negative Affect

Figure 6.2a shows the P-first group's average negative affect change was -1.8 points through the PlayTIME activity (score M=12.4, SD=2.245) and -1.4 points through the Unity activity (score M=11, SD=1). Figure 6.2b shows the U-first group's average negative affect change was -1.1 points through the Unity activity (score M=10.4, SD=0.663) and +0.1 points through the PlayTIME activity (score M=10.5, SD=0.922).

While the U-first group's positive affect changes are not significantly impacted by the activities, their negative affect changes were. The KW test returned a p-value of 0.02289 for this group, and the WSR test returned 0.04983.

Once again, the activity order had a notable effect: the KW p-value for negative affect through the PlayTIME activity is 0.04701, with the MWW p-value not quite there but still close (0.05159). The trend lines demonstrate this as well: for the P-first group, negative affect decreased quite a bit through the PlayTIME condition, but increased slightly for the U-first group, staying basically the same through the activity. The slope differences would explain why there is significance through the PlayTIME activity but not through the Unity activity; these slopes are basically the same.

It also appears that the order had an effect on the raw negative affect scores from the PlayTIME activity only. The KW p-value is 0.01674 and the MWW p-value is 0.01864. On the trend lines we see that the average negative affect scores for the

PlayTIME activity are different between the two groups, but the scores for the Unity activity are very close.

For the activities' impact on raw negative affect scores, the WSR test found slight significance for all participants, but the KW test does not agree (Table 6.1).

Discussion

The PANAS questionnaire was a simple and fun way to measure the emotional impact of the study activities on the participants. The PlayTIME activity caught participants' interest the moment they figured out what it would be used for, sometimes before they were briefed, and overall participants reported that their positive emotions had increased significantly during the activity. Furthermore, their negative emotions, which were low to begin with, had generally decreased. In contrast, the Unity activity generally had a much smaller or opposite effect on the positive affect scores. Although this measure does not contribute to system usability, the consistently positive emotions tell us something about PlayTIME that is important for users: *it is fun.* Perhaps seeing and using this novel technique for scenario design was exciting and enjoyable for users because it felt like play.

Some of the participants commented on this in the feedback portion of the post-study questionnaire:

> *"I enjoyed PlayTIME more... It also made it seem more like playing a game to create a game; I was almost more interested in making [my game] than playing it."*
> - Participant 50-P

> *"It was far more interesting to be using technologies that I'd never used before, whereas I am almost always using just a mouse and keyboard, or as in this case, a mouse alone."*
> - Participant 45-P

> *"PlayTIME was definitely more enjoyable because it felt like I was playing a game as opposed to working."*
> - Participant 43-M

> *"It was like playing rather than working."*
> - Participant 41-M"

This is great for scenario designers, since one of PlayTIME's purposes is to have game development feel more interactive by treating it like play. To see people agreeing with this purpose, that PlayTIME was indeed playful and fun, means it is doing its job at making game design more conductive.

6.5.2 CSUQ & Usability

Here we discuss the qualitative aspect of usability: *How did participants respond to what they did during the study?* The self-reported survey metrics tell us how participants received the systems they were given to complete their activities and how well they thought the systems worked.

Overall, the P-first group gave slightly higher ratings, as seen in Figure 6.4 and Figure E.1, but the activity order did not have a significant impact on the results (Table 6.2). For the PlayTIME responses, the interface quality of the AR paddles received the highest ratings across all participants (M=5.283, SD=1.221). The mouse received similar ratings (M=5.283, SD=1.217), but as discussed above, these ratings were regarding the mouse itself and not Unity's GUI as they should have been. As well, the interface quality metric was based on just 3 questions. Overall the condition's effect on the interface ratings were not at all significant (KW p=1, WSR p=0.9055).

More importantly, the system usability scores were generally good and were averaged over 8 of the questions. For PlayTIME, the overall system usability score averaged 5.069 (SD=1.166) and for Unity the overall average was 5.688 (SD=1.228). The system usability results between conditions were significantly different (KW p=0.03021, WSR p=0.009305). The effect was more significant on the P-first group, but the statistical tests found different significance levels (KW p=0.05327, WSR p=0.005666).

Interestingly, the lowest of all CSUQ ratings were consistently from Q9, regarding error messages. For PlayTIME the overall scores for Q9 averaged 2.7 (SD=1.382) and for Unity the scores averaged 2.85 (SD=1.74). Errors occurred often, and were frequently repeated, because neither system provided any indication that something had gone wrong in the first place. In some cases, users would commit an error and not realize until they visited that location in the map 5 minutes later. For example: with PlayTIME, objects were frequently left selected while working on a completely different area of the level, and an accidental or intentional occlusion of the B marker would cause the off-screen objects to be deleted. The exact user error statistics are explored below in the screen capture and performance discussion, but the point is that users learned by correcting their mistakes. They were also allowed to ask for help when things got too frustrating, in which case the observer would describe the necessary steps to correct the error.

This goes back to two of Nielsen's usability heuristics: error prevention, and error recognition, diagnosis and recovery [45] [47]. Based on the user responses and performance, as we will see below, both of the systems do a poor job at letting users know when an error has occurred. Furthermore, the fact that some people overlooked errors for several minutes, or never discovered their mistakes at all, tells us that PlayTIME has an opportunity to satisfy these heuristics. Specific errors are discussed per-feature in section 6.5.5.

6.5.3 Ease of Use & Preference

The ease of use questions in the post-study survey looked at the usability of the interfaces used for each activity: PlayTIME's AR paddles and the mouse for Unity. Figure E.3 shows the overall ease of use scores per-group and the breakdown of the scores.

The overall average ease of use rating for PlayTIME is 5.422 (SD=0.959) and the average rating for Unity is 6.089 (SD=0.603). A statistically significant effect was found on the overall scores due to the condition, for the whole population, by both the KW test (p=0.0222) and by the WSR test (p=0.01173). The effect was also significant within the P-first group (KW p=0.01902, WSR p=0.005859). The activity order had a highly-significant effect on the ease of use scores from the U-first group, as shown by the KW test (p=0.01108) and by the MWW test (p=0.01235). The reason for the highly significant results is simply because we are so familiar with the mouse.

The overall average preference rating was -0.13 (SD=0.43), indicating a preference for the mouse over the AR paddles. Since preference was a direct comparison between the conditions, the scores were not tested against the activities themselves, only activity order, which yielded no significance.

The actual factors rated in the ease of use and preference questionnaires are discussed throughout section 6.5.5.

6.5.4 Activity Time Overview

The screen captures were extremely helpful with identifying problems with the interfaces. Since the surveys do not tell us much about the interface limitations, as discussed above, the performance statistics provide most of the identification of users' troubles completing the study. The screen capture presents us with raw, quantitative usability facts: *What did the participants do during the study?*

Table 6.5 shows the overall activity times for the surveys-only group, and Table 6.10 shows the times for the full population. These distributions are visualized in Figures 6.7a and 6.14a respectively.

The overall average activity time for the surveys-only PlayTIME population was 16:57 (mm:ss). For the full population, the activity time was 15:09. The average Unity time was 12:37.

The overall averages were balanced by order: for PlayTIME Group 2, the P-first average was 21:18 and the U-first average was 12:35. For the full population, the P-first average was 18:00 and the U-first average was 12:17. For the Unity activity, the P-first average was 9:20 and the U-first average was 15:53.

The fact that all of the PlayTIME numbers were generally higher by several minutes is a clear indication that the condition had a significant effect on the outcome. This is agreed upon by the p-values: the KW test returned 0.02149 for both the overall activity time and construction time, and 0.0001571 for activity time in the P-first group and 0.00194 for construction time.

The significance values testing *condition* also implicitly tell us that order had an effect, given that the p-values in the U-first group were obviously higher. With the exception of previewing time and total user error time, which remained basically the same for all groups, notice how the P-first group had consistently high significance compared to U-first. This is likely because the activity times for the people who ran PlayTIME first were quite far apart, whereas the people who ran Unity first had less of a difference between their times. Notice how the participants' *first* activity always had longer times. This is a clear indication that order had a significant effect on the activity time as a whole and in parts, as shown by the P-values for both activities. The KW test returned 0.0001571 for PlayTIME and 0.005159 for Unity; these indicate extremely high significance. The feedback leads us to a very logical explanation for this.

The final question in the post-study survey was "*Did doing the first activity help you complete the second activity in any way?*" *All participants* reported that the *order had a definite impact on their results.* For Group 1 this was asked verbally at the end of the study and all participants said yes. Those who cited reasons generally agreed that the reason was simple: since they had already done the activity once, they were able to *skip the learning process* for the task and just complete the activity, which they had memorized; the learning curve for the system itself may have still been an issue.

The main reason cited in the participants' responses was that the first activity helped them prepare for the second; they learned the activity itself, what they were required to do, the level layout, placement locations, design, etc. (summarized here as "the task"). To quote a few responses, participant 48-U said that "[The order] let me know what needed to go where," meaning that they had the task and level layout memorized from their first activity. Participant 50-P similarly said "Knowing what I was supposed to do already is about it," which means the same thing. This was the most common response, explicitly coming from 13 participants in Group 2 and *everyone* from Group 1 (which was written in notes since it was verbal). Another 5 participants did not directly state the memorization or familiarity factor but implied it in their responses; for example, 49-P said "I understood everything with the mouse, it was simply [a matter of] remembering it and applying it to PlayTIME." Another participant, 46-U said "The first activity (in Unity) refreshed knowledge from past experiences with [Unity]... the second activity was more intuitive and user-friendly." The remaining two participants just said "Yes," which just says that order affected them for some reason.

Participant 39-U had an interesting response: "Doing the first activity did give me an advantage in the second activity; however, it was mitigated by the fact that I needed to learn a new system in order to complete the task." This is interesting because it implies familiarity but also acknowledges the challenge of having to use a new system; perhaps the advantage of order was balanced by the learning curve of the second activity. This makes sense since the activity times for the people who ran PlayTIME first are so far apart, whereas the people who ran Unity first have similar activity times. Since users were already familiar with Unity and the mouse, those who ran that activity first were only hindered by the *task*, and by the time they got to their second activity they were familiar with the task and their time was affected by the *learning curve of PlayTIME*. In contrast, those who ran PlayTIME first had to deal with both the task *and* the unfamiliarity of the new system; so when they got to Unity they were familiar with both the task and the system, which increased their times dramatically.

This can be demonstrated simply by taking the differences between activity times for corresponding groups: The P-first PlayTIME average time was 21:18 (Group 2 only). If we subtract that group's average Unity time of 9:20, we are left with an average difference of 11:58 between their two activities. Likewise for the U-first group, 15:53 for Unity subtract 12:35 for PlayTIME gives us an average difference of only 3:18. This difference alone emphasizes the effect of order, but we see it with construction time as well: For P-first the difference (computed the same way) is 5:12, while for U-first the construction time difference is only 51s. Tables 6.5 and 6.10 clearly show that *all* tasks took longer on average for participants' *first* activity, and therefore this justification for the significance of order carries through all discussion sections below. The effect that *order* had on the completion of tasks will not be discussed in detail past this point.

Most of the results pertain to time measurements for the individual features. Section 6.3.4 only summarizes the data using bar charts for time averages and tables for the main activity details and significance test results. Construction, idle, previewing and feature times are discussed separately in further detail below.

Previewing Time

Previewing was not included in the construction time because we were concerned with the level *construction*. Since previewing was restricted to the use of *both mouse and keyboard*, the feature did not make use of the *system* interfaces in a similar fashion and therefore it would not have been fair to the *data* if previews had been included.

Still it is important to note the differences. In Group 2 alone, the average overall previewing time was 3:08 for PlayTIME and 3:26 for Unity. The whole population overall average for PlayTIME was 2:49. Split into groups, the surveys-only PlayTIME averages were 4:25 for P-first and 2:01 for U-first. For Unity, the averages were 1:58 for P-first and 4:36 for U-first. The averages for the whole population in PlayTIME were 3:45 for P-first and 1:57 for U-first. In the surveys-only group, 15 participants used previewing and 33 from the full population for PlayTIME. For Unity 18 people used the preview feature.

The p-values show that condition had a significant effect within groups only (not overall), and this implicitly tells us that the activity order had a significant effect; the p-values for order explicitly show us that the order had a significant effect. Similar to the overall activity times, the preview times were noticeably higher for each activity within the group that did that activity first, and there were also more previews. A logical reason for this significance is that having done the experiment already had given users the experience with the *task* that they needed to *trust the quality of their design*, and the preview itself. There simply was not as much of a need to preview by the time they got to the second activity.

Another briefly notable difference with previewing is the time it took to load. Preview load and unload times were an issue in the TimeSplitters study (approximately 15 seconds both ways) and this bothered some participants. Unity was chosen as the editor for the current implementation of PlayTIME for its quick preview feature, but PlayTIME hindered this feature.

The average time per-preview for *all* previews was about 1:23 for the PlayTIME activity and 1:18 for the Unity activity, or about 94% of PlayTIME's previewing time. However, the average time per-*load* for all previews was about 9.7 seconds for PlayTIME and 6.2 seconds for Unity alone, which is 64% of PlayTIME's load time. Furthermore, the average time per-*unload* was about 8.5 seconds for PlayTIME and 3.7 seconds for Unity, which is 43% of PlayTIME's unload time.

A small portion of the load and unload durations was the time it took to press "play" button for both starting and stopping. However, the wait time was really impacted by the actual preview load and unload times. The reason for this is because the PlayTIME system was programmed to *terminate* while a preview was running. The low-level details are not important, but because of the way Unity handles its previews, PlayTIME was required to tap into this process so that AR would not interfere with the preview; the markers were not needed for the *game*, only for development. Therefore the AR detection system was shut down, disabling the camera. The AR markers were unloaded. In preview state PlayTIME was completely deactivated. When the play button was clicked to end the preview, the inverse occurred: PlayTIME was completely rebooted and the Unity scene was restored to its development state. In contrast, Unity on its own did not have to worry about the operation so it just started and stopped the preview as it would normally.

The shorter load and unload times that generally occur with Unity make the engine a good choice for rapid previewing; we have already emphasized the need for a good preview tool so that users are able to validate their work. However, future iterations of PlayTIME and other TIME tools will need to find a way to use previewing features from different engines without slowing down their natural pace.

Activity Idle Time

After accounting for the time spent previewing, the next thing that was done to find a way to balance construction time was analysing the activity idle time: this is the time in which participants were not *visibly* doing anything productive. With PlayTIME this meant that no markers were visible in-frame. In Unity alone it means that the cursor had stopped moving entirely or was moving very slightly.

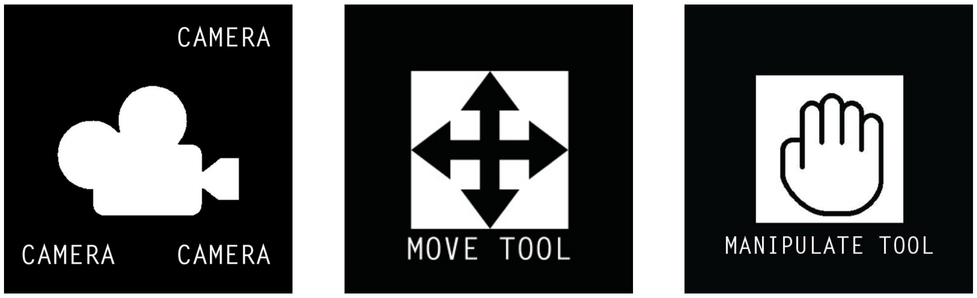
The conditions and the activity order both had significant effects on the time spent idling. For order, the reason is that for their second activity participants did not need to think about the task as much, discussed in a similar manner regarding the overall activity time and construction times.

The significance of the *conditions'* effect on idle time is essentially undeniable, with a KW p-value of 3.49e-5 overall for the surveys-only group and a p-value of 4.02e-6 for the full population. The surveys-only p-values were 0.0002122 for P-first and 0.06964 for U-first, indicating near-significance. The full population p-values were 1.62e-5 for P-first and 0.05852. The differences between the P-first and U-first significance can be explained using the same logic as the activity times, but strictly for condition, the reasons for the effect on idle time are different from the reasons for the order effect.

Although the participants were not *visibly* active during this time, this is not to say they were not doing anything at all. For both activities, some of the idle time was spent reading the task description and planning construction, or thinking. A lot of the idle time for PlayTIME was the time they spent searching through the markers and figuring out what they needed to do. This comes back to the unfamiliarity of the system: users were not quickly able to figure out what they needed because they did not recognize the paddles. Through observation we learned that three major components of the system contributed heavily to user confusion.

First, the AR paddles were not labelled on *either side*, instead relying purely on memorization of icons and their meanings and functions. Furthermore, users would also often leave paddles flipped over on the table, blank side up, until they needed to be used; this meant that the users who did this had neither the names nor the icons to help them recognize which paddle was assigned to which function. PlayTIME's learning curve and the unlabelled paddles clearly violate Nielsen's *recognition over recall* heuristic. Users frequently had to search for the marker they needed or look at the information sheet that was provided. The paddles themselves *must* be clearly labelled for future implementations using either a large, simple name (e.g. "PAN CAMERA") or explicit descriptions of the functions placed on the handles (e.g. "Use this to MOVE THE CAMERA").

Some of the *markers* were labelled, but this did not help on account of the second confusion: the markers were *upside-down* relative to the user. The bottom edge of the marker was towards the monitors while the top had the paddle handle. This was because during the development of the PlayTIME system, the camera was rotated this way and the markers followed so that they would not be upside down relative to what the AR system was detecting. This was an unjustifiable action; both the



(a) Camera pan (b) Object movement (c) AI manipulation

Figure 6.19: The AR paddle icons that were most commonly confused by participants.

camera and the markers should have been rotated 180 degrees and a simple rotation applied to the AR detection's output to fix the values within the editor.

Finally, some of the *icons* themselves represented on the markers were similar and therefore commonly confused. The markers that were frequently confused can be seen in Figure 6.19. There were too many unique confusions that occurred between these markers; here we overview what they were.

With Unity alone, using the middle mouse button to trigger a camera pan resulted in a *hand icon* taking the place of the cursor. When they wanted to pan the camera, users who were familiar with Unity ignored the fact that the camera paddle had a camera icon on it (Figure 6.19a) and they went for the icon they already knew: the manipulation paddle (Figure 6.19c). Furthermore, other users associated the object movement paddle with camera pan as well because it does give the impression that it is made to move something. The manipulation paddle and the movement paddle (Figure 6.19b) were also frequently confused. The marker confusions tell us there were issues with Nielsen's *matching between system and reality* heuristic: the PlayTIME icons did not match the Unity icons they were interfacing with. In essence, the AR markers did not faithfully represent the functions that they were made for.

Most of the activity idle time was wasted time, spent figuring out how to use the system instead of using it. The sheer amount of idle time all together violates a third heuristic: *flexibility and efficiency of use*; the system was slow and inefficient.

When asked about the systems' strengths and weaknesses in the post-study survey, participants reported the inefficiency in their feedback. Here are a few notable quotations about speed and efficiency:

> *"...If my goal was to be as efficient as possible, PlayTIME was substantially slower than the mouse, which I am more accustomed to."*
> - Participant 43-U

"Mouse: efficient... PlayTIME: takes longer, a little harder to give commands."
- Participant 38-U

"PlayTIME pros: ...being able to physically place objects. PlayTIME cons: Time consuming, not efficient (took time to get used to controls and time everything correctly). Mouse pros: Familiarity and efficiency; I have used a mouse my entire life, so the controls made sense."
- Participant 34-P

To summarize, we learned that future implementations of PlayTIME and other TIME systems can better satisfy Nielsen's heuristics by presenting more appropriate recognition of the tools. The tangible objects must clearly label what they do, and their icons, or other visual representation such as a small 3D toy or model, must be precisely representative of what it means within the system. Fixing these problems will speed up interactions and ultimately reduce time wasted, allowing users to produce their scenarios quickly and smoothly.

6.5.5 Level Construction & Usability

Table 6.5 shows the distribution of construction time for the surveys-only group, and Table 6.10 shows the times for the full population. These distributions are visualized in Figures 6.7b and 6.14b respectively. The observed distribution of usage for each feature can be found in Figure 6.8 for the surveys-only PlayTIME activity, Figure 6.9 for the Unity activity, and Figure 6.15 for the full population's PlayTIME activity.

For PlayTIME, the overall average construction time was 9:41 for the surveys-only group, and 8:28 for the full population. For Unity, the overall average construction time was 7:31. Since construction time scaled with activity time, the effects of condition and order and reasons are similar to those described above regarding the overall activity time. This is also indicated by the similar KW p-values: 0.02149 for all participants regarding condition, 0.00194 indicating high significance in the P-first group only, and no significance in the U-first group.

The construction time all together was the time spent completing specific tasks by using the systems' features. The performance for each task was evaluated in terms of the time spent using these features: the features were used correctly ("correct usage"); the users made errors while using the features, indicating struggle ("user error"); the users paused while using features ("feature idle"); or the system's limitations caused a problem that was not entirely the user's fault ("system error").

Error	Users	Count	Time	Average occur/user	Average time/occur
Nav. to wrong folder	6	7	7.5s	1	1.07s
Drag or place wrong asset	6	8	25.5s	1	3.19s
Try to add AI, miss target	2	4	9.5s	2	2.38s
Selected wrong object	4	13	16.5s	3	1.27s
Missed movement arrow	2	2	2s	1	1s
Move with wrong arrow	2	4	17s	2	4.25s
Try AI on selected spi.	10	15	61.5s	1	4.1s

Table 6.15: A small sample of errors that occurred, demonstrating the mouse's susceptibility to Fitts's law. The times shown here are approximated to half of a second.

One of the key differences between the activities, and the reason why "feature idle" was so common in PlayTIME is that with that system users can do multiple things simultaneously. In contrast, while using the mouse in Unity's editor only one thing can possibly be happening at any point in time. This is where the term "*feature idle*" comes from: with the mouse, users would have short pauses while completing a task, but with PlayTIME a user could pause by passively leaving a marker on the desk and actively complete another task using another marker. This was commonly seen while panning the camera; for example, a bunch of spiders were being placed in one area and the marker remained visible while the user panned to another area to *continue* placing spiders without removing the spider marker (exact occurrences of this not tracked).

Performance was analysed on a basis of *which features were visibly being used at any point in time.* That being said, PlayTIME markers that were "paused" but still visible heavily contributed to the feature idle statistics.

Here we discuss the breakdown of the construction time and look at the individual features used to complete the activity and how they differed between PlayTIME and Unity, with specific and important system usages, confusions or errors discussed in detail. We also discuss how the observed issues relate to Nielsen's usability heuristics and Fitts's law.

Fitts's Law

A small sample of errors that occurred, summarized by frequency in Table 6.15, demonstrate that the mouse was frequently susceptible to Fitts's law [49], which describes the time needed to reach a target as a function of the distance to and size of the target. Note that the times shown here do not include the time spent *correcting* the mistakes.

One common example was when users tried to navigate through folders and clicking on the wrong one, requiring the user to return to the parent folder or try again. This happened a total of 7 times over 6 users, costing an average of 1.07s per occurrence. A similar error occurred while placing objects: selecting or dragging the wrong prefab from the assets window. This happened 8 times over 6 users, averaging 3.19s per occurrence. Clicking on the wrong object while selecting was also an issue: this happened 13 times over 4 participants at 1.27s/occ.

The observance of Fitts's law with the individual features is discussed below.

Object Placement

The process of placing objects using PlayTIME AR paddles is easy: users just show the marker they want to place, where they want to place it, and press the C-button to place the object. It is even easier with the mouse: click on the object in the assets panel and drag it into the scene. The object placement categories had the highest ease of use ratings for the PlayTIME activity, with spiders having the highest ratings across all participants (M=6, SD=1.049). For the Unity activity, spiders also had the leading placement ratings (M=6.65, SD=0.572), but were second-highest overall after selection. Figures 6.8 and 6.9 show that the average time spent *correctly* placing objects with PlayTIME was 88% in PlayTIME and 96% with Unity alone. The times spent making mistakes while placing were 1.7% and 2.5% respectively. These numbers perfectly validate the high ease of use ratings.

This is interesting because it is consistent with the highly-rated importance of object placement in the demographics questionnaire (N=17, M= 4.176, SD=0.706; see Figure 5.11). Furthermore, object placement had the highest average time consumption across *all groups*, with an average of 3:46 for PlayTIME and 2:03 for Unity. Even in the context of the activity as a whole, we can see in Figure 6.7b that the overall time spent placing objects was around 34% for PlayTIME[2] and around 29% for Unity[3]. The differences were found significant by the KW p-values in the surveys-only group alone: 1.92e-5 overall, 0.0005041 within P-first and 0.02334 within U-first.

The fact that the most time was spent placing objects shows that participants focused the most on setting up the interactive objects within the world. This is important to the domain of prototyping because it directly relates to some of the questions we should consider early when building a game prototype: How is the en-

[2]PlayTIME feature percentages are *normalized* to show relativity to the other tasks only.
[3]Unity feature times are *not normalized* since only one thing happened at a time in Unity.

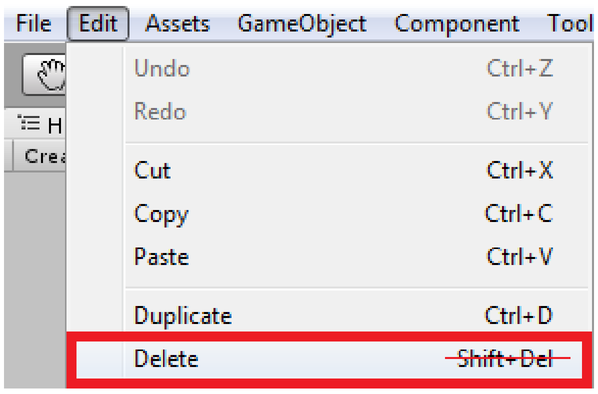


Figure 6.20: Since the keyboard was not allowed, users had to delete objects via the edit menu.

vironment set up? Where should objects go? How will this affect the environment? Since this was a spatial task it was nice to see that people were generally concerned about where things were located in space. Users spent the most time getting comfortable with their interactive objects, the enemies, and ensuring the game would be winnable.

An interesting note about placement: PlayTIME markers afford rotation while objects are being placed, but Unity does not. We restricted the ability to rotate objects so it would be one less variable to worry about. Two users tried to use Unity's rotation tool, which is entirely different from movement and placement, but they were told not to because PlayTIME did not allow rotation once objects were placed. The rotation should have been locked for PlayTIME as well; it did not appear to be a significant factor in the placement times but there were a few who did spend a few seconds getting the rotation correct. There are no definite numbers to reflect this; it is just an observation

Object Deletion

Furthermore, once objects were placed, users felt little to no need to remove them from the world. With PlayTIME, 15 users in the surveys-only group spent an overall average of 14.57 seconds deleting objects, and 29 users in the full population spent an overall average of 12.97 seconds deleting objects. In Unity the average was 13.31 seconds. Despite the numbers being close together, they were still significantly different: the overall KW p-value was 0.0009665 for surveys-only and 0.01228 for the full population. This is interesting because it shows that users were much more interested in *adding* to the world instead of taking away from it. The objects that *were* deleted were often the ones that were placed accidentally by double clicking the C button.

There were a total of 9 ways to commit this type of error, 3 for each object placement marker: the C button was occluded intentionally a second time while the marker was showing; the placement marker itself was occluded and then moved; or the C button was occluded accidentally while the marker was showing. Of the 40 samples analysed, 11 users visibly experienced at least one of these errors.

In Unity, deletion without the delete key consisted of a single option in the edit menu, pictured in Figure 6.20. They were not allowed to delete objects through the hierarchy since we ultimately decided it would be too hard to read for inexperienced users; plus, deletion through the edit menu is a standard in all kinds of software, which increases familiarity. Only 7 people used deletion in Unity; nobody in the P-first group committed any errors and the average error time in U-first was 3.29 seconds, or about 12% of their entire time deleting. Surprisingly, deletion received the second-lowest ease of use ratings, averaging 5.47 (SD=1.036).

In PlayTIME, users were often confused by the process of deletion. To delete an object, users had to first select the marker they wanted to delete using the selection wand, then show the marker representing that object (e.g. spider), and *then* occlude the B button to delete the object. This is not intuitive since with Unity a user only has to select the object and then delete it in the menu, or simply tap the delete key if the keyboard is being used. While trying to delete, the intuitive response was just to press the B button without showing the object marker they were trying to remove. This was counted as an error since it was not the correct way to delete, but users just did not think it had worked and often proceeded to try again, thus accumulating error time. After a few tries they were reminded that they needed to show the correct object marker. This mistake was made by 18 users. This is why the percentages for deletion error are so high for PlayTIME: 23% overall for surveys-only and about 35% for the full population. The complications here justify deletion having the lowest ease of use ratings for PlayTIME, averaging 4.687 (SD=1.261).

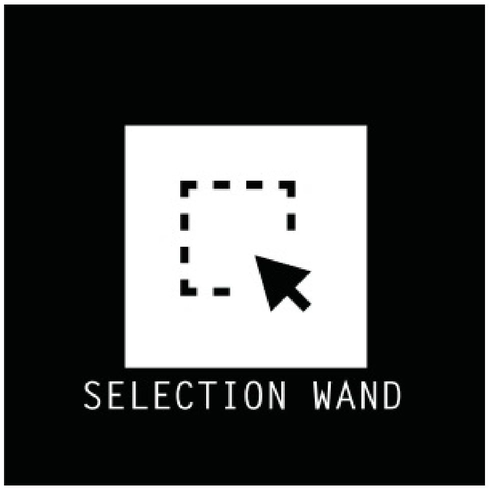
Selection

Selection was rated the highest overall for the Unity activity (M=6.75, SD=0.433). For PlayTIME, selection had relatively lower ratings (M=5.35, SD=1.492), with de-selection having slightly higher ratings (M=5.6, SD=1.497). Selection in Unity was done correctly 94% of the time. PlayTIME users had an average of 91% correct actions with selection.

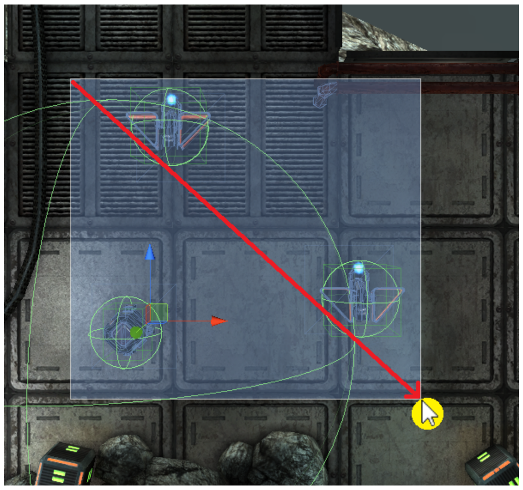
Selection with the mouse had the highest rating and time used correctly likely because is exactly the type of action we are *very* familiar with. We are most practised with this because it is the mouse's primary purpose: it is a means of interacting with a virtual pointer, the cursor, by using our hand. Whenever something needs doing, we simply push the mouse across the desk, which moves the cursor towards the desired location, and we click when are pointing at the right object. This is the staple of every component of every graphical interface: text boxes, radio buttons, web links, etc. all require a selection by pointing and clicking to trigger an interaction.

With PlayTIME, the 'pointer' equivalent is a red collision box within the editor, and its interface was the Selection Wand marker; unlike a cursor, the red box only becomes visible when the selection marker is active. Moving the tangible marker causes the virtual red box to move, just as the mouse does with the cursor. The difference between the interfaces occurs when the pointer collides with the virtual object to be selected: with the selection marker, selection occurs upon collision, whereas the mouse requires a click once the pointer is over the object. With PlayTIME, this 'click' gesture is automatic; the user is *not* required to do anything once the marker aligns with the object. This slight difference was the source of much confusion for participants who associated the act of clicking to select with using the C marker to select: 7 out of 40 people did this an average of 2 times each. Similarly, 15 of 40 participants tried to confirm manipulation an average of 2 times each, and 7 tried to confirm movement an average of 2 times each, where neither of these confirmations were required.

PlayTIME tried to optimize things like selection by not requiring that 'click' of the C button, but in the process made things confusing for users and wasted time. Users spent an overall average of 2:19 waving PlayTIME's selection wand and only 47 seconds to simply click on things in Unity. This is justified by the highly significant effect that the condition had on selection time on *all groups*: looking at Group 2 alone in Table 6.7, the KW test found a p-value of less than *one hundred-thousandth* for the whole population, less than one thousandth for P-first and less than one hundredth for U-first. For the full population in Table 6.7 we see that the KW test returned even smaller p-values. In both cases, even the selection features' normalized *percentage of construction time* were heavily affected by condition. Due to the shockingly low numbers, these tests were double-checked and returned the same results. This is a clear indication that the use of the selection features was most certainly different between the conditions. Regarding the effect of condition, selection had the lowest set of p-values thereby making it the most significantly-different feature. Order did not have any explicitly significant effects on selection, but the same explanation regarding



(a) Selection marker



(b) Multiple selection in Unity, which was not allowed

Figure 6.21: The paddle icon for selection was based on the multi-select feature in Unity, which participants were told not to use.

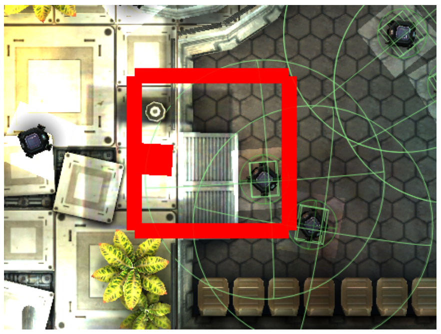
the differences between the first and second activity shows that there was an implicit effect.

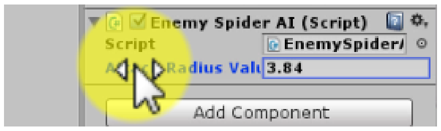
The difference in *how* the selection methods work is also the reason why PlayTIME's C button was categorized as an "other" feature along with mouse clicks; they technically have the same meaning but are *used* differently in the activities and therefore cannot be compared fairly. As seen in Figure 6.11, the overall average distribution of click actions was 91.142% correct clicks, 4.223% incorrect clicks, and 4.635% extra clicks, out of an average of 177 total click actions. A single "click action" is defined here as any type of click: left-click, double-click, middle-click or right-click.

Multiple Selections

There was a problem with how users were permitted to use Unity's selection feature. With PlayTIME, the marker behaved as a wand, so that "waving" the virtual pointer over an object would select it without having to click. Since users were required to click *on* an object in the Unity activity, it means the equivalent of the "waving" gesture was unavailable: multiple selection, pictured in Figure 6.21b, was triggered by clicking away from the object and moving cursor diagonally, creating a rectangular selection within which any object would be selected. This is exactly what the selection AR paddle's icon depicts (Figure 6.21a).

This goes back to the consistency heuristic: the icon on the marker described that it was used to simulate the click-and-drag action, yet when it came to selecting


(a) A close-up of the manipulation indicator in PlayTIME.



(b) A close-up of the manipulation indicator in Unity.

Figure 6.22: In this view, the manipulation marker is used to change the attack radius of multiple spiders.

with the mouse they were only allowed to click. This was a poor decision for the study design. Users reported that batch actions, such as applying AI to multiple enemies, were more helpful in PlayTIME, however this is a biased result since Unity was restricted.

Adding & Manipulating AI Behaviour

For selection and AI attachment, the targets were the objects in the scene, so the ability to select and apply AI correctly was hindered by the size of the object. Unfortunately this could not be fixed since all participants were required to keep the camera at a fixed distance, meaning the objects would always appear to be the same size.

A frustrating issue was that in Unity the AI behaviour could not be applied to a spider that was already selected; this is a silly caveat with Unity, but it was still counted as a user error since they were told at the start that this would happen and that they needed to deselect before applying AI. This occurred a total of 15 times over 10 participants, costing an average of 4.1s/occ. It resulted in users trying to repeat the action, thinking they had missed the target spider when they had not. *Actually* having missed the target spider while placing AI only occurred 4 times over 2 users, averaging 2.38s/occ.

Manipulation received the second-lowest ease of use ratings for PlayTIME: the overall average was 4.95 (SD=1.396). The Unity ratings were a fair bit higher, averaging 5.8 (SD=0.98). This is interesting because the feature was built to reflect the way it is done in Unity, which is a slider. In Unity, participants clicked on the attack radius property, pictured close-up in Figure 6.22a, and moved the cursor left and right to change the value. Similarly with PlayTIME, when the manipulation marker appeared, a box was displayed in the centre of the display, seen in Figure 6.22b, and in the editor in Figure E.7. Moving the marker left and right within this box changed the attack radius. Furthermore, this could be done with multiple spiders selected, whereas Unity objects could only be manipulated one at a time.

Panning the Camera

Panning the camera was treated differently between the conditions. In Unity, middle clicking and dragging would offset the camera along the axis of the cursor's movement. This only worked for a short distance and so the action had to be repeated some 5 or so times to get the camera from place to place.

Since PlayTIME did not have a notion of click-and-drag, we treated the camera marker like a joystick. When the camera marker was visible, the centre of the screen was the default position; moving the marker in some direction away from the centre would cause the camera to pan in that direction, just as a controller joystick has an effect on movement in a similar fashion.

For PlayTIME, panning had the fourth-lowest ease of use average rating: 5.25 (SD=1.639). Panning had the lowest ratings for Unity: 5.35 (SD=1.458). There was a fair amount of feedback regarding panning for both systems. A couple of users said that they enjoyed PlayTIME's method of panning:

> *"PlayTIME pros: Camera pan (much faster and smoother)... Mouse cons: Poor camera pan- it took forever to get places, and I don't enjoy that."* - Participant 34-P

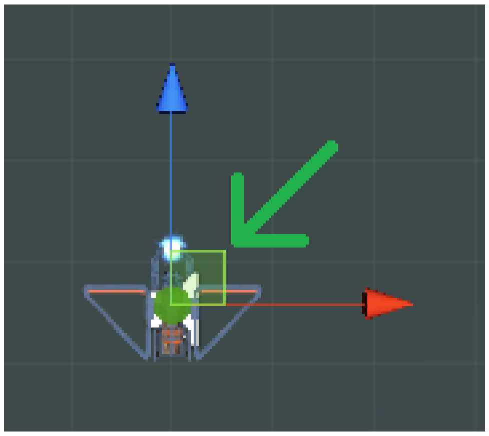
> *"PlayTIME was very fun to use as it made it much easier to move the camera around quickly."* - Participant 37-U

There were some who explicitly did not enjoy it:

> *"I find panning in PlayTime to be rather difficult, and it takes a while to get used to. Moreover, it requires that you readjust when you are out of the field of view of the [AR detection] camera, which I was not paying much attention to."* - Participant 33-P

Participant 49-U just used point-form, implying that PlayTIME's camera movement was too sensitive.

PlayTIME's approach to camera panning was interesting to try, but it would not be practical in 3D situations; this activity was constrained to two dimensions. This will be very important when developing future systems, such as *FilmTIME*, where


(a) Unity's move tool from a top-down view. The green arrow is pointing at the 'plane' tool, an alternative way to move objects.

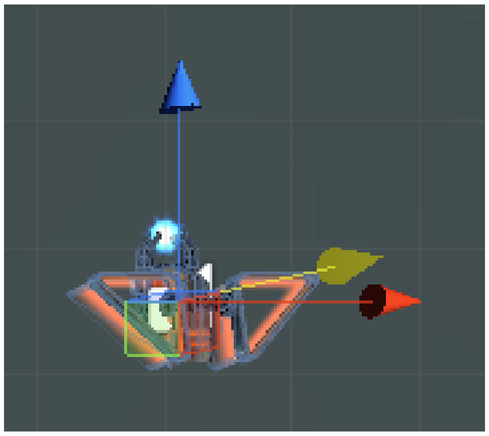

(b) In this example, the depth-axis movement arrow was visible due to the perspective; it was unintentionally selected instead of the red arrow.

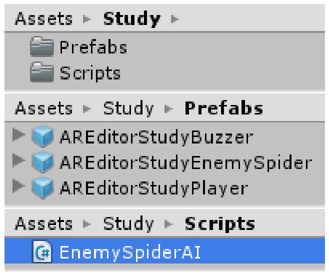
Figure 6.23: Use-cases with Unity's move tool. Participants were restricted to the red arrow for horizontal movement and the blue arrow for vertical movement.

tangibles will play a part in 3D film and animation production and cinematography techniques. For this application it would be more practical to have a panning tool that can be precisely tracked in 3D, such as a physical camera with motion capture nodes attached to it.

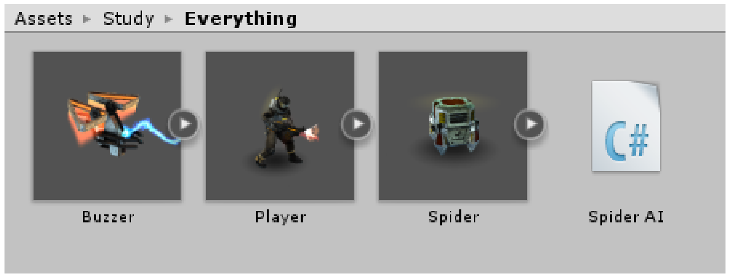
Moving Objects

Unity's move tool was used correctly by clicking on one of two tiny arrows (red for horizontal or blue for vertical movement). A the third movement arrow along the depth axis (yellow when clicked) was still visible and often got in the way, as seen in Figure 6.23b. The depth axis becomes visible when an object is closer to the edge of the work area; since the scene was viewed in perspective, it the incorrect arrow would appear in front of the intended one and cause an incorrect click. The depth axis movement was accidentally used 4 times by 2 participants and cost a total of 4.25s/occ. Objects were moved into "the abyss," requiring a correction. Also, 2 users completely missed the movement arrows, clicking on something entirely different. An alternative movement was attempted by 3 users: they tried to use another method of movement which provided a larger target to activate movement seen in Figure 6.23a. The move tool in Unity demonstrates Fitts's law since the targets the users were instructed to use for movement were very small.

Movement in Unity differs from PlayTIME's move tool. In Unity a user knew exactly which object would be moved. With PlayTIME, however, users often left



(a) The asset folders presented to users during the Unity activity. The blue ribbon shows the height of the target for selection.



(b) The way it should have been, with everything in one folder and large, clear icons and a simple label instead of just one long, ambiguous line of text.

Figure 6.24: A comparison between the asset layout during the Unity activity and what could have been done to prevent errors.

objects selected in off-screen areas before using the move tool, which would result in additional, unintended objects to be moved. Sometimes those objects were pushed through the floor or inside a wall, where they would never be accessible. Unfortunately the exact occurrences of this flaw could not be recorded since it was an off-screen error; users only noticed when previewing and finding they did not have enough enemies for the level to be winnable.

Movement had the third-lowest ease of use rating for Unity, averaging 5.75 (SD= 1.135), which was still higher than the average rating for PlayTIME, 5.412 (SD= 1.457). Of all the participants, 18 used movement in Unity and used it for an average of 46.7s, where only 11 from that group used movement in PlayTIME for an average of 43.1s, and 21 users overall for an average of 33.5s.

Folder Navigation (Unity)

The Unity activity's "other" feature was navigation through the asset folders in Unity, which was more difficult than it needed to be, despite the high ease of use ratings (M=6, SD=1.095). The time distribution for the navigation feature in Figure 6.12 shows that the average time spent navigating correctly was 94.556%, and 5.444% spent navigating incorrectly.

The only reason why the navigation feature was used is because the placeable objects had been previously sorted into their own folder and the AI behaviour had its own folder. Figure 6.24a shows the way assets were presented to users in Unity, divided into three folders that had to be navigated. To select an asset to be placed, the target was a narrow ribbon only the height of the asset's name. Participants had to first navigate through the parent folder, "Study," to reach assets stored in "Prefabs" and "Scripts." Figure 6.24b shows an alternate way that assets *should* be organized for this activity, using large, clearly identifiable icons.

Having everything in one place would be similar to how PlayTIME stores its "assets:" the paddles representing the placeable objects and the AI behaviour *were* located in one place: the desk. The user just had to pick the paddle they needed and put it in the camera's view to make it work, whereas with Unity alone the placement objects had to be navigated to before they could be placed. The alternative for Unity's asset organization described above would present users with large, clear icons, labelled with the simple yet descriptive name of the object, instead of a long, confusing name with a blue square. Not only do the large icons prevent target issues with Fitts's law, but they also agree with more of Nielsen's heuristics: using the same icons for both systems would mean the users had *consistency and standards* they could adhere to. Although the current AR markers show similar graphics, they would need to be exactly the same for consistency. Having the same icons for both systems would support *recognition* since they would be used to seeing the same icons repeatedly. Furthermore, it supports the *visibility of system status* heuristic because it would prevent users from having to focus on the *names* of the assets, as in Figure 6.24a, when they could just *see a picture* of what it is they want to use and immediately know it will work. All of these together give us *easier prevention of errors.*

6.5.6 Creativity Support & Preference

CSI Discussion

The Creativity Support Index (CSI) questionnaire was designed to describe how well a system supports one or more of six creativity-related metrics: collaboration, enjoyment, exploration, expressiveness, immersion and results worth effort. A higher score indicates that at least one area of creativity is supported very well.

The average overall CSI score for the PlayTIME activity is 70.125 (SD= 17.819), and for the Unity activity the average is 74.925 (SD=14.471). CSI scores map nicely to letter-grades [8]; in this respect, PlayTIME earned a B- whereas Unity earned a B. None of the statistical tests found any significant effects for these results (Table 6.3); the responses were generally balanced, but it is notable that PlayTIME was deemed to be less creative than Unity on its own. This may be because several of the users already had experience with Unity, again bringing us back to familiarity of not only the mouse but also the software.

One CSI question pertaining to enjoyment, "I enjoyed using the system," received the highest rating across all participants (M=5.85, SD=1.014), the highest rating in the P-first group (M=6.1, SD=0.7), and very nearly the highest rating in the U-first group (M=5.6, SD=1.2). After the ratings, the second part of the CSI questionnaire was a direct preference question, comparing each of the 6 CSI metrics with each other. The ratings are weighted by the number of times each of these is picked, meaning a metric that was picked fewer times will reduce the impact of the ratings pertaining to that metric. The average number of times *enjoyment* was picked over the other CSI metrics was 3 for PlayTIME and 2 for Unity. In the feedback questions, 14 participants explicitly stated they enjoyed using PlayTIME over Unity alone. There were 4 people who enjoyed both systems without preference. Of all the participants who enjoyed PlayTIME, 7 cited the main reason as the novelty of the experience and using a new system with an unfamiliar technology (AR).

The high prevalence of enjoyment indicates that PlayTIME was indeed a fun and new experience for people, agreeing with their emotional responses from the PANAS questionnaires. The other metrics that had high ratings and preferences were exploration (M=3) and results worth effort (M=3). Together, these imply that participants also enjoyed discovering what the features were capable of helping them *create*, and in the end felt that the tool helped them get what they wanted out of the activity.

Despite the feedback, the preference question in the post-study questionnaire shows that people leaned towards the mouse. Just because the users thought the mouse was more appropriate for certain features, this should not hinder the value of the experience they had while using the PlayTIME system *as a whole*, which is the point of the CSI questionnaire and feedback. Therefore it should be noted that the preference ratings are reflective of the *individual features of each system* and this should not take away from the meaning of the CSI results and feedback.

Participant 45-P summarized the experience and creativity very nicely:

> *"It was far more interesting to be using technologies that I'd never used before, whereas I am almost always using just a mouse and keyboard, or as in this case, a mouse alone. The mouse, however, is something that everyone already knows how to use, but lacks the immersion and level of creativity. In an industry that relies on creativity, I feel that the PlayTIME tiles would definitely make for a great experience in the game development, animation, game programming, etc. industry. I am a strong believer in being able to do work effectively and have fun doing it; this is proof as far*

as I'm concerned... It took me probably twice as long to complete the task with tiles just because I was having fun doing it."
- Participant 45-P

Since we are also interested in fostering collaboration with PlayTIME, we should be concerned with the collaboration metric of the CSI. On average, PlayTIME scored slightly lower than Unity on one collaboration question and slightly higher on the other, with the exact same preference counts. On the first question, "The system would allow other people to work with me easily," PlayTIME averaged 4.7 (SD=1.873) whereas Unity averaged 4.8 (SD=1.249). On the second question, "It would be really easy to share ideas and designs with other people using this system," PlayTIME scored 5.2 (SD=1.503) whereas Unity scored an average of 5 (SD=1.549). These mixed results show that users did not believe there was a difference between the collaborative potential of PlayTIME and Unity.

This is surprising because Unity, by nature, follows traditional computer-based design and development techniques; it *can only have one operator* per-computer at any given point in time, and resources are not instantly shared. On the other hand, PlayTIME has a collection of objects that multiple people could *physically* use simultaneously. One possible reason for this lack of collaborative interest by participants is that they *were not collaborating* during the experiment; perhaps if the task had been done in pairs, they would have realized the benefits of collaboration and PlayTIME may have received higher ratings for this metric.

Scenes & Task Deviation

Interestingly, the scenes themselves did not tell us much about creativity as anticipated. The Manhattan scores for each activity were similar enough that no statistical significance was found. As well, many of the scenes stuck to the *exact* task description with *zero* deviations: 10 out of 50 for PlayTIME and 14 out of 50 for Unity. Table 6.13 shows the complete distribution of objects for all scenes. The most frequent deviation from the instructions was placing extra spiders in reachable areas that were not labelled on the map. Simply placing an extra enemy anywhere would only earn one point since there was no removal involved. The placement of extra enemies was supported by the second-most frequent action: removing spiders from the main room of the level (and less frequently other areas) to have them re-placed in other areas. An combined action such as this would earn two points in the Manhattan score: one for taking an enemy out of a specified area, and another for placing an extra one in an unspecified area.

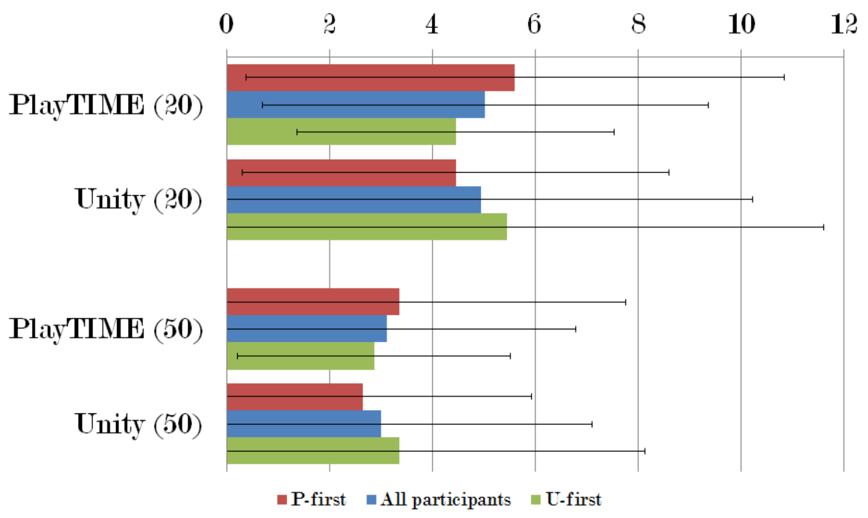


Figure 6.25: The average Manhattan scores for both activities, for each participant group, for both the surveys-only population (surv.) and the full population (all).

Aside from the zones or areas that were given to the participants as guidelines, the level was filled with specific locations to place their objects. The task description provided at the start was merely a guideline to make the level *winnable*, meaning they would have enough enemies to complete the objective that we had built into the game project. To name a few examples: the patio, near where the game started up, had many crates behind which enemies could be hiding. In the main room of the level, spiders blended in well with the plants that were placed there already. On the balcony, the last area visited by the player, spiders could hide behind barrels. There were many different outcomes (100 to be exact), so there would be too much to describe here.

It is also notable that the scene analysis for a population size of 20, Group 2 alone, against the full population size of 50, yielded significantly different p-values (KW p=0.03263, MWW p=0.03505; see Table E.2).

Figure 6.25 shows us that the overall Manhattan score averages were quite different: the average scores for Group 2 alone were 5.025 for PlayTIME and 4.95 for Unity, and the averages across all 50 participants were much lower, 3.11 for PlayTIME and 3 for Unity. This tells us that, despite the population of 50 *including* the sub-population of 20, adding the extra 30 Manhattan scores to the mix had a very noticeable effect on the results.

The p-values for both populations showed that neither condition nor order had a significant effect on the results, but by comparing the values in Tables 6.9 and 6.14

we can see that the effects were simply *greater* and therefore *more significant* in the full population of Groups 1 and 2.

Regardless of population size, as discussed earlier, most people had shorter run times during their second activity because they had already done the activity. Looking at Figure 6.25 we see that the higher averages occurred in whichever activity was done first. It is still interesting that the statistical tests did not find significance in the activity order because of this. The reason is likely due to the high standard deviations, indicating that, despite the averages having a clear difference, there was a high overlap in the raw data. It is possible for the second activity, participants had tried to complete it as fast as possible, strictly adhering to the task they were given and therefore reducing their Manhattan score.

6.6 Summary

6.6.1 Usability

Statistical analysis of the CSUQ results for each condition revealed that using PlayTIME over Unity *did* have a statistically significant effect, however this effect was not in PlayTIME's favour, thereby rejecting our first pair of hypotheses pertaining to usability. Investigating the screen capture videos for performance, we found that users generally took *extremely significantly* longer to accomplish tasks.

Since we are developing PlayTIME as a starting point for other tangible-based scenario development tools, it is important to acknowledge the limitations of the current implementation. Some issues arising from the technology used, augmented reality with fiduciary markers, also caused errors and difficulties for the users. PlayTIME also presented clear violations of important usability guidelines, Nielsen's ten usability heuristics that have much room for improvement. Most notably, it *must* be better at preventing user errors.

Regarding ease of use and preference, the statistics and participants' feedback generally agreed that the mouse was easier to use simply because it is more familiar. This is not just within the context of Unity or scenario design; it is applicable to all computer and software-related disciplines and it is a fact we see daily: *nothing is simpler than the mouse.*

6.6.2 Creativity

We also had some insight into PlayTIME's creative potential. The statistical tests did not find any significant effects by condition on CSI scores. Similarly, the Manhattan scores used to quantify deviations from the assigned task also yielded no significant differences. Therefore we must also reject our second pair of hypotheses pertaining to creativity.

We learned that users did not believe PlayTIME was more collaborative than Unity, despite its affordances to stray away from traditional design methods. We also learned that there were more bugs in PlayTIME which explicitly prevented users from making creative choices; hindering creativity is another reason why fixing usability issues is the priority.

6.6.3 Enjoyment & Fun

The PANAS results yielded statistically significant results. Both the condition *and* the activity order had effects on the participants' positive affect scores, showing that their positive emotions consistently improved through the PlayTIME activity. From the CSI we learned that PlayTIME also received higher scores on one of the two enjoyability metrics. Furthermore, of the 20 participants who provided feedback, 14 explicitly stated that they enjoyed PlayTIME over Unity because they felt it was a novel experience. Another 4 participants were indifferent. There were also participants who stated that PlayTIME felt more like play than work, indicating some potential to achieve the goal playful design.

This evidence gives us just cause to validate our hypotheses pertaining to enjoyability. PlayTIME has succeeded in providing people with a guide for building a scenario development tool that makes digital prototyping an *enjoyable, fun experience* that supported the exploration of ideas and the system itself.

6.6.4 Conclusion

By running this study, we saw how the PlayTIME system, currently an AR-based extension for Unity's editor, compared against Unity on its own. A variety of questionnaires presented us with qualitative data about the systems' usability, creative potential, as well as enjoyability and emotional response. We discovered that the systems themselves had major performance differences, significantly different emotional responses, and similar creative potential.

Chapter 7 ties the results of this study into the original expectations, and presents considerations for future development of PlayTIME and related systems.

Chapter 7

Conclusions

7.1 Hypotheses

Here we revisit the hypotheses introduced in Chapter 1. Each hypothesis is *accepted* or *rejected* based on the results presented in sections 6.3 and 6.4, and the factors discussed in section 6.5.

Recall that a "*statistically significant*" effect is defined as having the Kruskal-Wallis, Wilcoxon signed-rank or Mann-Whitney-Wilcoxon tests return a p-value less than 0.05. Here, p-values less than 0.01 are referred to as "*highly significant*," and p-values less than 0.001 are referred to as "*extremely significant.*"

1. **Usability:**

   Hypothesis: The use of PlayTIME will have a significant effect on the *performance* of the developers completing the assigned task.

   This hypothesis must be *accepted* due to the following evidence:

   (a) The use of PlayTIME (changing the activity condition or system) had a *statistically significant* effect on the qualitative CSUQ results specifically pertaining to the *system usability* factor (Table 6.2).

   (b) The use of PlayTIME had a *statistically significant* effect on the ease of use questionnaires comparing the *tangible interface* used with each system (Table 6.4).

   (c) The use of PlayTIME had *statistically significant* effects on some areas of overall activity performance and feature-wise performance, *highly significant* effects on several other areas and *extremely significant* effects on other areas, some of which are irrevocable due to p-values of approximately zero (tables 6.6, 6.7, 6.11 and 6.12).

(d) The performance values and figures throughout sections 6.3.4 and 6.4.1 show clear differences between the conditions.

Hypothesis: This effect will be in PlayTIME's favour.

This hypothesis must be *rejected* on the grounds that the overall CSUQ scores were lower, the ease of use scores were lower, and performance was overall slower for PlayTIME with more errors.

2. **Creativity:**
   Hypothesis: The use of PlayTIME will have a significant effect on users' *creative output.*

   This hypothesis must be *rejected* due to the following evidence:

   (a) No statistically significant effects found on the CSI results between the conditions (Table 6.3).

   (b) No statistically significant effects found on the scenes, which were the output of each activity, between the conditions (tables 6.9 and 6.14).

   Hypothesis: This effect will be in PlayTIME's favour.

   This hypothesis must be *rejected* on the grounds that it is dependent on the first hypothesis being accepted.

3. **Enjoyment and fun:**
   Hypothesis: The use of PlayTIME will have a significant effect on users' *emotions* and will positively affect users' *enjoyment* of the activity.

   This hypothesis must be *accepted* due to the following evidence:

   (a) The use of PlayTIME had a *statistically significant* effect on the *positive affect through condition* metric from the PANAS results (Table 6.1).

   (b) The order of the activities had a *statistically significant* effect on the same metric, as visualized in Figure 6.2.

   (c) Of the CSI metrics, enjoyment received the highest ratings for PlayTIME.

   Hypothesis: This effect will be in PlayTIME's favour.

   This hypothesis must be *accepted* on the grounds that the impact the system had on emotions was clearly in PlayTIME's favour, and the feedback regarding enjoyment was generally positive.

7.1.1 Discussion of Hypotheses

PlayTIME most certainly had an effect on the usability and performance. Unfortunately this effect was not in PlayTIME's favour. The CSUQ scores were lower overall, most noticeably on the system usability metric. The post-study ease of use questionnaire also yielded lower results for PlayTIME. Furthermore, almost everything was more time consuming with PlayTIME and it yielded more errors overall.

What we learned from the usability ratings and the performance statistics can help future renditions of PlayTIME and other technologies. Most importantly, Nielsen's heuristics, which are model metrics in HCI, must be upheld; of all of them the biggest violation was on error prevention.

The rejection of the second hypothesis is not at all bad news. It does not necessarily mean that PlayTIME was to blame for hindering creativity. As discussed in section 6.5.6, it was more likely an issue of the activity design. The task description was too specific and had participants believe they they were strict. A more open study design would be more appropriate in a future study to foster creativity.

Even if the creativity effect hypothesis had been accepted, the next would have been rejected anyway since it was hard to differentiate which system produced higher scores and by how much. The averages for both CSI and scene analyses were similar for both systems.

With respect to the final hypotheses, it was very interesting to see that PlayTIME had such a great effect on the positive emotions of participants. On average PlayTIME always increased positive affect scores, while Unity alone only had this effect on those who used Unity first. In the feedback, 14 of 20 participants preferred using PlayTIME over Unity alone, and 4 others had no preference, meaning they did not *dislike* it. Several users cited that it was a novel and playful experience. This means that the hypothesis can be confidently accepted in PlayTIME's favour.

7.2 Limitations & Future Work

The current version of PlayTIME proved to be a worthwhile creation. Looking at what participants did with the system they were given, they happily managed to *accomplish their goals*. Albeit the goals were given to them, they still completed a valid task using a novel tangible interface to operate the system.

7.2.1 Augmented Reality

One of the caveats of using fifteen-year-old software is that it is fifteen years old. ARToolKit was developed in 1999 and has scarcely been maintained. Fiduciary marker technology is definitely well-known, and it paved the way for other tracking technologies used for AR, but to use a primitive framework was probably not the wisest choice.

Primarily, ARToolKit's fiduciary marker detection is extremely sensitive. Different lighting can change the response of the system, so it was a good call to add a quick way to interface with ARToolKit's threshold value directly through Unity's editor; this allowed us to adjust the sensitivity to light when needed. Another sensitivity issue: during the evaluation study, so much as a fingertip would throw off detection. As participants became frustrated when PlayTIME did not show the marker they were trying to place, they were reminded countless times that their finger was in the way. This was not only annoying for the participants, but it was also tiresome to watch it happen frequently. This should not have been an issue from the start; a more reliable technology is needed.

My recommendation for future TIME products is to *not use primitive AR technologies*, namely ARToolKit. Other AR software development kits, including Qualcomm's *Vuforia* [109] and *Metaio* [110], exhibit similar functionalities but are more stable and kept up-to-date. Other tracking can be used for spatial awareness, such as PTAM, with entirely different methods of tracking specific objects. Motion capture could be used to track the positions and orientations of tangible objects, which could be made of basically anything the designer should choose. QR codes could then be used to carry the actual data of the game world between stations.A potential issue with this is that motion capture requires a fixed designated space, so this must be accounted for when developing work stations: the tracking stays where it is. An alternative for tabletop-based activities would be RFID tags, which rely on direct electronic signals. One small drawback is that RFID tags require a contact surface.

7.2.2 Studies & User Evaluations

Here are some ways that future implementations learn from the current PlayTIME for better evaluations:

1. The study should not have required each participant to do the activity twice, once for each condition. In other words, the study should follow a ***between-subjects*** design. It is clear that the first activity had a direct impact on the second, since all participants reported that directly and a lot of statistical tests found significance in the order. More importantly, this would *minimize the amount of data.* The study described in Chapter 5 obviously presented far too much data, given that 50 people each ran two activities. A between-subjects design would reduce the amount of data

by over 50% and the required analysis by upwards of 80%. 50% is immediately reduced since each participant only runs one activity. This also means only having *one* set of comparisons instead of *five*: All PlayTIME vs. All Other system, as opposed to that for all participants and each group, *and* comparisons to find the effect of order. Furthermore, there would be no need for a post-study questionnaire because participants would not have any comparison.

2. To better the effect of a between-subjects design, all participants should have ***exceptionally similar expertise and background experience*** to reduce the discrepancies in confidence that may influence the results. It would also balance the results for each condition without having to worry about order to do that job. Participants should preferably have *no experience* as professionals in scenario-driven industries, such as film, animation or games. Younger people would be better for this since they would presumably have no professional experience, and they are very imaginative and proud of it.

3. A tool that helps with performance analysis is incredibly important. It has not been reported at all yet, but there is an analysis tool integrated with the current PlayTIME. It tracked the usages of the main features: placement, deletion, selection, etc. Despite how much time was put into the tool and how much it could have helped, there two big drawbacks: first, it was not *automatic*. When the PlayTIME plugin was initialized, the metrics tool did not initialize with the rest of the system; it had to be started separately. For a tool to be truly useful it should track equivalent actions for *both* of the systems being compared; the current implementation only tracked PlayTIME actions. Such a tool must be built into the development pipeline so that it starts up automatically.

Furthermore, the metrics tool was only looking out for a few specific events. There is no way it could have directly identified all of the 130 PlayTIME events and the 80 Unity events that were observed. A metrics tool for both systems would help if it can at least identify a handful of specific events. Knowing *what* happened is decent, but having a system understand *why* these events happen and what they *mean* would be incredible.

A metrics tool must be *automatically integrated* and have *some knowledge* about the possible events that can occur and their meanings.

4. To *reduce* the number of possible events, largely populated by errors, it would help to actually prevent the errors from happening in the first place, as per the most-violated usability heuristic. Most of the system-related errors were because of the AR, so changing the technology would be a start. From the user side, using

familiar and consistent icons and objects to represent what the tangibles' functions are would help reduce the memory load for users, thereby enabling them to make correct decisions and do it faster. The complete set of events, correct and incorrect, can be found in Appendix D.3.

5. Pilot sessions are important and allow system errors to be identified and removed before gathering participants for the data population. I was worried about losing potentially good data, so I only had two pilot sessions, neither of which helped us identify potential issues. Especially if the target demographic is broadened to people who have little to no experience in the field, or a more general audience. Future studies should run as many pilots as are needed to eliminate frequently-repeatable errors.

6. Instead of having only screen capture to record performance, it would also help to include a video recording of the actual participants to see what they are doing during the idle times. In his study, activity idle time gave the appearance that users were not doing anything at all, however they may have been observed struggling to locate the correct PlayTIME marker or taking notes. Furthermore, most of the participants were silent while completing the activities, some only asking for help upon repeatedly committing the same mistake. Audio recording should be analysed and users should be encouraged to talk through their design processes; this could also help identify areas of struggle if users are consistently vocal about what they are trying to accomplish. The additional video and audio capture would provide more evidence for or against the usability of the system.

7. Finally, the activities planned for studies like this one should do a better job fostering creativity. We have seen that the CSI questionnaire here, which is an absolutely brilliant way to measure the contributing factors of creativity, did not yield any significance. Furthermore, the scenes mostly matched the task description. The participants likely felt they had to stick to "the rules" that were never really official and that hindered creativity.

A good starting point for a *completely* creative task is the one found in Chapter 3, the TimeSplitters study. Contrary to the final study, the TimeSplitters study activity was left very open. The resulting levels were so vastly different there would be no way to possibly assign them a score, such as the Manhattan distance method. The TimeSplitters study had barely any constraints, only a few on the items that they were and were not allowed to use, which was still a large number. There needs to be a way to *balance the constraints with creative freedom.*

7.2.3 Future Evaluations

The lessons learned provide opportunity for future user evaluations of PlayTIME to explore new directions in tangible prototyping. Here, I present three study concepts to further investigate the effects of familiarity, study the factors that enhance enjoyment of using the interface, and explore how different demographics complete tasks with PlayTIME.

Familiarity Study

For the study presented in this thesis, none of the participants had previously used PlayTIME, but some had experience with a game engine. The average activity duration with PlayTIME was approximately 17 minutes for the participants who used it first; they had experience with neither the system nor with the task they were given to complete. With an expert who has had extensive experience and familiarity with PlayTIME's interface, a control run of the study was completed in approximately 4 minutes.

Hypothesis: I hypothesise that familiarity would have a significant impact on performance. There is a noticeable difference between average activity time by users with no experience with PlayTIME, and the completion time by a user with experience and frequent use of the system.

Demographics: Participant groups may include people who have never used PlayTIME, people who have been using it for a day, a week, and a month.

Metrics: The most applicable metric would be performance to measure how the system is used by people with varying degrees of experience.

Method: A *between-groups* study design would investigate the performance differences between sets of participants who have been practising and gaining experience with PlayTIME over different windows of time, such as a week or a month. Participants in each group could be assigned appropriate tasks to complete to become accustomed to the system over time. A *within-subjects* study design with repeated measures would be able to follow particular subjects as they learn and become more familiar with the interface. This design would require participants to return for multiple sessions after the initial meeting to repeat the experiment and collect new data each time.

Data to be collected would include screen and video capture for performance analysis, and an interview or questionnaire asking about how often they practised using the system over time, and the depth of their activities. Performance analysis can also be used to compare the correct and incorrect usages of the system within

Potential Issues: A foreseeable difficulty with this study would be user retention. Motivation may also be a concern when it comes to having the participants practise to get where they need to be for each session. These concerns apply to both the between-groups and within-subjects designs since each requires some or all participants to continue using and becoming familiar with PlayTIME.

Biometrics Study

To further evaluate the enjoyment of PlayTIME, future studies may include biometric measures to collect data on physiological factors supporting enjoyment of the system. Direct physiological measurements of emotion to supplement self-reported measurements such as the PANAS questionnaire should provide a detailed overview of the true emotional and enjoyment responses from participants.

Hypothesis: I hypothesise that evaluation with biometrics will show that PlayTIME is more enjoyable than Unity alone, and will also help identify the strengths and weaknesses of each system.

Demographics: For simplicity, the study could be limited to game designers of intermediate experience.

Metrics: The metric evaluated is emotional response to PlayTIME, specifically focusing on the level of enjoyment and excitement.

Method: Following a task-based activity, topical electromyography (EMG) nodes attached to a participant's face can be used to measure the tenseness of facial muscles to determine whether participants are smiling, laughing, frowning or scowling during their experiences; facial expressions are used to identify the emotions experienced by participants. Galvanic skin response (GSR) is a measurement of skin conductance, used to determine excitement or frustration. Biometrics would provide some insight towards the emotions experienced by game designers using PlayTIME, and serve as quantitative evidence backing up qualitative questionnaires such as the PANAS. Screen capture could be used to align the emotional responses with specific events during the session. This study could be conducted using the current implementation of PlayTIME.

Demographics Study

A third study would focus strictly on the use of PlayTIME and evaluate how different demographics use the system. For the study described in this thesis, the target demographic included people within creative domains (games, film, animation) who were new to game development or had limited experience. A future study evaluating performance and enjoyment using PlayTIME could compare how it is used by two vastly different target audiences: experienced industry professionals, and children.

Although PlayTIME is catered to game designers, it may be worthwhile to see if children's imaginations and creativity affect the output when assigned a simple task to complete. Professionals may feel more inclined to complete the task as requested, as they were seen to do in the current study. Children are naturally more playful and may not understand *why* they are asked to follow a set tasks, and would therefore be more interested in just building a level that makes them happy.

Hypothesis: I hypothesise that children would yield more creative results whereas professionals would aim to complete a task more directly with less creativity. Children may also report that the system is more enjoyable, and professionals will be more critical of the system.

Demographics: The study should be catered towards children with varying creative interests, and towards game development professionals with expertise or advanced skills in design.

Metrics: Performance would yield the most useful data to determine how PlayTIME is handled by the different groups, time distributions, and how errors are committed. The final output should also be evaluated for patterns between the two groups.

Method: The study should follow a *between-groups* design, where the groups are children and professionals. A free-form activity with minor constraints, such as the one described in Chapter 3, would allow for more creative freedom.

7.3 Summary

This chapter discusses the results and takeaways from evaluating PlayTIME, a game prototyping tool that uses tangible interfaces to help us complete game prototyping tasks. Guidelines and suggestions for future implementations and studies are suggested. Following these steps should produce cleaner and more manageable results in future evaluations of TIME systems for all disciplines: games, film, animation, and whatever else the software suite may cover. There are many types of media out there that these lessons can help developers strive to improve.

7.4 Concluding Remarks

When my grandfather asked me one December evening in 2009, "How do you make a living?" I figured the best way to describe what I did in school was to show rather than tell. I had my laptop with me, and all of my assignments from the fall semester were on it.

For one class I had worked on a small platformer game to demonstrate animation principles, sprite sheets and simple physics. My grandfather, who I am sure had never touched a computer in his life, surely did not see the principles of animation as he played the game with a subtle smile on his face. What he saw was a new, unfamiliar form of art that he could *play* with, using only a few keys.

For another class I created a simple drawing program, like a tiny and boring MS Paint, to demonstrate what I had learned about graphics, primitives in OpenGL and programming. My grandmother did not see all of this technical jargon about graphics and who-knows-what, and it certainly wasn't boring to her. What she saw was a playful canvas that let her *create* the images in her mind of colourful flowers, hearts, stars, and write the message "Trudy loves Daniel" in surprisingly clean cursive, given how difficult it was to use with my laptop's track pad.

The playfulness of digital games and related technologies is recognized across the generations, and the ability to use these technologies to create and refine ideas is also widely understood.

Whether we are talking about a triple-A, flashy, realistic title, or a digital mock-up of what could one day be the next big indie game, or a board game with little figurines; the medium used does not change the fundamentals of *play.* The challenge is not creating and playing, the challenge is bridging the gap between the two. The creative side exists in all of us, and we have digital tools that help us harness creativity to build digital prototypes quickly. The playful side exists as well; we have physical utilities that let us explore our ideas and build hands-on, physical versions of what we want to eventually become some form of digital media. The media we need to design playful experiences exist in many forms, but the challenge remains: we need these media to work together to speed up our work and build top-quality products.

The good news is that some of PlayTIME's users found it playful. I hope the insight gathered by evaluating PlayTIME can make a difference in scenario development and lay a brick or two in the foundation for full-scale tangible scenario development.

References

[1] Unity Technologies, "Unity 3D." [Computer software], 2005. unity3d.com.

[2] G. W. Fitzmaurice, H. Ishii, and W. A. S. Buxton, "Bricks: Laying the foundations for graspable user interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, (New York, NY, USA), pp. 442–449, ACM Press/Addison-Wesley Publishing Co., 1995.

[3] M. Billinghurst, "Shared space: Collaborative augmented reality," in *ACM SIGGRAPH 99 Conference Abstracts and Applications*, SIGGRAPH '99, (New York, NY, USA), pp. 178–, ACM, 1999.

[4] H. Kato and M. Billinghurst, "Marker tracking and HMD calibration for a video-based augmented reality conferencing system," in *Augmented Reality, 1999. (IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop on*, pp. 85–94, 1999.

[5] J. R. Lewis, "IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use," *International Journal of Human-Computer Interaction*, vol. 7, no. 1, pp. 57–78, 1995.

[6] J. Lewis, "Psychometric evaluation of the computer system usability questionnaire: The csuq," tech. rep., Tech. Rep, 1992.

[7] J. R. Lewis, "Psychometric evaluation of the post-study system usability questionnaire: The PSSUQ," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 36, no. 16, pp. 1259–1260, 1992.

[8] E. Cherry and C. Latulipe, "Quantifying the creativity support of digital tools through the creativity support index," *ACM Trans. Comput.-Hum. Interact.*, vol. 21, pp. 21:1–21:25, June 2014.

[9] E. A. Carroll and C. Latulipe, “The creativity support index,” in *CHI ’09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA ’09, (New York, NY, USA), pp. 4009–4014, ACM, 2009.

[10] E. A. Carroll, C. Latulipe, R. Fung, and M. Terry, “Creativity factor evaluation: Towards a standardized survey metric for creativity support,” in *Proceedings of the Seventh ACM Conference on Creativity and Cognition*, C & C ’09, (New York, NY, USA), pp. 127–136, ACM, 2009.

[11] 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, p. 1063, 1988.

[12] Free Radical Design, “TimeSplitters: Future Perfect.” [Nintendo GameCube], 2005.

[13] K. Salen and E. Zimmerman, *Rules of Play: Game Design Fundamentals.* The MIT Press, 2003.

[14] J. Schell, *The Art of Game Design: A Book of Lenses.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[15] B. Brathwaite and I. Schreiber, *Challenges for Game Designers.* Rockland, MA, USA: Charles River Media, Inc., 1 ed., 2008.

[16] T. Fullerton, *Game Design Workshop: A Playcentric Approach to Creating Innovative Games.* Burlington, MA, USA: Morgan Kaufmann, 2008.

[17] T. Fullerton, “That’s entertainment: Playcentric design,” *interactions*, vol. 15, pp. 42–45, Mar. 2008.

[18] T. Fullerton, J. Chen, K. Santiago, E. Nelson, V. Diamante, A. Meyers, G. Song, and J. DeWeese, “That cloud game: Dreaming (and doing) innovative game design,” in *Proceedings of the 2006 ACM SIGGRAPH Symposium on Videogames*, Sandbox ’06, (New York, NY, USA), pp. 51–59, ACM, 2006.

[19] T. Fullerton, T. Furmanski, and K. ValaNejad, “Journey of discovery: The night journey project as "video/game art",” in *Proceedings of the 2007 ACM SIGGRAPH Symposium on Video Games*, Sandbox ’07, (New York, NY, USA), pp. 55–63, ACM, 2007.

[20] H. Sharp, Y. Rogers, and J. Preece, *Interaction Design: Beyond Human-Computer Interaction.* West Sussex, England: John Wiley & Sons, Inc., 2 ed., 2007.

[21] R. A. Virzi, "What can you learn from a low-fidelity prototype?," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 33, pp. 224–228, SAGE Publications, 1989.

[22] P. Green and L. Wei-Haas, "The rapid development of user interfaces: Experience with the wizard of oz method," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, pp. 470–474, SAGE Publications, 1985.

[23] P. Green and L. Wei-Haas, "The wizard of oz: a tool for rapid development of user interfaces," tech. rep., University of Michigan, Ann Arbor, MI, USA, 1985.

[24] J. Rudd, K. Stern, and S. Isensee, "Low vs. high-fidelity prototyping debate," *interactions*, vol. 3, pp. 76–85, Jan. 1996.

[25] M. Rettig, "Prototyping for tiny fingers," *Commun. ACM*, vol. 37, pp. 21–27, Apr. 1994.

[26] J. Nielsen, "Paper versus computer implementations as mockup scenarios for heuristic evaluation," in *Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction*, INTERACT '90, (Amsterdam, The Netherlands, The Netherlands), pp. 315–320, North-Holland Publishing Co., 1990.

[27] V. Bellotti, "Implications of current design practice for the use of hci techniques," in *Proceedings of the Fourth Conference of the British Computer Society on People and Computers IV*, (New York, NY, USA), pp. 13–34, Cambridge University Press, 1988.

[28] J. Nielsen, "Evaluating the thinking-aloud technique for use by computer scientists," in *Advances in Human-computer Interaction (Vol. 3)* (H. R. Hartson and D. Hix, eds.), pp. 69–82, Norwood, NJ, USA: Ablex Publishing Corp., 1992.

[29] R. Sefelin, M. Tscheligi, and V. Giller, "Paper prototyping - what is it good for?: A comparison of paper- and computer-based low-fidelity prototyping," in *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, (New York, NY, USA), pp. 778–779, ACM, 2003.

[30] J. Derboven, D. De Roeck, M. Verstraete, D. Geerts, J. Schneider-Barnes, and K. Luyten, “Comparing user interaction with low and high fidelity prototypes of tabletop surfaces,” in *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, NordiCHI ’10, (New York, NY, USA), pp. 148–157, ACM, 2010.

[31] J. Manker and M. Arvola, “Prototyping in game design: Externalization and internalization of game ideas,” in *Proceedings of the 25th BCS Conference on Human-Computer Interaction*, BCS-HCI ’11, (Swinton, UK, UK), pp. 279–288, British Computer Society, 2011.

[32] J. Manker, “Designscape - a suggested game design prototyping process tool,” *Eludamos. Journal for Computer Game Culture*, vol. 6, no. 1, pp. 85–98, 2012.

[33] C. Snyder, *Paper prototyping: the fast and easy way to design and refine user interfaces.* The Morgan Kaufmann series in interactive technologies, Morgan Kaufmann, 2003.

[34] Epic Games, “Unreal Engine.” [Computer software], 1998. unrealengine.com.

[35] Crytek, “CryEngine.” [Computer software], 2001. cryengine.com.

[36] YoYo Games, “GameMaker: Studio.” [Computer software], 1999. yoyo-games.com/studio.

[37] ProCore, “Prototype for Unity.” [Computer software]. protools-forunity3d.com/prototype/.

[38] “Global Game Jam.” globalgamejam.org.

[39] J. Musil, A. Schweda, D. Winkler, and S. Biffl, “Synthesized essence: what game jams teach about prototyping of new software products,” in *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*, vol. 2, pp. 183–186, May 2010.

[40] T. T. Hewett, “ACM SIGCHI Curricula for Human-computer Interaction,” tech. rep., ACM, New York, NY, USA, 1992.

[41] J. Nielsen and R. Molich, “Heuristic evaluation of user interfaces,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI ’90, (New York, NY, USA), pp. 249–256, ACM, 1990.

[42] J. Nielsen, “Finding usability problems through heuristic evaluation,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI ’92, (New York, NY, USA), pp. 373–380, ACM, 1992.

[43] J. Nielsen, *Usability Engineering.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[44] J. Nielsen, “Usability inspection methods,” in *Conference Companion on Human Factors in Computing Systems*, CHI ’94, (New York, NY, USA), pp. 413–414, ACM, 1994.

[45] R. Molich and J. Nielsen, “Improving a human-computer dialogue,” *Commun. ACM*, vol. 33, pp. 338–348, Mar. 1990.

[46] S. L. Smith and J. N. Mosier, “Guidelines for designing user interface software,” Aug. 1986. Available at http://hcibib.org/sam/.

[47] J. Nielsen, “Heuristic evaluation,” in *Usability Inspection Methods* (J. Nielsen and R. L. Mack, eds.), pp. 25–62, New York, NY, USA: John Wiley & Sons, Inc., 1994.

[48] J. Johnson, *Designing with the Mind in Mind.* Burlington, MA, USA: Morgan Kaufmann, 2010.

[49] P. M. Fitts, “The information capacity of the human motor system in controlling the amplitude of movement.,” *Journal of experimental psychology*, vol. 47, no. 6, p. 381, 1954.

[50] I. S. MacKenzie, “Fitts’ law as a research and design tool in human-computer interaction,” *Hum.-Comput. Interact.*, vol. 7, pp. 91–139, Mar. 1992.

[51] I. S. MacKenzie and W. Buxton, “Extending fitts’ law to two-dimensional tasks,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI ’92, (New York, NY, USA), pp. 219–226, ACM, 1992.

[52] A. F. Blackwell, G. Fitzmaurice, L. E. Holmquist, H. Ishii, and B. Ullmer, “Tangible user interfaces in context and theory,” in *CHI ’07 Extended Abstracts on Human Factors in Computing Systems*, CHI EA ’07, (New York, NY, USA), pp. 2817–2820, ACM, 2007.

[53] H. Ishii and B. Ullmer, "Tangible bits: Towards seamless interfaces between people, bits and atoms," in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, CHI '97, (New York, NY, USA), pp. 234–241, ACM, 1997.

[54] H. Ishii, "Tangible bits: Beyond pixels," in *Proceedings of the 2Nd International Conference on Tangible and Embedded Interaction*, TEI '08, (New York, NY, USA), pp. xv–xxv, ACM, 2008.

[55] H. Ishii, M. Kobayashi, and K. Arita, "Iterative design of seamless collaboration media," *Commun. ACM*, vol. 37, pp. 83–97, Aug. 1994.

[56] J. Blake, *Natural User Interfaces in. NET: WPF 4, Surface 2, and Kinect*. Manning, 2011.

[57] Microsoft, "Kinect for Windows." [Computer hardware]. microsoft.com/en-us/kinectforwindows/.

[58] Microsoft, "Xbox Kinect." [Gaming hardware]. xbox.com/en-CA/Kinect/.

[59] Leap Motion, Inc., "Leap Motion." [Computer hardware]. leapmotion.com.

[60] Sony, "PlayStation Move." [Gaming hardware]. us.playstation.com/ps3/playstation-move/.

[61] Nintendo, "Nintendo Wii." [Gaming hardware]. wii.com.

[62] R. Azuma, "Overview of augmented reality," in *ACM SIGGRAPH 2004 Course Notes*, SIGGRAPH '04, (New York, NY, USA), ACM, 2004.

[63] H. Kato, "ARToolKit." [Computer software], 1999. hitl.washington.edu/artoolkit/.

[64] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[65] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '07, (Washington, DC, USA), pp. 1–10, IEEE Computer Society, 2007.

[66] M. Billinghurst, H. Kato, and I. Poupyrev, “The magicbook: a transitional ar interface,” *Computers & Graphics*, vol. 25, pp. 745–753, 2001.

[67] M. Billinghurst, H. Kato, and I. Poupyrev, “Magicbook: Transitioning between reality and virtuality,” in *CHI ’01 Extended Abstracts on Human Factors in Computing Systems*, CHI EA ’01, (New York, NY, USA), pp. 25–26, ACM, 2001.

[68] H. Kaufmann and A. Dünser, “Summary of usability evaluations of an educational augmented reality application,” in *HCI (14)*, pp. 660–669, 2007.

[69] H. Kaufmann, “Collaborative augmented reality in education,” *Institute of Software Technology and Interactive Systems, Vienna University of Technology*, 2003.

[70] H. Kaufmann and D. Schmalstieg, “Mathematics and geometry education with collaborative augmented reality,” in *ACM SIGGRAPH 2002 Conference Abstracts and Applications*, SIGGRAPH ’02, (New York, NY, USA), pp. 37–41, ACM, 2002.

[71] C. Wolfe, J. Smith, W. Phillips, and T. Graham, “Fiia: A model-based approach to engineering collaborative augmented reality,” in *The Engineering of Mixed Reality Systems* (E. Dubois, P. Gray, and L. Nigay, eds.), Human-Computer Interaction Series, pp. 293–312, Springer London, 2010.

[72] M. Billinghurst, H. Kato, K. Kiyokawa, D. Belcher, and I. Poupyrev, “Experiments with face-to-face collaborative ar interfaces,” *Virtual Reality*, vol. 6, no. 3, pp. 107–121, 2002.

[73] M. Billinghurst, S. Weghorst, and I. Furness, T., “Shared space: An augmented reality approach for computer supported collaborative work,” *Virtual Reality*, vol. 3, no. 1, pp. 25–36, 1998.

[74] R. Grasset, P. Lamb, and M. Billinghurst, “Evaluation of mixed-space collaboration,” in *Proceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR ’05, (Washington, DC, USA), pp. 90–99, IEEE Computer Society, 2005.

[75] E. Hornecker and T. Psik, “Using artoolkit markers to build tangible prototypes and simulate other technologies,” in *Proceedings of the 2005 IFIP TC13 Inter-*

national Conference on Human-Computer Interaction, INTERACT'05, (Berlin, Heidelberg), pp. 30–42, Springer-Verlag, 2005.

[76] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana, "Virtual object manipulation on a table-top ar environment," in *Augmented Reality, 2000. (ISAR 2000). Proceedings. IEEE and ACM International Symposium on*, pp. 111–119, 2000.

[77] M. H. Rooke, "Organic board games with tangible tiles: interaction methods for small hexagonal tiles," Master's thesis, Queen's University, Kingston, Ontario, Canada, 2009. hdl.handle.net/1974/1817.

[78] D.-N. T. Huynh, K. Raveendran, Y. Xu, K. Spreen, and B. MacIntyre, "Art of defense: A collaborative handheld augmented reality board game," in *Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games*, Sandbox '09, (New York, NY, USA), pp. 135–142, ACM, 2009.

[79] J. Marco, E. Cerezo, and S. Baldassarri, "Toyvision: A toolkit for prototyping tabletop tangible games," in *Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '12, (New York, NY, USA), pp. 71–80, ACM, 2012.

[80] J. Marco, I. Oakley, E. Cerezo, and S. Baldassarri, "Designing and making a tangible tabletop game with toyvision," in *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction*, TEI '13, (New York, NY, USA), pp. 423–426, ACM, 2013.

[81] M. Kaltenbrunner and R. Bencina, "reactivision: A computer-vision framework for table-based tangible interaction," in *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, TEI '07, (New York, NY, USA), pp. 69–74, ACM, 2007.

[82] Tangible Play, Inc., "Osmo," 2014. playosmo.com.

[83] Technical Illusions, "castAR." [Computer hardware]. technicalillusions.com.

[84] Wacom, "Cintiq 24HD Creative Pen Display." [Computer hardware]. wacom.com/en/us/creative/cintiq-24-hd/.

[85] S. R. Khattak, D. S. Buckstein, and A. Hogue, "Reconstructing 3D buildings from LIDAR using level set methods," in *2013 International Conference on Computer and Robot Vision*, pp. 151–158, IEEE, May 2013.

[86] S. Khattak and A. Hogue, “Sculpting beyond the screen,” in *IEEE Games, Entertainment and Media Conference*, 2014.

[87] S. Khattak, B. Cowan, I. Chepurna, and A. Hogue, “A real-time reconstructed 3d environment augmented with virtual objects rendered with correct occlusion,” in *IEEE Games, Entertainment and Media Conference*, 2014.

[88] Oculus VR, “Oculus Rift.” [Computer hardware]. oculusvr.com.

[89] Virtuix, “Virtuix Omni.” [Computer hardware]. virtuix.com.

[90] Unity Technologies, “AngryBots.” [Computer software]. u3d.as/content/unity-technologies/angry-bots/5CF.

[91] Jenkins Software, LLC, “RakNet.” [Computer software]. jenkinssoftware.com.

[92] W. H. Kruskal and W. A. Wallis, “Use of ranks in one-criterion variance analysis,” *Journal of the American statistical Association*, vol. 47, no. 260, pp. 583–621, 1952.

[93] F. Wilcoxon, “Individual comparisons by ranking methods,” *Biometrics bulletin*, pp. 80–83, 1945.

[94] H. B. Mann and D. R. Whitney, “On a test of whether one of two random variables is stochastically larger than the other,” *The annals of mathematical statistics*, pp. 50–60, 1947.

[95] R Development Core Team, *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.

[96] SurveyMonkey, “SurveyMonkey.” [Website], 2014. surveymonkey.com.

[97] R. Likert, “A technique for the measurement of attitudes.,” *Archives of psychology*, 1932.

[98] J. R. Lewis, “Psychometric evaluation of an after-scenario questionnaire for computer usability studies: The ASQ,” *SIGCHI Bull.*, vol. 23, pp. 78–81, Jan. 1991.

[99] J. R. Lewis, “An after-scenario questionnaire for usability studies: Psychometric evaluation over three trials,” *SIGCHI Bull.*, vol. 23, pp. 79–, Oct. 1991.

[100] J. Brooke, "SUS-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, p. 194, 1996.

[101] J. Sauro and J. R. Lewis, "When designing usability questionnaires, does it hurt to be positive?," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, (New York, NY, USA), pp. 2215–2224, ACM, 2011.

[102] T. Tullis and B. Albert, *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics.* Interactive Technologies, Elsevier Science, 2013.

[103] J. R. Lewis and J. Sauro, "The factor structure of the system usability scale," in *Human Centered Design*, pp. 94–103, Springer, 2009.

[104] J. P. Chin, V. A. Diehl, and K. L. Norman, "Development of an instrument measuring user satisfaction of the human-computer interface," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '88, (New York, NY, USA), pp. 213–218, ACM, 1988.

[105] J. R. Crawford and J. D. Henry, "The positive and negative affect schedule (PANAS): Construct validity, measurement properties and normative data in a large non-clinical sample," *British Journal of Clinical Psychology*, vol. 43, no. 3, pp. 245–265, 2004.

[106] J. Sauro and J. S. Dumas, "Comparison of three one-question, post-task usability questionnaires," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, (New York, NY, USA), pp. 1599–1608, ACM, 2009.

[107] TechSmith Corporation, "Camtasia Studio." [Computer software], 2014. techsmith.com/camtasia.html.

[108] P. E. Black, "Manhattan distance." [Online], 2006. nist.gov/dads/HTML/manhattanDistance.html.

[109] Qualcomm Technologies, Inc., "Vuforia Augmented Reality SDK." [Computer software]. qualcomm.com/products/vuforia.

[110] Metaio GmbH, "Metaio SDK." [Computer software]. metaio.com/products/sdk.

Appendices

Appendix A

Designer Actions Study Materials

A.1 Activity Materials

Task Description

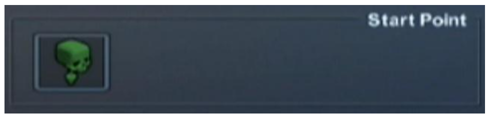
You are a puzzle designer working on a first-person shooter game. The game's protagonist has been travelling through time on a mission to collect all of the **Time Crystals**, artefacts used to power a time machine built by Earth's top scientists.

Prototype one of the levels in the game using the provided editor. The level must be puzzle-oriented: it will have some puzzle element to it. The player's objective in the level is to find the crystals, which you will place throughout the level. The player will have five minutes to complete the mission, so the level must be relatively small and straightforward.

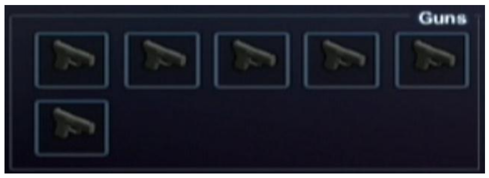
Constraints

- The game genre is first-person shooter
- The player will only have five minutes to play through the level
- The level must have a puzzle mechanic
- The player must collect crystals; you must place one or more crystals within the map
- You have doors, keys, switches and other objects to help you build a compelling puzzle mission

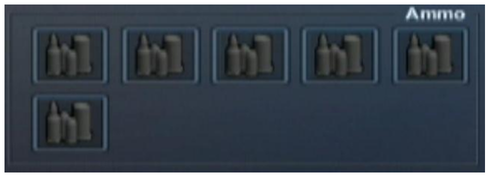
ITEMS ALLOWED:



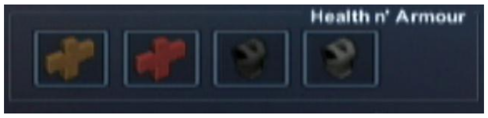
Start Point (1 max)



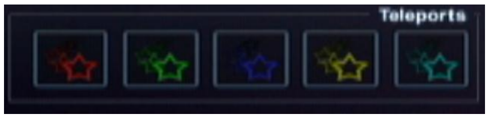
Guns



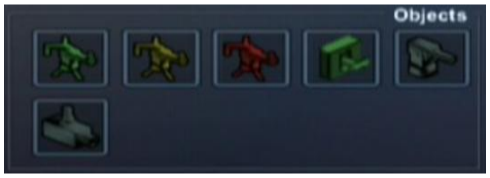
Ammo



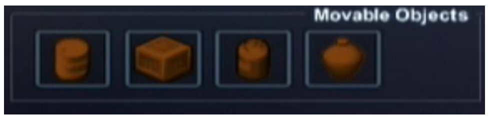
Health n' Armour



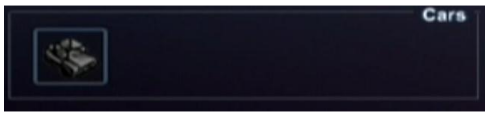
Teleports (2 per colour)



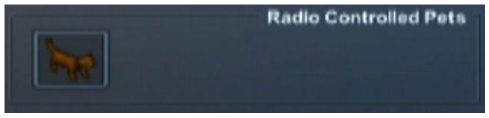
Objects



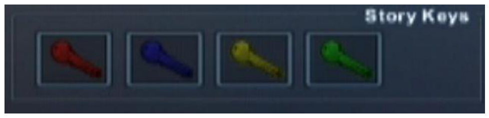
Movable Objects



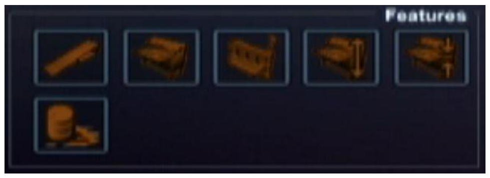
Cars



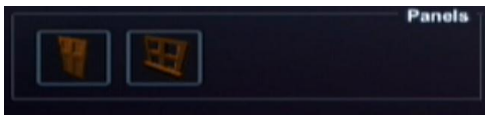
Radio Controlled Pets



Story Keys



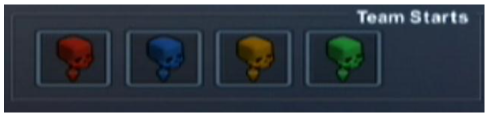
Features



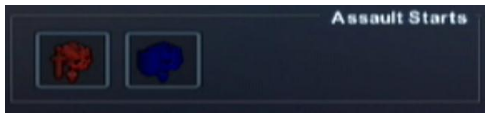
Panels

ITEMS NOT ALLOWED:

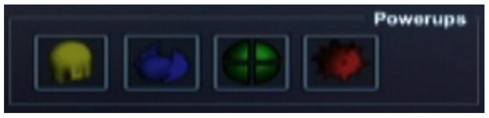
Team Starts



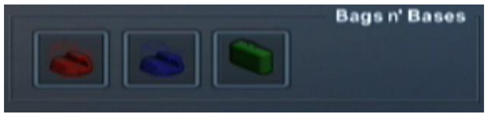
Assault Starts



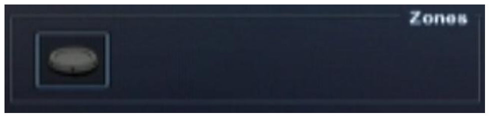
Powerups



Bags n' Bases



Zones



Additional Notes

ALL tiles allowed
ALL light effects allowed
ALL enemy A.I. allowed
ALL STORY logic allowed
ASSAULT logic **NOT** allowed!
Focus on ***logic***, ***triggers*** and ***actions***!
Only set up ***awards*** if you have time!

Control Scheme

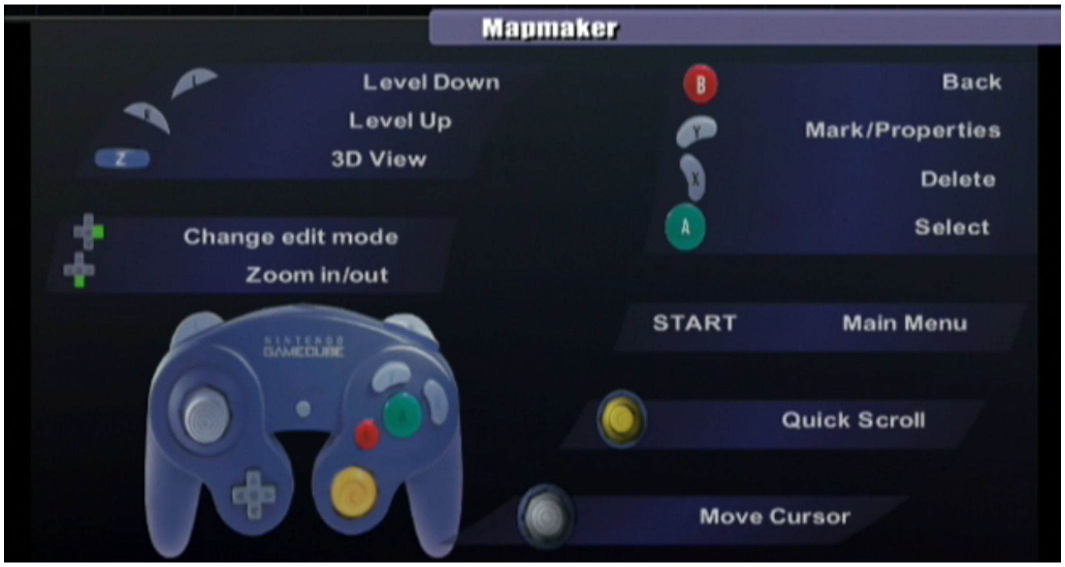


Figure A.1: Control layout for the TSFP mapmaker using a Nintendo GameCube controller.

Informed Consent Form for Experiment
This study has received ethical approval from the UOIT Research Ethics Board (REB)
File 13-022

Investigators:

Daniel Buckstein
Daniel.Buckstein@uoit.ca
Dr. Andrew Hogue, Faculty of Business and IT (Ext. 3698),
Andrew.Hogue@uoit.ca

Consent is an ongoing process. This consent form is only part of the process of informed consent. It should give you the basic idea of what this research is about and what your participation will involve. If you would like more detail about something mentioned here, or information not included here, please ask your experimenter or any of the investigators listed above. Please take the time to read this form carefully and to understand any accompanying information. A copy of this form is available for your records.

This purpose of this study is to gain a better understanding of the process of video game design and prototyping, and determine the steps developers use to reach a goal and why they are relevant. You will be assigned a design-related task using a level editing tool with a set of assets.

Risks of the experiment are minimal. Screen capture software will be used to record your progress, with video recording used as a backup. All data is stored confidentially and anonymously. Anything that can identify you as a participant will be known only to the principal investigator (D. Buckstein) and will be anonymized for data analysis. Opinions and thoughts or feelings expressed will not change your relationship with the researchers or UOIT.

Your participation in this study is completely voluntary and you may interrupt or end the study at any time without giving a reason or fear of being penalized.

If at any point during the experiment you feel uncomfortable and want to end your participation, please let the experimenter know and the study will end immediately. There are no consequences for withdrawal. Withdrawing participants will have all of their data destroyed.

The session will require about 70-90 minutes, during which you will be asked to complete assigned tasks, completing a questionnaire after your session.

At the end of the session, you will be given the opportunity to request more information about the purpose and goals of the study, and there will be time for you to ask questions about the research.

Thank you very much for your time and help in making this study possible. If you have any inquiries or wish to know more please contact Daniel Buckstein or Dr. Andrew Hogue.

Faculty of Business and Information Technology
2000 Simcoe St N, Oshawa, ON L1H 7K4

Daniel Buckstein:
Email: Daniel.Buckstein@uoit.ca

Dr. Hogue:
Phone: 905-721-8668 Ext. 3698 or email: Andrew.Hogue@uoit.ca

For any queries regarding this study, please contact the UOIT Research and Ethics Committee Compliance officer (compliance@uoit.ca and 905-721-8668 Ext. 3693).

After reading this information, you give consent.

- I understand that taking part in this study is my choice and that I am free to withdraw from the study at any time without reason and irrespective of whether or not payment is involved.
- This consent form will be kept in a locked filing cabinet in Oshawa for a period of seven years before being destroyed.
- I have read and understand all of the above information
- I understand that I am not waiving any of my legal rights

I, ____
(First name, Last name, Signature), agree to take part in this research.

Voluntary and optional consent for photographic release
Please sign below if you would like to give us photographic consent to use a video of you and the experimental setup in research reports and presentations.

I would like to explicitly grant Daniel Buckstein and Andrew Hogue, and their research assistants, the right to use the screen capture session for presenting this study in publications, such as scientific journals and magazines, and research presentations. I understand that the video material is not linked to any personal data outside of this experiment that may identify me.

The non-visual data collected from this study will be used in articles for publication in journals and conference proceedings. All data gathered is stored anonymously and kept confidential. Only the investigators of this study and their research assistants may access and analyze the data. All published data will be coded, so that your non-visual data is not identifiable.

I, ____
(First name, Last name, Signature), give consent to use video and image material of myself and the experimental setup in research reports and presentations.

Optional As one way of thanking you for your time, we will be pleased to make available to you a summary of the results of this study once they have been compiled (usually within two months). This summary will outline the research and discuss our findings and recommendations. If you would like to receive a copy of this summary, please leave your preferred contact information with the researchers.

____

Appendix B

Designer Actions Study Data

B.1 Questionnaires

This section presents the questionnaires used to collect self-reported data from participants.

First, the demographics questionnaire asked participants to provide a description of their experience by rating their expertise with a variety of tools and in a variety of developer roles.

The post-study questionnaire asked participants to rate their experiences with the system they used to complete their task.

Designer Actions Study:
Demographics Questionnaire

1. Please indicate your age.
2. How many years of game development experience do you have?

3. Please rate your level of expertise in the following game development roles:
   a. Gameplay programmer
   b. Graphics programmer
   c. Tools programmer
   d. Technical designer
   e. Systems designer
   f. Level designer
   g. User interface designer
   h. 3D modeller
   i. Animator
   j. 2D artist
   k. Concept artist
   l. Character artist
   m. Environment artist
   n. Interface artist
   o. Writer
   p. Sound designer/composer
4. From the above list (or other), which role do you feel suits you best?

5. Please rate your level of expertise using the following tools for developing games.
   a. Visual Studio
   b. Xcode
   c. Eclipse
   d. Unity3D
   e. Unreal
   f. CryEngine
   g. GameMaker
   h. Blender
   i. Maya
   j. 3DS Max
   k. Mudbox
   l. Zbrush
   m. Photoshop
   n. Illustrator
   o. MS Paint
   p. Paint.NET
   q. Gimp
   r. Pen and paper
   s. Lego blocks
   t. Other toys
6. Please rate how frequently you use these tools when you are developing games.
   (same options as above)
7. From the above list (or other), which tool best suits your everyday game development needs?

8. What methods and tools do you use to prototype your games, and/or pieces of your games (including any combination of these)?
   a. I write and draw ideas on paper
   b. I think about my game really hard
   c. I make a small demo in a game engine
   d. I modify existing game engine projects

e. I use existing games' built-in level editors
f. I draw environment layouts on paper
g. I use 3D modelling/animation software
h. I change the rules of existing board games
i. I make a text-based game
j. I make a card game (like Pokemon)
k. I build things out of Lego bricks
l. Who needs prototypes? Just go for it!
m. Other

9. Here is a list of features you might see in game development software. Please rate the importance of these features.
   a. Three-dimensional editing
   b. Tile-/grid-based scenario mapping
   c. Free-form scenario mapping
   d. Object placement and manipulation
   e. A variety of pre-made objects to use
   f. Objects with predefined behaviours and meanings
   g. Objects with properties you can change
   h. Change mood and atmosphere
   i. Character/actor placement
   j. Characters/actors with predefined behaviours
   k. Characters/actors with properties you can change
   l. Control over the camera
   m. Scripting or some programming tool
   n. Basic logic editor or layout (no programming, just describes the flow of logic in the scenario)
   o. Mark which objects and characters are involved with/controlled by logic
   p. Applicable textures and materials
   q. Animation and motion editors
   r. Scene hierarchy
   s. Sound effects and music (while editing)
   t. Being able to stop working and continue later

10. From the initial idea to a playable game, please briefly describe the steps you take during your design and prototyping process.

11. Any general comments about your game prototyping experiences?

<u>Designer Actions Study:</u>
<u>Post-Study Questionnaire</u>

1. Please rate how you feel about the features which you experienced while completing your task.
    a. Interactive preview
    b. Interactive preview quality
    c. Memory status bar
    d. Multiple floors
    e. Enemy patrol paths
    f. Game logic editor
    g. Changing item properties
    h. Overall experience
    i. Copy, rotate, delete
    j. "Drag n' drop"
    k. Changing lighting
    l. Placing enemy AI
    m. Displaying logic connections
    n. Connecting objects with logic
    o. Placing and moving items
    p. Choice of tiles
    q. Choice of items
    r. Choice of enemy AI
    s. Logic trigger creation
    t. Logic trigger editing
    u. Logic action creation
    v. Logic action editing
    w. Placing and moving tiles
    x. Changing enemy properties
    y. Control scheme
    z. Menu navigation

2. Had you ever used this editor before this experiment?

3. What things do you like the most about this editor?

4. What features of this editor work well for game prototyping?

5. What things do you like the least about this editor?

6. What features of this editor do not work well for game prototyping?

7. Please describe your favourite experience(s) while completing your task (e.g. anything that made you extremely happy, excited, etc.).

8. Please describe your least favourite experience(s) while completing your task (e.g. anything that drove you nuts).

9. Any general questions, comments or concerns about the experiment?

B.2 Design Process

"From the initial idea to a playable game, please briefly describe the steps you take during your design and prototyping process."

P01: "- Paper brainstorm

- Paper layout and breakdown of game mechanic fundamentals

- Implementation of fundamental via software"

P02: "- Find Inspiration (often other games, life experiences or my critiques of games)

- Sit down and find out what I really want from this game, and what experience I want to deliver

- Get rough idea on paper

- Implement (program) features that I know will be essential

- Keep designing the rest of the game

- Repeat the above two points until game is playable and at least sort of fun

- Polish by adding assets and pretty visuals, polish gameplay as well"

P03: "Initial idea

Rethink the initial idea

Talk to friends about the refined idea

Sketch idea on paper or build 3D prototypes (using e.g. Lego)

Build a prototype in Unity3D

Pretend the prototype was not a prototype and just continue developing on it for the actual game (I know, one shouldnt do that, but everybody does)"

P04: "Establish premise and main mechanics, implement current ideas,test them in a prototype build. Reiterate."

P05: "I would normally start with getting my ideas on paper to get an idea of what it is I'm going to tackle. I would then try to get basic mechanics down and play around using those mechanics and primitives. I would then lay out a basic level to test out something more complete. I would then start adding in actual assets to replace temporary objects. After that I would add in the more context sensitive mechanics and polish the game, fixing any bugs, until completion."

P06: "Design comes from an idea. The moment you have a base mechanic or idea you can expand that into a game with rules and story. After you get the story or "what is going to happen" you can flesh it out either vertically or horizontally to get a finished product/game"

P07: "-Crazy idea

-Write it down (main mechanic)

-Forget about it for a week

-Come back to it, design core systems

-Prototype something

-Realize X doesn't work

-Fix it

-Add to prototype

-Repeat"

P08: "I think about what kind of game I'd like to play and try to think of mechanics that would be different, perhaps, than something else that's out there. Sometimes, though, it's not about original mechanics, but a certain atmosphere I want to play in.

I start thinking about how to implement the mechanics and try to get them to work. I also start making placeholder assets and figure out the game structure"

P09: "- Idea

- Draw and brainstorm for ideas that conform to the original (or they may not and are better, at least on paper)

- Start building the core mechanics

- Play test the core mechanics

- Modify mechanics as needed

- Start building other mechanics

- Play test the mechanics

- Modify as needed

- Start building the game (at the same time the prototype may be extended, depending on the game, to include playable levels for testing while the 'actual' game catches up."

P10: "-concept of gameplay implement:

-character control

-main mechanic

-level restrictions/goals

-level design

-interactions with the level

-other mechanics

-enemies

-animations

-sounds

-menus/states

-polish"

P11: "1 - come up with a cool mechanic

2 - implement a paper prototype

3 - test it

4 - get feedback from subjects who test it

5 - tweek it

6 - return to 3 until i am happy with it

7 - document everything in a sort of game design doc. so that the idea and the mechanic itself is not lost

I can perhaps later add in one or two more mechanics to add some content, which would have followed the same process as well."

P12: "-Imagine a game idea

-Prototype what it will look like in photoshop

-Figure out how to make it look like that

-Write down how I need to implement the logic

-Attempt the basic logic in code

-Use placeholder art and finish mechanics

-Add final art, and add other essential elements such as menus

-polish the graphics to make everything smooth and nice"

P13: "- concept draft

- rough prototype

- evaluation

- prototype

- evaluation

- deliver"

P14: "-Come up with a hook

-Brainstorm the hook's surrounding game (2D, 3D, sidescroller, etc.) on paper or a digital analog

-Prototype the hook and test its level of ""fun"" using primitives or a basic level editor (Unity3D)

-Make changes and proceed to expanding the prototype, have others play

-Add/test complimenting features/mechanics and test their compatibility

-If the mechanics and gameplay work, continue with the game's creation"

P15: "- Brainstorm (both within my head and with others)

- Start to work some details out on paper

- Begin to rough prototype in Unity (very simple - use prefab shapes, concentrate on core idea)

- Iterate a lot

- Flush out prototype, define details of core mechanic and begin to conceptualize secondary ideas and how systems might work together.

- Basically I like to come up with a few small ideas, and then attempt to combine them together to create more complex systems.

- I like to design from narrative, what I'm I trying to say how will that translate into a players experience."

B.3 Feedback

"What things do you like the most about this editor?"

P01: "- Quick placement of items

- Easy lists to deal with"

P02: "Tiles are very modular, and they fit together nicely. Very little confusion, and you can build on multiple levels."

P03: "The 3 steps for previewing:

a) What you see is what you get in 2D

b) static 3D preview

c) interactive 3D preview of gameplay

The ability to manipulate also logical attributes, although I didnt really have the time to try them out."

P04: "Pretty robust and extensive. Ability to alter logic for the level was pretty awesome as well."

P05: "I liked that it was easy to set up an entire level filled with different object with unique properties with no previous experience. Being able to tie things together with logic was my favourite feature as it allowed for the opportunity of many different puzzle options."

P06: "The simplicity was very nice. It was pretty easy to pick up. Once I was able to figure out the menus I was able to design without thinking too much about what I was doing mechanically with the controller."

P07: "Liked that everything was sectioned off into its own categories, ie: enemies in this tab, items here, floor tiles here. Also liked the streamlined logic creation for basic things like switches."

P08: "It took a while to get used to but once you get the hang of it the controls feel natural. Like all of the controls for one object is on one side, and all of the menu controls are on the other side. The bigger buttons are used more often, and all that stuff."

P09: "- Multiple levels

- Fast preview (the semi-3D non-rotatable preview - I assume it was non-rotatable because I couldn't rotate it!)

- Preview

- Item copy/placement"

P10: "The ability to test your level with a press of a button and all the previews"

P11: "I like the way things are organized. Although, the UI clearly needed some improvements like having tabs instead of scroll down menus. The selection of items was very easy. I liked the fact that you could use the object properties themselves to set some levels of game logic, instead of having to script that every time through the logic editor."

P12: "I like the tile based movement, it gives a feeling of precision"

P13: "- Quick placement of tiles

- good level overview

- easy navigation across height levels"

P14: "Tile-based and previewing"

P15: "The controls and tiles worked well on a game pad - I would have preferred a mouse and keyboard. Level layout was good once I got used to it, placing enemies and items was relatively simple."

"What features of this editor work well for game prototyping?"

P01: "- Has basic components to place for level design blueprints"

P02: "Working on individual parts of the level and testing each separately. Quick build times and relatively small but purposeful selection of tiles"

P03: "Placement of items was straightforward

Previewing (static and interactive) was neat"

P04: "Preview provides instant feedback on the design. Not just the live game preview, but the map itself helped to fix a lot of errors I made while designing the level."

P05: "The ability to quickly layout out level chunks and then preview them ingame is a great thing for prototyping."

P06: "It is really easy to create rooms and see how the size of the area will impact the player. It is really simple and if you have an error in your logic it will tell you exactly where it is and what it relates too."

P07: "Quick level layouts and basic logic allow for fairly simple game prototypes. Basically, we can prototype any really low level FPS as long as we scope the prototype level down to a bare minimum"

P08: "If you want a shooter with door logic and teleporters then it's great. It's good that the controls are fluid and comfortable to use. It doesn't strain your fingers"

P09: "- Fast preview

- Automatic generation of walls (so I don't have to close things off and waste time)

- Preview

- Copy/move items

- Undo"

P10: "The ease of being able to pick it up and just start building with it"

P11: "The fact that I could change some logic of some objects on the object properties themselves, and not on the logic editor, worked well for me. It made it easy to constantly visualize the level I was designing, instead of going back and forth between editors. "

P12: "The tile editor for quick level design"

P13: "- placement of things

- designing the rough layout of the level area"

P14: "Huge selection of items to drop, multiple levels to work on, nice control and tile-based placement, and being able to change logic in the editor"

P15:"Tiles work well for map layouts. There are lots of prefab objects to quickly place and jump into the game. Took me two minutes to place a couple of locks, keys and enemies to populate the level."

"What things do you like the least about this editor?"

P01: "- Triggers and actions would take a bit to remember to be able to use again. - Have to make a new enemy rather than copying one that already exists"

P02: "Game logic item placement, due to it not being put in the item menu. I would assume the game logic item would just be like any item, just highlighted to signify its importance."

P03: "The fact that one always needs to be in the correct mode (like AI, Item, Tiles...) to interact with the items was really annoying."

P04: "Until I had the map (press Z) I didn't like how I had to play the game to see if I messed up or not."

P05: "The fact that it is controller based definitely makes things a bit harder to use. Navigating different levels of the world got confusing, I kept going up when I wanted to go down. I liked the logic editor, but the speed at which you can do things with it is limited, again because of the controller. While the choice of tiles is nice, I think it would have been good to be able to have an open space and add in walls, instead of pre-walled chunks."

P06: "I wasn't a huge fan of having to move through the menus with a gamecube controller. If they added a next function on the controller or made it switch through without the freeroam final fantasy style with sub menus that would probably have allowed to me move a lot faster. It was like using a track ball as opposed to a touch screen. Very hard."

P07: "Controls. This is a pretty big problem in most level editors that are done within existing games, especially when the input method is some kind of gamepad. Moving around this sort of interface without the ease of a mouse and keyboard just feels super clunky and even frustrating at times. I think the controller just adds an extra level of input complexity by feeling so unnatural for this specific task."

P08: "If you want to make any other type of game then you can't."

P09: "- Interface

- Preview takes a while to load and unload (understandable)

- No flythrough? Or at least it wasn't clear I could do a fly through

- Why not make the fast preview rotatable?"

P10: "Some of the logic connections were a little difficult to navigate with"

P11: "The should definitely be a copy and paste functionality for multiple tiles at once. It was annoying to copy and paste one tile at a time. The item and tile selection could benefit greatly from a tab sort of organization... it would have made the selection process of the items way easier."

P12: "Navigating it using a controller is difficult, as in it makes it slower to switch through items"

P13: "- logic not really accessible

- no real-time preview of the map

- height tiles have no intuitive representation in the map editor, the preview of the tiles is fine when selecting it, but when placing the tile, it looks very abstract and one cannot relate it back to the height level and tile type it represents

- navigation within the editor felt limited, I felt the GameCube controller made selecting things and navigating through the menus really cumbersome

- overall I did find assigning names and behaviours to enemies took too long

- real-time previews of the levels would have been helpful

- overall really cumbersome interface with hard-to-memorize icons kept me guessing at the functions"

P14: "Only that the load times for testing the level are rather high"

P15: "I found the elevation layers confusing for the majority of the experiment - but do feel I was close to getting it. Also some of the previews for certain tiles, gave me the wrong impression. The ability to rotate the preview would help."

"What features of this editor do not work well for game prototyping?"

P01: "- Due to a grid base system and components it is very rigid and not as flexible as I would like"

P02: "The AI would have to be tweaked around with extensively, and would probably not be a good tool for quickly dropping enemies in your level to assess the relative difficulty."

P03: "Connecting items (such as green key to enemy and to door) was very tricky

It was really hard to find out, where connections between the multiple floors need to be placed (like ramps), because there was only a top-down view while placing them."

P04: "Nothing really comes to mind. I mean the features all work really well together once you understand what the editor is telling you e.g. sliiiides."

P05: "The controller does not work well for prototyping as it greatly hinders speed of use. Also having a limit on certain objects is a large annoyance."

P06: "It seemed a bit confusing at times what the corridors were doing, how they fit together and what level you were on. You got a vague idea of what level you were on but 50 shades of grey isn't enough. Some color coding would have been great. IE level 1 blue, level 2 red, 5 orange etc."

P07: "There isn't a lot of room to add your own mechanics or a robust scripting tool (as it's on the GameCube). At the end of the day, this tool is still just a level editor that forces you to design within the constraints of the existing TimeSplitters engine."

P08: Well the fact that you're very limited in what you can do. You can't make your own objects or have the freedom to make puzzles that involve different things.

P09: "- Undo should have a button

- When I place multiple levelled item, I had to place it, go to the other level, grab it and replace it in the first level just so that I can get the cross-walks aligned with the level. Would have liked the buttons to go back and forth between levels when placing multi-level items

- Not clear enough what is walkable - a clear green outline and shading would have been better (or better yet, a nice connected graph showing what's walkable)"

P10: "Not as much character mobility options. Restricted to FPS with low movement"

P11: "There were some gimmicks with the teleports, where the exit direction is actually opposite to the one that the item arrow points to. I also think that the tile system is a bit confusing, since you have to lay all the tiles, one by one. I would have perhaps implemented something like allowing the user to draw a path and from that select the tiles. If the user does not want that, then he can pick a specific tile."

P12: "Difficulty getting puzzles and logic set up in a short amount of time"

P13: "I think the biggest problem is the controller interface. Even with a mouse and keyboard, I would have had a better time editing and probably it would have been possible to navigate around the editor quicker. I general, I found it hard to move from the abstract representation back to the actual 3D game at the start."

P14: "Being unable to easily attach multiple switches to doors (AND, OR, XOR logic gates"

P15: "It was obviously geared toward a certain genre of games. I don't prefer to use a game pad to design games - struggled a bit to understand the interface. Felt it got in the way, spent the majority of the first half learning it rather then designing the game."

"**Please describe your favourite experience(s) while completing your task (e.g. anything that made you extremely happy, excited, etc.).**"

P01: "- Setting up the enemy turrets to be used as both the difficult component and the key to some areas"

P02: "I was able to build a level that spanned 3 floors, and had a "final boss room" in the middle that would always harass the player until they have everything done"

P03: "Entering the interactive preview mode and seeing what a beautiful world I had created. Especially, driving around in the car was great. Otherwise, I liked conceiving the puzzle before / while actually putting it together and testing it."

P04: "Having it all work in the end. Good vibes."

P05: "I really liked using logic to tie multiple things together to make a puzzle. I also liked using lights to try to make a puzzle, even though I never completed it."

P06: "I really enjoyed previewing the music was great. I also liked how simple it was to create logic and that perked me up, it wasn't tedious to go through the options to create the logic."

P07: "I enjoyed finally understanding how the level editor worked. As with any prototyping tool or even software in general, there was a somewhat substantial learning curve. The a-ha moment is always satisfying, especially in this case when I was able to actually execute what was in my head."

P08: "It was fun being able to play-test it with real AI that would shoot you back."

P09: "- Many tile options when building the level"

P10: "Shooting monkeys and seeing the level work the way it was suppose to"

P11: "It was fun to set the AI enemies, as well as having automatic turrets that can be controlled and/or disabled from afar with a switch."

P12: "I enjoyed testing the level and seeing it come to life quickly"

P13: "I was really happy about the radio-controlled pets. I would say these things can be considered a delighter for gameplay that is very much action and run and gun focused. I liked the switches and timers. I also enjoyed naming things even though it was difficult to do with a controller."

P14: "Being able to quickly drop entire chunks of level down using prefabs, rather than manually defining size, applying textures, filling levels, etc. Drag and drop is amazing."

P15: "It worked :) I was able to complete and play something within 45 mins."

"Please describe your least favourite experience(s) while completing your task (e.g. anything that drove you nuts)."

P01: "- Not being able to use the triggers to use cameras and turrets together"

P02:

P03: "Trying to figure out where to point and what buttons to press in the GUI."

P04: "SLIIIIIIDES"

P05: "I disliked when I discovered certain objects had limits to the amount you could have in a level. It happened near the end and it stopped me from being able to create a puzzle. Although I never reached the limit, I feel as though the limit on how much can be in a level total could become a hindrance to people, especially people with more time to create a level."

P06: "I didn't like the way that the zoom was setup. I wanted to see more of my map or less, and it would switch what menu I was in. I would end up selecting the tile instead of the item. The worst part of the experiment was the moving of the cursor. If I was able to just press right once and it would jump to the next available spot icould palce my object that would have been great."

P07: "I mentioned this before, but the controls don't feel great. I suppose for the time and technology they are serviceable, but but it just feels awkward navigating these sorts of menus with a joystick."

P08: "I never figured out what Radio Controlled Pets were for :c"

P09: "- Placing the many tile options

- Fast preview (useful, but at the same time, wanted it to show me more)"

P10: "Navigating through the logic menus was a little difficult and I needed to ask the experiment runner how to do things"

P11: "Editing levels with a gamepad just feels weird. I would have preferred a mouse."

P12: "The time limit made it very difficult to use all of the games puzzles and logic so I was unable to make a challenging puzzle in time"

P13: "Navigating the editor menus was really hard and recalling what icons represented was difficult."

P14: "The...load...times!!!"

P15: "Elevations not matching up - I couldn't figure out how to get the tunnel to line up."

Appendix C

PlayTIME User Study Materials

C.1 Activity Materials

Task Description

You are a level designer tasked with completing one of the stages in a new top-down shooter project, "*Angry Bots.*" The game is anticipated to be a huge success, but it will be quite boring without a player avatar or any bad guys to shoot! Your expertise is required to place characters throughout the map of the level and make them behave properly.

The programmers have developed the player's and enemies' behaviours, and another level designer has provided you with a map of current game scene and a short list of tasks for you to complete before the next team meeting. The map has been segmented into zones, which should each have a number of enemies.

HERE IS YOUR TO-DO LIST (complete in *any order*, and *preview* as you feel necessary):

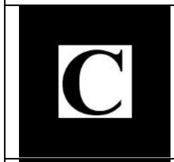
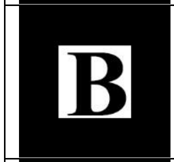
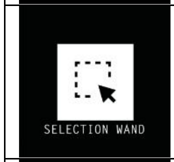
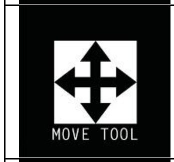
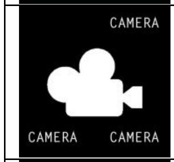
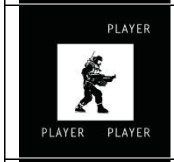
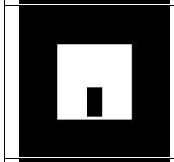
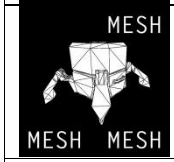
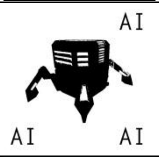
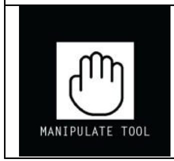
1. Place the ***Player*** object as close to the marker as possible (***X*** in the map, ***red square*** in Unity)
2. Place ***2 Buzzers each*** in zones 1, 2 and 3 (see map)
3. Place ***1 Spider each*** in zones 1, 2 and 3
4. Place ***7 spiders*** anywhere in zone 4
5. Place ***2 spiders*** anywhere in zone 5
6. Ensure ***all*** the spiders have ***AI behaviours attached***
7. Change the attack radius of at least ***4 spiders anywhere***

Things you need to know about gameplay:

- *Make the best possible prototype in a short amount of time!*
- *Do not place anything outside the boundaries!*
- The gameplay programming is done! You ***should not*** program anything!
- The ***door*** (see map) will open when all of the outdoor enemies have been destroyed (zones 1-3).
- The player must be on-level with enemies to shoot them.
- The game is ***won*** when all enemies have been destroyed (counters displayed in-game).
- The game is ***lost*** if the time limit (***two minutes***) is exceeded (timer displayed in-game).

INSTRUCTIONS AND CONSTRAINTS (PlayTIME activity)

1) Controls (tangible AR paddles)

Marker	Name	Description
	Confirm action 'C'	Treat like a button (hold your finger over it for a second to press the button). Press to confirm current action (manipulation, placement…) with a different marker. NOTE: You can move the button markers around as you feel necessary!
	Delete selection 'B'	Treat like a button. Press to delete currently-selected objects or components while the appropriate marker is visible. EXAMPLE: some spiders are selected and the AI marker is visible; pressing this button will delete those spiders' AR components.
	Selection wand	Move the selection cube over an object to select it. MULTI-SELECT FEATURE: repeat this action to add multiple objects to the selection to be manipulated simultaneously. DESELECT: when marker is visible, and away from objects, press confirm.
	Move tool	Move the selected object(s).
	Camera pan	Move the marker farther away from the red square in the middle of the view to pan the camera around. The direction of panning is relative to the red square.
	Player object	Position marker and press confirm to place the player object in the map.
	Buzzer object	Position marker and press confirm to place a buzzer (flying enemy) object in the map.
	Spider object	Position marker and press confirm to place a spider (mechanical exploding enemy) object in the map.
	Spider AI behaviour	Place marker over a spider and press confirm to add an AI component to the spider. If multiple spiders are already selected, press confirm to add AI to all spiders in the selection.
	Manipulate tool	Manipulate properties of the currently-selected object(s), namely the attack radius variable on the spiders' AI behaviour. When the red square appears, moving the marker towards the left side of the square will decrease the attack radius, and towards the right will increase the radius.

2) THINGS YOU ***<u>ARE</u>*** ALLOWED TO DO:

- Use the above markers to complete the assigned tasks
- Use the keyboard and mouse ***FOR PREVIEWING ONLY!!!!***
    - Press the play button to begin the preview
    - Use the keyboard and mouse to control the player object
        - WASD keys for movement, point and click with mouse to shoot
    - Press the play button again to end the preview

3) THINGS YOU ***ARE NOT*** ALLOWED TO DO:

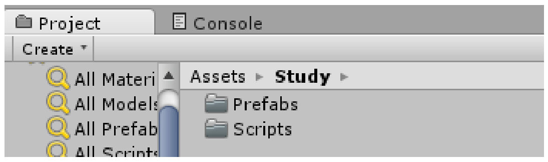
- Use the ***MOUSE and KEYBOARD*** for ***ANY REASON*** other than previewing
    - This includes all keyboard shortcuts and menus!!!
- Change the experiment setup (webcam, monitors, computer…)
- Change ***ANY*** part of Unity’s configuration

INSTRUCTIONS AND CONSTRAINTS (Unity activity)

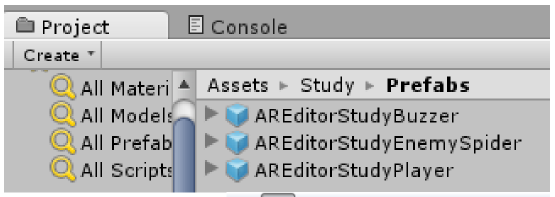
1) Controls (mouse)

The interface uses the MOUSE to control the placement and modification of objects.

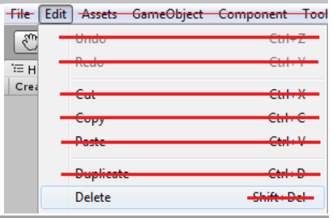
All objects you will need to use are found in "Assets > Study"; do not use assets in any other folder! To change folders, _left click_ on the folder you want to access.



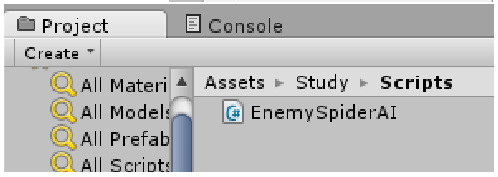
To place an object (PLAYER, SPIDER or BUZZER), _left click_ the desired PREFAB from the PROJECT tab near the bottom of the screen. The prefabs are located in the folder "Assets > Study > Prefabs". _Drag_ the prefab to the desired location and release the mouse button.



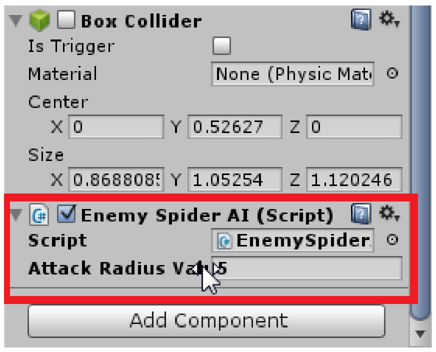
To delete an object, first _left click_ on the PLAYER, SPIDER(S) or BUZZER(S) that you wish to delete; this selects the object. The Delete command can be found under the Edit menu: _Edit > Delete_.



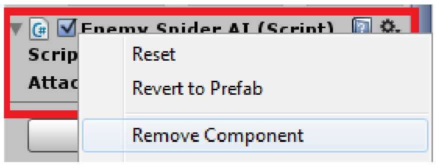
To add AI to a spider object, _left click_ the SCRIPT from the project tab. The spider AI script is located in the folder "Assets > Study > Scripts". _Drag_ the script on to the spider that you wish to apply the behaviour to and release. Objects must not be selected!



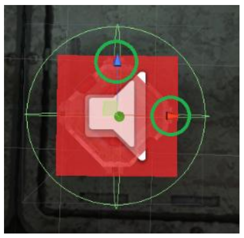
To change the spider's attack radius, locate the behaviour component in the right-most panel of Unity's layout; the spider behaviour is located at the bottom. Place the cursor over the variable name "Attack Radius Value", _left click and drag_ left and right to change the value.


Box Collider
Is Trigger
Material None (Physic Mat
Center X 0 Y 0.52627 Z 0
Size X 0.868808 Y 1.05254 Z 1.120246
Enemy Spider AI (Script)
Script EnemySpider
Attack Radius Val 5

To delete the AI component, _left click_ on the SPIDER that you wish to remove the AI from. _Right click_ on the name of the component and select "Remove Component".



To move an object, _left click and drag_ on the RED OR BLUE directional arrow on the selected object.



To pan the camera, _middle click and drag_ the mouse.

2) THINGS YOU ***ARE*** ALLOWED TO DO:

- Use the ***MOUSE ONLY*** to place, remove and modify assets in the scene to complete the assigned tasks
    - Worry about the ***PLAYER, SPIDERS and BUZZERS only!***
- Use the keyboard ***FOR PREVIEWING ONLY!!!!***
    - Press the play button to begin the preview
    - Use the keyboard and mouse to control the player object
        - WASD keys for movement, point and click with mouse to shoot
    - Press the play button again to end the preview

3) THINGS YOU ***ARE NOT*** ALLOWED TO DO:

- Use the ***KEYBOARD*** for any reason other than previewing
    - This includes all keyboard shortcuts!!!
    - DO NOT save the scene
    - DO NOT undo or redo actions
    - DO NOT copy, paste or duplicate objects
    - Etc.
- Zoom or rotate the camera
- Change the experiment setup (webcam, monitors, computer…)
- Change ***ANY*** part of Unity's configuration
- Navigate or use assets from outside the ***Assets > Study*** folder
- Use the hierarchy

Informed Consent Form for Experiment
This study has received ethical approval from the UOIT Research Ethics Board (REB)
File 14-014

Investigators:

Daniel Buckstein
Daniel.Buckstein@uoit.ca
Dr. Andrew Hogue, Faculty of Business and IT (Ext. 3698),
Andrew.Hogue@uoit.ca

Consent is an ongoing process. This consent form is only part of the process of informed consent. It should give you the basic idea of what this research is about and what your participation will involve. If you would like more detail about something mentioned here, or information not included here, please ask your experimenter or any of the investigators listed above. Please take the time to read this form carefully and to understand any accompanying information. A copy of this form is available for your records.

This purpose of this study is to compare the effectiveness of game design tools using different interfaces. The domain of study is level design and prototyping. You will be provided with a game design task along with the assets and tools required to complete the task. Please note that your skills are not the subject of evaluation; we are interested in your use of the provided tools.

Risks of the experiment are minimal. Screen capture software will be used to record your progress. All data is stored confidentially and anonymously. Anything that can identify you as a participant will be known only to the principal investigator (D. Buckstein) and will be anonymized for data analysis. Opinions and thoughts or feelings expressed will not change your relationship with the researchers or UOIT.

Your participation in this study is completely voluntary and you may interrupt or end the study at any time without giving a reason or fear of being penalized. This includes having no bearing or influence on one's privacy, reputation or academic standing at UOIT.

You do not waive any legal rights by agreeing to participate in this study.

If at any point during the experiment you feel uncomfortable and want to end your participation, please let the experimenter know and the study will end immediately. There are no consequences for withdrawal. Withdrawing participants will have all of their data destroyed. You may optionally request to have your visual data destroyed after participating in the study.

The session will require about 75-90 minutes. You will be required to complete a demographic questionnaire, an assigned prototyping task using a game design tool, and finally a questionnaire after your session.

At the end of the session, you will be given the opportunity to request more information about the purpose and goals of the study, and there will be time for you to ask questions about the research.

Thank you very much for your time and help in making this study possible. If you have any inquiries or wish to know more please contact Daniel Buckstein or Dr. Andrew Hogue.

Faculty of Business and Information Technology
2000 Simcoe St N, Oshawa, ON L1H 7K4

Daniel Buckstein:
Email: Daniel.Buckstein@uoit.ca

Dr. Hogue:
Phone: 905-721-8668 Ext. 3698 or email: Andrew.Hogue@uoit.ca

For any queries regarding this study, please contact the UOIT Research and Ethics Committee Compliance officer (compliance@uoit.ca and 905-721-8668 Ext. 3693).

After reading this information, you give consent.

- I understand that taking part in this study is my choice and that I am free to withdraw from the study at any time without reason and irrespective of whether or not payment is involved.
- This consent form will be kept in a locked filing cabinet in Oshawa for a period of seven years before being destroyed.
- I have read and understand all of the above information
- I understand that I am not waiving any of my legal rights

I, ____
(First name, Last name, Signature), agree to take part in this research.

Voluntary and optional consent for photographic release
Please sign below if you would like to give us photographic consent to use a video of you and the experimental setup in research reports and presentations. If you prefer to participate in the experiment, but not to have your image released, simply return this form without signing this optional section.

I would like to explicitly grant Daniel Buckstein and Dr. Andrew Hogue, and their research assistants, the right to use the screen capture session for presenting this study in publications, such as scientific journals and magazines, and research presentations. I understand that the video material is not linked to any personal data outside of this experiment that may identify me.

The non-visual data collected from this study will be used in articles for publication in journals and conference proceedings. All data gathered is stored anonymously and kept confidential. Only the investigators of this study and their research assistants may access and analyze the data. All published data will be coded, so that your non-visual data is not identifiable.

I, ____
(First name, Last name, Signature), give consent to use video and image material of myself and the experimental setup in research reports and presentations.

Optional As one way of thanking you for your time, we will be pleased to make available to you a summary of the results of this study once they have been compiled (usually in two months). This summary will outline the research and discuss our findings and recommendations. If you wish to receive a copy of this summary, please leave your preferred contact information with the researchers.

____

Appendix D

PlayTIME User Study Data

D.1 Questionnaires

This section presents the questionnaires used to collect self-reported data from participants.

First, the demographics questionnaire asked participants to provide a description of their experience by rating their expertise with a variety of tools and in a variety of developer roles. The lists of software and developer roles were extended for this study to reflect skills from a broader audience, such as animators and film makers, instead of only those involved with games.

The post-condition questionnaire was the same for both conditions. This included a few scientific questionnaires: the Positive and Negative Affect Schedule (PANAS) [11] questionnaire for rating momentary emotional evaluation; the Computer System Usability Questionnaire (CSUQ) [5], also known as the Post-Study System Usability Questionnaire (PSSUQ) for rating system usability; and the Creativity Support Index (CSI) [8] for rating the creative potential of the systems.

Finally, the post-study questionnaire asked participants to rate the ease of use of various factors that they experienced in each activity, and state their preferred interface for completing tasks.

PlayTIME User Study:
Demographics Questionnaire

1. Please indicate your age.
2. How many years of game development experience do you have? Include full-time, contract work, internships, teaching, studying, independent, etc.

3. Please rate your level of expertise using the following software.
   NO ANSWER: Do not answer if you have never used the tool, very rarely used it for its purposes, or have no interest in ever using the tool. If you have to ask what the tool is, this is probably the best option!
   BEGINNER: Select Beginner if you have used the tool a few times and/or are beginning to learn and become familiar with the interface.
   COMPETENT: You have used the tool enough to complete basic tasks on your own, but you may still refer to help from others for more difficult tasks pertaining to the tool.
   INTERMEDIATE: You can complete tasks of varying difficulty on your own using the tool. Help is not regularly needed.
   ADVANCED: You use the tool regularly, and have been using it regularly for some time, and you can solve complex problems on your own.
   EXPERT: Consider yourself an 'Expert' if you have been using the tool for an exceptionally long time, and very frequently, for most of your duties. (10K hours rule)
   a. Visual Studio
   b. Xcode
   c. Eclipse
   d. GameMaker
   e. Unity3D
   f. Unreal
   g. CryEngine
   h. Blender
   i. Maya
   j. 3DS Max
   k. Mudbox
   l. Zbrush
   m. Photoshop
   n. Illustrator
   o. Dreamweaver
   p. Flash
   q. Premiere
   r. FMOD Sound Designer
   s. Audacity
   t. Paint.NET
   u. Gimp
   v. MS Paint (*for scenario & production design only)
   w. Pen and paper (*for scenario & production design only)
   x. Lego blocks (*for scenario & production design only)
   y. Other toys (*for scenario & production design only)
   z. Statistics software (R, SPSS... *for scenario & production design only)
   aa. Excel (*for scenario & production design only)
   bb. Other (please specify)

4. Please rate how frequently you use these tools when you are developing games. (same options as above)
5. Which of the above tools (or other) best suits your everyday needs?

6. Please rate your level of expertise in the following production roles.
   NO ANSWER: Do not answer if you have never taken on this role and/or have no interest in being this role.
   BEGINNER: Select Beginner if you have taken on this role for a very short time and require supervision.
   COMPETENT: You have had responsibilities in this role for long enough that you can make some decisions on your own, but sometimes require feedback, advice or supervision.
   INTERMEDIATE: You have enough experience in this role to make some big decisions, and work with and manage other people.
   ADVANCED: You use frequently act in this role (e.g. as a job) and have enough experience to support, direct and guide other people in this role.
   EXPERT: Consider yourself an 'Expert' if you are a seasoned veteran of the responsibilities of this role, perhaps even leading teams from time to time. (10K hours rule)Gameplay programmer
   a. Gameplay programmer
   b. Tools programmer
   c. Graphics programmer
   d. Software developer (tool software design & programming...)
   e. Technical designer (asset management, integration...)
   f. Level designer (environments, worlds, object placement...)
   g. Systems designer (mechanics, rules, prototyping...)
   h. User interface designer (interaction with system)
   i. Writer (story, dialogue, characters...)
   j. Film/game director (instruct actors)
   k. Film/game producer (management & production)
   l. Film editor
   m. Cinematographer (camera work)
   n. Actor (physical presence in a scene, including film, mocap, voice...)
   o. Concept artist
   p. 2D artist
   q. Environment artist
   r. Character artist
   s. Interface artist
   t. 2D animator
   u. 3D animator
   v. 3D modeller
   w. Sound designer/composer
   x. Games user researcher (evaluation, user experience...)
   y. Other (please specify)

7. From the above list (or other), which role do you feel suits you best?

8. What methods and tools do you use to prototype your games, and/or pieces of your games (including any combination of these)?
    a. I write and draw ideas on paper
    b. I think about my game really hard
    c. I make a small demo in a game engine
    d. I modify existing game engine projects
    e. I use existing games' built-in level editors
    f. I draw environment layouts on paper
    g. I use 3D modelling/animation software
    h. I change the rules of existing board games
    i. I make a text-based game
    j. I make a card game (like Pokemon)
    k. I build things out of Lego bricks
    l. Other

9. Here is a list of features you might see in scenario development software (games, film, animation). Please rate the importance of these features. Feel free to think in terms of your specialization.
    a. Three-dimensional editing
    b. Tile-/grid-based scenario mapping
    c. Free-form scenario mapping
    d. Object placement and manipulation
    e. A variety of pre-made objects to use
    f. Objects with predefined behaviours and meanings
    g. Objects with properties you can change
    h. Change mood and atmosphere
    i. Character/actor placement
    j. Characters/actors with predefined behaviours
    k. Characters/actors with properties you can change
    l. Control over the camera
    m. Scripting or some programming tool
    n. Basic logic editor or layout (no programming, just describes the flow of logic in the scenario)
    o. Mark which objects and characters are involved with/controlled by logic
    p. Applicable textures and materials
    q. Animation and motion editors
    r. Scene hierarchy
    s. Sound effects and music (while editing)
    t. Being able to stop working and continue later

u. Being able to work with other people
v. Being able to preview your scenario instantly
w. Other (please specify)

(The PANAS questionnaire was inserted here for the pre-study measurement. See below for the PANAS questionnaire.)

10. Any general comments about your game prototyping experiences?

D.1.1 Post-Condition

Positive and Negative Affect Schedule (PANAS)

"This scale consists of a number of words that describe different feelings and emotions. Read each item and then select the number from the scale below. Indicate to what extent you feel this way."

Ratings: 1 - Not at all; 2 - A little; 3 - Moderately; 4 - Quite a bit; 5 - Extremely

1. Interested
2. Distressed
3. Excited
4. Upset
5. Strong
6. Guilty
7. Scared
8. Hostile
9. Enthusiastic
10. Proud
11. Irritable
12. Alert
13. Ashamed
14. Inspired
15. Nervous
16. Determined
17. Attentive
18. Jittery
19. Active
20. Afraid

Computer System Usability Questionnaire (CSUQ)

"This questionnaire gives you an opportunity to express your satisfaction with the usability of the system. Your responses will help us understand what aspects of the system you are particularly concerned about and the aspects that satisfy you. To as great a degree as possible, think about all the tasks that you have done with the system while you answer these questions. Please read each statement and indicate how strongly you agree or disagree with the statement by selecting a number on the scale. Base

your responses around the way you interacted with the system during the previous exercise."

Ratings: 1 - Strongly disagree; 2 - Disagree; 3 - Somewhat disagree; 4 - Neither agree nor disagree; 5 - Somewhat agree; 6 - Agree; 7 - Strongly agree

Overall, I am satisfied with how easy it is to use this system.

It is simple to use this system.

I can effectively complete my work (the assigned tasks and scenarios) using this system.

I am able to complete my work quickly using this system.

I am able to efficiently complete my work using this system.

I feel comfortable using this system.

It was easy to learn to use this system.

I believe I became productive quickly using this system.

The system gives error messages that clearly tell me how to fix problems.

Whenever I make a mistake using the system, I recover easily and quickly.

The information (on-screen messages, documentation) provided with this system is clear.

It is easy to find the information I needed.

The information provided for the system is easy to understand.

The information is effective in helping me complete the tasks and scenarios.

The organization of information on the system screens is clear.

The interface of this system is pleasant.

I like using the interface of this system.

This system has all the functions and capabilities I expect it to have.

Overall, I am satisfied with this system.

Creativity Support Index (CSI)

"Please rate your agreement with the following statements."

Ratings: 1 - Strongly disagree; 2 - Disagree; 3 - Somewhat disagree; 4 - Neither agree nor disagree; 5 - Somewhat agree; 6 - Agree; 7 - Strongly agree

The system would allow other people to work with me easily.

It would be really easy to share ideas and designs with other people using this system.

I would be happy to use this system on a regular basis.

I enjoyed using the system.

It was easy for me to explore many different ideas, options, designs, or outcomes, using this system.

The system was helpful in allowing me to track different ideas, outcomes, or possibilities.

I was able to be very creative while doing the activity inside this system.

The system allowed me to be very expressive.

My attention was fully tuned to the activity, and I forgot about the system that I was using.

I became so absorbed in the activity that I forgot about the system that I was using.

I was satisfied with what I got out of the system.

What I was able to produce was worth the effort I had to exert to produce it.

"When completing tasks using this system, it's most important that I'm able to..." (pick one from each pair)

Be creative and expressive/Become immersed in the activity

Be creative and expressive/Enjoy using the system or tool

Be creative and expressive/Explore many different ideas, outcomes, or possibilities

Be creative and expressive/Produce results that are worth the effort I put in

Be creative and expressive/Work with other people

Become immersed in the activity/Enjoy using the system or tool

Become immersed in the activity/Explore many different ideas, outcomes, or possibilities

Become immersed in the activity/Produce results that are worth the effort I put in

Become immersed in the activity/Work with other people

Enjoy using the system or tool/Explore many different ideas, outcomes, or possibilities

Enjoy using the system or tool/Produce results that are worth the effort I put in

Enjoy using the system or tool/Work with other people

Explore many different ideas, outcomes, or possibilities/Produce results that are worth the effort I put in

Explore many different ideas, outcomes, or possibilities/Work with other people

Produce results that are worth the effort I put in/Work with other people

D.1.2 Post-Study

"Please rate the ease of use for each of the following tasks using Play-TIME's Tangible AR Markers only. If you did not use a feature, leave that row blank."

Ratings: 0 - No answer; 1 - Very difficult; 2 - Difficult; 3 - Slightly difficult; 4 - Neither easy nor difficult; 5 - Slightly easy; 6 - Easy; 7 - Very easy

Confirming placement actions with the 'C' marker

Deleting world objects by presenting the object marker and pressing 'B'

Selecting one or more objects with the 'Selection Wand' marker

Deselecting all objects using the selection wand and C marker

Moving selections with the 'Move Tool' marker

Panning the camera around with the 'Camera Pan' marker

Placing the player object with the 'Player Object' marker

Placing flying buzzer enemies with the 'Buzzer Object' marker (has a small square)

Placing exploding spider enemies with the 'Spider Object' marker (labelled Mesh)

Adding AI behaviour to one spider enemy directly using the 'Spider AI Behaviour' marker (labelled 'AI')

Adding AI behaviour to multiple selected spiders using the spider AI marker

Changing the attack radius of selected spiders using the 'Manipulate Tool' marker

[Excluded] Deleting AI components by presenting the AI marker and pressing 'B'

"Please rate the ease of use for each of the following tasks using just the mouse. If you did not use a feature, leave that row blank."

Ratings: 0 - No answer; 1 - Very difficult; 2 - Difficult; 3 - Slightly difficult; 4 - Neither easy nor difficult; 5 - Slightly easy; 6 - Easy; 7 - Very easy

Navigating through the project assets folders (Assets > Study > Prefabs and Scripts)

Placing the player in the world by clicking and dragging the prefab

Placing flying buzzers in the world by clicking and dragging the prefab

Placing exploding spiders in the world by clicking and dragging the prefab

Selecting an object by clicking on it

Deleting an object from the world

Applying the spider's AI behaviour to a spider by clicking and dragging the script

Modifying a spider's attack radius

Moving an object around the world using the mouse

Panning the camera using the middle mouse button

[Excluded] Removing a spider's AI component

"If you were given the option of choosing PlayTIME's tangible paddles, or to just click and drag the mouse to complete similar tasks, please select which interface you would prefer to use for completing each the following tasks. You may leave answers blank."

Placing a game object in the world

Adding a behaviour component to ONE object

Adding a behaviour component to MULTIPLE objects

Changing behaviour parameters on ONE object

Changing behaviour parameters on MULTIPLE objects

Deleting ONE game object from the world

Deleting MULTIPLE game objects from the world

Deleting a behaviour component from ONE object

Deleting a behaviour component from MULTIPLE objects

Selecting ONE object

Selecting MULTIPLE objects

Moving ONE object

Moving MULTIPLE objects

Panning the camera

Extra: If you knew how to rotate the camera...

D.2 Feedback

"Overall, which activity (PlayTIME or mouse) did you enjoy the most and why?"

P51: "Overall I enjoyed using the mouse more, however I feel that a main reason of this is that using a mouse is a very common tool which I use everyday/every other day, so I've gotten very used to it, where as PlayTIME was a new tool that I was unfamiliar with. I believe that if I could use PlayTIME more, I may enjoy it just as much/possibly more than a mouse."

P50: "I enjoyed the PlayTIME more for its full outside of the computer interaction and complexion to complete what would be or should be simple tasks. When simple tasks are made difficult I makes you think of how easy things are made for us especially in this technologically driven time. It also made it seem more like playing a game to create a game, I was almost more interested in making it then playing it."

P49: "Mouse because it felt like playing a PC game; felt more natural; playtime felt like it was a lot of effort for something that was not needed vs just click and done"

P48: "I enjoy the mouse but i use it in my daily life but the playtime feels more like a toy i can see kids using it then sending levels to there friends or building full games using per-made objects"

P47: "I enjoyed playtime the most it was interactive!!!! I wish there was a little more functionality but totally worth while. this system was new to me and i was able to set the scene with minimal difficulty i would like to have been able place work on prefabs (spider ai on spider mesh) or just in general going back to the little more functionality. this system would be a blast spending hours with and i think it could be used a lot when working with other people to build levels via the internet and other network type situations ie LAN"

P46: "PlayTIME I actually thoroughly enjoyed the interactivity of being able to move the tangable aspects of the level around and use the "screen capture" technology to help determine my actions. It seemed almost intuitive and like I was physically building something more than as if I was simply sitting behind a computer screen editing a level."

P45: "PlayTIME was definitely a more enjoyable experience. I felt more creative, more willing to work, and really genuinely enjoyed was I was doing, so much so that this didn't feel much like an experiment as much as it felt like I was hanging out with a friend and playing a game"

P44: "I enjoyed PlayTIME and the Mouse, however I found play time to be more fun, exciting and new. It was much easer to use than the program that I have to use (MAYA). I really liked how interactive it was and found I lost myself in the activity. The mouse however was a bit easier for me to use and navigate for some things (selection) with mainly because it is something I am more familiar with and it is a bit more precise as to what your selecting. I would enjoy using both or a mixture of the two."

P43: "PlayTIME was definitely more enjoyable because it felt like I was playing a game as opposed to working. However, if my goal was to be as efficient as possible, PlayTIME was substantially slower than the mouse which I am more accustomed to. If I had a choice, I would prefer to use both keyboard and mouse to be as fast as possible."

P42: "I enjoyed using PlayTIME more then the mouse overall because it was interesting way to interface with a computer. It wasn't something I had done before, and besides the learning curve (which I imagine would come with more use) it was fun and fairly easy to use."

P41: "I enjoyed PlayTIME the most. As someone who is used to regular Unity, the mouse without keyboard seemed like a bothersome limitation. But PlayTIME was just cool. I was like a kid discovering a toy for the first time. I mean all motion tracking AR stuff is neat and interesting, but to have it as a tool for a program I use daily was pretty awesome. It was like playing rather than working."

P40: "The PlayTIME was more enjoyable, mostly because it was novel."

P39: "Overall I enjoyed using the PlayTIME system over the mouse system. For me, the PlayTIME system was new and interesting. However, it's still in it's developmental stages and needs to be iterated upon until it can become a viable solution."

P38: "I did enjoy both interfaces. I found the PlayTIME to be much more interactive. It was harder to 'pick up' but enjoyable. Playtime had a slower work pace to it from taking the time to select each tool appropriately. I think that Overall, I actually enjoyed the mouse more. Just because I was more familiar with it and I was faster/more effective using it."

P37: "I enjoyed using PlayTIME more. its a novel way of being able to move around and place objects and made completing the assignment much quicker, even though its less accurate for fine-tuning certain things like the position of the actors or camera."

P36: "playtime -it's different from what i am used to - using just a mouse without a keyboard is tedious"

P34: "I actually enjoyed the PlayTIME activity more than the mouse, mostly because it was something fresh and exciting. However, in terms of actual productivity, I would say I enjoyed the mouse more than the PlayTIME simply because PlayTIME was much more tedious to perform the simple tasks such as placing and changing properties"

P33: "I enjoyed using PlayTIME more than the mouse. I use the mouse everyday and there is no excitement in it for me, however it was the first time that I used anything like PlayTime so using the paddles and making the scene using the paddles was like a game in itself. It made the activity a lot more enjoyable and fun."

P32: "PlayTIME, by far. The work flow was easy, and using a real life workspace was easier than navigating folders. Modifying behaviours was easier with playTIME as well."

P31: "Moving objects, because gives the sensation that I have more control over the environment."

"Please identify the strengths and limitations of each system."

P51: "The strength of the mouse is that its almost common practice or second nature to most people, its quicker, and arguably more precise. PlayTIME is very easy to understand and pickup, but has a few hassles in using it (ie. have to move yourself or the buttons/paddles because they interfere with the camera's view)."

P50: "Speed and Efficiency divides these two systems entirely however PlayTIME was definitely more fun then using just a mouse. PT was slow but highly functional for almost anyone to learn and complete the tasks asked, the mouse might have been a little more complicated for some as it wasn't as visual. I think that you could still over look the complex format of the system with how much it can be enjoyed as long as more buttons/ features were added. The mouse was a lot faster for sure, functions and speed still a problem though."

P49: "Limitations: Playtime Camera movement (too high or too low, too far...) I just really like the mouse :)"

P48: "the mouse is the base line for usability, the playtime is much or fun and given better tech a parent could play along making a learning tool. learning tools are easier to sale to parents, and the learning curve is at this point at a very appropriate level"

P47: "TIME was easy to learn but had some limitations you needed to be aware of where your arms were so as not to block things like buttons the tools could have been refined a little more for time (maybe write the name of the tool on the back of it(i liked to flip them over and leave them in scene) I did not find limitations really with the mouse just lack of hot keys took and prefabs creation messed with my normal work flow"

P46: "Unity is pretty decent (as it is used in the industry) but I think it's interface is kind of programming heavy. It requires some background knowledge of how to code a level before you can actually figure out how to do what you want to do. PlayTIME on the other hand is alot more friendly to people with only a creative background. It makes it seem more hands on and although you may not see the background coding, it helps to perform basic functions needed for a prototype game."

P45: "The strengths I'd say that PlayTIME had over the mouse, easily, was my interest. It was far more interesting to be using technologies that id never used before, whereas I am almost always using just a mouse and keyboard, or as in this case, a mouse alone. the mouse however is something that everyone already knows how to use, but lacks the immersion and , I feel, level of creativity. in an industry that relies on creativity, I feel that the PlayTIME tiles would definitely make for a great experience in the game development , animation, game programming, etc. industry. I am a strong believer in being able to do work effectively and have fun doing it; this is proof as far as I'm concerned. the only set back I saw that would make me want to choose the mouse over the tiles was that rotation might be an issue if you crossed your arm in front of the camera, and that it took me probably twice as long to complete the task with tiles just because I was having fun doing it"

P44: "PlayTIME and the Mouse are user friendly. PlayTIME is very interactive, engaging and fun, it is however harder to select with precision. Mouse is quite simple to use as long as you do the instructions and dont mess with anything else -restraints/messing up the program/messing up the map etc- PlayTIME dose not need restrictions as it only allows you to do what is needed and nothing else. You wont mess up the original file."

P43: "PlayTIME takes a bit of time to get used to, and requires switching back and forth between the tools which slows progress. It is much more fun though. The mouse is the fastest of the two tools, but not very exciting. It also is a bit limiting in that I am slowed down by being unable to use the keyboard."

P42: "For PlayTIME: The Strengths are not having to be physically tethered to the computer. It's an interesting and intuitive way to complete a task. However, the bounding box that you're required to work in at times can be a bit limiting, and using the tools isn't as familiar as a mouse. For the mouse: I've used a mouse for years, so it comes really easy to me, and it was easier to use some of the unity controls with a mouse. A good middle would be nice."

P41: "The strengths of PlayTIME to me were how simple the basic controls were, and how setting up a scenario took less prior knowledge of the system to be able to work with the tools. The weakness is that it was hard to get used to at first, as my initial reaction was to treat the "b" and "c" panels like buttons, and would forget to remove my hands out of the way. As well, the very slight delay makes it hard to move accurately, and though I'm not sure how necessary it was, but I felt I needed to go super slow so that the system could keep track of the cards. The strengths for the mouse are that it is also simple, accurate, and can move quickly. The drawbacks are having to navigate through menus and folders manually, and having an understanding of where things will be and what types of clicks do what function."

P40: "The mouse is much faster and more precise, and generally had superior function in every way. The PlayTime was more fun and novel. The important distinction, however, is that the PlayTIME can recognize objects. This could have a lot of potential for other applications... just not basic work functionality."

P39: "In regards to the mouse and keyboard system, I was definitely more competent with it as I have at least a decade's worth of experience with using it. In terms of it's limitations, I didn't like the fact that I was not allowed to use the keyboard short cuts or even redesign unity to the setup that best suits me. (Side Note) In game design, there should not be restrictions to how the user on how to get to the end point goal. So long as they get there with the desired specifications it doesn't matter the process that was used. In regards to the PlayTIME system, it definitely had it's strengths in getting the user to physically interact with the system. Although, I found that the lag and lack of response time with the system hindered it's performance with creating the final product. I believe that in this early stage, it is not viable as a developer tool as it does not have the speed that is needed for the industry. Perhaps when a later iteration goes out it would be viable as a developer tool."

P38: "PlayTIME+: interactive, requires movement (activity), easy to work with others. Mouse+: efficient, more familiar, more immersion. PlayTIME-: takes longer, a little harder to give commands. Mouse-: not as easy to work with others."

P37: "Using the mouse is very slow although it is way more accurate then PlayTIME. PlayTIME was very fun to use as it made it much easier to move the camera around quickly and even let me multi-task unlike the solo mouse set-up."

P36: "strengths of playtime: - it's fun and easy limitations of playtime: - the materials used to make the markers were a bit flimsy"

P34: "PlayTIME Pros: camera pan (much faster and smoother), being able to physically place objects. PlayTIME Cons: Time consuming, not efficient (took time to get used to controls and time everything correctly) Mouse Pros: Familiarity and efficiency- I have used a mouse my entire life, so the controls made sense. Mouse Cons: Poor camera pan - it took forever to get places, and I dont enjoy that."

P33: "I find panning in PlayTime to be rather difficult, and it takes a while to get used to. Moreover, it requires that you readjust when you are out of the field of view of the camera, which I was not paying much attention to and myself forgetting about the fact that the camera was there. Also, the fact that I accidentally waved my hand over something and it would cause a reaction from the system was rather frustrating. It had to be more attentive when using PlayTime versus using the mouse which has become automatic to me. It is much easier to select and manipulate multiple objects using PlayTime rather than using the mouse."

P32: "Having a physical representation of the items infront of me was much better than dragging and dropping, especially because placing multiple objects required only one paddle and a button. The only real limitation that I can fault the system for is the lag time of the camera. I found myself staring at the screen more than my hands, and the lag was disorienting."

P31: "Overall was a nice experience, it fells like I was playing a game instead of just making one. The problem is that theres too much markers, a lot of function can be simplified in fewer markers."

"Did doing the first activity help you complete the second activity in any way?"

P51: "Yes, Once I had an idea of how the activity was supposed to be done and its requirements, I feel that it helped a lot towards completing it again the second time."

P50: "knowing what I was suppose to do already is about it, other then placement of objects of course I had to almost skip the learning process I was just handed better tools to complete the task at hand."

P49: "Definitely. Understood everything with the mouse, it was simply remembering it and applying it to the "weird pad thing" (aka playtime)"

P48: "it let me know what needed to go where"

P47: "yes i was able to get used to using the prefabs and what the all did. It was my learning the assets and level phase. This is partially why it took less time to create the second level i knew where i wanted things to go also i am more used to using the mouse."

P46: "Slightly but not really. The first activity (in Unity) basically refreshed my knowledge from past experiences with it. (Just the process of inserting an object, apply an AI to it and then changing it's specific attack radius.) I actually felt the second activity was more intuitive and user friendly to me."

P45: "Yes, and I feel that maybe if I had done everything with a mouse first to get used to it, I'd probably have been able to use the PlayTIME tiles more efficiently, but that was not the case."

P44: "Yes I believe it did help me in already understanding my tasks and knowing what i vaguely wanted to do already."

P43: "After completing the first activity, I didn't have to refer back to the instructions sheet which made me take a more efficient route to completing the goals. However, it did not make me use the tools themselves any faster."

P42: "I think it definitely helped me complete the second activity, I knew what I was required to do, and I used the PlayTIME system and my previous tests in-game as prior knowledge."

P41: "Yes, I did all my play testing and figuring in the first activity, so the second activity was simple placement and scaling the aggro radius on spiders."

P40: "Yes, I was more aware of the system's parameters, the requirements of the task, the map layout, and what worked and didnt."

P39: "Doing the first activity did give me an advantage in the second activity however, it was mitigated by the fact that I needed to learn a new system in order to complete the task."

P38: "Yes. I got the opportunity to understand how unity works before venturing to use PlayTIME."

P37: "Yes, it prepared me for getting used to where I could place certain objects and how they would interact with the environment."

P36: "yes"

P34: Definitely - the first activity got me used to the need for AI, planning for placement of the enemies and overall layout of the map. I was much quicker at placing enemies once I understood the entire map, which I looked into roughly while understanding the controls of the PlayTIME."

P33: "It helped me complete the second activity in that I was more familiar with the map now, and knew where the player could and could not go, so where I could place objects, and where they would be out of reach."

P32: "In some ways, yes. I spent more time on the first activity trying to make sure that mechanics worked, such as opening doors and verifying that spiders were working. I spent less time thinking about placement on the second activity."

P31: "Yes"

D.3 Observed User Actions

D.3.1 PlayTIME

A total of 130 user actions were observed during PlayTIME activities. Here they are listed by subtask, with correct usages listed first.

Object Placement

PLAYER marker: Used properly (defined as "visible to position and place one or more of the player prefab object")

Marker out of bounds

Idle visible time (feature idle)

Marker occluded (more likely by a fingertip!)

BUZZER marker: Used properly (defined as "visible to position and place one or more of the buzzer prefab object")

Marker out of bounds

Press marker to place

Idle visible time

Marker occluded

Marker misbehaving (system confused by pattern; actually because of the participant's shirt)

SPIDER marker: Used properly (defined as "visible to position and place one or more of the spider prefab object")

Marker out of bounds

Confuse with buzzer marker for deletion

Confuse with AI marker

Press marker to place

Idle visible time

Marker occluded

Object Deletion (Markers & B-Button)

PLAYER marker: Used properly (defined as "visible to delete player object")

BUZZER marker: Used properly (defined as "visible to delete buzzer object")

SPIDER marker: Used properly (defined as "visible to delete spider object")

B BUTTON: Used correctly (defined as "pressing the button to delete SPIDER")

Used correctly (defined as "pre-delete click used to select SPIDER to be deleted" [discovered feature])

Used correctly (defined as "pressing the button to delete BUZZER")

Used correctly (defined as "pre-delete click used to select BUZZER to be deleted" [discovered feature])

Used correctly (defined as "pressing the button to delete PLAYER")

B button: Press too quickly, no effect

False release (did not actually reveal marker after press; nothing happens)

Move button too fast, accidental press

Pushed out of frame

Try to deselect (wrong button)

Try to delete while pressing C (while placing something else)

Try to delete with wrong marker

Try to delete without showing marker

Try to delete while marker is occluded

Try to delete without selecting object

Occluded (by AI marker or other), spider showing, accidental deletion of spider(s)

Accidental unaware deletion of object (best guess only)

Return from out of view or occl (x1), accidental deletion (AI)

UNRESPONSIVE

Adding AI Behaviour

Used properly (defined as "show to apply AI to one or more spider (also switch from select)")

Used properly (defined as "visible to select one spider object")

Marker out of bounds

Try to place spider (confuse with spider mesh marker)

Try to manip (confuse with manip)

Try to move object (confuse with move)

Try to add to buzzer

Try to use for multi-select

Try to use (for selection) without desel. first

Wave to apply / re-select

Idle visible time

Marker occluded

Marker unresponsive (AR system marker confusion) (ONE instance of allowing marker to seelect a buzzer (5s, p33)

Selection & Deselection

Used correctly (select or multi-select) ("moving/waving over objects to perform selection or deselect")

Marker out of bounds

Reselection / wave to desel

Confuse with camera marker

Idle visible time (feature idle)

Marker occluded (it counts even if it's just a fingertip!)

Panning Camera

Used correctly (defined as "moving the marker towards or relative to the center square; camera moving")

Marker occluded
Marker out of bounds
Trying to tilt marker to make it move
Trying to use marker like click+drag
Spotted by system (camera goes flying)
Spotted by system with select marker (thinks it is select, requires reset)
Idle visible time

Moving Objects

Used properly (defined as "visible to move selected objects from entry point to desired location")

Marker out of bounds
Confuse with select marker
Confuse with manipulate marker
Confuse with camera marker
Completely ambiguous movement (probably trying to rotate, which isn't possible)
Move multiple things by accident (forgot to desel.)
Remove from view without hiding
Moving objects into the abyss
Idle visible time
Marker occluded

Manipulating AI

Used properly (defined as "moving slider towards and within center square to adjust radius")

Marker out of bounds
Confuse with select marker
Confuse with move marker
Confuse with camera marker
Try to extend beyond limits / outside square
Try to scale radially (relative to center of object instead of as a slider)
Try to manip without any AI selected
Manipulate multiple bots (no desel.) (technically correct, but sometimes not)
Remove from slider without hiding (mess up desired radius)

Idle visible time

Marker occluded

Marker unresponsive (AR system marker confusion)

Other: C-Button

Used correctly/as intended (placement) (defined as "pressing the button to place something")

Used correctly (placement 2, inverse of above) (defined as "intentionally returned to view to activate")

Used correctly (deselect) (defined as "pressing the button to deselect")

Used correctly (moving) (defined as "moving the button")

Press too quickly, no effect

False release (button still occluded)

Moving button, accidental press

Move button too fast, lost tracking (extra placement)

Try to desel. without selection marker shown

Try to desel. while selection marker still over object

Try to add AI without marker shown

Try to add AI without spiders selected

Try to add AI without desel. (repeat AI or incorrect selection)

Try to delete (wrong button, extra placement)

Indecisive or impatient button press (trying to delete or desel. or something other)

AI marker causing trouble (correct but did not work for some reason)

Trying to confirm spider using AI marker

Trying to confirm AI using spider

Confirm SELECTION (not required, result in deselection)

Confirm MANIPULATION (not required)

Confirm MOVEMENT (not required)

Double click (extra placement): PLAYER

Double click (extra placement): BUZZER

Double click (extra placement): SPIDER

Placement marker moved or occluded while confirming (extra placement): PLAYER

Placement marker moved or occluded while confirming (extra placement): BUZZER

Placement marker moved or occluded while confirming (extra placement): SPIDER

Placement marker moved or occluded while confirming (NO placement): PLAYER

Placement marker moved or occluded while confirming (NO placement): BUZZER

Placement marker moved or occluded while confirming (NO placement): SPIDER

Placement marker moved or occluded while confirming (NO placement): AI

Selection marker moved or occluded while deselecting (NO desel.)

C button occluded by player marker, extra

C button occluded by buzzer marker, extra

C button occluded by spider marker, extra

C button occluded by select marker, accidental desel.

Partially out of bounds, invalid press

Return from out of view, accidental press

Out of area, unused (also applies to B)

Button genuinely unresponsive (unknown cause; known instances (2): buzzer confusion)

D.3.2 Unity

A total of 80 user actions were observed during Unity activities.

Object Placement

Used correctly (defined as "moving towards and selecting PLAYER prefab in assets, drag-n-drop into scene")

Used correctly (defined as "moving towards and selecting BUZZER prefab in assets, drag-n-drop into scene")

Used correctly (defined as "moving towards and selecting SPIDER prefab in assets, drag-n-drop into scene")

Select wrong prefab

Place wrong prefab

Select prefab but did not drag

Double click object name (trigger rename)

Prefab modification warning (dragged one prefab into another)

Return PLAYER prefab to assets panel (didn't want it or just changed mind)

Return BUZZER prefab to assets panel

Return SPIDER prefab to assets panel

Object Deletion

Used correctly (defined as "moving towards and using Edit > Delete to remove selected PLAYER from scene")

Used correctly (defined as "moving towards and using Edit > Delete to remove selected BUZZER from scene")

Used correctly (defined as "moving towards and using Edit > Delete to remove selected SPIDER from scene")

Duplicate spider

Use wrong menu item instead of delete (did not listen to instructions!)

Adding AI Behaviour

Used correctly (defined as "moving towards and selecting AI script in assets folder and drag-n-drop on top of spider")

Try to add to already-selected spider (did not deselect or select other first)

Add to spider multiple times

Add to buzzer (intuitive response)

Add to other wrong object (once or multiple)

Double click object name (trigger rename or editor open)

Select but no drag

Return AI script to assets panel (likely because of pre-selected spider, would be error anyway)

Indecisive drag?

Selection & Deselection

Used correctly (defined as "moving towards and single-clicking on a PLAYER object to select it")

Used correctly (defined as "moving towards and single-clicking on a BUZZER object to select it")

Used correctly (defined as "moving towards and single-clicking on a SPIDER object to select it")

Used correctly (defined as "moving towards and single-clicking on unused object to deselect target")

Used technically correctly (defined as "moving... to select a SPIDER that has been duplicated")

Trying to select target but hitting wrong object (not to desel.)

Accidental click n' drag, deselect

Multi-select (click n' drag; not allowed)

Using hierarchy

Try to deselect object by clicking on the object itself (does nothing)

Panning Camera

Used correctly (defined as "middle-clicking and dragging to change the position of the camera (pan)")

Zoom and correct (significant change only; not allowed)

Zoom and correct; minor and noticeable

Rotating the camera (not allowed)

Left click to pan

Mouse roll-over to other window OR idle

Moving Objects

Used correctly (defined as "moving towards and using red or blue arrow to change PLAYER pos")

Used correctly (defined as "moving towards and using red or blue arrow to change BUZZER pos")

Used correctly (defined as "moving towards and using red or blue arrow to change SPIDER pos")

Used technically correctly (defined as "moving... to change a SPIDER that has been duplicated")

Deliberate translate on Y (green arrow not allowed)

Accidental translate on Y (try to move red or blue, hit green instead; due to perspective!)

Translate with plane tool (not allowed)

Miss arrows, deselect instead

Rotate (not allowed)

Translate wrong object

Wrong mouse button, result in pan

Mouse roll-over to other window

Manipulating AI

Used correctly (defined as "moving towards and scroll-wheeling through properties bar on spider WITH AI to locate component")

Used correctly (defined as "moving towards and using the scroll bar to navigate properties bar on spider WITH AI")

Used correctly (defined as "moving towards and modifying radius value by clicking and dragging text")

Used technically correctly (defined as "moving... to modify a spider that has been duplicated")

Moving to and searching properties bar without selecting spider

Moving to and searching properties bar on spider WITHOUT AI; try to change un-attached behaviour

Moving to and searching properties bar on other object without AI

Time spent in text box wanting to type number

Released click and tried to adjust, nothing happened

Missed text, closed component box

Tried to scroll in properties bar but ended up zooming (and correcting) instead

Tried to middle-click to adjust value; results in pan (people familiar with Maya)

Copying and pasting AI component

Mouse roll-over to other window OR idle

Other: Folder Navigation

Used correctly/as intended (defined as "moving towards*** and clicking STUDY folder [top bar]")

Used correctly (defined as "moving towards and clicking PREFABS folder [main assets window]")

Used correctly (defined as "moving towards and clicking SCRIPTS folder [main assets window]")

Used correctly (defined as "moving towards and clicking the STUDY folder [sidebar]")

Used correctly (defined as "moving towards and clicking the PREFABS folder [sidebar]")

Used correctly (defined as "moving towards and clicking the SCRIPTS folder [sidebar]")

Navigating and time spent outside study folder (wrong folder, probably intervention req)

PREFABS: Clicking too slow and triggering folder rename

SCRIPTS Clicking too slow and triggering folder rename

PREFABS: Clicking on or accidentally navigating to the wrong folder

PREFABS: Single-click only resulting in not entering the folder; random movements follow

SCRIPTS Single-click only resulting in not entering the folder; random movements follow

SCRIPTS: Trying to drag entire folder into scene (happened with scripts folder only)

Appendix E

Chapter 6 Complementary Material

E.1 Additional Tables & Figures

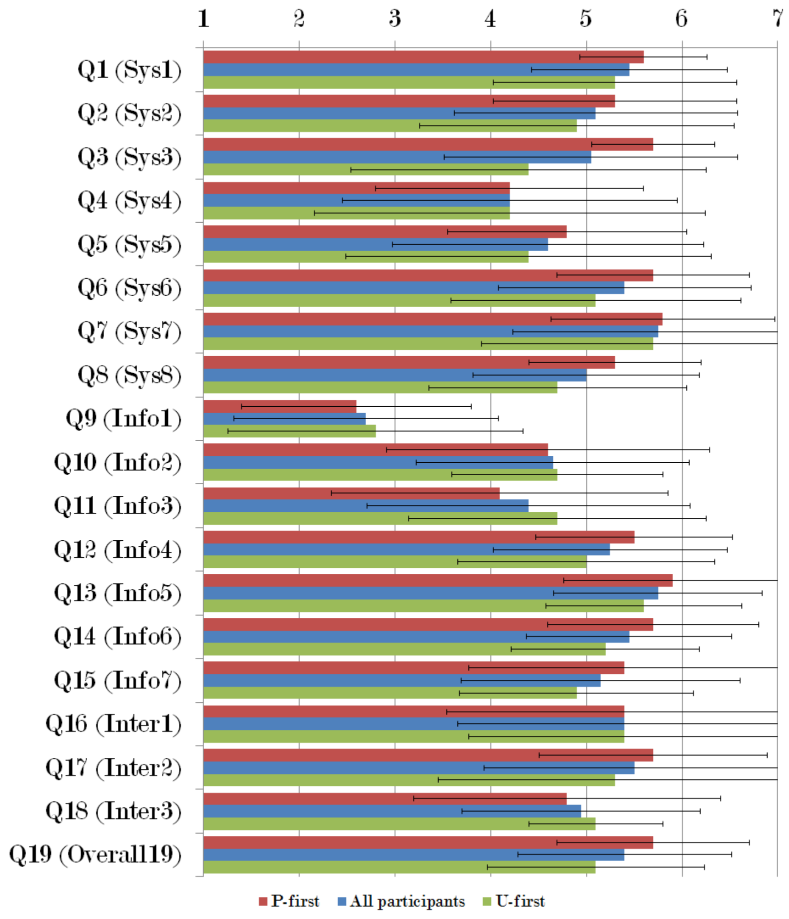
This section presents some additional figures that complement the materials presented and discussed in Chapter 6.

Performance: *N=20/20 vs. N=40/20*		
	KW test	**MWW test**
Total activity time	0.7569	0.7908
Construction time	0.4011	0.4263
Activity idle time	0.3533	0.3767
Previewing time	0.6269	0.6585
Total correct usage, % of constr. time	0.5656	0.5959
Total user error, % of constr. time	0.7906	0.8248

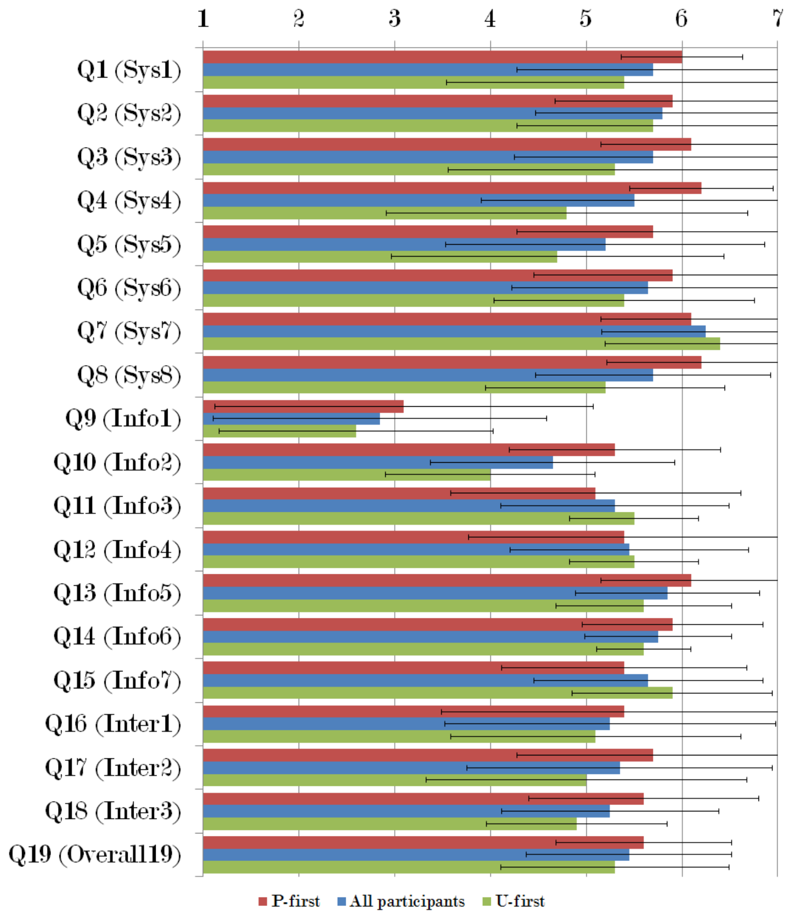
Table E.1: The p-values testing the effect of sample size on performance and scenes analyses. The Kruskal-Wallis test and Mann-Whitney-Wilcoxon test were used to see if changing the sample sizes had an effect on the significance levels within the two population divisions.

Scenes: *N=20/20 vs. N=50/50*		
	KW test	**MWW test**
Weighted Manhattan score	**0.03263**	**0.03505**

Table E.2: The p-values testing the effect of sample size on scenes analyses. The Kruskal-Wallis test and Mann-Whitney-Wilcoxon test were used to see if changing the sample sizes had an effect on the significance levels within the two population divisions.

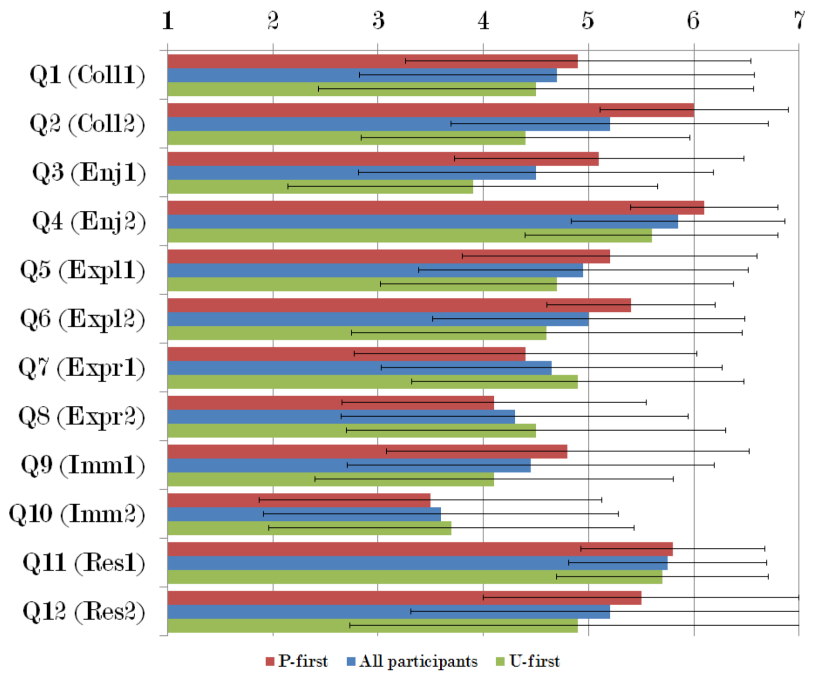


(a) Average CSUQ ratings per-question for PlayTIME

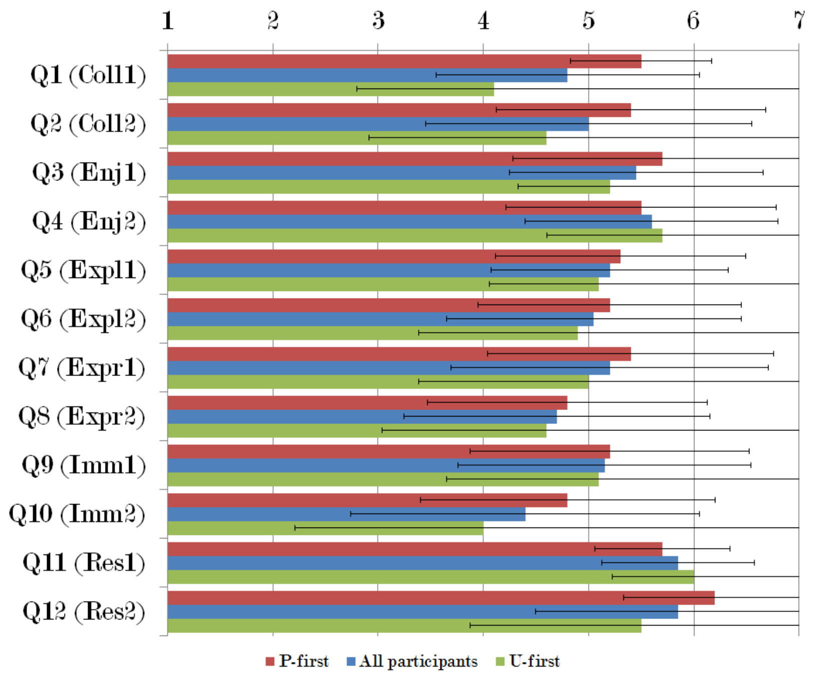


(b) Average CSUQ ratings per-question for Unity

Figure E.1: The average ratings for all CSUQ/PSSUQ questions for all participants between both groups. The error bars represent one standard deviation from the mean.

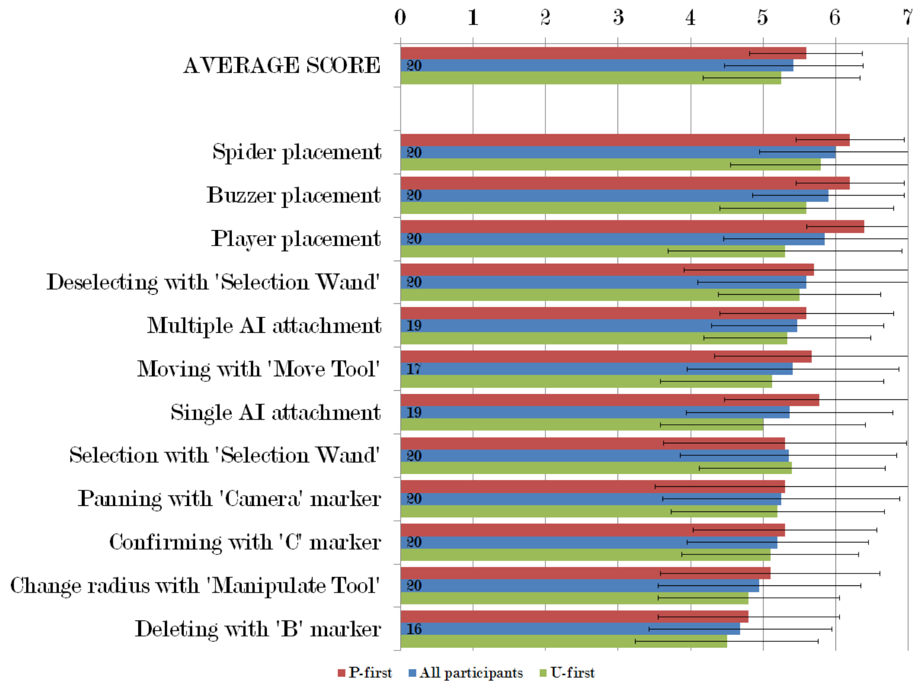


(a) Average CSI ratings per-question for PlayTIME activity

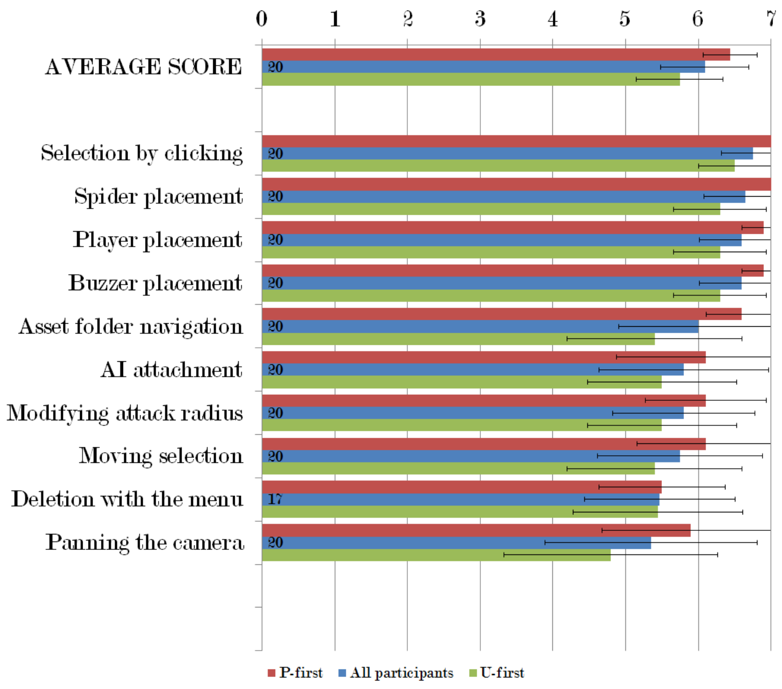


(b) Average CSI ratings per-question for Unity activity

Figure E.2: The average ratings for all CSI questions for all participants between both groups. The error bars represent one standard deviation from the mean. Note that these questions were presented to participants using the seven-point scale and were later corrected using a formula.



(a) Average ease of use ratings per-question for PlayTIME activity



(b) Average ease of use ratings per-question for Unity activity

Figure E.3: The average ratings for all ease of use questions for all participants. The error bars represent one standard deviation from the mean.

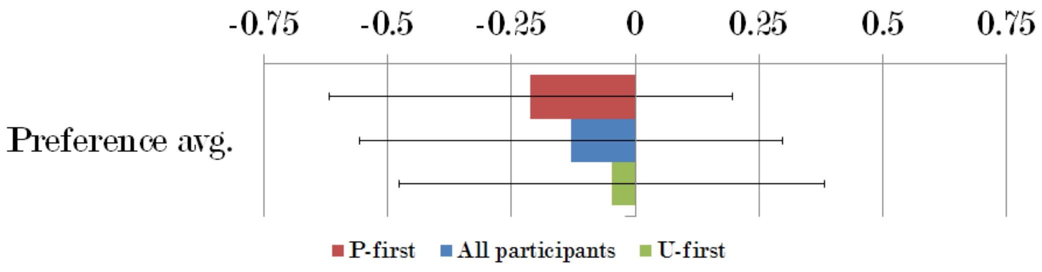


Figure E.4: The average overall preference between the activities. The error bars represent one standard deviation from the mean.

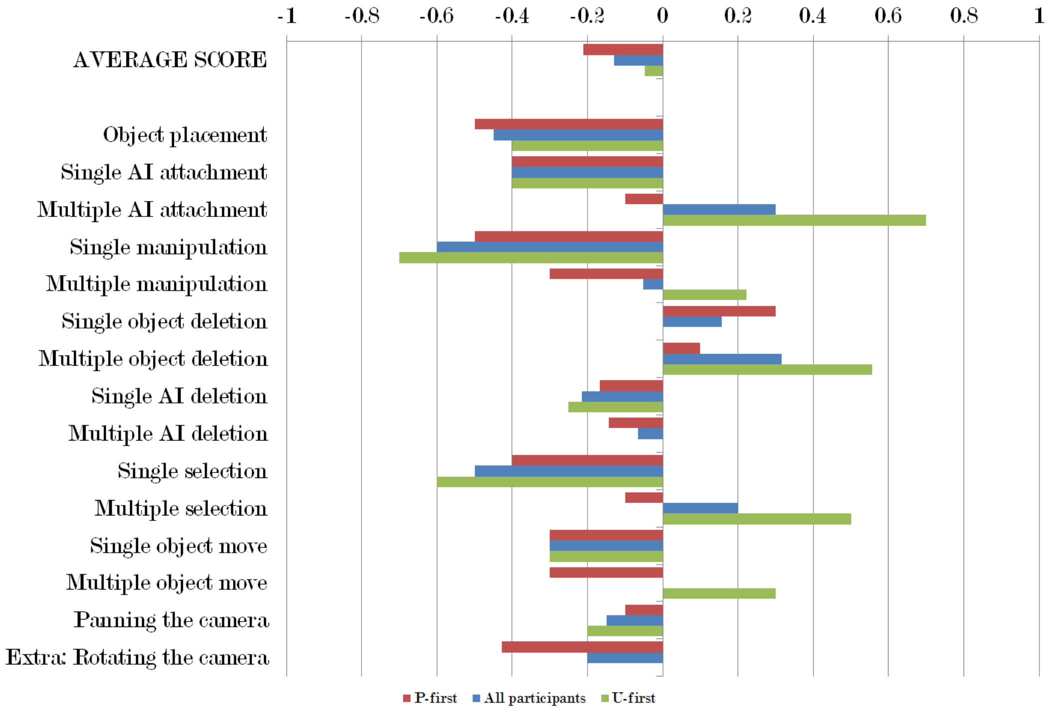
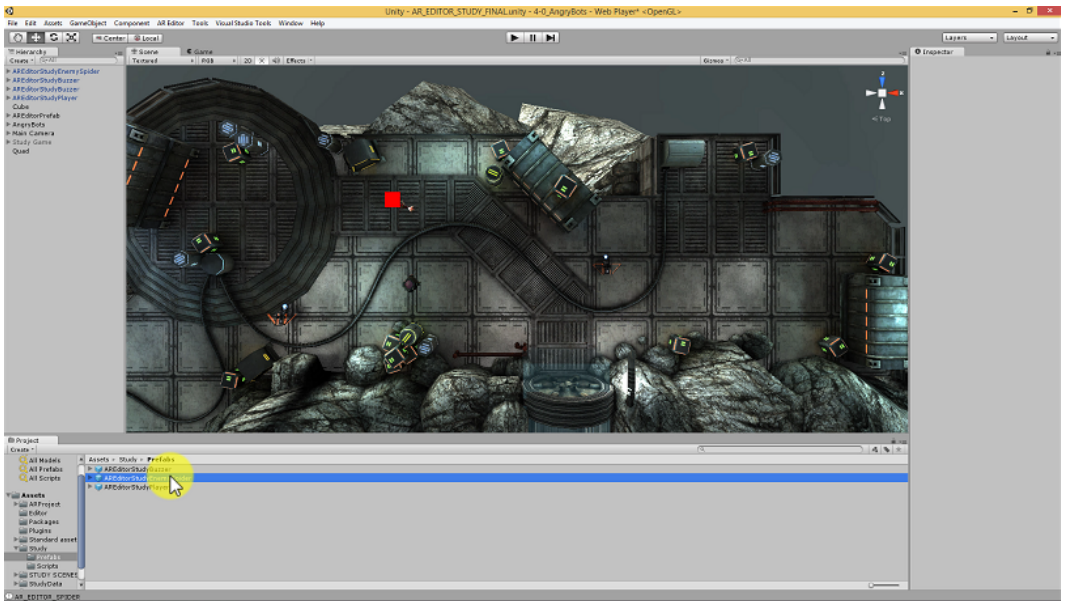


Figure E.5: The average preference ratings for the different tasks that were completed during the task. A positive rating leans towards PlayTIME and a negative rating leans towards the mouse.


(a) The object to be placed is selected in the assets window.

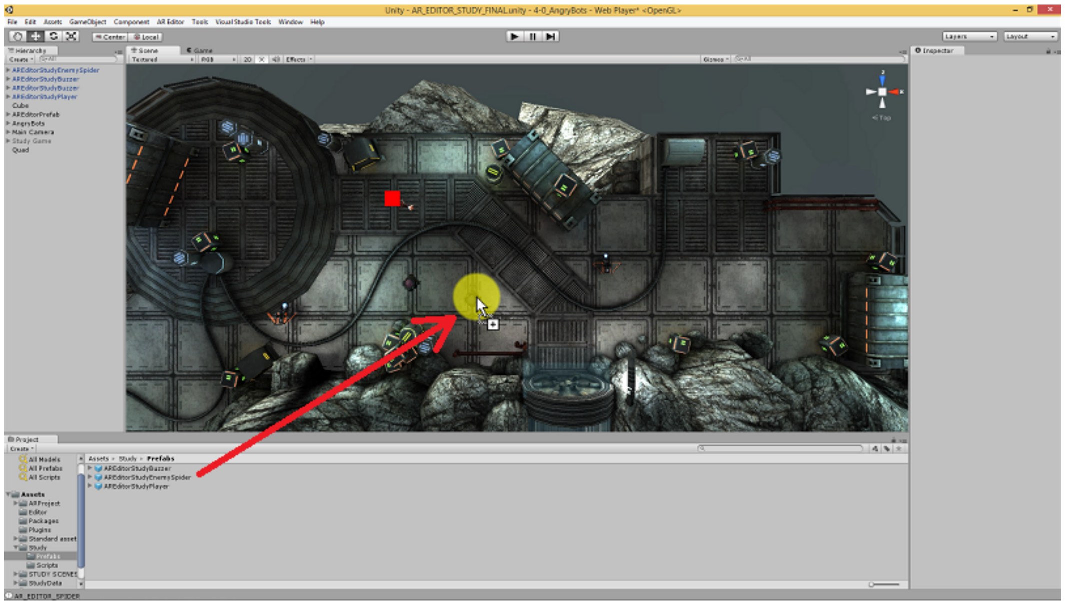
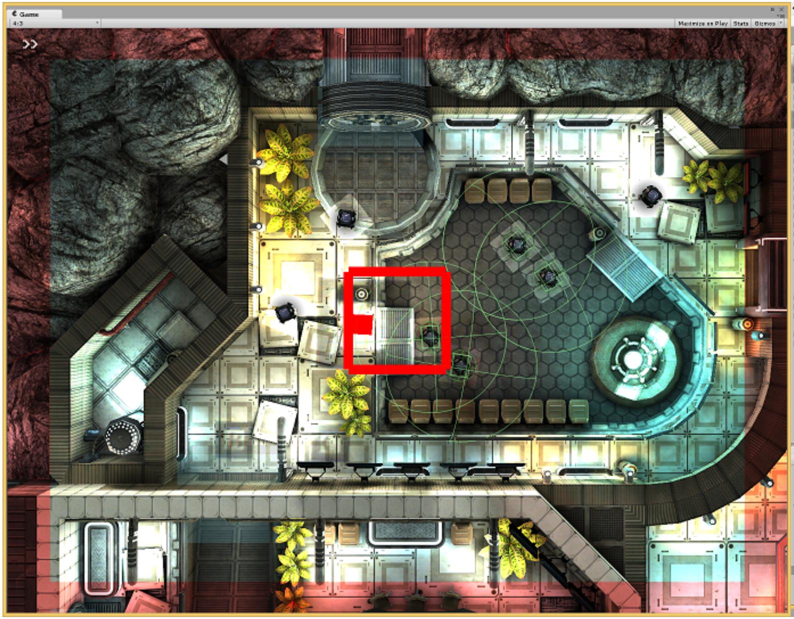
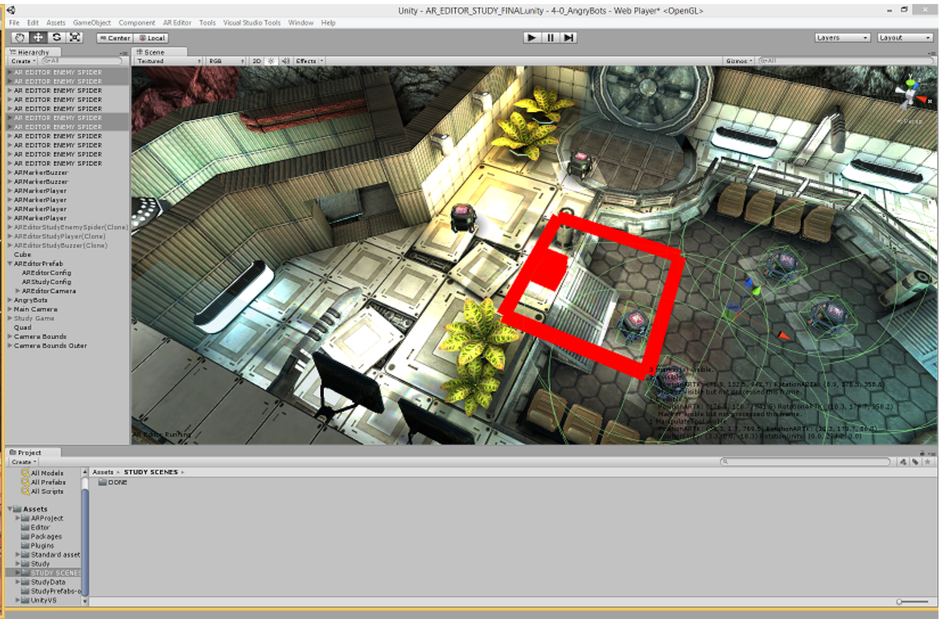

(b) The object is dragged from the assets window to the scene window and is dropped in place.

Figure E.6: The process of placing an object in Unity using the mouse. The cursor is highlighted by a yellow circle.

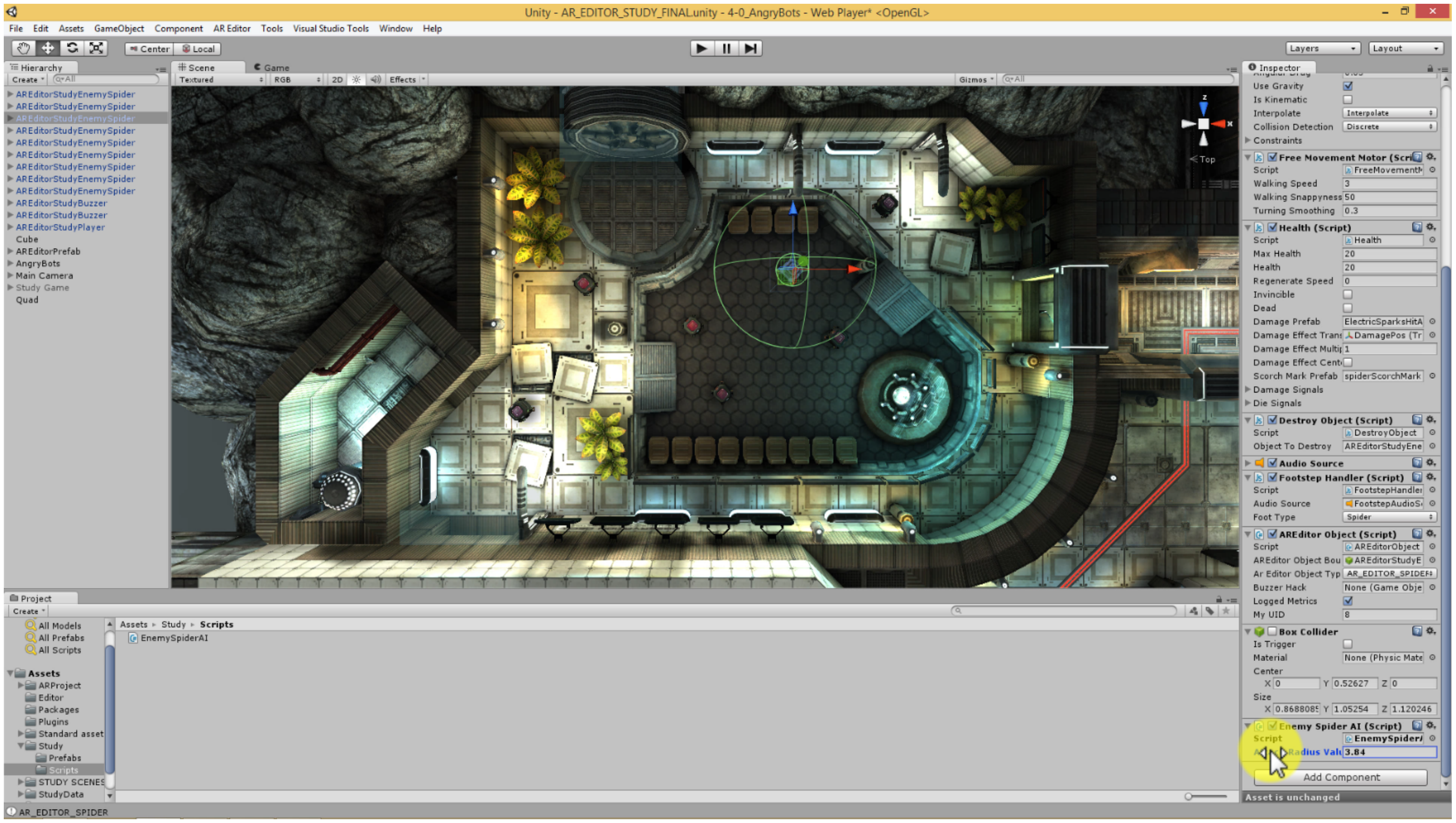


(a) The preview window used to display the PlayTIME view.



(b) The editor view, which follows the active marker.

Figure E.7: In this view, the manipulation marker is used to change the attack radius of multiple spiders.


Unity - AR_EDITOR_STUDY_FINAL.unity - 4-0_AngryBots - Web Player* <OpenGL>
File Edit Assets GameObject Component AR Editor Tools Visual Studio Tools Window Help
Center Local
Layers Layout
Hierarchy
Scene Game
Textured RGB 2D Effects
Gizmos
Inspector
Project
Create
Top
Add Component
Asset is unchanged
AR_EDITOR_SPIDER

AREditorStudyEnemySpider
AREditorStudyEnemySpider
AREditorStudyEnemySpider
AREditorStudyEnemySpider
AREditorStudyEnemySpider
AREditorStudyEnemySpider
AREditorStudyEnemySpider
AREditorStudyEnemySpider
AREditorStudyEnemySpider
AREditorStudyBuzzer
AREditorStudyBuzzer
AREditorStudyPlayer
Cube
AREditorPrefab
AngryBots
Main Camera
Study Game
Quad

All Models
All Prefabs
All Scripts
Assets
ARProject
Editor
Packages
Plugins
Standard asset
Study
Prefabs
Scripts
STUDY SCENES
StudyData

Assets ▸ Study ▸ Scripts
EnemySpiderAI

Use Gravity	✓
Is Kinematic	
Interpolate	Interpolate
Collision Detection	Discrete
Constraints	
Free Movement Motor (Script)	
Script	FreeMovementM
Walking Speed	3
Walking Snappyness	50
Turning Smoothing	0.3
Health (Script)	
Script	Health
Max Health	20
Health	20
Regenerate Speed	0
Invincible	
Dead	
Damage Prefab	ElectricSparksHitA
Damage Effect Trans	DamagePos (Tr
Damage Effect Multi	1
Damage Effect Cent	
Scorch Mark Prefab	spiderScorchMark
Damage Signals	
Die Signals	
Destroy Object (Script)	
Script	DestroyObject
Object To Destroy	AREditorStudyEne
Audio Source	
Footstep Handler (Script)	
Script	FootstepHandler
Audio Source	FootstepAudioS
Foot Type	Spider
AREditor Object (Script)	
Script	AREditorObject
AREditor Object Bou	AREditorStudyE
Ar Editor Object Typ	AR_EDITOR_SPIDER
Buzzer Hack	None (Game Obje
Logged Metrics	✓
My UID	8
Box Collider	
Is Trigger	
Material	None (Physic Mate
Center	X 0 Y 0.52627 Z 0
Size	X 0.8688085 Y 1.05254 Z 1.120246
Enemy Spider AI (Script)	
Script	EnemySpiderA
Radius Valu	3.84

Figure E.8: To manipulate AI using the mouse, the user must click on a single line of the properties bar on the right side. The cursor is highlighted by a yellow circle.

E.2 Factors Hindering Creativity

Task Description

Participant 45-P had made a choice that resulted in the exact number of enemies being placed, or something similar. In conversation, the observer said, "As a designer don't you feel that you are required to make creative decisions?" The participant's reply was along the lines of, "Yes, however I still need to please my employers." This was a golden quote because the participant directly said what many others may have only thought. People may have been worried about deviating from the task too much because they treated it as strict requirements, as if the activities were simulating a job or an interview perhaps. Some might have believed that it was a strictly professional situation, and therefore they had to have some sort of *justification*, or *persuasion* for their "employer" regarding the creative choices they *wanted* to make but did not feel they should.

It is quite possible that creativity was heavily affected by the task description itself: if participants believed they *had to follow instructions*, for whatever reason, then they would be less creative and line up with the task description. The task description implied that the tasks were not strict: "*Make the best prototype in a short amount of time!*" The task description *did not* say anywhere that they must follow it exactly. Even the following line about the zones was cleared up: "*Do not place anything outside the boundaries!*" This was *immediately* followed up with a rehearsed verbal explanation summarizing that this was only pertaining to the area visible within the map itself and they were free to use the areas not explicitly labelled within the map. Despite being fun, the task would have been better for creativity if it had been worded a little differently.

In-Game Bugs Hindering Creativity

There were a couple of a glitches in the level that had not been discovered early on and may have affected the Manhattan scores and possibly creativity results. The effects were not measured, but the bugs are worth noting.

The first bug is illustrated in Figure E.9. In the main room, one of the areas of interest was the back corner, since it was guarded by rails and would hide spiders well. However, due to the narrow opening between the rails and the walls in that area, the player would not be able to navigate to the corner to destroy any spiders placed there. The player would then shoot at the spider from a distance, and the spider would disappear through the floor as it approached, therefore rendering the level *not*

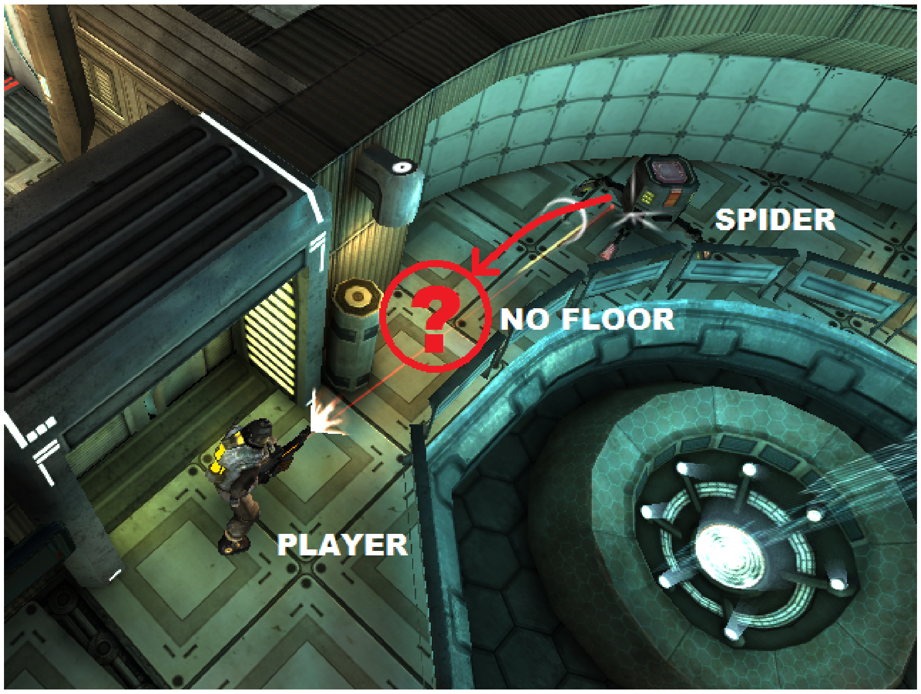


Figure E.9: An overview of the floor glitch. The player has shot the spider in the corner, triggering an attack. As the spider approaches the player, it disappears through the floor around the area of the question mark.

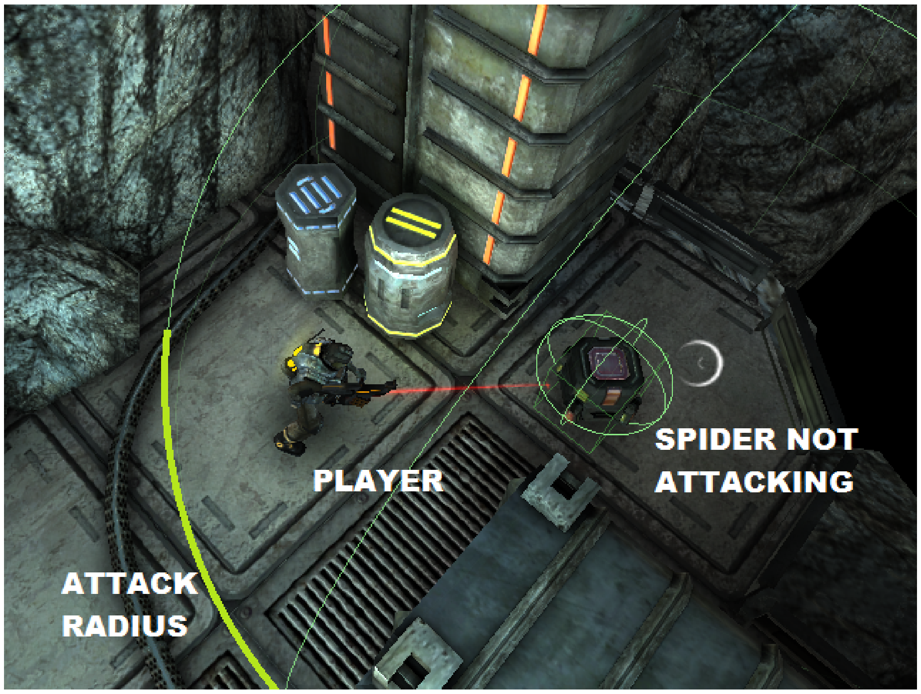


Figure E.10: An overview of the passive spider glitch. The larger sphere around the spider indicates that it does in fact have its AI behaviour attached; this is the attack radius. It is also clear that the player has crossed the threshold, yet the spider is not attacking.

winnable unless a replacement spider had been added somewhere else. Participants who discovered this glitch while previewing either left it alone citing that it was "just a prototype" or moved all enemies away from that corner. The bug is probably caused by a missing collider in that area, so that the spider does not actually make contact with what is supposed to be the floor.

The second bug is illustrated in Figure E.10. This one happened all over the place: spiders would simply not attack the player if they *did not have a direct line of sight when their attack zone was intruded.* A spider in this situation would remain passive until the player shot it or re-entered the attack radius. This but was not often fixed; users left the spiders where they were, having been told that it was a glitch and not their fault.

The bugs that occurred within the game itself hindered creativity because the users who decided to fix their levels to accommodate *somebody else's development errors* lost the opportunity to keep enemies in spots that were potentially unique, or would have increase their Manhattan score.