A Hybrid Digital/Physical Board Game JavaScript Library

by

Drew Tisdelle

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Interactive Media and Game Development

April ??, 2020

Approved by:

_____

Brian Moriarty, Thesis Advisor

_____

Gillian Smith, Committee

_____

Ralph Sutter, Committee

Abstract

Throughout the last decade handheld tablet devices have become a common part of the lives of many people. While these tablets have been widely used for games, they have been rarely used as a medium for board games in which the tablet is the game board and players interact with it using physical game tokens. The main topic of this thesis is to research past attempts at creating tablet-based board games and to use the knowledge gained from this research to develop a new, more accessible way to develop games for this medium. The Minotauros JavaScript Library, which was created to allow users to developers to make tablet-based board games for most tablets. The library's functionality was tested through users playing a simple game, *Roll*, that implemented the library. Through thorough testing, it was found that the library is fully functional and ready to be made publicly available. The research behind the functionality, testing, and results of the creation of this library are detailed in this thesis.

Acknowledgements

To be done last.

Table of Contents

Table of Figures

1. Introduction

This report describes Minotauros, a JavaScript library that allows developers to easily create tablet-based board games on a wide variety of standard Android or iOS tablets. It is hoped that the library will encourage the wider development of tablet-based board games, without the need for specialized hardware.

The Minotauros library started with the idea of creating a tablet-based board game that would demonstrate that specialized hardware is not needed to create hybrid physical/digital games. The prototype game, *The Minotaur's Labyrinth,* was put through a rigorous prototyping phase which helped to flesh out the details of what became the Minotauros library. The library was named after both the game concept as well as the Minotaur itself, the mythological blend of man and bull, just as the library is meant to help blend physical and digital games.

The Minotauros Library will be released publicly on the online software hosting platform GitHub. Once posted, the library will continue to be updated and improved based on user feedback.

2. Background Information

2.1. THE HISTORY OF BLENDING OF DIGITAL AND PHYSICAL MEDIA

In the past few decades, game creators have found ways to blend digital and physical media to create unique and interesting play experiences. What is meant by a blend of these types of media is that there have been creations in which users interact with something digital through physical means beyond using standard input, such as a mouse, keyboard, or video game controller. A simple example of this would be *Dance Dance Revolution* (Konami, 1998) arcade game in which users interact with the game by dancing on four directional pads built into the machine, thus resulting in the user being required to physically interact with the game in a unique way in order to digitally score points [1]. A blend of digital and physical media can also be in the form of a physical piece of media that is fully or partially controlled by something digital. An example of this would be the R2-D2 remote control robot (Sphero, 2017), which is a physical toy that a user can control the movements of digitally by using their phone. This section will go over some of these creations and the unique experiences that they have produced.

Figure 1. One Night Ultimate Werewolf and its companion app [Source].

The first creation that blends physical and digital media will be focused on is the use of smart phones to enhance physical card games. This has been seen in card games such as *Secret Hitler* (Blackbox, 2016) and *One Night Ultimate Werewolf* (Bezier Games, 2013) (Figure 1), hidden information games in which players try to deduce the identities of their fellow players [2]. Normally in these games, players would need to designate one person to sit out of the game to be the organizer of the game. This person would tell everyone when to perform certain actions during the setup of the game that need to be done for everyone to have the correct information revealed and correct information hidden. Due to this, a person in a group is always either excluded from a round of the game or is more likely to have their role revealed, leaving them at an unfair disadvantage throughout the game. However, both games decided to create a companion application for smart phones that would allow a person's phone to take the place of the organizer role. The application would be set to start and would play recordings that not only organized the game but included sound effects and jokes that would help players become more immersed in their roles within the game. By using this simple form of digital media, the problem of having the physical card game be unfair to a player each round has been negated and all players can enjoy the game equally and even have their experience enhanced by the application immersing them more within the game.

Figure 2. R2-D2 App Enabled Droid and the companion app that controls it [Source].

The second creation that will be focused on is the previously mentioned R2-D2 robot that was created by Sphero (Figure 2). This creation, as previously mentioned, is remotely controlled using a smart phone as the controller. To use the smart phone with the robot a user only needs to download the *Star Wars Droid App* (Sphero, 2017) [4]. From this app, the user can connect to the robot and tell it to perform various actions. This includes drawing out a digital path that the robot will then physically move itself along, using a digital joystick to manually move the robot around, and selecting from various movements and sounds for the robot to make. If this had been done using a simple physical controller, users would liken the experience more to having an RC car in the shape of a robot and instead of having an actual robot [5]. By allowing users to use something digital to control a physical toy, a unique experience has been made that causes users to feel as though they truly have a robotic companion that can program and control.

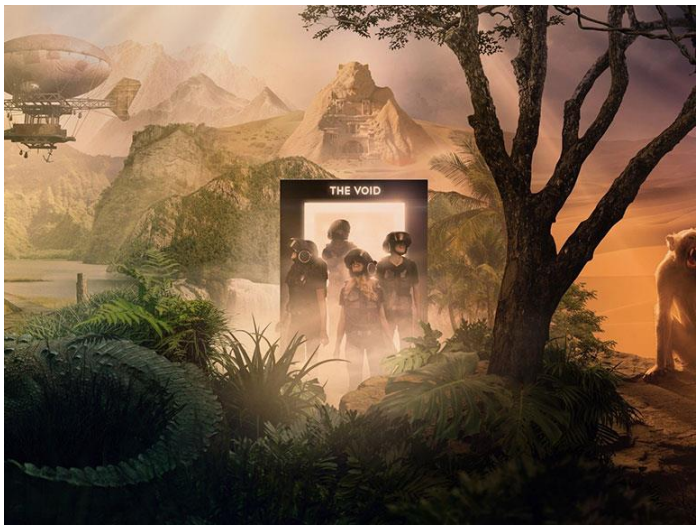


Figure 3. Jumanji VR: Reverse The Curse experience offered by The VOID [Source].

One last creation that will be focused on is the creation of mixed reality experiences, mainly those provided by *The VOID* (2015) (Figure 3) [7]. These mixed reality experiences, which are available only at select location, aim to make virtual reality as realistic as possible by including physical interaction with the virtual environment surrounding users. In these experiences, users enter a specialized area in a small group and are each given virtual reality headsets as well has special gear to wear on their body. This specialized gear simulates physical sensations, such as being hit by a shot from a Stormtrooper's blaster, using haptic feedback. The virtual reality headsets place the users in a world where they can look around and see their friends standing next to them as characters, such as Stormtroopers, and allows them to physically interact with the virtual environment by having the area the experience takes place in tailored to the virtual environment. For example, if a user were to see a lever in the virtual environment and were to reach out to grab it and pull it down, they would actually interact with a physical lever and cause the lever to be pulled down in both reality and virtual reality. By using this

mix of physical stimuli and virtual reality, *The VOID* can produce an experience that immerses players far more deeply into the virtual experience than average virtual reality headsets could.

As can be seen from the examples mentioned, the blend of digital and physical media has resulted in many unique, immersive experiences. By using digital components to handle the more fantastical parts of the experience and allowing for physical, in person components to handle the realistic, sensational parts of the experience, the creators of blends of digital and physical media are able to create unique, immersive experiences that couldn't have been accomplished through purely digital or purely physical media. As more of these experiences have been created, there is one type of creation that has struggled to find its footing to create this experience. This creation is the development of tablet-based board games.

2.2. ATTEMPTS AT TABLET-BASED BOARD GAMES

The blend of digital and physical media through the creation of tablet-based board games is a concept that has appeared within the last few years as tablets have become more popular and more technologically advanced. What is meant by a tablet-based board game is one in which a digital tablet is used as the game board. This tablet displays the board which responds and changes to physical game pieces touching the tablet screen. For example, if someone were playing a tablet-based board game of Monopoly, there would be a Monopoly board displayed on the screen. The players would then take their physical character token and move it along the board and when they land on a property they could purchase; the tablet would display a message asking if they would like to purchase the

property. A responsive game board like this helps to enhance the experience of players by removing the burden of needing to keep track of various aspects of games, allowing the board to change with the game, and can even produce music and sound effects that add to the atmosphere of the game. This is a concept that has been implemented in the past and we will now look at some of these creations.

Figure 4. The Lenovo Horizon 2 Multimode Table PC [Source].

The first of these creations that will be looked at the is Lenovo's *Horizon 2 Multimode Table PC* that was released at the beginning of 2015 (Figure 4.) [9]. This device was designed to function as both a tablet and as a personal computer that utilized the Windows 8.1 operating system. One of the main feature of this device that was heavily advertised was its inclusion of a variety of games for users to enjoy that incorporated specialized physical game accessories, such as air hockey strikers that players use to interact with a digital puck on the screen. The physical game accessories were designed specifically to work with only the *Horizon 2* and needed to be purchased separately from the device itself. While these games seemed fun and exciting, they suffered from the game accessories being very poorly implemented. In one review of the device, it was shown that the user needed to swipe the air hockey striker over the puck multiple times before the

device registered that the user was trying to hit the puck, resulting in the game being unplayable [10]. This is a problem that persisted with all the game accessories that were created for the device and thus resulted in the device never being fully utilized as a device for tablet-based board games. Another *Horizon Multimode Table PC* was never produced after the failure of the *Horizon 2* and so no further attempts at tablet-based board games were made by Lenovo.

Figure 5. The PlayTable using Nintendo's Amiibos as player tokens [Source].

The second device that this section will focus on is the tablet that was designed specifically for tablet-based board games, the *PlayTable* which was created by the startup of the same name [12]. This device, released at the end of 2019, was designed with RFID chip readers in the tablets screen that are used to detect and identify physical game pieces that contain RFID chips when they are placed on the screen (Figure 5.). Using this specially designed tablet and game pieces, *PlayTable* can offer a variety of games that can be played on it such as *Catan (*Catan Studio, 1995) and Checkers. The game boards

displayed on the tablet can change when specific pieces interact with it and the tablet is able to produce music and sounds to go with each game.

Figure 6. The PlayTable becomes inaccurate as more game pieces are used [Source].

While the tablet can accurately determine which pieces are placed on the screen and where they are placed, it does have many faults. The first of these is that the cheapest version of the tablet is $400 and does not include board games or the physical pieces needed to play them as they need to all be purchased separately, resulting in a rather large price tag that makes the tablet less accessible to potential players. The second fault is that the tablet, after having four or more pieces placed on the screen, becomes much less accurate at telling which pieces are placed on the screen and where they are placed, thus limiting what can be played on the tablet (Figure 6.) [14]. The third fault of the tablet is its small library of well-known games. While the tablet does have games such as *Catan, Checkers*, and *Poker*, it does not have much else available other than obscure, low quality games that are reminiscent of free to play games on smart phones. The final fault of the tablet is that the startup company itself has a very poor relationship with a decent portion

of its userbase. If one were to look at PlayTable's social media pages, nearly all the comments on their posts are complaints about how the company has been very unresponsive to users who have either had problems with their *PlayTable* tablet, or who have not even received their tablet despite having been one of the people who helped fund the *PlayTable* [15]. There have even been those who helped fund the *PlayTable* in 2017 who have not received their tablet while those who purchased one in 2019 have already received their tablets [16]. While it is still too early to see if the PlayTable will succeed or fail, I believe that they have many obstacles to overcome in order to make their tablet successful.

After looking at these recent inventions that attempted to create a medium for tablet-based board games, it was found that these attempts had something in common that severely limited what could be created with them. This common ground was that both companies made it so the technology needed to make tablet-based board games for their devices would work only with their devices. This meant that a person would have to, for example, buy a $400 *PlayTable* tablet in order to play tablet-based board games and then need to purchase individual games on it as well as the physical pieces, which is much more expensive than simply buying the game *Catan* for $40 on Amazon. While most video game consoles are roughly in this price range and follow the same model of needing to buy a console as well as games and extra controllers separately, the video games can only be played through the console while the board games can be played without a tablet for a much cheaper price. After doing additional research on other designs for tablet-based board games that were either not as popular or were only conceptual, it was found that they all fell under this category of creating a specialized tablet for their games. With there

already being over a billion tablet users around the world as of 2020, it is believed this idea of creating another tablet in this sea of tablets that was limited to only board games to be foolish [18]. Why would a person buy another tablet when they could simply use the one that they already have? That is why instead it was decided to introduce a new concept: creating the code and easy to produce physical pieces needed in order for developers to create tablet-based board games that are not limited to a single tablet and thus can be played on the tablets that people already own.

3. Developing a Proof of Concept

3.1. INTRODUCING A NEW APPROACH

When it was first decided to create the code and physical pieces necessary for developers to easily make tablet-based board games, it was first decided to research the capabilities of tablets that were released within the last five years to determine what the best approach would be to this problem. The reason for this short time span is that, as with most technology, users often only keep their current devices for a few short years before upgrading to a newer device and thus making it less likely that a user would still be using a tablet from more than five years ago. From here, research was done on tablets that had been created within the last five years, in the period of 2015 to 2020, and focused on what each tablet was capable of starting with the number of multi-touch points that the average tablet could handle. The number of multi-touch points that a tablet screen can handle is one of the most important factors to look at as it determines the number of physical game pieces that can be simultaneously sensed by the tablet.

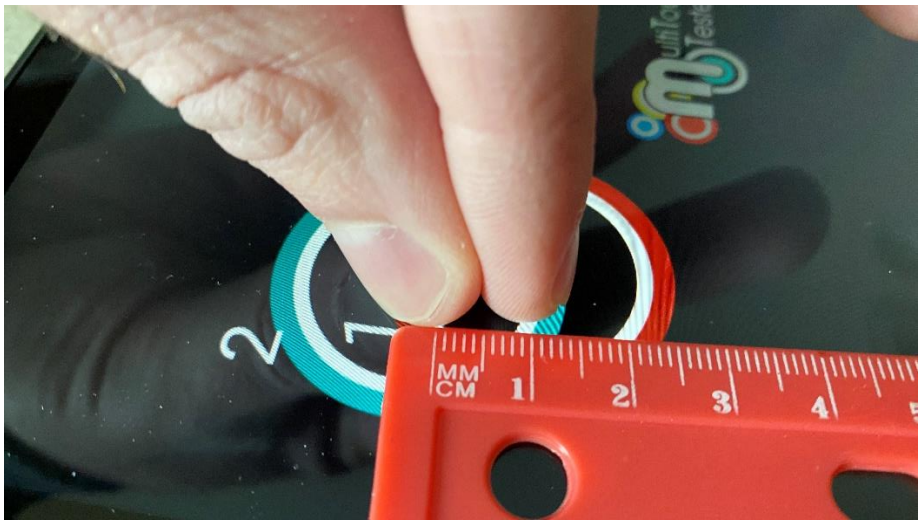


Figure 7. Average tablets need at least 1cm between separate touches [Source: Author].

While it was not possible to have access to many tablets, it was possible to research how the most popular tablet brands handled multi-touch by looking at the specifications of their hardware on their manufacturer's websites. The main tablet brands that were focused on were Apple's *iPads*, Microsoft's *Surface* tablets, and a variety of *Chromebooks* that were produced by Samsung [19]. After looking at the specifications for different versions of each of the tablets produced by the three brands, it was found that all of the tablets produced within the last five years were able to handle at least ten simultaneous touches, with ten often being the maximum number that could be handled by each tablet. In addition to obtaining this information through research, it was also possible to test one each of a *Surface Pro*, *iPad Pro*, and a *Chromebook*. Through extensive testing with each device, it was found that the tablet screens were able to accurately track each touch up to the maximum number of touches allowed by each tablet without any trouble. This, in contrast to the *PlayTable* tablet which became very inaccurate after just 4

simultaneous touches on the screen, proved to be very promising as it meant that it would be possible to track physical game pieces accurately on each screen without any trouble. Testing also allowed me to find the minimum distance needed between touch points for the touches to be registered as separate touches. After testing it on each tablet, it was found that the average minimum distance needed between touches before there was too much overlap for the screen to count them as separate touches was approximately 1cm (Figure 7.). This means that any physical game pieces that are placed on the screen need to be at least 1cm apart in order to both be detected at the same time.

Figure 8. The Chromebook doesn't register the 11th touch event [Source: Author].

Something that was found interesting when testing these three devices was how each handled a touch event that went over the maximum number of allowed touches. When trying after reaching the maximum number of touches allowed by each device, every additional touch that occurred was not registered by the tablets at all (Figure 8.). This meant that, if any additional touches were to occur outside of the expected maximum, they would not interfere with what was on the screen. For example. If there were to be ten game pieces on a screen that allowed a maximum of ten and a person accidentally placed an eleventh piece that wasn't supposed to be on the screen, the piece

would in no way disrupt the other pieces and thus there would not have to be countermeasures programmed for this scenario.

After determining the number of allowed touches and how far apart they needed to be to be individually sensed on the most common tablets, it was decided to next look at the best way to deliver the code needed to implement a tablet-based board game so that it would be compatible with most tablet devices. As each tablet that had been researched utilized completely different operating systems and therefore did not have one consistent format in which a program would run on each of them, it was found to be out of the question to create a code library designed to be compatible with each device. Therefore, it was decided to then look to what all the tablets had in common: web browsers.

Figure 9. Amazon's Kindle Paperwhite e-reader has a web browser [Source: Author].

While the operating system on each tablet was not consistent enough to create a library that would be compatible with each, it was found that reaching each tablet through web browsers would be the best solution as most if not all tablets now come with a web browser built in. Even Amazon's new *Kindle Paperwhite* e-reader, which is designed

only for reading and listening to books, has a web browser built in (Figure 9.) [20]. There are many in-browser games, such as *RuneScape* and *Club Penguin*, that can run on most tablets without being designed specifically for them due to the web browsers handling this aspect. While there are many different web browsers, for this thesis it was decided to focus only on the *Google Chrome* web browser in order to limit the scope of my testing to a reasonable level. The reason for choosing *Google Chrome* is because it is available on most tablets and has proven to be the most popular web browser in recent years with more than 50% of web users choosing to use *Google Chrome* [21].

With the focus now narrowed down to the *Google Chrome* web browser and the knowledge of how touch events are handled on the most popular modern tablets, it was time to start deciding how to implement the planned code library. Due to using *Google Chrome* as the web browser that the library will work with, it was decided that the code library would be done in the JavaScript programming language. The reason for this is that *Google Chrome*, as with most web browsers, fully support JavaScript and the language is heavily used for web development, making it so that the library will be more likely to be put to use as users will not have to familiarize themselves with a language that is less widely supported or less widely known. However, when planning out how this library would be implemented, it was realized that the needs of users of the library were not known and therefore it was not known how to structure the library. In order to determine what functionality the library would need to include it was decided to create a concept for a tablet-based board game that would help me determine what the library would need to do in order to support such a game. The result of this idea was the game prototype that would be called The Minotaur's Labyrinth.

3.2. CREATING A GAME CONCEPT

When first thinking about what I wanted my game concept to be, I decided to first think of some basic aspects to base the theme around. To do this, I looked back at the other blends of digital and physical media and how they had used the digital aspects to enhance the physical aspects. Looking at both *One Nigh Ultimate Werewolf*, I found that hidden information in a game is something that could be greatly enhanced through digital means. This is because there does not need to be a dedicated person to know the hidden information as a program could keep track of hidden information for the players, which helps to ensure that no one will cheat as well since the program would not be able to lie unlike a person. With it being decided that I would include a hidden information aspect into my game, I continued to look at how those experiences created by the blend of media were enhanced by the digital aspect. It's from here that I noted, as I had previously, that music and sound effects helped to greatly immerse users in their experience interacting with something physical. The main piece of media that I looked at during this time was Sphero's R2-D2 robot. This tiny little robot, while fun to see move around on its own and to control using a smart phone, was not very fun without sound. What really enhanced the experience was the little robot making constant beeps both on its own and in reaction to events that happened to it, making the user feel far more immersed in the experience of playing with the tiny robot. In addition, the *Star Wars* music that played from the smart phone while using the little robot also helps users to feel excited and helps to further immerse them in the experience. It was from here that I decided my concept game would be based around hidden information with a focus on sound, both of which can be done far more effectively in a tablet-based board game due to the tablet being able to produce a

variety of sounds in response to actions and being able to hide information from all players while ensuring that they do not cheat. By doing this I believe that the game concept will help to prove the worth of tablet-based board games as the attempts done by Lenovo and PlayTable have only served to convince users not to spend their money on this medium for games.

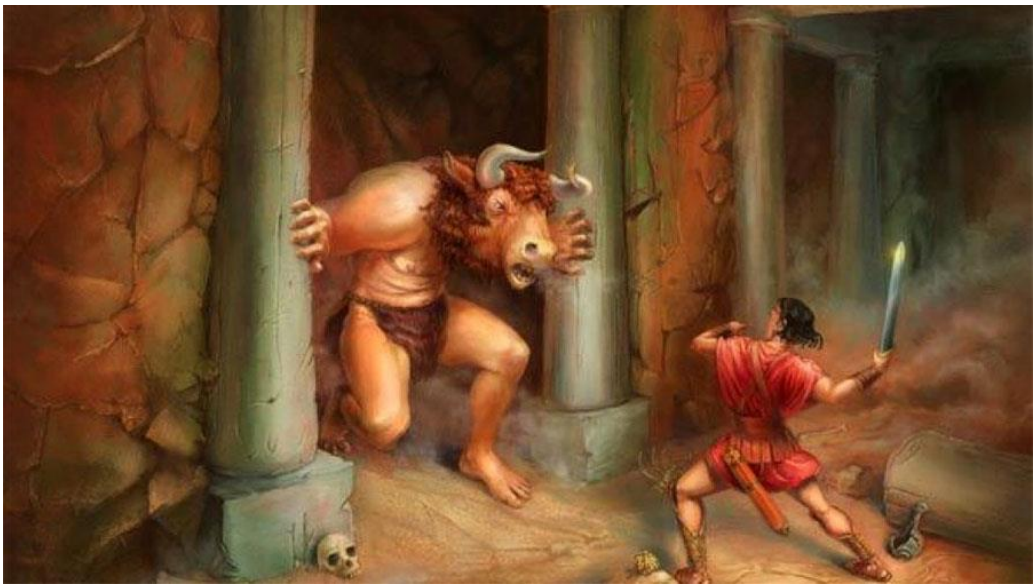


Figure 10. Theseus battles the Minotaur within the Labyrinth [Source].

With the base aspects of the test game settled on, it was time for me to pick a theme. During this time, I had been reading a book called *Circe*, a novel by Madeline Miller that focused on the witch Circe from Greek mythology [22]. In the novel the origins of the Minotaur, a create that is half man and half bull, are described as well as the famous Greek myth in which the Greek ruler Minos had commanded the master builder Daedalus to construct a great maze known as the Labyrinth in which to contain the Minotaur [23]. Each year sacrifices would be thrown into the maze in order to appease the Minotaur until one year when the Greek hero Theseus volunteered to enter the Labyrinth and slay the

monster. With the help of Minos's daughter, Ariadne, who had given Theseus a ball of thread to use to find his way through the maze, Theseus was able to slay the Minotaur as well as escape from the Labyrinth (Figure 10.). It was while reading this that I realized this myth would make for the perfect basis for a test game. The Labyrinth provides for information to be naturally hidden someone who were in such a maze would not be able to see more than the path ahead of them and the idea of the Minotaur roaming the maze compliments the game being sound based as players would be able to hear the roar of the Minotaur getting louder as they wander through the maze and grow closer to its location. With this idea in mind, I set about creating the most basic version of the game I could develop: a prototype done completely on paper.

3.3. CREATING A PAPER PROTOTYPE

In order to make sure the game concept was fully fleshed out, I believed it would be best to first make a paper prototype of the game. A paper prototype is a low fidelity version of a game where the basic rules and functionality are in place but the game itself is simulated completely using basic materials, often consisting mostly of paper [25]. For example, a paper prototype version of the game Checkers would be simply a piece of paper with the board roughly drawn on it while cut out bits of different colored paper are used as the Checker pieces. While the game would not be the most visually appealing, it would be functional enough to test the game and allow for changes to be made. The reason for this is because if high fidelity assets were used instead, such as creating a sturdy, visually appealing board and checker pieces, then if the rules were to change, it would be much harder to replace. For example, if the checker pieces were originally red and green but during testing it became clear that some users could not differentiate between the two

types of pieces due to colorblindness, it would be easy to toss out the low fidelity scraps of paper that were used for the pieces and it would take only a few minutes to make new ones. A higher fidelity one on the other hand would mean discarding pieces that took a longer amount of time and most likely cost more to make, putting the effort done to complete waste. By doing lower fidelity paper prototypes, the basic aspects of the game can be tested and easily changed quickly without much cost as well.

For the paper prototype for test game based around the Minotaur, which I decided to name The Minotaur's Labyrinth, I was presented with a great opportunity to both create and test the prototype. This occurred in Professor deWinter's Production Management course that I was a part of. My class, a total of six people, were given the task creating a game concept as well as creating a paper prototype of the game. I saw this as a perfect opportunity to rapidly prototype my game as I would have not only myself working on it, but five other graduate students as well. I suggested my idea to the class, and we agreed to use the game concept that I had created. Using this concept, we decided to build a simple, turn based paper prototype that consisted of one person playing as the Greek hero Theseus which was represented by a generic character token taken from the game *Sorry!*. Theseus would enter the Labyrinth, a maze that was generated using an online maze generator and printed on white paper, and try to get out the other end of the Labyrinth without running into the Minotaur, represented by another generic character token of a different color. A person was selected to control the Minotaur by simply having it move forward every time after Theseus moved and turn when it hit a wall. If Theseus made it to the other end of the maze without running into the Minotaur, he was victorious, otherwise, the game instantly ended as Theseus would be caught by the

Minotaur. With this simple, low fidelity prototype designed we were given another task: iteratively design and test the prototype using an Agile based approach.

3.4. ITERATIVE DESIGN AND TESTING

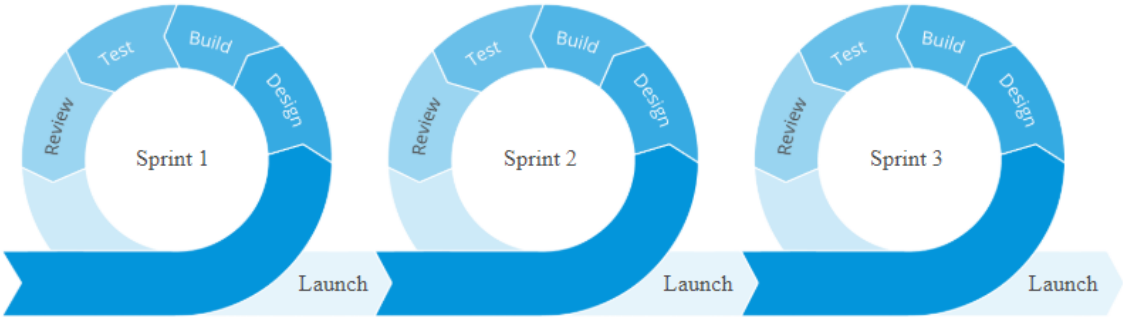


Figure 11. Agile is an iterative development process uses feedback loops [Source].

In order to iteratively design and test the paper prototype, we were given the task of taking an Agile based approach for this assignment. Agile is a software development approach that focuses around dividing up tasks into small, reasonable chunks that are placed up on whiteboard, often on notecards [26]. These smaller tasks are then put into different categories on the board and, while the categories differ from team to team, they often consist roughly of the task being not started, in-progress, blocked by another part of the project, currently being tested, and complete. Each of these tasks is often labeled with a rough estimate of how long they will take to complete, making it easier to efficiently use time. In addition, these tasks are often done in what is known as sprints. These sprints are short periods of time, often two weeks, in which all the tasks on the board are to be completed. Once the two weeks are up, a meeting is held to go over what went well during this sprint and what did not as well as to figure out what needs be done during the next sprint. Over the two weeks, short, daily meetings are held as well where team members

can bring up any troubles or concerns that have come up and, after the meeting is done, can receive help from their fellow teammates. This form of development is meant to efficiently divide up and organize work in a manageable fashion as well as make sure the development is iterative by moving tasks into the testing phase of development and back into the in-progress phase of development. This process takes place repeatedly until the task has been sufficiently tested and can then be marked as completed with iterative design such as this even taking place over several sprints (Figure 11.).

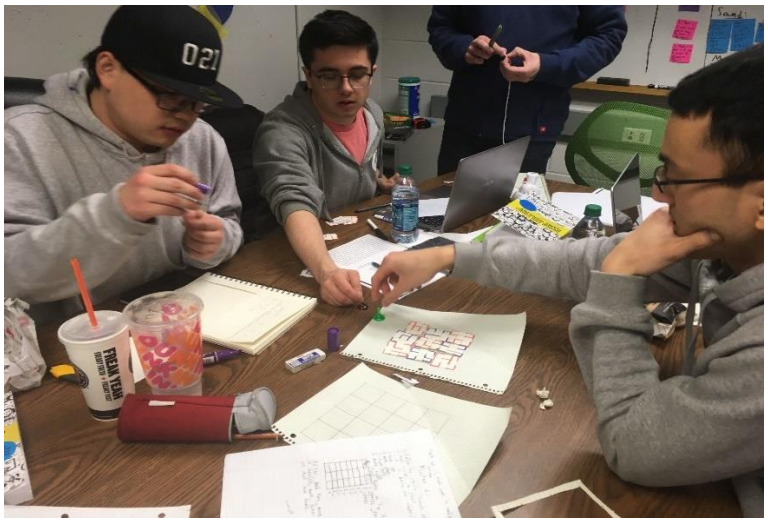


Figure 12. Meeting to test the prototype game and divide up tasks [Source: Author].

With our first paper prototype completed and this process in mind, we started with testing the game. As we played, we realized already that the game had many problems. As the maze had only one path that would lead to the exit, Theseus would rarely win if the Minotaur was along the path. In contrast, if the Minotaur was not along the path when the game started, Theseus was almost always guaranteed to win without any real effort. The game also didn't include the two key concepts that I intended to be the focus of the game, hidden information and sound. The player could always see the entirety

of the maze as well as the location of the Minotaur and there were no sound effects to go along with the game. With these flaws exposed in our initial prototype, we got to work on having our first meeting to decide how to break down the tasks needed to redesign the game from scratch using the Agile development method (Figure 12.).

While sprints normally last over a period of two weeks, we were given only 30 minutes to go through a rapid version of the process, quickly dividing up the tasks and breaking off into groups for roughly 20 minutes before coming back together to test the game once again. The groups we broke up into consisted of three groups of two people, with one group focusing on designing the rules of the game, one group focused on designing the mazes for the game, and one group focused on figuring out how to incorporate the hidden information and sounds into the game. To make the tasks easier to divide up, when we put our tasks up on our whiteboard we also labelled which group we thought the task would best be handled by. With our groups created and work divided up, we then got started on our first sprint. While there were many different versions of the game that were created through this process as we had many sprints, I will focus on the two main ones that had the most lasting impact on the final version of the prototype.

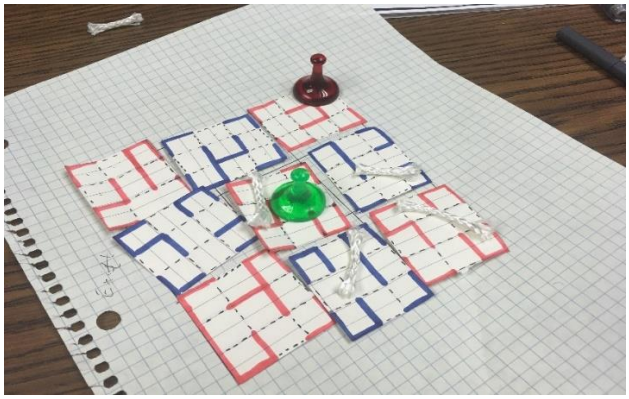


Figure 13. The first successful prototype still had many flaws. [Source: Author]

After a couple of iterations of development, the paper prototype we had produced was looking much different than when we had started (Figure 13.). In order to make the mazes fairer to both the Minotaur and the player, the maze was divided up into nine square sections with there being 27 total variations of these nine sections. Each section contained nine total spaces for Theseus and the Minotaur to move along and each section was designed in a way that they could connect to any other section that they were placed next to, allowing for a wide variety of mazes for players to work their way through. This allowed for multiple paths to reach the exit of the maze, making the game much more balanced in that it was never guaranteed who would win the game. In this iteration, the rules for the Minotaur were also redesigned in that, instead of moving forward and turning at random when running into a wall, the Minotaur would always try to take the shortest path to the player. We also decided during this iteration to play light background music that we thought would be fitting for the game. However, despite the progress made during this iteration there were still many flaws. The player could still see where the Minotaur was and the entirety of the maze, meaning that the hidden information aspect of the game was still not incorporated. In addition, the game could easily end in a stalemate as once the Minotaur was in one space of the player, the player need only to move a space away, preventing the Minotaur from ever catching them but also preventing themselves from ever advancing towards the exit of the maze. With the flaws in our design once again exposed, we continued onwards through the iterative design process.

Once we had gone through a few more iterations of development, we reached the final version of the paper prototype. This prototype, unlike the previous ones, incorporated hidden information into the game. This was done by flipping the maze sections over so

that the player could not see the entirety of the maze. Once the player moved to a different section of the maze, that section was flipped over so they could see it and the one they were on was flipped back down, preventing the player from seeing more than one section at time. The Minotaur's location was also now more obscured for the player, as not only could the player not tell what paths the Minotaur could take to reach them but the Minotaur's exact location within each section was not known. All the player would know was roughly how far the Minotaur was from them based on which section it was in. For example, if the Minotaur were in the section next to the player it could potentially only be one space away from entering the same section as the player or could be on the other end of the section, so the player would be unsure of their safety while navigating the maze. This helped to greatly enhance the experience of playing the game as the hidden information created a sense of fear and urgency as the Minotaur closed in on the player.

The final version of the paper prototype of Minotaur's Labyrinth was much more fleshed out than the initial concept and was well on its way to be a great game that would fit well as a tablet-based board game. While this concept was well developed, after the final version was created the development for the game switched back to being done by only myself as the assignment for the course was completed. Due to this, creating the entire game alone was far outside of the scope of what I could due within the time I had and thus the game never advanced much further than the paper prototype stage. While the game did not make it to a final version, it heavily influenced my thought process throughout developing the JavaScript library that I would name Minotauros after the influence that the prototype for Minotaur's Labyrinth had on its creation. This game was what helped me to identify what users of my library would need in order to make their

games easily and thus helped me develop the structure of the library. The needs of users are covered in detail in the next section of this paper. The game also helped me to create an example of a concept that could be best implemented through the medium of a tablet-based board game and thus helped to serve as an example of the value of tablet-based board games. I hope to one day fully implement the Minotaur's Labyrinth using the Minotauros JavaScript Library so that the game concept can be fully realized through the medium that it was intended for and using the library that it had helped inspire.

4. Designing the Library

4.1. DESIGNING GAME TOKENS

With the thorough testing of The Minotaur's Labyrinth having given me a number of ideas on the needs of users and the research and testing of various tablets providing me with the knowledge of the limitations I must work within, I set to work on designing game tokens that could interact with most touch screens. As seen by past examples, this is a task that has not been easy to accomplish. For *PlayTable*, they had designed a tablet that had RFID sensors in it that were able to detect figures with RFID chips inside, but most tablets do not have RFID sensors built into their screens. In addition, once four or more RFID chips were placed on the tablet the accuracy of the tablet's sensors decreased dramatically. In the case of the *Lenovo Horizon 2 Multimode Table PC,* the game tokens were partially made out of conductive materials so that the tablet could detect them, but the choice of material and padding used on the pieces to prevent scratches on the screen as well as a

poorly developed touch sensitivity prevented the pieces from being easily detected and made for a frustrating experience.

While the challenge has yet to be overcome for this medium, I believe that Lenovo's approach was a step in the right direction. By finding a way to create game pieces from conductive materials, users could effectively use these pieces on most tablets as modern tablets use capacitive screens. These screens use the conductivity of a person's finger to determine where a touch event is occurring by sensing the change in voltage caused by the person's finger coming into contact with the screen [28]. This can also be done by a person holding an object made from conductive material and touching the capacitive screen, with the change in voltage being sensed at the points where the object touches the screen and thus allowing it to detect the position of the object. As soon as a person stops touching the object however, the touch screen will no longer be able to sense where the object is. Keeping this in mind I decided to improve upon the approach that Lenovo had taken and started testing conductive materials that could be used to create the game pieces.

Figure 14. The pewter figure is easily sensed while the steel one isn't [Source: Author].

When first looking at different materials to make the game pieces out of, I first decided to do some tests using metallic figures that were already available to me. The reason for this was both to test if the materials they were made from were conductive and to see if users could potentially use game pieces that they already had instead of needing to buy or create specialized ones. Using an application on the Google Play Store called *MultiTouch Tester,* an app that indicates all touch events sensed on a screen at a given time using small circles, I was able to easily test various types of materials and how consistently they were sensed on the screen [29]. One of the first things I tried was a metal X-Wing from Star Wars that was created by the company Metal Earth [30]. The X-Wing was made of a series of thin steel sheets, bent and interwoven in order to hold together their shape. Unfortunately, the touch screen had a very difficult time detecting the figure and could not detect it most of the time. I tried taking one of the individual steel plates that made it up and seeing if the screen could more accurately detect it and, while it was bit easier for it to be tracked, it was nowhere near accurate enough to be considered a viable material to use. I continued to try other game pieces made of various metals but to no avail until I decided to turn to an old Star Wars themed Monopoly set that I had stored away [31]. The character tokens in this set were made of pewter, a cheap metal allow with conductive properties. From this set I took the General Grievous pewter figure and tried it with the *MultiTouch Tester* (Figure 14.). To my surprise, it easily detected the figure and was able to track it as I moved it around the screen if I kept my fingers touching the figure. At last I had found the material that could be used to make game tokens for tablet-based board games, and it was a material that was relatively inexpensive to make tokens

out of. With my material now found I shifted my focus to another problem: how would I identify individual tokens on the board.

Using the method of having the capacitive screens detect pewter game tokens allows the token to be accurately detected but comes with a major challenge overcome. While the capacitive screens are great at detecting where a touch event is occurring and accurately keeping track of the event, they are unable to identify what triggered the event. This means that a touch event caused by a finger or a touch event caused by a General Grievous pewter figure could not be differentiated at all and thus a game would treat them as if they were the same. This is not a problem in some cases, such as with the virtual air hockey game that the *Horizon 2* tablet had since the game only needed to know that the puck was being hit, it didn't matter what was hitting it. However, for games such as Monopoly, it does matter which player token is placed where on the board so that the game can know when each player can buy a property, needs to pay taxes, or passes Go. My problems came to an end when I found the research of another person who had been trying to solve the same problem.

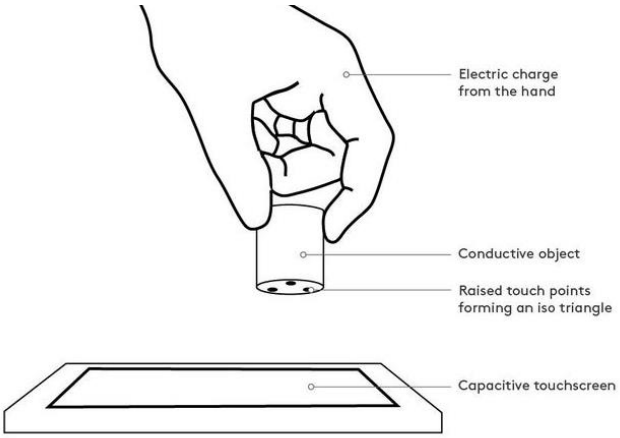


Figure 15. Paul Ferragut's diagram showing how his C++ application functions [Source].

On the website *Instructables*, a website filed with a variety of do it yourself guides, I found a guide made by a man named Paul Ferragut that was titled *Object Interaction With Touchscreens* [32]. This guide outlines how to create objects that can be sensed and identified by capacitive touchscreens by using conductive paint to make three dots that form different angled triangles on the part of the object that is placed on the screen (Figure 15.). These three dots would then be the points that are sensed by the screen when the user touches the conductive objective that they are drawn on and uses the unique angles of the triangle formed by the three dots to identify which piece was placed down. If the object is not conductive, one could instead make a trail of conductive paint that leads up along the object and when the user touches this trail, the electrical charge from the user's fingers will flow through it to the three dots below. This was done using a C++ application that Paul Ferragut had written called *oscTouchObject* and posted on GitHub with open access for anyone to use [33]. The program would quickly detect the three points that were placed down, calculate the angles that made up the triangle, and determine the piece that was placed and action that should be taken all within a split second of the user placing the piece on the tablet screen. After wondering how I could overcome such a difficult task, I was astonished to find that not only had this person already created the code needed to implement the technology, but also that it had not been used for anything more than simple artistic projects. Using the idea created by Paul Ferragut, the research I had done, and the testing of the Minotaur's Labyrinth, I would now be able to create the JavaScript library needed to allow for the easy development of tablet-based board games.

4.2. NEEDS OF THE USERS

As mentioned previously, while the Minotaur's Labyrinth game never moved beyond the paper prototype stage, it did heavily influence the design of the Minotauros JavaScript Library by helping me to identify the needs of potential users. The first need that was identified through this process is the need to move game tokens both by picking up them up and placing them in a new position as well as sliding them across the screen. This was discovered by observing the different playstyles that were used throughout testing the paper prototype and realizing that it would be difficult to enforce just one type of movement. Therefore, the library would need to be capable of allowing users to easily detect pieces that are both slid across the screen and moved by being picked up and placed down.

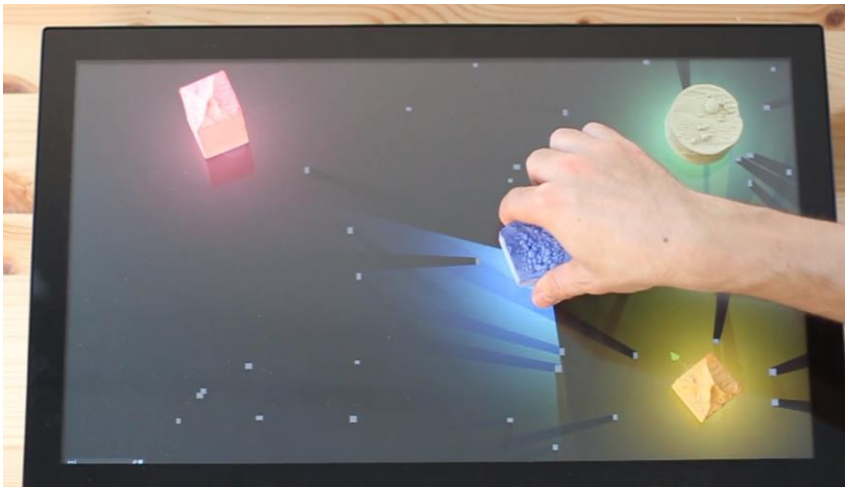


Figure 16. Paul Ferragut's C++ application provides immediate visual feedback [Source].

The second need for potential users that I discovered was the need to easily trigger events within the game. This was found both through testing the paper prototype game and through looking at the demonstration videos created by Paul Ferragut to show users

the capabilities of his C++ application. While playtesting the Minotaur's Labyrinth, the iterations of the game that were easiest to test and develop were those that had an easily developed feedback system. For example, in the final version of the prototype when the player moved to new position the person controlling the Minotaur immediately took note of the players new location, announced they were determining the optimal path to reach the player, and then announced that the Minotaur had moved one location closer to the player. With this feedback system setup beforehand, we were able to easily adjust it for testing to not only gather data for testing but to give feedback to players as well. The C++ application helped to confirm that a good feedback system needed to be in place not only to give feedback to the developer and users, but to make it easier to trigger events in a timely manner (Figure 16.). This is shown through the demonstrations done by Paul Ferragut in which he places conductive pieces he created on a tablet screen and his application immediately reacts by detecting the piece, determining which piece it is, and lighting up the area surrounding the piece based on the color of the piece it is. This immediate triggering of events was due to his system being setup so that users could call their own events to be easily triggered and so that immediate feedback could be provided to users.

The last major need of potential users of the library is easily keeping track of game tokens. This needs to be done in two forms: tracking the identity of each token and tracking the position of each identified token. The first is needed so that users of the system can easily enter the information needed to identify a token into the system with ease so that they do not have to worry about getting their program to the detect the token and only need to worry about the functionality they want to happen once it is detected.

The second is needed in order to know where the tokens are placed on the tablet screen at any given time. This is of great importance since the tablet screens themselves will not always know where the token is currently located due to the tokens being impossible to sense when a person's fingers are not currently in contact with them. This makes it very difficult to differentiate between when a piece is still on the screen and the person has stopped touching it and when the piece has been lifted off the screen, making it imperative that the library include a way to handle this for its users. With the needs of potential users in mind and a way to identify game tokens discovered, it was finally time to put all my research to use to create the Minotauros JavaScript Library.

4.3. HOW IT WORKS

The way in which I designed the Minotauros JavaScript Library was so that users could easily import the library and use it to call various functions that support the game they are developing to give it the capability to be a tablet-based board game. For example, a user could import the library as "Minotauros" and use it to call all the functions they require while developing their game. The first piece of functionality that I created was the ability to store game tokens that are placed on the tablet screen as all the rest of the functionality will not matter if the information for the tokens is not stored. To do this, I implemented a function called *registerToken* that requires the user to enter in all three angles of the triangle at the bottom of the token as well as a name to give the token. The token's information is then stored in an array of tokens which can be searched through using a function called *getTokens*, which returns the array of registered tokens. I have also included a function called *registerTokenTouch*, which, once called, waits for three simultaneous touch events to occur. Once three touches are detected, the function uses

the position of all three touches to calculate the three angles of the triangle that is formed by the three touch points. With these angles calculated, the function then calls the *registerToken* function and passes into it the three angles as well as the name "Token 1", with the number within the name increasing each time the *registerTokenTouch* function is called. This way, users can choose to allow players to register tokens at the start of a game so that they can use any token they want to play the game. Using these three functions, users can easily store and retrieve token information.

The second piece of functionality I added to the library was the ability to identify individual game pieces and note their position on the tablet screen. This is done through the *identifyToken* and *updateTokenPosition* functions. To use the *identifyToken* function, a user needs to add an event listener onto the section of the screen where they would like tokens to be detected. This can range from being either the whole screen if the user would like the token to be detected anywhere on the screen to a single square space on virtual gameboard. This event listener, or listeners, would be set to listen for a "touchstart" event that is triggered when a touch is detected on the screen. From here, the user would set their own function to run when a touch is detected. At the beginning of this function is where they would place the *identifyToken* function. This function takes in the touch event that occurred and checks to see how many simultaneous touches are currently occurring. If the number is three, then the function takes the positions of the three touch points and, as with the *registerTokenTouch* function, calculates the angles of the triangle formed by connecting the three points. These angles are then compared to the tokens stored in the array of registered tokens and if the angles match within an array margin of plus or minus five degrees, the function will return the registered token allowing the user to see which

token triggered the event and continue their function based on this knowledge. The function will also call the *updateTokenPosition* location which calculates the center point of the triangle on the screen and updates that position as part of the token's stored identity. If the token does not have a position yet because it was registered using the *registerToken* function, the location will initially be at coordinate (-1, -1) and will then be updated to the coordinates of where it was first detected. If the token has not yet been registered, then the *identifyToken* function returns a token named "error" with the angles all set at zero in order to indicate that the system could not identify the token and the user can continue their function with this knowledge.

The third and final piece of functionality that I added was detecting when a piece is no longer sensed on the tablet screen. This is done by adding another event listener in the same place as the event listener that checks for when a "touchstart" event occurs. This event lister instead looks for a "touchend" event that is triggered whenever a touch is no longer detected on the screen. From here, the user puts their own function that will trigger whenever a "touchend" event occurs and will run the *touchRemoved* function at the very beginning of their function. The user must first remove the touch that is no longer detected from the list of current touches, something that JavaScript requires to be manually done, and then the function checks to see how many touches remain. If the number is below three, then the function stores the last known location of the location of the token that was removed and returns a true to indicate that the token was indeed removed. If the number of remaining touches is three or more, the function returns false to indicate the token is still on the board and that a different touch event has been removed. The user can then use these two states to handle the removal of the token as

they see fit, be it that they count this as a player picking up a checker piece or placing it down in a new spot. The reason that I left this choice up to the user of the library, as I did with event listeners, is because it allows for more to be done with the library. If I forced users to use event listeners, it could limit how they could implement their games.

Overall, I designed the system to cater to the needs of users in general by allowing them to easily detect and track physical games pieces without limiting what they could otherwise do with the games they want to design using the system. With the system now designed and functional, it was time for me to get users to test it to make sure it was functioning to the best of its ability.

5. Testing the Library

5.1. THE TEST GAME

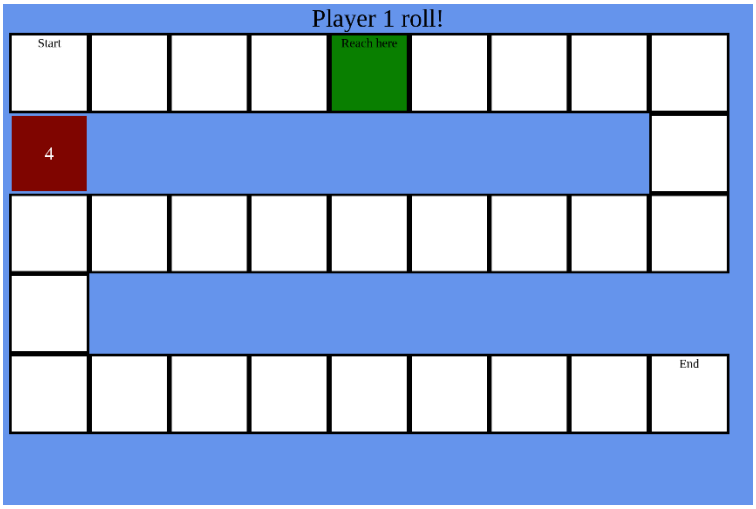


Figure 17. The gameboard as it appears in the test game Roll [Source: Author].

In order to test that the Minotauros JavaScript Library was working properly, I decided to create a test game that I simply called *Roll* (Figure 17.). This game was implemented using JavaScript and was tested on the *Google Chrome* browser on a *Chromebook*. The game consisted of 28 spaces and a virtual die that players tap to roll. The game would start with a simple prompt; "Player 1 Roll!." Once the player used their finger to tap the die displayed on the screen, the number it displayed would change and the space they needed to reach would change to green and display the words "Reach here." The player would then take the player one token and either slide it from the start position and release it on the green space or pick it up from the start position and place it on the green space. Once this action was complete, the space would turn blue to indicate it was occupied and the prompt at the top of the screen would change to "Player 2 Roll!" The second player would then tap the die to roll it and then move their piece, either by sliding or lifting, to the space that turned green. Once on this space, it would turn yellow to indicate the second player was occupying the space and the prompt at the top of the screen would switch back to reading "Player 1 Roll!" This would continue with both players performing these actions until one reached the final space on the game board. Due to the board being limited in size due to the small size of my *Chromebook's* screen, I made it so that players would never land on the same space as the spaces were too small to accommodate two tokens.

Figure 18. A Chess pawn covered with conductive ink used during Roll [Source: Author].

This game was designed to be very simple so that player would not need to spend an extensive amount of time learning the rules as the game was not what I was testing. The game was a means to test that the library functions were correctly implemented and thus could be used for creating more advanced games. The *registerToken*, *getTokens,* and *registerTokenTouch* functions were tested when the player first placed their token on the start space. From there, *identifyToken, updateTokenPosition*, and *touchRemoved* functions were tested when the player place their token on the space they were told to reach and released it, with the token's position being written to a log file to look at afterwards to make sure that it had kept track of the positions accurately. For this test game I decided to create tokens out of pawns from a Chess set and used conductive paint as Paul Ferragut had to make it so my *Chromebook's* touch screen could sense the pawns (Figure 18.). This was a cheaper and easier way of testing different angles at the bottom of the pawns since, if I had made them out of pewter, I would not be able to change where the dots were located after they were made. This would mean if a mistake were to occur I would have wasted time and resources on a higher fidelity token, which, as I went over in the *Developing a Proof of Concept* section, is something that can easily be avoided by making something low fidelity but able to test the functionality of the system. With the game that

I would test the system with now setup, it was time to create a testing methodology and recruit some volunteers to test the system through the game.

5.2. CONDUCTING THE TESTS

In order to conduct the tests, I followed WPI's IRB standard and first read a statement to all participants stating what the study was and that no personally identifiable information would be collected. I then provided them with the IRB's consent form for them to sign before proceeding with the rest of the test. To start, I explained to the testers how the system functioned by first showing them the game board on the tablet screen and allowing them to examine the tokens that I had made for the game, making sure to point out that their fingers needed to be in contact with the black paint on the token when holding it for the game to function properly. I then explained how to play the game which was kept brief due to the simplicity of the game and had them play it against me. During this time, I took notes of how players interacted with the game token when it was their turn. Once the game was completed, I presented the tester with a short survey hosted through *Google Forms* that focused on how easy they felt it was to use the token, how they preferred to move it, how they felt about the size of the screen, and any troubles they may have noticed. Once the survey was complete, I thanked testers for their time and dismissed them from the testing session.

The testing for this game was entirely done in person since both a tablet and specialized game tokens were needed for testers to play it. In addition, it also made it much easier for me to take note of how users interacted with the token in person than if they were to do so remotely. Unfortunately, due to the COVID-19 pandemic currently

taking place across the globe I was unable to get many testers as I was unable to meet with them in person. While I did not get as many testers as I had wanted, I believe that the information that I was able to gather from the testers who were able to help with it before the pandemic is very valuable and is enough to help me reach some solid conclusions.

5.3. THE RESULTS AND ANALYSIS

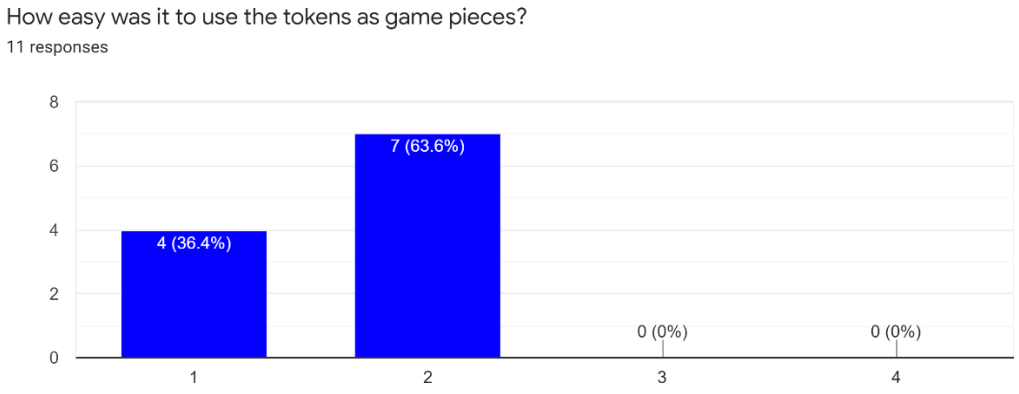


Figure 19. A scale of 1 being easy to use and 4 being difficult was used [Source: Author].

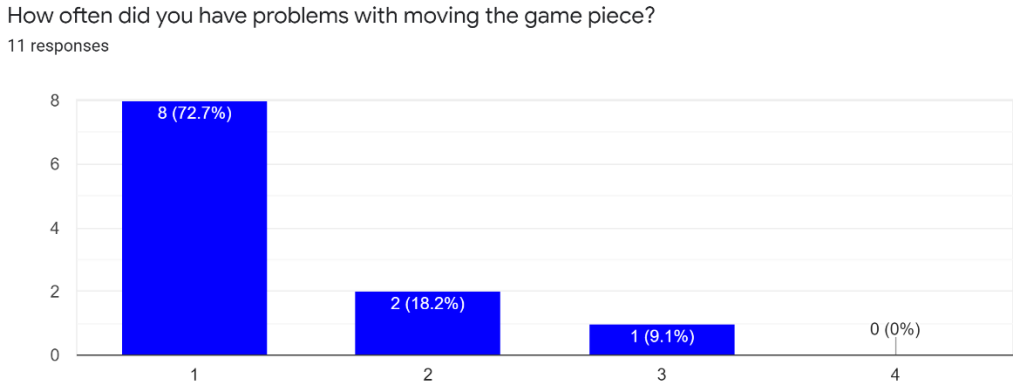


Figure 20. A scale of 1 being never and 4 being often was used [Source: Author].

Through testing, I was able to observe that the library was indeed working as intended. The game was correctly registering pieces at the start and was able to accurately detect the pieces when they were placed down on the tablet screen. As can be seen in Figure 19 and Figure 20 above, users very rarely had problems with moving the game token and most found it easier to use than not. Those that did not rate the piece as perfectly easy were often those that found the flaw with the low fidelity tokens: the paint rubbed off very easily. This resulted in people often picking up the token to find that some of the paint had rubbed off on their fingers and, after just one or two testing sessions, the tokens would need to be repainted in order to continue being accurately sensed.

Something that I noted from the survey results were the small handful of testers that had problems with the pieces being detected sometimes when moving them were also the same testers who had a comment at the end of the survey about how the paint rubbed off a lot on their fingers when playing the game. This leads me to believe that the inaccuracy of the pieces during these times was due not to the library not functioning properly, but due to the paint rubbing off on their fingers and preventing the electrical current from flowing through the paint correctly to be registered by the screen. While this is a problem, I believe it is something that can be easily overcome using pewter tokens as I had previously stated. The reason for this is that the tokens do not need any paint at all and do not need to be held in a certain way to be detected by the screen, and thus the problem of not being detected due to paint rubbing off on the fingers of users would not take place.

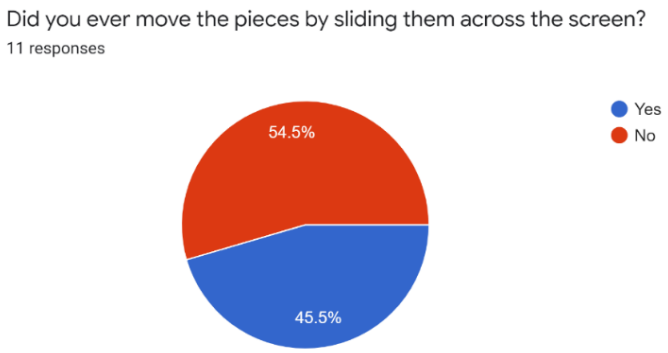


Figure 21. Most users did not even try sliding the token to move it [Source: Author].

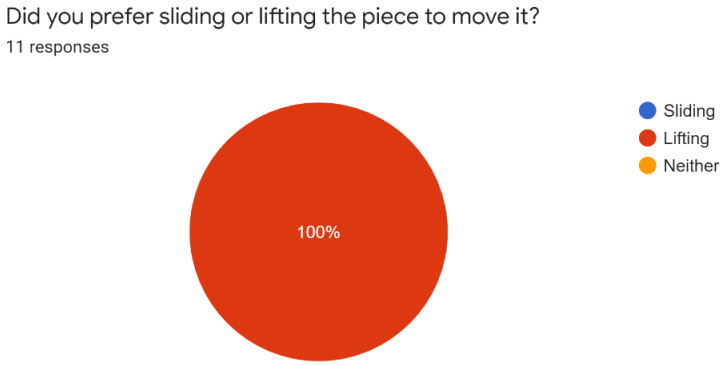


Figure 22. All users preferred lifting the game token to sliding it [Source: Author].

The next important result that I received from testing was how players preferred to move their tokens when playing the game. As I had included the ability for the library to detect both sliding and lifting up and placing down methods of movements, I believed that this information would help me to focus on which to emphasize as the main form of movement that the library would promote. From both observing testers and from the survey results, I found that testers mostly did not even try to slide the game token across the screen and all testers preferred to lift their tokens to move them (Figure 21.)

(Figure 22.). The reason behind this I found to be due a combination of not wanting to scratch my screen and not wanting paint to rub off on it. This information was found from testers making comments during the game. However, less than half of users made these comments and thus these reasons cannot be assumed to apply to all testers who preferred lifting their token.

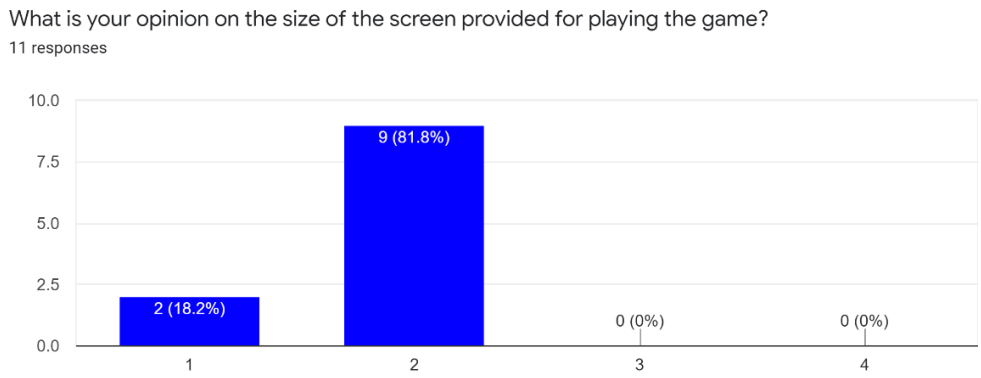


Figure 23. A scale of 1 being too small and 4 being too large was used [Source: Author].

The final important result that I found through testing was that users found the tablet screen that was used for testing to be somewhat small. While most did not find it to be far too small, none thought it was even slightly large either (Figure 23.). The size of the tablet screen that I was using was 10.5 inches by 7 inches, which is within the range of an average size tablet [36]. This indicates that standard tablets may feel a bit too small for testers to use for tablet-based board games, but since most testers did not think that the tablet was very small but rather just a little bit small, I don't believe this will prevent potential users from using the tablet they already own to play tablet-based board games.

6. Conclusion and Future Work

With the Minotauros JavaScript Library fully functional and tested, it is time for the project to come to its conclusion. While working on this project, I learned a great deal about the history of tablet-based board games and other blends of digital and physical media. From these past implementations I was able to learn both what approaches worked for making a blend of digital and physical and which approaching I should avoid if I were to have my library succeed. Through thorough testing, research, and hard work I was able to make a fully functional JavaScript Library that will soon be hosted on GitHub for all content creators to freely access. This library will allow users to create tablet-based board games for most available tablets and won't need expensive, specialized equipment to make their games functional. This not only makes tablet-based board games more accessible to developers but more accessible to players of their games as well.

In the future, I plan to continue updating the Minotauros JavaScript Library so that it remains compatible with web browsers. The reason for this being that the way that JavaScript works in web browsers is constantly changing and thus the syntax in the library may become outdated within a few years or even months. I would also like to continue my testing of the library's functionality on other tablets and web browsers, although this will take an extensive amount of time and resources. In addition, I plan to fully create a tablet-based board game version of the Minotaur's Labyrinth game using the library that it helped to create. I also plan to create high fidelity pewter game tokens for the final version of this game to show how well the library can perform when provided with the material it was designed to interact with. It is my hope that this game will serve as an example to

other content creators of the Minotauros JavaScript Library's worth and will inspire them to start using it to create tablet-based board games of their own.

7. Works Cited

1) DDRGame. "Dance Dance Revolution for Wii, PS2, PS3, Xbox 360, & PC." *Dance Dance Revolution for Wii, PS2, PS3, Xbox 360, & PC*, www.ddrgame.com/.
2) "Secret Hitler." Secret Hitler, www.secrethitler.com/.
3) Bezier Games. "One Night Ultimate Werewolf." *Bezier Games*, beziergames.com/products/one-night-ultimate-werewolf.
4) Sphero, director. *Star Wars App-Enabled Droids by Sphero - Getting Started*. *YouTube*, 7 Nov. 2017, www.youtube.com/watch?v=LpofP2CVBwA.
5) Simatupang, Joni & Yosua, Michael. (2016). A Remote Controlled Car Using Wireless Technology. 1.
6) "R2-D2 App-Enabled Droid." *Amazon*, images-na.ssl-images-amazon.com/images/I/91GsHOAPGnL._AC_SL1500_.jpg.
7) "What Is the VOID?" *The VOID*, 2019, www.thevoid.com/what-is-the-void/.
8) "Jumanji VR: Reverse the Curse." *The VOID*, 2019, www.thevoid.com/dimensions/jumanji-vr/.
9) "Lenovo Horizon 2 Multimode Table PC: Lenovo US." *Lenovo Horizon 2 Multimode Table PC | Lenovo US*, 2015, www.lenovo.com/us/en/desktops/lenovo/horizon-series/horizon-2/
10) MobileTechReview, director. *0:01 / 15:24 Lenovo Horizon 2 Review*. *YouTube*, 8 Jan. 2015, youtu.be/KunrlEr14Uc.

11) "Lenovo Horizon 2 Multimode Table PC." *Lenovo.com*, 2015, www.lenovo.com/medias/lenovo-desktop-horizon-2.png?context=bWFzdGVyfHJvb3R8MTAwNDY3fGltYWdlL3BuZ3x0M2YvaGJmLzk2NzcxNzgwNDQ0NDYucG5nfDhmMDJmZWM2YTk5MWIwM2RkZGIwZTBmMTE5M2EyYTgxNzZkMzVhNjA4ZmJkYmMwNmNlNzBkYmNiODEyNDU1MjQ.
12) "The Fun Starts Here!" *PlayTable*, www.playtable.com/?fbclid=IwAR19UWGsbQPqxkqRThXp42pUWh_xTaNIYd4hJzGvZz7Y5H2uin_benwm2bc.
13) "PlayTable Functioning with Nintendo Amiibos." *PlayTable.com*, assets.website-files.com/59a1346fb4ef790001d63d06/5aa92e81dfb04b40b66f0c70_thunder-p-500.jpeg.
14) Wittlake, Brandon. "Brandon Wittlake." *PlayTable Community Public Group*, 15 Mar. 2020, www.facebook.com/groups/playtablecommunity/permalink/648988215668399/.
15) "PlayTable." *Facebook*, 5 Sept. 2014, www.facebook.com/playtableofficial/.
16) "PlayTable Community." *PlayTable Community Public Group*, 29 Aug. 2019, www.facebook.com/groups/playtablecommunity/.
17) "PlayTable Malfunctioning with More than 4 RFID Tags on It." *Facebook*, scontent.fbed1-1.fna.fbcdn.net/v/t1.0-9/89489245_10218771909701902_7165081055567806464_0.jpg?_nc_cat=101&_nc_sid=1480c5&_nc_ohc=MU8DElUgGe4AX-sWkfT&_nc_ht=scontent.fbed1-1.fna&oh=7a72f0025fc2dd8a46515f8886cc359e&oe=5EA62752.
18) Statista Research Department. "Tablet Users Worldwide 2013-2021." *Statista*, 19 Feb. 2020, www.statista.com/statistics/377977/tablet-users-worldwide-forecast/.

19) Rubino, Daniel. “Please Stop Saying Surface Pro X Is Too Expensive - It's Definitely Not.” *Windows Central*, Windows Central, 29 Jan. 2020, www.windowscentral.com/stop-saying-surface-pro-x-too-expensive.

20) Nicoll, Leslie H. “How to Use the Browser on Your Kindle Paperwhite.” *Dummies*, www.dummies.com/consumer-electronics/tablets/kindle/how-to-use-the-browser-on-your-kindle-paperwhite/.

21) “The Most Popular Browsers.” *Browser Statistics*, 2020, www.w3schools.com/browsers/.

22) MILLER, MADELINE. *CIRCE: the Sunday Times Bestseller*. BLOOMSBURY Publishing, 2019.

23) Source. “The Myth of Theseus and the Minotaur.” *Knossos Guides*, 3 Dec. 2018, knossosguides.com/blog-view.php?id=53.

24) “The Myth of Theseus and the Minotaur.” *Knossos Guides*, 2018, knossosguides.com/blog/wp-content/uploads/2018/12/the-myth-of-theseus-and-the-minotaur.jpg.

25) Mifsud, Justin. “Paper Prototyping As A Usability Testing Technique.” *Usability Geek*, 26 Sept. 2019, usabilitygeek.com/paper-prototyping-as-a-usability-testing-technique/.

26) Goodman, Danielle. “Agile Process: Why You Need Feedback Loops During & After Sprints - Iterative Agile Loops.” *Mendix*, 27 Feb. 2020, www.mendix.com/blog/agile-process-why-you-need-feedback-loops-both-during-and-after-sprints/.

27) “Agile Feedback Loops.” *Mendix*, images.mendix.com/wp-content/uploads/agile-feedback-loops.svg.

28) Dube, Ryan. “Capacitive vs. Resistive Touchscreens: What Are the Differences?” *MakeUseOf*, 28 Nov. 2018, www.makeuseof.com/tag/differences-capacitive-resistive-touchscreens-si/.

29) “MultiTouch Tester - Apps on Google Play.” *Google*, Google, 1 Dec. 2012, play.google.com/store/apps/details?id=com.the511plus.MultiTouchTester&hl=en_US.

30) “X-WING STAR FIGHTER.” *Metal Earth*, www.metalearth.com/x-wing-star-fighter.

31) “Monopoly: Star Wars Saga Edition.” *BoardGameGeek*, boardgamegeek.com/boardgame/15046/monopoly-star-wars-saga-edition.

32) Ferragut, Paul. “Object Interaction With Touchscreens.” *Instructables*, Instructables, 24 Sept. 2017, www.instructables.com/id/Object-Interaction-With-Touchscreens/.

33) Ferragut, Paul. “Paul-Ferragut/OscTouchObject.” *GitHub*, 2017, github.com/paul-ferragut/oscTouchObject.

34) Ferragut, Paul. “How Ferragut's System Functions.” *Instructables*, 2017, cdn.instructables.com/FLL/UP35/IT22N6L0/FLLUP35IT22N6L0.LARGE.jpg?auto=webp&frame=1&width=700&fit=bounds.

35) Convivial, director. *Unedited - Object Interaction with Touchscreens*. *Vimeo*, 26 Mar. 2020, vimeo.com/182872656.

36) Type/Code. “Screen Sizes: Viewport Sizes and Pixel Densities for Popular Devices.” *Screen Sizes | Viewport Sizes and Pixel Densities for Popular Devices*, screensiz.es/tablet.

37) “Web Browser Market Share.” *W3Counter*, Feb. 2020, www.w3counter.com/globalstats.php.

38) Chandler, Nathan. “What's an NFC Tag?” HowStuffWorks, HowStuffWorks, 14 Mar. 2012, electronics.howstuffworks.com/nfc-tag.htm.

39) Horsey, Julian. “EPawn Arena, Combines Traditional Board Games With Digital Interaction (Video).” Geeky Gadgets, 28 May 2015, www.geeky-gadgets.com/epawn-arena-combines-traditional-board-games-with-digital-interaction-video-13-01-2012/.

Appendix A: IRB Protocol

Project: A Hybrid Physical/Digital Board Game

Purpose of study

To obtain playtest feedback in order to locate/address operational bugs, and to identify opportunities for design improvement.

Study protocol

Participants are provided a computer on which to play the sample game. Investigator observes participants during play, answering questions and providing guidance as needed. Afterward, participants are asked to fill out a short survey to characterize their subjective experience.

Opening briefing for testers

“Hello, and thank you for volunteering to test our game. Before we begin, could you please read and sign this Informed Consent form? [Tester signs IC form.] Thank you. During your test session, the game will be recording a variety of metrics about your play activity. When your session is complete, we will ask you to complete a brief survey about your play experience. At no point during your play session, or in the survey after, will any sort of personal and/or identifying information about you be recorded. Please begin playing when you feel ready.”

Appendix B: IRB Informed Consent Document

Informed Consent Agreement for Participation in a WPI Research Study

Investigator: Brian Moriarty, IMGD Professor of Practice

Contact Information: bmoriarty@wpi.edu, 508 831-5638

Title of Research Study: A Hybrid Digital/Physical Board Game

Sponsor: WPI

Introduction: You are being asked to participate in a research study. Before you agree, however, you must be fully informed about the purpose of the study, the procedures to be followed, and any benefits, risks or discomfort that you may experience as a result of your participation. This form presents information about the study so that you may make a fully informed decision regarding your participation.

Purpose of the study: The purpose of this study is to obtain feedback on the thesis project in order to facilitate design improvements and find/address operational bugs.

Procedures to be followed: You will be asked to play a brief game lasting less than ten minutes. After completing the game, you will be asked to complete a brief, anonymous survey describing your subjective experience.

Risks to study participants: There are no foreseeable risks associated with this research study.

Benefits to research participants and others: You will have an opportunity to enjoy and comment on a new game API under active development. Your feedback will help improve the game experience for future players.

Record keeping and confidentiality: Records of your participation in this study will be held confidential so far as permitted by law. However, the study investigators and, under certain circumstances, the Worcester Polytechnic Institute Institutional Review Board (WPI IRB) will be able to inspect and have access to confidential data that identify you by name. Any publication or presentation of the data will not identify you.

Compensation or treatment in the event of injury: There is no foreseeable risk of injury associated with this research study. Nevertheless, you do not give up any of your legal rights by signing this statement.

For more information about this research or about the rights of research participants, or in case of research-related injury, contact the Investigator listed at

the top of this form. You may also contact the IRB Chair (Professor Kent Rissmiller, Tel. 508-831-5019, Email: kjr@wpi.edu) and the University Compliance Officer (Jon Bartelson, Tel. 508-831-5725, Email: jonb@wpi.edu).

Your participation in this research is voluntary. Your refusal to participate will not result in any penalty to you or any loss of benefits to which you may otherwise be entitled. You may decide to stop participating in the research at any time without penalty or loss of other benefits. The project investigators retain the right to cancel or postpone the experimental procedures at any time they see fit.

By signing below, you acknowledge that you have been informed about and consent to be a participant in the study described above. Make sure that your questions are answered to your satisfaction before signing. You are entitled to retain a copy of this consent agreement.

_____ Date: _____

Study Participant Signature

_____

Study Participant Name (Please print)

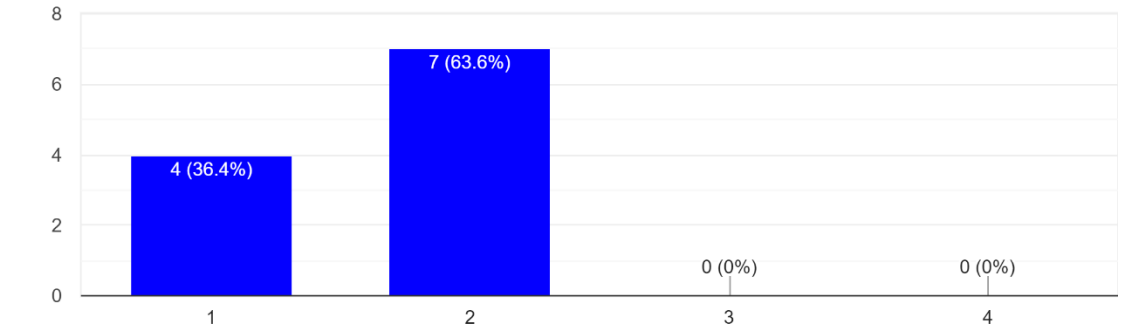
_____ Date: _____

Signature of Person who explained this study

Appendix C: Survey Questions and Results

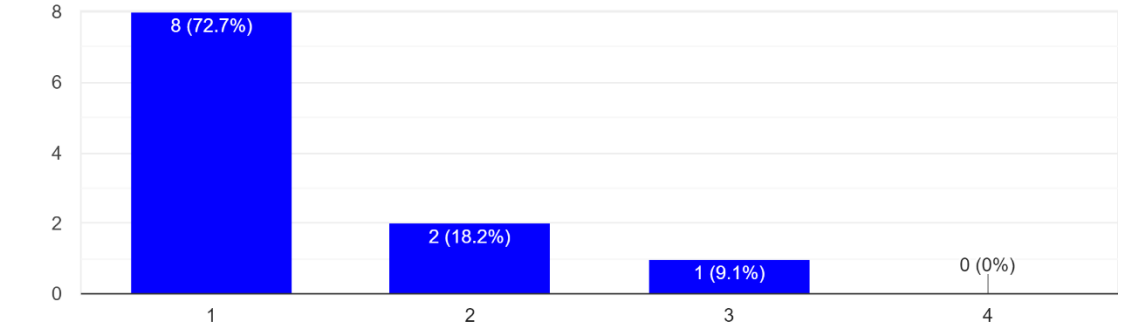
- Survey Question 1

How easy was it to use the tokens as game pieces?
11 responses



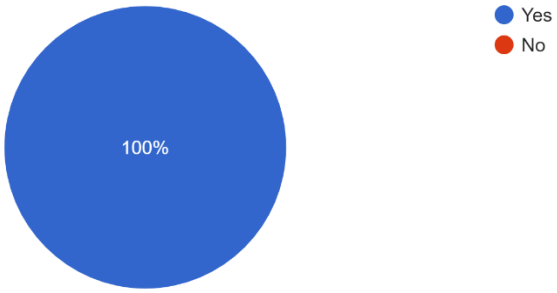
- Survey Question 2

How often did you have problems with moving the game piece?
11 responses



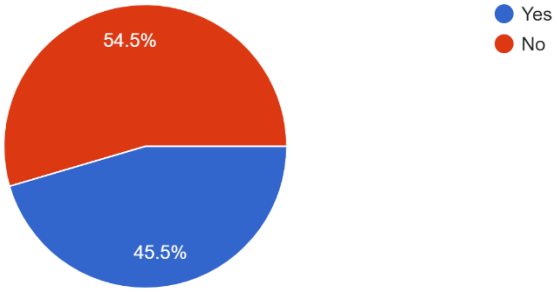
- Survey Question 3

Did you ever move the pieces by lifting them?
11 responses



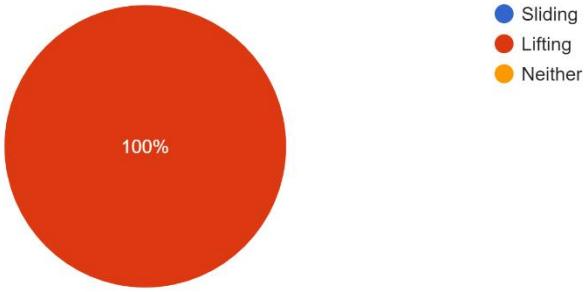
- Survey Question 4

Did you ever move the pieces by sliding them across the screen?
11 responses



- Survey Question 5

Did you prefer sliding or lifting the piece to move it?
11 responses



- Survey Question 6

Were there any specific problems that you noticed while playing?

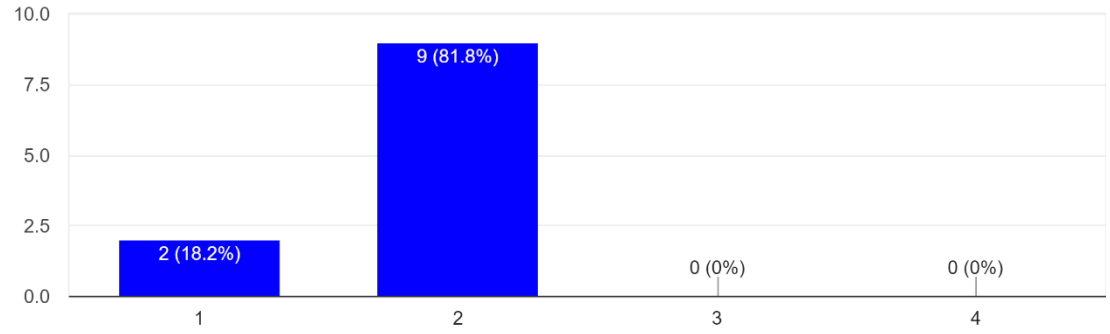
11 responses

None
Piece didn't always register right away
Moving didn't always work
N/A
Sometimes it would take a couple tries for the piece to work

- Survey Question 7

What is your opinion on the size of the screen provided for playing the game?

11 responses



- Survey Question 8

Do you have any comments or concerns about the way in which you played the game?

11 responses

None
A bigger screen might make playing easier
The paint was kind of annoying
My hands got covered in black paint
Paint kept rubbing off on my fingers
The paint rubbed off on my hands
I got paint on my fingers
N/A

Appendix D: Minotauros JavaScript Library JSDoc

Appendix E: GitHub Repository