SpellBound: A Mixed Reality game

A MIXED REALITY GAME

JOSHI, Abhinav 3035203857
CHEUNG CASADO, Sheng Leung 3035123863
COMP 4801
1st December 2017
Summary/Abstract

Virtual reality as a platform has taken off recently with the advent of industry leaders like HTC, Oculus by Facebook and even Apple trying to secure the ever-spreading market. Additionally, the gaming and entertainment industry is expected to gain the most with the spread of the new technology.

However, it is the mixed reality sector which has been overshadowed. Projects and games rarely incorporate the physical world with the virtual world. With them either being wholly in virtual space like Oculus rift or augmenting reality like the HoloLens. To have total immersion in the virtual space while having tactical and physically feedback in the real world creates a possibility to total immersion.

We, the development team (the Team) aim to develop SpellBound, a multiplayer virtual reality game with aspects of and role-playing game (RPG) and gameplay of a first-person-shooter (FPS). Furthermore, to create a sense of physical engagement and the Team will also develop haptic vests to showcase how integrating physical feedback can result in an immersive gaming experience.
Acknowledgment

We would like to express our deepest appreciation to those who have provided us with opportunity to work on this project. We would like to extend our gratitude to my supervisor Dr. CHOI without her we would never have thought of undertaking this project.
Table of Contents

Summary/Abstract.................................................................................................................2
Acknowledgment..................................................................................................................3
Table of Figures....................................................................................................................5
Project Background..............................................................................................................6
  General Background ......................................................................................................6
  Related Works ..............................................................................................................7
    A Vox eclipse and Dino Destroyer ...............................................................................7
    Haptic Suit Development .........................................................................................7
Motivation ..............................................................................................................................8
Project Methodology ..........................................................................................................9
  System setup ................................................................................................................9
Hardware Required ..........................................................................................................10
  Leap motion ...............................................................................................................10
  VR Headset (HTC Vive) ............................................................................................10
  Arduino boards (Haptic Suit controllers) ..............................................................10
Software required ............................................................................................................10
  UNITY (Game engine) ..............................................................................................10
  Leap motion API (Orion) ..........................................................................................11
  HTC Vive API (Steam VR) ........................................................................................11
  Uniduino Software (Unity plugin) ............................................................................11
Game Mechanics ..............................................................................................................12
Current Progress ...............................................................................................................14
  Game Development ..................................................................................................14
UI Design User Interface Design ......................................................................................14
  Leap Motion and UI Interactivity ............................................................................15
  Mechanics ................................................................................................................16
  Networking .................................................................................................................16
Haptic Suit Development Design ......................................................................................18
  Assets ........................................................................................................................18
  Custom Suit ...............................................................................................................18
Future Goals ......................................................................................................................19
  Photon Unity Network (PUN) ..................................................................................19
Game Aesthetics ...............................................................................................................19
Conclusion .........................................................................................................................20
SpellBound: A Mixed Reality game

References .............................................................................................................................................21
Appendix...............................................................................................................................................23

Table of Figures

Figure 1 Architecture Overview of System for SpellBound. .................................................................9
Figure 2: The different phases in SpellBound. .......................................................................................12
Figure 3: Rock-paper-scissors nature of the game mechanics in SpellBound .................................12
Figure 4: Menu UI ..................................................................................................................................14
Figure 5: HUD overlay ........................................................................................................................15
Figure 6: Select Phase Leap Motion to UI Interactivity: Player is choosing action for turn ...........15
Figure 7: Select Phase Leap Motion to UI Interactivity: Player guesses left lane .......................16
Figure 8 Correct Spawn Point Behavior (1 Player Use Case) .............................................................17
Figure 9: Haptic Feedback Outline (Front-view) .............................................................................18
Figure 10 Unity Map for SpellBound ...............................................................................................18
Figure 11 Simple Character model in development .......................................................................18
Project Background

General Background

There have been multiple market projections forecasting the rise of virtual, augmented, and mixed reality (VAMR) applications onto the next multibillion-dollar industry by the next decade [1, 2, 3]. With the advent of AR applications such as Pokémon Go! [4] and a plethora of consumer grade hardware such as the Oculus rift, HTC Vive, Sony PlayStation VR (PSVR) or Microsoft HoloLens being adopted at a high pace [5,6]. The demand for highly immersive and quality games is rising. The communities created by users, can attest to such claims with users demanding more immersion as the line between reality and the virtual space slowly fades, exemplifying the need of more VAMR apps. Virtual reality (VR) is defined as a computer-generated 3D environment, which can be interacted by a person, and is expected to become a household name in the likes of the entertainment and gaming industry. VR devices are the first front to this union appealing heavily to the senses of the vision to the users. However, haptic devices may enable a higher degree of simulation by creating the perception of touch and feedback, enhancing the VR experience [7].

The research efforts in creating a visual, auditory, kinesthetic, and haptic VR technology started off as early as the late 80s to early 90s, with fully commercial hardware available to the users. However, it was not until the early 2010s that widespread public attention came to VR when the likes of Google, HTC, Samsung, Sony, Facebook, Microsoft etc. started to invest in this technology [8]. VR headsets quickly gained traction and became the hot topic in the gaming industry as the next evolution of games. The $2 billion acquisition of Oculus VR by Facebook in 2014 [8], the cheap and ease of availability to VR by the introduction of Google Cardboard and the announcement of the PSVR created a whole new industry waiting for new development [9]. At the same time, development of several haptic devices came into place, but despite the technology outburst, these never gained similar traction or public appeal [13].

The need to provide a simple solution of an immersive game with already proven technologies is required. Moreover, providing simple libraries and compatibility to promote the viability of haptic devices in enhancing the entertainment industry is needed.

The remainder of this section will introduce similar endeavors to this project, clarify our motivation, and our objective.
Related Works

Software that emphasizes the immersive capabilities of VR technology merged with Leap Motion is not a new idea. The same is the case for haptic feedback devices that aim to improve gameplay immersion. The following outlines examples of each idea, but also emphasizes how they incentivize this project.

A Vox eclipse and Dino Destroyer

Vox eclipse is a VR game requiring the use of an Oculus Rift and a Rift-mounted Leap Motion Controller to enable the user to use his/her hands freely in the virtual world and interact with items using hand gestures such as picking up, pinch to zoom etc. Users use their hands to pick up rocks and shoot at dinosaurs. Reviews of the game praise its unique approach and the usage of leap motion as a controller to offer players a unique experience. The only thing lacking in the game is the sense of touch, or feedback while picking up rocks, making the game feel less immersive and gimmicky.

Haptic Suit Development

A haptic vest or suit is a piece of equipment worn by the user to provide him/her a feeling of tactile feedback when he/she interacts with the environment. This creates a sense of immersion with the user when used along auditory and visual simulation.

The idea of a sensory feedback system is not new. The Aura Interactor was one of the first instances of earlier haptic gaming devices. Conceived by Larry Shultz and launched in 1994, it was first of its kind to use bass sound waves to deliver vibrations providing the user a sense of touch [11]. However, this idea didn’t catch on due to lack of software and proper game titles to support it.

In 2010, students at the University of Pennsylvania developed the Tactile Gaming Vest (TGV), which enables users to play Half-Life 2 and feel haptic feedback similar to being shot, or slashed. Though reportedly, the vest only responded to receiving game events. No feedback would be provided if the game event originated from the user of the vest [12]. However, the vest was not integrated to VR equipment as users played the video game using a desktop computer [11,12]. But with the advent of VR and the outreach to users with the use of platforms such as Kickstarter many new companies have ventured into the haptic vest technology.

In 2017, Woojer a startup, started manufacturing a vest with better implementation using the latest technology along with a SDK for developers to provide an experience similar to what the Aura Interactor hoped to achieve [15]. A major competitor to Woojer is KOR-FX, another startup launched in 2014 from a successful Kickstarter campaign. The KOR-FX vest already has tie ins with games such as Resident Evil 7 and is available to consumers [16]. However, it lacks in proper game support and a user base. Also, the vest is very small in design, covering only the upper torso section.
Motivation

There are two issues with the existing efforts in haptic feedback for VR games: Firstly, there is a difference in focus. The majority of commercially available options for haptic devices do not offer support for VR games, and those that claim to do so are still under development [20]. Secondly, as of the time of this writing, search results from popular game portals such as Steam, among other online resources [21], indicate that there is a general lack of multiplayer VR games that provide haptic feedback.

The current focus for frontrunners in haptic VR devices is on providing the tactile experience as a whole package to maximize the commercial value of the technology. Consequently, not only are developers implementing the haptic sensation, but also valuable development time is being consumed building technologies to layer atop the haptic feedback functionality (e.g. independent position tracking, and gesture recognition) when there already are existing technologies that provide said services.

Although creating such APIs provide some long-term benefits in terms of the standardization in the future development process, it is also important to expose the haptic-enabled VR experience to the public quickly to explore potential short-term unknowns. For example: medical concerns analogous to those caused by VR HMDs could arise [22]; alternatively, the public may simply disapprove of the device design or find the gaming experience too cumbersome. In the case of multiplayer VR games that aim to include a haptic component, it is especially important to clarify such prospects since they are few in number.

Moreover, the hardware components required to realize the haptic suit are relatively cheap. This means that our development process can be packaged as a do-it-yourself kit by posting simple instructions and the parts required. As the suit doesn’t require any expensive components it can be built in a fraction of the cost of what the competitors are selling their vest from. Moreover, this will not only create a greater user base for haptic feedback devices but by doing so may lead to expansion of the games which utilize haptic feedback.

Thus, it is convenient and faster to integrate proven technologies such as Leap Motion and Microsoft Kinect to investigate such possibilities.
Project Methodology

System setup

A summary of the system setup can be seen in Figure 1. Under the motion capture block, the three devices from top to bottom are: the Microsoft Kinect, the Leap Motion, and the HTC Vive.

The Vive image features the Vive HMD on the center, the two motion controllers on the front, and the two lighthouse sensors. It is worth noting that we are not using the motion controllers in their traditional sense. Instead we plan to attach them to the haptic suit to take advantage of their inherent position tracking ability. By extension, the original motivation behind using the Kinect was because we could use it for position tracking of the entire body. However, the motion controllers for the Vive already fulfill this need. Therefore, the Kinect has been removed from development.

The directed arrows represent the expected input-output relationship. The position and orientation information of the hands is delegated to Unity via the Leap Motion SDK. The 3D rendering data is sent to the Vive HMD to produce the game visuals. In turn, the HMD also sends position and orientation information, so the engine can produce the correct perspective. The unidirectional arrow on the Vive image represents the tracking data from the motion controllers as mentioned on the paragraph above. Lastly, the engine should signal when the suit should trigger the tactile feedback.
Hardware Required

Leap motion
The leap motion is a small device designed to be placed on either a physical desktop facing upwards or mounted onto a VR headset. It utilizes two monochromatic infra-red cameras and multiple LEDs to reconstruct 3D objects in its path to a virtual 3D object. It can be used as a highly specialized device to capture users hand gestures precisely [24]. Another incentive to use leap motion is its modest dimensions and weight, so it can be easily attached to the Vive HMD without hindering the playing experience.

VR Headset (HTC Vive)
One of the key equipment required is the correct VR headset and it is imperative to choose the perfect headset for maximum compatibility of our game. The two main contenders were the Oculus Rift, or the HTC Vive. While both provide excellent VR experience and are similar in specifications, the deciding factor was spatial tracking. The Vive uses a lighthouse laser tracking system whereas the Oculus uses a Constellation based tracking system. The issue with Constellation system arises as the maximum space tracked by the camera is eight by eight meters. In comparison, the lighthouse laser system is capable of a maximum tracking space of 15 by 15 feet [19].

Arduino boards (Haptic Suit controllers)
To provide the suit with a controller the Arduino Uno best suits the requirements. It is a single-board microcontroller well supported by Unity with plugin support called Uniduino [26,27]. And has an active and large community support. Along with the microcontroller will be the actuator motors responsible for firing up to give the user the sensation of touch as haptic feedback. Depending on the development process, the first focus will be on recreating mechanically induced haptic sensations and later the possibility of including more transducers, actuators, or vibrators will be entertained.

Software required
UNITY (Game engine)
Unity is a free game engine that supports development of a variety of game types from 2D platformers to VR. It also supports a huge library of VR plug-ins. It provides a highly versatile and flexible environment for game development. From 3D Animations to spatial-audio. Unity provides an array of tools for developers to use as per their requirement. Moreover, with a strong community of developers and great product documentation, Unity offers the best solution.

The Team is also more familiar with Unity development, its documentation, and its community more than other well-reputed game engines such as Unreal.
Leap motion API (Orion)
Leap provides robust support for its product the Leap Motion. In early February 2016, Leap Motion released a major software update dubbed Orion, the updated software was designed specifically for precise hand tracking in virtual reality making it a valuable peripheral for the MR game [10]. This software is easily integrated with Unity with a simple plugin.

HTC Vive API (Steam VR)
The Steam VR API is supported by unity using openVR The OpenVR API provides a game with a way to interact with Virtual Reality displays without relying on a specific hardware vendor's SDK.

Uniduino Software (Unity plugin)
The Uniduino plugin provides improved development because it centralizes all coding on the Unity platform. The plugin enables the microcontroller to be configured on a script-by-script basis, which facilitates proper code encapsulation. Much of the low-level programing for the haptic suit will be based on this software [27].
Game Mechanics

After two players are successfully paired, they will be placed on opposing sides of the middle of one of five lanes on the virtual gameplay environment. There are three game parameters: health; move ammo, which allows the user to move for a turn; and, shield energy, which allows the user to shield for a turn. The game would proceed in four phases (see Figure 2). The purpose of each phase is described below.

**Select phase:** During the Select Phase, players have a limited time to interact with a virtual menu to choose their desired action for this turn. The actions include: move, attack, grab, and defend. For actions other than defend, the player must also target a lane. For example, if the player selects move, then the player must also choose which lane to switch to on his side of the field. Alternatively, if the player chooses to attack, then the player must select the lane on which the opponent may be in.

**Evaluation phase:** During the Evaluation Phase, the game decides what will occur during the Performance Phase (see Figure 3). The relationship between attack, grab, and defend is similar to rock-paper-scissors, but only resolves if the selected actions involve the same lane.

**Performance phase:** During the Performance Phase, the player must execute his action via gestures within a limited time span. For example, if a player chooses attack and it is the correct lane, then during this phase the player would have to select what body part to attack and perform a gesture to damage the opponent. It is the interactions during this exchange that will signal the tactile feedback on the respective haptic suit. Alternatively, if the player had chosen to move lanes, then the player must physically move to another location.

**Resolution phase:** During this phase additional game events that relate to the character the player chooses are resolved. For example, the ammunition for the move option is replenished, or the character heals some damage.

Table 1 (See Appendix A) summarizes the expected resolution to the possible use cases of the performance phase. This table assumes that the users have successfully executed the gestures that are prompted during the performance phase. For example, the second column
of case one describes the case when player one chooses to attack and player two chooses to defend. This then results in four cases depending on the correctness of the lanes designated by either player.

The table also clarifies that the user does not need to designate a lane when defending. That is why in case five to seven, as indicated by the strikethrough text, checking for the correctness of player one is unnecessary. The same is the case for the move action since the lane that is designated during the select phase moves the user and does not concern the opponent. This is evidenced in cases four, seven, nine, and ten as the outcome is independent of player two (i.e. the outcome is the same across columns).

Lastly, the descriptions that are underlined and italicized indicate the cases that should trigger the haptic feedback. There are five in total: slightly damaged, damaged, extra damaged, reflected damage, and, shield energy absorption. The first four can be targeted to a specific body part, whereas the last type is felt throughout all haptic feedback points.

<table>
<thead>
<tr>
<th>Use cases for Failed/Idle State</th>
<th>Attack versus Idle</th>
<th>Defend versus Idle</th>
<th>Grab versus Idle</th>
<th>Idle versus Idle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Attack successful regardless of lane chosen</td>
<td>No defend energy deducted</td>
<td>Grab successful regardless of lane chosen</td>
<td>Turn skipped, match cancelled after multiple consecutive idle states</td>
</tr>
<tr>
<td></td>
<td>Attacker wins after multiple consecutive idle states</td>
<td>Defender wins after multiple consecutive idle states</td>
<td>Grabber wins after multiple consecutive idle states</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Use Cases for Failed Gesture Scenarios

The primary winning condition in the game is to deplete the health of the opponent. There are two other conditions that lead to the end of the game. Firstly, after a specific amount of turns has passed, the game ends, and the winner is determined by the percentage of health remaining. Secondly, after three consecutive failed status has been detected for the same player (see Table 1). The failed, or idle state occurs when a player fails to perform the prompted gestures during the select or the performance phase. Notice that for the cases in which one of the two players has selected a valid action (attack, defend, or grab), the mechanics place that player at an advantage by either automatically deeming the action successful regardless of the lane, or relieving the player of the cost for defending.

The specific number of lanes, turns, and consecutive idle state occurrences before terminating the game have yet to be fixed, and will be determined after play testing is possible. These adjustments will be incorporated to expedite game match duration.
Current Progress

The development of SpellBound is as per schedule with work on the game mechanics, along with the initial development of the haptic suit still ongoing.

**Game Development**

The team has decided to work on separate scenes corresponding to the technologies involved in the development process for a functionality. For example, the leap motion interaction with the UI has been developed separately from the networking components, and will later be merged.

**UI Design User Interface Design**

![Figure 4: Menu UI](image)

Figure 4 illustrates the interface for the menu screen. This is the screen that players encounter upon game bootup. The UI is rendered on a virtual environment that is randomly selected from one of the in-game maps. The menu options are listed and explained below:

- **New Game:** Starts a new instance of a game and begins match-making process. Upon finding a pair of players, the UI switches to the game scene and the game begins.
- **Character Selection:** Prompts the player to choose which character to play as. We have decided that the project scope include at least characters for the user to play to add some variability to the game since the team decided not to include story, and quest elements to the game.
- **Settings:** Provides the player with audio, visual, and other game options to edit.
- **View Models:** Allows the player to view the 3D models in the game.
- **View Maps:** Allows the player to view the maps in the game.
SpellBound: A Mixed Reality game

An example of the heads-up-display (HUD) for a player is shown in Figure 5. The player is capable of viewing status information such as the health on the lower left. The other icons represent the actions that are available to the player during the Selection Phase as described earlier on the Game Mechanics section.

Leap Motion and UI Interactivity

The team has completed the use case for the select phase of the game in which the user interacts with hovering menu items which serve as the in-game UI (see Figure 6). A module of display items is related to the position of the leap hand palm, and the user can use the other hand to touch which menu item to select. There are also instructions that appear on the palm of the user. The selection corresponds to a player option chosen for a turn, and the use case is the same for choosing which lane to guess the opponent is in. Figure 7 shows the user first choosing to attack, and then guessing the left most lane.
The team originally had developed a simpler version of this UI interaction which calculated the intersection of a Cursor GameObject with the hovering UI item. However, the data structure that was originally used was an array associated with a singleton GameManager entity. Calculating intersections under heavy load and maintaining sound code encapsulation for the cursors an UI proved to be a challenge. The team attempted to switch our data structure to a tree to improve retrieval. However, due to time considerations the team resorted to the Hover-UI kit, which is an open licensed SDK that provided a better framework for UI interactivity with VR technologies.

**Mechanics**

Simple character movement in a flat environment has been established. The model however does not have any action specific animation mostly relying on a simple walking animation. Hand movements with the leap motion controller has been established but is still awaiting the integration with the character. Physics behind the leap motion controlled hands are still being worked upon.

**Networking**

The team has begun to define the communication protocol between both players and the server during player matchmaking, and from player to player during a match for SpellBound. The application is capable of lobbying players into rooms hosting a maximum of two players per room. In these rooms, the team has established simple character movement, tracking of character parameters such as player health, character-bound idle animations, and an example firing event to test potential synchronization issues over the PUN network.
SpellBound: A Mixed Reality game

The development of spawn point functionality has demonstrated similar concerns. Spawn points have been coded to recognize the existence of another spawn point and intelligently align its position such that one spawn opposes another spawn, depending on the map size. Figure 8 shows the intended behavior of the spawn points, denoted by the axes and green outline. The player model correctly occupies one spawn location, while the other spawn position orients itself to oppose the spawned player. This works in the use case of one connected user, but when two users are connected, the spawn points are duplicated on the same position, and only one spawn point is occupied by both players.

The team is currently focusing on resolving this issue and we suspect that if it is not a synchronization issue, then the problem may lie in mishandled flow of control arising from Unity scene loading through PUN. Currently, when PUN detects that a player has joined the room, Unity reloads the scene, which destroys pre-existing GameObjects, namely the GameManager and already existing spawn points. Meanwhile, the are other GameObjects which must be flagged to never be destroyed on scene loading to enable PUN synchronization to work as intended, that is, the GameObjects related to the player or user. This opens the possibility for scripts to be ran in an order that triggers referencing errors. Similar anomalies were experienced when a player disconnects and attempts to rejoin the same room.
Haptic Suit Development

The design for the haptic suit is expected to be able to produce haptic feedback at 8 different points (see Figure 8). The locations of the motors that will generate the haptic sensations are denoted by the red dots.

The different color regions indicate that the motors within that color are to be triggered when a player attacks that area. Since the HTC HMD physically obstructs the head region, there will be no haptic feedback on the head region. Moreover, all haptic feedback will be induced from the front of the body since the players always face one another during game interactions.

To help with tracking the position of a player in the game, the motion controllers of the HTC Vive, denoted by the black triangles, will also be attached to the suit.

Lastly, the blue square outlined in white is the expected location of the microcontroller. Since all motors and power supply will have to be wired to the microcontroller, it needs to be in a centralized position.

Assets

Character models still lack fluid geometry and texture (see Figure 10). Currently the plan is to create multiple character classes for players to choose from with each character having its own set of variables for health, movement ammo, attack damage and shield charge. This is also complemented visually by having different 3D models and animation for each character. Creation of a 3D map environment is completed (see Figure 9). However, the constraints are still being worked upon. As for the game Design and Aesthetics we plan on creating maps with a rich environment and verbose interactivity. This will result in the player having a better overall immersive experience while playing the game.
Future Goals

The project schedule is detailed on the Appendix, but further explanation on the next two milestones has been separated into two subsections for clarity. Project research, design, preliminary set up and testing was mostly done during October. Core game mechanics realization and haptic device communication implementation were the focus during November.

Photon Unity Network (PUN)
The following phase will expand the project to enable multiplayer functionality. The team will set up the communication protocol between both players and the server during player matchmaking, and from player to player during a match.

The team will first conduct some feasibility study and testing for the PUN framework to see if there are any concerns regarding latency, particularly regarding the haptic device signaling between both player during a match.

While the team aims to maintain good development practice, the team has also decided that the matchmaking protocol does not need to be rigorously implemented because the end goal of the project is not aimed at commercialization, but rather demonstration purposes. Instead, it is paramount that any relevant gameplay delay be accounted for during a match since, as discussed in the game mechanics section, the actions of the users are timed, which may easily lead to player dissatisfaction otherwise.

The team expects this to be an extensive phase of development due to the lack of networking experience on the Unity platform.

Game Aesthetics

The team has prioritized game functionality over aesthetics, but by no means undervalue the necessity of good aesthetics. Most of the feasibility testing has been done with basic geometric shapes. No additional visual effects, animations, or audio have been created. Not all users will play a game simply because it has interesting game mechanics, as such the team has designated ample time to provide a variety of game aesthetics.

The game object modelling will take priority after the networking component is finished. Next shall be the animations for the game model interactions. Furthermore, animations that prompt for a requested gesture will be crucial since these will be guiding the user throughout the gameplay. The audio and additional visual effects will be added last. These shall be triggered on UI interaction and other game events.
Haptic Suit

As the team does not have an electrical engineering background, to solve the problem of creating the vest we consulted a friend of mine to come up with design and specifications with the Arduino. Currently, we have sorted out the equipment required and how we are going to design the suit. However, proper software integration with Unity via Uniduino is remaining and we plan to finish this by the end of January.

The other challenges we face currently are to integrate the Arduino with a skeleton compromising of the vibrating actuators. Then increase the power of the motors using the L298 Bridge. As mentioned before this is a crucial challenge as our knowledge of electrical components is restricted. Once completed we can move to the next stage of the suit development by creating a workable prototype. In the future if time is available we plan to attach the wireless shield to the Arduino so the suit is connected to computer via Wi-Fi or Bluetooth. Granting more accessibility and better cable management for the end-user.

Conclusion

SpellBound potentially offers a new and unique experience to users by providing an immersive and articulate environment. It fills the gap between the virtual world and the real world creating a Mixed reality for users to enjoy. By utilizing upcoming technologies, it makes way for the next leap in the gaming and entertainment. The game itself has its roots inspired by Classic games such as Skyrim, Final Fantasy etc. making it user-friendly and at the same time enjoyable by all. The core mechanics create a challenging and fast pace game. Its rock, paper and scissor approach makes it easy to learn even when played the first time. Moreover, Unity engine provides the perfect software implementation by “Uniting” the various hardware elements. Although to play the game, some restrictions are present due to the hardware, by making the software open source in the future along with the details of the Haptic Vest as a DIY (do it yourself) kit, SpellBound hopes to create a simple and pleasant experience for the user. In the future, several new maps and characters might be added along with adaptive environments.
References


### Table 1.1 Use Cases for Resolution of Performance Phase

<table>
<thead>
<tr>
<th>Case</th>
<th>Player 1 (P1)</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Player 2 (P2)</td>
<td>P2 correct lane</td>
</tr>
<tr>
<td>1</td>
<td>Attack</td>
<td>P1 correct</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P2 shield energy decreased</td>
</tr>
<tr>
<td></td>
<td>Defend</td>
<td>P1 incorrect</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P2 shield energy decreased</td>
</tr>
<tr>
<td>2</td>
<td>Attack</td>
<td>P1 correct</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P1 <em>slightly damaged</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P2 <em>absorbs P1 shield energy</em></td>
</tr>
<tr>
<td>3</td>
<td>Attack</td>
<td>P1 correct</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P1 incorrect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Attack</td>
<td>P1 correct</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P2 move ammo decreased</td>
</tr>
<tr>
<td></td>
<td>Move</td>
<td>P1 incorrect</td>
</tr>
<tr>
<td>5</td>
<td>Defend</td>
<td><strong>P1-correct</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P2 shield energy decreased</td>
</tr>
<tr>
<td></td>
<td>Defend</td>
<td><strong>P1-incorrect</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P2 shield energy decreased</td>
</tr>
<tr>
<td>6</td>
<td>Defend</td>
<td><strong>P1-correct</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P1 <em>slightly damaged</em></td>
</tr>
<tr>
<td></td>
<td>Grab</td>
<td><strong>P1-incorrect</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P1 <em>slightly damaged</em></td>
</tr>
<tr>
<td>7</td>
<td>Defend</td>
<td><strong>P1-correct</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P2 move ammo decreased</td>
</tr>
<tr>
<td></td>
<td>Move</td>
<td><strong>P1-incorrect</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P2 move ammo decreased</td>
</tr>
<tr>
<td>8</td>
<td>Grab</td>
<td>P1 correct</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P2 <em>absorbs P1 shield energy</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Grab</td>
<td>P1 correct</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P1 <em>absorbs P1 shield energy</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P2 move ammo decreased</td>
</tr>
<tr>
<td></td>
<td>Move</td>
<td>P1 incorrect</td>
</tr>
<tr>
<td>10</td>
<td>Move</td>
<td><strong>P1-correct</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P2 move ammo decreased</td>
</tr>
<tr>
<td></td>
<td>Move</td>
<td><strong>P1-incorrect</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>P2 move ammo decreased</td>
</tr>
</tbody>
</table>