Danny Nsouli
October 15, 2020
CSCI 6907

VR Project Report: Temple Shoot

1. Introduction

Temple Shoot is a target practice mini game challenge. The story revolves around a temple arena in the middle of a jungle that has been cursed by spirits that have taken the form of evil masks. In order to lift the curse, the player must find and physically destroy the masks with their trusty handgun. In order to summon the masks, the player must step on each of the colored skull platforms. The curse has also risen some bodies from the dead in the form of skeleton warriors that will hunt the player down around the map while the player tries to reach each platform.

To summarize, the goal of the game is to get as many points as possible by shooting down the evil masks. Each mask will grant the player five points. The player will begin the game facing one of the four instructional screens that will tell the player to go to a specified colored platform as well as their current score. These screens also hold the user interface buttons that will allow the player to retry or quit the current game session. Once the player reaches a colored platform, a barrier will appear along with another screen that will show the player how much time they will have left to shoot the masks. After the time is up, they can move on to the next platform while making sure they don't run into any roaming enemies. These skeleton warriors will walk towards and attack the player if they get close, costing them five points if hit. The player cannot destroy these enemies. Once all platforms' shooting galleries have been played, the player will be sent to the end screen which will display their total score along with the options to retry or go back to the title screen.

2. VR Application Design

The most blatant design choice for this game was to put all of the user interface buttons in the actual environment of the player. This was done to not obscure the player's vision with constant instructional text and buttons. The player can check any of the instructional screens around the

level to check these interface components. Furthermore, another rule in VR is that everything in the game that implies interactivity should be interactive. By putting the player in the game's level via VR, they will try to interact with elements of the level in ways they would in real life unlike in a regular game where they would try to find a button command. All the environments I created don't have any misleading set pieces of objects such as doors or machines with non-interactive buttons. The only buttons in the level are those that have recognizable icons such as the "retry" button or "home" button. The title screen shows what interactive buttons look like, so it makes it easy for the player to imply that the similarly designed buttons in the main level are interactive.

With VR applications, it is also easier to accidentally overwhelm the player. Too much sensory overload can cause a player to want to stop playing or be confused. For this reason, Temple Shoot is a much more relaxing game. Most distractions are kept to a minimum with only one obstacle the player must overcome at a time. The masks for example do not attack the player. An onslaught of projectiles coming from the masks may have obscured the player's vision of the targets. The target galleries also trigger a barrier around the player, making it so that the user does not have to worry about any other distractions during those isolated segments. The roaming enemies are also spread out apart from each other. This helps in lessening the chance of multiple enemies attacking the player at once causing claustrophobia. In addition, the level was kept very open and large to allow the player to have an easy sense of position and to be able to feel as though they have a lot of freedom in their movements. Starting the game in a dark tunnel or smaller area would have been jarring to start and may have also triggered claustrophobia in some players.

Since VR applications are harder to naturally understand to some users, due to it being a relatively new way of playing games, the difficulty of the game was increased at a slow pace. The masks in the third and fourth round, for example, became more difficult to shoot than the previous two rounds by increasing two values: movement speed and distance from the player. The first two rounds are slower paced to give the user more time to get comfortable with the gameplay.

In terms of movement of the player, teleportation to the different platforms/shooting galleries was made accessible to account for players that would feel nausea. In order to decrease disorientation, the teleportation in this game is node based in that the user can only transport themselves to four fixed locations being the locations of each colored platform in each corner of the map. Otherwise a player can use artificial locomotion with physical controller inputs. This allows the player to move freely without the restriction of the VR headset's "play space." It also would reduce frustration in real life player movements not being translated correctly into the gameplay (also reducing physical exhaustion). This is also why the player was not given physical legs in-game to avoid confusion if the legs didn't move the way their own legs were. The locomotion variation in this case would be classified as turret movement.

Level metrics were also taken into account. While in normal games, object sizes can vary due to the lack of physical interactivity, in VR most real-world objects should be scaled correctly according to the average player. This is why the height of the player was increased in order to more closely match the height of the skeletal enemies, leaving some difference to make the

enemies more menacing. Guide screens, however, are kept larger to allow players to more easily read them from afar. Audio effects for different interactions were also kept positional to keep the player immersed.

3. Integration

To build this VR application, I developed it using Unity. The environmental elements of the game are taken from several different asset packages related to the jungle and ancient temple theme of the game. These packages were downloaded from the Unity Asset Store. One of the more complex assets, the roaming enemy skeletons, were more difficult to find due to most asset packages lacking pre-assigned animations. After finding a package with a prefab included, it was easier to program around the animations to elicit the behaviors necessary for the intended gameplay. The VRTK asset package was pertinent in testing the application as well, as it provided multiple testing scenes with VR capabilities that could be run without the need of a VR headset and high spec computer. However, in order to create VR compatible version of the application, the Steam VR library was imported in order to integrate connectivity with the HTC Vive headset and controllers.

4. User experience (referring to effective VR application design factors)

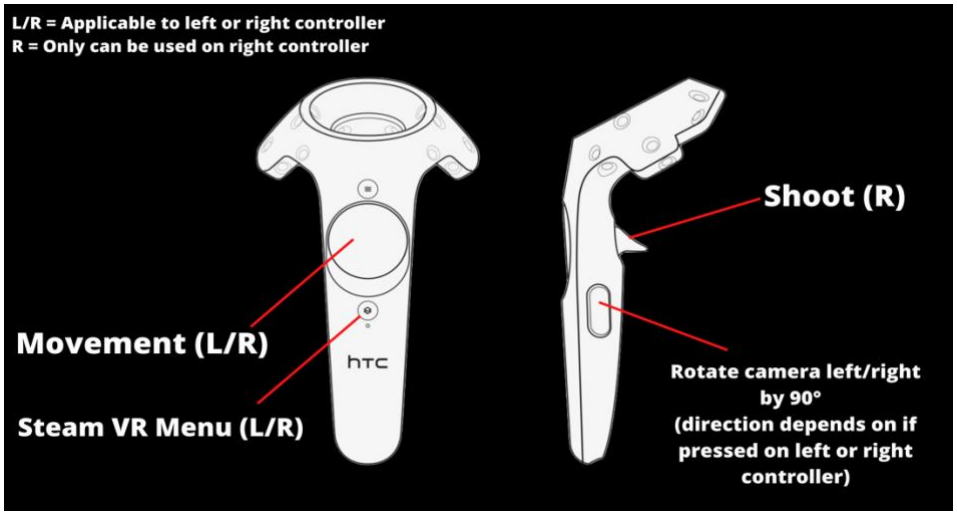
The camera control is locked to the player's head movements. This gives the player a much more natural and intuitive control feel, especially due to the turret controls. The field of view of the player is also not overridden by any distracting game objects. All interactable user interface buttons are kept within the game's level via instructional screens. There are also no screen space effects such as reflections.

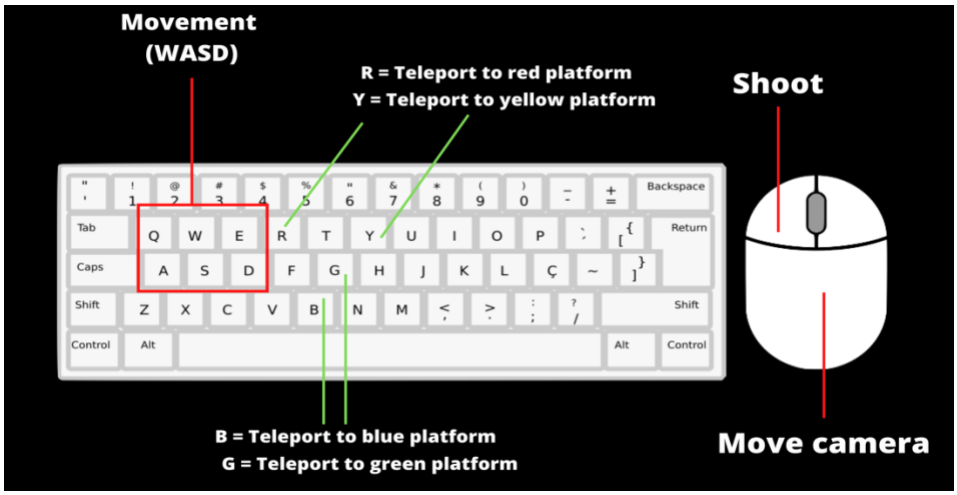
To avoid the issues of acceleration, the player movement speed was reduced from speeds that non-VR games would typically use. The slow movement of the player was compensated with manageable distances between the shooting galleries and slower enemy movement that would be easy to avoid or shoot towards. There is also an optional button command that will allow the player to rotate the direction of the camera either to the right or left by 90 degrees. This will help to reset the player's headset in case there are any detection issues. The frame rate has also been set to 90 Hz in order to account for the maximum frame rate of the HTC Vive.

5. User Guideline

This application has separate builds for three platforms: HTC Vive VR Headset, Mac, and PC. For the Vive and PC builds, the user need only launch the ".exe" file. The Mac application should have a trophy icon. Mac users can simply click on this icon to launch the application.

Below are diagrams showcasing the controls for the VR version and the regular version. To clarify, regarding the VR controls, the rotate camera button will activate when both grip buttons on a single controller are pressed at the same time. Other buttons should be self-explanatory.





6. Conclusions: Considering initial design goals, what worked, what needs improvement, what did not work?

The basic gameplay including moving and shooting worked as intended. A lot of issues came with the shooting gallery sections which I oversimplified in my design proposal. Once the player reached the platform it was easy enough to make the targets appear, however, in order to make the gameplay more intricate, I decided to make the targets move. This would allow me to alter the difficulty of each target gallery by changing the enemy speeds and movement patterns. I also realized that making the galleries time based would be more thrilling for the player, so I had to create a timer, as well as, make the masks regenerate in case the player shot them all before the timer ended. In order to keep the player from wandering away from the targets, a temple themed barrier was added to lock the player into a set area so that they couldn't leave the gallery nor get too close to the targets. Most of these ideas were inspired by play testing to better understand the mind of a casual player in those scenarios.

In the early stages of the application when configuring shooting controls, I tried to add a reticle for the gun to show where the player's bullet would go. However, there were multiple issues with the implementation. At first it was easy to get a reticle to appear on screen, but with VR it's

preferrable to put all user interface elements in the physical environment. Unfortunately, the reticle would constantly get stuck onto other objects or phase through them, becoming invisible. After watching some gameplay videos of official VR shooters, it seemed like a reticle wasn't necessary for this VR game genre as some games do without it. In a similar situation, the mini-map idea, which would help the player with directions, was switched in order to avoid the need for on-screen interfaces. The alternative idea used was to have floating arrow pointers that matched the color of the platform they were hovering over as wayfinders.

In terms of sounds, it was simple enough to figure out how to attach audio to scenes. Sound effects for enemy and player interactions were able to be implemented through programmed conditional statements. The more complex issue was figuring out how to get the colored platforms to emit sounds depending on the distance of the player. This was a form of guiding audio. Unity's interface made it possible by providing a logarithmic graph that could be manipulated to track the volume of the sound in accordance with the player's distance from the object.

I tried to also include platforming sections for the player to get from platform to platform, however, I felt that the platforming idea was not a good fit for VR. Platforming in first person is already difficult in a regular game due to the restrictive view the player has on their character. In addition, after installing the VRTK package and using their movement script, their script had not used physics in their code to move the character like my original player movement script did. Therefore, it was difficult to convert their script to be more physics based to include jumping. After some tinkering to convert the VRTK code, I was only able to get the player to jump when close to an object. As a result of both the gameplay and technical obstacles, I alternatively decided to add in roaming enemies so that there would still be an adversity the player would have to face when moving between shooting galleries on the main level terrain.

In terms of the roaming enemies, I used a prefab from an existing asset package. However, in order to make the model behave as needed, I needed to edit the code to trigger certain behaviors. One part that was somewhat difficult was getting the enemy to attack within a certain distance of the player. This was solved with the use of colliders that I had to attach to the player in order to allow the skeleton's attack animation to be triggered. This collider had to be in front of the player so that the skeleton would not only attack when right up against the player character. Another issue was hit detection due to the enemy's sword mesh collider. This collider needed to be lengthened and widened in order to provide a large enough hitbox for the player to be easily hit if close enough to the skeleton. This flexibility allowed for the enemies' attacks to be more difficult to avoid when approached since the sword is in the enemy's left hand while the player normally would be positioned in the center of the enemy's body when in contact. The colliders made it so that a large area in front of the skeleton was an attack trigger to more easily hurt the player.

Regarding improvements, I think that another idea for the galleries would have been to have the masks transport the player into a playbox. This way, the masks can visually pop from the background more. I tried to create a smaller skybox, but it didn't look natural enough in the game scene. However, I still believe that a different visual aesthetic during these gameplay sections could have helped make the targets more visually obvious to the player. This is similar

to how in some video games, when the type of gameplay changes to a short mini game, the graphics change to reflect that. The two images below are an example from *Super Mario World*.

Regular Stage:

Bonus Stage:

I also think that the skeleton enemies should walk around the stage more. For now, they stand in their designated positions and if the player approaches them, they will attempt to stalk the player. However, if the player does not get into their trigger zone and keeps their distance, the enemies will not pursue them. It was more difficult to program these enemies due to their dependence on animations, but the target moving script could be altered to allow the roaming enemies to also have predetermined walking patterns. The skeletons are placed far apart from each other, however, there is always a chance that they can wander onto a platform. This is an issue since if a player starts a shooting gallery, they could accidentally be locked in with a skeleton (a currently rare occurrence). There could also have been some more animations added such as an explosion effect when the masks are shot or red action lines when the player is attacked.

7. Suggestions for further improvements.

This VR application satisfies all the gameplay requirements. Regular movement of the player character is the activity involving Six Degrees of Freedom (6DOF) like in most first-person shooter games.

Text instructions are displayed on in-level instructional screens scattered about the map. These screens display to the user instructions on which platform to travel to next and the current score. The screens also house the user interface elements such as a home button to return to the title screen and a retry button to reset the level. Wayfinders were also included in the form of colored arrows that hover above each of the platforms allowing the player to easily tell the direction they should be heading in. Teleportation can be performed by shooting at the floating wayfinders. Shooting one, will transport the player to the corresponding platform. For keyboard controls, teleportation was simplified to different key commands that will warp the player to each platform.

Sound effects were added to a lot of actions in the game as well. The title screen, end screen, and main level each have theme music. The roaming skeleton enemies have attack sounds, the floating targets/evil masks make death sounds when shot. The player's gun also has an audio clip for when bullets are shot out. And lastly, each of the colored platforms have 3D sound which

means that when the player gets close enough to the platform, a sound will be triggered and will get louder as the player gets closer.

Moving forward with the project, there are many new ideas that could be incorporated. For one, there could be additional levels added. The player could begin outside the temple on the common grounds and once all the evil masks are defeated, they would move on to the second level in which they are in the temple itself. More levels would also entail more varying difficulty. A darker or more complex level could lead to the player having more difficulty finding the platforms for the target galleries. There could also be different types of guns that the player can use, such as a spread gun or shot gun. These would be on the ground as power ups the player can collect to change the way their gun fires bullets. A future build could have more enemy variety as well. There could be different types of roaming enemies who have different speeds and ways of attacking the player such as spiders that fall from the ceiling. In terms of the target galleries, these sections would be more fleshed out. Originally the player would have a set amount of time to begin shooting the masks, however, there could be special targets that give the player more time or targets that make the masks move slower so that they are easier to shoot.

A more advanced idea would be to add non-player characters into the game. In between levels, players could be sent to villages with NPC's that they can talk to and receive items from. There could be an inventory system to keep track of the items. This concept would give the game more personality and make the world feel more real.

8. Appendix I: Lessons Learned (Personal Journey)

I have used Unity in the past before, but this project allowed me to get even more familiar with this development tool. This was the first time I had dealt with implementing 3D sound and time-oriented events, for example. I would say one of the more difficult aspects of the design was figuring out how to link all the different gameplay scripts/mechanics. After some online tutorials, I found that Unity developers usually use a game manager script to control a lot of their gameplay. This was an efficient way of linking certain level behaviors. This project also gave me more experience with using animated assets. The roaming skeleton enemies were more difficult to program than the other characters in the game because I needed a way to edit their original behaviors. Rather than tampering with the animation state machine already setup, I decided to delve more into the code of the prefab in order to create conditional statements that trigger different actions.

In terms of VR development, it was interesting to learn and put into practice the different design factors VR games are suited for. The most difficult challenge for me was developing without a VR headset. At first, I created the game from the ground up as a regular keyboard-controlled game. Once I had the basic functionality completed for the parts of the game that were external to the player character, I began converting my game's interface elements to environmental objects to account for VR style gameplay. After these steps, the most crucial and difficult part of the project was converting the player controller to be compatible with VR.

Halfway through the project, I rented an Oculus headset. Upon receiving it, I realized that my computer was not compatible nor powerful enough to support it. In the meantime, I was able to

format the player to work with the VRTK asset package, which forced me to merge my “player move” script with the package’s more complex movement script. I tried formatting the player to work with other headsets through VRTK and testing with other students. However, this proved ineffective since many times my application would not work in VR on their side and I couldn’t easily debug and continually test the app on their system.

To solve this, I enlisted the help of a peer I knew outside of the course who had a Vive headset. Using TeamViewer, I was able to control his screen to program and run tests on the app while he used the headset. This was much easier since it allowed me to spend more time developing. I also really enjoyed creating the demo video. I decided to create an entertaining game trailer that preceded the instructional/overview demo part of the video. I feel that this allowed more personality to be expressed from the game, showing how it would be marketed to gamers. Thinking in a business sense in that way was very interesting to me.