Danny Nsouli
December 10, 2020
CSCI 6907

AR Project Report: Ghost Hunter

1. Introduction

Ghost Hunter is a Halloween themed AR application, in which malicious ghostly spirits exist in our world and the player takes on the role of a ghost hunter to help eliminate them. In this case, the player's phone acts as ecto-goggles, allowing any ordinary mortal to be able to see the ghostly spirits roaming around their surroundings. However, the goggles only work for a limited time before they need to recharge.

Functionally, the game is a first person shooter, in which the player moves their phone camera around their play space to find enemy ghosts to destroy. The challenge comes from having to quickly move the phone in order to find ghosts that are not in view of the player's camera. Each ghost destroyed will reward the player five points, tasking the player to score as many points as possible before the round ends. Each round lasts for fifty seconds before totalling up the user's score on the "game over" screen. The main function of shooting is activated by a button that the player taps with their thumb as they hold the phone.

2. AR Application Design

Firstly, in order to create a more immersive experience, I decided to allow the ghosts/enemies in my game to spawn out of the camera's initial frame/view. AR is unique in that developers are able to take advantage of the flexibility of where the user can point their camera. Normal applications are more restrained in that they are designed with the device screen's boundaries in mind. I also made sure to always keep the enemies at a distance from the player in order to avoid spawning enemies that would overwhelm or block the player's view. User comfort is an aspect that must be taken into account when creating such applications. This is why I also decided not to use screen/post-processing effects and created the game around being in one physical position for safety reasons and to avoid motion sickness.

Since most users aren't regularly dealing with AR applications, the game includes an information screen and was designed to be played in short bursts to encourage easy onboarding. The interface was also designed to be less cluttered in order to not visually obscure the user's camera display and minimal input is required. Simplifying input is a good idea when designing AR games since users will mainly be focusing on the real world around them and will be deterred when too much input is also taking their attention. The button icons are used to help declutter the screen as well as provide familiar UI guidance.

3. Integration

For the build process I decided to tackle some of the more standard features of my game first. These scripts included those of the timer and button functionalities. The button scripts would each take the player to specified scenes. The most vital scripts, however, were the Target, Spawning, and Shoot scripts. The code for these created the basic foundations for the main gameplay. For the spawning script, I created two arrays. One held the spawn points while the other held the enemy models that would be created in the game scene. I then wrote a coroutine that would randomly instantiate one of these ghost models in one of the spawn points. This would happen every two seconds in order to keep the enemies coming for more exciting gameplay. The target script was more simple in that it was attached to every enemy model in order to influence its behaviors in the game scene. Mainly, it would allow the ghosts to float upwards as they spawned at a specified speed. This included vector manipulation for their transforms.

The shoot script was written with a raycast in mind. Whenever the player clicked the shoot button, a "RaycastHit" would be created and the shoot( ) method would decide if an enemy had been destroyed based on an if statement that checked the transform name of the game object. If the conditional statement was satisfied, the method destroys said game object and adds points to the score, additionally altering the score interface text. Lastly, my end screen script would carry over the scored points value from the previous gameplay session in order to display the player's final score in the "game over" screen.

In terms of third party tools, in order to give myself more creative freedom with the art design of the game, I used Canva, an online graphic design platform. Using Canva, I was able to create the simplistic informational/instructional diagram that can be found by clicking the "i" button on the title screen, the "Game Over" text, and the logo of the game on the title screen.

4. User Experience

In order to create an AR application that was interactive in real time, I chose to design the app around the primary activity of a target shooting game. This way, the user is always kept actively interacting with the AR objects in their surroundings. In order to combine the virtual and real worlds, I decided to make the targets/enemies ghosts that would appear in the user's play space. I felt that this added a nice charm to the game because it makes it feel as if the enemies are actually flying around the user's real world environment. 3D animated models were used in order to make the game feel more life-like to foster that sense of immersion. In terms of sound, for each screen I chose unique music to reflect a different tone. The title and information screens have very ominous, spooky music in order to give the user a feeling of unpredictability since they have not seen the ghosts yet. The gameplay level has more upbeat music in order to enfuse more energy into the user as they hurriedly try to shoot as many ghosts as they can for a high

score. And the end screen has quirky and less intense music since the user has completed their interactions with the ghostly threat. There is also a sound as well as a smoke effect triggered for every ghost shot to provide visual and auditory feedback when points are scored.

Furthermore, text was used when necessary to help convey important information to users. This mainly applies to the score counter, countdown timer, and shoot button on the gameplay screen. The logos/text for the title and "game over" screens were made separately in a third party graphic design tool in order to contribute to the game's art direction. All user interface buttons are identifiable by simple icons in order to not run the risk of cluttering screens with text that may not be readable (especially on smaller buttons). The information screen, navigated to from the title screen, was also created using the design tool. This was created in order to allow new players to not feel caught off guard when entering a play session for the first time, by providing a basic peek at what the interface for the gameplay screen looks like with brief text descriptions beside each interface element.

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

Regarding initial design goals, the basic gameplay is fully functional. I was able to find a ghost model pack with prefabs to easily put into scenes with fluid animations. The ghosts were able to appear well through the phone's camera and none of the randomly spawning positions they appeared in were visually obscured by my testing surroundings. The game has good redundant navigation, providing the player access to most of the game's options cohesively from every screen. The raycast used for attacking enemies also allowed for a much more accurate and snappy shooting action, especially with the added help of the on-screen visual indicator/reticle. Background music and sound effects were functional during testing sessions as well. Different music was added to each screen based on the necessary tones. The game also has clear text labels that allow the player to look at their score and remaining time limit during gameplay. The score was able to be transferable to the "game over" screen in order to display the user's achieved points after the end of the previous shooting round.

An issue I had was with trying to get the ghosts to spawn in random locations along a set radius. Originally, I was going to have ghosts appear all around the player, however, whenever I did this, the ghosts appeared correctly but moved along with the camera, which interfered with the feeling of immersion. Also, when at certain spawn points behind the player, some ghosts would appear very dark and far off into the distance. This made it very difficult to see ghosts in specific locations. For this reason, I simplified the idea by making the ghosts randomly spawn in locations only in front of the player and to their right and left sides of the AR camera. For more creative flair, I also think that more sound effects could have been added. Different ghosts could make different sounds as well as fired/missed shots. Additionally, clicking interface buttons on different screens could have also triggered sound effects for auditory feedback.

The speech command function was the main component that I had the most trouble with. I tried using the PingAK9 speech plugin for iOS since most developers who used a Mac and iPhone combination were using it. However, I experienced many issues both in the development environment in Unity and when testing on my mobile device. I followed a recommended tutorial but did not achieve the effect I desired. I planned on having the plugin constantly listen during play sessions in order to shoot whenever the player said "shoot" into the microphone. However, modifying the script from the tutorial to do this did not work in my favor. I then resorted to using the plugin's sample scene in order to check if the plugin even worked on my phone.
This resulted in many build failures in my Xcode application. I then used the "Issues" tab in the GitHub repository for the plugin to see if I could resolve the issue. The troubleshooting tips suggested certain additions of packages through Xcode, which were able to help me at least run the application by getting a successful build. However, even then, I constantly got either a blank grey screen or a frozen interface that I could not interact with on my iPhone. Even when adjusting my phone's privacy microphone settings manually, the application still didn't work as intended. One of the issues I saw said that phones that run off iOS 14 or higher had similar issues that I was experiencing. So, I believe this was the problem in the end that messed with the functionality of the plugin, especially since the video tutorial was created a few months before iOS 14 was released.

6. Suggestions for further improvements.

In terms of improvements, I think that the game would benefit from more complex mechanics. Finite ammunition, for example, would put more stress on the player during the shooting rounds. Having to do a specific gesture on the touch screen or shooting at ammunition packs held by certain enemies would add more challenge to the game. In the same vein, the player could be given a health bar. The ghosts could attack the player, causing them to lose health when they let too many ghosts on screen stay alive for too long. Power-ups could also be used, in the form of floating targets, in order to allow players to have a better chance to reach higher scores. Examples include, clocks to increase a round's time and a gun upgrade that would allow the player to shoot three bullets at once.

Additionally, more enemy variety could be implemented. This includes giving the ghosts diversity in their appearances (3D model structure) as well as their behaviors. Giving the ghosts different movement patterns and even different sound effects to act as voices would help in that respect. Furthermore, boss characters could also be implemented in order to give players the satisfaction of battling against an enemy that takes more hits and is more visually impressive than regular common enemies. On another note, adding difficulty options would also be able to make the game more accessible for players of different skill levels. This would include a longer timer or maybe a larger health meter for the player.

7. Appendix 1: Lessons Learned

This project was much less stressful than the previous project for me. This was mainly because I was more comfortable with the Unity development environment. The experiences I had scripting for my VR game allowed me to become much more efficient in my project development for my AR application. For example, I used many serializable fields in order to make the specifications of certain scripts easier for me to modify during rapid testing sessions. It was also much nicer that we were able to test on more easily accessible hardware. However, since this was my first AR experience, I had some trouble when I tried to complicate my game's design. I tried to include more varied enemy behaviors and attempted to allow the entire play space to be used. However, when spawning ghosts all around the player with different movement patterns, I had some issues that I was not able to resolve such as having some ghosts be visually obscured or following the center of the camera. This forced me to keep my game's internal workings simple and instead focus more on the aesthetics.

For the voice command requirement, while having difficulties with my plugin, I reached out to the T.A. and we worked together on testing the feature through a sample scene and code I had written based on a tutorial. However, we experienced many obstacles as we troubleshooted certain issues. I also noticed that another student with a similar development setup was able to get the plugin to work but had issues with the voice command system shutting down frequently. Even though I could not get the speech command to work, I still believe that my AR application is fully functional despite this missing feature. This was definitely a feature that I felt I was forcing into the gameplay, anyway, since most players would rather use a button than their voice regarding an action that requires constant input. Since my game lacked this feature, I added in an extra information button on the title screen with a simplistic form of the gameplay interface for easier/improved onboarding. I feel that this information component ultimately added more to the entire user experience, since conveying information to the player on how the gameplay works is an important part of game development.

Otherwise, this was a great experience for me since I had never worked with AR applications in the past. This project definitely encouraged me to want to pursue similar projects in the future and possibly improve upon this specific project to see if it could potentially be uploaded onto the Apple App Store.

8. Appendix 2: Feedback to the instructor

This project was a very enjoyable experience. I think the idea of having status updates with expected requirements, however, wasn't as helpful on this second project. Especially, after finishing the minimum requirements, I felt like I had to think of more things for the sake of creating goals for the next update. It also forces the TA to create goals for the students even if the projects are "completed" based on the requirements. I think that it may be helpful to do this for students who are still working on achieving the baseline requirements. However, if a student feels that they have accomplished the basic requirements, I think, at that point it, their goals should be left up to them to decide so that they don't have to worry about making any false or

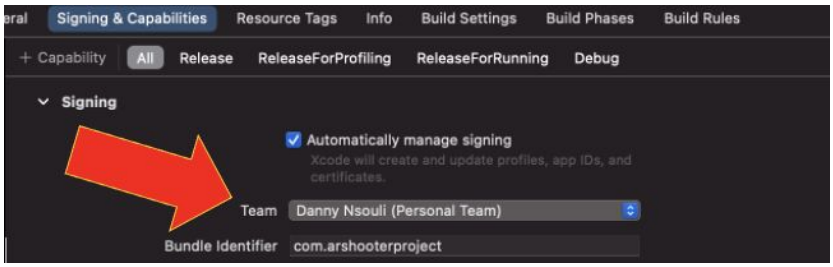
overambitious promises. On another note, I do appreciate the group meetings used to discuss our applications' features. It was helpful to talk to other students to see what has worked for their projects and to offer suggestions to students with similar development platforms.

9. Appendix 3: User Guideline

In order to install this application, you will need an iPhone and a Mac with the Xcode application installed. This application was tested using Xcode Version 12.2.

Firstly, open the build folder and double click on the file, "Unity-iPhone.xcodeproj." This should open Xcode. Once in Xcode, click on "Unity-iPhone" and then click on "Signing and Capabilities" in the top bar.

Once in this window, make sure a "Team" is selected using an apple account.



Next, make sure your iPhone is connected to the computer and that your device is recognized by Xcode. Now click the run button and let the application load onto your mobile device. The application will then open automatically and begin running when ready (also make sure your phone is not locked).