VR + Mobile Game Development

(Independent study Final Paper for CMSC498A)

Andrew Dilks
Fall 2016

Advisor: David Mount

For my last semester at UMD I decided to pursue the course of independent studies to get a better understanding of virtual reality technology and development. Throughout the semester with Professor David Mount's help, I was able to learn the process of VR application development and game dev. I used quite a few online resources such as video tutorials, API documentation, user guides and stack overflow to gather the information I required to learn tools that I used to make VR application development a possibility. Throughout this paper I will be giving you a higher level of understanding and walking you through the process I went through over the semester to create my final application demo that I call "VR Outrun Shooter." Throughout this document I'll be including screenshots of the application taken through my computer. It's important to understand that this is a 2D medium, thus it is not possible for me to represent my created content in its intended form (3D).

Figure 1: An example of what the player would see mid game.

VR Outrun Shooter

At the beginning of the semester I knew I wanted to create a game for the mobile VR environment, I was just not yet sure what I wanted to create. After a few weeks of thinking about what kind of games would work well in this new medium, and what I could create with my limited knowledge, I came across the idea of creating a sort of tabletop style gameplay. I got inspiration from an old game I used to play as a kid called geometry wars.

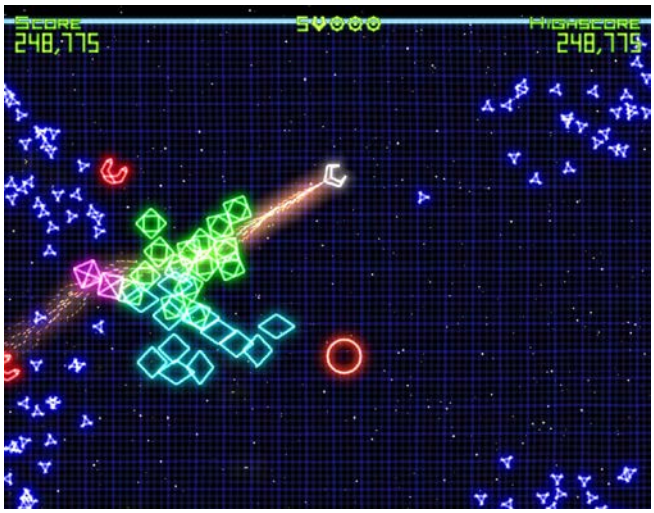

Figure 2: Inspiration, Geometry Wars Xbox 360

Much like that game, the main objective was to control a ship on a 2D plane and destroy the incoming obstacles, scoring the most points, before they destroyed you. What separated my game from this one were, the control scheme, the style, and the medium in which you played and observed the game. I decided to use the head movement tracking provided by the VR headset to steer the ship around the map and the VR headset to give you a sense of presence and a third dimension to achieve a sense of depth throughout the world. Once your ship spawned into the game asteroids would follow. You are to use the gun attached to the ship to shoot at these asteroids before you accidently collided with them, causing the game to end. The user shoots the weapons by tapping on a touchpad located on the right side of the headset.

Now that I have provided a brief understanding of the game/ application itself I can dive into the tools, development and learning process I went through throughout the semester.

Equipment

Just like many software development endeavors it was important for me to narrow down the tools and development target that I had planned to deploy to. My goals for this semester were create a basic game that could have the potential to reach the largest audience in case I wanted to publish my work, all while learning the VR and game development ecosystem. After some research I decided the best development target would be the Samsung Gear VR headset. This headset is one of the cheapest consumer VR headsets on the market and has the lowest barrier to entry considering it operates on technology already owned by the consumer. A cellphone! Now that I knew which platform I was to develop on I purchased the required materials, Gear VR headset, Samsung Galaxy S7, and started reading the documentation provided by the manufactures of these technologies. This specific

headset was created as a partnership with Oculus, the company that created the VR revolution of the 21st century, and Samsung, one of the leading mobile phone manufactures in the world.
Once I began research it was clear to me that I would need to learn how to use Unity, a game development engine compatible with a ton of technologies, including android VR capable devices.

Learning process and documentation

After discovering Unity was the best application for development of my game, I began to dive into their documentation and series of tutorials to get an understanding of how games are made. One of my greatest sources for learning was the tutorials provided on their site. Unity is known for providing an extensive library of tutorials, documentation and videos explaining the application and a lot of game development fundamentals required to make a game.
https://unity3d.com/learn

After going through Roll a ball and Space shooter tutorials, I felt that I had a lot of the basic understanding I needed to start to create my own game. Once I had this knowledge my next step in learning was to understand the difference between normal game development and VR game dev. From here I went to Oculus' site to start downloading the required extensions for Unity and reading up on their getting started guides.
https://developer3.oculus.com/documentation/mobilesdk/latest/

This site gave me some of the knowledge I needed like, Mobile best practices, Mobile application development basics, Native game engine integration and many more.

Next, I began to look up the required files I would need to have in order to build my games for the mobile device and run them on the headset I had purchased. After some reading I found that I needed java development kit installed
http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
the android SDK
https://developer.android.com/studio/index.html
and that I would need to extract my phones Oculus Signature File.
https://dashboard.oculus.com/tools/osig-generator

Setting up the development environment and making all of the parts mentioned work together was one of the largest obstacles I ran into during the semester. There is quite a large barrier to entry here as the learning curve is high and the steps required are numerous. Now that I had all the tools needed to begin development on my game I was ready to work. After reading a few of the Unity VR fundamentals
https://unity3d.com/learn/tutorials/topics/virtual-reality/vr-overview?playlist=22946

I learned about tools like the VR camera, UI for 3D space, and using the controls on the gear VR headset to navigate and control my game. I used my knowledge of the VR Ray cast to allow users of the game to navigate the menus by looking at them and pressing and holding on the touchpad to select the elements. This is very intuitive as you receive visual feedback in the form of the UI element moving closer to you when your gaze is over it, allowing you to understand this object is intractable. Also you receive visual feedback in the form of a progress bar over your reticle, allowing you to understand how long you need to continue to hold down for the object to be selected. I used these techniques in the main scene of my application so the user can navigate to settings, get more info about the game, or look at a leader board before being thrown into the game itself.

Once inside the game I used the players ray cast or gaze (the direction in which the player was looking) to control the player ship. At first this control was difficult because as soon as you looked to a new space on the board the ship would begin moving immediately, which would later cause trouble when it came to aiming the ship to fire at asteroids. I was able to solve this problem by thinking up a solution that involved a buffer zone. This buffer zone was represented as a small circle around the ship that gave the player the ability to aim the ship without yet moving in that direction. Once I implemented this feature I noticed it was much easier to aim the ship and control the overall movement of the ship.

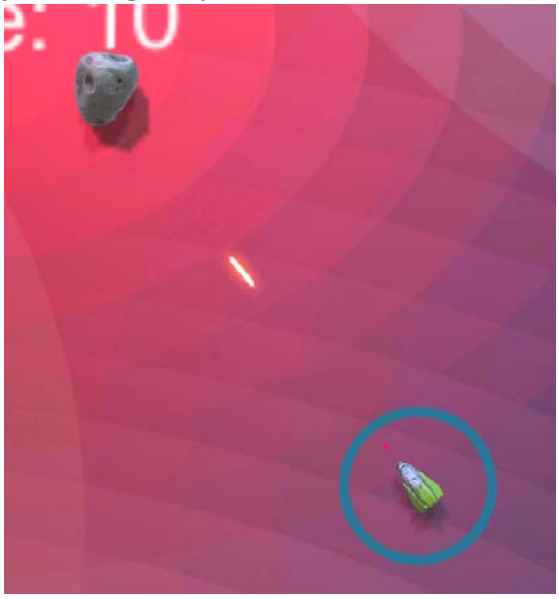

Figure 3: Player aiming and movement

The logic for controlling the ships movement was not easy either. I decided to restrain the ship to movement in the X and Y axis so that the payer ship would appear as if to hover over top of the game board. Using the ray cast library I was able to create a mathematical formula that would take the direction of the players gaze and the location in which it intersected the game board and determine how much the player would need to move on those two axis so that the ship would remain under the players reticle.

After I developed the movement I moved onto the obstacles for the player. I used an asteroid that I found for free asset from the Unity Store as the obstacle my player would be shooting at. I wrote the code so that these obstacles would spawn just above the game board in random locations with a few restraints. I didn't want the asteroids to spawn on top of the player as that would make it impossible to do well,

and I didn't want them to spawn off of the board because then they would not cause a threat to the player and just end up being free points.

Once I wrote the mechanics of the obstacle spawning, I then moved onto the player's defense mechanism, his laser. This part of the project was also particularly difficult to develop because I had to write the logic for the laser movement, spawning location, and collision rules. The collision rules were, that if the spawned laser were to hit an obstacle, it would destroy that obstacle, itself, and create an explosion effect with a sound. If the laser that was shot ended up not hitting anything it would travel indefinitely into space causing a waste of precious calculations on an object that the player could no longer see or care about. To solve this problem I made it so that once the laser reached outside of a set boundary, it would destroy itself. Freeing up resources for other computations.

At this point in the development process I had the basic fundamentals of my game complete. All that remained were the art of the game, the scoring system, and the menu navigation so that the player could go back and fourth between states of the game. One of the features I added at this stage of development was the reset functionality. Once the player died I needed a way for them to return to the menu or restart the game if they wish for another go. To solve this I added a check in the game to determine if the player was still alive, if he/she were not, I would display the score and enable the menu for those two functionalities. Another feature I added at this point was the ability for the user to maintain a High Score and reset it if they like. I achieved this by learning how Unity stores persistent data through game launches. This allowed me to enable a way for the user to track their progress and try to improve through each play through. I had planed on implementing a system for users to see how well they are doing compared to other players of this game but was not able to add this due to time constraints. Another finishing touch I put on the game was a difficulty curve. I made it so that as the user progressed through each stage, the asteroids would spawn faster, causing the game to get harder the longer the user played. I found that this was a very enjoyable feature because without it the user could easily stay alive for hours.

Closing statements

I'd like to thank the CS department at the University of Maryland, and David Mount for giving me this opportunity to learn in an area that I find truly fascinating. I was able to create a working and enjoyable application, all while gaining incite into game development and Virtual Reality technologies that I would not have otherwise received. This is an invaluable gift, as I believe VR could be the next revolution in computing platforms and could change the way in which millions of people around the world consume media and interact with each other.