Will Loughlin
Homework 5 Final Individual Project
Wills Game

My Game's Link: http://willsgame.herokuapp.com/
My Github Project Link: https://github.com/WillLoughlin/WillsGame
My Most Recent Commit:
https://github.com/WillLoughlin/WillsGame/commit/24d9aaf9224a71cd613a9af2ed27f275187a8fe3

This document can be found in the HomeworkX folder in my repo.

Section 1: What I planned on doing for this checkpoint

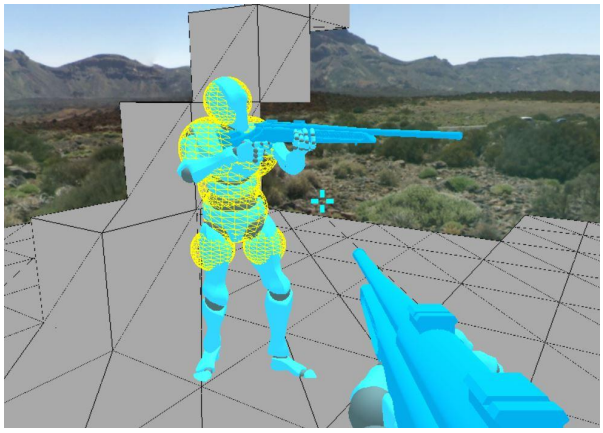
This is the final checkpoint for this project and my game is complete. I had planned to wrap everything up for this final checkpoint, specifically finish the gui and collision detection, make a map with functional spawn points, and create a website for my game. I thought that my collision detection was mostly done for the last checkpoint and I had a gui that was somewhat functional so I didn't plan on spending much time on those first two. I was very wrong about that, but I'm glad I got to test several different implementations of features to see what works best. I'll go into more detail about that in section 2 of what I did for this checkpoint.

As for what I planned to have completed at this point, I expected my game to be done but I hadn't decided on a specific map or gui layout that I liked. I have done a lot of experimenting for this checkpoint to find what I think is the best setup for my game, and I am proud of my result. I expected the map to be the most labor intensive part of this checkpoint, along with putting together an HTML landing page that takes you to the game.

Section 2: What I accomplished for this checkpoint

This checkpoint has been all about finalizing and optimizing my game to get it performing at its best in terms of speed, functionality, and gameplay. I began this checkpoint by completely reworking my collision detection system. At the last checkpoint the server was doing all of the math for every bullet-player collision check which slowed down the performance quite significantly. I ended up scrapping that code and going with a player-side collision system, meaning that all calculations done with players, blocks, and bullets are done within each player's browser. When a collision is detected the player communicates that to the server via socket.io, and the server relays that message to all other players. This allowed me to streamline the server's workload so it can focus on keeping each player updated with positions of all other players and bullets without needing to do any calculations.

The bullet-player collision detection uses a broad phase to minimize calculations, then in the narrow phase it uses 7 points of contact on the players body to determine a collision. If a bullet is more than 2 units from the player's head the broad phase removes it from consideration of collision. If it is within 2 units of a player the distance is calculated at each step to 7 points on the player model which are visualized below.



Since this is a one shot kill game any hits to the legs, feet, forearms, or hands are not recognized. If a bullet is within any of these spheres it is detected as a collision and the player is killed. These points are calculated using a 3 dimensional rotational matrix according to the player's position and direction. The bullet speed is fast, so to avoid jumping through the circles bullets are moved incrementally 20 times for the 2 units that they move in total each frame and collision is checked each time. This ensures that if a bullet's trajectory passes through a sphere it will be detected.

Next I added a block collision detection system so players can't shoot through walls. While I was implementing this part of collision detection I discovered that I couldn't make the map entirely out of 1/1/1 cubes because the three js rendering engine can't hold a steady framerate with anything over 150 cubes, which would make the map very small. After this discovery I rebuilt the player's movement and collision with respect to the blocks to allow any width, height, and depth of block. I also decided to stop using real images as textures because they slow the game down quite a bit and they looked strange on blocks that weren't perfect cubes. I decided to go with a more arcade style color scheme with each block having a single, monotone color and a wireframe to help with depth perception. Allowing different sized blocks made the map creation much

easier, which was what I tackled after implementing the new player-block collision system.

In the Call of Duty Modern Warfare games there is a map called Rust that has been a favorite of my friends and I since we first started playing video games. I decided to model my game's map after Rust because it isn't too large and it has a very recognizable layout that my friends would know by heart without ever playing my game before. I started by experimenting with reading a separate file for the map creation but JavaScript seems to struggle with reading in other files so I ended up hard coding the entire map. It was a very long process which involved first guessing the x,y,z coordinates of each block along with width, depth and height. I would then see where the block would be rendered and adjust my size and coordinates accordingly. There are 90 blocks total in this map, so this process took a while but I am proud of the result. The map is a 100x100 game unit square with a tower in the middle and many obstacles surrounding it. The walls on the edges are low enough to jump over at some points, but the collision detection system will not allow the player to get outside the map area.

After completing the map I moved on to the GUI. I had a working GUI prototype at the last checkpoint but I was having trouble optimizing it. That GUI consisted of creating a different 2d scene within three js that I could overlay onto the three dimensional game scene to use as a HUD. Three js was struggling with rendering both scenes at the same time though, and it made the map disappear sporadically as you moved through it. I ended up scrapping this GUI for a simpler object based approach. My new GUI that I created for this checkpoint consists of blocks placed in front of the camera with HTML canvases used as textures. These blocks use a three dimensional

rotation matrix to ensure that they don't move in the camera's frame of reference, and they are at 80% opacity to allow the player to see his entire fov. The bottom left displays Kills, Deaths, and the number of currently active players. The bottom right shows ammunition, and the top right shows a leaderboard of the top three active players. One downside to this implementation is that when you are close to objects the GUI gets cutoff by things that are closer to the player's camera than the GUI blocks, but this is a minor inconvenience.

After the GUI I implemented an ammunition system to keep players from spamming bullets. Each player is limited to a 7 round magazine with a 3 second reload time. While reloading the crosshairs becomes mostly translucent and there is a progress bar in the bottom right of the screen. If you try to shoot with 0 bullets you will automatically reload but you can also reload by pressing 'r' when you have less than 7 bullets loaded. This limitation helps with game performance because there is less collision detection needed and the gameplay is more fun because it adds more strategy.

Next I tackled the leaderboard system. At each instance of the game loop the server finds the top three players with the most kills and broadcasts this to each player. If there aren't three players or nobody has any kills these spots are just empty, and if a player leaves they are removed from the leaderboard on the server.

The next part I completed for this checkpoint was the landing page for the game when you go to the willsgame.herokuapp.com link. This page allows you to input a name for your character that you can use to identify your friends mid game. It also has some basic information about myself and the game for anyone who doesn't know what it is or how the controls work. If you don't input a name your name is 'anonymous' plus

the first few digits of your players identification number that is randomly generated. The name size is limited to 15 characters to avoid GUI issues.

After the game had working names attached to each player I implemented a notification system to let you know who you are killing or who you are getting killed by. This ended up being one of my favorite additions to this game because when you are playing with multiple people you know who is killing you and who you are killing. I also changed the leaderboard from player id to player name because it's fun to see the name you choose on the leaderboard.

This is a fairly simplified overview of the additions I made for this checkpoint, there are a lot of small additions and tweaks that I made but overall this checkpoint was a ton of fun putting it all together. The game works quite well, it gets pretty laggy with more than 3 or 4 players and can also get laggy if you are connected for a long period of time but that can be fixed by just refreshing the page. I would suggest trying it with a friend or two because it is a lot more fun than just jumping around the map.

Section 3: What I had planned for next deadline, and what plans changed

Instead of what I have planned for the next deadline in this part I'll talk about what I planned on doing but didn't end up getting to. The biggest part that I decided not to implement was an account system. I was planning on using MongoDB to allow people to create usernames and passwords to save progress in the game. I decided that I wanted this game to be more of an arcade type and progress would just be a number of kills, which can be easily cheated by just opening several tabs of the game on your computer and shooting them. I also wanted to have a map in the top left corner,

but the map ended up being small enough where I didn't think that this was necessary. Other than these two things though I got to everything I wanted to accomplish with this project and I learned a lot about JavaScript and HTML.

Section 4: Screenshots of current accomplishments

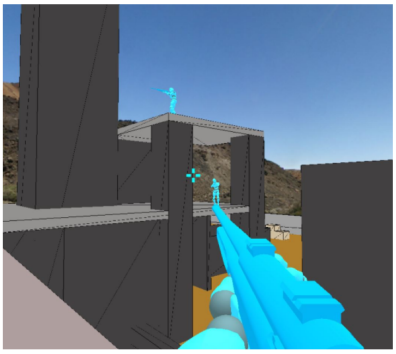
Along with this document I will submit my video presentation so I won't go too into detail with these screenshots but I'll show the basics here.

Here is what you first see on the landing page, the characters are a light blue so I used that color scheme for the website. All you need to do to play is enter a name and hit that button.

About the Game

Will's Game is an online multiplayer first-person shooter built with HTML and JavaScript. It uses Three JS to render the map, Node JS and Socket.io to create a server that allows multiple players on different browsers, and is hosted on Heroku's Cloud Application Platform.

After entering a name and hitting the Play Will's Game button the game will load and you will be spawned at a random location on the map. If you click the screen it will grab your mouse and allow you to look around, to release the mouse hit the 'esc' key. Use WASD to move, space to jump, and shift to sprint. Click the mouse to shoot, you have 7 bullets in each magazine with an unlimited amount of reloads. Hit r to reload which will take three seconds and you will be unable to shoot in that time. Your kills and deaths are displayed in the bottom left along with the number of active players, and your ammunition is displayed in the bottom right. The leaderboard in the top right will show you the top three players among active users by number of kills.

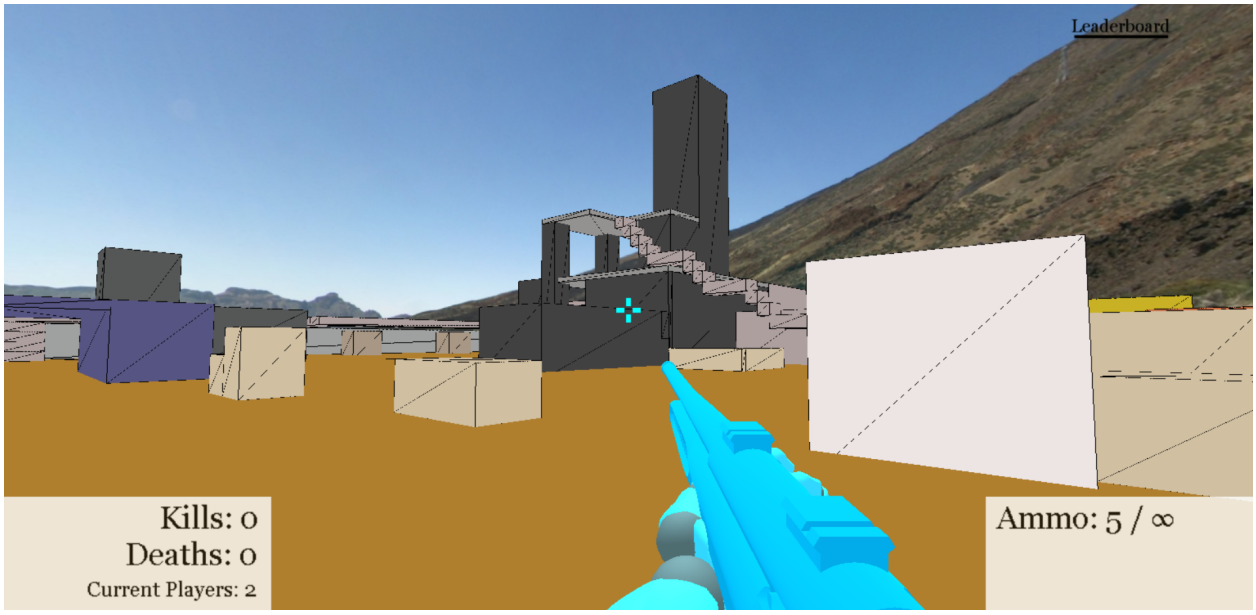


Below the title and game launching button is some simple overview information about the game for people who may not know what it is or how to play.

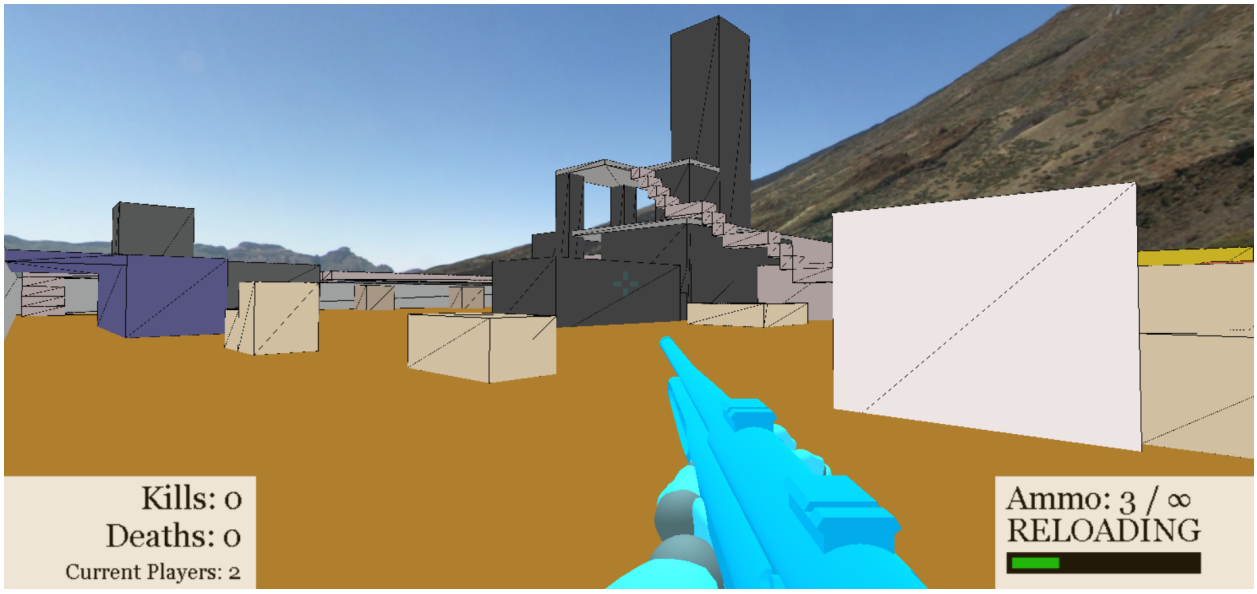
About Will

My name is Will Loughlin, I am pursuing a Bachelor of Science degree in Computer Science at the University of Colorado Boulder. I enjoy fun, simple games that I can play with my buddies, which was the goal of this game. Hopefully you can enjoy this game as much as I did designing, testing, and playing this game with my friends.

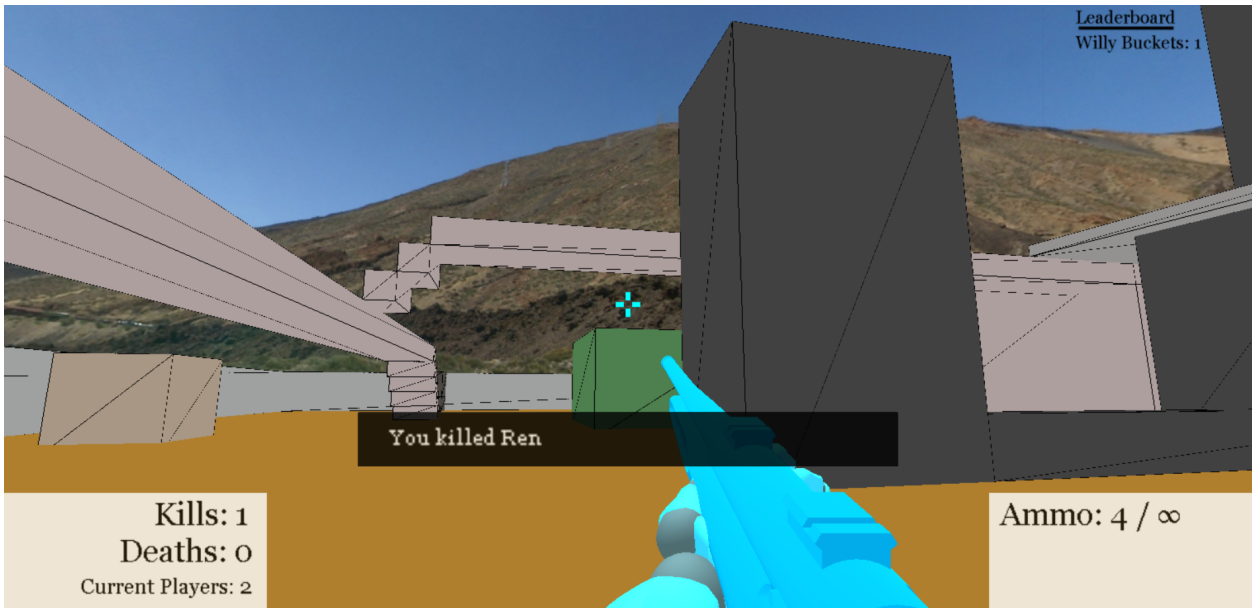
Below that is some basic information about myself and why I decided to go with this kind of game.



Here is a good view of the map with the GUI, you can see the tower in the middle and all of the gui components.



This is the same scene with the reloading animation active, you can see the cursor becomes transparent and there is a reloading progress bar in the bottom right.



When you kill a player or get killed this notification bar pops up for three seconds to let you know who you killed or who killed you. You can also see that the leaderboard has been updated because I got a kill and am now leading the lobby.

This is from the other player's point of view, they now have 2 kills so they moved above me in the leaderboard

That about wraps it up for this project, I'll have much more detailed gameplay in the video presentation. I really enjoyed making this game this semester and I learned a lot about project management, coding techniques, and three dimensional linear algebra.