Will Loughlin
CSCI 3010
Individual Project Proposal: WillsGame

1) What is your project?

My project is an online multiplayer first person shooter game hosted on heroku called WillsGame. The game will be a “free for all” where the players are competing for who can get the most kills in the lobby. Hosting on heroku means that anyone can access the game on a browser with a link from any device without having to download anything. A MongoDB database will allow user accounts to be created to track progress and certain statistics within the game.

2) What technologies are you planning on using?

I will be using Node.js and Socket.io to create a javascript server that will be hosted online via heroku. Github will be used for version control and to host the code. On the client side I will use three.js to render the game in three dimensions. I am completely familiar with Javascript, Node.js, Socket.io, github, and heroku because I used them in a similar way on my group project in software dev last year. We built a simple 2d tank game called WackyTanks where I learned how to implement multiplayer functionality in the browser using node.js, socket.io hosted on heroku. I experimented with three.js over the summer but haven’t built a whole project in it, but I am confident that I can combine what I already know with plenty of available online tutorials and should not have a problem with it. I will also use MongoDB as a database for user accounts that can be created to track progress within the game. I am not as familiar with MongoDB as the other technologies, but I am confident that I can develop a working implementation from online tutorials.

Technology	Familiarity
JavaScript	Completely Familiar
Node.js	Completely Familiar
GitHub	Completely Familiar
Socket.io	Completely Familiar
Heroku	Completely Familiar
Three.js	Becoming More Familiar
MongoDB	Not Very Familiar

3) What are the essential parts of the project?

a. **Heroku**: I need the hosting platform to host my server on the internet so people can play together.
b. **Three.js**: I need three.js to work correctly to render my game on the player side so the player can see what is happening.
c. **Socket.io**: I need socket.io's client-server connection to work correctly for the server to be able to update each client file with current game information.
d. **Node.js**: I need node.js's server functionality to work to create a server on heroku that allows connections from any browser.
e. **Map Functionality**: I need a working map that players can move around on to play the game and compete.
f. **Collision Detection**: I need a good collision detection system because the game is based on hitting other players with bullets without being hit yourself. This is a major part of the game and the math behind it needs to be sound.
g. **Movement and Aiming**: I need a working movement and aiming system so each player can interact effectively with the game.
h. **GUI**: I need a working user interface so the user can select game options and login to an account. The GUI will also show the user where they are aiming with small crosshairs.
i. **Account Creation and Stat Tracking**: I need a working user account creation system so users can be recognized by their accounts and their progress in the game can be recognized.
j. **MongoDB**: I will need MongoDB to work to store the data about usernames, passwords, and stats. I will need to be able to access this info on MongoDB for the users to login.
k. **Artwork**: I will need 3d models for the players and bullets, and images for the walls of the game. I will find this art on open source game art websites like OpenGameArt and I will be sure to credit my sources.\
l. **Leaderboard**: I will need a way to track who is winning currently in the game. This will be available to all players as part of the GUI.

4) What outside resources do you require?

The outside resources that I require will just deal with the rendering of the game. I need art for the map, players, bullets, and guns. I am going for a simple design, so I will only need a few images that can be repeated on cubes for the map which I will find on OpenGameArt. On the website cgtrader there are thousands of free 3d models for players, bullets, and guns that I can choose from. I have not decided on the exact models yet but there are several on cgtrader that would work great. The only other outside resources that I will need are people to test my game with me, which my roommates are happy to help with.

5) Architecture Proposal

A prototype of my game that I created to show the implementation of my concepts that can be found at https://github.com/WillLoughlin/WillsGame. This repo is connected to heroku and the game it runs can be tested at http://willsgame.herokuapp.com/. Click the screen to look around (esc key will release the mouse), WASD to move around. You can open the link from multiple tabs at the same time to see the multiplayer in action. As of now all that you can do is move and look around.

The front end of my game will be the GUI and controls rendered by three.js to show the user what is happening in the game. This is handled by the PlayerCode.js file in the client folder of my project. The contents of the client folder are sent to each player when they join the game. The PlayerCode.js file handles player movement and relates the information back to the server, while the server sends current information of all other players back to the PlayerCode.js file for three.js to render.

The back end of my game is the server found in the app.js file in the main folder of my project. It has a game loop at the bottom that sends x,y,z coordinates and looking directions of all players to every player in the game. When a player joins they are sent map information which will be updated if map or gamemode changes. The server will handle bullet collision with players and alert each client file when a player dies. The front end and back end interact using node.js and socket.io to send information from the

server file to the client file. Information can be sent both ways, and both will be constantly updated by the game loop in the server file.

Objects in my game are used to store information about the map, the players, and bullets. There will be a parent object called “Thing” which will store x,y,x information and direction information. The “Player”, “Block”, and “Bullet” objects will be children of this “Thing” object. The Player object will store additional information about name, amount of kills, amount of deaths, ammo, and more. It will also have all Player related functions that the game uses to operate (like collision). The Block object will store additional information about images used to render the block and its size. The Bullet object will store information about speed, damage, and model used to render. Getters and setters for x,y,z and direction information will be inherited from the Thing object.

The database used to store user account information will be very simple and only consist of one table. The table will have a row for username, password, kills, and deaths. It will be used to sign in users and display a small emblem that I will design corresponding to their number of kills.

The testing framework will use mathematical situations of players and bullets in x,y,z coordinates to test if the code accurately detects collision and responds accordingly. This framework will also be used to test player collision with other players and walls.

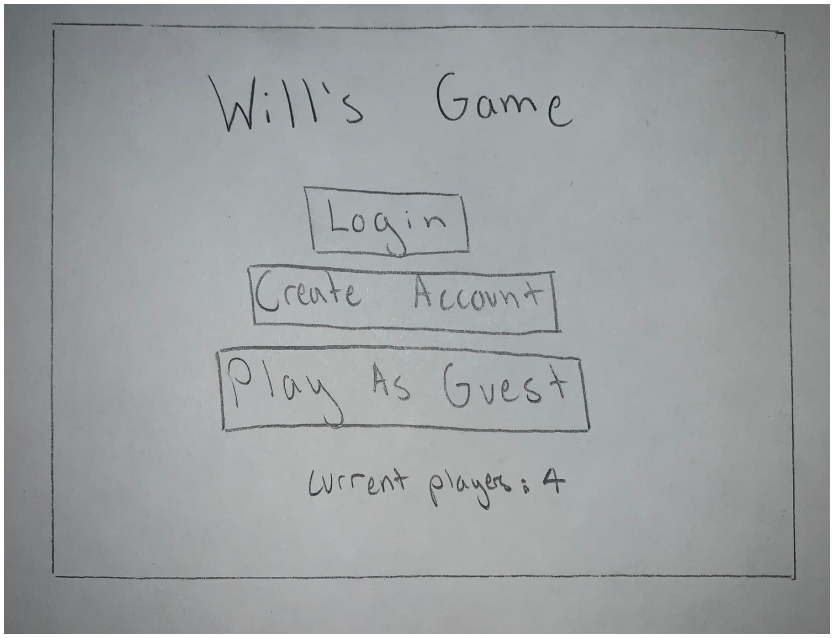
For my Design Patterns I will use the **Flyweight** and the **Factory** design patterns. I will use the **flyweight** design pattern to streamline the rendering of the map on the client side. The map will be composed of blocks, with several images that repeat many times. I will design the block object in a way that the image is not stored with every instance of the block, but rather each block has a pointer that points to the image location. Three.js supports instanced rendering, which will allow me to send a block of common data that will be rendered many times along with a long list of x,y,z block coordinates where the images will be placed. I will use the **factory** design pattern to simplify the creation of objects in my game by implementing a factory method that streamlines the creation of objects. This will be used to create the player, block, and bullet objects that make up the game.

The GUI will be rendered by three.js and will show the user current health, ammo, kills and death information. It will also assist with aiming weapons in the game, and will be similar to the gui in games like Counter-Strike. This will all be handled in the PlayerCode.js file, and will take advantage of the three.js tools that allow easy implementation of user interfaces. There will be a menu when a user joins to login or play as a guest, and there will be an options and player stats menu that can be displayed on the screen.

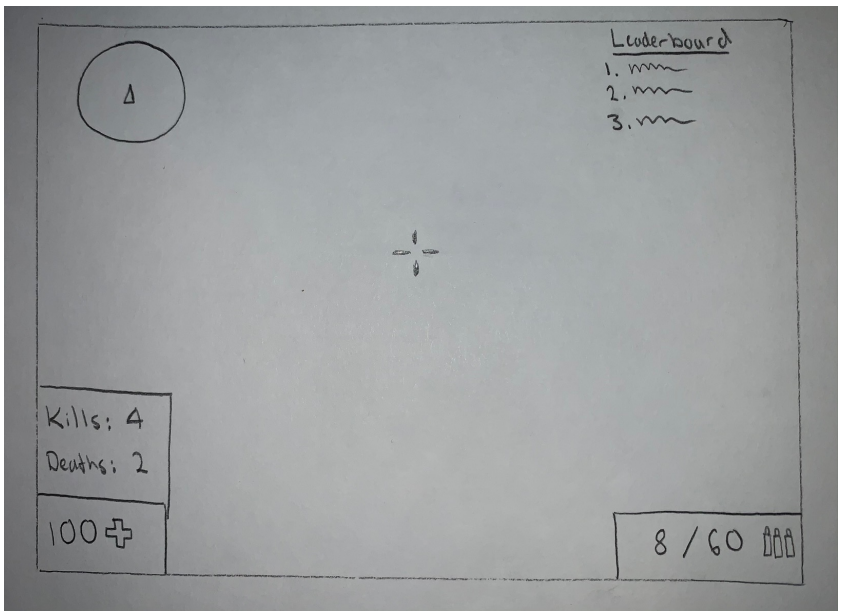
The technologies that I will use fit together to create a server with multiple players receiving constant updates about the game's state. Node.js is used to create the server, and Socket.io is used for communication between the server and the client. Heroku is used to host the server on the internet for easy and free access. The code will be hosted on Github, which connects directly to Heroku so you can see exactly what is used to create the server. MongoDB will be used to save and access account information, and will be accessed by the server. Three.js is used on the player side to render the game with information it receives from Socket.io.

6) GUI User Interface Proposal

When a player joins http://willsgame.herokuapp.com/ they will be greeted by a main menu with options to login, create an account, or play as a guest. The accounts will just be a username and a password used to track kills and deaths. A current player count will display the number of currently active players. Here is an example of what the menu will look like:



Once a player has selected an option and joined the game the GUI will display a map in the upper left. The kills, deaths, and health of the player will be in the lower left and the ammo count will be in the bottom right. A leaderboard will be in the top right showing who currently has the most kills in the game. A crosshair will be in the middle of the screen to help with aiming. An example in game GUI looks like this:



When the user clicks on the screen the mouse is grabbed and used for aiming. When the esc button is pressed the mouse is released and an options menu will open

allowing the user to change certain settings within the game and see a current in game leaderboard.

7) Detailed Plan

Due Date 1: Homework 2, Feb 28th (5 days)

a. For this due date I will turn in the basic code for my project with inheritance implemented for the players and blocks objects. Right now I have a simple version of my game working, for this deadline I will not be modifying the output of what I have now but I will be changing the organization of code by adding object inheritance.
b. I will need knowledge about using objects in javascript and how to implement inheritance.
c. From that knowledge I have a lot of experience in using objects in javascript, but I will need to learn about inheritance. I know how to use inheritance in c++ and I know that it is possible in javascript, I just need to learn the correct formatting.

Due Date 2: Homework 3, March 12th (12 days)

a. For this due date I will complete a functional map in the game with player-wall collision. I will also finish player movement. At this point each player will be able to move freely throughout the map without clipping through the floor or the walls. I will also implement a testing system to make sure the player-wall collision detection system is working correctly mathematically. The map will be rendered using the flyweight design pattern to make it more efficient.
b. The knowledge that I need for this due date will be about how to implement simple collision detection mathematically, how to create objects in three.js, how to use instanced rendering to implement the flyweight design pattern, and how to design a testing framework for my code.
c. I will need to learn how to use instanced rendering in three.js, and I will need to refresh on how to create a testing framework. I already know how to create objects in three.js and how to implement simple collision detection mathematically.

Due Date 3: Homework 4, April 6th (25 days)

a. For this due date I will add player models and movement animations to the game. I will also finish the weapons with shooting capability. Collision detection with bullets will not be implemented yet, but the players will be visible characters and they will be able to see and fire their weapons. This will require adding the Bullet object and server-player communication about when bullets are fired, their position, and direction. Since we have a long time to complete this deadline, I will also add account creation and database management to the game at this deadline. I will create the main menu with the database to allow users to create and login to accounts (or play as guests).
b. I will need knowledge about objects in javascript, communication via Socket.io, 3d models, and animations in three.js. I will also need to know how to use and communicate with MongoDB to implement account creation and use. For the menu I will need to know about html landing pages before it brings you into the game.
c. I will need to learn about using 3d models and animation in three.js. I will also need to learn about communicating with MongoDB to create and access accounts. I am very familiar with using objects in javascript and communication via Socket.io and will not need to learn about that. I also have experience with creating main menus from my last game project in software dev.

Due Date 4: Homework 5 Checkpoint, April 29th (23 days)

a. For this due date I will implement player-bullet collision. This will allow me to add kill and death tracking with respawning within the game. With the 3d collision I will add a testing framework to test 3d collision cases. I will also add the GUI with a minimap, a leaderboard, and health. Kills and deaths will be saved to the MongoDB database. The settings and leaderboard menu will be added for this due date which will conclude the main parts of the game.

b. I will need knowledge about complex 3d collision detection, creating a GUI in three.js, and editing a database in MongoDB. I will also need to know about creating a testing framework for the 3d collision.
c. I will need to learn more about the math behind 3d collision and how to create functional menus in three.js. I will already know how to edit a database and create a testing framework from the previous checkpoints.

Due Date 5: Homework 5 Final, May 2nd (3 days)

a. For this due date I will test my game and fix any bugs I can find. The main parts of the game were completed by the last checkpoint, and this will just be small adjustments and tweaks. The game will be a final product by this due date.
b. I will need general knowledge of the project and how each technology communicates with the others.
c. I will not need to learn much for this deadline, just use previous knowledge from the previous due dates.

8) How will you stay engaged?

I will stay engaged in this class by coming to lecture and completing each daily assignment and using what I learn in class on my project and for the PEs. The programming exercises will ensure that I stay focused on everything we are learning about because I need to know the stuff we go over for each PE. I may not be using c++, but javascript is a similar language with many of the same capabilities and I will be creating my own user interface in three.js. In my project I will be implementing almost everything that will focus on in this class. My objects will use inheritance, I will have a testing framework, I am using git with continuous integration, I will be implementing the factory and flyweight design patterns, I will use user testing with my roommates, and I will implement a GUI. Since I am using many of the topics that we will be discussing in this class for something that I am truly interested in I am confident that I will stay engaged in this course.