"no excuses." Design Document

100390459 – Seb Hall

October 20, 2025

1 Overview of "no excuses."

"no excuses." is the fairest multiplayer game ever made. It's a fast-paced online 1v1 FPS game where the only focus is shooting the other player in the head. Players are placed into an arena where the obstacles in front of them are randomized every time, giving you no advantage in learning layouts. The whole environment is a pristine white, so you can focus on nothing but shooting the other player.

Other online First-Person Shooter games are full of distractions. The enemies in **"Counter-strike"** blend right into the background, meanwhile the enemies in **"Valorant"** are surrounded by bright and distracting colours. It's an easy excuse to make as to why you missed that shot. Furthermore, both games named previously are Team-based shooters, giving you something else to blame as to why you're losing. What if you had a game where you couldn't make any such excuses? Where the only reason you lose is because you're worse at the game. You have no excuses. Competition at its core.

Players are placed at either end of an arena and have to find and kill each other. Who ever kills the other person first earns a point. To keep the game fast paced, the game works in a Best of 5 format. First to three points wins.

1.1 Genre

The genre of this game is an Online Multiplayer First-Person Shooter. The majority of games in this genre are team-based, in a 5v5 or 6v6 format, or sometimes a free-for-all "deathmatch" format. This game is instead a 1v1 game, featuring only you and your enemy.

1.2 Target Audience

I would give this game an age rating of 18. The plan is to include the player characters exploding and their blood splattering across the white walls like paint on a canvas. This should make it more satisfying and momentous for when a player kills the opponent, even if it does up the player rating. The game may seem boring to players below 18 anyway, since they might not be interested in

such a simple format which strips away the stimulation of bright colours that typically appeals to them.

1.3 Unique Selling Point

The unique selling point of my game is that it's a fair environment for people to face off against each other in 1v1 aim duels. People who play multiplayer FPS games love trying to prove they're better than one another. You'll often hear the phrase "1v1 me" in team multiplayer games such as **"Valorant"** despite the game not at all being designed around that. This can be extended to gamers in general, who talk about "settling it in smash" meaning 1v1ing each other in a game of **Super Smash Bros**. I want my game to be the fairest and most equal environment for a 1v1 FPS duel, hence the white background and coloured players.

2 Background

These are some games that helped to inspire the making of this game.

2.1 Game 1: Quake

Quake is a First Person Shooter released in 1996 by id Software. The game had both an expansive single-player campaign and multiplayer servers for people to face off against each other in deathmatches. The movement of **Quake** was always very fast and fluid, and the player could build up a lot of momentum. One of the most popular multiplayer modes was "Instagib", where every player had a railgun, and everyone could die in just one hit. This made for a chaotic mode of people trying to dodge and weave lasers while waiting for their railgun to recharge.

The elements of **Quake** that will be implemented into this game are the fast-paced and fluid movement, and the instant killing of enemy players found in the Instagib mode.

2.2 Game 2: Call of Duty Warzone

Call of Duty Warzone is an online multiplayer FPS Battle Royale game. The main part of COD Warzone is its large-world battle royale, where players fight to be the last one standing. The way they differentiated themselves from other battle royales, though, is by having the Gulag. The Gulag is a 1v1 arena you and another player are placed into when you die in the battle royale. The arena is small and rectangular, with several obstacles placed between you and the other player. Your job is to kill the other player in order to be resurrected and rejoin the battle royale world. When the mode was released in 2020, players were only allowed a pistol or revolver of some kind. Nowadays, though, they have full rifles and shotguns, which gives the player too much power in these

fights. In addition, there is skill expression in learning the maps/layouts of the Gulag, of which there are several.

The elements that will be taken from **Call of Duty Warzone**'s Gulag for this game are the enclosed 1v1 fights and the feeling of powerlessness within your weapons – having only a pistol rather than a rifle.

2.3 Game 3: SUPERHOT

SUPERHOT is a single-player FPS game where time only moves when you move. The game was developed by a small team as a Game Jam game in 2013 before being turned into a full game in 2016. It has levels where the player is placed into a perfectly white environment and is instructed to kill bright red enemies shooting at them. According to **SUPERHOT**'s Art Director, Marcin Surma, this art style was only chosen because it was easy to make and allowed the developers to focus on the gameplay, since this was a Game Jam game and they didn't have much time. After their decision to turn it into a full game, though, they kept the art style as it made it immediately clear what the player needed to focus on in the game, removing all distractions. This art style is a big inspiration for this project, as it aligns with the goal of removing all distractions from the player so they can focus on landing their shot.

The elements of **SUPERHOT** that will be implemented into this art style are its ultra-minimalistic art style and its focus on highlighting enemies in a primary color.

3 Summary

3.1 Game Story/Objective

Since this is a multiplayer game about shooting opponents, there isn't really a clear story to be told since it's not at all grounded in reality. The focus is entirely on the gameplay. Some multiplayer games have "lore" as to why the characters are here and fighting, like **Valorant**, which every time they have a new multiplayer season, has a cinematic video to go with it giving context as to why there's a new character or a new map. For this game, it's so abstract as it is that it's not necessary to give context via story.

Instead, we can talk about the objective of the game. To win, the player has to kill the opponent three times before the opponent kills them. The overall objective, though, is to be the best at the game. Players will be assigned a score based on how many games they win or lose, similar to a rank in other online games. This score will be used in matchmaking to make sure players play against opponents of similar skill levels. When playing against a friend, the player will be able to view their current record of wins and losses against the enemy, for example, 2-6.

3.2 Player Character

The player characters, like the world, will be designed to be minimalistic. They should stand out easily from the background while still being visually distinct. Since the head is the only part of the character that can be shot, the head should be a different color from the body to make that clearer. My design makes the character black with a bright blue head. I chose blue for several reasons. The first is that while red is more distinct, the game **SUPERHOT** (see background) already used red for their enemies, and I don't want to copy their aesthetic entirely. The next reason is that blue is a color seen by all types of color-blindness. Having Deuteranopia or Protanopia types of color-blindness means that the color red looks more like a strange yellow color. All three types of color-blindness can see blue, though. It may look a little different between the types, but it's visually distinct regardless. Therefore, color-blindness is not an excuse in this game.

Figure 1: Basic concept art drawing of the player character in the arena

I increased the size of the head as well since it's the only part of the character that matters and therefore should be more prominent. It does make the game easier since being larger makes it easier to hit, but the game is already difficult since the head is typically very small.

In this drawing, the character's gun is orange. This was purely to make it clear that it isn't part of the main body. This colour isn't decided on and may change as the game takes shape.

3.3 Aesthetic

This game is designed to be as minimalist as possible. The idea of the game is to give the player as few distractions as possible, so they can focus on aiming and killing the opponent. Therefore, the game will have completely white backgrounds and obstacles. This way, the opposing player should be the only focus on the screen when playing. The game will still be detailed, in that there will be a variety of interesting obstacles for the player to play around.

Lots of other multiplayer games either try to make the game as realistic-looking as possible or as colorful and vibrant as possible. By instead leaning into

this minimalist aesthetic, this game will stand out from the crowd as something truly different.

3.4 Level Design

For level design, I have decided to create a circular arena where the obstacles are randomized each time. My concept art showed placing players at a random angle around the arena, but since it is circular and obstacles will be randomized upon each death, there will be no difference in changing the angle.

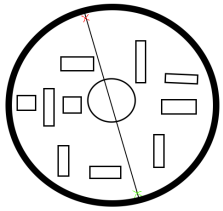

Figure 2: Basic concept art of the arena and the obstacles in it.

Since the layout changes every time, there will be a top-down view of the arena after each death for players to look at while they get ready for the next round. Once each player has accepted, the game will start.

The initial design for the arena's obstacles is a series of boxes, walls, and bends. These should hopefully give players plenty to play around and strategize about. The only downside of having this randomized layout is that it trivializes real level design. It gives off the idea that people designing maps and levels in FPS games can be replaced by randomly placing blocks around. This is not the case. These levels will be less fun than if they were hand-designed by a real person. But this is about artistic vision rather than lack of competence. Having randomized obstacles ensures competitive integrity by making sure that players cannot learn the layouts of the map or understand good angles that players always forget to check, making it more even. This is definitely something to test in the prototype and see how it is for players to play.

3.5 Controls

This game will be played with a mouse and keyboard. Aiming with a controller still isn't as precise as aiming with a mouse, and therefore mouse and keyboard is the absolute best way to play FPS games. The player will use WASD to move around and left-click to fire. Moving the mouse will move the camera. Right-click could be used for something such as a dash or instead to zoom in and aim down sights. This could either speed up or slow down the pace of the game depending on which one is implemented. The worry is that implementing

a dash may make the game too fast and change the game to become more about out-maneuvering your enemies rather than out-shooting them. The function of the right-click will be implemented further in development after different play styles are tested.

That being said, the movement of the game will be fast-paced. This ensures the rounds can take place quickly or slowly if desired. With slow and more tactical movement, it forces the players to play at a slower speed, perhaps holding angles for players to walk into them. Having faster and more satisfying movement inspires players to instead try different ways of pushing players for a kill, making for more interesting gameplay.

A lot of tactical shooters tend to implement "shooting error" into their games. This means that the player's bullets veer off into random directions if they are moving and shooting at the same time. This means that the players have to stand still or shoot; otherwise, their bullets will miss. This is yet another form of skill expression. Players now have to strafe, meaning to press one direction, then another direction, and shoot in the time standing still between changing directions. This technique does limit the amount of movement possible, since both players need to strafe; otherwise, they'll be hit. In addition, it increases the amount of random chance in the game, since the players could have "shooting error," and one of these random bullets could hit the player, causing the player who disobeyed shooting error to win. This is unfair, and in the pursuit to make the fairest game possible, this will not be included.

The game will also use hitscan bullet detection. This means that instead of a bullet being a physical projectile that has to collide with an enemy, as seen in **Overwatch** or **Marvel Rivals**, instead, the player has to line their crosshair up with the enemy's head and click to kill them, and the projectile is purely cosmetic. This is more precise as it allows the player to solely focus on hitting the enemy player's head rather than having to calculate travel distance if a player is moving. This is a technique seen in more tactical shooters such as **Valorant** or **Counter-Strike**.

3.6 Engine Choice – Unity

Unity is the choice of Game Engine for this project. This is because it has a wide array of tools for online functionality. In addition, since the graphics of the game are so simplistic, there is no need for the graphics capabilities of Unreal Engine, which should allow the game to run on lower-end computers.

The primary reason for picking Unity though is that I am familiar with the engine and understand C#. This is a great benefit, as it allows the game to be developed faster since there is a much smaller learning curve compared to starting fresh with an entirely new engine such as Unreal Engine.

Unity offers many different types of networking formats and has support for lots of hosting services. The formats for networking include Fishnet, Netcode for Game Objects, Mirror, and Photon. The hosting services include Unity Relay and Lobby, Edgegap, Epic Online Services, and more.

I want this game to be Peer-to-Peer (P2P) networked, which means one player hosts the server while another player joins. While this may seem like the host will always have an advantage due to being closer to the server, modern P2P solutions have solved this by having a middle-man server which both players connect to in order to transfer data to one another. Examples of this are Unity Relay, Epic NAT P2P, and Valve Steamworks.

I haven't decided which of these to use yet, since the prototype made only includes local networking currently. I will test the different services in development and then decide on which is a better fit for the game.

4 Prototype

Before talking about the two elements implemented so far, it's important to understand the state of this prototype. Developing a prototype like this is the best way to test out whether features work or if the game is even fun. Tons of features and aspects of the game talked about above will change once a full prototype is finished and the game is more stable to test things out. Currently, the game is a basic version of a few features.

4.1 Element 1: Movement, Shooting, and Level Design

4.1.1 Movement

The movement shown here looks completely different from the movement described in the design document above. This is because having the character move was one of the first features implemented, and at the time, the plan for the movement was to be slower-paced and more tactical. Once testing this out in-game, though, it felt unsatisfying compared to the size of the arena. In order to make this slower type of movement work, features like shooting error are required in order to give the game enough complexity. So after implementing it, the decision was made to shift toward faster and more fluid movement. Regardless, the character currently can move with WASD and do a small jump with space.

4.1.2 Shooting

Shooting has been implemented into the game using hitscan. It works by sending a raycast from the middle of the screen (where the crosshair is) straight forward until it hits something. If it hits the opponent in the head, a win is triggered, and the player gains a point. Currently, it works but occasionally fails to register a hit, so some debugging is required there.

4.1.3 Level Design

A basic white level has been implemented so far. There is a circular arena surrounding both players, who are placed on a Terrain GameObject for now.

There is a pillar in the center to stop the players from immediately seeing each other when they spawn in. I have also implemented the randomized obstacles. Currently, I have a Bend, a Box, and a Wall that can spawn anywhere within a certain area. There definitely are issues, such as objects spawning in the same places or completely blocking off the players from each other, but right now, it works well.

4.2 Element 2: Networking

The prototype also includes basic networking features. Currently, the networking only works for local players, since it hasn't been fully implemented, and the chosen server hosting service is still being decided on.

The network solution chosen for this project at this moment is Fishnet. We started this project by using Unity's Netcode for GameObjects but quickly found there weren't many resources online about it, and that most Unity developers working on multiplayer games nowadays were using Fishnet due to its greater feature set and better performance.

Having not worked on a networked game before, this was a learning experience to overcome, since Fishnet does not allow communication from server directly to client. Not unless you set the clients to become Observers of course, which allows communication between all other observers.

The network solution still isn't perfect at the moment. Despite using the best networking solution in Fishnet, using such a powerful tool is still a skill to understand. For example, occasionally the wrong client location is reported to the other client, causing one client to appear where they aren't. Implementing a feature such as client-side prediction might help to fix this and decrease latency, since the client moves locally and the server, instead of checking and then making the move, predicts where the client will move and rolls the client back if it makes an error. This would help the game be as smooth as possible, which is the goal for a competitive game such as this.

5 GitHub

I've used GitHub for this project to be able to store files and log progress. GitHub is such a useful tool for a project such as this. If, for example, Fishnet networking doesn't work for this project, reverting the project back to before it was implemented is incredibly easy thanks to GitHub.

This is the link to the GitHub project: here

6 Video Demo

This is the link to the Video Demo: here