# Design Doc for CA 2

Zach Ahearn

K00282339

10/04/2024

A video game screen with a group of people

Description automatically generated

Figure 1 - The start of the game

**INTRODUCTION**

My game is a top-down zombie survival shooter game. The game has a player and an increasing number of zombies. The player is in control of their own player object which they can move around by pressing down the ‘w’, ‘a’, ’s’, and ’d’ key to go up, left, down, and right on the game map. The player is also in control of a gun which they can shoot by clicking the left mouse button. The goal of the game is to make it through the waves of zombies while also completing objectives in order to make it to the final boss fight at the end. My game is different because I have added objectives to my game which come as an extra step in order to complete a wave. I also have a boss fight on the final wave. This also means I changed my game to not be endless, and instead, have a final goal to work towards.

**The Changes**

**A screen shot of a computer program

Description automatically generated**

Figure 2: upgradeHealth() class changed to be more effective

The first of the changes I made to my game is in the upgradeHealth() function in my player.cpp file. Originally, the m\_MaxHealth variable was increased by multiplying the START\_HEALTH by .2. This meant there was a 20% health increase when the upgrade was applied. I changed this to .66 in order to make it a 66% health increase. I did this for two reasons; I wanted the health increase to be very significant and rewarding to receive, and I changed how the upgrade system works which I was account for later in this document.

A computer screen with text and numbers

Description automatically generated

Figure 3: zombie.cpp file. Adding a new zombie type.

In my zombie.cpp file, I analysed how the zombies were added to the base code, From there, I learned how to add a new zombie type. This new zombie represents my boss zombie which has much higher health and speed compared to other zombies.

A computer screen with green and white text

Description automatically generated

Figure 4: new methods in zombie.cpp

I added new methods in my zombie.cpp file. Two of these new methods, setScale and setSpeed on line 147 and 153, are used to control the size and speed of the zombies. These are used during specific waves to act as the object for that wave, which is just to survive. My getPositionVector() function is used to get the position of a zombie as a vector rather than a float rect.



Figure 5: createHorde.cpp: Passing in the wave variable.

Going into my createHorde.cpp file, I had to pass in the wave variable from the zombieArena.cpp file. This was easy to do as createHorde.cpp includes the .h file for zombieArena so it was a simple as adding ‘int wave’ to the createHorde object in this file as well as in zombieArena.h.

A screen shot of a computer

Description automatically generated

Figure 6: Handling condition for boss spawning

The reason I did this is to control exactly when my boss zombie should spawn. The ‘type’ variable works by picking a random number from 0 to 2 which translates to the switch statement for the zombie types in zombie.cpp. To force only the boss zombie to be added to the ‘zombies’ array, I made the type automatically equal 3 when the wave is equal to seven.

A computer screen shot of a program

Description automatically generated

Figure 7: zombieArena.cpp: Making new variables.

Now looking at my zombieArena.cpp file, I have added many new variables revolving around the objectives I have implemented into my game. Figure 7 shows the initialization of the sprites with the logic written later in the code.

A screen shot of a computer program

Description automatically generated

Figure 8: UPGRADES!

My next change comes from my upgrade system. Originally, the player had the choice of what upgrade they wanted after each round. I have now changed this so that depending on the wave just completed, a predetermined upgrade is granted. An example on line 524 shows that when the wave is equal to two, an upgrade named ‘Bandolier Bandit’ is given which doubles the clip size of the gun.

A screen shot of a computer

Description automatically generated

Figure 9: Boss spawn condition

Figure 9 shows how I handle only one boss zombie to spawn. On line 611, when the wave is equal to 7, the number of zombies that are added to the ‘zombies’ array is equal to one. Otherwise, the number of zombies works as normal with it being 5 zombies multiplied by the wave count.

A screen shot of a computer program

Description automatically generated

Figure 10: Objective 2

Figure 10 shows the logic behind my first real objective. There are four buttons which the player must walk over to activate in order to progress the wave. It checks if the player has overlapped a button which changes the corresponding Boolean variable to true and changes its previously red, deactivated texture to a green activated texture. Once all booleans are true, the main objective variable is set to true. Finally, once all zombies are killed and the objective is true, the game state is changed to the ‘LEVELING\_UP’ state which ends the wave.

A screenshot of a computer program

Description automatically generated

Figure 11: Objective 3 & 4

Moving on, figure 11 shows my objective three and four. Objective 3 works similar to a pickup that doesn’t despawn. The player must collect the pickup five time in order to complete the objective. The pickup works that every time the player overlaps it, its position changed to a random x and y value from the width and height of the arena. Objective 4 works that there is a capture point which the player must stand on in order to fill a time bar. When the player fills the time bar, the objective is completed.

A screenshot of a computer program

Description automatically generated

Figure 12: Objective 5 & 6

Objective 5 works simply that when the wave is equal to 5, I call my setScale() function I made which will half the size of the zombies. I reset the once the wave is completed. This same simplistic approach is done for wave 6 too in that I change all zombies’ speeds to be equal to 100, which makes them much faster than normal. Again, I reset this once the wave is over.

A screenshot of a computer program

Description automatically generated

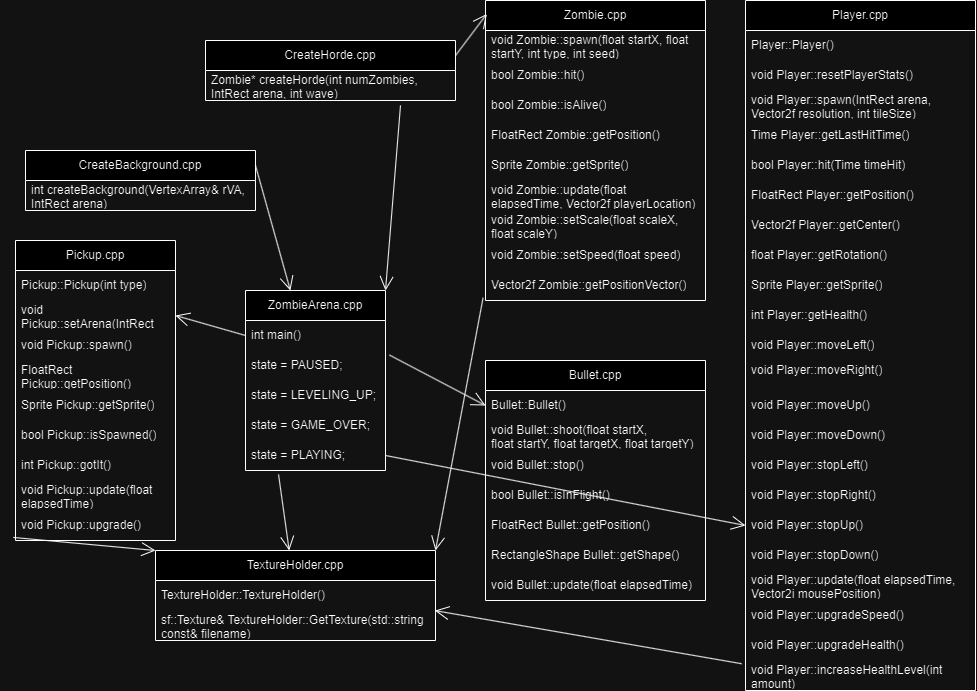
Figure 13: Final Objective 7

Onto my objective seven, my final objective is my boss fight. For the boss’ sprite, I have two textures which I switch between. I have a timer, which, every .25 seconds, 1 is added to the animationCounter variable. When the animationCounter is an even number, one texture is set. When it is an odd number, it is set to the other texture. This gives the boss a changing animation. In order to put the boss sprite onto the actual boss zombie, I use my previously mentioned getPositionVector() function. Using this function allows me to set the position of the sprite to that of the zombie. This was done to give the boss a separate hitbox to that of its texture.

A computer screen shot of a program code

Description automatically generated

Figure 14: Drawing orders

Lastly, by making it so my objectives are drawn only during a certain wave, they are removed once the wave ends. I also have my objectives drawn before everything else like my player and zombies so that they aren’t covered by the object are always visible.

**UML Diagram**

A screenshot of a computer

Description automatically generated

Figure 15: UML Diagram

**CONCLUSION**

I had a lot of fun completing this project. I was happy I was able to successfully implement all the ideas I had for the game, and I am happy with how they turned out. I did have some minor difficulty with the project just with the placement of variables and objects not working as intended but I was able to overcome these. The only remaining issue with my code is that it sometimes crashes due to a memory issue. I believe it happens based on the amount of time playing the game that is crashes and not any specific wave that causes it. It also only happens sometimes and not all the time.