Top Down Zombie Shooter “Barricade”

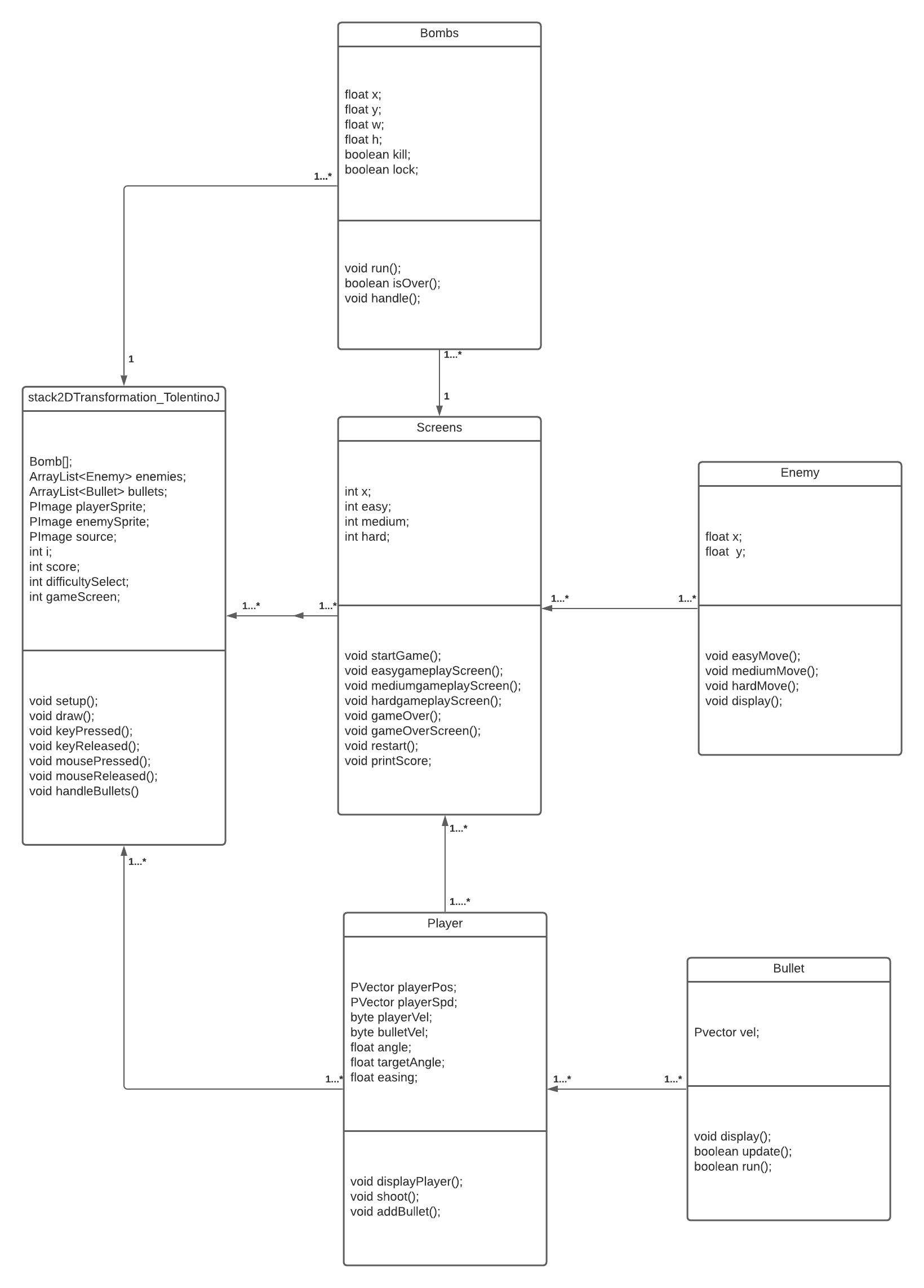
|  | Josiah Tolentino  Kwantlen Polytechnic University  Surrey, Canada  Josiah.tolentino@email.kpu.ca |  |
| --- | --- | --- |

# ABSTRACT

## Barricade is a top down shooter developed with processing 3. The projects implements processing designs as well as Java methods to develop the environment.

## Inspired by games like the last stand, alien swarm, and asteroids, the goal of the game is to hold off waves of zombies and hold the line. The zombies spawn randomly and the game ends when a zombie crosses the player through the bottom of the screen. From there they can see their score and try again on the same or different difficulty.

# INTRODUCTION

 Figure 1: Class Diagram of Barricade

Barricade utilizes multiple arrays, objects and functions in the game. The game has multiple child classes that sprout out from the main class and the “screens” class that are used to display the player and enemy classes and associations.

Barricades main class “stack2Tranformation” sets up the game as the main parent class by initializing the multiple arrays and calling on most of the 6 classes for further utilization in the game. The screen class contains the various states of the game, the pushes of the lever with the bomb class, sprite development for player and enemies, and removing bullets that go off screen.

The Screens class contains most of the user interfaces including buttons, various game states like start, difficulties, game over, and restarting, implementation of the other classes within their respective difficulties, and further difficulty mechanics of spawn rate, speed, and dynamite clear screens for the later difficulties. It initializes the player class in their movement, display and shooting and the bomb array to create the dynamite for the medium and hard difficulties.

The player class has player movement and creation with associations with the spawning of bullets. Having limitations to the player moving left to right with the “A” and “D” keys. This class also implements player rotation for the player sprite to point their gun where the mouse is pointed. It also calls on the bullet class to create bullets relative to where the players position is.

In association to the Player class, the Bullet class gets the location and velocity number, creates the bullets shape, size, and color, varies its speed, and returns the bullet to be fired.

Within the Enemy class contains variables for the multiple difficulties and creation of the enemy sprites to be called upon when the player selects a difficulty and that difficulty screen calls upon this classes spawning values..

And the Bomb class initializes and sets the screen clearing mechanics of the dynamite for the player to use. What is intended is there to be one dynamite in the medium difficulty and one in the hard difficulty. The player would need to manually push down the lever to the bottom of the screen before the zombies reach the end and that would clear the screen of enemies.

# Algorithm Examples

Barricades uses multiple methods which checks the game state and what happens which will be demonstrated.

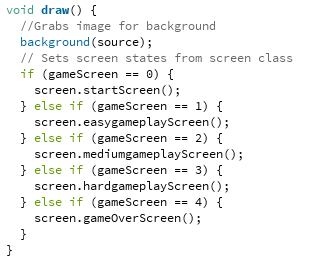


Figure 2: Main Menu Selection

When the player starts the game, the stage variable is set to 0. Through decisions made in game and the results of play the state will change. For example the state changes depending on the difficulty and when its game over.

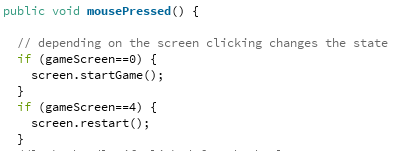


Figure 3: Change screen state with click

When the player clicks at the start of the game and at the game over screen specifically it invokes the screen class and changes the state of the game.

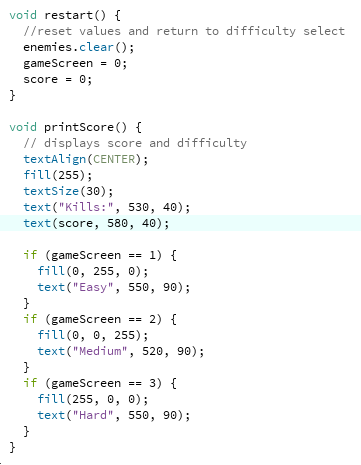


Figure 4: Game states changes

The gameScreen method is used further during difficulty changes that also effect the user interface. It is also reset when the restart method is invoked and forces the screen back to the start.

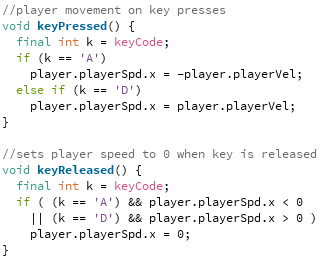


Figure 5: Player movement

Called on the main screen, pressing “A” and “D” will set player speed and velocity positively or negatively on the X axis to move the character left and right.

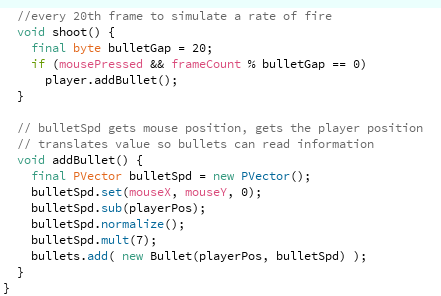


Figure 6: Player shooting

Both shoot and addBullet methods are called in the various gameplay screens. The rate of fire of the bullets is set by dividing the frame count by the bulletGap, having a bullet spawn ever 20 frames.

The addBullet method utilizes the player position PVector and the normalized bulletSpd data to set the creation of bullets.

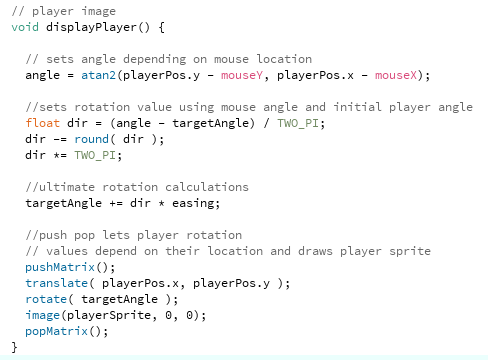


Figure 7: Display player and rotation

Display player doubles for setting the rotation of the character sprite with push pop matrix. Grabbing the angle from the player position and mouse position, it is translated with rotations via PI and uses a percentile with easing to allow the player sprite to rotate.



Figure 8: Gameplay Screen

The medium difficulty gameplay screen calls from player, bomb, bullets, and enemies classes. The enemies arraylist in particular is adjusted via the difficulty movement and the rate of how much the enemies are spawned with frame count of 30 frames. Having a bullet’s dimensions get close to an enemies width removes the zombie from the array and increases the score by 1 which is displayed on the user interface.

# constraints

The bomb implement doesn’t fully work, however the generation of the dynamite and pushing of the lever does work but not the spawn limitations and screen clearing effects. The detection of the lever rectangle object going down would not register and thus does not work. The character rotation is also not perfect as the bullets fire out of a certain point off the sprite and not out the rifle barrel itself.

The project implements multiple array lists for the enemy and bullet creation while a static array for the creation of the dynamite. The game can be further developed to be more complex with features like a barricade, grenades which would be a different area of effect attack, a wall of debris and cover for the zombies to destroy and more.

Other features I had to scrap were grenades that would destroy zombies in a area of effect, rockets that would shoot from the player sprite and explode, and a barrier for the zombies to break through with a life bar that diminishes the longer a zombie is on the barrier and the more zombies there are.

# conclusion

Constructing this game came with multiple issues that needed to be addressed whenever I wanted to implement a feature. It was a challenge where each feature will not work or break the game at some point, but also a great opportunity to learn and utilize the various lessons from class. I am happy that the game mostly works and that I can show my work mostly complete.

# REFERENCES

1. Processing: Multiple bullets in top down shooter. https://stackoverflow.com/questions/52045493/processing-multiple-bullets-in-top-down-shooter
2. Oguz Gelal. (2015). Ultimate Guide to the Processing Language Part II: Building a Simple Game. https://www.toptal.com/game/ultimate-guide-to-processing-simple-game
3. Shoot multiple bullets. - Processing Forum. (2013). https://forum.processing.org/one/topic/shoot-multiple-bullets.html
4. Handles \ Examples \ Processing.org. (2021). https://processing.org/examples/handles.html
5. rileygombart. (2015, March 10). Animated Top Down Survivor Player https://opengameart.org/content/animated-top-down-survivor-player
6. rileygombart. (2015, March 6). Animated Top Down Zombie. https://opengameart.org/content/animated-top-down-zombie
7. 12x12 Hills Battlemap by alecm on DeviantArt. (2020, March 20). https://www.deviantart.com/alecm/art/12x12-Hills-Battlemap-834513774