Shape Invaders

Esha Sarfraz, Ernesto Reyes, & Daniel Ramierez

Application Explanation

The application we chose to create is a rudimentary implementation of a popular 2D video game known as Space Invaders. The premise of Space Invaders is that the player character, a spaceship with three lives only able to move left and right, is out in space behind a few shields shooting bullets at rows of alien invaders. These alien invaders move in a side to side motion, slowly making their way down whilst shooting down at the spaceship.

However, our implementation of this game has a few less features than the original game. For instance, we didn't incorporate the shields provided in the original game. Instead the player character has to dodge the bullets. Additionally, we didn't provide actual images to our alien invaders and player characters, instead opting to use various shapes and colors to represent each element of the game.

Yet, despite all this, we did accomplish many similarities to the original game. The player character is able to move solely in a left to right motion and is able to shoot bullets, all of which is controlled by the user. Additionally, there are three different enemies, each that provide their own set of points once shot down by the player. Finally, we retained the three lives integration along with the score feature that shows up with each enemy taken down.

Significant Features

Each group member played a significant contribution in the development of this application.

Esha Sarfraz was responsible for the primary game logic along with the implementation of the user controlled player and the swarm class. She focused on incorporating basic player movement, player health, player score, and the swarm (a matrix of enemies) movement, and the swarm death.

Ernesto Reyes was responsible for the implementation of the enemy class, polyEnemy class, and the swarm class. He focused on including basic enemy movement in the enemy class, enabling the swarm matrix to work with polymorphed classes, the destruction/redisplacement of the swarm, how the swarm interacts with the player when it kills them, as well as a restart game functionality

Daniel Ramirez was responsible for the bullet class, enemy bullet class, and the implementation of player death.

Object-Oriented Programming Concepts

The player.h and player.cpp file cover many of the programming concepts covered in this course. Essentially, these files define a player class, which creates a player object capable of left-right movement. The player class consists of position variables, dimension variables, a constructor, three keyboard response functions, a mutator function, an accessor function, and a destructor. The constructor serves as a way to define the dimensions and position of the player object. The destructor serves as a way to take away a player's life and end the game once the player's lives are all used up. And finally the mutator function and accessor function allows us to change the players position in real time, pointing to the current position the user is in. The bullet.h, bullet.cpp, enemybullet.h and enemybullet.cpp files also cover some basic class definitions similarly to player.h and player.cpp.

In addition to this the enemy.h and enemy.cpp files touch on the same things as the player class, with the exception that the enemies are all placed into a matrix in a swarm.cpp and swarm.h file rather than being individually drawn. The enemies are also unable to be controlled by the player as they move of their own accord. Another thing that differentiates the enemy class from the player class is the use of inheritance and polymorphism. Inheritance is utilized in the polyEnemy.h and polyEnemy.cpp files, where we allowed each type of enemy to utilize the same function as the original enemy class. However, these files also incorporate polymorphism, as they take the virtual void display function from the primary enemy class and are manipulated in their respective classes to represent differently colored shapes.

Additional Comments

Esha Sarfraz: Overall I found this project very beneficial in my understanding of implementing new unfamiliar libraries. There was a multitude of research I had to do beyond simply asking others for assistance. I wish there was some more guidance on this project as I found myself stuck for a long period of time just trying to set up OpenGL itself on VSCode.

Ernesto Reyes: I thought this project was extremely necessary in allowing us to use what we learned in class and incorporate it in a fun and intuitive way. I struggled at times, but ultimately I'm satisfied with the end result. Hopefully, my group and I can fine tune it even after the due date.

Daniel Ramirez: This project really heightened my understanding of OOP and OpenGL, and made me see that even the simplest of applications can be a trial to create. However, it only made me appreciate even small-scale developers that much more, and only reinforced my career goals of becoming a game developer. Despite it being very hard at times, seeing it all come together in the final product was immensely satisfying.