Liam Ollive

Final Write Up

Abstract

This is a simple space shooter created using Java. The player is able to control a spaceship to shoot at incoming enemies, and collect a power up once they reach a score of 550 points. A boss enemy appears after the player encounters a certain number of waves of enemies. The Boss shoots at the player and moves on a set course, after the boss is defeated with a certain number of shots by the player, the game will be completed.

Introduction

I wanted to work on a space shooter because I am interested in game design/game programming. To start, I followed a tutorial to help me learn and understand the basics of a main game loop, setting up a JFrame, displaying images/graphics, object movement and collisions. This was helpful to me for working on future projects and my understanding of Java. On top of the tutorial I added and adjusted parts of the game as I want it to be designed. For example I added in an upgrade: once the player shoots at the upgrade the player can then shoot two objects at once. I also added in a score, a help screen with game control information, and a game over screen for when the player wins or loses. I added in a boss that moves and fires projectiles at the player. I also created my own sprites for the game in PhotoShop.

Detailed System Description

So, the main Game class is really where most of the game is controlled, though most of the objects are created in the Controller class and collisions between objects come from the physics class and the interfaces which most objects have a different entity interface. The game starts on the menu screen and the player can click on the play, help or quit button. The player actually clicks between certain coordinates instead of an actual button, this is in the MouseInput class. The game starts with one enemy, Once the first enemy is defeated it adds one more to be spawned in the next wave. This continues until the enemy\_count reaches a certain amount and the boss spawns. This is done in the Game class where enemy\_count goes up if enemy\_killed is greater than or equal to enemy\_count. And when the bullet fired from the player object hits the enemy it increases enemy\_killed by 1. For collisions the Player class implements EntityA and the Enemy class implements EntityB, so when the bounds of the Player object intersect with the bounds of the Enemy object it changes the state of the game to the GameOver state and displays the score. Objects like the Player, Enemy, Bullet, Upgrade, and Boss extend the GameObject class to have x, y and a getBounds method. Also while playing, if the player presses the spacebar the bullet is created from the player object and the bullet moves towards the top of the screen. The bullets can collide with an invisible wall at the top to destroy them so they do not infinitely fly off the screen and lag the game. If the bullet object hits into an enemy object, the collision between their implemented interfaces, it is set in the Enemy class that the enemy\_killed int goes up by one and the score goes up by 10. The collision also destroys both the bullet and enemy objects. Once the player reaches a score of 550 the upgrade moves on to the screen so if the bullet object collides with the upgrade it changes the bullet to missile. This makes two bullets spawn from the sides of player when they hit space instead of one and changes the image displayed as the bullet/missile. After a certain number of enemies have appeared, the boss spawns and makes it so the enemy\_count is too high to spawn any more enemies, as the enemy spawn count has a limit so the counter to increase enemy\_count does not spawn too many enemies. The boss moves on a set track using a series of else if statements based on the boss’ current x and y position. The boss also shoots out a fire object in four different directions. Once the player collides with one it causes the player health to decrease, as the fire objects are implementing the same interface as the Enemy class. The boss also has a health bar, to decrease the health bar the player has to shoot at the boss, once the boss heath is zero or below it changes the state of the game. This displays congratulations on the game over screen and displays the score. The player can quit from here.

UML of main Game Class

Game

--------------------------------------

-serialVersionUID:long

+width: int

+height: int

+scale: int

+title: String

-running: boolean

-thread: Thread

-image: BufferedImage

-sprite: BufferedImage

bullet: BufferedImage

-enemy: BufferedImage

-upgradeA: BufferedImage

missile: BufferedImage

-boss: BufferedImage

-fire: BufferedImage

exist:boolean

f1:boolean

f2:boolean

f3:boolean

f4:boolean

-up: int

-test1: int

enemy\_count: int

enemy\_killed: int

-p: Player

-c: Controller

-menu: Menu

-help: Help

-over: GameOver

-b: Boss

+ea: LinkedList

+eb: LinkedList

+ec: LinkedList

+ee: LinkedList

+HEALTH: int

+score: int

+bossHealth: int

+State: STATE

-------------------------------------

+getEnemy\_count(): int

+setEnemy\_count(enemy\_count: int): void

+getEnemy\_killed(): int

+setEnemy\_killed(enemy\_killed: int): void

+STATE:enum

+init(): void

-start(): void

-stop(): void

+run(): void

-tick(): void

-render(): void

+keyPressed(e: KeyEvent):void

+keyReleased(e: KeyEvent):void

+getSprite(): BufferedImage

+getBullet(): BufferedImage

+getEnemy(): BufferedImage

+getUpgradeA(): BufferedImage

+getBoss(): BufferedImage

+getFire(): BufferedImage

UML of Controller class

Controller

---------------------------------

-ea: LinkedList

-eb: LinkedList

-ec: LinkedList

-ed: LinkedList

-ee: LinkedList

enta: EntityA

entb: EntityB

entc: EntityC

entd: EntityD

ente: EntityE

enemy: BufferedImage

upgradeA: BufferedImage

boss: BufferedImage

r: Random

- game: Game

-----------------------------------

+Controller(game: Game, enemy: BufferedImage, upgradeA: BufferedImage)

+createEnemy(enemy\_count: int): void

+createUpgrade(up: int): void

+createTest(test1: int): void

+tick(): void

+render(g: Graphics): void

+addEntity(block: EntityA): void

+removeEntity(block: EntityA): void

+addEntity(block: EntityB): void

+removeEntity(block: EntityB): void

+addEntity(block: EntityC): void

+removeEntity(block: EntityC): void

+addEntity(block: EntityD): void

+removeEntity(block: EntityD): void

+addEntity(block: EntityE): void

+removeEntity(block: EntityE): void

+getEntityA(): LinkedList

+getEntityB(): LinkedList

+getEntityE(): LinkedList

UML of GameObject Class

GameObject

---------------------------

+x: double

+y: double

--------------------------

+GameObject(x: double, y: double)

+getBounds(width: int, height: int): Rectangle

UML of Player Class

Player

-------------------------

-velX: double

-velY: double

-player: BufferedImage

game: Game

controller: Controller

--------------------------------

+Player(x: double, y: double, game: Game, controller: Controller)

+tick(): void

+getBounds(): Rectangle

+render(g: Graphics): void

+getX(): double

+getY(): double

+setX(x: double): void

+setY(y: double): void

+setVelX(x: double): void

+setVelY(y: double): void

UML of Physics Class

Physics

--------------------------------

--------------------------------

+Collision(enta: EntityA, entb: EntityB)

+Collision(entb: EntityB, enta: EntityA)

+Collision(entd: EntityD, enta: EntityA)

+Collision3(entc: EntityC, enta: EntityA)

+Collision(ente: EntityE, enta: EntityA)

+Collision(enta: EntityA, ente: EntityA)

Requirements

Since I was developing a videogame my main goal was to make sure that it is functional, it may not be addressing a particular issue, but it is a form of entertainment. It has a score and the boss at the end is meant to be difficult to defeat, so the game is meant to offer a challenge to the player, and offers replayability with the score.

Literature Survey

I’ve used the main design off of classic space shooter video games like Galaga, Asteroids and Space Invaders. These were classic arcade games with similar themes of a player controlling a spaceships and fighting against enemies or asteroids. Like those earlier games I made this game with simple controls.

User Manual

Once the program is running the user just has to select the play button to begin. The player can also select the help button which gives info on the simple controls. From there the player can click on the quit button to quit or click on the play button to play. To move the spaceship: use the arrow keys to move in their respective directions. To fire at enemies: hit the spacebar.

Conclusion

Overall, since this was really my first large project I’ve worked on in Java alone, it was definitely a challenge for me. I spent a fair amount of time this semester working on this project, not only trying to learn the new code and techniques from the tutorial and online, but also refining the game and making it my own. I am glad I was able to create something both fun and challenging.

References

This is the tutorial I have used to help me make the game

<https://www.youtube.com/playlist?list=PLWms45O3n--6KCNAEETGiVTEFvnqA7qCi>