CSC 413 Term Project Documentation

Fall 2019

Jair Gonzalez

915762580

CSC413.03

Tank Game:

https://github.com/csc413-03-fall2019/csc413-secondgame-jairgonzalez

Super Rainbow Reef Game:

https://github.com/csc413-03-fall2019/csc413-secondgame-jairgonzalez

Introduction on both games

Hello Computer Science Colleagues, this documentation aims to cover all essential parts of the final term project for CSC413 at San Francisco State University. The composed of two major pieces of software that aims to practice good object-oriented programming. Both games take the form of games, one game titled Tank Game, and the other Super Rainbow Reef Game. I will be covering them both separately.

Tank Game: This project consisted in building a multiplayer based tank war game. The goal behind this game consisted of destroying the enemy tank game by roaming around, collecting power ups and shooting each other. In the game, there are breakable walls that can be destroyed after shooting the same block twice, and there are unbreakable walls that contribute to making the boundaries of the game. The screen of the game is composed of a split screen and a mini map that help the players navigate throughout the map. Tanks can take 10 bullet shots until they lose a life, and all tanks have only 2 lives. For the player on the right the controls are W.A.S.D for movement, and to shoot it is the letter Q. For the player on the right the controls are the arrows keys and to shoot it is the “ENTER” key.

Super Rainbow Reef Game: This project is very similar to a popular game called brick breaker. In this game the goal is to hit bricks to get the highest score. The game is set up with a star that that has the same physics as a ball, and the sea creature that the player can control. The sea creature can only move left and right, and if the sea creature fails to bounce the ball back up, the player loses a life.

Development Environment

Version of Java used: 11.0.3

IDE used: JetBrains IntelliJ

Other: No other special libraries or dependencies were used to composed this software

How to build and run game for each game

First step to be able to run this game is by cloning the repository you can do this by doing using the terminal. Here are the commands.

Tank game: git clone <https://github.com/csc413-03-fall2019/csc413-tankgame-jairgonzalez>

SRRG: git clone <https://github.com/csc413-03-fall2019/csc413-tankgame-jairgonzalez>

Following that you will have to open the files using IntelliJ, using default settings for importing the files, and using JDK 11.0.3. Then the game should be able to be executed using the play button within Intellij.

To build a Jar for either game you would have to go under file on the top left, click project structure and then Artifacts. There should be a Plus button (+), click on Jar , then click on select from modules and dependencies on the drop down. Everything should autofill, then you can click configure, the JAR file will be stored in the out file it generates.

Controls

Tank Game:

Player 1 controls: Movement = W (UP), S (DOWN), D (RIGHT) A (LEFT), Q (SHOOT)

Player 2 controls: Movement = Arrows keys + SHOOT (ENTER KEY)

SRRG:

Player controls = Left arrow key and Right arrow key.

Implementation discussion (Classes): Tank Game

BreakableWall: Holds the attributes that make a standard wall breakable by using a health system that starts at a 100 when it has not been hit by a bullet. Whenever the wall is hit the health goes down, and when it hits zero the block is deleted from the map.

Bullet: This class holds the physics behind the bullet including elements such as move rate and if it hits something is toggles the collision. Also sets the owner of the bullet so a player cant be hit by their own bullets.

CollisionHandler: This class controls all the collisions in this game that have to do either the bullet or the tanks themselves. This class toggles when a bullet hits either a wall or another tank.

GameObject: This class holds all the variables that are used to determine the rate of change in in the tanks and bullets.

GameWorld: The Main Class of this software that loads all the resources including the map design and window size.In addition, it Manages the game state for the game which are menu, game, and exit. then Houses the physics behind how the tanks move within the game.

Menu: Game State that controls the start of the game or the exit.

MouseReader: Event listener for the mouse controls in the game, which is only used in the menu.

Powerup: holds the attributes of the power ups, which are speed boost and extra health points. Also loads the resources for the power ups to be used in the game objects class.

Tank: Holds all the elements of the tanks including what to do when a control key is pressed and the health’s of each tank player. Also holds the orientation of where the bullet should shoot from.

Tank Control: this class contains the controls for both players of the game

Wall: This is the unbreakable wall, which also houses the boundaries of where to draw in the game.

Implementation discussion (Classes): Super Rainbow Reef Game

Application: This is the main class of the game, It loads all the resources, loads the paint components, controls collisions and sets the window size. Also updates the game anytime that objects are moved in the game

Brick: Contains the breakable brick aspects which include the positioning and points system.

Collidable: Abstract class that sets the places the x and y coordinates of the bricks that can be broken. Also load resources

GameEvents: creates the objects for when a key is pressed. It implements the class observables

Shell: Contains the key events for when a key is pressed which is then passed onto the game events class

ShellControl: Holds the controls to the shell

Star: draws the star and sets a movement on it to move freely. The star bounces off of bricks and walls.

Implementation discussion (Classes): Shared between both games

So within both games there were about 4 classes that were reused from the tank game, and reinvented to create the Super rainbow reef game.

GameWorld vs Application: Both classes share the same initializing factors of each game. They both load recourses, set the controls, set the view of the game, and add objects to load the game.

Bullet vs Star: Both classes extend to their collision handler and have their own positioning.

Tank vs Shell: Both classes extend to collisions, and also have there own functions for the controls.

TankControl vs Shell Control: Both classes have the event listeners for when keys are pressed and share two common controls which are left and right arrow for movement moving either left or right.

Diagrams

Tank Game Class diagram

A screenshot of a computer

Description automatically generated

Super Rainbow Reef Game

A circuit board

Description automatically generated

Project Reflection / Conclusion

In my honest opinion this term project was not an easy task to complete, I think that a big part of the reason would contribute to the high level of Object-Oriented Programming skills required to simply just plan out how to structure these game, and the other big factor is working through the mechanics / physics behind building this game, luckily most of that kind of code was given to us from the beginning. The process when working on either game was essentially the same, from the beginning I worked with a very general structure on how I thought the game would be structured, going through the game I started to incorporate more and more details as feature was being built. Although that the process seemed simple, I think that something that really slowed me down was working with UI, event listeners, and some build in java libraries that I wasn’t used to. Another big obstacle was working on collisions, the idea of being able to build a rectangle for the tank and having it interact with other objects in the game was one of the things that took the longest to fully develop. Nevertheless this project was good practice for our object oriented programming skills and I will look further into working on other projects including high level of OOP.