CSC 413 Project Documentation

Summer 2020

Naweeda

920462358

CSC0413-01

https://github.com/csc413-01-SU2020/csc413-tankgame-Naweeda.git

TABLE OF CONTENTS

Introduction

The intent of this term project is implementing a 2D java game. Which bases on the requirements given. This project includes resources and various functionalities following OOP principles.

a) **Project Overview**
The main purpose of this game is a good practice of OOP aspects and coming out with a neat and a proper design.

b) **Introduction of the Tank game (general idea)**
The goal of this project is designing a Tank game which consists of two players, breakable walls/unbreakable walls, a mini map, a split screen, power up, collision, and shooting. It should have been implemented in Java. The most important part of this project is considering OOP principles and rules and designing the classes in a very conceptual way. Designing a creative class diagrams is very crucial for the implementation of this game. However, I have changed my class diagrams so many times until I reached to a highly inventive design.

Development Environment

a) **Version of Java Used**
Java 13.0

b) **IDE Used**
IntelliJ IDE

c) **Any special libraries used or special resources and where you got them from.**
I have cited them at the end.

Building the Game

a) **Explain how to import and/or build the game.**

- Go to my repository link.
- Then click the green button which is called “Clone or Download”.
- After, click Download Zip.
- Then, extract the zipped folder.
- After loading the Ide, click import project.
- Then select “Create project from existing sources” and enter next.
- Then keep entering next until you get to the end, and press finish.

b) Steps taken to build the jar.

- Click file on the IntelliJ Ide and select project structure.
- The project structure must have marked correctly like sources and resources.
- Next, select Jar from the Artifacts bar.
- Then, select the main class, and apply and hit ok.
- Lastly build artifacts, right click, and run the jar.

c) Commands for running the build jar

- Use PowerShell or your terminal.
- Type java jar .\GameWindows.jar and enter.

Running the game

- For running the game, right click on the GameWorld and select "GameWindows.main".
- Right click on the Jar, show on explorer, then double click to run the game.

The rules and controls of the game

Player1 controls:

- UP key: Move Forwards
- LEFT key: Rotate Left
- DOWN key: Move Backwards
- RIGHT key: Rotate Right
- ENTER key: Shoot

Player2 controls:

- W: Move Forwards
- A: Rotate left
- S: Move Backwards
- Z: Rotate right
- SpaceBar: Shoot

How to play the game:

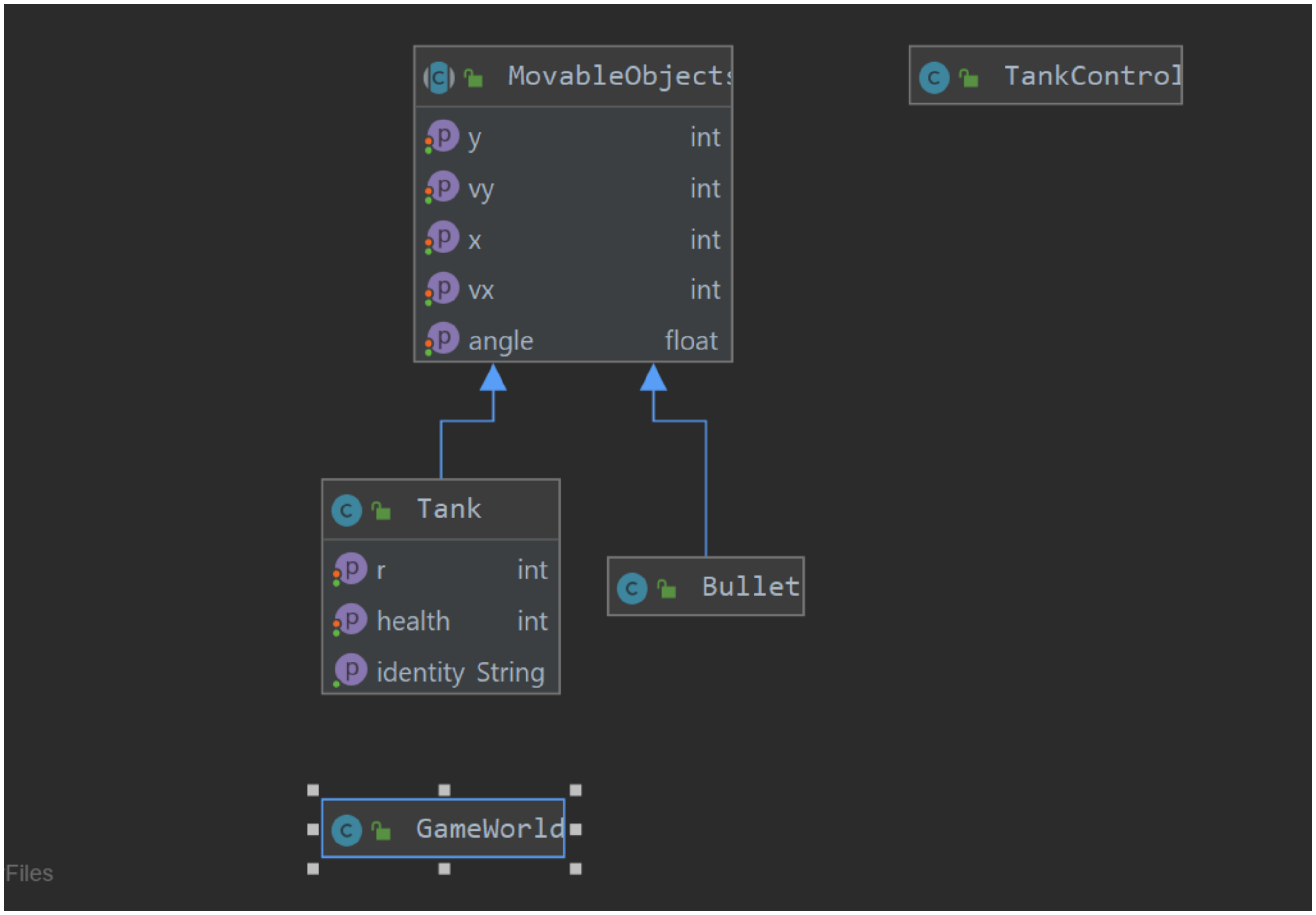
- The purpose of this game is two tanks playing until the end of the game.

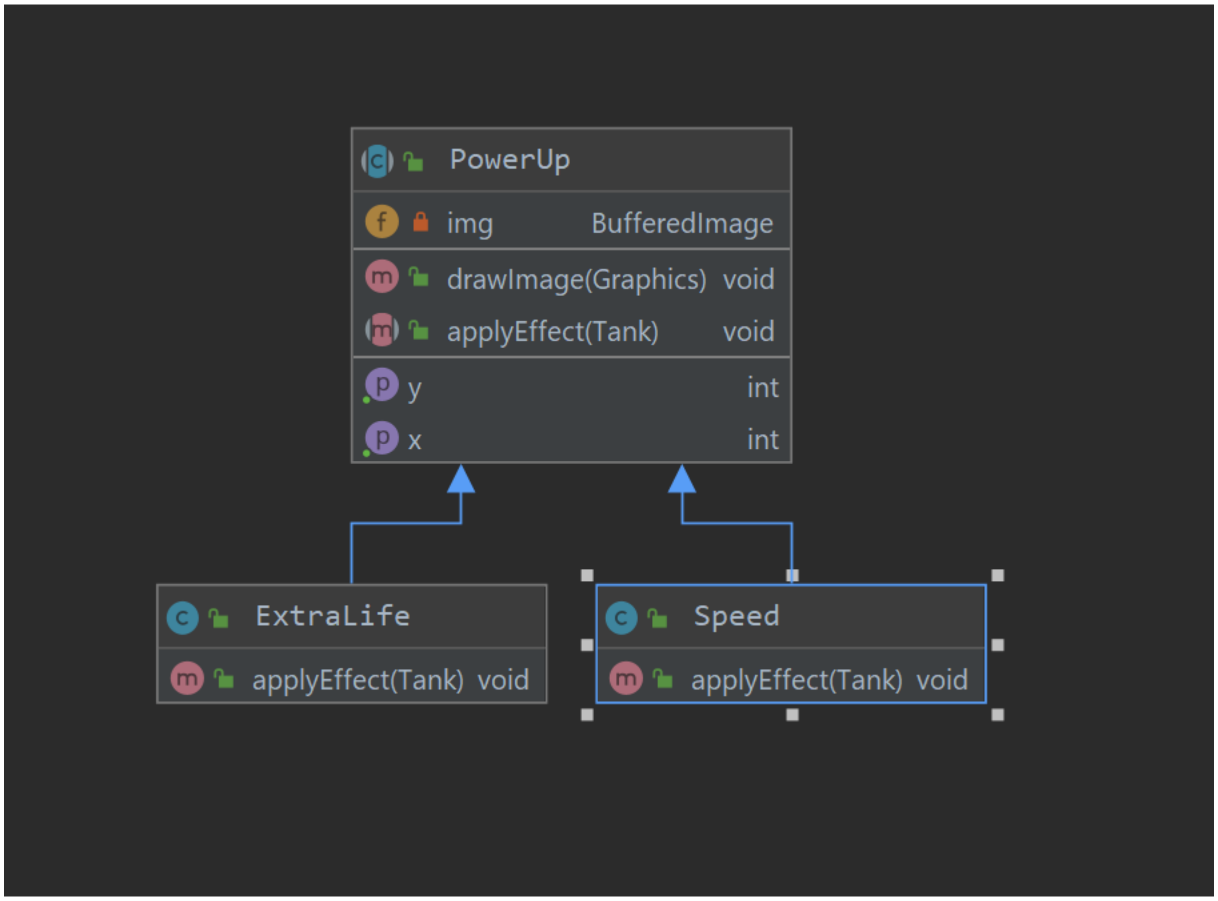
- The two tanks can shoot each other until the run of lives.
- There are two types powers ups, extra life, and speed boost. The extra life gives life for the tanks and the speed will boost a player's speed.
- There are two types of walls breakable and unbreakable. The tanks can shoot the walls and collide with the walls and the powerups.

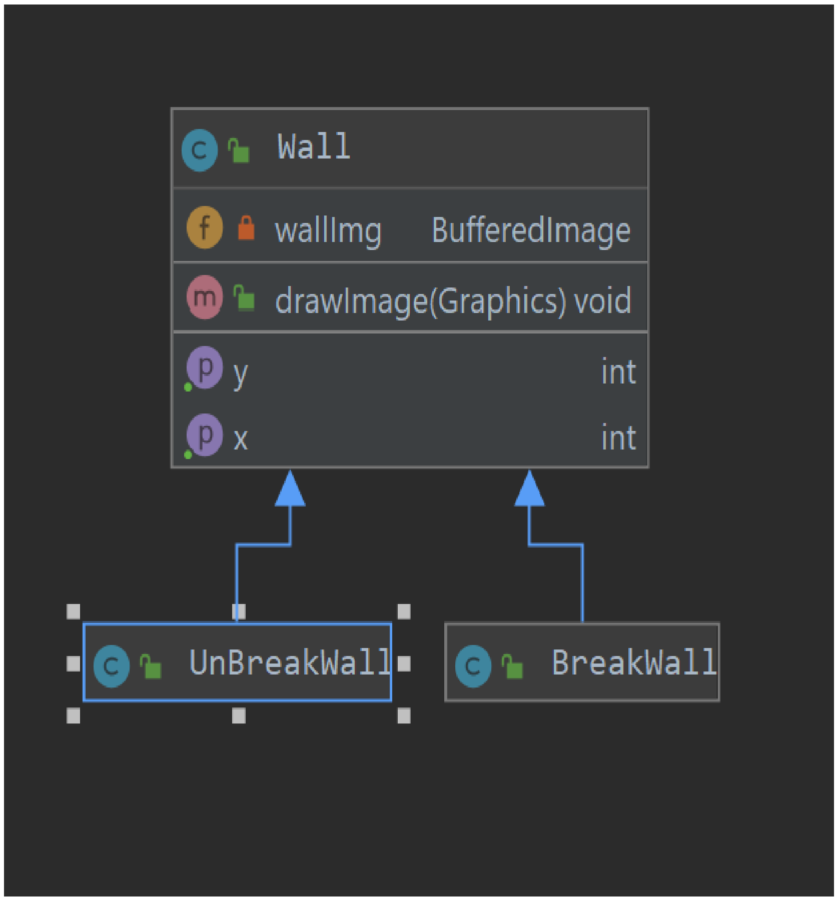
Assumptions made when designing and implementing the game.

Assumptions I made while designing the game, was how to be creative for implementing it. Especially designing the map, the split screen, and the collision detection.

Tank Game Class Diagram


MovableObjects
y int
vy int
x int
vx int
angle float
TankControl
Tank
r int
health int
identity String
Bullet
GameWorld
Files


PowerUp
img BufferedImage
drawImage(Graphics) void
applyEffect(Tank) void
y int
x int
ExtraLife
applyEffect(Tank) void
Speed
applyEffect(Tank) void



Class Descriptions

The Tank Game consists of two main packages, the “GameObjects” and the “TankGameCore”.

GameObjects Package:

a) PowerUps
b) Walls
c) MovableObjects
d) Bullet
e) GameWorld
f) Tank
g) TankControl

TankGameCore Package:

a) Menus
b) GameConstants

c) GameWindows

GameObjects Package:

a) PowerUps

PowerUps is a package that includes three classes, ExtraLife, PowerUp, and Speed. The PowerUp is an abstract class, sharing methods and objects with ExtraLife, and Speed class. The PowerUp class has methods and a constructor which mainly structure the location of x, y, and the BufferedImage. The drawImage method which draws our object on the screen, and the last method is an abstract which applyEffect to the healthbars of the tanks and adds life to them.

b) Walls

Walls is a package that includes three classes, BreakWall, UnbreakWall, and Wall. The Wall is an abstract class, sharing methods and objects with BreakWall, and UnbreakWall. The Wall class simply has a method which is drawImage for drawing the walls and a constructor which structures for the x/y positions and the BufferedImage.

c) MovableObjects

MovableObject is an abstract class shares the same methods and objects with Bullet and the Tank. It consists of a constructor that structures the game movements by the x-axis, y-axis, angle, x/y velocity, and the rectangles. Ant it has getters/setters for the mentioned objects. Also, it includes an abstract method which is for checking collision, and an abstract method for updating the game.

d) GameWorld

- GameWorld class extends JPanel implements runnable. The Runnable interface instances are intended to be executed by a thread. The class must define a method of no arguments called run. This interface is designed to provide a common procedure for objects that need to execute code while they are active. Inside the run method I have generated a try/catch along with the while loop which repeatedly updates the players and repaints the game.

- I have created ArrayList for the walls and the PowerUps, and a HashMap for creating the tanks.
- I have generated a gameInitialize method, and I have stored the BufferedImage inside the try/catch. It consists of the game map that reads the map for drawing the walls, which I have implemented according to the posted lecture on the iLearn by the Professor.
- I have implemented the split screen logic inside this class. For the x coordinate the width of both game screens needs to be divided by 4. And for the y coordinate the height of both game screens needed to be divided by 2. For implementing this logic, I have created methods that checks the x/y coordinates for both tanks.
- The last method is the painComponent, which draws the game objects on the screen like the walls, tanks, power up, split screen, health bars, and the mini map. I have spent huge amount of time for drawing the mini map, and importantly I used some math. For the x coordinate, I divided the width of game screen by two and subtracted from the width of game divided by twelve. For the y coordinated I implemented the same logic, but for the height. I honestly played with random numbers until I came up with a strategic design.

e) Tank

Tank class is a particularly important, it extends the MovableObjects class and share same methods and objects. It consists of the Boolean variables that were set in the TankControl class and it updates the tank movement in four directions. I have implemented Boolean method inside to class to check collisions for the walls, and I used the instanceof operator to determine the type of objects. The next method is checkCollision, which is for the PowerUps. I have created an ArrayList of powerup that loops through the x/y directions. When the tanks collided the PowerUps it will apply the effects.

f) **TankControl**

TankControl class is for controlling the movement of the players through the keyboard keys which I have taken from the source code on the iLearn.

TankGameCore Package:

a) **Menus**

Menus is a package that consists of two classes the EndGamePanel, and the StartMenuPanel. These classes were implemented by the Professor, and I used the source code on the iLearn, as starter classes for the game.

b) **GameConstants**

The class stores the height and width of the game screen and the game world, also includes the start/end menu width and height. They are all static final values, which means they are not changeable.

c) **GameWindows**

GameWindows class calls the main function, and it can run the game. This class was implemented by the professor, which creates the game windows by using JPanel.

Self-Reflection:

Implementing this tank game was remarkably interesting and challenging for me. I have learned and experienced many concepts that I had never undergone before. Even though this project was super challenging and difficult for me. But dividing the project into pieces was helpful for me to complete all the requirements. For instance, from the first day the project was posted, I started assuming how to implement a creative design, like in terms of OOP principles. Since I am a visible person, so I drew the whole game design on a scratch paper, and I pictured how many classes do I need, and what objects might need to share the same classes. I used the source code given by the Professor on the iLearn, and I spent two days understanding the code piece by piece then I started working on it. I also followed the lectures posting on the iLearn, they were extremely helpful. By following those videos, I was able to design the map, draw the walls, and implement shooting for the game. The most aspect that I struggled a lot and spent almost two weeks, was collision detection. I was very lost and confused how it works. But after my research I found out there is some logic and some math behind implementing the collision detection. Based on the requirement, the players must have collided with some objects in the game. I learned that it is basically checking where the bouncing rectangle will be instead of where it already is. This allowed me to check collision in the X and Y directions separately. And the algorithm is If I keep moving in my current X direction, will I collide, if so, reverse my X direction. I used the same logic with the Y direction. And I played with different numbers for checking the collisions for both X/Y directions until finally I figured which number will be proper to use. Also, for the collision check method I created an ArrayList which takes the games objects for handling collision between them. Generally, I am glad that I had this opportunity and enough time to complete this game. I enjoyed implementing it, and I am delighted for learning OOP principles in Java.

Conclusion

The term project, which was designing a 2D game in Java, was challenging. I want to admit that my game is not perfect, but I tried my best to complete all the requirements. However, I learned a lot from my mistakes and following OOP principles. I am glad that implementing this project will help to be a better programmer or a software developer in the future.

Reference:

1. I have used the “Tankrotationexample” # as a starter code on the iLearn
2. GameSprites: https://www.gameart2d.com/sprites.html

3. Collision Detection: https://happycoding.io/tutorials/processing/collision-detection#collision-detection-with-moving-objects #

.... #

.... #

.. #

.... #