CSC 413 Project Documentation

Spring 2021

Tank Game

Kyle Harvey

915139815

CSC 413-01

https://github.com/csc413-su21/csc413-p1-k-harvey

Table of Contents

[1 Introduction 3](#_Toc79095612)

[1.1 Project Overview 3](#_Toc79095613)

[1.2 Technical Overview 3](#_Toc79095614)

[2 Development Environment 3](#_Toc79095615)

[3 How to Build/Import your Project 3](#_Toc79095616)

[4 How to Run your Project 3](#_Toc79095617)

[5 Assumption Made 5](#_Toc79095618)

[6 Implementation Discussion 5](#_Toc79095619)

[7 Tank Game Class Diagram 6](#_Toc79095620)

[8 Class Descriptions of classes implemented in the Tank Game 6](#_Toc79095621)

[9 Project Reflection 6](#_Toc79095621)

[10 Project Conclusion/Results 7](#_Toc79095622)

# Introduction

## Project Overview

A multiplayer split screen tank game. Two users can play different tanks in two views. Each players goal is to shooting enemy tank until the health of the enemy tank is zero.

## Technical Overview

For this project, the goal is build a tank game where the two tanks will battle each other until one is destroyed. There will be two split screen windows where two users can play as different tanks. Each tanks have a health bar and 3 lives, they must pick up 3 of the power ups available that will boost their chances of winning. The tanks will fire a bullet that will shoot to take out the other tank. Also have a collision detection for all the ticking objects and game objects.

# Development Environment

Version of Java Used: Java 16

IDE Used: InteliJ IDEA 2021.1.2 Ultimate Edition

Resources: Downloaded sprites from google image searches.

# How to Build/Import your Project

In order to import this tank game project as a JAR file, first go into project structure and select artifacts. We add the out directory that we want to make a JAR which should just show the project you have open and the main class that’s running the game, after selecting hit apply and OK. After that we could and build artifacts with it inside the out folder. It’ll produce a JAR file zip that could be moved into another folder and open the game by right clicking to open and run without needing to build through the compiler.

# How to Run your Project

The project runs when you click run after building the project to make sure it compiles properly with no errors. Once you’re able to compile with no issues, you could run the main program and that will start the tank game. The process should look like this:

Run Intellij

**IntelliJ** : File -> New -> Project from Existing Sources -> Browse to the project folder and open all -> tankgame . To make the game run, you must run **TRE** class.

**JAR :** jar -> csc413-tankgame-k-harvey.jar -> right click -> Run

**Rules and controls:**

Player 1 Keys: Player 2 Keys:

Turn Left: **A Left Arrow**

Turn Right: **D Right Arrow**

Forward: **W Up Arrow**

Backward: **S Down Arrow**

Shoot: **Space Enter**

**Health:** How much health the players has left. Health power ups will replenish any lost health.

****

**Health Power up:**

****

**Lives:** How much lives a player has left, drops down once a health bar is depleted

****

**Shields:** Shows you how many shield power ups you picked up, protects you from taking damage from bullets.

****

**Shield Power up:**

****

**Range:** Shows how much range your bullets have, the more range power ups you pick up the further bullets will travel.

****

**Range Power up:**

****

**Temp:** Your temperature gauge for your bullets fired, the more bullets fired the hotter it gets and when fired too many will be needing to cool down. Temp power up will lower the cool down.

****

****

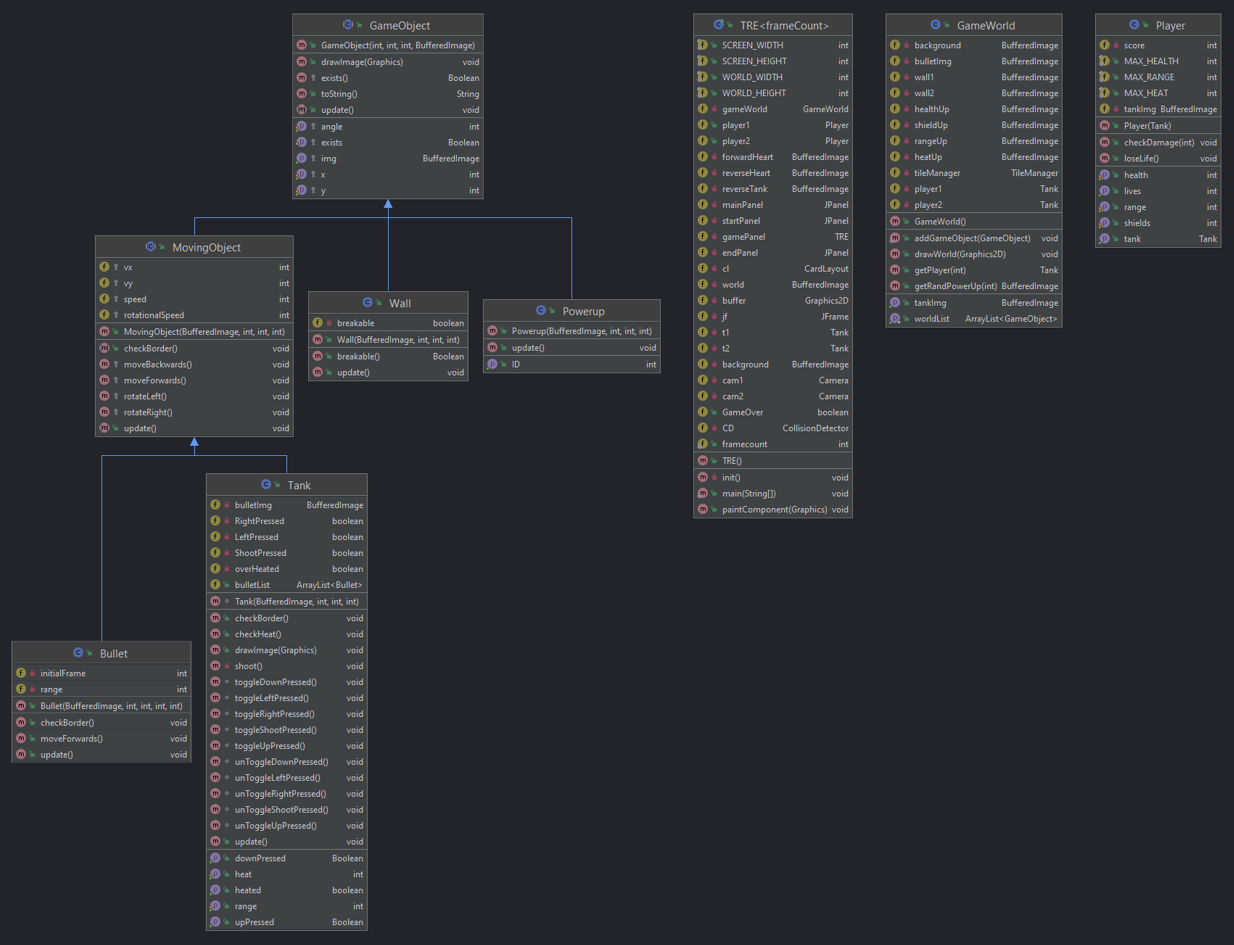
# Assumption Made

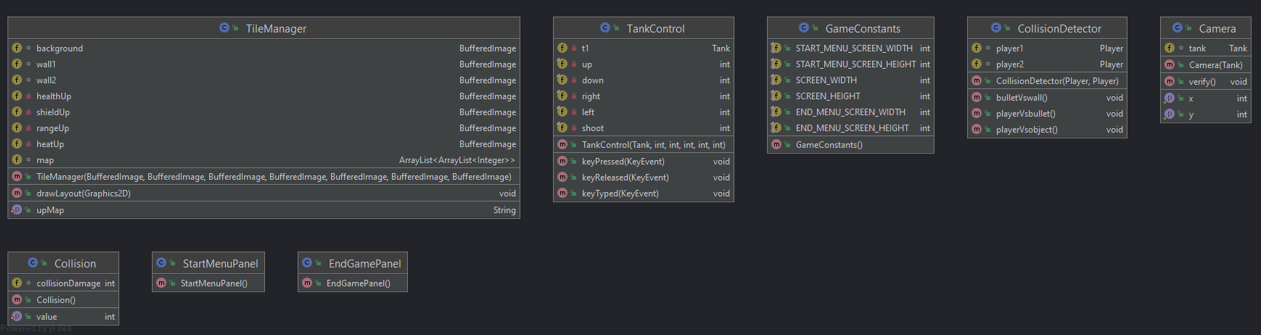
Some assumption made when it came to designing this project that it would be easy to follow from the videos alone but that was not the case. Most of the design and implementation portion came from the guidelines, such as the implantation containing a splitcreen and minimap. The game also requires a moving object, bullet, and tank. Within the game we needed to make a map that has walls both unbreakable and breakable when bullets hit it, which is the collision logic. Once we covered most of the fundamental parts of the game, the rest comes down to design process for the game how we want it to be played.

# Implementation Discussion

For implementing this project, we were provided with starter codes to help give that push which provided a Tank, a Tank Control, and TRE to put the Tank and game together. There’s also menus for the Start Menus and End Game panel that uses the GameConstants class, all of it could be ran with the Launcher class. Although I didn’t exactly use all the starter codes, I was able to work with some of it such as Tank and Tank control to make other Java files such as Player, the Camera following to show the Tank along with the camera for the minimap. Also needed to make a gameobject class for all the objects that we would be using such as the walls, tanks, powerups, and bullet. Once we have our objects created, we could focus on collisions for the tanks and bullets so we have both the collision detector class to take all the bullets from both the players and the damage if the bullet object enters the tank’s hitbox. Using the Wall class and power ups, we use the Tile Manager class to manage those objects to put our maps together after we design it in a text file using excel tables. Once we have all the java files needed for the game, we could focus on the TRE which is where main is for putting together everything and creating the game when we compile and run it.

# Tank Game Class Diagram





# Class Descriptions of classes implemented in the Tank Game

# Project Reflection

My thoughts on this project, although it wasn’t too difficult in the sense of what we needed to do the hard part was trying to figure out where to start and go from there. Most the time consuming part came from piecing everything to together and making sure they work with each other.

# Project Conclusion/Results

Overall this project was a good experience for learning Java programming, taking things from what we learned in pervious classes our goal was to finish the incomplete project and get it working properly. Although we started with the simple calculator, the functionality of it can get a bit more complex. Slowly going through the program and learning what we needed to do to get portion of the code to work and to pass the built in tests helped us understand where we went wrong and what we could fix. Although I wasn’t able to pass all the tests provided and was very close, I think given a bit more time I could have gotten the Evaluator test to work. Hopefully this will give us more experience for the next assignment.