San Francisco State University
Computer Science

Spring 2020
CSC 415

Tank Game

https://github.com/csc413-01-spring2020/csc413-tankgame-karinaaltheaabad

Super Rainbow Reef

https://github.com/csc413-01-spring2020/csc413-secondgame-karinaaltheaabad

Karina Althea F. Abad
May 20, 2020

Table of Contents

Project Information

Two projects were created for a software development course in San Francisco State University, CSC 415. The goal of the class is to strengthen programming knowledge, get more programming practice through implementation, acquire better design skills and implement everything in two final projects that will demonstrate the overall skills that were developed and strengthened while taking the class.

Introduction

The final overall project of the class was to showcase what was learned while taking CSC 415. Several topics discussed and practice were design patterns, object oriented programming, and a review of basics in java programming. The first game created was a Tank Game, which is a two player game and each player is represented by a tank in a huge map set up like a maze. The objective of the game is to try to destroy the other player by shooting, and maintaining lives by power ups or boosting speed. The next game, Super Rainbow Reef, was a reinforcement of the skills that were learned in the Tank Game. The objective of Super Rainbow Reef is to allow the player to control left or right movement in order to save pop from dying by passing by Katch. The game is similar to how you would play a brick breaker game, instead Katch is used to launch Pop back up every time Pop hits Katch. Pop then would have to hit the blocks and earn points through hitting blocks, and win by destroying the Big Legs through impact on them. The player is also able to stay alive via Pop hitting life blocks and gain more points by hitting specific blocks like a double block.

Development Environment

The version of Java used to implement both games is Java 14.0.1 with IntelliJ IDEA CE as the IDE. In order to fully implement the game, several libraries were used. The Java Swing library provided the graphical user interface for the game. The Java Event library was also used in order to indicate specific key strokes made by the user in a component. In order to process images, the Java Image library was used. The Java Utility library was used for data structures, specifically array lists. In terms of input and output the I/O library was accessed in order to used specific classes such as BufferedReader, IOException, and InputStreamReader, which handled reading the text files that was generated into maps.

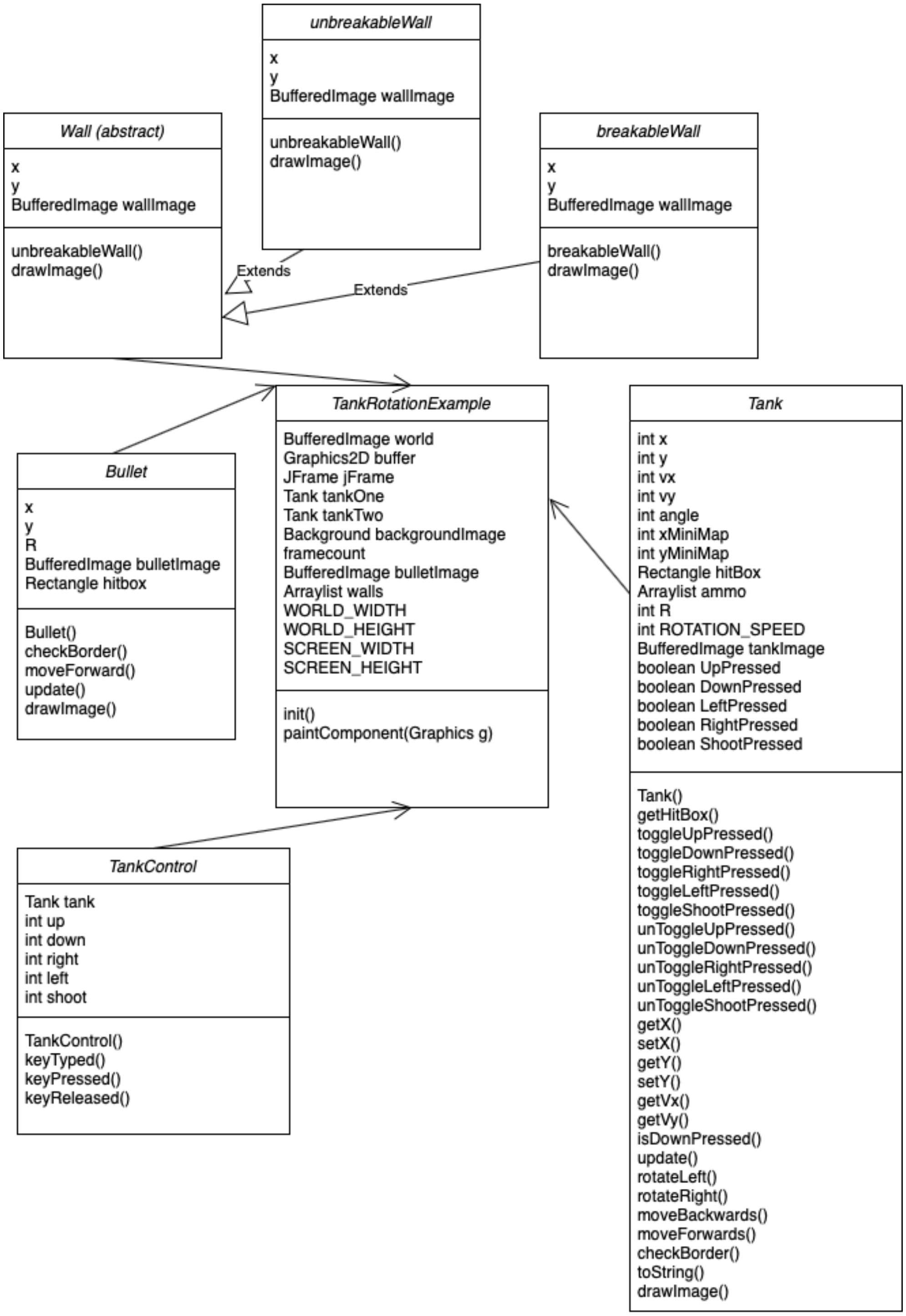
How To Run Tank Game

In order to run the Tank Game, you need to go to the specific repository csc413-01-spring2020/csc413-tankgame-karinaaltheaabad and clone the project (https://github.com/csc413-01-spring2020/csc413-tankgame-karinaaltheaabad), into IntelliJ VCS, and either run through the jar file by building the jar file found under the jar folder, or accessing project structure, clicking modules, and reconnecting src to source and resources to resources and then clicking apply.

How To Run Super Rainbow Reef

In order to run Super Rainbow Reef, you need to go to the specific repository csc413-01-spring2020/csc413-secondgame-karinaaltheaabad and clone the project (https://github.com/csc413-01-spring2020/csc413-secondgame-karinaaltheaabad.git), into IntelliJ IDEA CE VCS, and either run through the jar file by building the jar file found under the jar folder, or accessing project structure, clicking modules, and reconnecting src to source and resources to resources and then clicking apply.

Tank Game Class Diagram



unbreakableWall

x
y
BufferedImage wallImage

unbreakableWall()
drawImage()

Wall (abstract)

x
y
BufferedImage wallImage

unbreakableWall()
drawImage()

breakableWall

x
y
BufferedImage wallImage

breakableWall()
drawImage()

TankRotationExample

BufferedImage world
Graphics2D buffer
JFrame jFrame
Tank tankOne
Tank tankTwo
Background backgroundImage
framecount
BufferedImage bulletImage
Arraylist walls
WORLD_WIDTH
WORLD_HEIGHT
SCREEN_WIDTH
SCREEN_HEIGHT

init()
paintComponent(Graphics g)

Bullet

x
y
R
BufferedImage bulletImage
Rectangle hitbox

Bullet()
checkBorder()
moveForward()
update()
drawImage()

Tank

int x
int y
int vx
int vy
int angle
int xMiniMap
int yMiniMap
Rectangle hitBox
Arraylist ammo
int R
int ROTATION_SPEED
BufferedImage tankImage
boolean UpPressed
boolean DownPressed
boolean LeftPressed
boolean RightPressed
boolean ShootPressed

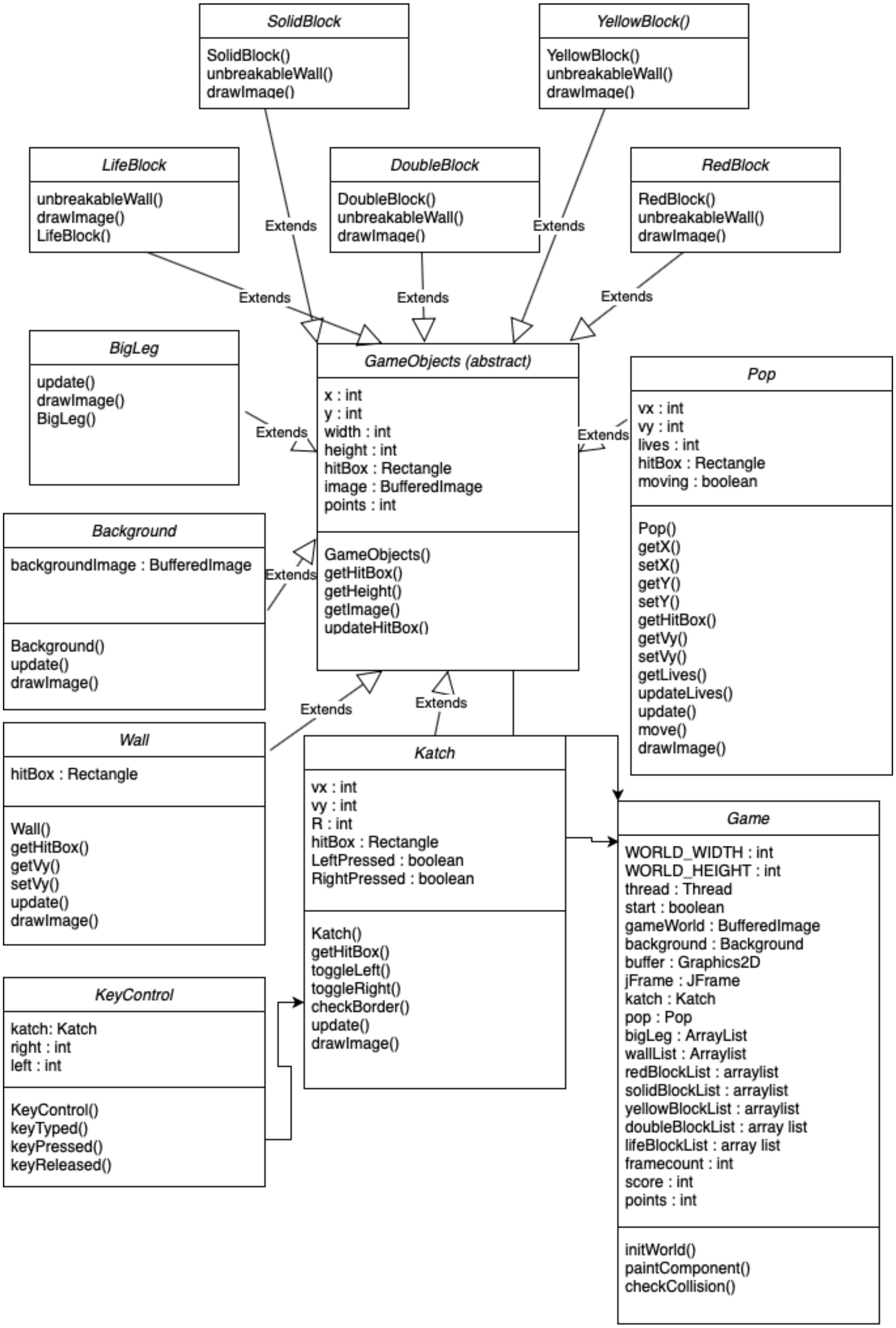
Tank()
getHitBox()
toggleUpPressed()
toggleDownPressed()
toggleRightPressed()
toggleLeftPressed()
toggleShootPressed()
unToggleUpPressed()
unToggleDownPressed()
unToggleRightPressed()
unToggleLeftPressed()
unToggleShootPressed()
getX()
setX()
getY()
setY()
getVx()
getVy()
isDownPressed()
update()
rotateLeft()
rotateRight()
moveBackwards()
moveForwards()
checkBorder()
toString()
drawImage()

TankControl

Tank tank
int up
int down
int right
int left
int shoot

TankControl()
keyTyped()
keyPressed()
keyReleased()

Super Rainbow Reef Class Diagram



SolidBlock

SolidBlock()
unbreakableWall()
drawImage()

YellowBlock()

YellowBlock()
unbreakableWall()
drawImage()

LifeBlock

unbreakableWall()
drawImage()
LifeBlock()

DoubleBlock

DoubleBlock()
unbreakableWall()
drawImage()

RedBlock

RedBlock()
unbreakableWall()
drawImage()

BigLeg

update()
drawImage()
BigLeg()

GameObjects (abstract)

x : int
y : int
width : int
height : int
hitBox : Rectangle
image : BufferedImage
points : int

GameObjects()
getHitBox()
getHeight()
getImage()
updateHitBox()

Pop

vx : int
vy : int
lives : int
hitBox : Rectangle
moving : boolean

Pop()
getX()
setX()
getY()
setY()
getHitBox()
getVy()
setVy()
getLives()
updateLives()
update()
move()
drawImage()

Background

backgroundImage : BufferedImage

Background()
update()
drawImage()

Wall

hitBox : Rectangle

Wall()
getHitBox()
getVy()
setVy()
update()
drawImage()

Katch

vx : int
vy : int
R : int
hitBox : Rectangle
LeftPressed : boolean
RightPressed : boolean

Katch()
getHitBox()
toggleLeft()
toggleRight()
checkBorder()
update()
drawImage()

KeyControl

katch: Katch
right : int
left : int

KeyControl()
keyTyped()
keyPressed()
keyReleased()

Game

WORLD_WIDTH : int
WORLD_HEIGHT : int
thread : Thread
start : boolean
gameWorld : BufferedImage
background : Background
buffer : Graphics2D
jFrame : JFrame
katch : Katch
pop : Pop
bigLeg : ArrayList
wallList : Arraylist
redBlockList : arraylist
solidBlockList : arraylist
yellowBlockList : arraylist
doubleBlockList : array list
lifeBlockList : array list
framecount : int
score : int
points : int

initWorld()
paintComponent()
checkCollision()

Descriptions of Classes Shared Between Both Games

Classes that were shared between both games are the following:

Background.java—The background class was created to be able to display the background on the screen through calling it via a constructor instead of creating a method that would do that within both games. The purpose was to try to segregate each object into their own classes as much as possible for design purposes.

KeyControl.java—KeyControl was the same in terms of the controls that the game would have to execute. For tank game, the controls consisted of up, down, left, right and space or enter for shooting depending on which player is handling specific controls. For the Super Rainbow Reef game, I chose to only implement left and right for controls since the player only needs to manipulate Katch's movements.

Game.java—The main structure and layout or idea of the game was copied from the original tank game which was gotten from Professor Souza. The program starts with a game loop and calls a function that initializes and draws the game.

Tank.java/Katch.java—Tank or Katch classes was almost copied and modified so that the controls would fit the game. I was not able to implement collisions in the tank game, so that was added into the Katch game, given that I had a lot of tutoring fixing my logic to be able to implement the controls, but essentially they're the player classes and consist of controls to be able to manipulate the player during the game. The class has functions that allows it to move within the specific constraints that it's allowed to do within or during the game. From this, we are also able to access the rectangle box's that surrounds the character to be able to detect collisions between objects or characters in the game. The class finally has a draw function that enables the character to be drawn and called from paintComponent method/function.

Descriptions of Tank Game Specific Classes

The following are specific classes from Tank Game:

breakableWall, unbreakableWall, Bullet and PowerUp classes were created to draw and be able to add them to arraylists, when accessed and converted from a numerical value to an object when converting the whole map text file to objects that would eventually displayed on the screen and manipulated to play the game. The PowerUp class was not implemented or used, because I wasn’t really able to implement collisions in the tank game as I was too late in asking for help to further understand the project. The Wall abstract class purpose was so that subclasses wouldn’t have to recreate each method and constructor that would be similar to either breakable wall and unbreakable wall objects in the game, although the only thing actually implemented was a draw abstract method, which is accessed by both unbreakable and breakable walls. The Bullet class was designed for functions that creates the bullets, be able to move forward after being shot, draw it on the screen and allow it to stay within the game frame.

Descriptions of Super Rainbow Reef Specific Classes

The following are specific classes from Super Rainbow Reef:

DoubleBlock.java, LifeBlock.java, RedBlock.java, SolidBlock.java, YellowBlock.java, Wall.java, BigLeg.java, are all classes that were implemented in order to be able to call them from Game, and create objects of each class that would be filled in array lists and handled by that. The Game class creates each object when the map is being converted from texts to objects, which is then added to the arraylist. This way I was able to manipulate the data when having to remove them from the screen with collisions, or convert them from numeric data to an object that will later on be displayed on the panel. GameObjects.java was created as an abstract class to be able to define the common functions between classes such as getting their rectangle height and width for collisions, as well as their image width and height in order to draw them out correctly on the screen. The GameObjects class also provides a constructor so each subclass can just call the main constructor instead of creating one in each subclass and implementing many of the same functions such as draw, get rectangle box, and getters and setters for their x and y values.

Self Reflection on Development of Games

In the beginning of the semester I was struggling because I didn’t know how to tackle large problems and solving them through code. I knew that I could do it, I was just constantly thinking of how ill equipped I was given my background and the way previous classes were taught. I didn’t start on the tank game early enough because every time I attempted to do it, I would get bogged down on how I couldn’t understand what was going on with other people’s code. I then luckily was forced to learn C in another class, another classmate had helped me apply my knowledge when we worked on a project together. That’s when I was able to pick up where I left off with CSC 413 and the next game, Super Rainbow Reef. Unfortunately, my loss of fear in coding and better understanding did not happen until after the tank game so when I had coded the tank game, I was able to do so simply by following tutorials that Professor Souza posted on iLearn, and continued to watch videos on how other games were created and even tried to learn implementation of other games through sites like CodeGym, only to find out that it was my fear into trying to understand other people’s code that stopped me from being able to generate a working tank game. With Super Rainbow Reef, I was able to figure out somehow by understanding what each part and method of the code does and implementation through object oriented programming which definitely strengthened my skills in programming, concepts and application, overall.

Project Conclusion

This project—tank game and super rainbow reef—enabled me to apply my introductory coding knowledge properly, for the first time, into creating projects that I am able to speak about and further enhance to build a portfolio. The Tank Game ended up to be an incomplete game with a ton of opportunity for growth and development in terms of my coding and understanding skills, however, when it came to the Super Rainbow Reef game, I was able to completely get a working game in, at the very least. Both games definitely presented opportunities to learn and enhance object oriented programming, along with design patterns in order to correctly and efficiently produce code that is efficient and maintainable. There was a lot of growth done, as well as a lot of more growth to be done, but through these projects, I was able to determine where I am going, how much I am able to improve given that I ask for help and ask the correct answers, and how much I need to continue to learn and improve.