Name: Alicia Ramirez
Student ID: 917379715
Class, Semester: CSC413-02, Fall 2019
Tank Game Link: https://github.com/csc413-02-fall2019/csc413-tankgame-aramirez23.git
Second Game Link: https://github.com/csc413-02-fall2019/csc413-secondgame-aramirez23.git

Introduction

Project Overview

The Term Project was to implement two games: a tank game and a second game of your choice. The tank game was to be designed in a way that much of the code would be able to be used in the second game in order to make its development easier.

Introduction of Tank Game (general idea)

The game is a two-player combat game where each player controls their own tank. Their goal is to destroy the other player's tank. They can shoot bullets at each other, pick up health power-ups to regain health, and break certain walls to reach their opponent. There is a mini-map to help the player situate themselves. There are tiles that will affect player movement like the mud that will slow the player down and make them unable to move diagonally, and fast tiles that push you in a certain direction. You can load different maps to play in, or even create your own.

Introduction of Second Game (general idea)

The second game is a one-player action RPG where the player controls the protagonist. They can shoot fireballs, level up, and pick up items. They receive a quest from their double to get a magic spellbook to save a town they find themselves in, though it's more of a field. Depending on if they decide to help the town will change their reputation with their double. There's other NPCs to talk to, but the double is the only one that gives a quest with a dialogue option, as well as follows the player in whatever map they're in.

Development Environment

Version of Java Used

Jdk-12.0.2

IDE Used

IntelliJ IDEA (Ultimate) 2019.2.1

Special Resources

Music/Sound:

Piano Remix - cheesepuff - https://freesound.org/people/cheesepuff/sounds/112071/
Powerup2 - AbbasGamez - https://freesound.org/people/AbbasGamez/sounds/411443/
Success 2 - Leszek_Szary - https://freesound.org/people/Leszek_Szary/sounds/171670/
Grunt - Reitanna - https://freesound.org/people/Reitanna/sounds/242623/

Aaaaeuuuuuuuuugh - Reitanna - https://freesound.org/people/Reitanna/sounds/351157/
JM_FX_Fireball 01 - Julien Matthey - https://freesound.org/people/Julien%20Matthey/sounds/105016/

Graphics:

Grass and Pine Tree Coated Hill during Cloudy Daytime - Wolfram K - https://www.pexels.com/photo/grass-and-pine-tree-coated-hill-during-cloudy-daytime-745242/
Zelda-like tilesets and sprites - ArMM1998 - https://opengameart.org/content/zelda-like-tilesets-and-sprites
8x8 critter pack - patvanmackelberg - https://opengameart.org/content/8x8-critter-pack
Fireball Flying Through the Air - ScagHound - https://opengameart.org/content/fireball-flying-through-the-air
Universal LPC Sprites - http://gaurav.munjal.us/Universal-LPC-Spritesheet-Character-Generator/

How to build/import/run your game in the IDE you used

Clone the repository from the link given above using the terminal of your choice. Type "git clone https://github.com/csc413-02-fall2019/csc413-tankgame-aramirez23.git" or "git clone https://github.com/csc413-02-fall2019/csc413-secondgame-aramirez23.git" depending on the game you want to accomplish this. The following instructions work for either game.

In IntelliJ, click "import project" and select the root folder as the source root of your project. Keep the "Create project from existing resources" radio button selected. You can leave all the default fields alone on the next screen. Continue until you are asked to select your version of JDK, which you put pick your version(I had jdk-12.0.2). No frameworks should be detected. To play the game without the jar, right click on TankGame class and then left click "Run 'TankGame.main()'. If you want to build the jar, then go to 'project structure' in the 'File' toolbar on IntelliJ. Go to artifacts, click on the plus button, click on Jar, then on from modules with dependencies. Click on main class and then select TankGame as your main class. Select extract to the target JAR, leave all other options the same and click ok. Click apply and exit out of that menu. Under the 'Build' toolbar, click on 'Build Artifacts'. Click on 'Build' in the dialogue that pops up. Now in the out/artifacts/csc413_secondgame_aramirez23_jar or out/artifacts/csc413_tankgame_aramirez23_jar folder of your project directory depending on the game should be the JAR. Double click on the jar to play it.

Rules/Controls of your game

Tank Game

Main Menu:

Press Enter to load the default map and begin playing.

Press Space to load a custom map. This will bring up a file selector window where you can choose the map file to load. Next will bring up another file selector window where you can choose the powerUp placement file to load.

Here are the current available maps and corresponding powerUp placement maps (All except "map.txt" are in resources/ExtraMaps):

"map.txt" - "mapPowerUp.txt" (Default Map - Balanced with all tiles)

"ArenaMap.txt" - "ArenaPowerUp.txt" (Arena with only floor)

"FastFastFastMap.txt" - "FastFastFastPowerUp.txt" (Arena with a lot of fast tiles)

"MazeMap.txt" - "MazePowerUp.txt" (Map that's a tight maze. Very confusing.)

"testMap.txt" - "testMapPowerUp.txt" (Map used in early stages of development, is unwinnable)

Instructions for creating a map are in the Creating Maps section further down.

Tank 1(Left Screen) Controls:

W - Up S - Down A - Rotate Left D - Rotate Right E - Shoot

Tank 2(Right Screen) Controls:

I - Up K - Down J - Rotate Left L - Rotate Right U - Shoot

You only get one shot until the bullet hits something. This is to allow for more strategic gameplay as players must think about where/when they shoot. Once someone hits -1 lives, the game will end and after 5 seconds will restart at the main menu.

Creating Maps:

To create a map, you start with a .txt file. This text file will consist of 50 lines with 50 characters seperated by spaces. Here is an example of one line:

"0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 0 0 0 0 0 1 1 1 1 1 2 2 2 2 2" (Without the "")

Each number corresponds to a tile:

0 - Floor 1 - Wall 2 - Breakable Wall 3 - Mud (Slows down tank and allows no diagonal movement) 4 - Fast Right (Moves tank right) 5 - Fast Left (Moves tank left)

Fill them out however you please, with one restriction. The tanks will be placed at these tiles:

Tank 1 - Row 25, Column 12, Facing Right

Tank 2 - Row 25, Column 38, Facing Left

Also make sure the tanks can't go beyond where the screen draws, as that will freeze the game until the tanks move back to where there are drawable tiles.

Same goes for the powerUp placement .txt files, but these are what the numbers stand for:

0 - No PowerUp 1 - Health PowerUp

Second Game (Action RPG)

Main Menu:

Press Enter to start the game.

In-game:

W - Move Up S - Move Down A - Move Left D - Move Right E - Shoot Fireball Hold(Q) - Show Quest Log Hold(C) - Show Stats Screen Hold(I) - Show Inventory

To talk to an NPC walk into them. When there's no dialogue choice, press space to continue and quit the dialogue. Try walking into NPCs multiple times, they might say something different. If there is a dialogue choice, press the 1 or 2 keys depending on the choice you want.

Depending on if you help your clone or not will lead to different outcomes/reputations, but either one gives 100xp, i.e one level up.

Go to the blue tile to transport to another map. It may take a bit to load.

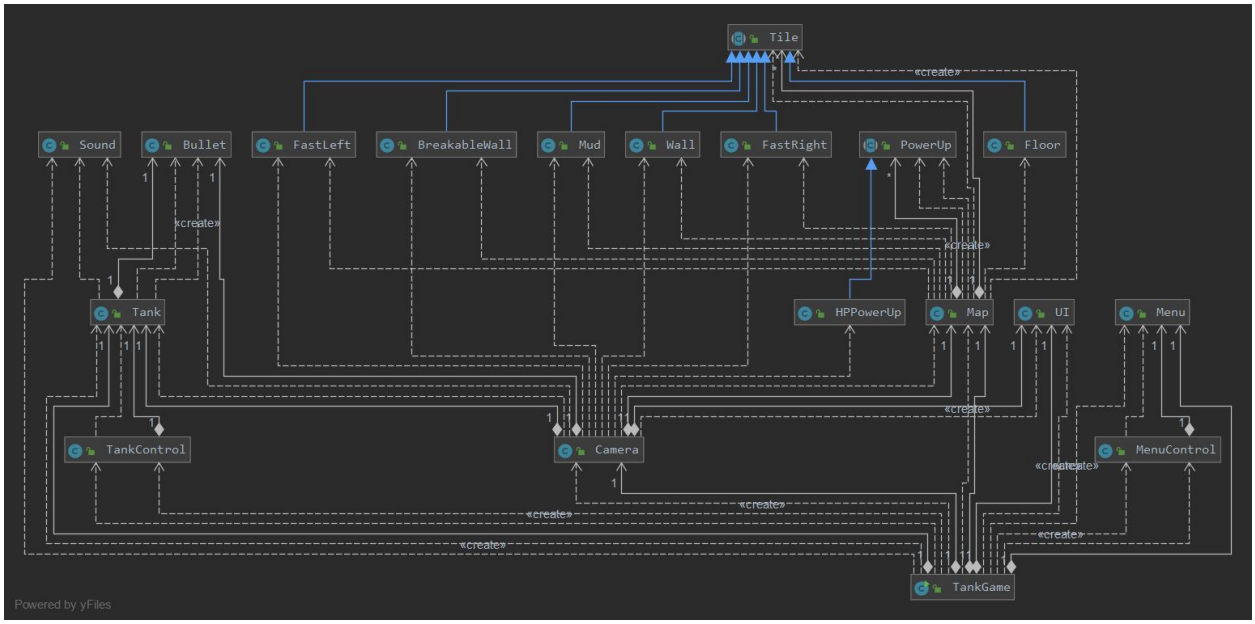
All items respawn when traveling between maps. The clone will remain on any map the player is on. There are only two maps: the area with NPCs, and the field with slimes.

Slimes will chase you within a certain range, and they take one fireball to defeat. They will give 10xp upon defeat. Getting 100xp will level you up, giving you 1+ maxHP.

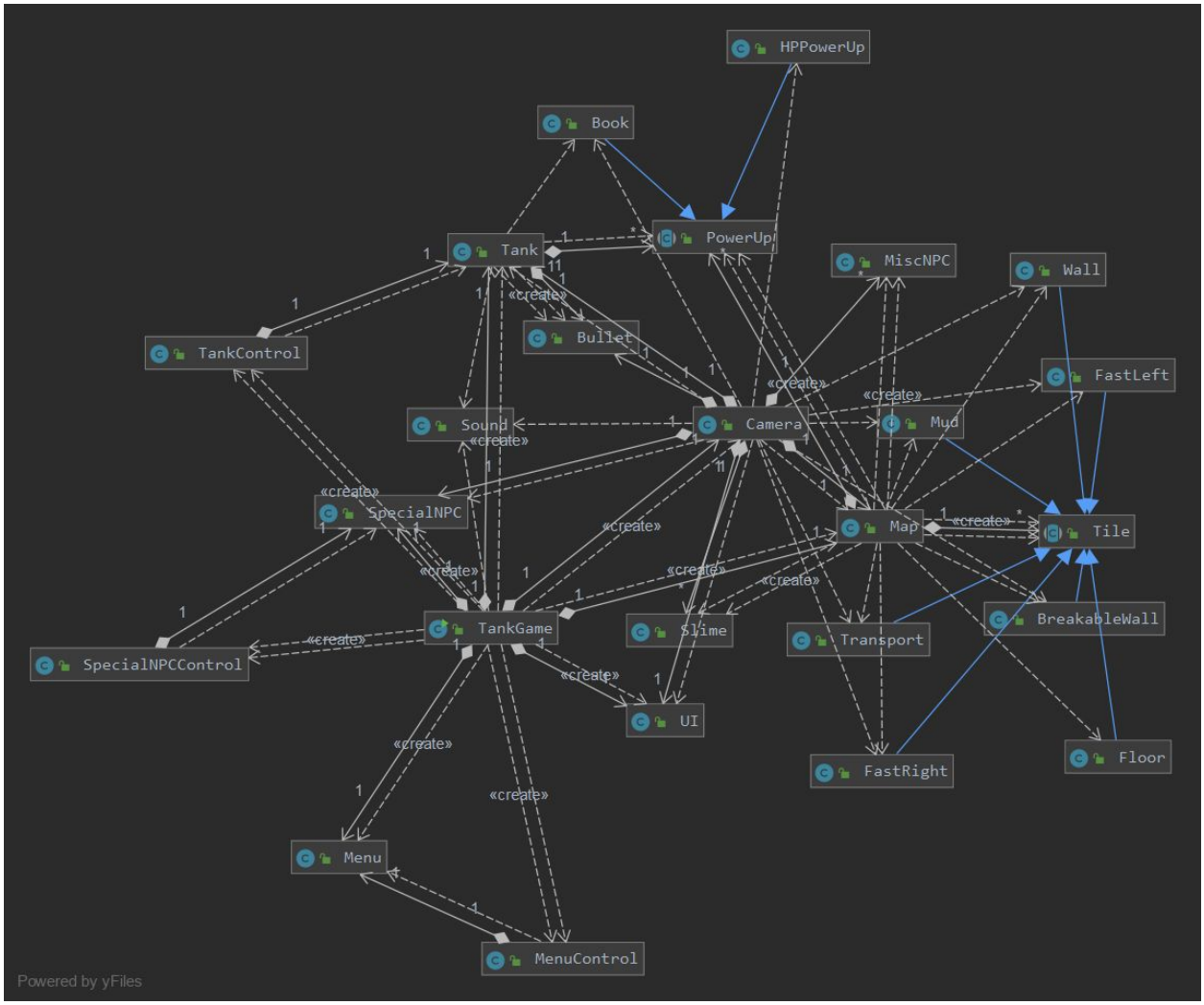
Assumptions made when designing and implementing both games

I assumed that the player would always spawn in the same x and y position and this designed the maps accordingly. I assumed no map would allow the player to go past the boundary where there are no tiles to draw, so I designed the maps accordingly. I assumed the player wouldn’t care if items respawned between maps for the action RPG, so I didn’t make them disappear forever once grabbed.

Tank Game Class Diagram



Second Game Class Diagram



Class descriptions of all classes shared between the two games

Since some shared classes had updates going from the tank game to the second game, each description will explain what parts they share and what parts are special in each class implementation.

TankGame Class

This class is the main class of both games. It initializes the main menu, as well as the primary components of the game (Tank object, Camera object, Map object, UI object). In the Tank Game, it also handles when the game ends in order to restart it. For the Tank Game's menu, it handles the file chooser dialogue windows when it is selected. For the second game, it removes this and just has the default option, while including an initialization of the SpecialNPC object.

MenuControl Class

This handles the keypresses for the main menu of both games. There are two keys it listens for: space and enter.

Menu Class

This class handles the image and text that appears on the main menu. It's a different image/text for each game, but the code is essentially the same.

Sound Class

The Sound class consists of various public static methods that play sound effects/music pertaining to each game. Each method loads the file and plays it.

UI Class

The UI Class handles the text for the UI, the hp and lives in the tank game, and just the hp and health bar in the second game.

TankControl Class

This handles the keypresses for the player characters of both games. There are two keys for movement and shooting in the tank game, and the addition of quest log, stats screen, inventory screen, and dialogue option buttons.

Tank Class

This class creates the player character: the tank in tank game, and protagonist in second game. In the tank game it handles movement, rotation, bullets, health/lives, and the effects of certain tiles. These are all checked in the update method. Once the tank is dead completely, it will lock the player's movement. In the second game it handles the same as above but changes the movement methods to be basic up, down, left, right without rotation. It also adds a rudimentary animation system in the drawImage method in order to have multiple frames for each movement along with frame delay. The hit method has been altered to remove lives and add invincibility frames so the player doesn't die in one hit.

Bullet Class

This class handles the bullets/fireballs the tank/player shoots. It handles their movement and position.

Map Class

The Map class handles creating the map, items, and NPCs(second game only). Each of these is created from a .txt file filled with 50 x 50 arrays of numbers. Each number corresponds to a tile, item, or NPC position. The Map class reads them and creates a 2d array from them of the corresponding objects and returns it to the Camera class(similar to the Interpreter project). It also handles drawing the mini-map. In the second game it also handles transporting maps, which is essentially the same as loading them, just replacing the arrays with new ones corresponding to the new map. This flexibility is what allowed for multiple maps and the ability to make custom maps quickly.

Camera Class

While it may be called the camera class, a better descriptor would be that it handles the game state. It holds all objects in the game, updates them, and calls their draw methods. It handles collisions for all objects in a collisionDetection method.

Tile Class

The Tile Class is an abstract class that is used when the map creates a 2d array of tiles. Each tile-inherited class corresponds to a type of tile in the game. They have an x/y position, a hitbox, and an image file. These are the types of Tiles that are possible(each is a class inheriting from Tile):

Floor - Default tile that can be moved through.

Wall - Unbreakable wall that can't be moved through or shot through.

BreakableWall - Wall that can be shot through and replaced by a Floor Tile(handled by the map class).

Mud - Tile that can be moved through but will slow the player down if they collide with it(This effect is handled in the Tank class).

FastLeft - Tile that can be moved through but will push the player left it they collide with it(This effect is handled in the Tank class).

FastRight - Tile that can be moved through but will push the player right it they collide with it(This effect is handled in the Tank class).

PowerUp Class

The PowerUp Class is an abstract class that is used when the map creates a 2d array of power-ups. Each PowerUp-inherited class corresponds to a type of power-up in the game. They have an x/y position, a hitbox, and an image file. These are the types of PowerUp that are possible(each is a class inheriting from PowerUp):

HPPowerUp - Power-up that will increase the players health by one (This effect is handled in the Tank class).

Class descriptions of classes specific to the Tank Game

There are none. They were all transferred to the second game.

Class descriptions of classes specific to the Second Game

Transport Class

Another Tile-inherited class that when collided with will transport the player to a new map. It receives it's number from the map class which will identify where it transfers to, as well as the map/NPCs/item files to hold.

Book Class

Another PowerUp-inherited class that when picked up by the player will add the "Magic Spellbook" to their inventory.

Slime Class

This class creates a slime enemy. It will move towards the player if they are within a certain range. They also have the same rudimentary animation system the Tank class does in the second game.

MiscNPC Class

This class creates a miscellaneous NPC. They have a custom image, dialogue sequence, and name. When the player collides with them they will enter a dialogue, advancing the NPCs line by one until there are no more new lines, where they will just repeat their last line. They do not move, so they only have one image. Their names are displayed in the dialogue box. The player freezing during dialogue is handled in the Tank class.

SpecialNPCControl Class

This handles the keypresses for when the player is in dialogue with the SpecialNPC object. It is for dialogue options 1 or 2. They do not move, but they also do not get erased in map transitions, so they will stay in the same position no matter which map the player is on.

SpecialNPC Class

This handles the special NPC character, which is a double of the player character. They do not move, but they also do not get erased in map transitions, so they will stay in the same position no matter which map the player is on. When the player collides with them, they will enter dialogue, where they can choose the desired dialogue option for the quest. This is handled by the dialoguePointer variable, which changes to the corresponding dialogue position depending on what the player chooses.

Self-reflection on the development process during the term project

Working on the Tank Game was for the most part pretty smooth. I was able to figure out all the problems I came across in development since I have had some experience making games with Java before. The only difficult one was handling collision since I hadn't done that before, though I was able to figure out that I had solved it mostly, it was just that I was calling the collisionDetection method at the wrong time. Since it went smoothly, I was able to add extra features that weren't necessary like the mud and fast tiles.

The second game I feel I overscoped a bit too much. I didn't end up having as much time as I would have liked to work on the game, but I was fortunately still able to implement everything I said I would. Everything in the tank game transferred over with some minor changes, which I'm pretty happy with. One thing I would've done differently is have the animation system I put in the Tank and Slime class be its own class so I didn't have to rewrite it for each class that used it.

Same for the dialogue system, which was the most problematic aspect of the project. My implementation of the MiscNPC class worked well, but the SpecialNPC and multiple dialogue choices felt too brittle for my liking. My original planned implementation would

have allowed for more complex and interesting scenarios, but the code would have been very messy and would have added buttons that weren't in the game itself(adding a new panel to the bottom of the frame instead of having a dialogue box in the actual game), ruining the cohesive feel I was going for, so I opted for what I did here instead. The dialogue pointer solution worked alright, but if I would have added more options, it would have gotten messy as well. If I would have shifted the dialogue system to its own class and did a similar reading from a script like the Interpreter assignment, it would have been much more flexible.

For something that didn't get put in the game, originally the controls were going to allow the player to move and aim in different directions like a twin-stick shooter, but it didn't look good with the player's animations, so I took it out. The remnant of this implementation is seen with the variable names for the quest log, stats screen, inventory screen, and dialogue screen buttons.

I enjoyed working on the game, but being strapped for time meant I had to sacrifice complexity for functionality, but at least the game technically has everything I wanted. With more time I would have added more quests, items, and tiles to make for a more interesting gameplay experience since right now it's just talking to NPCs and battling slimes.

Project Conclusion

For this term project I implemented the Tank Game and the Second Game I got approval for(Action RPG) with all the requirements listed fulfilled:

Tank Game:

1.Tank Game must have 2 Players

2.Tank Game must have tanks that move forwards and backwards3.Tank Game Must have tanks that rotate so they can move in all directions. Note when only rotating left or right, the tank MUST NOT move forwards or backwards.

4.Tank Game must have a split screen.

5.Tank Game must have a mini-map

6.Tank Game must have health barsfor each tank

7.Tank Game must have lives count (how many lives left before game over)for each tank

8.Tank Game must have power up (these are items that can be picked up to modify your tank. What these power upsare, is up to you).

9.Tank game must have unbreakable walls

10.Tank game must have breakable walls

11.Tank Game must have tanks that can shoot bullets that collide with walls

12.Tank Game must have tanks that can shoot bullets that collide with other tanks.

13. Tank Game must come with a brief readme.txt file that contains what IDE the project was made in, the version of java used to build the game, the current working directory used for the game relative to the GitHub repo and how to run the game and the controls for playing the game.
14. Tank Game must be built into a JAR and stored into the correct folder in the GitHub repo. This will be the folder named jar.

Second Game (Action RPG):

1. The Player Character(PC) can move up, down, left, and right
2. The Game will have a mini-map
3. PC will have a health bar
4. PC will have a level and stats
5. Game will have unbreakable walls
6. Game will have breakable walls
7. PC will be able to shoot bullets
8. NPCs will populate the area
9. PC and NPCs colliding brings up dialog
10. Enemies that move horizontally or vertically.
11. Game will have at least two maps the player can move to and from
12. Bullets will damage enemies
13. Enemies will damage PC
14. PC will have an inventory screen with at least one quest item
15. There will be at least one dialog choice
16. There will be a quest screen with a description of at least one quest
17. Game will come with a brief readme.txt file
18. Game will be built into a JAR.