CMPT 276 Group 15 Phase 2 Report

Overall approach

Graves Robber has a well functioning first level designed for Phase 2 of CMPT 276. The software development team has successfully converted the UML diagram that was created in Phase 1 into functional Java class code. In the Adjustments and Modifications section there are more details about the changes from Phase 1 to Phase 2. When creating the game we started making small working sections of code and building from there. We created the game in the following order:

1. We created a board on GitLab to mimic an Agile Environment.
2. We created the map using only rectangles.
3. We allowed the main character to move through keyboard input.
4. We generated obstacles on the map.
5. We implemented enemies on the map.
6. We added treasures and traps to the map.
7. We added image textures to the map, characters, and items.
8. We polished the game to complete all requirements.

Preparation

At the beginning of the phase the team created a new Milestone in GitLab and used the board feature to keep track of tasks that the team needed to do and was currently working on. The goal was to recreate an Agile type environment by treating the issues as Stories and the phase as a Sprint.

First Stages

The team used the JPanel built in Java module to create the GUI of the game. At first we sketched out the game using coloured rectangles before we had any textures to add. This way we could get an idea of how the game logic was coming along earlier.

Keyboard Input

In order to control the character and menu items in Grave Robber, we needed to add in a Keyboard Listener, which can be seen in Figure 2. This allows users to control the movement of the PlayerActor and select whether to start or exit the game on loading screens. Players are then able to move around the map collecting hearts (treasure) and dodge zombies, skeletons and traps.

Adding Obstacles

Although the walls were drawn, the player could still walk through them. In the Level class the obstacleDetection function was created to detect when Characters attempt to walk through walls and it will prevent this from happening.

Adding Enemies

Two types of enemies were created for our map. The skeleton and the zombie enemies. Both were children of Character as seen in Figure 2. The zombie will run in a straight line until he hits an obstacle then randomly choose a new direction to run. The skeleton will follow the player and will only spawn once a treasure has been picked up on the map.

Adding Items

The game will end when the player's health reaches 0 and the player's health is directly affected by items. When the player picks up a heart, they will regenerate health. We used hearts as both health and our form of score. If the player walks over a trap, they will receive a hit to their health. As seen in Figure 2, treasures, bonus treasures, and traps are all children of the Item class.

Adding Images

The selection of images of sprites, items, and background images were chosen to best match the game's theme of being in a graveyard. The background images like the graves and the walls were also following this criteria as well. The sprites will rotate through a series of images to create the illusion of movement through animation.

Final Touches

Finally we added in the final touches of the game. We designed the level, added in music, and adjusted the amount of points different items would give the player. In this stage we also needed to fix bugs and clean up our code.

Adjustments and Modifications

The original UML diagram can be seen in Figure 1. In our original approach we were ambitious and envisioned a game which had additional rooms which required keys, doors, and weapons. We envisioned that the player would be able to attack the enemies, but in the end we decided to scrap these features in light of time and focused on fulfilling all the requirements from Phase 1.

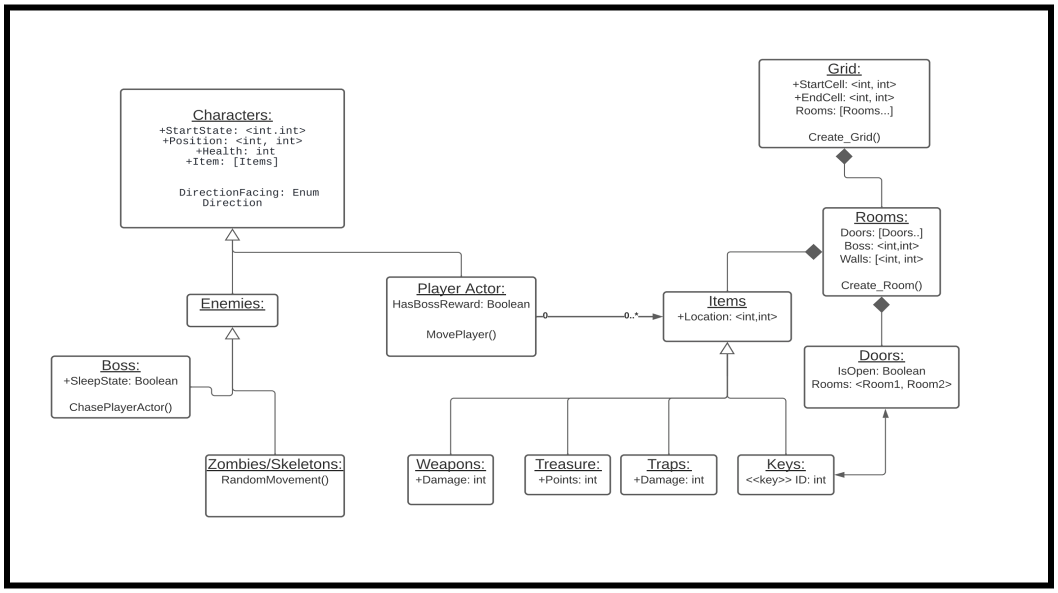

Figure 1: Phase 1 UML Diagram

The following list illustrates the classes from the UML diagram that were discarded completely:

- ☐ Characters
- ☑ ~~Enemies~~
- ☑ ~~Boss~~
- ☐ Zombies
- ☐ Skeletons
- ☐ PlayerActor
- ☐ Grid
- ☑ ~~Rooms~~
- ☐ Items
- ☑ ~~Doors~~
- ☑ ~~Keys~~
- ☐ Traps
- ☐ Treasure
- ☑ ~~Weapons~~

Below in Figure 2 is the revised UML Diagram. As you can see, the overall structure is somewhat similar, however the complexity is much greater than our original idea. New additions to not in the Phase 2 UML Diagram include:

- ➢ Zombie, Skeleton, and PlayerActor all share the parent class Character
- ➢ BonusTreasure was added as a child of Item
- ➢ UI, Keyboard, and Sound were added
- ➢ Level was added to store all level data
- ➢ CreateBackground was added to generate the background of the Grid
- ➢ Obstacle was added for walls and gravestones
- ➢ Skeleton and Zombie were given their own classes
- ➢ Grid now focuses on the size of the map that will be generated

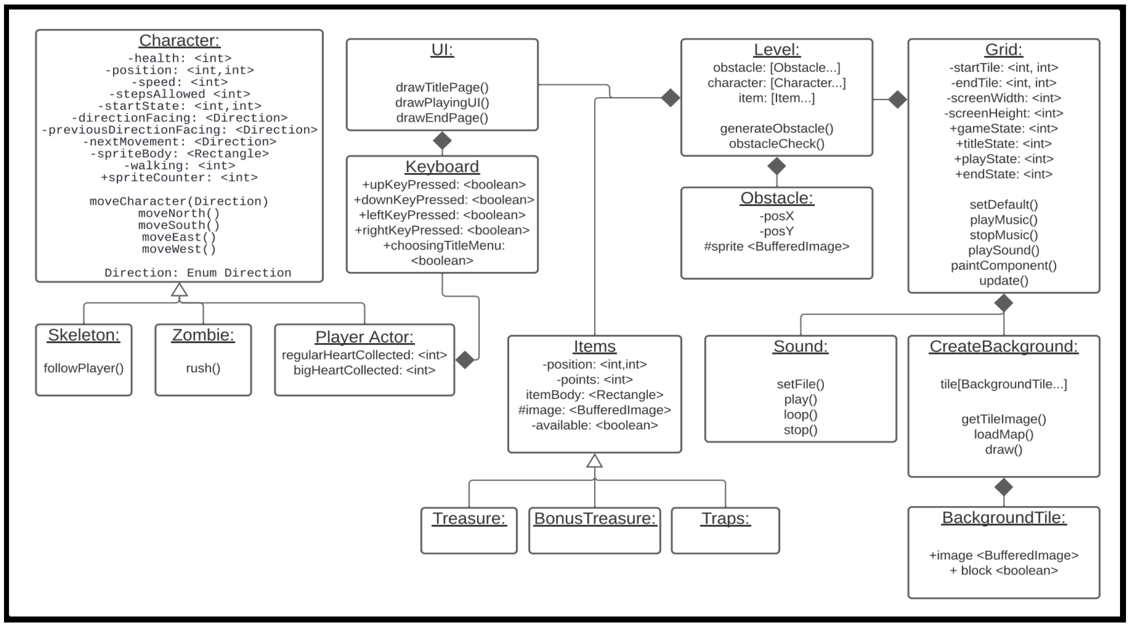

Figure 2: Phase 2 UML Diagram

The Division of Roles and Responsibilities

All members participated with the software development of the coding part of the project which made up the majority of tasks. Dylan took on the role of organizing the board and keeping track of tasks that were being completed. Some examples of how the tasks were divided can be seen below:

- Ling-Yu Keng (Software Developer, Designer)
  - Start and End Screen GUI Designs
  - Designed the Traps the on the map
  - Implemented Music and Sound Effects features
  - Created the system where players gain score to win the game
- Yuncong Chen (Software Developer, Designer)
  - Designing of obstacles on the level
  - Edited and inputted images into project
  - Writing of final report
  - Working with character animation
- Seenan Iftekhar (Software Developer, Quality Assurance)
  - Designed the enemies and algorithms for their movements
  - Created collision detection for characters hitting obstacles
  - Designed the generation of items on the map
  - Created the level class creating better organization from Grid
- Dylan Rowsell (Software Developer, Quality Assurance, Scrum Master)
  - Implemented the movement system (Characters moving one tile per tick)

- Created Treasure and Bonus Treasure (Spawning of Treasures periodically)
- Created overall Grid system responsible for drawing/updating objects on the map
- Writing of final report

External Libraries

- ➢ java.awt.image.BufferedImage
  - Used for loading in sprite images
- ➢ java.io.IOException
  - Used for catching possible system crashes
- ➢ java.util.Random
  - Used for creating random variables
- ➢ javax.imageio.ImageIO
  - Used for reading image files
- ➢ java.util.ArrayList
  - Used for creating arraylists
- ➢ java.text.DecimalFormat
  - Used for formatting time
- ➢ java.util.Objects
  - Used for dealing with null objects
- ➢ java.awt.event.KeyEvent
  - Used for listening to keyboard inputs
- ➢ java.awt.event.KeyListener
  - Used for listening to keyboard inputs
- ➢ java.io.InputStream
  - Used for reading files
- ➢ java.io.InputStreamReader
  - Used for reading files
- ➢ Jpanel javax.swing.*
  - Used for drawing GUI
- ➢ java.awt.Rectangle
  - Used for calculating player intersection
- ➢ javax.sound.sampled.*
  - Used for importing sound
- ➢ java.net.URL
  - Used for linking sound
- ➢ Maven-compiler-plugin
  - Support for Maven
- ➢ Junit
  - For use in Phase 3

Measures

To improve the quality of our code we took special care to comment all of our code so that documentation could be generated with JAVADOCS. We also practiced using encapsulation in our program using getters and setters where applicable. Also when possible we used inheritance to simplify similar classes.

Challenges

The team faced many difficulties during the progression of Phase 2. We encountered bugs with the characters movement which was sometimes difficult to solve. We also struggled to find common times for meetings. This is because the team is composed of students and it is difficult to find a common time when everyone is free. Another challenge was when working on another teammate's code and trying to understand their process.

References

Pipoya (n.d.). Pipoya free RPG character sprites 32x32 by Pipoya. itch.io. Retrieved March 19, 2022, from https://pipoya.itch.io/pipoya-free-rpg-character-sprites-32x32

Cainos (n.d.). Pixel art top down - basic by Cainos. itch.io. Retrieved March 19, 2022, from https://cainos.itch.io/pixel-art-top-down-basic

OpenGameArt.org. 2022. Animated Traps. [online] Available at: <https://opengameart.org/content/animated-traps> [Accessed 19 March 2022].

小森平的免費下載音效. (n.d.). Retrieved March 19, 2022, from https://taira-komori.jpn.org/freesoundtw.html