CMPT 276 Project - Group 17
Phase 2: Implementation Documentation

Created By:
Jonas Lam
Yanjun Qian
Brendan Shen
Sheila Nicholson

Overall Approach:

One overall theme the reader will notice throughout this document is that most of the details are contained in the 'Planning/Timeline' section. It is thus highly recommended that the reader thoroughly familiarize themself with said section in order to be able to gain a better understanding of how this group navigated phase 2 of this group project.

We split up responsibilities into what seemed at the time of planning in phase 1 to be reasonably equal levels of difficulty (detailed in 'Planning/Timeline' below), but gradually increased cooperation across individually assigned areas of responsibility as the deadline approached. We would work concurrently on our own branches, with the initial implementation and addition of basic structures (think constructors, getters, setters) based on the UML diagram we had from phase 1 (which was pretty comprehensive), and trimmed/tweaked/modified it as necessary (more detail in 'Notable Changes from Part 1'). Frequent communication allowed not just for quick troubleshooting and prompt resolutions to any problems that appeared, but also feedback on brainstorming or ambiguity when these opportunities presented themselves.

This was followed by building the environment first (see resources -> maps), then the hero (controllable character), then the items and enemy logic were implemented simultaneously, with constant updates to improve functionality.

Notable Changes from Part 1:

- **For the UML diagram**, an enum for PunishmentType and RewardType was added to the class folder for items in the code.
- For 'Tile', MysteriousSmokeTile and TileManager were also added in order for the game to function as intended.
- Sprites and Key were also added.
- GamePanel, AssetSetter, CollisionChecker, UI and UtilityTool were added without any specific folder classification.
- Owing to graphical difficulties, we decided to change the 'Racoon' to a 'Bear'
- The 'TimeElapsed' class was not needed, the timer was added and implemented within UI.java.
- HUD, Game, and MainMenu were not needed.

- **For the UI:** The map looks slightly different as there was a dead zone at the top where the hero could theoretically get teleported to by RNG by the vortex - it has since been removed.

- For the main menu pages and the postgame windows (basically all the images on page 9 and 10 of our phase 1 PDF submission excluding the one snapshot of what the map actually looked like), it is critical to note that this was generated during phase 1, which emphasized design. We mostly focused on backend implementation instead of worrying about the frontend. This could be changed and upgraded in phases 3 and 4 of the project.
- We contravened the statement in phase 1's instructions that the 'final score and time are shown for a winning player' - we extend this to any instance after a player has finished, that is, even if they lose they will see their final score and their final time.
- We have added some text that will briefly appear in the middle of the screen/map, the purpose of which is to indicate to the player that they have made contact with a certain item (and will describe the effects, such as books -> -points or bed -> +points).
- We have also put the score and the timer (both of which will be constantly updated during runtime as necessary) at the top of the map, which wasn't present in the phase 1 mockup.
- The main menu screen was altered to only have a red 'exit' button instead of using symbols as indicated in the phase 1 document. We also did away with the options button as that was deemed unnecessary. Furthermore, the coloured text for each of the levels was removed and replaced with generic text, along with the removal of the infinite level as that was deemed to be over the top.
- The 'options' menu described in the use case has not been implemented. This is also unnecessary to the core functionality of the game as described above.
- **In general:** Despite not being an explicitly mentioned requirement, we decided in phase 1 and intend to make multiple difficulty levels.
- We implemented the 'medium' difficulty first since the template was already existing from the phase 1 planning phase.
- Sound and visual effects are deemed 'extra' and owing to our busy schedule, may be implemented in future phases. It has not been implemented as of yet.

Planning/Timeline:

- We would frequently discuss and meet in person after class or virtually over our discord chatroom, as such not too much formal structure exists when it comes to deadlines. The two informal deadlines that are discussed below are the 'mid-point' deadline on March 4th, and the actual phase 2 'full game' deadline of March 18th.

The general responsibilities were assigned as follows:

- ***Tiles + Map -> Brendan***
- ***Items -> Sheila***
- ***Characters -> Yanjun***
- ***UI, Sprites + Documentation -> Jonas***

Owing to a strong working relationship, and the recognition of all group members as equals, our management process consisted of everyone working somewhat independently on their assigned responsibility, with any difficulty encountered promptly resulting in requests for assistance from other group members to rectify the problem.

Our work commenced on February 20th with a virtual meeting during reading break, the gist of which involved the division of responsibilities, and exploring how our UI/GUI would function. External libraries were also researched (java.awt.Graphics is a big example). Brendan created the skeleton classes based on the UML diagram we created in phase 1 with input from Yanjun and Sheila, and the next week was spent researching and configuring directories in Apache Maven.

Another meeting occurred on February 24th, during which the group investigated how to tinker with jpanel and successfully completed the reformatting of suitable 2d pixelated images necessary to represent the various items and characters in the game. While an after lecture meeting was planned on February 27th, this was aborted due to the consensus being everyone would benefit from having more time to investigate and research their own part. Jpanel and Jframe, along with the character sprites were discovered and successfully added to the project the next day, along with ways to connect the backend to the frontend (UI). Goals were set to implement the basic functions outlined in the UML diagram along with the importation of the UI (due to Jonas encountering difficulty with it) before the mid-way deadline. Importantly, these goals were met, however progress on the project stalled (and thus the week beginning on March 10th would entail an accelerated work pace to catch up) owing to the first midterm in this class among other conflicts in everyone's schedule.

A meeting was planned after the midterm on March 8th in person to discuss the current state of the project and next steps. It was agreed that by the end of the weekend, an executable game that followed the contours of the map generated for phase 1 (medium difficulty) along with item placement and a moveable, controllable character would be in place - which was met owing to the hard work of all group members and assistance from online resources. Implementing item collision was a bit tricky, but after that was completed a temporary message would be printed indicating to the user that a collision occurred, which would be improved in future patches to say punishment/reward collected.

We encountered difficulties with the algorithms pertaining to the enemies since our initial map creations led to spaces being 1 tile wide, with the enemy for each difficulty level not being able to recognize it could fit through. As such, another few days resulted in not just more progression when it came to the code, but also a revamp of the maps (medium looks quite similar, but the initial structure of easy and hard was totally changed to accommodate this restriction that all paths for the characters to walk through had to be minimum 2 tiles wide). It was also during this last week when we changed the default plain cheese coloured yellow tiles to something which more resembled a pixelated 2D arcade game which had elements of RCB to it (old, rusty, perhaps even falling into disrepair or in need of renovations). Numerous improvements to the individual sprite images for the items and characters made the game look significantly better than before as well.

The final stretch of coding from Friday through the weekend leading up until Monday (the due date) saw some tweaking with the messages which initially told the user how many more items they had to pick up before they could ‘escape’ or win the level, along with the implementation of a message which would indicate to the user that since not all the beds were collected, they could not ‘exit’ the level through the door in the bottom right.

External Libraries:

- javax.imageio.ImageIO: Necessary for core functionality of this game, lets us load images from files and save images to files (read + write)
- Javax.swing: Gives us access classes that lets us build GUI (like jframe)
- Java.awt: Supports jframe with classes to manage GUI parts like buttons and windows.
- Java.awt.event.ActionEvent and Java.awt.event.ActionListener: As the name implies, action listener checks for when an action event happens (like clicking a button).
- Java.awt.image.BufferedImage: This lets us manipulate image data.
- Java.io.IOException: Allows for IO errors to be handled. This is present throughout our code.
- Java.util.Objects: This gives us access to methods when it comes to manipulating objects such as null check.
- Java.util.Random: We need this to make one of our bonus rewards, the A+ paper, be able to teleport to random places.
- Java.util.concurrent.ThreadLocalRandom: Similar to Java.util.Random above but this is for multi threaded/concurrent situations.
- Java.awt.event.KeyEvent: Handles keyboard input.
- Java.awt.event.KeyListener: Not only handles keyboard input but specifically allows for the methods keyPressed and keyReleased to be used (without which a game would be pretty lame).
- Java.util.ArrayList: Allows for the implementation of a dynamic array in Java - Sheila’s corner.
- Java.io.InputStream: Since this is the superclass representing an input stream in bytes, this means we used it to read data from files.

- Java.io.InputStreamReader: Allows for bytes to be read and decoded in characters, which bridges the gap between bytes and characters.
- Java.awt.Color: Allows for the use of RGB for graphics.
- Java.awt.Font: Allows for the use of fonts in graphics (style, size and typeface).
- Java.awt.Graphics2D: Provides specific functionality support for text layout and coordinate transformations (tileSize * 16 as an example, would be difficult in the default Graphics class, so 2D extends this).
- Java.awt.DecimalFormat: Allows for the formatting of decimal numbers, used to truncate the timer which runs during gametime so the number wouldn't be 13.05652…seconds.

Code Enhancement:

- Specific details were listed in great detail above in the section titled 'Notable Changes from Part 1'. Manual code reviews (with crude, rudimentary requests in the group chat for others to look at a certain piece of code being the only way our group did enhancement for this phase) took place often, with the feedback we'd give each other allowing everyone to benefit from this win-win arrangement.

Challenges:

- Item resolution was not acceptable, new images were added.
- Tile image was updated to more reflect RCB 'vibe'
- CollisionChecker was very buggy, thus it took a lot of time to resolve this.
- Main enemy (boss) of each level was too fat and/or too dumb (the algorithm wasn't perfect TLDR) to fit through one tile wide spaces, so a major overhaul of each of the difficulty level maps had to be done to make all spaces at least 2 tiles wide.
- File pathways for the images to be loaded and recognized by our code led to some troubleshooting and late night debugging sessions.
- Project structure with respect to resource directories and how maven organized it provided a learning experience for the group to figure out why exceptions were being thrown (IllegalArgument to be specific).
- **Note: The requirements in the phase 2 document states that this document should be at most 4-5 pages. This report is 5 pages long if you exclude the title page, which really isn't a page, more so of a cosmetic addition to provide structure to this document.**