THE WIZARD

GROUP A.
Mark Lucernas
Tamer Elsawaf
Jordan Tobin
Jennifer Nguyen
Seth Steen-Fuentes

CISC 191

1. **Team introductions and roles**

Jordan: Refactoring, game design and level system.

Mark: Unit testing, UML class diagrams, Maven management, and GUI development.

Seth: Event handling, design management, structure management.

Tamer: Functionality, resizing/optimization to fit all devices, Save/Load serialization assistance, GUI and graphics for help menu, background menu and throughout the game.

Jen: Graphics, image and file management.

2. **Project summary - What does your project do?**

The Wizard is a top down shooter, the player progresses through various levels, facing a variety of enemies. Functionality exists for sound effects, save/load, pause. The game is also open to further expansion.

3. **Methodology - What topics that you learned in class were used in the project?**

A thorough knowledge of Java including inheritance, polymorphism and abstract classes is a prerequisite for developing the codebase. The topics of threading, serialization, linked lists, GUI's and file handling IO were critical in the development of the application. Threading is used to manage the tick/render of the game as well as multi-threading for the loading screen. Serialization is utilized in the save/game function. Linked lists are used in all aspects of the game handling classes. GUI's are the backbone of the gameplay functionality, and file handling is utilized in loading sprites and sounds effects, as well as processing serialized files.

4. **Results - Does your app work? What features are included? What features were dropped or not finished? Why?**

The app works with all functionality suggested at the start of planning. However, further builds were dropped, for example an in-game economy. Different weapons, further levels and gameplay. These were dropped in order to focus on refactoring the code base over gameplay. Initial project structure required refactoring to incorporate a more object-oriented programming paradigm.

5. **Live Demonstration or Videos/Screenshots**

https://www.youtube.com/watch?v=Xq3uBXrzOb4 - Playthrough/Code explanation

6. **Reflections**

What went well?

We communicated very well and everyone was able to work through their own branch without stepping on each other. New button layouts, action events and action listeners, the ability to pause, quit and access the help menu. The resizing and optimization was accomplished, the save and load features took working together but was accomplished.

What could be improved on?

The codebase itself could benefit from additional refactoring in areas. Introduction of more abstract classes to organize class structure and refactoring of code into other classes to create more well structured project. Methods could be narrowed down and respecified. More sophisticated threading could result in smoother gameplay, as well as proper object management which keeps the overhead lower. Aside from backend design, the gameplay itself could have further functionality and actually develop the plot, mechanics and graphics. Implementing user profiles is a potential feature as well.

What features, if any were dropped from the initial proposal, and why?

All the proposed features have been created, however we would like to add further functionality.

What features would you still like to add?

An ingame economy. NPC characters, more weapons, further sprites and actions such as doors, levers etc. Different enemies and additional enemy movement. Adding the ability to customize your character and choose which levels you want to play on based on ones you have already completed. The ability to change your difficulty (i.e. more enemies and less starting health/ammo).

7. **Conclusions - a concise summary of all the above:**

Overall we were able to accomplish everything that we proposed, there were some obstacles but we were able to communicate effectively and work together to get by them. For example, the Save/Load function took a bit more time but was able to be done by working together and everyone doing their part in implementing the code and asking for help when they got stuck. One thing that could be improved is the code itself,

although we refractored and tried to use efficient calling, some methods could be more specified. Some of the features that would have been nice to add is an ingame economy, the ability to switch players, weapons and choose which level and difficulty you want to play.

8. **Codebase (link to source repo, zip of structure, or executable .jar, javadocs.jar, and sources.jar)**

   https://github.com/MiramarCISC/CISC191-SUM20-A

10. **References:**

https://www.youtube.com/watch?v=e9jRfgjV4FQ&t=3s - Tutorial for programming a basic top down shooter. Utilized for help with the collision system, camera and sprite rendering.

https://github.com/AlmasB/FXTutorials/blob/master/src/com/almasb/tutorial9/Main.java - Utilized for help with serialization.

https://www.ntu.edu.sg/home/ehchua/programming/java/J8c_PlayingSound.html - Used for help with Sound Effect class.

https://github.com/iyyel/celestialoutbreak - Used for help with structuring code base in a more Object Orientated way.

https://git-lfs.github.com/ - Used for handling .wav files.

https://github.com/FreeCol/freecol - Inspiration for Actions package.

https://www.youtube.com/watch?v=2E3WqYupx7c&list=PLqq-6Pq4lTTa4ad5JISViSb2FVG8Vwa4o&index=1 - JUnit5 Basics tutorial series for unit testing.

https://www.youtube.com/watch?v=-3KjiEga-cI - Java 2D animation tutorial used as a reference in building the loading screen or LoadPanel.java.

https://stackoverflow.com/questions/32853110/jframe-size-resize-fullscreen - Helped for reference with resizing