Implementation

3^3 Studios | Team 27

James Hooper, Daniel O'Brien, Connor Davies, Joe Krystek-Walton, Mitchell Simpson, Yiyang Ye

Implementation

Introduction

Within this document will be outlined the process of implementation, with additionally reference to features required that were not fully implemented by the deadline of the project.

For fully documented code, please refer to the Javadocs generated for the project.
This can be found at the following link: https://threecubedstudios.github.io/auber/javadocs/.

Evaluation of Requirements

The specific features required by the client have all been met and thoroughly tested/documented, as well as most of the requirements set out in our requirements process. However, there were some requirements that could not be implemented within the given time frame, or which couldn't be solved by anyone in the group in the limited time available. Issues link: https://github.com/threecubedstudios/auber/issues.

Animated Doors (referenced by the requirement **FR_ANIMATED_DOOR** within **UR_MAP**) is one example of a feature where there was insufficient time to implement, as it would require either a new event inserted into the GameScreen render loop or code added to GameEntity render loop that checks for nearby door collision regions (which would have to be added to the object layer of the world) and updates the doors if need be.

The minimap implementation was an idea brought upon us from the interview with the stakeholder (referenced by **FR_MINIMAP**, part of **UR_MAP** within the Functional System Requirements) - initially we wanted multiple images representing sections of the ship cycling for ease of implementation; this was determined to be too much work especially given the time frame. We then decided on an arrow system, in addition to a list of systems that turn red when attacked and are removed when destroyed.

Respawning was a highly debated feature in our group to implement - (**FR_RESPAWN** within **UR_PLAYER**) - once the player loses all their health the game ends. The group settled on no respawning allowed, as allowing the player to respawn would make it much harder for the player to lose. This was decided with the help of play testing which determined that the game was too easy to play with the current implementation of respawning allowed. Moreover, due to time constraints the implementation of a fully functional respawn system would have delayed the process of developing the key system requirements set out by the stakeholders.

The ability to teleport to any pad of your choice from the current pad the player is on would've been a useful implementation for first time players, however due to the fact that the player can already teleport to the infirmary from anywhere and the size of the map means that the user doesn't necessarily need to use teleporters to reach key systems being attacked in time. (**FR_TELEPAD_DESTINATION** within **UR_PLAYER**)

Bibliography

https://codegym.cc/groups/posts/182-java-game-programming-for-beginners-where-to-start
Justification: Beginner Java game design tutorial

https://libgdx.badlogicgames.com/
Justification: Game engine built for Java

https://acodez.in/12-best-software-development-methodologies-pros-cons/
Justification: Research on different software methodologies in order to help us understand which to choose

https://www.thedesigngym.com/seven-principles-of-game-design-and-five-innovation-games-that-work/
Justification: Principles of game design

http://zetcode.com/javagames/
Justification: Some basic information/tutorial on Java 2D games development

https://www.tutorialspoint.com/uml/uml_basic_notations.htm
Justification: Tutorial on UML notation

https://www.vogella.com/tutorials/JUnit/article.html
Justification: Information on JUnit for unit testing

AI and Games - YouTube
Justification: Information regarding AI, to help with implementation of AI functions within the game

Gradle
Justification: Java project build engine, useful for managing versioning, unit tests etc

https://gist.github.com/antimatter15/822345
Justification: Example of the simple game

https://www.youtube.com/watch?v=RGOj5yH7evk
Justification: Video crash course on Git and GitHub

https://zenkit.com/en/blog/agile-methodology-an-overview/#:~:text=Agile%20methodology%20is%20a%20type,functional%20teams%20and%20their%20customers.
Justification: Overview of Agile Methodology