Name of project: Dungeon Crawler

Names of all team members: Anna Rahn, Jason Mu, Sam Carruthers

Final State of System Statement
A paragraph on the final state of your system: what features were implemented, what features were not and why, what changed from Project 5 and 6

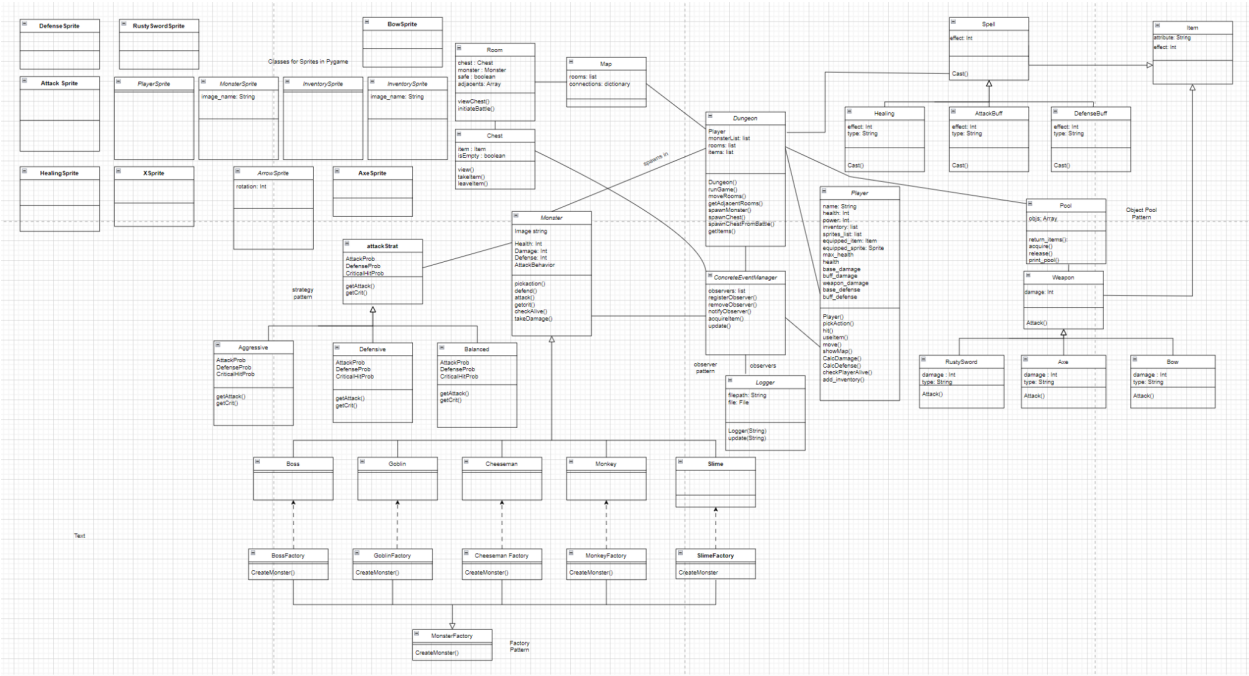
When the user launches the game, they first see the title screen. They can press play to enter the game (instructions on how to play included in the readme). The player spawns in a safe room. Each safe room has a chest with an item they can acquire by clicking on the chest to open it. They can also check their inventory, which tells them their currently equipped item and current stats, switch their currently equipped weapon, use a spell, or click on an item to view its stats. At the upper left corner, there is a map of the dungeon. Grey cells have not been explored yet, blue is safe, and red is a monster room. There are four arrows the player can click - each will move them to another room on the map (unless they are along the border, in which case their choices will be reduced). Upon moving rooms, the player has a random chance of spawning in another safe room, which has the same functionality as the starting room, or in a monster room. In monster rooms, they enter a turn-based battle. The monster's attack, defense, and battle strategies are all randomly generated. The player, on their turn, can do one of these actions: run (which has a random chance of being successful and moving the player to a random adjacent room, which takes a turn), use a spell from inventory (which does not take a turn and only is effective in this room), change the equipped item, or attack the monster (which takes a turn). The player must defeat four monsters and a fifth boss (automatically spawned after four monsters are defeated) to win the game.

If we had more time, we would like to implement a status bar for battle that represents the player and monster health, some type of indicator for how strong a monster is (such as size, a display in the UI, or color code), critical hit probability for weapons, items with effects that increase with number of monsters defeated, more visual/audio feedback when a user clicks buttons, and unit tests through Pytest. Since we submitted our original design in project 5, we have defined the map more precisely using a coordinate system and created an Items class that spells and weapons now inherit from. Since our update in project 6, we have eliminated all abstract classes (specifically for eventManager and observer), decided to use the player class to store information on the currently equipped weapon, and added many more sprites.

Final Class Diagram and Comparison Statement

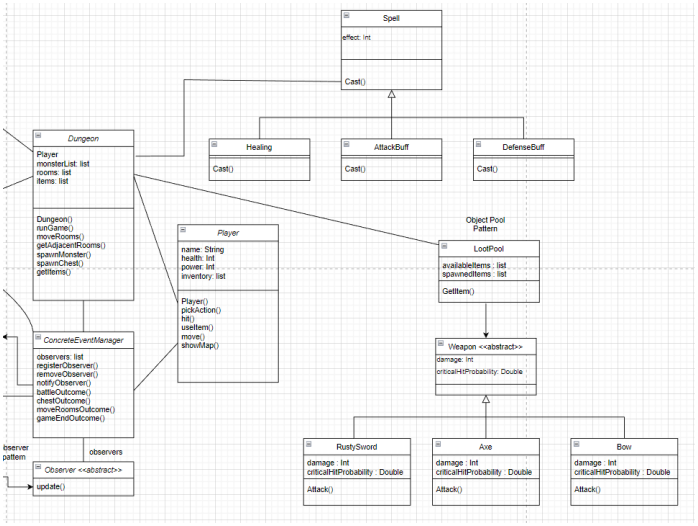
A thorough UML class diagram representing your final set of classes and key relationships of the system. Highlight and document in that diagram any patterns that were included (in whole or part) in your design. Include the class diagram submitted in Project 5, and use it to show what changed in your system from that point into the final submission. Support the diagrams with a written paragraph identifying key changes in your system since your design/work was submitted in Projects 5 and 6

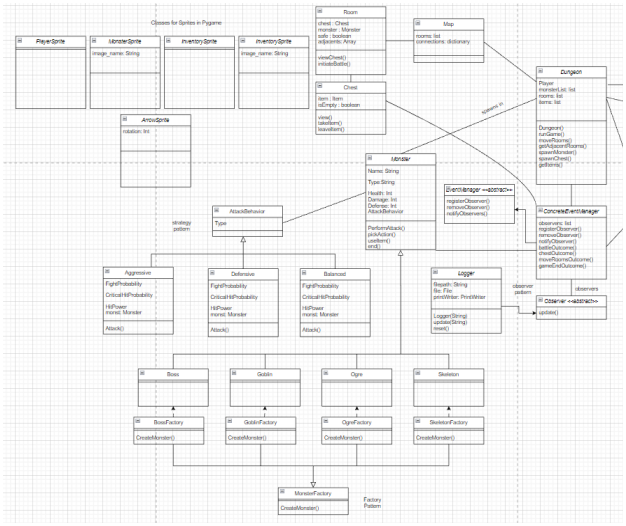
New Diagram
https://drive.google.com/file/d/1x8PwVCxOtDwCFdSjgD8FVYPjWmEfL6GE/view?usp=sharing



Since we submitted our original design in project 5 (see below), we have defined the map more precisely using a coordinate system and created an Items class that spells and weapons now inherit from. Since our update in project 6, we have added many more sprites - this is because we increased the number of items in our object pool and the number of elements in our UI (such as the campfire on the title screen, the X button on the inventory, etc.). Each sprite must be its own class, which adds a lot to this diagram. Additionally, we have removed all abstract classes, particularly the eventManager and observer classes. Since we had only one of each, we did not need any standardization between these object types, so found these abstract classes to be unnecessarily complexities. Finally, we used the player class to store information regarding the currently equipped weapon, which added a lot of attributes.

Old diagram (for larger images, see our code repo)





Third-Party code vs. Original code Statement

A clear statement of what code in the project is original vs. what code you used from other sources – whether tools, frameworks, tutorials, or examples – this section must be present even if you used NO third-party code - include the sources (URLs) for your third-party elements.

Most credits are given in comments at the top of the relevant file, so we have compiled our sources into the following list:

- Visualization Framework: pygame
- Musics
    - Battle Rooms: Hitman by Kevin MacLeod
    - Title screen, normal and chest rooms, win screen: My Dark Passenger by Darren Curtis
- Tutorials
    - Pygame setup and sprites: realpython.com
    - Adding background images in pygame: Stack Overflow
    - Mouse position and cursors in pygame: pygame.org
    - Object Pool: sourcemaking.com
- Graphics
    - Figma: used for prototyping UI
    - Pixel Art: used PixilArt.com to draw our own assets
        - What we drew: Arenas, monsters, player sprite, closed and open chest, spells (healing, attack, defense), weapons (rusty sword, bow), title screen, losing endscreen
    - Campfire on title screen
    - Win endscreen with mountains

    - Final Boss from https://pixelartmaker.com/art/158aec029546e94
- Font: ARCADECLASSIC

All other code and design is our own.

Statement on the OOAD process for your overall Semester Project
List three key design process elements or issues (positive or negative) that your team experienced in your analysis and design of the OO semester project.

Drawing and updating class diagrams was a key positive element that we used during our design process. It helped us to think critically about OO patterns and other code decisions before immediately jumping into code, was a very helpful reference once we started working on the different features, and gave each team member a common reference point for how certain elements should be implemented.

We also learned a new framework for our visualizations - pygame, a Python package, helped us create an interactive UI. None of us had used it before, so figuring out how it worked with our diagrams at the beginning of the design process was a challenge, but we ultimately really liked this package for its versatility, ease of setup and use, and range of customization. We learned a lot about how to incorporate this framework into our overall design and will have a better idea in the future of how to incorporate the learning curve into our project timeline.

Github was very influential in our design process as we were careful to use it for division of labor while also keeping the main branch intact and bug-free. When we split up features of the project, we worked on our own feature branches for those updates, and made sure that the code was reviewed before merging a pull request. This helped a lot with being able to work simultaneously, and overall it went pretty smoothly in assisting our workflow and how we split up features in our design process.

Contributions
Listing what each team member contributed to the project.

Anna Rahn: pygame setup, UI (title screen, inventory items list, arena and buttons), battle (attack/defend), inventory (acquire items, graphics), items (object pool pattern), observer pattern, music

Jason Mu: monsters (pixel art, attack strategy pattern, factory), battle (running, attack and defend buttons)

Sam Carruthers: rooms (class, map/arena/arrow graphics), map, inventory (select item, equip item, use item, UI), monsters (stats)