OOAD - Project 7

Chelsea Stockberger, Bella Barbieri

Froggy Forest

Final State

Game Description:

Froggy Forest is a roguelike pixel-art RPG in which the player must defeat all enemies in one level in order to progress, and defeat the final boss to win the game.

Game Features:

- All art and music was made custom.
- Randomly selected tilemap loads
- Movement and collision with tile edges
- Ability for player to attack and defeat enemies, and vice versa
- Objects placed randomly on tilemap
- Player able to upgrade weapon or obtain items from objects
- Graphical updates based on game events (ex. Chest opens once interacted with, enemy image changes upon attack, etc)
- Level progression with harder enemies
- Custom start game screen, and end game screen
- Design patterns State, Singleton, Game Loop, Factory, Flyweight
- Music loops, changes during boss battle
- Boss battle with shooting projectiles able to be reflected
- Blocking system, knockback system
- Projectile system for player when they get magic staff

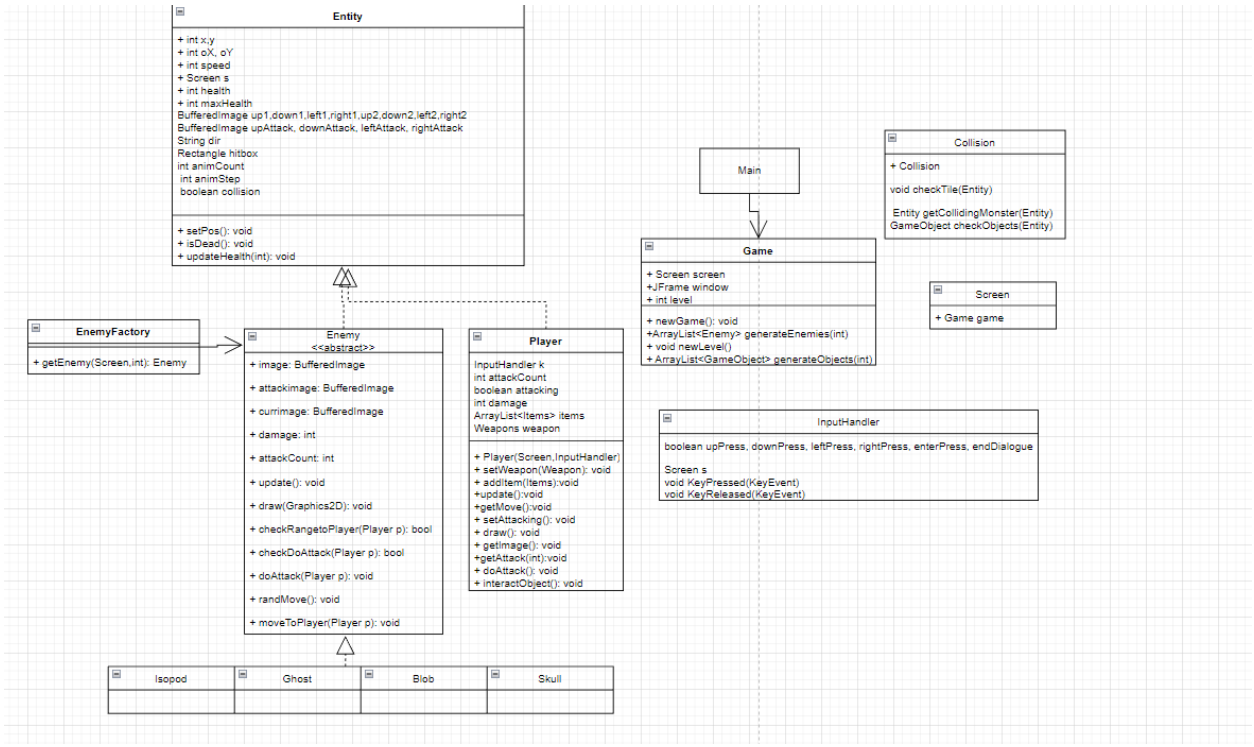
Since project 6, design patterns State and Flyweight have been implemented. Boss battle and game-ending circumstances have been implemented. Further graphical additions have been added. More tilemaps have been added. Projectiles have been added. Flyweight has been implemented to create tiles.

In project 6, a goal to create NPCs was established. This was dropped due to time and prioritizing code organization and design patterns. Also, it was unplanned to

implement Flyweight for Tile data creation, but implementing it created much more efficiency and allowed for removal of many inherited subclasses of Tile.

Final Class Diagram + Comparison Statement

Project 5 Diagram:



Final Diagram:Too large to view on page, please visit link: https://drive.google.com/file/d/1qexRSrO5VL0E_3pE1iY9UkceOr9tx_ak/view?usp=sharing

Our diagram has changed a ton since project 5, since we didn't realize how many classes we would end up needing to add!

As you can see, the factory design pattern is clearly shown in the diagram. The other patterns we used don't show up in UML as clearly.

Third-Party Code v.s. Original Code

Tools Used:

- Java Swing (JPanel)
    - UI design, graphics
- Runnable
    - Interface for thread execution
- Sound (AudioSystem, Clip, etc)
    - For controlling music

Resources References:

- To figure out how to add Audio
    - https://stackoverflow.com/questions/4875080/music-loop-in-java
- To learn about JPanel and Graphics
    - https://www.bogotobogo.com/Java/tutorials/javagraphics3.php

OOAD Process

When designing our project, it was unknown the complete range of features and functionality to be implemented. It all started from the idea of making a rogue-like level based game with a boss battle to finish it off.

One key design process element was to ensure time was an element in our game. Time allows us to display certain graphics for only a certain amount of time, run functionality based on time, and is essential to our game. It helped having this concrete functionality when designing everything in our game.

When adding new functionality, since it was not accounted for early on, it meant having to modify many different elements in multiple classes. This was indicative of poor design structure, and through work on our project, it proved helpful to constantly optimize and organize code, classes, and how it all works together in order to avoid that problem. This process was positive, seeing as it has assisted me in looking more ahead when writing and planning programs, as well as fixing programs when they are designed poorly initially.