

CMPT 276 Phase 4 Report

By:

Nathan Chan

Michael Chan

Sri Srisathanantham

Shawn Xie

Due: Sunday, December 8, 2023

Semester: FALL 2023

The Game

Description:

“DA ONE PIECE” is a 2D maze game written in Java following Object Oriented Programming principles. It was developed and refined with techniques and ideas learned from lecture such as proper version control and refactoring methods.

Unchanged:

The overall structure of our game remained the same from when we designed it in Phase 1 of this project. The idea of class hierarchy and subclasses inheriting off of them was maintained. For the most part, the classes we implemented and their associated subclasses also remained the same. As for the core mechanics of the game (i.e. rewards, punishments, moving enemies etc.), those remained the same as well.

Changed:

Although we stayed faithful to our UML document for the most part, there were some changes we had to make, either because our initial plans didn't make sense (like having punishments inherit our rewards class), or because we found an easier way to do things (like getting rid of the movement interface and using a keyhandler class instead).

Thematically, we had to deviate from our original plan. Whilst the title of the game is still “One Piece” related, it was challenging finding complete art assets. As such, we had to use art assets that were complete and easily accessible (like the ones we are currently using).

In our initial plans, we had wanted to implement many features such as randomly generated mazes, ChatGPT integration (for NPC dialog), items for the main character to use, and multiple levels. However, due to time constraints and difficulties in implementing other features, we had to significantly narrow down the scope of our game, only managing to implement randomly generated mazes and hardcoded NPC dialog.

Lessons Learned:

Over the course of this project, we learned incredibly important lessons from our various trials and tribulations:

1. Don't rush the planning phase and be too eager to start coding

By rushing the planning phase, we found that we were left without a concrete plan so everyone just began implementing classes their own way. This led to a very incoherent structure that quickly became very convoluted and implementing features like collision became very difficult.

We learned the importance of slowing down our eagerness to code in order to make a well thought out plan.

2. Communicate with teammates constantly





In part due to poor planning, we had ended up implementing classes assuming a specific feature of the game, like collision, would be implemented in a specific way (i.e. box collision, 2D array etc.). This happened for a variety of different concepts, like map generation. Because of this, our individually implemented classes ended up conflicting with each other and implementations didn't end up working or ended up doing too much (the blob anti-pattern). This resulted in many rewrites and scrapped classes. We learned from this the importance of talking with each other frequently and not assuming that another person is automatically on the same page as you.

3. Enforce consistent coding standards

Coming from four different programming backgrounds, it was no surprise that we'd have four different coding styles. Because we did not enforce a single coding standard, different classes began to take on different shapes, and because of this, helping each other with implementations of classes became difficult. Any differences, from variable names to even where you put your setters/getters, added to the overall overhead of cooperation. We learned that the next time we work on a project, it is paramount that we decide on a coding style and stick to it, so that the readability of the code and our ability to jump between classes increases.

Tutorial

Controls:

Key	Explanation	Screenshot
W	Moves the player character upwards	
A	Moves the player character left	
S	Moves the player character downwards	
D	Moves the player character right	

Shift	The player character sprints in the current direction	N/A
O	Quits the game	N/A

How to Play:

The main goal of the player is to escape the maze. They can do this by collecting all the gold coins on the map. After they collect all the gold coins, a green portal will open, allowing them to escape and win the game. Stopping the player from doing this are stationary traps, which will immediately kill the player, and moving enemies that will subtract score from the player. Should the player's score drop below 0 as a result of touching the moving enemies, the player will die. There are also bonus beer drops scattered around the map. Although they are not necessary to escape the maze, they will offer a significant boost to the player's score.

Screenshot	Explanation
	Collect all gold coins to escape the maze!
	Moving enemies will subtract the player's score until it hits zero. Don't let that happen or you lose!
	Touch a stationary trap and you automatically lose!
	Collect these optional beers and boost your score!
	Once you collect all the coins, the portal opens up and you can leave!