1. Team Members

Cao Chen UID:3036100507  
Hu Jiarui UID:3033108211  
Xiao Donglu UID:3036100820  
Xie Yadong UID:3036052231  
Yang Liuhaichen UID:3036101642

1. Description and Rules of the game

2.1 Game Structure

We create a main class Maze-game to handle the overall game flow, player state, and interactions with the smaller games.Some rooms will contain smaller games (game1, game2, game3, game4, game5), and some rooms will be empty or contain other elements like monsters, treasures, or health potions.The player will navigate the maze by moving between connected rooms.Diagram

Description automatically generated

2.2 Norse Mythology Theme

Set the game in the world of Norse Mythology, with locations inspired by the Nine Realms (Asgard, Midgard, Jotunheim, etc.).The player takes the role of a Norse hero on a quest to retrieve a powerful artifact guarded by monsters and challenges.

2.3 Player State

We implement a Player class to store the player's current state, including health points, weapons, and current room.Health points decrease when the player loses a small game or encounters a monster.Weapons can be collected by winning small games and used to defeat monsters.

2.4 Small Games

We implement each small game as a separate class (e.g., Game1, Game2, etc.) and include them using the header files (game1.h, game2.h, etc.).The main Maze-game class will handle interactions with the small games, such as starting a game when the player enters a room containing one.The player will be rewarded with weapons or other items when they win a small game, and lose health points when they lose a small game.

2.5 Game Flow

The game starts with the player in an initial room and a brief introduction to the story and objective.The player navigates the maze by choosing directions (e.g., north, south, east, west) to move between rooms.When entering a room, the game displays a description of the room, any items or monsters present, and available actions.If the player encounters a small game, they have the option to play it. The game's outcome affects the player's state (e.g., winning gives a weapon, losing reduces health points).The player can use weapons to defeat monsters and progress through the maze.The game ends when the player reaches the room containing the powerful artifact or runs out of health points.

2.6 Background and rule of game in detailed

The game starts in room 1 and ends in room 9. Players cannot see the maze and have to explore it themselves. We recommend that players draw the maze on paper while playing game.

Background: The main character called Siguard, in this maze to find the artifact Gungnir (Odin's weapon) Not only do you have to go through the maze to get to Room 9 but you also have to collect 2 treasure and 3 items to get Gungnir The nine rooms represent the nine worlds of Norse mythology There are 9 rooms in the maze divided by 1 and 9 as the starting room. 2,3,4,5,6,7,8 all have different scenarios

Start:

The player has health point 100

Bag begin with empty

Rooms 2 and 3 have treasure to pick, respectively

Rooms 3,5,6,7,8 have 5 items

There are 4 mini games and 4 items in rooms 3, 5, 7 and 8. The plot design is as follows: in different realms, players need to play games to get items. Once players pass the mini games, players can pass through this room at will. However, if the player fails, health point will be deducted. Players cannot explore new rooms without first playing the game.

For 5, 7, and 8, if the game fails, the health points of -10, -20, and -30 will be deducted respectively

For room 3, a health point of -40 will be deducted if the player chooses quit to skip the game (because this game is a puzzle game, it is only possible for the puzzle to succeed, or the player cannot continue to choose quit)

There is a monster in Room 6 (no mini game) The player needs to attack the monster by using the items collected in Room 3 or Room 7.

Go through room 6 to the final room 9 only collect 2 treasure and 3 items to win Gungnir game and end, otherwise the game fails and end.

1. Implemented features

3.1 Generation of random game sets or events

In Game2.cpp , we use srand() function to generate a random number from 1 to 100.

3.2 Data structures for storing game status

In Game5.cpp , we use new char\* to store board size. In other cpp files we also make use of many kinds of structs , pointers and dynamic variable allocations.

3.3 File input/output

In Game4.cpp , we use concept of file input to read a file ‘words.txt’

3.4 Program codes in multiple files

We split the whole game into different parts, including four game.cpp files, eight rooms.cpp files etc. and their relative head files.We also call all those functions in our main program(text-based game.cpp).

3.5 Proper indentation and naming styles

We have well-defined indentation for all functions and also we name every functions in a readable way.Reader can easily understand the meaning of functions and how it works in the whole program.

3.6 In-code documentation

We do have lots of in-code documentation to help readers can understand some steps in functions especially in game.cpp files and room.cpp files.

1. Run the game

Put all files in a folder. As we have a makefile, so just type ‘make game’ in the terminal.Then ‘./ game’ to run the game.