John Stibbards
Thomas Young
Cassidy Carpenter

TBD, The Game

Student designed, TA approved

You play as an intrepid adventurer who enters a dungeon in search of fame and glory. Traverse the randomly generated labyrinth, but be careful, for enemies lurking around every corner in search of novice adventurers to consume.

In our project we will create an interactive experience by creating a dungeon crawler game. The game will be composed of three main parts: exploration, battles, and puzzle/mini games. The exploration aspect of the game will consist of finding your way out of a dungeon while interacting with various obstacles and items. Possible obstacles we might implement are enemy, puzzles, and traps. Items will aid the player with their quest to escape the dungeon by making them stronger and more resilient to the perils that lay before them. The second part of the game will be triggered when interacting with an enemy. The player will have to participate in a basic turn based battle simulation. Finally, when encountering traps the player will have to participate in a puzzle to escape.

This one should spark joy.

Main Features

1. Exploration
    a. Randomly generated dungeon floor generation
        i. Composed of different room types (decorator)
            1. Enemy
            2. Boss
            3. Traps
            4. Treasure
            5. Normal
            6. *Visited*
        ii. Movement between rooms will consist of selecting adjacent room tiles that are explicitly connected
        iii. Player can only see adjacent rooms
        iv. To progress to the next level, play must find the level's exit room
    b. Interactions
        i. Triggered when entering a room
        ii. Obstacles:

            1. Enemies
                a. Easy/Medium/Hard Enemies chosen (there is a pattern for this™)
        iii. Bonuses:
            1. Items (see below)
            2. Healing
            3. etc.
2. Battles
    a. Turn-based
        i. Enemies/Player
            1. Health Points (HP)
            2. Magic Points (MP)
            3. Skills
        ii. Enemies:
            1. Min-Max decision tree
3. Traps/Mini Games
    a. Simple games based on memory or keyboard dexterity
        i. Potential Examples: Give up mechanic, Button Mashing, Timed button pressing
4. Player progression/Loot
    a. Player has 4 item slots (main hand, off hand, armor, power)
        i. Player skills and stats dictated by items in these slots
    b. Loot found after battles or finding it during exploration

Requirements

Functional

1. The Player will be able to interact with a 2D board that represents a single level of the game
2. When entering a new room the player may be required to either participate in a battle, solve a puzzle, or be offered new items to swap out.
3. For battles, the player will participate in turn based moves until the battle has completed
4. For puzzles, the player may be asked to answer questions or perform tasks with their keyboard.
5. For new items, the player will be offered a new item. They can either keep the one they have equipped or equip the new found item and lose the old one.

Non-Functional

1. Should be able to support 1 user
2. Testable

3. The game should be responsive with no input delay
4. Rooms should be easy to construct from basic underlying features (if they have enemies, treasures, or traps)
5. User's should be held to rules of the game with no possible exploits

Constraints

1. Runs on a personal computer
2. Single-player game (1 User)
3. Need to use free license resources (images, sounds, etc.) or create our own

Data Storage

Since our game is a rogue-lite (you have a single life to beat the game) the amount of data storage we need will be very minimal, with the data we need to keep track of being contained in the game itself.

One piece of information we will need to persist throughout the game will be the player object which includes the player, their items, and their skills. The player object is instantiated when the user clicks play on the main menu. It is then kept track of during the game by having reference to it as a field on the floor. When a room is triggered the player object is passed to the room in an execute function as an argument.

Another object which needs to be kept track of within each level is the floor. When a character enters a room, the floor needs to be saved temporarily while the player interacts with the rooms. There is a graphics object that is in charge of rendering the graphics for exploration. This object stores a reference to the room so when a user goes to a new room, this object will save the reference to the floor. When the player is done interacting with the room the floor graphics object can be used to return to the room. Rooms will be stored in the floor object so we do not have to do anything extra to store them.

Finally, the last object we will need to access information from during the game is some metadata about the game such as current level and how many levels are in the game. This will be done with static methods that other objects will be able to access.

Patterns

1. For this delivery, you should include your concepts on using applicable patterns in your UML Class Diagram for the system above.

2. In your final delivery you will be asked to document and discuss what OO patterns or variations you considered or used, and how they impacted your design.

Decorator

- Customization of game aspects:
  - Item handling
    - Having various types of items (helmet, amor, mainhand, and offhand)
    - Equiping items

Simple Factory

- Creating different versions of game parts:
  - Floors
  - Rooms
  - Enemies
  - Items
  - Puzles

MVC

- Model
  - Controlling the states and actions within the game
    - Main Menu
    - Exploration
    - Battle
    - Puzzles
    - Equipping new items
- View/Controller
  - Displaying the current state
  - Will take input from the user and translate it to the model

Observer

- When model objects change state the view will be notified so it can update

Use Cases

- How many different types of users will your system have? What tasks do they need to accomplish with your system?
    - Our system will only have one type of user which will be the player for the game. As it is a singleplayer game there will also be one such user at any given time interacting with the game.
    - The only task to accomplish in our system will be completion of the game through interaction with its various systems and progression
        - To complete the game the user will explore a dungeon, solving puzzles and fighting enemies.
    - “User wants to play video game” (user goal/any user)
    - Main Scenario
        - 1. User launches video game (begins running program)
        - 2. User can start a new game
            - There is no saving since we aim to make a rogue-lite
        - 3. User interacts with game mechanics to progress through the game
            - For example: Traps/Puzzles, Battles, and Exploration
        - 4. User (eventually) completes the game.
- Try to think of the various problems or variations that can occur while trying to accomplish these tasks and document them in the use case
    - The only task available in the game is mostly defined by the game itself, so it is subject to very little variation on part of the user.
    - The major problem that could arise accomplishing this task is the user being given a poor understanding of how to interact with the game, i.e. what options they have, how to progress, and understanding the mechanics.
    - If the game is not balanced well, the player may struggle with getting through the game due to it being too difficult.
        - This can manifest as either enemies being too strong/smart, puzzles being to difficult, or items not being good enough.