Aalto University

ELEC-A7151 - Object Oriented Programming with C++

Project Plan

Dungeon Crawler

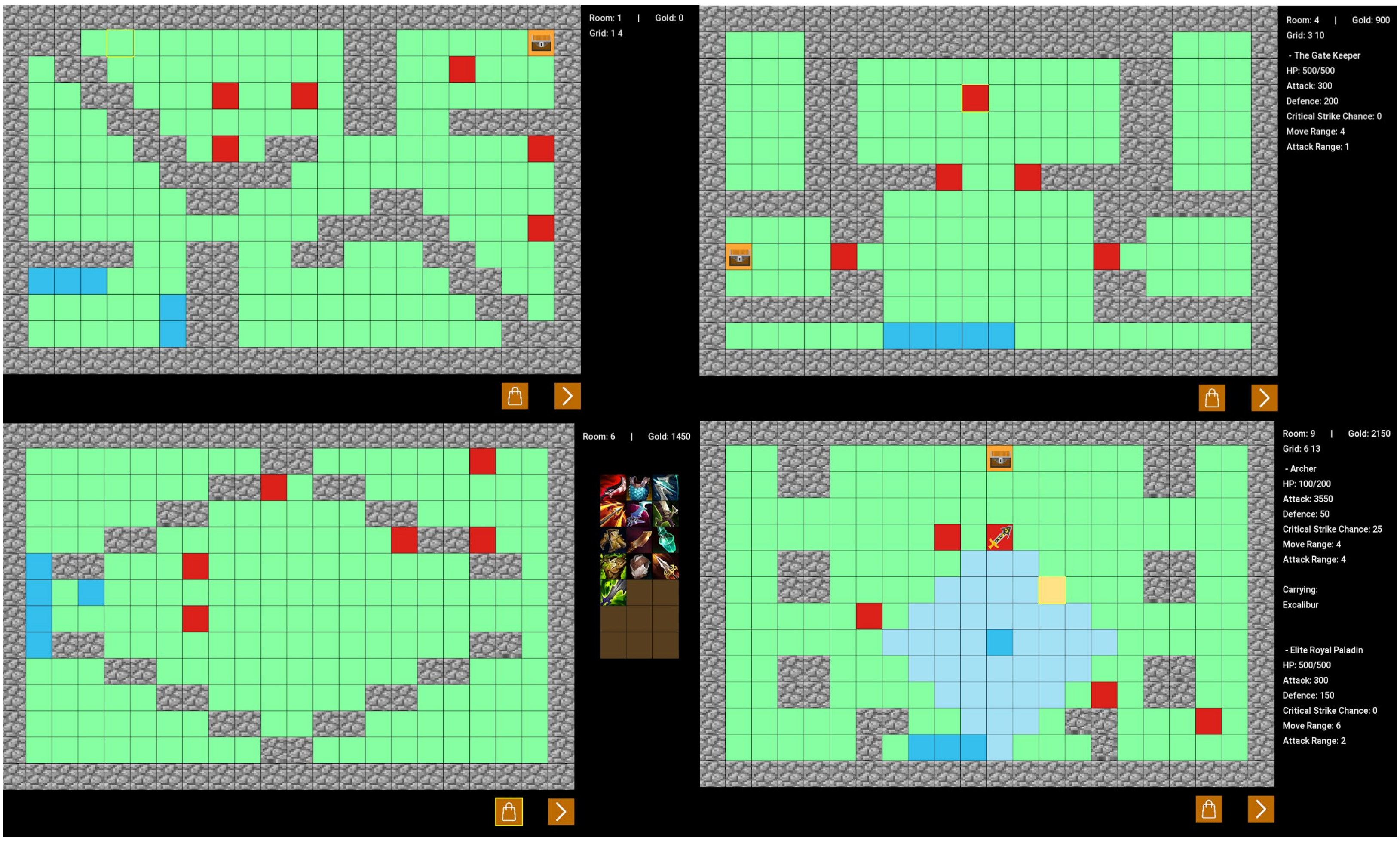
Member: Phi Dang, Mai Vu, Khoa Nguyen, Taige Wang

1. Overview

The Dungeon Crawler project is a Rogue-lite, turn-based gameplay. The Rogue-lite element will be implemented with perma-death features, random dungeons, and a currency/upgrade system that carries on after the player's death, which helps to reduce the difficulty of the game. The turn-based gameplay element takes inspiration from the famous game franchise Fire Emblem, where each room of the dungeon is a tile-based map.

a. Rogue-lite gameplay

The dungeon is essentially a randomized tree of rooms inspired by a rogue-like game Dead Cells. There are two types of rooms, Battlefield (figure 1) and Rest (figure 2). Dungeon rooms are Battlefield rooms with randomized terrains as well as enemies to make the gameplay unique for



Room: 1 | Gold: 0
Grid: 1 4

Room: 4 | Gold: 900
Grid: 3 10
- The Gate Keeper
HP: 500/500
Attack: 300
Defence: 200
Critical Strike Chance: 0
Move Range: 4
Attack Range: 1

Room: 6 | Gold: 1450

Room: 9 | Gold: 2150
Grid: 6 13
- Archer
HP: 100/200
Attack: 3550
Defence: 50
Critical Strike Chance: 25
Move Range: 4
Attack Range: 4

Carrying:
Excalibur

- Elite Royal Paladin
HP: 500/500
Attack: 300
Defence: 150
Critical Strike Chance: 0
Move Range: 6
Attack Range: 2

Figure 1: Layout of the GUI showing the Battlefields of room 1, 4, 6, and 9.

each run. The world map is a Tree filled with those randomized rooms. However, the structure of the dungeon is set to the following stages every run:

Room 1 → Room 2 → Room 3 → Rest 1 → Room 4 (Boss 1)

→ Room 5 → Rest 2 → Room 6 → Room 7 (Boss 2)

→ Room 8 → Rest 4 → Room 9 → Final Boss → Room 1 → …

Rest, however, is always the same. Rests have a Store where the player can spend their gold collected during the run on upgrading or purchasing items and units. It is also a place for healing the units. The Player's units are automatically healed to full health when the Player arrives at Rest.

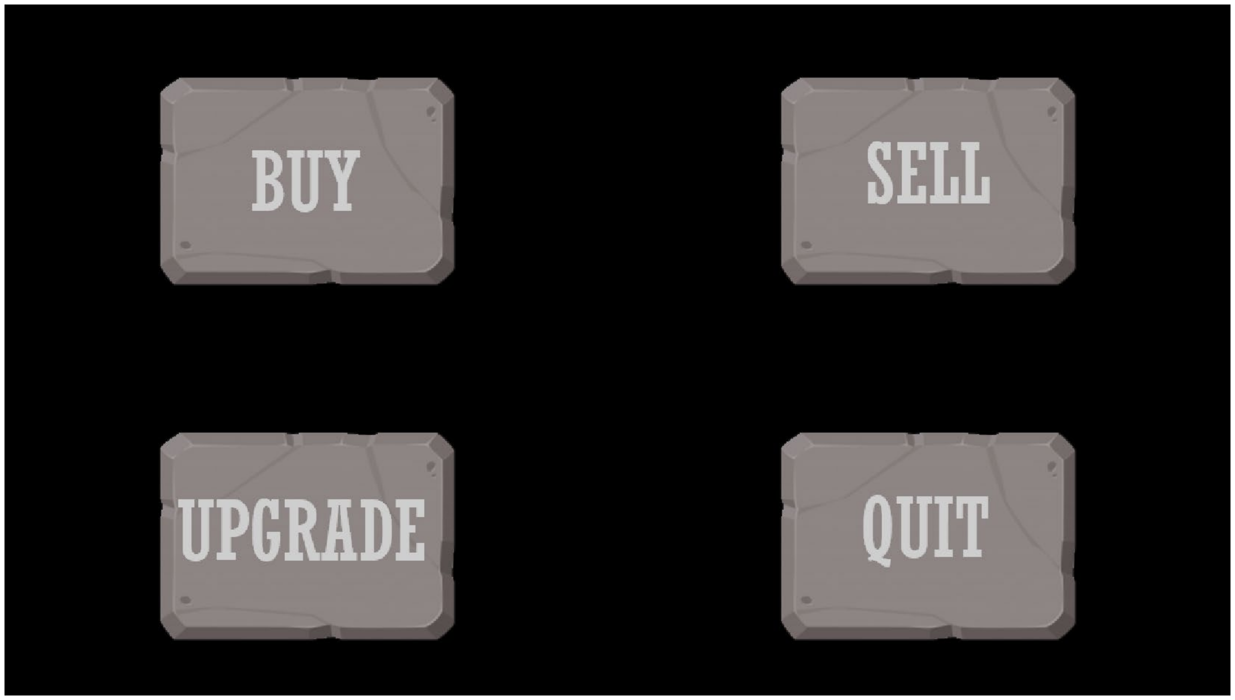


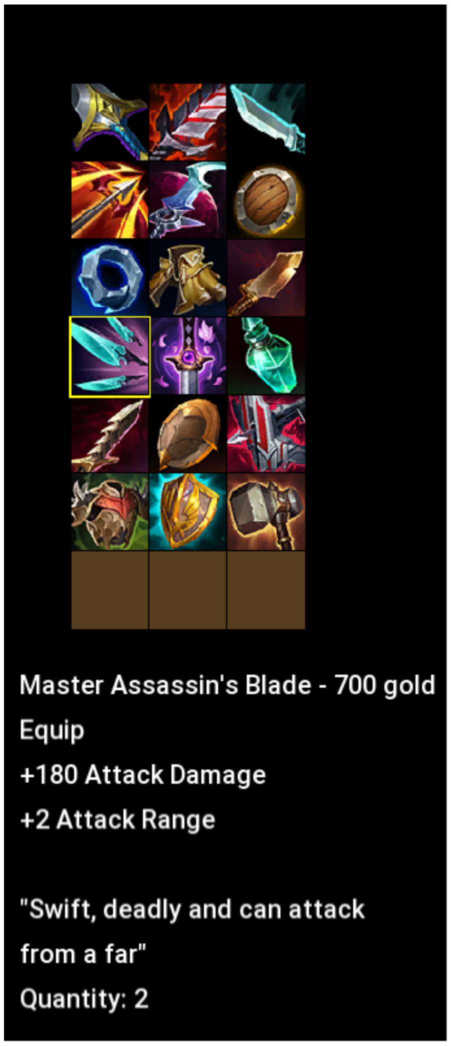
Figure 2: Layout of the GUI showing the Rest features.

The last room of the dungeon is the boss room (figure 3). Beating the boss room counts as a win. The player will be reseted back to Room 1 after beating the final boss. Because of the nature of a rogue-lite game with every item, gold gets reseted. However, the upgrades for the unit should remain after each run.

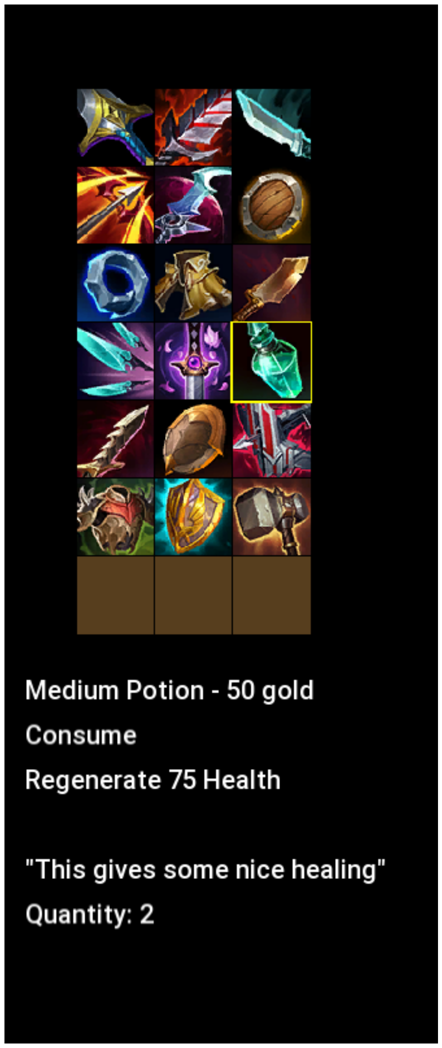
b. Turn-based gameplay

The player has a team of 5 members/units. At the start of the whole game, the player is given 5 basic units. During the runs, the player can spend their gold to upgrade the existing units in the Rest to make the runs easier.

Each playable unit has its own type, such as range, melee, etc., with varying pros and cons. For example, Range units can attack at a great distance but they have less health, damage, and defense compares to other units. This is when equipping items becomes an important part of the gameplay. Equipping items offers a strategic aspect to the game where the player can customize their unit to suit their playstyle.


Master Assassin's Blade - 700 gold
Equip
+180 Attack Damage
+2 Attack Range

"Swift, deadly and can attack from a far"
Quantity: 2


Medium Potion - 50 gold
Consume
Regenerate 75 Health

"This gives some nice healing"
Quantity: 2

Figure 3: An example of an equipable item and consumable item.

The player's units all share the same inventory where the player can apply consumables and equipment items (figure 3). The item equipped to the unit temporarily increases the unit's stats. The item and the player's currency are lost if a run ends. The items purchased at shops. It can also be found as loot from treasures in some rooms.

The Player can swap in and out units and make the team with different types of units which offers a more strategic aspect to the game. Different combinations of units create different play styles, such as a melee comp tends to be more aggressive and approach combat head-on, while range comp tends to kite back. Building a team with each unit supporting each other well is up to the player. Also, this feature can make every player's experience of the game unique. However, we haven't been able to implement a GUI to choose starting units. Therefore, instead of having 5 chosen Units, the Player starts with 5 Units randomly chosen from 10 premade Units that we have created. We believe this feature not only does not affect the strategic side of the game but it also makes the game more unpredictable as well as challenging.

At each turn, each player's unit and enemy's unit get to make a move. A move is complete when a unit moves or uses a consumable item. Equipping an item and openning treasure is not considered a move. After moving, consuming, or equipping an item, a unit gets to attack if it is able to. When all the player's units make their move, the player's turn ends. Then, it is the bot's turn to make a move. The run ends when all player's units are killed, or when the boss is slain, or when the player quits the game.

2. Implement Instruction

After cloning the project, it is possible to build and run the GUI program/Tests by using CMake. CMake is required for generating the make file, and then the make file can be used by make program, which compiles and packs codes to executable. In this way, the program can be executed on all platforms.

In short, the project can be executed by running these commands in the terminal under the project directory

```
cd dungeon-2020-4
mkdir build
cd build
cmake .
make dungeon
./dungeon
```

The project also requires SFML 2.5.1 pre-installed. In addition, the Google Test framework also requires a recent version of Python 3 installed on the machine.

In addition, the project was configured as a Visual Studio Code workspace, so after installing suggested plugins by the workspace, users can build and execute the program on GUI interface.

The project README also provides instructions and descriptions on building, compiling and running the programme / test.

3. Gameplay Instruction

When running the game, the start screen is shown (figure 4). After clicking to start, the story of the game is displayed (figure 5). When the whole text is printed, the game begins; a screen with the first room is shown (figure 6).

The room takes the form of a Grid with Walls are the grey squares, Floors are the green square, and the yellow square with a chest image is the Treasure. Each Floor Square can be occupied by a unit. Ally units are depicted as Blue Squares and Enemy Units are depicted as red. On the right-hand side of the GUI, the current Coordinate where the mouse is hovering is shown as well as the Player's current gold and current level the Player is in. If the mouse is hovering on a Unit, the Unit's description is also shown. By clicking and dragging a unit, the player should be able to command a unit to either move, attack or equip an item.

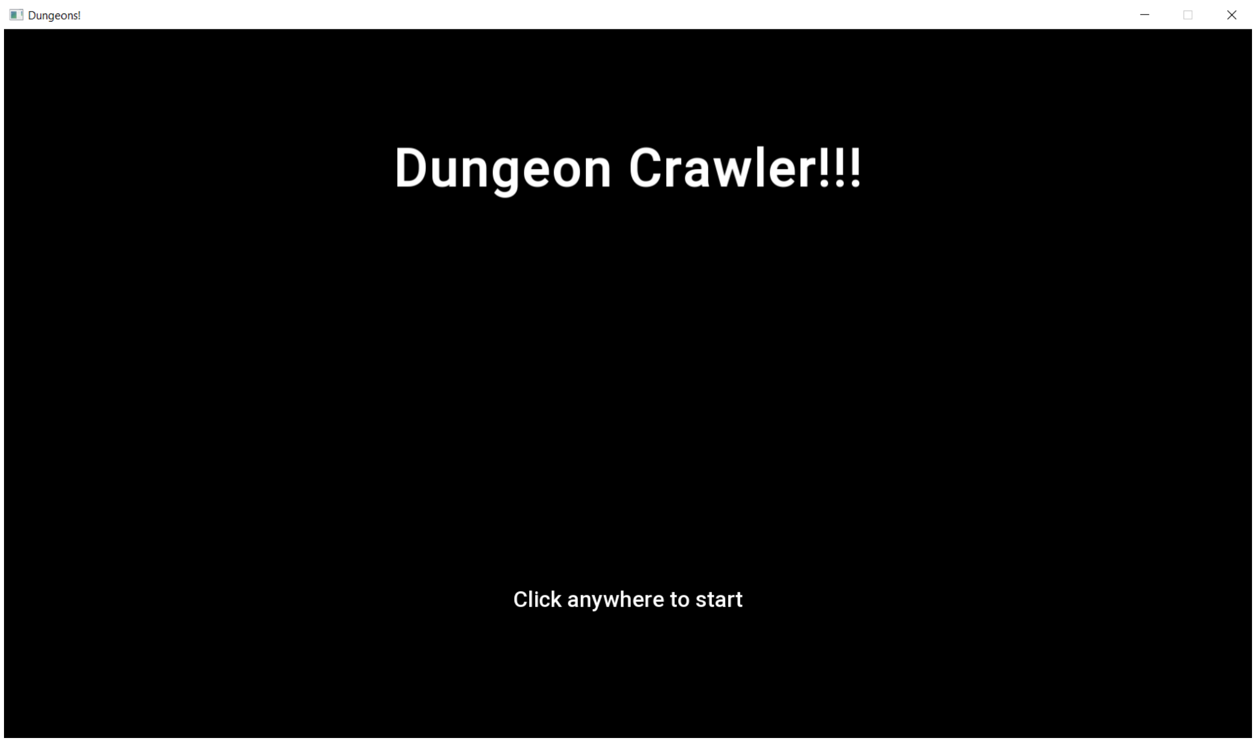
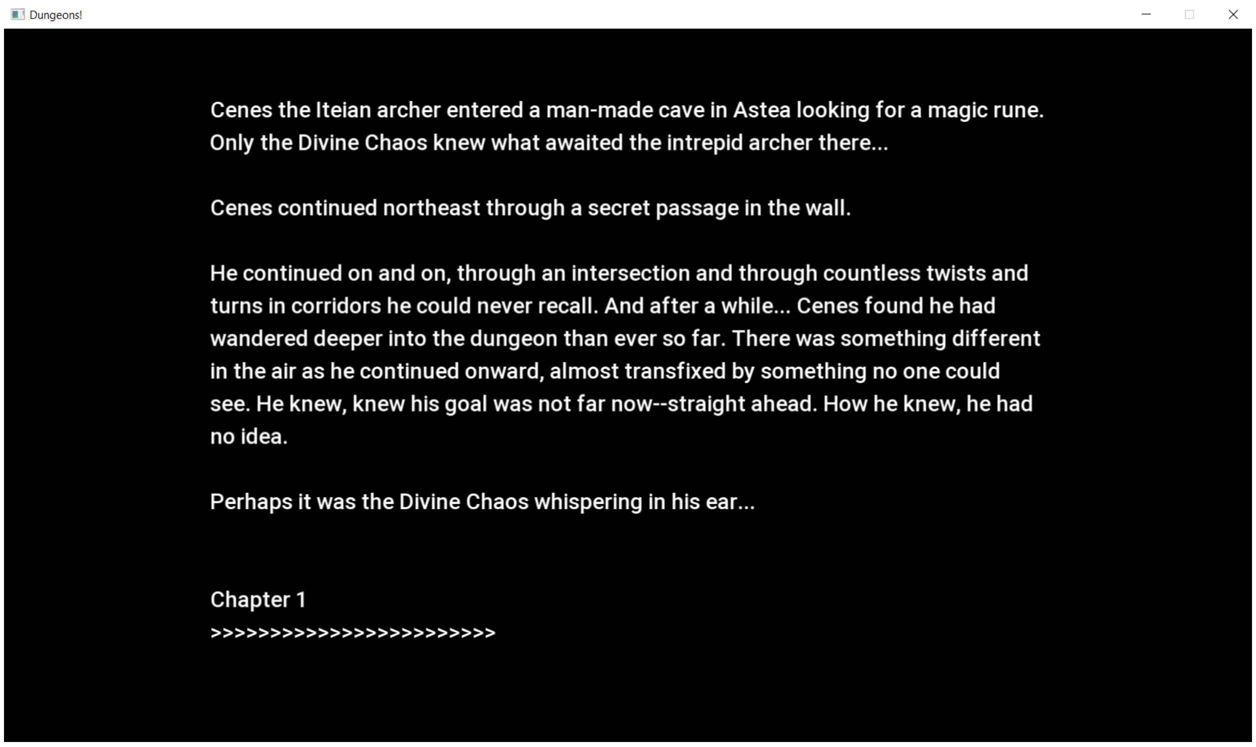


Figure 4: The project start screen.



Cenes the Iteian archer entered a man-made cave in Astea looking for a magic rune. Only the Divine Chaos knew what awaited the intrepid archer there...

Cenes continued northeast through a secret passage in the wall.

He continued on and on, through an intersection and through countless twists and turns in corridors he could never recall. And after a while... Cenes found he had wandered deeper into the dungeon than ever so far. There was something different in the air as he continued onward, almost transfixed by something no one could see. He knew, knew his goal was not far now--straight ahead. How he knew, he had no idea.

Perhaps it was the Divine Chaos whispering in his ear...

Chapter 1
>>>>>>>>>>>>>>>>>>>>>>>>

Figure 5: The project story.

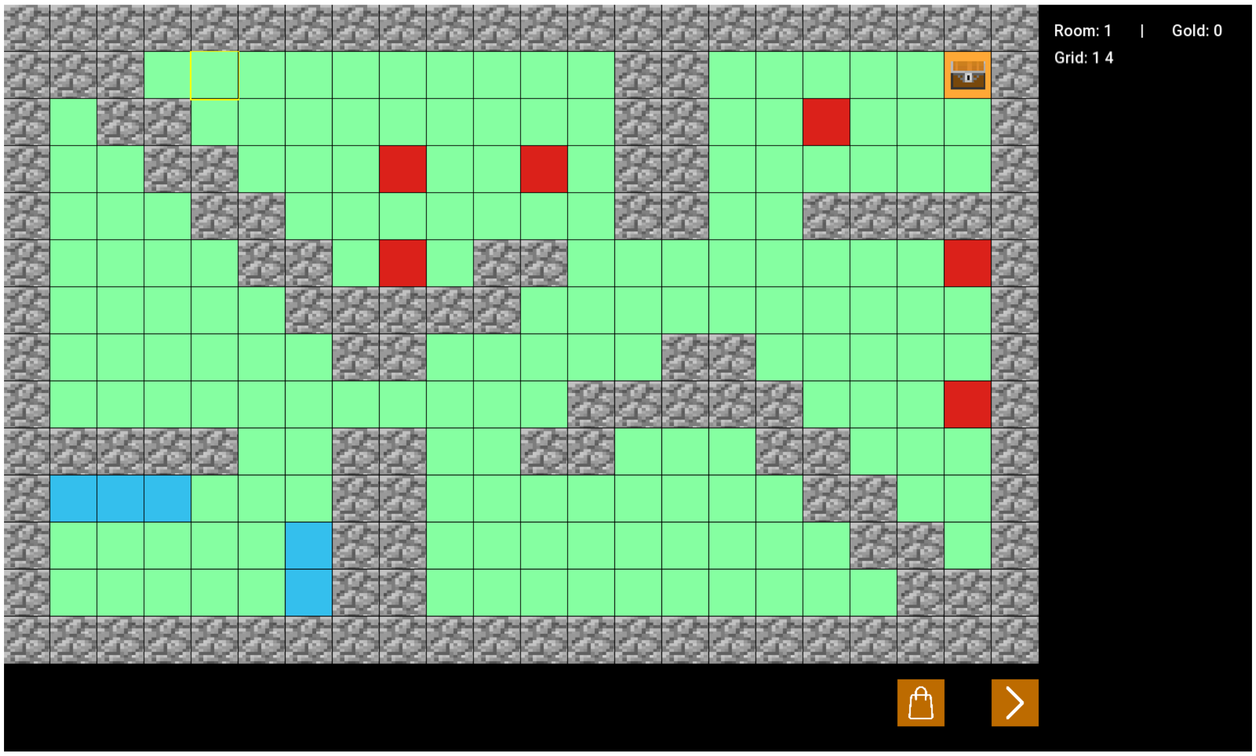


Figure 6: The first room.

a. Move Units

When clicking on a unit, there should be an array of light-blue squares showing up around that unit indicating where the unit can reach. The player can then move the unit to one of those lit squares by clicking on that unit and drag it to the desired destination. However, if the unit is already moved that turn, it instead stays at its location and a text line below the grid will be written indicating that it has made it turn. The unit will also stay at its original location if the destination Square does not belong in the array of lit Square and a text returns "Cannot move unit to Coord(x, y)" to indicate that the unit cannot reach that destination and therefore cannot move. If the move was successful, a text "(Unit name) moved successfully from ("Original location) to (Destination location)".

b. Access the Inventory and Equip items

Below the grid, there are two buttons, Inventory and End turn. The Player can check their inventory by clicking on the "Inventory" button with a "bag" icon. When the Inventory button is clicked, the area where information of units is usually shown changes to a 3x7 grid that shows the first 21 items in the inventory. Hovering above an item will result in a text showing that item's description below the inventory grid.

The player can equip an item to a desired unit by dragging the item on to the square that the desired unit is occupying. However, equipping an item that the desired unit already has or if the desired unit, there will be text that shows either "Inventory of (Unit name) is already full or already has this item". Moreover, equipping the item to a square that does not contain a unit results in a text "Item can only be equipped to an ally".

The item equipped to a Unit is considered permanent during that run. With each unit having only 4 available slots for equipping items, choosing which item to equip can make the game much more challenging.

There is another type of item that has type Consumable. These items are consumed with similar click and drag action to equipping items. When "equip"/consume these items, the Unit's inventory will not be changed.

c. Attack

When clicking on an ally, if there is an enemy that is in its attack range, the enemy will show a "Sword" icon indicating that the enemy is within reach and therefore can be attacked. The player chooses which unit to attack by dragging the selected ally unit on to the desired enemy unit. If the selected ally unit has already attacked that turn, the enemy unit will not lose its health and a text "Unit has already attacked this turn" will be shown. If the selected ally unit has not attacked, the enemy unit loses its health and a text "(Ally name) just attacked (Enemy name)" is shown to indicate that the attack was successful.

d. End turn

The "End turn" button is located next to the Inventory button. When clicking the End turn button, the bot makes its turn resulting in ally units being attacked as well as the bot's units moving around the room if the Room is not cleared. If the room is cleared, the player proceeds to the next room.

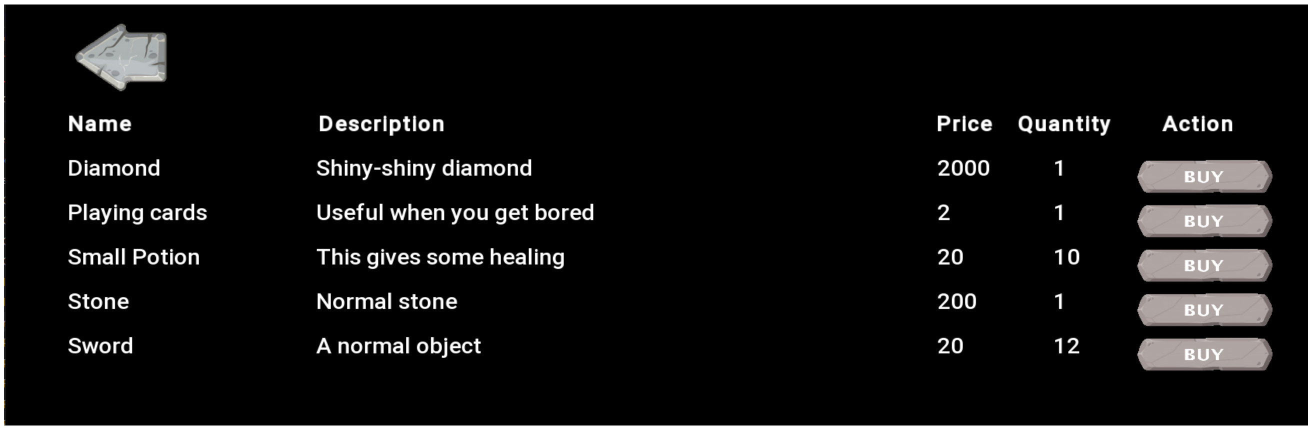
e. Winning/ Losing

If every unit of the player gets killed, the run is counted as a loss and the game will bring the Player back to Room 1 with a fully healed army.

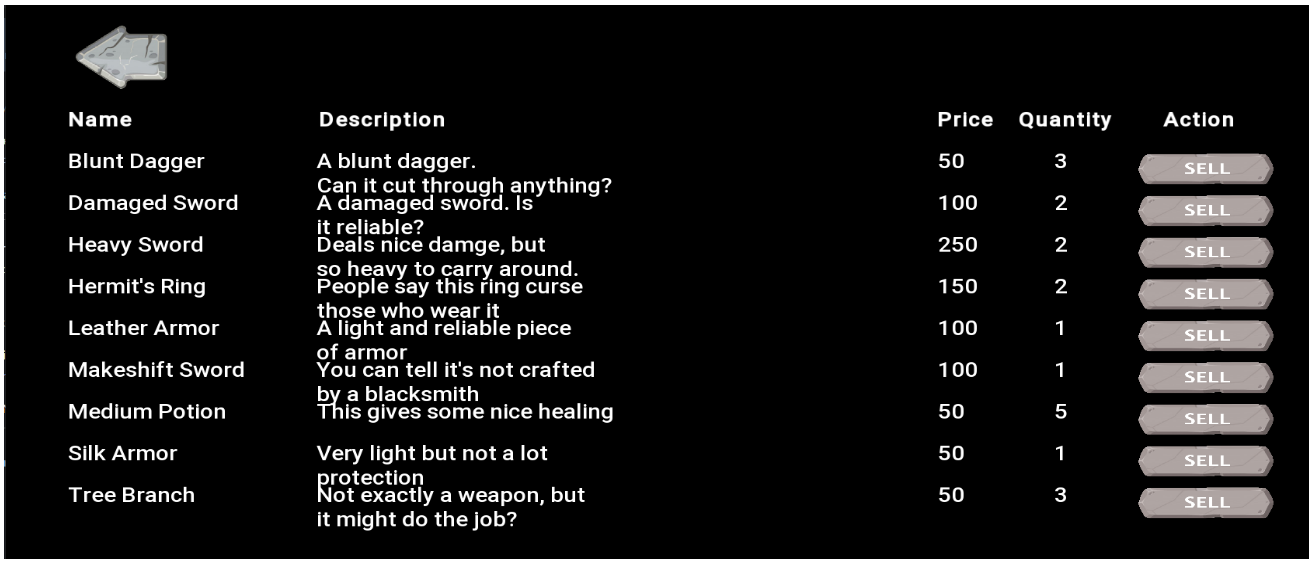
If the Player beats the final boss, the run is counted as a win and the game will also bring the Player back to Room 1 with a fully healed army.

f. Buying and Selling

At Rest, the Player has 4 options shown as 4 buttons on the screen (figure 2). Buy, sell and upgrade buttons behave the same as they take the Player to a buy and sell menu (figure 7). The menu includes items 'name/ upgrades' name, description, cost and quantity of items available in stock for purchasing as well as in the Player's inventory for selling. However, currently, the GUI can only display 12 items at a time. Therefore, if the player wants to get to the "out of range" item, the solution is to buy the top items so that they make way for the bottom ones.



Name	Description	Price	Quantity	Action
Diamond	Shiny-shiny diamond	2000	1	BUY
Playing cards	Useful when you get bored	2	1	BUY
Small Potion	This gives some healing	20	10	BUY
Stone	Normal stone	200	1	BUY
Sword	A normal object	20	12	BUY

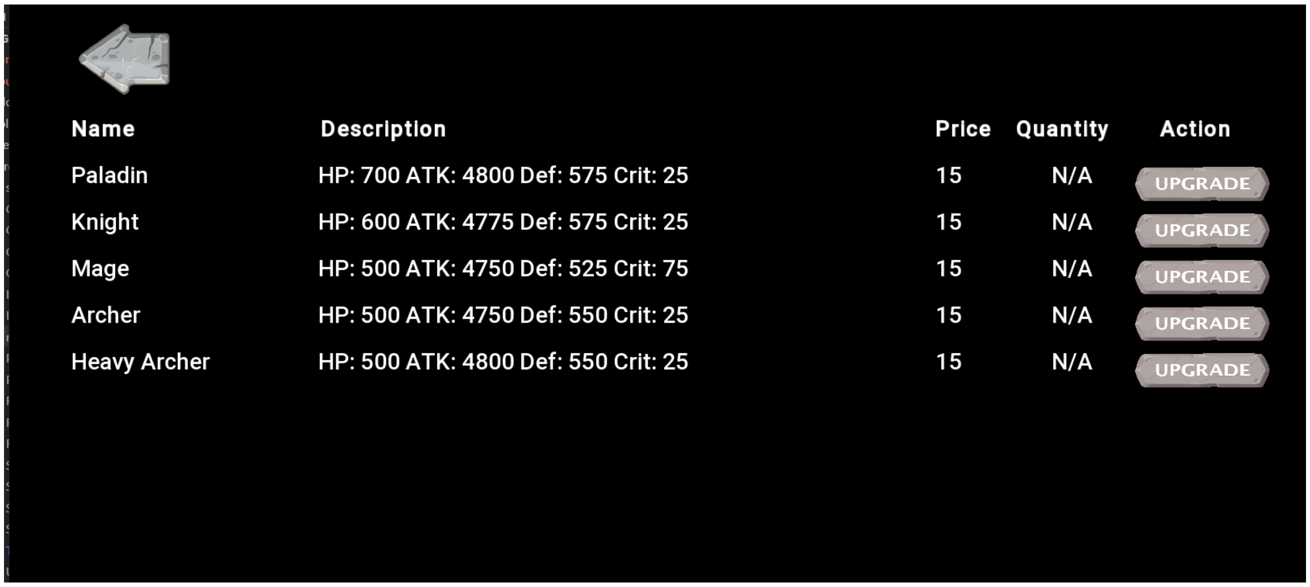


Name	Description	Price	Quantity	Action
Blunt Dagger	A blunt dagger. Can it cut through anything?	50	3	SELL
Damaged Sword	A damaged sword. Is it reliable?	100	2	SELL
Heavy Sword	Deals nice damge, but so heavy to carry around.	250	2	SELL
Hermit's Ring	People say this ring curse those who wear it	150	2	SELL
Leather Armor	A light and reliable piece of armor	100	1	SELL
Makeshift Sword	You can tell it's not crafted by a blacksmith	100	1	SELL
Medium Potion	This gives some nice healing	50	5	SELL
Silk Armor	Very light but not a lot protection	50	1	SELL
Tree Branch	Not exactly a weapon, but it might do the job?	50	3	SELL

Figure 7. Buy and Sell menu

g. Upgrading

As mentioned, the upgrade menu looks and behaves similarly to the buy and sell menu (figure 8). Each time the Player upgrades a Unit, every Stat of the Unit increases by 150% for the first purchase and by 120% afterwards.



Name	Description	Price	Quantity	Action
Paladin	HP: 700 ATK: 4800 Def: 575 Crit: 25	15	N/A	UPGRADE
Knight	HP: 600 ATK: 4775 Def: 575 Crit: 25	15	N/A	UPGRADE
Mage	HP: 500 ATK: 4750 Def: 525 Crit: 75	15	N/A	UPGRADE
Archer	HP: 500 ATK: 4750 Def: 550 Crit: 25	15	N/A	UPGRADE
Heavy Archer	HP: 500 ATK: 4800 Def: 550 Crit: 25	15	N/A	UPGRADE

Figure 8. Upgrade menu

The figure above would differ from the actual game menu where the price would be much higher after each time the Player buys an upgrade for their Units.

h. Known bugs:

- Sometimes, an attack from one ally can kill 2 enemies at once.
- Sometimes, “ghost” can appear in a room. Mainly, a dead ally unit spawn and cannot be controlled by the player. However, when player try to control an alive unit, the “ghost” moves instead

According to our observation so far, these bugs rarely occur and they occur in some very rare special occasions.

4. Implementation

a. UML design

- **World/Game:** This is basically where the whole progress of a game occurs. This class contains all information about ally and enemy units, items as well as responsible for creating the game tree where the Player plays. In this class, information of every room designs, every item in the game, enemies as well as playable units for the Player. The items are also divided based on their level (how good it is) so that after each boss fight, the player can start receiving better items throughout the dungeon and also keeping the game “balanced”.

  Inside this class, there is a function named CreateWorld() that creates the dungeon for each run using the following procedure: It loops through 10 iterations representing 10 levels of the dungeon. For each level, except for boss fights where the room design is fixed, it randomly chooses 1 out of 3 designs for that level and creates a room with Wall, Floor and Treasure placement based on the design. If the room has a Treasure chest, it will consider which level the items should be (how good the item should be) and randomly pick 5 items as Treasure. Finally, it will randomly pick 6 out of 10 enemy units designed for that level and randomly spawn them inside the room.

- **Room:** This class is the Component for the class Game. A room keeps track of its treasure, allies and enemies in the room, who are alive as well as offer functions auxiliary for units to move, open treasure and attack each other. Inside the Room, there are functions that add units (enemies and allies) to the Room and spawn them. Allies' spawn coordinates are fixed while Enemies' spawn coordinates are optional. If the number of spawn coordinates for enemies declared for that room is less than the actual number of enemies, the enemies without spawn coordinates will be spawned randomly. This helps keeping the boss' spawn coordinate fixed while normal enemies are spawned randomly.

- **Rest:** This class plays the role of the shop as well as Rest for the Player. Rest inherits from Room

- **Battlefield:** This class is where the action happens. The Battlefield keeps track of every unit currently on it, its treasure and also feeds information to both the GUI and Player to execute features such as moving units, attacking, opening treasures and so on. Battlefield inherits from both class Room and Grid

- **Grid:** Grid is the foundation class to create Battlefield. Grid is basically a 2D vector that keeps track of the Squares that it contains.

- **Square:** Square can be considered as the unit to make up the game. Square has 3 types: Wall, Floor and Treasure. Wall is responsible for forming the room's terrain; Square is responsible for making the playable area of the room as well as keeps track of the units that occupy it. Finally, Treasure plays the Role of Wall which is an obstacle that can store items where the Player can get the items out in the form of Treasure.

- **Coord:** Coord is a system that helps keeping track of the contents of the Grid/ Room. By calling Coord, the Player can get the Square that contains the desired units as well as Treasures.

- **Player:** There are 2 players in the game, the controller/player and the bot. Each Player has their own army as well as inventory. The Player is the one calling out the actions such as Move, Attack, Equip and so on working with Room and feed the outcome to GUI.
  - Move: We make sure if a player clicks on a unit and drops it at the same place, it will not count as a move. Also, a unit only can move once in a turn and move within its move range.
  - Attack: This function has the same mechanism as Move. When player tells their unit to attack an enemy unit, the enemy unit also returns fire if the attacker is in range as well as if it survived the initial attack.
  - Equip: We make sure that each unit does not have the same weapon or be equipped when its inventory is full

- Bot is also considered a Player. Instead of taking in command, the bot controls its Unit, moving and attacking, with random decisions. In the future, this randomness can be replaced with a decision tree that makes the bot command more logical and realistic.

- **Unit:** The Unit plays the role of armies for both Players. The gameplay is based around the Units. Each unit has its own Stat and will die when its HP = 0. The Units attack each other, equip items and also can be upgraded. Most methods in Unit are called by the player.

- **Item:** An Item might be collected through fighting, or available in a Room, or bought at a shop. Each Item has a unique ability such as raising certain stats temporarily, regenerating health. The Item also has its own Stat which represents the amount of Stat that will be added (or taken away) from the wearer.

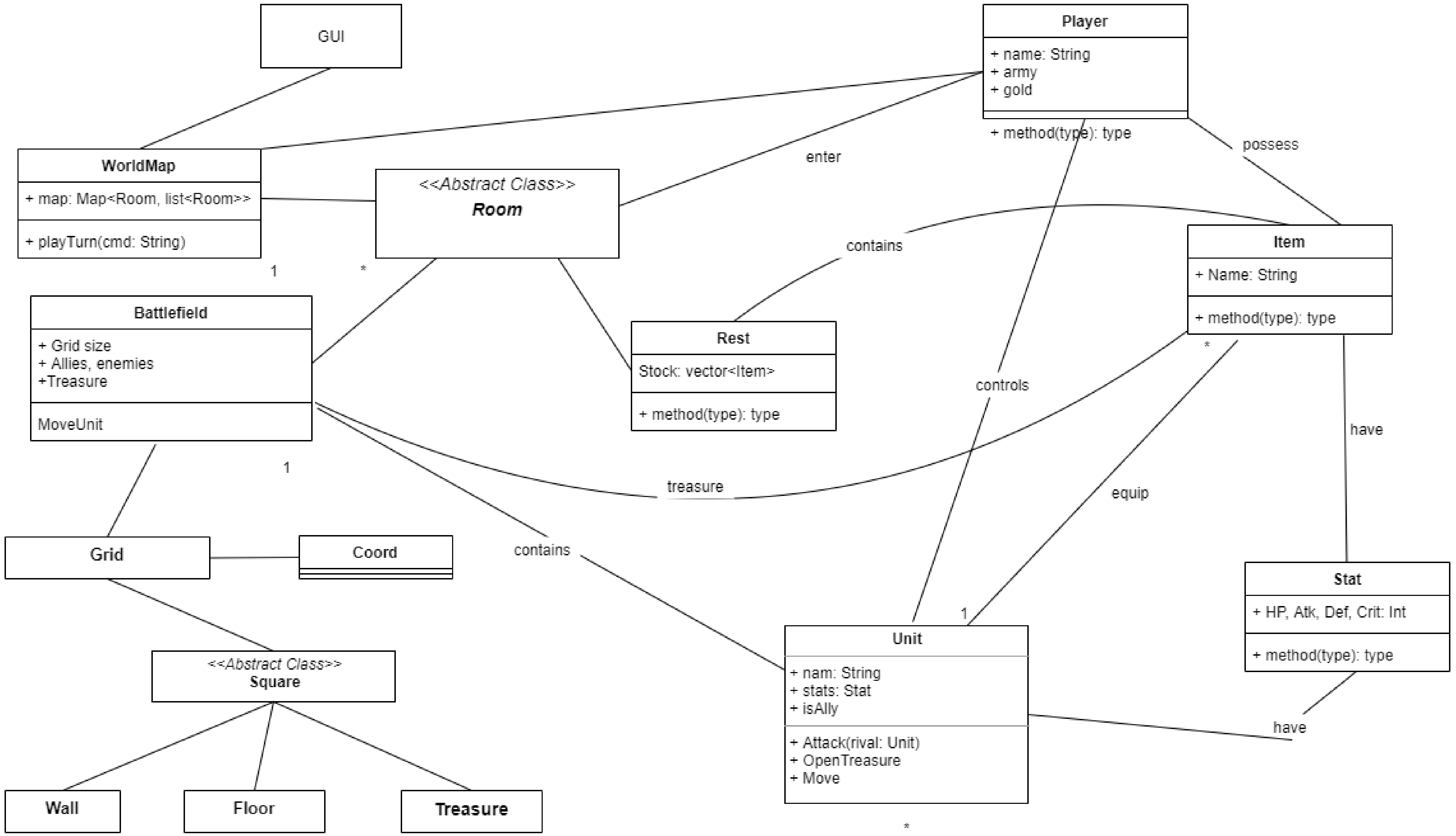
- **Stat:** Stat can be considered as a vector of numbers that decide the stats of the items and the units. The structure of the Stat is as follow: (MaxHP, HP, Atk, Def, Crit, MoveRange, Attack Range)

- **GUI:** An object produces a graphical user interface.

The figure on the next page shows more details.

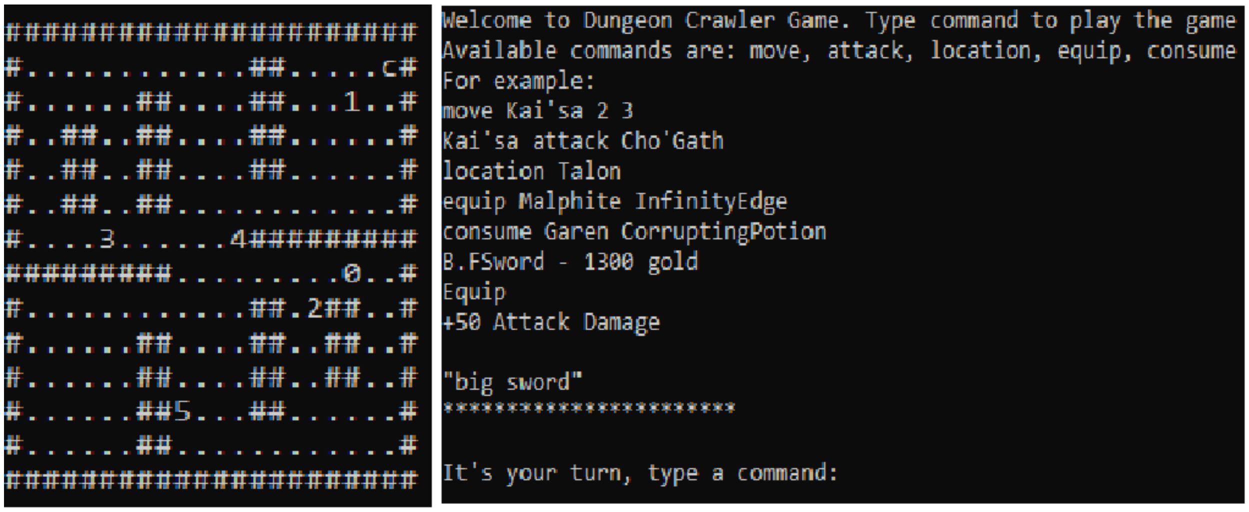
b. External libraries

The project uses Simple and Fast Multimedia Library (SFML) as a 3rd party library for building the game. SFML is free and widely used, especially in developing 2D game worlds. It also supports different types of media, such as sound, font, audio, etc. In addition, Google Test was also used as the unit test framework.


GUI
Player
+ name: String
+ army
+ gold
+ method(type): type
WorldMap
+ map: Map<Room, list<Room>>
+ playTurn(cmd: String)
<<Abstract Class>>
Room
enter
possess
contains
Item
+ Name: String
+ method(type): type
1
*
Battlefield
+ Grid size
+ Allies, enemies
+Treasure
MoveUnit
Rest
Stock: vector<Item>
+ method(type): type
controls
*
have
equip
1
treasure
Grid
Coord
contains
Stat
+ HP, Atk, Def, Crit: Int
+ method(type): type
1
Unit
+ nam: String
+ stats: Stat
+ isAlly
+ Attack(rival: Unit)
+ OpenTreasure
+ Move
have
<<Abstract Class>>
Square
Wall
Floor
Treasure
*

5. Testing

For testing, we implemented a TextUI. The TextUI can take in commands typed by the Player and perform actions such as moving, attacking, equipping and so on. The TextUI also prints the current room as a String with the format similar to the figure 9 below. With this TextUI, most of our features were tested before integrating the classes with the final GUI. Moreover, this TextUI also helps us see the changes that are required when important functions need to be changed in order to better adapt to the GUI as well as make it easier for the GUI to also adapt to the functions.



```
######################
#............##.....c#
#......##....##...1..#
#..##..##....##......#
#..##..##....##......#
#..##..##............#
#....3......4#########
#########.........0..#
#............##.2##..#
#......##....##..##..#
#......##....##..##..#
#......##5...##......#
#......##............#
######################
```

```
Welcome to Dungeon Crawler Game. Type command to play the game
Available commands are: move, attack, location, equip, consume
For example:
move Kai'sa 2 3
Kai'sa attack Cho'Gath
location Talon
equip Malphite InfinityEdge
consume Garen CorruptingPotion
B.FSword - 1300 gold
Equip
+50 Attack Damage

"big sword"
***********************

It's your turn, type a command:
```

Figure 9. TextUI output

6. Work Log

a. General Role

- Project Manager: In charge of team's Github, checking team's progress and code - Phi
- Members: Code – Mai, Khoa, Taige

b. Implementation Phase Role

- Class World, Room, Grid, Square and TextUI - Khoa
- Class Player, Unit, Item, Stat, TextUI and Bot moving algorithm - Phi
- GUI – Mai & Taige

c. Workload spreadsheet

Week	Name	Task	Working hour(s)
44	Phi	Diccus and create project plan	7
	Mai		7
	Khoa		8
	Taige		7
45	Phi	Break week, first meeting with TA	1.5
	Mai		1.5
	Khoa		1.5
	Taige		1.5
46	Phi	Created Player, Item, Unit, Stat class	8
	Mai	Tried to do Cmake, SFML	8
	Khoa	Created Square.hpp, Grid.hpp, Room.hpp, World.hpp	8
	Taige	Added .gitignore, added Makefile config	8
47	Phi	Add Enemy, Ally classes Update Item, Stat classes Implement methods of class Player, allow a player to interact with items and units.	8
	Mai	Created the project with SFML Create grid map with movable square	10
	Khoa	Update Square, Rest classes Create abstract class Room, class Rest inherited from Room Implement methods for saving and create rooms from strings/files	8

	Taige	Updated CMakeLists file, VSCode configuration file, made SFML library to work Tried CTest testing framework	8
48	Phi	Fix errors created during integration between World and Player	16
	Mai	Tried to create Class and Game State Update the Grid with color indicates Enemy, Ally, Obstacle,.. Movable Ally, remember to map Ally cannot move to Obstacles and other Allies	16
	Khoa	Create TextUI Fix errors created during the last 2 weeks as well as errors arise during the integration between World and Player	16
	Taige	Tried with CTest, continued with GUI but no progress	3
49	Phi	Player can do most of basic commands such as: Move, Attack, End Turn, Enter and Exit room Bot inherit Player, can Move and Attack randomly Units interact (fighting) correctly: Damage causing based on Stats, defender lose health... Each unit has a certain range to move in the room. All testings are done through a Text UI	24
	Mai	Implemented classes to graphic	30
	Khoa	Implement functions to create random world tree. Add items, units and designs to the Room. Debug functions and integrate classes to the GUI	20
	Taige	Continued with GUI, tried GameState but eventually moved to simple if-else	10
50	Phi	Integrate classes to the GUI Finalize important features such as Attack, Move, Equip Modify functions to suit classes World	24

Mai	Implemented classes to graphic	50
Khoa	Finalize the classes Integrate the classes to the GUI Build the turn based system element of the GUI Finalize the GUI so that it now create the whole dungeon, change through rooms and recreate the dungeon when the Player wins or loses the game	24
Taige	Implemented GUI Added unittest framework	50