DUNGEON

B璇

109550041
資工13 常乃璇

2021.4.30

INTRO

You woke up, it's pitch black around you...
"You're in the dungeon..."

You, a human or goblin or elf, are trapped in the dungeon.

Here, you'll meet Gandalf, elf queen, and a merchant

Watch out for the fellow lost goblin, the mad time traveler, and Lil Nas X.....

CATALOG

ACTIONS MENU

- In "dungeon" "chooseAction" function, player will be able to choose to "move", "check status", or "save to file"

- "Move" will call "handleMovement" function
- "Check status" will call "player" "triggerEvent" (to show status)
- "Save to file" will create a Record class, and use its "saveToFile" function. Then, set player's current health = 0, and end = 1 (more on "end" later), because my idea was that, when the player save to file it means they're leaving the game.

- For each round the process is:

1. Check Game Logic
2. Show current room index
3. Show player's status
4. Called triggerEvent of the object in the room, if there's item, npc, or monster in there
5. (Finish trigger event) Call action menu
6. Repeat from beginning

MOVEMENT

- In "dungeon" "handleMovement" function, player will see all the all the direction they can go, and will be able to choose from them and move to the room.

1. It'll first detect which direction of the current room is a room and not a NULL pointer (by player-> currentRoom() -> get*blank*Room() != NULL, to know whether there's a room in the direction), then display it to the player.
2. The player can then choose with "u", "d", "l", "r" input to move, each will call player's "changeRoom" function which will change player's "previousRoom" to their current room, and their "currentRoom" to the room of their choice (illegal input (including wrong direction) would out put "wrong input" and end this round).

SHOWING STATUS

- In "player" "triggerEvent", it'll output current health, max health, attack and defense of the player, using getter to those variables.
- It'll also output the item name in the player's inventory, with a simple for loop detecting the inventory size and out put all items' name one by one till all item are showed.

- This function is called by two places, one in "chooseAction", the other is the beginning of each round (because I find it a lot easier to play if I can see the status at all time).

PICK UP ITEMS

- In "item" "triggerEvent", there are two ways my items work (because there's three type of items in my game, more in bonus), one is to be able to pick up and directly add attack, current health, defense points to player, one is to put it in player's inventory vector.

- For the first one, it'll first tell player what the item is by getter of item name, then ask if the player wants to pick it up.
- When choose "yes!", it'll tell you that you've picked up *blank* (use getter to get its name) and health, attack, defense raise by *blank* (use getter to get those info from item) points. Then added those points to your status (with your current health, attack ,defense setter and the item's variables' getter). And it'll pop out the item from the room using pop object function in room class. (By the way, I designed it so that the player's current health can be higher than max health, because it works like that in Mario Galaxy 2.)
- When choose "nah", it'll let you carry on, and won't pop out the item (You can come back and pick it up later).

- For the second way, it'll use push back to put itself into the player's inventory vector (the function is very minimal because it's only use for merchant exchanging with player).

FIGHTING SYSTEM

- In "monster" "triggerEvent", it'll first show both sides status and check if they both are still alive, then player can choose to "retreat", "attack", or "use item" (I'll talk about the first two in this section) for every round.

- If player choose retreat, it'll use "changeRoom" function to move player back to the previous room.

- If player choose fight,
- it will let you attack the monster (with monster's current health setter, monster's current health = (monster current health - (your attack - monster defense))), it'll then show the damage on the monster.
- Check if it's still alive, if monster's current health > 0, it'll then attack you (with your current health setter, your current health = (your current health - (monster's attack - your defense))), it'll then show the damage on you.
- if you're still alive, then show your status.

- repeat the whole process from "showing bother side's status and choose to retreat, fight or use item", till either one die or love potion used (more on that later) or retreated.
- If you defeat the monster, pop object function will pop it out of the room's object vector.

NPC

- In "npc" "triggerEvent", I have three kinds of npc, I'll only talk about the "merchant" here (the other two in bonus section)

- Merchant has two potions in his commodity, lovepotion and hatepotion.
- He'll first listed out all the items he owns with item's name getter, then asked if the player want to exchange health points for potion.
- If player choose one of the potion, it'll call the item's trigger event (through the second way I described how my item trigger event implement) and push back itself into the player's inventory vector. And with player's current health setter, setting current health to (current health - 20).

- Then I let the merchant disappears (pop object function to pop merchant out of the room's object vector), so the player can only meet the merchant once.

GAME LOGIC

- In "dungeon" "checkGameLogic", it'll check if either player's current health == 0 or boss's current health == 0 or player's "end" == 1, then it'll return 1, else return 0. With the returning of "1", the game will stop going into the while loop, and determine whether the player win, lose or save file.

- Player's "end" is used to identify alternative ways to end the game including "chatting with elf queen continuously", "using love potion on boss" (more in bonus section), "save to file"

- Once check game logic = 1, and stop going into while loop in run dungeon. How did I detect whether player win/ lose/ save file?
- With "player's current health <= 0" and "end == 0" --> player lose
- With "player's current health <= 0" and "end == 1" --> player save to file and leave
- The rest of the case --> player win!

RECORD SYSTEM

- In save player,
- I saved the player's name, current health, max health, attack ,defense, race (more on this in bonus section)
- For inventory, I'll first save the inventory size so it's easier to load inventory back using for loop. Then the item's name, health, attack, defense, and type (more on "type" in bonus).

- In save room,
- I save whether the room's objects vector is empty or not, with 0, 1.
- And I save the two normal monsters' current health, and boss's current health, attack, defense, and type (more on "type" and why I have to save boss's all status in bonus).

- When I load it back, since I will first use createMap to create the full map, I only decide to pop out object in the rooms' object vector according to the "empty or not" "0/1" data I saved. And I also create a player with minimal setup, then update my player's status with setter. Finally, I update current health and other info to my normal and boss monsters.

ENHENCEMENT

- Throughout the game,
- added in "clock" (sleep(*1*)) in various places, for the dialogue to appear one-by-one.

- In player, monster, npc, item class,
- Pretty much all the stuff I added are an "extra variable" in class (whether it is "kind" or "type"), and will make trigger event do different things according to those type/ kind. (I added if/else in trigger event to do different thing according to it's extra variable).

- See the details in the pages below...

PLAYER BONUS

- I added three "type": "human", "goblin", "elf"
  --> which will affect the preset of max health, attack, and defense, and what direction Gandalf suggests to you and how the dungeon talk to you.
- I added bool "end"
  --> because I added more ways to end the game (more on that later)

NPC BONUS

- I added three "kind": "Gandalf", "elf queen", "merchant" (they will do different things in trigger event)

- Gandalf will give player suggestion for direction based on their type (human, goblin, elf).
- Elf queen will ask player to chat with her, if player continuously chat with her, will be able to win immediately (because you're a good person! It's totally not because elf queen wants you to go to her ex girlfriend immediately!), by setting player's "end" to 1 (alternative way to end the game).
- Merchant is already talked about in basic function.

MONSTER BONUS

- I added two"type": "normal" monster (traveler and goblin) and "boss" (Lil nas x as satan)
- For boss, if his current health is less or equal to half of his max health --> he will goes to insane mode, attack and defense raise by 20 points (which will make the game a lot tricker). I'll also set boss's "type" to normal monster, because how I determine whether monster goes to insane mode is by their current health and their type.

ITEM BONUS

- I added three "type": "appear alone" in room item (sword and star), "love potion" and "hate potion" (you can get potion from merchant)
- When call trigger event, appear alone item will ask if player want to pick it up. Then add status to the player directly. While for potions' trigger events, it will push back the potion into player's inventory vector.
- I added triggerEvent2 for using potions
  --> let player use item when combating monster
  --> love potion will make the monster disappears immediately, love saves the world!! (alternative way to end the game when use on boss)
  --> hate potion will give monster -20 health

DISCUSSION

- Most problems I had before I started, was solved by doing and understanding the Lab exercises (Thanks a lot to TAs for designing Lab this way!).
- One HUGE problem for me was including header files. In the beginning I was terrifies of "vector". Therefore I thought about putting everything in the room as a new variable. But then I ran into the problem that some of my header files are including each other, and soon I found out that's a big no. Because then the complier won't know which to follow. Thankfully it was a quick fix.
- I also ran some issue with first adding my "end" variable as public. I ended up putting it into private and created getter and setter for it.
- When doing the record system, my roommate gave me the suggestion to create map first before you load record back, then you only have to save very little info instead of the full map. I find that extremely clever!
- Another thing I found very important is that if you're thinking about designing the game more complicated (I would consider mine a bit complicated), you really have to plan everything out in the beginning. Otherwise, it's not always easy to add new stuff in later.

CONCLUSION

I found this project incredibly fun! Once I understand how class works, and once I linked and connected everything together, it is super simple and straight forward in my opinion. The designing part was a lot of brainstorming and imagination, I found graphing out the map first very helpful. Even till the very end, I still go back to the first map I drew to double check my game. I originally came up with this complicated idea that I honestly did not expect myself to be able to actually make it. Because it seems so complicated. But it was a lot easier and faster than I expected.

The beginning of building was really a scary and clueless, because I have so many different trigger events. I was pretty much coding in the dark and not knowing if it'd even work. Thankfully, once I connected everything, it almost worked like magic, and there weren't a lot of issue with the early stuff I wrote!

I learned so much! And it also surprisingly made the midterm writing exam easier for me as well!

My roommates were incredible, they finished the project SUPER early and therefore were always there to help me out! They told me that my dungeon was the most interesting and that boosted my confident so much!

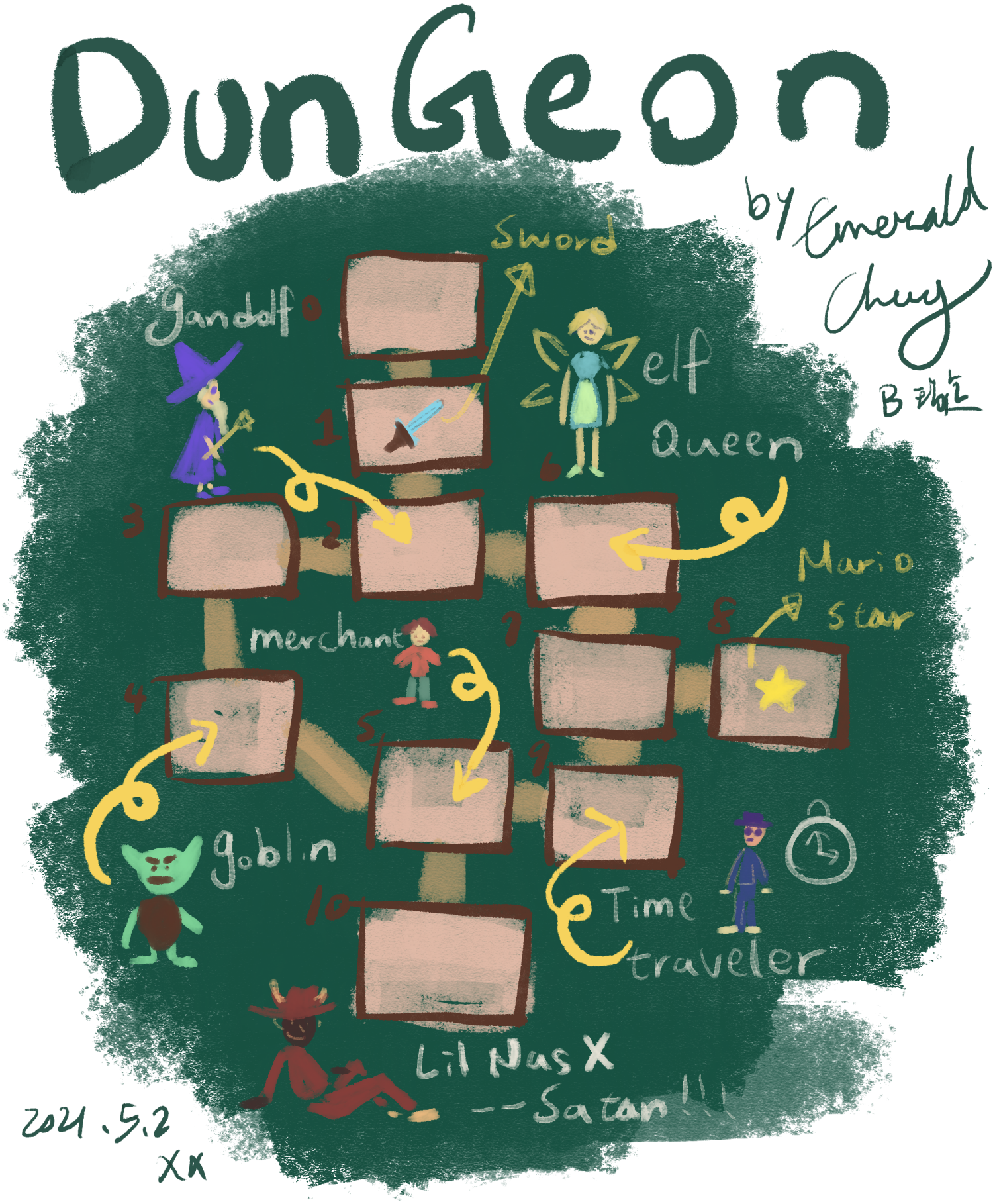
Without even noticing, I spent quite some times on this project. But I think it's totally worth it. The things I learned really paid it off. Besides, the process was super entertaining!

There's definitely room for improvement. For example, there's actually a lot of functions that TAs placed in the template that I didn't make use of, and those could've make my code even more simple and clear. But at the same time, I realize the me from a month ago wouldn't have known how to use those function to its maximum anyway. So I am already really proud of what I've made.

With more and more practices, hopefully one day I can be like my roommate, to have an idea on how to write by simply looking at the template. But at the same time I want to still be as imaginative as I am today, because I think creativity cannot be replace or copy by others.

Thanks to professor and TAs for teaching me about class and for designing lab exercises. And the instructions on the dungeon project in the beginning of each Labs were really helpful!

MAP LAYOUT


DunGeon
by Emerald
Sword
Gandolf
elf
Queen
Mario
star
merchant
Goblin
Time
traveler
Lil Nas X
--Satan!!!
2024.5.2
XX

GAME PLAY WALK THROUGH

- You can choose to whether start new game or load previous one
- Then choose character and race

- Enter room 1 and encounter the sword
- Enter room 2 and encounter Gandalf, he'll suggest you which way to go
- Turn left or right
- Meet elf queen and Mario star and time traveler (if turn right)/ meet goblin (if turn left)
- Enter Room 5 and meet merchant
- Goes to the wrong direction or encounter Lil Has X aka the BOSS

- 4 ways to end the game:

1. chat with elf queen continuously --> win
2. take love potion from merchant and use it on boss --> win
3. combat the boss --> win
4. get kill by the monster (especially when boss enter insane mode) --> lose

THANKS YOU!!
HOPE YOU ENJOY MY DUNGEON!!