Y2_2022_10527 Lucas Foley

1. **Personal information**

Strategy Game Y2_2022_10527

Lucas Foley 910527

Kone- ja rakennustekniikka

05.05.2022

2. **General description**

I have made a turn-based 2D dungeon strategy game. The game involves killing different enemy mobs and progressing through rooms until the player reaches and beats the final boss.

As a concept my game is still the same as I planned it from the start. However, the layout and mechanics have changed from having a team of 3 characters to now having 3 different playable characters: It made more sense to me to have 3 playable character that you can choose from instead of having a set team since it allows for more replayability and more focus on the uniqueness of the characters.

The difficulties for the project are medium and hard. My program doesn't meet the requirements for the hard difficulty mainly because it can't be saved. Having said that my game does deviate quite a bit from what might have been envisioned when creating the criteria as it is not a board game type of game. My game does have many feature and details that make it quite a unique experience compared to what I think was envisioned for this task and I didn't use the robotworld as a base for the project instead the whole program is made from scratch so I would say my game lands somewhere between the medium and hard difficulty.

3. **Instructions for the user**

Download the zip file and extract the files. After the files have been extracted locate the src folder which contains the RUN_ME.py file. Simply run the RUN_ME.py file to start the program.

Once the program is launched the user can follow the instructions in the README file to familiarise themselves with how the program operates and what the different buttons do.

There are 3 different playable characters with different abilities. The first one is the wizard that has a stun ability. The stun ability deals damage and has a chance to stun the target. There is also a small chance of instantly killing the target. It can be used after the stun is applied to deal reduced damage and try to get the instant kill.

Secondly is the warrior. The warrior's special ability increases his attack and armor until it reaches a set sealing.

Lastly is the shaman whose special ability is one of three randomly selected abilities. The shaman can freeze, burn, or absorb his targets HP (health points). In order to keep the shaman character balanced only the first ability applied to the target is activated after each turn. The other abilities can only affect the target once.

4. **External libraries**

   Only PyQt was used for the project since it was the only allowed library.

5. **Structure of the program**

   The program runs on a GUI file which uses a single Window(QWidget) class with a bunch of methods. The reasoning for using a single QWidget class is that it allows to change and add button and background images relatively easily without having to close and reopen anything or open a different class. The GUI handles all graphical related things such as backgrounds, buttons, and dialog boxes. It also allows the user to directly interact with and input choices into the game by using the buttons.

   Then there is the class BasicUnit which is used as a base for the subclasses of playable characters and enemy units. The BasicUnit class handles all unit and combat interactions for the player and enemy units. It performs the calculations, stores information about the units, and any statuses applied to them or by them. There are subclasses for each enemy unit which changes their base values. The playable characters also have their own subclasses with changed values and a method for their special abilities. The only thing that basic units doesn't take care of is the enemies turn in combat. Enemy combat is handled in the enemy_combat_AI file because enemies have varying strategies but the basic combat calculations and such are still handled in the BasicUnit class.

Lastly there is the world_map file which is used as a storage system for all the room related information. It will give the enemies for each room as well as the dialog and backgrounds related to each room.

To summarize and clarify the GUI file interacts with all the other files and classes and is responsible for presenting the interactive window for the user. The GUI is also used to input choices and output combat information that it receives from the BasicUnit class and subclasses. The player and enemy combat is called in the GUI file and then handled with the BasicUnit classes and the enemy_combat_AI file. To keep track of rooms, enemies, dialog, and background images the world_map file is used to retrieve the correct and relevant information.

6. **Algorithms**

   The first and most important algorithm is the combat, and it is handled one turn at the time. It can be split into a normal attack and a special ability attack. For each of those there will be player combat and enemy combat. The simpler normal attack directly compares the players attack to the enemy's armor and calculates an amount to deal to the enemy and wise versa. The special attack is a bit more complex as it differs depending on which character has been chosen. It will call the character specific ability and then use it on the enemy or the player depending on which character you are playing. This is the combat for the players side of things.

   The enemy combat is handled in a separate enemy_combat_AI file which takes in the enemies in a list and sorts out which kind of enemy it is and then checks for any potential statuses that can affect it or possibly stop it from attacking. If there is no status stopping the enemy, it will perform its specific combat instructions.

   The GUI file handles the active combat in a for loop and tracks that the player and enemies are alive. If the enemies die it will move to the next room and if the player dies will send the player to the end game screen.

   The world_map file stores all room specific information and is called by the GUI when needed. It works by having predefined enemies, dialog texts, and background pictures which are called based on the room number. In other words, it is a storage system for all the relevant room information.

7. **Data structures**

   The data structures consist of immutable predefined stored strings and objects. Most of which are in the world_map file. The reason for choosing this

type of data structure is that the program needs string for the dialog box so it can communicate to the player what is happening. In addition to that it also needs a string to set the QPixmap by giving it's path location. The enemy objects are also stored in the same file since they are needed at the same time as the strings.

The reasoning for this is that having the strings and objects stored and predefined allows the GUI to loop and thus greatly reduces the amount of hard coding needed by instead retrieving only the relevant information at the start of each loop.

Having a stored file that would be read from would have been possible but didn't make much sense since there needed to be object and simple strings stored which all can simply be stored in a file directly. Another benefit of the current implementation is that rooms can be added, changed, removed by simply altering a few lines of code. This means there is potential to expand and develop further on the program without having to do much work.

8. **Files**

   The main type of files that the program handles are image files. They are stored in a game_img folder and are called by the program when needed. The only condition is that the files must be 800x800p since this is the set resolution of the game. Smaller image files can be used but there will be graphical problems like duplicate images in the background. Larger image files will not be shown resulting in the background being white but the game still operating as usual.

   I have however left in a base_background and a battle_base that are 800x800p and can be used to add more customized images if wanted. The user only has to change the name of the image file in world_map.py to match the custom image and then it will change to the new image.

9. **Testing**

   Due to the program being so heavily graphically dependent testing was very difficult. Testing was mainly done by printing to the terminal and deliberately causing bugs and debugging to see how the program functioned. When the program had a functioning dialog box it allowed for easier testing by printing to the dialog box values and added markers in the code to see the order in which the program functions and if the correct values were being handled.

Early testing found a lot of issues in combat and unit to unit interactions while later testing mostly involved getting the correct values at the correct timing for combat and room related information.

10. **The known shortcomings and flaws in the program**

    There are a few interactions that aren't being displayed in the dialog box such as the instant kill by the wizard and the giga absorb by the shaman. The dialog box structure and printing could also be cleaned up and optimized a bit better to show the player more clearly what is happening. It wouldn't take a lot of added code, but it is quite time consuming to find the order and positions for all the text and it is quite a small detail.

    Another shortcoming is the lack of accessible character information. Earlier there was planned to be a stats button that would allow the player to get this information but due to implementation problems and time constraints the idea was scrapped. Its implementation wasn't too hard but graphically it didn't match the rest of the program. It could probably be done in a few days with some research and design planning.

    The gameplay could also be more captivating as it is quite one sided after a few run throughs of the game. This is more so a game related problem and not so much a program related problem. It could take a lot of time and might even require reprogramming of the game which could be hard to do depending on what would be changed.

11. **3 best and 3 worst areas**

    By far the best thing about the program is its simplicity and controls. They are very intuitive, and it could be played by simple running it and not knowing anything else. Visual design is also a good aspect of the game as it does look fairly good, and the relevant information is always on the screen. My proudest accomplishment is the game balancing. The characters have varying difficulty going from warrior being the easiest to shaman being an intermediate and wizard being the hardest. It is possible to beat the game with each character, but the difficulty greatly increases depending on who you pick.

    For the worst areas I have to say that there is a lack of information to the player when picking and using the characters. There is no way for the player running the game to find or see the character specific information. The second problem is that there is no continue buffer or mechanism to slow down the turns. Meaning the game can be completed and skipped though by just pressing the continue/attack buttons. Lastly the combat does get a bit boring after a few runs.

12. **Changes to the original plan**

    The only changes made where that of changing from a 3-player team to 3 playable characters resulting in more fun and unique experience for each of the characters. There was and idea for a stats button, but the implementation didn't make a lot of sense and visually it looked out of place, so the idea was scrapped.

13. **Realized order and scheduled**

    After creating the git repository, the first file that I started working on was the basic_unit file. It was created (19 March) and the combat, characters, and enemy units were designed. After that the GUI file was created (5 April), which would make up the majority of the program. I worked on the GUI although April as it required most of work and programming. Once it was in a semi working state, I created the world_map file (11 April) and started working on the interaction between the 2 files. After the GUI and world_map files were in a working state, I created the enemy_combat_AI file. It needed a lot of work and took some time to figure out how I wanted it to interact and work.

    As period 5 started the time I could put on my project reduced but I manged to complete it during the first week of May adding small changes all the way up until the deadline.

    The main change in the plan was in which order the program was to be made in. I had severely underestimated how much work the GUI would need and quickly realised I needed to work on it the most and that's what I did.

14. **Assessment of the final result**

    Overall, I am very pleased with the final result. As mentioned earlier the program was fully designed and programmed from scratch. It is fully working as intended and visually it is quite pleasing and somewhat informative. It has the potential to easily be expanded and changed without too much difficulty because of how it loops and gets only relevant information from the world_map file. The usage of methods and functions have been kept clean and useful to allow someone else to understand what parts do what and how the program functions and there is documentation in the files to allow others to understand what each part and section does. There is still room for improvements and cleaning up the combat interactions and dialog box outputs. It is lacking in features allowing the player to get information about the characters beyond what is printed in the dialog box.

Despite this I would rate my program quite well as it does what is needed and allows the player to enjoy and replay the game a few times for different experiences.

15. **References**

The following sites have been used for learning as well as implementation of the program:

https://www.w3schools.com/python/

https://pythonbasics.org/

https://www.geeksforgeeks.org/

https://doc.qt.io/

https://stackoverflow.com/

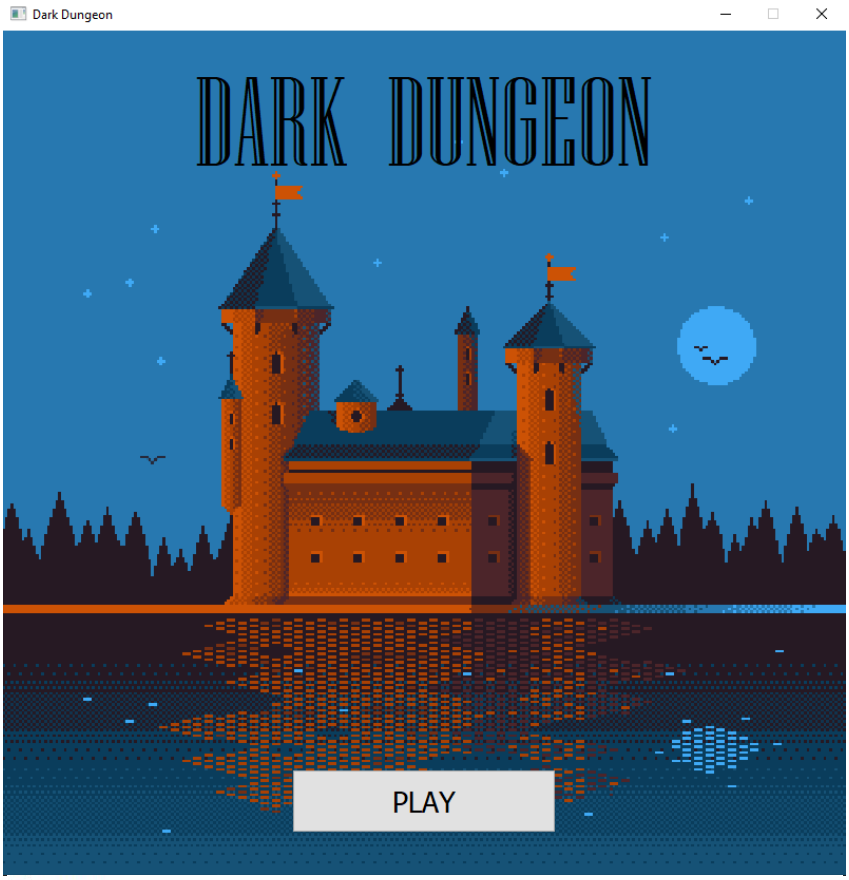
I have also used a lot of youtube videos for various problems and implementations that where needed. One main channel that was used for learning about PyQt was:

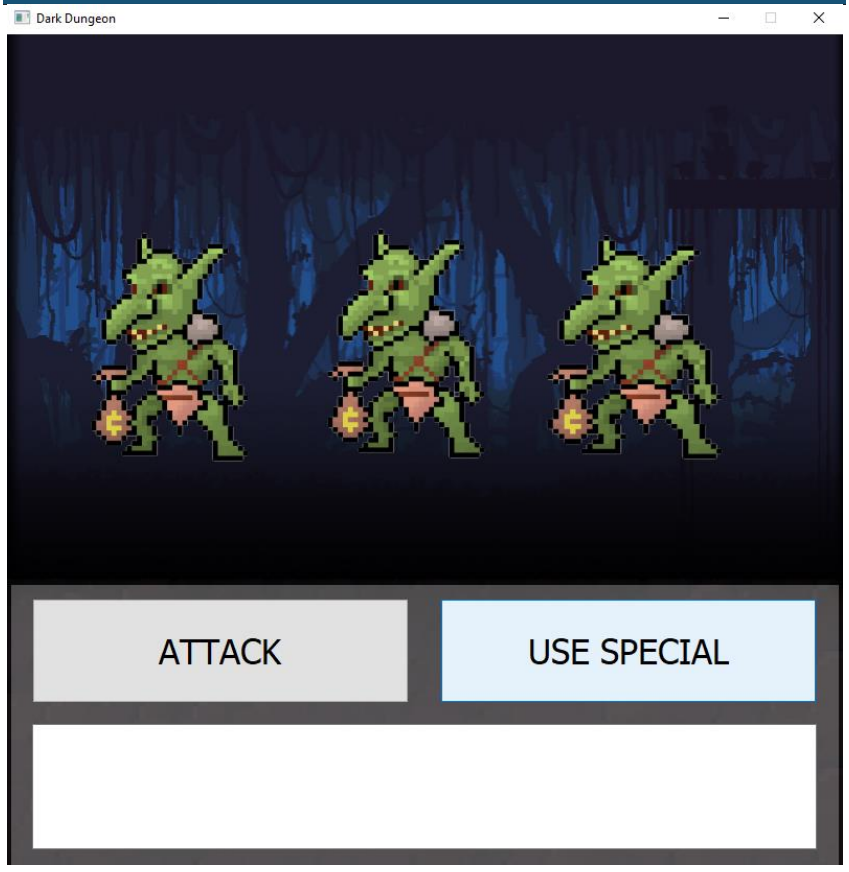
https://www.youtube.com/channel/UCywpR6E1lpk66CHhGziz8Bg

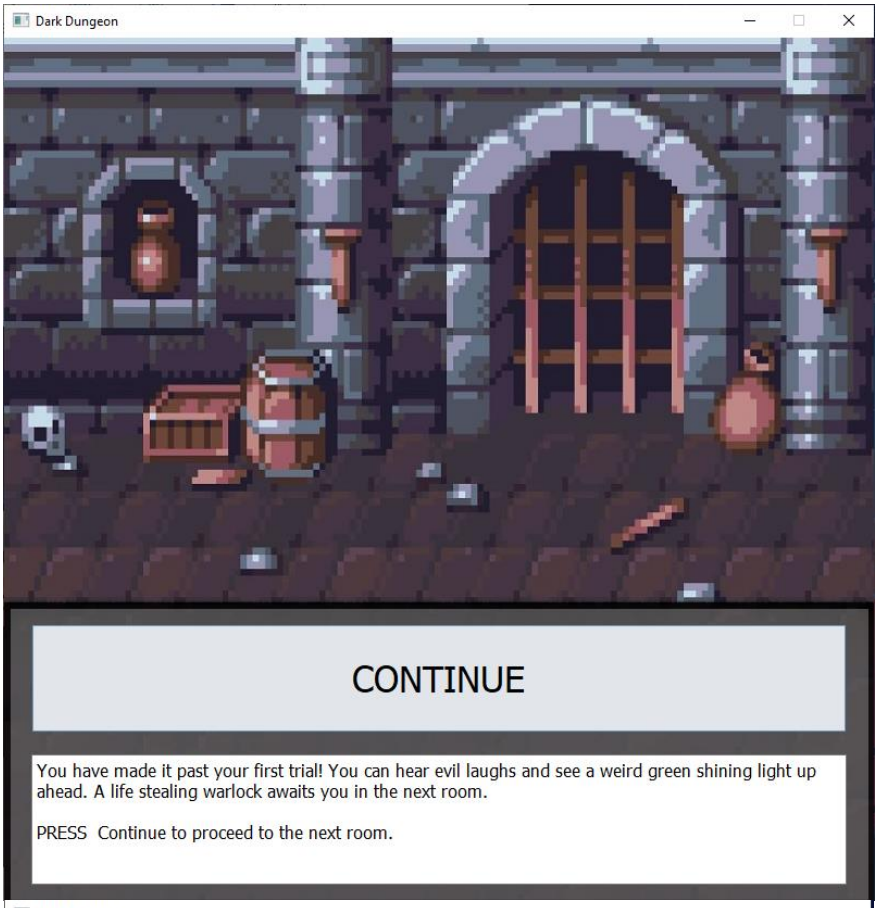
For a game reference I had darkest dungeon since it is at its core something very similar to my game:

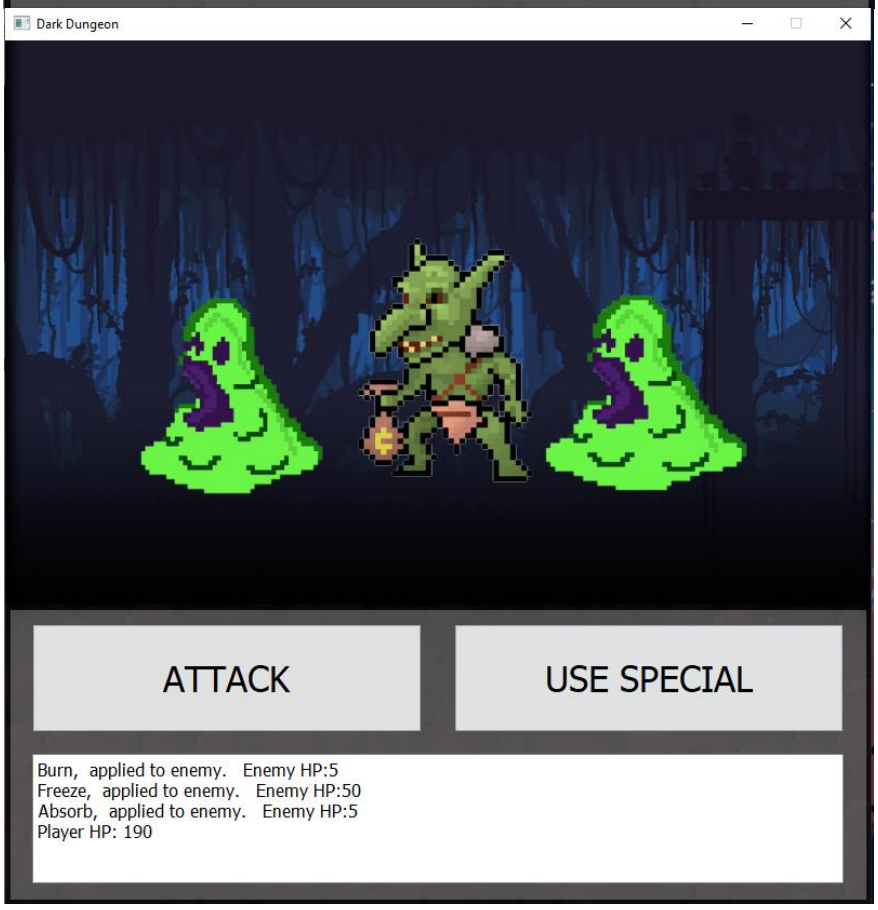
https://en.wikipedia.org/wiki/Darkest_Dungeon

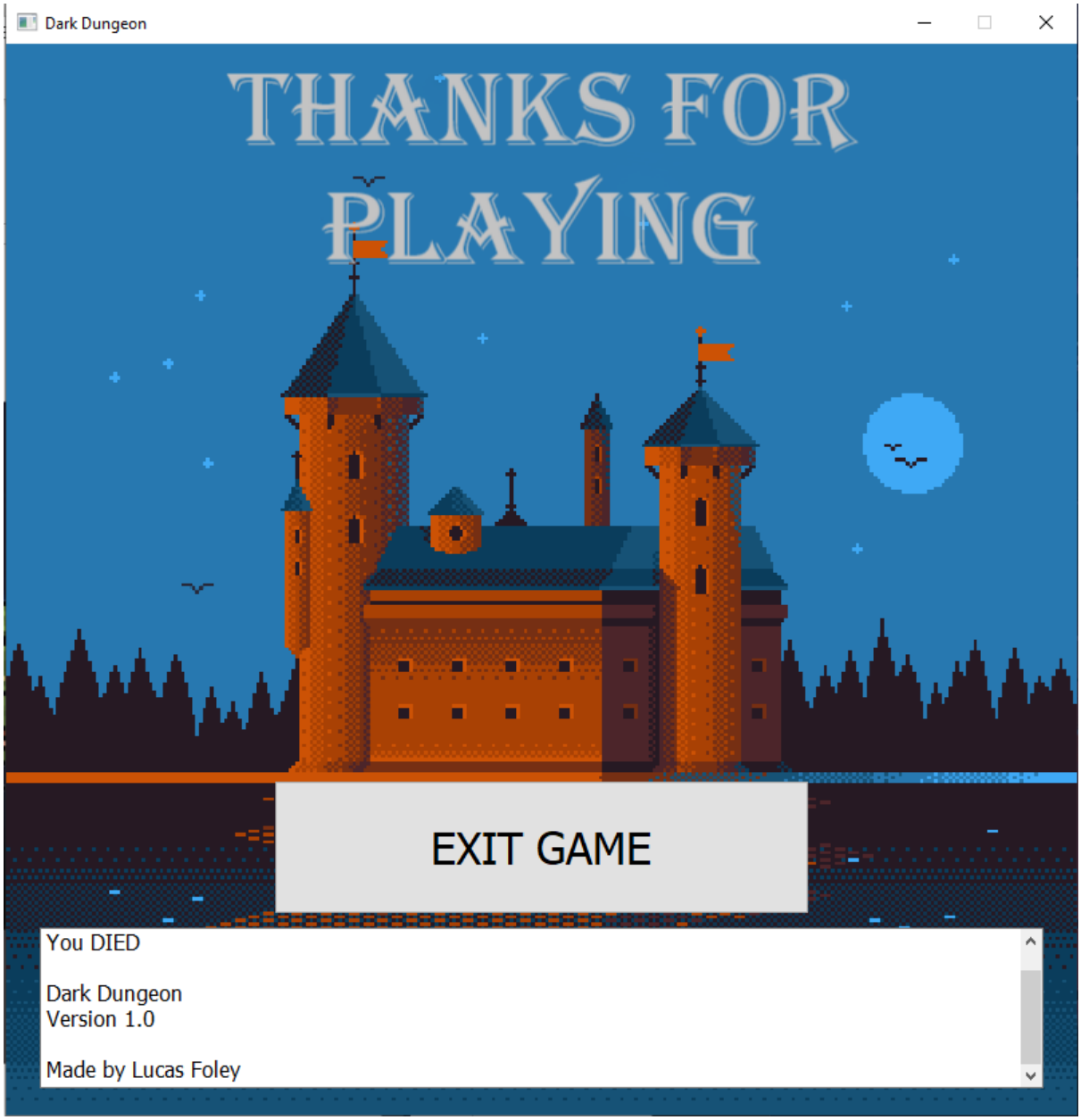
16. **Attachments**






Dark Dungeon
CONTINUE
You have made it past your first trial! You can hear evil laughs and see a weird green shining light up ahead. A life stealing warlock awaits you in the next room.
PRESS Continue to proceed to the next room.


Dark Dungeon
ATTACK
USE SPECIAL
Burn, applied to enemy. Enemy HP:5
Freeze, applied to enemy. Enemy HP:50
Absorb, applied to enemy. Enemy HP:5
Player HP: 190


Dark Dungeon
THANKS FOR PLAYING
EXIT GAME
You DIED
Dark Dungeon
Version 1.0
Made by Lucas Foley