A Text-based Game

Goals

- Design a game with pointer-linked spaces
- Use object oriented programming concepts

Requirements

This is the final project and you will be able to design and implement a one-player, text-based game where the player can move through spaces to get items and accomplish goals.

You have the freedom to decide what kind of them you would like to create for your game as long the requirements below are met.

Space class

1. The game requires a **Space** class, which represents the space the player can be in. The Space class must be an **abstract class** that will have pure virtual functions.

Inside the **Space** class, there must be at least **4 Space pointers**: top, right, left, and bottom**.**

Use the class to create a game with structure of linked space. (You are free to add more Space pointers to the Space class, but must have at least 4 Space pointers)

Note:

- Even if your structure is linear, such as a train, you still have at least 4 pointers in the **Space** class
- Any **unused pointers** need to point to NULL.

2. The game must have at least **3 derived classes** that is derived from the **Space**

Each representing a different type of space, and need to have a special action for the player to interact with. It can be opening the door to another space, or maybe attack the monster, or turn on light switch, or sing a song to please the king.

3. The game must have at least **6 spaces**.

Gameplay

1. The game must have a **theme**. It can be a crime-solving theme, a fantasy game. The game must also have a **goal**for the player, so the player can complete the goal to achieve victory.
2. The game must **keep track of which space the player is in**. It can be in a form of visualized space, by printing the map out and indicate where the player is, or printing text describing where the player is at and what adjacent space is around the player's space.
3. You must create a **container** for the player, to carry "items". The container must have a capacity limit. The game must contain some **items** for player to obtain in the game and one or more of these items **must be required as part of the solution** to accomplish game's goal, such as a key to open a locked door, etc.
4. The game must have a **time limit**, which limit the amount of time/steps/turn the user can take before losing the game. The following are some notable examples of the time limit:
    1. Step limit that limit the number of times user can switch spaces.
    2. Health system which decrease the player's health point from space to space, and maybe some painkillers scattered around the spaces.

Note: make sure you give enough steps to allow the game to perform through testing.

5. Again, the user must be able to interact with parts of the space structure, not just collecting items to complete the game.

Interface

1. At the beginning of the game, the goal of the game must be declared and printed to let the player know the goal of the game.

2. The game cannot contain free-form input. An example of free-form input would be to type out “kitchen” to go to the kitchen space in the game. It is tedious and counter-intuitive.
3. The game must provide the user a menu option for each scenarios of the game.
4. You are not required to have a printed map to visualize space, a text-based game is sufficient. But, it would be great to have a printed map, it is easier to interact with, and it’s cool to show a cool game with a map to friends and family.

Extra Credit

Best theme +10%: the graders will have the option to give extra credit for the most creative, interesting and well-designed games in the class.

What you need to Submit

- All the program files including header and source files (.cpp/.hpp)
- makefile
- Your reflection pdf file

Important: Put all the files in a single .zip file and submit it on Canvas.

GRADING

- Programming style and documentation 10% :
- program runs smoothly as intended on flip, no segmentation fault, no memory leak, all files are put into a zip file, the code is well-commented.
- Create basic game structure 50%:
  - Create at least 6 spaces of at least 3 different types (the abstract and derived classes): 20%
  - Correctly use at least 4 pointers for each space to link them together: 10%
  - Implement a goal for the player to achieve (sequence of actions to win and exit): 10%
  - Keep track of player status: 10%

- Meet the advanced requirement 20%:
  - Properly implement the step limit: 10%
  - Properly implement the interaction of player and space structures: 10%

- Create a menu function to run the game: 10%
- Reflection document 10%: include the design description, test plan and results, comments about how you solved the problems