Design

Space

The base class of all types of Spaces in the game. It has 4 pointers which are used to link other Spaces. There is a function for moving to another Space through the Space pointers. There is also a pure virtual function which shall be defined in derived class, as different Spaces may have different interactions with the player.

Entrance

This class inherits from Space. It is the exit of the building. In order to pass through this space, the player has to defeat all the enemies. Therefore, this class includes some enemy objects to fight with the player.

First Aid station

This class inherits from Space. The only purpose of this class is for the player to recover HP after fighting with the enemies.

Garden

This class inherits from Space. It provides tricky options. Some options may simply waste the time or cause the player dies. However, if the player come to this place with a rope (an item in the game), a new option will be provided.

Street

This class inherits from Space. It allows the player to grab a weapon before fighting with the enemies. This weapon will increase the player's ability.

Activity Room

This class inherits from Space. It provides options for the player to change items to carry. It also serves as the "End turn" button to make the game move on.

Bandit

The class of enemy objects. It mainly contains the ability data of an enemy.

Fighter

The class of the main character. It inherits from Bandit. Other than the ability data, an item container is also included for the players to carry different items. With different items, the player may have different abilities or different options in the game.

Gameplay

The class that act as the flow control of the game. It constructs the game structure and interacts with the player. Game details will be displayed through this class.

Test

Display	Input	Expected output
Start game or not	- 1 - 2 - Symbols and characters	- Start the game introduction - Exit program - Error message, re-prompt
Change space	- Valid space - N/A option - Symbols and characters	- Options in the destination space - Error message, re-prompt - Error message, re-prompt
Action options	- Action options - Leave space option - Get items options	- Perform action (e.g. fight an enemy / recover HP), no. of move – 1 - Ask user the next destination - Check the container capacity, ask the user what to throw away if it is full
End game (play again?)	- 1 - 2 - Symbols and characters	- Start the game introduction - Exit program - Error message, re-prompt

Difficulties and reflections

Everything is fine unless the choice of using heap / stack to create the objects. Originally I plan not to use dynamic memory allocation because I find it from Piazza that we are not required to do so. It is just optional. My program clashes at the very beginning with “invalid read of size x” issues. I proofread all of the code again and again and again and again…and cannot understand the problem. They are all logical correct. I search from stackoverflow and understand that “invalid read of size x” is a problem about memory allocation, which sends even more question marks to my head as I never apply dynamic memory allocation to the whole program.

At last, I give up and try to start with dynamic memory allocation, which I just aim at finding more references to deal with the problem: if it is something about dynamic memory allocation, then I start with implementing dynamic memory allocation. And then everything is fine……the whole program runs smoothly. The

lesson is that, I shall simply use dynamic memory allocation whenever I have to link objects with pointers.