Design:

Problem:

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)

Any **unused pointers** need to point to NULL.

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.

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.
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 scenario 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.

Outline:

Portal type game.
The game will have a time limit for each level
each level will consist of the player trying to unlock a door to the next level

Space Class – must be an abstract class
-Top, left, right, and bottom pointers
Derived Space Classes
- Dungeon
- Entrance
- Exit

Character Class
- Holds the data that defines how the character appears on the screen
- Holds the current location of the character

Event Class
- Handles key presses

Space
- Each of the nodes points to a new DA derived space object. These objects are initialized and added to the queue in the gameplay class. First the 5 levels are added to the queue with each of their pointers set up then the maps are loaded into each object and the objects automatically find the L and D in each map and set them equal to their left and right pointers.

The game will start by asking the player which part of the dungeon they would like to start in first and what color they would like their character to be
The player will choose where they would like to start and the game will set up the map accordingly
They first start at the entrance where they find the portal gun and receive their task
They must navigate through each dungeon room by finding a key and using it to unlock each locked door.
At the end they receive their prize, cake!

Architecture

Space
- up, down, left, right pointers
- Entrance
    - only the up pointer points to a dungeon
- Exit
    - only the down pointer points to a dungeon
- Dungeon
    - up, down left and right pointers are initialized depending on where the player is in the dungeon

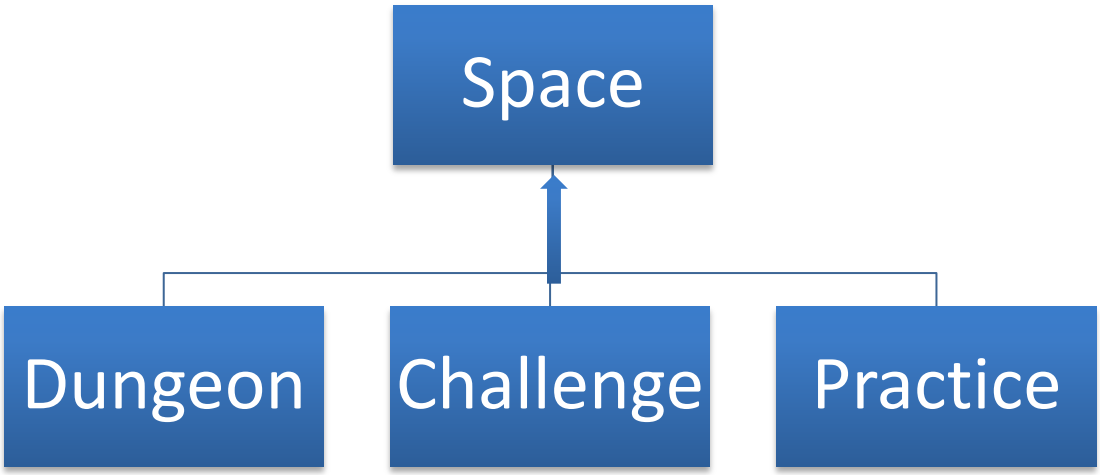
gameplay will loop 10x a second
- poll function will get the key pressed
- gameloop()
    - event->poll() – get the key press
    - event->action (exit, fire portal gun, move player)
    - playerAction()
    - renderMap(keyPressed)
- playerAction()
    - check location
    - check to see if the key pressed can be executed
    - update location

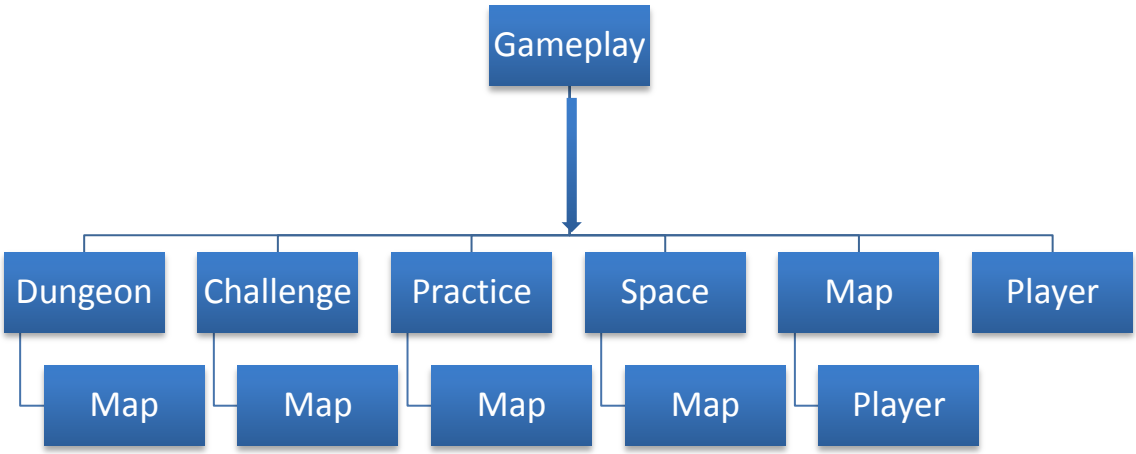
Player
- location – hold the value of the character the player is currently on top of
- direction
- backpack

map
- currentUp, currentDown, currentLeft, currentRight
- renderMap() – checks the condition of portalShot, playerLocation
    - if portal shot is true execute portalShotAnim()
    - portalShotAnim()
        - get the current location of the player and start the portal animation from their current location and direction
- hold *Player and *Space
- bool portalShot – if true render map needs to update the location of the portal shot and player->takeShot is false
- loadMap(direction)
- initialize()
    - can be given a value that will set the first dungeon to a specific map
    - creates linked list of space objects and knits their direction pointers together in a random way

Test Table:

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
Input too low	-1 All inputs	checkInt() intMenu()	Int menu should ask the user to enter a valid input	Int menu asks the user to enter a valid input
Input too high	5 All inputs	checkInt() intMenu()	Int menu should ask the user to enter a valid input	Int menu asks the user to enter a valid input
Input in correct range	1 or 2 All inputs	checkInt() intMenu()	Int menu should ask the user to enter a valid input	Int menu asks the user to enter a valid input
Input extreme high and extreme low	99999999999 And -99999999999 All inputs	checkInt() intMenu()	Int menu should ask the user to enter a valid input	Int menu asks the user to enter a valid input
Input a string of letters	“asdf” All inputs	checkInt() intMenu()	Int menu should ask the user to enter a valid input	Int menu asks the user to enter a valid input
Test use of keyboard to input player direction, fire gun, and quit	w, a, s, d, q, and f keys	poll()	Character should move around on the map, gun should fire, game should return to main menu	Character moves correctly, gun fires when g is pressed and returns to menu when q is pressed
Select practice option	Select option 1 in the game menu	gameMenu() loadMap() printMap()	Practice level should be loaded with the correct time on the timer	Practice level is loaded with correct time amount
Select challenge option	Select option 2 in the game menu	gameMenu() loadMap() printMap()	The challenge level should be loaded with the correct level on the timer	Challenge level is loaded with correct time amount
Select dungeon option	Select option 3in the game menu	gameMenu() loadMap() printMap()	The dungeon level should be loaded with the correct level on the timer	Dungeon level is loaded with correct time amount
Different difficulty levels	Select option 3 in the game menu then select different difficulty levels	gameMenu() loadMap() printMap() setDifficulty()	Each difficulty selection should set the game timer appropriately	Each difficulty level changes the timer for each level to the correct amount of time
Start the game on different map levels	Select option 3 in the game menu then select different map levels	chooseStart() chooseDungeonLevel()	The game should start at the chosen level	The player starts at the level chosen.
Create portals and test teleport ability	Create portals in game and test to see if player is teleported	poll() printMap() makePortal() getEnergyTile()	Player should be transported to the second portal location on the map	The player is transported to the second portal’s location.


Space
Dungeon
Challenge
Practice


Gameplay
Dungeon
Challenge
Practice
Space
Map
Player
Map
Map
Map
Map
Player

Reflection

I enjoyed the time I spent programming this project. This was by far the most difficult project that I have worked on this quarter. The reason it was so difficult was because I had to learn to do new things that I had very little knowledge going into the project. When I first started designing the game I knew that I was going to have to have some means of getting input from the user without disrupting the flow of the game. I wanted the map to refresh a few times a second because the premise of the game required the player to shoot an energy gun. This would require the map to be updated often and so that the energy could follow its trajectory to the place where the player intended it to go. Because of this the first thing I did was research ways of collecting user input in the terminal.

I found several ways of doing this but I only found one way that was portable and could be implemented easily. I decided to use the curses library because it worked on Linux and I found a version of the library that I could install in Visual Studio. All of this took a long time and it was a very frustrating process. Any games I create in the future most certainly won't be using the terminal.

I also had a lot of the confusion in the beginning when I was using arrays to store the map information and output that information to the console. I stored the map elements in 1d char arrays with 2 elements and referred to them using a 1 or a 0 (e.g. cPlayerLoc[0]) but this got very confusing and I started to mix up the row and the column. I decided to create a file with definitions of enumerated data types that I could include in my classes to make things more readable and clear. Once I did this it was much easier to visualize what I was trying to do in my head and it made the code much more "self-documenting".

I am satisfied with the way my program turned out but if I had more time I would probably try to tweak a few things to make it better. First of all I would add some color to the game. The black and white is a bit boring and it would add more interest if the tiles, the player, and the portals were colored. The other thing I would do would be to try to find a way to get the game to flicker less. When I run the game on the school server it is displayed pretty smoothly with only a little bit of flickering now and then. But when I run the game from the debug in Visual Studio I can barely look at the window before I get nauseous. For some reason the Windows terminal does not refresh as fast as the Putty terminal and it makes it very difficult to play.

I learned a lot from this project and I feel more comfortable with abstract and derived classes after it. I also feel much more competent in C++ after this quarter. There was a lot of work between all the labs, projects, and quizzes but it did a good job of reinforcing the concepts from the book and getting me to think critically and problem solve.