Requirements

- Create a game
    - This game must have 10 rooms, with each room having 4 pointers to other rooms (or nullptrs)
    - Have a defined goal to win the game
    - Have a time limit to win the game
    - There must be at least three different types of rooms, inherited from one room type
    - There must be a defined sequence of actions to win that involves more than stepping through the rooms.

Design

I'm going to create a game where the character is in a spaceship. They start at the tail end of the spaceship and need to work their way to the bridge. The goal is to shut down an overloaded warp core. They will need to fight monsters (from assignments 3 and 4) in order to get keys to shut down shards of the warp core.

The design for this is pretty light. I'm simply gluing together different components from our various labs. The rooms will use the room classes from lab8, though I will also inherit this class to make several different types of rooms (Bridge, Standard room, Warp room, battle room). Each of these classes will have the implementation details (like what monster is in it) needed to resolve that player's interaction when they enter the room. Entering the room will trigger text to display the story and also prompt the user with valid moves. Moves will be entered as one character responses, denoted by (letter). Valid moves will be a direction (to move to rooms) or the 'i' key in order to interact. There will also be a special prompt to choose an ally to battle the creatures one finds along the way.

New classes I think I will need to make are a world class and a player class. These two classes will hold data on what the player has accomplished and also server to store the number of turns (time limit) left. In addition, I'll make the world class hold a vector of all the rooms, to serve as a nice way to delete all of my heap allocated memory.

Once the user has turned off all of the warp shards, they can make their way to the bridge, assuming they have the turns left. The game will exit if they run out of turns, win, or request to e(x)it.

Testing

Testing was fairly easy.

- I tested each battle room/warp room pair to make sure I they went together (ie, I couldn't shut down the warp core without the key each creature had).
- I made sure that I couldn't win the game at the bridge without all three warp cores being turned off
- I made sure that the game ended when the number of turns ran out

- I made sure the character couldn't move anywhere they weren't supposed to go.
- I validate most user input.
- I disabled interaction once the room had no more things to interact with
- I ran through the game with valgrind to make sure all memory that got alloc'ed was freed

Reflection

This was a nice assignment because it allowed us to use components from previous weeks to make a cohesive game. I didn't really find there to be many things of challenge in the requirements, though there were certainly a lot of ways that one could push the assignment above and beyond (in terms of creativity or technically). One thing that I don't like about my submission is that the rooms are hard coded in the actual source code. I was trying to use a JSON library to define the rooms in an external file, but wasn't able to get it working in time. This meant I had to roll back to the version that had hard coded rooms, but I think that still meets the requirements of the assignment.

It is cool to see how far we have come this term! I am looking forward to data structures, where I am sure I will learn about many more things that will assist in making even better programs!