Adam Kniffin
OSU ID#: 931492308
Email: kniffina@oregonstate.edu

Requirements

- Create a game with the following aspects.
    - The program must have at least 5 different spaces of at least 3 different types.
        - Each Space will be a class with at least four pointer variables that link to other spaces.
    - Have a space abstract class that will have a special pure virtual function.
        - Each type of space will have a special action.
        - These spaces must have at least 3 different types
    - There must a goal for the player.
    - There must be some sort of time structure.
        - Does not have to be a countdown, but some way that the game can end in a given time period.
    - Must have some way of keeping track which space the player is in.
    - The player must have some sort of container (backpack, knapsack, etc).

Design

- When creating this game I decided that I would make 4 different classes.
    - User
        - Will be able to add and drop items from their bag.
        - Will be used to keep track where the player is in the spaces.
    - Item
        - Will be made up Items that are found in the caves that the user to can find.
        - They will be simple and just hold string variables that describe each item. They will also have a point when they are picked up that will increase the User's strength that will be used later on.
        - Some will be key's that open other doors to progress through the game.
            - This is so the user cannot just walk through all of the spaces and end the game.
    - Cave
        - The cave class will be the "spaces" for my Item's to live.
        - Will hold a vector of items.
        - The cave class will also allow the user to "look". This will give them the choice to look to the north, east, south, or west, and see if they can find an item.
            - If an item is found then they can add it to their bag and continue on their journey.
        - Each Cave object will also contain an array for 4 Cave pointers. These are pointers to Caves that are connect to the current object.

            - Probably be done with an array for this one, to initialize variables to NULL of there is not an item in the space in the cave that the User looked.
            - If they are connected the user can move to them (if they have correct Items).
    - World
        - This will glue all of the Classes together.
            - World will hold a vector of Cave pointers and be initialized in the constructor.
            - This class will allow the user to move to different caves.
                - If the User has proper Items then it will allow them, otherwise it will not.

Testing 1

- With this assignment, as with most of the other ones in this class I will use an incremental testing design.
- I am going to start with the Item class, since it is the base of the whole game.
    - Start by Creating an Item abstract class
    - Create the Item and give it the proper member variables so it can be differentiated between the others.
- I am going to output all of the member variables of the Item class to make sure that they have all been set, and output correctly.
    - The virtual function are all going to be pretty basic get / set functions that will generally just return integers, strings, and simple variables.

Results of Testing 1

- All of my Item classes work as intended.
    - I used an output statement in a main file and created multiple instances of all of my items to make sure the strings, and integers were outputting correctly.
        - They all did. This was a very straight forward class so the implementation was quick, and testing was even fast.

Testing 2

- The next part of my testing was the User.
    - I wanted to make sure that the user class would have the ability to add Item's and remove items from their bag.
    - I will use a vector to hold the Item pointers and then create and add, and remove function that will put an item, or remove an item from the user's bag (the objects vector).

Results of Testing 2

- There was almost no problems with these functions
    - The add function was pretty simple to implement and when testing I just used the main program to try and add an Item that I created to the bag. I then made a

print function that would display the items in the bag so that the user could see what they were carrying.
    - I did the same thing for the remove function and there were just minor tweaks that had to take place during my testing.

Testing 3

- My third phase of testing consisted of setting up the Caves.
- This is also an abstract class and will consist of multiple types of Cave objects that will live in the world. Each Cave object will also have an array of 4 other Cave pointers that will be initiated to NULL in the constructor, and later, set in the World constructor.
    - I want to make sure that all of the variables are returning correctly and will use output statements in the main function to ensure that the strings, integers, are being set correctly.
    - One of the biggest parts of testing this class will be making sure that Item objects are being added to the Cave vector. I plan to test this by using an output statement in main to display the Item's name.
        - I plan to add Items to the Cave in the constructor so that each cave is preset with certain Item objects at instantiation.

Results of Testing 3

- Most of the testing for this class went pretty well.
- I had a little trouble with adding the Items to the vector.
    - I found out that I needed to create a new Instance of the Item's that I wanted to in the constructor by using Item pointers (polymorphism). Once I had this figured out everything seemed to work well.
- When working with this class I also realized that I would need some sort of function for each of my Cave objects to look around the cave and determine if an item was there. If an Item was present I wanted the user to be able to see what it was, and then add it to their bag if they wanted.
    - I created a look function that allows the user to look a certain direction (North, East, South, or West).
        - Initial problems with this was having the correct functions to return the position of the Item. I had to add a get position function to my item class that would return what geographical location the Item was located in the Cave.
            - These were all preset, in the Item's constructor and mapped out prior to creating the class so they all had unique locations.
        - I also had to use the correct pointer arithmetic in order to access the Item in the Cave's vector and then use the get function to ask an if statement if an item is in the direction that the user asked.

Testing 4

- The world Class is where I am going to tie all of the classes together.
- I plan to have a vector of caves that will be instantiated in the constructor.
- I will have to test to make sure that they are being added correctly.

- I am going to have to implement a move function that allows the user to see what caves are connected to the Cave that they are currently in, and allow them to move to one that they choose.
    - Because I am going to have to keep track of what cave the User is in, I will have to add a Cave pointer that will keep track of the current cave.
- I am also going to have a couple of special functions in the World that make the user have a Certain item, or strength to continue to a different Cave.
    - I plan to create a check function that is specific to these caves and returns true if they are allowed access, and false otherwise.

Results of Testing 4

- This was the biggest testing that I had to overcome while making my program.
- As I was creating the World I also realized that I needed a set function to put all of the Cave pointers that each Cave could move to in the Cave function.
    - I created a set array function that take 4 cave pointers as its parameters.
    - The Cave pointers were put into the Cave objects array and could be accessed at a later time.
- The move function in my class also caused a bit of trouble.
    - I ran into trouble with having access to the pointers and accessing all of the cave elements.
        - I solved this by using my current cave pointer and creating a return cave pointer function that would return the Cave pointer in a given spot of the current objects array.
        - This allowed me to set all of the Cave objects array Caves to certain parts and be accessed, rather then sitting in an array.
- The play function is pretty simple and just takes a user input and then uses all of the other functions to play the game based on what the user chooses.
    - 2 basic options
        - Move to a different cave
        - Or use the special ability, look

Reflection

This was the biggest programming assignment I have ever accomplished. When I was finished I had around 30+ files that consisted of multiple abstract classes and used polymorphism throughout the assignment. A TON of code was reused because most objects created from the abstract class was very similar. I think there were many ways I could of made the assignment easier on myself. I could of reused code from previous assignments rather just making this game completely organic. Overall, it was a lot of work but I had a lot of fun making the game and enjoyed the challenges that I faced while doing it. This term I have come a long way in confidence towards programming and am excited about continuing my education in later terms.

I have always enjoyed playing video games throughout my life whether it was MMO's, MOBA's, FPS's, etc, and being able to create a game (albeit very small and boring), is a huge stepping

stone for me. I would love to be a programmer for a game creator and this assignment got me even more excited to go through the program and achieve this goal.