DESIGN & REFLECTION DOCUMENT

SANDRO AGUILAR
FINAL- SECTION 400

March 9, 2019

DESIGN

I began my project by starting with an overall design using an UML diagram so that I could begin thinking about how to plan the overall design of my program. However, even before getting to this stage, I had to first brainstorm game ideas before designing the UML chart.

I decided to use a rooms where the outline would be a grid of spaces each representing a room. The first theme that came to mind was creating some sort of Mad Scientist type of theme where the scientist would go around completing tasks until they could beat the game. However, I did not think a text-based game would do it justice so I decided to do a slower paced game. The theme I decided to go with was an office setting where a worker is just trying to get through his office job without losing his sanity or getting fired. The theme is from the movie “Office Space” where the main character is really jaded from his day to day job. Below are the notes I took in brainstorming what the game would look like:

Space 1: Water Cooler
Space 2: Friend’s Cubicle
Space 3: Your cubicle
Space 4: Meeting room
Space 5: Lunch room
Space 6: Boss’s office

Friend’s Cubicle	Boss’s Office	Meeting Room
My Cubicle	Water Cooler Hall	BreakRoom

Objective

- Get through the day without running out of sanity points or getting fired

Time
- Time passes half an hour for each space you visit
- Time starts at 9am and ends at 5pm

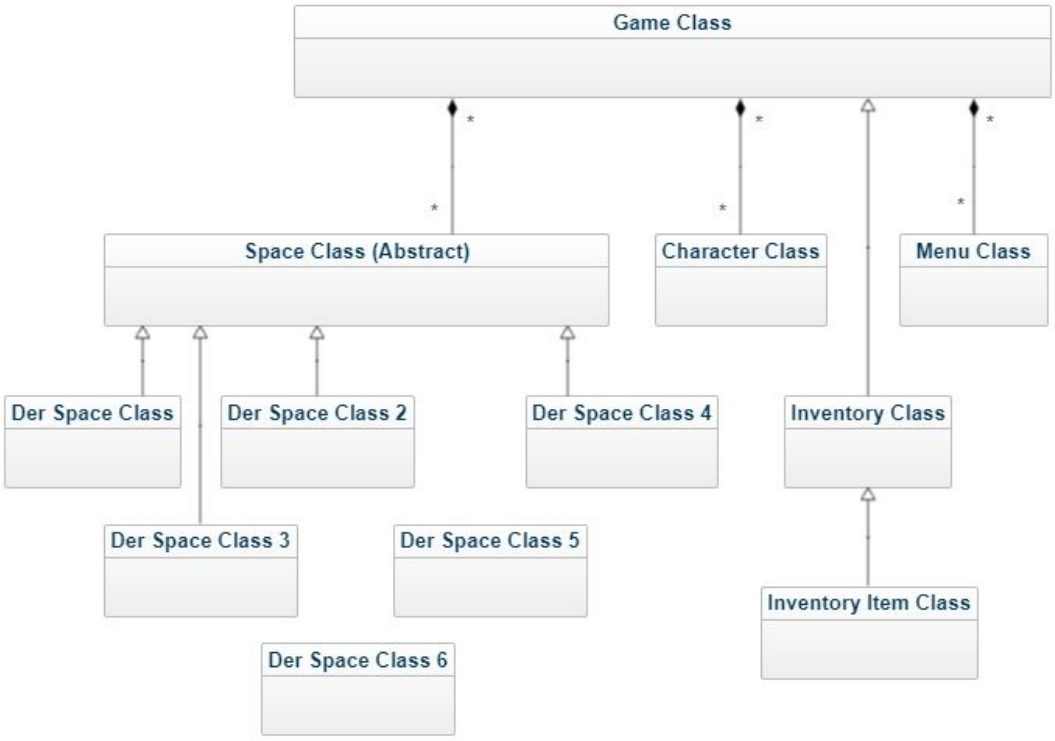
Inventory
- Donuts
- Stapler
- Calculator
- Cell phone
- Car keys

Player Stats
- Sanity points
- Performance points

Activities
1. Your Cubicle
    a. Get to work
    b. Leave work
    c. Boss comes by, lose sanity
    d. Inventory: car keys
2. Water Cooler
    a. Relax with co-workers, gain sanity
    b. Run across annoying boss, lose performance points
3. Lunch Room
    a. Participate in office lotto, 1/10 chance of winning game
    b. Go out to lunch with employees, gain sanity
4. Friend's Cubicle
    a. Pep talk, gain sanity
    b. Walk to coffee shop, gain sanity
    c. Helps you with a project, increase performance
5. Boss's Office
    a. Get more work, lose sanity
    b. Get reviewed, lose sanity
    c. Submit work, increase performance
    d. Stay late, lose sanity, increase performance
    e. Incomplete work, get fired
6. Meeting Room
    a. Bring donuts
    b. Bring cell phone
    c. Get out of meeting early
    d. Stay for the whole meeting

In the following page, after brainstorming the rooms and activities in each room, I was able to move on in creating a UML flowchart that shows the overall structure of the game. Creating this structured really allowed me to get a

good head start in writing the basic code required to hold the classes needed without having to worry too much about detail.



(**NOTE:** I was unable to draw arrows connecting space 5 and space 6 to the abstract class; UML designer I was using did not allow for more).

When thinking of how to implement the linked spaces, I had trouble in deciding how to create the structure to hold them. I did not think that it should be as complicated as in lab 7 where we had a Struct inside of a class since that did not seem to fit for this design. Lab 6 seemed to more closely resemble what I was looking for. So I decided that the linked spaced classes should be inside of the game class since this will allow for access to the spaces from the game class without the need of friend functions. Therefore, as you can see in my diagram above, the Space class is an abstract class inside of the Game class (has-a relationship) and the derived spaces all derive from the Game class

My next consideration was where to keep the player. I think it made sense to create the player using a pointer because I can create pointer variables

inside of the game class as well as inside of the spaces. This will allow me to create a player character inside of the game class and then move the player around inside of each space as needed depending on where it needs to go.

Next, since our player needs to have a container for inventory, I decided to create a class that will have the ability to hold inventory items. The inventory class will be inside of the game class and the inventory items will be objects within the inventory class.

Finally, the menu will be inside of the game class just like how I have had it in prior projects.

TEST TABLE

This test plan is provided to show how I tested the program. I made to include tests for memory leaks since the memory from the heap will be required in order to create objects using the word new.

Test Case	**Input Values**	**Expected Output**	**Actual Output**
Main Menu (start game and exit)	Input < 1 Input > 2 Input = any characters, spaces, and number combination I.e., 1 1 1, i.1 10000, -100000, 11.1, 11. 12	Any input that is not 1 or 2 should be rejected. Selecting 1 starts the game and selecting 2 quits the game.	Only characters 1 or 2 are accepted, all others rejected. Selecting 1 starts the game and selecting 2 quits the game.
Main Menu - Show office map	Input < 1 Input > 1 Input = any characters, spaces, and number combination I.e., 1 1 1, i.1 10000, -100000, 11.1, 11. 12	Selecting 1 should display the office map with the right spaces and indicate where player currently resides	The map is displayed and player's current location is displayed
Main Menu - Hang out at	Input < 2 Input > 2	A submenu should appear that should	A submenu appears that shows interactions they can

current location	Input = any characters, spaces, and number combination I.e., 1 1 1, i.1 10000, -100000, 11.1, 11. 12	provide a user with activities/interactions they can have in the current room they are in	have in the room they are in.
Main Menu - Go to available spaces	Input < 3 Input > 3 Input = any characters, spaces, and number combination I.e., 1 1 1, i.1 10000, -100000, 11.1, 11. 12	A submenu should appear that shows a list of available locations. Only the spaces that are adjacent to a player are allowed as possible spaces to move to.	A submenu is shown that indicates the adjacent spaces for which a player can move to.
Current Room Interaction	Input < 1 Input > 3 Input = any characters, spaces, and number combination I.e., 1 1 1, i.1 10000, -100000, 11.1, 11. 12	A player should be able to pick from a list of things to do in the current space the player is in. Each room has its own interactions a player can have with it.	All six rooms show the activities a player can perform. Entering an integer allows a player to select an activity.
Move To A New Space	Input < 1 Input > 4 Input = any characters, spaces, and number combination I.e., 1 1 1, i.1 10000, -100000, 11.1, 11. 12	A player should be moved to a new location when they select a location to move to. The info display should list the new location the player is in.	The player is moved to the selected location. The info display shows the new player's location. Displaying the map also shows the player's new location.
End of Game Condition - Sanity Points	N/A	If a player's sanity drops to zero, a player loses the game	The game ends and the player loses when their sanity points drop to zero
End of Game Condition - Performance Points	N/A	If a player's performance drops to zero, a player loses the game	The game ends and the player loses when their performance points drop to zero
End of Game Condition - Time	Steps Taken	Every time a player moves, time shown on the info display should advance by half an hour. Once the time reaches 5PM, the game should end. If the player still has all of their sanity and	The time advances by half hour each time a player moves to a new step. A player wins the game if they have any remaining sanity points and performance points when the time reaches 5PM.

		performance points, the player wins.	
Inventory Pick up items	N/A	A player should be able to pick up any of the inventory items available throughout the game and the heads up display should display the inventory in the player's pocket.	A player is given an option to pick up an inventory item where available and add it to to their pocket. The inventory a player is carrying is displayed in the heads up display.
Memory Leaks	N/A	No memory leaks should appear	No memory leaks were detected using Valgrind

REFLECTION

The implementation of the project required several modifications to the structure of the game although they were not very significant. If anything, they actually simplified the way things worked with each other.

For example, I decided to do away with the class holding the inventory object. I realized that I was over complicating something that did not require such significant investment in code or detail. I decided to implement an STL vector to hold various strings that could be used to indicate which items a player was carrying. A vector is simple to use and a string gives is perfect in letting you know what type of item a player is using.

The hardest part about this project was learning how to move the player object around between the linked spaces. I knew that using pointers was key however I was uncertain if I was going about it right. I started out by only creating two spaces and linking them together. I created a menu that would only populate the spaces available (i.e., direction pointer is not null) and this allowed to provide a player a menu that lists the available locations to move to. With this out of the way, I placed various cout statements in key locations to see the address of the player object in order to determine that I was moving the same object around and not any copies of it. I was able to successfully pass the object around by pointing the player pointer to the same address the player object exists and then setting the prior player pointer to

null (see movePlayer(Space *space) function in game class). I was able to successfully implement a nice and short algorithm that receives the space in which the player object needs to be moved to and this allows for the quick and easy transfer of the player object.

I really enjoyed having to implement linked spaces in this final project. I think that Labs 7 and 8 really helped in driving home how linked lists really work especially project 4. I especially like that linked lists are similar in concept to arrays/vectors except that they are more efficient and after learning how to use them, not that hard to use! I really like the flexibility in their design and how we can create them from scratch. However, knowing that there are STL containers for them is also convenient.

UPDATED DESIGN