CS 162 - Final Project - Die Hard OKC
Jeovani Vela
3/20/18

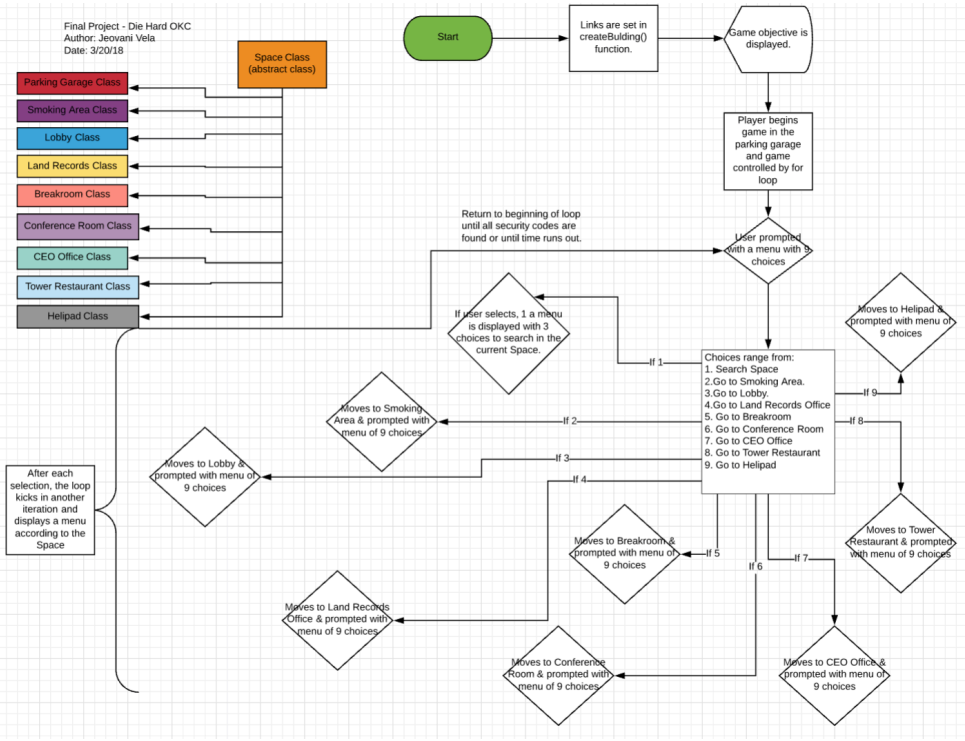
The final project was an awesome project to build and a great coding experience. It consisted of creating your own game with certain requirements to be met. For my final project I created a game named “Die Hard OKC” where NYPD officer John McClane is thrown back into action as he must traverse through Devon Tower to find the 3 security codes necessary to diffuse a bomb that was left in the parking garage by terrorists.

Requirements:

-Create a 1 player game where the player can move through linked spaces to get items and accomplish goals.
-The game must have a theme.
-Must have an abstract class named Space with at least 3 classes derived from it.
    -Each derived class must represent a different Space in the game.
-Must have 4 Space pointers: top, bottom, left, right. (May have more if necessary)
-Use the Space class to create a game with the structure of linked spaces.
-Game must keep track of which Space the player is in.
-Must create a container for the player, to carry “items”.
    -Items collected must serve as a solution to accomplish game’s goal.
-The game must have a time limit.
-User/Player must be able to interact with parts of the Space structure.

Design:

When I began building the game, I followed the same approach as I did in project 3. I began to design my abstract class Space first, which consisted of the required pointers for the game, plus many more as I designed my game to include 9 spaces that linked in a circular linked list format. Next I built one of the derived classes to make sure it inherited the correct member variables and would utilize its own pure virtual function(s). Once I had one of the derived classes in working order I took Professor Zhang’s advice and wrote down the process I followed and steps I implemented to make my first derived so that it could help me streamline building the next eight Spaces and minimize potential coding errors along the way. That technique alone was very useful as it enabled me to become more efficient building the rest of the derived

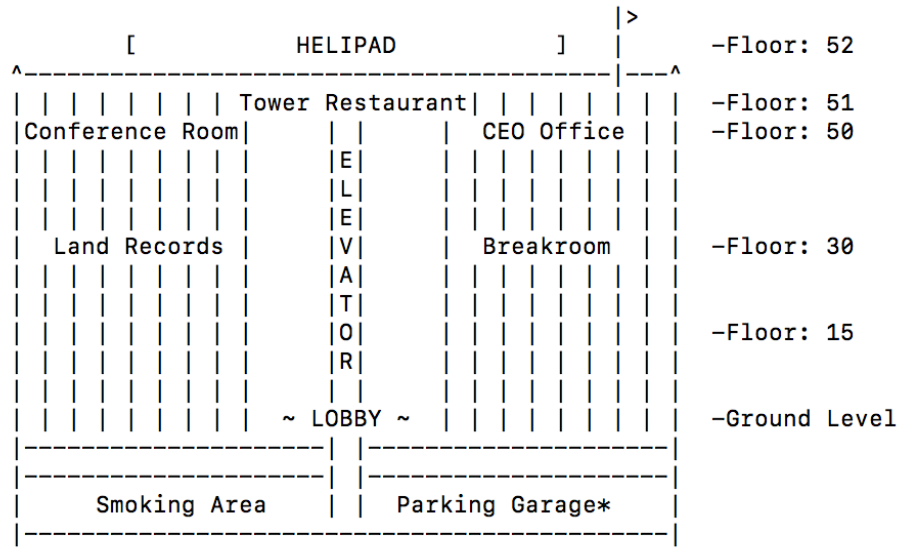


classes before proceeding to include additional code for the interaction portion of the game in each Space.

When the game begins the user is displayed a brief rundown of what it going on in the game and what he/she must do in order to win the game. In this case, they must collect 3 security codes that are somewhere in Devon Tower. I specifically placed one in the land records office on the table, the CEO office in a jacket pocket and lastly one in the tower restaurant kitchen. However in the process of searching for those security codes, clues will be given if the Player decides to interact/help within the current environment/Space and the Player will also encounter potential deadly encounters that can end the game. For example there are certain areas within a Space that if searched, will release a toxic gas or deadly chemical in the air that will instantly kill John McClane and end the game pre-maturely.

Additionally, John begins the game in the parking garage where the bomb is located. The program then displays a menu with 9 selections ranging from deciding to search the current Space or moving to 1 of the other 8 Spaces in the game via a fantasy elevator in the game. The Player will encounter this menu after each move to a different Space and after conducting a search of the Space. If a security code is found and collected after a Space is searched, it is added to the container and a static member variable is changed to true. Each search of an area in a Space, regardless whether it turns up a security code or not deducts an hour from the game. John has 15 hours to find the three security codes required to diffuse the bomb or the tower explodes and everyone perishes. Once all three security codes are collected before time expires, a message will appear that the Player has won the game as they helped John achieve the game's objective of diffusing the bomb in the parking garage before time expires.

Devon Tower Map:



* Location of bomb.

Map located in the lobby security desk and acquired only after saving hostages in parking garage.

Test Plan:

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
promptMenu()- User prompted with 9 choice: Range from searching current Space or moving to 1 of the other 8 Spaces -In parking garage	`45,afasf34,<!#$, fdsg, 45ko()-fe, 11, 0, -2, 2`	Input validator will only accept a number between 1 and 9.	User will get message stating input is incorrect if input is not an integer between 1 and 9.	Program does not continue until user enters correct number. User enters 2 and player moves to a new space, the designated smoking area.
promptMenu()- User prompted with 9 choice: Range from searching current Space or moving to 1 of the other 8 Spaces -In designated smoking area	`xvv,hjgk879, !@#$nm534, -1, 1`	Input validator will only accept a number between 1 and 9.	User will get message stating input is incorrect if input is not an integer between 1 and 9.	Program does not continue until user enters correct number. User enters 1 and player decides the current space, designated smoking area.
promptMenu()- User selects to search a room and is prompted with a menu with 3 choices. -In designated smoking area	`sf,435,-1, 0, 10000, 100, srt546g,<: {()F, 3`	Input validator will only accept a number between 1 and 3.	User will get message stating incorrect selection if input is not between the numbers 1 and 3.	Program does not continue until the user enters a valid input. User inputs 3 and searches a certain part of the Space.Program then returns back to promptMenu() with 9 selections.
promptMenu()- User prompted with 9 choice: Range from searching current Space or moving to 1 of the other 8 Spaces -In designated smoking area	`0, dfsg., 99, 100, dfsg, -4, 3121, a24fsg, 7`	Input validator will only accept a number between 1 and 9.	User will get message stating input is incorrect if input is not an integer between 1 and 9.	Program does not continue until user enters correct number. User enters 7 and moves to the CEO Office
promptMenu()- User prompted with 9 choice: Range from searching current Space or moving to 1 of the other 8 Spaces -In CEO Office	`435, fsdgf, 423aawew234, ()@#$!, fdsgm3, adsf, 1`	Input validator will only accept a number between 1 and 9.	User will get message stating input is incorrect if input is not an integer between 1 and 9.	Program does not continue until user enters a valid choice. User enters 1 and decides to search the Space - CEO Office.
promptMenu()- User selects to search a room and is prompted with a menu with 3 choices. -In CEO Office	`sfdg,bs435mm asf, 121, -1, 0, dafsds, ^&*(DF, 3`	Input validator will only accept a number between 1 and 9.	User will get message stating input is incorrect if input is not an integer between 1 and 9.	Program does not continue until user enters a valid choice. User enters 3 and searches coat rack. Finds a security code, added to the container and static member variable is changed to true.
promptMenu()- User prompted with 9 choice: Range from searching current Space or moving to 1 of the other 8 Spaces -In CEO Office	`sfdgs, dgfs., <><, 111.324., afdsg., p, adsf231!@#B, 4`	Input validator will only accept a number between 1 and 9.	User will get message stating input is incorrect if input is not an integer between 1 and 9.	Program does not continue until user enters correct number. User enters 4 and moves to the Land Records Office

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
promptMenu()- User prompted with 9 choice: Range from searching current Space or moving to 1 of the other 8 Spaces - In Land Records Office	`fcvbv,sdfg/, f5436354, {} DFFS, q, deaf, UT, sdfai12243, 1`	Input validator will only accept a number between 1 and 9.	User will get message stating input is incorrect if input is not an integer between 1 and 9.	Program does not continue until user enters a valid choice. User enters 1 and decides to search the Space - Land Records Office.
promptMenu()- User selects to search a room and is prompted with a menu with 3 choices. -In Land Records Office	`fdsg456, *(*DSF<, 3243124, fdg456., 2`	Input validator will only accept a number between 1 and 3.	User will get message stating input is incorrect if input is not an integer between 1 and 3.	Program does not continue until user enters a valid choice. User enters 2 and searches table. Finds a security code, added to the container and static member variable is changed to true.
promptMenu()- User prompted with 9 choice: Range from searching current Space or moving to 1 of the other 8 Spaces - In Land Records Office	`()*#$#$, F#W$DF, jes23445, 12312,43524, !&*&*#$, 8`	Input validator will only accept a number between 1 and 9	User will get message stating input is incorrect if input is not an integer between 1 and 9.	Program does not continue until user enters correct number. User enters 8 and moves to the Tower Restaurant
promptMenu()- User prompted with 9 choice: Range from searching current Space or moving to 1 of the other 8 Spaces - In Tower Restaurant	`vbnmgj, 234fddgfh, 345<>(*(FD<, 1`	Input validator will only accept a number between 1 and 9.	User will get message stating input is incorrect if input is not an integer between 1 and 9.	Program does not continue until user enters a valid choice. User enters 1 and decides to search the Space - Tower Restaurant.
promptMenu()- User selects to search a room and is prompted with a menu with 3 choices. -In Tower Restaurant	`zdffds, 768()*##$, {}|} {DF@WRER, 2`	Input validator will only accept a number between 1 and 3.	User will get message stating input is incorrect if input is not an integer between 1 and 3.	Program does not continue until user enters a valid choice. User enters 2 and searches kitchen. Finds a security code, added to the container and static member variable is changed to true. -Game ends as user/player collects all 3 codes before time ends.

Reflection:

As enjoyable as it was to build this program due to the minimal restrictions and freedom, I also personally found it frustrating at times. For example before writing any code I sat down and designed my code and what I intended the game theme to be, structure, etc. However as I began to write my code, I began to get distracted away from my original design and began to implement new things on the fly which did not bode well come compile time. I ran into numerous memory leaks which required dedicating numerous hours to re-coding parts of my program or searching for errors in order to fix the memory leaks. As exciting as it is to be able to perform a multitude of different coding things as you write code, it can also lead to extreme deviation from your original design. Thus, after going back through my code and deciding what to keep and change, it served as a leaning experience for me that as excited as I may be to implement something new or additional to my original design on the fly, it is probably not the best idea. I learned I should first finish my code based on my design and instead go back and change or implement those ideas I thought of while coding. This approach will serve me better in the workplace as the other format led me to much wasted time finding errors and not actually building my program which put me behind in my schedule.

Additionally, one thing I learned from this project that I normally overlooked in my design or approach to coding in general are static member variables. Those are a blessing in disguise for a novice c++ programmer like myself. Obviously when you declare a bool variable in a class it must be initialized in the constructor or otherwise you run into an undefined error later. Plus, when you declare an object of that class in another class, it uses the constructor where the bool variable has to be set already and thus makes it very difficult to change later in the game as the program proceeds. Thus, by using static member variables I learned it is much easier to track and change these critical bool values (or any values in that matter) from another class than if I declared them as normal member variables. I firmly believe I will be using more static member variables in my coding future because of the learning experience from building this project.