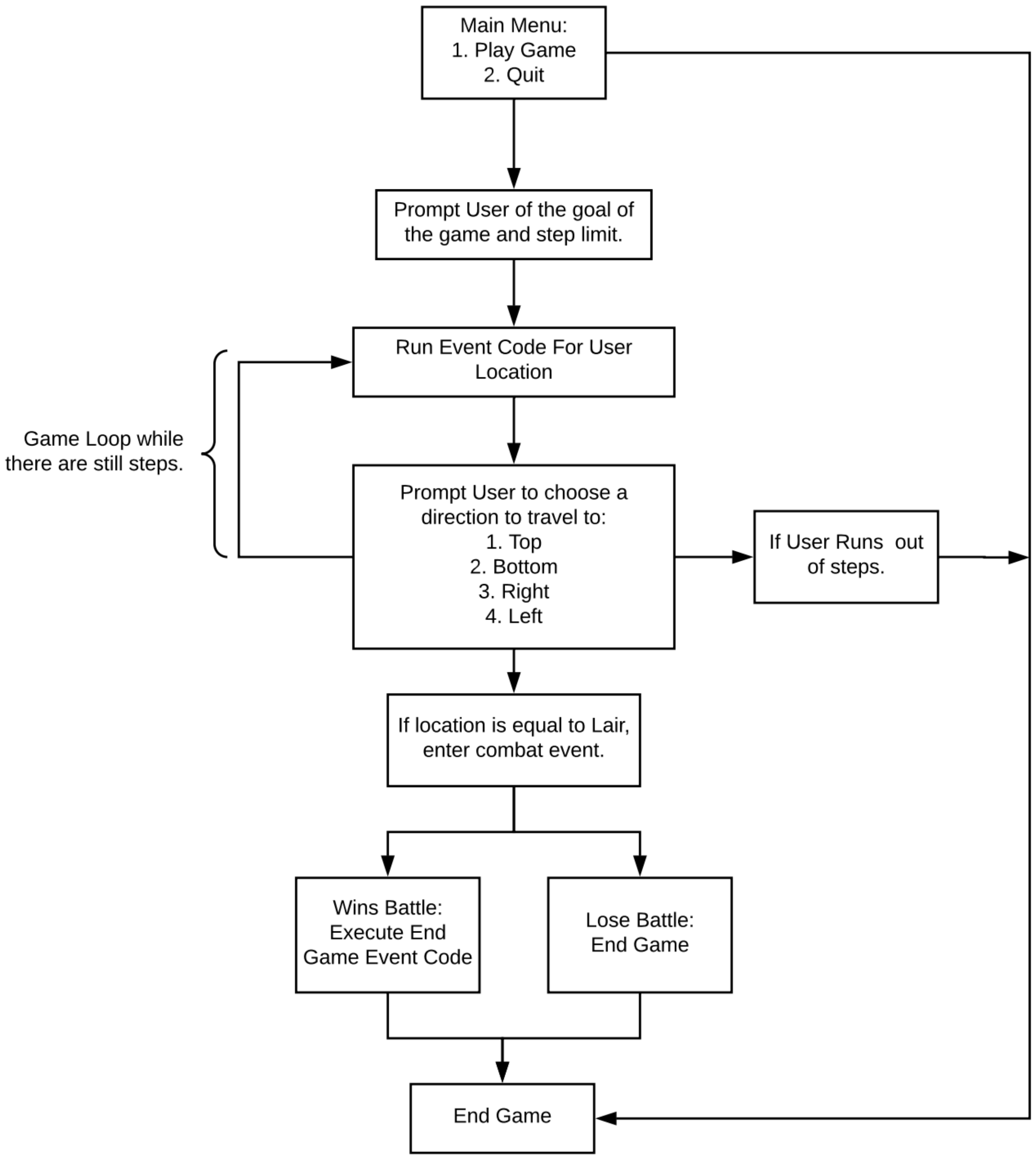
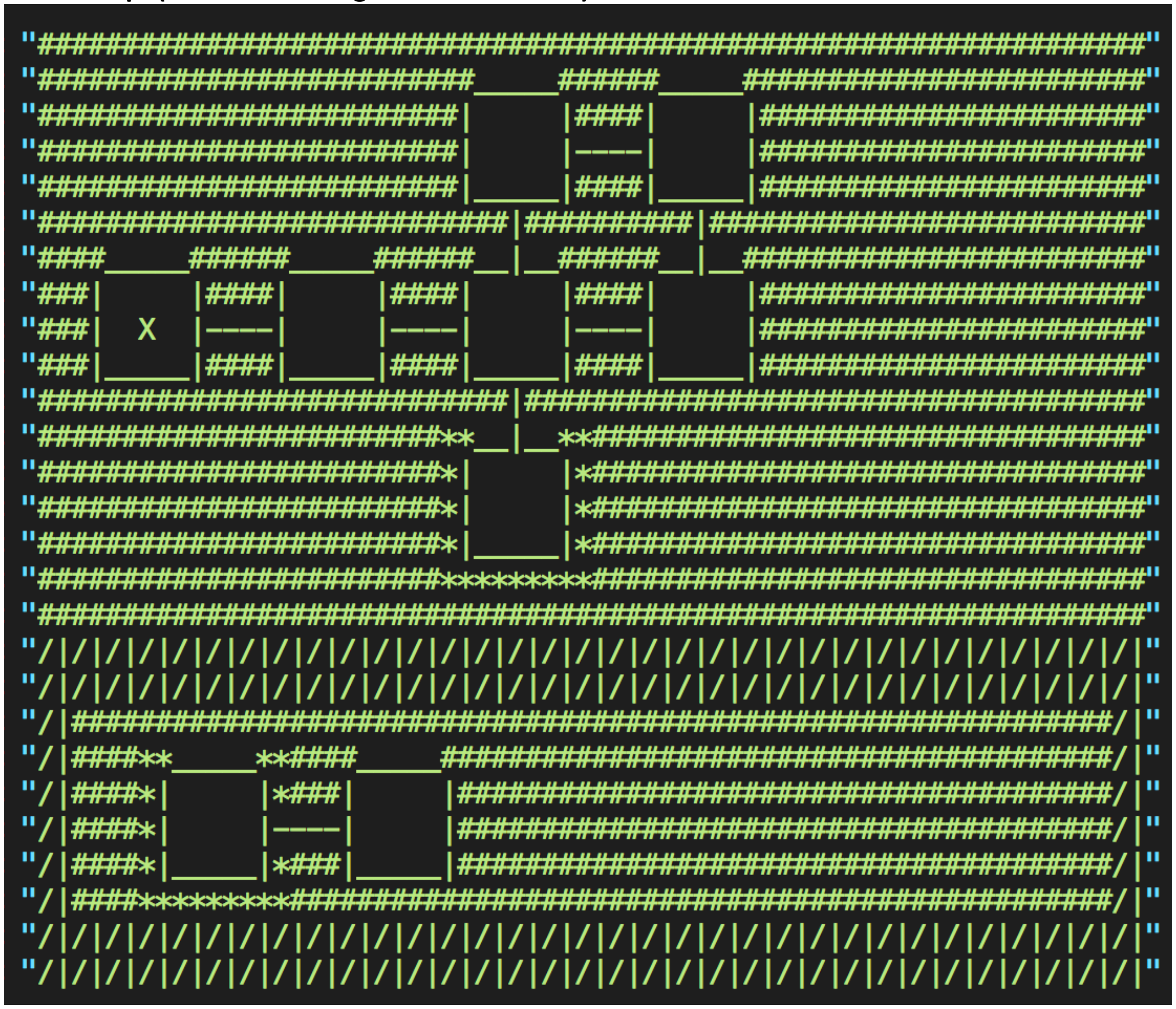
Devin Gendron
Final Project – Text Based Game : SirCodesAlot's Quest

Flowchart:


Main Menu:
1. Play Game
2. Quit
Prompt User of the goal of the game and step limit.
Run Event Code For User Location
Game Loop while there are still steps.
Prompt User to choose a direction to travel to:
1. Top
2. Bottom
3. Right
4. Left
If User Runs out of steps.
If location is equal to Lair, enter combat event.
Wins Battle: Execute End Game Event Code
Lose Battle: End Game
End Game

Game Map: (X marks starting location – Forest)



```
"########################################################################"
"############################______######______##########################"
"###########################|      |####|      |#########################"
"###########################|      |----|      |#########################"
"###########################|______|####|______|#########################"
"###############################|###########|############################"
"####______######______######___|__######___|__##########################"
"###|      |####|      |####|      |####|      |#########################"
"###|   X  |----|      |----|      |----|      |#########################"
"###|______|####|______|####|______|####|______|#########################"
"###############################|########################################"
"##########################**___|__**####################################"
"##########################*|      |*####################################"
"##########################*|      |*####################################"
"##########################*|______|*####################################"
"##########################**********####################################"
"########################################################################"
"/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|"
"/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|"
"/|####################################################################/|"
"/|####**_____**####_____##############################################/|"
"/|####*|     |*###|     |#############################################/|"
"/|####*|     |----|     |#############################################/|"
"/|####*|_____|*###|_____|#############################################/|"
"/|####**********######################################################/|"
"/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|"
"/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|/|"
```

PseudoCode:

```
Main()
    //Menu function
    //switch statements to play game or quit
Menu()
    //main_menu()
        //input validation for main
    //char_choice()
        //menu validation for player movement
    //contCombat()
        //validation for continued combat
Space
    //parent object structure
        //sets directions to null
        //getters for directions
        //function to create child objects (map locations)
Character
    //Character object
    //constructor sets health, attack, defense, weaponry, armor, key bool, and current
    location
    //Creates inventory
    //attack/defense functions for combat
    //Travel functions for each direction to move character

Event
    //runs all game events
        //door event for portal (key accesses portal)
        //combat → occurs when lair is reached
        //item events → add items to user inventory
        //dialogue events → activate story and quest stories
Game_Simulation
    //creates character and event objects in constructor
    //simulation function → main game loop
        //sets character and event objects
        //creates map objects and sets them
        //begins game loop
            //prints player's current location on map
            //runs events
            //prompts user to choose direction for movement
                //if rift_gate map is reached and user has key → travel through
                portal
                //if user is at lair → enter combat.
                //If user wins, game is won! If user loses, … game over!
```

Test Plan:
Menu();

Test Case	Expected Outcomes	Observed Outcomes
Input letters	Inputvalidation(); returns “Incorrect entry…”. Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input symbols or space	Inputvalidation(); returns “Incorrect entry…”. Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input too low	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Input too high	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Empty input	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Correct Input	Counter for exit increments, print correct entry statement. Program continues.	Returns value to main to continue with program

charChoice();

Test Case	Expected Outcomes	Observed Outcomes
Input letters	Inputvalidation(); returns “Incorrect entry…”. Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input symbols or space	Inputvalidation(); returns “Incorrect entry…”. Restarts loop.	Incorrect entry statement printed, loops to restart input request.
Input too low	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Input too high	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Empty input	badEntry bool sets to true, restarts loop until correct input.	badEntry statement printed, loops to restart input request.
Correct Input	Counter for exit increments, print correct entry statement. Program continues.	Returns value to main to continue with program

setCurrentLocation();

Test Case	Expected Outcomes	Observed Outcomes
Sets character location	Ability to use locations event functions through inheritance	Event functions executed without issue.

getCurrentLocation();

Test Case	Expected Outcomes	Observed Outcomes
Finds characters current location	Accesses characters private members to see Location it is currently set to.	Character location is accessed.

Travel functions

Test Case	Expected Outcomes	Observed Outcomes
Travel_top();	Attempts to travel to “top” map location. Travels if location is found, returns null and statement otherwise.	Character travels or receives proper statement due to a null location.
Travel_bottom();	Attempts to travel to “bottom” map location. Travels if location is found, returns null and statement otherwise.	Character travels or receives proper statement due to a null location.
Travel_right);	Attempts to travel to “right” map location. Travels if location is found, returns null and statement otherwise.	Character travels or receives proper statement due to a null location.
Travel_left();	Attempts to travel to “left” map location. Travels if location is found, returns null and statement otherwise.	Character travels or receives proper statement due to a null location.

Item Event

Test Case	Expected Outcomes	Observed Outcomes
Executes event that adds an item to the characters inventory in accordance with it's current map location using inheritance.	Item event executes and item is added to players inventory	As expected

Dialogue Event

Test Case	Expected Outcomes	Observed Outcomes
Executes event that prints story/dialogue in accordance with the current map location using inheritance.	Dialogue or story is printed	As expected

Action Event

Test Case	Expected Outcomes	Observed Outcomes
Character inventory is checked for key. If key is found, the user can enter the teleportation event and move to the next map location.	User has key and is teleported from the rift gate location to the mountain pass location.	As expected
Character enters combat loop.	Enters combat event with seg fault enemy.	As expected

New Location

Test Case	Expected Outcomes	Observed Outcomes
Creates a new map location using the Space class members.	Creates a map node that connects to the other map locations that the player can traverse through.	As expected

Simulation

Test Case	Expected Outcomes	Observed Outcomes
Game run from start to finish	All objects for the game are properly created and the simulation runs to completion (win or lose).	As expected

Reflection:

When I read the requirements for this assignment, I knew immediately what I wanted to design. I began drafting in my notebook and developing my program's structure. After the Node and Queue labs, designing the Space class for the map locations was not difficult. When planning the functions to create the derived maps and their objects, a creation function was implemented to easily take the members of the Space class and create corresponding map node locations. These locations made up the game map and inherited event functions that correspond to each location.

The Event Class contains many of the assets for the game. It contains the game story, the games items, and two specific events – the teleportation "door" event and the combat event. It acts as the main controller asset for the game as the user moves through the map locations as the game loop persists. This is where I hit my first roadblock, the event class. The game was constantly crashing whenever the character object was called within an event. I discovered this by force setting my character's starting location to a different map location to see what the issue was dependent on. I found that dialogue and story was executed without issue. Thus, it was dependent on the character's interaction with events. Since the game was having an issue with segmentation faults (ironically based on the game's story), I knew some kind of memory access violation was occurring. While attempting to solve this problem, it allowed me to discover other smaller issues has I combed through the code. What I found to be the issue was in my simulation function. Although I had set my character and game events objects to null in the game simulation constructor, I did not properly place them in the simulation function when they were assigned. Previously, game_event was being passed as null and then being assigned Fixing this logic issue allowed for the game_events object to be properly passed into the map location objects, allowing the program to properly run.

Once this issue was fixed, I was able to cleanup some redundant code such as print statements when a character tried to move to an impassible location where there wasn't a map node. It was here I also implemented a visited bool, which would determine if my character has visited the space yet. This prevented the main story from being replayed over again if a player decided to revisit a map location. Combat was taken from the Fantasy Combat project and used for the final combat sequence. All of these final implementations came without issue and my program ran fine within my IDE. I then transferred it to the school server to do a final test and ran into my final issue. My character file was receiving errors. Seeing as how my IDE didn't find any problems with my code, my initial reaction was to check my makefile. Upon closer inspection, I found that I included my character object file twice instead of a cottage object file. After that quick fix, everything ran without issue.

As I planned and designed this project, I felt like I didn't learn much. What I did feel was that this project was a representation of what I have learned since starting this course. Nodes were very difficult for me when I first started them, but in this project, I found that they were simple after planning out the map locations. Dynamic memory and pointers have also become a much easier force to tackle. I've been very happy with my progression and find myself solving issues and implementing these features much faster than I did just a month ago. A recent example was my makefile error. I was able to understand that the issue originated from that location because it wasn't connected to my IDE when the program was compiled. Previously, I may have scoured over my code looking for something that might have caused the issue – now, these issues are quickly resolved. As for changes in design decisions, none were made other than changing one map location thus making the whole map more intuitive to the player as well as to simplify it.

Overall, I was extremely happy with this course. It was very difficult and time consuming the first four weeks, but afterwards, I found myself understanding what I needed to do and after some designing, being able to create the assignments in a timely fashion. They definitely built upon each other, which helped a lot. By the end, it wasn't a matter of learning material, but creating a strong design and then setting aside the time to implement it. As I head into the next term, I find my mind buzzing with ideas of programs I can make on my own as well as analyzing everyday software and thinking about "how it must work". I find myself more excited after every course and 162 was no exception.