Travis Laxson
CS162 – Final Project

# Final Project Plan and Reflections

## Game Description

'SaveTheGang' is a CLI text-based game in which the player names their protagonist tasked with the ultimate challenge of maneuvering through cold, dark sewer tunnels to save the Loser Gang members from certain death at the hands of Pennywise the Dancing Clown. Defeat IT and 'SaveTheGang' before the time expires or else!

## Design Descriptions

Plan and solve for:

- Game environment
  - Game Universe
    - Setting of the game
      - Sewer
    - Space availability of the game
      - Sewer Tunnels
      - Sewer Rooms
      - IT lair
    - Characters in game
      - Protagonist: User input
      - Loser Gang
        - Billy
        - Ben
        - Eddie
        - Richie
        - Beverly
      - Antagonist: IT
    - Game objects
      - Items needed to save the gang and defeat IT
        - Knife
        - Flashlight
        - Lighter
        - Inhaler
        - Slingshot
        - Silver rock
  - Input validation
    - Movement direction
    - Menu choice action

Note: A visual copy of the SewerWorld universe is included along with a step-by-step solutions manual to complete the game efficiently.

# Design Descriptions, cont.

## Space class

The Space class required the space the player can be in during gameplay and also must be an abstract class with some pure virtual functions. Thus, all of the locations in the game will inherit from the Space class. The concept to meet this requirement was pointers. Pointers could be used to connect the Space class objects. During gameplay, the player will be able to move forward, back, left, and right. Thus, pointers will need to be created for each space. Edge cases and game universe design will dictate NULL pointer usage. In addition, to interact in the SewerWorld universe, Space class will also contain five virtual functions:

1. talk() = 0;  (pure virtual)
2. look() = 0;  (pure virtual)
3. battle();
4. read();
5. rescue();

```
                          Space Class
protected member variables:
    Space *forward
    Space *back;
    Space *left;
    Space *right;
    bool isRescued;
    bool allRescued;
    Character *character;
    string spaceName;

public member function:
    Space();
    Space(Space*, Space*, Space*, Space*);
    ~Space();
    void setLocation(Space* f, Space* b, Space* l, Space* r,
                     Character*, string);
    bool getIsRescued();
    bool getAllRescued();
    string getSpaceName();
    Space* getForward();
    Space* getBack();
    Space* getLeft();
    Space* getRight();
    virtual void talk() = 0;
    virtual void look() = 0;
    virtual void battle();
    virtual void read();
    virtual void rescue();
```

# Design Descriptions, cont.

## Character class

The Character class object serves as the game's main protagonist. The player can enter the name of the hero Character class object since the class contains a string variable for the name of the hero. The Character class also uses a vector of strings to represent and hold the names of items acquired during gameplay. Both of the preceding variables are private member variables. The public member variables will need to have setter/getter functions for the hero's entered name. Also, it will need a method to add items and check the items in the vector list. Public member functions will also include movement functions for moving in each of the four main directions: Forward, Back, Left, and Right. The current space getter function will point to Space class to get the current space of the hero character.

```
                        Character class

protected member variables:
    string name
    Space *currentSpace
    vector<string> items

public member variables:
    Character(){};
    Space* getCurrentSpace()
    string getHeroName()
    bool itemSearch(string)
    void setHeroName(string)
    void setCurrentSpace(Space* currentSpace)
    void addItem(string)
    void moveForward();
    void moveBack();
    void moveLeft();
    void moveRight();
```

# Design Descriptions, cont.

## SewerWorld class

SewerWorld class was designed and implemented to instantiate all of the game settings (i.e. spaces) allowed to navigate by the player. The character will be instantiated in this class in addition to defining the starting point location of the hero character. The pointers used in this class helped connect the different Space class objects to create the SewerWorld universe for the game. This class was designed to hold the main pointer variable to the Character class object, thus allowing the character and subsequent pointer variable to move/point in each Space class object (i.e. game location).

```
                          SewerWorld class
private member variable:
    Character *character;
public member variable:
    Character* getCharacter();
    SewerWorld();
```

### Game Design and Setup

In order to create the SewerWorld universe, all of the Space class objects are instantiated in SewerWorld. Each of the four movement Space object poiners will point to their spaces accordingly—forward, back, left, or right. Each Space class object also contains a pointer to the SewerWorld primary Character class object (i.e. hero character) to get the location of the hero. SewerWorld is first instantiated in the playGame( ) function, found in GameFunctions.cpp, and then the Character class object is stored in a pointer used during gameplay.

### Planned Space Class Objects

I planned the following Space class objects to create the SewerWorld gameplay universe:
Sewer Entrance – Starting point of the game
Sewer Tunnels 1-5 – Navigate through each of the tunnels
Main Sewer – First Sewer Room
Deep Sewer – Second (Deeper) Sewer Room
Pennywise Lair – Location of Pennywise and final battle
Loser Gang Members (1 – 5) – Billy, Ben, Eddie, Richie, Beverly

Travis Laxson
CS162 – Final Project

| Test Cases | Input Values | Functions | Expected Outcome | Observed Outcome |
|---|---|---|---|---|
| Value too low | choice < 1<br>d < 1<br>(Note: d pertains to the direction of movement) | validate (int &choice)<br>validate_direction (char &d) | Input validation function will signal unacceptable values and prompt for new values. The player will be forced to enter an acceptable value for gameAction function to proceed and ultimately return the value back to the runGame function.<br>Likewise for the validate direction function. However, the player is forced to move in an acceptable direction based on the SewerWorld design. Values less than integer value 1 are seen as garbage values since it only accepts character arguments. | action/choice < 1: Value is tossed back and the player is asked to reenter acceptable value 1 – 6<br>d < 1: Seen as garbage value and the validation function tosses the inputs until an accepted character argument is received. |
| Value of 0 | choice = 0<br>d = 0<br>(Note: d pertains to the direction of movement) | validate (int &choice)<br>validate_direction (char &d) | Same as value too low. All input validation functions will continue to throw values back until either 1 - 6 is entered for action/choice and character arguments are inputted for the directional movement of the character. | action/choice = 0: Value is tossed back and the player is asked to reenter acceptable value 1 – 6 repeatedly until the input is within the accepted range.<br>d = 0: Seen as garbage value and the validation function tosses the input until an accepted character argument is received. |
| Value in accepted range | choice = 1 - 6<br>d = 'f','b','l','r'<br>(Note: d pertains to the direction of movement) | validate (int &choice)<br>validate_direction (char &d) | Input validation function for choice will accept all arguments 1 – 6 and the direction validation function will accept all character arguments for 'f','b','l', and 'r'. | The program compiles and runs smoothly, but it does contain a small memory leak. |
| Value too high | choice > 6<br>d > 'r' (V, z, Z)<br>(Note: d pertains to the direction of movement) | validate (int &choice)<br>validate_direction (char &d) | If the player enters values greater than 6 for action/choice, then the validation function will throw the values out until the user enters a valid input 1 – 6.. Likewise for direction (d), any other letters besides 'f','b','l', or 'r' will throw the values until an acceptable character argument input. | action/choice > 6: Values are tossed and player is asked to reenter acceptable value<br>d != f,b,l,r: Value is tossed and program will not proceed until acceptable character argument inputted. |
| Negative values | choice < 1<br>d < 1<br>(Note: d pertains to the direction of movement) | validate (int &choice)<br>validate_direction (char &d) | Same as if values entered were too high or too low<br>All input validation functions will toss values until valid integers are entered. | All input validation functions toss negative values until the us enters acceptable values. |
| Garbage value with characters | choice < 1<br>d < 1<br>(Note: d pertains to the direction of movement) | validate (int &choice)<br>validate_direction (char &d) | Symbols and values that may cause the program to crash will be tossed by both 'validate' functions. Each function will continue asking for valid inputs until received. | All invalid characters or symbols are flagged by each 'validate' function until acceptable arguments are inputted by the player. |

Travis Laxson
CS162 – Final Project

## Reflections

This final project ultimately tested our ability to apply most of the concepts learned throughout the semester into a large interactive CLI-based game. Although I ran into several challenges during the design and build during this project it proved to be a valuable learning experience to cap the semester.

One of the initial difficulties was designing a plan and game style not only to meet the project requirements but also to add a bit of creativity for players to enjoy. It took me a little while to brainstorm through ideas in which the player would be able to move throughout the game, use deductive logic to figure out a solution, and maintain the importance of the game objective. I solved this stage of the project after watching the new 2017 movie 'IT', based on the novel written by Stephen King. It dawned on me that the spaces could be the sewer system, the Losers Club are my characters, along with the antagonist IT, Pennywise the Dancing Clown, as the main objective to defeat. This proved to be an excellent idea to try and implement.

Next, the trouble began with designing a solution to implement the proper movement of the main protagonist (i.e. player) through the sewer system. In other words, I needed to find a method to move the character between each of the spaces. One idea I began exploring used the concept of Boolean values that would be reassigned as the player moves from one space to the next. However, that meant that I would need to constantly check several Boolean variables and introduce a possibly difficult bug within the deeper parts of the sewer system. Thus, I eventually decided it was more optimal to use pointers and have each space 'point to' the same Character class object outlined in the Character class. In essence, this would allow the same ability to move throughout the sewer system, and the current space value would change to the adjacent space value in the Character object accordingly.

Additionally, I ran into other extraneous design decisions for how to implement a solution to rescue the Loser Club members. One of my original ideas was to use the protagonist, or player, just search for each through the sewer system. Then, I added Boolean variables to the final project plan so that the main character needed a starting object at the very beginning of the game to free the members. If the user didn't acquire this object before entering the sewers the game was almost surely lost from the start. I solved the issue of difficulty and added another layer of complexity, but it ended up being a better solution to keep the game intriguing.

Finally, the last bit of trouble came from not linking all of the appropriate .hpp and .cpp files to each of the respective sewer paths, characters, and game functions files. There were numerous files in this game that I struggled to keep track of all the moving pieces at times. Nevertheless, despite a few syntax errors in the main implementation I succeeded by adding more abstraction to the program to keep the main implementation function clean. Overall, I thoroughly enjoyed this project since it gave us a sense of freedom to really branch out and utilize our creativity to create something. We had very few restraints in this final project, and I was able to come up with a solution and ultimately a game that I was proud of as well as a game that was challenging, engaging, and a little fun (albeit corny). Nevertheless, I was happy with the final result and enjoyed the creative design process and greater sense of freedom for this final project.

Travis Laxson
CS162 – Final Project

# Save the Gang and Defeat IT

**Game Notes:**

1. Game time limit is set to 65 min. where 1 move = 1 min. Thus, player has at most 65 turns to complete the game before the program terminates.
2. Est. time to completion: 10 – 15 min.
3. There are approximately 80—90 total moves in the game if everything is used once and, in some cases, more than once. (i.e. use 'talk' twice). Hence, to make the game more challenging I chose 65 moves as the starting time limit.
4. If a player does not finish in 65 min. (i.e. 65 moves) a message will display that the game is over and you failed the gang, so they're all dead.

## Gameplay Solutions Manual

1. Begin game with Time Remaining showing game clock (deduct 1 min/action)
2. Use 'look' to check orientation and acquire the knife
    a. Acquires the knife to cut all of the Loser Gang members free
    b. Current location: Sewer Main Entrance
        i. Look 1x
        ii. Talk 1x
        iii. Read 0x
        iv. Rescue 0x
3. Use 'move' and 'right' to enter Sewer Tunnel 1
    a. Current location: Sewer Tunnel 1
        i. Look 1x
        ii. Talk 1x
        iii. Read 0x
        iv. Rescue 0x
4. Use 'move' and 'forward' to get to Loser Gang member 1 ("Billy")
    a. Current location: Next to Billy
        i. Look 1x,
        ii. Talk 1x,
        iii. Read 0x,
        iv. Rescue 1x
            1. Acquires the flashlight to proceed further into sewer tunnels
5. Use 'move' and 'back' to get back to Sewer Tunnel 1 and then use 'move' and 'left' to return to the Sewer Main Entrance
6. Use 'move' and 'forward' to enter the Main Sewer Tunnel
    a. Current location: Main Sewer Tunnel
        i. Look 1x,
        ii. Talk 1x,
        iii. Read 1x,
            1. Reads the pamphlet on the wall and acquires it to conquer fear
        iv. Rescue 0x

7. Use 'move' and 'right' to enter Sewer Tunnel 2
    a. Current location: Sewer Tunnel 2
        i. Look 1x
        ii. Talk 1x
        iii. Read 0x
        iv. Rescue 0x
8. Use 'move' and 'forward' to get to Loser Gang member 2 ("Ben")
    a. Current location: Next to Ben and Billy
        i. Look 1x
        ii. Talk 1x
        iii. Read 0x
        iv. Rescue 1x
            1. Acquires the lighter needed for extra light in Deep Sewer
9. Use 'move' and 'back' to get back to Sewer Tunnel 2 and then use 'move' and 'left' to return to the Main Sewer Tunnel
10. From Main Sewer Tunnel use 'move' and 'left' to enter Sewer Tunnel 3
    a. Current location: Sewer Tunnel 3
        i. Look 1x
        ii. Talk 1x
        iii. Read 0x
        iv. Rescue 0x
11. Use 'move' and 'forward' to get to Loser Gang member 3 ("Eddie")
    a. Current location: Next to Ben, Billy, and Eddie
        i. Look 1x
        ii. Talk 1x
        iii. Read 0x
        iv. Rescue 1x
            1. Acquires the inhaler because Eddie probably needs it
                a. Note: Inhaler was used in battle of original 'IT' movie
12. Use 'move' and 'back' to get back to Sewer Tunnel 3 and then use 'move' and 'right' to return to the Main Sewer Tunnel
    a. Current location: Main Sewer Tunnel
13. Use 'move' and 'forward' to enter the Deep Sewer Tunnel
    a. Current location: Deep Sewer Tunnel
        i. Look 1x
        ii. Talk 1x
        iii. Read 0x
        iv. Rescue 0x
14. Use 'move' and 'right' to enter Sewer Tunnel 4
    a. Current location: Sewer Tunnel 4
        i. Look 1x
        ii. Talk 1x
        iii. Read 0x
        iv. Rescue 0x
15. Use 'move' and 'forward' to get to Loser Gang member 4 ("Richie")

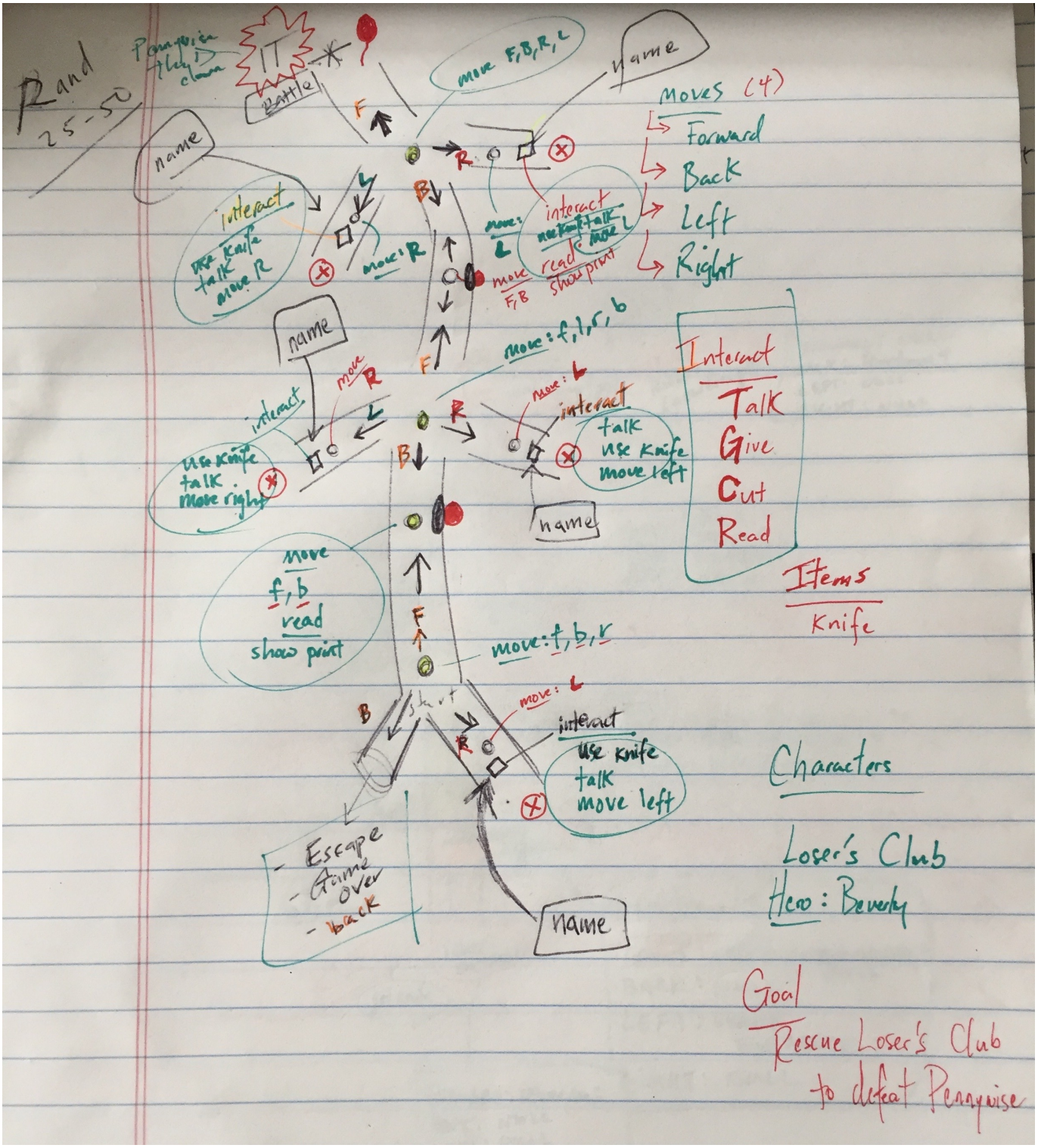
a. Current location: Next to Ben, Billy, Eddie, and Richie
    i. Look 1x
    ii. Talk 1x
    iii. Read 0x
    iv. Rescue 1x
        1. Acquires the slingshot needed to defeat IT

16. Use 'move' and 'back' to get back to Sewer Tunnel 4 and then use 'move' and 'left' to return to the Deep Sewer Tunnel
    a. Current location: Deep Sewer Tunnel

17. From Deep Sewer Tunnel use 'move' and 'left' to enter Sewer Tunnel 5
    a. Current location: Sewer Tunnel 5
        i. Look 1x
        ii. Talk 1x
        iii. Read 0x
        iv. Rescue 0x
18. Use 'move' and 'forward' to get to Loser Gang member 5 ("Beverly")
    a. Current location: Next to Ben, Billy, Eddie, Richie, and Beverly
        i. Look 1x
        ii. Talk 1x
        iii. Read 0x
        iv. Rescue 1x
            1. Acquires the silver rock to use in slingshot
19. Use 'move' and 'back' to get back to Sewer Tunnel 5 and then use 'move' and 'right' to return to the Deep Sewer Tunnel
    a. Current location: Deep Sewer Tunnel
    b. Everyone is rescued
    c. All items acquired:
        i. Knife
        ii. Flashlight
        iii. Lighter
        iv. Inhaler
        v. Slingshot
        vi. Silver rock
20. Use 'move' and 'forward' to enter the Pennywise Lair and faceoff against IT
    a. Current location: Pennywise Lair
        i. Look 1x
        ii. Talk 1x
        iii. Read 1x
        iv. Rescue 0x
        v. Battle 1x
    b. Battle IT
    c. Auto-Ddefeat IT
    d. **GAME OVER – You win!**

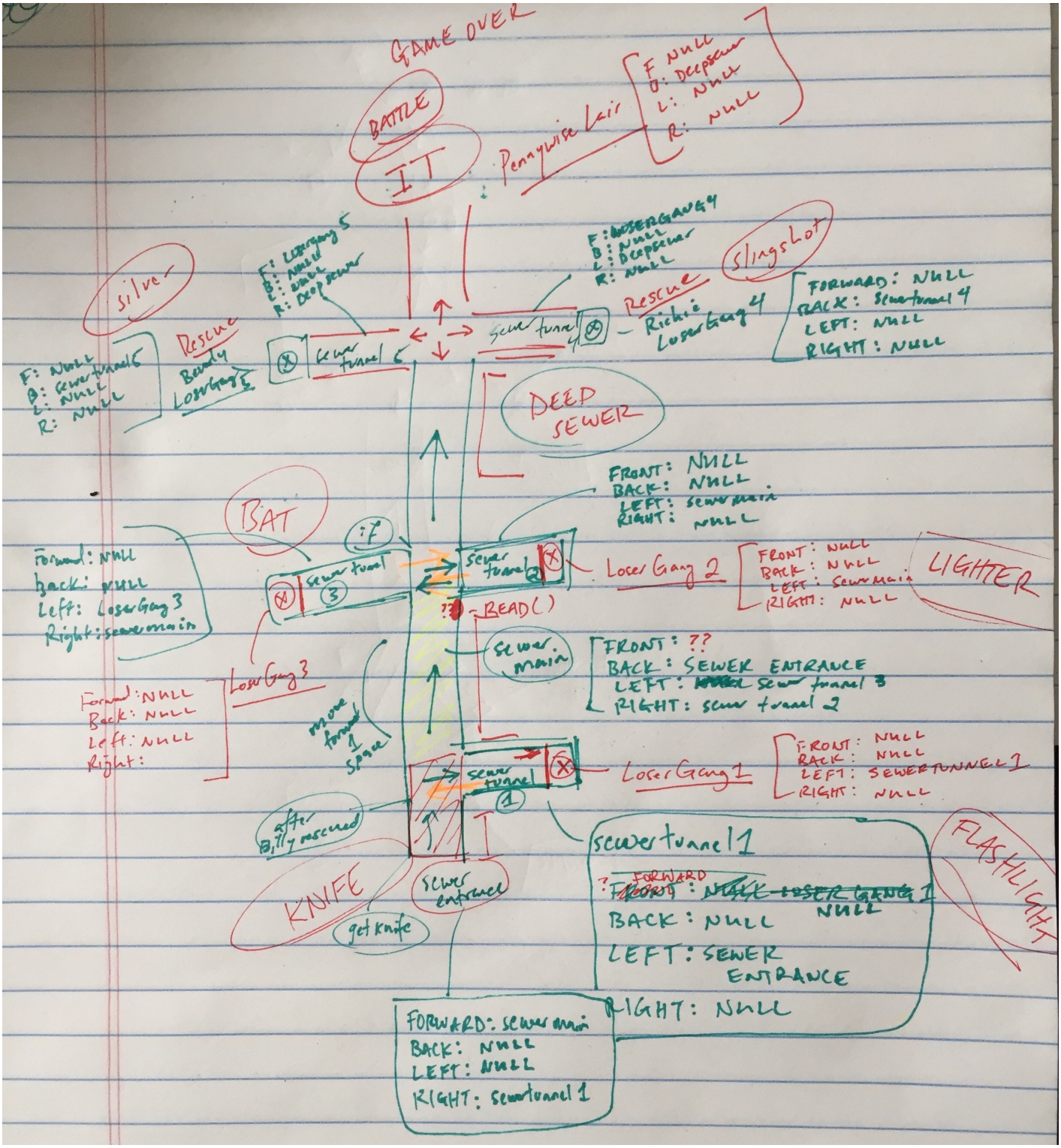
Travis Laxson
CS162 – Final Project

# Initial Project Design



Travis Laxson
CS162 – Final Project

# Edited Project Design



Travis Laxson
CS162 – Final Project

# Class Hierarchy Diagram