Corey Savage
6/3/16
CS 162
Final

Design Plan:

Goal: Inmate escapes prison

Areas:

Starting Cell
- Directions:
  - North - Cell Hallway (locked) - unlocked with cell key
  - South - Outside (locked) - dig out - need shovel x2 and other prisoner (chance 50%)
  - East - "Other Cell"
  - West - "Other Cell"
- Items:
  - Mattress
  - Cell Key - unlocks cell (hidden in other item)
  - Toilet
  - Sink

Cell Hallway
- Directions:
  - North - "Other cell" (locked) - unlocked with cell key - chance to help/kill
  - South - Starting Cell
  - East - Hallway1
  - West - Hallway2
- Items:
  - Clipboard

Hallway1
- Directions:
  - North - Guard Break Room - chance to get caught/kill them (gain uniform)
  - South - Utility Closet (locked) - unlocked with master key
  - East - Laundry
  - West - Cell Hallway
- Items:
  - Light switch

Hallway2

Directions:

North - Kitchen

South - Outside - Chance to escape (10%). With guard uniform (+80). With guns (+50). With other inmate (-50)

East - Cell Hallway

West - “Other”

Items:

Map

Guard Break Room

Special: Upon entering Chance to overtake guard by self (10%). With other inmate (+20%). With guns (+60%). With shovel (+10) With Knife (+20)

Directions:

North - Armory

South - Hallway1

East - Laundry Room

West - Kitchen

Items:

Guard uniform

Utility Closet

Directions:

North - Hallway1

South - “Other”

East - Laundry

West - “Other”

Items:

Master Key

Shovel

Laundry

Directions:

North - Guard Break Room

South - Utility Closet

East - “Other“ Laundry Basket - Chance to escape (40%). With other inmate (-30%)

West - Hallway1

Items:

Prisoner uniforms

Kitchen

Directions:

North - Armory (locked) - unlocked with master key
South - Hallway2
East - Guard Break Room
West - “Other”

Items:

Knife

Armory

Directions:

North - “Other”
South - “Other:
East - Guard Break Room
West - Kitchen

Items:

Gun
Shovel

Script:

Intro Description:
Welcome to Prison Escape!
With Prison Escape there are multiple ways to escape, but there are also multiple ways to get caught. As with real prison escapes, there is no guaranteed method of success. However, you actions greatly dictate your likelihood of success or failure. Proceed with caution.

Starting Cell:
Ugh I can’t it any longer! Today is the day I get out of this place.
All I have to do, is just get out of this cell…
and then… umm… I’m sure I’ll figure it out.
I know this isn’t the most fleshed out plan, but I’ve figured one thing out.
I know this isn’t the most fleshed out plan, but I’ve figured one thing out.
But with all the toilet wine from the night before I don’t remember where I hid it.

Test Plan:

The final assignment will require a large amount of testing to ensure the program is running functionally. Prison Escape features 11 unique areas for the player to explore, as a result each space needs to be properly tested for a multitude of different functions. These 11 unique areas carried the bulk of the testing requirements as a result. Below I will list each aspect I plan to test for the program, I will break the task into which file the tasks will be performed in.

Main:
- Win requirements
- Begin game (y/n)

Items:
- Can add items
- Item modifiers work for leaving prison and entering guard break room
- Item names appear properly

Map:
- Blank inventory initializes
- Map set properly
- Inventory gets printed properly
- Can add Items to inventory
- Game runs until turns run out, or player is caught, or wins
- Rooms properly lock/unlock
- Guard Break Room event
- Exit event
- Other Cell event

Spaces:
- Can move between spaces
- Cant move to NULL pointers
- Cant move into locked rooms

Specific Areas:
- Special works for each

Style:
- Consistent spacing

Test Results:

Main:

- Win requirements ☑
- Begin game (y/n) ☑

Items:

- Can add items ☑
- Item modifiers work for leaving prison and entering guard break room ☑
- Item names appear properly ☑

Map:

- Blank inventory initializes ☑
- Map set properly ☑
- Inventory gets printed properly ☑
- Can add Items to inventory ☑
- Game runs until turns run out, or player is caught, or wins ☑
- Rooms properly lock/unlock ☑
- Guard Break Room event ☑
- Exit event ☑
- Other Cell event ☑

Spaces:

- Can move between spaces ☑
- Cant move to NULL pointers ☑
- Cant move into locked rooms ☑

Specific Areas:

- Special works for each ☑

Style:

- Consistent spacing ☑

Testing Reflections:

While testing was tedious, it went very smoothly. There was only one major problem that required extensive problem solving. At the beginning of the programming process, after I finished implementing movement through spaces, I began working on calling each area's special() function. While testing the special() functions my program would result in a segmentation fault error, and I could not find the cause. After hours of problem solving I went to a help session hosted by a TA for assistance. After the help session we could still not figure out why the program was resulting in a segmentation fault.

I eventually realized the segmentation fault was a result of me initializing the Spaces objects in my Map class constructor, as opposed to properly being initialized as a class data member. The segmentation fault error was by far the largest obstacle I faced during testing and resulted in hours of undesired debugging.

Reflections:

With regards to design, there were many changes made throughout the entire programming process. The largest design change that occurred was the redesigning of my linked list to classes that held the pointers. Originally I planned to have a struct container that would be manipulated to navigate the room.Besides, the Spaces redesign there were numerous other design changes that occurred throughout the process. Items were removed such as the clipboard,

light switch, prisoner uniforms, and one of the shovels. Additionally, two of the potential exits were removed from the game. The reasons for these design changes were a result of me trying to simplify the game. The removed items and removed exits served to misdirect the player, I originally wanted the player to pursue incorrect paths until they can determine the correct path to escape. However, I believe this resulted in a too difficult game, which required the player to restart too many times. I left a few aspects of misdirection in the game, like the shovel and other prisoner, but I removed the Items and exits that served very little purpose.

I also added a few design elements into the game from the original design plan. These elements mostly consisted of new areas, such as an actual Exit space and OtherCell. Originally, I planned for these spaces to be just containers for the relevant adjacent space. However, with the pointer design and all the space modifiers I wished to implement it seemed easier to create these areas as their own unique Spaces.

The final was a great learning experience. Besides improving my programming abilities through practice, the main lesson I learned from the assignment was to avoid over designing. Feature creep can occur when a designer is too ambitious with what they want to program to do and include more

than what is needed. In the future I will be sure to design the necessary components first, then after that has been tested, begin adding additional features.