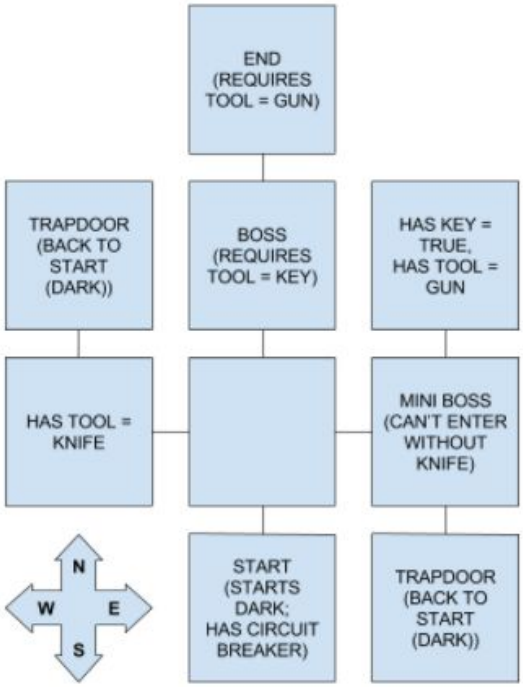
Kenny Ngo
CS 162
06/13/17
Final Project

Area Design



I wanted to design a small version of a dungeon from a game like Legend of Zelda, where the main character is running around in different rooms/chambers looking for keys and weapons in order to defeat the boss and get through to the end of the dungeon. This took several revisions. When I initially started out trying to develop a map to use for this game I used a 5x5 grid to represent the playing area, then condensed it way down to nearly the bare minimum that I needed and ended up with 9 separate rooms. The rooms will be accessed by "doors" that contain pointers to the correct room using N, W, S, E directions to specify which is which.

To meet project specifications, the dungeon starts in the dark and requires the player to turn on the lights

Character Design

Player - needs pointer to current room
Has small bag that can hold 2 weapons
Bool key is initially set to false
Health starts at 4, goes down by one point each time you fall through trapdoor or approach mini boss without knife or approach boss without gun

This design took me awhile to come up with as I didn't know how to incorporate the current space of the player into the design on the player class itself. I turns out all I needed was for a pointer to the current space the player is in, similar to a queue class with the current node being pointed towards.

I also wanted a way for the player class to interact with the space, so I gave the player a few booleans such as key and light, indicating whether the player had unlocked these things or not.

Reflection & Testing

Unfortunately, my program had failed all my tests and try as I might, I was not successful in debugging them in time for the final project submission deadline. I attribute this to the unforeseen complexity of the reliance of the player class on the various room classes. I was not able to work out the relationships between the classes clearly enough before beginning and ran into large snags in the debugging phase of my project.

Testing this program was especially difficult because each testing step required that the previous step have been passed successfully. This is unfortunate because if a certain aspect of a program does not work early in the beginning, then it can not be tested further than that point.

Another difficult part that I did not have time to fully implement was the program timer. I understand that I need to use the clock() object to measure time, but the while loop doesn't seem to work since the gameMenu() function is waiting for input.

Luckily I was doing a form of white box testing, meaning I know the inner workings of the product and can test the program using the optimum parameters.

My testing plan would've been as follows if I had been able to get it working correctly:

Rooms with only 1 door: Test 'y' vs 'n' inputs as well as random other inputs like '1' or 'string'

Rooms with 2 or 3 doors: Test a door that I know is 'NULL' and see what the program does as well as test the correct 'N', 'S', 'W', 'E' directions that point to actual doors.

Mid room: Test boss door with and without key (this part of my program failed catastrophically and wouldn't allow me to progress so I had to comment out the key check portion and assume that the player had indeed received the key)

Trap rooms: ensure that the trap hurt the player, also ensure that the player died after four trap room visits

Mini boss room: visit room with and without knife, testing the pushing back of the miniboss when the player is without knife

Boss room: visit room with and without gun, testing the pushing back of the miniboss when the player is without gun