Zeyad Shureih
8/14/2018

Final Project Reflection

Initial Design:

I didn't have one, and this might have been my biggest downfall. I approached the Final Project as a display of every C++ skill I had acquired over CS161 and CS162. With this approach, I started with making a board, characters, combat, a crude form of collision detection, and win parameters. I mention several times in my code (or at least I meant to) where I seemed to just throw something together and it is obvious that it wasn't planned out at all. I feel like the only skill that I learned that I didn't include is recursive functions, though I do know where I could have implemented one and it is marked in my code.

Because I lacked a design, I ran into many errors that could have easily been avoided. Stuff like deleting dynamically allocated data in the wrong spot and declaring objects redundantly.

Final Design:

My final product is still pretty messy, but at least it's a working program without any memory leaks, except for if you issue an interrupt, then it leaks. I had no idea how to even begin fixing this.

I elected to create the Game class which was a culmination of my usual Menu and Board functions. I did this so my main function would remain clean and more modular, as well as allowing for easier interactions between objects from different classes like Characters, Spaces and Items.

I decided to go with a dungeon crawler-esque game as my final project. The player moves around, picking up items, fighting enemies, and then a boss, finally using the key the boss drops to open the gate to the end of the level. Hint: The easiest way to win is to use two potions on the 4th enemy so you go into the boss fight with 8 hp, and then use your last potion when you reach 2hp. Occasionally one of the base enemies will drop a potion, but only if the player's bag is not full, so it is better to hold on to them for when they are needed.

I went with turn based combat because I felt like it would be easier for the player to process. I experimented with live combat (moving on to an enemy would deal damage to them and not lock you into combat) but because I made it so enemies will always move towards a player if they are adjacent to them, I found it created pseudo turn based combat. I decided to just turn combat into a different phase of the game with its own menu in order to further immerse the player with a sort of *standoff* based tone when in combat.

Debugging:

By far, my biggest issue with this final project came from the board. Because the Game class contained the board, along with every other important function in the program, I often ran into issues involving moving characters on the board and segmentation faults when deleting the objects on the board. It took me far too long to realize that if everything (bar potions) was being initialized onto the board, then I would only need to delete object if the player would no longer be able to interact with it. I only realized this after I started spitting out the address of every single object as it was allocated/deleted in order to figure out what exactly was causing a segmentation fault, when exactly it occurred, and where the object was supposed to be.

This debugging practice helped immensely during the final stages of development when I was trying to clear up the last of my memory leaks. I was testing the base game, just initializing and deleting a Game object, and had 33 blocks of memory still allocated. I had this error for 3 days before I decided to count all the memory addresses being allocated and deleted. Obviously I had 33 extra blocks being allocated, but what objects were they and where were they being allocated. I eventually narrowed it down to Wall objects, and found that I was creating walls under walls I had already initialized. After fixing it I made a note of it in my code, so I could always remind myself to check the board for memory leaks before anything else.

Test Table:

Test Case	Input Values	Driver Functions	Expected Outcomes	Actual Outcomes
Move	WAS D	move	Player moves up/down/left/right	Player moves up/down/left/right
Open Chest	-	getTreasure	Player picks up x-saber	Player picks up x-saber
Attack	-	Move Battle displayBattleMenu attack	Begin battle after player move, player attacks first	Begin battle after player move, player attacks first
Get Attacked	-	Move enemies	Begin battle after	Begin battle after

		Battle displayBattleMenu attack	enemies move, player attacks first	enemies move, player attacks first
Use Potion	-	use	Heal 5 hp	Heal 5 hp
Run	-	move	Ends battle	Ends battle
Spawn boss	-	If statement followed by spawnBoss	Boss spawns where player spawned	Boss spawns where player spawned
Boss battle	-	bossBattle	After battle, drop key	After battle key drops
Open gate	No key	move	Error message	Error message
Open gate	key	move	Gate is replaced with floor	Gate is replaced with floor
Enter flag	-	move	End game, play gain?	End game, play again?
death	-	battle	End game, play gain?	End game, play again?