Long Le
CS 162
Professor Zhang
Final Project - Reflections

Game - Isa's Adventures:

This is a text-based RPG game with a turn-taking battle style. The user will assume the role of Isa, who is a young female warrior from a nearby village. A princess has been captured by Billy the Bandit and his gang. Your goal is to defeat all opponents and reach the princess to rescue her.

--Game Specs--

Spaces:

- The game environment exists on a 15x15 grid of pointers to spaces.
- Only ground spaces can be walked on by characters. Rock spaces can only be interacted with.
- Trees and water spaces have a ⅛ chance of spawning with an item. Trees might have apples, which restore up to 2 hit points. Water might have medicinal herbs, which restore up to 4 hit points.
- Campfire spaces can warm the character up and prevent hypothermia.
- Door spaces are locked and can be unlocked with a key.

Mobs:

- Built with a very basic AI. The mob will follow the player and attack. The mobs dance (move back and forth) often when lined up with the character and there is an obstacle in the way.
- Certain mobs are given items that can be looted by the player after the mob dies (and the battle has ended).

Win/Lose Conditions:

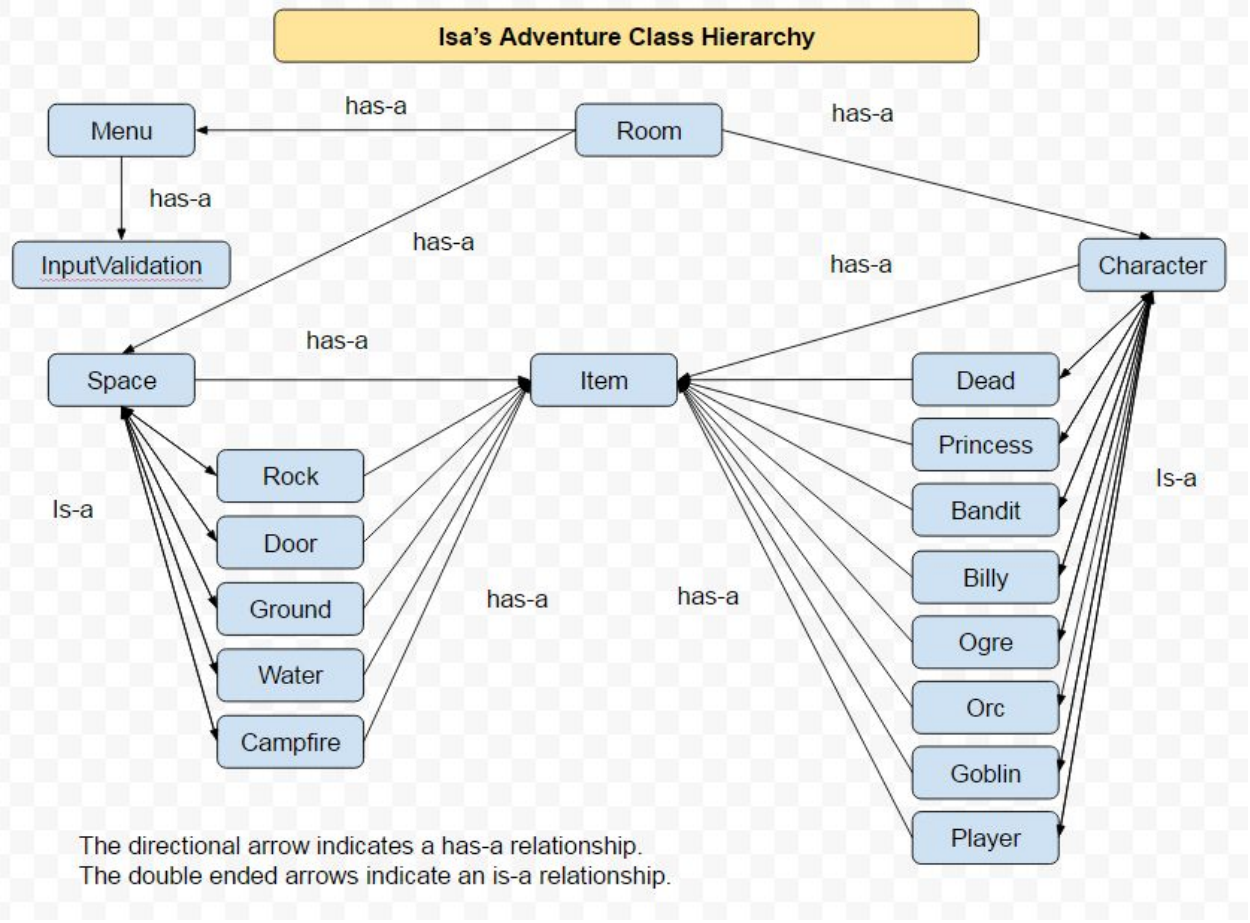
- There is a time limit to rescuing the princess. A counter will continue to count down after every player or enemy movement.
- There is a survival element as Isa needs to utilize campfires to warm up, or find a travel cloak somewhere in the game. If the cold counter reaches 0 the player will die and the game will be over.
- To win, you have to defeat all the enemies, find the keys, and reach the princess locked in the dungeon.

--Design--

I will not list every class as I have done in the past reflections. The first reason is because I did not design the game before I started programming. Previously, I have always designed the program prior to programming, but this time I decided to design the game features on-the-fly.

The reason why I chose this method of programming/designing because I didn't know exactly what my design strategy was or how it will change during programming. This is due to my lack of experience in designing any large projects. I knew that my design would change too many times and since my time is very limited for this project, I felt it would be more beneficial to start coding right away and make decisions as I go.

The second reason I did not list the classes in this document is that I have too many classes to list for this document. I also feel that it is very redundant to list the final class variables/functions that are already listed in the source code header files. However, I have created a class relationship diagram to view the below.



Programming Strategy:

From the start I planned on using what I learned from the following class projects to create this game:

1. Langton's Ant - Creating a 2D array, tracking coordinates of objects, printing the display. Credit to Harlan for providing the terminal clear screen command, a heads up for the delay timer function (usleep), and for the idea of using <, >, ^, v for the graphics.
2. Grocery List - Creating a good menu and managing containers.
3. OSU info system - Working with dynamically allocated objects and memory management.
4. Creature Tournament - Polymorphism and inheritance. Creating character object stats. Attacking and defending.

I used the following strategies in an attempt to prevent bugs in the finished program:

1. I compiled and tested (if possible) after every function.
2. I ran valgrind every time I dynamically allocated memory.

Final Design Choices

1. In-Game Menu - I made the menu receive letter inputs as opposed to numbers. This allows a user to choose 'l' for left and 'r' for right as opposed to 1 or 2, which intuitively does not make sense. I kept this consistent for every portion of the in-game menu but not the main menu.
2. Backpack - I make this container different than others. The in-game menu allows you to type the first letter of an item and the first item to select a choice, so this backpack searches through the container and selects the first item that matches the first letter. I only have two user selectable items that will function: apple and medicinal herb. Since it doesn't matter which apple or medicinal herb it uses, the first one the container finds to use is fine. The looting to backpack and deleting items from backpack works the same way.
3. Battle - Every character in the battle has a move counter. When the counter reaches zero their turn ends. This simulates taking turns. Every character can skip their turn. The player can choose to end term with the choice 'e'. A mob will skip their turn if they are surrounded and cannot move.
4. Hypothermia - I just decided that it would be kind of fun to have a survival feature in the game. I added a second counter that represents hypothermia. The player can find a travel cloak to keep warm or use campfires to warm up.

5. Interacting with spaces - I made trees have lootable apples and water have lootable medicinal herbs. The campfire warms up the player and the door unlocks with a key.
6. Interacting with characters - I made dead bodies be lootable. Interacting with the princess will end the game.
7. There are two input grabbers that violate the requirements, but for good reason. The first one is that you have to type ‘toss’ in order to get to delete items mode. This is so that the user doesn’t accidentally delete any items. The second one is that in the in-game menu -> submenu, you have to type ‘quit’ in order to quit the game. Since there is no save feature, quitting the game means starting over again. These design choices are just for precaution.
8. I chose to use external .txt files to set the locations of the spaces and the mobs. This made it easy to design the rooms. I realized that it will be easier to create a custom game editor to edit and design levels instead of hard coding every game.

--Issues--

I ran into so many issues that I can’t remember them all. I will just list the problems that were most difficult for me. The primary problems were memory leaks, seg faults, and the fact that there were so many lines of code it was hard for me to keep track of what everything was, and how they linked together. I have never used gdb before this program, but I quickly realized the power and need for a debugger after reaching 1000 lines of code in the room.cpp source file.

There was no way I can trace all of the function calls and links. I learned how to set a breakpoint at certain points which automatically stop and allow me to step through the program. This is such an upgrade to so many couts! I also learned that gdb can tell me where seg faults occur which was extremely helpful.

1. One memorable issue was trying to create the loop for the main menu, so I can start a game over and over. To accomplish this, I deleted the rooms made in heap and made new again after a game ending event occurs, or the user chooses to quit. This was a simple mistake that took hours to figure out. I did Room* begin = new etc…, then Room* begin = new etc... . Valgrind was giving me a message saying I was trying to access inaccessible memory. My mistake was copying and pasting. I have run into so many issues by copying and pasting, I should not do that anymore. I finally spotted the mistake and deleted the Room* part where I tried to make new a second time.
2. Another memorable issue was that after a door space unlocks, the movement in the player became very odd. I finally found that after the door space was deleted and replaced with a ground space (unlocking), the space did not reconnect with other spaces. I fixed this by running my connect spaces function again after a door unlocks. This fixed the issue, but took a long time.

3. The hardest part of my game was my basic enemy AI. It took me a very long time to create, test, and fix. The end result is that the enemy functions right(ish). The mob will still move back and forth during certain space alignments with the character.

--Final Testing Results--

To save time, I will not try to identify the driving functions. I did most of the testing during the coding process, which was extensive, so I will just list what I tested after. I still don't know if I covered everything. There could very well be bugs.

Test	**Where**	**Actual Outcome**
Choose (1) Start Game	Main Menu	Start Game
Choose (2) Credits	Main Menu	Shows Credits
Choose (3) Exit	Main Menu	Exits Game
Choose any choice outside of those three. (I've tested my input validation function and menu extensively)	Main Menu	Shows correct validation prompt.
Player attempts to move out of room in battle.	Any rooms.	Prompts user that they cannot leave during battle.
Player interacts with space.	Any rooms.	Searches grounds, skips turn if in battle.
Player interacts with Tree.	Rooms with trees.	Player receives apple approx. ⅕ of the time. Skips turn in battle.
Player interacts with Water.	Rooms with water.	Player receives medicinal herb approx. ⅕ of the time. Skips turn in battle.
Player interacts with Rocks.	Any rooms.	Prompts a description.
Player interacts with mob when mob is alive.	Most rooms.	Attacks the mob. Damage calculation activates.
Player interacts with mob when mob is dead.	Most rooms.	Loots mob.
Player interacts with campfire.	Rooms with campfires.	Increases warmth by up to 10 pts per interaction.
Player interacts with door with no key.	Rooms with Doors	Shows a prompt.

Player interacts with door with key.	Rooms with Doors	Unlocks door. Replaces Door space with Ground.
Bandits Escape countdown to 0	Any rooms.	Game ends.
Hypothermia counts down to 0	Any rooms.	Game ends.
In game menu chooses any other option than the available choices	Any rooms.	Do nothing. Loops back for another input.
In game menu - Chooses a directional choice u, d, l , r.	Any rooms.	Player will face that direction.
In game menu - Chooses f	Any rooms	Player will move forward one space.
Sub menu choose t.	in game submenu	Enters travel pack
travel pack menu choose any letter besides first letters of items.	travel pack menu	Do nothing. Loops back for another input.
travel pack menu choose s.	travel pack menu	Displays player max stats.
travel pack menu choose toss.	travel pack menu	enters delete mode.
Delete mode choose first letter of an item	delete mode menu	deletes that specific item.
Sub menu choose t.	submenu	Prompts the user if they really want to quit.
quit menu choose y	quit question prompt	The game quits and loops to the outside menu.
quit menu choose n	quit question prompt	The game goes back to submenu.
Player hit points drop to 0	Any battle	player dies and game ends.
Mobs hit points drop to 0	Any battle	mob is dead and displays an x. The mob does not move or attack. The mob becomes lootable.
Loot dead mob in battle	Any room in battle	Displays prompt but does not loot mob.
Loot dead mob not in battle	Any room not in battle	Loots mob
Player use item	Any room	item is removed and hp increase effect is activated.
Player deletes item	Any room	item is removed.

Player interacts with princess	dungeon	Game ends. Winning prompt displays
User chooses to start game again after a game ending event occurs. This event could be a losing condition or a winning condition.	Main menu	The game starts again with no problems. The user can choose to quit the game and no leaks will occur.