The Cabin Functionality

Contents

[2 Brief Storyline/Walkthrough 1](#_Toc27454164)

[2.1 Scene 1 – Rude Awakening 1](#_Toc27454165)

[2.2 Scene 2 – Confusion and the Saw 2](#_Toc27454166)

[2.3 Scene 3 – Hiding from The Creature 2](#_Toc27454167)

[2.4 Scene 4 – Escaping the cabin 2](#_Toc27454168)

[2.5 Scene 5 – Final Scene 3](#_Toc27454169)

[3 Coding Perspective 3](#_Toc27454170)

[3.1 Game.cpp 3](#_Toc27454171)

[3.2 Beginning/Ending Animation – beginScene() and finalAnimation() 3](#_Toc27454172)

[3.3 Intro Warning -introWarning() 3](#_Toc27454173)

[3.4 All Scenes – Scene1(), Scene2(), Scene3(), Scene4(), Scene4Skip(), Scene5() 4](#_Toc27454174)

[3.5 Inventory – inventory() 4](#_Toc27454175)

[3.6 All Options – Options1(), Options2(), Options3(), Options4() 4](#_Toc27454176)

[3.6.1 Array Searching – Search() 4](#_Toc27454177)

[3.7 Game Overs – gameOver1(), gameOver2(), gameOver3(), gameOver4() 5](#_Toc27454178)

# Brief Storyline/Walkthrough

My final project for C++ is a short horror game. Its in the style of choose-your-own-adventure games in that you have various choices. This also plays like old style text-based adventure games in that you have to type commands such as “Open door”, “Hide under bed”, etc. Currently, due to slight time constraints and being as this is my first major project with C++, I’d rather not bite off more than I can chew. The user can check their current items by typing in “Inventory” at any point. The game contains 5 scenes, with various choices to complete each task, we’ll discuss each one briefly now:

## Scene 1 – Rude Awakening

You awaken on a straw bed with a sore head. You’re unsure of how you got there. You hear the wind outside of the cabin and you’re left to type out your next decision.

* Choosing to go back to sleep – This results in your death as shortly after you fall asleep, the creature shows up.
* Choosing to stand up – This ends up with you falling to the floor and hurting your ankle, but you come to realize you’re cuffed to the bed. The next scene begins after this.

## Scene 2 – Confusion and the Saw

You fall to the floor and feel a pain in your ankle. After a short while looking around, you find a hacksaw. Checking your inventory reveals that its very good for sawing through iron. However, being that it’s a horror game, many may think of the worst option with the saw. You’re left to type out your next decision.

* Sawing your foot – This results in your death as, well, you sawed through your own foot. This is mostly playing off the old survival cliché in order to sort of trick the user into an unnecessary death.
* Sawing off the chains – This results in a transition to the next scene.

## Scene 3 – Hiding from The Creature

You come to realize that while your legs in pain, you’re able to walk perfectly fine. You get up and walk around, noticing 4 large furniture pieces/potential hiding places. You walk to the door to check outside of the cabin but the doorknob starts to open as you reach to it. You’re now left to decide where to hide.

* Hiding under the pelts/Under the bed – The bed and pelts are described as “straw bed” and “ragged pile of old furs”, both being negative descriptions. These result in the death of the player as the creature finds the player if they hide there.
* Hiding in the Cabinet/Behind the door – Both are described in positive ways, such as “Large door” and “sturdy empty wardrobe”, both result in you surviving the creature coming into the cabin. You progress onto the next scene.

## Scene 4 – Escaping the cabin

After the creature leaves and you wait a few minutes, you also leave the cabin. Once you’ve left a short way out of the building, you come upon a path surrounded by woods. You’re now given control again.

* Heading back – The player refuses to go back in the cabin out of fear. You’re brought back to the current decision.
* Heading down the path – This results in the player’s death, as, sort of obviously, the creature headed down the path that left from the cabin as it’s the most likely path to take.
* Heading into the woods – Shortly after heading into the woods, you hear the footsteps of the creature heading down the path, indicating that you made the right choice. The player is given the impression that they’re still at risk for the sake of suspense. You then go onto the final scene that no longer contains options.

## Scene 5 – Final Scene

Once you enter the woods, you’re of course very cold, but continue out of fear of the creature. You come across a destroyed tent that happened to be your own. You also come across a map that you had that has directions to the closest lodge. You walk throughout the night and get to the lodge. You notify the police, and while in the safety of the lodge, the police leave to investigate the attack.

You’re left with a short snow animation of the cabin itself.

# Coding Perspective

I’m going to give a deep description on how things run in terms of coding, I’ll be going down the list in terms of start to finish, just as I had with the scenes. However, once I’ve explained the main coding functions, I’ll stop as a lot of it is simply repeated towards the end or just reimplemented for the story. Everything is well commented on, so I believe it’d be easy enough to follow. In the code, it runs in order of each type of function, animations are grouped together, scenes are grouped together, etc. But I’ll be running through how the code runs in order of their execution.

## Game.cpp

This is the main function and the only part that is needed to be compiled to run the game. It contains a library that links into most of the game’s functions themselves. The library actually also has a small library for itself as well for small code snippets that are used very often like the ascii art, clear function to clear out any potential issues with the cin buffer, and a segment of code used to clear the screen and add the border to keep things a bit prettier and easier to look at.

## Beginning/Ending Animation – beginScene() and finalAnimation()

This works by creating a sort of matrix. It runs a for loop within a for loop, the first for loop sets an integer into a random number within the line, then this runs down for 10 lines. The second for loop then asks if the random integer for that line is met, if not, it adds a space, if it does, it adds a twinkle (\*). It then ends the line for the next loop to go down a line rather than continuously off into space.

A title card is shown then the same matrix as described above is re-shown. At the end of this matrix, it sleeps for 1 second then clears the screen.

This whole thing is also nestled within yet another for loop that sets it to do this code 5 times.

The final animation works the same way, except that it changes the final text and adds an ascii art of the cabin at the bottom each time it loops.

## Intro Warning -introWarning()

This simply notifies the player that the game will only successfully run on windows machines due to it requiring access to “cls”. I suppose it can run on windows, but it would be extremely ugly looking. This also gets the user to configure their console screen in order to fit the two borders into their window. It also warns the user of potentially gore-y segments. Finally, this introduces the player to the idea that they must type commands in order to continue the game, via typing “continue” into the console.

## All Scenes – Scene1(), Scene2(), Scene3(), Scene4(), Scene4Skip(), Scene5()

Each Scene, apart from Scene4, functions by clearing the cin buffer, clearing the screen and setting the border, then giving the story details. After its given the story details, it sets a specific number depending on what scene its at in order for the player to be able to see what is currently in their inventory at later segments. It then gets the line for the player’s input choice and runs to the options segment of the corresponding number (Scene1 -> Options1).

Scene 4 is slightly different in that it runs a cutscene where the player then goes into Scene4Skip() after the player has scene the cutscene. It does this in order to allow the game over restart function and the inventory function to skip the Scene4 cutscene.

## Inventory – inventory()

This will receive a selected number from the scene that the player is currently on, it then checks this number and displays what is in the player’s inventory at that current moment. This can be crucial for getting further details on items in the future. It then sends the player back to the corresponding scene.

## All Options – Options1(), Options2(), Options3(), Options4()

These all contain options that the player can use during the scenes of the corresponding number. It does this by first initializing several large arrays. Each command that could lead to a response is included within an array that details what response should be taken.

The player, during all of these segments can go to the inventory() section that has been described above. There are then “wrong” options which will lead the player into a game over scene and “correct” options that will lead the player into the next scene. Or, on rare occasions, will reply a retort and force the player to take an option that turns to one of the other two types of options.

### Array Searching – Search()

This segment took the longest because it’s the easiest and cleanest way of doing this, however, I went across many…many prototypes that simply didn’t work or would break. First, it takes the input of: Search(Array to check, Size of array to check, Choice of the player to check against the array). It then responds with a number. If the number is anything other than -1, it will be ignored and jump down to the next array.

Within the search it initially starts by setting a variable called “found” to false, it then starts a for loop that is the size of the array and it reads each section of the received array, then sets itself to check the next word of that array. If the Choice that it received at the beginning is matched by the word in the array, it changes “Found” to true. When its looped, it hits an if that tells the function to send a 1 if the found has been checked, and if it hasn’t been checked, it’ll send a -1, telling the if of the options function to go onto the next array.

## Game Overs – gameOver1(), gameOver2(), gameOver3(), gameOver4()

Each one of these starts by changing the scene, then displaying a text describing in what way you died. In 2 of these, there is a sort of cutscene that played. After the description is done, it clears the screen and displays the gameoverScene() animation. This is simply the beginScene() function but with a game over screen added. Once its displayed this 5 times, it will change the scene and ask you if you wish to retry from the scene from which you died.

Giving anything other than lowercase or uppercase y will result in the program killing itself. If a “y” is given, it sends the player back to the corresponding cutscene.