Clinton Hawkes
CS162-400
Final Project Design Document

Bunker Escape

Design:

My final project is a game where the player is trapped in an underground bunker and the only way for them to escape is to blow the sealed hatch open. There are six different rooms in the bunker and the parts needed to make an explosive can be found scattered around. After the player finds the required parts, they must be assembled by taking them to the workbench. Once the explosive is assembled, the player can take it to the hatch and use it to blow the door open. If the player is able to make the explosive and escape the bunker before using all the available steps, they win the game.

1) Main function – used to create the Game object and call the run() function to start the game.
    A) Create a bool variable “state” to hold the current state of the game (run or quit)
    B) Create a bool variable “win” to hold the result of the game
    C) Create int variable “round” to keep track of the player’s step count
    D) StartMenu() is called to ask user if they want to play or quit
        I. If the user wants to quit, false is returned and the program terminates
        II. If the user want to play the game
            a) The goal of the game is printed to the screen to give the user direction
    E) Map of the bunker with player’s position is printed to the screen with the count of remaining moves
        I. Player starts in the ladder room
    F) While loop is created to loop until the user wins or runs out of available moves
        I. Menu() is called to get user’s directive
            a) If player chooses to “1. Investigate”
                i. The overridden Investigation() function of the player’s current room location is called
                    1) Each room contains an object that the player can interact with
                        A) If user chooses to interact with the object
                            I. A unique menu is displayed and the user is given further direction
                                a) If player blows the hatch open
                                    1) state is set equal to false
                                    2) win is set to true
                        B) If user chooses not to interact with the object
                            I. Screen returns to previous menu
            b) If player chooses to “2. Move”
                i. The overridden move() function of the player’s current room location is called
                    1) Player is given a list of directions they can move
                        A) Player selects the direction they want to move
                            I. Player’s current location is updated
                            II. Player’s remaining move count is decreased by 1

III. Overridden showMap() function of the current room is called

a) Map of bunker with player's current position is printed

c) If player chooses to "3. View backpack contents"

i. Items that have been taken by the player are displayed

1) The backpack has three available slots for items to be stored

A) If the slots are not filled, "Slot is empty" will print

II. Number of moves remaining is checked

a) If the number of moves remaining is 0

i. State is set to false and while loop ends

G) If win is equal to true

I. A message congratulating the user is displayed and the game is won

H) If win is not equal to true

I. The game is lost and a losing message is printed

Test Plan for integerValidation()

Test Case	Input Value	Expected Result	Observed Result
Char	A, c, +, .	"Please enter a valid integer."	"Please enter a valid integer."
String	Four, .five, yes	"Please enter a valid integer."	"Please enter a valid integer."
Float	6.31, 07.0, 8.8	"Please enter a valid integer."	"Please enter a valid integer."
< Min (set at 1)	0, -1, -5	"Please enter a valid integer."	"Please enter a valid integer."
> Max (set at 3)	5, 9999	"Please enter a valid integer."	"Please enter a valid integer."
Lower limit (min=1)	1	Function accept and return value to main()	Function accepted 1 and returned 1 to main()
Upper limit (max=2)	2	Function accept and return value to main()	Function accepted 2 and returned 2 to main()

Test Plan for Bunker Escape

Test Case	Description	Expected Result
Ladder Room Investigate	1. Blow the hatch open 2. Nothing	1. If the user has built the bomb, the hatch is blown open the the game is won If the user has not built the bomb, a message stating the user has no explosive is printed 2. Returns to the previous menu
Tools Room Investigate	Tool box is against the wall 1. Open it	1. The tool box is opened. "Nothing is useful in here… " prints to the screen

	2. Nothing	2. Return to previous menu
Chemical Room Investigate	Chemical locker in the corner 1. Open it 2. Leave it closed	1. Player sees gunpowder. Has the option to take it or leave it. 2. Return to previous menu
Gunpowder	1. Take the gun powder	Gunpowder is moved to the player's backpack.
Gear Room Investigate	See a bundle of rope 1. Take some rope 2. Leave it	1. Rope is move to the player's backpack 2. Return to previous menu
Workbench Room Investigate	Stand in front of a workbench 1. Assemble the explosive 2. Nothing	1. If user has all three items required for a bomb, a bomb is stored in the backpack and all the other items are removed. If the user does not have all three items, a message stating they don't have them is printed. 2. Return to previous menu
Parts Room Investigate	Pile of parts and scrap metal 1. Rifle through the pile 2. Nothing. Looks dangerous	1. Find a pipe. Have the option to take or leave it. 2. Return to previous menu
Find Pipe	1. Take the pipe 2. Leave it	1. Pipe is put in the player's backpack 2. Return to previous menu
Ladder Room Move	1. Choose to move left 2. Choose to move down	1. Player location is moved to the tool room. The showMap function is called 2. Player location is moved to the parts room. The showMap function is called
Ladder Room showMap	Called when player moves to the Ladder room	Map of bunker should print with the stick figure in the Ladder room
Tools Room Move	1. Choose to move left 2. Choose to move down 3. Choose to move right	1. Player location is moved to the chemical room. The showMap function is called 2. Player location is moved to the workbench room. The showMap function is called 3. Player location is moved to the ladder room. The showMap function is called
Tools Room showMap	Called when player moves to the tools room	Map of the bunker is printed to the screen with a stick figure in the tools room
Chemical Room Move	1. Choose to move right 2. Choose to move down	1. Player location is moved to the tools room. The showMap function is called 2. Player location is moved to the gear room. The showMap function is called.
Chemical Room showMap	Called when player moves to the chemical room	Map of the bunker is printed to the screen with a stick figure in the chemical room

Gear Room Move	1. Choose to move up 2. Choose to move right	1. Player location is moved to the chemical room. The showMap function is called. 2. Player location is moved to the workbench room. The showMap function is called.
Gear Room showMap	Called when player moves to the gear room	Map of the bunker is printed to the screen with a stick figure in the gear room
Workbench Room Move	1. Choose to move left 2. Choose to move up 3. Choose to move right	1. Player location is moved to the gear room and the showMap function is called 2. Player location is moved to the tools room and the showMap function is called 3. Player location is moved to the parts room and the showMap function is called
Workbench Room showMap	Called when player moves to the workbench room	Map of the bunker is printed to the screen with a stick figure in the workbench room
Parts Room Move	1. Choose to move left 2. Choose to move up	1. Player location is moved to the workbench room and the showMap function is called 2. Player location is moved to the ladder room and the showMap function is called
Parts Room showMap	Called when player moves to the parts room	Map of the bunker is printed to the screen with a stick figure in the parts room
View Backpack Contents	Choose option 3 in menu	All the items that the player has chosen to pick up is shown. If the backpack does not have a total of 3 items, the empty slots show “slot is empty”
Start Menu	1. Select Play 2. Select Quit	1. Message stating the goal of the game with brief instructions is printed to the screen 2. Message telling user to “Please play again” is printed and the program terminates
Menu	1. Investigate 2. Move around 3. View backpack contents	1. The overridden investigate function of the room the player is currently in is called. Follow the test plan for this function above 2. The overridden move function for the room the player is currently in is called. Follow the test plan for this function above 3. The showBackPack function for the player is called. All items the player has taken in the game up to the point this function is called will be shown.

Player runs out of available moves	Player does not assemble the bomb and blow the hatch in 8 moves	The game ends and a message stating the player died of dehydration prints to the screen.
Player wins	Player gathers all parts, assembles explosive and blows the hatch in 8 or less moves	The game ends and a message stating the user has escaped and won the game is printed to the screen.
Memory Leak	1. Start and run program using valgrind. Choose to play Play game and win by escaping 2. Start and run program using valgrind Choose to play Run out of moves and lose 2. Start program and immediately quit.	There should be no memory leaks for any of these cases.

Reflection:

Issues:

I did not run into any issue while working on the final project. Usually I would run into issues on the previous projects because I would second guess what the specification were asking for. The final project pretty much let me do what I wanted and let me decide how I wanted to implement them. There were a few guidelines regarding what needed to be included in the project, but the rest of the details were left up to me. I typically hated the ambiguity in the earlier projects, but this one I enjoyed it.

After thinking about it a little longer, I did hit encounter one issue. The issue had nothing to do with the coding part. The issue was coming up with an idea for the game. I am not very creative, and I think I spent more time pondering the different possibilities than I spent actually coding. I was getting pretty stressed when nothing was coming to mind, but after I decided on the theme, it was smooth sailing.

Changes:

I had originally planned to make my bunker with nine unique rooms derived from space, but part way through coding the rooms I decided to make six rooms. I follow the Slack channel, and the prevailing advice on there regarding the final project was to just hit the requirements to make sure everything was included and working well. People that had already take the course said not to overdo it and make it more complicated than it had to be, so I didn't. If I want to go back and change the project after I turn it in to make it more immersive and fun, then I can do it without having to worry about the possibility of bugs popping up from unnecessary functionality.

I had originally planned to use a linked list for the player container, but I heard the instructor mention a vector could be used for the container. I jumped all over that because there is much less chance of a bug when I use vectors. I set the limit of three items for the vector, and the vector will not have anymore than that because of the way I add the items to the container. The vector made that portion of the project easy.

I changed the number of available moves the user can take before losing the game from 15 down to 8. When the number was 15, it was very difficult for the player to lose the game. They could wander around the map with no strategic thought about what moves to make and not die. After I changed the number to 8, the user may die the first time through, but it is almost guaranteed that they will win the next round if they paid attention to where the items were the first time through the game. If you (my TA) needs to be able to make more moves to fully test out the game, the Player variable moveCount can be changed to a higher number.