Text-Based Adventure Game Design Doc

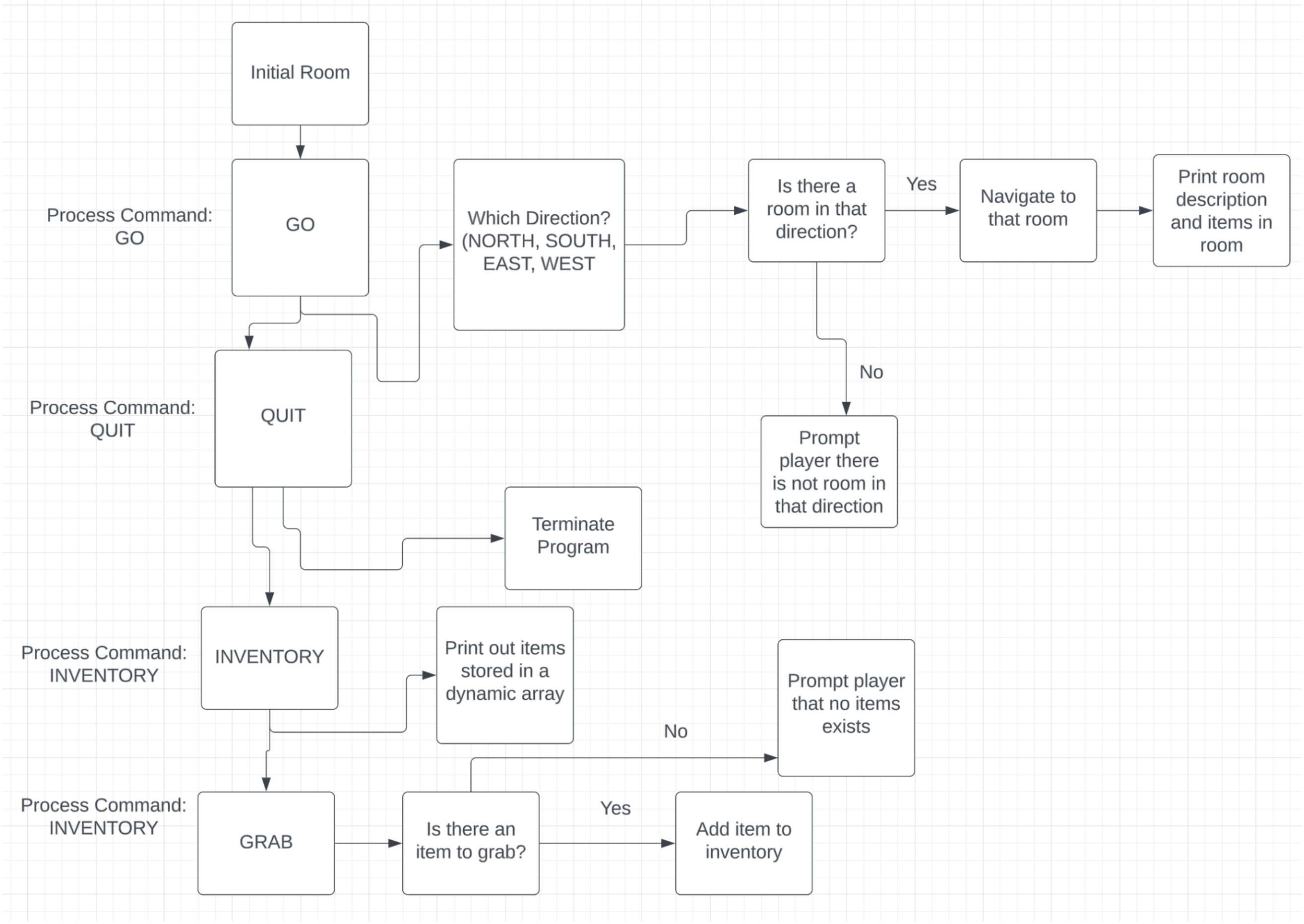
Author: Raed Kabir

Program Overview

This program is a text-based adventure game, where the player navigates through multiple rooms and is able to perform specific actions depending on the room they are in. The game acts as a large network of interconnected rooms with a few added random elements such as random items that will appear in each room. The game occurs in a dark cave; the story starts with the player going spelunking and becoming lost. Thus, the objective of the game is to find an exit. This implies there could be various solutions to beat the game because there will be multiple exits.

Picture

Below is a flowchart or block diagram of how the program will operate. The flowchart accounts for invalid inputs as well.


Initial Room
Process Command: GO
GO
Which Direction? (NORTH, SOUTH, EAST, WEST
Is there a room in that direction?
Yes
Navigate to that room
Print room description and items in room
No
Prompt player there is not room in that direction
Process Command: QUIT
QUIT
Terminate Program
Process Command: INVENTORY
INVENTORY
Print out items stored in a dynamic array
Prompt player that no items exists
No
Process Command: INVENTORY
GRAB
Is there an item to grab?
Yes
Add item to inventory

Inputs

The program will be governed by 4 commands: GO, INVENTORY, GRAB, and QUIT. Each command controls the game's rules. The command GO allows the player to navigate to other rooms. GO will be used the most, as it is the only command capable of allowing the player to explore. The command inventory will print out the randomized items placed in each room. This command is relatively simple. The command GRAB is a command that adds whatever item the user grabbed into their inventory. Lastly, the QUIT command terminates the program.

Outputs

The outputs of this program are largely room descriptions. For example, after a player uses the GO command, the program will print out a room description of the room the player just navigated to. This room description creates a sense of immersion and confirms to the player that they are moving; other outputs will be input checking. The input checking in this game simply prompts the user that their input was wrong. Then, the program will ask the user to input again without terminating the program.

Test Plan

Below is a test plan of the input and outputs of the program. Below demonstrates what a room description will look like.

INPUT

"GO North"

OUTPUT

"You are now in a large, wet room. The walls are covered with blood. You hear faint footsteps slowly approaching you. "

"There is a metal rod in the room. What will you do?"

"COMMANDS: GO, INVENTORY, GRAB, QUIT"

INPUT

"GRAB metal rod"

OUTPUT

"The metal rod has been added to your inventory"

Approach

The approach of my program will be outlined through pseudo-code. Below is the program's skeleton written through pseudo-code.

```
Declare libraries

Declare input string via char arrays


While the program is running {

        Print available commands;

        Prompt the user to enter a command;


        if (command equals GO) {

                Prompt user of the available directions they can navigate

                Ask which direction they want to go

                If (direction does not exist) {

                        Prompt the user that their input is invalid. Let them try again

                }

                else {

                        Navigate to that next room

                        Print out the room's description and available items in that room

                }

        }

        if (command equals INVENTORY) {

                For loop through an implementation of a vector in C {

                        For every element in the vector, print out the items
        }
    }
    If (command equals GRAB) {
        if  the current room is the room where the item should be in and the item exists {
            Add item to dynamic array i.e. their inventory
        }
        else {
            Prompt user that the item they want to grab does not exist.
        }
    }
    if (command equals QUIT) {
        Terminate prorgam using exit(0);
    }
}
```