User Script

Description

Since this product is a game with a high degree of randomness, a detailed script will be used to test that all user requirements are met. It will chiefly ensure satisfactory user experience through usage of the product.

Script

Steve is a new graduate from the Computer Science Diploma at Vancouver Island University and is on the hunt for his first software developer position. He is having some trouble reviewing all the material he has learned during his time at VIU, to prepare for the technical interviews he has lined up. He is searching for a fun interactive way to sharpen his problem solving skills, and review basic computer science principles. His friend has suggested he tries out Hacker's Labyrinth to help with the interview preparation.

He opens the application by running the executable for the game on his CSCI lab computer at VIU, and this immediately resizes his terminal screen and the title "Hacker's Labyrinth" appears on the screen in ASCII Art.

Then the opening menu pops up where Steve chooses the difficulty of the game, which he sets to easy and reads the game instructions. He then selects the start game option where he is told a story about being transported into a dungeon where his character must solve a multitude of puzzles within a given time parameter or face sudden death.

See below for a continuation of the scenario for each puzzle, the door selection, the in-game menu, and the endgame.

Door Selection

Case 1:
After completing a puzzle about encryption, Steve is shown a prompt asking him to pick either a left or right door to open. He enters 'L' to choose the left door and is prompted that he will be sent to the final level.

He presses a random key and then presses enter again to leave the door selection and is immediately given the prompt for the final level.

Case 2:
After completing a puzzle about encryption, Steve is shown a prompt asking him to pick either a left or right door to open. He does not read the instructions and enters a random key, the is told

the key he has entered is a invalid input and is prompted to enter again. This time he winters ‘R’ to choose the right door. Steve received no powerup so he presses a random key and then presses enter again to leave the door selection

In-game Menu

While inside the rising acid puzzle, Steve has some ideas that he wants to quickly jot down into a notepad. He enters the ingame menu by pressing the M key followed by enter. He is shown a large prompt which shows actions he can perform, such as leaving the game, returning to the game, and entering the notepad. He presses N followed by enter to go to the notepad, and is sent into a vim-like interface for entering text.

While in this interface, he enters a few ideas he had for combining blocks in his puzzle. He does this quickly since the game is being timed. Once finished, he exits by pressing colon “:”, “wq”, and finally presses enter to leave.

Steve returns to the game by pressing “Q”, followed by enter. He is then sent back to the rising acid game.

Rising Acid Puzzle

Steve is prompted about how acid will rise and fill the room he is in unless he stacks blocks on top of each other in pairs of equal height. This will create a wall of uniform height which will block the acid. He is shown, using ASCII art, what the blocks look like and what their heights are.

Case 1:
He first tries entering a height of 0 to be cheeky, and the game indicates that this was incorrect and that he now has 1 less try.

The smallest block has a height of 2 and the largest a height of 7, so he deduces that since every block must be stacked on top of another, these two must be put together. He enters 9 as his input and is given a prompt indicating that he was correct. He is also shown ASCII art of the wall he built.

Case 2:
Steve is confused about how to play the game. He enters the in-game menu and presses “H” to get help for this game. While reading these instructions, the allotted time limit runs out and he is shown a game over screen.

Case 3:
Steve is confused about how to play the game. He continually tries entering the incorrect input, and after three incorrect attempts, he is shown the game over screen.

Fifteen Tiles Puzzle

Steve is shown a 4-by-4 grid filled with 15 numbers represented in hexadecimal notation, since Steve is a computer science student he understands that A represents 10 B represents 11 and so forth. The numbers are randomly scattered throughout the grind and Steve is told to arrange the numbers from low to high.

Case 1:
Steve reads the instructions and begins to solve the puzzle by moving the numbers up-down-left-right using the WASD keys. After 2 minutes Steve completes the puzzle, he is congratulated and the level ends.

Case 2:
Steve reads the instructions and begins to solve the puzzle by moving the numbers up-down-left-right using the WASD keys. After 5 minutes Steve is still unable to solve the puzzle, the time limit is exceeded so the level ends and Steve is shown the game over screen.

Case 3:
Steve does not read the instructions but instead starts to press random keys, the program disgrads the invalid inputs and the puzzle does not change, he eventually sees the instructions and continues to solve the puzzle using the WASD keys.

Decrypt Keeper Puzzle

Steve is given an encrypted message, where the prompt says each word is encrypted with a cipher key, which Steve remembers from his CSCI162 class. He is also given a set of tools: arrows D to add one to the cipher key, and A to subtract one from the cipher key, a transfer to notePad key (to keep track of the decrypted words) and a view notepad key.

Case 1:
Steve starts navigating different encrypted messages with the arrows until he sees his first word “MULTIPLY”. He presses the T key to transfer the first word to the notepad. He subsequently finds the other words one at a time and adds them to the notepad as well. Once he finishes all four words he views the notepad and can decipher the message: “MULTIPLY THIRTY-SEVEN WITH EIGHTEEN“, he pulls out a calculator and gets the resulting “666”. He presses the E key to enter the passcode, 666 and receives a success message and is then prompted to choose a door.

Case 2:
Steve cannot work out the right strings in time so he decides to enter a random three digit number. He enters 123 and gets prompted “You entered the wrong code, you die!” and fails the game.

Case 3:
Steve thinks he can bypass the code by breaking the system. He decides to enter a string “foo” instead of an integer. It does not work! He gets prompted “You entered the wrong code, you die!” and fails the game.

Case 4:
Steve misunderstands the message and decides to subtract the two values, he ends up with a negative integer “-19”. He enters this as the code, it does not work! He gets prompted “You entered the wrong code, you die!” and fails the game.

Regex Challenge Puzzle

Case 1:
Steve is prompted with a message asking him to enter a regular expression for a phone number. He first tries entering “*” (an asterisk) to match all strings and he is notified that it is incorrect and he now has only two tries left.

He still has to do some review on how regular expressions work, so he quickly opens google and searches for this type of regular expression. He finds a stackoverflow answer which seems to offer the correct expression. He enters “^(\+\d{1,2}\s)?\(?\d{3}\)?[\s.-]?\d{3}[\s.-]?\d{4}$” into the prompt and it tells him that he was correct. The level exits successfully.

Case 2:
After trying one carefully crafted regular expression for a phone number which the puzzle says was incorrect, Steve is confused about how to complete the level.

He tries going into the “regex_challenge.py” code to see if he can make sense of it, but it doesn’t help him. He sees a file called “regex_types.json” and runs the terminal command “cat regex_types.json” to see what the file contains. He notices that it contains the correct regular expressions.

Steve copies the one for a phone number and tries entering it into the puzzle. It tells him it is incorrect.

Steve remember from his computer science class that JSON strings must escape backslashes. He realizes that the backslashes in the regular expression are escaped. He tries entering the expression without the double back slashes and this works. The level exits successfully.

Cipher Escape Puzzle (For Cass to do)

Final Puzzle

Case 1:
Steve

Case 2:
Steve reaches the final level. He is prompted to write a code. After entering three wrong answers, he runs out of time and fails the level. A message takes over the screen that reads “GAME OVER, YOU DIED” and the game closes itself. Steve realizes that he still has a lot of reviewing to do before his technical interview.