Bop It Lab

LAB #6

SECTION #2

Jaden Burke

SUBMISSION DATE: 10/11/2022

DATE: 10/11/2022

Problem

We need to write a program in C that will simulate the bop it game by taking input from the controller and prompting the user to press a button within a time limit.

Analysis

The first and easiest part is the scanning user input which can just directly be taken from a previous lab. I also will need to adjust my call in Cygwin to use the appropriate flags in order to get only the time and button data from the controller which are the -t and -b flags. I also need to make several functions in order to put them in a while loop which will be used to keep the game running. I will need three functions: a function to begin the game, a function to print what the user needs to input, and a function to check user input. I also need to implement the failure conditions with if statements.

Design

To start, I directly took the scanf from the previous lab and just trimmed it down to only accept five instead of eight values; the time value and the four buttons. The first function was the easiest, I just needed to check if a button has been pressed by adding all the values of the buttons together. If the value is 0, re-run the while loop from the beginning and if its not 0, continue the while loop and never re run this function. This was done with an if statement and the continue function. I set the condition for the if statement to the opposite of what was returned by the start function and used the continue function until the if statement stops running. For the function that determines user input, I just had to assign a variable to the rand() function mod 4 in order to decide which button needed to be pressed, and depending on what the variable was equal to I just needed to print which button to press. I also print how many milliseconds the user has by using a global variable. Lastly the function that checks if the user input is correct just needs to read the button values and compare it to which button needs to be pressed. If it's the right button, the program will know continue and if its not the program will know to stop. The first failure condition which is pressing the button is pretty simple, because I just need a global variable that will be changed to true if I press the wrong button and have an if statement in the while loop that will cause it to break if the wrong button has been pressed. The other failure condition was fulfilled by storing the start time when the user input decision function is called. After that I just need to make sure that the difference between the current time and the starting time is greater than the time limit, and if it is break the while loop.

Testing

The testing for most of it was very simple. The scanf, starting, and user input decision functions all worked as intended, however the correct button function was the only one I had issues with. Originally I did something really funky by comparing which button has been pressed to the right button using addition since the right button is a value from 0-3, however that caused it to sometimes not register that

the wrong button was pressed. The fix was as easy as making the condition for a correct press more rigorous using an and statement that checks if the specific button has been pressed and if the right button value is the specific value that correlates to that button.

Comments

Question 1: I randomized the buttons that needed to be pressed using the rand() function. My function that I used to do this called the rand function, assigned it to a variable, then modded that value by 4. This new number between 0-3 was then used to decide what button needed to be pressed.

Question 2: I Kept track of what round you were one as well as when you started a round. The rounds needed to be tracked because when you lose you have to print what round you lost on, and the start time of a round was important to track whether you timed out as well as to make the button buffer I used.

Question 3: To make sure that extraneous button presses were not registered, I made a buffer inside my function that checked if the button pressed was correct or not. Since the controller takes much longer than a millisecond to register a button press, it will automatically fail you if I don't do this with the way that I checked correct button presses. I chose to have a buffer of 250 milliseconds before the function would recognize your button press as valid.

Screen Shots

Screenshot 1:

```
jadenb04@CO1318-14 /cygdrive/u/fall2021/se185/lab06
$ ./ds4rd -d 054c:09cc -D DS4_USB -t -b | ./lab06
Please press a button to start the game
Press the triangle button!
You have 10000 milliseconds to respond!
Press the square button!
You have 9500 milliseconds to respond!
Press the square button!
You have 9000 milliseconds to respond!
Press the square button!
You have 8500 milliseconds to respond!
Press the circle button!
You have 8000 milliseconds to respond!
Press the circle button!
You have 7500 milliseconds to respond!
Press the square button!
You have 7000 milliseconds to respond!
Press the triangle button!
You have 6500 milliseconds to respond!
Press the cross button!
You have 6000 milliseconds to respond!
Press the triangle button!
You have 5500 milliseconds to respond!
Wrong button! :(
You lose!
You made it through 10 rounds!
```

Screenshot 2:

```
jadenb04@CO1318-14 /cygdrive/u/fall2021/se185/lab06
$ ./ds4rd -d 054c:09cc -D DS4_USB -t -b | ./lab06
Please press a button to start the game
Press the square button!
You have 10000 milliseconds to respond!
You ran out of time. :(
Thanks for playing!
You made it through 1 rounds!
```

\