Bop-It!

LAB #6

SECTION #3

James Gaul

SUBMISSION DATE: 3/8/24

LAB DATE: 3/1/24

Problem

For this lab, we had to design a simple game in the style of “Bop-It!”, where the program commands the player to submit an input, and the player seeks to provide this input within the time limit. As with previous labs, the input was through a “Dualshock” Bluetooth controller.

Analysis

1. How did you randomize the buttons that needed to be pressed?

I used the standard formula for a random variable between two numbers with the Rand() function. This value was saved to the variable targetButton, then submitted into the gameRound, checkButton, and requestButton functions.

Design

2. What game states, if any, did you keep track of?

I essentially had two game states, one which monitored the user input, and one which determined the results of that input.

The first of these was executed in the gameRound function: This program would receive the desired button and the time interval to press this button. This function would then monitor the controller’s time and input values, looping until the time ran out or a button was pressed. It would return a 1 or 0, indicating whether the correct button had been pressed within the time limit.

The second game state was handled in a while loop within the main() function. This function started the gameRound function, assigning it the time limit and targetButton. If the gameRound function returned 0, the loop would end. Otherwise, targetButton would be assigned a new random value, and timeLimit would be decreased by approximately 10%.

Testing

3. What mechanism did you use to make sure extraneous button presses were not registered?

During testing, I discovered that the function looped into a new game round far too quickly, meaning that a “correct” result would usually be followed by an immediate “incorrect”, as a new button had been requested while the player’s finger was still pressing the previous one.

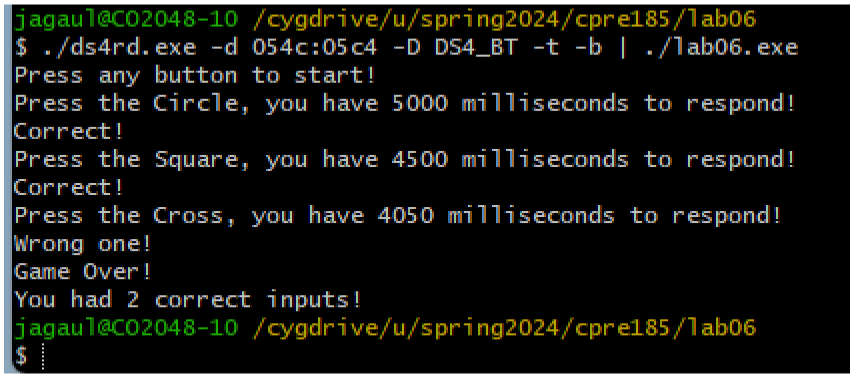
My solution was to add a “delayFor” function, which caused the gameRound function to wait for a tenth of a second before returning a value, giving the player time to release the button.

Comments

This lab's use of constant input values and continuous looping functions heavily reminded me of Unity Game Engine, where scripts are often executed once every frame.

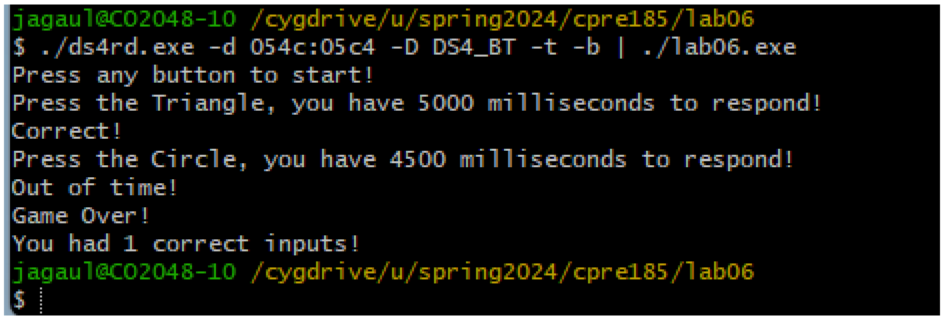
Screen Shots

Game Demo 1 (Game over due to wrong button pressed):



```
jagaul@CO2048-10 /cygdrive/u/spring2024/cpre185/lab06
$ ./ds4rd.exe -d 054c:05c4 -D DS4_BT -t -b | ./lab06.exe
Press any button to start!
Press the Circle, you have 5000 milliseconds to respond!
Correct!
Press the Square, you have 4500 milliseconds to respond!
Correct!
Press the Cross, you have 4050 milliseconds to respond!
Wrong one!
Game Over!
You had 2 correct inputs!
jagaul@CO2048-10 /cygdrive/u/spring2024/cpre185/lab06
$ 
```

Game Demo 2: (Game over due to time running out):



```
jagaul@CO2048-10 /cygdrive/u/spring2024/cpre185/lab06
$ ./ds4rd.exe -d 054c:05c4 -D DS4_BT -t -b | ./lab06.exe
Press any button to start!
Press the Triangle, you have 5000 milliseconds to respond!
Correct!
Press the Circle, you have 4500 milliseconds to respond!
Out of time!
Game Over!
You had 1 correct inputs!
jagaul@CO2048-10 /cygdrive/u/spring2024/cpre185/lab06
$ 
```

Source Code:

```
/*----------------------------------------------------------------------------
-
-                SE 185: Lab 06 - Bop-It!
-      Name: James Gaul

-      Section: 3

-      NetID:      947125207
-     Date: 3/01/24
----------------------------------------------------------------------------
*/

/*--------------------------------------------------------------------------
-
-                                                Includes
                                -
----------------------------------------------------------------------------
*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/*--------------------------------------------------------------------------
-
-                                 Prototypes
-
----------------------------------------------------------------------------
*/
void waitForButtonPress();

void waitForButtonRelease();

void delayFor(int milliseconds);

void requestButtonPress(int requestedButton);

int gameRound(int targetButton, int timeLimit);

int checkButton(int targetButton, int button0, int button1, int button2, int
button3);

/*--------------------------------------------------------------------------
-
-                                     Notes
-
----------------------------------------------------------------------------
*/
// Compile with gcc lab03-2.c -o lab03-2
// Run with ./ds4rd.exe -d 054c:05c4 -D DS4_BT -b | ./lab03-2

/*--------------------------------------------------------------------------
-
-                                                Implementation
                                -
----------------------------------------------------------------------------
*/
int main(int argc, char *argv[])
{
      int roundTime = 5000;

      printf("Press any button to start!\n");
      waitForButtonPress();
      delayFor(1000);
```

```c
	int continueGame = 1;
	int gameScore = 0;

	int targetButton;

	while (continueGame == 1){
		targetButton = (rand() % 4);

		continueGame = gameRound(targetButton, roundTime);

		if (continueGame == 1){
			gameScore++;
			roundTime -= (roundTime / 10);
		}

	}
	printf("Game Over!\n");
	printf("You had %d correct inputs!", gameScore);

    return 0;
}

/* Put your functions here, and be sure to put prototypes above. */



void waitForButtonPress(){
	int button0, button1, button2, button3, buttonCount;
	while (1){
		scanf("%*d, %d,%d,%d,%d", &button0, &button1, &button2,
&button3);
		buttonCount = button0 + button1 + button2 + button3;
		if (buttonCount > 0){
			break;
		}
	}
}

void delayFor(int milliseconds){
	int startTime, currentTime;
	scanf("%d, %*d,%*d,%*d,%*d", &currentTime);
	startTime = currentTime;

	while((currentTime - startTime) < milliseconds){
		scanf("%d, %*d,%*d,%*d,%*d", &currentTime);
	}
}

int gameRound(int targetButton, int timeLimit){

	int currentTime, triangleB, circleB, crossB, squareB, buttonCount;
	int buttonPressed = 0;


	scanf("%d, %d,%d,%d,%d", &currentTime, &triangleB, &circleB, &crossB,
&squareB);
```

```
	int timeCutoff = currentTime + timeLimit;

	requestButtonPress(targetButton);
	printf("you have %d milliseconds to respond!\n", timeLimit);


	while((currentTime < timeCutoff)){

		scanf("%d, %d,%d,%d,%d", &currentTime, &triangleB, &circleB,
&crossB, &squareB);

		buttonPressed = checkButton(targetButton, triangleB, circleB,
crossB, squareB);

		if(buttonPressed == 1){
			printf("Correct!\n");
			delayFor(100);
			return 1;
		}
		if(buttonPressed == 2){
			printf("Wrong one!\n");
			delayFor(100);
			return 0;
		}
	}

	printf("Out of time!\n");
	delayFor(50);
	return 0;

}

int checkButton(int targetButton, int button0, int button1, int button2, int
button3){

	if(targetButton == 0){ //the triangle button is the goal

		if((button0 == 1) && (button1 + button2 + button3 == 0)){
			return 1;
		}
		else if(button1 + button2 + button3 > 0){
			return 2;
		}
		else{
			return 0;
		}
	}

	if(targetButton == 1){ //the circle button is the goal
		if((button1 == 1) && (button0 + button2 + button3 == 0)){
			return 1;
		}
		else if(button0 + button2 + button3 > 0){
			return 2;
		}
		else{
			return 0;
```

```c
		}
	}

	if(targetButton == 2){ //the cross button is the goal
		if((button2 == 1) && (button0 + button1 + button3 == 0)){
			return 1;
		}
		else if(button0 + button1 + button3 > 0){
			return 2;
		}
		else{
			return 0;
		}
	}

	if(targetButton == 3){ //the square button is the goal
		if((button3 == 1) && (button0 + button1 + button2 == 0)){
			return 1;
		}
		else if(button0 + button1 + button2 > 0){
			return 2;
		}
		else{
			return 0;
		}
	}

	printf("Invalid target button %d", targetButton);
	return 0;
}

void requestButtonPress(int requestedButton){
	switch(requestedButton){
		case 0:
			printf("Press the Triangle, ");
			break;

		case 1:
			printf("Press the Circle, ");
			break;

		case 2:
			printf("Press the Cross, ");
			break;

		case 3:
			printf("Press the Square, ");
			break;

		default:
			printf("Error, invalid input of %d\n", requestedButton);
	}
}
```