Conditionals

LAB # 5

SECTION # 3

James Gaul

Submitted: 3/1/24

Lab Date: 2/23/24

Problem

For this lab, we had to create a program which would read inputs from the DualShock controller, use gyroscope readings to determine the controller's orientation, outputting the direction and updating only when it changed. The program was to terminate upon pressing the triangle key.

Analysis

1. How did you approach the design?

   I started by working on the program input, assigning the controller values to variable then printing these out again, to confirm that the data was being properly recorded. Once this worked, I began designing component functions, focusing on detecting when the controller was pointing up only. Once this worked well, I proceeded to expand this to the other five directions.

2. What data did you have to read in?

   I had to read in the acceleration values, gyroscopic values, and button inputs. The acceleration and gyroscope values were in double format, while the buttons were integers.

Design

3. What functions did you choose to implement and why?

   I started with the "close_to" int function specified in the prelab, detecting if a variable was within a tolerance margin of a target value, returning 1 or 0 based on this. As the challenge involved detecting if the function was within range of six different directions, this "close_to" function greatly simplified my code.

   I then created a "getSideUp" function, which first received the gyroscopic values as inputs, then used the "close_to" function to detect if the controller was oriented towards one of the six directions. If it wasn't, this function would return an int value of 0, otherwise, it would return 1-5 based on the direction the controller was pointing. Converting three "double" values into a single "int" made the following steps far easier.

   Finally, I implemented a "printSideUp" void function. This function printed the controller's orientation, based on the integer value from the previous "getSideUp" function. While this function was only executed once and performed no conversions, isolating the numerous "printf" commands for each direction improved the readability of the code.

4. What tolerance values did you pick and how did you decide on them?

I chose a standard tolerance value of 0.2. The controller's x, y, and z gyroscope inputs scaled from -1 to 1, with each cardinal direction returning a value of 1 or -1 on one of the three axis. Thus, a tolerance value of .2 ensured that any orientation within 10% of a direction would return that direction. As the primary goal was simply to detect such cardinal orientations, a more precise tolerance would have simply made testing more finicky.

Testing

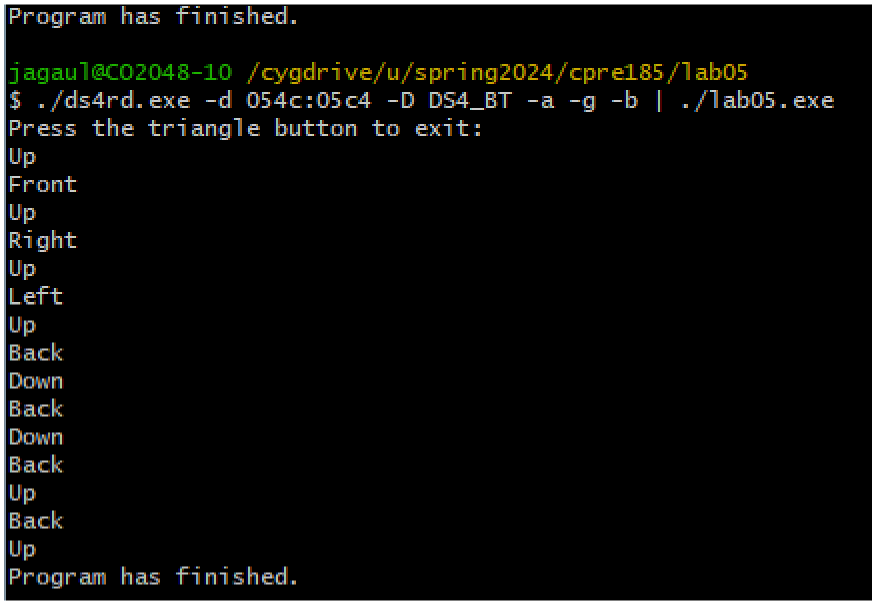
While most of the project went smoothly, I experienced some problems when attempting to ensure that the program would only print when a new orientation was detected. While my code logically seemed to work, it would still sometimes output double inputs. Ultimately, I solved this issue by moving the statement “prevDirection = direction;” inside the if statement, ensuring that the “prevDirection” value could only be updated when it changed, rather than every frame. This resulted in a properly functioning code, as shown in screenshot 1.

Comments

I enjoyed having a single challenge that we were able to approach in a variety of ways, rather than numerous linear tasks to complete.

Screen Shots

1. Finished Program Output



```
Program has finished.

jagaul@CO2048-10 /cygdrive/u/spring2024/cpre185/lab05
$ ./ds4rd.exe -d 054c:05c4 -D DS4_BT -a -g -b | ./lab05.exe
Press the triangle button to exit:
Up
Front
Up
Right
Up
Left
Up
Back
Down
Back
Down
Back
Up
Back
Up
Program has finished.
```