**Conditionals**

**LAB 5**

**SECTION 2**

**RYAN AUN SHERN CHIN**

**SUBMISSION DATE:**

**2/28/24**

# Problem

The objective of this lab is to develop a simple program that can determine the orientation of a DualShock4 controller and interpret acceleration values in relation to gravity. This involves utilizing gyroscope values to establish orientation and accelerometer values to detect movement. A key aspect of this exercise is to refine problem-solving skills and proficiency in using conditional statements in the C programming language. Additionally, the lab aims to foster an understanding of testing multiple conditions and employing tolerances when comparing floating-point values. The end goal is to create a program capable of displaying the facing direction of the DualShock4 controller, which includes identifying whether it's facing front, back, top, bottom, left, or right.

# Analysis

In this lab, our objective is to develop a program structure featuring a while loop to continuously read inputs. This program should be designed in a modular fashion, incorporating at least three custom functions. One of these functions, close\_to, with the prototype ‘int close\_to(double tolerance, double point, double value);’ must be written. Its purpose is to return true when a given value falls within the specified tolerance range of a designated point. This function is crucial for accurate real number comparisons within our code.

Moreover, as per requirements of this lab, we need to modify the ‘while’ loop to terminate and end the program when the user presses the triangle button on the DS4. Furthermore, adjustments must be made to ensure the program only outputs a new line when the orientation changes, where ‘TOP shouldn’t appear multiple times in a row.

# Design

**What functions did you choose to implement and why?**

In solving the problem given to us in this lab, I implemented custom functions called ‘close\_to’ and ‘magnitude’. Furthermore, I also implemented another custom function called ‘orientation’ to detect the orientation of the DualShock4 controller. The function returns different values corresponding to the direction in which the DualShock4 is facing. Crucially, the function used in the main program employs a mechanism where it prints a new line only when there is a change in the orientation of the DualShock4, thus avoiding multiple consecutive output appearances. This behavior is achieved by evaluating whether the difference between a value and a reference point falls within a predefined tolerance. This behavior is also achieved by checking whether if the ‘current\_orientation’ does not equal to ‘last\_orientation’, and the value of magnitude from the magnitude function is less than or equal to 0.2. This is achieved by initializing ‘current\_orientation’ with the value of the current orientation of the DS4 controller determined by the ‘orientation’ function before the if branch and updating the ‘last\_orientation” with the ‘current\_orientation’ at the end of the if branch. If both of the condition is met, indicating a change in orientation, the program proceeds to print the corresponding output. For the other requirement we had, I implemented a function called ‘exit\_program’ that exited the program whenever a specific button is pressed. I chose to implement these functions because it enhances the readability of my code. Without these functions, my code would be filled with repetitions, which would make it harder to understand and maintain.

**How did you approach the design?**

I approached the design by focusing on efficiency and simplicity. For example, the integration of the 'close\_to' function within the 'orientation' function. This integration allowed for estimating and determining the orientation of the DualShock4 by comparing captured raw data from the controller with gyroscope values (-g) within a predefined range of tolerable values for each orientation (Top, Bottom, Left, Right, Front, Back). This approach enabled a clear and structured implementation, facilitating accurate comparisons and efficient processing of the controller's data.

# Testing

To verify the results of the solution, I captured raw data from the controller using the command: `/ds4rd.exe -d 054c:05c4 -D DS4\_BT -t -a -g -b`. This command allowed me to obtain data for acceleration, gyroscope, time, and buttons pressed. I focused specifically on the gyroscope values (gx, gy, gz) and manipulated the orientation of the DualShock4 controller to observe how the program and values behaved. Figuring out the rough estimates of the range of the gyroscope values for each orientation of the DS4 controller that went from -1 to 1, I adjusted and narrowed down the tolerance values within the range of 0.15 to 0.30.

**What data did you have to read in?**

I read in the gyroscope’s values from the DualShock4. This is for the input for the

program. I obtained the data from the DualShock4 and then it directs to scan in the main

class, and next to the ‘close\_to’ function.

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

The tolerance values I picked were 0.25 and 0.30. These values are within the

range of resulting in outputs which are rational in terms of the gyroscopes values. Hence,

the program will result in outputs of the current orientation with the tolerance values

within the value

# Comments

Some comments.

# Screen Shots

<Number the screenshots and paste here. The point of numbering the screenshots is so that you can refer to them during your discussion in the various parts above. Alternatively, you can include the screenshots in-line with the text above as part of your discussion.>