**Conditionals**

**LAB 5**

**SECTION A**

**SUBMITTED BY:**

**Brian Reber**

**SUBMISSION DATE:**

**10/9/2009**

**Lab Problem**

The purpose of this lab is to help us become more comfortable using conditional statements. We will try to have the program output a line stating the direction of the Wiimote when the direction changes.

**Analysis**

The goal of this lab is to print out a line stating which part of the Wiimote is facing up, and then print out a new line when the direction changes. We will need to use many if statements, and learn about the acceleration values that the Wiimote is returning. Also, we will need to know about the tolerances that are necessary because the Wiimote is not perfect.

**Inputs**:

There will be data coming in via the Wiimote and wiiwrap.

**Outputs**:

We will output a line stating which direction is facing up. We will do this only when the direction changes.

**Design**

Algorithm:

1. First off, we will create a function called “tolerance” so that when we need to check if the value is within a certain range, we can call this function instead of repeating it.
2. Then inside the while loop, we will check to see if the magnitude is approximately 1. If not, we don’t need to go through the rest of the checks.
3. Then inside that if statement, we will set the previous orientation to the previous variable, and then set the current orientation to the value returned by our function “whatsup”.
4. We will create the function “whatsup.” This function will check the acceleration values and return the proper orientation. We created an int variable at the beginning. This variable can be any int between 0 & 5. Each value represents a different orientation. We have the key to what value is represented by each int listed at the top of our code. This will then return the int that represents the current orientation.
5. We will then check to see if the current orientation is equal to the previous orientation. If they are, we bypass the rest of the statements. If they are different, we will go inside the if statement.
6. Inside that if statement, we have a series of if statements that checks each int value to find the proper orientation. When it finds the correct one, it will print out the proper orientation (Left or Up or Front etc).
7. Still within the while statement, we will check whether the “B” button is pressed. If it is, we will exit out of the while loop.

**Testing**

We ran into quite a few small problems when writing this program. Many of them were silly mistakes, and some had to do with not knowing much about the wiiwrap program.

* Our first problem was when we tried running the program with the “/B” in the command call. This was making our program go crazy. So for the whole time we ran it without it and it worked. We knew that this dealt with the buttons on the Wiimote, but we weren’t sure how to work with that. We then ran wiiwrap alone with that, and we learned that we needed to add many int parameters that represented the binary value of each button’s state to our scanf statement. It then worked well.
* We chose to use our magnitude function from our previous labs instead of rewriting it.
* We created a close\_to function to test whether a number was within a certain tolerance of another number. This made it easier to check tolerances, because we only needed to have one place to test for tolerance. This reduced the possibility of making typing errors.
* We made a whatsup function that checked the accelerations and found which one was up using that data. We didn’t necessarily need this function, but figured it would be easier if that were outside of the main function.
* We also needed to change the while loop a little. At the beginning, it was running while TRUE. TRUE was always equal to 1 because it was a constant. We instead created a local variable in the main function so that we could change that value to get out of the loop when the “B” button is pressed.

We chose to have a tolerance of .05. This value seemed to work well for us. One thing that we should have done but forgot was to make this tolerance a constant and define it at the top. That would have prevented errors. We originally had our tolerance at .1, but decided to change it, and we had to change all of them instead of having one at the top that when changed would change all of them.

**Comments**

This lab was a lot of fun. I enjoyed having a problem to solve, and having to figure out how to solve it. Going through step by step with my partner was a great way to figure out how to solve the problems. We each noticed errors and were able to correct them pretty quickly. I also think it is really cool that we can control what and when things are printed out on the screen using a Wiimote. This type of lab is cool because we are problem solving both with hardware and software. It is also different because we can see a somewhat practical application to our programs.

**Implementation**

/\* Lab 5 Wrapper Program

Brian Reber & Nathan Brinkman

\*/

#include <stdio.h>

#include <math.h>

/\* Put your function prototypes here \*/

double mag(float x, float y, float z);

int minutes(float ms);

int seconds(float ms);

int leftover(float ms);

int close\_to(float tolerance, float point, float value);

int whatsup(float ax, float ay, float az, int orient, int t);

int main(void) {

int t;

float ax, ay, az;

int a,b,plus,home,minus,one,two,dup,ddown,dleft,dright;

int true = 1;

int orient, previous;

/\*

0 = Front

1 = Tail

2 = Top

3 = Bottom

4 = Left

5 = Right

\*/

while (true) {

scanf("%d,%f,%f,%f,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d", &t, &ax, &ay, &az,

&a,&b,&plus,&home,&minus,&one,&two,&dup,&ddown,&dleft,&dright);

if (close\_to(.05,1,mag(ax,ay,az)))

{

previous = orient;

orient = whatsup(ax,ay,az, orient, t);

if (!(orient == previous))

{

if (orient ==0)

printf("Front\n");

if (orient ==1)

printf("Tail\n");

if (orient ==2)

printf("Top\n");

if (orient ==3)

printf("Bottom\n");

if (orient ==4)

printf("Left\n");

if (orient ==5)

printf("Right\n");

}

}

if(two)

true = 0;

}

return 0;

}

double mag(float x, float y, float z)

{

return sqrt(x\*x + y\*y + z\*z);

}

int close\_to(float tolerance, float point, float value)

{

return (value < point+tolerance && value > point-tolerance);

}

int whatsup(float ax, float ay, float az, int orient, int t)

{

if (close\_to(.05,-1,ay))

{

orient = 0;//Front

}

if (close\_to(.05,1,ay))

{

orient = 1;//Tail

}

if (close\_to(.05,-1,az))

{

orient = 3;//Bottom

}

if (close\_to(.05,1,az))

{

orient = 2;//Top

}

if (close\_to(.05,-1,ax))

{

orient = 5;//Right

}

if (close\_to(.05,1,ax))

{

orient = 4;//Left

}

return orient;

}