Brenyn Jungmann

bwjmbb@mail.missouri.edu

Abstract

This combination of hardware and software is used to determine what pass a quarterback threw based on the real time event of when he released the football.

ECE 4220 PROJECT   
FInal Report

Quarterback Passing Estimator

**Abstract:**

The reason I chose this is because my brother is a quarterback in high school and I originally wanted to be able to track the ball in the air to determine the route that was thrown. My thought was, is there a way to track a football throughout it's time in the air. When a football flies through the air there is a lot of factors that contribute to the point where it lands. Some of these factors are air resistance, gravity, velocity, angle, wind, and the weight of the football. The way that I thought I could do this was to track the football somehow via tracking device, which would have allowed me to get very accurate data. I also was thinking that I could attach a wireless sensor on the football so that I could get the time it was released and when it was caught.

**Introduction:**

My project was meant to be able to track a football throughout its path in the air, which I determined to be a very large project instead a small one that could be completed this semester. I decided that I would have to implement some of the smaller parts of this big project to be able to finish in a reasonable amount of time. One component of my project was the pressure sensor but it was not wireless due to cost. Another component that I wasn't able to implement was to have a tracking device on a football that would allow me to track it on its path from release till it was caught. Instead I used a random generator to create a distance and an initial angle of travel based on the amount of time the ball was in the air. This then allowed me to make some calculations based off the trajectory of the football. The motivation behind this project was to learn more about the physics of the path of a football as well as to learn how to use hardware components that could simulate the real implementation of the big system. The constraints of my project were that I was creating the timing of the pass, randomly generating the distance and initial angle, and I did not take into the effects of air resistance, wind, or the spin of that football coming out of the quarterbacks hand.

**Background:**

Football has been around since the late 1800's so I am sure that someone out there has messed with tracking a football but as far as I know no one had created a product to do this. The tracking of a football throughout the air can be estimated very close to that of the actual thing by using the physics equations for projectile motion. The easiest way to calculate these values is to use the 2 dimensional equations which allow us to assume no left of right but just horizontal and vertical parabola. The background of the physics behind projectile motion is that around 400 BC Aristotle discovered it but it was not accurately defined until the late 1500's. Every object on earth falls to the ground at the same rate of 9.8 m/s because of gravity, which I used in my calculations. Also the horizontal and vertical components of an object flying through the air are independent of each other and there is no horizontal acceleration. (Proj 2015)

**Proposed Method:**

I implemented this project by first programming an Arduino Uno which I hooked up to a force sensitive resistor which acted as my pressure sensor. I programmed the Arduino so that it would not doing anything when the sensor was pressed the first time but when it was released it would get the time in milliseconds and then when it was pressed again I would get the time again and then subtract the two values and get the time from the start to finish. Once this was done I needed to send the time to the local machine via serial communication, sending it from the Arduino was very easy. Now that the Arduino was sending the data I needed to write a program on the local machine that received the data from the Arduino and then forwarded that same data to the TS-7250 board directly by using serial communication once again. I then needed to receive that same data on the TS-7250 board program, and read the incoming data the same way as I did in the local machine program but with a few modifications. I used the serial\_ece4220.c and srial\_ece4220.h files that we were given in lab 3. Once the data was on the board I could now start using my constraints that I set and calculate the data of the travel, to do this I created a randomly generate double that would get a random value out of a range based on the amount of time the football was in the air. I did the same thing to get a randomly generated integer for the angle of projection so that I would be able to calculate the initial velocity, acceleration, and the maximum height of the pass. Once I calculated this data I used an array of structures to hold all the values, this seemed like the fastest and most reasonable way to store the data. I tried to use the fairCom database to store the data but I was unsuccessful in doing so.

**Experiments and Results:**

I tested my project over a hundred times total, the first thing that I tested was the serial write and read. I did this very thoroughly so that I was certain that I was getting the correct values each and every time I was sending and receiving data. I was able to check these values very easily because I tested on my personal Linux machine which I ran the Arduino program and I also ran serial4220Proj.c which both printed out the values that was being sent to the local machine. Once I knew that the serial port was working to the local machine from the Arduino it was time to write and test the program on the TS-7250 board. I then did the exact same testing on the local machine code and the TS-7250 board program ece4220Project.c. While I was testing this the first few times I was not receiving the correct data and I was actually getting blank characters but then I realized that I was writing the wrong character array to the serial port and determined that the code I wrote was correct so it was an easy fix. The next thing to do was to check the equations that I had used to determine my calculated values, to do this I let the equations in the computer calculate its data and then I checked it by using the initial values that were generated. I tested around 20 different sets of calculations and after that I was very confident that my calculations part was correct. Once I had all my calculations I saved that data into an array of structures so that I could save multiple instances in a timely matter and then print it out. I checked these values about 10 times before and after I had stored the data and it all was the same output. The next step was to determine which pass was thrown and this was not very hard since I was determining all the possible pass routes. I determined the possible pass routes by the distance the football was thrown since I was doing the 2 dimensional simulation. If I was to do the 3 dimensional simulation then I would have needed to include the angle as well as the distance to find the calculations. I ran multiple tests on the passing routes to make sure they were consistently based on the distance that was randomly generated based on the time of the pass.

**Diagrams:**

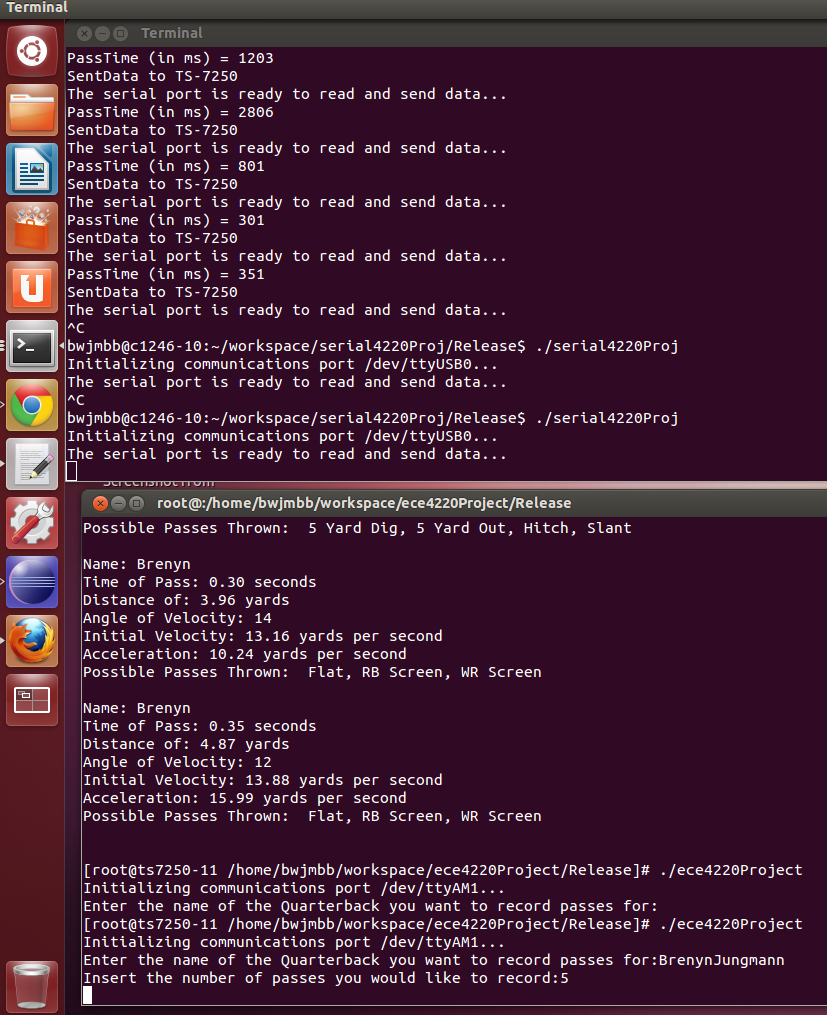
****

Figure 1: Waiting for time from Arduino

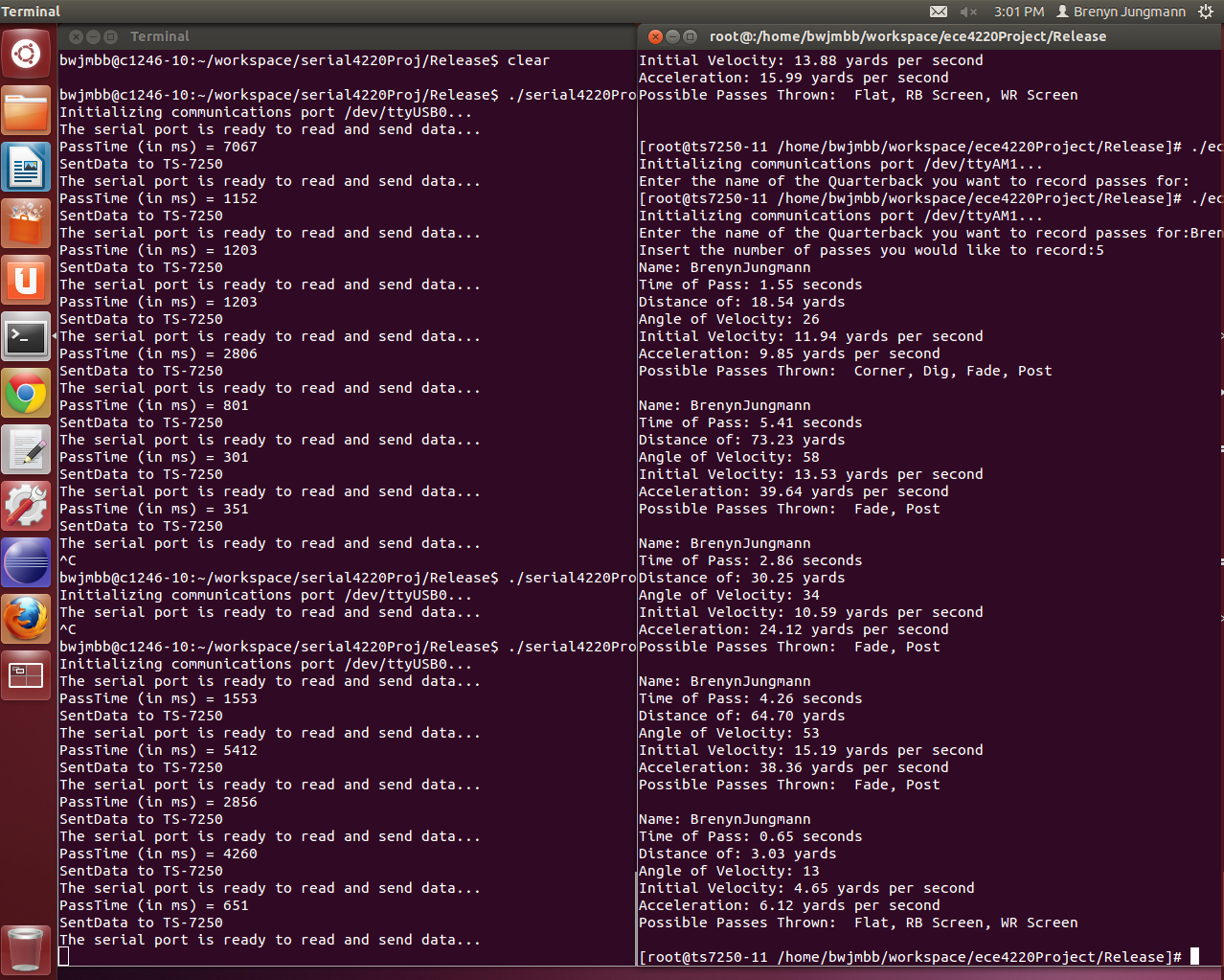


Figure 2: 5 Passes from Brenyn (QB Name)

**Discussion and Conclusion:**

This project was a good project in theory but as I dug deeper and deeper into it, it seemed to be a very complicated and expensive project to do the correct way and in a way that could produce a product that people would want and use. I changed a lot of things that I originally wanted to do because of costs, time, and knowledge of the design and implementation. I came to the realization that the only thing that I would be able to do this semester would be to simulate some of the small parts of the much bigger complete project. This project was a very good review from physics about projectile motion and now I think understand a little more about what all it would take to truly design the real application. Although I had a lot of things go wrong and couldn’t figure them out, I feel that I learned a lot from this project and I definitely learned a lot about the Arduino and to transfer data via serial port. I also learned a lot more about Linux machines because a lot of the debugging of the serial port was in terminal and using the command line to determine where the serial port devices were. The calculations for this seem a little heavy on randomly generated numbers based of the time and I wish I could have figured out another way to get more accurate data for these but with the time I had to work on it I think it turned out pretty good. Another thing that would have really helped the complexity of my project would have been to get the fairCom database implemented and being able to read and store data into the fairCom tables.

**Appendices:**

**About FSR (Force Sensitive Resistor):**

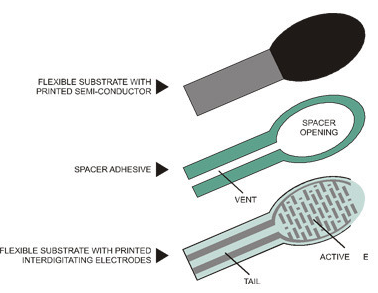
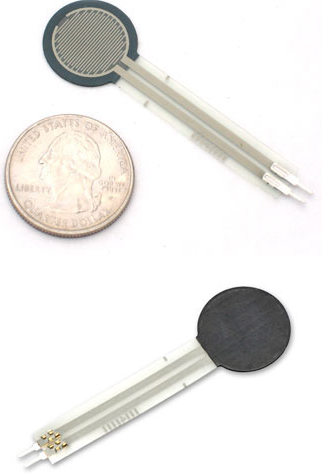


Figure 3: FSR Design

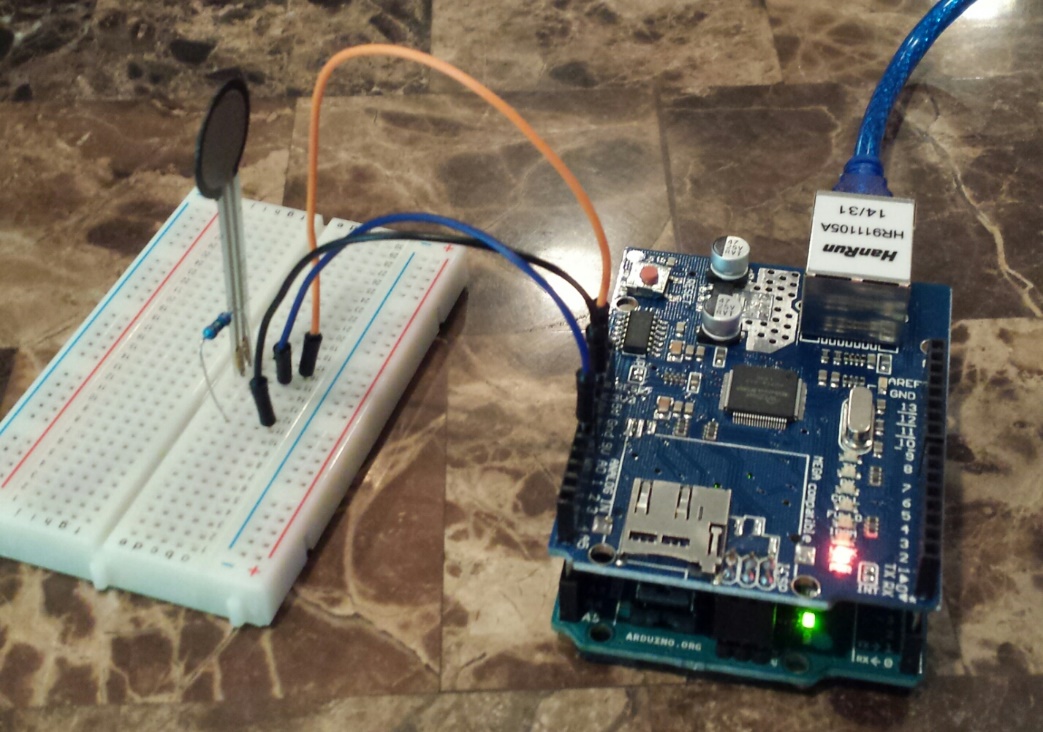


Figure 4: Arduino and FSR Circuit

**Flow Chart:**

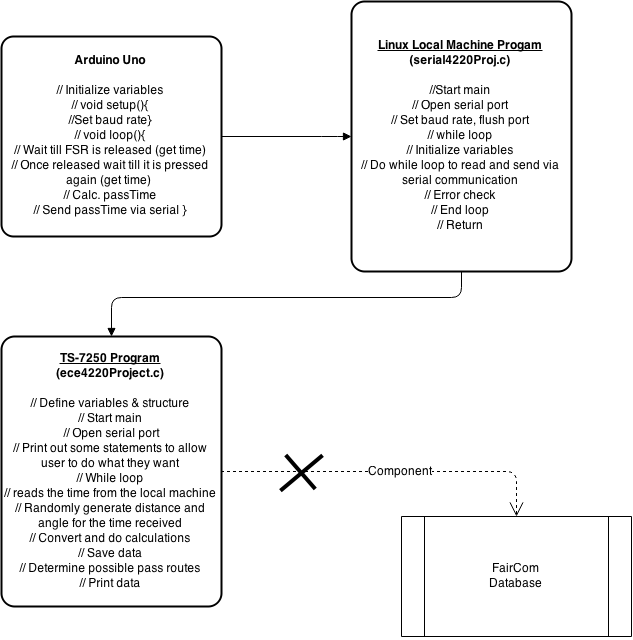
****

Figure 5: Project Flow Chart

**Arduino Code:**

int fsrAnalogPin = 0; // FSR is connected to analog 0

int fsrReading; // the analog reading from the FSR resistor divider

unsigned long releaseTime = 0;

unsigned long catchTime = 0;

unsigned long passTime = 0;

char buffer[20];

void setup(void) {

Serial.begin(9600); // We'll send debugging information via the Serial monitor

}

void loop(void) {

// Wait till sensor is pressed

while(fsrReading == 0){

fsrReading = analogRead(fsrAnalogPin);

delay(50);

}// While pressed down wait till released

if(fsrReading > 0){

while(fsrReading != 0){

fsrReading = analogRead(fsrAnalogPin);

delay(50);

}

// Serial.println("Quaterback Released the football");

// Get release time

releaseTime = millis();

// Wait till sensor pressed again

while(fsrReading == 0){

fsrReading = analogRead(fsrAnalogPin);

delay(50);

}

// Get catch time

catchTime = millis();

//Serial.println("Receiver caught the football");

// Calculate pass time and send via serial

passTime = catchTime - releaseTime;

Serial.println(passTime);

}

// Wait again till the sensor is released

while(fsrReading != 0){

fsrReading = analogRead(fsrAnalogPin);

delay(50);

}

}

**Linux Local Machine (serial4220Proj):**

#include <stdio.h> // standard input / output functions

#include <stdlib.h>

#include <string.h> // string function definitions

#include <unistd.h> // UNIX standard function definitions

#include <fcntl.h> // File control definitions

#include <errno.h> // Error number definitions

#include <termios.h> // POSIX terminal control definitions

#include "serial\_ece4220.h"

#define MSG\_SIZE 30

int main(void)

{

// Variable definitions

struct termios tty;

struct termios tty\_old;

int n = 0, m = 0;

// Open serial port from arduino

int USB = open("/dev/ttyACM0",O\_RDWR| O\_NOCTTY);

// Open serial port to the TS-7250 board

int a = serial\_open(0,5,5);

// Check to make sure file opened correctly

if(USB < 0){

printf("File USB did not open correctly.\n");

exit(0);

}

memset (&tty, 0, sizeof tty);

// Error Handling

if ( tcgetattr ( USB, &tty ) != 0 ) {

printf("Error1\n");

}

// Save old tty parameters

tty\_old = tty;

// Set Baud Rate

cfsetospeed (&tty, (speed\_t)B9600);

cfsetispeed (&tty, (speed\_t)B9600);

// Setting other Port Stuff

tty.c\_cflag &= ~PARENB; // Make 8n1

tty.c\_cflag &= ~CSTOPB;

tty.c\_cflag &= ~CSIZE;

tty.c\_cflag |= CS8;

tty.c\_cflag &= ~CRTSCTS; // No flow control

tty.c\_cc[VMIN] = 1; // Read doesn't block

tty.c\_cc[VTIME] = 5; // 0.5 seconds read timeout

tty.c\_cflag |= CREAD | CLOCAL; // Turn on READ & ignore ctrl lines

// Make raw

cfmakeraw(&tty);

// Flush Port, then applies attributes

tcflush( USB, TCIFLUSH );

if ( tcsetattr ( USB, TCSANOW, &tty ) != 0) {

printf("Error2\n");

}

// Loop to read from arduino multiple times

while(1){

//Initialize Variables

int spot = 0;

char response[MSG\_SIZE] = "";

memset(response, '\0', MSG\_SIZE);

// Serial Port is ready

printf("The serial port is ready to read and end data...\n");

// Read data from the arduino via serial to local machine

// Once I have data from arduino forward to the TS-7250 board

// Do this for every character received from the arduino

do {

n = read(USB, &response[spot], sizeof(char));

m = write(a,&response[spot], sizeof(char));

spot += n;

} while(response[spot-1] != '\n' && n > 0);

// Error check the read from the arduino

if (n < 0) {

printf("Error reading\n");

fflush(stdout);

}

else if (n == 0) {

printf("Read nothing\n");

fflush(stdout);

}

else {

printf("PassTime (in ms) = %s",response);

fflush(stdout);

}

// Error check the write to TS-7250 via serial

if (m < 0) {

printf("Error writing\n");

}

else if (m == 0) {

printf("Wrote nothing\n");

}

else {

printf("SentData to TS-7250\n");

}

}

// Exit

return 0;

}

**TS-7250 Code (ece4220Project):**

/\*

============================================================================

Name : ece4220Project.c

Author : bwjmbb

Version :

Copyright :

Description : Hello World in C, Ansi-style

============================================================================

\*/

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include "serial\_ece4220.h"

#include <fcntl.h>

#include <time.h>

#include <math.h>

// Define the maximum message size that can

// be recieved via serial

#define MSG\_SIZE 30

#define NAME\_SIZE 25

#define PASS\_SIZE 12

typedef struct{

char \*name;

double passTime;

double distance;

double initVelocity;

double maxHeight;

double acceleration;

int angle;

char passRoute[5][PASS\_SIZE];

}Data;

// Function prototypes

double randFloat(double, double);

void print\_data(Data);

void print\_route(int,Data);

// Main

int main(int argc,char \*argv[]) {

// Define variables

int portId = 0, numBytes = 0, angle = 0,totalPasses = 0, counter = 0;

double distance = 0.00, initVelocity = 0.00, passTime = 0.00, maxHeight = 0.00,

Vox = 0.00, Voy = 0.00,Vx = 0.00, Vy = 0.00, accel = 0.00;

char name[NAME\_SIZE] = "";

srand(time(NULL));

// Open up port from the local machine

portId = serial\_open(0,5,5);

// Ready to start reading

printf("Enter the name of the Quarterback you want to record passes for:");

fflush(stdout);

scanf("%s",name);

printf("Insert the number of passes you would like to record:");

fflush(stdout);

scanf("%d",&totalPasses);

Data person[totalPasses];

// Loop to read in multiple times from serial

while(counter < totalPasses){

// Initial Variables

int indexP = 0;

char timeRec[MSG\_SIZE] = "";

memset(timeRec, '\0', sizeof(timeRec));

// Read while the condition is true to get whole serial value

do {

numBytes = read(portId,&timeRec[indexP],sizeof(char));

indexP += numBytes;

} while(timeRec[indexP-1] != '\n' && numBytes > 0);

// Check to make sure the read function worked

if (numBytes < 0) {

printf("Error reading\n");

fflush(stdout);

exit(0);

}

else if (numBytes == 0) {

printf("Read nothing\n");

fflush(stdout);

exit(0);

}

// Convert char array passTime to double to check

// and perform operations on

passTime = (double)atoi(timeRec);

//Use time to randomly generate a distance and angle

if(passTime >= 0 && passTime < 750){

distance = randFloat(2.00,7.00);

angle = (rand() % 3) + 12;

}

else if(passTime >= 750 && passTime < 1500){

distance = randFloat(7.00,15.00);

angle = (rand() % 6) + 15;

}

else if(passTime >= 1500 && passTime < 2250){

distance = randFloat(15.00,25.00);

angle = (rand() % 7) + 21;

}

else if(passTime >= 2250 && passTime < 3000){

distance = randFloat(25.00,35.00);

angle = (rand() % 7) + 28;

}

else if(passTime >= 3000 && passTime < 3750){

distance = randFloat(35.00,45.00);

angle = (rand() % 7) + 37;

}

else if(passTime >= 3750 && passTime < 4250){

distance = randFloat(45.00,55.00);

angle = (rand() % 7) + 44;

}

else if(passTime >= 4250 && passTime < 5000){

distance = randFloat(55.00,65.00);

angle = (rand() % 7) + 51;

}

else if(passTime >= 5000 && passTime < 5750){

distance = randFloat(65.00,80.00);

angle = (rand() % 7) + 58;

}

else if(passTime > 5750){

distance = randFloat(65.00,80.00);

angle = (rand() % 7) + 65;

}

// Convert to seconds to calculate velocity

passTime = passTime/1000;

// Calculate velocity (f/s)

initVelocity = distance/passTime;

// Calculate

Vox = initVelocity\*cos(angle);

Voy = initVelocity\*sin(angle);

Vx = Vox;

Vy = Voy - 9.8\*passTime;

accel = sqrt(pow(Vx,2.00) + pow(Vy,2.00));

/\*

\* Using Eq. h = (Vf^2 - Vi^2) / 2g

\* h = max height, Vf = final velocity, Vi = initial velocity

\* g = gravity (negative)

\*/

maxHeight = (0.00-pow(initVelocity,2.00)) / (2 \* -9.8);

// Set all the values calculated into a structure array

person[counter].acceleration = accel;

person[counter].distance = distance;

person[counter].initVelocity = initVelocity;

person[counter].maxHeight = maxHeight;

person[counter].name = name;

person[counter].passTime = passTime;

person[counter].angle = angle;

print\_data(person[counter]);

// Determine pass route

if(distance < 5){

strcpy(person[counter].passRoute[0],"Flat");

strcpy(person[counter].passRoute[1],"RB Screen");

strcpy(person[counter].passRoute[2],"WR Screen");

print\_route(3,person[counter]);

}

else if(distance >= 5 && distance < 10){

strcpy(person[counter].passRoute[0],"5 Yard Dig");

strcpy(person[counter].passRoute[1],"5 Yard Out");

strcpy(person[counter].passRoute[2],"Hitch");

strcpy(person[counter].passRoute[3],"Slant");

print\_route(4,person[counter]);

}

else if(distance >= 10 && distance < 15){

strcpy(person[counter].passRoute[0],"Comeback");

strcpy(person[counter].passRoute[1],"Curl");

strcpy(person[counter].passRoute[2],"Fade");

strcpy(person[counter].passRoute[3],"Out");

strcpy(person[counter].passRoute[4],"Post");

print\_route(5,person[counter]);

}

else if(distance >= 15 && distance < 25){

strcpy(person[counter].passRoute[0],"Corner");

strcpy(person[counter].passRoute[1],"Dig");

strcpy(person[counter].passRoute[2],"Fade");

strcpy(person[counter].passRoute[3],"Post");

print\_route(4,person[counter]);

}

else if(distance >= 25){

strcpy(person[counter].passRoute[0],"Fade");

strcpy(person[counter].passRoute[1],"Post");

print\_route(2,person[counter]);

}

++counter;

}

// Exit

return 0;

}

// Function to generate a random float value

double randFloat( double low, double high ) {

return ((double)rand()\*(high - low)) / (double)RAND\_MAX + low;

}

// Function to print data

void print\_data(Data person){

printf("Name: %s\n",person.name);

printf("Time of Pass: %.2f seconds\n",person.passTime);

printf("Distance of: %.2f yards\n",person.distance);

printf("Angle of Velocity: %d\n",person.angle);

printf("Initial Velocity: %.2f yards per second\n",person.initVelocity);

printf("Acceleration: %.2f yards per second\n",person.acceleration);

printf("Possible Passes Thrown: ");

}

// Function to print passRoute data

void print\_route(int size, Data person){

int i = 0;

for(i = 0; i < size; i++){

if(i != size -1){

printf(" %s,", person.passRoute[i]);

}

else{

printf(" %s\n\n", person.passRoute[i]);

fflush(stdout);

}

}

}

**References:**

"Projectile Motion: Basic Info." Projectile Motion: Basic Info. Alabama Team #1, n.d. Web. 15 May 2015.

"Projectiles." The Physics Hypertextbook. Glenn Elert, n.d. Web. 15 May 2015.