THE PHYSICS OF PAINTBALL

The New Mexico Adventures in

Supercomputing Challenge

Final Report

April 6, 2005

Team 035

Manzano High School

Team Member

Darren Kartchner

Sponsoring Teacher

Steve Schum

EXECUTIVE SUMMARY

My project is to write a C++ program that will calculate the trajectory of a fired paintball based on values input by the user. The core of the programming code is based on the equations of two-dimensional trajectory physics, with additional formulas to increase the practical accuracy of the program. After researching the factors that will most likely influence the movement of a paintball, I decided to use gravity and air drag as the two factors that prevent a paintball from traveling in a simple straight line.

The project began with consulting both the Internet and my physics book for applicable formulas and the basic stats of a fired paintball, including initial velocity, mass, and cross-sectional area. Once this was accomplished, I began writing a program that would apply these formulas, calculating the instantaneous position and velocity of the paintball in one-tenth second intervals, and then outputting these values in a table, which could be sent to a graphing program to view possible trajectories. The user will input initial vertical height, launch angle, and initial velocity, then the program will compute values and graph.

The program successfully completes its purpose, and results will be available in the future.

Project Definition

Paintballing is a sport where the players are at the whim of physics. Countless factors are in play while paintballs are launched, giving even the most accurate of paintballers a miniscule advantage. From the most obvious factors, like gravity, to the less known, like the air drag based on air density, a paintballer's only control of each projectile is the initial velocity of the paintball. This project is intended to use mankind's knowledge of physics as of now to predict the trajectory of each paintball, depending on natural and artificial forces. While it is unlikely that a paintballer will use this program with utmost practicality, it will give him or her an idea of an optimum firing angle or how even the slightest breeze can mean blowing one's paintball several yards off-course. Physics is obviously the central science in this problem, but to actually construct a program that only requires values to be input by the user is what I plan to do with my project. Implementing gravity and air drag, I will write a program that computes the position, velocity, and acceleration of the paintball in one-tenth second intervals, then outputs them into an organized format that can easily be transferred to Excel or another graphing utility, where a visual trajectory can be seen.

Procedure

The program begins by asking the user to input values for initial vertical position (or the height of the gun from the ground), initial velocity (dependent on the make of the gun, but usually around 200 ft/s), and launch angle. The values will be inserted into formulas using these variables:

- x = horizontal displacement (m)
- y = vertical displacement (m)
- v0 = initial velocity (m/s)
- Θ = angle of launch (°)
- x0 = initial horizontal displacement (m)
- y0 = initial vertical displacement (m)
- t = time (s)
- vh = horizontal velocity (m/s)
- vv = vertical velocity (m/s)
- g = Earth's gravity (m/s2)

The formulas (minus air drag) are seen here:

- x = x0 + (v0cosΘ)(t)
- y = y0 + (v0sinΘ)(t) + ½ gt2
- vh = v0cosΘ = constant, if no air drag
- vv = v0sinΘ + gt

After this, air drag is added, along with these variables:

- ρ (rho) = air density (kg/m^3)
- C = drag coefficient (since paintballs are round, $C = 0.5$ in this program)
- A = cross sectional area (m^2)
- F_h = horizontal force (N)
- F_v = vertical force (N)
- R = drag force (N)
- m = paintball mass (kg)
- a_h = horizontal acceleration (m/s^2)
- a_v = vertical acceleration (m/s^2)

The formulas, with air drag are seen here:

- $R = -0.5C\rho Av^2$ ($R = F_h$ in horiz; $R = F_v$ in vert)
- $a_h = R/m$
- $x = x_0 + v_h t + 0.5a_h t^2$
- $v_h = v_0\cos\Theta + a_h t$
- $F_v = mg + R$
- $a_v = (mg + R)/m$
- $y = y_0 + v_v t + 0.5a_v 2$
- $v_v = v_0\cos\Theta + a_v t$

The program will compute the position of the paintball in 0.1-second increments (the time-step method). The information will be output into an organized table displaying time, horizontal position, horizontal velocity, vertical distance, and vertical velocity. This information can be transferred to a graphing program where a visual trajectory can be seen.

RESULTS

I have successfully written a C++ program that computes vertical distance, vertical velocity, horizontal distance, and horizontal velocity of a paintball based on values input by the user. Using the information output by the program, a trajectory model can be created using a graphing utility.

The accuracy of the model has been checked and found to be precise enough for the purpose of my program.

CONCLUSION

The past year I have used my new skills with C++ to write a very practical program that should prove useful to paintballers, as well as provide a physics lesson to anyone willing to fiddle with values and observe the differences in the trajectories. I was not able to attach any runs to my project, but I should be able to bring some to the expo. By then I hope to be able to input the influence of wind on the paintball's trajectory. This project could have used more input of my time, but it still fulfills its purpose.

ACKNOWLEDGEMENTS

- http://home.comcast.net/~dyrgcmn/pball/pballphys1.html
- http://www.paintball-gun.net
- http://www.directpaintball.com/article_mike_myths.html
- http://www.paintballzone.com/sniper_math_physics.html
- Zitzewitz, Physics: Principles and Problems (1999)
- Mr. Schum, whose physics experience and persistence made sure this project was finished.

APPENDIX A: COMPUTER CODE

/* Team 035 Darren Kartchner 06Apr05

Steve Schum, MHS Teacher and Programming Advisor

filename: pbtraj2.cpp

Part 1:

This program will compute and store, in a class, the velocity and displacement for a
paintball
launched in a trajectory with parameters defined by the user. The program will ask
for the
initial velocity, initial position above the horizontal, and angle of launch. The
program will
then output results in 0.1 second intervals.

Part 2:

This program will repeat Part 1, only now it will include the effect of air drag on the trajectory
of the paintball. The user will need to input air density of environment (in kg/m^3), cross-sectional
area of paintball (in m^2), and the total mass (in kg). The program will compute the drag force, then
apply it to the results of Part 1 to calculate changing velocities, accelerations, and displacements.*/

```
#include <iostream.h>

#include <math.h>                   //needed for trigonometry and exponents

#include <iomanip.h>                //needed for setw()

#include <fstream.h>                //needed for file input and output

#include <stdlib.h>                 //needed for exit()


const float g = -9.80;              //gravity acceleration

const float pi = 3.14159;   //convert degrees to radians

const float mph = 0.6818; //converts ft/s to mph
```

```
#define FILE_IN "DVDataInput.in"        //sends results to project file

#define FILE_OUT "DVDataOutput.out"


class DVData                              //Class contains data members and member
functions that can act on these data members

{
 public:

 float v0, v0fps, v0mph, angle, dvft;        // d = units of meters; v = units of
meters/second

 int maxnum;

 double rad;

 float t[101], dv[101], vv[101], dh[101], vh[101]; //d = meters; v = m/s

 float C,     //drag coefficient
       rho,   //air density
       A,     //cross-sectional area
       m,     //mass of paintball
       delta_t;        //time-step interval

 float Fhwd[101],  //horizontal drag force, in newtons
       ahwd[101],      //horizontal deceleration
       vhwd[101],    //horizontal veloctiy w/ drag
       dhwd[101];    //horizontal displacement w/ drag
```

```
 float Fvwd[101],  //vertical drag force, in newtons
       avwd[101],       //vertical deceleration
       vvwd[101],    //vertical veloctiy w/ drag
       dvwd[101],    //vertical displacement w/ drag

       Fvwdnet[101];

 void input_dv_initial(void);
 void compute_dv(void);
 void output_dv(void);
};

//------------------------------------------End DVData Definition--------------------------------------------

using namespace std;        //!!!!!

int main()
{
 DVData DVD1;               //declare instance of class

 DVD1.input_dv_initial();

 DVD1.compute_dv();

 DVD1.output_dv();

 return 0;
}
```

```
//-----------------------------------------End of main-------------------------------------------
------------

void DVData::input_dv_initial(void)      // :: is "class resolution operator"

{
 int choice;

 C = 0.5                                  //no units, C = 0.5 for spherical objects

 cout << "\nThis program computes the velocity and displacement of a trajectory."
     << "\nEnter 1 if you want to enter your own initial launch values."
     << "\nEnter 2 if you want to input data from an existing file."
     << "\nEnter 3 if you want to exit the program." << endl;

 cin >> choice;

 cout.precision(1);                       //sets choice precision

 cout.setf(ios::showpoint |ios::fixed);             //sets choice precision

 switch(choice)

 {case 1:

 {
   cout << "\nEnter an initial speed at launch in miles per hour." << endl;

   cin >> v0mph;

   v0fps = (v0mph * 5280) / 3600;
```

```
v0 = v0fps / 3.28;


cout << "\nInitial speed at launch = " << v0mph << " mph" << endl;
cout << "\nInitial speed at launch = " << v0 << "m/s" << endl;

cout << "\nEnter an angle of launch above the horizontal in degrees." << endl;

cin >> angle;

cout << "\nInitial launch angle = " << angle << " degrees" << endl;

 rad = double((2.0 * pi * angle) / 360.0);

vv[0] = v0 * float(sin(rad));

cout << "\nInitial vertical velocity = " << vv[0] << " m/s" << endl;

vh[0] = v0 * float(cos(rad));

cout << "\nInitial horizontal velocity = " << vh[0] << " m/s" << endl;

cout << "\nEnter an initial height at launch in feet." << endl;

cin >> dvft;

dv[0] = dvft / 3.28;

cout << "\nInitial vertical height = " << dv[0] << " meters" << endl;
```

```
  break;
}

 case 2:
 {
  ifstream input;                    //Set up input stream object

  input.open(FILE_IN, ios::in | ios::nocreate);
                                     //Check to make sure file is there

  if( !input)

     {
     cout << "\nCan't find input file " << FILE_IN;
     cout << "\nExiting program" << endl;
     exit(1);
      }

   input >> v0;

   cout << "\nInitial speed at launch = " << v0 << " mph." << endl;

   input >> angle;

   cout << "\nInitial launch angle = " << angle << " degrees" << endl;

   input >> vv[0];

   cout << "\nInitial vertical velocity = " << vv[0] << " m/s" << endl;

   input >> vh[0];
```

```
   cout << "\nInitial horizontal velocity = " << vh[0] << " m/s" << endl;

   input >> dv[0];

   cout << "\nInitial vertical height = " << dv[0] << " ft" << endl;

   input.close();

   break;
   }

   default:

   cout << "\nExiting program." << endl;

   exit(1);

  }
}

//-------------------------------------End of input------------------------------------------

void DVData::ocmpute_dv(void) // Begin compute_dv function
{

  int i;

  dh[0] = 0.0                              //Initializing variables

  vhwd[0] = vh[0];
```

```
dvwd[0] = dv[0];

vvwd[0] = vv[0];

t[0] = 0.0;

Fhwd[0] = 0.0;

Fvwd[0] = 0.0;

delta_t = 0.1                    //time step

for(i = 1; i < 10; i++)

{
  t[i] = float(i/10);

  dv[i] = dv[0] + vv[0] * t[i] + 0.5 * g * pow(t[i],2);

  vv[i] = (vv[0] + g * t[i]);

  vh[i] = vh[0];          //constant horiz. velocity

  dh[i] = vh[0] * t[i];

  Fhwd[i] = -.500 * C * rho * A * pow(vh[i-1],2);

  ahwd[i] = Fhwd[i]/m;

  vhwd[i] = vhwd[i-1] + ahwd[i] * delta_t;
```

```
dhwd[i] = dhwd[i-1] + vhwd[i] * delta_t + .500 ahwd[i] * pow(delta_t,2);

Fvwd[i] = -.500 * C * rho * A * pow(vvwd[i-1],2);

if (vvwd[i-1] >= 0 )

avwd[i] = (m * g + Fvwd[i]) / m;

else

avwd[i] = (m * g - Fvwd[i]) / m;

vvwd[i] = vvwd[i-1] + avwd[i] * delta_t;

dvwd[i] = dvwd[i-1] + vvwd[i] * delta_t + 0.500 * ahwd[i] * pow(delta_t, 2);
 }
}

//---------------------------------------End compute_dv----------------------------------

 void DVData::output_dv(void)            //begin cout_dv

 {
  int i = 0;

  cout precision(1);

  cout.setf(ios::showpoint |ios::fixed);

  cout << endl << endl;
```

```
  cout << setw(10) << "time(sec)" << setw(14) << "vv(mph)"
       << setw(12) << "dv(ft)" << setw(14) << "vh(mph)"
     << setw(12) << "dh(ft)" << endl;
  cout << setw(10) << t[i] << setw(14) << vv[i] * 3.28 << setw(12)
       << dv[i] * 3.28 << setw(14) << vh[i] * 3.28 << setw(12)
       << dh[i] * 3.28 << endl;
}
                        //Write data in table form to output file.
  ofstream output;                      //set up output stream object
  output.open(FILE_OUT, ios::out);
  output.precision(1);
  output.setf(ios::showpoint | ios::fixed);
  output << setw(9) << v0 * 3.28 * mph << setw(9) << angle << setw(9) << vv[0]
       << setw(9) << setw(9) << vh[0] << setw(9) << dv[0] << "10";
  output << "\n\nTeam 035 Manzano HS Paintball Trajectory" << endl << endl;
```

```
output << "Table: 2-D Trajectory without Drag (pbtraj2.cpp)"

     << endl;

output.setf(ios::showpoint | ios::fixed);

output << endl << endl;

output << setw(10) << "time(sec)" << setw(14) << "vv(ft/s)"

     << setw(12) << "dv(ft)" << setw(14) << "vh(ft/s)"

     << setw(12) << "dh(ft)" << endl;

for(i=0; i < 10; i++)

{
  output << setw(10) << t[i] << setw(14) << vv[i] * 3.28 << setw(12)
         << dv[i] *3.28 << setw(14) << vh[i] * 3.28 << setw(12)
              << dh[i] * 3.28 << endl;

 }

  //Now write data in sigle column for to output file.
  // Then import sing. column into spreadsheet, cut & paste into pairs of columns to
graph results.

  output << setw(10) << "Begin" << endl << setw(10) << "Time" << endl
              << setw(10) << "(sec)" << endl;

for(i=0; i < 10; i++)
```

```
output << t[i] << endl;

output << setw(10) << "Vertical" << endl << setw(10) << "Velocity" << endl
     << setw(10) << "(ft/s)  " << endl;

 for(i=0; i < 10; i++)

output << vv[i] * 3.28 << endl;

output << setw(10) << "Vertical" << endl << setw(10) << "Distance" << endl
                                             << setw(10) << "(ft)   " << endl;

for(i=0; i < 10; i++)

output << dv[i] * 3.28 << endl;

output << setw(10) << "Horizontal" << endl << setw(10) << "Velocity << endl

     << setw(10) << "(ft/s)" << endl;

 for(i=0; i < 10; i++)

 output << vh[i] * 3.28 << endl;

output << setw(10) << "Horizontal" << endl << setw(10) << "Distance" << endl
                                               << setw(10) << "(ft)" << endl;

for(i=0; i < 10; i++)

output << dh[i] * 3.28 << endl;
```

```
//---------------------------------------------------------------------------------------
//---------------------------------------------------------------------------------------
//--------------------------------DRAG FORCE ADDED-------------------------------------
----

   cout << endl << "WITH DRAG FORCE ADDED" << endl;

   cout.precision(1);

   cout.setf(ios::showpoint |ios::fixed);

   cout << endl << endl;

   cout << setw(10) << "time(sec)" << setw(14) << "vvwd(ft/s)"

       << setw(12) << "dvwd(ft)" << setw(14) << "vhwd(ft/s)"

     << setw(12) << "dhwd(ft)" << endl;

   for(i=0; i < 10; i++)

   {

     cout << setw(10) << t[i] << setw(14) << vvwd[i] * 3.28 << setw(12)

         << dvwd[i] * 3.28 << setw(14) << vhwd[i] * 3.28 << setw(12)

         << dhwd[i] * 3.28 << endl;
   }
```

```
//Write in table form for output file

output << setw(10) << "Vertical" << endl << setw(10) << "Velocity" << endl
       << setw(10) << "(ft/s)" << endl;



for (i=0; i < maxnum; i++)

    output << vvwd[i] * 3.28 << endl;

    output << setw(10) << "Vertical" << endl << setw(10) << "Distance" << endl
           << setw(10) << "(ft)" << endl;

for (i=0; i < maxnum; i++)

    output << vvwd[i] * 3.28 << endl;

    output << setw(10) << "Horizontal" << endl << setw(10) << "Velocity" << endl
           << setw(10) << "(ft/s)" << endl;

for (i=0; i < maxnum; i++)

    output << vvwd[i] * 3.28 << endl;

    output << setw(10) << "Horizontal" << endl << setw(10) << "Distance" << endl
           << setw(10) << "(ft)" << endl;

for (i=0; i<maxnum; i++)

   output << dhwd[i] * 3.28 << endl;
```

```
output.close();

cout << "\nEnd of Data" << endl << endl;

}

//----------------------------------------End of Output_dv-------------------------
```