Simple Wing Simulator

New Mexico Adventures in

Supercomputing Challenge

Final Report

March 16, 2004

Team 004

Alamogordo High School

Team Members

Weston Pendergrass

Daniel Travis

Matt Dodson

Shane Hardin

Teacher

Mr. Albert Simon

Project Mentor

Thomas Hainline

Table of Contents

EXECUTIVE SUMMARY-Our problem was to design a simple program written in Java that would allow the user to input how much lift the wing provided, how much thrust the engine used, how much weight is onboard, and how big the wingspan is and the end result is to tell the user whether or not the plane would fly. Our current program, the second one we have made, now has the user input mass, acceleration, wing width, and wing length. The reason for choosing this problem was that our original Team Leader, Daniel Travis, chose it for the challenges we would face, such as finding the information and completing the program from scratch. We did not find anything else our team would agree on, so we used this problem. The program would have gravity pre-programmed to 9.86 m/s. The program would take the user input values, mass, acceleration, wing width, and wing length, and the preprogrammed gravity value, then would tell the user whether or not your plane would get off the ground in a perfect-world scenario. There would be no storms, no crosswinds, headwinds, tailwinds, or any catastrophic events. A perfect world scenario is a situation where there are no elements that are unwanted, such as storms, no wind, or catastrophic events. All of the elements are created by the person doing the testing. An example would be a wind tunnel used to test airflow over wings.

The program would use graphical elements to allow the user to input the data and to display the results. We used a Java applet create the graphical part of the program. Our equations used in our current program were found online. See Appendix A for the program. Right now, we have a basic, working program that accomplishes almost everything we had hoped for. Our new program is much more seemingly realistic to us than the original. Given more time, we would continue to make our program more complex and accurate. We would also test it in the real world. We used Java instead of C++ for the graphical features, a decision discussed later.

MAIN BODY

Introduction-Our problem was chosen because it concerned our team in the fact that we live very close to Holloman Air Force Base. We also could not come up with any other interesting ideas, so we picked a wing simulator. Our class would be learning the Java programming language and we decided to use that rather than teach ourselves C++ or another programming language on our own. We did not have the time and Java gave us much better graphical options. Java is a programming language very similar to C++. However, it is much newer, is much more universal, and more complex than C++. We created our program as a Java Applet. A Java Applet is essentially a Java program that uses a new window to display graphical objects and manipulate them. This helped us create the GUI, or graphical user interface, of the program. A GUI is a graphical input/output system that the user can use by controlling the mouse and/or the keyboard. It is similar to the desktop on your computer, or the web browser used to surf the internet.

The program focuses right now on the four forces acting on a plane, weight, lift, drag, and thrust. Weight is the downward force on the plane. Lift is the force trying to 'push' the plane higher. Drag is the force that slows down the plane. Thrust is the force that speeds up the plane. Because of Newton's First Law of Motion, ordinarily these four forces cancel each other out, resulting in the plane standing still. Newton's First Law of Motion is: Any object in a state of motion tends to remain in that state of motion unless an external force is applied on it. The objective is to unbalance these forces by creating thrust to overcome drag and to create lift to overcome weight. Our program calculates these forces and tells whether the plane will fly.

Description of the Method to Solve the Problem- We decided that to solve the problem of telling someone whether their valuable plane will fly or not, we needed to find already made equations. We did this by searching online for them. At first, we did not find them. We made a

trial program by using estimated and simple calculations. We had the user input every basic variable. They were: lift, thrust, drag, and weight. We took these four values and created two overall values, the overall negative value and the overall positive value. The overall positive value was the combined total of lift and thrust and was called the positive value because it represented the positive forces acting on the plane. The overall negative value was the combined total of drag, weight, and the value of gravity and was called the negative value because it represented the negative forces acting on the plane. Then, we compared the two totals. If the positive value was greater than the negative value, the plane would fly. Otherwise, the plane would not even leave the ground. At our pre-evaluation, the judges suggested that we enhance our project. Afterwards, we went online and searched more diligently. We did find several more equations we needed. We simply researched all the elements in each equation and integrated them into the program. Now that we had better equations, we tested the program again. We got much better results, which are discussed later in this report. We have not been able to test these results in reality.

Results-We found in the original program, that basic equations were not very accurate. We simply didn't feel that our program was very complete or accurate. However, after finding more accurate equations online at http://web.mit.edu/16.00/www/aec/flight.html and http://hypertextbook.com/facts/; we had a program that seems much more accurate to us. We also found that in order to make a plane fly, it takes a lot of study, testing, and knowledge on the subject. This project has opened our eyes to the world of flight.

Conclusions-We have concluded that making a program to *accurately* tell a user if a plane will fly or not based on the variables input and the variables built into the program is possible, yet difficult. There are many factors involved and variables to manipulate. Integrating

them into a program was difficult because, it was difficult to keep track of all the variables.. Also, you have to understand math and the equations governing flight. Learning them was every bit just as difficult as any day in math class. After research and hard work, it was possible to make a program that tells the user if a plane flies when they input wing length, wing width, mass, and acceleration.

Recommendations-We would recommend that our program still needs more work done to it, and anyone who wishes to enhance it should speak to us. We think that the practical use side of this program would be, for instance: a pilot that needed a quick weight check, a student that wanted to play around with mechanics of flight, and as a training tool for the US Air Force. Of course, before that could happen, the program would need more work done to it such as finding and researching more equations, adding more input and output to the program, making it more dynamic. We would like to think that we commented the program well enough for anyone who understands math and Java could follow how our program works, but that also needs some work.

Significant Original Achievement-Our achievement in originality came from looking through past projects and finding that everyone else's had just done how flight affects objects and such. One of the teams in our class was doing another program on flight; however, theirs deals with weight and balance, where as ours deals with the problem of: will the plane fly. Our program is simple and will tell the user whether or not the plane will fly as opposed to more complex programs dealing with how something will fly.

Acknowledgements-We would like to thank Thomas Hainline for being our mentor and helping us with the programming and some of the math for our program. We would like to thank Joseph Hardin for helping overview our project. All of the websites that we used for mathematics are found in Appendix B.

APPENDICES

Appendix A: Program-

```
import java.applet.*;                 //imports all of the applet packages
import java.awt.*;                    //imports all of the awt packages
import javax.swing.JOptionPane;       //imports the JOptionPane extension packages


public class FlightSim extends Applet{   //begins the applet class FlightSim
   {
   String mass;                  //user input of total mass of the plane
   String acceleration;          //user input of the acceleration of the plane
   String winglength;            //user input of wing length
   String wingwidth;             //user input of wing width
   double massValue;             //value of mass
   double gravityValue;          //value of gravity
   double accelerationValue;     //value of acceleration
   double thrust;                //thrust value
   double dragconstant;          //value of drag constant
   double drag;                  //value of drag
   double lift;                  //value of lift
   double weight;                //the weight in double
   double winglengthValue;       //length of the wing in double format
   double wingwidthValue;        //width of the wing in double format
   double densityofair;          //density of dry air at sea level
```

```
    double wingpressure;           //pressure on the wing at sea level
    double liftconstant1;          //first lift constant
    double wingarea;               //wing area
    double pressure;               //pressure on the wing

    mass = JOptionPane.showInputDialog( //asks the user to input the mass
       "Enter the total mass of the plane in kilograms:" );
    acceleration = JOptionPane.showInputDialog( //asks the user to input the acceleration
       "Enter the acceleration provided by the engines in meters/second/second:" );
//meters/second/second is the metric unit of measurement for acceleration
    winglength = JOptionPane.showInputDialog(//asks user to input the wing length
       "Enter the length of the wing in meters:" );
    wingwidth = JOptionPane.showInputDialog(//asks user to input the wing width
       "Enter the width of the wing in meters:" );

    massValue = Double.parseDouble( mass ); //converts the string mass into a double
    accelerationValue = Double.parseDouble( acceleration ); //converts the string acceleration into
a double
    gravityValue = 9.86;  //sets the gravityValue, this is in meters/second/second on earth. Our
physical science teacher told us that this is what the measure of gravity on earth was
    winglengthValue = Double.parseDouble( winglength );//converts the string winglength into a
double which is the LENGTH of the wing
```

```
    wingwidthValue = Double.parseDouble( wingwidth );//converts the string wingwidth into a
double which is the WIDTH of the wing
    densityofair = 1.2929; //density of dry air at sea level measured in kg/meters cubed found this
value online
    weight = massValue * gravityValue;  //calculates the weight of the plane
    thrust = massValue * accelerationValue; //calculates the thrust on the plane
    wingarea = winglengthValue * wingwidthValue;//calculates the wing area
    pressure = densityofair / wingarea; //calculates the pressure on the wing
    liftconstant1 = pressure * thrust * wingarea;//calculates the lift constant for the equation of lift
    lift = liftconstant1 * ( .5 * pressure * ( thrust * thrust ) ) * wingarea;//calculates lift, .5 comes
from the equation online, the original said ½ and java does not understand fractions, replaced it
with .5
    dragconstant = pressure * thrust * wingarea;//calculates the drag constant for drag equation
    drag = dragconstant * ( .5 * pressure * ( thrust * thrust ) ) * wingarea; //approximates drag

    if ( lift <= drag && thrust <= weight ) {/*begin the if/else structure if both of the negative
forces
                                    equal or outweigh the positive forces*/
        JOptionPane.showMessageDialog(      //display the message that the plane won't fly
            null, "The plane will not fly.", "Results",
            JOptionPane.WARNING_MESSAGE );
    }
```

```
   else if ( lift<= drag || thrust <= weight ){//if the either of the plane's negative forces equal or
outweight, then
                                        //display the message the the plane won't fly
      JOptionPane.showMessageDialog(
         null, "The plane will not fly.", "Results",
         JOptionPane.INFORMATION_MESSAGE );
   }
   else {
      JOptionPane.showMessageDialog(//if the other two tests are failed, then the plane should
fly
         null, "The plane should fly.", "Results",
         JOptionPane.INFORMATION_MESSAGE );
   }                                    //end the if/else structure
   System.exit( 0 );                    //end the program
   }
}
```

Appendix B: Reference List-

1. Glenn Elert and his students, “The Physics Fact Book”, http://hypertextbook.com/facts/ Under Topics Index, Density, Density of Air, Whole Page, 1987.
2. MIT Department of Aeronautics and Astronautics, “Theory Of Flight”, http://web.mit.edu/16.00/www/aec/flight.html, March 16, 1997, Whole Page.

Appendix C: Equations Used-

1. lift constant = p * A * V used in lift, where p is pressure, A is area of flow, V is velocity
2. lift = C_L * (½ p * V^2) * S where C_L is lift constant from above, p is pressure, V is velocity, S is wing area
3. drag constant = p * A * V used in drag, where p is pressure, A is Area of flow, V is velocity
4. drag = C_D * (½ p * V^2) * A where p is pressure, A is velocity, and C_D is the drag constant from above