Air Traffic Control: The Next Step!

New Mexico Supercomputing Challenge
Final Report
April 4, 2012

Team 83
Manzano High School

Team Members:
Tommy Soudachanh
Khiem Tang
Ian Wesselkamper

Teacher:
Steve Schum

Executive Summary

Problem Definition:

Air Traffic Control is a process that takes many people communicating, and thinking to prevent aircraft collisions, and keep planes on time if possible. There are many mistakes on both ground, and in air. Planes have been miss tracked, misplaced, and often are delayed due to slight mistakes that could have been avoided. Although there are many people who work in air traffic control, many mistakes are made as they try to manage air traffic. Also human controlled air traffic controls systems are slow, and you can't use the airport to its full potential.

Problem Solution:

Our goal is to reduce the amount of human error by creating a program that efficiently directs airplanes at a medium sized airport. With this program, the risk of human error will decrease for managing air traffic on and off runways and in result, increase the safety of air travel and maximize the capability of an airport. The program will be based on the layout of a medium sized airport, but can easily be adaptable to other airports, big or medium or small. Eventually we will have a working simulation of an airport, with different problems like weather delays, pilot error, emergencies, and other things that an airport would confront.

Table of Contents

Introduction

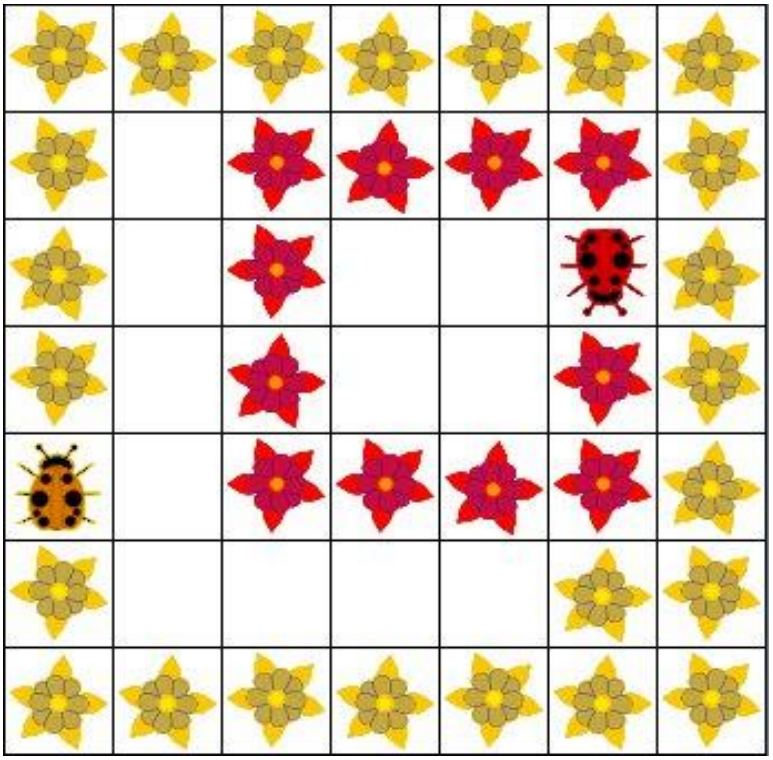
Although there are many people who work in air traffic control, many mistakes are made as they try to manage air traffic. Our goal is to reduce the amount of human error by creating a program that efficiently directs airplanes at a medium sized airport. With this program, the risk of human error will decrease for managing air traffic on and off runways and in result, increase the safety of air travel and maximize the capability of an airport. The program will be based on the layout of a medium sized airport, but can easily be adaptable to other airports.

Java Knowledge

We incorporated multiple aspects and concepts of programming for our project. In terms of object-oriented programming, classes, objects, class accessibility, and instantiating were concepts we used in programming. In addition, we used the KeyboardReader, JOptionPane, ActionListener, and ItemListener input methods and JOptionPane, JFrame, JTextArea, and JApplet output methods to satisfy the requirements needed for our program. Knowledge on loops (if-else, for, while, nested), Garbage Collecting, SDLC, and Debugging were needed as well in order to fully complete the program.

In terms of creating the outlook and flight environment, we incorporated the GridWorld program, developed by the College Board that is used in tandem to teach Java in AP Computer Science. This will allow us to create a field for the objects in our program, showing how the flight program will physically work and calculate.

- Uses a two dimensional array to output an area for critters to interact in:



Program

The program utilizes the basic physics formulas, such as velocity and acceleration, in order to calculate the speed and time needed to land an incoming plane that is five miles away. We plan to be able to use a modified version of A* path-finding to track and guide planes in air and on the ground, and we hope to make the program flexible enough to use any airport with only minor changes to the setup of the airport and software.

Data

Time	Distance	Velocity
(seconds)	(miles)	(mph)
0.0	0.0	300.00000000000006
10.0	0.8096851046198468	282.97327532628964
20.0	1.5720737518127204	265.9465506525793
30.0	2.2871659415786207	248.9198259788689
40.0	2.9549616739175475	231.89310130515852
50.0	3.575460948829501	214.86637663144813
60.0	4.148663766314481	197.83965195773777
70.0	4.674570126372489	180.81292728402735
80.0	5.153180029003522	163.786202610317
90.0	5.584493474207583	146.75947793660663
100.0	5.96851046198467	129.73275326289624
110.0	6.305230992334785	112.70602858918583
120.0	6.594655065257926	95.67930391547544
130.0	6.836782680754093	78.65257924176508
140.0	7.0316138388232865	61.6258545680547
150.0	7.179148539465507	44.59912989434432
160.0	7.279386782680756	27.572405220633943
170.0	7.332328568469029	10.545680546923567

∉ Inputs were 300 and 100

∉ Data has been formatted from origin due to length and logical errors

∉ Data has not been implemented in further usages in the program due to errors

Conclusion

All experimental data has not been collected yet in order to develop a complete conclusion.

References

- Equations
    - David J Lilja, “Measuring Computer Performance: A Practitioner’s Guide”, Cambridge University Press, 2000.
    - Deitel, Harvey M. Java How to Program , Fifth Edition. Prentice Hall, 2002
    - Edwards, Bruce H., Robert P. Hostetler, and Ron Larson. Calculus of a Single Variable. Boston: Houghton Mifflin Company, 2006
    - Gargenta, Aleksandar. "Java Fundamentals Tutorial." Marakana. 2005. Marakana Inc. 1, Oct. 2011 <http://marakana.com/bookshelf/java_fundamentals_ tutorial/index.html>
    - Giancoli, Douglas, C. Physics: Sixth Edition. Upper Saddle River: Pearson Education, Inc., 2005
    - Horstmann, Cay. "Gridworld Case Study." 2006. College Board. 25, Sept. 2011 <http://apcentral.collegeboard.com/apc/public/courses/ teachers_corner/151155.html>
    - Johnston, Barbara. Java Programming Today. Prentice Hill, 2004.
        - Lester, Patrick. "A* Pathfinding for Beginners." Almanac of Policy Issues. 2004. Almanac of Policy Issues. 15, Oct. 2011 <http://www.policyalmanac.org/games/aStarTutorial.htm>
    - Scott, Jeff. "Airliner Takeoff Speeds." Aerospaceweb. 2002. The Aircraft Museum. 20, Dec. 2011 <http://www.aerospaceweb.org/question/performance/ q0088.shtml>

Source Code

Working Code:

```
import javax.swing.*;
public class Airplane {
  int i; // int = integer variable type (+9,0,-133)
  int ttotalA=120, ttotalB=240; // ; ends statements.
  double aA=-0.761, aB=-2.00; // aA=accel before touchdown
  // aB=accel after touchdown
  // Variables for landing a plane Part A and Part B
  double time, ttemp; /*float = real number variable type with 8 chars max 1234567.
  1.000001 0.000009 including the decimal point */
  double vai; // vai = initial velocity part A at 5.0 miles before
  double va; // va = velocity in part A every 10 seconds
  double vbi; // vbi = initial velocity part B at 0.5 miles into runway
  double vb; // va = velocity in part B every 10 seconds
  double dai; // vai = initial position part A at 5.0 miles before
  double da; // va = position in part A every 10 seconds
  double dbi; // vbi = initial position part B at 0.5 miles into runway
  double db; // vb = position in part B every 10 seconds
  // Variables for an outbound plane Part C and Part D
  double vci; // vci = initial velocity part A at 5.0 miles before
  double vc; // vc = velocity in part A every 10 seconds
  double vdi; // vdi = initial velocity part B at 0.5 miles into runway
  double vd; // vd = velocity in part B every 10 seconds
  double dci; // vci = initial position part A at 5.0 miles before
  double dc; // vc = position in part A every 10 seconds
  double ddi; // vdi = initial position part B at 0.5 miles into runway
  double dd; // vd = position in part B every 10 seconds
  JFrame frame = new JFrame("final outputs");
  JTextArea textArea = new JTextArea();
  String area = "";
  JScrollPane pane = new JScrollPane(textArea);

  public Airplane() {
  //const float x=value; Declare a constant for a given scope of the program.
  //cout.precision(3); // 3 = 3 digits past the decimal point
  //cout.setf(ios::showpoint | ios::fixed);
    String temp =JOptionPane.showInputDialog("Enter the initial velocity of an incoming
plane 5 miles away:"); // We entered 300 mph
    vai = Double.parseDouble(temp);
    vai = vai * 1609 / 3600; // Converts mph to m/s
    temp=JOptionPane.showInputDialog("Enter touchdown velocity of an incoming
plane.");// We entered 150 mph
    vbi = Double.parseDouble(temp);
    vbi = vbi * 1609 / 3600; // Converts mph to m/s
    System.out.println("the starting distance from runway is 5.00 miles away.");
    dai = 0.0; // We set dai = 0 = initial position
    area = "\t\t" + "Time" +"\t\t" + "Distance" +"\t\t\t" + "Velocity" +"\n";
    area += "\t\t" + "(seconds)" +"\t\t" + "(miles)"+"\t\t\t" + "(mph)" +"\n";
    for(i=0; i<=ttotalA; i++) // i++ means i = i + 1
    {
      ttemp=(float)i * 10; // Trick: type cast int i to a real number as float.
      da=dai + vai * ttemp +.5 * aA * Math.pow(ttemp,2); // pow = power 2 = 2nd order
power
      da = da / 1609; // Converts meters to miles
      va=vai+aA*ttemp;
      va = va * 3600 / 1609; // converts back from m/s to mph
      area += "\t\t" + ttemp +"\t\t" + da + "\t\t\t" + va + "\n" ;
    }
    textArea.setText(area);
    frame.setSize(500,500);
    frame.add(pane);
    frame.setVisible(true);
  }
  public static void main(String[] args) {
    Airplane airplane = new Airplane();
  }
}
```

Non-Working Code:

```
//no package
import java.util.ArrayList;
//in progress last edited january 9 2012

public class AircraftController {
        private int ctime = (int) System.currentTimeMillis(), otime = (int)
System.currentTimeMillis();//ctime is current time otime is old time
        private boolean isRunning = true;
        private ArrayList<Aircraft> airCraft;//this is for doing actions to the planes
        private ArrayList<Aircraft> removeList;//this is for cueing planes for deletion
        public AirCraftController() {
                airCraft = new ArrayList<Aircraft>();
                removeList = new ArrayList<Aircraft>();
        }

        public voidAirCraftLoop() {
                while(isRunning) {
                        ctime = System.currentTimeMillis();
                        //this pretty bit of code adds and removes old and new planes
                        ArrayList<Aircraft> AddList = RecievePlanes();
                        for(adding:AddList) {
```

```
                airCraft.add(adding);
            }
            airCraft.removeAll(removeList);
            removeList.clear();

            for(airCraft AirPlane:airCraft) {
                //error handling needs to be re written
                //take off handling
                if(AirPlane.isTakingOff()) {
                    if(checkPath(AirPlane)) {//this will need an extra
value for handling the size to check for and the path its going to go on
                        //checkPath is going to just check to see if
theres any collisions in the near future
                        AirPlane.TakeOffApprove();
                    }else{AirPlane.hold();
                }
                if(AirPlane.isLanding()) {
                    //uhhh
                    //not sure if i should throw code for landing to a
later section
                }
                if(AirPlane.isLeaving()) {
                    //do code related to heading to a new airport
                }

            }

            otime = ctime;
        }
    }

    public ArrayList<Aircraft> RecievePlanes() {
        ArrayList<Aircraft> recieving = new ArrayList<Aircraft>();//just blank
for now
        //for now this will remain blank, later it will be input by user
        //unlikely future plan, be gotten for a central server
        return recieving;
    }

    public void addAircraft(Aircraft aircraft) {
        airCraft.add(aircraft);
    }

    public airCraft getAirCraft(int number) {
        return airCraft.get(number);
    }
```

```
        public static void main(String[] args) {
                AircraftController controller = new AircraftController();
                controller.AirCraftLoop();
        }
}
```

Acknowledgements

- Adams, et al (1998) C++ An Introduction to Computing, 2nd Ed, Prentice Hall
- Albuquerque Academy-Team 7 (2008) "Modernizing the U.S. Air Traffic Control System", NMSCC
- Jamsa, Kris (1996), Rescued by C++; 2nd Ed, Jamsa Press
  Scott, Jeff (2002), "Airliner Takeoff Speeds", URL: http://www.aerospaceweb.org/question/performance/q0088.shtml
  Zitzewitz, et al (2005), Physics Principles and Problems, Glencoe
- Thank you to Mr. Schum for his time and knowledge
- Thank you to the NMSSC program for getting us started with computer programming
- Thank you to UNM and the Evaluators for their efforts