Efficient Air Traffic Control

New Mexico
Supercomputing Challenge
Final Report
April 7, 2010

Team 127
Manzano High School

Team Members

Ryan Hensel
Elisabeth Keller
Jelke Adema

Teacher Sponsor
Steve Schum

Table Of Contents

Executive Summary

Our team intended to create a program that eliminates the issue of human error from the issue of air traffic control. Without human error the process of air traffic will be completely automated, letting the machine and/or program think entirely on its own. If the said issue is automated, then the program that controls the issue is able to make a logical decision, and the only thing air traffic control requires is logic. A program that is programmed to not make any interfering decisions is required for air traffic control; if the program were to cross take off times, it could result in a crash. We are trying to prevent any mishap to happen on the runway. This is ultimately our goal, to prevent any thing that is not supposed to happen from happening. We wish to write a computer model to simulate handling airplane traffic in a midsize airport. We are going to use the C++ programming language.

Phase I: Develop mathematical model to land five commercial jets and take off five commercial jets on one runway in one hour. Our adviser has taught us the physics equations and we have already computed the position, speed, and acceleration of one jet while landing. We have computed the position, speed, acceleration of one jet on takeoff.

Phase II: Develop mathematical model to land five private planes and take off five private planes on a shorter runway that parallels the jet runway in one hour. We have computed the position, speed, acceleration of one plane on takeoff on a 2nd shorter runway that parallels the commercial jet runway, also for a one hour time frame.

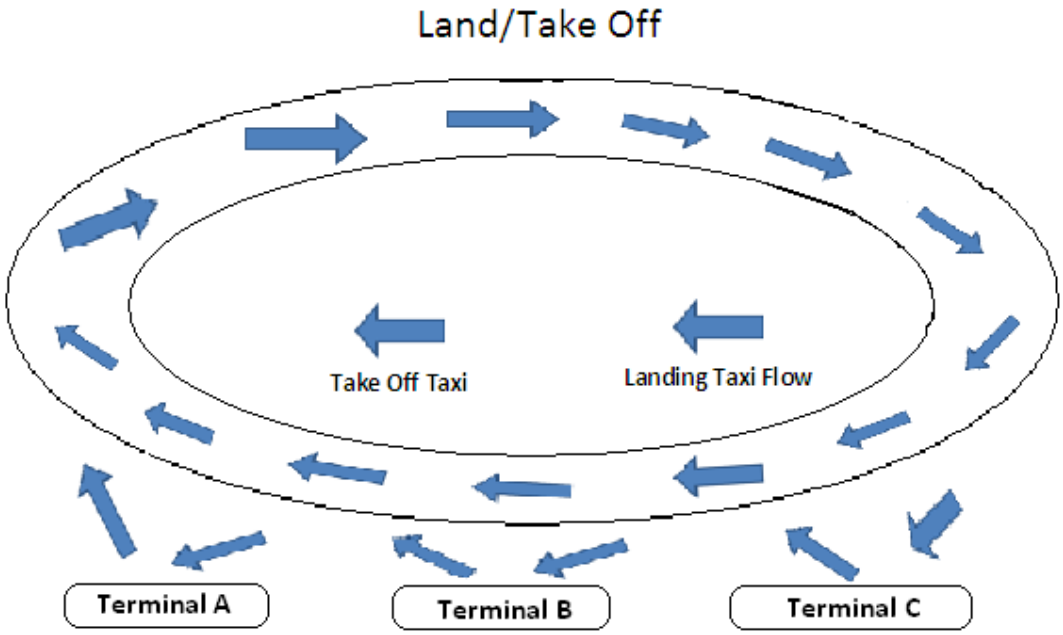
Phase III-A: Our team wrote a computer model to simulate handling airplane traffic in midsize airport based on the mathematical handle to land five small private planes and take off five small private planes on a 2nd shorter runway that parallels the commercial jet runway, also for a one hour time frame. We will then develop the C++ computer program to perform all these calculations and output time, position, speed, and acceleration for all the planes in ten second time intervals to a neatly formatted data table.

Phase III-B: Then we want to modify the C++ program to include decisions and communications with the airplanes based on normal air traffic scenarios, as well as to accommodate disruptions in the traffic pattern such as flat tires, flocks of birds on the runway, accidents, weather conditions, etc.

Introduction

We wish to write a computer model using the C++ programming language to simulate the handling of commercial airplane traffic in a midsize airport.

Sample Airport Traffic Flow



Mathematical Model

In Phase I, we developed a mathematical model to compute the position, speed, and acceleration of one jet on landing and one on takeoff. Then in Phase II, we expanded our mathematical model to land five commercial jets and send off five commercial jets on one runway in one hour. In Phase III, we plan to expand our mathematical model to include traffic decisions to be communicated to the aircraft based on normal and disrupted traffic flow.

Math Variables and Symbols For Acceleration Physics Formulas

- d_i = initial horizontal position (m), (where d_i is 5.0 miles before the runway)
- d = horizontal displacement (m)
- v_i = initial horizontal velocity (m/s)

- v = horizontal velocity (m/s)
- t = time (s)
- a = acceleration (m/s^2)

Physics Acceleration Formulas

- $a = (v - v_0) / (t - t_0)$
- $v^2 = v_i^2 + 2 (a) (x - x_0)$ → Then solve for acceleration (a) both before touchdown and after touchdown when landing and takeoff.
- Then substitute for v_0 and a, and solve for x in $d = d_i + (v_i)(t) + ½ (a) (t^2)$ for landing
- Then substitute for v_0 and a, and solve for v in $v = v_i + (a)(t)$ for landing and takeoff.
- Use these same equations to compute & output position vs time and speed vs time for both inbound and outbound planes in 6 min intervals.

Plan For C++ Program

We developed a C++ computer program to compute and output the position, speed, acceleration of one jet on landing and one on takeoff. Expand C++ computer program to output results of landing 5 commercial jets and sending off five commercial jets on one runway in one hour. Expand computer model to include traffic decisions to be communicated to the aircraft based on normal and disrupted traffic flow.

Phase I

Compute and output position vs. time, speed vs. time for one inbound plane beginning 5.0 miles from runway and ending at end of runway in < 4.0 minutes. Compute and output position vs. time, speed vs. time for one outbound plane beginning at start of runway and ending 5.0 miles beyond end of runway.

Phase II

Write “for loop” to alternate above calculations for inbound and outbound air traffic starting inbound landing at 0, 12, 24, 36, 48 minutes and starting outbound traffic at 6, 18, 30, 42, 54 minutes.

Phase III

We will add the communications to every inbound and outbound plane at key time intervals for normal traffic flow. We will add the communications to every inbound and outbound plane for “alerts”, “holding patterns” and “alternate landing and takeoff times” during disrupted traffic flow. For normal traffic flow, write code to communicate traffic instructions to inbound and outbound air traffic -- starting inbound landing at 0, 12, 24, 36, 48 minutes and starting outbound traffic at 6, 18, 30, 42, 54 minutes. For disrupted traffic flow, write code to communicate to planes the “alerts”, “holding patterns” and “alternate landing and takeoff times” for both inbound and outbound air traffic.

Results and Conclusions

In Phase I, our C++ model computes and outputs position vs. time and speed vs. time for one inbound commercial jet in a 6 min interval. We are in the final stages of debugging this program and generating tables of output, both to the display and to a spreadsheet formatted file. We still need to complete the Phase I code to compute and output position vs. time and speed vs. time for one outbound commercial jet.

Then we need to expand the C++ code for Phase II to run five twelve minute cycles per hour for alternating inbound and outbound commercial jets. Finally we need to expand the program for Phase III to include communications to all jets during normal traffic flow and communications and alerts to all jets during disrupted traffic flow/events. We hope to complete the computer code and formatted output for Phases I, II, and III by April 26 to present at the NMSCC 2010 Expo.

References

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

Acknowledgements

Team 127 would like to acknowledge the NM Supercomputing Challenge Program for 2009-2010. We also wish to thank Mr. Stephen Schum for teaching the Pre-Engineering Electronics program here at Manzano High School and for being our teacher sponsor throughout the duration of our NMSCC project. We would also like to give a special thanks to the MHS Administration for supporting the Pre-Engineering Electronics class and the NM Supercomputing Challenge Program. Finally we wish to acknowledge the Kyle Kortkamp Memorial Fund for donating $800 to the Manzano High School Pre-Engineering Electronics and NM Supercomputing Challenge programs.

Appendix

C++ Code

```
/* filename: landplane1.cpp    Apr-6-2010
  Team 127 in 2009-10 NMSCC
  Phase I of this program will show the position & velocity every 10 seconds
     of planeslanding and taking off for 1.0 hour at a mid-sized airport.
  First landplane1.cpp asks the user for initial velocity of an incoming plane
     starting 5.0 miles from the start of the runway.
  Then landplane1.cpp asks the user for touch down velocity of an incoming plane
```

```
      starting 0.5 miles on to the runway.
   Then landplane1.cpp will compute the position and velocity every 10 sec during
      the landing of the plane.
   Then landplane1.cpp will use similar formulas to compute the position and velocity
      every 10 sec during take-off of an out-going plane until it reaches the
      outer 5.0 mile mark.
*/


// C++ brings other programs to your program via #include <filename> syntax


#include <iostream.h>   // Allows input and output to screen and/or file.
#include <math.h>       // Allows math functions (power, sine, cos, tan, etc)
#include <iomanip.h>    // Allows program to manipulate data.


int main(void)              // Starts main program.  int = Return an integer
{                           // Put {} around code segments. {} must match.
  int i;              // int = integer variable type (+9,0,-133)
  int ttotalA=120, ttotalB=240;                    // ; ends statements.
  float aA=-0.761, aB=-2.00;    // aA=accel before touchdown
                                // aB=accel after touchdown
```

```
                    // Variables for landing a plane Part A and Part B
float time, ttemp;  /*float = real number variable type with 8 chars max 1234567.
                            1.000001  0.000009 including the decimal point   */

float vai;          // vai = initial velocity part A at 5.0 miles before
float va;           // va = velocity in part A every 10 seconds
float vbi;          // vbi = initial velocity part B at 0.5 miles into runway
float vb;           // va = velocity in part B every 10 seconds

float dai;          // vai = initial position part A at 5.0 miles before
float da;           // va = position in part A every 10 seconds
float dbi;          // vbi = initial position part B at 0.5 miles into runway
float db;           // vb = position in part B every 10 seconds

                    // Variables for an outbound plane Part C and Part D
float vci;          // vci = initial velocity part A at 5.0 miles before
float vc;           // vc = velocity in part A every 10 seconds
float vdi;          // vdi = initial velocity part B at 0.5 miles into runway
```

```
 float vd;                // vd = velocity in part B every 10 seconds


 float dci;               // vci = initial position part A at 5.0 miles before
 float dc;                // vc = position in part A every 10 seconds
 float ddi;               // vdi = initial position part B at 0.5 miles into runway
 float dd;                // vd = position in part B every 10 seconds



 //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);


/*


cout << "Phase I of this program will show the position & velocity every 10 seconds" << endl
          "  of planeslanding and taking off for 1.0 hour at a mid-sized airport." << endl
          "First landplane1.cpp asks the user for initial velocity of an" << endl
           "incoming plane starting 5.0 miles from the start of the runway." << endl
         "Then landplane1.cpp asks the user for touch down velocity of an"  << endl
```

```
            "incoming plane starting 0.5 miles into the runway." << endl
          " Then landplane1.cpp will compute the position and velocity" << endl
            "every 10 sec during the landing of the plane." << endl
          "Then landplane1.cpp will use similar formulas to compute the" << endl
            "position and velocity every 10 sec during take-off of an" << endl
            "out-going plane until it reaches the outer 5.0 mile mark."<< endl;
*/

  cout << "Enter the initial velocity of an incoming plane 5 miles away:";
  cin >> vai;                                   // We entered 300 mph
  vai = vai * 1609 / 3600;                  //  Converts mph to m/s

  cout << "" << endl;
  cout  << "Enter touchdown velocity of an incoming plane." << endl;
  cin >> vbi;                                                 // We entered 150 mph
  vbi = vbi * 1609 / 3600;                  //  Converts mph to m/s


cout << "the starting distance from runway is 5.00 miles away." << endl;
    dai = 0.0;                                  // We set dai = 0 = initial position
```

```
 cout << setw(12) << "Time" << setw(12) << "Distance" << setw(12) << "Velocity" << endl
      << setw(12) << "(seconds)"  << setw(12) << "(miles)"  << setw(12) << "(mph)" << endl;


 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 * 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
    cout << setw(12) << ttemp << setw(12) << da << setw(12) << va << endl;
   }

 return 0;
}
```