Creation of an Air Traffic Simulation Using Agent Based Modelling

Sam Eberspacher
TJHSST Computer System Lab
2007-2008

May 23, 2008

Abstract

As the skies over the United States become increasingly crowded, airports in the United States are increasingly stressed to adapt to this increased demand. The goal of this project is to visually represent the strain on airports and passengers as a variety of problems generate record delays. By using agent based modelling, along with real air traffic information, this simulation may accurately predict the proliferation of delays through out airports in the United States.

1 Introduction

The purpose of this project is to visually represent the proliferation of a delay throughout a system of airports. By using techiques such as agent based modelling, the simulation will predict actual delays with decent accuracy. Additionally by repeating the simulation multiple times, the simulation generates increasingly accurate results as the number of trials approached infinity. While a simulation such as this would take a human enormous amounts of time, a computer may be able to run a simulation of 24 hours in a matter of minutes. Due to the scale of the problem, efficiency will be key for the computer to run the simulation in a timely matter.

2 Development

2.1 Agent Based Modelling

In order to simulate such a large system, this project will use a technique known as agent based modelling. The deveopment of a system using agent based modelling is key for the success of the project. Each agent must interact with other agents in the system in the most realistic way possible in order to generate the most accurate results. One benefit of the agent based modelling is that parameters for interaction between agents define the overall behaviour of the system. This allows the programmer to work on much smaller problems with the agent in order to alter the overall system.

2.2 Embedded Statistical Analysis

Embedded statistical analysis is a done when real time statistics are needed in a simulation. The program uses data at each time step to readjust statistical values for the desired population. These statistics are useful when determining if the system is able to handle the introduction of new agents or constraints such as weather data.

2.3 Geocoding

Geocoding is a process by which a formatted address such as 6560 Braddock Rd. Alexandria, VA 22312 is converted to a longitude and latitude. This process is important when dealing with map information that is displayed on a computer. The computer is unable to relate formatted addresses so longitudes and latitudes are used to generate accurate relationships about location. This project uses the process to determine the location of each airport and accurately plot the distance between airports.

2.3.1 Google Geocoder

Geocoding is a complex process which involves a significant amount of computing power relative to web requests. Due to these requirements for geocoding many companies charge a small fee per request. Alternativeley, there are some companies which offer geocoding free of charge but wiht limitation on the number of geocoder requests per day. I found that Google offered free geocoding with a maximum of 5000 requests per day.

2.3.2 Request formatting

In order to interact with the Google geocoder, each request was done through an HTTP request sent to Google servers. These servers then interperet the parameters in the URL of the request and return the ouput specified by the user. The parameters in a request are as follows:

- q - The formatted address to be geocoded
- output - The desired output format (xml, kml, csv, or json)
- key - Google Maps API key

Sample Request (Key removed for privacy reasons)

`http://maps.google.com/maps/geo?q=BWI+airport&output=csv&key=API_KEY`

2.4 Software

Computer languages used in this project.

1. Java was used for the bulk of the project including all classes and the display of information.
2. Python was used for interacting with the Google geocoder.

2.5 Procedure

2.5.1 Static Information

To start the project, initial classes were written to display static information. The Aiport class and Simulation class were drawn up and coded, leaving room for additional modifications which would take place at later stages of development. The first step was to normalize the data points in an effort to maximimize used space and increase effeciency. A randomly generated set of coordinates was generated and used for this part of testing. The normalizing equation for used in this project is given by the following equations:

$$x_{norm} = \frac{x_i - x_{min}}{x_{range}} \cdot width_{screen}$$

$$y_{norm} = \frac{y_i - y_{min}}{y_{range}} \cdot height_{screen}$$

2.5.2 Geocoding

The second stage of the process was to geocode the airport locations and provide a list of latitudes and longitudes which would make up the airport map. In order to make the most robust program possible, a Python script was written to automate the geocoding process. The script reads from a list of aiports and formats an address for geocoding. The script then sends the HTTP request to the Google geocoder and parses the returned file. The parsed data is then validated to verfiy that the location is in the United States and written to a file.

The formatting of the address itself was very important to the geocoding process. Since the data abotained did not contain street addresses, a straight address request was not possible. However, it was determined that using the three letter code for the address yielded desirable results. Some airports though, still failed to meet validations standards (due to conflicts with other country codes) so a second format was needed. The aiport code concatenated with "airport" gave the second best results and the Python code was then optimized to maximimze the number of valid airports.

2.5.3 Waypoints and Collision Detection

Due to the large amount of air traffic which is needed to create an accurate model, a waypoint system needed to be implemented to minimize collisions of aircraft. Initially, each plane sets the next waypoint to the destination airport, which would be the shortest travel path. The plane will only modify it's waypoint stack if a collision is detected and a new waypoint is pushed on to the stack.

The process for deconfliction was based on the algorithm developed by students at the Czech Technical University and operates as follows: (Logic diagram goes here)

2.5.4 Path Tracing and Arc Rendering

In order to better visualize the data, both path tracing and arc rendering where included in the program. Path tracing is a process by which previous locations of the aircraft remains rendered to the screen. This allows the user too better visualize where the plane came from as well as the speed and direction of the airplane. Arc renderuing supplements path tracing, by helping the user see where the plane is going. Arc rendering is done by falsely rendering the location of the aircraft on the two dimensional viewing plane to give the illusion of alititude. The progression of the aircraft is based on an arc from the previous waypoint to the next waypoint on the stack. This illusion allows the user to better project where the plane will land.

2.5.5 Embedded Statistics

The implementation of embedded statistics was done mostly through new variables added to the Airport class. Due to the statistical properties of the mean and standard deviation, the mean and standard deviation of the whole simulation could be calculated without polling all of the agents for a second time. The mean and standard deviation were calculated using the following formulas.

$$\mu = \sum_{k=1}^{n} \mu + \mu_k$$

$$\sigma = \sum_{k=1}^{n} \sqrt{\sigma^2 + \sigma_k^2}$$

3 Results

...

References

[1] Koblin, Aaron, cmps. Flight Patterns. University of California at Los Angeles. 21 Nov. 2007 <http://users.design.ucla.edu/ akoblin/work/faa/>.

[2] Tumer, Kagan, and Adrian Agogino, comps. Distributed Agent-Based Air Traffic Flow Management. University of Oregon, UCSC, NASA Ames Research Center. 18 Jan. 2008 <http://web.engr.oregonstate.edu/ ktumer/ktumer-aamas07.pdf>.

[3] Benson, Kirk C., David Goldman, and Amy R. Pritchett, comps. Applying Statistical Control Techniques to Air Traffic Simulation Georgia Institute of Technology. 18 Jan. 2008 <http://portal.acm.org/citation.cfm?id=1161734.1161979>.

[4] ichal Pěchouček and David Šišlák and Dušan Pavlíček and Miroslav Uller Autonomous agents for air-traffic deconfliction. Czech Technical University. 30 March 2008 <http://portal.acm.org/citation.cfm?id=1160633.1160925>.