

AQM SHELL DEVELOPMENT

CREATING A FRAMEWORK FOR AIRSPACE AND AIRFIELD OPERATIONS AND AIR QUALITY VISUALIZATION SOFTWARE

Todd Alan Peterson, EIT

Project and Report submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
Civil Engineering

APPROVED:

Dr. Antonio A. Trani, Chair
Dr. Donald R. Drew
Dr. Antoine G. Hobeika

September 22, 1997
Blacksburg, Virginia

Key words: Airports, Air Traffic, Air Pollution, SIMMOD, Visualization

Copyright 1997, Todd Alan Peterson

AQM SHELL DEVELOPMENT

CREATING A FRAMEWORK FOR AIRSPACE AND AIRFIELD OPERATIONS AND AIR QUALITY VISUALIZATION SOFTWARE

Todd Alan Peterson, EIT

(ABSTRACT)

It is believed that the analysis of air traffic impacts on air quality will benefit from attention to the three-dimensional nature of the air traffic network as well as the actions of individual aircraft during the study period. With the existence of air traffic simulation models, the actions of individual aircraft may already be defined in a simulated environment. SIMMOD, the Federal Aviation Administration's airport and airspace modeling software, performs such models of scheduled air traffic. The results of such models may be used to determine the impacts of scheduled air traffic on air quality as well as other parameters.

This report addresses the interpretation of output from SIMMOD models for use in air quality analysis and visualization of the air traffic network, and the application of these techniques in a stand-alone computer program. This program, named AQM for its purpose in assisting development of Air Quality Models, provides a working framework for future development of software for detailed air quality analysis and visualization.

Recognition and Dedications

Completion of this project and report would not have been possible without the support and guidance provided by my friends, family, and the faculty at Virginia Tech. The following people, in particular, deserve special recognition beyond any words I could use to express my thanks:

Dr. Antonio Trani, for his endless guidance, patience, and generosity during this project and, indeed, my entire college career. I thank him, as well, for sharing and nurturing my zest for programming applications of civil engineering which have culminated in this project.

My family, who served as strongholds in my resolve to see this project through to completion, for being there when I needed a push, and for the gift of my undergraduate college education.

Greg Schultz, who shared the pleasant distraction of the occasional round of golf in between programming sessions and report preparation.

Anderson & Associates who, as my employer, allowed much flexibility in my full-time work schedule during my coursework and throughout development of this project.

Tina Marie Loane who motivated me to finish this project as she endured two years' worth of weekends with me in front of the computer.

Libby, Jerry, and Foster, for unconditional love.

Finally, I give thanks to the many faculty members at Virginia Tech who broadened my exposure to the engineering concepts that have shaped my skills and helped define my career path.

Author's Note

When my fiancé first inquired, “what does writing a computer program have to do with Civil Engineering?” I had a hard time providing an answer. The more I thought about it, however, the more I came to understand the fundamental relationship between the engineering profession and what I was accomplishing with this project.

The public tends to associate the civil engineering profession with “building things”, such as buildings, bridges, dams, and roads. While this is not incorrect, the role of the civil engineer may be more completely described as “an agent in successful design and implementation”. Civil engineering is a profession based on the design and implementation of the physical components and theoretical systems that comprise our civil infrastructure. The engineers’ role in planning and administration is as critical as creating it. Streamlining the planning process by developing tools to assist the engineer is just as important, and this project certainly falls into that category.

The idea for an air-quality postprocessor for SIMMOD was presented by my graduate advisor, Dr. Antonio Trani, as one of several projects that had been considered by the university but never actively pursued. Taking on this project afforded me the opportunity to pursue one of my personal ambitions; that being the development of professional software products that would have useful applications to the industry. My academic experience in airport and airspace simulations (gained under the guidance of Dr. Trani) provides a foundation of engineering knowledge and technical understanding which made this project possible.

Having justified this project to anyone who would ask (including myself), the remainder of this report will serve to describe the development of AQM as it pertains to fulfilling the qualifications for my Masters’ degree, and how the program will provide a foundation for further development beyond the scope of this project. It is my belief that such a program has potential for its use as an empowering tool in the air transportation industry and for mitigating the impacts of air quality in this age of diminishing physical resources.

Todd Alan Peterson
August 15, 1997

Table of Contents

Recognition and Dedications	iii
Author's Note	iv
Table of Contents.....	v
Table of Figures.....	vii
1. Introduction.....	1
2. Project Scope.....	3
2.1. OBJECTIVE.....	3
2.2. SCOPE OF WORK PERFORMED.....	3
3. Methodology: Tools and Techniques	5
3.1. CONCEPTS FOR AIRSPACE MODELING.....	5
3.1.1. <i>Definition of a study region</i>	5
3.1.2. <i>Discrete definition of airspace and time</i>	6
3.1.3. <i>Cumulative effects during simulation period</i>	6
3.2. PROGRAMMING TOOLS	8
3.2.1. <i>Visual C++: The programming language of choice</i>	8
3.2.2. <i>OpenGL: The selected graphics programming standard</i>	9
3.3. SIMMOD AS A SIMULATION DATA SOURCE	9
3.3.1. <i>SIMMOD file formats</i>	9
3.3.2. <i>SIMMOD's definition of the air transportation network</i>	10
3.3.3. <i>Event-specific information generated by SIMMOD</i>	12
3.4. READING SIMMOD DATA PROGRAMMATICALLY	12
3.4.1. <i>General considerations</i>	13
3.4.2. <i>AIRSPACE file</i>	13
3.4.3. <i>AIRFIELD file</i>	15
3.4.4. <i>LOG file</i>	16
3.5. CREATING AQM-SPECIFIC DATA	17
3.5.1. <i>AQMNodes</i>	17
3.5.2. <i>AQMLinks</i>	18
3.5.3. <i>AQMFlights</i>	18
3.5.4. <i>AQMEvents</i>	18
3.6. DEVELOPING THE AQM USER INTERFACE	19
3.6.1. <i>Support for different data types</i>	19
3.6.2. <i>User interface features</i>	20

4.	Result: The AQM Shell.....	24
4.1.	OVERVIEW	24
4.2.	AQM CLASS STRUCTURE	24
4.2.1.	<i>SIMMOD element classes</i>	24
4.2.2.	<i>AQM element classes</i>	25
4.2.3.	<i>Functional classes</i>	25
4.3.	AQM DOCUMENT / VIEW ARCHITECTURE.....	25
4.4.	USING AQM TO INTERPRET SIMMOD PROJECTS.....	26
4.4.1.	<i>Starting from scratch</i>	26
4.4.2.	<i>Generating AQM elements from SIMMOD data</i>	28
4.5.	VIEWING SIMMOD AND AQM DATA.....	30
4.5.1.	<i>Element attribute listings</i>	31
4.5.2.	<i>Network view</i>	31
4.6.	PROGRAM OUTPUT.....	32
5.	Next Steps: Future development of AQM.....	34
5.1.	POLLUTION GENERATION PARAMETERS.....	34
5.2.	DISPERSION MODELS.....	34
5.3.	VISUALIZATION CAPABILITIES.....	34
5.3.1.	<i>On-screen visualizations</i>	35
5.3.2.	<i>Report generation processes</i>	35
5.4.	GENERAL ENHANCEMENTS	36
5.4.1.	<i>Compatibility with later versions of SIMMOD and third-party products</i>	36
5.4.2.	<i>Airport, airspace, and event editing</i>	36
5.4.3.	<i>Support for metric (SI) units</i>	36
6.	Conclusions.....	37
	References	38
	Appendices	39
	Vita	90

Table of Figures

Figure 1 – Incremental modeling of pollutant dispersion	7
Figure 2 - AQM Classes Defined for SIMMOD Data Types	10
Figure 3 – Conversion of SIMMOD data types.....	17
Figure 4 – Blank AQM Document.....	27
Figure 5 – AQM Document following SIMMOD model input	28
Figure 6 – AQM Document with AQM Objects.....	29
Figure 7 – SIMMOD Airspace Node List View	30
Figure 8 – AQM Network View	32
Figure 9 - AQM "Write ASCII Event file" output.....	33

1. Introduction

The catalyst for this project is the desire for visualizations of air quality impacts generated from airport and air traffic models. It is believed that such visualizations, based on discrete actions of individual aircraft, will be helpful in analyzing and mitigating the impacts of air traffic on air quality.

Aircraft operations present a challenge for estimating the pollution they generate. Their actions must be described in three dimensions as opposed to two dimensions or less, as required for ground based pollution generators. They involve many different aircraft with widely varying pollution generation regimes. They are also dynamic – aircraft are effectively point sources of pollution that move in three dimensions. Pollution generation along common flight paths are subject to variations in air traffic demand that depend on season, economy, and location.

To create the most accurate air quality simulation, one must consider pollution generation by individual aircraft as a variable dependent on time, position, aircraft type, environment, and the “flight regime” of aircraft. The flight regime describes the specific actions of each aircraft (i.e. takeoff, cruise, etc.) for applying appropriate pollution generation parameters. This project will demonstrate a framework for generating such aircraft-based information from existing airport and airspace modeling tools. The AQM application is that framework.

AQM is an acronym for “Air Quality Modeling”. AQM is being developed as a stand-alone application to aid in the analysis of pollution-causing activities and to visualize their impact on air quality. To accomplish its purpose, AQM needs to know quite a bit about the air transportation network. It needs to keep track of all aircraft within the study region at any given moment during the study period. It needs to be aware of which ones are creating pollutants (any aircraft in the air or on the ground with an engine running), what type of aircraft they are and what “flight regime” they are in (both used to determine how much pollution they are generating), and where this generation is taking place (given by the positions of individual aircraft in three dimensional space).

This project addresses the question, “How can all this information be obtained for a simulated air transportation network?” This is where the air traffic model becomes the focus of this report.

The airport and airspace simulation model produced by the FAA, called *SIMMOD*, models airport and airspace operations over a given period of time. Given the physical layout of the airfield and airspace and a timetable of scheduled air traffic, SIMMOD is capable of generating a chronological record of specific aircraft events. AQM correlates these events with the location of aircraft in the airport/airspace model. The resulting information may be used for determining pollution generation characteristics of all aircraft within the model at any time during the simulation period. This interpretation of SIMMOD output is a very important element of AQM and constitutes the scope of this project.

2. Project Scope

2.1. Objective

The objective of this project is to develop and present a computer program which interprets air traffic models created using SIMMOD and convert the information contained therein to a format which may be used in modeling and visualizing their impacts on air quality.

The basic tasks that the program must perform are listed as follows:

1. Read physical model elements (airfield and airspace nodes and links, gates, and airfield elevations) from SIMMOD data.
2. Read specific time-based events created by SIMMOD's report post-processing.
3. Correlate events with physical model elements.
4. Create list of events describing all aircraft operations in a time-space domain.
5. Display the physical layout of the airport(s) and corresponding airspace described by the physical model elements in a 3D graphical environment, with point-of-view to be controlled by the user.

These items describe the functionality of the program, named AQM , as created for this project.

2.2. Scope of Work Performed

Efforts focused on two levels of development: conceptual planning and actual program development. First, it was necessary to develop algorithms for interpreting SIMMOD model data. Secondly, implementing these algorithms in a computer program involved significant effort in coding the program. The following section, "Methodology", will describe the tools and techniques used in development of AQM.

Possibilities for future improvements were considered during the course of this project. While these considerations are not directly apparent to the end user, the current version of AQM has features built in to facilitate future improvements to the program. Providing for future expansion of AQM's power and functionality was a governing principal during development, resulting in a program that truly represents an expandable framework for modeling and visualizing air quality.

3. Methodology: Tools and Techniques

3.1. Concepts for Airspace Modeling

The ultimate framework for airspace modeling that AQM would use to simulate air quality was a governing factor in development of the program. There are many ways to define an air traffic network in a simulation environment and this section will describe how this will be handled by AQM. While these concepts are not currently implemented by the program, they guided the development of processes used to manipulate airspace simulation data and will serve to define the development of airspace and airfield operation and air quality models in the future.

3.1.1. Definition of a study region

AQM deals with the impacts of air traffic, therefore, the study region should include the physical space in which air traffic exists. This region is determined by the scope of the air traffic network as defined in the airspace simulation model used as input to AQM. This region may be limited to one airport, or may include a network linking many airports and the airborne paths of aircraft between airports. It may be stated that the actions of all aircraft occurring during the simulation will take place within the study region.

A study region for an air transportation network is realistically a three-dimensional space enclosing the horizontal limits of the network as well as the variations in altitude which aircraft must traverse. In AQM, this region is mapped in a cartesian coordinate system with x, y, and z axes correlated to eastings, northings, and altitude, respectively. Each aircraft may be located at a point in space at any time for which the flight is active. The contribution of pollutants from flight actions must also be correlated to their location in space.

For purposes of modeling air quality, it may be desired to expand the study region to include surrounding areas which may be affected by aircraft emissions. Likewise, the

study region may be cropped to focus on a particular area. In any case, the study region is ultimately defined by the operator of the simulation to address the specific concerns of individual projects, but should enclose enough space to include all aircraft actions having an appreciable impact. Currently, AQM's default study region includes the physical limits of the air transportation network as defined by the SIMMOD airspace model, described later.

3.1.2. Discrete definition of airspace and time

The computational requirements for an air quality model covering such a large region may be greatly simplified by envisioning the study region as a collection of discrete points in three-dimensional space. Additionally, analyzing the cumulative impacts of aircraft operations on air quality may be simplified by considering the simulation period as a sequence of discrete time intervals.

The environmental conditions at each point may be described for a given time interval during the simulation. Wind direction, temperature, air density, and other environmental parameters should be defined at all points in order to model the dispersion of concentrated aircraft emissions. Once the air quality simulation has been run, the resulting data set will contain the concentration of pollutants at all points within the study region, for each time interval during the simulation. The amount of data that this process will generate is dependent on the separation of discrete points and on the duration of time intervals. A coarse distribution of physical points and longer time intervals will result in a smaller data set, but will likely have decreased accuracy.

3.1.3. Cumulative effects during simulation period

Air quality impacts in an air traffic network are naturally time-dependent; emissions are generated by aircraft only during the period in which they are active. By considering the collective impacts of all active flights for each time interval and applying a dispersion model to the study region, the time-dependent dispersion of

pollutants may be modeled. This process is shown two-dimensionally in Figure 1, beginning with the first time interval for which flights are active in the simulation. Darkened squares represent concentrations of pollutants created by aircraft (a). A dispersion model may then be applied to the entire study region to determine the resulting distribution of pollutants (b). Subsequent aircraft actions are likewise considered, and their respective emissions are accounted for in each time interval (c and d). This process, repeated for all time intervals, will create a data set containing the momentary distribution of airborne pollutants for all time intervals during the study period.

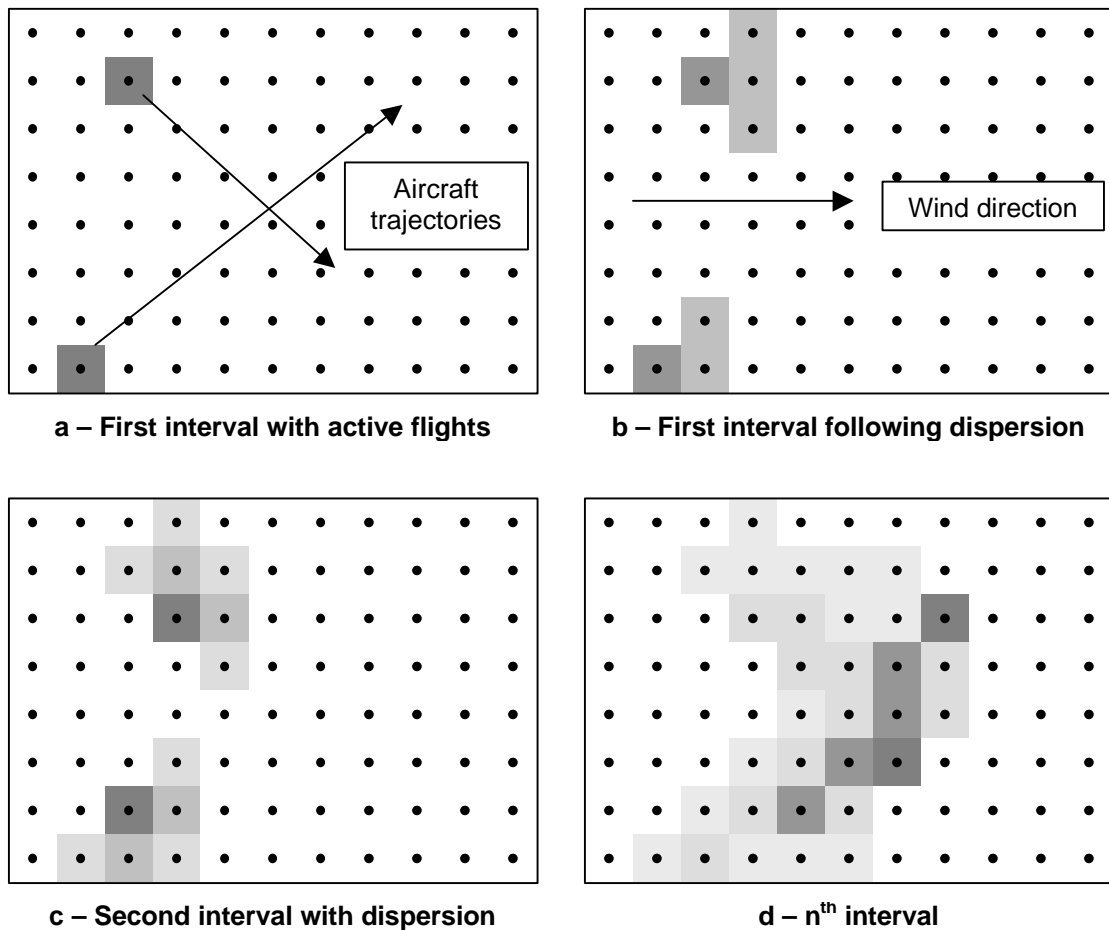


Figure 1 – Incremental modeling of pollutant dispersion

3.2. Programming Tools

In the very early stages of this project, programming tools were selected that would be used in the creation of AQM. Knowledge of the capabilities and limitations of these tools served to guide project development towards creation of a powerful, efficient program. Reasons for selection of specific programming tools are described in this section.

3.2.1. Visual C++: The programming language of choice

In addition to the author's familiarity with C++ programming, Microsoft Visual C++ was used for the following reasons:

- Speed – Given AQM's computational requirements, speed was a primary consideration. Programs written using C++ can be much faster than programs performing the same functions written in Basic or other popular programming languages.
- Organization – Object-oriented programming is a fundamental improvement in C++ over the standard C programming language. Object-oriented programming allows for creation of custom data types called **classes**, which facilitate a modular programming structure that organizes access to the many variables in large programs. In AQM, classes were used to organize the information read from SIMMOD and created by AQM.
- Microsoft Windows compatibility – Using Microsoft Visual C++ allowed this program to be developed for Microsoft Windows; an operating system with a significant installed user base. AQM version 1.0, the finished program for this project, runs under Windows 95 and Windows NT version 4.0. Visual C++ simplified the integration of the familiar Windows interface in AQM through use of the Microsoft Foundation Class (**MFC**) library, included with Visual C++. MFC provides many C++ class definitions which simplify user interface development and extend the functionality of AQM.

- Cross-platform compatibility – C++ is an increasingly popular programming language and lends itself relatively well to cross-platform development should that be an option in the future.

3.2.2. OpenGL: The selected graphics programming standard

OpenGL is a graphics programming standard initially developed by Silicon Graphics for writing applications performing visualization in 3D space, incorporating lighting, perspective, points of view and other parameters.

OpenGL offers some of the same benefits as programming in Visual C++, including compatibility with Microsoft Windows and other platforms. It is a comprehensive tool that also reduces the programming overhead associated with graphics and is used to visualize the air traffic network in AQM.

3.3. SIMMOD as a Simulation Data Source

Much of the conceptual development required an understanding of how SIMMOD data would be interpreted in a finished product. This required an understanding of the file structure of a SIMMOD model. This section describes the information that SIMMOD provides, where it is obtained, and how it is used by AQM.

3.3.1. SIMMOD file formats

Currently, AQM is designed to read data from SIMMOD projects created and run using SIMMOD version 1.0. Upgrading AQM to use models created with later versions of SIMMOD is discussed in Chapter 5, “Next Steps: Future development of AQM”.

SIMMOD 1.0 takes formatted ASCII text files as input. Simulation results are documented in output files, also created as ASCII text files. AQM uses the AIRSPACE and AIRFIELD input files, and the LOG output file, described in the SIMMOD Data Input Manual and the SIMMOD Reference Manual. Figure 2 shows

these files and the information they provide to AQM. The class names given to specific SIMMOD data types are part of the AQM program structure and are explained in more detail later in this report.

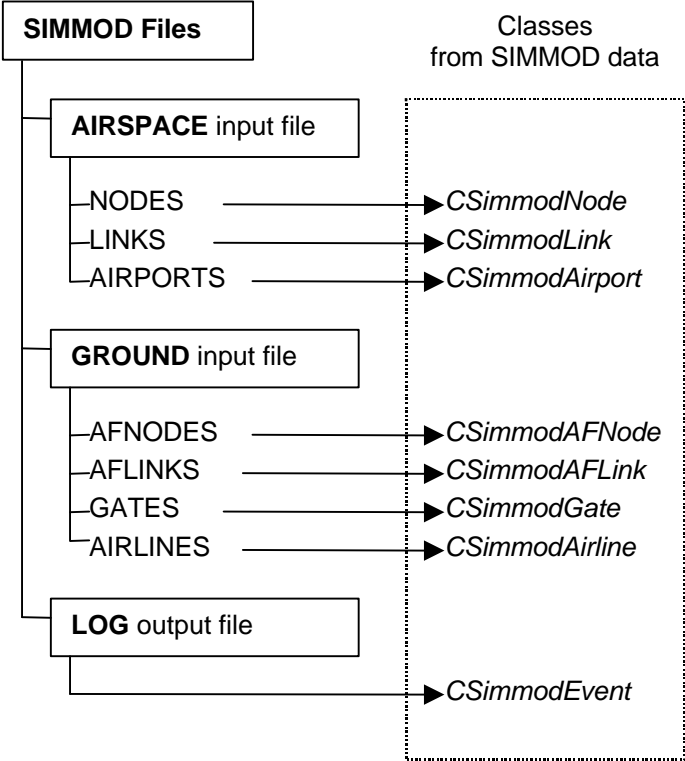


Figure 2 - AQM Classes Defined for SIMMOD Data Types

3.3.2. SIMMOD’s definition of the air transportation network

SIMMOD models aircraft movements in a node / link structure in a fashion common to models of transportation networks. Nodes represent physical points in space that serve as vertices between links along the paths that aircraft can move in the simulation. While the real life movements of aircraft are not so limited, such a data input structure greatly improves the simplicity of the simulation algorithm.

SIMMOD differentiates between airfield and airspace elements. Airfield elements, defining physical elements of the airport ground transportation network, are defined

in the simulation's corresponding **AIRFIELD** file. Airspace elements, which define the airborne travel paths of aircraft around and between airports, are defined in the **AIRSPACE** file. Specific elements contained in each of these files are shown in Figure 2 and are discussed here:

3.3.2.1. Nodes

Nodes are waypoints along the various flight paths of aircraft in the simulation. SIMMOD distinguishes between two types of nodes: airspace nodes and airfield nodes. Airspace nodes form the paths along which aircraft travel, around and in between airports. Airfield nodes are associated with the ground features of an airport; the gates, taxiways, and runways.

3.3.2.2. Links

Links connect the nodes to define travelways in the air transportation network. Links, like nodes, are subdivided into airspace and airfield links. Airspace links connect airspace nodes and define the approach and departure paths, as well as any paths between airports. Airfield links connect airfield nodes and make up the runways and taxiways.

3.3.2.3. Gates

Gates are nodal features that represent the actual loading and unloading points for an aircraft. Each gate is assigned to a particular node that defines the origin or termination point for most arrivals and departures in the simulation.

3.3.2.4. Airports

Support for multiple airports is provided by AQM, as it is in SIMMOD. Gates, runways, and other airfield objects are assigned to a particular airport defined in the SIMMOD input files.

3.3.3. Event-specific information generated by SIMMOD

3.3.3.1. Events

AQM defines “events” as discrete actions by each aircraft in a SIMMOD simulation. These events are derived from the LOG file, which contains chronologically ordered information describing simulated aircraft activity. AQM was developed to work with a complete LOG file, containing all possible information generated by SIMMOD. AQM automatically filters out irrelevant information during the interpretation process.

3.3.3.2. Flights

Event listings in the SIMMOD LOG file contain the corresponding flight name for each event. AQM detects active flights in a simulation by the appearance of flight names in the LOG file.

3.3.3.3. Aircraft

Like flights, the type of aircraft participating in a particular event are listed in the LOG file. AQM extracts the aircraft type from event listings. These aircraft types may be correlated with pollution generation profiles for specific aircraft; this is an idea which is addressed in Chapter 5, “Next Steps: Future development of AQM”.

3.4. Reading SIMMOD Data Programmatically

A significant portion of work for this project involved development of a method for extracting data from SIMMOD model files. These files are ASCII text files, a low-level text format easily readable by most computer systems, which AQM parses to get the information it needs. Samples of the AIRSPACE, AIRFIELD, and LOG files corresponding to a simulation of the air transportation network between Mexico City International and Acapulco airports is included in Appendix B. These files are the starting point for an AQM project that will be referred to as a “case study” during the course of this report.

3.4.1. General considerations

SIMMOD input files are structured to convey information to the SIMMOD simulation engine. That is, data in these files is organized in relatively concise and well-ordered fashion with a minimum of formatting for the sake of readability. This sort of file, since it is created with the purpose of serving as input to a computer program, is fairly easy to interpret programmatically.

SIMMOD output files, on the other hand (LOG.* in particular) are created with the intention of serving as printed output. The event descriptions contained therein are phrased in a “plain-English” fashion which makes them fairly easy for a human to follow, but poses a challenge for AQM. AQM must make sense of the useful information while neglecting comments, page breaks, column headings, and other formatting provided for purposes of easy reading.

The remainder of this section will describe the development and application of interpretive algorithms by AQM to read and “understand” what it is being given from SIMMOD.

3.4.2. AIRSPACE file

The layout of the AIRSPACE input file is provided in Chapter 4 of the SIMMOD Data Input Manual. Data types read by AQM from this file are described in the following paragraphs, with corresponding variable designations as provided in the Data Input Manual.

3.4.2.1. NODES

Num – Node number used by SIMMOD to identify the node in other data records.

Nam – Node name (optional).

Alt – Node altitude (in feet)

Coor – Node coordinates (in nautical miles). This is not required information for SIMMOD, but must be included for use in AQM. These coordinates are arbitrary and need only be accurate relative to each other.

3.4.2.2. LINKS

Num – Link number for identification in other data records.

Nam – Link name (optional).

Nod1, Nod2 – Begin and end node of link. These numbers refer to the “Num” variable assigned to nodes.

3.4.2.3. AIRPORTS

Num – Airport number

Cod – Three-letter airport code name

Elev – Airfield elevation (in feet)

Aptdes – Airport description

3.4.3. AIRFIELD file

The layout of the AIRFIELD file is provided in Chapter 5 of the SIMMOD Data Input Manual. AQM reads data regarding AFNODES, AFLINKS, GATES, and AIRLINES from this file.

3.4.3.1. AFNODES

The location of airfield nodes is actually provided under the COORDINATES heading of the AIRFIELD file. Airfield node data is contained under this heading as follows:

Nodnum – Airfield node number having defined coordinates (for AQM’s purposes, this should be provided for all airfield nodes).

Xcoor, Ycoor – East and north coordinates, respectively, of the airfield node in feet. These coordinates are relative to an arbitrary point that must match the same origin used to define airspace nodes; that is, an airfield node coordinate of (0,0) should be in the same spatial location as an airspace node having coordinates (0,0).

3.4.3.2. AFLINKS

Num – Airfield link number for identification in other data records

Nod1, Nod2 – Begin and end airfield nodes. These numbers refer to the “Num” variable assigned to airfield nodes.

3.4.3.3. GATES

Num – Gate number.

Nod – Airfield node corresponding to gate location.

Nam – Gate name used for identification in event listings.

3.4.3.4. AIRLINES

Num – Airline number.

Nam – Airline name used for identification in other data records.

3.4.4. LOG file

As mentioned previously, the LOG file conveys information about discrete events during the simulation period, with each line containing information for a single event as shown in the following sample line:

```
1 06:04:19; MXN MA102_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 99 (NODE.ARR-071)
```

Events are listed chronologically for the duration of the simulation and include information read by AQM as follows:

Event time – Each event is assigned a particular time accurate to one second which is used by AQM to determine sequence.

Airline & Flight Name – Each event lists a corresponding airline and flight number. The airline name references available airlines defined in the AIRFIELD file. The flight name is read by AQM and is used later to create a flight listing.

Event description – This is a textual “plain English” description of the event. Examination of LOG.MEX in Appendix B shows that there is a large variety of event descriptions corresponding to different actions performed by the simulator.

Event code – This is a descriptive code which defines the event type. A typical event code is a descriptive phrase followed by a three-digit number, such as “NODE.ARR-071”. The syntax for event descriptions is the same for all events with the same event code; this is a fact which is used to further parse the event descriptions for useful information by AQM. These event codes are also used to distinguish between events relevant to AQM and those which are not, such as

simulation trace events and other events which are not descriptive of flight behavior.

3.5. Creating AQM-Specific Data

After reading raw SIMMOD data, AQM needs to interpret its content. Figure 2 showed how SIMMOD information is read directly by AQM. The next step involves further processing within AQM so that this information may serve as useful input for air quality modeling. This step is shown diagrammatically in Figure 3.

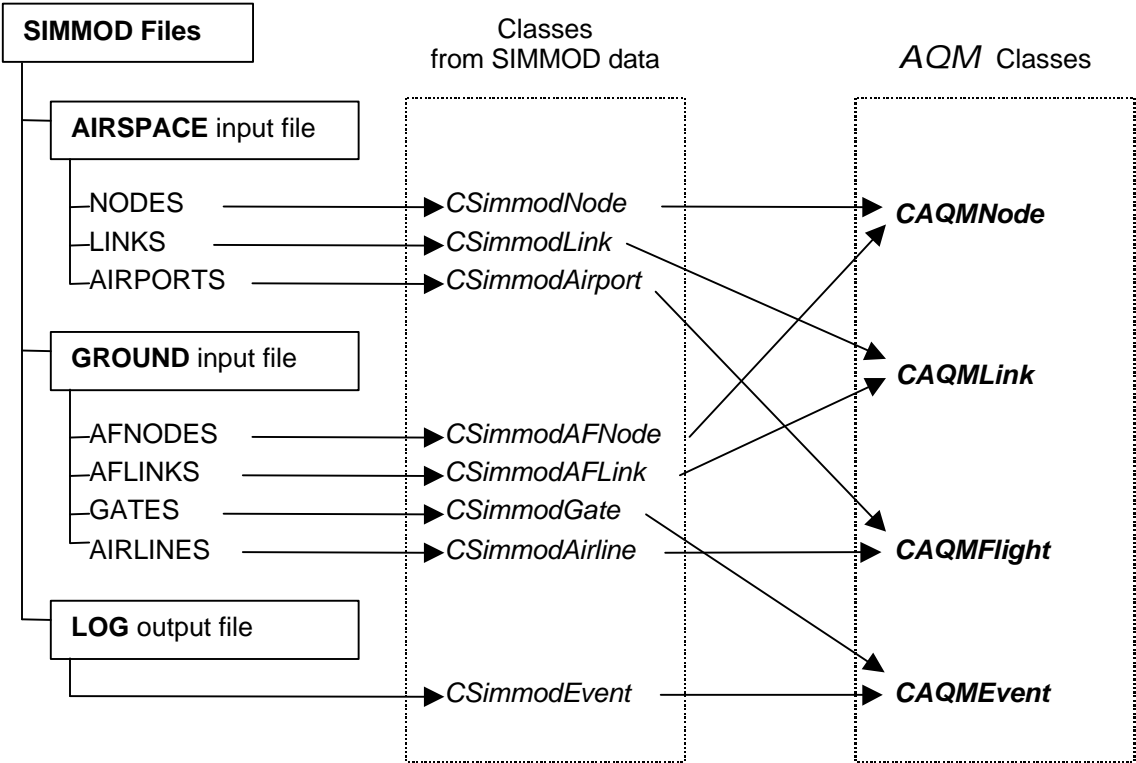


Figure 3 – Conversion of SIMMOD data types

3.5.1. AQMNodes

AQMNodes are given the class designation *CAQMNode* within the AQM program as shown in Figure 3. For the sake of simplicity, it was desired to consolidate

airfield and airspace nodes into a single type, referred to in this report as “AQMNodes”. The SIMMOD designation of “airfield” or “airspace” nodes is maintained within the *CAQMNode* data structure. SIMMOD provides coordinates in different units for airspace and airfield nodes and AQMNodes converts these into a consistent coordinate system with northings and eastings given in nautical miles and altitudes given in feet. AQMNodes created from SIMMOD airfield nodes are assigned the field elevation of the airport they are a part of. The end result is a list of AQMNodes that are related in physical space by a consistent coordinate system in three dimensional space.

3.5.2. AQMLinks

AQMLinks are created to link AQMNodes according to the SIMMOD node-link relationships. AQMLinks reference the AQMNodes rather than the SIMMOD nodes. Just as with AQMNodes, the SIMMOD airspace/airfield designation is maintained in the class *CAQMLink*.

3.5.3. AQMFlights

AQMFlights are derived from the *CSimmodEvent* objects read from the LOG file. The algorithm for creating AQMFlights recognizes all flights referenced by the *CSimmodEvent* objects, ensuring that each AQMFlight is relevant to the simulation.

3.5.4. AQMEvents

When AQM extracts *CSimmodEvents* from the LOG file, it does not perform the interpretation necessary to make sense of the textual descriptions provided for each event. Parsing these descriptions into AQMEvents serves this function, and is arguably the most important step of the AQM data interpretation process.

3.5.4.1. Reading individual events

As mentioned previously, the syntax of event data depends on the event type. AQM uses this fact to assist interpretation of event descriptions. Each event description contains specific information that is necessary for flight actions to be completely described.

3.5.4.2. Importance of event sequence

When event data is interpreted for AQMEvents, there needs to be consideration for the sequence of events for each flight. AQMEvents keep track of what “mode” the flight is in at the time of that event; the mode will be used in future correlation with pollution generation profiles to determine how much pollutants are being generated by each aircraft for the duration of each event.

3.6. Developing the AQM User Interface

While the data input functions do not require an extensive user interface, support for viewing this information is provided in AQM. This facilitates verification of data read into AQM and serves as a starting point for an enhanced user interface.

3.6.1. Support for different data types

All the raw data that AQM currently deals with may be organized in lists. There is a view type directly supported by MFC, called *CListView*. A new list view was derived from *CListView* for AQM called *CAQMListView*. This class extends the functionality of the MFC list view class to meet the needs of AQM and is used for displaying all textual information.

While list views are sufficient for displaying the raw data in AQM, a different approach was required to visualize the air transportation network. For this, a new view class was created in AQM that takes advantage of the OpenGL graphics programming library. *CAQMNetworkView* is the class created for this purpose,

which allows the user to interactively view the air transportation network defined by AQMNodes and AQMLinks.

3.6.2. User interface features

AQM is unlike many other applications in that it is designed to perform a relatively fixed number of functions. While a typical word processing or spreadsheet application may have several hundred commands for modifying the documents they work with, AQM's command structure is relatively fixed: "Read a SIMMOD project" and "Generate AQM Events" are really the only two functions that AQM supports at this time. Looking forward to future development, AQM will still require a minimal selection of document-level commands: "Perform Simulation" might be used to actually perform an air quality analysis with information loaded from SIMMOD. "Create Visualization" could add a rendered time-dependent movie of air quality impacts computed during an AQM simulation. In any case, the addition of commands will be limited to a very few, high level commands.

Modularity is a recurring theme in AQM development. This concept is extended to the view architecture of AQM in the form of a user interface that is designed to grow to accommodate the needs of the program. Again, the functionality provided through MFC allowed for some creativity in how this was accomplished. This section gives a brief overview of the development of AQM's view architecture. A detailed description of the user interface is provided in Chapter 4.

3.6.2.1. Tree view control

In accordance with the limited command structure of AQM, the organization of information in any AQM document can be described in terms of its limited components. A new AQM document is empty when it is created. In its current form, AQM creates "finished" documents that contain SIMMOD data and generated AQM data. In future versions, AQM may add a number of Simulation objects to a document for air quality simulations, storing the

conditions under which each simulation was run and the results obtained. Multiple Visualizations may be created as well which would apply to specific Simulations within a document. All in all, the AQM document contents are and will always be a well organized structure of objects. What all this means, in terms of establishing a user interface, is that the program needs to work with documents in a manner consistent with their form - as a hierarchical collection of objects.

The MFC Tree Control View (class *CTreeView*) is ideally suited to displaying this type of structure; it is the same technique used in Microsoft Windows Explorer to navigate through the directory structure of a computer's file system.

Sidenote:

Other operating systems makes use of a similar type of control for browsing hierarchical collections of objects. These controls would be implemented in future versions of AQM ported to operating systems other than Microsoft Windows.

In AQM, the tree control (class name *CAQMTree*) is used to select among information contained in an AQM document. The tree control differentiates between SIMMOD project information, AQM-generated information, and auxiliary views based on the information contained therein. It branches out into subsets of those items, such as Simmod Nodes and AQM Events. By picking an item from this tree, the corresponding information may be shown in the adjacent view frame provided in the AQM application. This layout provides a persistent view into the document contents which, in turn, serves to remind the user of what data exists. Further examples of how the tree control enhances the user interface are provided in Chapter 4.

3.6.2.2. Collection header views

Each main object collection (SIMMOD project information and AQM objects, specifically) is described by its constituent parts. By selecting one of these folders from the tree control, the user is given a view containing information about what the collection contains.

3.6.2.3. Object lists

By selecting a particular object type from the tree control, a list view containing the characteristics for every such object is shown. This feature lets the user review data that has been read into the document or created by AQM. The class *CAQMListView* is used for all objects and is designed to accommodate the different display needs of each data type.

3.6.2.4. Network view

Once AQM project information is generated, the user may view the air transportation network created by the class *CAQMNetworkView*. This view incorporates much of the functionality that will be useful in creating visualizations in future versions of AQM.

The default network view shows the entire air transportation network, defined by AQMNodes and AQMLinks, from a birds-eye perspective. Control of the view's perspective is provided by monitoring mouse actions within the window. View altitude, heading, location and "angle of attack" (view angle relative to the ground) are controlled by the user. The network may be viewed from a point between airports, from a point near the ground, or the user may zoom in on a particular airport. These capabilities are useful now as assurance that the network is created properly in the SIMMOD input files, and will be more useful as visualization features are added.

There are a very few rough edges to the view control provided in the network view. Support for user-selectable color is a feature that would be effective but is not currently implemented. Also, the user may move their point of view below airports with a field elevation greater than 0, providing an “underground” perspective of airport operations. These are problems that are relatively easy to fix and there is, of course, potential for fixing these problems and providing further enhancements in future versions of AQM.

4. Result: The AQM Shell

4.1. Overview

This chapter describes, in detail, the program structure and operating characteristics of AQM. This information is included for those who would have an interest in the inner workings of AQM.

4.2. AQM Class Structure

Chapter 3, “Methodology: Tools and Techniques”, introduces some of the classes used in the AQM program without much explanation of their purpose within the programmatic framework. The term “class” is a familiar term to anyone involved in object oriented programming. Classes allow the programmer to create “objects” in the programming environment that possess characteristics and behaviors of the real-world objects they represent. In the case of AQM, classes were defined within the Visual C++ workspace used to create the AQM program.

Simply put, AQM classes can be broken down into three categories: SIMMOD element classes, AQM element classes, and functional classes. The C++ header files for these classes are included in Appendix C along with a brief description of their function in AQM. Without delving into a textbook style review of object-oriented programming, the remainder of this section will describe classes created for AQM and how they relate to the finished program.

4.2.1. SIMMOD element classes

SIMMOD element classes, distinguished by the “*CSimmod*” prefix in their name, serve to describe the individual data elements read directly from SIMMOD project files. The member variables of these classes mimic the structure of these elements as they are defined in the SIMMOD input and output files.

4.2.2. AQM element classes

AQM element classes, prefixed by “*CAQM*”, define the elements derived from SIMMOD elements that describe the air transportation network, flights, and events in a format that AQM can understand. Saying that AQM “understands” these classes means that AQM could use the information contained therein in an air quality analysis without further conversion.

4.2.3. Functional classes

Functional classes serve as the foundation for the AQM program – while they are not necessarily related to either SIMMOD or AQM derived data, functional classes are necessary for the program to operate as a stand-alone program under Microsoft Windows. Some of these classes are AQM-specific versions of classes that are present in all Visual C++ applications made to run under Windows. Functional classes define the AQM user interface, the structure of AQM documents, and allow for manipulation of an AQM document through the user interface. The “document-view architecture”, described in the following section, describes the relationship between the user interface and the document and how this is utilized in AQM.

4.3. AQM Document / View Architecture

To facilitate the creation of multiple projects that the user may store, retrieve, and inspect on demand, AQM takes advantage of a program structure known as a document/view architecture. Simply put, a document/view architecture stores data for individual projects in documents which can be saved as files and provides the user access to information contained in these documents through on-screen views. Examples of document/view architecture are prevalent in many Windows-based applications; a word processor stores the text, formatting, and other objects in document files and allows the user to view and manipulate these contents within views. Without this document/view architecture, the usefulness of AQM would be limited.

The documents that AQM creates contain all the information that is read from SIMMOD and that which is subsequently generated by AQM. Referring to the class structure described previously, AQM documents contain all the SIMMOD and AQM element classes that describe the air transportation network and the flights and events that take place within the scope of the original SIMMOD project. AQM documents are given the default extension “AQM” to distinguish these from other files. The final size of these documents is related to the size of the original SIMMOD project which is read into AQM.

AQM also features a “multiple document interface”, or MDI. An MDI allows more than one document to be open at one time, allowing a user to switch between projects without having to exit and restart AQM for each project. This feature will be useful for comparing results between projects once the air quality modeling processes are provided.

4.4. Using AQM to Interpret SIMMOD Projects

The primary function of AQM is, currently, to read information from SIMMOD projects and assemble it into a form that is suitable for use in a air quality model. This section describes the sequence of events that an AQM user must go through to make this happen.

4.4.1. Starting from scratch

When AQM is started, the user is provided with a new, empty document as shown in Figure 4. This emptiness is reflected in the tree view on the left which shows no data and has no branches. From the “Project” menu, the user is provided with a single option - “Open SIMMOD Project” (other menu options are visible, but are “grayed out” – a technique in Windows programming used to distinguish inactive menu options from those that are currently active). Until this operation is performed, there is nothing else for AQM to do with the document. When this option is selected, the user is provided with a dialog box that allows them to select the folder containing the SIMMOD project to be read into AQM. This dialog box provides a “Browse” button that lets the user select this folder from a directory tree similar to Windows Explorer. Once the desired project is selected, clicking on the

“OK” button starts the process through which AQM reads SIMMOD elements into the document. The user can monitor the progress of these operations through the status bar in the lower left portion of the AQM window. A SIMMOD project with 200 nodes, 100 links, and 2000 events should take no longer than about 10 seconds on a Pentium-class machine, so the status bar is mostly useful for large projects or slow machines. When this process is completed, the tree view is updated to reflect the fact that the document now contains SIMMOD project information. The number of SIMMOD elements read into the document is shown in the SIMMOD project view, and the user may view the contents of individual element lists by selecting the corresponding entry in the tree. The final appearance of an AQM document after this step has been completed is shown in Figure 5.

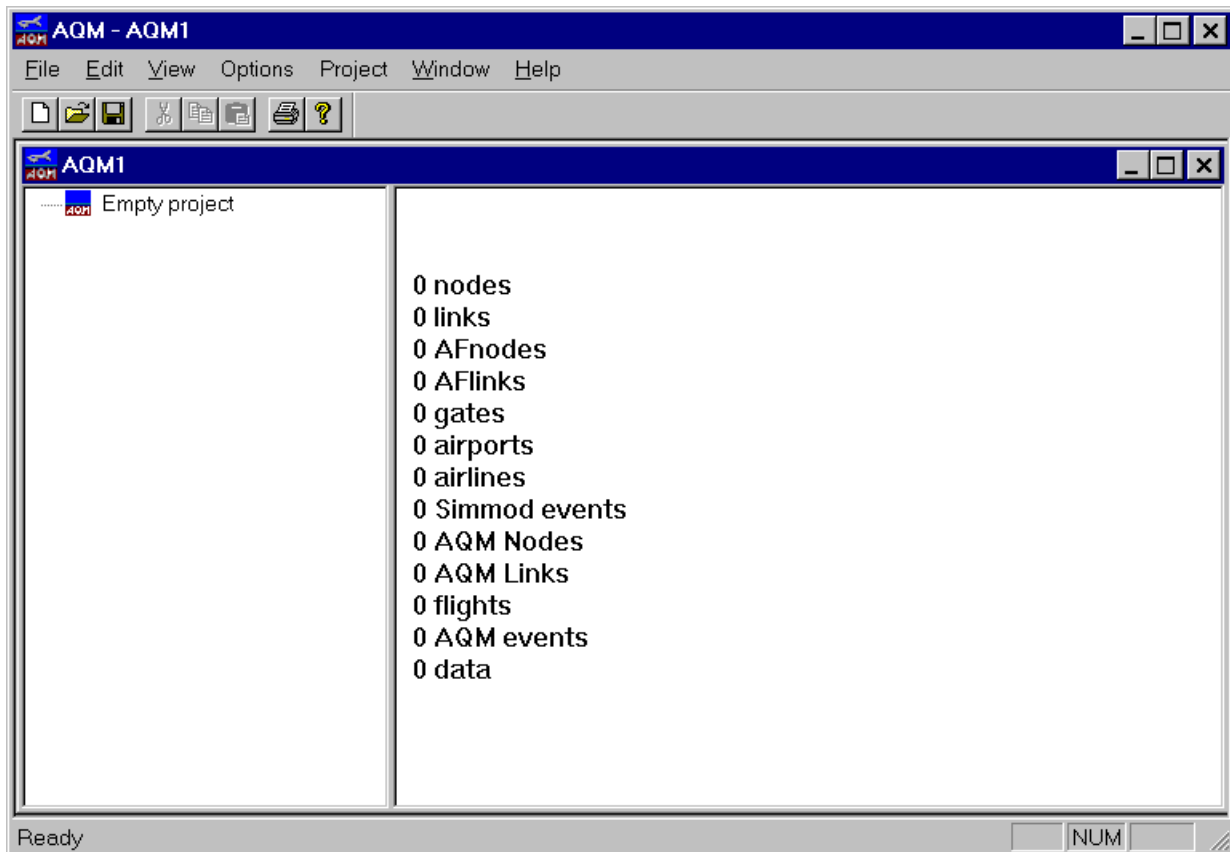


Figure 4 – Blank AQM Document

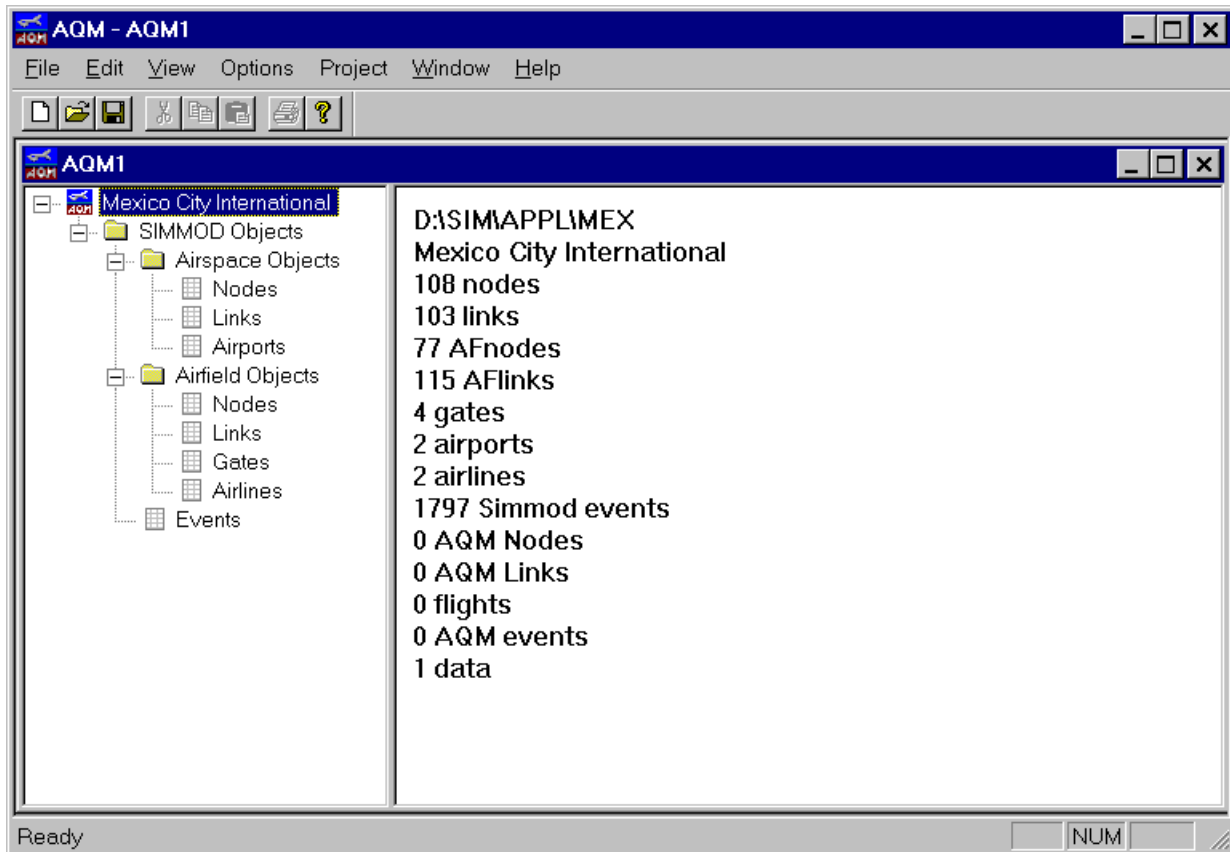


Figure 5 – AQM Document following SIMMOD model input

Sidenote: *If the “Open SIMMOD Project” option is selected in a document that contains data, the user is given the opportunity to decide whether to continue, erasing the existing data, or to abort the process altogether.*

4.4.2. Generating AQM elements from SIMMOD data

Once a SIMMOD project has been read, AQM is ready to convert this information into a format that will be useful in modeling the air transportation network and performing air quality analysis. After the “Open SIMMOD Project” option has completed successfully, a new menu option is enabled – “Build Events”. This is the next (and final) step towards this goal.

Once this menu option is enabled, selecting “Build Events” creates AQM elements from the SIMMOD elements. This process involves conversion of SIMMOD airfield and airspace elements into AQM elements related by a consistent coordinate system, and interpreting the raw text contained in SIMMOD events to create flight and event records that are relevant at a programmatic level.

Following selection of the “Build Events” option, the tree view expands to reflect the fact that the document contains this new information similar what is done with SIMMOD elements, as shown in Figure 6. A new item is also created within the tree, named “Network View”; this tree item contains no data, instead, it allows the user to view the physical layout of the air transportation network in the airspace coordinate system.

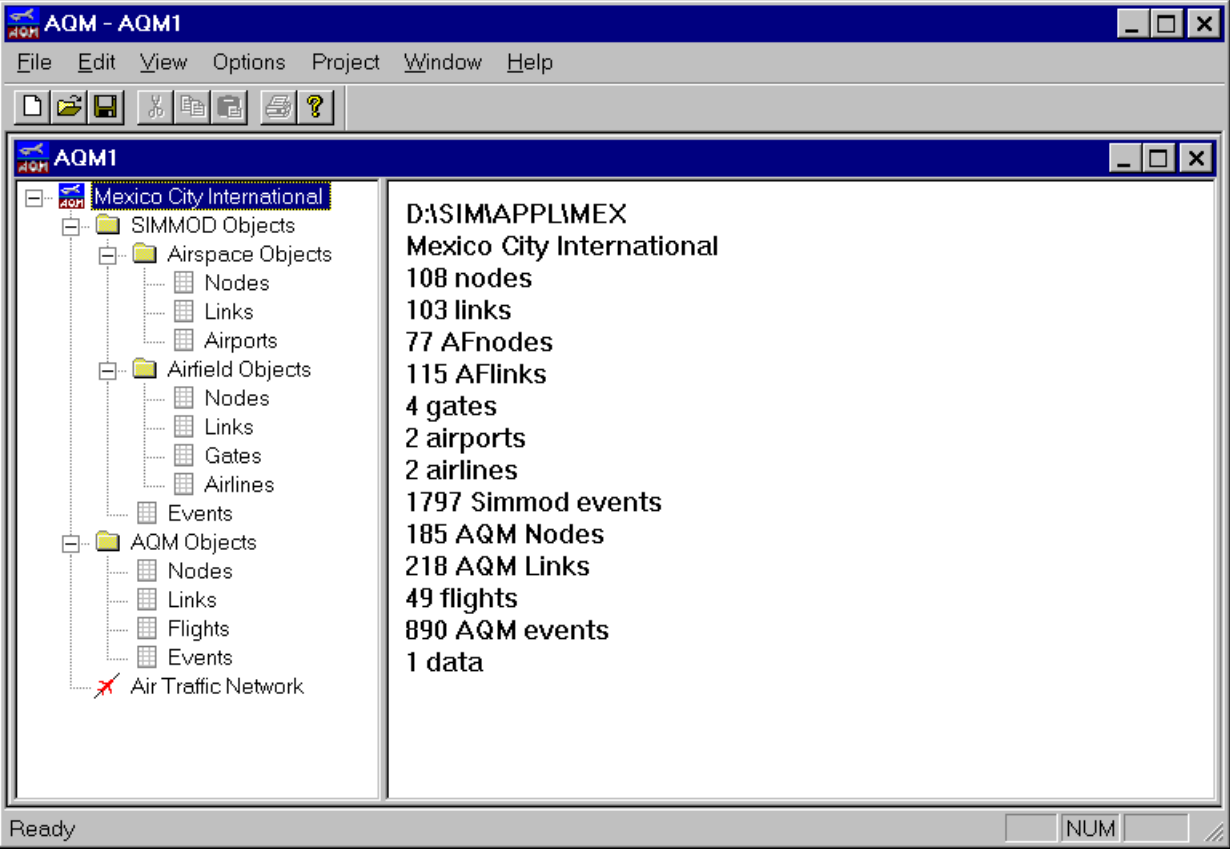


Figure 6 – AQM Document with AQM Objects

The two steps described in this section demonstrate the current functionality of AQM. While it is a simple process for the end user, the program goes through many complex procedures to ensure the end product (AQM elements, flights, and events) retain the essential attributes that are described in the original SIMMOD project.

4.5. Viewing SIMMOD and AQM Data

The tree control to the left of an AQM document view serves two purposes. First, it provides information to the user at a glance, describing what information is currently included in the document and, therefore, what information needs to be created. Second, it provides a mechanism through which the user may view information contained within the document.

ID	Num	Name	Northing (nm)	Easting (nm)	Altitude (ft)
1	1	I-1	-0.44897	-0.43860	7361
2	2	I-2	0.71378	0.97661	7361
3	3	L_3	-0.34430	-0.52073	7361
4	4	L_4	0.69419	0.76578	7361
5	5	N_5	-19.08800	16.32024	18000
6	6	N_6	-27.48672	13.64792	20000
7	7	N_7	-46.00208	-10.21208	21000
8	8	N_8	-106.41559	10.02120	21000
9	9	N_9	-135.42935	-25.67336	12500
10	10	N_10	-155.67123	-27.68717	5500
11	11	N_11	-159.89215	-28.79265	4200
12	12	N_12	-162.17651	-33.33250	3500
13	15	N_15	-134.70673	-28.35931	11500
14	16	N_16	-155.65529	-33.33382	3000
15	17	N_17	-158.21776	-34.99884	2000
16	20	N_20	-160.42697	-46.40450	800
17	22	N_22	-155.81075	-59.78350	8000
18	23	N_23	-141.61217	-63.51997	10700
19	24	N_24	-125.91900	-48.57409	14500
20	26	N_26	-45.90955	-12.66470	20000
21	27	N_27	-27.03419	-7.15389	20000
22	28	N_28	10.95540	1.39502	12500

Figure 7 – SIMMOD Airspace Node List View

There are basically two types of information contained within an AQM document. SIMMOD and AQM objects may be viewed in list form, showing attributes of every element of a particular type. Since some of these elements describe the physical layout of the air traffic network described in the SIMMOD project, it makes sense to provide some sort of view in which to visualize this network.

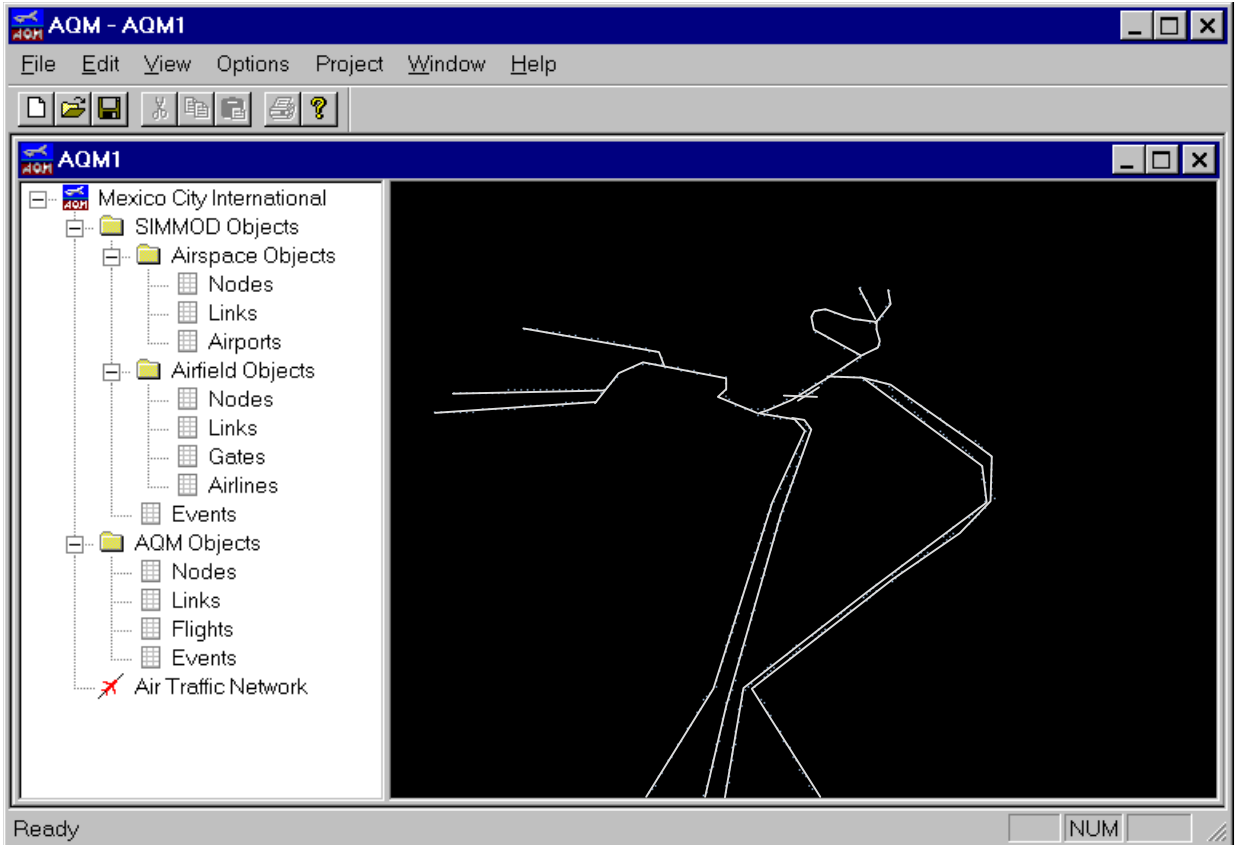
4.5.1. Element attribute listings

The function of the element list view is best described by example. Figure 7 shows the information that is displayed by opening the SIMMOD Objects tree and selecting “Nodes” from the Airspace branch for the Mexico City International airport example. Depending on the item selected, AQM will show a list of all such elements and the members of those elements.

4.5.2. Network view

AQM provides the ability to view the air traffic network defined by the AQM elements. While this is not absolutely necessary, such a view will be a desirable feature once AQM is expanded to support actual air quality simulations. This view, shown in Figure 8, will be a basis for the creation of time-dependent, spatial visualizations of air quality. This feature was included at this early stage of development to demonstrate its usefulness in future enhancements.

AQM currently provides the opportunity to manipulate the air traffic network view to accommodate different points of view; in this sense, the view acts almost like a CAD interface that allows the designer to view the project from a desired point of view to get the best perspective on the information contained therein. The view control is provided for through mouse movements within the Air Traffic Network view window as shown in the table below the screen shot in Figure 8.



Mouse Action	View Response
Left button + move forward & backward	Rotate view within vertical plane to show network from different angles from the horizontal (azimuth angle)
Left button + move sideways	Rotate view within horizontal plane to show network from different heading
Right button + move forward & backward	Zoom in and out from the center of the view (use Pan feature to change location of view center)
Left button + SHIFT key + move	Pan view within horizontal plane in direction of mouse movement

Figure 8 – AQM Network View

4.6. Program output

Besides creating documents, AQM provides the ability to export out the converted event information into an ASCII-format text file. This event listing contains all events, ordered chronologically, with the coordinates of their origin and termination points and the type of

aircraft that are tracked by each event. This feature is made available through the “File” menu, under “Write ASCII Event file”. The format of this file is a tab-delimited ASCII text format as shown in Figure 9.

ID	AQC	Time1	North1	East1	Elev1	Time2	North2	East2	Elev2
1	10106	06:00:22	-2.38689	-41.87915	16000	06:01:58	-1.82371	-31.96856	15000
2	10106	06:01:58	-1.82371	-31.96856	15000	06:03:02	-1.39460	-25.42466	14000
3	10106	06:03:02	-1.39460	-25.42466	14000	06:03:43	2.38689	-23.79662	13000

Figure 9 - AQM "Write ASCII Event file" output

The headings "ID" and "AQC" represent the ID of the AQM Event and the corresponding Air Quality Code, respectively. Time1 and Time2 represent the beginning and end simulation time of the event, followed by the North, East and Elevation coordinates of the location of the aircraft at Time1 and Time2. Loyal to the AQMNode coordinate system, North and East coordinates are given in nautical miles and elevations are given in feet.

The option for extracting textual data from AQM demonstrates the potential for generating reports or other summary information from the long lists of information AQM deals with internally.

5. Next Steps: Future development of AQM

In keeping with the objective set forth earlier in this report, this project should serve as a foundation for further development of AQM, an application for modeling and visualizing the air quality impacts of air traffic. The following items describe features which may be supported in future enhancements to AQM.

5.1. Pollution Generation Parameters

To simulate air traffic impacts on air quality, enhancements will need to be made to correlate flight actions with the pollution they generate. Such enhancements will include the addition of a database containing aircraft types and the corresponding emission rates for different flight regimes. This database should be user-editable so that changes or updates to the aircraft list may be made on demand. The types of pollutants to be tracked would include all compounds or substances typically considered in air quality studies involving aircraft emissions, and could be expanded to include new categories.

5.2. Dispersion Models

Dispersion models are an integral part of many air quality modeling processes. AQM, in particular, must incorporate dispersion processes to properly simulate the impacts on air quality over a period of time. Research into dispersion models applied to emissions from aircraft has yielded results which will be used as a foundation for applying a dispersion model in AQM.

It is recognized that the effective application of dispersion models will require some support for user input of environmental parameters. Wind direction, altitude, temperature, ambient pressure and air turbulence are parameters to be considered in future versions of AQM.

5.3. Visualization Capabilities

The primary goal of AQM as set forth in this report is to create a framework for air quality visualization software. Visualization is a feature that is not currently provided in existing air

quality evaluation models. It is believed that including this feature in AQM will provide much more valuable information than can be derived from numeric results alone. On-screen visualizations, as well as representative visualizations formatted for hardcopy output will serve the analyst in determining a strategy for mitigating the impacts of air quality and will enhance the presentation value of this information.

5.3.1. On-screen visualizations

The Network View provided in the current version of AQM serves as a demonstration of the basic capabilities that could be extended to visualize air quality. The Network View shows the air traffic network from any point of view the user selects. Once an air quality simulation has been run in AQM, the Network View can be expanded to show the time-dependent aircraft movements and the corresponding emissions. This could be accomplished with media controls attached to the Network View to navigate the simulation period much like an online movie; the user would have complete control over where and when the visualization is focused. This capability will allow for detection and mitigation of isolated cases of excessive concentration of pollutants once AQM is used for full-scale air quality analysis.

5.3.2. Report generation processes

In addition to on-screen visualizations, AQM should provide the option to create hardcopy output of the simulation results and the corresponding visualizations. This will enhance the usefulness of AQM as a supplementary tool for reports and future research. In addition to hardcopy output, in this digital age AQM should be able to create digital copies of the results and images from air quality simulations for use in spreadsheets, online presentations, and distribution.

5.4. General Enhancements

Due to time constraints, AQM provides only the basic features necessary to demonstrate a working product. There are many possibilities for enhancements that can make AQM more robust and user-friendly.

5.4.1. Compatibility with later versions of SIMMOD and third-party products

Currently, AQM is designed to read data created from SIMMOD version 1.0. The version-specific nature of the AQM routines that read from SIMMOD files leads to compatibility problems with models created with later versions of SIMMOD. This is a limitation that will certainly be corrected in later releases of AQM.

5.4.2. Airport, airspace, and event editing

Software packages, such as **SimmodPLUS!** exist that allow editing of the network and refinement of the model structure within SIMMOD. Never the less, it would be desirable to have some degree of editing capabilities inside AQM for fine-tuning the network, renaming elements, or adding descriptive monikers to elements and events.

5.4.3. Support for metric (SI) units

AQM is currently limited to imperial units. Given the worldwide proliferation of SI (metric) units, AQM should provide the user with the option to convert data between unit systems. This will make AQM more appropriate for current and future air traffic planning in an international market.

6. Conclusions

AQM demonstrates the basic functions that are required to derive pollution-generating events from SIMMOD project data. AQM has also been developed to facilitate future expansion to include full-scale air quality simulations and detailed visualization of simulation results. In this sense, this project has demonstrated the successful creation of a framework for air quality visualization software, satisfying the objective of this report.

Future inquiries into this report and the AQM program may be directed to the author.

References

1. SIMMOD Data Input Manual, Federal Aviation Administration, September, 1989.
2. SIMMOD Reference Manual, Federal Aviation Administration, September, 1989.
3. Shaojie Xu, "An Airport Air Quality Evaluation Model", Research report, Department of Civil Engineering, Virginia Tech, March 1995.
4. Michael T. Moss and Howard M. Segal, "The Emissions and Dispersion Modeling System (EDMS): Its Development and Application at Airports and Airbases", *Journal of the Air & Waste Management Association*, June 1994, p. 787.
5. B. Gordon Woodmansey and Judith G. Patterson, "New Methodology for Modeling Annual-Aircraft Emissions at Airports", *Journal of Transportation Engineering*, Vol. 120, No. 3, May/June 1994, pp. 339-357.
6. William Bowlby and Roger L Wayson, "Use of Microcomputers to Perform Airport Emission Inventories", *Journal of Transportation Engineering*, Vol. 116, No. 3, May/June 1990,. pp. 328-337.
7. William Bowlby and Roger L Wayson, "Inventorying Airport Air Pollutant Emissions", *Journal of Transportation Engineering*, Vol. 114, No. 1, January/February 1988,. pp. 1-20.

Appendices

A. Glossary

B. Sample SIMMOD files – "Mexico City International" Example

C. AQM Class Header Files

Appendix A -Glossary

airfield – Pertaining to the physical layout of the ground-based operations at an airport (essentially applies to all aircraft movements at the airfield elevation).

airspace – Pertaining to the physical layout of the air-based operations at and around an airport (includes all approach, departure, and flyover operations generated by the model).

class – A data structure provided for by the C++ programming language, which allows a user to include any number of variable types and/or functions, called "members". Objects may be created from these class definitions, which allows the member variables and/or functions defined for the class to be used for these objects. Traditional naming convention is to begin class names with a capital "C" and this has been followed in the development of AQM.

member – Refers to a specific variable or function defined as part of a C++ class in the AQM program.

MFC – Acronym for the "Microsoft Foundation Class" library included as part of the development platform for programs written in Visual C++. MFC provides numerous class definitions to simplify the process of creating programs that run under Microsoft Windows. The MFC provides higher-level access to many of the low-level functions that all Windows-compatible programs (including Windows itself) use to provide a consistent user interface. The MFC also provides some class definitions that simplify traditional C programming tasks.

SIMMOD - the Federal Aviation Administration's (FAA's) airport and airspace simulation model; a comprehensive planning tool for the study and evaluation of en route air traffic, terminal area air traffic, and airport and airline ground operations at one or more airports. SIMMOD is a registered trademark of the Department of Transportation's Federal Aviation Administration (FAA)

this project – Refers to the portion of AQM development that is considered part of the work related to fulfillment of the Master's degree.

Appendix B –Sample SIMMOD Files –Mexico City International’Example

created for use with SIMMOD version 1.0

Created by Dr. Antonio Trani for use with Virginia Tech Transportation Systems Lab

File Extension = "MEX"

Contents

AIRSPACE.MEX – "Airspace" input file (6 pages)

GROUND.MEX – "Airfield" input file (4 pages)

LOG.MEX – Event log file generated by SIMMOD post-processor (5 pages)

AIRSPACE.MEX

PRINT 1

NODES Default separation is 3 nmiles, Max nodes = 108
 1 I-1 7361 1 0 3.000 0 1 0 ; -.43860 -.44897
 2 I-2 7361 1 0 3.000 0 1 0 ; .97661 .71378
 3 I_3 7361 1 0 3.000 0 1 0 ; -.52073 -.34430
 4 I_4 7361 1 0 3.000 0 1 0 ; .76578 .69419
 5 N_5 18000 1 0 3.000 0 1 0 ; 16.32024 -19.08800
 6 N_6 20000 1 0 3.000 0 1 0 ; 13.64792 -27.48672
 7 N_7 21000 1 0 3.000 0 1 0 ; -10.21208 -46.00208
 8 N_8 21000 1 0 3.000 0 1 0 ; 10.02120 -106.41559
 9 N_9 12500 1 0 3.000 0 1 0 ; -25.67336 -135.42935
 10 N_10 5500 1 0 3.000 0 1 0 ; -27.68717 -155.67123
 11 N_11 4200 1 0 3.000 0 1 0 ; -28.79265 -159.89215
 12 N_12 3500 1 0 3.000 0 1 0 ; -33.33250 -162.17651
 15 N_15 11500 1 0 3.000 0 1 0 ; -28.35931 -134.70673
 16 N_16 3000 1 0 3.000 0 1 0 ; -33.33382 -155.65529
 17 N_17 2000 1 0 3.000 0 1 0 ; -34.99884 -158.21776
 20 N_20 800 1 0 3.000 0 1 0 ; -46.40450 -160.42697
 22 N_22 8000 1 0 3.000 0 1 0 ; -59.78350 -155.81075
 23 N_23 10700 1 0 3.000 0 1 0 ; -63.51997 -141.61217
 24 N_24 14500 1 0 3.000 0 1 0 ; -48.57409 -125.91900
 26 N_26 20000 1 0 3.000 0 1 0 ; -12.66470 -45.90955
 27 N_27 20000 1 0 3.000 0 1 0 ; -7.15389 -27.03419
 28 N_28 12500 1 0 3.000 0 1 0 ; -1.38593 -10.85649
 29 N_29 11900 1 0 3.000 0 1 0 ; -3.09079 -7.40724
 30 N_30 10000 1 0 3.000 0 1 0 ; -6.49628 -6.06319
 31 N_31 8300 1 1 3.000 0 1 0 ; -5.10499 -4.48243
 34 N_34 9000 1 0 3.000 0 1 0 ; -56.79432 -151.32698
 35 N_35 10300 1 0 3.000 0 1 0 ; -58.40660 -144.30824
 38 N_38 6500 1 0 3.000 0 1 0 ; -50.81598 -153.56886
 39 N_39 10000 1 0 3.000 0 1 0 ; -50.81598 -146.09593
 40 N_40 10500 1 0 3.000 0 1 0 ; -44.09033 -140.86487
 41 N_41 16500 1 0 3.000 0 1 0 ; -40.35322 -99.10751
 55 I_55 0 1 0 3.000 0 1 0 ; -39.54291 -161.46815
 56 I_56 0 1 0 3.000 0 1 0 ; -41.16549 -161.19775
 57 I_57 0 1 0 3.000 0 1 0 ; -40.36350 -161.12303
 58 I_58 0 1 0 3.000 0 1 0 ; -41.09999 -161.56789
 59 N_59 11500 1 0 3.000 0 1 0 ; -28.36931 -134.70673
 60 N_60 20000 1 0 3.000 0 1 0 ; -12.76470 -45.90955
 61 N_61 8000 1 0 3.000 0 1 0 ; 3.89753 3.02819
 62 N_62 8500 1 0 3.000 0 1 0 ; 6.70838 2.65148
 65 N_65 8600 1 0 3.000 0 1 0 ; 5.54383 5.11503
 66 N_66 8816 1 0 3.000 0 1 0 ; 8.05102 7.46762
 67 N_67 21000 1 0 3.000 0 1 0 ; 10.03120 -106.41559
 69 N_69 400 1 0 3.000 0 1 0 ; -46.07862 -163.99159
 70 N_70 950 1 0 3.000 0 1 0 ; -48.79505 -168.01593
 71 N_71 14000 1 0 3.000 0 1 0 ; -11.83922 -46.39323
 72 N_72 21000 1 0 3.000 0 1 0 ; 10.51157 -30.00576
 73 N_73 12500 1 0 3.000 0 1 0 ; -42.61434 -124.37260
 74 N_74 20000 1 0 3.000 0 1 0 ; -32.08077 -82.04727
 75 N_75 21000 1 0 3.000 0 1 0 ; -7.86893 -120.80260
 76 N_76 12500 1 0 3.000 0 1 0 ; 9.27941 -.31104
 77 N_77 14500 1 0 3.000 0 1 0 ; -10.81208 -46.00208
 79 N_79 9300 1 0 3.000 0 1 0 ; 8.90078 -.71206
 80 N_80 10500 1 0 3.000 0 1 0 ; 15.66537 -19.22568
 81 N_81 11500 1 0 3.000 0 1 0 ; 13.17315 -27.41439
 82 N_82 1450 1 0 3.000 0 1 0 ; -45.37436 -170.73236
 83 N_83 2000 1 0 3.000 0 1 0 ; -41.40003 -167.20566
 84 N_84 13000 1 0 3.000 0 1 0 ; -29.51943 -114.21207
 85 N_85 10000 1 0 3.000 0 1 0 ; 13.83675 -14.19460
 86 N_86 15000 1 0 3.000 0 1 0 ; -12.91058 -56.13297
 87 N_87 11000 1 0 3.000 0 1 0 ; -33.55076 -130.24185
 88 N_88 3000 1 0 3.000 0 1 0 ; -40.26167 -161.75652
 89 N_89 15000 1 0 3.000 0 1 0 ; -21.56796 -103.58295
 90 N_90 11000 1 0 3.000 0 1 0 ; -50.59737 -137.47516
 91 N_91 14000 1 0 3.000 0 1 0 ; -11.80919 -46.42792
 92 N_92 13500 1 0 3.000 0 1 0 ; -6.96829 -29.34721
 93 N_93 10200 1 0 3.000 0 1 0 ; -.64708 -9.62530
 94 N_94 9400 1 0 3.000 0 1 0 ; -2.58832 -7.19875
 95 N_95 8400 1 0 3.000 0 1 0 ; -4.72219 -6.48143
 96 N_96 14000 1 0 3.000 0 1 0 ; -15.39753 -60.16607
 97 N_97 16000 1 0 3.000 0 1 0 ; -41.87915 -2.38689
 98 N_98 15000 1 0 3.000 0 1 0 ; -31.96856 -1.82371
 99 N_99 14000 1 0 3.000 0 1 0 ; -25.42466 -1.39460
 101 N_101 12500 1 0 3.000 0 1 0 ; -21.40972 7.30534

AIRSPACE.MEX

102	N_102	12000	1	0	3.000	0	1	0	;	-19.52914	8.75195
103	N_103	12000	1	0	3.000	0	1	0	;	-17.50389	8.17331
104	N_104	10500	1	0	3.000	0	1	0	;	-9.08948	3.87818
105	N_105	9500	1	0	3.000	0	1	0	;	-9.81664	-.12119
106	N_106	8800	1	0	3.000	0	1	0	;	-8.30773	-3.36800
108	N_108	15000	1	0	3.000	0	1	0	;	-41.58983	2.67622
109	N_109	15000	1	0	3.000	0	1	0	;	-31.31895	2.24223
110	N_110	13000	1	0	3.000	0	1	0	;	-25.67720	1.95291
111	N_111	13000	1	0	3.000	0	1	0	;	-23.79662	2.38689
112	N_112	13000	1	0	3.000	0	1	0	;	-36.38206	22.63933
113	N_113	13000	1	0	3.000	0	1	0	;	-27.55778	17.86555
114	N_114	13000	1	0	3.000	0	1	0	;	-22.35001	14.68302
115	N_115	12000	1	0	3.000	0	1	0	;	-17.28690	11.78981
117	N_117	9600	1	0	3.000	0	1	0	;	7.98102	9.38119
118	N_118	9800	1	0	3.000	0	1	0	;	6.44082	12.04153
119	N_119	9900	1	0	3.000	0	1	0	;	4.06052	16.52210
120	N_120	10000	1	0	3.000	0	1	0	;	3.64046	19.04242
121	N_121	10000	1	0	3.000	0	1	0	;	4.06052	22.54287
123	N_123	9600	1	0	3.000	0	1	0	;	9.80125	9.10116
124	N_124	10000	1	0	3.000	0	1	0	;	11.48146	10.64135
125	N_125	11000	1	0	3.000	0	1	0	;	12.18155	14.00178
126	N_126	12000	1	0	3.000	2	1	1	;	12.88164	17.50223
128	N_128	12000	1	0	3.000	0	1	0	;	15.82201	23.80303
129	N_129	12000	1	0	3.000	0	1	0	;	17.78226	30.52388
131	N_131	12000	1	0	3.000	0	1	0	;	13.02166	25.34322
132	N_132	12000	1	0	3.000	0	1	0	;	13.16167	31.78404
134	N_134	12000	1	0	3.000	0	1	0	;	8.68110	20.86265
135	N_135	12000	1	0	3.000	0	1	0	;	5.74073	24.36310
136	N_136	11000	1	0	3.000	2	1	1	;	-13.79418	6.17122
137	N_137	13000	1	0	3.000	0	1	0	;	-4.52445	-21.72253
138	N_138	11500	1	0	3.000	0	1	0	;	-2.53736	-15.52288
139	N_139	8000	1	0	3.000	0	1	0	;	-31.96047	-150.30821
140	N_140	10000	1	0	3.000	1	1	0	;	-30.45645	-143.54012
141	N_141	1000	1	1	3.000	1	1	0	;	-38.15464	-159.92680
142	N_142	1500	1	1	3.000	1	1	0	;	-37.37277	-161.71568

LINKS Overtake: 0 WakeTurb: 1 AC Cap: 5 Delay: 1 min, Maxlnks: 103

2	LINK2	2	61	37	51	7	1	0	5	1	0	;
3	LINK3	61	62	28	97	7	1	0	5	1	0	;
4	LINK4	62	76	39	139	3	1	0	5	1	0	;
5	LINK5	5	6	88	197	1	1	0	5	1	0	;
6	LINK6	6	72	40	231	1	1	0	5	1	0	;
7	LINK7	7	67	637	161	1	1	0	5	1	0	;
8	LINK8	8	75	230	231	1	1	0	5	1	0	;
9	LINK9	9	10	203	185	5	1	0	5	1	0	;
10	LINK10	10	11	44	194	6	1	0	5	1	0	;
11	LINK11	11	12	51	243	6	1	0	5	1	0	;
12	LINK12	12	142	41	276	8	1	0	5	1	13	;
1	0	2.00	30	;								
13	LINK13	77	86	103	191	7	1	0	5	1	12	;
1	0	2.00	30	;								
14	LINK14	59	140	91	193	7	1	0	5	1	0	;
15	LINK15	16	17	31	213	7	1	0	5	1	0	;
16	LINK16	17	141	36	241	7	1	0	5	1	0	;
17	LINK17	56	20	53	278	8	1	0	5	1	0	;
19	LINK19	20	22	142	289	6	1	0	5	1	0	;
20	LINK20	22	23	147	345	5	1	0	5	1	0	;
21	LINK21	23	24	217	43	1	1	0	5	1	0	;
22	LINK22	24	41	280	17	1	1	0	5	1	0	;
23	LINK23	41	74	190	25	1	1	0	5	1	0	;
24	LINK24	26	27	197	16	1	1	0	5	1	0	;
25	LINK25	27	28	172	19	3	1	0	5	1	0	;
26	LINK26	28	29	38	333	4	1	0	5	1	0	;
27	LINK27	29	30	37	291	5	1	0	5	1	0	;
28	LINK28	30	31	21	41	5	1	0	5	1	0	;
29	LINK29	31	1	62	49	7	1	0	5	1	0	;
30	LINK30	20	34	138	311	6	1	0	5	1	0	;
31	LINK31	34	35	72	347	5	1	0	5	1	0	;
32	LINK32	35	90	104	48	2	1	0	5	1	0	;
33	LINK33	20	38	82	327	6	1	0	5	1	0	;
34	LINK34	38	39	75	0	5	1	0	5	1	0	;
35	LINK35	39	40	85	52	3	1	0	5	1	0	;
36	LINK36	40	73	166	5	1	1	0	5	1	0	;
37	LINK37	31	3	62	47	7	1	0	5	1	0	;
38	LINK38	4	65	65	47	6	1	0	5	1	0	;
40	LINK40	65	66	34	46	6	1	0	5	1	0	;

AIRSPACE.MEX

```

41 LINK41 58 69 55 244 6 1 0 5 1 0 ;
42 LINK42 69 70 49 214 6 1 0 5 1 0 ;
43 LINK43 70 82 44 128 6 1 0 5 1 0 ;
44 LINK44 72 7 262 232 1 1 0 5 1 0 ;
45 LINK45 73 41 254 5 1 1 0 5 1 0 ;
46 LINK46 74 60 410 28 1 1 0 5 1 0 ;
47 LINK47 75 9 230 230 3 1 0 5 1 0 ;
48 LINK48 76 5 201 159 1 1 0 5 1 0 ;
49 LINK49 62 79 40 146 7 1 0 5 1 0 ;
50 LINK50 79 85 144 159 7 1 0 5 1 0 ;
51 LINK51 80 81 86 196 7 1 0 5 1 0 ;
52 LINK52 81 77 303 232 7 1 0 5 1 0 ;
53 LINK53 82 83 53 48 6 1 0 5 1 0 ;
54 LINK54 83 88 56 11 7 1 0 5 1 0 ;
55 LINK55 84 96 559 14 7 1 0 5 1 0 ;
56 LINK56 85 80 54 160 7 1 0 5 1 0 ;
57 LINK57 86 89 482 190 7 1 0 5 1 0 ;
58 LINK58 87 84 165 14 7 1 0 5 1 0 ;
59 LINK59 88 87 322 12 7 1 0 5 1 0 ;
60 LINK60 89 15 319 192 7 1 0 5 1 0 ;
61 LINK61 90 24 117 9 1 1 0 5 1 0 ;
62 LINK62 91 92 178 15 7 1 0 5 1 0 ;
63 LINK63 92 137 80 17 7 1 0 5 1 0 ;
64 LINK64 93 94 31 321 7 1 0 5 1 0 ;
65 LINK65 94 95 23 288 7 1 0 5 1 0 ;
66 LINK66 95 31 20 349 7 1 0 5 1 0 ;
67 LINK67 96 71 142 14 7 1 0 5 1 0 ;
68 LINK68 97 98 99 86 3 1 0 5 1 0 ;
69 LINK69 98 99 66 86 3 1 0 5 1 0 ;
70 LINK70 99 111 41 23 3 1 0 5 1 0 ;
71 LINK71 111 101 55 25 3 1 0 5 1 0 ;
72 LINK72 101 102 24 52 3 1 0 5 1 0 ;
73 LINK73 102 103 21 105 3 1 0 5 1 0 ;
74 LINK74 103 136 42 118 3 1 0 5 1 0 ;
75 LINK75 104 105 41 190 5 1 0 5 1 0 ;
76 LINK76 105 106 36 155 5 1 0 5 1 0 ;
77 LINK77 106 31 34 109 5 1 0 5 1 0 ;
78 LINK78 108 109 103 92 3 1 0 5 1 0 ;
79 LINK79 109 110 56 92 3 1 0 5 1 0 ;
80 LINK80 110 111 19 77 3 1 0 5 1 0 ;
81 LINK81 112 113 100 118 3 1 0 5 1 0 ;
82 LINK82 113 114 61 121 3 1 0 5 1 0 ;
83 LINK83 114 115 58 119 3 1 0 5 1 0 ;
84 LINK84 115 103 36 183 3 1 0 5 1 0 ;
85 LINK85 66 117 19 357 6 1 0 5 1 0 ;
86 LINK86 117 118 31 329 6 1 0 5 1 0 ;
87 LINK87 118 119 51 332 6 1 0 5 1 0 ;
88 LINK88 119 120 26 350 6 1 0 5 1 0 ;
89 LINK89 120 121 35 6 6 1 0 5 1 0 ;
90 LINK90 66 123 24 46 6 1 0 5 1 0 ;
91 LINK91 123 124 23 47 6 1 0 5 1 0 ;
92 LINK92 124 125 34 11 4 1 0 5 1 0 ;
93 LINK93 125 126 36 11 1 1 0 5 1 0 ;
94 LINK94 126 128 70 25 1 1 0 5 1 0 ;
95 LINK95 128 129 70 16 1 1 0 5 1 0 ;
96 LINK96 126 131 78 1 1 1 0 5 1 0 ;
97 LINK97 131 132 64 1 1 1 0 5 1 0 ;
98 LINK98 126 134 54 308 1 1 0 5 1 0 ;
99 LINK99 134 135 46 319 1 1 0 5 1 0 ;
100 LINK100 136 104 52 115 4 1 0 5 1 0 ;
101 LINK101 137 138 65 17 7 1 0 5 1 0 ;
102 LINK102 138 93 62 17 7 1 0 5 1 0 ;
103 LINK103 139 16 55 194 7 1 0 5 1 0 ;
104 LINK104 140 139 69 192 7 1 0 5 1 0 ;
105 LINK105 141 57 25 61 7 1 0 5 1 106
1 0 3.00 30 ;
106 LINK106 142 55 22 96 8 1 0 5 1 105
1 0 3.00 30 ;
ROUTES Def sep dist: 2.0 nm Max route: 22 Nodes/route: 13
1 R_1 ;
2 61 62 76 5 6 72 7 67 ;
;
; C
2 R_2 ;
2 61 62 79 85 80 81 77 86 89 15 ;

```

AIRSPACE.MEX

```
;
; C
3 R_3 ;
56 20 22 23 24 41 74 60 ;
;
; C
4 R_4 ;
56 20 34 35 90 24 41 74 60 ;
;
; C
5 R_5 ;
56 20 38 39 40 73 41 74 60 ;
;
; C
6 R_6 ;
8 75 9 10 11 12 142 55 ;
;
; C
7 R_7 ;
59 140 139 16 17 141 57 ;
;
; C
8 R_3 ;
26 27 28 29 30 31 1 ;
;
; C
9 R_9 ;
26 27 28 29 30 31 3 ;
;
; C
10 R_10 ;
4 65 66 123 124 125 126 134 135 ;
;
; C
11 R_11 ;
58 69 70 82 83 88 87 84 96 71 ;
;
; C
12 R_12 ;
91 92 137 138 93 94 95 31 1 ;
;
; C
13 R_13 ;
91 92 137 138 93 94 95 31 3 ;
;
; C
14 R_14 ;
97 98 99 111 101 102 103 136 104 105 106 31 1 ;
;
; C
15 R_15 ;
108 109 110 111 101 102 103 136 104 105 106 31 1 ;
;
; C
16 R_16 ;
112 113 114 115 103 136 104 105 106 31 1 ;
;
; C
17 ROUTE_17 ;
97 98 99 111 101 102 103 136 104 105 106 31 3 ;
;
; C
18 ROUTE_18 ;
108 109 110 111 101 102 103 136 104 105 106 31 3 ;
;
; C
19 ROUTE_19 ;
112 113 114 115 103 136 104 105 106 31 3 ;
;
; C
20 R_20 ;
4 65 66 117 118 119 120 121 ;
;
; C
21 R_21 ;
```

AIRSPACE.MEX

```
4 65 66 123 124 125 126 128 129 ;
;
; C
22 R_22 ;
4 65 66 123 124 125 126 131 132 ;
;
; C
PROCEDURES DefTimeSep: 50 seconds, DefDistSep: 3.00 nmiles
1 ARR MEX 05R 1 ; ARR AND DEP ON RWY 05R & 05L
1 2 3 4 ; 50 50 50 50 ; 2.00 0. 2.00 0. ;
2 DEP MEX 05R 1
D1 ; DEP ON 05R
1 2 3 4 ; 50 50 50 50 ; 2.00 0. 2.00 0. ;
3 ARR MEX 05L 3 ; ARR ON RWY 05L
1 2 3 4 ; 50 50 50 50 ; 2.00 0. 2.00 0. ;
4 DEP MEX 05L 3
D3 ; DEP ON 05L
1 2 3 4 ; 50 50 50 50 ; 2.00 0. 2.00 0. ;
5 ARR ACA 28X 55 ; OPT ON ACA AIRPORT
1 2 3 4 ; 50 50 50 50 ; 2.00 3.00 0. 0. ;
6 ARR ACA 24X 57 ; ARR ON 24X
1 2 3 4 ; 50 50 50 50 ; 3.00 2.00 0. 0. ;
7 DEP ACA 28X 55
D2 ; DEP ON 28X
1 2 3 4 ; 50 50 50 50 ; 2.00 3.00 0. 0. ;
8 DEP ACA 24X 57
D4 ; DEP ON 24X
1 2 3 4 ; 50 50 50 50 ; 3.00 2.00 0. 0. ;
AIRCRAFT There are 4 aircraft groups
1 GA Min ceiling = 1000 feet, min rwy vis = 2500
I 220.00 I 220.00 I 220.00 I 200.00 I 200.00 I 180.00 I 160.00 I 140.00
I 130.00 ;
I 200.00 I 200.00 I 200.00 I 180.00 I 180.00 I 160.00 I 140.00 I 120.00
I 110.00 ;
I 180.00 I 180.00 I 180.00 I 160.00 I 160.00 I 140.00 I 120.00 I 100.00
I 100.00 ;
5.00 ; 5.00 ; 175 ;
3.00 4.00 5.00 6.00 ;
0. 1.0 .1 1.0 .5 1.2 .9 1.4 1.0 1.4 ;
73 74 75 76 ;
1.00 1.00 1.50 1.50 ;
2 SML Min ceiling = 200 feet, min rwy vis = 1800
I 270.00 I 270.00 I 250.00 I 240.00 I 220.00 I 200.00 I 180.00 I 160.00
I 140.00 ;
I 250.00 I 250.00 I 230.00 I 220.00 I 200.00 I 180.00 I 160.00 I 140.00
I 120.00 ;
I 230.00 I 230.00 I 210.00 I 200.00 I 180.00 I 160.00 I 140.00 I 120.00
I 110.00 ;
5.00 ; 5.00 ; 175 ;
3.00 3.00 4.00 5.00 ;
0. 1.0 .1 1.0 .5 1.2 .9 1.4 1.0 1.4 ;
53 54 56 57 58 59 60 61 62 64 65 66 68 72 ;
1.00 1.00 1.50 1.50 ;
3 LRG Min ceiling = 100 feet, min rwy vis = 1200
I 380.00 I 350.00 I 320.00 I 290.00 I 260.00 I 230.00 I 200.00 I 170.00
I 150.00 ;
I 360.00 I 330.00 I 300.00 I 270.00 I 240.00 I 210.00 I 180.00 I 150.00
I 130.00 ;
I 340.00 I 310.00 I 280.00 I 250.00 I 220.00 I 190.00 I 160.00 I 130.00
I 120.00 ;
5.00 ; 5.00 ; 210 ;
3.00 3.00 3.00 4.00 ;
0. 1.0 .1 1.0 .5 1.2 .9 1.4 1.0 1.4 ;
15 24 25 26 27 28 30 37 38 39 40 41 42 43 44 45 46 47 48 49 51 52 ;
1.00 1.00 1.00 1.50 ;
4 HVY Min ceiling = 0 feet, min rwy vis = 700
I 380.00 I 350.00 I 320.00 I 290.00 I 260.00 I 230.00 I 200.00 I 170.00
I 150.00 ;
I 360.00 I 330.00 I 300.00 I 270.00 I 240.00 I 210.00 I 180.00 I 150.00
I 130.00 ;
I 340.00 I 310.00 I 280.00 I 250.00 I 220.00 I 190.00 I 160.00 I 130.00
I 120.00 ;
5.00 ; 5.00 ; 210 ;
3.00 3.00 3.00 3.00 ;
0. 1.0 .1 1.0 .5 1.2 .9 1.4 1.0 1.4 ;
```

AIRSPACE.MEX

```
1 2 3 4 5 14 18 19 20 23 31 32 33 ;
.50 1.00 1.00 1.00 ;
AIRPORTS Def ceiling: 5000 ft, Def rwy vis range: 30000 ft
          Def node gap: 10 sec, Def spread factor: 20 %
1 MEX    Elev = 7361    1 2 3 4 ;
10 10 10 10 ; 50 50 50 50 ;
    OPERATIONS ON MEX AIRPORT
1 1 ; ; ;
1 ; 2 ; ;
1 3 ; ; ;
1 ; 4 ; ;
2 ACA    Elev = 0      55 56 57 58 ;
10 10 10 10 ; 50 50 50 50 ;
    OPERATIONS ON ACA AIRPORT
1 5 ; ; ;
1 ; 7 ; ;
1 6 ; ; ;
1 ; 8 ; ;

GO
```

GROUND.MEX

PRINT 1

AFNODES In this run, 89 is the largest node number

MEX 1 2 3 4 5 9 10 11 12 14 15 16 17 18 19 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 39 40 41 42 43 44 45 46 47 48 49 50 51 52
ACA 53 54 55 56 57 58 59 61 62 65 66 68 69 70 72 74 75 76 77 78 79 80 81
82 83 84 85 86 87 88 89

COORDINATES This data set has 77 nodes with defined coordinates.

1	-2665	-2728	MEX
2	5934	4337	MEX
3	-3164	-2092	MEX
4	4653	4218	MEX
5	6607	-1430	MEX
9	2298	1358	MEX
10	6172	-1937	MEX
11	5163	3705	MEX
12	4247	4706	MEX
14	-2309	-3207	MEX
15	2769	958	MEX
16	4886	3984	MEX
17	5645	4616	MEX
18	2245	2280	MEX
19	2824	1789	MEX
21	-1788	-2776	MEX
22	-1934	-301	MEX
23	-658	-1850	MEX
24	-1141	1350	MEX
25	1191	-1448	MEX
26	666	1821	MEX
27	1959	288	MEX
28	2957	3673	MEX
29	3859	2637	MEX
30	3536	305	MEX
31	3989	798	MEX
32	4438	-462	MEX
33	4859	58	MEX
34	5287	-1184	MEX
35	5719	-674	MEX
36	-2999	-1170	MEX
37	-2999	-1171	MEX
39	-2999	-1171	MEX
40	-3539	-1610	MEX
41	1759	2712	MEX
42	-2618	-1675	MEX
43	-2149	-2303	MEX
44	-1534	-787	MEX
45	-1031	-1383	MEX
46	-239	265	MEX
47	249	-320	MEX
48	-648	748	MEX
49	648	-798	MEX
50	1075	1340	MEX
51	1573	754	MEX
52	3375	3191	MEX
53	-247535	-979926	ACA
54	-247406	-979312	ACA
55	-240268	-981102	ACA
56	-250127	-979459	ACA
57	-245254	-979005	ACA
58	-249729	-981708	ACA
59	-246912	-980033	ACA
61	-250036	-978861	ACA
62	-240152	-980563	ACA
65	-249925	-981384	ACA
66	-245473	-978662	ACA
68	-249235	-980961	ACA
69	-249034	-981295	ACA
70	-248550	-980544	ACA
72	-247848	-980117	ACA
74	-248709	-979089	ACA
75	-248830	-979691	ACA
76	-244855	-979750	ACA
77	-244955	-980348	ACA
78	-243046	-980063	ACA
79	-243163	-980636	ACA
80	-241289	-980367	ACA

GROUND.MEX

```
81      -241400      -980920 ACA
82      -248066      -979199 ACA
83      -247902      -978507 ACA
84      -246004      -978994 ACA
85      -246307      -978489 ACA
86      -246718      -979429 ACA
87      -246099      -979535 ACA
88      -247627      -980458 ACA
89      -248354      -980890 ACA
AFLINKS  TaxiSpeed: 15  DefCap: 2  PassingCode: 2  MaxLnk: 116
1      1 43 50 668 0 2 1 4 0 15 ;
1      1 43 50 668 0 2 2 4 0 15 ;
1      1 43 50 668 0 2 2 4 0 15 ;
1      1 43 50 668 0 2 2 4 0 15 ;
2      3 42 52 687 0 33 2 4 0 15 ;
3      5 35 310 1166 0 23 2 4 0 15 ;
4      10 5 40 668 0 3 2 4 0 15 ;
5      9 15 130 618 0 2 2 4 0 15 ;
6      11 16 315 393 0 2 2 4 0 15 ;
7      12 28 231 1653 0 6 2 4 0 15 ;
8      40 3 142 611 0 2 2 4 0 15 ;
9      14 21 50 676 0 2 2 4 0 15 ;
11     9 19 50 680 0 2 2 4 0 15 ;
12     15 30 130 1007 0 3 2 4 0 15 ;
13     1 14 143 597 0 2 2 4 0 15 ;
14     3 1 141 808 0 3 2 4 0 15 ;
15     11 2 50 997 0 3 2 4 0 15 ;
16     4 12 320 635 0 2 2 4 0 15 ;
17     16 17 50 988 0 3 2 4 0 15 ;
18     17 2 133 402 0 2 2 4 0 15 ;
19     16 4 315 330 0 2 2 4 0 15 ;
20     18 52 51 1451 0 5 2 4 0 15 ;
21     19 18 310 759 0 3 2 4 0 15 ;
22     18 41 311 650 0 2 2 4 0 15 ;
23     19 29 50 1338 0 4 2 4 0 15 ;
25     22 44 140 629 0 2 2 4 0 15 ;
26     24 48 140 778 0 3 2 4 0 15 ;
27     26 50 139 631 0 2 2 4 0 15 ;
28     28 52 139 638 0 2 2 4 0 15 ;
29     30 31 42 670 0 2 2 4 0 15 ;
30     32 33 38 669 0 2 2 4 0 15 ;
31     34 35 40 668 0 2 2 4 0 15 ;
32     36 42 142 633 0 2 2 4 0 15 ;
33     37 36 0 1 0 2 2 4 0 15 ;
35     39 37 0 0 0 2 2 4 0 15 ;
36     40 36 50 697 0 2 2 4 0 15 ;
37     42 44 50 1401 0 5 2 4 0 15 ;
38     43 21 142 595 0 2 2 4 0 15 ;
39     36 22 50 1375 0 5 2 4 0 15 ;
40     42 43 143 784 0 3 2 4 0 15 ;
41     43 45 50 1448 0 5 2 4 0 15 ;
42     21 23 50 1461 0 5 2 4 0 15 ;
43     22 48 50 1660 0 6 2 4 0 15 ;
44     23 49 51 1677 0 6 2 4 0 15 ;
45     44 46 50 1668 0 6 2 4 0 15 ;
46     45 23 141 598 0 2 2 4 0 15 ;
47     44 45 139 780 0 3 2 4 0 15 ;
48     45 47 50 1664 0 6 2 4 0 15 ;
49     46 47 140 762 0 3 2 4 0 15 ;
50     47 51 50 1705 0 6 2 4 0 15 ;
51     48 46 139 633 0 2 2 4 0 15 ;
52     49 27 50 1702 0 6 2 4 0 15 ;
53     48 26 50 1696 0 6 2 4 0 15 ;
54     46 50 50 1698 0 6 2 4 0 15 ;
55     47 49 140 623 0 2 2 4 0 15 ;
56     49 25 140 847 0 3 2 4 0 15 ;
57     50 18 51 1501 0 5 2 4 0 15 ;
58     51 9 50 944 0 3 2 4 0 15 ;
59     26 41 50 1410 0 5 2 4 0 15 ;
60     50 51 139 769 0 3 2 4 0 15 ;
61     51 27 140 605 0 2 2 4 0 15 ;
62     27 15 50 1051 0 4 2 4 0 15 ;
63     52 4 51 1640 0 5 2 4 0 15 ;
64     28 41 231 1536 0 5 2 4 0 15 ;
65     52 29 138 736 0 2 2 4 0 15 ;
```

GROUND.MEX

```
66 29 11 50 1686 0 6 2 4 0 15 ;
67 31 19 310 1529 0 5 2 4 0 15 ;
68 30 32 130 1184 0 4 2 4 0 15 ;
69 33 31 310 1142 0 4 2 4 0 15 ;
70 32 34 130 1114 0 4 2 4 0 15 ;
71 35 33 310 1129 0 4 2 4 0 15 ;
72 34 10 130 1162 0 4 2 4 0 15 ;
73 53 54 11 627 0 2 2 4 0 15 ;
74 55 81 279 1147 0 4 2 4 0 15 ;
75 57 87 237 997 0 3 2 4 0 15 ;
76 59 53 279 632 0 2 2 4 0 15 ;
77 59 88 239 832 0 3 2 4 0 15 ;
78 56 61 8 605 0 2 2 4 0 15 ;
79 61 74 99 1346 0 4 2 4 0 15 ;
80 62 55 192 551 0 2 2 4 0 15 ;
81 58 65 328 379 0 2 2 4 0 15 ;
82 65 68 58 809 0 3 2 4 0 15 ;
83 66 57 147 407 0 2 2 4 0 15 ;
84 68 69 148 390 0 2 2 4 0 15 ;
85 70 89 150 398 0 2 2 4 0 15 ;
86 72 88 147 406 0 2 2 4 0 15 ;
87 74 75 191 614 0 2 2 4 0 15 ;
88 76 77 189 606 0 2 2 4 0 15 ;
89 78 79 191 585 0 2 2 4 0 15 ;
90 80 81 191 564 0 2 2 4 0 15 ;
91 82 83 13 711 0 2 2 4 0 15 ;
92 84 85 329 589 0 2 2 4 0 15 ;
93 74 82 99 652 0 2 2 4 0 15 ;
94 75 56 280 1318 0 4 2 4 0 15 ;
95 82 54 99 670 0 2 2 4 0 15 ;
96 54 86 99 698 0 2 2 4 0 15 ;
97 53 75 280 1316 0 4 2 4 0 15 ;
98 53 86 58 956 0 3 2 4 0 15 ;
99 68 70 58 802 0 3 2 4 0 15 ;
100 69 58 239 808 0 3 2 4 0 15 ;
101 70 72 58 822 0 3 2 4 0 15 ;
102 89 69 239 791 0 3 2 4 0 15 ;
103 72 53 58 367 0 2 2 4 0 15 ;
105 84 66 57 626 0 2 2 4 0 15 ;
106 86 87 99 628 0 2 2 4 0 15 ;
107 87 59 238 953 0 3 2 4 0 15 ;
108 86 84 58 836 0 3 2 4 0 15 ;
109 87 76 99 1262 0 4 2 4 0 15 ;
110 80 62 99 1154 0 4 2 4 0 15 ;
111 81 79 279 1786 0 6 2 4 0 15 ;
112 78 80 99 1783 0 6 2 4 0 15 ;
113 79 77 279 1815 0 6 2 4 0 15 ;
114 77 59 279 1982 0 7 2 4 0 15 ;
115 76 78 99 1836 0 6 2 4 0 15 ;
116 88 89 239 846 0 3 2 4 0 15 ;
RUNWAYS Taxi Speed = 35 mph. Crossing delay = 5
1 05R 23L 50 0 0 15 1 41 48 50 58 11 23 66 15 MEX
2 05L 23R 51 0 0 15 2 37 45 54 57 20 63 MEX
3 31X 13X 309 0 0 15 3 71 69 67 21 22 MEX
4 28X 10X 279 0 0 15 74 111 113 114 76 97 94 ACA
5 24X 06X 238 0 0 15 75 107 77 116 102 100 ACA
DEPARTQ There are 4 departure queues, with up to 4 routes
1 0 10 0 MEX D1 1 0
1 2 ;
2 0 10 0 ACA D2 55 0
3 4 5 ;
3 0 10 0 MEX D3 3 0
10 20 21 22 ;
4 0 10 0 ACA D4 57 0
11 ;
GATES There are 4 boarding gates in this simulation
1 24 4 20 MEX-1 MEX YES 0 ALL 26 ;
; ;
; ;
2 25 4 20 MEX-2 MEX YES 0 ALL 56 ;
; ;
; ;
3 83 4 20 ACA-1 ACA YES 0 ALL 91 ;
```

GROUND.MEX

```

; ;
; ;
; ;
4 85 4 20 ACA-2      ACA YES 0 ALL 92 ;
; ;
; ;
; ;
TAXIPATHS These are the user defined taxi paths.
1 DEPM-2 56 44 42 9 13 ;
TAMPS      There are 4 ground data groups in this run
1 30 15
0. 1200 1.00 1200 ;
0. 2000 1.00 2000 ;
0. 15 1.00 15 ;
0. 10 1.00 10 ;
76 77 60 73 74 75 ;
2 30 15
0. 3500 1.00 3500 ;
0. 4000 1.00 4000 ;
0. 15 1.00 15 ;
0. 10 1.00 10 ;
37 38 41 44 53 54 55 56 57 65 66 67 68 69 71 79 80 58 39 59 61
62 64 72 ;
3 30 15
0. 4000 1.00 4000 ;
0. 5000 1.00 5000 ;
0. 22 1.00 22 ;
0. 22 1.00 22 ;
6 7 8 10 11 12 24 25 26 27 28 29 30 34 40 42 43 45 46 47 48 51
52 63 81 70 78 81 15 35 36 49 50 ;
4 30 15
0. 4500 1.00 4500 ;
0. 5500 1.00 5500 ;
0. 35 1.00 35 ;
0. 35 1.00 35 ;
1 2 3 4 9 13 14 14 16 17 18 19 20 21 22 23 31 32 5 33 ;
AIRLINES These are the airlines serving the airports in this run.
1 AMX AIRL
2 MXN AIRL
COST 35 50
GO
```

LOG.MEX

START SIMULATION LOG AT 01:49:56 FOR ITERATION #1 OF:
08/10/1996 AT 01:49:43 (0) Mexico City International

ITERATION 1 STARTING RANDOM SEED VALUES FOR EACH OF 10 STREAMS ARE:
1-2116429302 2-683743814 3-964393174 4-1217426631 5-618433579
6-1157240309 7-15726055 8-48108509 9-1797920909 10-477424540

1 00:00:00; NEW PLAN IN EFFECT IS 1, WITH PLAN FLAG = 0 (SETPLAN-000)
1 00:00:00; DEACTIVATED RUNWAYS: MEX 23L/(05R) MEX 23R/(05L) MEX 13X/(31X) ACA 10X/(28X) ACA 06X/(24X) (SETPLAN-000)
1 00:00:00; NEWLY ACTIVATED RUNWAYS: MEX 05R/(23L) MEX 05L/(23R) MEX 31X/(13X) ACA 28X/(10X) ACA 24X/(06X) (SETPLAN-000)
1 00:30:00; TRACE VALUES HAVE NOT BEEN RESET AT REAL TIME 01:49:56 (FROM INDEX 0) FOR ITERATION 1 (TRACE-000)
1 01:00:00; TRACE VALUES HAVE NOT BEEN RESET AT REAL TIME 01:49:56 (FROM INDEX 0) FOR ITERATION 1 (TRACE-000)
1 01:30:00; TRACE VALUES HAVE NOT BEEN RESET AT REAL TIME 01:49:56 (FROM INDEX 0) FOR ITERATION 1 (TRACE-000)
1 01:30:00; TRACE VALUES RESET AT REAL TIME 01:49:56 (FROM INDEX 182) AS 1; (TRACE-000)
1 01:30:00; TRACE VALUES RESET AT REAL TIME 01:49:56 (FROM INDEX 1) AS 1; (TRACE-000)
1 01:30:00; TRACE VALUES RESET AT REAL TIME 01:49:56 (FROM INDEX 2) AS 1; (TRACE-000)
1 01:30:00; TRACE VALUES RESET AT REAL TIME 01:49:56 (FROM INDEX 181) AS 1; (TRACE-000)
1 01:30:00; TRACE VALUES RESET AT REAL TIME 01:49:56 (FROM INDEX 3) AS 1; (TRACE-000)
1 01:30:00; TRACE VALUES RESET AT REAL TIME 01:49:56 (FROM INDEX 12) AS 1; (TRACE-000)
1 01:30:00; TRACE VALUES RESET AT REAL TIME 01:49:56 (FROM INDEX 4) AS 1; (TRACE-000)
1 01:30:00; TRACE VALUES RESET AT REAL TIME 01:49:56 (FROM INDEX 5) AS 1; (TRACE-000)
1 01:30:00; TRACE VALUES RESET AT REAL TIME 01:49:56 (FROM INDEX 6) AS 1; (TRACE-000)
1 01:30:00; TRACE VALUES RESET AT REAL TIME 01:49:56 (FROM INDEX 9) AS 1; (TRACE-000)
1 01:30:00; TRACE VALUES RESET AT REAL TIME 01:49:56 (FROM INDEX 10) AS 1; (TRACE-000)
1 02:00:00; TRACE VALUES HAVE NOT BEEN RESET AT REAL TIME 01:49:56 (FROM INDEX 0) FOR ITERATION 1 (TRACE-000)
1 02:30:00; TRACE VALUES HAVE NOT BEEN RESET AT REAL TIME 01:49:56 (FROM INDEX 0) FOR ITERATION 1 (TRACE-000)
1 03:00:00; TRACE VALUES HAVE NOT BEEN RESET AT REAL TIME 01:49:56 (FROM INDEX 0) FOR ITERATION 1 (TRACE-000)
1 03:30:00; TRACE VALUES HAVE NOT BEEN RESET AT REAL TIME 01:49:56 (FROM INDEX 0) FOR ITERATION 1 (TRACE-000)
1 04:00:00; TRACE VALUES HAVE NOT BEEN RESET AT REAL TIME 01:49:56 (FROM INDEX 0) FOR ITERATION 1 (TRACE-000)
1 04:30:00; TRACE VALUES HAVE NOT BEEN RESET AT REAL TIME 01:49:56 (FROM INDEX 0) FOR ITERATION 1 (TRACE-000)
1 05:00:00; TRACE VALUES HAVE NOT BEEN RESET AT REAL TIME 01:49:56 (FROM INDEX 0) FOR ITERATION 1 (TRACE-000)
1 05:30:00; TRACE VALUES HAVE NOT BEEN RESET AT REAL TIME 01:49:56 (FROM INDEX 0) FOR ITERATION 1 (TRACE-000)
1 06:00:00; TRACE VALUES HAVE NOT BEEN RESET AT REAL TIME 01:49:56 (FROM INDEX 0) FOR ITERATION 1 (TRACE-000)
1 06:00:22; MXN MA104_AAMEX CREATED ENROUTE TO MEX AIRPORT; 0 SEC LATE; TAIL 1.40; B747SP/JT9DFL (ARRIVAL-021)
1 06:00:22; MXN MA104_AAMEX OF ATC GROUP HVY HAS BEEN INJECTED ON ROUTE 14 AT NODE 97; B747SP/JT9DFL (INJECT-029)
1 06:00:22; MXN MA104_AAMEX LEAVING NODE 97 FOR NODE 98 ON ROUTE 14; ARRIVAL TIME IS 06:01:58 (NODE.DEP-078)
1 06:00:55; MXN MA101_AAMEX CREATED ENROUTE TO MEX AIRPORT; 0 SEC LATE; TAIL 1.24; B747SP/JT9DFL (ARRIVAL-021)
1 06:00:55; MXN MA101_AAMEX OF ATC GROUP HVY HAS BEEN INJECTED ON ROUTE 14 AT NODE 97; B747SP/JT9DFL (INJECT-029)
1 06:00:55; MXN MA101_AAMEX LEAVING NODE 97 FOR NODE 98 ON ROUTE 14; ARRIVAL TIME IS 06:02:38 (NODE.DEP-078)
1 06:00:56; MXN MA102_AAMEX CREATED ENROUTE TO MEX AIRPORT; 0 SEC LATE; TAIL 1.40; B747SP/JT9DFL (ARRIVAL-021)
1 06:00:56; MXN MA102_AAMEX OF ATC GROUP HVY HAS BEEN INJECTED ON ROUTE 14 AT NODE 97; B747SP/JT9DFL (INJECT-029)
1 06:00:56; MXN MA102_AAMEX MUST HOLD AT NODE 97 BECAUSE OF WAKE TURBULENCE THERE (HOLD.CHECK-081)
1 06:00:56; MXN MA102_AAMEX PLACED IN HOLDING AT NODE 97 WHERE THE HOLD COUNT IS NOW 1 AIRCRAFT (HOLD.CHECK-083)
1 06:01:24; MXN MA102_AAMEX IS RELEASED FROM HOLDING AT NODE 97 AFTER A DELAY OF 29 SECONDS (MOVE.CHK-084)
1 06:01:24; MXN MA102_AAMEX IS RELEASED FROM HOLDING AT NODE 97 AFTER A TOTAL HOLD TIME OF 29 SECONDS (NODE.DEP-085)
1 06:01:24; MXN MA102_AAMEX LEAVING NODE 97 FOR NODE 98 ON ROUTE 14; ARRIVAL TIME IS 06:03:14 (NODE.DEP-078)
1 06:01:28; MXN MA105_AAMEX CREATED ENROUTE TO MEX AIRPORT; 0 SEC LATE; TAIL 1.26; B747SP/JT9DFL (ARRIVAL-021)
1 06:01:28; MXN MA105_AAMEX OF ATC GROUP HVY HAS BEEN INJECTED ON ROUTE 14 AT NODE 97; B747SP/JT9DFL (INJECT-029)
1 06:01:28; MXN MA105_AAMEX MUST HOLD AT NODE 97 BECAUSE OF WAKE TURBULENCE THERE (HOLD.CHECK-081)
1 06:01:28; MXN MA105_AAMEX PLACED IN HOLDING AT NODE 97 WHERE THE HOLD COUNT IS NOW 1 AIRCRAFT (HOLD.CHECK-083)
1 06:01:53; MXN MA105_AAMEX IS RELEASED FROM HOLDING AT NODE 97 AFTER A DELAY OF 26 SECONDS (MOVE.CHK-084)
1 06:01:53; MXN MA105_AAMEX IS RELEASED FROM HOLDING AT NODE 97 AFTER A TOTAL HOLD TIME OF 26 SECONDS (NODE.DEP-085)
1 06:01:53; MXN MA105_AAMEX LEAVING NODE 97 FOR NODE 98 ON ROUTE 14; ARRIVAL TIME IS 06:03:54 (NODE.DEP-078)
1 06:01:58; MXN MA104_AAMEX ON ROUTE 14 HAS ARRIVED AT NODE 98 (NODE.ARR-071)
1 06:01:58; MXN MA104_AAMEX LEAVING NODE 98 FOR NODE 99 ON ROUTE 14; ARRIVAL TIME IS 06:03:02 (NODE.DEP-078)
1 06:02:11; MXN MA106_AAMEX CREATED ENROUTE TO MEX AIRPORT; 0 SEC LATE; TAIL 1.06; B747SP/JT9DFL (ARRIVAL-021)
1 06:02:11; MXN MA106_AAMEX OF ATC GROUP HVY HAS BEEN INJECTED ON ROUTE 14 AT NODE 97; B747SP/JT9DFL (INJECT-029)
1 06:02:11; MXN MA106_AAMEX MUST HOLD AT NODE 97 BECAUSE OF WAKE TURBULENCE THERE (HOLD.CHECK-081)
1 06:02:11; MXN MA106_AAMEX PLACED IN HOLDING AT NODE 97 WHERE THE HOLD COUNT IS NOW 1 AIRCRAFT (HOLD.CHECK-083)
1 06:02:22; MXN MA106_AAMEX IS RELEASED FROM HOLDING AT NODE 97 AFTER A DELAY OF 11 SECONDS (MOVE.CHK-084)
1 06:02:22; MXN MA106_AAMEX IS RELEASED FROM HOLDING AT NODE 97 AFTER A TOTAL HOLD TIME OF 11 SECONDS (NODE.DEP-085)
1 06:02:22; MXN MA106_AAMEX LEAVING NODE 97 FOR NODE 98 ON ROUTE 14; ARRIVAL TIME IS 06:04:31 (NODE.DEP-078)
1 06:02:38; MXN MA101_AAMEX ON ROUTE 14 HAS ARRIVED AT NODE 98 (NODE.ARR-071)
1 06:02:38; MXN MA101_AAMEX LEAVING NODE 98 FOR NODE 99 ON ROUTE 14; ARRIVAL TIME IS 06:03:43 (NODE.DEP-078)
1 06:03:02; MXN MA104_AAMEX ON ROUTE 14 HAS ARRIVED AT NODE 99 (NODE.ARR-071)
1 06:03:02; MXN MA104_AAMEX LEAVING NODE 99 FOR NODE 111 ON ROUTE 14; ARRIVAL TIME IS 06:03:43 (NODE.DEP-078)
1 06:03:14; MXN MA102_AAMEX ON ROUTE 14 HAS ARRIVED AT NODE 98 (NODE.ARR-071)
1 06:03:14; MXN MA102_AAMEX LEAVING NODE 98 FOR NODE 99 ON ROUTE 14; ARRIVAL TIME IS 06:04:19 (NODE.DEP-078)
1 06:03:43; MXN MA104_AAMEX ON ROUTE 14 HAS ARRIVED AT NODE 111 (NODE.ARR-071)
1 06:03:43; MXN MA104_AAMEX LEAVING NODE 111 FOR NODE 101 ON ROUTE 14; ARRIVAL TIME IS 06:04:38 (NODE.DEP-078)
1 06:03:43; MXN MA101_AAMEX ON ROUTE 14 HAS ARRIVED AT NODE 99 (NODE.ARR-071)
1 06:03:43; MXN MA101_AAMEX LEAVING NODE 99 FOR NODE 111 ON ROUTE 14; ARRIVAL TIME IS 06:04:24 (NODE.DEP-078)
1 06:03:54; MXN MA105_AAMEX ON ROUTE 14 HAS ARRIVED AT NODE 98 (NODE.ARR-071)
1 06:03:54; MXN MA105_AAMEX LEAVING NODE 98 FOR NODE 99 ON ROUTE 14; ARRIVAL TIME IS 06:05:00 (NODE.DEP-078)
1 06:04:19; MXN MA102_AAMEX ON ROUTE 14 HAS ARRIVED AT NODE 99 (NODE.ARR-071)

LOG.MEX

1 06:04:19; MXN MA102_AMEX LEAVING NODE 99 FOR NODE 111 ON ROUTE 14; ARRIVAL TIME IS 06:05:00 (NODE.DEP-078)
1 06:04:23; MXN MA103_AMEX CREATED ENROUTE TO MEX AIRPORT; 0 SEC LATE; TAIL 1.26; B747SP/JT9DFL (ARRIVAL-021)
1 06:04:23; MXN MA103_AMEX OF ATC GROUP HVY HAS BEEN INJECTED ON ROUTE 14 AT NODE 97; B747SP/JT9DFL (INJECT-029)
1 06:04:23; MXN MA103_AMEX LEAVING NODE 97 FOR NODE 98 ON ROUTE 14; ARRIVAL TIME IS 06:05:58 (NODE.DEP-078)
1 06:04:24; MXN MA101_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 111 (NODE.ARR-071)
1 06:04:24; MXN MA101_AMEX LEAVING NODE 111 FOR NODE 101 ON ROUTE 14; ARRIVAL TIME IS 06:05:20 (NODE.DEP-078)
1 06:04:31; MXN MA106_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 98 (NODE.ARR-071)
1 06:04:31; MXN MA106_AMEX LEAVING NODE 98 FOR NODE 99 ON ROUTE 14; ARRIVAL TIME IS 06:05:37 (NODE.DEP-078)
1 06:04:38; MXN MA104_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 101 (NODE.ARR-071)
1 06:04:38; MXN MA104_AMEX LEAVING NODE 101 FOR NODE 102 ON ROUTE 14; ARRIVAL TIME IS 06:05:02 (NODE.DEP-078)
1 06:05:00; MXN MA105_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 99 (NODE.ARR-071)
1 06:05:00; MXN MA105_AMEX LEAVING NODE 99 FOR NODE 111 ON ROUTE 14; ARRIVAL TIME IS 06:05:42 (NODE.DEP-078)
1 06:05:00; MXN MA102_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 111 (NODE.ARR-071)
1 06:05:00; MXN MA102_AMEX LEAVING NODE 111 FOR NODE 101 ON ROUTE 14; ARRIVAL TIME IS 06:05:57 (NODE.DEP-078)
1 06:05:02; MXN MA104_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 102 (NODE.ARR-071)
1 06:05:02; MXN MA104_AMEX LEAVING NODE 102 FOR NODE 103 ON ROUTE 14; ARRIVAL TIME IS 06:05:23 (NODE.DEP-078)
1 06:05:20; MXN MA101_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 101 (NODE.ARR-071)
1 06:05:20; MXN MA101_AMEX LEAVING NODE 101 FOR NODE 102 ON ROUTE 14; ARRIVAL TIME IS 06:05:44 (NODE.DEP-078)
1 06:05:23; MXN MA104_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 103 (NODE.ARR-071)
1 06:05:23; MXN MA104_AMEX LEAVING NODE 103 FOR NODE 136 ON ROUTE 14; ARRIVAL TIME IS 06:06:06 (NODE.DEP-078)
1 06:05:37; MXN MA106_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 99 (NODE.ARR-071)
1 06:05:37; MXN MA106_AMEX LEAVING NODE 99 FOR NODE 111 ON ROUTE 14; ARRIVAL TIME IS 06:06:20 (NODE.DEP-078)
1 06:05:42; MXN MA105_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 111 (NODE.ARR-071)
1 06:05:42; MXN MA105_AMEX LEAVING NODE 111 FOR NODE 101 ON ROUTE 14; ARRIVAL TIME IS 06:06:39 (NODE.DEP-078)
1 06:05:44; MXN MA101_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 102 (NODE.ARR-071)
1 06:05:44; MXN MA101_AMEX LEAVING NODE 102 FOR NODE 103 ON ROUTE 14; ARRIVAL TIME IS 06:06:05 (NODE.DEP-078)
1 06:05:57; MXN MA102_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 101 (NODE.ARR-071)
1 06:05:57; MXN MA102_AMEX LEAVING NODE 101 FOR NODE 102 ON ROUTE 14; ARRIVAL TIME IS 06:06:21 (NODE.DEP-078)
1 06:05:58; MXN MA103_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 98 (NODE.ARR-071)
1 06:05:58; MXN MA103_AMEX LEAVING NODE 98 FOR NODE 99 ON ROUTE 14; ARRIVAL TIME IS 06:07:02 (NODE.DEP-078)
1 06:06:05; MXN MA101_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 103 (NODE.ARR-071)
1 06:06:05; MXN MA101_AMEX LEAVING NODE 103 FOR NODE 136 ON ROUTE 14; ARRIVAL TIME IS 06:06:49 (NODE.DEP-078)
1 06:06:06; MXN MA104_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 136 (NODE.ARR-071)
1 06:06:06; MXN MA104_AMEX LEAVING NODE 136 FOR NODE 104 ON ROUTE 14; ARRIVAL TIME IS 06:07:06 (NODE.DEP-078)
1 06:06:20; MXN MA106_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 111 (NODE.ARR-071)
1 06:06:20; MXN MA106_AMEX LEAVING NODE 111 FOR NODE 101 ON ROUTE 14; ARRIVAL TIME IS 06:07:17 (NODE.DEP-078)
1 06:06:21; MXN MA102_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 102 (NODE.ARR-071)
1 06:06:21; MXN MA102_AMEX LEAVING NODE 102 FOR NODE 103 ON ROUTE 14; ARRIVAL TIME IS 06:06:43 (NODE.DEP-078)
1 06:06:39; MXN MA105_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 101 (NODE.ARR-071)
1 06:06:39; MXN MA105_AMEX LEAVING NODE 101 FOR NODE 102 ON ROUTE 14; ARRIVAL TIME IS 06:07:03 (NODE.DEP-078)
1 06:06:43; MXN MA102_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 103 (NODE.ARR-071)
1 06:06:43; MXN MA102_AMEX LEAVING NODE 103 FOR NODE 136 ON ROUTE 14; ARRIVAL TIME IS 06:07:27 (NODE.DEP-078)
1 06:06:49; MXN MA101_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 136 (NODE.ARR-071)
1 06:06:49; MXN MA101_AMEX LEAVING NODE 136 FOR NODE 104 ON ROUTE 14; ARRIVAL TIME IS 06:07:54 (NODE.DEP-078)
1 06:07:02; MXN MA103_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 99 (NODE.ARR-071)
1 06:07:02; MXN MA103_AMEX LEAVING NODE 99 FOR NODE 111 ON ROUTE 14; ARRIVAL TIME IS 06:07:43 (NODE.DEP-078)
1 06:07:03; MXN MA105_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 102 (NODE.ARR-071)
1 06:07:03; MXN MA105_AMEX LEAVING NODE 102 FOR NODE 103 ON ROUTE 14; ARRIVAL TIME IS 06:07:25 (NODE.DEP-078)
1 06:07:06; MXN MA104_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 104 (NODE.ARR-071)
1 06:07:06; MXN MA104_AMEX LEAVING NODE 104 FOR NODE 105 ON ROUTE 14; ARRIVAL TIME IS 06:07:59 (NODE.DEP-078)
1 06:07:17; MXN MA106_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 101 (NODE.ARR-071)
1 06:07:17; MXN MA106_AMEX LEAVING NODE 101 FOR NODE 102 ON ROUTE 14; ARRIVAL TIME IS 06:07:41 (NODE.DEP-078)
1 06:07:25; MXN MA105_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 103 (NODE.ARR-071)
1 06:07:25; MXN MA105_AMEX LEAVING NODE 103 FOR NODE 136 ON ROUTE 14; ARRIVAL TIME IS 06:08:10 (NODE.DEP-078)
1 06:07:27; MXN MA102_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 136 (NODE.ARR-071)
1 06:07:27; MXN MA102_AMEX LEAVING NODE 136 FOR NODE 104 ON ROUTE 14; ARRIVAL TIME IS 06:08:36 (NODE.DEP-078)
1 06:07:41; MXN MA106_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 102 (NODE.ARR-071)
1 06:07:41; MXN MA106_AMEX LEAVING NODE 102 FOR NODE 103 ON ROUTE 14; ARRIVAL TIME IS 06:08:03 (NODE.DEP-078)
1 06:07:43; MXN MA103_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 111 (NODE.ARR-071)
1 06:07:43; MXN MA103_AMEX LEAVING NODE 111 FOR NODE 101 ON ROUTE 14; ARRIVAL TIME IS 06:08:38 (NODE.DEP-078)
1 06:07:54; MXN MA101_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 104 (NODE.ARR-071)
1 06:07:54; MXN MA101_AMEX LEAVING NODE 104 FOR NODE 105 ON ROUTE 14; ARRIVAL TIME IS 06:08:53 (NODE.DEP-078)
1 06:07:59; MXN MA104_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 105 (NODE.ARR-071)
1 06:07:59; MXN MA104_AMEX LEAVING NODE 105 FOR NODE 106 ON ROUTE 14; ARRIVAL TIME IS 06:08:46 (NODE.DEP-078)
1 06:08:03; MXN MA106_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 103 (NODE.ARR-071)
1 06:08:03; MXN MA106_AMEX LEAVING NODE 103 FOR NODE 136 ON ROUTE 14; ARRIVAL TIME IS 06:08:48 (NODE.DEP-078)
1 06:08:10; MXN MA105_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 136 (NODE.ARR-071)
1 06:08:10; MXN MA105_AMEX LEAVING NODE 136 FOR NODE 104 ON ROUTE 14; ARRIVAL TIME IS 06:09:24 (NODE.DEP-078)
1 06:08:36; MXN MA102_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 104 (NODE.ARR-071)
1 06:08:36; MXN MA102_AMEX LEAVING NODE 104 FOR NODE 105 ON ROUTE 14; ARRIVAL TIME IS 06:09:42 (NODE.DEP-078)
1 06:08:38; MXN MA103_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 101 (NODE.ARR-071)
1 06:08:38; MXN MA103_AMEX LEAVING NODE 101 FOR NODE 102 ON ROUTE 14; ARRIVAL TIME IS 06:09:02 (NODE.DEP-078)
1 06:08:46; MXN MA104_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 106 (NODE.ARR-071)
1 06:08:46; MXN MA104_AMEX LEAVING NODE 106 FOR NODE 31 ON ROUTE 14; ARRIVAL TIME IS 06:09:31 (NODE.DEP-078)
1 06:08:48; MXN MA106_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 136 (NODE.ARR-071)

LOG.MEX

1 06:08:48; MXN MA106_AMEX LEAVING NODE 136 FOR NODE 104 ON ROUTE 14; ARRIVAL TIME IS 06:10:07 (NODE.DEP-078)

1 06:08:53; MXN MA101_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 105 (NODE.ARR-071)

1 06:08:53; MXN MA101_AMEX LEAVING NODE 105 FOR NODE 106 ON ROUTE 14; ARRIVAL TIME IS 06:09:41 (NODE.DEP-078)

1 06:09:02; MXN MA103_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 102 (NODE.ARR-071)

1 06:09:02; MXN MA103_AMEX LEAVING NODE 102 FOR NODE 103 ON ROUTE 14; ARRIVAL TIME IS 06:09:24 (NODE.DEP-078)

1 06:09:24; MXN MA103_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 103 (NODE.ARR-071)

1 06:09:24; MXN MA103_AMEX LEAVING NODE 103 FOR NODE 136 ON ROUTE 14; ARRIVAL TIME IS 06:10:06 (NODE.DEP-078)

1 06:09:24; MXN MA105_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 104 (NODE.ARR-071)

1 06:09:24; MXN MA105_AMEX LEAVING NODE 104 FOR NODE 105 ON ROUTE 14; ARRIVAL TIME IS 06:10:36 (NODE.DEP-078)

1 06:09:31; MXN MA104_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 31 (NODE.ARR-071)

1 06:09:31; MXN MA104_AMEX LEAVING NODE 31 FOR NODE 1 ON ROUTE 14; ARRIVAL TIME IS 06:11:22 (NODE.DEP-078)

1 06:09:41; MXN MA101_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 106 (NODE.ARR-071)

1 06:09:41; MXN MA101_AMEX LEAVING NODE 106 FOR NODE 31 ON ROUTE 14; ARRIVAL TIME IS 06:10:26 (NODE.DEP-078)

1 06:09:42; MXN MA102_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 105 (NODE.ARR-071)

1 06:09:42; MXN MA102_AMEX LEAVING NODE 105 FOR NODE 106 ON ROUTE 14; ARRIVAL TIME IS 06:10:29 (NODE.DEP-078)

1 06:10:06; MXN MA103_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 136 (NODE.ARR-071)

1 06:10:06; MXN MA103_AMEX LEAVING NODE 136 FOR NODE 104 ON ROUTE 14; ARRIVAL TIME IS 06:11:06 (NODE.DEP-078)

1 06:10:07; MXN MA106_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 104 (NODE.ARR-071)

1 06:10:07; MXN MA106_AMEX LEAVING NODE 104 FOR NODE 105 ON ROUTE 14; ARRIVAL TIME IS 06:11:25 (NODE.DEP-078)

1 06:10:26; MXN MA101_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 31 (NODE.ARR-071)

1 06:10:26; MXN MA101_AMEX LEAVING NODE 31 FOR NODE 1 ON ROUTE 14; ARRIVAL TIME IS 06:12:37 (NODE.DEP-078)

1 06:10:29; MXN MA102_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 106 (NODE.ARR-071)

1 06:10:29; MXN MA102_AMEX LEAVING NODE 106 FOR NODE 31 ON ROUTE 14; ARRIVAL TIME IS 06:11:14 (NODE.DEP-078)

1 06:10:36; MXN MA105_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 105 (NODE.ARR-071)

1 06:10:36; MXN MA105_AMEX LEAVING NODE 105 FOR NODE 106 ON ROUTE 14; ARRIVAL TIME IS 06:11:24 (NODE.DEP-078)

1 06:11:06; MXN MA103_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 104 (NODE.ARR-071)

1 06:11:06; MXN MA103_AMEX LEAVING NODE 104 FOR NODE 105 ON ROUTE 14; ARRIVAL TIME IS 06:12:07 (NODE.DEP-078)

1 06:11:14; MXN MA102_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 31 (NODE.ARR-071)

1 06:11:14; MXN MA102_AMEX MAX HOLDING CAPACITY IS 0 AT NODE 31 WHICH IS OF STACK TYPE 0 (CONTROL-087)

1 06:11:14; MXN MA102_AMEX IS HOLDING FOR 5 SECONDS AT NODE 31 (NODE.DEP-086)

1 06:11:17; MXN MA104_AMEX PLANNING ARRIVAL AT MEX ON RWY 05R TO GATE MEX-2 (OPEN) VIA AN 'OPTIMAL' PATH (ARR.TXWY.PL-125)

1 06:11:19; MXN MA102_AMEX IS RELEASED FROM HOLDING AT NODE 31 AFTER A TOTAL HOLD TIME OF 5 SECONDS (NODE.DEP-085)

1 06:11:19; MXN MA102_AMEX LEAVING NODE 31 FOR NODE 1 ON ROUTE 14; ARRIVAL TIME IS 06:13:43 (NODE.DEP-078)

1 06:11:22; MXN MA104_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 1 (NODE.ARR-071)

1 06:11:22; MXN MA104_AMEX HVY IS LANDING AT NODE 1, ON MEX 05R USING PROC 1, AFTER A 0 SEC ENROUTE DELAY (LANDING-041)

1 06:11:22; MXN MA104_AMEX IS ABOUT TO WORK 30 SECONDS LANDING ROLL TIME ON RWY 05R (LANDING-052)

1 06:11:22; MXN MA104_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 1 WHILE LANDING ON RWY 05R (LANDING-053)

1 06:11:24; MXN MA105_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 106 (NODE.ARR-071)

1 06:11:24; MXN MA105_AMEX LEAVING NODE 106 FOR NODE 31 ON ROUTE 14; ARRIVAL TIME IS 06:12:09 (NODE.DEP-078)

1 06:11:25; MXN MA106_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 105 (NODE.ARR-071)

1 06:11:25; MXN MA106_AMEX LEAVING NODE 105 FOR NODE 106 ON ROUTE 14; ARRIVAL TIME IS 06:12:13 (NODE.DEP-078)

1 06:11:25; MXN MA104_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 43 WHILE LANDING ON RWY 05R (LANDING-053)

1 06:11:33; MXN MA104_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 45 WHILE LANDING ON RWY 05R (LANDING-053)

1 06:11:43; MXN MA104_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 47 WHILE LANDING ON RWY 05R (LANDING-053)

1 06:11:52; MXN MA104_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 51 WHILE LANDING ON RWY 05R (LANDING-053)

1 06:11:52; MXN MA104_AMEX HAS FINISHED ITS 30 SEC LANDING ROLL ON RWY 05R (LANDING-042)

1 06:11:52; MXN MA104_AMEX AFTER SAFELY LANDING ON RWY 05R, RELEASES MEX NODE(S): 9 19 29 11 2 (LANDING-054)

1 06:11:52; MXN MA104_AMEX IS ABOUT TO TAXI FOR 24 SECONDS ON LINK 61 FROM NODE 51 TO NODE 27 (PLNE.TXNG-142)

1 06:12:07; MXN MA103_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 105 (NODE.ARR-071)

1 06:12:07; MXN MA103_AMEX MUST HOLD AT NODE 105 BECAUSE OF WAKE TURBULENCE THERE (HOLD.CHECK-081)

1 06:12:08; MXN MA103_AMEX IS RELEASED FROM HOLDING AT NODE 105 AFTER A DELAY OF 1 SECONDS (MOVE.CHK-084)

1 06:12:08; MXN MA103_AMEX IS RELEASED FROM HOLDING AT NODE 105 AFTER A TOTAL HOLD TIME OF 1 SECONDS (NODE.DEP-085)

1 06:12:09; MXN MA105_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 31 (NODE.ARR-071)

1 06:12:09; MXN MA105_AMEX MAX HOLDING CAPACITY IS 0 AT NODE 31 WHICH IS OF STACK TYPE 0 (CONTROL-087)

1 06:12:13; MXN MA106_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 106 (NODE.ARR-071)

1 06:12:13; MXN MA106_AMEX MUST HOLD AT NODE 106 BECAUSE OF DOWNSTREAM HOLDING AT NODE 31 (HOLD.CHECK-080)

1 06:12:13; MXN MA106_AMEX PLACED IN HOLDING AT NODE 106 WHERE THE HOLD COUNT IS NOW 1 AIRCRAFT (HOLD.CHECK-083)

1 06:12:13; MXN MA105_AMEX IS RELEASED FROM HOLDING AT NODE 31 AFTER A TOTAL HOLD TIME OF 4 SECONDS (NODE.DEP-085)

1 06:12:13; MXN MA105_AMEX LEAVING NODE 31 FOR NODE 1 ON ROUTE 14; ARRIVAL TIME IS 06:14:58 (NODE.DEP-078)

1 06:12:13; MXN MA106_AMEX IS RELEASED FROM HOLDING AT NODE 106 AFTER A DELAY OF 0 SECONDS (MOVE.CHK-084)

1 06:12:13; MXN MA106_AMEX IS RELEASED FROM HOLDING AT NODE 106 AFTER A TOTAL HOLD TIME OF 0 SECONDS (NODE.DEP-085)

1 06:12:13; MXN MA106_AMEX LEAVING NODE 106 FOR NODE 31 ON ROUTE 14; ARRIVAL TIME IS 06:12:58 (NODE.DEP-078)

1 06:12:16; MXN MA104_AMEX IS ABOUT TO TAXI FOR 67 SECONDS ON LINK 52 FROM NODE 27 TO NODE 49 (PLNE.TXNG-142)

1 06:12:32; MXN MA101_AMEX PLANNING ARRIVAL AT MEX ON RWY 05R TO GATE MEX-2 (OPEN) VIA AN 'OPTIMAL' PATH (ARR.TXWY.PL-125)

1 06:12:37; MXN MA101_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 1 (NODE.ARR-071)

1 06:12:37; MXN MA101_AMEX HVY IS LANDING AT NODE 1, ON MEX 05R USING PROC 1, AFTER A 42 SEC ENROUTE DELAY (LANDING-041)

1 06:12:37; MXN MA101_AMEX IS ABOUT TO WORK 30 SECONDS LANDING ROLL TIME ON RWY 05R (LANDING-052)

1 06:12:37; MXN MA101_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 1 WHILE LANDING ON RWY 05R (LANDING-053)

1 06:12:40; MXN MA101_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 43 WHILE LANDING ON RWY 05R (LANDING-053)

1 06:12:48; MXN MA101_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 45 WHILE LANDING ON RWY 05R (LANDING-053)

1 06:12:55; MXN MA103_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 106 (NODE.ARR-071)

1 06:12:55; MXN MA103_AMEX LEAVING NODE 106 FOR NODE 31 ON ROUTE 14; ARRIVAL TIME IS 06:13:40 (NODE.DEP-078)

LOG.MEX

1 06:12:57; MXN MA101_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 47 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:12:58; MXN MA106_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 31 (NODE.ARR-071)
1 06:12:58; MXN MA106_AMEX MAX HOLDING CAPACITY IS 0 AT NODE 31 WHICH IS OF STACK TYPE 0 (CONTROL-087)
1 06:12:58; MXN MA106_AMEX IS HOLDING FOR 8 SECONDS AT NODE 31 (NODE.DEP-086)
1 06:13:06; MXN MA106_AMEX IS RELEASED FROM HOLDING AT NODE 31 AFTER A TOTAL HOLD TIME OF 8 SECONDS (NODE.DEP-085)
1 06:13:06; MXN MA106_AMEX LEAVING NODE 31 FOR NODE 1 ON ROUTE 14; ARRIVAL TIME IS 06:16:05 (NODE.DEP-078)
1 06:13:07; MXN MA101_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 51 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:13:07; MXN MA101_AMEX HAS FINISHED ITS 30 SEC LANDING ROLL ON RWY 05R (LANDING-042)
1 06:13:07; MXN MA101_AMEX AFTER SAFELY LANDING ON RWY 05R, RELEASES MEX NODE(S): 9 19 29 11 2 (LANDING-054)
1 06:13:07; MXN MA101_AMEX IS ABOUT TO TAXI FOR 24 SECONDS ON LINK 61 FROM NODE 51 TO NODE 27 (PLNE.TXNG-142)
1 06:13:23; MXN MA104_AMEX IS ABOUT TO TAXI FOR 33 SECONDS ON LINK 56 FROM NODE 49 TO NODE 25 (PLNE.TXNG-142)
1 06:13:31; MXN MA101_AMEX IS ABOUT TO TAXI FOR 67 SECONDS ON LINK 52 FROM NODE 27 TO NODE 49 (PLNE.TXNG-142)
1 06:13:38; MXN MA102_AMEX PLANNING ARRIVAL AT MEX ON RWY 05R TO GATE MEX-2 (OPEN) VIA AN 'OPTIMAL' PATH (ARR.TXWY.PL-125)
1 06:13:40; MXN MA103_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 31 (NODE.ARR-071)
1 06:13:40; MXN MA103_AMEX MUST HOLD AT NODE 31 BECAUSE OF WAKE TURBULENCE THERE (HOLD.CHECK-081)
1 06:13:40; MXN MA103_AMEX PLACED IN HOLDING AT NODE 31 WHERE THE HOLD COUNT IS NOW 1 AIRCRAFT (HOLD.CHECK-083)
1 06:13:43; MXN MA102_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 1 (NODE.ARR-071)
1 06:13:43; MXN MA102_AMEX HVY IS LANDING AT NODE 1, ON MEX 05R USING PROC 1, AFTER A 108 SEC ENROUTE DELAY (LANDING-041)
1 06:13:43; MXN MA102_AMEX IS ABOUT TO WORK 30 SECONDS LANDING ROLL TIME ON RWY 05R (LANDING-052)
1 06:13:43; MXN MA102_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 1 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:13:46; MXN MA102_AMEX IS RELEASED FROM HOLDING AT NODE 31 AFTER A DELAY OF 6 SECONDS (MOVE.CHK-084)
1 06:13:46; MXN MA103_AMEX MAX HOLDING CAPACITY IS 0 AT NODE 31 WHICH IS OF STACK TYPE 0 (CONTROL-087)
1 06:13:46; MXN MA103_AMEX IS HOLDING FOR 14 MORE SECONDS AT NODE 31; HOLD WAS FIRST BEGUN AT 06:13:40 (NODE.DEP-086)
1 06:13:47; MXN MA102_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 43 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:13:55; MXN MA102_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 45 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:13:57; MXN MA104_AMEX REACHES GATE MEX-2 AFTER A DELAY OF 0 SEC IN 125 SEC OF INBOUND TAXI TIME (GATE.ARIV-143)
1 06:13:57; MXN MA104_AMEX WILL OCCUPY MEX GATE MEX-2 FOR 2100 SECONDS WHILE UNLOADING (GATE.ARIV-128)
1 06:14:00; MXN MA103_AMEX IS RELEASED FROM HOLDING AT NODE 31 AFTER A TOTAL HOLD TIME OF 19 SECONDS (NODE.DEP-085)
1 06:14:00; MXN MA103_AMEX LEAVING NODE 31 FOR NODE 1 ON ROUTE 14; ARRIVAL TIME IS 06:17:01 (NODE.DEP-078)
1 06:14:04; MXN MA102_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 47 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:14:13; MXN MA102_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 51 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:14:13; MXN MA102_AMEX HAS FINISHED ITS 30 SEC LANDING ROLL ON RWY 05R (LANDING-042)
1 06:14:13; MXN MA102_AMEX AFTER SAFELY LANDING ON RWY 05R, RELEASES MEX NODE(S): 9 19 29 11 2 (LANDING-054)
1 06:14:13; MXN MA102_AMEX IS ABOUT TO TAXI FOR 24 SECONDS ON LINK 61 FROM NODE 51 TO NODE 27 (PLNE.TXNG-142)
1 06:14:37; MXN MA102_AMEX IS ABOUT TO TAXI FOR 67 SECONDS ON LINK 52 FROM NODE 27 TO NODE 49 (PLNE.TXNG-142)
1 06:14:38; MXN MA101_AMEX IS ABOUT TO TAXI FOR 33 SECONDS ON LINK 56 FROM NODE 49 TO NODE 25 (PLNE.TXNG-142)
1 06:14:53; MXN MA105_AMEX PLANNING ARRIVAL AT MEX ON RWY 05R TO GATE MEX-2 (OPEN) VIA AN 'OPTIMAL' PATH (ARR.TXWY.PL-125)
1 06:14:58; MXN MA105_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 1 (NODE.ARR-071)
1 06:14:58; MXN MA105_AMEX HVY IS LANDING AT NODE 1, ON MEX 05R USING PROC 1, AFTER A 151 SEC ENROUTE DELAY (LANDING-041)
1 06:14:58; MXN MA105_AMEX IS ABOUT TO WORK 30 SECONDS LANDING ROLL TIME ON RWY 05R (LANDING-052)
1 06:14:58; MXN MA105_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 1 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:15:01; MXN MA105_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 43 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:15:09; MXN MA105_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 45 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:15:11; MXN MA101_AMEX REACHES GATE MEX-2 AFTER A DELAY OF 0 SEC IN 125 SEC OF INBOUND TAXI TIME (GATE.ARIV-143)
1 06:15:11; MXN MA101_AMEX WILL OCCUPY MEX GATE MEX-2 FOR 2100 SECONDS WHILE UNLOADING (GATE.ARIV-128)
1 06:15:19; MXN MA105_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 47 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:15:28; MXN MA105_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 51 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:15:28; MXN MA105_AMEX HAS FINISHED ITS 30 SEC LANDING ROLL ON RWY 05R (LANDING-042)
1 06:15:28; MXN MA105_AMEX AFTER SAFELY LANDING ON RWY 05R, RELEASES MEX NODE(S): 9 19 29 11 2 (LANDING-054)
1 06:15:28; MXN MA105_AMEX IS ABOUT TO TAXI FOR 24 SECONDS ON LINK 61 FROM NODE 51 TO NODE 27 (PLNE.TXNG-142)
1 06:15:44; MXN MA102_AMEX IS ABOUT TO TAXI FOR 33 SECONDS ON LINK 56 FROM NODE 49 TO NODE 25 (PLNE.TXNG-142)
1 06:15:52; MXN MA105_AMEX IS ABOUT TO TAXI FOR 67 SECONDS ON LINK 52 FROM NODE 27 TO NODE 49 (PLNE.TXNG-142)
1 06:16:00; MXN MA106_AMEX PLANNING ARRIVAL AT MEX ON RWY 05R TO GATE MEX-2 (OPEN) VIA AN 'OPTIMAL' PATH (ARR.TXWY.PL-125)
1 06:16:05; MXN MA106_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 1 (NODE.ARR-071)
1 06:16:05; MXN MA106_AMEX HVY MISSED AN APPROACH AT MEX DUE TO BLOCKED PROC #1 COUNT = 2 (MSSD.APPR-000)
1 06:16:05; MXN MA106_AMEX HAS NO CONTINUATION PATH AVAILABLE, AND WILL BE FORCED TO LAND ANYWAY. MISSES=1 (MSSD.APPR-000)
1 06:16:05; MXN MA106_AMEX HVY IS LANDING AT NODE 1, ON MEX 05R USING PROC 1, AFTER A 174 SEC ENROUTE DELAY (LANDING-041)
1 06:16:05; MXN MA106_AMEX IS ABOUT TO WORK 30 SECONDS LANDING ROLL TIME ON RWY 05R (LANDING-052)
1 06:16:05; MXN MA106_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 1 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:16:09; MXN MA106_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 43 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:16:17; MXN MA106_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 45 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:16:18; MXN MA102_AMEX REACHES GATE MEX-2 AFTER A DELAY OF 0 SEC IN 125 SEC OF INBOUND TAXI TIME (GATE.ARIV-143)
1 06:16:18; MXN MA102_AMEX WILL OCCUPY MEX GATE MEX-2 FOR 2100 SECONDS WHILE UNLOADING (GATE.ARIV-128)
1 06:16:26; MXN MA106_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 47 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:16:35; MXN MA106_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 51 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:16:35; MXN MA106_AMEX HAS FINISHED ITS 30 SEC LANDING ROLL ON RWY 05R (LANDING-042)
1 06:16:35; MXN MA106_AMEX AFTER SAFELY LANDING ON RWY 05R, RELEASES MEX NODE(S): 9 19 29 11 2 (LANDING-054)
1 06:16:35; MXN MA106_AMEX IS ABOUT TO TAXI FOR 24 SECONDS ON LINK 61 FROM NODE 51 TO NODE 27 (PLNE.TXNG-142)
1 06:16:56; MXN MA103_AMEX PLANNING ARRIVAL AT MEX ON RWY 05R TO GATE MEX-2 (OPEN) VIA AN 'OPTIMAL' PATH (ARR.TXWY.PL-125)
1 06:16:59; MXN MA106_AMEX IS ABOUT TO TAXI FOR 67 SECONDS ON LINK 52 FROM NODE 27 TO NODE 49 (PLNE.TXNG-142)
1 06:16:59; MXN MA105_AMEX IS ABOUT TO TAXI FOR 33 SECONDS ON LINK 56 FROM NODE 49 TO NODE 25 (PLNE.TXNG-142)
1 06:17:01; MXN MA103_AMEX ON ROUTE 14 HAS ARRIVED AT NODE 1 (NODE.ARR-071)
1 06:17:01; MXN MA103_AMEX HVY IS LANDING AT NODE 1, ON MEX 05R USING PROC 1, AFTER A 99 SEC ENROUTE DELAY (LANDING-041)
1 06:17:01; MXN MA103_AMEX IS ABOUT TO WORK 30 SECONDS LANDING ROLL TIME ON RWY 05R (LANDING-052)
1 06:17:01; MXN MA103_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 1 WHILE LANDING ON RWY 05R (LANDING-053)

LOG.MEX

```

1 06:17:05; MXN MA103_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 43 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:17:13; MXN MA103_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 45 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:17:22; MXN MA103_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 47 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:17:32; MXN MA103_AMEX HAS PASSED THROUGH MEX AIRFIELD NODE 51 WHILE LANDING ON RWY 05R (LANDING-053)
1 06:17:32; MXN MA103_AMEX HAS FINISHED ITS 30 SEC LANDING ROLL ON RWY 05R (LANDING-042)
1 06:17:32; MXN MA103_AMEX AFTER SAFELY LANDING ON RWY 05R, RELEASES MEX NODE(S): 9 19 29 11 2 (LANDING-054)
1 06:17:32; MXN MA103_AMEX IS ABOUT TO TAXI FOR 24 SECONDS ON LINK 61 FROM NODE 51 TO NODE 27 (PLNE.TXNG-142)
1 06:17:33; MXN MA105_AMEX REACHES GATE MEX-2 AFTER A DELAY OF 0 SEC IN 125 SEC OF INBOUND TAXI TIME (GATE.ARIV-143)
1 06:17:33; MXN MA105_AMEX WILL OCCUPY MEX GATE MEX-2 FOR 2100 SECONDS WHILE UNLOADING (GATE.ARIV-128)
1 06:17:56; MXN MA103_AMEX IS ABOUT TO TAXI FOR 33 SECONDS ON LINK 52 FROM NODE 27 TO NODE 49 (PLNE.TXNG-142)
1 06:18:06; MXN MA106_AMEX IS ABOUT TO TAXI FOR 33 SECONDS ON LINK 56 FROM NODE 49 TO NODE 25 (PLNE.TXNG-142)
1 06:18:40; MXN MA106_AMEX REACHES GATE MEX-2 AFTER A DELAY OF 0 SEC IN 125 SEC OF INBOUND TAXI TIME (GATE.ARIV-143)
1 06:18:40; MXN MA106_AMEX WILL OCCUPY MEX GATE MEX-2 FOR 2100 SECONDS WHILE UNLOADING (GATE.ARIV-128)
1 06:19:03; MXN MA103_AMEX IS ABOUT TO TAXI FOR 33 SECONDS ON LINK 56 FROM NODE 49 TO NODE 25 (PLNE.TXNG-142)
1 06:19:36; MXN MA103_AMEX REACHES GATE MEX-2 AFTER A DELAY OF 0 SEC IN 125 SEC OF INBOUND TAXI TIME (GATE.ARIV-143)
1 06:19:36; MXN MA103_AMEX WILL OCCUPY MEX GATE MEX-2 FOR 2100 SECONDS WHILE UNLOADING (GATE.ARIV-128)
1 06:30:00; TRACE VALUES HAVE NOT BEEN RESET AT REAL TIME 01:50:02 (FROM INDEX 0) FOR ITERATION 1 (TRACE-000)
1 06:48:57; MXN MA104_AMEX IS LEAVING MEX GATE MEX-2 AFTER 2100 SECONDS OF UNLOADING TIME (GATE.ARIV-129)
1 06:48:57; MXN MA104_AMEX OF ATC GROUP HVY HAS REACHED THE END OF ITS FLIGHT PATH AT NODE 1 (EJECT.AC-031)
1 06:48:57; MXN MA104_AMEX REMOVED FROM THE SIMULATION; TRUNRND=0 (EJECT.AC-032)
1 06:50:11; MXN MA101_AMEX IS LEAVING MEX GATE MEX-2 AFTER 2100 SECONDS OF UNLOADING TIME (GATE.ARIV-129)
1 06:50:11; MXN MA101_AMEX OF ATC GROUP HVY HAS REACHED THE END OF ITS FLIGHT PATH AT NODE 1 (EJECT.AC-031)
1 06:50:11; MXN MA101_AMEX REMOVED FROM THE SIMULATION; TRUNRND=0 (EJECT.AC-032)
1 06:51:18; MXN MA102_AMEX IS LEAVING MEX GATE MEX-2 AFTER 2100 SECONDS OF UNLOADING TIME (GATE.ARIV-129)
1 06:51:18; MXN MA102_AMEX OF ATC GROUP HVY HAS REACHED THE END OF ITS FLIGHT PATH AT NODE 1 (EJECT.AC-031)
1 06:51:18; MXN MA102_AMEX REMOVED FROM THE SIMULATION; TRUNRND=0 (EJECT.AC-032)
1 06:52:33; MXN MA105_AMEX IS LEAVING MEX GATE MEX-2 AFTER 2100 SECONDS OF UNLOADING TIME (GATE.ARIV-129)
1 06:52:33; MXN MA105_AMEX OF ATC GROUP HVY HAS REACHED THE END OF ITS FLIGHT PATH AT NODE 1 (EJECT.AC-031)
1 06:52:33; MXN MA105_AMEX REMOVED FROM THE SIMULATION; TRUNRND=0 (EJECT.AC-032)
1 06:53:40; MXN MA106_AMEX IS LEAVING MEX GATE MEX-2 AFTER 2100 SECONDS OF UNLOADING TIME (GATE.ARIV-129)
1 06:53:40; MXN MA106_AMEX OF ATC GROUP HVY HAS REACHED THE END OF ITS FLIGHT PATH AT NODE 1 (EJECT.AC-031)
1 06:53:40; MXN MA106_AMEX REMOVED FROM THE SIMULATION; TRUNRND=0 (EJECT.AC-032)
1 06:54:36; MXN MA103_AMEX IS LEAVING MEX GATE MEX-2 AFTER 2100 SECONDS OF UNLOADING TIME (GATE.ARIV-129)
1 06:54:36; MXN MA103_AMEX OF ATC GROUP HVY HAS REACHED THE END OF ITS FLIGHT PATH AT NODE 1 (EJECT.AC-031)
1 06:54:36; MXN MA103_AMEX REMOVED FROM THE SIMULATION; TRUNRND=0 (EJECT.AC-032)

```

[Omitted events from 07:00:00 through end of simulation]

END SIMULATION LOG AT 01:52:15 FOR ITERATION #1 OF:
08/10/1996 AT 01:49:43 (0) Mexico City International

ITERATION 1 FINAL RANDOM SEED VALUES FOR EACH OF 10 STREAMS ARE:
1-1800844157 2-586956138 3-1179282513 4-469235726 5-214807712
6-1157240309 7-15726055 8-48108509 9-1797920909 10-477424540

ITERATION 1 RUNWAY CROSSING DELAY STATISTICS (IN SECONDS) FOR 5 RUNWAYS AT 2 AIRPORTS:

RWY#	APT	RUNWAY NAME	DELAYED XNGS	MEAN DELAY	MAX DELAY	TOTAL DELAY
1	MEX	05R/23L	0	0.	0.	0.
2	MEX	05L/23R	0	0.	0.	0.
3	MEX	31X/13X	0	0.	0.	0.
ALL	MEX	TOTALS	0	0.	0.	0.
4	ACA	28X/10X	4	14.9	30.9	59.6
5	ACA	24X/06X	0	0.	0.	0.
ALL	ACA	TOTALS	4	14.9	30.9	59.6

Appendix C - AQM C++ Header Files

These files contain declarations of member variables and functions used in the AQM program, and comprise only part of the source code created for AQM. The reader is directed to contact the author for inquiries related to source code beyond what is contained in this report.

Contents

SIMMOD element classes

SimmodNode.h, SimmodLink.h, SimmodAirport.h, SimmodAFNode.h, SimmodAFLink.h, SimmodGate.h, SimmodAirline.h, and SimmodEvent.h.

Defines the respective SIMMOD elements described in this report.

AQM element classes

AQMNode.h, AQMLink.h, AQMFlight.h, and AQMEvent.h.

Defines the respective AQM elements described in this report.

Functional classes

AQM.h - Main header file for the AQM application

AQMDoc.h - AQM Document class; contains all project-specific information for AQM documents including objects created from SIMMOD data, derived AQM objects, and project-dependent parameters.

AQMTree.h - Tree control view

AQMView.h - Main AQM view showing the number of elements contained in the current AQM document.

AQMListView.h - Boilerplate class for display of element information in table format.

AQMNetworkView.h - Defines three-dimensional view of air traffic network.

SimmodView.h - Displays information specific to SIMMOD source project file (currently not implemented).

ChildFrm.h - Frame class for document windows.

MainFrm.h - Main frame class for application window.

SettingsPages.h - Dialog class; defines property pages for AQM settings

SelectSimmodProject.h - Dialog class; provides directory pick list for SIMMOD project.

SetProjectOrigin.h - Dialog class; allows user entry of project origin for conversion of node coordinates (implemented for future enhancements allowing background reference of raster images or real-world coordinate mapping).

Settings.h - Dialog class; sets AQM global settings on local machine.

ProgressStatusBar.h - Wrapper class for progress bar displayed during lengthy operations.

Splash.h - Displays opening graphic

Splitter.h - Helper class for *CChildFrame*, provides MFC "splitter bar" between tree view and alternate view frame.

```
// SimmodNode.h : interface of the CSimmodNode class
//
///////////////////////////////////////////////////////////////////

class CSimmodNode : public CObject
{
public:
    DECLARE_SERIAL( CSimmodNode )
    // empty constructor is necessary
    CSimmodNode(){};

    // Node ID for AQM files
    UINT ID;

    // Loaded from SIMMOD files
    UINT num; // Node number as indicated in
SIMMOD input files
    CString name; // Node name as indicated in
SIMMOD input files
    UINT altitude; // Node altitude (ft)
// East (X) coordinates (nmiles)
// North (Y) coordinates (nmiles)

    void ReadSimmodNodeInformation( UINT ID, CString inputline );
    CString GetFormattedOutputString();
};
```

```
// SimmodLink.h : interface of the CSimmodLink class
//
/////////////////////////////////////////////////////////////////

class CSimmodLink : public CObject
{
public:
    DECLARE_SERIAL( CSimmodLink )
    // empty constructor is necessary
    CSimmodLink(){};

    // Link ID for AQM files
    UINT ID;

    // Loaded from SIMMOD files
    UINT num; // Link number as indicated in
SIMMOD input files
    CString name; // Link name as indicated
in SIMMOD input files
    UINT node1; // Link origin node number
    UINT node2; // Link terminus node number

    void Serialize( CArchive& archive );
    void ReadSimmodLinkInformation( UINT ID, CString inputline );
    CString GetFormattedOutputString();
};
```

```
// SimmodAirport.h : interface of the CSimmodAirport class
//
///////////////////////////////////////////////////////////////////

class CSimmodAirport : public CObject
{
public:
    DECLARE_SERIAL( CSimmodAirport )
    // empty constructor is necessary
    CSimmodAirport(){};

    // Node ID for AQM files
    UINT ID;

    // Loaded from SIMMOD files
    UINT num; // Airport number as indicated in
SIMMOD input files
    CString name; // Airport name as indicated in SIMMOD
input files
    CString code; // Airport three-letter code designator
    UINT elevation; // Airport field elevation

    void Serialize( CArchive& archive );
    void ReadSimmodAirportInformation( UINT ID, CString inputline );
    CString GetFormattedOutputString();
};
```

```
// SimmodAFNode.h : interface of the CAQMNode class
//
///////////////////////////////////////////////////////////////////

class CSimmodAFNode : public CObject
{
public:
    DECLARE_SERIAL( CSimmodAFNode )
    // empty constructor is necessary
    CSimmodAFNode(){};

    // Node ID for AQM files
    UINT ID;

    // Loaded from SIMMOD files
    UINT num; // Node number as indicated in
SIMMOD input files
    int east; // East (X) coordinates (feet)
    int north; // North (Y) coordinates (feet)
    CString airport; // Three-letter airport identifier

    void Serialize( CArchive& archive );
    void ReadSimmodAFNodeInformation( UINT ID, CString InputLine );
    CString GetFormattedOutputString();
};
```

```
// SimmodAFLink.h : interface of the CSimmodAFLink class
//
///////////////////////////////////////////////////////////////////

class CSimmodAFLink : public CObject
{
public:
    DECLARE_SERIAL( CSimmodAFLink )
    // empty constructor is necessary
    CSimmodAFLink(){};

    //    Link ID for AQM files
    UINT ID;

    //    Loaded from SIMMOD files
    UINT num; //    Link number as indicated in
SIMMOD input files
    UINT node1; //    Link origin node number
    UINT node2; //    Link terminus node number

    void Serialize( CArchive& archive );
    void ReadSimmodAFLinkInformation( UINT ID, CString inputline );
    CString GetFormattedOutputString();
};
```

```
// SimmodGate.h : interface of the CSimmodGate class
//
///////////////////////////////////////////////////////////////////

class CSimmodGate : public CObject
{
public:
    DECLARE_SERIAL( CSimmodGate )
    // empty constructor is necessary
    CSimmodGate(){};

    // Gate ID for AQM files
    UINT ID;

    // Loaded from SIMMOD files
    UINT num; // Gate number as indicated in
SIMMOD input files
    UINT node; // Simmod AFnode associated with
this gate
    CString name; // Gate name
    CString airport; // Airport identifier

    void Serialize( CArchive& archive );
    void ReadSimmodGateInformation( UINT ID, CString inputline );
    CString GetFormattedOutputString();
};
```

```
// SimmodAirline.h : interface of the CSimmodAirline class
//
///////////////////////////////////////////////////////////////////

class CSimmodAirline : public CObject
{
public:
    DECLARE_SERIAL( CSimmodAirline )
    // empty constructor is necessary
    CSimmodAirline(){};

    // Node ID for AQM files
    UINT ID;

    // Loaded from SIMMOD files
    UINT num; // Node number as indicated in
SIMMOD input files
    CString name; // Airline name as indicated in
SIMMOD input files
    CString type; // Airline type as indicated in
SIMMOD input files

    void Serialize( CArchive& archive );
    void ReadSimmodAirlineInformation( UINT ID, CString inputline );
    CString GetFormattedOutputString();
};
```

```

// SimmodEvent.h : interface of the CSimmodEvent class
//
//
//
class CSimmodEvent : public CObject
{
public:
    DECLARE_SERIAL( CSimmodEvent )
    // empty constructor is necessary
    CSimmodEvent(){};

    // Event ID for AQM files
    UINT ID;

    // Loaded from SIMMOD files
    CTime time; // Event begin time
    CString code; // Event code
    CString airline; // Active airline for event
    CString flight; // Active flight number for event
    CString description; // Event description

    void Serialize( CArchive& archive );
    void ReadSimmodEventInformation( UINT ID, CString EventCode, CString
InputLine );
    CTime GetEventTime( CString InputLine );
    CString GetFormattedOutputString();
};

```

AQM C++ Header Files

```
// AQMNode.h : interface of the CAQMNode class
//
///////////////////////////////////////////////////////////////////

class CAQMNode : public CObject
{
public:
    DECLARE_SERIAL( CAQMNode )
    // empty constructor is necessary
    CAQMNode(){};

    // Node ID for AQM files
    UINT ID;

    // Loaded from SIMMOD files
    CString name; // Node name as indicated in
Simmod input files
    double north; // North (Y) coordinates (nmiles)
    double east; // East (X) coordinates (nmiles)
    UINT altitude; // Node altitude (ft)
    BOOL type; // TRUE = part of airfield node
list
    UINT refID; // Corresponding Simmod Node ID

    void Serialize( CArchive& archive );
    CString GetFormattedOutputString();
};
```

```
// AQMLink.h : interface of the CAQMLink class
//
///////////////////////////////////////////////////////////////////

class CAQMLink : public CObject
{
public:
    DECLARE_SERIAL( CAQMLink )
    // empty constructor is necessary
    CAQMLink(){};

    // Link ID for AQM files
    UINT ID;

    // Loaded from SIMMOD files
    CString name; // Link name as indicated in
Simmod input files
    UINT node1; // Link origin node number
    UINT node2; // Link terminus node number
    BOOL type; // Airfield/airspace type of
corresponding Simmod link
    UINT refID; // ID number of corresponding
CSimmodLink

    void Serialize( CArchive& archive );
    CString GetFormattedOutputString();
};
```

```

// AQMFlight.h
//
//
// Forward declarations
class CSimmodEvent;
class CAQMEvent;

// CAQMFlight class declaration
class CAQMFlight : public CObject
{
public:
    DECLARE_SERIAL( CAQMFlight )
    // empty constructor is necessary
    CAQMFlight(){};
    CAQMFlight( UINT i, CSimmodEvent* event );

    UINT ID; // Flight ID number
    CString name; // Flight name (assigned in
SIMMOD project files)
    CString aircraft; // Aircraft type used by this flight
    CString airline; // Airline in charge of flight
    CTime time1; // Time flight begins
    CTime time2; // Time flight ends

    UINT m_NumEvents; // Number of events
    CObArray FlightEvents; // Pointers to CAQMEvents that make up
the flight

public:
    void Serialize( CArchive& archive );
    UINT AddFlightEvent( CAQMEvent* pEvent );
    CString GetFormattedOutputString();
    CString GetEventList();
};

```

```
// AQMEvent.h : interface of the CAQMEvent class
//
///////////////////////////////////////////////////////////////////

class CAQMEvent : public CObject
{
public:
    DECLARE_SERIAL( CAQMEvent )
    // empty constructor is necessary
    CAQMEvent(){};

    UINT ID;                //      Event ID number

    UINT mode;
    //      can be any of the following:
    //      idle, taxi_departure, takeoff, climbout, approach, taxi_arrival

    CTime time1, time2;
    UINT node1, node2;
    UINT flight;

    void Serialize( CArchive& archive );
    CString GetFormattedOutputString();
};
```

AQM C++ Header Files

```
// AQM.h : main header file for the AQM application
//

#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"          // main symbols

////////////////////////////////////
// Forward References
class CMainFrame;
int CALLBACK SetSelProc( HWND hWnd, UINT uMsg, LPARAM lParam, LPARAM lpData );
BOOL DirectoryPicker( CWnd *pWnd, CString& title, CString& dir, CString&
initial );

////////////////////////////////////
// CAQMApp:
// See AQM.cpp for the implementation of this class
//

class CAQMApp : public CWinApp
{
//    AQM Globals
public:
    virtual BOOL PreTranslateMessage(MSG* pMsg);
    CString m_SimmodRootDirectory;        //    Root directory for all SIMMOD
Projects

    CString m_AQMAircraftDirectory;      //    Full path name of AQM Aircraft
database
    UINT m_NumAQMAircraft;               //    Number of aircraft in database
    CObArray AQMAircraft;                //    Array containing pointers to
AQMAircraft objects

public:
    CAQMApp();

// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CAQMApp)
public:
    virtual BOOL InitInstance();
    virtual int ExitInstance();
//}}AFX_VIRTUAL
public:
    static CAQMApp * GetApp() {return (CAQMApp *)AfxGetApp(); }
    CMainFrame * GetMainFrame() {return (CMainFrame*)
CWinThread::m_pMainWnd; }

// Implementation

//{{AFX_MSG(CAQMApp)
afx_msg void OnAppAbout();
//}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

AQM C++ Header Files

```
// AQMDoc.h : interface of the CAQMDoc class
//
//
// Forward declarations
class CSimmodNode;
class CSimmodAFNode;
class CSimmodLink;
class CSimmodAFLink;
class CSimmodGate;
class CSimmodAirport;
class CSimmodAirline;
class CSimmodEvent;
class CAQMNode;
class CAQMLink;
class CAQMFlight;
class CAQMEvent;
class CProgressStatusBar;

#define FEET_PER_NM (double) 6076.11 // Number of feet in a nautical
mile

class CAQMDoc : public CDocument
{
protected: // create from serialization only
    CAQMDoc();
    DECLARE_DYNCREATE(CAQMDoc)

public:
// Attributes
/*****
/**/ SIMMOD Source Data Variables
/*****/
/**/public:
/**/ SIMMOD project source file information
/**/ BOOL m_HasOpenedSimmodProject; // AQM project contains Simmod data
/**/ CString m_SimmodProjectFullPath; // File path location
/**/ CString m_SimmodProjectExtension; // Three-letter file extension /
project subdirectory under SimmodRoot
/**/ CString m_SimmodProjectName; // SIMMOD Project name
/**/
/**/ Object Array element counts
/**/ UINT m_NumSimmodNodes; // Number of Simmod NODEs
/**/ UINT m_NumSimmodAFNodes; // Number of Simmod AFNODEs
/**/ UINT m_NumSimmodLinks; // Number of Simmod LINKs
/**/ UINT m_NumSimmodAFLinks; // Number of Simmod AFLINKs
/**/ UINT m_NumSimmodGates; // Number of Simmod GATES
/**/ UINT m_NumSimmodAirports; // Number of Simmod AIRPORTSs
/**/ UINT m_NumSimmodAirlines; // Number of Simmod AIRLINES
/**/ UINT m_NumSimmodEvents; // Number of Simmod Events
/**/
/**/ Object Arrays containing project information
/**/ CObArray SimmodNodes;
/**/ CObArray SimmodAFNodes;
/**/ CObArray SimmodLinks;
/**/ CObArray SimmodAFLinks;
/**/ CObArray SimmodGates;
/**/ CObArray SimmodAirports;
```

```

/**/ CObArray SimmodAirlines;
/**/ CObArray SimmodEvents;
/**/
/**///      SIMMOD project functions
/**/protected:
/**/  BOOL OpenSimmodProject( void );
/**/  BOOL ReadSimmodProject( void );
/**/  void ClearSimmodProject( void );
/**/  BOOL GetSimmodNodes( void );
/**/  BOOL GetSimmodLinks( void );
/**/  BOOL GetSimmodAFNodes( void );
/**/  BOOL GetSimmodAFLinks( void );
/**/  BOOL GetSimmodGates( void );
/**/  BOOL GetSimmodAirports( void );
/**/  BOOL GetSimmodAirlines( void );
/**/  BOOL GetSimmodEvents( void );

/*****
/**///      AQM Project Variables
/*****
/**/public:
/**///      General project information
/**/  double m_ProjectOriginNorth;      // Northing of project origin
/**/  double m_ProjectOriginEast;      // Easting of project origin
/**/
/**///      AQM object information
/**/  BOOL  m_HasGeneratedAQMEvents;    // AQM objects created
/**/  UINT  m_NumAQMNodes;              // Number of AQM Nodes
/**/  UINT  m_NumAQMLinks;              // Number of AQM Links
/**/  UINT  m_NumAQMEvents;             // Number of AQM Events
/**/  UINT  m_NumAQMFlights;           // Number of AQM Flights
/**/
/**/  CObArray AQMNodes;
/**/  CObArray AQMLinks;
/**/  CObArray AQMFlights;
/**/  CObArray AQMEvents;
/**/
/**/  CRect GetAQMNNodeExtents();
/**/  GLdouble GetRemoteNodeDist();
/**/
/**///      AQM project functions
/**/protected:
/**/  BOOL Parse_IsSimmodInputComment( LPTSTR pToken );
/**/  BOOL Parse_ReadFormattedLine( CString inputline, UINT formatID );
/**/  LONG Parse_GetIntegerToken( char* token );
/**/  void BuildAQMEvents();
/**/  void GenerateAQMNodes();
/**/  void GenerateAQMLinks();
/**/  void GenerateAQMFlights();
/**/  void ReadAQMEvents();
/**/  UINT GetAirfieldElevation( CSimmodAFNode* pSimmodAFNode );
/**/  CSimmodEvent* FindCreationEvent( CAQMFlight* pAQMFlight );
/**/  CSimmodEvent* FindNextFlightEvent( UINT SimmodEventID, CAQMFlight*
pAQMFlight );
/**/  CAQMEvent* GetOldEvent( CAQMEvent* pAQMEvent );
/**/  void ClearAQMProject();
/**/  void ParseSimmodEvent( CSimmodEvent* event );

```

```

/*****
/**///      AQM Simulation Variables
/*****
/**/public:
/**///      General
/**/
/**///      AQM simulation information
/**/  BOOL  m_HasGeneratedSimulations;      //      AQM has generated simulations

/*****
/**///      AQM Document Functions
/*****
/**/public:
/**///      Search functions
/**/  BOOL  IsFlight( CString flightname );
/**/  CAQMFlight* GetFlightPtr( CString flightname );
/**/  UINT  GetAQMNodeIDforSimmodNode( UINT SimmodNumber );
/**/  UINT  GetAQMNodeIDforSimmodAFNode( UINT SimmodAFNumber );
/**/  UINT  GetGateID( CString gatename );
/**/  CString GetFormattedString( CRuntimeClass* pObjType, CObject* pObj );
/**/
/**///      Conversion functions
/**/  CTimeSpan GetTimeSpanFromSeconds( UINT num );

/*****
/**///      AQM Document Settings
/*****
/**/public:
/**///      SIMMOD project source file information
/**/  UINT  m_SelectedView;
/**/  UINT  m_SelectedData;

// Operations
public:
    UINT GetSelectedView() { return( m_SelectedView ); }
    void SetSelectedView( UINT viewID );
    CProgressStatusBar* Initialize( UINT string, UINT start, UINT end );

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAQMDoc)
    public:
    virtual BOOL OnNewDocument();
    virtual void Serialize(CArchive& ar);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CAQMDoc();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

    BOOL KillEverything();

// Generated message map functions
protected:

```

```
    //{{AFX_MSG(CAQMDoc)
    afx_msg void OnProjectSimmodproject();
    afx_msg void OnProjectClearProject();
    afx_msg void OnProjectGenerateEvents();
    afx_msg void OnFileWriteAsciiTextFile();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```

////////////////////////////////////
// CAQMTree view

class CAQMTree : public CTreeView
{
protected:
    CAQMTree();          // protected constructor used by dynamic creation
    DECLARE_DYNCREATE(CAQMTree)

// Attributes
public:
    CAQMDoc* GetDocument();

// Operations
public:
    HTREEITEM AddTreeItem(
        LPCTSTR lpszItem,
        int nImage,
        int nSelectedImage,
        HTREEITEM hParent,
        HTREEITEM hInsertAfter,
        UINT treeID);

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAQMTree)
    public:
        virtual void OnInitialUpdate();
    protected:
        virtual void OnDraw(CDC* pDC);          // overridden to draw this view
        virtual void OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint);
    //}}AFX_VIRTUAL

// Implementation
protected:
    virtual ~CAQMTree();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

    // Generated message map functions
protected:
    //{{AFX_MSG(CAQMTree)
    afx_msg void OnSelectItem(NMHDR* pNMHDR, LRESULT* pResult);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#ifdef _DEBUG    // debug version in AQMTree.cpp
inline CAQMDoc* CAQMTree::GetDocument()
    { return (CAQMDoc*) m_pDocument; }
#endif
////////////////////////////////////

```

```

// AQMView.h : interface of the CAQMView class
//
///////////////////////////////////////////////////////////////////

class CAQMView : public CView
{
protected: // create from serialization only
    CAQMView();
    DECLARE_DYNCREATE(CAQMView)

// Attributes
public:
    CAQMDoc* GetDocument();

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAQMView)
public:
    virtual void OnDraw(CDC* pDC); // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
protected:
    virtual BOOL OnPreparePrinting(CPrintInfo* pInfo);
    virtual void OnBeginPrinting(CDC* pDC, CPrintInfo* pInfo);
    virtual void OnEndPrinting(CDC* pDC, CPrintInfo* pInfo);
    virtual void OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CAQMView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected:

// Generated message map functions
protected:
    //{{AFX_MSG(CAQMView)
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#ifdef _DEBUG // debug version in AQMView.cpp
inline CAQMDoc* CAQMView::GetDocument()
    { return (CAQMDoc*) m_pDocument; }
#endif
/////////////////////////////////////////////////////////////////

```

```

// AQMListView.h : header file
//

/////////////////////////////////////////////////////////////////
// CAQMListView view

class CAQMListView : public CListView
{
protected:
    CAQMListView();        // protected constructor used by dynamic creation
    DECLARE_DYNCREATE(CAQMListView)

// Attributes
public:
    CObArray* m_pObArray;
    UINT m_ListViewColumns;
    UINT m_HeadingsID;
    CStringList m_Headings;
    UINT m_ColumnsID;
    CUIIntArray m_Columns;

// Operations
public:
    void SetListViewOptions(
        CObArray* pObArray,
        UINT headingsID,
        UINT columnsID
    );

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAQMListView)
public:
    virtual void OnInitialUpdate();
protected:
    virtual void OnDraw(CDC* pDC);        // overridden to draw this view
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    virtual void OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint);
    //}}AFX_VIRTUAL

// Implementation
protected:
    virtual ~CAQMListView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

    // Generated message map functions
protected:
    //{{AFX_MSG(CAQMListView)
    // NOTE - the ClassWizard will add and remove member functions here.
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
/////////////////////////////////////////////////////////////////

```

AQM C++ Header Files

```
// AQMNetworkView.h : header file
//

/////////////////////////////////////////////////////////////////
// CAQMNetworkView view

class CAQMNetworkView : public CView
{
protected:
    CPoint m_LDownPos;
    CPoint m_RDownPos;
    BOOL m_LButtonDown;
    BOOL m_RButtonDown;
    HGLRC m_hGLContext;
    HPALETTE m_hPalette;
    BOOL CreateViewGLContext(HDC hDC);
    HPALETTE GetOpenGLPalette(HDC hDC);
    int m_GLPixelIndex;
    BOOL SetWindowPixelFormat(HDC hDC);
    CAQMNetworkView();
    DECLARE_DYNCREATE(CAQMNetworkView);
// Attributes
public:
    CAQMDoc* GetDocument();
    CRect m_Viewport;
    BOOL m_ViewportIsZoomed; // Indicates whether or not zoom
operations have been performed on the view
    void RenderScene(void);

    GLdouble m_zNormal; // z-translation required to
display entire network in top view
    GLdouble m_xRotated;
    GLdouble m_zRotated;
    GLdouble m_xTranslated;
    GLdouble m_yTranslated;
    GLdouble m_zTranslated;

// Operations
public:

    void SetViewport( CDC* pDC );
    void DrawElements( CAQMDoc* pDoc );

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAQMNetworkView)
protected:
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    virtual void OnDraw(CDC* pDC);
    //}}AFX_VIRTUAL

// Implementation
protected:
    virtual ~CAQMNetworkView();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif
};
```

```
#endif

    // Generated message map functions
protected:
    //{{AFX_MSG(CAQMNetworkView)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnSize(UINT nType, int cx, int cy);
    afx_msg void OnPaint();
    afx_msg void OnDestroy();
    afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
    afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
    afx_msg void OnMouseMove(UINT nFlags, CPoint point);
    afx_msg void OnRButtonDown(UINT nFlags, CPoint point);
    afx_msg void OnRButtonUp(UINT nFlags, CPoint point);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#ifdef _DEBUG // debug version in GLSample2View.cpp
inline CAQMDoc* CAQMNetworkView::GetDocument()
    { return (CAQMDoc*)m_pDocument; }
#endif

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```

// ChildFrm.h : interface of the CChildFrame class
//
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
class CAQMDoc;

class CChildFrame : public CMDIChildWnd
{
    DECLARE_DYNCREATE(CChildFrame)

// Attributes
public:
    CChildFrame();
    CSplitterWnd m_wndSplitter;
    CSplitterWnd* GetSplitter() { return &m_wndSplitter;}
    CView* GetTreePane();
    CView* GetViewPane();
    CSize GetViewPaneSize();
    CAQMDoc* GetDoc();

// Operations
public:
void SwitchToView(
    CRuntimeClass* pNewView,
    CObArray* pObArray,
    UINT headingsID,
    UINT columnsID
    );

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CChildFrame)
    public:
        virtual BOOL OnCreateClient(LPCREATESTRUCT lpCS, CCreateContext*
pContext);
        virtual BOOL PreCreateWindow(CREATESTRUCT& CS);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CChildFrame();
#ifdef _DEBUG
        virtual void AssertValid() const;
        virtual void Dump(CDumpContext& DC) const;
#endif
// Generated message map functions
protected:
    //{{AFX_MSG(CChildFrame)
    afx_msg void OnUpdateProjectClearProject(CCmdUI* pCmdUI);
    afx_msg void OnUpdateProjectGenerateEvents(CCmdUI* pCmdUI);
    afx_msg void OnUpdateProjectViewFlightPaths(CCmdUI* pCmdUI);
    afx_msg void OnUpdateProjectNewSimulation(CCmdUI* pCmdUI);
    afx_msg void OnUpdateFileWriteAsciiTextFile(CCmdUI* pCmdUI);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

```

```

// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////

class CMainFrame : public CMDIFrameWnd
{
    DECLARE_DYNAMIC(CMainFrame)
public:
    CMainFrame();

// Attributes
public:
    CProgressStatusBar *GetStatusBar() { return &m_wndStatusBar; }

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CMainFrame)
    virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CMainFrame();
#ifdef _DEBUG
    virtual void AssertValid() const;
    virtual void Dump(CDumpContext& dc) const;
#endif

protected: // control bar embedded members
    CProgressStatusBar m_wndStatusBar;
    CToolBar m_wndToolBar;

// Generated message map functions
protected:
    //{{AFX_MSG(CMainFrame)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnOptionsSettings();
    afx_msg void OnOptionsAircraft();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////

```

```

// CSettingsPages.h : header file
//

#ifndef __SETTINGSPAGES_H__
#define __SETTINGSPAGES_H__

////////////////////////////////////
// CSettingsSimmod dialog

class CSettingsSimmod : public CPropertyPage
{
    DECLARE_DYNCREATE(CSettingsSimmod)

// Construction
public:
    CSettingsSimmod();
    ~CSettingsSimmod();

// Dialog Data
   //{{AFX_DATA(CSettingsSimmod)
    enum { IDD = IDD_SETTINGS_SIMMOD };
    CString        m_SimmodRootDir;
    //}}AFX_DATA

// Overrides
    // ClassWizard generate virtual function overrides
    //{{AFX_VIRTUAL(CSettingsSimmod)
protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
    //{{AFX_MSG(CSettingsSimmod)
    afx_msg void OnChangeSimmodRootdir();
    afx_msg void OnBrowse();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()

};

////////////////////////////////////
// CSettingsSimulation dialog

class CSettingsSimulation : public CPropertyPage
{
    DECLARE_DYNCREATE(CSettingsSimulation)

// Construction
public:
    CSettingsSimulation();
    ~CSettingsSimulation();

// Dialog Data
   //{{AFX_DATA(CSettingsSimulation)

```

```
enum { IDD = IDD_SETTINGS_SIMULATION };
    // NOTE - ClassWizard will add data members here.
// DO NOT EDIT what you see in these blocks of generated code !
//}}AFX_DATA

// Overrides
// ClassWizard generate virtual function overrides
//{{AFX_VIRTUAL(CSettingsSimulation)
protected:
virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
//}}AFX_VIRTUAL

// Implementation
protected:
// Generated message map functions
//{{AFX_MSG(CSettingsSimulation)
// NOTE: the ClassWizard will add member functions here
//}}AFX_MSG
DECLARE_MESSAGE_MAP()

};

#endif // __SETTINGSPAGES_H__
```

```
// SelectSimmodProject.h : header file
//

////////////////////////////////////
// CSelectSimmodProject dialog

class CSelectSimmodProject : public CDialog
{
// Construction
public:
    CSelectSimmodProject( CWnd* pParent = NULL );    // standard constructor
// Dialog Data
   //{{AFX_DATA(CSelectSimmodProject)
    enum { IDD = IDD_SELECT_SIMMOD_PROJECT };
    CString      m_SelectSimmodRootDirectory;
    CString      m_SelectSimmodProjectExtension;
    //}}AFX_DATA

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CSelectSimmodProject)
protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    // Generated message map functions
   //{{AFX_MSG(CSelectSimmodProject)
    afx_msg void OnBrowse();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};
```

```

// SetProjectOrigin.h : header file
//

////////////////////////////////////

// CSetProjectOrigin dialog

class CSetProjectOrigin : public CDialog
{
// Construction
public:
    CSetProjectOrigin(CWnd* pParent = NULL);    // standard constructor

// Dialog Data
   //{{AFX_DATA(CSetProjectOrigin)
    enum { IDD = IDD_SET_PROJECT_ORIGIN };
    double          m_SetOriginEast;
    double          m_SetOriginNorth;
    //}}AFX_DATA

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CSetProjectOrigin)
protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:

    // Generated message map functions
   //{{AFX_MSG(CSetProjectOrigin)
afx_msg void OnUseDefaultOrigin();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

```

```

// Settings.h : header file
//
// This class defines custom modal property sheet
// CSettings.

#ifndef __SETTINGS_H__
#define __SETTINGS_H__

#include "CSettingsPages.h"

/////////////////////////////////////////////////////////////////
// CSettings

class CSettings : public CPropertySheet
{
    DECLARE_DYNAMIC(CSettings)

// Construction
public:
    CSettings(CWnd* pParentWnd = NULL);

// Attributes
public:
    CSettingsSimmod p_SettingsSimmod;
    CSettingsSimulation p_SettingsSimulation;

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CSettings)
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CSettings();

// Generated message map functions
protected:
    //{{AFX_MSG(CSettings)
    // NOTE - the ClassWizard will add and remove member functions
here.
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////

#endif // __SETTINGS_H__

```

```

// ProgressStatusBar.h : header file
//
const int PROGRESS_CTRL_CX = 160;

const int X_MARGIN = 5; // X value used for margins and control spacing
const int Y_MARGIN = 2; // Y value used for margins and control spacing

////////////////////////////////////
// CProgressStatusBar window

class CProgressStatusBar : public CStatusBar
{
// Construction
public:
    CProgressStatusBar();

protected:
    CProgressCtrl    m_ProgressCtrl;
    CStatic          m_ProgressLabel;
    BOOL             m_bProgressMode;
    int              m_nProgressCtrlCX;

// Attributes
public:
    void SetProgressCtrlWidth(UINT nWidth = PROGRESS_CTRL_CX);
    void RecalcProgressDisplay();
    void SetProgressLabel(LPCSTR lpszProgressLabel);
    void ShowProgressDisplay(BOOL bShow = TRUE);

    CProgressCtrl * GetProgressCtrl() { return &m_ProgressCtrl; }
    void SetProgress( UINT progress );
    void Done();

// Operations
public:

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CProgressStatusBar)
    //}}AFX_VIRTUAL

// Implementation
public:
    virtual ~CProgressStatusBar();

    // Generated message map functions
protected:
    //{{AFX_MSG(CProgressStatusBar)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnPaint();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

////////////////////////////////////

```

```

// CG: This file was added by the Splash Screen component.

#ifndef _SPLASH_SCRN_
#define _SPLASH_SCRN_

// Splash.h : header file
//

////////////////////////////////////

// Splash Screen class

class CSplashWnd : public CWnd
{
// Construction
protected:
    CSplashWnd();

// Attributes:
public:
    CBitmap m_bitmap;

// Operations
public:
    static void EnableSplashScreen(BOOL bEnable = TRUE);
    static void ShowSplashScreen(CWnd* pParentWnd = NULL);
    static void PreTranslateAppMessage(MSG* pMsg);

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CSplashWnd)
    //}}AFX_VIRTUAL

// Implementation
public:
    ~CSplashWnd();
    virtual void PostNcDestroy();

protected:
    BOOL Create(CWnd* pParentWnd = NULL);
    void HideSplashScreen();
    static BOOL c_bShowSplashWnd;
    static CSplashWnd* c_pSplashWnd;

// Generated message map functions
protected:
    //{{AFX_MSG(CSplashWnd)
    afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
    afx_msg void OnPaint();
    afx_msg void OnTimer(UINT nIDEvent);
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#endif

```

```
// Splitter.h : header file
//

////////////////////////////////////
// CSplitter frame with splitter

#ifdef __AFXEXT_H__
#include <afxext.h>
#endif

class CSplitter : public CSplitterWnd
{
    DECLARE_DYNCREATE(CSplitter)

// Attributes
public:
    CSplitter();

// Operations
public:
    BOOL ReplaceView(int row, int col, CRuntimeClass * pViewClass, SIZE size);

// Implementation
public:
    virtual ~CSplitter();

    // Generated message map functions
    //{{AFX_MSG(CSplitter)
        // NOTE - the ClassWizard will add and remove member functions
here.
        //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

////////////////////////////////////
```

Vita



*Todd Alan Peterson, EIT
Candidate, MSCE*

Todd Alan Peterson was born to Robert and Mary Lee Peterson on the first of February, 1970 in San Mateo, California. Raised in Clifton, Virginia, he attended James W. Robinson, Jr. Secondary School in Burke, Virginia. His lifelong appreciation for the outdoors is exemplified by his attaining the rank of Eagle Scout in May of 1988.

He holds a B.S. in Civil Engineering from the Virginia Polytechnic Institute and State University in Blacksburg, Virginia. He has worked with the Virginia Department of Transportation, the Center for Transportation Research at Virginia Tech, and is currently employed by Anderson & Associates, Inc. in Blacksburg, Virginia. There, his responsibilities span a dual role as assistant project manager and project engineer, performing project management and engineering tasks for highway design and land development projects.

Following completion of his Masters' degree, his professional plans include becoming licensed as a Professional Engineer and to continue his work with Anderson & Associates. He is engaged to Tina Marie Loane whose family recently moved to Virginia Beach, Virginia from Hilversum, in the Netherlands.