GEOMETRIC MODELING OF FLIGHT INFORMATION FOR GRAPHICAL COCKPIT DISPLAY

Capt Mark A. Kanko, Maj Phil Amburn
Air Force Institute of Technology
Wright-Patterson AFB, OH

Abstract
The purpose of this investigation was to design and implement a computer graphics-based environment capable of modeling tactical situation arenas as viewed from the cockpit. A user can position mountains, hostile threat envelopes, and a projected flightpath through the region. The resulting three-dimensional models were to be used to prototype new graphical display formats for future aircraft. An overall goal of this investigation was to allow the cockpit display researcher to create an entirely new tactical situation display model in less than one hour via mouse input.

1 INTRODUCTION
As aircraft flight and weapon systems get more complex, the pilot's job gets more difficult as well, especially when in a hostile environment. For this reason, efficient methods of data presentation must be developed so the pilot can more easily comprehend, but not be overwhelmed by the information he must process to successfully complete all phases of a mission.

Much of the systems and status information needed by the pilot to perform a mission is currently presented in alphanumeric formats. A more efficient presentation method is pictorial formats.

In 1980, the U.S. Air Force Flight Dynamics Laboratory (AFWAL/FL) at Wright-Patterson AFB began investigating methods to compress much of the data needed by the pilot into graphical formats suitable for various cockpit displays. The research was initially based on computer graphics portrayal ideas which were first implemented in the Navy F-18 aircraft [6:3]. Under a 13-month government contract beginning in May 1980, McDonnell Douglas Corporation developed artists' conceptions of various cockpit displays.

This study intentionally neglected the then-current technology and budget constraints associated with the implementation of any of the displays. Further, the contractor was instructed to rely primarily on pictorial representation, using alphanumeric only when absolutely necessary [6:4]. The final report [3] depicted eighteen formats including those for primary flight displays and tactical situation displays.

In September of 1981, Boeing Military Airplane Company undertook a government contract to implement (in a simulator environment) the displays developed by the McDonnell Douglas contract effort. Their study evaluated pilot acceptance of the pictorial formats using three different types of hardware displays. The final report [7] contained photos of the displays actually used as well as questionnaire responses provided by pilots when asked to assess the usefulness/desirability of the pictorial displays.

The questionnaire responses pertinent to this research effort involve the displays used for threat warning: Pilots felt the displayed threat envelopes were too boxy or harsh and therefore were not optimized depictions of how they perceived the threats. Further, the envelopes were opaque so no information of what might exist beyond the closest threat was available. It is possible that another, more lethal threat envelope or the target itself was being masked by the closer threat envelope. The intent of this research effort was to provide a modeling tool that could more accurately represent the tactical environment as the pilot perceived it.

A major tool currently used by the Crew Systems Development Branch (AFWAL/FIGR) to explore new display techniques is the Microprocessor-Based Application of Graphics and Interactive Communication (MAGIC) Cockpit. This cockpit is used to prototype new graphical display formats. Display formats containing flight and navigation information, system advisory and status information, and tactical situation information are generated and displayed to pilots in a dynamic mission simulation. Afterwards, the
Pilots are asked to assess the effectiveness of the individual displays.

Prior to the design effort described in this paper, the generation of a tactical situation model containing terrain, flight path, and threat information required modifying software that generated displays in the MAGIC Cockpit. Consequently, the model could not be easily changed. Further, as the number of objects in the model increased, the amount of time consumed to build the model and the chance of errors being introduced both increased. A software package that would allow a non-programmer to generate a tactical situation model of varying resolution or complexity in a reasonable amount of time was needed.

2 USER REQUIREMENTS

To create tactical situation displays that looked similar to the artist conception formats produced by the 1982 McDonnell Douglas study [3], we provided three main capabilities to the user: (1) three-dimensional model generation, (2) short time for modeling sessions, and (3) database compatibility.

The first capability was to generate the three-dimensional descriptions of objects used to model threat envelopes, mountains, and flightpath (specified by waypoints). These characteristics are a mixture of modeling and display issues that must be separated to correctly implement the modeling system. The distinction between the two will be further explained in the next section.

The second capability was to reliably generate these tactical situation models interactively in a relatively short amount of time (defined as one hour). Additionally, the model had to be easily modifiable once it was created. This would make it the most useful in the fast-prototyping environment of the MAGIC Cockpit.

The third capability was that of database compatibility. The database representing the model must be transportable between differing computer systems. Most display formats would be developed by the user on the Silicon Graphics IRIS 3130 workstation and then moved over to a faster computer for real-time display. One potential real-time system was a General Electric Compuscene. Data compatibility between systems was critical.

3 DESIGN CONSIDERATIONS

3.1 User Interface

As mentioned above, a fundamental requirement was the ability to easily modify any part of the displayed model. The user should be able to easily change the level of detail dedicated to the representation of the terrain, the threat objects, or the flightpath.

The user requested that a mouse input device be used to position the threats and flightpath in the model. The screen format used when laying out a tactical region should resemble that shown in Figure 1. This particular format is typically called a horizontal situation display.

When laying out such a tactical situation, the user wanted to interactively specify (in nautical miles) the dimensions of the tactical region being created. For example, the user may start a design session knowing that the region being modeled extends 320 nautical miles to the east and 220 nautical miles to the north of some reference point.

When creating a flightpath, the first waypoint was designated by "I" (for "initial"). All subsequent waypoints were sequentially numbered starting with "1". Additionally, the user wanted to specify waypoints either (1) in absolute east and north displacements from the origin (the southwest corner of the terrain region) or (2) in terms of range and heading from the previous waypoint. For the case of Waypoint I, the range and heading were relative to the origin.

![Figure 1. Example Threat Locations with Flightpath](image_url)
3.2 Modeling, Display, and Data Representation Issues

In a modeling environment, it is sometimes difficult to separate the attributes that allow the model to be defined from those that allow the model to be displayed. If the distinction between the two is not clearly model, incorrect interpretation might result.

These comments are not meant to imply that display issues should not be considered when constructing the model. Organization of the data representation can have significant impact on the performance of the display system. However, the efficiency of the data representation was not of major concern for this application. In this case, compatibility with other systems was more important.

In order to keep the data representation as generic as possible, both color components and transparency values were expressed in normalized values. The red, green, and blue color components of a polygon were expressed individually as floating point values ranging from 0.0 to 1.0. A value of 1.0 for the red component of a polygon represents maximum red intensity regardless of the system being used to display the model. The transparency of a polygon was also expressed as a floating point value ranging from 0.0 for opaque to 1.0 for completely clear. Additionally, all surface normals were normalized so that only unit normals would exist in the data representation.

Together, these conventions strengthened the distinction between display and modeling issues in the modeling environment. No thought of how the model will be displayed (display issue) is given when the database is being created (modeling issue). This approach was acceptable because the user did not require a dynamic model that could change in real-time. Instead, the model would remain static while the viewpoint was moved through it.

3.3 Data Representation

The user specifications outlined the requirements for a modeling system capable of building a model representing a tactical situation scenario. The result of a modeling session was to be a data representation of the model that could be interpreted and displayed by a number of dissimilar computer systems. The data representation chosen was a critical component in the success of the entire effort.

There are actually two types of data representations of concern here: physical and logical. The physical representation deals with the format of the data, e.g., ASCII or binary. The logical representation deals with the abstract objects being modeled, e.g., points, lines, or polygons.

When considering the underlying data structures for a modeling environment, one must consider the varied representation of data that can be used. Some representations are more economical, others are more explicit and possibly more redundant. The drawback of the more economical (less explicit) representation is that some information always has to be recomputed when it is required. The drawback of the less economical (more explicit) representation is that it physically takes up more space.

Since the user required the data representation of the model be transportable, an ASCII format was chosen. In terms of storage requirements, this was an inefficient data representation but it was necessary to support the transportability requirement. This is mainly a physical data representation issue.

Related to transportability is the issue of clarity: What good is a data representation if, when ported to another dissimilar system, it cannot be easily used because of unclear meaning? The data representation needed to be as generic as possible. That is, the meaning of the data representation must be clear, even if separated from the modeling tools that created it. This moves toward a more inefficient (more explicit) representation. This is mainly a logical data representation issue.

3.4 Abstraction Levels

The concept of a model is easy to grasp. Models are used around us every day: weather forecast models, financial models, and population models, to name a few. We understand that "the model" will allow simulation, testing, and prediction of the behavior of the entities modeled for such purposes as understanding, visualization, experimentation, and learning [2;319]. But how does the model help this "understanding"? The model is a kind of superstructure supporting levels of abstraction that promote a better understanding of the problem.

When levels of abstraction in a modeling system are provided to a user, he can ignore the lower, less important details of the modeling task at hand. Although the model may have to deal with these details at some lower level, the user does not need them. Abstraction allows the user to focus his attention on the level he is most concerned with. His understanding of the basic problem will not be cluttered by unnecessary detail.
As an example, consider the design and construction of a house. The architect wants a modeling environment that allows him to express the house in terms of walls, windows, and doorways, etc. The building contractor is more concerned with the lumber, glass, and hinges needed to build the physical representation the architect’s plan (model). If the architect is forced to work at the contractor’s abstraction level, his assigned task will be much harder to accomplish. A single model could provide both levels of abstraction simply by presenting the model differently to each user. However, the current level of abstraction must always be clear.

The level of abstraction provided by the system is more completely understood when the lowest component modifiable by the user is specified as well. In the above example, how low will our house construction model allow us to go? Is a wall the smallest manipulable unit, or does the model allow wall studs to be moved and paneling seam placement to be specified as well? The user must know the level of detail provided at each level of abstraction.

3.5 Procedural Models

In his dissertation, Carlson presented the mathematical foundations, issues, and techniques for generating geometric data. His goal was to provide for the intuitive creation of objects possessing "...significant geometric and topological complexity" [1:93]. One of the techniques covered was the procedural model [5].

Newell defined a procedural model as being a model which represents its subject as a procedure with which other procedures can interact [5:28]. The interaction can include passed parameters that further specify the type or degree of the interaction. The model can be classified as "strongly parameterized" if the form of the represented object can vary widely depending on the parameters [5:90]. This means the use of strongly parameterized procedural models to create the objects allows the user to express a complex object with very little input.

The main procedural model used for this application creates a three-dimensional surface of revolution by sweeping a two-dimensional profile about the z axis. The profile is specified by a ordered set of connected line segments in the x-z plane (Figure 2a). This profile is then revolved to form a symmetric surface of revolution about the z axis (Figure 2b). The revolution is not continuous but is broken up into a number of sectors much as a round pie is cut into slices (the user can specify how many sectors are to be used). The resulting surface is composed of polygons bounded by the

![Figure 2. Procedural Model Examples](image)

sector boundaries and the horizontal lines connected the same profile endpoint in adjacent sectors. Note that these polygon boundaries can be viewed as the lines that define a wire mesh of the surface.

The number of segments specified in the profile directly controls the vertical resolution of the surface contour, while the number of sectors specified controls the radial resolution around the symmetric surface. By specifying only the profile and sector count, the user can construct a relatively complex object in a short amount of time. This indicates the procedural model is strongly parameterized. Each profile can be saved and used to generate a number of different objects, each having a different number of sectors in the revolution. This allows the user to run experiments that determine the relationship between display system performance versus the number of polygons displayed per object.

In order to simplify object generation, the unit normals along the two-dimensional contour are automatically generated and saved. The three-dimensional surface normals are then generated by revolving the contour normals with the profile. The direction of a contour normal is along the line that bisects the outside angle between two adjacent line segments at their common endpoint. If the particular endpoint is the first or the last in the profile, the unit nor-
mal direction is defined to be perpendicular to the segment owning the endpoint. The exception to this is when the first or last endpoint lies on the z-axis. This indicates that the object will be closed on the bottom or top, respectively. In this case, the tangent plane is parallel to the x-y plane which means the normal points straight down (-z) if it's the first endpoint or straight up (+z) if it's the last endpoint. The outside of a profile is considered to lie to the right of the profile when “walking” in the x-z plane from the first endpoint to the last endpoint (see Figure 2a).

The normalized transparency value of the object that will be generated from the profile is also included as profile data. It is assumed that one transparency value will be allowed for each object, i.e., variable transparency across the surface of an object is not supported.

To conclude the discussion on procedural models, note that Figure 2 also depicts a profile (2c) with its resulting surface of extrusion (2d). A similar procedural model will be used to create a "flightpath in the sky" by moving a profile from waypoint to waypoint.

4 IMPLEMENTATION

4.1 Modeling Process Overview

Figure 3 depicts the relationship between the modeling tools and the data representations. The user starts at Abstraction Level 2 by defining two-dimensional profiles of threat envelopes and mountains via a text-driven profile and object modeling tool. These are stored and can then be revolved in space to define three-dimensional surfaces that are then stored into an object library. These surfaces represent the objects that will be placed (instantiated) into the tactical model. After this point, the user need not work at Abstraction Level 2 unless new objects must be created.

Next, the tactical situation modeling tool is used at Abstraction Level 1 to place the previously-created objects at desired positions in the tactical region being modeled. Optionally, a previously-created model can be loaded for modification. This tool is completely menu driven via a mouse interface. When done, the user saves a modifiable version of the model in the symbolic library. Lastly, the user may optionally create the ASCII file that can be used as input to a display program.

4.2 Abstraction Levels

Two levels of abstraction, depicted in Figure 3, were needed for the tactical situation model.

The lower of the two, Abstraction Level 2, supports creation of two-dimensional profiles and objects (mountains and threats) that will be used at the higher level. Each object is specified relative to its own coordinate system. The lowest level of component detail available at this level of abstraction is the endpoints of line segments that make up a profile. Each line segment represents the edge of a polygon that will be generated when the profile is revolved in space. The primary output of this level is a polygon file saved in the Object Library. This library constitutes the interface between Abstraction Levels 1 and 2.

The higher level, Abstraction Level 1, allows the placement of objects created at the lower level as well as the placement of waypoints that define the flightpath of the aircraft. Object locations are specified relative to the two-dimensional origin of the tactical region being modeled. The user will specify the dimensions of this rectangular region before any objects are placed within it. The primary output of this level is a polygon file that can be transferred to any other computer system easily. The user will spend most of his time at this abstraction level since he is mainly concerned with modeling the tactical environment.
4.3 Data Representation

Two data representations must be used at Abstraction Level 1 (see Figure 3) to support modifiability and portability of the model. The first, Data Representation 1A (DR 1A), promotes model modification and is, in fact, the internal representation used by the main modeling tool. This particular representation depicts the model as object identifiers or icons located at particular x-y (east-north) positions in the world coordinates of the model, hence it might be considered a two-dimensional symbolic representation of the model.

DR 1A consists of the east-north positions of all mountains, threats and waypoints. An altitude value for each of the waypoints is also represented. Since this is a symbolic representation of the modeled objects, a pointer to the geometric data for each object is also included so the modeling tool can later construct the entire geometric model.

Data Representation 1B (DR 1B) is the second data representation implemented at Abstraction Level 1. DR 1B provides portability and is (in the form of a data file) the primary output of the modeling system.

In order to assure usability of the model representation between various hardware display systems, the format of the output representation had to be as generic as possible. The actual output took the form of a pure ASCII file to ensure transportability. The file contained a very explicit data representation comprised solely of the three-dimensional polygons that make up all surfaces in the model. This type of representation resulted in a data representation that was quite redundant in some respects. This redundancy was a direct result of each polygon being explicitly specified.

It’s important to understand that DR 1B and the graphical representation cannot be directly generated from DR 1A because the geometric description of each individual object is not included in DR 1A. This information must be retrieved from object descriptions created at Abstraction Level 2.

Once output to the data file, each polygon is independently defined, therefore it loses its association with all other polygons (except that all polygons are specified relative to the same origin). This polygon independence also results in each polygon losing its association with any particular object, so the modeling tool can no longer modify this data representation.

DR 1B consists of a collection of polygons. The information needed to specify each polygon follows:

1) Number of vertices
2) Transparency value of polygon
3) Ordered list of vertices; each vertex specified as:
   3a) x, y, z position
   3b) x, y, and z components of unit normal that is normal to the surface at this vertex
   3c) red, green, and blue components of the vertex color.

Two data representations must be used at Abstraction Level 2 (see Figure 3) to support object generation via procedural models. The first, Data Representation 2A (DR 2A), is used to represent the two-dimensional profile.

DR 2A consists of a set of points representing the endpoints of the line segments that make up the profile. The information needed to specify each profile follows:

1) Transparency of the object resulting from this profile.
2) Ordered list of segment endpoints; each endpoint specified as:
   2a) x-z position values
   2b) x and z components of unit normal that is normal to the line tangent to the profile at this point.
   2c) red, green, and blue components of the endpoint color

Data Representation 2B (DR 2B) is the second data representation implemented at Abstraction Level 2. DR 2B is used to represent the three-dimensional objects generated by the procedural models. Note that the DR 2B format is identical to that of DR 1B. The difference is that DR 2B is assumed to be specified in object-centered coordinates while DR 1B is specified in the world coordinate system of the entire tactical region. If the tactical situation model were to be considered an object itself, it would, in fact, be specified in its own object-centered coordinate system. However, this violates the abstraction levels defined for the modeling system.

DR 2B consists of a collection of polygons. The information needed to specify each polygon follows:

1) Number of vertices
2) Transparency value of polygon
3) Ordered list of vertices; each vertex specified as:
   3a) x, y, z position
   3b) x, y, and z components of unit normal that is normal to the surface at this vertex
   3c) red, green, and blue components of the vertex color.
4.4 Coordinate System Conventions

The x, y, and z axes of the data representation coordinate system were mapped into east displacement, north displacement, and vertical altitude, respectively, in the modeling environment. Data representation coordinates were all specified in feet. East and north displacements within the model were specified in nautical miles while altitude was specified in feet. This necessitated some conversion between the (physical) data representation and the (logical) model as the data representation used the same units for all displacements.

4.5 Hardware, Software, and Firmware

All applications were developed for the Silicon Graphics IRIS 3130 graphics workstation. An extensive graphics library residing on the system provides access to any graphics routine via subroutine calls. The 3130 comes standard with 32 bitplanes that support a number of different display modes. The two used for this project were double-buffer mode for tactical situation modeling tool and single buffer mode with z-buffering for the model viewing tool.

Double buffering [2:84] provides smooth movement of graphics objects without any perceived flicker. This was necessary in the tactical situation modeling tool so objects could be dragged into position and so control bars specifying position and altitude could be utilized.

The IRIS provides a powerful window manager called "mex" which allows multiple windows with pull-down menus to be easily controlled from an application program.

On the positive side, the user interface provided by mex is excellent. One of the sub-goals when implementing the tactical situation modeling tool was to restrict most user inputs to the mouse device. This sub-goal was met, but only because the window manager took care of so many tasks.

On the negative side, the special subroutine calls that must be included in an application program to use mex make the code non-portable to other systems (note that this is also true for the other graphics routine calls). Further, when invoked, mex requires two dedicated bitplanes for its own use when in single buffer mode; four when in double-buffer mode. This can be a limitation, depending on the number of bitplanes needed for color. It was this restriction that prevented mex from being used in the model viewing tool.

Another feature provided by the Silicon Graphics was polygon backface removal. This technique checks to see if the viewer is on the "inside" or "outside" of the plane of a polygon based on the normal to the polygon. The direction of the normal vector is calculated from the order in which the polygon vertices are specified. If the viewer is on the "inside" of a polygon, it will not be rendered. This technique works best for solid volumes modeled with polygons because there will always be an "outside" face toward the viewer, regardless of his position. All surfaces of a solid object disappear (i.e., are not rendered) if the viewer's position happens to lie within the solid object being modeled.

A nice feature that the 3130 did offer was textured pattern fill for polygons. This would effectively allow an alternating pattern of background and foreground color to be used when rendering a polygon. The background color portions of the polygon appear to pass the color of whatever is behind the polygon. A simple semi-transparency viewing tool was implemented to see what the results looked like.

5 CONCLUSIONS

The overall purpose of this effort was to develop the capability to easily model tactical situation displays for the Air Force Flight Dynamics Laboratory. The implementation of this capability took the form of a number of interactive computer applications that can model mountains, threat envelopes, and flightpath shapes (channels). By using these programs, the user has been able to reduce his model construction time from 1-4 weeks down to less than a day.

The quality of the modeling system user interface depends on how well the system presents the model abstraction to the user. Levels of abstraction within the model help the user to organize his activities in a more logical manner. The modeling environment developed during this effort employs two levels of abstraction: a lower one for modeling objects (mountains and threat envelopes); a higher one for modeling the entire tactical region made up of terrain, hostile threat regions, and a projected flightpath.

The data representation chosen for a model is closely related to abstraction levels used within the model. Careful consideration must be given to the choice of data representation as the performance of the modeling system in terms of speed and ease of use is directly related.

The data representation chosen for this application allowed the model to be easily modifiable. Additionally, the machine-independent format allowed for transfer between dissimilar computer systems. This allows a tactical situa-
tion model to be created and verified on a prototyping system before it is transferred to a faster, simulator-type environment.

More detailed information including photographs of results, sample data files, and design methodology is contained in [4].

6 ACKNOWLEDGEMENTS

This effort was sponsored by the Crew Systems Development Branch within the USAF Flight Dynamics Lab, and in particular, Dr. John Reising. A special thanks to the Armstrong Aerospace Medical Research Lab for providing a Silicon Graphics IRIS 3130 computer in support of graphics research at the Air Force Institute of Technology.

References


