Integration of Flight Simulators and Geographic Databases

Alexandra Diehl

Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Ciudad Universitaria
Pab. I, Universidad de Buenos Aires,
Buenos Aires, Argentina,
adiehl@dc.uba.ar

and

Horacio Abbate

CITEFA (Centro de Investigaciones Científicas y Técnicas para la Defensa),
Buenos Aires, Argentina,
habbate@citefa.gov.ar

and

Claudio Delrieux

Departamento de Ing. Eléctrica y Computadoras, UNS - IIIE Instituto de Investigaciones en
Ingeniería Eléctrica, CONICET,
Bahía Blanca, Argentina,
cad@uns.edu.ar

and

Juliana Gambini

Instituto de Ciencias-Universidad Nacional General Sarmiento
J. M. Gutiérrez 1150 -Los Polvorines, Buenos Aires, Argentina
juliana.gambini@gmail.com

Abstract

This work proposes the integration of geographical databases and topographic environment models within current flight simulators technology. This process requires the definition of standard interfaces to integrate different geographical data sources in the topographic environment model, then the visualization and edition of the topographic environment, and finally the preparation of the scene for the flight simulator.

The combined use of geographical databases and the emerging GPU computational power provides an opportunity to develop interactive applications that use massive geospatial data. This allows the visualization of geospatial databases in real time using low cost desktop computers.

Keywords: flight simulators, geodatabases, geospatial data, geovisualization.

1. INTRODUCTION

The use of geographic databases is a powerful tool to perform qualitative and quantitative data interpretation. On the other hand, 3D visualization of topographic environments and associated geospatial information allows the exploration of a diversity of geographic phenomena, global climate changes, environment disaster prevention, ecological studies, and real time remote sensing and tracking. Both contribute to increase the analysis and understanding capabilities of these and other related subjects.

Such goals can be achieved by integrating flight simulator technologies together with geographic databases and scientific visualization. This may allow a virtual navigation of real or synthetic geographic environments, dynamic

updating based on the navigation state (flight attitude, position and speed), and graphic representation of abstract ancillary data (i.e., climate data).

The topographic environment may also contain information about terrain features, and georeferenced objects (facilities, geographical accidents, roads, etc.) Furthermore, additional data layers can be added from other available information sources related to Earth Sciences repositories.

This work presents an integration process that has three main parts. The first part involves the definition of standard interfaces for storing and exchanging data between different available data sources. The second part involves the interactive visualization and edition of topographical models. Finally, we overview the preparation process of exporting topographical environment data and associated geospatial information of the scenes (using their own internal data format) into the flight simulator.

This process is currently a major challenge in geovisualization. The definition of simple standards is needed to exchange diverse and massive information, as well as the design of a generic and extensible model to allow the addition of new geographical data taken from different sites in different data formats, and the construction of a visual system to manage potentially huge geographic terrain models and large amounts of geospatial information.

2. RELATED WORK

2.1 Data Interface Standards

To further the integration of geographic information, the Open Geospatial Consortium (OGC) [1] proposes a set of technical documents and specifications grouped in a standard called OpenGIS [3]. The main goal of this standard is to generate open and public interfaces and encodings to store and exchange spatial data all over the world. OpenGIS implements Geography Markup Language (GML) [11] as a standard language to store and exchange geospatial information. GML is an XML grammar written in XML Schema for the description of application schemata as well as the transport and storage of geographic information.

For its product Google Earth, on the other hand, Google has developed its own markup language called Keyhole Markup Language (KML) [2] to exchange and transport geographic data. KML is a language focused on data visualization, including maps, and annotations using the Internet. Its community is wide and varied. Casual users upload their information to Internet using KML to add spatial references to their own homes, to describe trips, adventures, and also scientists share information of models, climate phenomena, volcanic eruptions, weather patterns, earthquake activity, etc. Organizations such as National Geographic, UNESCO, and the Smithsonian Institute use KML to display their rich sets of global data as services using the Internet and other electronic media.

KML and GML have recently joined. As a result of an agreement between OGC and Google, KML 2.2 is now a part of GML, and it uses certain elements derived from GML 2.1.2.

2.2 Flight Simulator Topographical Models

Even though there are several widespread GIS commercial applications that are capable of visualizing topographic environment models and geospatial data (using static imaging, maps or other global-scaled 2D/3D representations), most of them adopt a global or high-scale representation standpoint of the information layers. The interactive standpoint, instead, represents the information using a dynamic, interactive viewpoint. This interactive standpoint can be achieved by using flight simulators.

Flight simulator technologies, first used for military goals, evolved to wider applications, including entertainment and games, offering better and more realistic 3D visualization of topographies. Nowadays, there is a wide variety of civil flight simulators, some are commercial applications and others are open source applications, and each of them has their own topographical model. There are only a few works on 3D visualization and exploration of geographic information using flight simulators. Among them, McDonald et al. [13] developed a system for geographic visualization of landscape patterns, ecosystems and biodiversity, using flight simulator technologies to explore geographic regions and also Google Earth has an integrated flight simulator as a new navigation feature.

Probably the most popular commercial flight simulators are FlightGear 1.0 [6] [7], X-Plane 7.3 [9], and Microsoft Flight Simulator 2004 [8], and the de-facto standard OpenFlight [4] [5] to build flight simulators scenes.

2.2.1 FlightGear

FlightGear Simulator is a product of an open-source, multi-platform, cooperative flight simulator development project. Its topographic model divides the Earth into tiles limited by the Earth's latitude and longitude lines. Each tile itself is properly a scene. The scene's structure could be separated in three main parts: terrain elevation data, airports geometry information, and other objects information over the topography.

2.2.2 Microsoft Flight Simulator (MSFS)

MSFS is a flight simulator commercialized by Microsoft. Its topographical model also divides the Earth in tiles that are parallel to the Earth's latitude and longitude lines. Each tile is constructed with a set of textures: water textures, terrain textures, and texture ground polygons used to model object geometry structures.

2.2.3 X-Plane

X-Plane is a flight simulator built by Laminar Research. It also divides the Earth in a grid with rows and columns parallel to the latitude and longitude Earth lines. The model is organized in three types of information: terrain data, airport data, and navigational aids (i.e., light beacons and windsocks, etc.).

2.2.4 OpenFlight

OpenFlight is described by a hierarchical tree structure representing the scene components. There are several types of nodes that model levels of detail, degrees of freedom, sound, instancing, replication, animation sequences, bounding volumes for real-time culling, lighting features of the scenes, light points and light point strings, transparency, texture mapping, material properties, and many other features (see Fig. 1).

Figure 1: Class diagram of a simplified OpenFlight specification.

The different models described above have several common features. All of them divide the Earth in a uniform grid. Each cell of the grid contains information about the terrain elevation data and objects positioned over the terrain. These objects can be point-based positioned over the terrain (i.e., light beacons), or distributed in an area (i.e., rivers, roads, etc).

On the other hand, these models differ in the representation and semantics used to model entities. The first three flight simulators classify entities with a strong semantics and the last OpenFlight standard does not give semantics to the entities. Instead, it has a generic and hierarchical tree model to represent entities and their relationships.

3. DATA MODEL DESIGN

The goals of this data model are to represent topographical environments independently of their extension, to allow a high scale model of the overall terrain surface, and to enable setting up its own reference system and modeling real existent areas, in real time. Our model is intended to comply with the OpenFlight de-facto standard and other widespread flight simulator SDKs that may incorporate topographical data models (for instance, Microsoft Flight Simulator, FlightGear, and X-Plane). However, the design model is not limited to flight simulators, since our aim is to extend it to a wider list of simulators, i.e., terrestrial, undersea, pluvial and urban simulations.

3.1 Topographical Information

Fig. 2. shows a class diagram representing a subset of our topographic environment data model. Our model divides the Earth in a uniform grid with parallel lines to the latitude and longitude lines. The main idea is to factorize the Earth (Earth class) in a set of areas (Area class). Each area represents a tile (Tile class) in a grid (Grid class) with limits that are parallel to the latitude and longitude lines. An area has a covered surface extension (Coverage class) and a spatial reference system (Spatial ref sys class) containing information about the coordinate system type, datum, and projections. The area contains information of different layers. Also, raster information about latitude, longitude, and height, is represented in a Terrain layer (Terrain class), and a weather layer (Weather class) containing georeferenced data about weather (temperature, humidity, wind speed and direction, etc) that can be read online from existing data services (i.e., National Climatic Data Center, NOAA [14] [15]). The model supports different Levels of Detail (Level class), dividing each area in one or more reticulated divisions. Each grid tile keeps information related to the initial position using the original geographic coordinates.

3.2 Scene Model Objects

Fig. 3. shows a class diagram representing the scene objects of our data model. Each area has an associated model list (ModelList class) that represents the objects that are contained on that parcel of the terrain and that compose the scene. For this work, we have only modeled punctual positioned objects. A model (Model class) is a high level object that can be composed of one or more components (Component class). Each component is associated with a local coordinate system (LocalCoordSys class), and a box or a minimal volume that includes it (Bounding Box class). The components are built from geometric structures (Geometry class) that include faces (Face class), meshes (Mesh class), or lighting points (LightPoint class). Each geometry is formed by one or more vertices (Vertex class) that have the position of the point in the space in local coordinates to the object they belong to. The Vertex class stores the vertices of each component. These vertices are shared among each of the geometries using the Geometry Vertex class. The model incorporates the concept of level of detail by using the Skin class, SkinApplied class and LOD (Level Of Detail) class. Each component is associated to one or more skins. Skins store the textures of each Level of Detail (LOD) that will later be applied on each face, depending on the distance to the observer viewpoint. This approach allows an adequate modeling of global information, both at a very large scale representation of the entire Earth surface, and at a small scale representation of information that is meaningful to the human operator.

Figure 2: Class diagram of a subset of the adopted topographic environment model, representing topographic information.

4. IMPLEMENTATION

Implementation of this work involves each part of the complete process: data interfaces implementation, topographic data model implementation, and 3D visualization using flight simulators. Compliance with the actual standards and regulations on geographical data management and software development is an essential requirement for us when implementing data interfaces.

We have analyzed OpenGIS standards to represent geographical information and decided to use GML markup language. This provides the following options: GML schema and GML profiles. GML schema provides a template and a closed syntax while GML profile allows for adjusting the semantic and syntax to the application that is using it. For this work, we have adopted the second option, and particularly, “GML Point profile” that only uses the geometry object `<gml:Point>` to describe data interfaces between our data model and other information sources.

To implement topographical database model, there are several commercial and open source database management systems (DBMS) options that allow managing massive geographical data. These software applications include special functions, geospatial indexing and new data types to manage geospatial data. Among these available options, we have chosen PostgreSQL with PostGIS extension for the following reasons: it is a well known open source software, and its PostGIS extension to manage geospatial data is compliant with the standards OpenGIS.

The implementation of the visual application is based on OpenGL and Visual C++ technologies. It allows loading terrain elevation data and point-based objects.

Figure 3: Class diagram of a subset of the adopted topographic model representing the object modeling of the scene.

5. RESULTS

As a result of this work, we present a data model representing the topographical environments and the associated geospatial data and a 3D visualization application to explore the geographical information from several viewpoints. Our model has several advantages. It is generic, since it allows modeling simple and complex objects. It is flexible, since it allows specifying the global reference system used in the database (global coordinate system, datum, projection), and a local coordinate system for each modeled object in the scene. Our model also allows specifying several LOD (Level of Detail) that are global of a tile, or local LOD (Level of Detail) of an object. Finally, it is extensible, since it allows adding new geometries that are inherited from the geometry entity without modifying existing entities and their relationships.

Fig. 4. shows the visualization of a terrain area represented using our topographic environment model. The area belongs to a terrain grid in projected coordinates Universal Transverse Mercator UTM, Datum=WSG84, zone=12 of 169 rows and 262 columns of 30 meters each side, with a coordinate unit in meters. It has initial latitude and longitude coordinates of 247.038, 10603026 meters, and 4.205.582, 3212307 meters respectively.

Fig. 5. displays a detail of the grid and visualized entities. Three oversized objects have been placed in the terrain so that it is easier to observe them in the picture of the complete covered area.

Figure 4: Visualization of a terrain area represented with our model.

Table 1 shows the center of the projected coordinates (x,z) and the height (y) of each of the objects. Each projected coordinate (x,z) has an associated geometric data type POINT(x,y), which is native from Post-GIS with its corresponding spatial reference system to allow making spatial queries.

Table 1: Position of the objects in the scene. The measure unit is in meters.

Model Id	Center X	Center Y	Center Z	Side	Height
1	250.608,10603	3.522	4.207.682,321231	200	300
2	250.918,10603	3.522	4.207.682,321231	300	200
3	250.608,10603	3.522	4.207.992,321231	300	400

The use of PostGIS to support geographical information allows us to validate and ensure compliance with the OpenGIS standard. Furthermore, PostGresSQL allows us to make a robust implementation of our model. Also, it allows us to improve performance of database operations taking advantage of the specific functions to manage geographical data, coordinate system conversions and spatial indexing.

6. CONCLUSIONS AND FUTURE WORK

The underlying data model is implemented with geodatabase technologies and takes advantage of existing OpenGIS standards. The operation model is also compatible with the OpenFlight and other SDKs models. Also, we present a 3D visualization system for topographic environment model and its associated geospatial information.

Future work will be focused on the addition of new layers of geospatial data to the data model taken from different areas: climatology, hydrology, etc, and their visualization. Another feature we will develop is the integration of a flight simulator to explore the topographies and, also, the integration of new data interfaces to exchange data using KML and GML, and to export scenes to other flight simulators.

Acknowledgment This work was partially supported by the Research Grant PAE2004-22614 ANPCyT. Alexandra Diehl is currently supported by a CONICET Doctoral Fellowship. Claudio Delrieux is partially supported by SECyT-UNS and IIIE-CONICET.

Figure 5: Visualization of a terrain area represented with our model. Reticular grid and amplified view.

References

- [1] Open Geospatial Consortium, Inc. *OGC Web Site*. <http://www.opengeospatial.org>. 2009.
- [2] Google Inc. *KML Tutorial*. <http://code.google.com/intl/en/apis/kml>. 2009.
- [3] Open Geospatial Consortium Inc.: *The OGC. Abstract Specification Overview*. Version 5. Project Document Number 04-084. June, 2005.
- [4] MultiGen-Paradigm, Inc. *OpenFlight Scene Description Database Specification*. Version.16.3. Document revision A. November 2007.
- [5] MultiGen-Paradigm, Inc. *Creator Terrain Studio. Terrain generation innovation from MultiGen-Paradigm, Inc.* Version 1.2. 2005.
- [6] Olson, C. *FlightGear Scenery Designer*. http://wiki.flightgear.org/index.php/FlightGear_Scenery_Designer. 2009.
- [7] Basler, M., Spott, M., Olson, C. *The FlightGear Manual*. Version 1.0. December, 2007.
- [8] Microsoft, Corp. *Microsoft Flight Simulator 2004 Terrain SDK*. May, 2004.
- [9] Laminar Research: *X-Plane 8 Scenery Information Library*. <http://scenery.x-plane.com/library.php/>. 2007
- [10] Percivall, G., Open Geospatial Consortium Inc.: *OGC Reference Model* Version 0.1.3. Reference number: OGC 03-040. September 16, 2003.
- [11] Open Geospatial Consortium Inc.: *OpenGIS® Geography Markup Language (GML) Encoding Standard*. Version 3.2.1. Reference number: OGC 07-036. 2007.
- [12] PostgreSQL version 8.2.14 *Documentation. PostGIS Manual*. 2006.
- [13] McDonald, S. and Stevenson, R. D. 2002. Visualizing geo-referenced data with the Eco Flight Simulator. In *Proceedings of the 2002 Annual National Conference on Digital Government Research* (Los Angeles, California, May 19 - 22, 2002). dg.o, vol. 129. Digital Government Society of North America, 1-4.
- [14] NOAA: National Climatic Data Center: <http://www.ncdc.noaa.gov/oa/ncdc.html>
- [15] Glahn, Harry R., Ruth, David P.: The New Digital Forecast Database of the National Weather Service. *Bulletin of the American Meteorological Society*: Vol. 84, No. 2 (February 2003), pp. 195–201.