УДК 004.92

S.I. VYATKIN
Institute of Automation and Electrometry SB RAS, Novosibirsk
A.N. ROMANYUK, O.V. ROMANYUK, B.L. VOIT
Vinnytsia National Technical University

DATABASE COMPONENTS FOR VISUAL SYSTEMS

From information supplied by the simulation host, the computer that controls the simulator, the database subsystem computes a set of viewing parameters describing the location and direction from which the scene is to be viewed. The database subsystem also manages the visual database, a simplified geometric representation of the real world's scenery, and looks after weather and other special effects. The database components are presented in this paper.

Keywords: Visual system, Database subsystem, Database components, Database modeler

С.І. ВЯТКИН
Інститут автоматики та електрометрії СВ РАН, Новосибірськ
А.Н. РОМАНЮК, О.В. РОМАНЮК, Б.Л. ВОЙТ
Вінницький національний технічний університет

КОМПОНЕНТИ БАЗИ ДАНИХ ДЛЯ ВІЗУАЛЬНИХ СИСТЕМ

З інформації, яка отримується від хост- процесора системи візуалізації, підсистема бази даних обчислює набір параметрів, що описують місце розташування та напрямок, з якого сцена готова для перегляду. Підсистема бази даних також управляє візуальною базою даних, спрощує геометричне подання реального світу, з урахуванням погодних умов та інших спеціальних ефектів. Компоненти бази даних описані в цій статті.

Ключові слова: візуальна система, підсистема баз даних, компоненти бази даних, засоби моделювання баз даних.

С.И. ВЯТКИН
Институт автоматики и электрометрии СО РАН, Новосибирск
А.Н. РОМАНЮК, О.В. РОМАНЮК, Б.Л. ВОЙТ
Винницкий национальный технический университет

КОМПОНЕНТЫ БАЗЫ ДАННЫХ ДЛЯ ВИЗУАЛЬНЫХ СИСТЕМ

Из информации, получаемой от хост процессора системы визуализации, подсистема базы данных вычисляет набор параметров, описывающих местоположение и направление, с которого сцена готова для просмотра. Подсистема базы данных также управляет визуальной базой данных, упрощает геометрическое представление реального мира, с учетом погодных условий и других специальных эффектов. Компоненты базы данных представлены в этой статье.

Ключевые слова: визуальная система, подсистема базы данных, компоненты базы данных, средства моделирования баз данных.

Introduction

Real-time computer graphics oriented to 3-D scene visualization has attained appreciable success nowadays. Though a sufficiently high realism of real-time scene imaging has been attained, some problems are still present (e.g., imaging of large terrain regions), where it is necessary to store and visualize scenes containing a greater number of polygons than it is implemented in the present-day systems. On the other hand, exact modeling of shapes of the car, airplane, submarine frames requires thousands of spline-surfaces (curvilinear areas defined by polynomial functions) whereas the case of definition by polygons will require tens and sometimes hundreds of thousands of polygons.

Virtual reality systems, where viewer merges into the model world, require high quality visualization. In addition there are a number of application area specific problems, for example, dynamic distortion correction, which is required for image projection to non-planar screens taking into account viewer and projection device movements. These problems are solved in high-end real-time visualization systems [1-8].

All visual systems need at least one database. A typical database for commercial pilot training will have an airport, or two, along the surrounding countryside. There's relatively little detail beyond the immediate vicinity of an airport because, with the exception of takeoffs and landings, most commercial flying takes place at high altitude. Military database, on the other hand, are highly detailed and may cover more than a million square kilometers. Database isn't easy to create and so only a small portion of the world is represented accurately.

Dabases and the Modeling Subsystem

Traditional databases consist of two elements: light points and polygons. Polygons are used to model the terrain, building, mountains, trucks, aircraft, etc. Light points are used for runway, airport and city lighting. This presents the customer with two problems. First, the light point and polygon elements form a single database. If the customer wished to increase the number of lights in a database they would have to reduce the number of polygons being displayed. And second, a large number of polygons must be used to present a high quality terrain scene. Thus when near the ground, the number of polygons available for modeling other object is limited. Databases are created by a database modeler. Maps, blueprints, photographs, and satellite data are fed through various software tools to create models of terrain, buildings, trees, airport, and vehicles. Individual models are compiled and linked to create a database, which is then translated into a form the image generator understands. Commonly-used models are kept in model libraries, enabling the modeler to simply copy them into the database rather than creating them over and over again. The database modeler, tools, and model libraries are collectively known as the modeling subsystem. Although many database can reside together on the disk, only one of them, the active database, can be in use at any given time. Models are assembled from faces, convex polygons (triangles). The sides of a face are referred to as edges, which meet at points called vertices. A nifty property of faces is that you can only see the front of them; they vanish if you try to sneak up from behind. Faces viewed from behind (and hence not visible) are said to be back-facing.

Modern databases consist of three elements: light points, surface polygons, and terrain. The terrain generation allows the system to present highly detailed scenes with no practical limits on the number of polygons used, amount of detail or fractal texture. The terrain is a two-dimensional representation of the database upon which light point and surface polygons are overlaid. The terrain element is independent of the light point and surface polygon elements. This means it is neither affected by, nor does it effect the performance of the other two elements. Also, as no surface polygons are used in terrain scene generation, 100% of surface polygons are available for modeling buildings, aircraft and other ground traffic. Databases consist of a set of specially-constructed computer-generated images of a gaming area, usually containing one or more airport together with general terrain and cultural features such as roads, fields, vegetation, forest, urban areas, buildings, lights, etc. The gaming area may be representative of the real world or it may be generic.

To appreciate the full advantage of the terrain elements, it is helpful to understand how it is generated in the scene. First, the terrain and airport are digitized from maps. The amount of detail included is simply a function of the data used and amount of time to be allocated to the digitization process.

A face's base color is just one of a number of attributes that the modeler can control to match the characteristic of the object being modeled. Base implies that the color the viewer sees isn't necessary what the modeler specified; the color of the ambient light, atmospheric haze, and other face attributes like texture all interact with the base color to produce the actual color. Transparency can be incorporated into the base color to simulate windows and canopies.

The airport area and linear features away from the airport are modeled using sharp-edged polygons. Each polygon has an assigned color and texture representing the surface appearance of the area being modeled. General terrain away from the airport is also modeled as polygonal digitized areas with assigned colors and textures, but the final database, those areas not sharp-edged. Rather, they have an irregular, fractal edge in order to improve realism.

Texture is perhaps the most important attributes of a face. Textures are patterns derived from full-color photographs, drawn by an artist, or synthesized by a computer. When applied to the face, texture adds a lot of detail and realism. Without the motion and distance cues provided by texture, low-level flight training in simulators would be almost impossible.

Shading, the intensity of the light reflecting off the surface of an object, gives the viewer important perceptual clues to the object's shape. The amount of light reflected from a surface depends largely on the slope of the surface with respect to the sun or other light source.

Most of the faces in a database are normal faces. Like many of the objects around us, they must be illuminated by sunlight, landing lights, or some other source of light to be seen. Without some form of illumination, there would be no light for them to reflect; they'd be completely black. During the day, light faces are similar to normal faces. At night, however, they emit light like bulbs, making them useful for modeling luminous signs, windows in buildings, and other large-area light sources. Unlike real lights, light faces can't illuminate their surroundings; a normal face placed next to a light won't be seen at night. Illumination faces are never displayed on the screen; instead, they illuminate the faces directly beneath them. An illumination face may placed at the base of a street lamp to give the illusion that the lamp is shining on the road below. Shadow faces are illumination faces that remove light from other faces to create shadows. While most illumination faces are added to a database by modeler, generated illumination faces are created on the fly by image generator to simulate illumination from movable sources like landing lights.

Light sources that are small enough to be treated as points, such as street light and runway lights, are modeled by light points. Light points come in two flavours: raster and calligraphic. If, as is the case with runway lights, many identical lights are spaced evenly along a line, all of the lights can be modeled with a single light string.

Like faces, there are several different types and attributes of models that the reader should be familiar with. Models that have fixed positions within the database are static models, whereas those that can be moved around are moving models.

A model is geospecific if it represents a unique feature such as an airport or generic if it is representative of a whole group of similar features like trees.

All runways, taxiways and ramp areas are modelled to accurately represent the length, width, marking and surface appearance of the actual features. In addition, tire skid marks are randomly placed in the runway touchdown zone.

The Dabases Subsystem

As its name implies, this subsystem is responsible for getting database information to the rest of the system, but it also looks after control functions and anything else that needs to been done relatively infrequently, no more than once per field interval, the time required by the display to scan from the top of one image to the top of the next.

One of the database subsystem’s most important functions is to act as an interface between the image generator and the simulation host. The host controls the operation of the visual system by sending commands to the database subsystem. There are commands to update the viewpoint, load databases, change the brightness of the scene, enable special effects, and adjust the intensity of lights in the database. The database subsystem examines the commands and sets up the image generator accordingly. It also gathers status information from the image generator at the end of every field interval, packages it neatly, and relays it back to the host.

Before simulator training can begin, the database subsystem must select, or load, whichever database is appropriate for the type of training to be done. As part of the loading process, the database is copied from disk into environment memory. Texture maps, tables, and other data that go with the database are copied into various memories distributed throughout the image generator.

All special effects are directed by the database subsystem. Many effects are weather-related, such as rain, blowing snow, lighting, and transitions into cloud and fog. Other effects include aircraft docking system, matshallers, and moving models that follow pre-determined paths through the database. Military systems support a whole slew of additional effects; smoke, explosions, tracers, wakes, and rotor downwash, to name a few.

Environment memory isn’t always large enough to hold an entire military database that may extend over a million square kilometers or more. For this reason, military visual systems often have enhanced database subsystem capable of database paging. Background database loading lets the database be copied into environment memory without interrupting training or disturbing the images currently being displayed. Database paging and background database loading are done whenever the database subsystem isn’t busy doing something else.

Most novel real-time visual systems have terrain skinning processors, which guarantee continuous levels of details (LODs) change with respect to surface roughness and viewpoint distance. One of the main reasons to incorporate terrain skinning processors in the visual system is its ability to generate terrain skin with low depth complexity and hence reduce the image generator load when compared to traditional including terrain polygons in an environment database and levels of details switching. Terrain skinning processors could be either an installable hardware device, or software process executed by geometry processor.

Terrain visualization is a difficult problem for applications requiring accurate images of large datasets at high frame rates, such as flight simulation. Many terrain skinning processors (TSP) projects use regular terrain elevation grid with square cells. It leads to algorithmic simplicity of computations, database uniformity, and strict definition of relationship between adjacent LODs, which results in database generation simplification. On the other hand regular grid obviously involves information excess when considering the number of grid posts.

If TSP is an application specific device having local processor then it might be able to generate in real-time about 1000.000 terrain triangles when using regular grid and which is a considerable part of the image generator (IG) system performance. This simulates search for a terrain skinning methods of image generator IG unloading at the cost of more heavy TSP load, and particularly non-gridded terrain. Rendering photo-realistic, complex terrain features at interactive rates requires innovative techniques. A polygonal model and geometric pipeline can be used but this introduces massive storage requirements and, ideally, a parallel implementation of the algorithm. However, features with high spatial frequency context (ridge lines and canyons) require large numbers of polygons to meet a specified level of terrain accuracy.

TSP with terrain database in its memory can be treated from the visual system point of view as a sequential access read-only file. The observer position load into TSP serve as OPEN operation for fat file, and READ operation results in flow of triangles being output to the IG accompanied with appropriate status code.

Here we wouldn’t touch problems like Intervisibility, height above terrain (HAT), Conforming, LOS Culture (LOS – could easily be solved). Nevertheless it is clear that status code can serve as a pointer to the Culture object to be inserted into the input data flow; triangle can hold an attributes like neighbour pointers serving for line of sight determination, contour outline, or shadowing.

The READ operation is executed by recursive walk down the tree composed of planes, segments and other structure elements. While plane processing the half space without observer is scanned first. If for that half space clusters the screen error is less than predefined constant (pixel, subpixel size) and there is no priority inexact between clusters then appropriate set of triangles is output else this half space is divided further or next more detailed LOD is entered. After triangles output the current tree state is stored as a file pointer.

Sphere mechanism is used for LOD change. Sphere diameter is equal to maximum error for an underlying terrain subtree, and its center is approximately at the center of this part of terrain. If screen size of the sphere is in the specified error limit then current LOD is processed else next LOD subtree traversal is accomplished.

Conclusion

Databases for the system consist of distinct components: terrain, light points, polygons, and calligraphic lights. Each component is independent of the other; this means there is no-trade off between the components. Components can be mixed and matched. For example, using the same terrain it is a simple matter to over-lay a completely different set of light points.

References

1. Романюк О.Н. Високопродуктивні методи та засоби зафарбовування тривимірних графічних об’єктів. Монографія. / О.Н. Романюк, А.В. Чорний. — Вінниця : УНІВЕСУМ-Вінниця, 2006. — 190 с.
2. Романюк О.Н. Комп’ютерна графіка / О.Н. Романюк. — Вінниця: ВДТУ, 2001. — 129 с.
3. Романюк О.Н. Особливості архітектурної побудови систем формування тривимірних зображень / О.Н. Романюк, М.Д. Обідник, О.В. Романюк, Н.С. Костюкова // Наукові праці Донецького національного технічного університету. Сер.: Інформатика, кібернетика та обчислювальна техніка. — 2010. — Вип. 12. — С. 87-93.
4. Вяткин С.И. Моделирование и визуализация сложных поверхностей на основе скалярных функций возмущения / С.И. Вяткин // Сборник трудов ДонНТУ. Серия: Информатика, кибернетика и вычислительная техника. — 2009. — Вып. 10(153). — С. 105-110.
5. Evans & Sutherland, Technical Report 517902-904 AA, Evans & Sutherland Computer Corporation, USA, September 1992.
6. Вяткин С.И. Метод формирования изображений рельефных поверхностей / С.И. Вяткин, О.В. Романюк // Реєстрація, зберігання і обробка даних. — 2009. — Т. 11. — № 4. — С. 51-58.