6
3D GRAPHIC SCENE MANAGEMENT IN DRIVING SIMULATOR

Mohd Khalid Mokhtar
Mohd Shahrizal Sunar

INTRODUCTION

Simulate is the word that have same meaning as fabricate, feign, pretend, copy, mimic, or imitate. The word “simulation” can be defined as a "technique of substituting a real environment to fake one, so that it is possible to work under laboratory conditions of control. Within experimentally controlled environment, performance measures can be defined, collected, and repeatly tested which is in a cost-effective manner (Olsen 1996).

A virtual driving simulator is a device that allows user to feel a life-like experience of driving an actual vehicle within virtual reality. It is effectively used for studying the interaction of a driver and vehicle and for developing new vehicle systems, human factor study, and vehicle safety research by enabling the reproduction of the actual driving environments in a safe and tightly controlled environment (Kang et al. 2004). Mostly vehicle simulators consist of physical mockups as examples steering wheel, gearshift and pedals. These are essential in trying to simulate real conditions, then it’s become as a drawback for system becomes more expensive, more huge (non mobile), and then limited to reflect changes on the vehicle type, dimensions, or interior design (Kallmann et al. 2003).

The virtual driving simulator environment consists of static universe, dynamic objects and interior of driver's vehicle (Kang et al. 2004). The static universe can be building, trees, road and others. The dynamics objects can include any moving objects in virtual scene like cars, people, and crowd. With more complex virtual scene will contain many thousands of polygons which need more graphic processing power and more computation cost to render the scene. Even on latest graphic hardware processing, the increase in complexity in virtual environment will increase computational power. The complex scene is needed to manage so it can be efficiently render in real time and avoid memory leak. Implement of 3d scene graphic management techniques help to reduce computational burden in complex driving simulator environment. Figure 6.1 and 6.2 shows an example of a driving simulator and complex scene in driving simulator virtual environment.

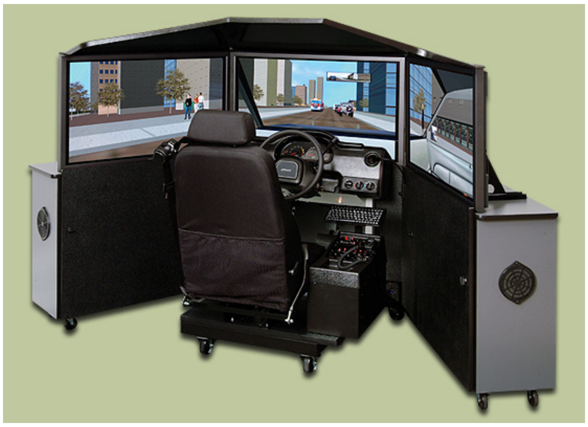

Figure 6.1 An example of driving simulator

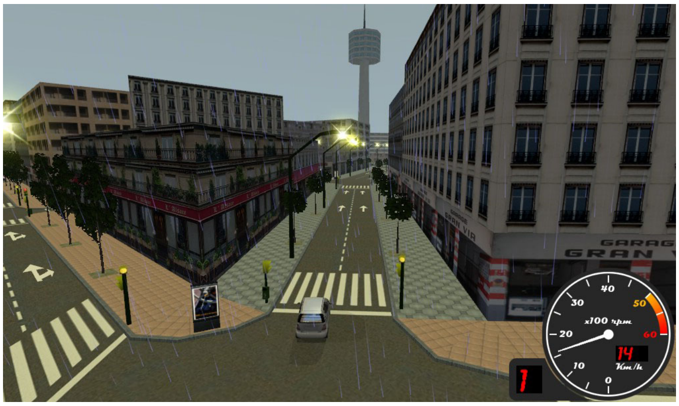

Figure 6.2 The complex virtual environment of driving simulator

HISTORY OF DRIVING SIMULATOR

It not a latest research when we related to driving simulator for its development and application. As early of 1900s, driving simulators start having their roots on flight simulators. It has begun to appear in primitive forms in the 1970s. The advent of computer technologies brought Daimler-Benz to launch a high-fidelity driving simulator in 1980s, since then brought many automotive makers and research institutions worldwide to develop their own simulators (Lee 1998).

After that, computer generated imagery was more extensively being introduced in driving simulation research on the early of 1980s. Computer-generated image systems advancement went through several stages in the 1980s, from very angular and lack of shading and detail to the appearance of photo-driven texturing. Through a significant software development cost realistic images look like real (Decina 1996).

TYPES OF SIMULATOR

Simulators can be divided as interactive and non-interactive. Interactive driving simulator mean that the simulator responds to the driver just as the driver responds to the simulator or called as a closed-loop driving simulator. Apart in a non-interactive system, the driver responds to the simulator, but the simulator does only what it was programmed to do, regardless of the driver's actions, often referred to as an open-looped system.

Interactive driving simulation represents real world such as actual roadways through computer generated imagery (CGI), auditory feedback, and realistic vehicle instruments and controls such as brake pedal, steering wheel, turn signal indicator, speedometer, and mirrors.

ARCHITECTURE OF DRIVING SIMULATION

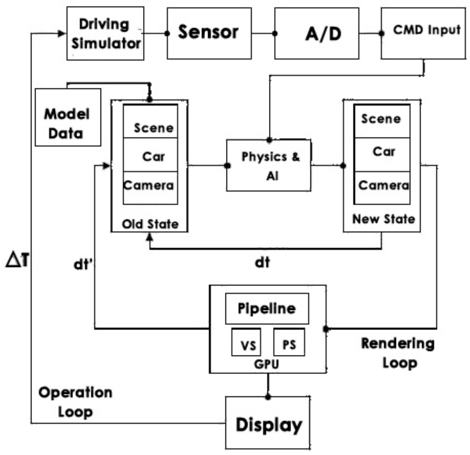

Figure 6.3 Architecture of driving simulator

As picture in Figure 6.3 from Sun et al. (2007) above shows driving simulation system architecture is a human-in-the-loop mode. Human is placed in the outer loop operating the control equipments. Then, GPU and CPU are in the inner loops functions in the work of rendering, physical simulation and AI processing. As in this diagram, triangle-arrows describe control flow and point-arrows describe data flow.

Operation Loop, driver of driving simulator send inputs control operations to the inner loops. Then in Rendering Loop, GPU renders the virtual scene to the Frame Buffer. At the same time GPU apply effect to scene using Vertex Shader and Pixel Shader. Lastly Physics Loop, the Physics engine used to calculate the driving car's position, velocity, acceleration, and orientation.

DRIVING SIMULATOR SYSTEM DESIGN

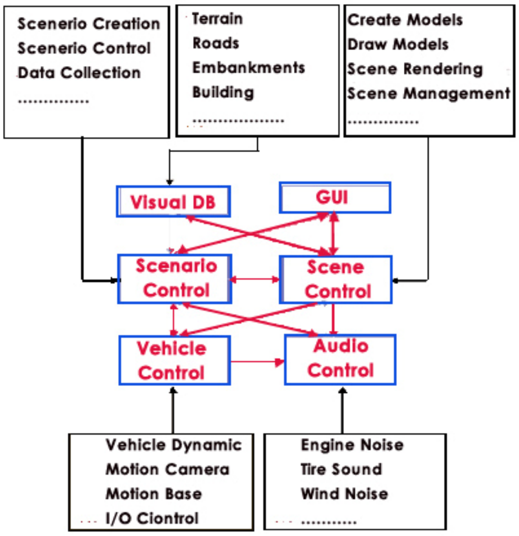

Figure 6.4 The framework of major components Software Architecture

The architecture of software has been developed to support scalability, extendibility, flexibility, and evolve ability. It main focus consists of six components: visual database, scene control, scenario control, operations control: GUI, vehicle control, and audio control. The software framework of six main components is described in the following sections and is shown in Figure 6.4.

The first component is Scene Control that functions to create or draw objects in the scene and also manage the scene contents by renders the specified views in real time. Next, Scenario Control is control component in generate and distribute viewing parameters as well as send rendering commands to the related component. GUI is component provides human-computer interface for user to interact with the system. Vehicle Control is component includes motion control and car cab I/O control.Lastly, Audio Control is component able to produce sound effects like engine, tire, wind, and tire squeal. All this compenent will interact between themselves to manipulate the database content visual model of the simulator in Visual Database.

VIRTUAL ENVIRONMENT IN DRIVING SIMULATOR

Rendering the virtual environment (VE) will includes full texturing, shading, fog, and lighting effects with an aim to stimulate a real driving environment. The development of the 3D model is the most difficult and interesting component of the simulator. The visual database of the simulator can be divided into three types of 3-D models: 3-D Street Models, 3-D roadway Models, 3-D mobile Models (Liao D. 2006).

3-D Street Models- Includes terrain, tree, buildings and trafic devices models.

3-D roadway Models- Modelled separately to allowed collusion detection between road surface and vehicle.

3-D mobile Models- The vehicles modelled so can be able to create the feel inside real driving simulator.

GRAPHICS DELAY CAUSES SIMULATOR SICKNESS

Simulator sickness was firstly documented by Havron and Butler in 1957 in a helicopter trainer. It almost similar to motion sickness, but can occur without actual motion of the subject. Simulator sickness is potential treat that will cause severe discomfort some user during or after using driving simulator (Kolasinski 1995). It is one of the biggest problems that faced many driving simulators. Two factors can probably causes simulator sickness are transport delay (time lag) and update rate (frame rate). Transport delay is a problem whereby the response of a dynamic element slower in time relative to its input (Lee et al. 1998).

Three major sources of transport delay in the driving simulator: the vehicle simulation, visual, and motion systems. Real time vehicle simulation is related with vehicle dynamics computation remains lower than the perception threshold of the driver. Next, in the visual system includes data acquisition, image processing and display time (is in the range of 25 and 50 msec depending on the quality of images). Lastly, the delay in the motion system is around 50 msec with including data acquisition, drive logic computation, and motion platform response time. As result, transport delay is critical issue in to make sure the fidelity of the simulator. The delay will degrade system performance, and further system stability.

Update rate also known as frame rate is can be defined as speed of simulation. This frame rate is different the refresh rate where it depend on the scene complexity and graphic processing power of hardware used by the driving simulators. The effects of this problem will cause visual lag. Frame rate can be referred as benchmark. When render the scene with high complexity, the frame rate will drop.

With these two factors the solution needed to minimize the problems which may be not lead directly to simulator sickness problem. The important is to ensuring the high fidelity in complex driving simulator virtual environment.

3D SCENE MANAGEMENT

The term of 3D scene management can be defined as algorithms and methods that select only the polygons that are needed for viewer depending on location and orientation of virtual camera in virtual environments (Zerbst and Duvel 2004). The scene management system is responsible for efficiently rendering complex scenes. Within complex VE, most polygons are not visible to the user needed to remove. Three scene management elements can be found in most VE (Young 2004).

i. The ability to load and destroy a scene properly.
ii. Management of scene of data, object in the scene.
iii. Display of the scene to player.

The design of scene management is different depend on types of VE. The way to do that depends on the capabilities of modern graphic adapter (Zerbst and Duvel 2004). The need of scene management is important especially when deal with complex virtual environment to minimize data to be processed and rendered for each frame. With aims to make sure suitable frame rate is processed by renderer in driving simulator. In the following sections, the main data structures and algorithms that deal with scene management can be applied in driving simulator will be explained.

Culling

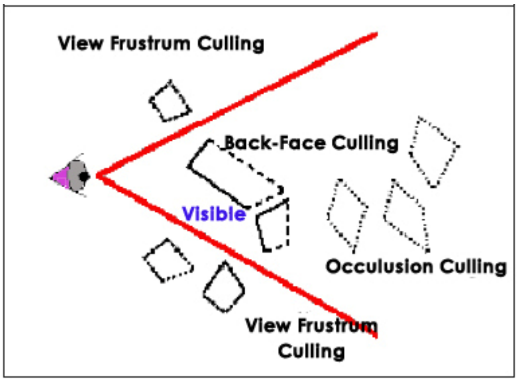

Figure 6.5 Three types of visibility culling – view-frustum culling, back-face culling and occlusion culling

Culling is defined as method that filtering of faces that not required rendered by the scene (Young 2004). Mean that any faces of polygon that do not require will not send to OpenGL or DirextX graphic pipeline. Visibility culling starts develop with two conventional techniques, which is back-face culling and view-frustum culling. Back-face culling removes geometry faces that are away from the viewer, while view-frustum culling algorithm omits the rendering geometry that is outside viewing frustum. Next, occlusion culling also known as visibility culling or output-sensitive visibility calculation is a special case of visibility calculation introduced. Occlusion culling finds these visible parts without wasting time on the occluded parts, not even to determine that they are occluded. Three types of culling can be seen in Figure 6.5.

The next figure shows the relationship between culling and frame rate per second (fps). It can be concluded that the balance between amount this method performed by CPU, the amount of polygon rendered by GPU and the amount need to do to achieve satisfaction to the result (Young 2004).

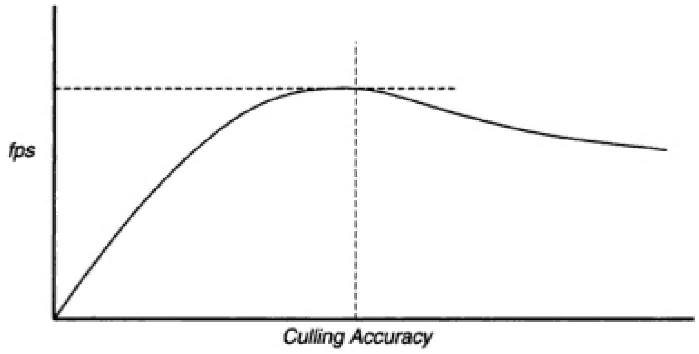

Figure 6.6 The relationship between culling and frame rate

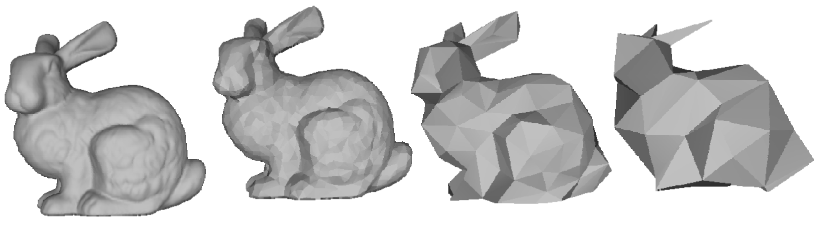

Figure 6.7 Traditional Level Of Detail

Level of Detail

Level of detail (LOD) involves reducing the complexity of a 3D object representation as it moves away from the 3D viewer in VE (Luebke et al. 2000). This technique improves the efficiency of rendering by reducing the data structures on graphics pipeline stages, usually vertex transformations. LOD used to improve system performance and quality of graphic system (Reddy 1999).

LOD also know as polygonal simplification, geometric simplification, mesh reduction, decimation, multi-resolution modeling. There are two type of LOD. The first is known as Discrete LOD (DLOD). DLOD is that create individual levels of detail in a pre-process. Secondly, considers the polygon mesh being rendered as a function which must be evaluated requiring avoiding excessive errors which are a function of some heuristic (usually distance) themselves known as Continuous LOD (CLOD). Figure 6.7 below shows us about discrete LOD..

Spatial Subdivision Structures

Spatial subdivision is defined as dividing objects based on their location. In order to accelerate the procedure searching of two objects are visible needed to answer such a query, various structures of spatial subdivision of an environment have been introduced for important procedures in occlusion culling and ray tracing. So this method is good when it combine with culling techniques. Popular spatial subdivision structures will be outlined, namely, the regular grid, octrees and bounding volume hierarchies. Others spatial subdivision structure BSP-trees and kD-trees. There are algorithms are only suitable for static scenes while some are suitable for both static and dynamic scenes because the difference amount of computation needed to update the hierarchy in real-time (Baldeve 2006).

Bounding Volume Hierarchy

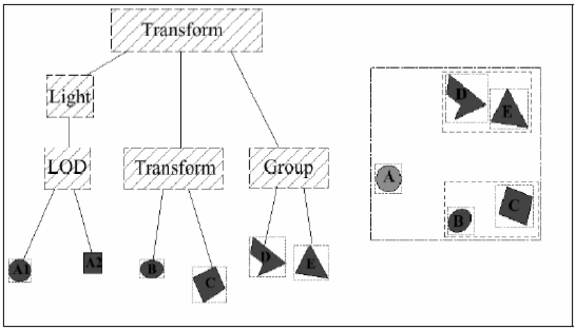

Figure 6.8 Bounding volume hierarchy

Steven Rubin and Turner Whitted (1980) introduced bounding volumes to envelope the objects in the scene to reduce intersection calculations in ray tracing. The objects are put at the correct level of the constructed tree, base on size and semantics. As example, the bounding volume of a desk is considered as a child volume of the room volume where it is in Figure 6.8.

Regular Grids

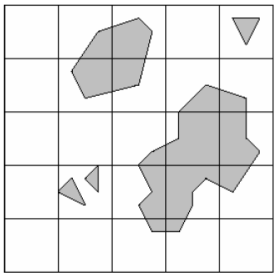

Figure 6.9 Regular grid

One of easiest spatial subdivision is the regular grids. A regular grid is placed over the 3D scene and assigns the geometry to the grid's voxel that encloses its centre point as shown in Figure 6.9. Polygon also can be clipped to fit into voxels but this can increase the amount of polygons. Total of dimension of the cells are decided early and thus this structure lacks of adaptability to the environment.

Octrees

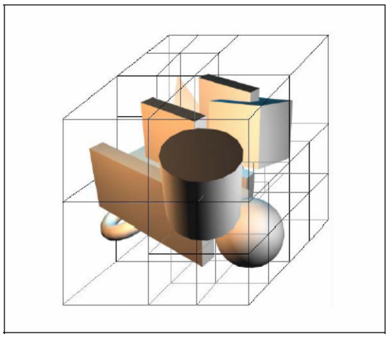
Figure 6.10 Octree space partition

Glassner (1984) introduced hierarchical subdivision of a box called octree correspond to the regular grids. Initially one voxel is used to enclose the entire scene's geometry. The every voxel is divided recursively into eight sub-voxels of equal size as shown in Figure 6.10. The subdivision continues until the maximum depth or until no object is included in the cell. Finally leaf of trees stores a list of included objects.

CONCLUSION

Driving simulator is content a complex virtual environment that needed to manage to make sure efficiently rendered by graphic processing. The first issues discussed are all related to driving simulator: history, type of simulator, architecture of the driving simulator and system design. Part that needed to be highlighted here existence of scene management in driving simulator system design. The next issues discussed are the virtual environment in driving simulator and graphic delay cause simulator sickness. The issue mark here is the increase of complexity will lead to long time lag and decrease frame rate. Last issue discuss is about what is 3D scene management and type of techniques can be used in driving simulator VE. The potentials of scene management in complex scene for driving simulator will lead used to others real time graphic system so complexity of the 3D scene able to manage to achieve fidelity and realism.

REFERENCES

Decina L. E, Gish K. W, Staplin L., and Kirchner A. H., 1996, "Feasibility of New Simulation Technology to Train Novice Drivers", Technical Report, U.S. Department of Transportation.

Glassner A., 1984, "Space subdivision for fast ray tracing." IEEE Computer Graphics and Applications

Kallmann M., Lemoine P.,Thalmann D., Cordier F., Thalmann N. M., Ruspa C., and Quattrocolo S., 2003, "Immersive Vehicle Simulators for Prototyping, Training and Ergonomics", Proceedings of the Computer Graphics International (CGI'03).

Kang H., Jalil A., and Mailah M., 2004, "A PC-based Driving Simulator Using Virtual Reality Technology", In Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry, Singapore.

Kolasinski E. M., 1995, "Simulator Sickness in Virtual Environments", Technical Report 1027, U.S. Army Research Institute.

Lee W., Kim J. and Cho J., 1998, "A Driving Simulator as a Virtual Reality Tool", Proceedings of the 1998 IEEE International Conference on Robotics & Automation Leuven, Belgium.

Levine, O. H. and Mourant, R. R., 1995, "A driving simulator based on virtual environments technology", Washington, DC: Transportation Research Board.

Liao D., 2006, “A Real-time High-fidelity Driving Simulator System Based on PC Clusters”, Proceedings of the 11th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'06).

Luebke, D., Slide Show, Level of Detail & Visibility:A Brief Overview, University of Virginia.

Luebke D.,Reddy M., Cohen J, Varshney A, Watson B., and Huebner R., 2000, “Level of Detail for 3D Graphics”, Morgan Kaufmann, USA.

Narayanasamy V., Wong K. W., Fung C. C. and Rai S., 2006, “Distinguishing Games and Simulation Games from Simulators”, ACM Computers in Entertainment, Vol. 4, No. 2, April 2006. ART 6A.

Olsen E. C. B., 1996, “Evaluating Driver Performance on The Road and In a Simulator”, Thesis-degree Master of Science, faculty of the human factors/ergonomics program, San Jose State University.

Paunoo, B., 2006, “Dynamic Scene Occlusion Culling in Architectural Scenes Based on Dynamic Bounding Volume”, Thesis : Degree of Master of Science (Computer Science).

Reddy, M., 1999, “Specification and Evaluation of Level of Detail Selection Criteria”, Virtual Reality: Research, Development and Application, 3, 2, pp. 132-143.

Rubin, S. and Whitted, T, 1980, “A 3-dimensional representation for fast renderingof complex scenes.” Computer Graphics (Proceedings of SIGGRAPH 80), 14(3): pp. 110–116.

Samuels S. H., 2002, “Driving Simulator”, DISCOVER Magazine, <http://discovermagazine.com/2002/apr/featdrive>, Date Access: 6/17/2008.

Sun C., Xie F., Feng X., Zhang M., and Pan Z., 2007, “A Training Oriented Driving Simulator”, IFIP International Federation for Information Processing 2007.

Sutherland, I.A., 1968, "Head-Mounted Three-Dimensional Display". Fall Joint Computer Conference, AFIFS Conference Proceedings, Vol 33, pp. 757-64.

Young V, 2004, "Programming a Multiplayer FPS in DirectX", Game Development Series, pp. 322.

Yoshimoto K. and Suetomi T., 2008, "The History of research and development of driving simulators in Japan", Journal of mechanical systems for transportation and logistics, vol. 1, No. 2.

Zerbst S. and Duvel O., 2004, "3D game engine programming", Thomson Course Technology.