Paper Title:
An image generation sub-system for a realistic driving simulator

Authors:

J. Miguel Leitão
INESC Porto, Instituto Superior de Engenharia do Porto
jml@isep.ipp.pt

A. Augusto Sousa
INESC Porto, Faculdade de Engenharia da Universidade do Porto
aas@fe.up.pt

F. Nunes Ferreira
Faculdade de Engenharia da Universidade do Porto
fnf@fe.up.pt

In:
Proceedings of CISST’99 - International Conference on Imaging Science, Systems, and Technology
June 28 - July 1, 1999
Las Vegas, Nevada, USA
An image generation sub-system for a realistic driving simulator

J. Miguel Leitão
ISEP / INESC
Instituto Superior de Engenharia do Porto
Rua São Tomé, 4200 Porto, Portugal

A. Augusto Sousa
INESC / FEUP
Instituto de Engenharia de Sistemas e Computadores do Porto
Praça da República 92, 4000 Porto, Portugal

F. Nunes Ferreira
FEUP
Faculdade de Engenharia da Universidade do Porto
Rua dos Bragas, 4000 Porto, Portugal

Abstract
This paper presents a description of the image generation sub-system developed to allow the presentation of realistic visual feedback in interactive visual simulation with large scene databases. The developed image generator applies all the standard state-of-art image generation algorithms aimed to real-time interactive simulation. Some of these important algorithms are also explained in this document. In addition, some innovative optimization techniques like the hierarchical back face rejection of objects, the visibility preprocessing and the automatic optimization of levels-of-detail are being developed and detailed in this paper. These techniques will allow a better use of any image generation system and improve significantly the visualization of huge scene databases even in high-end graphics architectures.

Keywords: Visual simulation, Image generation, Visualization

1 Introduction
A realistic driving simulator is a complex and expensive system. In order to be convincing and valid, it’s necessary to give reliable sensorial information to the driver. Among the several kinds of sensorial information that must be supplied to the driver, the visual feedback is the one that most affect the resulting overall realism, due to the dominant role of vision in human perception.

In addition, because of the cost of high-quality image generation systems and image presentation devices, the image generation sub-system has usually an important influence in the final cost of any realistic interactive simulator.

With in the development of DriS, a driving simulator of a car aimed to the study of driver’s behavior, we are trying to cut down the relation between performance and cost. We are developing some innovative optimization techniques, in order to obtain a faster and reliable screen update. Some of those techniques like hierarchical culling of back-facing polygons, hierarchical visibility preprocessing, and automatic model simplification, can dramatically improve the rendering time of huge scene databases even in high-end graphics architectures.

This paper provides a short description of the architecture of DriS and a detailed presentation of its image generation subsystem. All the innovative optimization techniques used in
This subsystem are also described together with the expected benefits.

2 Related Work

Visual simulators built in the 1970s relied upon rigid models and closed-circuit video cameras to generate the images presented to the user. Only in the 1980s, visual simulators started to use real time image syntheses, based on the digital description of three-dimensional scenes. At the beginning, these systems only allowed a very small level of detail, which was inferior to the one achieved with the direct shooting the rigid model. However, the virtual nature of the 3D digital model quickly revealed the benefits of computer-generated images. The use of digital models granted the introduction of moving objects in the scene, the supervision of weather effects and lightning. The major level of interaction achieved by the synthetic images allowed the use of visual simulators to areas that required a great interaction, like the driving simulation of terrestrial vehicles.

Urban models have, in general, very high geometric complexity, typically requiring very large datasets. Even with only a crude geometric representation of buildings, the model for a small area of a city will often consist of a very high number of polygons that far exceeds the real-time capabilities of even current top range computers. Important works studied the requirements of image quality to driving simulation uses and how the lack of image realism can affect the driver’s performance [14] [15] [16].

Since the early stages of computer graphics, some algorithms for accelerating visualization of large architectural models have been developed. These algorithms are usually based on database partition and culling, on level-of-detail approaches, and on image-based rendering [1] [12].

Sckumaker [9] and Newell [10] were among the first researchers to employ polygon clustering. Coherence has also been the key to many visibility algorithms. In his classic paper, Sutherland [12] showed how visibility algorithms can take advantage of coherence and identified eight different kinds of coherence.

3 DriS - Driving Simulator

A description of our driving simulator (DriS) can be found in [13]. Its main core runs on a SGI Onyx Reality Engine 2 graphical workstation (figure 1). This workstation holds the scene database and performs the simulation and the image synthesis tasks. It is connected to the sound PC and to the cockpit PC by local area network. Communications are implemented synchronously with the rendering process, using a socket library.

In DriS, the driver sees a realistic image either in a large screen display or in a HMD. The generated image is applied to a large flat screen (3.40mx2.00m) by a video projector. Typical experiments are usually performed with resolution of 1280x1024, with an image updating of 20 images per second, and a frame rate of 60 frames per second.

High-quality audio information is also supplied in either stereo sound with headphones or in multi-channel surround sound using loudspeakers, and an additional channel for the typical sound of the vehicle [3].

DriS is being developed by a large team of researchers from several institutions and different scientific areas. The development team includes experts in computer systems, image analysis, computer graphics, road and traffic engineering and psychology. Their main goals are the study of driver’s behavior and road analyses under conditions that are difficult or even impossible to reproduce:

- risk situations
- accident
- new roads
- new managing systems
- new road signs
- new vehicle models

DriS was developed using Genes (figure 2). Genes is a house made Generic Environment Simulator, still in development. It is made on top of Performer, a well-known visual simulation software from Silicon Graphics.

Genes is a house made Generic Environment Simulator, still in development. It is made on top of Performer, a well-known visual simulation software from Silicon Graphics.

DriS also includes autonomous vehicles to simulate traffic. Autonomous vehicles are driven by virtual drivers that try to emulate human drivers, as exact as possible. Although their autonomy, they can be controlled and directed using multi-level scripting [18].

3.1 Applications of DriS

In [5], Noriega and others presented a study of vehicle's motion detection with concurrent self motion with three kinds of road pavement:
- concrete pavement
- bituminous pavement
- bituminous pavement with chromatic bands

The DriS driving simulator allowed the simulation and an easy preparation of the stimulus presented to each one of the 106 persons with driving license that participated in this study. It was concluded that the pavement with chromatic bands produces a higher number of wrong detections of vehicle's motion, while differences in image contrast were considered not relevant.

A new study is now running to evaluate the influence of external publicity panels (outdoors) in driving performance and several others are being prepared [2][17].

4 Image synthesis

Because of the dominant role of vision in human affairs, visual stimuli are undoubtedly, the most important component in the creation of the computer-based illusion that users are in a virtual environment. The visual stimuli can give the user the perception of being enclosed and the sensation of depth.

The image synthesis task is performed by the SGI Onyx Reality Engine 2 graphical workstation. The generated image is applied to a large flat screen (3.40 x 2.00 m) by a video projector. Typical experiments are usually simulated with resolution of 1280x1024, with an image update of 20 images per second, and a frame rate of 60 frames per second.

Rendering an extremely complex geometric database has always been a challenge for visibility computations. To handle such large data sets, some kinds of visibility approaches have been used along with Z-buffering.

4.1 Database partition and culling

The object database is usually hierarchical organized in a tree. This tree can be inherited from the modeling process or it can be created by scene optimizers that group together polygons or objects that are found close to each other. Many algorithms take advantage of this hierarchical organization of scene database [7].

The culling to the viewing frustum is one of those common techniques. It consists of a hierarchical test of inclusion in the viewing pyramid. Within this technique, enclosing boxes of all objects are hierarchically tested against the viewing pyramid and only those that are found inside, are then sent to the graphics pipeline. The viewing pyramid is defined as the one whose vertex lies in viewing point and that contains the screen boundaries. In order to simplify the culling process and so speed up the image rendering, it is common practice to eliminate the culling for small branches of the scene database.

Culling away objects based only on the distance to the viewing point (also called depth clipping) is not a very successful idea as it may eliminate large objects in the background.
4.2 Level-of-detail

Complex 3-D objects easily overload even hardware-assisted rendering and compromise interactive frame rates needed for visual simulation applications. To overcome this restriction, objects (usually defined as a polygonal representation) are stored in multiple levels of detail with progressively decreasing polygon counts. At runtime, an appropriate level of detail is selected to obtain the maximum image quality, avoiding the rendering pipeline overload. This selection is usually performed based on the distance to the viewing point and some stress parameters. This can give an important improvement in the drawing time but it requires that several representations are available to each object.

These representations can be created in the modeling process; however, manual creation of levels of detail for complex models is a too intensive labor. Therefore a tool is needed to generate LODs (levels of detail) automatically. Some approaches are known that try to use images to simulate object geometry [4].

4.3 Billboards

A Billboard is a special object that rotates itself to follow the viewing direction. Billboards are useful for complex objects that are roughly symmetrical about one or more axes. The billboard tracks the viewer by rotating about an axis or a point to present the same image to the viewer using fewer polygons than a solid model. A classic example of a billboard is a single textured quadrilateral representing a tree.

5 Improved culling

Performer provides a fast implementation of the traditional culling to the viewing frustum. This technique uses spatial data structs such as oct-trees and hierarchical traversals of such structs to cull out portions of the model not lying in the current view volume. We are working in a new approach of the culling algorithm. Our new culling will provide a better selection of viewable objects, by rejecting occluded objects based on the occlusion information assigned in a visibility pre-processing. It will also be able to reject groups of back facing polygons due to a better organization of the scene database, using information about polygons orientation.

5.1 Occlusion pre-processing

Some occlusion culling techniques were presented, based on partitioning the model into cells and portals, and computing the partial visibility set of polygons from each cell [11]. A hierarchical Zbuffer algorithm combining spatial and temporal coherence with hierarchical structures was presented by Greene [8].

With our algorithm, still in development, occlusion will be tested against big polygons (occluders) that can be defined dynamically or by preprocessing the scene geometry. Figure 3 shows an example scene with two main occluders (O₁ and O₂) and one object (Obj). During the preprocessing phase, each occlusion zone (Z₁ and Z₂) is determined and associated with the related object (Obj). The union of these occlusion zones defines the space from where the object can not be seen.

![Figure 3: Occlusion zones.](image)

To limit the amount of memory needed and the processing required, we are using some simplifications:

- Objects are treated only by its bounding box and occluders are defined as polygons. This allows the specification of occlusion zones as sets of pyramids parts.
- Only significant occlusion zones are stored. Small occlusion zones are discarded.
- Occlusion information is assigned only to large objects or groups.
- Only few large occluders are considered.

5.2 Back-facing culling

Rejecting back facing polygons is a particular form of occlusion culling used on solid modeling that can be easily combined with other visibility culling methods. This technique is very
common, and is usually applied at polygon level. At this level, it is very simple to implement, but it doesn’t produce a very impressive acceleration on fast graphics pipelines, because the rendering time for a single polygon in one of those systems is also very short. A hierarchical approach of back-facing culling was recently presented by Kumar [6].

The algorithm being developed performs a hierarchical back-facing test, allowing big speedups that can achieve near 50% of the total drawing time, for large datasets. This implies that the scene has an improved organization, grouping together polygons that are close to each other and that have similar facing directions. So, we always must perform an classification in a multi dimensional space.

In our approach, the scene is hierarchical organized in groups of objects. Each group is associated with a bonding box and with four bounding angles. The bounding box represents the surrounding axis oriented volume where descendents objects can be found. The four angles \((h_{\text{min}}, h_{\text{max}}, p_{\text{min}}, p_{\text{max}})\) represent the boundaries of orientation (heading and pitch) of all the descendents polygons. Figure 4 presents the definition of the heading and the pitch angles for the direction \(n\).

The heading angle \(\theta_h\) is the rotation along the \(z\) axis measured from the \(x\)-\(z\) plan and can take any value between \(-180^\circ\) and \(180^\circ\). The pitch angle \(\theta_p\) is the elevation measured from the \(x\)-\(z\) plan and can take any value between \(-90^\circ\) and \(90^\circ\).

Groups are made using polygons that are considered to be near to each other in 3D space and that have similar heading and pitch angles. This implies a 5 dimensional organization that seems to be a very hard job, but for terrestrial based simulations some simplifications can be done. In these simulations, it is sometimes possible to consider only the heading angle in the grouping process. This allows the use of only 4 dimensions.

One enclosing box and two angles intervals are computed to each new group. The bounding box is smallest box that includes all the original objects. The heading interval is defined as the angle space between the smallest \((h_{\text{min}})\) and the largest \((h_{\text{max}})\) heading angles (figure 5).
5.3 Culling algorithm

In real-time, during the culling phase, the scene structure is parsed as usually. It is now possible to reject objects or groups that are not in the viewable zone, objects that are facing back to the camera and big objects that are occluded behind any considered occluder.

For each group, the intersection between the facing directions and the viewable angle is performed. If this intersection is null then the entire group and its descendents can be eliminated from the drawing phase. If the entire group is found included in both the viewable volume and the viewable directions, and it is not considered as occluded, it must be considered all together for the next phase. When only a part of the group is found in the viewable area or directions, and it is not considered as occluded, the same culling process is recursively performed to each descendent. Table 1 summarizes the decision taken by the culling process depending on the direction test and on the position test.

<table>
<thead>
<tr>
<th>Position</th>
<th>Direction test</th>
<th>Position</th>
<th>Direction test</th>
<th>Position</th>
<th>Direction test</th>
</tr>
</thead>
<tbody>
<tr>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>divide</td>
<td>reject</td>
<td>divide</td>
</tr>
<tr>
<td>divide</td>
<td>divide</td>
<td>accept</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6 Level-of-detail optimization

We are developing an automatic tool to optimize level of detail of objects in complex scenes.

The optimization of level of detail takes place immediately after loading the scene database, and before the simulation starts. This automatic task consists in creating, for each object definition, several geometric representations with different levels of detail.

The object database is usually hierarchical organized in a tree. In order to optimize level of detail, the tree is traversed in a depth first way. For each visited node, an optimization can take place.

If the node is a group of objects (figure 6), it is switched with a level-of-detail node. Its descendents are sorted by its largest dimensions and several sets of descendents are generated, discarding successively the smallest parts.

![Figure 6: Group of 3 objects.](image)

The result is presented in figure 7. The original group of 3 objects (figure 6) became a LOD node with 4 sons. The first is the complete original group. The second is the same but without Obj3. The last one will always be a null object.

![Figure 7: LOD generated for a group.](image)

If the node is a real geometric definition node, it is first switched with a level of detail node with at least two sons:
- The first son is cloned from the original definition.
- The last son will be a null object.
- Some other sons will be defined from generated object impostors (figure 8).
Switching distances must also be defined using the size of original object and the expected image quality and resolution.

The original definition of the object being optimized is then recursively optimized.

Finally, if any descendent became a level of detail node, it is merged with the object being optimized.

7 Conclusion

This paper presents a short description of DriS, a Driving Simulator of a car, its goals and its applications. The developed image generation subsystem and some state-of-the-art optimization techniques used like hierarchical culling to the viewing frustum and automatic selection of level-of-detail are also discussed.

In this paper, we also present our recent work in developing a new approach of the culling algorithm. Our new culling will provide a better selection of viewable objects, by rejecting occluded objects based on the occlusion information assigned in a visibility pre-processing. It will also be able to reject groups of back facing polygons due to a better organization of the scene database, using information about polygon orientation.

These techniques will dramatically reduce the rendering time of huge scene databases even in high-end graphics architectures, allowing an improvement in visual realism and a reduction in hardware requirements.

8 References

[6] Subodh Kumar; Dinesh Manocha; William Garrett; Ming Lin, Hierarchical Back-Face Computation, 7ª Eurographics Workshop on Rendering, Porto, June 1996