Occlusion Culling for the Visualization of Aeronautical Engines Digital Mock-ups

Aiert Amundarain$^1$, Diego Borro$^1$, Luis Matey$^2$, Alex Garcia Alonso$^3$

$^1$ CEIT (Centro de Estudios e Investigaciones Técnicas de Guipúzcoa)
Lardizábal 15, E-20018 San Sebastián
(34) 943 212800 / (34) 943 213076
E-mail : {aamundarain }@ceit.es

$^2$ University of Navarra
Lardizábal 13, E-20018 San Sebastián
(34) 943 219877 / (34) 943 311442
E-mail : lmatey@tecnun.es

$^3$ University of the Basque Country
Lardizábal 1, E-20018 San Sebastián
(34) 943 015104 / 943 219306
E-mail : agalonso@si.ehu.es

Abstract: We present a visualization module where we have implemented an occlusion culling strategy for aeronautical engine display. The occlusion culling method has been adapted to the shape and characteristics of an engine digital mock-up. Due to the shape of the engines we can identify in which areas of the engine we can find the occluded elements. The occluded areas are identified using an image plane where we project the occluders. We have used a special bounding volume, which fits perfectly to the features of the engine.

This occlusion method has been integrated inside REVIMA. REVIMA is a virtual reality system for maintainability simulation in Aeronautics. REVIMA has designed a haptic system to provide the sensation of working with a physical mock-up. A collision module and a control module control the haptic system. We will analyse the advantages of introducing the occlusion culling method inside REVIMA.

Keywords: virtual reality, real time visualization, maintainability, digital mock-up.

1. Introduction

The maintenance analysis of the engine aircrafts is a very important issue in the field of aeronautics. These analyses are accomplished using physical aircraft engines mock-ups. The expenses of these mock-ups led aeronautical industry to research alternatives to replace them.

Nowadays, it is widely extended the use of electronic mock-ups to design the external parts of the aircraft engines (piping, harnesses and installations). This allows a group of designers to work quasi-concurrently over an assembly, copying and automating the original process. This technology is known in the industry as DMU/DPA (Digital Mock Up / Digital Pre-Assembly). The use of physical mock-up is mandatory in order to evaluate the maintainability of externals during the development stage. Although these mock-ups can be used for other applications, the ultimate purpose of the construction is to check the maintainability.

In order to replace the use of the physical mock-ups, we have developed REVIMA, a haptic system to simulate the maintainability operation in digital mock-ups. Our haptic device gives the user a realistic feedback and provides a workspace similar to the size of a turbo-engine.

The digital mock-ups are created using CAD systems and consist of millions of polygons. Unfortunately such large polygonal environments are difficult to visualize interactively despite recent advances in computer graphics hardware. An interactive visualization is necessary to simulate the maintainability operation in a realistic way; therefore we must develop visualization techniques to achieve the desired frame rate.

To accelerate the visualization times, it is important to reduce the number the polygons that are sent to the graphic hardware. This aim can be achieved following different methods. One of them is culling away the parts of the model not visible from the current viewpoint. From the different culling methods, occlusion culling has been an intense research area and many occlusion techniques have been developed. Some of them are specific to the certain types of virtual environments, such as architectural and urban environments. In this case, the occlusion culling algorithm has been adapted to the features of the virtual engine aircraft mock-ups.
2. General overview of REVIMA

REVIMA is a haptic system developed to check the maintainability of aircraft engines. The system has been created from scratch by CEIT Applied Mechanical Department. The development of the system has required the research in different disciplines: mechanical design, control theory, computer graphics, computational geometry and human-machine interaction.

A haptic device has been designed and built to offer the user force feedback and therefore, simulate the maintainability operation in a realistic way. A control module drives the haptic device response. The force that the haptic device provides depends on the contacts between the different objects of the virtual aircraft engine and the virtual tools. A collision module calculates these contacts and the characteristics of the contacts. A graphic engine displays the virtual environment in a interactive frame rate. Finally, a graphic user interface (GUI) allows working with the system in a friendly way.

The architecture of the system is based in two PC-s. The first one (control PC) executes the control module to command the haptic interface. The second one (simulation PC) runs the main module, the Graphical User Interface (GUI), the collision solver and the graphics engine. The simulation PC has two processors, in one processor runs the collision solver and in the other one runs the graphics engine with the GUI, optimising the response of the different modules. Both PCs are connected through an Ethernet LAN network (see Figure 1).

2.1. Haptic system

The haptic device, which has been designed and built, is an example of nonportable force feedback hardware. In contrast with the prevalent forms of force feedback in use today [1], which are desk-grounded masters, this system is floor grounded. This is due to the large workspace needed for the maintainability application. The basic workspace of the haptic interface occupies a cylindrical sector, which corresponds to a wide work area of a virtual 3D aircraft engine full-scale mock-up.

The dimensions of the basic cylindrical workspace are: internal radius, 242 mm; external radius, 772 mm; depth 1500 mm; angle 120º (see Figure 2).

The system provides force feedback in three translational degrees of freedom. While three additional orientations are measured by a special wrist, but not actuated.

An interesting design feature of the haptic system is that its workspace can be relocated. Therefore, it is possible to reproduce different maintainability operations and check different situations from an ergonomic point of view.

2.2. Collision module

The collision module sends messages to the control module each time that detects a collision among the objects of the virtual scene. Due to this information the PC control drives the haptic response, providing force feedback in case of collision.

The used algorithm is based on uniform spatial grid decomposition: voxels [2]. All the static objects are considered as a unique solid, which is decomposed in voxels [3]. The dynamic objects are defined by their facets. The algorithm uses two levels of accuracy: interferences between dynamic facets and static voxels; and interferences between dynamic and static facets.

Besides the detection problem, the collision solver also computes a collision response. This response consists of...
two values, the normal contact and the penetration between two 3D objects. This is the information that is sent to the control module.

2.3. Control module

This module has two tasks. It acquires the position and orientation of the tracking device, the haptic system, and sends this information to the simulation PC. It must also calculate the force feedback that the haptic device provides, taking account of the information sent by the collision module.

The control loop, located in the control PC, has a sampling period of 1 KHz [4]. Since the control loop runs faster than the collision module, different numerical strategies has been developed to achieve a stiff and a smooth touch. These strategies are based on the interpolation of the values sent by the collision module [5].

3. Visualization module

The visualization module generates images that replace the physical viewing of the mock-up. Our aim was to develop a graphic engine where the virtual aircraft engine (see Figure 3) can be displayed at interactive frame rate.

The virtual aircraft engines have been created in CAD systems. These models are very complex with a great density of faces per element. They have millions of polygons and consist of thousands of elements. There is a remarkable difference among the size of the elements. Some elements have around a hundred polygons while others have more than a hundred thousand polygons.

In spite of the rapid progress in the performance of graphics processing units (GPU), it is not possible to display these models at interactive frame rates. Many acceleration techniques for interactive display of complex datasets have been developed. These include visibility culling [6], object simplification [7] and the use of image-based [8] or sampled representations [9].

Visibility culling deals with the identification of those portions of the scene potentially visible from a dynamic viewpoint. We have focused the research in the development of visibility culling algorithms for virtual aircraft engine models.

3.1. Overview of visibility culling algorithms

An element could be no visible due to three reasons (see Figure 4): it is outside of the viewing volume (view-frustum culling); its normals face away from the view point (backface culling) or it is hidden by other elements (occlusion culling).

The identification of the objects that are outside of the view volume is made using bounding volumes. Analysing the relative position of the bounding volume respect to the view volume it is possible to know if the object is outside.

There are effective methods to identify the polygons that face away the viewpoint [10]. The problem is that only a part of the polygons of the element will face away the viewpoint each frame, and this polygons will be different each time. This feature does not suit appropriately to the most efficient display modes: vertex arrays, display list…However, the current graphic cards offer the option to do the backface culling in hardware.

The occlusion culling technique is global as it involves interrelationship among polygons and is thus far more complex than backface and view-frustum culling. It is usual to estimate a set of objects that includes at least all the visible objects plus maybe some additional invisible ones. This concept is known as conservative visibility. In some applications, it is more important the frame rate than the image quality, and they use a more aggressive technique, without assuring that all visible objects will be in the estimated set of objects [11].

One class of visibility algorithms calculates visibility from a point. Among these algorithms some of them do their analysis in the image space and others in the object space. Image-space methods render occluders and thus generate an occlusion map or a hierarchy of maps. Scene objects, organized in a special data structure, are tested against the maps. Image based methods become popular in practice because of their robustness and ability to make use of the
graphic hardware [12][13]. The object space methods calculate the visibility in the space that the scene is defined. In these methods, scene objects are geometrically tested against the shadow volumes generated by the selected occluders [14][15].

A second class of visibility algorithms splits the view space into regions (typically called cells) and precomputes a potentially visible set of objects for each region. This information is stored in disk and retrieved on demand during an interactive walkthrough. These algorithms are suited to work in architectonical scenes [16].

3.2. **Description of the problem**

It is essential to know the characteristics of the digital aircraft engine models to develop suitable visibility culling techniques. We present them in the following paragraphs.

All the objects that are part of the aircraft engine can take part in assembly operation; therefore any object can be released and moved at any moment. However, only a few objects can be moving at the same time.

The visibility state of every object is a parameter. In run-time any object can become invisible and invisible objects can recover their visibility.

It is also important to realize an important feature: the shape of the virtual aircraft engine models is cylindrical. The largest objects, the objects with more polygons, also have a cylindrical shape. The cylindrical objects are around 2% of the total objects, but they use to have around 25% of the polygons of the scene.

A navigation system has been developed to make easier to accomplish the simulation operations. If the user is working with the haptic device, it is very difficult to control at the same time the virtual camera. Therefore, we place the camera on a cylindrical surface, looking at the center of the engine all the time and also pointing toward the place the simulation operation is happening (see Figure 5). The movement of the virtual tool guides the camera.

It is important to know the available hardware. In the description of REVIMA, we have mentioned that the visualization module will run in a computer with two processors. However, it uses one processor, because the collision module employs the other. The graphic hardware is a common commercial graphic card.

All this properties must be taken into account to develop the most efficient visibility culling techniques.

3.3. **View Frustum Culling**

Usually, the scene geometry has been stored in hierarchical structures to optimize the view frustum culling. The view frustum tests began in the upper nodes of the structure, and finished in the lower nodes if it was necessary [17]. This way the view frustum tests were reduced, decreasing the computational time. However, these tests are simple and the time spent on them is not considerable. Although we were able to reduce the number of tests; the time saved was not noticeable. So we had to explore other approaches.

The view frustum culling tests are made between the viewing volume and the bounding volume of the nodes. The selection of an appropriate bounding volume is very important [18]. A very common bounding volume is the sphere. The tests are simpler when this bounding volume is used. However, this volume does not fit efficiently to the shape of the objects. Due to this feature, some objects that are outside of the viewing volume are not culled away, because part of their bounding volume is inside the viewing volume.

There are bounding volumes that fit perfectly to the shape of the objects, but they used to be too complex, increasing the computational burden. The objective is to find a bounding volume that fits to the shape of the objects of our model but without increasing its complexity too much. Taking account of the cylindrical shape of our model, we have selected a bounding volume based in a cylinder. We project the object along the main axis and we calculate a bidimensional convex hull that fits more accurately with the shape of the objects. Then, our bounding volume is created extruding that polygon along the main axis (see Figure 6).
Fig 6 Front and lateral view of an objects with their bounding volume based in bidimensional convex hull

Generally this bounding volume adjusts more exactly to the objects. This characteristic is more important in cylindrical objects, where the bidimensional convex hull comprises at least four times less volume than a sphere. Therefore, with the bidimensional convex hull, around 30% more objects are culled away, reaching in some situations the 70%.

3.4. Occlusion Culling

It is important to develop an effective occlusion culling algorithm. The geometry of our models is too complex to make the computations in object space. The algorithm will be based in HOM (hierarchical occlusion maps) [12]. This method is generic, but due to certain properties of our system, it can lose its effectiveness. The time spent in computing the hidden objects with this method is not negligible, then to keep its effectiveness, either the computations must be parallelized or the number of identified hidden objects must be important.

In our system, we have available only one processor; therefore it is impossible to parallelize the process. The number of hidden objects uses to be around the half of the objects of the scene. Therefore, we cannot invert too much time identifying the occluded objects, if we want to keep the effectiveness of our method.

The first step of our algorithm is to identify the occluders. The occluders are the objects that have the best features to occlude other objects. To identify the occluders we have split the scene in cylindrical sectors (see Figure 8). We have chosen this way to split the scene taking account of the shape of our models and the navigation system. Then in a pre-process step we have selected the most effective occluders for each sector.

In runtime, we must identify the sector where the viewpoint is. Then the occluders of that sector are rendered in an off-screen buffer. We are interested in knowing which areas of the images are occupied by the projection of the occluders. So, the occluders are rendered in the simplest way, without any kind of lighting. Then this projected area information must be retrieved from the graphic hardware. This operation is quite time consuming then, instead of projecting the occluders with the final image resolution, it is projected with lower resolution. The selected resolution is 256x256. From this image, lower resolution images are obtained applying the average operator to rectangular blocks of pixels, obtaining hierarchical occlusion maps (see Figure 7).

Fig 7 Hierarchical occlusion maps

Fig 8 The scene divided in cylindrical sectors
It is important to know how many occluders are necessary to obtain correct occlusion maps. If we use too many occluders the occlusion maps are more accurate, however we spend too much time rendering them. The optimum number of occluders has been established in 50 (see Figure 9).

![Fig 9 Frame rate vs. Nº of occluders](image)

The steps that we have followed until this point are very similar to the HOM algorithm. An object is occluded if the occlusion maps cover its projection and if it is behind of the objects that generated the occlusion map. Then, to be able to classify an object as occluded, we need the depth information of the occlusion maps. To obtain an accurate depth information is very time consuming, because we must read again information from the graphic hardware. We must also consider the shape of our occluders. Most objects that are good candidates to act as occluders have a tubular shape, so they contain in their interior many objects. This situation creates a cyclic overlap that must be broken in order to make useful the occlusion algorithm.

This problem is solved defining a plane, we call crossing plane (see Figure 10). This plane is perpendicular to the view direction and crosses the aircraft engine. When we project the occluders to calculate the occlusion maps, instead of using the view frustum that we will use to render the scene, we use a similar view frustum but the crossing plane defines the back plane. This way the occlusion map will be generated only for that part of the occluders that is in front of the crossing plane.

![Fig 10 Crossing plane](image)

In this situation an object is considered as occluded if the projection map covers its projection and if it is behind the crossing plane, saving the step of computing the depth information. To check if an object is behind the crossing plane, the bounding volume of the object is tested against the crossing plane. Well-adjusted bounding volumes improve performance and ours culls a 15% more objects than the sphere, and this generates a 5% increase in the overall frame rate.

It is important to find the best position to place the crossing plane. The plane, in first instance, was placed crossing the axis of the aircraft engine. The results obtained in this location were good. However, as the scene is rendered in perspective projection, the characteristics of this projection allow placing the crossing plane nearer to the viewpoint. In this way, the occlusion maps created will have less quality because a smaller part of the occluders is in front of the crossing plane, but at the same time more objects will be behind the plane. There must be an optimum distance to place the plane. In the Figure 11, we can observe how the frame rate increases when we decrease the distance, using different number of occluders in the test. The optimum distance is established in 0.34*R from the main axis, considering R the radio of the engine.

![Fig 11 Distance vs. frame rate](image)

Although we have developed this technique for some particular conditions, this method can be extended to more general scenes. It is important to be able to separate most visible objects from the most no visible objects by a few planes.

3.5. Analysis of results

The results that we present here are a mean value of the results calculated for different positions of the viewpoint. We have obtained results placing the viewpoint in all the cylindrical sectors.

To know exactly the level of effectiveness of our visibility culling algorithms it is important to know how many hidden
objects are identified from all hidden objects. We have calculated that around 47% of the polygons of the model are visible. Then the aim is to identify almost all the 53% hidden polygons, using our visibility algorithms.

Using only the view frustum technique, 16.5% of the polygons are identified. This is a mean value, in certain viewpoints there will not be polygons outside the view volume, and in other positions the percentage will be bigger. In a second step our occlusion method identifies a further 21% of the polygons as occluded. The result is that a 37.5% of the polygons are culled.

However, if instead of considering all the polygons in the scene, we consider only those that are no visible, our method finds a 72% of them. Even more, as we have already shown the low computer power consumed by our algorithm makes profitable their use: the overall display frame rate is always improved.

Before applying any technique our frame rate was around 7 fps in a GeForce3 graphic card, when we displayed a model with around 2 million polygons. After applying both techniques we achieve around 10.5 fps. The increase is around 50 % in the frame rate.

4. Conclusions

The main conclusion of this work is that virtual reality systems with force feedback based on low cost hardware can be directly applied to the industry and that these systems may lead to important reductions in maintainability costs. Nowadays, there are two completely developed systems that have been validated and are being used by ITP (Industria de turbopropulsores). These systems have been useful to evaluate satisfactorily several maintainability cases when elements of the engine have been developed.

The visualization module has achieved to represent the virtual aircraft engine models at an interactive frame rate. This objective has been achieved using the most optimized functions that the graphic cards can offer us and developing visibility culling techniques.

We have improved view frustum culling using a new bounding box and we propose a new effective algorithm for occlusion culling

The visualization module has been integrated with a haptic system whose workspace has the dimensions of an aircraft engine. This characteristic allows working with a virtual aircraft engine in 1:1 scale, and make ergonomic studies.

5. Acknowledgment

The authors would like to thank ITP and Sener for promoting and funding the application REVIMA. This application has been partially financed by the project of the Basque Government, number CI01TP03.

We would like to specially thank Iker Aguinaga, Jorge Juan Gil, Joan Savall, Emilio Sanchez, Jaime Rubi and Angel Rubio for their support and ideas in the development of this work.

6. References