SP-Octrees Visualization Using Impostors

F.J. Melero, P. Cano and J.C.Torres

Dpto. Lenguajes y Sistemas Informáticos, Universidad de Granada, Spain
{fjmelero, pcano, jctorres}@ugr.es

Abstract

SP-Octrees (Space-Partition Octrees) allow a multiresolution representation of polyhedral objects, and have been successfully used in progressive transmission. Nonetheless, the SP-Octree is a solid modeling structure, intended to solve problems as solid intersections and point inclusion tests. At intermediate levels of the SP-Octree, the visualization is a rough approximation of the object due to the fact that grey nodes are shown as the convex hull of the part of the solid included in that node.

In this paper we present an approach to use SP-Octrees for adaptative visualization, that improves the visualization of these multiresolution models by applying impostors over those planes of the hierarchical structure that belongs to the convex hull but are not part of the solid boundary (ficticious planes). By doing so, it is possible to obtain a better approximating visualization of the object, although the maximum LOD is not used. At every moment, the impostor is selected depending on the viewpoint.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

The interactive visualization of three-dimensional models is traditionally one of the most important applications of Computer Graphics. It is used in medicine, architecture, archaeology, etc... The interactive visualization of complex models requires a great effort of the graphics hardware, due to the large number of polygons needed to represent the geometry at the expected level of detail and realism. This resource requirement may lead to a lack of interactivity.

The conflict between the rendered level of detail and the visualization speed has motivated the development of several techniques to reach both goals. Under the generic name of level-of-detail (LOD) techniques we gather techniques that represent a complex geometry in a simplified way when the observer is not close enough to the object.

1.1. Geometry-based LOD Techniques

Geometry-based LOD techniques can be classified as: *discrete* [Cla76], i.e. the object is represented in several instances, each one at different LOD; *progressive* [Hop96], i.e. the detail is extracted from an unique data structure during runtime; and *view-dependent* [Hop97, XV96], which are an extension of progressive techniques, in a way that the LOD is not uniform along the object, it is *anisotropic* depending on the point of view.

Although the LOD concept is not restricted to a concrete area (images, solids, volumes...), in three-dimensional computer graphics it is usually applied to the simplification of polygonal meshes, mainly triangular meshes.

Another approach to represent volumes and solids with variable LOD is to represent them with schemes based on spatial decomposition, using hierarchical structures. Among these methods we can find the *Octree* and several extensions to these (Extended-Octrees, PM-Octrees, etc.).

1.2. Image-based LOD techniques

Another approach to solve the conflict between realism and interactivity, is the image-based techniques. The idea underlying this approach is to replace a complex three-dimensional geometry with a 2D image that shows what should be represented with the removed geometry. This image is an *impostor*.

Many image-based techniques have been presented to represent solids [Mac95, Ali96, SDB97, DSSD99], and all of them can be considered as view-dependent. The complexity of the geometric model where impostors are applied moves from a single plane to triangular meshes.

2. SP-Octrees

The classic Octree only allows an approximate representation of the object, so in order to obtain a better (but never exact) representation, it is necessary to increase the depth of the tree, and therefore to increase the storage cost.

To solve this inconvenience, several extensions to the Octree scheme have been proposed.

The *SP-Octrees* (Space-Partition Octrees) [CT02], include in the internal nodes information about the boundary that defines partially the represented object at that level. When descending the tree, whenever a plane is found at an upper level node, it is not necessary to repeat the data of these planes.

2.1. Nodes

By using the same octal-tree structure as in classical Octrees and extended Octrees, we represent the boundary of the solid in the internal and end nodes.

Basically, the idea is to classify each node once depending on the characteristics (convexity or concavity) of the planes that appear in it. Also, the internal nodes store those planes that do not change in the children nodes, so it is not necessary to repeat the information. This information allows to delimit the space where the solid is defined inside the node.

SP-Octrees manage with six different nodes:

- *black* node (fig. 1), when the node is completely inside the solid.
- *white* node (fig. 1), when the node is completely outside the solid.
- *convex* node, if the intersection between the solid and the node is convex. In this case, the node contains the set of planes from the boundary that defines the convexity, except those that have been used at previous levels of the tree.

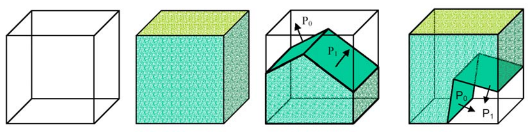

Figure 1: *White, black, convex and concave node.*

- *concave* node (fig. 1) if the intersection between the solid and the node is concave. In this case, the node contains the set of planes that defines the concavity.
- *grey* nodes (fig. 2) contain concavities and convexities, and must be divided into eight equal octants. This node is represented by planes from the convex part of the boundary, filling the holes with the node bounding box.

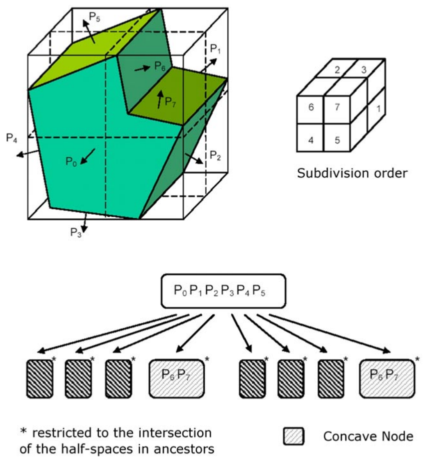

Figure 2: *Grey node, subdivision order and tree.*

- *vertex* nodes are used when a node contains an unique vertex where converge convex and concave edges (fig. 3). Vertex nodes are needed because it is not enough with categories below to cover all cases [CT02].

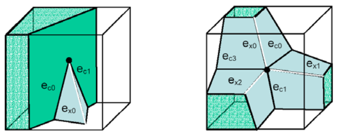

Figure 3: *Two vertex nodes*

With this set of nodes, we keep the same expressivity as in Extended Octrees [BN90], as the end nodes used in that approach can be directly translated into our scheme.

3. SP-Octrees Visualization

In order to visualize an object represented by means of the proposed scheme, we traverse the tree level by level, representing at each node the intersection of the planes that appear in it with its surrounding box and with the planes that appear in their ancestors. In this way, as we have information of the boundary of the object in the upper nodes, the higher levels of the tree allow us to obtain quickly the convex part of the boundary of the object. This mechanism allows us to make an adaptative visualization.

In order to draw the object faces it is necessary to trim the planes in one node against those in its descendants. We can do this while drawing or we can modify the data structure to store in each node of the tree, not only the definition of the planes, but also the geometry of the faces of the solid. We can obtain this easily using a secondary B-Rep scheme to accelerate the process.

3.1. Applying *Impostors*

Projective texture mapping was proposed by Segal [SKv*92], and is part of the OpenGL standard [SA]. Although in the original paper was used only to accelerate the shadow and lighting generation, it is possible to apply directly *Projective Texture Mapping* to image-based rendering, because it can simulate the inverse process of taking photographs with a camera, allowing to project images over the scene like a light projector.

To project the texture, the user specifies the projector position and orientation, and a virtual projection plane where the image is projected. With these data, and the model transformation and projection matrices of the scene, we can calculate the right texture coordinates [Eve, SA]

3.2. Impostors Generation

The set of images used as impostors is generated once in a preprocessing time. The images are taken from the vertexes of a sphere that contains the solid bounding box. These images are taken by rendering the real solid at the maximum LOD. By varying the LOD of the sphere, it is possible to control the amount of images or impostors to be used.

Another approach is to determine the visibility points for each plane stored in the SP-Octree, and take a set of images for each plane from different view points. We think that such approach needs a higher amount of memory, as well as an increase of rendering time, because each plane has its own texture and the texture switching spends CPU time.

Having done several tests at different sphere resolution, and checking the quality of transitions between impostors and the subjective appearance of interactive visualization of the three-dimensional model, we think that having 258 images (fig. 4) it could be enough in order to obtain a right appearance of the SP-Octree, specially when the solid is far enough to not have to descend in the hierarchy and it is seen quite small.

We position the camera at each vertex of the sphere, oriented towards the centre. By using the orthographic projection, we only have to control that the object fits entirely in the projection plane. All the images are stored in a binary file, as well as information about all the OpenGL matrices used when the image was taken. The object was rendered with a controlled background, so that transparent pixels can be easily recognized.

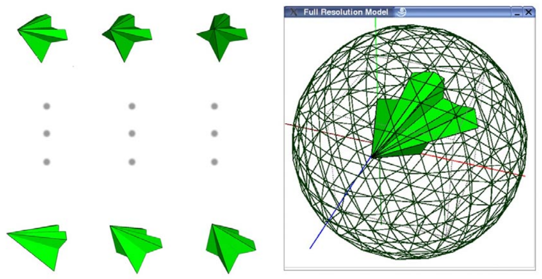

Figure 4: *Sphere with 258 vertexes, used to take the impostors.*

3.3. Impostors Application

To render the model, the tree is traversed in a descendent way, generating the corresponding polygons for each level. If the generated polygons do not belong to the solid boundary, we apply the selected impostor, otherwise, if it is a boundary polygon, we apply its original texture.

Since we store each image modelview matrix, it is possible to know where it was taken from, and to select as impostor the one that best fits to the view seen by the observer.

1) Before rendering the scene, we check the observer position relative to the object. This relative position determines the vector that will be used to locate the image from the set of impostors. Note that the position is not based on the lookAt vector, but it is based on the position of the SP-Octree relating to the observer.
2) From the impostors table, we select the one that was taken in a similar position, and take its texture identifier (this impostor will be used for all ficticious planes).
3) The tree is traversed in preorder until the desired level, generating those polygons that are seen at each node.
4) The boundary polygons at the original model are rendered with its original texture or assigned colour. But, if we are on a non-boundary plane of a grey node, we apply the impostor previously selected.

4. Results

SP-Octree is a very useful approach to make a progressive transmission [CTV03] and visualization of the solid, saving space with respect to other hierarchical structures.

This paper adds to this approach an improvement of visualization capabilities, being able to recognize the model from the first steps of visualization. As the observer gets closer to the object, the tree refines the nearest nodes to the camera, leaving at low resolution those that are not being seen.

In figure 5 we can see as the proper impostor is selected at each moment, and as from the first level of the SP-Octree, the user perceives the original solid.

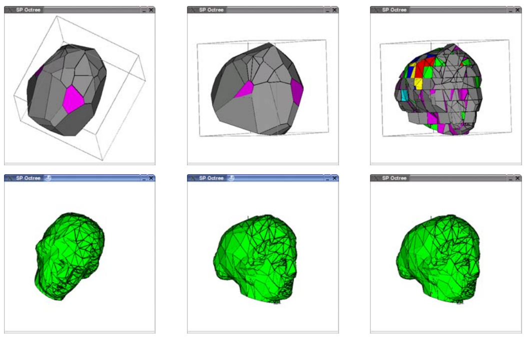

Figure 5: *SP-Octree visualization without impostors (upper) and using impostors (lower).*

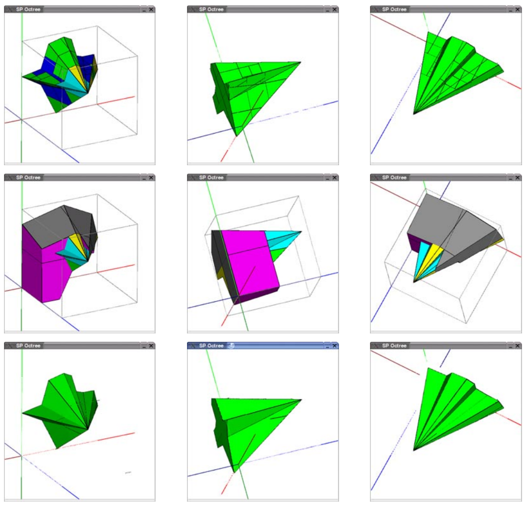

Figure 6: *SP-Octree at maximum LOD (upper), without impostors at level 1 (centre) and with impostors at level 1 (lower)*

In sequence 6 we can appreciate as from several points of view, the right impostor is selected, being the result very similar to the last level of the SP-Octree.

5. Conclusions and Future Work

The use of impostors in SP-Octrees allows to use this hierarchical structure for adaptative visualization.

We will investigate its suitability for interactive rendering of complex environment.

6. Acknowledgments

This work was partly supported by the Spanish Ministry of Science and Technology (MCYT) under project TIC2001-2099-C03-02.

References

[Ali96] ALIAGA D.: Visualization of complex models using dynamic texture-based simplification. In *IEEE Visualization'96* (October 1996), IEEE.

[BN90] BRUNET P., NAVAZO I.: Solid representation and operation using extended octrees. *ACM Transactions on Graphics 9*, 2 (1990), 170–197.

[Cla76] CLARK J.: Hierarchical geometric models for visible surface algorithms. In *Communications of the ACM* (1976), vol. 19.

[CT02] CANO P., TORRES J.: Representation of polyhedral objects using sp-octrees. In *Journal of WSCG* (2002), vol. 10, pp. 95–101.

[CTV03] CANO P., TORRES J., VELASCO F.: Progressive transmission of polyhedral solids using a hierarchical representation scheme. In *Journal of WSCG* (2003), pp. 95–101.

[DSSD99] DECORET X., SCHAUFLER G., SILLION F., DORSEY J.: Multi-layered impostors for accelerated rendering. In *Computer Graphics Forum* (199), Brunet P., Scopigno R., (Eds.), vol. 18.

[Eve] EVERITT C.: *Projective Texture Mapping*. http://developer.nvidia.com.

[Hop96] HOPPE H.: Progressive meshes. In *Proceedings of SIGGRAPH 96* (1996).

[Hop97] HOPPE H.: View-dependent refinement of progressive meshes. In *Proceedings of SIGGRAPH 97* (1997), pp. 99–108.

[Mac95] MACIEL P.W.C AN SHIRLEY P.: Visual navigation of large environments using textured clusters. In *Symposium on Interactive 3D Graphics* (1995), Hanrahan P., Winget J., (Eds.), ACM SIGGRAPH.

[SA] SEGAL M., AKELEY K.: *The OpenGL Graphics System: A Specification (Version 1.2.1)*. http://www.opengl.org.

[SDB97] SILLION F., DRETTAKIS G., BODELET B.: Efficient impostor manipulation for real-time visualization of urban scenery. *Computer Graphics Forum 16*, 3 (1997).

[SKv*92] SEGAL M., KOROBKIN C., VAN WIDENFELT R., FORAN J., HAEBERLI P.: Fast shadows and lighting effects using texture mapping. In *Proceedings of SIGGRAPH'92* (1992), pp. 249–252.

[XV96] XIA J., VARSHNEY A.: Dynamic view-dependent simplification for polygonal models. In *Proceedings of IEEE Visualization'96* (San Francisco, Oct 1996), pp. 327–334.