A Hybrid LOD Based Rendering Approach for Dynamic Scenes

Karsten Hilbert
Computer Graphics and Visualization
Chemnitz University of Technology
karsten.hilbert@informatik.tu-chemnitz.de

Guido Brunnett
Computer Graphics and Visualization
Chemnitz University of Technology
brunnett@informatik.tu-chemnitz.de

Abstract

In this paper we present a novel LOD based rendering approach that integrates geometry based and image based LOD mechanisms in one system. We assume that every LOD object in the scene is represented by a textured polygonal mesh. In a preprocess we generate a continuous multiresolution model for all LOD objects. During runtime we first check for every LOD object if an appropriate cached approximation generated for former frames is available. If not, we extract a new suitable approximation out of the objects continuous multiresolution model depending on view-dependent criteria. After that, we replace this adaptive geometry-based approximation by an image-based impostor if this is reasonable and efficient. The selected approximations of LOD objects are used for rendering the next frame.

1. Introduction

1.1. Definitions

Level-of-Detail (LOD) based rendering approaches process a multiresolution model (MRM) either for the whole scene or for each object in the scene. One of the key issues in this context is the proper choice of the approximation which should be less detailed (i.e. contain less polygons) but should look very similar to the original object at least from the current viewpoint. If this can be achieved high quality images can be rendered efficiently with the LOD approach.

For the generation of approximations two different approaches have been suggested.

Geometry-based approximations are represented by polygonal meshes and can have a constant (non-adaptive approximations) or a varying Level-of-Detail (adaptive approximations). Impostors, billboards, sprites with depth and layered depth image (LDI) are examples for image-based approximations. An impostor is a simple primitive that is represented by a few textured quadrilaterals in most of the cases or in special cases by a few simple textured polygonal meshes. An impostor texture only looks reasonable for a small range of object views.

1.2. Previous work

The first proposed LOD based rendering approaches are based on discrete MRMs. Sets of pre-generated impostors have been used as approximations of objects or object groups.

Approaches based on discrete MRMs suffer from so-called popping artifacts. Because of the limited number of usable approximations of an object the used one is not always optimal for the given view configuration. Therefore more current LOD based rendering approaches process a continuous MRM for each single object or the whole scene.

There are two main categories of continuous MRMs. Incremental continuous MRMs contain a much reduced base model and a sequence of refinement operations. But it is hard to extract any adaptive approximation out of this representation. A hierarchical continuous MRM [1] represents a dependency graph of refinement or simplification operations. In this case approximations extracted depending on view-dependent criteria are always optimal for the current viewing configuration. However, these LOD algorithms are time-consuming.

Some image-based rendering approaches suggested in the last decade use dynamically generated impostors [2] or other dynamically generated image-based approximations for objects or object groups. If dynamically generated impostors can be used the number of polygons that have to be rendered is reduced dramatically. Again the high computational cost for

impostor generation based on the original geometry is the main drawback of this approach.

Hierarchical image caches [3] are a very sophisticated application of image-based and geometry-based rendering techniques, but these approaches can not be used to process dynamic scenes and it seems very difficult to parallelize them.

1.3. Our approach

We are interested in using LOD techniques as a necessary ingredient for real time rendering of dynamic scenes in virtual reality (VR) applications. We assume the scene to be given as a set of individual objects represented by textured polygonal meshes of arbitrary topology.

Our approach combines geometry-based and image-based LOD-techniques and thus avoids the disadvantage of existing approaches mentioned in 1.2. In a preprocess we create a vertex hierarchy [1] for each LOD object in scene. Out of a vertex hierarchy adaptive approximations can be extracted based on view-dependent criteria. Our data structure differs in the following respects from the original one suggested by Luebke and Erikson. In addition to the vertex tree and the list of active triangles we assume normal and color information to be given as textures (represented by a series of mip maps). At runtime we compute adapted texture coordinates for the vertices of active triangles to obtain color and normal information. In this way we avoid the occurrence of discontinuities in the texture of an approximation. Note also that we create a vertex hierarchy for each object in the scene. This allows us to apply the approach to dynamic scenes and to support instantiation of objects. In order to reduce the computational expense for the generation of approximations we realized a caching mechanism both for geometry- and image-based approximations.

Before rendering an object we perform the following operations: First we check if there exists a valid approximation of the object in the cache. If a suitable approximation could be found, it is rendered. Otherwise a new approximation has to be created. If it is reasonable and efficient, we replace the geometry based approximation by an impostor. In contrast to former approaches we use the valid adaptive approximation to generate an impostor for the object.

Before the newly created object approximation is used to render the next frame it is stored in the cache.

Although it is not the focus of this paper we would like to mention that our approach has been designed for a distributed implementation on a PC cluster.

2. Creating and processing a vertex hierarchy for an LOD object

Our MRM of an object consists of a vertex tree, a list of active triangles and textures that contain attributes (represented by series of mip maps).

In contrast to [1] we create a vertex tree for each LOD object of the scene in a preprocess. Coordinates of vertices are saved as coordinates of object coordinate system. All computations are done in object coordinate system. This enables us to support dynamic scenes and instantiation of objects. We use an octree-clustering-scheme suggested in [1] to produce an object's MRM. This scheme does not require any knowledge about the polygonal mesh. Manifold topology is neither assumed nor preserved. A disadvantage of this vertex tree generation strategy is that meshes with degeneracies (as cracks, T-junctions, and missing polygons) may appear.

The list of active triangles contains the active triangles that form the current adaptive geometry-based approximation of the LOD object.

In addition to the vertex tree and the list of active triangles we produce a series of mip maps of a color texture and a series of mip maps of a normal map. Less complex versions of object's geometry can be used for rendering high quality images if these simple versions of object's geometry are used in conjunction with these textures.

The extraction of a suitable adaptive object approximation requires the following steps:

Because we perform all computations in the object coordinate system we need the current model transformation of the object to be able to transform all necessary parameters in this coordinate system. That causes additional computational effort, but it enables us to process dynamic scenes and to support instantiation of objects. Knowledge about the current viewing configuration is necessary to be able to extract the correct adaptive geometry-based approximation of an object out of its vertex tree.

The list of active triangles corresponds to a cut that is moved through the vertex tree depending on a view-dependent criterion at runtime.

This cut movement is performed incrementally because this is less expensive than determining the current cut by traversing the vertex tree starting at the root node.

The cut movement through the vertex tree is controlled by a view-dependent criterion [1] that is based on screen-space error and takes care of silhouette regions.

In contrast to [1] we compute texture coordinates for every vertex of an active triangle depending on its position in object coordinate system at runtime to obtain color and normal information because our approximations are represented by textured meshes. So discontinuities in the approximation's texture can be avoided.

3. Creating and processing an impostor for an LOD object

The algorithm has to do the following steps to create an impostor for an object. First it determines the proper viewing volume defined by a bounding rectangle in the proper image plane and the current view point as described in [2]. In contrast to [2] where the impostor texture is rendered using the original geometry we use an adaptive approximation of the object that was created before to render the impostor texture using the determined viewing volume. The alpha masks that masks out regions of the impostor texture that do not belong to the object's projection and the RGB image are produced in one pass. Now we read the generated RGBA image back from frame buffer to get the impostor texture. After that we determine the position of the vertices that form a screen-aligned quadrilateral where the impostor texture is applied to.

Creating an impostor for an object is expensive. The textured mesh representing the current valid approximation of the object has to be rendered and a time-consuming read back from the frame buffer has to be done. It isn't reasonable and efficient to do this in every situation.

So we further reduce an adaptive approximation of an object to an impostor only

- if the object can be separated from all other objects in the scene,
- if the ratio of needed texture resolution to frame resolution is smaller than a given threshold
- and if the costs for using an impostor are smaller than the costs for using the adaptive approximation (based on [3]).

4. Caching approximations of LOD objects

Caching the created image-based or geometry-based approximations of LOD objects in an approximation cache is useful because approximations often can be reused for several frames and do not have to be re-created for these frames. This causes an additional speed up of frame rate. Note that it is not necessary to distinguish between geometry-based and image-based approximations in this context because both are represented as textured polygons.

We save the created approximation together with an object id, a time stamp, the viewpoint v_0 it was created for and the bounding box vertices of the object it approximates. These additional parameters are used to check the validity of approximations. We check the validity using an angle-based criterion similar to the one suggested by Shade et al. in the context of hierarchical image caches [3]. For every vertex of the object's bounding box we compute two vectors in object coordinate system. The first vector points from the bounding box vertex B_i to the current viewpoint v. The second vector points from the bounding box vertex B_i to the viewpoint v_0 the approximation was created for.

For each bounding box vertex B_i we compute the angle $\theta_{size,i}$ between these two vectors:

$$\theta_{size,i} = \angle(v_0, B_i, v). \quad (1)$$

If the maximum of these eight angles exceeds a given threshold the object approximation is considered to be invalid and will not be used for rendering the next frame.

If no valid approximation of the object is available a new approximation of the object has to be created and inserted in the approximation cache controlled by a LRU mechanism.

5. Computing adaptive texture coordinates at runtime

After generation of approximation and before rendering a frame we compute the texture coordinates of an approximation's vertices. If the approximation is an impostor the determination of texture coordinates is trivial. If the approximation is an adaptive geometry-based approximation the following operations have to be performed. Color and normal information are sampled in color and normal textures in such a way that the correct color and normal information for an object's vertex can be obtained by using texture coordinates that are computed based on the position of vertex in object coordinate system.

After extracting an adaptive approximation out of the object's MRM we can use the inverse of the current modelview matrix and the position of vertex in object coordinate system to determine the corresponding

position in texture space where the correct color and normal information for the vertex is saved.

6. Supporting dynamic objects

The proposed LOD based rendering approach supports display of scenes that contain objects that can be transformed during simulation and whose geometry is constant. The majority of possible scenes can be rendered using this approach.

Sometimes there are scenes that contain objects whose geometry is changed by interaction with the user. New MRMs have to be produced for these objects during simulation. This is possible only if the proposed LOD based rendering approach is implemented in a distributed manner. In this case a separate task produces updated MRMs for objects based on a copy of the original scene graph. If an updated MRM is available all cached approximations of the object are deleted and this new MRM is used. This will be content of further studies.

7. Results

A distributed version of our approach was implemented. It consists of a rendering and a reduction component. The rendering component was developed at Chemnitz University of Technology in the context of another research project. The caching mechanism is integrated in this rendering component. The reduction component was implemented using functionality of VDSlib [4]. In a preprocess the rendering component sends the original geometry of each LOD object to the reduction component that produces an MRM for each object. During rendering the rendering component sends approximation requests to the reduction component that produces object approximations. These approximations are sent back to rendering component. Each component is implemented on a Pentium 4 with 1 GB RAM and a 3Dlabs Wildcat4 7210 graphics card. Both computers are connected by Gigabit Ethernet.

Our experience with our current implementation shows that it is possible to render complex scenes at high frame rates. Furthermore the proposed approach can be applied to dynamic scenes because we process a continuous MRM for each object. The ultimate speed up of frame rate is reached by using our approach if it is applied to scenes containing a lot of complex objects. In the third example an optimized model of an urban scenery that contains objects consisting of less polygons textured with a lot of textures is used. In this case we only reach a small speed up of frame rate.

Table 1. Statistics

Scene (#polygons, #objects, # object types, note)	Time for MRM generation (sec)	Frame rate (fps)	
		w/o LOD	w LOD
Cars (2,86 Mio., 19, 4, animation showing a race)	11,45	4,4	12,5
Turbine Blade (1,76 Mio., 1, 1, rotating blade)	28,89	1,3	20,2
Urban Scenery (197000, (15000, ca. 5000, walkthrough)	27,1	15,0	20,0

In the near future we will realize a complete distributed implementation of this system that also includes an improved caching mechanism. Further on an intelligent distribution mechanism has to be designed and an occlusion culling mechanism should be integrated in the rendering approach.

Figure 1. Shaded (top) and wire frame (bottom) representation of a dynamic car scene rendered without (left) and with (right) LOD.

8. References

[1] David Luebke and Carl Erikson, "View-Dependent Simplification of Arbitrary Polygonal Environments", *Computer Graphics (31), SIGGRAPH `97 Proceedings*, ACM, August 1997

[2] G. Schaufler. "Dynamically Generated Impostors.", D. W. Fellner, editor, *Modeling - Virtual Worlds - Distributed Graphics*, MVD'95 Workshop, November 1995, pp. 129–136.

[3] J. Shade et al., "Hierarchical Image Caching for Accelerated Walkthroughs of Complex Environments." H. Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, ACM SIGGRAPH, Addison Wesley, New Orleans, Louisiana, 04-09, August 1996, pp 75–82.

[4] VDSlib Homepage, University of Virginia, http://vdslib.virginia.edu