Volume 0 (1981), Number 0 pp. 1–4

Unpopping: Solving the Image-Space Blend Problem for Smooth Discrete LOD Transitions

Markus Giegl and Michael Wimmer

Institute of Computer Graphics and Algorithms
Vienna University of Technology

Abstract

This paper presents a new, simple and practical algorithm to avoid artifacts when switching between discrete levels of detail (LOD) by smoothly blending LOD representations in image space. We analyze the alternatives of conventional alpha-blending and so-called late-switching (the switching of LODs "far enough" from the eye-point), widely thought to solve the LOD switching discontinuity problem, and conclude that they either do not work in practice, or defeat the concept of LODs. In contrast we show that our algorithm produces visually pleasing blends for static and animated discrete LODs, for discrete LODs with different types of LOD representations (e.g. billboards and meshes) and even to some extent totally different objects with similar spatial extent, with a very small runtime overhead.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

Discrete level-of-detail rendering is a well known acceleration technique where complex objects are replaced by successively simpler (in geometry and/or shading) representations the farther they are from the eye point. During runtime, the renderer chooses from a series of so-called *levels of detail* (LODs) for each object. This *LOD selection* is often based either on the distance of the object from the observer, or on an estimate of the number of projected pixels of the object [FS93]. A detailed and comprehensive discussion of most level-of-detail techniques can be found in [LRC*02].

While discrete LOD creation and selection have been widely researched, *LOD switching*, i.e., the question how to stage the transition between two different discrete levels of detail of an object, is a significant open problem in using discrete LODs, as evidenced by popping artifacts due to "hard" LOD switching in state-of-the-art computer games such as FarCry or Half Life 2 (You can find screenshots of popping examples in games at `http://www.cg.tuwien.ac.at/research/vr/unpopping/pop`; the images on the page take a few seconds to load. Move your mouse over/out of the images to observe the popping).

Since LOD switching seems to be a simple task, we have found that most computer graphic researchers assume that the straightforward approach of doing a conventional $(\alpha, 1\text{-}\alpha)$-blend between different levels of discrete LODs solves the problem. However, this approach does not work, as can be seen by considering the intermediate states of such a blend, where both LOD representations—and therefore the whole object—are rendered semitransparently, blending with previously rendered objects and/or the background-color. The blend could be done using offscreen buffers, but this would have a high performance and memory usage penalty. Another commonly held opinion is that the problem can be solved by simply hard-switching at a large enough distance; while this may be true in a mathematical sense for purely geometric LODs that do not depend on the environment (e.g., via shaders), we argue that in practice this does not work due to the nature of the human perception system and that it defeats the purpose of LODs.

In this paper, we present a novel solution to discrete LOD switching that is at the same time visually pleasing, simple to implement and has a very small runtime overhead. The method is based upon a new image-space blend formulation of two representations of an object. Only standard, fixed pipeline graphics hardware features are used. Therefore the method is fast, leads to no conflict with modern hardware acceleration features such as hierarchical depth buffers, and is also suitable for less potent devices such as handhelds. Note that a short description of the idea based on an early draft of this paper already appeared in [MH02].

Another important contribution of this paper not yet found in literature is an in-depth discussion of the problems with commonly used LOD switching techniques, providing answers to common misunderstandings in the usage of LODs.

2. LOD Switching

Several solutions have been proposed for the problem of *LOD switching*, i.e., the question how to stage the transition between two different discrete levels of detail of an object:

2.1. Hard Switching

The simplest approach is *hard switching*, i.e., one LOD is replaced by another at some point in time. Although simple and fast, due to the nature of human perception this method leads to very noticeable and visually disturbing temporal discontinuity artifacts (“popping”).

2.2. Late Switching

One seemingly obvious solution is so-called *late switching*, i.e., to hard-switch at a “large enough” distance, where the difference between the two LODs is “no longer noticeable”.

There are several problems with this approach, which make it unfit for practical use:

It tries to fulfill two conflicting goals: on the one hand, increasing the frame rate to keep the application real time by switching as early as possible, and on the other hand, trying to reduce popping artifacts by switching as late as possible.

To *switch as early as possible* is necessary because the number of objects (and therefore the render cost) in a typical 2.5D scene (with a homogeneous object distribution) for which a more complex LOD has to be rendered increases quadratically with the switching distance. This is because the number of objects with distance smaller than r to the viewpoint is $\propto r^2$.

In addition, fill rate for costly shaders used for nearer LODs of an object is only saved when the object still covers a significant amount of screen space when the switch occurs.

As if this were not enough by itself, the second, contradicting goal to *reduce popping by switching as late as possible*, can in general not even be attained in itself, since the human visual system is fine-tuned to notice even small discontinuous changes. This is problematic because two LODs, even when rendered far from the viewpoint, cannot be guaranteed to produce identical images. The goal becomes completely unattainable when different shading effects (e.g., environment maps vs. simpler shading, or rendering specular highlights vs. diffuse shading only) are used for the different LODs, which is frequently the case in today’s games.

We conclude that *late switching* is not a practical approach. On the contrary, one wants to switch to lower LODs as *early* as possible in order to make good use of LODs.

2.3. LOD Blending in Image Space

LOD *blending in image space* is often mentioned as a generic way to do the transition between LODs. However, just linearly blending the two LODs is not possible, since an incorrect, semitransparent object would result during the blend, when both LODs get rendered semitransparently into the scene (see Figure 1 and the 2^{nd} paragraph in Section 3).

A variation of this method would be to render the two LODs opaquely into separate offscreen buffers and do an $(\alpha, 1\text{-}\alpha)$-blend between them: The render cost to do so is high, because it requires two offscreen buffers (+ depth buffers), the size of the frame buffer, combined with costly rendertarget switches to the offscreen buffers and back to the frame buffer to render the result of the LOD blend into the scene. This also requires depth buffer writes in the pixel shader, prohibiting modern hardware depth buffer speedup techniques, such as hierarchical depth buffers. In addition, the fringe area (where the two blended objects do not cover the same screen space area) leads to problems, because there a blend with the background color of the offscreen buffer would occur.

Multisampling hardware (as was supported by the SGI Infinite Reality [EJ00]), produces worse results than our approach, since the number of stages in the transition is limited to the number of samples in the multisample mask.

2.4. Geomorphing

Some mesh representations allow a technique called *geomorphing* [Hop96, Hop98], where vertex positions are interpolated between the two LODs. This requires that each vertex of one LOD can be uniquely identified with a vertex of the second LOD, which is only possible for special multiresolution mesh representation techniques. While geomorphing is widely referred to as the highest-quality LOD-switching technique, it can make solid structures (like terrains) appear to be moving in a quasi-organic way, and this effect can be as disturbing as hard switching itself (although recent approaches have brought significant improvements in this area, for example for terrain visualization applications [CGG*03]). While a lot of very interesting research has been done in this field, due to the comparatively large implementation effort required (and also partly due to the need for restripification), very few computer games use geomorphing. Note that even if the geometry transition can be made fairly smooth, appearance switches (e.g., less complex shaders, which might be even more important than geometric simplification in today’s games) are still hard switches

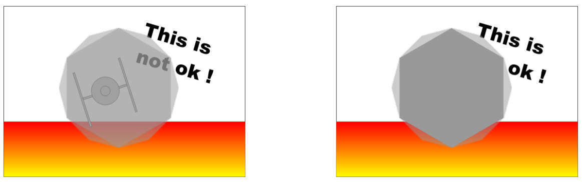

Figure 1: *A spherical object in transition between two LODs that project to a 6- and 12-sided polygon in screen space respectively. In the background are a spaceship, a planet surface and a text object. Left: Conventional blending leads to semitransparency during the blend, making objects behind the LOD object visible—the player could wrongly see the spaceship. Right: In our new method, one of the LODs is drawn opaquely, solving the transparency problem.*

		Unpopping		$(\alpha, 1\text{-}\alpha)$
		$t \in [0, 0.5[$	$t \in [0.5, 1]$	$t \in [0, 1]$
LOD_1	α	1	$2(1-t)$	t
	z-test	true	true	true
	z-writes	true	false	false
LOD_2	α	$2t$	1	$1-t$
	z-test	true	true	true
	z-writes	false	true	false

Figure 2: *The table shows render state settings during the LOD blend. The last column shows the render state settings of the conventional* $(\alpha, 1\text{-}\alpha)$*-blend in comparison.*

and will therefore pop. Also, progressive representations do not allow completely different LOD representations (e.g., disjointed vs. skinned LODs). Discrete LODs do not suffer from these limitations, and can stage such transitions smoothly using the method presented in this paper.

3. A new method for LOD blending

The aim of LOD blending is to stage a smooth transition between two level-of-detail representations, called LOD_1 and LOD_2. The transition should take place for a parameter t, depending for instance on time or distance, running from 0 to 1. The problem then is how to render the object as a blend between LOD_1 and LOD_2 at any parameter value t between 0 and 1.

The traditional interpretation of alpha-blending suggests a blend of $1-t$ times LOD_1 (usually implemented by setting the transparency value of the object to $1-t$), and t times LOD_2. However, this interpretation is unsuited for LOD blending in practice because during the blend both LOD representations would be rendered semitransparently over previously rendered objects and the background (see Figure 1). If the LOD-blend parameter t is based on distance to the eyepoint, the user will see a permanently transparent object as long as his distance to the object stays the same. But even if the blend is done over time, there is no sweet spot for the blend time: Either the blend is short enough, then we perceive popping, or it is longer, then the object is perceived to be semitransparent (which is more disturbing than the pop).

What we would need is an object that at least approximately fills the depth buffer correctly before the blend, so that the scene behind does not shine through. The main idea of our new algorithm is to use the LODs themselves for this purpose, which together with the right choice of depth buffer writes and depth comparison gives a continuous LOD transition in image space:

First, from $t = 0$ to 0.5, render the current LOD, LOD_1, opaquely and with depth writes, and “fade in” the new LOD, LOD_2, by rendering it alpha blended without depth writes but depth compares on top of it. Then, when LOD_2 is faded in completely (i.e., both LODs are rendered opaquely atop of each other and are therefore interchangeable), switch roles, rendering LOD_2 opaquely, and “fading out” LOD_1 from $t = 0.5$ to 1. Thus, at any one time during the transition, one of the two objects is rendered opaquely, and will therefore create a valid depth buffer.

Figure 2 shows the different render state settings during the blend.

The new algorithm can also be interpreted the following way: the first stage blends from LOD_1 towards the CSG-*union* of the two LODs, whereas the second stage blends from the union towards LOD_2.

4. Discussion

In the algorithm presented, the opaque LOD provides the depth buffer content also for the transparent LOD. This leads to the following minor artifacts:

First, in areas where the silhouettes of the two LODs do not match exactly, the semitransparent LOD will blend with the rest of the scene. Also, the part of the semitransparent LOD protruding from the opaque LOD will be hidden by a more distant opaque object if it is drawn afterwards. We have found that these effects are negligible for all practical purposes, since the areas were these effects occur are very small and appear and disappear smoothly. If one still wanted to get rid of the latter effect and minimize the former, one simply would have to draw the opaque LODs first, and the semitransparent LODs back-to-front afterwards (just like conventional semitransparent objects in the scene).

Second, if a LOD-blended object *intersects* other geometry, there will be a discontinuity in depth space at $t = 0.5$, when the switch between the LOD being rendered into the depth buffer occurs. In this case a discontinuity at the intersection between the object and the scene will occur, because the transition is not smooth in depth space. This is not a problem in practice, because in correctly modeled scenes intersections between objects and the scene do not occur. In any case, the artifacts that occur are much less pronounced than the popping coming from a hard switch, because the screen space area that changes abruptly is only a small part compared to the area affected by a hard LOD switch.

One other possible problem is "z-fighting", which can occur if two LODs contain very similar polygons in depth space, which at the same time differ significantly in image space (e.g. due to shading, for example if the normal vectors of the two LODs differ by a large amount). If the LODs are modeled by hand, this problem can easily be avoided by the artist. If they are created automatically and the problem appears, one can extend the algorithm to add a small polygon offset to one of the two LODs.

5. Results

We found that a very short transition phase (below one second) is sufficient to eliminate popping and provide a smooth transition. This also gives the best performance since only a small number of objects incur the overhead of being drawn with two LODs at any one time.

The method also works well for *animated models* with LODs—another area where progressive mesh techniques have a hard time providing good visual quality for low-detail meshes. Geomorphed meshes have a tendency of showing effects like foldovers and too-slim body parts when animated. With discrete LODs, on the other hand, disjointed models (even using line rendering) can be used for the lower-detail LOD representations. The algorithm we presented can also be combined with progressive mesh techniques, for instance doing very high resolution progressive mesh rendering of a model in the foreground, and blending to a discrete, disjointed model LOD representation farther away.

The two objects being blended are not necessarily restricted to represent levels of details. It is possible to blend two totally different objects, obtaining the effect of a visually smooth transition between them.

At http://www.cg.tuwien.ac.at/research/vr/unpopping/examples you can find videos and screenshots comparing hard switching (artifact: popping), the "standard" $(\alpha, 1\text{-}\alpha)$-blend (artifact: see-through of background; popping, when blend time becomes $\ll 1s$), and our algorithm. Please note that the artifacts are more pronounced during the actual rendering than in the videos, due to the unavoidable blurring/softening done by the video codec. You can also find videos that show the use of the technique in the commercial computer game "FBI Academy", and screenshots depicting LOD popping in current computer games.

6. Conclusions

LOD switching is an underestimated problem: Even the most current computer games still show visible popping artifacts, since the obvious approaches to reduce the popping do not work.

This paper shows a new interpretation of blending which works on current graphics hardware, with all LOD schemes, and integrates well with the rendering pipeline by providing a valid depth buffer. It can be added to all real-time applications with minimal implementation effort.

Acknowledgements

This research was partly supported by the EU (project no. IST-2-004363) in the European Union GameTools Project (http://www.gametools.org).

References

[CGG*03] CIGNONI P., GANOVELLI F., GOBBETTI E., MARTON F., PONCHIO F., SCOPIGNO R.: BDAM – batched dynamic adaptive meshes for high performance terrain visualization. *Computer Graphics Forum 22*, 3 (2003), 505–514. 2

[EJ00] ECKEL G., JONES K.: *OpenGL Performer Programmer's Guide*. SGI techpubs library, 2000. Document Number 007-1680-060. 2

[FS93] FUNKHOUSER T. A., SÉQUIN C. H.: Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *ACM SIGGRAPH 93 Conference Proceedings* (1993), pp. 247–254. 1

[Hop96] HOPPE H.: Progressive meshes. In *ACM SIGGRAPH 96 Conference Proceedings* (1996), pp. 99–108. 2

[Hop98] HOPPE H.: Smooth view-dependent level-of-detail control and its application to terrain rendering. In *Proceedings of the conference on Visualization '98* (1998), pp. 35–42. 2

[LRC*02] LUEBKE D., REDDY M., COHEN J. D., VARSHNEY A., WATSON B., HUEBNER R.: *Level of Detail for 3D Graphics*. Elsevier Science Inc., 2002. see also www.lodbook.com. 1

[MH02] MÖLLER T., HAINES E.: *Real-Time Rendering, Second Edition*. A. K. Peters Limited, 2002. 2