# Smooth View-Dependent Rendering in Animations*

Axel Friedrich†
Konrad Polthier‡
Markus Schmies§

Technical University Berlin
Department of Mathematics, Sfb288

## Abstract

View-dependent rendering allows interactive visualization of larger scenes. A well-known artifact is the popping problem in animations resulting from temporal differences in the level of detail between the view-dependent representations used in subsequent frames. We solve the popping problem by computing view-dependent representations of a scene only at every $n$-th frame, and smoothly interpolate adjacent keyframe sections to obtain the representations for all in-between frames. Additionally, we accelerate rendering at only minor accuracy costs since interpolation is much faster than computing a view-dependent for each displayed frame. We use scenes represented as triangle hierarchies and fulfilling a special constraint allowing for fast interpolation without remeshing. In contrast to other approaches our method naturally extends to animated scenes whose geometry and mesh may adaptively change in time.

**Keywords:** animation, shape interpolation, adaptive refinement, level-of-detail, multiresolutional representation, view-dependence, morphing

## 1 Introduction

The visualization of large scenes at interactive frame rates often uses view-dependent rendering. A well-known artifact is the popping problem in such animations and interactive fly-throughs. This results from view-dependence computations applied to the same region of a scene in subsequent frames because the view-dependence criterium is applied to a scene at a static moment and the criterium assures no temporal coherence from one frame to a next. There exist solutions to avoid the popping problem based on a screen space tolerance, i.e. the geometry is only rendered until the level of detail falls below the size of a pixel [9][11][6][18]. Such a solution works with any level of detail geometry representation and requires no continuous level transitions. A drawback is its dependence on the quality of an image device which makes it hard to use coarser level of detail representations.

Smooth level of detail representations exist e.g. for vertex-based hierarchies [9], wavelet-based representation, and triangle-based hierarchies [7] [6], and such representations may avoid popping if there is a temporal coherence in the view-dependence criterium. Although view-dependence techniques reduce the rendering time, they still require more or less expensive view-dependence computations.

In this paper we describe a different approach to avoid popping, and simultaneously speed-up rendering by reducing the number of necessary view-dependence computations. Additionally, we reduce the complexity of each remaining view-dependence computation. The principal idea of our approach is to compute a view-dependent representation of a scene only at every $n$-th frame, in the following referenced as keyframe representation of the scene, and generate the in-between representations by smooth interpolation of two adjacent keyframe representations. Since it is seldom that adjacent keyframes have the same mesh, see figure 1, our approach requires a fast interpolation algorithm for differently refined meshes. We ensure this interpolation property by making use of the concept on interpolating triangle hierarchies described [7], and are able to define even higher order interpolation between keyframe hierarchies.

The idea of reducing the number of view-dependence computations has been mentioned already in the paper by Rohlf and Helman [15] as a possible choice for obtaining high constant frames, and, more recently, Hoppe [10] has included it in the vertex-based progressive mesh concept. Our solution is defined on triangle-based hierarchies and therefore has the advantage, that it naturally extends to animated scenes whose geometry and mesh change adaptive in time. This extension to animated scenes is original, and is a direct consequence of the results of [7] combined with this work.

We discuss a further problem occuring in interactive situations where the current time lies between two keyframes, and the camera position at the next keyframe is not available yet. We discuss three approaches to extrapo-

*Unintentionally, this paper was concurrently published by the WSCG'99 conference. However, the publication here is considered as the official version.

†axel@sfb288.math.tu-berlin.de
‡polthier@math.tu-berlin.de
§schmies@sfb288.math.tu-berlin.de

late the current camera position by minimizing a temporal error against a geometric error, respectively vice versa.

For an overview of keyframe animations we refer to the book [13] and its detailed bibliography. A different method avoiding popping artifacts is based on image blending which softens the difference between two levels of detail [15][5].

In section 2 we review the technique of interpolating triangle-based hierarchies for different, adaptive refined meshes, which is the basis for the interpolation between view-dependent scene with varying mesh representations. In section 4 we discuss the extrapolation of camera positions in interactive situations to minimize temporal against geometric errors

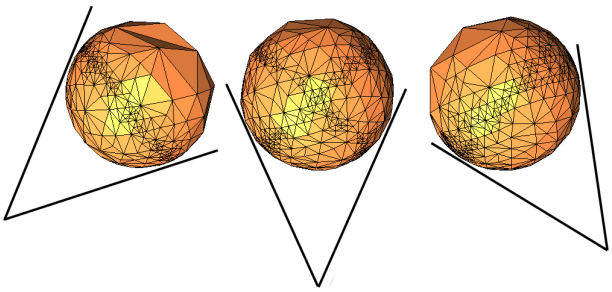

Figure 1: Left and right image show view-dependent representations of a sphere depending on the indicated camera position. The middle image shows the sphere at an intermediate time. Its rendered mesh has been obtained by interpolating the two view-dependent meshes of the adjacent keyframes, thereby avoiding a new view-dependence computation.

# 2 Review on Interpolating Triangle Hierarchies

We consider interpolation between keyframe hierarchies which are allowed to have different meshes. If there is no coherence between the meshes then the interpolation is usually a tedious task and often requires expensive remeshing operations for the generation of the in-between mesh. Therefore we introduced in Friedrich et al. [7] two weak constraints that ensure smooth and higher-order interpolation between two adaptive refined keyframe hierarchies in an animation. This concepts are essential for the ideas in this paper and we use this section to review the interpolation principle.

We consider hierarchies of triangles obtained by a triangle bisection algorithm, rather than by vertex split/edge collapse methods, and impose the following two constraints: firstly, the bisection scheme in each hierarchy follows the rules of Rivara in definition 1 and, secondly, a correspondence between the sets of root triangles of all hierarchies is required. These assumptions are rather weak, especially since only the second constraint requires a correlation between different hierarchies. After an initial adjustment, each keyframe hierarchy may be arbitrarily refined and coarsened locally depending only on its own error criteria without spoiling the interpolation property. This freedom for local refinement and coarsening is the property allowing for interpolation between view-dependent scenes with different meshes.

Instead of using the Rivara bisection method, one might try to use a 4-1 split as subdivision rule. After a local refinement the 4-1 split must be accompanied by a process called conformal closure to remove hanging vertices, e.g. by introducing so-called green edges [2]. These green edges are responsible for case distinctions and require further subdivisions when interpolating between different hierarchies. Such tasks can be handled, but only at the cost of complex remeshing operations which we try to avoid.

A triangulation is called *conforming* if two adjacent triangles either share exactly one common vertex or one common edge. Conforming triangulations have no hanging nodes, i.e. vertices which lie in the interior of an adjacent edge, and therefore avoid cracks when deformed.

## 2.1 Bisection method of Rivara

The bisection algorithm of Rivara [14] addresses the problem of how to locally refine a conforming triangulation to a new conforming triangulation and, additionally, of how to ensure that all angles in subsequently refined triangulations are greater than or equal to half of the smallest angle in the original triangulation. The method leads to nested triangulations and allows smooth transition between different levels of detail. In his original formulation, Rivara bisects a triangle exactly at the longest edge. Bänsch [3] generalized the method by introducing a formal refinement edge. In each triangle a single edge is marked as *refinement edge*, i.e. if the triangle is refined, then it is refined by bisecting its refinement edge, and the two child triangles inherit a refinement edge in the manner shown in figure 2. In the simplicial complex an additional vertex is inserted at the midpoint of the refinement edge but the geometric position of the new vertex may change in space, of course.

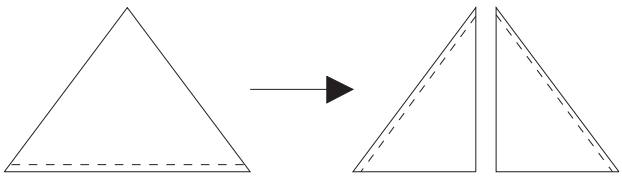

Figure 2: The Rivara bisection method refines a triangle exactly at its refinement edge. Each child inherits a refinement edge as shown.

Formally, the Rivara/Bänsch algorithm assumes that in a

triangulation $T$ each triangle has an arbitrary edge marked as refinement edge. Let $T_k$ be a conforming triangulation with a subset of triangles $S \subset T$ marked for refinement, usually according to some local error criteria, then the method consists of the following steps:

**Definition 1** Rivara/Bänsch Bisection Method

*1. All marked triangles $S$ are bisected according to the Rivara bisection rule. This produces a (possibly empty) new set of non-conforming triangles.*

*2. Mark all non-conforming triangles for refinement; the set is again denoted with $S$.*

*3. If $S$ is not empty, then go to 1. Otherwise, there are no marked triangles and the algorithm stops. The new triangulation is $T_{k+1}$.*

When the algorithm stops the new triangulation is conforming. As shown in [3] the algorithm stops after a finite number of steps since in one pass it inserts at most a single vertex on each edge. This is a fairly rough upper estimate for theoretical purposes – and one can construct such badly behaved examples – but, in practice, the subdivision has only local influence on the triangulation, see [3], [14]. The sequence $\{T_k\}$ is stable, i.e. all triangle angles are bounded from below by half the minimum triangle edges of the first triangulation $T_1$.

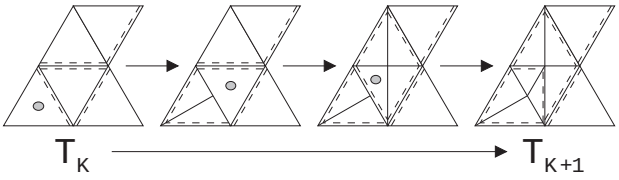


Figure 3: A single step of the Rivara/Bänsch bisection method with refinement edges and marked triangles.

## 2.2 Interpolation Constraints

A *triangle hierarchy* is a hierarchical structure of triangles where each triangle has a reference to one *parent* element, to one *child*, and to a *sibling*. The sibling is a child of the same parent, and all children are produced by subdivision of the parent. Triangles with no parents are called *root triangles*, and triangles with no children are *leaf triangles*. We assume that the geometric vertices of a hierarchy are given in a global vertex array, and each triangle is determined by three vertex indices. Vertices and triangles usually have color and material properties, or carry texture coordinates. Triangles may have references to neighbour triangles.

We now specify constraints on the key hierarchies that, firstly, guarantee a smooth interpolation without the need to remesh during the interpolation process, and, secondly, ensure the freedom for local grid modifications separately on each keyframe.

**Definition 2** *A family of triangle hierarchies $F$ must fulfill the following* Interpolation Constraints:

*1. The simplicial complex of the root triangles of each hierarchy is the same for all hierarchies in $F$, i.e. for each pair of hierarchies $G, H \in F$ there exists a bijective simplicial map $\phi_{GH}$ between the set of root triangles.*

*2. Each root triangle has a refinement edge, and the simplicial map $\phi_{GH}$ maps each refinement edge to a refinement edge, i.e. the root triangles of all hierarchies are marked in the same way.*

*3. Each hierarchy is refined using the Rivara/Bänsch Algorithm 1.*

The root triangles can be interpreted as charts of each hierarchy, and condition 1. requires a bijective correspondence between the charts of different hierarchies. It is important to note, that condition 1. does not ensure the interpolation property. It is essential to have conditions 2. and 3. satisfied additionally to ensure that hierarchies are automatically refined in a synchronized way. Each hierarchy can be refined according to its own error criteria without *a posteriori* synchronization with the other key hierarchies and without spoiling the interpolation property. Once one has agreed to use the Rivara/Bänsch bisection method, it only remains to ensure properties 1. and 2. for the family $F$ in an initial synchronization step.

The following central theorem was proved in [7]:

**Theorem 1** *Let $F = \{H_1, H_2, ..\}$ be a family of hierarchies which fulfill the interpolation constraints 2 and let $b = \{b_1, b_2, ..\}$, $b_i \in R$, be a set of weights. Then there exists an interpolated hierarchy*

$$H(b) = \sum_i b_i H_i$$

*which depends smoothly on $b$, and its underlying simplicial hierarchy is the union of the simplicial hierarchies of each $H_i$. Further, the interpolated hierarchy $H$ depends smoothly on $b$ and fulfills the same interpolation constraints as the elements of $F$.*

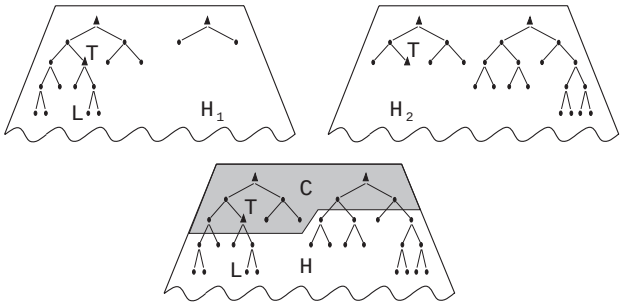


Figure 4: Two hierarchies fulfilling the interpolation constraints of definition 2. are subsets of the same infinite hierarchy. The interpolation hierarchy $H$ is the union of $H_1$ and $H_2$ .

The interpolation property of theorem 1 immediately allows higher order interpolation in a keyframe animation. Let $H_i$,..,$H_{i+n}$ be $n+1$ successive keyframe hierarchies fulfilling the interpolation constraints of definition 2, then

$$H(t) = \sum_{i=0}^{n} B_{i,n}(t) H_i \quad (1)$$

is a polynomial hierarchy interpolant of degree $n$ where $B_{i,n}(t)$ are the Bernstein polynomials.

# 3 View-Dependent Keyframes

For simplicity we assume a scene with a single geometry. The geometry is either given by a static triangle-based hierarchy obtained by Rivara/Bänsch bisection and only the camera is animated in the scene, e.g. a fly-through of a terrain. Or the scene consists of an animated geometry whose shape and mesh might change in time additional to the camera flight. In the second case the geometry must be given by a set of keyframe hierarchies which satisfy the interpolation constraints 2 and therefore allow a smooth interpolation by theorem 1.

Let $t$ be the time in an interactive fly-through or of an animation of the geometry. We perform the view-dependent extraction of the geometry only at certain discrete timesteps $\{t_1, t_2, ...\}$ of the time interval which are (by a factor of, say 10) less than the number of actual shown frames. A frame generated at time $t$, where $t$ lies in an open interval $(t_i, t_{i+1})$ between two timesteps, renders the geometry obtained by interpolating the view-dependent keyframe sections at times $t_i$ and $t_{i+1}$. Theorem 1 ensures that the interpolation is continuous over the whole time line since the view-dependent extraction of a subhierarchy at each timestep is the same as a coarsening step of the hierarchy. Such local modifications on a single key hierarchy are allowed by the assumption of the theorem. They do not spoil the original interpolation property. In practice, the proportion of the number of frames to the number of timesteps is between 5 and 10, and it depends on the amount of camera movement. Figure 1 gives a good impression for timesteps further apart. Nevertheless, the interpolation property holds for each choice.

In view-dependent computations, at each frame each vertex or triangle of the hierarchy is assigned a level of detail error. If this level of detail error depends continuously on the position of the camera and the geometry then it is possible to generate a section which is continuous in time.

It is important to note that computing the interpolation between the two hierarchy levels above and below the current view-dependence threshold can be avoid in our method. This leads to a further essential speed-up for the view-dependence computation. In our method we accelerated view-dependence computation of the keyframe subhierarchies by using the floating point view threshold as a boolean value: if the threshold is between the error values of two levels then we simply take the upper level and do not interpolate between the upper and next lower level. Such interpolation is the basis in the continuous level of detail concept but can be avoid here. Therefore, we do not employ the continuity feature with respect to the level of detail in the hierarchy when generating a view-dependent keyframe. In fact, this introduces a minor error term, but it leads to a significant acceleration of the view-dependent extractions since no interpolation between level of details must be computed, triangles are just assigned a visibility flag.

This acceleration introduced in the above paragraph might sound like introducing discontinuities, but the continuity returns when we interpolate between different view-dependent sections, i.e. two keyframes. Here it does not matter that both keyframes did not interpolate between their levels.

In practice, we mark at each keyframe $t_i$ those triangles with a visibility error above a certain threshold with a separate flag, say $i$, and do not compute any in-between vertices as usually done in continuous level of detail representations. Triangles which are also visible in the next keyframe carry an additional flag $i+1$. In case we use linear interpolation between keyframe hierarchies a ternary flag suffices.

The interpolation hierarchy between two keyframe sections at $t_i$ and $t_{i+1}$ consists of those triangles marked either $i$ or $i+1$. Its triangle tree is therefore encoded as a subtree of the triangle tree of the full hierarchy by theorem 1. There is no need to store the interpolation hierarchy explicitly. It requires only an additional vertex array for those interpolated vertices referenced by triangles from only one of the two keyframe sections. Therefore in practice, we work with just one hierarchy but have an additional vertex array for the interpolated vertices.

It is important to note that the work-load of the computer is higher at the timesteps $t_i$, where a view-dependence computation must be performed, than at the intermediate frames between timesteps, where only interpolation is required. The handling of such temporal variations of the load balance occurs in many other occasions and is a delicate problem when achieving constant frames rates [15][12][4][8].

# 4 View-point Extrapolation

## 4.1 Finding the Next Keyframe

A problem occurs from the fact, that in an interactive fly-through we do not know in advance the next keyframe. If we are at time $t \in (t_i, t_{i+1})$ then the keyframe section at time $t_{i+1}$ has not been computed yet. We experimented with three solutions which employ three types of errors. Here we assume to know the exact current, and previous, position of the camera, which might not be the case in ap-

plications, e.g. when tracking head mounted displays discussed in [17][1] [16]

We set the view-dependent section $S_{view}(t)$ at time $t$ as the optimal geometry since it is the geometry shown in a usual view-dependent rendering.

1. We allow a delay in the rendering, i.e. at current time $t \in (t_i, t_{i+1})$ we render the geometry corresponding to a delayed time $t_d \in (t_{i-1}, t_i)$ with

$$t_d = t_{i-1} + \frac{t - t_i}{t_{i+1} - t_i} (t_i - t_{i-1}),$$

and use the transformation matrix $M(t_d)$ at time $t$. When interactively steering the camera this approach leads to a slight delay in all actions. The delay obtained from showing the interpolated section $S_{inter}(t_d)$ at some later time $t > t_d$ is only recognizable by the "pilot" of the fly-through. Beside the delay and the necessity to cache all transformation matrices in the interval $(t_d, t)$, we only have an **interpolation error**

$$err_{inter}(t_d) = |S_{view}(t_d) - S_{inter}(t_d)|$$

between the view-dependent section $S_{view}(t_d)$ of the scene at time $t_d$ and the interpolated scene $S_{inter}(t_d)$. The error is measured with respect to some surface norm.

2. The delay $t - t_d$ can be avoided if we render the interpolated scene $S_{inter}(t_d)$ with the current transformation matrix $M(t)$. Here we accept a slight view-dependent error in the representation additional to the interpolation error since we show the view-dependent interpolated scene $S_{inter}(t_d)$ with the current transformation matrix at time $t$. The **view-dependent error**

$$err_{view}(t, t_d) = |S_{view}(t) - S_{view}(t_d)|$$

is determined by the view-dependent sections $S_{view}(t_d)$ and $S_{view}(t)$ at two different times $t_d$ and $t$.

3. In the third approach the camera position at time $t_{i+1}$ is extrapolated from previous positions, and the view-dependent keyframe at time $t_{i+1}$ is computed with respect to the extrapolated matrix $M_{extra}(t_{i+1})$. This allows to compute the interpolated geometry $S_{inter}(t)$ at the current time $t$ with the current transformation matrix $M(t)$. Here we have no delay in the rendering, but we have included an additional **extrapolation error** given by

$$err_{extra}(t_{i+1}) = |S_{view}(t_{i+1}) - S_{extra}(t_{i+1})|$$

at a keyframe position $t_{i+1}$ between the extrapolated and actual sections.

Our approaches balance between these three errors. In approach 1, we have at time $t$ an interpolation error $err_{inter}(t_d)$ and a delay of $|t - t_d|$, i.e.

$$err_1(t) = err_{inter}(t_d) + delay(t, t_d).$$

In approach 2, we avoid the delay by adding an additional view-dependent error, i.e.

$$err_2(t) = err_{inter}(t_d) + err_{view}(t, t_d).$$

Approach 3 avoids the view-dependent error at the cost of an extrapolation error at time $t_{i+1}$, i.e.

$$err_3(t) = err_{inter}(t) + err_{extra}(t_{i+1}).$$

# 5 Summary and Future Work

We increase the performance of animations by interpolating between view-dependent representations which restricts view-dependence computations to only every $n$-th frame and avoids popping artifact. To ensure a cheap interpolation property between different keyframe meshes we employ the interpolation constraints for triangle hierarchies defined in [7]. The method not only applies to static scenes where the camera moves, but, without change, applies to animated scenes whose geometry and meshes vary in time.

It is worth to discretize the time line adaptive depending on the amount of camera movement and a visual respective screen space error following ideas by Lindstrom et al. [11]. Although we use a screen based error for the definition of the view-dependent keyframes, our errors used above should also include a screen based distance.

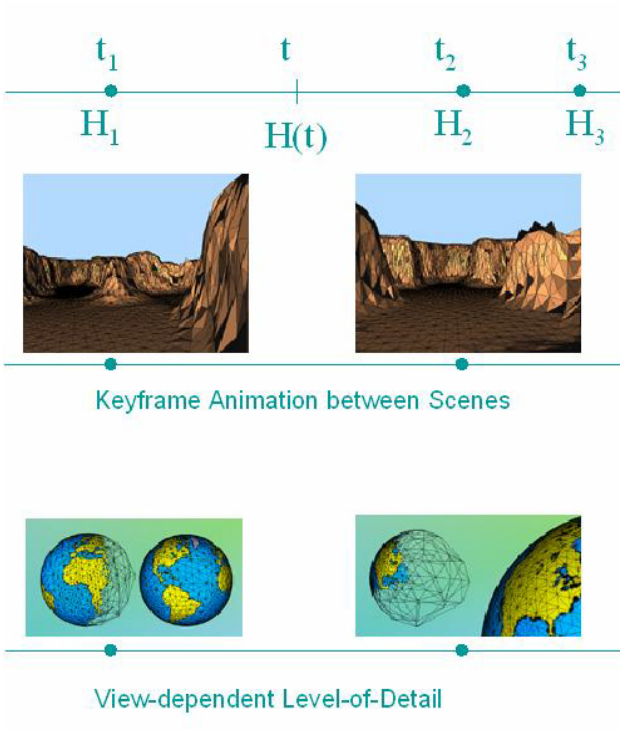


Figure 5: Interpolation between view-dependent keyframe representations H($t_1$) and H($t_2$) at times $t_1$ resp. $t_2$ of an animated scene. Since the view-dependence computation has been performed on the keyframe representations, the interpolated representation H(t) can be rendered directly. This approach avoids the view-dependence computations at every frame.

## References


[1] Ronald Azuma and Gary Bishop. A frequency-domain analysis of head-motion prediction. In *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 401–408, August 1995.

[2] R. E. Bank and A. H. Sherman. The use of adaptive grid refinement for badly behaved elliptic partial differential equations. *Advances in Computer Methods for Partial Differential Equations*, 3:33–39, 1979.

[3] Eberhard Bänsch. Local mesh refinement in 2 and 3 dimensions. *IMPACT of Computing in Science and Engineering*, 3:181–191, 1991.

[4] Steve Bryson and Sandy Johan. Time management, simultaneity and time-critcal computation in interactive unsteady visualization evironments. In Roni Yagel and Gregory M. Nielson, editors, *Proceedings Visualization '96*, pages 255–661. IEEE Computer Society Press, October 1996.

[5] Daniel Cohen-Or and Yishay Levaloni. Temporal continuity of levels of detail in delaunay triangulated terrain. In Roni Yagel and Gregory M. Nielson, editors, *Proceedings Visualization '96*, pages 37–41. IEEE Computer Society Press, October 1996.

[6] Mark Duchaineau, Murray Wolinsky, David E.Sigeti, Mark C. Miller, Charles Aldrich, and Mark B. Mineev-Weinstein. ROAMing terrain: Real-time optimally adapting meshes. In Roni Yagel and Hans Hagen, editors, *Proceedings Visualization '97*, pages 81–88. IEEE Computer Society Press, October 1997.

[7] Axel Friedrich, Konrad Polthier, and Markus Schmies. Interpolating triangle hierarchies. In David Ebert, Holly Rushmeier, and Hans Hagen, editors, *Proceedings Visualization '98*, pages 391–396. IEEE Computer Society Press, October 1998.

[8] Thomas A. Funkhouser and Carlo H. Séquin. Adaptive display algorithm for interactive frame rates durin visualization of complex virtual environments. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, pages 247–254, August 1993.

[9] Hugues Hoppe. View-dependent refinement of progressive meshes. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 189–198, August 1997.

[10] Hugues Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. In David Ebert, Holly Rushmeier, and Hans Hagen, editors, *Proceedings Visualization '98*, pages 35–42. IEEE Computer Society Press, October 1998.

[11] Peter Lindstrom, David Koller, William Ribarsky, Larray F. Hodges, Nick Faust, and Gregory A. Turner. Real-time, continuous level of detail rendering of height fields. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 109–118, August 1996.

[12] David Luebke and Carl Erikson. View-dependent simplification of arbitraty polygonal environments. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 199–208, August 1997.

[13] Nadia Magnenat-Thalmann and Daniel Thalmann. *Computer Animation*. Computer Science Workbench. Springer Verlag, second edition, 1985.

[14] M. Rivara. Algorithms for refining triangular grids suitable for adaptive and multigrid techniques. *International J. for Numerical Methods in Engineering*, 20:745–756, 1984.

[15] John Rohlf and James Helman. Iris performer: A high performance multiprocessing toolkit for real-time 3d graphics. In Andrew Glassner, editor, *Computer Graphics (SIGGRAPH '94 Proceedings)*, pages 381–394, July 1994.

[16] Greg Welch and Gary Bishop. Scaat: Incremental tracking with incomplete information. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 333–344, August 1997.

[17] Jiann-Rong Wu and Ming Ouhyoung. Reducing the latenxy in head-mounted displays by a novel prediction method using grey system theory. *Computer Graphics Forum (EUROGRAPHICS '94 Proceedings)*, 13(3):503–512, 1994.

[18] Julie C. Xia and Amitabh Varshney. Dynamic view-dependent simplification for polygonal models. In Roni Yagel and Gregory M. Nielson, editors, *Proceedings Visualization '96*, pages 327–334. IEEE Computer Society Press, October 1996.