Perceptually Driven Interactive Rendering

David Luebke and Benjamin Hallen

University of Virginia Tech Report #CS-2001-01

Abstract

We present a framework for accelerating interactive rendering, grounded in psychophysical models of visual perception. This framework is applicable to rendering techniques that regulate resolution—and hence rendering load—using a hierarchy of local simplification operations. Our method drives those local operations directly by perceptual metrics: the effect of each simplification on the final image is considered in terms of the contrast the operation will induce in the image and the spatial frequency of the resulting change. A simple and conservative perceptual model determines under what conditions the simplification operation will be perceptible, enabling *imperceptible simplification* in which operations are performed only when judged imperceptible. Alternatively, simplifications may be ordered according to their perceptibility, providing a principled approach to best-effort rendering. Our approach addresses many interesting topics in the acceleration of interactive rendering, including imperceptible simplification, silhouette preservation, and gaze-directed rendering. We demonstrate our framework applied to two quite different multiresolution rendering paradigms: view-dependent polygonal simplification and the *QSplat* point-based rendering system of Rusinkiewicz and Levoy [26].

1 Introduction

Interactive rendering of large-scale geometric datasets continues to present a challenge for the field of computer graphics. Such interactive visualization is an enabling technology for many far-flung fields, ranging from scientific and medical visualization to entertainment, architecture, military training, and industrial design. Despite tremendous strides in computer graphics hardware, the growth of large-scale models continues to outstrip our capability to render them interactively. A great deal of research has therefore focused on algorithmic techniques for managing the rendering complexity of these models. *Level of detail* management offers a powerful tool for this task. Level of detail or *LOD* methods hinge on the observation that most of the complexity in a detailed 3-D model is unnecessary when rendering that model from a given viewpoint. These methods simplify small, distant, or otherwise unimportant portions of the scene, reducing the rendering cost while attempting to retain visual fidelity.

Visual fidelity, however, has traditionally been difficult to quantify, so most LOD algorithms settle for geometric measures of quality. For example, the most common use of LOD management is *polygonal simplification*, in which a 3-D polygonal model is replaced with a simpler model using fewer polygons. Here geometric fidelity of the simplified surface may be measured with the distance of that surface from the original mesh, or with the volume of distortion created by the simplification. Such metrics are useful for certain CAD applications, such as finite element analysis, and for certain medical and scientific visualization tasks, such as co-registering surfaces or measuring volumes. Probably the most common purpose of simplification, however, is to accelerate interactive rendering. For this purpose, the most important measure of fidelity is not geometric, but perceptual: does the simplification *look* like the original?

In this paper, we describe a framework for LOD management guided directly by perceptual metrics. These metrics derive from the *contrast sensitivity function* or *CSF*, a measure of the perceptibility of visual stimuli. Testing local simplification operations against a model of the CSF provides a principled approach to the tradeoff between fidelity and performance. This approach addresses several interesting problems in regulating level of detail:

- **Imperceptible simplification:** We evaluate simplification operations by the "worst-case" contrast and spatial frequency they induce in the image, and apply only those operations judged imperceptible. We hypothesize that the resulting simplified model is indistinguishable from the original, and test that hypothesis in this paper.
- **Best-effort simplification:** Often we wish to render the best image possible within time or polygon constraints. Ordering simplification operations according to the viewing distance at which their effect on the image becomes perceptible furnishes a framework for simplifying to a budget.
- **Silhouette preservation:** Silhouettes have long been recognized as visually important, but how important? Our model quantifies silhouette importance by accounting for their increased contrast, and preserves them accordingly.
- **Gaze-directed rendering:** If the system can monitor the user's gaze, the image may be simplified more aggressively in the periphery than at the center of vision. We can extend our model to incorporate *eccentricity*, or the falloff of visual acuity in the periphery.

Our framework applies to any rendering system based on hierarchical simplifications or approximations. Many interactive rendering acceleration schemes fall into this category, including

Figure 1: Perceptually driven QSplat. Splats drawn in blue have been simplified. Left: QSplat's highest quality rendering mode traverses until each splat is less than a pixel in size. Right: perceptually driven rendering traverses nodes only where dictated by the local contrast and spatial frequency. Our model can also account for gaze direction: here, the user's gaze rests on Lucy's torch.

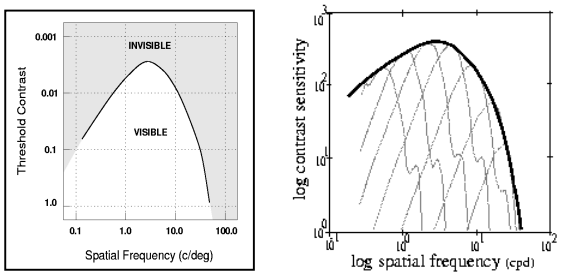


Figure 2: (a) The *contrast sensitivity function* measures the perceptibility of visual stimuli (sinusoidal gratings) in terms of their contrast and spatial frequency (cycles per degree). (b) The shape of the CSF is attributed to the aggregate response of multiple bandpass mechanisms in the visual system. Courtesy Martin Reddy, Mahesh Ramasubramanian.

the use of polygonal simplification, texture-based imposters, and some forms of image- and point-based rendering. We have applied perceptually driven interactive rendering to two such schemes: view-dependent polygonal simplification and the point-based renderer QSplat.

Below we briefly present some background on visual perception, followed by an overview of our framework. We illustrate this overview using QSplat, which was relatively straightforward to dovetail with perceptually driven rendering. We then turn to the application of our framework to polygonal simplification, which was more involved. We describe the underlying algorithm and the evaluation of contrast and spatial frequency induced by simplification, which is ultimately the heart of the problem. Finally, we consider our framework in relation to previous work on perceptually guided rendering, and close with some thoughts on future extensions and improvements.

2 The Contrast Sensitivity Function

A large body of perceptual psychology literature focuses on the perceptibility of visual stimuli. The simplest relation established in this literature is *Weber's law*, which predicts the minimum detectable difference in luminance between a test spot on a uniform visual field. Weber's law states that at daylight levels the threshold difference in luminance increases linearly with background luminance. Interesting scenes are not uniform, however, but contain complex frequency content. Outside a small frequency range, the threshold sensitivity predicted by Weber's law drops off significantly. Many perception studies have therefore examined the perceptibility of *contrast gratings*, sinusoidal patterns that alternate between two extreme luminance values L_{max} and L_{min}. Campbell first characterized the perceptibility of a contrast grating in terms of its contrast and spatial frequency [4]. Contrast grating studies use *Michaelson contrast*, defined as $(L_{max} - L_{min}) / (L_{max} + L_{min})$, and *spatial frequency*, defined as the number of cycles per degree of visual arc. The *threshold contrast* at a given spatial frequency is the minimum contrast which can be perceived in a grating of that frequency, and *contrast sensitivity* is defined as the reciprocal of threshold contrast. The *contrast sensitivity function* (CSF) plots contrast sensitivity against spatial frequency, and so describes the range of perceptible contrast gratings [Figure 2a].

Of course, most interesting images are more complex than the simple sinusoidal patterns used in contrast gratings. Campbell found that the perceptibility of complex signals could be determined by decomposing a signal into sinusoidal components using Fourier analysis [5]. In particular, if no frequency component of a signal is perceptible, the signal will not be perceptible.

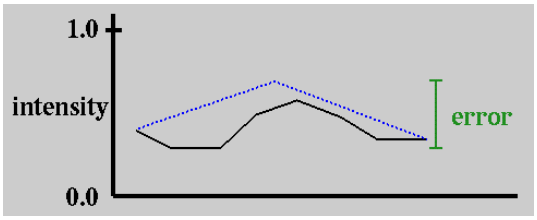


Figure 3: We use a conservative approximation of the change in intensity (error) induced by folding a node. Here the black line represents the original surface and the blue line the simplified surface.

Note that the CSF predicts the maximum perceptibility of a stationary grating at the center of view; many other factors can lower contrast sensitivity. Among these is *eccentricity*, or distance from the direction of gaze. The *fovea* is the region of highest sensitivity on the retina, occupying the central 1º or so of vision. *Visual acuity*, measured as the highest perceptible spatial frequency, is significantly lower in the visual periphery than at the fovea. The relationship between visual acuity and eccentricity, defined as angular distance from the fovea, was first characterized in humans by Rovamo and Virsu [25]. By extending our perceptual model to incorporate eccentricity, we can predict the visibility of peripheral features for use in gaze-directed rendering.

3 Overview

Our goal is to analyze LOD-based rendering algorithms within a principled perceptual framework inspired by the contrast grating studies described above. We map the change resulting from a local simplification operation to a *worst-case* contrast grating, meaning a grating with the most perceptible combination of contrast and frequency possibly induced by the operation. We apply the simplification only if we would not expect a grating with that contrast and frequency to be visible. Our hypothesis: if we can isolate the most perceptible frequency component possibly induced by a simplification operation, and determine that a contrast grating at that frequency would not be visible, we can perform the operation without perceptible effect.

3.1 Determining the Worst Case

To determine the worst-case frequency and contrast efficiently, we make some conservative simplifying assumptions. First, we observe that the peak contrast sensitivity occurs at approximately 2-4 cycles per degree, and that most local simplification operations on a complex model affect only much higher frequencies. We therefore assume that contrast at lower spatial frequencies is more perceptible than at higher frequencies (Section 6.1 describes how we adjust our perceptual model to ensure this assumption holds). The minimum frequency component of a region in the image spanning n degrees of the user's angular field of view is one cycle per $2n$ degrees. Put another way, the maximum wavelength needed to represent a region of the image is twice the maximum spatial extent of that region [Figure 3]. Consequently, finding the worst-case frequency induced by a simplification reduces to finding the screen-space extent of the affected region.

For the worst-case contrast, we determine a bound on the maximum change in luminance among all the pixels affected by the simplification. Put another way, the worst-case contrast of a

simplification operation is the maximum contrast between an image of the affected region at full resolution and an image of the region simplified. For 3-D models, there are two basic cases:

- The entire affected region lies interior to a surface that entirely faces the viewer. This is the simplest case: the contrast between the original region and the folded region is completely determined by the luminance of the local surface before and after the fold.
- The affected region includes a silhouette edge. This expands the possible contrast incurred by the simplification to include the portion of the scene *behind* the affected region, since simplifying the surface may expose a very bright or very dark feature occluded before simplification.

Consequently, silhouette regions of the object are simplified less aggressively—exactly the behavior we should expect in a perceptually driven simplification algorithm. Note, however, that even at these high contrast levels silhouette regions can still be simplified if they represent very fine details (high spatial frequencies) or are in the viewer's peripheral vision (high eccentricity).

3.2 An Empirical Perceptual Model

Many researchers have characterized the contrast sensitivity function. In early work, Kelly derived an abstract relationship for the perceptibility of sinusoidal gratings over a narrow range: $C_T = \alpha^2 e^{-\alpha}$ [14]. Here C_T represents the threshold contrast and α represents spatial frequency. The more accurate CSF curve given by Barten [2] is used in recent advanced global illumination algorithms, such as the physically based metric of Ramasubramanian, which also account for adaptation effects due to background illumination [23][3]. Modern perceptual theory attributes the shape of the CSF to the combined response curves of multiple bandpass mechanisms in the visual system, each processing only a small range of the visible spatial frequency spectrum [Figure 2b]. This multiscale visual processing can be emulated with a Laplacian pyramid for spatial decomposition [17]. Current perceptual rendering techniques also account for *contrast masking*, which represents the visual system's decreased contrast sensitivity in the presence of strong patterns. This can further increase the allowable error in an image [9][23][3].

Unfortunately, these sophisticated perceptual models, which employ the latest advances in understanding perception, are far too costly for the interactive framework we propose. In our framework, thousands of simplification operations must be considered every second, leaving less than a millisecond to evaluate the induced contrast and frequency. Clearly, we must forego the state-of-the-art perceptual models used in current global illumination work for a model that is simple, fast, and conservative.

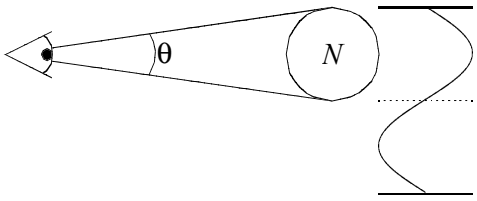


Figure 5: The lowest spatial frequency that can be affected by a node spanning θº of visual arc has one cycle per 2θº.

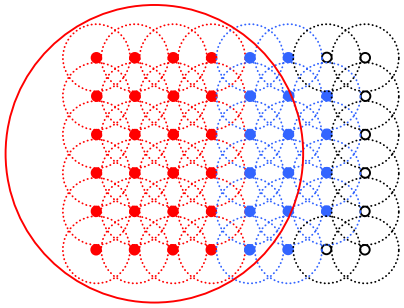


Figure 4: Estimating contrast in QSplat. Here the small circles represent leaf nodes (samples) with their bounding spheres. The sphere of an internal node **N** (drawn in red) must contain the spheres of all its descendents (also in red). To compute contrast induced by simplifying **N**, we must also account for nodes intersecting **N** (drawn in blue), since these nodes may be occluded by **N**'s splat.

To achieve simplicity and speed while still accounting for real-world factors that affect perception, such as ambient light, we chose to take an empirical approach. Recall our hypothesis: a simplification operation, mapped to a "worst-case" contrast grating, can be performed imperceptibly if that grating would not be perceptible. We build our perceptual model directly from contrast grating tests performed under the same conditions—room illumination, monitor, etc—under which our final system will run. A calibration procedure, detailed below, tests the ability of a user to detect contrast gratings, recording threshold contrast over a wide range of spatial frequency and eccentricity. We then build a lookup table from the resulting CSF curves and use linear interpolation at runtime to determine whether the user can perceive a given contrast at a given spatial frequency and eccentricity.

This model could certainly be improved, but we chose to focus on developing a framework for driving interactive rendering with a perceptual model, rather than on developing the model itself. Our empirical model is simple to implement and works well in practice; Figure 6 shows example CSF curves determined from a typical calibration procedure.

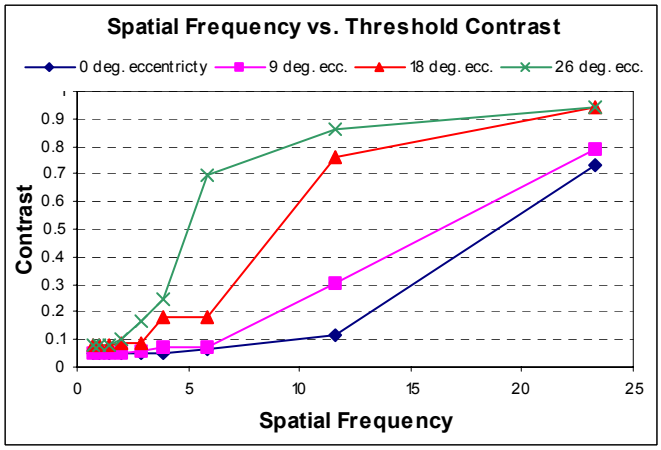


Figure 6: We use a simple model of contrast sensitivity based on an empirical calibration procedure. Shown here are results from one user's calibration.

4 Perceptually Driven Point Rendering

The QSplat point-rendering system lends itself nicely to our perceptual rendering framework. While Rusinkiewicz and Levoy describe many careful optimizations, tradeoffs, and design decisions that make QSplat exceptionally fast [26], at its heart it is a simple algorithm. Qsplat uses a hierarchy of bounding spheres for culling, rendering, and level of detail management. The leaf nodes in this hierarchy of spheres form a dense tiling of the original surface and typically represent samples from a 3-D scanning device. Internal nodes completely contain the spheres of their descendents, and average the color and normal of their descendents. The hierarchy is traversed in a view-dependent fashion: backfacing and invisible nodes are skipped and their descendents not traversed, while nodes below a certain size on screen are drawn (using a *splat*) without descending further. The core simplification operation in QSplat, then, is to draw a node splat rather than traverse its descendents.

4.1 Applying the Framework

To apply perceptually guided rendering to QSplat, we must evaluate the spatial frequency and contrast induced by a simplification. Since the sphere at each node completely contains all its descendents, we can bound the affected region of the image by the projected screen-space extent of the sphere. Since QSplat already computes this diameter, we need only convert it into the appropriate spatial frequency.

Determining Maximal Contrast

Given a worst-case spatial frequency for the change induced by folding a node, we next need to compute the worst-case contrast of that change. Recall that Michaelson contrast depends on L_{max} and L_{min}, the maximum and minimum brightness of the region affected by the fold operation. Brightness, or luminance, is measured in cd/m^2, so we must convert OpenGL vertex colors to actual luminance.

Simplifying a node **N** draws its splat rather than the splats of its descendents, but may also occlude nodes that do not descend from **N**. Though it is difficult to determine which nodes may be occluded in certain pathological cases, in practice we can account for occlusion by considering all nodes intersecting the bounding sphere of **N** [Figure 4]. The maximum contrast that can be caused by folding **N** is then determined by the color of **N** and the range of colors of nodes intersecting **N**. We then calculate the induced Michaelson contrast by converting all colors to luminance, and comparing the minimum and maximum luminance of the original surface to the luminance of the simplified node. We can efficiently compute this information during preprocessing, quantize contrast to 8 bits, and store it with the node.

Figure 7: Gaze-directed rendering. Left: At high quality QSplat uses 2.9 million points to render Lucy. Right: With the user's gaze 29° away, our system imperceptibly simplifies the model to 0.8 million points. Inset: at this distance all nodes are simplified. Originally rendered at 1600x1200 on a 23" monitor.

Of course, this precomputed contrast becomes invalid if the node lies on the silhouette. QSplat stores a normal and *normal cone* [28] with each node, and evaluates whether the node represents a surface region that faces entirely away from the viewer (for backface culling) or entirely towards the viewer (to disable backface culling). We can thus identify silhouette nodes with no additional computation, since all nodes that are neither entirely backfacing nor entirely frontfacing might lie on the silhouette. As with contrast, we need to modify the preprocessing to account for contained nodes when computing normal cones. For simplicity, and to avoid extra storage, we assume that simplifying any silhouette node could induce the maximum possible contrast.

Putting It All Together

Given the worst-case contrast and spatial frequency, both readily available at run-time, our perceptually driven algorithm for QSplat rendering is nearly as simple as the original:

```
TraverseHierarchy(node)
{
    if (node outside view frustum)
        skip this branch
    if (node is backfacing)
        skip this branch
    // Worst-case frequency has period twice sphere diameter:
    frequency = 1/(4*node->radius);
    if (node is frontfacing)
        contrast = node->contrast;
    else
        // Node is on silhouette, use maximum possible contrast
        contrast = maxContrast;
    if (IsPerceptible(frequency, contrast)
        // Simplifying node would be perceptible, keep going
        foreach child of node
            TraverseHierarchy(child);
    else
        // Can simplify node imperceptibly
        DrawSplat(node);
}
```

Here spatial frequency is represented in cycles per radian. The function `IsPerceptible()` looks up the given contrast and frequency and returns `TRUE` if simplifying the node to a single splat might be perceptible. In practice, we store the lookup table in units of wavelength rather than frequency to avoid the extra divide. If gaze-directed rendering is desired, the traversal function calculates angular distance from the gaze point to the node, and passes this value to `IsPerceptible()` as eccentricity.

Figures 1 and 7 demonstrate our perceptually driven QSplat system.

5 Perceptually Driven Polygonal Simplification

Regulating scene complexity and rendering time by simplifying small or distant objects was first proposed in Clark's seminal 1976 paper [6]. The basic approach described there remains the most common approach today: create several versions of each object at progressively coarser levels of detail in a preprocess, and choose at run-time which version (called *LODs*) will represent the object. The past decade has seen a flurry of research into *polygonal simplification*: algorithms for generating coarse LODs from full-resolution polygonal models and for managing which LODs replace each object. Several recent surveys examine the field of polygonal simplification [11][22][19]; in Section 7 we examine some algorithms relevant to perceptually based rendering.

One difficulty with traditional LOD-based approaches is their reliance on a few discrete levels of detail to represent each object. This limits the degree to which perceptual metrics can be applied, since the entire object must be simplified uniformly. For example, silhouette details tend to be more perceptible than interior details because of higher contrast, so the entire object must be treated as if it were on the silhouette. Similarly, if the user's eye rests on any portion of the object, a system that accounts for eccentricity must treat the entire object as if it were under direct scrutiny. The worst-case assumptions of per-object LOD can severely handicap perceptually guided polygonal simplification.

5.1 View-Dependent Simplification

View-dependent simplification methods offer a solution. Rather than calculating a series of static levels of detail in a preprocess, view-dependent systems build a dynamic data structure from which a desired level of detail may be extracted at run time. Objects in a view-dependent algorithm may span multiple resolutions, solving the worst-case behavior of traditional LOD. For example, portions of the object under the viewer's gaze can be represented at higher fidelity than portions in the peripheral vision, and regions of the object moving slowly across the visual field could utilize higher resolution than fast-moving regions.

Several researchers have independently proposed view-dependent algorithms, including Hoppe, Luebke, and Xia [12][18][29]. These algorithms share a common feature: each is a hierarchy of *vertex merge* operations that can be applied or reversed at run-time. Our chief contribution is a method for evaluating the perceptibility of a vertex merge operation, using factors such as contrast, spatial frequency, and eccentricity. We have implemented our system using *VDSlib*, a public-domain library based on the view-dependent simplification framework of Luebke [20]. VDSlib allows users to plug in custom callbacks for building, culling, simplifying, and rendering the vertex tree. We first augment the nodes of a VDSlib vertex tree with data specific to our perceptual simplification process, such as the contrast induced by a fold operation and the normal mask used for silhouette detection. Then at run time, our callback examines nodes, using contrast, spatial frequency, and possibly eccentricity to decide whether VDSlib should fold the node. Before describing the details of this process, we briefly review the VDSlib algorithm and notation.

The main data structure of VDSlib is the *vertex tree*, a hierarchical clustering of vertices. Vertices from the original model are grouped with nearby vertices into clusters, then the clusters are clustered together, and so on. Leaf nodes of the tree represent a single vertex from the original model; interior nodes represent multiple vertices clustered together, and the root node represents all vertices from the entire model, merged into a single cluster. In VDSlib parlance, a node **N** *supports* a vertex **V** if the leaf node associated with **V** descends from **N**. Similarly, **N** *supports* a triangle **T** if it supports one or more of the corner vertices of **T**. The set of triangles in the model supported by a node is called the *region of support* of the node.

Each node stores a representative vertex called the *proxy*. For leaf nodes, the proxy is exactly the vertex of the original model that the node represents; for interior nodes, the proxy is typically some average of the represented vertices. *Folding* a node merges all of the vertices supported by that node into the node's single proxy vertex. In the process, triangles whose vertices have been merged together are removed from the scene, decreasing the overall polygon count. Since folding a node is the core simplification operation of VDSlib, to apply our perceptual framework we must evaluate the contrast and spatial extent of the change in the rendered image induced by a fold.

5.2 Applying the Framework: VDSlib

The effect of folding a node in VDSlib is more complex than the effect of drawing a splat in QSplat. As the vertices and triangles supported by the node merge and shift, features in the image may shrink, stretch, or disappear completely. Shifting triangles on the visual silhouette may expose previously occluded features. To analyze the effect of folding a node, we should consider all of these changes. One possibility, recently demonstrated by Lindstrom and Turk for static LOD generation, is to render the scene before and after the operation and analyze the resulting images [16]. At present, however, the requisite rendering and image processing appears too expensive for dynamic simplification. Instead, we want a conservative worst-case bound on the changes in the image caused by folding the node. Since our goal is to evaluate a hypothetical change *at least* as perceptible as any changes that folding actually incurs, we consider the removal of a feature with worst-case size and contrast.

Spatial Frequency: Estimating Extent

Again, the minimum frequency induced by a simplification is determined by the spatial extent of the resulting change in the image. Notice that features in the image affected by a fold consist of triangles connecting vertices involved in the fold. The largest feature that can be removed or exposed by geometric distortion upon folding a node is therefore constrained by the distance vertices move during the fold. Thus, the problem of computing the minimum frequency induced by folding a node reduces to computing the screen-space extent of all vertices supported by the node.[1] As with QSplat, we use bounding spheres to estimate this extent, associating with each node a tight-fitting sphere that contains all vertices in the node's region of support. The angular extent of these bounding spheres, as seen from a given viewpoint, can be calculated very quickly. The minimum frequency affected by folding a node is then one cycle

[1] Technically, this holds when the model is flat shaded; for Gouraud-shaded models, adjacent vertices should also be included. However, we have not found this necessary in practice.

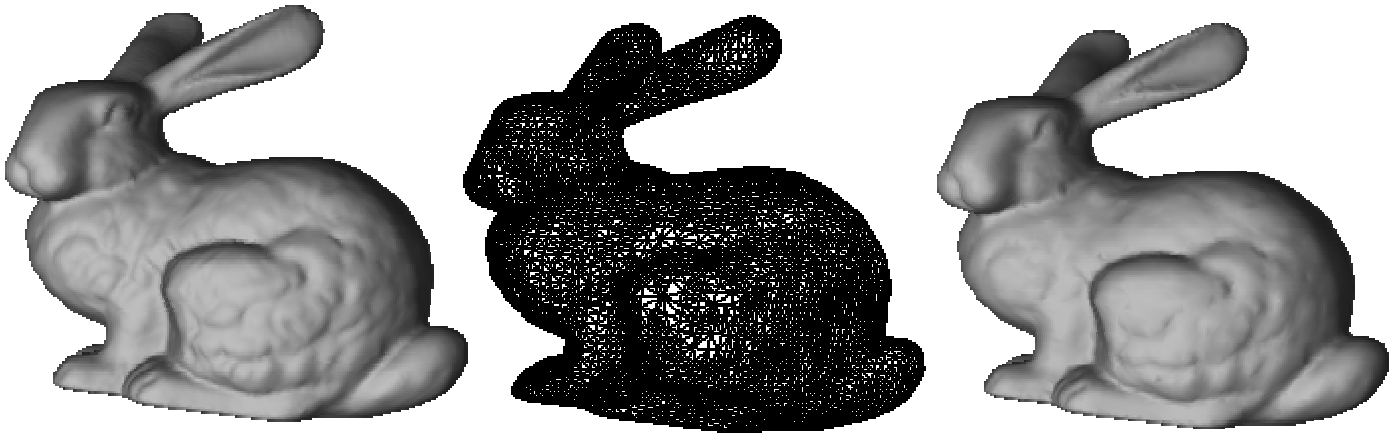


Figure 8: The original Stanford Bunny model (69,451 faces) and a simplification by our perceptually driven system (29,866 faces). In this view the user's gaze is 29° from the center of the bunny…equivalent to looking at the page number on this page from a distance of 10". Note that the silhouette is well preserved, along with strong interior details (the line of the haunch, the shape of the eye, etc.) while subtle bumps on the surface are simplified.

per two degrees of angular extent spanned by the node's bounding sphere [Figure 4].

Contrast: Estimating Intensity Change

Determining the exact contrast induced by folding a node would be as expensive as rendering the unfolded geometry. Instead, we obtain a conservative lower bound by comparing the intensities of all the vertices the node supports in the original model with the intensities of the vertices in the simplified surface [Figure 3]. The greatest difference between the intensities of the surface vertices before folding and after folding bounds the maximum contrast between the simplified surface and the original surface, since in a Gouraud-shaded model extremes of intensity always occur at the vertices. This conservative test may overestimate the contrast induced by folding, but will not underestimate it.

When the node's region of support includes a silhouette, we must be even more conservative. Lacking knowledge about what lies behind the model, we must assume the worst: moving a silhouette edge might expose the darkest or brightest object in the scene, including the background. Hence we must compare the range of vertex intensities of the node's region of support against the brightest and darkest intensities in the scene, and use the maximum possible difference in intensity for calculating the contrast induced by the fold.

Determining Silhouette Nodes

Since nodes affecting silhouette edges must be treated differently, we require an efficient method for identifying such nodes. For a given view, we define *silhouette node*s as those nodes supporting both front-facing and back-facing triangles in the original mesh. We initially employed Shirman's *cone of normals* approach [28], used by both Luebke and Hoppe [18][12], to determine silhouette nodes. Unfortunately, the cone of normals sometimes proves overly conservative, especially in models with sharp edges; we found that too many interior polygons were being classified as belonging to silhouette nodes. Instead, we used a bitwise approach inspired by the rapid backface culling technique of Zhang and Hoff [30]. We map the Gauss sphere of normal space to a *normal cube* whose faces are tiled into cells, in effect quantizing the space of normals. Each node in the model stores a *normal mask*, a bit vector representing the normals of all its supported triangles. A bit in the mask is set if a triangle normal falls within the corresponding cell of the normal cube.

The accuracy of the normal mask is bounded only by the number of cells, which depends on the length of the bit vector. This improves significantly over the cone of normals, which can greatly overestimate the range of normals. Normal masks are efficient to compute, since they can be propagated up the vertex tree using bitwise-OR operations. Testing whether the node might lie on the silhouette can also be made very efficient by precomputing two bitmasks, representing the space of normals that might be backfacing and frontfacing, respectively. A node may be on the silhouette if its normal mask overlaps with both the frontfacing and the backfacing bitmasks. The test to classify a silhouette node therefore reduces to two bitwise-AND operations, whose cost depends on the length of the bit vector. We chose 48 bytes (64 bits per face of the normal cube) for accuracy, but if memory is at a premium, fewer bits could be used to trade off accuracy for compactness.

5.3 Perceptually Guided Best-Effort Rendering

Imperceptible simplification makes a guarantee about the visual fidelity of the simplified scene. Often, however, a guarantee about the complexity, and thus rendering time, is desired instead. VDSlib supports *triangle budget simplification*, which allows the user to specify how many triangles the scene should contain. Using a user-specified run-time error metric, VDSlib then minimizes the total error induced by all folded nodes within this triangle budget constraint. Internally, VDSlib performs triangle budget simplification using a greedy approach. A priority queue of boundary nodes is sorted by induced error, as evaluated by a user-supplied callback. The node **N** with the greatest error is unfolded, adding some triangles to the scene. **N** is then removed from the priority queue and its children inserted back into the queue. This process iterates until unfolding the top node of the queue would exceed the triangle budget. [2]

For principled best-effort rendering, then, we must generate a sound perceptual measure of the error introduced by folding a node. The key is to recast our metric for evaluating the

[2] Note that this assumes error decreases monotonically; folding a node should not induce more error than folding its parent.

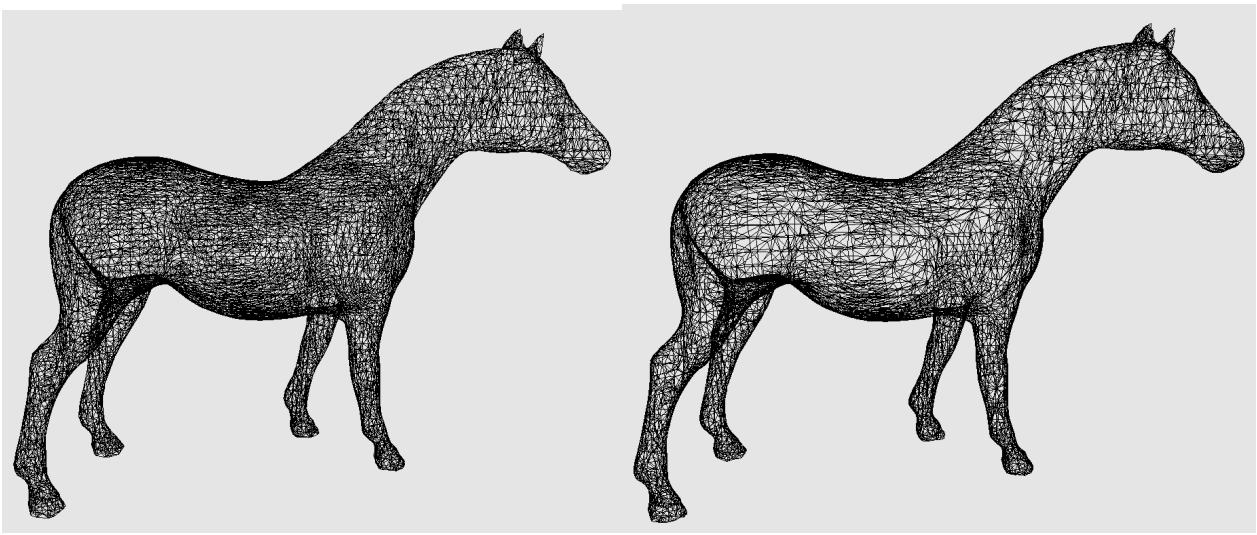


Figure 9: Perceptually driven best-effort simplification. Both images show the horse model (originally 96,966 faces) reduced to 18,000 faces using triangle budget rendering in VDSlib. Left, the default VDSlib error metric uses screenspace node size, leading to unnecessarily even tesselation. Right, our perceptually driven metric uses fewer polygons in interior and low-contrast regions.

perceptibility of fold operations. Rather than a binary perceptible/not perceptible decision, we need a scalar to express *how* perceptible a fold operation could be. We chose to cast the question in terms of distance: how far would the viewer have to be from the screen before the node could be folded imperceptibly? The answer can be computed from our current perceptual model, in effect by inverting our lookup tables. Rather than computing the spatial frequency of a node and looking up the threshold contrast at which folding is perceptible, we use the precomputed contrast induced by folding and look up the threshold spatial frequency. From this we can compute the distance at which folding the node would be perceptible, and sort nodes in the priority queue by this distance.

To recap, we order folds based on the viewing distance at which they become perceptible. This provides a convenient framework for best-effort triangle budget simplification, and an intuitive physical measure of the fidelity achieved: after simplifying to the user-specified number of triangles, the system can report the distance at which that simplification should be imperceptible.

5.4 Results

All results given are on an 866 MHz Pentium III computer with NVidia GeForce^2 graphics. Figures 8 and 9 show models simplified with our perceptually driven algorithm. Since we are guaranteeing imperceptible simplification, the reductions in polygon count may seem modest. However, these results and the user study below clearly show that perceptually driven simplification can reduce model complexity without visual effect.

Perceptually driven best-effort rendering may be of more use to many 3-D applications. Figure 9 compares our results to VDSlib's built-in triangle budget rendering, which orders fold operations only by size of the node. Note that the perceptually driven algorithm preserves more triangles near silhouettes, and simplifies more aggressively in regions of low contrast.

6 Implementation and Evaluation

6.1 Calibration: Building a Perceptual Model

The goal of our calibration step is to build a simple empirical model of threshold contrast across different values of spatial frequency and eccentricity. Building this model at runtime allows us to account for factors that affect perception but are not likely to vary over the course of a viewing session, such as room illumination and visual acuity of the individual user. During the calibration, the user fixates on a target while a grating fades in, slowly increasing in contrast. The user is instructed to press the mouse button when something becomes visible. To verify that the user actually saw the grating, he then clicks on it (without looking away from the target). By varying the spatial frequency and eccentricity of the gratings presented to the user, we find the threshold contrast across these parameters [Figure 6]. From the data sampled during calibration we build the threshold contrast lookup table used at runtime.

Recall that our perceptual model assumes low frequencies are more perceptible than high frequencies, so that we can use node spatial extent to determine a worst-case contrast. This assumption holds true for most simplification operations we are concerned with, which extend a few pixels at most. To ensure that we make a conservative choice, we modify our lookup table to effectively clamp the threshold contrast below the most sensitive frequencies.

We must also calibrate the monitor, in order to translate 24-bit OpenGL color values into luminance values (cd/m^2) used by our definition of contrast. This step involves measuring with a photometer the light levels produced by different red, green, and blue OpenGL intensities, and building a lookup table from the results. Fortunately, this part of the calibration need be repeated only as often as monitor drift warrants. If full precision were not necessary, simple gamma correction would suffice.

6.2 User Study: Evaluating Imperceptibility

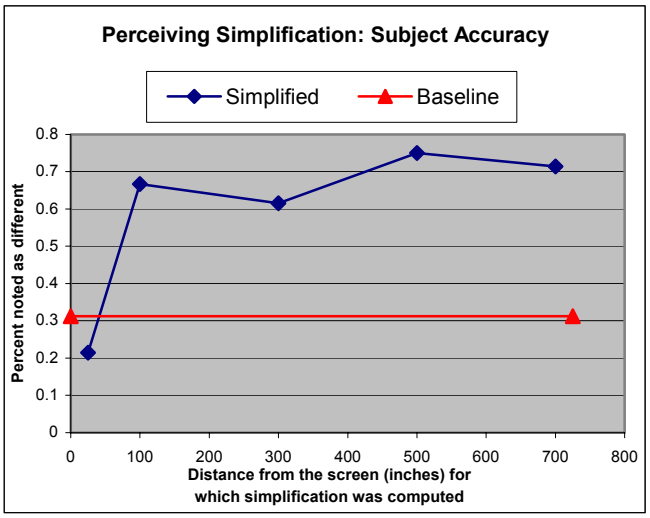
We performed a user study to evaluate our system more formally, determining whether our algorithm can indeed produce a simplification imperceptible from the original model. The study tested whether subjects could perceive the difference between a rendering of a full-resolution model and a rendering of a model simplified with our algorithm. If our hypothesis holds, a subject's ability to discern the simplification will be no better than chance.

As a control, we also evaluated the ability of subjects to discern simplifications that our model predicts could be visible. To this end, we randomly interspersed trials in which the "imperceptible" simplification was calculated for an incorrect field of view. Since the field of view varies with the distance δ of the user to the monitor, this amounts to calculating simplifications for incorrect viewing distances. For example, a simplification that assumes the user is ten times further from the screen than in reality will probably be visibily different than a simplification computed for the correct distance. The viewing distance δ in these trials was chosen so that simplifications should range from imperceptible (δ = correct distance to screen) to clearly perceptible.

The study consisted of 4 subjects, each of whom performed 200 trials. During each trial, the subjects fixated on a target (a short line segment) in the center of the screen. They were then shown the same 3-D object twice in succession, identical in all parameters except resolution. 25% of the trials displayed the object twice at full resolution, while another 25% displayed once at full resolution and once using our imperceptible simplification. For the remaining trials, one was presented at full resolution and the other was presented at a reduced resolution computed using "imperceptible" simplification for a random viewing distance.

Objects were displayed for 1 second and separated by 500 milliseconds, with a neutral grey background before, after, and between scenes. When the second object finished displaying, the subject pressed a key to indicate whether they thought the models differed. To avoid subject fatigue, the next trial did not begin until 500 milliseconds after the subject had pressed a key.

After a practice session of 20 trials, each subject performed 200 trials in a continuous session. Subjects viewed 3 models (bunny, horse, rhino) from 40 random viewpoints for each viewing distance used by the simplification algorithm. Viewing parameters were chosen so that the subject viewed the object at randomly distributed orientations from randomly distributed directions. Object size was randomly chosen such that the visual angle subtended by the object was uniformly distributed from 5° to 60°. The screen occupied approximately 46° of the subject's field of view, so that in some views the object filled most of the screen. When adjusting viewing distance, the value was chosen from 1-50 times the actual distance of the viewer from the screen. We picked these values to span the range between clearly perceptible and completely imperceptible simplification.



Each subject reported normal eyesight, some with corrective lenses. Subject accuracy is plotted against distance δ. *Baseline* represents the willingness of subjects to report a difference between the models when none existed. As the graph shows, our system does produce imperceptible simplification: at the correct viewing distance (25 inches), subject accuracy, defined as ability to perceive simplification, is no better than baseline.

7 Previous Work

7.1 Perceptually Based Offline Rendering

Many researchers have worked on perceptually based rendering algorithms; Bolin and Meyer [3] and Ramasubramanian et al [23] provide nice surveys of the field. These algorithms take advantage of the limitations of human vision to avoid rendering computation where the result will be imperceptible. Unlike our work, which seeks to accelerate interactive rendering, almost all previous perceptually based rendering approaches have addressed realistic offline rendering approaches such as ray and path tracing. Since image creation times in such approaches are typically measured in seconds or minutes, these algorithms are able to use very sophisticated perceptual models.

The state of the art in perceptually guided realistic rendering is well exemplified by the perceptually based physical error metric of Ramasubramanian et al [23]. Their perceptual model combines threshold sensitivity for varying illumination, contrast sensitivity with multi-scale spatial frequency sensitivity, and visual masking to predict the maximum imperceptible change in luminance at a given pixel in the rendered image. This advanced model, which incorporates the latest advances in our understanding of the visual system, is costly to evaluate by interactive rendering terms. Despite a novel framework that does not require reevaluation at every stage of the progressive illumination computation, the authors report that evaluating the model for a 512x512 image required about 12 seconds on a 400 MHz CPU.

7.2 Perceptually Based LOD Selection

Comparatively few systems have attempted to guide interactive rendering with explicit perceptual metrics. Funkhouser and Sequin's system for dynamic LOD selection uses a cost-benefit estimate to pick the best levels of detail within a specified time budget [10]. LOD benefit is assigned heuristically, based primarily on an object's screen-space size. Their system also takes into account eccentricity and *velocity*, the speed at which the image of an object moves across the retina. Lacking eye or head tracking, the user's gaze is assumed to lie in the center of the screen; lacking accurate perceptual models, the effects of velocity and eccentricity are controlled with sliders set by the user. Though the use of these factors is ad hoc, this important work

introduced the notion of perceptually guided metrics to drive LOD selection.

Ohshima et al described a system for gaze-directed stereoscopic rendering [21]. Although the paper mentions an eye-tracking system in progress, the results were gathered using head-tracked viewing direction to approximate gaze direction. Their system uses eccentricity, velocity, and convergence to guide selection of precomputed LODs. The equations used to model the three perceptual effects, and the method for combining all three effects to choose an LOD, appear to have been determined empirically. Thus their algorithm, while clearly demonstrating the potential of a gaze-directed approach, still employs a fundamentally heuristic model of the visual system.

Reddy was the first to attempt an LOD selection system guided throughout by a principled perceptual model [24]. Using images rendered from multiple viewpoints, Reddy analyzes the frequency content of objects and their LODs. A model of the visual acuity, defined as highest perceptible spatial frequency, guides LOD selection. If a high-resolution and a low-resolution LOD differ only at frequencies beyond the visual acuity of the viewer, those differences are imperceptible and the low-resolution LOD may be used. Reddy analyzes the frequency content of LODs, rendering each from several directions, and models the decrease in visual acuity with eccentricity and velocity to decide which LOD to use.

8 Summary And Discussion

Perceptually guided interactive rendering is a broad and difficult topic. Our system shows the feasibility and potential of imperceptible view-dependent simplification, but many avenues for further research remain. Below we summarize our contribution and results, and address what we see as the most pressing and interesting directions for future work.

8.1 Summary

We have demonstrated a novel approach to accelerating interactive rendering that is directly driven by perceptual criteria. Our principle contribution is a practical framework for perceptually guided interactive rendering that equates local simplification operations to worst-case contrast gratings whose perceptibility we can evaluate. We have shown an application of the framework to the QSplat point-rendering system, and demonstrated it in depth in the context of view-dependent polygonal simplification. Our approach addresses several interesting problems, including silhouette preservation and imperceptible simplification. An optional gaze-directed component uses eye tracking to obtain further simplification by reducing fidelity in the viewer's peripheral vision.

8.2 Improving the Current System

We see several opportunities to improve the current system. Incorporating dynamic lighting into the contrast calculation is an obvious extension. The difficulty lies in determining the possible range of intensities across the original surface without having to render that surface. Since lighting calculations depend on the surface normal, we might use the current normal masks, which bound the space of normals subtended by each node's region of support, to compute the minimum and maximum intensities of that region under the given lighting conditions. Accounting for non-directional light sources complicates the problem, but a solution certainly appears feasible.

Our estimate of the spatial frequency induced by a fold is overly conservative. We currently treat the worst case by assuming the entire screenspace extent of the node changes; the resulting spatial frequency bounds the minimum frequency change induced. In practice, however, the actual frequencies affected by a fold are typically higher (and hence less perceptible) than this worst-case estimate. A tighter estimate of spatial frequency would, we suspect, greatly improve the amount of simplification possible at a given contrast. For example, we currently use spheres to bound each node's region of support, which can overestimate the change caused by folding. A more direct measure of surface distortion, such as those used by Hoppe [12] or Cohen [7], might allow considerably more aggressive imperceptible simplification.

8.3 Extending the Perceptual Model

Like other perceptually based rendering algorithms to date (e.g., that of Ramasubramanian [23]), we base our decisions on the viewer's ability to perceive static stimuli. In animated or interactive rendering, we should also account for *temporal contrast sensitivity*: the ability to see a sudden change or flicker when a more gradual change would be invisible. In practice, our algorithms do not introduce visible flicker, but we could gaurantee this with a more sophisticated perceptual model that accounted for temporal contrast sensitivity. Such a model could prevent folds that would cause a visible "pop", or regulate a transition-softening technique such as alpha blending or Hoppe's *geomorphs* [12]. Unfortunately, temporal contrast sensitivity is less well studied than the static case, and incorporating current psychophysical models into our framework seems difficult. More research is needed in this area.

One significant factor affecting perceptibility of features is *contrast masking*, in which the presence of a high-contrast pattern can decrease sensitivity to features at other frequencies. For example, subtle discolorations on a brick wall may be less noticeable than on a uniformly colored brick-red wall. In polygonal models high-frequency patterns such as bricks are often represented using texture maps, whose frequency components could be computed in advance. In future work we hope to exploit contrast masking by patterns in texture maps for more aggressive simplification.

The velocity of a feature across the visual field also affects perception of detail. Consider, for example, a foreground object moving rapidly across a static background. If the user's eye tracks the moving object, the background will have high velocity and be simplified more aggressively. However, if the user's gaze remains fixed on the background, the moving object has high velocity and may be simplified. The perceptibility of moving contrast gratings may be related to that of static contrast gratings by a scaling function [15], so it should be straightforward to incorporate node velocity with respect to gaze into our folding criteria. Experiments by Reddy indicate the potential of using eccentricity and velocity together to guide LOD management [24], and integrating retinal velocity into our perceptual metrics seems an obvious and very promising avenue for future work.

8.4 Applicability of Gaze-Directed Rendering

Gaze-directed rendering is a powerful concept with some clear limitations. Accurately monitoring the user's gaze requires tracking the eye, but eye tracking is still emerging as a commodity technology: some current systems are fast enough, accurate enough, robust enough, and posses low enough latency for our application, but no existing eye tracker meets all of these needs at once. For example, our current system is quite fast and accurate, but restricts the user's head to a small volume, requires a room without sunlight, and involves a short calibration step before use.

It seems likely that eye-tracking technology will improve, eliminating these limitations. However, even without eye tracking gaze-directed rendering may still be a viable option. When allowed free range of head motion, user gaze is almost always restricted to ±15º of head direction [1]. We can thus substitute head direction for eccentricity in our system simply by subtracting a 15º error term. For multi-screen wide-angle displays, such as video wall or CAVE™ systems, head-tracked gaze-directed rendering may be a very attractive option.

Our system could easily handle multiple viewers by calculating the eccentricity of a node as the minimum distance to any viewer's gaze direction. Obviously, multiple viewers can reduce the impact of gaze-directed rendering, since viewers might examine different parts of the display at once. Such a scenario increases the demand on the eye-tracking system and limits the degree of simplification possible. In a multi-screen wide-angle display scenario, however, most of the scene will still be outside any viewer's fovea and therefore still eligible for aggressive simplification. Even with head tracking, which forces a more conservative estimate of eccentricity, we suspect that gaze-directed rendering will prove a powerful technique for managing rendering complexity in such situations.

9 Acknowledgements

Many of our models are provided courtesy of the Stanford 3-D Scanning Repository. The authors would like to thank the maintainers of the repository for making these excellent models public.

10 References

[1] Barnes, G. "Vestibulo-ocular function during coordinated head and eye movements to acquire visual targets", *Journal of Physiology*, 287, (1979).

[2] Barten, Peter. "The Square-Root Integral (SQRI): A new Metric to Describe the Effect of Various Display Parameters on Perceived Image Quality", In *Human Vision, Visual Processing, and Digital Display*, vol. 1077, Proceedings SPIE (1989)

[3] Bolin, Mark. and G. Meyer. "A Perceptually Based Adaptive Sampling Algorithm", *Computer Graphics*, Vol. 32 (SIGGRAPH 98).

[4] Campbell, F. and Gubisch, R. "Optical Quality of the Human Eye", *Journal of Pysiology*, 186 (1966)

[5] Campbell, F.W. and Robson, J.G. "An Application of Fourier Analysis to the Visibility of Contrast Gratings", *Journal of Physiology*, 187 (1968)

[6] Clark, James H. "Hierarchical Geometric Models for Visible Surface Algorithms," *Communications of the ACM*, Vol. 19, No 10, pp 547-554.

[7] Cohen, J, M. Olano, and D. Manocha. "Appearance-Preserving Simplification," *Computer Graphics*, Vol. 32 (SIGGRAPH 98).

[8] Cosman, M., and R. Schumacker. "System Strategies to Optimize CIG Image Content". *Proceedings Image II Conference* (Scotsdale, Arizona), 1981.

[9] Ferdwada, James, S. Pattanaik, P. Shirley, and D. Greenberg. "A Model of Visual Masking for Realistic Image Synthesis", *Computer Graphics*, Vol. 30 (SIGGRAPH 96).

[10] Funkhouser, Tom, and C. Sequin. "Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments", *Computer Graphics*, Vol. 27 (SIGGRAPH 93).

[11] Heckbert, Paul, and M. Garland. "Survey of Polygonal Surface Simplification Algorithms", SIGGRAPH 97 course notes (1997).

[12] Hoppe, Hughes. "View-Dependent Refinement of Progressive Meshes", *Computer Graphics*, Vol. 31 (SIGGRAPH 97).

[13] Hutchinson, Thomas E. "Human-Computer Interaction Using Eye-Gaze Input", *IEEE Transactions on Systems, Man, and Cybernetics* Vol 19, No. 6 (Novermber 1989).

[14] Kelly, D.H. Spatial Frequency Selectivity in the Retina, *Vision Research*, 15 (1975)

[15] Koenderink, J.J. *et al.* "Perimetry of contrast detection thresholds of moving spatial sine wave patterns", *Journal of the Optical Society of America*, 68 (1978)

[16] Lindstrom, P. and Turk, G. "Image-Based Simplification", To appear in *ACM Transactions on Graphics*. Available as technical report GIT-GVU-99-49, Georgia Institute of Technology (1999).

[17] Lubin, Jeffery. "A Visual Discrimination Model for Imaging System Design and Evaluation", *Vision Models for Target Detection and Recognition*, E. Peli, ed. World Scientific (1995).

[18] Luebke, David, and C. Erikson. "View-Dependent Simplification of Arbitrary Polygonal Environments", *Computer Graphics*, Vol. 31 (SIGGRAPH 97).

[19] Luebke, David. "A Developer's Survey of Polygonal Simplification Algorithms", To appear in *IEEE Computer Graphics & Applications* (March 2001). Available as technical report CS-99-07, University of Virginia.

[20] Luebke, David. See *http://vdslib.virginia.edu*.

[21] Oshima, Toshikazu, H. Yamammoto, and H. Tamura. "Gaze-Directed Adaptive Rendering for Interacting with Virtual Space", *Proceedings of VRAIS 96* (1996).

[22] Puppo, Enrico, and R. Scopigno. "Simplification, LOD and Multiresolution—Principles and Applications", *Eurographics '97 Tutorial Notes*, PS97 TN4 (1997).

[23] Ramasubramanian, Mahesh, S. Pattanaik, and D. Greenberg. "A Perceptually Based Physical Error Metric for Realistic Image Synthesis", *Computer Graphics*, Vol. 33 (SIGGRAPH 99).

[24] Reddy, Martin. "Perceptually-Modulated Level of Detail for Virtual Environments", Ph.D. thesis, University of Edinburgh, 1997.

[25] Rovamo, J. and Virsu, V. "An Estimation and Application of the Human Cortical Magnification Factor", *Experimental Brain Research*, 37 (1979)

[26] Rusinkiewicz, S. and Levoy, M. "QSplat: A Multiresolution Point Rendering System for Large Meshes", *Computer Graphics*, Vol. 34 (SIGGRAPH 2000).

[27] Savoy, R.L. and McCann, J.J. "Visibility of low-spatial-frequency sine-wave targets: Dependence on number of cycles", *Journal of the Optical Society of America*, 65 (1975)

[28] Shirman, L., and Abi-Ezzi, S. "The Cone of Normals Technique for Fast Processing of Curved Patches", Computer Graphics Forum (*Proc. Eurographics '93*) Vol 12, No 3, (1993), pp 261-272.

[29] Xia, Julie and Amitabh Varshney. "Dynamic View-Dependent Simplification for Polygonal Models", *Visualization 96*.

[30] Zhang, Hansong, and K. Hoff. "Fast Backface Culling Using Normal Masks", *Proceedings of ACM Symposium on Interactive 3D Graphics* (1997).