# Perceptual Evaluation of an Interactive Forest Walk-through

Paulius Micikevicius
Computer Science Department
Armstrong Atlantic State Univ.
paulius@drake.armstrong.edu

Charles E. Hughes
School of Computer Science &
Media Convergence Laboratory
University of Central Florida
ceh@cs.ucf.edu

J. Michael Moshell
School of Digital Media &
School of Computer Science
University of Central Florida
j.m.moshell@dm.ucf.edu

Valerie Sims
Hana Smith
Psychology Department
University of Central Florida
vsims@pegasus.cc.ucf.edu
smthda2@yahoo.com

## 1. Introduction

Recent advances in CPU and commodity graphics hardware performance as well as head-mounted display (HMD) resolution allow for rendering of more realistic virtual environments. While in the past virtual environments were geometrically simple, complex geometry and imagery are required for compelling Mixed Reality (MR) experiences which aim to seamlessly blend virtual and real objects. The ever-increasing scene complexity overwhelms even the fastest modern hardware and the trend will remain at least until real-time photo-realistic rendering is attained. Thus, techniques for efficiently managing the scene complexity at run-time are necessary to ensure interactive frame rates. In this paper we describe a visibility-based level of detail (LOD) selection method, along with a hierarchical LOD model. The framework is described in the context of a real-time forest walk-through (intended for viewing through a video see-through HMD), however it is equally applicable to other virtual environments.

Human-perception experiments are critical to achieving high visual quality in the walk-through. Switching between levels of detail often results in a visual pop, which can interrupt a virtual experience, causing attentional shifts, reallocation of resources, and a reduced sense of presence in the environment. The severity of popping artifacts for tree LODs was evaluated by experiments with a diverse population of human subjects. Experimental data were then used to adjust a number of thresholds in the implementation in order to minimize the visual popping artifacts.

## 2. Tree Model

In our system we generate trees using L-systems [13], which were chosen because they provide a compact representation as well as the methods necessary for advanced biological simulation. We have modified the stochastic L-system 5, described by Prusinkiewicz *et al.* [14], to generate trees for the forest walk-through implementation. The rules of the system as well as our modifications are briefly described below.

$$
\begin{aligned}
\omega&: \quad FA(1)\\
p_1&: \quad A(k) \rightarrow /(\varphi)\,[+(\alpha)FA(k+1)] - (\beta)FA(k+1): \\
&\qquad\qquad \min\{1,\ (2k+1)/k^2\}\\
p_2&: \quad A(k) \rightarrow /(\varphi) - (\beta)FA(k+1): \\
&\qquad\qquad \max\{0,\ 1-(2k+1)/k^2\}
\end{aligned}
$$

The initial string is specified by the *axiom* $\omega$, which consists of two modules. Module $F$ is interpreted and rendered as a branch segment. Module $A()$ is used for "growing" the tree and has no graphical interpretation (the integer in the parentheses denotes the number of times rewriting rules have been applied). Modules +, – denote rotation around the z-axis, while / denotes rotation around the y-axis. The angles for the rotations are specified in the parentheses ($\alpha = 32°$, $\beta = 20°$, $\varphi = 90°$). There are two possibilities when rewriting module $A(k)$. The first rewriting rule $p_1$ produces two branches with probability $\min\{1, (2k+1)/k^2\}$, while $p_2$ produces a single branch segment. For more details on L-systems and their interpretation refer to [13] and [14].

### 2.1. Tree LOD Model

We propose a hierarchical scheme for computing tree levels of detail, similar to Max's approach [12]. Human perception experiments suggest that the structure of lower level branches is critical to memorization and recognition [15],[16].

A simplifying assumption is made that a tree consists of branch segments and leaf clusters. The *level* of a branch segment is the rewriting step in which the corresponding module was added. Any level-$k$ branch segment is connected to a single level-($k$ – 1) segment, or *parent*, and may have multiple *children*, or level-($k$ + 1) segments connected to it. We say that a tree is *rooted* at the level-0 branch segment, which by construction does not have a parent. Given any branch segment we define the *subtree* to be the set of all successor segments and associated leaf clusters. A *k-subtree* is a subtree rooted at an arbitrary level-$k$ branch segment.

In the $k^{th}$ level of detail, LOD-$k$ replaces each k-subtree with a textured cross-polygon impostor. Each instance of the impostor is transformed in 3D space as the $k$-subtree it approximates. The lowest level of detail, LOD-0 replaces the entire tree with a cross-polygon. The LOD-$k$ textures are rendered from several views of an arbitrary $k$-subtree.

This hierarchical approach is especially applicable to L-system based tree generation methods. In our implementation two texture maps are generated from two perpendicular views of a 12-production L-system tree. Each texture map is a 256x256 RGBA (8 bits/component) image, scaled appropriately at run-time by the graphics hardware. Since the silhouette of a subtree is the same from any two views at an angle of 180° to each other, it is not necessary to use distinct textures for the two sides of a polygon. While this simplification does provide an incorrect view from one side, we observed minimal discrepancies due to occlusion and distance to trees rendered at lower levels of detail.

Six sample levels of detail for a 12-level tree are shown in Figure 1. LOD-0 is the lowest level of detail (a single cross-polygon) and LOD-12 is the full level of detail. Each LOD was rendered at the same angle and distance from the viewpoints. Note that LOD-0 appears smaller than the tree it approximates because both polygons in the impostor are at a 45° angle to the viewer. Orientation of individual impostors is less significant for higher levels of detail. Nevertheless, this does result in some popping artifacts.

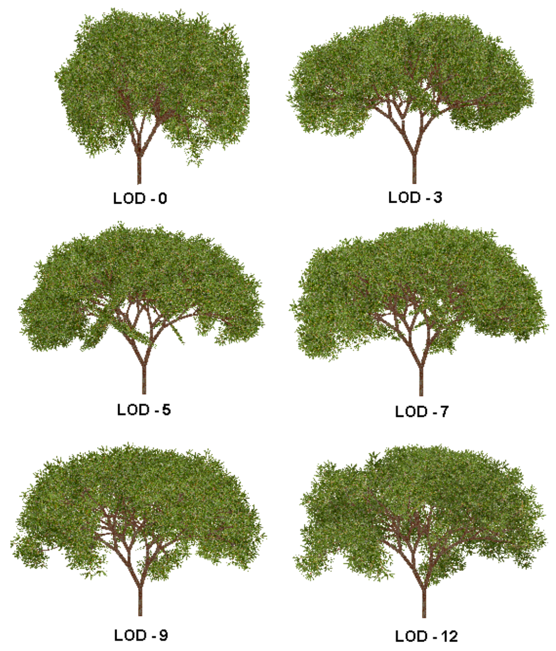

**Figure 1. Six levels of detail for a 12-production tree**

Rendering the 0th, 3rd, 5th, 7th, and 9th levels of detail results in speedups of 30, 22, 10, 4, and 2, respectively, over rendering a full-detailed tree.

## 3. Visibility-based Framework

A large body of research exists for visibility-based object culling (see [5] for a recent comprehensive survey). The methods fall into two general categories: *object* and *image* based. Object-based methods clip and cull primitives against a set of pre-selected objects or occluders [6],[11] and culling takes place before any rendering. Some examples of object-based methods include Prioritized-Layered Projection algorithms [11], Binary Space Partition algorithms, and algorithms using shadow frusta [10]. Object-based algorithms perform best in the presence of a small number of portals or large occluders, which makes them unsuitable for a forest walk-through since the objects making up trees (branch segments and leaf clusters) are many and tend to be relatively small.

Image-based visibility methods cull in window coordinates, thus rendering is required. The z-buffer algorithm is the most common example of image-based culling. In order to cull primitives or even objects, depth tests are performed on the projections of the bounding boxes. Computation is accelerated by hierarchical methods, such as Hierarchical Z-Buffer [7] and Hierarchical Occlusion Buffer [17]. These techniques store a pyramid of the buffers, where a pixel in the $k^{th}$ level of the pyramid corresponds to a 2x2 area in the $(k-1)^{th}$ level. When computing the visibility of an object, computation is started in the level that minimizes the number of pixels to be examined. Related methods were proposed by Bartz *et al.* [1][2] and Hey *et al.* [8],[9].

While the above methods were designed to cull polygons, we adopt a similar approach to select levels of detail for the trees. We propose an image-based level of detail selection method for rendering forest scenes at interactive frame rates. Given a tree, the appropriate level of detail is chosen at run-time and is based on visibility of the tree in addition to its projected size. The trees that do not fall within the view frustum are culled before any processing. The trees inside the frustum are rendered in the front-to-back order, which enables iterative visibility computation.

### 3.1. Visibility Computation

Visibility information can be computed at run-time since trees at distance $k$ are rendered before any trees in cells at distance $(k + 1)$. Visibility $Vis(t)$ of a given tree $t$ is the ratio of the number of fragments (pixels) that pass the depth test and the total number of fragments generated when rendering the tree. Visibility of a bounding box can be defined similarly. In order to simplify computations for bounding boxes, pixels are counted for the smallest window-aligned rectangle that includes the projection of the box. Most of the time this simplification is *conservative*: visibility of the box is seldom greater than that of the rectangle. The same relationship holds for any tree and its bounding box except for a few pathological cases, an example of which is a narrow tall object along a diagonal of its bounding rectangle. If the image rendered so far is available, visibility of a tree can be predicted by projecting its bounding volume into the view plane and counting the number of pixels in the bounding rectangle that have not yet been rendered.

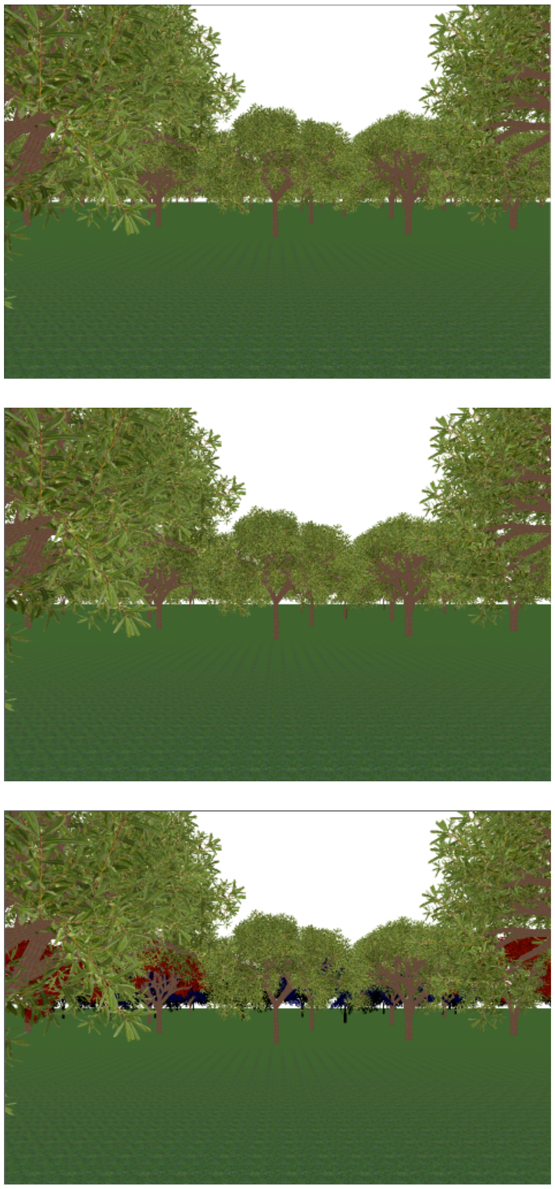

**Figure 2. A forest scene**

A number of recent graphics cards implement OpenGL GL_ARB_occlusion_query extension for occlusion queries: primitives can be submitted to the graphics hardware which then determines how many pixels would pass the depth test. Visibility of a given object is computed by issuing two queries with different depth-functions: one passes the pixels that are in front of the current z-buffer, the other passes the pixels that are behind. The sum of the queries approximates the projected size of the tree after clipping (it is an approximation since some pixels are counted twice due to object self-occlusion). LOD-0 of a tree is used for occlusion queries. Multiple queries are issued to achieve parallelism between the CPU and the GPU.

The following framework is based on the OpenGL occlusion query extension:

```
For each cell at distance $d$ within the frustum do
    Compute visibility
    Select level of detail
    Render trees
```

A sample scene is shown in Figure 2. Each tree in the top image is rendered at full level of detail, while the middle and bottom images were generated using the proposed framework. The bottom image shows color-coded trees to indicate their LODs: green is LOD-12 (highest), red is LOD-9, blue is LOD-4, and black is LOD-0 (lowest, only a single cross-polygon). Note that trees at the same distance are displayed at varying levels of detail, based on their visibility. Entire computation is performed at run-time, leading to 8 – 15FPS for forest scenes with up to 2500 unique trees. For comparison, rendering all trees at full level of detail and using view frustum culling alone results in only 0.5FPS when traversing a scene with 2500 trees. Trees are rendered by interpreting their L-system string for each frame, a necessity for MR experiences that allow interactively adjusting the age of a forest. The experiments were carried out on a 2.4GHz Pentium 4 workstation with 256MB RAM and nVidia GeForce FX5800 Ultra graphics card.

## 4. Perception Experiments

Anecdotally, many people report seeing a pop when rendering details change in a virtual environment. Such perceived disfluencies in visual stimuli can interrupt a virtual experience, causing attentional shifts, reallocation of resources, and a reduced sense of presence in the environment. In spite of the importance of this issue to those involved in the creation of realistic virtual environments, little research has examined when a pop actually is perceived.

### 4.1. Experimental Setup

The purpose of the present study is to examine when one perceives visual disfluencies in a virtual forest. Seventeen participants (12 females, 5 males) completed a pop detection task. In this task, participants were seated in front of a computer screen displaying three animated trees. The test application displayed three trees at the same distance from the viewer. The viewer travels at different speeds toward the trees and an LOD change occurs in the left, right, or none of the trees. The following four variables were tested:

LOD (6 values),
tree size in pixels (3 values),
speed of travel (4 values),
tree with changing LOD (3 values).

Experiments were executed in a random order; users indicated the perceived visual discontinuity by pressing a key. Movement could be at 1.8, 6, or 12 meters/second. The pop could occur when the tree appeared to be at 50, 100, or 200 meters in the distance. In addition, the altering tree could appear at one of 6 levels of detail and shift to an adjacent level of detail. These LOD shifts were from 1-2, 3-4, 5-6, 7-8, 9-10, or 11-12. Participant accuracy and reaction times were recorded.

### 4.2. Analysis

Only data for trials with a pop occurring were used in the analyses. Thus, accuracy data reflect the percentage of correct identifications of the pop, and error rates reflect misses of the "pop." To examine accuracy, percent correct data were subjected to a 2 (Tree: left or right) x 3 (Speed: 1.8, 6, or 12) x 3 (Distance: 50, 100, or 200) x 6 (Level of Detail: 1-2, 3-4, 5-6, 7-8, 9-10, or 11-12) repeated measures Analysis of Variance. This analysis yielded main effects for Speed, F(2, 32) = 8.16, P <.01; Distance, F(2, 32) = 32.07, P <.001, and Level of Detail, F(5, 80) = 28.17, P <.001. These analyses indicate that independently, Speed of Movement, Distance of Object, and Level of Detail can be used to predict the detection of a pop in an animated forest. Participants more accurately detected the pop when movement toward the trees was faster. While this finding is somewhat counterintuitive, it is conjectured that participants were more likely to miss a pop when moving slowly because the change appeared gradually in front of them. With a faster speed, the animation is less gradual. Experimentally, it is nearly impossible to tease apart the speed of movement toward the trees and the rate at which the "pop" must take place. Additionally, a faster pace may heighten vigilance through perceived arousal.

Pop detection was more accurate for trees presented close to the viewer. Not surprisingly, each increment of distance closer to the participant was associated with greater accuracy. This effect is largely the result of perceived size. Distant trees appear smaller, and thus are not given the same attentional weight as those that are close to the observer. Furthermore, minute changes in a very small stimulus are likely to be attributed to pixelation during animation, rather than an actual interruption of the animation.

Overall, pop detection becomes less accurate with lower levels of detail. Further analysis of this effect showed that all LODs differed from each other, with the exception of LOD shifts 7-8 and 9-10. This finding suggests that on a continuum of very high detail to very low detail, higher levels of detail facilitate the detection of pop. In addition, performance appears to plateau for middle levels of detail, before again decreasing with the lowest LOD. From a more applied perspective, detail is a double-edged sword. It makes perception more accurate, even when that involves perceiving problems.

The percent correct analysis also yielded a significant interaction between Distance and Level of Detail, F(10, 160) = 4.60, P <.001, and a marginal interaction between Speed and Level of Detail, F(10, 160) = 1.88, P =.05. While Distance and Level of Detail can independently affect pop detection, different pairings of these variables yield very different performance. In short, the lower the level of detail, the more distance becomes important as a visual cue for pop. With a high level of detail, there are non-significant differences in performance for trees at different distances. However, as detail becomes progressively worse, distance becomes a significant predictor variable. Performance for very distant trees (i.e. 200 meter) changes most with level of detail. These results suggest that proximity can compensate for lack of detail.

Level of Detail and Speed interact in a very different pattern. At walking speed, accuracy generally declines with level of detail, plateauing at the 4th LOD shift. However, at faster speeds (e.g., speed of vehicle movement), performance does not drop off until later levels of detail, and is not a linear downward trend. This pattern of results supports the idea that speed may compensate for some loss of detail (e.g., nine-ten), but cannot compensate for extremely low levels of detail.

Prior to analysis, reaction times for incorrect trials were deleted. This transformation left only 8 highly accurate participants for analysis. All other participants missed all of the trials in at least one of the 108 cells. The reaction time data were analyzed using a 2 (Tree: left or right) x 3 (Speed: 1.8, 6, or 12) x 3 (Distance: 50, 100, or 200) x 6 (Level of Detail: 1-2, 3-4, 5-6, 7-8, 9-10, or 11-12) repeated measures Analysis of Variance. This analysis yielded main effects for Speed, F(2, 8) = 8.56, P <.05; Distance, F(2, 8) = 12.04, P <.01, and Level of Detail, F(5, 20) = 4.75, P <.01. Once again, the data show that Speed of Movement, Distance of Object, and Level of Detail independently predict speed of pop detection in participants who accurately identify pop. Reaction time drops as a function of speed. In essence, when the animation moves the participant toward the trees at a faster rate, then he or she is forced to make a judgment more quickly. Speed only affects reaction time beyond 6 m/s. It may be the case that when one gets beyond this point, cognitive mechanisms switch from those used for self-locomotion to those used when one perceives that he or she is being transported. This finding is in line with research suggesting that different spatial abilities exist for exocentric and intrinsic manipulations [2], and also with the idea that resources may be freed when one does not have to allocate processes for self-locomotion.

Reaction time to detect a pop also increases when the tree to be detected appears further away. This outcome may be the result of smaller trees causing greater indecision because one cannot decide if there is a pop or just pixelation. Reaction time is generally similar for all LODs. Post hoc comparisons showed that reaction times were higher for the seven-eight LOD shift than for 3-4, 5-6, or 9-10. This LOD appears to be crucial for causing longer decision times, and should be explored further. Although the lowest LOD shift appears to take the longest, as the error bars indicate, this increase in reaction varied greatly by participant.

The reaction time analysis also yielded a significant Speed x Distance interaction, F(4, 16) = 3.14, P <.05. For very close trees, reaction time became faster as a function of speed of movement toward the tree. However, for the farthest trees, reaction time did not vary as a function of speed of movement. This finding is in line with the idea that at far distances, one's ability to detect perceptual cues is not a function of one's own perspective, an outcome that is similar to that observed with binocular disparity as a depth cue.

## 5. Conclusions

This paper presented a visibility-based level of detail management framework for virtual walk-though applications as well as a study of human perception of visual popping artifacts when switching between representations of the same object at different levels of detail. Experimental findings were used to tune the visibility-based forest walk-through in order to achieve a balance between visual quality and performance. The resulting system performed at least 16 times faster than the one without any LOD management (8FPS vs. 0.5FPS).

The perception study also demonstrates that it is possible to empirically investigate difficult to define constructs such as the notion of "pop" on a visual display, and that level of detail, speed of movement toward objects, and distance from objects all contribute to the accurate and fast perception of breaks in the visual scene. A first major outcome of the study is that participants appear to vary considerably in terms of where these effects can be seen. For some participants, certain manipulations of these variables appear to cause errors, whereas for others, it causes a reduction in speed. Thus, future research may want to investigate the role of expertise, and variables such as spatial ability, in predicting pop detection performance. Second, this research suggests that specific pairings of speed or distance can be used to compensate for rendering with poor detail, and that certain LODs are more vulnerable to the influences of outside variables. Third, practitioners who vary LOD in order to preserve resources can use these data to show that under specific speed and distance conditions, certain LODs are not significantly different from neighboring LODs in terms of effects on performance. Finally, the data show that type of perceived movement (walking vs. slow driving) is associated with differences in performance in a virtual environment. Future research may want to investigate whether this effect remains with even faster movement speeds.

## 6. Acknowledgements

The work reported was supported by the Army Research Development and Engineering Command, the National Science Foundation (under grant EIA 9986051) and the Canon Mixed Reality Systems Laboratory in Tokyo.

## 7. References

[1] D. Bartz, M. Mei_ner, T. Hüttner. Extending graphics hardware for occlusion queries in OpenGL. *Proc. of Workshop on Graphics Hardware 98*, pp. 97-104, 1998.
[2] D. Bartz, M. Mei_ner, T. Hüttner. OpenGL-assisted occlusion culling for large polygonal models. *Computer and Graphics*, **23**(5), pp. 667-679, 1999.
[3] J. B. Carroll. *Human cognitive abilities*. Cambridge University Press, Cambridge England, 1993.
[4] D. Cohen. Computer simulation of biological pattern generation processes. *Nature*, **216**, pp. 246-248, 1967.
[5] D. Cohen-Or, Y. Chrysanthou, C. T. Silva. A survey of visibility for walkthrough applications. *IEEE Transactions on Visualization and Computer Graphics*, to appear, 2002.
[6] S. Coorg, S. Teller. Real-time occlusion culling for models with large occluders. *1997 Symposium on Interactive 3D Graphics*, pp. 83-90, 1997.
[7] N. Greene, M. Kass, G. Miller. Hierarchical z-buffer visibility. *Proc. of SIGGRAPH 93*, pp. 231-240, 1993.
[8] H. Hey, R. F. Tobler. Lazy occlusion grid culling. *Technical Report TR-186-2-99-09*, Vienna University of Technology, March 1999.
[9] H. Hey, R. F. Tobler, W. Purgathofer. Real-time occlusion culling with a lazy occlusion grid. *Technical Report TR-186-2-01-02*, Vienna University of Technology, January 2001.
[10] T. Hudson, D. Manocha, J. Cohen, M. Lin, K. Hoff, H. Zhang. Accelerated occlusion culling using shadow frusta. *Proc. of the 13th Annual ACM Symposium on Computational Geometry*, pp. 1-10, 1997.
[11] J. T. Klosowski, C. T. Silva. Efficient conservative visibility culling using the prioritized-layered projection algorithm. *IEEE Transactions on Visualization and Computer Graphics*, **7**(4), pp. 365-379, 2001.
[12] N. Max, O. Deussen, B. Keating. Hierarchical image-based rendering using texture mapping hardware. In *Eurographics Workshop on Rendering 99*, pp. 57-62, 1999.
[13] P. Prusinkiewicz, A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990.
[14] P. Prusinkiewicz, M. James, R. M_ch. Synthetic topiary, *Pro. of SIGGRAPH 94*, pp. 351-358, 1994.
[15] V. K. Sims, J. M. Moshell, C. E. Hughes, J. E. Cotton, J. Xiao. Recognition of computer generated trees. *Proceedings of the Human Factors and Ergonomics Society*, **46**, to appear.
[16] V. K. Sims, J. M. Moshell, C. E. Hughes, J. E. Cotton, J. Xiao. Salient characteristics of virtual trees. *Proceedings of the Human Factors and Ergonomics Society*, **45**, pp. 1935-1938, 2001.
[17] H. Zhang, D. Manocha, T. Hudson, K. Hoff. Visibility culling using hierarchical occlusion maps. *Proc. of Siggraph 97*, pp. 77-88, 1997.