Rendering Realistic Trees and Forests in Real Time

Alberto Candussi, Nicola Candussi, Tobias Höllerer
Department of Computer Science, University of California, Santa Barbara

Abstract

Real-Time rendering of realistic trees on common graphics hardware represents a big challenge due to their inherent geometric complexity. In most cases, trees are composed of hundreds of thousands of leaves and branches with complex lighting interrelations. We present novel techniques to render and animate photorealistic trees in real-time. The described techniques are easily implemented with commonly available graphics cards, making them suitable to applications such as visual simulations and video games. Polygon counts are significantly reduced by the use of simplified textured geometry for minor branches and bill-boarded leaf textures. Fast and realistic lighting and shadowing of the leaves and surroundings enhances realism. We animate the tree branches and leaf textures using simple vertex shaders, creating a realistic effect of the tree swaying in the wind. Discrete levels of detail allow rendering of a large number of trees, making it possible to represent realistic forest scenes made of 1000-1500 trees.

Categories and Subject Descriptors: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

We present techniques for realistically rendering trees in real-time using common graphics hardware with programmable shader support. The main applications of our technique are visual simulations and videogames. It is necessary to find a surface representation of the tree that has manageable geometric complexity while at the same time maintaining high rendering realism, even at close-up views. Nearly all consumer graphics cards that are the target of our applications rely on triangles as the main rendering primitive. Even if only two triangles per leaf were used, the polygon count would be prohibitive for our purposes. On the other extreme, trees have also often been approximated by a small number of billboards [Jak00, Per95]. This sacrifices realism when viewing trees from close view points. We present a technique to group leaves and branches in clusters that look realistic and maintain low geometric complexity.

The realism of rendered plants and trees is highly influenced by how complex lighting interrelations between branches and leaves are modeled. We adopt a combination of custom per-vertex and per-pixel lighting, together with customized shadow mapping. Figure 1 shows an example of the realism achieved with our method.

Thanks to the use of level-of-detail techniques, our solution is very scalable, and allows the rendering of complex scenes consisting of thousands of trees in real-time. In order to avoid the "popping effect" typical of discrete level of detail methods, we adopt a custom transition based on alpha testing.

We also introduce an animation method for all the elements of the tree (trunk, branches and leaves) that completely fits inside a vertex shader.

Figure 1: *Procedurally generated tree rendered in real time with shadows and our custom lighting approach.*

1.1. Related Work

Modern videogames continuously raise the level of realism for trees and plants appearing in their scenes. Most of them use a non-billboarded approach for leaves rendering and, where level-of-detail (LOD) control is used,

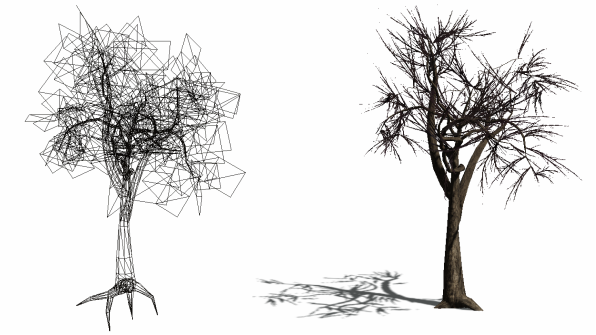

Figure 2: *Minor branches are approximated by simple non-planar(curved) textured polygon strips.*

Figure 3: *Leaves are rendered as billboarded polygons with varied textures showing a bunch of perspective leaves each (cf. Figure 5)*

frequently a single billboard representing the whole plant [Per95] is used for far views. This makes the rendering appear quite flat; with our technique the leaves of a tree with a low LOD are represented by more billboards, but we are still maintaining interactive frame rates even drawing a very large number of trees (1000 and more, cf. Figure 5 and table 2).

Deussen et al. [DCMD02] use point and line rendering for plants far away from the viewer, while maintaining a full triangle representation for near ones. The resulting scene complexity allows interactivity only if a small number of trees are rendered.

Jakulin [Jak00] approximates the entire tree by using a variable number of textured slices. While this gives a good visual appearance for distant objects, it is not quite convincing for close-up images, for which a geometrically detailed surface representation is more appropriate.

Decauding et al. [DN04] use a similar approach based on 3d textures to render an entire landscape full of trees. Again, their technique is more focused on real-time rendering of a high number of distant trees (e.g., for flight simulations), not on close-up forest walk-throughs.

Real time shadows add great realism to the scene. Due to the high complexity of the tree geometry, we chose shadow mapping as our base algorithm. In particular, we employ perspective shadow mapping [SD02] to make better use of the shadow map resolution.

2. Modeling and Real-Time Rendering

Figure 4: *Tree close up showing leaf detail.*

In this section, we describe the major ideas behind representing our realistic trees so that large sets of them can be rendered in real-time. We describe the components of a tree and the methods to reduce geometric complexity.

2.1. Tree Model

Three different kinds of geometric primitives compose a tree: deformed cylindrical meshes, grid meshes and billboards.

Deformed cylindrical meshes are used to draw roots, trunk, and major branches. They can have sub-branches, in order to create the typical tree structure.

Grid meshes (in the simplest case, polygon strips), are attached to branches to represent, for example, groups of minor branches or palm fronds. They have a user defined profile and they are rendered using alpha test transparency. Figure 2 compares the wireframe geometry of branches with the same geometry textured.

Leaves are grouped into clusters and each cluster is rendered as a separate billboard. During tree generation, special care is put in the distribution of the billboards. The billboards must be as few as possible while maintaining an overall feeling of volume. Figure 3 shows leaves' wireframe geometry as compared with the complete tree.

Our generation algorithm allows the creation of a wide range of trees based on different parameters such as branch radius, angle, length, swaying flexibility, and leaf distribution.

2.2. Acceleration Techniques

This section discusses the techniques used to increase rendering performance, such as billboarding, discrete level of detail transitions (LOD), render-state-sorting, and instancing.

Since most of the geometric complexity can be attributed to leaves, grouping them together into single billboards greatly reduces the polygon count while maintaining the volumetric appearance (cf. Figure 3).

We use grid meshes to approximate groups of smaller branches. A discrete set of LODs is used for roots, trunk, major and minor branches, and leaves. For cylindrical

Figure 5: *Example textures for leaf and minor branch geometry.*

meshes like trunks and branches the complexity is reduced by decreasing the radial and longitudinal subdivision. Some elements that are less visible (for example, because hidden by leaves) are completely removed in lower LODs.

The detail of the grid meshes used for minor branches is reduced by reducing the grid subdivision.

In the case of billboards, lower detail levels have fewer and bigger billboards. This way, even if the geometric complexity is reduced, the difference between LODs becomes less apparent. To avoid the popping effect while going from one LOD to the other, we implemented a smooth fade-in/fade-out transition for billboards. The fade effect is done with alpha test; when fading out, the number of rendered pixels is smoothly reduced, when fading in, it is smoothly increased. This is done by assigning random alpha values for the billboard texture pixels and then varying the alpha comparison value for the alpha test.

When the number of objects to be rendered becomes large, instancing and state ordering is essential. In the forest of Figure 6, there are only 3 kind of trees, instantiated around the landscape and sorted by render state. To decrease the number of texture swaps, a texture dictionary technique is used for grid meshes and billboards (cf. Figure 5).

3. Realism

A realistic virtual tree must be fully customizable through an editor and it must have convincing lighting and animation systems (cf. Figure 4). In this section we describe our lighting and animation approach.

3.1. Realistic Lighting and Shadows

Roots, trunk, and major branches are rendered with bump mapping and shadow mapping. Grid meshes have shadow mapping and per vertex lighting.

Leaf lighting is obtained as the product of four factors: per vertex lighting, density factor, shadow map factor, and occlusion factor.

Per vertex lighting is computed assuming that the vertex normals of the leaves are oriented along the radius of the leaves' bounding sphere, pointing outside. This way, leaves on the side of the tree that is closer to the light are brighter than leaves on the opposite side. In order to avoid completely unlit leaves on the side opposite the light and to have a smoother transition between bright and dark leaves,

Figure 6: *Forest consisting of 1000 trees, rendered in real time with realistic lighting and shadows. This illustrates real-time performance.*

we halve the vertex normals for the diffuse lighting calculation and correct for the loss in lighting with a global ambient term. Figure 7a illustrates the vertex lighting factor effect.

The density factor, shown in Figure 7b, takes into account the number of leaves surrounding the current leaf and is pre-computed.

The shadow map factor is the result of the shadow map test on the current leaf. Shadows are smoothed using percentage closer filtering (supported in hardware by most modern graphics cards).

The occlusion factor determines the intensity of the projected shadow based on the depth value of the occluder saved on the shadow map; the farther the occluder, the darker the shadow. Figure 7c shows the final result without occlusion factor and Figure 7d with it. Figure 1 shows the result of taking all the factors into account. For the generation of the shadow map, billboards are rendered facing the light and using a lower LOD. Even though not physically correct, the resulting shadows appear realistic, because the sheer number of real leaves in a tree and the variance in leaf orientation makes it impossible for a human to anticipate the correct shadow projection.

3.2. Realistic Animation

All the tree's animation properties (such as branch flexibility) are pre-computed. At run-time the tip of every branch is displaced and the rest of the branch movement is done by interpolating this transformation based on each vertex's distance from the tip.

The displacement is computed as the sum of two vectors, one along the wind direction and the other one orthogonal to it. The second vector is used to add some noise to the movement. At each frame, periodically changing factors are added to these vectors, in order to modify their magnitude and direction. Leaves are animated depending on wind strength and direction; billboards perform an oscillation around the viewing axis, based on sine waves. All the displacements and rotations are passed to the vertex shader, which performs all the vertex transformation needed.

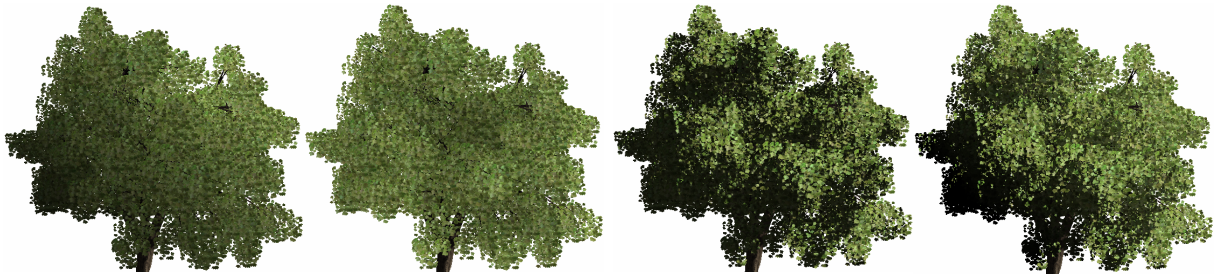

Figure 7: *Leaf lighting factors; a) per vertex lighting, b) density factor, c) shadow map factor, d) shadow map factor combined with occlusion factor.*

4. Results

The tree of Figure 1, in its most detailed LOD, consists of about 4000 triangles for deformed cylindrical meshes, about 400 triangles for grid meshes, and about 4000 triangles for billboards. It is rendered on a GeForce 6800 GT at an average frame rate of 75 fps.

The forest shown in Figure 6, composed of 1000 trees, is rendered in realtime; Table 1 shows the polygon count for each LOD of one of the trees in the forest. Table 2 shows the average rendering performance of the forest at 1024x768 screen resolution, varying the number of trees.

	LOD1	LOD2	LOD3	LOD4
Cylindrical Meshes	640	165	36	3
Grid Meshes	120	36	4	0
Billboards	3406	510	76	10

Table 1: *Polygon count of trees in forest of Figure 6.*

5. Conclusions and Future Work

It has been shown that with this approach it is possible to render a large number of highly realistic trees in real-time. Billboards drawn with our lighting system represent an optimized and realistic solution to simulate leaves' complexity. Our representation is highly scalable to balance scene complexity and rendering quality.

The time necessary to model a new tree species varies between less than half an hour and multiple hours. We implemented a rudimentary tree editor, which allows for interactive control of the many parameters governing the appearance of a tree. Automating an increasing number of the design decisions and group them into templates for the tree editor is one of our goals for future work.

We also have plans to implement lighting in more complex environments with environment mapping and multiple light sources. We are also working on optimizations that will allow us to render even larger landscapes with on the order of 10,000 trees.

	GeForceFX 5950 Ultra	GeForce 6800 GT
500 Trees	20 fps	43 fps
1000 Trees	12 fps	26 fps
1500 Trees	8 fps	18 fps

Table 2: *Rendering speeds for forest of Figure 6.*

6. References

[DN04] Philippe Decaudin, Fabrice Neyret, 2004: Rendering Forest Scenes in Real-Time. In Rendering Techniques '04 (Eurographics Symposium on Rendering), Norrköping, Sweden, 93-102.

[DCMD02] Oliver Deussen, Carsten Colditz, Marc Stamminger and George Drettakis, 2002: Interactive visualization of complex plant ecosystems. In Pro. VIS '02 *(Proceedings of the conference on Visualization '02*), Boston, Massachusetts, 219-226.

[Jak00] Aleks Jakulin, 2000: Interactive Vegetation Rendering with Slicing and Blending. Eurographics 2000 short paper , Interlaken, Switzerland.

[SD02] Marc Stamminger and George Drettakis, 2002: Perspective shadow maps. In Pro. SIGGRAPH '02 (*Proceedings of the 29'th annual conference on Computer graphics and interactive techniques*), San Antonio, Texas, 557-562

[Per95] Iris performer programmer's guide, 1995