Leaf Cluster Impostors for Tree Rendering with Parallax

Ismael Garcia, Mateu Sbert, and László Szirmay-Kalos

University of Girona and Budapest University of Technology
Emails: igarcia@ima.udg.es, mateu@ima.udg.es, szirmay@iit.bme.hu

Abstract

This paper presents a simple method to render complex trees on high frame rates while maintaining parallax effects. Based on the recognition that a planar impostor is accurate if the represented polygon is in its plane, we find an impostor for each of those groups of tree leaves that lie approximately in the same plane. The groups are built automatically by a clustering algorithm. Unlike billboards, these impostors are not rotated when the camera moves, thus the expected parallax effects are provided. On the other hand, clustering allows the replacement of a large number of leaves by a single semi-transparent quadrilateral, which improves rendering time considerably. Our impostors well represent the tree from any direction and provide accurate depth values, thus the method is also good for shadow computation.

1. Introduction

One of the major challenges in rendering of vegetation is that the large number of polygons needed to model a tree or a forest exceeds the limits posed by the current rendering hardware [DHL*98]. Rendering methods should therefore apply simplifications. To classify these simplification approaches, we can define specific scales of simulation at which rendering should provide the required level of realism:

- *Insect scale*: A consistent, realistic depiction of individual branches and leaves is expected. The images of individual leaves should exhibit parallax effects when the viewer moves, including the perspective shortening of the leaf shape and its texture, and view dependent occlusions.
- *Human scale*: Scenes must look realistic through distances ranging from an arm's reach to some tens of meters. Consistency is desired but not required.
- *Vehicle scale*: Individual trees are almost never focused upon and consistency is not required. Viewing distance may exceed several hundred meters.

There are two general approaches for realistic tree rendering: geometry-, and image-based methods. As it takes roughly hundred thousand triangles to build a convincing model of a single tree, geometry-based approaches should apply some form of Level of Detail technique [LRC*02].

Image-based methods [RMD04, ABC*04] represent a trade-off of consistency and physical precision in favor of more photorealistic visuals (figure 1). *Billboard rendering* is analogous to using cardboard cutouts. In order to avoid the shortening of the visible image when the user looks at it from grazing angles, the billboard plane is always turned towards the camera. Although this simple trick solves the shortening problem, it is also responsible for the main drawback of the method. The tree always looks the same no matter from where we look at it. This missing parallax effect makes the replacement too easy to recognize. In order to handle this problem, the *view-dependent sprite* method pre-renders a finite set from a few views, and presents the one closest in alignment with the actual viewing direction. A popping artifact is visible when there is an alignment change. The *complex cutout* approach uses texture transparency and blending to render more than one views at the same time onto properly aligned surfaces. In order to handle occlusions cor-

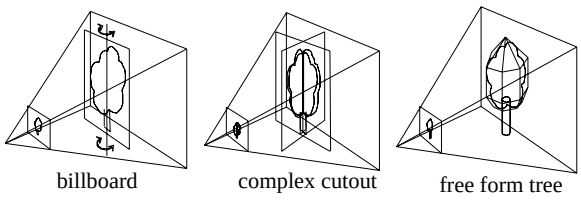

Figure 1: *A tree representation with images*

rectly, billboards can also be augmented with depth information [MO95, Szi05]. One of the most advanced methods actually implemented in commercial entertainment software is the basic *free-form textured tree model*, where the images are imposed on the approximated geometry.

The method of this paper falls into the class of complex cutouts, but it prepares them so carefully that the results are satisfactory not only on vehicle scale, but also on human scale, and with some compromises on insect scale. The method is also good to generate shadows [MNP01] and can be extended to a hierarchical approach [MDK99].

2. The new method

In order to reduce the geometric complexity of a tree, we use impostors having transparent textures to represent leaf groups. Since impostors are not rotated when the camera changes, the expected parallax effects are provided [DDSD03]. The key of the method is then the generation of these impostors. We should first observe that an impostor is a perfectly accurate representation of a polygon (i.e. leaf) if the plane of impostor is identical to the plane of the polygon. On the other hand, even if the leaf is not on the impostor plane but is not far from that and is roughly parallel to the plane, then the impostor representation results in acceptable errors. Based on this recognition, we form clusters of the leaves in a way that all leaves belonging to a cluster lie approximately on the same impostor plane and replace the whole group by a single impostor. In order to control clustering, we shall introduce an error measure for a plane and a leaf, which includes both the distance of the leaf from the plane and the angle of the plane and leaf normals. The applied *K-means clustering algorithm* [Mac67] starts with a predefined number of random planes, and iterates the following steps:

1. For each leaf we find that plane that minimizes the error. This step clusters the leaves into groups defined by the planes.
2. In each leaf cluster, the plane is recomputed in order to minimize the total error for the leaves of this cluster with respect to this plane.

Let us examine the definition of the error measure and these steps in details.

2.1. Error measure for a leaf and a plane

Those $\mathbf{p} = [x,y,z]^T$ points are on a plane that satisfy the following plane equation (T denotes matrix transpose):

$$D_{[\mathbf{n},d]}(\mathbf{p}) = \mathbf{p}^T \cdot \mathbf{n} + d = 0,$$

where $\mathbf{n} = [n_x, n_y, n_z]^T$ is the normal vector of the plane, and d is a distance parameter. We assume that $\mathbf{n}$ is a unit vector and is oriented such that d is non-negative. In this case $D_{[\mathbf{n},d]}(\mathbf{p})$ returns the signed distance of point $(\mathbf{p})$ from the plane.

From the point of view of clustering, leaf i is defined by its unit length average normal $\mathbf{n}_i$ and its center $\mathbf{p}_i$, that is by pair $(\mathbf{n}_i, \mathbf{p}_i)$.

Leaf i approximately lies in impostor plane $[\mathbf{n},d]$ if $D^2_{[\mathbf{n},d]}(\mathbf{p}_i)$ is small, and its normal is approximately parallel with the normal of the plane, which is the case when $1-(\mathbf{n}_i^T \cdot \mathbf{n})^2$ is also small. Thus an appropriate error measure for plane $[\mathbf{n},d]$ and leaf $(\mathbf{n}_i, \mathbf{p}_i)$ is:

$$E([\mathbf{n},d] \leftrightarrow (\mathbf{n}_i, \mathbf{p}_i)) = D^2_{[\mathbf{n},d]}(\mathbf{p}_i) + \alpha \cdot (1-(\mathbf{n}_i^T \cdot \mathbf{n})^2), \quad (1)$$

where α expresses the relative importance of the orientation similarity with respect to the distance between the leaf center and the plane.

2.2. Finding a good impostor plane for a leaf cluster

To find an optimal plane for a leaf group, we have to compute plane parameters $[\mathbf{n},d]$ in a way that they minimize total error $\sum_{i=1}^N E([\mathbf{n},d] \leftrightarrow (\mathbf{n}_i, \mathbf{p}_i))$ with respect to the constraint that the plane normal is a unit vector, i.e. $\mathbf{n}^T \cdot \mathbf{n} = 1$. Using the Lagrange-multiplier method to satisfy the constraint, we have to compute the partial derivatives of

$$\sum_{i=1}^N E([\mathbf{n},d] \leftrightarrow (\mathbf{n}_i, \mathbf{p}_i)) - \lambda \cdot (\mathbf{n}^T \cdot \mathbf{n} - 1) =$$

$$\sum_{i=1}^N (\mathbf{p}_i^T \cdot \mathbf{n} + d)^2 + \alpha \cdot \sum_{i=1}^N (1-(\mathbf{n}_i^T \cdot \mathbf{n})^2) - \lambda \cdot (\mathbf{n}^T \cdot \mathbf{n} - 1) \quad (2)$$

according to $n_x, n_y, n_z, d, \lambda$, and have to make these derivatives equal to zero. Computing first the derivative according to d, we obtain:

$$d = -\mathbf{c}^T \cdot \mathbf{n}. \quad (3)$$

where

$$\mathbf{c} = \frac{1}{N} \cdot \sum_{i=1}^N \mathbf{p}_i$$

is the *centroid* of leaf points. Computing the derivatives according to n_x, n_y, n_z, and substituting formula 3 into the resulting equation, we get:

$$\mathbf{A} \cdot \mathbf{n} = \lambda \cdot \mathbf{n}, \quad (4)$$

where matrix $\mathbf{A}$ is

$$\mathbf{A} = \sum_{i=1}^N (\mathbf{p}_i - \mathbf{c}) \cdot (\mathbf{p}_i - \mathbf{c})^T - \alpha \cdot \sum_{i=1}^N \mathbf{n}_i \cdot \mathbf{n}_i^T. \quad (5)$$

Note that the extremal normal vector is the eigenvector of symmetric matrix $\mathbf{A}$. In order to guarantee that this extremum is a minimum, we have to select that eigenvector which corresponds to the smallest eigenvalue. We could calculate the eigenvalues and eigenvectors, which requires the solution of the third order characteristic equation. On the other hand, we can take advantage that if $\mathbf{B}$ is a symmetric

3×3 matrix, then iteration $\mathbf{B}^n \cdot \mathbf{x}_0$ converges to a vector corresponding to the largest eigenvalue from an arbitrary, nonzero vector $\mathbf{x}_0$ [Wei03]. Thus if we select $\mathbf{B} = \mathbf{A}^{-1}$, then the iteration will result in the eigenvector of the smallest eigenvalue.

2.3. Indirect texturing

The proposed method uses a single texture for the impostor that includes many leaves. It means that the impostor texture should have high resolution in order to accurately represent the textures of individual leaves. The resolution of the impostor texture can be reduced without sampling artifacts if the impostor stores only the position and orientation of the leaves, while the leaves rotated at different angles are put in a separate texture (figure 2). Let us identify the rectangles of the projections of the leaves on the impostor. Each rectangle is defined by its lower left coordinate, the orientation angle of the leaf inside this rectangle, and a global scale parameter (due to clustering the impostor planes are roughly parallel to the leaves thus leaf sizes are approximately kept constant by the projections). If we put the lower left coordinate of the enclosing rectangle and the orientation angle into the impostor texture, and fill up the texels that are not covered by leaves by a constant value outside [0,1], then this texture has constant color rectangles, which can be down-sampled. In fact, this down-sampling could replace a rectangle of the leaf projection by a single texel. We call this low resolution texture as the *leaf distribution impostor*. During rendering, we address this leaf distribution impostor with texture coordinates u, v. If the first color coordinate of the looked up texel value is outside the [0,1] interval, then this texture coordinate does not address a leaf, and thus should be regarded as transparent. However, if the first coordinate is in [0,1], then first two color coordinates r, g are considered as the texture coordinates of the lower left corner of the leaf rectangle, and color coordinate b is regarded as the rotation angle of the leaf. The rotation angle selects the texture that represents this rotation, and u, v texture coordinates are translated by r, g and scaled by size s of the leaf rectangle, i.e. $u' = (u-r)/s, v' = (v-g)/s$ will be the texture coordinates of the single leaf texture selected by component b.

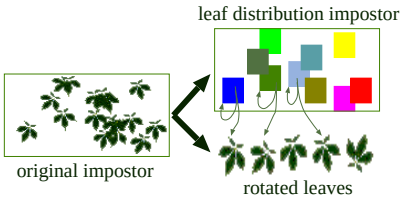

Figure 2: *Indirect texturing. The impostor is replaced by a leaf distribution impostor and the rotated images of the leaves.*

2.4. Smooth level of detail

The accuracy of the impostor representation can be controlled by the number clusters. If clusters are organized hierarchically, then the accuracy can be dynamically set by specifying the used level of the hierarchy. The level is selected according to the viewing distance, which results in a level of detail approach. To make the changes smooth, we can interpolate the plane parameters of the two enclosing levels.

3. Results

The proposed algorithm has been implemented in OpenGL/Cg environment, integrated into the Ogre3D game engine, and run on an NV6800GT graphics card. The tree is a European chestnut (Castanae Sativa) having 11291 leaves defined by 112910 faces and 338731 vertices. The trunk of the tree has 46174 faces. The leaves have been converted to 32 impostors using the proposed algorithm. When we did not apply indirect texturing, the impostor resolution was 512×512. Setting the screen resolution to 1024×768, leaf rendering is speeded up from 40 FPS to 278 FPS when the leaf polygons are replaced by the leaf cluster impostors. The comparison of the images of the original polygonal tree and the impostor tree is shown in figure 4. Note that this level of similarity is maintained for all directions, and the impostor tree also provides realistic parallax effects.

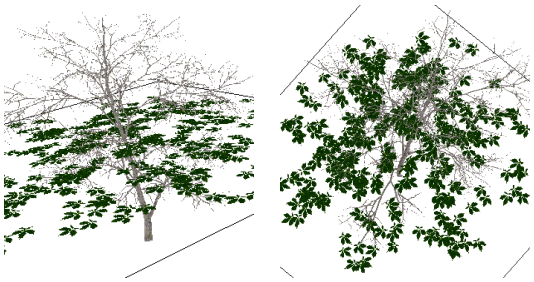

Figure 4: *Two from the 32 leaf clusters representing the tree*

Figure 5 shows a terrain of 4960 polygons with 2000 trees rendered on 30 FPS without instancing (the rendering is CPU limited). We used the proposed level of detail techniques to dynamically reduce the number of leaf cluster impostors per tree from 32 to 16 and even to 8.

4. Conclusions

This paper presented a tree rendering algorithm where clusters of leaves are represented by semi-transparent impostors. The automatic clustering algorithm finds an impostor in a way that the represented leaves lie approximately in its plane. This approach is equivalent to moving and rotating the leaves a little toward their common planes, thus this representation keeps much of the geometry information. Since

polygonal tree | 107 impostors, 163 FPS | 54 impostors, 256 FPS | 29 impostors, 292 FPS

Figure 3: *Visual and speed comparisons of the original polygonal tree and the tree rendered with different number of impostors.*

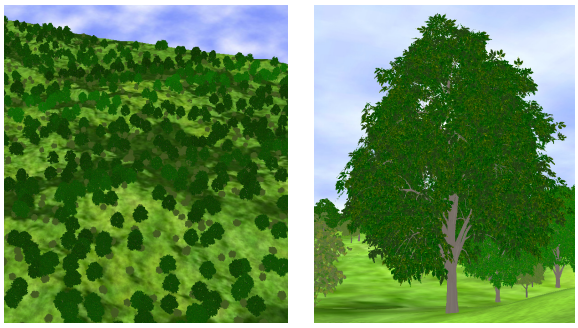

Figure 5: *Two snapshots from a 30 FPS animation showing 2000 trees*

the impostors are not rotated with the camera, the proposed representation can provide parallax effects and view dependent occlusions for any direction. We also discussed how leaf textures can be visualized without posing high resolution requirements for the impostors, and the possibility of including smooth level of detail techniques.

5. Acknowledgement

This work has been supported by OTKA (ref. No.: T042735), GameTools FP6 (IST-2-004363) project, TIN2004-07451-C03-01 project from the Spanish Government, and by the Spanish-Hungarian Fund (E-26/04).

References

[ABC*04] ANDUJAR C., BRUNET P., CHICA A., NAVAZO I., ROSSIGNAC J., VINACUA J.: Computing maximal tiles and application to impostor-based simplification. *Computer Graphics Forum 23*, 3 (2004), 401–410. 1

[DDSD03] DÉCORET X., DURAND F., SILLION F., DORSEY J.: Billboard clouds for extreme model simplification. In *SIGGRAPH '2003 Proceedings* (2003), pp. 689–696. 2

[DHL*98] DEUSSEN O., HANRAHAN P., LINTERMANN R., MECH R., PHARR M., PRUSINKIEWICZ P.: Interactive modeling and rendering of plant ecosystems. In *SIGGRAPH '98 Proceedings* (1998), pp. 275–286. 1

[LRC*02] LUEBKE D., REDDY M., COHEN J., VARSHNEY A., WATSON B., HUEBNER R.: *Level of Detail for 3D Graphics*. Morgan-Kaufmann, 2002. 1

[Mac67] MACQUEEN J. B.: Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability* (1967), pp. 281–297. 2

[MDK99] MAX N., DEUSSEN O., KEATING B.: Hierarchical image-based rendering using texture mapping. In *Eurographics Workshop on Rendering* (1999), pp. 57–62. 2

[MNP01] MEYER A., NEYERET F., POULIN: Interactive rendering of trees with shading and shadows. In *Eurographics Workshop on Rendering* (2001). 2

[MO95] MAX N., OHSAKI K.: Rendering trees from precomputed z-buffer views. In *Eurographics Workshop on Rendering* (1995), pp. 74–81. 2

[RMD04] RECHE A., MARTIN I., DRETTAKIS G.: Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Transactions on Graphics 23*, 3 (2004). 1

[Szi05] SZIJÁRTÓ G.: 2.5 dimensional impostors for realistic trees and forests. In *Game Programming Gems 5*, Pallister K., (Ed.). Charles River Media, 2005, pp. 527–538. 2

[Wei03] WEISSTEIN E.: *World of Mathematics*. 2003. http://mathworld.wolfram.com/Eigenvector.html. 3