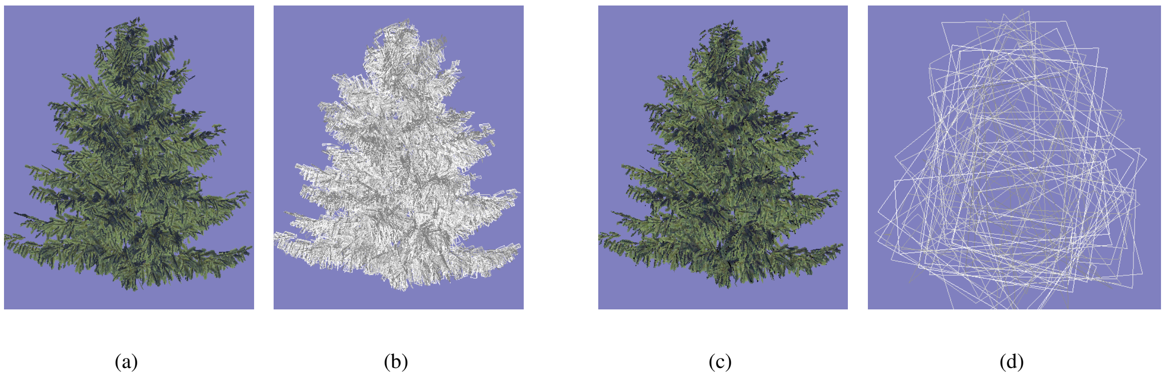


Figure 1: Spruce tree before and after transition. (a) Shaded base mesh (20610 triangles), 60 FPS. (b) Wireframe base mesh. (c) Shaded billboard cloud (78 billboards), 300 FPS. (d) Wireframe billboard cloud.

Stochastic Billboard Clouds for Interactive Foliage Rendering

J. Dylan Lacewell Dave Edwards Peter Shirley William B. Thompson
School of Computing, University of Utah

Abstract

We render tree foliage levels of detail (LODs) using a new adaptation of *billboard clouds*. Our contributions are a simple and efficient billboard cloud creation algorithm designed specifically for tree foliage, and a method for smooth transitions between LODs. The cloud creation algorithm performs stochastic search to find a set of billboards that approximate the base mesh. Billboard clouds offer an alternative to traditional triangle reduction methods, which break down for foliage with its small, disconnected pieces of geometry. By projecting foliage geometry onto large precomputed textures, our method shifts the bulk of the runtime rendering to the fragment processing stage. This results in higher framerates for most viewing distances, with adjustable visual accuracy. We give results for foliage from two fully detailed tree models and discuss implementation issues.

1 Introduction

It is challenging to render large amounts of vegetation interactively on current graphics hardware. Foliage in particular presents unique challenges: a mesh representing a mature tree may contain 10K to 30K small, unconnected leaves distributed in a volume. In an outdoor scene, tens or even thousands of trees might be visible per frame, producing 6 million to 1.8 billion or more triangles per second (TPS) at interactive rates. Throughput on the order of 100 million TPS is perhaps possible on high-end hardware, given careful tuning. However, most interactive applications such as games can afford to spend only a fraction of their rendering time budget on vegetation.

Traditional triangle reduction methods fail when applied to foliage. Occlusion culling algorithms require large planar occluders, yet leaves are all small, and form complex spatial arrangements. Mesh simplification algorithms require connected meshes, yet leaves are mostly unconnected.

We propose that foliage can be rendered efficiently using a new adaptation of *billboard clouds* [5]. A *billboard* is a quad with a fixed position and orientation, and a partly transparent texture mapped onto it. A *billboard cloud* is a set of billboards which approximates a base triangle mesh. Billboard planes are chosen to align with nearly co-planar triangles in the base mesh. Triangles are projected and rendered onto the nearest billboard to create the billboard textures. Traditional billboard clouds have some nice properties:

- **Adjustable Accuracy** A billboard cloud can approximate the base mesh to a desired visual accuracy, depending on the number of billboards and the texture resolution. In practice, acceptable accuracy is possible with less than 100 billboards.
- **Realistic depth cues:** Billboard clouds are stationary, and occupy a volume of space. They provide correct motion parallax for unrestricted camera movements.
- **Anti-aliasing:** Billboards anti-alias small, adjacent triangles in the base mesh via mipmapping. This is very useful for our purposes, since foliage exhibits aliasing when rendered as discrete leaves unless super-sampling is performed.

We adapt billboard clouds to foliage rendering with the following contributions:

- **Stochastic creation algorithm:** Instead of doing a principled optimization, we randomly search the space of billboards. In practice our algorithm produces visually acceptable foliage LODs after a few minutes of computation.
- **Recursive Levels of Detail:** Our billboard cloud creation algorithm may be given a billboard cloud as input, producing a second billboard cloud at a lower LOD. Recursive billboard clouds seem possible with earlier algorithms, but have not been discussed explicitly.
- **Smooth transitions:** We linearly interpolate vertices to transition between the base mesh and billboard clouds at lower LODs, without popping. Transitions were not addressed in previous work because triangles were projected onto multiple billboards.

The algorithm acts on a base mesh which represents foliage as an unordered list of triangles (a "triangle soup"). One or more triangles make up a leaf, which also has an optional texture map. Leaves are assumed to be unconnected. We do not deal with more connected plant parts such as trunks, stems, or branches, for which effective mesh simplification algorithms [8, 9] already exist.

As an example, Figure 1 shows screenshots of a spruce tree rendered in shaded and wireframe modes, just before and just after a transition from base mesh to billboard cloud. The base mesh on the left contains about 20K triangles, is vertex bound, and renders at about 60 FPS. The corresponding billboard cloud on the right contains 78 billboards, is fill bound, and renders at about 300 FPS. Both shaded images are identically gamma-corrected, occupy the same screen area (about 421 by 512 pixels), and are nearly indistinguishable.

2 Background

A great deal of research has been done on LOD rendering, including mesh simplification, image-based, and point-based rendering. We give a brief, incomplete overview of methods targeted at vegetation.

Many interactive games have represented entire trees with *sprites*, or single billboards which always face the camera. Sprites have incorrect depth cues and do not allow general camera motions (e.g., both flyovers and walkthroughs) unless textures are dynamic. A recent game engine [18] uses multiple sprites for clusters of leaves. This results in surprisingly convincing foliage, especially with wind effects, but correct motion parallax is still missing. Game artists have built trees using ad hoc collections of sprites or billboards [11].

Generalized billboard methods capture some of the depth complexity of the base mesh using sets of billboards. Layered depth images (e.g., [12, 16]) consist of parallel, alpha-mapped billboards arranged along the line of sight. These only give correct depth cues for limited, distant viewpoints. Smooth transitions seem possible, but are not addressed. Image-based rendering (IBR) approaches, which dynamically switch billboard textures to account for changes in viewpoint and lighting, have also been proposed (e.g., [14]). However, large amounts of stored texture are required to render nearby trees from general viewpoints.

Billboard clouds [5] use static textures and exhibit correct motion parallax from general viewpoints. A discrete optimization algorithm was originally used to fit billboards to the base mesh. Steps were taken to minimize cracks and favor tangential planes, to approximate connected meshes. Subsequent papers proposed other optimization methods [1, 10, 13], and suggested mixing connected meshes for tree trunks and billboard clouds for tree leaves [2]. We recently learned of work which extends the original billboard cloud algorithm specifically for trees [7], and briefly mentions recursive LODs but does not describe smooth transitions between them.

We also note that point- and line-based methods have been used for rendering foliage (e.g., [19, 6, 4]). Points are not well supported in current hardware, but these methods seem promising.

3 Algorithm

We use a random search algorithm to fit billboards to triangles. Stochastic algorithms, including random search, simulated annealing and genetic algorithms, can find practical, approximate solutions to difficult optimization problems [17].

We could uniformly sample the space of planes that intersect the bounding sphere of the base mesh, but this is expensive. As triangles are matched by billboards and removed, the mesh becomes increasingly sparse. This decreases the likelihood that a sample plane provides a good fit for any triangles at all. Instead, we sample more efficiently by basing each plane on a "seed" triangle chosen at random. Pseudo-code is given in Algorithm 1. The inputs to the algorithm are N, the number of sample planes evaluated per seed triangle, and ϵ, the maximum perturbation of each seed triangle vertex in object space. Briefly, the steps of the algorithm are as follows:

- Select a random seed triangle T_s from the base mesh
- Perturb the vertices of T_s by independent random distances ($-\epsilon < d_j < \epsilon$) along the triangle normal, and consider the plane B supporting these three locations.
- Evaluate plane B by iterating over the remaining triangles T in the mesh. If all vertices $\vec{v}_j$ of T are contained in a slab of width 2ϵ about plane B, then project T onto the plane. Sum the total projected areas for all triangles. If this is higher than the current maximum area, then save B as B_{max}, the best plane found so far. (Figure 2)
- Repeat the steps above N times and save the plane B_{max} with the most projected area. Remove triangles matched by B_{max} from the mesh.
- Repeat the steps above until all triangles are matched by some billboard. The total number of billboards is not a parameter; it is implicitly determined by ϵ and N.

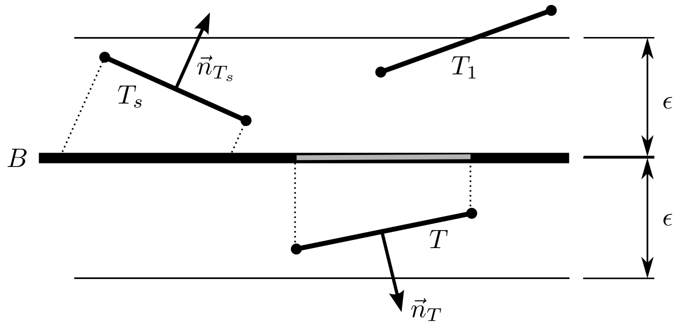


Figure 2: Seed triangle T_s, triangle T that is projected onto billboard plane B, and triangle T_1 that is not projected because some of its vertices do not lie within distance ϵ of B

We create the texture for a billboard by first trimming the billboard to fit its projected triangles (e.g., using [3]), then rasterizing the triangles. We align the image plane with the billboard by placing the virtual camera at the billboard center, setting the viewport to match the billboard dimensions, and using orthogonal projection. The texture (viewport) dimensions are scaled relative to the texture of the largest board, so texture resolution per unit area remains constant over the billboard cloud. Ambient and diffuse lighting are baked into the textures. To correctly capture directional lighting, separate textures are created for the front and back sides of a billboard. The textures are read back from the frame buffer and packed onto one or more large textures which are stored on disk.

Algorithm 1 Random search for billboard planes

```
Input: Mesh, $N$, $\epsilon$
Output: Set of billboard planes
  repeat
    for $i \leftarrow 1$ to $N$ do
      maxarea $\leftarrow 0$
      $T_s \leftarrow$ random "seed" triangle in Mesh
      $\vec{n}_{T_s} \leftarrow$ normal of $T_s$
      for all vertices $\vec{v}_j$ in $T_s$ do {perturb triangle vertices}
        $d \leftarrow$ random real number in $(-\epsilon, \epsilon)$
        $\vec{p}_j \leftarrow \vec{v}_j + d \cdot \vec{n}_{T_s}$
      end for
      $B \leftarrow$ makeBillboardPlane($\vec{p}$)
      $\vec{n}_B \leftarrow$ normal of $B$
      $r_B \leftarrow$ distance from origin to $B$ in normal direction
      area $\leftarrow 0$
      for all triangles $T$ in Mesh do {project triangles onto $B$}
        if $|r_B - \vec{v} \cdot \vec{n}_B| < \epsilon$ for all vertices $\vec{v}$ in $T$ then
          area $\leftarrow$ area + area($T$) $\cdot |\vec{n}_B \cdot \vec{n}_T|$ {increment projected area}
          save reference to $T$ with $B$
        end if
      end for
      if area > maxarea then
        $B_{max} \leftarrow B$
      end if
    end for
    add billboard $B_{max}$ to the set of output billboard planes
    remove triangles referenced by $B_{max}$ from Mesh
  until Mesh is empty
```

We can generate a LOD hierarchy by applying this algorithm recursively, using the billboard cloud at each level as input to produce a billboard cloud at the next level. To do so, we choose seed billboards at random, and use any three of their vertices as the seed triangle. The LOD parameter is ϵ; increasing it produces fewer and coarser billboards. The texture resolution also should be decreased for each new level. We found that increasing ϵ by at least a factor of two and halving the texture dimensions at each level produces good results. The very last LOD is a special case; we choose two perpendicular billboards orthogonal to the ground plane and cutting the center of the tree bounding sphere.

Smooth transitions are achieved by linearly interpolating, or re-projecting, vertices. Refer to Figure 3, which shows the transition from a triangle to the *parent* billboard at the next lowest LOD. When the triangle reaches a distance threshold from the camera ($a = 0$), we begin sliding its vertices toward their orthogonal projections on the parent billboard. When the interpolation is finished ($a = 1$), the vertices all lie in the billboard plane, so the silhouette does not "pop" when we stop drawing the vertices and begin drawing the billboard. The transition is only noticeable if it occurs over too short a distance, or if the parent billboard has insufficient texture resolution. The distance computation and interpolation can be efficiently implemented in a vertex program.

4 Results

We applied the billboard cloud algorithm to spruce (Figure 4) and maple (Figure 5) foliage from a commercial tree library [20]. Each model required about two minutes of preprocessing on a modern PC to create two billboard cloud LODs. We show triangle and texture statistics for the base model and the two billboard clouds, and give measured framerates (FPS) vs viewing distance. We also give values of N and ϵ for each LOD, with ϵ in units of R, the bounding sphere radius of the foliage. We found by trial-and-error that setting N to about 500 produced an acceptable billboard cloud in less than a minute.

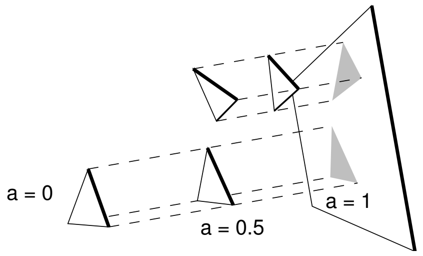


Figure 3: Transition from triangles to the parent billboard using linear interpolation.

The x-axis of each performance plot measures Z, the distance from the virtual eyepoint to the center of the unitized mesh bounding box, plotted on a log scale. At distance $Z = Z_0 = 1/(2\tan(\theta))$, where θ is the field of view, the tree exactly fits the height or width of the screen. We only consider distances $Z \geq Z_0$, and assume that applications will use the base mesh for $Z < Z_0$.

Data was collected from an OpenGL-based renderer and an NVidia GeForce 5900 graphics card, using a scene-graph library [15] to manage resources. We enabled 8:1 (DXT1) hardware texture compression. Frame rates were measured in software and averaged over half-second intervals. The viewing direction was fixed, and parallel to the ground plane.

The advantage of using fill bound billboard cloud LODs is immediately clear from the performance plots. The base mesh is already vertex bound at distance Z_0, so the framerate remains constant at all further distances. However, the billboard cloud framerate increases according to a power law ($Z^p, 1 < p \leq 2$), until it becomes vertex bound at some distance (e.g. $log(Z/Z_0) \approx 0.55$ for the spruce, and $log(Z/Z_0) \approx 0.35$ for the maple). Applications can substantially increase framerate by switching from the base mesh to a billboard cloud.

It is interesting to do a back-of-the-envelope memory footprint comparison for the base mesh and the billboard clouds. For example, the spruce base mesh has about 20K triangles. If we assume that a triangle requires 18 floats (1 face normal, plus 3 position coordinates and 2 texture coordinates per vertex), and each float is stored in a 2 byte compressed format, then the spruce triangles use about 720 KB. The footprint of a billboard cloud is dominated by the packed textures. The first-level billboard cloud uses 1.33MB of texture memory, assuming 8:1 fixed-ratio compression, and 33% extra space for mipmaps. The second-level cloud uses 166 KB of texture memory. So the first-level cloud is 85% larger than the base mesh, and the second-level cloud is only 23% as large as the base mesh. Current graphics cards have 256 MB of video memory, which is probably sufficient to keep billboard textures resident along with other application textures.

5 Discussion

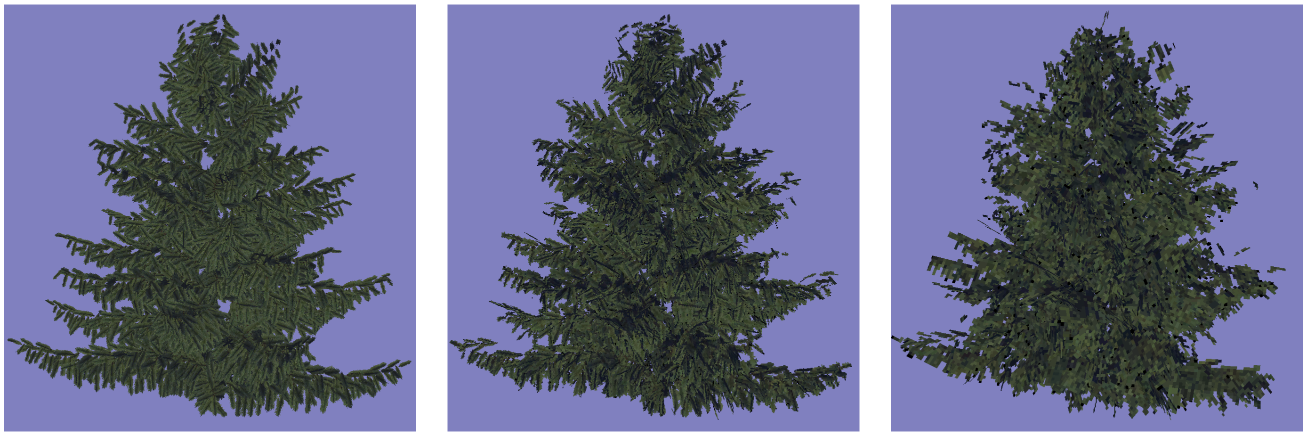
We encountered a number of practical implementation issues. First, when a large billboard is viewed at a grazing angle during camera motions, the planarity of the billboard is sometimes noticeable. Isotropic mipmapping further overemphasizes billboard edges.

There are several ways to address this problem. Enabling anisotropic texture filtering helps, but at the expense of framerate. Another solution is to cull billboards when the dot product of the unit view vector, $\vec{e}$, and the billboard normal, $\vec{n}_B$, is near 90 degrees ($|1 - \vec{e} \cdot \vec{n}_B| \ll 1$). We could also resume drawing triangles in place of the culled billboard, perhaps interpolating the triangles toward their original positions in the base mesh.

Slight modifications to Algorithm 1 can improve framerate and visual quality. Limiting the size of billboards often reduces the number of transparent fragments. This limit can be enforced by only projecting triangles that are within a given distance of the seed triangle for a billboard. Also, if a model will be viewed from a limited set of angles, visual quality is improved by favoring billboards which have a high probability of facing the camera.

Texture packing is important for minimizing context switches. We used a simple binary-split packing heuristic, looped over random texture sequences, and achieved packing ratios of about 85%.

Better hardware mipmapping and compression would be useful, e.g., the ability to mipmap non-power-of-two textures. Default hardware mipmapping produced dark fringes around small images on our billboard textures. Evidently some transparent pixels on the packed texture were blended with opaque pixels. We corrected this with custom



Base	BC1	BC2
20610 triangles	78 quads (156 triangles)	43 quads (86 triangles)
256×128 texture	Two $2K \times 2K$ textures	$1K \times 1K$ texture
	$N = 600, \epsilon = 0.04$	$N = 300, \epsilon = 0.10$

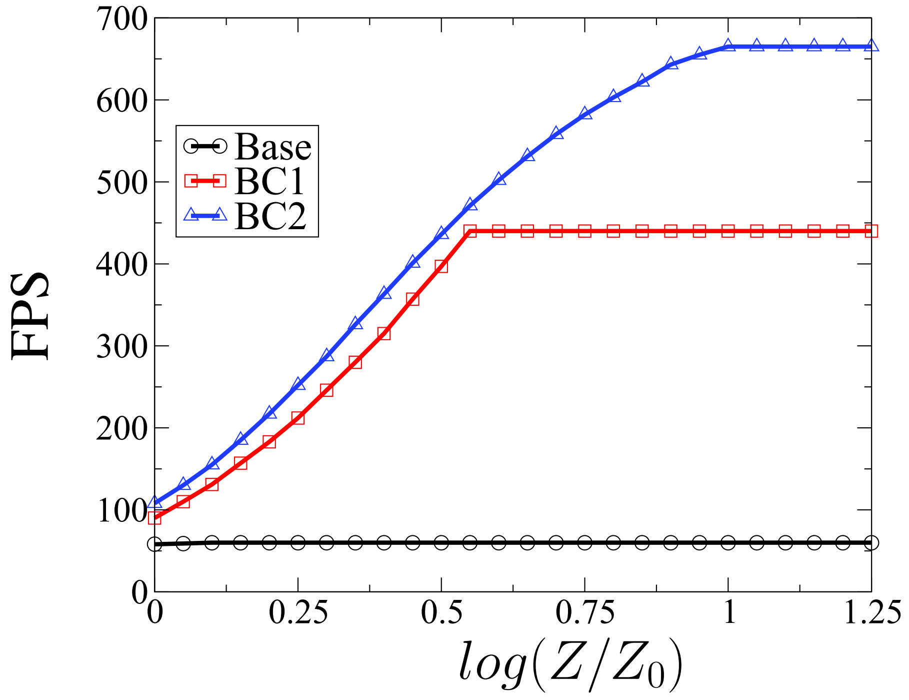
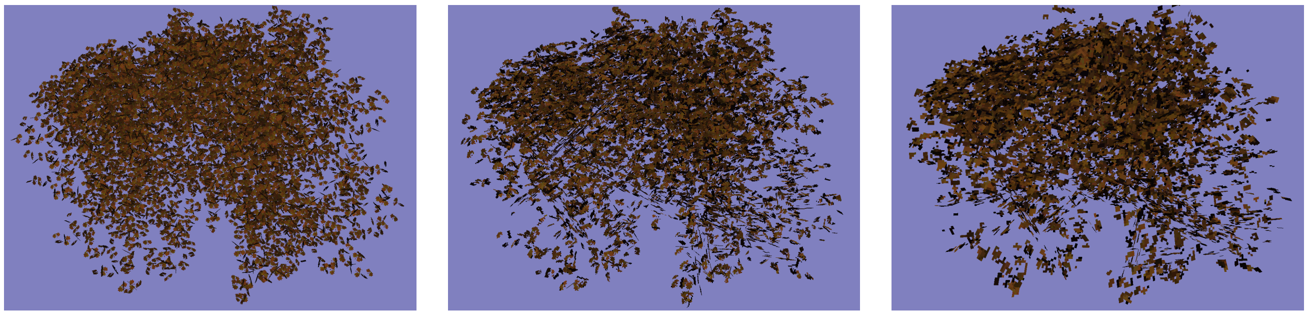


Figure 4: Statistics and log-linear performance plot for spruce tree base mesh (Base) and billboard clouds (BC1, BC2). Z is the viewing distance, and Z_0 is the minimum distance at which the entire model is in the view frustum. The base mesh is vertex bound for $Z > Z_0$. The billboard clouds are fill bound up to a limiting distance, and thus achieve much higher framerates.



Base	BC1	BC2
12904 triangles	132 quads (264 triangles)	43 quads (86 triangles)
256×128 texture	$2K \times 2K$ texture	$1K \times 1K$ texture
	$N = 500, \epsilon = 0.04$	$N = 500, \epsilon = 0.10$

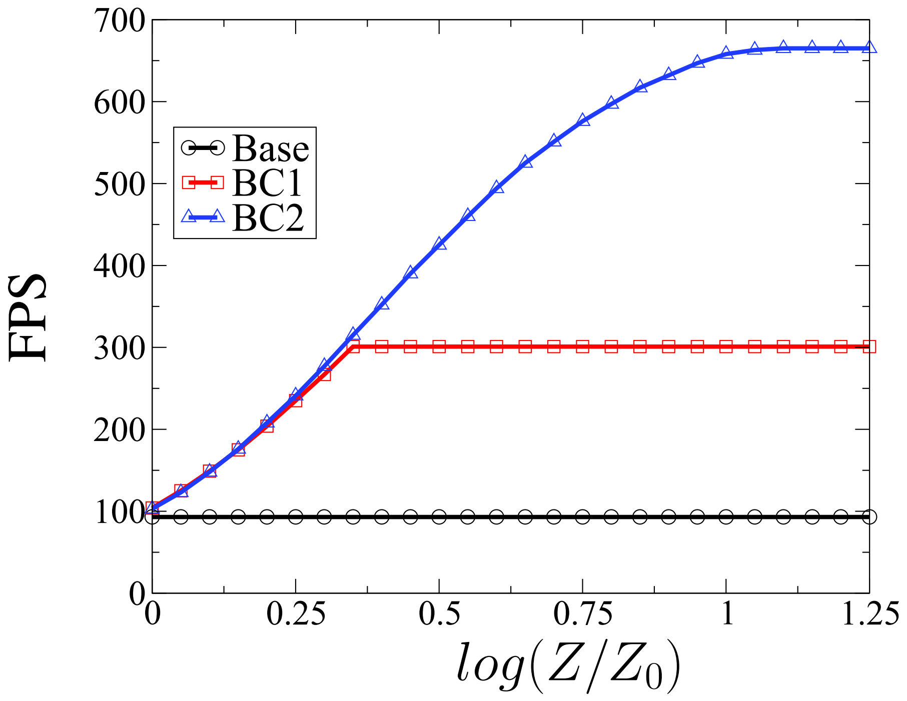


Figure 5: Statistics and log-linear performance plot for maple tree base mesh (Base) and billboard clouds (BC1, BC2).

mipmap code which rounds alpha values to either 0 or 1.

Since lighting is baked into billboards during preprocessing, we would like to experiment with advanced lighting models, e.g., ambient occlusion, or path-traced global illumination. The base mesh would still use a local lighting model, so smooth transitions to it would be a problem. Still, the first-level billboard cloud might provide enough detail for some applications. Specular highlights could be added to billboard textures to simulate glossy leaves, provided that we save a normal map with each billboard.

We wish to reiterate that our billboard cloud algorithm was not designed for continuous meshes (e.g., the Utah teapot), for which effective triangle reduction methods already exist. Neighboring triangles in a continuous base mesh may project onto different billboards in the cloud, resulting in visible surface cracks. Steps could perhaps be added to the algorithm to reduce cracking, but we prefer to keep it simple. Cracks are not an issue for most foliage.

Finally, we believe that billboard clouds would also improve performance in a ray tracing program. Intersecting rays with a billboard cloud should be substantially faster than intersecting with the base mesh, and billboard texture anti-aliasing should reduce the required number of rays per pixel.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0312479.

References

[1] ANDÚJAR, C., BRUNET, P., CHICA, A., ROSSIGNAC, J., NAVAZO, I., AND VINACUA, A. Computing maximal tiles and application to impostor-based simplification. *Comput. Graph. Forum 23*, 3 (2004), 401–410.

[2] BROMBERG-MARTIN, E., ÁRNI MÁR JÓNSSON, MARAI, G. E., AND MCGUIRE, M. Hybrid billboard clouds for model simplication. In *Proceedings of the 31st annual conference on Computer graphics and interactive techniques* (2004).

[3] CGAL library, 2004. http://www.cgal.org.

[4] DACHSBACHER, C., VOGELGSANG, C., AND STAMMINGER, M. Sequential point trees. *ACM Transactions on Graphics 22*, 3 (July 2003), 657–662.

[5] DÉCORET, X., DURAND, F., SILLION, F., AND DORSEY, J. Billboard clouds for extreme model simplification. In *Proceedings of the ACM Siggraph* (2003), ACM Press.

[6] DEUSSEN, O., HANRAHAN, P. M., LINTERMANN, B., MECH, R., PHARR, M., AND PRUSINKIEWICZ, P. Realistic modeling and rendering of plant ecosystems. In *Proceedings of SIGGRAPH* (1998), pp. 275–286.

[7] FUHRMANN, A., UMLAUF, E., AND MANTLER, S. Extreme model simplification for forest rendering. In *Natural Phenomena 2005: Eurographics Workshop on Natural Phenomena* (2005), Eurographics Association, pp. 57–66.

[8] GARLAND, M., AND HECKBERT, P. S. Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 209–216.

[9] HOPPE, H. Progressive meshes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM Press, pp. 99–108.

[10] HUANG, I.-T., NOVINS, K., AND WUENSCHE, B. Improved billboard clouds for extreme model simplification. In *Proceedings of Image and Vision Computing* (2004), pp. 255–260.

[11] LUEBKE, D., REDDY, M., COHEN, J. D., VARSHNEY, A., WATSON, B., AND HUEBNER, R. *Level of Detail for 3D Graphics*. Morgan Kaufmann, 2002.

[12] MAX, N. Hierarchical rendering of trees from precomputed multi-layer z-buffers. In *Eurographics Rendering Workshop* (June 1996), pp. 165–174.

[13] MESETH, J., AND KLEIN, R. Memory efficient billboard clouds for btf textured objects. In *Vision, Modeling, and Visualization 2004* (November 2004), B. Girod, M. Magnor, and H.-P. Seidel, Eds., Akademische Verlagsgesellschaft Aka GmbH, Berlin, pp. 167–174.

[14] MEYER, A., NEYRET, F., AND POULIN, P. Interactive rendering of trees with shading and shadows. In *Proceedings of the Eurographics Workshop on Rendering* (2001), pp. 183–196.

[15] OpenSceneGraph software, 2005. http://www.openscenegraph.org.

[16] SHADE, J., GORTLER, S. J., HE, L. W., AND SZELISKI, R. Layered depth images. In *Proceedings of SIGGRAPH 98* (July 1998), Computer Graphics Proceedings, Annual Conference Series, pp. 231–242.

[17] SPALL, J. *Introduction to stochastic search and optimization : estimation, simulation, and control*. Wiley-Interscience, 2003.

[18] SpeedTreeRT software (Interactive Data Visualization, Inc.), 2005. http://www.idvinc.com/.

[19] WEBER, J., AND PENN, J. Creation and rendering of realistic trees. In *Proceedings of SIGGRAPH* (Aug. 1995), pp. 119–128.

[20] XFrogPlants model collection, 2005. http://www.greenworks.de/.