EurographicsWorkshop on Natural Phenomena (2005),
E. Galin, P. Poulin (Editors)

Extreme Model Simplification for Forest Rendering

Anton L. Fuhrmann, Eike Umlauf and Stephan Mantler

VRVis center for virtual reality and visualization

Abstract

Models of large forest scenes are of a geometric complexity that surpasses even the capabilities of current high end graphics hardware. We propose an extreme simplification method which allows us to render such scenes in real-time. Our work is an extension of the image based-simplification method of Billboard Clouds. We automatically generate tree model representations of 15-50 textured polygons. In this paper, we focus on the algorithmic details to improve the simplification process for foliage. We use the simplified models as static levels-of-detail in the medium to far field and demonstrate how our approach yields real-time rendering of dense forest scenes for walk-throughs and flyovers.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Curve, surface, solid, and object representations; I.3.5 [Computer Graphics]: Geometric algorithms, languages, and systems; I.3.8 [Computer Graphics]: Applications -- Real-time visualization

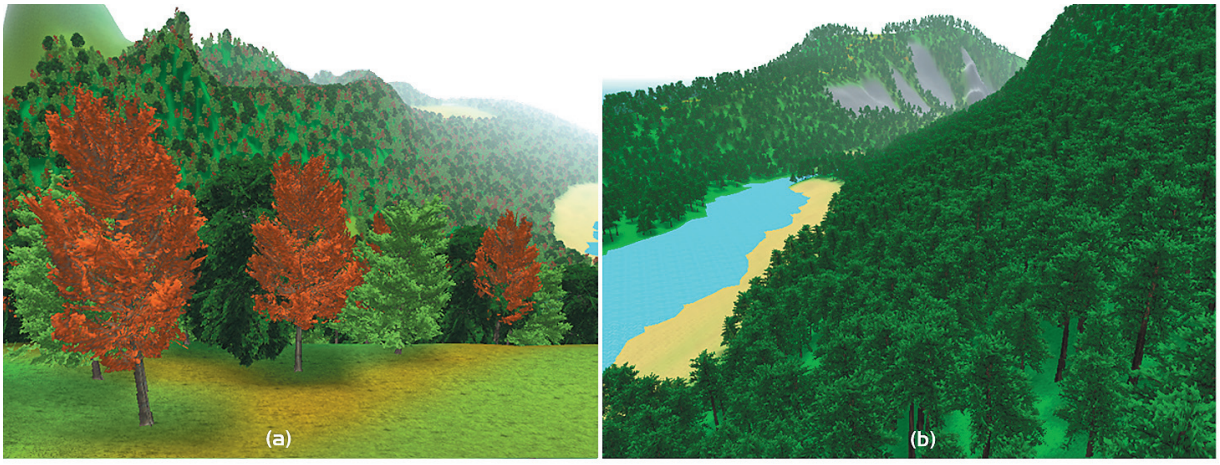


Figure 1: *Sample screen shots of real-time rendered billboard cloud forests*

1 Introduction

One of the biggest challenges of three-dimensional computer graphics is the generation and real-time rendering of vast and realistic outdoor sceneries, especially forests.

While the modeling of polygonal vegetation is a rather well-researched topic and there exists a palette of specialized tools which allows the rapid creation of realistic tree models, those consist of a large number of polygons. For example, a 50.000-tree forest of models with average detail of 200.000 polygons per tree amounts to 10 billion polygons, which prohibits real-time rendering of such a forest. Therefore, most tree rendering techniques are still either slow or of insufficient visual quality.

Our approach to forest rendering is based on the image-based *billboard cloud* simplification algorithm as introduced by Décoret [DEC02, DEC03]. We introduce a number of adaptations to improve the quality of the simplification, especially for tree models. We use this extended algorithm to generate image-based representations of arbitrarily complex tree models in various levels of details during a preprocessing step.

The resulting billboard clouds consist of sufficiently few primitives to allow rendering of tens of thousands of trees at interactive rates, with a visual quality high enough for walkthrough applications. Figure 1 shows two different forests rendered this way in real-time, with a slight OpenGL fog as depth cue. Every visible tree in this image is an individual billboard cloud, no additional simplification techniques have been used.

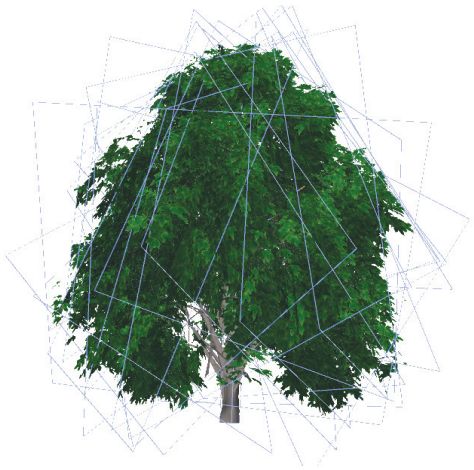


Figure 2: *Billboard Cloud tree composed of 25 billboards (50 triangles). Original model consists of 159.853 triangles.*

2 Related work

A popular approach to the modeling of vegetation is the use of Lindenmayer-systems and generalizations thereof [LIN90, TOB02], while other works blend grammar-based modeling with traditional techniques [WEB95, DEU99]. A number of commercial products such as XFrog [XFR], natFX [BIO] or SpeedTreeCAD [IDV] are available that generate high-quality polygonal or hybrid polygon/image-based models.

From a rendering point of view, various techniques are used to create immersive forests, with mixed success.

A very popular image-based approach is billboarding, where a tree is usually represented by a single quadrilateral with a tree texture applied. In the view-aligned case, the billboard always faces the user. Inspected closely, this looks very unrealistic because of lacking depth and parallax. Therefore, several billboards with fixed orientation are often set up orthogonally to produce a more three-dimensional impression, resulting in a "cardboard" look. Our approach essentially produces results like this, but with independent orientation and automatic generation of the billboards.

Dynamic impostors [SCH95, SCH97] are view-aligned billboards whose texture is dynamically updated by rendering the original polygonal model to texture depending on the viewing direction every few frames. Doing this for a high-polygon model every few frames is still expensive, but performing the impostor update for an entire forest of such models is prohibitively slow.

Precomputed impostor approaches [CHE93, MAX95, MEY01] avoid the dynamic texture regeneration by interpolating images from a stored set of viewpoints, but these image-warping operations are relatively slow and - when inspected from up close - the same limitations as with view-aligned billboards apply.

Low-polygon modeling in combination with texturing yields more solid-looking trees, but is considerably slower than billboarding. Besides, visual quality suffers if very few polygons are used.

Levels of detail and multiresolution solutions are frequently used in conjunction with above techniques, but generating multiple levels of detail for trees automatically [HOP93, TUR00] is a non-trivial task, and doing so manually is labor-intensive.

Hybrid approaches try to combine the advantages of polygonal and image-based modeling:

IDV's SpeedTree middleware engine [IDV] merges low-polygon stems and view-aligned billboard foliage to achieve a good tradeoff between speed and visual quality. While providing visually pleasing result in many cases, the billboards pose problems when viewed from certain angles where the foliage suddenly seems to rotate, and rendering times do not allow dense forests.

Remolar et al [REM02] rely on multi-resolution polygonal and impostor representations of the foliage. The main problem lies in obtaining low-detail representations at sufficient quality.

Point-based rendering of trees [DAC03, MAN03] is effective for distant objects, but visually unacceptable up close.

Harnessing the texturing power of current graphics hardware, Decaudin et al [DCD04] utilize aperiodic tiles

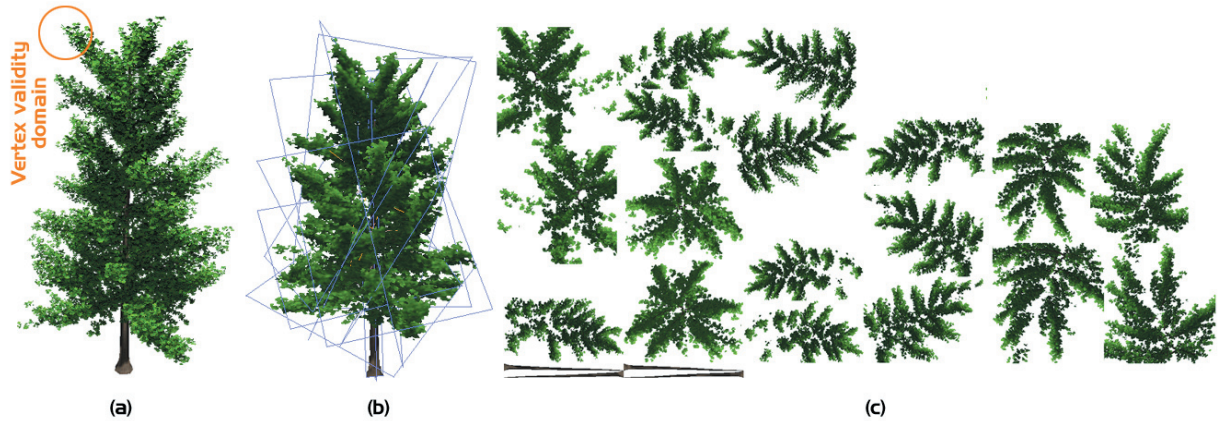


Figure 3: *A 110,000 polygons tree model (a) simplified to 11 billboards (b). The error threshold is shown by the vertex validity*

of volumetric textures to achieve vast and dense forests. This approach does not allow independent placement or rotation and scaling of individual trees and is not suited for walkthroughs. Furthermore, existing rendering engines do not support this technique and would have to be upgraded, whereas the ability to draw billboards is implemented in virtually any engine, thus making our approach more versatile.

Billboard clouds, introduced by Décoret [DEC02, DEC03], are a means of extreme simplification, where the polygonal input model is simplified to a minimum set of textured planes that feature independent size, orientation and texture resolution, which basically equates to a set of static billboards, as mentioned before. The working and shortcomings of the basic algorithm are discussed in the next sections.

A similar approach by Andujar et al [AND04] produces comparable output, but relies on a different computation process that apparently cannot simplify non-manifold tree models because it involves volume inside/outside tests.

All of the previously discussed approaches have their advantages and shortcomings, making them appropriate for different applications. Similarly our solution is just another compromise between realism and speed, which addresses only some of the problems of forest rendering, namely model simplification, level-of-detail generation and lighting for static trees.

3 Methodology

Section 3.1 briefly explains the original billboard cloud algorithm; for a detailed derivation we refer to the original papers [DEC02, DEC03]. We address problems of the simplification process in section 3.2 and introduce our own adaptation for foliage simplification in section 3.3.

3.1 The original billboard cloud algorithm

The idea of the simplification process is that a high-polygon, textured input model is projected onto a set of planes that approximate the original geometry within a defined error bound. The most difficult and computationally expensive part of the algorithm is to derive this plane set.

The quality of the simplification is determined by the error threshold ε, measured in percent of the model's bounding sphere radius. A vertex can only be simplified to a plane if its normal distance is less than ε. Therefore, the spherical region around a vertex with radius ε is its validity domain **V**. If a plane intersects the validity domains of all vertices of a face, it is declared *valid* for the face (i.e. it can simplify the face) and vice versa. There is an infinite number of valid planes for each face.

The simplification process can be formulated as a clustering problem in a dual space constructed by the *Hough transform* [DUD72], in which planes (in primal space) map to individual points. In this dual space, the ground plane can be visualized as varying θ and φ, i.e. the orientation of the plane (coordinate limits are ±180° and ±90°, respectively), while the up axis represents the distance from the origin ρ (≥0).

A vertex in primal space can be described as the intersection point of all planes passing through the vertex. Thus, a point in primal space becomes an infinite set of points in dual space, more conveniently represented by a sheet, or height field. The vertex validity domain simply is the region between this sheet translated up/down on the ρ axis by ε.

Consequently, a face is not only valid for a single point in dual space, but for the intersection of its vertex validity domains.

For the numerical evaluation, dual space is represented as a grid with a finite number of cells. Grid cells are marked valid for a face (and vice versa) if they lie within its discretized validity domain. The discretized validity domains of the faces of the input model are accumulated to a value called *density* (**D**). To give more weight to larger faces and promote planes tangent to large faces, **D** is calculated as a contribution factor **C**, which is the "geometric coverage" of a face projected to the plane at the center of a cell. A penalty factor P is then subtracted for planes that nearly miss vertices; note that a large penalty factor may eradicate **D**.

Once all values of **D** have been computed, a recursive greedy algorithm searches the dual grid for the cell with maximum **D**. This cell is further recursively subdivided until the best plane is found. Both **V** and **D** need to be calculated for each set of the recursion, which makes this part of the algorithm very expensive. After the best plane has been found, the **D** contributions of the simplified faces are

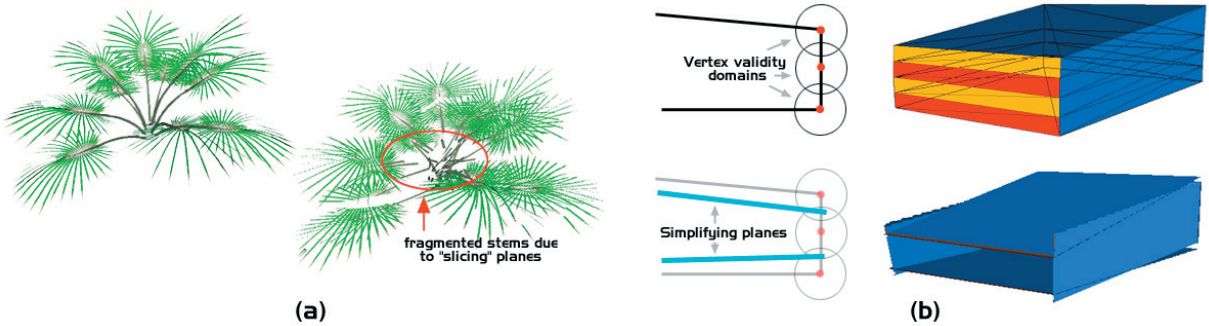


Figure 4: *Billboard Cloud artifacts: (a) loss of continuity. (b)The front faces of the box should ideally be simplified to one vertical plane. However, the dominant top and bottom planes preemptively simplify the front faces and leave a large hole in the*

removed from the dual grid.

Repeating the greedy procedure until all faces have been simplified yields the finished plane set.

Finally, plane textures are generated by projecting the faces of the input model to their respective simplifying plane.

3.2 Problems of the Algorithm

In general, the simplification process is quite demanding on memory and CPU time. Unfortunately the behavior of the algorithm, especially for working with tree models, can be hard to predict. Parameter values that produce an excellent result for one model might work less well on another. Worse, the algorithm as outlined in [DEC03], frequently fails to produce simplification results or delivers mathematically correct, but visually unpleasing solutions.

3.2.1 Numerical issues. A primary source of problems are numerical issues with the algorithm. Discretization of a continuous volume to a set of cells is a non-trivial and expensive task that requires some trade-off between speed and precision. The method used to discretize face validity domains actually discretizes the corresponding vertex validity domains, then intersects them. While this is computationally effective, it is known to produce false positives where a face may be marked valid for a grid cell although it is not. Consequently, the greedy algorithm fails to find a plane that simplifies this false positive, potentially causing the algorithm to terminate without returning a simplification.

Another problem surfaces if a very small value is chosen for ε. Because of the imprecise discretization of validity domains, the recursive algorithm might visit sub-cells that, due to dominant penalty, feature zero **D**, but seemed promising in the previous step. The algorithm will not know which sub cell to pick in order to continue and once again cannot produce a correct solution.

Both issues are similar and the less-than-perfect discretization step is the major cause, but since the discretization of validity domains is done very often (especially during the greedy select phase), the increased computational cost of employing a more precise discretization is prohibitive. Instead, we introduce a fail-safe solution in section 3.3.1.

3.2.2 Sub-optimal planes. It is generally very difficult to simplify rounded shapes to polygons satisfactory using any algorithm, and billboard clouds are no exception. While precomputed shading or normal mapping on the textures help to convey a sense of curvature, often overlapping planes and gaps in the billboard clouds spoil the overall impression.

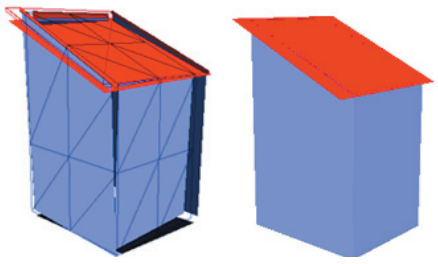


Figure 5: *A simple Billboard Cloud generated without (left, original geometry overlaid) and with (right) post-plane tweaking.*

An important trait of the algorithm is that connectivity of the input model is neither needed, nor retained. For certain cases, this behavior is greatly beneficial. For instance, leaves of a tree model are not interconnected, but it is desirable that many individual leaves are simplified to one plane. On the other hand, planes that simplify different regions of a model may also simplify parts of sub-objects that would look better if taken care of by a different plane.

This phenomenon, which we call "slicing" can be particularly witnessed with trunks and branches of vegetational models (Fig. 4a). To avoid this persistent problem, we split tree models into two parts, one containing the trunk, the other foliage and branches. The simplification results are merged to produce a representation of superior quality than a single pass would yield.

On the other hand, sometimes slicing can be advantageous. Calculations for sub cells include the penalty factor, which is supposed to keep planes to the perimeter of the model to yield a good representation of the "outside". Yet, in order to obtain a solid-looking representation, planes that slice the foliage are desirable, which is why we omit penalty from **D** calculations for sub cells.

Not all models can be simplified by the greedy algorithm. For instance, we can frequently observe cases where dominant planes accidentally simplify smaller faces that would be better handled by a separate plane (Fig. 4b).

Remedies for sub-optimal planes are discussed in sections 3.3.3 and 3.3.5.

3.2.3 Gaps. Gaps may occur in a billboard cloud if a vertex shared by multiple faces is simplified to different planes. Gaps show up especially while simplifying curved surfaces, when simplifying planes do not match or overlap. We present a technique to avoid gaps in section 3.3.4.

3.3 Our Improvements

Thankfully, trees can be relatively good represented as billboard clouds. Except for the trunk, they feature little connectivity that could be disturbed and are mostly devoid of large, curved surfaces, where discontinuities are especially noticeable. Nevertheless, the original algorithm tended to produce complex or mediocre BBC representations of trees. We introduce some extensions made to the original algorithm to enhance the quality of our simplified tree models in sections 3.3.5 to 3.3.7. Additionally, we explain some of the fixes employed to improve general results in sections 3.3.1 to 3.3.4.

3.3.1 Angular Contribution. In general, the contribution component **C** of density **D** aims to keep billboard planes tangent to the original geometry. By default, a face's **C** to a

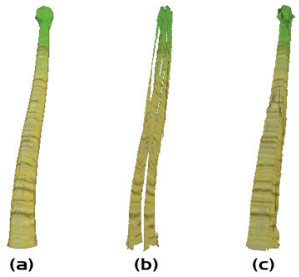


Figure 6: *Palm tree stem polygonal model (a), Billboard Cloud with same polygon count without (b), and with (c) vertex welding.*

plane is its geometric area, projected to the plane. Thus, **C** decreases as the difference of orientation increases, so that non-tangent planes are penalized.

However, it can be observed that employing geometric coverage is not a strong enough metric for measuring deviating orientation between faces and planes, since it works in a non-linear way and varies very little for orientation discrepancies in the range of less than 30° degrees. However, typically faces are valid for a plane almost entirely in this range only. Therefore, a better solution is to derive the angular deviation, i.e. calculate the angle between a face and plane with the vector dot product, and use it as a weight for the unprojected area of a face. Thus, changes in **C** for varying θ, φ are linear and are more successful in keeping planes tangent to the surface of the polygonal model. Fig. 7 demonstrates the effects of projected area **C** and angular **C**.

Plane/Face deviation	Projected area **C**	Angular **C**
0°	100	100
10°	98.4	88.8
30°	86.6	66.6

Figure 7: *Projected area and angular* **C** *of a face (geometric area is 100 units) for planes that differ from the orientation of the face by 10 and 30 degrees. It can be clearly seen that angular* **C** *behaves linearly, while projected area* **C** *is a rather imprecise representation of deviation.*

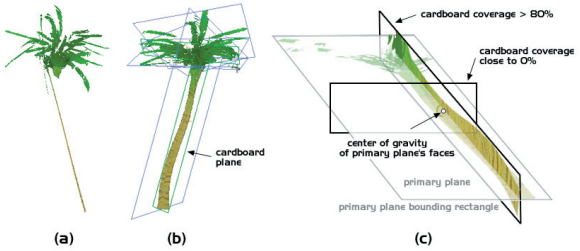


Figure 8: *Employing additional "cardboard" planes to improve the simplification of trunks.*

3.3.2 Fail-safe mode. To ensure that the algorithm always comes up with a simplification result, we employ a fail-safe routine. Whenever the greedy algorithm fails, we iterate on the set of faces valid for the original grid cell with highest **D**, assuming the supporting plane as simplifying plane and testing all other faces for validity against that plane. Additionally, we create an averaged plane as outlined in section 3.3.3. The plane that can simplify the most faces is then accepted.

3.3.3 Post-plane tweaking. The recursive greedy algorithm terminates as soon as a sufficient plane is found, but that does not guarantee that said plane is visually optimal. In fact, often a plane is quite out of orientation compared to the faces it simplifies, so we attempt to correct this flaw by tweaking position and orientation of a plane as follows:

We calculate the average of all normal vectors of the faces simplified by the plane, weighted by the geometric area covered by the respective faces, to obtain the normal vector for our refined plane. To find its new position, we simply sum up the center points of the faces, again each one weighted by their area, then divide by the total face area. Using area weighting gives more relevance to the orientation and position of large, dominant faces.

Finally, we test if the faces concerned are valid for the refined plane and if the projected faces cover an area equal to or greater than on the previous plane. If both conditions hold

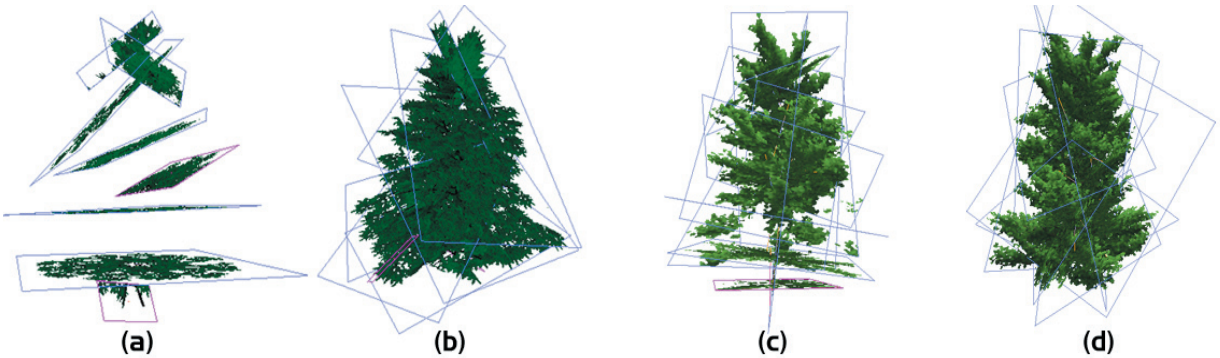


Figure 9: *Large values of ε result in few, horizontal planes, which can be corrected using horizontal-plane penalty. Spruce without (Fig. a, 8 planes) and with horizontal-plane penalty (Fig. b, 9 planes), Tupelo without (Fig. c, 11 planes) and with horizontal-plane penalty (Fig. d, 8 planes).*

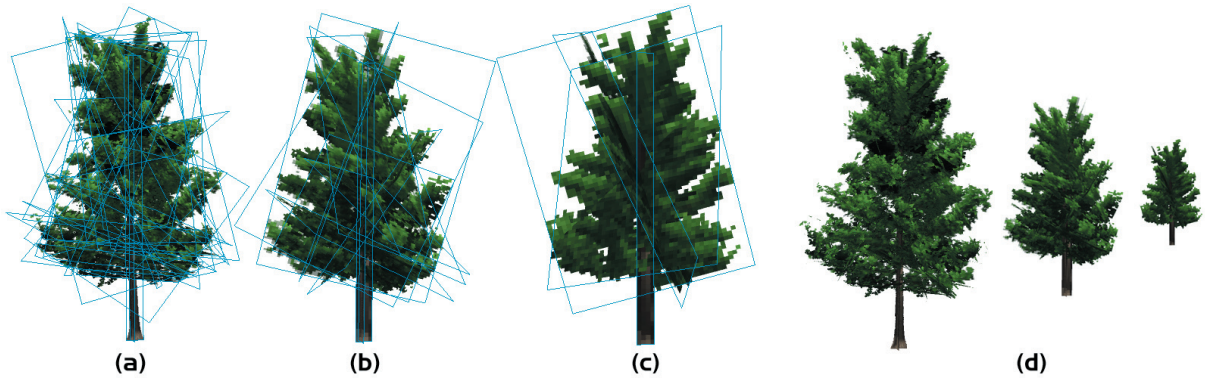


Figure 10 : *Levels of detail for a Billboard Cloud tree. (a) 24 planes, (b) 11 planes, (c) 4 planes. (d) shows the LoDs as they appear in our application.*

true, we use the refined plane instead of the one generated by the greedy select step.

This approach carries a numerical issue that has to be dealt with when working with two-sided faces, which is generally the case with leaves. Averaging the normal vectors of two faces oriented back-to-back effectively cancels their contribution. We therefore limit the allowed directions of normal vectors to one half-space by flipping a normal if it points into the wrong half-space. Put plainly, we want all normal vectors to point roughly in the same direction. We use the untweaked plane normal to define the half-space and flip the normals of the input planes accordingly.

Especially for models with rather flat topology, this approach improves a significant number of planes (Fig. 5), and is more versatile than an optimization handling coplanar polygons only [DEC03].

3.3.4 Vertex welding. Since billboard clouds ignore topology, some continuous structures exhibit irritating gaps when simplified (Fig. 6b). This happens mostly when faces that share vertices are simplified to different planes. If the shared vertices do not by coincidence lie on the intersection of the planes, they are torn apart. We reduce this artifact by permanently displacing ("welding") the vertex in the original model to its first simplifying plane. This generally reduces the gaps for convex objects, since it moves the projected vertex on the other plane closer to their intersection. While it can easily be shown that this does not work in all cases, the new simplified faces tend to overlap for many viewing directions.

This displacement does not violate the notion of validity defined in section 3.1, but **D** and V have to be updated accordingly after a welding operation.

We use this approach when simplifying large trunks and branches.

3.3.5 Horizontal-plane penalty. When simplifying tree models, it can be observed that to achieve satisfying results, ε can vary widely in world space from less than 5% to 15% and beyond. However, a disturbing visual problem occurs for large values of ε. The best simplification in terms of minimizing the amount of planes can then be achieved by collapsing multiple layers of foliage to one almost horizontal plane, which, although correct by definition of the algorithm, is visually unacceptable (Fig. 9a). In many 3D applications, simulations and computer games the vertical viewing angle is rather small, i.e. the viewer is at roughly the same height as the trees, so that purely horizontal planes should be avoided. Completely vertical planes are not advisable either because they would make trees look awkward when viewed directly from above, and many horizontal elements of foliage cannot be captured well by vertical planes.

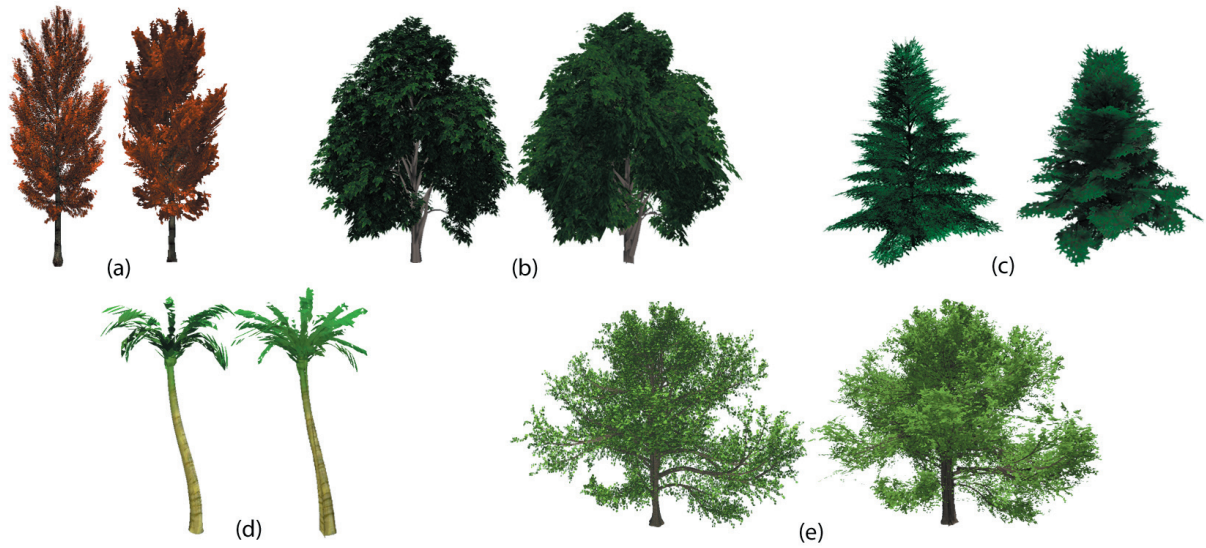
Therefore, a solution is needed that discourages the algorithm from picking horizontal planes, but does not entirely prevent them.

A feasible solution is the manipulation of density values in the dual grid. Recalling that in dual space, regions with $\varphi = \pm 90°$ represent horizontal planes, the desired effect can be achieved by reducing density in those areas. We therefore define a slope function that is imposed on the grid whenever a density or sub cell density read access is being made. The input parameters for this function are:

- Penalty value. Denotes the maximum penalty imposed on **D**, measured in percent. Thus, a penalty of 50% for a grid region reduces its density by half. The maximum value is reached only at the poles of the sampling sphere, that is, for $\varphi = \pm 90°$.
- Penalty cutoff angle. Penalty is linearly interpolated from the maximum penalty value to zero between $\varphi = \pm 90°$ and the cutoff angle.

That way, the closer to being horizontal a plane is, the less likely it is to be picked because its density is kept low. Of course, at some point there may be faces left that can be simplified by horizontal planes only, but this will only be a small remaining subset.

The described extension works well for most classes of tree models and comes with no perceptible performance hit. Empirical tests show that for an average tree model, a penalty value of 60% and cutoff angle of 50° is sufficient (Fig. 9).



Tree	ε	#Faces	#Billboards	Preprocessing Time (s)	approx. trees / s
a	10.0	108,782	12	342	143,000
b	12.0	159,160	14	403	122,000
c	12.5	20,547	13	62	132,000
d	6.5	7,292	8	25	214,000
e	6.5	169,781	21	496	81,000

Figure 11: *Polygonal Trees (left), their Billboard Clouds (right) and performance data. Measured on a 3,2GHz P4, 2GB, GeForce 6800GT machine.*

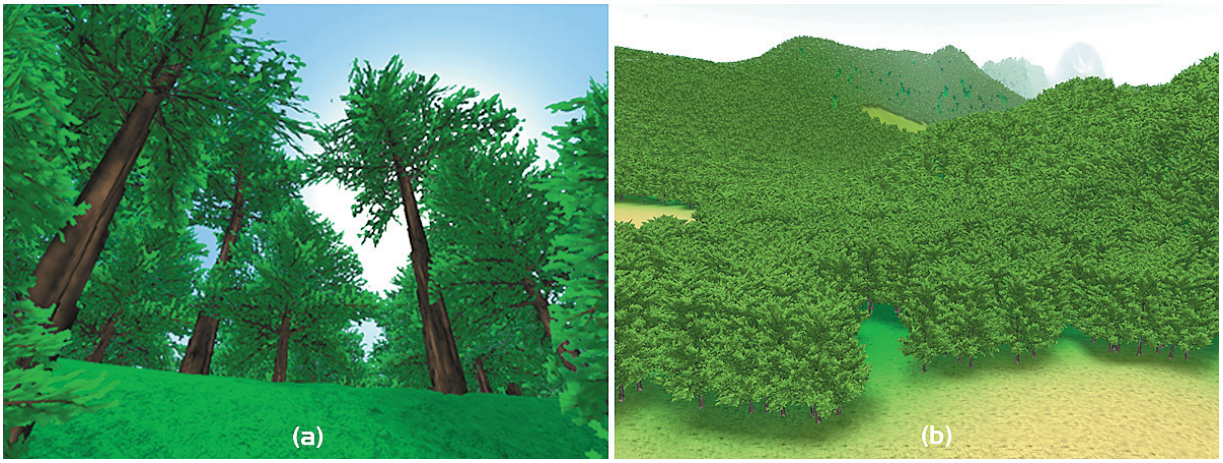


	Fig. 1a	Fig. 1b	Fig. 12a	Fig. 12b
Trees total	80.000	50.000	50.000	150.000
Trees in View	41.007	15.067	67	83.151
Frames per Second	9,3	20,8	142	4,7

Figure 12: *Screenshots of Billboard Cloud forests. Notice precomputed shadows on ground texture. Rendered on a 2GHz Pentium 4, GeForce 6800GT, 2GB RAM machine.*

3.3.6 Cardboard mode. As mentioned in section 2, the "Cardboard look" is an inexpensive modeling technique, but looks unrealistic and is now seldom used.

However, the basic principle is not totally without its benefits. Particularly long, thin objects like the stem of a tree are often simplified to a single plane by the billboard cloud algorithm, which looks obviously flat from the side (Fig. 8a). An additional, orthogonal billboard (dubbed *cardboard plane* in our algorithm) for such planes results in a much better visual impression (Fig. 9b) at very little expense to rendering performance.

The following extension to the algorithm produces such planes automatically.

After the primary plane has been picked in dual space, the center of gravity of its simplified faces projected to the plane is calculated, as well as its bounding rectangle. Calculating the bounding rectangle, which is also necessary for the texture generation step, is performed with the aid of the Jarvis' March a.k.a. gift-wrapping algorithm [JAR72]. We then set up two potential cardboard planes perpendicular to the primary plane that pass through its center of gravity and are parallel to the edges of its bounding rectangle (Fig. 8b). The faces of the primary plane that are also valid for a cardboard plane are projected onto the latter and the area covered by the projected faces is calculated. The ratio of the total area covered on a cardboard plane compared to the area covered on the primary one is called cardboard coverage and is measured in percent of primary plane coverage. The cardboard plane with the larger coverage is accepted, as long as it exceeds the user-specified cardboard coverage threshold.

If the threshold is set low enough, cardboard planes for many other portions of models than long and thin ones are generated. Note that their general use is limited, since only the faces of the primary plane are projected onto a cardboard plane, thus limiting the width of a cardboard plane to 2ε. Furthermore, they do not simplify any additional faces, but serve the sole purpose to be visually enriching.

3.3.7 Texture generation. Generating textures for the billboard cloud planes is straightforward: rotate and translate the input model by the inverse plane parameters and render all faces valid for the plane, using orthogonal projection, into an off-screen buffer the size of the desired texture resolution. Note that rendering all valid faces instead of only the ones simplified by the plane improves the visual solidity of the billboard cloud.

A problem that surfaces when simplifying models with small details using this approach is that a lot of these details are lost unless texture resolution is rather large. Since most tree-modeling software produces models that have each leaf separately represented by one or more polygons, leaf polygons are typically extremely small. Consequently, we have to super-sample by rendering geometry to a buffer of such size that each polygon is guaranteed to be rasterized, i.e. covers at least one pixel. The required size of said off-screen buffer can be calculated using the texel size in world space.

However, the textures may not be arbitrarily large because we have to respect the user-defined maximum texture size, so the images have to be downsampled to their correct size.

Rendering a billboard cloud without proper depth-sorting is only possible if alpha values of texels are restricted to being either zero (fully transparent) or one (fully opaque) and the graphics API's alpha test is set to pass opaque texels only. Therefore, the downsampling process must not produce any semi-transparent pixels. We use a rectangular median filter in conjunction with a thresholding function. A threshold of around 0.3 results in sufficiently dense looking foliage that yet does not appear too impenetrable.

3.3.8 Incremental levels of detail. Different levels of detail (LoD) can be easily produced by varying the error threshold parameter and texture resolution. The main disadvantage of this approach is the completely independent selection of simplifying planes for different LoDs. For the most detailed LoDs we employ an incremental approach, using one LoD as input for the next lower LoD. While this produces bad results at extreme simplifications (<10 billboards), for the detailed LoDs, where switches between levels are most visible, it reduces the switching artifacts (Fig. 10).

3.3.9 Dynamic Lighting. We use instancing of trees in our scenes, reusing the same tree model in different positions, with arbitrary rotations. This works fine with pre-lighted trees, as long as no strongly directional lighting is used. Otherwise, the illuminated part of an instanced tree faces in arbitrary directions. This can be resolved by using dynamic lighting.

As already stated in Décoret original paper, projecting normals as well as colors onto the billboards results in

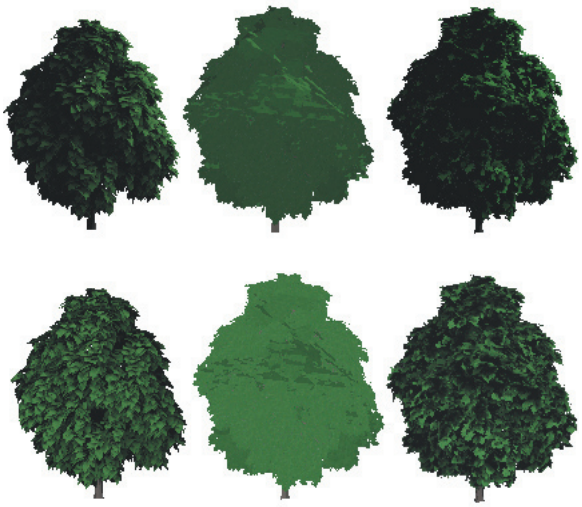


Figure 13 : *Left column shows original geometry, middle column flat shaded Billboard Clouds, right column dynamically per-pixel lighted Billboard Clouds. Top and bottom row show different light directions.*

normal maps which can be lighted dynamically. This can be extended to more sophisticated shading models.

The main disadvantage of lighted billboards results from leaves projected on billboards not coplanar to them. Such billboarded leaves are from every viewpoint smaller than the original geometry, thereby contributing less to the overall intensity of the tree. This holds true for simple normal maps as well as more sophisticated shading algorithms.

Our lighting model for the original tree uses two parameters: face normal of geometry and an ambient occlusion factor [LAN02]. It produces results similar to the one presented by Qin [QIN01], but at real-time rates.

When lighting the original geometry, we calculate for every leaf a factor representing the percentage of the sky visible from this leaf, and an averaged horizon representation for this leaf.

When rendering the textures for the billboards, we do not render the original tree with the applied lighting model, but produce a normal map and an ambient occlusion map and a compressed horizon representation. These maps are evaluated by a fragment shader when rendering the billboards. Fig. 13 shows the comparison between lighted geometry (using the lighting described above on triangles), flat lighted billboards, and dynamically lighted billboards. The shader itself is very simple and needs only one additional texture fetch for the lighting, delivering high performance when rendering lit trees.

This is just an example for dynamic lighting, other lighting models may transfer as easily from geometry to billboard representation.

4 Results

Fig. 11 shows simplification results for various tree models. We render the billboards as static geometry, with distance-based levels of detail switches. In our walkthrough application we employ only view frustum culling and additionally vertex-buffer objects for caching.

Our implementation of the algorithm packs all textures of a billboard cloud to one large texture to minimize texture switching overhead. A further reduction of texture switching could be accomplished by packing textures of all tree models in use to one large texture.

The amount of trees displayable at interactive frame rates is satisfying, while the visual quality of the models is very good at medium and far viewing distances.

Dynamic self-shadowing [WOO90] is not recommended either, since casting a shadow onto the billboard planes, no matter which shadowing technique is used, would reveal their flatness. Dynamic shadow-casting on the ground can be easily performed with the use of projective texture mapping.

In our application we chose a precomputed, static lighting model for the trees and precomputed shadows on the ground texture (Fig. 12).

5 Conclusion

We have shown an adaptation of the billboard cloud generation algorithm that specifically targets the simplification of tree models, but also improves the output for more general models. With this system, highly complex foliage models can be greatly simplified automatically with different levels of detail, a process which was previously performed manually in most cases.

Billboard clouds retain the overall look of the original model from any viewing angle and consist of static geometry, so that they can be easily cached for highly efficient rendering. Trees can be rendered trivially, without requiring any adaptations to existing rendering engines. For many applications, these advantages outweigh the fact that self-shadowing and animation of billboard clouds is not readily possible.

Billboard clouds are an excellent tool for medium and far distance representations of trees. Only for close-up viewing of models, other forms of representation would be more appropriate. For instance, a simple approach such as using a low-polygon stem and billboard cloud foliage would increase the feeling of solidity and could be sufficient for certain applications.

6 Future work

While the billboard cloud representation in general fares well, the generation algorithm proposed by Décoret is rather time consuming. In our future work, we intend to examine simpler clustering algorithms to reduce the computational burden of billboard cloud generation.

We further plan to approximate dynamic self-shadowing using pre-calculated visibility information, and compensations for the billboard lighting problems mentioned in chapter 3.9.9.

We will incorporate and further investigate billboard clouds in the GeomeTree forest rendering project currently in development at the VRVis Research Center, Vienna.

Acknowledgements

Special thanks go to Xavier Décoret for in-depth discussions of algorithmic details and problem solutions.

This work was funded by the Austrian KPlus program and the Austrian FWF grant # P17260-N04.

Tree models obtained from Xfrog Public Plants, 3DCafe.com and VRVis Tree Designer.

References

[AND04] C.Andujar, P.Brunet, A.Chica, J.Rossignac, I.Navazo, A.Vinacua, 2004. "Computing Maximal Tiles and Applications to Impostor-Based Simplification". Computer Graphics Forum Vol. 23 Issue 3 Page 401 - September 2004

[BIO] Bionatics tree modeling Software. www.bionatics.com

[CHE93] S.E.Chen, L.Williams. „View Interpolation for Image Synthesis". Computer Graphics Vol 27, 1993, pp. 279-288

[DAC03] C.Dachsbacher, C.Vogelsang, M.Stamminger. „Sequential Point Trees". ACM Transactions on Graphics, Vol 22, Issue 3 (July 2003), pp. 657-662

[DCD04] P.Decaudin, F.Neyret. „Rendering Forest Scenes in Real-Time". Eurographics Symposium on Rendering June 2004.

[DEC02] X.Décoret, 2002. „Pré-Traitement de grosses bases de données pour la visualisation interactive", PhD thesis, Université Joseph Fourrier, Oct. 2002.

[DEC03] X.Décoret, F.Durand, F.X.Sillion, J.Dorsey. "Billboard Clouds for Extreme Model Simplification", ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003), Vol. 22 , Issue 3 (July 2003), pp 689 - 696.

[DEU99] O.Deussen, B.Lintermann. "Interactive modelling of plants". IEEE Computer Graphics and Applications Vol.19 n.1, January 1999, pp.56-65.

[DUD72] R.O.Duda, P.Hart, 1972. „Use of the Hough-Transform to detect Lines and Curves in Pictures", Communications of the ACM archive, Vol. 15 , Issue 1 (January 1972), pp. 11 - 15

[HOP93] H.Hoppe, T.DeRose, T.Duchamp, J. McDonald, W.Stuetzle, „Mesh Optimization". Proceedings of the 20th annual conference on Computer graphics and interactive techniques, Vol 27, 1993, pp. 19-26

[IDV] Interactive Data Visualization Inc. www.idvinc.com

[JAR72] R.A. Jarvis, 1972. „On the Identification of the Convex Hull of a Finite Set of Points in the Plane". Information Processing Letters 2, pp. 18-22

[LAN02] Hayden Landis, Production-Ready Global Illumination, "RenderMan in Production" SIGGRAPH Course Notes, 2002

[LIN90] A.Lindenmayer, P.Prusinkiewicz, 1990. „The algorithmic beauty of plants"; Springer Verlag, New-York 1996, ISBN 0-387-94676-4

[MAN03] S.Mantler, A.Fuhrmann, 2003. "Fast approximate visible set determination for point sample clouds". Proceedings of the workshop on Virtual environments 2003, pp. 163-167

[MAX95] N.Max, K.Ohsaki, 1995. "Rendering Trees from Precomputed Z-Buffer Views". Proceedings of the 6th Eurographics Workshop on Rendering

[MEY01] A.Meyer, F.Neyret, P.Poulin, 2001. "Interactive Rendering of Trees with Shading and Shadows". Eurographics Workshop on Rendering 2001, pp. 183-196

[QIN01] X.Qin, E.Nakamae, K.Tadamura, Y.Nagai, "Fast photo-realistic rendering of trees in daylight", Proc. of Eurographics 2003, pp. 243-252.

[REM02] I.Remolar, M.Chover, Ó.Belmonte, J.Ribelles, C.Rebollo, 2002. "Real-Time Tree Rendering". Departamento de Lenguajes y Sistemas Informáticos, Universitat Jaume I, Technical Report DLSI 01/03/2002, Castellón (Spain), March, 2002.

[SCH95] G.Schaufler, 1995. „Dynamically generated Impostors". GI Workshop „Modeling - Virtual Worlds - Distributed Graphics", pp. 129-135

[SCH97] G.Schaufler, 1997. „Nailboards: A Rendering Primitive for Image Caching in Dynamic Scenes". Eurographics Rendering Workshop 1997, pp. 151-162

[TOB02] R. F. Tobler, S. Maierhofer, and A. Wilkie, 2003. „Mesh-Based Parametrized L-Systems and Generalized Subdivision for Generating Complex Geometry". International Journal on Shape Modelling, Volume 8, Number 2, World Scientific, pp. 173-191, December 2002

[TUR00] P.Lindstrom, G.Turk, 2000. „Image-driven Simplification". ACM Transactions on Graphics Vol. 19, Issue 3 (July 2000), pp. 204-241.

[UML04] E.Umlauf, 2004. „Image-based Rendering of Forests". Diploma Thesis and Tech Report: VRVis Center, 2004 TR-VRVis-2004-038

[WEB95] J. Weber, J. Penn, 1995. „Creation and rendering of realistic Trees". ACM Transactions on Graphics (SIGGRAPH '95), pp. 119-128

[WOO90] A. Woo, P. Poulin, A. Fournier, 1990. „A Survey of Shadow Algorithms". IEEE Computer Graphics and Applications Vol.10, pp. 13-32

[XFR] Greenworks Software. www.greenworks.de

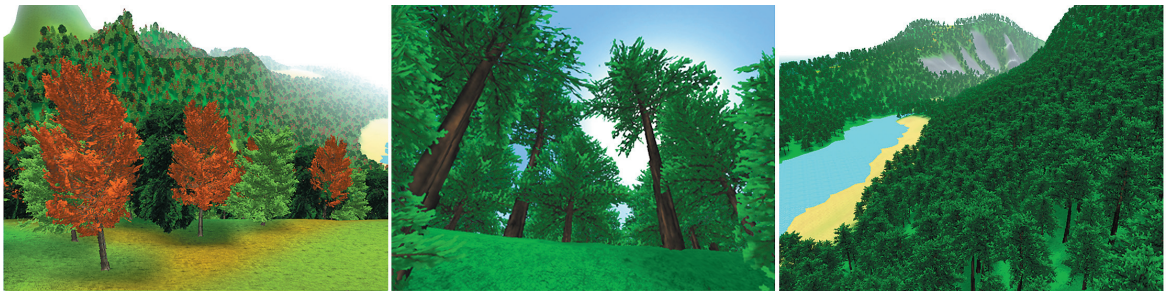


Plate 1: *Sample screen shots of our real-time rendered forests*

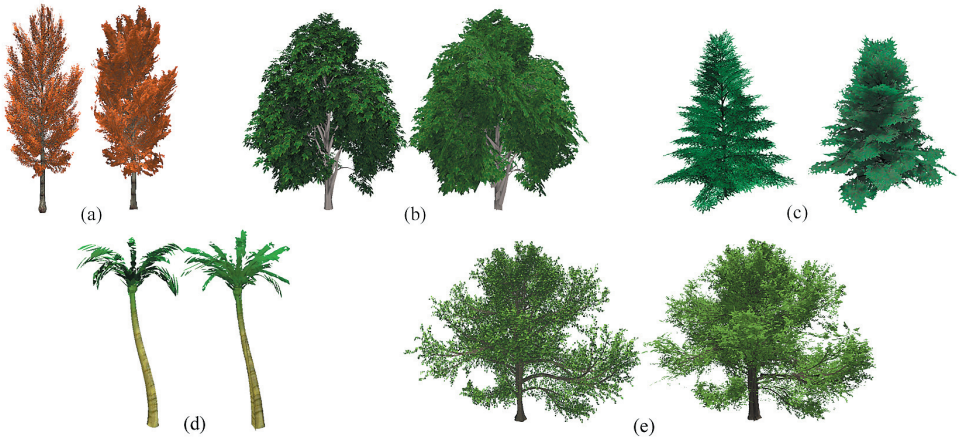


Plate 2: *Polygonal Trees (left) and their Billboard Clouds (right).*