Billboard Clouds for Extreme Model Simplification

Xavier Décoret, Frédo Durand, François X. Sillion, Julie Dorsey

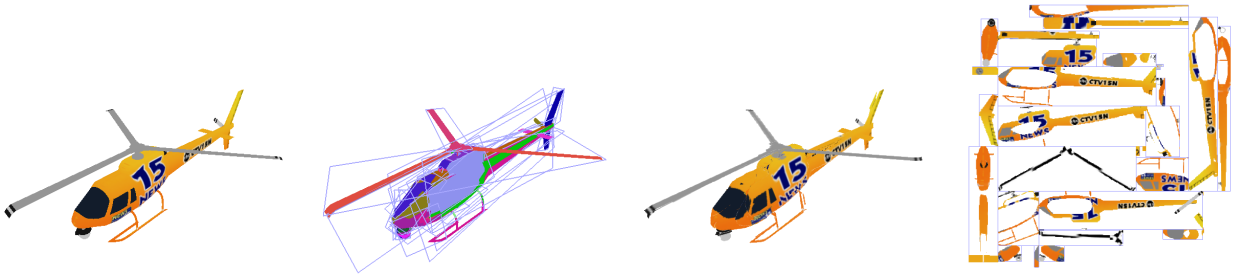
Billboard Clouds for Extreme Model Simplification

Xavier Décoret
MIT LCS

Frédo Durand
MIT LCS

François X. Sillion
Artis INRIA*

Julie Dorsey
Yale University



(a) (b) (c) (d)

Figure 1: Example of a billboard cloud: (a) Original model (5,138 polygons) (b) false-color rendering using one color per billboard to show the faces that were grouped on each (c) View of the (automatically generated) 32 textured billboards (d) the billboards side by side.

Abstract

We introduce *billboard clouds* – a new approach for extreme simplification in the context of real-time rendering. 3D models are simplified onto a set of planes with texture and transparency maps. We present an optimization approach to build a billboard cloud given a geometric error threshold. After computing an appropriate density function in plane space, a greedy approach is used to select suitable representative planes. A good surface approximation is ensured by favoring planes that are "nearly tangent" to the model. This method does not require connectivity information, but instead avoids cracks by projecting primitives onto multiple planes when needed. For extreme simplification, our approach combines the strengths of mesh decimation and image-based impostors. We demonstrate our technique on a large class of models, including smooth manifolds and composite objects.

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

Keywords: Billboard, model simplification, LOD, image-based rendering, error-driven simplification, real-time rendering

1 Introduction

An important challenge in computer graphics is to adapt the representation of a model to the available resources, by constructing a *simplified model* that guarantees an "optimal" combination of image quality and rendering speed. Note that model simplification is not required solely for performance reasons. As models become more precise geometrically, they typically incorporate smaller details, and the limited resolution of the display produces aliasing. The model should therefore be *pre-filtered* to minimize artifacts such as flickering and disappearing objects.

Mesh decimation has dramatically progressed, and techniques such as edge collapse permit efficient and accurate simplification, e.g., [Heckbert and Garland 1997; Puppo and Scopigno 1997; Luebke 2001]. However, these methods work best on finely tessellated smooth manifolds, and often require mesh connectivity. Moreover, the quality of the simplified model becomes unacceptable for extreme simplification (less than a couple hundred polygons): The nature of the simplification primitives – opaque triangles – makes it difficult to treat complex, disconnected objects such as trees. Another great difficulty with extreme simplification is maintaining visual detail. Large models combining many different textures are especially difficult to simplify, since some vertices have to be retained in the model because they play a special role as texture coordinate holders. Therefore in situations such as game authoring, extremely simplified models have to be optimized manually.

On the other hand, image-based acceleration is very efficient for distant scenery, as it can naturally fuse multiple objects, but offers only limited parallax effects. As a special case, *billboards* are a popular representation to represent complex geometry such as trees. They consist either in a single texture-mapped rectangle kept parallel to the image plane, or in two rectangles arranged as a cross. They work well in the distance, but the planes become unacceptably conspicuous when the viewpoint gets closer, resulting in the "cardboard-set look," about which gamers complain.

This paper introduces a new approach for *extreme simplification*, i.e. the simplification of complex static 3D models to a few primitives. We propose a new representation for the model, composed of a set of planar polygons with texture and transparency maps. We call this representation a *billboard cloud* to emphasize the complex and uncorrelated geometry of the billboards. Previous techniques typically perform well only on either smooth connected manifolds or on distant geometry with few parallax effects. Billboard clouds effectively bridge this gap and permit extreme simplification of arbitrary models, with good visual fidelity including appropriate par-

*ARTIS is team of the GRAVIR/IMAG laboratory, a joint effort of CNRS, INRIA, INPG and UJF

allax and silhouettes. We stress that our method is not competing with mesh decimation for the creation of finely tessellated simplified models. Rather it is complementary and aimed at *extreme* simplification, where the small number of primitives in the simplified model makes mesh simplification less attractive.

1.1 Previous work

Our technique is related to a large body of work, since it offers a continuum of simplification – from a single billboard to a geometric representation similar to face clustering.

Mesh decimation, e.g. [Heckbert and Garland 1997; Puppo and Scopigno 1997; Luebke 2001] has mostly focused on the simplification of single manifolds, where the simplified model has the same topology as the input. Silhouette clipping [Sander et al. 2000] dramatically improves the rendering quality, but suffers from the same limitation to manifolds. There are notable exceptions that merge different manifolds, such as vertex clustering [Rossignac and Borrel 1993; Luebke and Erikson 1997; Low and Tan 1997], the use of lower-dimensional manifolds [Popovic and Hoppe 1997; Low and Tan 1997], or the use of virtual edges [Garland and Heckbert 1997; Erikson and Manocha 1999]. These techniques work extremely well when the size of the simplified representation is moderate (several hundreds or thousands of polygons). However, for extreme simplification, they suffer from the inherent limitation of opaque polygons or simplices for representing complex geometry. Moreover, the handling of multiple textures is not trivial, and basically requires the computation of new textures [Cohen et al. 1998; Cignoni et al. 1998; Sander et al. 2000]. While our approach does also require new textures, we make this texture computation an inherent part of the representation, which essentially decouples texture and geometric complexity, while allowing complex transparency effects.

Our technique is closely related to superfaces [Kalvin and Taylor 1996] and face clustering [Garland et al. 2001; Sheffer 2001], which group connected sets of nearly coplanar faces. Computer vision has also studied techniques to cluster a range image into planar regions, e.g. [Faugeras et al. 1983]. These approaches require appropriate triangulation to be rendered by current graphics hardware, increasing the number of primitives. A key difference is that our technique does not require connectivity of the clustered faces. This allows us to treat a larger class of inputs, such as vegetation and small parts of objects, and to perform more aggressive simplification. Moreover, our use of transparency provides a much more flexible and faithful planar primitive.

Maciel and Shirley proposed to replace distant geometry with view-dependent or view-independent *impostors* [1995], which was later extended to a hierarchy of cached images facing the viewer [Schaufler and Sturzlinger 1996; Shade et al. 1996]. Meshed impostors, e.g. [Sillion et al. 1997; Darsa et al. 1997], layered depth images [Shade et al. 1998; Popescu et al. 1998] and layered impostors [Schaufler 1998; Décoret et al. 1999] have been developed to better capture parallax effects, at the cost of increased rendering complexity. Other techniques permit better transitions between impostors and geometry, e.g. [Aliaga 1996; Aliaga and Lastra 1998]. The The Delta tree proposes an encoding of reference views from which new views can be efficiently generated [Dally et al. 1996]. Talisman graphics hardware system also uses an image-based strategy to reduce the load of the 3D engine [Torborg and Kajiya 1996]. The approach by Aliaga et al. [1999] and Wilson et al. [2002] is particulary related to our technique. They use an optimization strategy to choose the placement of impostors. Their methods however are based on sample views, while ours relies on a direct analysis of the model.

Oliveira et al. proposes an efficient pre-warping scheme for rendering complex geometry using textured quads [Oliveira et al. 2000]. Parallel textured planes similar to layered impostors have also been used to render volumetric objects such as trees or hair [Meyer and Neyret 1998; Lengyel 2000]. Max et al. [1999] have developed specialized techniques to exploit the natural hierarchy of trees. Point-based rendering, e.g., [Levoy and Whitted 1985; Grossman and Dally 1998; Rusinkiewicz and Levoy 2000], and surfels [Pfister et al. 2000] are other alternative representations for fast display that work well with complex objects such as vegetation. Our technique combines both the flexibility of the latter unstructured representation and the efficiency of the more structured image-based techniques.

1.2 Billboard clouds

We introduce the *billboard cloud* as an alternative representation of a complex static 3D model. This representation consists of a set of textured, partially transparent polygons (or billboard), with independent size, orientation and texture resolution (see Figure 1). A billboard cloud is built by choosing a set of planes that capture well the geometry of the input model, and by projecting the triangles onto these planes to compute the texture and transparency maps associated with each plane of the billboard. Textures can also include more than simple color information (e.g., normal maps for pixel shaders). Billboard clouds are rendered by rendering each polygon independently.

A major advantage of billboard clouds is that no topological information (such as polygon connectivity) is required. Moreover the visual complexity of the billboard cloud results mainly from the texture, and no complex boundary description is necessary. Many previous techniques for accelerated display of complex models have employed the notion of multiple, partially transparent polygons, and can indeed be seen as special cases of billboard clouds.

We express the generation of a billboard cloud from a 3D model as an optimization problem defined by an error function, and a cost function. We use an error metric based on the Euclidean L_∞ norm; it is computed in object-space and is view-independent. The cost function involves the number of planes and the compactness of textures.

We then use an error-based construction strategy. A maximum tolerable error ε is set, and the aim is to build the billboard cloud respecting this threshold with the minimum cost. Our algorithm is a greedy optimization in that we iteratively select planes that can "collapse" the maximum number of faces. This selection is based on a *density* computed in a discretized version of plane space: The density tells us the amount of faces that can be simplified by a plane. Faugeras et. alhave studied this problem of fitting planes to a set of primitives [Faugeras et al. 1983]. Their approach is to successively merge adjacent primitives while enforcing the error bound. This approach is also used by Garland et. alto cluster faces [Garland et al. 2001]. Our solution does not rely on merging but on a representation of plane space where dominant planes of the model can be directly identified.

In Section 2, we define density and explain how to rasterize this quantity onto a grid in plane space. Our greedy plane selection is then described in Section 3.

2 Density in plane space

In order to estimate the relevance of planes, we compute a *density* in plane space. Density is defined using three important notions: *validity*, *coverage* and *penalty*. Validity is binary and assesses whether a plane is a valid simplification of a face. Coverage includes an additional notion of representation quality. Finally, penalty prevents undesirable planes. We define the density $d(\mathcal{P})$ of a plane $\mathcal{P}$ as :

$$d(\mathcal{P}) \quad = \quad C(\mathcal{P}) \quad - \quad \text{Penalty}(\mathcal{P}) \qquad (1)$$

where $C(\mathcal{P})$ is the *coverage*, that is the "amount" of faces f for which $\mathcal{P}$ is a *valid* representation. We will see in Section 2.2 how this is estimated. We clamp $d(\mathcal{P})$ to zero to ensure positive values.

We consider oriented planes, that is, the orientation of the normal matters.

2.1 Validity

We use the following error criterion: the Euclidean distance between a point and its simplification should be less than ε. This means that a point is allowed to move only in a sphere of radius ε.

We say that a plane $\mathcal{P}$ yields a *valid* representation of a point v if there exists a point p on $\mathcal{P}$ such that $||vp|| < \varepsilon$. We define the *validity domain* of v as the set of planes that can represent it, which we denote by $valid_\varepsilon(v)$. It is the set of planes that intersect the sphere of radius ε centered on v.

A plane is valid for a polygon if it is valid for all its vertices. The validity domain $valid_\varepsilon(f)$ of a face f is the intersection of the validity domain of its vertices. Geometrically, this corresponds to the set of planes intersecting a set of spheres. We will interchangeably say that a face is valid for a plane, and that the plane is valid for the face. Each plane $\mathcal{P}$ is associated with a set of valid primitives called its *validity set*, which we note $valid_\varepsilon(\mathcal{P})$. Without loss of generality we will consider all polygons to be triangles, although a major benefit of our approach is that models need not be triangulated.

The notion of validity defines our simplification problem as a geometric cover: We want to find a minimal set of planes $\{\mathcal{P}_i\}$ such that for each primitive f of the scene, there exists at least one i such that $f \in valid_\varepsilon(\mathcal{P}_i)$. Geometric cover problems are known to be NP-hard, and the discrete version, the set cover problem, is NP-complete [Hochbaum 1997].

2.2 Coverage

The size of the validity set of a plane is a good initial estimate of its relevance. Using only this estimate, however, would label a plane covering several very small faces as more relevant than one covering a single large one, which we obviously do not want. Therefore we weight each valid face by its projected area $area_\mathcal{P}(f)$, defined as the area of the orthographic projection of the face in the direction orthogonal to plane $\mathcal{P}$. This puts more weight on large faces, and favors planes parallel to the faces. The coverage used in equation (1) is therefore computed with :

$$C(\mathcal{P}) = \sum_{f \in valid_\varepsilon(\mathcal{P})} area_\mathcal{P}(f) \qquad (2)$$

2.3 Penalty and tangency

We introduce a penalty that prevents planes that miss primitives in their neighborhood. This addresses the classical pitfall with greedy algorithms: A choice for one iteration might be locally optimal, but can result in poor overall performance because the remaining problem is less well conditioned. In our case, the missed primitives would then be hard to simplify. In contrast, our penalty favors planes that are quasi-tangent to the model, and that do not miss primitives in one direction.

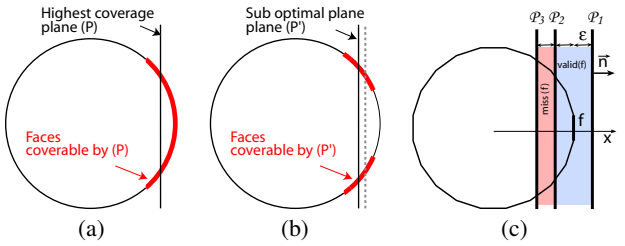


Figure 2: Near-tangency and penalty due to missed primitives.

Consider the example of Fig. 2 where we want to simplify a tessellated sphere. The plane with optimal coverage is the one that covers exactly a spherical lens (Fig. 2 (a)). However, due to the discretization we use to evaluate coverage we might select a slightly translated plane ending up with a billboard as indicated on Fig. 2 (b). To prevent this, we let each face of the model penalize the planes that would "miss" them. Consider for examples face f on Fig. 2 (c). For a given normal direction $\vec{n}$, the valid planes for f are the ones between $\mathcal{P}_1$ and $\mathcal{P}_2$. On the left of $\mathcal{P}_2$, planes will "miss" f. We introduce the missing set $miss_\varepsilon(f)$ as the set of planes that are "almost" valid for f. More precisely, a plane $\mathcal{P}$ with unit normal $\vec{n}$ misses f if $\mathcal{P} \notin valid_\varepsilon(f)$ and if there exists a plane $\mathcal{P}' \in valid_\varepsilon(f)$ such that $\mathcal{P}'$ is the image of $\mathcal{P}$ by a translation of $a\vec{n}$ with $0 \leq a \leq \varepsilon$ (Fig. 2). Thus, in a given direction, $valid_\varepsilon(f)$ and $miss_\varepsilon(f)$ are two contiguous segments of length 2ε and ε. Note that $miss_\varepsilon(f)$ is on only one side of $valid_\varepsilon(f)$. This direction dependency is quite important for the robustness of the method. The quasi-tangents on the left are captured by the opposite direction $-\vec{n}$.

The constraint $0 \leq a \leq \varepsilon$ defines the *width* for $miss(f)$. Choosing an infinite width would have restricted us to planes that are quasi-tangent to the convex hull of the input, while our width of ε allows us to capture features and holes larger than ε inside the convex hull.

The penalty formula is then similar to the coverage, but it is weighted more heavily.

$$\text{Penalty}(\mathcal{P}) = w_{penalty} \sum_{f \in miss(\mathcal{P})} area_\mathcal{P}(f) \qquad (3)$$

In practice, we use $w_{penalty} = 10$, but we note that the behavior of the method is robust to the setting of $w_{penalty}$: we simply need a weight that is large enough to strongly prevent the choice of planes that miss primitives.

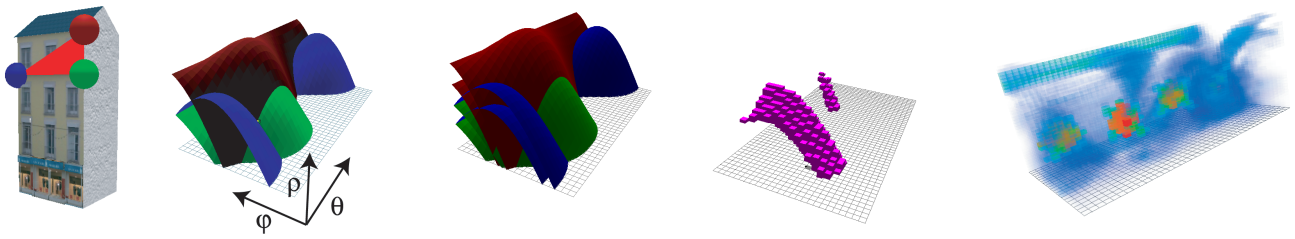
2.4 Discretization of plane space

We estimate density in a discretized plane space. We parameterize planes using the spherical coordinates (θ, φ) for the normal direction, and the distance ρ to the origin. This is the 3D equivalent of the Hough transform [Hough 1962]. This parameterization is not uniform and has singularities at the poles, but this is not a problem for our approach: Some portions of plane space will simply be oversampled.

In this parameter space, the set of planes going through a point of the primal space defines a sheet $\rho = f(\theta, \varphi)$, as shown in Fig. 3(b). The validity domain of a vertex is the envelope defined by the sheet translated by $-\varepsilon$ and $+\varepsilon$ in the ρ dimension (Fig. 3(c)). The validity domain of a triangle f is the intersection of the envelopes of its three vertices.

We use a uniform discretization of this parameter space into bins $\mathcal{B}_{\theta_i \varphi_j \rho_k}$, and compute a density value for each bin. A naïve approach would use the density in the center of the bin (sampling). This would result in artifacts because some contributions would be overlooked. We therefore use a more conservative rasterization. For a given face f, we consider all bins that intersect $valid_\varepsilon(f)$. Such bins are said to be *simply* valid in the sense that there exists one plane in the bin that is valid for f. This can also be seen as using a box filter before sampling. We use this plane-space density only as a guide for the selection of planes, and if a bin is simply valid for a large number of faces, there is usually a plane in the bin or in its neighborhood that is valid for a large subset of them.

The density of a bin $d(\mathcal{B})$ is obtained by summing the coverage and penalty values contributed by the set of faces $valid_\varepsilon(\mathcal{B})$ for which it is simply valid. This can be computed efficiently, with iterations only on the θ, φ coordinates, avoiding a full θ, φ, ρ loop. For each face f, and each direction (θ_i, φ_j), we conservatively compute the range $[\rho_{min}, \rho_{max}]$ that intersect $valid_\varepsilon(f)$ (see appendix). We compute $C(f)$, which is constant for the given direction, and add it *as a coverage value* to the bins between ρ_{min} and ρ_{max} (in blue on figure 4) and *as a penalty* to the bins between $\rho_{min} - \varepsilon$ and ρ_{min} (in red). To account for aliasing, $C(f)$ is weighted by the percentage of the bin that is covered (bins i, j and k on figure 4).



(a) (b) (c) (d) (e)

Figure 3: Density in plane space. (a) Scene with face f and its three vertices highlighted. (b) Set of planes going through each vertex. represented in plane space. The plane of f corresponds to the intersection of the three sheets. (c) Validity domain of each vertex. (d) Discretized validity domain of f. (e) Coverage for the whole house. We clearly see 6 maxima (labelled) corresponding to the 4 side faces and the 2 sides of the roof. Note in addition the degenerate maximum that spans a whole row for $\phi = \pi/2$. All values of θ match the same plane: the plane of the ground.

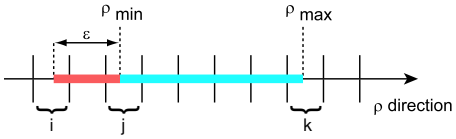


Figure 4: Rasterization in plane space.

Greedy (input model, threshold ε)
  set of faces $\mathcal{F}$=input model
  billboard cloud $\mathcal{BC} = \emptyset$
  while $\mathcal{F} \neq \emptyset$
    Pick bin $\mathcal{B}$ with highest density
    Compute $valid_\varepsilon^{\mathcal{F}}(\mathcal{B})$
    $\mathcal{P}_i$ = **RefineBin** ($\mathcal{B}$, $valid_\varepsilon^{\mathcal{F}}(\mathcal{B})$)
    UpdateDensity($valid_\varepsilon^{\mathcal{F}}(\mathcal{P}_i)$)
    $\mathcal{F} = \mathcal{F} \setminus valid_\varepsilon^{\mathcal{F}}(\mathcal{P}_i)$
    $\mathcal{BC} = \mathcal{BC} \cup \mathcal{P}_i$
  Compute textures($\mathcal{BC}$)

Figure 5: Pseudocode of the greedy selection of planes.

3 Greedy optimization

Now that we have defined and computed a density over plane space, we present our greedy optimization approach to select a set of planes that approximate the input model. We iteratively pick the bin with the highest density. We then search for a plane in the bin that can collapse the corresponding set of valid faces. This may require an adaptive refinement of the bin in plane space as explained below. Once a plane is found, we update the density to remove the contribution and penalty due to the collapsed faces, and proceed to the next highest-density bin. Once all the faces have been collapsed, we compute the corresponding textures on each plane.

3.1 Adaptive refinement in plane space

Our grid only stores the simple density $d(\mathcal{B})$ of each bin, and for memory usage reasons, we do not store the corresponding set of faces $valid_\varepsilon(\mathcal{B})$. We iterate on the faces that have not yet been collapsed to compute those that are valid for $\mathcal{B}$. Further computations for the plane refinement are performed using only this subset of the model. We note quantities such as density or validity set restricted to such a subset of faces $\mathcal{F}$ with the superscript $^{\mathcal{F}}$.

Recall that the density stored in our plane-space grid uses the simple validity, and that the faces that are simply valid for a bin are not necessarily valid for all its planes. We therefore need to refine our selection of a bin to find the densest plane. We subdivide bins

RefineBin (bin $\mathcal{B}$, set of faces $\mathcal{F}$)
  plane $\mathcal{P}$ = center of $\mathcal{B}$
  if ($valid_\varepsilon^{\mathcal{F}}(\mathcal{P}) == valid_\varepsilon^{\mathcal{F}}(\mathcal{B})$)
    return $\mathcal{P}$
  bin $\mathcal{B}_{max}$=NULL
  for each of the 27 neighbors $\mathcal{B}_i$ of $\mathcal{B}$
    Subdivide $\mathcal{B}_i$ into 8 sub-bins $\mathcal{B}_{ij}$
    for each $\mathcal{B}_{ij}$ *// there is a total of 8*27 such* $\mathcal{B}_{ij}$
      Compute $d^{\mathcal{F}}(\mathcal{B}_{ij})$
      if ($d^{\mathcal{F}}(\mathcal{B}_{ij}) > d^{\mathcal{F}}(\mathcal{B}_{max})$)
        $\mathcal{B}_{max} = \mathcal{B}_{ij}$
  return RefineBin ($\mathcal{B}_{max}$, $valid_\varepsilon^{\mathcal{F}}(\mathcal{B}_{max})$)

Figure 6: Pseudo-code of recursive adaptive refinement in plane space.

adaptively until the plane at the center of a sub-bin is valid for the entire validity set of the sub-bin (Fig. 6).

We allow the refinement process to explore the neighbors of the bin as well. Indeed, because we use simple validity, the densest plane can be slightly outside the bin we initially picked, as illustrated by figure 7. We use a simple strategy: We subdivide the bin and its 26 neighbors, and pick among these 27×8 sub-bins, the one with highest simple density.

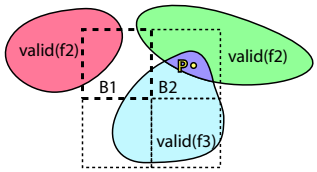


Figure 7: Simple density in plane space. Although bin $\mathcal{B}_1$ has maximum simple density, the densest plane is $\mathcal{P}$, which is in bin $\mathcal{B}_2$.

Bins are then updated to remove the contributions and penalties of the collapsed faces. We iterate over the faces collapsed on the new plane and use the same procedure described in Section 2.4, except that contributions and penalties are *removed*. The algorithm then proceeds until all faces of the model are collapsed on planes.

3.2 Computing textures

Each plane $\mathcal{P}_i$ is assigned a set of collapsed faces $valid_\varepsilon^{\mathcal{F}}(\mathcal{P}_i)$ during the greedy phase. We first find the minimum bounding rectangle of the projection of these faces in the texture plane (using the CGAL library[CGA n. d.]), then we shoot a texture by rendering the faces

using an orthographic projection. Texture resolution is automatically selected according to the object-space size of the bounding rectangle.

For proper alpha-compositing and mip-mapping, we use a black background and precomposited alpha [Porter and Duff 1984]. We also compute a normal map when relighting is desired.

We observe that as with any geometric cover technique, faces can belong to the valid set of several planes. A natural solution to minimize "cracks" between adjacent billboards is therefore to render such faces onto the textures of *all* planes for which they are valid. This operation is easily performed by rendering the entire scene in the texture, using extra clipping planes to restrict imaging to the valid zone around the plane. This treatment is legitimate since by definition the reprojection error in this valid zone is sufficiently small.

3.3 Optimizing texture usage

In this section, we show how we can also optimize the compactness of textures. We note that the contribution of a plane as defined by Eq. 2 can be due to primitives that are widely separated. This could lead to billboards that contain mostly empty spaces. In order to alleviate this, we include a step that restricts the set of triangles that are projected onto a plane to a *compact* set.

We modify the greedy algorithm and restrict $valid_\varepsilon^{\mathcal{F}}(\mathcal{B})$ to a maximal compact subset. We note that compactness issues are usually due to different components of the scene being collapsed onto the same plane. We can therefore analyze the validity set $valid_\varepsilon^{\mathcal{F}}(\mathcal{B})$ into *clusters* and use only the cluster with the highest density.

The bucket-like partitioning algorithm by Cazals et al. [Cazals et al. 1995] is ideally suited to this task. We project all the facets in $valid_\varepsilon^{\mathcal{F}}(\mathcal{B})$ onto the plane in the center of bin $\mathcal{B}$. We iteratively attempt to separate this set along a direction $\vec{d}$. If the projection of the facets along $\vec{d}$ contains an empty interval, we split the set $valid_\varepsilon^{\mathcal{F}}(\mathcal{B})$ into two components and only keep the largest one. The same procedure is recursively applied until the set cannot be separated. In practice, we use a discretization of the 2D direction space into four directions separated by 45°.

The rest of the greedy algorithm remains unchanged. It is important to note that primitives in $valid_\varepsilon^{\mathcal{F}}(\mathcal{B})$ that were not selected in the largest compact set will be handled by subsequent iteration of the greedy selection. A plane might therefore be selected multiple times to handle different clusters.

4 Implementation and results

We demonstrate the results of our greedy optimization technique. We have focused on the visual quality of the results, without necessarily optimizing for resource usage. The technique always converges and yields a billboard cloud where all original faces have been collapsed on (at least) a billboard, while enforcing the error bound. It is also robust with respect to internal parameter choices, such as the resolution of the discrete plane space.

The only input parameters to our algorithm are (a) an (object-space or image-space) error threshold ε and (b) the desired texel size τ in object space. In practice we advise to use $\tau = 5\varepsilon$ because the texture alpha map encodes the complexity of the silhouette, which leads to visually richer simplification.

In all the examples shown below, we characterize the cost of the billboard cloud representation by (a) the number of billboards used and (b) the number of texels used for the entire collection of textures. We note that traditional mesh simplification also requires the storage of additional texture if the appearance of the object is to be preserved, e.g. [Cohen et al. 1998; Cignoni et al. 1998; Sander et al. 2000]. In particular, if the original object contains multiple textures, they need to be resampled in order to prevent object vertices, in their role as texture-coordinate holders, to impede simplification. Billboard clouds require more texels than traditional appearance-preserving simplification because of empty texels. Recall, however, that this use of transparency yields a higher-quality preservation of the visual richness of silhouettes. Also note that texture compression can typically reduce memory requirements by a factor of 4 to 8. The technique by Kraus and Ertl [2002] for adaptive textures could also be used to dramatically reduce the cost of empty portions.

We used a simple greedy packing algorithm [E. G. Coffman et al. 1997] to store all the textures of an entire billboard-cloud into a single texture, which optimizes memory usage and minimizes texture switch.

Figure 8 shows images of a number of models and their corresponding billboard clouds. Note that we are in the range of extreme simplification, since we typically obtain fewer than 200 billboards. Still, both the silhouette and important 3D structure of the objects are well captured (also see accompanying video).

We studied the number of billboards needed as a function of the maximum error (Fig. 9). From the curve in log-log units, we observe that the number of billboards appears to be roughly in $1/\varepsilon$. The regularity of these curves is important because it suggests that our error-based algorithm can be used in a budget-based fashion by inverting the curve. The non-monotonicity at the end of the curve is due to the greedy nature of our optimization method.

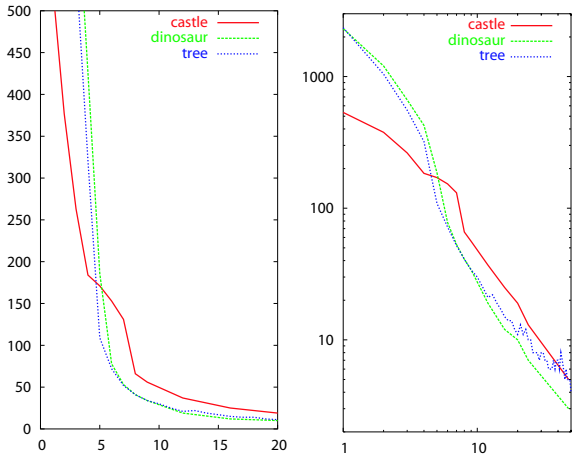


Figure 9: Cost/error curves for the castle, dinosaur and a tree. The x axis is the value of ε (percentage of the bounding box) and the y axis is the number of billboards. The leftmost curves are in linear units, while the rightmost curve is in log-log units.

	# of polys	error bound	# of planes	# of texels	comp. time (s)
Castle	4,064	6%	167	944 k	49
Dino	4,300	6%	110	3,151 k	51
Madcat	59,855	6%	171	1,638 k	529.0
Eiffel	13,772	6%	40	658 k	23

Figure 10: Statistics for the simplifications presented in Figures 8 and 11. Timings measured on a 2 GHz Pentium IV processor with NVidia GeForce 4 Ti4600 with 128MB.

Figure 11 shows a case where parallax and transparency effects are visually important, which would be extremely difficult to simplify using other techniques.

As shown in the table in Figure 10, computation times are reasonable for pre-computation, but too long for online simplification. The complexity of the method is essentially $O(kn)$ where n

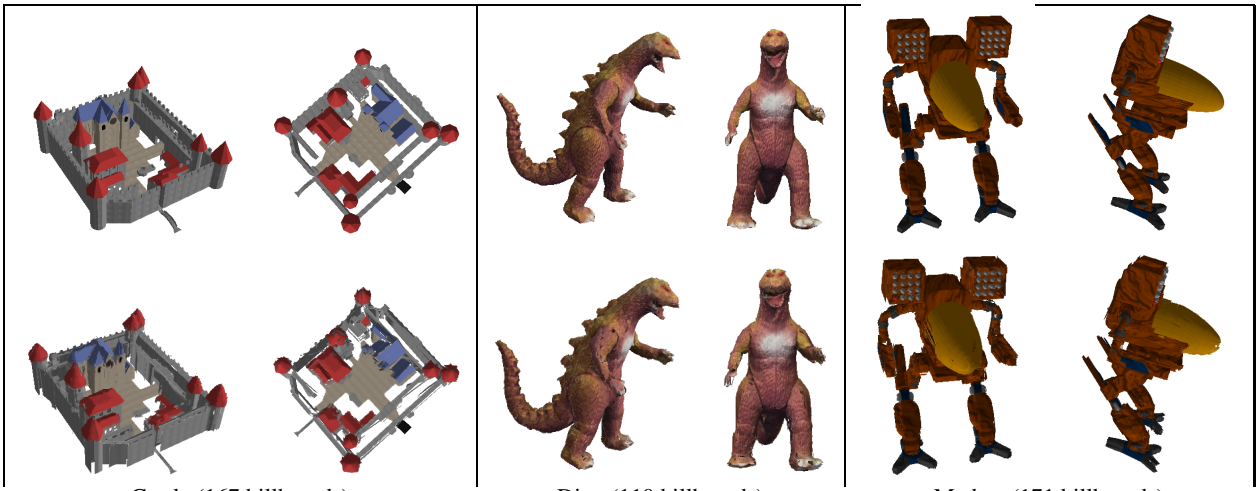


Figure 8: Results for multi-textured, possibly non-manifold models. Top row: polygonal model. Bottom row: billboard clouds.

Figure 11: An example with complex parallax and transparency through object parts. Left: polygonal model. Right: 40 billboards.

is the size of the input, and k the number of planes in the billboard cloud (density computation is $O(n)$, and each iteration costs another $O(n)$). Note that the density computation could be greatly accelerated by using only a random subset of faces. Since density is used only as a guide for plane selection, this should not affect the behavior of the method.

Table 1 provides rendering speed data for a billboard cloud. We must emphasize that rendering time measurements are always hard to interpret, because of large variability with hardware, context switches, and measurement protocols. Here we rendered many instances of a billboard cloud multiple times and took the average of the framerate to deduce a rendering time per billboard cloud. Display lists were used, and the texture binding was performed for each instance (which incurs a penalty compared to a context-sort of the objects). The projected width of the billboard was approximately 128 pixels, which is in the range expected for extremely simplified objects. Informal comparisons show that the rendering time for a 60-plane billboard cloud is comparable to that of a 250-polygon simplified mesh. The precise validity domain of billboard clouds and simplified meshes, however, deserves further study because of the rendering speed variations discussed above. In particular, billboard clouds are usually limited by fill rates and the cost of context switching. However using a billboard cloud with a larger number of planes often does not significantly impede rendering performance. Similarly, using a large number of instances can dramatically reduce the cost of each instance, because it greatly reduces the number of context switches.

If normal maps are computed together with the textures, a billboard cloud can be relit in real-time using bump mapping and pixel shaders, e.g. [Wynn n. d.], as demonstrated in Fig. 12. Billboard clouds can also be used for simple shadow effects, by projecting them onto the ground plane or using projective texture mapping [Segal et al. 1992] or textured polygons using only the alpha value of the billboard textures.

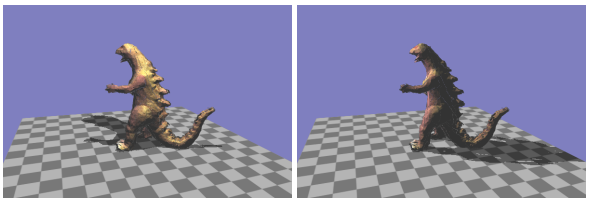


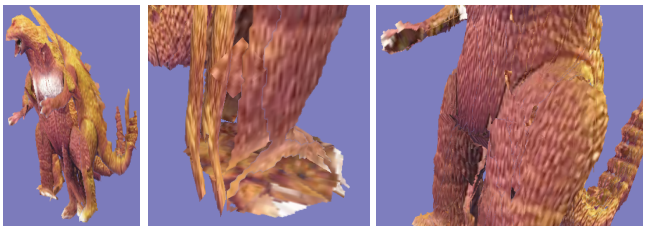
Figure 12: Relighting and shadow example (29 billboards).

In order to discuss the potential artifacts generated by image-based techniques, we use the typology proposed by Décoret et al. [Décoret et al. 1999]: Billboard clouds do not suffer from *deformation* because an error-bound is used for their construction. No *resolution mismatch* occurs because texture resolution is adaptive in object space. More importantly, we completely alleviate the *incomplete representation* issue since our image-based representation is constructed in object space and represents all triangles. This is at the cost of potentially displaying parts of the model that are not visible. A strategy such as image- or visibility-driven simplification could alleviate this limitation [Lindstrom and Turk 2000; Zhang and Turk 2002]. Our method does not suffer from *rubber sheet effects* thanks to both our object-space construction and our maximum error bound. The *cracks* are the only artifacts that are not explicitly prevented by the method. They are however greatly minimized by projecting triangles on multiple planes. We are working on reducing crack artifacts by redistributing triangles and re-optimizing plane locations in a final relaxation step.

Indeed, when the number of billboard becomes too small, a

# billboards	31	60	86	128	141	183	210	263	315	416	552
ms/frame	0.068	0.072	0.094	0.097	0.098	0.114	0.130	0.129	0.159	0.220	0.208

Table 1: Rendering time (in ms) for different number of billboards in the cloud. Bounding box of screen projection is about 128×128.



7 billboards 45 billboards 110 billboards

Figure 13: Evolution of the cracks with respect to the number of billboards. 7 billboards are not enough to produce a high-quality simplification. The general appearance is well-captured, but cracks and the billboard structure become problematic, for example in the legs. In addition, some features such as the top of the head or the tail seem doubled. With more billboards the cracks are noticeable only when zoomed.

view-independent approach cannot faithfully capture the appearance of an object and cracks occur (see Fig. 13 where the use of only 7 billboards results in cracks).

We must emphasize again that billboard clouds are not intended to replace mesh simplification for closeups of objects. Instead, they are intended for distant geometry and complex objects, when mesh-simplification techniques become less attractive. In a typical application, we believe that billboard clouds and meshes should co-exist in the same way that games currently use both meshes and cross-billboards for trees.

5 Conclusions

We have introduced the *billboard cloud* as a new representation of 3D models, which effectively decouples the visual or geometrical properties of the scene from its original description. By operating in plane space, the algorithm gains a lot of flexibility. We have presented a greedy optimization procedure for constructing billboard clouds in an error-based approach using either an object-space or image-space error metric.

Billboard clouds are effective in simplifying complex models with multiple textures into a small number of textured polygons. Compared to other simplification methods, billboard clouds maintain precise silhouettes as well as interesting parallax effects even into the range of extreme simplification. Additionally, the texture resampling step and the replacement of detailed geometry by textures allows much better filtering of complex models, and dramatically reduces aliasing artifacts.

Our results demonstrate the potential of this representation, as well as its flexibility. We plan to extend our approach to the construction of *view-dependent* billboard clouds, where the simplification is valid for a restricted region of space. We believe that view-dependent billboard clouds will facilitate even higher levels of simplification.

We believe that there exist many other applications for billboard clouds, where the expression of a complex scene as a small set of planar primitives simplifies further operations; thus there is ample room for interesting future work. This includes collision detection, but also soft shadows, where the planar nature of billboard clouds might lead to efficient approximations.

Many different error metrics should be tested with the method, including application-dependent criteria. A budget-based optimization method, providing the "best" billboard cloud for a given polygon/texture budget, would be extremely useful and only involve a modification of the optimization stage.

Depth ordering improves the rendering of semi-transparent polygons. We plan to investigate whether we can apply the idea of BSP trees, e.g. construction to enhance billboard cloud quality. Relighting can be improved by better combining the normal and shading model stored with each billboard. We currently use a simple diffuse model, and the correct handling of specularity requires the filtering of different specular components that project onto the same pixel.

Acknowledgments

This work was initiated during the first author's PhD at INRIA Rhone-Alpes, funded by the french government. This research was partially supported by an NTT partnership MIT9904-20. We thank Addy Ngan, Ray Jones and Eric Chan for comments on the paper.

Appendix - Computation of ρ_{min}, ρ_{max}

For each vertex M_i of the face, for a direction $\mathbf{d_j}$ among of the 4 directions delimiting a bin in (θ, ϕ), we compute the range $[\rho^-_{i,j}, \rho^+_{i,j}]$ of planes that intersect the sphere centered in M_i. It is given by $\rho^{\pm}_{i,j} = \mathbf{VM_i}.\mathbf{d_j} \pm \varepsilon$. We union these ranges on j and intersect them on i, that is $\rho_{min} = \max_i \min_j(\rho^-_{i,j})$ and $\rho_{max} = \min_i \max_j(\rho^+_{i,j})$. In the view-dependent case, we use $\rho^{\pm}_{i,j} = \mathbf{VP^i_{\pm}}.\mathbf{d_j}$ where $[P^i_+ P^i_-]$ is the validity segment of V_i.

References

ALIAGA, D., AND LASTRA, A. 1998. Smooth transitions in texture-based simplification. *Computers & Graphics 22*, 1, 71–81.

ALIAGA, D., AND LASTRA, A. 1999. Automatic image placement to provide a guaranteed frame rate. In *Proceedings of SIGGRAPH '99*.

ALIAGA, D. 1996. Visualization of complex models using dynamic texture-based simplification. In *IEEE Visualization '96*.

CAZALS, F., DRETTAKIS, G., AND PUECH, C. 1995. Filtering, clustering and hierarchy construction: a new solution for ray-tracing complex scenes. *Computer Graphics Forum (Eurographics '95)*.

CGAL library. `http://www.cgal.org`.

CIGNONI, P., MONTANI, C., SCOPIGNO, R., AND ROCCHINI, C. 1998. A general method for preserving attribute values on simplified meshes. In *IEEE Visualization '98*, 59–66.

COHEN, J., OLANO, M., AND MANOCHA, D. 1998. Appearance-preserving simplification. In *Proceedings of SIGGRAPH '98*.

DALLY, W. J., MCMILLAN, L., BISHOP, G., AND FUCHS, H. 1996. The delta tree: An object-centered approach to image-based rendering. Tech. rep., MIT AI Lab #1604.

DARSA, L., SILVA, B. C., AND VARSHNEY, A. 1997. Navigating static environments using image-space simplification and morphing. In *ACM Symp. Interactive 3D Graphics*.

DÉCORET, X., SILLION, F., SCHAUFLER, G., AND DORSEY, J. 1999. Multi-layered impostors for accelerated rendering. *Computer Graphics Forum (Eurographics '98) 18*, 3.

E. G. COFFMAN, J., GAREY, M. R., AND JOHNSON, D. S. 1997. Approximation algorithms for bin packing: a survey. 46–93.

ERIKSON, C., AND MANOCHA, D. 1999. GAPS: General and automatic polygonal simplification. In *ACM Symp. Interactive 3D Graphics*.

FAUGERAS, O., HEBERT, M., AND PAUCHON, E. 1983. Segmentation of range data into planar and quadratic patches. In *Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition*.

GARLAND, M., AND HECKBERT, P. 1997. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH '97*.

GARLAND, M., WILLMOTT, A., AND HECKBERT, P. 2001. Hierarchical face clustering on polygonal surfaces. In *ACM Symp. Interactive 3D Graphics*.

GROSSMAN, J. P., AND DALLY, W. 1998. Point sample rendering. In *Eurographics Rendering Workshop*.

HECKBERT, P., AND GARLAND, M. 1997. Survey of polygonal surface simplification algorithms. Tech. rep., Carnegie Mellon University.

HOCHBAUM, D., Ed. 1997. *Approximation Problems for NP-hard Problems*. PWS Publishing Company.

HOUGH, P., 1962. Method and means for recognizing complex patterns. US Patent 3,069,654.

KALVIN, A., AND TAYLOR, R. 1996. Superfaces: Polygonal mesh simplification with bounded error. *IEEE CG&A 16*, 3, 64–77.

KRAUS, M., AND ERTL, T. 2002. Adaptive Texture Maps. In *Proc. SIGGRAPH/EG Graphics Hardware Workshop '02*, 7–15.

LENGYEL, J. 2000. Real-time hair. In *Eurographics Rendering Workshop*.

LEVOY, M., AND WHITTED, T. 1985. The use of points as a display primitive. Tech. Rep. 85-022, U. of North Carolina.

LINDSTROM, P., AND TURK, G. 2000. Image-driven simplification. In *ACM Transactions on Graphics*, vol. 19, 204–241.

LOW, K., AND TAN, T. 1997. Model simplification using vertex-clustering. In *ACM Symp. Interactive 3D Graphics*.

LUEBKE, D., AND ERIKSON, C. 1997. View-dependent simplification of arbitrary polygonal environments. In *Proceedings of SIGGRAPH '97*.

LUEBKE, D. 2001. A developer's survey of polygonal simplification algorithms. *IEEE CG&A 21*, 3, 24–35.

MACIEL, P., AND SHIRLEY, P. 1995. Visual navigation of large environments using textured clusters. In *ACM Symp. Interactive 3D Graphics*.

MAX, N., DEUSSEN, O., AND KEATING, B. 1999. Hierarchical image-based rendering using texture mapping hardware. In *Eurographics Rendering Workshop*.

MEYER, A., AND NEYRET, F. 1998. Interactive volumetric textures. In *Eurographics Rendering Workshop*.

OLIVEIRA, M. M., BISHOP, G., AND MCALLISTER, D. 2000. Relief texture mapping. In *Proceedings of SIGGRAPH 2000*.

PFISTER, H., ZWICKER, M., VAN BAAR, J., AND GROSS, M. 2000. Surfels: Surface elements as rendering primitives. In *Proceedings of SIGGRAPH 2000*.

POPESCU, V., LASTRA, A., ALIAGA, D., AND DE OLIVEIRA NETO, M. 1998. Efficient warping for architectural walkthroughs using layered depth images. In *IEEE Visualization '98*.

POPOVIC, J., AND HOPPE, H. 1997. Progressive simplicial complexes. In *Proceedings of SIGGRAPH '97*.

PORTER, T., AND DUFF, T. 1984. Compositing digital images. *Computer Graphics (Proceedings of SIGGRAPH '84) 18*, 3.

PUPPO, E., AND SCOPIGNO, R., 1997. Simplification, lod and multiresolution - principles and applications.

ROSSIGNAC, J., AND BORREL, P. 1993. Multi-resolution 3D approximation for rendering complex scenes. In *2nd Conf. on Geometric Modelling in Computer Graphics*.

RUSINKIEWICZ, S., AND LEVOY, M. 2000. Qsplat: A multiresolution point rendering system for large meshes. In *Proceedings of SIGGRAPH 2000*.

SANDER, P., GU, X., GORTLER, S., HOPPE, H., AND SNYDER, J. 2000. Silhouette clipping. In *Proceedings of SIGGRAPH 2000*.

SCHAUFLER, G., AND STURZLINGER, W. 1996. A three-dimensional image cache for virtual reality. *Computer Graphics Forum (Eurographics '96) 15*, 3.

SCHAUFLER, G. 1998. Per-object image warping with layered impostors. In *Eurographics Rendering Workshop*.

SEGAL, M., KOROBKIN, C., VAN WIDENFELT, R., FORAN, J., AND HAEBERLI, P. E. 1992. Fast shadows and lighting effects using texture mapping. *Computer Graphics (Proceedings of SIGGRAPH '92) 26*, 2, 249–252.

SHADE, J., LISCHINSKI, D., SALESIN, D., DEROSE, T., AND SNYDER, J. 1996. Hierarchical image caching for accelerated walkthroughs of complex environments. In *Proceedings of SIGGRAPH '96*.

SHADE, J., GORTLER, S., HE, L., AND SZELISKI, R. 1998. Layered depth images. In *Proceedings of SIGGRAPH '98*.

SHEFFER, A. 2001. Model simplification for meshing using face clustering. *Computer-Aided Design (CAD) 33*, 925–934.

SILLION, F., DRETTAKIS, G., AND BODELET, B. 1997. Efficient impostor manipulation for real-time visualization of urban scenery. *Computer Graphics Forum (Eurographics '97) 16*, 3.

TORBORG, J., AND KAJIYA, J. 1996. Talisman: Commodity real-time 3d graphics for the pc. In *Proceedings of SIGGRAPH '96*.

WILSON, A. 2002. *Spatially Encoded Image-Space Simplifications for Interactive Walkthrough*. PhD thesis, University of North Carolina at Chapel Hill's Dpt of Computer Science.

WYNN, C. Implementing bump-mapping using register combiners. `http://developer.nvidia.com/`.

ZHANG, E., AND TURK, G. 2002. Visibility-guided simplification. In *IEEE Visualization '02*.