INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Billboard Clouds

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

N° 4485

juin 2002

THÈME 3

Rapport de recherche

ISSN 0249-6399

Billboard Clouds

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

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet iMAGIS

Rapport de recherche n° 4485 — juin 2002 — 20 pages

Abstract: We introduce *billboard clouds* – a new approach for extreme simplification. 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 very good surface approximation is ensured by favoring planes that are "nearly tangent" to the model. This method does not require connectivity information, but still avoids cracks by projecting primitives onto multiple planes when needed. The technique is quite flexible through the appropriate choice of error metrics, which can include image-space or object-space deformation, as well as any application-dependent objective function. It is fully automatic and controlled by two intuitive user-supplied parameters. 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, as well as entire scenes containing buildings and vegetation.

Key-words: geometric simplification, image-based model, error-driven simplification, computer games, accelerated rendering

* MIT (Boston)
† MIT (Boston)

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 Montbonnot-St-Martin (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Nuages de panneaux

Résumé : Nous introduisons les *nuages de panneaux* – une nouvelle approche pour la simplification extrême. Un modèle est simplifié en un ensemble de plans et de textures transparentes. Nous présentons une approche d'optimisation pour construire un nuages de panneaux étant donné un seuil d'erreur tolérable. Après avoir calculé une fonction de densité appropriée dans l'espace des plans, une approche gloutonne est utilisée pour sélectionner des plans représentatifs. Une très bonne approximation des surfaces est obtenue en favorisant les plans « presques tangents » au modèle. Cette méthode ne requiert aucune information de connectivité et pourtant évite les craquelures en projetant les primitives sur plusieurs plans quand cela est nécessaire. La technique est assez flexible grâce à l'emploi de métriques d'erreur appropriées travaillant dans l'espace image comme dans l'espace objet et pouvant intégrer des fonctions d'objectifs spécifiques à une application donnée. La méthode est complètement automatique et contrôlée par deux paramètres intuitifs spécifiés par l'utilisateur. Pour la simplification extrême, notre approche combine la force de la décimation de maillage et des imposteurs à base d'images. Nous démontrons notre technique sur une large classe de modèles, notamment des variétés, des objets composés ou encore des scènes entières contenant des bâtiments et de la végétation.

Mots-clés : simplification géométrique, modèle à base d'image, simplification orientée erreur, jeux vidéos, accélération de rendu

1 Introduction

This paper introduces a new approach for *extreme simplification*, that is the simplification of complex 3D models to models several orders of magnitude smaller. It is based on a new representation, which we call *billboard clouds*, composed of a set of planar polygons with texture and transparency maps. We show how to construct billboard clouds with a fully automatic optimization procedure.

Virtual environments continue to grow in both geometric and visual complexity. The demand for realism and visual detail is such that virtual models are likely to exceed the raw capacity of rendering hardware for several years to come, despite the expected improvements in graphics hardware technology, as long as real-time viewing is required. An important challenge for Computer Graphics is therefore to adapt the representation of the virtual model to the available resources, by constructing a *simplified model* that guarantees an “optimal" combination of image quality and rendering speed. Simplified models are also required to perform computationally intensive tasks such as collision detection, in CAD/CAM or gaming applications.

Of course the notion of optimality is ill-defined, and largely application dependent. However it is clear that in general we want to maximize visual fidelity (the appearance of the simplified model should be as close as possible to that of the original), and minimize rendering cost (measured in relation to the resources available, e.g. in terms of polygon count and/or total texture usage per frame). Other constraints can be considered, such as the ability to modify lighting conditions in the scene, or to blend in other models or images.

It is important to note that model simplification is not required solely for performance reasons. As models become more precise geometrically, they typically incorporate explicit models of smaller details and parts, as well as richer textures. The limited resolution of all existing display technologies therefore produces aliasing problems, and the model should be *pre-filtered*, or *resampled*, to minimize such artifacts as flickering and disappearing objects.

Mesh decimation has dramatically progressed over the last decade, and techniques such as edge collapse permit fast, efficient and accurate simplification [15, 35, 24]. 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, as well as to include transparency effects. Another great difficulty with extreme simplification is to maintain visual detail: While purely geometric simplification can be conducted to within well-defined error bounds, it is all but impossible with existing approaches to guarantee a specific level of visual fidelity. Large models combining many different textures are especially difficult to simplify, since some vertices have to be kept 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 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” that most gamers complain about.

The representation introduced in this paper is both very generic and extremely flexible, spanning a great range of applicability which includes extremely simplified models. Previous techniques typically perform well only on either smooth connected manifolds or on distant geometry with little parallax effects. Billboard clouds effectively bridge this gap and permit extreme simplification of arbitrary models, with good visual fidelity including appropriate parallax and silhouettes.

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. We outline related approaches, and highlight the key differences with our technique.

We must emphasize that our method does not, nor is it intended to, compete 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.

Mesh decimation, e.g. [15, 35, 24] has mostly focused on the simplification of single manifolds, where the simplified model has the same topology as the input. Nonetheless, vertex clustering [37, 25, 23] can naturally merge multiple objects into a single representation. Edge collapse can also be adapted to handle virtual edges between nearby vertices [12, 11]. Some approaches use lower-dimensional simplices such as lines and points to better represent shapes after the fusion of multiple parts [33, 23]. 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 to represent complex geometry. Moreover, the handling of multiple textures is not trivial, and basically requires the computation of new textures [7]. While our approach does also require new textures, we make this texture computation an inherent part of the representation, which essentially decouples texture and geometry complexity, while allowing complex transparency effects.

Silhouette clipping [39] dramatically improves rendering quality by clipping extremely simplified models with the real silhouette of the input model. It is however costly, works only on manifolds, and does not improve the appearance of internal silhouettes.

Our technique is closely related to superfaces [19] and face clustering [13, 44], which group connected sets of nearly coplanar faces. Coplanarity can be defined by the normal of faces [19], or using quadric error metrics [13]. Like our representation, face clusters can be used for lighting simulation, collision detection, or fast display. In the latter case, they need to be triangulated 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* [26] . In particular, they describe the use of classical billboards [36], and bounding boxes textured with rendered images of the objects. Shade et al. [43] and Schaufler et al. [41] dynamically cache parts of the scene, and render them on billboards parallel to the image plane. Meshed impostors, e.g. [45, 9, 2], layered depth images [42, 32] and layered impostors [40, 10] 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. [1, 3], or optimize the placement of impostors [4]. The Talisman graphics hardware system also uses an image-based strategy to reduce the load of the 3D engine. The image is decomposed into sprites, and 2D image transformation are used [46, 21].

Parallel textured planes similar to layered impostors have also been used to render volumetric objects such as trees or hair [29, 20]. Max [27, 28] has developed specialized techniques to exploit the natural hierarchy of objects such as trees. Point-based rendering [22, 14, 38, 47], and surfels [31] are other alternative representations for fast display that work well with complex objects such as vegetation. Unfortunately, current graphics hardware is not optimized for the rendering of such primitives. Hybrid approaches have also been developed to switch between impostors and simplified meshes [2], or points and simplified meshes [8]. This permits the selection of the more appropriate representation, but preserves the limitations of the underlying approaches: limited parallax for impostors, and rendering performance for points.

2 Billboard clouds

In this section we introduce the notion of a *billboard cloud* as an alternative representation of a complex 3D model. This new approach has strong potentials for many applications and opens a variety of research issues, which we outline in this section. Our technique will then be presented in Section 3 through 5.

2.1 Definition of a *billboard cloud*

We define a *billboard cloud* as a set of textured, partially transparent polygons (or billboard), with independent size, orientation and texture resolution (see Figure 1). Such a cloud can be used to speed up the display of a complex model, to conduct collision checks, to perform radiative transfer simulations using radiosity techniques, or in any situation where the description of a model as a (small) set of planar primitives is useful.

Thus a billboard cloud is essentially a set of independent, alpha-blended, textured polygons. 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.

Depending on the particular usage context, the texture of each billboard can either be completely computed in object space, or incorporate some view-dependent information. Textures can also include more than simple color information, to take advantage of advanced rendering capabilities of modern graphics cards (e.g. using normal maps or pixel shaders). Billboard clouds are rendered by rendering each polygon independently.

Billboard clouds can be used to render face clusters, trading geometry (triangulation of the cluster) for texture (alpha map of the cluster). 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 (classical billboards, stacked impostors where all billboards are parallel, image caches where a billboard is associated with each region of a space partition, etc.).

2.2 Obtaining a billboard cloud

We express the generation of a billboard cloud from a 3D model as an optimization problem: the input to the problem is a virtual model consisting of textured geometry in any form. The output is a billboard cloud. We assume that two scalar functions are defined for any billboard cloud: first, an error function, second a cost function.

The error norm can be e.g. the geometric L_∞ or L_2 norm between the input and points in the billboard cloud, or any application-dependent metric. It can be view-independent or view-dependent, and can include perceptual aspects. The cost can include dimensions such as the number of textured planes in the representation, but also the total memory required by the textures.

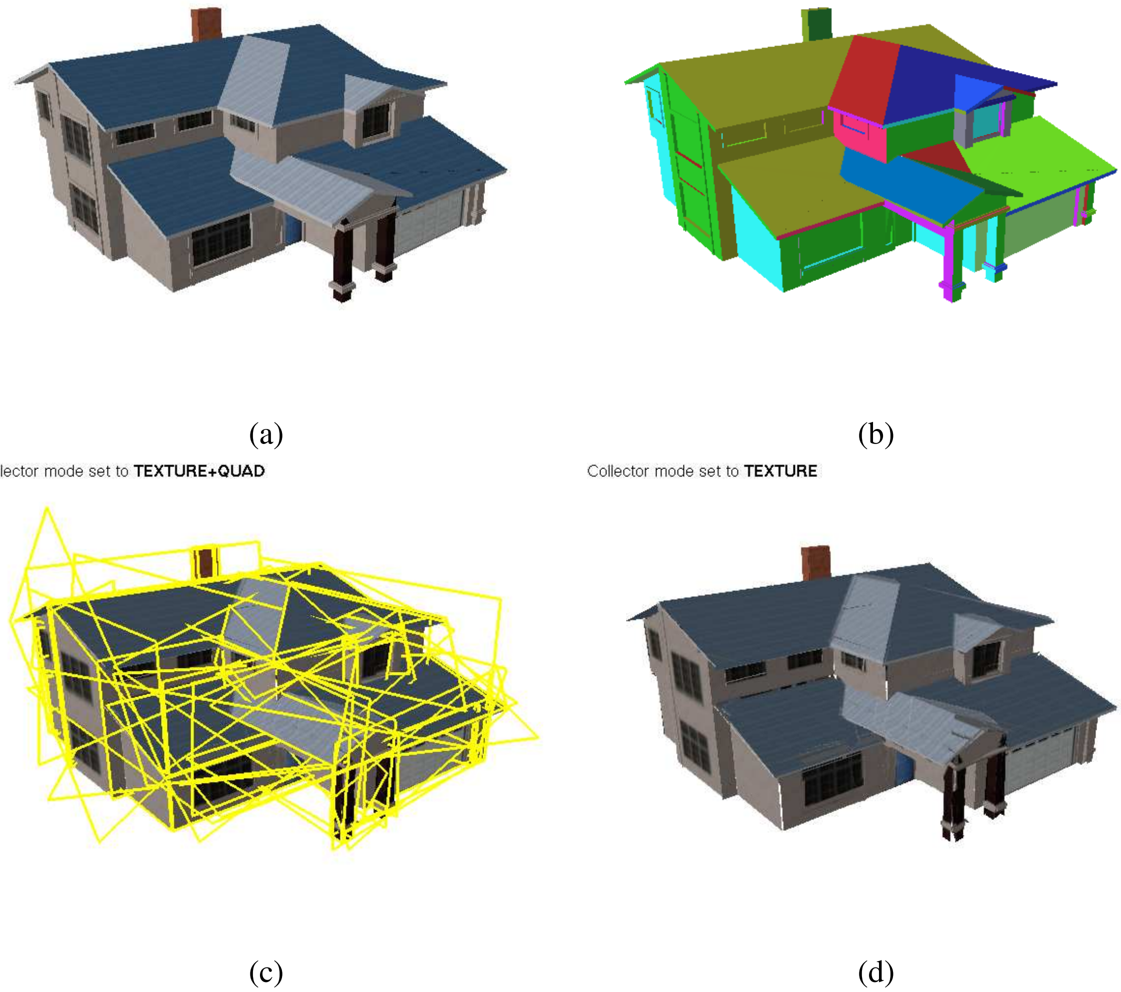


Figure 1: Example of a billboard cloud: (a) Original model (1,960 polygons) (b) false-color rendering using one color per billboard to show the faces that were grouped on it (c) View of the (automatically generated) 52 textured billboards, with their border highlighted (d) rendered billboard cloud.

Two distinct strategies exist: *budget-based* and *error-based* simplification. In the former, a budget is given for the cost (number of planes, total texture size), and the goal is to build the billboard cloud that minimizes the error. In the latter, a maximum tolerable error ϵ is set, and the aim is to build the billboard cloud respecting this threshold with the minimum cost.

2.3 Overview of the new technique

We describe here an error-based optimization. Our input is a polygonal model, and no connectivity information is required. We propose two error metrics based on the Euclidean L_∞ norm: one in object-space that is view-independent, and one image-space metric with respect to a volumetric view-cell. The cost function is simply defined as the number of planes. The cost of texture size is not included in our implementation, but is discussed in Section 7.

Our algorithm is a greedy optimization guided by a density computed in plane space. 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 number of faces that can be simplified by a plane.

We first present our method for the view-independent case. In Section 3, we define density and explain how to rasterize it onto a grid in plane space. Our greedy plane selection is then described in Section 4. Finally, in Section 5, we extend the method to the view-dependent with respect to a volumetric view-cell.

3 Density in plane space

We present a generic strategy to estimate the relevance of a plane, which allows us to compute a *density* in plane space. Density is defined using three important notions: *validity*, *contribution* and *penalty*. Validity is binary and enforces the absolute error bound. Contribution includes an additional notion of representation quality. Finally, penalty prevents undesirable planes. Extreme flexibility is gained by specifying different validity or contribution functions, for instance to introduce an application-dependent bias (e.g. to favor vertical planes), or an orientation bias similar to Garland et al.’s [13].

We define the density $d(\mathcal{P})$ of a plane $\mathcal{P}$ as :

$$\begin{aligned} d(\mathcal{P}) &= C(\mathcal{P}) - Penalty(\mathcal{P}) \\ &= \sum_{f \in valid_\epsilon(\mathcal{P})} C(f) - Penalty(\mathcal{P}) \end{aligned}$$

where $C(\mathcal{P})$ is the contribution, which depends on the set of faces f for which $\mathcal{P}$ is a *valid* representation, and clamp $d(\mathcal{P})$ to zero to ensure positive values.

In this section, we discuss our choices of validity, contribution and penalty for the view-independent case. The extension to view-dependent will be discussed in Section 5. We consider oriented planes, that is, the orientation of the normal matters.

3.1 Validity

In the view-independent case, 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 $||vp|| < \epsilon$. We define the *validity domain* of a point v for an error bound ϵ as the set of planes that can represent it, which we note $valid_\epsilon(v)$. In the view-independent case, it is the set of planes that intersect the sphere of radius ϵ centered on v.

A plane can represent a face only if it can represent all its points. A plane is valid for a polygon if it is valid for all its vertices. Note that this is true also for concave polygons, but that the vertices of the convex hull yield a sufficient condition. Without loss of generality we will consider in the remainder of the paper all polygons to be triangles, although a major benefit of our approach is that models need not be triangulated [18]. The validity domain $valid_\epsilon(f)$ of a face f is thus 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 $valid_\epsilon(\mathcal{P})$.

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 $f \in valid_\epsilon(\mathcal{P}_i)$. Geometric cover problems are known to be NP-hard, and the discrete version, the set cover problem, is NP-complete [16].

3.2 Contribution

The size of the validity set of a plane is a good initial estimate of its relevance. For better accuracy we weight the contribution of each valid face by its projected area. For the view independent case, the projected area $area_\mathcal{P}(f)$ is 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 contribution of a plane $\mathcal{P}$ is then:

$$C(\mathcal{P}) = \sum_{f \in valid_\epsilon(\mathcal{P})} area_\mathcal{P}(f) \tag{1}$$

3.3 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.

Consider the example of Fig. 2 where we want to simplify a tessellated sphere. Consider the set of planes with normal $\vec{n}$. The validity domain of face f is the set of planes between $\mathcal{P}_1$ and $\mathcal{P}_2$. On the left of $\mathcal{P}_2$, planes will "miss" f. We introduce the missing set $miss_\epsilon(f)$ 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 $\mathcal{P} \notin valid_\epsilon(f)$ and if there exists a plane $\mathcal{P}' \in valid_\epsilon(f)$ such that $\mathcal{P}'$ is the image of $\mathcal{P}$ by a translation of $a\vec{n}$ with $0 \leq a \leq \epsilon$ (Fig. 2). Thus in the view-independent case, in a given direction, $valid_\epsilon(f)$ and $miss_\epsilon(f)$ are two contiguous segments of length 2ϵ and ϵ. Note that $miss_\epsilon(f)$ is on only one side of $valid_\epsilon(f)$. This direction dependency is quite important for the robustness of the method. The quasi-tangents on the left are captured by the opposite direction $-\vec{n}$

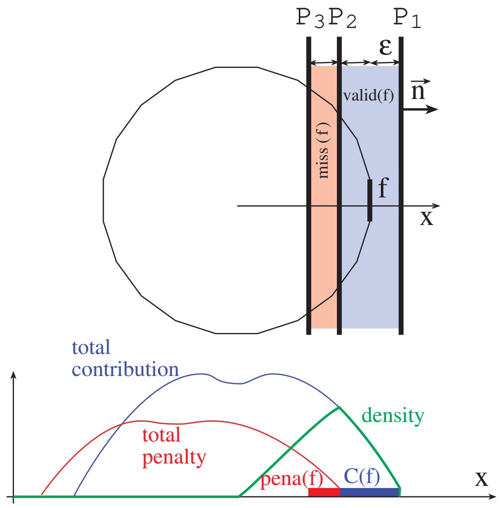


Figure 2: Near-tangency and penalty due to missed primitives.

We do not use an infinite width for $miss(f)$ because it would basically restrict 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.

The penalty formula is then similar to the contribution, but it is weighted more strongly.

$$penalty(\mathcal{P}) = w_{penalty} \sum_{f \in miss(\mathcal{P})} area_{\mathcal{P}}(f) \tag{2}$$

In practice, we use $w_{penalty} = 10$, but we noticed that the behavior of the method was robust to the setting of $w_{penalty}$. As can be seen from the bottom part of Fig. 2, we simply need a weight that is large enough to strongly prevent the choice of planes that miss primitives.

Fig. 2 shows a plot of the contribution and penalty, and the resulting density. The density behaves as we wish, it favors the plane that is quasi tangent to the sphere, and that is valid for the maximum number of faces. Recall that the planes are oriented and that the tangency on the right will be captured by the symmetric direction.

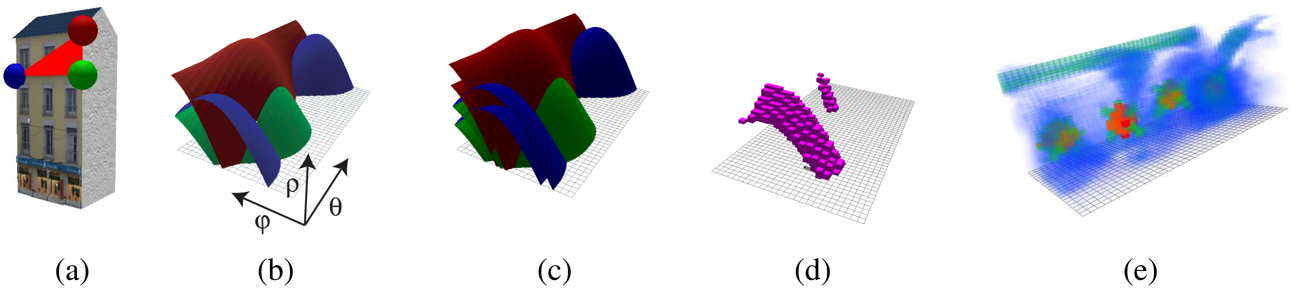


Figure 3: Density in plane space. (a) Scene with face f and its three vertices highlighted. (b) Set of planes going through each vertex. represented in plane space. The plane of f corresponds to the intersection of the three sheets. (c) Validity domain of each vertex. (d) Discretized validity domain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

3.4 Discretization of plane space

We estimate density in a discretized plane space. For this, we need a parameterization of planes. We use spherical coordinates (θ, φ) for the normal direction, and the distance ρ to the origin. This is the 3D equivalent of the Hough transform [17]. 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 just be oversampled. A more uniform parameterization based on icosahedra could be used, but they would make the code more complex for little gain. In contrast, the separation of direction and distance is important, because many computations are simpler on a set of parallel planes.

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 $-\epsilon$ and $+\epsilon$ 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 then 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 using the formula described above. A naïve approach would use the density in the middle of the bin (sampling). This would result in artifacts (some contributions would be overlooked). This is why we use a more conservative rasterization. For a given face f, we consider all bins that intersect $valid_\epsilon(f)$. Such bins are said to be *simply* valid in the sense that there exists one plane in the bin that is valid for f. Note that if a bin is simply valid for two faces f_1 and f_2, there is not necessarily a plane that is valid for both faces. However, 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 then obtained by summing the contributions and penalties induced by the set of faces $valid_\epsilon(\mathcal{B})$ 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 $valid_\epsilon(f)$ (see appendix). We compute $C(f)$, which is constant for the given direction, and add it *as a contribution* to the bins between ρ_{min} and ρ_{max} (in blue on figure 4) and *as a penalty* to the bins between $\rho_{min} - \epsilon$ 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).

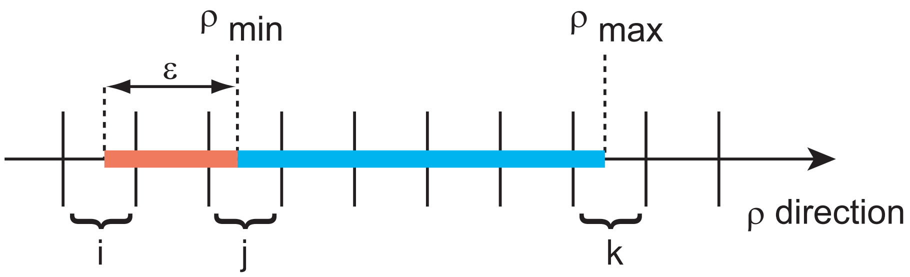


Figure 4: Rasterization in plane space.

4 Greedy optimization

Now that we have defined a density over plane space and shown how to compute a discretized version, we can present our greedy optimization to select a set of planes that approximate the input model.

Our greedy strategy proceeds as follow: 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

```
Greedy (input model, threshold ε)
  set of faces F=input model
  billboard cloud BC = ∅
  while F ≠ ∅
    Pick bin B with highest density
    Compute valid_ε^F(B)
    P_i = RefineBin (B, valid_ε^F(B))
    UpdateDensity(valid_ε^F(P_i))
    F= F \ valid_ε^F(P_i)
    BC = BC ∪ P_i
  Compute textures(BC)
```

Figure 5: Pseudo code of the greedy selection of planes.

```
RefineBin (bin B, set of faces F)
  plane P = center of B
  if (valid_ε^F(P) == valid_ε^F(B))
    return P
  bin B_max=NULL
  for each of the 27 neighbors B_i of B
    Subdivide B_i into 8 sub-bins B_ij
    for each B_ij  // there is a total of 8*27 such B_ij
      Compute d^F(B_ij)
      if (d^F(B_ij) > d^F(B_max))
        B_max = B_ij
  return RefineBin (B_max, valid_ε^F(B_max))
```

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

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.

4.1 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 $valid_\epsilon(\mathcal{B})$. We iterate on the faces that have not been collapsed yet to compute those that are valid for $\mathcal{B}$. Further computation 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 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 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.

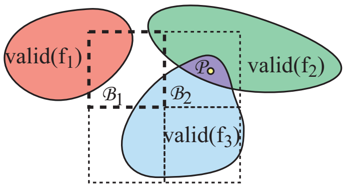


Figure 7: Simple density in plane space. Although bin $\mathcal{B}_1$ has maximum simple density, the densest plane is $\mathcal{P}$, which is in bin $\mathcal{B}_2$.

This pick-and-refine phase can be understood as follows: we reduce the problems of geometric cover (finding an *unknown* number of planes that approximate the *whole* model) to the problem of finding *one* plane that approximates a *subset* of the model. We use high density as a hint to find a subset for which we may find such a plane. We then test planes against this set, removing irrelevant faces and adapting the plane to the remaining ones until it is valid for all of them.

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 3.4, except that contributions and penalties are *removed*. The algorithm then proceeds until all faces of the model are collapsed on planes.

4.2 Shooting textures

Each plane $\mathcal{P}_i$ is assigned with a set of collapsed faces $valid_\epsilon^{\mathcal{F}}(\mathcal{P}_i)$ during the greedy phase. We first find the minimum bounding rectangle of the projection of these faces in the texture plane (using the CGAL Computational Geometry library [6]), 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 as advised by Porter and Duff [34][1].

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 textured polygons 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.

Note that texture shooting is important not only for real-time rendering, but that the alpha map is also crucial if the billboard cloud is to be used for collision detection or finite-element lighting simulation.

[1] In OpenGL, the billboards then have to be rendered using the blending parameters `glBlendFunc(GL_ONE, GL_ONE_MINUS_SRC_ALPHA)`.

5 View-dependent optimization

We now adapt our billboard cloud construction to the view-dependent case, where we minimize error with respect to a volumetric view-cell. View-dependent image-based acceleration has been shown to be useful in a variety of situations, e.g. [45, 9, 2, 2]. Following these approaches, a reference viewpoint in the view-cell is used to compute the textures, but also to define validity. We choose to build a billboard cloud such that the view of the billboard cloud is strictly correct from the reference viewpoint. This results in a more consistent billboard cloud, which is moreover less prone to cracks. We first derive a formula for the validity domains, then discuss other minor modifications to the construction algorithm.

5.1 Validity

In this section, we assume that the reference viewpoint V has been chosen. In order to obtain a correct view from V, a vertex M of the model is allowed to be simplified only along a segment of the line (VM). If M is simplified to a position P on (VM), the reprojection error for another viewpoint T is defined as the solid angle under which $[MP]$ is seen from T (figure 8). This reprojection is maximal when (MT) is tangent to the view-cell and $|MT|$ minimum. The segment $[P_-, P_+]$ in which M can be translated for a reprojection error less than θ_{max} is defined by:

$$|VP_\pm| = \frac{|VM|}{1 \pm \tan\theta \sqrt{\frac{|VM|^2}{|VT|^2} - 1}}, \tag{3}$$

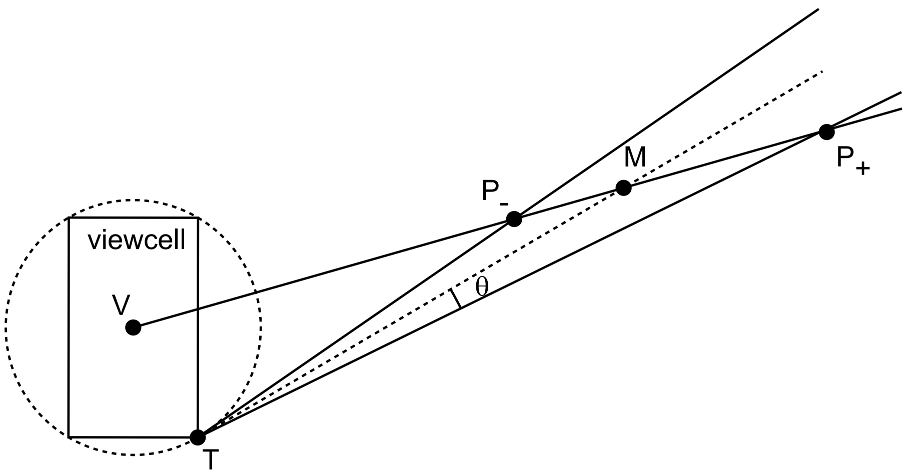


Figure 8: Reprojection error

To convert $\tan(\theta_{max})$ to a pixel distance in image space, we use a linearization of the viewing angle θ in function of the length e in pixel:

$$\tan(\theta) = \frac{1}{2n} \times \frac{w}{W} \times e \tag{4}$$

where n is the distance of the near plane of the camera, w the width of its screen (in world units) and W the width of the viewport (in pixels).

The contribution $C(f)$ is then defined as the projected area in a reference view from V. It does not depend on the direction of the plane, but favors faces that are most visible in the reference view.

5.2 Billboard cloud construction

We choose the center of the view cell as our reference point for shooting the textures. Note that this typical choice is indeed optimal in terms of minimizing the reprojection error, for a large set of vertices at a significant distance from the view-cell [30].

With our view-dependent definition of validity, the set of planes that intersects a validity domain is no longer an envelope of constant height in the ρ dimension. However, the computations are almost the same, as detailed in appendix.

We compute the validity domains using (3) and run the greedy optimization to find a set of planes. Textures are then shot from the reference viewpoint, and the transformation matrix is stored along with the texture. It will be used as texture matrix when rendering the billboard, to account for perspective distortion.

6 Implementation and results

Castle (106 billboards) Dino (86 billboards) Madcat (71 billboards)

Figure 9: Results for complex, multi-textured, possibly non-manifold models. Top row: polygonal model. Bottom row: billboard clouds.

We demonstrate the results of a simple implementation of our greedy optimization technique. We have focused in this demonstration on the visual quality of the results, without necessarily optimizing for resource usage. Specifically we have investigated the geometric cover problem without introducing the issue of texture compacity, which is discussed at the end of this section. The only input parameters to our algorithm are (a) an (object-space or image-space) error threshold and (b) the maximum texture resolution in object space. 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 very robust with respect to internal parameter choices such as the resolution of the discrete plane space.

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. However we note that the number of *opaque* texels better represents the actual information content of the textures, and is typically only a fraction of the total number of texels. In the examples below the proportion of useful texels is always less than 25%. This suggests that the notion of texture compacity should be incorporated with great profit in the greedy optimization phase. Also note that texture compression can typically reduce memory requirements by a factor of 4 to 8.

Figure 9 shows images of a number of models and of corresponding billboard clouds. Note that we are in the range of extreme simplification, since we typically only obtain fewer than 100 billboards. Still both the silhouette and important 3D structure of the objects are well captured (also see accompanying video).

	# of polys	error bound	# of planes	# of texels	comp. time (s)
Castle	4,064	0.01	106	218 k	8.5
Dino	4,300	0.03	86	3,935 k	51.0
Madcat	59,855	0.03	71	5,884 k	129.0
Eiffel	13,772	0.04	20	1,105 k	14.2
Farm	174,325	0.01	275	4,527 k	491.6

Figure 10: Statistics for the view-independent simplifications presented on figures 9,12 and 13. Timings measured on a 800 MHz Pentium III processor.

Figure 11 shows different billboard clouds obtained for a tree with varying error thresholds. In this case we have used the view-dependent error measure of Section 5, and we observe as expected a bounded image-space error as the object recedes in the distance, while fewer billboards are constructed.

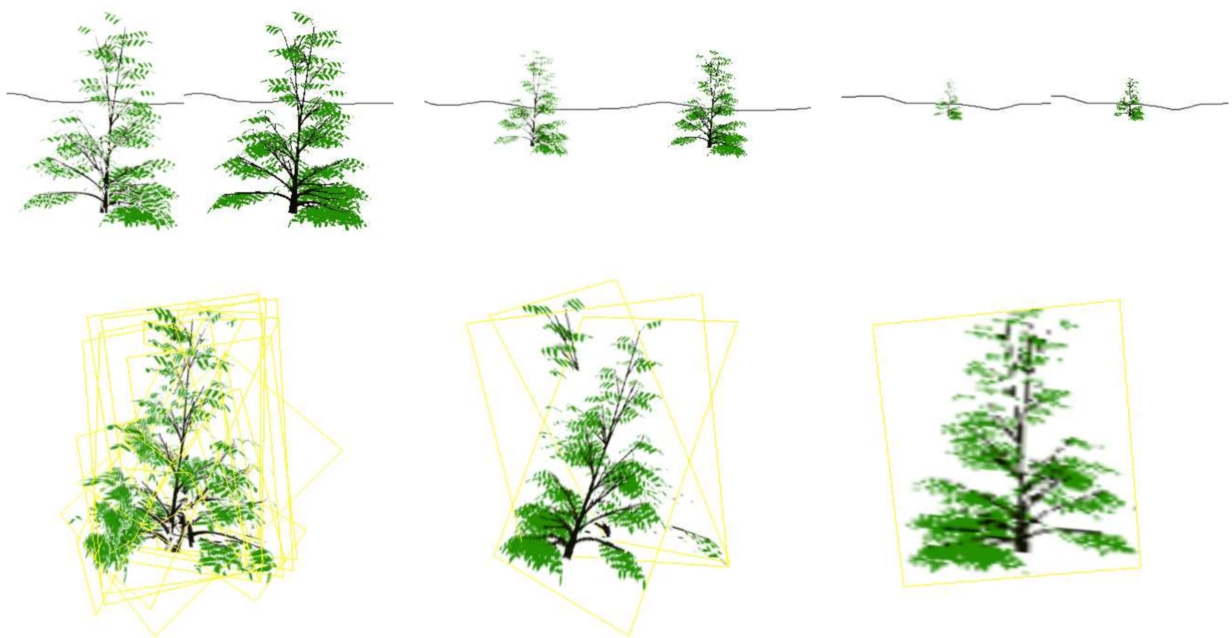
Figure 12 shows a case where parallax and transparency effects are visually important, which would be extremely difficult to simplify using other techniques. Figure 13 shows that our technique can also be applied directly to an entire scene, instead of a per-object basis. The extreme simplification of this model allows real-time rendering with *increased* visual quality thanks to automatic texture filtering.

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

6.1 Optimizing texture usage

The version of the optimization we have implemented does not explicitly take texture size into account. In this section, we show how the compactness of texture can be optimized as well, using two complementary strategies. First, we modify the definition of density to include a *compactness* term, and next, we adapt the greedy construction algorithm to restrict the primitives collapsed on a plane to *maximal compact sets*.

The contribution of a plane as defined by Eq. 1 can be due to distant primitives. In order to favor planes where valid primitives are more compactly distributed, we propose to include a compactness term similar to Garland et al.'s compact shape bias [13]. We store with each Bin $\mathcal{B}_i$ of plane space the



(a) 20 billboards (b) 3 billboards (c) 1 billboard

Figure 11: View-dependent simplification, with varying distance to the view cell. The top row shows a side-by-side renderings of the polygonal model (right-hand side) and the billboard cloud (shown underneath from close-up and a different angle), as seen from the view-cell.

bounding box of $valid_\epsilon(\mathcal{B}_i)$, which will allow us to compute an irregularity term proportional to the squared perimeter divided by the area [13].

We then propose to modify the greedy algorithm and restrict $valid_\epsilon^{\mathcal{F}}(\mathcal{B})$ 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 $valid_\epsilon^{\mathcal{F}}(\mathcal{B})$ into *clusters* and use only the cluster with the highest density. The bucket-like partitioning algorithm by Cazals et al. [5], using an appropriate criterion of compactness, is ideally suited to this task. The rest of the greedy algorithm remains unchanged, and other primitives in $valid_\epsilon^{\mathcal{F}}(\mathcal{B})$ will be handled by subsequent iteration of the greedy selection. Note that a plane might therefore be selected multiple times to handle different clusters The development of an effective criterion of compactness based on the fill-rate/geometry performances of the hardware is a crucial aspect of future work.

7 Conclusions

We have introduced the notion of *billboard clouds* 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 the space of supporting planes, the algorithm gains a lot of flexibility compared to geometric simplification techniques that operate in regular 3d space.

Billboard clouds are highly effective in simplifying complex models with multiple textures into a few tens of textured polygons. The billboard cloud representation then allows very fast rendering, or efficient collision testing. The visual quality of the simplified models is very high, and controlled by the user: compared to other simplification methods, either view-dependent or not, billboard clouds

Figure 12: An example with complex parallax and transparency through object parts. Left: polygonal model. Right: 20 billboards.

Figure 13: Simplification of an entire scene (left) into 245 billboards (right).

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 comprising small parts, and dramatically reduces aliasing artifacts.

We have presented a greedy optimization procedure for constructing billboard clouds in an error-based approach, where we guarantee a maximal error bound using either an object-space or image-space error metric. This is accomplished by expressing the problem as a geometric cover in plane space. This construction method provides a complete range of models, from accurate/complex ones down to the limit case of a single billboard.

Our results demonstrate the great potential of this representation, as well as its flexibility: quite surprisingly, the technique behaves very well even with a view-independent metric, and (perhaps more naturally) it provides excellent results in the view-dependent case.

We believe that there exist many 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. 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 involves a modification of the optimization stage. Extending the texture maps with normal and/or shading parameters should allow real-time re-lighting of billboard clouds.

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 $\rho^{\pm}_{i,j} = \mathbf{VM_i}.\mathbf{d_j} \pm \epsilon$. 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

[1] D. Aliaga. Visualization of complex models using dynamic texture-based simplification. In *IEEE Visualization*, 1996.

[2] D. Aliaga, J. Cohen, A. Wilson, H. Zhang, C. Erikson, K. Hoff, T. Hudson, W. Stürzlinger, E. Baker, R. Bastos, M. Whitton, F. Brooks, and D. Manocha. Mmr: An interactive massive model rendering system using geometric and image-based acceleration. In *ACM Symp. Interactive 3D Graphics*, 1999.

[3] D. Aliaga and A. Lastra. Smooth transitions in texture-based simplification. *Computers & Graphics*, 22(1):71–81, 1998.

[4] D. Aliaga and A. Lastra. Automatic image placement to provide a guaranteed frame rate. In *Proc. SIGGRAPH*, Computer Graphics, Annual Conference Series, 1999.

[5] F. Cazals, G. Drettakis, and C. Puech. Filtering, clustering and hierarchy construction: a new solution for ray-tracing complex scenes. *Computer Graphics Forum, Proc. Eurographics*, 1995.

[6] CGAL library. http://www.CGAL.org.

[7] J. Cohen, M. Olano, and D. Manocha. Appearance-preserving simplification. In *Proc. SIGGRAPH*, Annual Conference Series, 1998.

[8] J. D. Cohen, D. G. Aliaga, and W. Zhang. Hybrid simplification: combining multi-resolution polygon and point rendering. In *IEEE Visualization*, 2001.

[9] L. Darsa, B. Costa Silva, and A. Varshney. Navigating static environments using image-space simplification and morphing. In *ACM Symp. Interactive 3D Graphics*, 1997.

[10] X. Décoret, F. Sillion, G. Schaufler, and J. Dorsey. Multi-layered impostors for accelerated rendering. *Computer Graphics Forum (Proc. Eurographics)*, 18(3), 1999.

[11] C. Erikson and D. Manocha. Gaps: General and automatic polygonal simplification. In *ACM Symp. on Interactive 3D Graphics*, 1999.

[12] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. *Proceedings of SIGGRAPH 97*, pages 209–216, August 1997.

[13] M. Garland, A. Willmott, and P. Heckbert. Hierarchical face clustering on polygonal surfaces. In *ACM Symp. on Interactive 3D Graphics*, 2001.

[14] J. P. Grossman and W. Dally. Point sample rendering. In *Eurographics Rendering Workshop*, 1998.

[15] Paul S. Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. Technical report, CS Dept., Carnegie Mellon U., 1997.

[16] D. Hochbaum, editor. *Approximation Problems for NP-hard Problems*. PWS Publishing Company, Boston, MA, 1997.

[17] P. Hough. Method and means for recognizing complex patterns, 1962.

[18] Martin Isenburg and Jack Snoeylink. Face fixer: Compressing polygon meshes with properties. In *Siggraph 2000, Computer Graphics Proceedings*, Annual Conference Series, pages 263–270, 2000.

[19] A. Kalvin and R. Taylor. Superfaces: Polygonal mesh simplification with bounded error. *IEEE Computer Graphics & Applications*, 16(3):64–77, 1996.

[20] J. Lengyel. Real-time hair. In *Eurographics Workshop on Rendering*, 2000.

[21] J. Lengyel and J. Snyder. Rendering with coherent layers. In *Proc. SIGGRAPH*, Annual Conference Series, 1997.

[22] M. Levoy and T. Whitted. The use of points as a display primitive. Tech. Report 85-022, U. of North Carolina at Chapel Hill, 1985.

[23] K. Low and T. Tan. Model simplification using vertex-clustering. In *ACM Sym. on Interactive 3D Graphics*, 1997.

[24] D. Luebke. A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics & Applications*, 21(3):24–35, 2001.

[25] D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygonal environments. *Proc. SIGGRAPH*, 1997.

[26] P. Maciel and P. Shirley. Visual navigation of large environments using textured clusters. In *ACM Symp. Interactive 3D Graphics*, 1995.

[27] N. Max. Hierarchical rendering of trees from precomputed multi-layer z-buffers. In *Eurographics Rendering Workshop*, 1996.

[28] N. Max, O. Deussen, and B. Keating. Hierarchical image-based rendering using texture mapping hardware. In *Eurographics Rendering Workshop*, 1999.

[29] A. Meyer and F. Neyret. Interactive volumetric textures. In *Eurographics Rendering Workshop*, 1998.

[30] A. Nonymous. *Title withheld for anonymity*. PhD thesis, The University, 2002.

[31] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Proc. SIGGRAPH 2000*, Computer Graphics, Annual Conference Series, pages 335–342, 2000.

[32] V. Popescu, A. Lastra, D. Aliaga, and M. de Oliveira Neto. Efficient warping for architectural walkthroughs using layered depth images. In *IEEE Visualization*, 1998.

[33] J. Popovic and H. Hoppe. Progressive simplicial complexes. *Proc. SIGGRAPH*, 1997.

[34] Thomas Porter and Tom Duff. Compositing digital images. In *Computer Graphics (Proc. SIGGRAPH)*, 1984.

[35] E. Puppo and R. Scopigno. Simplification, lod and multiresolution - principles and applications. In *Eurographics'97 Tutorial*, 1997.

[36] J. Rohlf and J. Helman. IRIS performer: A high performance multiprocessing toolkit for real–Time 3D graphics. In *Proc. SIGGRAPH*, 1994.

[37] J. Rossignac and P. Borrel. Multi-resolution 3D approximation for rendering complex scenes. In *2nd Conf. on Geometric Modelling in Computer Graphics*, 1993.

[38] S. Rusinkiewicz and M. Levoy. Qsplat: A multiresolution point rendering system for large meshes. In *Proc. SIGGRAPH 2000*, Computer Graphics, Annual Conference Series, pages 343–352, 2000.

[39] P. Sander, X. Gu, S. Gortler, H. Hoppe, and J. Snyder. Silhouette clipping. In *Proc. SIGGRAPH 2000*, Computer Graphics, Annual Conference Series, 2000.

[40] G. Schaufler. Per-object image warping with layered impostors. In *Eurographics Rendering Workshop*, 1998.

[41] G. Schaufler and W. Sturzlinger. A three-dimensional image cache for virtual reality. *Computer Graphics Forum*, 15(3), 1996.

[42] J. Shade, S. Gortler, L. He, and R. Szeliski. Layered depth images. In *Proc. SIGGRAPH*, Annual Conference Series, 1998.

[43] J. Shade, D. Lischinski, D. Salesin, T. DeRose, and J. Snyder. Hierarchical image caching for accelerated walkthroughs of complex environments. In *SIGGRAPH 96 Conference Proceedings*, 1996.

[44] A. Sheffer. Model simplification for meshing using face clustering, 2000.

[45] F. Sillion, G. Drettakis, and B. Bodelet. Efficient impostor manipulation for real-time visualization of urban scenery. *Computer Graphics Forum (Proc. of Eurographics)*, 16(3), 1997.

[46] J. Torborg and J. Kajiya. Talisman: Commodity real-time 3d graphics for the pc. In *Proc. SIGGRAPH*, Computer Graphics, Annual Conference Series, 1996.

[47] M. Wimmer, P. Wonka, and F. Sillion. Point-based impostors for real-time visualization. In *Eurographics Workshop on Rendering*, 2001.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399