



**CS 563 Advanced Topics in
Computer Graphics
*Billboard Clouds***

by Jared Krechko



Presentation Overview

- Overview of Billboard Clouds
 - Big picture
 - Videos
- How to Compute Billboard Clouds
 - Lots of formulas and algorithms
- Optimizing Cloud Selection
- Developments Since the Article



What is a billboard cloud?


- Extreme simplification of object
- Project object surfaces onto plane
- Arrange planes so the object looks like the original rendered object
- Can track surface normals so relighting is possible



Why use billboard clouds?

- Simplified geometry
- Improve rendering time
- More detailed than other simplification methods like impostors or billboards
- Series of textures (texture operations)

- [Eiffel Tower](#)
- [Overview of how to obtain a billboard cloud simplification](#)



Steps to Create Billboard Clouds

- Obtain density of planes
 - Coverage, validity and penalty
- Select appropriate planes based on density
 - Bins to select highest density
- Compute textures



Selecting appropriate planes

- Define density
 - $d(P) = C(P) - \text{Penalty}(P)$
- $C(P)$ is coverage, or the amount of faces for which P is valid
- $\text{Penalty}(P)$ is used to prevent undesirable planes, i.e. locally optimal but globally costly



Plane Validity

- A plane is valid if $||vp|| < ?$
 - v is a point on the object
 - p is a point on the plane
 - $?$ is a defined error margin
- Validity domain of a point – set of planes that can represent a point
- Valid for polygon – plane is valid if all vertices of polygon are valid
- Validity domain of a face – intersection of validity domains for its vertices
- Goal: Find minimal set of planes $\{P_i\}$ so for each face, there is one i such that the face is valid on a plane



Plane Coverage

- Size of a validity set – how many primitives are valid on the plane
- Makes many small surfaces count more
- Add projected area on the plane
- Finally, $C(P) = \text{Sum}(\text{projected area on the plane}(\text{each face being projected}))$

Penalty

- Goal: Prevent planes that miss nearby faces
- Small faces hard to simplify, so more computation

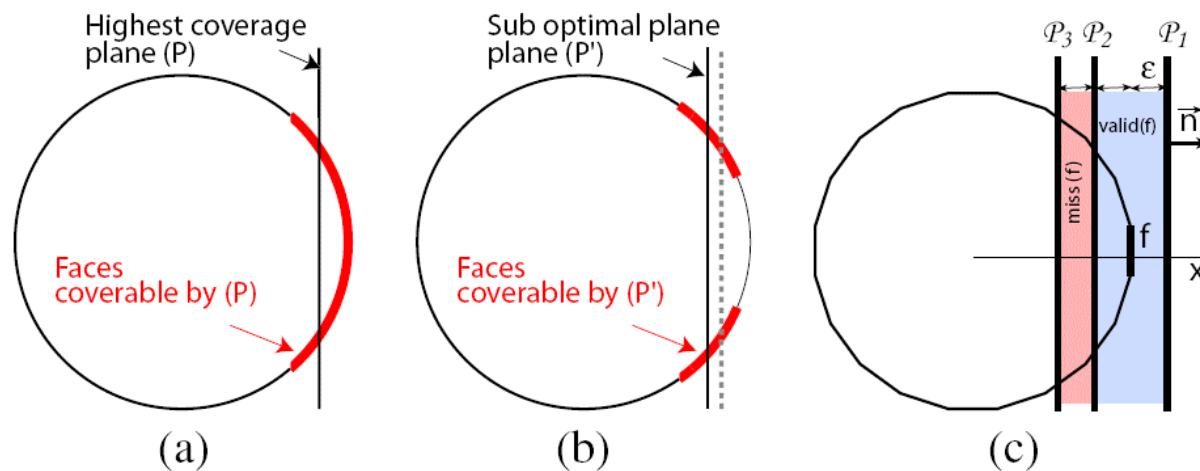


Figure 2: Near-tangency and penalty due to missed primitives.



Penalty

- Favor planes that are near tangent
- Each face can add penalty to planes that miss them
- A plane misses a face if
 - The plane is not valid for the face
 - There exists another plane within the given error ? when translated along the normal that is valid
- $\text{Penalty}(P) = W * \text{Sum}(\text{ area of faces that are in the miss set})$
 - W is a penalty weight (10 usually), used to strongly prevent using planes that closely miss primitives

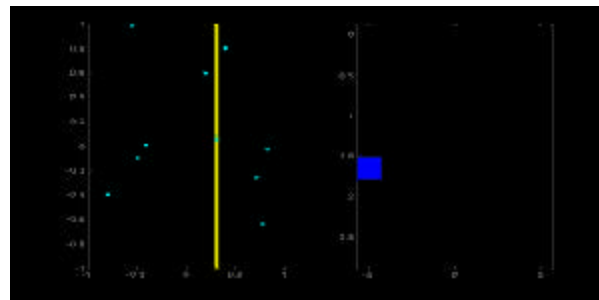


Discretizing Plane Space

- Estimate density in discretized plane space
- Parameterize planes using spherical coordinates (θ, ϕ) for normal direction, and r for distance to origin
- Set of planes going through a point defines a sheet $r = f(\theta, \phi)$
- 3D equivalent of Hough Transform

Discretizing Plane Space

- Hough Transform
 - Method to find straight lines in large amounts of data
- Hough Space
 - Each point (d, T) is a line at angle T a distance of d from the origin in the data space
- Increment T a set amount and compute d
- Value of function in Hough space gives density
 - High value corresponds to a line that goes through a lot of points



Discretizing Plane Space

- We do Hough Transform in 3D

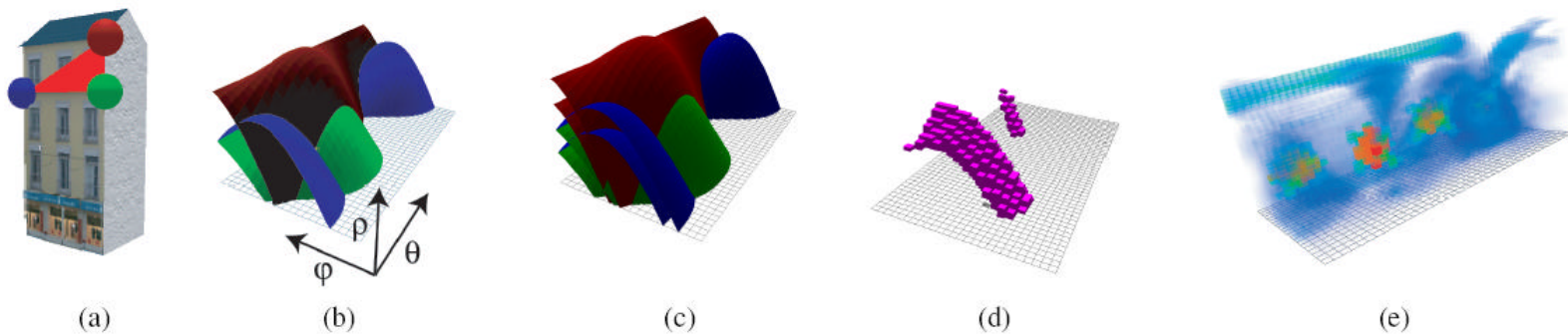


Figure 3: Density in plane space. (a) Scene with face f and its three vertices highlighted. (b) Set of planes going through each vertex, represented in plane space. The plane of f corresponds to the intersection of the three sheets. (c) Validity domain of each vertex. (d) Discretized validity domain 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.



Discretizing Plane Space

- Divide parameter space into bins
- Compute density for each bin
- Consider all bins that intersect a face valid for that face to reduce artifacts
- Density of bin – sum coverage and penalty values for each face valid for the bin

Greedy Optimization

- Pick bin with highest density
- Search for a plane in the bin that can eliminate the set of valid faces
- Update density due to removed faces
- Go to next highest density bin
- Then, compute textures for the selected planes

```
Greedy (input model, threshold  $\epsilon$ )
  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  $valid_{\epsilon}^{\mathcal{F}}(\mathcal{B})$ 
     $\mathcal{P}_i = \mathbf{RefineBin}(\mathcal{B}, valid_{\epsilon}^{\mathcal{F}}(\mathcal{B}))$ 
    UpdateDensity( $valid_{\epsilon}^{\mathcal{F}}(\mathcal{P}_i)$ )
     $\mathcal{F} = \mathcal{F} \setminus valid_{\epsilon}^{\mathcal{F}}(\mathcal{P}_i)$ 
     $\mathcal{BC} = \mathcal{BC} \cup \mathcal{P}_i$ 
  Compute textures( $\mathcal{BC}$ )
```

Figure 5: Pseudocode of the greedy selection of planes.

Refine Bin

- Used simple validity to compute density (any bin that intersects a face is valid for that face)
- Not always most dense
- Use algorithm to get most dense plane

```
RefineBin (bin  $\mathcal{B}$ , set of faces  $\mathcal{F}$ )
  plane  $\mathcal{P}$  = center of  $\mathcal{B}$ 
  if ( $valid_{\epsilon}^{\mathcal{F}}(\mathcal{P}) == valid_{\epsilon}^{\mathcal{F}}(\mathcal{B})$ )
    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 ( $\mathcal{B}_{max}$ ,  $valid_{\epsilon}^{\mathcal{F}}(\mathcal{B}_{max})$ )
```

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



Compute Textures

- Each plane assigned a set of faces during greedy algorithm
- Find bounding rectangle, then orthographically project onto plane
- Use black background for texture
- Compute normal map at same time to relight
- If a face belongs to a valid set of multiple planes, render it on all planes to reduce cracks between billboards



Optimize Texture Usage

- Restrict billboards that have mostly empty spaces by introducing a compact set
- Restrict in greedy algorithm to compact subsets of bins for each face
- Break validity set for each bin for each face into clusters
- Pick the most dense cluster

Effect of Error Bound

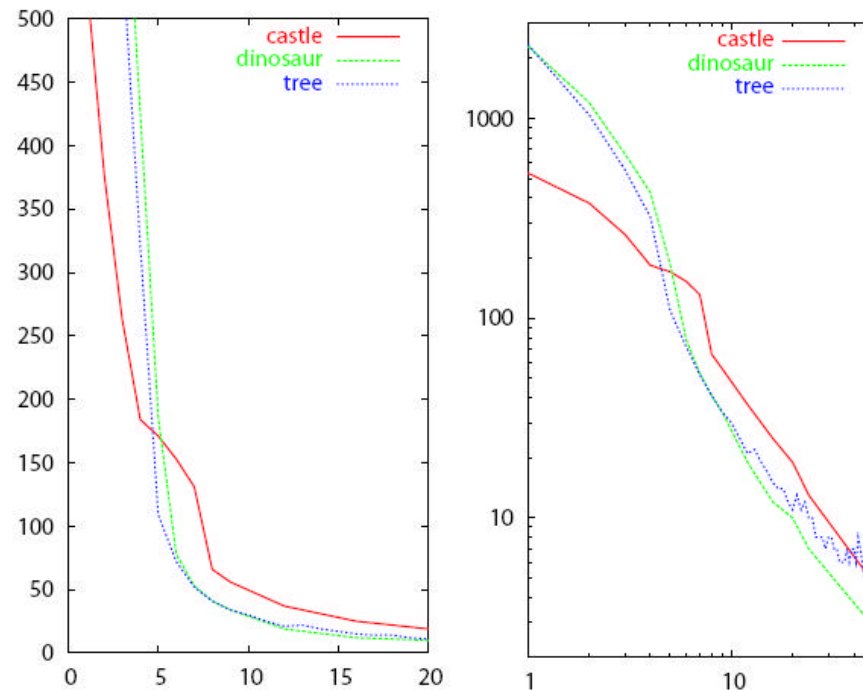


Figure 9: Cost/error curves for the castle, dinosaur and a tree. The x axis is the value of ϵ (percentage of the bounding box) and the y axis is the number of billboards. The leftmost curves are in linear units, while the rightmost curve is in log-log units.

Examples

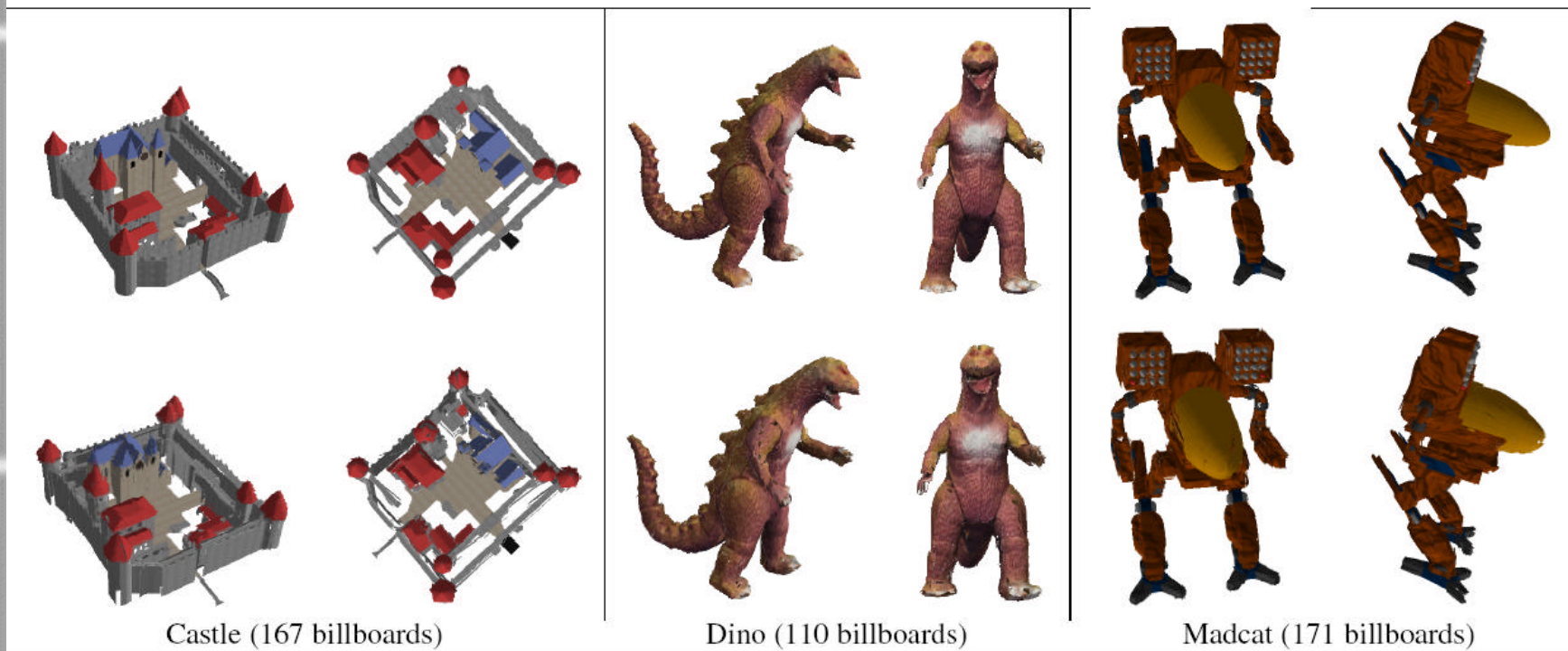
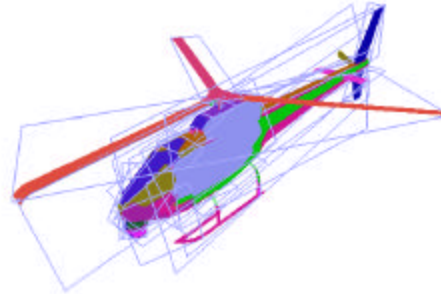


Figure 8: Results for multi-textured, possibly non-manifold models. Top row: polygonal model. Bottom row: billboard clouds.

Examples



Examples

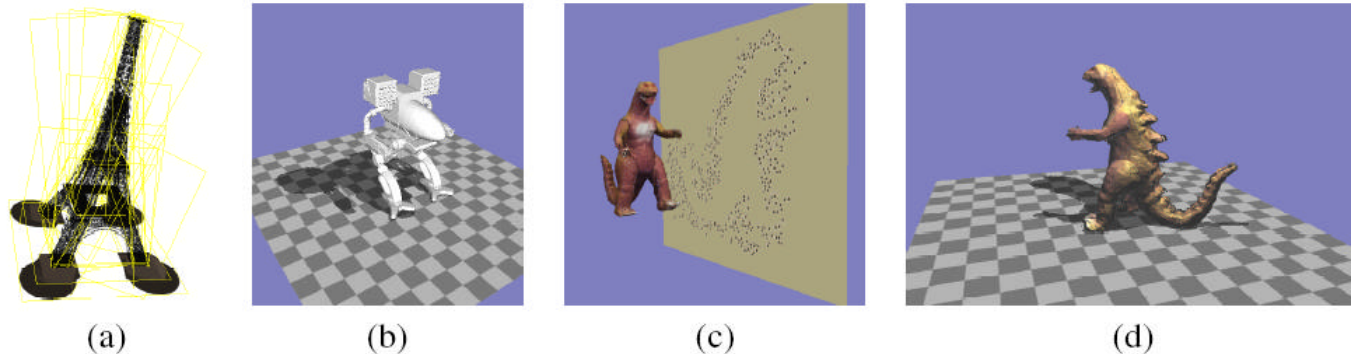
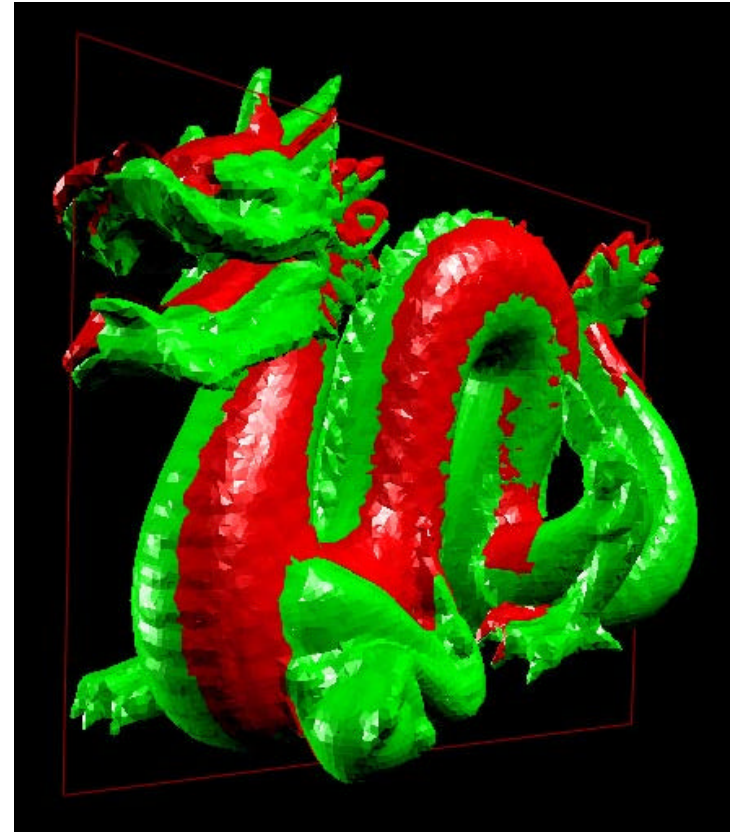
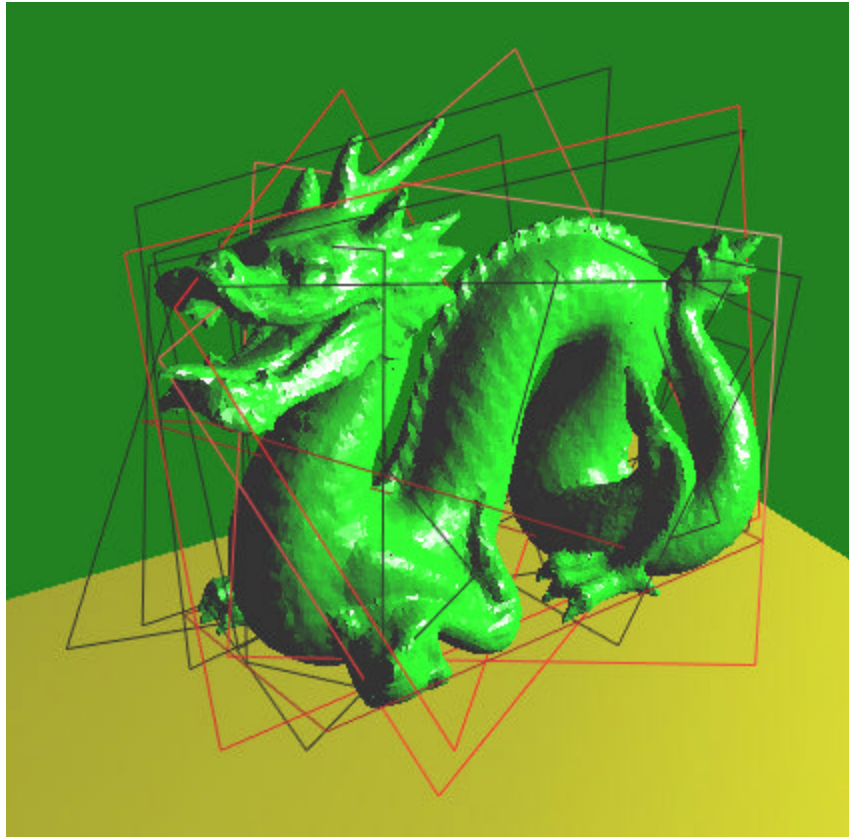


Figure 2: Billboard clouds encode an object shape and appearance through a few quads and transparent textures. This representation can be used for fast rendering of complex objects (a), for shadow computations (b) or ray-object intersection (c). Storing additional information such as normal map in the textures also allow real-time relighting for improved realism (d).

Examples



http://www.cs.wpi.edu/~emmanuel/MQPs/extreme_simplification/data.htm



Conclusions

- A good way to take the complicated geometry out of models
- Still further areas to develop but looks promising
- Widely used soon?



Beyond Billboard Clouds

- Hybrid Billboard Clouds
 - Presented at SIGGRAPH 2004
 - Use part mesh, part billboard to simplify models
 - Combine strengths of image-based rendering and full model rendering
 - Basically compute a cost for billboard and geometric model and pick the lesser of the two

BTF Textured Billboard Clouds

- Bidirectional Textured Function Billboard Clouds
 - Recover self-shadowing, reflectance properties and changing silhouettes
 - Memory efficient storage of such properties to make rendering “quick”
 - Uses less resources but still uses a lot



Figure 2: Comparison of standard BC with texture and normal map (left) with BTF textured one (middle) and original model (right). The resolution of the texture and BTF are optimized for distant viewing (small images).



Improved Methods

- Improved algorithms
 - K-means clustering
 - Set number of billboards so can vary complexity
 - Reducing cracks in images
 - Compute triangles on multiple planes



Future Methods

- Improve greedy algorithm
- Apply more texture operations on the billboards themselves
- Other improvements?

References

- Bromberg-Martin, Ethan et. al. "Hybrid Billboard Clouds for Model Simplification." Brown University. <http://graphics.cs.brown.edu/games/HybridBillboardClouds/bromberg-jonsson-marai-mcguire-2004-hybrid.pdf>
- Décoret, Xavier et. al. "Billboard Clouds for Extreme Model Simplification." <http://w3imagis.imag.fr/Publications/2003/DDSD03/bc03.pdf>
- Décoret, Xavier et. al. "Billboard Clouds for Extreme Model Simplification." MIT Laboratory for Computer Science, March 2003. <http://www.csail.mit.edu/research/abstracts/abstracts03/interfaces-applications/10decoret.pdf>
- Huang, I.-T., et. al. "Improved Billboard Clouds for Extreme Model Simplification." University of Auckland. http://www.cs.auckland.ac.nz/~burkhard/Publications/IVCNZ04_HuangNovinsWuensche.pdf
- Lehtinen, Tuomo. "Billboard Clouds." Helsinki University of Technology, 2004. http://www.tml.hut.fi/Opinnot/Tik-111.500/2004/papers_final/Lehtinen_final.pdf
- Meseth, Jan and Reinhard Klein. "Memory Efficient Billboard Clouds for BTF Textured Objects." University of Bonn. <http://cg.cs.uni-bonn.de/docs/publications/2004/meseth-2004-memory.pdf>
- Moreno-Fortuny, Gabriel. "Billboard Clouds." University of Waterloo. <http://www.cgl.uwaterloo.ca/~gmoreno/bclouds.html>
- Reynolds, John et. al. "Extreme Graphical Simplification." Worcester Polytechnic Institute. April 28, 2004. http://www.cs.wpi.edu/~emmanuel/MQPs/extreme_simplification/Documents/EGSReport.pdf
- Reynolds, John et. al. "Extreme Graphical Simplification – Project Data." Worcester Polytechnic Institute. April 28, 2004. http://www.cs.wpi.edu/~emmanuel/MQPs/extreme_simplification/data.htm
- Storkey, Amos. "Hough Transform." Institute for Adaptive and Neural Computation. <http://www.anc.ed.ac.uk/~amos/hough.html>