Point-Based Level-of-Detail with Object Textures

André Maximo[*] Ricardo Marroquim[†] Claudio Esperança[‡]
Universidade Federal do Rio de Janeiro

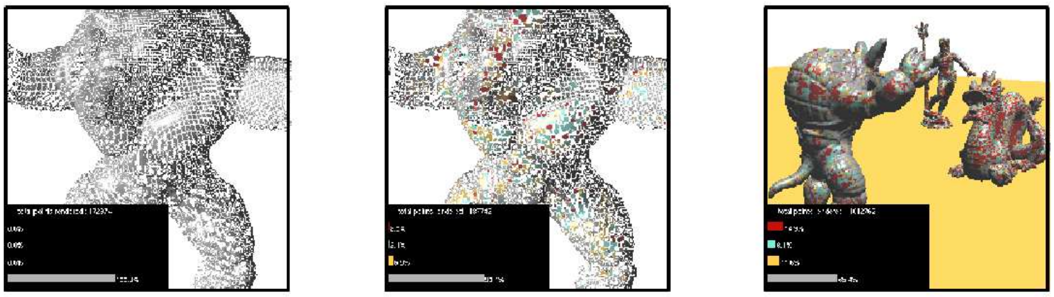

Figure 1: *Left: Armadillo with no Level-of-Details and reduced surfel radii. Middle: Different resolution levels according to the perpendicular error estimation, where the colors from coarsest to finer are: Red, Blue, Yellow and Gray. Right: Three different models with Level-of-Details.*

1 Introduction

We introduce Object Textures, a technique for mapping complex objects onto small texture patches which are dynamically retrieved during rendering. Each patch contains a subset of the object's primitives which are read from GPU texture by a geometry shader. As a proof of concept, we demonstrate the method with a Level-of-Detail application for Point-Based Rendering. Only the coarsest resolution level is sent to the GPU while the more refined levels are recovered from an Object Texture.

2 Exposition

An Object Texture stores the object's primitives grouped in patches, where each patch is associated with a single primitive that is actually sent to the GPU. The number of primitives per patch is arbitrary and limited only by the graphics card specifications. To allow the entire object to be retrieved during rendering, each patch contains information about where its first primitive is stored and how many primitives it contains.

We illustrate the use of Object Textures with a Level-of-Detail application for Point-Based Rendering. The different resolutions levels are created in a simple manner by merging adjacent samples, i.e. surfels in this case. Each sample in a coarsest resolution level is created by combining up to four adjacent samples from the immediate higher resolution level. The number of levels is limited to four in this application, but this restriction can be easily extended to more levels. The three highest resolution levels are stored in an Object Texture and ordered from coarsest to finest inside each patch. The fourth level with the coarsest resolution contains the vertices sent to GPU and used to reference the patches, thus called *patch vertices*. These vertices are coarse resolution samples containing some extra information: the texture coordinates of the first vertex of the patch and the number of vertices in each level, where the third level may contain up to four vertices, the second up to 16, and the finest level up to 64 vertices per patch.

Each merged sample stores a perpendicular error pre-computed as in [Dachsbacher et al. 2003]. This error estimates the local surface smoothness prioritizing details along the silhouette, while, at the same time, taking into account the distance to the camera position. In the geometry shader, the error of the patch vertex (coarsest resolution) is projected and compared with a threshold. If it's smaller than the threshold only the patch vertex is projected, otherwise, one of the finer levels is selected depending on the amount of error. The primitives of the chosen resolution level are fetched from the Object Texture and projected for rendering.

After all patches have been processed, the object is rendered using the algorithm proposed in [Marroquim et al. 2007]. In Figure 1 some examples are illustrated. Note how smoother regions are rendered with coarser levels while more detailed regions are rendered with the finest level.

3 Conclusion

By projecting a single primitive per patch, the use of CPU-GPU bandwidth is considerably reduced when compared with sending the complete geometry. The number of primitives sent in the Point-Based LOD application is one order of magnitude less than the model's samples, or surfels in this case.

In our experiments a nVidia GeForce 8800 GT graphics card was used. However, we noted that the performance decreases drastically with the increase of the maximum number of output primitives of the geometry shader, thus no significant performance was gained. On the other hand, we expect that next generation GPUs shall have faster and more optimized geometry shaders thus making this technique more attractive.

References

DACHSBACHER, C., VOGELGSANG, C., AND STAMMINGER, M. 2003. Sequential point trees. *ACM Trans. Graph. 22*, 3, 657–662.

MARROQUIM, R., KRAUS, M., AND CAVALCANTI, P. R. 2007. Efficient point-based rendering using image reconstruction. In *PBG '07: Proceedings of the Eurographics Symposium on Point-Based Graphics*, 101–108.

[*]e-mail: andre@lcg.ufrj.br
[†]e-mail: ricardo@lcg.ufrj.br
[‡]e-mail: esperanc@lcg.ufrj.br