Relief Clipping Planes for Real-Time Rendering

Matthias Trapp and Jürgen Döllner*
Hasso-Plattner-Institute, University of Potsdam, Germany

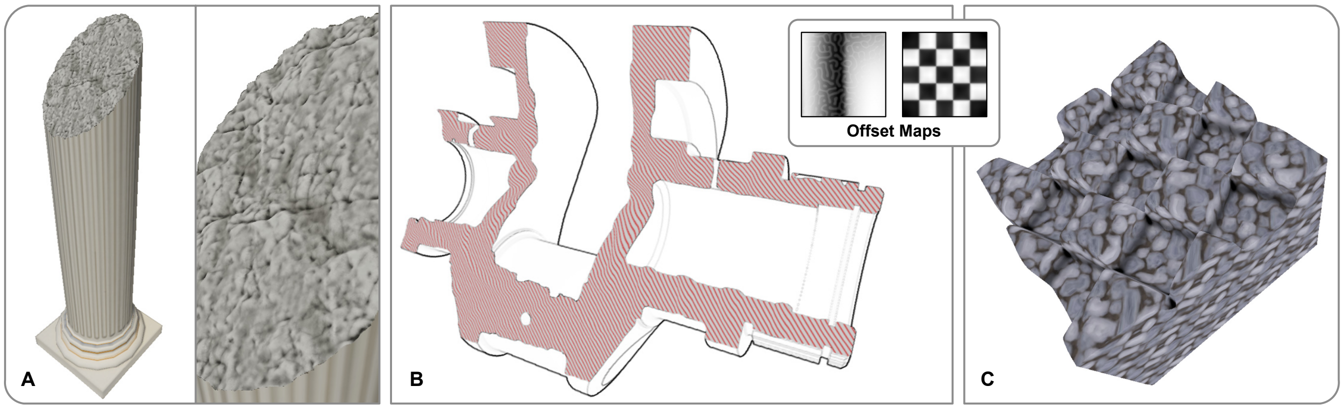

Figure 1: *Results of our rendering technique: A: A clipped and capped Roman column. The close-up shows the non-regularity and shading of the cut-surface to create the impression of a solid column. B: Application of relief clipping planes to a non-convex mesh. The crank is rendered using edge enhancement and a hatched texture is applied to the cap. C: shows a clipped, capped, and solid textured convex object.*

Introduction

The concept of clipping planes is well known in computer graphics and can be used to create cut-away views. But clipping against just analytical defined planes is not always suitable for communicating every aspect of such a visualization. For example, in hand-drawn technical illustrations, artists tend to communicate the difference between a cut and a model feature by using non-regular, sketchy cut lines instead of straight ones.
To enable this functionality in computer graphics (Figure 1), we present a technique for rendering *relief clip planes* (RCP) in real-time. Therefore, we extend the clip plane equation with an additional *offset map* (OM), which can be represented by a texture map that contains height values. Clipping is then performed by varying the clip plane equation with respect to such an offset map. Further, we propose a capping technique that enables the rendering of caps onto the clipped area to convey the impression of solid material. It avoids a re-meshing of a solid polygonal mesh after clipping is performed. Our approach is pixel precise, applicable in real-time, and takes fully advantage of graphics accelerators.

Relief Clipping Planes

Briefly, a $RCP = (N, P, OM, s)$ is defined by a normal vector $N = (A, B, C)$ and origin P, which are required to construct the respective normal form, an offset map OM, and a height value scaling factor s. Given an arbitrary shaped solid mesh and a RCP, clipping is performed on fragment level as follows. For each fragment with the clip space coordinate $C = (x, y, z)$ the function:

$$clip(RCP, C) = xA + yB + zC - f(C, P, OM) > 0$$

is evaluated using a fragment shader program. Therefore, f delivers a scalar $D \in \mathbb{R}$ by first generating texture coordinates into the offset map, then samples OM, and finally scales the resulting height sample by s. If the above equation is satisfied, the fragment program discards the tested fragment. This step can be performed for a number of clipping planes within a single rendering pass.

*{matthias.trapp, juergen.doellner}@hpi.uni-potsdam.de

Capping Solid Meshes

Due to the possibly non-regularity of the clip surface, capping techniques based on stencil buffer capabilities [Blythe et al. 1999] cannot be applied. Especially for non-convex shapes, the association of a cap to a clipped area cannot be made definitely in image space using stencil masks.
Our image-based approach works for every clipped arbitrary solid. Therefore, a *volumetric depth sprite* [Trapp and Doellner 2008] of the polygonal mesh is created in a preprocessing step. Following to that, two steps are performed per frame: First, the solid mesh is rendered into the frame buffer with applied relief clipping. Second, the capping meshes is rendered using per-vertex displacement mapping. In this step, a *volumetric depth test* is performed per fragment that determines if it lies inside the volume and thus associated with a gap, or if it located outside the volume and therefore is discarded. GPU based-mesh refinement [Boubekeur and Schlick 2005] is applied to fit the subdivision of the cap mesh to the resolution of the offset map.

Conclusions & Future Work

We presented a new rendering technique for performing clipping and capping of arbitrary solid meshes against relief clip planes in real-time. For future work, we adapt this technique for apply capping for clipping against volumes [Trapp and Doellner 2008]. Further, we want to replace displacement mapping with parallax mapping to increase performance.

References

BLYTHE, D., MCREYNOLD, T., B. GRANTHAM, B., KILGARD, M., AND SCOTT, R. 1999. Programming with OpenGL: Advanced Rendering. In *SIGGRAPH Course*, New York: ACM, A. Rockwood, Ed.

BOUBEKEUR, T., AND SCHLICK, C. 2005. Generic Mesh Refinement On GPU. In *Proceedings of ACM SIGGRAPH/Eurographics Graphics Hardware*.

TRAPP, M., AND DOELLNER, J. 2008. Real-Time Volumetric Tests Using Layered Depth Images. In *Eurographics*, 49–52.