sketches_0104: Image-Space Construction of Displaced Normal Maps

Ivan Neulander
Rhythm and Hues Studios

Overview

We present a hybrid 2D/3D algorithm for effeciently deriving a normal map of displacement-mapped polygonal geometry. Unlike the traditional bump-mapping approach by [Blinn 1978], ours is based on a physical representation of the true displaced mesh. It is therefore more accurate and less restrictive, easily handling a variety of procedural displacements. What makes our approach unique is that it does not entail finely tessellating the polygonal mesh or dicing it into REYES micropolygons.

Implementation

Our geometry consists of a set of triangles with predefined texture coordinates suitable for normal mapping (C1 continuous except across a few well-chosen seams). We proceed as follows:

(1) We rasterize the geometry into the normal map's texture space at a desired resolution. At each pixel, we store the original and displaced position, and the undisplaced shading normal. Using a software rasterizer, this is easily computed in a single pass. Hardware rasterization generally requires multiple passes. Depth-buffering and edge antialiasing are not needed, and a single shading sample per pixel is usually sufficient.

(2) We compute a *geometric displaced normal* map from the displaced position map, as described in the following section. Because these normals are not suitable for shading, we proceed with the next two steps.

(3) We compute a *geometric normal* map from the undisplaced position map we derived in (1). This operation is the same as (2), but operates on different input. Thus, (2) and (3) are good candidates for multiprocessing.

(4) We modify the normal map computed in (1) by adding to it the difference between the geometric displaced normal and the geometric original normal, i.e. (2)-(3). After being renormalized, this yields the *displaced normal* map, which is what we want.

Texture-Space Normal Computation

To build the normal maps in (2) and (3), we exploit the coherence of the point clouds represented by our position maps. Each cluster of 3×3 pixels can be viewed as a triangle fan within a larger mesh. To determine the normal of the center pixel, N_c, we compute the usual area-weighted sum of these triangles' normals, as shown in Figure 1. Alternative triangle constructions could be used to more robustly treat edge pixels, but we find our approach adequate.

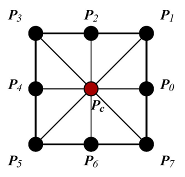

Figure 1: $\mathbf{N}_c = \Sigma_{i=0}^{8}[M_i M_{i+1}(\mathbf{P}_i - \mathbf{P}_c) \times (\mathbf{P}_{i+1} - \mathbf{P}_c)]$
$M_i = \{1$ if pixel $\mathbf{P}_i$ is covered, 0 otherwise $\}$. Indices are mod 8.

Complications

Depending on the mesh granularity, map resolution, and displacement function, there may be noticeable faceting artifacts in the displaced normal map, despite steps (3) and (4). One solution is to refine the mesh using (for instance) Loop's subdivision. Though expensive, this tessellation can be applied sparingly since pixel-sized triangles are not needed. A faster but less accurate solution is to smooth the mesh by blurring the position map. This preserves displacement detail since only the pre-displaced position is blurred. As future work, we are researching a faster triangle smoothing scheme based on [Smits et al. 2000], which does not require neighbor information.

Results

Our multithreaded software implementation, running on a dual Athlon 2133MHz, generated the 1024x1024 normal map shown below in 1.8 seconds, based on a procedurally displaced 38k triangle model with one shading sample per pixel. No tessellation or positional blur was used. The displacement function (which would be difficult to capture with traditional bump mapping) was:

$$\begin{aligned}
\mathbf{P}_x &\mathrel{+}= [2\sin(17\mathbf{N}_x) + \sin(1\mathbf{P}_x) - \mathbf{N}_x \sin(600uv)]/80 \\
\mathbf{P}_y &\mathrel{+}= [2\sin(23\mathbf{N}_y) + \sin(2\mathbf{P}_y) - \mathbf{N}_y \sin(600uv)]/80 \\
\mathbf{P}_z &\mathrel{+}= [2\sin(31\mathbf{N}_z) + \sin(3\mathbf{P}_z) - \mathbf{N}_z \sin(600uv)]/80
\end{aligned}$$

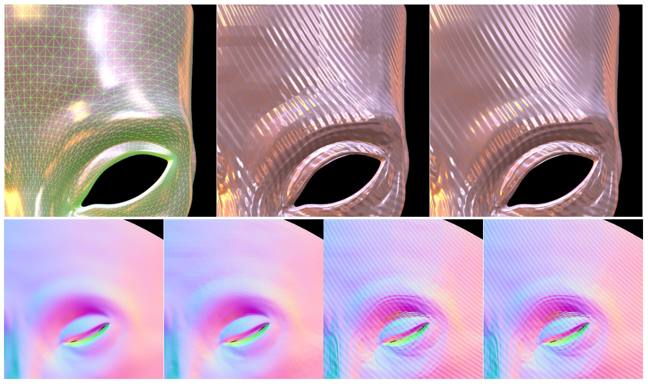

Figure 2: Top: renderings using original mesh, geometric displaced normal, displaced normal. Bottom: maps of normal, geometric normal, geometric displaced normal, displaced normal.

References

Blinn, J. F. 1978. Simulation of wrinkled surfaces. In *SIGGRAPH '78*, ACM Press, New York, NY, USA, 286–292.

Smits, B. E., Shirley, P., and Stark, M. M. 2000. Direct ray tracing of displacement mapped triangles. In *Eurographics 2000*, Springer-Verlag, London, UK, 307–318.