Contact
Bert Freudenberg
Institut für Simulation
und Graphik
Otto-von-Guericke-Universität
Magdeburg
Universitätsplatz 2, 39106
Magdeburg, Germany
bert@isg.cs.uni-magdeburg.de

Shading in line drawings is expressed by varying the stroke width or density of strokes covering an object's surface. In their pioneering work, Winkenbach and Salesin introduced the concept of "prioritized stroke textures" to the emerging field of non-photorealistic rendering.[1] Despite the advances in processing power in the years since then, the sheer number of lines to draw prevents this method from running in real time. A real-time approach for hatching was presented by Lake et al,[2] which chooses from a set of textures based on the brightness at vertices, subdividing polygons if necessary. However, the method is very CPU-heavy and requires many polygons.

Our new technique uses per-pixel shading graphics hardware to implement non-photorealistic shading. The texture-combining facilities accessible via OpenGL on NVIDIA GeForce and ATI Radeon cards provide the flexibility necessary to vary the line width or number of strokes per area.

To indicate shading by variable-width hatching, a 3D halftone pattern is created as texture T and compared at every pixel with the target intensity I, creating black or white pixels.[3] Halftoning, however, yields unsatisfying results for interactive applications because of the computer screen's limited resolution. To smooth the harsh transition from white to black we instead take the inverted sum of T and I and scale it by some constant $c > 1$. After clamping the result to the range (0…1)we get a mostly black and white output while still preserving a few gray levels (see Figure 1).

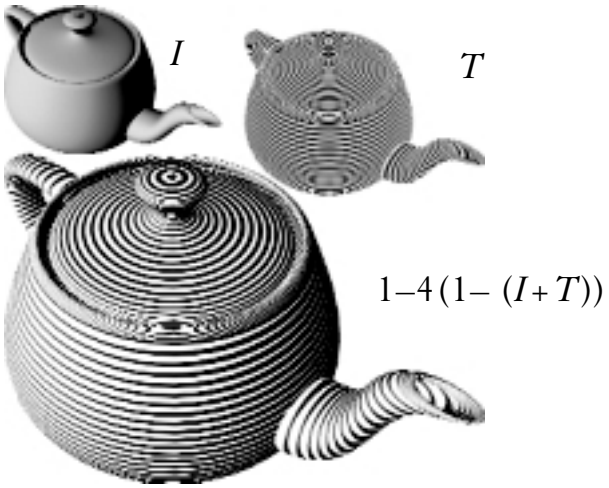

Figure 1. Lighting-dependend stroke width.

We follow a similar approach with stroke textures. Strokes are drawn on different layers. To facilitate one-pass rendering, all stroke layers are composited into the texture T in a pre-processing step, using a different gray level for each layer (Figure 2). At runtime, all layers needed to visually approximate the intensity I are selected for drawing, employing the same combiner operations as introduced above (see Figure 3).

Figure 2. Stroke map construction.

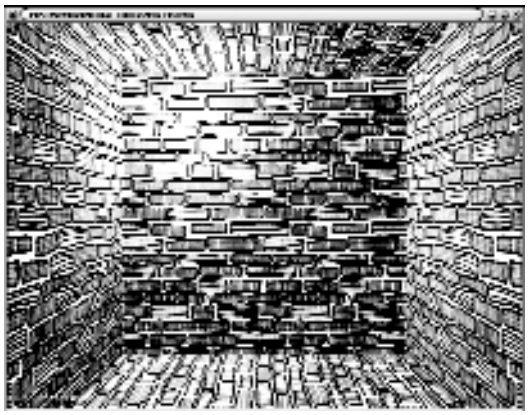

Figure 3. Real-time stroke textures.

There are many ways to extend the basic idea of stroke map shading. Lightmaps can be used to vary the amount of detail drawn on a surface. With bump maps, individual strokes can be made sensitive to lighting. Shadows can be rendered in additional passes. We are working on integrating all these into our interactive line drawings.

Our method does not require any additional CPU work at runtime. The configurability or even programmability of recent graphics hardware is a very powerful device to achieve non-standard looks. We are looking forward to seeing more visually rich interactive rendering styles emerge in the future.

References

1. Winkenbach, G. & Salesin, D.H. (1994). Computer-generated pen-and-ink illustration. In *Proceedings. SIGGRAPH 94*, 91-100.
2. Lake, A., Marshall, C., Harris, M., & Blackstein, M. (2000). Stylized rendering techniques for scalable real-time 3D animation. In *Proceedings NPAR 2000*, 13-20.
3. Haeberli, P. & Segal, M. (1993). Texture mapping as a fundamental drawing primitive. In *Fourth Eurographics Workshop on Rendering*, 259-266.