Real-Time Stroke Textures

Bert Freudenberg

Institut für Simulation und Graphik
Universität Magdeburg
Real-Time Stroke Textures

Overview

- Pen-and-ink style in CG
- Texture based real-time approaches
- Basic stroke-map technique
- Extensions
- Conclusion
Pen-and-ink style in CG

Winkenbach and Salesin, 1994
Real-Time Techniques

Line based
- Lines drawn individually
- To slow for shading
- Outline only

Texture based
- Multiple lines drawn at once
- Suited for shading
Texture based approaches

Lake et al., 2000

- 1 texture per triangle
- Lit and split by CPU
- Flat shading
Texture based approaches

Praun et al., 2001 (presented tomorrow)

- 2 textures per vertex = 6 per face
- Lit by Vertex Program
- Blended by GPU
- Gouraud shading
Texture based approaches

Freudenberg, 2001 (presented now)

- Multiple layers per pixel
- Lit completely by GPU
- Per-pixel shading
Varying Line-Width Shading

Idea

- Create half-toning pattern $T$
- Per-pixel compare to target intensity $I$
- Output black or white pixels
Varying Line-Width Shading

**Idea**
- Create half-toning pattern $T$
- Per-pixel compare to target intensity $I$
- Output black or white pixels

**Problem**
- Aliasing

**Solution**
- Scaling instead of thresholding
Varying Line-Width Shading

Scaling

\[
\begin{align*}
& T \\
& I \\
& T + I \\
& 1 - (T + I) \\
& 4 (1 - (T + I)) \\
& 1 - 4 (1 - (T + I))
\end{align*}
\]

Anti-aliased result
Varying Line-Width Shading
Stroke Maps

Idea

- Strokes are drawn in layers
- Encoded into one texture
- Expanded at run-time
- Selected by reference intensity
Stroke Maps

Layering of Strokes
Stroke Maps

Encoding

- Pre-processing step
- Encode layers as gray
  - 1\textsuperscript{st} layer black
  - 2\textsuperscript{nd} layer 66\% gray
  - 3\textsuperscript{rd} layer 33\% gray
- Paint last-to-first into texture = Stroke Map
Stroke Maps

Expansion

- At run-time
- Using per-pixel operations
- EXACT same formula as for line-width variation
  
  - $1 - 4 \left( 1 - (T + I) \right)$
  
  - General combiner:
    $$r0 = \text{scale\_by\_4}(\text{sum}(\text{invert}(t), \text{negate}(i)))$$
  
  - Final combiner:
    $$\text{out} = \text{invert}(r0)$$
Stroke Maps

$I$  $T$  sum  scaled
Stroke Maps

NVPARSE code

```c
{ rgb {
    discard = unsigned_invert(tex0); // 1-T
    discard = -col0; // -I
    spare0 = sum(); // 1-T-I
    scale_by_four(); // 4(1-T-I)
}
out.rgb = unsigned_invert(spare0); // 1-4(1-(T+I))
```
Extensions

Indication Mapping

- Enabled by per-pixel evaluation
- Bias intensity by indication
- Needs one additional combiner stage
Extensions

Shadows

- Combinable with most shadow algorithms
- Adds greatly to realism
Conclusion

Shortcomings

- Limited accuracy
- Layers not strictly separated
- Only one-pass shading supported
  - Multiple passes via render-to-texture
Conclusion

Advantages

- Cheap:
  - no CPU effort
  - one texture unit
  - one register combiner
  - one pass
- Even works on “old” GeForce
- Well suited for highly interactive environments
Real-Time Stroke Textures

Questions?