

*S A N A N T O N I O*

# SIGGRAPH

÷ 2002 ÷



# Real-Time Image-Space Outlining for Non-Photorealistic Rendering

**Jason L. Mitchell**

**ATI Research**

**[JasonM@ati.com](mailto:JasonM@ati.com)**

**Chris Brennan**

**ATI Research**

**[CBrennan@ati.com](mailto:CBrennan@ati.com)**

**Drew Card**

**ATI Research**

**[DCard@ati.com](mailto:DCard@ati.com)**

# Outline

- **Image-Space Outlining**
  - Silhouettes and Crease Outlines
  - Shadow Outlines
  - Texture Boundary Outlines
- **Stylization with Morphology**
- **Antialiasing**
- **Optimizations**
  - Alpha testing
  - Multiple Render Targets



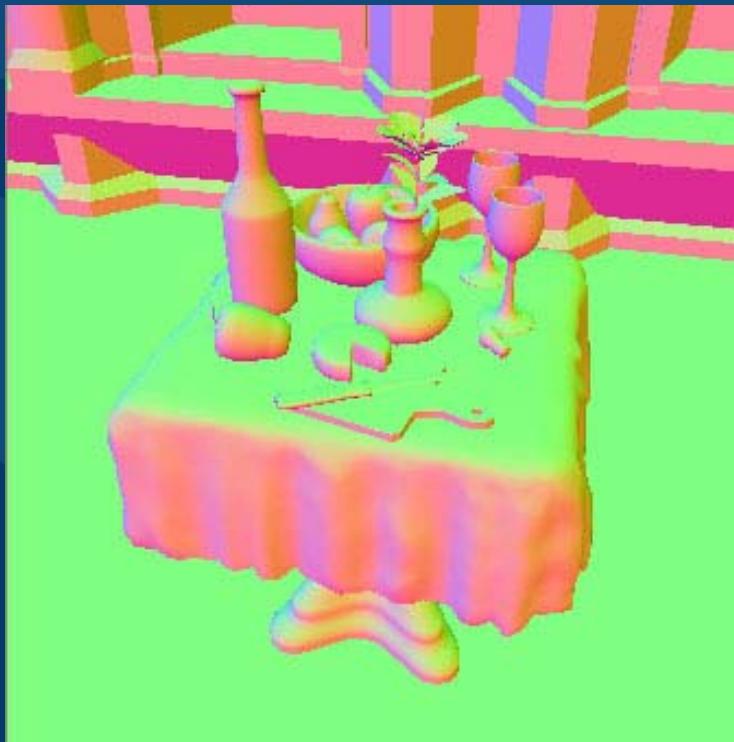
# Outlining Strategy

- Render alternate representation of scene into texture map
  - With the RADEON 9700, we're able to render into up to four targets simultaneously, effectively implementing Saito and Takahashi's g-buffer
- Run filter over image to detect edges
  - Implemented using pixel shading hardware



# Normal and Depth

SAN ANTONIO  
**SIGGRAPH**  
2002

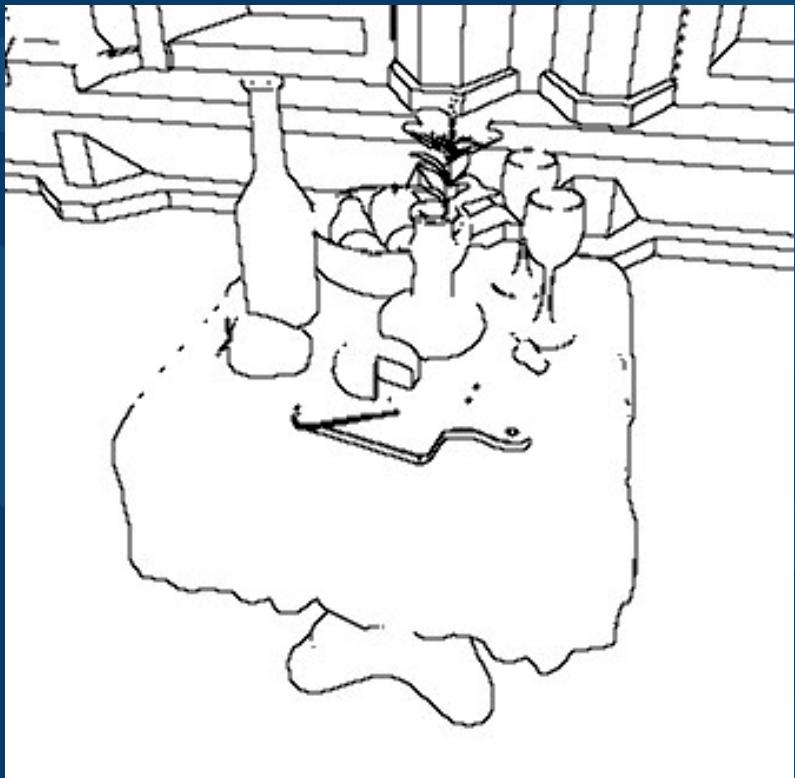


**World Space Normal**



**Eye Space Depth**

# Outlines



Normal Edges

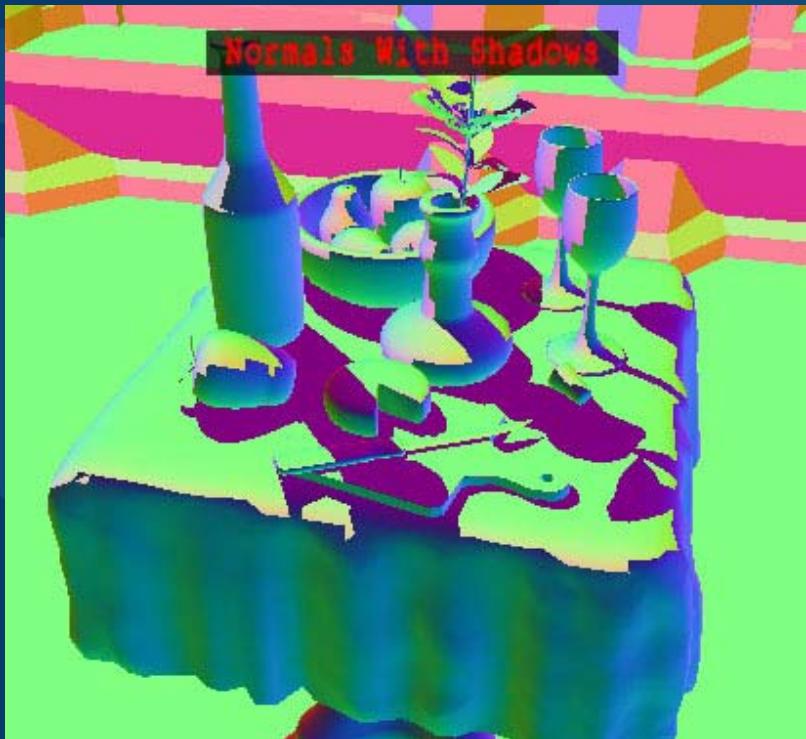


Depth Edges

SAN ANTONIO  
**SIGGRAPH**  
2002

# Normal and Depth Negated in Shadow

SAN ANTONIO  
**SIGGRAPH**  
2002



**World Space Normal  
Negated in Shadow**



**Eye Space Depth  
Negated in Shadow**

# Normal and Depth Outlines

SAN ANTONIO  
**SIGGRAPH**  
2002



Edges from Normals



Edges from Depth



# Object and Shadow Outlines

SAN ANTONIO SIGGRAPH  
2002



**Outlines from selectively  
negated normals and depths**

Real-Time Image-Space Outlining for Non-Photorealistic Rendering



# Texture Region IDs

SAN ANTONIO  
**SIGGRAPH**  
2002

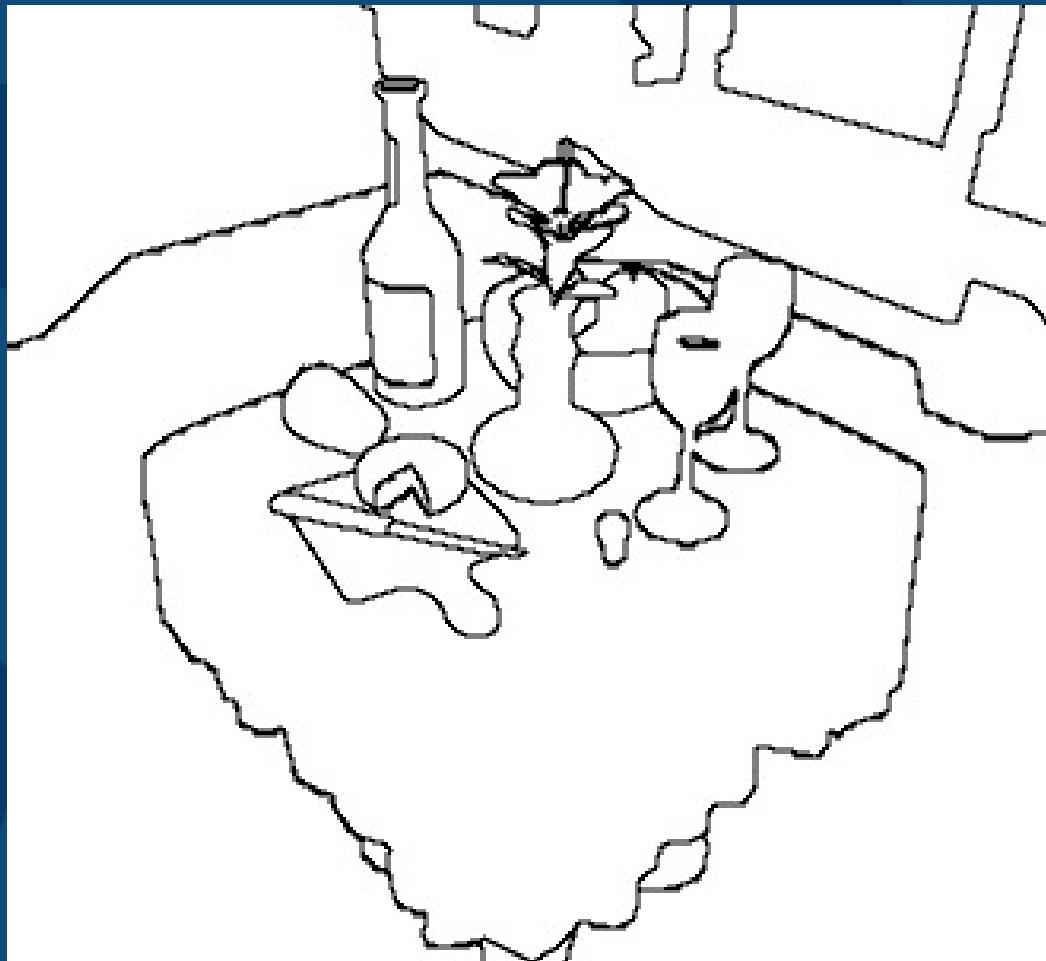


Real-Time Image-Space Outlining for Non-Photorealistic Rendering



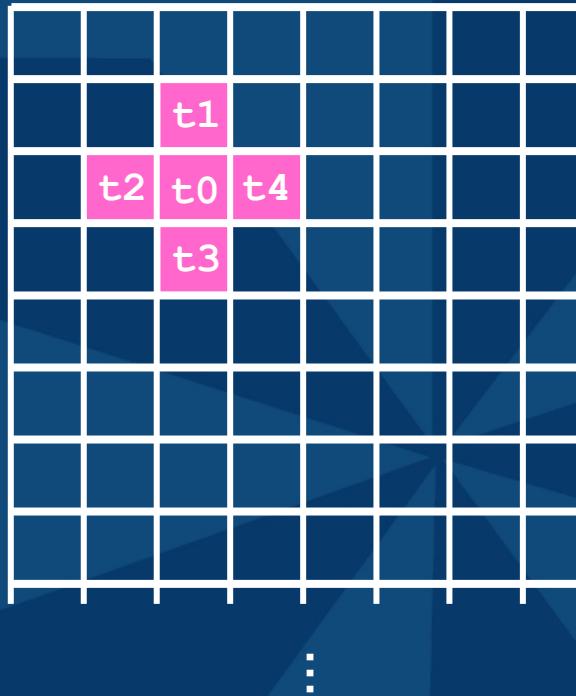
# Edges at Texture Region Boundaries

SAN ANTONIO  
**SIGGRAPH**  
2002



# Edge Filter

SAN ANTONIO  
**SIGGRAPH**  
2022



5-tap Filter



# Edge Filter Code

```

ps.2.0
def c0, 0.0f, 0.80f, 0, 0           // normal thresholds
def c3, 0, .5, 1, 2
def c8, 0.0f, 0.0f, -0.01f, 0.0f    // Depth thresholds
def c9, 0.0f, -0.25f, 0.25f, 1.0f
def c12, 0.0f, 0.01f, 0.0f, 0.0f     // TexID Thresholds
dcl_2d s0
dcl_2d s1
dcl t0
dcl t1
dcl t2
dcl t3
dcl t4

// Sample the map five times
texld r0, t0, s0 // Center Tap
texld r1, t1, s0 // Down/Right
texld r2, t2, s0 // Down/Left
texld r3, t3, s0 // Up/Left
texld r4, t4, s0 // Up/Right

-----
// NORMALS
-----
mad r0.xyz, r0, c3.w, -c3.z
mad r1.xyz, r1, c3.w, -c3.z
mad r2.xyz, r2, c3.w, -c3.z
mad r3.xyz, r3, c3.w, -c3.z
mad r4.xyz, r4, c3.w, -c3.z

// Take dot products with center (Signed result -1 to 1)
dp3 r5.r, r0, r1
dp3 r5.g, r0, r2
dp3 r5.b, r0, r3
dp3 r5.a, r0, r4

// Subtract threshold
sub r5, r5, c0.g

// Make black/white based on threshold
cmp r5, r5, c1.g, c1.r

// detect any 1's
dp4 sat r11, r5, c3.z
mad_sat r11, r11, c1.b, c1.w // Scale and bias result

-----
// Z
-----
add r10.r, r0.a, -r1.a      // Take four deltas
add r10.g, r0.a, -r2.a
add r10.b, r0.a, -r3.a
add r10.a, r0.a, -r4.a

```

```

        cmp r10, r10, r10, -r10          // Take absolute value
        add r10, r10, c8.b               // Subtract threshold
        cmp r10, r10, c1.r, c1.g         // Make black/white
        dp4 sat r10, r10, c3.z          // Sum up detected pixels
        mad_sat r10, r10, c1.b, c1.w   // Scale and bias result
        mul_r11, r11, r10               // Combine with previous

-----
// TexIDs
-----
// Sample the map five times
texld r0, t0, s1 // Center Tap
texld r1, t1, s1 // Down/Right
texld r2, t2, s1 // Down/Left
texld r3, t3, s1 // Up/Left
texld r4, t4, s1 // Up/Right

// Get differences in color
sub r1.rgb, r0, r1
sub r2.rgb, r0, r2
sub r3.rgb, r0, r3
sub r4.rgb, r0, r4

// Calculate magnitude of color differences
dp3 r1.r, r1, c3.z
dp3 r1.g, r2, c3.z
dp3 r1.b, r3, c3.z
dp3 r1.a, r4, c3.z

        cmp r1, r1, r1, -r1          // Take absolute values
        sub r1, r1, c12.g            // Subtract threshold
        cmp r1, r1, c1.r, c1.g       // Make black/white
        dp4 sat r10, r1, c3.z          // Total up edges
        mad_sat r10, r10, c1.b, c1.w // Scale and bias result
        mul_r11, r10, r11             // Combine with previous

// Output
mov oC0, r11

```



# Morphology

SAN ANTONIO  
**SIGGRAPH**  
2002

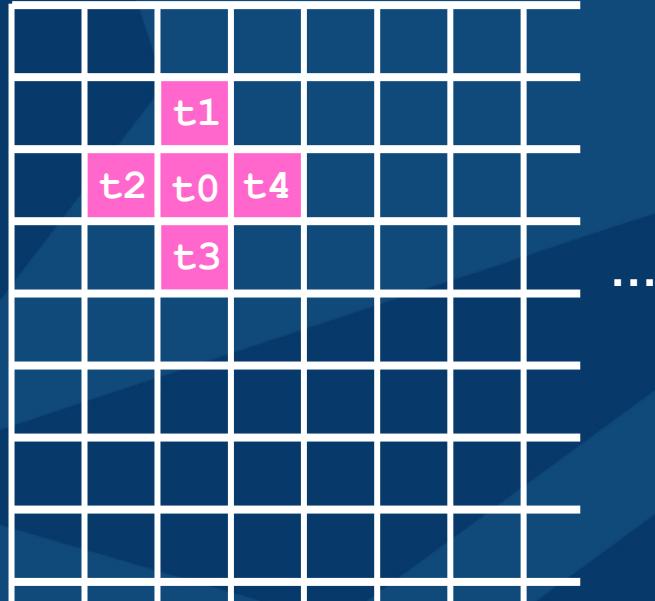


Dilate



# Dilation Shader

```
ps.2.0  
  
def c0, 0, .5, 1, 2  
def c1, 0.4f, -1, 5.0f, 0  
dcl_2d s0  
dcl t0  
dcl t1  
dcl t2  
dcl t3  
dcl t4  
  
// Sample the map five times  
texld r0, t0, s0 // Center Tap  
texld r1, t1, s0 // Up  
texld r2, t2, s0 // Left  
texld r3, t3, s0 // Down  
texld r4, t4, s0 // Right  
  
// Sum the samples  
add r0, r0, r1  
add r1, r2, r3  
add r0, r0, r1  
add r0, r0, r4  
mad_sat r0, r0.r, c1.r, c1.g // Threshold  
mov oC0, r0
```



:



# Antialiasing

SAN ANTONIO  
SIGGRAPH  
2022

- Can make original edge detection filters “fuzzy”
- Can make morphological operators “fuzzy”



# Optimizations

- Alpha Test
- Multiple Render Targets



# Summary



- **Image-Space Outlining**
  - Silhouettes and Crease Outlines
  - Shadow Outlines
  - Texture Boundary Outlines
- Stylization with Morphology
- Antialiasing
- Optimizations
  - Alpha testing
  - Multiple Render Targets



# References

- 1.** P. Decaudin, "Cartoon-looking rendering of 3D-scenes," Technical Report INRIA 2919, Universite de Technologie de Compiegne, France, June 1996.
- 2.** J. C. Hourcade and A. Nicolas, "Algorithms for Antialiased Cast Shadows," *Computer and Graphics*, 9, 3, (1985), 259-265.
- 3.** J. L. Mitchell, "Image Processing with Direct3D Pixel Shaders" in *Vertex and Pixel Shader Tips and Tricks*, Wolfgang Engel editor, Wordware, 2002.
- 4.** T. Saito and T. Takahashi, "Comprehensible Rendering of 3-D Shapes." *Computer Graphics* (SIGGRAPH '90 Proceedings), pages 197-206, 1990.

