Johns Hopkins
Engineering for Professionals

605.767 Applied Computer Graphics

Brian Russin

Module 9I
Procedural Textures in Shaders

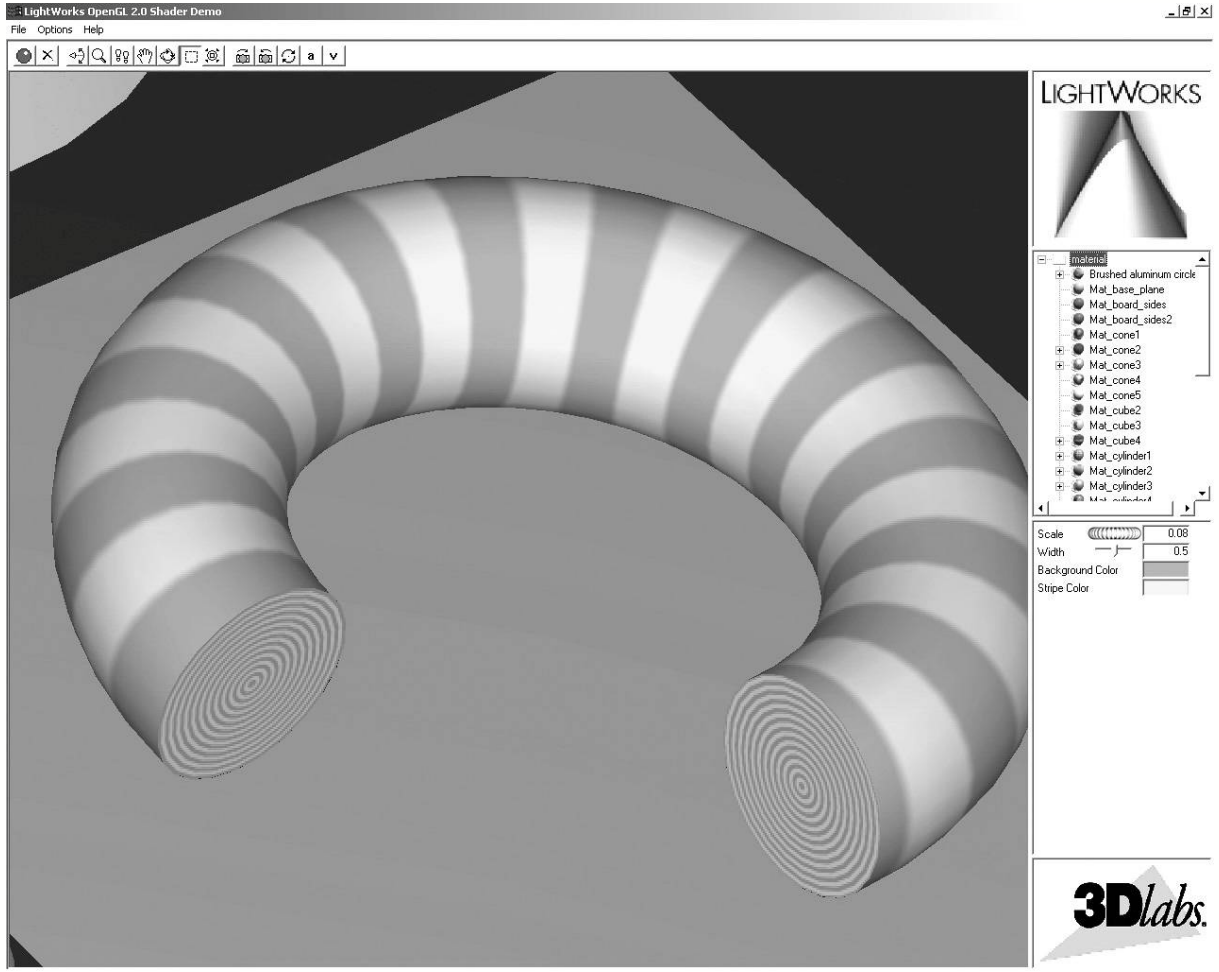
Procedural Texturing

- Recent (relatively) advances make this possible
  - GPUs are more capable than before
    - Programmable shaders
    - Built-in functions
  - Powerful, faster GPUs
- Advantages
  - Small memory footprint
    - Usually orders of magnitude smaller than using textures, especially 3D textures
  - No fixed resolution
  - Can be parameterized to work with different objects
- Disadvantages
  - More expensive than sampling shaders
  - Aliasing artifacts
    - No filtering support like mip-maps
  - Non-deterministic across platforms
    - Built-in functions are platform-specific (i.e. noise)

Case Studies

- Implementation Options
  - GPU functions
    - noise, clamp, smoothstep, fract, etc.
    - sometimes optimized in hardware
  - User-defined functions
  - Uniform variables, textures, vertex attributes
- OpenGL Shading Language 3rd Edition by Randi Rost, Bill Licea-Kane
    - Chapter 11
  - Stripes
  - Luxo Ball
  - Bump Mapping
    - Regular grid
    - 3D Noise

Stripes on Torus



https://www.yaldex.com/open-gl/ch11lev1sec1.html

Stripes

- Regular pattern based on fragment coordinate
  - Works best with local coordinate
    - Texture coordinate is a good option
      - Like a texture map
  - Exterior coordinates
    - World coordinates, view coordinates, etc.
    - Pattern anchored to some 3D volume
    - Effect of passing through a solid object

Stripes (cont.)

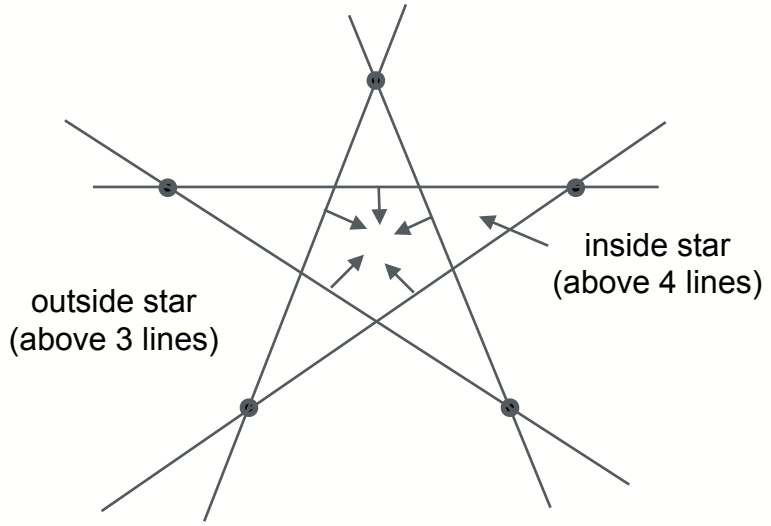
```
// fragment shader
// From OpenGL Shading Language, 3rd Edition, section 11.1.2

uniform vec3 stripe_color;
uniform vec3 back_color;
uniform float width;
uniform float fuzz;
uniform float scale;

in vec2 tex_coord;
out vec4 frag_color;

main {
    float scaled_s = fract(tex_coord.s * scale);
    float frac1 = clamp(scaled_s / fuzz, 0.0, 1.0);
    float frac2 = clamp((scaled_s - width) / fuzz, 0.0, 1.0);

    frac1 = frac1 * (1.0 - frac2);
    frac1 = frac1 * frac1 * (3.0 - (2.0 * frac1));

    vec3 final_color = mix(back_color, stripe_color, frac1);
    frag_color = vec4(final_color, 1.0);
}
```

Luxo Ball (Pixar)

https://www.deviantart.com/luxoveggiedude9302/art/Luxo-Ball-Model-Render-773476116

Luxo Ball

- Features
  - Yellow base color
  - Red stars on opposite sides
  - Blue stripe
- Strategy
  - Fragment points in local sphere coordinates
  - Blue Stripe
    - 2 planes
    - Blue if point in between planes
  - Red Stars
    - Define 5 planes
    - Red if above 4 or more planes



Star shape in 2D
Shaded region if above 4 or more lines

Luxo Ball (cont.)

- Algorithm
  - If above stripe-plane-1 and above stripe-plane-2
    - output blue
  - else if above four or more star planes
    - output red
  - else
    - output yellow

Bump Mapping

- Texture-based
  - Surface normal defined per-fragment
    - Texture mapping uses normal maps
  - Sampled per-fragment
  - Lighting performed in tangent space
    - Alternatively, transform normal to world space
- Procedural
  - Produce surface normal
    - Regular pattern
      - e.g. grid of dimples
      - can use texture coordinates like the stripe pattern
    - 3D Noise function
      - vec3 noise3(vec3 x)
  - Everything is the same as texture-based bump mapping