Procedural Texturing and Shading
Procedural Texturing/Shading

Paradigm for programmability in the graphics pipeline

Allows for a wide variety of surface materials and embellishments

May be facilitated by a custom shading language

• e.g. Pixar’s RenderMan, NVIDIA’s CG
Potential Advantages of Procedural Textures

Compact representation
No fixed resolution
No fixed area
Parameterized - generates class of related textures
Disadvantages of Procedural Textures

Difficult to build and debug

Surprising results

Slow evaluation

Antialiasing handled manually
Procedural Texture Conventions

Avoid conditionals

- Convert to mathematical functions when possible
- Makes anti-aliasing easier

Parameterize rather than building in constants

- Assign reasonable defaults which may be overridden
Simple Building Blocks

Mix (lerp)
Step, smoothstep, pulse
Min, max, clamp, abs
Sin, cos
Mod, floor, ceil
Mix

mix(a, b, x)
Step

\[ \text{step}(a, x) \]
Smoothstep

smoothstep(a,b,x)
Pulse

\[ \text{pulse}(a,b,x) = \text{step}(a,x) - \text{step}(b,x) \]
Clamp

$$\text{clamp}(x, a, b) = \min(\max(x, a), b)$$
Mod

\[ \text{mod}(x, a) / a \]
Periodic Pulse

\[ \text{pulse}(0.4, 0.6, \text{mod}(x,a)/a) \]
Example 1 - brick (see handout)

Brick is primarily a 2D pulse

Input parameters may include:

- color of brick and mortar
- size of brick
- thickness of mortar
- mortar bump size
- frequency of brick color variation
- etc.
Brick

Example 2 - star (see handout)

Exploit symmetry of star geometry

Input parameters may include:

- Inner and outer star radii
- Number of points
- Star and background colors
- Star bump parameters
- Parameters for star distribution
Star