


# Procedural Shaders

CS 563 Advanced Topics in Computer Graphics

Fan Wu

Feb. 24, 2005

## What is Procedural Shading?

- Not that straightforward to define
- An image-based texture has parameters (u, v, texture scale)
- We use parametric surface models to describe the appearance of a material or a light source
- Procedural shaders also use images
- Examples
  - http://www.nzone.com/object/nzone_squiddemo_home.html

## Why procedural shading?

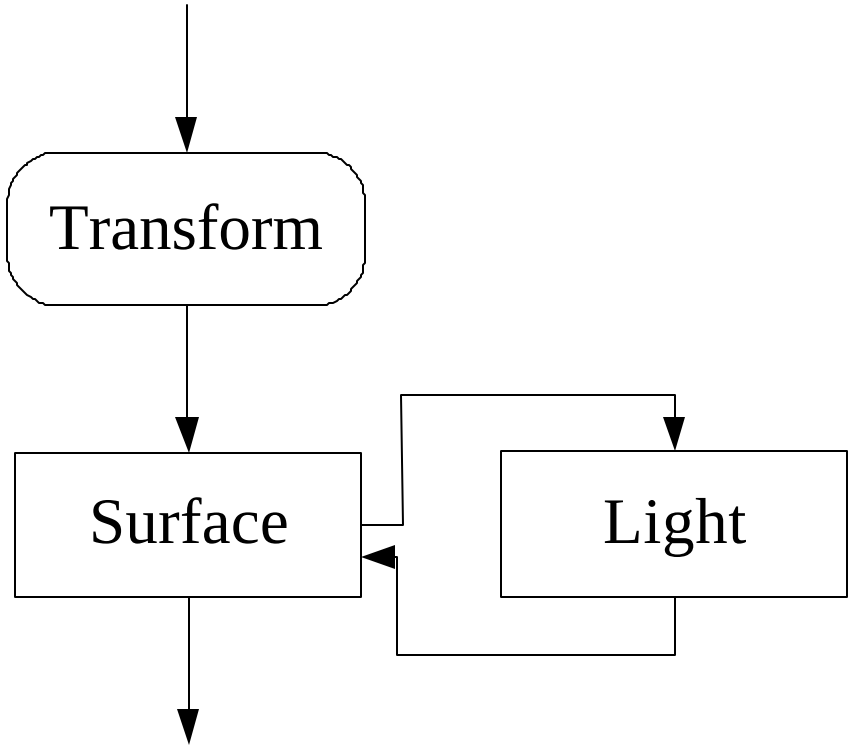
- Compactness (especially for 3D textures)
- No fixed resolution
- Can make time varying
- Parameters you can manipulate to get the look you want (of course, you must manipulate them!)

## Characteristics

- Parameters are often not intuitive
- Antialiasing takes programming effort
  - Imagine procedural checkerboard
- More than just surfaces
  - Lights
  - Displacement
  - Volumes (fog)
  - Primitives

# Logical Model

- The logical model of Procedural shaders



## RenderMan

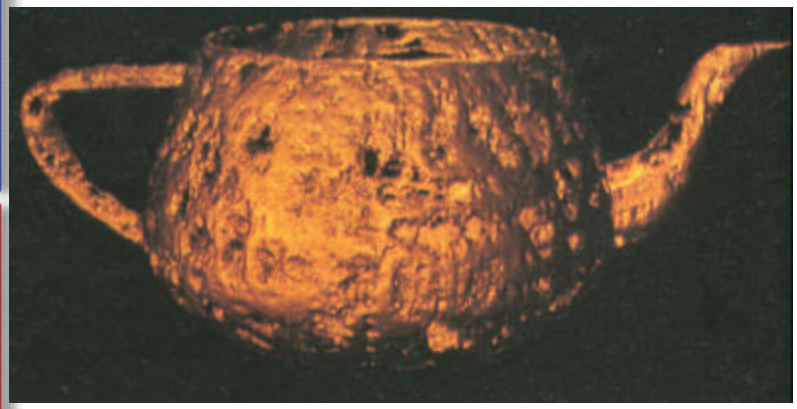
- RenderMan standard was presented by Hanrahan and Lawson in 1990
- Provides a geometry description library similar to OpenGL
- Provides a geometric file format (RIB)
- Provides a shading language --Pixar

# RenderMan (Cont.)

- RenderMan is mainly applied in movie maker
  - A Bug Life
  - Toy Story
  - Monsters Inc.
- The core technology is a shading language, which provide a flexible description of shading effect

# RenderMan (Cont.)

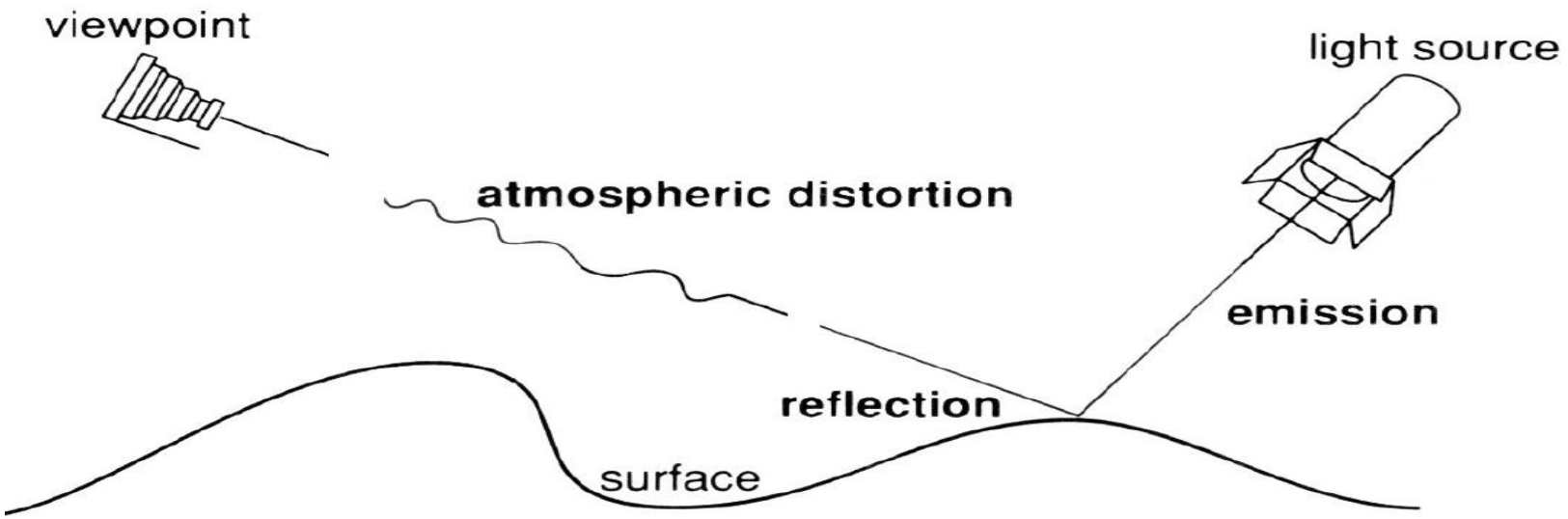
- Example of RederMan Shading Language



```
Surface dent(float Ks=.4, Kd=.5, Ka=.1, roughness=.25, dent=.4) {
    float turbulence;
    point Nf, V;
    float I, freq;
    /* Transform to solid texture coordinate system */
    V = transform("shader",P);
    /* Sum 6 octaves of noise to form turbulence */
    turbulence = 0; freq = 1.0;
    for (i = 0; i < 6; i += 1) {
            turbulence += 1/freq + abs(0.5*noise(4*freq*V));
            freq *= 2;
    }
    /* sharpen turbulence */
    turbulence *= turbulence * turbulence;
    turbulence *= dent;
    /* Displace surface and compute normal */
    P -= turbulence * normalize(N);
    Nf = faceforward(normalize(calculatenormal(P)),I);
    V = normalize(-I);
    /* Perform shading calculations */
    Oi = 1 – smoothstep(0.03,0.05,turbulence);
    }
```

# RenderMan (Cont.)

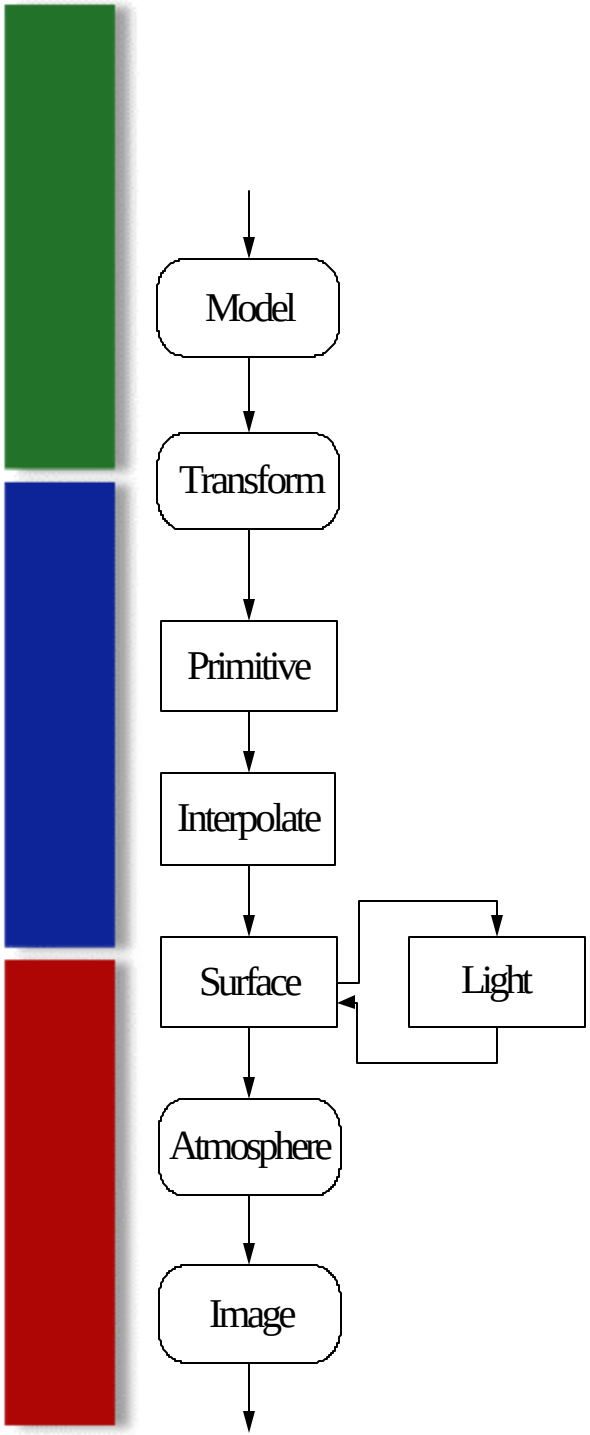
- The Basic content of RenderMan Shading Language
  - Light Source Shader
  - Surface Shader
  - Volume Shader



# RenderMan (Cont.)

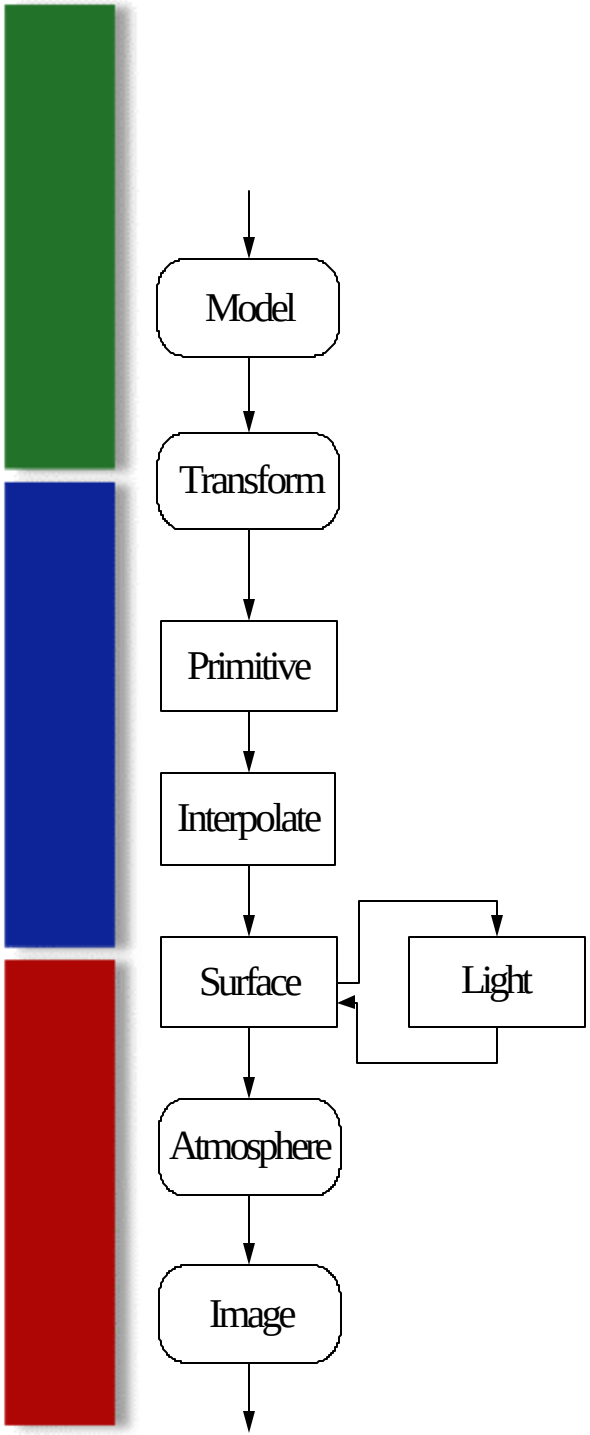
- The Advantages of RenderMan Shading Language
  - Lighting of complex surface
  - Random or noise effect of lighting
  - Easy to simulate detail of image
  - Comparing with Texture Mapping, lighting effect can be various as time, distance or angel changes.

# PixelFlow

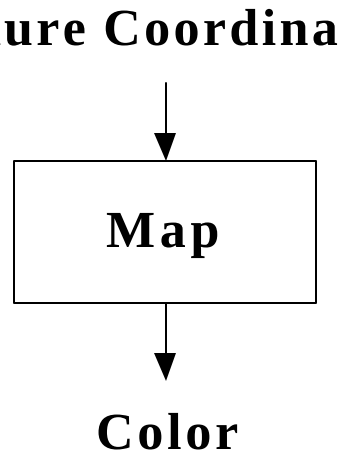



- A fully programmable machine
- Provides a good start for looking at the organization of the elements used for a real-time shading

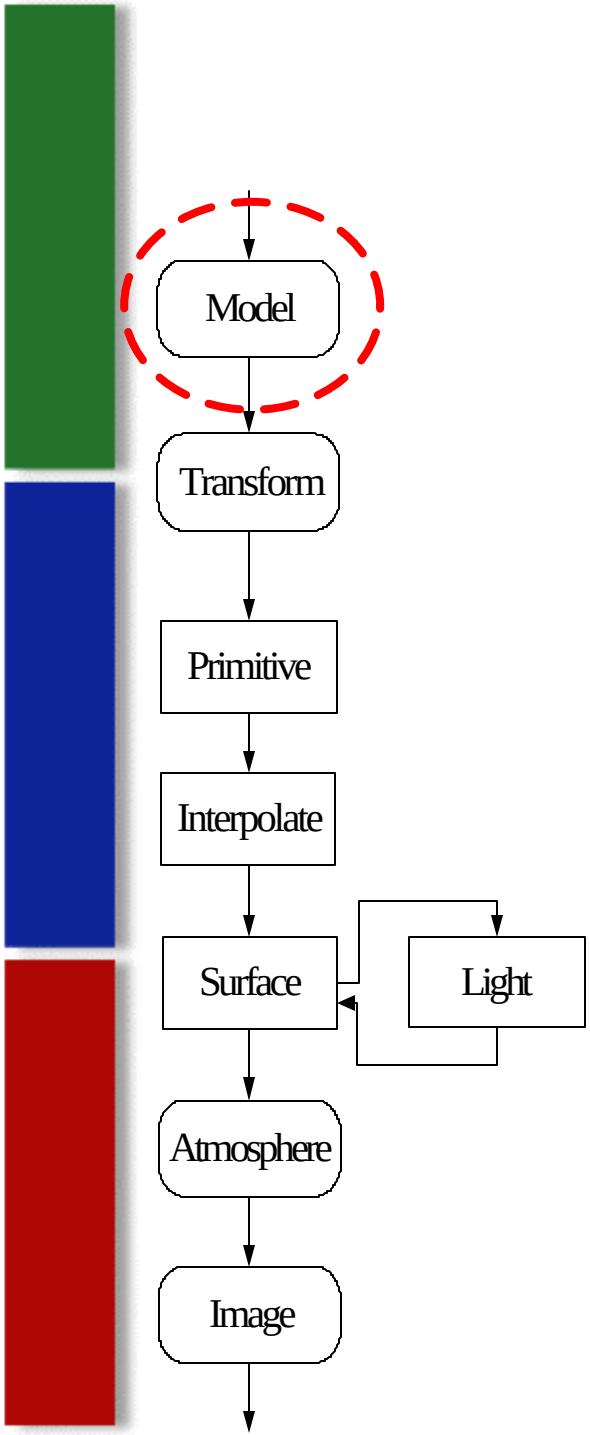
# PixelFlow (Cont.)



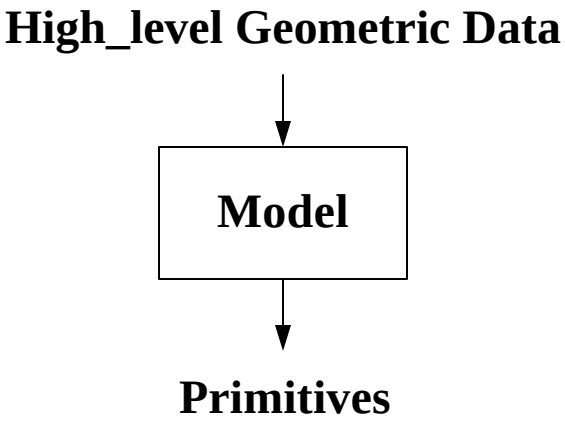

- Maps
  - Not a stage, but a type of procedure that may be used by any of the stages
  - Start with a two- or three- dimensional texture
  - The resulting value is used as a parameter to the shading model



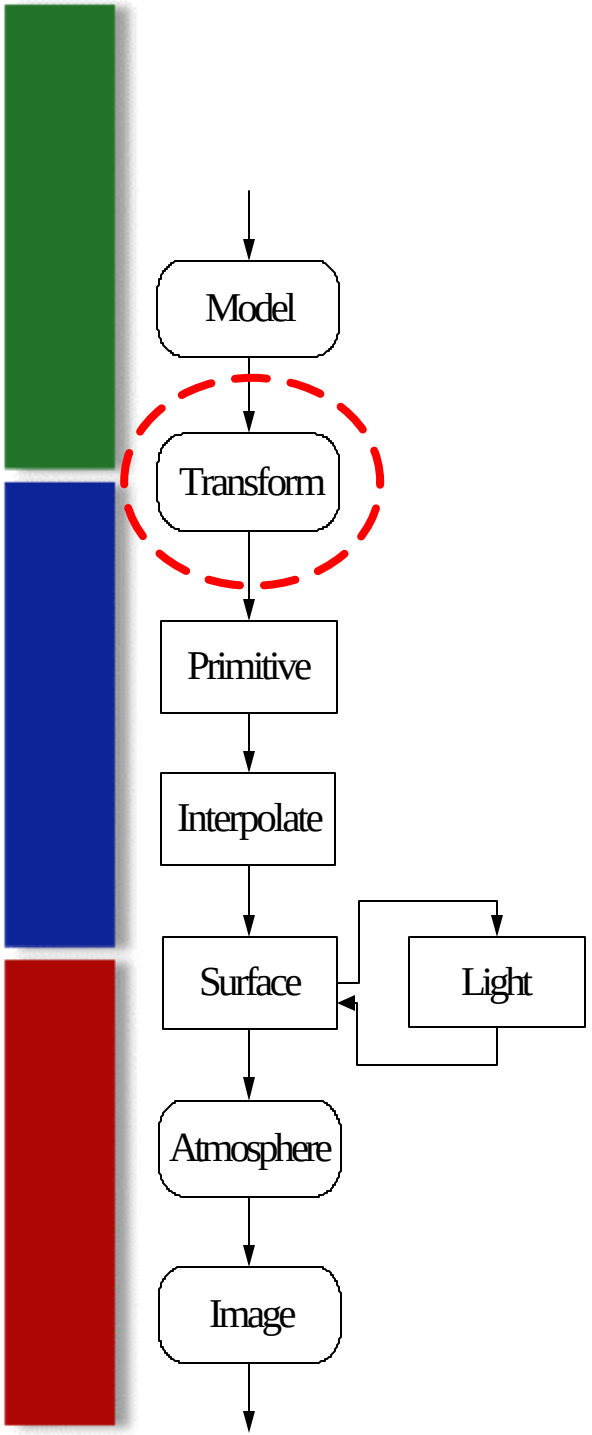
# PixelFlow (Cont.)



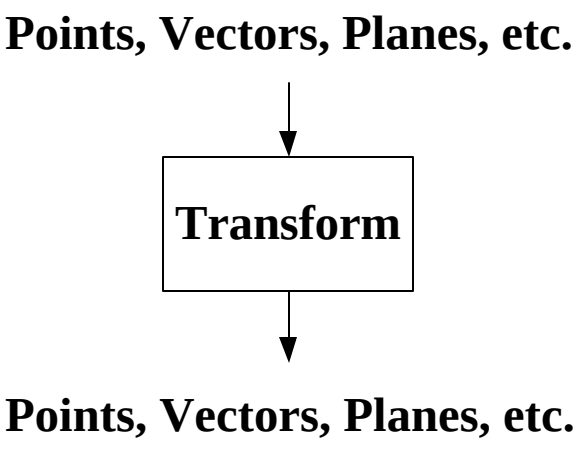

- Modeling
  - Construction of objects and scenes out of basic geometric promitives
  - Use a set of control parameters to generate a description of the model



# PixelFlow (Cont.)

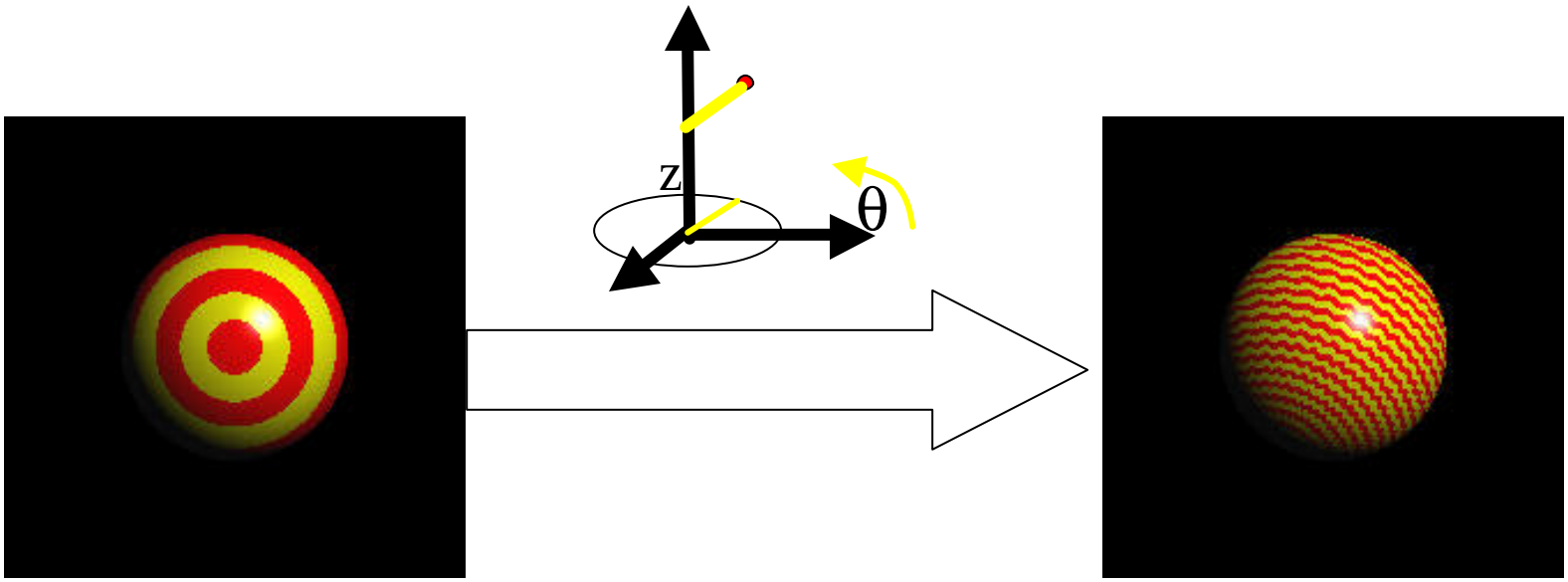



- Transformation
  - Mappings of an object from one coordinate system to another
  - Takes a 3D point or vector as its input and produces a new 3D point or vector
  - Linear mapping, Global and local deformation, Free form deformation, etc.

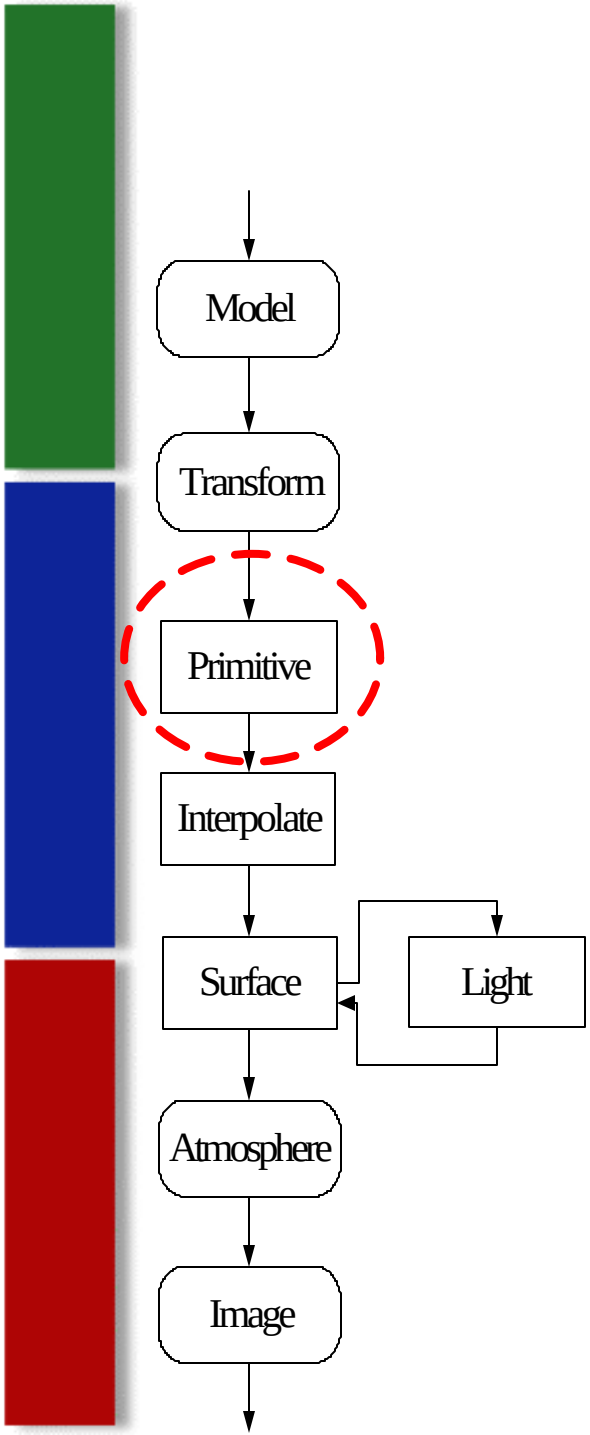


# PixelFlow (Cont.)

- Example:

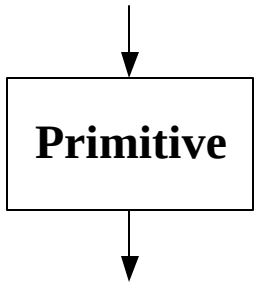


# PixelFlow (Cont.)

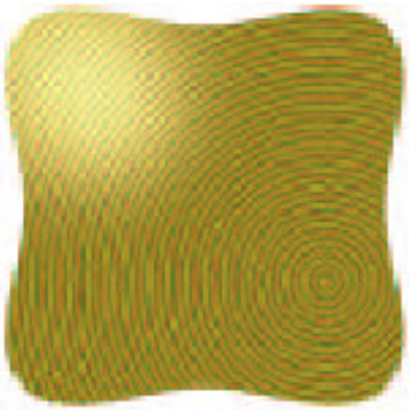



- Primitive
  - Basic building units
  - Decides which pixels are inside the primitive
  - Compute values for some shading parameters

**Primitive Geometric Data**

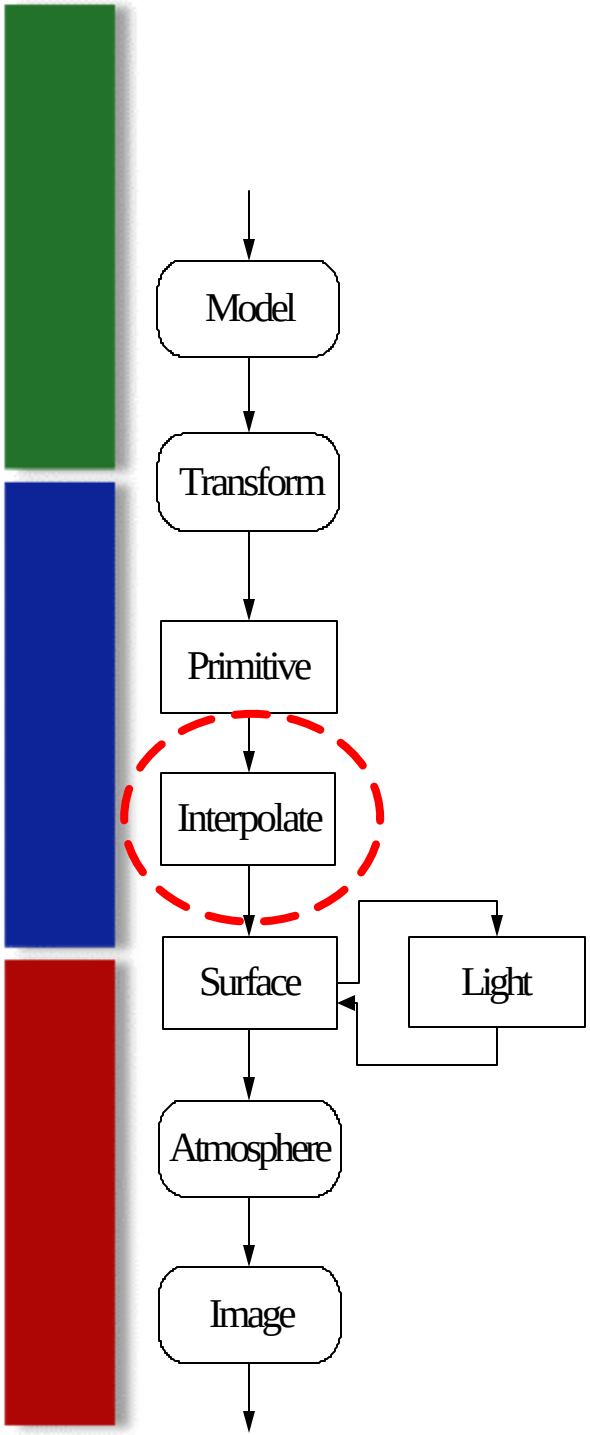



**Pixels and Shading Parameters**

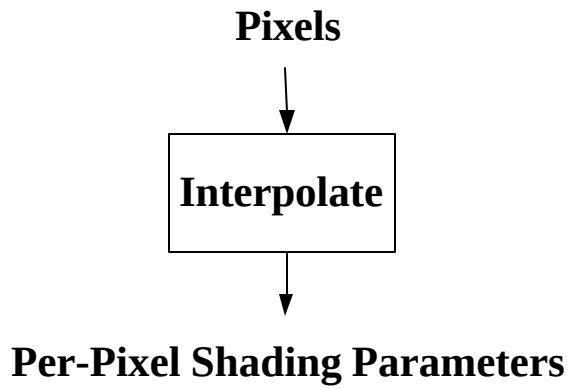


Example of a procedural primitive

# PixelFlow (Cont.)



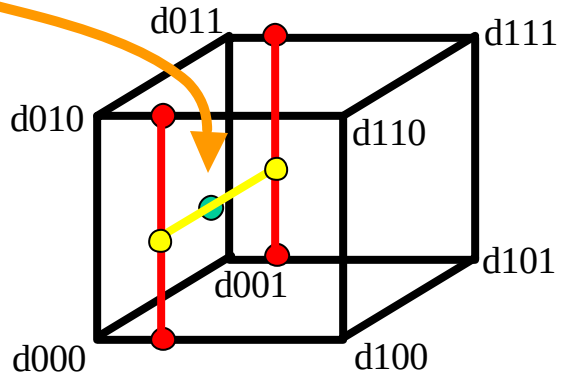

- Interpolate
  - The computation of shading parameter values across each primitive
  - Independent of the shading procedure or its parameters
  - Eg: Texture coordinate generators in OpenGL, Ebert's solid spaces



# PixelFlow (Cont.)

## Example: Interpolate values within cube

(x,y,z)




fx = FRACT(x)
fy = FRACT(y)
fz = FRACT(z)

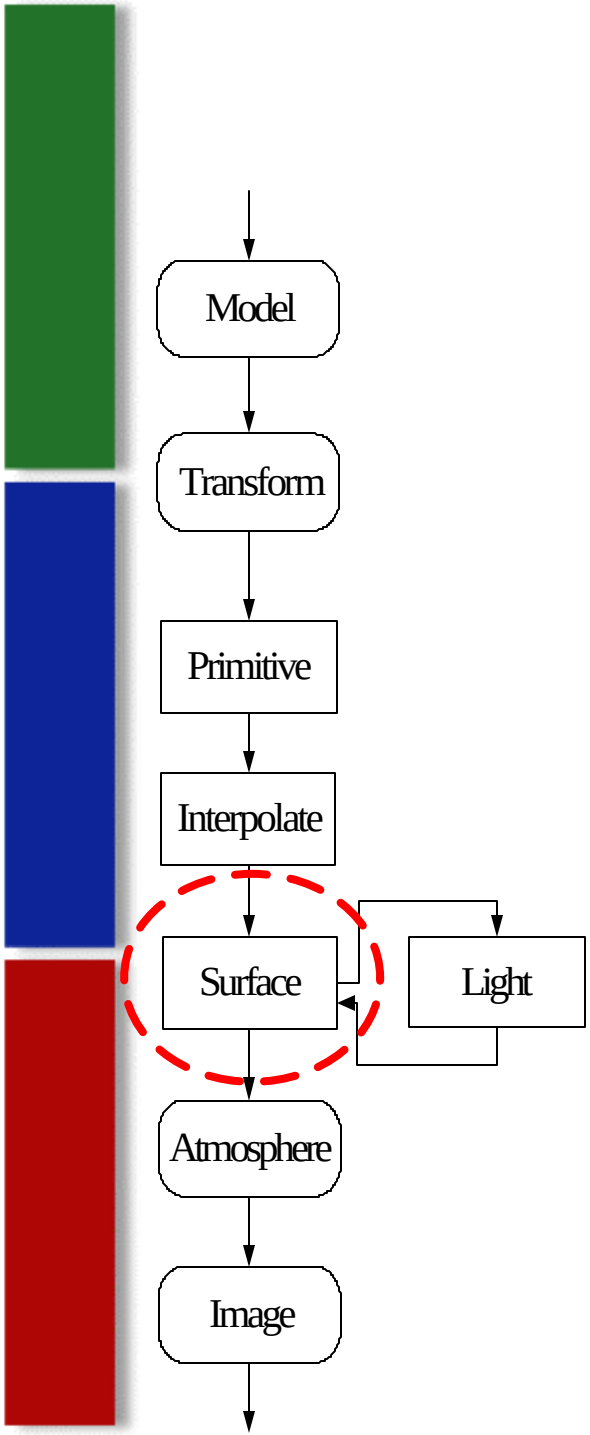
Use tri-linear interpolation

d00 = d000+fx(d100-d000)
d10 = d010+fx(d110-d010)
d01 = d001+fx(d101-d001)
d11 = d011+fx(d111-d011)

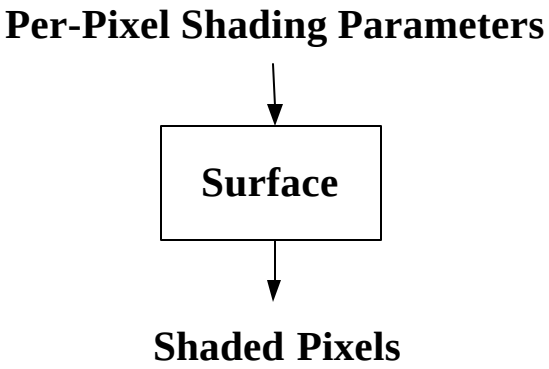
d0 = d00+fy(d10-d00)
d1 = d01+fy(d11-d01)

d = d0+fz(d1-d0)

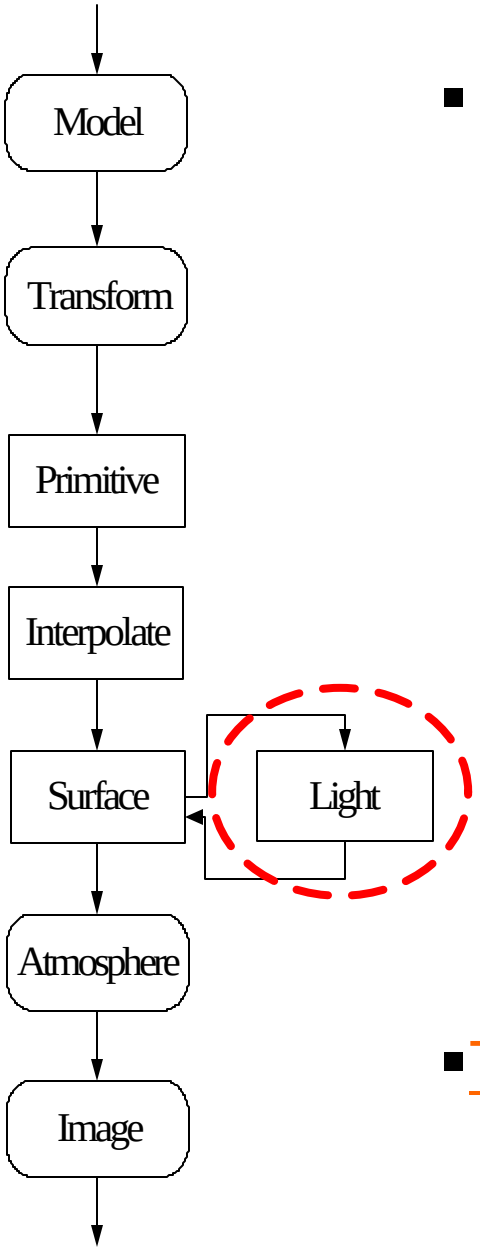
# PixelFlow (Cont.)



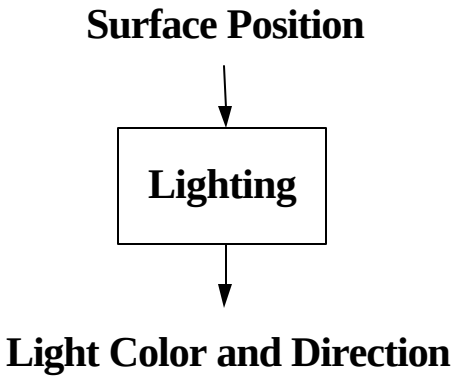

- Surface Shading
  - Describes the shading of a surface through a simple function to turn the surface attributes and shading parameters into a color
  - Eg: Cook's shade trees, Perlin's image synthesizer, RenderMan Shading Language,etc.

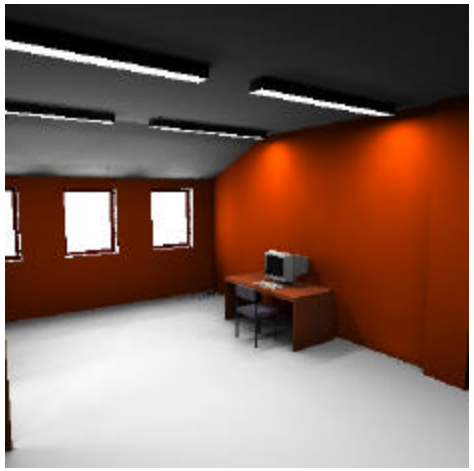


# PixelFlow (Cont.)



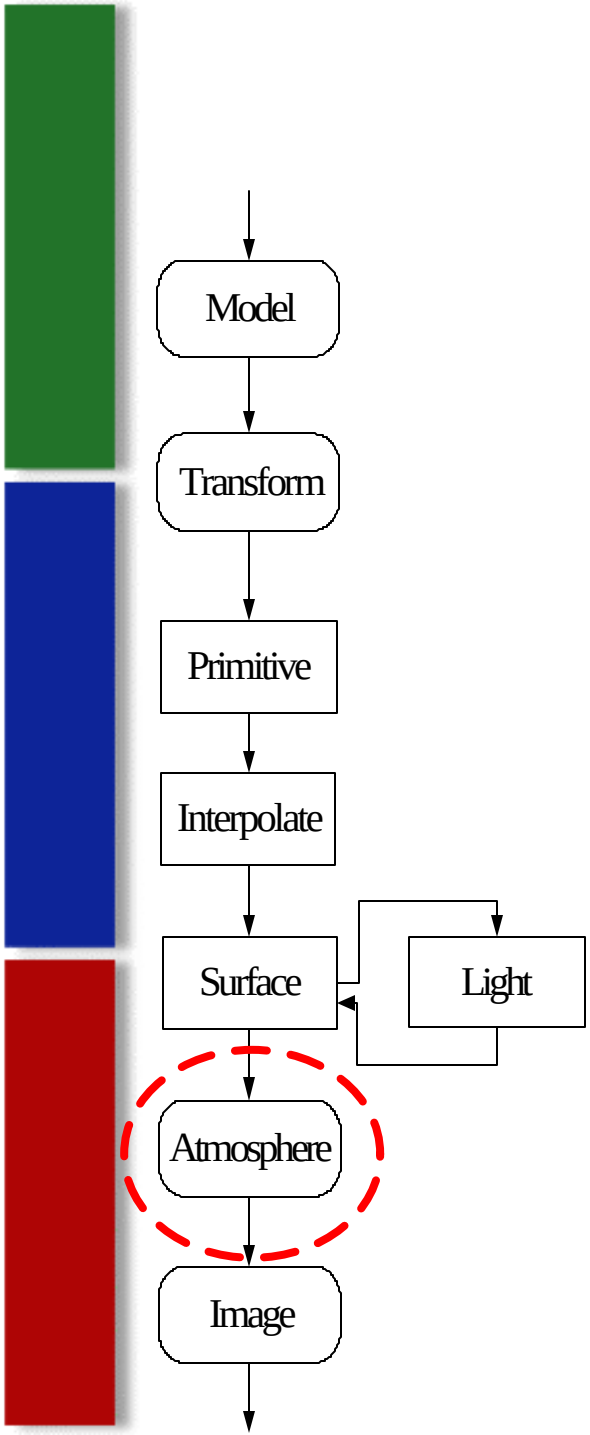

- Lighting
  - Determine the intensity and color of light that hits a surface point from a light source
  - A lighting procedure may be used by all surface procedure
  - Eg: Pixar's Tin Toy, Slusallek's LightOp, etc.



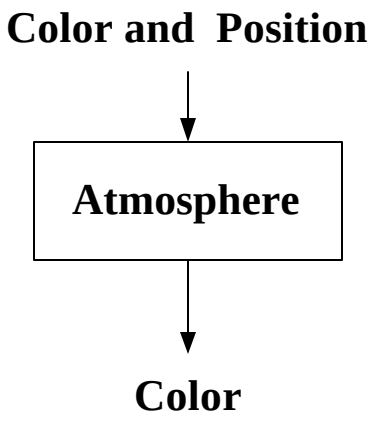



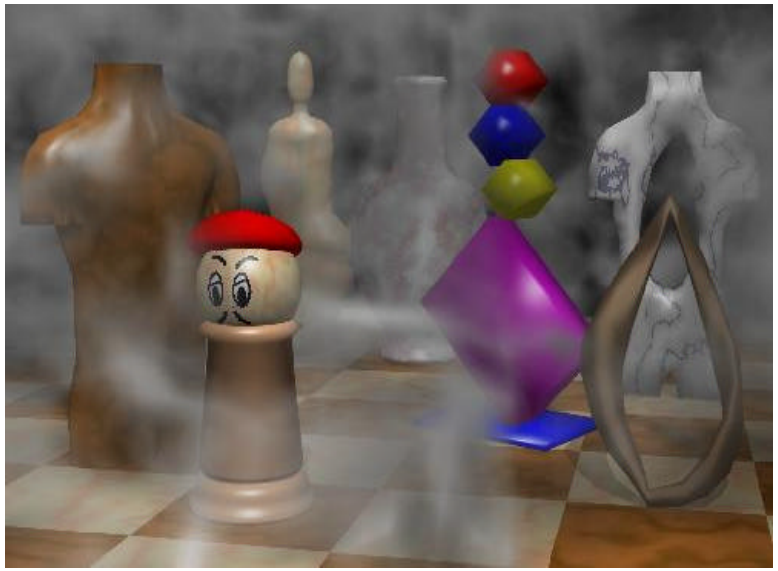
- Transformation and Lighting Demo
  - http://www.nzone.com/object/nzone_cavedemo_home.html

# PixelFlow (Cont.)

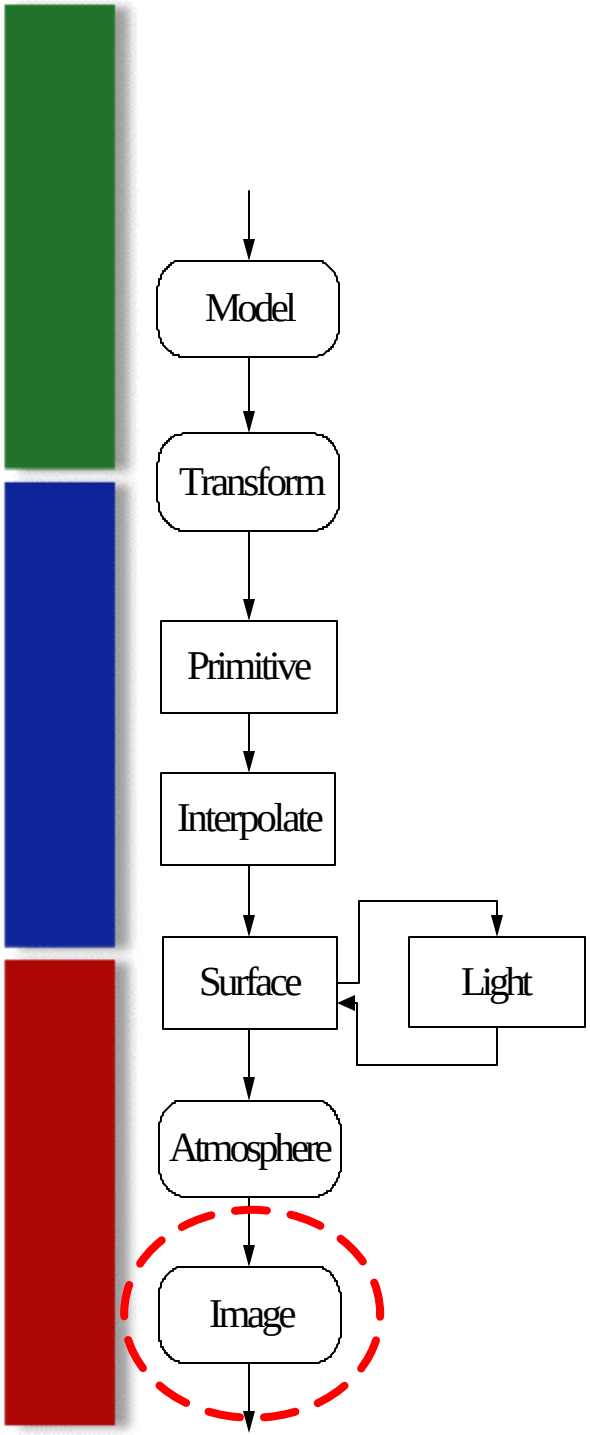



- Atmosphere
  - Handle the behavior of light as it pass through a medium, such as fog, haze and so on
  - Take in a color produced from a surface in the scene and modify it.

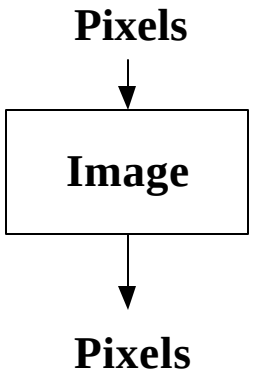


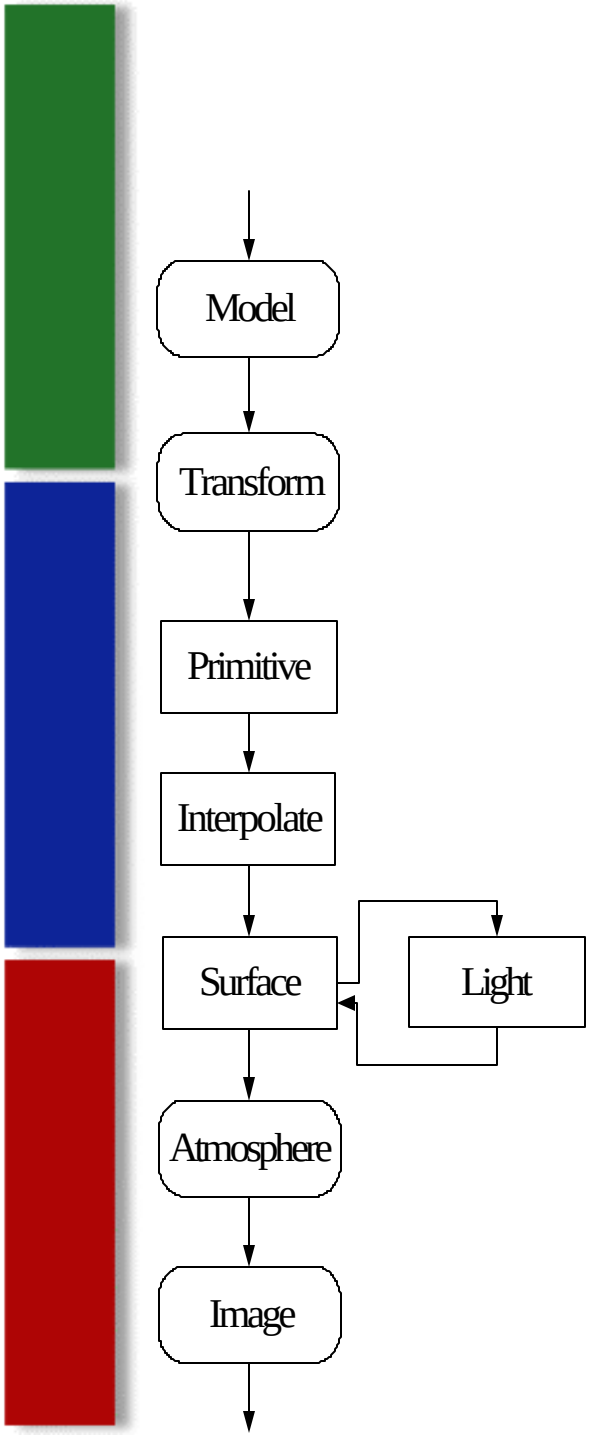
# PixelFlow (Cont.)




- Image
  - Image Warping
    - Support a host of video-warping special effects
    - Compensate for the barrel distortion
  - Image Filtering
    - Combine image pixels to achieve effects like blurring, sharpening, etc.
  - Eg: Photoshop, GIMP

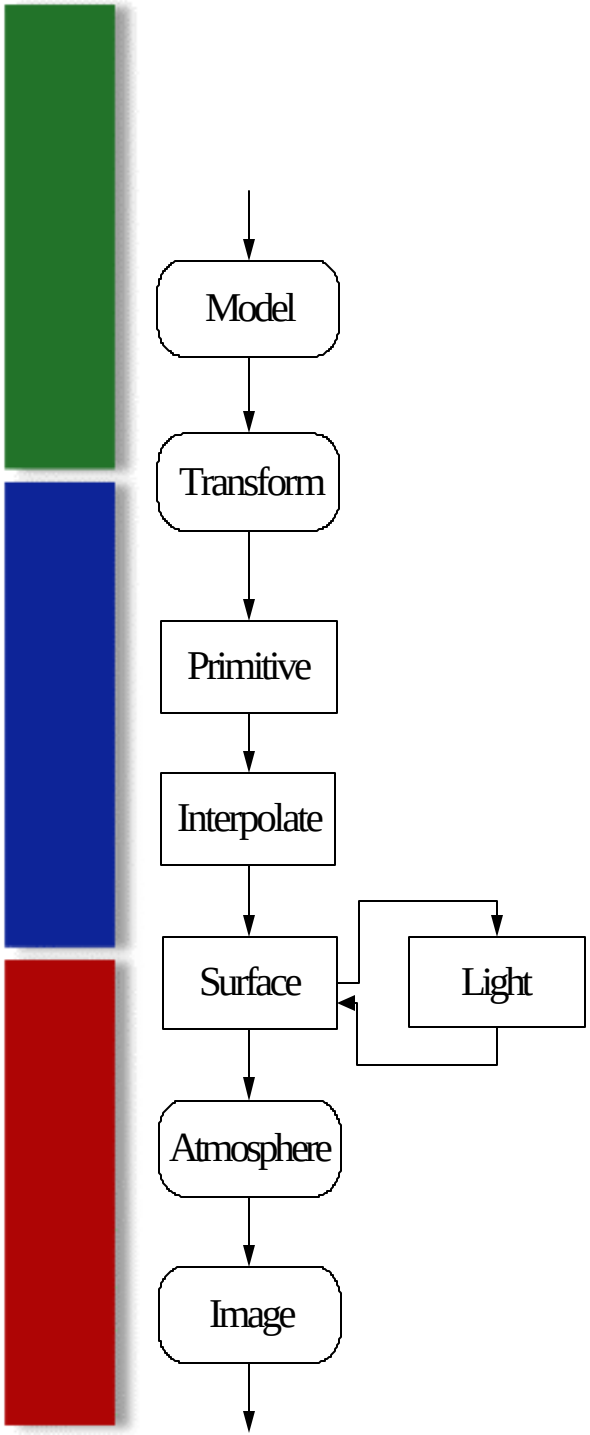


# PixelFlow (Cont.)

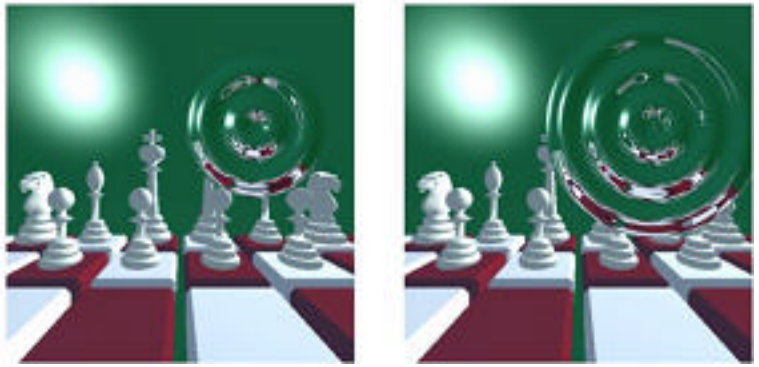



- Shading Capabilities
  - Animated shaders
  - Volume shaders
  - Shaders with great computed detail
  - Shaders that do automatic antialiasing

# PixelFlow (Cont.)

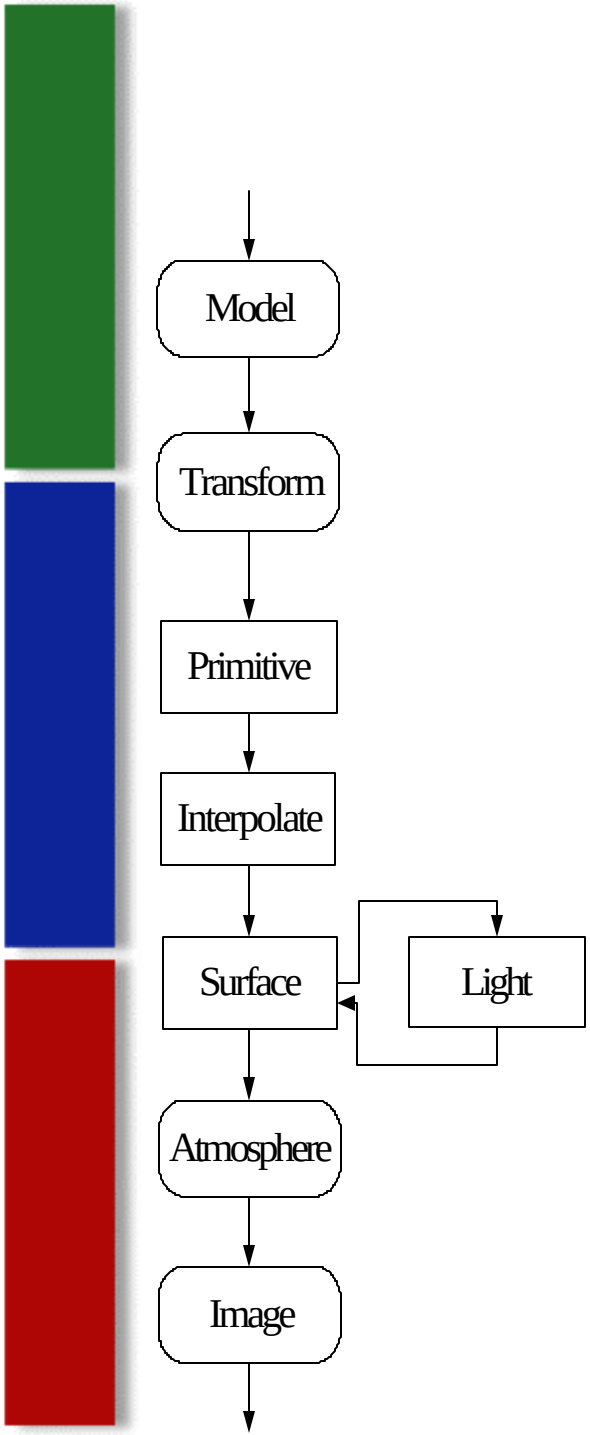



- Shading Capabilities
  - Animated shaders
  - Volume shaders
  - Shaders with great computed detail
  - Shaders that do automatic antialiasing

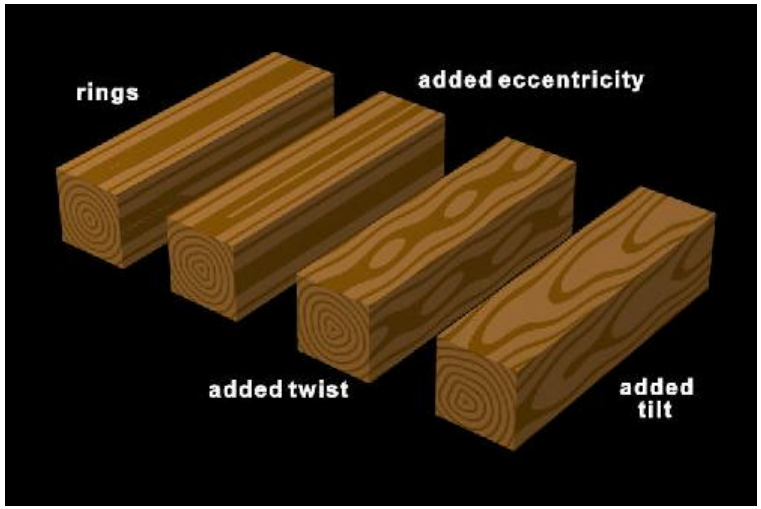


Two frames of rippling mirror

# PixelFlow (Cont.)

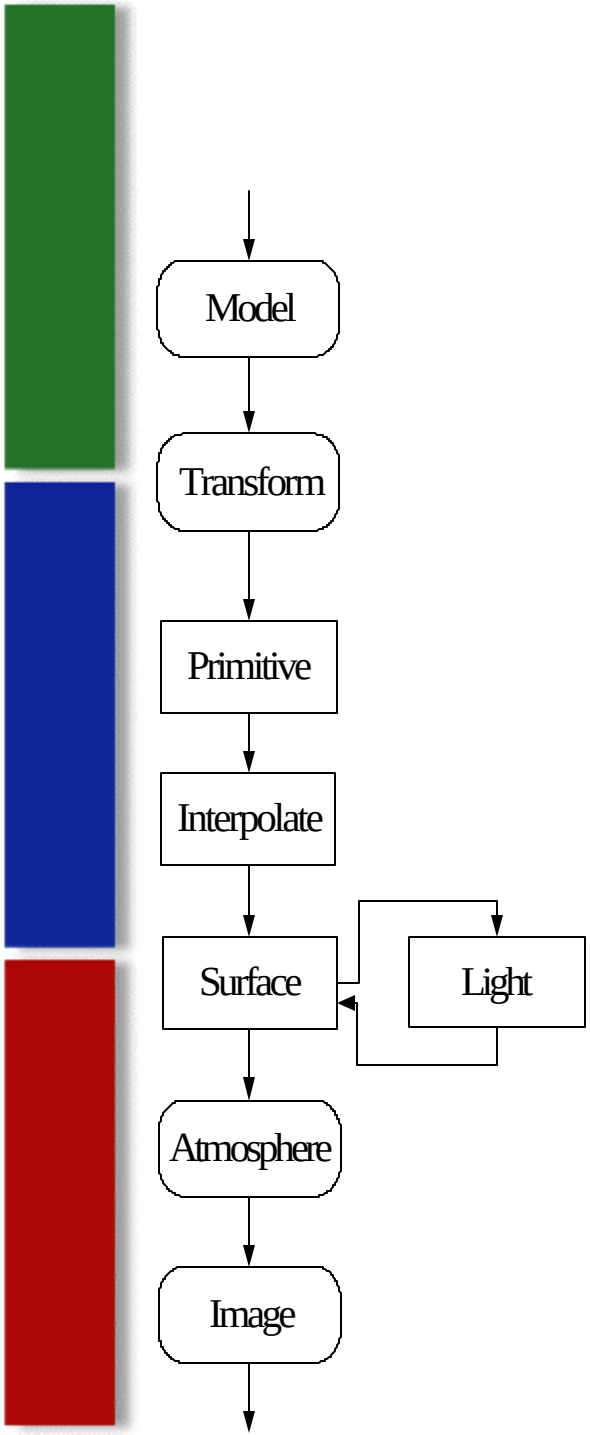



- Shading Capabilities
  - Animated shaders
  - Volume shaders
  - Shaders with great computed detail
  - Shaders that do automatic antialiasing

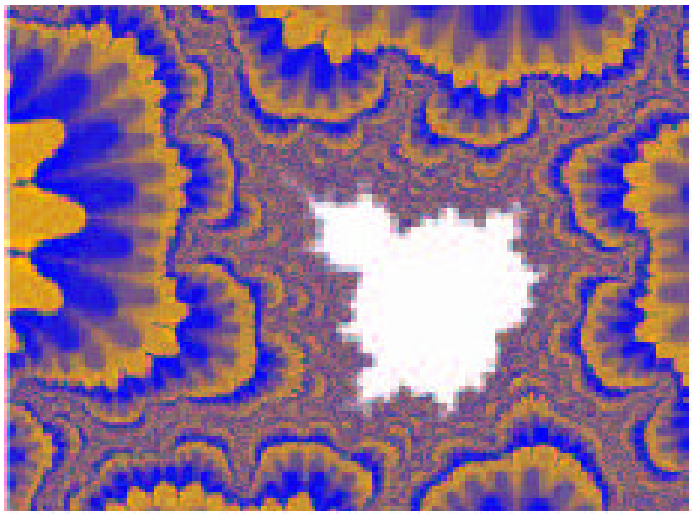



Wood volume shader

# PixelFlow (Cont.)

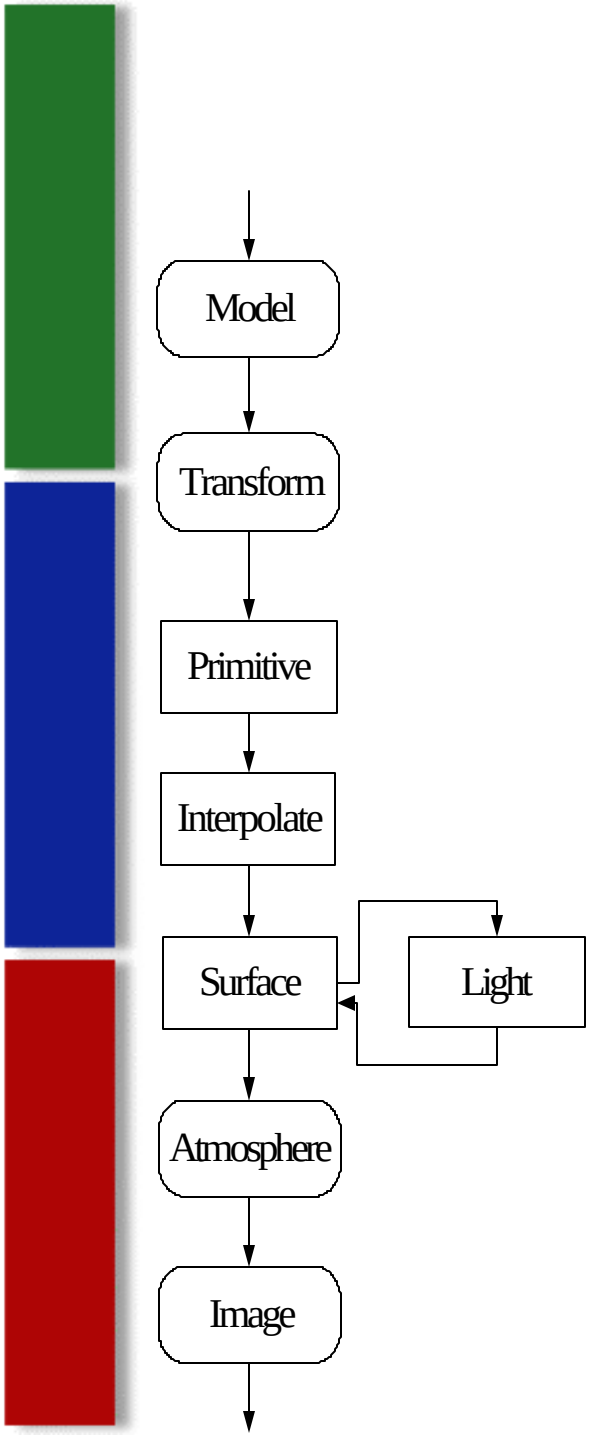



- Shading Capabilities
  - Animated shaders
  - Volume shaders
  - Shaders with great computed detail
  - Shaders that do automatic antialiasing



A surface shader that computes the Mandelbrot Set

# PixelFlow (Cont.)




- Shading Capabilities
  - Animated shaders
  - Volume shaders
  - Shaders with great computed detail
  - Shaders that do automatic antialiasing

# Antialiasing

- Avoid the ‘jaggies’!
- Major techniques
  - Analytical Filtering
    - Convolve a simple shader with a filter kernel
    - Peachy, Step functions (Step(t))
    - RenderMan, Boxstep, smoothstep, filterstep
  - Frequency attenuation
    - Band-limited noise function, $f(x) = \sum_{i}^{n} 2^{-i}\, n\left(2^{i} x\right)$
    - $n(\ )$ is any periodic function like sine or the perlin noise function
  - Super Sampling
    - Samples are rendered for each pixel, then combined
    - Relatively easy, but costly
- Demo
  - http://www.nzone.com/object/nzone_twisterdemo_home.html

# Useful URLs

- http://www.csee.umbc.edu/~olano
- http://graphics.stanford.edu/projects/shading/
- http://mrl.nyu.edu/~perlin/
- http://freespace.virgin.net/hugo.elias/models/m_perlin.htm
- http://www.nzone.com/object
- Rendering by Procedural Shader
  - http://meshuggah.4fo.de/

.....