CSCI 4530/6530 Advanced Computer Graphics

https://www.cs.rpi.edu/~cutler/classes/advancedgraphics/S25/

Lecture 19: Shaders and Procedural Modeling

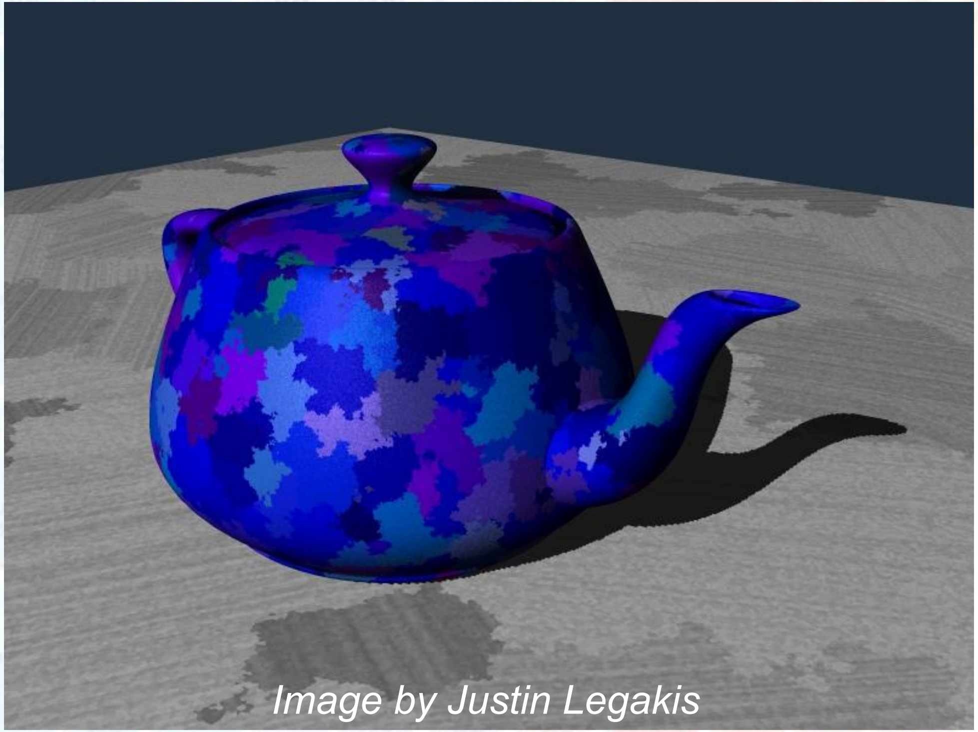
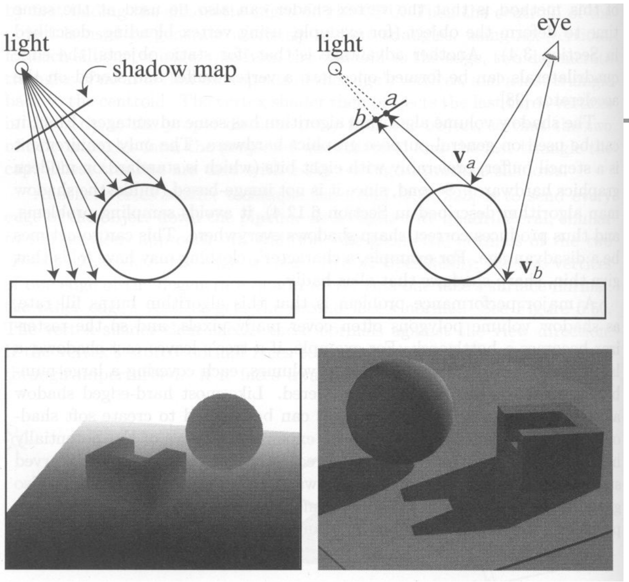


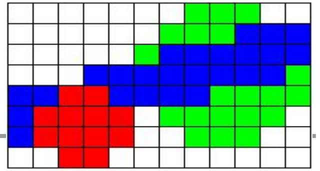
Image by Justin Legakis

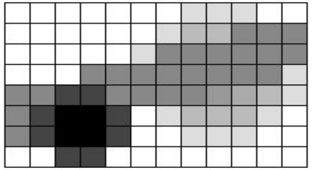
For the Birds, Pixar, 2000

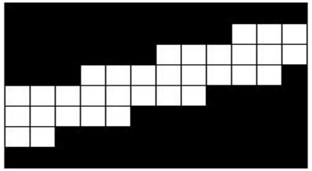
Last Time?

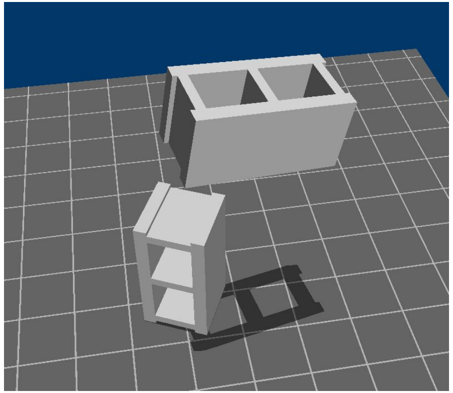
- Planar Shadows
- Projective Texture Shadows
- Shadow Maps
- Shadow Volumes
  - Stencil Buffer

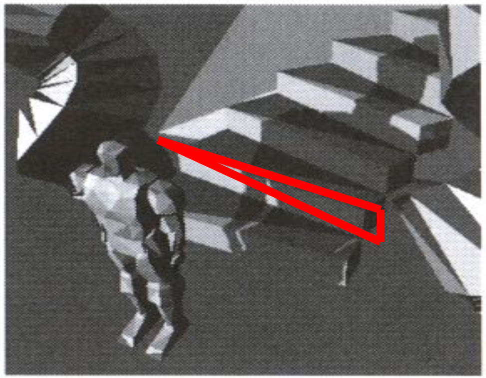


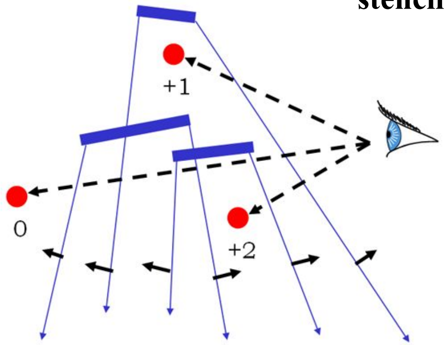

frame buffer


depth buffer


stencil buffer





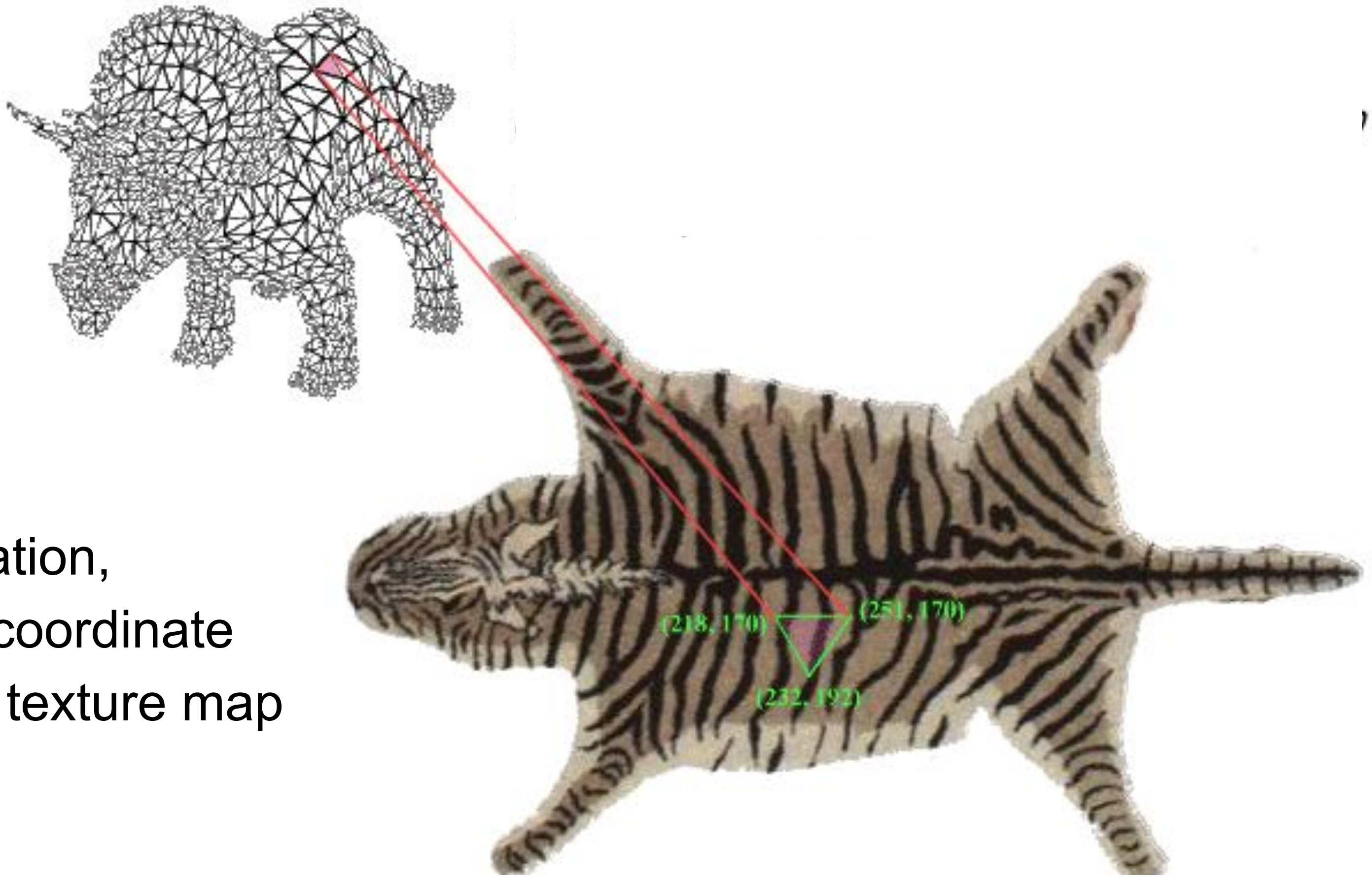


Today

- Texture Mapping & Other “Mapping” Techniques
  - Bump Mapping
  - Displacement Mapping
  - Environment Mapping
  - Light Mapping
  - Normal Mapping
  - Parallax Mapping
  - Parallax Occlusion Mapping
- Programmable Shader Examples
  - Modern Graphics Hardware
  - Per-Pixel Shading
- Procedural Textures & Modeling
- Papers for Today
- Papers for Next Time

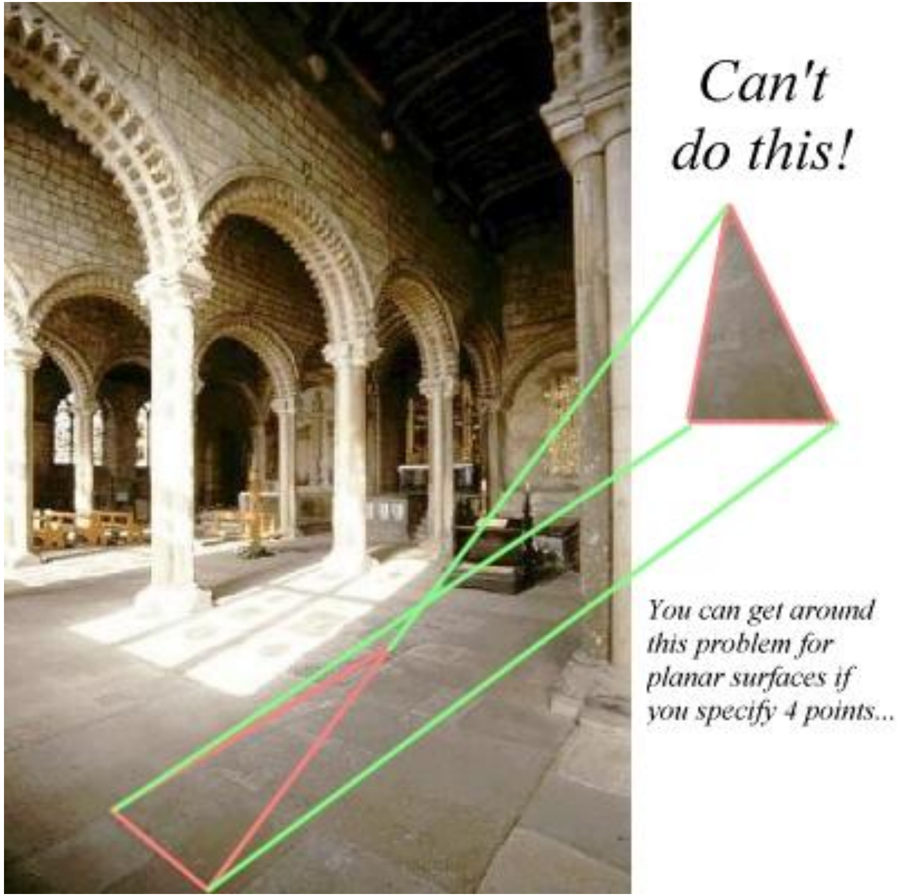
Texture Mapping

- For each triangle in the model, establish a corresponding region in the texture map.
- During rasterization, interpolate the coordinate indices into the texture map



Texture Mapping Difficulties

- Tedious to specify texture coordinates
- Acquiring textures is surprisingly difficult
  - Photographs have projective distortions
  - Variations in reflectance and illumination
  - Tiling problems

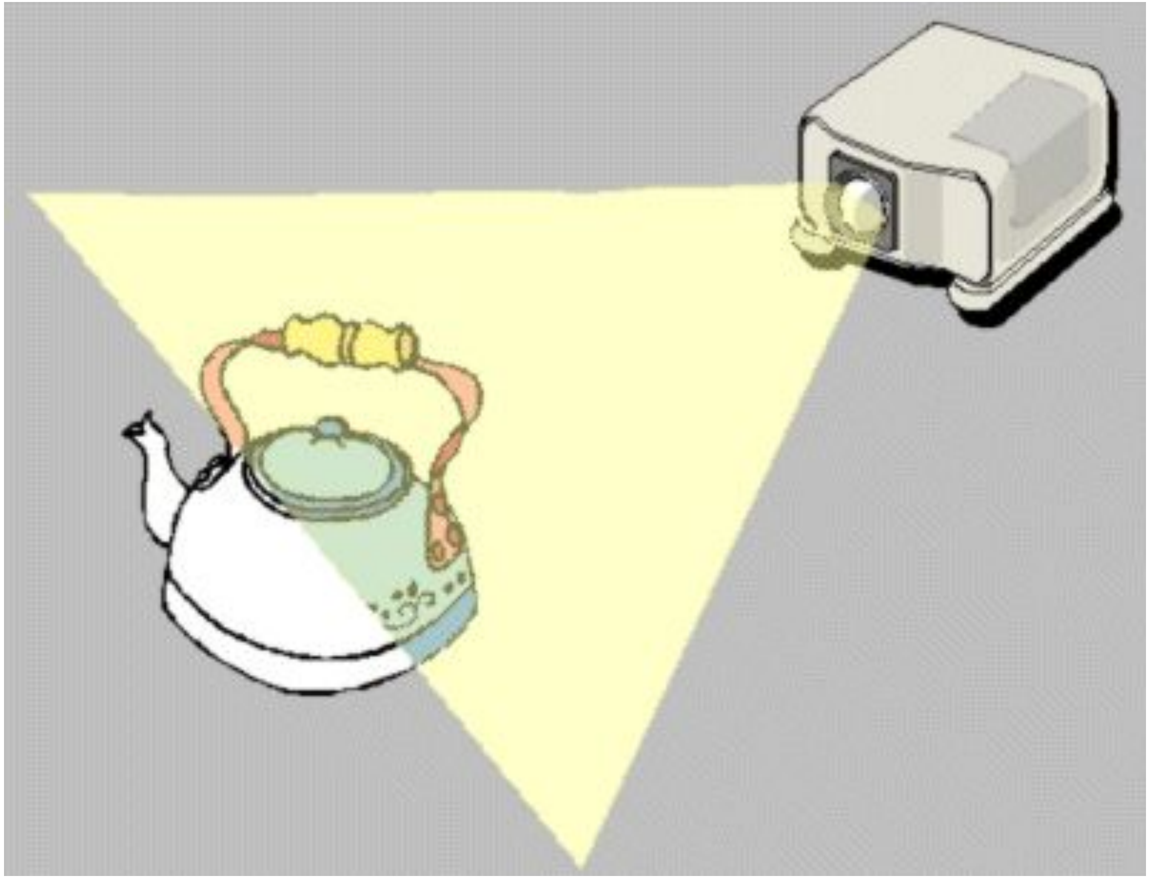


Common Texture Coordinate Mappings

- Orthogonal
- Cylindrical
- Spherical
- Perspective Projection
- Texture Chart

Projective Textures

- Use the texture like a slide projector
- No need to specify texture coordinates explicitly



Projective Texture Example

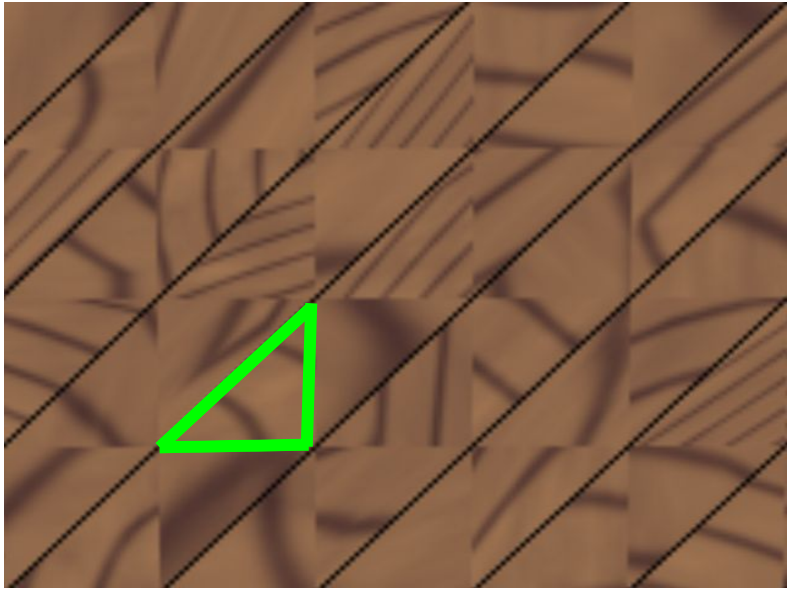
- Modeling from photographs
- Using input photos as textures

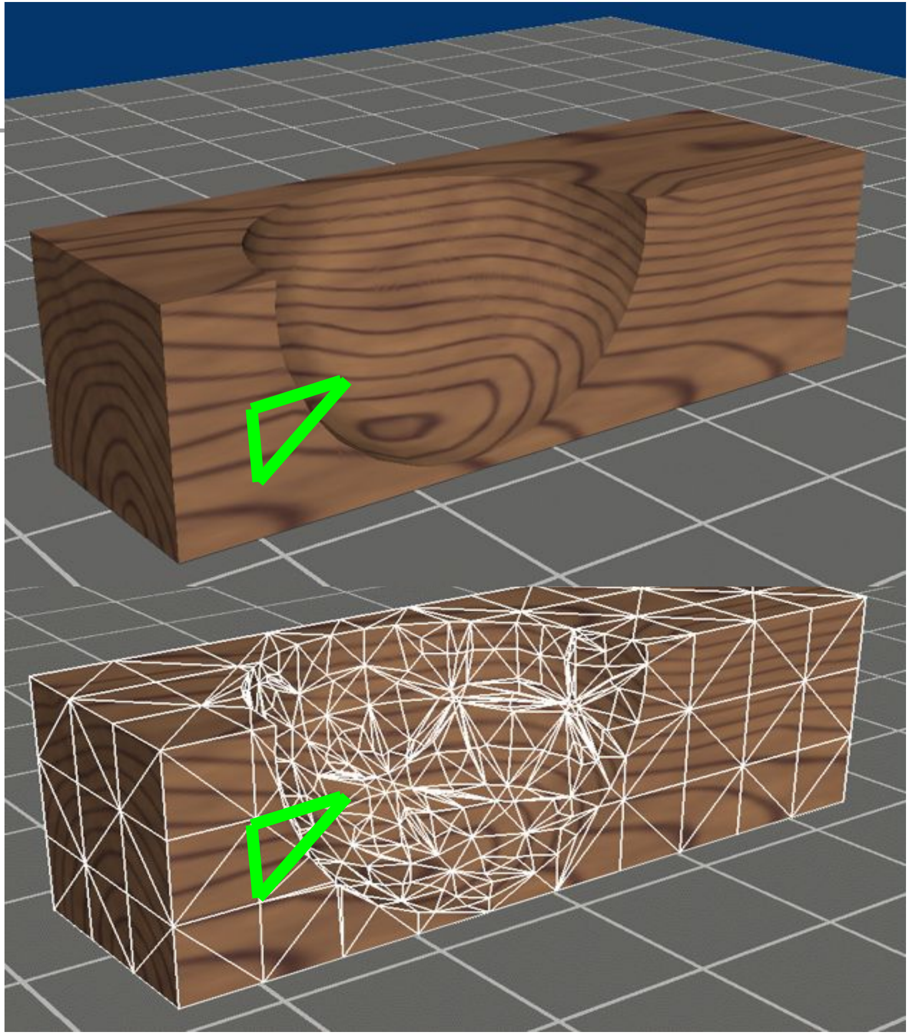
Figure from Debevec, Taylor & Malik
http://www.debevec.org/Research

Original photograph with marked edges | Recovered model | Model edges projected onto photograph | Synthetic rendering

Texture Chart

- Pack triangles into a single image





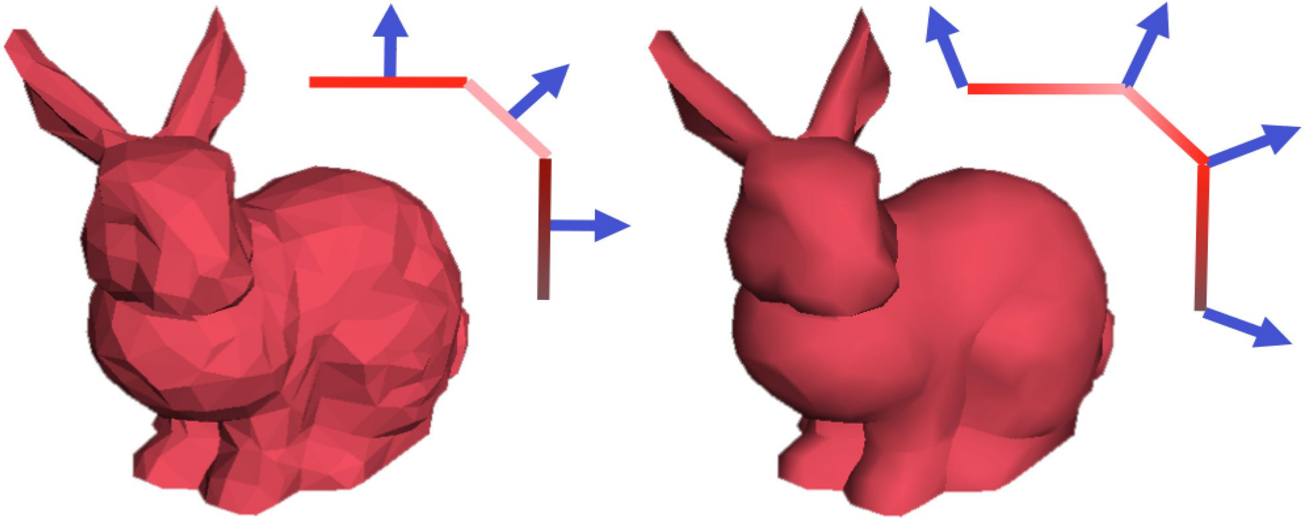
Today

- Texture Mapping & Other “Mapping” Techniques
  - Bump Mapping
  - Displacement Mapping
  - Environment Mapping
  - Light Mapping
  - Normal Mapping
  - Parallax Mapping
  - Parallax Occlusion Mapping
- Programmable Shader Examples
  - Modern Graphics Hardware
  - Per-Pixel Shading
- Procedural Textures & Modeling
- Papers for Today
- Papers for Next Time

Remember Gouraud Shading?

- Instead of shading with the normal of the triangle, we'll shade the vertices with the *average normal* at the vertex and *interpolate the shaded color* across each face. This gives the *illusion of a smooth surface* with smoothly varying normals.

- ***Cheat normals to DISGUISE actual geometry***



Bump Mapping / Normal Mapping

- Use textures to alter the surface normal
  - Does not change the actual shape of the surface
  - Just shaded as if it were a different shape

Sphere w/Diffuse Texture

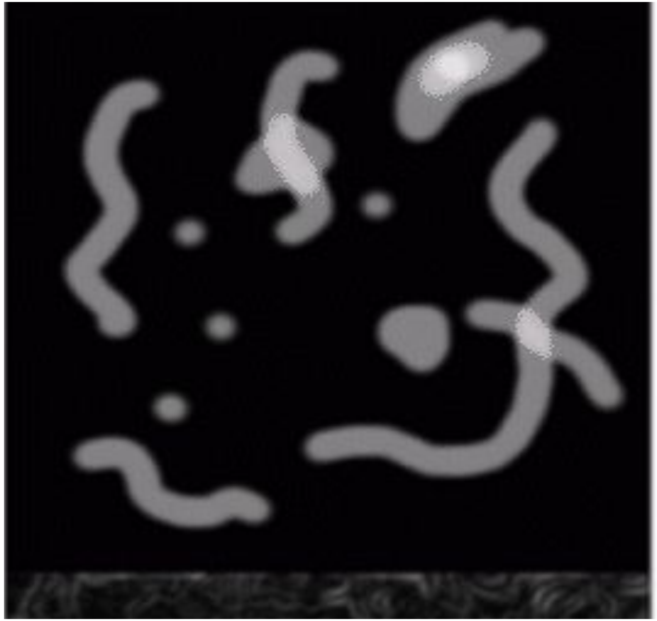
Swirly Bump Map

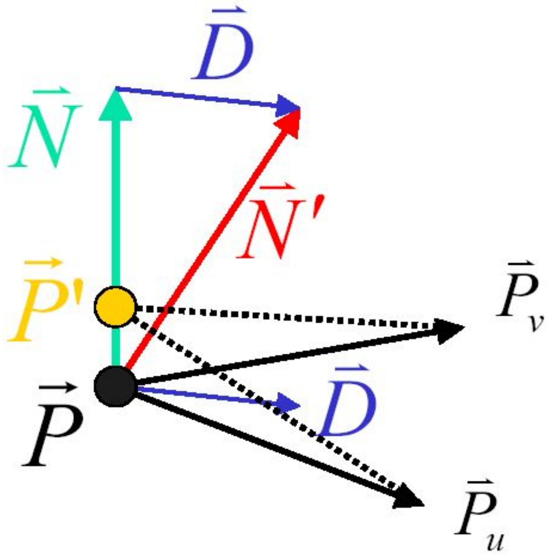
Sphere w/Diffuse Texture & Bump Map

Cheat normals to give illusion of ADDITIONAL/FAKE geometric detail

Bump Mapping

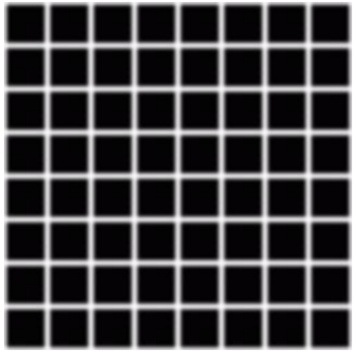
- Treat a greyscale texture as a single-valued height function
- Compute the normal from the partial derivatives in the texture





Another Bump Map Example

Cylinder w/Diffuse Texture Map



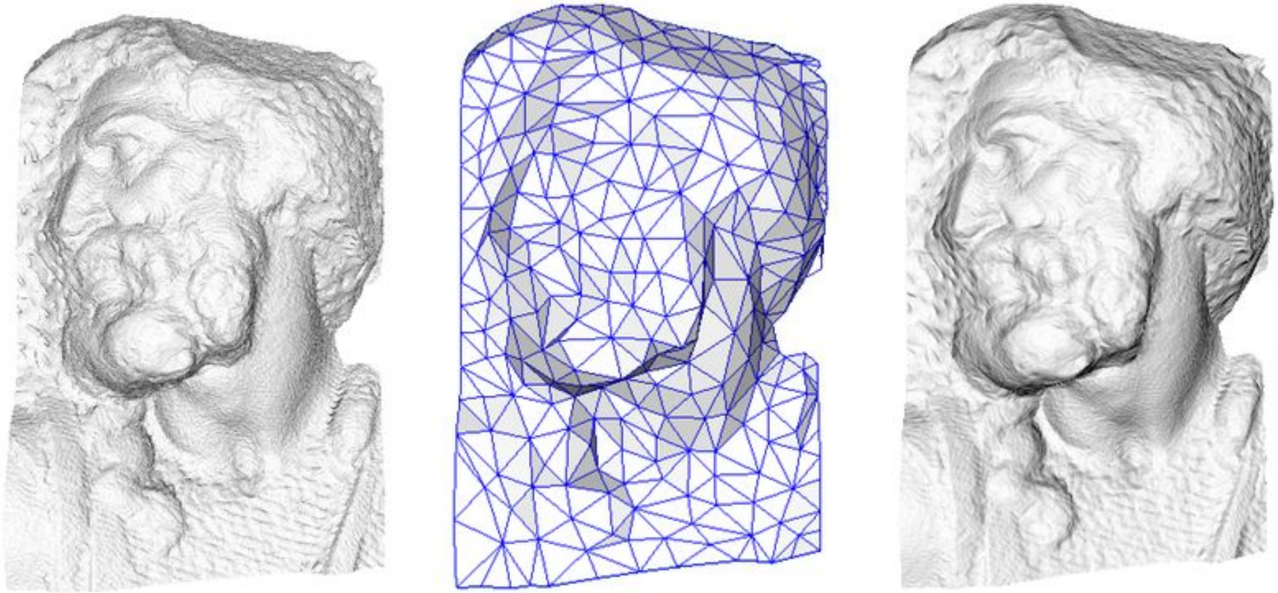
Bump Map

Cylinder w/Texture Map & Bump Map

Normal Mapping

http://en.wikipedia.org/wiki/File:
Normal_map_example.png

- Variation on Bump Mapping:
  Use an RGB texture to directly encode the normal



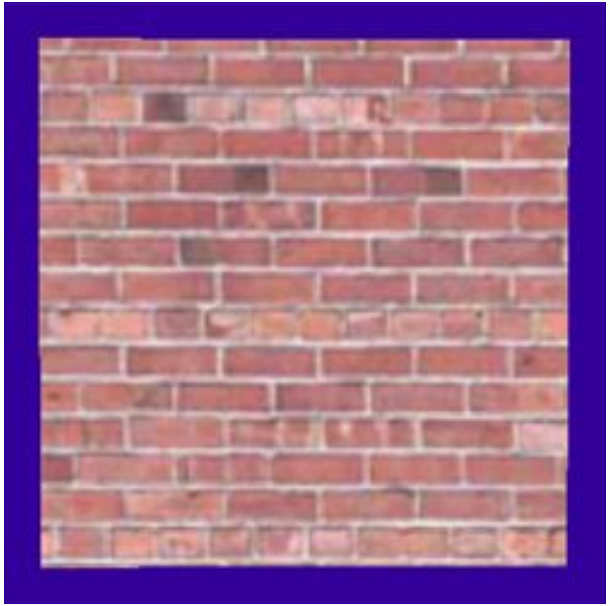
original mesh
4M triangles

simplified mesh
500 triangles

simplified mesh
and normal mapping
500 triangles

What's Missing?

- There are no bumps on the silhouette of a bump-mapped or normal-mapped object
- Bump/Normal maps don't allow self-occlusion or self-shadowing



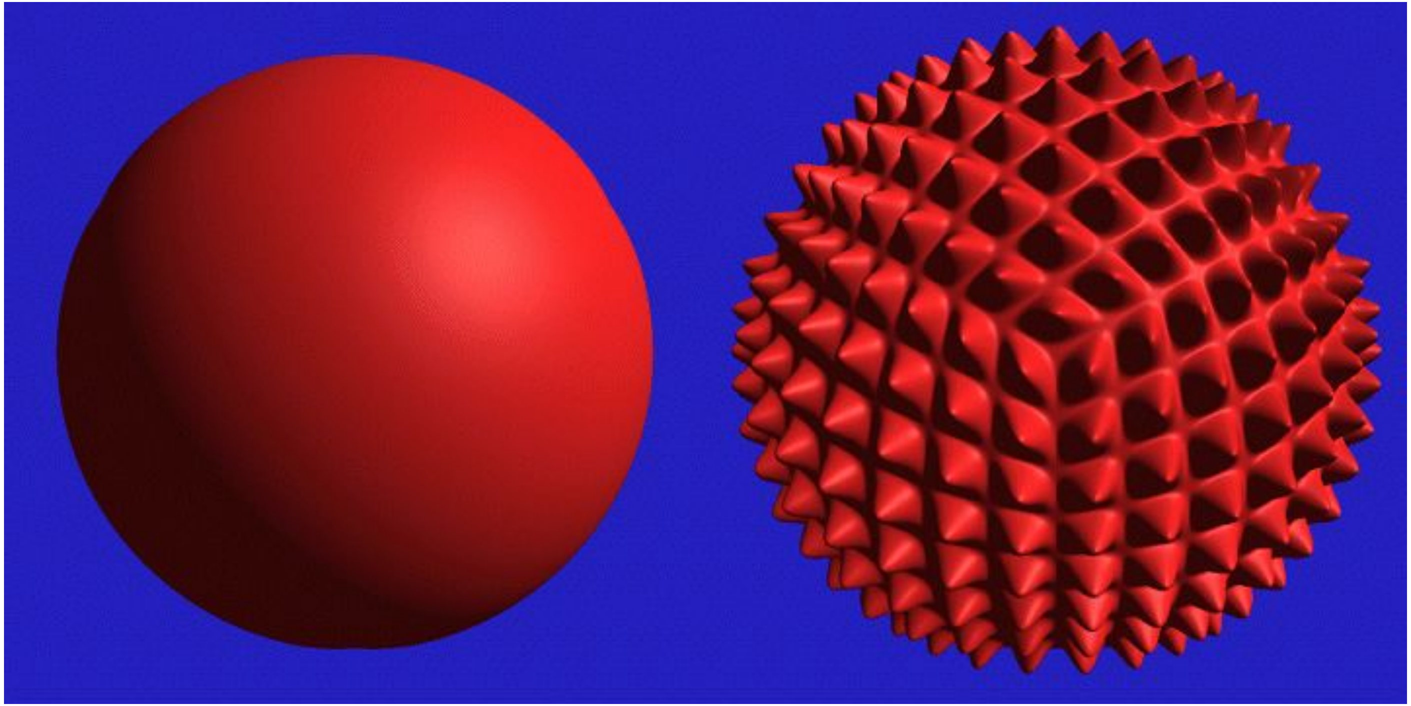
Today

- Texture Mapping & Other "Mapping" Techniques
  - Bump Mapping
  - Displacement Mapping
  - Environment Mapping
  - Light Mapping
  - Normal Mapping
  - Parallax Mapping
  - Parallax Occlusion Mapping
- Programmable Shader Examples
  - Modern Graphics Hardware
  - Per-Pixel Shading
- Procedural Textures & Modeling
- Papers for Today
- Papers for Next Time

Displacement Mapping

- Use the texture map to actually move the surface point
- The geometry must be displaced before visibility is determined

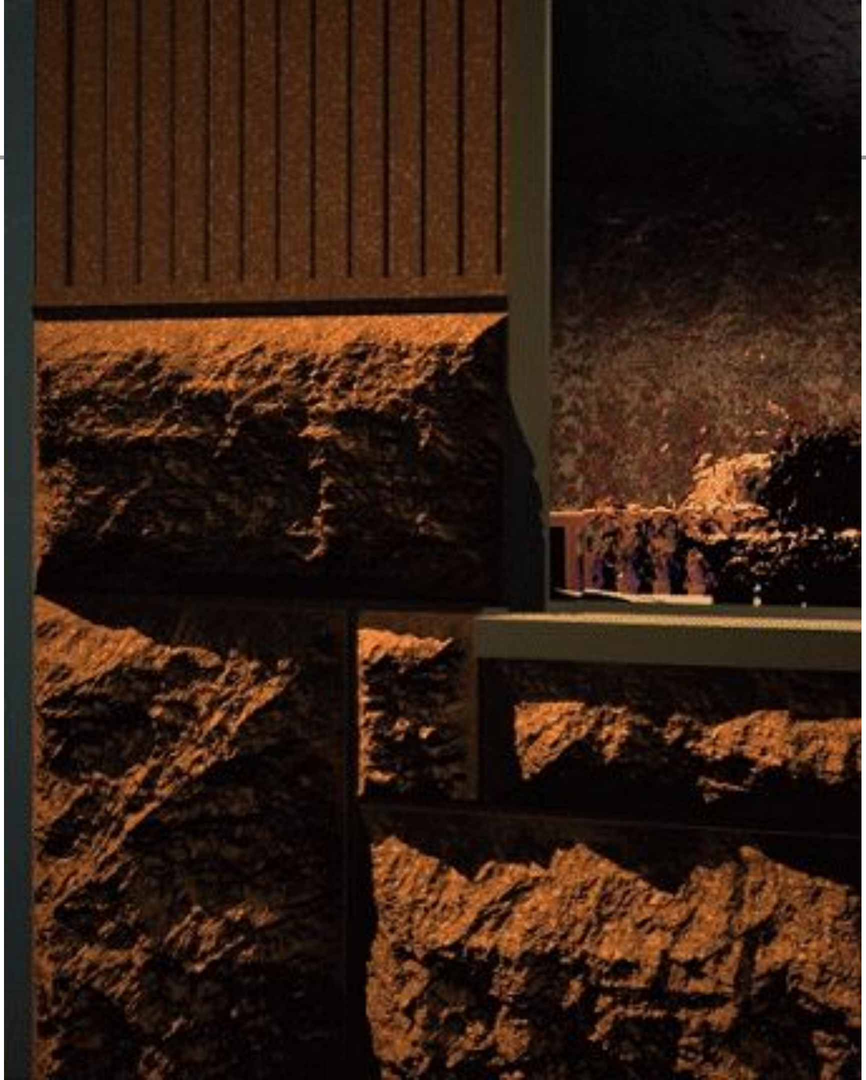
Originally a CPU-only, post-user-modeling step



Displacement Mapping

- “Geometry Caching for Ray-Tracing Displacement Maps” EGRW 1996. Pharr & Hanrahan

- *note the accurate and detailed shadows cast by the stones*



Procedural Displacement Mapping

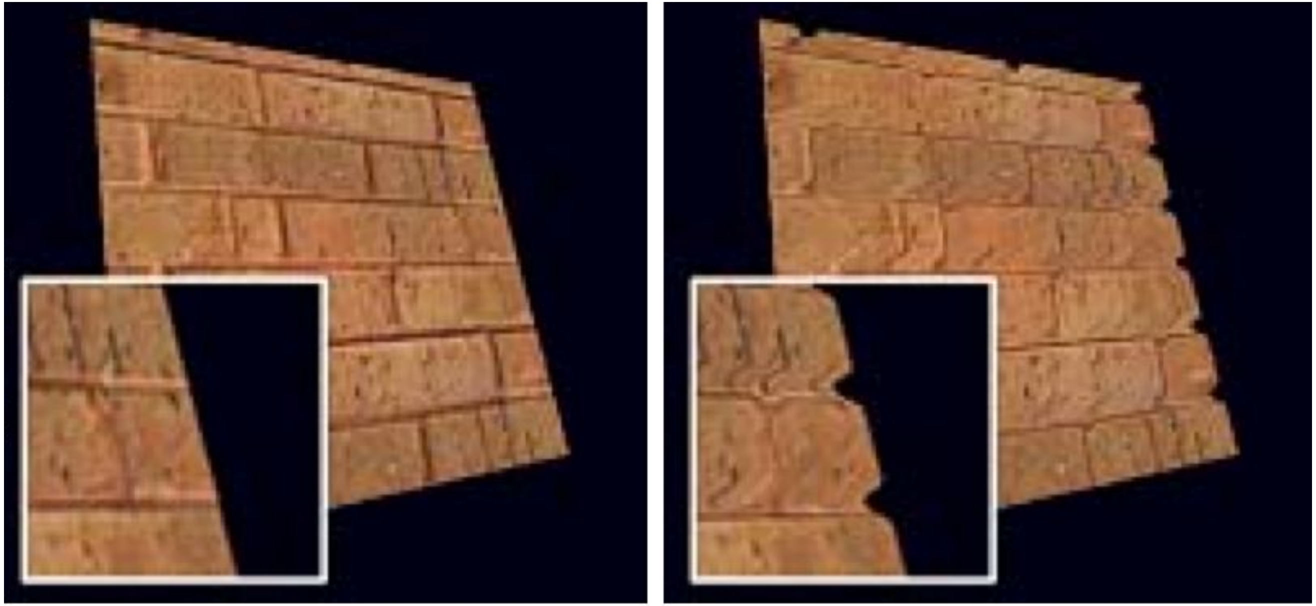
Ken Musgrave   www.kenmusgrave.com

Parallax Mapping

a.k.a. Offset Mapping or Virtual Displacement Mapping

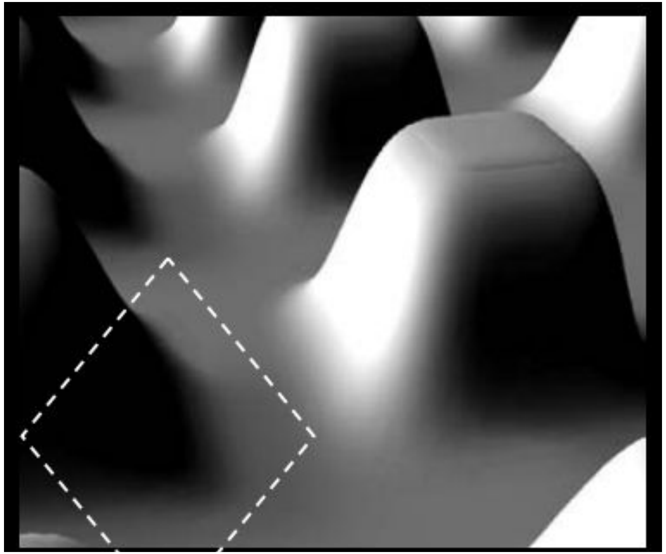
- Displace the texture coordinates for each pixel based on view angle and value of the height map at that point
- At steeper view-angles, texture coordinates are displaced more, giving *illusion of depth* due to parallax effects

"Detailed shape representation with parallax mapping", Kaneko et al. ICAT 2001



Parallax Occlusion Mapping

- Brawley & Tatarchuk 2004
- Per pixel ray tracing of the heightfield geometry
- Occlusions & soft shadows

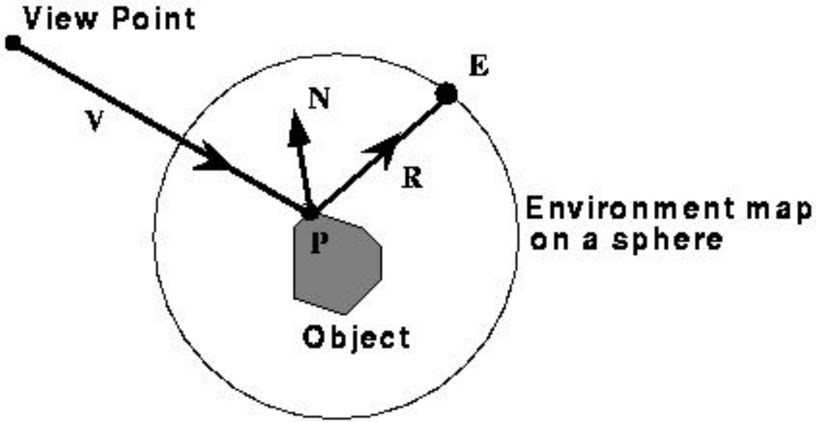
*http://developer.amd.com/media/gpu_assets/
Tatarchuk-ParallaxOcclusionMapping-Sketch-print.pdf*

Today

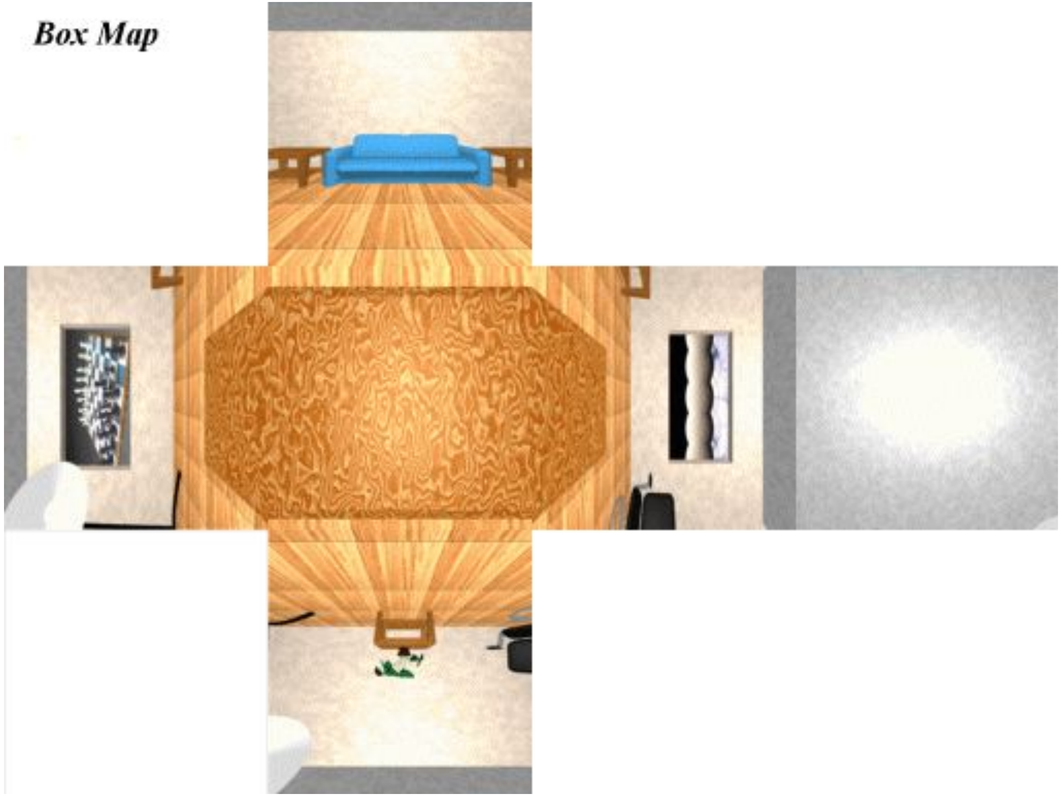
- Texture Mapping & Other "Mapping" Techniques
  - Bump Mapping
  - Displacement Mapping
  - Environment Mapping
  - Light Mapping
  - Normal Mapping
  - Parallax Mapping
  - Parallax Occlusion Mapping
- Programmable Shader Examples
  - Modern Graphics Hardware
  - Per-Pixel Shading
- Procedural Textures & Modeling
- Papers for Today
- Papers for Next Time

Environment Maps

- We can simulate reflections by using the direction of the reflected ray to index a spherical texture map at "infinity".
- Assumes that all reflected rays begin from the same point.



What's the Best Chart?



Environment Mapping Example

Terminator II

Texture Maps for Illumination

- Also called "Light Maps"

Quake

Questions?

Image by Henrik Wann Jensen

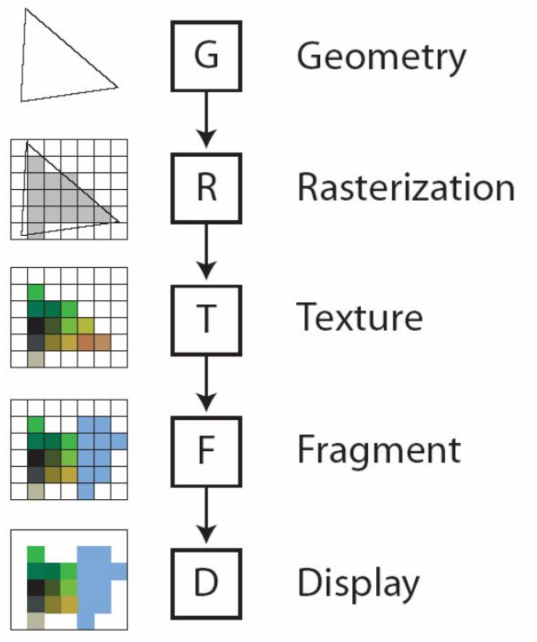
Environment map by Paul Debevec

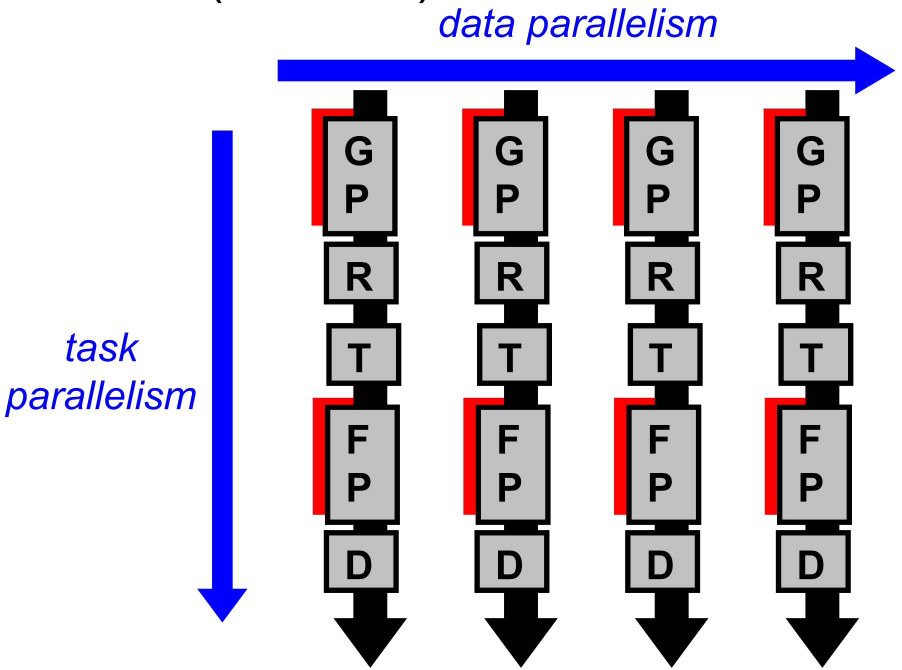
Today

- Texture Mapping & Other “Mapping” Techniques
  - Bump Mapping
  - Displacement Mapping
  - Environment Mapping
  - Light Mapping
  - Normal Mapping
  - Parallax Mapping
  - Parallax Occlusion Mapping
- Programmable Shader Examples
  - Modern Graphics Hardware
  - Per-Pixel Shading
- Procedural Textures & Modeling
- Papers for Today
- Papers for Next Time

Modern Graphics Hardware

- Increased parallelism
- Programmable geometry and pixel/fragment stages
- General-purpose computation on GPU (GPGPU)

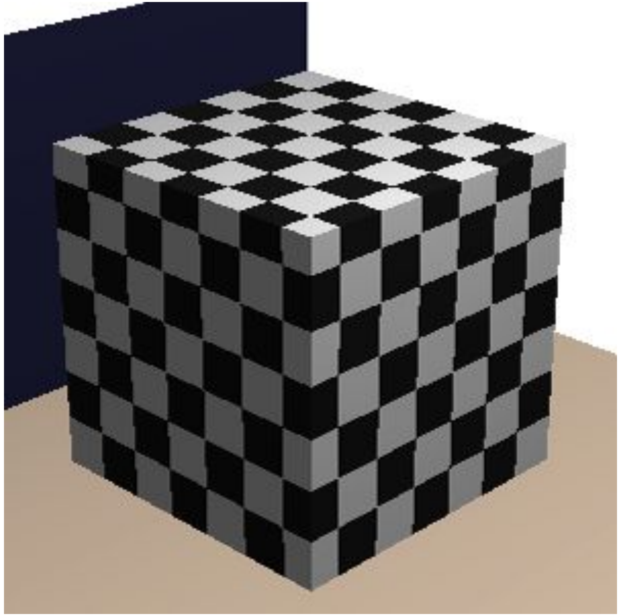




GLSL example: hw4_shader.vs

hw4_shader.vs

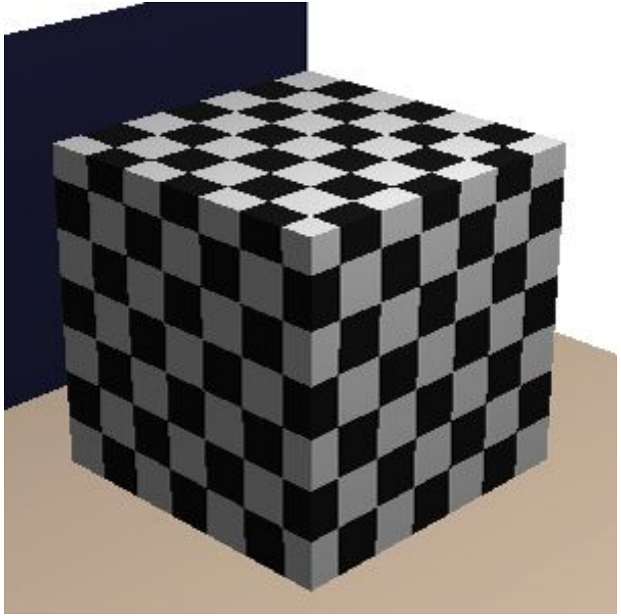
```
#version 330 core

// Input vertex data, different for all executions of this shader.
layout(location = 0) in vec3 vertexPosition_modelspace;
layout(location = 1) in vec3 vertexNormal_modelspace;
layout(location = 2) in vec3 vertexColor;

// Output data
out vec3 vertexPosition_worldspace;
out vec3 vertexNormal_worldspace;
out vec3 vertexNormal_cameraspace;
out vec3 EyeDirection_cameraspace;
out vec3 myColor;

// Values that stay constant for the whole mesh.
uniform mat4 MVP;
uniform mat4 V;
uniform mat4 M;
uniform vec3 LightPosition_worldspace;

void main(){

  // Output position of the vertex, in clip space : MVP * position
  gl_Position =  MVP * vec4(vertexPosition_modelspace,1);

  // Position of the vertex, in worldspace : M * position
  vertexPosition_worldspace = (M * vec4(vertexPosition_modelspace,1)).xyz;

  // Vector that goes from the vertex to the camera, in camera space.
  // In camera space, the camera is at the origin (0,0,0).
  vec3 vertexPosition_cameraspace = ( V * M * vec4(vertexPosition_modelspace,1)).xyz;

  EyeDirection_cameraspace = vec3(0,0,0) - vertexPosition_cameraspace;

  vertexNormal_worldspace = normalize (M * vec4(vertexNormal_modelspace,0)).xyz;

  // pass color to the fragment shader
  myColor = vertexColor;
}
```



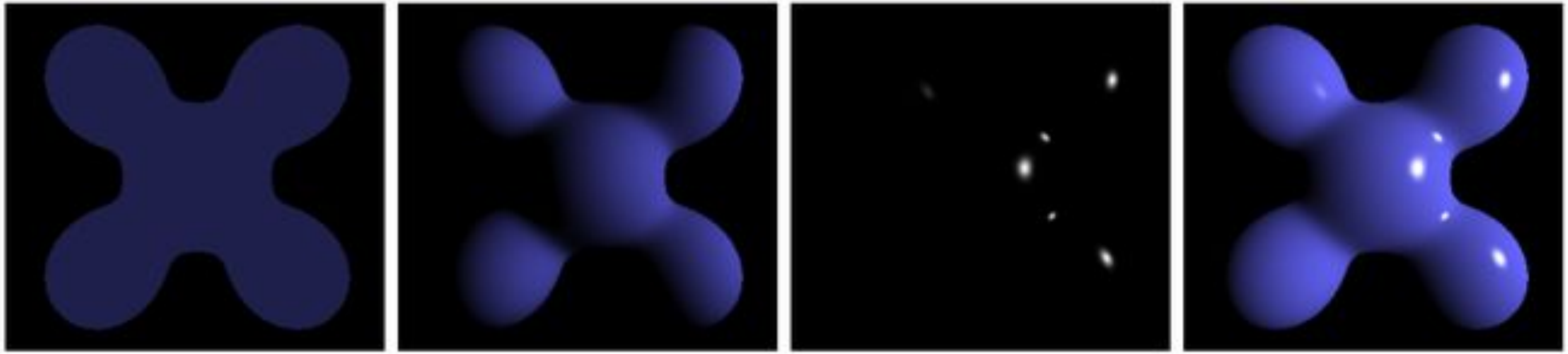
GLSL example: hw4_shader.fs

hw4_shader_checkerboard.fs

```
in vec3 vertexNormal_worldspace;

// Ouput data
out vec3 color;

// Values that stay constant for the whole mesh.
uniform vec3 LightPosition_worldspace;
uniform int colormode;
uniform int whichshader;

// -----------------------------------------------
// a shader for a black & white checkerboard
vec3 checkerboard(vec3 pos) {
  // determine the parity of this point in the 3D checkerboard
  int count = 0;
  if (mod(pos.x,0.3)> 0.15) count++;
  if (mod(pos.y,0.3)> 0.15) count++;
  if (mod(pos.z,0.3)> 0.15) count++;
  if (count == 1 || count == 3) {
    return vec3(0.1,0.1,0.1);
  } else {
    return vec3(1,1,1);
  }
}

// -----------------------------------------------
void main(){

  vec3 LightColor = vec3(1,1,1);
  float LightPower = 4.0f;

  // surface normal
  vec3 surface_normal =  vertexNormal_worldspace;

  // Material properties
  vec3 MaterialDiffuseColor = myColor;
  if (whichshader == 1) {
    MaterialDiffuseColor = checkerboard(vertexPosition_worldspace);
  } else if (whichshader == 2) {
    vec3 normal2;
    MaterialDiffuseColor = orange(vertexPosition_worldspace,surface_normal);
  } else if (whichshader == 3) {
    MaterialDiffuseColor = wood(vertexPosition_worldspace,surface_normal);
  }
```



Phong Reflection/Lighting Model



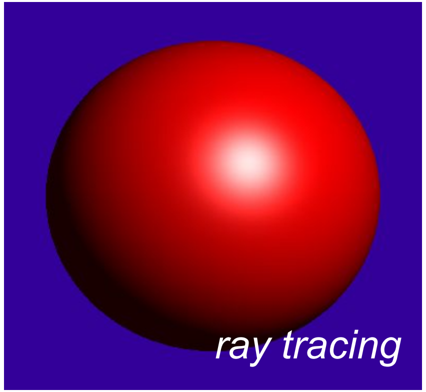
Ambient + Diffuse + Specular = Phong Reflection

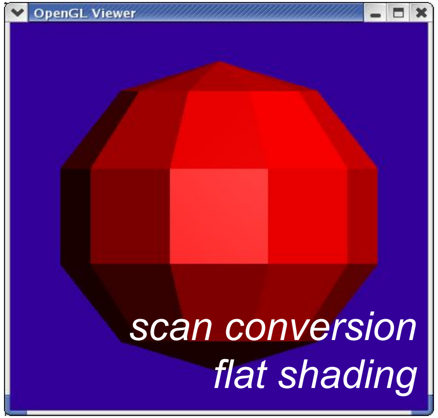
$$I_p = k_a i_a + \sum_{m \in \text{lights}} \left(k_d (\hat{L}_m \cdot \hat{N}) i_{m,d} + k_s (\hat{R}_m \cdot \hat{V})^{\alpha} i_{m,s} \right)$$

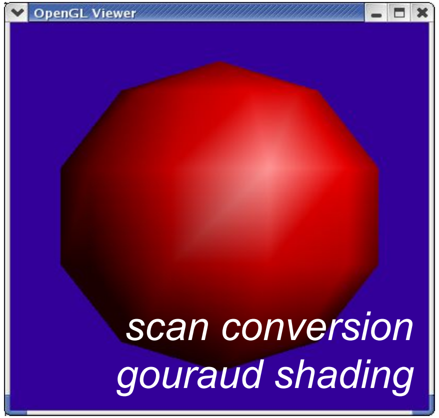
Color & Normal Interpolation

- It's easy in OpenGL to specify different colors and/or normals at the vertices of triangles:
- Why is this useful?

Originally, all we could afford to do in hardware was interpolate colors

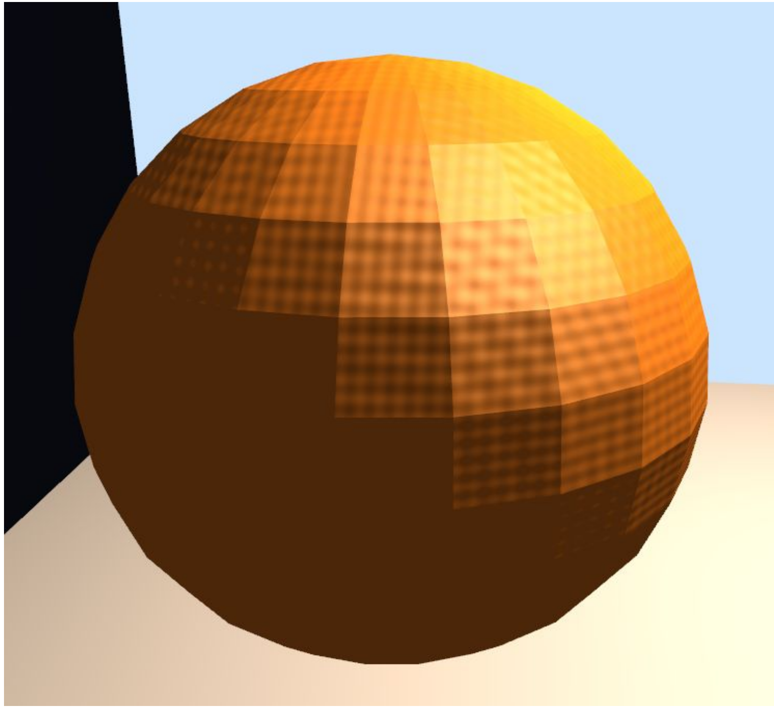


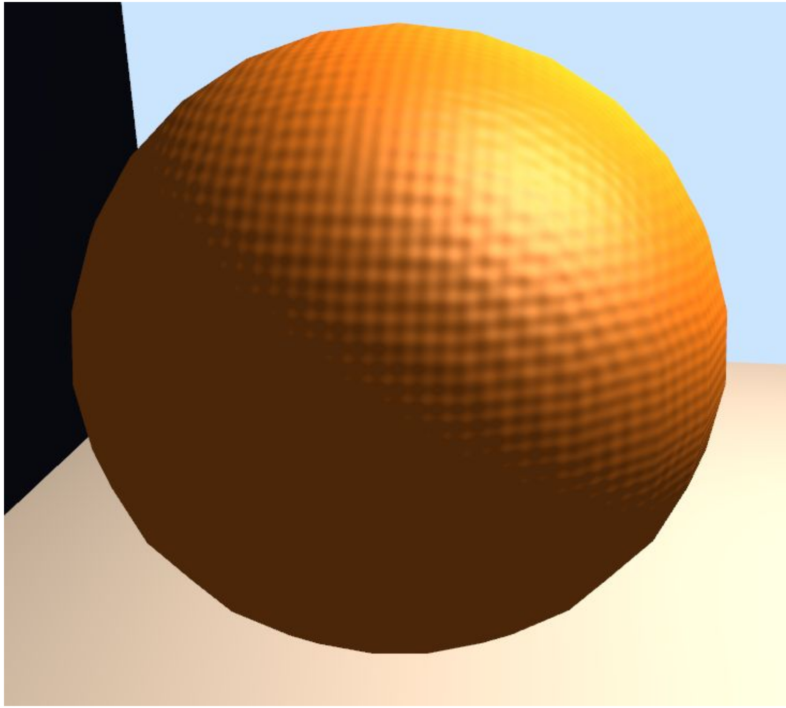




Per-Pixel Shading!

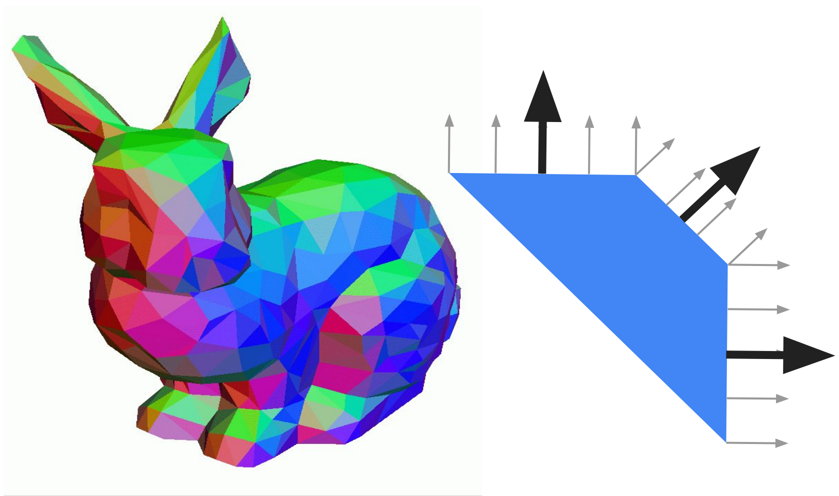
- We are not just interpolating the color
- Phong Reflection/Lighting can be calculated per pixel, not just per vertex

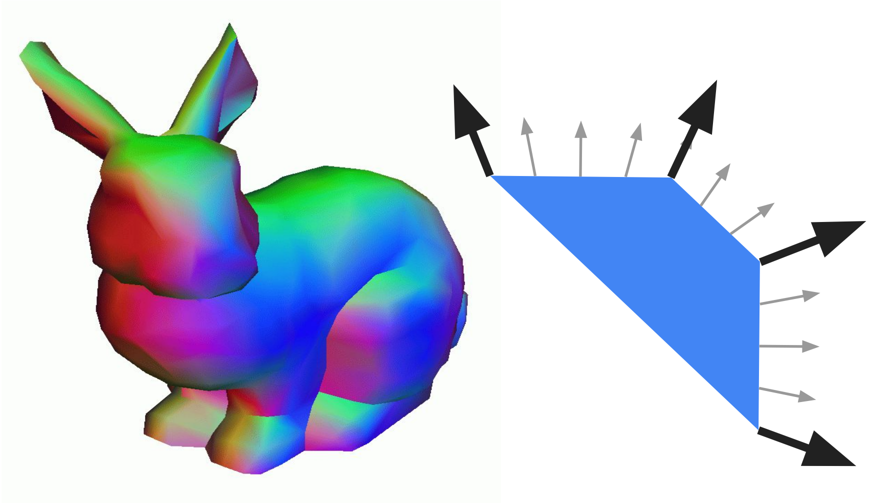




Phong Normal Interpolation (Not Phong *Shading*)

- *Interpolate the average vertex normals* across the face and compute *per-pixel shading*
    - Normals should be re-normalized (ensure length=1)
- Before shaders, per-pixel shading was not possible in hardware (Gouraud shading is actually a decent substitute!)

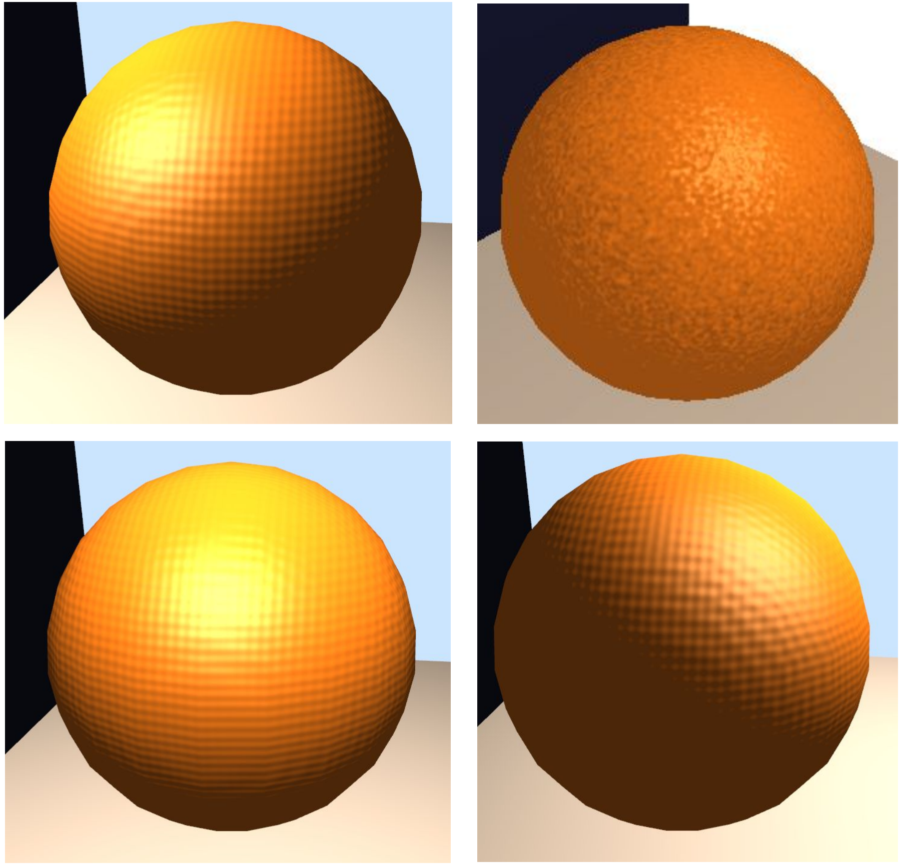




Another GLSL example: orange.vs

Emacs@tony.dyn.cs.rpi.edu

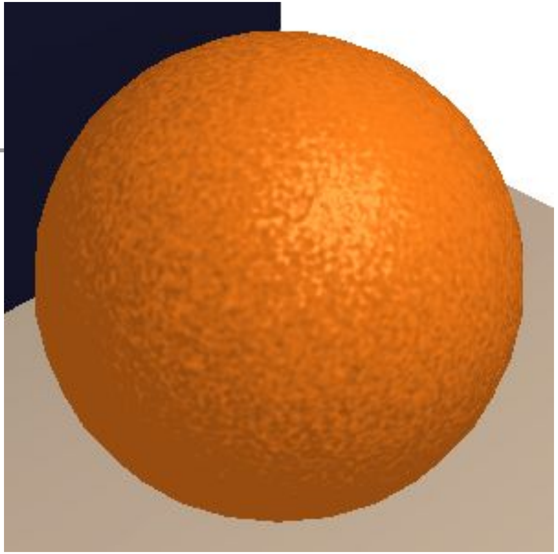
```
varying vec3 normal;
varying vec3 position_eyespace;
varying vec3 position_worldspace;

// a shader that looks like orange peel

void main(void) {

  // the fragment shader requires both the world space position (for
  // consistent bump mapping) & eyespace position (for the phong
  // specular highlight)
  position_eyespace = vec3(gl_ModelViewMatrix * gl_Vertex);
  position_worldspace = gl_Vertex.xyz;

  // pass along the normal
  normal = normalize(gl_NormalMatrix * gl_Normal);

  gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
 }
```

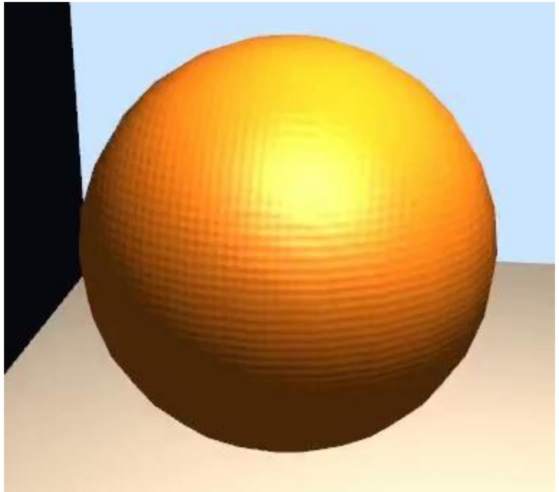


Another GLSL example: orange.fs

Emacs@tony.dyn.cs.rpi.edu

```
varying vec3 normal;
varying vec3 position_eyespace;
varying vec3 position_worldspace;

// a shader that looks like orange peel

void main (void) {

  // the base color is orange!
  vec3 color = vec3(1.0,0.5,0.1);

  // high frequency noise added to the normal for the bump map
  vec3 normal2 = normalize(normal+0.4*noise3(70.0*position_worldspace));

  // direction to the light
  vec3 light = normalize(gl_LightSource[1].position.xyz - position_eyespace);
  // direction to the viewer
  vec3 eye_vector = normalize(-position_eyespace);
  // ideal specular reflection
  vec3 reflected_vector = normalize(-reflect(light,normal2));

  // basic phong lighting
  float ambient = 0.6;
  float diffuse = 0.4*max(dot(normal2,light),0.0);
  float specular = 0.2 * pow(max(dot(reflected_vector,eye_vector),0.0),10.0);
  vec3 white = vec3(1.0,1.0,1.0);
  color = ambient*color + diffuse*color + specular*white;
  gl_FragColor = vec4 (color, 1.0);
}
```

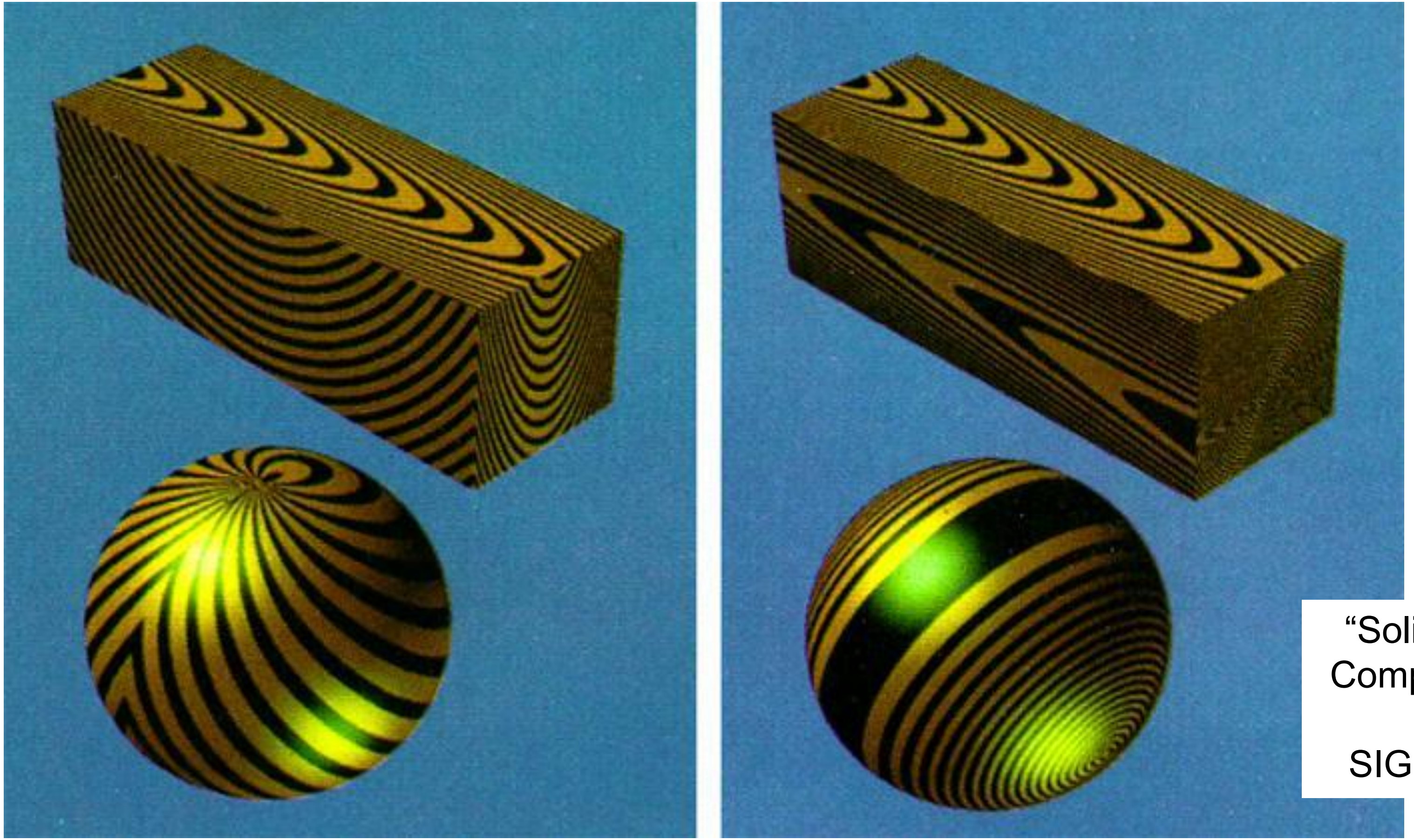




Today

- Texture Mapping & Other “Mapping” Techniques
  - Bump Mapping
  - Displacement Mapping
  - Environment Mapping
  - Light Mapping
  - Normal Mapping
  - Parallax Mapping
  - Parallax Occlusion Mapping
- Programmable Shader Examples
  - Modern Graphics Hardware
  - Per-Pixel Shading
- Procedural Textures & Modeling
- Papers for Today
- Papers for Next Time

Texture Map vs. Solid Texture



"Solid Texturing of Complex Surfaces", Peachey, SIGGRAPH 1985

Procedural Textures

f (x,y,z) → color

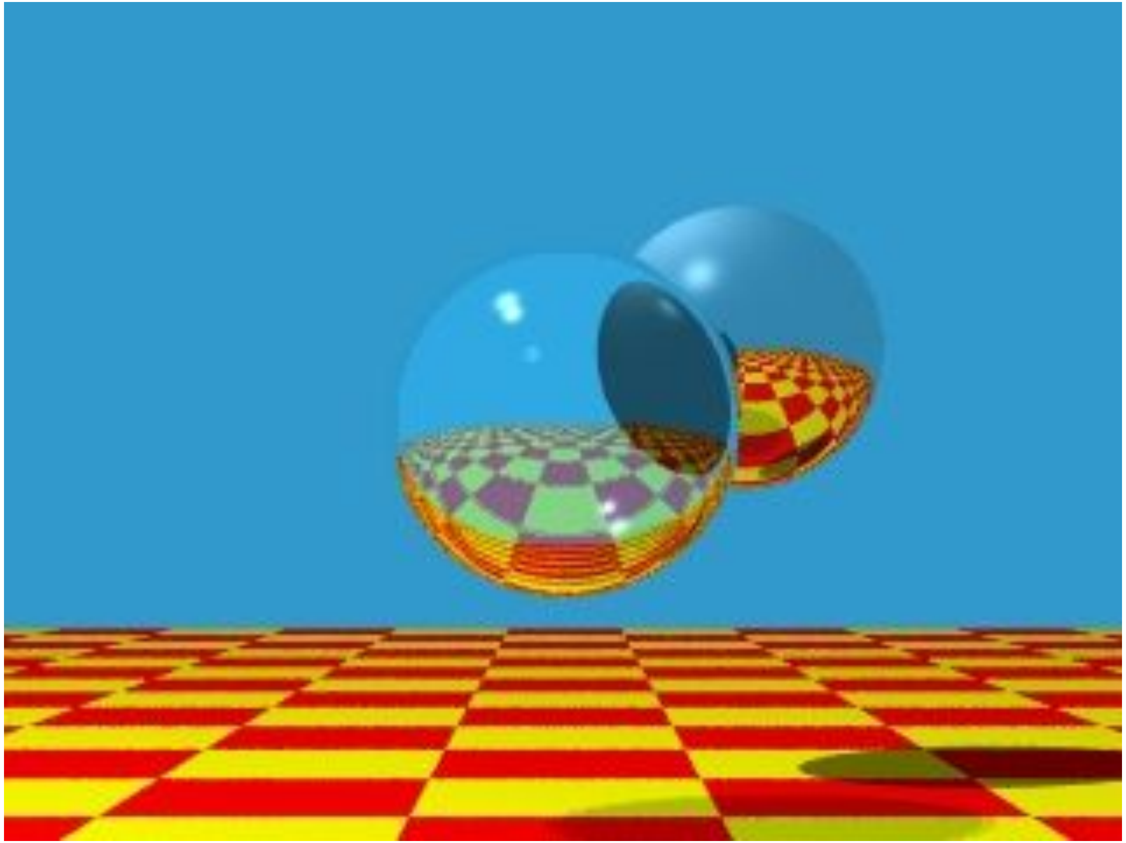
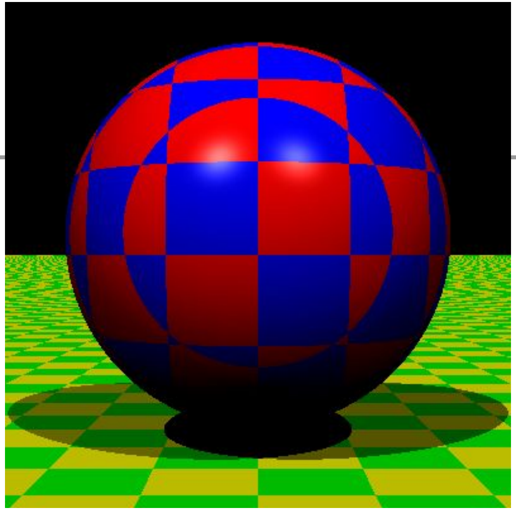


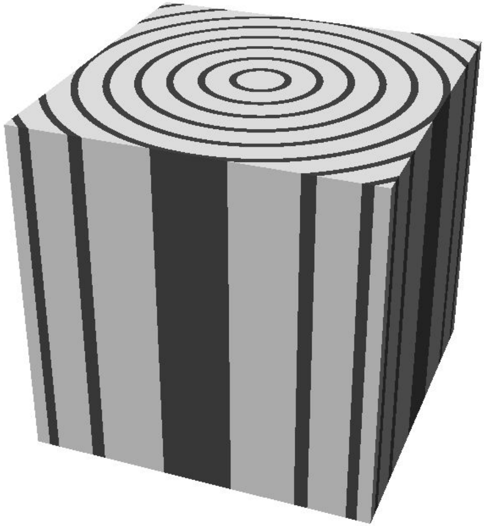
Image by Turner Whitted

Procedural Textures

- Advantages:
  - easy to implement in ray tracer
  - more compact than texture maps (especially for solid textures)
  - infinite resolution

- Disadvantages
  - non-intuitive
  - difficult to match existing texture



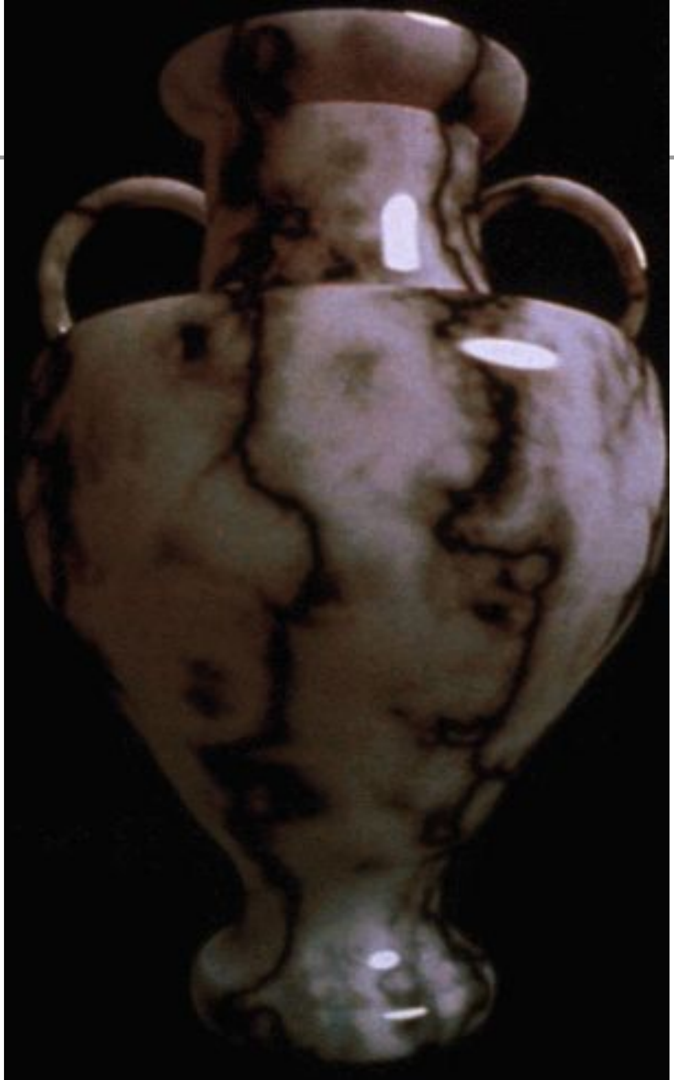


Today

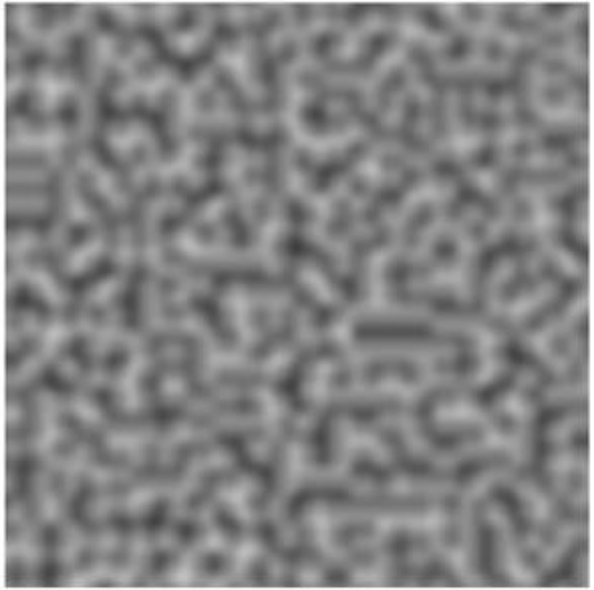
- Texture Mapping & Other "Mapping" Techniques
  - Bump Mapping
  - Displacement Mapping
  - Environment Mapping
  - Light Mapping
  - Normal Mapping
  - Parallax Mapping
  - Parallax Occlusion Mapping
- Programmable Shader Examples
  - Modern Graphics Hardware
  - Per-Pixel Shading
- Procedural Textures & Modeling
- Papers for Today
- Papers for Next Time

Reading for Today

- "An Image Synthesizer", Perlin, SIGGRAPH 1985 – *and* –
- "Improving Noise", Perlin, SIGGRAPH 2002



Perlin Noise

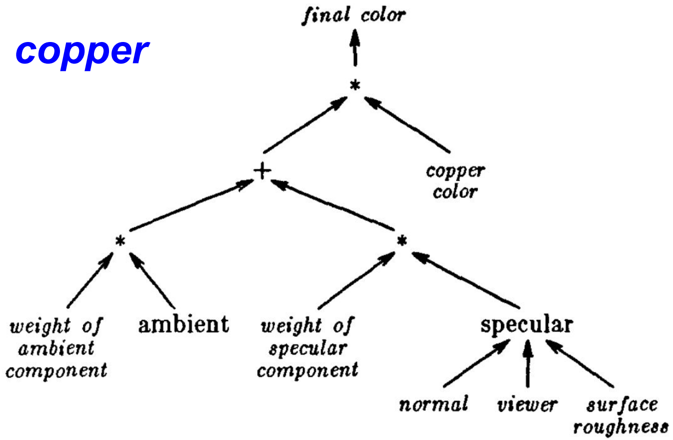


- Properties:
  - Looks “random”, but is deterministic (always returns the same answer for a specific coordinate)
  - Small memory footprint & fast to compute
  - Known amplitude & frequency
  - Smooth interpolation when zoomed in
- Can be combined/layered:
  - Add multiple noise functions w/ different frequencies and amplitudes
  - Simple arithmetic operations (thresholding, sine waves, etc.)

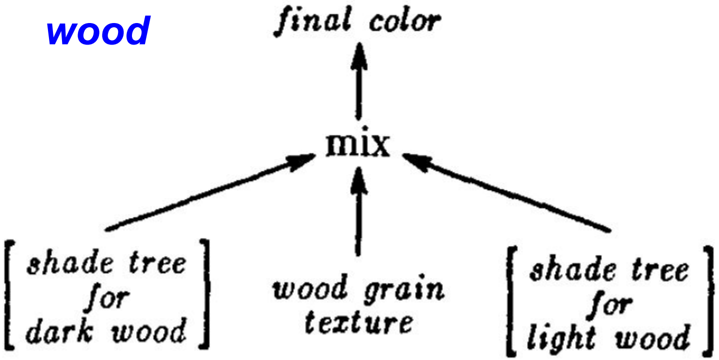
- Clear motivation & methodology,
  - Even distribution, prevent clumping, avoid struggle with UV mapping
  - Focus on efficiency, minimize memory/storage, unlimited resolution
- Use of a hash function for noise was interesting
  - Complex patterns from simple math
- Versatile - 1D or 2D or 3D (or higher), lots of example uses (excessive?)
- What is “realistic”? Not actually a scientifically provable thing!
  - How do we know its good? It “looks good”?
- More casual paper style
- 2 page paper fixing a flaw in original algorithm is interesting (unsure/unclear about the actual necessity / difference in the output)

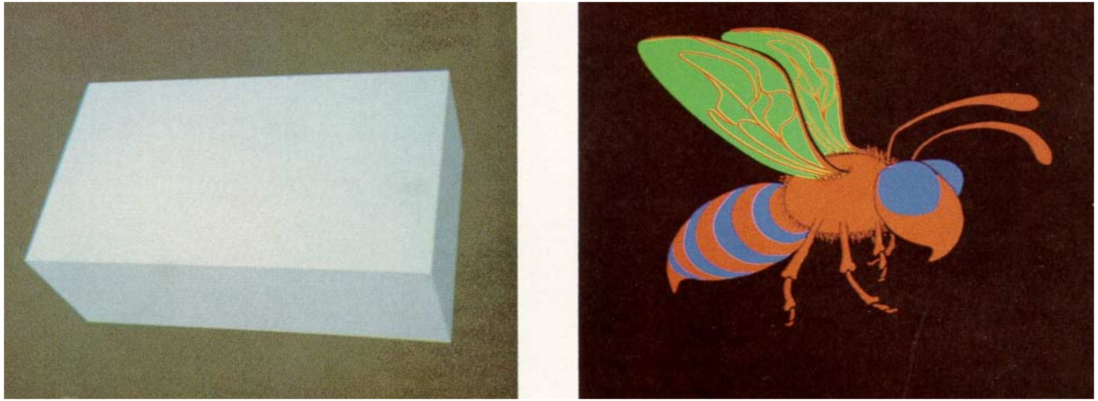
“Shade Trees”, Cook, SIGGRAPH 1984

copper

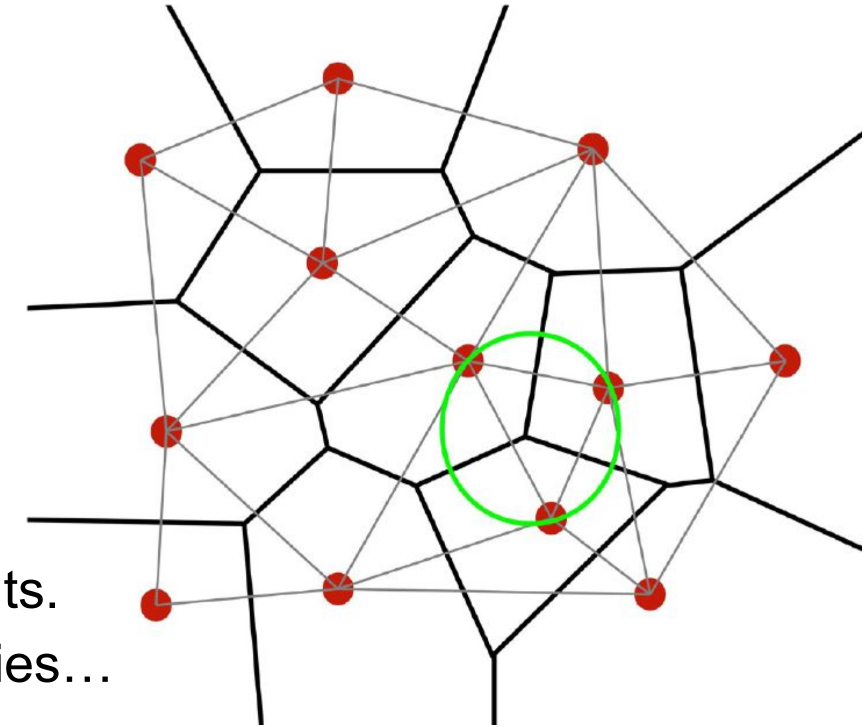


wood

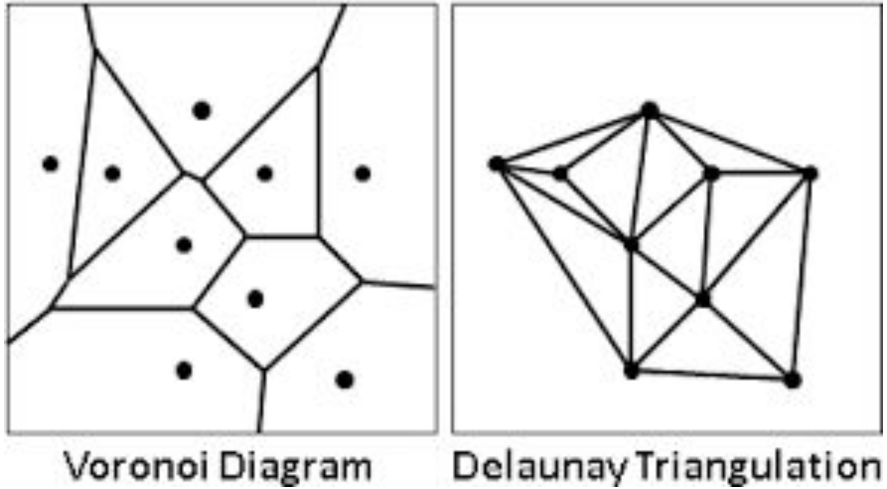


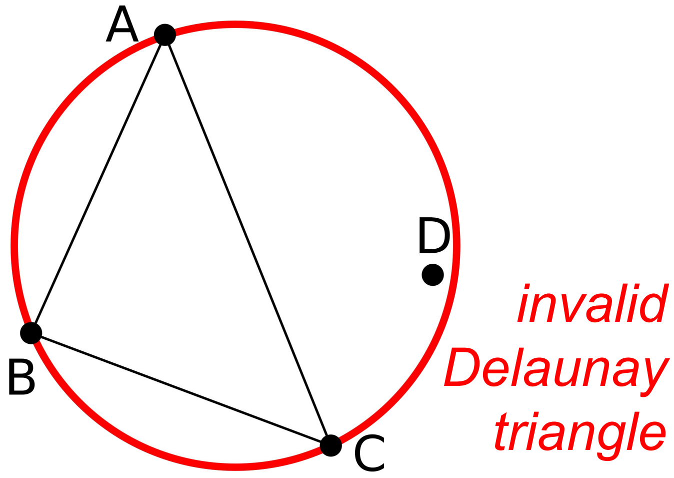


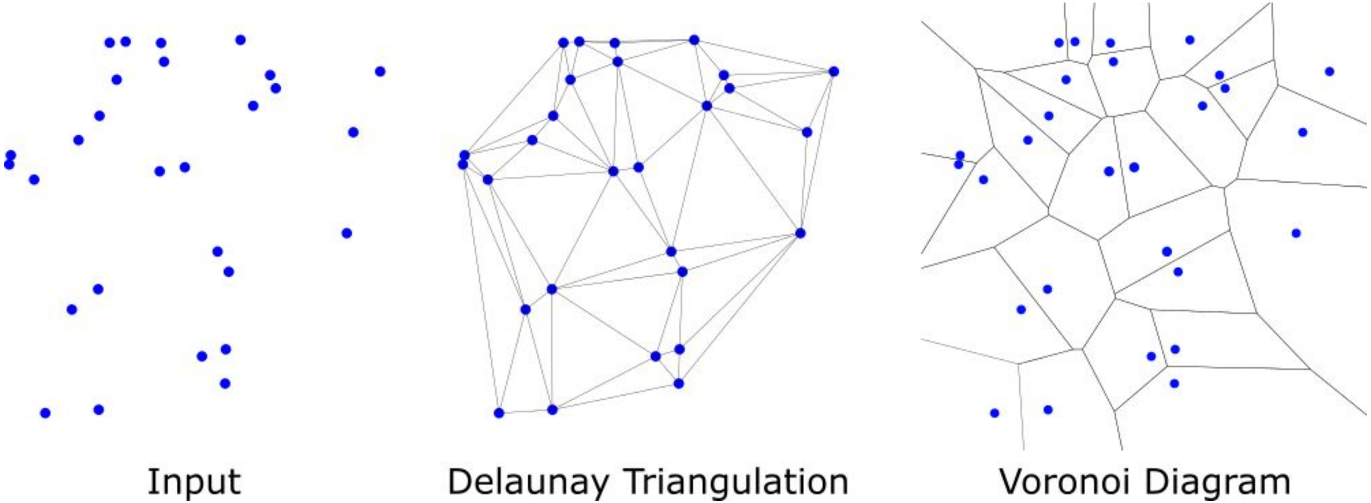
- Grey: Delaunay Triangulation
  - “Best” triangulation of the red dots (most equilateral)
  - A specific triangle is in the Delaunay Triangle *if and only if* the circle defined by those 3 points *does not* contain any other red dot
  - Note: Well defined for random points. Points on a uniform grid will have ties…



- **Black: Voronoi Diagram**
  - Each cell is the set of all points in the plane that claim that cell’s red dot as the closest
- *Note: The black edges perpendicularly bisect the grey edges*

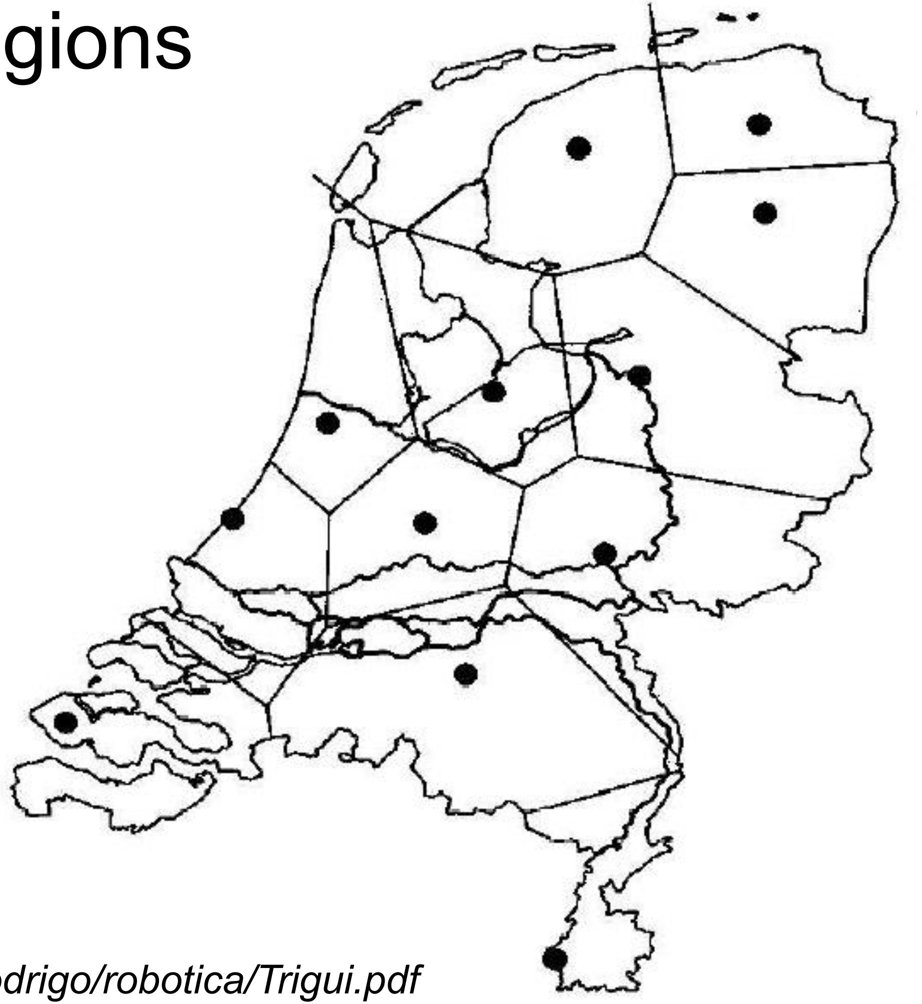

Voronoi Diagram
Delaunay Triangulation


A
B
C
D
invalid
Delaunay
triangle


Input
Delaunay Triangulation
Voronoi Diagram

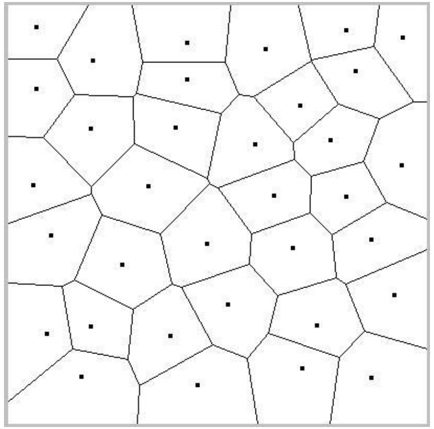
Voronoi Diagram/Cells/Regions

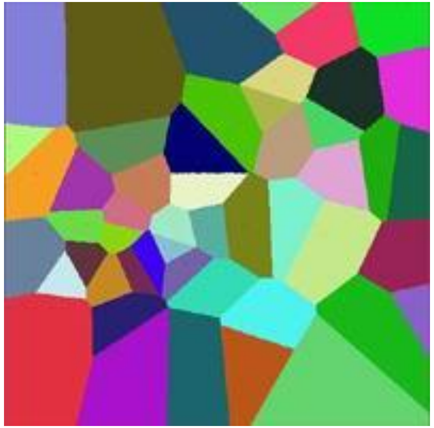
- How to re-district the Netherlands into provinces so that everyone reports to the closest capital
- Cell edges are the perpendicular bisectors of nearby points
- 2D or 3D
- Supports efficient *Nearest Neighbor* queries



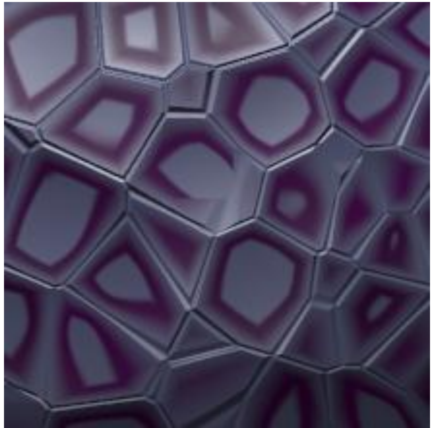
http://ccc.inaoep.mx/~rodrigo/robotica/Trigui.pdf

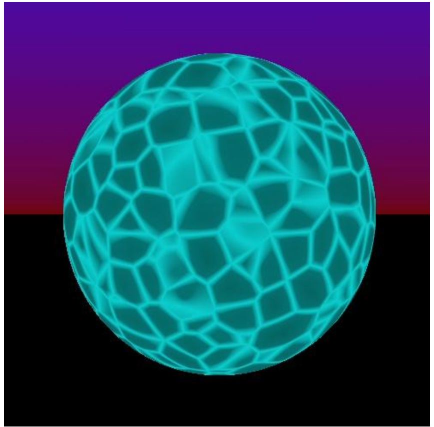
Cellular Textures (using a Voronoi Diagram!)

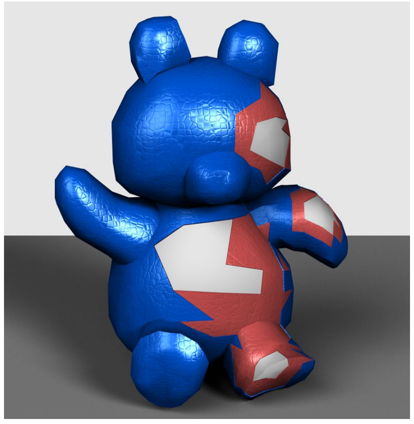




"A Cellular Texture Basis Function", Worley, SIGGRAPH 1996
www.worley.com







Questions?

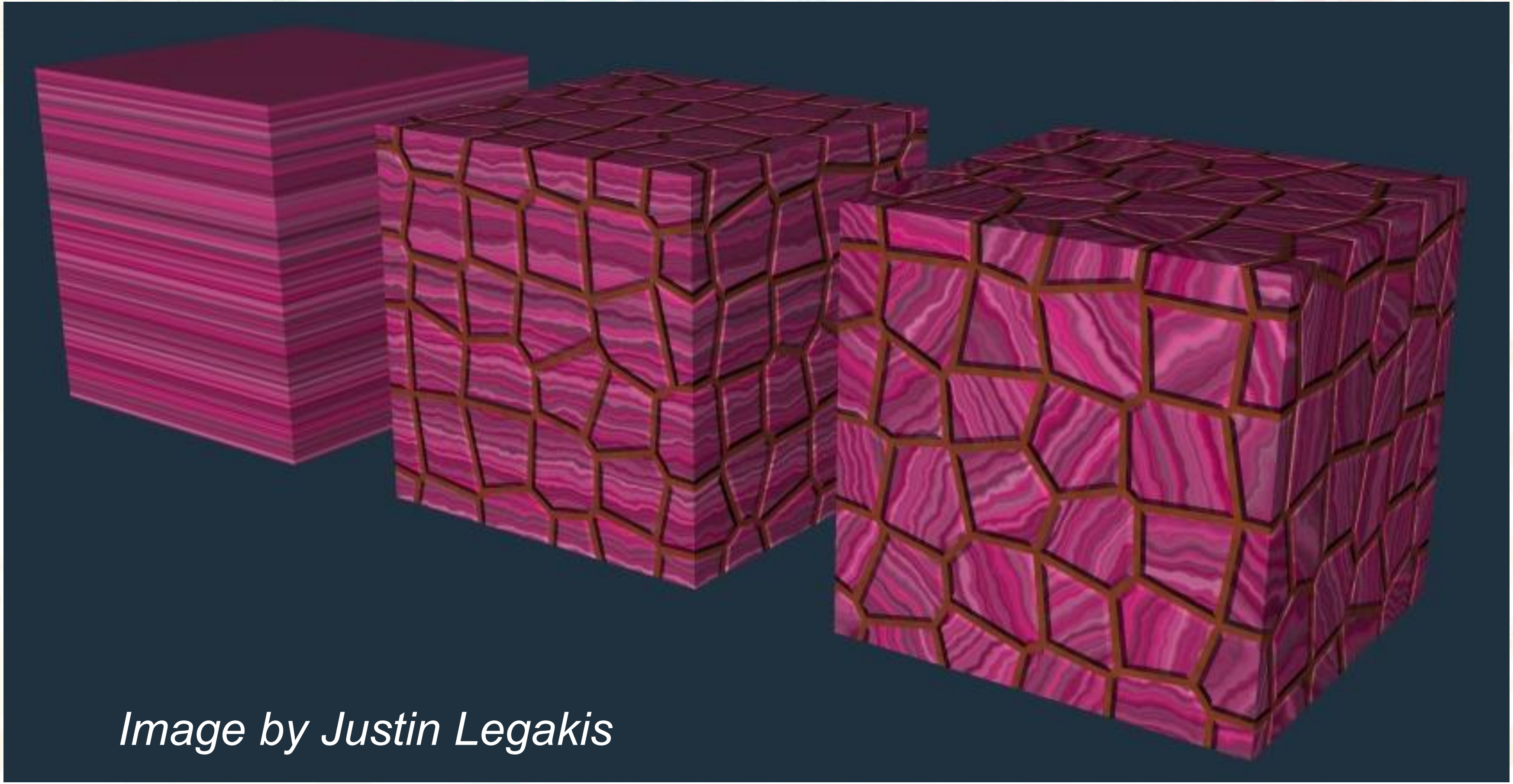
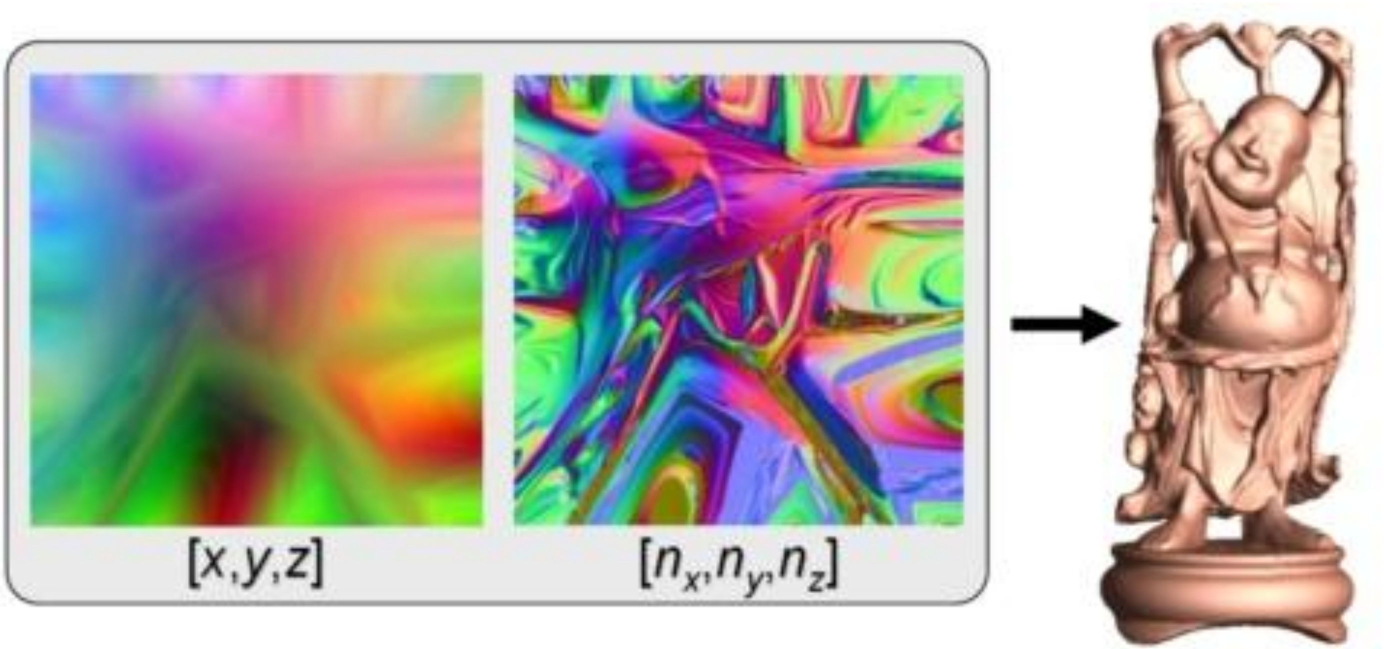


Image by Justin Legakis

Optional Reading for Today

- "Geometry Images", Gu, Gortler, & Hoppe, SIGGRAPH 2002
- 3D shape is unrolled/flattened/stretched into a square image.
- Stored using existing image formats and compression methods.



Optional Reading for Today

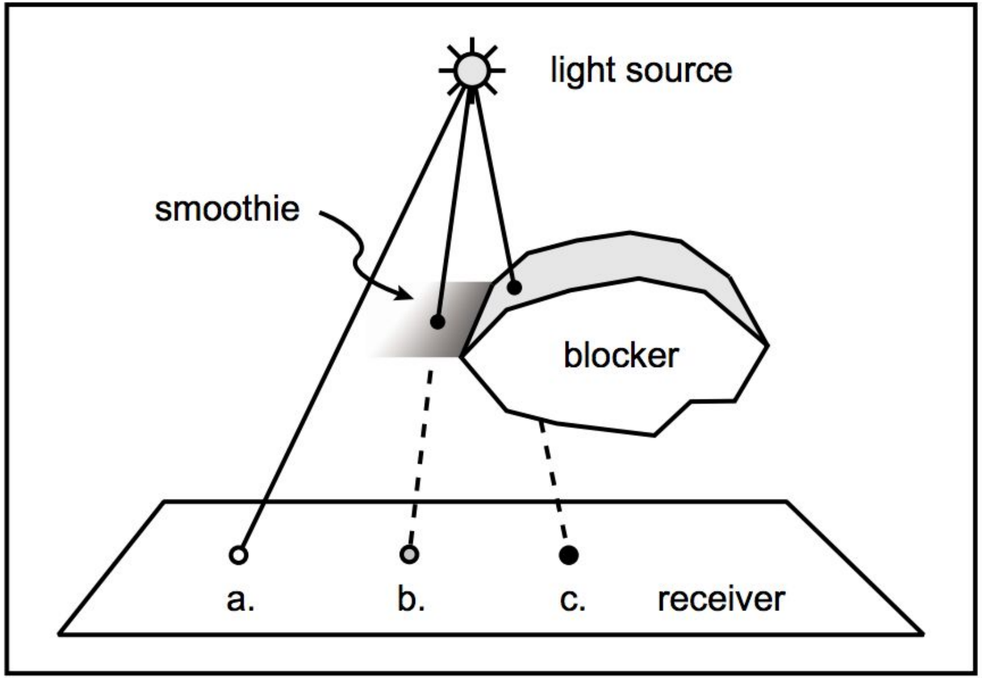
- “Hardware-Accelerated Global Illumination by Image Space Photon Mapping” McGuire & Luebke, HPG 2009

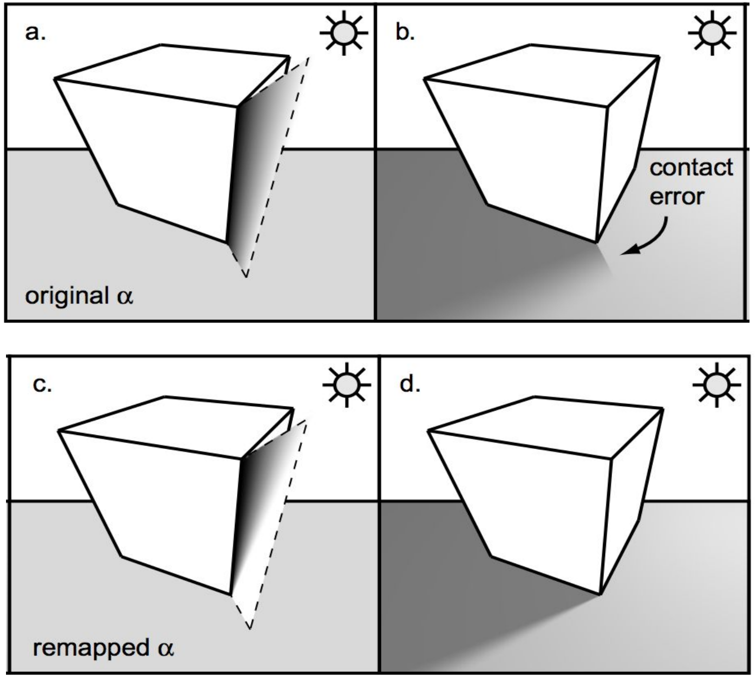
Direct Illumination Only | Direct + Constant Ambient | Image Space Photon Mapping

Figure 1: Image-space photon mapping can compute global illumination at interactive rates for scenes with multiple lights, caustics, shadows, and complex BSDFs. This scene renders at 26 Hz at 1920×1080. (Indirect and ambient intensity are amplified for comparison in this image.)

Reading for Last Time

- "Rendering Fake Soft Shadows with Smoothies", Chan & Durand, EGSR 2003

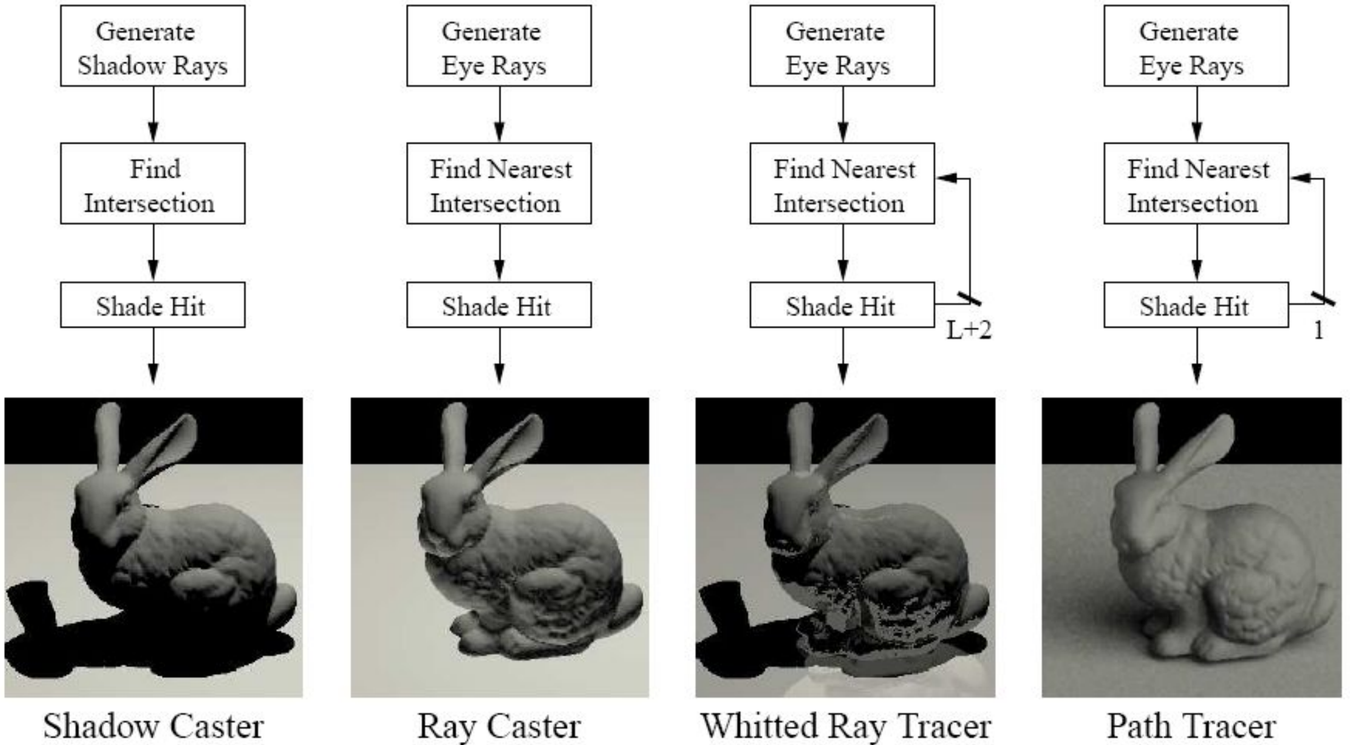




- Interesting to read how hardware capabilities have evolved over years
    - Do researchers plan/think ahead about algorithms for future hardware
- creative/unexpected solution
- Seems expensive for a scene with moving objects (must recompute silhouette edges)
- Limitations on specific shape of light source
- Just because an algorithm is a ‘hack’ doesn’t mean its without use/purpose - for the right scene & application its great

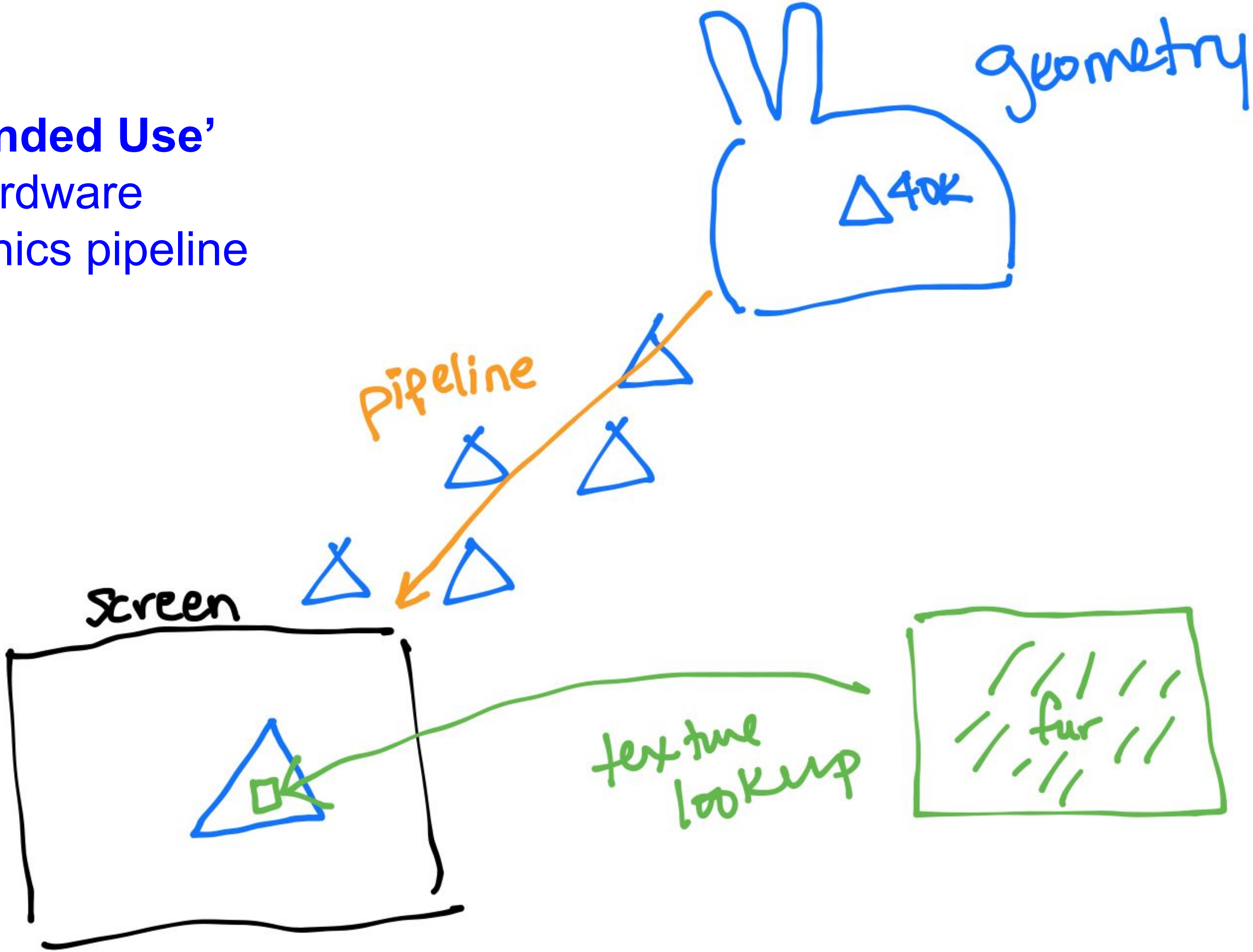
Reading for Last Time

- "Ray Tracing on Programmable Graphics Hardware", Purcell, Buck, Mark, & Hanrahan SIGGRAPH 2002

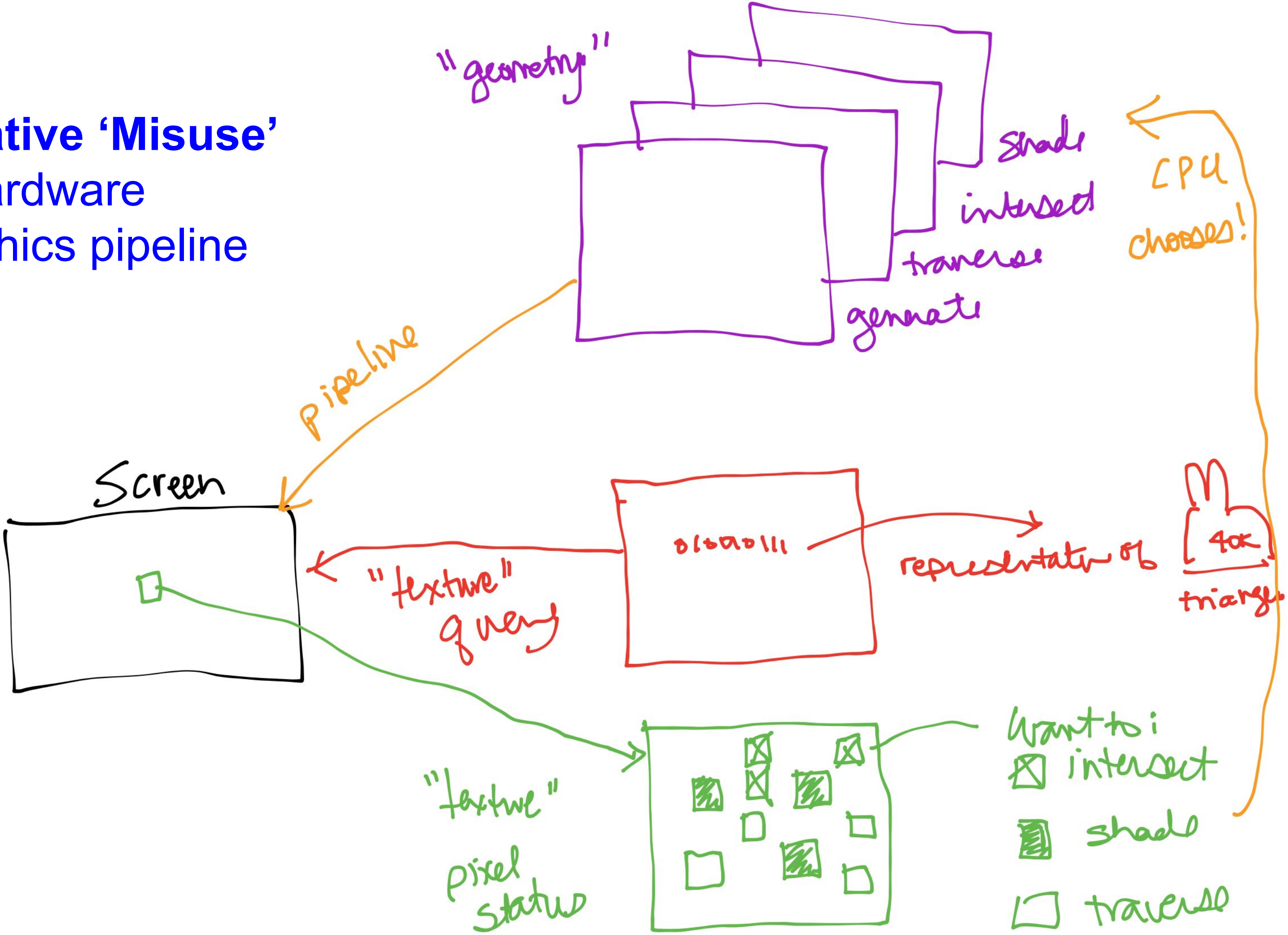


- In order to write best software, you must understand the hardware. In order to create best hardware, you must understand the software.
- How long were they waiting for this hardware? When did they first imagine that this would be possible with upcoming hardware?
- Interesting how the evolution of graphics hardware: fixed function → programmable → now returning to fixed function/non programmable
- Moore’s law, variation in relative development rates of GPU vs CPU

‘Intended Use’
of hardware
graphics pipeline



Creative 'Misuse'
of hardware
graphics pipeline

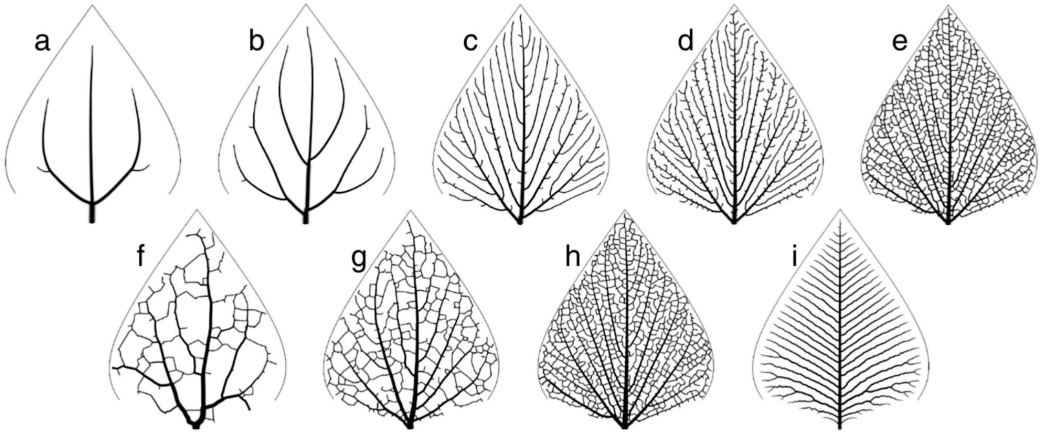


Today

- Texture Mapping & Other “Mapping” Techniques
  - Bump Mapping
  - Displacement Mapping
  - Environment Mapping
  - Light Mapping
  - Normal Mapping
  - Parallax Mapping
  - Parallax Occlusion Mapping
- Programmable Shader Examples
  - Modern Graphics Hardware
  - Per-Pixel Shading
- Procedural Textures & Modeling
- Papers for Today
- Papers for Next Time

Papers for Next Time *(pick one)*

- “Modeling and visualization of leaf venation patterns”, Runions, Fuhrer, Lane, Federl, Roggan-Lagan, & Prusinkiewicz, 2007.



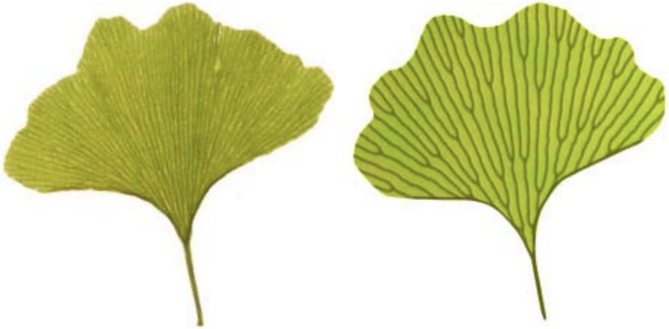


Figure 10: A photograph (left) and a rendered model of ginkgo venation (right).

Figure 11: A photograph(left) and a rendered model of lady’s mantle venation (right).

Papers for Next Time *(pick one)*

- “Procedural Modeling of Cities”, Parish & Müller, SIGGRAPH 2001