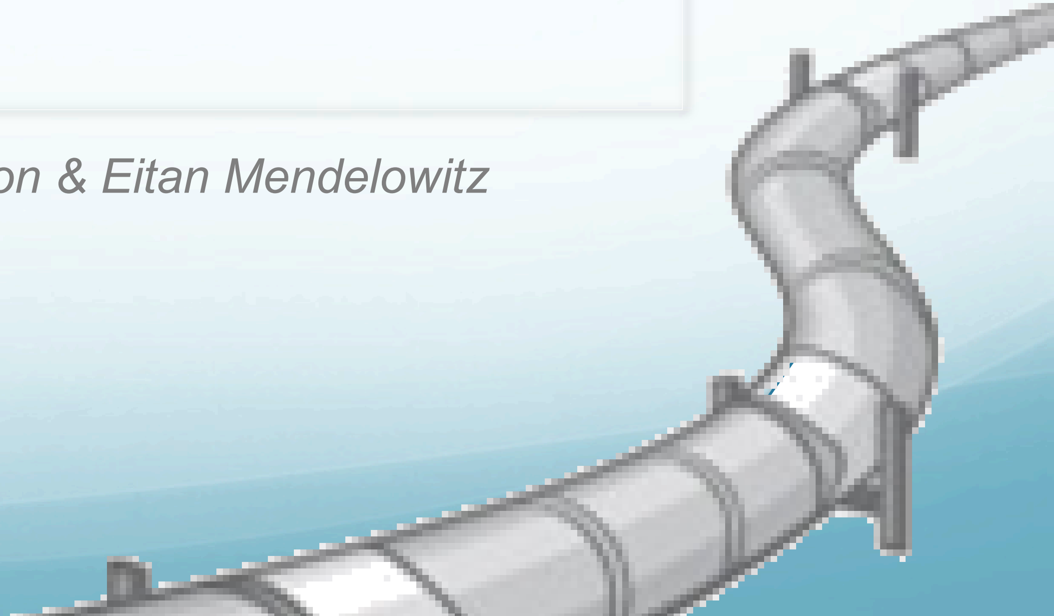
CSC 240 Computer Graphics
Graphics Pipeline & Texture Mapping

Nick Howe
Smith College

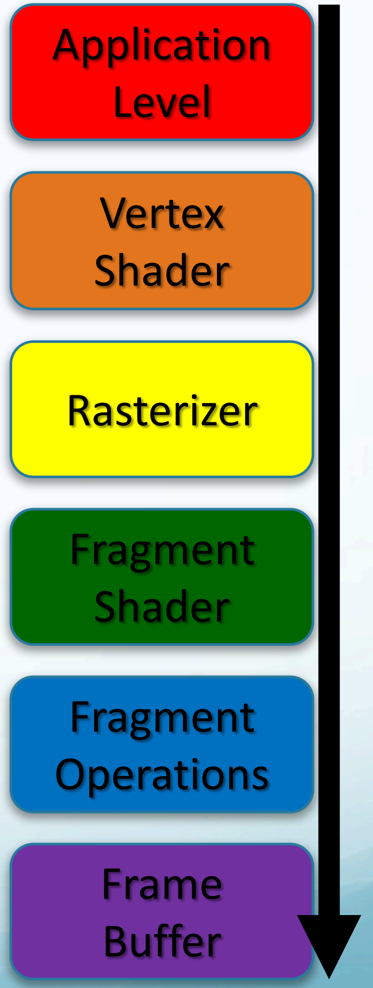
Some slides & content courtesy Sara Mathieson & Eitan Mendelowitz



Graphics Pipeline

Refers to the sequence of steps that takes code to image

- Application: Sets up 3D scene models
- Vertex shader: projects points to 2D image
- Rasterizer: breaks polygons into pixels
- Fragment shader: colors polygons
- Fragment operations: ordering (z buffering, etc.)
- Frame buffer: ready for display



Graphics Pipeline

We can choose among a few standard options for the fragment shader

Advanced graphics programmers write their own vertex and fragment shaders

Our programs are at this level, setting up cameras, lights, & geometry

Our choice of lighting, material, geometry, and texture all determine what happens here

Camera settings determine what happens here

Geometry and camera position determine what is visible

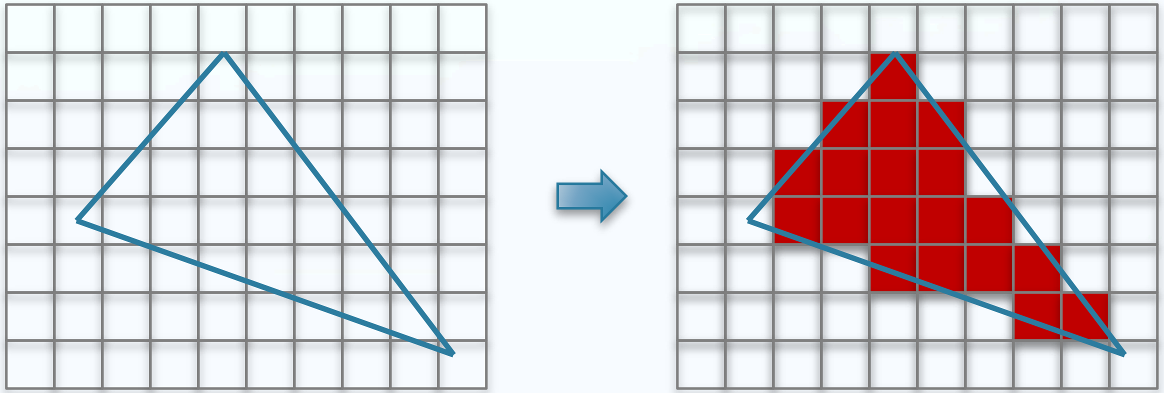
- Application Level
- Vertex Shader
- Rasterizer
- Fragment Shader
- Fragment Operations
- Frame Buffer

Fragment Shading

Photo by **Sharon McCutcheon** from **Pexels**

Flat Shading

After the vertex shader and rasterizer, we have polygon boundaries on a grid.



The job of a fragment shader is to fill in the pixels inside the boundaries

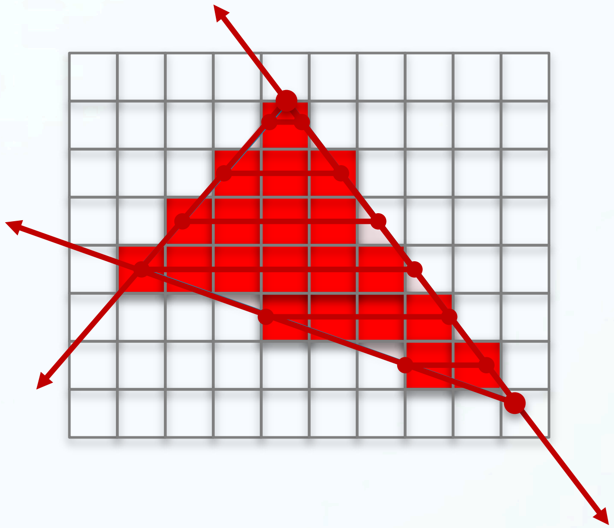
Flat shading is the simplest type of shading – all pixels are colored the same

We could use our old fill algorithm, but there are more efficient ways

Triangle Shading

Every triangle can be split into two half-triangles with horizontal baselines:

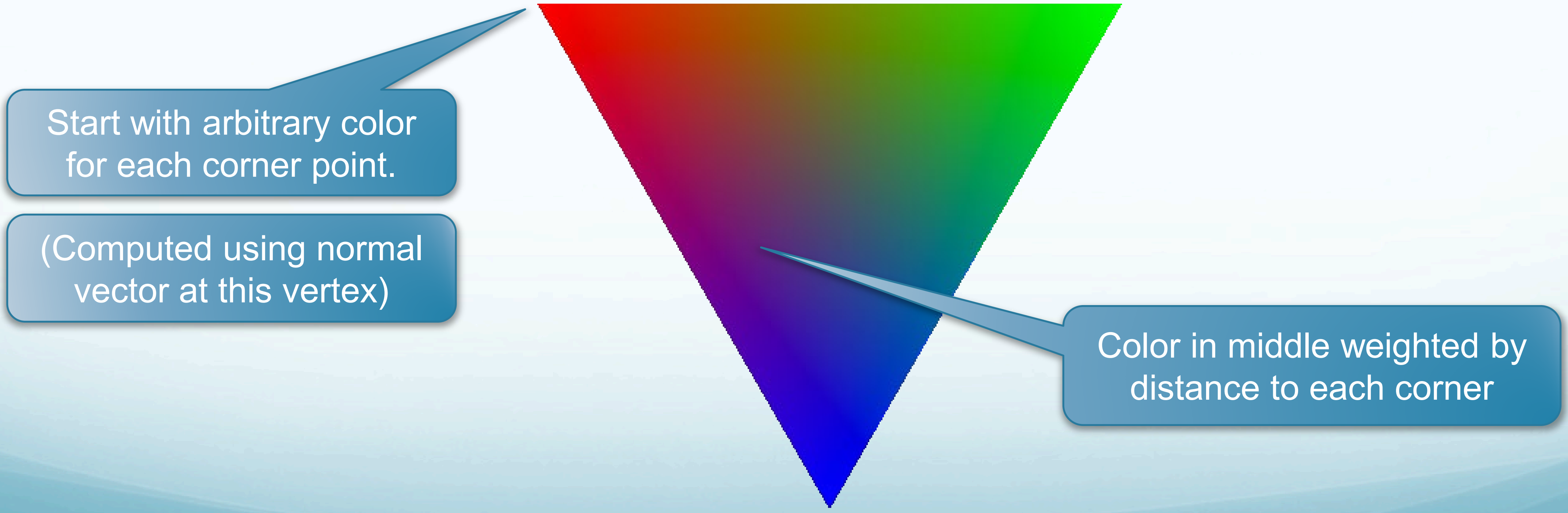
- Other two sides are defined by lines
- Algorithm:
  - Find intersection of two side lines: (x_i, y_i)
  - Loop for y from from y_i to the baseline at y_b
    - Compute points on two side lines: (x_l, y) and (x_r, y)
    - Fill pixels with centers between x_l and x_r
  - Repeat for other half-triangle



Gouraud Shading

Flat shading is simple, but usually we want something more complex

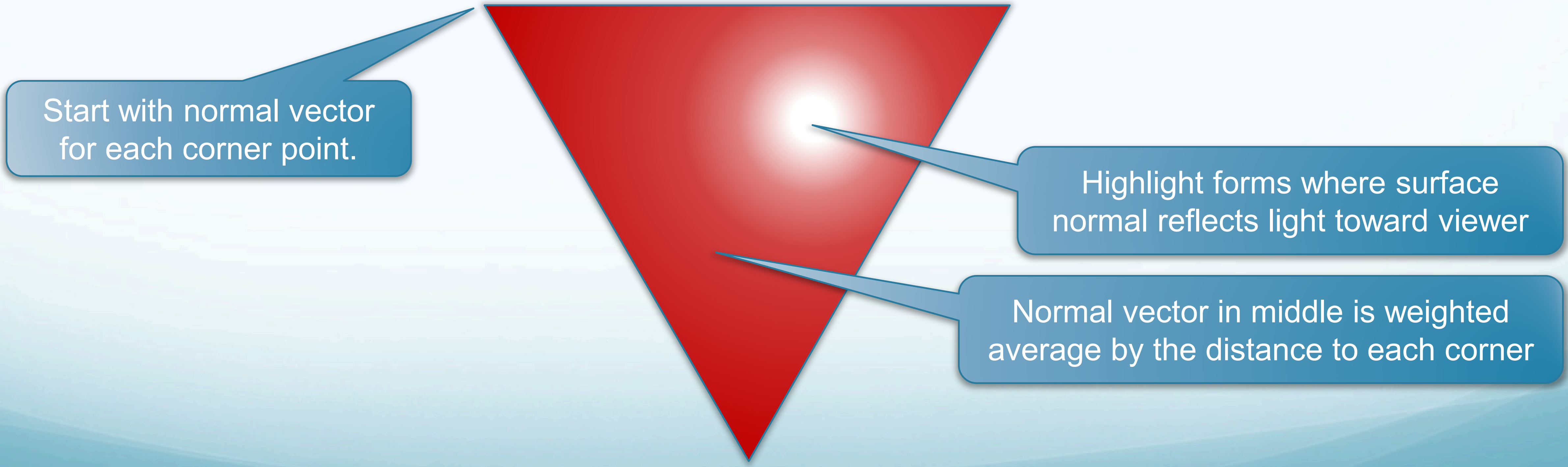
- Gouraud shading fills using colors that smoothly vary between 3 corners



Phong Shading

Phong shading combines diffuse transmission plus specular highlights

- Need to know the normal vector at each pixel to compute



Barycentric Coordinates

Both Goraud and Phong compute an average weighted by distance to corners

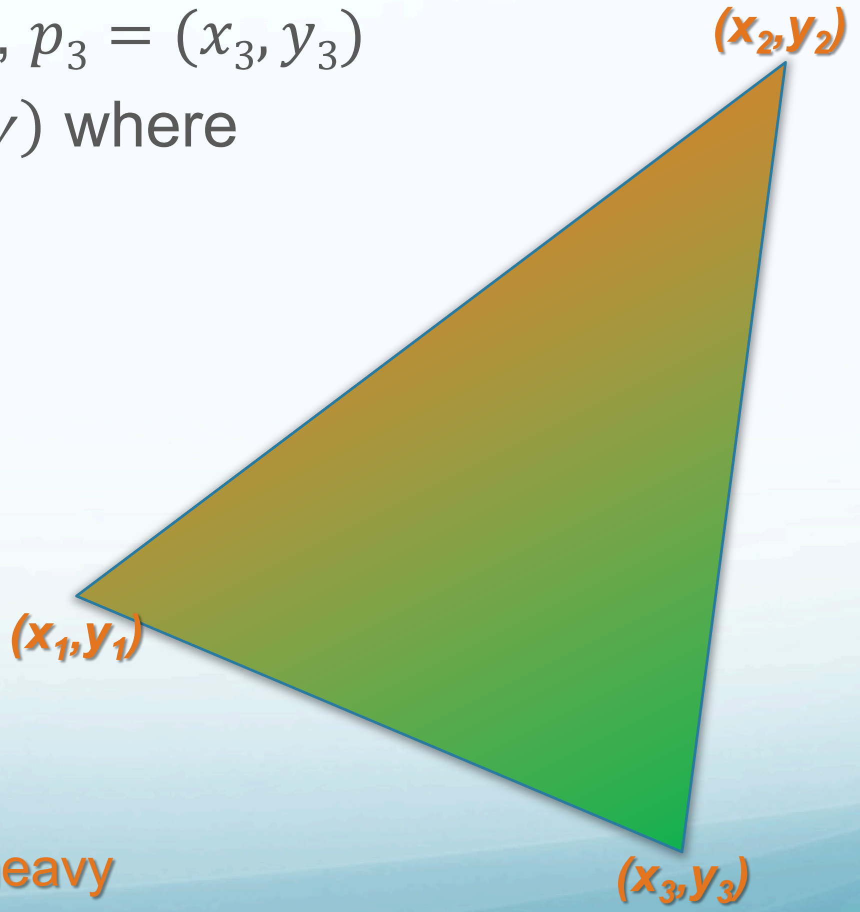
- Corners are $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$, $p_3 = (x_3, y_3)$
- Use **barycentric coordinates*** (α, β, γ) where

$$p = \alpha p_1 + \beta p_2 + \gamma p_3$$

Properties:

- $\alpha + \beta + \gamma = 1$ for edge points
- $\alpha > 0, \beta > 0, \gamma > 0$ for interior points
- Some $\{\alpha, \beta, \gamma\} < 0$ for exterior points

- Use for weighted averages:
$c = \alpha c_1 + \beta c_2 + \gamma c_3$



*Greek *barus* = heavy

Barycentric Coordinates

Given (x, y) we can compute (α, β, γ):

$$\alpha = f_{23}(x, y)/f_{23}(x_1, y_1)$$

$$\beta = f_{31}(x, y)/f_{31}(x_2, y_2)$$

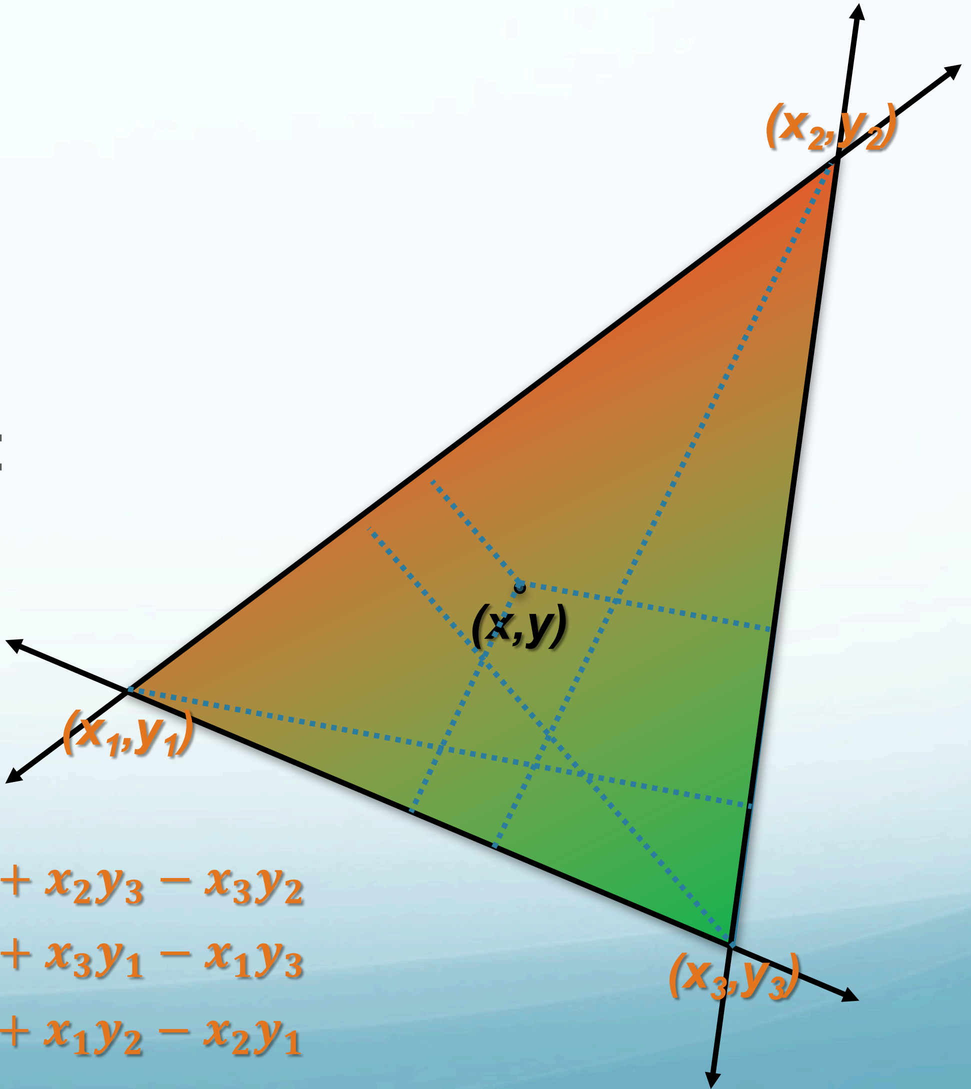
$$\gamma = f_{12}(x, y)/f_{12}(x_3, y_3)$$

Average color for Gouraud shading:

$$c = \alpha c_1 + \beta c_2 + \gamma c_3$$

Average normal for Phong:

$$\hat{n} = \alpha \hat{n}_1 + \beta \hat{n}_2 + \gamma \hat{n}_3$$



Distances from boundary lines:

$$f_{23} = (y_2 - y_3)x + (x_3 - x_2)y + x_2 y_3 - x_3 y_2$$

$$f_{31} = (y_3 - y_1)x + (x_1 - x_3)y + x_3 y_1 - x_1 y_3$$

$$f_{12} = (y_1 - y_2)x + (x_2 - x_1)y + x_1 y_2 - x_2 y_1$$

Questions

PAUSE NOW & ANSWER

1. What are the commonly customized parts of the graphics pipeline called?

   Shaders (specifically, the vertex shader and the fragment shader)

2. Why is the half-triangle algorithm more efficient than recursive fill?

   It doesn't waste any time on exploration. Each pixel is visited exactly once.

3. As the shader iterates over pixel (x, y) of a triangular face, how does it compute the weighted averages needed for Gouraud and Phong shading?

   The location is converted to barycentric coordinates, and the coefficients are used as weights.

Texture Mapping

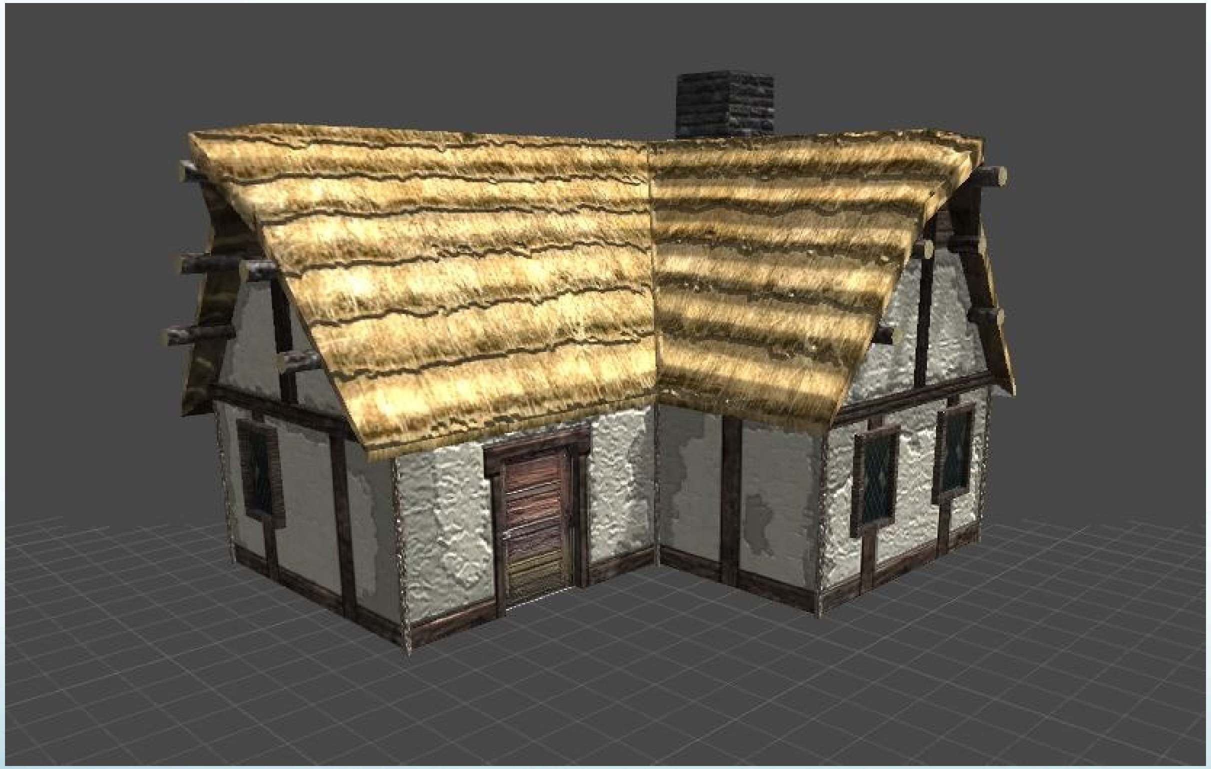
Texture Mapping

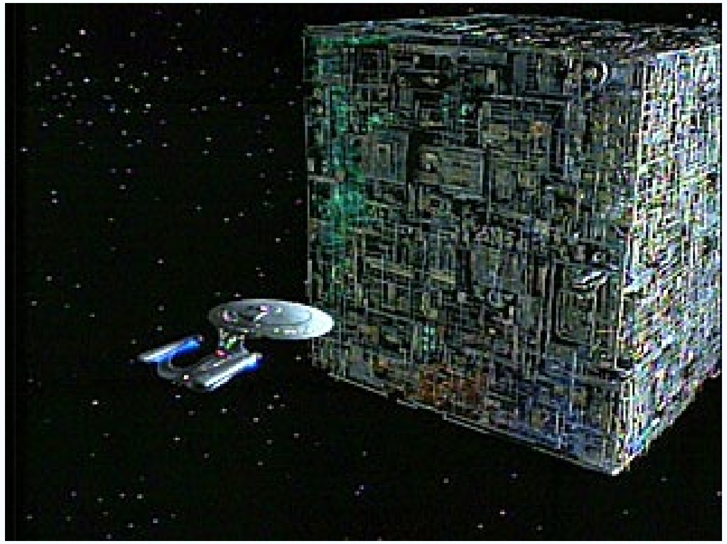
Uniform color is rare in real objects

https://wallpapersafari.com/hippie-van-wallpaper/
https://www.bbc.com/news/science-environment-28094441
https://www.hcn.org/issues/46.5/the-farm-bill-and-the-precipitous-decline-of-monarch-butterflies
https://www.1stdibs.com/furniture/building-garden/pedestals-columns/18th-century-breccia-marble-column-pedestal/id-f_763759/

Texture Mapping

How many polygons?



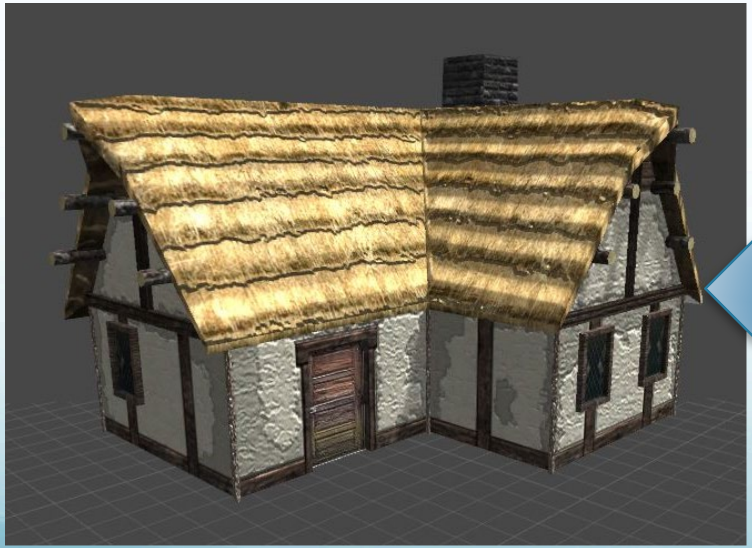


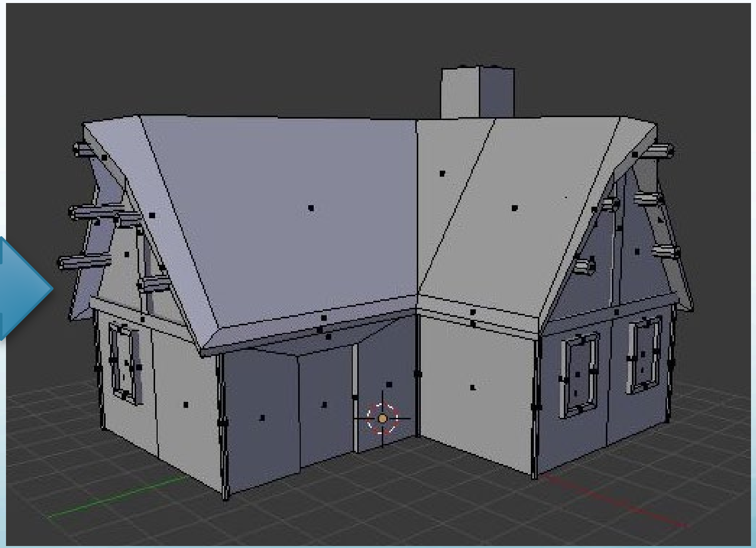
https://forum.unity.com/threads/how-to-make-uv-maps.224634/
http://www.startrek.com/database_article/borg-cube
https://www.pcgamesn.com/minecraft/15-best-minecraft-servers

Texture Mapping

Texture Mapping "paints" an image on rendered surfaces

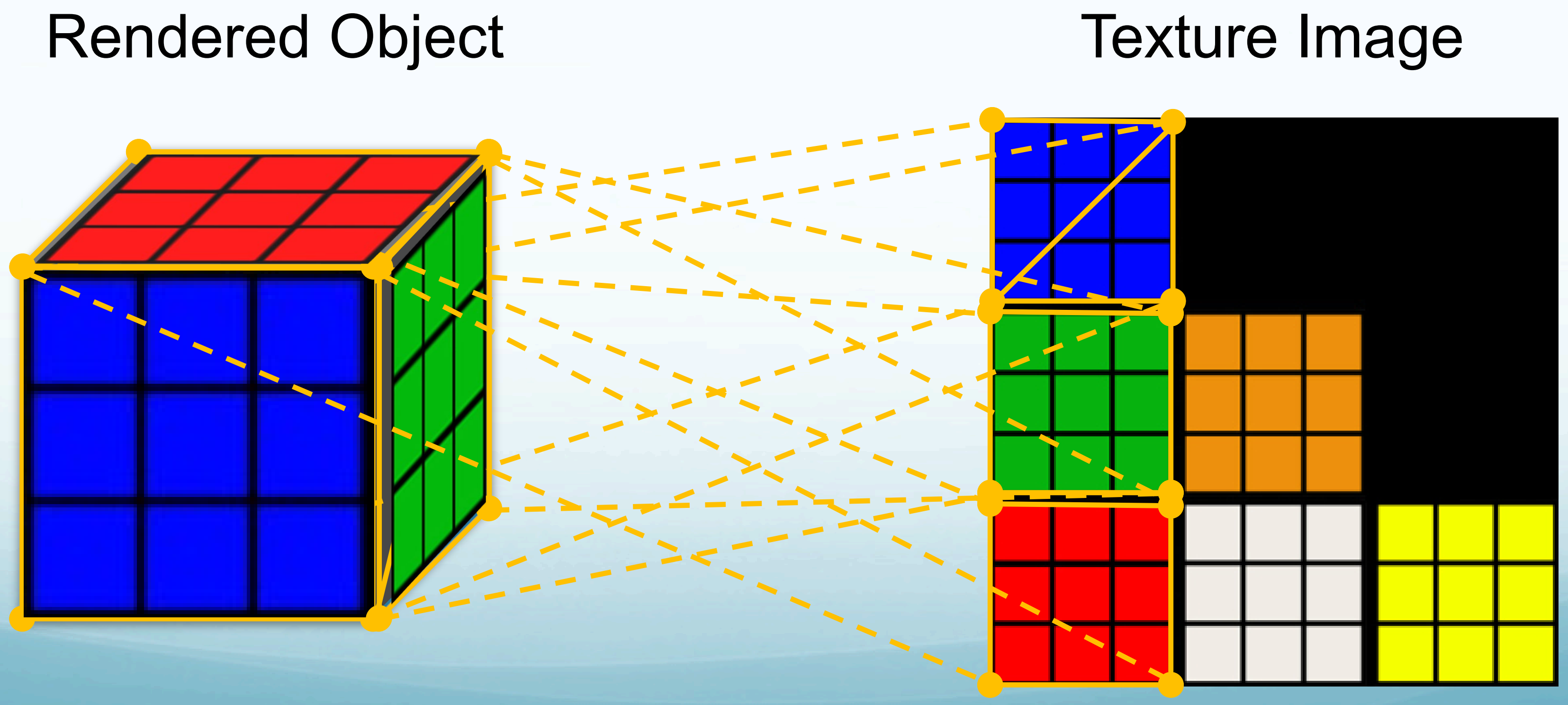
- Used in lieu of solid color
- For realism, data display, fun, etc.
- Realistic detail with many fewer polygons





Texture Mapping

- Specify a mapping between polygon vertices and a 2D texture image
- Shader copies colors from the texture image onto rendered surfaces
- Different parts of texture image for different faces



https://adrianpedersen.wordpress.com/2013/09/

Texture Mapping

- Texture images should be square, dimensions 2^k
- Texture coordinates designated as (u, v)
- Range from 0 to 1 in each dimension

**The textbook calls them (s,t)*

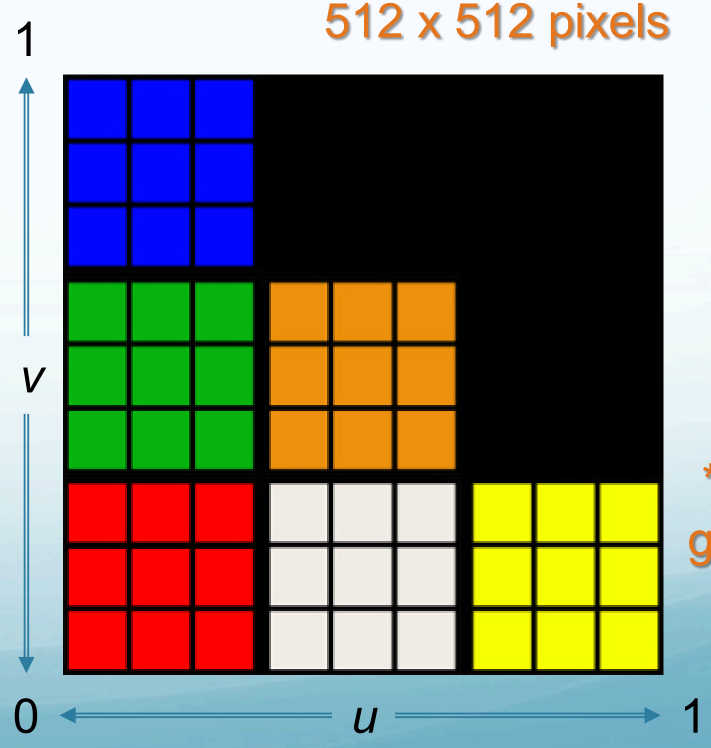
PAUSE NOW & ANSWER

Quiz:

Write down the coordinates of the four corners of the green cube face

(0,0.33), (0.33,0.33), (0.33,0.67), (0,0.67)

512 x 512 pixels



*Note that v axis goes up not down

Texture Mapping in Three.js

Three.js offers functionality to implement texture maps

- A class to load textures:

```
var loader = new THREE.TextureLoader();
var myTexture = loader.load("myTexture.jpg");
```

- Add the texture map to a Phong material:

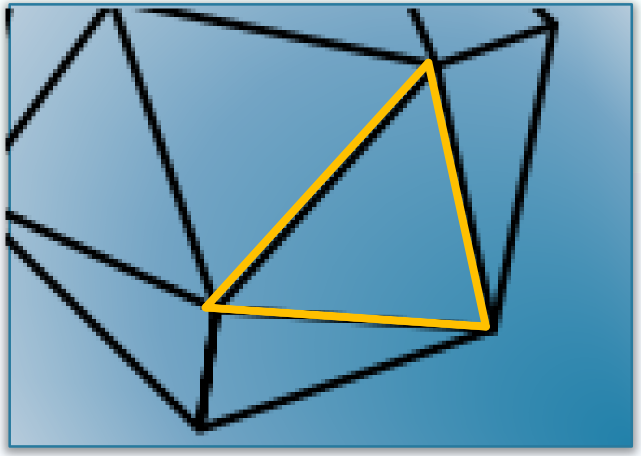
```
var myMaterial = new THREE.MeshPhongMaterial(
                        { map: myTexture } );
```

- Add (u, v) coordinates to each face of a geometry:

```
var uvcoords = [new THREE.Vector2(0, 0),
                new THREE.Vector2(1, 0),
                new THREE.Vector2(0, 1)];
myGeom.faceVertexUvs[0].push([
        uvcoords[0], uvcoords[1], uvcoords[2]]);
```

Texture Mapping

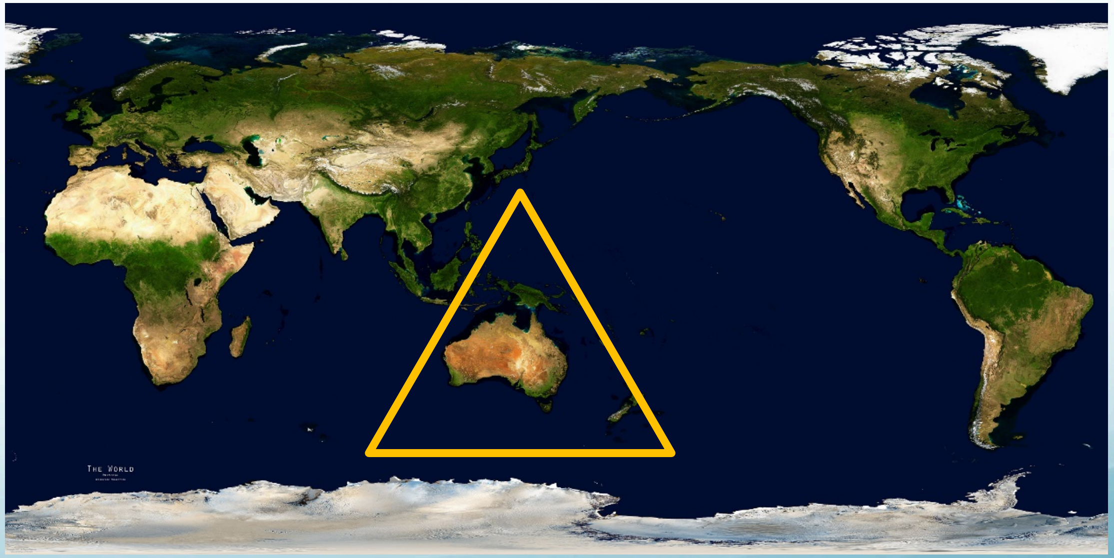
Rendering at large scale
Many samples
Closely spaced



Rendering at small scale
Fewer samples
Widely spaced

A given polygon in a scene may appear large or small

- Number of texels* per pixel varies widely



*Texel = texture element.
Like a pixel for texture!

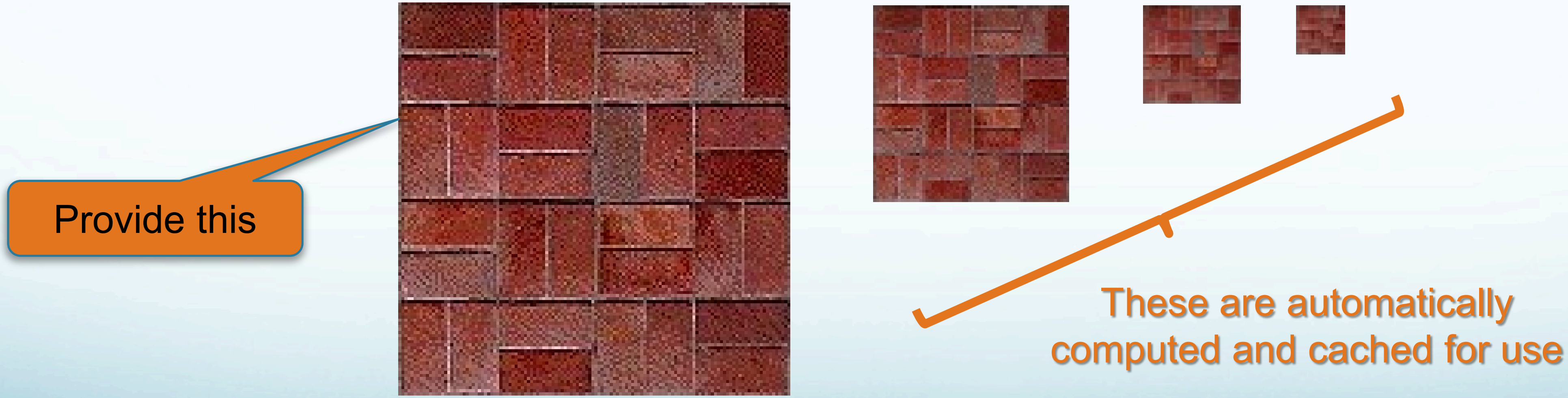
https://thenounproject.com/term/icosahedron/138194/

https://www.worldmapsonline.com/Satellite-Image-Maps.htm

Texture Mapping

Solution: use multiresolution **mipmap**

- Minification/magnification deals with further adjustment
- Automatically computed from high-resolution texture map



Texture Mapping

Can you guess what object these textures are for?

http://portfolio.sama-sama-studio.com/portfolio_page/steampunk-archer-hunter/

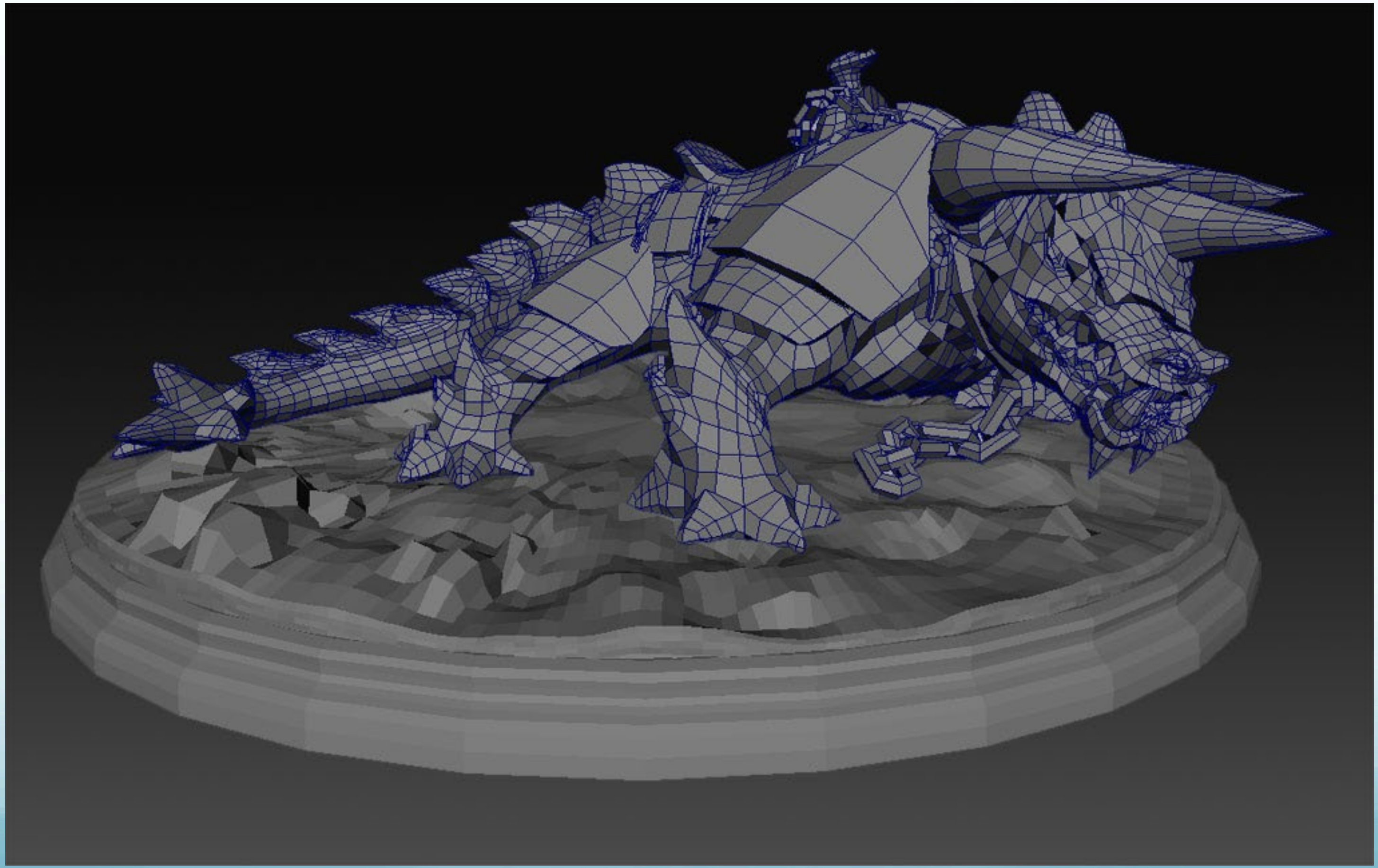
Texture Mapping

Can you guess what object these textures are for?

http://www.carmageddon.com/forum/topic/28868

Texture Mapping

Can you guess what object these textures are for?



http://www.aarongranofsky.com/Atma.html

Other Mappings

The mapping concept can be applied for many other tasks

- Bump map
- Normal map
- Gloss map
- Alpha map
- Reflections, shadows, environment, etc.

Can get these from a high-poly render!

… more on these later!

Questions

PAUSE NOW & ANSWER

1. What are the best dimensions for a texture image?

   A square image whose sides are a power of two

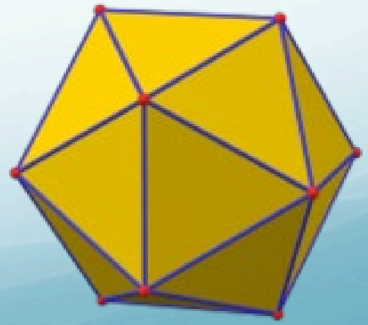
2. How does the program find the texture image?

   Use a texture loader. The image file should be in the same folder as the html file

3. This icosahedron has 20 faces and 12 vertices total. How many uv coordinates will you need to provide in order to fully map it with a texture?

   60 (three per face)

   Even though each vertex appears in five faces, its texture map coordinates will be different for each face.



Review

After watching this video you should be able to…

- Describe the stages of the graphics pipeline & what each does
- Implement a half-triangle fill algorithm for a polygon shader
- Compute barycentric coordinates and use them to find weighted averages
- Map textures to objects using a texture image and uv coordinates
- Explain the advantages of a texture mipmap
- Implement texture mapping in Three.js