15-462 Computer Graphics I
Lecture 13

Texture Mapping

Texture Mapping
Dealing with Texturing problems
Advanced Texturing techniques
[Angel 9.2]

March 14, 2002
Shayan Sarkar
Carnegie Mellon University

http://www.cs.cmu.edu/~fp/courses/graphics/

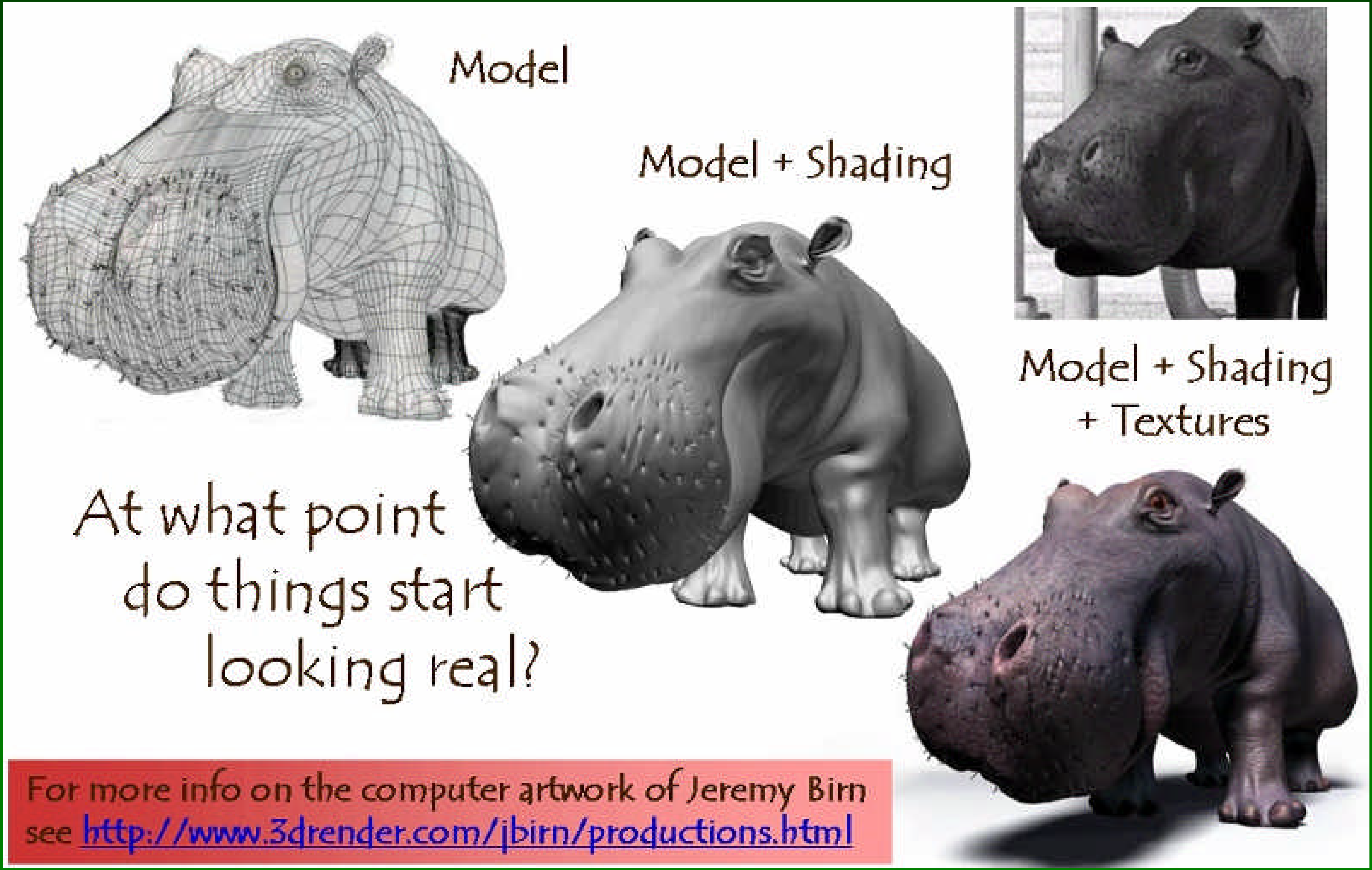
Administrative Stuff

- Assignment 5 is due next week
- Any questions on the lab so far?

Why Texture Map?

- So far we have done flat shading and Gouraud shading
  - Not good to represent everything in real world
- What are some of our other options?
  - Represent everything with tiny polygons
    - Geometry would get complicated very quickly
  - Apply textures across the polygons
    - This allows for less geometry but the image looks almost as good

Texture mapping sample



At what point do things start looking real?

For more info on the computer artwork of Jeremy Birn see http://www.3drender.com/jbirn/productions.html

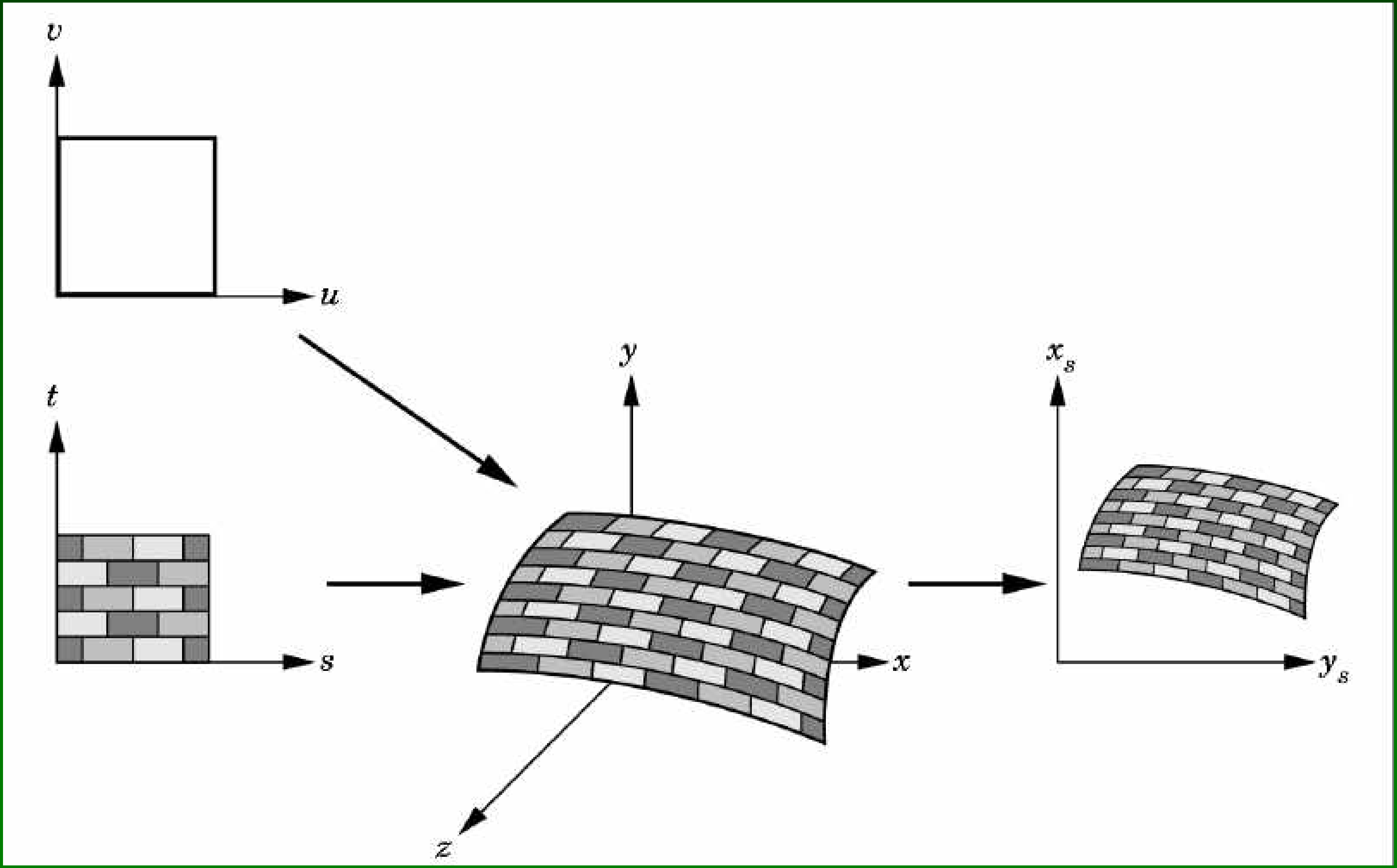
Basic Concept

- We need to find a way to relate a 2D image to a 3D model
- That’s where texture coordinates come in
  - A texture coordinate is a 2D coordinate (s,t) which maps to a location on a texture map
  - Generally the texture coordinates are over the interval [0,1]
- Assign a texture coordinate to each vertex
  - The texture coordinate is applied using some function which relates a spot on the texture with a vertex on the 3D model

Basic Concept

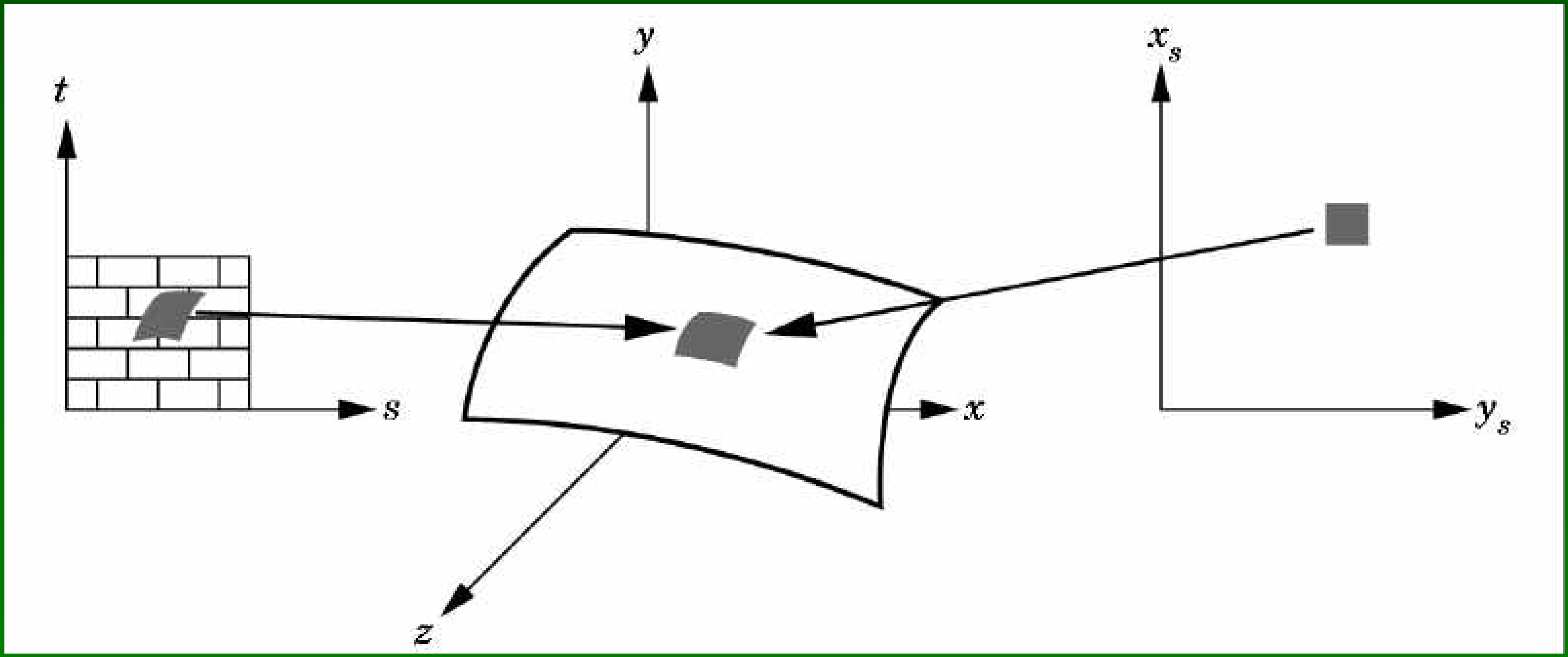
- Once you know the spot of the texture which applies to an area of your model, change the RGB value of that part of the model by the values in the texture
- This is called *parametric texture mapping*

Texture Coordinate sample



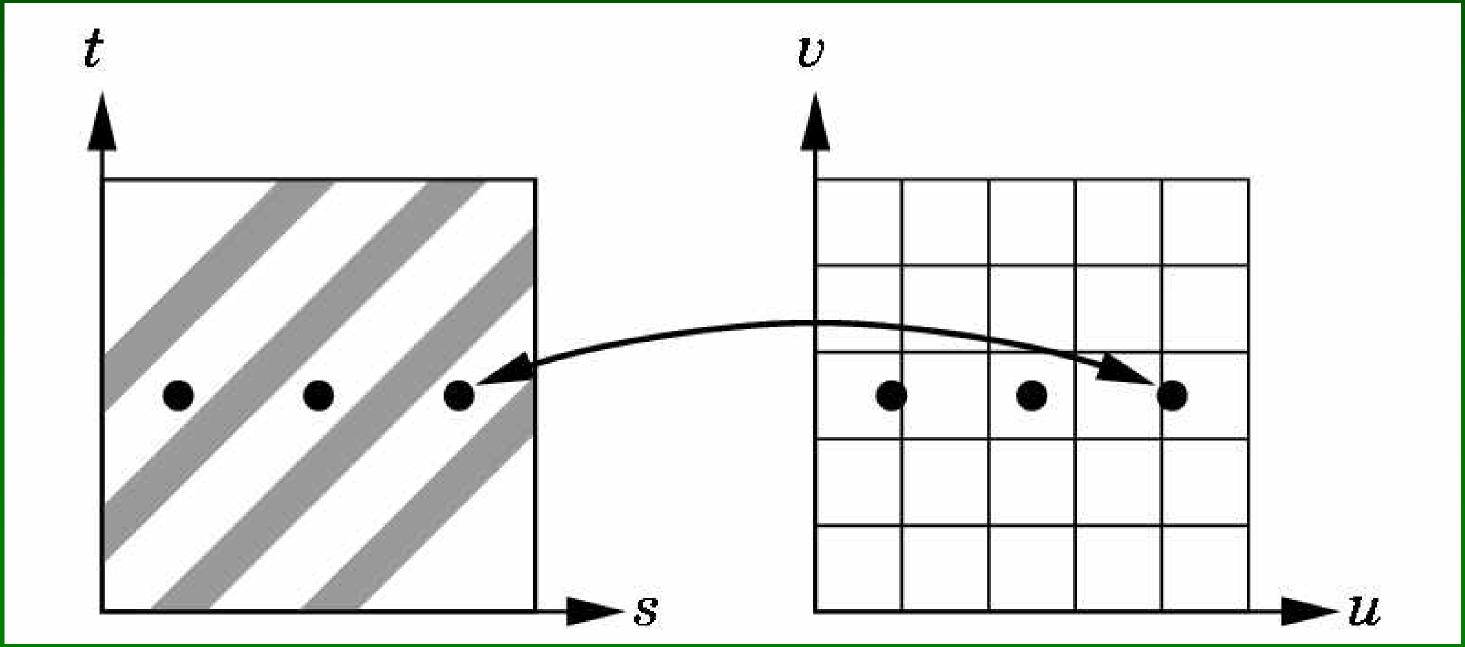
Difficulties with texture mapping

- Interested in mapping from screen to texture coordinates



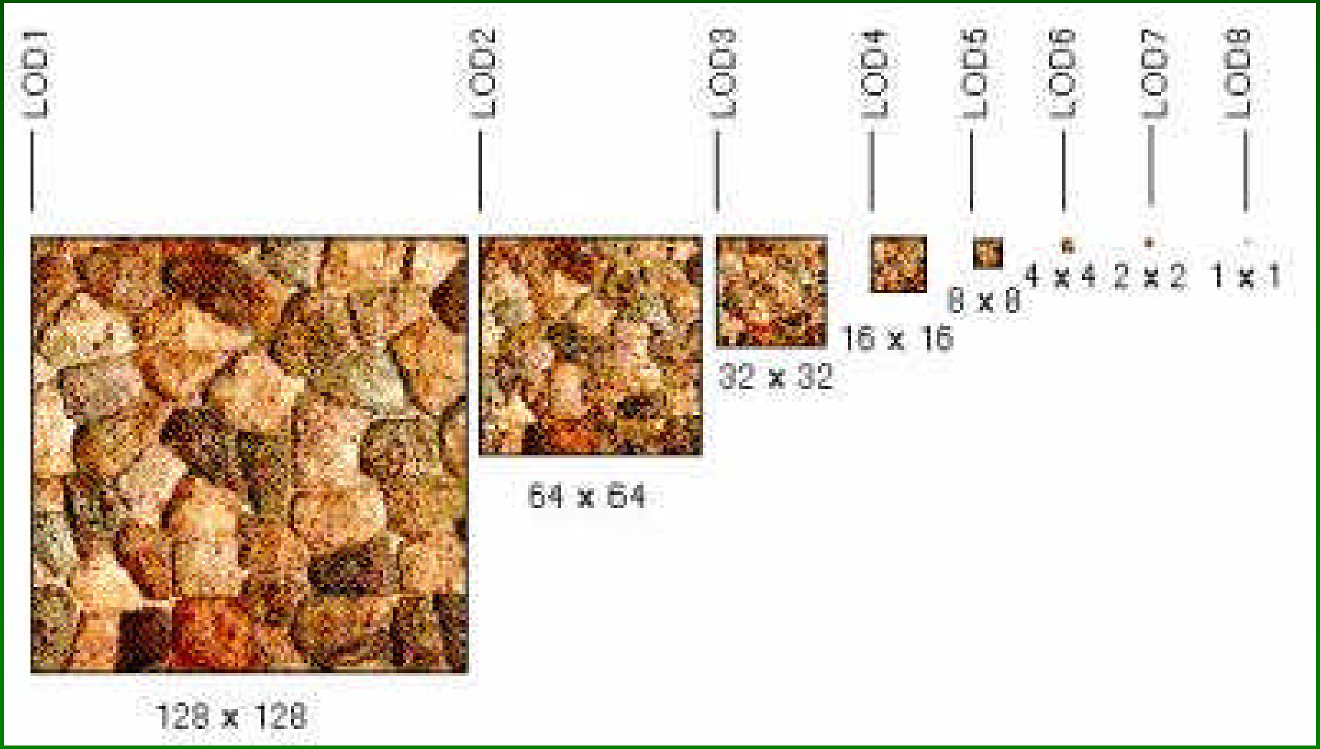
Difficulties with texture mapping

- Aliasing issues, it is especially apparent with regular textures



Some Solutions to Aliasing

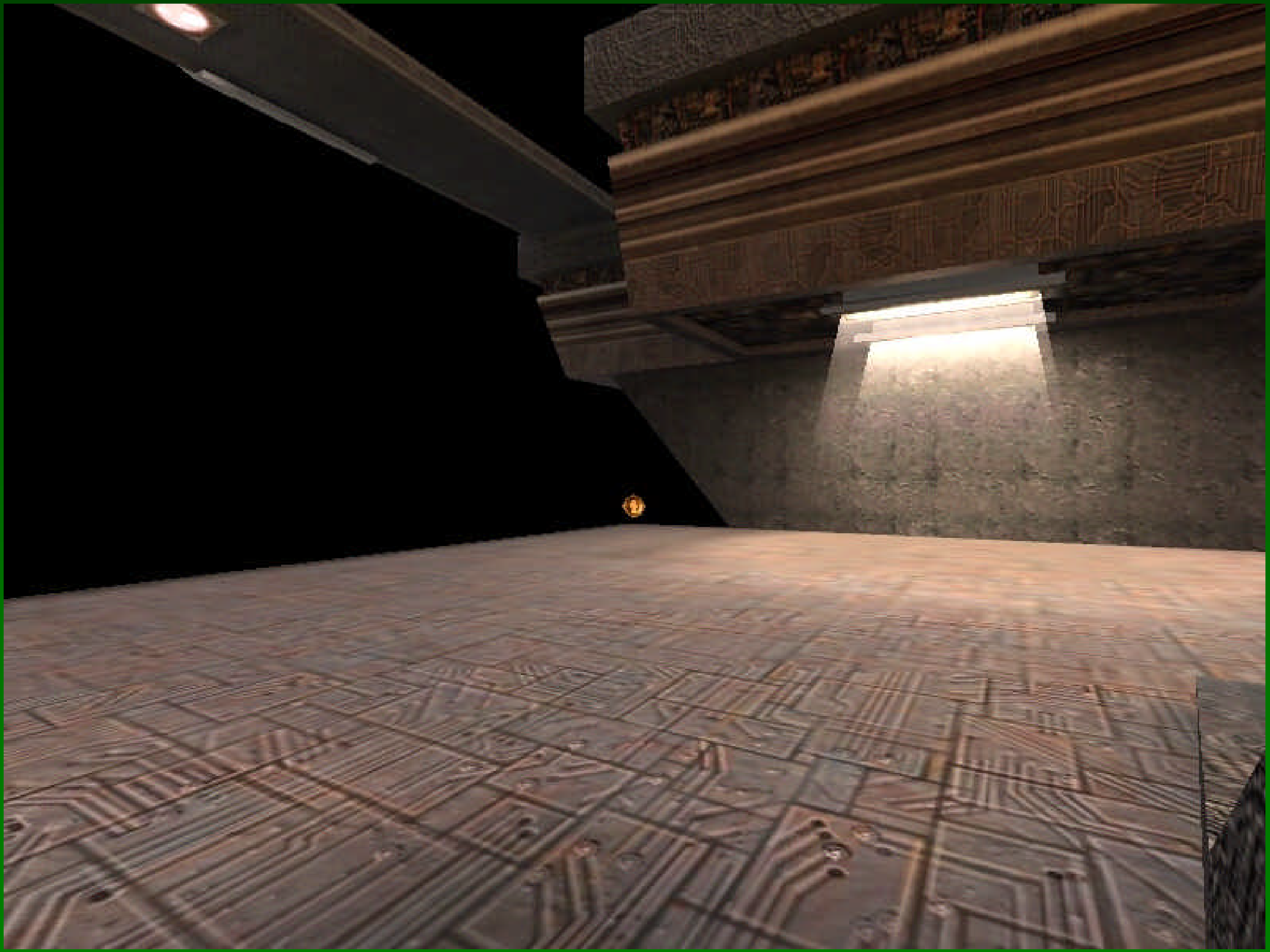
- Pre-calculate how the texture should look at various distances, then use the appropriate texture. This is called *mipmapping*



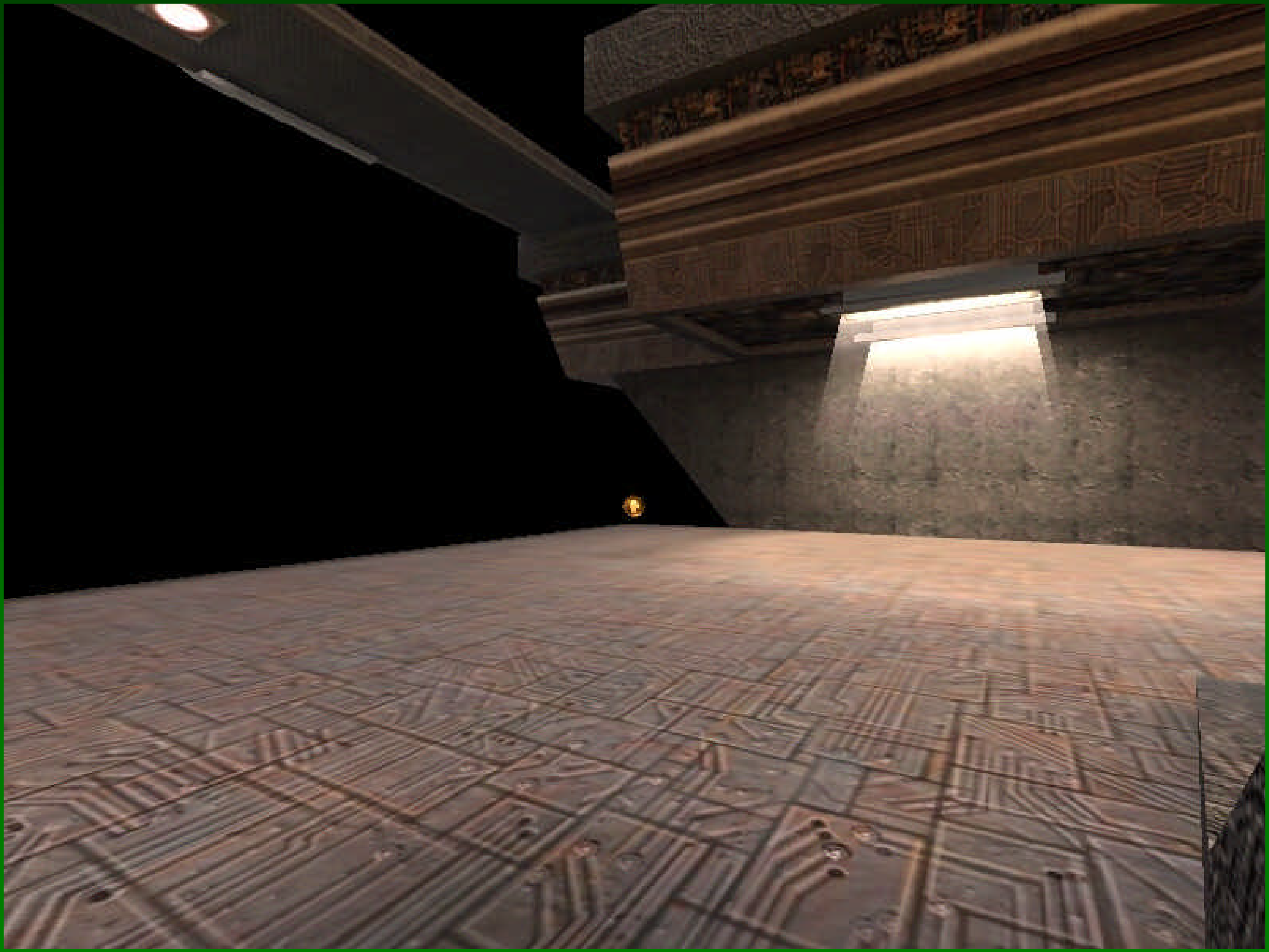
Some Solutions to Aliasing

- Take the average of surround values within the texture before applying the RGB value
- *Bilinear*
  - average of 4 surrounding pixels
- *Trilinear*
  - two bilinear filters
- *Anisotropic*
  - elliptical mask over the texture
  - expensive since elliptical mask can change so nothing can be pre-calculated

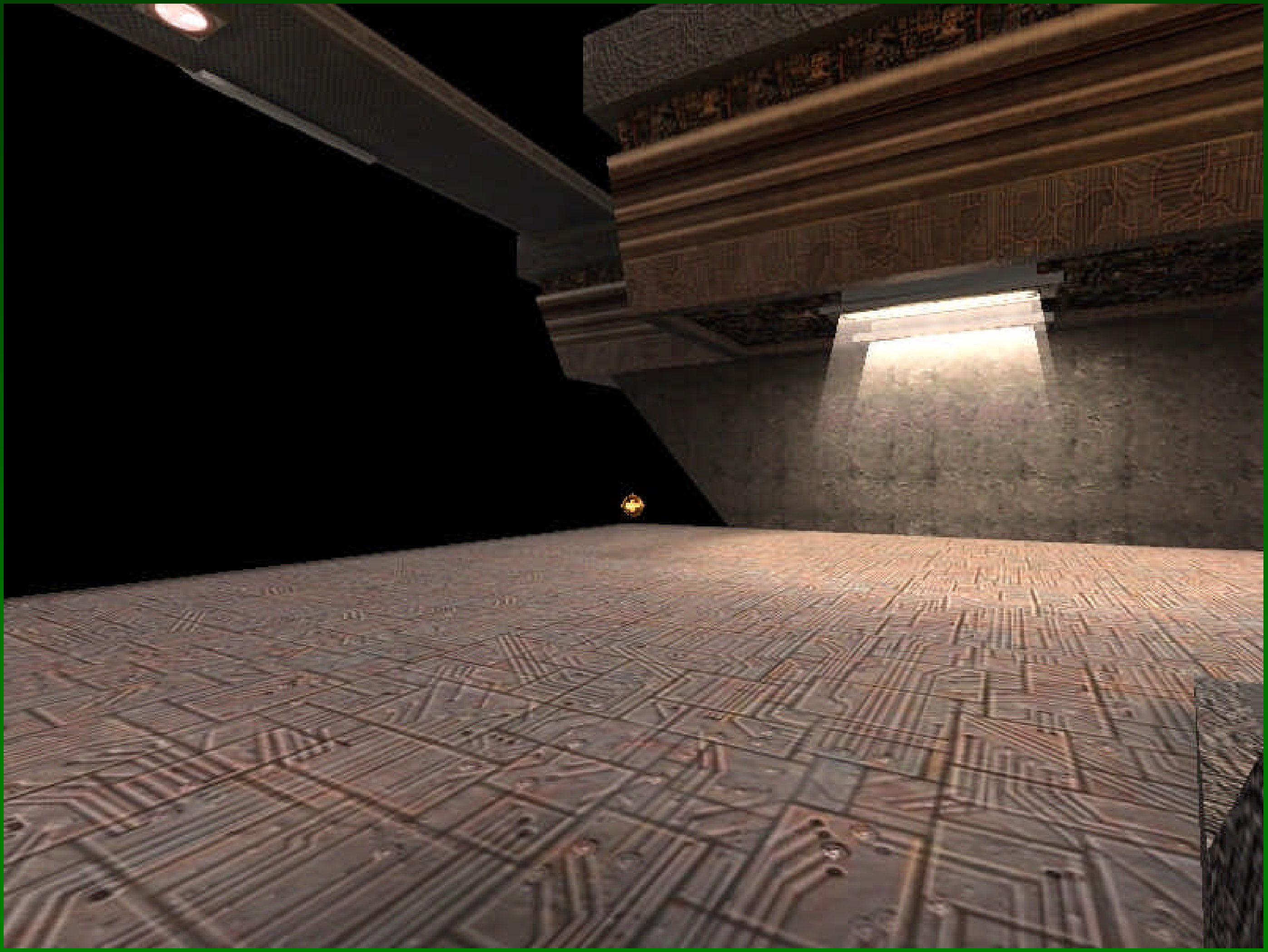
Bilinear Filtering



Trilinear Filtering

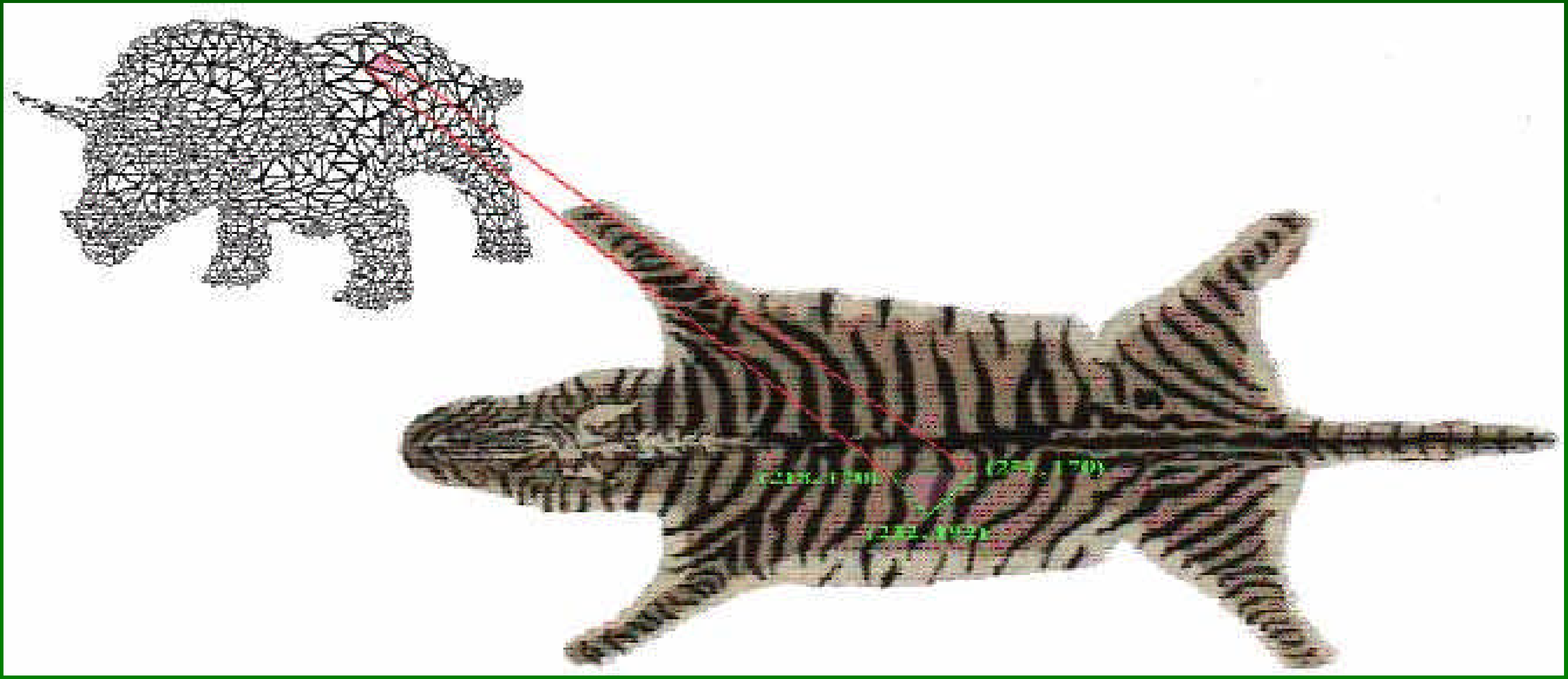


Anisotropic Filtering



Difficulties with texture mapping

- Another difficulty is how to map a 2D image onto an arbitrary 3D model
  - Ex: If you wanted to map onto a sphere, you can't do it without distorting the texture

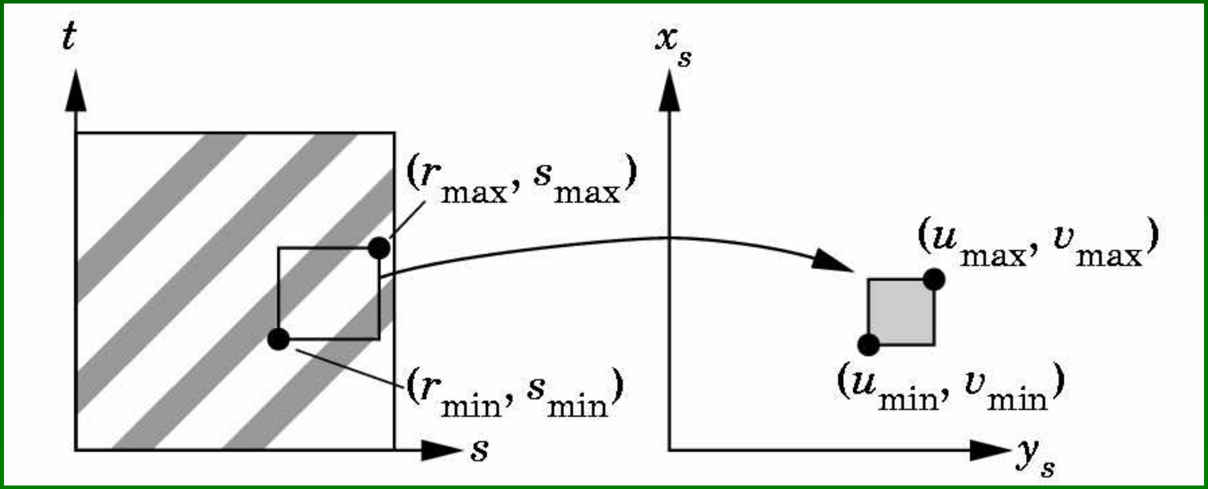


Linear Texture mapping

- Do a direct mapping of a block of texture to a surface patch

$$u = u_{\min} + \frac{s - s_{\min}}{s_{\max} - s_{\min}}(u_{\max} - u_{\min})$$

$$v = v_{\min} + \frac{t - t_{\min}}{t_{\max} - t_{\min}}(v_{\max} - v_{\min})$$

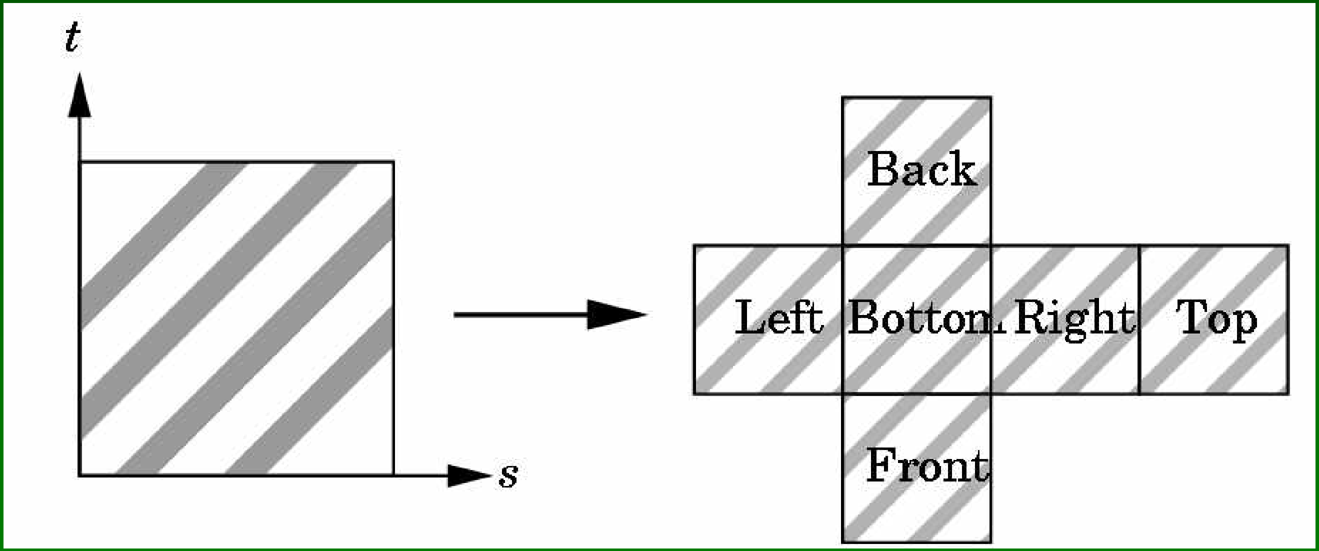


Two-part mapping

- Map texture to intermediate 3D object
- Map 3D object to the final model
- Common intermediate objects
  - Cylinder
  - Cube

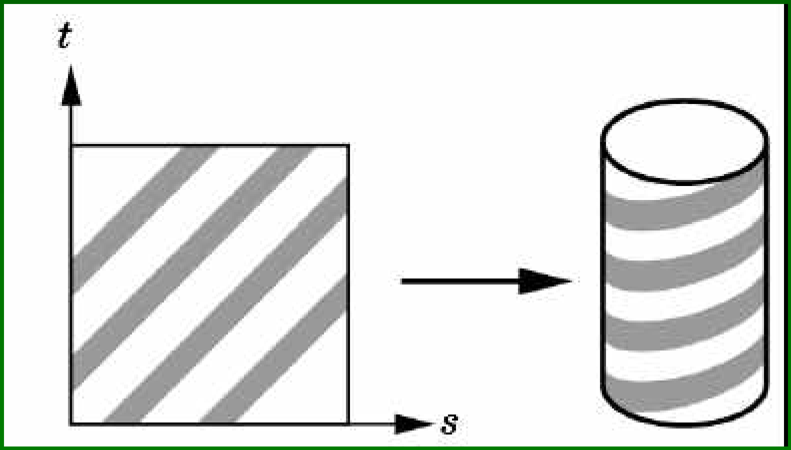
Cube Mapping

- "Unwrap" cube and map texture over the cube



Cylinder Mapping

- Wrap texture along outside of cylinder, not top and bottom
  - This stops texture from being distorted



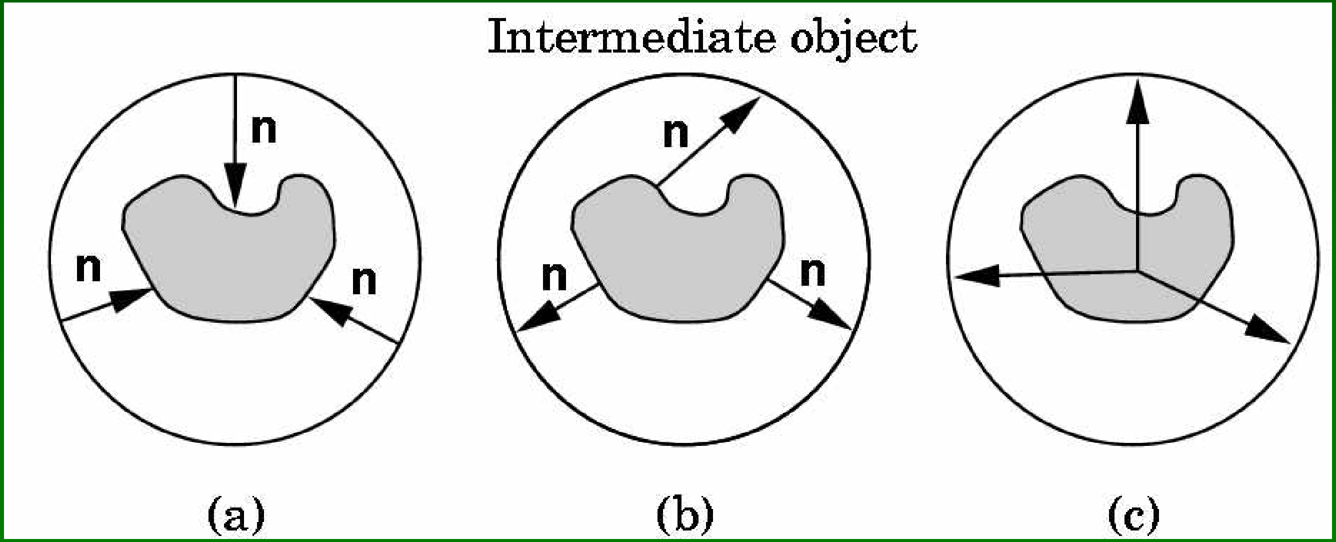
$$x = r \cdot \cos(2pu)$$

$$y = r \cdot \sin(2pu)$$

$$z = v / h$$

3D Object to Model

- This step can be done in many ways
  - Normal from intermediate surface
  - Normal from object surface
  - Use center of object



glTexImage2D

- glTexImage2D(GL_TEXTURE_2D, level, components, width, height, border, format, type, tarray)
- GL_TEXTURE_2D
  - Specify that it is a 2D texture
- Level
  - Used for specifying levels of detail for mipmapping (more on this later)
- Components
  - Generally is 0 which means GL_RGB
  - Represents components and resolution of components
- Width, Height
  - The size of the texture must be powers of 2
- Border
- Format, Type
  - Specify what the data is (GL_RGB, GL_RGBA, …)
  - Specify data type (GL_UNSIGNED_BYTE, GL_BYTE, …)

glTexCoord2f

```
glEnable(GL_TEXTURE_2D);
glTexImage2D(GL_TEXTURE_2D, 0, 3, texture->nx, texture->ny, 0, GL_RGB,
                    GL_UNSIGNED_BYTE, texture->pix);

glBegin(GL_POLYGON);

  glTexCoord2f(1.0, 1.0);
  glVertex3f(1.0, 0.0, 1.0);

  glTexCoord2f(1.0, -1.0);
  glVertex3f(1.0, 0.0, -1.0);

  glTexCoord2f(-1.0, -1.0);
  glVertex3f(-1.0, 0.0, -1.0);

  glTexCoord2f(-1.0, 1.0);
  glVertex3f(-1.0, 0.0, 1.0);

glEnd();
```

Other Texture Parameters

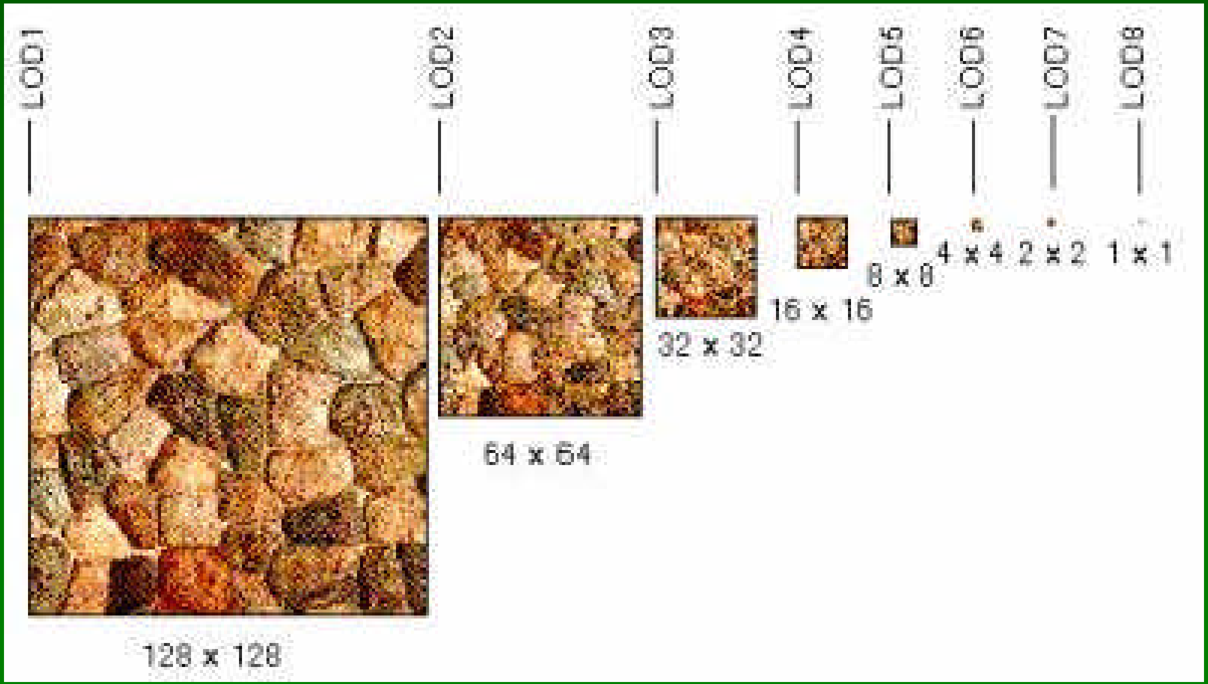
- glTexParameterf()
  - Use this function to set how textures repeat
    - glTexParameterf(GL_TEXTURE_WRAP_S, GL_REPEAT)
    - glTexParameterf(GL_TEXTURE_WRAP_S, GL_CLAMP)
  - Which spot on texture to pick
    - glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST)
    - glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST)

Mipmapping in OpenGL

- gluBuild2DMipmaps(GL_TEXTURE_2D, components, width, height, format, type, data)
- This will generate all the mipmaps using gluScaleImage
- glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST_MIPMAP_NEAREST)
  - This will tell GL to use the mipmaps for the texture

Mipmapping in OpenGL

- If you design the mipmaps yourself
  - glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, 128, 128, 0, GL_RGB, GL_UNSIGNED_BYTE, LOD1)
  - glTexImage2D(GL_TEXTURE_2D, 1, GL_RGB, 64, 64, 0, GL_RGB, GL_UNSIGNED_BYTE, LOD2)



Other Texturing Issues

- glTexEnvi(GL_TEX_ENV, GL_TEX_ENV_MODE, GL_MODULATE)
  - Will balance between shade color and texture color
- glTexEnvi(GL_TEX_ENV, GL_TEX_ENV_MODE, GL_DECAL)
  - Will replace shade color with texture color

- glHint(GL_PERSPECTIVE_CORRECTION, GL_NICEST)
  - OpenGL does linear interpolation of textures
    - Works fine for orthographic projections
  - Allows for OpenGL to correct textures for perspective projection
    - There is a performance hit

- Texture objects
  - Maintain texture in memory so that it will not have to be loaded constantly

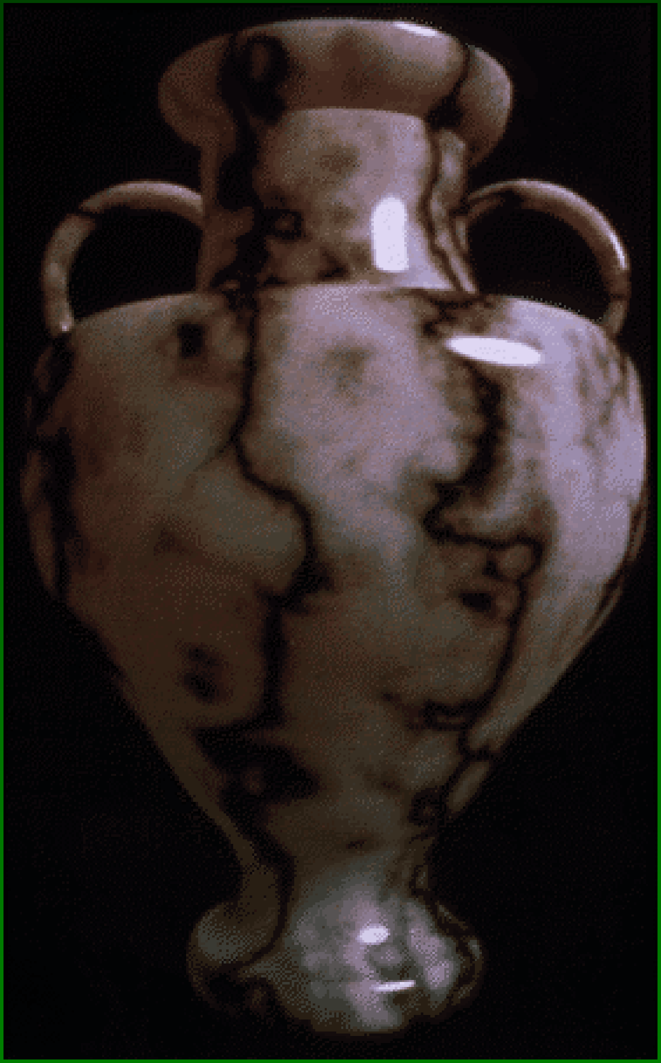
OpenGL texturing code

Look at sample code

This code assumes that it's an RGB texture map

3D Texture Mapping

- Almost the same as 2D texture mapping
  - Texture is a “block” which objects fit into
  - Texture coordinates are 3D coordinates which equal some value inside the texture block

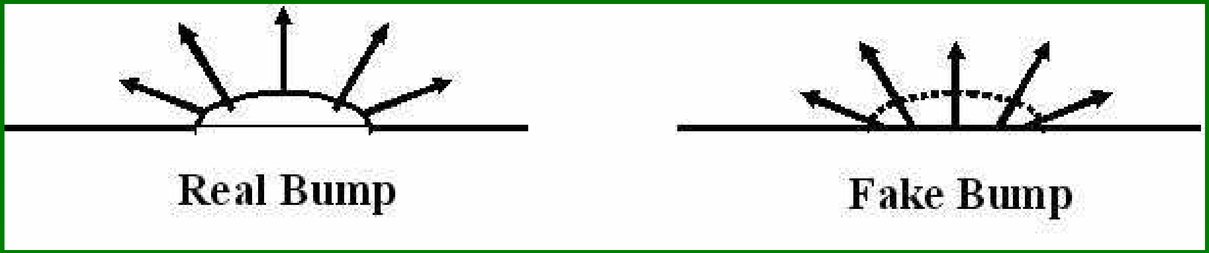


Tricks with Texture mapping

- Bump Mapping
- Environment Mapping
- Light Maps

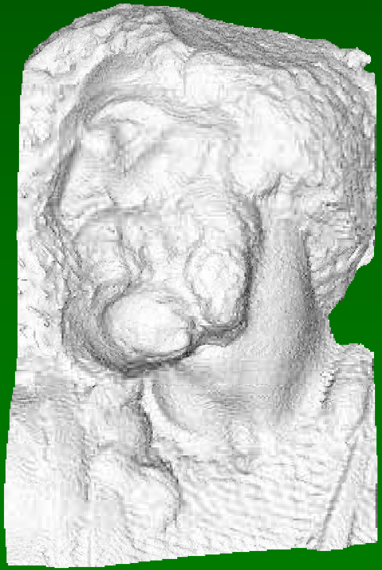
Bump Mapping

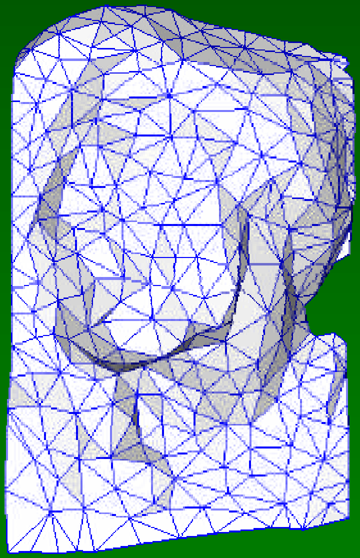
- How do you make a surface look *rough*?
  - Option 1: model the surface with many small polygons
  - Option 2: perturb the normal vectors before the shading calculation
    - the surface doesn't actually change, but shading makes it look that way
    - bump map fakes small displacements above or below the true surface
    - can use texture-mapping for this
    - For the math behind it all look at Angel 9.4

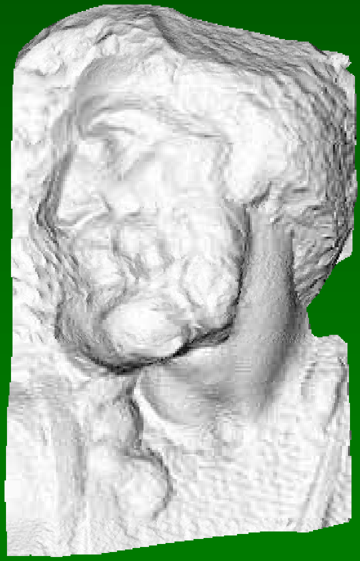


Bump Mapping

- In CG, we can perturb the normal vector without having to make any actual change to the shape.
- What would be the problems with bump mapping?


Original model (5M)


Simplified (500)

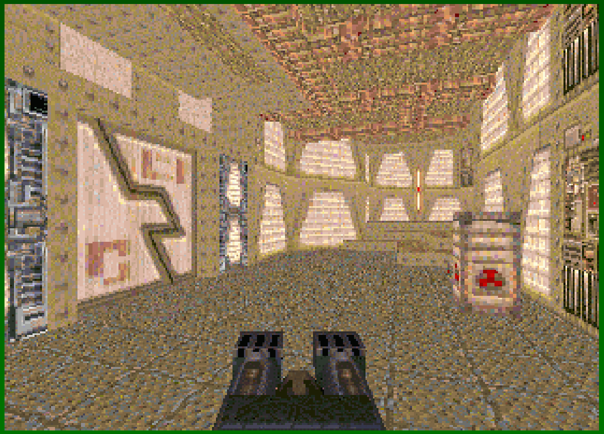

Simple model with bump map

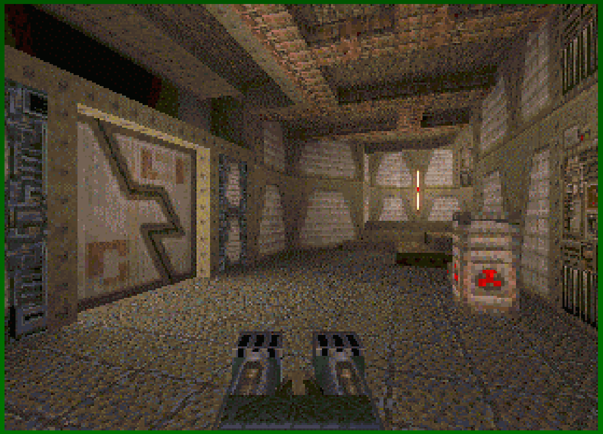
Environment Mapping

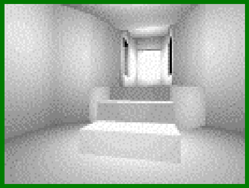
- Allows for world to be reflected on an object without modeling the physics
- Map the world surrounding an object onto a cube
- Project that cube onto the object
- During the shading calculation:
  - Bounce a ray from the viewer off the object (at point P)
  - Intersect the ray with the environment map (the cube), at point E
  - Get the environment map’s color at E and illuminate P as if there were a virtual light source at position E
  - You see an image of the environment reflected on shiny surfaces

Light Mapping

- *Quake* uses *light maps* in addition to (radiance) texture maps. Texture maps are used to add detail to surfaces, and light maps are used to store pre-computed illumination. The two are multiplied together at run-time, and cached for efficiency.


Radiance Texture Map Only


Radiance Texture + Light Map


Light Map

Summary

- Texture Mapping
  - Why is it good?
  - Ways of projecting the texture
  - Problems with texture mapping
- Hacks you can do with it
  - Bump mapping
  - Environment Mapping
  - Light Mapping

Where we're headed

- This is the last lecture using OpenGL
- Going to be going into more low-level rendering
  – Per-pixel stuff
  – Ray-tracing