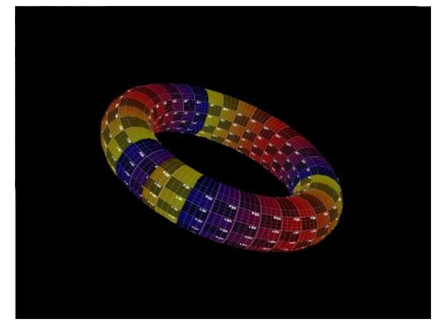
CS 248 Review Session #4

Lighting

Texture Mapping

Assignment 2



- Due Monday, 2/13/12
- Reference images up
- Materials
  - Ambient, diffuse, specular, shininess
- Texturing
  - Coordinates, filtering
- Lighting
  - Coordinates/normals, lighting equations

Pixels vs Fragments

Pixels have:

- color

Fragments have:

- color
- depth
- normals
- eye coordinates
- texture coordinates
- …

Lighting and Shading

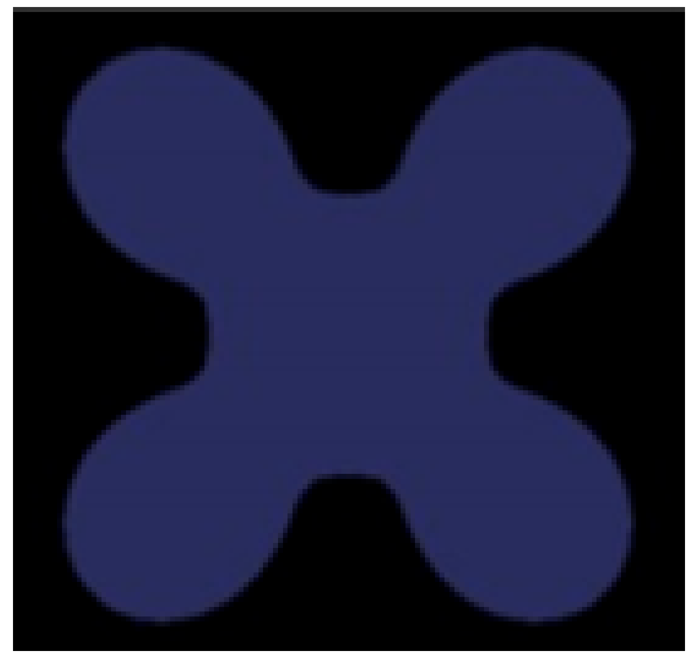
- Ambient, diffuse and specular
- Affected by both materials **and** lights
- Break up the intensity, I, into two components
  - Amount of light = L
  - Reflectance = R
- $I_* = L_* R_*$ for $* \in \{\text{ambient, diffuse, specular}\}$
- $I_{final} = I_a + I_d + I_s$
  - Remember to clamp your I_*!

Light Properties

- For each component, there is actually 1 equation *per color channel (R, G, B)*
- Three components (L_a, L_d, L_s) * 3 channels
  - Actually 9: ($L_{a,red}, L_{a,blue}, L_{a,green}$, …)
- mglLight(which light?, which parameter?, what value?)
  - `Float rgb[] {1.0, 0.1, 0.2};`
  - `mglLight(MGL_LIGHT, MGL_LIGHT_SPECULAR, rgb);`

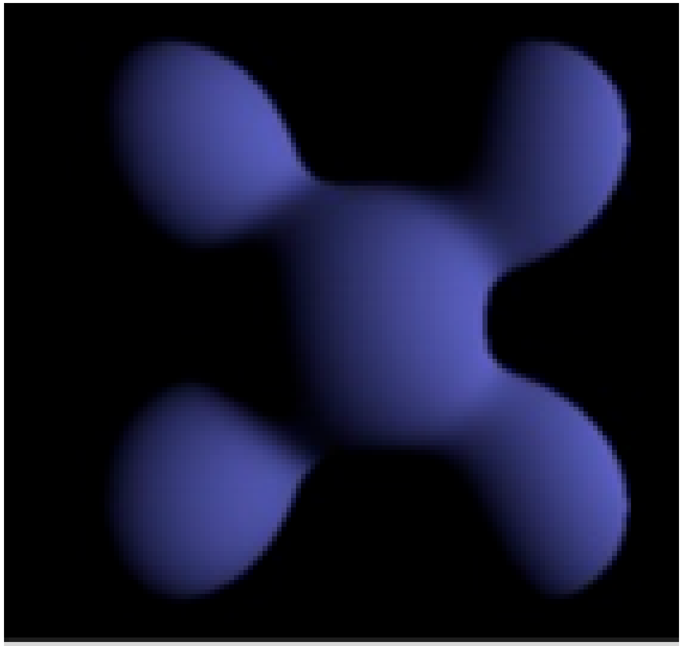
Ambient Equation

- $I_a = L_a * k_a$
  - $I_{a,red} = L_{a,red} * k_{a,red}$ and so on
- Flat, global, does not depend on direction



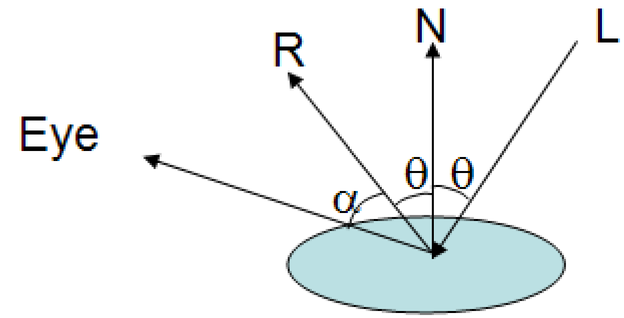
Diffuse Equation

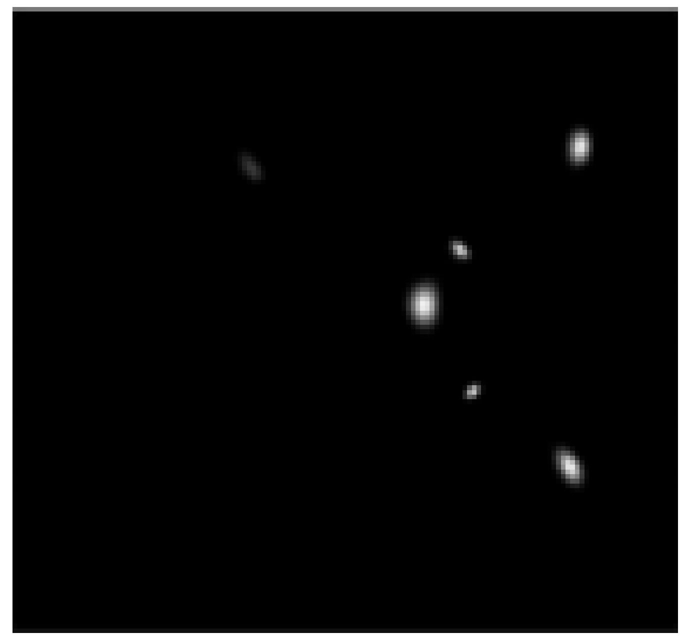
- $I_d = R_d L_d$
- $R_d = k_d(N \cdot L)$
  $= k_d \cos(\theta_d)$
  - Why dot product?
  - N = fragment normal vector
  - L = fragment eye coordinate to light



Specular Equation

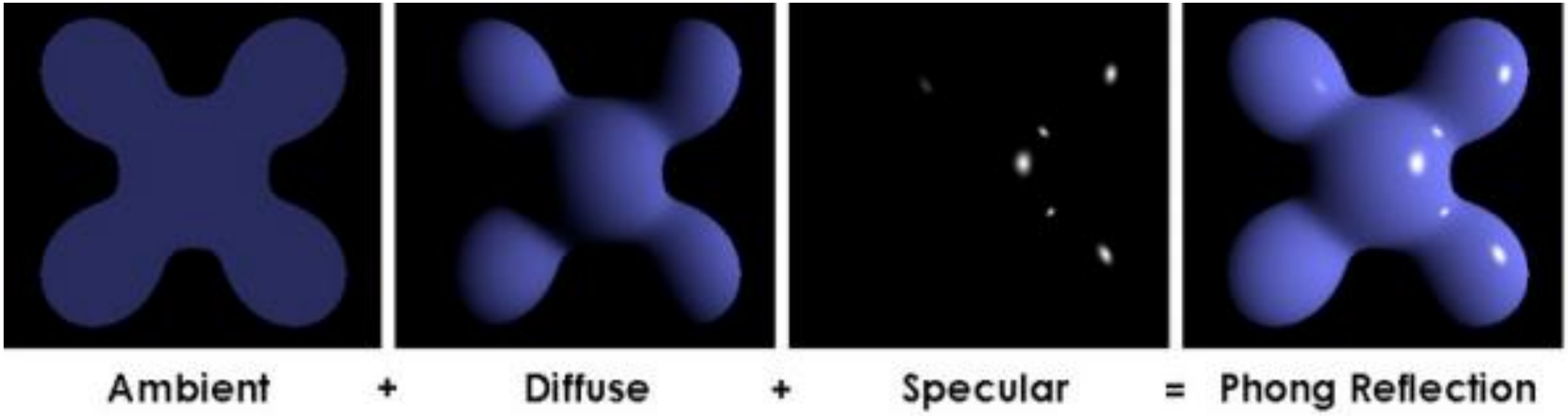
- $I_s = L_s R_s$
- $R_s = k_s (R \cdot V)^{shininess}$
  - R = mirror reflection of light vector off surface
  - V = fragment eye coordinate to the camera
  - Where is the camera?
- Phong model
  - Feel free to implement Blinn-Phong



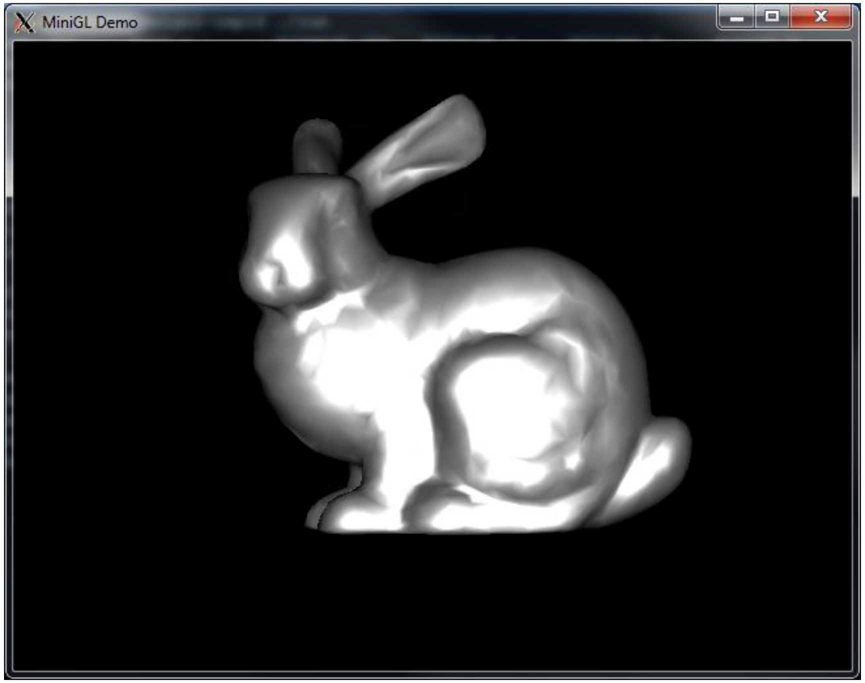


Final Color

- Add them all up!
- $I = I_a + I_d + I_s$



Toon Shading



Toon Shading

- Typically, $N \cdot L, R \cdot V \in [0,1]$
- Toon shading discretizes this range to get a "banded" effect
- Trivial to implement
  - Convert value of dot products to a step function
  - For example, $N \cdot L, R \cdot V \in \{0, 0.1 , 0.7, 1\}$

Textures

- Textures are more than just images painted onto objects!
- Think of them as arbitrary ***functions*** that can be applied to surfaces
- Modern graphics techniques makes heavy use of textures for advanced rendering effects
  - Shadows maps
  - Light maps
  - Normal maps
  - Opacity maps

Textures

- Each vertex has $(\mathrm{u}, \mathrm{v}) \in [0,1]^2$, that index into a location in the texture.
- In OpenGL, texture mapping requires the specification of:
  - Image
  - Mapping between object and texture coordinates
  - Filtering technique
  - Application

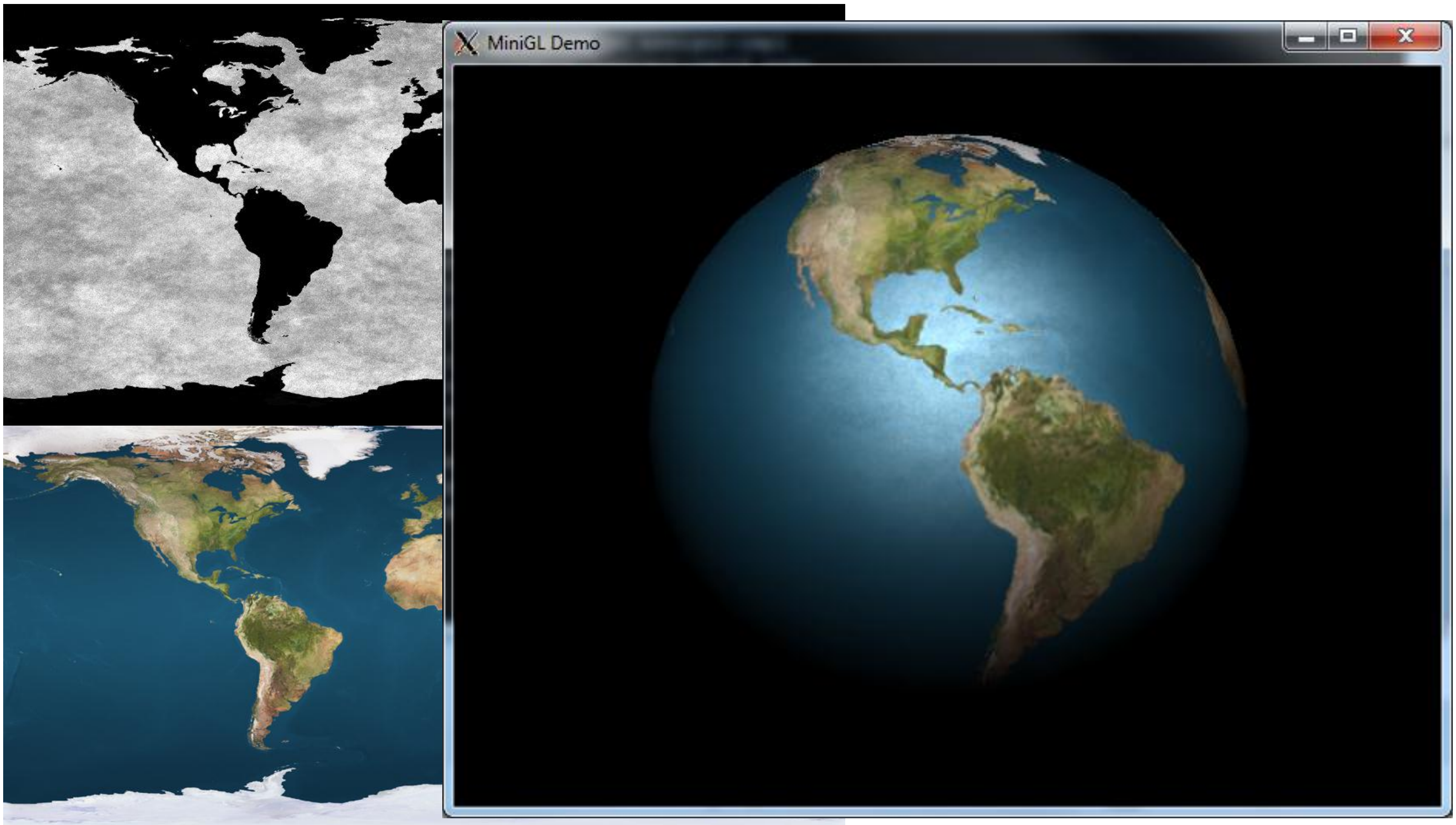
Textures

- In Assignment 2, we are just modulating the material color by the texture color
- $R_d = t_d(u, v) k_d \cos(\theta_d)$
- $R_s = t_s(u, v) k_s (R \cdot V)$
- If either texture is disabled, $t_*(u, v) = 1$

Texture Slots

- Notice that the texture applied to the diffuse component can *different* from the one applied to the specular component
- In MiniGL, textures for each slot (diffuse, specular) are enabled and specified separately

Earth Render

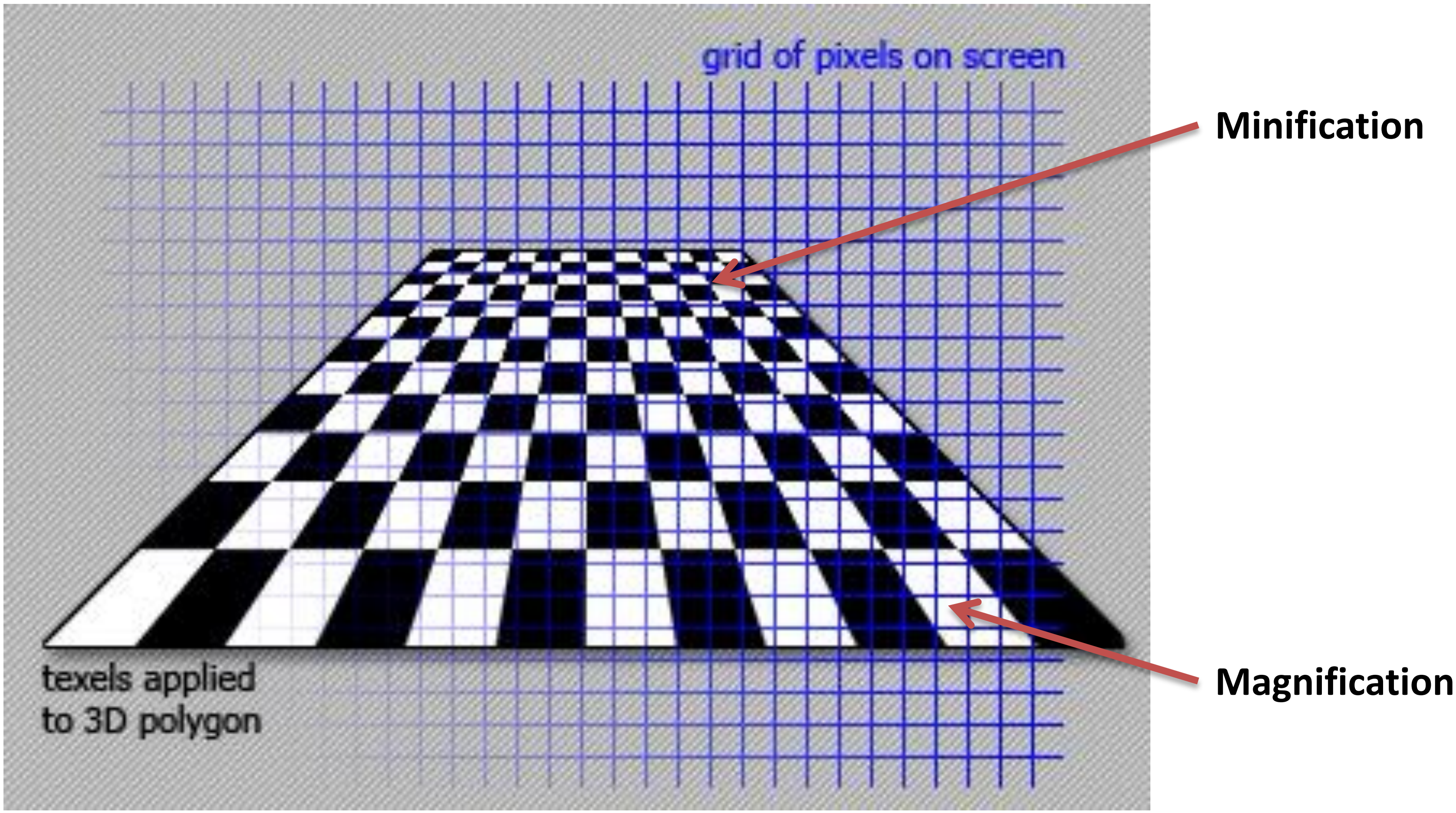


Texture Slots Usage

```
// Add a specular texture
// Turn off diffuse textures
mglTextureSlot(MGL_TEX_SPECULAR);
mglTexturesEnabled(true);
mglUseTexture(2);

mglTextureSlot(MGL_TEX_DIFFUSE)
mglTexturesEnabled(false);
```

Texture Filtering

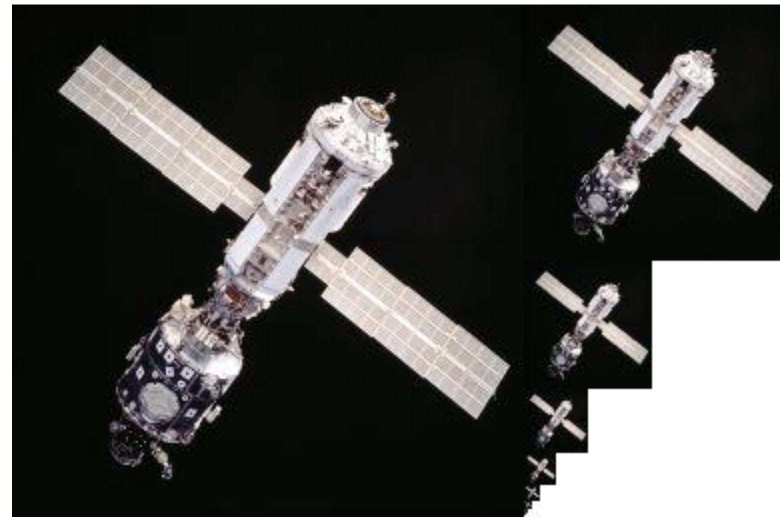


Texture Filtering Methods

- Nearest-neighbor
  - Sample texel closest to pixel center interpolated (u,v) coordinate
- Bilinear
  - Perform weighted average on 2x2 patch of texels
- For magnification, both nearest and bilinear filters give reasonable results
- For minification, filtering is necessary

Mipmapping

- Not required for assignment 2
  - But feel free to experiment with!
- Pros
  - Reduced artifacts since scaled images are already “anti-aliased”
- Cons
  - Requires 33% more space



Assignment 2

- Good luck!