CS541 Project 1.5 (optional)

Synopsis

Add some interesting texture maps to the Phong lighting calculation.

Instructions

Enhance project 1 by reading in two provided texture maps describing various features of the surface of the earth, and using them to control the Phong lighting calculation for the sphere.

Produce a "nice" image of the earth using data from the two provided images. You may use the two provided images:

- **earth.raw**: A 1024x512 RGB image of earth's surface
- **effects.raw:** A 1024x512 RGB image containing 3 single-channel images (r=specular factor, g=night image, b=cloud image)

A "nice" image of the earth should include at least the following:

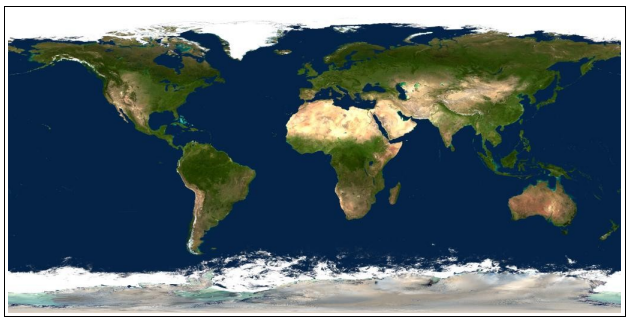
- Use Phong shading but restrict the specular reflection to watery regions using the effects.r channel.. (This channel contains a 1.0 for watery regions, and a 0.0 for land regions.)
- Choose day-time colors (from earth.rgb) when $N \cdot L$ is positive, and night-time lights (from the effects.g channel) when $N \cdot L$ is negative.
- Since the terminator (the line between night and day) is not a sharp line, due to the atmospheric scattering, implement a region of dawn/dusk between the two.
- Implement transparent clouds by using the cloud channel (effects.b) as a blending factor between ground color and cloud color (pure white).
- Be creative in using these these raw images to produce a nice earth image. For instance, you might try to add sunset/sunrise colors to the clouds along the terminator.

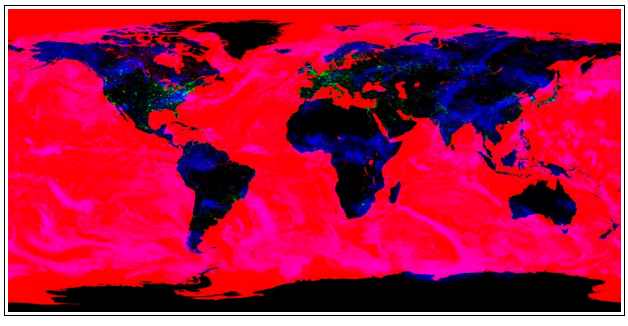
The images

The images are provided in a "raw" format that is trivially easy to read. To read a raw image of width **w,** height **h,** and depth (or channels) **d,** do:

```
unsigned char bytes[w*h*d];
FILE* f = fopen("....raw", "rb");
fread(bytes, w*h, d, f);
fclose(f);
```

Both images are 1024 by 512 in size. The first contains the (diffuse) color of the surface, while the second contains specular, night lights, and clouds in the three channels:





Implementation details

See https://faculty.digipen.edu/~gherron/OpenGL/TOC.html for a complete set of Reference Pages
describing all the following calls in detail.

Texture coordinates

The framework produces a sphere with texture coordinates exactly matched to the two images (i.e., polar coordinates). Your vertex shader must propagate the texture coordinate to the fragment shader via a built in varying variable like this:

```
gl_TexCoord[0] = gl_MultiTexCoord0;
```

and your pixel shader can access the resulting interpolated texture coordinates with

```
gl_TexCoord[0]
```

Sending texture

To put the texture on the graphics card and associate it with a texture id,do this:

```
int texId;
glGenTextures(1, &texId);
glBindTexture(GL_TEXTURE_2D, texId);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGB, 1024, 512, GL_RGB,
                                              GL_BYTE, bytes);
glBindTexture(GL_TEXTURE_2D, 0);
```

Many parameters can be set here. Some affect how the texture is stored in graphics memory, but most affect how the texture is accessed (e.g., bilinear/trilinear blending, mipmap usage, wrap/repeat/clamp modes …).

Activate texture

To use a texture in a shader, the texture must be active in a texture unit, and when no longer drawing with a texture, it must be inactive.

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, texId); //Activate in unit 0
<<draw geometry here>>>
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, 0); //Deactivate from unit 0
```

Repeat both sections with other textures using **GL_TEXTURE1** and so on for the other texture units.

Shader access

Shaders can access active shaders through samplers:

```
uniform sampler2D earth;
vec4 Kd = texture2D(earth, gl_TexCoord[0].st);
```

Both the vertex and pixel shaders can access textures through such a sampler. Some sampler details vary between the two types of shaders.

Notify shader

The final connection between the shader and the application is to associate a texture ID (like texId above) with the sampler used to access the texture (**samper2D earth** above):

```
int loc = glGetUniformLocation(program, "earth");
glUniform1i(loc, 0);
```

This must be done after the shader is bound, but before the shader is executed – meaning before the geometry is drawn. Repeat for other textures in other texture units.