Discussion 5
CSE 167

Announcements

- Project 2 Due Nov 8th @ 11:59 PM
  - Project 2 Late Deadline exactly 1 week after

Defining Lights and Materials

Outline

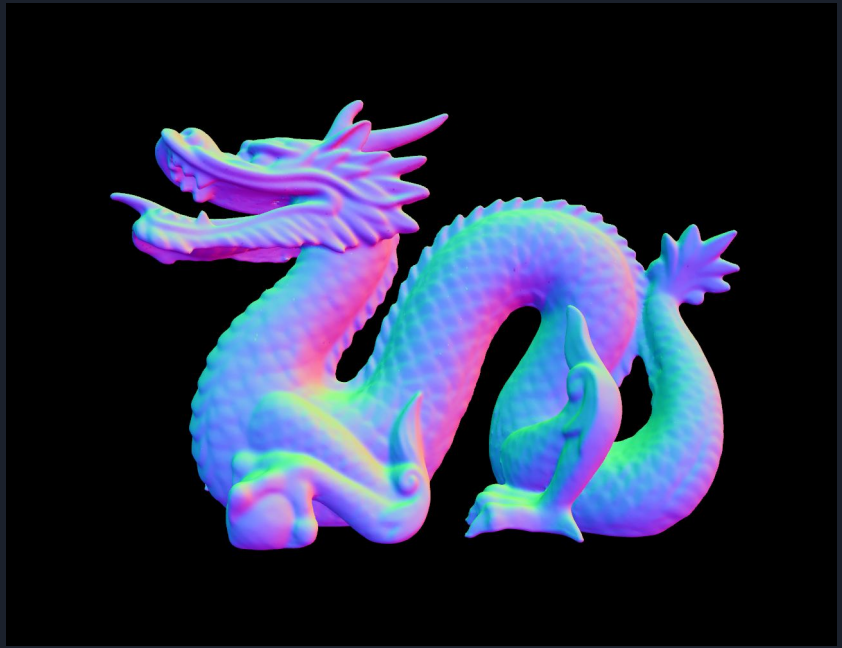
All on 2nd half of Project 2

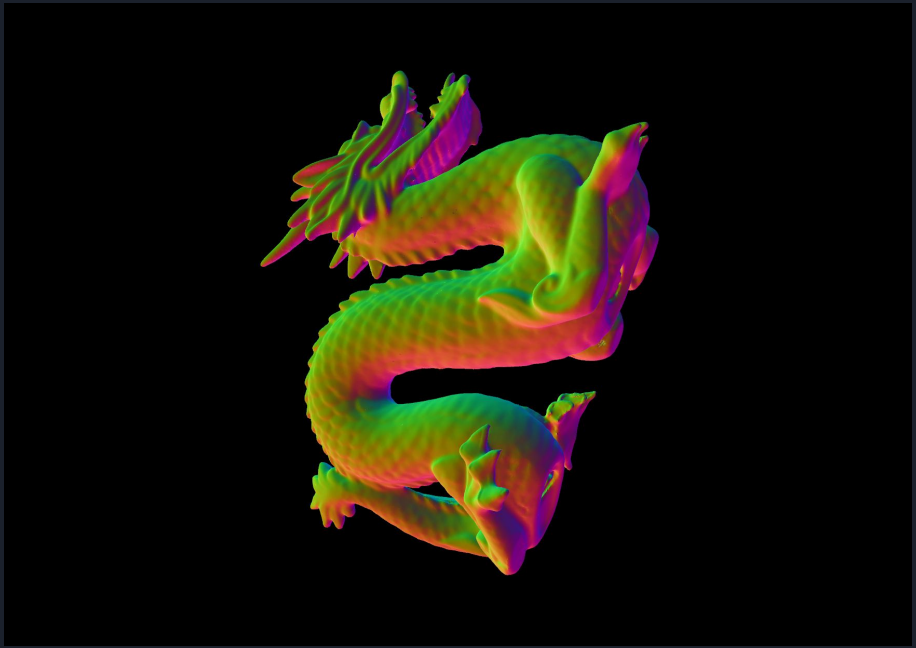
- Defining Lights and Materials
  - Defining Materials
  - Defining a Point Light
- Interactive Light Controls

Normal Coloring

- Color of object depends on object normals
- Each normal should be of unit length
- Range of each normal component is [-1, 1]. We want a color range from [0, 1]
- Conversion algorithm:
  - Normalize to unit length if necessary
  - 0.5 * normal + 0.5
  - *don’t do the inverse-transpose transform on the normals

Normal Color Example





The Phong Illumination Model

- Invented by Bui Tuong Phong in his 1973 PhD dissertation
- The 1st algorithm for simulating specular highlights

$$c = \sum_i c_{l_i}(k_d(L_i \cdot n) + k_s(R \cdot e)^p + k_a)$$

- See Piazza for an explanation of the equation above

Materials

- Define materials for each object application and shader side
  - Define application side (most conveniently as a data member of your 3D object class)
  - Define materials in shader as a set of uniform variables

```
class Materials {
private:
    glm::vec3 ambient, diffuse, specular;
    float shininess;

public:
    void sendMatToShader(const int shaderID);

};
```

```
uniform vec3 ambient;
uniform vec3 diffuse;
uniform vec3 specular;
uniform float shininess;
```

Material Requirements

- Each of the 3 models should have different appearances
  - One very shiny (high specular) with no diffuse
  - Another very diffuse with no specular
  - The last should have both diffuse and specular reflection
- Which object has which material characteristics is up to you

*Check this link for some interesting parameters, but keep in mind requirements

Example Renders

Point Lights

- A light source that radiates in all directions
- We add some extra properties to this
  - Attenuation strength
  - Light color
- Once again define lights in both application and shader

```
class PointLight {
private:
    /*
     * Attenuation vector consists of
     *     atten.x = constant factor
     *     atten.y = linear factor
     *     atten.z = quadratic factor
     */
    glm::vec3 pos, color, atten;

public:
    void sendLightToShader(const int shaderID);

};
```

```
uniform vec3 lightPos;
uniform vec3 lightCol;
uniform vec3 lightAtten;
```

Some Tips and Tricks

- To any Visual Studio users out there: check out the GLSL language integration
- REMEMBER THAT WE ARE ASKING FOR **LINEAR ATTENUATION**
- Any uniform that isn't set will be set to zero (if not initialized to some constant value at link time)
  - uniform vec3 color = vec3(0.7, 0.7, 0.2); // value assigned at link time
- Don't forget to keep your dot products clamped with a lowest value of 0
- You can't print values out in shaders, but you can set the output fragColor to that value to check it

Interactive Light Controls

- Light Representation
  - Display a sphere object at the location of the light
  - Color with light's ambient reflection value
  - Because light exists inside sphere
    - No diffuse
    - No specular reflection
  - Keep the sphere small, but visible (refer to example renders, slide 10)

Interactive Light Controls

- Light Position
  - Toggle between modes for controlling object/light

Mode	Mouse Button	Mouse Wheel (or similar)
'1'	Rotates 3D model about its center. Light source stays fixed for the viewer.	Scales 3D model about its center. Light source stays fixed for the viewer.
'2'	Rotates the light source around the center of the 3D model. 3D model stays fixed for the viewer.	Moves the light source closer to or further from the center of the 3D model.
'3'	Rotates both 3D model and light source by the same amount.	Scales the 3D model about its center and moves the light source like above.

Interactive Light Controls

- Light Position Mode 1 (Model only controls)
  - Mouse Button
    - Rotate 3d model
    - Do not rotate light source
  - Mouse Wheel
    - Scales 3d model
    - Do not move light source

Interactive Light Controls

- Light Position Mode 2 (Light only controls)
  - Mouse Button
    - Rotate light source
    - Do not rotate 3d model
  - Mouse Wheel
    - Moves light source closer/further from 3d model
    - Do not scale 3d model
- Pivot point is at (0, 0, 0)

Interactive Light Controls

- Light Position Mode 3 (Both controls)
  - Combines mode 1 and mode 2
  - Mouse Button
    - Rotate 3d model and light source by same amount
    - We want the lighting to stay the same on the object
  - Mouse Wheel
    - Scale 3d model about it’s center
    - Move light source closer/further from center

Submission

- Canvas
  - Add a comment on what you completed to Canvas
  - Files to submit: *.cpp, *.h, shader files, executable
- Video
  - Keep it concise, maximum 5 minutes
  - Show off all functionality required for points, but no need to go into the code
  - If there are issues attaching the video, you can always include it in the zip file

Any questions?