CS300: Assignment 2
Lighting Using GLSL

TA: Ben Henning <b.henning@digipen.edu>

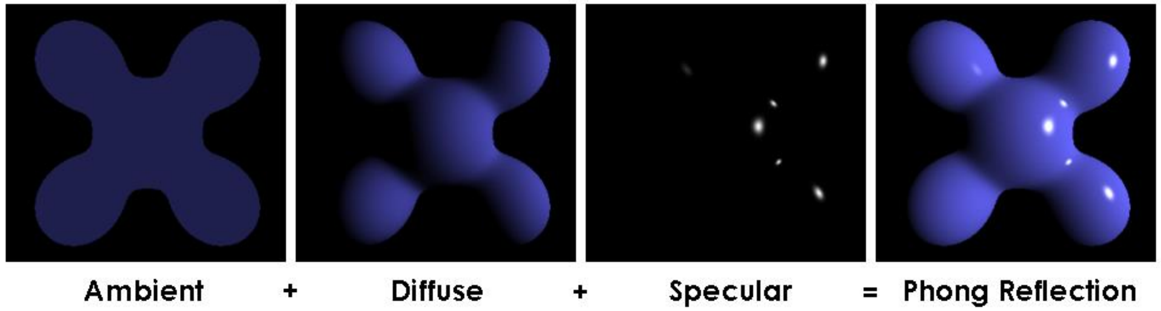
TA: Tyler Pugmire <tyler.p@digipen.edu>

Framework Errors and Feedback

- Comment reading *TODO(Ben)* instead of *TODO(student)* on Main.cpp:337
- Euler Angle rotation was wrong in the sample because x-rotation and z-rotation were flipped
- Scrolling in ImGUI with the scroll wheel does not work
- Dragging the scroll bar in ImGUI sort of spazzes out toward the bottom of the scroll
- A ShaderProgram is architecturally required and bound when building a VertexArrayObject, but this is not actually necessary
- Matrix4's BuildTransform function is a bit confusing on how to work with

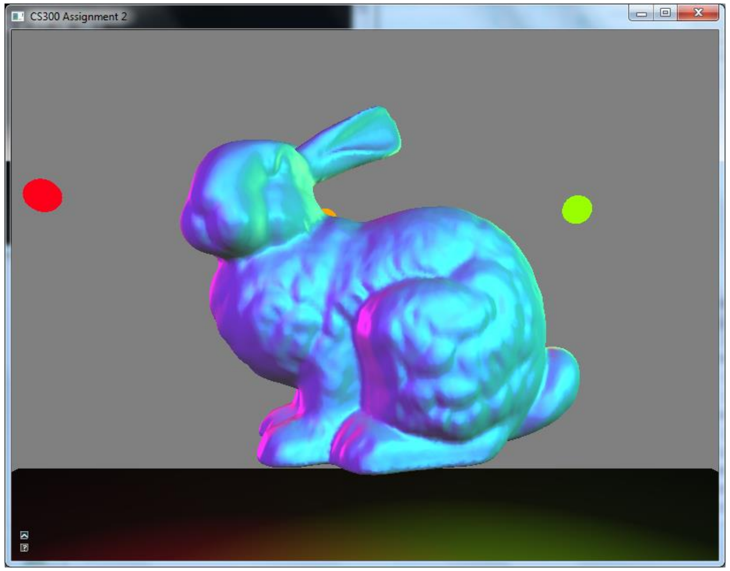
Introducing Assignment 2

- Implementing 3 lighting models
  - Phong Lighting using vertex interpolation along the polygon (also known as Gouraud Shading)
  - Phong Shading (perform per-fragment)
  - Blinn-Phong Shading (optimization of Phong Shading)
- Support the complete Phong Illumination Model
  - Ambient, Diffuse, and Specular lighting terms
  - Emissive, Ambient, Diffuse, and Specular material properties
  - Distance and atmospheric attenuations
  - Global ambience



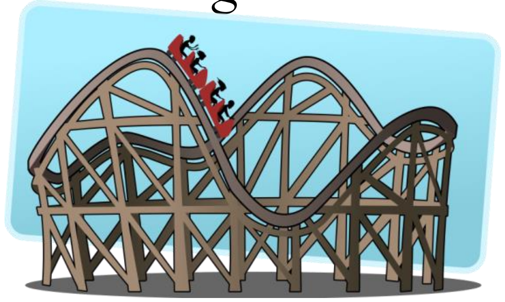
Introducing Assignment 2

- Support of the 3 major light types
  - Directional Lights
  - Point Lights
  - Spot Lights (including fall-off)
- Must support up to 8 active lights at any one time
- Support for diffuse and specular texture maps
- Support for cylindrical and spherical texture projection



Introducing Assignment 2

- Scene setup
  - Lights are debug drawn with spheres and support animated revolution around a loaded object
  - Have a plane underneath the object loaded
- Have 3 different lighting scenarios
  - One mode where the same light is duplicated
  - One mode where each light is individually tweakable
  - One mode where you manually setup all 8 lights and have them make an "interesting effect"
- Extra credit: implementing an animated camera that simulates a "roller-coaster" effect
- More details on the actual assignment 2 rubric; these are just the finer details



Changes in the Framework

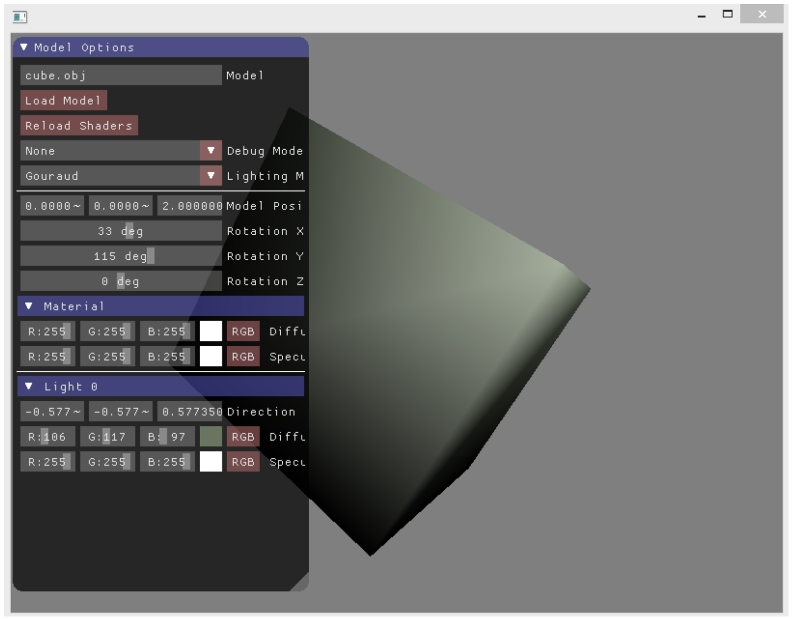
- Support for loading in textures from either TGA or PNG files
- Support for binding multiple textures for rendering
- Ability to attach the bound slot of a texture to a uniform location in shaders
- Fixes to the errors mentioned earlier
- Expect to see an up-to-date framework posted on distance in the coming week
- You do not need the updates to start working on the assignment

Sample

- There is no sample being provided this time, due to it being primarily shader development
- With gDEBugger (shown later in class), getting the shader code of a sample would be easy
- Instead, we will provide pictures of what each of the lighting models will look like
- Will also be showing what the diffuse and specular renders will look like using the provided diffuse and specular texture maps
- We may provide a video showing the light rotation and camera extra credit, but this is only if we have time to put that together

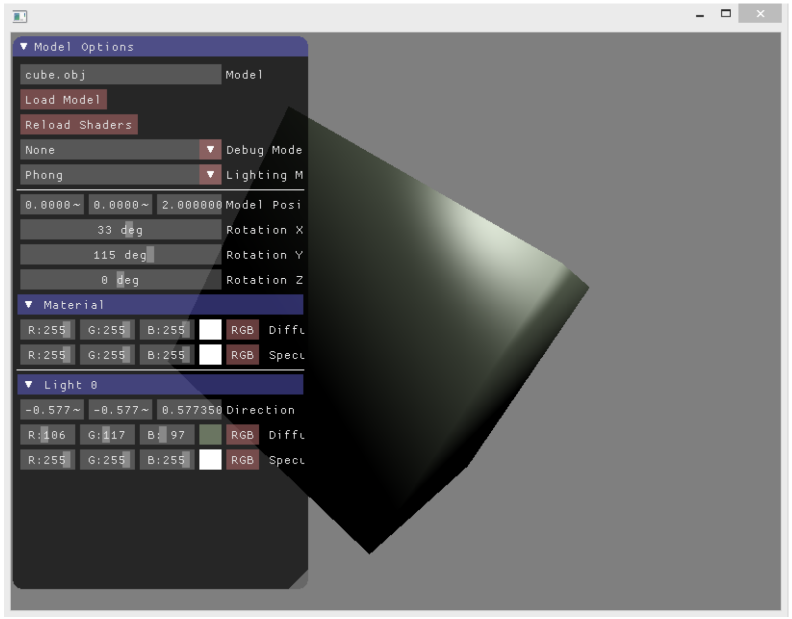
Lights in GLSL from Scratch Demo

Gouraud Shading



```
▼ Model Options
cube.obj                    Model
Load Model
Reload Shaders
None                    ▼   Debug Mode
Gouraud                 ▼   Lighting M
0.0000~  0.0000~  2.000000  Model Posi
         33 deg             Rotation X
        115 deg             Rotation Y
          0 deg             Rotation Z
▼ Material
R:255  G:255  B:255  RGB  Diffu
R:255  G:255  B:255  RGB  Specu
▼ Light 0
-0.577~  -0.577~  0.577350  Direction
R:106  G:117  B: 97  RGB  Diffu
R:255  G:255  B:255  RGB  Specu
```

Phong Shading



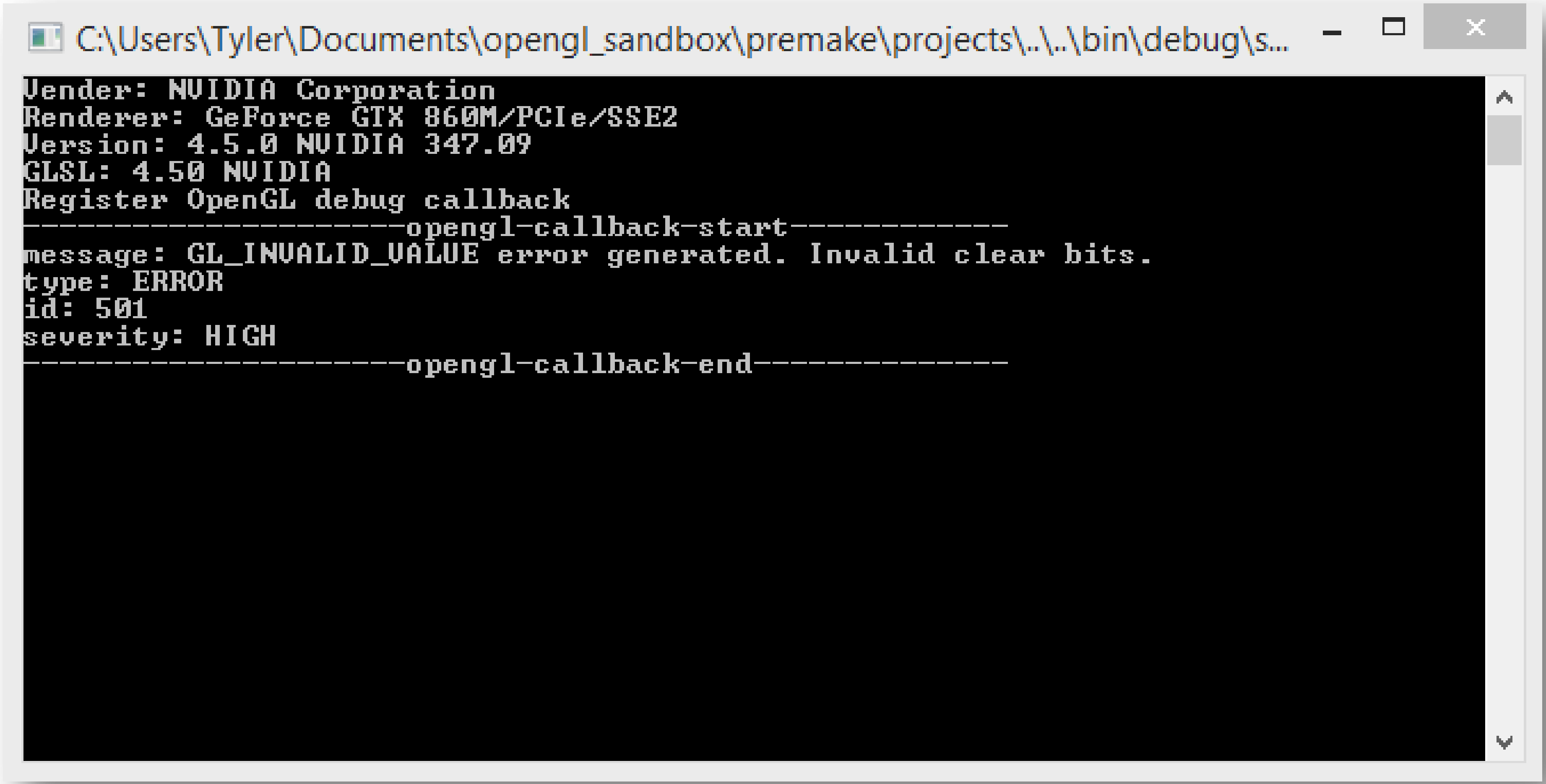
```
▼ Model Options
cube.obj                    Model
Load Model
Reload Shaders
None                    ▼   Debug Mode
Phong                   ▼   Lighting M
0.0000~  0.0000~  2.000000  Model Posi
         33 deg             Rotation X
        115 deg             Rotation Y
          0 deg             Rotation Z
▼ Material
R:255  G:255  B:255  RGB  Diffu
R:255  G:255  B:255  RGB  Specu
▼ Light 0
-0.577~  -0.577~  0.577350  Direction
R:106  G:117  B: 97  RGB  Diffu
R:255  G:255  B:255  RGB  Specu
```

Debugging in OpenGL

Debugging in OpenGL: gDEBugger

- gDEBugger is an excellent OpenGL debugging application supplied by Graphic Remedy
- Used to cost money, but is now free and cross-platform!
- http://www.gremedy.com/
- Supports:
  - Debugging buffer contents currently on the GPU
  - Viewing shader code uploaded to GPU
  - Verify the current context is valid
  - Check for redundant OpenGL calls
  - Do profile and performance analysis for OpenGL calls
  - Check for wrongful usage of OpenGL functions
  - Memory analysis and checks for graphics memory leaks
- Probably the best way of debugging for OpenGL if not using OpenGL 4

Debugging in OpenGL 4: Callbacks


C:\Users\Tyler\Documents\opengl_sandbox\premake\projects\..\..\bin\debug\s...

```
Vender: NVIDIA Corporation
Renderer: GeForce GTX 860M/PCIe/SSE2
Version: 4.5.0 NVIDIA 347.09
GLSL: 4.50 NVIDIA
Register OpenGL debug callback
---------------------opengl-callback-start------------
message: GL_INVALID_VALUE error generated. Invalid clear bits.
type: ERROR
id: 501
severity: HIGH
---------------------opengl-callback-end--------------
```

Debugging in OpenGL 4: Callbacks

- Added in OpenGL 4.3
- Allows for verbose output on OpenGL errors (similar to DirectX)
- Allows debugging of parts of OpenGL that were not even possible to effectively debug in the past
- OpenGL 4.5 also has built-in query performance metrics for highly accurate profiling:
- https://www.opengl.org/wiki/Query_Object

Debugging in OpenGL 4: Callbacks

```
void GLAPIENTRY debugCallback(GLenum source, GLenum type, GLuint id,
  GLenum severity, GLsizei length, const GLchar* message, const void* userParam)
{

  std::cout << "---------------------opengl-callback-start------------" << std::endl;
  std::cout << "message: " << message << std::endl;
  std::cout << "type: ";
  switch (type)
  {
  case GL_DEBUG_TYPE_ERROR:
    std::cout << "ERROR";
    break;
  case GL_DEBUG_TYPE_DEPRECATED_BEHAVIOR:
    std::cout << "DEPRECATED_BEHAVIOR";
    break;
  case GL_DEBUG_TYPE_UNDEFINED_BEHAVIOR:
    std::cout << "UNDEFINED_BEHAVIOR";
    break;
  case GL_DEBUG_TYPE_PORTABILITY:
    std::cout << "PORTABILITY";
    break;
  case GL_DEBUG_TYPE_PERFORMANCE:
    std::cout << "PERFORMANCE";
    break;
  case GL_DEBUG_TYPE_OTHER:
    std::cout << "OTHER";
    break;
  }
  std::cout << std::endl;

  std::cout << "id: " << std::hex << id << std::endl;
  std::cout << "severity: ";
  switch (severity)
  {
  case GL_DEBUG_SEVERITY_LOW:
    std::cout << "LOW";
    break;
  case GL_DEBUG_SEVERITY_MEDIUM:
    std::cout << "MEDIUM";
    break;
  case GL_DEBUG_SEVERITY_HIGH:
    std::cout << "HIGH";
    break;
  }
  std::cout << std::endl;
  std::cout << "---------------------opengl-callback-end--------------" << std::endl;
}

if (glDebugMessageCallback){
  std::cout << "Register OpenGL debug callback " << std::endl;
  glEnable(GL_DEBUG_OUTPUT_SYNCHRONOUS);
  glDebugMessageCallback(debugCallback, nullptr);
  GLuint unusedIds = 0;
  glDebugMessageControl(GL_DONT_CARE, GL_DONT_CARE, GL_DONT_CARE, 0,
    &unusedIds, true);
}
else
  std::cout << "glDebugMessageCallback not available" << std::endl;
```

- Feel free to look at this code outside the presentation
- Copy code on write into your favorite text editor to learn more
- No example provided for query objects, but many examples available online

Wrapping Up

Checklist

- Before turning in, make sure your application:
  1. Follows all of the points in slide 3 (introduction to Assignment 2)
  2. Make sure you have completely satisfied the requirements of the posted Assignment 2 rubric, which is more detailed than slide 3 (and directly what we are grading off of)
  3. Make sure your application supports all the features demonstrated in the sample (see the sample's README)
- Do not submit the premake or sample folders
- **Remember to run clean.bat before archiving; build artifacts deduct points**
- If you changed premake4.lua, please email one of us
- Be sure to read the framework's README and update it
- Submission name format:

  digipen.login_cs300_2.zip

Final Remarks

- Assignment 2 may involve a lot more hands-on work if you are using the framework
- As always, email any of us if you need help with OpenGL or the framework
- Both of the TAs are on a team occupying the team space near the cafeteria stairwell in 3rd floor Tesla and sit there frequently
- Ben also has tutor hours listed for additional help
- Again, you do **not** have to use the framework
- Good luck!

Questions & Comments