Final Project Report

https://gitlab.com/raymww/raytracing

Project Background:

Our final project is on non-photorealistic lighting. We decided to do non-photorealistic lighting because it was intriguing to us to learn more about all the different methods to implement lighting and the different techniques needed to do as such. Additionally, because there was already a code base for the project, as we built off the ray tracing project. This meant that we would have an existing testing framework for easier debugging. We decided to implement the Cel shading model and Technical Drawing shading.

Non-photorealistic lighting is about changing shader models to render objects in ways that aren't photorealistic, with light and shadows defining darker and brighter colors, such as with the Phong shading model. These non realistic models are important for those trying to render artistic styles that are difficult to replicate on a computer. For example, Cel shading attempts to replicate a cartoon art style, which replicates the brush strokes of artists through strictly defined color levels, instead of a gradient. Additionally, Technical Drawing shading takes the original shadows and highlights defined by luminance and changes it to be defined by hue, in order to preserve the full shape of the object, which is extremely important for technical drawings.

What We Accomplished:

We have implemented both Cel Shaders as well as the Technical Drawing Shader. We have also added the GUI elements to render any scene in all 3 different lighting models.

We implemented the cel shader by taking the diffuse and specular values already calculated by the phong lighting model and dividing them into levels. To do this, we first multiply the value by the amount of levels, then floor the value, and lastly divide the value by the number of levels. This will separate the color values from diffuse and specular into levels of the same color, creating the toon effect. We implemented outlines by first defining outlines as surfaces facing away from the camera. We calculated the outline by taking the dot product of the normal and the ray direction, and inverting it.

```
else if(traceUI->cel()) {
   glm::dvec3 sum = glm::dvec3(0.0, 0.0, 0.0);
   for ( const auto& pLight : scene->getAllLights() )
   {
     glm::dvec3 temp = glm::dvec3(0.0, 0.0, 0.0);
     // diffuse term
     double diffuse = pLight->distanceAttenuation(r.at(i)) *
max(glm::dot(pLight->getDirection(r.at(i)), i.getN()), 0.0);
     temp += kd(i) * (floor(diffuse * 4.0) / 4.0);
```

```
    // specular term
    double specular = pLight->distanceAttenuation(r.at(i))
      * pow(max(glm::dot(glm::reflect(pLight->getDirection(r.at(i)) * -1.0,
i.getN()), r.getDirection() * -1.0), 0.0), shininess(i));
    temp += ks(i) * (floor(specular * 4.0) / 4.0);
    ray shadowRay(r.at(i) + pLight->getDirection(r.at(i)) * RAY_EPSILON,
pLight->getDirection(r.at(i)), glm::dvec3(0, 0, 0),
        ray::SHADOW);

    isect shadowI;
    if(scene->intersect(shadowRay, shadowI)) {
      temp *= pLight->shadowAttenuation(shadowRay, shadowI);
    }
    temp *= pLight->getColor();
    sum += temp;

  }
  double rimDot = 1 - glm::dot(r.getDirection(), -1.0 * i.getN());
  return ke(i) + (ka(i) * scene->ambient()) + sum - glm::smoothstep(0.79, 0.81,
rimDot) * glm::dvec3(1.0, 1.0, 1.0);
```

We implemented the technical drawing shader by changing the diffuse lighting. In the Phong model, shadows get darker and lose detail. To preserve this detail for a technical drawing, where the full geometric shape is vitally important, we use a warm color shift for bright areas and cool color for darker areas. We implemented this through interpolating the diffuse light with a warm / cool light shift and shading appropriately. This way, there is still some luminance shift but the hue shift shows light and shadow without a loss of detail. Additionally, we modified the shadow ray test to not completely black out anything in shadow to conform with this approach.

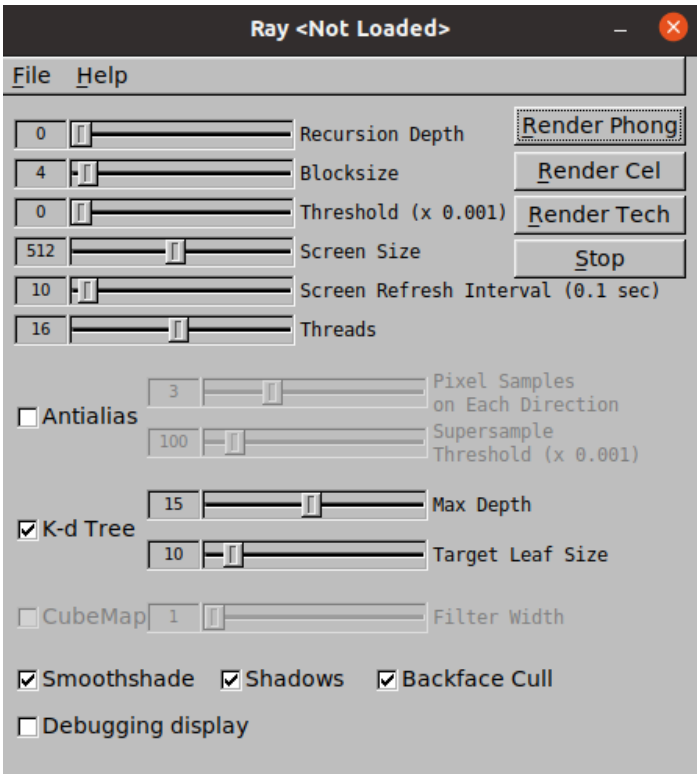
```
glm::dvec3 sum = glm::dvec3(0.0, 0.0, 0.0);
  for ( const auto& pLight : scene->getAllLights() )
  {
    // diffuse term
    glm::dvec3 temp = glm::dvec3(0.0, 0.0, 0.0);
    // double diffuse = pLight->distanceAttenuation(r.at(i)) *
max(glm::dot(pLight->getDirection(r.at(i)), i.getN()), 0.0);
    double dot = glm::dot(pLight->getDirection(r.at(i)), i.getN());
    double blue = 0.6;
    double yellow = 0.8;
    double alpha = 0.2;
    double beta = 0.2;
```

```
    glm::dvec3 kCool = glm::dvec3(0.0, 0.0, blue) + (alpha * kd(i));
    glm::dvec3 kWarm = glm::dvec3(yellow, yellow, 0.0) + (beta * kd(i));
    glm::dvec3 techDiffuse = ((1 - ((1 - dot) / 2)) * kWarm) + ((1 + ((1 - dot) / 2))
* kCool);
    temp += techDiffuse;
    sum += temp;
    break;
  }
  double rimDot = 1 - glm::dot(r.getDirection(), -1.0 * i.getN());
  return ke(i) + (ka(i) * scene->ambient()) + sum - glm::smoothstep(0.79, 0.81,
rimDot) * glm::dvec3(1.0, 1.0, 1.0);
```

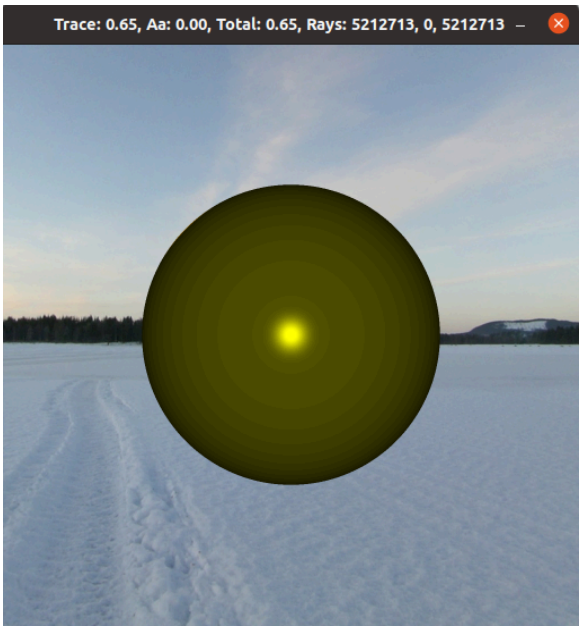
Finally, we also added GUI buttons to render scenes from the project in any of the three lighting methods.

Artifacts We Produced:

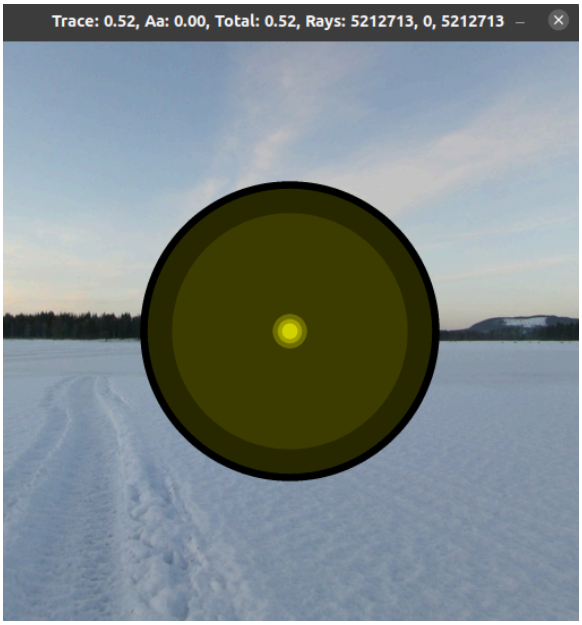
GUI with elements



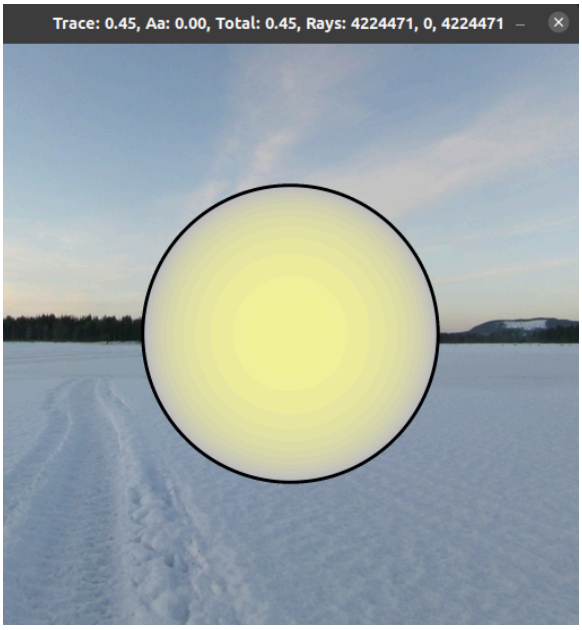
Phong



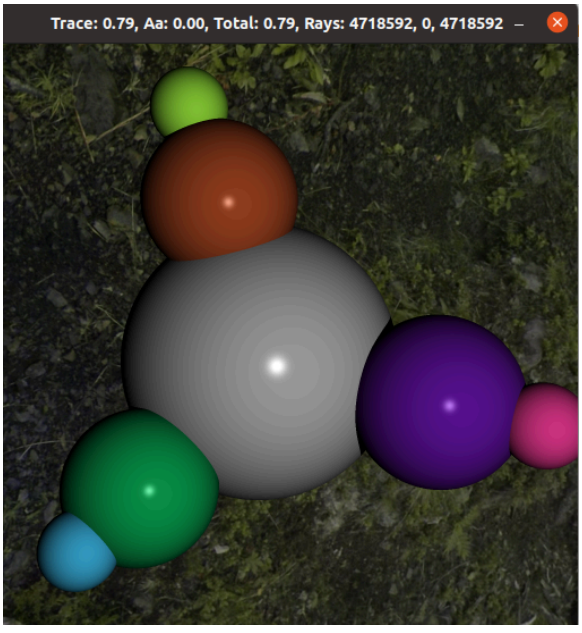
Cel Shading



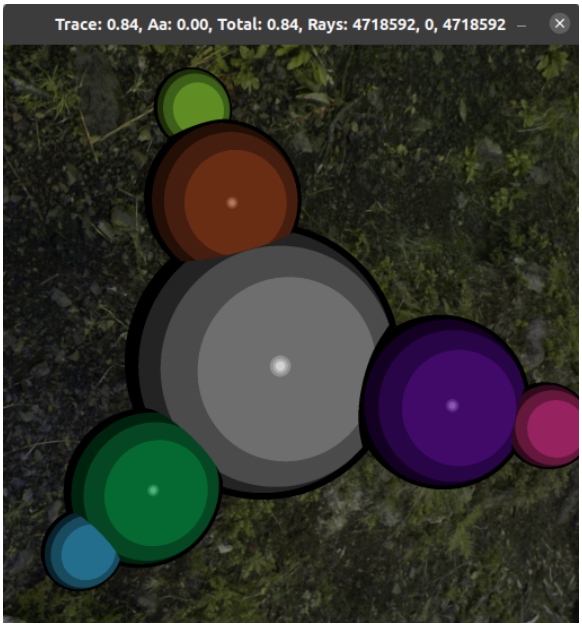
Technical Drawing Shading



Phong

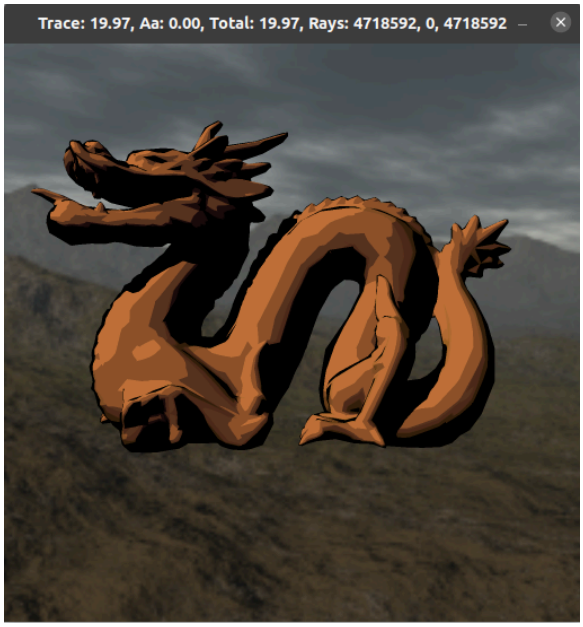


Cel Shading

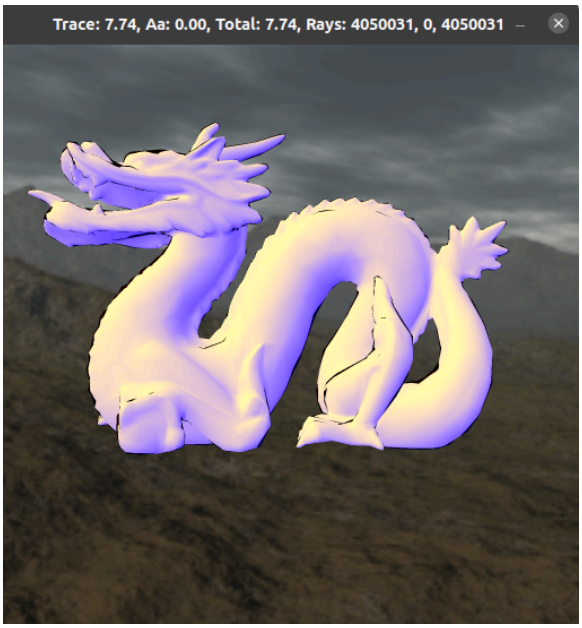


Phong

Cel Shading



Technical Drawing Shading



Sources

Cel Shading:
https://en.wikipedia.org/wiki/Cel_shading
https://dev.to/longevitysoftware/cel-toon-shading-103m
https://roystan.net/articles/toon-shader/

Technical Drawing Shading:
https://dl.acm.org/doi/10.1145/280814.280950