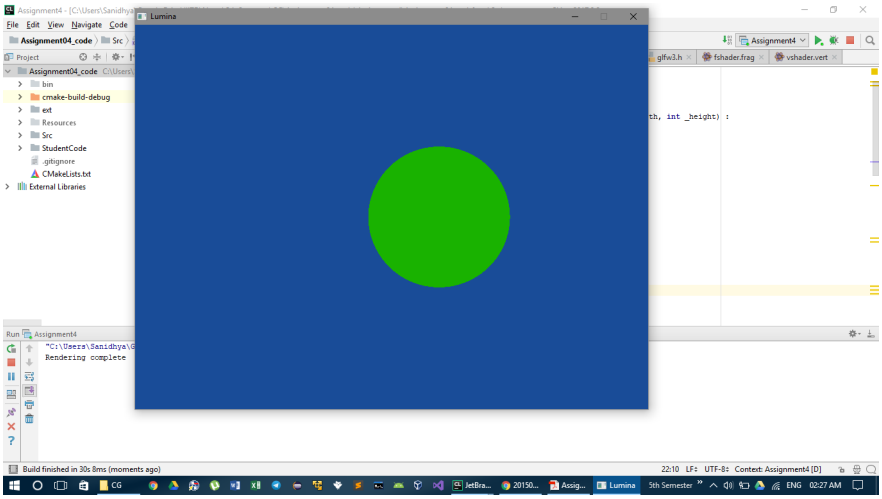
Assignment 4: Raytracing

SANIDHYA SINGAL, Indraprastha Institute of Information Technology Delhi

Initial Run:



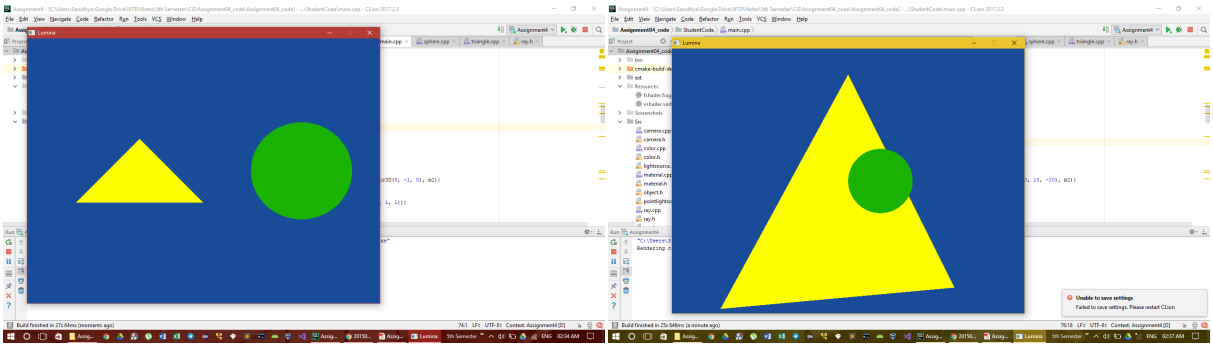
ANS 1

A `Triangle` class is created by extending the `Object` class. The corresponding files are `triangle.cpp` and `triangle.h`. A triangle is a parametric surface. So, here, we intersect the incident ray with a parametric surface, unlike the sphere, which was an implicit surface. We utilize the barycentric coordinates (β, γ) of the triangle vertices. The equation is given by:

$$\mathbf{e} + t\mathbf{d} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$$

Here, $\mathbf{a}, \mathbf{b}, \mathbf{c}$ are the vertices of the triangle, $\mathbf{e}$ and $\mathbf{d}$ are the origin and direction of the incident ray respectively.

The ray intersects the triangle if and only if $\beta > 0$, $\gamma > 0$ and $\beta + \gamma < 1$. We have 3 unknowns: t, β, γ. The values for these are computed as described in the text and with the help of Cramer's rule. All the computation is done in the `Triangle.intersect()` function.
Attached are the screen-shots of triangles hence generated. The two figures have different values of vertices for the triangles.



ANS 2

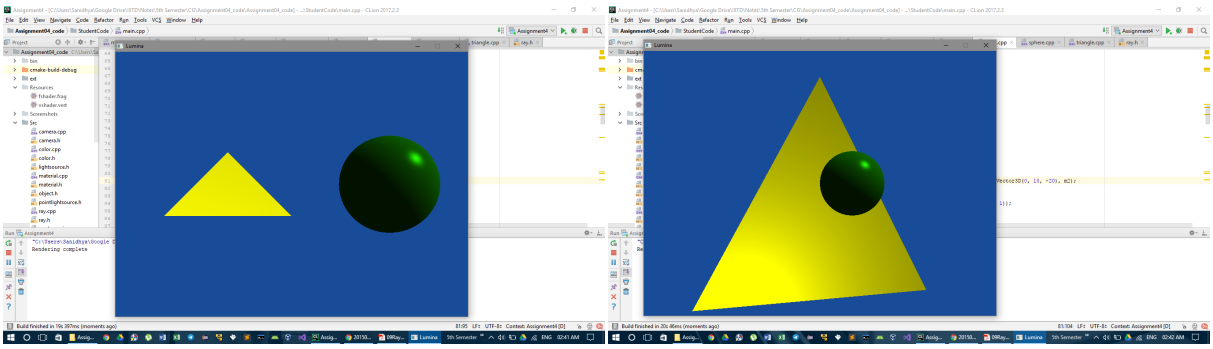
Phong shading has 3 components:

(1) **Diffuse**: $L_d = k_d I \max(0, \mathbf{n} \cdot \mathbf{l})$
(2) **Specular**: $L_s = k_s I \max(0, \mathbf{n} \cdot \mathbf{h})^{\alpha}$
(3) **Ambient**: $L_a = k_a I_a$

The final color is given by: $L = L_d + L_s + L_a$
Here, $k_d, k_s, k_a, I, I_a, \alpha$ are already known to us. We need to compute $\mathbf{n}, \mathbf{l}$ and $\mathbf{h}$. Also, $\mathbf{v}$ is computed as it is needed to compute $\mathbf{h}$.

```
v = -unitVector(incident.getDirection());
n = unitVector(incident.getNormal());
l = unitVector(world->getLightSource()[0]->getPosition() - incident.getPosition());
h = unitVector(v+l);
```

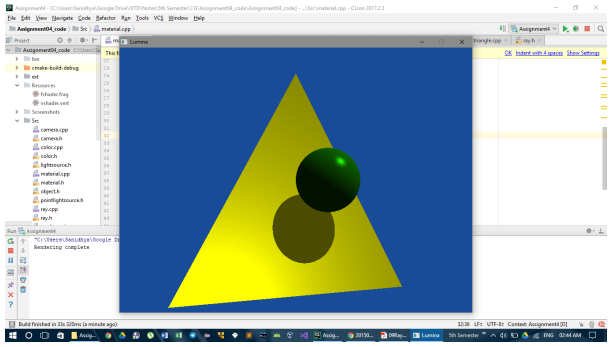
All the changes were made in `material.cpp` file. Following are the screenshots:



ANS 3

Shadows are drawn with the help of *shadow feeler* rays. These rays are the extensions of incident rays, which originate from the point of incidence and move towards the light source. In case, there is an obstacle in their path, we say that the obstacle is in the shadow of the incident point. Doing this for every incident ray, we are able to perceive shadows in the raytracer.
In order to prevent the drawing of shadow of the incident point itself, we originate the shadow feeler ray from a

little distance further from the point of incidence. This is common written as ε. Hence, ray's origin is $\mathbf{p} + \varepsilon\mathbf{l}$ and its direction is $\mathbf{l}$.
In the figure below, the shadow of sphere is seen on the triangle.



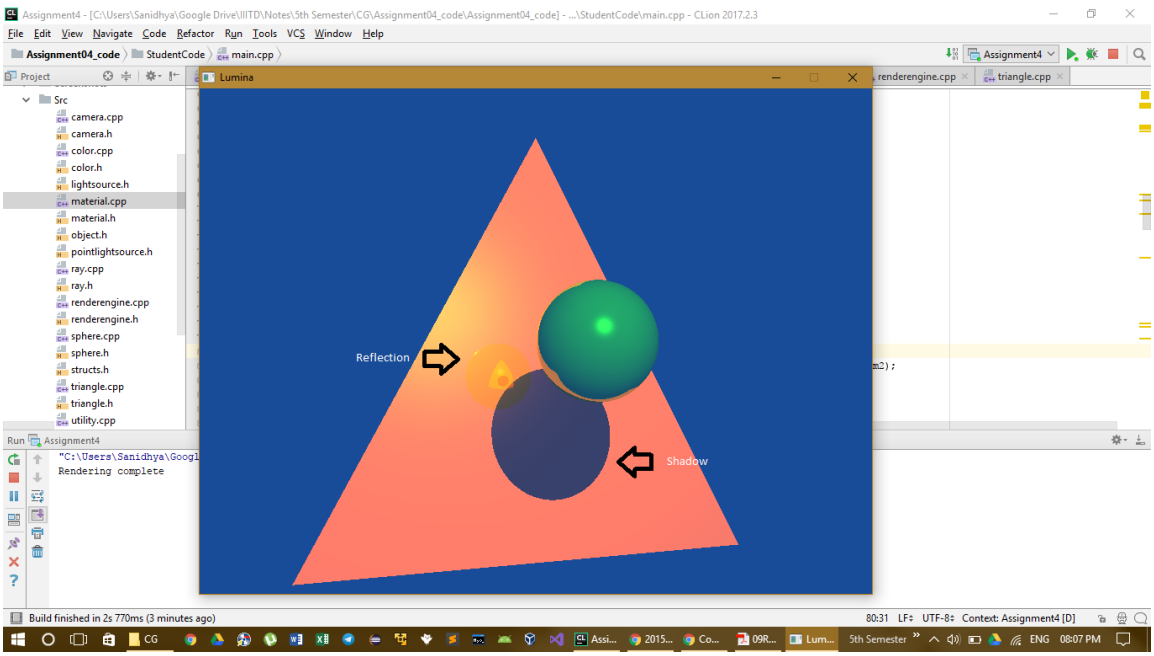
ANS 4

Reflection

Specular reflections are added in the raytracer by adding reflected rays as follows:

$$\mathbf{r} = \mathbf{d} - 2(\mathbf{d} \cdot \mathbf{n})\mathbf{n}$$

Here, $\mathbf{r}$ is the reflected ray drawn at the point of incidence. Recursion has been done by modifying the `shade_ray()` function in `world.cpp`. The depth is chosen to be 5. Attached is the screenshot:



[1]

[1] All the screenshots are included in `Assignment04_code\Assignment04_code\Screenshots`

1 BONUS ANS 1

Cylinder has been implemented as described in http://woo4.me/wootracer/cylinder-intersection/. Attached is the screenshot: