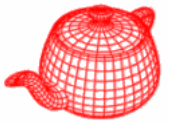
Project #4: Ray Tracer

Assign: 9/3

Due: 11:59pm 9/8

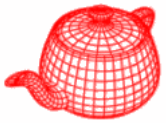
Submission: send all your java sources in a zip file and send it to me. Note that the project is accumulated. You may update files in previous projects. Thus, for each submission, please include all files including ones from previous projects.

PointLight

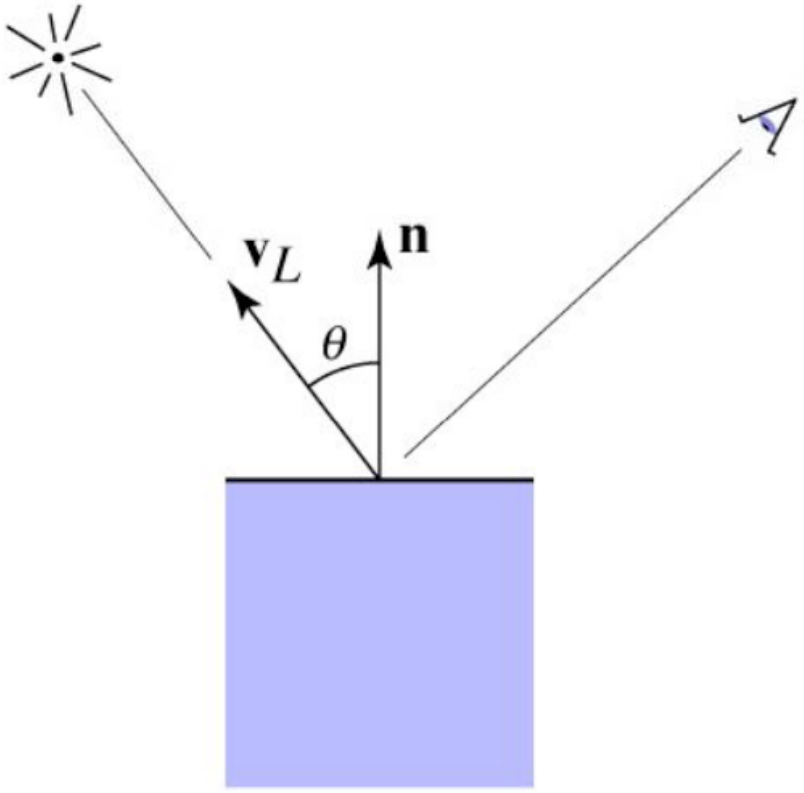


```
public class PointLight {
   Point p;
   RGB intensity;
}
```

Diffuse shading (Gouraud 1971)



- Applies to *diffuse*, *Lambertian* or *matte* surfaces



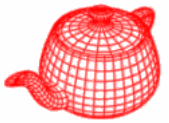
illumination from source

$$L_d = k_d \, I \, \max(0, \mathbf{n} \cdot \mathbf{v}_L)$$

diffuse coefficient (albedo)

diffusely reflected light

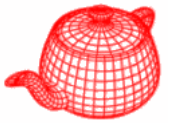
Shape



- Add material into shape

```
public abstract class Shape
{
  RGB rd;
  …
  public RGB getMaterial() {
    return new RGB(rd);
  }
}
```

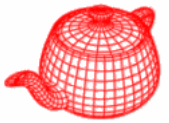
Modify constructors for material



- For example,

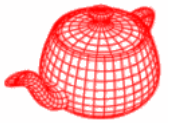
```
Sphere(Point p, float r, RGB rd);
```

Scene file



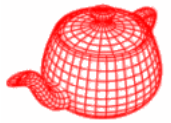
```
Perspective -0.5 0.5 0.375 -0.375 640 480
PointLight 10.0 10.0 0.0 100.0 100.0 100.0
Sphere -1.0 -1.0 10.0 1.0 0.6 0.75 1.0
Triangle -5.0 1.1339746 15.696152 5.0 -3.8660254
  7.035898 -5.0 -3.8660254 7.035898 0.0 0.8660254 -
  0.5 0.0 0.8660254 -0.5 0.0 0.8660254 -0.5 0.8 0.8
  0.8
```

Scene file



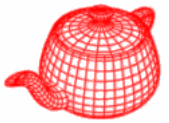
- `Perspective left right top bottom width height`
- `Orthographic left right top bottom width height`
- `PointLight x y z r g b`
- `Sphere x y z radius r g b`
- `Triangle p1 p2 p3 n1 n2 n3 r g b`
- `(p1 is a point; n1 is a vector; both consist of three float)`

RayTracer



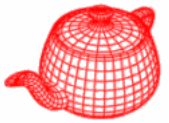
```
public class RayTracer {
  Camera camera;
  Shape[] shape;
  PointLight[] light;

  int ShapeCount;
  int LightCount;
}

Java RayTracer scene_file image_file
```

Main loop



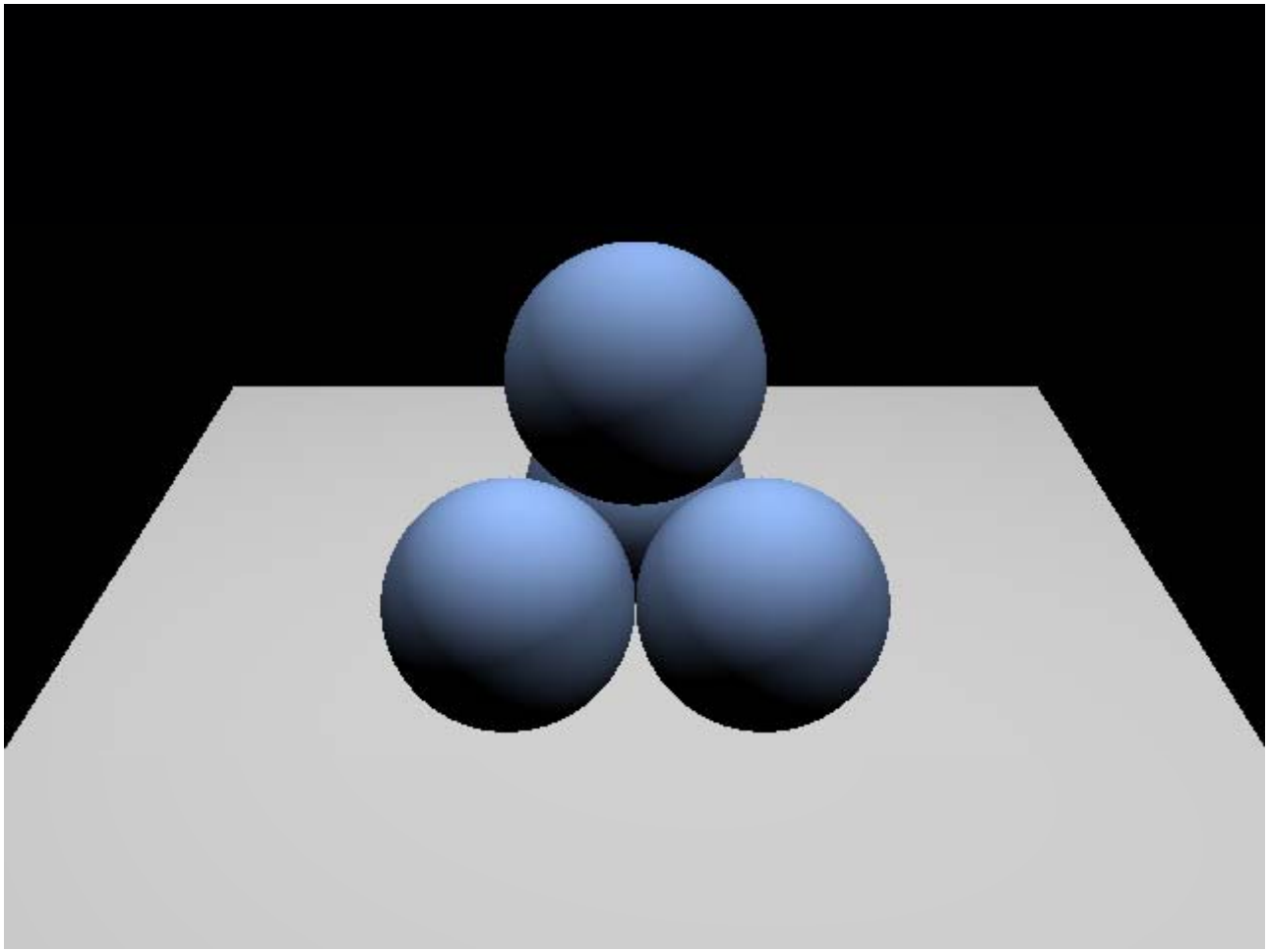
```
for (y=0; y<height; y++)
  for (x=0; x<width; x++) {
    Ray r=camera.GenerateRay(x, y);
    Intersection isect=null,new_isect;
    for (i=0; i<ShapeCount; i++) {
      new_isect=shape[i].Intersect(r);
      if (new_sect!=null)
        isect = new_isect;
    }
    if (isect!=null)
      camera.AddSample(x, y,
        isect.Shade(light, LightCount))
  }
```

Shade intersections

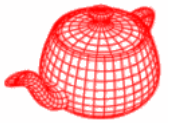


```
public class Intersection {
  …
  RGB Shade(PointLight[] light,
            int count)
}

  for (int i=0; i<count; i++) {
    …
    if (dot(n,lv)>0)
      shade += …
  }

  return shade;
```

Example



Notes



- You could support add, mul and scaling for RGB
- Be careful about checking **`(s1+s2+s3)==1`**
- It is usually better to check whether

  `abs(s1+s2+s3-1)<threshold`