Computer Graphics

Tutorial week 4

Today

- Your presentations
- OpenGL: 3D meshes and Vertex Arrays
- Raytracer: shadows, multiple light sources, reflection/refraction

Your presentations

OpenGL

- Addition of 3D meshes to your scene
- Using Vertex Arrays and Vertex Buffer Objects

3D meshes: glm.c/glm.h

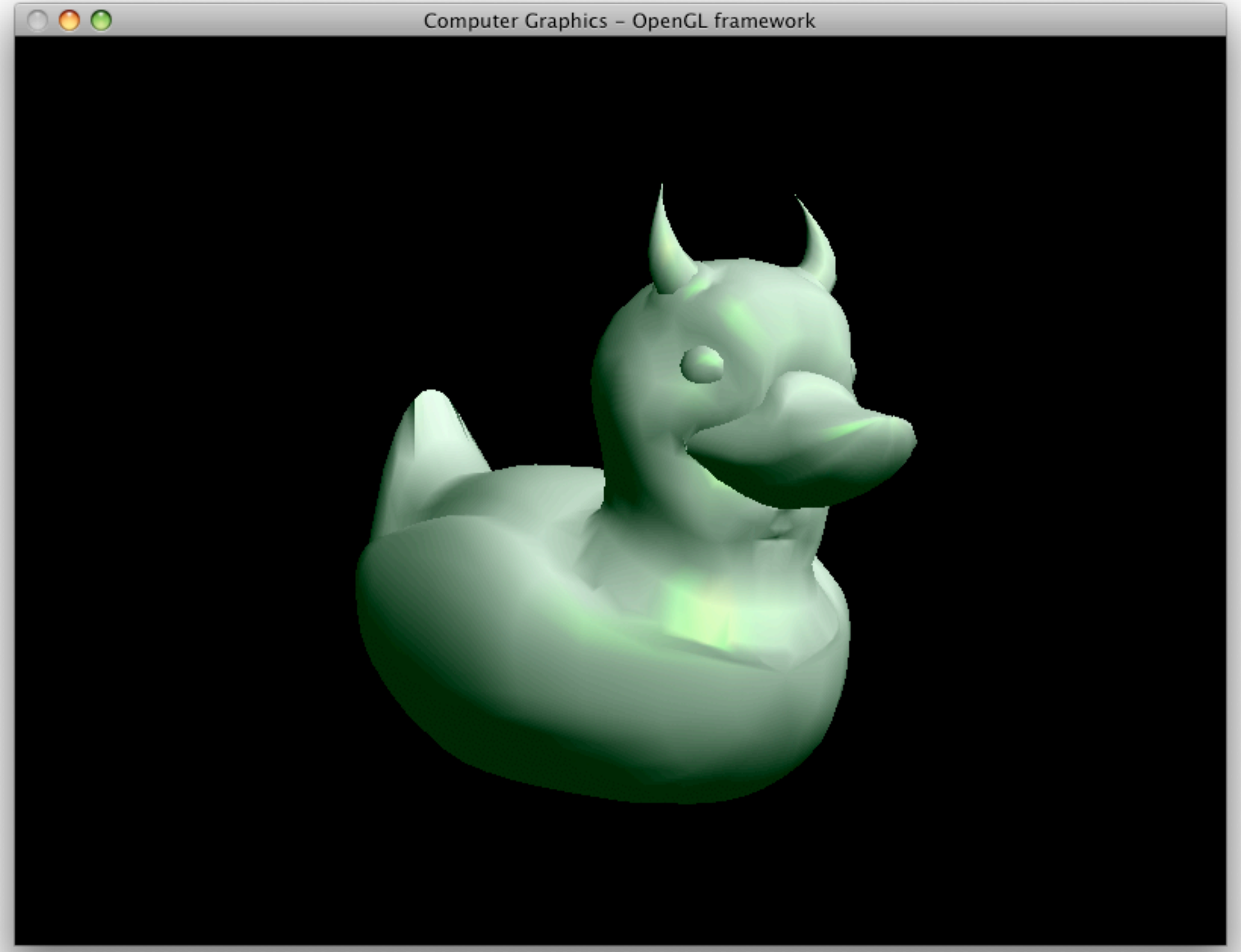
- Look at the files `glm.h` and `glm.c` to see what is implemented for you
  - `glmUnitize()`
  - `glmFacetNormal()`
  - `glmVertexNormals()`

3D meshes

- `glm.c/glm.h` from GLUT examples
- Use `glmDraw()` to test whether loading models works
- Create a draw function that uses Vertex Arrays and VBOs

OpenGL: Vertex Arrays

- The 'Modern way' of specifying geometry in OpenGL
- Follow the links in the assignment
- Think about how to reorganize the data
- You can keep it simple


Computer Graphics - OpenGL framework

Intersection calculation

From: http://www.cs.unc.edu/~rademach/xroads-RT/RTarticle.html

Intersecting a Sphere

For each type of object that the ray tracer supports, you need a separate intersection routine. We will limit our ray tracer to handling spheres only, since the intersection routine for a sphere is among the simplest. The following derivation of a sphere intersection is based on [Glassner90], page 388.

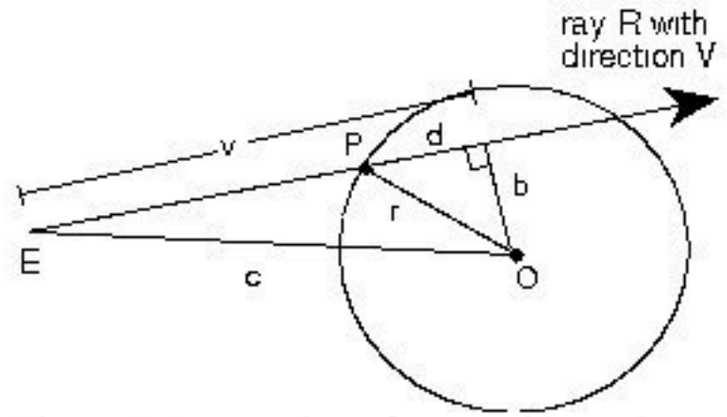


Figure 7. Intersection of a ray with a sphere.

Fig. 7 shows the geometry of a ray **R** (with origin at **E** and direction **V**) intersecting a sphere with center at **O** and radius **r**. According to the diagram,

$v^2 + b^2 = c^2$ and $d^2 + b^2 = r^2$ (by simple geometry)

and so

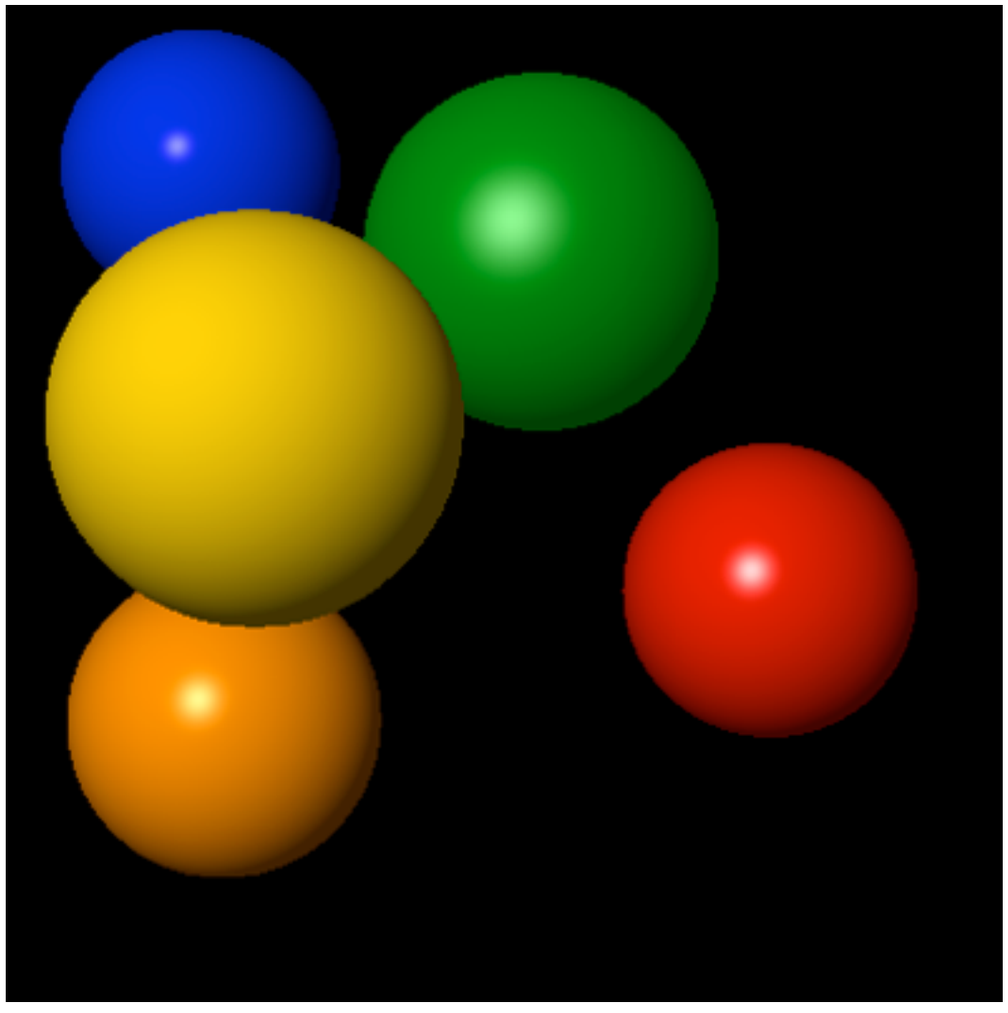
$d = \mathbf{sqrt}(r^2 - (c^2 - v^2))$

To determine whether an intersection occurs, we compute the value of **d**. If **d** >= 0, then a valid intersection occurs. If the ray does not intersect, then **d** will be less than zero. After finding the value of **d**, the distance from **E** to the intersection point **P** is (**v** - **d**). The pseudocode for all this is:

```
v = dot_product( EO, V )
disc = r^2 - ((dot_product( EO, EO ) - v^2 )
if ( disc < 0 )
     no intersection
else
     d = sqrt( disc )
     P = E + (v - d) * V
```

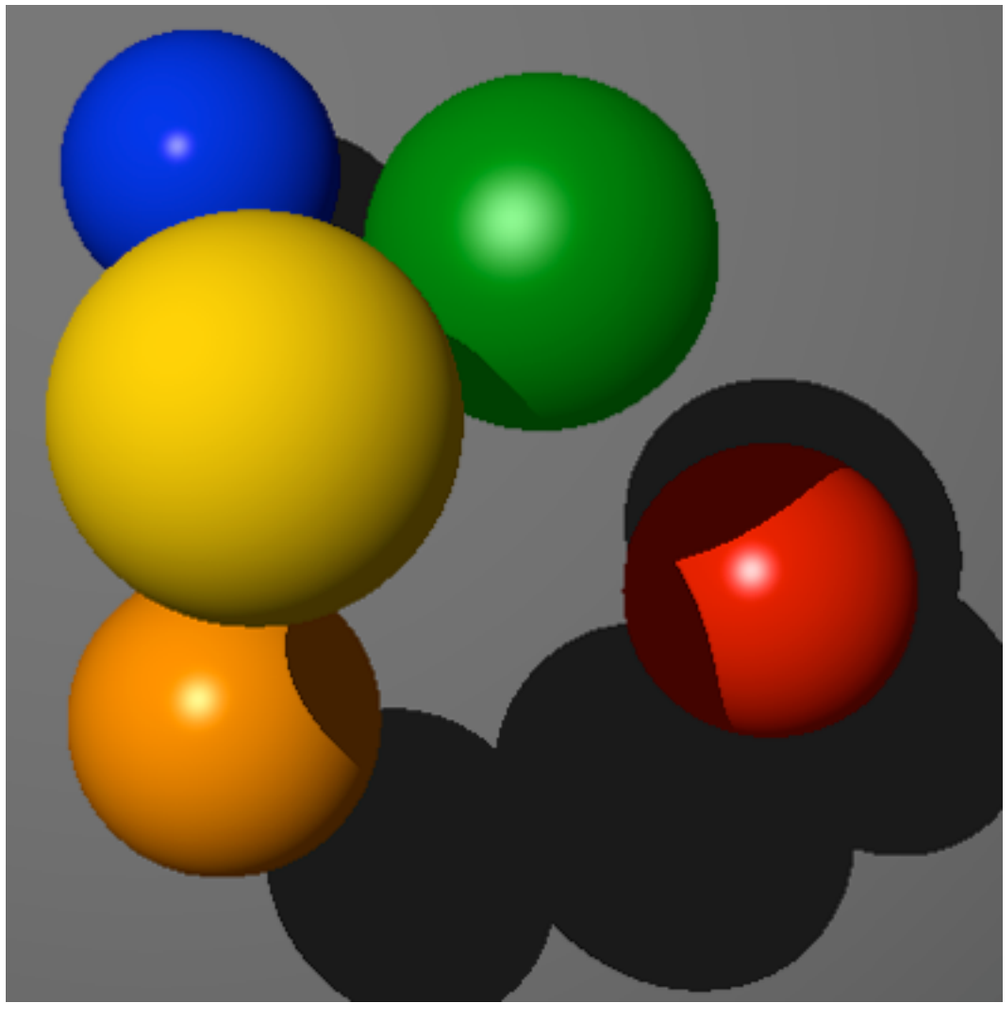
Raytracer: optical laws

- Implement shadows
- More than one light source
- Reflection
- Refraction



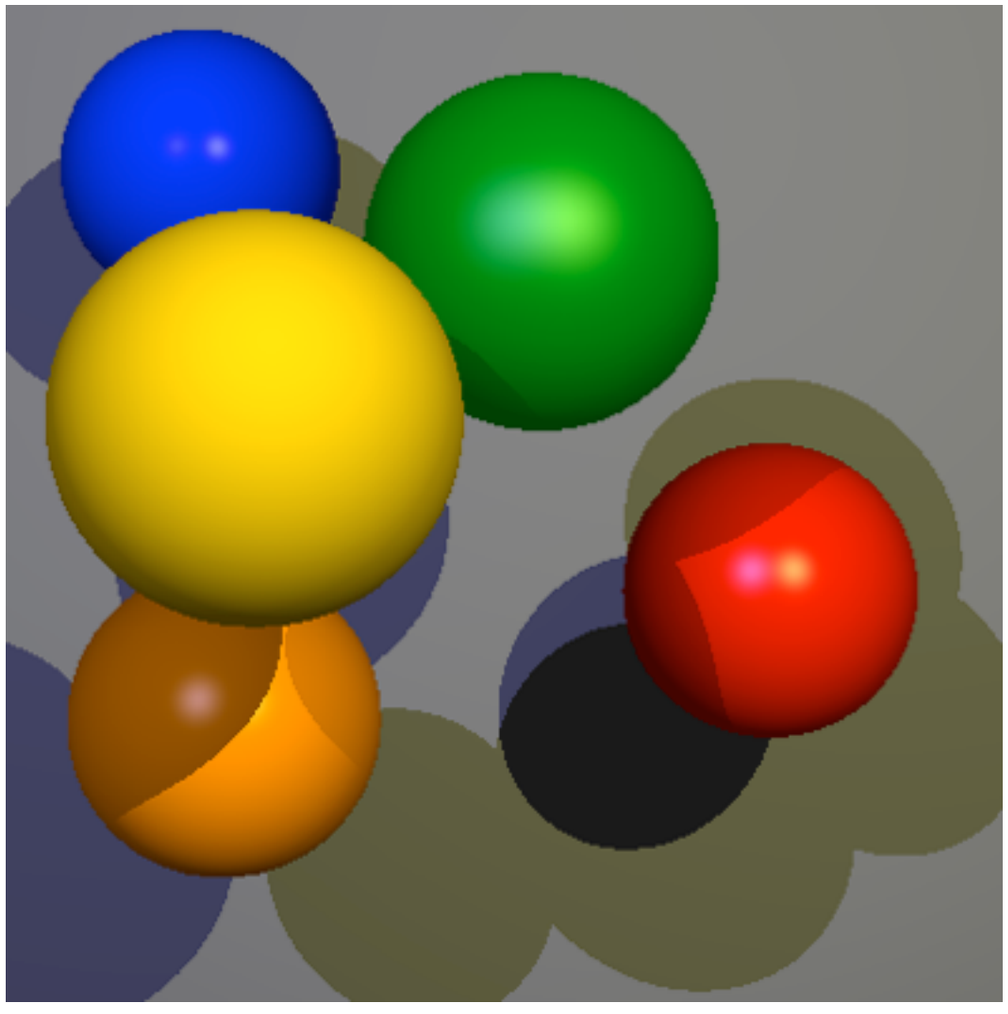
Raytracer: optical laws

- Implement shadows
- More than one light source
- Reflection
- Refraction



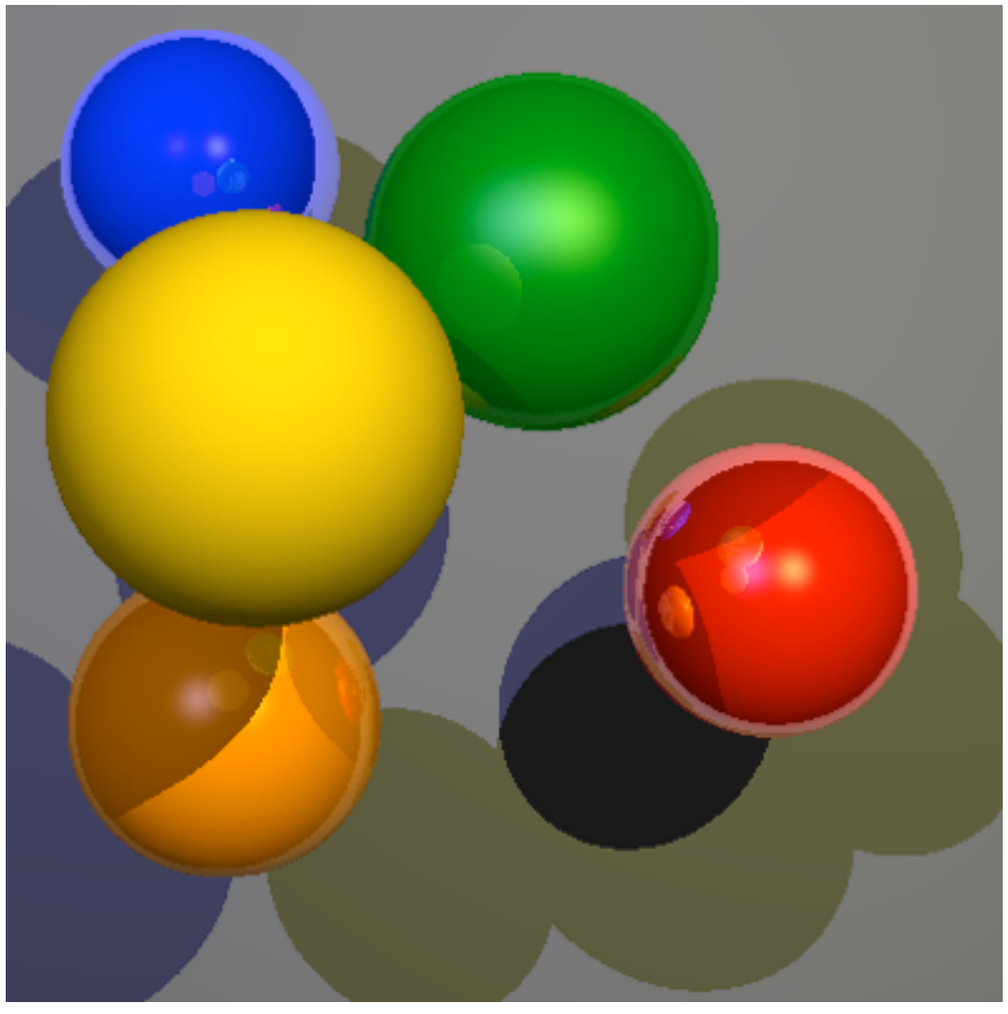
Raytracer: optical laws

- Implement shadows
- More than one light source
- Reflection
- Refraction



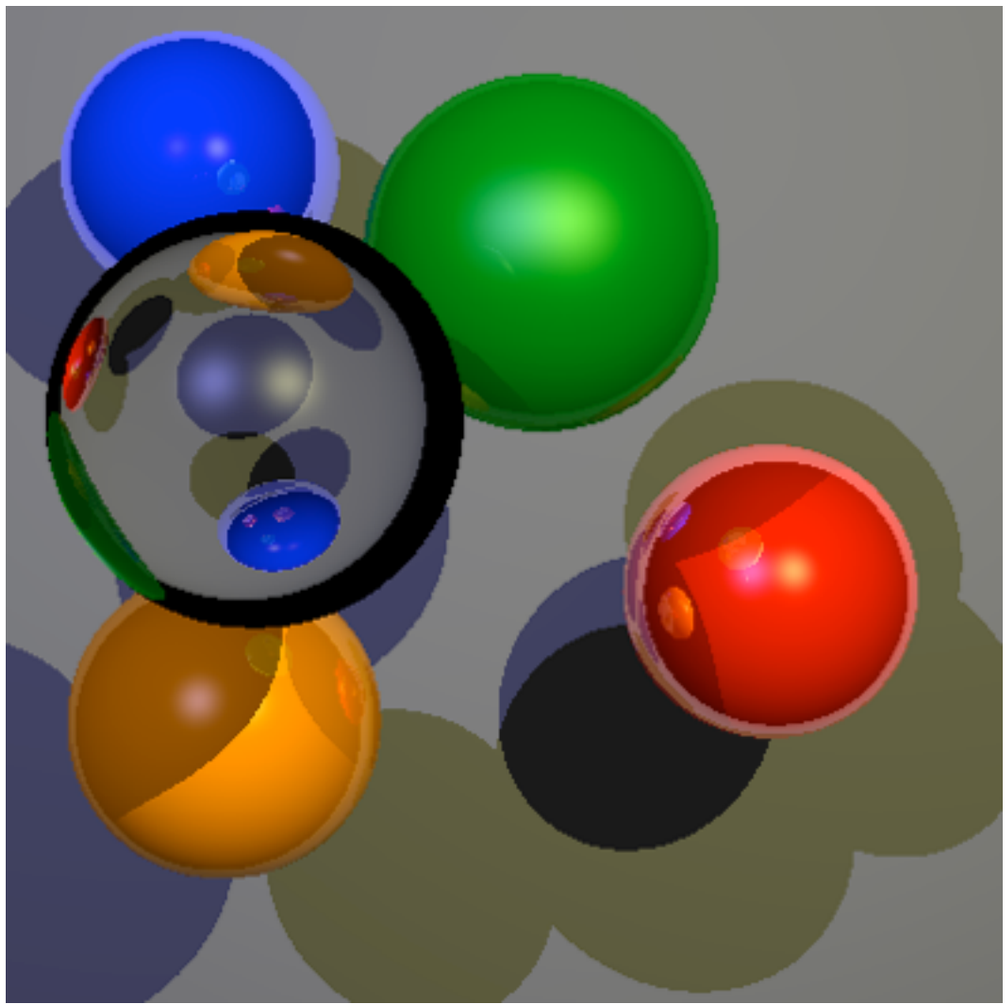
Raytracer: optical laws

- Implement shadows
- More than one light source
- Reflection
- Refraction

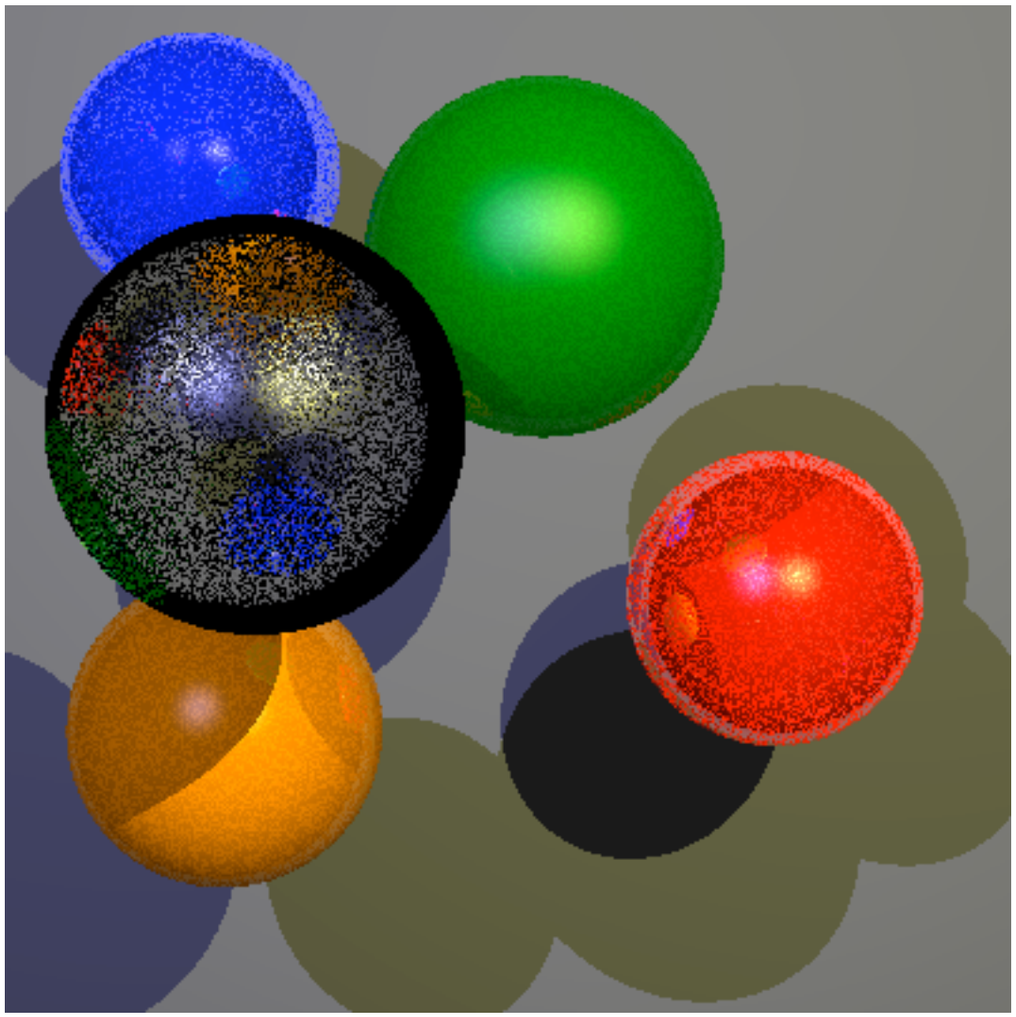


Raytracer: optical laws

- Implement shadows
- More than one light source
- Reflection
- Refraction



Possible pitfalls



Consider numerical stability

Refraction

- See your book (page 600)
- `eta` in input file is the index of refraction for the refracting material (η_r), use 1.0 for the other
- What happens if the ray is inside the object?