COMP557 Assignment4: Ray Tracing

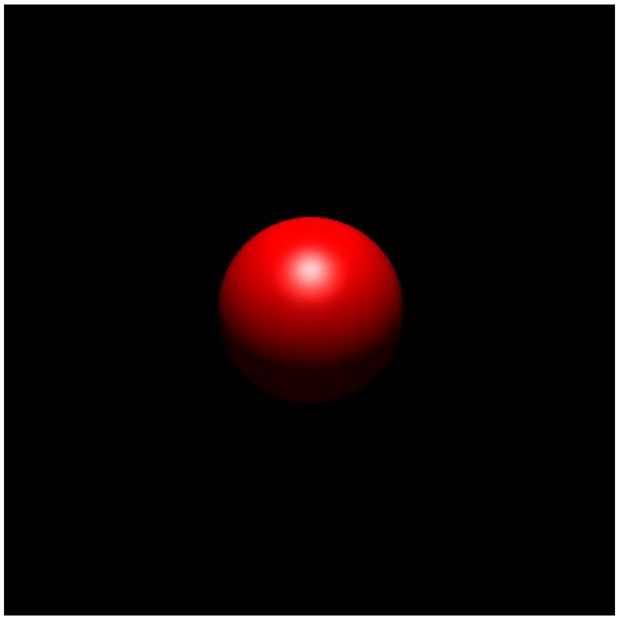
Agnes Liu 260713093

I have modified the .xml files to test for newly implemented features, but all images shown are reproducible by my code if using the original .xml files. In the image-xml correspondence description, if original appeared, that is because I modified them without changing the names.

Please see the images below as well in the folder as the objective outputs.

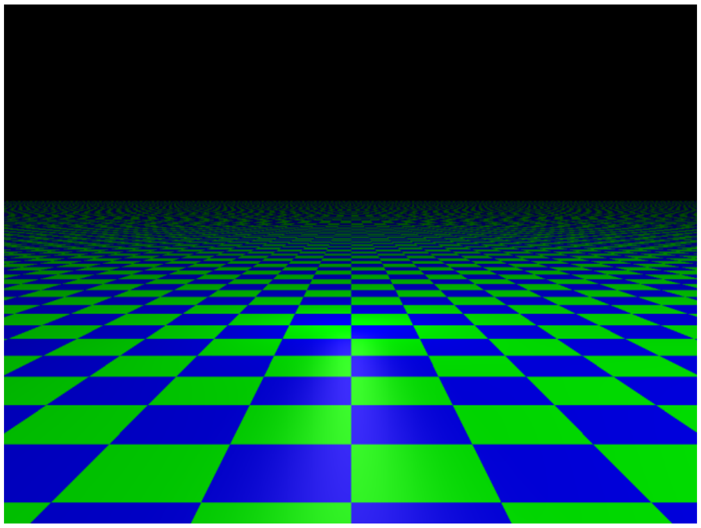
Sphere (**Obj1-3**: generating ray, sphere intersection, lighting and shading)

Produced by original Sphere.xml



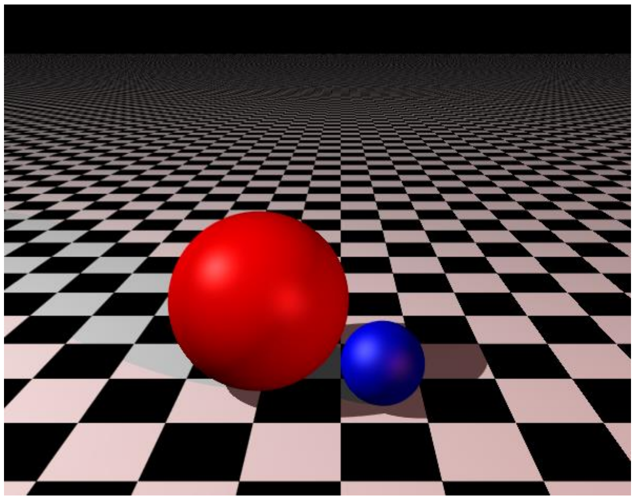
Plane & Plane2 (**Obj4**: Plane Intersection)

Produced by Plane.xml

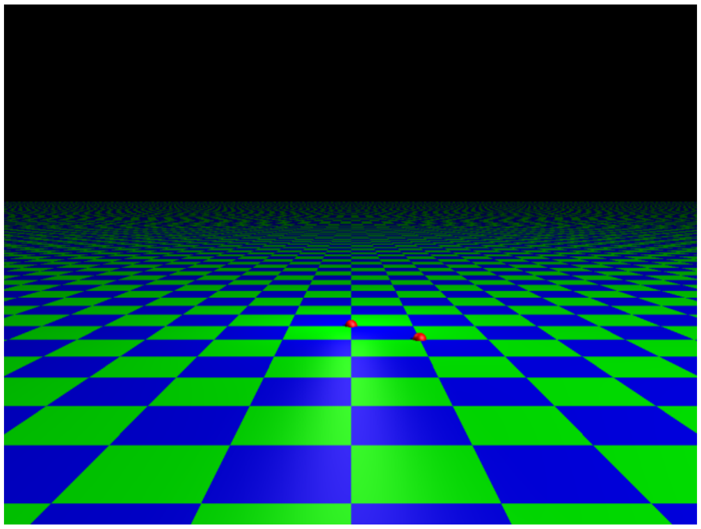


Original TwoSpheresPlane (**Obj 5**: shadows)

Produced by original TwoSpheresPlane.xml

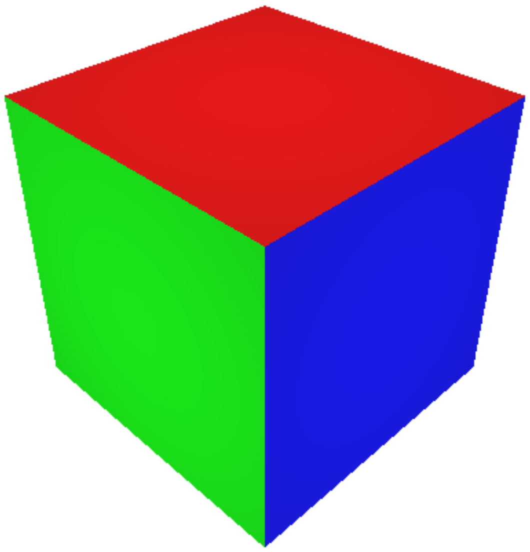


Produced by Plane2.xml



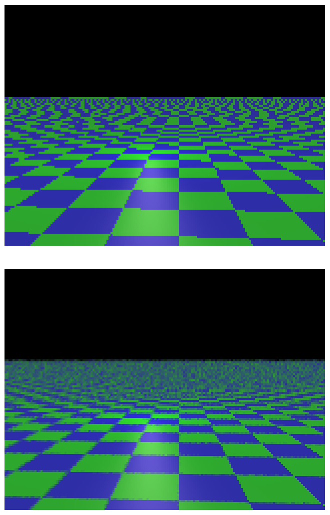
Obj6: Box

Produced by BoxRGBLights.xml



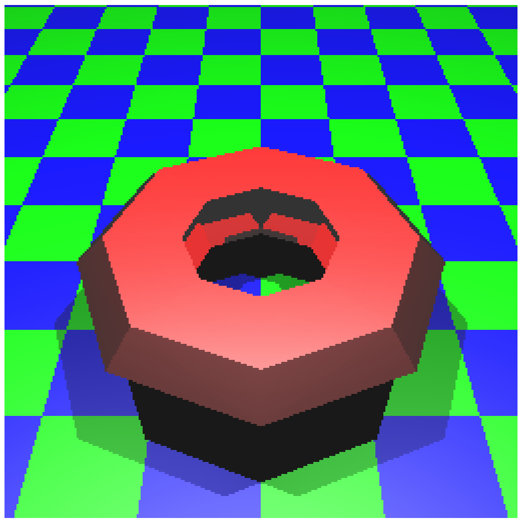
AACheckerPlane (**Obj8**. AA & super-sampling. Top: jitter==false; bottom: jitter==true)

Produced by AACheckerPlane.xml



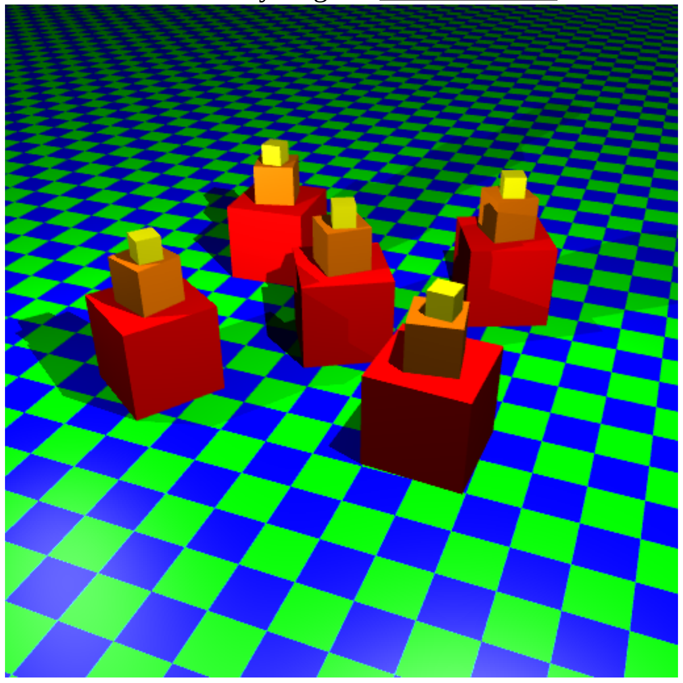
TorusMesh (**Obj9**. Triangle Meshes)

Produced by TorusMesh.xml



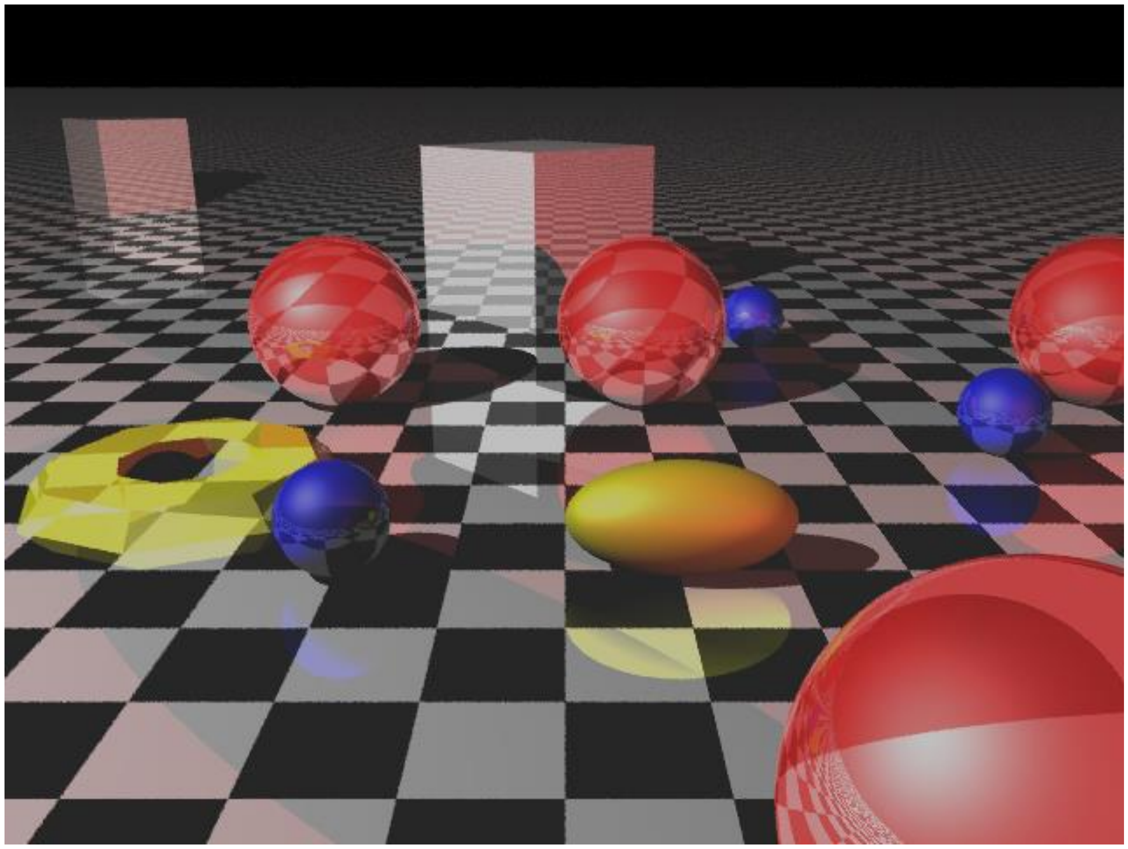
Obj7: Hierarchy and box stack

Produced by original BoxStacks.xml



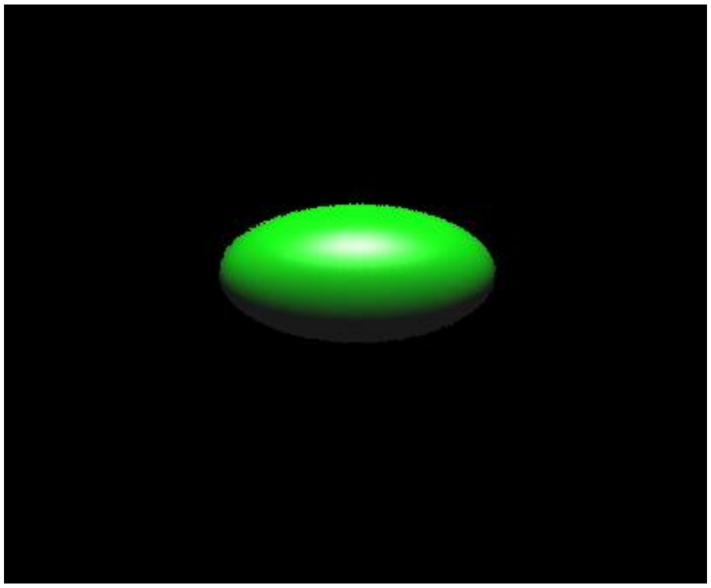
Obj10. My own scene (with all features previously & in 11)

Produced by NewFile.xml

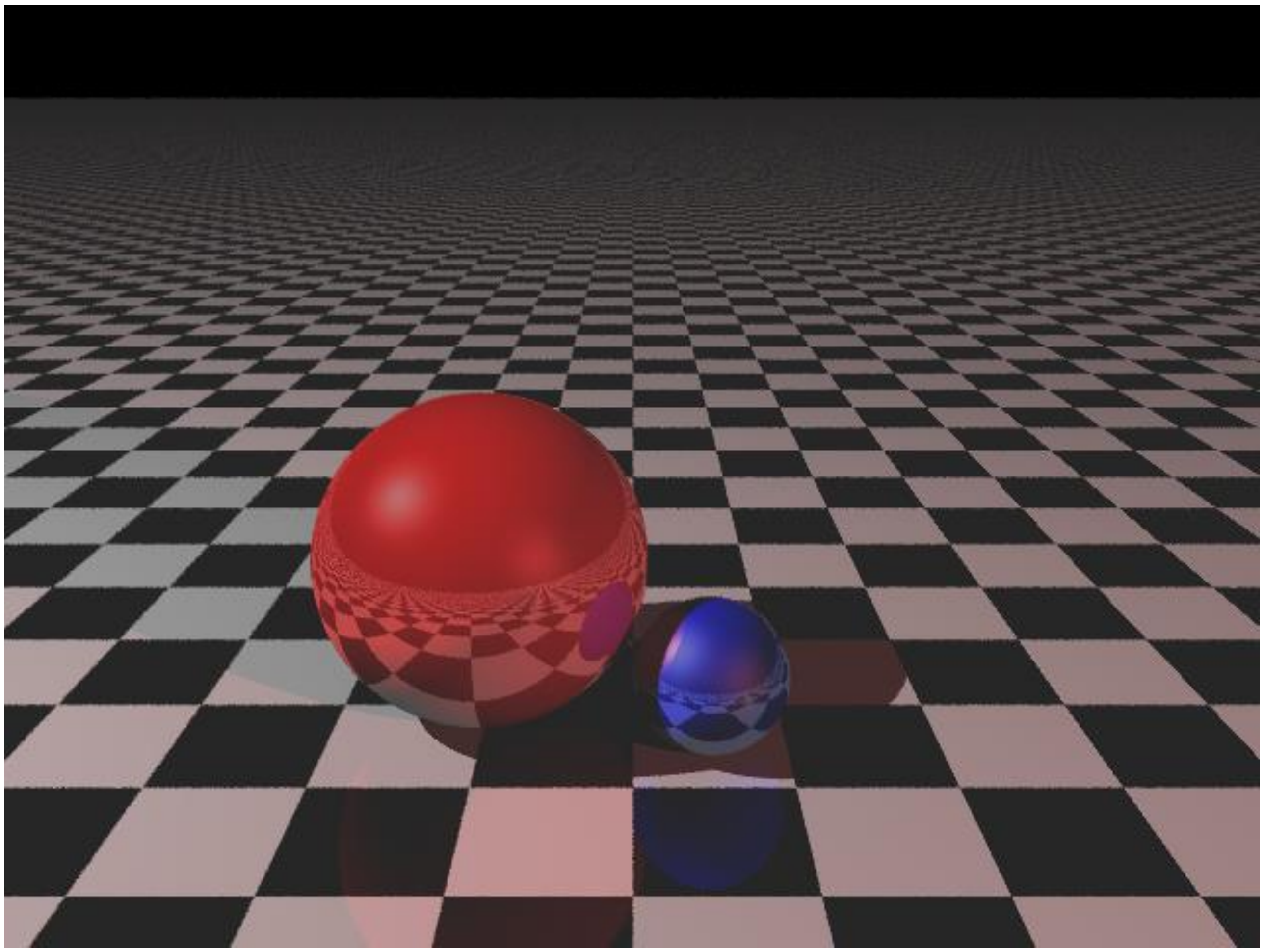


Obj11. Others

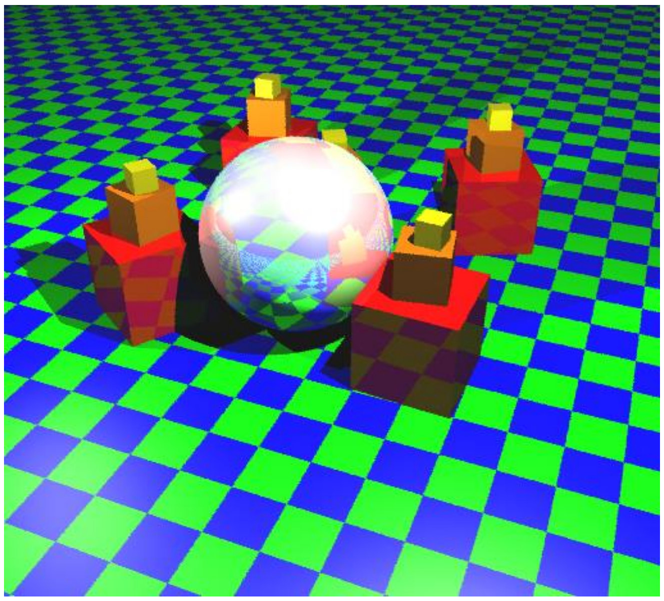
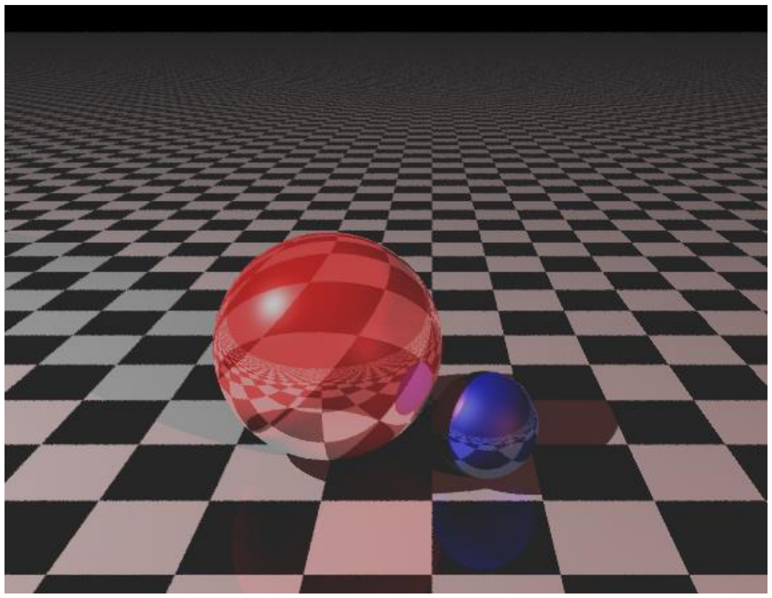
- **quadric intersection** (reference lecture slides “Raytracing” pg.44)
  produced by ellipsoid.xml



- **mirror reflection**
  produced by TwoSpheresPlane.xml without n and nt specified for red material

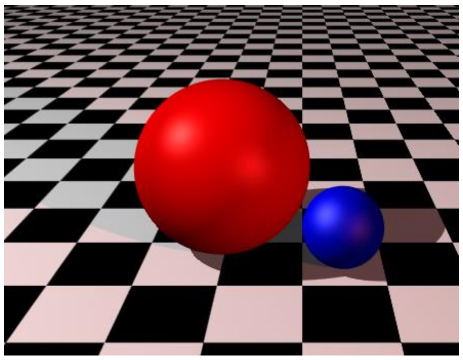
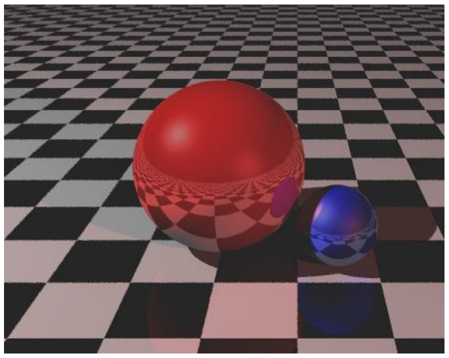
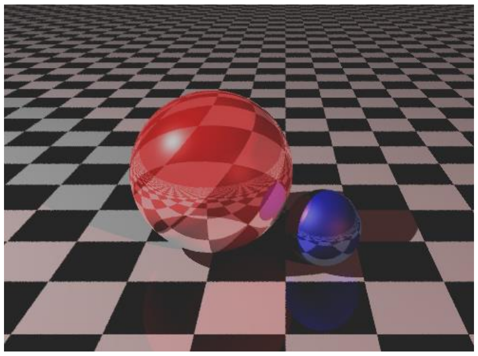


- **Refraction**
  Images showing both reflection and refraction.
  Produced by (submitted) BoxStacks.xml and TwoSpheresPlane.xml.

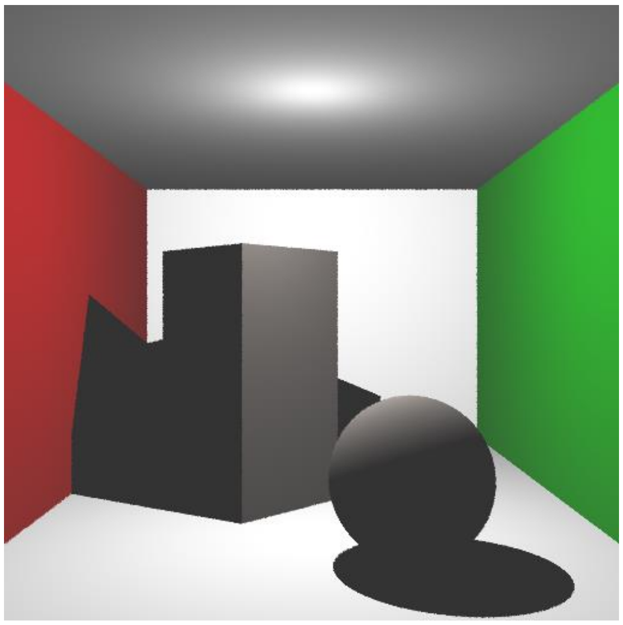
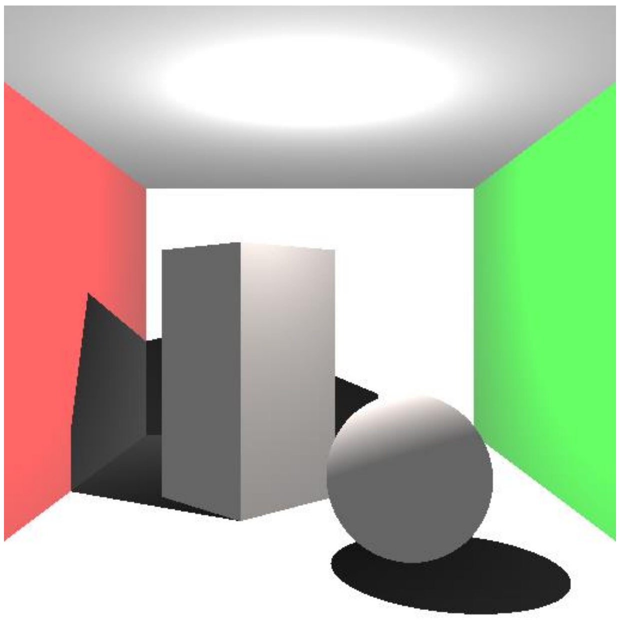



Some images for comparison:

from left to right: the original, with reflection only, with both reflection and refraction;

- Implemented **area light source** (by specifying light.type=”area”)
  Left: original Cornell.xml; right: submitted Cornell.xml with light.type = “area”

A majority of the implementation ideas came from slides and FCG textbook.

Quadric: same logic as sphere (finding the root to the quadratic equation for solving direction magnitude to identify the intersection point), except for some calculations changed from scalar parameters to vectors and matrices.

For reflection, refraction, and area lights: mainly achieved by generating new rays with different functionalities.

Reflection: generate reflection ray from incident point and extract color information on hit points of other objects. Negation for calculating the direction dropped as it was required to negate twice. Specified mirror surface by adding mirror=”true” in xml materials.

Refraction: generate new refraction ray from incident point and extract color information on hit points of other objects. Specified refraction index by n=speed of travel in from-object and nt=speed of travel in to-object.

Epsilon was added to the incident points along its direction to avoide scattering caused by hitting the object of interest itself.

Area light: simulate a bunch of other light sources on the same plane as our Light object, with all other attributes the same except light.from attribute. Then randomize the colors calculated on this pixel from these lights and average them to produce a blending effect. Specified by type=”area” for light node in xml.

References:

Peter Shirley and Steve Marschner. 2015. Fundamentals of Computer Graphics (3rd. ed.). A. K. Peters, Ltd., USA.

COMP557 course slides: Ray Tracing, Ray Tracing2.

Quadric formula: https://people.cs.clemson.edu/~dhouse/courses/405/notes/quadrics.pdf