Program 2: Ray Tracing 1.2
Updated: February 5, 2024

CS 5465: Computer Graphics
R. M. Parry

Read Chapter 4: "Ray Tracing" and Chapter 5: "Surface Shading" in the textbook before attempting.

This assignment is to be completed individually and the work you submit must be your own. You may edit any of the classes in the framework code and you may create new classes to complete the requirements. However, you do not need and should not use any other libraries.

In this programming assignment you will continue to improve your ray tracer. Again, scenes will be defined in XML files and the parser is provided. Using the details provided in the XML file, your ray tracer will add the following:

- Ellipsoids
- Lambertian and Phong materials
- Multiple point light sources
- Shifted perspective cameras (where the view direction is not parallel to the projection normal)

1 XML Scene Files

For example, the XML file might look like this:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--  Test scene with a single sphere.
 -->
<scene>
  <camera>
    <viewPoint>5 4 3</viewPoint>
    <viewDir>-5 -4 -3</viewDir>
    <projNormal>5 4 3</projNormal>
    <viewUp>0 1 0</viewUp>
    <projDistance>5</projDistance>
    <viewWidth>2.5</viewWidth>
    <viewHeight>2.5</viewHeight>
  </camera>
  <image>
    300 300
  </image>
  <material name="blue" type="Lambertian">
    <color>.2 .3 .8</color>
  </material>
  <surface type="Sphere">
    <material ref="blue" />
    <center>0 0 0</center>
    <radius>1</radius>
  </surface>
  <light type="PointLight">
```

```
    <position>5 4 3</position>
    <intensity>35 35 35</intensity>
  </light>
</scene>
```

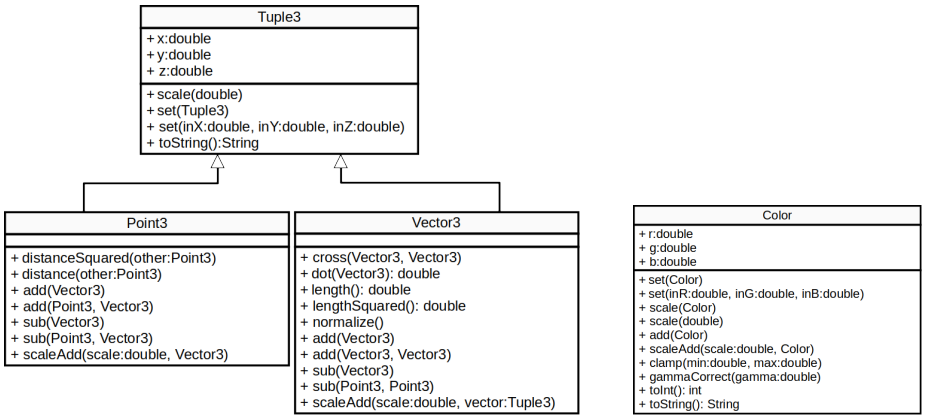
2 Framework Classes

The framework code parses XML scene files and creates the `Scene` object which contains everything you need to render the scene to produce the output image. The `RayTracer.renderImage` method takes the scene as the input argument and fills in the output image with the appropriate colors.

The framework code is the same as the first programming assignment, available here:
`http://cs.appstate.edu/rmp/cs5465/ray1.2.zip`

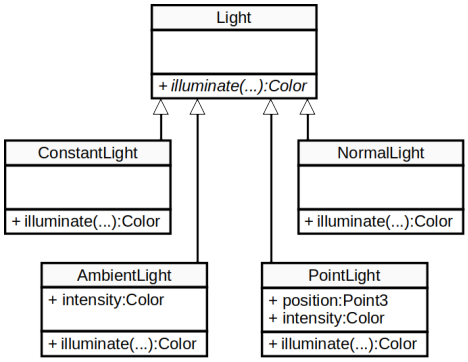
2.1 Three-dimensional Data

In order to render the scenes you will be using `Point3`, `Vector3`, and `Color` classes. `Point3` objects represent points in three-dimensional space, such as vertices in a triangle, the center of a sphere, a point on a plane, etc. `Vector3` objects represent vectors in three-dimensional space, such as surface normal vectors, the displacement between two points, a basis vector, etc.



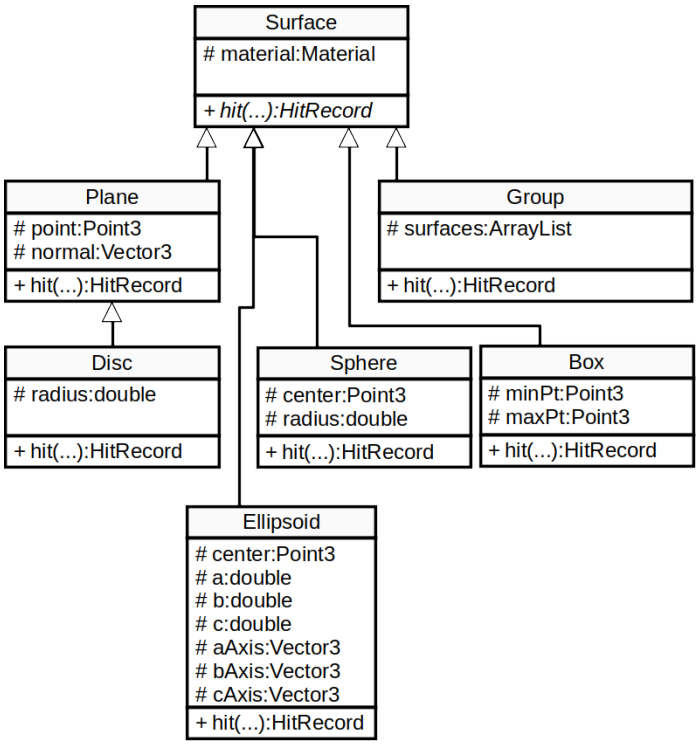
2.2 Lights

Lights in the scene are represented by concrete subclasses of the `Light` class. The `AmbientLight` has been provided. You will implement the `PointLight`.



2.3 Surfaces

Surfaces in the scene are represented by concrete subclasses of the `Surface` class. The `Sphere` and axis-aligned `Box` are provided. You will implement the `Ellipsoid` class.



The standard form for the implicit equation for the surface of an ellipsoid is the following:

$$f(\mathbf{q}) = \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 = 0,$$

where $x = x_q$, $y = y_q$, and $z = z_q$.

This assumes that the ellipsoid is centered at the origin $(0, 0, 0)$ with a-axis: $\mathbf{a} = (1, 0, 0)$, b-axis: $\mathbf{b} = (0, 1, 0)$, and c-axis: $\mathbf{c} = (0, 0, 1)$. To solve for the intersection point, using a ray: $\mathbf{q} = \mathbf{e} + t\mathbf{d}$, you would substitute:

$x = x_e + tx_d$

$y = y_e + ty_d$

$z = z_e + tz_d$

and solve for t like we did with spheres.

For example, when $a = b = c = 1$, the result is a sphere with unit radius at the origin. Implementing the ellipse to behave like a sphere when it should gets partial credit.

However, when a, b, and c are not all equal, the resulting surface is not a sphere. You earn additional credit for an axis-aligned ellipsoid centered at the origin.

If the ellipsoid has a center that is not at the origin, $\mathbf{center} = (x_{\text{center}}, y_{\text{center}}, z_{\text{center}})$, the query point for the implicit equation has a new interpretation:

$x = x_e + tx_d - x_{\text{center}}$

$y = y_e + ty_d - y_{\text{center}}$

$z = z_e + tz_d - z_{\text{center}}$

Implementing ellipsoids that can shift position earns additional credit.

Finally, an ellipsoid need not be axis-aligned, *i.e.*, the a-, b-, and c-axis need not equal the x-, y-, and z-axis. In this case the query point gets a new interpretation:

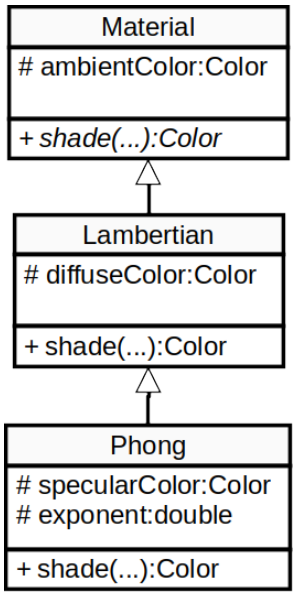
$x = (\mathbf{q} - \mathbf{center}) \cdot \mathbf{a}$
$y = (\mathbf{q} - \mathbf{center}) \cdot \mathbf{b}$
$z = (\mathbf{q} - \mathbf{center}) \cdot \mathbf{c}$

That is, the point $\mathbf{q}$'s direction away from the **center** of the ellipsoid projected onto the **a**, **b**, and **c** axes. You may assume that these vectors form an orthonormal basis.

Implementing ellipsoids with variable center points, axes lengths and basis vectors, received full credit.

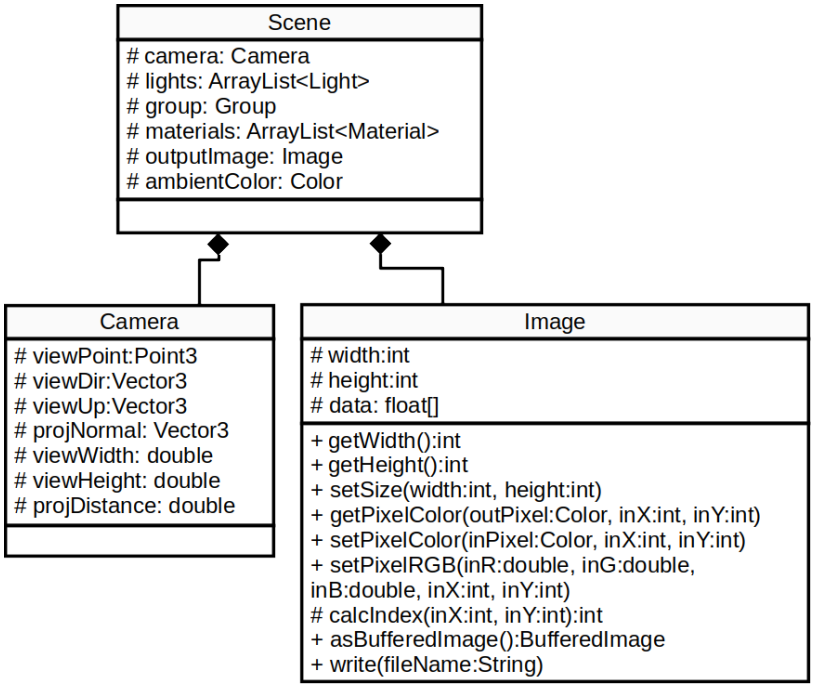
2.4 Materials

Materials represent how light bounces off surfaces. Materials in the scene are represented by concrete subclasses of the `Material` class. You will be implementing the `Lambertian` and `Phong` materials.



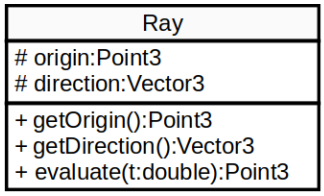
2.5 Scene

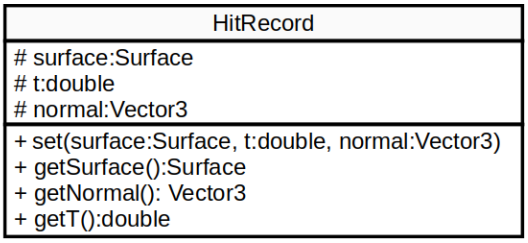
The scene contains in a `Camera`, a list of `Lights`, a `Group` of surfaces, a list of `Materials`, an output `Image`, and a `Color` for the ambient light source:

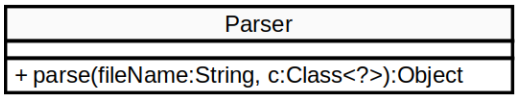


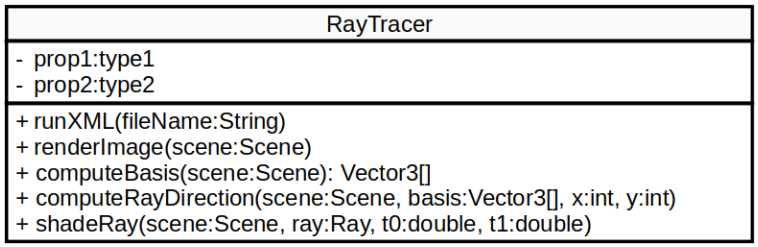
2.6 Miscellaneous Classes

It is convenient to represent a ray as a `Ray` object and the result of a ray-surface intersection as a `HitRecord`. The `Parser` parses XML and constructs the objects in the scene. The `RayTracer` renders the scene.









3 Illumination

Lights illuminate the scene. The `Light` class handles the illumination of an intersection point on a surface, and uses the `Material` to determine how much light is reflected, returning the `Color` "seen" by the `Ray`.

The 3^{rd} and 4^{th} editions of the textbook use Equation 4.4:

$$L = k_a I_a + \sum_{i=1}^{N} [k_d I_i \max(0, \mathbf{n} \cdot \mathbf{l}_i) + k_s I_i \max(0, \mathbf{n} \cdot \mathbf{h}_i)^p] \tag{1}$$

We will implement an `AmbientLight` that takes care of the first term in the equation above. Each `PointLight` takes care of one term in the summation. All `Materials` have an ambient color. Lambertian materials have a diffuse color. Phong materials have Lambertian properties and a specular color and exponent.

3.1 Ambient Light

An ambient light has no position and adds a bit of color to every surface in the scene. The amount of light that hits the surface is the `intensity` of the light. The reflected light depends on this intensity and the material's ambient color.

3.2 Point Light

A point light has a position and intensity. The amount of light that hits a surface depends light's intensity and distance to the point on the surface. The material determines what fraction of this light is reflected.

3.3 Shadows

Point light sources cannot illuminate a surface if there is another surface in between the two.

4 Materials

The material determines how much light is reflected toward the incident ray.

4.1 Lambertian

A lambertian material reflects light equally in all directions.

4.2 Phong Material

A Phong material reflects additional light in the direction of mirror reflection from the light source.

5 Shifted Perspective

Shifted perspective cameras have a projection normal that does not point opposite to the view direction. For these cameras, the center of the view plane is determined by following the view direction from the view point. Then, the basis is constructed so that **w** points in the direction of the projection normal. If the view direction happens to point opposite of the projection normal, the same computation produces a regular perspective camera.

6 Java Development

To work on this program you can use your favorite Java IDE such as Eclipse or VS Code. We will be using **Java 8** with **JUnit4**. Later versions of Java will not work for some of the assignments. The working directory for your project should be the `ray1` folder. The source location should be the `ray1/src` folder.

You can install the JDK for Java 8 here:
`https://adoptium.net/temurin/releases/?version=8`

On linux, you may install OpenJDK version:
`https://openjdk.org/install/`

You can get JUnit4 here:
`https://github.com/junit-team/junit4/wiki/Download-and-Install`

6.1 Running the Program

To run the program on a particular XML file, you can run `ray.RayTracer` as a Java Program with one or more XML files as arguments. Or, you can provide a path to a folder containing XML files as the argument. The resulting image(s) will be created in the same directory as the XML file with a ".png" extension.

For example, when I run the program in VS Code I get:

```
mitch@mitch-XPS-15-9575:~/Documents/cs5465/vscode/ray1$  cd
↪  /home/mitch/Documents/cs5465/vscode/ray1 ; /usr/bin/env
↪  /usr/lib/jvm/java-8-openjdk-amd64/bin/java -cp
↪  /tmp/cp_zh8lfqcq01mogvle85xb5w7n.jar ray.RayTracer
↪  scenes/diffuse/one-sphere-diffuse.xml
scenes/diffuse/one-sphere-diffuse.xml
Done.  Total rendering time: 0.092 seconds
Writing scenes/diffuse/one-sphere-diffuse.xml.png
```

6.2 Running the Tests

Use the standard way to run JUnit4 tests in your IDE to run the tests in `tests.Ray1Test2`.

For example, when I run the tests in VS Code I get:

`Debug Console`:

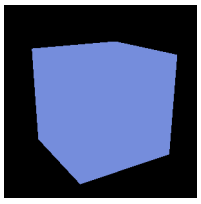
```
scenes/shifted-perspective/wire-box-sper2.xml
Done.  Total rendering time: 0.311 seconds
```

```
scenes/shifted-perspective/wire-box-sper2.xml : rmse = 0.0
...
scenes/shadows/one-box-specular-shadow.xml
Done.  Total rendering time: 0.017 seconds
scenes/shadows/one-box-specular-shadow.xml : rmse = 0.0
```

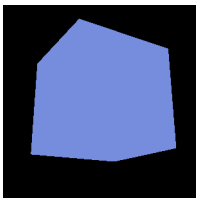
Again, the resulting image(s) will be created in the same directory with a ".png" extension.

7 Examples

7.1 Ambient Light



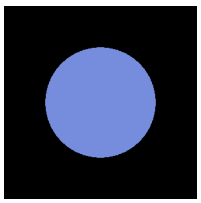
ambient/box-ambient/one-box-ambient.xml



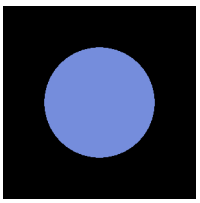
ambient/box-ambient/one-box-ambient-reverse.xml

ambient/box-ambient/two-boxes-ambient.xml

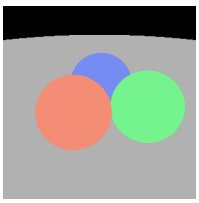
ambient/box-ambient/two-boxes-ambient-reverse.xml



ambient/sphere-ambient/one-sphere-ambient.xml



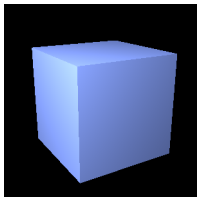
ambient/sphere-ambient/one-sphere-ambient-reverse.xml



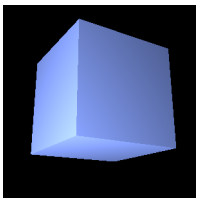
ambient/sphere-ambient/four-spheres-ambient.xml

ambient/sphere-ambient/four-spheres-ambient-reverse.xml

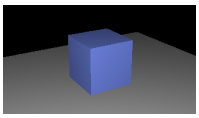
7.2 Diffuse Materials



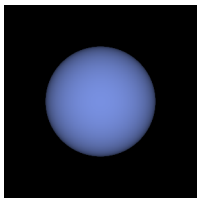
diffuse/one-box-diffuse.xml



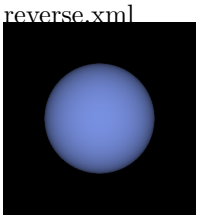
diffuse/one-box-diffuse-reverse.xml



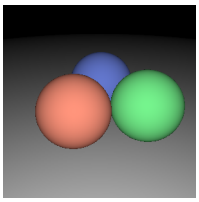
diffuse/two-boxes-diffuse.xml



diffuse/one-sphere-diffuse.xml

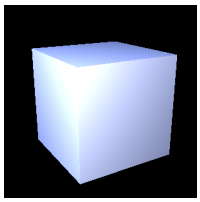


diffuse/one-sphere-diffuse-reverse.xml

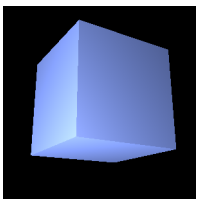


diffuse/four-spheres-diffuse.xml

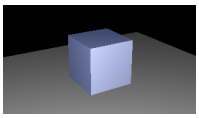
7.3 Specular Materials



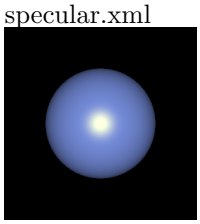
specular/one-box-specular.xml



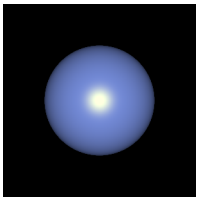
specular/one-box-specular-reverse.xml



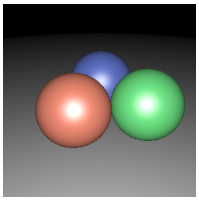
specular/two-boxes-specular.xml



specular/one-sphere-specular.xml



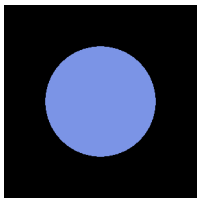
specular/one-sphere-specular-reverse.xml



specular/four-spheres-specular.xml

7.4 Ellipsoids

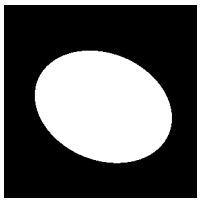
7.5 Spheres



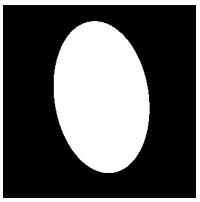
ellipsoid/ellipsoid-sphere-constant/one-ellipsoid-sphere-constant.xml

ellipsoid/ellipsoid-sphere-normal/one-ellipsoid-sphere-normal.xml

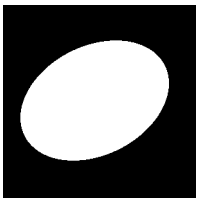
7.6 Stretched



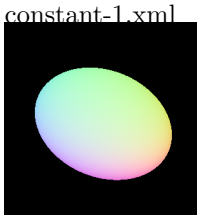
ellipsoid/ellipsoid-stretched-constant/one-ellipsoid-stretched-constant-1.xml



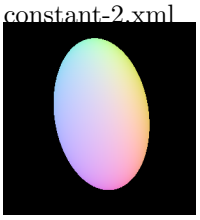
ellipsoid/ellipsoid-stretched-constant/one-ellipsoid-stretched-constant-2.xml



ellipsoid/ellipsoid-stretched-constant/one-ellipsoid-stretched-constant-3.xml



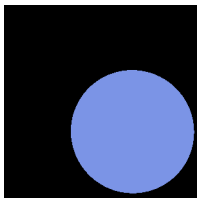
ellipsoid/ellipsoid-stretched-normal/one-ellipsoid-stretched-normal-1.xml



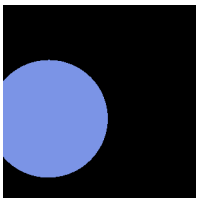
ellipsoid/ellipsoid-stretched-normal/one-ellipsoid-stretched-normal-2.xml

ellipsoid/ellipsoid-stretched-normal/one-ellipsoid-stretched-normal-3.xml

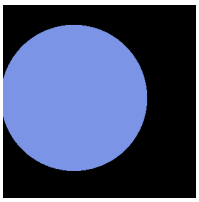
7.7 Shifted



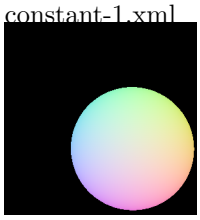
ellipsoid/ellipsoid-shifted-constant/one-ellipsoid-shifted-constant-1.xml



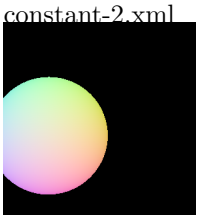
ellipsoid/ellipsoid-shifted-constant/one-ellipsoid-shifted-constant-2.xml



ellipsoid/ellipsoid-shifted-constant/one-ellipsoid-shifted-constant-3.xml



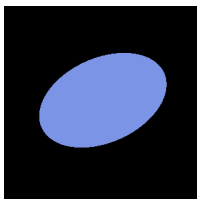
ellipsoid/ellipsoid-shifted-normal/one-ellipsoid-shifted-normal-1.xml



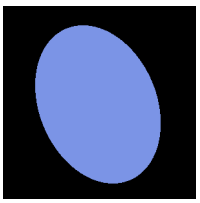
ellipsoid/ellipsoid-shifted-normal/one-ellipsoid-shifted-normal-2.xml

ellipsoid/ellipsoid-shifted-normal/one-ellipsoid-shifted-normal-3.xml

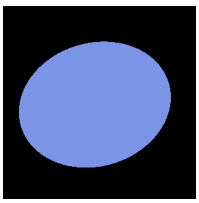
7.8 Rotated



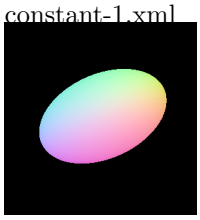
ellipsoid/ellipsoid-rotated-constant/one-ellipsoid-rotated-constant-1.xml



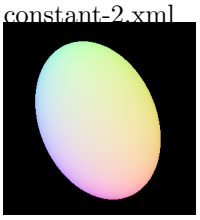
ellipsoid/ellipsoid-rotated-constant/one-ellipsoid-rotated-constant-2.xml



ellipsoid/ellipsoid-rotated-constant/one-ellipsoid-rotated-constant-3.xml



ellipsoid/ellipsoid-rotated-normal/one-ellipsoid-rotated-normal-1.xml



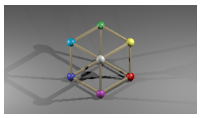
ellipsoid/ellipsoid-rotated-normal/one-ellipsoid-rotated-normal-2.xml

ellipsoid/ellipsoid-rotated-normal/one-ellipsoid-rotated-normal-3.xml

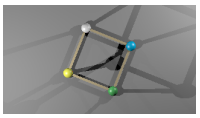
7.9 Easter Eggs

ellipsoid/easter-eggs/easter-eggs.xml

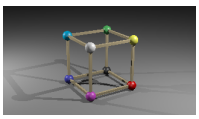
7.10 Wire Box



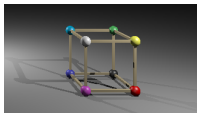
wire-box/wire-box-axon.xml



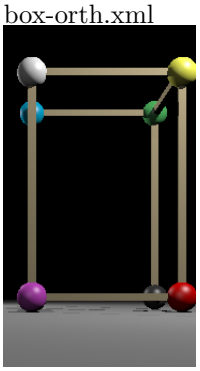
wire-box/wire-box-orth.xml



wire-box/wire-box-per.xml



shifted-perspective/wire-box-sper.xml



shifted-perspective/wire-box-sper2.xml

8 Web-CAT Submission

Your assignment will be graded by Web-CAT. ZIP your `src` directory and upload it here: `http://webcatvm.cs.appstate.edu:8080/Web-CAT`