CSci 490, Spring 2005, Assignment 6

This assignment, worth 40 points, is due at 3pm, Friday, April 8. *Submit it by attaching your modified files to an e-mail to* `cburch@cburch.com`.

You can download the base code from the course Web page, which implements a ray tracer in Java. It includes the following classes.

AbstractIntersection	Partial implementation of Intersection interface.
Canvas	Maintains a canvas whose contents are unmanaged, but which has an underlying array that can be used to modify the contents directly.
Color	Represents a color with red, green, and blue components.
ColorVector	Represents a factor by which to multiply a color's components, including different factors for each component.
Intersection	Represents information about where a ray reaches a surface.
Material	Represents information about the specular properties of a surface, including how reflective or refractive the surface is.
ModelViewer	Manages a window containing a scene.
Point	Represents a point in three-dimensional coordinates.
Polygon	Represents a three-dimensional flat surface. This surface should be fully opaque.
Ray	Represents a ray in three-dimensional coordinates, including an originating point and a direction vector.
Renderer	Object for drawing a view of the scene onto pixels.
Scene	Represents a scene, including both the surfaces and the light sources in the scene.
Sphere	Represents a solid sphere.
Surface	Represents an abstract surface in the scene.
Vector	Represents a vector in three-dimensional space.

Your job is to modify the ray tracer so that it implements reflection and refraction. The only class you will need to modify will be the Renderer class.

One thing worth noting: A reflected ray's direction should be scaled to have a very short length before recursing on the ray. This is because the Scene class will only return surfaces that are at least the direction's length away from the ray's originating point. (This is important so that an object doesn't end up reflecting itself at the same point.) The same concept applies to refraction.

For refraction, you will need to know whether you are entering or exiting an object in order to determine the refracted ray's direction. Note that the surface normal returned by an Intersection object always points toward the hit object's exterior. You can compute the dot product of the view vector and the surface normal; if it is negative, the view ray is going into the solid (the ray hits the surface's exterior), and if it is positive, the view vector is going out of the solid.

Also included in the handout code are three classes rendering scenes useful for testing purposes.

- The scene for the Test1 class (`java Test1`) consists of one silver sphere and two stacked gold spheres. This is for testing reflection.
- The scene for the Test2 class (`java Test2`) consists of one gold sphere and one glass sphere side by side. This is for testing refraction.
- The scene for the Test3 class (`java Test3`) consists of three spheres stacked like a snowman, with a gold head, a glass body, and a silver bottom, all on a flat white floor and with a flat white background. It includes two light sources. The most notable reflection will be off the silver bottom; the glass body will of course refract light.