Assignment Four

Neil Edelman

2010-11-29

1) The **Scene.java** includes a method **private static Ray generate(final Camera cam, final int x, final int y)** that generates the rays. In $-d\hat{\vec{w}}$, d is 1. It would be this method that I would modify to set supersampling.

2) **Sphere.java** includes sphere intersections. The **Matrix4d transform** transforms the ray so I don't need to have a complicated expression for intersecting. Looking back, I would have made **Intersectable** an abstract class instead of an interface to get rid of the massive re-use of code.

3) **IntersectionResult.java** gives a framework for keeping track of intersections; **Vector3d n** is the normal at the intersection, **double rayParameter** is the parametre on the ray, **Point3d intersection** is the intersection point, **Intersectable shape** is the shape, and **Point3d eye** is for doing Phong lighting, a copy of the **eyePosition** because I didn't want to include another parametre on lighting. In **Light.java**, only thing public is **public static Light magic(Node dataNode)**, which sorts based on XML into categories auto-magically. The subclasses each implement **public abstract Color3f manifest(final Intersectable root, final IntersectionResult result)** in their own way. I've done ambient (ambient lights are a light type, not a parametre of the scene,) directional, and point lights. In **Scene.java** under **private Color3f colour** in the **for(Map.Entry<String,Light> lightHash : lightmap.entrySet())**, each light is intersected with the whole scene; there would be an increase of performance in static scenes (like ours) if the light, instead of **nodeMap.get("root")**, contained a **light.root** of all the scene visible to the light (especially in a high-occlusion scene.)

4) In the **public boolean intersect(final Ray ray, IntersectionResult result)** defined in **Intersectable.java** I set a flag to say that it's a shadow ray, that is, **result** is null (we don't care about result, just whether it hit something.) In **Ray.java**, I have a new variable, **private double interval = Double.POSITIVE_INFINITY**. For point lights, I clamp the interval at the distance from the object to the light so that occluders on the other side of the light have no effect. Each light type is responsible for implementing their own shadows. I have a boolean shadows; if I had more time, I would make them **Color4f** for transparent objects.

5) I added a new file, **Cube.java**. This was the most difficult part of the assignment.

6) I modified the **PolygonSoup.java** to load object files. Only loads triangles. Very simple bounding sphere. If I had time, I would divide it into convex meshes, each with their own bounding sphere. Then I would have **Edge** structure that connects two edges. I would scrap going thought the whole shape and use the values of the baricentric coordinates to direct the current shape towards an intersection. I think it would work much faster; $\mathcal{O}(l)$ instead of $\mathcal{O}(l^2)$ where l is the length of the model?

7) I made a new class in **Instance.java**. **Intersectable** should totally be an abstract class. It stores a **SceneNode** in **sn**. *Do something reasonable with the material definition of an instance (i.e., replace the material of the intersection, or modulate the colours).* the *only* thing that makes sense (without having materials stack on top of each other) is ignore the material; in the case of an interface, just don't store it and return null in **material()**.

8) Using Blender, I exported a dinosaur that was going to eat the cows, but the dinosaur wouldn't load.

The Bresenham line-drawing algorithm was used to do the progress bar. I'd hoped to do a transparent Voxel object for which the Bresenham algorithm was used for accumulating colour, but then I'd better start my lab.

I have a **Ground** object that is just a plane at some level. The ground object uses hard-coded normal mapping.

I used barycentric coordinates to do Phong shading on the mesh objects. It would be easy to do texture mapping, but I have no textures, and time is up.

screens/pandorica.png is my latest incarnation.