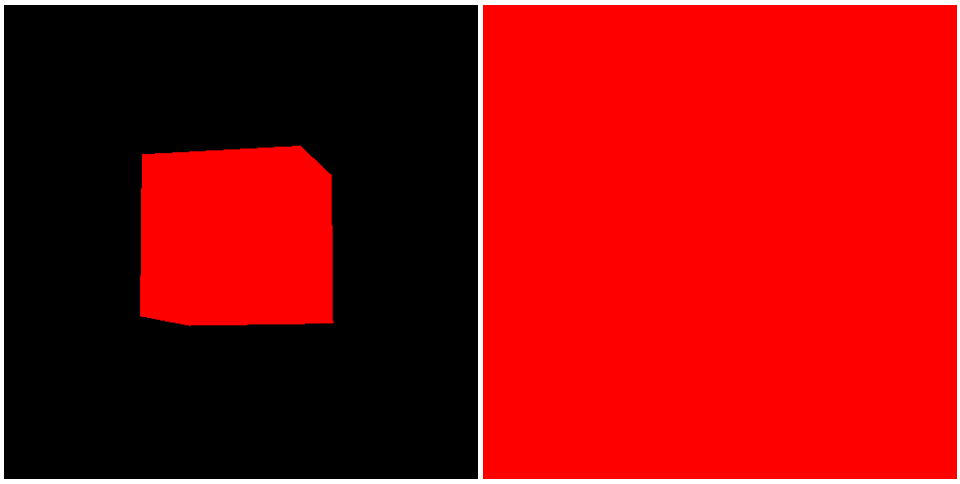
Program 3

Parallel Hardware Ray Tracing

Due: 11:59:59 PM, September 26, 2011

In this assignment you will implement boxes (in preparation for BVH traversal) and triangles, and render a simple scene loaded in TRaX memory. For more info on boxes and triangles, see the lecture 4 slides.

1. **Part 1** (40 points): Add boxes to your ray tracer. They do not need to support shading, but they do need to correctly implement ray-box intersection. We will be using boxes for bounding volume hierarchy traversal in the next assignment. To verify your box intersection is correct, generate the following images:



   Both images have a single box in the scene centered around the origin:

```
Box b(Vector(-1.f, -1.f, -1.f), Vector(1.f, 1.f, 1.f));
```

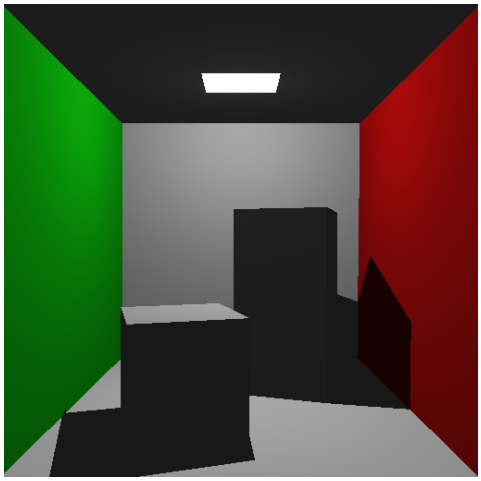
   The image on the left has the camera positioned at:

```
PinholeCamera camera(Vector(4.f, -15.f, -2.f), // eye
                     Vector(0.f, 0.f, 0.f), // lookat
                     Vector(0.f, 0.f, 1.f), // up
                     0.194f, (float)xres / (float)yres); // uLen and aspect ratio
```

   The image on the right has the camera positioned anywhere inside the box, looking in any direction (just make sure it's not looking in the same direction as the up-vector). These images don't have any shading, as they are just designed to verify that your intersection test is correct. For each pixel, if the camera ray hits the box, color it red, else color it black.

2. **Part 2: Triangles and TRaX scene memory** (40 points) Implement a triangle class in your ray tracer (but don't call it "Triangle", do avoid conflicts). Devise your own tests to first verify that your intersection is implemented correctly, similar to the box test above, then implement simple Lambertian shading from a single point light source for your triangles (just like you did for spheres). See lecture 9 for triangle normal computation.

   Once you are convinced you can render triangles (including Lambertian shading), render the Cornell scene loaded in to TRaX memory. Don't use the BVH, just loop through every triangle in memory, as seen in lecture 9. In the next assignment, we will traverse the BVH. The example Makefile automatically loads the Cornell scene for `run_rt`, so just write your ray tracer assuming that a scene is loaded in to memory (use start_triangles and num_triangles to find it).



To load the Cornell scene for `simhwrt`, use the following command:

```
./simhwrt --load-assembly <path to your assembly>
--model test_models/cornell/CornellBox.obj
--view-file views/cornell_obj.view --light-file lights/cornell.light
```

The ambient light color is [0.4f, 0.4f, 0.4f], `Kd` is 0.7f, `Ka` is 0.3f for every material.

Attach the output of rendering the Cornell scene in simhwrt to your handin webpage.

3. **Creative Image** (20 points): Generate a creative image of your choice with the ray tracer you have built so far using any type of geometry and shading. Your creative image should include triangles and boxes (if you decide to implement box shading).

4. **simhwrt output**: Link a file containing the output of rendering the Cornell scene in the TRaX simulator

5. **Code listing**: Link your source code to your web page (preferably .tar). All of the code that you use should be included. You will not be graded on the quality of your code or comments, only on the presence of your source. We will verify that the code you hand in will produce the image(s) turned in.

6. **Initial project idea**: Write a paragraph or two describing any ideas you have for a project you want to implement in TRaX. This can be anything from advanced ray tracing techniques to modifications to the TRaX architecture. If you don't have any ideas, that's fine, just indicate this. We will be talking in more detail about projects and potential project ideas shortly.

7. **Time required**: How many hours did it take you to complete this assignment?

8. **Diffuculty**: What was the hardest part of this assignment, why?

You will not be graded on these last two items. They will be used to help improve the class in future assignments and in future years.

By midnight on the due date, you should send e-mail to teach-cs6965@list.eng.utah.edu with a URL for your assignment web page.