Foundations of Computer Graphics

Assignment 7

Instructions:

- Assignments can be submitted in groups of at most three. The purpose of groups is to learn from each other, not to divide work. Each member should participate in solving the problems and have a complete understanding of the solutions submitted.
- Submit your assignments as a zip file (one per group) which includes the Common directory and a separate directory for each of the assignments so that we can run your code by just extracting the zip file and double-clicking the html files.

Problem 1 (10 points)**.**
Modify the ray tracing program so that most of the computation is done in the fragment shader. The idea is to draw a square with corners $(-1,-1)$ and $(1,1)$. Each fragment then corresponds to a pixel and the corresponding fragment shader can then figure out the color of the pixel by ray tracing. You need to move the entire scene set up and the relevant functions to the fragment shader. You may need to use C-style structs in the fragment shader since GLSL ES 2.0 does not support C++ style classes. Please have a look at `http://math.hws.edu/eck/cs424/notes2013/19_GLSL.html` to learn how to use these.

Further modify the code to add *anti-aliasing*. This can be done by doing recursive ray tracing by shooting multiple random rays through a pixel and taking the average of computed colors.

Your program should display an animated scene with at least two lights and two objects.

Bonus (10 points): Display a billiard ball by putting the texture `billiard_ball_texture.jpg` on a sphere. This ball should be spinning slowly along an axis passing through its center so that the texture is not static. The ball should look similar to the ball shown in `billiard_ball.jpg` (ignoring the reflection of the area lights in the image).

Problem 2 (10 points)**.**
Create a terrain by creating a large square grid on the xz-plane and giving a height (y-coordinate) to each grid point using a 2D noise function. Apply the texture `moss-diffuse.jpg` and the normal map `moss-normal.jpg` on the terrain. In order to avoid streching the images, you need repeat the texture and normal maps over the surface. For instance, the texture coordinates of a point (x, y, z) can be set to $(x - \lfloor x \rfloor, z - \lfloor z \rfloor)$. Set the shininess constant in Phong Lighting to a small value since we don't expect the terrain to be very reflective. The user should be able to move around the scene.

Problem 3 (10 points)**.**
In the previous assignment, you implemented a skybox. Add a teapot (`Models/More/teapot.obj`) to the scene and implement reflection mapping on the teapot. It should be possible to move the camera in the scene and when it is translated, the teapot should move relative to the camera but the skybox shouldn't since it represents scenery at a distance. This can be achieved by

making the sky box cube very large but that is very inefficient. An efficient way is to keep the cube small and always move it so that its center is at the camera location. To avoid the cube from obstructing the view, you can disable writing into the depth buffer when drawing the cube. Effectively then it appears that the cube is very large. To disable writing into the depth buffer, you can use `gl.depthMask(false)` and to enable it again use `gl.depthMask(true)`.