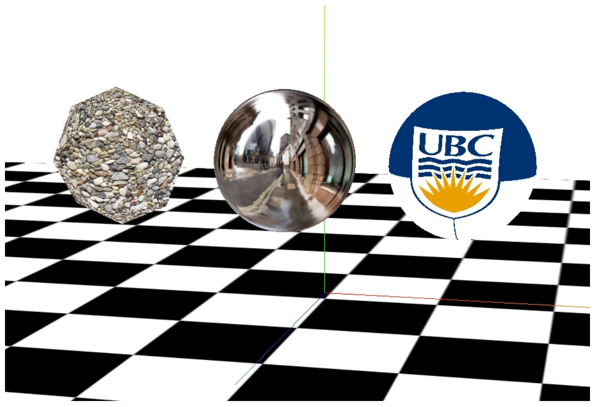
CPSC 314
Assignment 4: Introduction to Texture Mapping

Due 4PM, April 2, 2015

1 Introduction

The goal of this assignment is to get a first exposure to the huge area of texture mapping. You will start with a JavaScript template provided by the instructor. Built in Three.js materials, such as `MeshBasicMaterial`, allow you to quickly and easily apply textures. This assignment includes a simple scene with three objects. The main challenge is to understand how texture mapping is implemented in shaders, and to experiment with different uses of textures. The figure shows a sample result for Part 1 of the assignment.



2 Work to be done (100 pts)

Before begining your work, please take a look at the template, which contains hints in the comments marked with "TODO". Also take note of the images sub-directory. We have provided a number of sample images that you can use, but feel free to use your own. The textbook has detailed explanations of all texturing schemes you are asked to implement, in Chapter 15.

Part 1 : Exercises (70 points)

1. **20 pts** Basic Texturing of an Octahedron.

   In the most basic texturing scheme, the texture image is used to look up the final color of the object. UV texture coordinates are associated with each vertex by the modeler. The interpolated values of these UV coordinates are passed to the fragment shader, which simply sets that fragment color to that in the texture image. You may use the standard Three.js library ImageUtils to load the texture, but you should implement your own custom shaders to look up the fragment color in the texture map. You may use the gravel texture included in the images directory.

2. **20 pts** Environment Cube Mapping on a sphere

   As discussed in class and in the textbook, cube mapping simulates a perfectly reflective object that mirrors the environment around it. In this part, you will apply a basic static cube map (that is, one that reflects a static scene, but not other objects in the scene) to a sphere. Three.js includes functions in ImageUtils to help load a texture cube that you may use. You do not have to write new shaders. The images directory contains two directories: `cubemap_debug`, whose images have discontinuities at the edges, so that you can see more easily which images are reflected; and `cubemap_real`, whose images constitute a proper environment cube map.

3. **30 pts** Projected Textures on a sphere

   In this part you will project an image on the object, as if a slideshow projector was displaying a picture on it. Using Three.js you may pass the projector data to the vertex shader as uniform variables. Texture coordinates are computed in the vertex shader, and passed to the fragment shader. This has been discussed extensively in class. See Chapter 15 of the textbook for implementation details and be careful about your projection matrices.

   Hint: The textbook describes the projector as essentially a second camera in the scene. However, do not use the Three.js camera class to get your perspective and view matrices (Three.js does not maintain the matrices for cameras not used to render the scene properly). Its much easier to compute the view and projection matrices with Matrix4's `lookAt` and `makePerspective` functions. Beware: `lookAt` only computes the rotation part; you will have to add the translation part of the view matrix.

Part 2 : Creative License (30 pts)

For this part we want to see what you can do. Your ideas should use at least one new shader, and should be of a similar complexity to the previous tasks. You should apply the texture to a more complicated object, such as the Armadillo model from previous assignments. If you have any doubts, make sure to OK it with a prof or TA. Some possible suggestions might be:

- Create dynamic cube maps to reflect other scene objects. This is easy with Three.js

- Implement a normal mapping scheme to create more complex looking geometry.
- See http://threejs.org/examples/ and http://www.shadertoy.com/ for more inspiration.

Bonus marks may be given at the discretion of the marker for particularly noteworthy explorations.

2.1 Hand-in Instructions

You do not have to hand in any printed code. Create a README.txt file that includes your name, student number, and login ID, and any information you would like to pass on to the marker. Create a folder called “a4” under your “cs314” directory. Within this directory have two subdirectories named “part1,” and “part2”, and put all the source files, your makefile, and your README.txt file for each part in the respective folder. Do not use further sub-directories. The assignment should be handed in with the exact command:

```
handin cs314 a4
```