CS541 Project 3

Synopsis

Implement a reflection-map using a two-pass algorithm, the first pass generating a reflection map of the environment viewed from the viewpoint of a reflective object, and the second pass using the reflection-map to draw a reflective object reflecting the environment.

Instructions

This project comes with a framework which:

- Implements the rudiments of a two pass algorithm, including the creation of FBOs (Frame Buffer Objects) to act as render targets for pass 1 and texture maps for pass 2.
- Defines a scene consisting of a reflective object (teapot or sphere) and an environment of objects to reflect.
- Produces all the transformations necessary to draw the scene twice, once in pass 1 to generate the reflection-map and again in pass 2 to produce the on-screen image.
- Defines four nearly empty shaders for you to complete.
- Look for comments containing **//@@** in **scene.cpp** for places you need to provide some implementation.

The shaders for this project must perform the following actions:

- **Pass 1:** This pass draws the reflecting objects from the viewpoint of the reflecting object (which for this pass is at the origin).
  - **Vertex shader:** The unwrapping of the world to project onto several textures is far beyond the capabilities of a 4x4 transformation, but the vertex shader can compute this transformation. (See below.) This shader must also provide the quantities needed for the pixel shader to compute the object's Phong lighting.
  - **Pixel shader:** Calculates the Phong lighting at each pixel. The output of this shader goes to the textures associated with the FBOs.
- **Pass 2:** This pass draws all objects to the screen with the usual Phong lighting, with the exception of a single reflective object which is drawn specially.
  - **Vertex shader:** Prepares the quantities needed for Phong lighting
  - **Pixel shader:** A uniform boolean variable can differentiate between objects that need Phong lighting and the reflective object. For the reflective object, calculate the direction of the reflection (from the normal which you will have from the Phong lighting calculations) and use that direction to access the reflection-maps created in pass1. (See below for the equations.) The value resulting from the texture look-up can go directly as the output color (for a perfectly reflective object) or can be blended with the reflective objects color (to implement a partially reflective, partially colored object).

Grading Basis

Grading will be on a 100 point basis, with the following distribution:

- **Accuracy:** 70%, judgment based on looking at the results on the screen.
- **Efficiency:** 20%. The application must run in real-time.
- **Code:** 10%, judgment based on looking at the code for proper implementation and reasonable commenting practices.

Maps of the sphere

When creating a reflection map, you need to decide on a map from all 3D directions to points in one or more rectangular textures. I am explicitly excluding the two map types which map to a single texture. (They produce too much distortion to be usable in a general situation.) Here are the mapping equations and texture sizes for each of the three types discussed in class. An aspect not mentioned in class: When the graphics pipeline renders objects to the render target, the clipper is still running, so the Z-coordinates in each of the following are carefully manipulated to fall within the clippers range.

Notation: In each of the following, point R is the point being transformed. In the Pass1 vertex processor R = gl_ModelViewMatrix*gl_Vertex; In the Pass2 pixel processor, R should be the reflection vector R = 2.0*dot(N,E)*N-E;

- **Tennis-ball map:**
  - **Component 1** (size: $n \times 3n$ pixels)
    $(x, y, z) = R/\|R\|$
    $a = asin(-x)/\frac{1}{4}\pi$
    $b = atan(y, -z)/\frac{3}{4}\pi$
    Pass1 output vertex $= (a,\ b,\ \|R\|/100 + b^2 - 0.75)$
    Pass2 lookup texture coordinate $= (a, b)*(½, ½)+(½, ½)$

  - **Component 2** (size: $3n \times n$ pixels)
    $(x, y, z) = R/\|R\|$
    $a = atan(x, z)/\frac{3}{4}\pi$
    $b = asin(y)/\frac{1}{4}\pi$
    Pass1 output vertex $= (a,\ b,\ \|R\|/100 + a^2 - 0.75)$
    Pass2 lookup texture coordinate $= (a, b)*(½, ½)+(½, ½)$

- **Dual Paraboloid map:**
  - **Component 1** (size: $n \times n$ pixels)
    $(x, y, z) = R/\|R\|$
    $d = 1 - z$
    Pass1 output vertex $= (x/d,\ y/d,\ -z\|R\|/100 - 0.9)$
    Pass2 lookup texture coordinate $= (x/d, y/d)*(½, ½)+(½, ½)$

  - **Component 2** (size: $n \times n$ pixels)
    $(x, y, z) = R/\|R\|$
    $d = 1 + z$
    Pass1 output vertex $= (x/d,\ y/d,\ z\|R\|/100 - 0.9)$
    Pass2 lookup texture coordinate $= (x/d, y/d)*(½, ½)+(½, ½)$