CS569 Assignment 2

Due: March 4, 11:59 pm.

Instructor: Steve Marschner

Overview

This assignment is all about textures – you will use both static and dynamically generated texture maps together with vertex and pixel shaders. As in the last assignment, it is strongly recommended that you work in groups of two. Cite any resources that you have used. The main tasks are:

1. To become familiar with the added framework code, and to add a texture preview function.
2. Using texture maps to implement three additional GLSL shaders:
   - Phong with a texture map for the diffuse component
   - Normal-mapped Phong
   - Reflection-mapping using cube maps
3. Adding support for the *frame buffer object* (FBO) OpenGL extension.
4. Implementing hardware shadow maps using FBOs to generate depth textures.
5. **One** of the following two:
   (a) Adding support for dynamically generated cube maps and using them to create fake reflections (also using FBOs). Additionally, you will have to add a so-called "sky-box" to surround your scene.
   (b) Writing a shader based on the 2005 SIGGRAPH paper "Measuring and Modeling the Appearance of Finished Wood" by Stephen R. Marschner, Stephen H. Westin, Adam Arbree, and Jonathan T. Moon. If you choose to work on this part, your efforts can additionally earn you **up to 10%** of extra credit.

Getting started

First, go to CMS and download the new framework version. It contains all solutions to the last assignment as well as new infrastructure to support texturing. You will notice that a few other things have changed as well: For example, it is now possible to watch the scene from a static position while rotating the light source. There also is a new "Texture" menu, which can be used to load arbitrary bitmaps into texture memory. For compatibility reasons, the `TangetSpaceMeshObject` has been modified to pass the tangents to the vertex program using the `gl_Color` attribute. The `Viewer` class contains a new method `initializeTextures`, which loads a set of specified texture maps and initializes any implemented frame buffer objects when starting the program.

Have a look at the class `cs569.texture.Texture`. It contains a method `blit`, which should display a texture to the screen by drawing a single screen filling quadrilateral with the texture mapped on it. When completed, the texture menu's preview functionality will correctly display any textures you have loaded into memory.

Shaders

In this part, you will implement three texture-based shaders: `TexturedPhong`, `NormalMappedPhong` and `Reflection`.

1. **TexturedPhong**: First, adapt the existing Phong model so that it uses a texture map to modulate the diffuse component.

2. **NormalMappedPhong**: The `texture` directory contains a *normal map*, which stores encoded normal vectors in the tangent space of the object.

   The normals were converted into colors values by applying the following transformation:

   $$\begin{pmatrix} r \\ g \\ b \end{pmatrix} = \frac{1}{2} \begin{pmatrix} x+1 \\ y+1 \\ z+1 \end{pmatrix}$$

   Normal maps are often used to create the impression of complex geometry without having to store and draw massive amounts of triangles. Write a new Phong normal mapping shader that performs the shading in the object's tangent space and uses perturbed normals from the map.

3. **Reflection**: A cube map is a series of six images, which correspond to the sides of a virtual cube (three sample cube maps are provided with the framework). Spherical texture look-ups can then be performed by intersecting a ray with the cube and returning the color value at the intersection (All of this is accelerated by the hardware).

   Write a shader that renders surfaces as perfect mirror reflectors in a cube map environment, by computing the reflection direction and corresponding texture coordinates, then to look up the the reflected color in the cube map.

Frame buffer objects

Frame buffer objects are a new extension of OpenGL. They are not always supported on older hardware (but they will work in Rhodes 455). FBOs are primarily meant to facilitate pre-process operations, which should not be visible on the screen. They are much more convenient and faster than the older and commonly used PBuffers. In this part of the assignment, you will use FBOs to add render-to-texture features to the framework.

The class `cs569.texture.FrameBufferObject` already contains some initialization and error detection code. Write the parts that create the actual FBO objects as well as any required renderbuffers and textures. Before the end of this initialization, it is crucial that you check for any kinds of errors using the `checkForErrors()` method. The reason is that most graphics drivers have very strict expectations on what can be attached to an FBO and what can't.

Hardware shadow maps

Shadow maps are just one of many methods used to produce shadows in 3D applications. Their efficiency on current GPUs and their independence of the scene's geometry make them a good choice for interactive applications.

Shadow mapping works by rendering the scene from the light's position in a pre-process step. However, instead of storing the rendered color values, only the depth information is kept. Later, when the scene is rendered from the actual viewpoint, this depth information can be used to determine whether a fragment has a free line of sight to the light source. You will use your FBO implementation to render into a *depth texture* during a pre-process step. Afterwards, write the `ShadowedPhong` shader, which makes use of this depth texture to visualize any shadows cast by the light source.

Choice A: Dynamic Cube Maps

One very useful application of FBOs is to render directly into the sides of a cube map. This texture can then be used to add fake reflections to objects contained inside the scene. Extend the framework to have such a cube-map generation pre-process. You should also add a "sky-box", which is a textured box surrounding the entire scene. Objects with the cube map applied to them should reflect their surroundings as well as this virtual sky.

Choice B: Finished wood shader

The 2005 SIGGRAPH paper "Measuring and Modeling the Appearance of Finished Wood" presents an efficient technique to visualize the distinctive appearance of wood coated with a transparent finish. Write a GLSL shader implementing this BRDF model. The framework comes with measured data for several different wood samples. On the paper's website, you will also find a working RenderMan shader, which you may use as a reference for your own implementation.

Acknowledgments

The assignment's software framework was originally written by Adam Arbree for CS467/CS468 (Instructor: Kavita Bala). It was adapted for CS569 by Wenzel Jakob.