Project 8 Final Project

Introduction to Computer Graphics, Fall 2011

1 Introduction

The **demoscene** is a computer art subculture that specializes in producing demos, which are non-interactive audio-visual presentations that run in real-time on a computer. The main goal of a demo is to show off programming, artistic, and musical skills. Here is a compilation (by nVidia) of some of the best demoscenes: http://www.youtube.com/watch?v=PidTKpKLYZM.

This fall we've surveyed a number of topics in Computer Graphics to give you a broad introduction to the field. Now, we're giving you the reins. Pick your favorite topic, do a little reading and research, and make a cool demo. In the final project you will implement some of the advanced rendering effects and modeling techniques discussed in class, which have not previously been covered in other homework assignments, and show them off in an interesting scene. We provide a list of suggestions, but you are free to choose what scene and effects you want to implement, as long as you clear it with the TAs. We will evaluate your project based on technical and artistic merits (music is not required). You can find the previous year's final projects in /course/cs123/final_projects_2010.

2 Requirements

Teams. We highly encourage you to work on this project in teams of two students. You may work by yourself, but it will be much easier and you will be able to accomplish much more if you work in pairs. The requirements of this project are the same whether you work alone or in a pair. Furthermore, by combining your efforts you have the opportunity to implement more advanced rendering effects and show them off in your demonstration. You can also talk through difficult concepts together. If you cannot find a partner the TAs will help you find one. Please make sure that you partner with someone whose schedule is compatible with your own. The TAs will not mediate scheduling conflicts between partners.

This Week. Your group must sign up for a mentor TA by Sunday, 11/20. The signup sheet will be posted on the door of the fishbowl (CIT271). While you may ask for help from any other TAs on hours, most of your project-specific questions should be directed to your mentor TA. You should email your mentor TA once you've signed up to schedule a time to meet after the break (or before, if you're ambitious) for a design check. For the design check we ask that you have a good idea of what you and your partner would like to do to so that you can it discuss with the TA and get useful feedback. Please email your TA your project ideas before the design check.

We will apply a penalty of 10 points if you fail to sign up for a design check on time (you cannot use a late pass).

Final Deadline. The final project is due by 9AM on 12/14. Late submissions will not be allowed, and you cannot use a late pass on the final project. Your final presentation (more on that later) is also due at this time.

Implementation. This project must be implemented in C++ using OpenGL/GLSL and, optionally, Qt (highly recommended!). We allow and encourage you to re-use the code you wrote for the earlier projects in this semester, including labs and the support code used for each lab. You are permitted to look at code from books and on-line sources, but you may *not* copy others' code and call it your own. You may use third party libraries if you wish, however, they must be cleared with your mentor TA first. Keep in mind that your use of a third party library does not decrease the amount of work we expect you to do. Using a third party to make one part of your program easier means you must put more work into other part of your program. In the end, the amount of work done by you should be the same.

Theme. Many of the recent labs provide environment maps that give the feel of a room or outdoor space. Your project should similarly have a *theme*, a cohesive setting. You can create a simple environment map, or you can model an environment using 3D geometry. You can also experiment with combinations of both, for example, by creating a skybox. The theme may also provide basic interaction controls to the user. Here are some ideas for themes:

- A museum room. Include artwork as texture maps. Include lights, benches, wood floors, rugs, doors, etc. if you wish. The user should be able to navigate the scene and change view direction with keyboard or mouse controls.
- A planet modeled using fractal terrain, with a sky box.
- Build a virtual roller coaster. Let the user's viewpoint follow along (in or behind) the roller coaster car. Include some interesting scenery.
- Take photographs of a room or two from around Brown or from your residence with a digital camera. Also take photographs of furniture. Make a three-dimensional model of this area and its furnishings[1], using your photographs as texture maps. Allow the user to navigate with arrows or other keyboard controls around the scene.
- Render a number of marbles bouncing around and bumping into each other, following the laws of physics, and casting shadows on each other and the surface

[1]Google Sketchup (freeware) has a large database of furniture models and can export to the OBJ mesh format.

they are on. You might want to apply a texture map to the surface and include a skybox or a "fog" in the distance.

Technical Features. By default, you must implement at least one technical feature covered in prior assignments/labs and at least two features *not* covered in prior assignments/labs. You must also have at least two different shaders. If you want to implement something really complicated and want to focus on that one feature, your mentor TA can waive the requirement for a second technical feature. Here are some ideas for technical features you might implement (of course, some exploration and research will be needed here).

- Real time per-pixel illumination with multiple light sources and different light source types such as spot lights; combination with texture mapping: ability to enable/disable shader.
- Bump mapping or displacement mapping (not both): enable/disable
- Piecewise Bezier surfaces
- Move the camera or objects in the scene along a path defined as a piecewise Bezier curve, for example to render a ride on a roller coaster
- Fractal terrain
- Fractal plants or trees
- Complex, procedurally modeled city or other environment (more than just a bunch of cubes or other simple geometric shapes)
- Lindenmayer Systems
- Collision detection (bounding boxes/spheres or polygon level): enable/disable mode in which colliding objects are highlighted
- Shadow mapping, with ability to toggle shadows on/offs
- Shadow volumes, with ability toggle shadows on/off
- Depth of field
- Deferred lighting
- Ambient occlusion

Here are some examples of technical features that you have already seen in the assignments; again, this is a non-exhaustive list:

- Glass, cartoon, pulse shaders
- Texture mapping
- Environment mapping, with ability to enable/disable environment map
- Particle effect
- Procedural geometric shapes like cone, cube, sphere, cylinder, etc.

Real-time. Your demo must run in real time, i.e. at several frames per second on the machine you plan to present on. Keep in mind that the sunlab machines are not exactly top of the line, so if you wish to have more GPU power, consider bringing your laptop in on demo day - but please don't bring a full desktop computer. This is not the time to implement Metropolis Light Transport.

Presentation. Presentations will take place in the Sun Lab on 12/14 from 9AM to noon.. You and your partner will give a brief 3-minute presentation and explain what you did while showing your implementation to the class. You can bring your own computer and plug it into the projector, or you can use the CS machine that is attached to the projector. The projector has a resolution of 1024x768. Make sure you practice before the final presentation day!

Non-Graphics Features. Although we will appreciate and enjoy any additional features you choose to add to your project, we are not requiring that you do so. We request that you spend any extra time that you have implementing more advanced graphical features, rather than sound effects, music, game logic, etc.

3 Grading

Your grade for this project will be determined by several factors.

Technical excellence. How well did you accomplish what you set out to do? Is your implementation solid (i.e. bug-free and free of memory leaks)? Does your project represent a significant, yet reasonable extension on the topics covered in the labs and assignments? We also expect that you will do something non-trivial. Your mentor TA will be able to give you a better idea about what this means.

Artistic quality/creativity/theme. Does your demo look and feel cohesive? You don't have to create a full-fledged environment, but we also won't be impressed with a single shape in the center of the screen. For example, if you're going to make an awesome shader, compliment it with a compelling environment map, like we do in the labs. We also encourage things like nice 3D models (downloading models from the internet or designing them with a 3D modeling tool is encouraged - but make sure you complete the basic requirements of the project before spending the time to model your own objects), nice textures (consider taking digital photographs and converting them to textures), the choice of colors and materials, the positioning of camera and lights, user interaction, etc.

Presentation. Does your program fit in the screen resolution provided by the projector? Does it look like your presentation was thrown together at the last minute? Is your presentation sensible and well-organized? Were you able to effectively explain your implementation in the 3 minute period?

Time management. Did you meet all the deadlines?

A note about partners. By default, both partners in a group will get the same grade. If you have any problems with your partner, you must tell us about them as soon as the problem occurs; we will not make retroactive adjustments to grades due to partner issues we didn't know about until the last minute.

4 Handing In

To hand in your code, run `/course/cs123/bin/cs123_handin final` in the directory containing your code. If you use Qt, you should turn in a .pro file which we can use to build your project. Otherwise, you should turn in a Makefile. Make sure your build scripts do not contain absolute paths or we will not be able to compile your code.

Before you hand in, please create a Readme.txt file which details the design decisions you made and identifies any bugs or known issues in your program. Make sure you thoroughly test and debug your program and ensure that it does not have any memory leaks.