1 Introduction

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.

2 Requirements

Teams. We highly encourage you to work on this project in teams of two or three 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 teams of two or three. The requirements of this project are the same whether you work alone or in a team. 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.

Project Proposals. Your group must submit a project proposal by Sunday, 11/12 at 5:00pm. The proposal should be a brief description of the theme and technical features you plan to implement. Possible options for features are listed below. Also include the logins of all group members. If you have a preference for a TA you would like to work with, you may also list that here (but we make no guarantees about the TA you will be matched with). To hand in the proposal, run cs123_handin finalProposal from within a directory containing a file with your proposal. Only one group member needs to hand in your proposal. Late project proposals will receive no credit. You cannot use a late pass.

Mentor TAs. Your group will be assigned a mentor TA shortly after proposals are handed in. This TA will be your go-to resource when working on your final project. Design Checks. You should email your mentor TA once you’ve been assigned one to meet for a design check. Design checks should take place by Friday, 11/18. This will be an opportunity for you to discuss your ideas with the TA and get useful feedback.

Project Plans. After meeting with your mentor TA, your group must make a detailed plan for how you will implement the technical features in your final project. This should include:

- Resources that you will use while implementing each technical feature. This may include SIGGRAPH papers, tutorials, or anything else you find that will be helpful when learning about the feature and how to implement it.
- General description of how each feature will be implemented. You may want to include pseudocode for any complicated algorithms that will be involved.
- High-level overview of the flow of your final program. How will all the pieces fit together?
- Division of labor and plan of action. Who will do what? Where will you start?

The purpose of the final project plan is to ensure that each part of your project is doable, and that you have a plan for how to implement it before you start. To hand in the plan, run cs123_handin finalPlan from within a directory containing a file with your plan. Only one group member needs to hand in your plan. Plans are due on Saturday, 11/19 at 11:59pm.

Late project plans will receive no credit. You cannot use a late pass.

Project Status Updates. By Saturday, 12/10, you should meet with your mentor TA to show your progress on the final project. You must have some preliminary results to show at this meeting.

Final Deadline. The final project is due by demo day, 12/18 at 11am. 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 online 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.
Support Code. Optional starter code for this project will be published in /course/cs123/src/final. The starter code creates an OpenGL window with mouse and keyboard input events using Qt. Once again, feel free to augment the support code.

Theme. Some of the recent labs provide environment maps that give the feel of a room or outdoor space. Your project should similarly have some sort of theme. 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 planet modeled using fractal terrain, with a sky box.
- 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.
- Build a virtual roller coaster. Let the user’s viewpoint follow along (in or behind) the roller coaster car. Include some interesting scenery.
- 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 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 one custom shader. 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):

- Bump mapping or displacement mapping (not both) with ability to 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
- 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/off
- Shadow volumes, with ability toggle shadows on/off
- Depth of field
- Deferred lighting
- Ambient occlusion (for example, SSAO or screen space ambient occlusion)
- Anti-aliasing (for example, FXAA or fast approximate anti-aliasing)
- 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
- GPU raytracing
- HDR, perhaps with Bloom
- Non-photorealistic rendering

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

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

Real-time. Your demo must run in real-time, i.e. at a minimum of 30 frames per second on the machine you plan to present on. The machine in the presentation room performs comparably to the Sunlab machines with NVIDIA GTX 970 GPUs. Alternatively, you may present on your laptop.

Presentation. Presentations will take place on Sunday, 12/18 at 11am in CIT 368. Your team will give a brief 3 minute presentation explaining what you did and 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 in the room we are using.
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.

Virtual Reality Projects. As of 2016, we’re allowing (some) VR projects! The department will allow a few select teams to use its HTC Vive headsets for the duration of this project. Deciding to do a VR project is a serious commitment and will significantly increase the amount of time and effort needed to produce a satisfactory final project. If your team would like to do a VR project, state so on your project proposal. The project specifications are the same, but more interactivity is encouraged in order to take advantage of virtual reality. All development will still be in C++, OpenGL/GLSL, and Qt. Your team will have to work with the C++ device API for the HTC Vive headset in order to integrate your project into virtual reality. If you end up on a VR project, you will be assigned a mentor TA who is familiar with virtual reality.

3 Grading

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

Proposal. Did you hand in a proposal on time?

Plan. Did you hand in a detailed plan on time?

Technical excellence. How well did you accomplish what you set out to do? Is your implementation stable (i.e. bug free and no 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 guide you here.

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, nice textures (consider taking digital photographs and converting them to textures), thoughtful choice of colors and materials, careful positioning of camera and lights, significant user interaction, etc.

Presentation. 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?

A note about partners. By default, all members of a group will get the same grade. If you have any problems with a teammate, 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 project proposal, run cs123_handin finalProposal in the directory containing your proposal file.

To hand in your project plan, run cs123_handin finalPlan in the directory containing your plan file.

To hand in your code, run cs123_handin final in the directory containing your code file. 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 your code, please create a README.txt file which describes how to use the program, details the design decisions you made, and identifies any bugs or known issues. Make sure you thoroughly test and debug your program and ensure that it does not have any memory leaks.

5 Deadlines Summary

- Saturday, 11/12 at 5pm: Project proposals due.
- Friday, 11/18: Meet with mentor TA for design check.
- Saturday, 11/19 at 11:59pm: Project plans due.
- Saturday, 12/10: Meet with mentor TA for project status updates.
- Sunday, 12/18 at 11am: Final project handins due.

6 The “Oh God Nothing Is Working” FAQ

When programming in OpenGL, sometimes you’ll get a black screen. There are a number of causes for this, including but not limited to:

- Drawing triangles clockwise instead of counter-clockwise
  - To debug, try reversing the winding order or using glEnable(GL_CULL)
• Camera facing the wrong direction, or near/far planes don’t contain object being rendered
  – To debug, try configuring your camera so you can view a single triangle in the center of the world

• Data being sent incorrectly to the GPU
  – You might see a black screen if your arguments in glVertexAttribPointer, glBufferData, or glDrawArrays don’t match the size or organization of your VBO. It’s also worth triple checking your uniforms. Try simplifying by rendering a single triangle.

• Lights misconfigured, or incorrect normal vectors
  – To debug, try setting glClearColor to any color but black. Alternatively, try having your fragment shader output a solid, bright color. It’s possible your objects are appearing, but being shaded in all black due to issues with lighting.

• General tip: if you’re having troubles with complex shapes, just try drawing one triangle and see if you can get that to render, and build from there. As always, you can come to hours if you’re really struggling and you’ve already tried all of these things.