CS 371 Project 5:

Midterm Meta-Specification

Preproduction Start:	Thursday,	September 30	1:00 pm
Proposals Due:	Wednesday,	October 6	12:00 pm
Description Due:	Thursday,	October 14	10:00 pm
Production Start:	Monday,	October 16	9:00 am
Checkpoint 1:	Wednesday,	October 20,	10:00 pm
Checkpoint 2:	Friday,	October 22,	3:00 pm
Project Due:	Wednesday,	October 27,	10:00 pm

1 Introduction

This project is an opportunity to explore a topic of your own choice in modeling and physically-based rendering. You choose the topic, the team, the tools, and the specification yourself.

This is also a time to learn about what happens before and after the development part of a project. For this project, you're in charge of the process of specifying your project and then presenting it afterward. Those steps are often more critical to the success of the project than development itself!

You should scope the project based on the expertise and time constraints of your team. A two-person team might choose to implement a software rasterization algorithm for eye rays. A four person team might chose to implement refraction, real-time ray tracing, and implicit surfaces. Keep in mind that this is a moderate, team project, and that presentation and preproduction are new parts of your workload.

1.1 Preproduction

Every research and industry project undergoes a **preproduction** stage. This is preliminary exploration before committing to deliverables and scaling up resources to a full project. For the midterm, you will run preproduction as a background activity in parallel with the Photon Mapping project. During preproduction you will create a plan for who is in your group and what your project goals are.

Brainstorming, reading papers and non-assigned chapters of the textbook, and meeting with me are the primary methods that you will use to refine your ideas. You are welcome to write some exploratory code but **should not be spending more than six hours total in preproduction**. That's not a lot of time! I have a lot of experience with the regular projects in this course, and it still takes me about 40 hours of preproduction to create the specification for each of the regular projects. It takes me that long to write a specification because I have to plan it in such a way that you don't run into roadblocks when implementing it. Since you won't be able to do the entire project multiple times during preproduction like I do, that

means that the scope of your project must also be smaller than the regular projects. You should anticipate many things not working out the way that you planned and reserve time to work around that.

2 Educational Goals

On this project you're learning is the *process* of running a project independently. If you learn the process and your program doesn't produce the right image, that was still a success. The process includes:

- Scope the work
- Research previous ideas in this area
- Plan the software architecture
- Choose tools
- Design experiments
- Plan a presentation
- Create a robust schedule and plan
- Plan a presentation

You'll also learn a lot about one specific aspect of a modeling or rendering program. What you learn there is up to you. Part of your project description will be what you think you're going to learn. Part of your report will be describing what you actually learned, which hopefully will overlap with your own educational goals.

3 Rules

I'm willing to negotiate any of these rules. For example, in the past I've been persuaded to allow the entire class to work on the same project, and to allow students to use custom equipment. You are more likely to convince me if you approach me early in the preproduction stage.

1. Each project group must have 2-4 people[1]. Groups can mix students and game developers, and that is encouraged. However, be aware that the game developers may have to leave the project early if their work responsibilities change unexpectedly.
2. Once you start production, the top of your report must always contain a Status section that contains a single image and a bulleted list or short description that together describe current issues and next steps. This will help your team coordinate and will help me monitor your progress. You're encouraged but not required to keep a log on a separate page that shows how these evolved over time.

[1] Corey has no-one else local, so he may work solo if he wishes.

3. Different groups are allowed to work on projects with similar specifications.

4. The project must be about modeling or physically-based rendering. It does not have to be a ray tracer.

5. Your group (or groups, at this stage–they aren't locked down yet) must submit three one-paragraph ideas on paper during preproduction. These should be fairly different from each other, and all ideas that you're interested in developing. I'll approve a subset of these for you to continue with.

6. You must have a Project Description in PDF format approved by me at the Description Due deadline. That means that you need to submit revisions to me well before that deadline. The project description should look like the ones that I distribute each week. It must contain:

   (a) Team members (names, e-mails, and svn names)
   (b) Group name
   (c) Specification. Be careful and explicit. I'm going to use this to evaluate your success.
   (d) Rules and Honor Code
   (e) *Two* checkpoint specifications
   (f) Report specification. This is your plan for how to present your work. Since the presentation and results are your primary deliverables (...and not the software artifact that you used to produce them!) this is what ultimately drives your project. This is therefore the most critical part of your project description. It should be much more detailed than the report specifications that I give you to reduce your risk, and also because unlike regular projects, I don't know what you're working on beforehand.
   (g) The report must contain a typical description of how you spent your time and what you learned. Distinguish time to meet your minimum specification from polish time, and preproduction/production/presentation time.
   (h) Informal description in support of your project, parallel to my Advice section. What tools will you use? How will you debug? What are your contingency plans?

7. The project must compile and run with iCompile on my OS X laptop, and the documentation must be able to be generated by iCompile on my laptop.

8. Your project may use any code from any source. All code that is not written by you must be properly attributed in your documentation. The part that you implement must represent the core educational value of the project.

9. After your project has started, you may adjust your specification with my approval. I primarily approve specification changes that are due to new facts that you discovered during production. I do not approve changes that are motivated by to poor time or team management on the part of the group.

10. Your group will make an in-class presentation at the conclusion of the project. The length will be determined by the number of groups–probably around 15 minutes. This should be informal but professional, like the colloquium talks by external speakers. It should emphasize qualitative and quantitative experimental results. A good strategy (which we'll discuss further in class) is:

    (a) Lead with a compelling image–tease the audience
    (b) Explain the key mathematical or algorithmic idea in less than 2 minutes
    (c) Show experimental results for detailed, isolated cases that confirm the correctness of your method
    (d) Show a failure case. This is where your algorithm is pushed beyond its applicable domain, not a bug in your program.
    (e) Show and interpret quantitative results related to scalability, performance, quality, etc.
    (f) Show compelling images and videos that
    (g) Conclude with what the next steps would be if you continued the project

    I recommend that you plan to have at most one slide per minute and no animated slides (except for video). Test and rehearse your presentation several times before you present. Use plain but precise language. Avoid telling the "making of" story, inside jokes, and casual speech.

4 Evaluation

I will evaluate your proposal, your development, and your presentation. The checkpoints will be pass/fail. I will evaluate your report and source code as for other projects, but using your own specification as the rubric. I'll evaluate your proposal for completeness, references, and presentation quality. The quality of your idea does not affect the grade; if I approved the project, your idea is good.

There should be constant communication between us about the status of your project, both passively through your status updates and actively through lab, office hours, and e-mail. This means that you should always have a good idea of how I would evaluate your project. If you are ever unsure about how you're doing, that is a sign that you should be talking to me more.

5 Some Ideas

Here are some ideas from our lab brainstorming session. You might choose a few of these, or use them as a starting point for your own ideas. I've annotated them with additional technical terms that you'll want to look up to learn more about each topic.

1. Refraction (transmission)
2. Software rasterization of eye rays (see textbook)
3. Multiple wavelengths

4. Remote [server-side] rendering (see the G3D remoteRender sample)
5. Voxel rendering (marching cubes, grid, voxels)
6. New primitives: square, torus, cylinder, cone, capsule, gaussian
7. New platforms: JavaScript, iPhone, DS, Android, Flash, calculator, Excel, Matlab, Scheme, Arduino
8. True curves and amorphous surfaces (subdivision surfaces, metaballs, implicit surfaces, level set surfaces, splines, NURBS)
9. Stereoscopic 3D rendering (we have anaglyph red/blue glasses and a shutter-glass display)
10. Rendering huge models (out of core rendering)
11. Skin (subsurface scattering)
12. Natural language and abstract scene descriptions
13. Motion blur (distribution ray tracing, path tracing, stochastic ray tracing, distributed ray tracing)
14. Fur and hair (deep shadow maps, displacement mapping, shells)
15. Clouds and fog (participating medium)
16. Using the sky as a light source (ambient occlusion, environment mapping, area light sources)
17. Using surfaces as light sources (area light sources, volume light sources)
18. Animation (note that G3D contains several animated model classes, a video input and video output class, and "spline" primitives for specifying smooth motion as used in the starter)
19. Lens camera (depth of field, depth of focus, distribution ray tracing, distributed ray tracing, stochastic ray tracing)
20. Antialiasing (supersampling, adaptive supersampling)
21. Real-time ray tracing (adaptive refinement, networked ray tracing, software rasterization, grid tracing, frameless rendering, reprojection, irradiance caching, light maps, OptiX)
22. Terrain-optimized rendering (heightfield tracing, voxel tracing, atmospheric perspective, sphere tracing, min/max mipmaps)
23. Modeling terrain (fractal terrain, terrain texturing)
24. Light rays through clouds and fog (crepuscular rays, single-scattering, god rays, participating medium)