CS488 PROJECT PROPOSAL 2

Spring 2011

Anthony Cameron
20231689
a5camero

For Gladimir Baranoski

CS488 Project Proposal 2

Topics/Purpose

Continue to explore various ray-tracing technologies, potential efficiency gains, and advanced rendering effects such as reflection, refraction, textures, bump mapping, and potentially radiosity-based global illumination techniques. The goal is to allow a user to create a realistic rendition of a scene, while still allowing the user to control the viewing angle and modify material properties without having to alter a script file and/or wait an absurd amount of time to see the final (potentially undesirable) rendition of the scene.

The focus is on simulating material properties and their effects on light with respect to simple surfaces/objects, expanding to more complex surfaces if time permits.

This project was designed in mind to alleviate my original frustrations with A4 – waiting inordinate amounts of time to see I had rendered a scene with no objects in it (poor camera placement), constant annoyance with tweaking small parameters that might be better tweaked in visual space to obtain a visually pleasing scene, and immediate user feedback with respect to rendering.

Statement

A "real-time" raytracer – the initial render is low resolution, and as time passes, the render is updated periodically (every X seconds, user configurable) with the newly calculated light values. The user has an option to save the current picture state or toggle automatic ray tracing at any time.

OpenGL will be used to output the current pixel buffer to the UI.

My extra feature in A4 was super-sampling – I used 4 subpixel rays, positioned halfway between the left/right/top/bottom of the real pixel and its neighbor pixel.

Planned Scene

A room with 4 walls, a square hole in the ceiling where the light will be coming in. There may be an arbitrary number of depressions/elevations in certain walls for the purpose of testing shadow effects in the world. Objects may be placed near the walls and in the open. 1 spherical and 1 planar mirror will exist to demonstrate reflections. A bump-mapped diffuse object will be included to demonstrate bump-mapping. The floor will be a texture-mapped hardwood floor, ideally glossy (time permitting).

TECHNICAL OUTLINE

Materials will be altered to accept a reflectivity index between [0,1]. 0 represents no image reflection, whereas 1 represents complete image reflection. Specular reflection will continue as per usual.

Primitives will have an option to specify a height-map texture file for bump-mapping purposes. Each primitive that can take a height-map will then have its own procedure to determine exactly how to apply it to itself. For example, spheres will require a different interpretation of a square height-map than might rectangular prisms to distortion.

Primitives will have an index of translucency from [0,1]. 0 represents complete opacity whereas 1 represents complete transparency. This represents the fraction of transmitted rays.

A 3D array might be used for uniform spatial subdivision. The spatial bounds thereof are determined by the positions of the furthest objects that lie on the extremities of the scene.

Two pixel buffers are maintained, one for use in standard OpenGL projection matrix while to provide a quick and simplistic view of the scene while rotating / changing material properties, the other to store the results of the ray tracer thus far for said current view. It may be a good idea to include many buffers to act as an "undo" function, which would be helpful if the user accidentally moved a scene that had been rendering for many minutes.

Algorithmically, I believe I'll need some kind of quad-tree system in pixel space to do the progressive enhancement. This could potentially lend itself well to "easy" CPU parallelization, a possible future avenue for this project.

Perhaps the best approach would be to use a rasterizer (à la OpenGL projection matrix) while altering camera view to obtain a nice >=30FPS, then slowly overwrite this image with new ray tracer results after user interaction has halted.

Some attempt may be made to avoid recomputing unnecessary pixels after an object's material properties have been altered. I describe one potential scheme: seep outwards from the boundaries of the altered object (in pixel space) as if you were a slime mould, calculate colour differentials. Stop a "tendril" after the newly computed colour is below a certain visual threshold (obtained empirically). This could probably just be implemented in a scan-line approach based upon the bounding areas of the altered object. If altering a tiny diffuse object that takes up 10% of the scene and offers very little to global illumination, we might only need to re-render 20% of the pixel space before its colour contribution is not noticeable. If colour bleeding is to be implemented, a simple linear scanline approach may not be sufficient and instead some random ray sampling might be more desirable to find large colour differentials. Special consideration may need to be given with respect to radiosity, so this paragraph might be attempted only if there is extra time.

Otherwise, everything else will be implemented as discussed in class with minimal consultation of various Internet resources when necessary.

OBJECTIVE LIST:

1. Glossy surfaces
2. Texture mapping
3. Shadows, both hard and soft
4. Reflections, various degrees of reflectivity are possible
5. Uniform spatial subdivision is employed to speed up the processing by eliminating unnecessary intersection tests
6. Bump mapping[i]
7. UI objective: user is able to reposition camera in the scene interactively at "real-time speeds" (hopefully >10 FPS)[ii], the user sees a lower detail version of the scene while moving. When the user stops moving the camera, he has the option to begin ray tracing.
8. Progressive refinement via radiosity: step manually, or step automatically towards N refinements (N is user configurable)
9. Refraction[iii]
10. Unique final scene

---

[i] "A Ray Tracing Method for Illumination Calculation in Diffuse-Specular Scenes (1990)", Peter Shirley, http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.5088

[ii] "Combining Rasterization and Ray Tracing Techniques to Approximate Global Illumination in Real-Time", João Cabeleira

[iii] "Refraction in Discrete Ray Tracing (2001)", David Rodgman, Min Chen, In Proceedings Volume Graphics 2001