CSE168: Rendering Algorithms
Assignment 1

Due Thursday, April 22 at 6:00pm

Henrik Wann Jensen
Toshiya Hachisuka

Introduction

This assigment involves implementing intersection of triangles and bilinear patches in Miro. Percentages indicate percentage value for this assigment.

Task 1: Triangle intersection: 25%

Implement a triangle intersection routine that computes the intersection of a ray and a triangle (specified as three vertices).

Task 2: Barycentric interpolation: 15%

Implement barycentric interpolation of normals for a triangle with three normals (one per vertex).

Task 3: Bilinear patch intersection: 50%

Implement a bilinear patch object capable of ray tracing (intersecting and shading) a bilinear patch. Additionally, the intersection point should only be accepted if ($u < 0.1 \parallel u > 0.9 \parallel v < 0.1 \parallel v > 0.9$). – i.e. effectively rendering the outer shell of the bilinear patch. Render a bilinear patch where the color is a function of the barycentric coordinates within the patch.

Task 4: Bilinear patch normals: 10%

Compute the normal of the barycentric patch at the intersection point. Render an image where the color is visualizing the normal (e.g. red = fabs(n.x), green = fabs(n.y), and blue = fabs(n.z)).

Hacker points:

Triangle intersection using Plücker Coordinates +8%

Implement the triangle intersection code with interpolation of barycentric coordinates using Plücker coordinates. Compute the barycentric coordinates directly from the Plücker evaluation.

SSE implementation: +10%

Implement the triangle intersection routine using one of SSE/SSE2/SSE3/SSE4 intrinsics or assembly code, so that a ray intersects one triangle as efficiently as possible. Report timings showing the speed improvement over the standard implementation.

Submitting Your Work

You need to show your work to the TA by the due day during lab hours. We encourage you to submit (show) your work earlier than the due day. If you wait until the due day, you will be facing a long waiting line for grading - it is not recommended.

In addition to showing your work, you are also required to submit your code starting from this assignment. To submit your code, you should package up all source files and project files into a `zip` archive. Do not include any of the provided `.obj` files or any generated files from compilation (i.e. clean your project). You should only need to include the `.cpp`/`.h` files, the `.sln` and `.vcproj` Visual Studio files, plus the Makefiles. If your code archive is larger than 200KB, you might be doing something wrong. Note that the grading itself is done on-site, so just sending the code is not enough.

You should also submit your rendered images at 512^2 resolution. Render the `sphere.obj`, `teapot.obj`, and `bunny.obj` models available on the class website. Render the bunny scene using the `makeBunnyScene` function already provided in the additional code snippet. Create two new functions, `makeSphereScene` and `makeTeapotScene`, to render the other two models.

Convert all your images to PNG format before submission (do not send them as .ppm files), and package up your images into a zip file.

Your archive and images should be named according to your last and first name. For instance, John Smith should submit `smith_john.zip` which contains `smith_john1.png` and `smith_john2.png`. Include the text "CSE168 Assignment 1" in the subject of your email. Please check you followed the instructions correctly before you send your images.

Email your archive to cse168-turnin@graphics.ucsd.edu.