CMSC 427 Computer Graphics
Programming Assignment 5: Ray-Triangle Intersection Acceleration
Assignment Release Date: April 27, 2017
Due Date: Friday, May 12, 2017 at 11:59pm
Project Submission:

1) Delete all intermediate files (run the command make distclean) and the Makefile created during the compilation process. Also delete the executable file.
2) Delete all models that were included in the starter code. We will have copies of these.
3) Place all files for the project in a folder and ZIP up the folder. You will submit your project via the submit server. To submit a zip file, login to the submit server webpage and look for the link to make a *web submission*.

Assignment Description

We have provided you with a basic ray casting renderer. It works by checking all triangles in the model for each viewing ray. Since models can have thousands or millions of triangles, this approach is very inefficient. The computational cost is O(# of trangles * # of pixels). With the current implementation, you will not be able to render all but models with very few triangles and large models at very low resolution. You should note that for these models the output is nearly the same as the OpenGL rasterization approach. Effectively, the use of ray casting provides visibility information for each pixel. This can be used to set the pixel color according to Phong shading. For this assignment, you are required to implement an acceleration data structure and add a few extensions to this renderer.

Bounding Volume Hierarchy (80% of grade)

Implement building and traversal of a bounding volume hierarchy data structure. We have provided you with a ray axis aligned bounding box (AABB) and ray-triangle intersection functions. Your approach should render large models with hundreds of thousands of faces within seconds for a wide range of a camera positions.

Surface Area Heuristic - SAH (5% of grade)

Using the SAH will give you a different split position than the default mid-point for constructing a BVH. You need to score each split position. This score is calculated as the following

A = (# of elements left of split) * (Surface Area of Bounding Box around these left-side elements)
B = (# of elements right of split) * (Surface Area of Bounding Box around these right-side elements)
Z = Total surface area of elements in current recursion

SAH Score = (A + B) / Z

You can pre-compute the bounding box areas for each of the splits so you do not have to re-build the bound box for each split you create.

Run several experiments with the SAH on and off and compare how it improves on the number of ray-intersection tests e.g. ray triangle and ray AABB tests. In the write-up explain how you conducted this experiment for a range of .OBJ files.

Textures - (10% grade)

Texture mapping: Implement texture mapping. The texture is to modulate the diffuse reflectance component across a surface, i.e. for every ray intersection at a surface point with texture coordinate (u,v), you will use Texture(u,v), kd, rather than kd of the material in the diffuse reflection calculations.

Shadows (5% of grade)

For every direct illumination calculation, cast a "shadow ray" from the reflection point to the light source position and include the contribution of that light source only if the shadow ray is not blocked by another surface in the scene.

Extra Credit (Up to 50%)
Implement any combination of the following:
Ambient occlusion.
Path tracing.
Soft shadows.
Anti-aliasing by randomly sampling rays offset from the center of a rendered pixel.
Inter-reflections.
Refraction.
Glossy reflection and refraction.
Ray differentials.

You are encouraged to discuss these extra credit options with the professor and/or the TA, should you have the required parts completed.

Model Files
For this assignment you will need to use multiple models to fully test your program on. We have provided them in the starter code, but please do not include them in your submission.

Grading
There are 2 parts that will be graded: 1) the write-up and 2) your code. The write-up should include screenshots demonstrating each operation you chose to implement. There should be at least 2 screenshots per operation (before & after) – enough to illustrate the correctness of your approach. Screenshots are required for all operations. The TA will use an undisclosed set of .OBJ files to test your implementation as well so we encourage you to find more online to test your program. We have provided you with several .OBJ files with textures to start off with.

Your code should be well commented in header files and source files. Note that it will also be judged subjectively based on the simplicity and clarity of implementation. An implementation that is easy to understand, but has few minor bugs will be scored higher than a messy implementation with the same number of minor bugs.

Documentation Grading

Please include documentation in the form of a Markdown (https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet) document with your assignment. It should be a text file in the folder ./cmsc427 called DOCUMENTATION.md. You will be graded on the completeness, clarity, and conciseness of your documentation. Your documentation should links to inline images showing the before and after of your image processing operations. It should also include an explanation of the algorithm you implemented. The more complete and clear your explanation is in this documentation the easier it will be for us to assign partial credit. It is in your interest to be as concise and clear as possible. Please make sure you only use relative links in your mark down file. If the TA has difficulty finding your images and the links, you will lose points. It is additionally suggested that you convert your Markdown file to .PDF so the TA.

Additional Submission Details
- Exactly one person of your team should be responsible for the submission. The latest submission will be the one your team be graded on.
- Your document should be properly written with markdown notations, and with correct extension names: it should be either .md or .markdown, not .txt. (If you prefer an extended version of Markdown (e.g. GitHub flavored Markdown or Pandoc's Markdown), it is also fine if you have included your PDF file.)
- It's recommended (required if you are using extended version of Markdown) to generate a PDF file for your markdown document. and put it under the same directory with same basename (i.e. DOCUMENTATION.pdf), take a look at it to make sure the layout is intended and

screenshots are properly included. If the PDF file is missing, your document will be converted from your DOCUMENTATION.md or DOCUMENTATION.markdown to a PDF file using pandoc before grading.

- Most online markdown editors would have the option to export your document as PDF files, and there are tools that do this Markdown-PDF conversion online.
- It's not recommended to include your images as website URLs or encode your images into URLs. and do not use local or private URLs otherwise I can't find your screenshots. You should include your images as separated files in your submission.
- Make sure to crop your screenshot properly, it should be enough to show your work, but not too much information (e.g. another screen with youtube or twitch opened).
- In your submission there should be just project files, screenshots, documents and other things as requested. Top-level zip files are fine, but do not add any other level of archive in your submission, which means things like *.zip, *.7z are not recommended, compressing data more than once barely gives you a smaller file anyways.
- For your project directory, it's recommended to use just a combination of alphabetic and numeric characters and underlines (or in other words, your project directory name should be an ASCII string accepted by regular expression \w+), otherwise qmake would have some trouble with project path.
- Your source code should be compatible with C++11 standard and cross-platform:
    - one way to make your compiler keep an eye on this is to ensure c++11 appears in the *.pro file of your project in CONFIG += ... line (which it should have been if you download the latest starter project). also try to address compiler warnings instead of suppressing them.
    - be consistent with file names, which means if you name one of your header Foo.h, you should write your include directive like #include "Foo.h", not #include "foo.h" or #include "FOO.H"