Assignment 3 - Ray Tracer

CS174A Introduction to Computer Graphics

Winter 2019

Due: Friday March 15, 2019, 11:55 PM
Points: 40 possible points corresponding to 15% of your grade
Collaboration: None allowed. If you discuss this assignment with others you should submit their names along with the assignment material.
Start working on this assignment early. You will not have time to do it well at the last minute.

Submission: Submit your assignment on CCLE in an archive named <uid>.zip, where <uid> denotes your UCLA ID. This should include ALL of the code necessary to run the program (where the host related files are optional), and should not contain any functionality beyond what is requested below. Include a README file that indicates which of the tests the output of your program passes and which ones if any it has not passed.

Instructions:
For this assignment, you will be implementing a **Ray Tracer** capable of handling the rendering of planes and triangles lit by point light sources. Local illumination with Phong shading, shadows, and reflections must be implemented.

The starter code for this assignment provides a skeleton implementation of a raytracer, and a set of tests to help with implementation and debugging. Please note that the code base for this assignment requires a modern browser, such as an updated version of chrome, to run successfully.

All of the code you need to implement is located in `core.js`, and is labelled with comments that begin with `TODO`. Additionally, the starter code includes an auxiliary html file (`barycentric.html`) to help visualize and debug your code for barycentric coordinates.

Your solution should not contain any WebGL or tiny-graphics API calls. However, you may use WebGL/tiny-graphics during debugging if you wish.

You are not allowed to publicly post your completed code online or share it with other students, both during and after your work for this assignment. If you wish to use version control (e.g., github, gitlab) to track your changes, your repository must be private.

Rubric [40 points possible]:

1. Intersection of rays with planes	[5 points]
2. Intersection of rays with triangles (using barycentric coordinates)	[7 points]
3. Local illumination with Phong shading (ambient, diffuse, and specular)	[10 points]
4. Shadows	[6 points]
5. Reflections	[6 points]
6. Speed	[3 points]
7. Coding Style (i.e., well designed, clean, commented code)	[3 points]

Notes:

- Do your implementation for this assignment incrementally. You'll find that if you follow the numerical ordering of the requirements in the rubric, you should be able to pass the tests using the same order. For example, after getting intersections between rays and planes working (Requirement 1), your code should be able to pass `1a_plane_silhouette` and `1b_double_plane_silhouette`. After implementing both Requirements 1 and 2, your code should be able to pass all tests beginning with `2`, including `2a_triangle_silhouette`.
- For all of the tests provided, your program should be able to finish in about a minute or less (given the default number of workers). Some of the tests will finish more quickly than others.
- When testing shadow rays from the intersection point to the light source, you are looking for any intersections with hit time between ~0.0001 and 1. To deal with numerical error, you should not consider an intersection at time 0 to be blocking the light from the object, as this may include the surface that is currently being tested for shadows.
- You should calculate the reflection color up to the depth specified in the testcase, so there should be no chance of infinite reflections even when between parallel planes.
- You will be using the Phong local illumination model with reflections, as follows:
    - $PixelColor[c] =$
    $$K_a\, O[c] + K_r * \text{(Color returned from reflection ray)}$$
    $$+ \sum_{p \in lights} K_d\, I_p[c]\, (N \bullet L)\, O[c] + K_s\, I_p[c]\, (R \bullet V)^n$$
    - O is the base object color.
    - $X[c]$ means that the X has three different color components, so the value may vary between the red, green, or blue component calculations.
    - The other components of this equation are explained in the lecture notes.