CSE 386
Fall 2021
Project Raytracing

Notes:

- Your project must be an interactive program that responds to the keystrokes, as outlined in the keystroke document.
- Three videos have been provided to illustrate the behavior of the project.
    - Video A - Positional light, Attenuation
    - Video B - Spotlight, Shadows, Objects, Textures, Transparency, Resizing
    - Video C - Light tied to camera, Reflections, Viewports, Anti-aliasing
- You may not use code distributed in previous semesters or code developed by other students; you are expected to complete this project on your own (of course collaboration at the pseudo code/algorithm level is appropriate and encouraged). It is also worth noting that there are subtle differences between this semester's code base and previous semesters. In addition, we may run scripts to detect unnatural similarities between all student submissions.
- fullraytrace.cpp has been provided to you to serve as a starting point for your main driver. Its main purpose:
    - Handle the keystrokes listed in the keystroke document. You should not need to alter the function keyboard().
    - You can add/delete items from your scene as needed.
    - You should not need to make changes to this file (other than adding/deleting from the scene).
- You may not use `glViewport` or any other `gl` or `glu` functions. We never talked about these and you should not be using them. Of course, you may use `glm` functions.
- If your render times are very slow while developing (not for your submission)
    - Make the window smaller (e.g., 200 wide by 100 tall).
    - Reduce the number of objects in your scene.
    - Run the program in release mode. You won't be able to debug in this mode, but it will run faster.
    - Comment out the call to frameBuffer.showAxes() in raytraceScene()
- You will submit three separate items to Canvas (follow the submission instructions). The time posted by Canvas will be used as the time of completion (latest of the three submissions).
    1. Video demonstration of your project. This video will be less than 7 minutes long and will demonstrate all the features of your program.
    2. A zip file containing your MS VS Project (or XCode project). Zip up your top-level folder containing all of the project. Then make the zip file smaller by deleting some unneeded directories. Open this file and delete:
        a. For Windows developers, delete the following directories from the zip file:
            i. CSE386/.vs
            ii. CSE386/packages
            iii. CSE386/x64 and CSE386/Release and CSE386/Debug
            iv. CSE386/CSE386/x64 and CSE386/CSE386/Debug and CSE386/CSE386/Release

v. After this, your zip file should be about 2MB

b. For XCode developers, simply leave all the files intact.

3. A PDF report (contents are described below see below).

(76) REQUIRED FEATURES

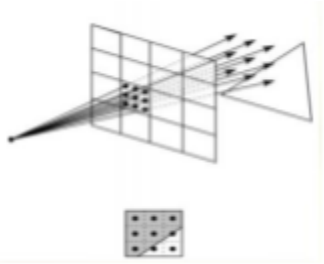
1. **(45) Lighting, shading, and shadows**. The lighting/shading of your scene must include:
   a. **(28)** One positional light and the correct implementation of the equations to render the objects with a realistic appearance. **(Video A time 0:00-2:15)**
   b. **(3)** Attenuation that can be turned on/off via keystrokes. **(Video A time 2:15-4:25)**
   c. **(7)** Spotlight. The spotlight will not illuminate any intersection points that fall outside the light's cone. If outside the cone, the light considers the intersection point as black; otherwise, the intersection point is rendered like a regular positional light. (Note: spotlights often have a different approach for rendering the interior of the cone. Our class will simply render the interior of the cone as would be done with a normal positional light.) **(Video B time 0:00-2:14)**
   d. **(7)** Shadows. Implement shadows by checking shadow feelers for each surface intersection. If the shadow feeling intersects with another surface before the shadow feeler reaches the light source, that particular light should provide only ambient contribution. **(Video B time 2:14-3:10)**
2. **(16) Objects.** Your scene must include, at a minimum, the following objects, each with their own material common properties (e.g., gold) **(Video B time 3:10-4:30).**
   a. **(7 pt)** 2 finite-length cylinders with open ends. One aligned with Y and one with Z axis. These must be different sizes and the interior of at least one visible from at least one camera position. Backfaces must be rendered correctly.
   b. **(7 pts)** Cylinder with closed ends and finite length. The cone should be aligned with the Y axis and the implementing class should be named `IClosedCylinderY`. You can implement this using a `ICylinderY` and two `IDisks`.
   c. **(1 pt)** At least 1 sphere
   d. **(1 pt)** At least 1 plane
3. **(7) Textures**. The Y oriented cylinder must have its sides textured (the video shows the X oriented cylinder being textured) with any recognizable PPM file (not a texture such as cement or a pile of rocks) that you choose. Do this by merging the texel value and lighted value (using the underlying, lighted, material property) at 50-50 proportions. The top and bottom, if any, do not need to be textured. **(Video B time 4:30-5:17 & 6:54-7:55)**
4. **(7) Transparency.** At least planar shape in your scene should be transparent. The position of the transparent object must be placed on a timer to illustrate that the display is correct, given its relative position to opaque objects. That is, the transparent object must move in front of and then behind opaque objects. **(Video B time 5:17-6:54)**

   Transparent objects are not to be textured or subjected to the lighting calculations. Instead, treat it as a simple color (i.e., its ambient color). Also, assume that at most one transparent object will intersect the viewing ray.

5. **(1) Resizing**. The display must not display skewed images when the window is resized. **(Video B time 7:55-8:32)**

(24) IMPLEMENT ANY 4 OF THE FOLLOWING 5 SECONDARY FEATURES - 6 PTS EACH

1. **Light tied to the world or tied to the camera**. When the light's position is (10, 10, 10), it is typically thought of as being at (10, 10, 10) in the global world coordinate system. However, it is sometimes useful to have the light's position relative to the camera's frame. Keystrokes should allow the user to switch between these two modes. Further, the keystrokes that alter the spotlight's position and direction should respond accurately. **(Video C time 0:00-5:05)**
2. **Reflections**. Add inter-object reflections by tracing a reflection ray to the closest surface intersection for each view ray. Once generated, the reflection ray can be traced in exactly the same way as the viewing rays. The best way to accomplish this is by calling the `traceIndividualRay` method recursively and adding what it returns to the total color for the pixel. To do this, it is necessary to keep the recursion from being infinite. This can be accomplished by adding an additional parameter -- `depth` -- to the `traceIndividualRay` method. This parameter can be reduced by one prior to each recursive call. The recursion would stop when the value is less than or equal to zero. **(Video C time 5:05-6:32)**
3. **Viewports**. The scene will be rendered from three different vantage points. The three images will occupy the lower-left, upper-left, and right-hand side of the screen. The camera positions must be chosen in a way to allow the user to obviously see that the images come from the same scene, yet different enough to allow the user to discern that the images come from different vantage points. **(Video C time 6:32-7:40)**
4. **Anti-aliasing**. Because of the discrete nature of raster image representation, rendered images will include aliasing artifacts. These artifacts create a jagged or stair stepped in objects that should appear smooth. In ray tracing applications, anti-aliasing can be performed by tracing multiple rays per pixel. Your approach should subdivide each pixel into the, say, 3x3 grid, casting one ray per subpixel. The resulting color should then be the average color of the 9 rays (for a 3x3). A 3x3 will provide a more refined image than a 2x2 or 1x1 grids. **(Video C time 7:40-8:55)**



5. **Cone.** Include 1 finite-length cone aligned with the y axis in your scene. The cone can have an open bottom. (Video B).

You must provide a report (PDF file) that contains:

- An itemized list of the functionalities that work and those that do not. You do not need to elaborate on how you did things; simply list features that work and those that do not.
- A minimum of three screenshots of your program's display should be inserted into your report. The images should be annotated to highlight the functionalities that were completed. This might include, for example, images of the scene with and without antialiasing. To do this, add a small number of screenshots and then annotating them with arrows and text to describe the feature. This

can done in MS Word, for example, by adding the image to the document and then adding textboxes and arrows to point out the specific features.

- If you do not provide a report, or has missing items, or is inaccurate, 15 points will be automatically deducted from your overall score. Your report should not be misleading or deceitful.

1. Create a video that is at most 7 minutes long. (It is OK if your video is 7:20 but it certainly should not be 10 minutes.) You should assume the audience is the instructor and is knowledgeable about terminology and techniques. Your video should include:
    a. When demonstrating your project
        i. Run your code using Release mode, as it will be faster than Debug mode.
        ii. Make sure that the axes are visible
        iii. Call out the keystrokes as you make them (e.g., "turn off first light", "increase x", etc).
        iv. Make sure that your project is running before recording.
    b. Script to follow:
        i. Announce your name and give a brief summary of what aspects of your project work and those that do not.
        ii. Display your source for `Raytracer::raytraceScene` and `Raytracer::traceIndividualRay` (if applicable). Walk through the code, describing the main parts of the code. Assume that you are explaining this to the instructor. This part should be 30 seconds.
        iii. Demonstrate lighting
            1. Basic lighting
                a. Make sure only one positional light is active. Make sure attenuation is turned off.
                b. Talk about your shapes and convince the viewer that the shading on your objects is accurate.
                c. Increase x one or more times, identify the changes to the objects' shading that are consistent with the light moving toward the positive x axis.
                d. Decrease x. Make quick observations.
                e. Repeat for y and z.
                f. Point to your open-ended cylinder and show that both the inside and outside of the cylinder is rendered properly.
            2. Demonstrate shadows
                a. Identify several properly rendered shadows. Move the light to verify the shadows are properly updated.
            3. Attenuation
                a. Turn on attenuation.
                b. Set the coefficients: quadratic = 0, linear = 1, and constant = 0.
                c. Place the light over an empty section of the plane and increase and decrease the light's y position to demonstrate the increasing

and decreasing of the light's intensity. Toggle attenuation on/off to show the difference.

4. Spotlight
    a. Turn on only the spotlight.
    b. Move the spotlight's position. Identify the parts of the scene that suggest your spotlight is correct.
    c. Move the spot light's direction. Identify the parts of the scene that suggest your spotlight is correct.
    d. Turn on positional light and show that both lights work together to jointly brighten parts of the scene. That is the spotlight and positional light add up.

iv. Demonstrate Objects
    1. Point out the required objects

v. Demonstrate Textures
    1. Point out the textured object.

vi. Demonstrate Transparency
    1. Identify the transparent object and indicate why it is correct.

vii. Demonstrate Resizing
    1. Resize the window and show that the sphere's still look spherical. When resizing, make sure that you show very narrow and very thin window sizes. These extreme window shapes will uncover any skewing that still exists.

viii. Demonstrate Additional Features
    1. Light tied to the world or tied to the camera. When demonstrating this, you must use two completely different vantage points. Also, you must move the light enough, and describe, to convince the viewer that the light is actually tied to the camera.
    2. Reflections
    3. Viewports
    4. Anti-aliasing