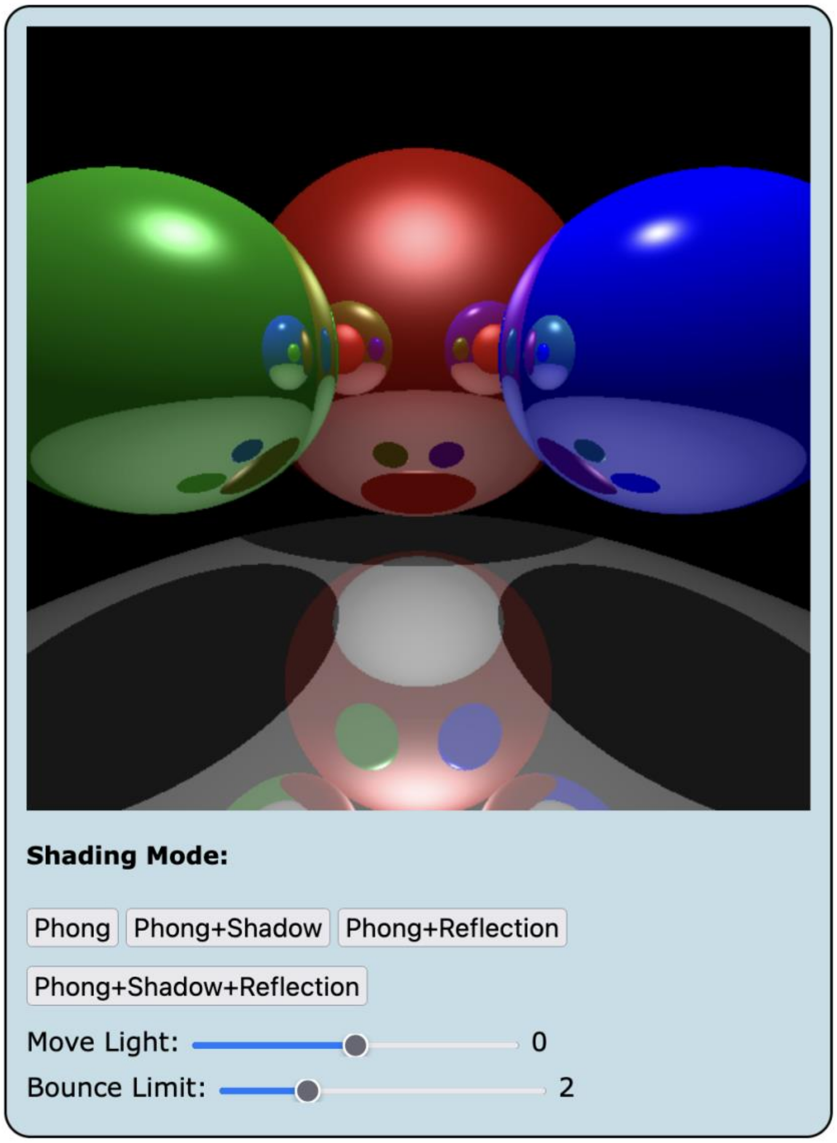
Assignment 5 – CS360A
Ray Tracing

Due Date: November 14, 2023, 11:59pm (No extension/Hard deadline)

Grade: 150 points (15% of the course grade)

In this assignment, you will create a ray tracer in GLSL shader. Your scene will only contain spheres for simplicity. In this assignment, you will implement ray-traced shadow and reflection. For reflection, your ray tracer will be able to perform multiple levels of reflection, as discussed in class, by allowing multiple bounces of the reflected ray. No additional library can be used to complete your assignment other than standard JavaScript functions. The glMatrix library is also not required for this assignment.

Here is the ray tracer app that you will create (See the attached video for more details):


Shading Mode:

Phong | Phong+Shadow | Phong+Reflection | Phong+Shadow+Reflection

Move Light: 0

Bounce Limit: 2

Pointers for your assignment:

1. Your application will allow changing the location of light through a slider. Based on the light location, shadows will change. Your scene will have just one light source.
2. Your application should also have a second slider that controls the bounce limit for the reflection ray. Set the maximum value of bounce to 5, i.e., your reflection ray will bounce maximum 5 times.
3. Your ray tracing application should have 4 buttons for selecting 4 different rendering modes (see the video and the above image for reference):
    1. Phong illumination
    2. Phong illumination + Shadow
    3. Phong illumination + Reflection
    4. Phong illumination + Shadow + Reflection
4. The scene will have just 4 spheres as shown above. You should have different shininess values for specular lighting for each sphere when computing the Phong illumination model.
5. There will be partial credits for implementing just the ray tracer, ray tracer + shadow, and ray tracer + reflection, ray tracer + shadow + reflection. So, if you are not able to add all the effects, you may follow the steps below to complete as many as you can.

Steps you may follow to complete the assignment:

- To build the scene, you do not need any transformation matrices. You can simply change the center and radius of the spheres to construct the scene. To affine transformations are needed.
- Your first focus should be to build a simple ray tracer without shadow and reflection. You should just render a single sphere and make sure your ray tracer is working correctly, and you see the sphere rendered.
- Here, you can simply assign a constant color to the fragments that belong to the sphere without any Phong illumination component. The output will look like a 2D circle.
- The next step is to implement the Phong illumination model so that the boring 2D circle will look like a proper 3D sphere considering light. Note that you can assume all the vectors that you need to implement the Phong illumination model are already in the same coordinate space, so you do not have to worry about coordinate transformation for normal vectors or any other vectors. You can simply apply the Phong illumination equations for ambient, diffuse, and specular lighting, and it should work just fine.
- The next step should be to add multiple spheres and construct the scene that is shown in the image above. This step should not require any modification to your base ray tracer except just adding multiple spheres into the scene.
- Next, you can implement the shadow algorithm by tracing the shadow ray from each intersection point. Be careful to calculate the ray origin and ray direction of the shadow ray. The direction should be normalized. If needed, you may have to use a bias correction to remove any self-shadow/ shadow artifacts.

- Finally, add reflection to your ray tracer. Trace the reflection ray and compute the reflection color for the fragments. Again, be careful to determine the reflection ray origin and direction. You can use the GLSL reflect () function to compute the reflation ray direction by providing appropriate arguments. You could just implement 1 level reflection first to see if it is working correctly and then extend it to implement the multi-level reflection by bouncing the reflection ray. For each bounce level, carefully compute the ray origin and ray direction. The ray origins will be the new intersection points, and the direction can be computed by using the reflect () function by passing the new normal vectors and the current incident rays to it.
- Consult the slides where I have given you fragments of code that you can use.
- The entire ray tracer will be implemented in fragment shader. In JavaScript, you only have to render a simple 2D quad that fills up the entire canvas in clip space. For this, you can define a quad with its corner points span across [-1,1] range. Check the slides for more details. We have discussed this in class.
- Map the light location to a slider.
- Map the bounce limit to a slider.

<u>How to submit?</u>
The **HelloIITK** portal will be set up for submission. There will be a time limit set. Please start early and finish it by the deadline. No extension will be provided. Your submission should ideally contain one main JavaScript and one HTML file. Zip everything into a single compressed file and upload it. Your code should run out-of-the-box on TA's computer without needing to do any modification. You can test it in both Chrome and Firefox before submitting it. Name your compressed submission file as "**Last-name_roll-number_Assignment5.zip**", and replace last-name, roll-number with your last name and roll number.

<u>Grading:</u>
We will grade your submitted version only. So DO NOT MISS THE DEADLINE, else you may get 0.

<u>Start early and talk to me if you have doubts/confusion. Good luck!</u>