Raytracer Project Documentation
Name: Bowen Luo

Final Project:

Purpose/Statement :

The purpose of this project was to create a ray tracer that renders a scene using a variety of techniques. In particular, this ray tracer should be able to demonstrate mirror and glossy reflection and refraction as well as soft shadows on the primitives in the scene. It should also feature texture and bump mapping and anti-aliasing through supersampling. Altogether, the ray tracer should be able to render an anti-aliased final scene consisting of a collection of spheres, each exhibiting varying degrees of both types of reflection and refraction. Some will have textures or geometric surface details and the scene will have soft shadows throughout.

Technical Outline :

For the first objective of mirror reflection, the purpose is to be able to treat certain surfaces as perfectly reflective and cast an additional ray in the mirror direction at each consecutive intersection. This should result in a recursive accumulation of additional color for the first intersection of the primary ray up to a certain depth.

For the second objective of refraction, the purpose is to be able to allow certain objects a transitivity and refractive index to allow the bending of rays as it passes the threshold between medium of different indices. This should model the behaviour of light as it passes through different densities resulting in a certain distortion when viewing through a transparent object. Snell's law is used for this objective.

For the third objective of glossy reflection, the purpose is to be able to treat certain surfaces as reflective, but not perfectly so, and cast multiple additional rays through a plane perpendicular to the perfectly reflected ray centred at the intersection point. This should result in a fuzzier reflection on the surface of the object. The sampling method over a square region of a plane introduced in the lectures is used for this objective.

For the fourth objective of glossy transmission, the purpose is to be able to have certain refractive medium behave as more translucent than transparent. A perturbation of the refracted ray by once again sampling multiple additional rays from a perpendicular plane will allow for a fuzzier refracted view through the translucent object.

For the fifth objective of texture mapping, the purpose is to be able to map coordinates on the surface of a primitive to coordinates of an image to allow for a non-uniform colouring. This is done through a projector function to first convert the intersection coordinate to a UV coordinate and then convert the UV coordinate into an image coordinate. For the sphere, the azimuthal and polar angles to the intersection point with respect to the x, y, and z axes will be used in the projector function. This intersection point must also be converted to the model frame of the sphere. Referencing the tutorial in [1]. To map to the image, multiplying u and v by the width and height rounded down should give the image coordinate. The textures of an Earth and a Checkerboard for a sphere come from [2] and [4] respectively.

For the sixth objective of bump mapping, the purpose is to be able to emulate geometric details on the smooth surfaces of primitives. This is done using a greyscale heightmap and perturbing the surface normal of the intersection point in its perpendicular plane by the heightmap's gradient at the texture coordinate which the intersection maps to. The mapping is done similarly to texture mapping, and the normal perturbation references the one given in the lectures. This should result in approximated surface geometry and corresponding lighting as the illumination model uses the new normal, displayed using different light angles. Heightmap used comes from [3].

For the seventh objective of supersampling, the purpose is to be able to reduce the aliasing or "jagginess" from the final rendered image. This is done by firing multiple rays through each pixel of the image plane and returning the average RGB values. Instead of firing one ray through the center of each pixel, each pixel is split into a grid and a ray is fired through each section of the grid. This should result in an image with smoother edges when compared to the rendered image without anti-aliasing.

For the eighth objective of soft shadows, the purpose is to be able to treat the point light sources of the scene as area lights and cast shadows that have soft edges. At each intersection point, instead of casting one shadow ray to each light source, we sample the area light for multiple random sources, splitting the light intensity evenly across them and firing a shadow ray to each sampled light point. The shadow density at the intersection point will then be determined by the percentage of sampled shadow ray intersections. The extrapolation of a point light into an area light will use the perpendicular plane to the initial shadow ray and the sampling of the plane will be done randomly, both formula introduced in the lectures.

For the ninth objective of the final scene, the purpose is to be able to display a rendered image that illustrates the previous eight objectives all together. That is, it should exhibit anti-aliasing and soft shadows on a collection of spheres. Some primitives in the scene should be textured or bump-mapped and exhibit varying degrees of mirror/glossy reflection and perfect/glossy refraction.

For the tenth objective of the UI, the purpose is to clearly present in a list each of the other 9 objectives in isolation in a PDF. That is, the PDF should show individual objects exhibiting mirror/glossy reflection/refraction, texture mapping, bump mapping, anti-aliasing, soft shadows, and display the final scene.

Additional Notes and Manual :

Since I was not able to finish objective 10 on Assignment 4, I do not have an official extra objective.

I was also not able to finish the extra objective of additional cone and cylinder primitives, and the objective was removed from the list above from the proposal.

There were not any new commands that were added to the input language.

The file doc/README provides all the necessary information needed to build and run the program, which hasn't changed from the default premake4/make combination. It also includes an overview of the expected behaviour and guide to the changes that were made to the constructors and classes for the materials and spheres. The file doc/UI is the completed tenth objective of a PDF containing descriptions for the extra images that were included in the submitted code for the purpose of showing each of the objectives in isolation.

Implementation :

The first concept implemented was that of supersampling anti-aliasing. This implementation was solely influenced by the discussion of grid-based supersampling from the lectures. Instead of firing one incident ray from the eye through the centres of each pixel of the image in world space as was already done for A4, each pixel is split into a 4x4 grid with vertical and horizontal "lines" at the 0.25, 0.5, 0.75 marks of the pixel from one edge to the other. A vector is then built through each of these intersections of grid for a total of 9 incident rays per pixel. The RGB colors of these rays are then averaged to form the final color of the pixel.

The second concept implemented was that of mirror reflection/refraction. This implementation was more of a re-factoring of the existing ray-object intersection code already done for A4. Moving it out of the incident ray firing loop into a recursive function to allow for repeated calls for each consecutive intersection after each reflection. The reflection of the ray vector itself is done through the glm::reflect method. Similarly, the refraction of the ray vector is done through the glm::refract method leveraging Snell's Law documented in [5].

The third concept implemented was the perturbation of secondary rays used for glossy reflection/refraction as well as soft shadows. This implementation was solely influenced by the discussion of sampling additional rays from a perpendicular plane from the lectures. At each intersection, depending on the properties of the surface, a secondary ray is either transmitted through the object, reflected across its surface, or shot towards each light source to find shadows. In the first two cases, the secondary refracted/reflected ray is crossed with the incident ray to get a vector perpendicular to the secondary ray, making up the 'x' axis of the sampling plane. This perpendicular ray is then crossed with the

secondary ray to form the 'y' axis of the plane. Adding the normalized secondary vector to the intersection point finally gives the center of the sampling plane. Two random floats between 0 and 0.99 multiplied by the normalized axis vectors of this 1x1 sampling plane determine a new secondary vector from the intersection point. This process is repeated to create multiple secondary rays whose returned RGB colors are averaged to form the final color at this surface. In the case of reflection, this color is further augmented by Phong illumination. In the case of a shadow ray, the creation and sampling of the generated perpendicular plane is done identically with the exception of the center of the plane being determined by the location of the point light source. Since the distance to the sampling plane can now be greater than 1 unit away, the plane is scaled to be greater than 1x1 as well. The darkness of the shadow is then proportional to the number of blocked soft shadow rays normalized to the number of rays fired.

The fourth concept implemented was the projection function mapping 3D sphere intersections to 2D coordinates on a texture/bump PNG image. This implementation was done referencing the algorithm in [1]. The intersection point on the sphere in world space is first multiplied by the inverse transformation matrix performed to the sphere to get coordinates in model space. Since the only transformations were translations, this results in just a vector subtraction. The azimuthal and polar angles are then constructed from the arctangent of the x and z coordinates and arc-cosine of the y coordinate and radius. Corresponding u and v coordinates on the unwrapped spherical surface are then constructed from these angles, normalized by 2 pi (horizontally 360 degrees) and pi (vertically 180 degrees) respectively and clamped between 0 and 1. The color assigned to the section of the sphere surface represented by this UV coordinate is retrieved from a corresponding pixel of a texture image. The pixel is then determined by multiplying the u and v clamped coordinates by the width and height of the image respectively, rounded down. Taking PNG image input is handled by the methods included in the external library file stb_image.hpp referenced by [6] and [7].

The fifth concept implemented was the perturbation of surface normals used in the Phong illumination model for bump mapping via a heightmap. This implementation was solely influenced by the discussion of bump mapping using heightmaps from the lectures. The projection and corresponder functions are used as before for texture mapping, except instead of just retrieving the single heightmap value at the coordinate, the gradients across the heightmap horizontally and vertically are calculated. At each coordinate, the greyscale value is subtracted from its neighbouring values below and to the right to get the height delta as we traverse down and across the PNG respectively. These delta values are then used to scale the corresponding normalized 'v' and 'u' axis vectors in world space to find how much the normal vector at this intersection is tilted in the vertical and horizontal directions respectively. Considering glm::cross uses right hand rule, the positive 'u' vector corresponding to traversing right across the heightmap is found by crossing the world positive y axis vector with the normal of the intersection in order. Similarly, by crossing this 'u' vector with the normal of the intersection again, the orthogonal 'v' vector will always point 'down' with a negative y component corresponding to traversing down across the heightmap. Both vectors will also follow the contour of the sphere due to the perpendicularity of the normal vector to the surface. With the 'u' and 'v' vectors properly scaled by the height delta, they are added to the normal vector, perturbing the Phong illumination model's lighting calculation.

Bibliography :

[1] "Texture-Mapping." Texture Mapping,
http://raytracerchallenge.com/bonus/texture-mapping.html.
(Texture mapping reference)

[2] https://www.solarsystemscope.com/textures/
(Earth and planet textures)

[3] http://www.edharriss.com/golfball.jpg
(Golfball heightmap)

[4] https://snappygoat.com/o/e344a3cbb7ab9084889a7d621610e7b34aacc0e9/
Seamless-checkerboard-pattern.jpg
(Checkerboard texture)

[5] \Glm::Refract." Refract - Opengl 4 Reference Pages, https://
registry.khronos.org/OpenGL-Refpages/gl4/html/refract.xhtml.
(glm::refract reference)

[6] \Get Pixel Color Values from Image.png - C++ Forum." Cplusplus.com, https://
cplusplus.com/forum/beginner/267364/.
(stb usage reference)

[7] \stb_image.h." GitHub, https://raw.githubusercontent.com/nothings/stb/master/
stb_image.h.
(stb_image header)

Objectives:

___ 1: Mirror Reflection

___ 2: Refraction

___ 3: Glossy Reflection

___ 4: Glossy Transmission

___ 5: Texture Mapping

___ 6: Bump Mapping

___ 7: Supersampling Anti-aliasing

___ 8: Soft Shadows

___ 9: Final Scene

__ 10: UI PDF