A5 Project Proposal
Title: Terrarium Ray Tracing

Name: Jenny Lei
Student ID: 20824513
User ID: j37lei

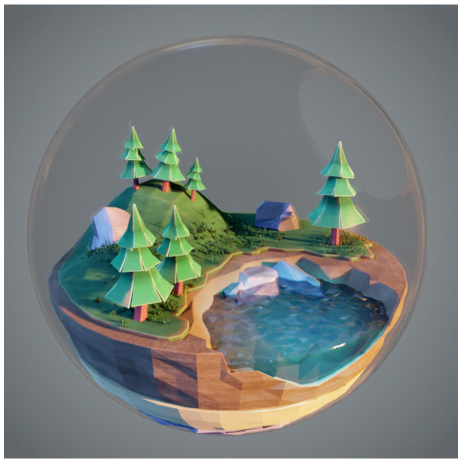
1 Purpose

To extend the ray tracer developed in assignment 4 to be able to handle more complex effects and render an interesting scene. New features include effects like reflections, refractions, texture/bump mapping, and caustics.

2 Statement

I want to render a scene containing a terrarium-like object. In the center of the scene will be a glass container with a miniature world inside of it. The world will have interesting features like water, plants, mountains, etc. The terrarium will be placed on top of a table with light hitting it from behind.

Here are some reference images



The left image is a bit closer in terms of what will be inside the glass container (i.e. plants, water, rocks, etc) but the one of the right contains some nicer environment/background details (i.e. table, light, and shadows).

There are several parts of this scene that makes it interesting and challenging. For example, the glass should have some refraction while water should have some reflection. There should also be caustics caused by the light and glass container which should show up on the table underneath. Texture and bump mapping can also make objects in the scene like the ground and rocks look more realistic.

3 Technical Outline

3.1 Reflection and Refraction

Reflection and refraction should be implemented to see through the glass container and to have more realistic water surfaces.

Reflection and refraction amount should be controlled by a constant separate from the Phong Illumination model constants.

For reflection, reflect the incoming ray about the surface normal and cast a new ray in that direction.

For refraction, determine the new refraction ray's direction using Snell's law and indices of refraction. Be careful of total internal refraction.

The Fresnel equations can be used to determine the distribution of reflected versus refracted light on an object.

Each material will need to store an additional reflection/refraction constant to indicate how much colour is diffuse and how much is reflected/refracted. It will also need to store it's index of refraction in order to implement refraction and to calculate the Fresnel formulas.

3.2 More Primitives

Extend the modelling language to support cylinders and cones. This will include adding new primitive classes and adding the corresponding ray-intersect-cylinder and ray-intersect-cone functions.

Code to a stable quadratic solver has already provided so what remains to be done is rearrange the quadratic surface equations to fit the input for the quadratic solvers.

3.3 Adaptive Anti-Aliasing

Add adaptive anti-aliasing to reduce the ”jaggies”.

For each pixel cast a ray to a initial grid. For each cell in the grid, if the colour variance exceeds some threshold, further subdivide the cell into another grid until either the variance in colour is below the threshold or after some number of subdivisions (to prevent the recursion stack from increasing too much).

Will need to add functionality to generate the rays, gather the colours, and determine when to further subdivide.

3.4 Texture Mapping

Determine the diffuse colour of a material based on a stochastic or deterministic function.

For each point of intersection on the surface of a primitive compute a set of texture coordinates. Using the texture coordinates and the object material's diffuse colour, determine a new diffuse colour.

Each primitive will need a way to index the texture image. For example, a sphere can use the elevation and azimuth of the point of intersection. For a cube, the face and where on the face the point of intersection is can be used to index the texture image.

This may include creating a new Texture Map class that can be attached to a primitive which will be used when determining the diffuse colour of an object.

3.5 Bump Mapping

Similar to Texture mapping except modulate the object's surface normal instead of it's diffuse colour.

This may include creating a new Bump Map class that can be attached to a primitive which will be used when determining the diffuse colour of an object.

3.6 Phong Shading

In OBJ files, include the normals of each vertex. When a ray intersects a face, determine the normal at the point of intersection using some form of interpolation (ex: Barycentric interpolation) of the face's vertex normals (instead of just using the face's normal).

3.7 Photon Mapping

To get caustic effects, add in Photon Mapping.

This is a two-part task: generating the photon map, and then using the map to determine the photon intensity at a point.

Before rendering the scene, cast rays from each light source in the scene. Trace the path of each ray and find where it hits any diffuse surface. If a diffuse object is hit by a light ray, store the point and other information like the object, it's normal, and the photon intensity, in some data structure for the photon map (ex: a kD-tree).

When generating the rays from a light source, imagine some polyhedron centered at the light. Cast some large number of rays from the light through each face of the polyhedron. A good polyhedron may be an icosahedron since more faces allow for a more even distribution of ray in the scene.

During ray tracing, when a ray hits an object, average out the photon intensity of the k closest points in the photon map to get the irradiance of that point. Be careful of noise that may be introduced during the k neighbours lookup since the kD-tree is space-based and not surface-based.

Some k nearest neighbour search will need to be implemented for Photon mapping.

4 Bibliography

A few articles / books on K-nearest neighbours / kD-trees / Photon Mapping:

- Hapala, M., & Havran, V. (2011). Review: KD-tree traversal algorithms for Ray Tracing. *Computer Graphics Forum*, *30*(1), 199–213. https://doi.org/10.1111/j.1467-8659.2010.01844.x
- Jensen, H. W. (2001). *Realistic image synthesis using photon mapping.* A. K. Peters, Ltd.
- Vanco, Brunnett, & Schreiber. (1999). A hashing strategy for efficient K-nearest neighbors computation. *Proceedings Computer Graphics International CGI-99.* https://doi.org/10.1109/cgi.1999.777924

Objectives:

Full User ID: j37lei **Student ID:** 20824513

___ 1: Add reflection

___ 2: Add refraction

___ 3: Add primitives for cylinders and cones

___ 4: Add adaptive anti-aliasing

___ 5: Add soft shadows

___ 6: Add texture mapping

___ 7: Add bump mapping

___ 8: Add Phong shading with Barycentric interpolation

___ 9: Add photon mapping (generating the photon map)

___ 10: Add photon mapping (rendering)

A4 extra objective: Supersampling