Algo 3: Ray

Introduction to Computer Graphics, Fall 2022

Due:

1 Camera operation

When implementing the ray tracing algorithm, it would be common that you will need to transform between different coordinate spaces. One transformation would be between the camera space and the world space.

We are going to use these term for the rest of our course.

- **Camera's view matrix**: The matrix that transform from world space to camera space.
- **Camera's transformation matrix**: The matrix that transform from camera to world space.

Now, given the position, look vector, and up vector, how would you use these info to compute the camera's view matrix? **You are not required to write any detailed math equation, just talk about what should be computed in order to get the camera's view matrix.**

2 Generating rays

[2 points]2.1 Given the following info of a pixel in an image, calculate the corresponding point on the view plane. You can assume that the pixel (0, 0) corresponds to the top left pixel of the image.

- W: image width
- H: image height
- i: pixel row index
- j: pixel column index
- k: depth of the view plane from camera position
- θ_h: horizontal view angle
- θ_v: vertical view angle

[2 points]2.2 Given the position of your camera p_{eye} and a point on the view plane p_{view} in the camera space, write down the equation for the ray you want to cast into the scene. Specify your ray in the format $p + td$, where p is the origin point and d is a normalized vector.

3 Implicit shape primitives

[3 points for the cone body(3.1), 2 points for the cone cap(3.2)]

For this problem, write down all your work and highlight the final answers either by bold text or by boxing them.

For the purpose of the Ray assignment, you can assume that **all of the shape primitives have the size of a unit cube with its center at the origin**. For example, if you are dealing with a sphere, then its radius will be 0.5 and its center is placed at the origin.

You have learned how to deal with sphere and cylinder using implicit functions in the lecture. So there is only one primitive left, the cone.

Write out both of the intersection equations for the cone body and the cone bottom cap. Use the format $p + td$ from the previous question to specify a ray.

Recall that the equation of a circle on the 2D XZ coordinate plane is $x^2 + z^2 = r^2$. Think of our canonical unit cone as an infinite number of "differential" circles in the XZ plane stacked on top of one another in the Y direction; the bottom-most circle has a radius of 0.5 and the topmost circle has a radius of 0. Then the equation of the unit cone is $x^2 + z^2 = f^2(y)$, where f is a function that linearly interpolates the radius of the differential circle from 0.5 at the base to 0 at the top.

The intersection points you compute are possible intersection points as they must fit certain criteria to be considered true intersection points (such as the $-0.5 \leq y \leq 0.5$ restriction for the body of the cylinder in the lecture notes). These possible points would therefore need to be further examined; however for this algo problem you are **NOT** required to list these restrictions. Do keep in mind these restrictions for the actual project!

Note that in your program you will need to find intersection points by finding a value for t. If you do not find an explicit formula for t (i.e. t = some value(s)) for both the cone and the cap then you will have a very hard time writing the program.

Finally the equations you write should not use vectors but should be functions of the individual components of the vectors. By reducing your equations after deriving them, you eliminate computations and thereby optimize your code before you even write it!

4 Illuminating samples

[2 points]4.1 In the first part of the ray project, you are **NOT** required to handle any lighting. Instead, you only need to use the normal at the intersection point as its color. Therefore, you have to compute the normal correctly! Assume you know the normal vector in object-space, $\vec{n}_{object}$. Give an equation for the normal vector in world-space, $\vec{n}_{world}$, using the object's modeling transformation M and $\vec{n}_{object}$.