Algorithm 6 Intersect

Introduction to Computer Graphics, Fall 2021

Solution Key

Instructions: You may discuss this assignment with other students, but you must abide by the rules stated in the collaboration policy (no notes from any discussions, your handin must be completely written up by you and contain only your own work, 15 minute delay between discussions and writing down any notes of your own) and write the logins of the student(s) you collaborated with next to each problem. Hand in the assignment on Gradescope as a PDF (please make sure the document is anonymous) no later than 6:00pm EST on the due date. **Late hand-ins are not accepted under any circumstances.**

1 Generating rays

[2 points] 1.1: Given a pixel with column and row indices i and j in an image with with $W$and height H, and a horizontal and vertical field of view θ_w and θ_h, what is the corresponding point, $p_{view}\,(p_x,\,p_y,\,p_z)$, on a view plane located at depth k? Assume that a pixel row index of 0 corresponds to the **top** of the screen.

Solution:

$$\mathbf{p}_{view} = \left(2k\tan(\theta_w/2)(\frac{i+0.5}{W} - 0.5), \quad 2k\tan(\theta_h/2)(0.5 - \frac{j+0.5}{H}), \quad -k\right)$$

The y-coordinate has $0.5 - \frac{j+0.5}{H}$ instead of $\frac{j+0.5}{H} - 0.5$ (which we saw in class) due to the y-origin being at the top of the screen instead of the bottom

[2 point] 1.2: Given your camera space eye-point p_{eye} and the point on the view plane p_{view} from the last question, give the equation for the ray you want to shoot into the scene. Specify your ray in the format $p + t\vec{d}$, where p is a point and $\vec{d}$ is a normalized vector.

Solution:

$$\begin{aligned} \vec{\mathbf{d}} &= \frac{\mathbf{p}_{view} - \mathbf{p}_{eye}}{\|\mathbf{p}_{view} - \mathbf{p}_{eye}\|} \\ r(t) &= \mathbf{p}_{eye} + t\vec{\mathbf{d}} \end{aligned}$$

Grading notes: 1/2 point was deducted if you forgot to normalize $p_{view} - p_{eye}$

2 Cone-Ray Intersection

Write out both of the cone-ray intersect equations in terms of t and solve for t. Remember, there are two equations: one for the body of the cone, and one for the bottom cap. For your cone, use the same dimensions that you did in Shapes. Use the defintion of a ray used above, i.e. $p + t\vec{d}$. To get you started you might want to define an intersection point as $(x,\,y,\,z) = \left\langle p_x + \vec{d}_x t,\, p_y + \vec{d}_y t,\, p_z + \vec{d}_z t\right\rangle$, where p is the eyepoint, and $\vec{d}$ is the direction of the ray we are shooting. Looking over the the derivation of the implicit equations for the cylinder in the Raytracing lecture might prove to be useful.

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 bottommost circle has a radius of 1/2 and the topmost circle has a radius of 0. Then the equation of the unit cone is $x^2 + z^2 = f\,(y)^2$, where f is a function that linearly interpolates the radius of the differential circle from 1/2 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 $-0.5 \le y \le 0.5$ restriction for the body of the cylinder in the lecture notes). These possible points would therefore need to be further examined; however for this algoproblem 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!

2.1 For the cone body [3 points]

Given *top*, which in our case equals 0.5, and slope m, which equals 2, since

$$\frac{\Delta y}{\Delta x} = \frac{1}{1/2} = 2.$$

What do we already know?

$$\begin{aligned} x^2 + z^2 &= radius^2 \\ y &= top - m \cdot radius \\ radius &= \frac{top - y}{m} \end{aligned}$$

Although it was not necessary for the algo you should also note that the bounds of the y-value are:

$$-top \leq y \leq top \tag{1}$$

Let our $(x, y, z) = (P_x + d_x t, P_y + d_y t, P_z + d_z t)$ based on the ray equation, so that we can express the point in terms of t. Then

$$(P_x + d_x t)^2 + (P_z + d_z t)^2 = \left(\frac{\frac{1}{2} - (P_y + d_y t)}{2}\right)^2 \tag{2}$$

$$(d_x^2 + d_z^2 - \tfrac{1}{4}d_y^2)t^2 + (2P_x d_x + 2P_z d_z - \tfrac{1}{2}P_y d_y + \tfrac{1}{4}d_y)t + \left(P_x^2 + P_z^2 - \tfrac{1}{4}P_y^2 + \tfrac{1}{4}P_y - \frac{1}{16}\right) = 0 \tag{3}$$

$$\begin{aligned} Let\ A &= (d_x^2 + d_z^2 - \tfrac{1}{4}d_y^2) \\ Let\ B &= (2P_x d_x + 2P_z d_z - \tfrac{1}{2}P_y d_y + \tfrac{1}{4}d_y) \\ Let\ C &= \left(P_x^2 + P_z^2 - \tfrac{1}{4}P_y^2 + \tfrac{1}{4}P_y - \tfrac{1}{16}\right) \end{aligned}$$

$$At^2 + Bt + C = 0 \tag{4}$$

$$t = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A} \tag{5}$$

This is not complete because we need to check the discriminant. The above equation for t will give rational values if and only if $B^2 - 4AC \geq 0$.

2.2 For the cone cap [2 points]

The easiest way to approach this problem is to simply intersect with a plane and then check that the intersection point lies within the circle of radius 0.5 centered at $x = 0$ and $z = 0$. The bottom cap lies along the $y = -0.5$ plane.

$$\begin{aligned} y &= -0.5 && (6) \\ (P_y + d_y t) &= -0.5 && (7) \\ t &= \frac{-0.5 - P_y}{d_y} && (8) \end{aligned}$$

The intersection point x, z will lie within the bounds of the cap if $x^2 + z^2 \leq (0.5)^2$. In general, to find t, where a plane is specified with a point (x_0, y_0, z_0), and a normal $\vec{N}$:

$$\begin{aligned} N_x x + N_y y + N_z z - (N_x x_0 + N_y y_0 + N_z z_0) &= 0 \\ N_x(P_x + d_x t) + N_y(P_y + d_y)t + N_z(P_z + d_z)t - (N_x x_0 + N_y y_0 + N_z z_0) &= 0 \\ \frac{(N_x x_0 + N_y y_0 + N_z z_0) - (N_x P_x + N_y P_y + N_z P_z)}{N_x d_x + N_y d_y + N_z d_z} &= t \end{aligned}$$

Note that this general formula can be applied to *all* planes, such as those of cubes and cylinder caps.

3 Illuminating Samples

[**2 points**] **3.1:** When you are attempting to illuminate a transformed object, you will need to know that object's normal vector in world-space. Assume you know the normal vector in object-space, $\vec{\mathbf{n}}_{object}$. Give an equation for the normal vector in world-space, $\vec{\mathbf{n}}_{world}$, using the object's modeling transformation $\mathbf{M}$ and $\vec{\mathbf{n}}_{object}$.

Solution: $\vec{\mathbf{n}}_{world} = (\mathbf{M}_{3\times3}^{-1})^T * \vec{\mathbf{n}}_{object}$, where $\mathbf{M}_{3\times3}$ is the upper 3 by 3 transformation matrix (we don't care about the translation part because vectors cannot be translated).

[**1 point**] **3.2:** In the lighting equation, what does $\vec{\mathbf{n}} \cdot \vec{\mathbf{L}}$ represent, i.e. what trigonometric function is equivalent to it? What is its purpose?

Solution: If θ is the angle between $\vec{\mathbf{n}}$ and $\vec{\mathbf{L}}$, then $\vec{\mathbf{n}} \cdot \vec{\mathbf{L}}$ represents $cos(\theta)$. It indicates to what extent a surface with normal $\vec{\mathbf{n}}$ is facing away from a light with direction $\vec{\mathbf{L}}$, and therefore how much the light is illuminating the surface.

4 Finally...

[**1 point**] **4.1:** What is the difference between lighting (illumination) and shading?

Solution: Lighting, or illumination, is the evaluation of the lighting model (in other words, an approximation to the rendering equation) at a sample point. Shading is the interpolation of computed lighting values at intermediate points (an approximation of lighting).