# Algorithm 3: Intersect

October 5, 2021

Introduction to Computer Graphics, Fall 2021

*Due Date*: 6:00 PM EST on Oct 8, 2021.

**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 screen-space coordinates $x$ and $y$, and the width and height of the screen $x_{max}$ and $y_{max}$, what is the corresponding point, $p_{\text{film}}\,(p_x,\, p_y,\, p_z)$, on the normalized film plane? Use the far clip plane as your film plane, and remember that a pixel y-value of 0 corresponds to the top of the screen.

**[2 points] 1.2:** In lecture, you learned how to transform a point from world space to screen space by applying $M_{\text{translate}}$ and $M_{\text{rotate}}$. In Intersect and Ray, you need to transform $p_{\text{film}}$ on the normalized *film* plane into an untransformed world-space point, $p_{\text{world}}$ using matrices like these. In terms of $M_{\text{persp}}$, $M_{\text{scale}}$, $M_{\text{rot}}$, and $M_{\text{trans}}$, (or their inverses) what is $M_{\text{film\_to\_world}}$, where $p_{\text{world}} = M_{\text{film\_to\_world}} \times p_{\text{film}}$ ? You may not need all 4 matrices. Please also explain why this transformation is needed for the Intersect assignment.

**[2 point] 1.3:** Given your world space eye-point $p_{eye}$ and the world point on the film plane $p_{world}$ give the equation for the world-space 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.

## 2 Cone-Ray Intersection

**[3 points for the cone body(2.1), 2 points for the cone cap(2.2)]**

***Special Instructions:*** *For this problem, show all your work and circle, box, or bold your final answers.*

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 definition 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 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 bottom-most 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 $-0.5 \leq y \leq 0.5$ 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!

## 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 $\vec{\mathbf{n}}_{object}$.

**[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?

## 4 Finally...

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