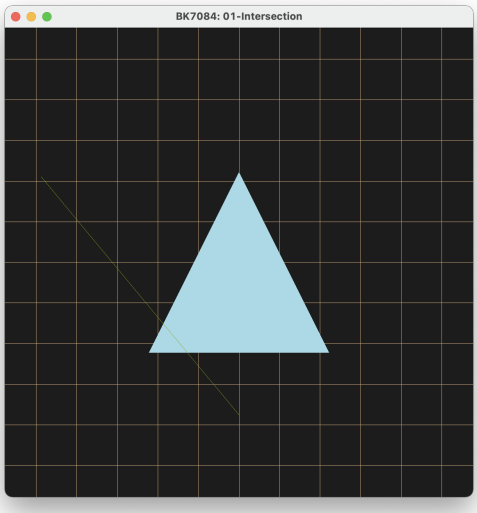
Assignment 1: Intersection

BK7084 Computational Simulations

1 Description

Your first assignment is to write a program for a ray-triangle intersection. This is one of the core tests for many light simulation algorithms, especially ray-tracing. You can think of the ray as being a ray of light and the triangle is part of some surface. You want to know whether the light ray intersects with the surface (i.e. the triangle).

Figure 1: Starting Situation



Most of the framework is already implemented for you. When you first run the exercise you should see a blue triangle with a moving ray (in green). Your task is to complete the `intersection.py` file. Once it is correctly implemented, the triangle will change its color to red when the ray intersects it.

There are three subtasks, all to be completed in `intersect_ray_triangle`:

1. Build matrix A and vector b to create a linear system (see next page).
2. Solve the linear system using NumPy's solve function: `np.linalg.solve(A, b)`.
3. Implement the intersection conditions based on the solution vecto (see next page).

1.1 Ray-Triangle intersection

In lecture 1 we saw the following equation:

$$\mathbf{r} + t\mathbf{d} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}), \tag{1}$$

where the left side defines a point on a ray and the right side a point on the plane of a triangle. More specifically:

- The ray is defined by its origin $\mathbf{r}$ and a direction $\mathbf{d}$. The parameter t defines any point along the ray by scaling the direction vector.
- The triangle is defined by three vertices $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ and the parameter β and γ combines the three vertices to define a point on the triangle's plane.
- When the following conditions for β and γ are satisfied, the point is inside the triangle:

$$\begin{aligned} \beta &\geq 0, \\ \gamma &\geq 0, \\ \beta + \gamma &\leq 1. \end{aligned}$$

We also saw that Equation 1 is actually not *one* equation, but three: one equation for each axis. We can thus rewrite Equation 1 as:

$$\begin{aligned} r_x + td_x &= a_x + \beta(b_x - a_x) + \gamma(c_x - a_x), \\ r_y + td_y &= a_y + \beta(b_y - a_y) + \gamma(c_y - a_y), \\ r_z + td_z &= a_z + \beta(b_z - a_z) + \gamma(c_z - a_z). \end{aligned}$$

The linear system above contains three unknowns: β, γ and t. We have a ray and a triangle, and we want to find a point defined on the plane by the parameters β, γ and the same point defined by the parameter t from the ray. In other words, we want to find β, γ and t that will make the left side of the equations equal the right side. When this happens, we have found a point that is on the ray and at the same time on the plane of the triangle.

Linear systems are commonly expressed in a matrix notation as it makes any computation much simpler. The above equations can then be written as:

$$\begin{bmatrix} a_x - b_x & a_x - c_x & dx \\ a_y - b_y & a_y - c_y & dy \\ a_z - b_z & a_z - c_z & dz \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} a_x - r_x \\ a_y - r_y \\ a_z - r_z \end{bmatrix}$$

We encourage you to check this. Real understanding comes from working through problems. Try to go from the matrix notation back to the three equations.

We often write the combination of matrices and vectors as $\mathbf{Ax} = \mathbf{b}$. This is called a linear system. Here, we know the elements of $\mathbf{A}$ and $\mathbf{b}$, and we want to find the solution vector $\mathbf{x}$. For our specific ray-triangle intersection $\mathbf{x} = [\beta, \gamma, t]^\mathsf{T}$. One way to solve this system is by sweeping the matrix (Gaussian elimination). We will take a shortcut to find a solution by using NumPy's solver for linear system. It's used as follows: `x = np.linalg.solve(A, b)`.