

# Ray Tracing & Ray Casting

Realistic Graphics Inspired by Nature

---

Ashrafur Rahman

Adjunct Lecturer

Department of Computer Science and Engineering  
Bangladesh University of Engineering and Technology (BUET)

Motivation

The Story of Light

Ray Casting: Foundation

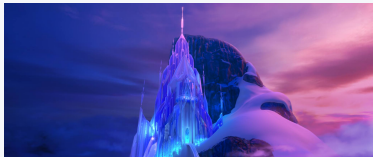
Ray Generation

# Motivation

---

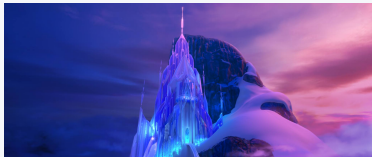
# Why Learn This?

# Why Learn This?

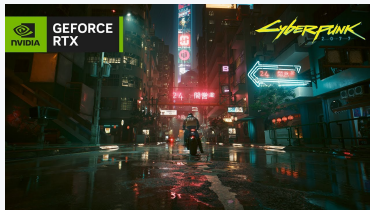


Elsa's Castle in Frozen

# Why Learn This?



Elsa's Castle in Frozen



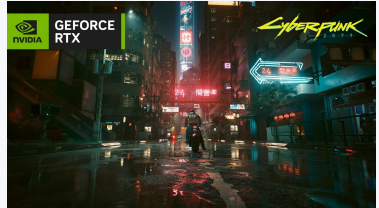
Cyberpunk 2077 with RTX

- **Realistic graphics** of your favourite animated movies are the result of ground-breaking work in Ray Tracing by studios like Disney, Pixar, and DreamWorks. Do you know these films take years to render? 30 hours per frame!

# Why Learn This?



Elsa's Castle in Frozen



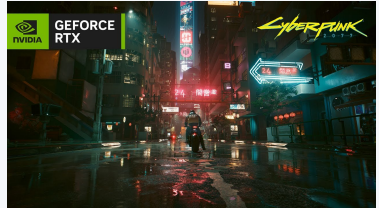
Cyberpunk 2077 with RTX

- **Realistic graphics** of your favourite animated movies are the result of ground-breaking work in Ray Tracing by studios like Disney, Pixar, and DreamWorks. Do you know these films take years to render? 30 hours per frame!
- Lately, **RTX** is all the rage in gaming. New titles boast ray-tracing effects in real-time, not 30 hours per frame!

# Why Learn This?



Elsa's Castle in Frozen



Cyberpunk 2077 with RTX

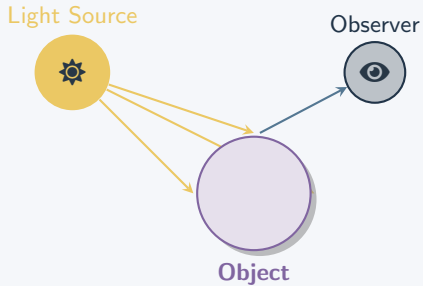
- **Realistic graphics** of your favourite animated movies are the result of ground-breaking work in Ray Tracing by studios like Disney, Pixar, and DreamWorks. Do you know these films take years to render? 30 hours per frame!
- Lately, **RTX** is all the rage in gaming. New titles boast ray-tracing effects in real-time, not 30 hours per frame!
- It's fun! You will know when you create your first ray-traced image!



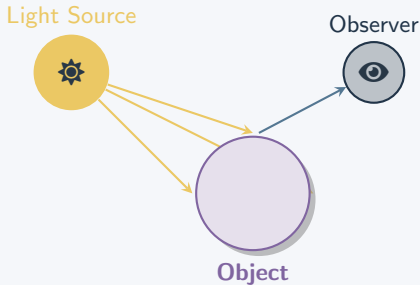
# The Story of Light

---

# How Do We See?



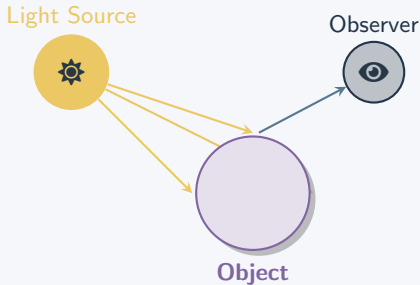
# How Do We See?



## Natural Process

1. Light travels from source
2. Light hits objects
3. Light bounces to our eyes
4. Our brain interprets the signal

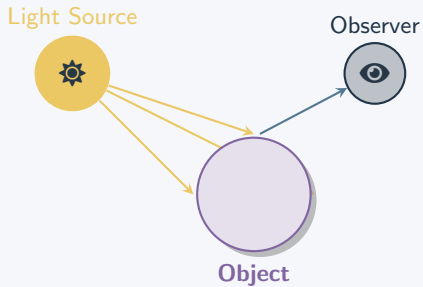
# How Do We See?



## Physical Process

1. Photon is emitted from source
2. Photon hits objects
3. Part of the photon is reflected or absorbed
4. The reflected photons reach our eyes
5. The rods and cones in our retina detect the photons
6. Our brain interprets the signal
7. **Colour:** The wavelength of the photons
8. **Brightness:** The number of photons

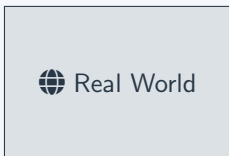
# How Do We See?



Question: How do we simulate this?

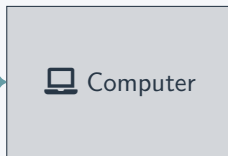
# The Computer Graphics Challenge

**Infinite Complexity**



Simulate

**Finite Pixels**



## Challenges:

- Infinite light rays/photons
- Complex physics
- High computational cost

# Ray Casting: Foundation

---

# The Key Insight

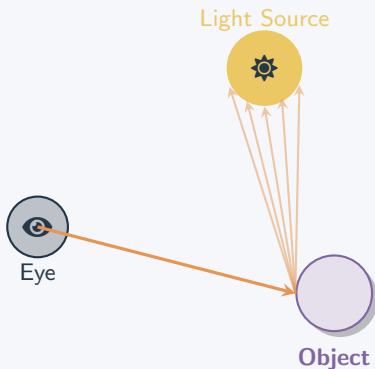
## 1. Reverse Engineering

Instead of following light rays from light sources —

**Let's trace backwards!**

**Shoot rays from the eye,**  
find where it hits and find out  
how much light reaches there.

This is the opposite of what happens in reality. **Why does this work?**





# The Key Insight

## 1. Reverse Engineering

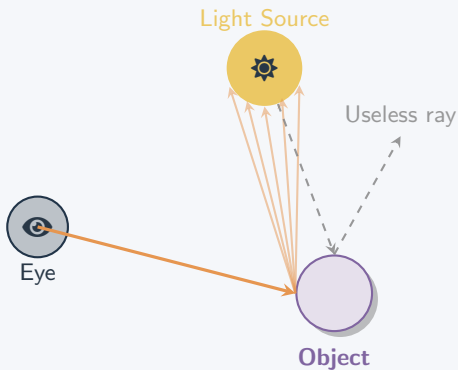
Instead of following light rays from light sources —

**Let's trace backwards!**

**Shoot rays from the eye,**  
find where it hits and find out  
how much light reaches there.

This is the opposite of what happens in reality. **Why does this work?**

- Most light never reaches our eyes



# The Key Insight

## 1. Reverse Engineering

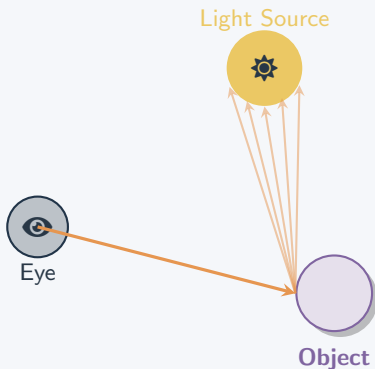
Instead of following light rays from light sources —

**Let's trace backwards!**

**Shoot rays from the eye,**  
find where it hits and find out  
how much light reaches there.

This is the opposite of what happens in reality. **Why does this work?**

- Most light never reaches our eyes
- Only trace rays that matter
- Much more efficient!



# From Infinite Rays to Finite Pixels

## 2. Cutting Costs

Instead of tracing infinite rays —

**Trace one ray per pixel.**

This comes with little tradeoff, because:

# From Infinite Rays to Finite Pixels

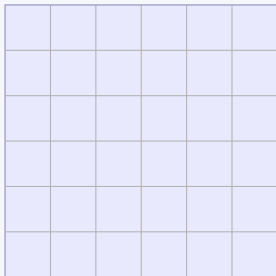
## 2. Cutting Costs

Instead of tracing infinite rays —

**Trace one ray per pixel.**

This comes with little tradeoff, because:

- An image is just a grid of pixels



# From Infinite Rays to Finite Pixels

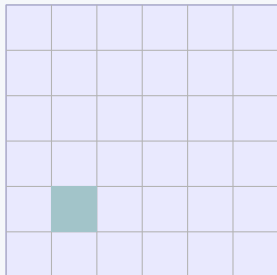
## 2. Cutting Costs

Instead of tracing infinite rays —

**Trace one ray per pixel.**

This comes with little tradeoff, because:

- An image is just a grid of pixels
- Each pixel can only be of one color



# From Infinite Rays to Finite Pixels

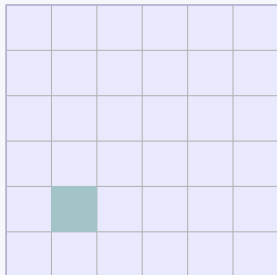
## 2. Cutting Costs

Instead of tracing infinite rays —

**Trace one ray per pixel.**

This comes with little tradeoff, because:

- An image is just a grid of pixels
- Each pixel can only be of one color
- In the end, we just need to know the *color of each pixel*



# From Infinite Rays to Finite Pixels

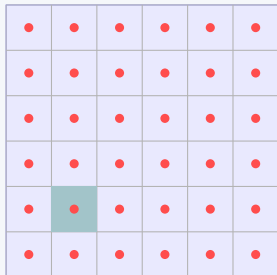
## 2. Cutting Costs

Instead of tracing infinite rays —  
**Trace one ray per pixel.**

This comes with little tradeoff, because:

- An image is just a grid of pixels
- Each pixel can only be of one color
- In the end, we just need to know the *color of each pixel*
- Hence, one ray from the mid-point of each pixel should be a good approximation\*

\* We will discuss more advanced techniques later that improve quality



# The Full Picture

Light Source



Eye

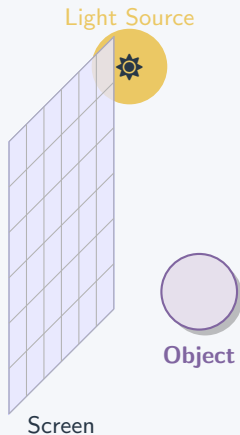


Object

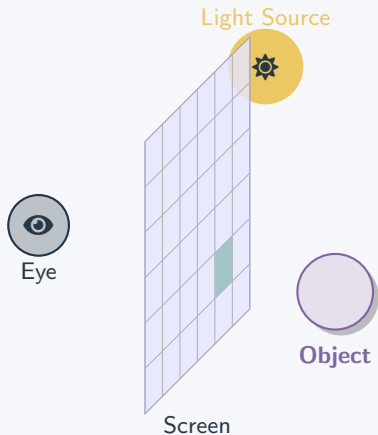
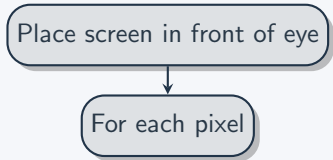


# The Full Picture

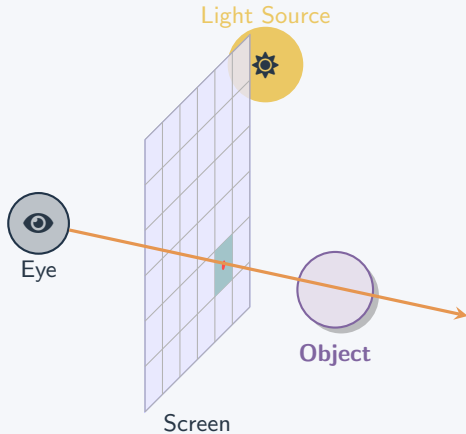
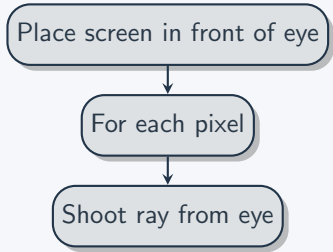
Place screen in front of eye



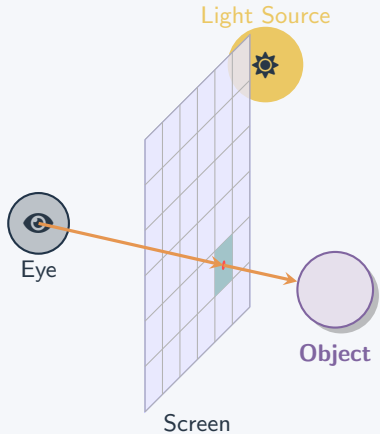
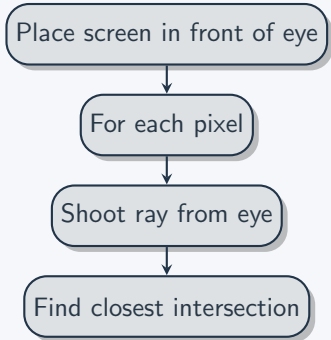
# The Full Picture



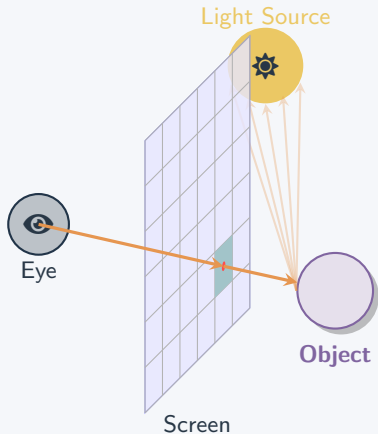
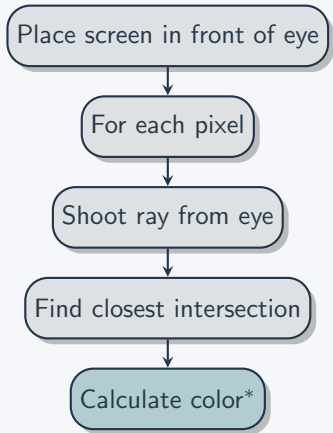
# The Full Picture



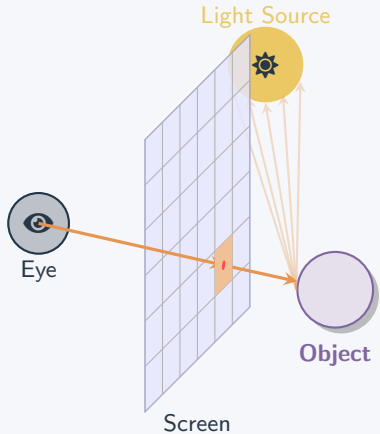
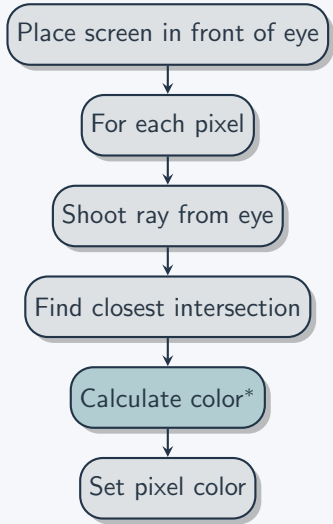
# The Full Picture



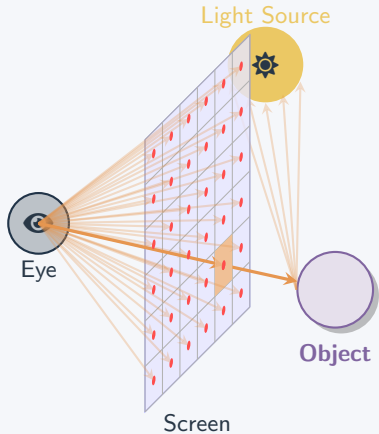
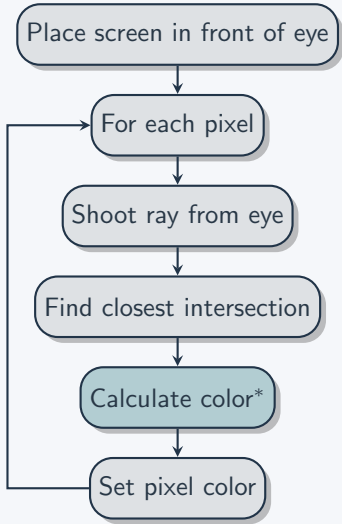
# The Full Picture



# The Full Picture



# The Full Picture



# Ray Generation

---



# What is a Ray?

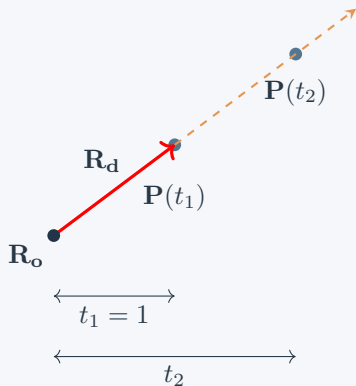
## Ray Representation

A ray is defined by:

$$\mathbf{P}(t) = \mathbf{R}_o + t \cdot \mathbf{R}_d \quad (1)$$

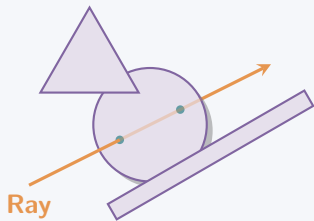
where:

- $\mathbf{R}_o$  = Origin point
- $\mathbf{R}_d$  = Direction vector
- $t$  = Parameter ( $t \geq 0$ )



Check out here on desmos.

## Finding Intersections



### Key Objects:

- Planes
- Spheres
- Triangles
- General Quadrics

**Challenge:** Find the **closest** intersection efficiently!

# 3D Plane Representation

## Plane Definition

A plane is defined by:

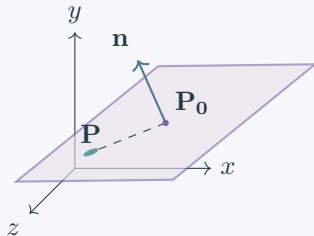
- Point  $\mathbf{P}_0 = (x_0, y_0, z_0)$  on plane
- Normal vector  $\mathbf{n} = (A, B, C)$

**Implicit equation:**

$$\mathbf{n} \cdot (\mathbf{P} - \mathbf{P}_0) = 0$$

$$\mathbf{n} \cdot \mathbf{P} + D = 0 \text{ where } D = -\mathbf{n} \cdot \mathbf{P}_0$$

$$Ax + By + Cz + D = 0$$



# 3D Plane Representation

## Plane Definition

A plane is defined by:

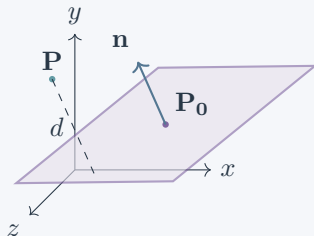
- Point  $\mathbf{P}_0 = (x_0, y_0, z_0)$  on plane
- Normal vector  $\mathbf{n} = (A, B, C)$

**Implicit equation:**

$$\mathbf{n} \cdot (\mathbf{P} - \mathbf{P}_0) = 0$$

$$\mathbf{n} \cdot \mathbf{P} + D = 0 \text{ where } D = -\mathbf{n} \cdot \mathbf{P}_0$$

$$Ax + By + Cz + D = 0$$



## Point-Plane Distance

If  $\mathbf{n}$  is normalized:  $d = \mathbf{n} \cdot \mathbf{P} + D = \mathbf{n} \cdot (\mathbf{P} - \mathbf{P}_0)$

**Signed distance:**  $d > 0$  (front),  $d < 0$  (back),  $d = 0$  (on plane)

# Ray-Plane Intersection

## Intersection Method

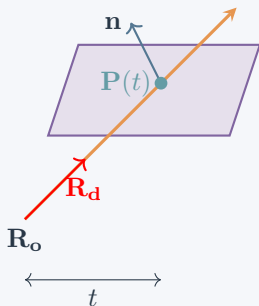
**Step 1:** Substitute ray into equation

$$\mathbf{n} \cdot (\mathbf{R}_o + t\mathbf{R}_d) + D = 0$$

$$\mathbf{n} \cdot \mathbf{R}_o + t(\mathbf{n} \cdot \mathbf{R}_d) + D = 0$$

**Step 2:** Solve for parameter  $t$

$$t = -\frac{D + \mathbf{n} \cdot \mathbf{R}_o}{\mathbf{n} \cdot \mathbf{R}_d}$$



# Ray-Plane Intersection

## Intersection Method

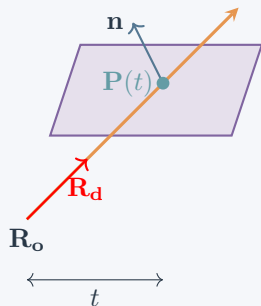
**Step 1:** Substitute ray into equation

$$\mathbf{n} \cdot (\mathbf{R}_o + t\mathbf{R}_d) + D = 0$$

$$\mathbf{n} \cdot \mathbf{R}_o + t(\mathbf{n} \cdot \mathbf{R}_d) + D = 0$$

**Step 2:** Solve for parameter  $t$

$$t = -\frac{D + \mathbf{n} \cdot \mathbf{R}_o}{\mathbf{n} \cdot \mathbf{R}_d}$$



## Cases

- If  $\mathbf{n} \cdot \mathbf{R}_d = 0$ : Ray parallel to plane (0 or infinite)
- If  $\mathbf{n} \cdot \mathbf{R}_d < 0$ : Ray hits front face
- If  $\mathbf{n} \cdot \mathbf{R}_d > 0$ : Ray hits back face

# Additional Checks

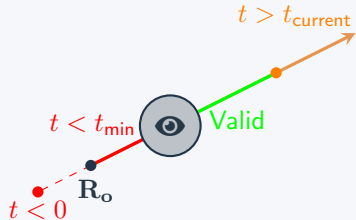
## Validation Rules

After computing  $t$ , verify:

1. **Behind check:**  $t > t_{\min}$
2. **Closest check:**  $t < t_{\text{current}}$
3. **Valid range:**  $t \geq 0$

Where:

- $t_{\min}$ : Minimum ray distance (not behind eye)
- $t_{\text{current}}$ : Distance to closest intersection so far



# Ray-Triangle Intersection Overview

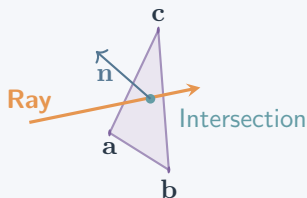
## Two Main Approaches

### Method 1: Two-Step Process

1. Ray-plane intersection
2. Inside/outside triangle test

### Method 2: Direct Barycentric

1. Set up  $3 \times 3$  linear system
2. Solve for  $t$ ,  $\beta$ ,  $\gamma$  simultaneously





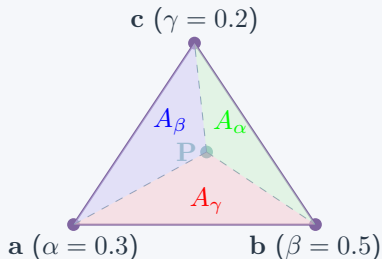
# What Are Barycentric Coordinates?

## Barycentric Definition

Any point  $P$  in the triangle's plane:

$$\mathbf{P}(\alpha, \beta, \gamma) = \alpha \mathbf{a} + \beta \mathbf{b} + \gamma \mathbf{c}$$

where:  $\alpha + \beta + \gamma = 1$



Check out the Desmos demo.

# What Are Barycentric Coordinates?

## Barycentric Definition

Any point  $P$  in the triangle's plane:

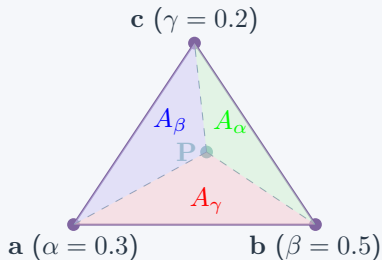
$$\mathbf{P}(\alpha, \beta, \gamma) = \alpha \mathbf{a} + \beta \mathbf{b} + \gamma \mathbf{c}$$

where:  $\alpha + \beta + \gamma = 1$

## Physical Interpretation:

- $\alpha, \beta, \gamma$  are *weights*
- $P$  is the *center of mass*
- Also called *barycenter*

Check out the Desmos demo.



# What Are Barycentric Coordinates?

## Barycentric Definition

Any point P in the triangle's plane:

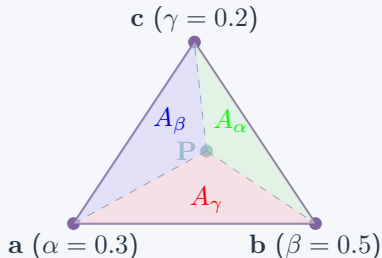
$$\mathbf{P}(\alpha, \beta, \gamma) = \alpha \mathbf{a} + \beta \mathbf{b} + \gamma \mathbf{c}$$

where:  $\alpha + \beta + \gamma = 1$

## Physical Interpretation:

- $\alpha, \beta, \gamma$  are *weights*
- P is the *center of mass*
- Also called *barycenter*

Check out the Desmos demo.



## Area Relationship

$$\alpha = \frac{A_\alpha}{A_{total}}, \quad \beta = \frac{A_\beta}{A_{total}},$$
$$\gamma = \frac{A_\gamma}{A_{total}}$$

# Barycentric Coordinates: Inside vs Outside

## Triangle Interior Test

Point P is **inside** triangle if:

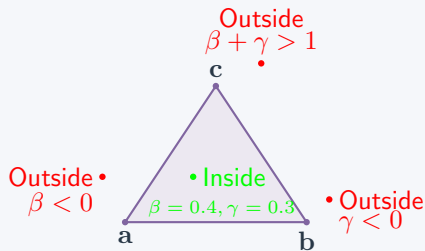
$$\alpha, \beta, \gamma \geq 0$$

Since  $\alpha + \beta + \gamma = 1$ , we can rewrite as:

$$\beta \geq 0$$

$$\gamma \geq 0$$

$$\beta + \gamma \leq 1$$



# Barycentric Coordinates: Inside vs Outside

## Triangle Interior Test

Point P is **inside** triangle if:

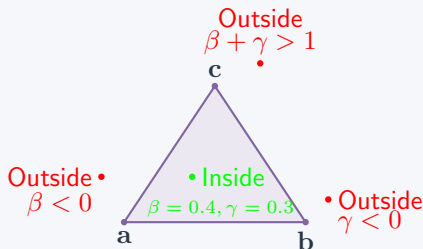
$$\alpha, \beta, \gamma \geq 0$$

Since  $\alpha + \beta + \gamma = 1$ , we can rewrite as:

$$\beta \geq 0$$

$$\gamma \geq 0$$

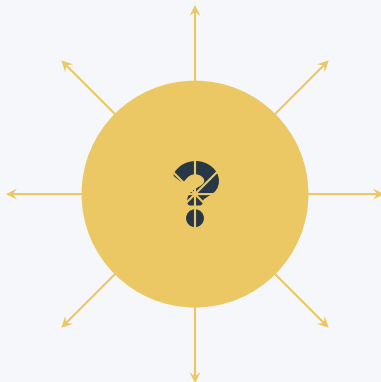
$$\beta + \gamma \leq 1$$








## Insight

Barycentric coordinates doesn't just tell us if a point is inside a triangle, but also it's position with respect to other vertices.

# Questions?



## References & Further Reading

-  Peter Shirley and Steve Marschner et al. *Fundamentals of Computer Graphics (4th Edition)*. CRC Press, 2016.  
Available as PDF
-  Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation (4th Edition)*. Morgan Kaufmann, 2023.  
Available online
-  Peter Shirley. *Ray Tracing in One Weekend*. Self-published, 2016–2020.  
Project Website
-  MIT OpenCourseWare: 6.837 Computer Graphics.  
[ocw.mit.edu/6-837](https://ocw.mit.edu/6-837)
-  Scratchapixel: Learn Computer Graphics Programming.  
[scratchapixel.com](https://scratchapixel.com)