Ray Tracing & Ray Casting

Realistic Graphics Inpsired by Nature

Ashrafur Rahman

Adjunct Lecturer

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

Index

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?

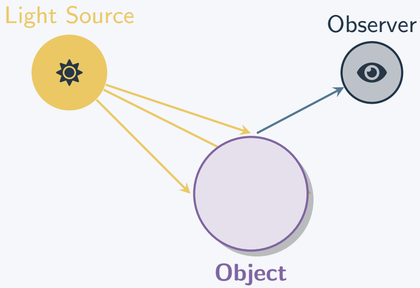
Elsa's Castle in Frozen

Cyberpunk 2077 with RTX

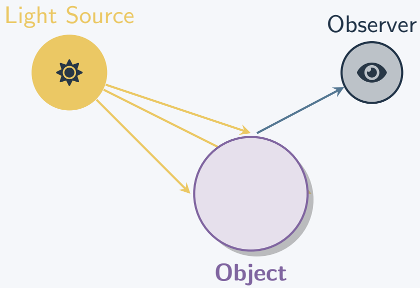
- **Realistic graphics** of your favourite animated movies are the result of ground-breaking work in Ray Tracing by studios like Disney, Pixar, and DreamWorks. Do you know these films take years to render? 30 hours per frame!
- Lately, **RTX** is all the rage in gaming. New titles boast ray-tracing effects 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?



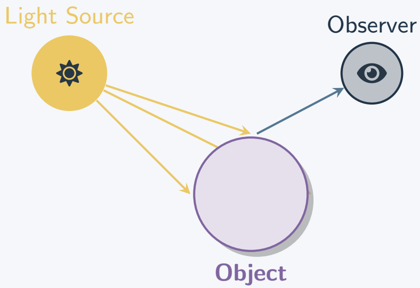
How Do We See?



Natural Process

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

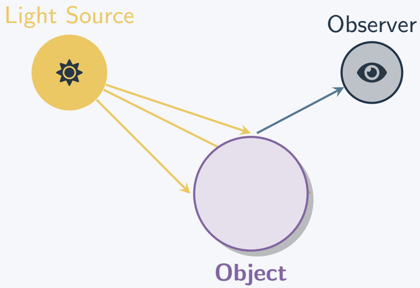
How Do We See?



Physical Process

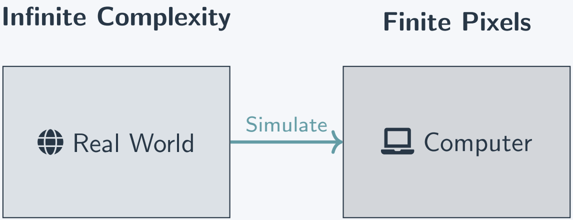
1. Photon is emitted from source
2. Photon hits objects
3. Part of the photon is reflected or absorbed
4. The reflected photons reach our eyes
5. The rods and cones in our retina detect the photons
6. Our brain interprets the signal
7. **Colour**: The wavelength of the photons
8. **Brightness**: The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

How Do We See?



Question: How do we simulate this?

The Computer Graphics Challenge



Challenges:

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

Ray Casting: Foundation

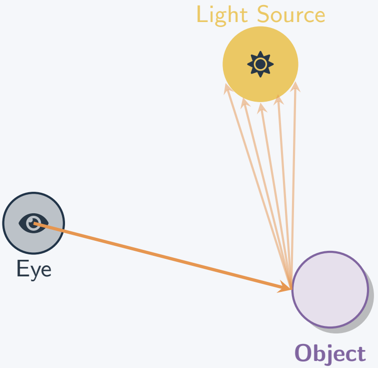
The Key Insight

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?**

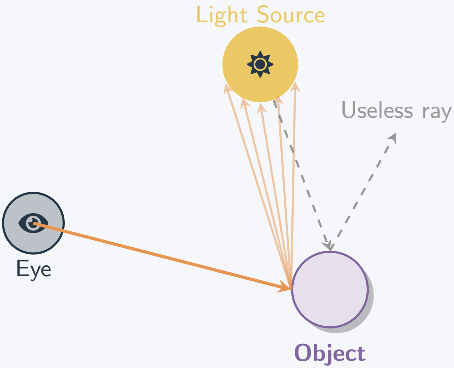


The Key Insight

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?**

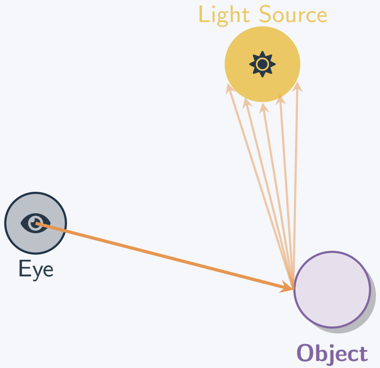
- Most light never reaches our eyes

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

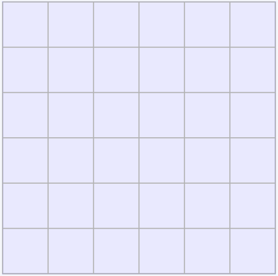
2. Cutting Costs

Instead of tracing infinite rays —

Trace one ray per pixel.

This comes with little tradeoff, because:

- An image is just a grid of pixels

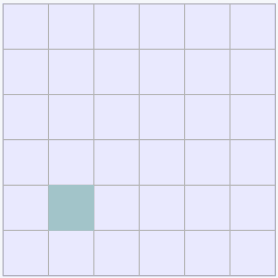


From Infinite Rays to Finite Pixels

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

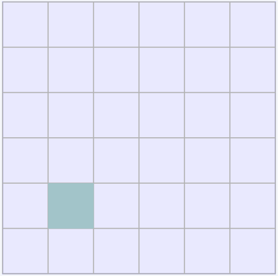
2. Cutting Costs

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

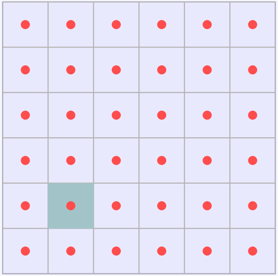
Instead of tracing infinite rays —

Trace one ray per pixel.

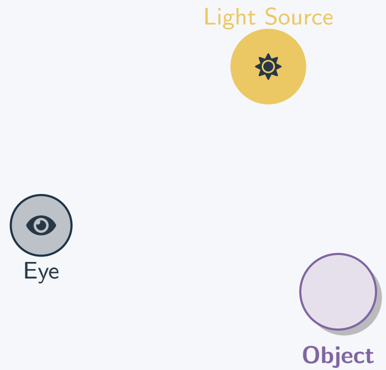
This comes with little tradeoff, because:

- 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

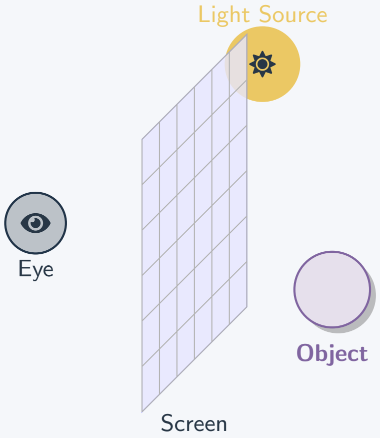


The Full Picture



The Full Picture

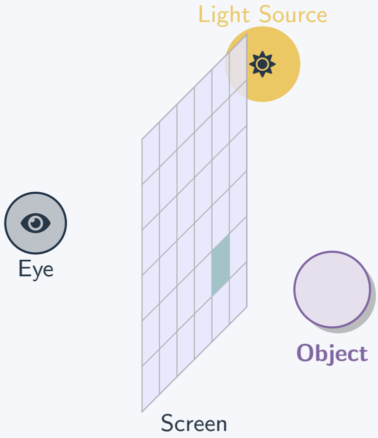
Place screen in front of eye



The Full Picture

Place screen in front of eye

For each pixel



The Full Picture

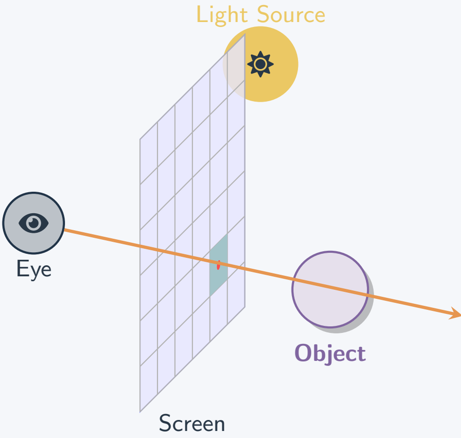
Place screen in front of eye

↓

For each pixel

↓

Shoot ray from eye



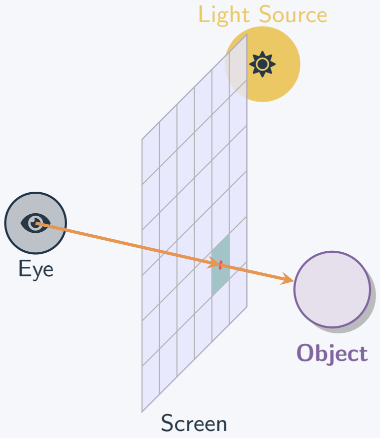
The Full Picture

Place screen in front of eye

For each pixel

Shoot ray from eye

Find closest intersection



The Full Picture

Place screen in front of eye

↓

For each pixel

↓

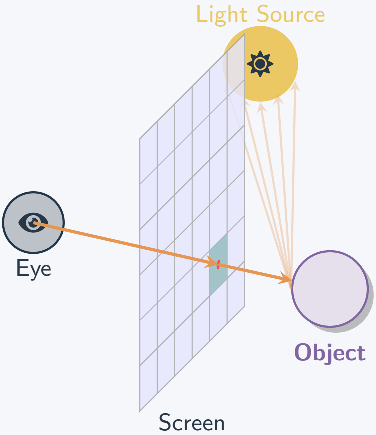
Shoot ray from eye

↓

Find closest intersection

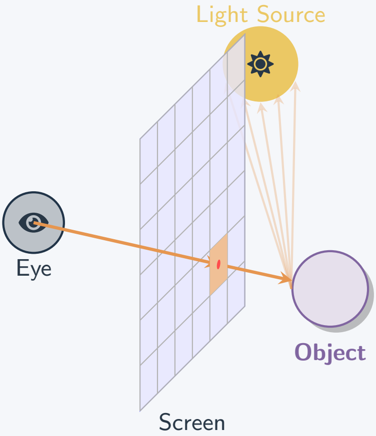
↓

Calculate color*

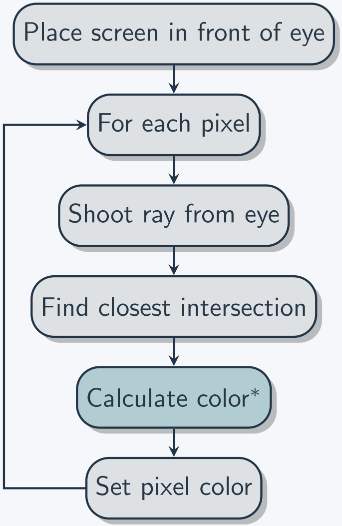


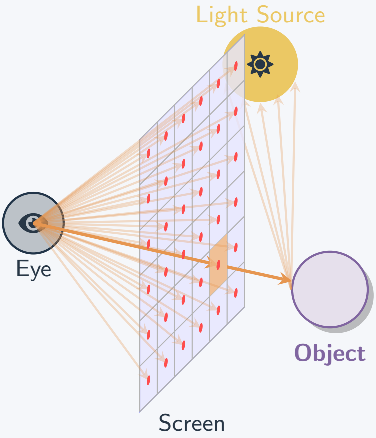
The Full Picture

1. Place screen in front of eye
2. For each pixel
3. Shoot ray from eye
4. Find closest intersection
5. Calculate color*
6. Set pixel color



The Full Picture





Mathematics of Rays

What is a Ray?

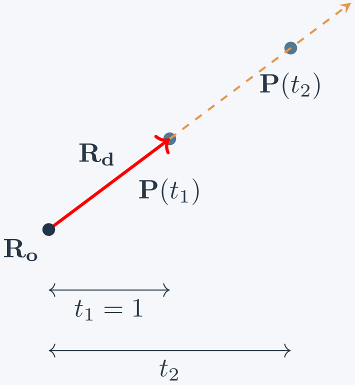
Ray Representation

A ray is defined by:

$$\mathbf{P}(t) = \mathbf{R_o} + t \cdot \mathbf{R_d} \quad (1)$$

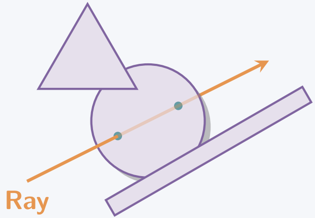
where:

- $\mathbf{R_o}$ = 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:

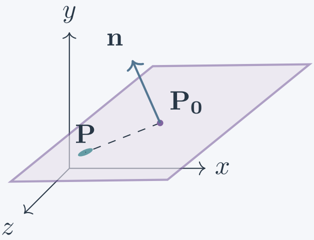
- Point $\mathbf{P_0} = (x_0, y_0, z_0)$ on plane
- Normal vector $\mathbf{n} = (A, B, C)$

Implicit equation:

$$\mathbf{n} \cdot (\mathbf{P} - \mathbf{P_0}) = 0$$

$$\boxed{\mathbf{n} \cdot \mathbf{P} + D = 0} \text{ where } D = -\mathbf{n} \cdot \mathbf{P_0}$$

$$\boxed{Ax + By + Cz + D = 0}$$



3D Plane Representation

Plane Definition

A plane is defined by:

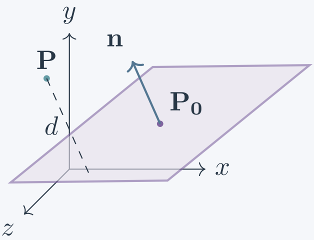
- Point $\mathbf{P_0} = (x_0, y_0, z_0)$ on plane
- Normal vector $\mathbf{n} = (A, B, C)$

Implicit equation:

$$\mathbf{n} \cdot (\mathbf{P} - \mathbf{P_0}) = 0$$

$$\boxed{\mathbf{n} \cdot \mathbf{P} + D = 0} \text{ where } D = -\mathbf{n} \cdot \mathbf{P_0}$$

$$\boxed{Ax + By + Cz + D = 0}$$



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

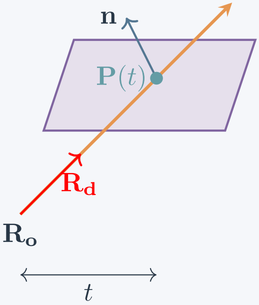
Intersection Method

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

$$\boxed{t = -\frac{D + \mathbf{n} \cdot \mathbf{R_o}}{\mathbf{n} \cdot \mathbf{R_d}}}$$



Ray-Plane Intersection

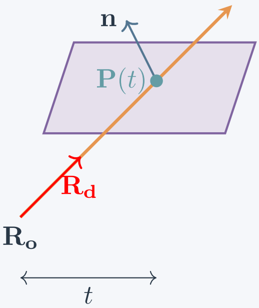
Intersection Method

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

$$\boxed{t = -\frac{D + \mathbf{n} \cdot \mathbf{R_o}}{\mathbf{n} \cdot \mathbf{R_d}}}$$



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

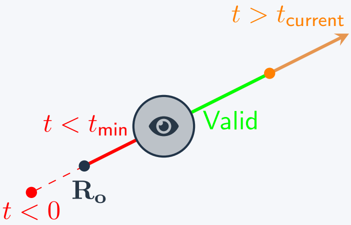
Validation Rules

After computing t, verify:

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

Where:

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



Ray-Triangle Intersection Overview

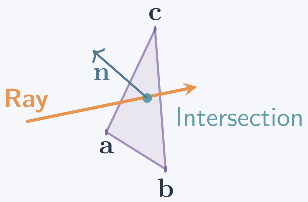
Two Main Approaches

Method 1: Two-Step Process

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

Method 2: Direct Barycentric

1. Set up 3×3 linear system
2. Solve for t, β, γ simultaneously



What Are Barycentric Coordinates?

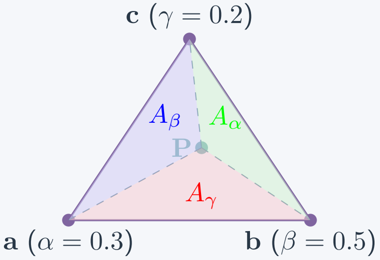
Barycentric Definition

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:

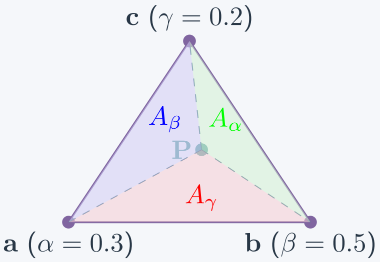
$$\mathbf{P}(\alpha, \beta, \gamma) = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$$

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

Physical Interpretation:

- α, β, γ 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:

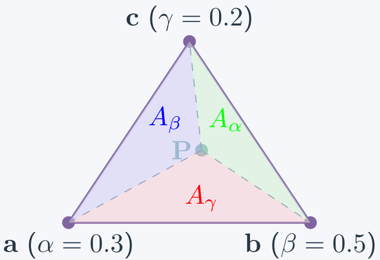
$$\mathbf{P}(\alpha, \beta, \gamma) = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$$

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

Physical Interpretation:

- α, β, γ are *weights*
- P is the *center of mass*
- Also called *barycenter*

Check out the Desmos demo.



Area Relationship

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

Barycentric Coordinates: Inside vs Outside

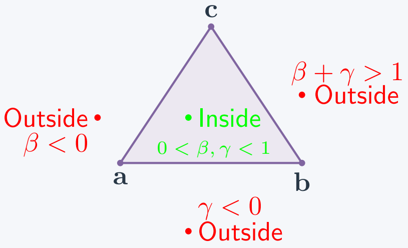
Triangle Interior Test

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

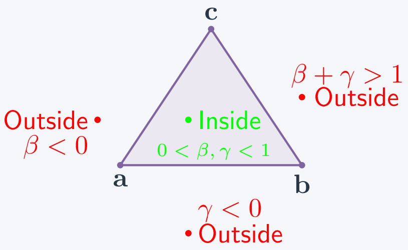
Triangle Interior Test

Point P is **inside** triangle if:

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

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

$$\beta \geq 0$$
$$\gamma \geq 0$$
$$\beta + \gamma \leq 1$$



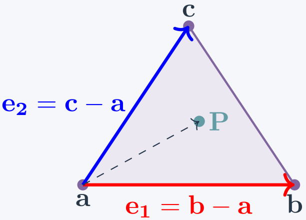
Insight

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

Barycentric Coordinates: Derivation

Key Idea

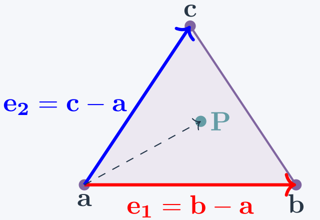
- The sides $\mathbf{e_1} = \mathbf{b} - \mathbf{a}$ and $\mathbf{e_2} = \mathbf{c} - \mathbf{a}$ are linearly independent vectors on the triangle's plane.



Barycentric Coordinates: Derivation

Key Idea

- The sides $\mathbf{e_1} = \mathbf{b} - \mathbf{a}$ and $\mathbf{e_2} = \mathbf{c} - \mathbf{a}$ are linearly independent vectors on the triangle's plane.
- Therefore, any vector in the triangle's plane (e.g. $\mathbf{P} - \mathbf{a}$) can be expressed as a linear combination of these vectors.



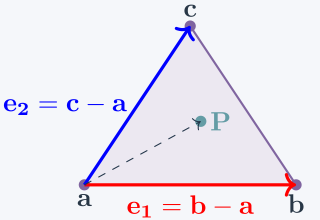
Barycentric Coordinates: Derivation

Key Idea

- The sides $\mathbf{e_1} = \mathbf{b} - \mathbf{a}$ and $\mathbf{e_2} = \mathbf{c} - \mathbf{a}$ are linearly independent vectors on the triangle's plane.
- Therefore, any vector in the triangle's plane (e.g. $\mathbf{P} - \mathbf{a}$) can be expressed as a linear combination of these vectors.
- We can express $\mathbf{P}$ as:

$$\begin{aligned}\mathbf{P} &= \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}) \\ &= \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}\end{aligned}$$

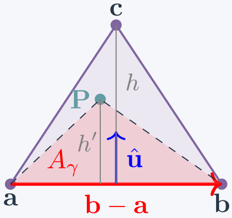
Where $\alpha = 1 - \beta - \gamma$.



Barycentric Coordinates: Derivation

Area Interpretation

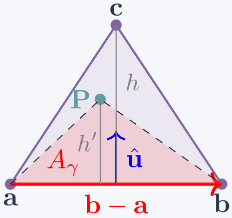
- Let $\hat{\mathbf{u}}$ be an unit vector in the direction of the altitude towards C.



Barycentric Coordinates: Derivation

Area Interpretation

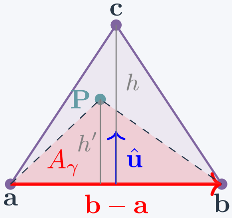
- Let $\hat{\mathbf{u}}$ be an unit vector in the direction of the altitude towards C.
- The height of the triangle is $h = \hat{\mathbf{u}} \cdot (\mathbf{c} - \mathbf{a})$ (projection).



Barycentric Coordinates: Derivation

Area Interpretation

- Let $\hat{\mathbf{u}}$ be an unit vector in the direction of the altitude towards C.
- The height of the triangle is $h = \hat{\mathbf{u}} \cdot (\mathbf{c} - \mathbf{a})$ (projection).
- The height of the shaded triangle is $h' = \hat{\mathbf{u}} \cdot (\mathbf{P} - \mathbf{a})$.

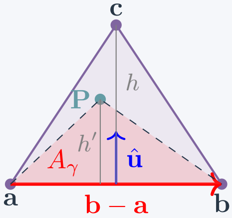


Barycentric Coordinates: Derivation

Area Interpretation

- Let $\hat{\mathbf{u}}$ be an unit vector in the direction of the altitude towards C.
- The height of the triangle is $h = \hat{\mathbf{u}} \cdot (\mathbf{c} - \mathbf{a})$ (projection).
- The height of the shaded triangle is $h' = \hat{\mathbf{u}} \cdot (\mathbf{P} - \mathbf{a})$.
- Hence,

$$
\begin{aligned}
A_\gamma &= \frac{1}{2} \cdot h' \cdot |\mathbf{b} - \mathbf{a}| \\
&= \frac{1}{2} \cdot (\hat{\mathbf{u}} \cdot (\mathbf{P} - \mathbf{a})) \cdot |\mathbf{b} - \mathbf{a}| \\
&= \frac{1}{2} \cdot \gamma\, (\hat{\mathbf{u}} \cdot (\mathbf{c} - \mathbf{a})) \cdot |\mathbf{b} - \mathbf{a}| \\
&= \gamma \frac{1}{2} \cdot h \cdot |\mathbf{b} - \mathbf{a}| = \gamma A_{total}
\end{aligned}
$$

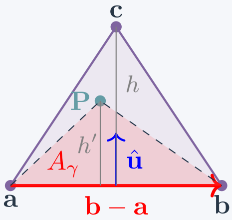


Barycentric Coordinates: Derivation

Area Interpretation

- Let $\hat{\mathbf{u}}$ be an unit vector in the direction of the altitude towards C.
- The height of the triangle is $h = \hat{\mathbf{u}} \cdot (\mathbf{c} - \mathbf{a})$ (projection).
- The height of the shaded triangle is $h' = \hat{\mathbf{u}} \cdot (\mathbf{P} - \mathbf{a})$.
- Hence,

$$\begin{aligned} A_\gamma &= \frac{1}{2} \cdot h' \cdot |\mathbf{b} - \mathbf{a}| \\ &= \frac{1}{2} \cdot (\hat{\mathbf{u}} \cdot (\mathbf{P} - \mathbf{a})) \cdot |\mathbf{b} - \mathbf{a}| \\ &= \frac{1}{2} \cdot \gamma \left(\hat{\mathbf{u}} \cdot (\mathbf{c} - \mathbf{a})\right) \cdot |\mathbf{b} - \mathbf{a}| \\ &= \gamma \frac{1}{2} \cdot h \cdot |\mathbf{b} - \mathbf{a}| = \gamma A_{total} \end{aligned}$$



Since,

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

$$\begin{aligned} \mathbf{P} - \mathbf{a} &= \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}) \\ \hat{\mathbf{u}} \cdot (\mathbf{P} - \mathbf{a}) &= \gamma(\hat{\mathbf{u}} \cdot (\mathbf{c} - \mathbf{a})) \end{aligned}$$

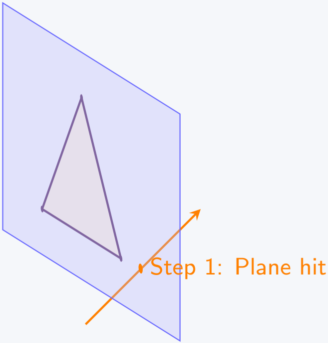
Since $\hat{\mathbf{u}}$ is perpendicular to $\mathbf{b} - \mathbf{a}$.

Method 1: Two-Step Ray-Triangle Intersection

Algorithm Steps

Step 1: Ray-Plane Intersection

$$t = -\frac{D + \mathbf{n} \cdot \mathbf{R_o}}{\mathbf{n} \cdot \mathbf{R_d}}$$



Method 1: Two-Step Ray-Triangle Intersection

Algorithm Steps

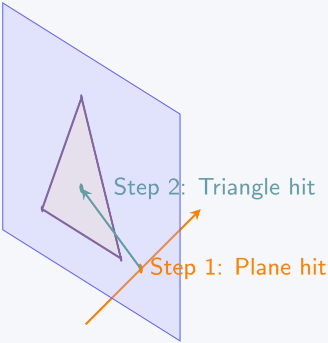
Step 1: Ray-Plane Intersection

$$t = -\frac{D + \mathbf{n} \cdot \mathbf{R_o}}{\mathbf{n} \cdot \mathbf{R_d}}$$

Step 2: Inside/Outside Test

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

Solve for β, γ and check bounds.



Method 2: Direct Barycentric Intersection

Direct Approach

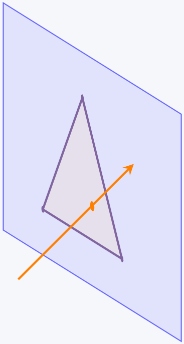
Set ray equation equal to barycentric form:

$$\mathbf{R_o} + t\mathbf{R_d} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$$

Rearrange to linear system:

$$\begin{bmatrix} -\mathbf{R_d} & (\mathbf{b} - \mathbf{a}) & (\mathbf{c} - \mathbf{a}) \end{bmatrix} \begin{bmatrix} t \\ \beta \\ \gamma \end{bmatrix} = \mathbf{R_o} - \mathbf{a}$$

Solve using Cramer's rule or
LU decomposition.



Cramer's Rule Solution

Matrix Form

$$\underbrace{\begin{bmatrix} -R_{dx} & (b_x - a_x) & (c_x - a_x) \\ -R_{dy} & (b_y - a_y) & (c_y - a_y) \\ -R_{dz} & (b_z - a_z) & (c_z - a_z) \end{bmatrix}}_{A} \begin{bmatrix} t \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} R_{ox} - a_x \\ R_{oy} - a_y \\ R_{oz} - a_z \end{bmatrix}$$

Cramer's Rule

$$t = \frac{1}{|A|} \begin{vmatrix} (R_o - a)_x & (b - a)_x & (c - a)_x \\ (R_o - a)_y & (b - a)_y & (c - a)_y \\ (R_o - a)_z & (b - a)_z & (c - a)_z \end{vmatrix}$$

$$\beta = \frac{1}{|A|} \begin{vmatrix} -R_{dx} & (R_o - a)_x & (c - a)_x \\ -R_{dy} & (R_o - a)_y & (c - a)_y \\ -R_{dz} & (R_o - a)_z & (c - a)_z \end{vmatrix}$$

$$\gamma = \frac{1}{|A|} \begin{vmatrix} -R_{dx} & (b - a)_x & (R_o - a)_x \\ -R_{dy} & (b - a)_y & (R_o - a)_y \\ -R_{dz} & (b - a)_z & (R_o - a)_z \end{vmatrix}$$

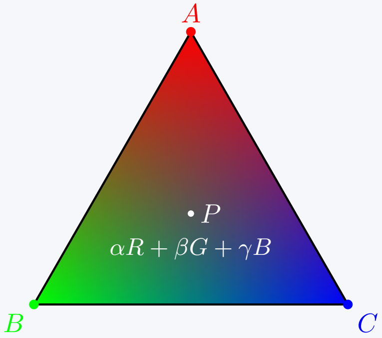
Checks

- $t_{\text{min}} < t < t_{\text{current}}$ (valid intersection)
- $\beta, \gamma \geq 0$ and $\beta + \gamma \leq 1$ (inside triangle)

Added Benefits of Barycentric Coordinates

Advantages

- Efficient to compute
- Get Barycentric coordinates for free
- Enables interpolation of vertex attributes
  Used in —
  - Textures
  - Normals
  - Colors



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.
  Availabe 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
- Scratchapixel: Learn Computer Graphics Programming.
  scratchapixel.com