CSCI 420 Computer Graphics
Lecture 16

Geometric Queries for Ray Tracing

Ray-Surface Intersection
Barycentric Coordinates
[Angel Ch. 11]

Oded Stein
University of Southern California

Backward Ray Tracing

- Main components of the backward ray tracing algorithm:

1. For each pixel (x,y), fire a ray from COP through (x,y)
2. For each ray & object, calculate closest intersection
3. For closest intersection point **p**
   - Calculate surface normal
   - For each light source, fire shadow ray
   - For each unblocked shadow ray, evaluate local Phong model for that light, and add the result to pixel color

Backward Ray Tracing

- Main components of the backward ray tracing algorithm:

1. For each pixel (x,y), fire a ray from COP through (x,y)
2. For each ray & object, calculate closest intersection
3. For closest intersection point **p**
   - Calculate surface normal
   - For each light source, fire shadow ray
   - For each unblocked shadow ray, evaluate local Phong model for that light, and add the result to pixel color

But how do we calculate the closest intersection?

How do we calculate occlusions for shadow rays?

Ray-Surface Intersections

- Necessary in ray tracing
- General implicit surfaces
- General parametric surfaces
- Specialized analysis for special surfaces
  - Spheres
  - Planes
  - Polygons
  - Quadrics

Ray-Surface Intersections

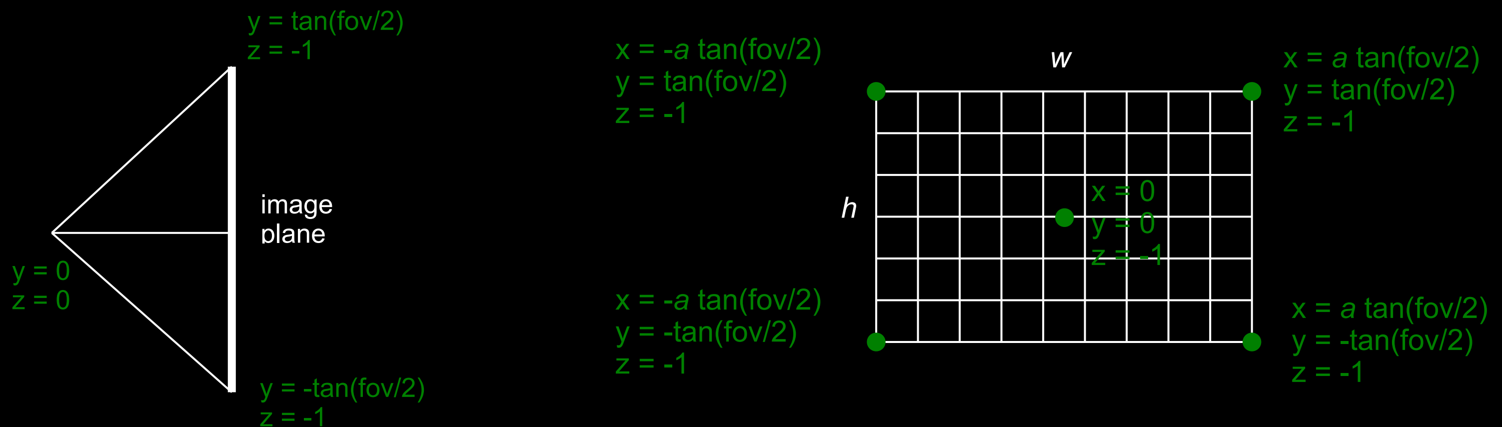
- Necessary in ray tracing
- General implicit surfaces
- General parametric surfaces
- Specialized analysis for special surfaces
  - Spheres
  - Planes
  - Polygons
  - Quadrics

- **Triangles**

Mathematical parametrization of rays

- Ray in parametric form
  - Origin $\mathbf{p}_0 = [x_0 \;\; y_0 \;\; z_0]^T$
  - Direction $\mathbf{d} = [x_d \;\; y_d \;\; z_d]^T$
  - Assume $\mathbf{d}$ is normalized $(x_d^2 + y_d^2 + z_d^2 = 1)$
  - Ray $\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{d}\, t$ for $t > 0$

- Remember from last lecture:



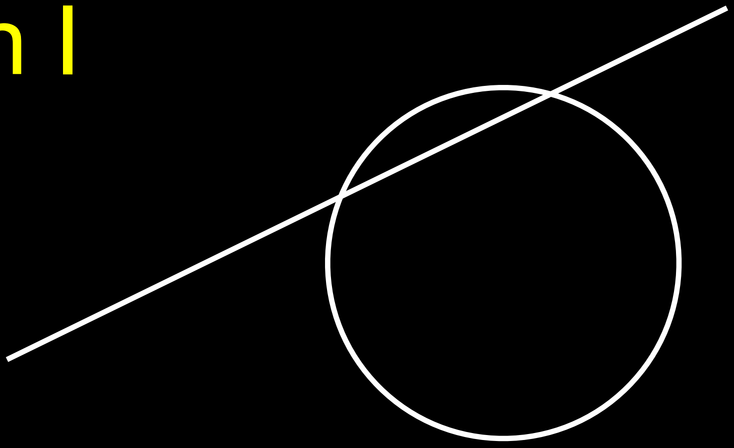
Intersection of Rays and Parametric Surfaces

- Ray in parametric form
  - Origin $\mathbf{p}_0 = [x_0 \ \ y_0 \ \ z_0]^T$
  - Direction $\mathbf{d} = [x_d \ \ y_d \ \ z_d]^T$
  - Assume $\mathbf{d}$ is normalized ($x_d^2 + y_d^2 + z_d^2 = 1$)
  - Ray $\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{d}\, t$ for $t > 0$
- Surface in parametric form
  - Point $\mathbf{q} = g(u, v)$, possible bounds on u, v
  - Solve $\mathbf{p}_0 + \mathbf{d}\, t = g(u, v)$
  - Three equations in three unknowns (t, u, v)

Intersection of Rays and Implicit Surfaces

- Ray in parametric form
  - Origin $\mathbf{p}_0 = [x_0 \;\; y_0 \;\; z_0]^T$
  - Direction $\mathbf{d} = [x_d \;\; y_d \;\; z_d]^T$
  - Assume $\mathbf{d}$ normalized ($x_d^2 + y_d^2 + z_d^2 = 1$)
  - Ray $\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{d}\, t$ for $t > 0$
- Implicit surface
  - Given by $f(\mathbf{q}) = 0$
  - Consists of all points $\mathbf{q}$ such that $f(\mathbf{q}) = 0$
  - Substitute ray equation for $\mathbf{q}$: $f(\mathbf{p}_0 + \mathbf{d}\, t) = 0$
  - Solve for t (univariate root finding)
  - Closed form (if possible), otherwise numerical approximation

Ray-Sphere Intersection I



- Common and easy case
- Define sphere by
  - Center **c** = $[x_c \ \ y_c \ \ z_c]^T$
  - Radius r
  - Surface f(**q**) = $(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 - r^2 = 0$
- Plug in ray equations for x, y, z:

$$x = x_0 + x_d t, \qquad y = y_0 + y_d t, \qquad z = z_0 + z_d t$$

- And we obtain a scalar equation for *t*:

$$(x_0 + x_d t - x_c)^2 + (y_0 + y_d t - y_c)^2 + (z_0 + z_d t - z_c)^2 = r^2$$

Ray-Sphere Intersection II

- Simplify to

$$at^2 + bt + c = 0$$

where

$$a = x_d^2 + y_d^2 + z_d^2 = 1 \qquad \text{since } |d| = 1$$
$$b = 2(x_d(x_0 - x_c) + y_d(y_0 - y_c) + z_d(z_0 - z_c))$$
$$c = (x_0 - x_c)^2 + (y_0 - y_c)^2 + (z_0 - z_c)^2 - r^2$$

- Solve to obtain t_0 and t_1

$$t_{0,1} = \frac{-b \pm \sqrt{b^2 - 4c}}{2}$$

Check if t_0, t_1 > 0 (ray)
Return min(t_0, t_1)

Ray-Sphere Intersection III

- For lighting, calculate unit normal

$$n = \frac{1}{r}\left[(x_i - x_c) \quad (y_i - y_c) \quad (z_i - z_c)\right]^T$$

- Negate if ray originates inside the sphere!
  - (what should we do if the ray originates inside the sphere?)
- Note possible problems with roundoff errors

Simple Optimizations

- Factor common subexpressions
- Compute only what is necessary
  - Calculate $b^2 - 4c$, abort if negative
  - Compute normal only for closest intersection
  - Other similar optimizations

$$at^2 + bt + c = 0$$

$$t_{0,1} = \frac{-b \pm \sqrt{b^2 - 4c}}{2}$$

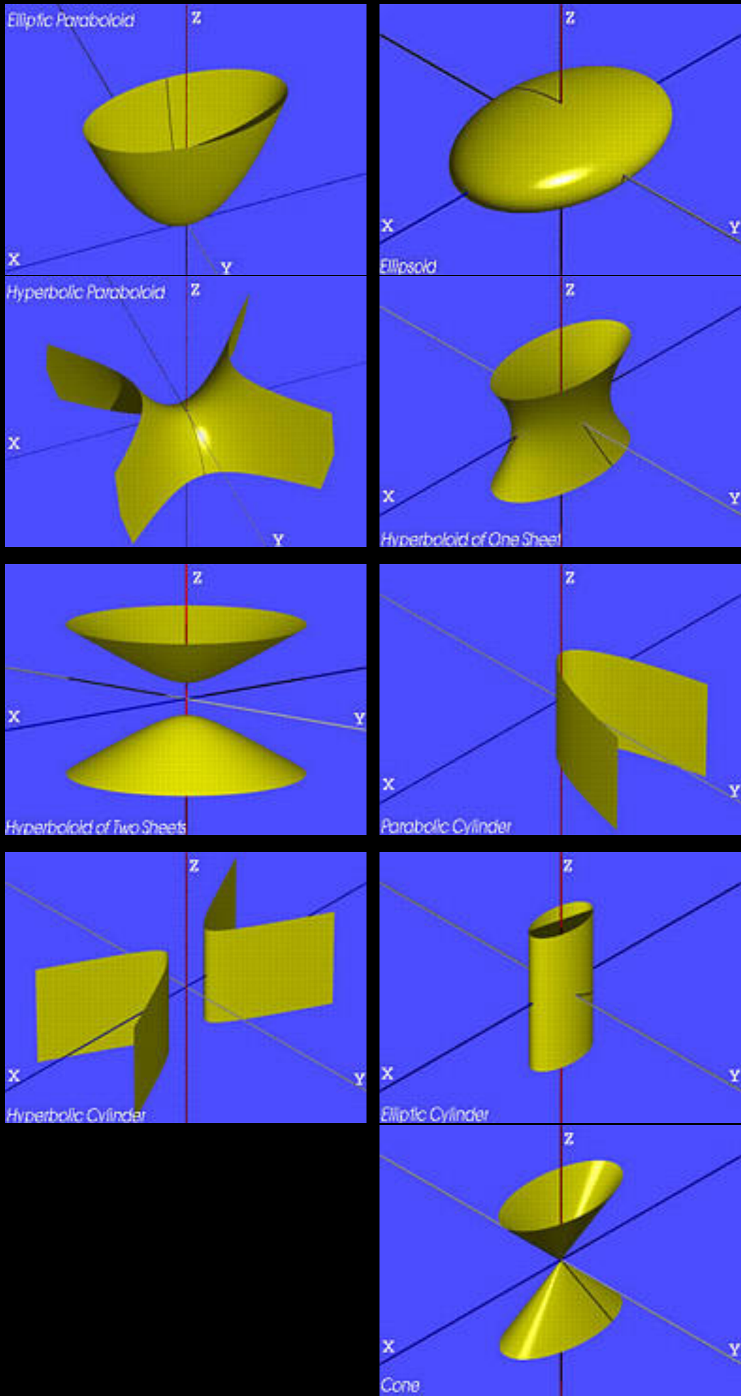
$$a = x_d^2 + y_d^2 + z_d^2 = 1 \qquad \text{since } |d| = 1$$

$$b = 2(x_d(x_0 - x_c) + y_d(y_0 - y_c) + z_d(z_0 - z_c))$$

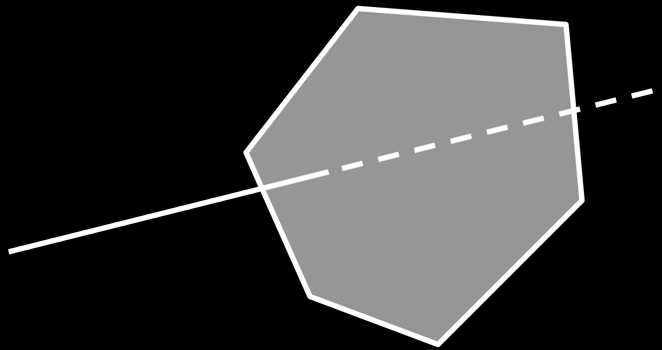
$$c = (x_0 - x_c)^2 + (y_0 - y_c)^2 + (z_0 - z_c)^2 - r^2$$

Ray-Quadric Intersection

- Quadric f(**p**) = f(x, y, z) = 0,
  where f is polynomial of order 2
- Sphere, ellipsoid, paraboloid, hyperboloid, cone, cylinder
- Closed form solution as for sphere
- Important case for modelling in ray tracing
- Combine with CSG
- (not on final)



Ray-Polygon Intersection I



- Assume planar polygon in 3D
  1. Intersect ray with plane containing polygon
  2. Check if intersection point is inside polygon
- Plane
  - Implicit form: $ax + by + cz + d = 0$
  - Unit normal: $\mathbf{n} = [a \;\; b \;\; c]^T$ with $a^2 + b^2 + c^2 = 1$
- Substitute:

$$a(x_0 + x_d t) + b(y_0 + y_d t) + c(z_0 + z_d t) + d = 0$$

- Solve:

$$t = \frac{-(ax_0 + by_0 + cz_0 + d)}{ax_d + by_d + cz_d}$$

Ray-Polygon Intersection II

- Substitute *t* to obtain intersection point in plane

d coefficient
of plane

- Rewrite using dot product

$$t = \frac{-(ax_0 + by_0 + cz_0 + d)}{ax_d + by_d + cz_d} = \frac{-(n \cdot p_0 + d)}{n \cdot d}$$

ray
direction

- **n • d** = 0 - danger of division by 0?

Ray-Polygon Intersection II

- Substitute *t* to obtain intersection point in plane
- Rewrite using dot product

d coefficient of plane

$$t = \frac{-(ax_0 + by_0 + cz_0 + d)}{ax_d + by_d + cz_d} = \frac{-(n \cdot p_0 + d)}{n \cdot d}$$

ray direction

- If **n** • **d** = 0, no intersection (ray parallel to plane)
- If t ≤ 0, the intersection is behind ray origin

Test if point inside polygon

- Could use even-odd rule, or winding rule
- Easier if polygon is in 2D
  (project from 3D to 2D)
- Easier for triangles (tessellate polygons)

Test if point inside polygon

- Could use even-odd rule, or winding rule
- Easier if polygon is in 2D
  (project from 3D to 2D)
- Easier for triangles (tessellate polygons)
- In practice, **everyone tesselates**
  - This also solves problem that polygon might not be completely planar...

Point-in-triangle testing

- Critical for polygonal models

- Project the triangle, and point of plane intersection, onto one of the planes
  $x = 0$, $y = 0$, or $z = 0$
  (pick a plane not perpendicular to triangle)
  (such a choice always exists)

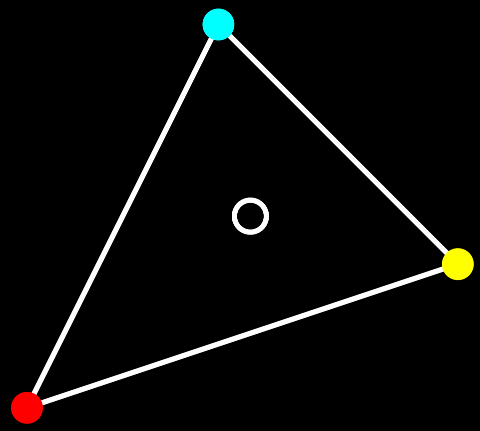
- Then, do the 2D test in the plane, by computing barycentric coordinates

Outline

- Ray-Surface Intersections
- Special cases: sphere, polygon
- Barycentric Coordinates

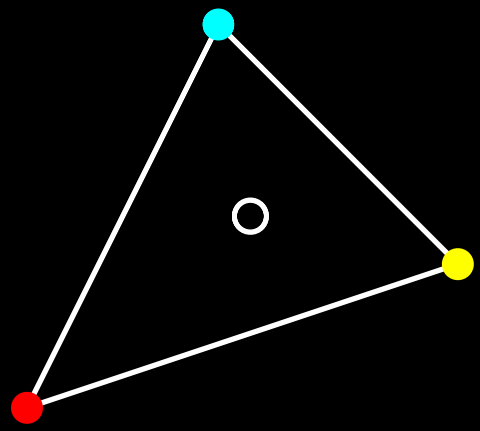
Barycentric coordinates - what are they good for?

- Barycentric coordinates will allow us to express a point in the interior of a triangle using the coordinates of the vertices
- **Useful for many things**
  - inside/outside test
  - pre-fragment shader interpolation



Interpolated Shading for Ray Tracing

- Assume we know normals at vertices
- How do we compute normal of interior point?
- Need linear interpolation between 3 points
- Barycentric coordinates



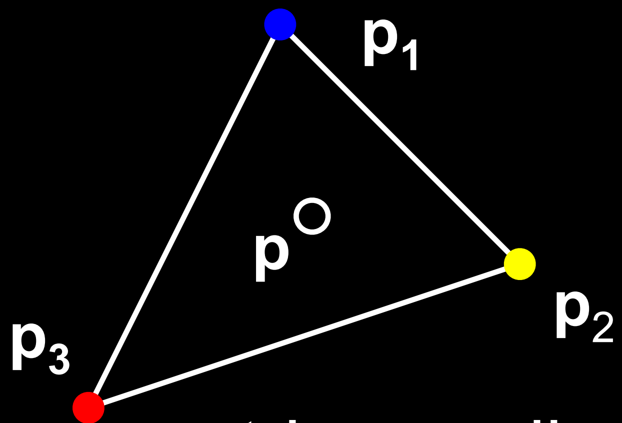
Barycentric Coordinates in 1D

- Linear interpolation
  - $\mathbf{p}(t) = (1 - t)\mathbf{p}_1 + t\,\mathbf{p}_2,\ 0 \le t \le 1$
  - $\mathbf{p}(t) = \alpha\,\mathbf{p}_1 + \beta\,\mathbf{p}_2$ where $\alpha + \beta = 1$
  - $\mathbf{p}$ is between $\mathbf{p}_1$ and $\mathbf{p}_2$ iff $0 \le \alpha, \beta \le 1$
- Geometric intuition
  - Weigh each vertex by ratio of distances from ends

- α, β are called barycentric coordinates

Barycentric Coordinates in 2D

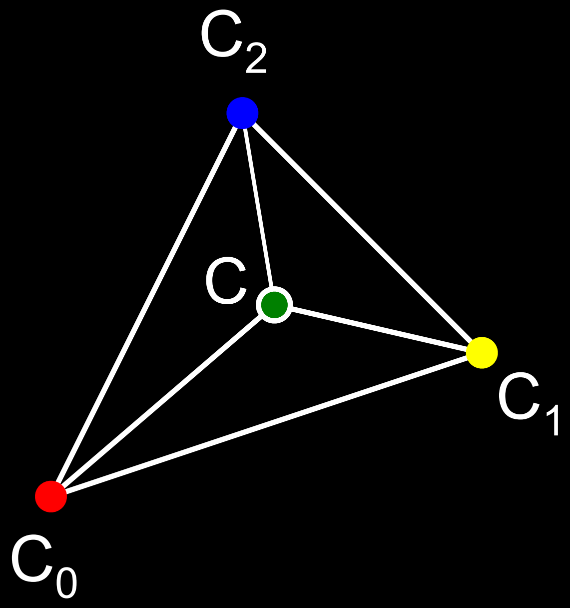
- Now, we have 3 points instead of 2



- Define 3 barycentric coordinates, α, β, γ
- $\mathbf{p} = \alpha\, \mathbf{p}_1 + \beta\, \mathbf{p}_2 + \gamma\, \mathbf{p}_3$
- $\mathbf{p}$ inside triangle iff $0 \leq \alpha, \beta, \gamma \leq 1$, $\alpha + \beta + \gamma = 1$
- How do we calculate α, β, γ given $\mathbf{p}$?

Barycentric Coordinates for Triangle

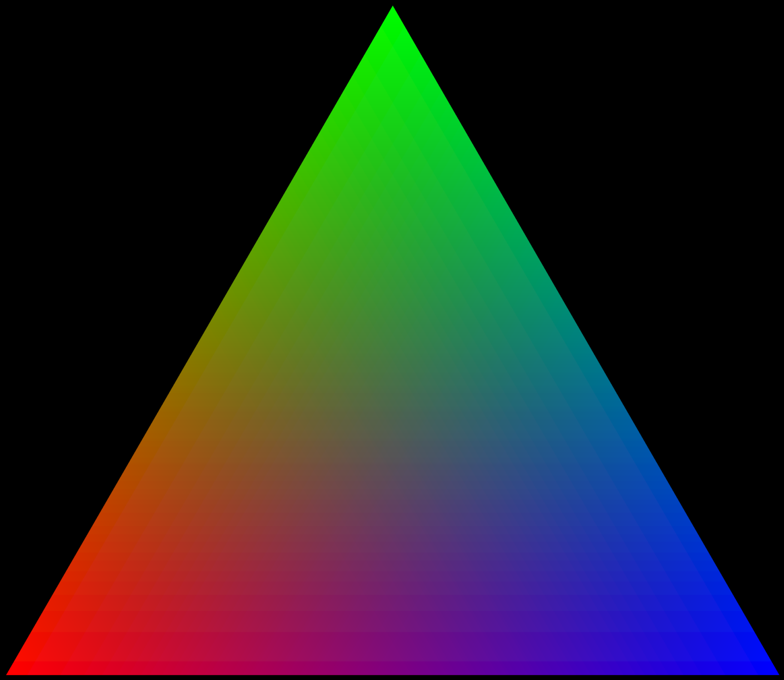
- Coordinates are ratios of triangle areas



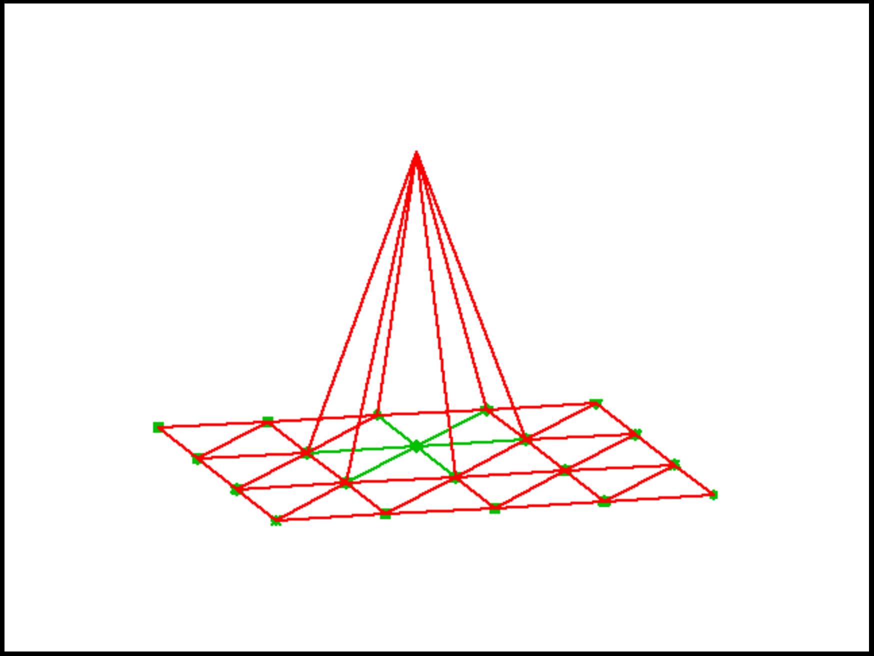
- Areas in these formulas should be signed, depending on clockwise (-) or anti-clockwise orientation (+) of the triangle! Very important for point-in-triangle test.

Barycentric Coordinates for Triangle

- Coordinates are also basis functions
- "Triangle weights"
- Barycentric coordinate functions

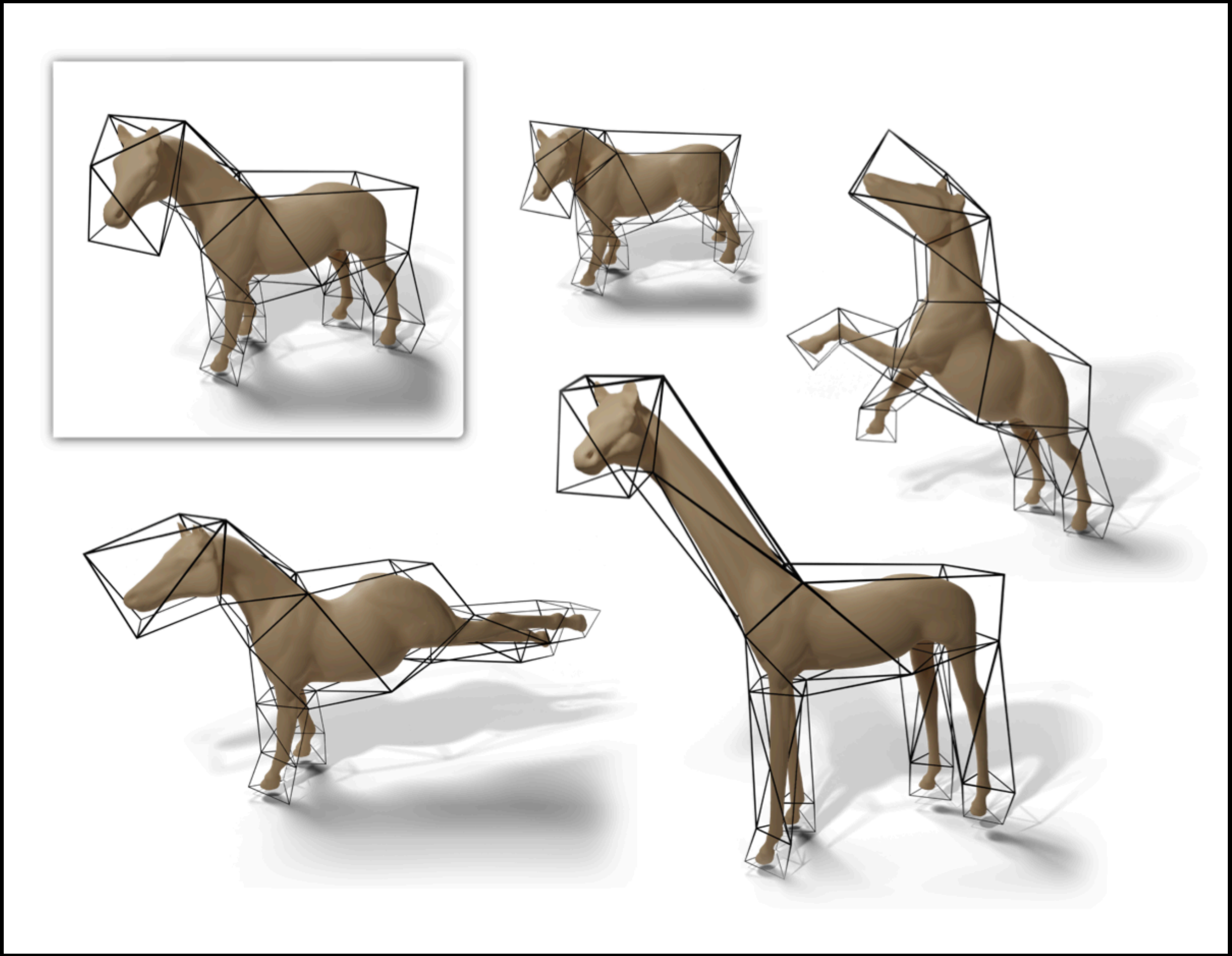


wikipedia.org



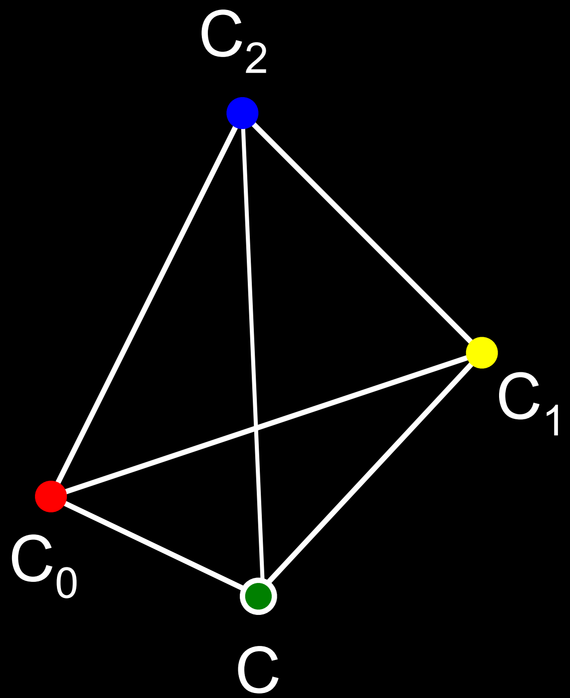
hplgit.github.io

Other uses for Barycentric Coordinates



Dodik et al. 2023. "Variational Barycentric Coordinates"

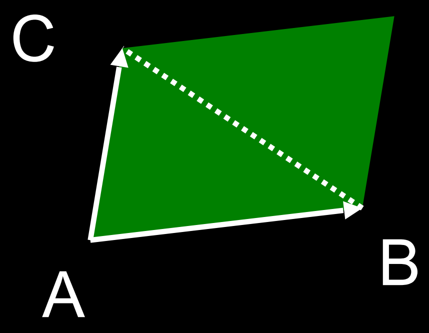
Negative Area



< 0

Point C is outside of the triangle!

Computing Triangle Area in 3D



- Use cross product
- Parallelogram formula
- Area(ABC) = (1/2) |(B – A) x (C – A)|
- How to get correct sign for barycentric coordinates?
  - tricky, but possible:
    compare directions of vectors (B – A) x (C – A), for triangles CC_1C_2 vs $C_0C_1C_2$, etc.
    (either 0 (sign+) or 180 deg (sign-) angle)
  - easier alternative: project to 2D, use 2D formula
  - projection to 2D preserves barycentric coordinates

Computing Triangle Area in 2D

- Suppose we project the triangle to xy plane

- Area(xy-projection(ABC)) =

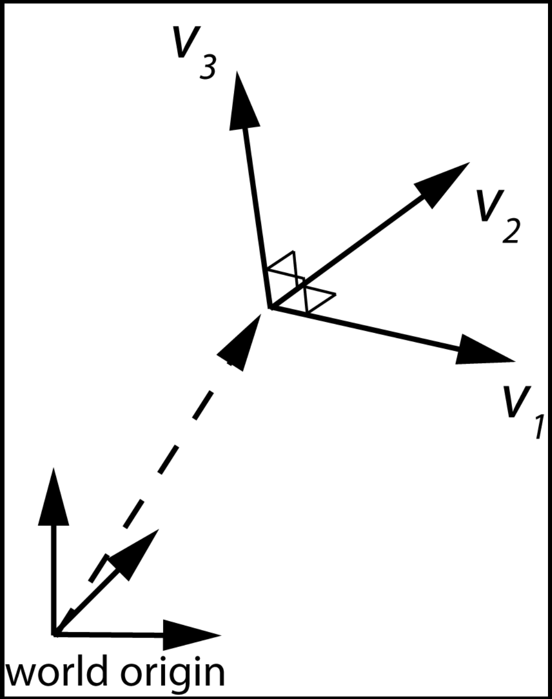
$(1/2)\ ((b_x - a_x)(c_y - a_y) - (c_x - a_x)\ (b_y - a_y))$

- This formula gives correct sign
(important for barycentric coordinates)

- Much easier in 2D, because there is always a constant up direction.

Computing Triangle Area in 2D

- But how to we project to 2D?
- Remember modelview matrices...



$v_1 \cdot v_2 = 0$

$v_2 \cdot v_3 = 0$

$v_1 \cdot v_3 = 0$

$\| v_1 \| = \| v_2 \| = \| v_3 \| = 1$

Orthonormal coordinate system

Change of Coordinate System

- Bases $\{u_1, u_2, u_3\}$ and $\{v_1, v_2, v_3\}$
- Express basis vectors u_i in terms of v_j

$$u_1 = \gamma_{11} v_1 + \gamma_{12} v_2 + \gamma_{13} v_3$$
$$u_2 = \gamma_{21} v_1 + \gamma_{22} v_2 + \gamma_{23} v_3$$
$$u_3 = \gamma_{31} v_1 + \gamma_{32} v_2 + \gamma_{33} v_3$$

- Represent in matrix form:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = M \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \qquad M = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} \end{bmatrix}$$

Computing Triangle Area in 2D

- But how to we project to 2D?
- There are easier alternatives.
- Just zero the z-coordinate.
  - Barycentric coordinates will be preserved.
  - Must be careful in case the triangle is in the xz or yz planes.

Summary

- Ray-Surface Intersections
- Special cases: sphere, polygon
- Barycentric Coordinates