EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

FACULTY OF
SCIENCE
Computer Graphics

Computer Graphics (Graphische Datenverarbeitung)

- Basic Geometry -

Hendrik Lensch

WS 2022/2023

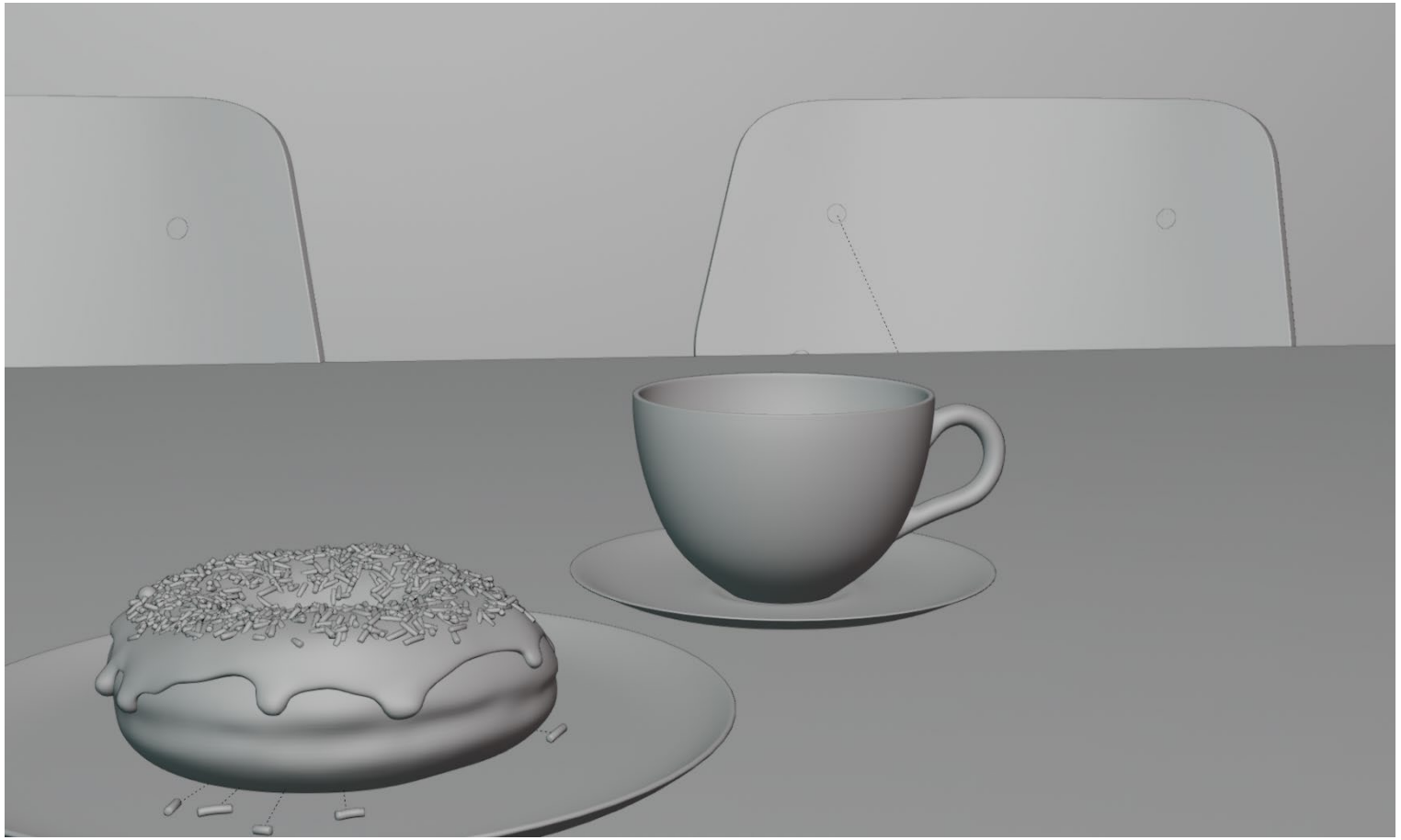
Surface Representation

Rendered Scene

- Photo realistic
- Geometry
- Illumination
- Shading
- Materials
- Textures

Shaded Scene

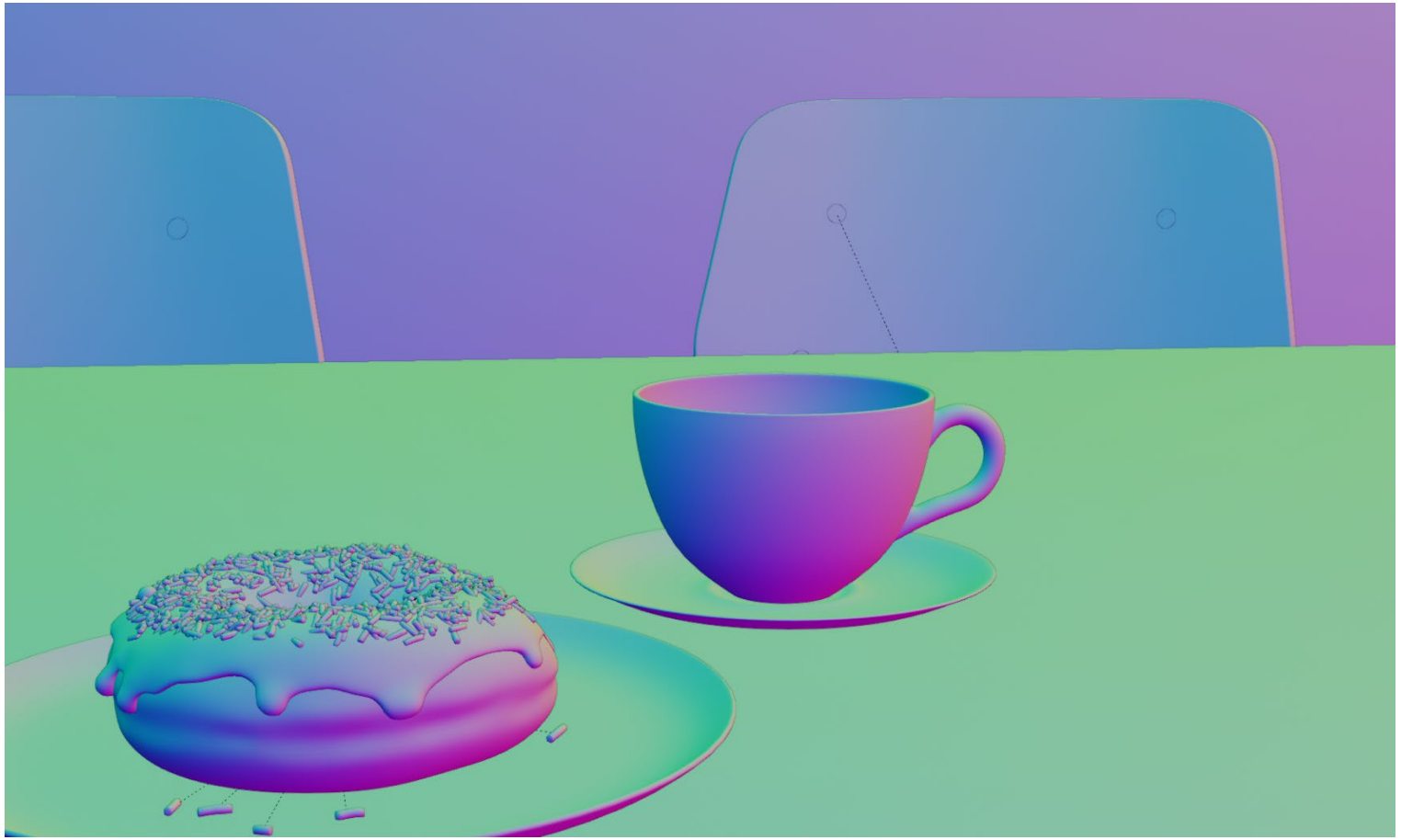
- Photo realistic
- Geometry
- Illumination
- Shading
- Materials
- Textures



Scene Geometry

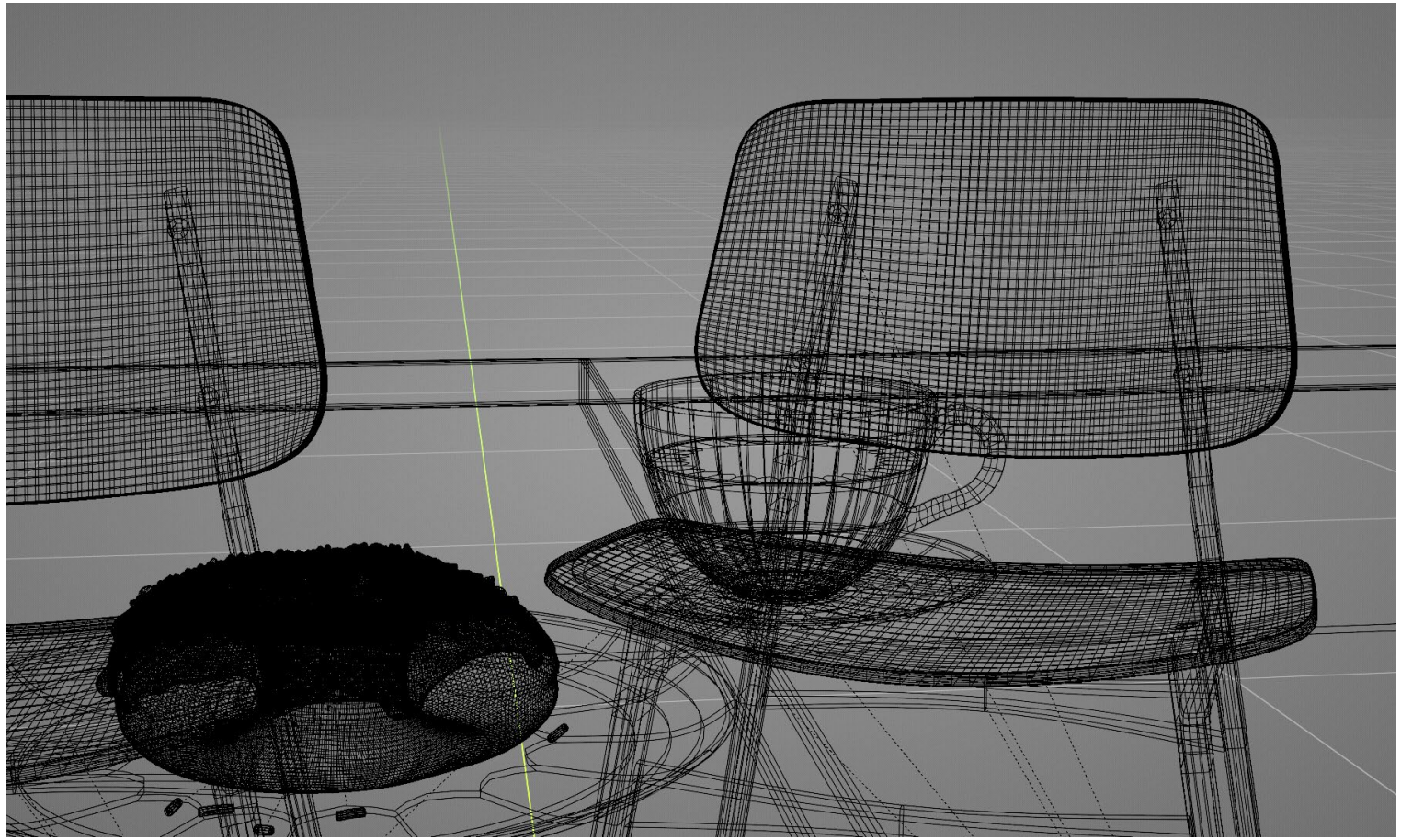
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Photo realistic
- Geometry
  - Smooth surfaces
  - Surface normal
- Illumination
- Shading
- Materials
- Textures



Scene Geometry

- Photo realistic
- Geometry
  - Wireframe
- Illumination
- Shading
- Materials
- Textures



Geometric Primitives

Triangle Meshes

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Stanford Bunny at different mesh resolutions

https://www.wikiwand.com/

Representations of (Solid) 3D Objects

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

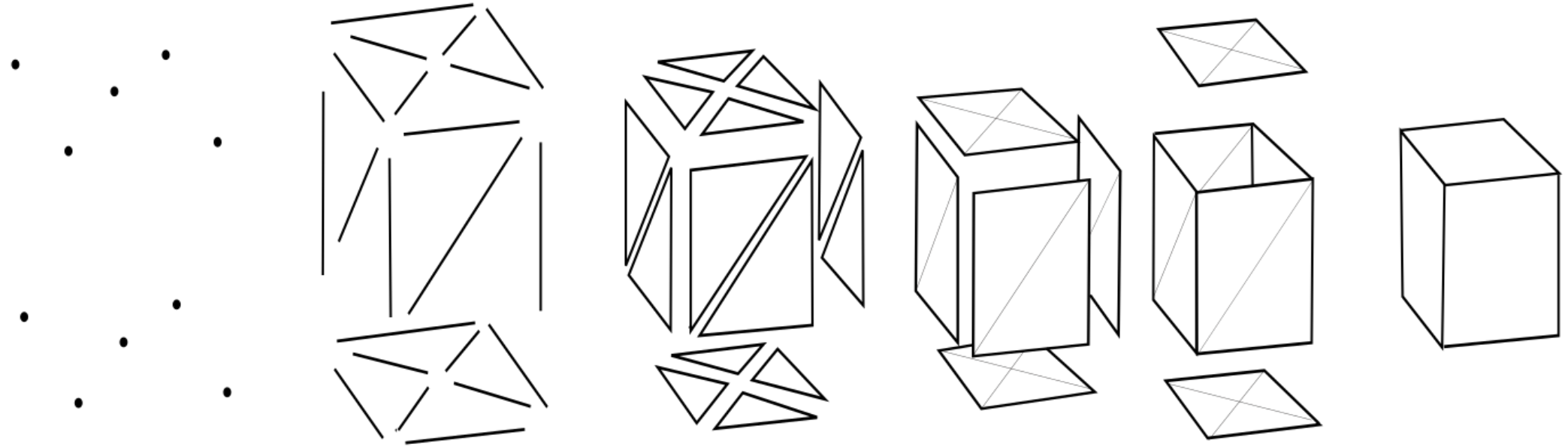
- Complex 3D objects need to be constructed from a set of primitives
  - Representation schema is a mapping of 3D objects to primitives
  - Primitives should be efficiently supported by graphics hardware

- Desirable properties of representation schemata:
  - Can represent many (or all) possible 3D objects
  - Accuracy
  - Compact storage
  - Simple algorithms for manipulation and rendering

- Most popular on modern graphics hardware:
  - Boundary representations (B-Reps) using vertices, edges and faces

Polygon Meshes

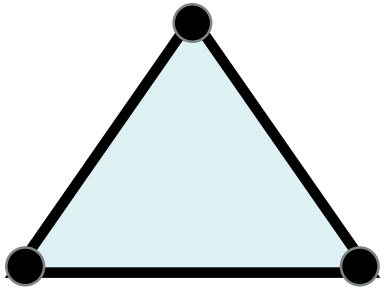
- Describe the surface of an object as a set of polygons
- Mostly use triangles, since they are trivially convex and flat
- Current graphics hardware is optimized for triangle meshes

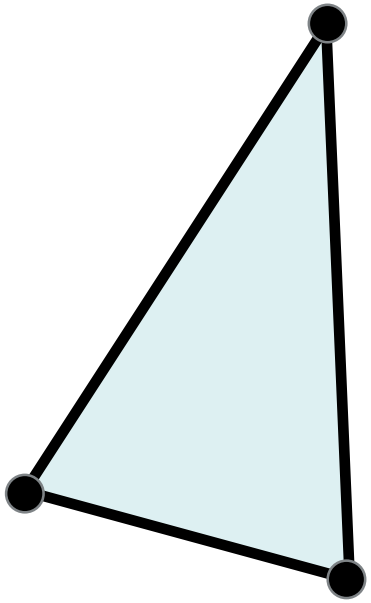


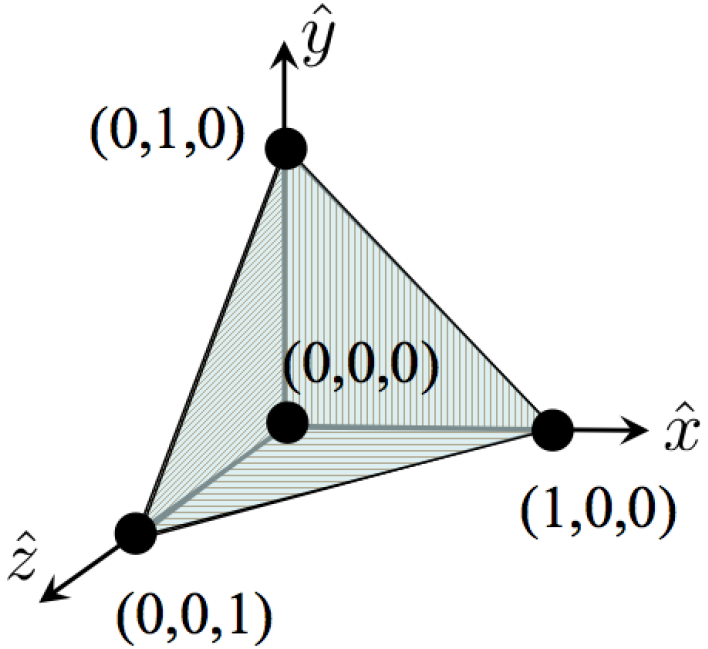
https://en.wikipedia.org/wiki/File:Mesh_overview.svg

The Triangle

- A triangle is the simplest polygonal surface
- Minmal building block, simplex in 2D (tetrahedron in 3D)
- Always planar, i.e. all points in one plane (not always the case with a quadrangle)
- Defined by three vertices (points in 3D)
- Parameterizes the locations (between) the vertices (see barycentric coordinates)







3D Triangles and Planes

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Mathematical descriptions of a 2D plane in 3D space (hyperplane)

1. Point $\vec{p}$ and two non-parallel vectors $\vec{v}$ and $\vec{w}$

$$\boldsymbol{x = \vec{p} + s\vec{v} + t\vec{w}}$$

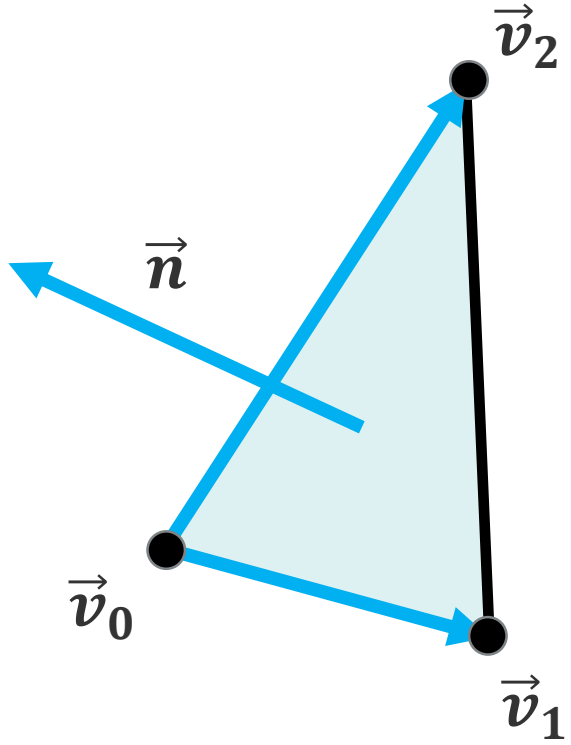
2. Three non-collinear points
   (take one point and the difference vectors to the other two)
3. Point $\vec{p}$ and normal vector $\vec{n}$ for the plane

$$\boldsymbol{\vec{n} \cdot (x - \vec{p}) = 0}$$

4. Single plane equation

$$\boldsymbol{ax_1 + bx_2 + cx_3 + d = 0 \quad a, b, c, d \in \mathbb{R}}$$

(a, b, c) is the normal vector of the plane



- All description methods easily convertible from one to the other
  (e.g. using cross product to compute normal vector)

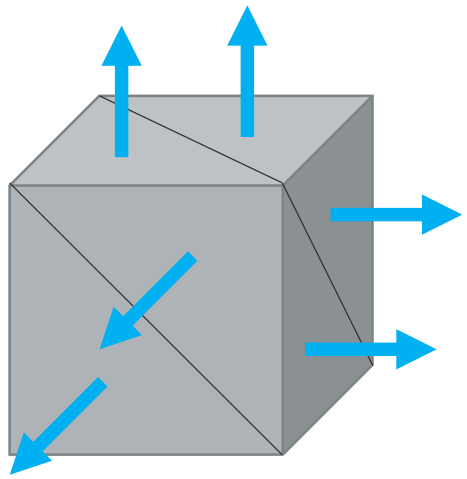
Normal Direction

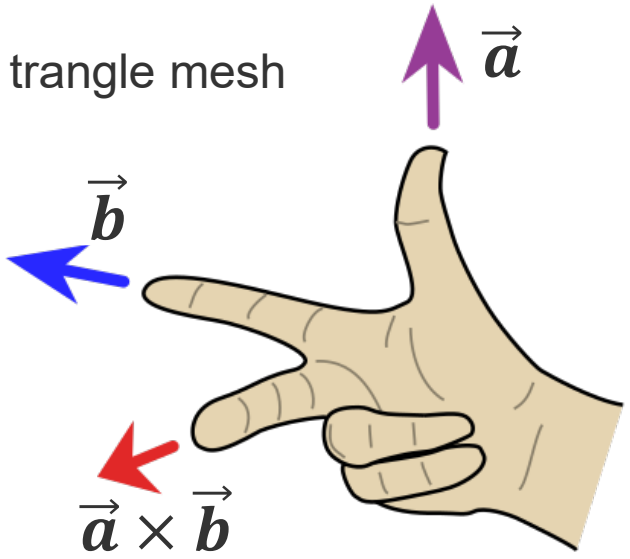
- Normal given by cross-product

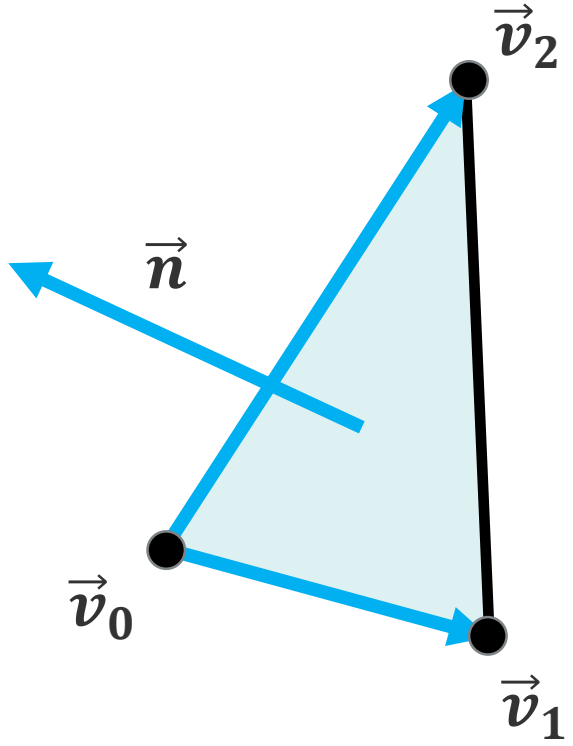
$$\vec{n} = (\vec{v}_1 - \vec{v}_0) \times (\vec{v}_2 - \vec{v}_0)$$

- Follow the right-hand rule

- Defines a front-face of a triangle
  - Triangles are one-sided
  - Normals should point to the outslide of a trangle mesh







Triangle Mesh

- Define the complete surface by enumerating all triangles

$$T_1: (\vec{v}_0, \vec{v}_1, \vec{v}_3)$$

$$T_2: (\vec{v}_1, \vec{v}_2, \vec{v}_3)$$

$$T_3: (\vec{v}_3, \vec{v}_2, \vec{v}_6)$$

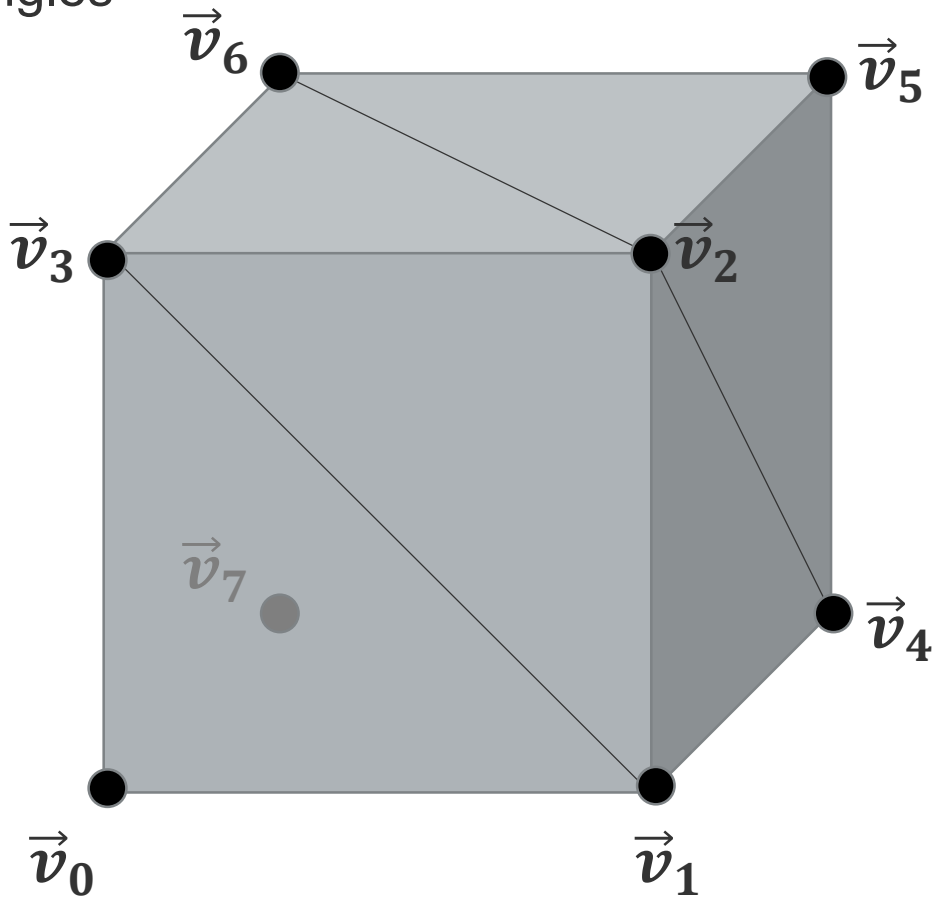
$$T_4: (\vec{v}_2, \vec{v}_5, \vec{v}_6)$$

$$T_5: (\vec{v}_1, \vec{v}_4, \vec{v}_2)$$

$$T_6: (\vec{v}_4, \vec{v}_5, \vec{v}_2)$$

$$\vdots$$

$$T_{12}: (\vec{v}_7, \vec{v}_3, \vec{v}_6)$$



Triangle Mesh

- Define the complete surface by enumerating all triangles

$T_1: (\vec{v}_0, \vec{v}_1, \vec{v}_3)$

$T_2: (\vec{v}_1, \vec{v}_2, \vec{v}_3)$

$T_3: (\vec{v}_3, \vec{v}_2, \vec{v}_6)$

$T_4: (\vec{v}_2, \vec{v}_5, \vec{v}_6)$

$T_5: (\vec{v}_1, \vec{v}_4, \vec{v}_2)$

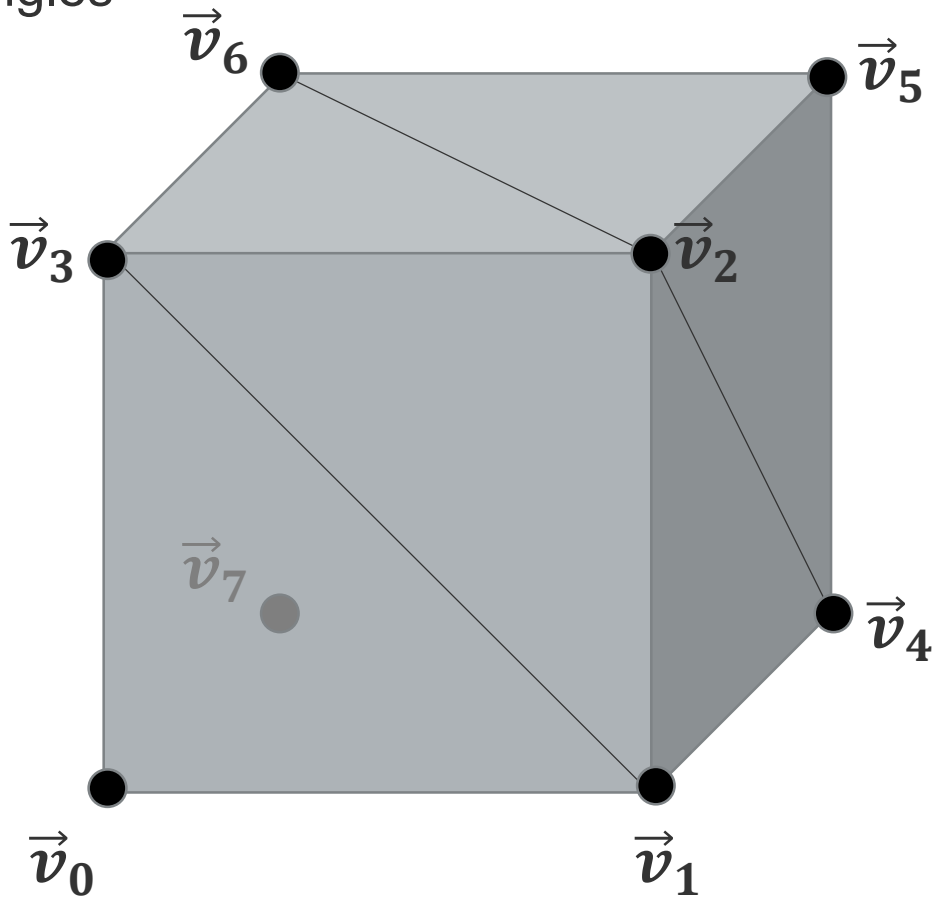
$T_6: (\vec{v}_4, \vec{v}_5, \vec{v}_2)$

$\vdots$

$T_{12}: (\vec{v}_7, \vec{v}_3, \vec{v}_6)$

$\vec{v}_2$ reoccurs 5 times

- Redundant representation

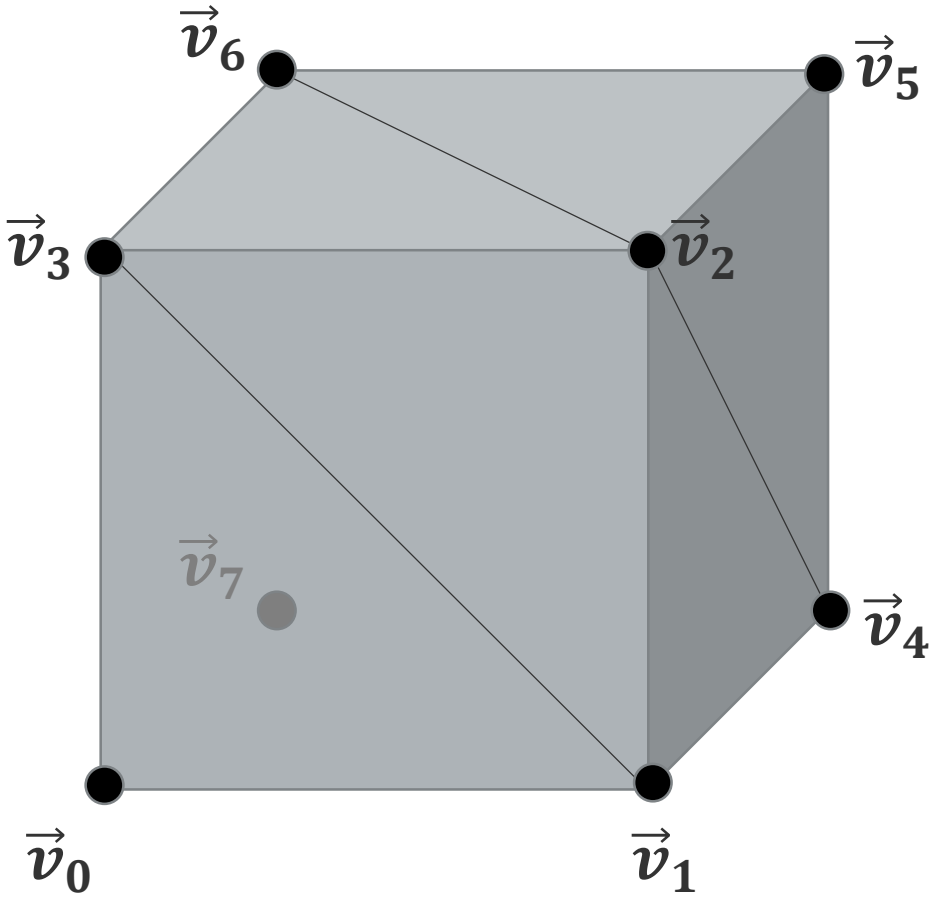


Indexed Triangle Mesh

- Separate list of vertices and triangles
- The vertex array holds the unique vertices
- The index array contains the order in which to visit those vertices to form the triangles of the mesh:

$$V: \vec{v}_0, \vec{v}_1, \vec{v}_2, \vec{v}_3, \vec{v}_4, \vec{v}_5, \vec{v}_6, \vec{v}_7$$

$$T_1: 0, 1, 3$$
$$T_2: 1, 2, 3$$
$$T_3: 3, 2, 6$$
$$T_4: 2, 5, 6$$
$$T_5: 1, 4, 2$$
$$T_6: 4, 5, 2$$
$$\vdots$$
$$T_{12}: 7, 3, 6$$

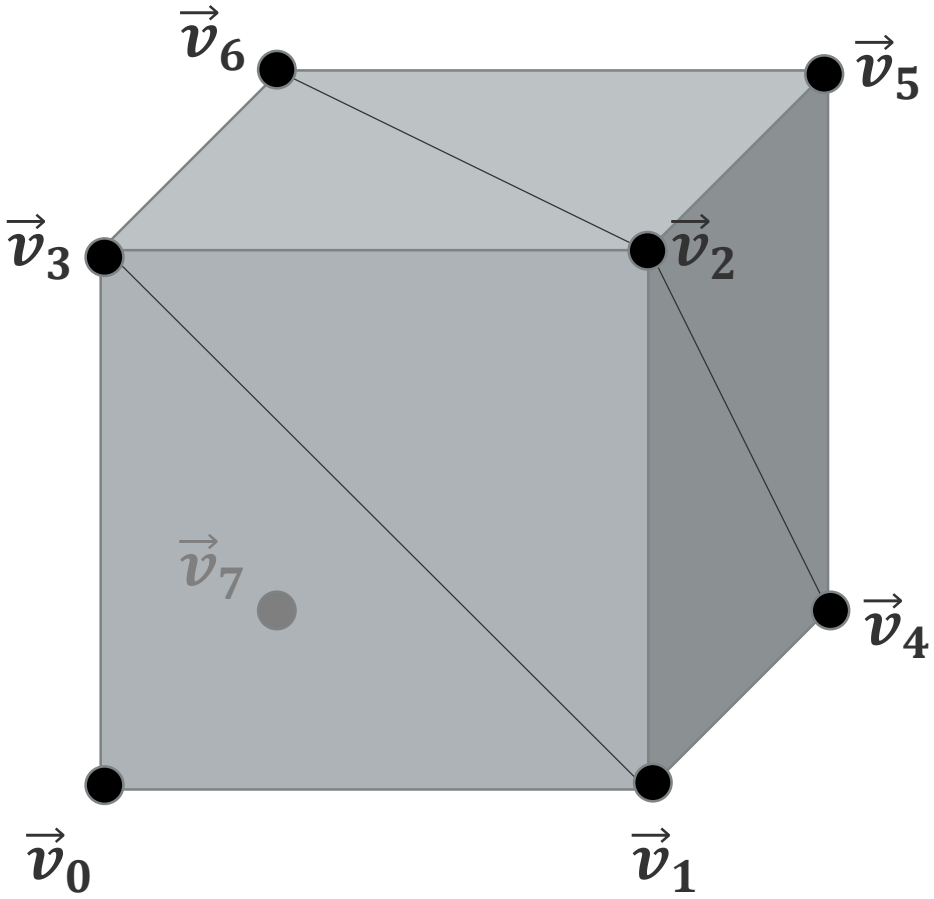


Indexed Triangle Mesh

- Separate list of vertices and triangles
- The vertex array holds the unique vertices
- The index array contains the order in which to visit those vertices to form the triangles of the mesh:

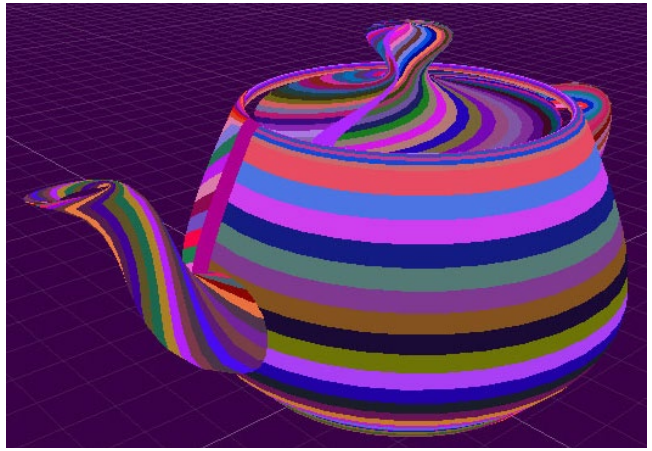
V: $\vec{v}_0, \vec{v}_1, \vec{v}_2, \vec{v}_3, \vec{v}_4, \vec{v}_5, \vec{v}_6, \vec{v}_7$

T: $0, 1, 3,$
$1, 2, 3,$
$3, 2, 6,$
$2, 5, 6,$
$1, 4, 2,$
$4, 5, 2,$
$\vdots$
$7, 3, 6$

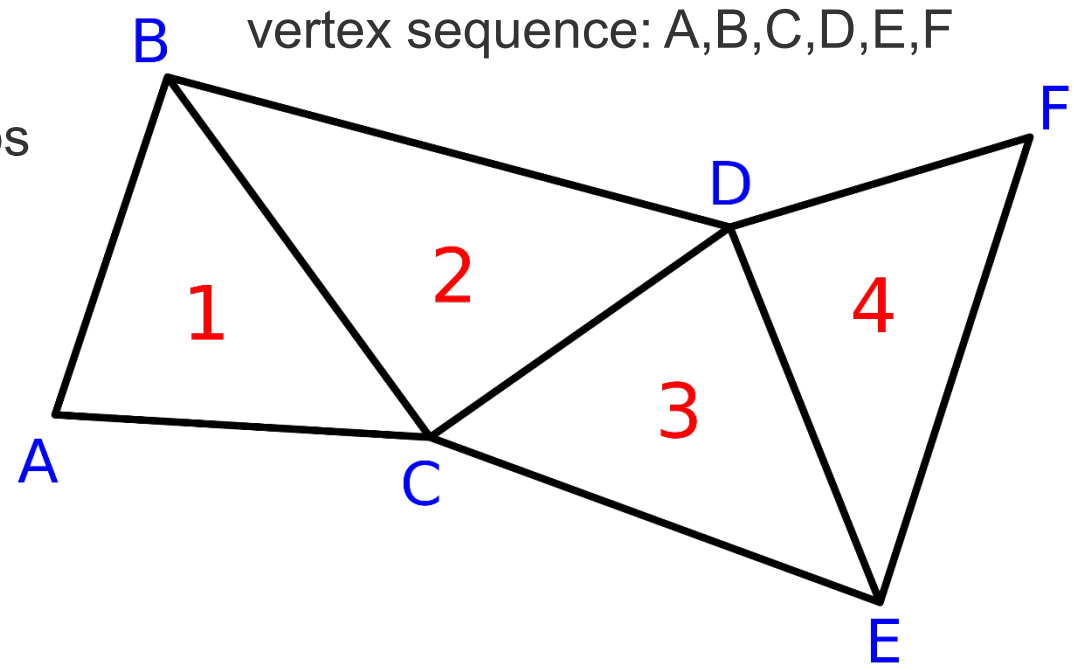


Triangle Strip

- Just emit a single new vertex per triangle
  - Reduce the memory footprint even further
- Needs N+2 points for N polygons
  - Implicit definition of the triangles
  - Optimized for graphics hardware
- A complex 3D object might require multiple strips



http://www.codercorner.com/Strips-Teapot.JPG



https://en.wikipedia.org/wiki/Triangle_strip

Image Formation

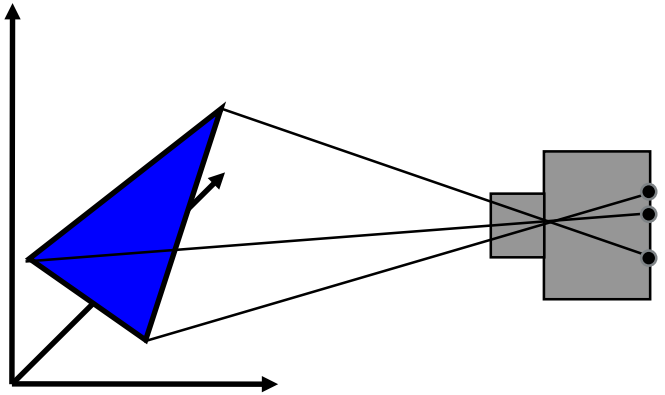
Why do images look like they do?

Rasterization vs. Ray Tracing

Image Formation: Rasterization

- Primitive operation of all interactive graphics
- Scan convert a single triangle at a time

1. Project the vertices into the image plane
2. Scanline by scanline enumerate and fill the pixels covered by the triangle



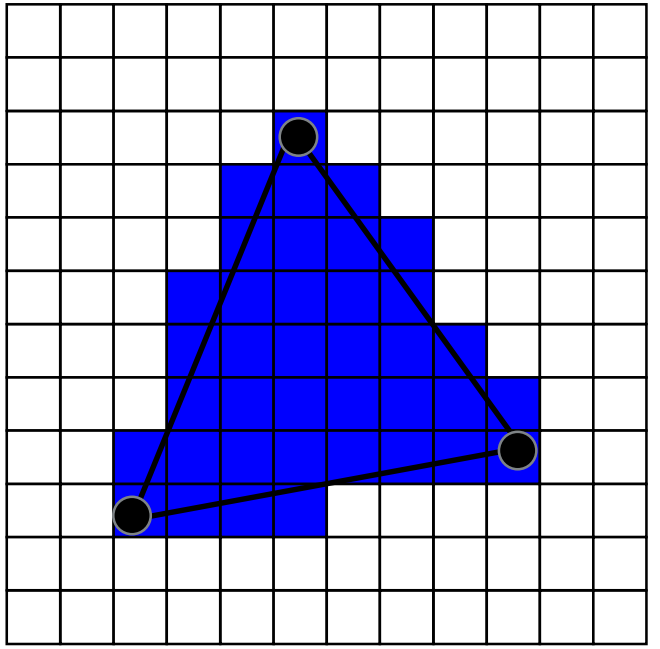
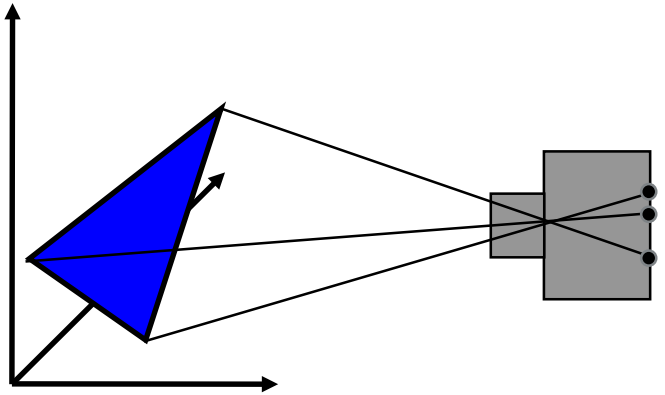
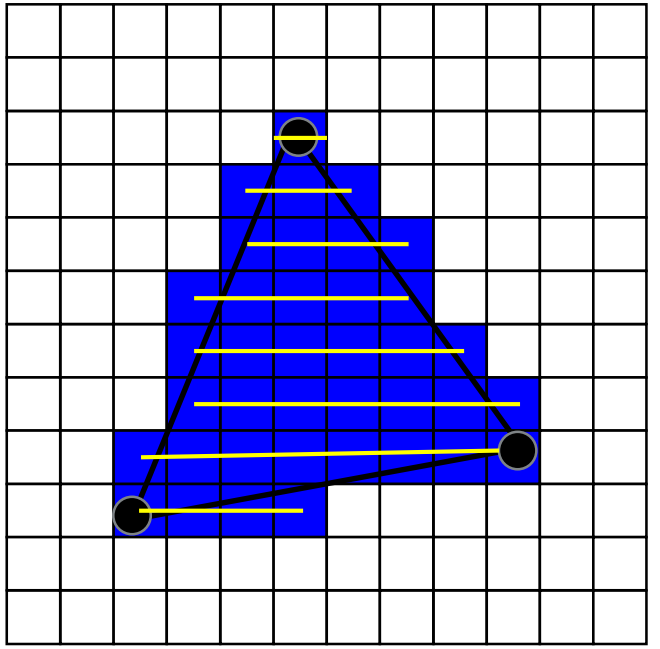


Image Formation: Rasterization

- Primitive operation of all interactive graphics
- Scan convert a single triangle at a time

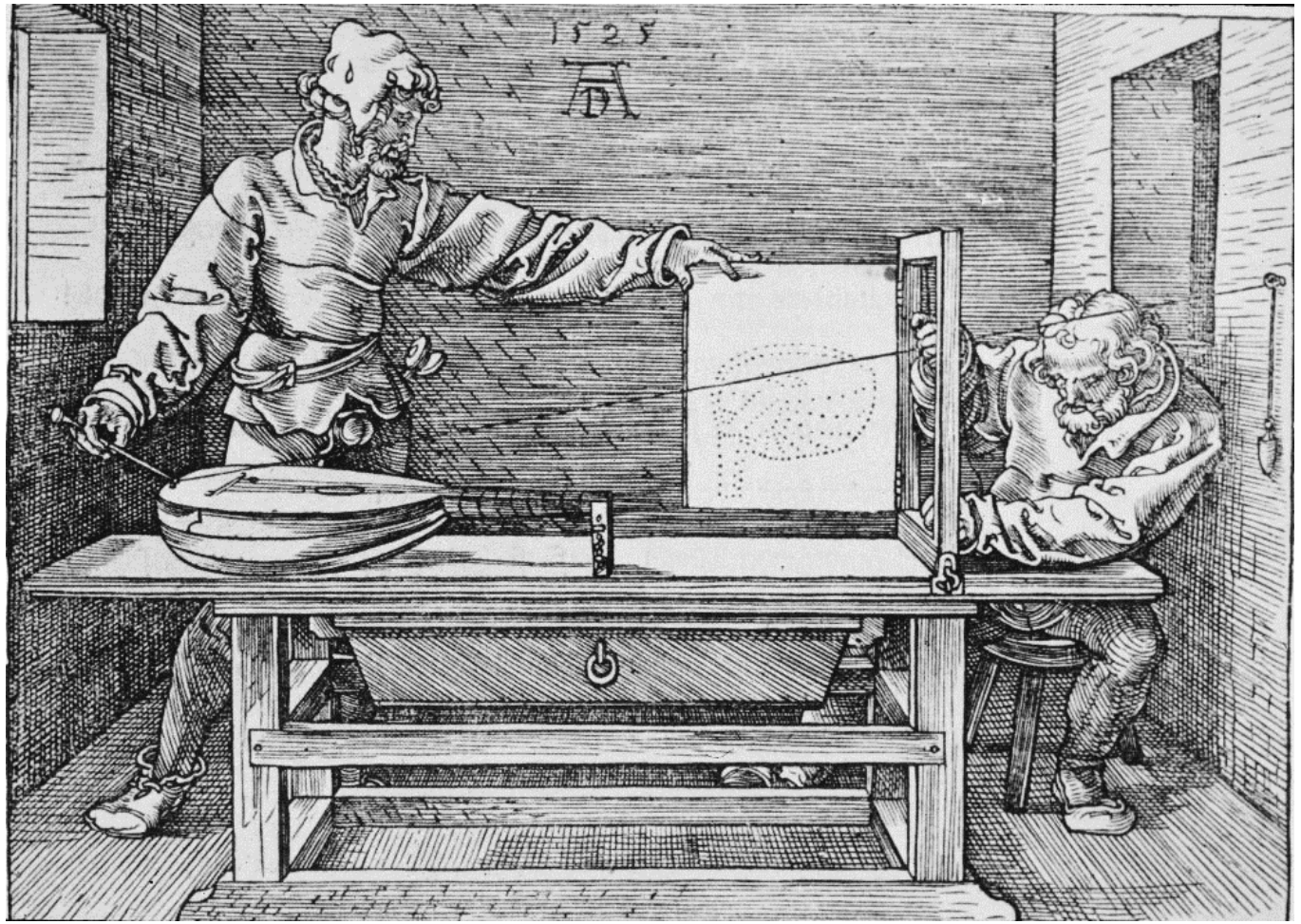
1. Project the vertices into the image plane
2. Scanline by scanline enumerate and fill the pixels covered by the triangle





Ray Tracing

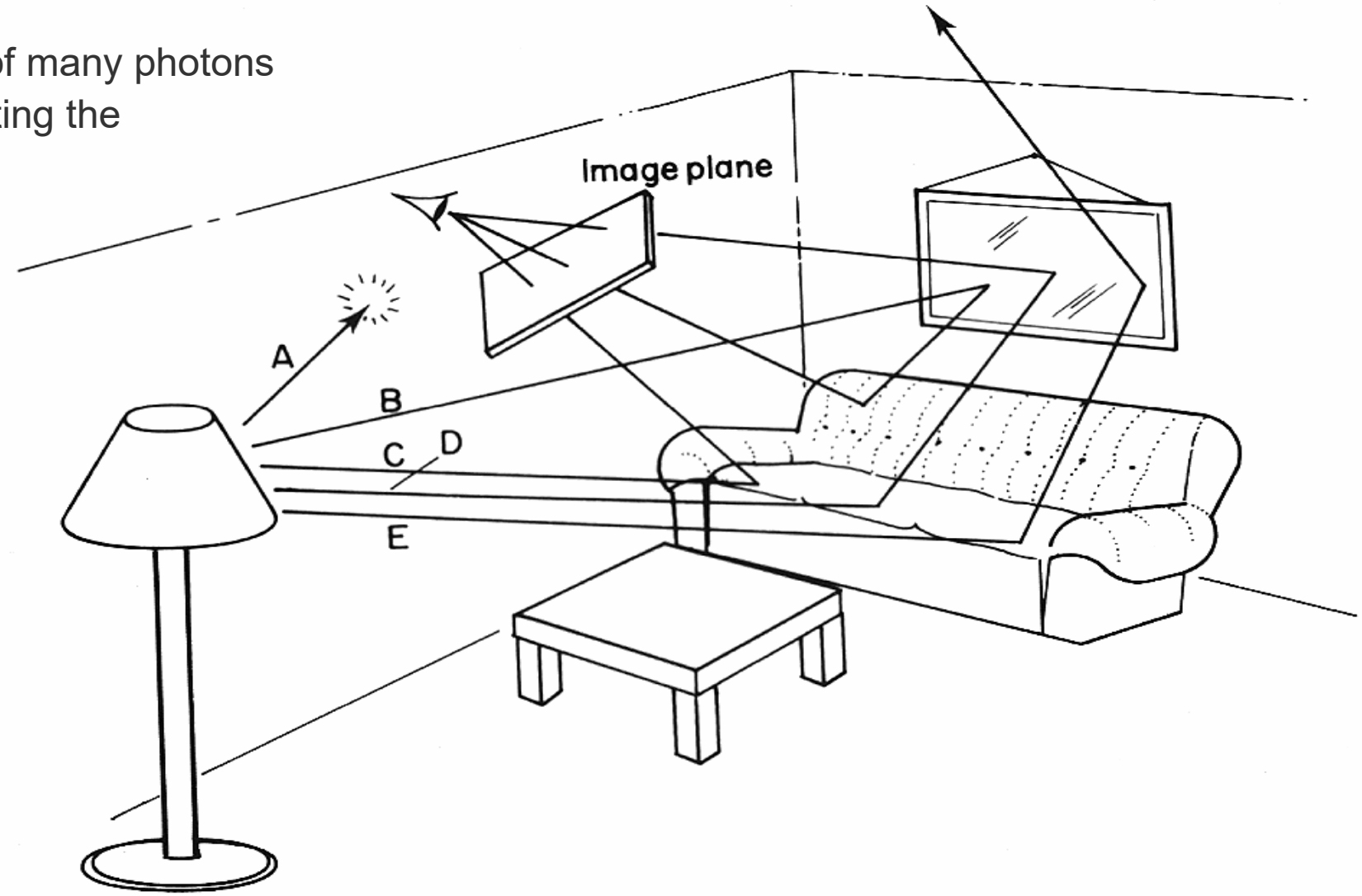
1. Send a ray through every pixel
2. Determine which object is visible, i.e. ***trace a ray*** from the camera center and compute which object is intersected



Ray tracing in a woodcut
by Albrecht Dürer, 1525

Tracing the Paths of Light

- Nature:
  - Follow the path of many photons
  - Record those hitting the film in a camera



Fundamental Ray Tracing Steps

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Generation of primary rays
  - Rays from viewpoint along viewing directions into 3D scene
  - (At least) one ray per picture element (pixel)

- Ray tracing
  - Traversal of spatial index structures
  - Intersection of ray with scene geometry

- Shading
  - From intersection, determine “light color” sent along primary ray
  - Determines “pixel color”
  - Needed
    - Local material color and reflection properties
      - Object texture
    - Local illumination of intersection point
      - Can be hard to determine correctly

Global Illumination Effects

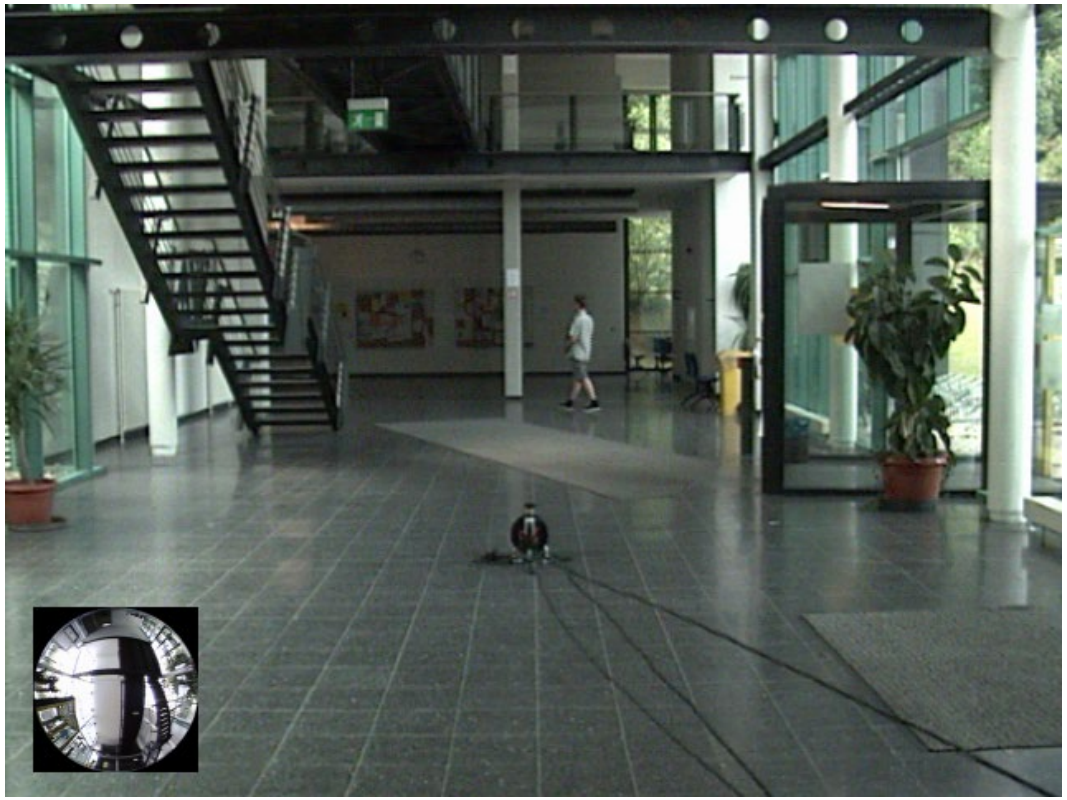
- Highly realistic images
  - Ray tracing enables *correct* simulation of light transport

Enabled by Ray Tracing

- Models Physics of Global Light Transport
  - Dependable, physically-correct visualization

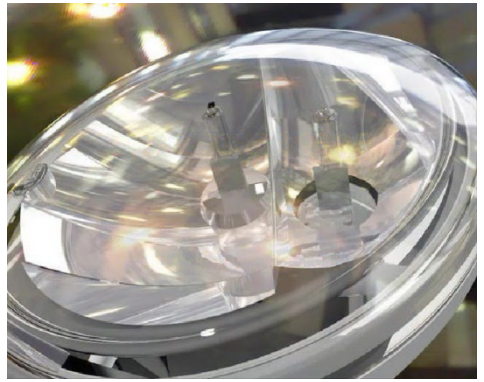
Realistic Visualization: VR/AR

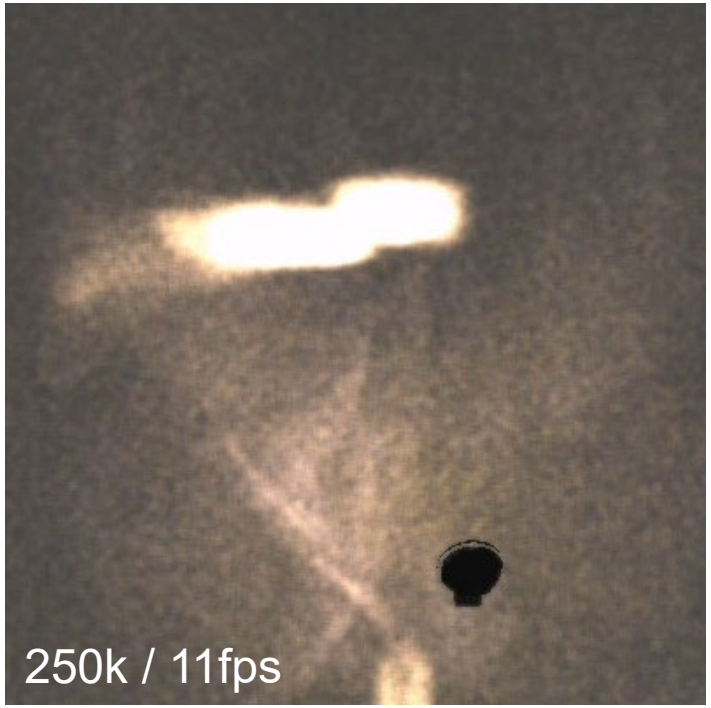
- Lighting measured from environment

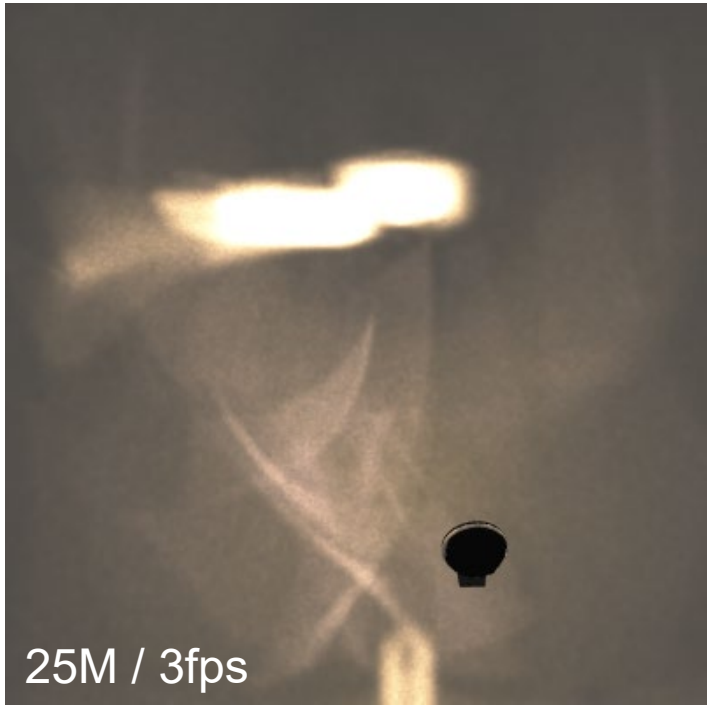


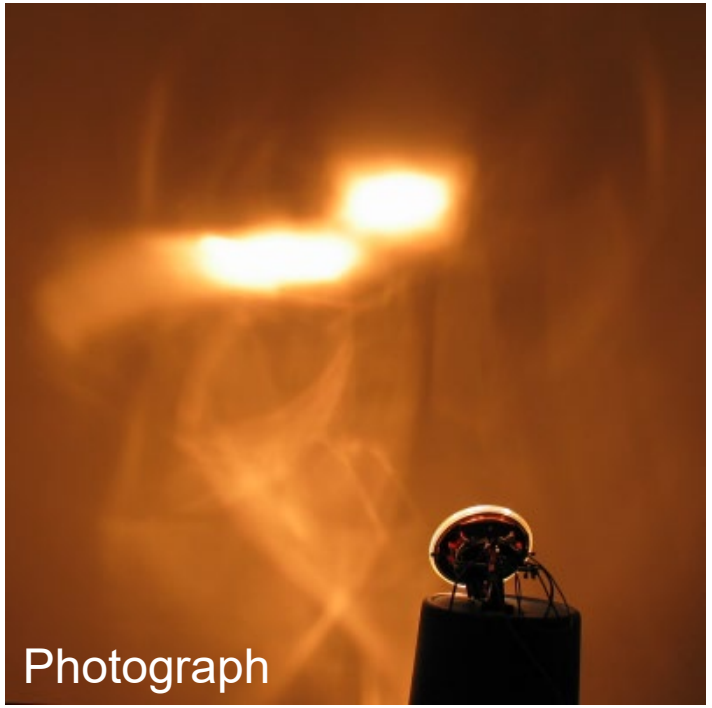
Lighting Simulation

- Complex scattering
- Highly accurate results







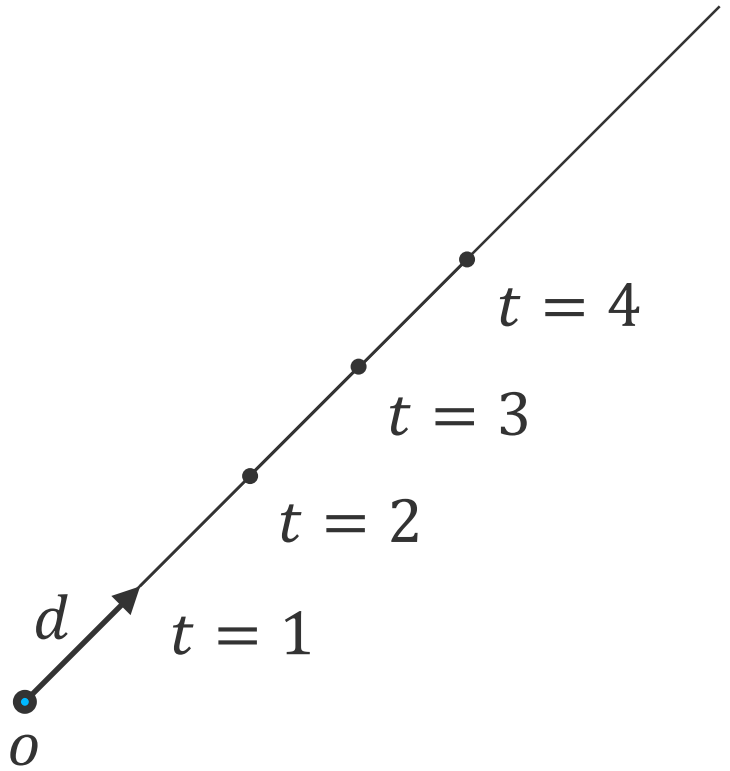


Ray Triangle Intersections

Where does the ray hit an object?

Ray and Object Representation

- Ray in space:
  - Ray equation: $\mathbf{r}(\boldsymbol{t}) = \boldsymbol{o} + \boldsymbol{td}$
  - Origin: $o = (o_x, o_y, o_z)^T$
  - Direction: $d = (d_x, d_y, d_z)^T$ - often normalized

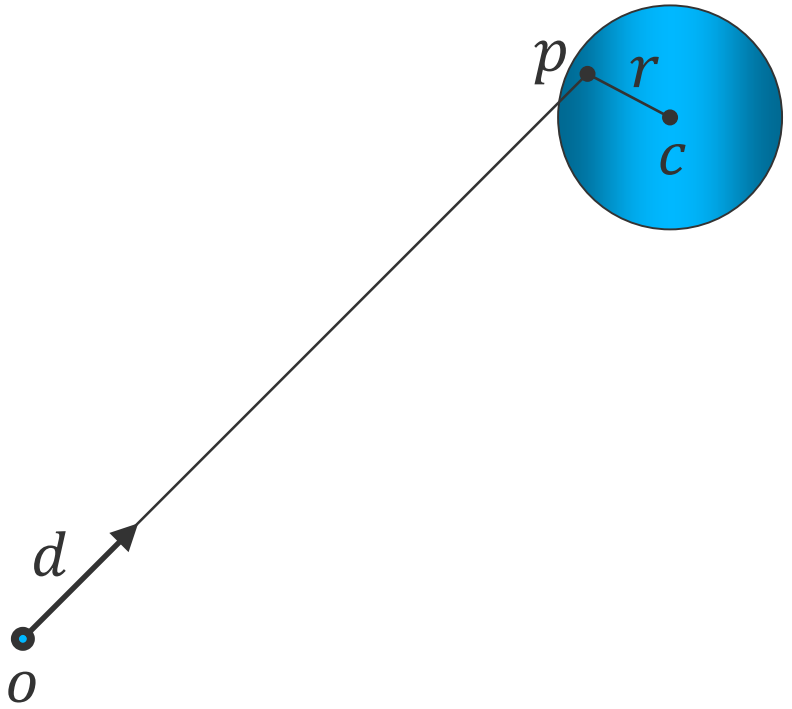


ray parameterization

Ray and Object Representation

- Ray in space: $\mathrm{r}(t) = o + td$
  - $o = (o_x, o_y, o_z)^T$
  - $d = (d_x, d_y, d_z)^T$

- Scene geometry
  - Sphere: $0 = (p - c)\cdot(p - c) - r^2$
    - c : sphere center
    - r : sphere radius
    - p : any surface point

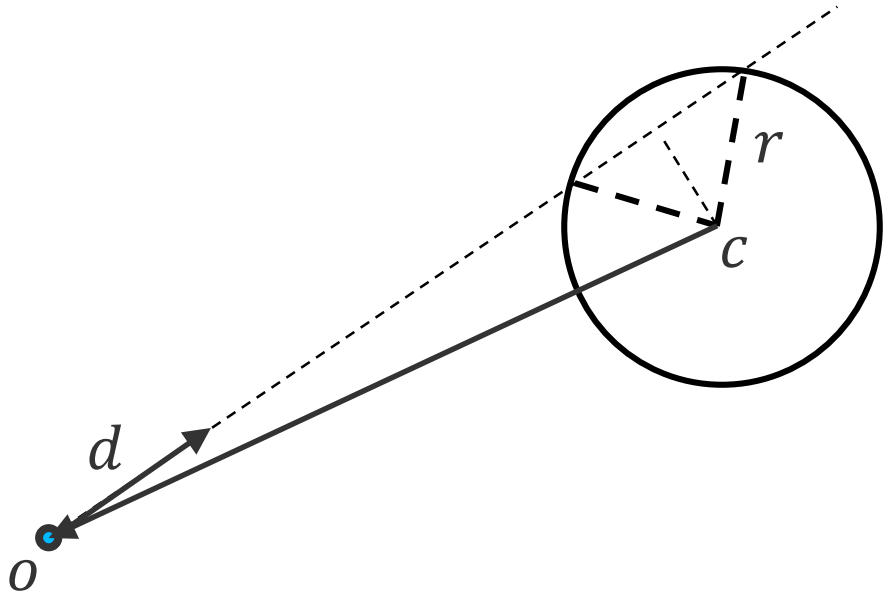


Intersection Ray – Sphere

- Sphere
  - Given a unit sphere at the origin $0 = x^2 + y^2 + z^2 - 1$
  - Given a ray $r(t) = o + td$

  - Substituting the ray into the equation for the sphere gives
    - $0 = t^2\left(d_x^2 + d_y^2 + d_z^2\right) + 2t\left(o_x d_x + o_y d_y + o_z d_z\right) + \left(o_x^2 + o_y^2 + o_z^2\right) - 1$
    - Easily solvable with standard techniques
    - But beware of numerical imprecision

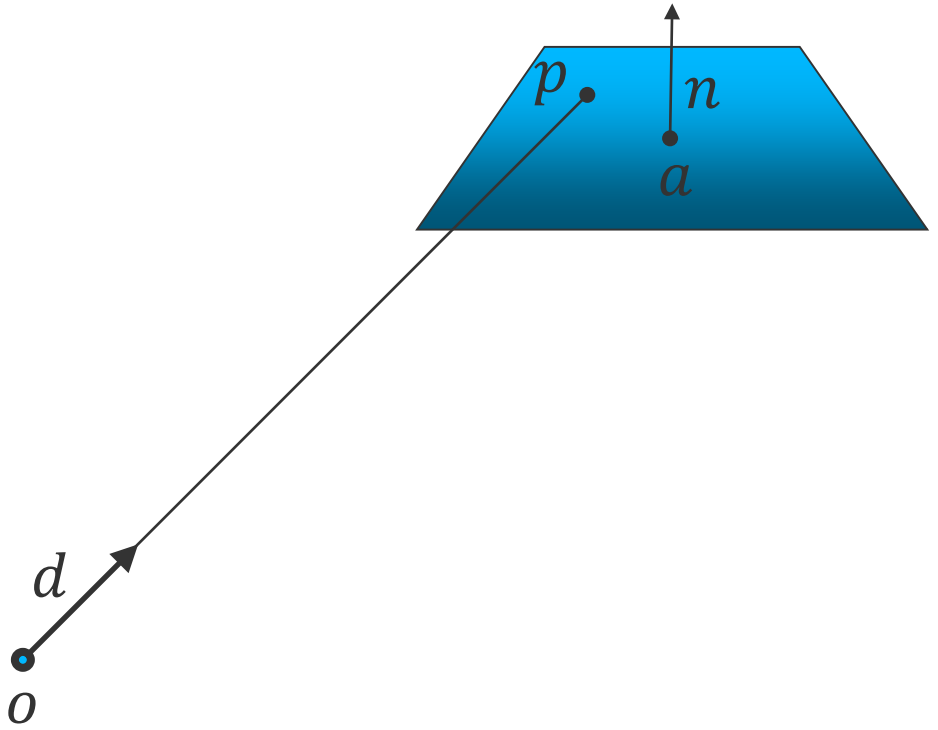
  - Alternative: Geometric construction
    - Ray and center span a plane
    - Simple 2D construction



Ray and Object Representation

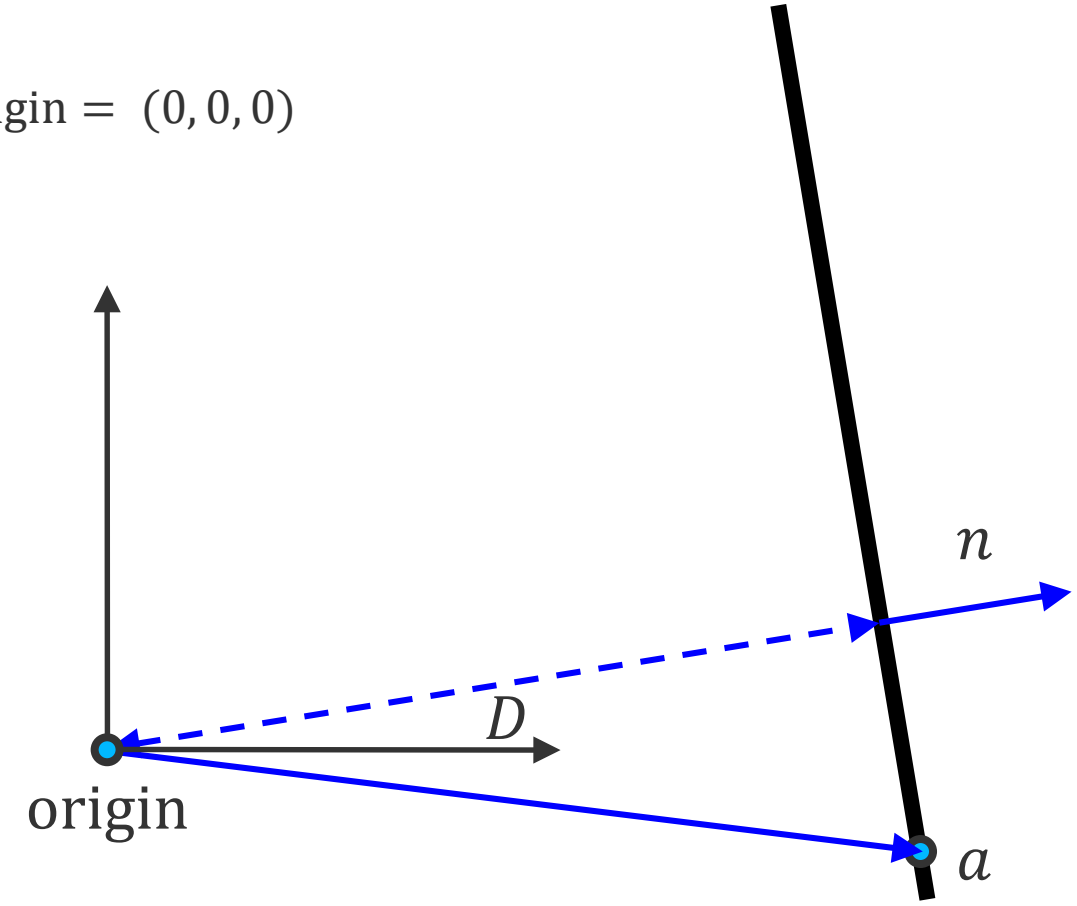
- Ray in space: $\mathrm{r}(t) = o + td$
  - $o = (o_x, o_y, o_z)^T$
  - $d = (d_x, d_y, d_z)^T$

- Scene geometry
  - Sphere: $0 = (p - c)\cdot(p - c) - r^2$
    - c : sphere center
    - r : sphere radius
    - p : any surface point
  - Plane: $0 = (p - a)\cdot n$
    - Implicit definition
    - n : surface normal
    - a : one given surface point
    - p : any surface point



Intersection Ray – Plane

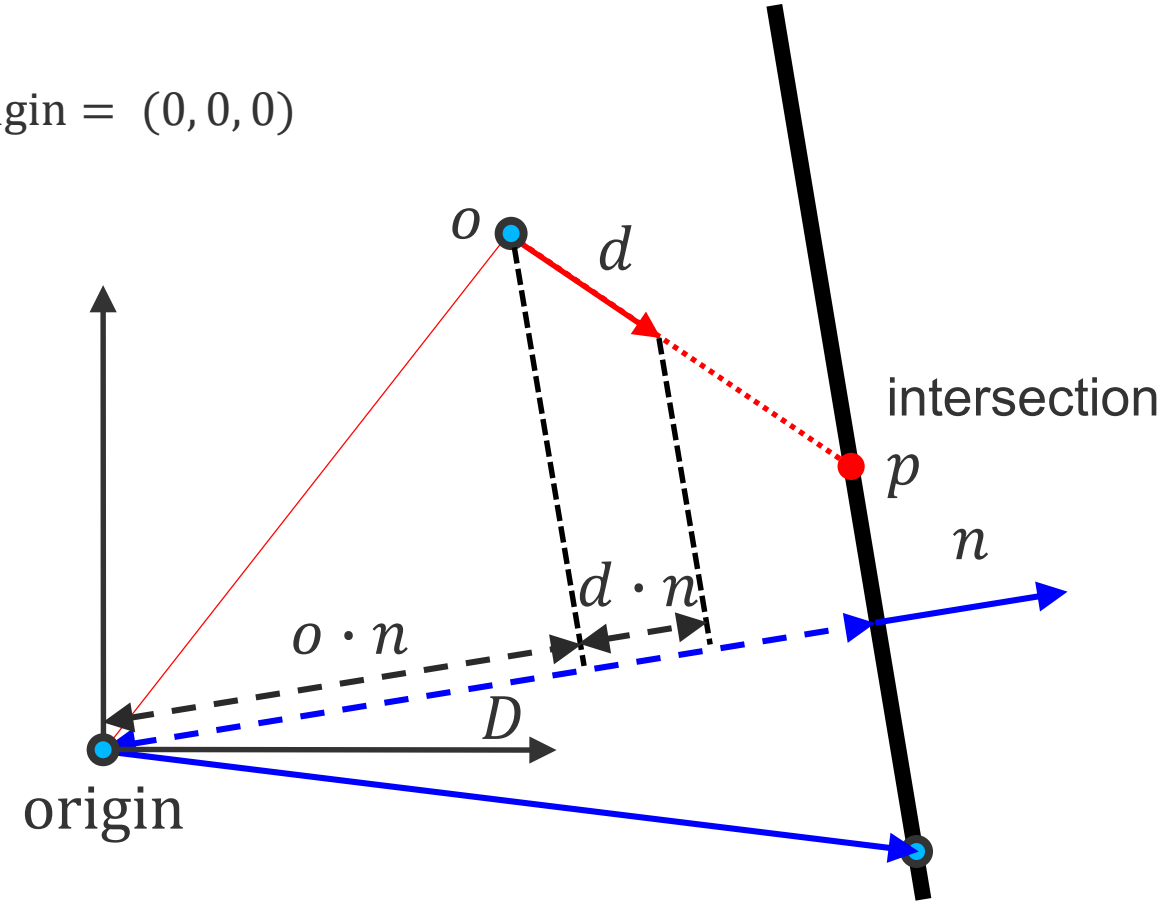
- Plane: Implicit representation (Hesse form)
  - Plane equation: $0 = p \cdot n - D$
    - n : Normal vector $|n| = 1$
    - p : Point on the plane
    - D : Normal distance of plane from origin $= (0, 0, 0)$



Intersection Ray – Plane

- Plane: Implicit representation (Hesse form)
  - Plane equation: $0 = p \cdot n - D$
    - n : Normal vector $|n| = 1$
    - p : Point on the plane
    - D : Normal distance of plane from origin $= (0, 0, 0)$

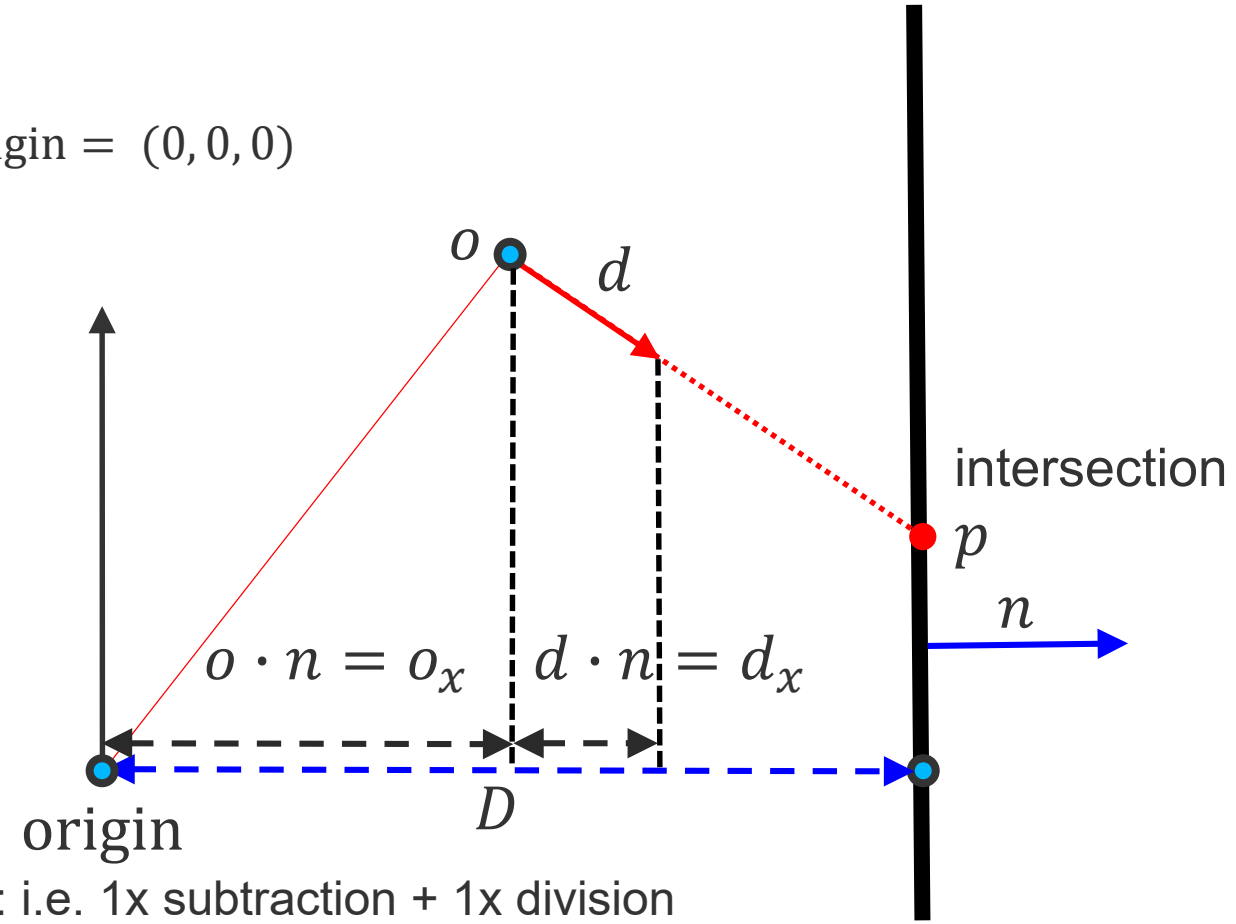
- Two possible approaches
  - Geometric
  - Algebraic
    - Substitute ray $\mathrm{r(t)} = o + td$ for p
    - $(o + td) \cdot n - D = 0$
    - Solving for t gives $t = \frac{D - o \cdot n}{d \cdot n}$



Intersection Ray – Axis-aligned Plane

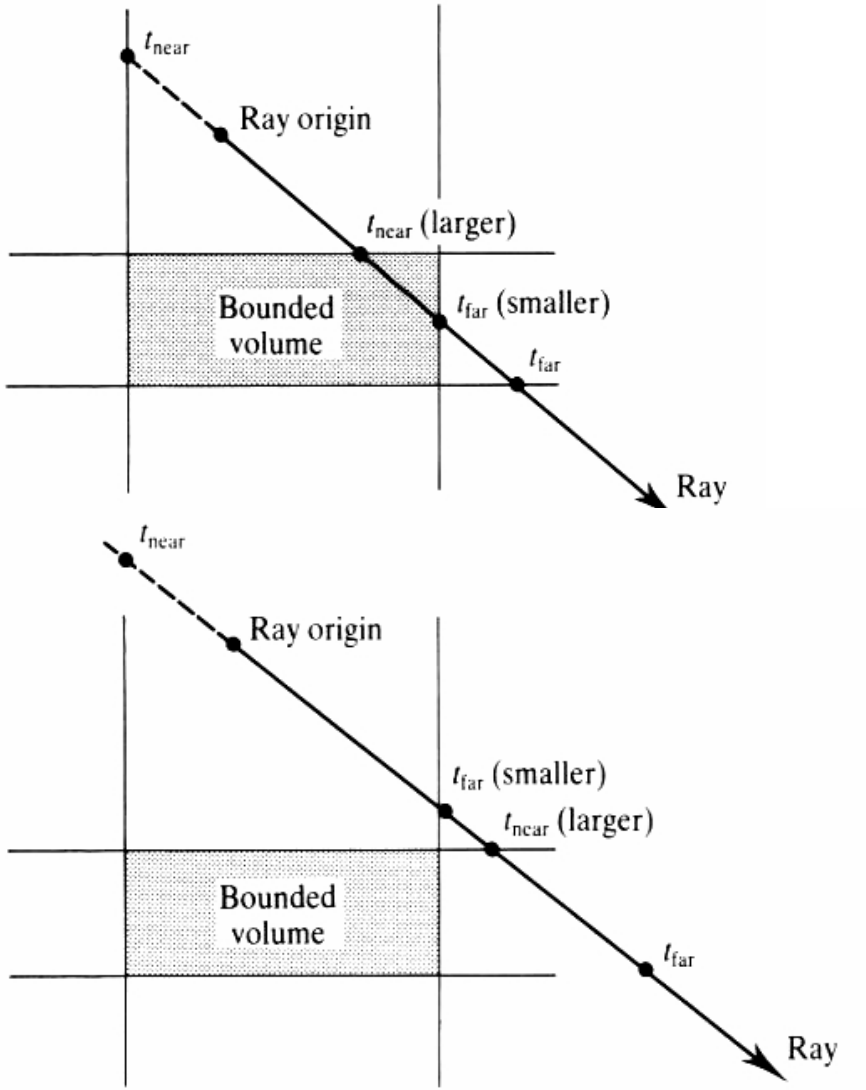
- Plane: Implicit representation (Hesse form)
  - Plane equation: $0 = p \cdot n - D$
    - n : Normal vector $|n| = 1$
    - p : Point on the plane
    - D : Normal distance of plane from origin $= (0, 0, 0)$

- Two possible approaches
  - Geometric
  - Algebraic
    - Substitute ray $\mathrm{r}(\mathrm{t}) = o + td$ for p
    - $(o + td) \cdot n - D = 0$
    - Solving for t gives $t = \frac{D - o \cdot n}{d \cdot n}$
    - Simplification for axis alignment:
      - $o \cdot n = o_x$, …
      - $t = \frac{D - o_x}{d_x}$, pure scalar computation: i.e. 1x subtraction + 1x division



Intersection Ray – Box

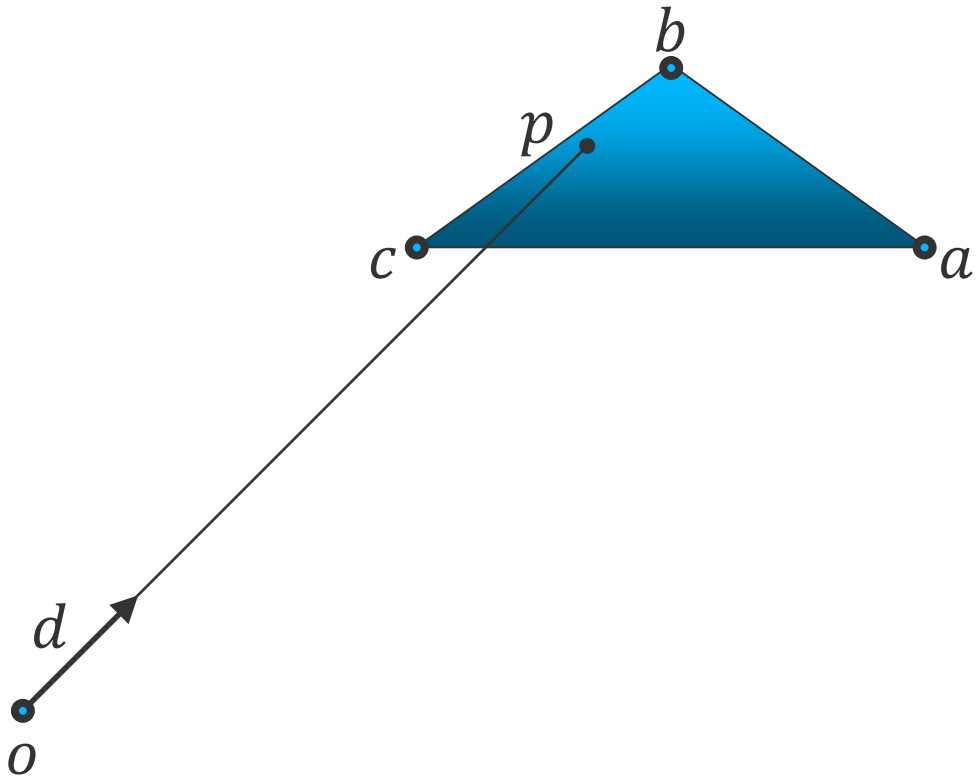
- Boxes are important for
  - Bounding volumes
  - Hierarchical structures
- Intersection test
  - test pairs of parallel planes in turn
  - calculate intersection distances
    - t_{near} first plane
    - t_{far} second plane
- The ray does not intersect the box if
  - $t_{near} > t_{far}$ for one pair of planes



Ray and Object Representation

- Ray in space: $\mathrm{r}(t) = o + td$
  - $o = (o_x, o_y, o_z)^T$
  - $d = (d_x, d_y, d_z)^T$

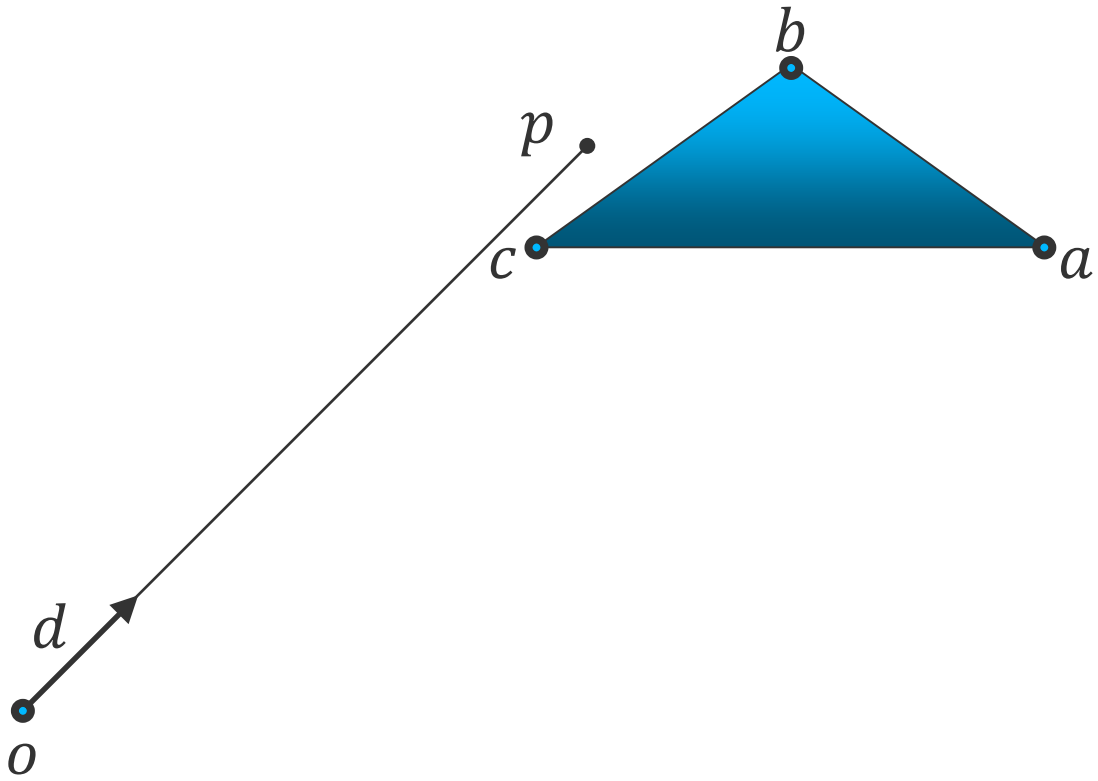
- Scene geometry
  - Sphere: $0 = (p - c)\cdot(p - c) - r^2$
    - c : sphere center
    - r : sphere radius
    - p : any surface point
  - Plane: $0 = (p - a)\cdot n$
    - Implicit definition
    - n : surface normal
    - a : one given surface point
    - p : any surface point
  - Triangle:
    - Plane intersection
      plus barycentric coordinates



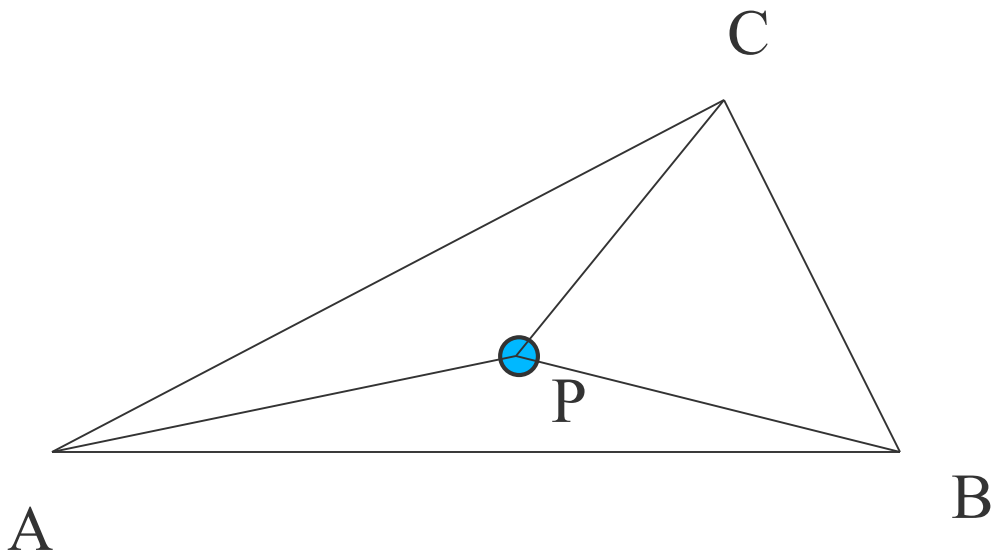
Ray and Object Representation

- Ray in space: $\mathrm{r}(t) = o + td$
  - $o = (o_x, o_y, o_z)^T$
  - $d = (d_x, d_y, d_z)^T$

- Scene geometry
  - Sphere: $0 = (p - c)\cdot(p - c) - r^2$
    - c : sphere center
    - r : sphere radius
    - p : any surface point
  - Plane: $0 = (p - a)\cdot n$
    - Implicit definition
    - n : surface normal
    - a : one given surface point
    - p : any surface point
  - Triangle:
    - Plane intersection
      plus barycentric coordinates

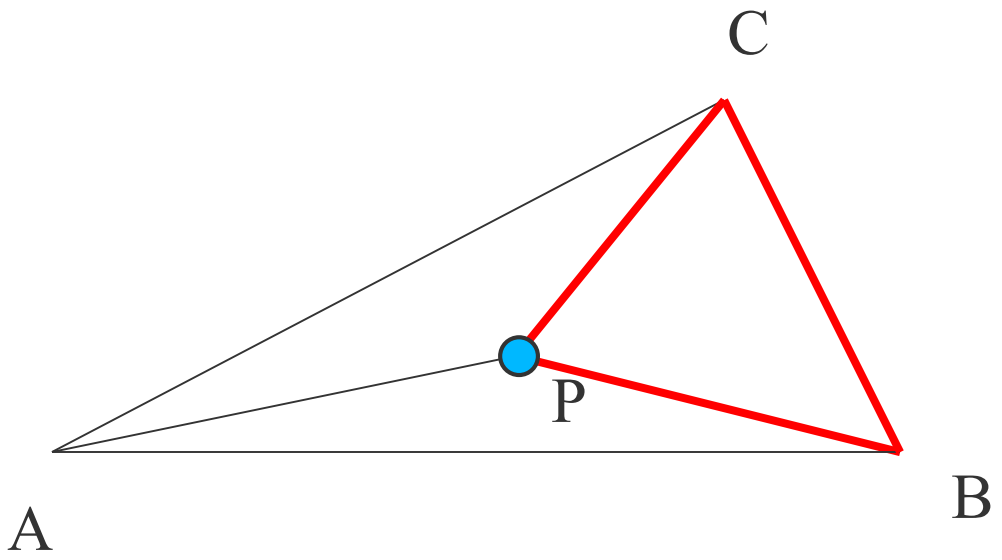


Barycentric Coordinates



$$P = \lambda_1 A + \lambda_2 B + \lambda_3 C$$

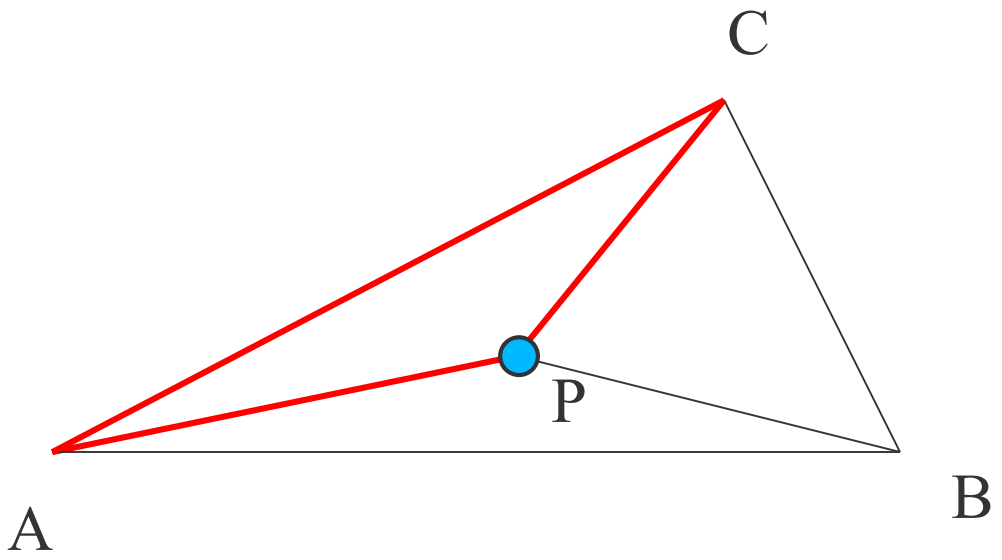
Barycentric Coordinates



$$P = \lambda_1 A + \lambda_2 B + \lambda_3 C$$

$$\lambda_1 = \frac{S_A}{S_\Delta}$$

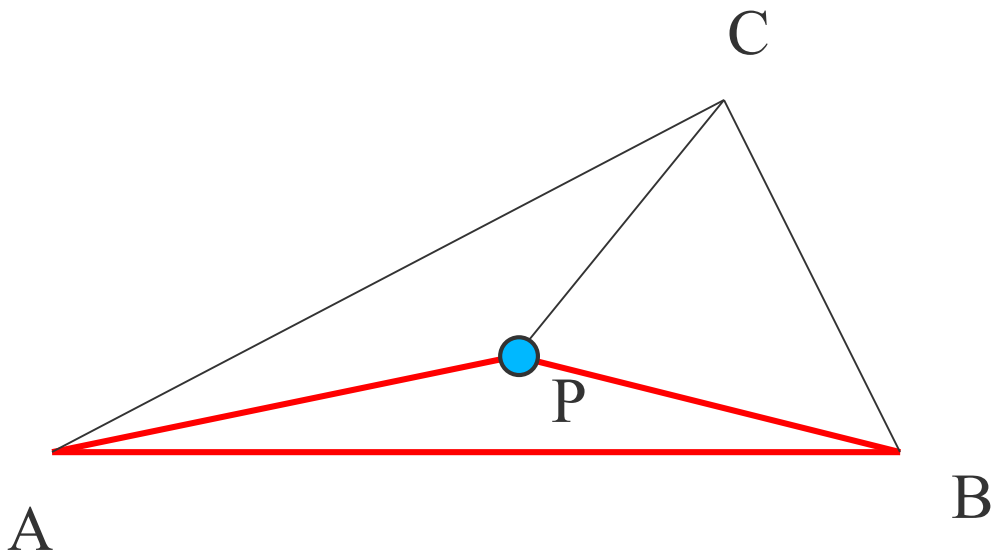
Barycentric Coordinates



$$P = \lambda_1 A + \lambda_2 B + \lambda_3 C$$

$$\lambda_2 = \frac{S_B}{S_\Delta}$$

Barycentric Coordinates

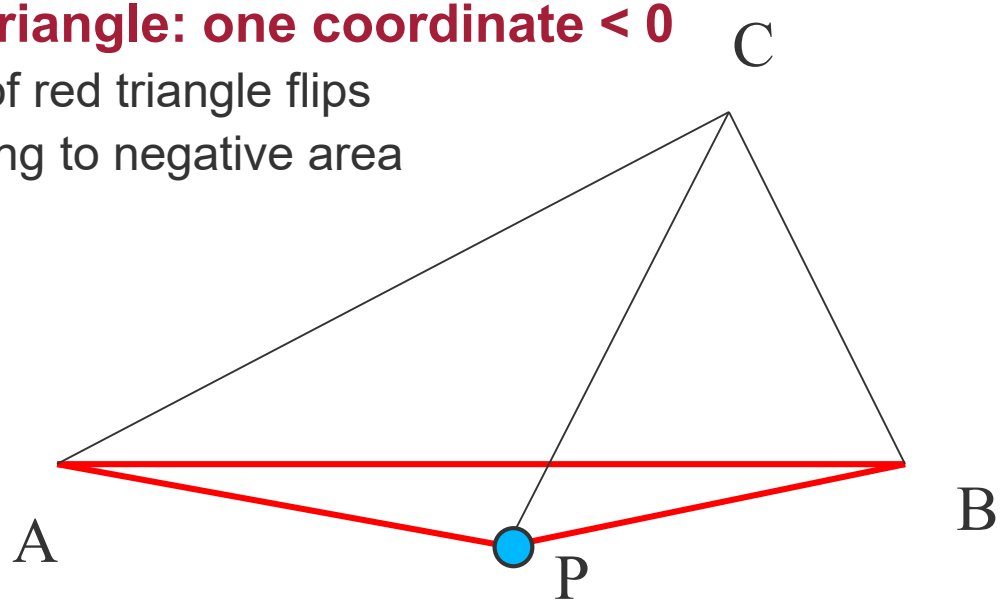


$$P = \lambda_1 A + \lambda_2 B + \lambda_3 C$$

$$\lambda_3 = \frac{S_C}{S_\Delta}$$

Barycentric Coordinates

- P **outside of triangle: one coordinate < 0**
  - Orientation of red triangle flips
  - Corresponding to negative area

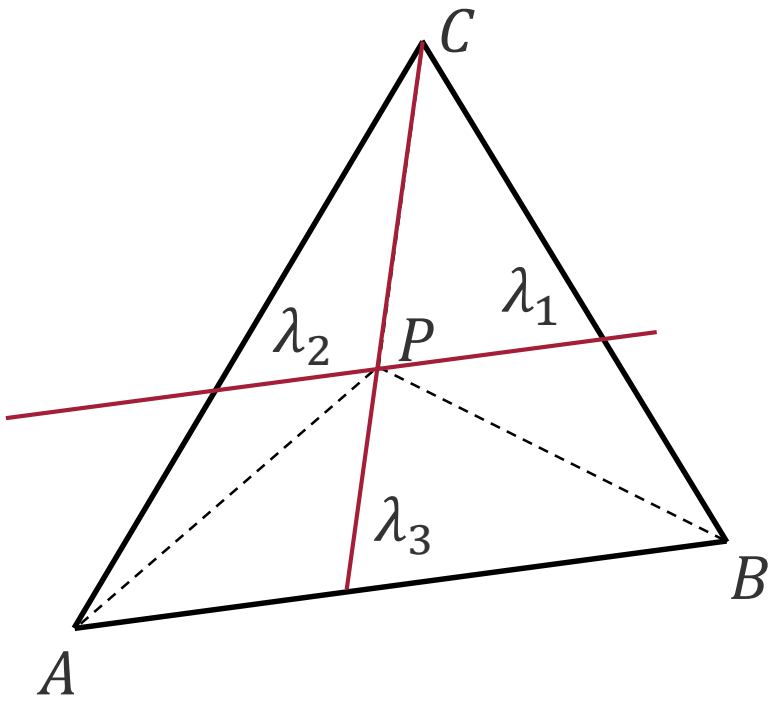


$$P = \lambda_1 A + \lambda_2 B + \lambda_3 C$$

$$\lambda_3 = \frac{S_C}{S_\Delta}$$

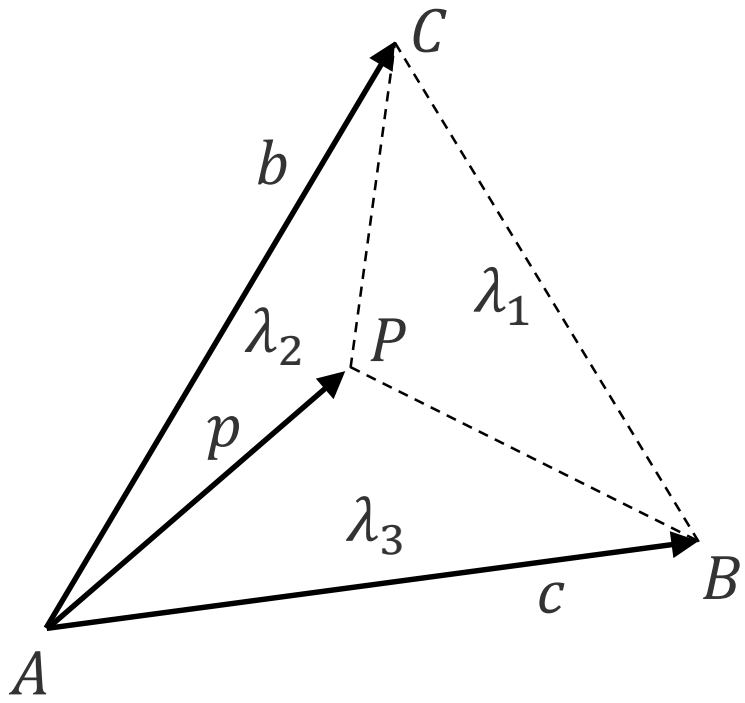
Barycentric Coordinates – Definition

- Barycentric coordinates
  - Non-degenerate triangle ABC
  - Ratio of signed areas: $\lambda_1 = \Delta(BCP)/\Delta(ABC)$
    $\lambda_2 = \Delta(CAP)/\Delta(ABC)$
    $\lambda_3 = \Delta(ABP)/\Delta(ABC)$
  - **Every point P in the plane** can be described using
    - $P = \lambda_1 A + \lambda_2 B + \lambda_3 C$
  - $\boldsymbol{\lambda_1 + \lambda_2 + \lambda_3 = 1}$
  - Examples:
    - For fixed λ_3, P may move parallel to AB
    - For $\gamma_1 + \gamma_2 = 1$ and $0 < \gamma_3 < 1$
      - $P = (1 - \gamma_3)(\gamma_1 A + \gamma_2 B) + \gamma_3 C$
      - P moves between C and AB
- Point is in triangle, iff all $0 \leq \lambda_i$



Calculation of Barycentric Coordinates 2D

- Hitpoint on 2D Plane P
  - $b = C - A$
  - $c = B - A$
  - $p = P - A$
  - $\lambda_2 = \frac{b_x p_y - b_y p_x}{b_x c_y - b_y c_x}$
  - $\lambda_3 = \frac{p_x c_y - p_y c_x}{b_x c_y - b_y c_x}$
  - $\lambda_1 = 1 - \lambda_2 - \lambda_3$

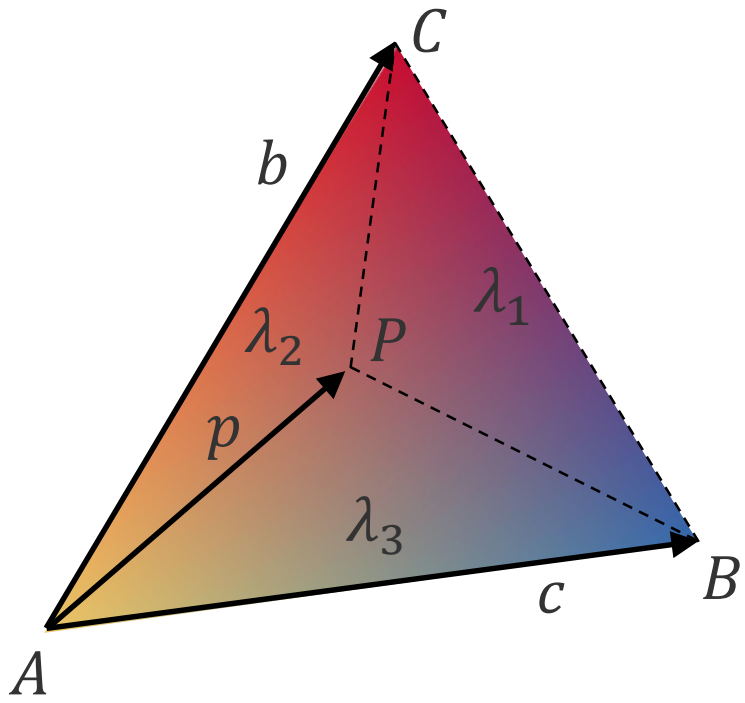


Barycenctric Coordinates – Color Interpolation

- Defining colors at the three vertices c_A, c_B, c_C
- The color at P is interpolated using the barycentric coordinates:

$$c_P = \lambda_1 c_A + \lambda_2 c_B + \lambda_3 c_C$$

- Works with any other quantity
  - Normals
  - Texture coordinates
  - …



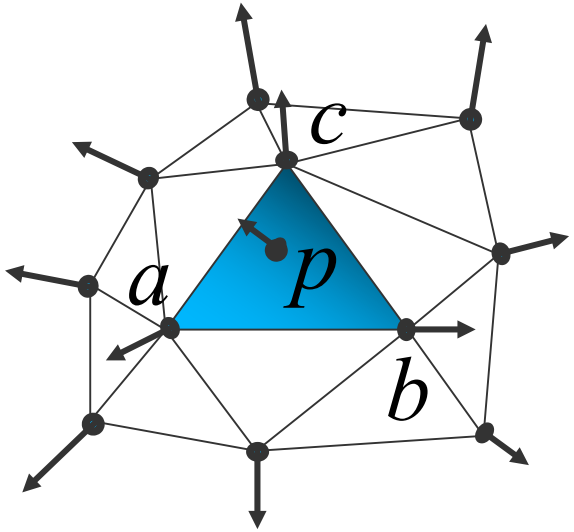
Barycentric Coordinates – Interpolating Normals

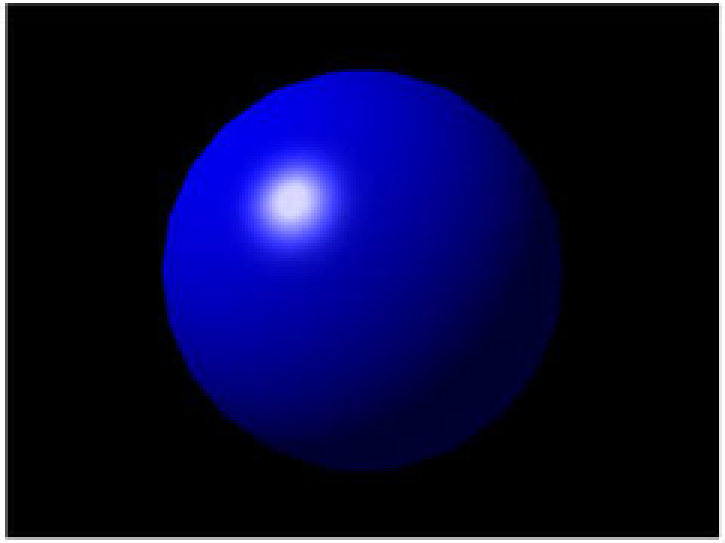
EBERHARD KARLS UNIVERSITÄT TÜBINGEN

- Per-pixel normal
  - linear interpolation of the surface normal (spherical interpolation)

$$\vec{n}_p = \frac{\lambda_1 \vec{n}_a + \lambda_2 \vec{n}_b + \lambda_3 \vec{n}_c}{||\lambda_1 \vec{n}_a + \lambda_2 \vec{n}_b + \lambda_3 \vec{n}_c||}$$

  - evaluate reflectance model at every point with interpolated normal





[wikipedia]

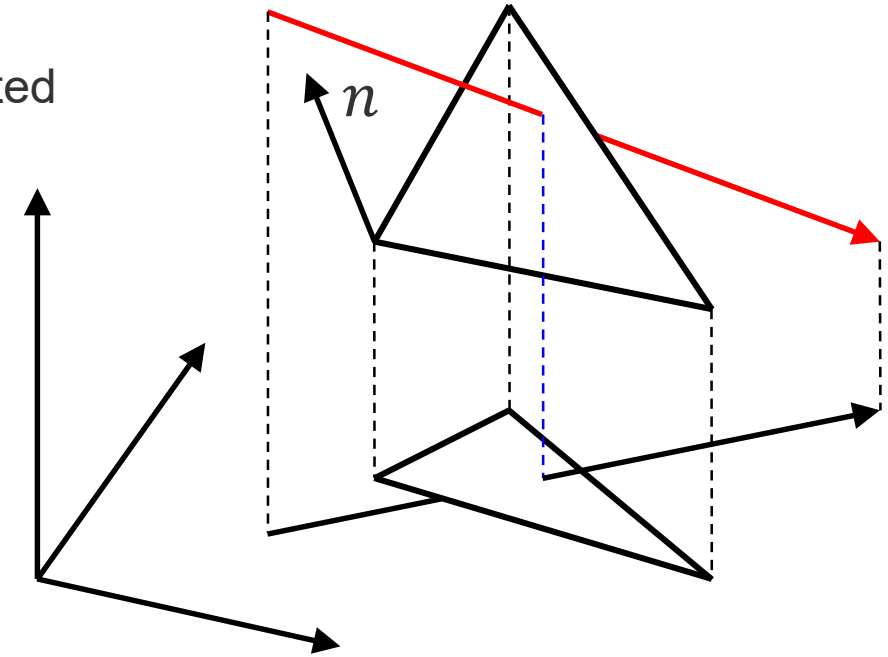
Intersection Ray – Triangle (1)

- Compute intersection with triangle plane

- Given the 3D intersection point
  - Project point into xy, xz, yz coordinate plane (ignore one of the three coordinates)
  - Use coordinate plane that is most aligned
    - xy: if n_z is maximal, etc.
  - Coordinate plane and 2D vertices can be pre-computed

- Compute barycentric coordinates in 2D

- Test for positive BCs



Intersection Ray – Triangle (1)

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

```
# calc edges and normal
b = B-A
c = C-A
n = cross(c,b) # normalize
D = dot(A,n)

# distance test
t_plane = (D-dot(o,n)) / dot(d,n)
if (t_plane < eps or t_plane > t_max):
    return NO_HIT

# determine projection dimensions
if (abs(n[0]) > abs(n[1])):
        #x                                  #z
    k = 0 if (abs(n[0]) > abs(n[2])) else 2
else:
        #y                                  #z
    k = 1 if (abs(n[1]) > abs(n[2])) else 2

u = (k+1) % 3
v = (k+2) % 3
```

```
# calculate hitpoint
p[u] = O[u] + t_plane * d[u] - A[u]
p[v] = O[v] + t_plane * d[v] - A[v]

beta = (b[u]*p[v] - b[v]*p[u]) /
       (b[u] * c[v] - b[v] * c[u])

if beta < 0:
    return NO_HIT

gamma = (c[v]*p[u]-c[u]*p[v] /
        (b[u] * c[v] - b[v] * c[u])

if gamma < 0:
    return NO_HIT

if (beta + gamma) > 1:
    return NO_HIT;

return HIT(t_plane, beta, gamma)
```

Fast Triangle Intersection

- see

Fast, Minimum Storage Ray/Triangle Intersection

Tomas Möller
Prosolvia Clarus AB
Chalmers University of Technology
E-mail: tompa@clarus.se

Ben Trumbore
Program of Computer Graphics
Cornell University
E-mail: wbt@graphics.cornell.edu

Abstract

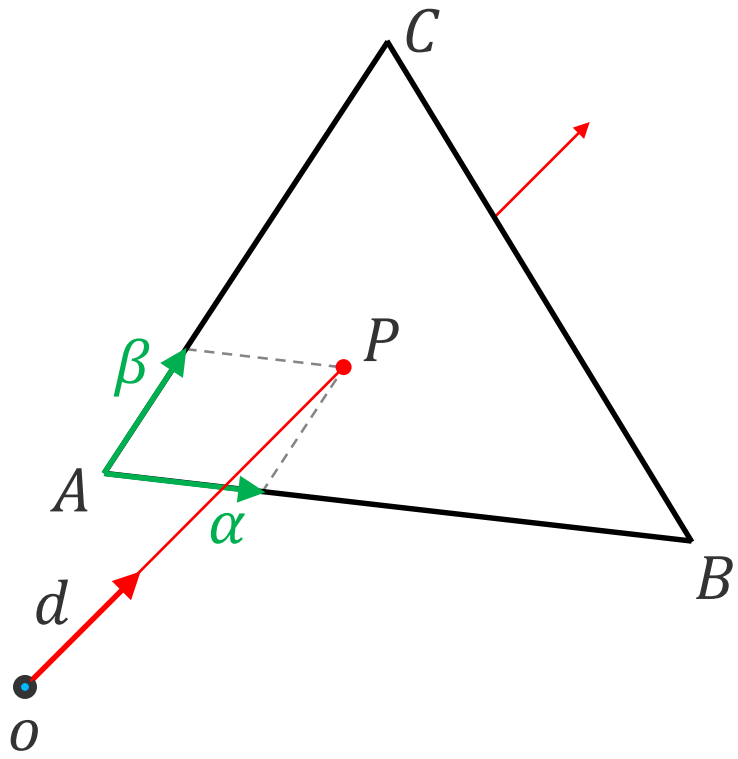
We present a clean algorithm for determining whether a ray intersects a triangle. The algorithm translates the origin of the ray and then changes the base of that vector which yields a vector $(t\ u\ v)^T$, where t is the distance to the plane in which the triangle lies and (u, v) represents the coordinates inside the triangle.

One advantage of this method is that the plane equation need not be computed on the fly nor be stored, which can amount to significant memory savings for triangle meshes. As we found our method to be comparable in speed to previous methods, we believe it is the fastest ray/triangle intersection routine for triangles which do not have precomputed plane equations.

Keywords: ray tracing, intersection, ray/triangle-intersection, base transformation.

Intersection Ray – Triangle (2)

- Check if point is inside triangle parametrically
- Compute α, β
  - $P = \alpha(B - A) + \beta(C - A)$
- Check if point inside triangle
  - $0 \leq \alpha$
  - $0 \leq \beta$
  - $0 \leq \alpha + \beta \leq 1$

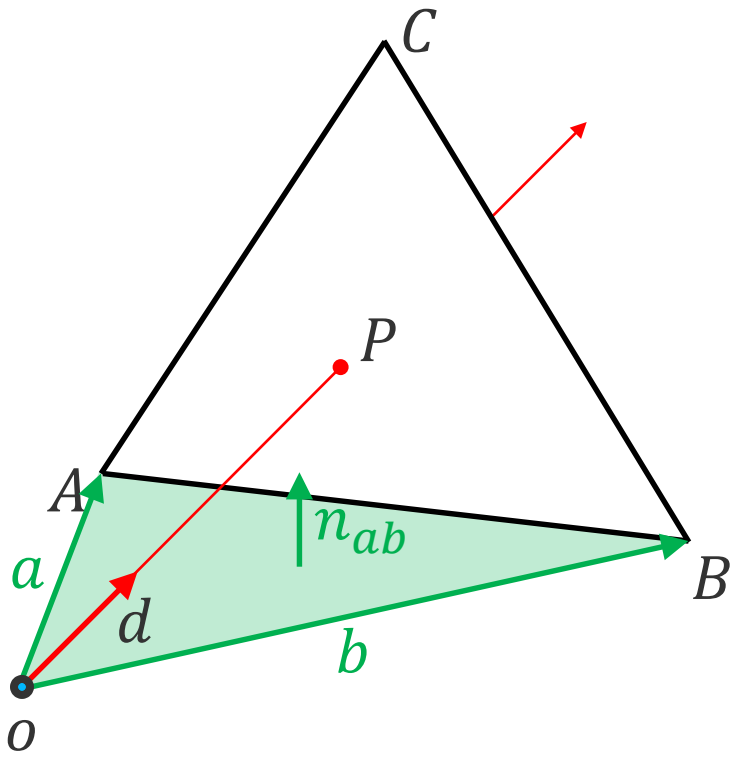


Intersection Ray – Triangle (3)

- Check if point is inside triangle algebraically

- Compute
  - $\langle a, b, c \rangle = \langle A, B, C \rangle - o$
  - $n_{ab} = \frac{a \times b}{|a \times b|}$
  - $n_{bc} = \frac{b \times c}{|b \times c|}$
  - $n_{ca} = \frac{c \times a}{|c \times a|}$

- for $i \in (ab, bc, ca)$:
  - $s_i = -o \cdot n_i$
  - if $P \cdot n_i + s_i < 0$:
    - return false



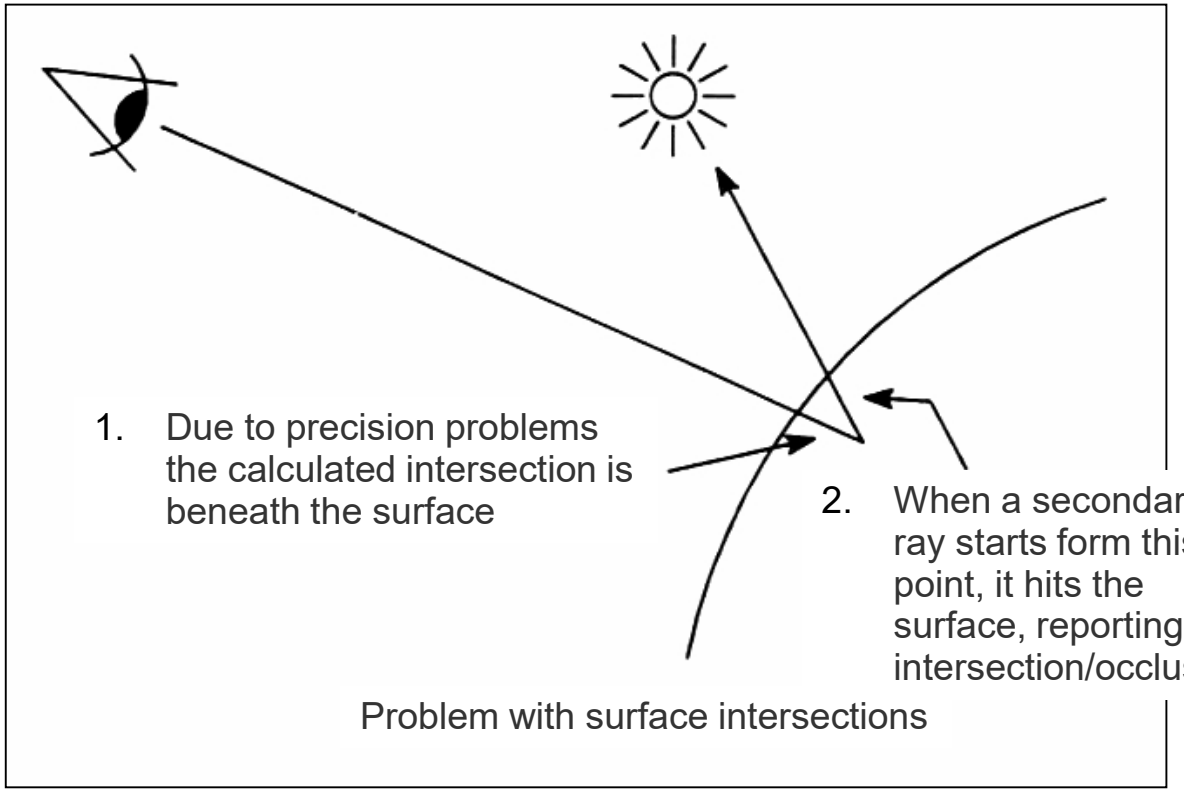
- P is in the triangle if n_{ab}, n_{bc}, n_{ca} all point towards the inside of the triangle

Precision Problems

- Due to floating point arithmetic
- Shadow rays might start just below the surface, reporting the surface itself as occluded
- Solution:
  - check if the value of parameter t is within some tolerance

    $abs(t) < \varepsilon$ with $\varepsilon = 0.000001$

    (pseudo intersection)



Problem with surface intersections

Questions

- How do you represent the surface of an object?

- How do you compute the ray intersections with:
  - Plane
  - Box
  - Triangle
  - Sphere
  - Cylinder
  - ?

- How are Barycentric Coordinates defined?
- What can you use them for?

Wrap-Up

- Basic Geometry
  - Triangle Mesh Representations
- Ray-geometry intersection calculation
  - Sphere, plane, triangle, box
- Barycentric Coordinates
  - For triangle intersection test
  - To interpolate color, normal, …
- Next lecture
  - Transformations

History of Intersection Algorithms

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

- Ray-geometry intersection algorithms
  - Polygons: [Appel ’68]
  - Quadrics, CSG: [Goldstein & Nagel ’71]
  - Recursive Ray Tracing: [Whitted ’79]
  - Tori: [Roth ’82]
  - Bicubic patches: [Whitted ’80, Kajiya ’82]
  - Algebraic surfaces: [Hanrahan ’82]
  - Swept surfaces: [Kajiya ’83, van Wijk ’84]
  - Fractals: [Kajiya ’83]
  - Deformations: [Barr ’86]
  - NURBS: [Stürzlinger ’98]
  - Subdivision surfaces: [Kobbelt et al ’98]
  - …