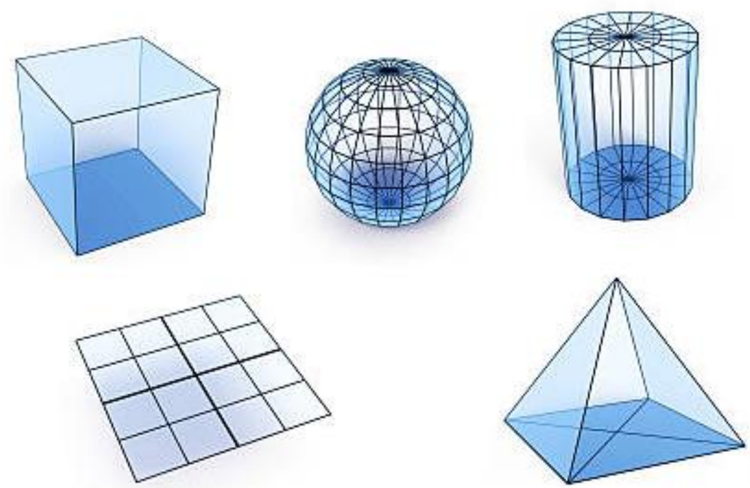
Computational geometry
V – Geometric Primitives



Stefan BORNHOFEN

What is a geometric primitive?

A *geometric primitive* is a simple geometric object, such as

- Point
- Line / Ray
- Sphere
- Plane
- Triangle

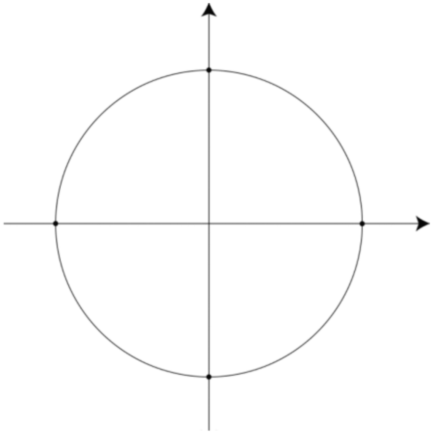
There are several common ways to represent a geometric primitive. Each representation has advantages and drawbacks.

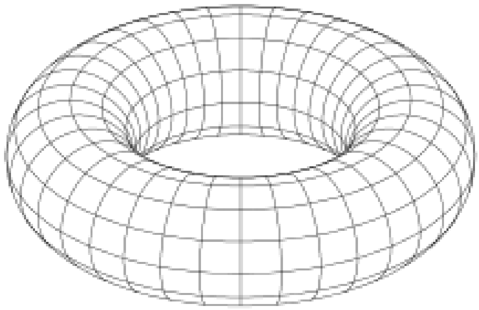
1. Implicit form
2. Parametric form
3. Direct form

Implicit Form

A boolean function which is true for all points of the primitive and false for all other points. Implicit means that the equation is not solved for x or y or z.

Implicit representations they are well suited for deciding whether a given point is on a curve, however may make it difficult to generate points of the curve.





<u>Examples</u>

$f(x,y) = x^2+y^2 = 1$

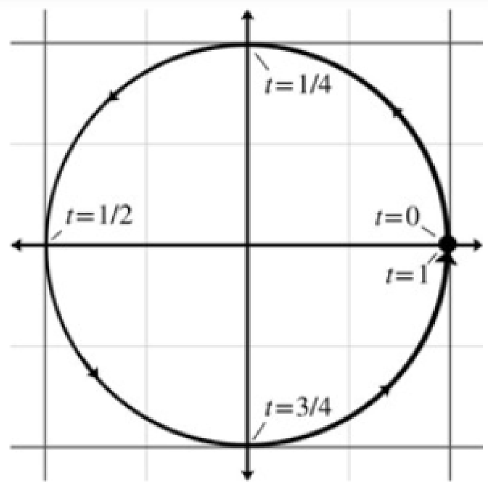
$g(x,y) = x+y = 0$

$h(x,y,z) = x^2+y^2+z^2 = 1$

$k(x,y,z) = (x^2+y^2+z^2+R^2-a^2)^2 - 4R^2(x^2+y^2) = 0$

Parametric Form

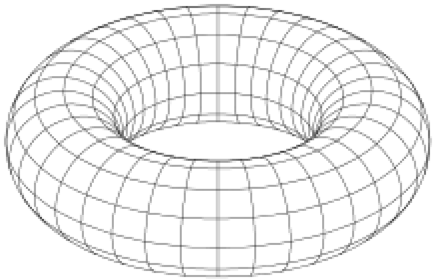
Express x,y,z as functions of other parameters that vary between certain limits. Parametric representations are best suited for generating points on a curve, and for plotting it.



<u>Examples</u>

$x(t) = \cos(2\pi t)$
$y(t) = \sin(2\pi t)$
$0 \le t \le 1$

$x(s,t)=(R+a*\cos(s))*\cos(t)$
$y(s,t)=(R+a*\cos(s))*\sin(t)$
$z(s,t)= a*\sin(s)$
$0 \le s,t \le 1$



Direct Form

A non-mathematical expression describing the most essential information of the primitive. This form is the most understandable for humans.

Some representations may use more values than others, however each primitive has an inherent number of **degrees of freedom**, i.e. the minimum amount of numbers to uniquely define the object.

<u>Examples</u>

- Sphere
  - Center and radius
- Plane
  - Three points lying on the plane
  - Normal vector and distance to origin
- Segment
  - Start point and end point
  - Start point, direction, length

Rays

- A *line* extends infinitely in two directions
- A *line segment* is a finite portion of a line with two endpoints
- In mathematics, a *ray* has an origin and extends infinitely in one direction
- In computer science, a *ray* is a directed line segment.

Rays are important geometric primitives, they are used to compute:

- Firelines and projectile trajectories
- Light rays in raytracing algorithms

Direct form

Parametric form

Rays

- A *line* extends infinitely in two directions
- A *line segment* is a finite portion of a line with two endpoints
- In mathematics, a *ray* has an origin and extends infinitely in one direction
- In computer science, a *ray* is a directed line segment.

Rays are important geometric primitives, they are used to compute:

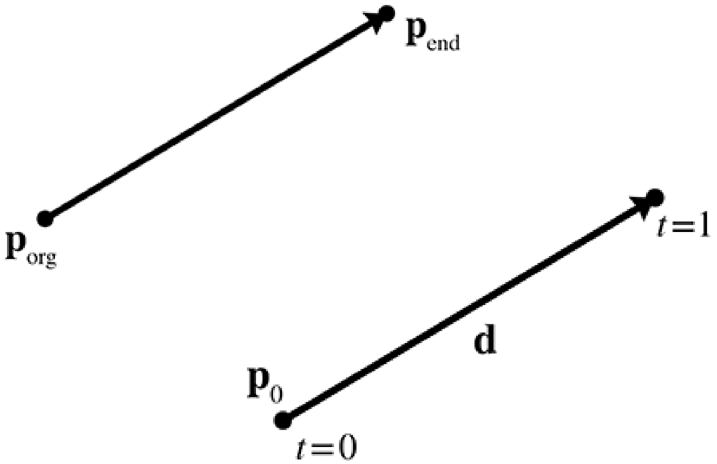
- Firelines and projectile trajectories
- Light rays in raytracing algorithms

Direct form

Two points: origin and end

Parametric form

$\mathbf{p}(t) = \mathbf{p}_0 + t\mathbf{d}$, $0 \leq t \leq 1$

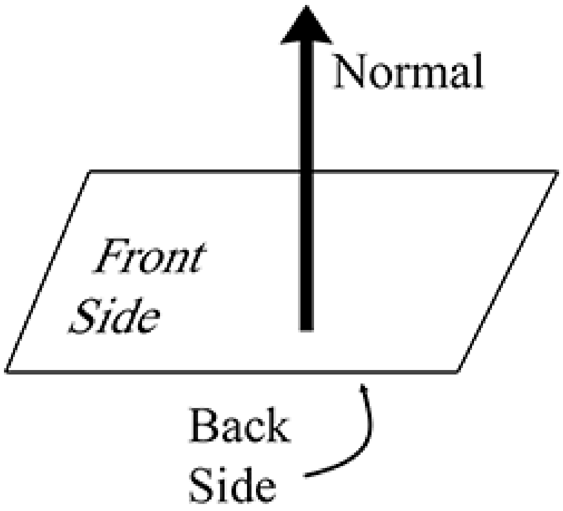


Planes

A plane in 3D is the set of points equidistant from two given points. A plane is perfectly flat, has no thickness, and extends infinitely. Planes are very common primitives to model the ground in a 3D simulation or in a video game.

Direct form

Implicit form



Parametric form

Planes

A plane in 3D is the set of points equidistant from two given points. A plane is perfectly flat, has no thickness, and extends infinitely. Planes are very common primitives to model the ground in a 3D simulation or in a video game.

Direct form

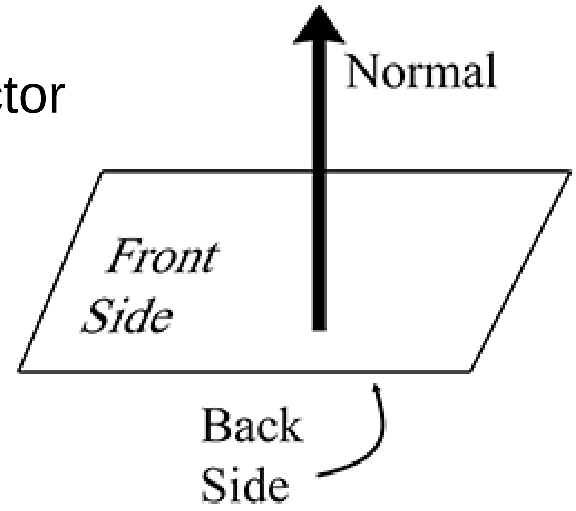
A distance d from the origin, and a normal vector n with length 1

Implicit form

$f(p) = p \cdot n = d$

$f(x,y,z) = ax + by + cz = d$, $n=(a,b,c)$ unit vector

The “front” of the plane is the side that n points away from.



Parametric form $p(\lambda,\mu) = a + \lambda v + \mu w$

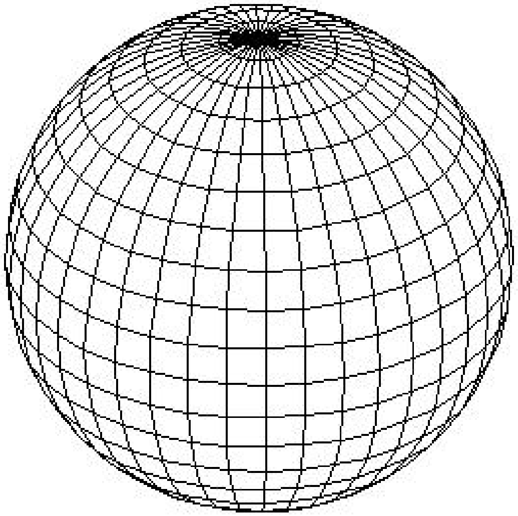
$$\begin{aligned} x &= a_1 + \lambda \cdot v_1 + \mu \cdot w_1 \\ y &= a_2 + \lambda \cdot v_2 + \mu \cdot w_2 \\ z &= a_3 + \lambda \cdot v_3 + \mu \cdot w_3 \end{aligned}$$

Spheres

A *bounding sphere* can be used in collision detection for fast rejection because intersections with a sphere are easy to compute. Moreover, rotating a sphere does not change its shape, so a bounding sphere is immune to rotation.

Direct form

Implicit form



Parametric form

Spheres

A *bounding sphere* can be used in collision detection for fast rejection because intersections with a sphere are easy to compute. Moreover, rotating a sphere does not change its shape, so a bounding sphere is immune to rotation.

Direct form

A center c and a radius r

Implicit form

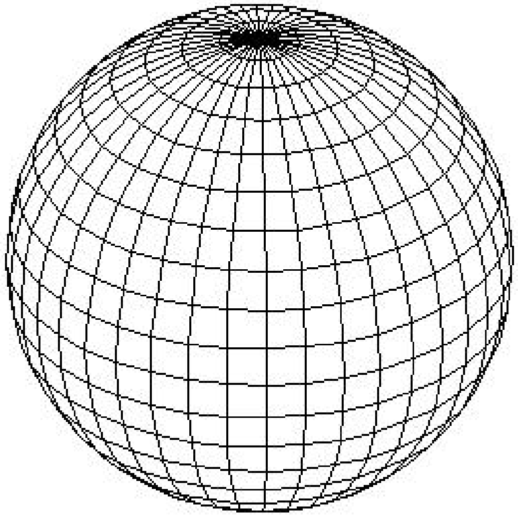
f(p) = ||p - c|| = r

$$(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 = r^2$$

Parametric form

$$\begin{aligned} x &= \sin\phi\cos\theta \\ y &= \sin\phi\sin\theta \\ z &= \cos\phi \end{aligned}$$

$$0 \le \phi \le \pi, \quad 0 \le \theta \le 2\pi$$

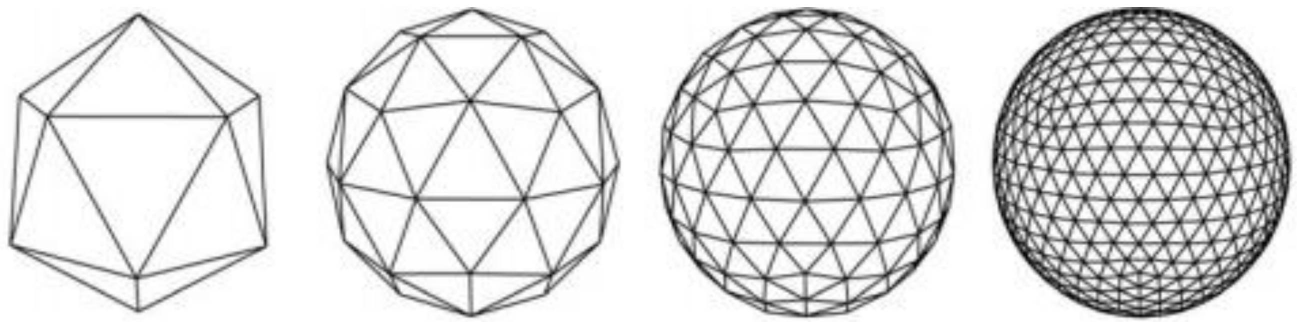


Triangles

Triangles are of fundamental importance in modeling and graphics. Triangles are the simplest possible polygon and the building blocks in 3D computer graphics.

In particular:

- The vertices are always coplanar
- The edges do not cross
- The surface is convex



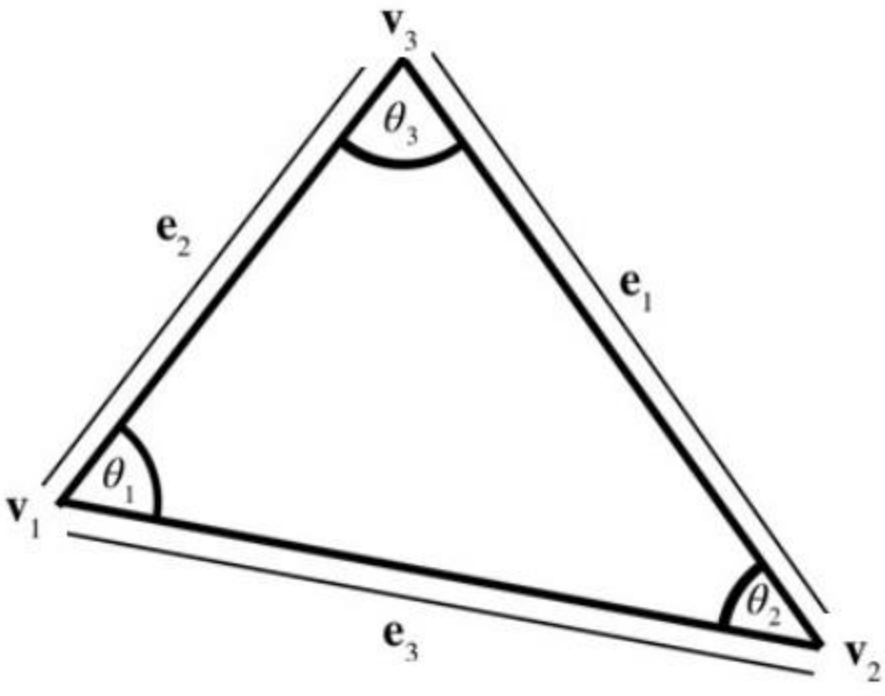
Triangles

Triangles are defined by three vertices $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ orderer couter-clockwise in a right-handed system.

$$\mathbf{e}_1 = \mathbf{v}_3 - \mathbf{v}_2,$$
$$l_1 = \|\mathbf{e}_1\|,$$

$$\mathbf{e}_2 = \mathbf{v}_1 - \mathbf{v}_3,$$
$$l_2 = \|\mathbf{e}_2\|,$$

$$\mathbf{e}_3 = \mathbf{v}_2 - \mathbf{v}_1,$$
$$l_3 = \|\mathbf{e}_3\|.$$



Triangles

Law of Sines

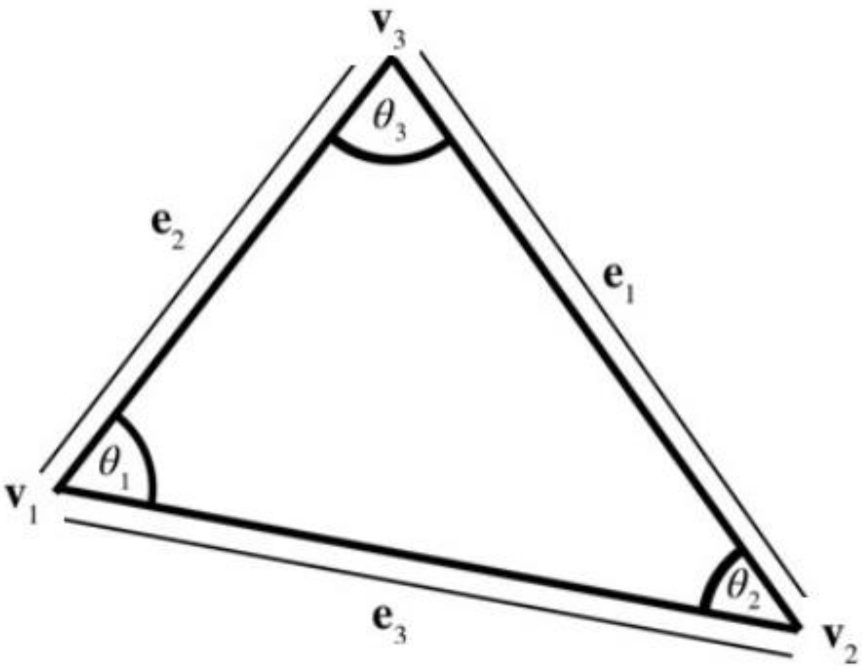
$$\frac{\sin\theta_1}{l_1} = \frac{\sin\theta_2}{l_2} = \frac{\sin\theta_3}{l_3}$$

Law of Cosines

$$l_1{}^2 = l_2{}^2 + l_3{}^2 - 2l_2l_3\cos\theta_1$$

$$l_2{}^2 = l_1{}^2 + l_3{}^2 - 2l_1l_3\cos\theta_2$$

$$l_3{}^2 = l_1{}^2 + l_2{}^2 - 2l_1l_2\cos\theta_3$$



Triangles

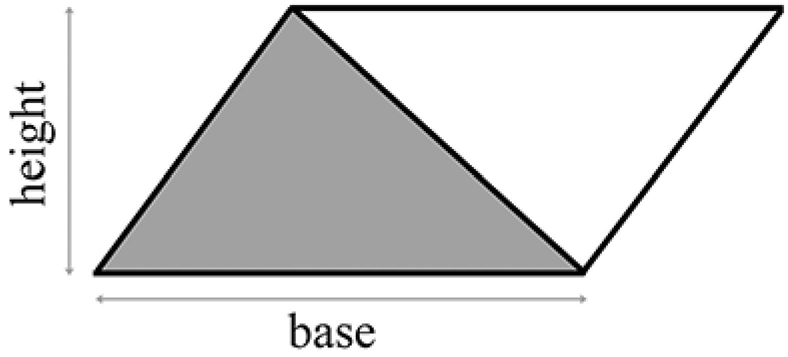
Perimeter $p = l_1 + l_2 + l_3$

Surface

$A = b*h / 2$

Equivalent for triangles in 3D space:

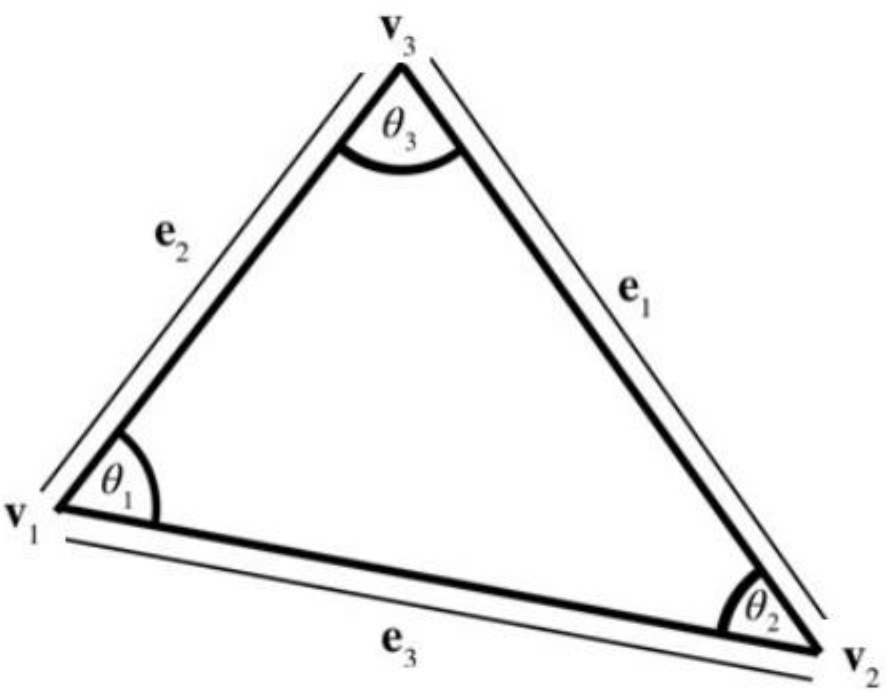
$$A = \frac{\|\mathbf{e}_1 \times \mathbf{e}_2\|}{2}$$



If you only know the length of the edges, use Herons's formula:

$$s = \frac{l_1 + l_2 + l_3}{2} = \frac{p}{2}$$

$$A = \sqrt{s(s - l_1)(s - l_2)(s - l_3)}$$

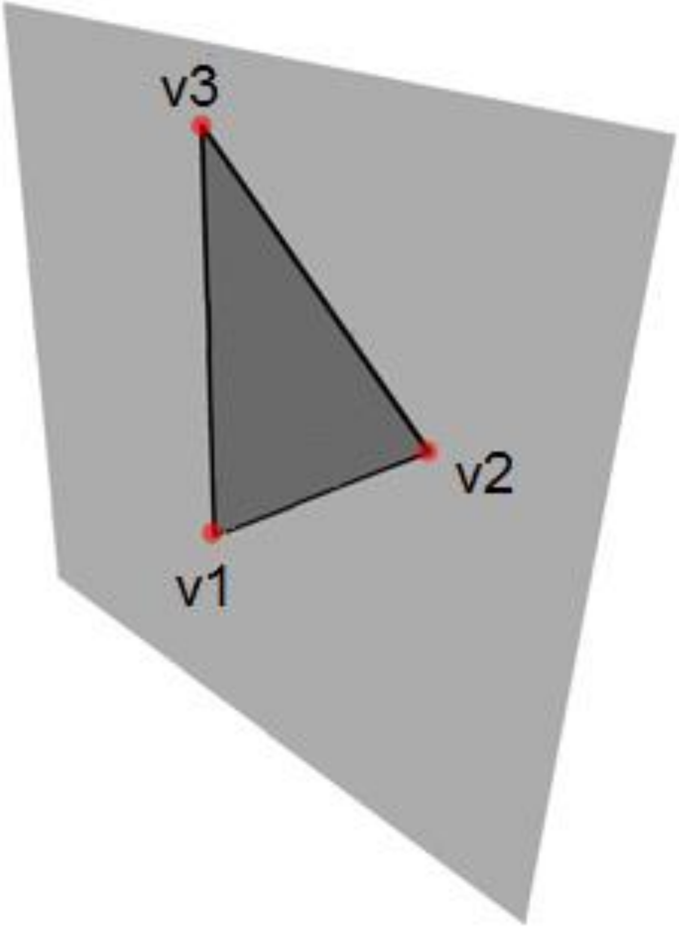


Triangle Plane

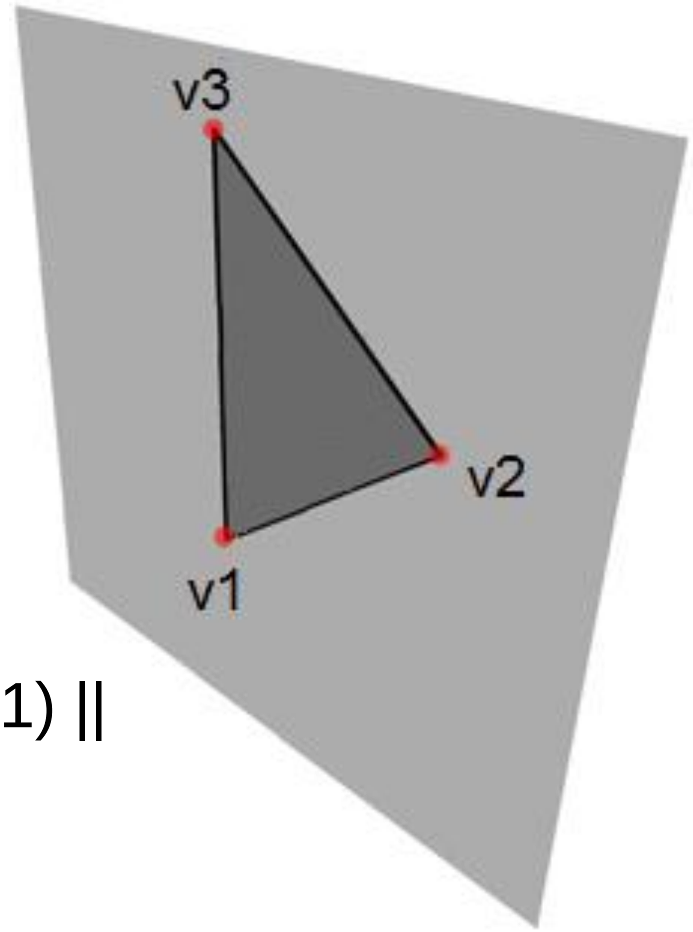
In 3D space, each triangle lies on a specific plane

$p \cdot n = d$

How can we compute n and d?



Triangle Plane



In 3D space, each triangle lies on a specific plane

$p \cdot n = d$

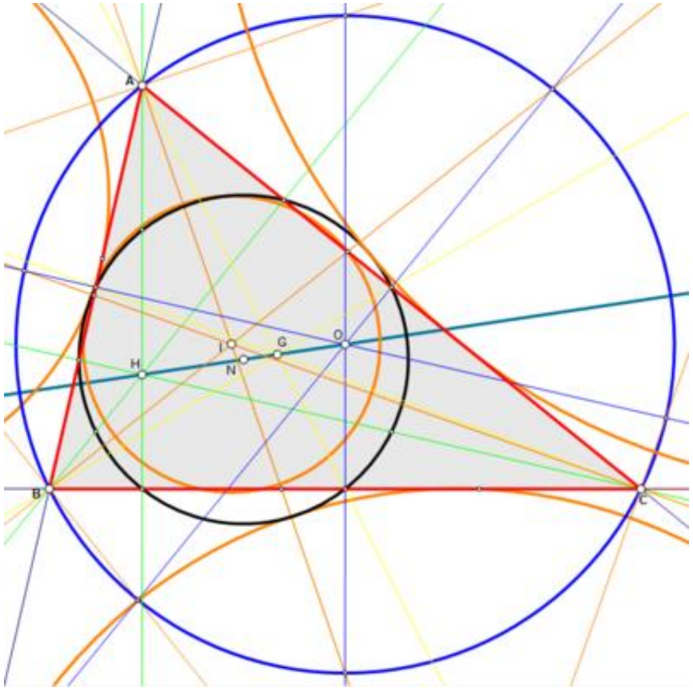
How can we compute n and d?

$n = (v2-v1)x(v3-v1) / || (v2-v1)x(v3-v1) ||$

$d = n \cdot v1 = n \cdot v2 = n \cdot v3$

Triangle Centers

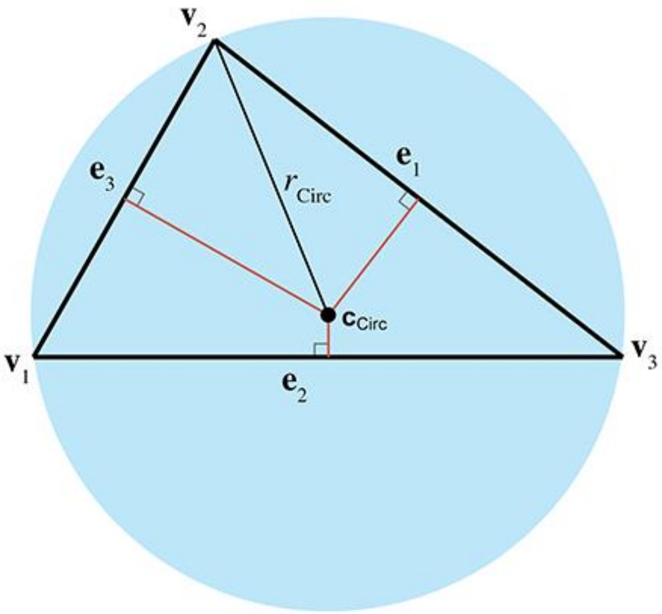
- A triangle center is a point in the plane that is in “the middle” of the triangle by some measure.
- Triangle centers are invariant under similarity transformations such as a rotation, reflection, dilation, or translation.
- For an equilateral triangle, all triangle centers coincide at the same point. However the they may take different positions from each other on general triangles.
- Four triangle centers were already known by the ancient Greeks:
  - Circumcenter
  - Incenter
  - Orthocenter
  - Barycenter
- Today, mathematicians have discovered over 50,000 triangle centers: https://faculty.evansville.edu/ck6/encyclopedia/ETC.html

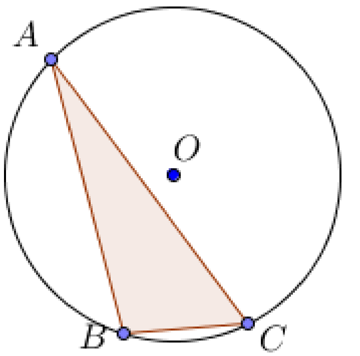


Circumcenter

The circumcenter is found by taking the midpoint of each leg of the triangle, and constructing a line perpendicular to it. Where all three lines intersect is the circumcenter.

It is the center of a circle that passes through all the three vertices of the triangle. The circumcenter does not always lie inside the triangle.



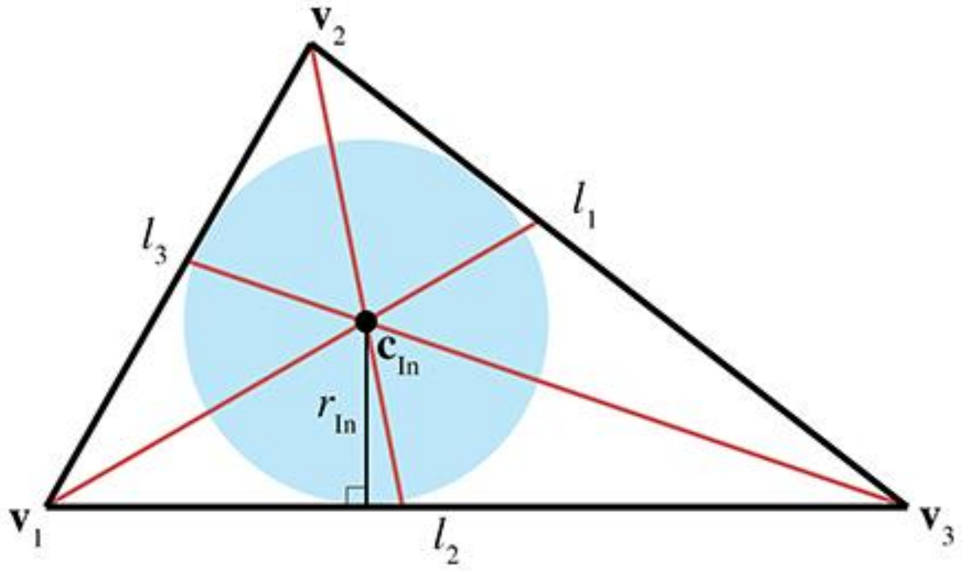


Obtuse Angled Triangle

Incenter

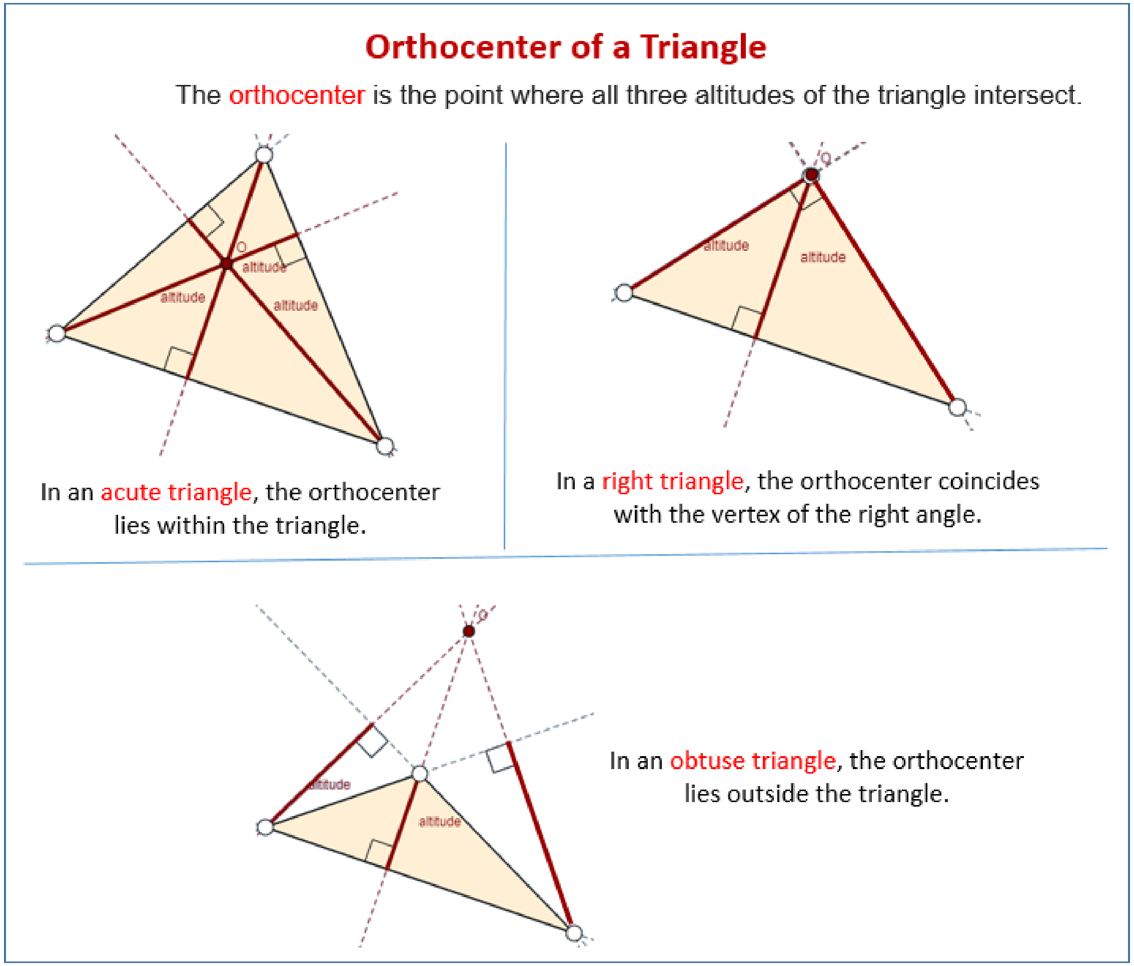
The incenter of a triangle is constructed by taking the intersection of the angle bisectors of the three vertices of the triangle.

It is the point equidistant from the all three sides, and therefore the center point of the inscribed circle of the triangle. The incenter always lies inside the triangle.



Orthocenter

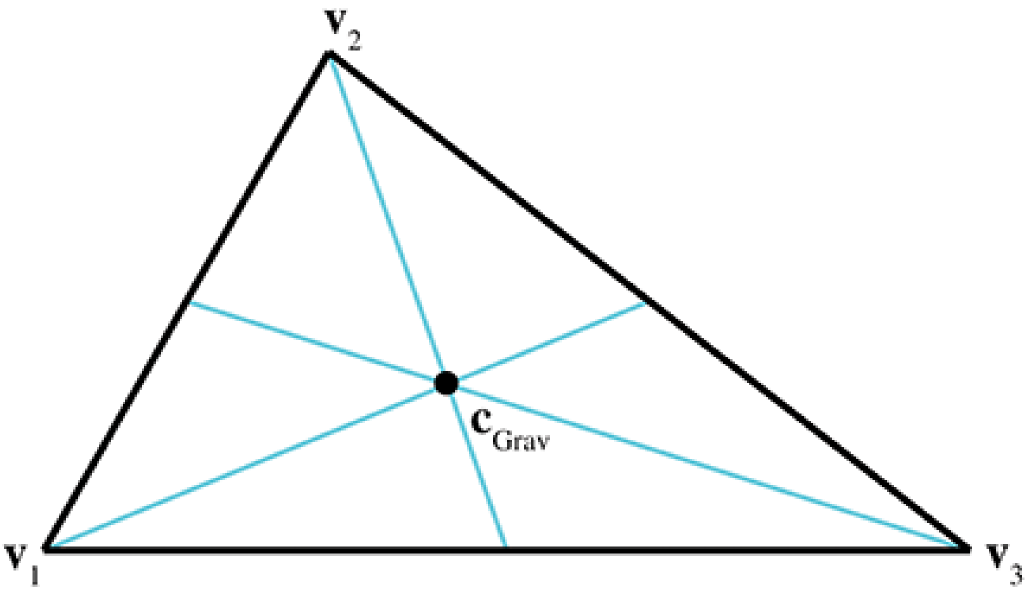
The three altitudes intersect in a single point, called the orthocenter of the triangle. The orthocenter does not always lie inside the triangle.



Barycenter

The barycenter, or centroid, of a triangle is defined as the mean of the three vertices. It also represents the “center of gravity” of the whole triangle. It is constructed by connecting the midpoints of each leg of the triangle to the opposite vertex. The barycenter always lies inside the triangle.

$$\mathbf{c}_{\text{Grav}} = \frac{\mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3}{3}$$



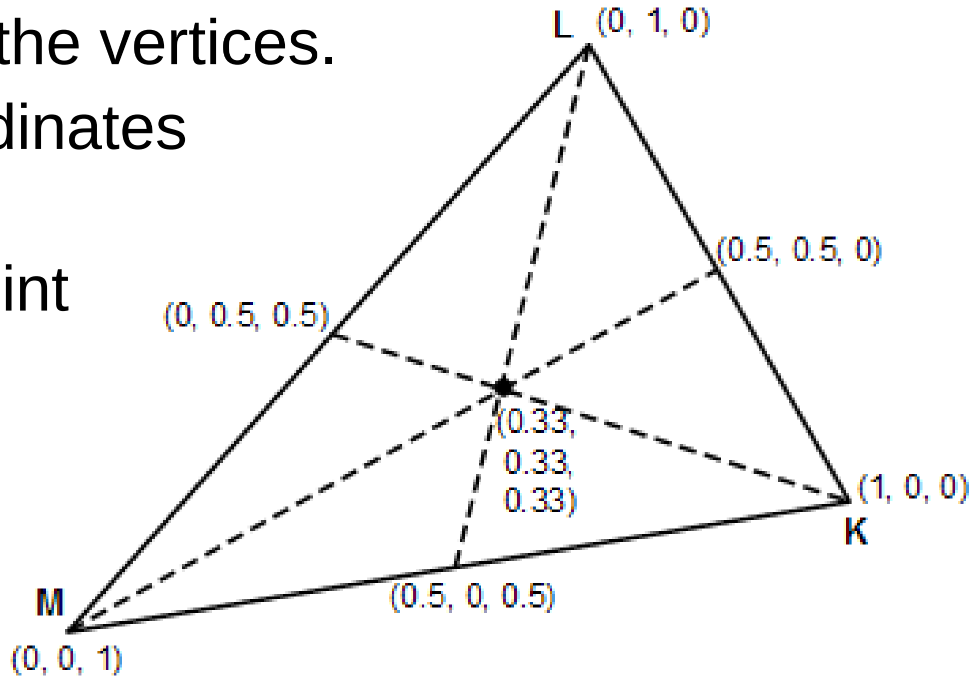
Barycentric space

- A coordinate space related to the surface of the triangle.
- Any point in the triangle plane can be expressed as the weighted average of the vertices.
- The barycentric coordinates

  (b_1, b_2, b_3)

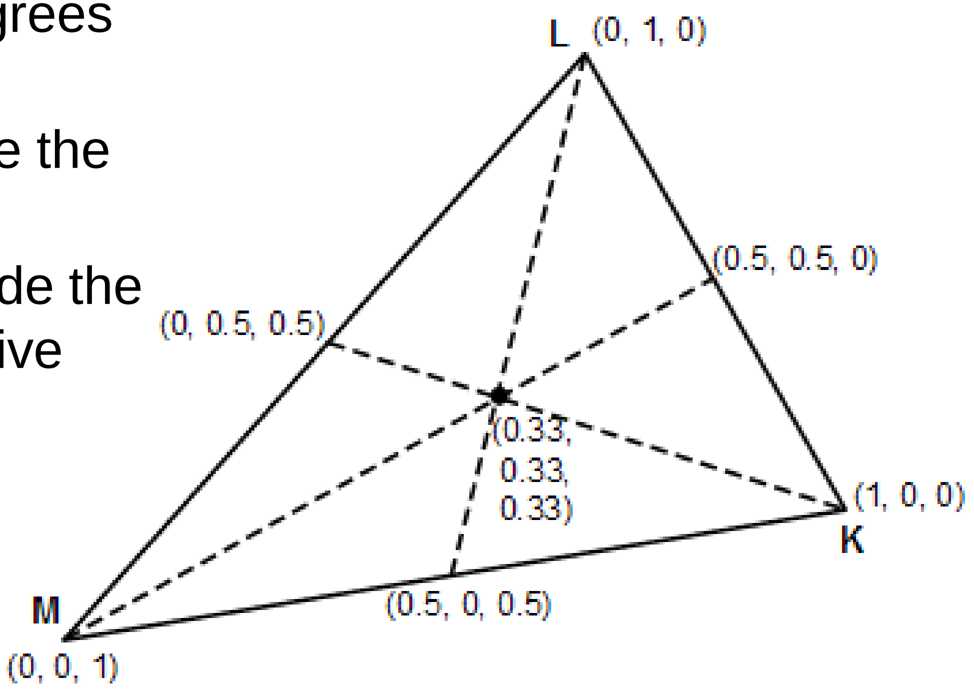
  correspond to the point

  $b_1\mathbf{v}_1 + b_2\mathbf{v}_2 + b_3\mathbf{v}_3$



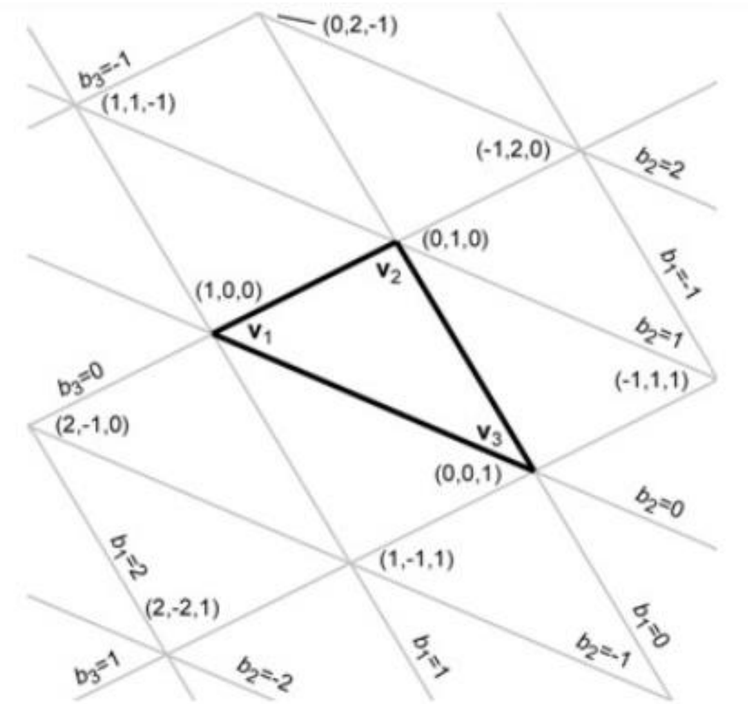
Barycentric space

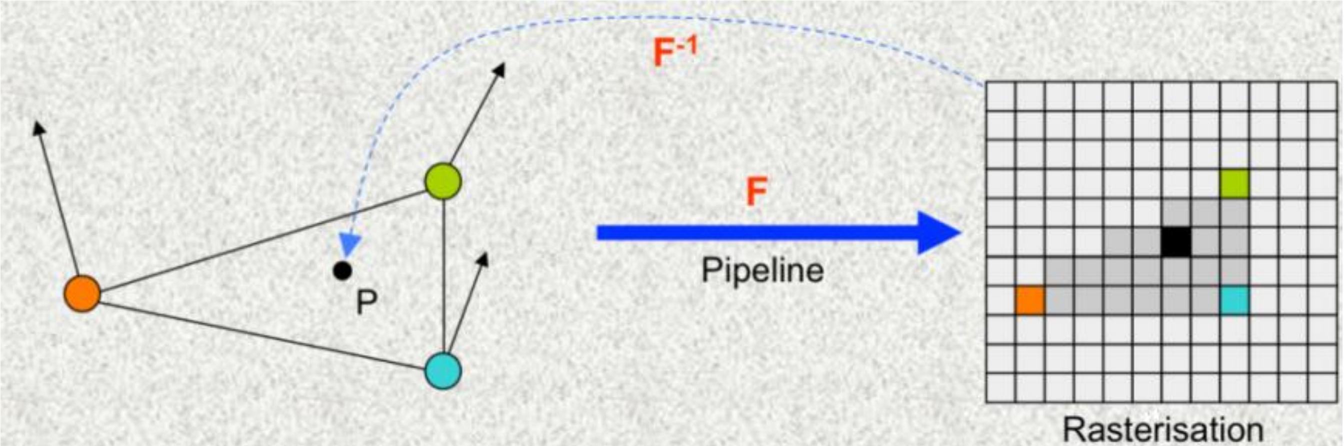
- The baycentric space is 2D, but there are 3 coordinates
- To stay on the plane, the sum of the coordinates must be one, i.e. the space actually only has two degrees of freedom
- The coordinates of a point inside the triangle are between 0 and 1
- The coordinates of a point outside the triangle have at least one negative value



Barycentric space

Many practical problems such as interpolation, tesselation and intersection can be solved using barycentric coordinates.



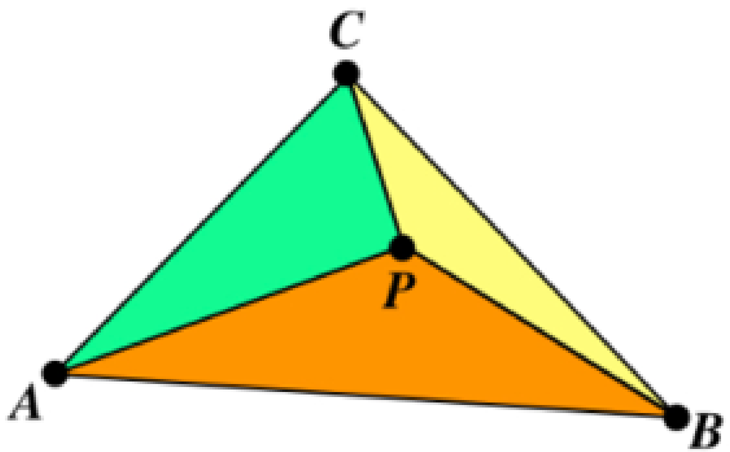


Barycentric space

Given a point P, how can we find its barycentric coordinates?

Barycentric space

Given a point P, how can we find its barycentric coordinates?

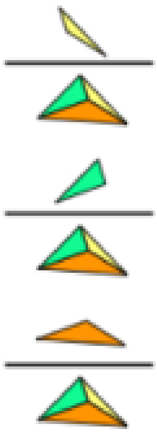


$$\boldsymbol{P} = w_A \times \boldsymbol{A} + w_B \times \boldsymbol{B} + w_C \times \boldsymbol{C}$$

$$w_A = \frac{\Delta PBC}{\Delta ABC}$$

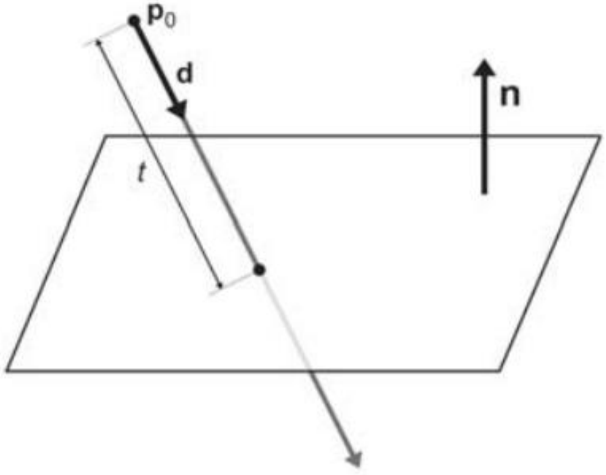
$$w_B = \frac{\Delta PCA}{\Delta ABC}$$

$$w_C = \frac{\Delta PAB}{\Delta ABC}$$



Note: By using signed triangle areas (negative values if clockwise order), these expressions are valid for points outside the triangle as well!

Intersection Ray-Plane

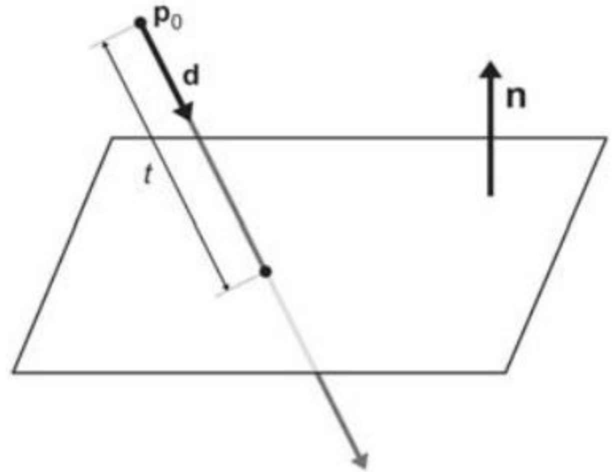


Given a ray
$r(t) = p0 + t*d,\ 0 \le t \le 1$

and a plane P with
$p \cdot n = d$

Do r and P intersect?
If so, where do they intersect?

Intersection Ray-Plane



Given a ray
r(t) = p0 + t*d, 0 ≤ t ≤ 1

and a plane P with
p · n = *d*

Do r and P intersect?
If so, where do they intersect?

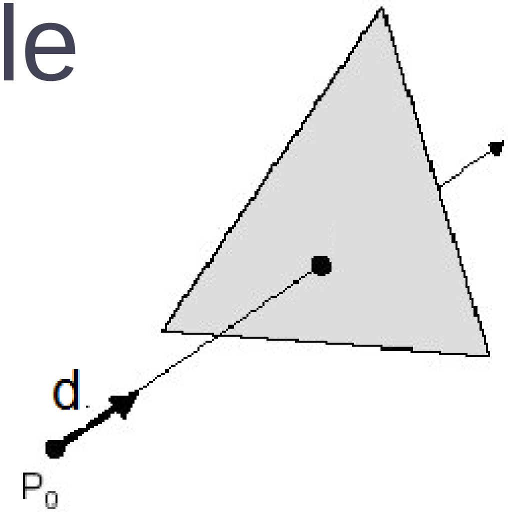
r(t) · n = *d*

$$\begin{aligned}
(\mathbf{p}_0 + t\mathbf{d}) \cdot \mathbf{n} &= d \\
\mathbf{p}_0 \cdot \mathbf{n} + t\mathbf{d} \cdot \mathbf{n} &= d \\
t\mathbf{d} \cdot \mathbf{n} &= d - \mathbf{p}_0 \cdot \mathbf{n} \\
t &= \frac{d - \mathbf{p}_0 \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}}
\end{aligned}$$

Remarks:

- If the ray is parallel to the plane, then the denominator d·n is zero and there is no intersection, or the ray lies on the plane.
- You may want to check the boundaries: if the value of t is out of range, then the ray does not intersect the plane.

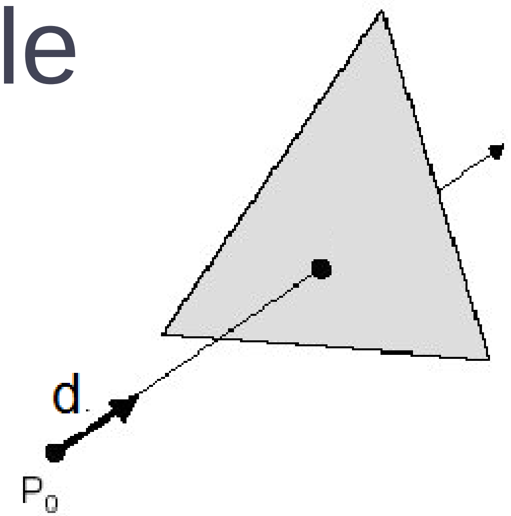
Intersection Ray-Triangle



The ray-triangle intersection test is very important in graphics and computational geometry. In the absence of a special raytrace test against a given complex object, we can always approximate the surface of an object as a triangle mesh and then raytrace against this triangle mesh representation.

Given a ray
r(t) = p0 + t*d, 0 ≤ t ≤ 1
and a triangle T in 3D space with the vertices v1, v2, v3.
Do r and T intersect?

Intersection Ray-Triangle



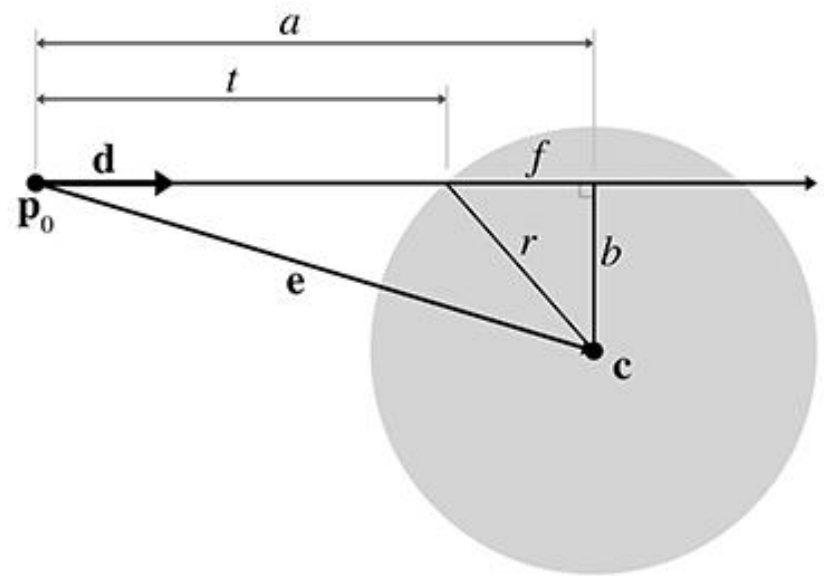
The ray-triangle intersection test is very important in graphics and computational geometry. In the absence of a special raytrace test against a given complex object, we can always approximate the surface of an object as a triangle mesh and then raytrace against this triangle mesh representation.

Given a ray
$r(t) = p0 + t*d, 0 \leq t \leq 1$
and a triangle T in 3D space with the vertices v1, v2, v3.
Do r and T intersect?

1) Compute the triangle plane
2) Compute the intersection point between the ray and the triangle plane
3) Test if the intersection point lies within the triangle using the barycentric coordinates

Intersection Ray-Sphere

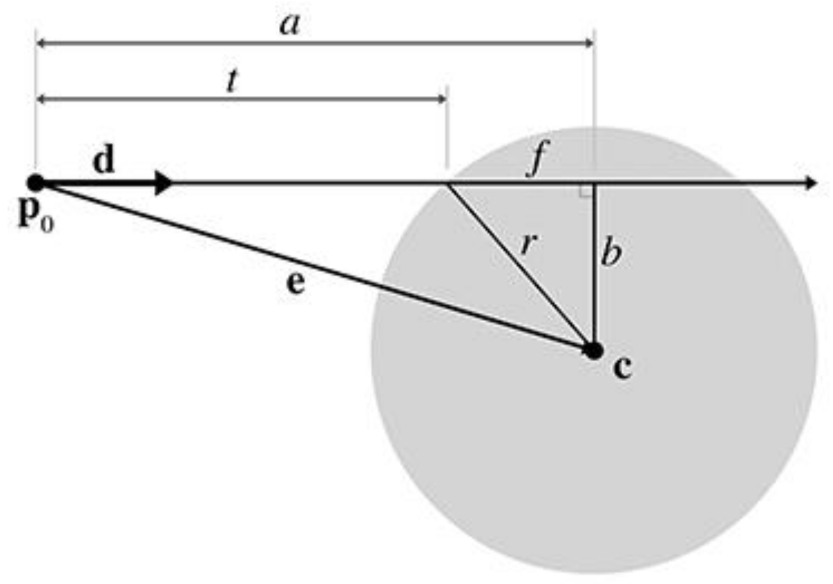
Given a ray
r(t) = p0 + t*d, 0 ≤ t ≤ 1
and a sphere S with
||p - c|| = r
Do p and S intersect?
If so, where do they intersect?



Hint:
1) Compute a
2) Compute f
=> t = a ± f

Intersection Ray-Sphere

Given a ray
$r(t) = p0 + t*d$, $0 \le t \le 1$
and a sphere S with
$||p - c|| = r$
Do p and S intersect?
If so, where do they intersect?



1) $a = (c - p0) \cdot d$
2) $f^2 + b^2 = r^2$ and $a^2 + b^2 = e^2$
   $\Rightarrow f^2 = r^2 - e^2 + a^2$
   $\Rightarrow f = \pm \text{sqrt}(r^2 - e^2 + a^2)$

$\Rightarrow t = a \pm \text{sqrt}(r^2 - e^2 + a^2)$

Remarks:
- If $r^2 - e^2 + a^2$ is negative, then the ray does not intersect the sphere.
- Use boundary checks if the ray is finite.
- The origin of the ray could be inside the sphere. This is indicated by $e^2 < r^2$.

Intersection Ray-Ray

Given two rays
$r1(t1) = p1 + t1*d1, 0 \le t1 \le 1$
$r2(t2) = p2 + t2*d2, 0 \le t2 \le 1$

Do r1 and r2 intersect?

Intersection Ray-Ray

Given two rays
r1(t1) = p1 + t1*d1, 0 ≤ t1 ≤ 1
r2(t2) = p2 + t2*d2, 0 ≤ t2 ≤ 1

Do r1 and r2 intersect?

$$\begin{aligned}
\mathbf{r}_1(t_1) &= \mathbf{r}_2(t_2) \\
\mathbf{p}_1 + t_1\mathbf{d}_1 &= \mathbf{p}_2 + t_2\mathbf{d}_2 \\
t_1\mathbf{d}_1 &= \mathbf{p}_2 + t_2\mathbf{d}_2 - \mathbf{p}_1 \\
(t_1\mathbf{d}_1) \times \mathbf{d}_2 &= (\mathbf{p}_2 + t_2\mathbf{d}_2 - \mathbf{p}_1) \times \mathbf{d}_2 \\
t_1(\mathbf{d}_1 \times \mathbf{d}_2) &= (t_2\mathbf{d}_2) \times \mathbf{d}_2 + (\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{d}_2 \\
t_1(\mathbf{d}_1 \times \mathbf{d}_2) &= t_2(\mathbf{d}_2 \times \mathbf{d}_2) + (\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{d}_2 \\
t_1(\mathbf{d}_1 \times \mathbf{d}_2) &= t_2\mathbf{0} + (\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{d}_2 \\
t_1(\mathbf{d}_1 \times \mathbf{d}_2) &= (\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{d}_2 \\
t_1(\mathbf{d}_1 \times \mathbf{d}_2) \cdot (\mathbf{d}_1 \times \mathbf{d}_2) &= ((\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{d}_2) \cdot (\mathbf{d}_1 \times \mathbf{d}_2) \\
t_1 &= \frac{((\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{d}_2) \cdot (\mathbf{d}_1 \times \mathbf{d}_2)}{\|\mathbf{d}_1 \times \mathbf{d}_2\|^2}
\end{aligned}$$

t_2 is obtained in a similar fashion:

$$t_2 = \frac{((\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{d}_1) \cdot (\mathbf{d}_1 \times \mathbf{d}_2)}{\|\mathbf{d}_1 \times \mathbf{d}_2\|^2}$$

Remarks:

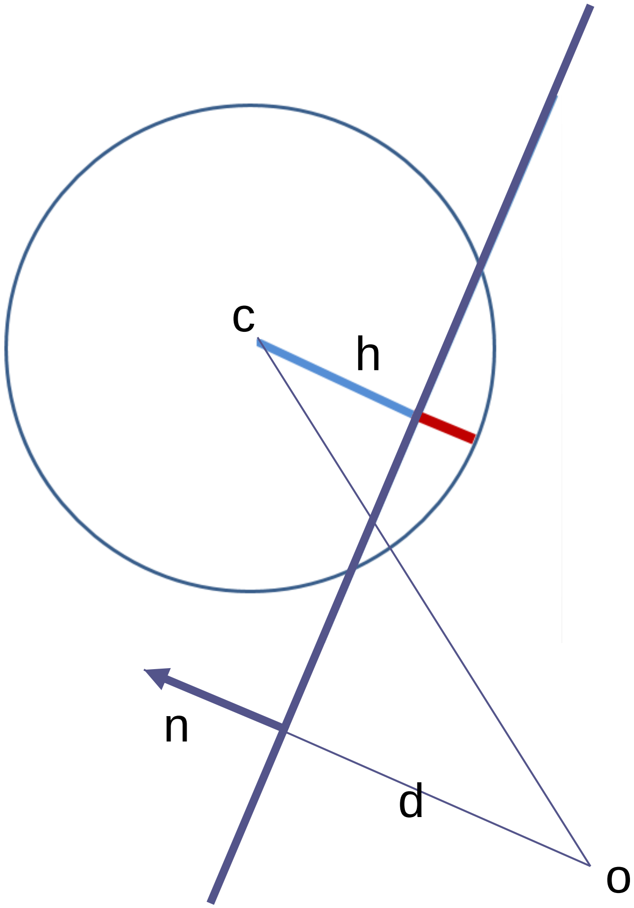
- If the rays are parallel, then d1 x d2 is the zero vector, and the denominator of both equations is zero.
- If the lines are skew, then r1(t1) and r2(t2) are the points of closest approach.
- To distinguish between skew and intersecting lines, we can examine the distance between r1(t1) and r2(t2). In practice, an exact intersection rarely occurs due to floating-point imprecision, and a tolerance must be used.
- t1 and t2 have not been not restricted. If the rays have finite length, then a boundary test must be applied.

Intersection Sphere-Plane

Given a Sphere S
$||p - c|| = r$

and a plane P with
$p \cdot n = d$

Do S and P intersect?
If so, where how deeply do they intersect?

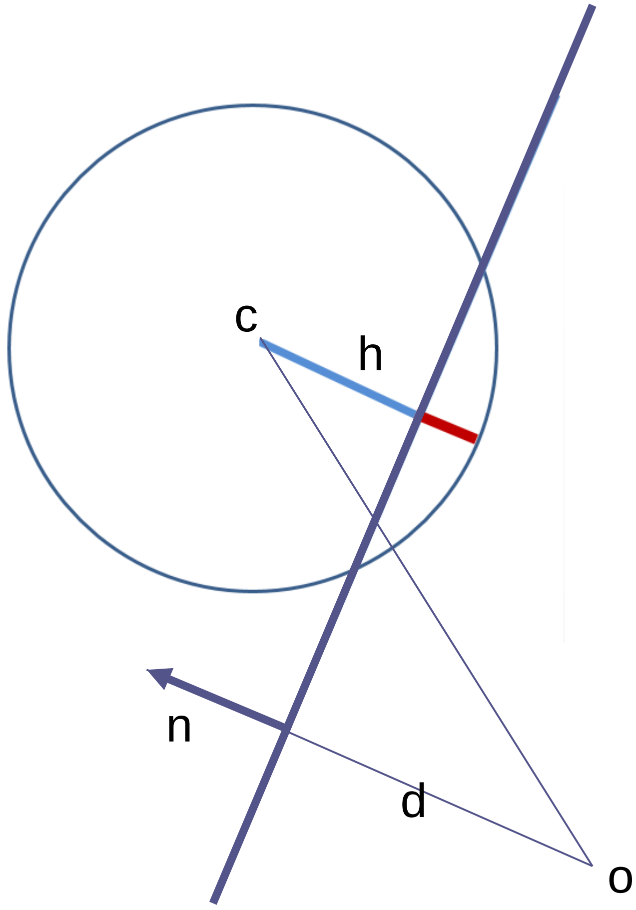


Intersection Sphere-Plane

Given a Sphere S
||p – c|| = r

and a plane P with
p · n = d

Do S and P intersect?
If so, where how deeply do they intersect?

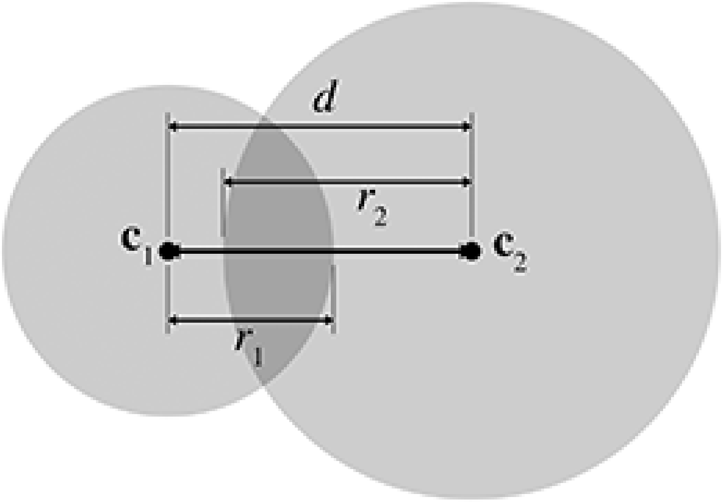


h = c · n - d
S and P intersect if |h| ≤ r, with depth r-|h|.

Intersection Sphere-Sphere

Given two spheres
S1: ||p - c1|| = r1
S2: ||p - c2|| = r2

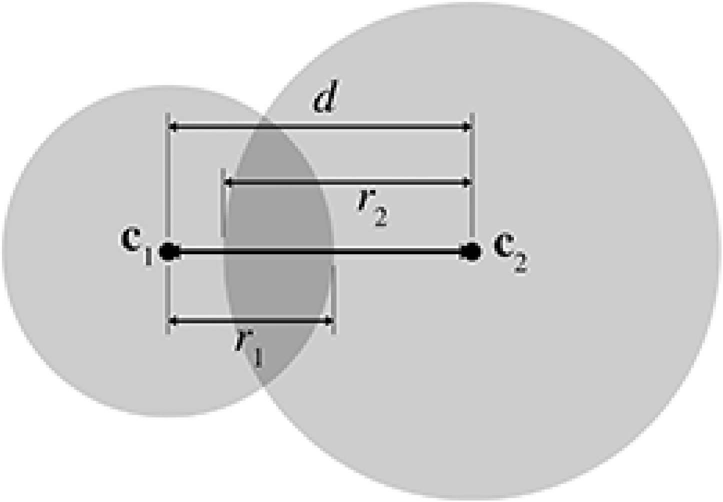
Do S1 and S2 intersect?
If so, how deeply do they intersect?



Intersection Sphere-Sphere

Given two spheres
S1: ||p - c1|| = r1
S2: ||p - c2|| = r2

Do S1 and S2 intersect?
If so, how deeply do they intersect?



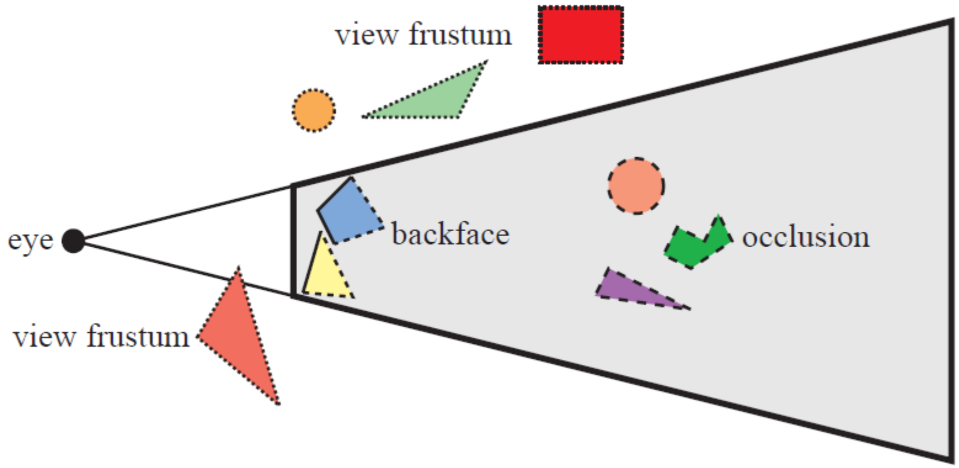
Define d := ||c2 – c1||

- d > r1 + r2 => no intersection
- |r1 – r2| ≤ d ≤ r1 + r2 => intersection with depth d
- d < |r1 – r2| => one sphere inside the other

Back-Face Culling

In wireframe mode, all triangles in the render queue must be drawn. With solid triangles however, back-facing triangles should be discarded. This step is called **back-face culling**.

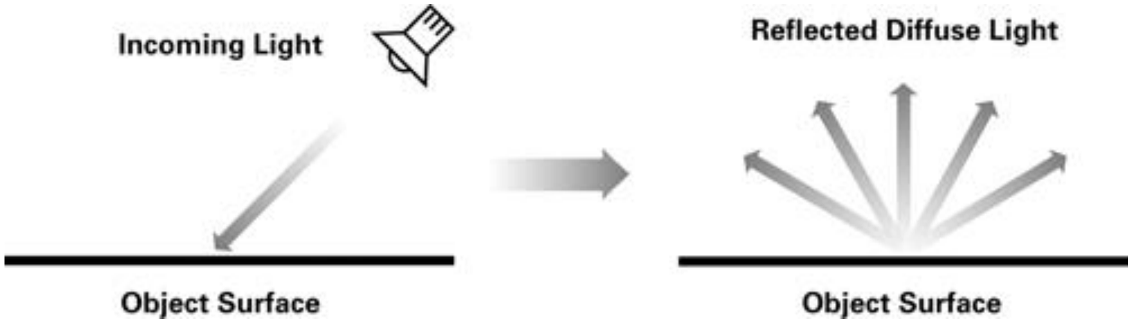
In camera coordinates, the normal vector of a facing triangle has positive z.
In screen coordinates, the vertices of a facing triangle are counter clockwise.



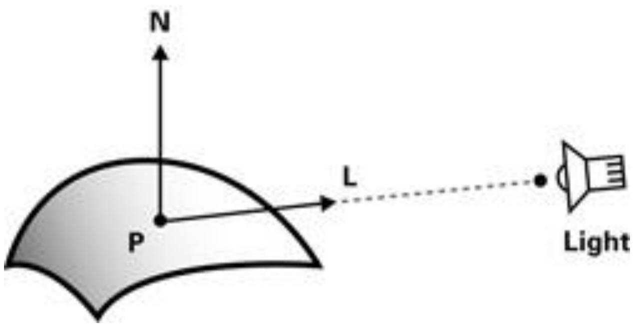
View-Frustum culling, Back-face culling and Occlusion culling

A simple light model

You can compute the illumination of a flat surface by considering its angle to the rays of a directional light source (« diffuse lighting »).



The reflection is calculated by taking the dot product of the surface normal vector N, and a normalized light-direction vector L pointing from the surface to the light source. This number represents the illumination level and is then multiplied by the color of the surface: max(L · N, 0) * color



Exercise

Toggle between wireframe mode and solid triangles.
In solid mode:

1) Add a light source emitting directional light to the scene. Illuminate the solid triangles with the diffuse lighting model.

2) Activate/deactivate back-face culling. You will notice that no back-face culling leads to visual flaws.

3) When the mouse hovers on a triangle which is facing you, the application highlights the selected triangle and plots the 3D coordinates of the corresponding surface point.