Geometric Tests

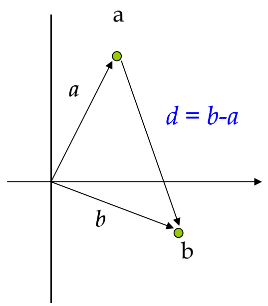
305890
Spring 2013
6/14/2013
Kyoung Shin Park
kpark@dankook.ac.kr

Overview

- Closest Point and Distance Test
  - Closest point on Line, Sphere, AABB, OBB, and Plane
  - Distance from Point to Point, Line, Sphere, AABB, OBB, and Plane
  - Distance from Sphere to Line, Sphere, AABB, OBB, and Plane
- Inside/Outside Test
  - Sphere, AABB, and OBB contains Point
  - Sphere contains Sphere
  - AABB contains AABB
- Intersection Test
  - Line-Line, Line-Sphere, Line-AABB, Line-Plane, Line-Triangle Intersection
  - Sphere-Sphere, Sphere-AABB, Sphere-Plane Intersection
  - AABB-AABB, AABB-Plane Intersection
  - OBB-OBB, OBB-Plane Intersection – not discussed today

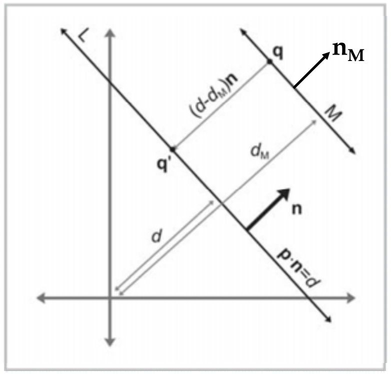
Distance

- Compute the distance between two points a and b
  - construct vector *a*
  - construct vector *b*
  - compute vector $d = b - a$
  - Distance(a,b) $= \| d \|$
    $= \| b - a \|$
    = sqrt(dot(d, d))



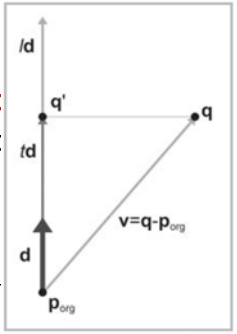
Closest Point on 2D Implicit Line

- Consider an infinite line L in 2D defined implicitly by all points **p** such that **p•n** = d, where **n** is a unit vector.
- Find the point **q**' that is the closest point on L to **q**.
- Let $\mathbf{n}_M$ & d_M be the normal & d value of the line M.
- Since L and M are parallel,
  - $\mathbf{n}_M = \mathbf{n}$
  - $d_M = \mathbf{q} \cdot \mathbf{n}$
- The signed distance from M to is $d - d_M = d - \mathbf{q} \cdot \mathbf{n}$
- $\mathbf{q}' = \mathbf{q} + (d - d_M)\mathbf{n}$
  $= \mathbf{q} + (d - \mathbf{q} \cdot \mathbf{n})\mathbf{n}$



Closest Point on Parametric Ray

- Consider the parametric ray R defined by: **p**(t) = **p**$_{org}$+ t**d**, where t varies from 0…l (l is the length of R).
- For a given point **q**, find the point **q**′ that is closest to **q**.
- Let **v** = **q** – **p**$_{org}$.
- Then, t = **v•d** = (**q** – **p**$_{org}$)•d
- Then, **q**′ = **p**$_{org}$+ t**d**= **p**$_{org}$ + ((**q** – **p**$_{org}$)•d)d
- If t<0 or t>l, then **p**(t) is not in the portic contained by R.
  - If t<0, **q** will be the origin.
  - If t>l, **q** will be endpoint.
- If **d** is not a unit vector, then $t = \frac{(q - p_{org})\cdot d}{\|d\|}$



$$\text{Distance}(q, \text{line}) = \sqrt{\|v\|^2 - t^2}$$

Projecting v onto d : $v_{\|} = d\frac{v\cdot d}{\|d\|^2}$ & $\|v_{\|}\| = \frac{v\cdot d}{\|d\|}$

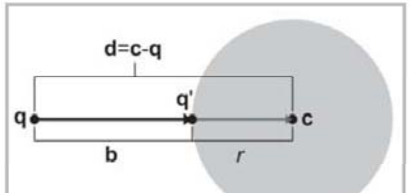
Closest Point on Circle/Sphere

- Imagine a 2D point **q** and a circle with center **c** and radius r.
- Find **q**′, which is the closest point on the circle to **q**.
- Let **d** be the vector from **q** to **c**. This vector intersects the circle at **q**′.
- Let **b** be the vector from **q** to **q**′.

$$b = \frac{\|d\| - r}{\|d\|} d$$

- Thus, $q' = q + b$

$$= q + \frac{\|d\| - r}{\|d\|} d$$



$$\text{Distance}(q, \text{sphere}) : \|b\| = \|d\| - r$$

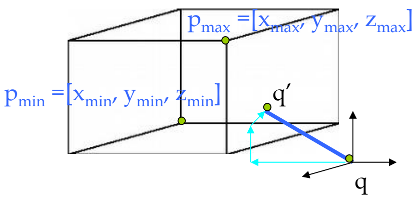
Closest Point on AABB

- Let AABB defined by the extreme points **p**$_{min}$ and **p**$_{max}$.
- Compute **q**′, the closest point in AABB to **q**.
- This is done by "pushing" **q** into AABB along each axis.

if q[0] < min[0]: q[0] = min[0]
elif q[0] > max[0]: q[0] = max[0]

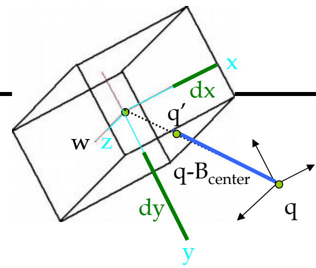
if q[1] < min[1]: q[1] = min[1]
elif q[1] > max[1]: q[1] = max[1]

if q[2] < min[2]: q[2] = min[2]
elif q[2] > max[2]: q[2] = max[2]



$$\text{Distance}(q, \text{AABB}) : \|(q - q')\|$$

Closest Point on OBB



- Closest point on the OBB, q′

$x = \frac{B_u \bullet (q - B_{center})}{\|B_u\|}$, if $x < -\|B_u\|, x = -\|B_u\|$ & $dx = -\|B_u\| - x$

if $x > \|B_u\|, x = \|B_u\|$ & $dx = x - \|B_u\|$

$otherwise, dx = 0$

$y = \frac{B_v \bullet (q - B_{center})}{\|B_v\|}$, if $y < -\|B_v\|, y = -\|B_v\|, dy = -\|B_v\| - y$ & if $y > \|B_v\|, y = \|B_v\|, dy = y - \|B_v\|$ & $dy = 0$

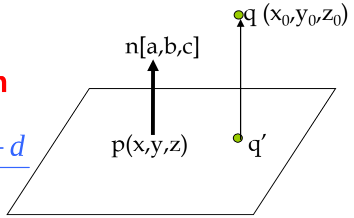
$z = \frac{B_w \bullet (q - B_{center})}{\|B_w\|}$, if $z < -\|B_w\|, z = -\|B_w\|, dz = -\|B_w\| - z$ & if $z > \|B_w\|, z = \|B_w\|, dz = z - \|B_w\|$ & $dz = 0$

- Distance from a point to the nearest point on the OBB

$$\text{Distance} = \sqrt{dx^2 + dy^2 + dz^2}$$

Closest Point on Plane

- Consider a plane P defined by p•n + d = 0
- Given a point **q**, find the point **q**′ (the closest point on P to **q)**, which is the result of projecting **q** onto P.
- Imagine a line containing the point **q** on the plane P.
  - **q** = **q**′ + *a***n**
  - q•n = q′•n + (*a***n)**•n
  - q•n = -d + *a*
  - *a* = q•n + d
- Therefore, **q**′ = **q** - (q•n + d)**n**



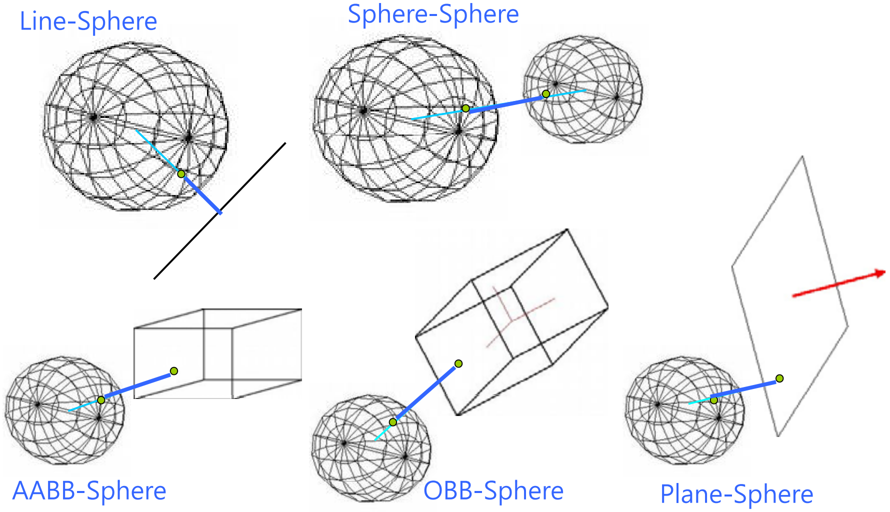
$$\text{Distance}(q\text{ , plane}) = \frac{ax_0 + by_0 + cz_0 + d}{\sqrt{a^2+b^2+c^2}}$$

where $q(x_0, y_0, z_0)$ *and Plane* $ax + by + cz + d = 0$

Distance – Sphere to Others

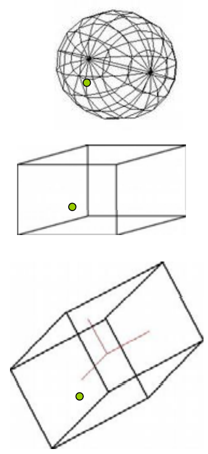
- Distance from a line to the sphere
  - Distance(line, s) = Distance(s.center, line) – s.radius
- Distance from a sphere to the sphere
  - Distance(sphere, s) = Distance(s.center, sphere) – s.radius
- Distance from AABB to the sphere
  - Distance(aabb, s) = Distance(s.center, aabb) – s.radius
- Distance from OBB to the sphere
  - Distance(obb, s) = Distance(s.center, obb) – s.radius
- Distance from a plane to the sphere
  - Distance(plane, s) = Distance(s.center, plane) – s.radius

Distance – Sphere to Others



Inside/Outside

- Sphere contains a point
  - Point is inside Sphere
    if $|p - c|^2 < r^2$
- AABB contains a point
  - Point is inside AABB
    if $x_{min} \le p_x \le x_{max}$
    and $y_{min} \le p_y \le y_{max}$
    and $z_{min} \le p_z \le z_{max}$
- OBB contains a point
  - Point is inside OBB
    if $(p - b_{center}) \cdot b_u \le |b_u|^2$
    and $(p - b_{center}) \cdot b_v \le |b_v|^2$
    and $(p - b_{center}) \cdot b_w \le |b_w|^2$



Inside/Outside

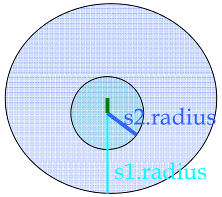
- Sphere contains sphere
  - Sphere is inside sphere

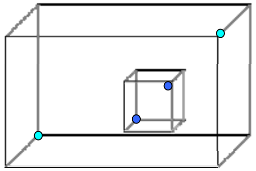
    if $|c1 - c2|^2 < |r1 + r2|^2$
- AABB contains AABB
  - AABB is inside AABB

    if $x1_{min} \le x2_{min}$ and $x1_{max} \ge x2_{max}$

    and $y1_{min} \le y2_{min}$ and $y1_{max} \ge y2_{max}$

    and $z1_{min} \le z2_{min}$ and $z1_{max} \ge z2_{max}$





Intersection Tests

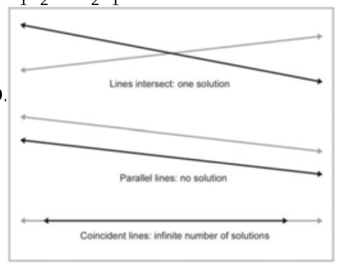
- These tests are designed to determine if two geometric primitives intersect and locate the intersection.
- The foundation for a collision detection system
  - To prevent objects from passing through each other
  - To make things appear to bounce off each other convincingly
- Two different types of intersection tests
  - A static test – checks two stationary primitives and detects if the two primitives intersect.
  - A dynamic test – checks two moving primitives and detects if and when two primitives intersect.
    - A time value (the value of the parameter *t*)
    - Easier to view the problem from the point of view of one of the primitives.

Intersection of Two Implicit Lines in 2D

- To find the intersection of two lines defined implicitly in 2D, use a straightforward matter of solving a system of linear equations.

$$a_1x + b_1y = d_1$$
$$a_2x + b_2y = d_2$$
$$\Rightarrow \quad x = \frac{b_2d_1 - b_1d_2}{a_1b_2 - a_2b_1} \qquad y = \frac{a_1d_2 - a_2d_1}{a_1b_2 - a_2b_1}$$

- Three possibilities:
  - Intersect at one point – The denominators will be nonzero.
  - Parallel (i.e., no intersection) – The denominators are zero.
  - Coincident – The denominators are zero.



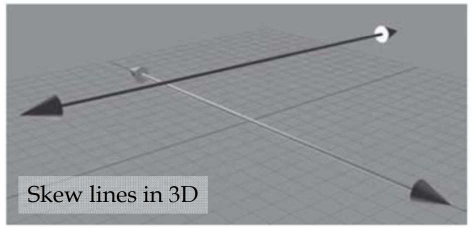
Intersection of Two Rays in 3D

- Consider two rays in 3D defined parametrically by:

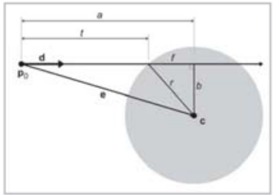
$$p_1(t_1) = \mathrm{p}_1 + t_1\mathrm{d}_1$$
$$p_2(t_2) = \mathrm{p}_2 + t_2\mathrm{d}_2$$

- If the rays lie in a plane, then there are three cases, similar to case of two lines.
- However, in 3D there is a fourth case where the rays are skew and do not share a common plane or intersect.
- Solutions for t_1 and t_2

$$t_1 = \frac{((p_2 - p_1) \times d_2) \cdot (d_1 \times d_2)}{\|d_1 \times d_2\|^2}$$
$$t_2 = \frac{((p_2 - p_1) \times d_1) \cdot (d_1 \times d_2)}{\|d_1 \times d_2\|^2}$$



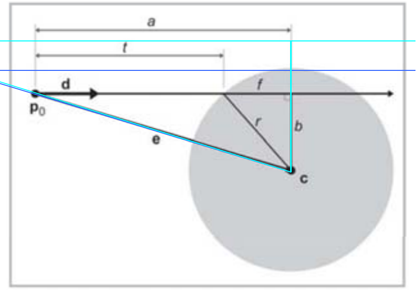
Intersection of Ray and Circle/Sphere



- We will consider the line **e** that contains the ray and the center of circle.
- The circle/sphere is defined by its center **c** and radius *r*.
- The ray defined by: p(t) = $\mathbf{p_0}$ + *t***d**
- The value of *t* at the point of intersection is *t* = *a* - *f*.
- *a* is computed by *a* = **e**·**d**, where **e** = **c** – $\mathbf{p_0}$.
- By the Pythagorean theorem, $f^2 + b^2 = r^2$.
- On the larger triangle, we can solve for b^2. $a^2 + b^2 = e^2$.
- As a result, $f^2 = r^2 - e^2 + a^2$,
- Finally, solving for *t*. $t = a - f = a - \sqrt{r^2 - e^2 + a^2}$
- Ray does not intersect the sphere if $r^2 - e^2 + a^2 < 0$

Intersection of Ray and Circle/Sphere

```
e = c – p0;
a = dot(e, d);
if (r*r – e*e + a*a < 0) {
    return None;
}
elseif (r*r – e*e + a*a == 0) {
    t = a;
    return p0 + t*d;
}
else {
    f = sqrt(r*r – e*e + a*a);
    t1 = a – f;
    t2 = a + f;
    return Ray(p0 + t1*d, p0 + t2*d);
}
```



Intersection of Ray and AABB

- Computing the intersection of a ray and an AABB is an important calculation.
  - The result of this test is commonly used for trivial rejection on more complicated objects.
  - For example, ray-tracing against multiple triangle meshes,
    - First, ray trace against the AABBs of the meshes to trivially reject entire meshes at once.
    - Then, ray trace against each triangle not rejected.

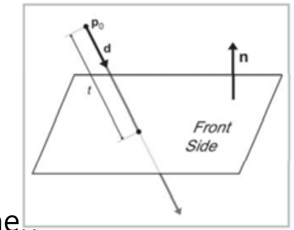
Intersection of Ray and Plane

- Let the ray be defined by **p**(*t*) = $\mathbf{p}_0$ + *t***d** and the plane be defined, by all points **p** such that **p**·**n** + *d* =0.
- To solve for *t* at the point of intersection, assume an infinite ray for the moment.

$$(\mathrm{p}_0 + t\mathrm{d}) \cdot \mathrm{n} + d = 0$$

$$t\mathrm{d} \cdot \mathrm{n} = -d - \mathrm{p}_0 \cdot \mathrm{n}$$

$$t = \frac{-(\mathrm{p}_0 \cdot \mathrm{n} + d)}{\mathrm{d} \cdot \mathrm{n}}$$



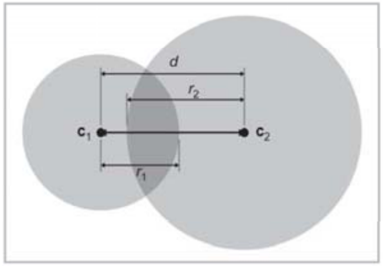
- If the ray is parallel to the plane, the denominator **d**•**n**=0 and there is no intersection.
- If the value of *t* is out of range, then the ray does not intersect the plane.

Intersection of Ray and Triangle

- The ray-triangle intersection test is very important in graphics and computational geometry.
- To find the intersection of ray and triangle,
  - First, compute the point where the ray intersects the plane containing the triangle.
  - Then, test to see if the point is inside the triangle by computing the barycentric coordinates of the point.
- To make this test as fast as possible,
  - Detect and return a negative result (no collision) as soon as possible.
  - Defer expensive mathematical operations, such as division, as long as possible
  - Only detect collisions where the ray approaches the triangle from the front side
- Optimizing expensive calculations, such as precomputing the polygon normal, is one more significant strategy.

Intersection of Two Circles/Spheres

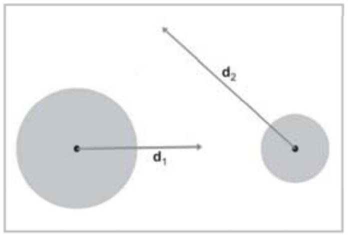
- We will consider two spheres defined by centers $\mathbf{c}_1$ and $\mathbf{c}_2$ and radii r_1 and r_2, respectively.

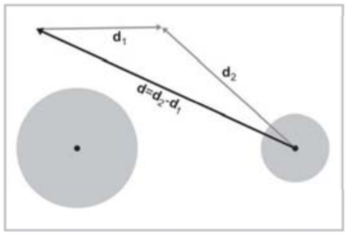


- Detection of the static intersection of two spheres
  - Let d be the distance between their centers $\mathbf{c}_1$ and $\mathbf{c}_2$.
  - Then, the spheres intersect if $d < r_1 + r_2$.
  - In practice, we use $d^2 < (r_1 + r_2)^2$ for avoiding the square root.

Intersection of Two Circles/Spheres

- Detection of the intersection of two moving spheres
- Assume two spheres with displacement vector $\mathbf{d}_1$ and $\mathbf{d}_2$, respectively.

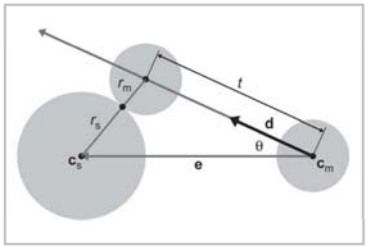




- Consider that sphere to be "stationary", while the other sphere is the "moving" sphere.
- Then, we can compute a single displacement vector **d**.

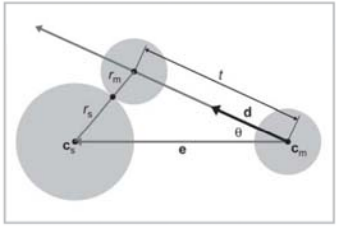
$\mathbf{d} = \mathbf{d}_2 - \mathbf{d}_1$

Intersection of Two Circles/Spheres



- Let the stationary sphere be defined by its center $\mathbf{c}_s$ and radius r_s.
- We will normalize **d** and vary t from 0 … l, where l is the distance traveled.
- The position of the center of the moving sphere at time t: $\mathbf{c}_m + t\mathbf{d}$.
- Let $\mathbf{e} = \mathbf{c}_s - \mathbf{c}_m$ and $r = r_m + r_s$.

Intersection of Two Circles/Spheres



- $r^2 = t^2 + |\mathbf{e}|^2 - 2t|\mathbf{e}|\cos\theta$ (the law of cosines)
- $0 = t^2 - 2(\mathbf{e}\cdot\mathbf{d})\,t + \mathbf{e}\cdot\mathbf{e} - r^2$ (the geometric interpretation of the dot product)

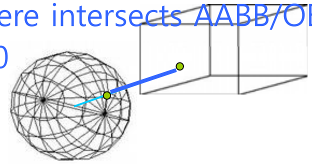
– → begin point of intersection
+ → cease point of intersection

$$t = \frac{2(e\cdot d) \pm \sqrt{(-2(e\cdot d))^2 - 4(e\cdot e - r^2)}}{2} = e\cdot d \pm \sqrt{(e\cdot d)^2 + r^2 - e\cdot e}$$

- If $|\mathbf{e}|^2 < r^2$, sphere are intersecting at t=0.

Intersection of Sphere and AABB/OBB

- To detect the static intersection of a sphere and an AABB/OBB
  - Find the point on the AABB/OBB that is closest to the center of the sphere.
  - Compute the distance from this point to the center of the sphere.
  - Compare this distance with the radius.
  - The sphere intersects AABB/OBB if this distance is smaller than the radius.
- The sphere intersects AABB/OBB if Distance(sphere, box) ≤ 0



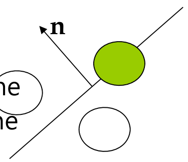
Intersection of Sphere and Plane

- To detect the static intersection of a sphere and a plane
- We simply compute the distance from the center of the sphere to the plane.

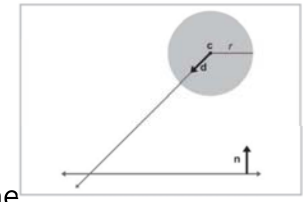
$$\text{Distance}(c, \text{plane}) = \frac{ac_x + bc_y + cc_z + d}{\sqrt{a^2 + b^2 + c^2}}$$

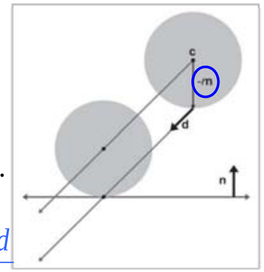
$$\text{where } c(c_x, c_y, c_z) \text{ and Plane } ax + by + cz + d = 0$$

- If this distance is less than the radius of the sphere, then the sphere intersects the plane.
- Three classifications
  - The sphere is completely on the front of the plane
  - The sphere is completely on the back of the plane
  - The plane straddles the sphere



Intersection of Sphere and Plane





- In dynamic situation, we will consider the plane to be stationary.
- We will define the plane using a normalized surface normal **n** and distance value *d*. **p**·**n** + *d* =0
- As *t* varies from 0…*l*, the motion of the center of the sphere is given by **c** + *t***d**, where c is the center of the sphere and d is a unit vector specifying the direction.
- Now, a ray defined by **p**(*t*) = (**c**-*r***n**) + *t***d**.
- We can give *t* as follows:

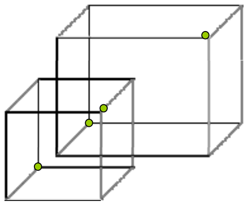
$$t = \frac{-(\mathrm{p}_0\cdot\mathrm{n} + d)}{\mathrm{d}\cdot\mathrm{n}} = \frac{-((\mathrm{c} - r\mathrm{n})\cdot\mathrm{n} + d)}{\mathrm{d}\cdot\mathrm{n}} = \frac{-\mathrm{c}\cdot\mathrm{n} + r - d}{\mathrm{d}\cdot\mathrm{n}}$$

Ray-plane intersection

Intersection of Two AABBs

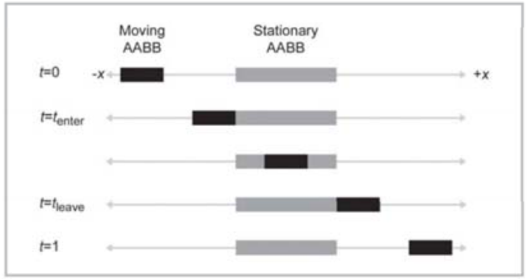
- Detecting the intersection of two static AABBs is trivial.
  - Simply check for overlapping extents on each dimension independently.
  - Two AABBs intersect if they overlap on all three axes

```
# return true if all three axes overlap
xmin = max(b1.min[0], b2.min[0])
xmax = min(b1.max[0], b2.max[0])
ymin = max(b1.min[1], b2.min[1])
ymax = min(b1.max[1], b2.max[1])
zmin = max(b1.min[2], b2.min[2])
zmax = min(b1.max[2], b2.max[2])
return (xmin <= xmax) and (ymin<= ymax) and (zmin<=
zmax)
# OR, exit with no intersection if separated along an axis
```



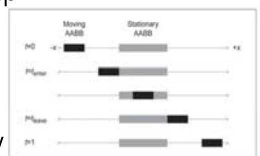
Intersection of Two AABBs

- Dynamic intersection of AABBs
  - Consider a stationary AABB defined by extreme points $\mathbf{s}_{min}$ and $\mathbf{s}_{max}$ and a moving AABB defined by extreme points $\mathbf{m}_{min}$ and $\mathbf{m}_{max}$ and displacing by an amount given by the vector $\mathbf{d}$, as t varies from 0…1.
  - Compute t, the parametric point in time where the moving box first collides with the stationary box.
  - First of all, we will solve the one-dimensional problem.



Intersection of Two AABBs

- First of all, we will solve the one-dimensional problem.
  - A point t_{enter} where the boxes first begin to overlap
  - A point t_{leave} where the boxes cease to overlap
- Let $m_{min}(t)$ and $m_{max}(t)$ be the minimum and maximum values of the moving box at time t, given by:
  - $m_{min}(t) = m_{min}(0) + td$, $m_{max}(t) = m_{max}(0) + td$

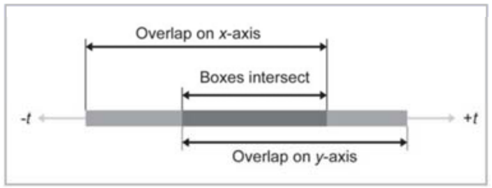


- Let s_{min} and s_{max} for the stationary box.
- Then, t_{enter} is the t value for which $m_{max}(t)$ is equal to s_{min} and t_{leave} is the t value for which $m_{min}(t)$ is equal to s_{max}.

$$t_{enter} = \frac{s_{min} - m_{max}(0)}{d} \qquad t_{leave} = \frac{s_{max} - m_{min}(0)}{d}$$

Intersection of Two AABBs

- In multi-dimensional problem, the intersection of these intervals on all dimensions gives the interval of time where the boxes intersect with each other.



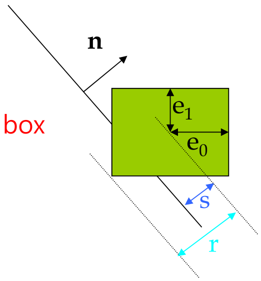
- If the interval is empty, then the boxes never collide.
- If the interval lies completely outside the range t=0…1, then there is no collision over the period of time we are interested in.
- We will need to maintain the interval during which the boxes overlap, mainly to determine if it is empty.

Intersection of AABB and Plane

- Let a AABB defined by extreme points $\mathbf{p}_{min}$ and $\mathbf{p}_{max}$ and a plane defined by all points $\mathbf{p}$ such that $\mathbf{p}\cdot\mathbf{n} + d = 0$.
- A simple static intersection test is to check whether the box is completely on one side of the plane or the other.
  - Actually, we don't have to check all eight corner points.
  - If $\mathbf{n}_x > 0$, then the corner with the minimum dot product has $x = x_{min}$ and the corner with the maximum dot product has $x = x_{max}$.
  - If $\mathbf{n}_x < 0$, then the opposite is true.
  - Similar statements apply to $\mathbf{n}_y$ and $\mathbf{n}_z$.
  - Compute the minimum and maximum dot product values.
    - No intersection if (the minimum dot product value > d) or (the maximum dot product < d)
    - Otherwise, an intersection is detected.

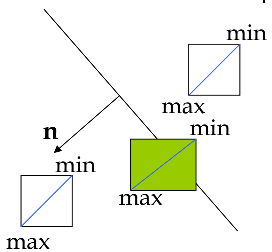
Intersection of AABB and Plane

```
# compute box center
c = (max+min)/2
# compute positive extent length
e = max - c
# compute the projection interval radius of box
r = e[0]*abs(n[0]) + e[1]*abs(n[1]) +
e[2]*abs(n[2])
# compute the distance of box center from
plane
s = distance(c, plane)
# intersect if s is within [–r, r] interval
return abs(s) <= r
```

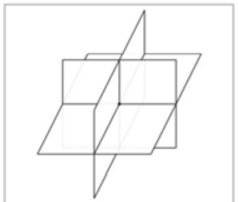


Intersection of AABB and Plane

- For a dynamic test, consider the plane to be stationary.
  - Locate the corner points with the minimum and maximum dot product and move these points to the plane,
  - If AABB is not initially intersecting the plane, it must strike the plane at the corner point closest to the plane.
  - For colliding with the "front" of the plane, we use the corner with the minimum dot product value



Intersection of Three Planes



- In 3D, three planes intersect at a point.
- Let the three planes be defined implicitly as:

$$p\cdot n_1 + d_1 = 0 \qquad p\cdot n_2 + d_2 = 0 \qquad p\cdot n_3 + d_3 = 0$$

$$p = \frac{-(d_1(n_2\times n_3) + d_2(n_3\times n_1) + d_3(n_1\times n_2))}{(n_1\times n_2)\cdot n_3}$$

$$n_1 = (a,b,c) \quad n_2 = (d,e,f) \quad n_3 = (g,h,i)$$

$$p = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}^{-1} \begin{pmatrix} -d_1 \\ -d_2 \\ -d_3 \end{pmatrix} \quad p = A^{-1}\begin{pmatrix} -d_1 \\ -d_2 \\ -d_3 \end{pmatrix} = \frac{1}{\det A}\, adjA \begin{pmatrix} -d_1 \\ -d_2 \\ -d_3 \end{pmatrix}$$

$$\det A = g(bf - ce) - h(af - cd) + i(ae - bd) = (n_1\times n_2)\cdot n_3$$

$$adjA = \begin{pmatrix} \overset{n_2\times n_3}{ei - fh} & \overset{n_3\times n_1}{ch - bi} & \overset{n_1\times n_2}{bf - ce} \\ fg - di & ai - cg & cd - af \\ dh - eg & bg - ah & ae - bd \end{pmatrix}$$

Other Tests

- Triangle-triangle intersection tests – many, many different ways to do this
- AABB-OBB (oriented bounding box) intersection tests
- Triangle-box intersection tests
- Tests using more "exotic" primitives, such as cylinders, cones, and tori