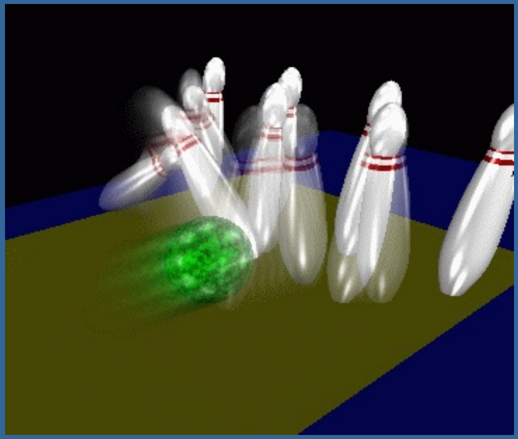
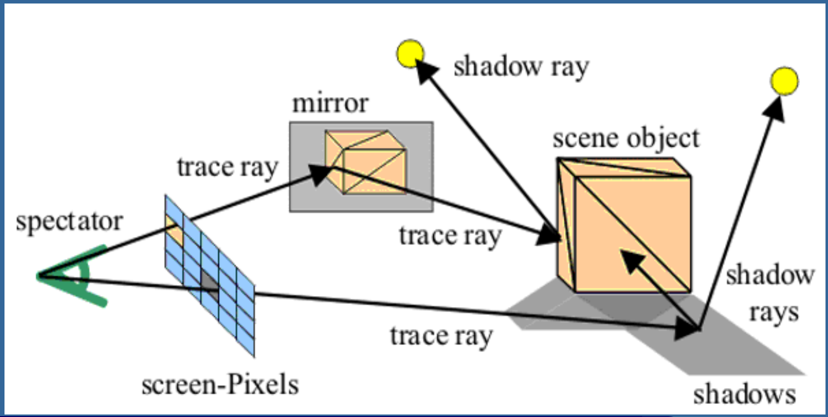
Intersection Testing
Chapter 16





Department of Computer Engineering

Chalmers University of Technology

What for?

- A tool needed for the graphics people all the time…
- Very important components:
  - Need to make them fast!
- Finding if (and where) a ray hits an object
  - Picking
  - Ray tracing and global illumination
- For speed-up techniques
- Collision detection (treated in a later lecture)

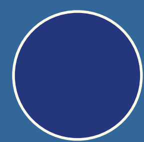
Example

Midtown Madness 3, DICE

Some basic geometrical primitives

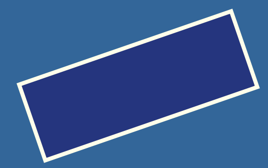
- Ray:

- Sphere:

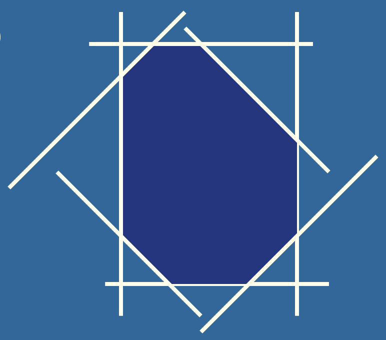


- Box
  - Axis-aligned (AABB)

  - Oriented (OBB)



- *k*-DOP

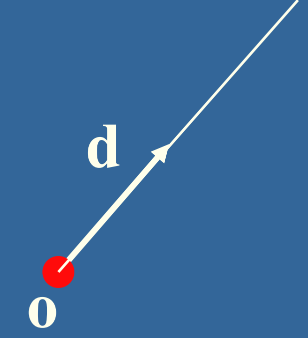


Some different techniques

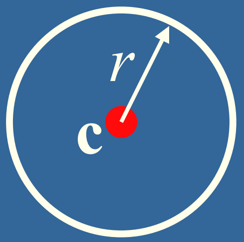
- Analytical
  - ”Solve an equation system”
  - E.g., ray/sphere, ray/plane, ray/triangle,
- Geometrical
  - ”Follow a set of steps”
  - Ray/box, ray/polygon
  - Separating axis theorem (SAT)
- Dynamic tests (to find time of collision)

- Given these, one can derive many tests quite easily
  - However, often tricks are needed to make them fast

Analytical: Ray/sphere test



- Sphere center: $\mathbf{c}$, and radius r
- Ray: $\mathbf{r}(t)=\mathbf{o}+t\mathbf{d}$
- Sphere formula: $||\mathbf{p}-\mathbf{c}||=r$
- Replace $\mathbf{p}$ by $\mathbf{r}(t)$, and square it:



$$(\mathbf{r}(t)-\mathbf{c})\cdot(\mathbf{r}(t)-\mathbf{c})-r^2=0$$

$$(\mathbf{o}+t\mathbf{d}-\mathbf{c})\cdot(\mathbf{o}+t\mathbf{d}-\mathbf{c})-r^2=0$$

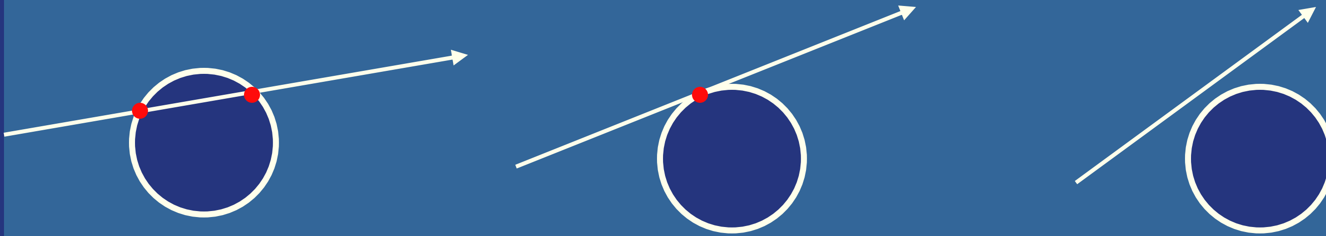
$$(t\mathbf{d}+(\mathbf{o}-\mathbf{c}))\cdot(t\mathbf{d}+(\mathbf{o}-\mathbf{c}))-r^2=0$$

$$(\mathbf{d}\cdot\mathbf{d})t^2+2((\mathbf{o}-\mathbf{c})\cdot\mathbf{d})t+(\mathbf{o}-\mathbf{c})\cdot(\mathbf{o}-\mathbf{c})-r^2=0$$

$$t^2+2((\mathbf{o}-\mathbf{c})\cdot\mathbf{d})t+(\mathbf{o}-\mathbf{c})\cdot(\mathbf{o}-\mathbf{c})-r^2=0 \qquad ||\mathbf{d}||=1$$

Analytical, continued

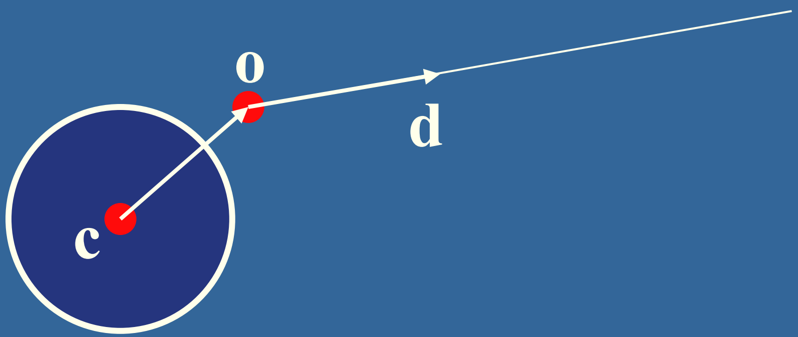
$$t^2 + 2((\mathbf{o}-\mathbf{c})\cdot\mathbf{d})t + (\mathbf{o}-\mathbf{c})\cdot(\mathbf{o}-\mathbf{c}) - r^2 = 0$$



- Be a little smart…

$$(\mathbf{o}-\mathbf{c})\cdot\mathbf{d} > 0 \quad ?$$

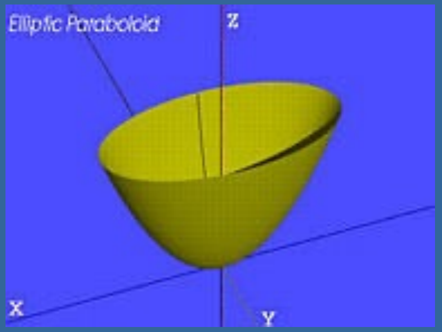
$$(\mathbf{o}-\mathbf{c})\cdot(\mathbf{o}-\mathbf{c}) - r^2 < 0 \quad ?$$



- Such tests are called ”rejection tests”
- Other shapes: $p_x^2 + p_y^2 = r^2$

$$(p_x/a)^2 + (p_y/b)^2 + (p_z/c)^2 = 1$$

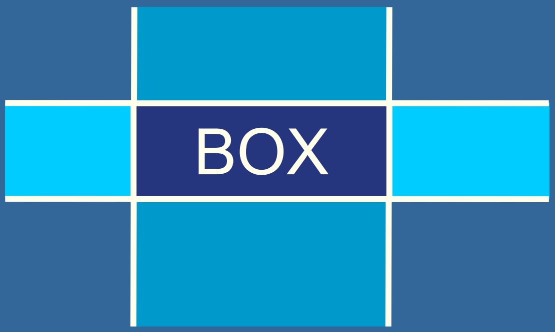
$$(p_x/a)^2 + (p_y/b)^2 - p_z = 0$$



Geometrical: Ray/Box Intersection

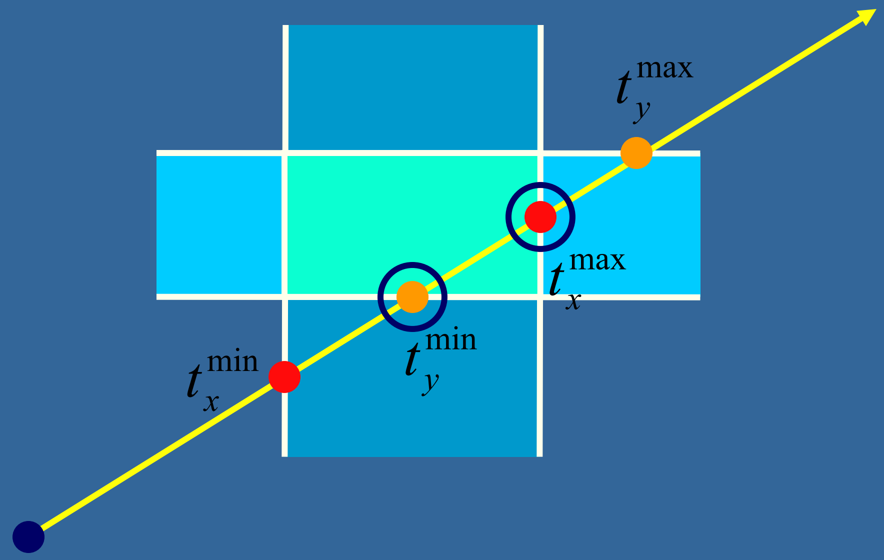
- Boxes and spheres often used as bounding volumes
- A slab is the volume between two parallell planes:

- A box is the logical intersection of three slabs (2 in 2D):



Geometrical: Ray/Box Intersection (2)

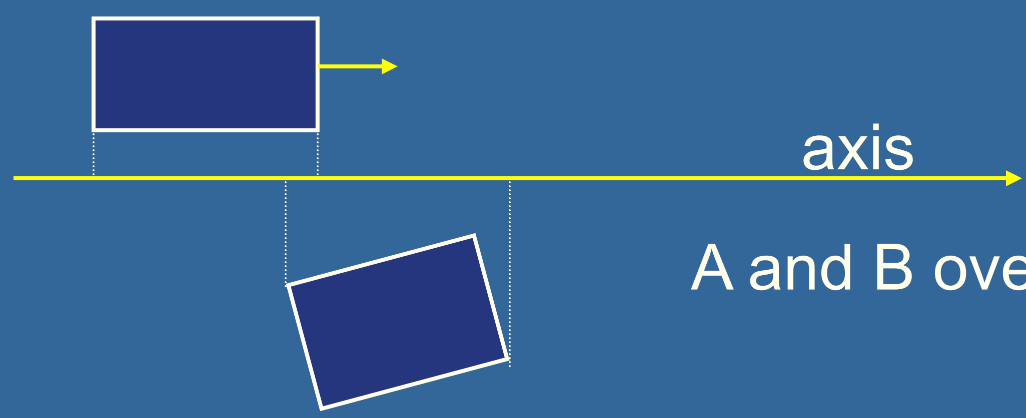
- Intersect the 2 planes of each slab with the ray



- Keep max of t^{min} and min of t^{max}
- If $t^{min} < t^{max}$ then we got an intersection
- Special case when ray parallell to slab

Separating Axis Theorem (SAT) Page 563 in book

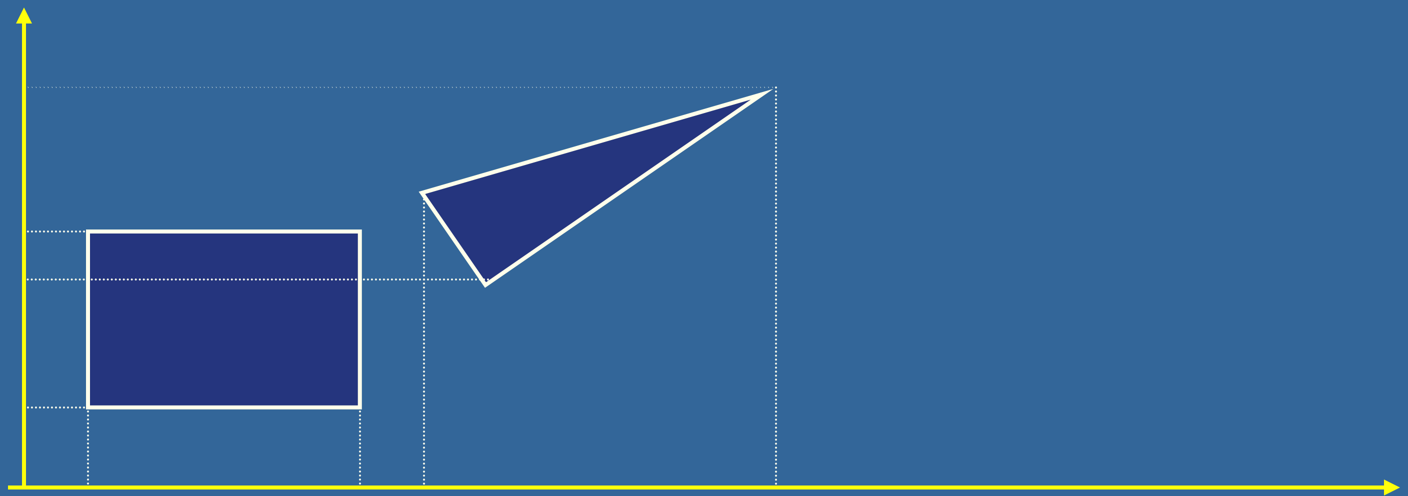
- Two convex polyhedrons, A and B, are disjoint if any of the following axes separate the objects’ projections:
  - A face normal of A
  - A face normal of B
  - Any $edge_A \times edge_B$ ($\times$ is crossproduct)



A and B overlaps on this axis

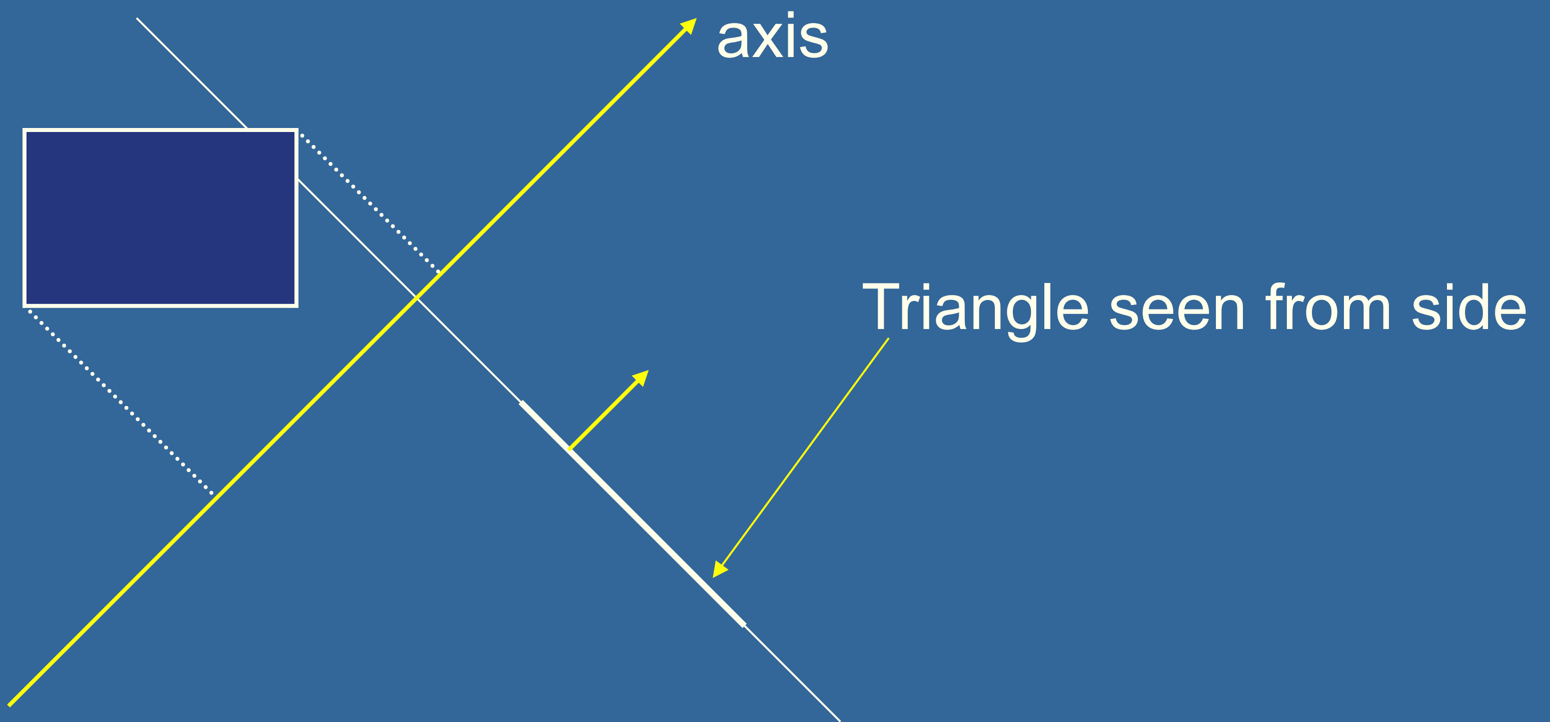
SAT example: Triangle/Box

- E.g an axis-aligned box and a triangle
- **1)** test the axes that are face normals of **the box:**
  - That is, x,y, and z



Triangle/Box with SAT (2)

- Assume that they overlapped on x,y,z
- Must continue testing
- **2)** face normal of **the triangle**



Triangle/Box with SAT (3)

- If still no separating axis has been found…
- **3**) Test axes that are cross products of the edges: $\mathbf{t}=\mathbf{e}_{box} \times \mathbf{e}_{triangle}$
- Example:
  - x-axis from box: $\mathbf{e}_{box}=(1,0,0)$
  - $\mathbf{e}_{triangle}=\mathbf{v}_1-\mathbf{v}_0$
- Test up to all such combinations
  - If there is at least one separating axis, then the objects do not collide
  - Else they do overlap

Rules of Thumb for Intersection Testing

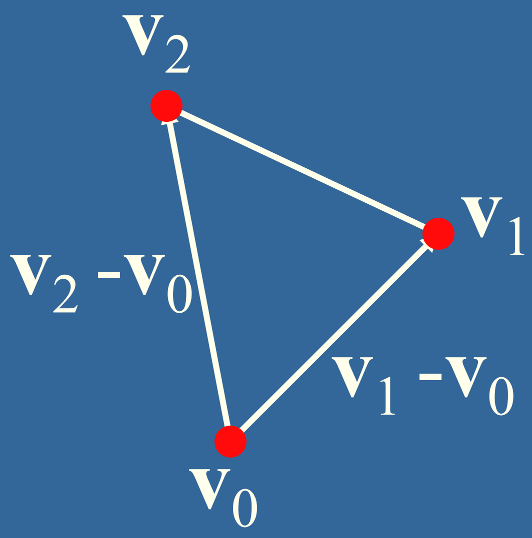
- Acceptance and rejection test
  - Try them early on to make a fast exit
- Postpone expensive calculations if possible
- Use dimension reduction
  - E.g. 3 one-dimensional tests instead of one complex 3D test, or 2D instead of 3D
- Share computations between objects if possible

Another analytical example: Ray/Triangle in detail

- Ray: $\mathbf{r}(t)=\mathbf{o}+t\mathbf{d}$
- Triangle vertices: $\mathbf{v}_0$, $\mathbf{v}_1$, $\mathbf{v}_2$
- A point in the triangle:

$$\mathbf{t}(u,v) = \mathbf{v}_0 + u(\mathbf{v}_1 - \mathbf{v}_0) + v(\mathbf{v}_2 - \mathbf{v}_0)$$

where $[u,v>=0,\ u+v<=1]$ is inside triangle



- Set $\mathbf{t}(u,v)=\mathbf{r}(t)$, and solve for t, u, v:

$$\mathbf{v}_0+u(\mathbf{v}_1 - \mathbf{v}_0) + v(\mathbf{v}_2 - \mathbf{v}_0) = \mathbf{o}+t\mathbf{d}$$

$$\Rightarrow -t\mathbf{d} + u(\mathbf{v}_1 - \mathbf{v}_0) + v(\mathbf{v}_2 - \mathbf{v}_0) = \mathbf{o}-\mathbf{v}_0$$

$$\Rightarrow [-\mathbf{d}, (\mathbf{v}_1 - \mathbf{v}_0), (\mathbf{v}_2 - \mathbf{v}_0)]\ [t, u, v]^T = \mathbf{o}-\mathbf{v}_0$$

$$\begin{pmatrix} | & | & | \\ -\mathbf{d} & \mathbf{v}_1 - \mathbf{v}_0 & \mathbf{v}_2 - \mathbf{v}_0 \\ | & | & | \end{pmatrix} \begin{pmatrix} t \\ u \\ v \end{pmatrix} = \begin{pmatrix} | \\ \mathbf{o} - \mathbf{v}_0 \\ | \end{pmatrix}$$

$$\mathbf{Ax}=\mathbf{b}$$
$$\mathbf{x}=\mathbf{A}^{-1}\mathbf{b}$$

BONUS

Ray/Triangle (1)

$$\begin{pmatrix} | & | & | \\ -\mathbf{d} & \mathbf{v}_1-\mathbf{v}_0 & \mathbf{v}_2-\mathbf{v}_0 \\ | & | & | \end{pmatrix}\begin{pmatrix} t \\ u \\ v \end{pmatrix} = \begin{pmatrix} | \\ \mathbf{o}-\mathbf{v}_0 \\ | \end{pmatrix}$$

- Solve for *t,u,v* using Cramer's rule for a system of *n* linear equations with *n* unknowns: $A\mathbf{x} = \mathbf{b}$

Cramer's rule:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} j \\ k \\ l \end{bmatrix} \Rightarrow x = \frac{\begin{vmatrix} j & b & c \\ k & e & f \\ l & h & i \end{vmatrix}}{\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix}},\quad y = \frac{\begin{vmatrix} a & j & c \\ d & k & f \\ g & l & i \end{vmatrix}}{\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix}},\ \text{and}\ z = \frac{\begin{vmatrix} a & b & j \\ d & e & k \\ g & h & l \end{vmatrix}}{\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix}}.$$

Simplify our equation system by setting:

$$\mathbf{e}_1 = \mathbf{v}_1 - \mathbf{v}_0 \qquad \mathbf{e}_2 = \mathbf{v}_2 - \mathbf{v}_0 \qquad \mathbf{s} = \mathbf{o} - \mathbf{v}_0$$

$$\Rightarrow \begin{pmatrix} | & | & | \\ -\mathbf{d} & \mathbf{e}_1 & \mathbf{e}_2 \\ | & | & | \end{pmatrix}\begin{pmatrix} t \\ u \\ v \end{pmatrix} = \begin{pmatrix} | \\ \mathbf{s} \\ | \end{pmatrix}$$

Cramer's rule gives:

$$\begin{pmatrix} t \\ u \\ v \end{pmatrix} = \frac{1}{\det(-\mathbf{d}, \mathbf{e}_1, \mathbf{e}_2)}\begin{pmatrix} \det(\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2) \\ \det(-\mathbf{d}, \mathbf{s}, \mathbf{e}_2) \\ \det(-\mathbf{d}, \mathbf{e}_1, \mathbf{s}) \end{pmatrix}$$

BONUS

Ray/Triangle (2)

$$\begin{pmatrix} t \\ u \\ v \end{pmatrix} = \frac{1}{\det(-\mathbf{d}, \mathbf{e}_1, \mathbf{e}_2)} \begin{pmatrix} \det(\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2) \\ \det(-\mathbf{d}, \mathbf{s}, \mathbf{e}_2) \\ \det(-\mathbf{d}, \mathbf{e}_1, \mathbf{s}) \end{pmatrix}$$

- To compute determinant

Use this fact: $\det(\mathbf{a},\mathbf{b},\mathbf{c}) = (\mathbf{a}\times\mathbf{b})\cdot\mathbf{c} = -(\mathbf{a}\times\mathbf{c})\cdot\mathbf{b}$

This gives:

$$\begin{pmatrix} t \\ u \\ v \end{pmatrix} = \frac{1}{(\mathbf{d}\times\mathbf{e}_2)\cdot\mathbf{e}_1} \begin{pmatrix} (\mathbf{s}\times\mathbf{e}_1)\cdot\mathbf{e}_2 \\ (\mathbf{d}\times\mathbf{e}_2)\cdot\mathbf{s} \\ (\mathbf{s}\times\mathbf{e}_1)\cdot\mathbf{d} \end{pmatrix}$$

- Share factors to speed up computations:

$$\mathbf{p} = \mathbf{d}\times\mathbf{e}_2$$
$$a = \mathbf{p}\cdot\mathbf{e}_1$$
$$f = 1/a$$

- Compute as little as possible. Then test.

Compute $u = f(\mathbf{p}\cdot\mathbf{s})$

Then test valid bounds:

```
if (u<0 or u>1) exit;
```

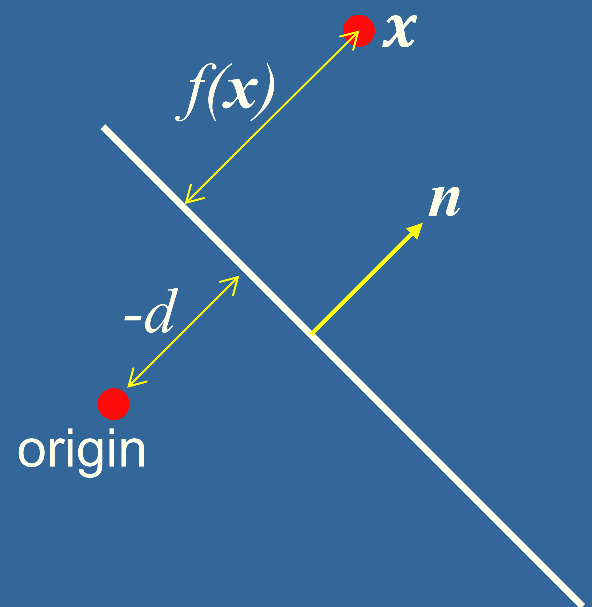
The Plane Equation

$$\text{Plane}: \quad \pi : \mathbf{n} \cdot \mathbf{p} + d = 0$$

If $\boldsymbol{n} \cdot \boldsymbol{x} + d = 0$, then $\boldsymbol{x}$ lies in the plane.

The function $f(\mathbf{x}) = \mathbf{n} \cdot \mathbf{x} + d$ gives the signed distance of $\mathbf{x}$ from the plane. ($\mathbf{n}$ should be normalized.)

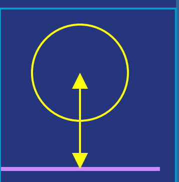
- $f(\mathbf{x})>0$ means above the plane
- $f(\mathbf{x})<0$ means below the plane

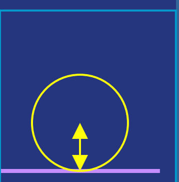


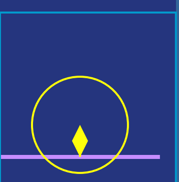
-d is how far the origin is behind the plane

Sphere/Plane
Box/Plane

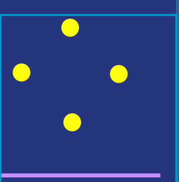
Plane: $\pi : \mathbf{n} \cdot \mathbf{p} + d = 0$
Sphere: $\mathbf{c} \quad r$
AABB: $\mathbf{b}^{\min} \quad \mathbf{b}^{\max}$







- Sphere: compute $f(\mathbf{c}) = \mathbf{n} \cdot \mathbf{c} + d$
- $f(\mathbf{c})$ is the signed distance ($\mathbf{n}$ normalized)
- $\text{abs}(f(\mathbf{c})) > \text{r}$ no collision
- $\text{abs}(f(\mathbf{c})) = \text{r}$ sphere touches the plane
- $\text{abs}(f(\mathbf{c})) < \text{r}$ sphere intersects plane

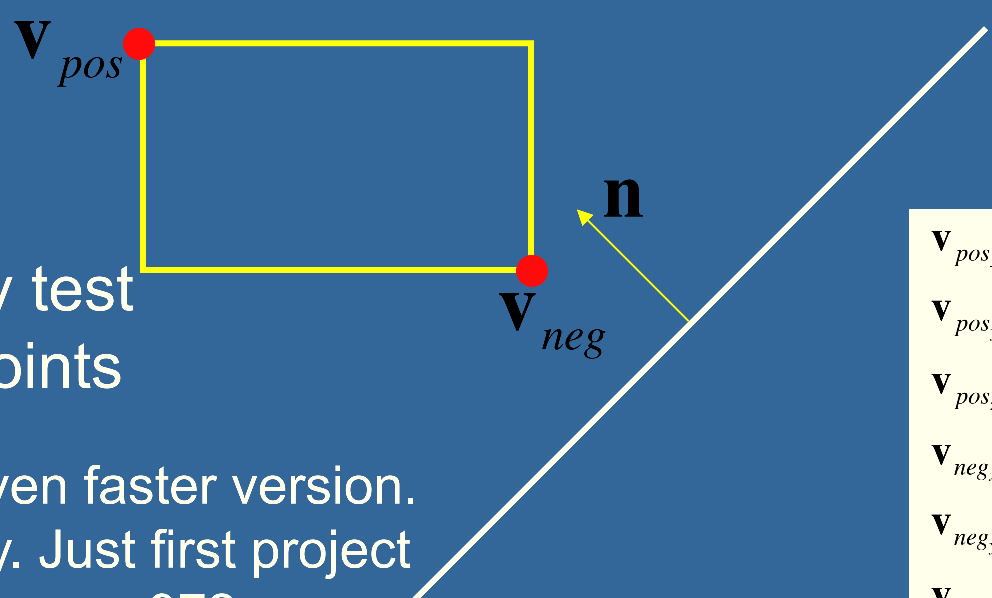


- Box: insert all 8 corners
- If all f's have the same sign, then all points are on the same side, and no collision

$$\text{Plane}: \quad \pi: \mathbf{n}\cdot\mathbf{p} + d = 0$$
$$\text{Sphere}: \quad \mathbf{c} \qquad r$$
$$\text{Box}: \quad \mathbf{b}^{\min} \qquad \mathbf{b}^{\max}$$

AABB/plane

- The smart way (shown in 2D)
- Find the two vertices that have the most positive and most negative value when tested againt the plane



Need only test the red points

$$\mathbf{v}_{pos_x} = (\mathbf{n}_x > 0)\,?\,\mathbf{b}_{\max_x} : \mathbf{b}_{\min_x}$$
$$\mathbf{v}_{pos_y} = (\mathbf{n}_y > 0)\,?\,\mathbf{b}_{\max_y} : \mathbf{b}_{\min_y}$$
$$\mathbf{v}_{pos_z} = (\mathbf{n}_z > 0)\,?\,\mathbf{b}_{\max z} : \mathbf{b}_{\min_z}$$
$$\mathbf{v}_{neg_x} = (\mathbf{n}_x < 0)\,?\,\mathbf{b}_{\max_x} : \mathbf{b}_{\min_x}$$
$$\mathbf{v}_{neg_y} = (\mathbf{n}_y < 0)\,?\,\mathbf{b}_{\max_y} : \mathbf{b}_{\min_y}$$
$$\mathbf{v}_{neg_z} = (\mathbf{n}_z < 0)\,?\,\mathbf{b}_{\max z} : \mathbf{b}_{\min_z}$$

See page 970 for even faster version.
OBB almost as easy. Just first project **n** on OBB's axes – see p: 972

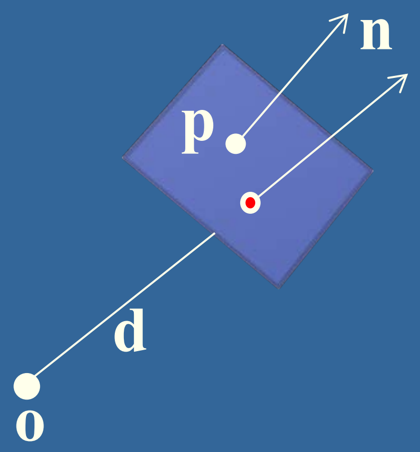
Ray/Plane Intersections

- Ray: r(t)=**o**+t**d**
- Plane: **n•x** + d = 0; (d=-**n•p**)
- Set **x**=r(t):

  n•(o+t**d)** + d = 0

  n•o+t(**n•d)** + d = 0

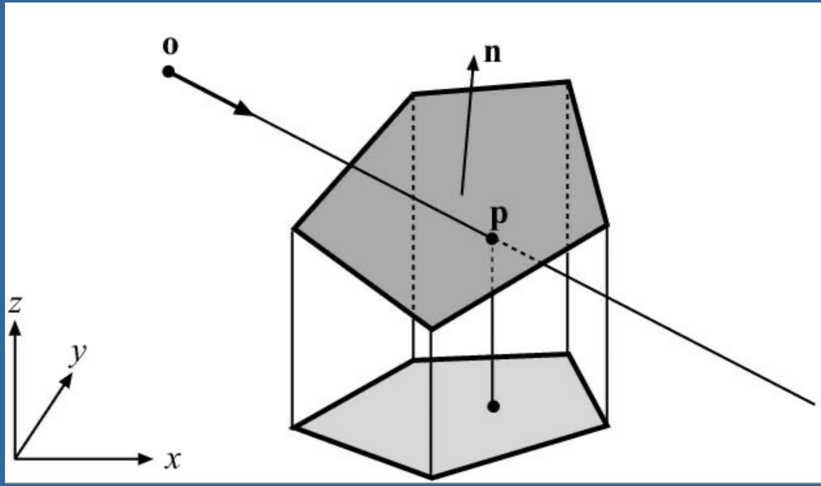
  t = (–d –**n•o) / (n•d)**

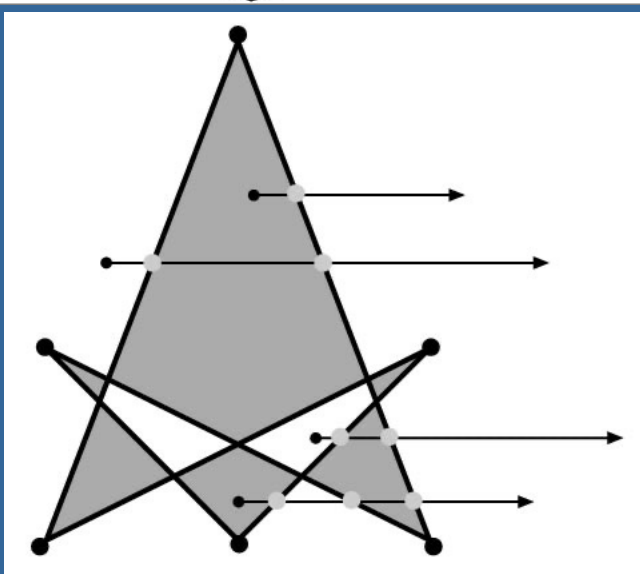



```
Vec3f rayPlaneIntersect(vec3f o,dir, n, d)
{
        float t=(-d-n.dot(o)) / (n.dot(dir));
        return o + dir*t;
}
```

Ray/Polygon: very briefly

- Intersect ray with polygon plane
- Project from 3D to 2D
- How?
- Find $\max(|n_x|,|n_y|,|n_z|)$
- Skip that coordinate!
- Then, count crossing in 2D

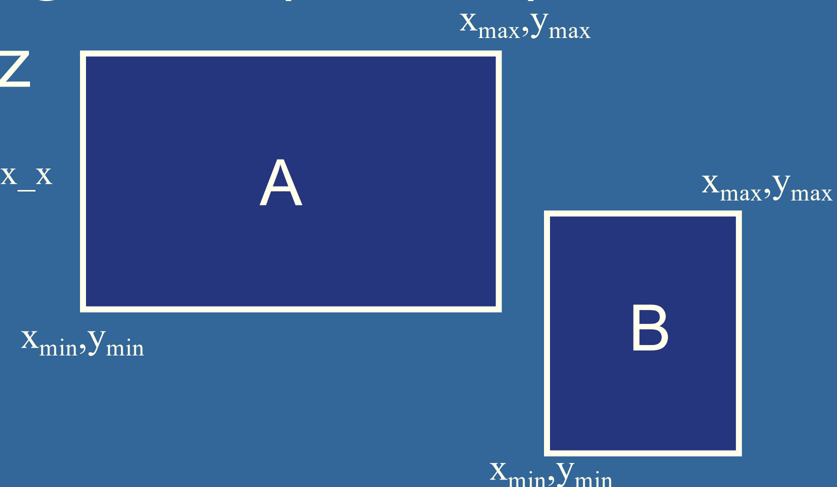




Volume/Volume tests

- Used in collision detection
- Sphere/sphere
  - Compute squared distance between sphere centers, and compare to $(r_1+r_2)^2$
- Axis-Aligned Bounding Box (AABB)
  - Test in 1D for x,y, and z

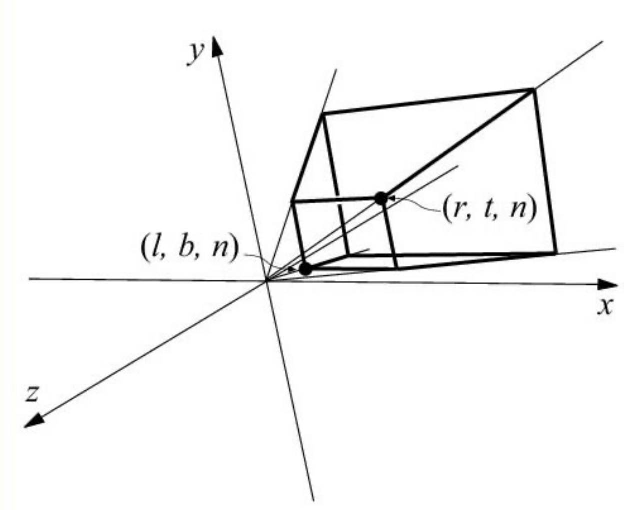
    If $B_{min\_x} > A_{max\_x}$ or $A_{min\_x} > B_{max\_x}$
    => no intersection.
    … same with y,z …



- Oriented Bounding boxes
  - Use SAT [details in book]

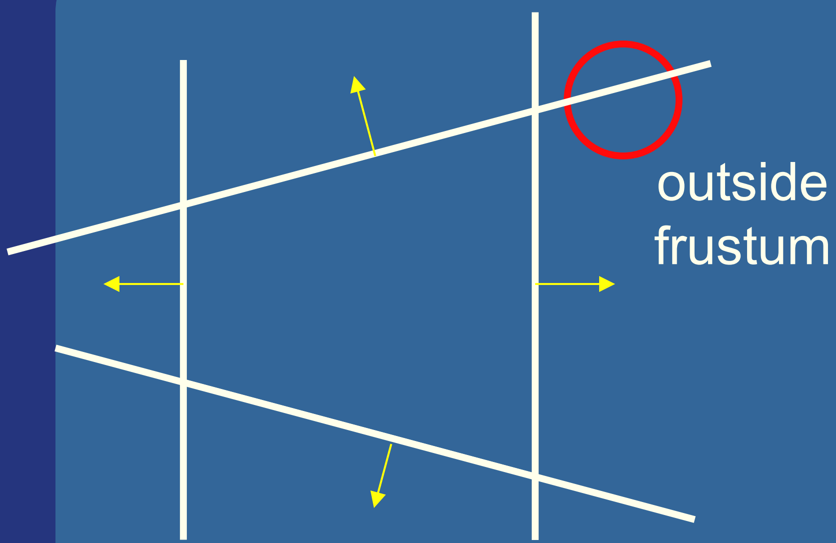
If $A_{min\_x} > B_{max\_x}$ or
$A_{min\_y} > B_{max\_y}$ or
$A_{min\_z} > B_{max\_z}$ or
$B_{min\_x} > A_{max\_x}$ or
$B_{min\_y} > A_{max\_y}$ or
$B_{min\_z} > A_{max\_z}$
return no_intersection
Else
return intersection.

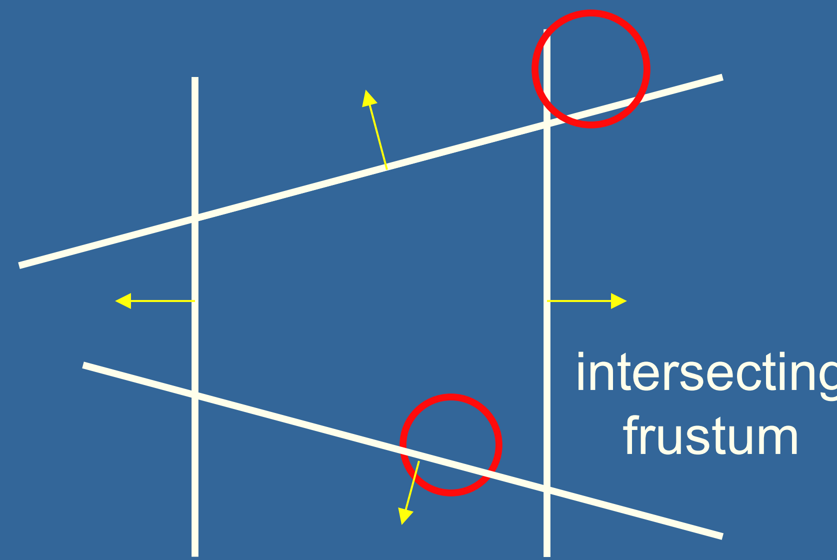
View frustum testing



- View frustum is 6 planes:
- Near, far, right, left, top, bottom
- Get their plane equations from projection matrix
  - Let all positive half spaces be outside frustum
  - p. 983-984.
- Sphere/frustum common approach:
  - Test sphere against each of the 6 frustum planes:
    - If outside the plane => no intersection
    - If intersecting the plane or inside, continue
  - If not outside after all six planes, then conservatively concider sphere as inside or intersecting
- Example follows…

View frustum testing example





- Not exact test, but not incorrect
  - A sphere that is reported to be inside, can actually be outside
  - Not vice versa
- Similar frustum test for boxes

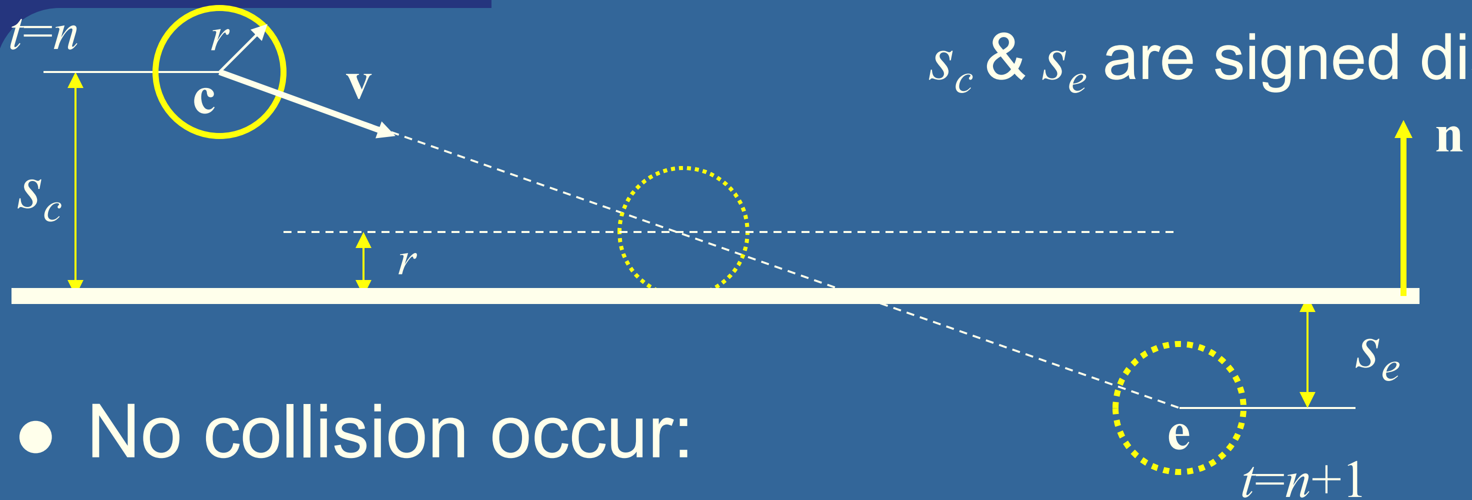
Dynamic Intersection Testing
[In book: 620-628]

- Testing is often done every rendered frame, i.e., at discrete time intervals
- Therefore, you can get ”quantum effects”

- Dynamic testing deals with this
- Is more expensive
- Deals with a time interval: time between two frames

Dynamic intersection testing Sphere/Plane

BONUS



s_c & s_e are signed distances

- No collision occur:
  - If they are on the same side of the plane ($s_c s_e > 0$)
    - and: $|s_c|>r$ and $|s_e|>r$
- Otherwise, sphere can move $|s_c|-r$
- Time of collision:

$$t_{cd} = n + \frac{s_c - r}{s_c - s_e}$$

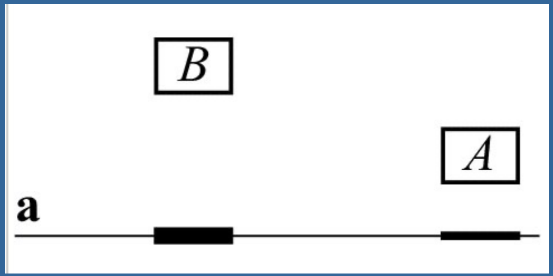
s_e is signed distance

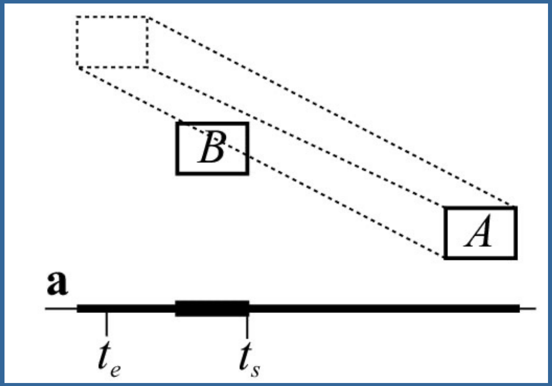
- Response: reflect $\mathbf{v}$ around $\mathbf{n}$, and move: $(1-t_{cd})\mathbf{r}$ ($\mathbf{r}$=refl vector)

BONUS

Dynamic Separating Axis Theorem

- SAT: tests one axis at a time for overlap





- Same with DSAT, but:
  - Use a relative system where B is fixed
    - i.e., compute A's relative motion to B.
  - Need to adjust A's projection on the axis so that the interval moves on the axis as well
- Need to test same axes as with SAT
- Same criteria for overlap/disjoint:
  - If no overlap on axis => disjoint
  - If overlap on all axes => objects overlap

Exercises

- Create a function (by writing code on paper) that tests for intersection between:
  - two spheres
  - a ray and a sphere
  - view frustum and a sphere

What you need to know

- Analytic test:
  - Be able to compute ray vs sphere or other similar formula
  - Ray/triangle, ray/plane
  - Point/plane, Sphere/plane,
  - Know expressions for ray, sphere, cylinder, plane, triangle
- Geometrical tests
  - Ray/box with slab-test
  - Ray/polygon (3D->2D)
  - box/plane
  - AABB/AABB
  - View frustum vs spheres/AABB:s/BVHs.
  - Separating Axis Theorem (SAT)
- Know what a dynamic test is