11-0: **Computational Geometry**

- Basic computational gemoetry
- Motivation: collision detection – when do do objects collide?
  - Static Collision – are two objects colliding this frame?
  - Dynamic Collision – will two moving objects collide within the next frame (why do we need this in addition to static collision?)

11-1: **AABB**

- Axis-Aligned Bounding Box
  - A Bounding box is a box that completely contains your model
  - An Axis aligned bounding box has its sides aligned with the global x,y and z axes
  - What would we need to store in memory to represent the axis aligned bounding box for a model?

11-2: **AABB**

- To represent an axis-aligned bounding box, we need two points (6 floats)
  - Minimum x, minimum y, minimum z point in the model
  - Maximum x, maximum y, maximum z point in the model

11-3: **AABB**

- There are other ways to represent an AABB
  - Minimum x,y,z and vector from min to max point
  - center of the AABB and vector to the maximum point
  - ... etc

11-4: **Computing AABB**

- Assuming that we are using the maximum point / minimum point representation of an AABB, how can we compute an AABB from the model?

11-5: **Computing AABB**

- Assuming that we are using the maximum point / minimum point representation of an AABB, how can we compute an AABB from the model?
  - Go through each point
  - Find the maximal and minimal x,y,z values

11-6: **Computing AABB**

```
void AABB::clear()
{
    mMin.x = mMin.y = mMin.z = MAXINT;
    mMax.x = mMax.y= mMax.z= MININT;
}

void AABB::merge(const Vector3 &p) {
   if (p.x < mMinimum.x) mMinimum.x  = p.x;
   if (p.x > mMaximum.x) mMaximum.x = p.x;
   if (p.y < mMinimum.y) mMinimum.y = p.y;
   if (p.y > mMaximum.y) mMaximum.y = p.y;
   if (p.z < mMinimum.z) mMinimum.z = p.z;
   if (p.z > mMaximum.z) mMaximum.z = p.z;
}
```

11-7: **AABB Translation**

- We have an AABB for a model
  - How should the AABB change if the model translates?

11-8: **AABB Translation**

- We have an AABB for a model
  - How should the AABB change if the model translates?
    - Just translate the minimum and maximum point in the AABB

11-9: **AABB Rotation**

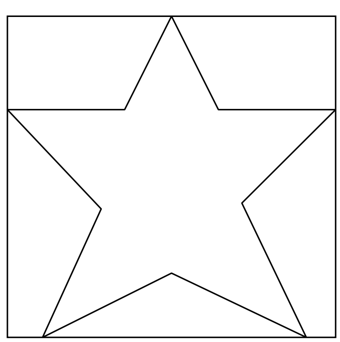
- We have an AABB for a model
  - How should the AABB change if the model rotates?

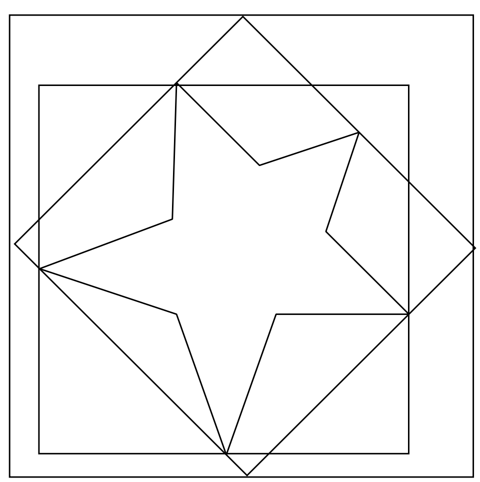
11-10: **AABB Rotation**

- We have an AABB for a model
  - How should the AABB change if the model rotates?
    - Can we just rotate the two minimum and maxiumum points? Why or why not?

11-11: **AABB Rotation**

- We have an AABB for a model
  - How should the AABB change if the model rotates?
    - Rotate each of the 8 points in the AABB (need to rotate all 8 – why?)
    - Construct a new AABB that contains these points
  - What are the problems with this method?





11-12: **AABB Rotation**

11-13: **AABB Rotation**

- How many multiplications are required for this transformation?

- Assuming we use a 3x3 rotational matrix for the transformation

11-14: **AABB Rotation**

- How many multiplications are required for this transformation?
  - Assuming we use a 3x3 rotational matrix for the transformation
  - Multiply all 8 points by the matrix
  - Each marix-vector multiplication requires 9 multiplications
  - Total: 8*9 = 72

11-15: **AABB Rotation**

- We can do better!
- Let's take a closer look at the multiplications we are doing:

$$[x', y', z'] = [x, y, z] \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

- We do this for all 8 points in our bounding box $[x_{min}, y_{min}, z_{min}]$, $[x_{min}, y_{min}, z_{max}]$, $[x_{min}, y_{max}, z_{min}]$, etc, and take the smallest and largest resulting x', y' and z' values

11-16: **AABB Rotation**

$$[x', y', z'] = [x, y, z] \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

$$x' = xm_{11} + ym_{21} + zm_{31}$$

- x can have the values x_{min} or x_{max}
- y can have the values y_{min} or y_{max}
- z can have the values z_{min} or z_{max}

How do we find the smallest possible x'? The largest? 11-17: **AABB Rotation**

$x'_{min} = 0, x_{max} = 0$
if $m_{11} \geq 0$
  $x'_{min} += x_{min} * m_{11}$
  $x'_{max} += x_{max} * m_{11}$
else
  $x'_{min} += x_{max} * m_{11}$
  $x'_{max} += x_{min} * m_{11}$
if $m_{21} \geq 0$
  $x'_{min} += y_{min} * m_{21}$
  $x'_{max} += y_{max} * m_{21}$
else
  $x'_{min} += y_{max} * m_{21}$
  $x'_{max} += y_{min} * m_{21}$
if $m_{31} \geq 0$
  $x'_{min} += z_{min} * m_{31}$
  $x'_{max} += z_{max} * m_{31}$
else
  $x'_{min} += z_{max} * m_{31}$
  $x'_{max} += z_{min} * m_{31}$

11-18: **AABB Rotation**

- Total number of multiplications required to find new max/min

  - 6 multiplications per dimension * 3 dimensions = 18
- Actually multiplying all 8 points by rotational matrix:
  - 9 multiplications per point transformation times
  - 8 points
  - = 72 multiplications

11-19: **AABB Rotation**

- How can we get the true AABB after a rotation?

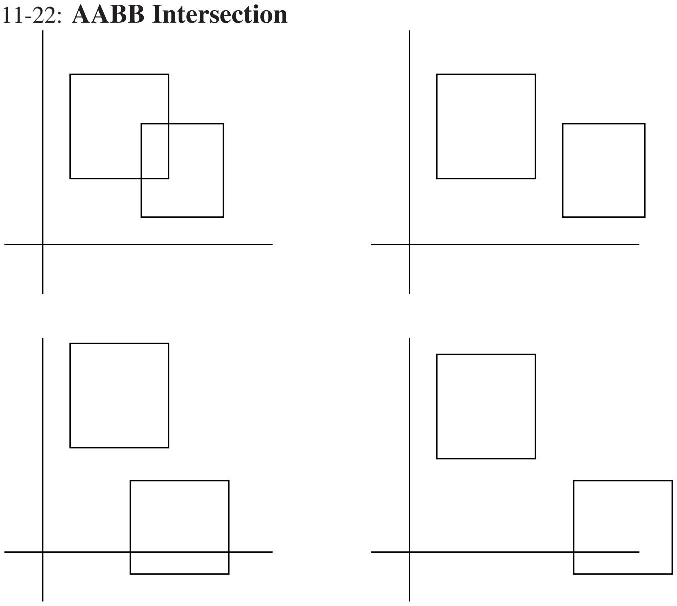
11-20: **AABB Rotation**

- How can we get the true AABB after a rotation?
  - Alas, we need to recalculate it
  - Requires going through all points in the model

11-21: **AABB Intersection**

- Given two AABBs, how can we determine if they intersect?
  - Start with the 2D case
  - See some examples ...

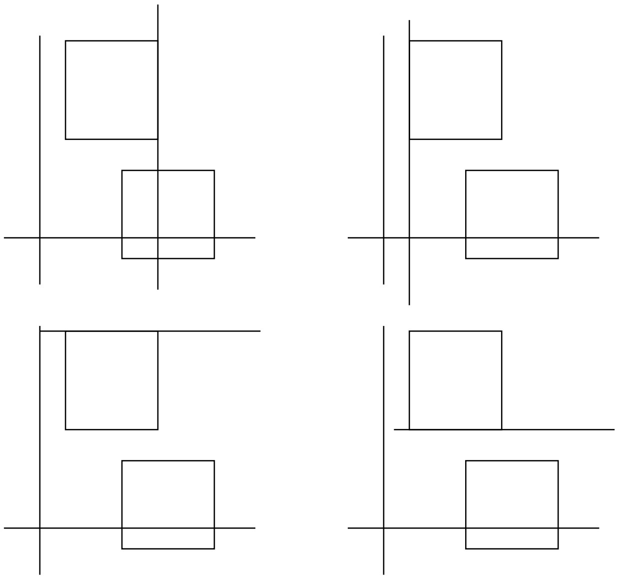
11-22: **AABB Intersection**



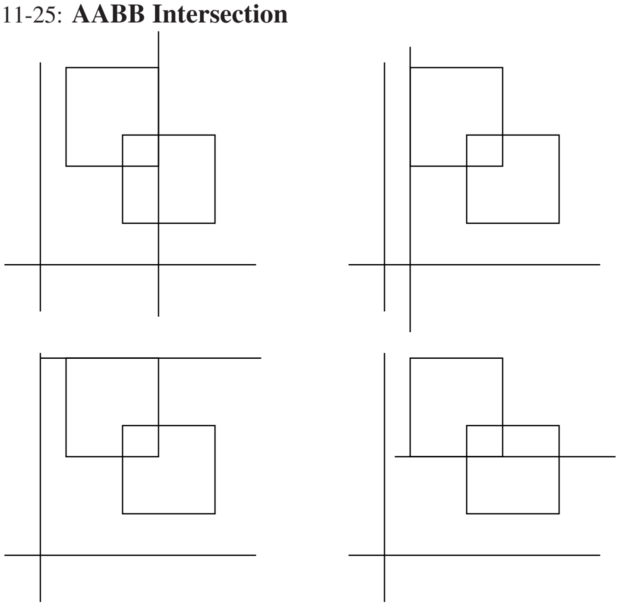
11-23: **AABB Intersection**

- Two 2D AABBs *don't* intesect if we can create a line such that one AABB is one one side of the line, and the other is on the other side of the line
- Since we are axis-aligend, we only need to test 4 lines – the 4 sides of one of the boxes

11-24: **AABB Intersection**



11-25: **AABB Intersection**



11-26: **AABB Intersection**

- Given two 2D AABBs B_1 and B_2, they *don't* intersect if:
    - $B_1.x.max < B_2.x.min$
    - $B_1.x.min > B_2.x.max$
    - $B_1.y.max < B_2.y.min$
    - $B_1.y.min > B_2.y.max$

11-27: **AABB Intersection**

- How can we extend this into 3D?

- Two 2D AABBs B_1 and B_2 *don't* intersect if there exists some line L such that B_1 is on one side of the line, and B_2 is on the other side of the line
- Two 3D AABBs B_1 and B_2 *don't* intersect if ...

11-28: **AABB Intersection**

- Two 3D AABBs B_1 and B_2 *don't* intersect if ...
  - There exists some plane P such that B_1 is on one side of P, and B_2 is on the other side of P.
  - How many planes do we need to check? Which ones?

11-29: **AABB Intersection**

- Given two 3D AABBs b_1 and b_2, they *don't* intersect if:
  - $B_1.x.max < B_2.x.min$
  - $B_1.x.min > B_2.x.max$
  - $B_1.y.max < B_2.y.min$
  - $B_1.y.min > B_2.y.max$
  - $B_1.z.max < B_2.z.min$
  - $B_1.z.min > B_2.z.max$

11-30: **OBB Defintion**

- Axis-aligned bounding boxes are good as a first step
  - Computing insersections is fast
  - Actual intersection =¿ AABB intersection
- AABBs are not great for final collision detection – not accurate enough
- We can do a little better with oriented bounding boxes

11-31: **OBB 2D**

- First, we'll look at Object Oriented Bounding Boxes in 2D
- Then, we'll extend to 3D

11-32: **OOBB Defintion**

- We will define an OOBB using 4 points
  - Points of each corner of the box
- Our techniques will work for any convex polygon, not just boxes

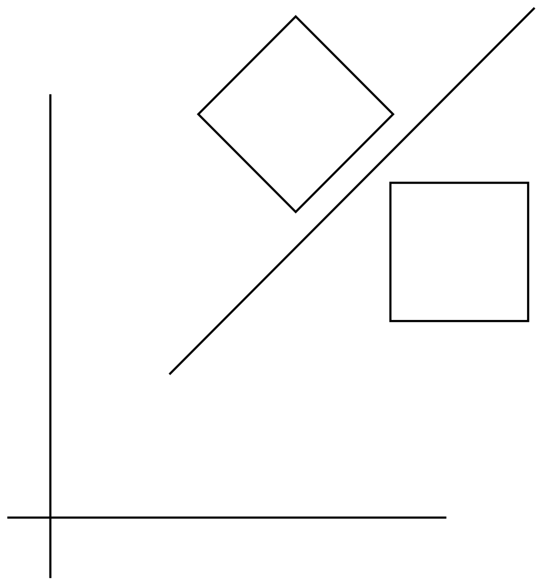
11-33: **OOBB Intersection**

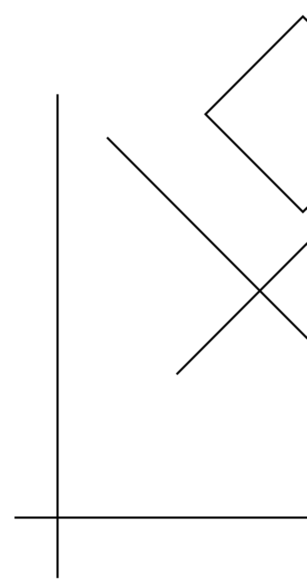
- We can define intersection of OOBB similarly to intersection of AABB:
  - If a line can be inserted between two OOBBs, such that one object is on one side of the line, and the other object is on the other side of the line, the boxes do not intersect

- If no such line exists, then the boxes do intersect

11-34: **OOBB Intersection**

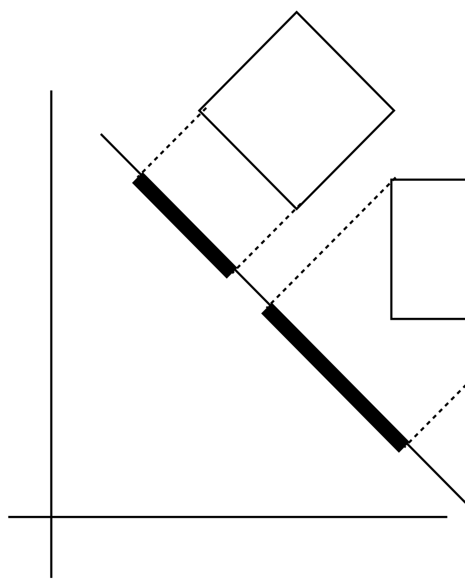
- Let's look at another way doing the same thing ...
  - Pick any line you like (axis of separation)
  - Project both boxes onto this axis
  - If the projections don't overlap, no intersection





11-35: **OOBB Intersection**

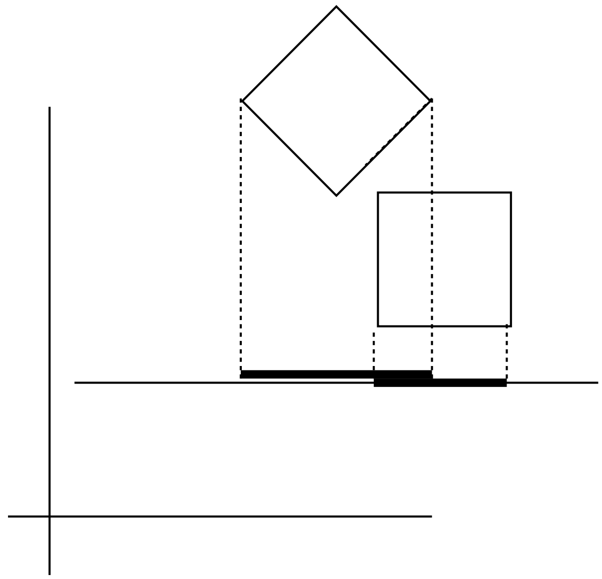
11-36: **OOBB Intersection**



11-37: **OOBB Intersection**

11-38: **OOBB Intersection**

- If there is some axis of separation with no overlap, no intersection
- However, if there is an axis of separation with overlap, may not be any intersection
- Need to pick the correct one!



11-39: **OOBB Intersection**

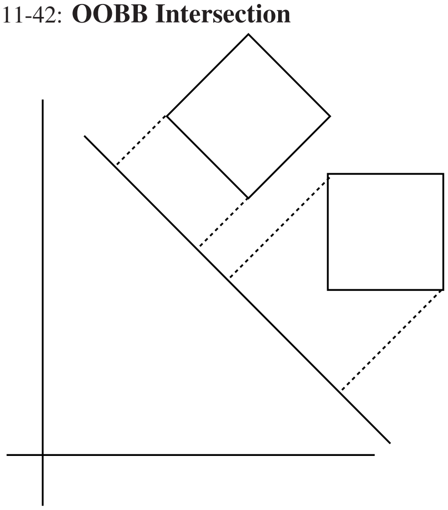
11-40: **OOBB Intersection**

- There are a infinite number of axes to try!
- Can't try them all!
- If there is an axis of separation, then it must be ...

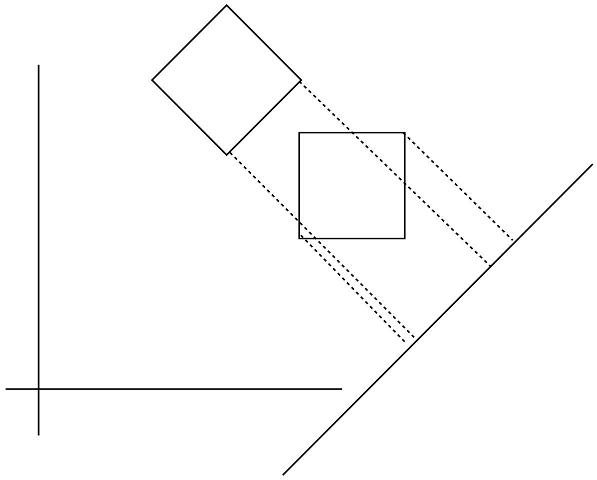
11-41: **OOBB Intersection**

- There are a infinite number of axes to try!
- Can't try them all!
- If there is an axis of separation, then it must be
  - Perpendicular to one of the edges
  - There are only 4 possibilities if we are working with OOBBs

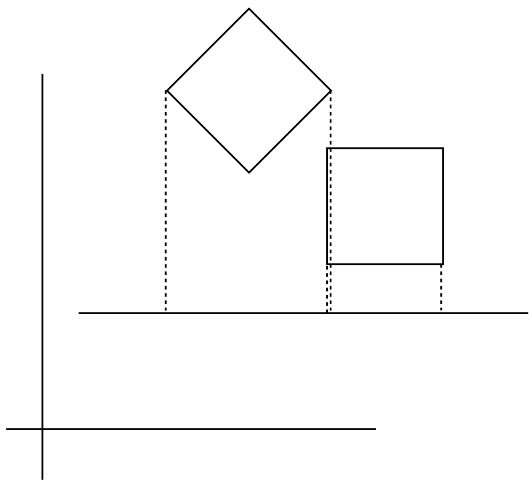
11-42: **OOBB Intersection**



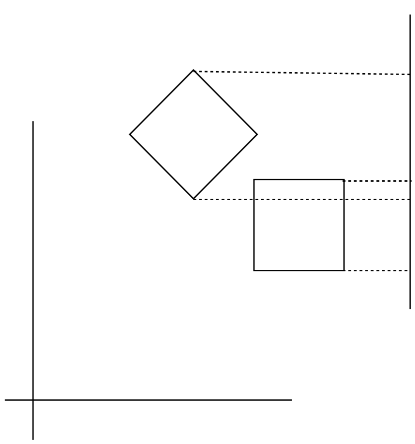
11-43: **OOBB Intersection**



11-44: **OOBB Intersection**



11-45: **OOBB Intersection**



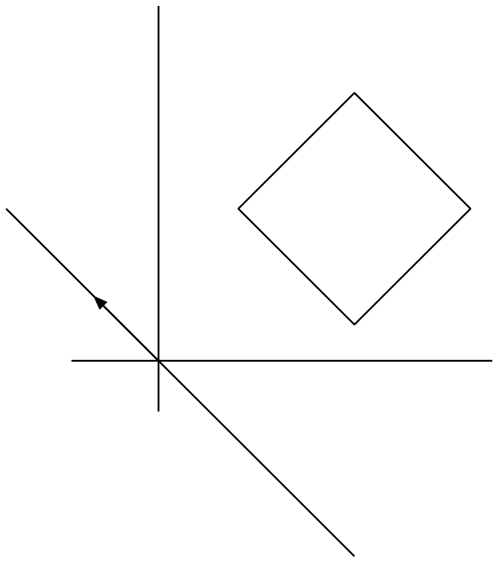
11-46: **OOBB Intersection**

- Pick an axis of separation (will need to try 4 in all)
- Project boxes onto axis
- See if there is overlap
- If there is no overlap on any axis, we can stop – no intersection

11-47: **OOBB Intersection**

- How do we project a box onto our axis?
    - Project each point onto axis
    - Find the maximum and minimum values

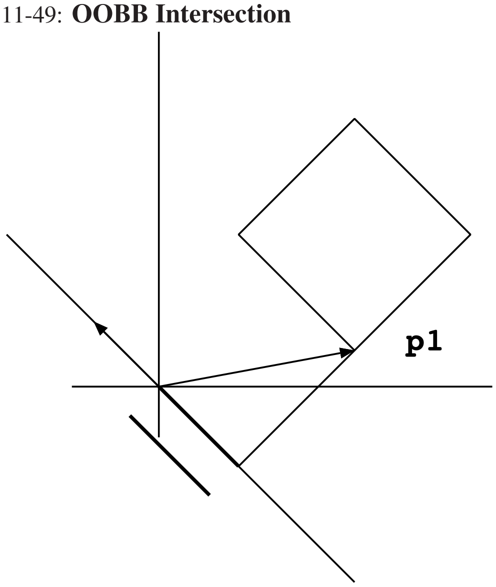
11-48: **OOBB Intersection**



Box: p1, p2, p3, p4

Axis: [x,y]
(normalized)

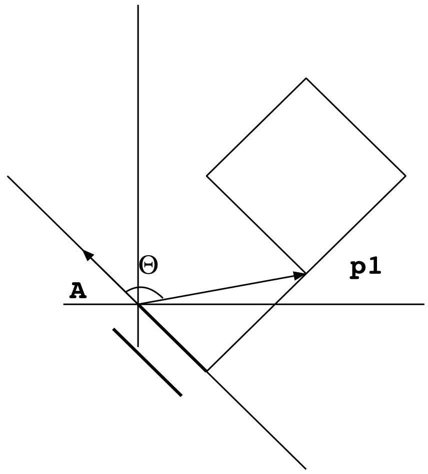
11-49: **OOBB Intersection**



Box: p1, p2, p3, p4

Axis: [x,y]
(normalized)

11-50: **OOBB Intersection**




```
Box: p1, p2, p3, p4

    Axis: [x,y]
      (normalized)


A * p1 = ||A|| ||p1|| cos Θ
 = ||p1|| cos θ
 = length of projection
```

11-51: **OOBB Intersection**

- Given a box p_1, p_2, p_3, p_4, what are the two axes of intersection?

11-52: **OOBB Intersection**

- Given a box p_1, p_2, p_3, p_4, what are the two axes of intersection?
  - Perpendicular to an edge
  - Vector parallel to an edge: $v = p_2 - p_1$
  - Vector perpendicular to $v = [v_x, v_y]$?

11-53: **OOBB Intersection**

- Given a box p_1, p_2, p_3, p_4, what are the two axes of intersection?
  - Perpendicular to an edge
  - Vector parallel to an edge: $v = p_2 - p_1$
  - Vector perpendicular to $v = [v_x, v_y]$?
    - $[-v_y, v_x]$
  - Axis of intersection: $\frac{[-v_y, v_x]}{||[-v_y, v_x]||}$

11-54: **OOBB Intersection**

- Given two boxes $(p_{11}, p_{12}, p_{13}, p_{14})$ and $(p_{21}, p_{22}, p_{23}, p_{24})$
  - $v_{||} = p_{12} - p_{11}$
  - $a = \frac{[-v_{||y}, v_{||x}]}{||[-v_{||y}, v_{||x}]||}$
  - Calculate $a \cdot p_{11}$, $a \cdot p_{12}$,$a \cdot p_{13}$, $a \cdot p_{14}$, store minimum and maximum values
  - Calculate $a \cdot p_{21}$, $a \cdot p_{22}$, $a \cdot p_{23}$, $a \cdot p_{24}$, store minimum and maximum values
  - Check for overlap
  - Repeat for $p_{13} - p_{12}$, $p_{22} - p_{21}$, $p_{23} - p_{22}$

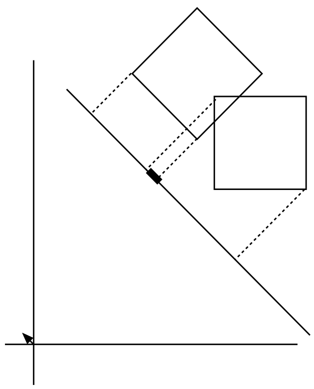
11-55: **OOBB MTD**

- How can we calculate the Minimum Translation Distance vector?
  - Smallest distance needed to move an object to no longer be colliding
  - Really handy for undoing collisions

11-56: **OOBB MTD**

- How can we calculate the Minimum Translation Distance vector?
  - Find the separating axis with the minimum overlap
  - Vector is along the separating axis, length equal to the overlap

11-57: **OOBB MTD**

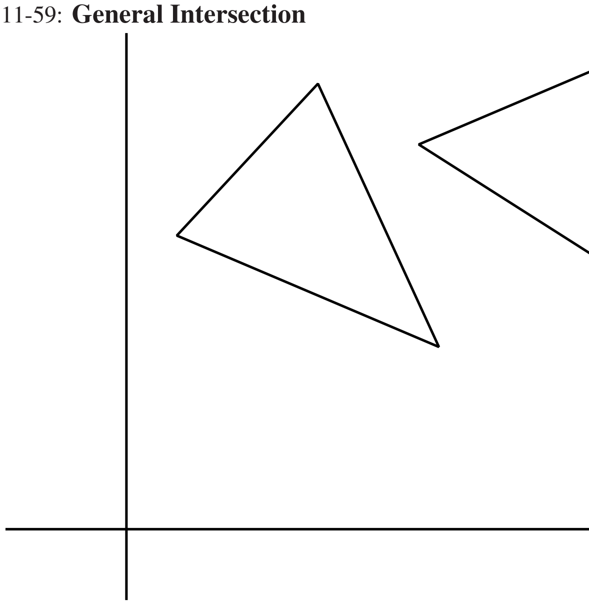


- We've already calculated the separating axis vector, and the amount of offset – multiply the offset size (scalar) by separating axis vector (already normalized)

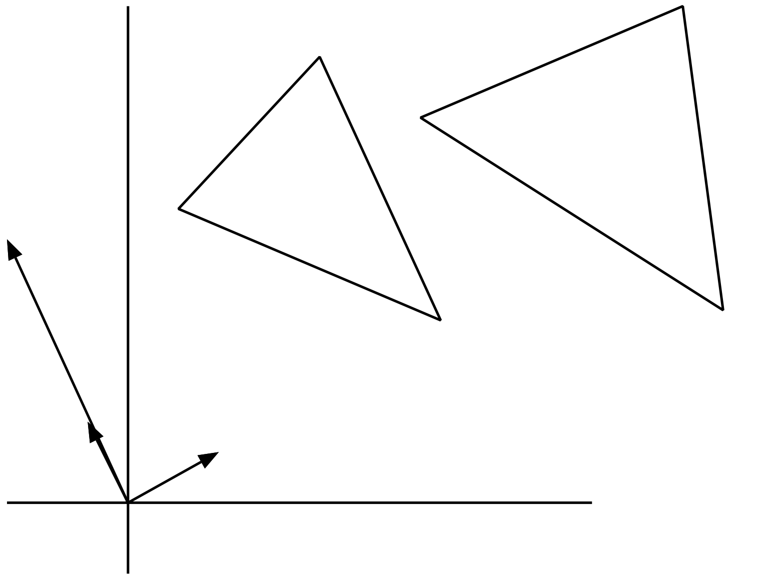
11-58: **General Intersection**

- This method works for any convex surface:
  - For axes of separation, try vector perpendicular to each edge

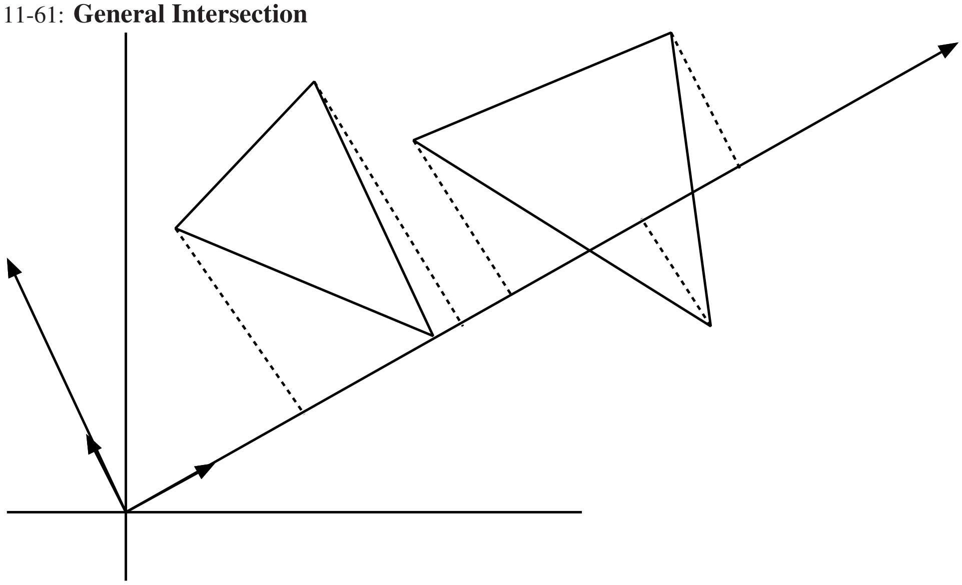
11-59: **General Intersection**



11-60: **General Intersection**

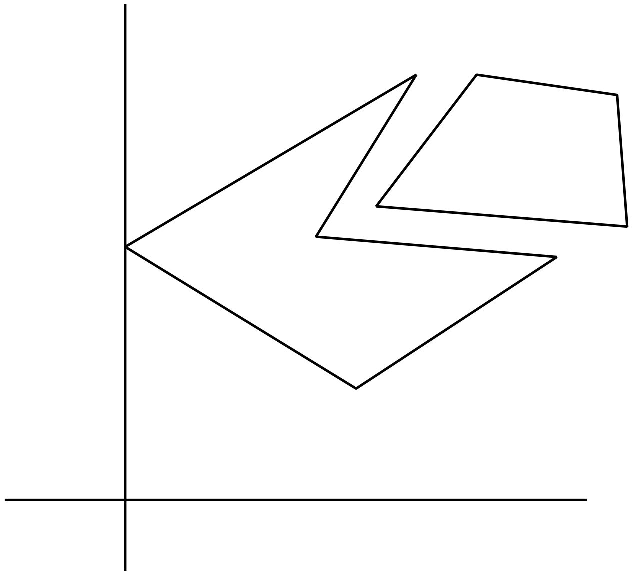


11-61: **General Intersection**

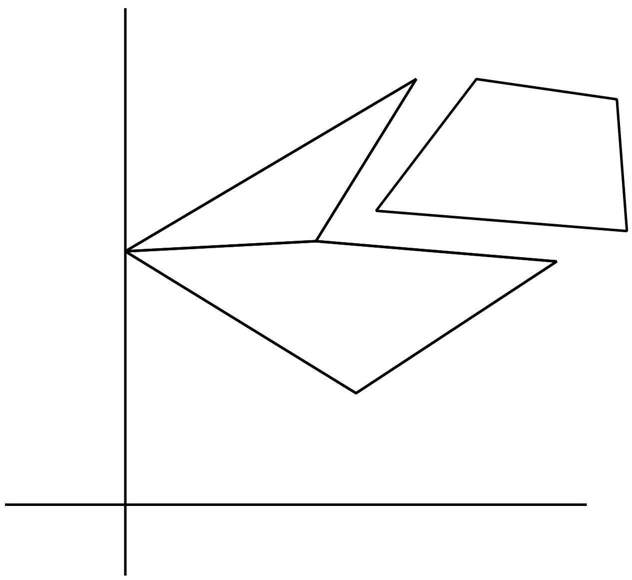


11-62: **General Intersection**

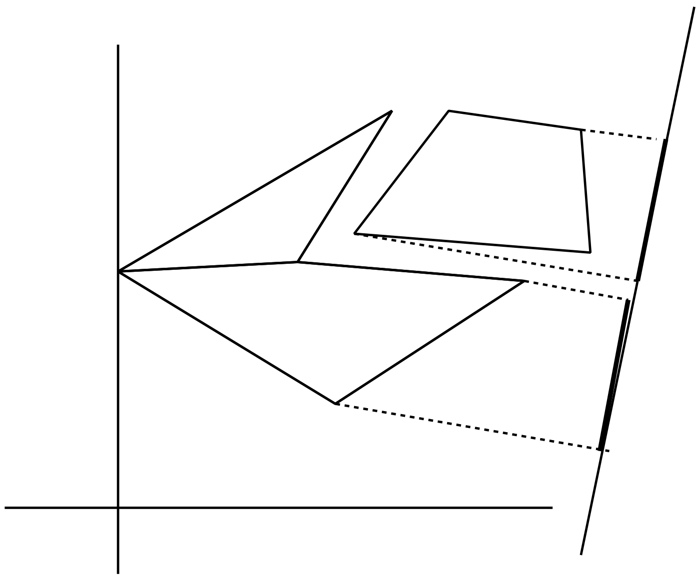
- Only works for convex objects, not concave



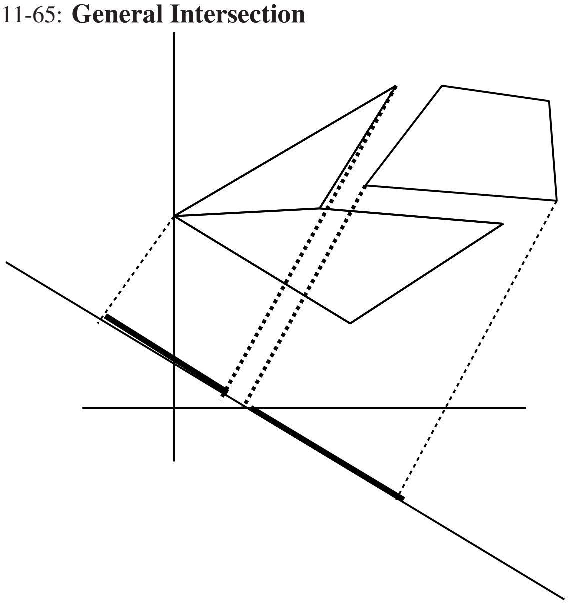
11-63: **General Intersection**



11-64: **General Intersection**



11-65: **General Intersection**



11-66: **OBB Defintion**

- OK, Back to 3D
- We can define an OOB as follows:
  - Extents (sometimes called half-extents) – distance from center of box to largest x,y,z point
  - Center point
  - Orientation (commonly a 3x3 matrix)

11-67: **OBB Intersection**

- We can define intersection of OBB similarly to intersection of AABB:

- If a plane can be inserted between two OBBs, such that one object is on one side of the plane, and the other object is on the other side of the plane, the boxes do not intersect
- If no such plane exists, then the boxes do intersect

11-68: **OBB Intersection**

- Let's look at another way doing the same thing ...
  - Pick any line you like (axis of separation)
  - Project both boxes onto this axis
  - If the projections don't overlap, no intersection

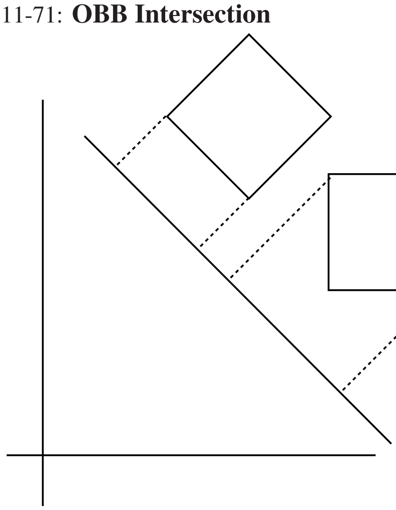
11-69: **OBB Intersection**

- If there is some axis of separation with no overlap, no intersection
- However, if there is an axis of separation with overlap, may not be any intersection
- Need to pick the correct one!

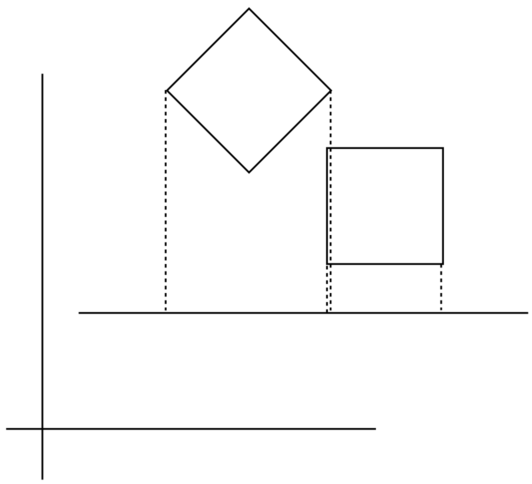
11-70: **OBB Intersection**

- There are a infinite number of axes to try!
- Can't try them all!
- Normals of each face are a good start

11-71: **OBB Intersection**



11-72: **OBB Intersection**



11-73: **OBB Intersection**

- Finding an Axis of Separation
  - If there is an axis of separation, then one of the following will be an axis of separation:
    - Normals of each face (3 + 3) = 6 to try
    - Cross products of each pair of normals (3 * 3) = 9 to try

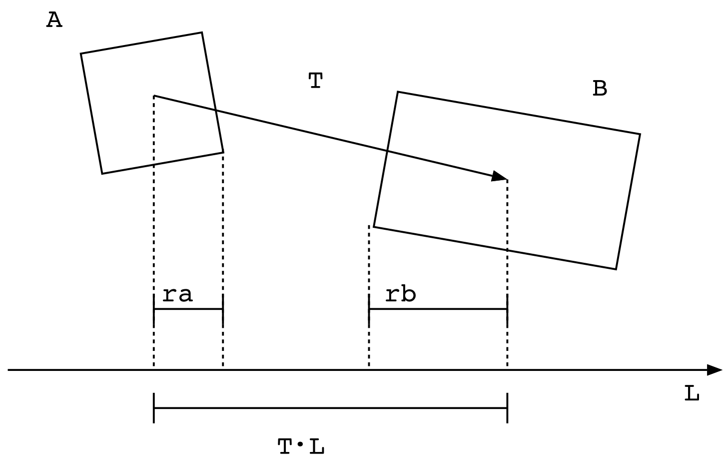
11-74: **OBB Intersection**

- So, logically we will:
  - Pick an axis of separation (will need to try 15 in all)
  - Project boxes onto axis
  - See if there is overlap

11-75: **OBB Intersection**

- Actually do something slightly different (but equivalent!)
  - Pick an axis
  - Calcluate size of the projected radius of each box
    - 1/2 the size of the box projected onto the axis
  - Take the line between the midpoints of the boxes, project it onto the axis as well
  - If the sum of the radii is smaller than the length of the projected distance between midpoints, no intersection

11-76: **OBB Intersection**




```
T = Center(B) - Center(A)  ra = radius of projection of A
L = Axis of separation     rb = radius of projection of B
```

11-77: **OBB Intersection**

- Given:
  - extents a_x, a_y, and a_z
  - Rotational matrix A
  - L: Axis of separation
  - T: Vector from center of box A to center of box B
- $r_a = a_x * |A_x \cdot L| + a_y * |A_y \cdot L| + a_z * |A_z \cdot L|$
- L is a separating axis if $T \cdot L > r_a + r_b$

11-78: **OBB Intersection**

- Need to try 15 different axes:
  - 3 normals of A
  - 3 normals of B
  - 9 cross normals
- Much easier of we calculate normals of A in A's reference frame, normals of B in B's reference frame, and so on.

11-79: **OBB Intersection**

- Given:
  - Rotational matrix M_A, that transforms a point in box A's space into world space
  - Rotational matrix M_B, that transforms a point in box B's space into world space
- How do we create a rotational matrix M, that transforms a point in B's space into A's space?

11-80: **OBB Intersection**

- Given:
  - Rotational matrix M_A, that transforms a point in box A's space into world space
  - Rotational matrix M_B, that transforms a point in box B's space into world space
- How do we create a rotational matrix M, that transforms a point in B's space into A's space?
  - $M = M_B M_A^T$

11-81: **OBB Intersection**

- Once we have M, that translates a point from Box B's space into box A's space, how can we go the other way?
- In terms of M, what is a matrix than translates a point from box A's space into box B's space?

11-82: **OBB Intersection**

- Once we have M, that translates a point from Box B's space into box A's space, how can we go the other way?
- In terms of M, what is a matrix than translates a point from box A's space into box B's space?
  - M^T
  - If we've calculated M, we don't need to calculate M^T – just reverse the accessing indices

11-83: **OBB Intersection**

- What we have:
  - Center point of A in world space, p_A
  - Transformation from A's space to world space, M_A
  - Extents of A in A's space: e_A
  - Center point of B in world space, p_B
  - Transformation from B's space to world space, M_B
  - Extents of B in B's space: e_B
- What is T, the vector from the center of A to the center of B in A's reference space?

11-84: **OBB Intersection**

- What we have:
  - Center point of A in world space, p_A
  - Transformation from A's space to world space, M_A
  - Extents of A in A's space: e_A
  - Center point of B in world space, p_B
  - Transformation from B's space to world space, M_B
  - Extents of B in B's space: e_B
- What is T, the vector from the center of A to the center of B in A's reference space?
  - $(p_B - p_a)M_A^T$

11-85: **OBB Intersection**

- For Axis of separation [1,0,0] in A's reference frame (what face(s) is this normal to?):
  - What is ra, the radius of A projected onto [1,0,0]?

11-86: **OBB Intersection**

- For Axis of separation [1,0,0] in A's reference frame (what face(s) is this normal to):
  - What is ra, the radius of A projected onto [1,0,0]?
    - e_{Ax} ($e_a[0]$)

11-87: **OBB Intersection**

- For Axis of separation [1,0,0] in A's reference frame:
  - What is rb, the radius of B projected onto [1,0,0] (in A's reference frame?
  - (recall that $r_b = \mathbf{e}_{bx} * |B_x \cdot L| + \mathbf{e}_{by} * |B_y \cdot L| + \mathbf{e}_{bz} * |B_z \cdot L|$)
  - Also note that we are doing all of this in A's reference frame, *not* the global reference frame

11-88: **OBB Intersection**

- For Axis of separation [1,0,0] in A's reference frame:
  - What is rb, the radius of B projected onto [1,0,0] (in A's reference frame?
  - (recall that $r_b = \mathbf{e}_{bx} * |B_x \cdot L| + \mathbf{e}_{by} * |B_y \cdot L| + \mathbf{e}_{bz} * |B_z \cdot L|$)
  - Also note that we are doing all of this in A's reference frame, *not* the global reference frame
    - B_x (in A's space) is the first row of $M_B M_A^T$
    - B_y (in A's space) is the second row of $M_B M_A^T$
    - B_z (in A's space) is the second row of $M_B M_A^T$

11-89: **OBB Intersection**

- For Axis of separation [1,0,0] in A's reference frame:
  - Given that the vector from A to B in A's reference frame is L
  - What is the projetion of L onto [1,0,0]

11-90: **OBB Intersection**

- For Axis of separation [1,0,0] in A's reference frame:
  - Given that the vector from A to B in A's reference frame is L
  - What is the projetion of L onto [1,0,0]
    - Just the first component of L, $L_x = L[0]$

11-91: **OBB Intersection**

- All calculations are in A's reference frame
  - $L = (p_b - p_a)M_A^T$
  - Length of projection of L into axis = $L_x = L[0]$
  - $ra = \mathbf{e}_{\mathbf{a}x} = \mathbf{e}_{\mathbf{a}}[0]$

  - $rb + \mathbf{e_b}_x|M_B M_A^T[0][0]| + \mathbf{e_b}_y|M_B M_A^T[1][0]| + \mathbf{e_b}_z|M_B M_A^T[2][0]$
  - If $L_x > ra + rb$, no intersection

11-92: **OBB Intersection**

- Do something similar for the other 14 possible axes for separation
  - 3 basis vectors of A (normals to A's faces)
  - 3 basis vectors of B (normals to B's faces)
  - 9 cross products of normals

11-93: **General Intersection**

- This method works for any convex surface:
  - For axes of separation, try the normals of all faces, and the cross-product of normals for all faces
  - Note that much of the math simplifies for OBB, general convex surfaces are a little more complicated
  - For concave surfaces, use the covex hull, or split into separate concave surfaces, and check each one

11-94: **More fun with Geometry**

- Recall that a plane can be described by the formula: $\mathbf{p} \cdot \mathbf{n} = d$ where
  - $\mathbf{n}$ is the normal vector of the plane (usally normalized)
  - $\mathbf{p}$ is a point p_x, p_y, p_z on the plane
  - d is a scalar that describes the offset of the plane

11-95: **More fun with Geometry**

- Given a plane, described by $\mathbf{n}$ and d, and a point q not on the plane
  - What is the shortest distance from q to the plane?

11-96: **More fun with Geometry**

- Given a plane, described by $\mathbf{n}$ and d, and a point q not on the plane
  - What is the shortest distance from q to the plane?
  - Let p be the point on the plane closest to q. The line from p to q must be along the normal of the plane (why?)

$$\mathbf{p} + a\mathbf{n} = \mathbf{q}$$

11-97: **More fun with Geometry**

- Given a plane, described by $\mathbf{n}$ and d, and a point q not on the plane
  - What is the shortest distance from q to the plane?
  - Let p be the point on the plane closest to q. The line from p to q must be along the normal of the plane (why?)

$$\begin{aligned}
\mathbf{p} + a\mathbf{n} &= \mathbf{q} \\
(\mathbf{p} + a\mathbf{n}) \cdot \mathbf{n} &= \mathbf{q} \cdot \mathbf{n} \\
\mathbf{p} \cdot \mathbf{n} + a\mathbf{n} \cdot \mathbf{n} &= \mathbf{q} \cdot \mathbf{n} \\
d + a &= \mathbf{q} \cdot \mathbf{n} \\
a &= \mathbf{q} \cdot \mathbf{n} - d
\end{aligned}$$

11-98: **More fun with Geometry**

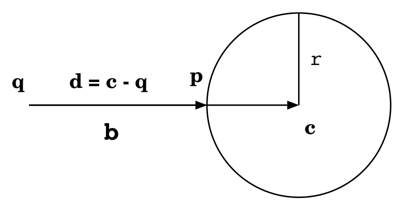
- Given a plane $\mathbf{n}$, d and a point $\mathbf{q}$, how do we find $\mathbf{p}$, the point on the plane closest to $\mathbf{q}$?

11-99: **More fun with Geometry**

- Given a plane $\mathbf{n}$, d and a point $\mathbf{q}$, how do we find $\mathbf{p}$, the point on the plane closest to $\mathbf{q}$?
  - We already know the distance from the point to the plane, and the fact the line from $\mathbf{p}$ to $\mathbf{q}$ must lie alone $\mathbf{n}$, so we have:
    - $\mathbf{q} = \mathbf{q} - (\mathbf{q} \cdot \mathbf{n} - d)\mathbf{n}$

11-100: **Closest point on a sphere**

- Given a sphere ($\mathbf{c}$, r) and a point $\mathbf{q}$, what is the point on the sphere closest to q?



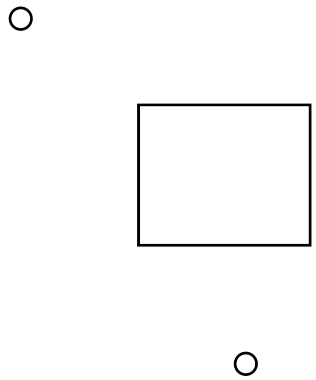
11-101: **Closest point on a sphere**

$$\begin{aligned}
b &= \frac{||d|| - r}{||d||} d \\
p &= q + \frac{||d|| - r}{||d||} d
\end{aligned}$$

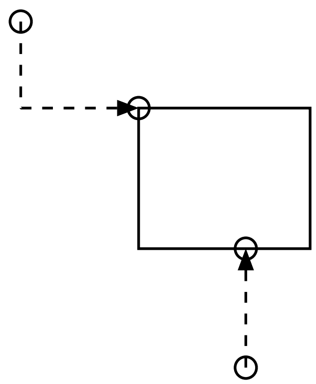
11-102: **Closest point in AABB**

- Given
  - AABB (minX, minY, minZ, maxX, maxY, maxZ)
  - point q
- What is the point closest to q that is inside the AABB?

11-103: **Closest point in AABB**

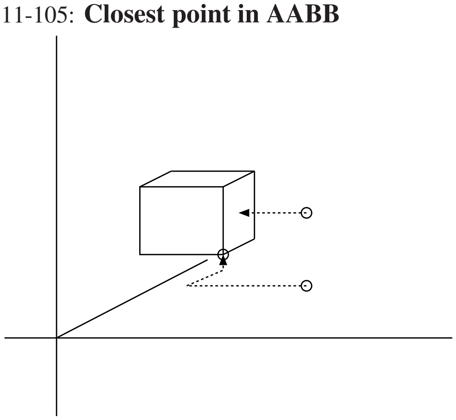


11-104: **Closest point in AABB**



- “Push in” the point along each axis

11-105: **Closest point in AABB**



11-106: **Closest point in AABB**

```
if (q.x < minX)
   p.x = minX
else if (q.x > maxX)
   p.x = maxX
else
   p.x = q.x
if (q.y < minY)
   p.y = minY
else if (q.y > maxY)
   p.y = maxY
else
   p.y = q.y
if (q.z < minZ)
   p.z = minZ
else if (q.z > maxZ)
```

```
   p.z = maxZ
else
   p.z = q.z
```

11-107: **Closest Point in OBB**

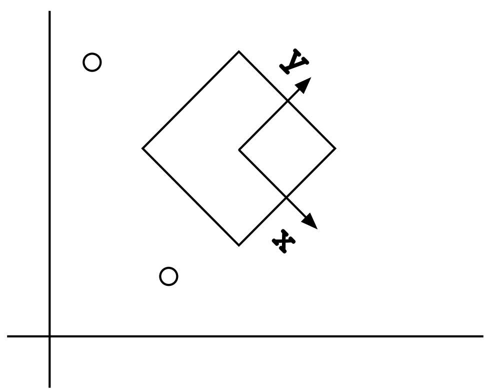
- Given
  - OBB (Origin, extents, basis vectors)
  - point q
- What is the point closest to q that is inside the OBB?

11-108: **Closest Point in OBB**

- What is the point closest to q that is inside the OBB?
  - Solve it the same way!
  - Push point in along axis of OBB
    - No longer axis-aligned, so there is a litte more math ...

11-109: **Closest Point in OBB**

- How do we determine if the point is in range for the x-axis?



11-110: **Closest Point in OBB**

- We know how to find the distance from a plane (given a point an a normal)
  - Normal = $-x$ basis (for min extent) and x basis (for max extent)
  - Point = center of OBB
  - Check to see if disntance is $> x$ extent

11-111: **Closest Point in OBB**

- Distance a from point $\mathbf{q}$ to center of box, along box's x axis:
  - Let X_B be the local x-axis of the box
  - Let $\mathbf{c}$ be the center of the box
  - Plane that goes through origin, perpendicular to X_B is described by: $\mathbf{p} \cdot \mathbf{X_B} = d$. How do we get d?

11-112: **Closest Point in OBB**

- Distance a from point $\mathbf{q}$ to center of box c, along box's x axis:
  - Let X_B be the local x-axis of the box
  - Let $\mathbf{c}$ be the center of the box
  - Plane that goes through origin, perpendicular to X_B is described by: $\mathbf{p} \cdot \mathbf{X_B} = d$. How do we get d?
    - Plug in a known point on the plane – center of the box c: $d = \mathbf{p} \cdot \mathbf{X_B}$.

11-113: **Closest Point in OBB**

- Distance a from point $\mathbf{q}$ to center of box c, along box's x axis:

$$
\begin{aligned}
a &= \mathbf{q} \cdot \mathbf{X_B} - d \\
a &= \mathbf{q} \cdot \mathbf{X_B} - \mathbf{c} \cdot \mathbf{X_B}
\end{aligned}
$$

- If $|a| < x$ extent, in range
- else if a is negative, add $\mathbf{X_B} * (|a| - x$ extent) from q
- else if a is positive, subtract $\mathbf{X_B} * (|a| - x$ extent) from q

11-114: **Intersection of two Spheres**

- Given two spheres $\mathbf{c_1}, r_1$ and $\mathbf{c_2}, r_2$, how can we determine if they interserct?

11-115: **Intersection of two Spheres**

- Given two spheres $\mathbf{c_1}, r_1$ and $\mathbf{c_2}, r_2$, how can we determine if they interserct?
  - Distance between centers is $<$ sum of the radii, intersect
  - $||\mathbf{c_1} - \mathbf{c_2}|| < r_1 + r_2$
  - $||\mathbf{c_1} - \mathbf{c_2}||^2 < (r_1 + r_2)^2$

11-116: **Sphere and AABB**

- Given a sphere $\mathbf{c}, r$ and an AABB (min and max points), how do we determine if the sphere intersects the AABB?

11-117: **Sphere and AABB**

- Given a sphere $\mathbf{c}, r$ and an AABB (min and max points), how do we determine if the sphere intersects the AABB?
  - Find the point $\mathbf{p}$ in the AABB closest to the center of the sphere $\mathbf{c}$
  - Compute the distance from $\mathbf{c}$ to $\mathbf{p}$, check to see if it is less than r
  - (actually, likely to use the squared distance, to save on a square root)

11-118: **Sphere and Plane**

- Given a sphere $\mathbf{c}, r$ and a plane $\mathbf{n}d$, do the sphere and plane intersect?

11-119: **Sphere and Plane**

- Given a sphere $\mathbf{c}, r$ and a plane $\mathbf{n}d$, do the sphere and plane intersect?
    - Compute the point on the plane $\mathbf{p}$ closest to the center of the circle
    - Compute the distance from $\mathbf{p}$ to $\mathbf{c}$, check to see if it is less than r
    - (once again, really use squared distance)

11-120: **AABB and Plane**

- Given
    - Plane ($\mathbf{n}$ and d)
    - AABB (min and max points)
- Does the plane intersect with the AABB?

11-121: **AABB and Plane**

- Given a plane $\mathbf{n}$ and d, we can determine which side of the plane a point p is on:
    - $\mathbf{n} \cdot \mathbf{p} > d$, point is on normal side of plane
    - $\mathbf{n} \cdot \mathbf{p} < d$, point is opposide normal side of plane
- Test all 8 points, see if they are on the same side of the plane (8 dot products)
- 24 multiplications
    - Can we do less?

11-122: **AABB and Plane**

- Given a plane $\mathbf{n}$ and d, we can determine which side of the plane a point p is on:
    - $\mathbf{n} \cdot \mathbf{p} > d$, point is on normal side of plane
    - $\mathbf{n} \cdot \mathbf{p} < d$, point is opposide normal side of plane

```
x1 = (aabb.minX * n[0]);
x2 = (aabb.maxX * n[0]);
y1 = (aabb.minY * n[1]);
y2 = (aabb.maxY * n[1]);
z1 = (aabb.minZ * n[2]);
z2 = (aabb.maxZ * n[2]);
if ((min(x1,x2) + min(y1,y2) + min(z1,z2) < d) &&
    (max(x1,x2) + max(y1,y2) + max(z1,x2) > d))
  intersection!
```

11-123: **Dynmamic Intersection**

- Doing only static intersections will miss intersections that happen between frames
- Small, fast-moving objects (like bullets) can penetrate thin objects (like walls)
- We can model a small object like a bullet as a ray
    - $\mathbf{p}(t) = \mathbf{p}_0 + t\mathbf{d}$
    - $\mathbf{p}_0$ is the endpoint of the ray
    - $\mathbf{d}$ is the direction of the ray
    - t is "time" parameter

11-124: **Intersection of Ray and Plane**

- Given a plane $\mathbf{n}, d$ and a ray $\mathbf{p}_0, \mathbf{d}$, at what point do the plane and ray intersect?
    - Both at what point in space, and at what time?

11-125: **Intersection of Ray and Plane**

- Given a plane $\mathbf{n}, d$ and a ray $\mathbf{p}_0, \mathbf{d}$, at what point do the plane and ray intersect?
    - When a point on the ray is also on the plane

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

11-126: **Intersection of two 3D rays**

- Given two rays:
    - $\mathbf{r}_1(t_1) = \mathbf{p_1} + t_1\mathbf{d_1}$
    - $\mathbf{r}_2(t_2) = \mathbf{p_2} + t_2\mathbf{d_2}$
- When do they intersect?

11-127: **Intersection of two 3D rays**

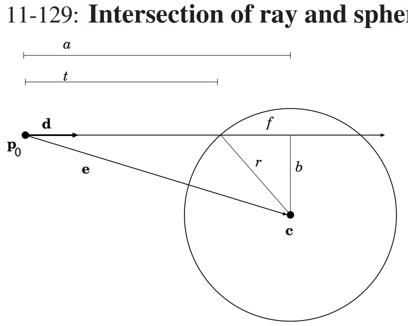
$$\begin{aligned}
\mathbf{r}_1(t_1) &= \mathbf{r}_2(t_2) \\
\mathbf{p}_1 + t_1\mathbf{d}_1 &= \mathbf{p}_2 + t_2\mathbf{d_2} \\
t_1\mathbf{d}_1 &= \mathbf{p}_2 + t_2\mathbf{d_2} - \mathbf{p}_1 \\
t_1\mathbf{d}_1 \times \mathbf{d}_2 &= (\mathbf{p}_2 + t_2\mathbf{d_2} - \mathbf{p}_1) \times \mathbf{d}_2 \\
t_1(\mathbf{d}_1 \times \mathbf{d}_2) &= t_2\mathbf{d_2} \times \mathbf{d}_2 + (\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{d}_2 \\
t_1(\mathbf{d}_1 \times \mathbf{d}_2) &= t_2 * 0 + (\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{d}_2 \\
t_1(\mathbf{d}_1 \times \mathbf{d}_2) &= (\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{d}_2 \\
t_1(\mathbf{d}_1 \times \mathbf{d}_2) \cdot (\mathbf{d}_1 \times \mathbf{d}_2) &= (\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{d}_2 \cdot (\mathbf{d}_1 \times \mathbf{d}_2) \\
t_1 &= \frac{(\mathbf{p}_2 - \mathbf{p}_1) \times \mathbf{d}_2 \cdot (\mathbf{d}_1 \times \mathbf{d}_2)}{||\mathbf{d}_1 \times \mathbf{d}_2||^2}
\end{aligned}$$

- if $||\mathbf{d}_1 \times \mathbf{d}_2||^2 == 0$, rays do not intersect

11-128: **Intersection of ray and sphere**

- Intersection of a ray a an a sphere is slightly more complicated

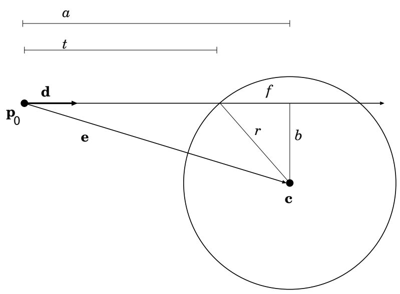
11-129: **Intersection of ray and sphere**



$$
\begin{aligned}
\mathbf{e} &= \mathbf{c} - \mathbf{p}_0 \\
a &= \mathbf{e} \cdot \mathbf{d}
\end{aligned}
$$

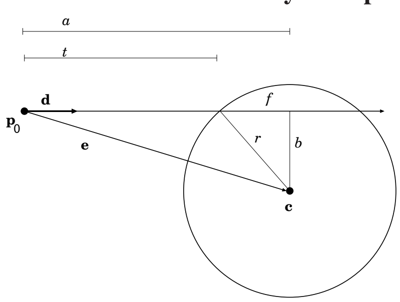
Need t, which we can get from f, which we can get from b ...

11-130: **Intersection of ray and sphere**



$$
\begin{aligned}
a^2 + b^2 &= ||\mathbf{e}||^2 \\
a^2 + b^2 &= \mathbf{e} \cdot \mathbf{e} \\
b^2 &= \mathbf{e} \cdot \mathbf{d} - a^2
\end{aligned}
$$

11-131: **Intersection of ray and sphere**



$$
\begin{aligned}
f^2 + b^2 &= r^2 \\
f^2 + (\mathbf{e} \cdot \mathbf{e} - a^2) &= r^2 \\
f^2 &= r^2 - (\mathbf{e} \cdot \mathbf{e}) + a^2 \\
f &= \sqrt{r^2 - (\mathbf{e} \cdot \mathbf{e}) + a^2}
\end{aligned}
$$

11-132: **Dynmamic: Plane & AABB**

- Assume that the plane is stationary, AABB is moving
  - Always change reference frame if plane is “really” moving, or if both are moving
- Plane $\mathbf{n}, d$,
- AABB $\mathbf{p}_{min}, \mathbf{p}_{max}$,

- displacment ray for AABB **r**

11-133: **Dynmamic: Plane & AABB**

- Plane $\mathbf{n}, d$,
- AABB $\mathbf{p}_{min}, \mathbf{p}_{max}$,
- displacment ray for AABB **r**
  - Find the corner of the AABB closest to the plane
  - That is the corner that will intersect
  - Ray/plane intersection problem