DEPARTMENT OF INFORMATION SYSTEMS AND COMPUTER SCIENCE

Bounding Boxes

Axis-aligned and Oriented

Lecture Time!

- Collision Requirements: Bounding Volumes
- Bounding Boxes: Axis-Aligned and Oriented

Circles Are Cool, But...

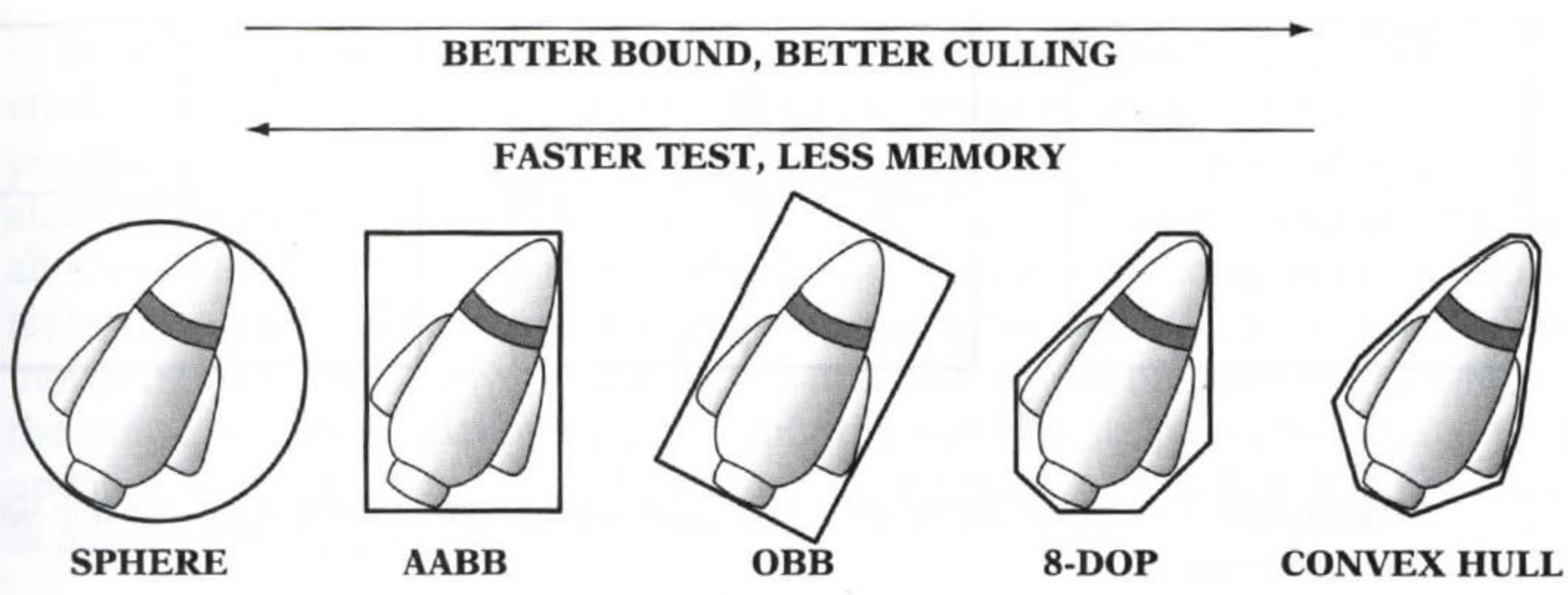
- What about other geometrical shapes?
- How do we check for collision?
  - What values do we need to store?
  - What are the conditions to check?
  - What if the two shapes being checked are different?
- What about collision responses?

Bounding Volumes

- Circle (sphere if 3D) → fastest test, lowest memory requirements
- AABB (axis-aligned bounding box)
- OBB (oriented bounding box)
- 8-DOP (eight-direction discrete orientation polytope)
- Convex hull → best bounding and culling

DISCS

Bounding Volumes



Bounding Volumes

- Can use bounding volume tests for several purposes:
  - For an "early out" or to allow collision test to exit early without needing to run the expensive/super-accurate geometric tests
  - Queries such as point inclusion, ray intersection with the volume, and intersection with planes and polygons

Bounding Volumes

- Bounding volumes are typically computed in a preprocessing step instead of at runtime
- However, some BVs must be realigned at runtime when their contained objects move
  - Realigning the BV is usually cheaper than recomputing it from scratch

Axis-Aligned Bounding Box

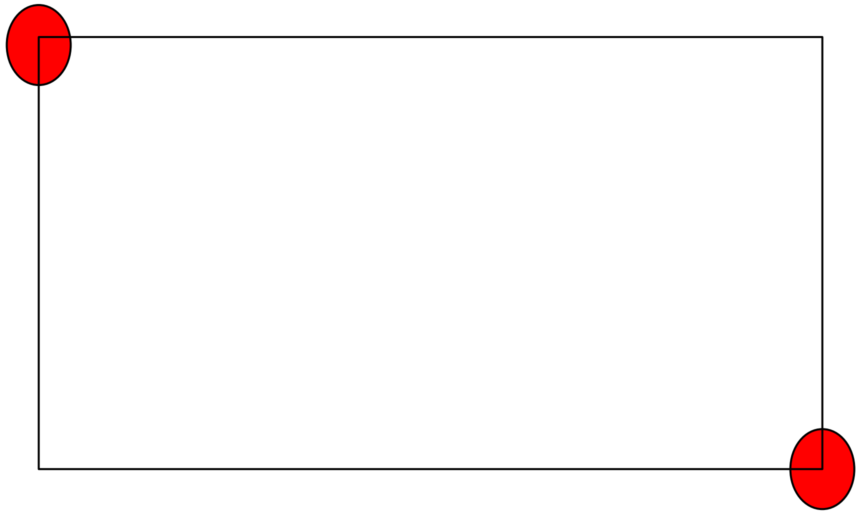
- Rectangular box with faces oriented such that its face normals are at all times parallel with the axes of a given coordinate system
  - Fast overlap check
  - Low storage requirements

Axis-Aligned Bounding Box

- Three common representations of AABB:
  - Min-max
  - Min-widths
  - Center-radius

Min-Max Representation

- For min-max, simply get the minimum and maximum coordinate values along each axis
  - In other words, the two opposing corner points



Min-Max Overlap

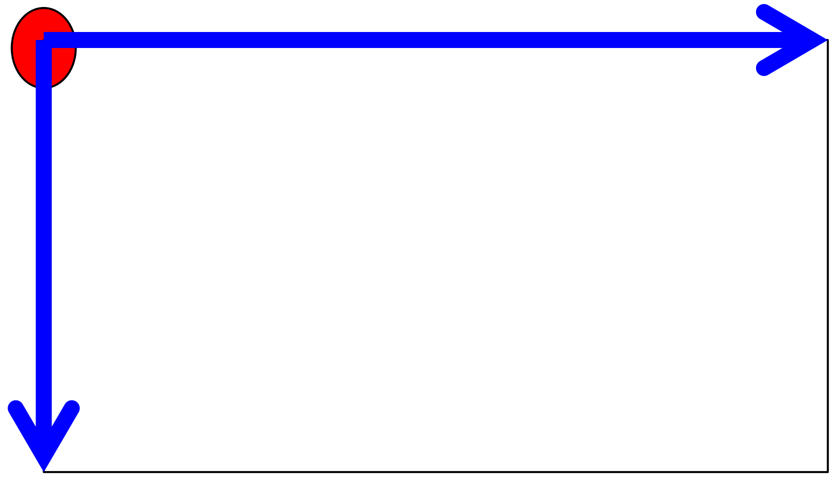
- Overlap test involves checking each coordinate axis for two AABB's a and b
  - If, for any axis, the minimum coordinate value of one is greater than the maximum coordinate of the other, then there is no intersection
  - Overlapping on all axes means there is an intersection

Min-Max Exercise

- Check each pair of AABBs to see if they intersect:
  - Min: (0, 0) ; Max: (8, 8)
  - Min: (4, -1) ; Max: (5, 2)
  - Min: (-3, 3) ; Max: (1, 5)

Min-Width Representation

- For min-width, get the minimum corner point and the dimensions of the box



Min-Width Overlap

- Overlap test involves checking each coordinate axis for two AABB's a and b
- For each axis, solve for difference of minimum coordinate values (the next slide assumes computation is `a.min[axis] - b.min[axis]`)

Min-Width Overlap

- If the difference is greater than the corresponding dimension of b, then there is no intersection
- If the negative of the difference is greater than the corresponding dimension of a, then there is no intersection
- Computed for all axes but no early out? Then there is an intersection

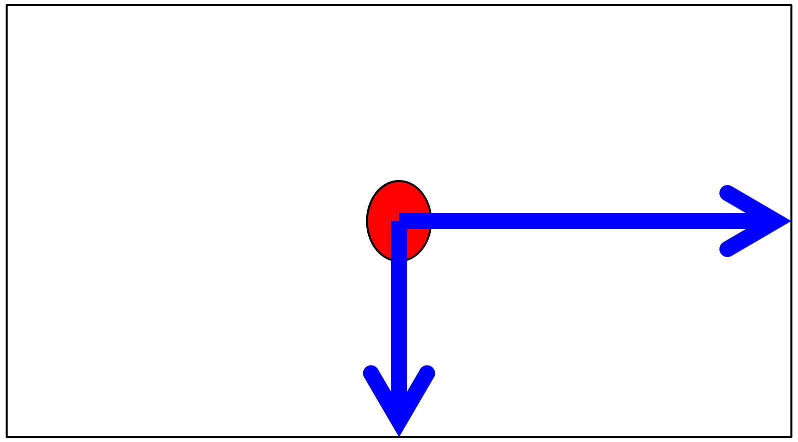
Min-Width Exercise

- Check each pair of AABBs to see if they intersect:
  - Min: (0, 0) ; Dimensions: (8, 8)
  - Min: (4, -1) ; Dimensions: (1, 3)
  - Min: (-3, 3) ; Dimensions: (4, 2)

DISCS

Center-Radius Representation

- For center-radius, get the center point and half the dimensions of the box



Center-Radius Overlap

- Overlap test involves checking each coordinate axis for two AABB's a and b
- For each axis, solve for absolute value of difference of center coordinate values (the next slide assumes computation is `abs( a.c[axis] - b.c[axis] )`)

Center-Radius Overlap

- If this value is greater than the sum of the corresponding half-dimensions of the boxes, then there is no intersection
- Computed for all axes but no early out? Then there is an intersection

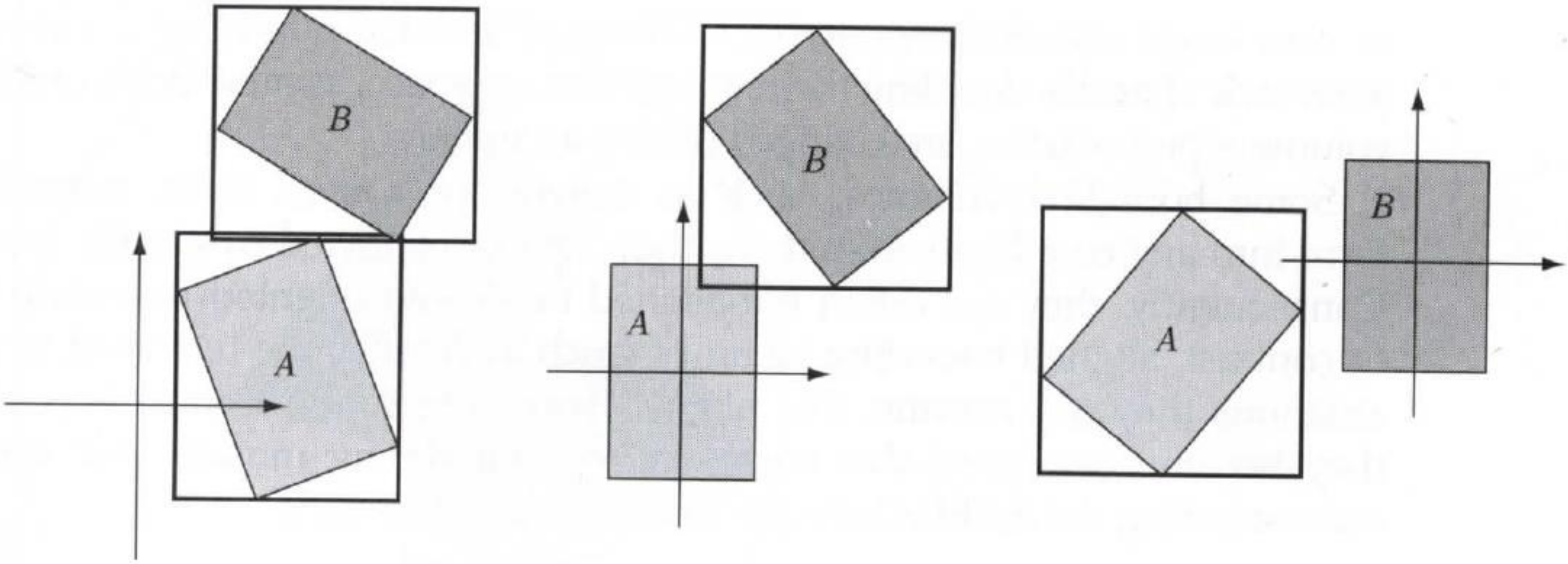
Center-Radius Exercise

- Check each pair of AABBs to see if they intersect:
  - Center: (4, 4) ; Radii: (4, 4)
  - Center: (4.5, 0.5) ; Radii: (0.5, 1.5)
  - Center: (-1, 4) ; Radii: (2, 1)

Computing AABBs

- Both BVs must use the same coordinate system
  - Transform both BVs into world space (better if high number of overlap checks)
  - Transform one BV into the local space of the other (half the work and usually a tighter BV result)

Computing AABBs



Computing AABBs

- If the containing object rotates, the AABB must be recomputed
- Here are two of several strategies that can be used:
  - Computing a tight dynamic reconstruction from the original point set
  - Computing an approximate dynamic reconstruction from the rotated AABB

From Original Point Set

- Simply create another AABB that contains the object in its current orientation
  - Find minimum and maximum coordinate values along all axes
  - Will make more sense when we get to polygons

From Rotated AABB

- Same as above, but use the original AABB with the given rotation
  - The original AABB must be used, otherwise this strategy will result in an object's AABB growing indefinitely

Recitation

- Assuming no rotation applied yet:
  - What is the AABB of a square?
  - What is the AABB of a rectangle?

Recitation

- What is the AABB of a circle?
- What happens to the AABB when the circle rotates?
  - Do you have to recompute the AABB?

Exercises

- Create a program that renders 2 rectangles that rotate at different speeds
- In addition, render each rectangle's AABB
- Allow input to move the two rectangles and change their colors to indicate whether or not their AABBs are overlapping

Exercises

- Add a circle and render its AABB
- Change the color of any shape when its AABB intersects another shape's AABB

Oriented Bounding Box

- An OBB is like an AABB but has an arbitrary orientation
- Testing for overlap between two OBBs does NOT require both of them to use the same coordinate system
  - That would be just AABB all over again

Computing OBBs

- OBB representation needs:
  - OBB's center point
  - Local axes
  - Halfwidth extents of OBB along each axis
    - In other words, half the dimensions of the OBB

Computing OBBs

- Can construct an AABB for each OBB and use AABB intersection test
  - Not really much of a point except as a cheap early out
- Can use the separating axis test
  - To be covered when we get to polygons
  - Up next: Polygons

Homework

- Create a program that displays 5 RectangleShapes that have the same color but are of different dimensions and not initially overlapping
- Then, give each of these shapes a different speed of rotation
- Render each shape's AABB as it rotates
- Shapes with AABBs that are intersecting should be displayed with a different color

DISCS