Data Structures for Computer Graphics

Incidence Operations used in Computer Graphics

Lectured by Vlastimil Havran

Incidence Operation Usage

- Testing the data against query, typically in leaves of a hierarchical data structure
- Query is represented by some geometric entity (point, shape, or another hierarchy)
- Incidence operation classification:
  - Intersection *detection* – Boolean result (yes/no)
  - Distance *computation* – distance between query and data (or penetration depth)
  - Intersection *computation* – intersection between the query and the data (a point on the ray intersecting triangle)
  - Other constructive queries – computes also some other results (example: closest points between two lines)

Common Incidence Examples

- Does sphere (query) contain a point (data) ?
- Does the sphere (query) intersect a sphere (data) ?
- Does the sphere (query) completely contain a sphere (data)?
- Does a query ray intersect a sphere/triangle (data) ? (and where?)
- Does a box intersect another box (collision detection) ?

Distance Incidence

- What is a distance between (query) point and data point ?
- What is a minimum/maximum distance between query point and a box ?
- What is a distance between query point and some more complex shape (cone, cylinder, NURBS) ? For closed forms:
  - Positive distance : outside the object
  - Negative distance: inside the object
  - Zero distance: on the object's surface

Multiple Incidence

- Compute the distance between the data and many queries at once
- Could be programmed by a loop
- Specialized algorithms are usually more efficient (branch prediction, spatial and data locality in cache, use of SSE instructions, preprocess queries)
- Examples:
  - compute for N rays the intersection with a single triangle
  - compute for a ray the intersection with 4 triangles (implementation by SSE instructions)

Types of Primitives used in Incidence

- **1D (line based):** 2D line, 2D line segment, 2D ray, 3D line, 3D line segment, 3D ray.
- **2D planar (plane based):** plane, triangle, convex polygon, general polygon, polygon with holes.
- **Quadrics:** sphere, (capped) cylinder, (capped) cone, ellipsoid.
- **3D shapes:** axis-aligned bounding box (AAAB), oriented bounding box (OBB), discrete orientation polytope (k-DOP), viewing frustum, convex polyhedron, non-convex polyhedron.
- Hundreds of of mutual combinations!

Dynamic Incidences on Objects

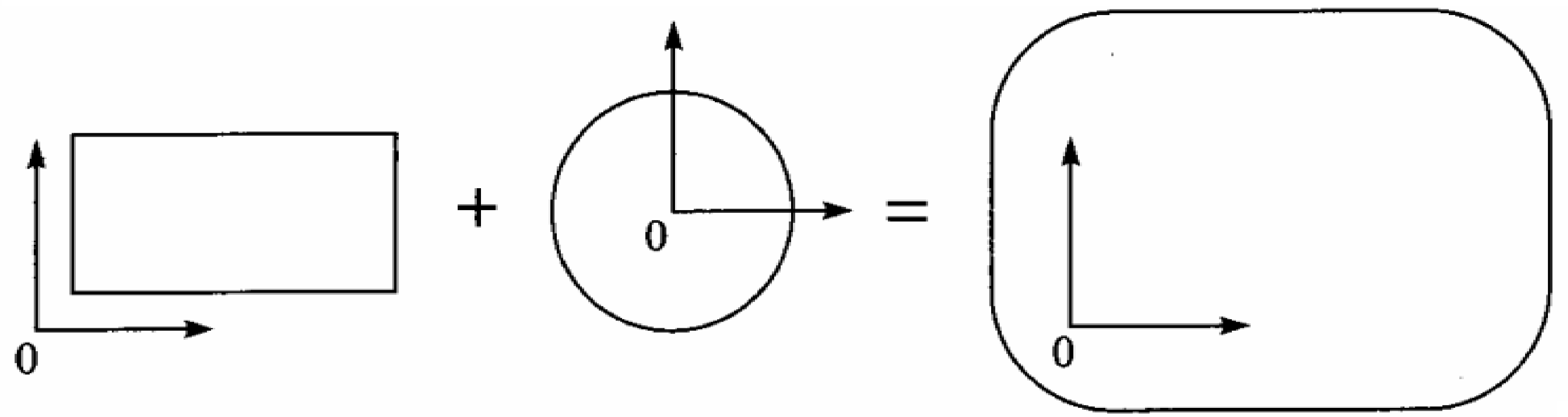
- Typically used in collision detection
- Either only one or both objects are moving
- Typical examples
  - Ray/moving sphere
  - Ray/moving triangle
  - Ray/moving AAAB or OBB
  - Plane/moving sphere or AAAB
  - Moving sphere/sphere or triangle
  - Moving triangle/moving triangle

  Note: AABB – axis aligned bounding box

  OBB – oriented bounding box

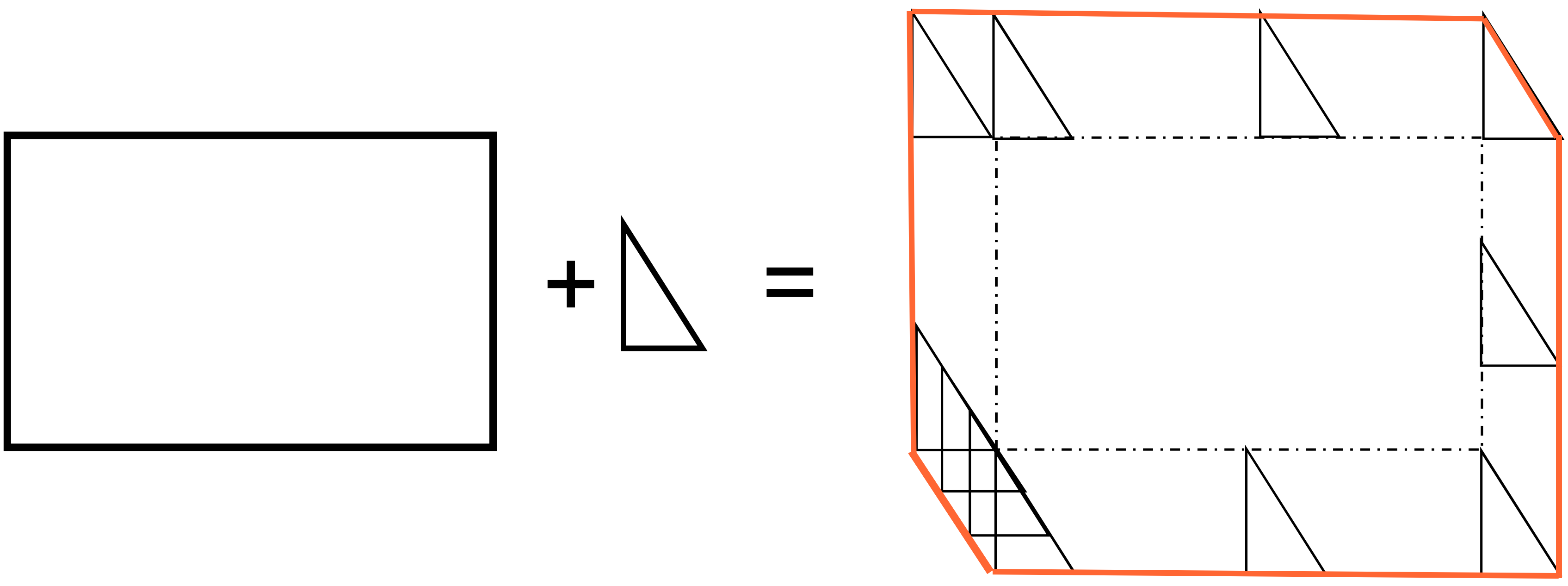
Non-Oriented Minkowski Sum

- Uses sphere as adding primitive
- Formally A+B = {x+y: x in A, y in B}
  - x and y vectors from A and B, resp.
- Corresponds to a convolution
- Note: Minkowski sum of convex objects is convex



Oriented Minkowski Sum

- Uses any primitive to add original shape



Use of Minkowski Sum and Difference

- In collision detection
- Instead of computing collision detection with original shape, we compute collision detection with the object(s) extended by Minkowski sum
- Use simpler query primitive for collision (point, ray)
- Directly only for precise computation, it could be expensive
- Indirectly the principles can be used in the design of incidence algorithms

More about 2D Minkowski Sum

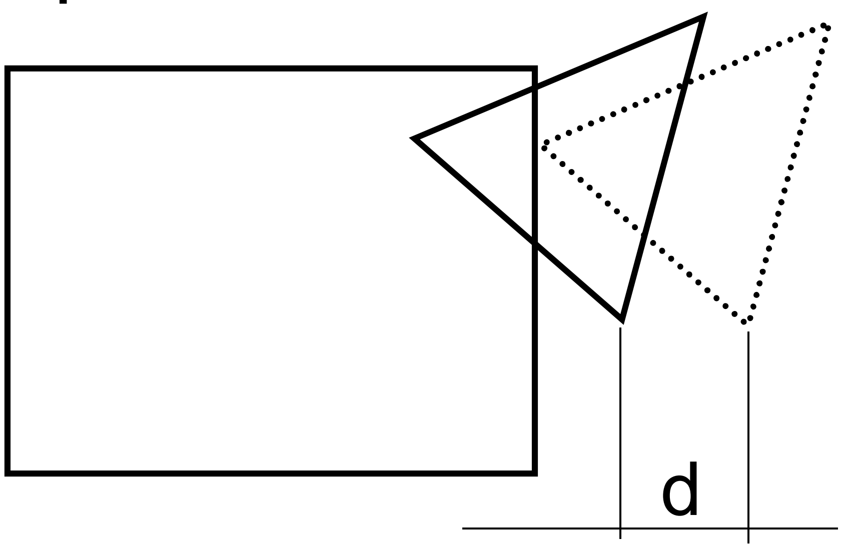
- http://cs.gmu.edu/~jmlien/research/mksum/
- http://en.wikipedia.org/wiki/Minkowski_addition
- http://cs.stonybrook.edu/~algorith/files/minkowski-sum.shtml
- http://www.geometrylab.de/minkowski/index.html.en
- http://www.cgal.org/Manual/3.4/doc_html/cgal_manual/Minkowski_sum_2/Chapter_main.html

Computation Precision

- **Exact**
  - gives correct results (up to number representation precision)
- **Conservative**
  - if says no, then no; if says yes, then maybe
- **Aggressive**
  - if says yes, then yes; if says no, then maybe
- **Approximate**
  - No answer is sure
  - Usually relative maximum error with respect to the size of object or absolute maximum error
  - It can be much faster than exact computation

Penetration Depth: for Collision Detection

- Definition: If two objects intersect, then the penetration depth is the **minimum distance** you have **to move one object** that the objects do not intersect
- Example:

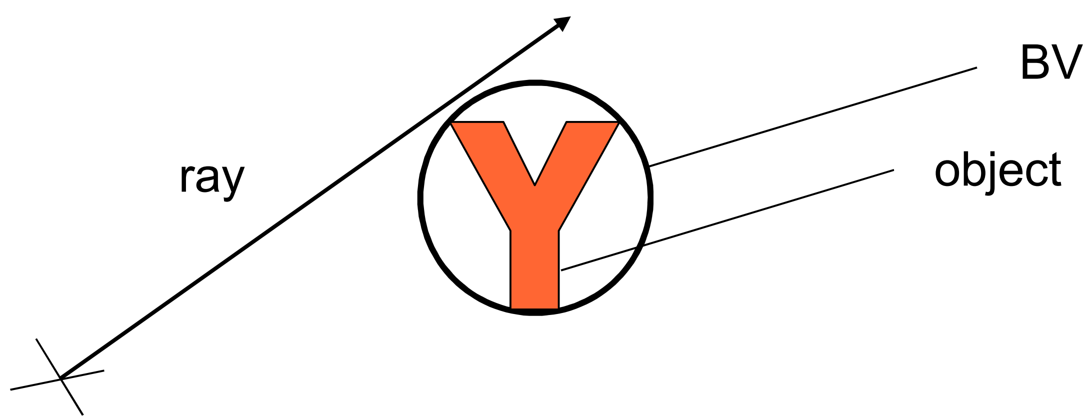


Bounding Volumes (BV)

- Bounding volume encloses the objects or a set of objects
- Desirable properties:
  – Inexpensive intersection test (between two BVs)
  – Tight fitting of BV to the object
  – Inexpensive computation of BV given an object
  – Easy to rotate and transform
  – Small use of memory

BV Usage in Applications

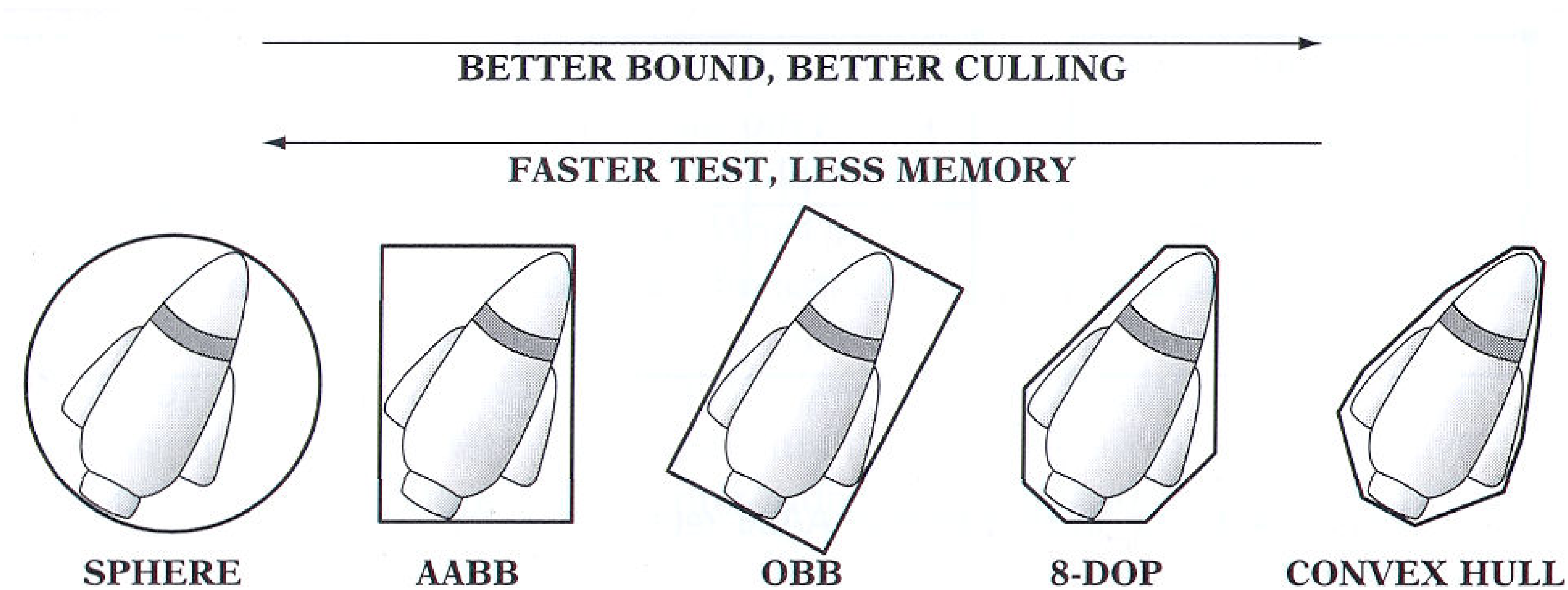
- **Collision detection** in conservative test: if two BVs do not intersect, then neither objects intersect.
- **Ray tracing** with BV is also conservative: if a ray does not intersect BV of an object, then ray cannot intersect an object



BV Types

- Sphere
- Axis-Aligned Bounding Box (AABB)
- Oriented Bounding Box (OBB)
- Discrete Orientation Polytope (k-DOP)
- Convex-Hull

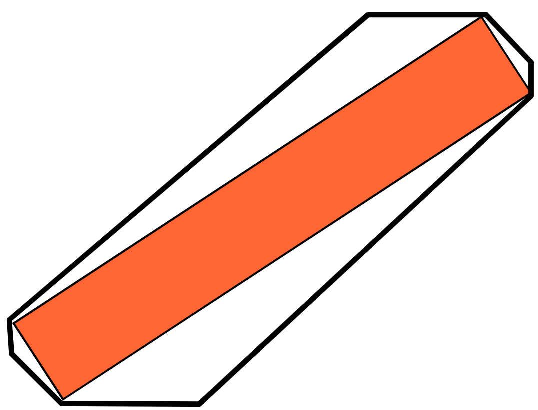
BV Types and Properties



K-Dops

- k=6: AABB (6-DOP is exactly AABB)
  - directions (1,0,0), (0,1,0), (0,0,1)
- k=14: chamfer vertices
  - add directions (1,1,1), (1,-1,1), (-1,1,1), (1,1,-1)
- k=18: chamfer edges
  - add directions (1,1,0), (1,-1,0), (1,0,1), (1,0,-1), (0,1,1), (0,1,-1)
- k=26: chamfer vertices and edges

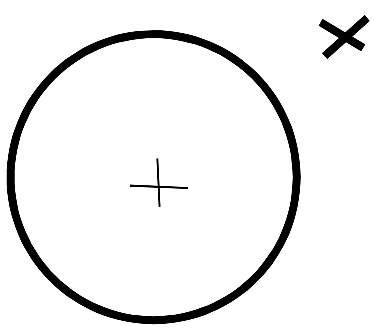
Example of 8-DOP in 2D:



Example 1: Sphere and Point

- Trivial to compute for
  - Point (Px,Py,Pz)
  - Sphere center (Sx,Sy,Sz) and radius R
- Point inside a sphere:

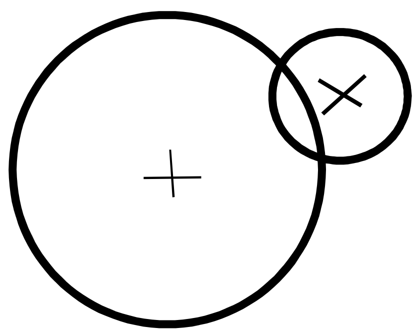
$$(Px\text{-}Sx)^2 + (Py\text{-}Sy)^2 + (Pz\text{-}Sz)^2 < R^2$$



Example 2: Sphere versus Sphere

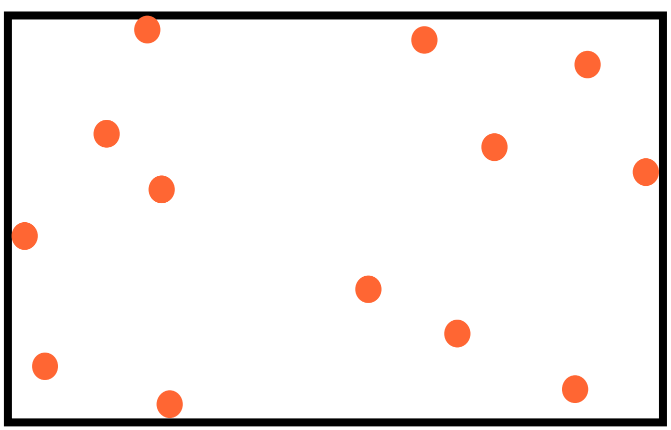
- Also trivial to compute for
  - Sphere center (Px,Py,Pz) and radius R2
  - Sphere center (Sx,Sy,Sz) and radius R1
- Point inside a sphere:

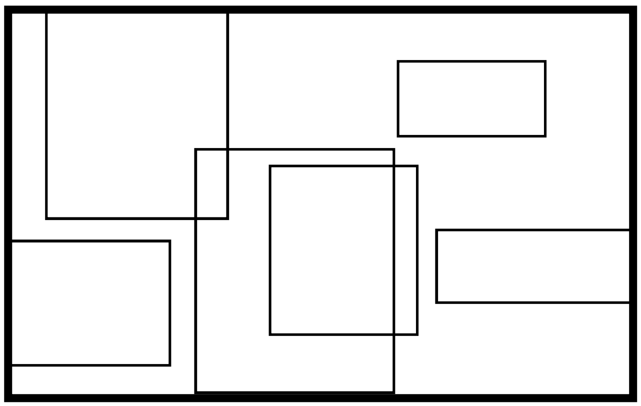
$$(Px-Sx)^2 + (Py-Sy)^2 + (Pz-Sz)^2 < (R1+R2)^2$$



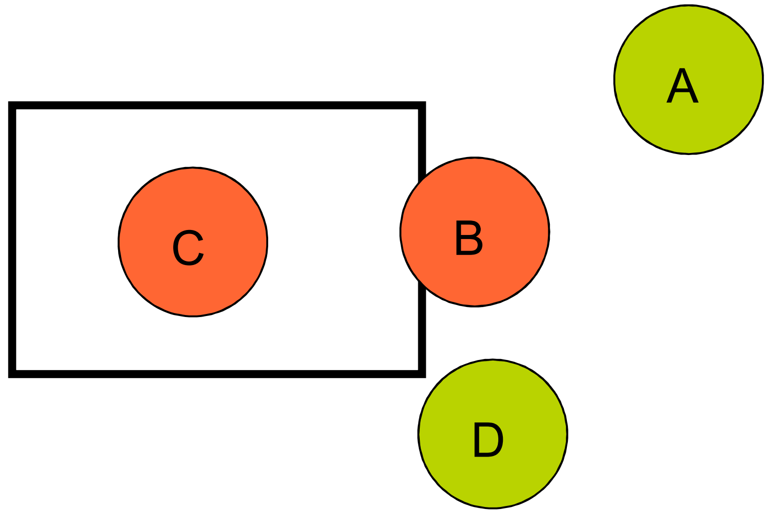
Example 3: AABB over set of points

- For each coordinate x, y, z
  AABB min(x) = min(x) for all points
- The same for AABB over set of AABBs





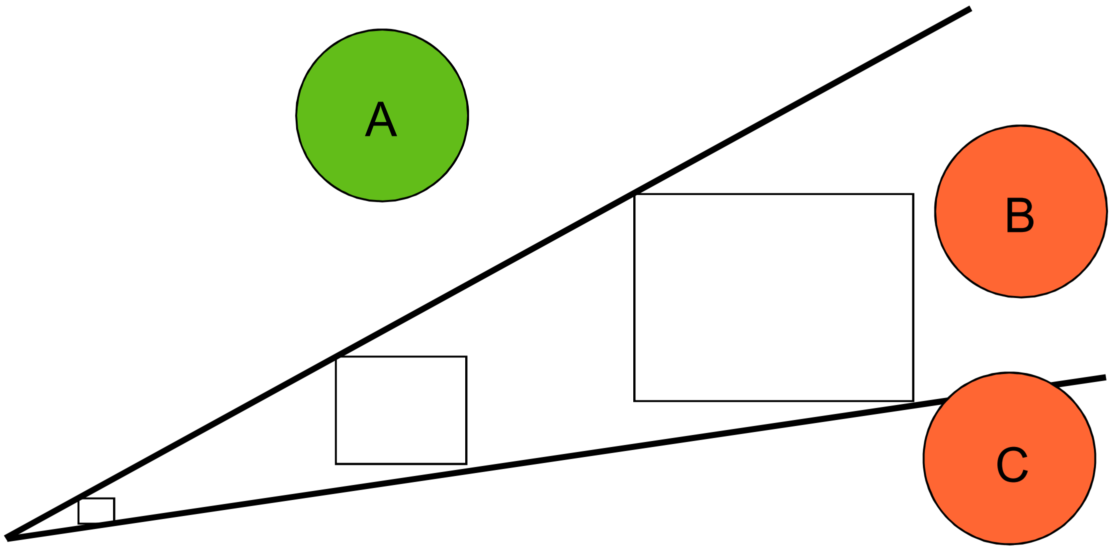
Sphere - AABB intersection test



- Easy cases: A and C
- More difficult cases: B and D
- What about 3D sphere/AABB ?

Viewing Frustum - Sphere

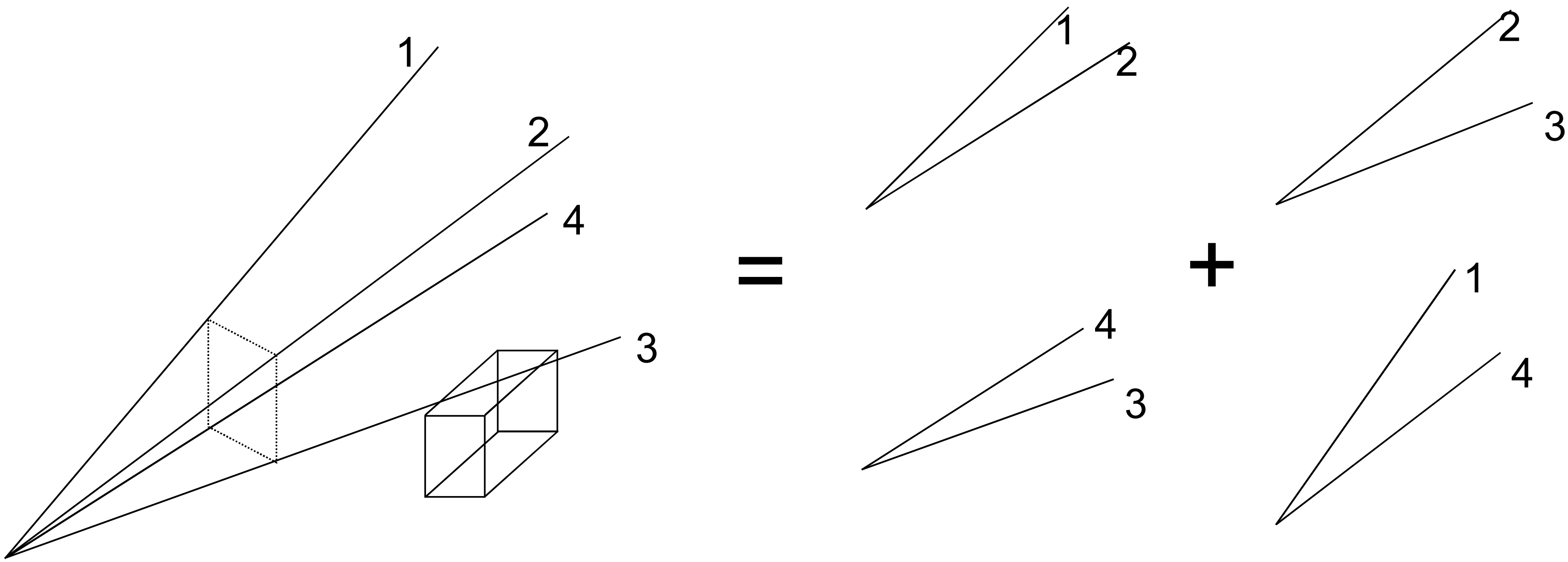
- Viewing frustum can be seen as expanding AABB along some direction:



- Note that for 2D viewing frustum is easy to solve, for 3D viewing frustum more difficult !

View Frustum - AABB

- View frustum can be arbitrarily oriented
- The viewing frustum can be decomposed into four triangles:

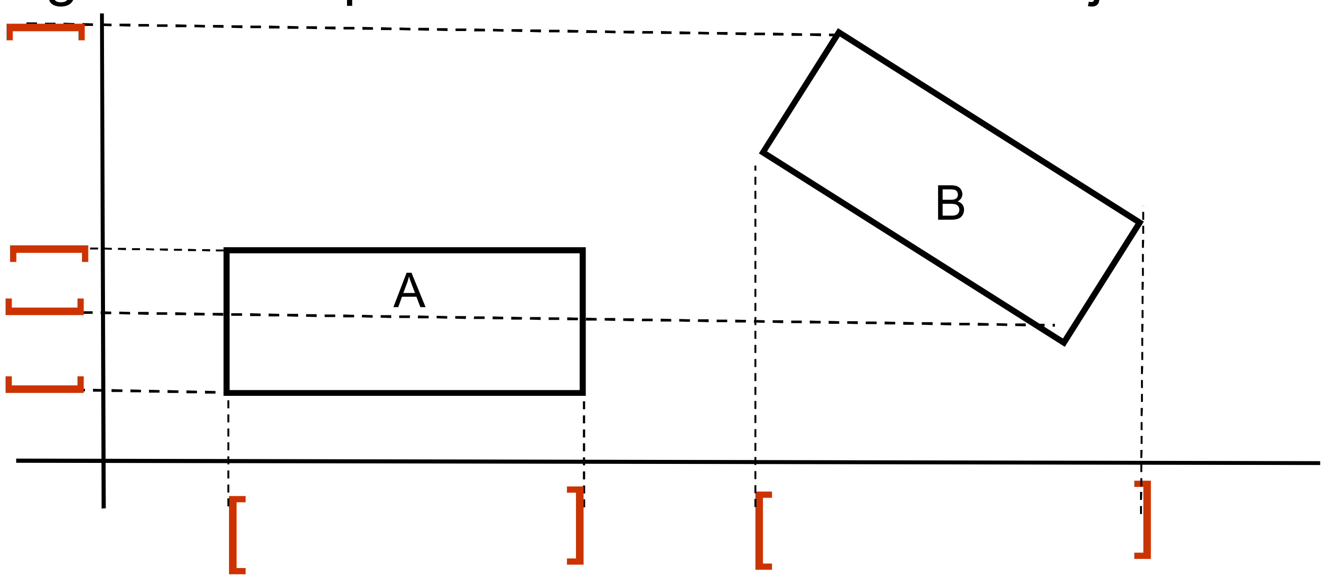


Method: Decomposition

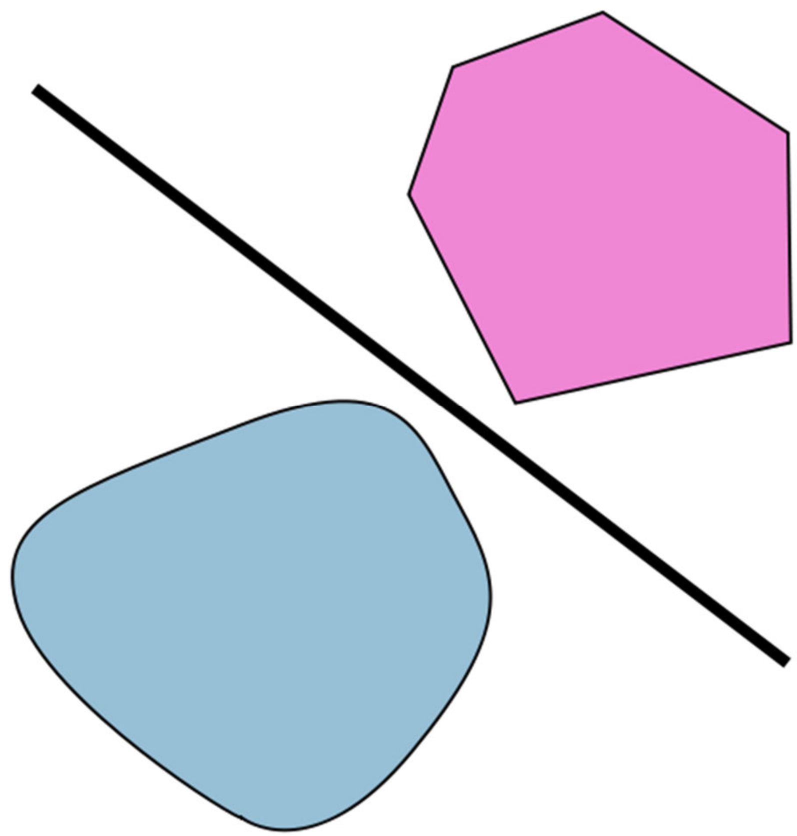
- Easy case: center of the box inside viewing frustum
- Otherwise viewing frustum is decomposed into simple primitives (triangles)
- Each triangle can be tested separately, if intersection is found for any triangle, then it exists
- Easier problem: rectangle against triangle

Generic Method: Separating Axis Theorem (SAT)

- SAT: two convex objects **A** and **B** are disjoint if for some vector ***V*** the projections ***V*.A** and ***V*.B** do not overlap.
  - SAT can be applied to all facets of both convex polytopes and to all planes given by two faces, the first face of object **A** and the second face of object **B**
  - ***V*** defines an axis, a plane perpendicular to axis has two halfspaces: positive halfspace contains the first object and negative halfspace contains the second object.



Separating Axis Theorem (SAT)



SAT for box *versus* box in 3D

- 3 axis for the first box normals
- 3 axis for the second box normals
- 3 x 3 = 9 axis for all vector products for normal of the first box and normal of the second box
- In total 15 tests

More in the paper: S. Gottschalk, M. Lin, and D. Manocha. "*OBBTree: A hierarchical Structure for rapid interference detection,*" Proc. Siggraph 96. ACM Press, 1996.

SAT for some combinations

Polytope	Polytope	Number of axes for testing
Line segment	Triangle	0+1+(1x3)=4
Line segment	Box	0+3+(1x3)=6
Triangle	Triangle	1+1+(3x3)=11
Triangle	Box	1+3+(3x3)=13
Box	Box	3+3+(3x3)=15

Bibliographic Resources

- 3D Games: Real-time Rendering and Software Technology, Watt and Policarpo, Addison Wesley, 2001
- Game Programming Gems, DeLoura, Charles River Media, 2000
- Geometric Tools for Computer Graphics, Schneider and Eberly, MKP, 2002
- The Graphics Gems Series (books)
- Introduction to Ray Tracing, ed. Glassner, Academic Press, 1989
- Journal of Graphics Tools (on web)
- Geometric Tools repository by Dave Eberly
- Real Time Collision Detection, Ericson, MKP, 2004
- Collision Detection in Interactive 3D environments, van den Bergen, MKP 2004
- Real Time Rendering, 3rd edition, Tomas Akenine-Moeller and Eric Haines, A.K. Peters Ltd. 2008
- Simple Geometric Library by Steve Baker's
- Talina Gaming System Collision by Andrew Aye

Literature - Survey

- 3D Object Intersection page

  http://www.realtimerendering.com/intersections.html

  ray, plane, sphere, cylinder, cone, triangle, AABB, OBB, frustum, polyhedron

  times

  ray, plane, sphere, cylinder, cone, triangle, AABB, OBB, frustum, polyhedron

Thank you for your attention!

Minkowski Sum Proof

- w1, w2 in A + B
- x1, x2 in A
- y1, y2 in B
- w1 = x1 + y1
- w2 = x2 + y2
- Any convex combination of w1 and w2 is a point in A+B
- Convex combination:

$$w = k1 * w1 + k2 * w2, \qquad k1 + k2 = 1,\ k1>0,\ k2>0$$

$w = k1.(x1 + y1) + k2.(x2 + y2) =$

$= k1.x1 + k2.x2 + k1.y1 + k2.y2 = x + y$

, where $x = k1.x1 + k2.x2$

$y = k1.y1 + k2.y2$

That is x … convex combination of x1 and x2

y … convex combination of y1 and y2

Since A and B are convex, that is for any x1 in A, x2 in A, y1 in B, y2 in B and also x in A, y in B also holds w=x+y in (A+B)

Some ISSUES to think about

1. Does have Minkowski sum symmetry, i.e., is A+B "equal to" B+A ?
2. What about specifying Minkowski difference as A-B similarly to A+B ?
3. Is Minkowski sum and difference reversible operation, i.e.:
   - Is (A+B) – B "equal to" A ?
   - Is (A-B) + B "equal to" A ?

Exercises

- Point-box min/max method
- Ray-box min/max method
- Ray-triangle
- Ray-sphere
- Advanced task: two AABB – do they cover completely another specified AABB ?