Physics for Game Programmers: Collision Detection

Gino van den Bergen
gino@dtecta.com

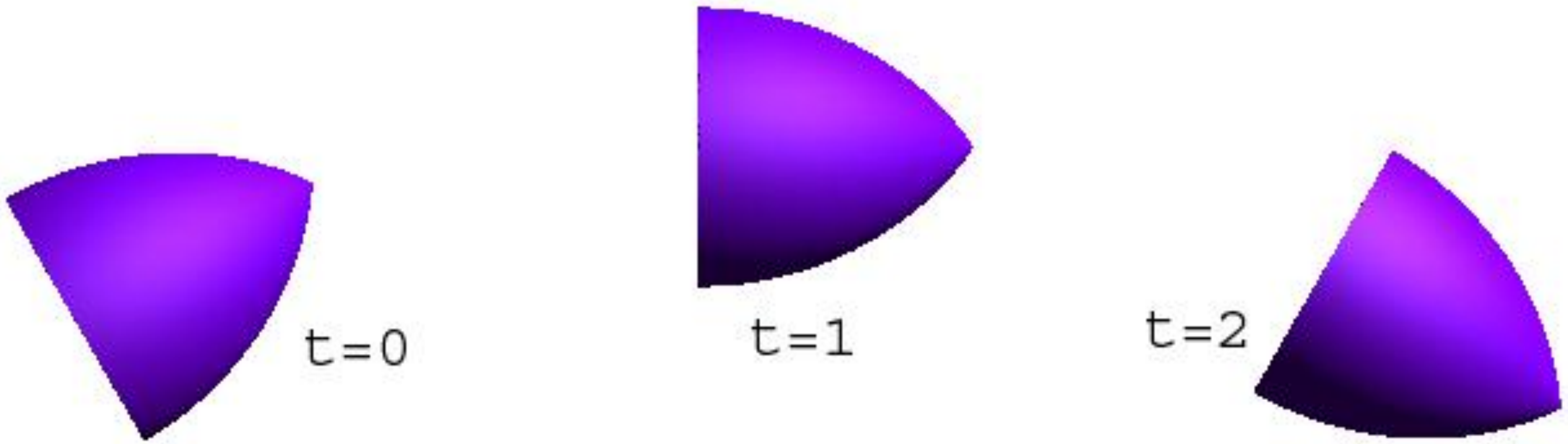
GDC 12

GAME DEVELOPERS CONFERENCE®
SAN FRANCISCO, CA
MARCH 5-9, 2012
EXPO DATES: MARCH 7-9
2012

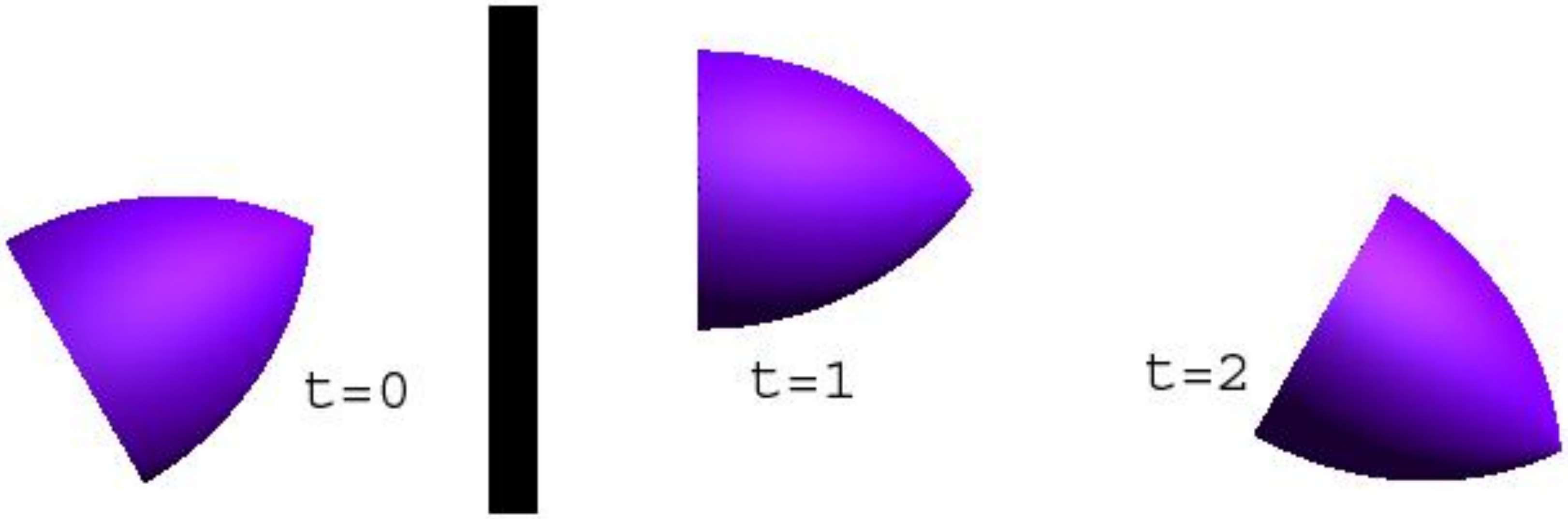
Collision Detection

- Find all pairs of objects that are colliding now, or will collide over the next frame.
- Compute data for response:
  - Contact normal
  - Contact point
  - Penetration depth

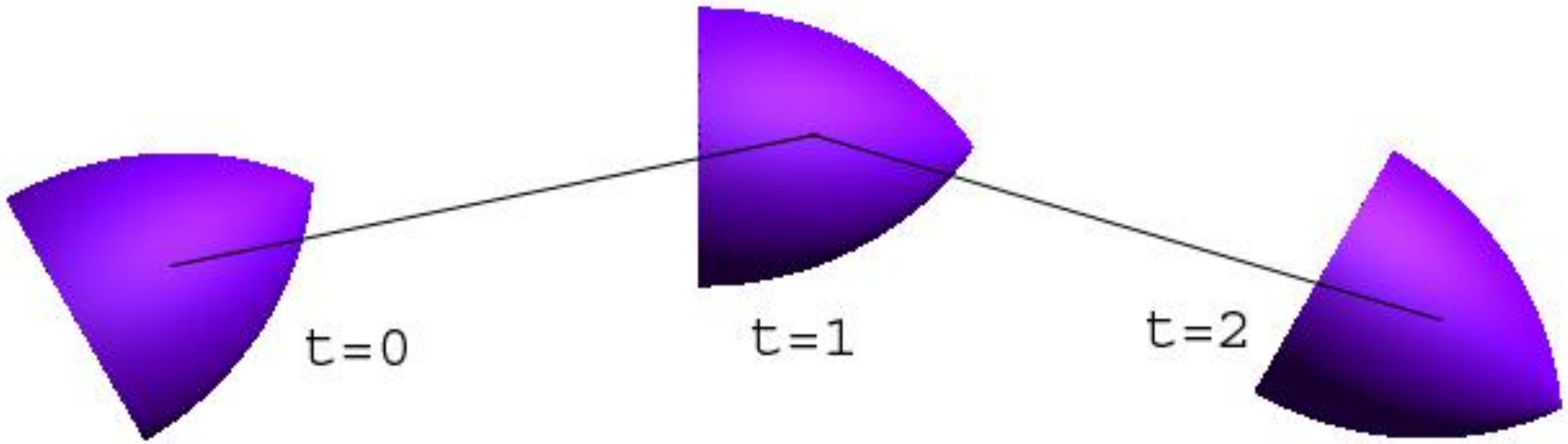
The Problem



The Problem



The Solution



Construct Plausible Trajectories

- Limited to trajectories involving piecewise constant linear velocities.
- Angular velocities are ignored. Rotations are considered instantaneous.

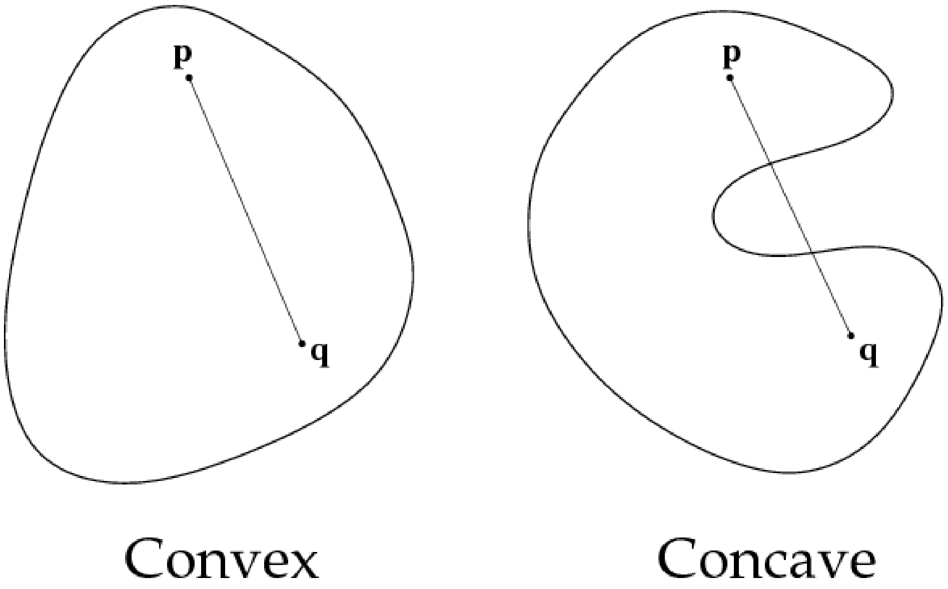
No Continuous Rotations?

- Solving continuous rotations is a lot trickier, so we dodge the issue.
- Tunneling may occur for rotating objects, but is less visible and often acceptable.
- Only doing continuous translations fixes our problems and is doable in real time.

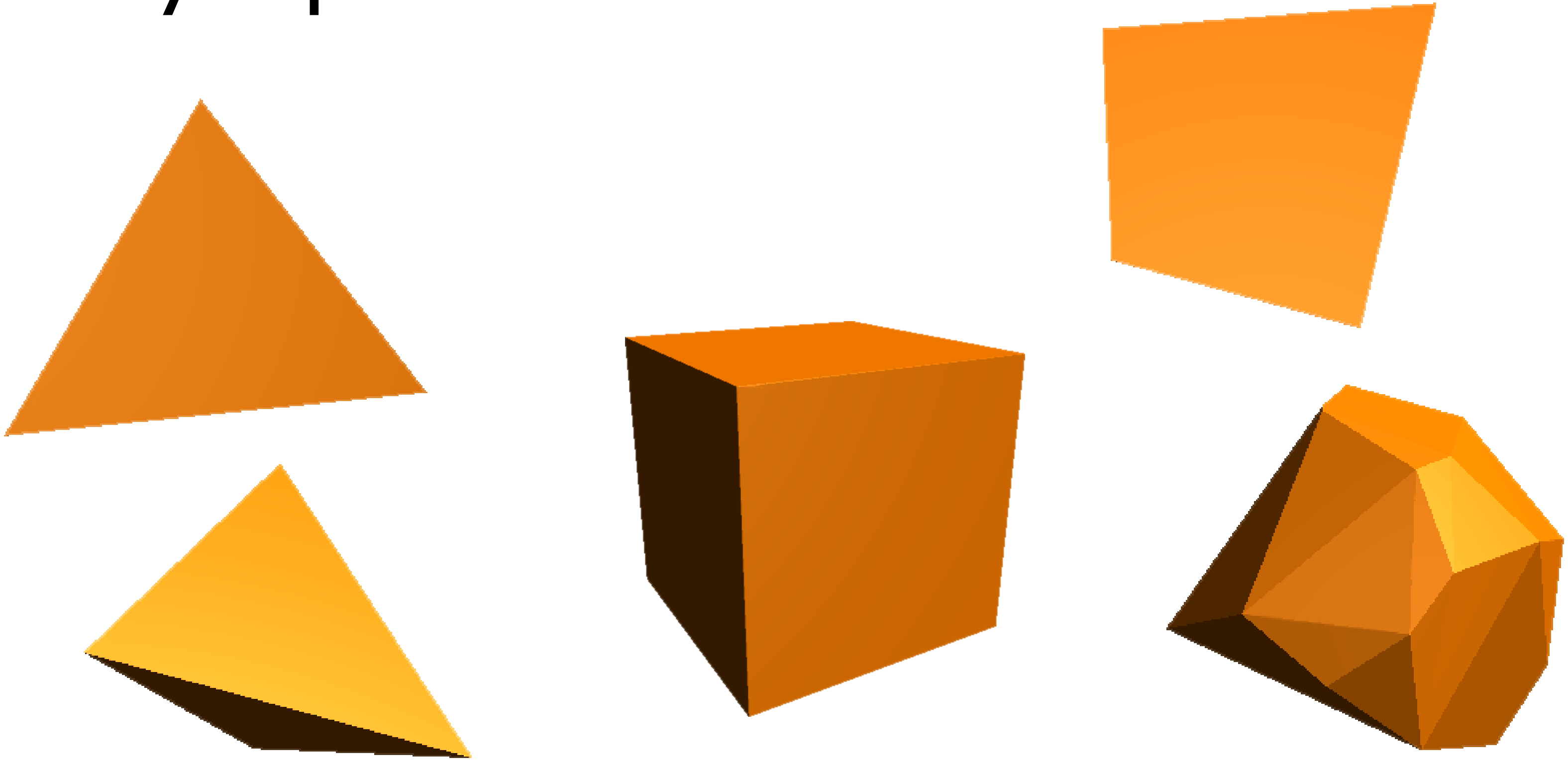
Collision Objects

- Static environment (buildings, terrain) is typically modeled using polygon meshes.
- Moving objects (player, NPCs, vehicles, projectiles) are typically convex shapes.
- We need to detect convex-convex and convex-mesh collisions.

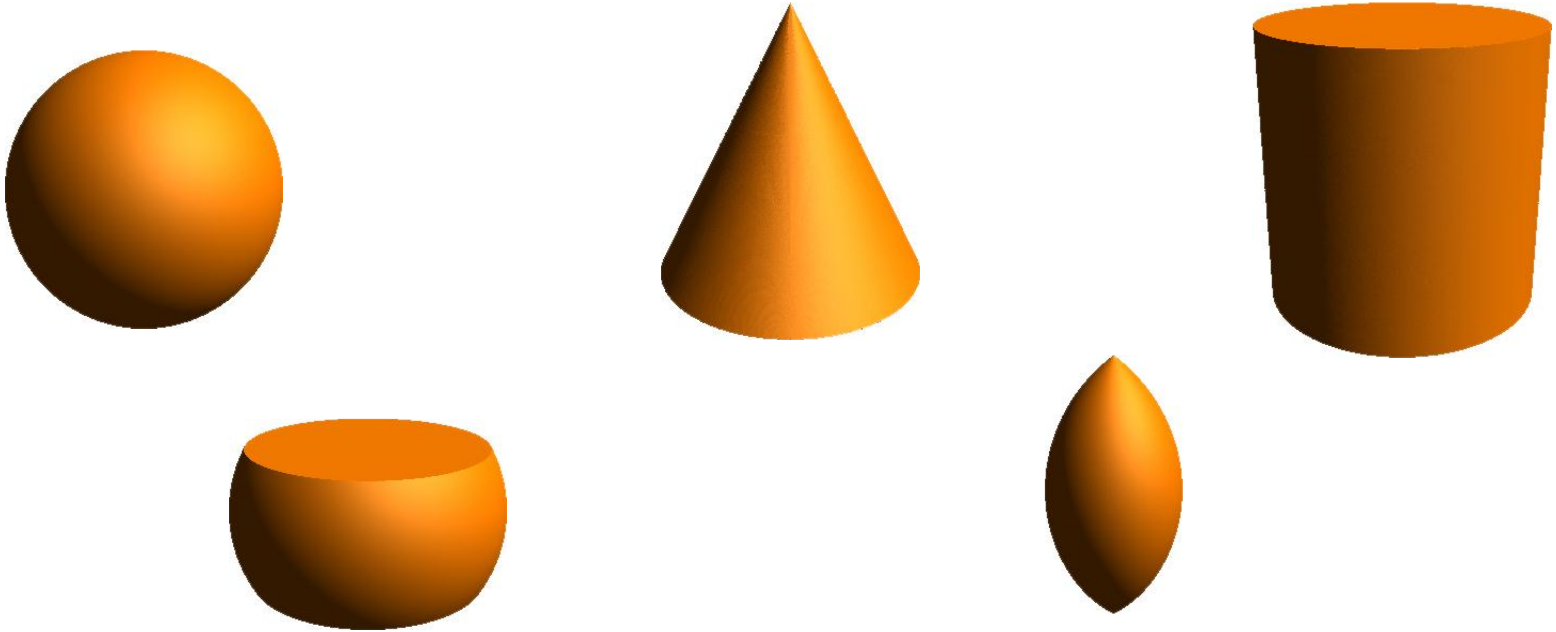
Convex Shapes



Polytopes



Quadric Shapes



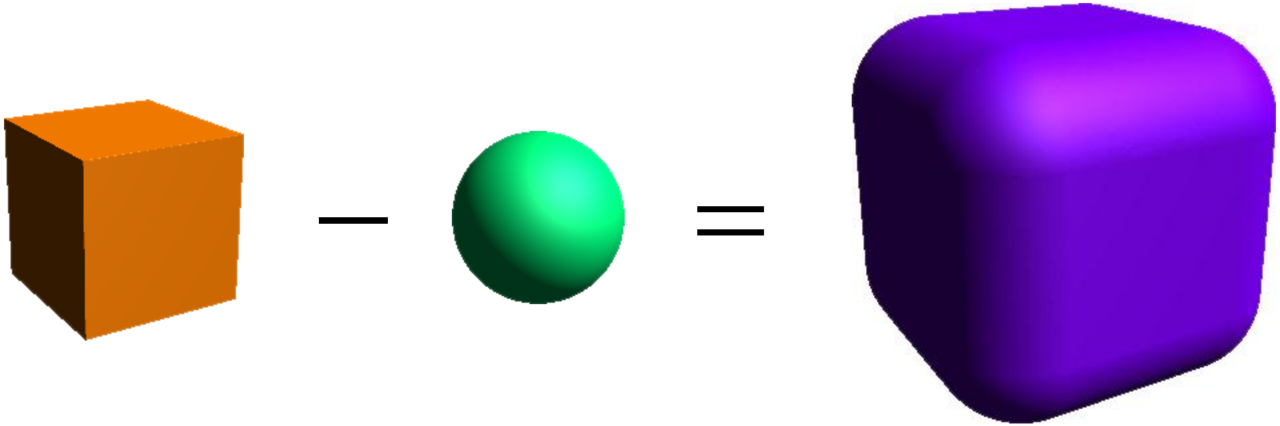
Configuration Space

- The *configuration space obstacle* (CSO) of objects A and B is the set of all vectors from a point of B to a point of A.

$$A - B = \{\mathbf{a} - \mathbf{b} : \mathbf{a} \in A, \mathbf{b} \in B\}$$

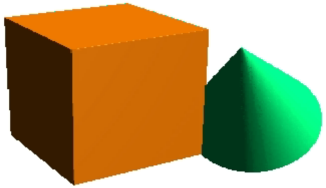
Configuration Space (cont'd)

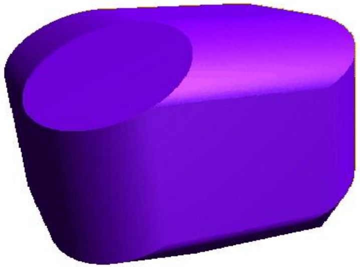
- CSO is basically one object dilated by the other:



Translation

- Translation of A and/or B results in a translation of $A - B$.





Rotation

- Rotation of A and/or B changes the shape of $A - B$.

Configuration Space?

- Collision queries on a pair of convexes are reduced to queries on the position of the origin with respect to the CSO.
- Point queries are easier than queries on pairs of shapes.

Queries: Distance

- The distance between two objects is the distance from the origin to the CSO.

$$d(A,B) = \min\left\{\left\|\mathbf{x}\right\| : \mathbf{x} \in A - B\right\}$$

Queries: Intersection Testing

- The objects intersect (have a common point) if the origin is contained by the CSO.

$$A \cap B \neq \emptyset \Leftrightarrow \mathbf{0} \in A - B$$

Queries: Penetration Depth

- The penetration-depth vector is the shortest translation that resolves a penetration, i.e., the point on the CSO's boundary closest to the origin.

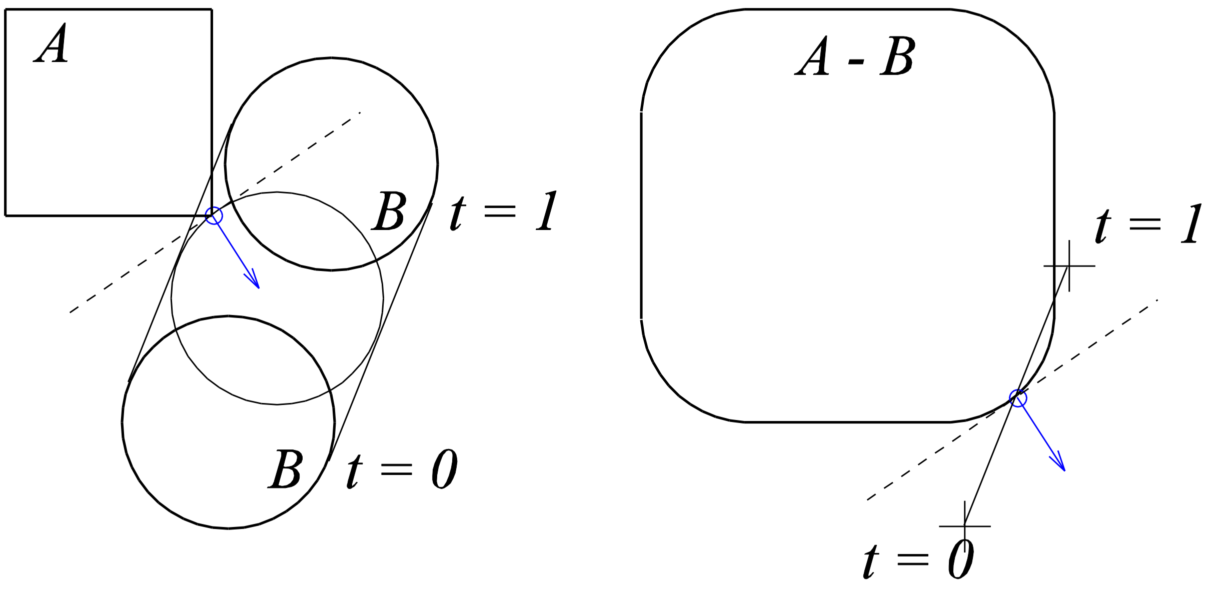
$$p(A,B) = \inf\left\{\|\mathbf{x}\| : \mathbf{x} \notin A - B\right\}$$

Queries: Shape Casting

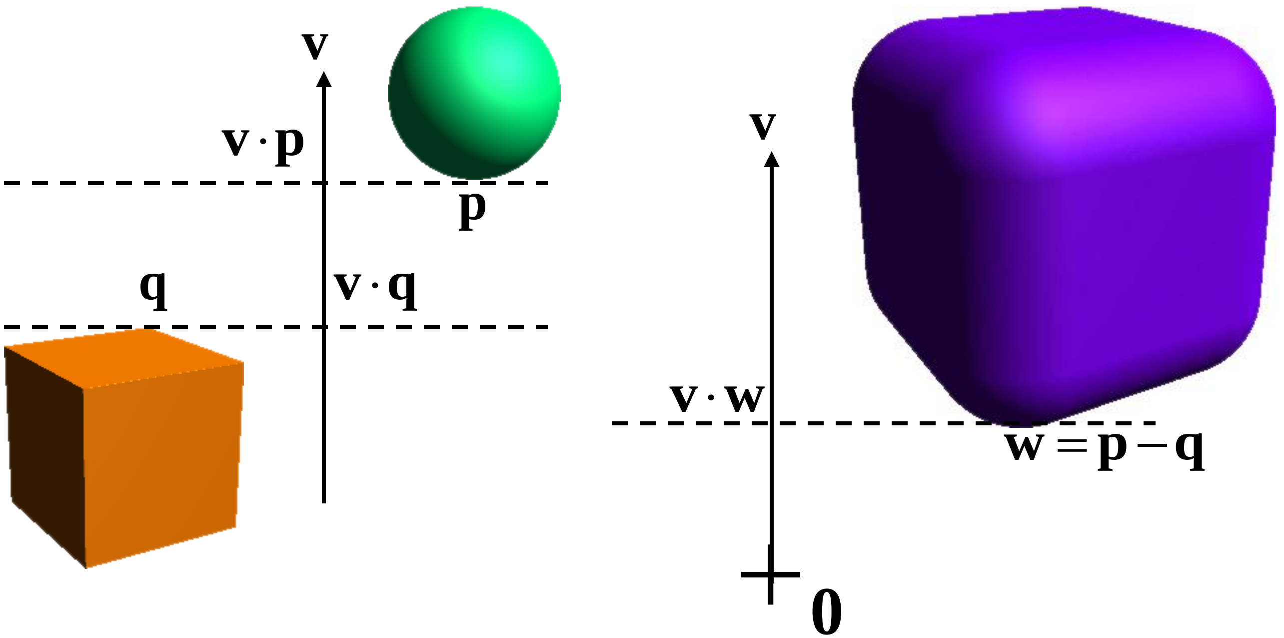
- Finding collisions that occur over a frame for A translated over $\mathbf{s}$ and B over $\mathbf{t}$ boils down to a ray cast from the origin onto the CSO along the vector $\mathbf{r} = \mathbf{t} - \mathbf{s}$.

$$\min\{\lambda : \lambda\mathbf{r} \in A - B, 0 \leq \lambda \leq 1\}$$

Ray Query on the CSO



Separating Axis



Separating Axis Theorem (SAT)

- For each pair of disjoint polytopes, of which at least one has a volume, there exists a separating axis that is orthogonal to:
- a face of either polytope, or
- an edge from each polytope

SAT Sketchy Proof

- The CSO of polytopes is a polytope and has a volume.
- For disjoint polytopes, the origin lies on the outside of at least one face of the CSO.
- A face of the CSO is either the CSO of a face and a vertex or of two edges.

Separating Axis Method

- Test all face normals and all cross products of edge directions.
- If none of these vectors yield a separating axis then the polytopes must intersect.
- Given polytopes with resp. f_1 and f_2 faces and e_1 and e_2 edge directions, we need to test at most $f_1 + f_2 + e_1 * e_2$ axes.

Separating Axis Method

Polytope 1	Polytope 2	#Axes
Line segment	Box	
Triangle	Box	
Box	Box	
Tetrahedron	Tetrahedron	

Separating Axis Method

Polytope 1	Polytope 2	#Axes
Line segment	Box	0 + 3 + 1*3 = 6
Triangle	Box	
Box	Box	
Tetrahedron	Tetrahedron	

Separating Axis Method

Polytope 1	Polytope 2	#Axes
Line segment	Box	0 + 3 + 1*3 = 6
Triangle	Box	1 + 3 + 3*3 = 13
Box	Box	
Tetrahedron	Tetrahedron	

Separating Axis Method

Polytope 1	Polytope 2	#Axes
Line segment	Box	0 + 3 + 1*3 = 6
Triangle	Box	1 + 3 + 3*3 = 13
Box	Box	3 + 3 + 3*3 = 15
Tetrahedron	Tetrahedron	

Separating Axis Method

Polytope 1	Polytope 2	#Axes
Line segment	Box	$0 + 3 + 1*3 = 6$
Triangle	Box	$1 + 3 + 3*3 = 13$
Box	Box	$3 + 3 + 3*3 = 15$
Tetrahedron	Tetrahedron	$4 + 4 + 6*6 = 44$

Separating Axis Queries

- Suitable for intersection testing, most notably in bounding box hierarchies.
- Too expensive for general polytopes due to $O(n^3)$ complexity.
- In case of intersection, the axis for which overlap is shallowest is a proper direction for the penetration depth vector.

GJK Does It All

- GJK is an iterative method that computes closest points.
- The GJK ray cast can perform continuous collision detection.
- The *expanding polytope algorithm* (EPA) returns the penetration-depth vector.

GJK Algorithm

- Approximate the point of the CSO closest to the origin by generating a sequence of *simplices* inside the CSO.
- A *simplex* is a point, a line segment, a triangle, or a tetrahedron.
- Each new simplex lies closer to the origin than its predecessor.

GJK Algorithm (cont'd)

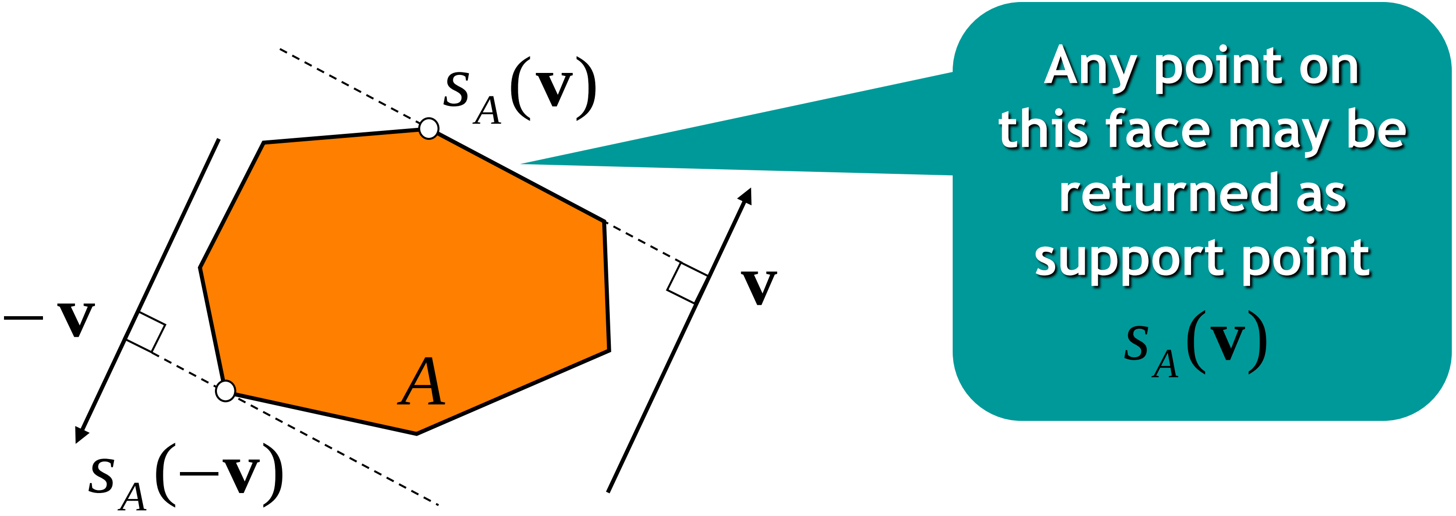
- Simplex vertices are computed using support mappings. (Definition follows.)
- Terminate as soon as the current simplex is close enough.
- In case of an intersection, the simplex contains the origin.

Support Mappings

- A support mapping s_A of an object A maps a vector $\mathbf{v}$ to a point of A that lies furthest in the direction of $\mathbf{v}$.

$$\mathbf{v} \cdot s_A(\mathbf{v}) = \max\{\mathbf{v} \cdot \mathbf{x} : \mathbf{x} \in A\}$$

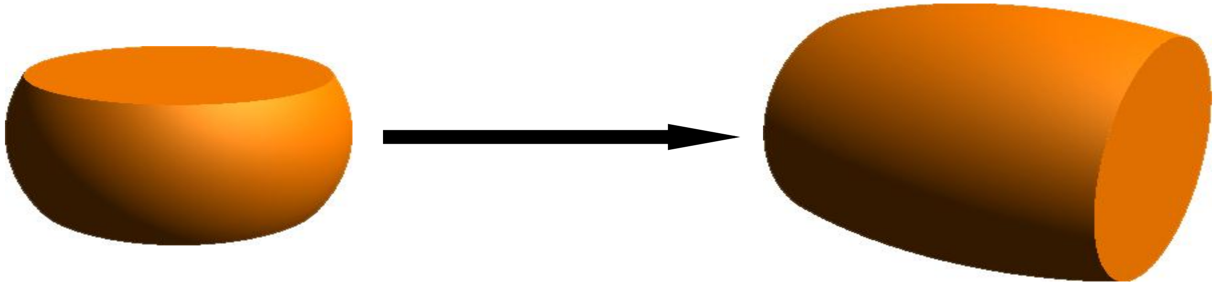
Support Mappings



Affine Transformation

- Shapes can be translated, rotated, *and* scaled. For $\mathbf{T}(\mathbf{x}) = \mathbf{B}\mathbf{x} + \mathbf{c}$, we have

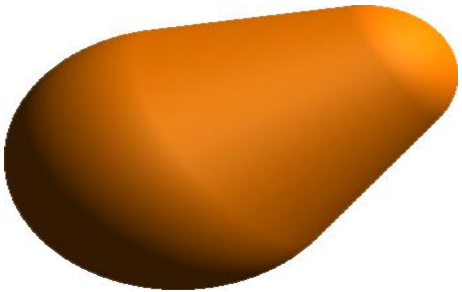
$$s_{\mathbf{T}(A)}(\mathbf{v}) = \mathbf{T}(s_A(\mathbf{B}^{\mathrm{T}}\mathbf{v}))$$

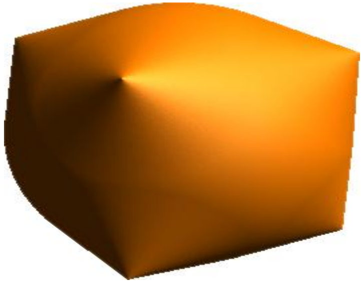


Convex Hull

- Convex hulls of arbitrary convex shapes are readily available.

$$s_{\text{conv}\{X_0,\ldots,X_{n-1}\}}(\mathbf{v}) = s_{\{s_{X_0}(\mathbf{v}),\ldots,s_{X_{n-1}}(\mathbf{v})\}}(\mathbf{v})$$



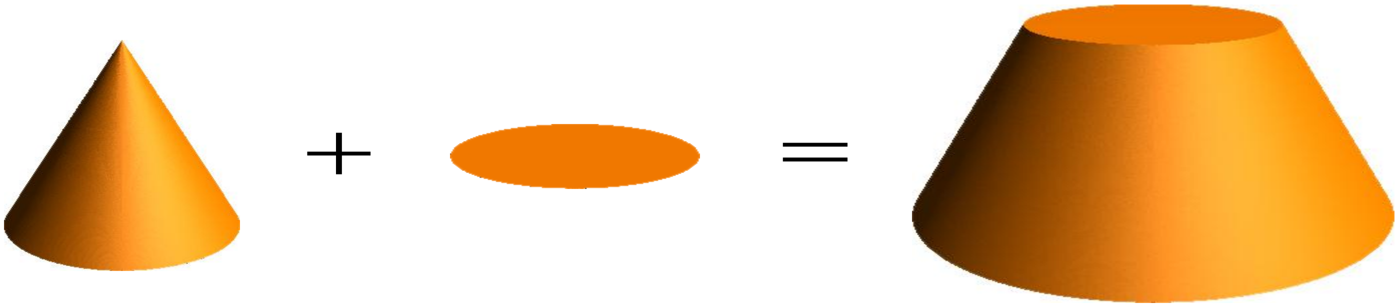


Minkowski Sum

- Shapes can be fattened by *Minkowski addition*.

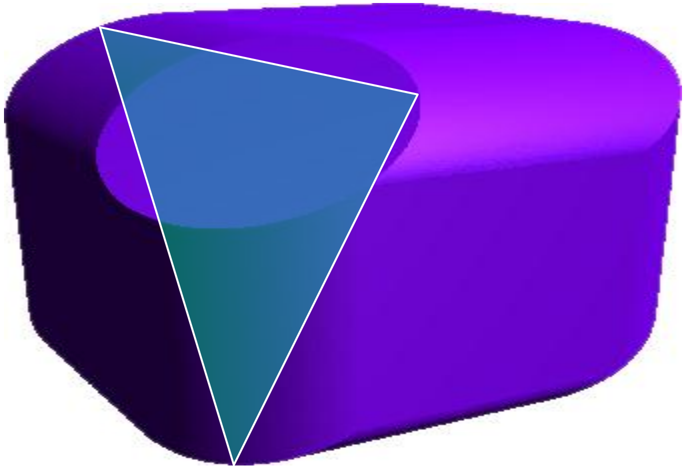
$$s_{A+B}(\mathbf{v}) = s_A(\mathbf{v}) + s_B(\mathbf{v})$$

$$s_{A-B}(\mathbf{v}) = s_A(\mathbf{v}) - s_B(-\mathbf{v})$$



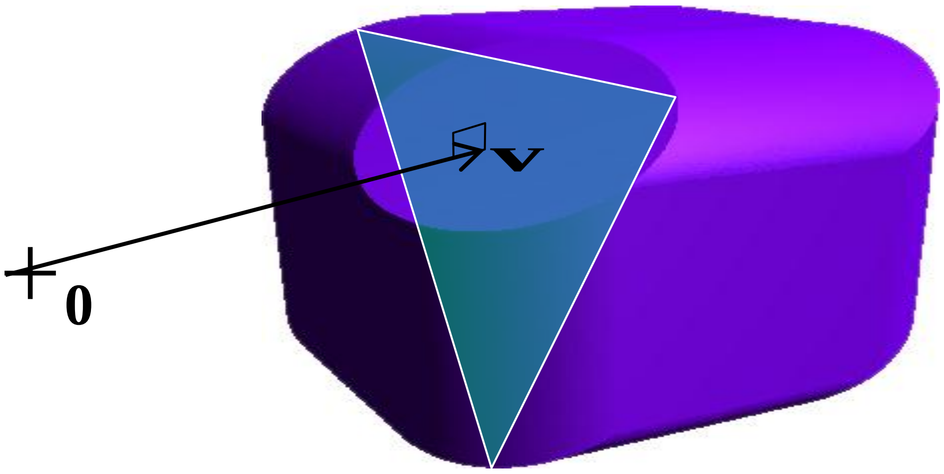
GJK Steps (1/6)

- Suppose we have a simplex inside the CSO…



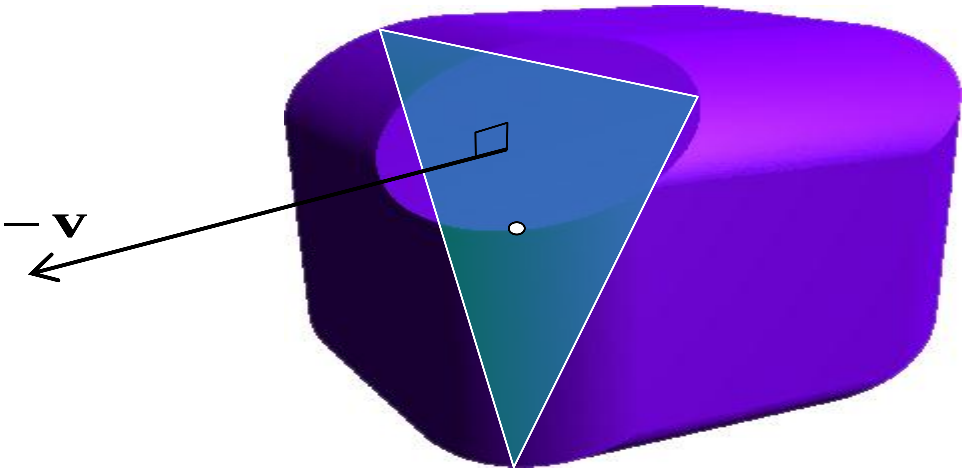
GJK Steps (2/6)

- …and the point **v** of the simplex closest to the origin.



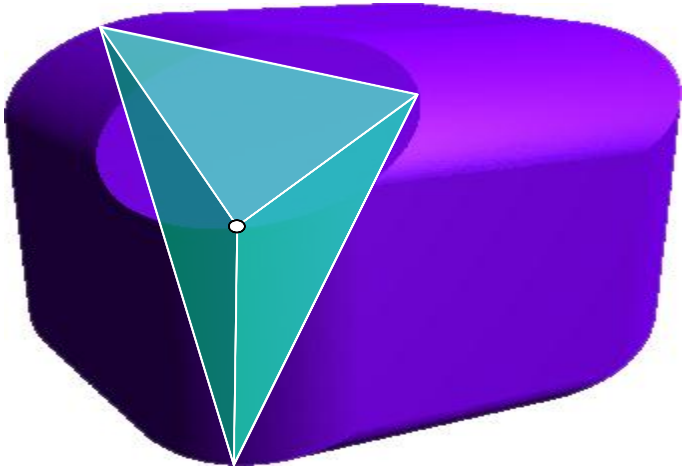
GJK Steps (3/6)

- Compute support point $\mathbf{w} = s_{A\text{-}B}(-\mathbf{v})$.



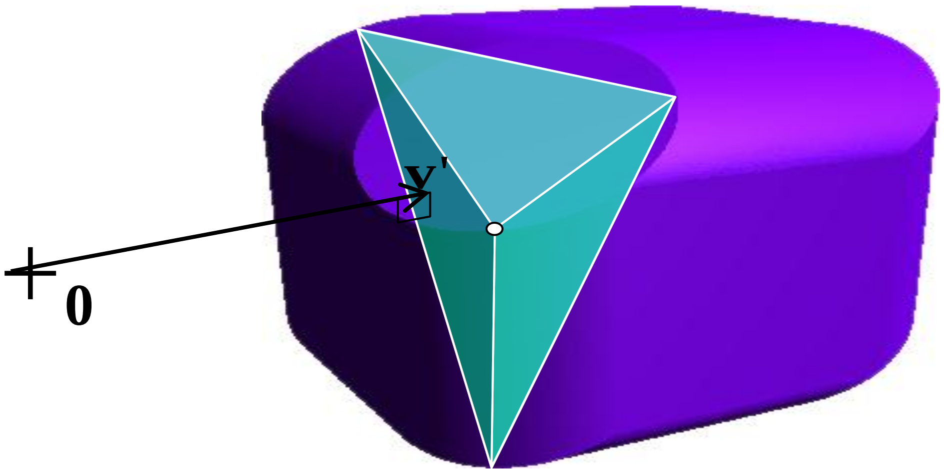
GJK Steps (4/6)

- Add support point **w** to the current simplex.



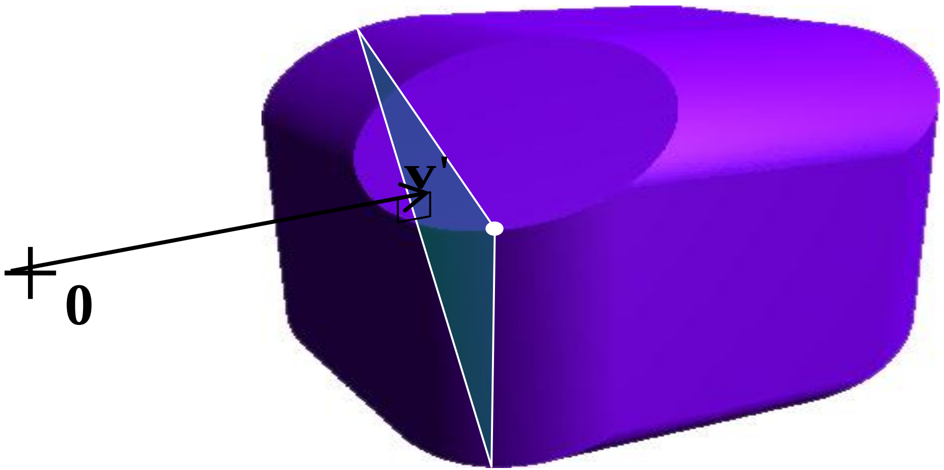
GJK Steps (5/6)

- Compute the closest point **v'** of the new simplex.



GJK Steps (6/6)

- Discard all vertices that do not contribute to **v**′.

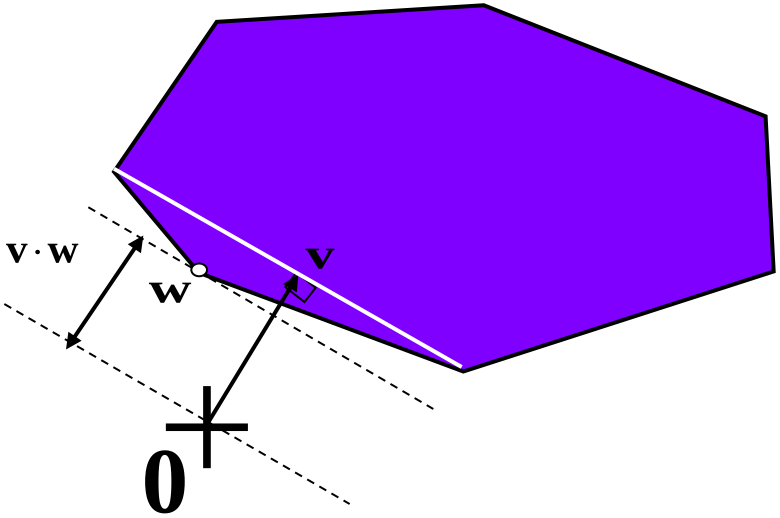


Separating Axis

- If only an intersection test is needed then let GJK terminate as soon as the lower bound $\mathbf{v} \cdot \mathbf{w}$ becomes positive.
- For a positive lower bound $\mathbf{v} \cdot \mathbf{w}$, the vector $\mathbf{v}$ is a separating axis.

Separating Axis (cont'd)

- The supporting plane through $\mathbf{w}$ separates the origin from the CSO.

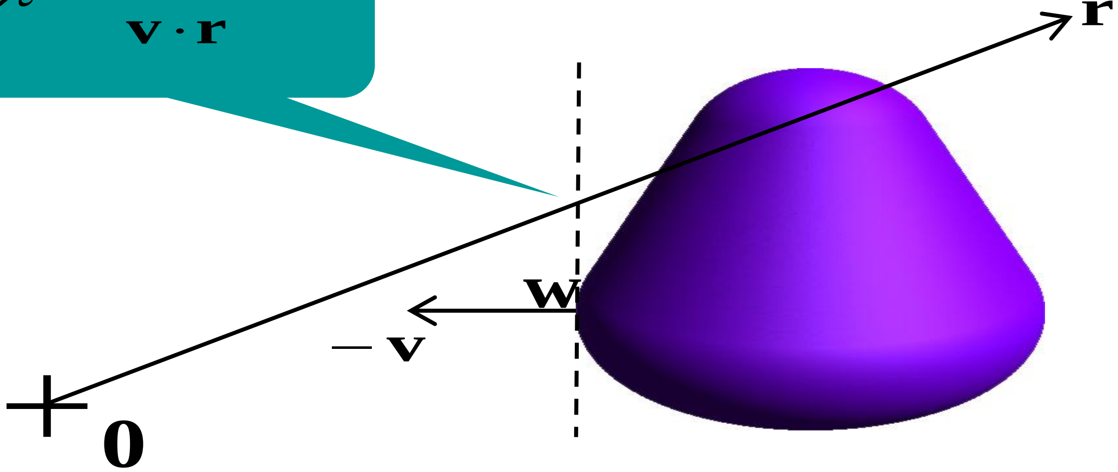


Separating Axes and Coherence

- Separating axes can be cached and reused as initial $\mathbf{v}$ in future tests on the same object pair.
- When the degree of frame coherence is high, the cached $\mathbf{v}$ is likely to be a separating axis in the new frame as well.
- An incremental version of GJK takes roughly one iteration per frame for smoothly moving objects.

GJK Ray Cast

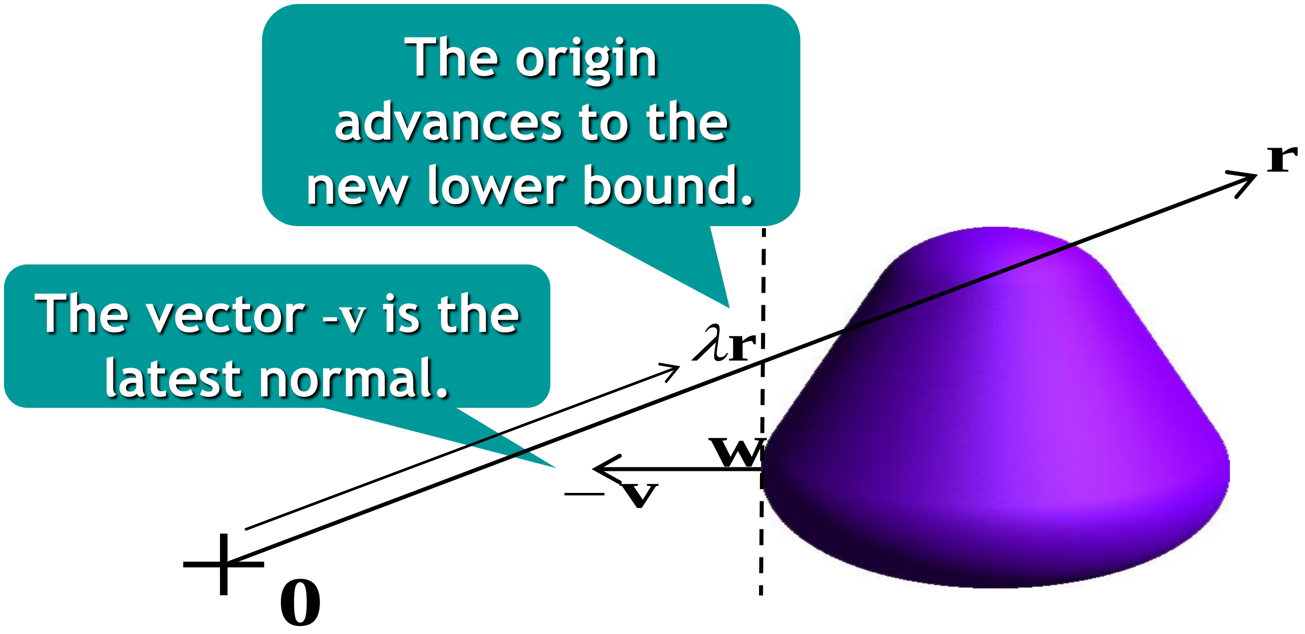
$$\lambda = \frac{\mathbf{v} \cdot \mathbf{w}}{\mathbf{v} \cdot \mathbf{r}}$$



GJK Ray Cast

- Do a standard GJK iteration, and use the support planes as clipping planes.
- Each time the ray is clipped, the clip point $\lambda\mathbf{r}$ becomes the new origin.
- …and the new simplex is the last-found support point $\mathbf{w}$ wrt the new origin.
- The normal $-\mathbf{v}$ of the last clipping plane is the normal at the hit point.

GJK Ray Cast

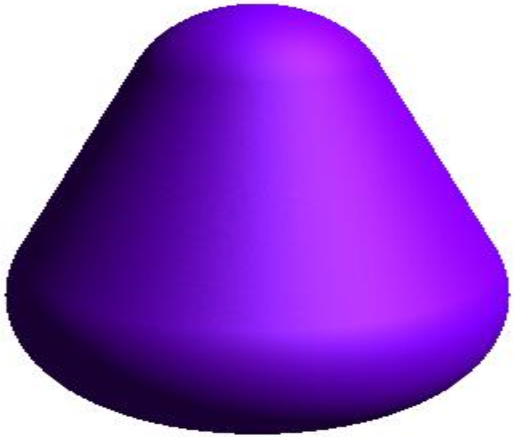

The origin advances to the new lower bound.
The vector -v is the latest normal.
r
λr
w
−v
0

Accuracy vs. Performance

- Accuracy can be traded for performance by tweaking the error tolerance ε_{tol}.
- A greater tolerance results in fewer iterations but less accurate hit points and normals.

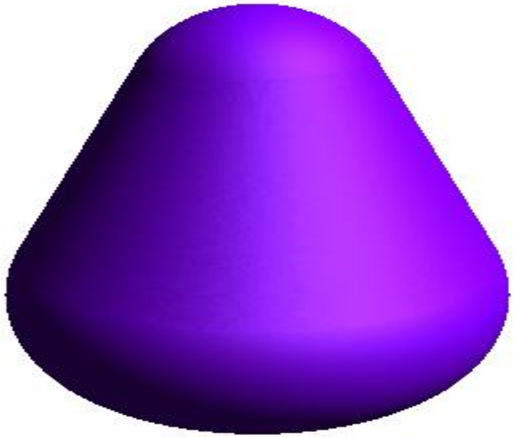
Accuracy vs. Performance

$\varepsilon_{tol} = 10^{-7}$, avg. time: 3.65 μs @ 2.6 GHz



Accuracy vs. Performance

$\varepsilon_{tol} = 10^{-6}$, avg. time: 2.80 μs @ 2.6 GHz



Accuracy vs. Performance

$\varepsilon_{tol} = 10^{-5}$, avg. time: 2.03 μs @ 2.6 GHz

Accuracy vs. Performance

$\varepsilon_{tol} = 10^{-4}$, avg. time: 1.43 μs @ 2.6 GHz

Accuracy vs. Performance

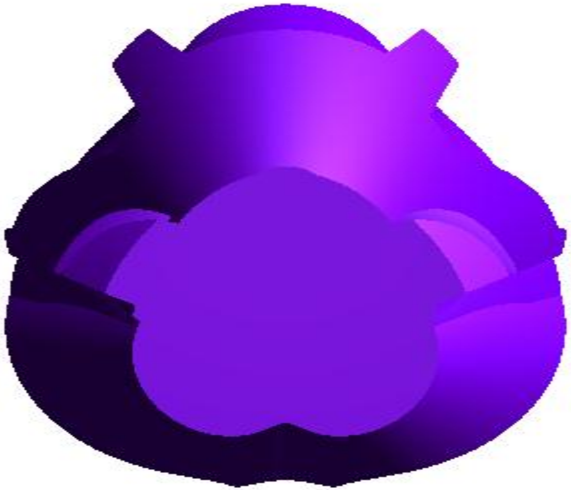
$\varepsilon_{tol} = 10^{-3}$, avg. time: 1.02 μs @ 2.6 GHz

Accuracy vs. Performance

$\varepsilon_{\text{tol}} = 10^{-2}$, avg. time: 0.77 µs @ 2.6 GHz

Accuracy vs. Performance

$\varepsilon_{tol} = 10^{-1}$, avg. time: 0.62 μs @ 2.6 GHz



GJK Algorithm: Pros

- Extremely versatile:
  - Applicable to any combination of convex shape types.
  - Computes distances, common points, and separating axes.
  - Can be tailored for finding space-time collisions.
  - Allows a smooth trade-off between accuracy and speed.

GJK Algorithm: Pros (cont'd)

- Performs well:
  - Exploits frame coherence.
  - Competitive with dedicated solutions for polytopes (Lin-Canny, V-Clip, SWIFT) .
- Despite its conceptual complexity, implementing GJK is not too difficult.
- Small code size.

GJK Algorithm: Cons

- Difficult to grasp:
  - Concepts from linear algebra and convex analysis (determinants, Minkowski addition), take some time to get comfortable with.
  - Maintaining a “geometric” mental image of the workings of the algorithm is challenging and not very helpful.

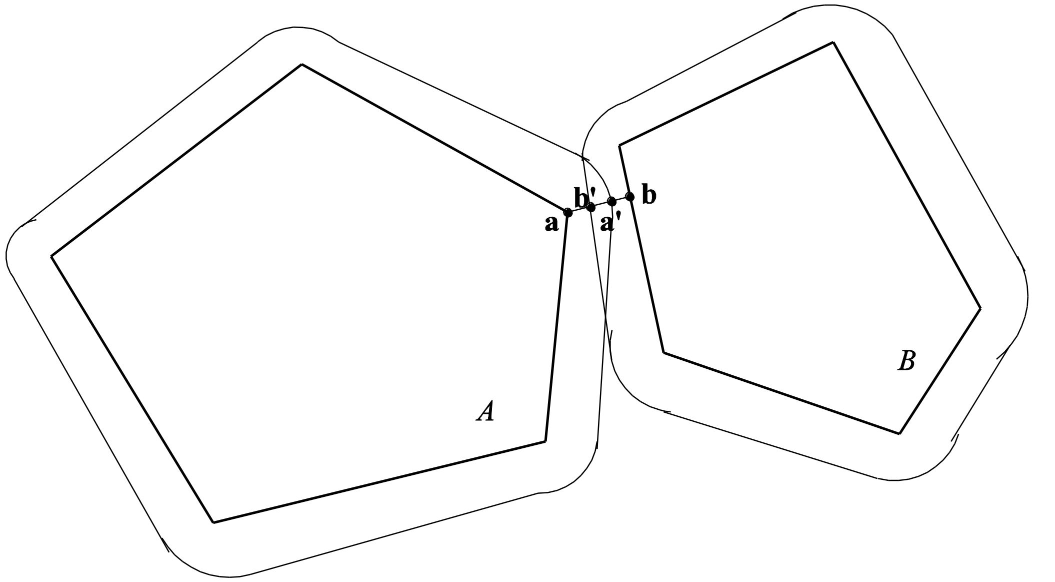
GJK Algorithm: Cons (cont'd)

- Suffers from numerical issues:
  - Termination is governed by predicates that rely on tolerances.
  - Despite the use of tolerances, certain “hacks” are needed in order to guarantee termination in all cases.
  - Using 32-bit floating-point numbers is doable but tricky.

Resting Contacts

- Contact data for resting contacts are obtained through a hybrid approach.
- Objects are dilated slightly to add a skin.
- For interpenetrations that are only skin-deep the closest points of the “bones” give us the contact data.

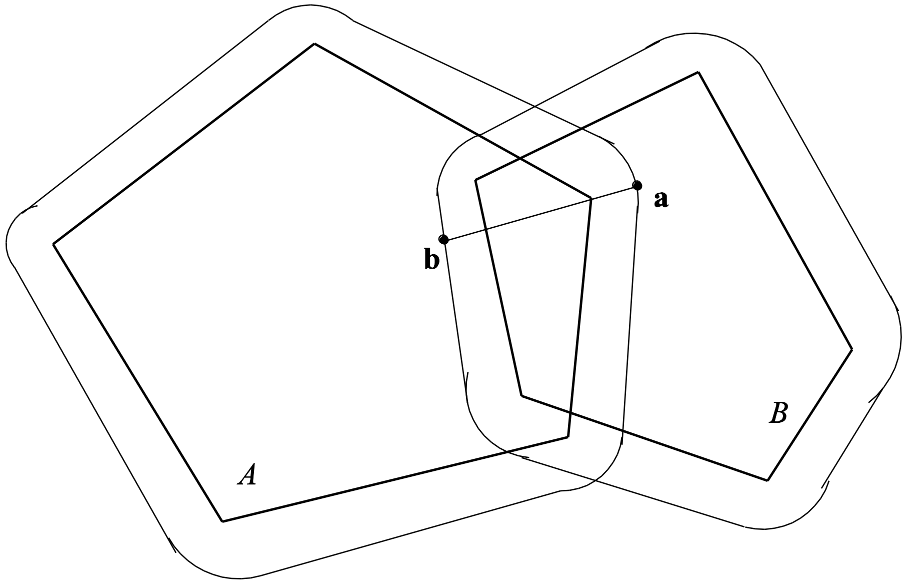
Shallow Interpenetrations



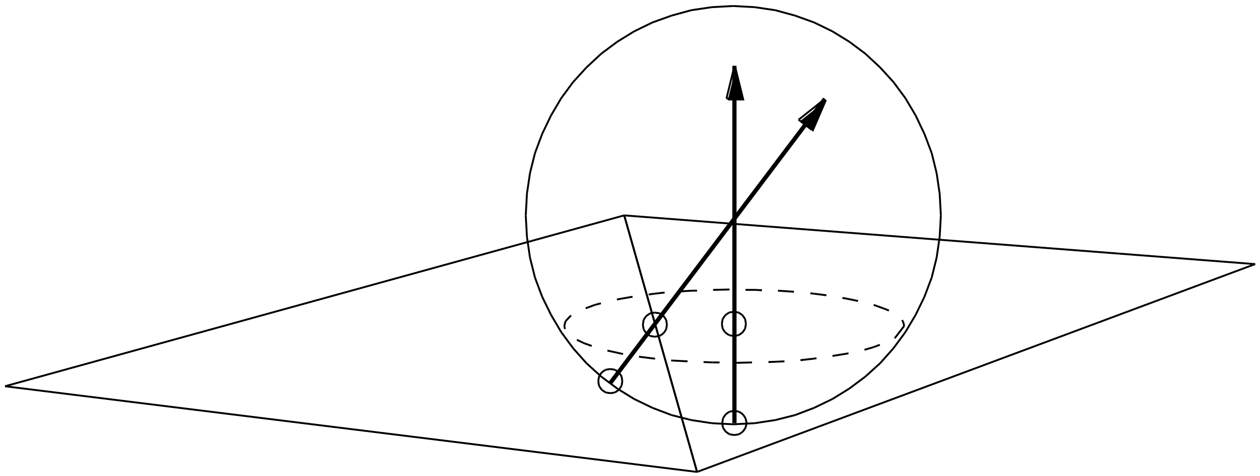
Resting Contacts

- For deeper interpenetrations contact data are obtained from the penetration-depth vector.
- This should only be necessary in emergencies.

Deep Interpenetrations



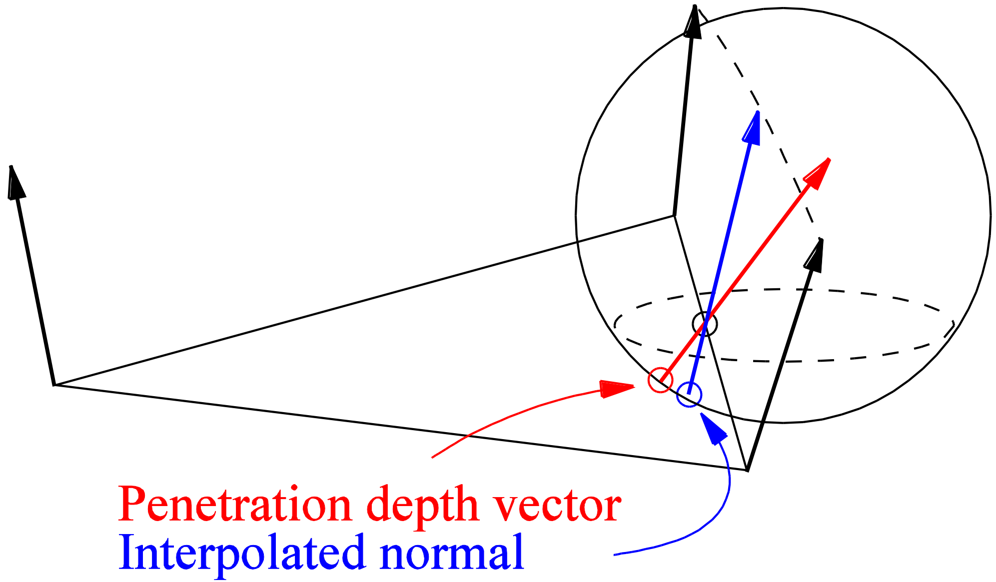
Meshes Have Bumpy Edges



Solving Bumpy Edges

- Obtain *barycentric coordinates* of the closest point returned by GJK.
- Use these coordinates to interpolate the vertex normals.
- Similar to Phong shading: Use a normalized lerp.

Smooth Interpolated Normals



References

- Gilbert, Johnson, and Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4(2):192-203, 1988.
- Gottschalk, Lin, and Manocha. OBBTree: a hierarchical structure for rapid interference detection. Proc. SIGGRAPH '96.

References (cont'd)

- Gino van den Bergen. *Collision Detection in Interactive 3D Environments*. Morgan Kaufmann Publishers, 2004.
- Gino van den Bergen. *Smooth Mesh Contacts with GJK*. In *Game Physics Pearls*, A K Peters, 2010.

Thank You!

- For papers and other information, check:

  www.dtecta.com