The Gilbert-Johnson-Keerthi (GJK) Algorithm

Christer Ericson
Sony Computer Entertainment America

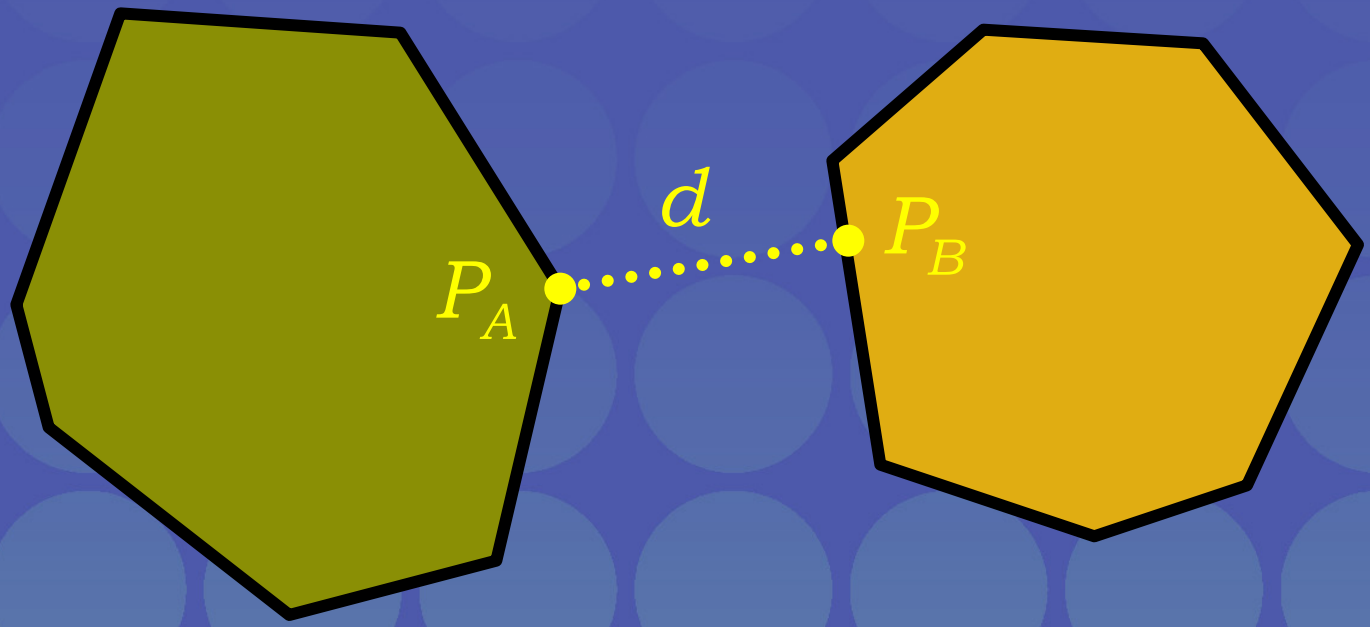
christer_ericson@playstation.sony.com

Talk outline

- What is the GJK algorithm
- Terminology
- “Simplified” version of the algorithm
  - One object is a point at the origin
  - Example illustrating algorithm
- The distance subalgorithm
- GJK for two objects
  - One no longer necessarily a point at the origin
- GJK for moving objects

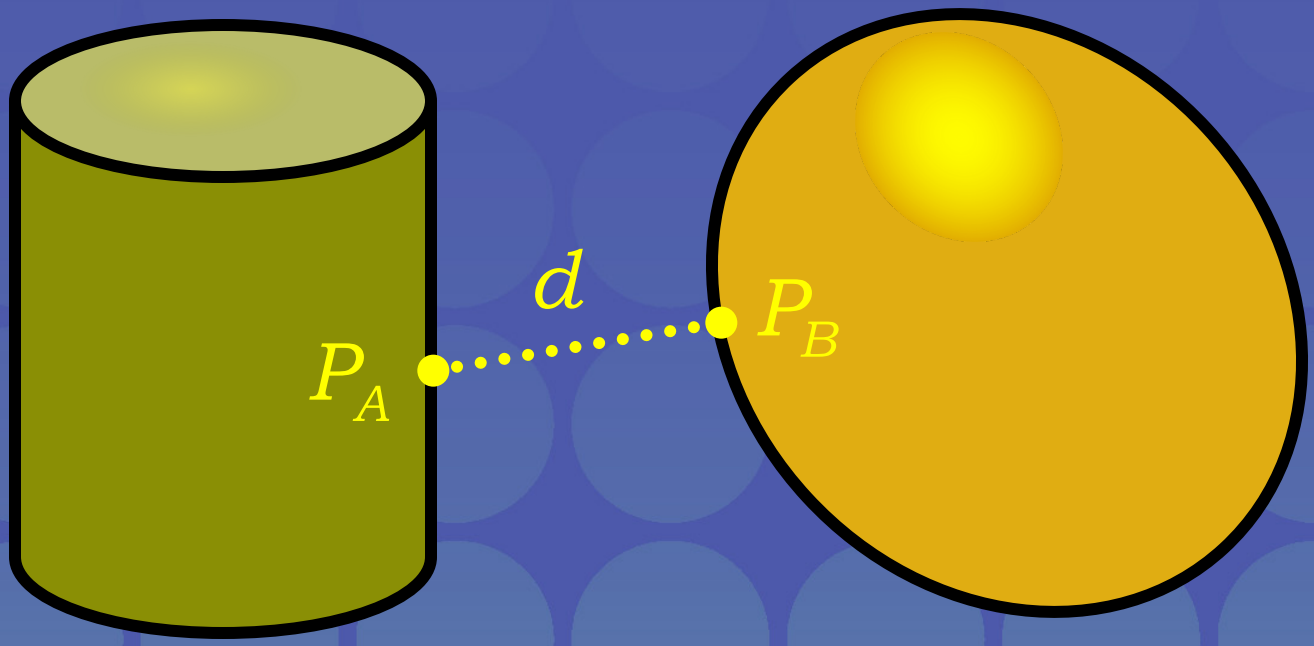
GJK solves proximity queries

- Given two convex polyhedra
  - Computes distance *d*
  - Can also return closest pair of points P_A, P_B

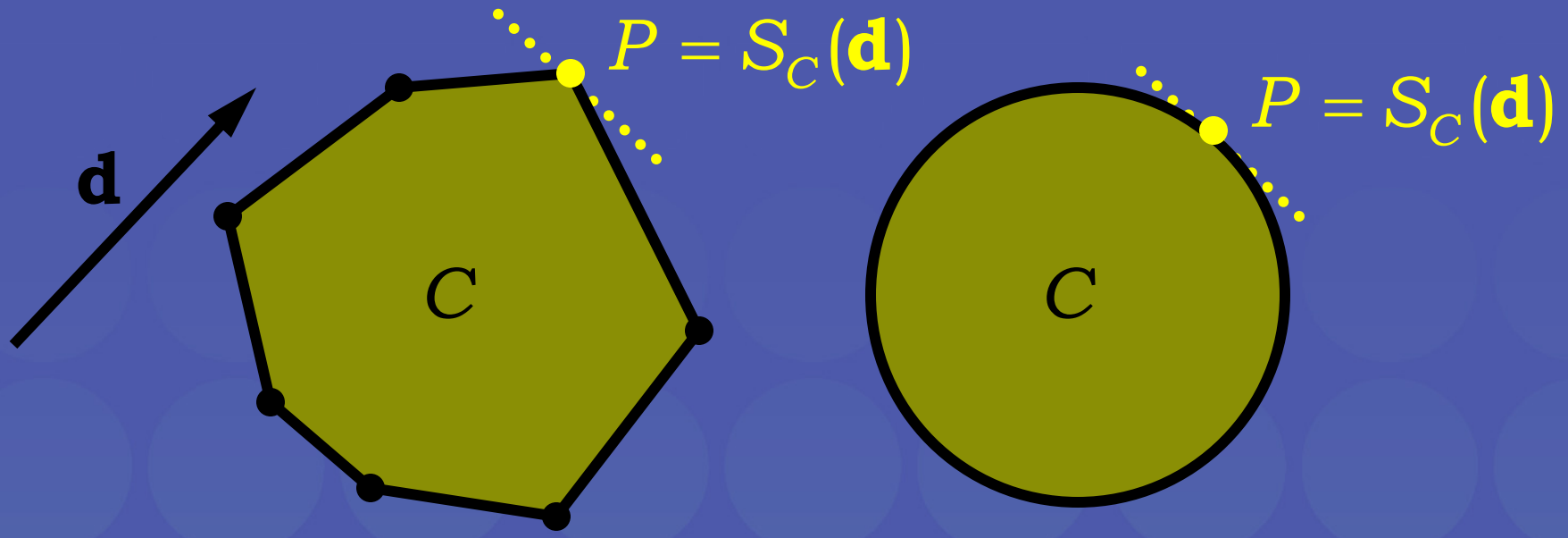


GJK solves proximity queries

- Generalized for arbitrary convex objects
  - As long as they can be described in terms of a *support mapping* function



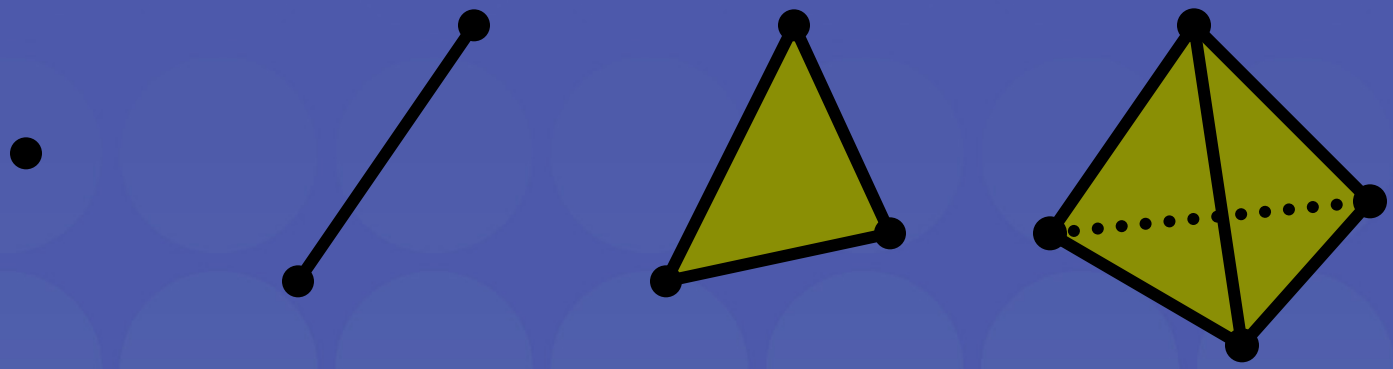
Terminology 1(3)



Supporting (or extreme) point P for direction $\mathbf{d}$

returned by support mapping function $S_C(\mathbf{d})$

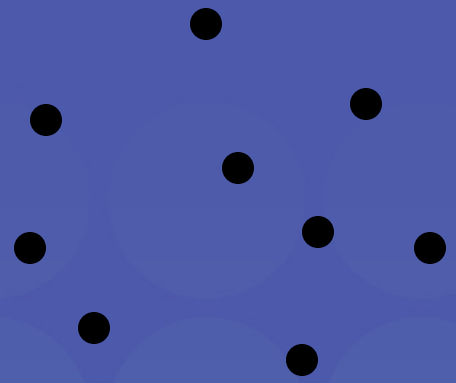
Terminology 2(3)



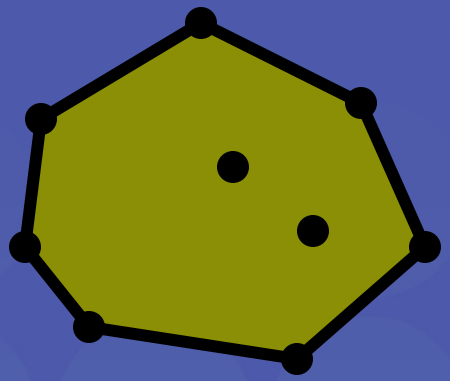
0-simplex 1-simplex 2-simplex 3-simplex

simplex

Terminology 3(3)



Point set *C*



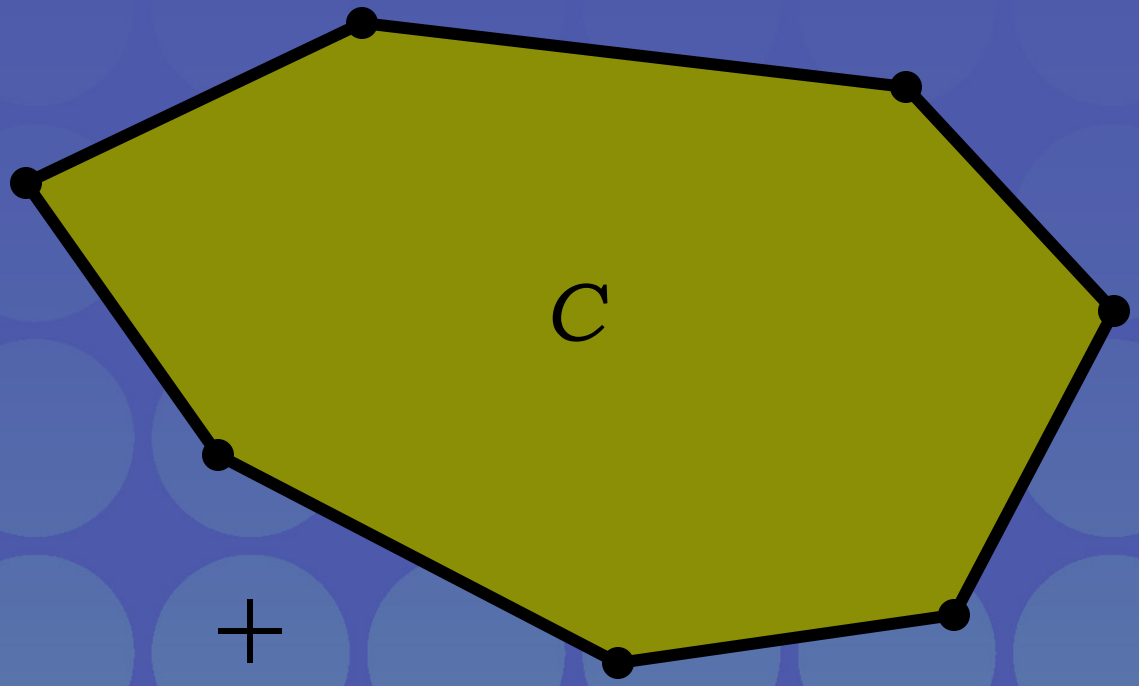
Convex hull, CH(*C*)

The GJK algorithm

1. Initialize the simplex set Q with up to d+1 points from C (in d dimensions)
2. Compute point P of minimum norm in CH(Q)
3. If P is the origin, exit; return 0
4. Reduce Q to the smallest subset Q' of Q, such that P in CH(Q')
5. Let $V=S_C(-P)$ be a supporting point in direction $-P$
6. If V no more extreme in direction $-P$ than P itself, exit; return $||P||$
7. Add V to Q. Go to step 2

GJK example 1(10)

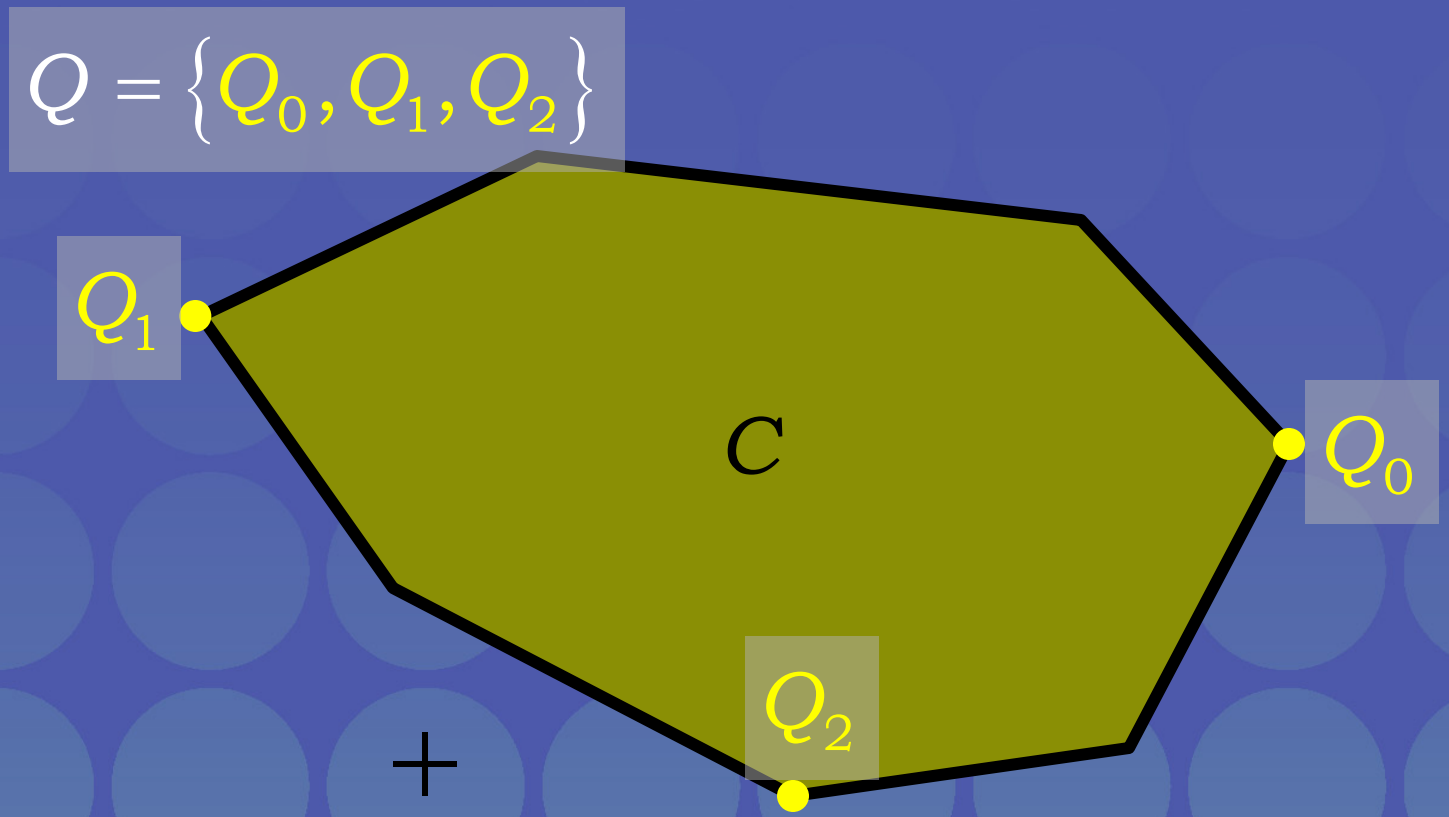
INPUT: Convex polyhedron *C* given as the convex hull of a set of points



GJK example 2(10)

SIGGRAPH2004

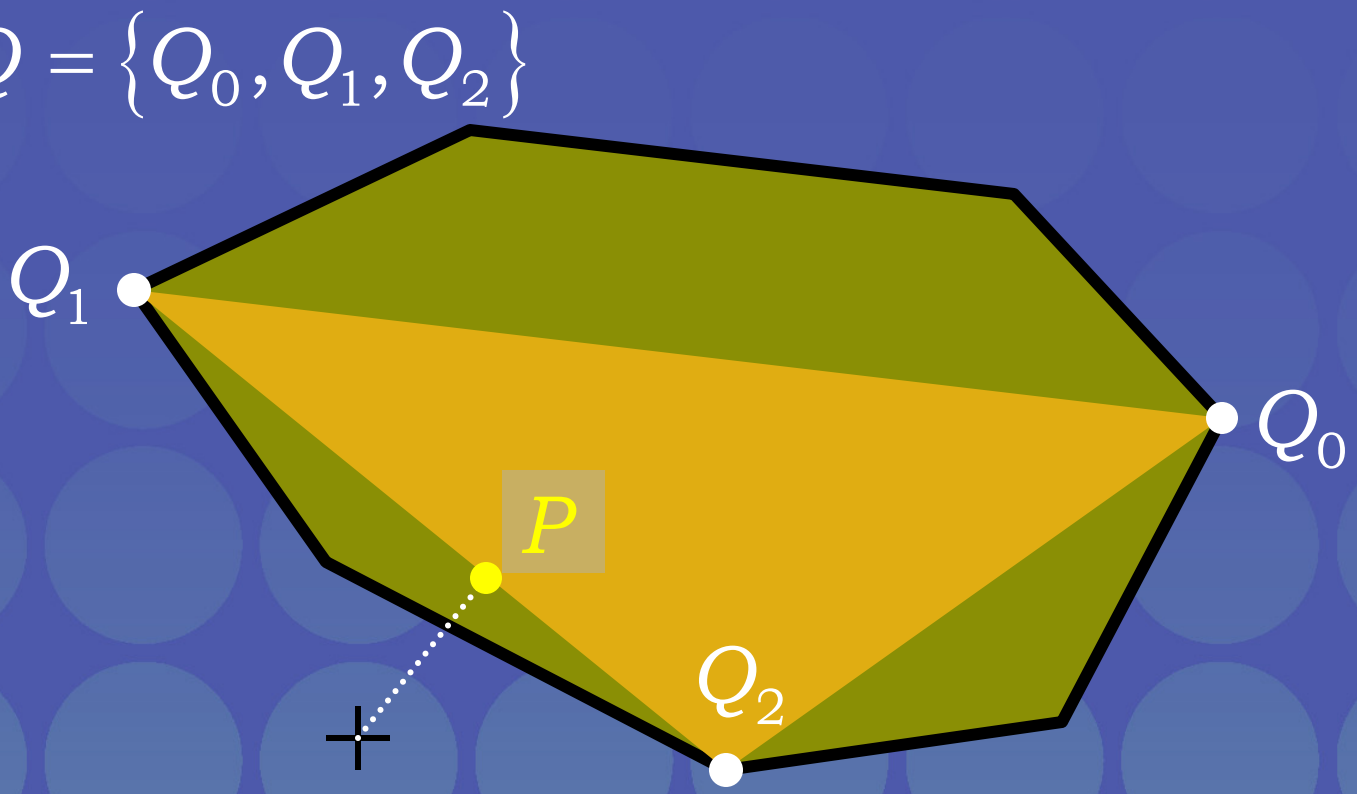
1. Initialize the simplex set *Q* with up to *d*+1 points from *C* (in *d* dimensions)



GJK example 3(10)

SIGGRAPH2004

2. Compute point *P* of minimum norm in CH(*Q*)

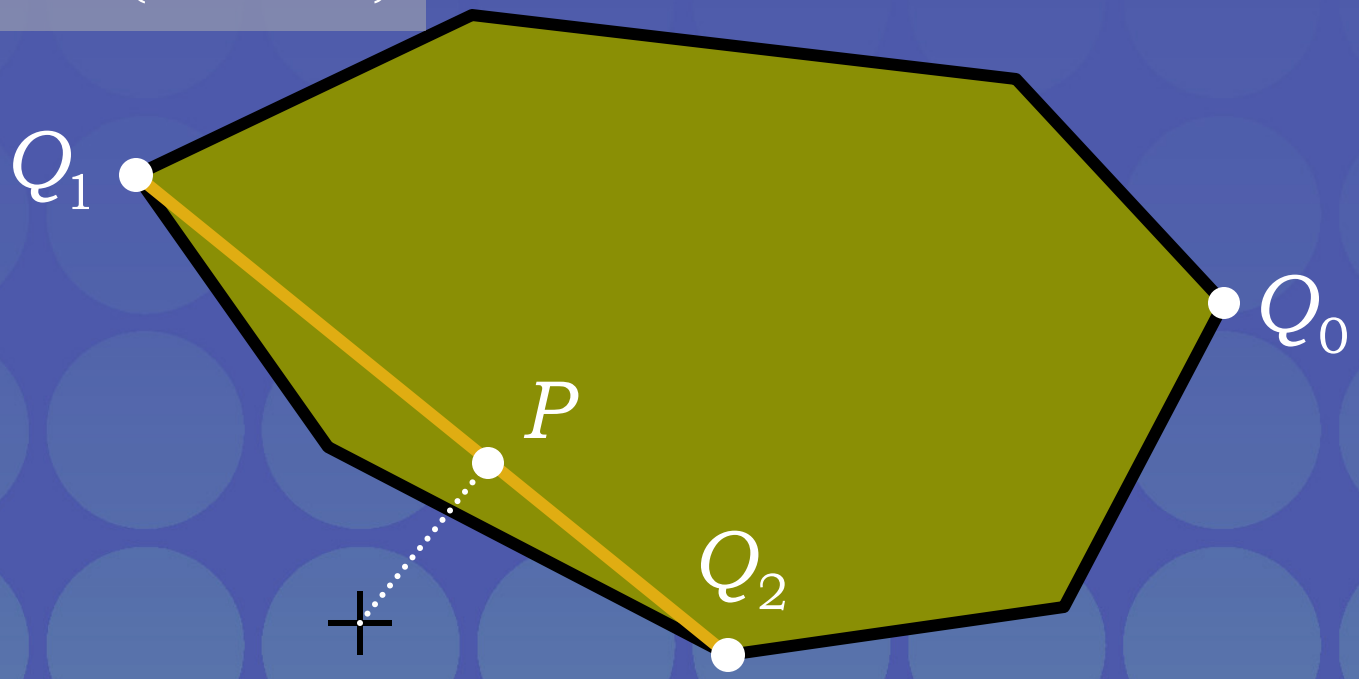


GJK example 4(10)

SIGGRAPH2004

3. If *P* is the origin, exit; return 0
4. Reduce *Q* to the smallest subset *Q'* of *Q*, such that *P* in CH(*Q'*)

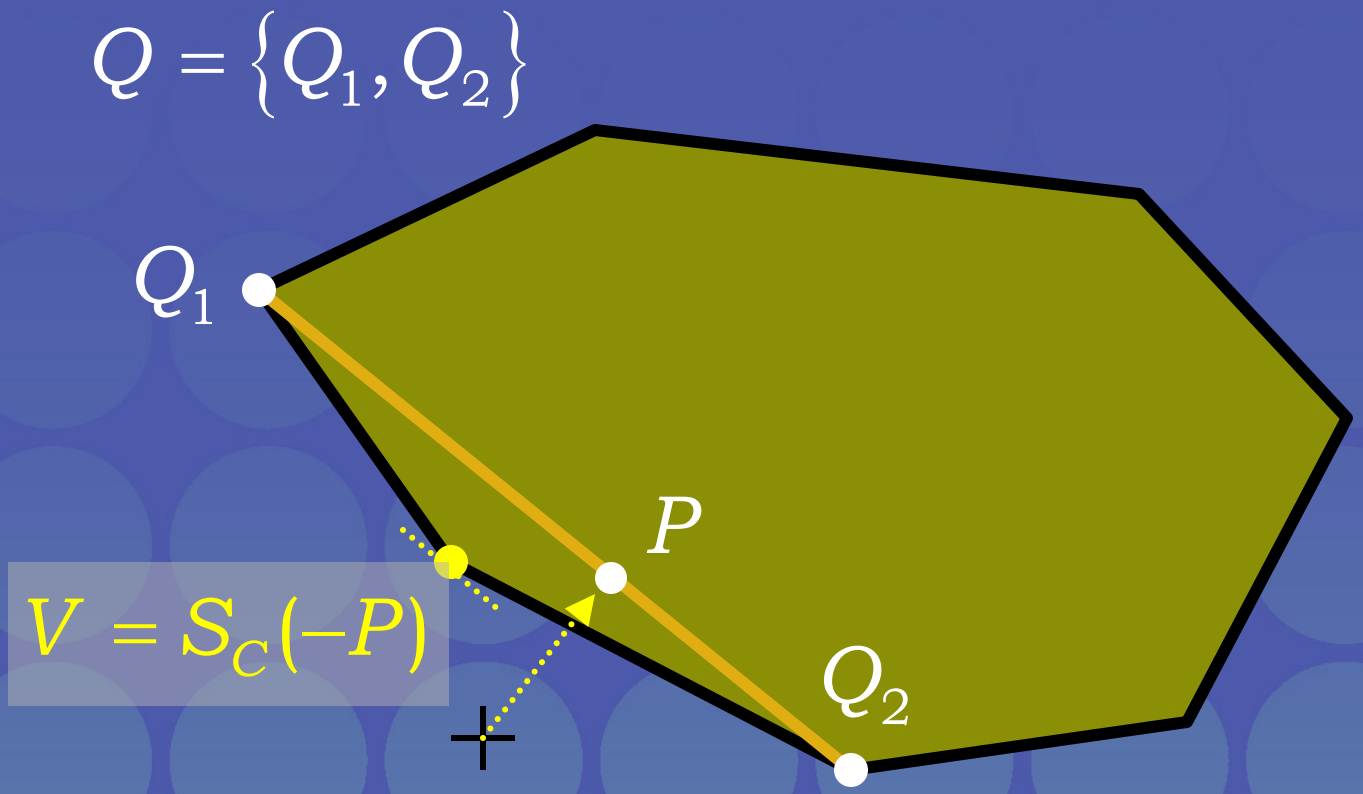
$$Q = \{Q_1, Q_2\}$$



GJK example 5(10)

SIGGRAPH2004

5. Let $V=S_C(-P)$ be a supporting point in direction $-P$

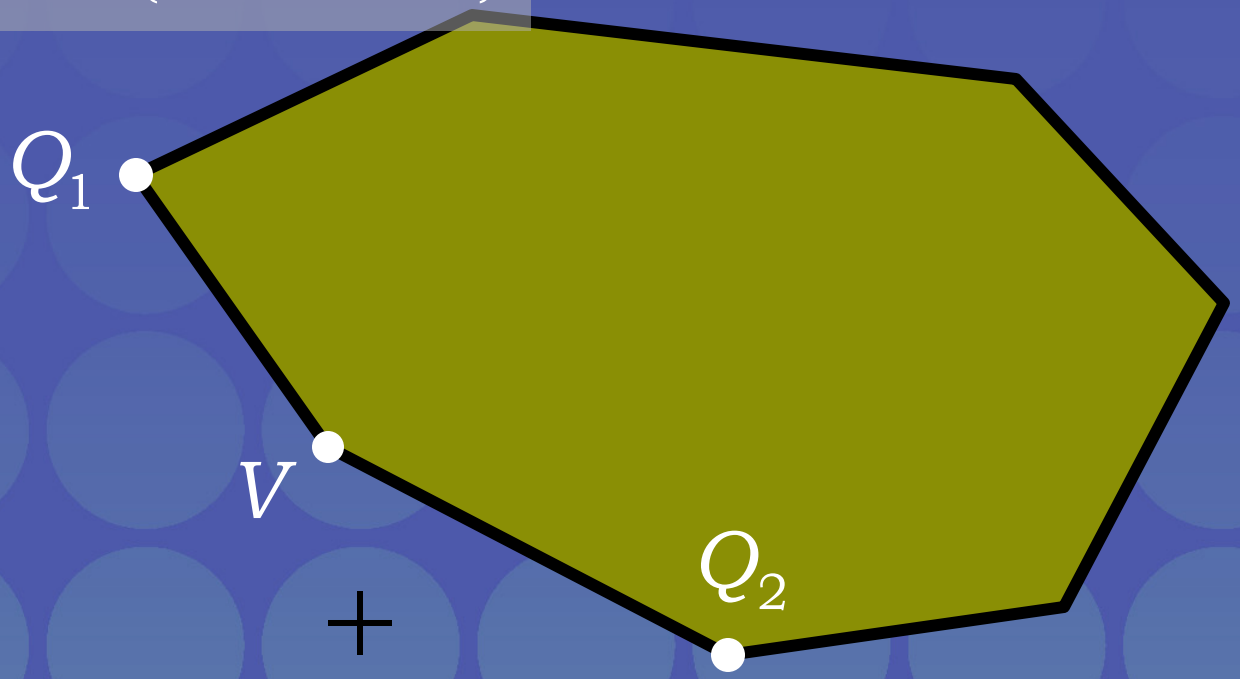


GJK example 6(10)

SIGGRAPH2004

6. If *V* no more extreme in direction –*P* than *P* itself, exit; return ||*P*||
7. Add *V* to *Q*. Go to step 2

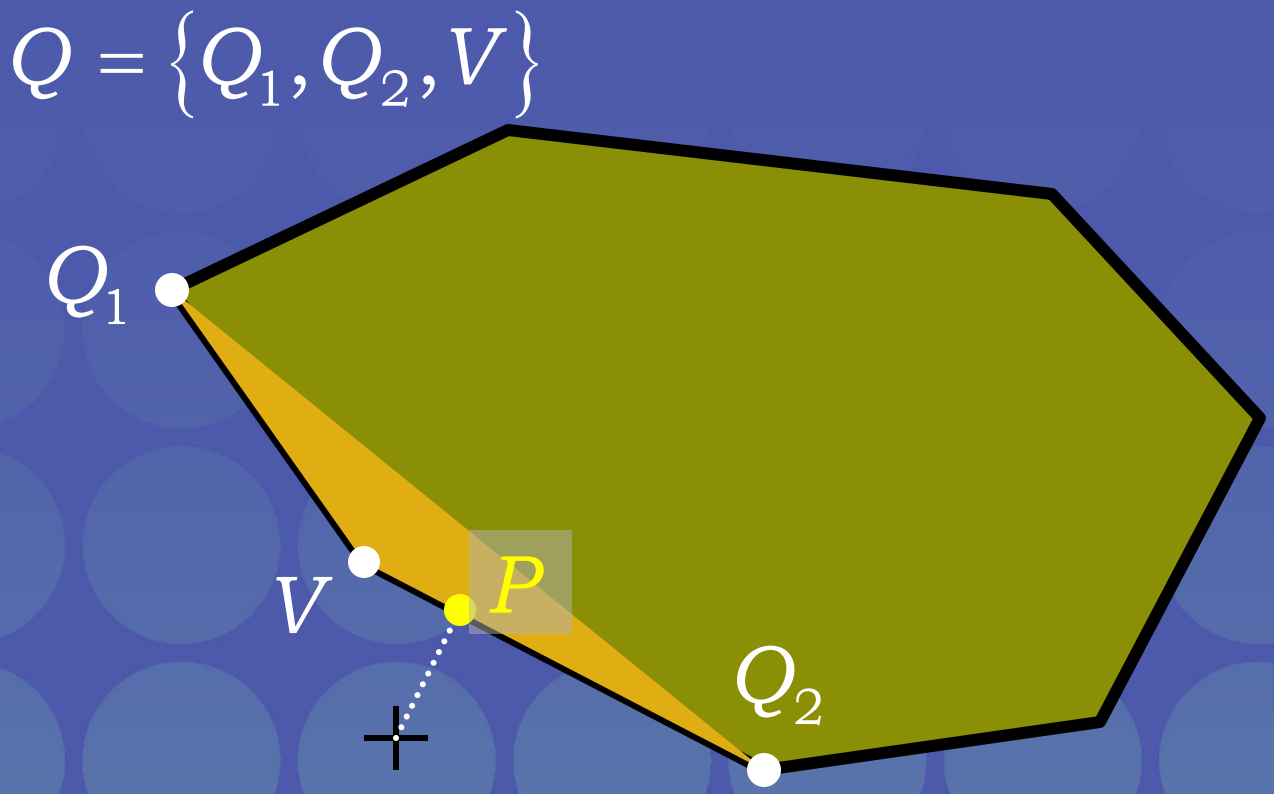
$$Q = \{Q_1, Q_2, V\}$$



GJK example 7(10)

SIGGRAPH2004

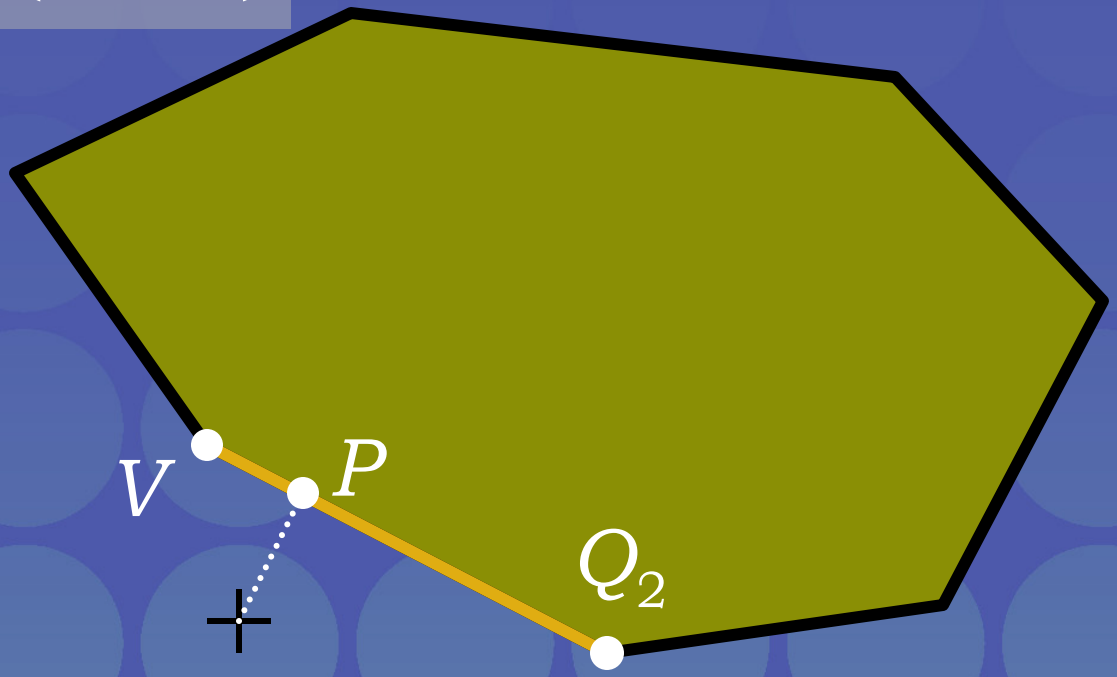
2. Compute point *P* of minimum norm in CH(*Q*)



GJK example 8(10)

3. If *P* is the origin, exit; return 0
4. Reduce *Q* to the smallest subset *Q'* of *Q*, such that *P* in CH(*Q'*)

$$Q = \{Q_2, V\}$$

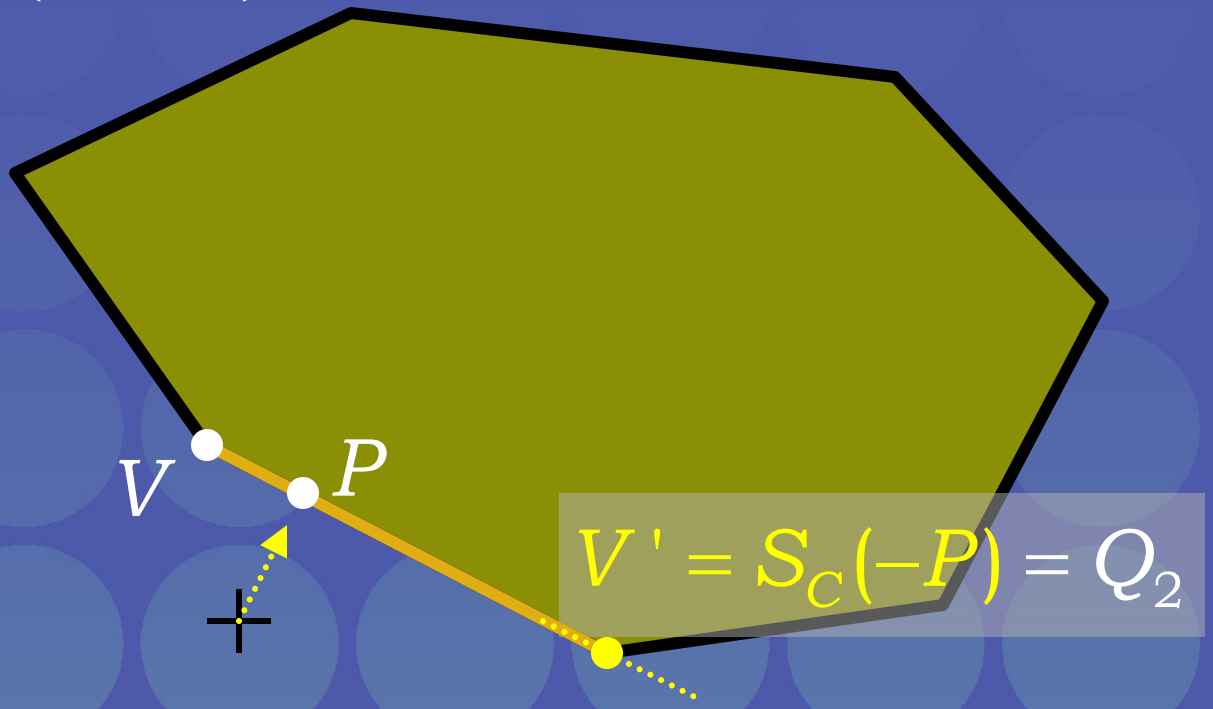


GJK example 9(10)

SIGGRAPH2004

5. Let $V=S_C(-P)$ be a supporting point in direction $-P$

$$Q = \{Q_2, V\}$$

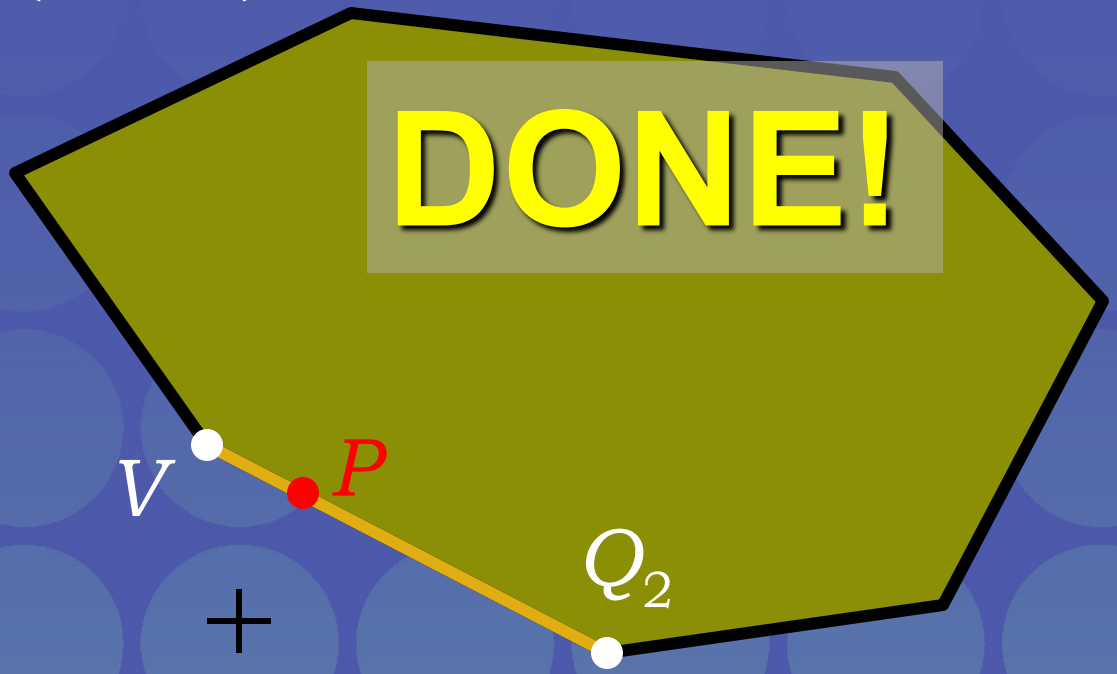


GJK example 10(10)

SIGGRAPH2004

6. If *V* no more extreme in direction –*P* than *P* itself, exit; return ||*P*||

$$Q = \{Q_2, V\}$$



Distance subalgorithm 1(2)

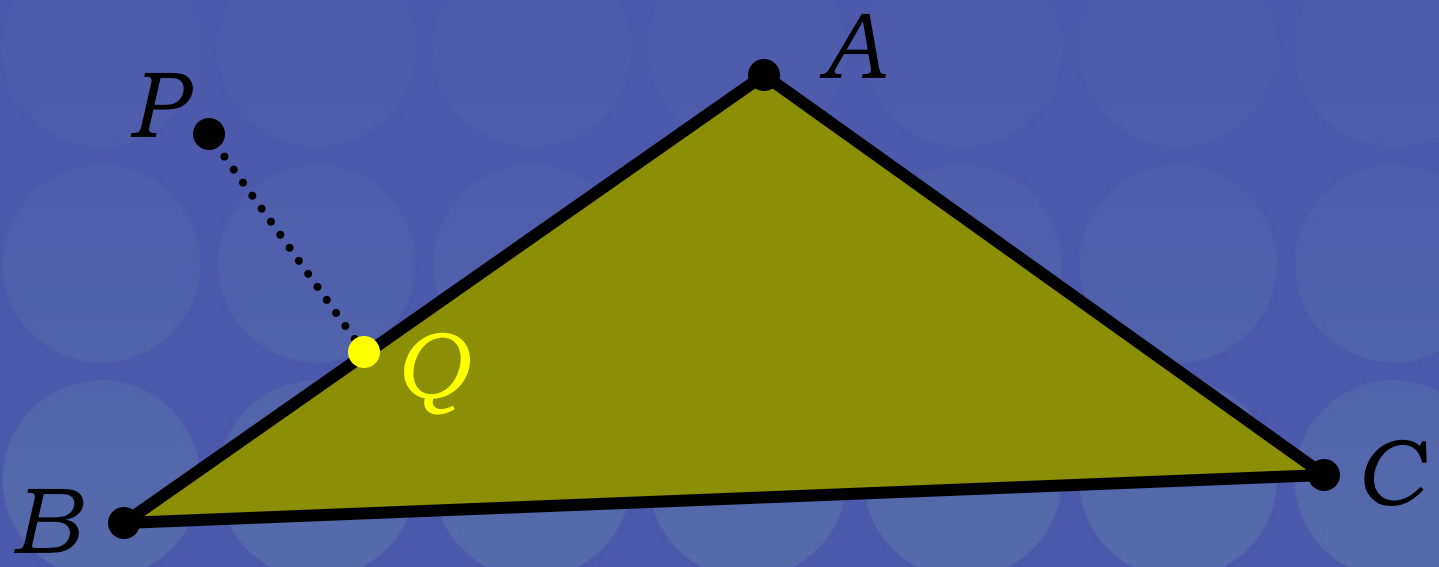
- Approach #1: Solve algebraically
  - Used in original GJK paper
  - Johnson's distance subalgorithm
    - Searches all simplex subsets
    - Solves system of linear equations for each subset
    - Recursive formulation
    - From era when math operations were expensive
    - Robustness problems
  - See e.g. Gino van den Bergen's book

Distance subalgorithm 2(2)

- Approach #2: Solve geometrically
  - Mathematically equivalent
    - But more intuitive
    - Therefore easier to make robust
  - Use straightforward primitives:
    - `ClosestPointOnEdgeToPoint()`
    - `ClosestPointOnTriangleToPoint()`
    - `ClosestPointOnTetrahedronToPoint()`
  - Second function outlined here
    - The approach generalizes

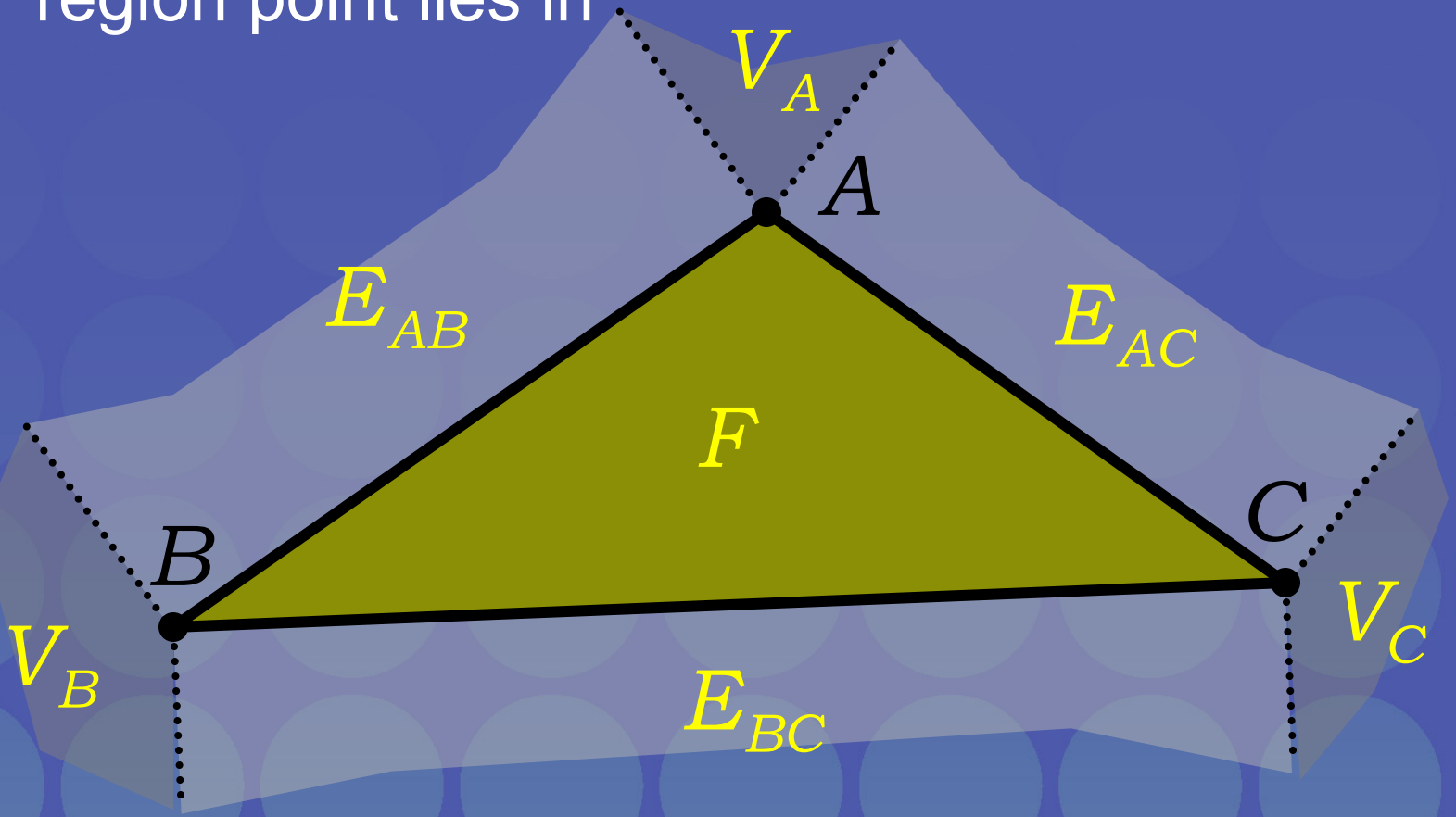
Closest point on triangle

- ClosestPointOnTriangleToPoint()
  - Finds point on triangle closest to a given point

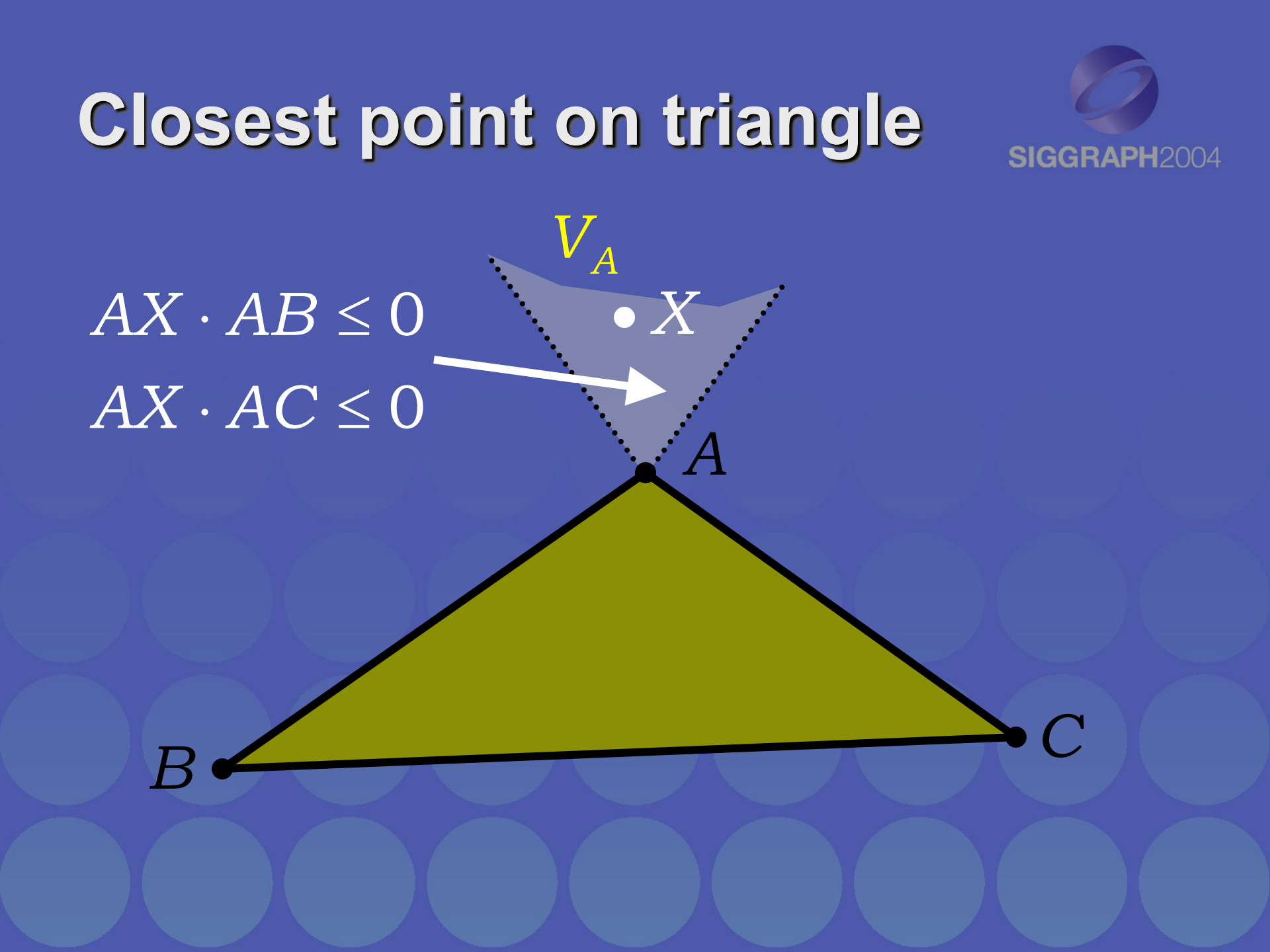


Closest point on triangle

- Separate cases based on which feature Voronoi region point lies in



Closest point on triangle



Closest point on triangle