Collisions:

Separating collisions

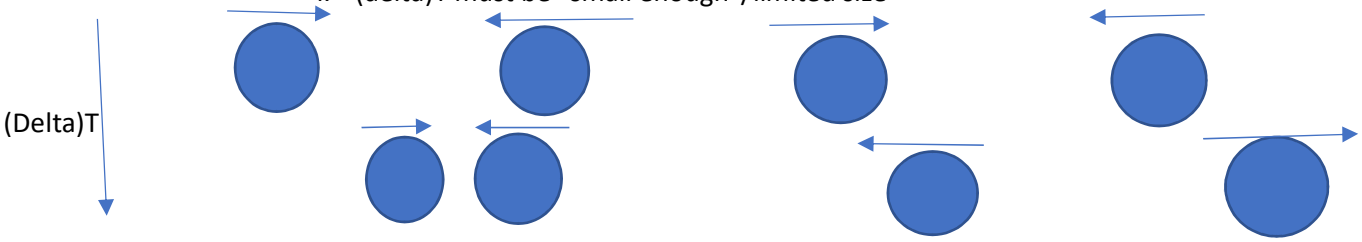
Non-separating collisions (contact problem)

Separating collisions:

1. collision detection
2. collision resolution

Collision Detection:

1. Time:
    a. Discrete time : 0, (delta)T, 2(delta)T, 3(delta)T, …
        i. (delta)T must be “small enough”/limited size



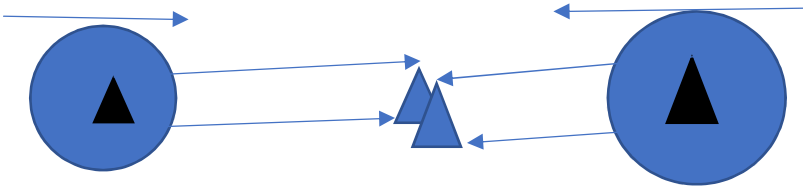
(delta)x -> min edge length of the bounding box over all objects

|(delta)V| -> max velocity over all objects

(delta)T < (alpha) ((delta)x)*(.1))/|(delta)V| -> time step restriction

(alpha) -> CFL constant: Courant-Freidrids-Levy constant 0 < (alpha) < 1

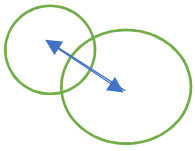
    b. continuous time



Cubic equation ^

Expensive to detect all of the cubic eqs for each triangle

Instead, use ***SPHERE HIERARCHIES***

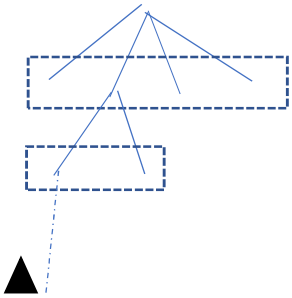


Utilize sphere bounding boxes ad calculate radius and distance with those

Measure how good collision detection with cloth sims

Collision detection:

1. Object representation
    a. mesh representation (verts and tris)



Bounding volume hierarchies(BVH)

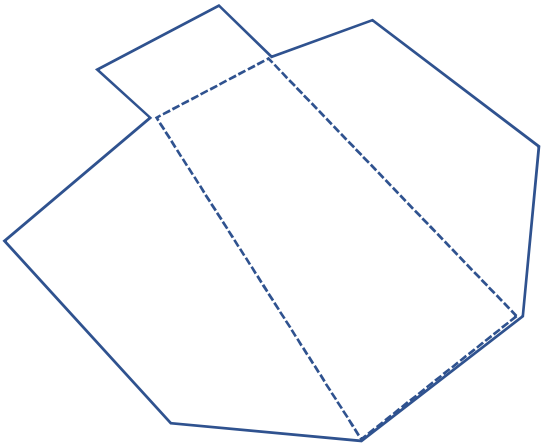
Check box-box, going down the hierarchy until you hit the tri-tri intersections.

Box-box are robust. tri-tri are complex. Use (Jon Shewchuk robust predicates) (usually first result. if not, look for one with 690 citations)

This detection can take a while.

To optimize, use

Convex Decomposition:

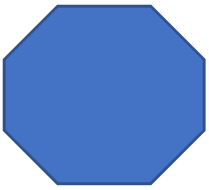


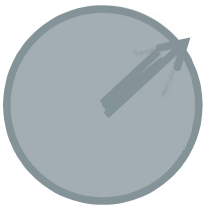
Break concave pieces into convex pieces. No efficient way to do this precisely is known.

Approximate is what is used.

Implicit Surfaces:

Explicit -> individual points/triangles on the surface





Isocontour => same height (topography maps)

(phi): R^n -> R

(phi)(x, y) = x^2 + y^2

(phi)(x, y) = 0

(phi)(x, y) = c

c – isocontour

Surface:

o-isocontour of (phi)