This sounds hard!

Does it have to be so complicated?

so let's talk about

Simplified collision detection

Sometimes it doesn't have to be hard at all!

Approximative methods
Collision shapes

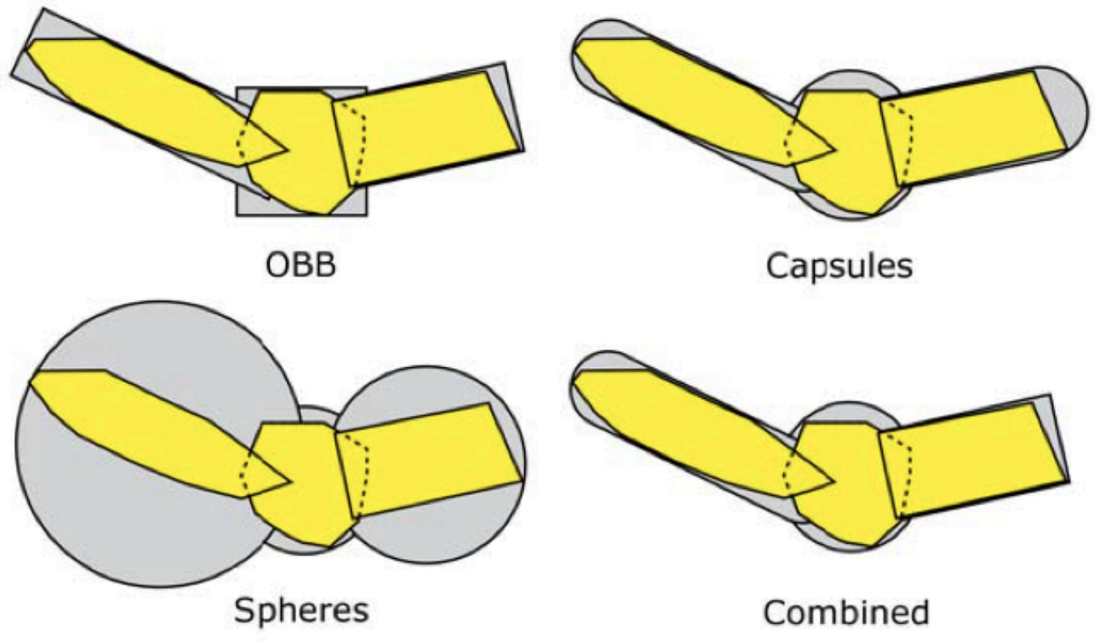
Simplify the narrow phase with simplified versions of the geometry

- Low-polygon version of a polyhedra
- A "standard" shape that follows the shape as closely as possible

The "narrow phase" more or less blends into the "broad phase"

Multiple simple shapes

Use a number of very simple primitive shapes to approximate a complex shape



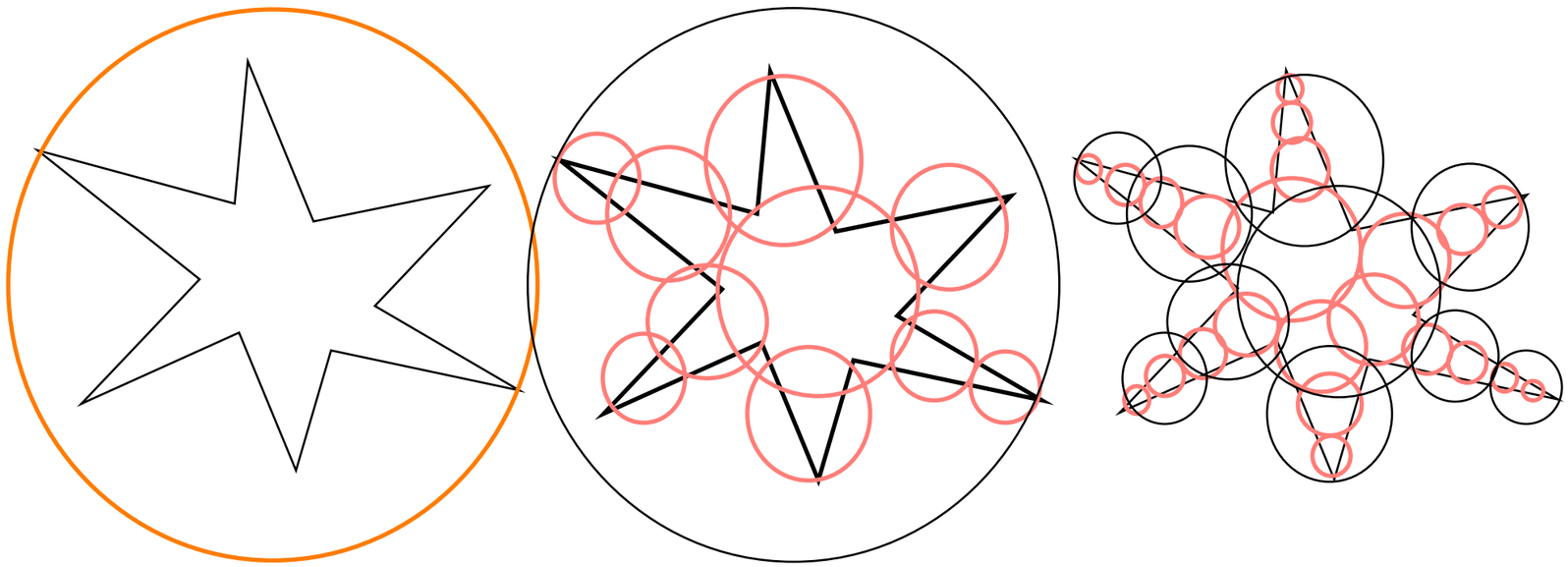
From:
Henrik Bäcklund,
Niklas Neijman,
"AUTOMATIC MESH DECOMPOSITION FOR REAL-TIME COLLISION DETECTION",
2013

Decomposing to spheres

Sphere-sphere collisions only

Very simple tests!

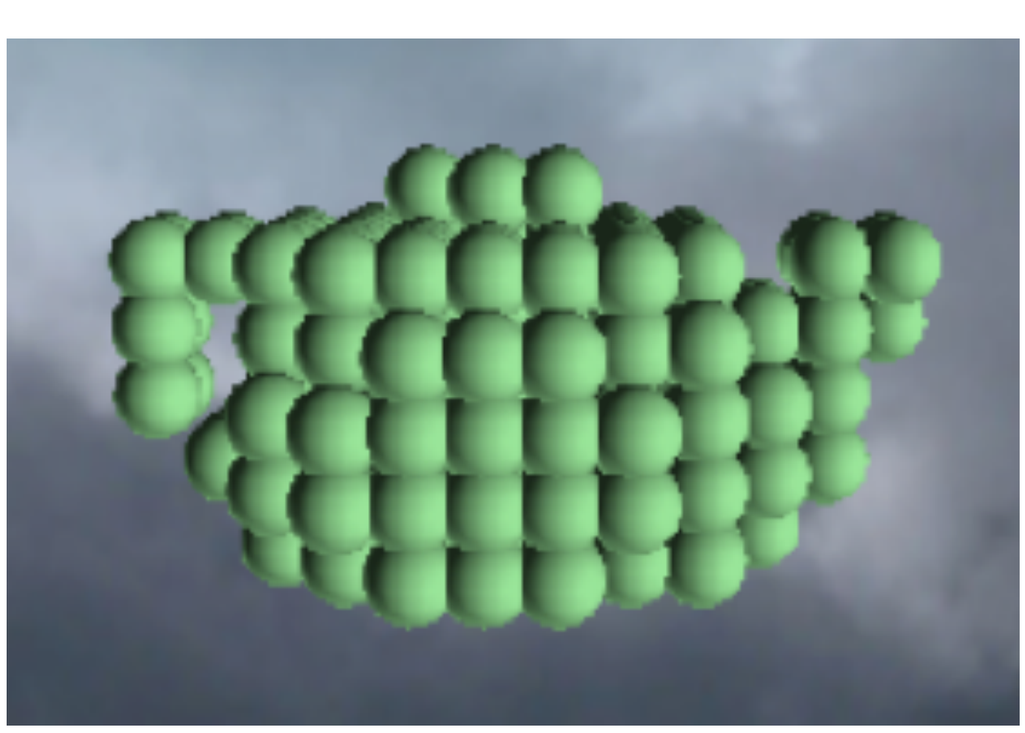
Object hierarchies



Collision detection is done to the level that available time permits

Voxelization

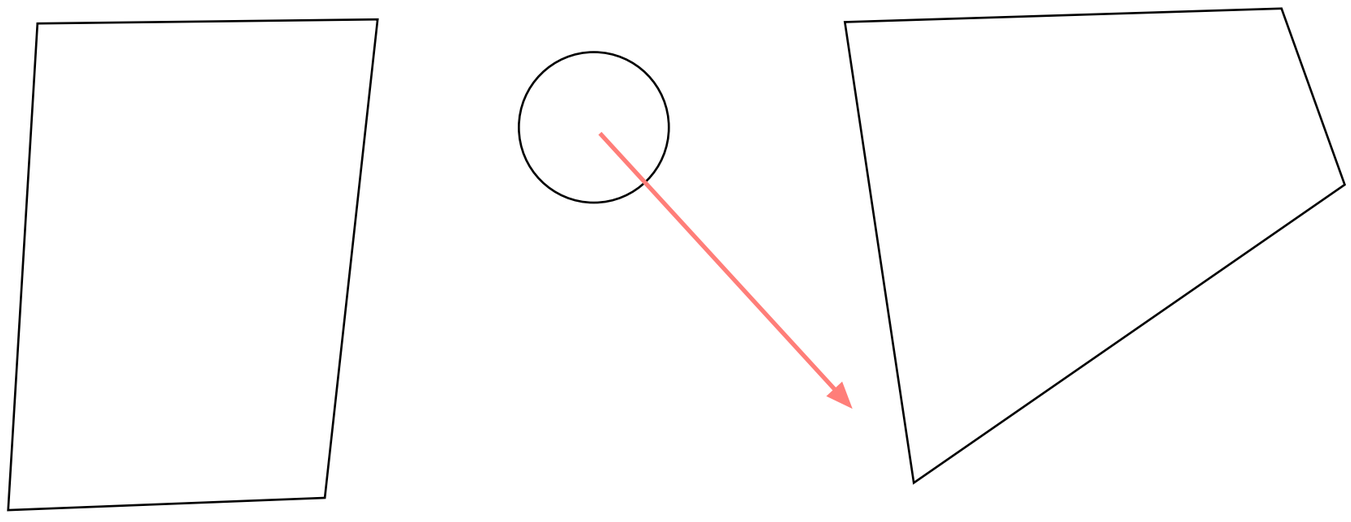
Break down model to voxels, test with one sphere per one voxel



Teapot voxelized to 171 spheres

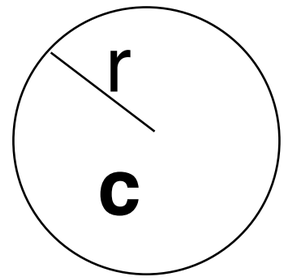
From Edhammer, "Rigid Body Physics for Synthetic Data Generation", 2016

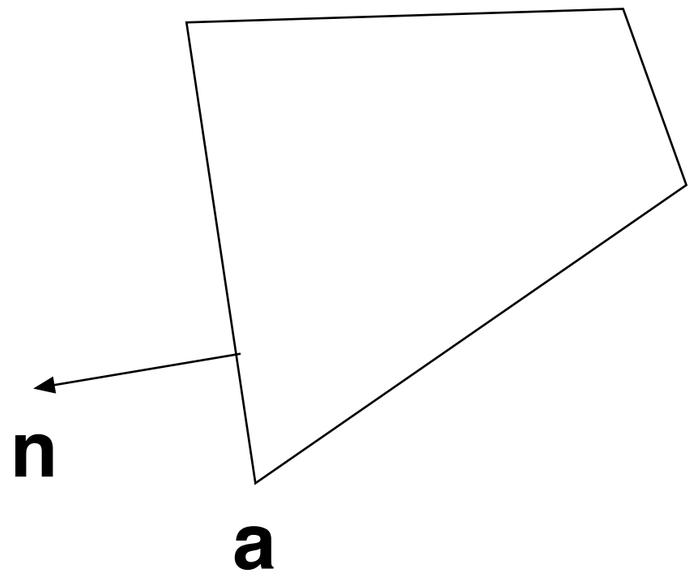
Spheres in polyhedra world: cameras as well as objects



The size gives us a minimum distance to walls!
(E.g. more than the near Z clipping distance.)

Sphere - polyhedra distance





$\mathbf{n} \cdot (\mathbf{c} + r \cdot \mathbf{n}) > \mathbf{n} \cdot \mathbf{a}$

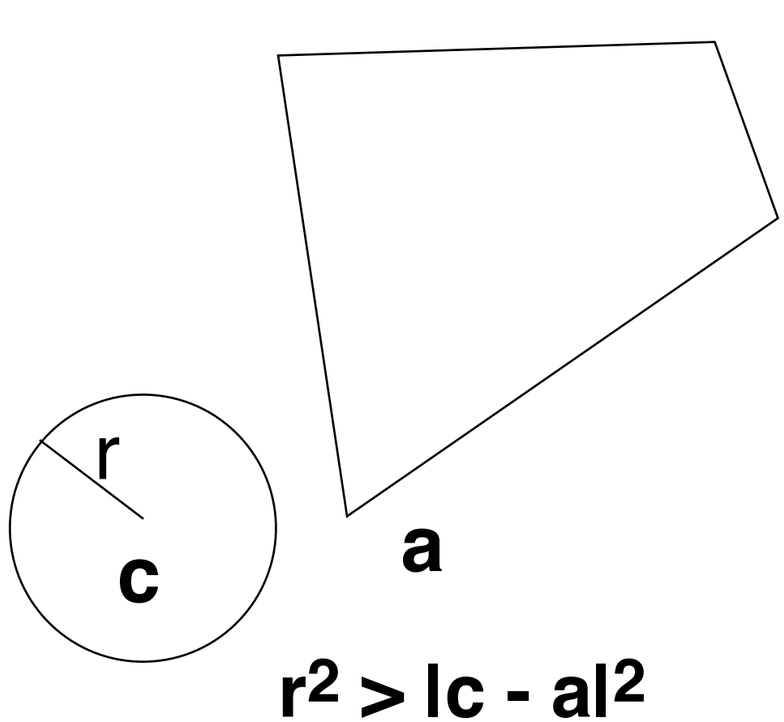
=>

Distance from center to plane > r

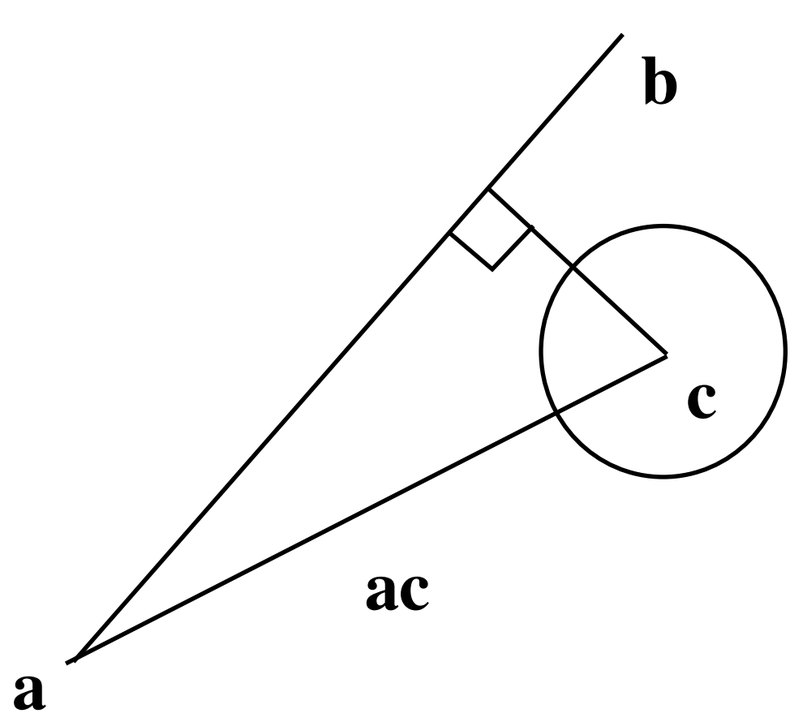
$r > \mathbf{n} \cdot (\mathbf{a} - \mathbf{c})$

Valid when a line through **c** along **n** intersects the polygon!

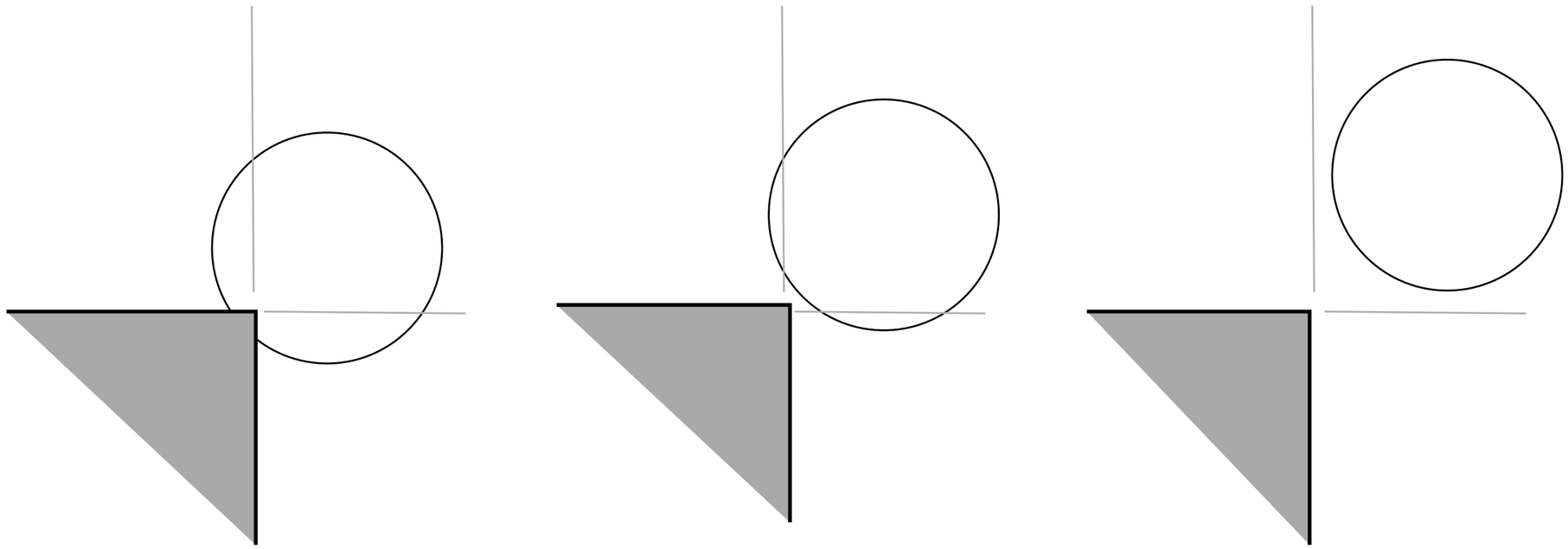
Exact test: Check corners and edges too



$r^2 > |c - a|^2$



Simplification: Test all planes only! If outside *any* plane - outside, otherwise it intersects.

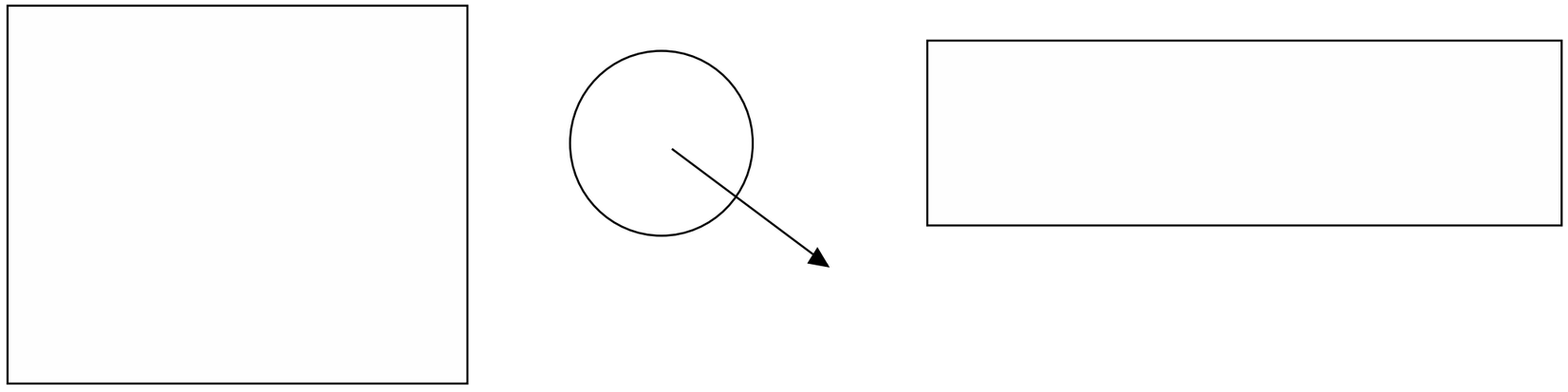


Correct hit **False hit** **Correct miss**

Even simpler case: Axis aligned 3D maze

All walls are AABBs All objects are spheres/cylinders

All tests are AABB tests



Final notes on the simplified camera collisions:

Resolving: Pick the closest intersected plane as the one to ”hit”, and use that for collision handing. The smallest change is usually correct.

Conclusion: Don’t overdo it if you can fake it. Be ambitious, but don’t waste time on effects that nobody will notice.

Global phase collision detection

Avoid many-to-many checks

Space subdivision

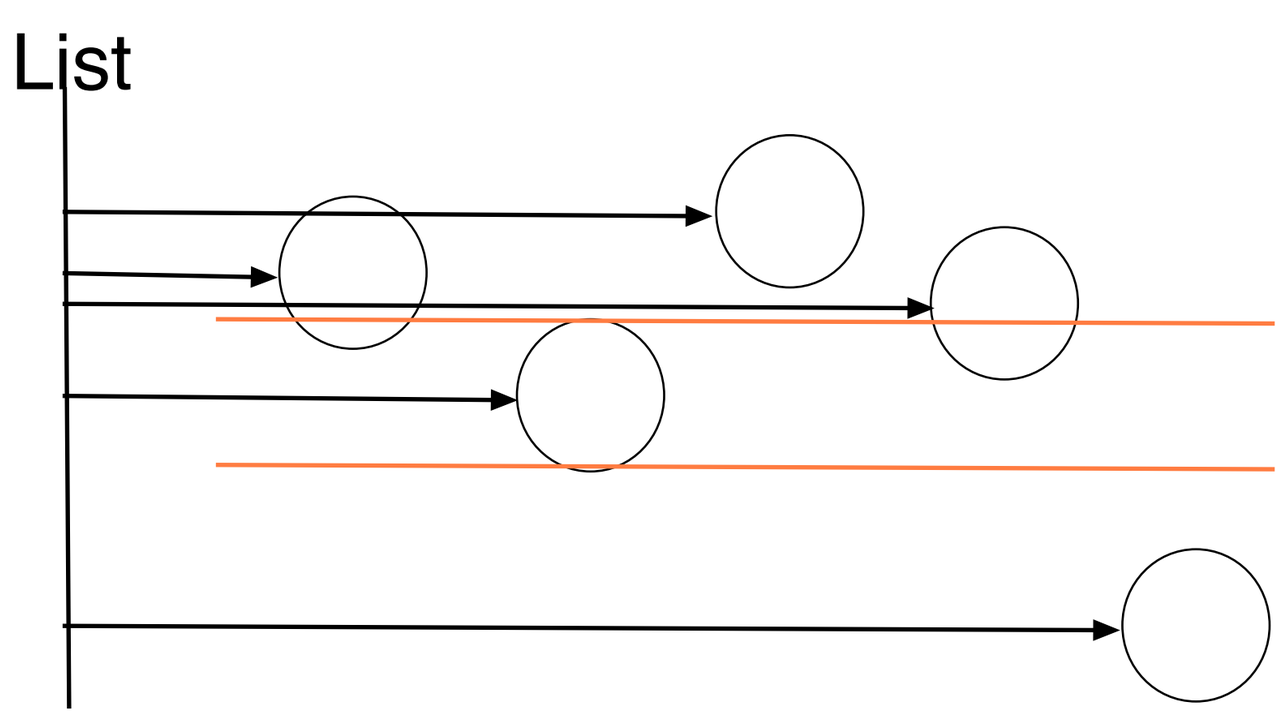
Simple: Linear sorting

More elaborate: Hierarchies, octrees…

Simplify objects out of view?

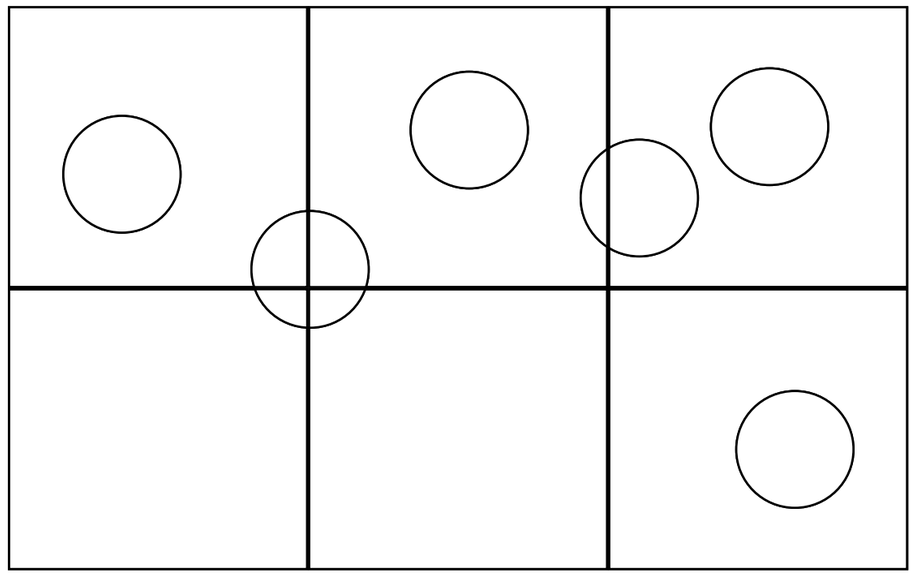
Linear sorting

Reduces the problem by 1 dimension



Subdivision for collision detection

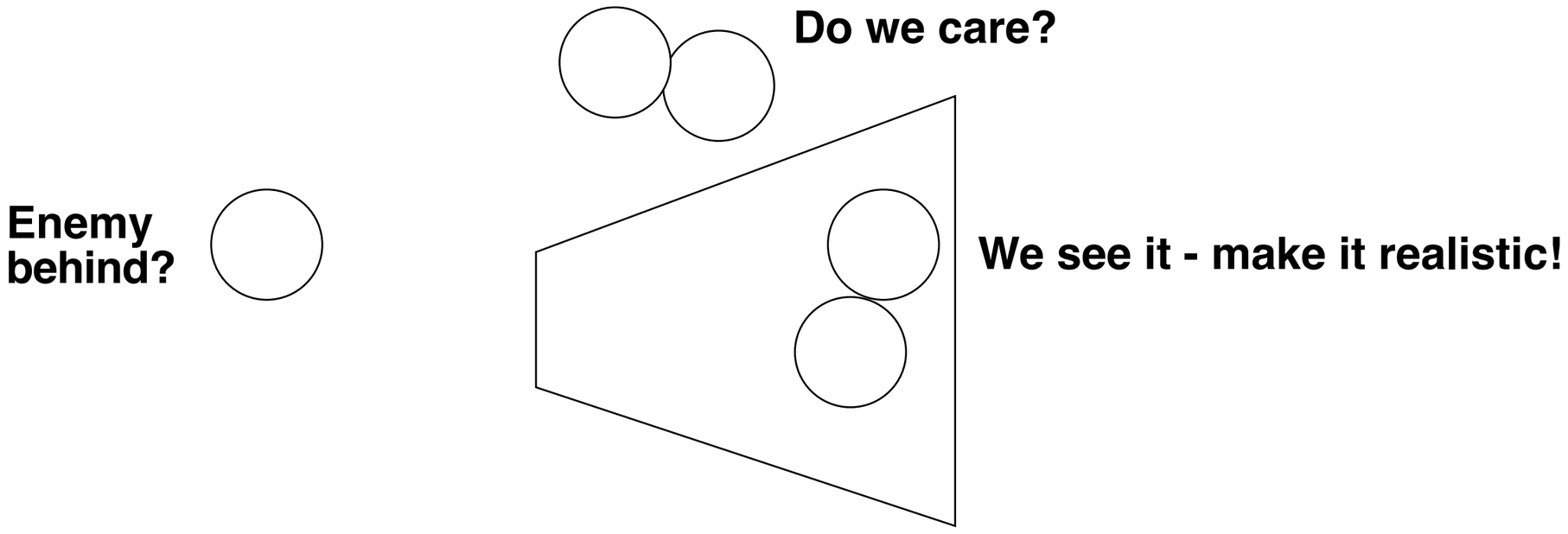
Subdivision by BSP trees, octrees, quad-trees
Generally useful for most large world problems



Frustum culling and collision detection

If we can't see it, do we care?

Frustum culling is already applied for drawing. Use it also for simplifying collision detection.



Collision detection and portals

Again, we can take advantage of visibility to reduce workload.

Portals lead to problems, objects near portals in cells and portals systems need to be present in both cells.

Process collission detection and AI in visible cells only, or visible and nearby.

Collision handling

What to do once a collision is found.

- **Separate**
- **Change velocities**
- **Deform**
- **Maintain constraints**

Full (narrow-phase) tests are hard to resolve.

Simple particle physics (again)

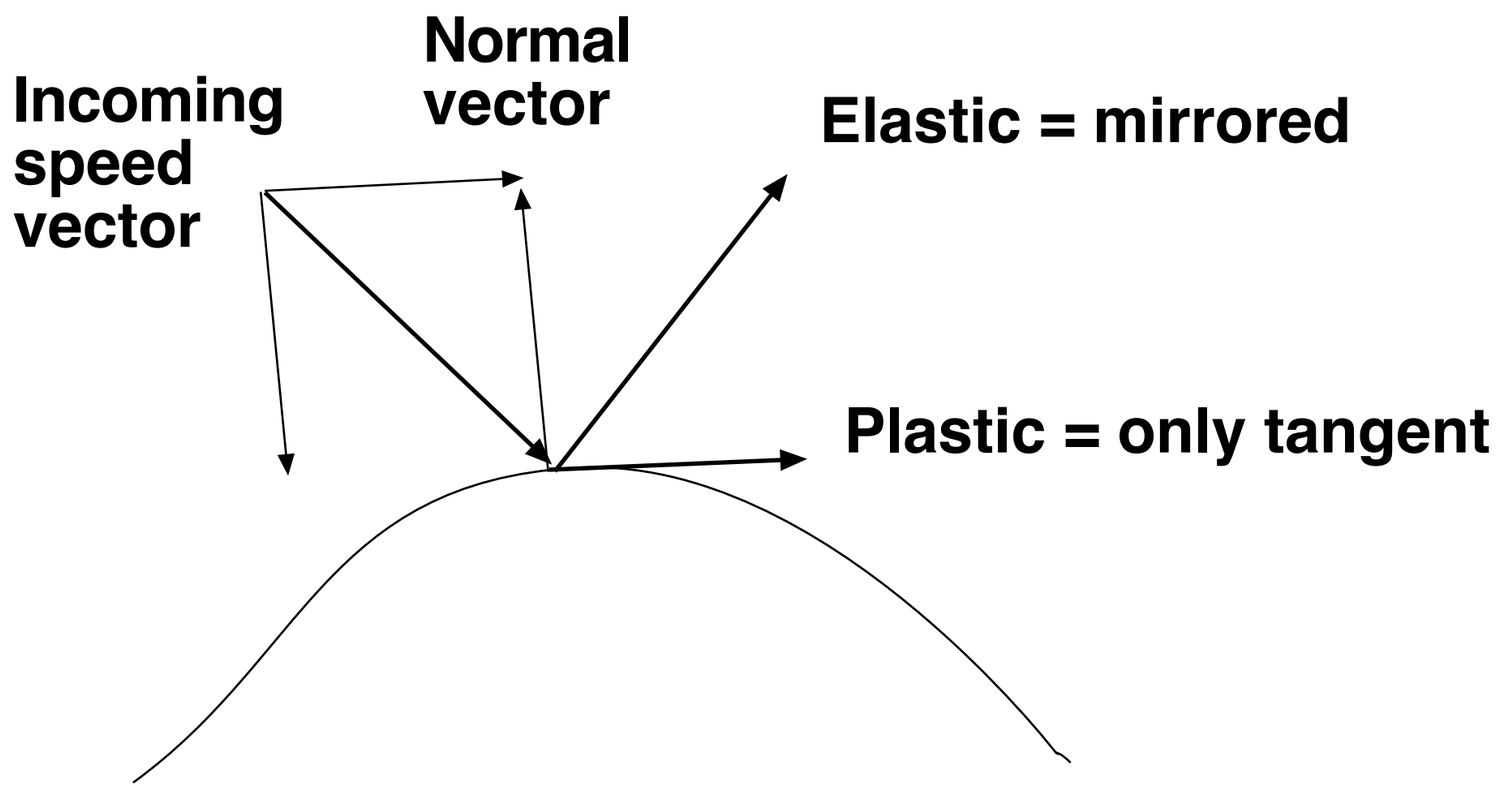
acceleration = gravity + forces/mass
speed = speed + acceleration
position = position + speed

$$a = g + \sum f/m$$
$$s = s + a$$
$$p = p + s$$

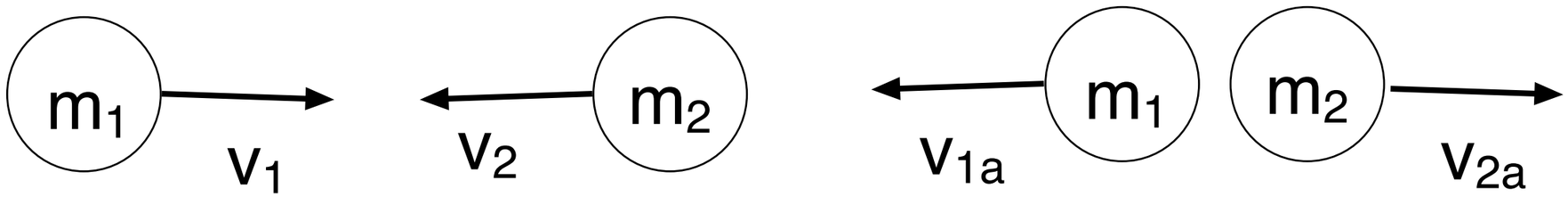
(“Euler integration”)

Modify speed and position in collisions

Simple particle-surface collision



Plastic and elastic collisions



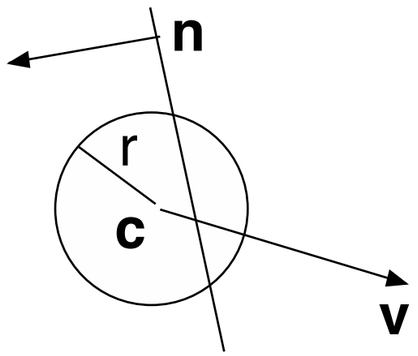
Preserve momentum

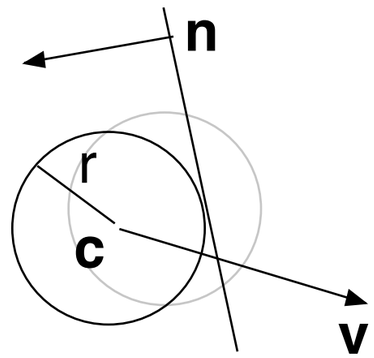
$$m_1v_1 + m_2v_2 = m_1v_{1a} + m_2v_{2a}$$

Elastic collisions also preserve kinetic energy

$$m_1v_1^2 + m_2v_2^2 = m_1v_{1a}^2 + m_2v_{2a}^2$$

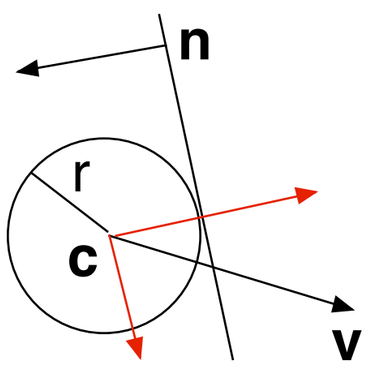
Collision handling sphere-polyhedra



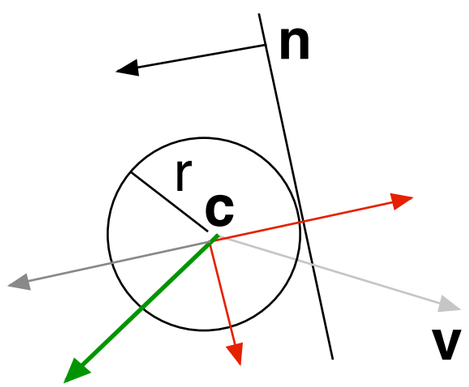


Assuming stationary polyhedra

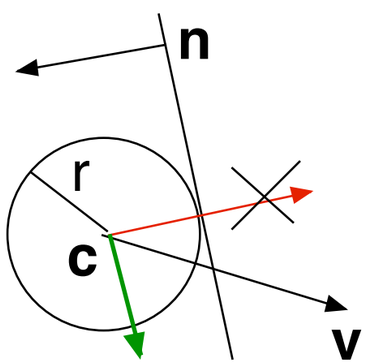
Separate - move object away along normal vector



Split velocity along n

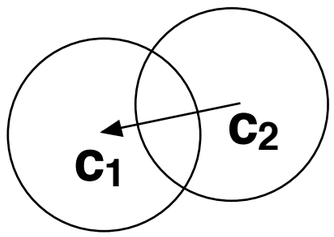


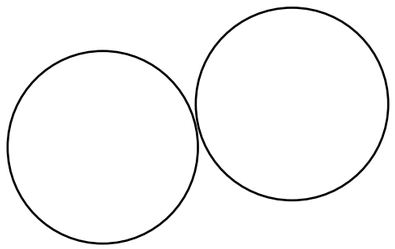
Elastic



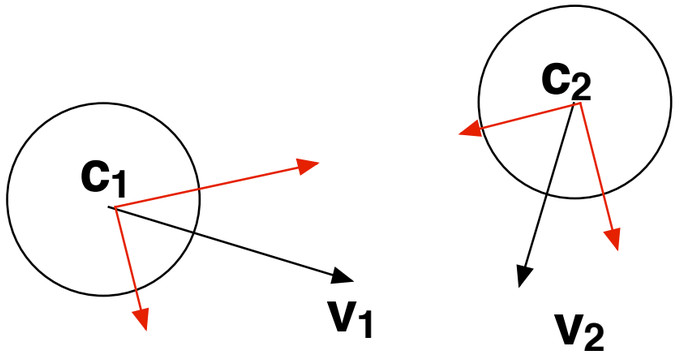
Plastic

Collision handling sphere-sphere (simplified, point masses)

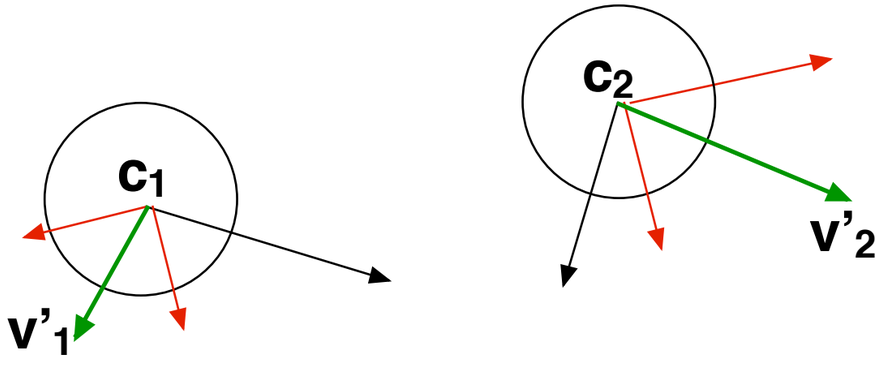




Separate - move object away along vector through centers



Split velocities along vector though centers (c_2-c_1)



Elastic: Exchange components along c_2-c_1

Beyond point-mass mechanics

Rigid body mechanics
Better integration
Stacking
Applying forces and backing time to avoid overlap
Deformable bodies
Breakable bodies

Conclusions, collision detection

Must focus on convex shapes

Simple collision detection with AABB or spheres. Spheres vs polyhedra also easy. Polyhedra vs polyhedra hard to do.

Global phase also important

Narrow phase expensive and complex