



# **Global phase collision detection**

**Avoid many-to-many checks**

**Space subdivision**

**Simple: Linear sorting**

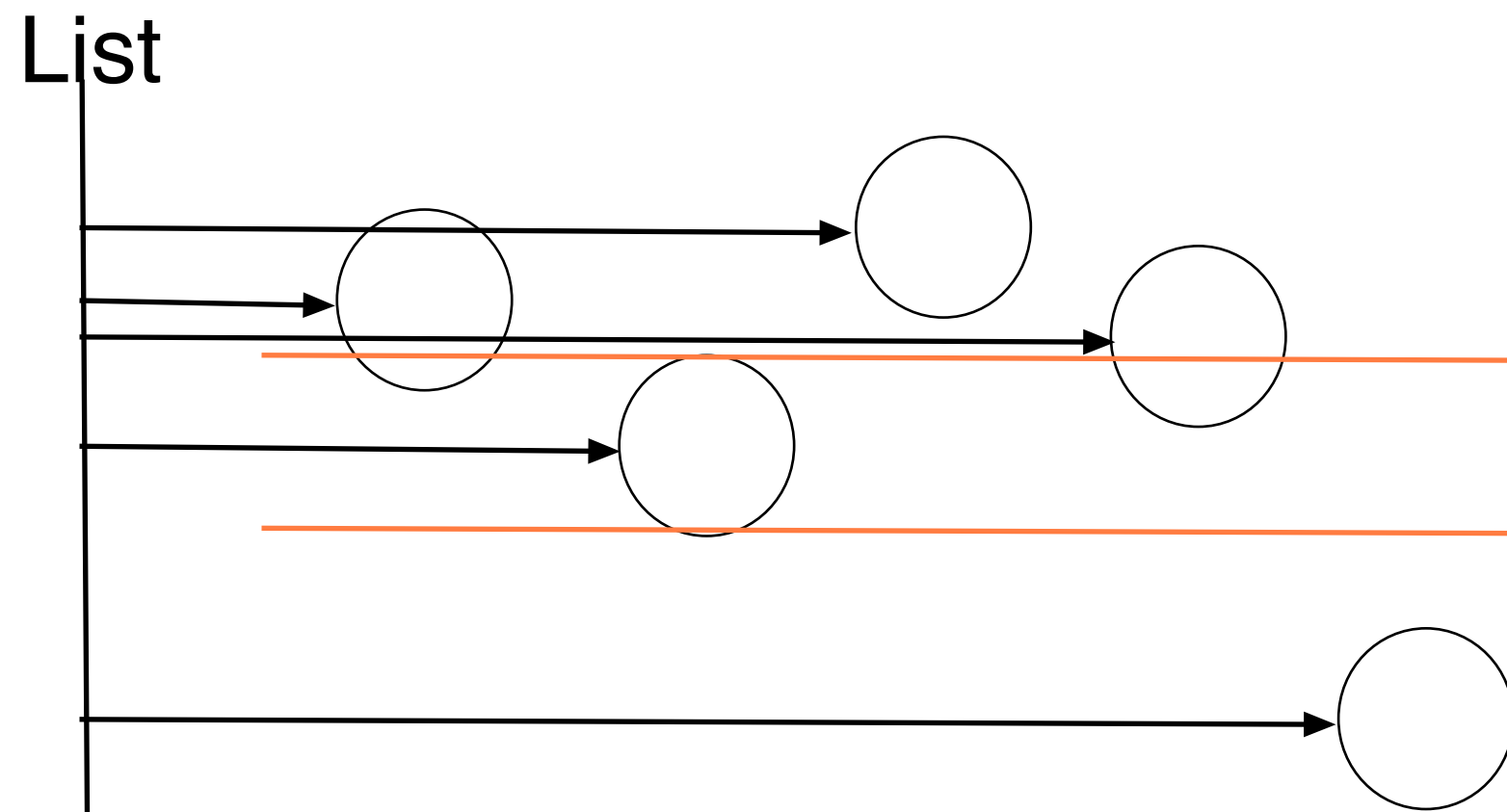
**More elaborate: Hierarchies, octrees...**

**Simplify objects out of view?**



# Linear sorting

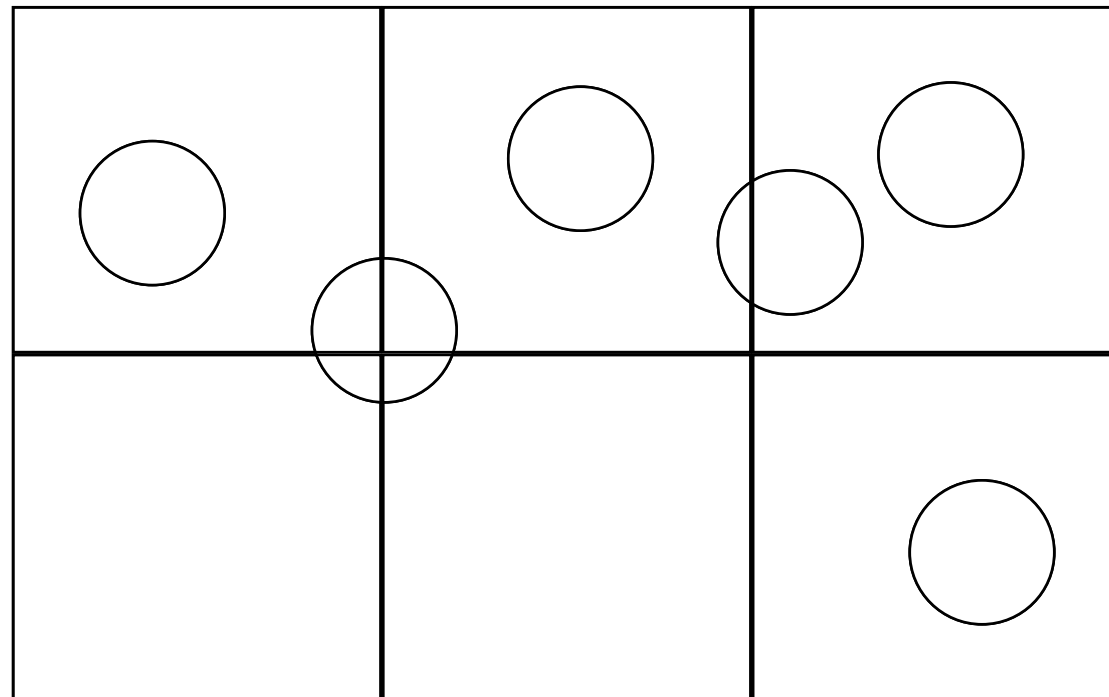
**Reduces the problem by 1 dimension**





# Subdivision for collision detection

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

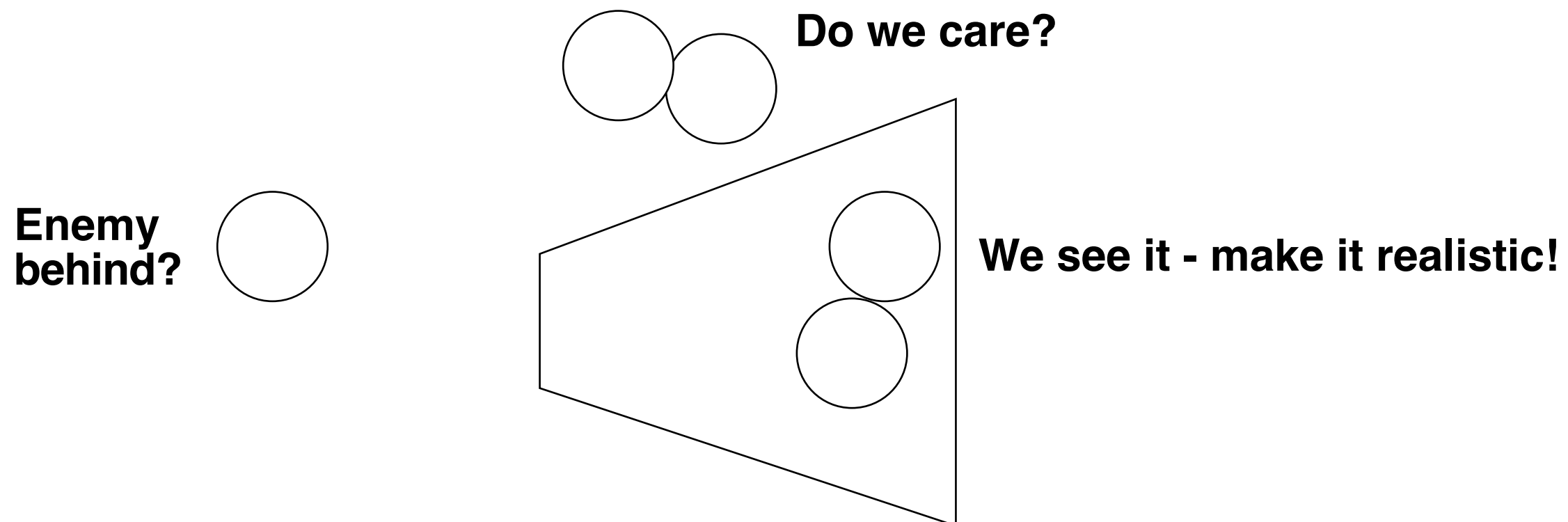




## Frustum culling and collision detection

**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 collision detection and AI in visible cells only, or visible and nearby.**



**This sounds hard!**

**Does it have to be so complicated?**

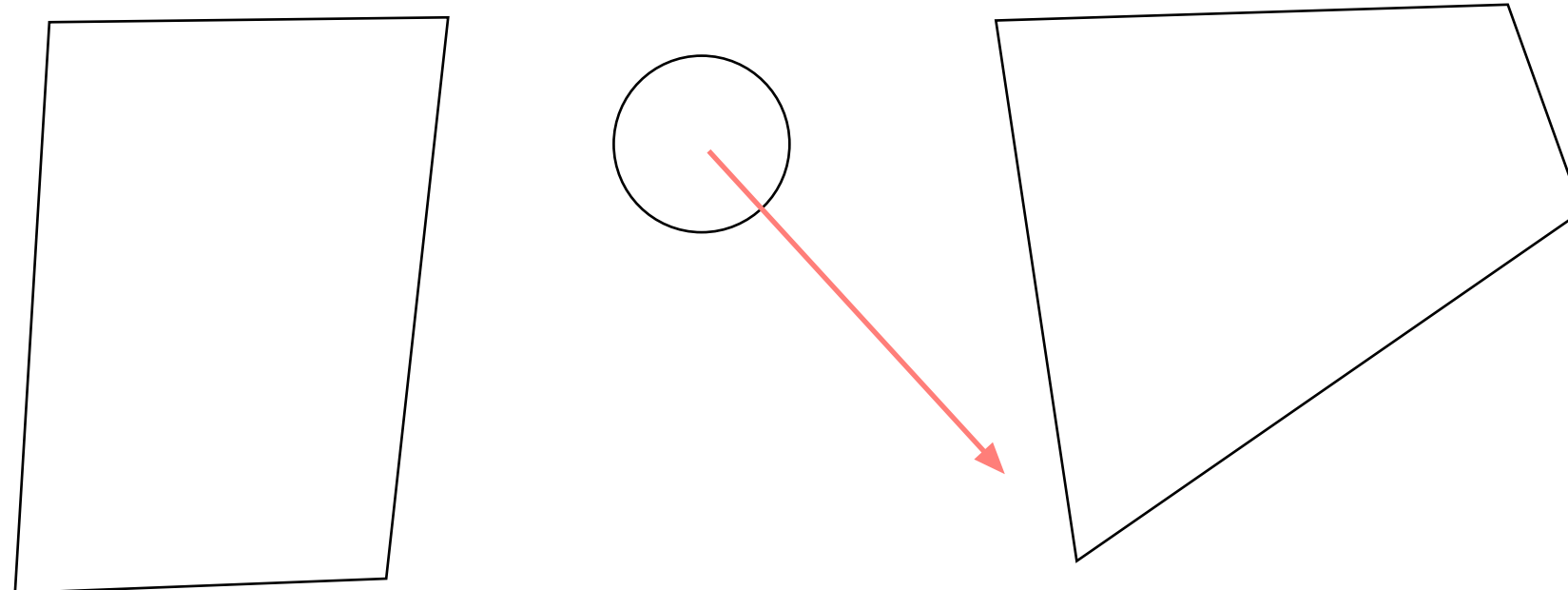
**so let's talk about**

**Simplified collision detection**

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



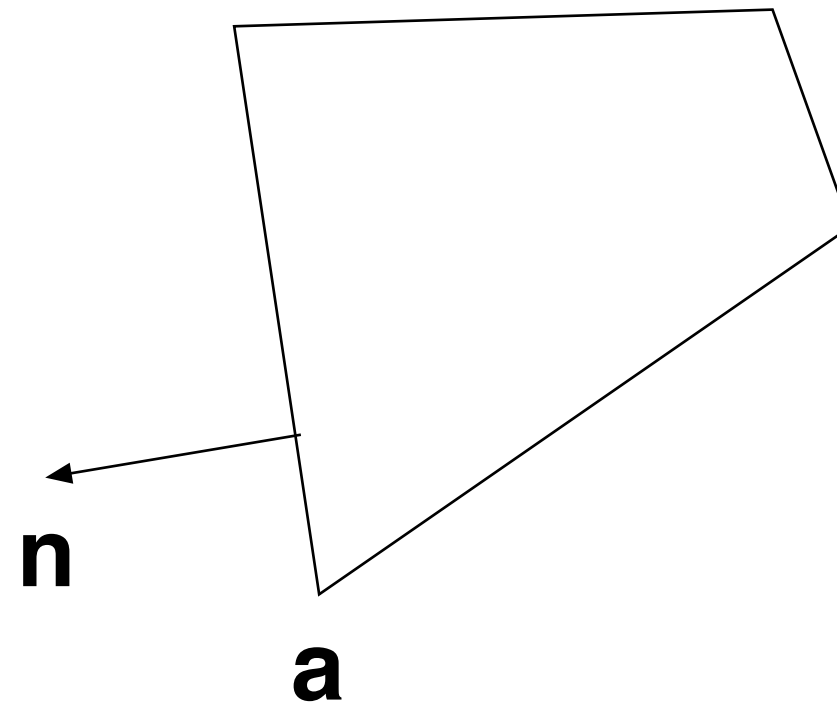
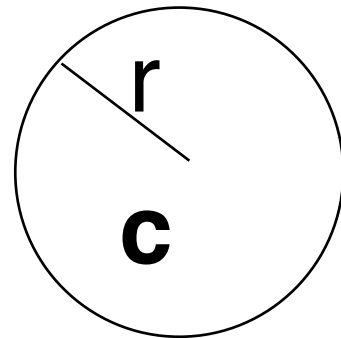
## Spheres in polyhedra world: cameras as well as objects



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



## Sphere - polyhedra distance



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

$\Rightarrow$

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

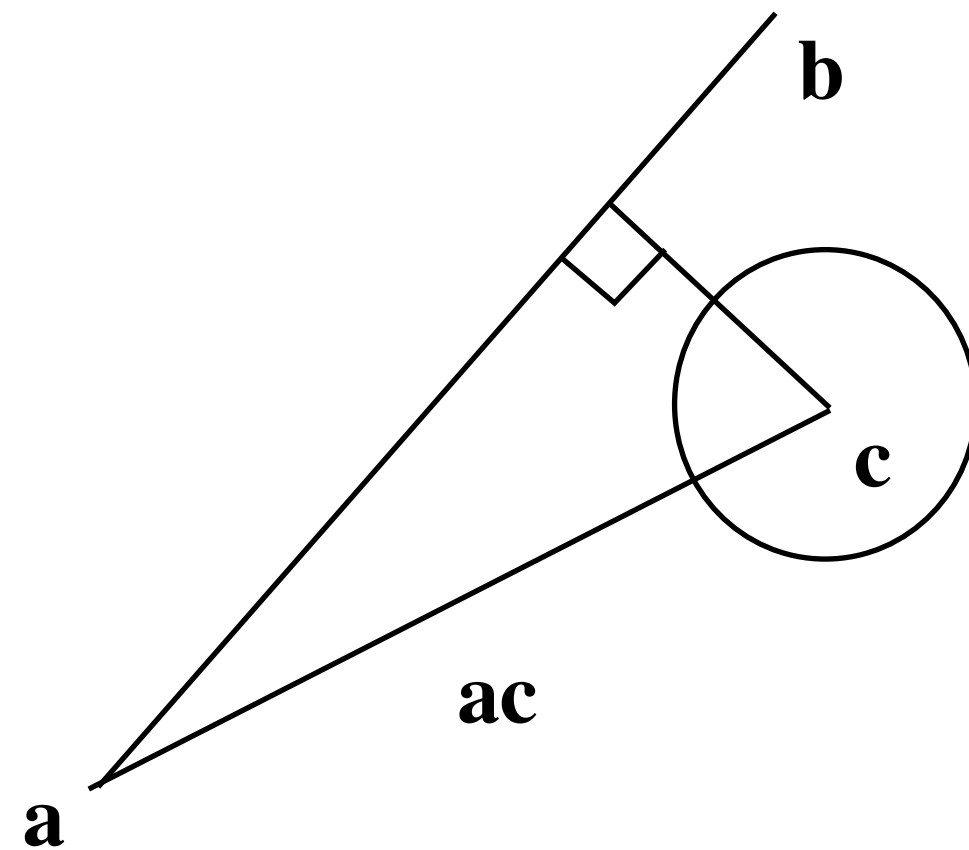
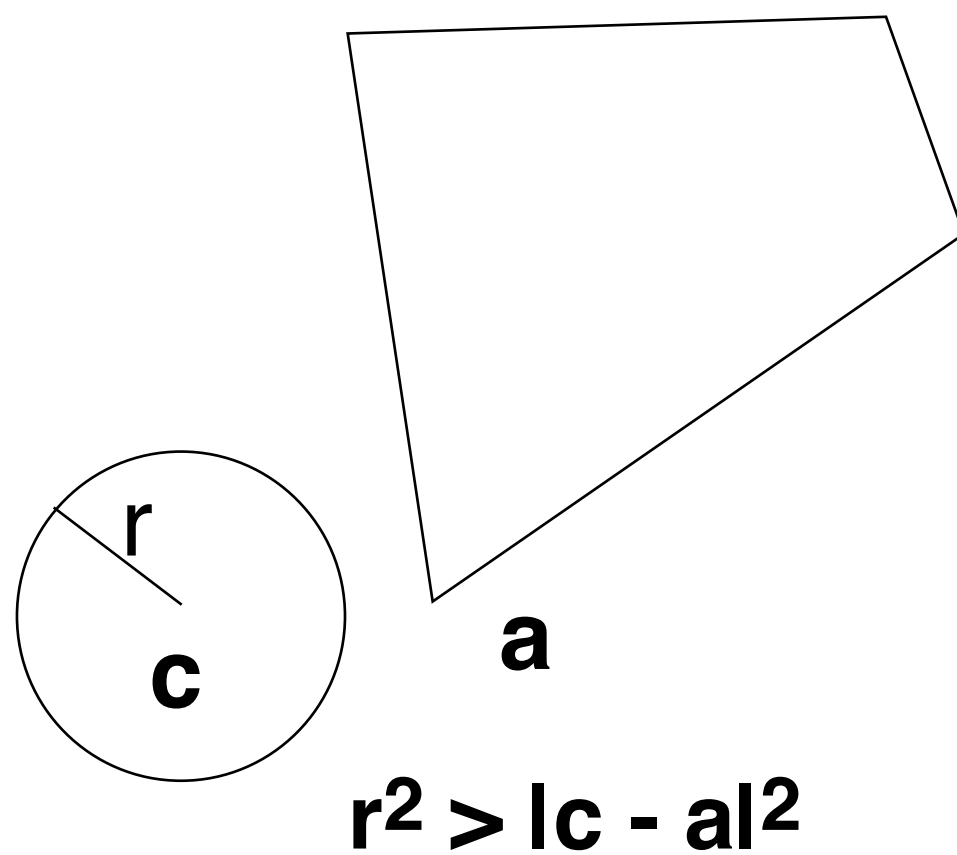
Distance from center to plane  $> r$

Valid when a line through  $\mathbf{c}$  along  $\mathbf{n}$  intersects the polygon!



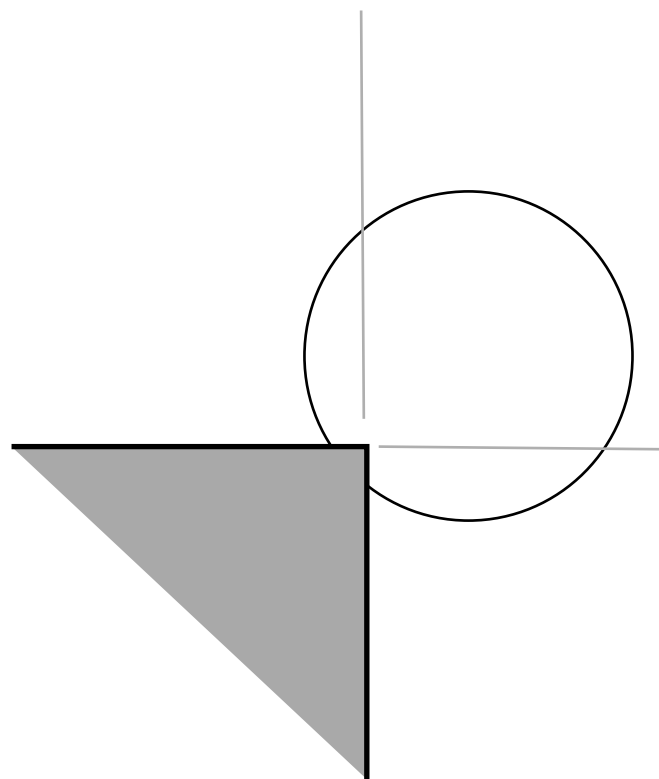


## Exact test: Check corners and edges too

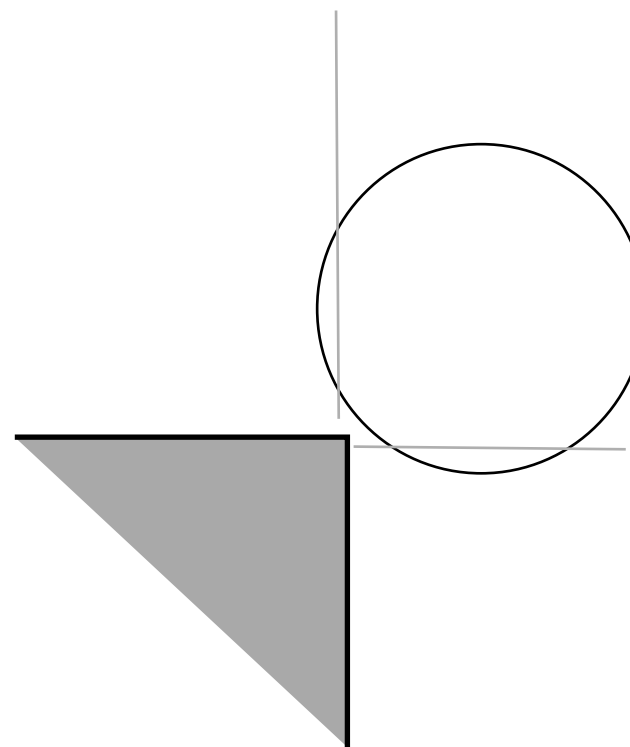




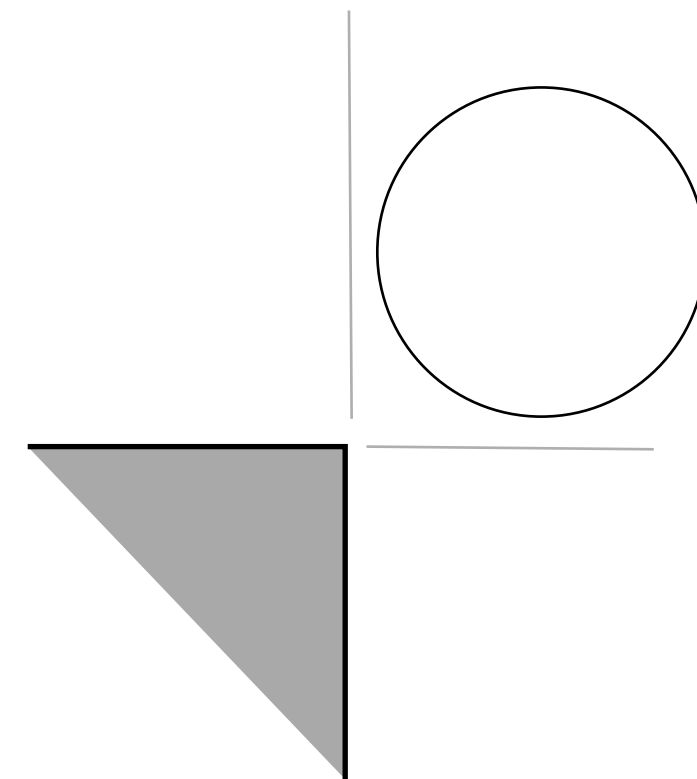
**Simplification: Test all planes only! If outside *any* plane - outside, otherwise it intersects.**



**Correct hit**



**False hit**



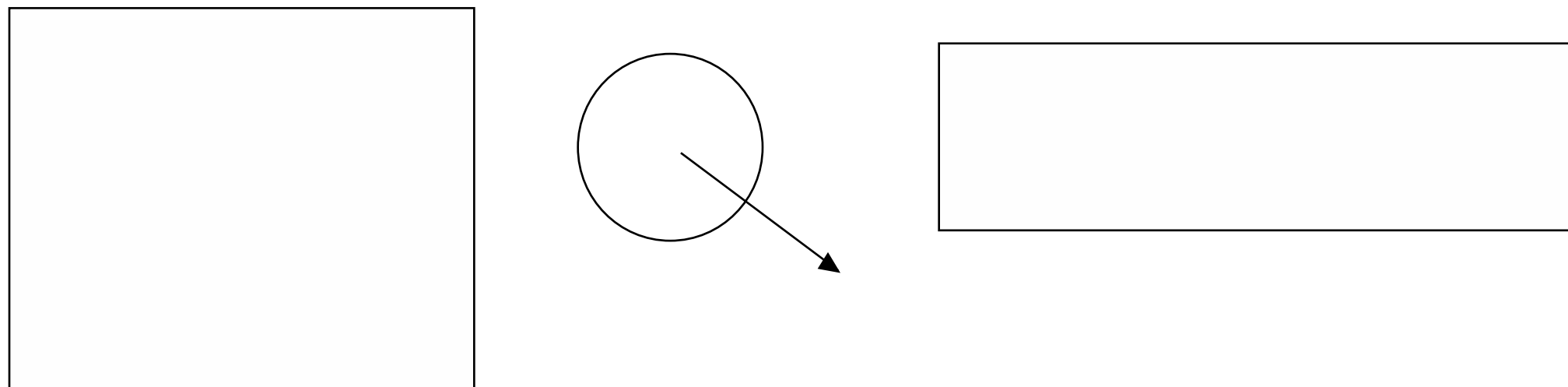
**Correct miss**



# Collision detection & handling

**Simple case: Axis aligned 3D maze**

**All walls are AABBs All objects are spheres/cylinders**





## **Final notes on the simplified camera collisions:**

**Resolving: Pick the closest intersected plane as the one to "hit", and use that for collision handling. 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.**



# 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**

$$\mathbf{a} = \mathbf{g} + \sum \mathbf{f}/m$$

$$\mathbf{s} = \mathbf{s} + \mathbf{a}$$

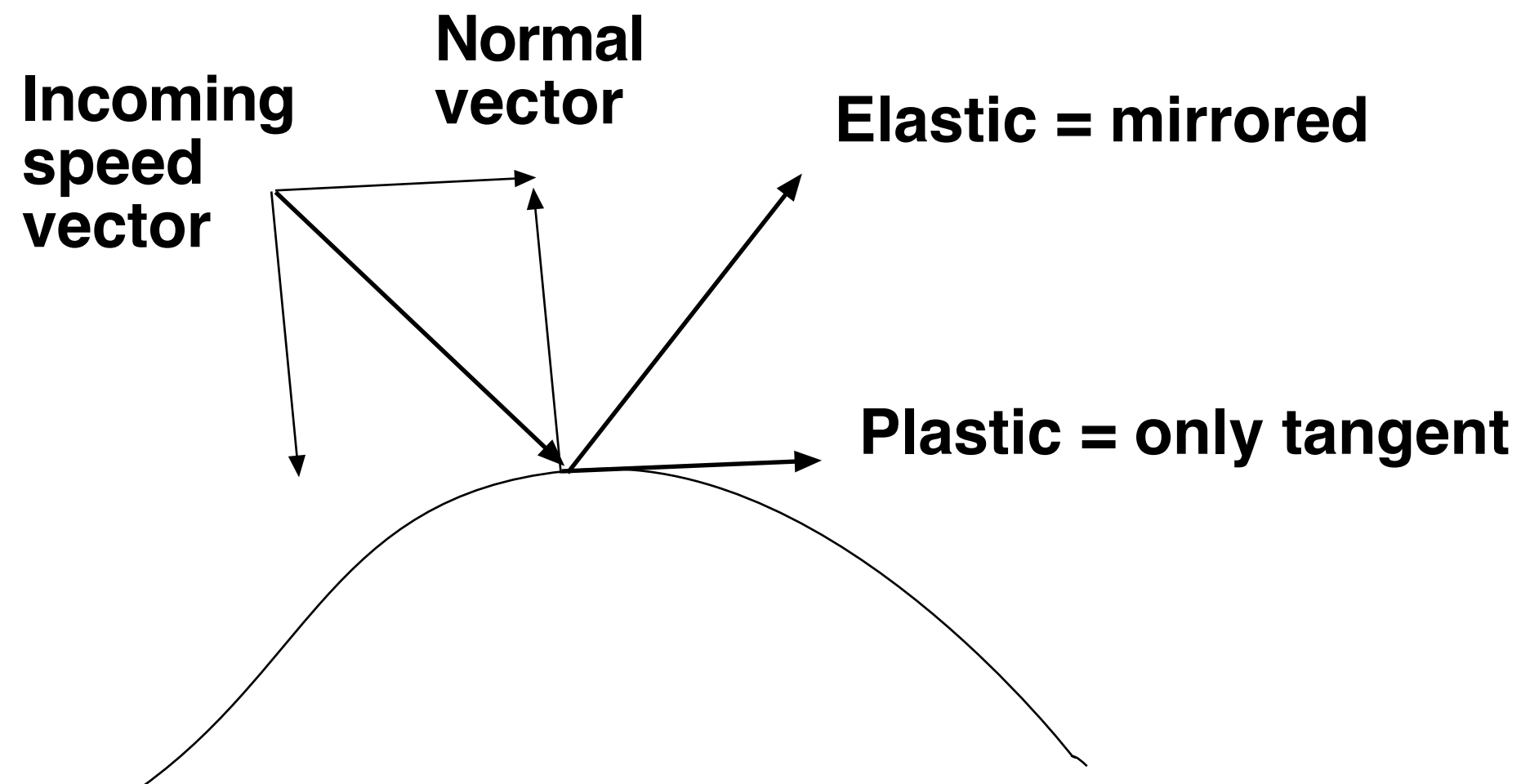
$$\mathbf{p} = \mathbf{p} + \mathbf{s}$$

**(“Euler integration”)**

**Modify speed and position in collisions**

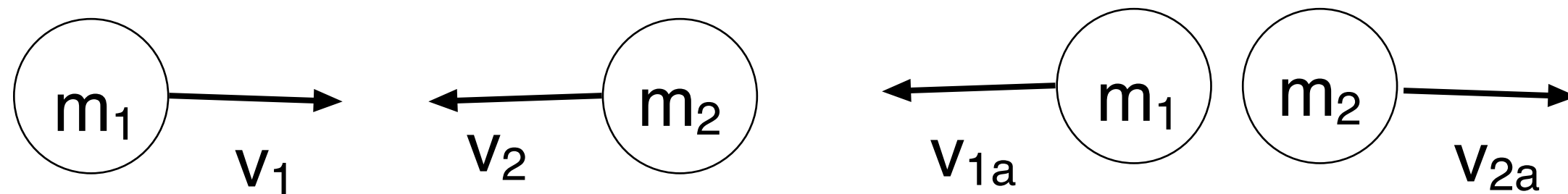


# Simple particle-surface collision





## Plastic and elastic collisions



**Preserve momentum**

$$m_1 v_1 + m_2 v_2 = m_1 v_{1a} + m_2 v_{2a}$$

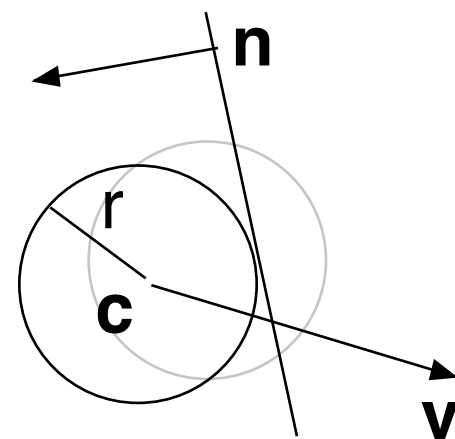
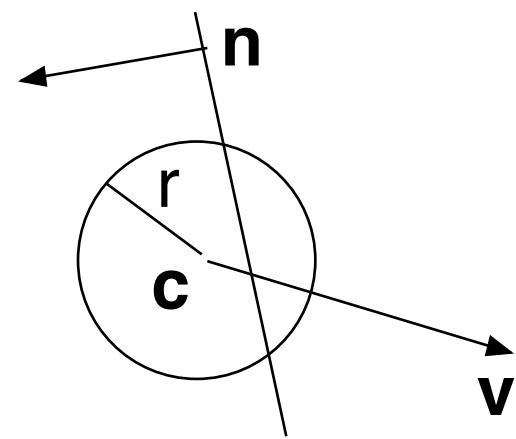
**Elastic collisions also preserve kinetic energy**

$$m_1 v_1^2 + m_2 v_2^2 = m_1 v_{1a}^2 + m_2 v_{2a}^2$$



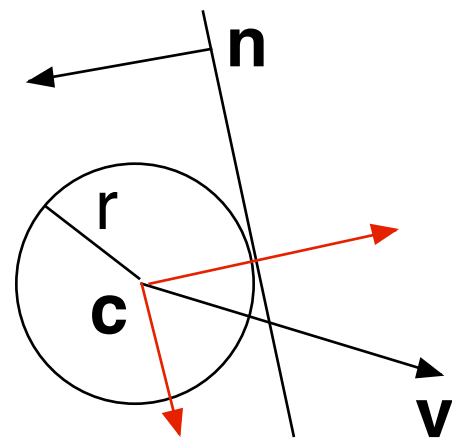


# Collision handling sphere-polyhedra

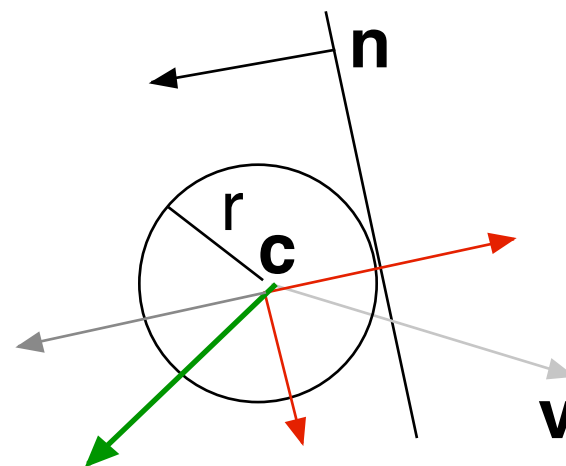


Assuming stationary polyhedra

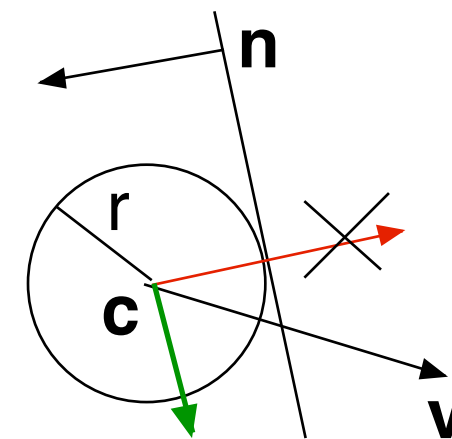
Separate - move object away along normal vector



Split velocity along  $n$



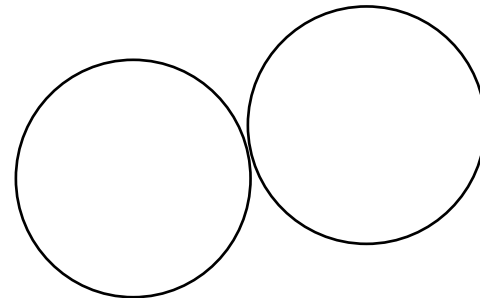
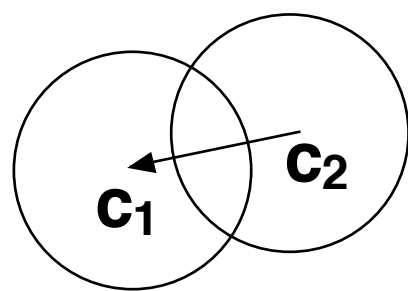
Elastic



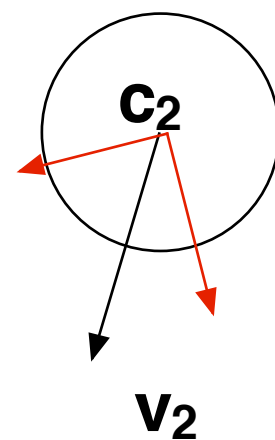
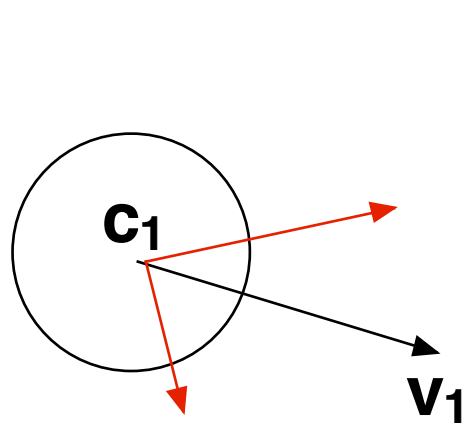
Plastic



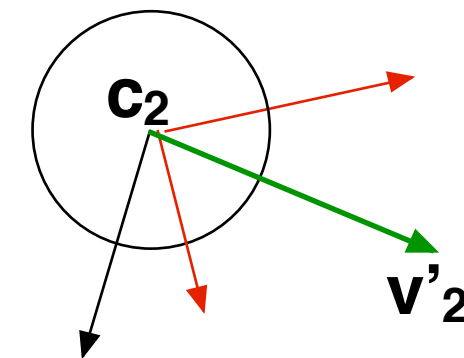
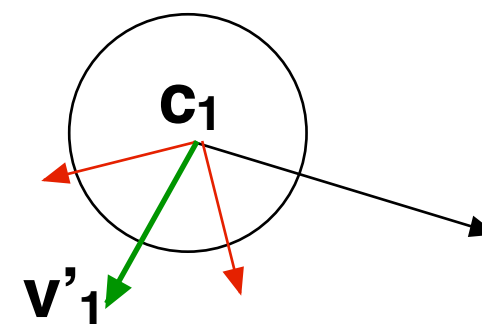
## Collision handling sphere-sphere (simplified, point masses)



Separate - move object away along vector through centers



Split velocities along vector through 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**

**Global phase also important**

**Narrow phase expensive and complex**