Lecture 11

Collision detecton Ch 14.8-

Creating natural shapes:

Fractals Noise Ch 9

Information Coding / Computer Graphics, ISY, LiTH

Collision detection and handling

A major problem in many real-time animations.

Even modern high-budget games have problems!

2D collision detection

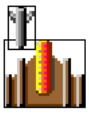
Even in 2D, collisions is a non-trivial problem.

- Rectanges - easy
- Enclosing convex polygon(s)
- Pixel-level testing

Information Coding / Computer Graphics, ISY, LiTH

2D collision detection with rectangles

Testing with the bounding box: often unsatisfactory. Too crude approximation!



Smaller rectangles make decent compromises.

Special cases

When using pseudo-3D, sprites in perspective, the shape outline is not usable.

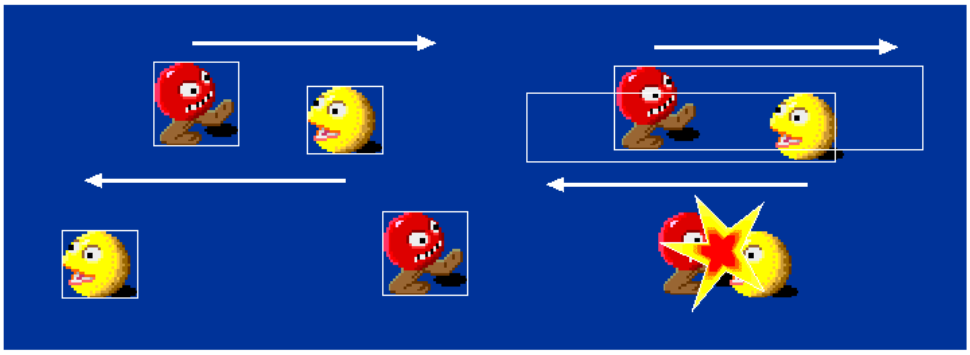
Simple approach: Modified box:

Wrong

Right

Moving objects

Fast moving objects may pass right through each other.



Wrong Right

An example of *temporal aliasing*

Fast-moving objects

- Test with elongated shapes (sweeping)
- Test several times along the trajectory (multisampling)

Information Coding / Computer Graphics, ISY, LiTH

Polyhedra-polyhedra collisions

Naive method: Check all polygons in all objects against all polygons in all objects!

Polyhedra-polyhedra collisions

Practical methods:

1) Separate into broad phase and narrow phase

2) Object hierarchies

Broad phase - narrow phase

Really three phases:

Global phase

Bounding shapes phase

Narrow phase

Broad phase

Bounding shapes phase

Make simple checks to see if objects are so close that we should test in detail!

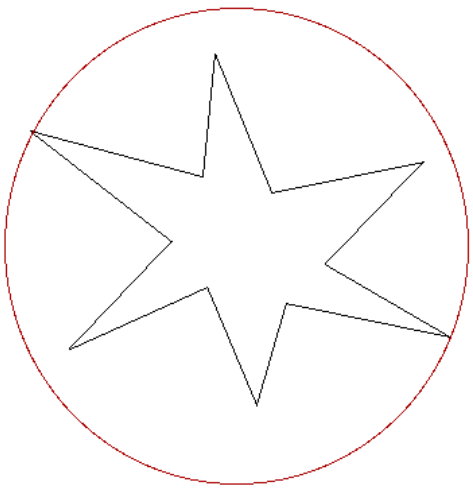
Use simple bounding shapes!

Typical shapes:

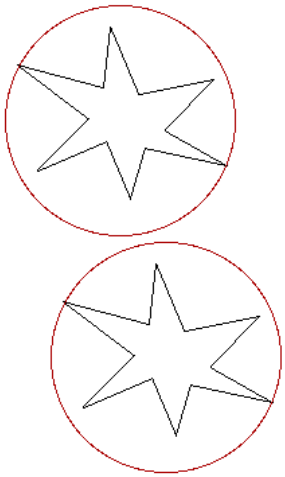
- Sphere
- AABB (axis aligned bounding box)
- OBB (oriented bounding box)

Information Coding / Computer Graphics, ISY, LiTH

Sphere



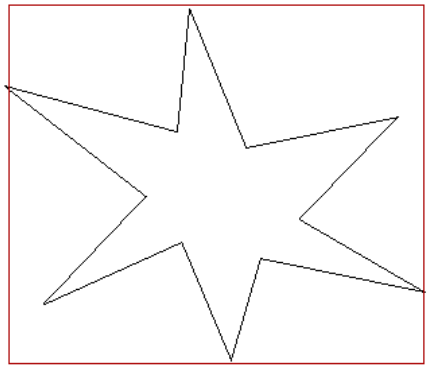
Sphere



- Very simple tests. $d^2 < (r_1 + r_2)^2$
- Calculate radius once and for all. Never changes for rigid objects. Loop through all vertices, pick biggest distance to center.
- Rotation allowed!
- Good fit for compact objects.
- No flat surfaces, object can not rest on each other.

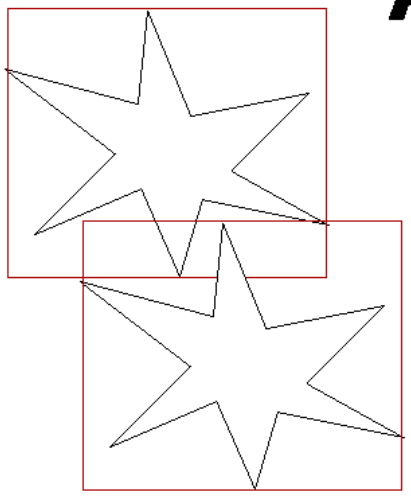
Information Coding / Computer Graphics, ISY, LiTH

AABB



Axis aligned bouding box

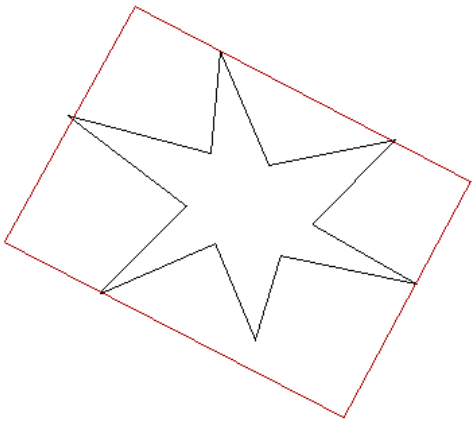
AABB



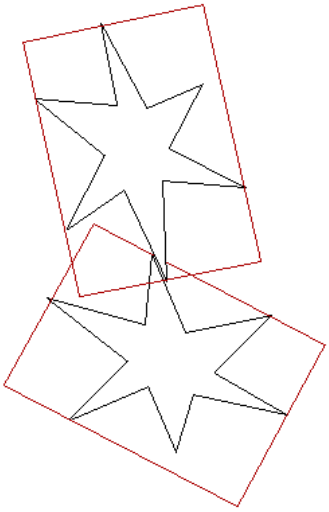
- Very simple tests. 6 tests, like the 2D case.
- Calculate by finding max and min along each axis.
- Can't be rotated freely!
  1) Recalculate on rotation.
  2) Make new AABB from rotated vertices of AABB.
- Not very good fit on many objects.

Information Coding / Computer Graphics, ISY, LiTH

OBB
Oriented bounding box



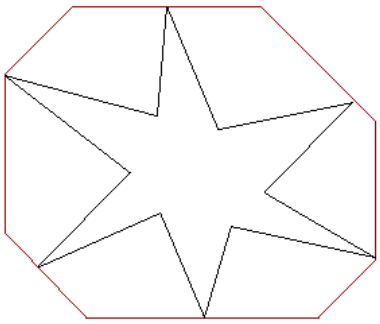
OBB
Oriented bounding box



- Good fit, you can find smallest possible box. Few false hits.
- Complicated tests.
- Rotation allowed.

Information Coding / Computer Graphics, ISY, LiTH

k-DOP
Discrete oriented polytope with k sides



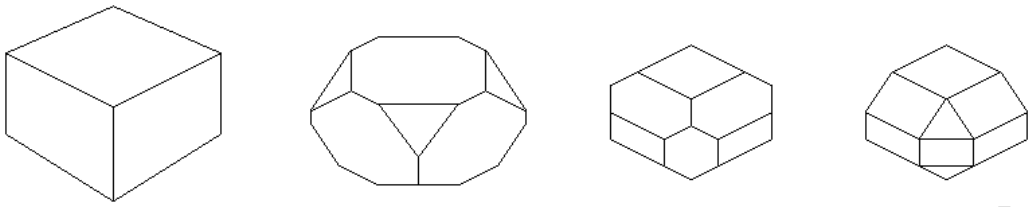
Like AABB but you can cut off corners and sides
Fairly simple tests
Fewer false hits

k-DOP

Discrete oriented polytope with k sides

Four kinds:

6-DOP (= AABB)
14-DOP (corners)
18-DOP (sides)
26-DOP (sides and corners)



6-DOP 14-DOP 18-DOP 26-DOP

Information Coding / Computer Graphics, ISY, LiTH

Balancing the broad and narrow phases

$$T = N_V C_V + N_p C_p$$

Simple broad phase - many false hits - high N_p.

Elaborate broad phase - few false hits - lower N_p but high C_V.

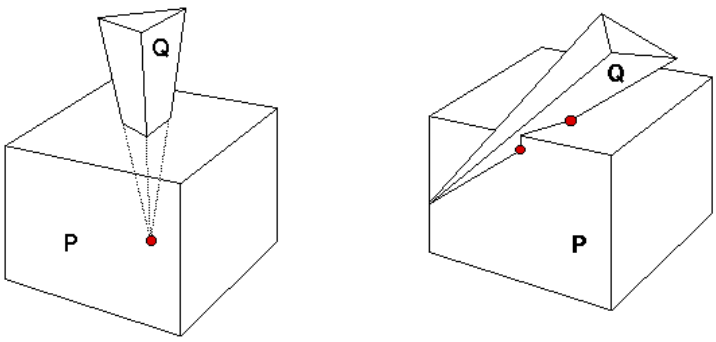
Narrow phase

Containment/intersection test

Volume intersection

Advanced methods: SAT, GJK

Containment/intersection



Containment/Intersection

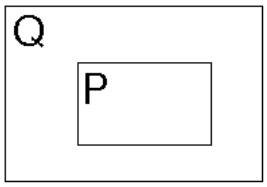
Exact test between two polyhedra, Q and P.

Two tests are needed:

1) Is any vertex in Q contained in P or vice versa?
2) Does any edge of Q penetrate a face of P?

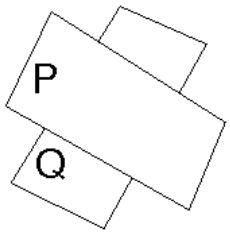
The need for all tests

1) in both directions:



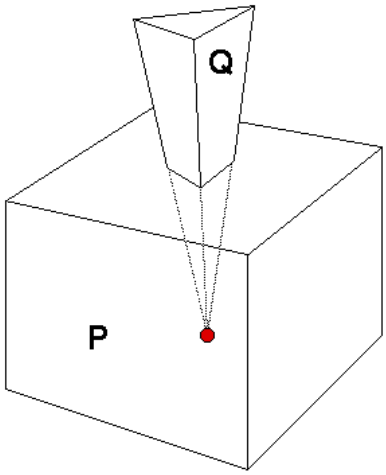
Q contains points in P,
P contains no points in Q

2) when 1) fails:



Neither P nor Q contain points of each other, but still overlap!

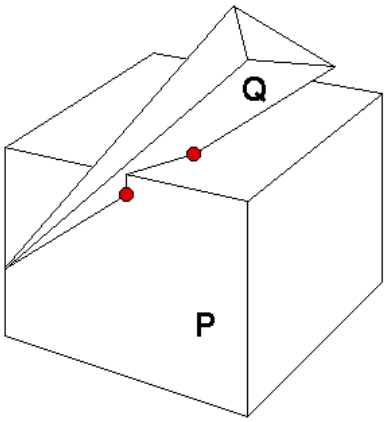
Test 1:



A point in Q is inside P

Information Coding / Computer Graphics, ISY, LiTH

Test 2:

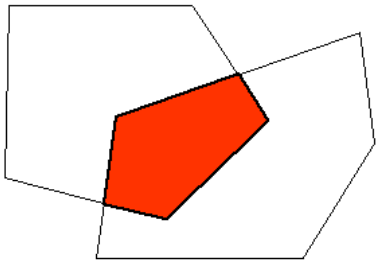


An edge of Q intersects sides of P

Volume intersection

Split P by any plane in Q, repeat. When all planes have been used, the intersection has been calculated.

Can be fairly fast if a good first plane is chosen.



Comparison

Containment/intersection:

$V_p{*}P_q+P_p{*}V_q + E_p{*}P_q+P_p{*}E_q$ $O(N^2)$

Volume intersection:

$\sum k^i P_q \approx P_q \log P_p$ $O(N\log N)$

where $i = 0$ to P_p and $k < 1$.

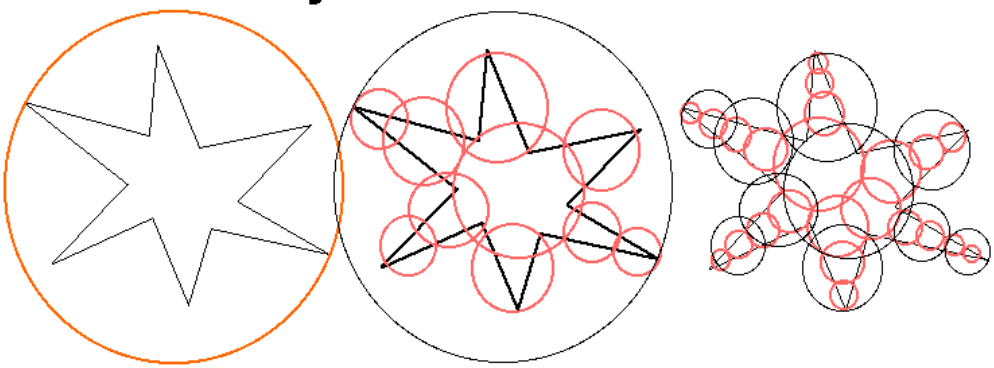
Advanced methods

Volume intersection can sometimes simplify by starting ”right”.

Advanced methods take this even longer. Remember last time, reduce test to very few. Just test the ”right test”.

Information Coding / Computer Graphics, ISY, LiTH

Single phase algorithms Object hierarchies



Collision detection is done to the level that available time permits

Global phase collision detection

Avoid many-to-many checks

Space subdivision

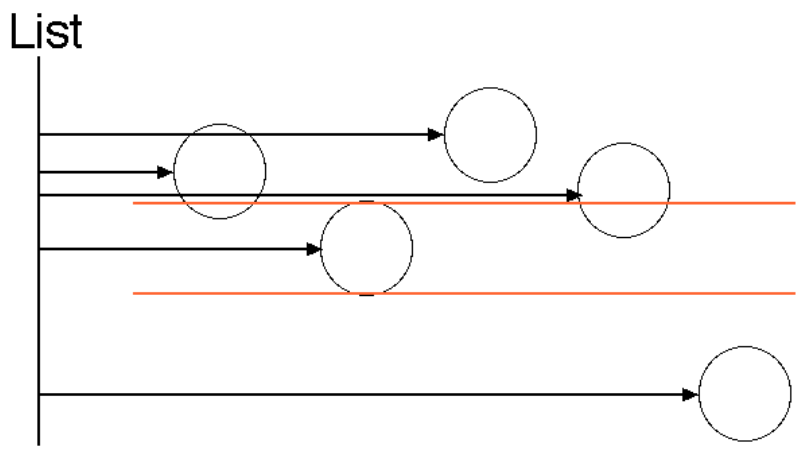
Simple: Linear sorting

More elaborate: Hierarchies, octrees…

Simplify objects out of view?

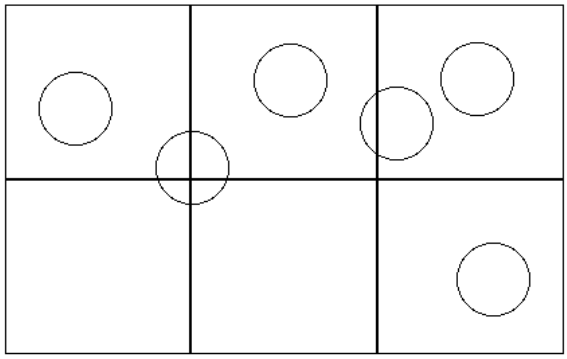
Linear sorting

Reduces the problem by 1 dimension



Subdivision for collision detection

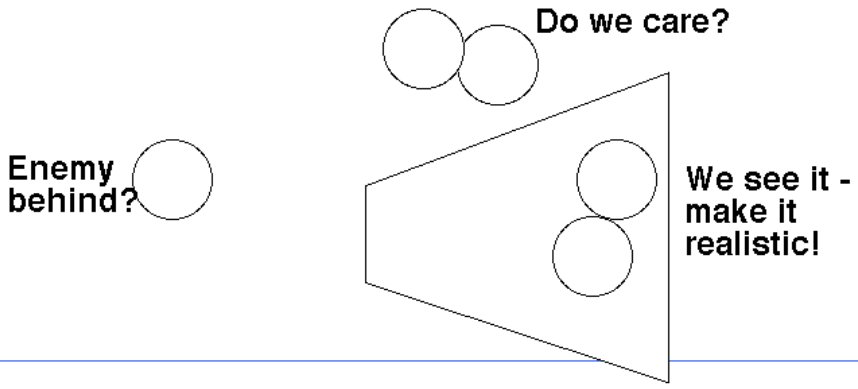
Subdivision by BSP trees, octrees, quad-trees
Generally useful for most large world problems



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

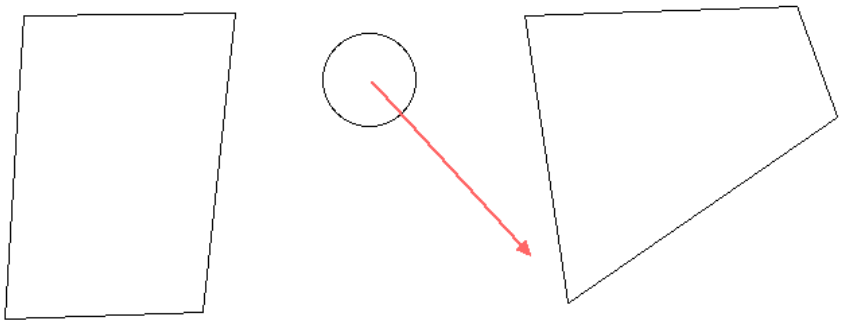
Again, we can take advantage of visibility to reduce workload.

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.

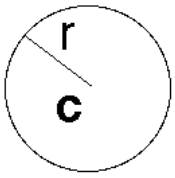
Information Coding / Computer Graphics, ISY, LiTH

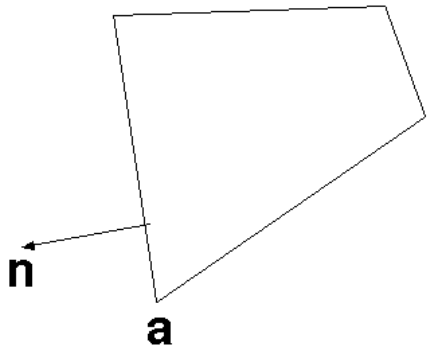
Spheres in polyhedra world: cameras as well as objects



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

Sphere - polyhedra distance



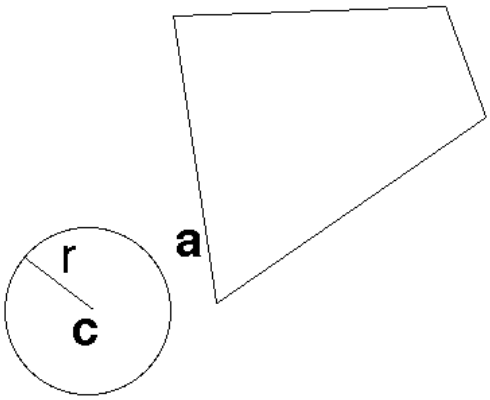


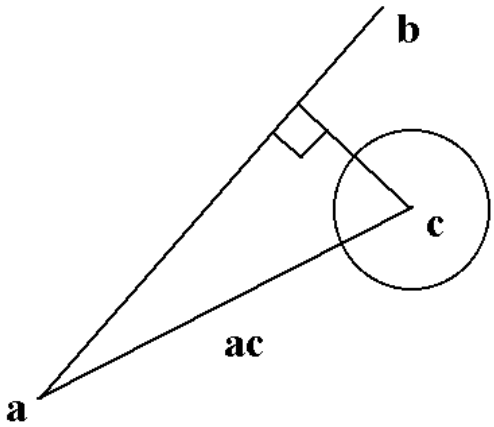
$\mathbf{n} \cdot (\mathbf{c} + r \cdot \mathbf{n}) > \mathbf{n} \cdot \mathbf{a}$

=> 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!

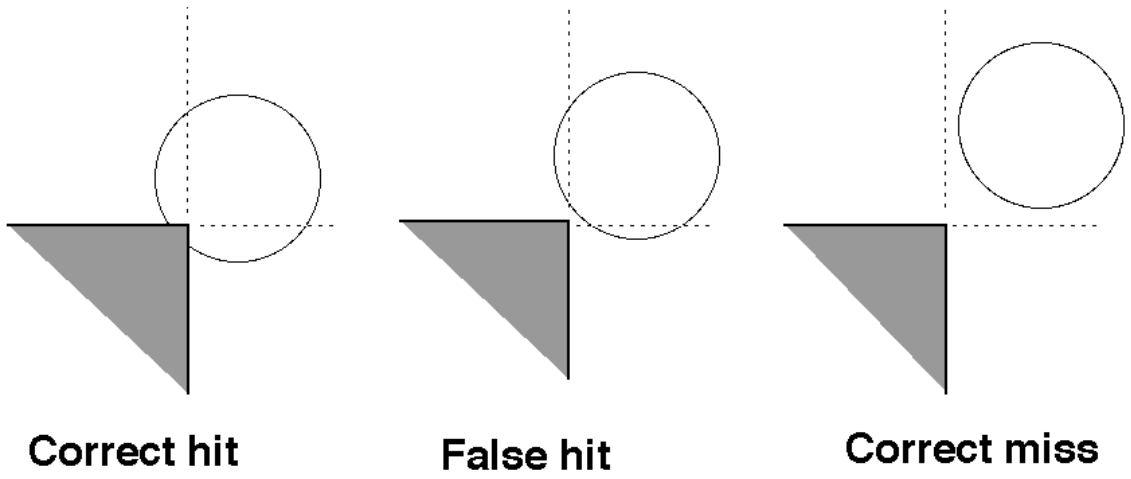
Exact test: Check corners and edges too





$|c - a| < r^2$

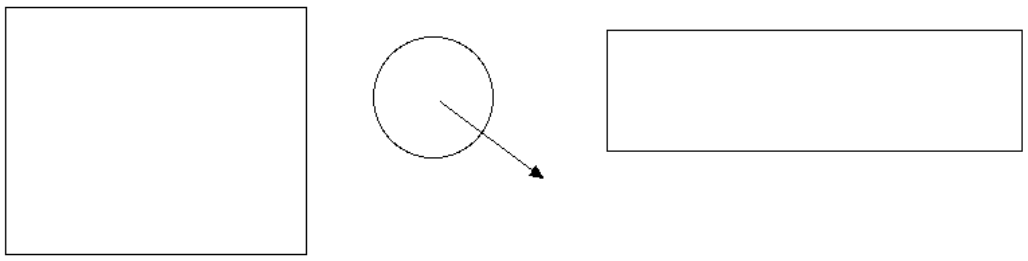
Simplification: Test all planes only! If outside *any* plane - outside, otherwise it intersects.



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 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.

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

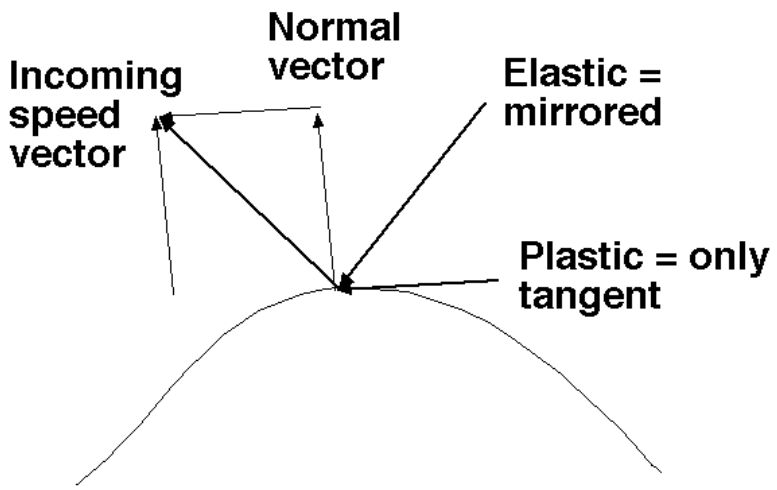
$$a = g + \sum f/m$$
$$s = s + a$$
$$p = p + s$$

("Euler integration")

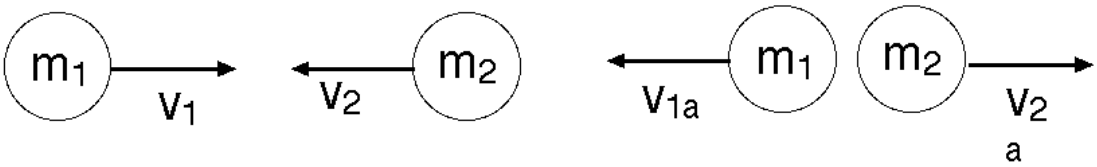
Modify speed and position in collisions

Information Coding / Computer Graphics, ISY, LiTH

Simple particle-surface collision



Plastic and elastic collisions



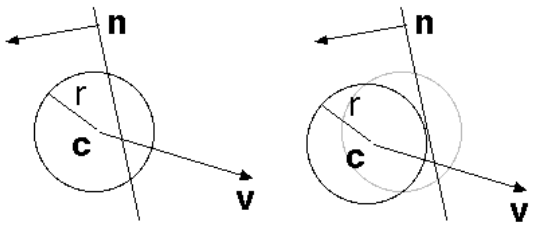
Preserve momentum

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

Elastic collisions also preserve kinetic energy

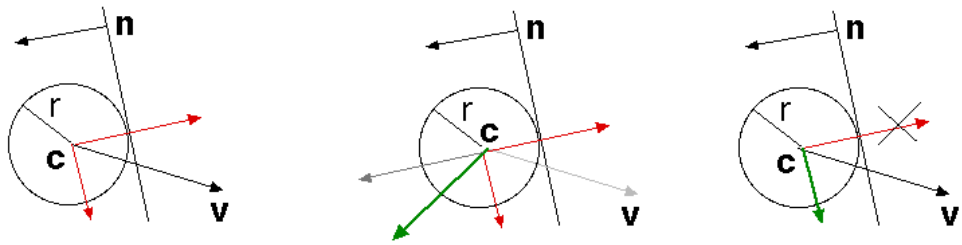
$$m_1v_1^2 + m_2v_2^2 = m_1v_{1a}^2 + m_2v_{2a}^2$$

Collision handling sphere-polyhedra



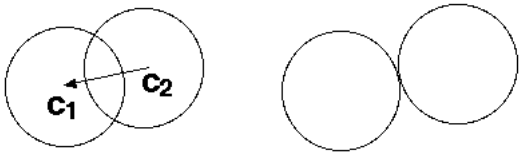
Assuming stationary polyhedra

Separate - move object away along normal vector

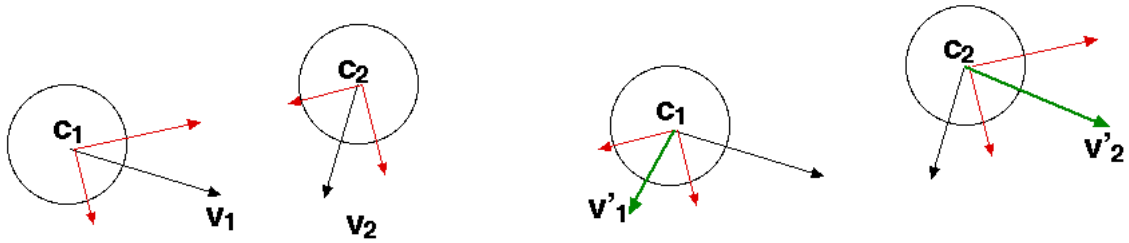


Split velocity along n | Elastic | Plastic

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 about animation

Must focus on convex shapes

Simple collision detection with AABB or spheres

Global phase also important

Narrow phase expensive and complex