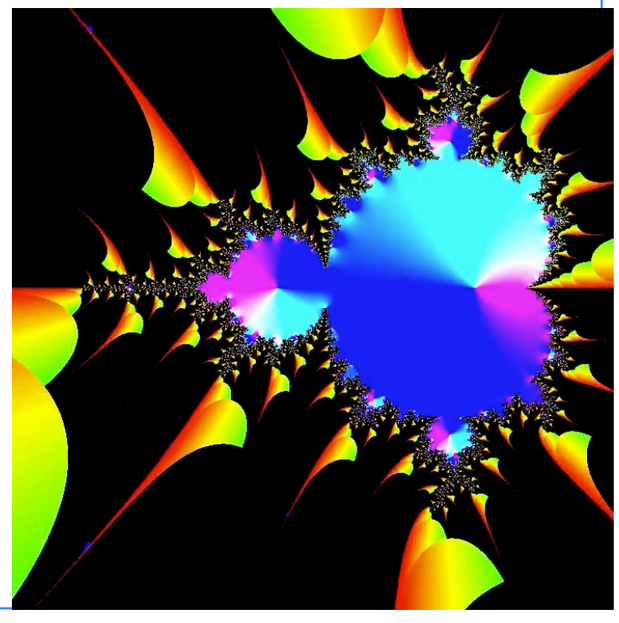
TSBK07/TSBK11

Computer Graphics

Ingemar Ragnemalm, ISY





Lecture 11

Collision detection

Collision handling

Next lab

Dugga 5 - last before the retake (9th of march)

Many have already reached "pass" level.

Time to decide on a project

- Should be preliminary decided wednesday the 4th (second to last lecture)
- Final specification same day as last lecture (friday 6th)

Title
Participants
Brief summary
Will-do
Might-do

Project ideas:

(The more basic ones)

TSBK11: Look here: ->

- 3D labyrinth
- Interactive solar system
- Driving simulator
- Flight simulator
- Enhanced terrain rendering

Rather TSBK07: ->

- Robot (or Godzilla) with moveable limbs
- Large virtual reality with frustum culling/level of detail/portals/etc
- Real-time raytracer or path tracer

Important: Anything course relevant that you feel is interesting is valid!

Unfinished labs?

Don't panic!

3 sessions left, but also:

The course does not end here.

"Project sessons" thursdays VT2.

Project work, but labs are also welcome.

Additional tools reminder

1) **SimpleGUI**: GUI controls in an OpenGL window

One source file, minimal dependencies

2) Code for **simpler upload of shader data**

You may already use uploadMat4ToShader() in the VectorUtils files. *Add more as needed!*

3) **Geometric Utilities (GLUGG)** for procedural geometry.

4) **LoadTextures**, a bigger texture loader (4 formats)

5) **SimpleFont** and **my-stbtt**, for drawing text

6) **Call me AL**, sound playing API using OpenAL

Lecture questions

1. What bounding shapes are suitable for the broad phase?

2. What does SAT mean, as in the SAT theorem, and how can you apply it to collision detection?

3. How can spheres vs polyhedra be a really simple but working collision detection?

4. How do you calculate the collision response between a sphere and a flat stationary surface?

5. Suggest a global phase collision detection algorithm

Collision detection and handling

A major problem in many real-time animations.

Even modern high-budget games have problems!

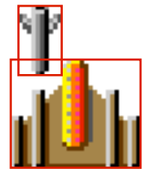
2D collision detection

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

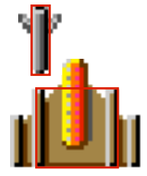
- Rectanges or circles - easy
- Enclosing convex polygon(s)
- Pixel-level testing

2D collision detection with rectangles

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



Smaller rectangles make decent compromises.



Special cases

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

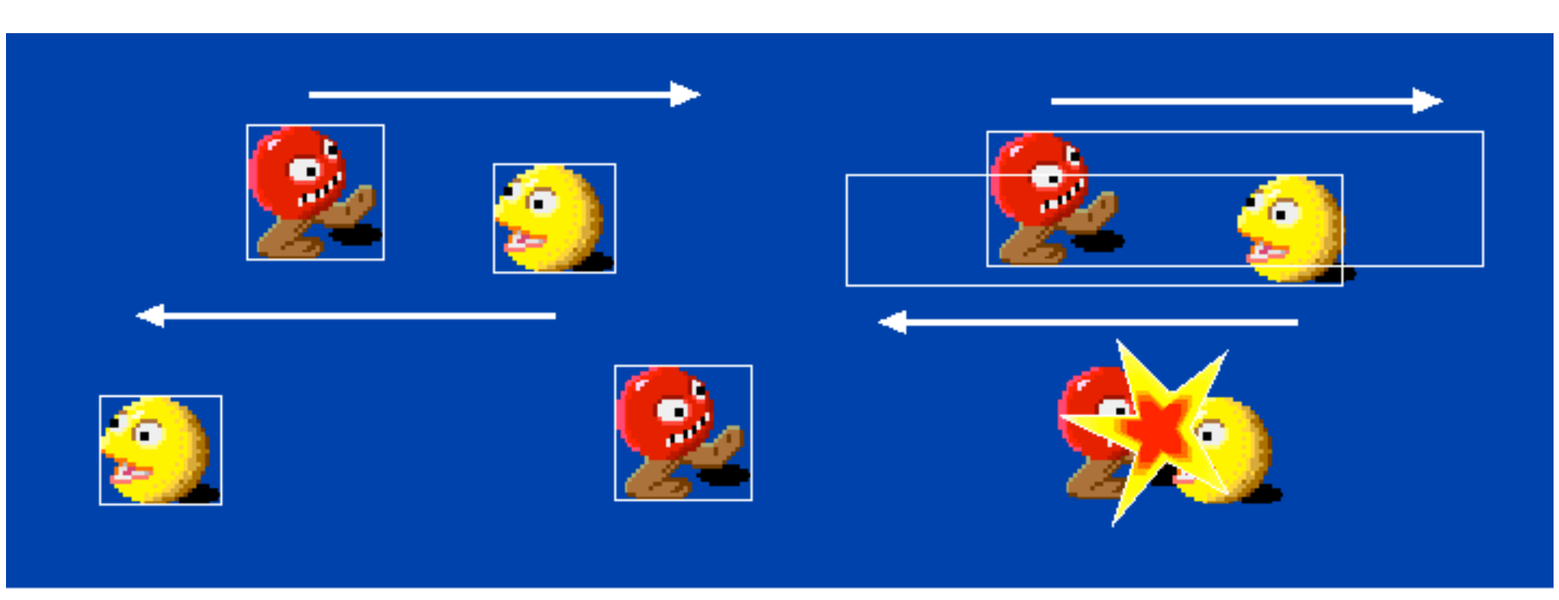
Simple approach: Modified box:

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)

Polyhedra-polyhedra collisions

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

How would you do that?

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

} Broad phase

Narrow phase

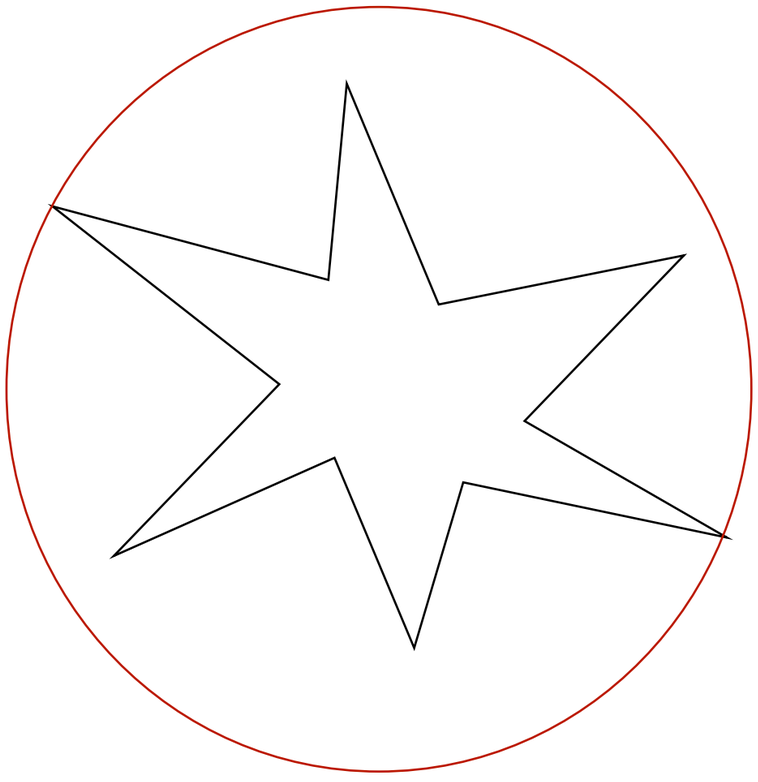
Bounding shapes phase

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

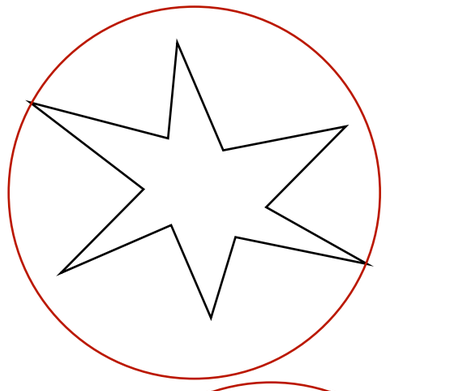
Use simple bounding shapes!

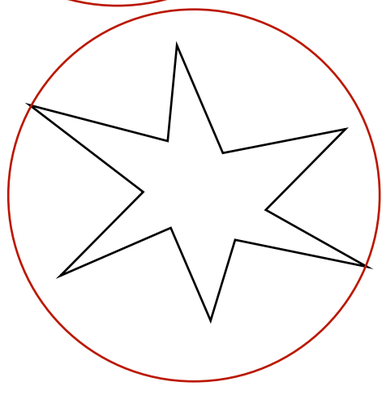
Typical shapes:
- Sphere
- AABB (axis aligned bounding box)
- OBB (oriented bounding box)

Sphere



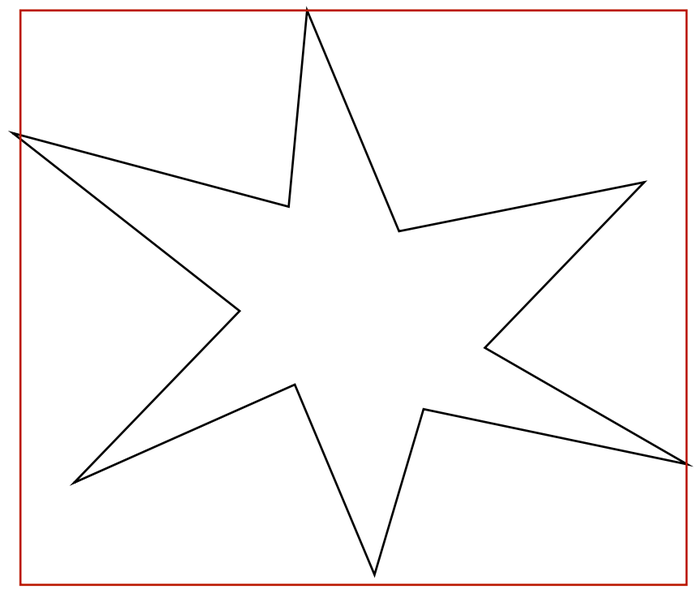
Sphere



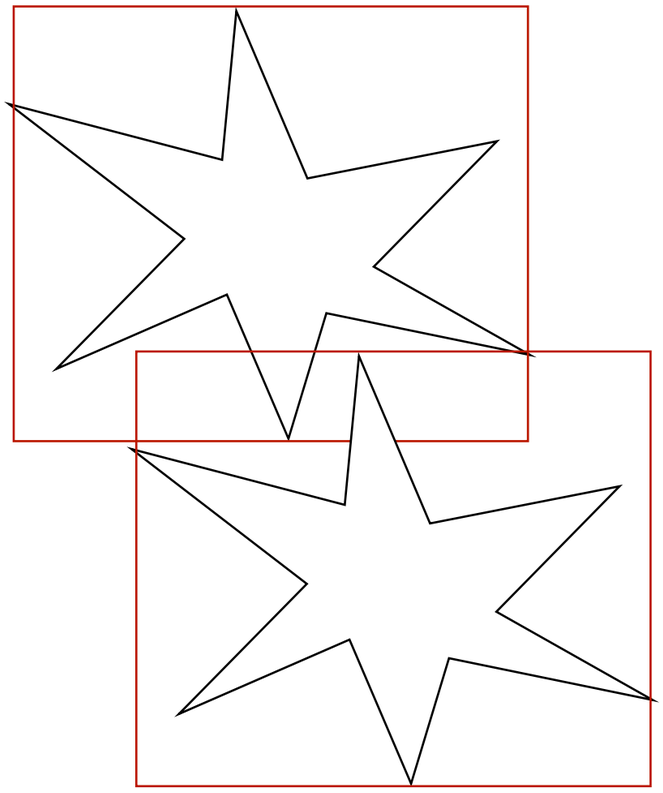


- Very simple tests. $d^2 < (r_1 + r_2)^2$
- Calculate radius once and for all. Never changes for rigid objects. Loop through all vertices, pick biggest distance to center.
- Rotation allowed!
- Good fit for compact objects.
- No flat surfaces, object can not rest on each other.

AABB



Axis aligned bounding box



AABB

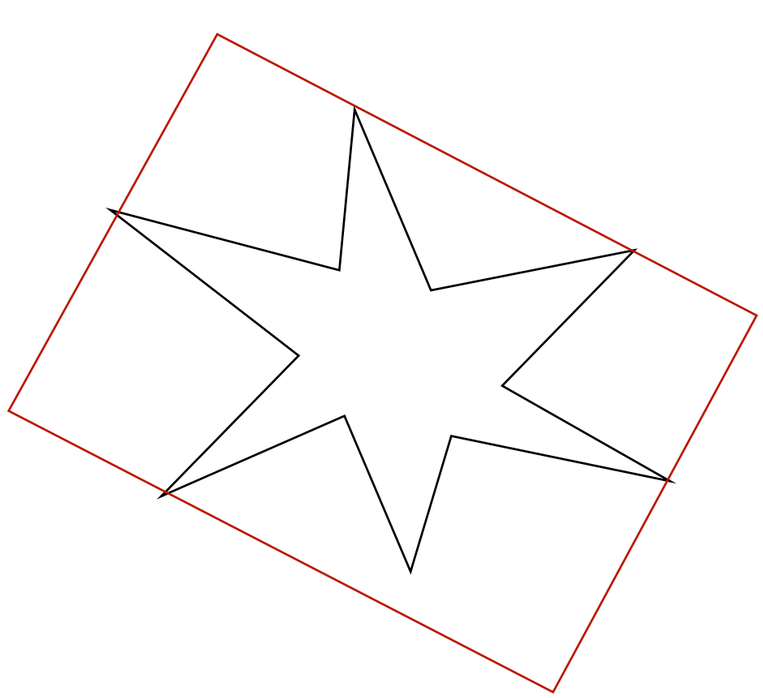
- Very simple tests. 6 tests, like the 2D case.

- Calculate by finding max and min along each axis.

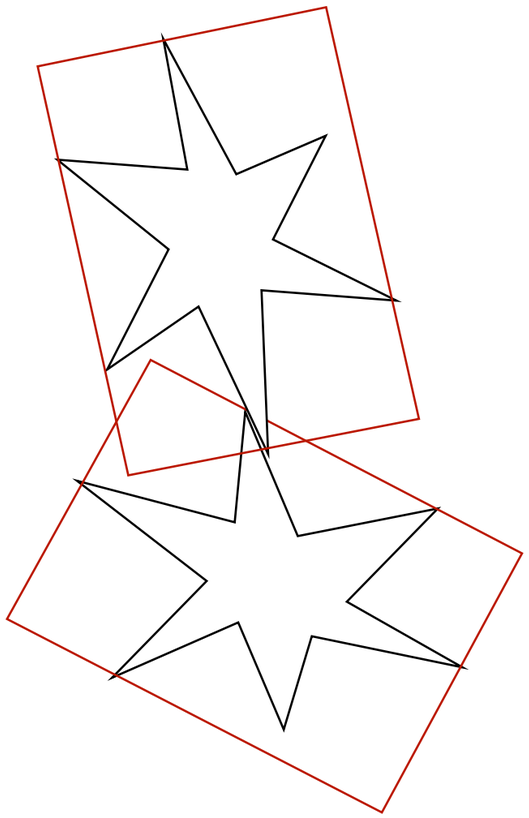
- Can’t be rotated freely!
1) Recalculate on rotation.
2) Make new AABB from rotated vertices of AABB.

- Not very good fit on many objects.

OBB
Oriented bounding box

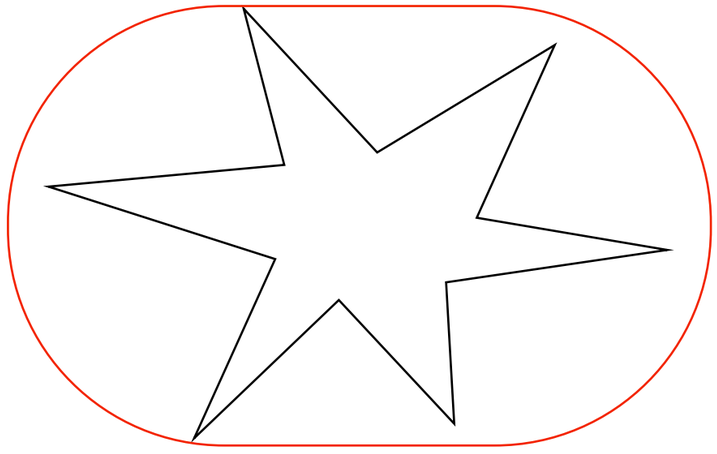


OBB
Oriented bounding box



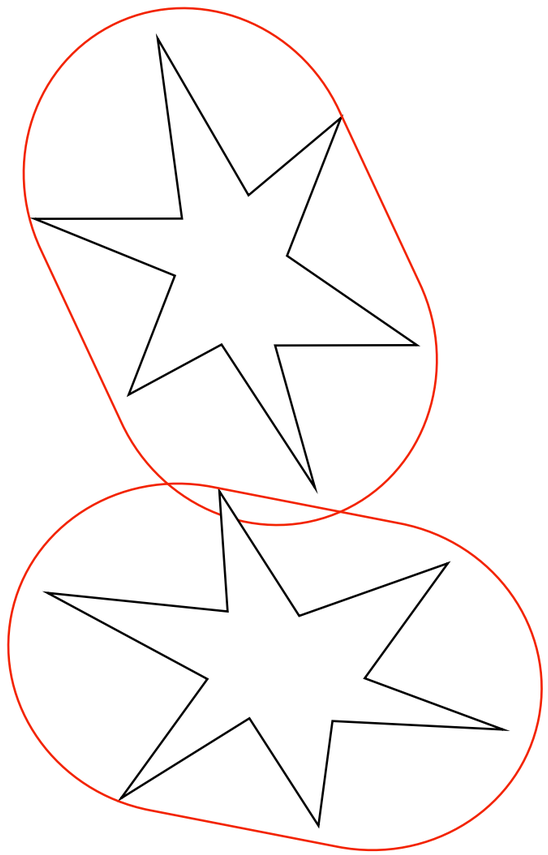
- Good fit, you can find smallest possible box. Few false hits.
- Complicated tests - but can be simplified. (SAT, more later)
- Rotation allowed.

Capsule



Capsule

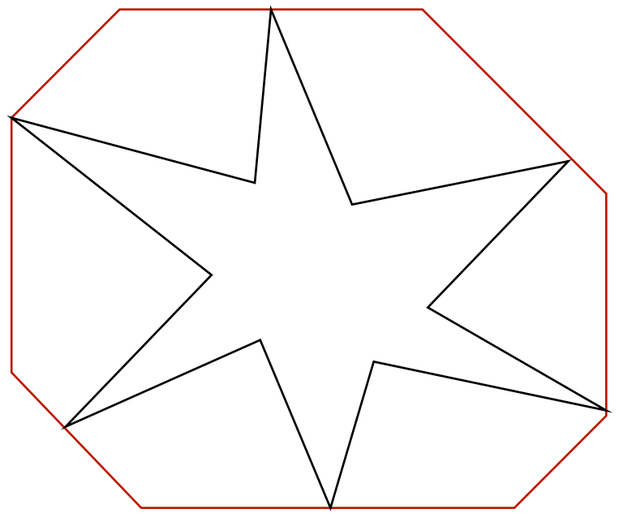
Cylinder + 2 half spheres



- Good fit, you can find smallest possible box. Few false hits.
- Fairly complicated tests (sphere + OBB)
- Rotation allowed.

k-DOP

Discrete oriented polytope with k sides



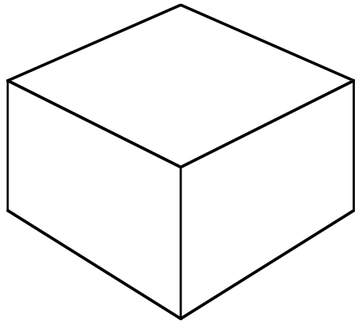
Like AABB but you can cut off corners and sides
Fairly simple tests
Fewer false hits

k-DOP

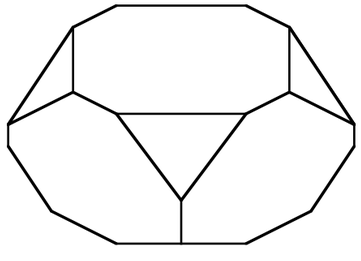
Discrete oriented polytope with k sides

Four kinds:

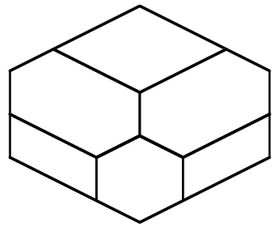
6-DOP (= AABB)
14-DOP (corners)
18-DOP (sides)
26-DOP (sides and corners)



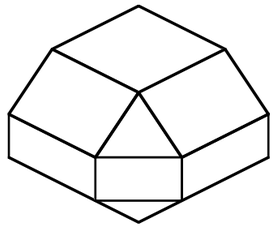
6-DOP



14-DOP



18-DOP



26-DOP

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

Separating Axis Theorem (SAT)

Containment/intersection test

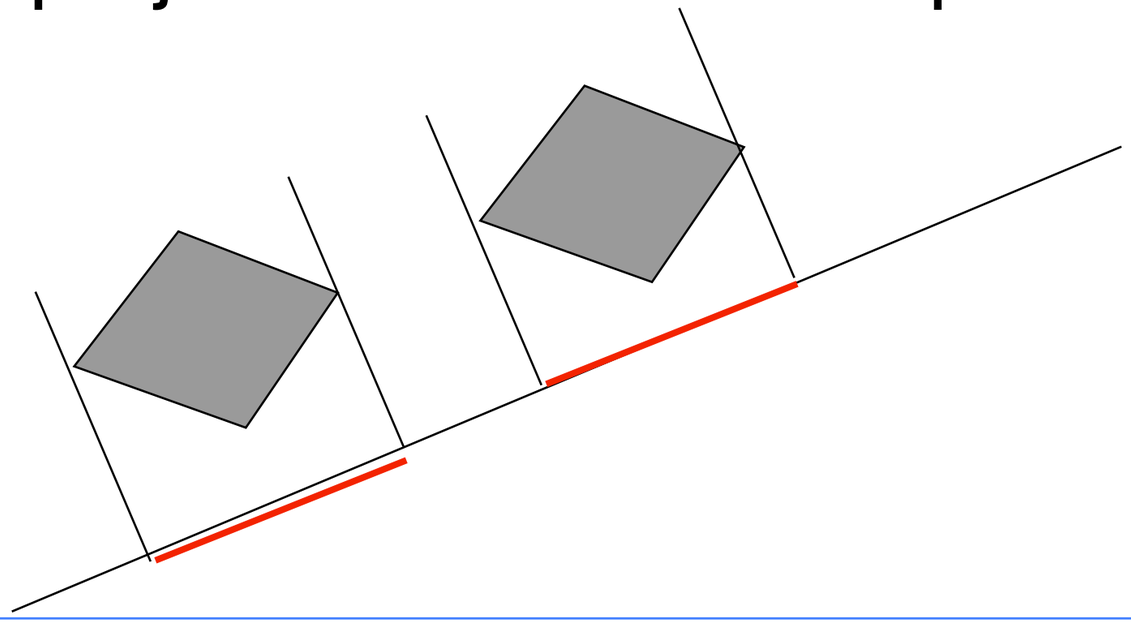
Advanced methods: GJK

Separating axis theorem (SAT)

and its application on collision detection

Separating axis theorem

Two convex objects do not overlap if there exists a line (axis) onto which the two object's projection do not overlap



Really ”Separating plane”

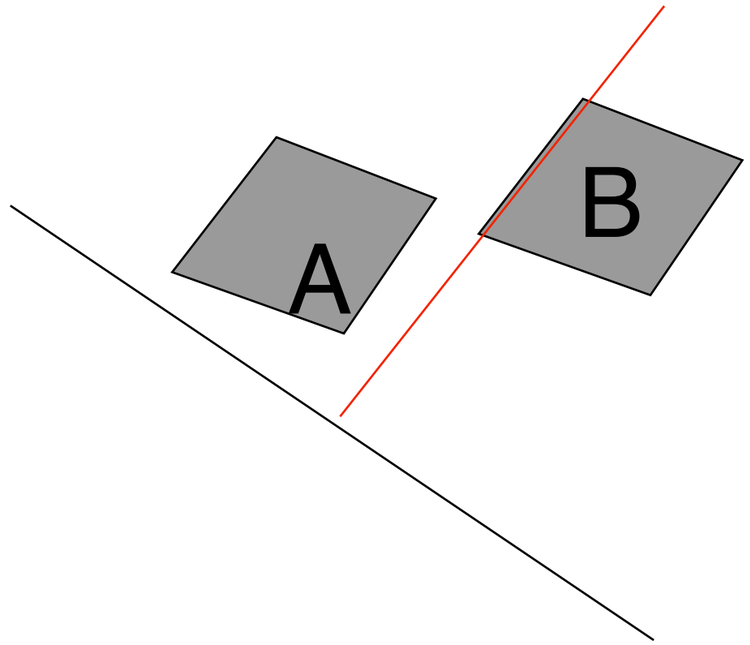
Axis = normal vector of separating plane

Can you find a plane between them?

Problem: The number of possible planes/axes to test are infinite!

SAT in practice

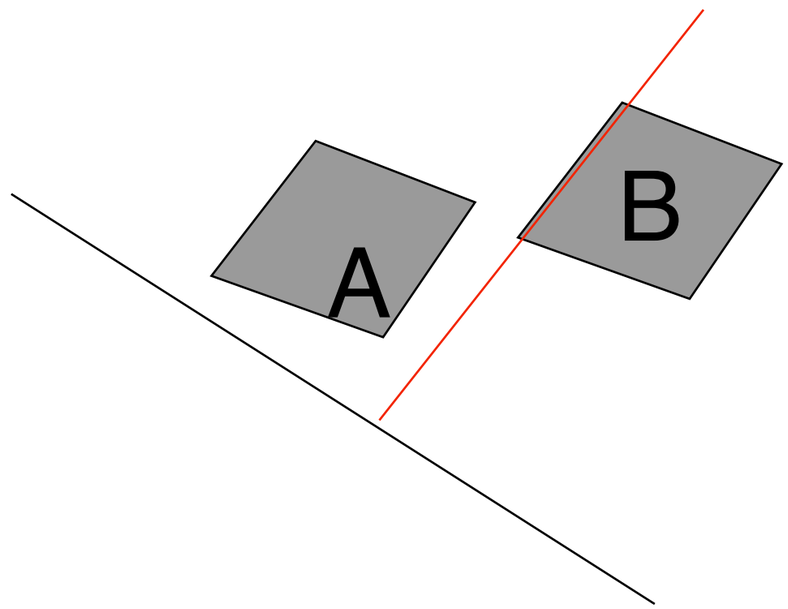
For polyhedra models, use the faces of the models!



Fairly simple in the 2D case

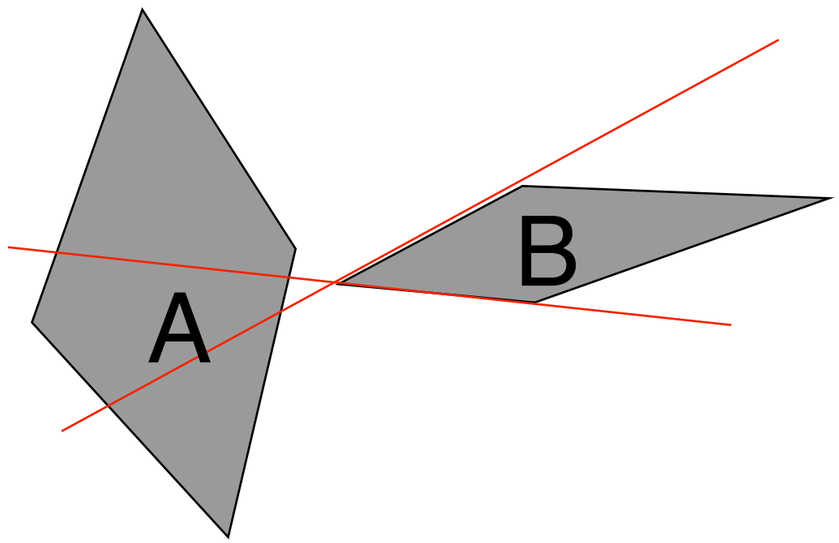
For every face in B, find plane equation

Test all vertices in A with dot product!



But you must go both ways!

There are configurations where one model has no single face that will exclude the other!



2D SAT algorithm

for all faces in A
let a be a vertex in A
hit = false
for all vertices b in B
diff = n • a - n • b
if diff > 0 then
hit = true
if not hit then
return false

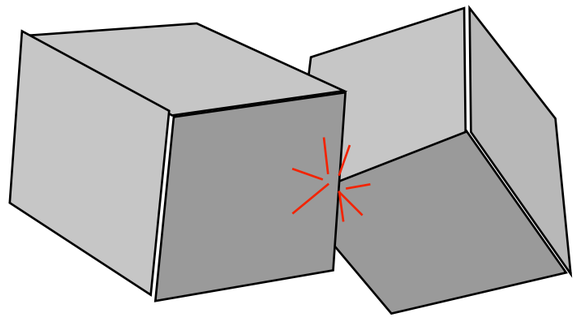
for all faces in B
(same algorithm with A and B reversed)

return true

SAT in 3D

Not as simple!

Two object meeting at an edge may fail the test!



We must add tests with more planes.

SAT in 3D

We must add tests with more planes. Which ones?

Cross product of an edge in a with an edge in b = additional potential separating plane

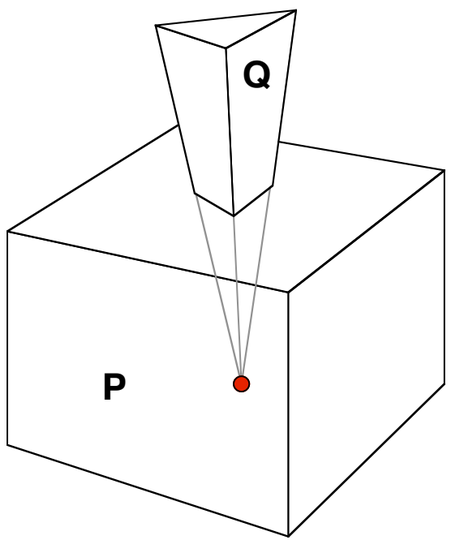
All edges in a * all edges in b!

=> Complete SAT on complex 3D objects is expensive!

So how can SAT be useful?

- Simplify. Skip some tests, accept some false hits.
- Use SAT on simplified bounding shapes. (OBB)
- Remember separating axis from previous test!

Containment/intersection



Containment/Intersection

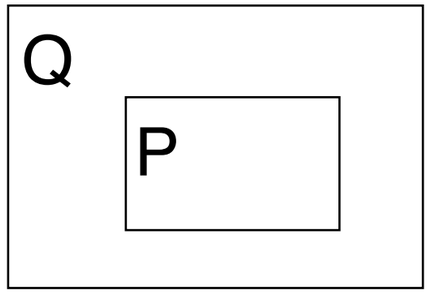
Exact test between two polyhedra, Q and P.

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?

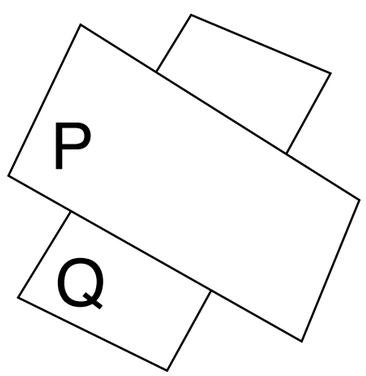
The need for all tests

1) in both directions:



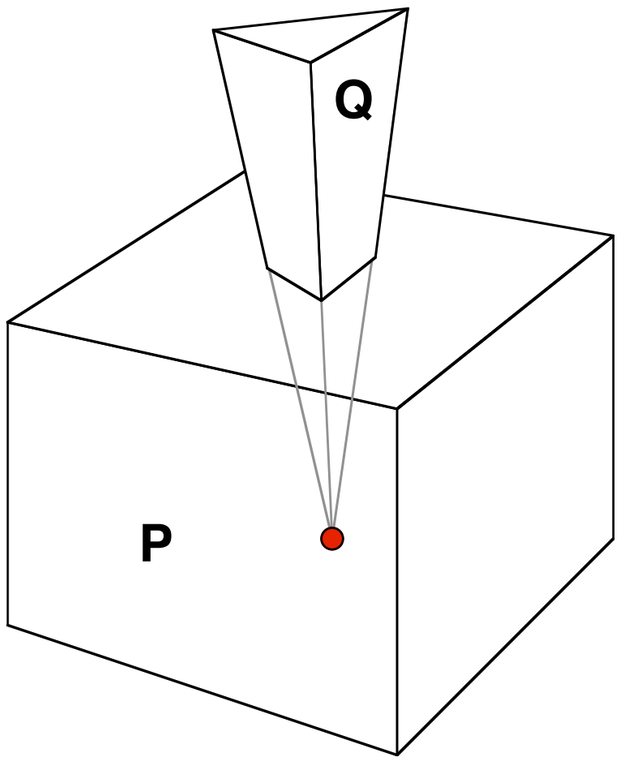
Q contains points in P,
P contains no points in Q

2) when 1) fails:



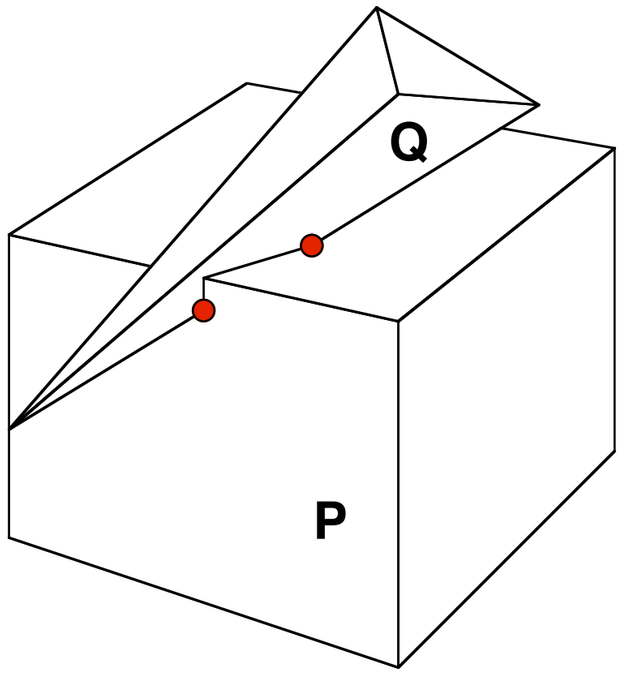
Neither P nor Q contain points of each other, but still overlap!

Test 1:



A point in Q is inside P

Test 2:



An edge of Q intersects sides of P

Comparison

Full SAT

$V_p*P_q+P_p*V_q + Ep*Eq*Vp + Ep*Eq*Vq$ $O(N^3)$

Containment/intersection:

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

Acceleration method: Optimize order of tests

Similarity over time

Remember last time, test that first, the ”right test”.

Search in the right direction

Use the center of each shape, start with most likely tests

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"

Level-of detail

Simplified models are extremely useful in collision detection!

We can go much lower than what we can do when rendering!

Reduce narrow phase complexity (C_p)!

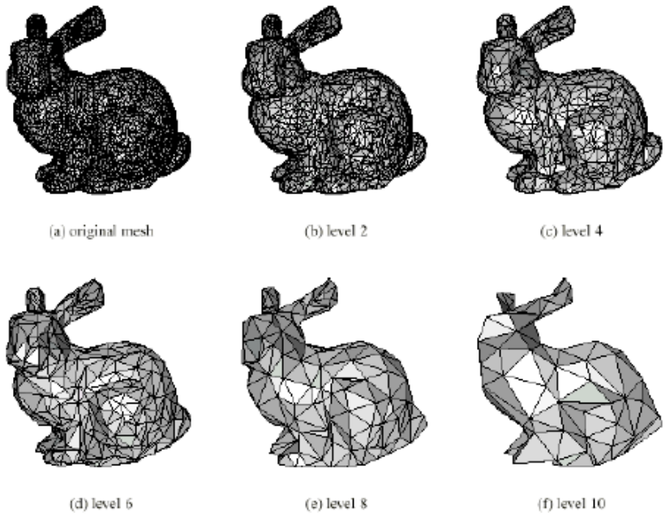


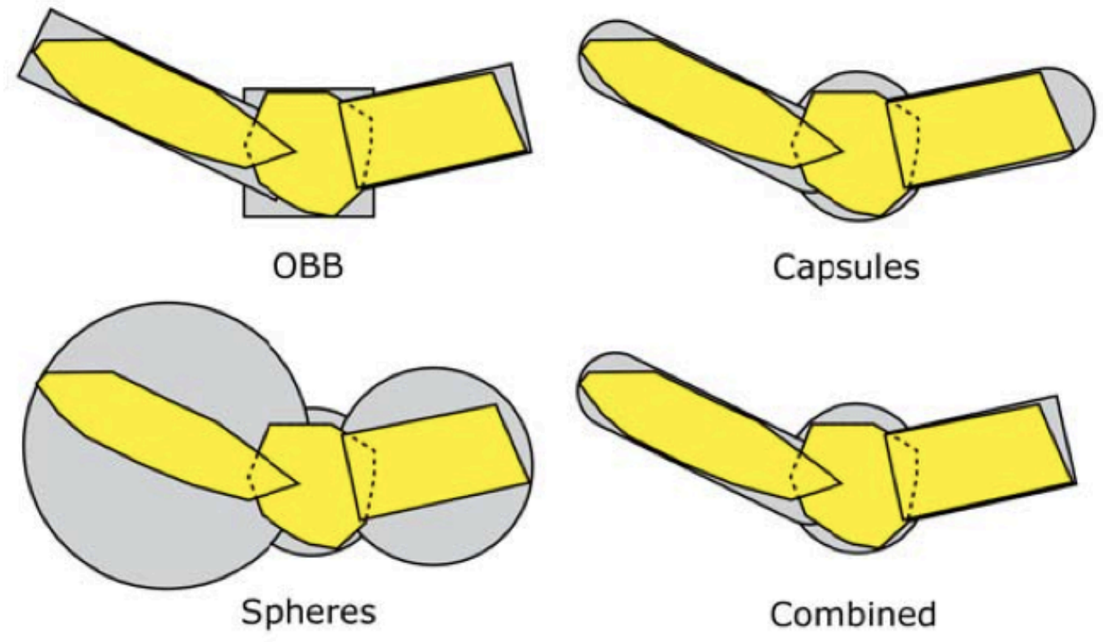
Figure 8. Stanford Bunny. Simplified meshes with 10000, 4577, 2106, 988, 463, 245 triangles

Problem: These tests only work for convex models!

Split to convex parts? Very hard for some models.
We can go much lower than what we can do when rendering!

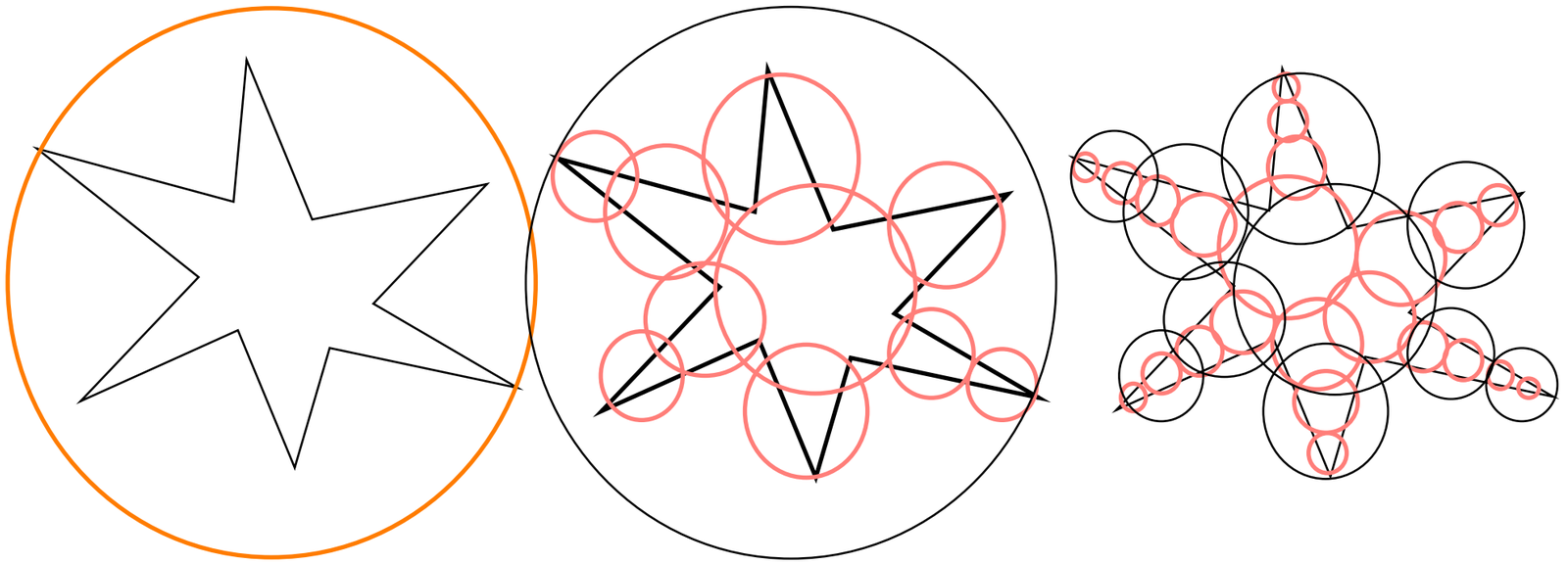
Split to parts

Split to convex parts.



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

Single phase algorithms
Object hierarchies



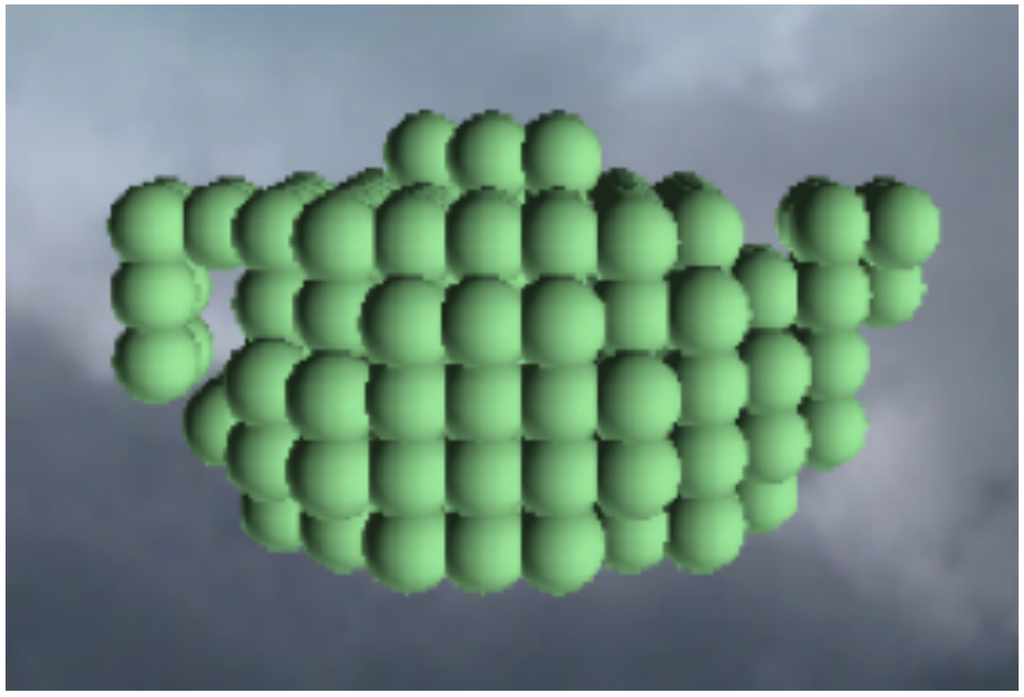
Collision detection is done to the level that available time permits

The hierachy can be produced using distance maps

Hubbard, Time-critical collision, 1996

Voxelization

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

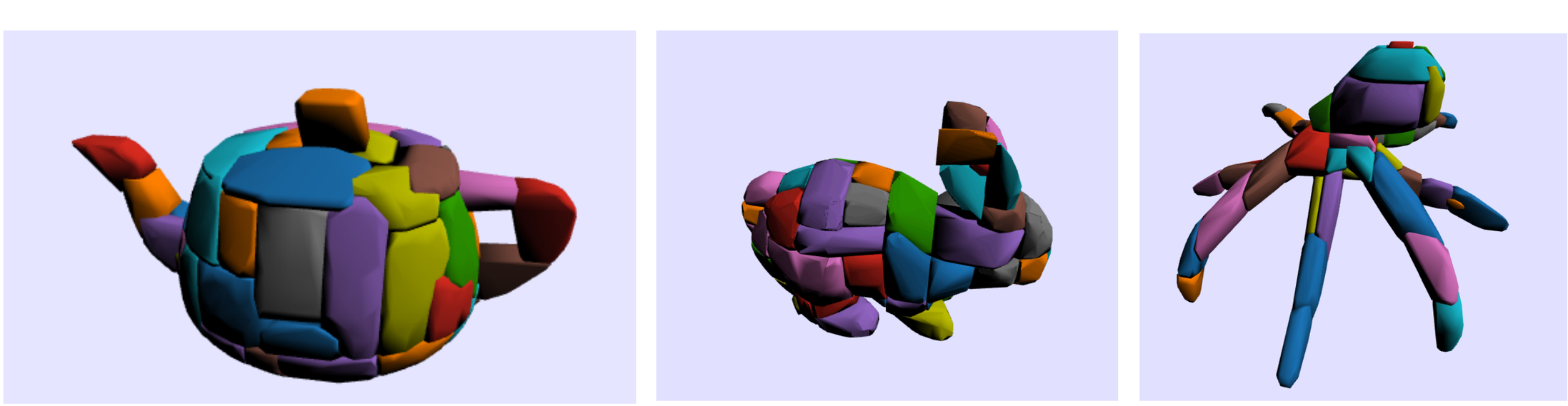


Teapot voxelized to 171 spheres

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

HACD

Hierarchical Approximate Convex Decomposition



Decomposed by v-hacd, displayed with LittleOBJLoader

Collision detection so far

Broad phase - narrow phase

Enclosing shapes

SAT

Containment/intersection

Exact and approximate

Exact:

Broad/narrow

SAT for broad phase with OBBs

Containment/intersection, simple but not fast narrow method

Moore&Williams, 1988

Approximate:

Simplified narrow phase, or no narrow phase

LOD for collisions

Hierarchies of spheres = sphere trees

Hubbard, Time-critical collision, 1996

That sounded 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!

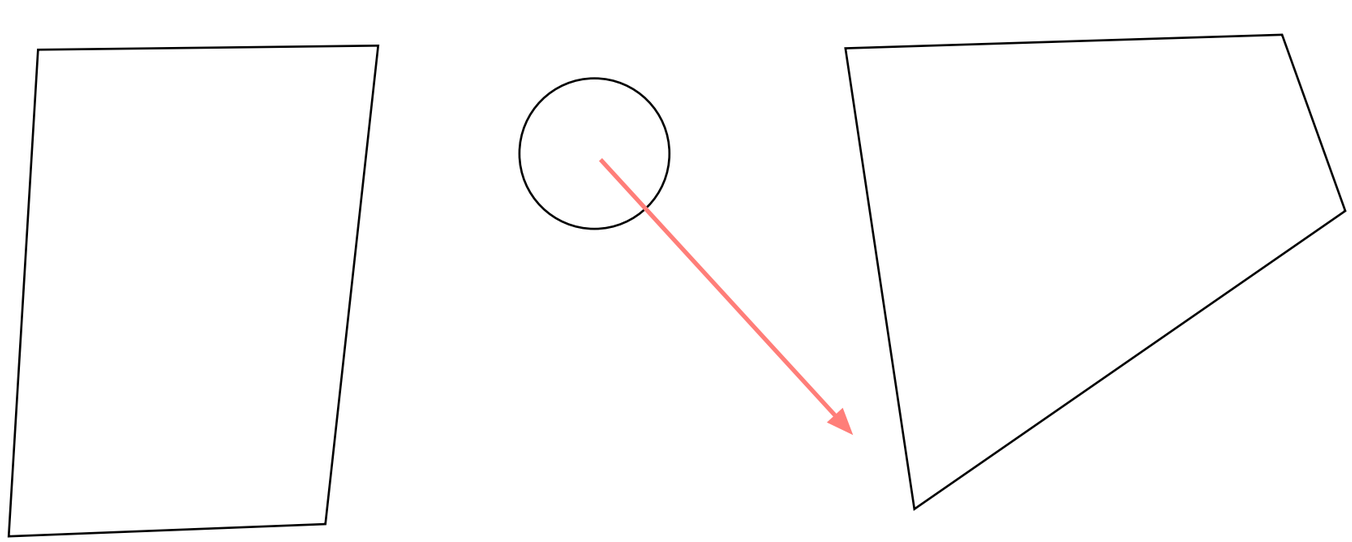
Simplified collision detection

Simplify shapes even more

Sphere-polyhedra tests

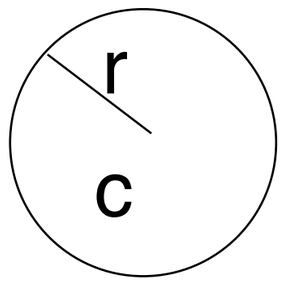
AABB worlds

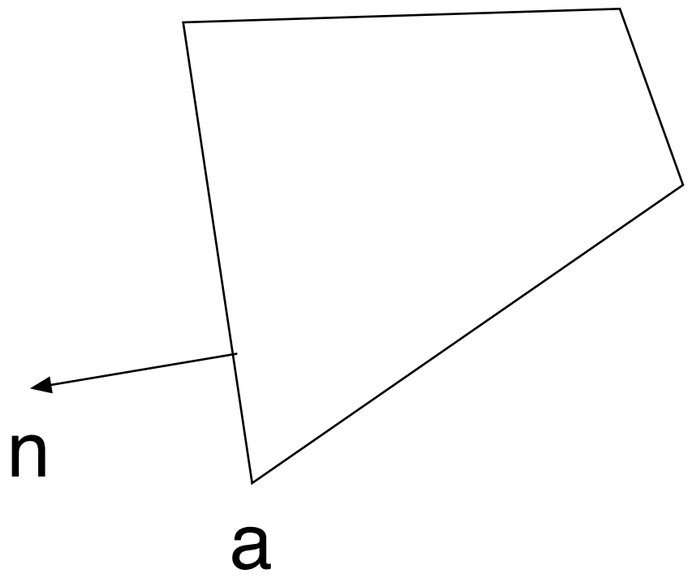
Spheres in polyhedra world: cameras as well as objects



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

Sphere - polyhedra distance



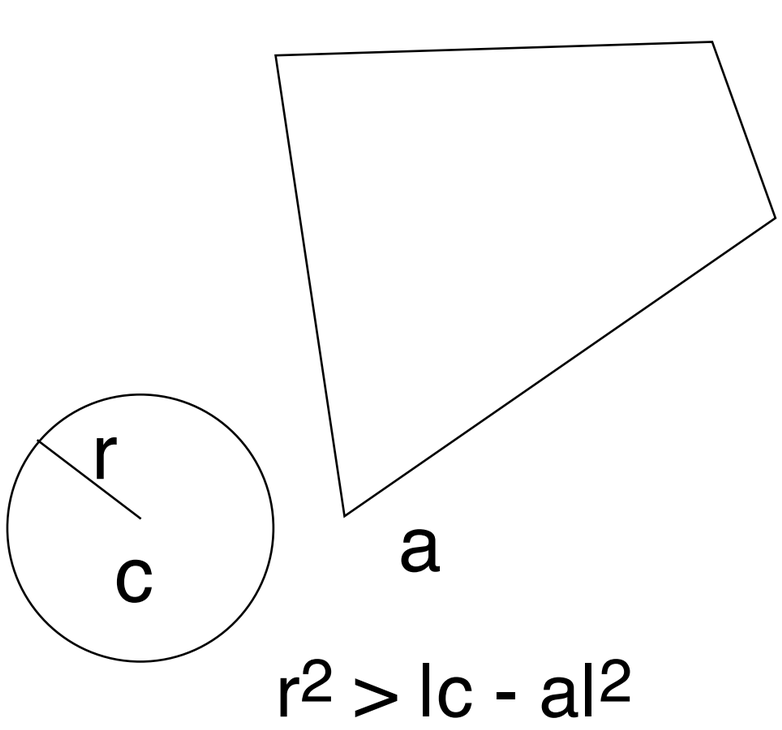


$n \bullet (c - r \cdot n) > n \bullet a$

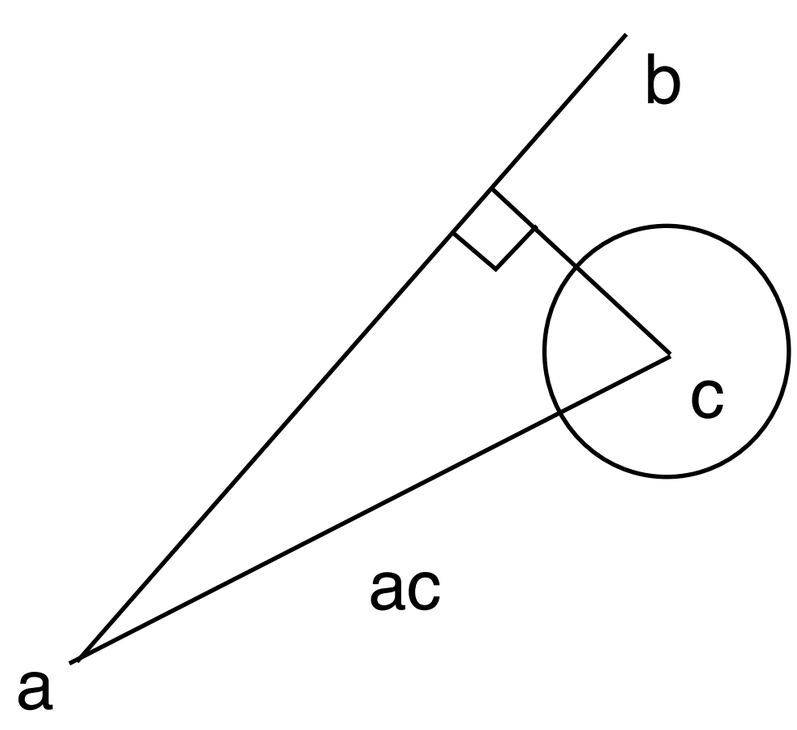
$\Rightarrow$ Distance from center to plane > r

$r < n \bullet (a - c)$ Valid when a line through c along n intersects the polygon!

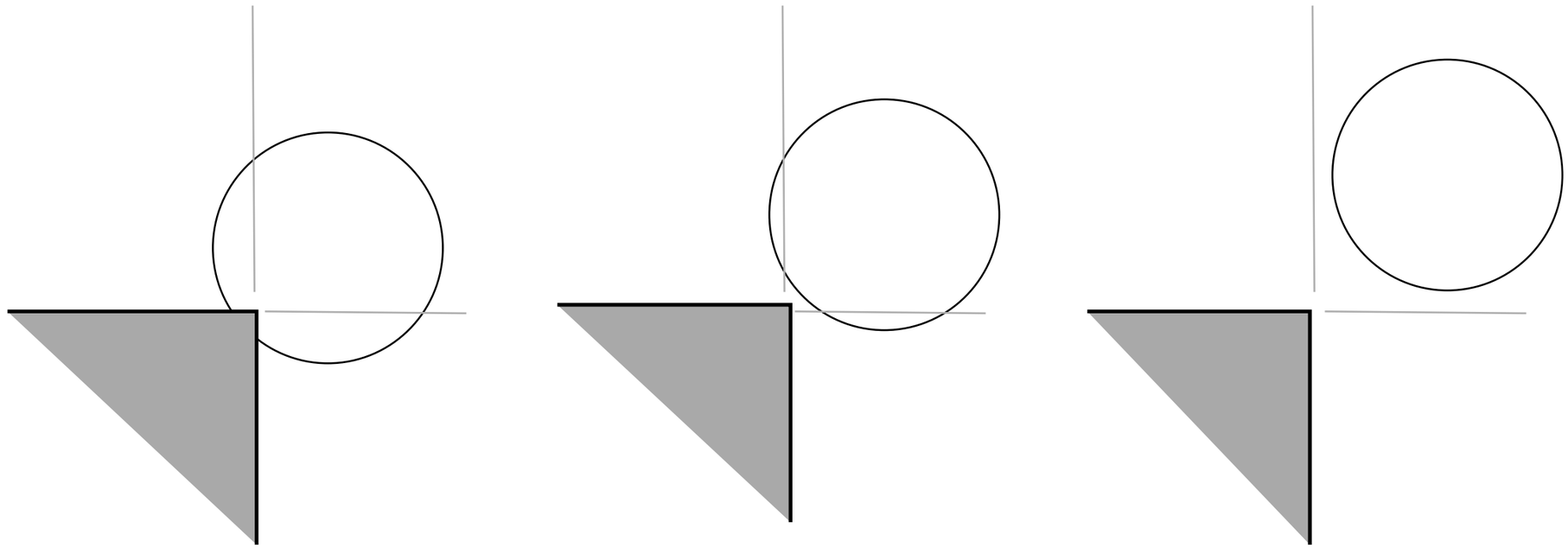
Exact test: Check corners and edges too



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



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



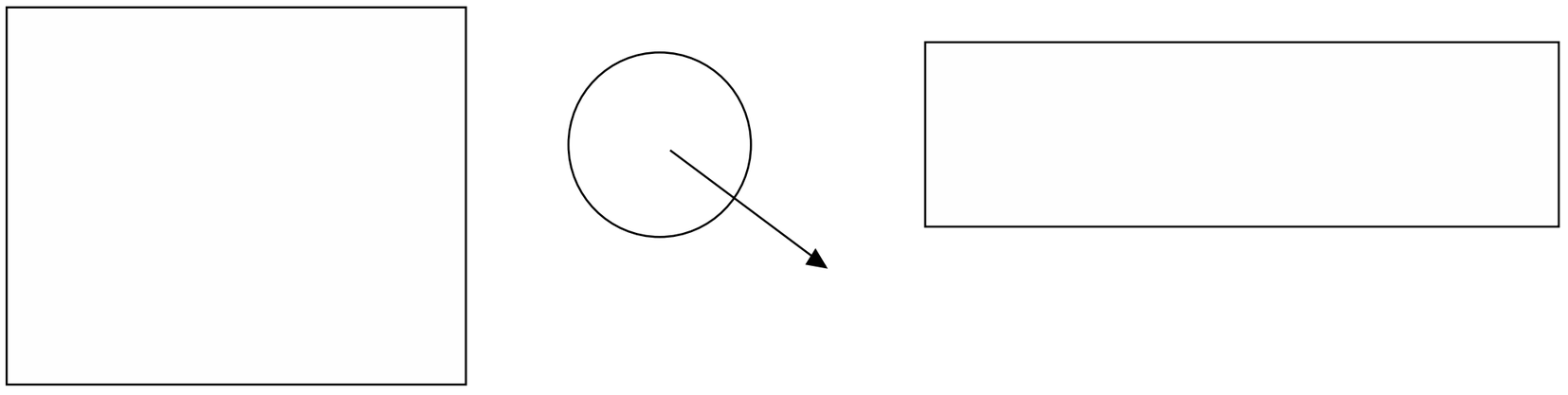
Correct hit

False hit

Correct miss

Simple case: Axis aligned 3D maze

All walls are AABBs All objects are spheres/cylinders



Final notes on the simplified camera/sphere-polyhedra 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

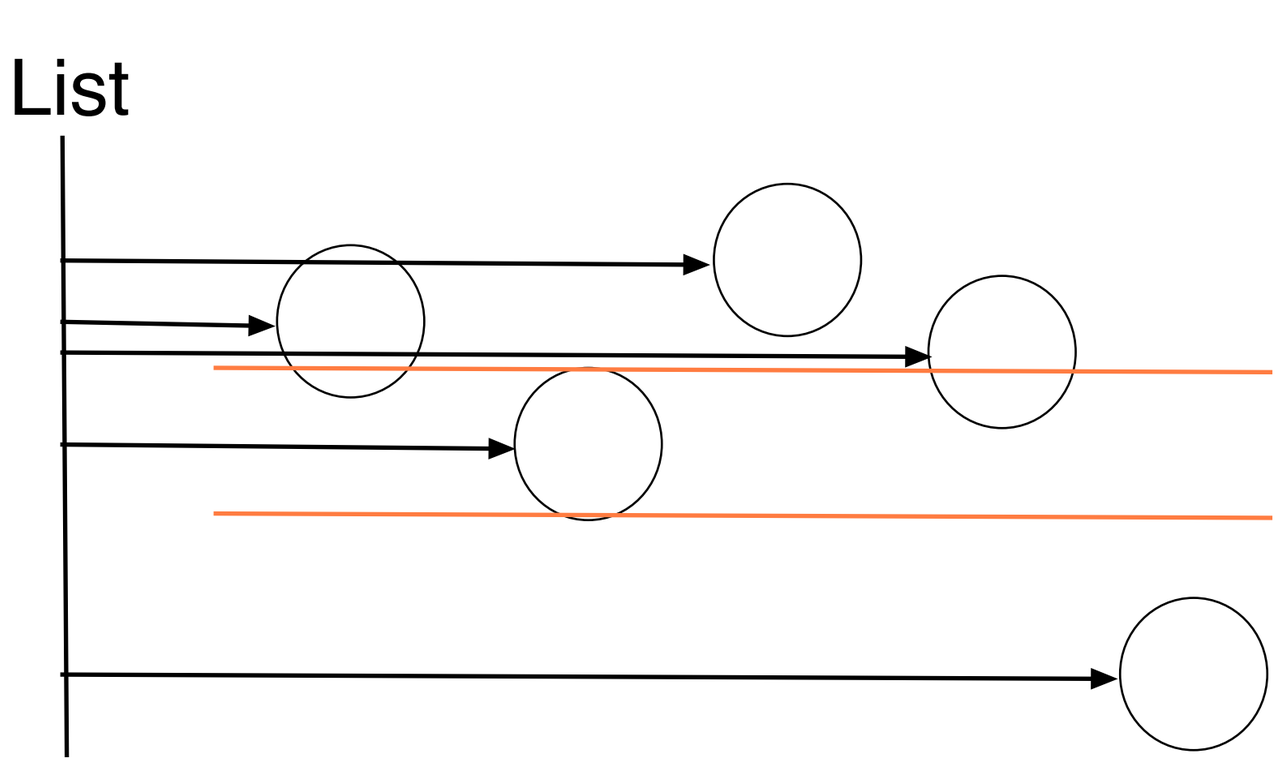
Simple: Linear sorting

More elaborate: Hierarchies, octrees…

Simplify objects out of view?

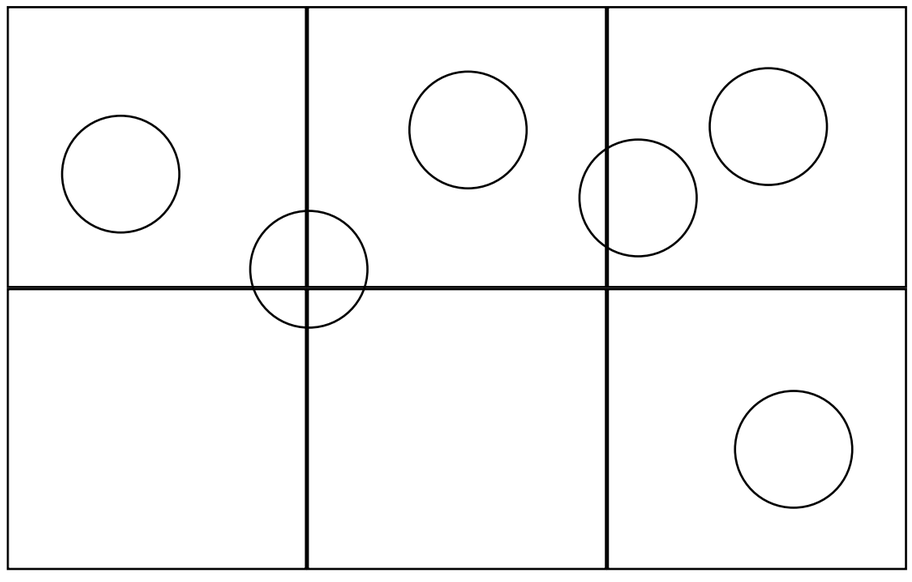
Linear sorting

Reduces the problem by 1 dimension



Subdivision for collision detection

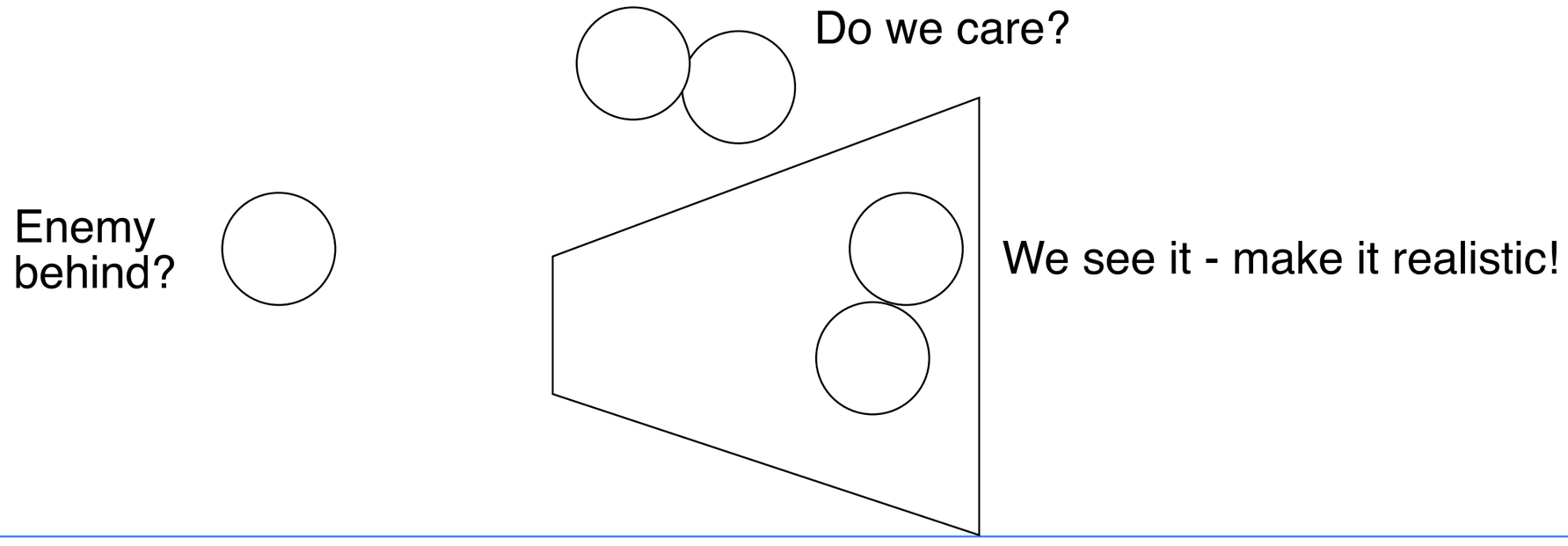
Subdivision by BSP trees, octrees, quad-trees,
regular grid.
Generally useful for most large world problems



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.

Collision handling

What to do once a collision is found.

- Separate
- Change velocities

- Apply torque/rotations
- Deform
- Maintain constraints

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

Simple cases for collision detection

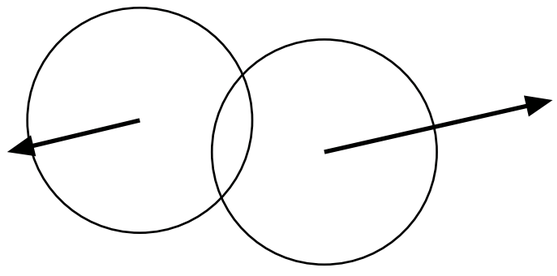
- Collisions between objects with same mass
- Collision with stationary (infinte mass) object
- No rotation (particles, spheres)

Separate

1) Intuitive: Immediately push objects from each other.
May add/change energy.

2) Better: Disregard collisions between objects moving away from each other.

Test with dot product of position and velocity difference.



These two have already collided!

Simple particle physics (again)

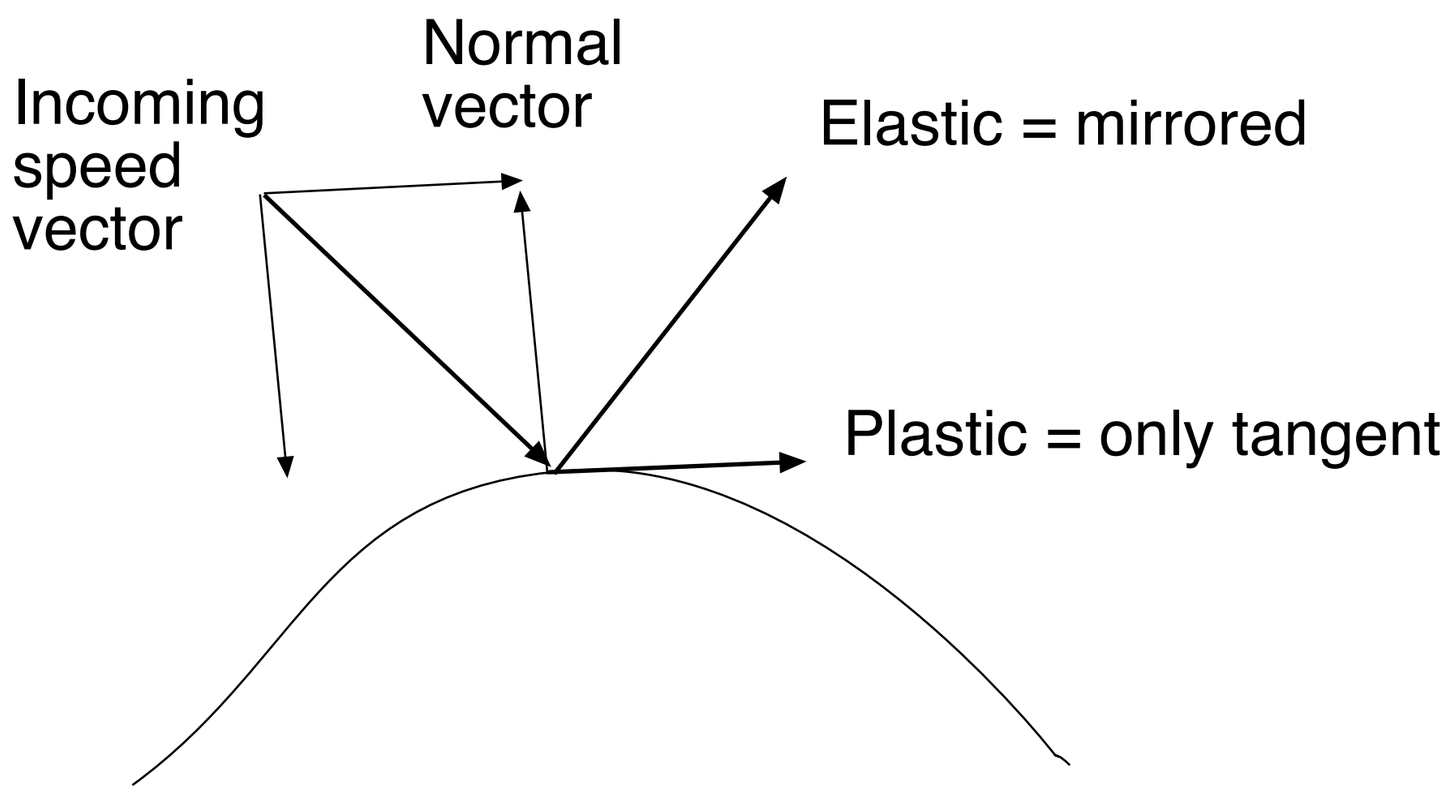
acceleration = gravity + forces/mass
speed = speed + acceleration
position = position + speed

$$a = (g + \sum f/m)\cdot\Delta t$$
$$s = s + a\cdot\Delta t$$
$$p = p + s\cdot\Delta t$$

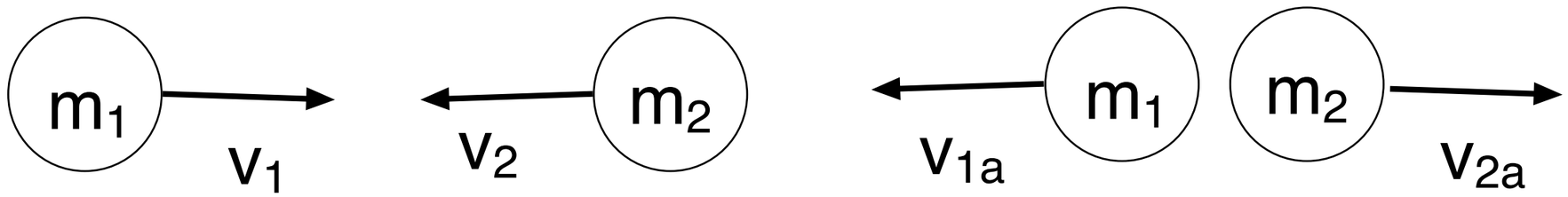
(“Euler integration”)

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

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

Plastic collisions

Two objects with different mass collide.

Very easy to solve:

$$v_{1b}m_1 + v_{2b}m_2 = v_a m_1 + v_a m_2 = v_a(m_1 + m_2)$$

$$v_a = (v_{1b}m_1 + v_{2b}m_2)/(m_1 + m_2)$$

Elastic collisions

Two objects with different mass collide.

Trickier to solve. Solution is:

$$v_{1a} = (v_{1b}(m_1 - m_2) + 2v_{2b}m_2)/(m_1 + m_2)$$

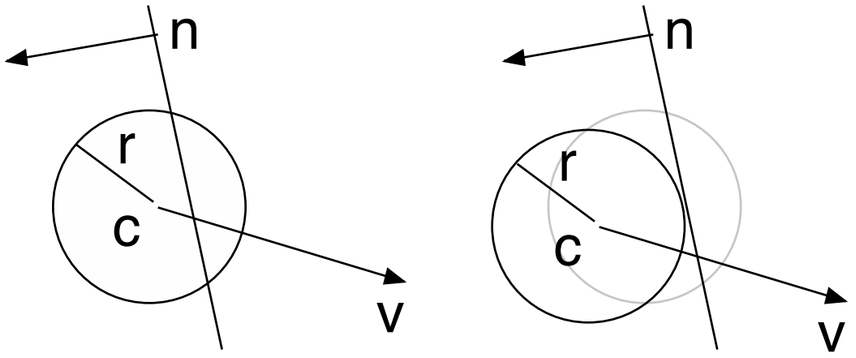
$$v_{2a} = (2v_{1b}m_1 + v_{2b}(m_2 - m_1))/(m_1 + m_2)$$

Add a restitution parameter:

$$v_{1a} = (v_{1b}(m_1 - \varepsilon m_2) + (1+\varepsilon)v_{2b}m_2)/(m_1 + m_2)$$

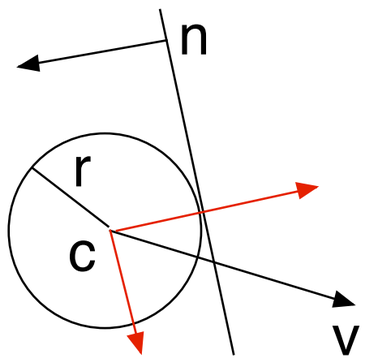
$$v_{2a} = ((1+\varepsilon)v_{1b}m_1 + v_{2b}(m_2 - \varepsilon m_1))/(m_1 + m_2)$$

Collision handling sphere-polyhedra

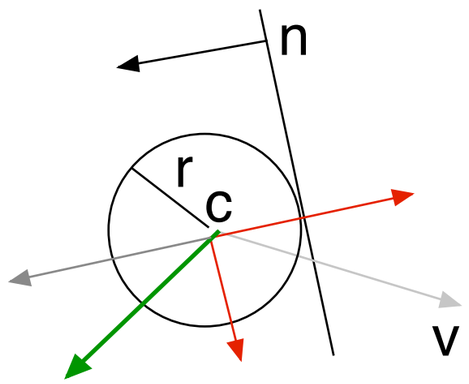


Assuming stationary polyhedra

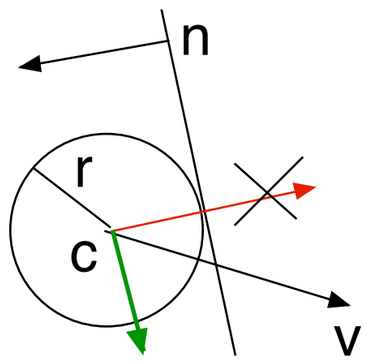
Want to separate? Move object away along normal vector



Split velocity along n

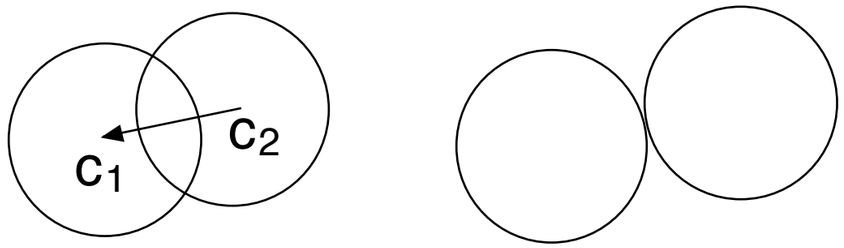


Elastic

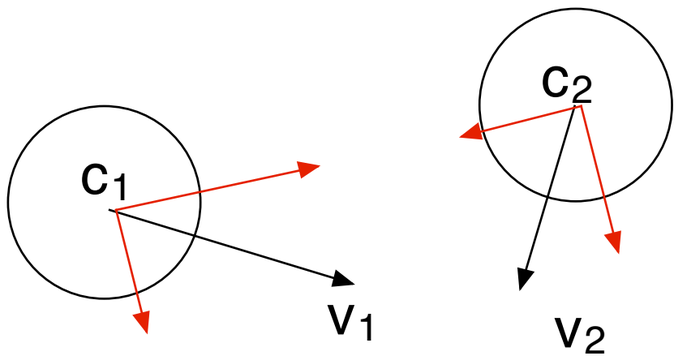


Plastic

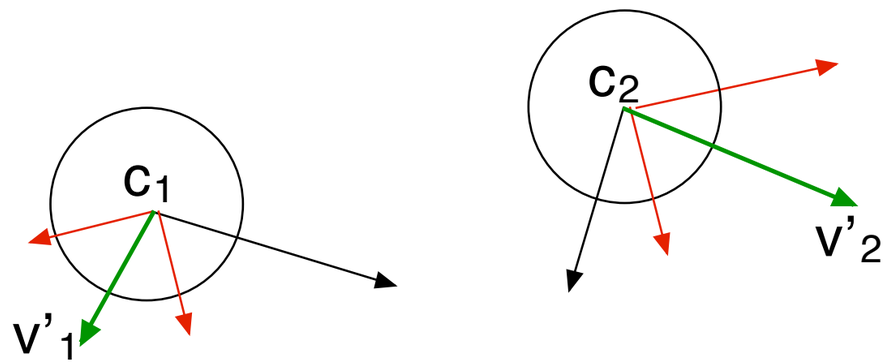
Collision handling sphere-sphere with same weight (simplified, point masses)



Want to 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
Rotations
Better integration
Stacking
Applying forces and backing time to avoid overlap
Deformable bodies
Breakable bodies

Supplement on collision detection and handling

Some points above were corrections/clarifications over the book and there should be a text about it.

The supplement is uploaded to the course page in its first, preliminary form.

Next time: Fractals