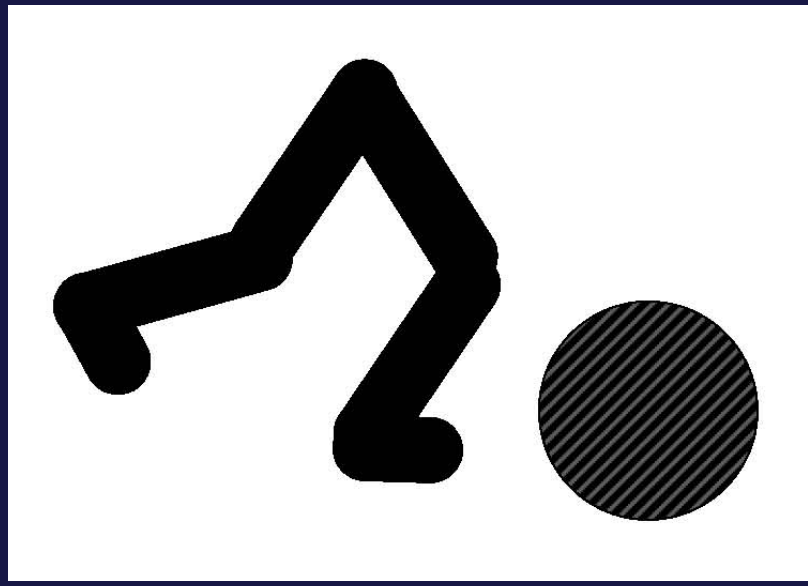
Simulation (last lecture and this one)

How used in games?
Dynamics
Collisions—simple

Collisions—harder
- detection
- bounding boxes
- response

Controllers
Mocap + simulation
User control
What is the future?



Credits

Demos, slides, figures from

- Michiel van de Panne (UBC)
- Michael Mandel's talk at GDC (CMU alum)
- Victor Zordan (UC Riverside)
- UNC for collision detection
- Petros Faloutsos (UCLA)
- Gamasutra
- other web sources as noted

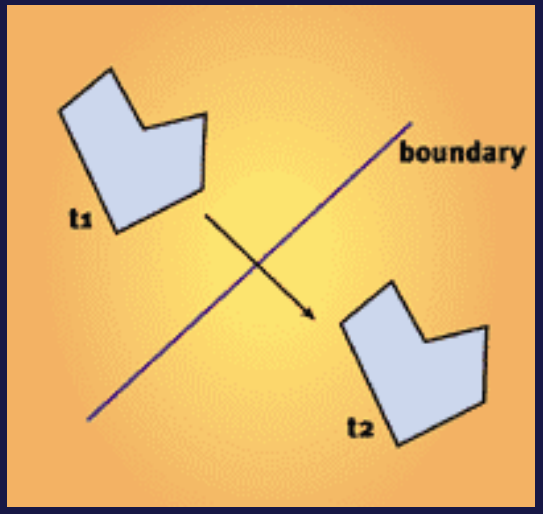
Collision Detection

- Not just points against planes (as last time)
- Bad collision detection has spoiled games
  - Body parts embedded in walls
  - Bullets that shouldn’t have collided or should have
- Modern games have WAY too many polygons to do a naïve n^2 triangle against triangle test

When to test?

Check for intersection with some time step

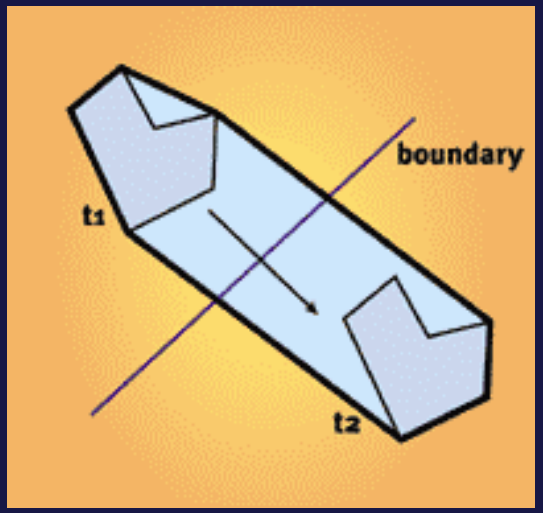
But here testing at time t1 and t2 won't work – we missed the intersection point (tunneling)



http://www.gamespp.com/algorithms/AdvancedCollisionDetectionTechniques1.html

What are our options?

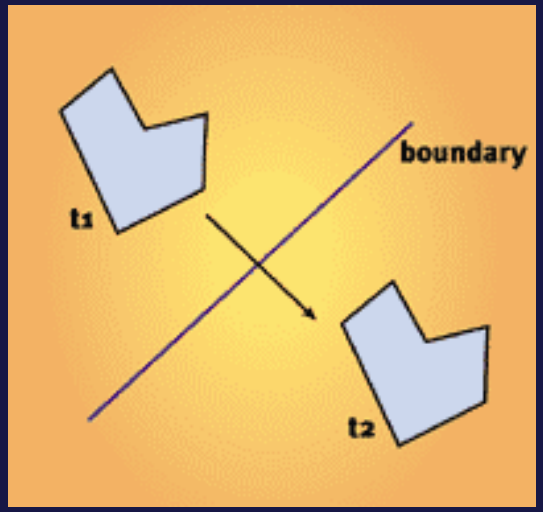
Swept volume: Expensive to calculate (2D poly moving in 3D -> 3D swept volume)



http://www.gamespp.com/algorithms/AdvancedCollisionDetectionTechniques1.html

What are our options?

- Multi-sampling: Subdivide time in half, check for intersections, repeat
- Not a guarantee unless you have a bound or test for the object's velocity
- Sampling rate should allow the object to travel less than one radius between tests



http://www.gamespp.com/algorithms/AdvancedCollisionDetectionTechniques1.html

What are our options?

Multi-sampling:

Actually a bit more complicated—checking linear velocity is not enough.

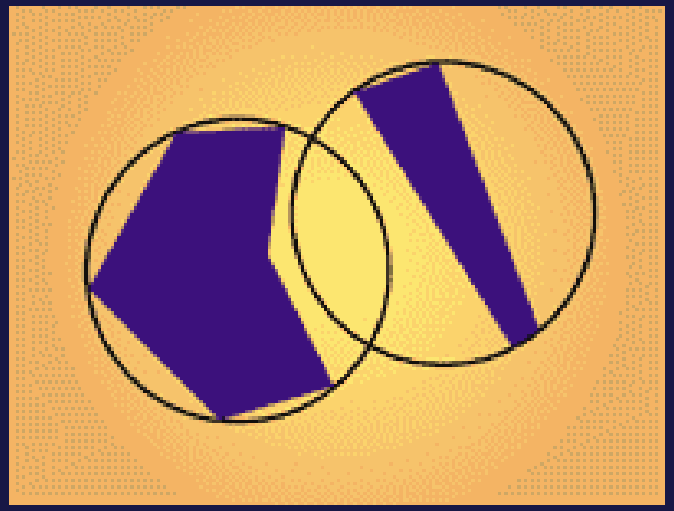
http://www.gamespp.com/algorithms/AdvancedCollisionDetectionTechniques1.html

What about the actual test?

Testing the polygons in one object against all those in another object is too expensive

Use bounding boxes or spheres.

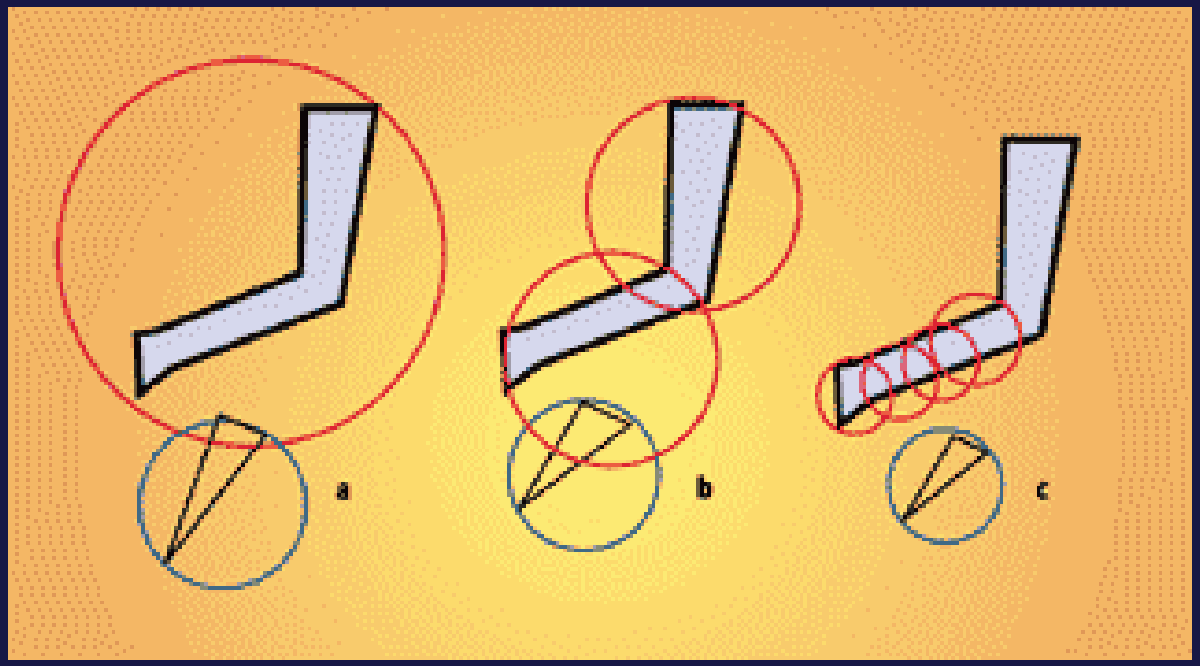
- Cheap: d^2 > dx^2 + dy^2 +dz^2
- Not a very tight fit
- But this has been done as the actual test and probably still is



http://www.gamespp.com/algorithms/AdvancedCollisionDetectionTechniques1.html

Culling with bounding spheres

A better idea: use bounding shapes just to cull not for the final test



http://www.gamespp.com/algorithms/AdvancedCollisionDetectionTechniques1.html

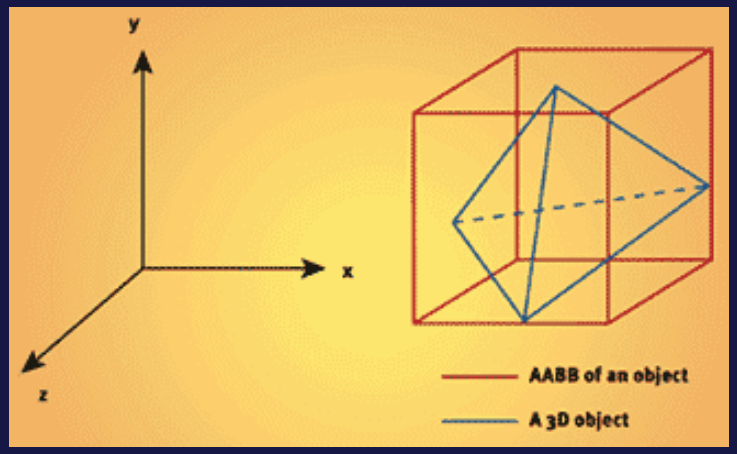
Culling with AABB

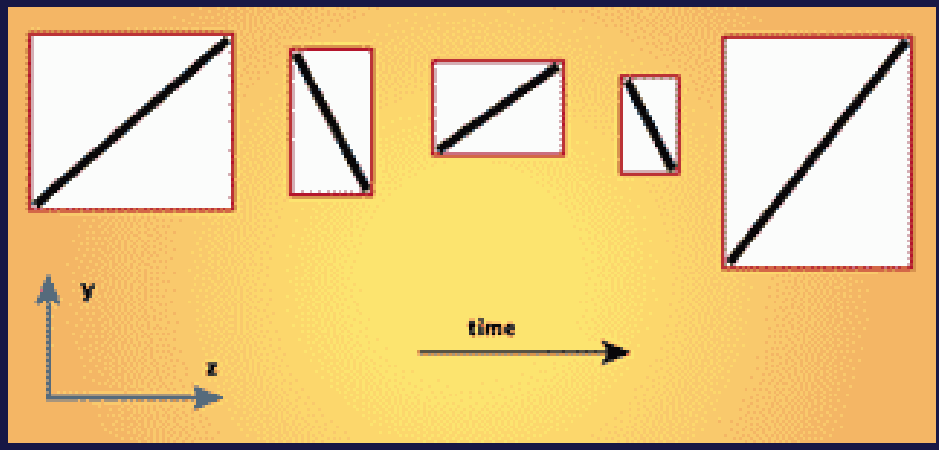
Axis-aligned Bounding Boxes

Why axis-aligned? For speed of test:

If ((p.x >= b.minX && p.x <= b.maxX) && …

But have to recompute on each frame





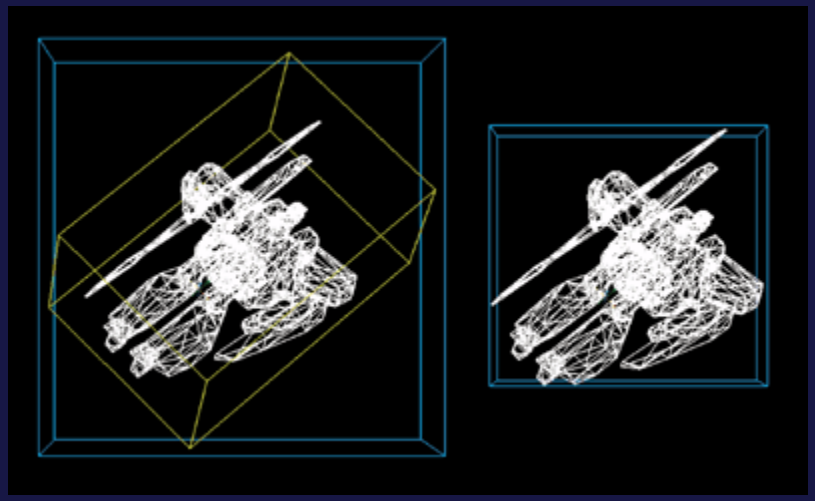
http://www.gamespp.com/algorithms/AdvancedCollisionDetectionTechniques1.html

Culling with AABB

Axis-aligned Bounding Boxes

But have to recompute on each frame

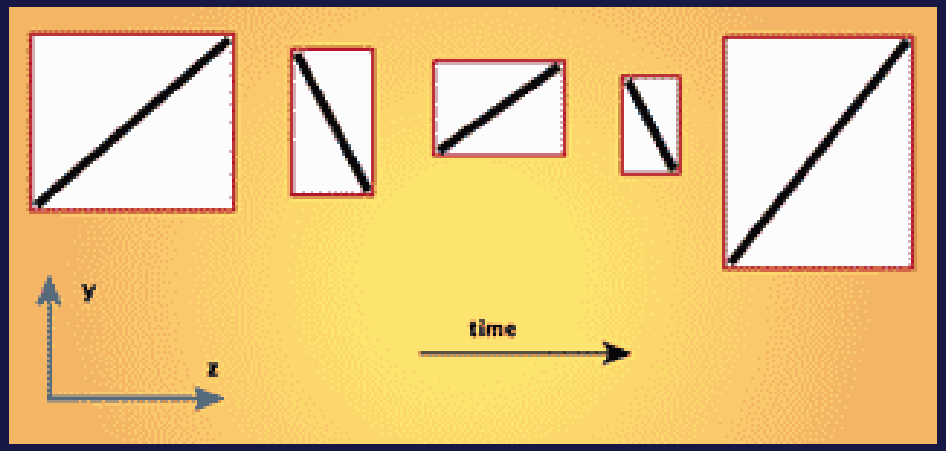
Can cheat by rotating/translating the bbox

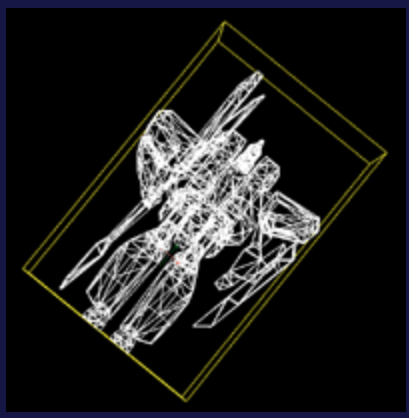


http://www.gamasutra.com/features/20000203/lander_02.htm

Culling with OBB

- Oriented Bounding Boxes
  - Tighter fit
  - Expensive to compute the box (preprocessing)
  - More expensive to test (at run time)

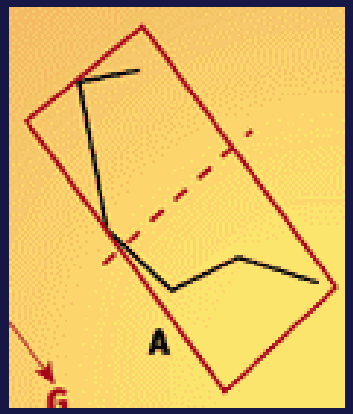




http://www.gamasutra.com/features/20000203/lander_02.htm

OOBs

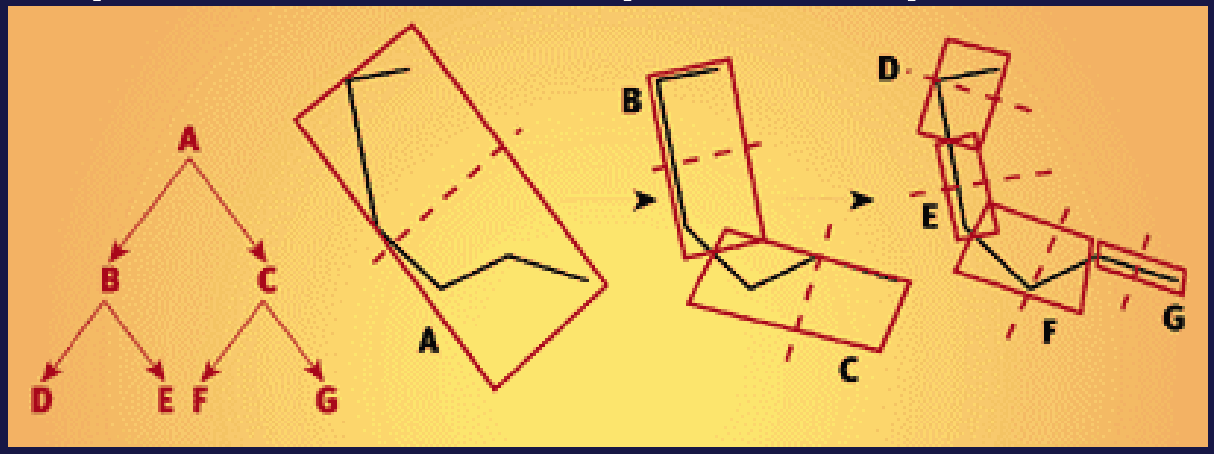
- Compute mean of the distribution of vertices
- Calculate the covariance matrix
- Use eigenvectors of the covariance matrix to align the box with the geometry



http://www.gamespp.com/algorithms/AdvancedCollisionDetectionTechniques1.html

OOB Trees

- Split the box along its longest axis and repeat
- Stop when leaves are triangles or planar polygons (or perhaps earlier)
- A bit expensive but a precomputation step



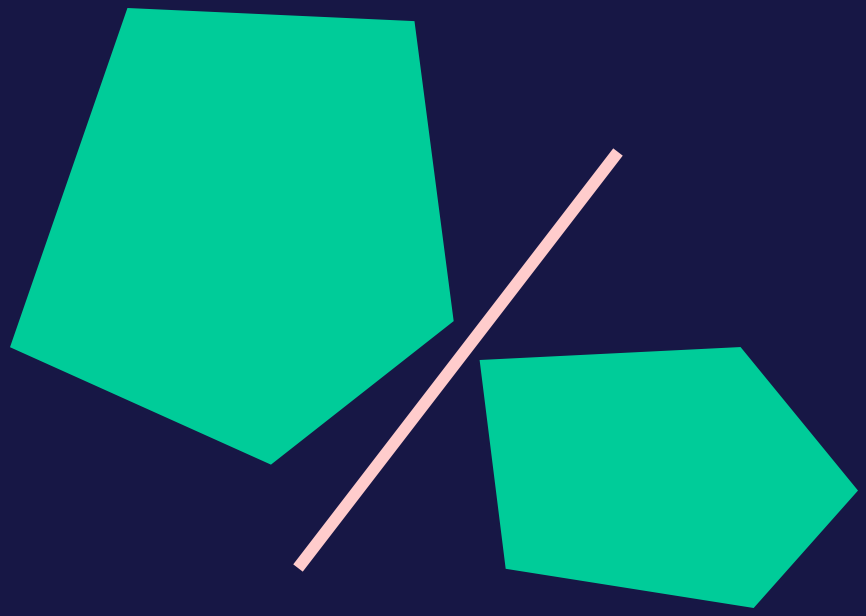
http://www.gamespp.com/algorithms/AdvancedCollisionDetectionTechniques1.html

What kind of BB to pick?

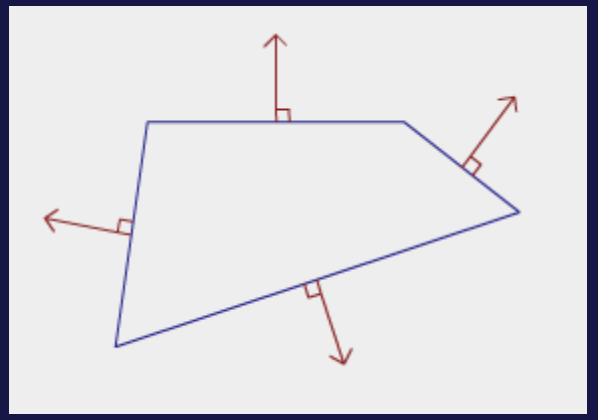
- Depends on the objects and scene
  - What shapes are moving?
  - How are they moving?
  - What is it colliding with? Vertical walls? Arbitrary shaped objects?
  - Deformable objects?
  - What level of accuracy is required for good game play?
- Remember that the graphics card may change the rules too…

http://www.gamespp.com/algorithms/AdvancedCollisionDetectionTechniques1.html

Detecting collisions between polygons



A separating axis exists -> the objects do not intersect

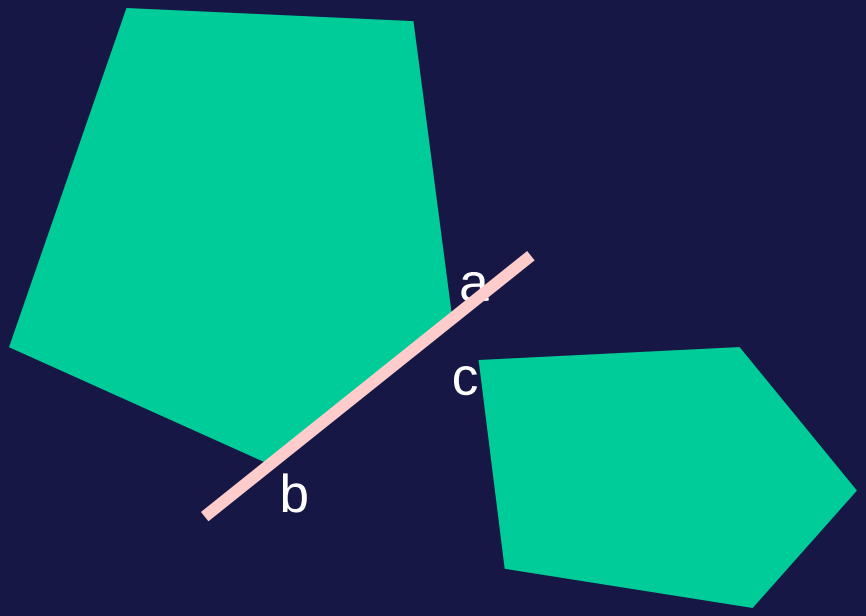


A sufficient set of axes to test in 2D for convex objects

http://www.gamespp.com/algorithms/AdvancedCollisionDetectionTechniques1.html

http://www.harveycartel.org/metanet/tutorials/tutorialA.html (demo)

Detecting collisions between polygons



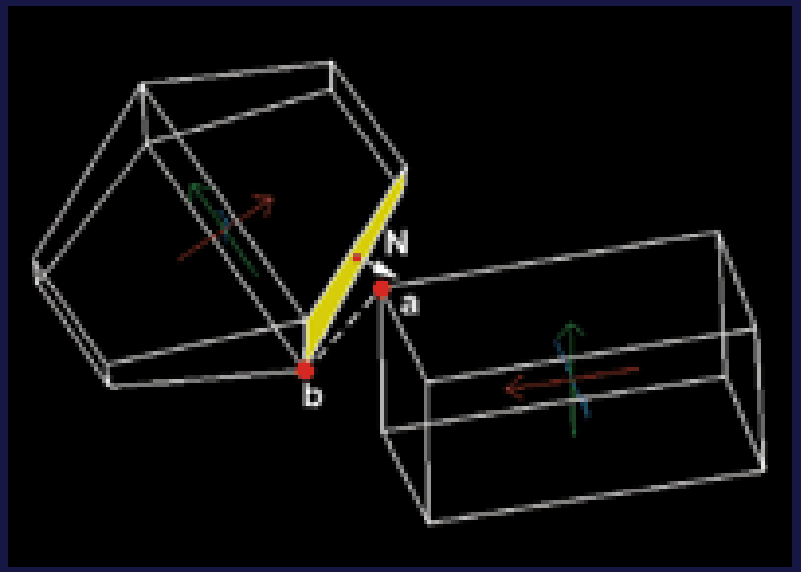
Test each vertex of one poly against the lines of the other.

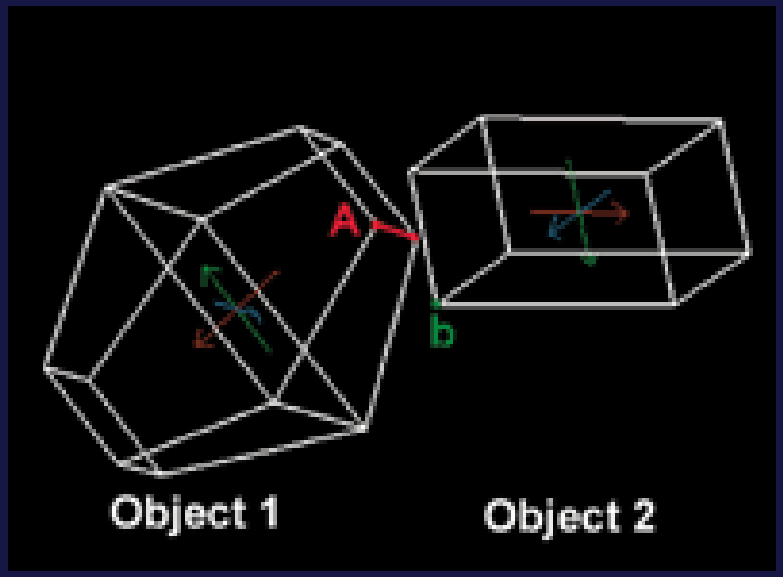
If they are ALL on one side of any of the lines, the polys are not colliding. Store this separating edge for next time for efficiency.

Similar in 3D but with a plane not a line.

http://www.gamasutra.com/features/20000203/lander_02.htm

Detecting collisions between polygons





Similar in 3D but with planes instead of lines.

But none of the faces may be a separating plane even if one exists!

Now need to test planes formed by edge from one object and vertex from another. If none of these separate then we have a collision. Save the plane for next time!

http://www.gamasutra.com/features/20000203/lander_02.htm

What has to collide with what?

- Everything with everything—expensive
- Can divide objects into stationary (walls) and moving (characters, balls, etc.)
- Build a spatial tree (octree) for stationary objects
- Cull further based on knowledge about the game – assume bullets won't collide with each other?

http://www.gamespp.com/algorithms/AdvancedCollisionDetectionTechniques1.html

Trees

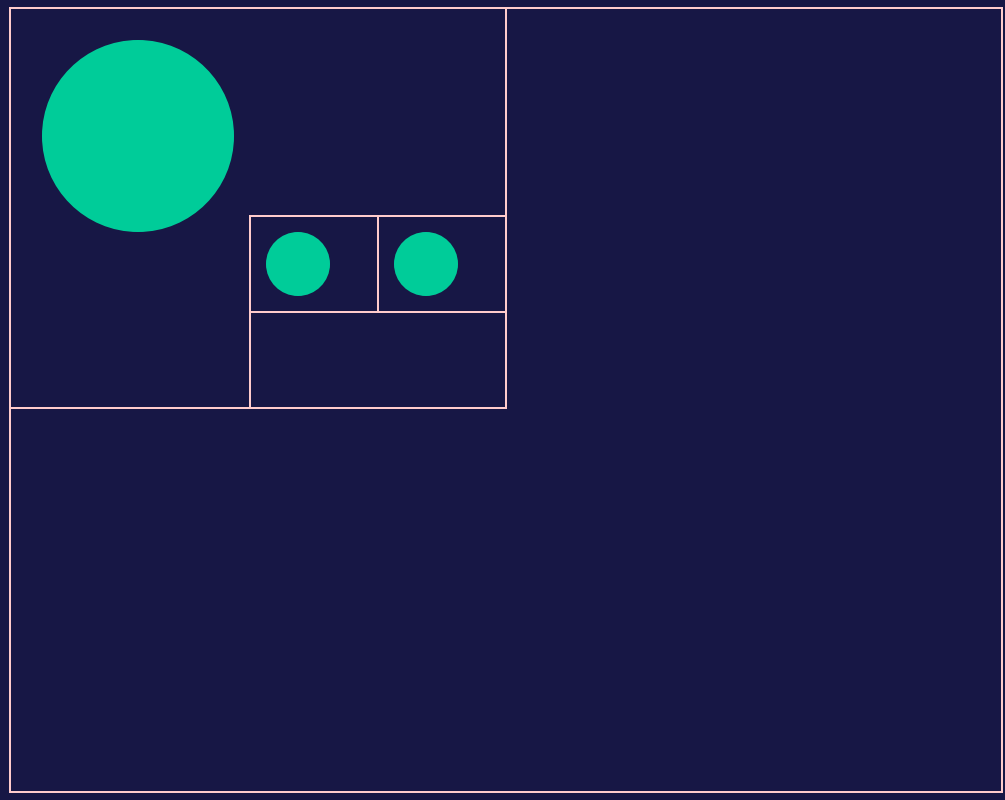
Plane test is easy:

ax+by+cz+d > 0

Point on positive side of plane

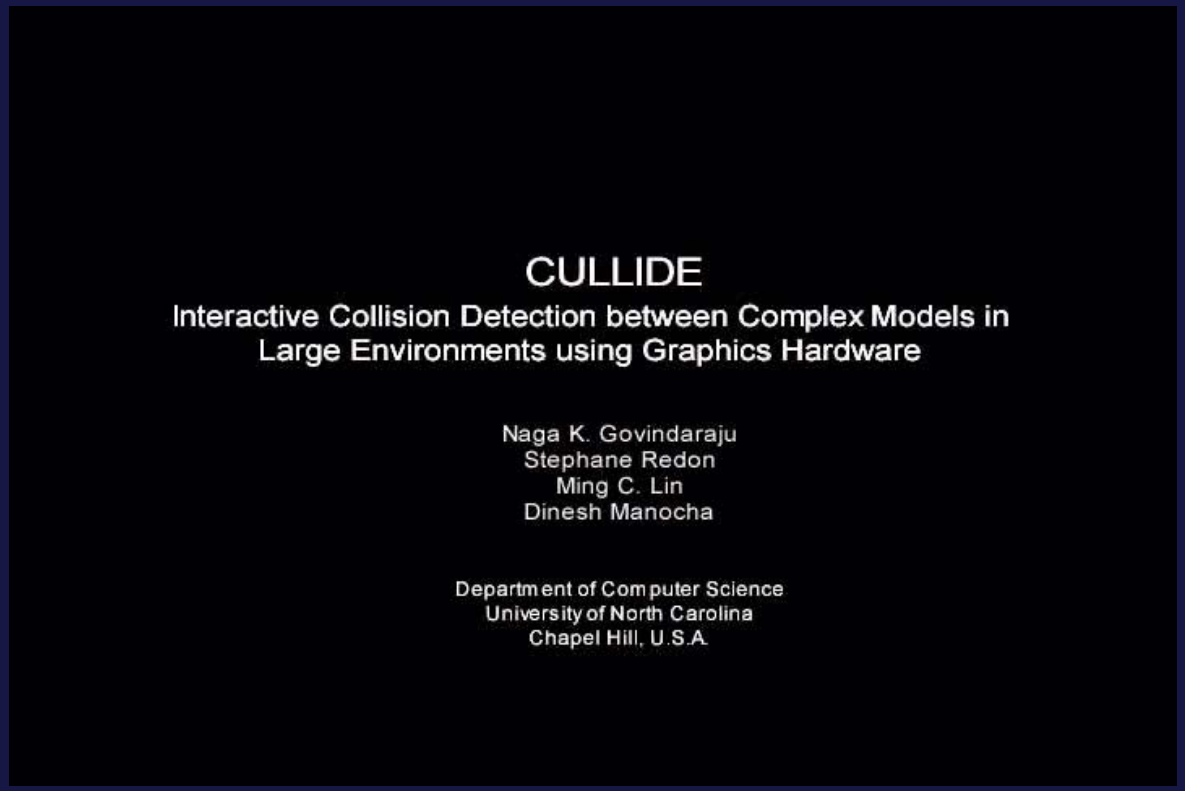
Ax+by+cz+d < 0

Point on negative side of plane



Open problems and new ideas

Use the graphics hardware (UNC) to detect collisions – algorithm animation



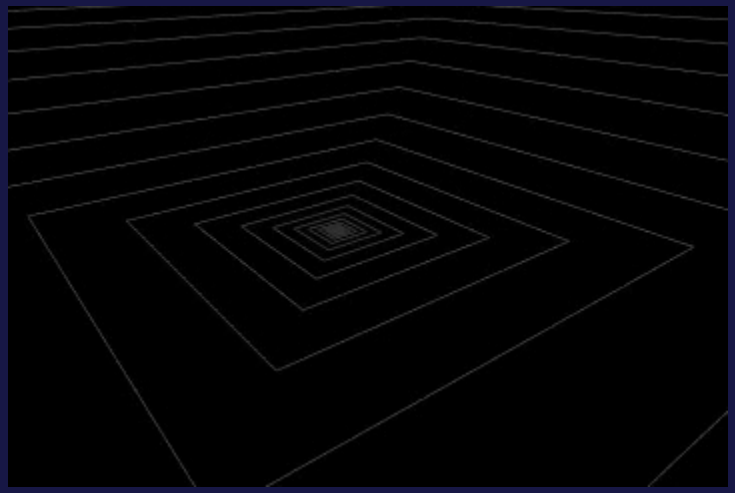
Open problems and new ideas

Use the graphics hardware (UNC) to detect collisions—results

Open problems and new ideas

Deformable objects?

- all this goes out the window
- precompute expected deformations?



Collision Response

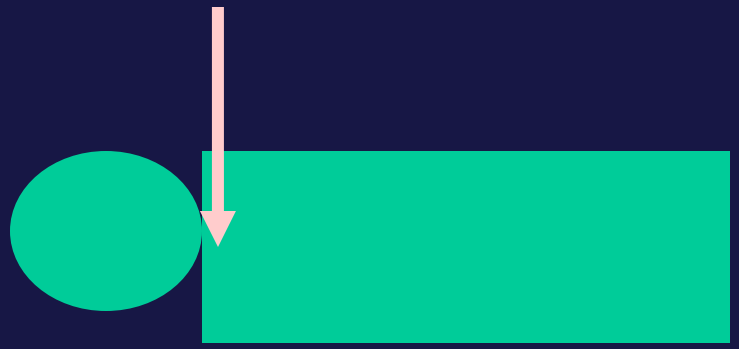
- Forces/accelerations: penalty method
  - Springs to push objects out of collisions
- Velocities: impulse-based
  - Instantaneous change in velocity to prevent collision
- Projection: positions
  - Teleportation to remove collision (shortest distance)
- All of the above require more info than that two objects are colliding – direction of collision, etc.
- Destroy one or both of the objects…

Projection

- Teleportation to remove collision
- Shortest distance?
- Direction that it entered?

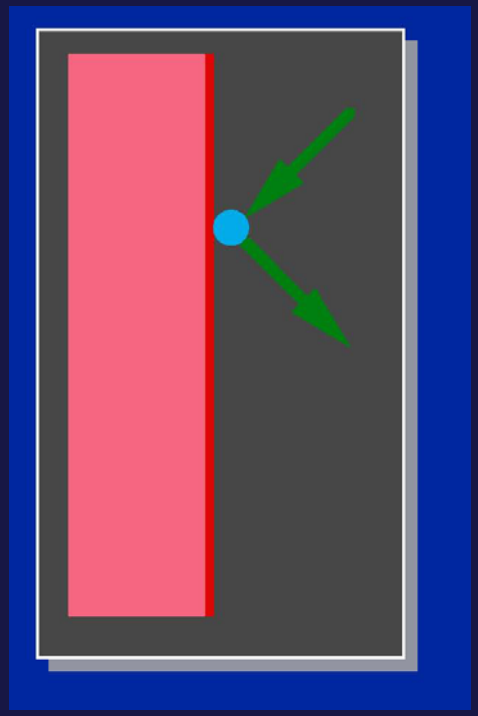
Projection

- Teleportation to remove collision
- Shortest distance?
- Direction that it entered?
- Not so easy in the general case



Impulse-based

Instantaneous change in velocity



Acceleration or Penalty Methods

Each contact is classified as a collision or a resting contact

if (relative velocity * normal > [magic threshold]) collision

otherwise, resting contact

Two linear potentials are created. One acts along normal to repel objects, the other acts on the tangent plane to represent friction.

Nonlinear force along contact normal is approximated by a piecewise cubic spline

Friction cone is approximated by a piecewise cubic patch

Original Slides from David Wu

Equations

Resting contact normal force:

- Fn = L* (K*deltaz + kd * deltazd)
- Deltaz is velocity into the surface

Collision normal force:

- kd is 0
- K is large

in both cases:

- L is a piecewise cubic spline that smoothly interpolates (e.g. with continuity of G^2)
  - 0 when |f| <= 0
  - 1 when |f| >= [insert magic threshold]

continued

Friction forces:

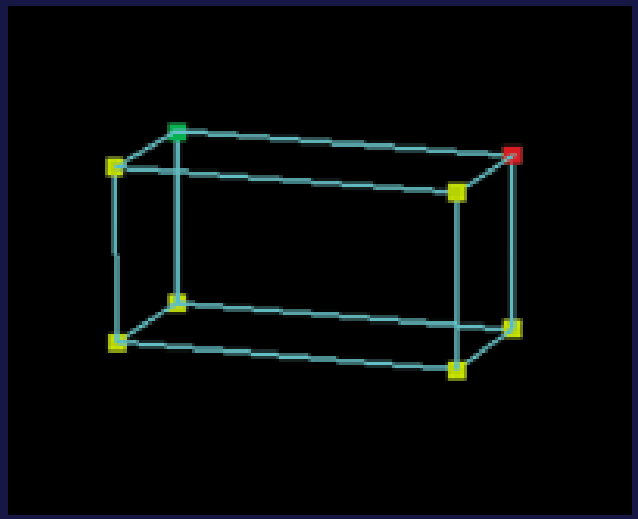
- f_f = L * k2*(velocity along the surface)
- L is a piecewise cubic spline that smoothly interpolates (e.g. with continuity of G^2)
  - 0 when |f| <= [insert small magic threshold]
  - 1 when |f| >= [insert big magic threshold]

L is smooth to not give the integrator fits

Back to simulation…

We can easily combine particles with springs to get cloth or jello

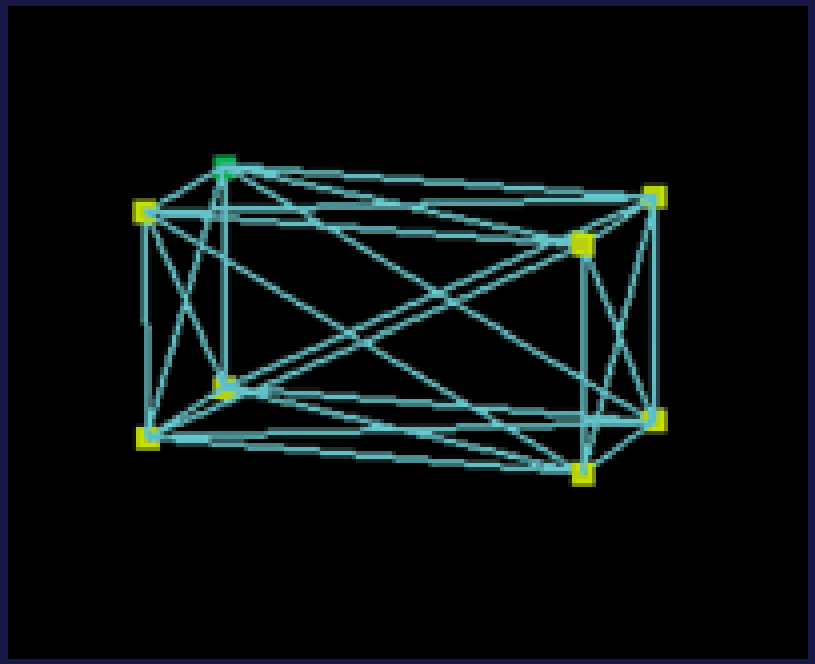
What happens if you simulate this?



http://www.gamasutra.com/features/20000208/lander_pfv.htm

Back to simulation…

Instead use this…



http://www.gamasutra.com/features/20000208/lander_pfv.htm

Control

Much like springs and dampers of pure simulation but now you (the controller) can pick the setpoint!