ANIMACIÓN POR ORDENADOR

Tema 6

Captura de movimiento.
Animación basada en físicas. Sistemas de partículas. Simulación de cuerpos rígidos. Colisiones

CONTENIDO

1. Captura de movimiento.
2. Animación basada en físicas.
3. Sistemas de partículas.
4. Simulación de cuerpos rígidos.
5. Colisiones.

RIGID BODY SIMULATION

Various forces to be simulated are modeled.

When the forces are applied to objects, they induce

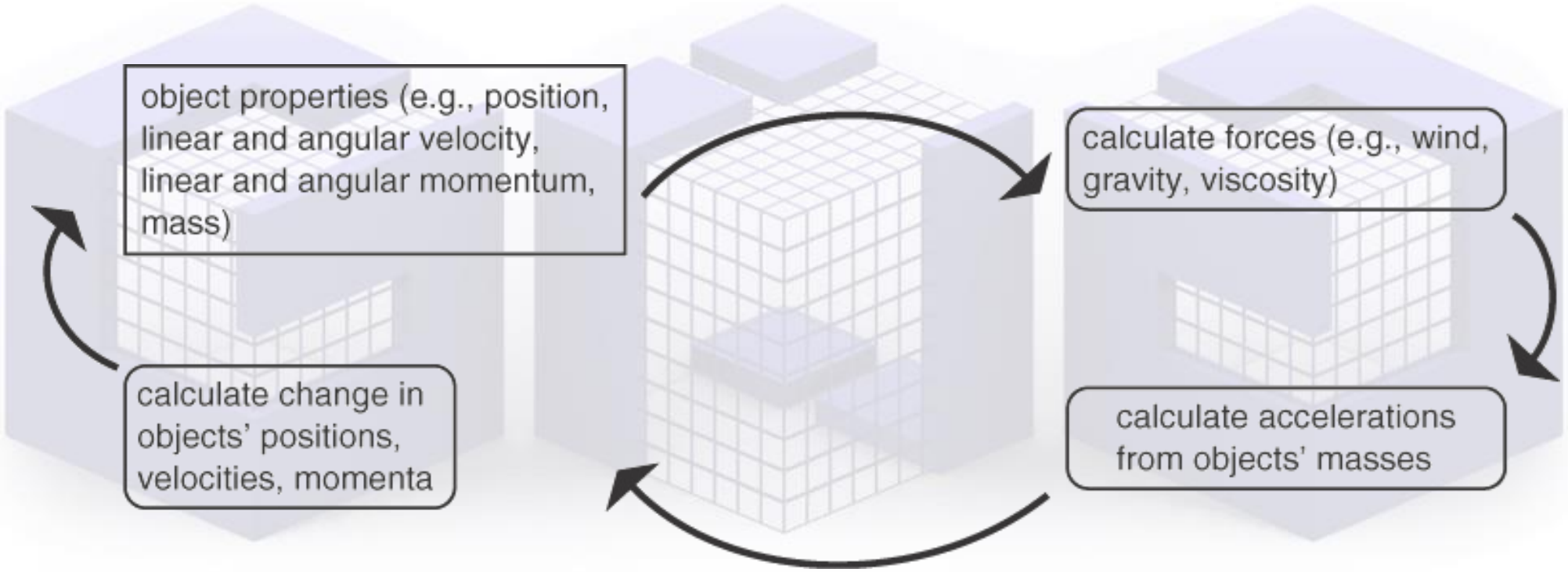
- linear acceleration (based on object's mass)
- angular acceleration (based on mass's distribution)

These accelerations are integrated over a delta time step to get changes of object's velocities, which in turns integrated over a delta time step to produce changes in position and orientation.

Read Witkin and Baraff's SIGGRAPH'01 course notes:
Physics-based modeling

http://www.pixar.com/companyinfo/research/pbm2001/index.html

RIGID BODY SIMULATION: UPDATE CYCLE



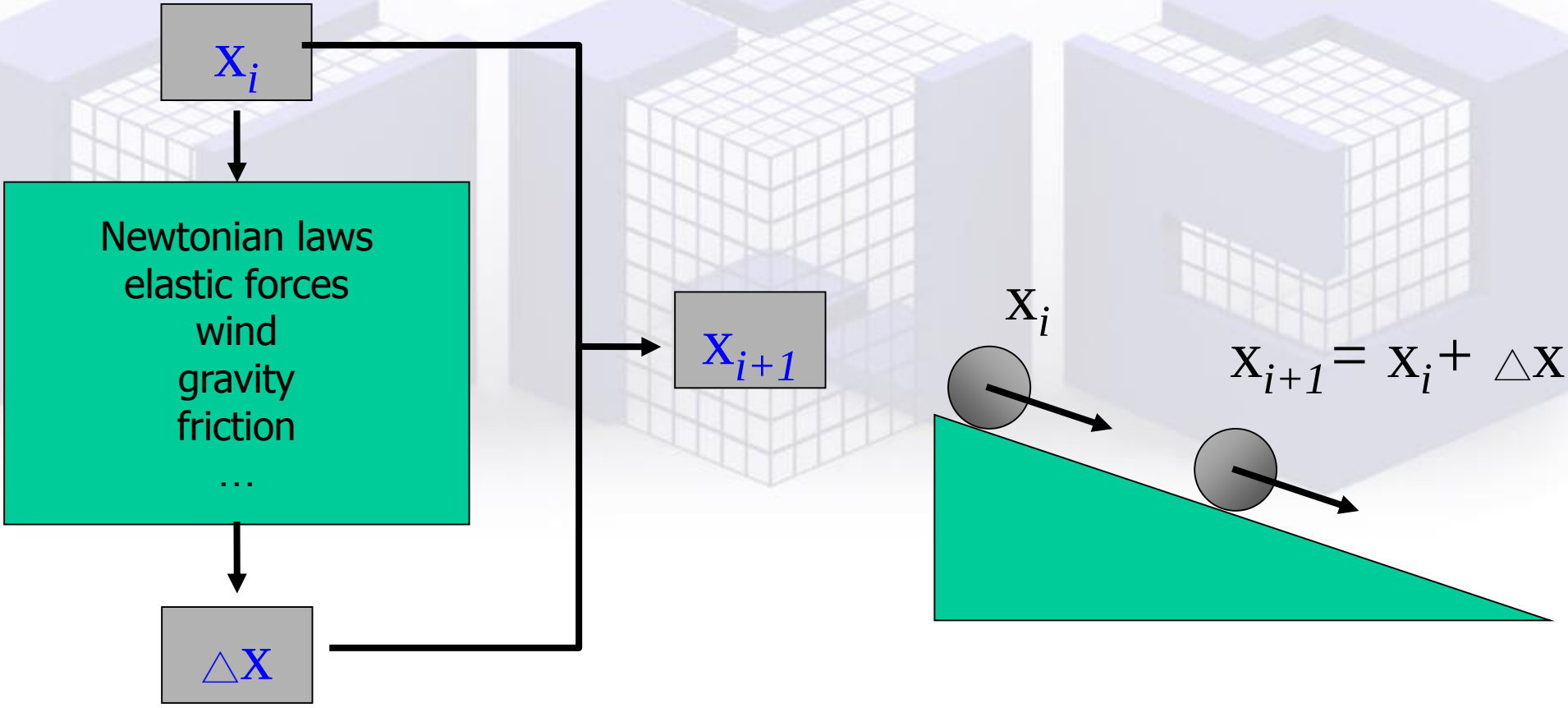
RIGID BODY SIMULATION

Continuous process vs. discrete time simulation

1. Assume acceleration is constant over the delta time step
2. These accelerations are integrated over a delta time step to get changes of object's velocities, which in turns integrated over a delta time step to produce changes in position and orientation.
3. How to update?
    - Euler integration method
    - Runge –Kutta method: Second-order in magnitude of error term

PHYSICS-BASED SIMULATION

A procedure that generates a sequence of the states of a system based on physics laws



BODIES IN COLLISION

Objects moving in a dynamic scene

- Objects in collision – need to do the detection
- Objects sliding against on other object
- Objects resting on each other

Need to calculate forces for computing accurate reaction

BODIES IN COLLISION

Ultimate goal in VR

Real and virtual objects need to behave like real ones.

At a minimum

Objects should not pass through each other, and things should move as expected when pushed, pulled, or grasped.

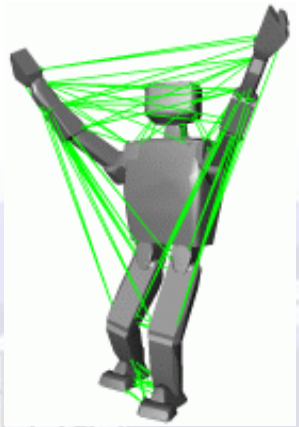
Overall

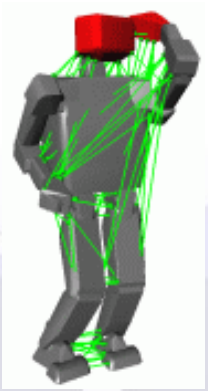
Physical simulation in VR must run reliably, seamlessly, automatically, and in real time.

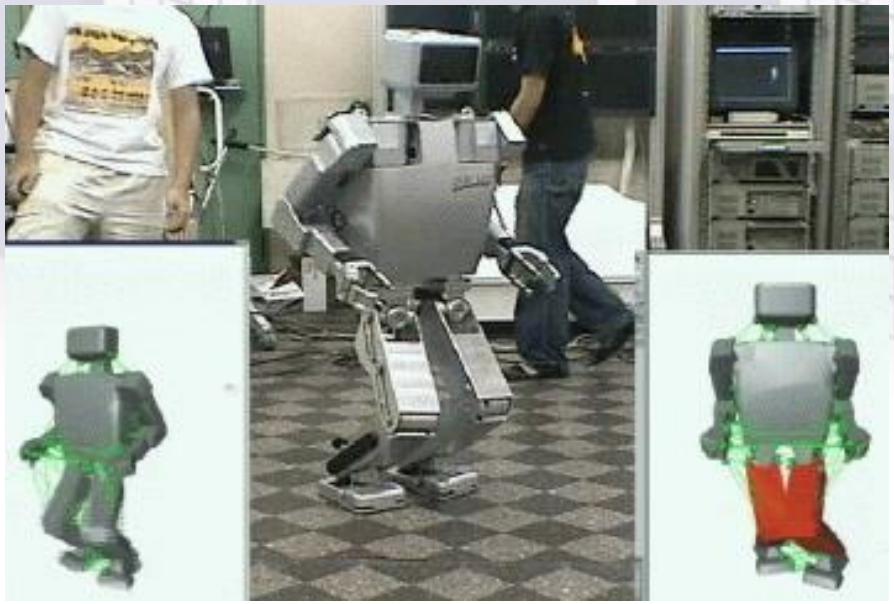
COLLISION DETECTION: COMPUTER ANIMATION

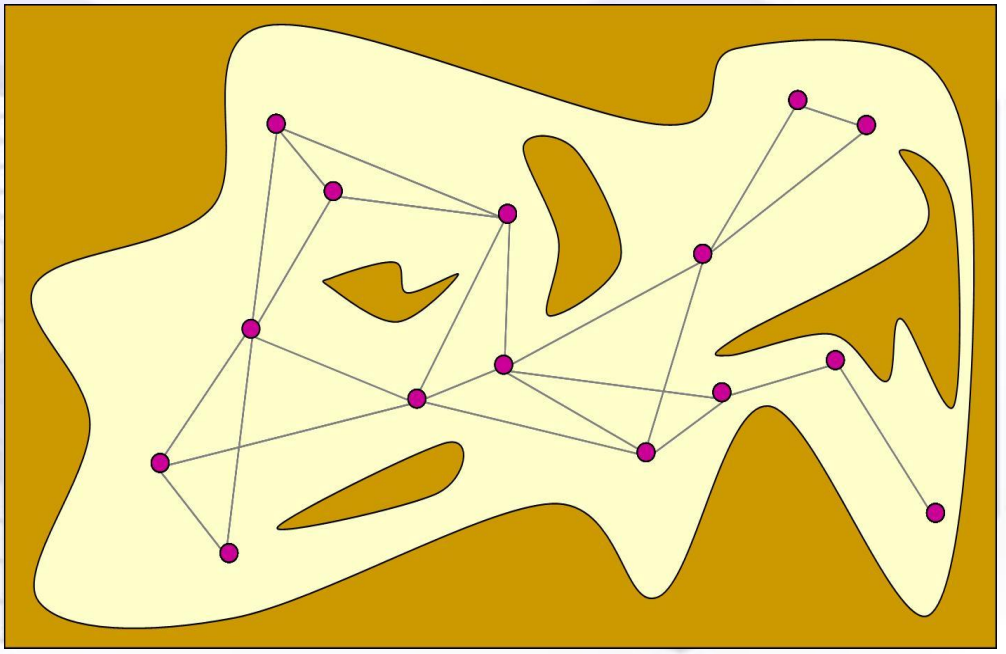
COLLISION DETECTION: HAPTIC INTERACTION

COLLISION DETECTION: MOTION/PATH PLANNING

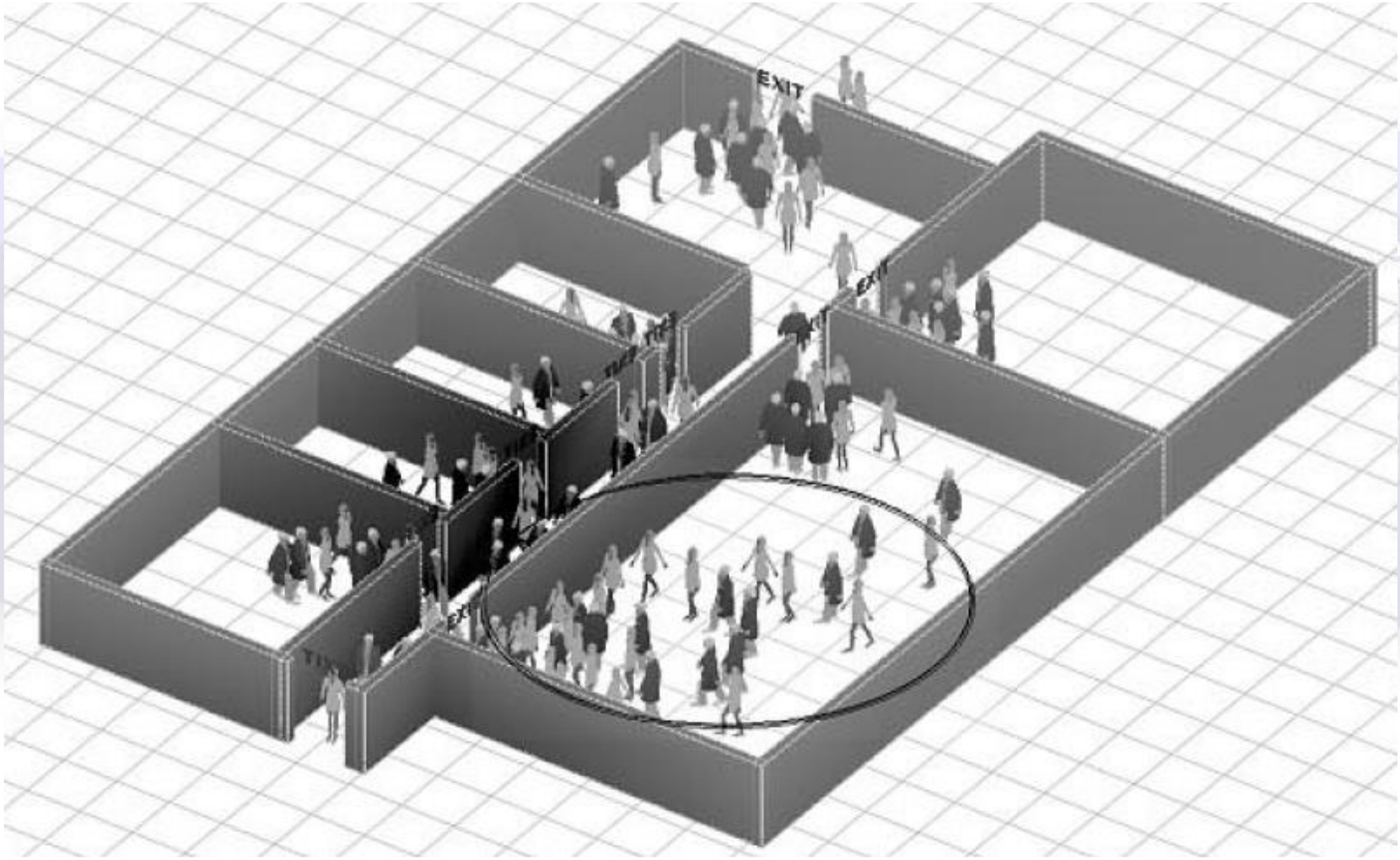








COLLISION DETECTION: CROWD SIMULATION



BODIES IN COLLISION

To prevent interpenetration

Collisions must be detected.

Velocities must be adjusted in response to collisions.

Collision response must be computed.

- If the collision response does not cause the objects to separate immediately, contact forces must be calculated and applied until separation finally occurs.

PROBLEMS OF COLLISION DETECTION

A naive collision detection has

- Fixed-timestamp weakness
- All-pair weakness
- Pair-processing weakness

Features that current methods have

- Interactive rate
- Handle polygon soaps
- Models can undergo rigid-body motion
- Provide well-fit bounding volume
- Collision detection only occurs at discrete times.

PERFORMANCE OF A NAIVE METHOD

All-pair weakness

If the scenario contains n moving objects and m static objects, total object test for each frame is

Pair processing weakness $nm + \binom{n}{2}$

Note:

Performance evaluations for collision detection are extremely difficult since algorithms are sensitive to the actual scenarios, and there is no algorithm that performs best in all cases.

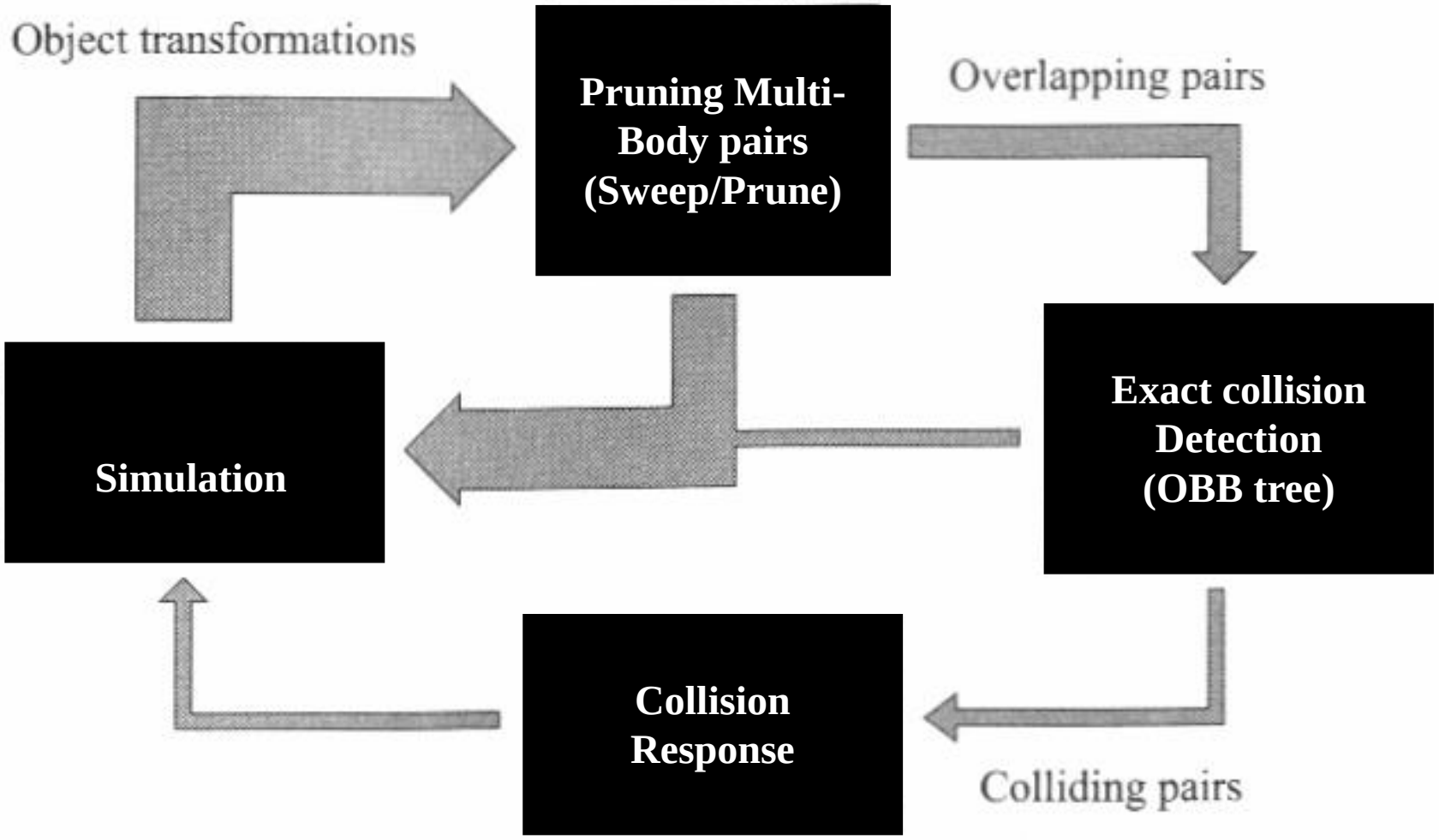
SPEED UP METHODS

- Bounding volume for object level and polygon level.
  - Pure bounding volume
  - Hierarchical bounding volumes
    - Support progressive collision detection.
- Spatial coherence
  - Usually large regions of the space are occupied by only one object or none at all. If some objects share the same region, they are likely intersecting.
- Time coherence
  - Moving objects usually move on a continuous path. Results of earlier collision query can be exploited.

TWO-LEVEL CD SYSTEM

- Globally searching pairs of objects that have potential collision.
  - Spatial subdivision
  - Sweep-and-Prune
- Collision detection on a pair of objects
  - Hierarchical CD based on OBBTree, AABBTree,…
  - Polygon-polygon overlap test

TWO-LEVEL CD SYSTEM



SPACE PARTITION

- Spatial coherence is usually exploited by partitioning the space.
  - All-pair weakness can be resolved by the use of spatial coherence.
- Space partition methods differ in fast neighbor-finding (for static environments) and fast updating for moving objects.
  - Grid (uniform partition)
  - Octree
  - Binary space partition

TEMPORAL COHERENCE

ONE OR TWO-DIMENSION SWEEP AND PRUNE

Sorting objects in 3-D space by dimension reduction

- If two objects collide in a 3D space, their orthogonal projections onto the xy, yz, and xz-planes and x, y, and z-axes must overlap.
- Project bounding volume of objects.

Based on Axis-aligned bounding boxes (AABB).

- Fixed-size bounding cube: large enough to contain the object at any orientation.
- Dynamically-resized rectangular bounding boxes

ONE OR TWO-DIMENSION SWEEP AND PRUNE

Project each 3D box onto x, y, and z axes.

Construct 3 lists, one for each dimension.

- Each list contains the values of the endpoints of the projected interval corresponding to that dimension.

Determine which intervals overlap by sorting these lists.

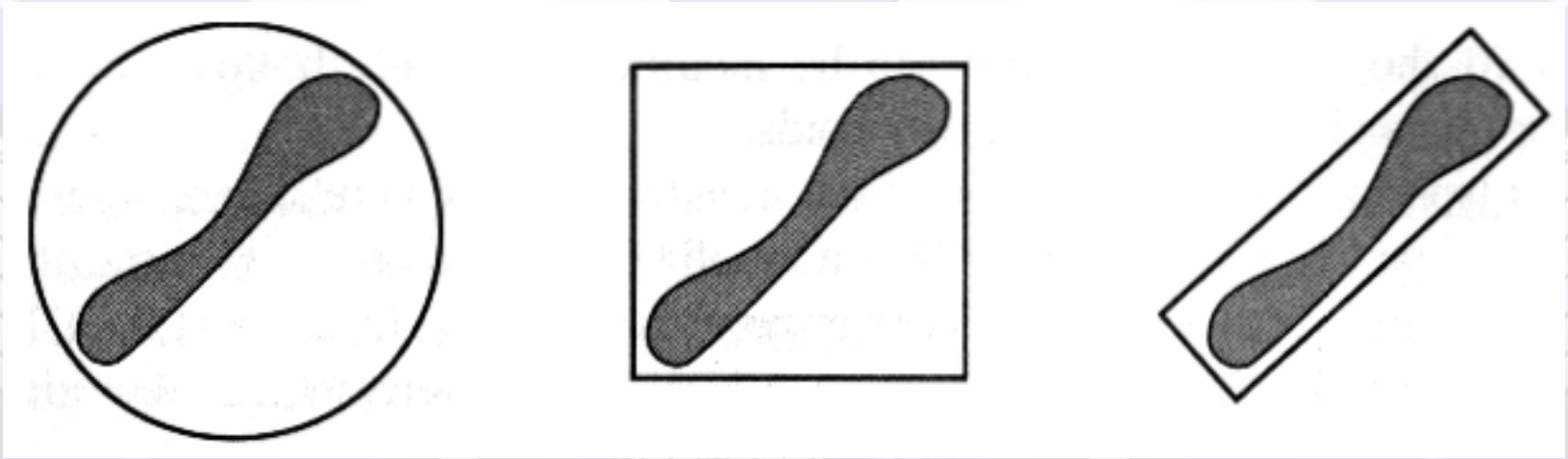
- Utilize temporal or frame coherence by changing only the values of the interval endpoints.
- Bubble sort or insertion sort work well for previously sorted lists.

BOUNDING VOLUMES

Bounding volume is used to do a pre-check before doing any further collision detection.

Bounding sphere (BS) overlapping check, not a tight fit.	Fast
Axis-aligned bounding box (AABB) check, a better tight fit.	Fast overlapping
Oriented bounding box (OBB) expensive overlapping test.	Tight fit, but with relatively
K-DOPS (direct oriented polytopes) axes	OBB + more cutting

BOUNDING VOLUMES



HIERARCHICAL BOUNDING VOLUMES

Hierarchical bounding volume is useful

- For efficiently resolving the pair-processing weakness, and
- Providing a basis for the hierarchical or time-critical scheme for collision detection.

Three types:

- Hierarchical Axis-aligned bounding box
- Hierarchical Oriented bounding box
- Hierarchical Bounding sphere

GENERAL HIERARCHICAL COLLISION DETECTION SCHEME

Common denominators of these schemes

- A hierarchical bounding volume is built for each object.
- High-level code for a collision query is similar, regardless of the BV type used.
  - BV-BV overlapping tests and primitive-primitive overlapping tests are different depending on what BVs and primitives are used.
- A simple cost function can be used to trim, evaluate, and compare performance.

HIERARCHICAL COLLISION DETECTION

HIERARCHY BUILDING

Common hierarchy used:

- *K*-ary tree, where each node may at most have *k* children.
- At each internal node, there is a BV enclosing all its children.
- At each leaf, there are one or more primitives.

Three ways of building a hierarchy

- Bottom-up
- Incremental tree insertion
- Top-down

HIERARCHICAL COLLISION DETECTION

BOTTOM-UP HIERARCHY BUILDING

Start

Starts by combining a number of primitives and finding a BV for them.

Grouping

This BV is grouped with one or more BVs constructed in a similar way, thus yielding a new, larger parent BV.

Recursive grouping

Repeat the grouping until only one BV exists, which becomes the root of the hierarchy.

HIERARCHICAL COLLISION DETECTION

HIERARCHY BUILDING BY INCREMENTAL TREE-INSERTION

Start with an empty tree

All other primitives and their BVs are added one at a time to the tree.

- To make an efficient tree, an insertion point should be selected so that the total tree volume increase is minimized.

Little is known about this scheme in the context of collision detection.

HIERARCHICAL COLLISION DETECTION

TOP-DOWN HIERARCHY BUILDING

Start

- Find a BV for all primitives of the object, which is then acts as the root of the hierarchy.

Recursively apply a divide-and-conquer strategy

- First to split the BV into k or fewer parts and find all included primitives for each such part. A BV is then created for each part.

Potential advantage

- A hierarchy can be created on an as-needed basis, i.e., we can construct the hierarchy for those parts of the object where it is actually needed.
- Used by majority of hierarchical CD.

HIERARCHICAL COLLISION DETECTION

K-ary tree

K =2 minimizes the work to be done when traversing a path from root to the worst-case leaf, and it is easier to compute the hierarchy than the case of higher *k*.

Higher *k* gives a tree with lower height, but it requires more work at each node.

HIERARCHICAL COLLISION DETECTION

Goals

Find tight-fitting bounding volume.

Find hierarchy construction that creat balanced and efficient tree.

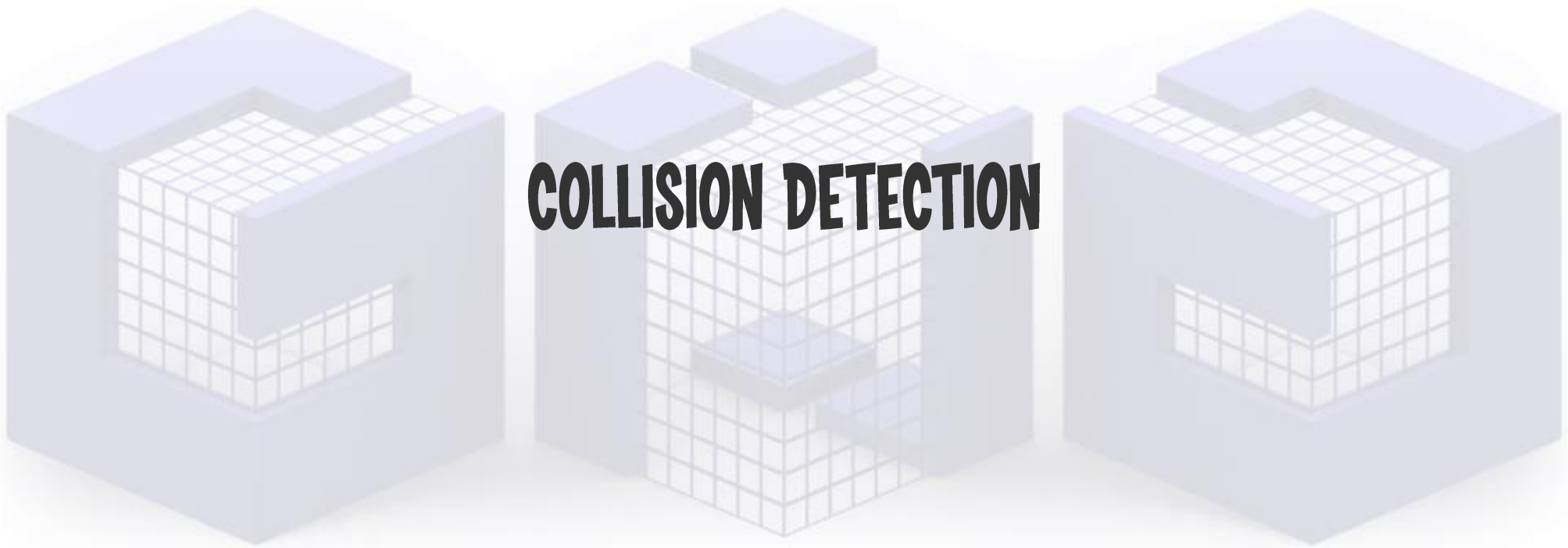
Efficient BV-BV overlapping test.

Efficient primitive-primitive overlapping test.

Notes

Balanced trees are expected to perform best in all cases, since the time of CD query will not vary.

But, it does not mean that it is best for all inputs, e.g., those parts that seldom or never be queried for a collision can be located deep in the hierarchy.



COLLISION DETECTION

COLLISIONS

Collision Detection

- Collision detection is a geometric problem
- Given two moving objects defined in an initial and final configuration, determine if they intersected at some point between the two states

Collision Response

- The response to collisions is the actual physics problem of determining the unknown forces (or impulses) of the collision

COLLISION DETECTION

‘Collision detection’ is really a geometric intersection detection problem

Main subjects

- Intersection testing (triangles, spheres, lines…)
- Optimization structures (octree, BSP…)
- Pair reduction (reducing N2 object pair testing)

INTERSECTION TESTING

General goals: given two objects with current and previous orientations specified, determine if, where, and when the two objects intersect

Alternative: given two objects with only current orientations, determine if they intersect

Sometimes, we need to find all intersections. Other times, we just want the first one. Sometimes, we just need to know if the two objects intersect and don’t need the actual intersection data.

PRIMITIVES

We often deal with various different ‘primitives’ that we describe our geometry with. Objects are constructed from these primitives

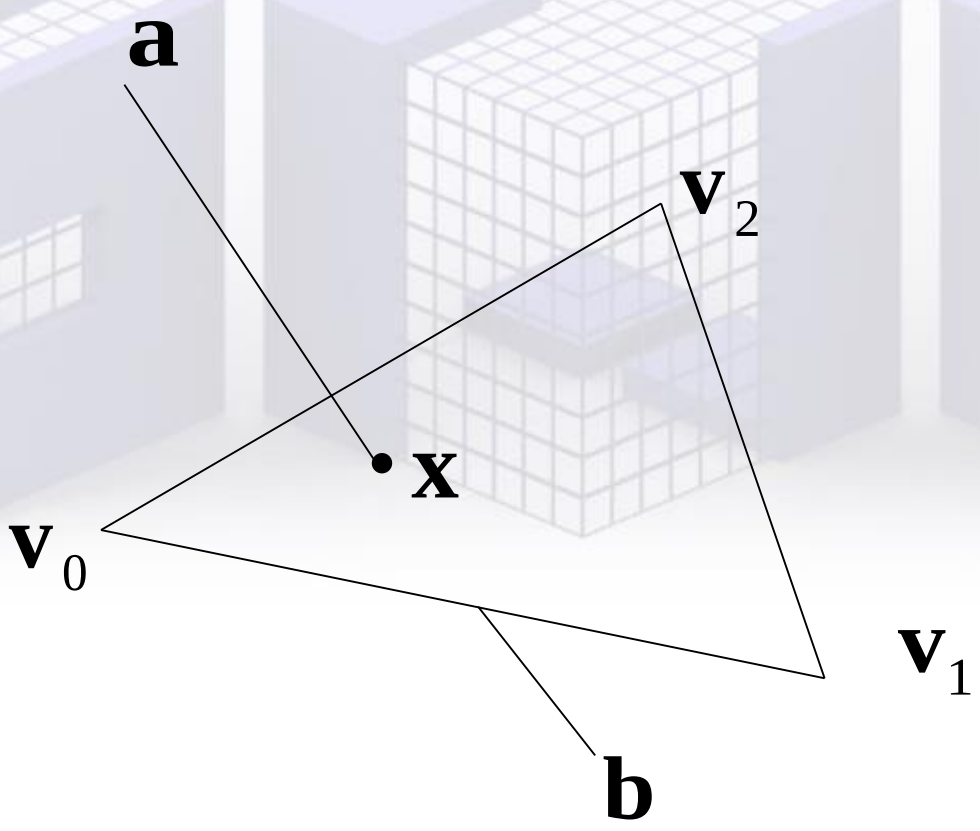
Examples

- Triangles
- Spheres
- Cylinders
- AABB = axis aligned bounding box
- OBB = oriented bounding box

At the heart of the intersection testing are various primitive-primitive tests

SEGMENT VS. TRIANGLE

Does segment **ab** intersect triangle $\mathbf{v}_0\mathbf{v}_1\mathbf{v}_2$?

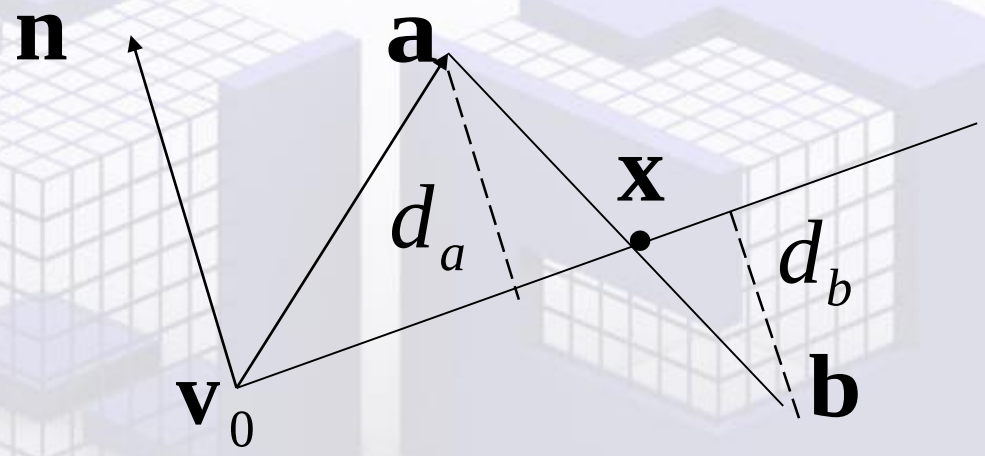


SEGMENT VS. TRIANGLE

First, compute signed distances of a and b to plane

$$d_a = (\mathbf{a} - \mathbf{v}_0) \cdot \mathbf{n}$$

$$d_b = (\mathbf{b} - \mathbf{v}_0) \cdot \mathbf{n}$$



Reject if both are above or both are below triangle

Otherwise, find intersection point **x**

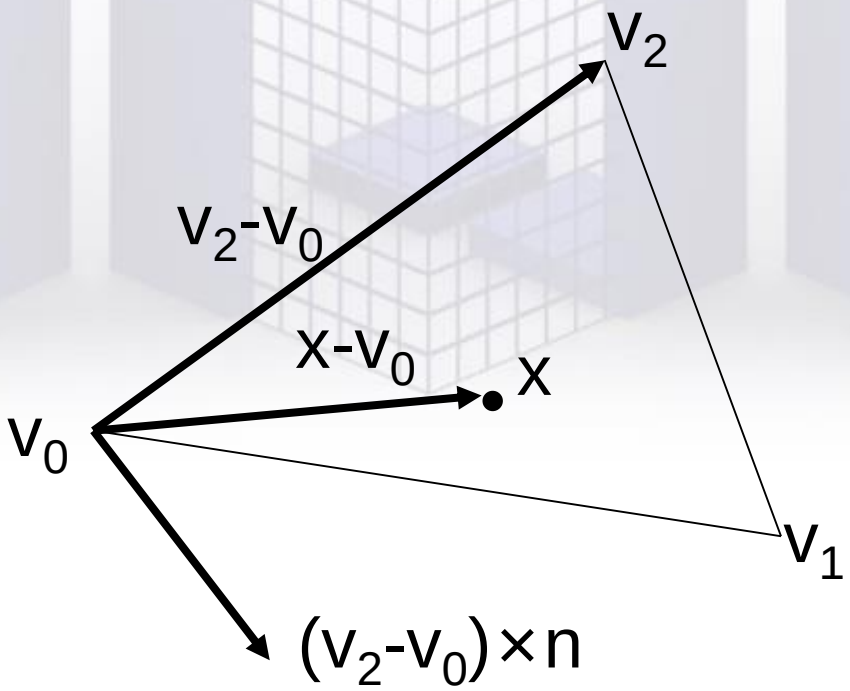
$$\mathbf{x} = \frac{d_a \mathbf{b} - d_b \mathbf{a}}{d_a - d_b}$$

SEGMENT VS. TRIANGLE

Is point x inside the triangle?

$$(x-v_0)\cdot((v_2-v_0)\times n) > 0$$

Test all 3 edges

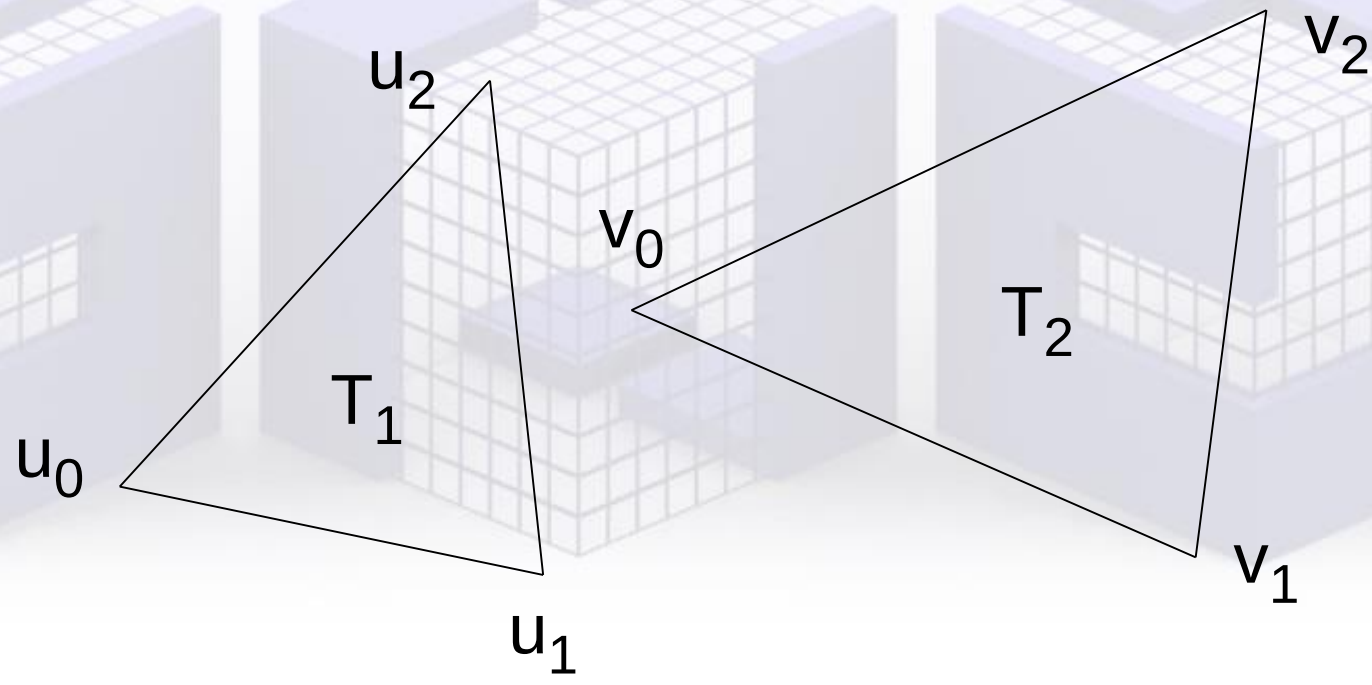


SEGMENT VS. MESH

- To test a line segment against a mesh of triangles, simply test the segment against each triangle
- Sometimes, we are interested in only the ‘first’ hit along the segment from **a** to **b**. Other times, we want all intersections. Still other times, we just need any intersection.
- Testing against lots of triangles in a large mesh can be time consuming. We will look at ways to optimize this later

TRIANGLE VS. TRIANGLE

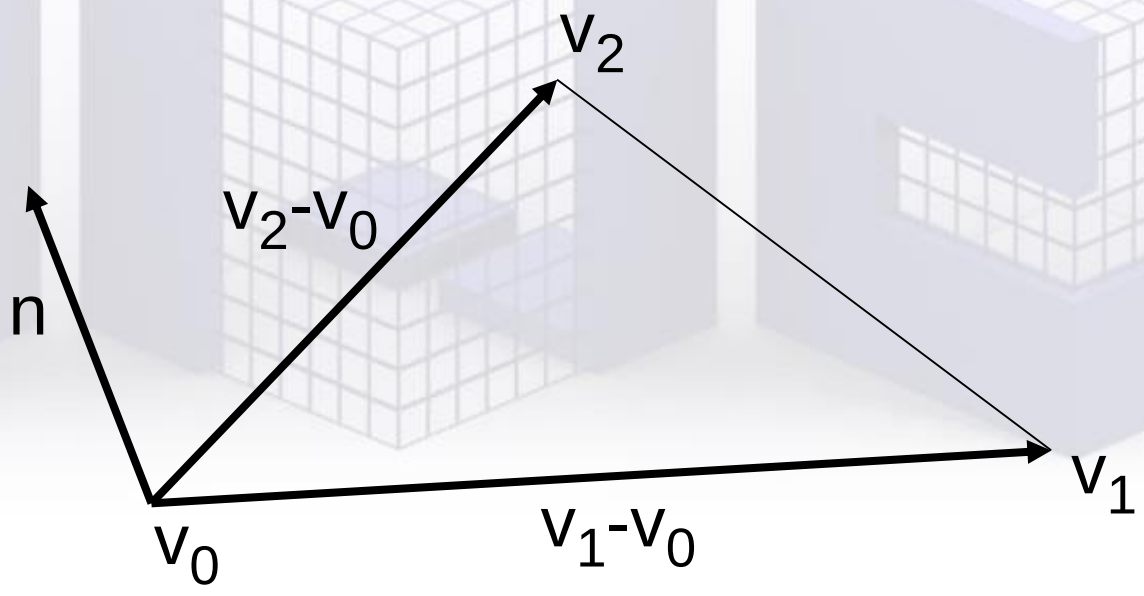
Given two triangles: T_1 ($u_0u_1u_2$) and T_2 ($v_0v_1v_2$)



TRIANGLE VS. TRIANGLE

Step 1: Compute plane equations

$n=(v_1-v_0)\times(v_2-v_0)$



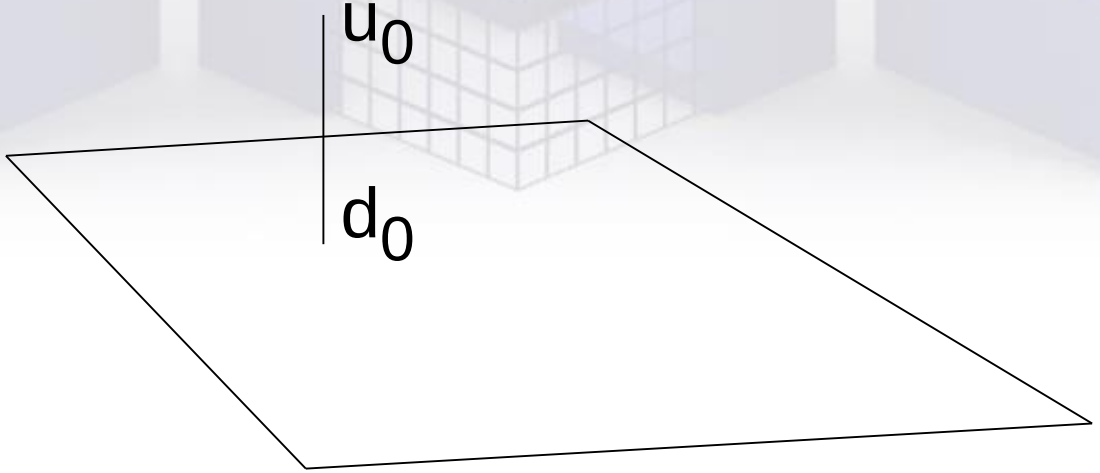
TRIANGLE VS. TRIANGLE

Step 2: Compute signed distances of T_1 vertices to plane of T_2:

$d_i = n_2 \cdot u_i + d_2 \qquad (i=0,1,2)$

Reject if all $d_i<0$ or all $d_i>0$

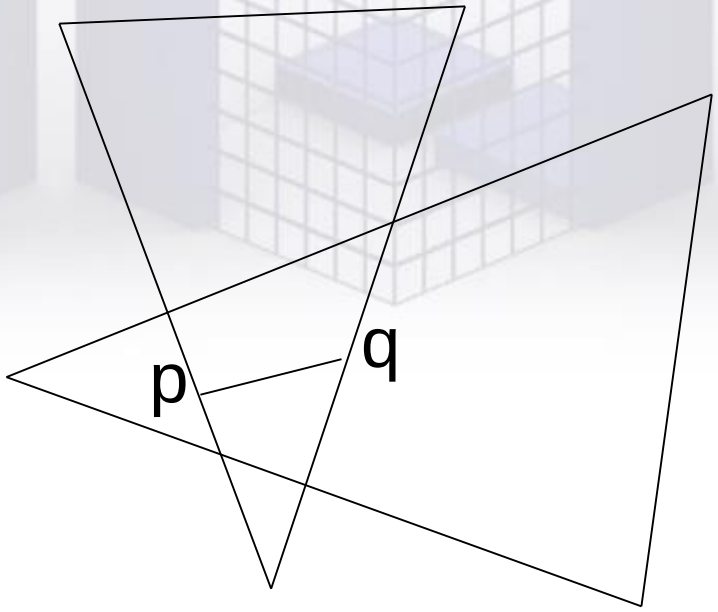
Repeat for vertices of T_2 against plane of T_1



TRIANGLE VS. TRIANGLE

Step 3: Find intersection points

Step 4: Determine if segment pq is inside triangle or intersects triangle edge



MESH VS. MESH

Geometry: points, edges, faces

Collisions: p2p, p2e, p2f, e2e, e2f, f2f

Relevant ones: p2f, e2e (point to face & edge to edge)

Multiple simultaneous collisions

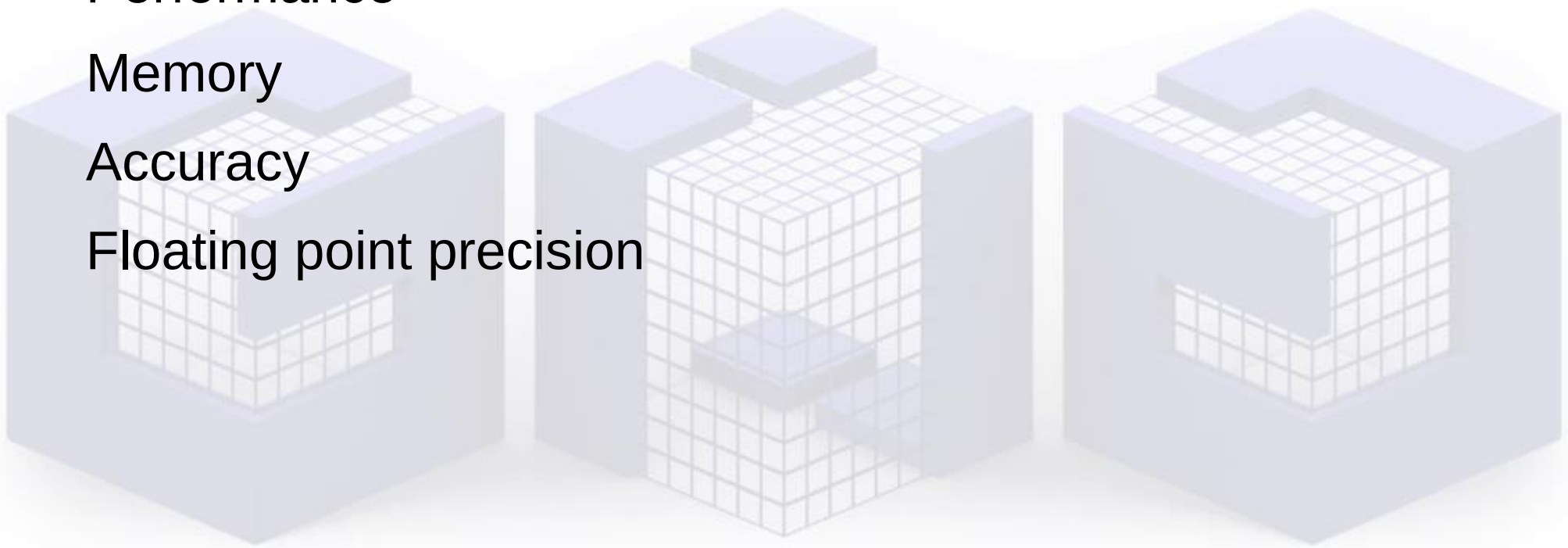
INTERSECTION ISSUES

Performance

Memory

Accuracy

Floating point precision



COLLISION RESPONSE

IMPACT VS. CONTACT

- In physics simulation, there is usually a distinction between impacts and contacts
- Impacts are instantaneous collisions between objects where an impulse must be generated to prevent the velocities at the impact location from allowing the objects to interpenetrate
- Contacts are persistent and exist over some range of time. In a contact situation, the closing velocities at the contact location should already be 0, so forces are needed to keep the objects from accelerating into each other. With rigid bodies, contacts can include fairly complex situations like stacking, rolling, and sliding

IMPACT VS. CONTACT

- Neither impact nor contact is particularly easy to handle correctly
- In the case of particles, it's not so bad, but with rigid bodies, it can be tough
- As we are mainly just concerned with the physics of particles, we will not worry about the more complex issues for now
- Also, we will just focus on handling impacts, as they are generally needed first. Continuous contact will just be handled by allowing particles to impact frame after frame

IMPACTS

- When two solid objects collide (such as a particle hitting a solid surface), forces are generated at the impact location that prevent the objects from interpenetrating
- These forces act over a very small time and as far as the simulation is concerned, it's easiest to treat it as an instantaneous event
- Therefore, instead of the impact applying a force, we must use an *impulse*

IMPULSE

- An impulse can be thought of as the integral of a force over some time range, which results in a finite change in momentum:

$$\mathbf{j} = \int \mathbf{f} dt = \Delta \mathbf{p}$$

- An impulse behaves a lot like a force, except instead of affecting an object's acceleration, it directly affects the velocity
- Impulses also obey Newton's Third Law, and so objects can exchange equal and opposite impulses
- Also, like forces, we can compute a total impulse as the sum of several individual impulses

COMPRESSION & RESTITUTION

- The collision can be thought of as having two phases: compression & restitution
- In the compression phase, the energy of the two objects is changed from kinetic energy of motion into deformation energy in the solids
- If the collision is perfectly *inelastic* (e=0), then all of the energy is lost and there will be no relative motion along the collision normal after the collision
- If the collision is perfectly *elastic* (e=1), then all of the deformation energy will be turned back into kinetic energy in the restitution phase and the velocity along the normal will be the opposite of what it was before the collision

COMPRESSION & RESTITUTION