Animation II

COM3404

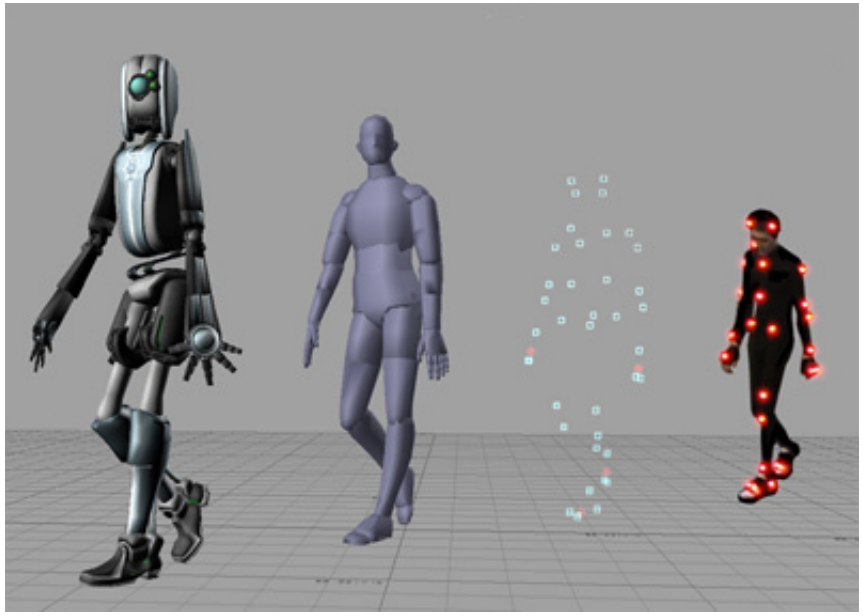
Richard Everson

School of Engineering, Computer Science and Mathematics
University of Exeter

R.M.Everson@exeter.ac.uk
http://www.secamlocal.ex.ac.uk/studyres/COM304

Outline

1. Kinematics
   - Rigging
   - Forward kinematics
   - Inverse kinematics
2. Motion dynamics
   - Collision detection
3. Particle systems
   - Fabrics



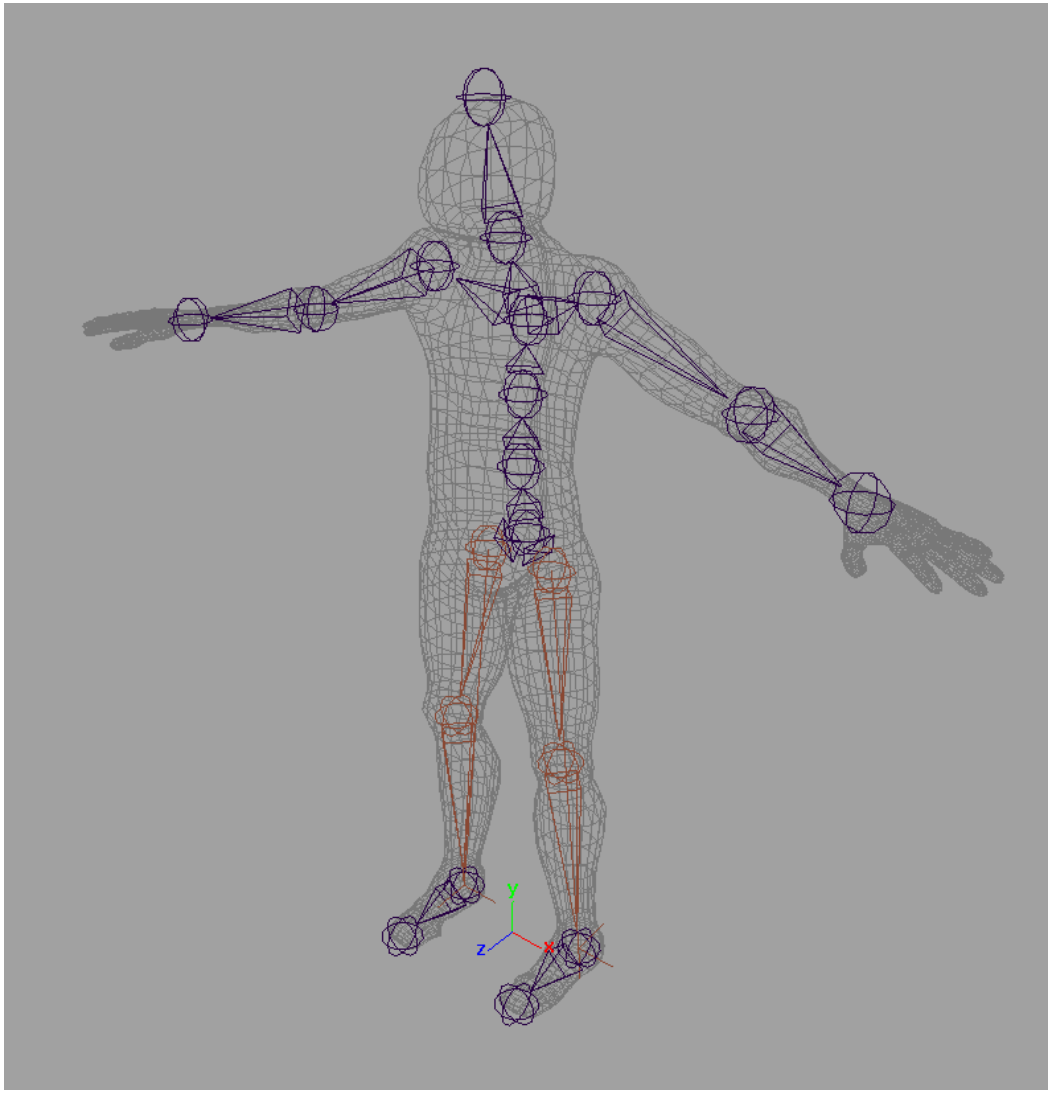
References

- Fundamentals of 3D Computer Graphics. Watt.
- Computer Graphics: Principles and Practice. Foley et al.
- Principles of Three-Dimensional Computer Animation. M. O'Rourke.
- Introducing Character Animation with Blender. T. Mullen.

Skeletal animation: rigging

Animating complex polygonal meshes

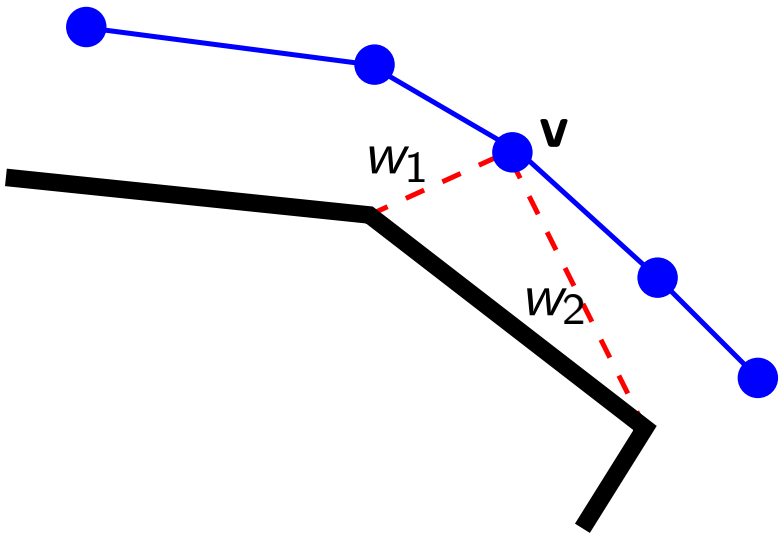
- Construct a *skeleton* of *bones* capturing the components and joints of the body
- Rig the skeleton by attaching vertices to skeleton
- Animate skeleton
- Each vertex location is a weighted combination of bone locations:
  eg: 1 part shin and 2 parts thigh for a vertex in the knee



Matrix palette skinning

Bones of skeleton each has a transformation matrix M_i describing its animation from the untransformed location in world coordinates

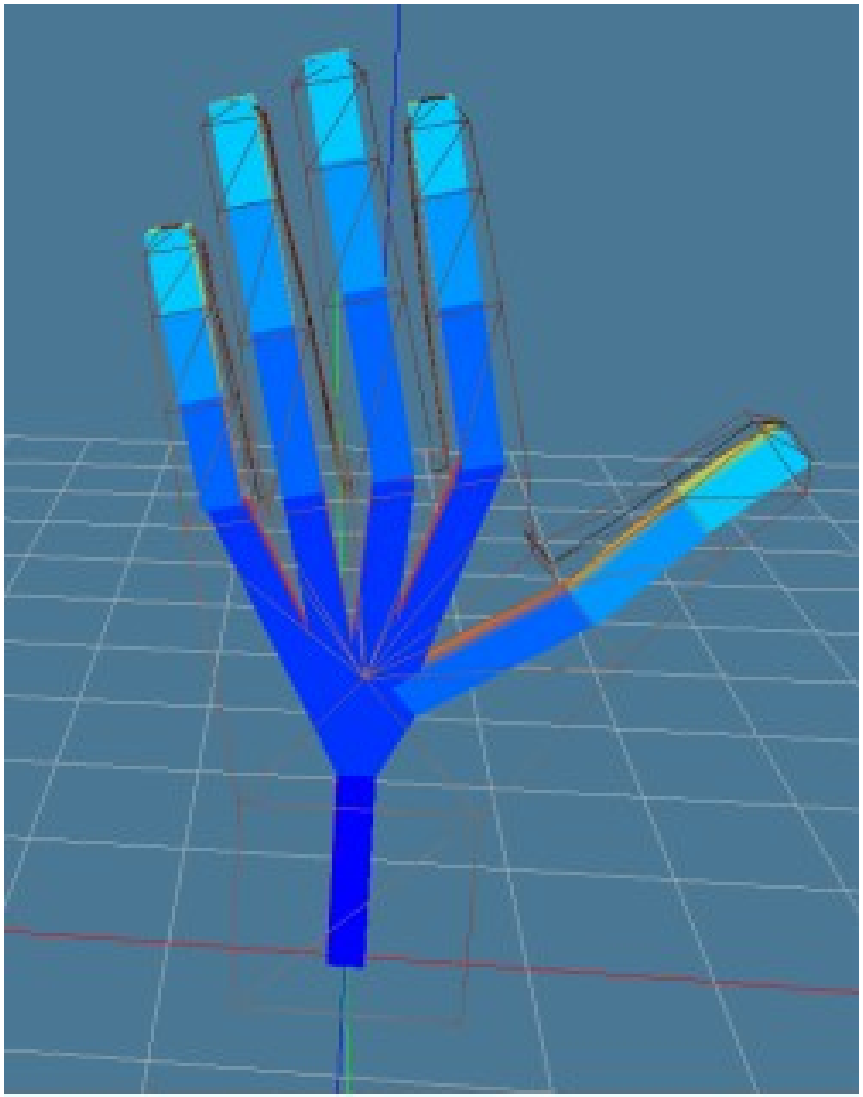
Vertices are a linear combination (blend) of a palette of neighbouring bone locations:



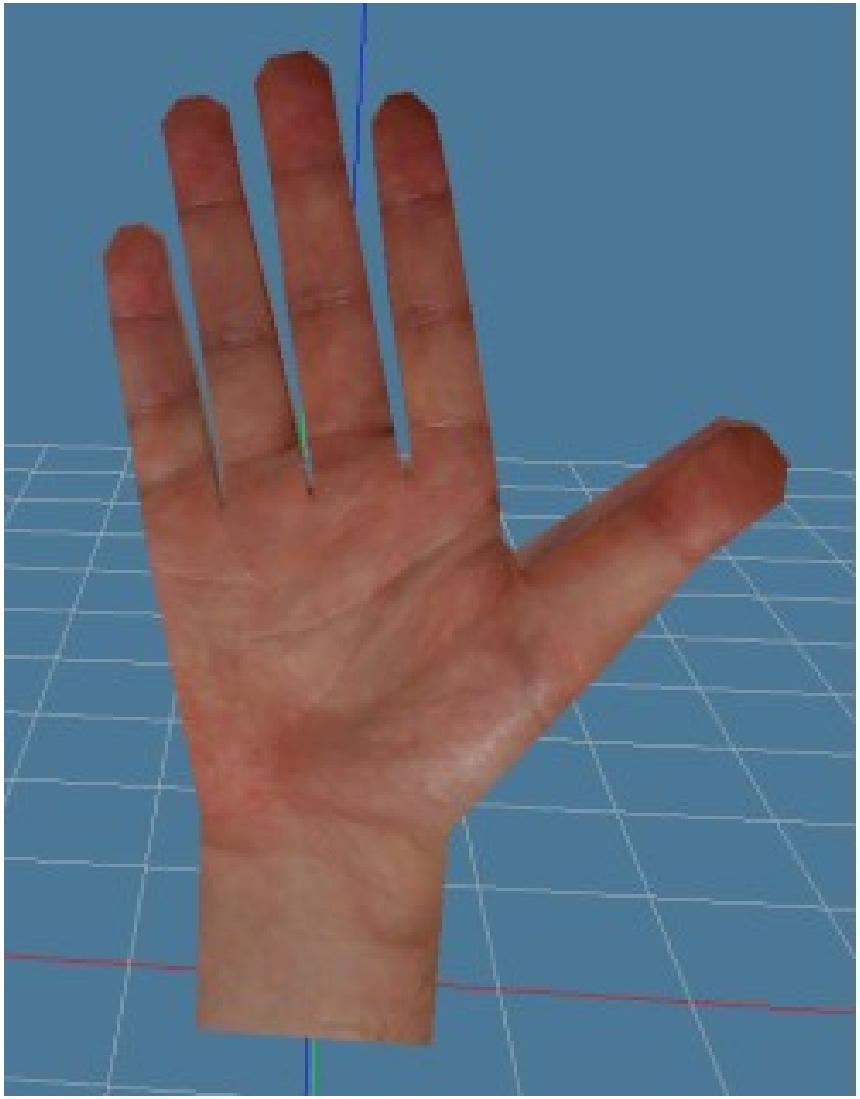
$$\mathbf{v}' = \sum_i w_i M_i \mathbf{v}$$

- $\mathbf{v}$ untransformed vertex location
- $\mathbf{v}'$ transformed vertex location
- M_i transformation matrix describing motion of bone i
- w_i weighting for ith bone. $\sum_i w_i = 1$

Matrix palette skinning



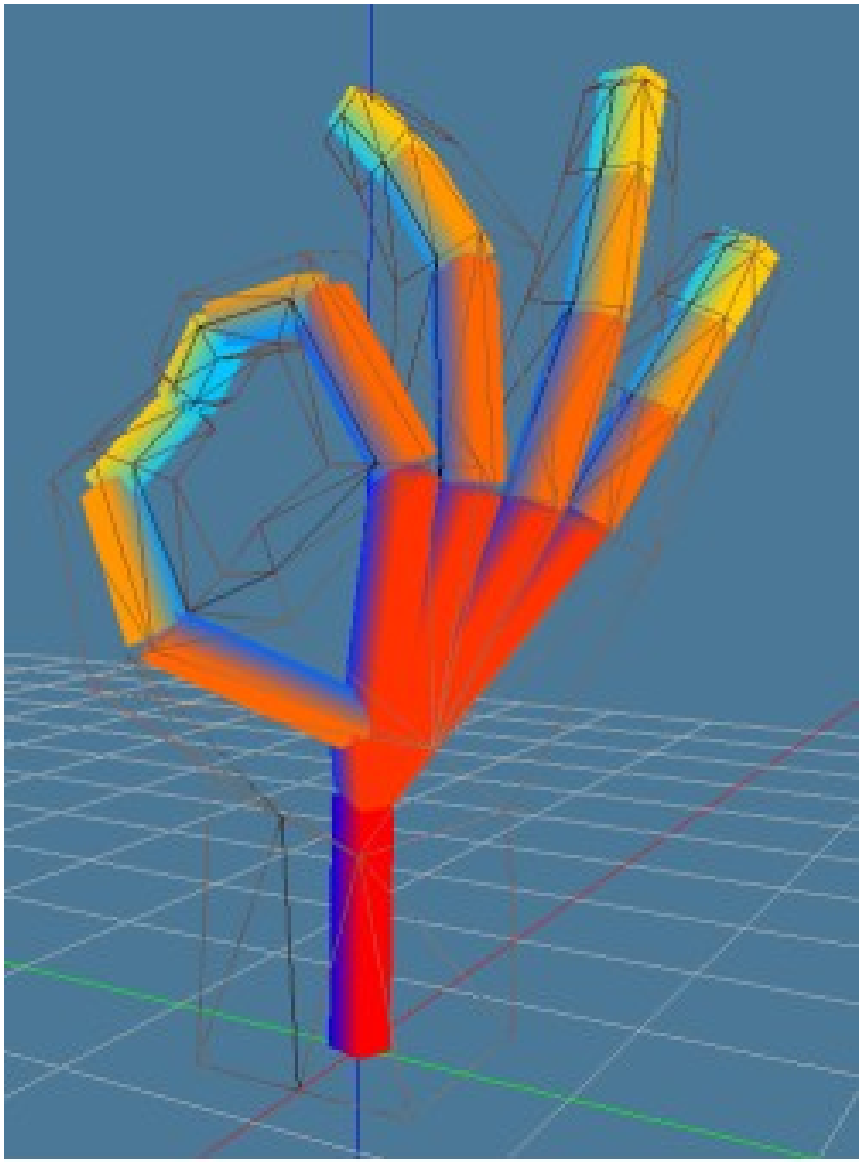
Untransformed bones and mesh



Texture mapped mesh

http://www.gup.uni-linz.ac.at/~gk/Praktika/meshskin_webdata/

Matrix palette skinning



Transformed bones and mesh

Texture mapped mesh

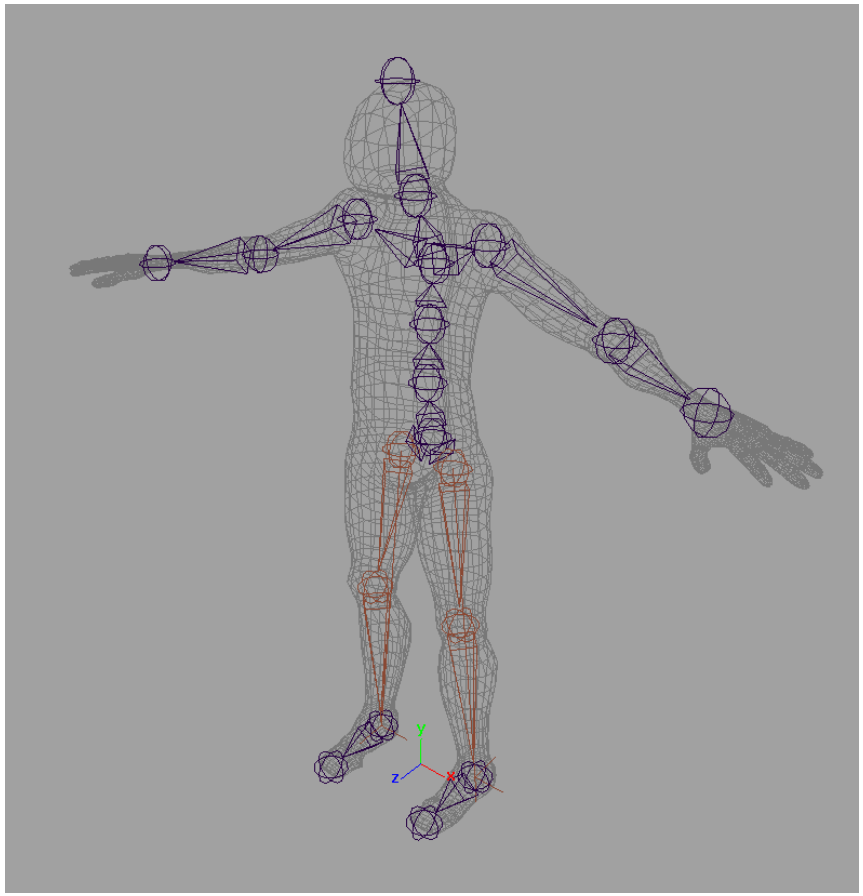
http://www.gup.uni-linz.ac.at/~gk/Praktika/meshskin_webdata/

Forward kinematics

How to determine the positions of a skeleton?

Forward kinematics

- Animator specifies rotations at each joint
- Calculate the location of object from hierarchical object structure

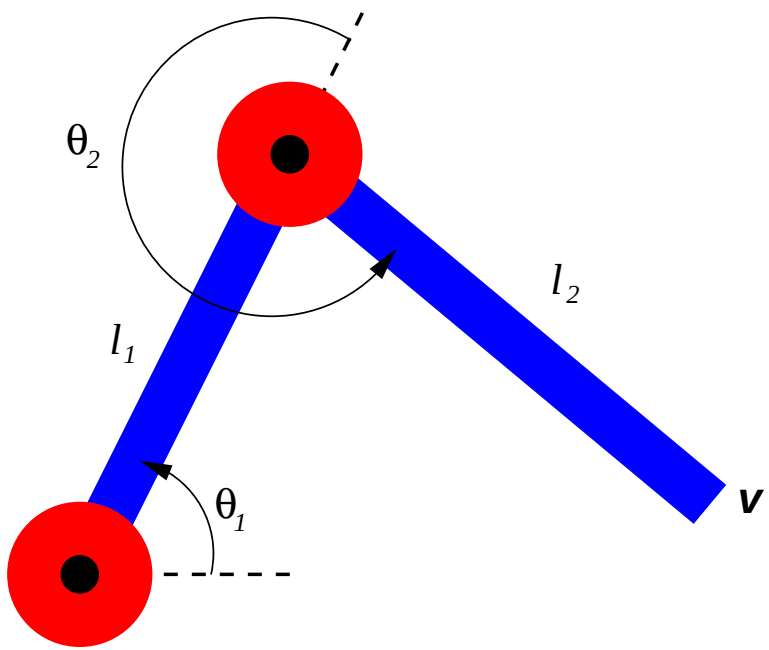


Example: Two links connected by rotational joints

Location of end

$$\mathbf{v}' = \begin{bmatrix} l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) \\ l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) \end{bmatrix}$$
$$= T_1 R_1 T_2 R_2 \mathbf{v}$$

if initial location is $\mathbf{v}$



Dynamics

Joint motions specified by:

- spline curves for $\theta_1(t)$ and $\theta_2(t)$
- initial locations and velocities and differential equations

Animation

- Compute vertex locations from $\mathbf{v}' = T_1 R_1(t) T_2 R_2(t) \mathbf{v}$
- Use matrix palette skinning to compute polygon vertex locations

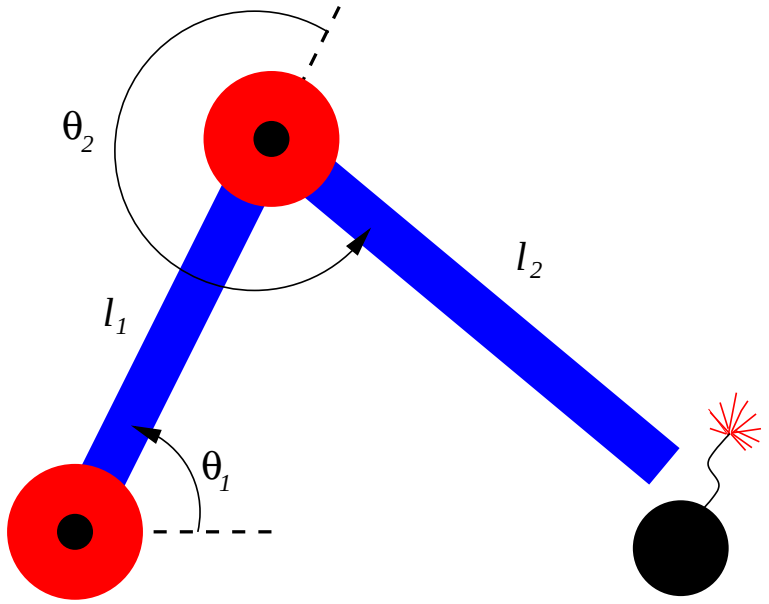
Inverse kinematics

- Animator specifies location of ends
- Calculate the appropriate joint angles

If $\mathbf{v}' = (x, y)$ then

$$\theta_2 = \cos^{-1} \frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}$$

$$\theta_1 = \frac{-(l_2 \sin\theta_2)x + (l_1 + l_2 \cos\theta_2)y}{(l_2 \sin\theta_2)y + (l_1 + l_2 \cos\theta_2)x}$$



Dynamics

- Location of ends $\mathbf{v}(t)$ specified by the animator
- Intermediate joint angles calculated: $\theta_1(t), \theta_2(t)$
- Compute vertex locations from $\mathbf{v}' = T_1 R_1(t) T_2 R_2(t) \mathbf{v}$
- Matrix palette skinning to compute polygon vertices

Inverse kinematics

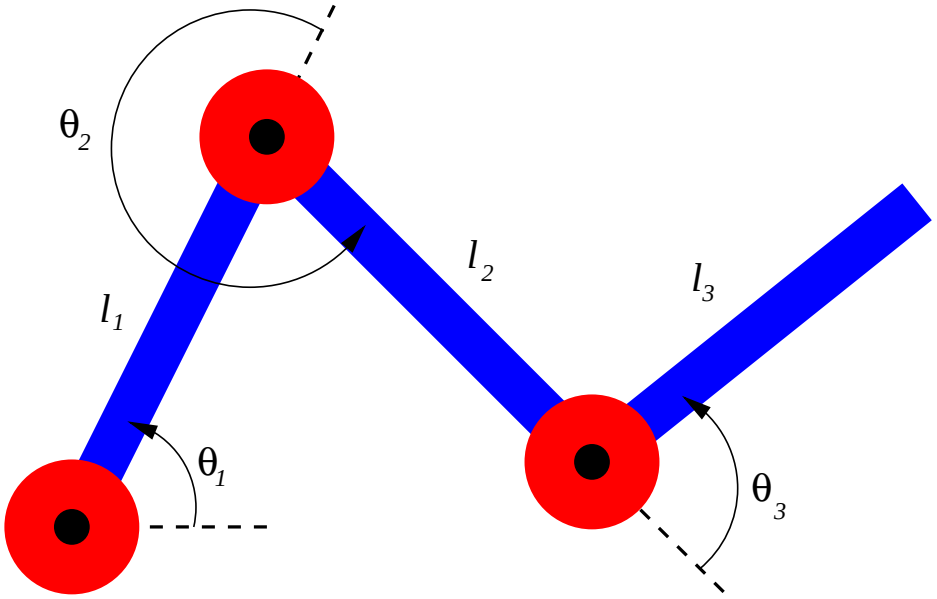
Problems with complex structures

- Systems of equations under-determined
- Multiple solutions exist

Example With three links:

Two equations: x and y

Three unknowns: $\theta_1, \theta_2, \theta_3$



Remedy

- Find best solution by, e.g., minimising energy in motion
- Apply as many constraints as possible. For example:
  - Possible angles for a shoulder joint
  - Physics of the motion
- Requires costly non-linear optimisation

Forward/inverse kinematics

Forward kinematics

- Specify joints
- Compute positions
- Computationally straightforward
- Difficult to precisely position limbs

Inverse kinematics

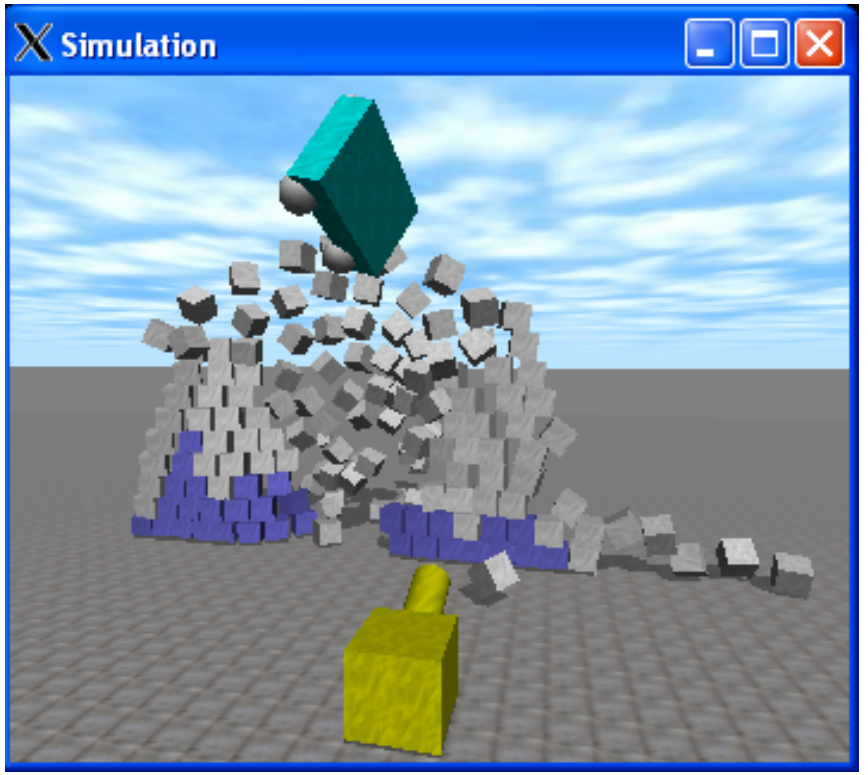
- Goal-directed motion
- Specify goal positions
- Compute joint angles and locations to achieve goals
- Very suitable for animating precise movements; naturalism may be a problem

Inverse kinematics are easily specified for animator, but computationally demanding

Motion dynamics

Why

- Some systems are very simple to model from first principles; e.g. a bouncing ball
- Complicated systems with many, interacting elements can only be realistically modelled using equations of motion; e.g. building blocks falling down-stairs, snooker balls.



Motion dynamics

Physics of rigid bodies obeys Newton's laws of motion. For each body:

$$\mathbf{F} = m\mathbf{a} = m\frac{d^2\mathbf{x}}{dt^2}$$

- $\mathbf{F}$ Forces acting on particle: gravity, friction, other particles
- $\mathbf{a}$ Acceleration
- m Mass

Similar equations describing rotation in terms of moment of inertia.

Integration Equation of motion analytically integrated when no collisions

- General purpose "physics engines"
- Collisions must be detected and dealt with

Collision detection

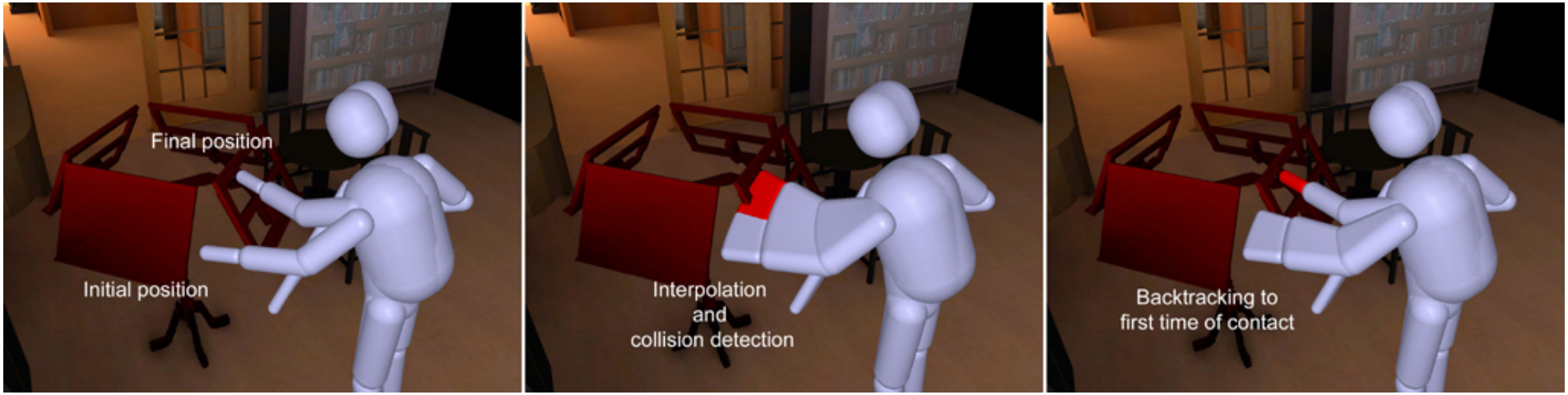
Detect whether two polygonal meshes intersect

Complexity If meshes have N and M facets $O(MN/2)$ checks are required.

Simplifications

- Embed objects into *bounding volumes* and check these first
- Use grids or hierarchical structures to avoid checking triangles far away from each other

Prior and posterior detection. Prior detection is usually difficult, but recomputation after collision required otherwise.



Bounding volumes

Axis aligned bounding box

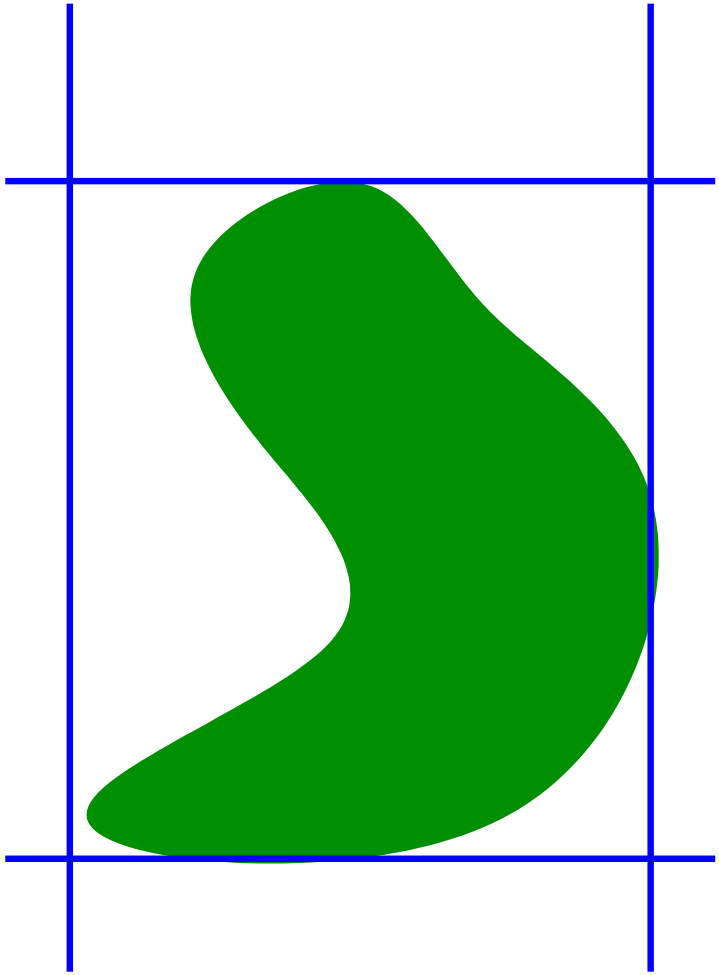
Intersection iff there is interval intersection between all coordinate axes

Object oriented bounding box

More difficult intersection checks to account from different rotations of BB for each object

Spheres

Compare radii and centres



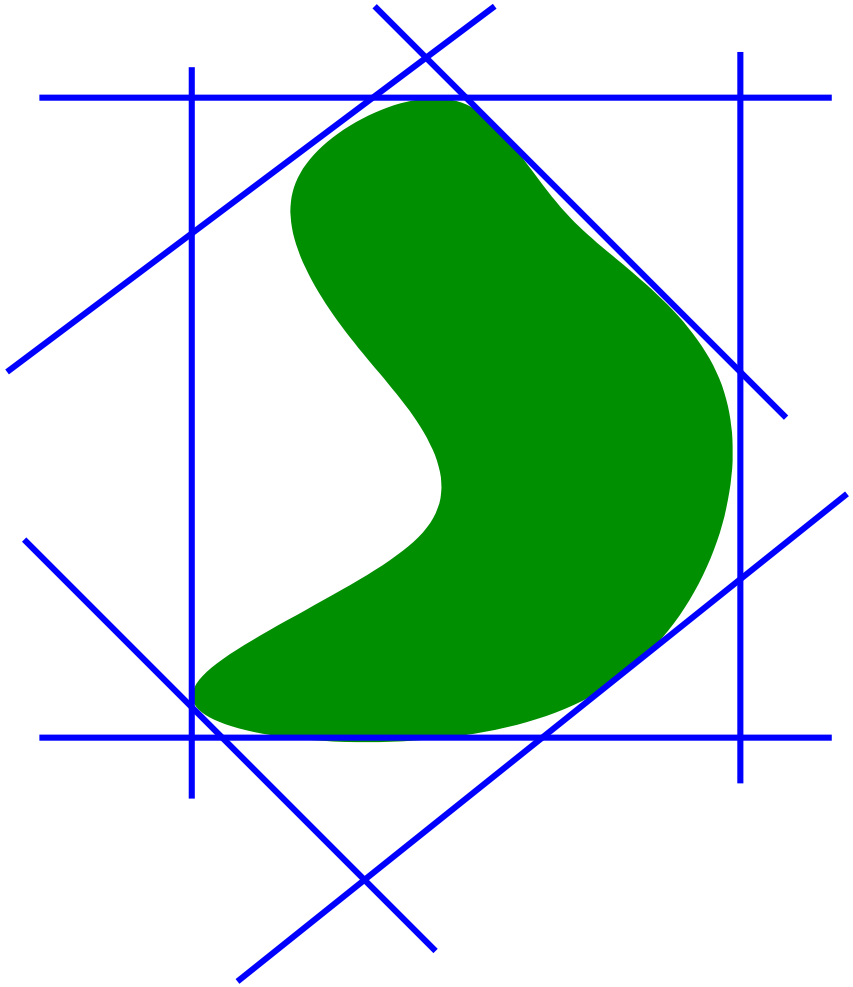
Bounding volumes: k-DOPS

k-discrete oriented polytopes

k-DOP: convex polyhedron defined by k half-spaces enclosing the object

- Rapid check for intersection if all half-spaces intersect.
- Examples:

6-DOP	AABB
8-DOP	Octahedron
14-DOP	6-DOP + 8-DOP
18-DOP	AABB with clipped edges



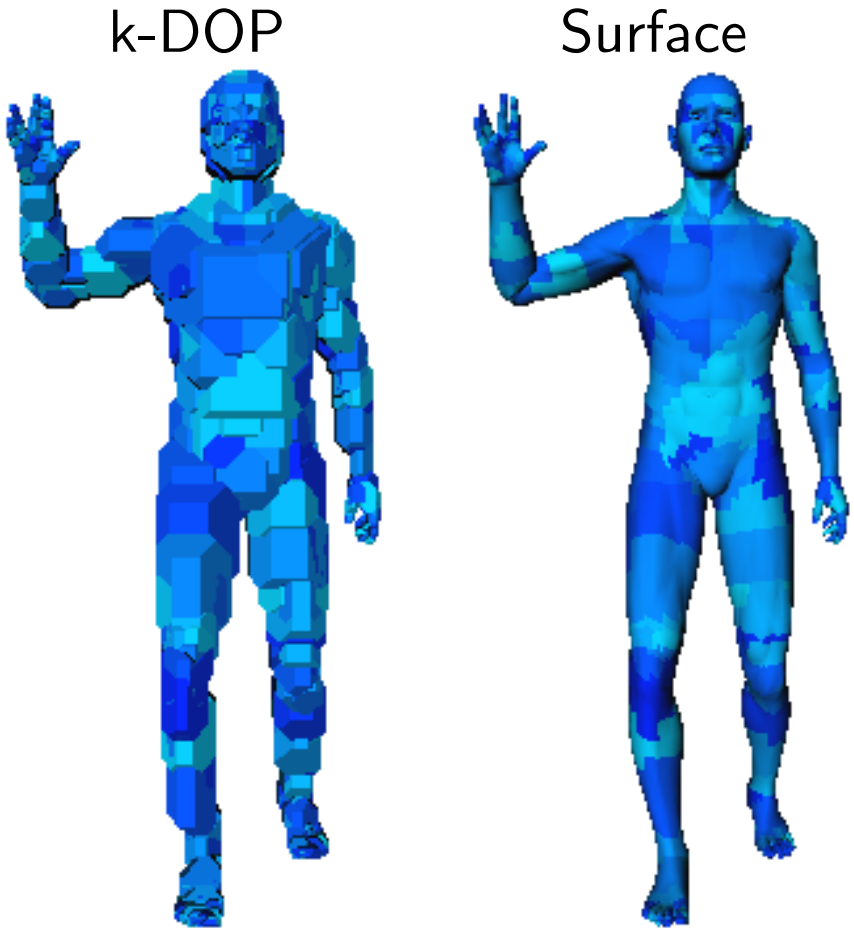
Bounding volume hierarchies

Grid subdivision Divide space into voxel grid linking to the vertices in each grid

Hierarchical subdivision Octrees organise AABBs

k-DOP object hierarchy Divide each k-DOP into two by splitting along an axis side and recomputing subdivision.

- Good for representing deformable surfaces with constant topological structure.



k-DOP splitting criteria

Choose the axis according to:

Min Sum Choose the axis minimises the sum of the volumes of the two resulting children.
Slow.

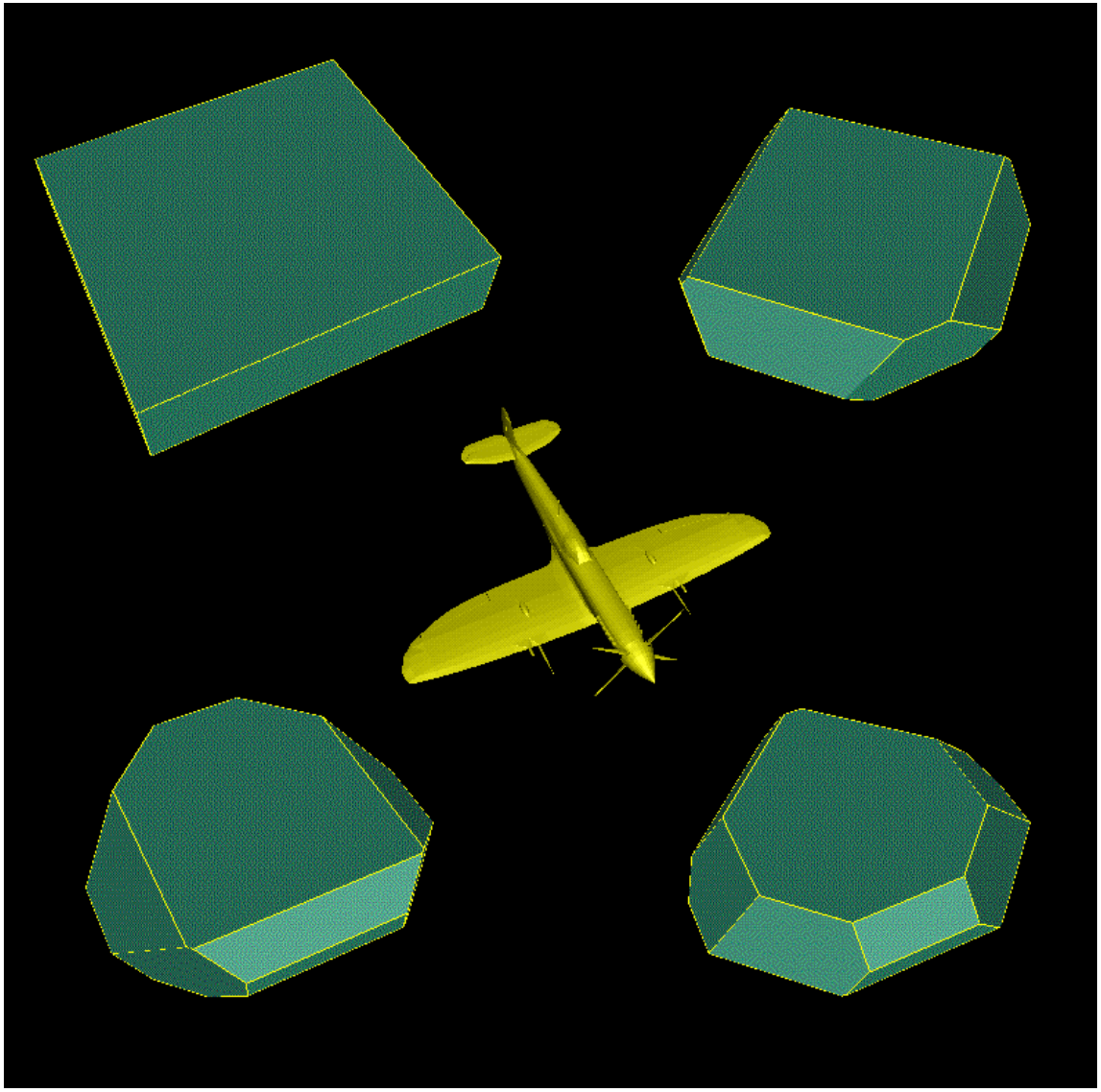
Min Max Choose the axis that minimises the larger of the volumes of the two resulting children. Slow.

Splatter Project the centroids of the facets onto each axis and calculate the variance of the resulting distributions. Choose the axis with the largest spread (variance).
Time proportional to number of primitives in the current k-dop.

Longest Side Choose the longest axis of the current k-dop.
Constant time.

J.T. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, K. Zikan (1997): "Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs" IEEE Transactions on Visualization and Computer Graphics, March 1998, Volume 4, Number 1.

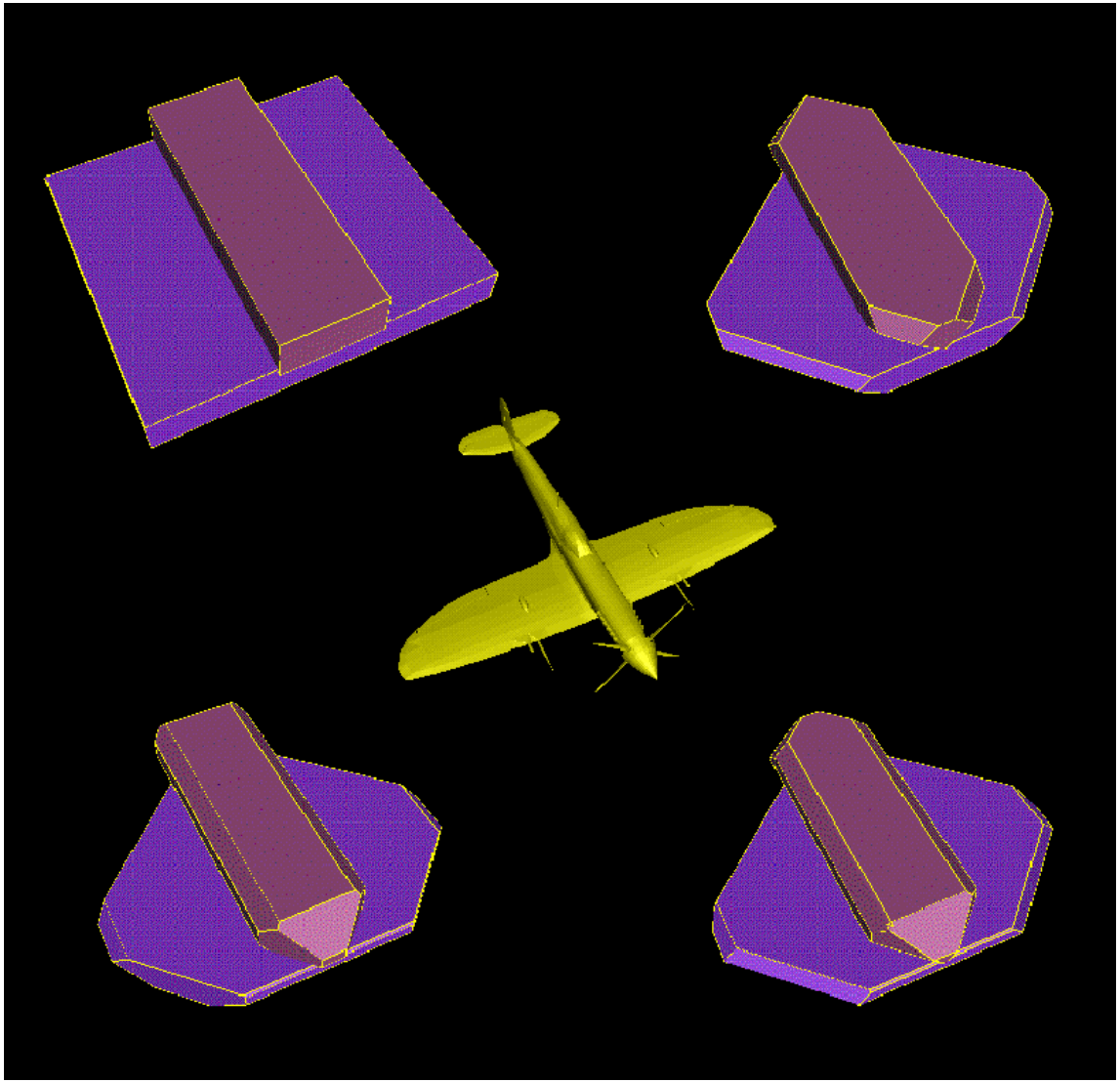
k-DOP hierarchies



Level 0 bounding volume trees for 6-DOPS, 14-DOPS, 18-DOPs and 26-DOPS

`www.ams.sunysb.edu/~jklosow/quickcd/QCD_kdops.html`

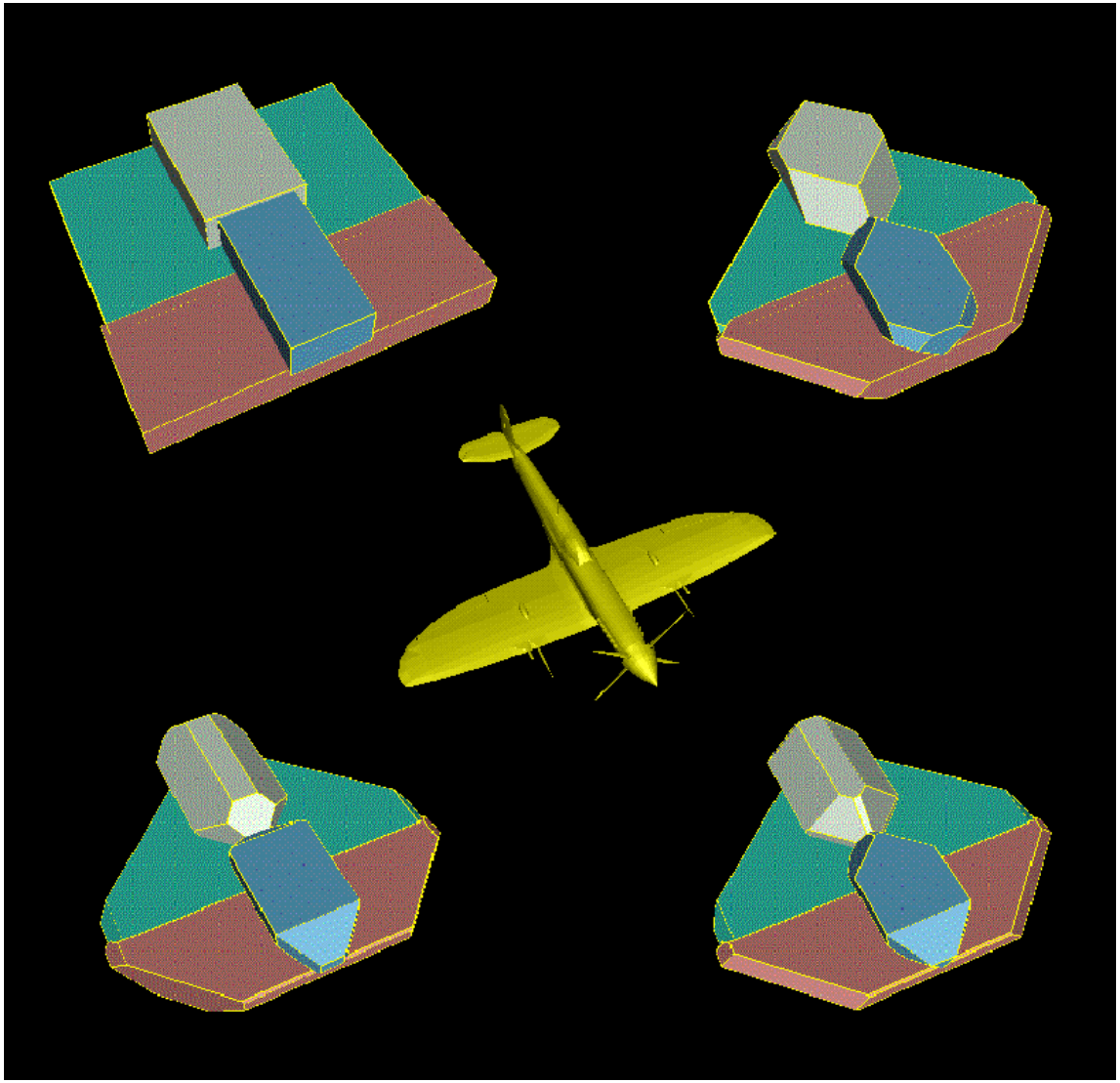
k-DOP hierarchies



Level 1 bounding volume trees for 6-DOPS, 14-DOPS, 18-DOPs and 26-DOPS

`www.ams.sunysb.edu/~jklosow/quickcd/QCD_kdops.html`

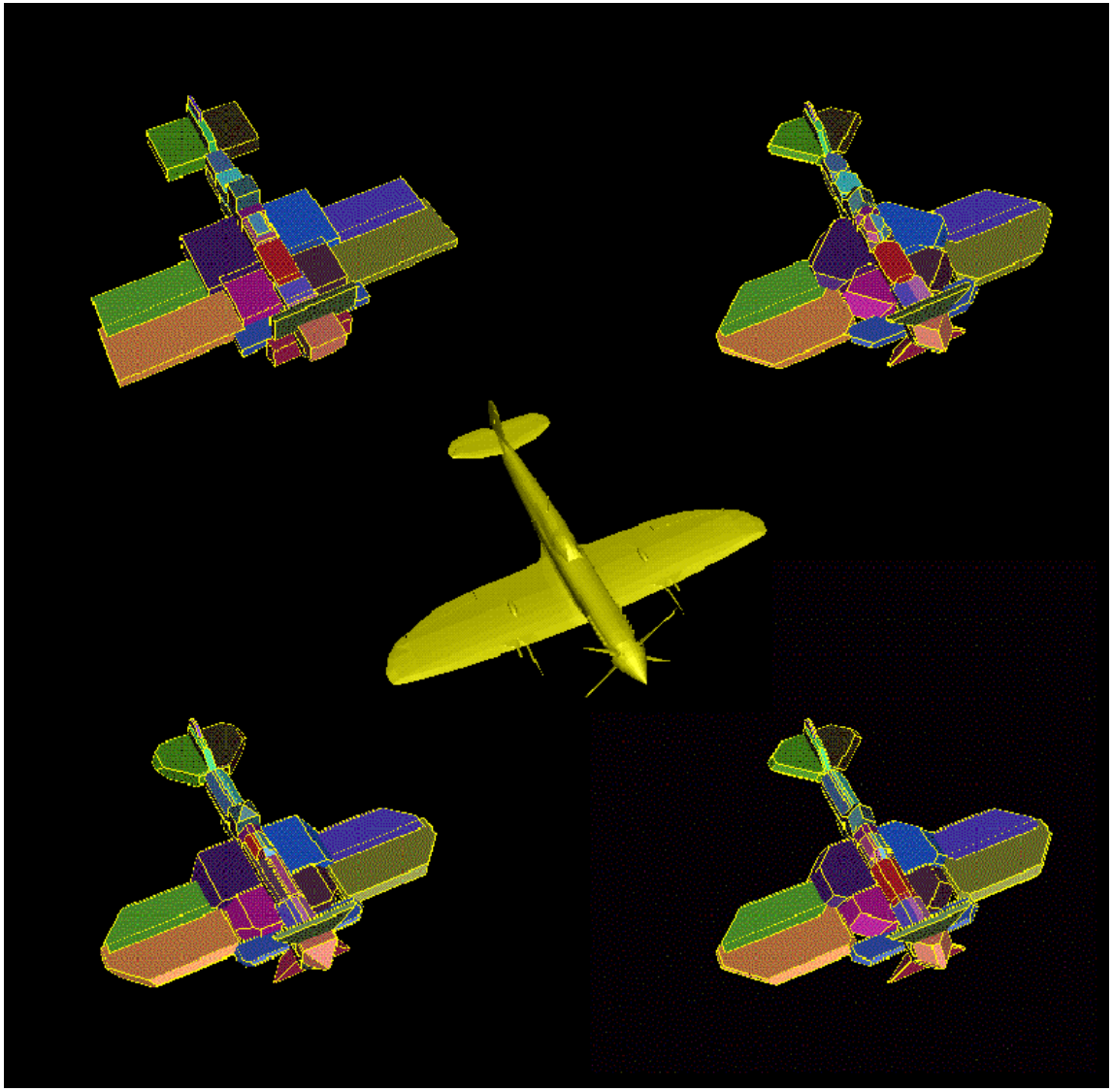
k-DOP hierarchies



Level 2 bounding volume trees for 6-DOPS, 14-DOPS, 18-DOPs and 26-DOPS

`www.ams.sunysb.edu/~jklosow/quickcd/QCD_kdops.html`

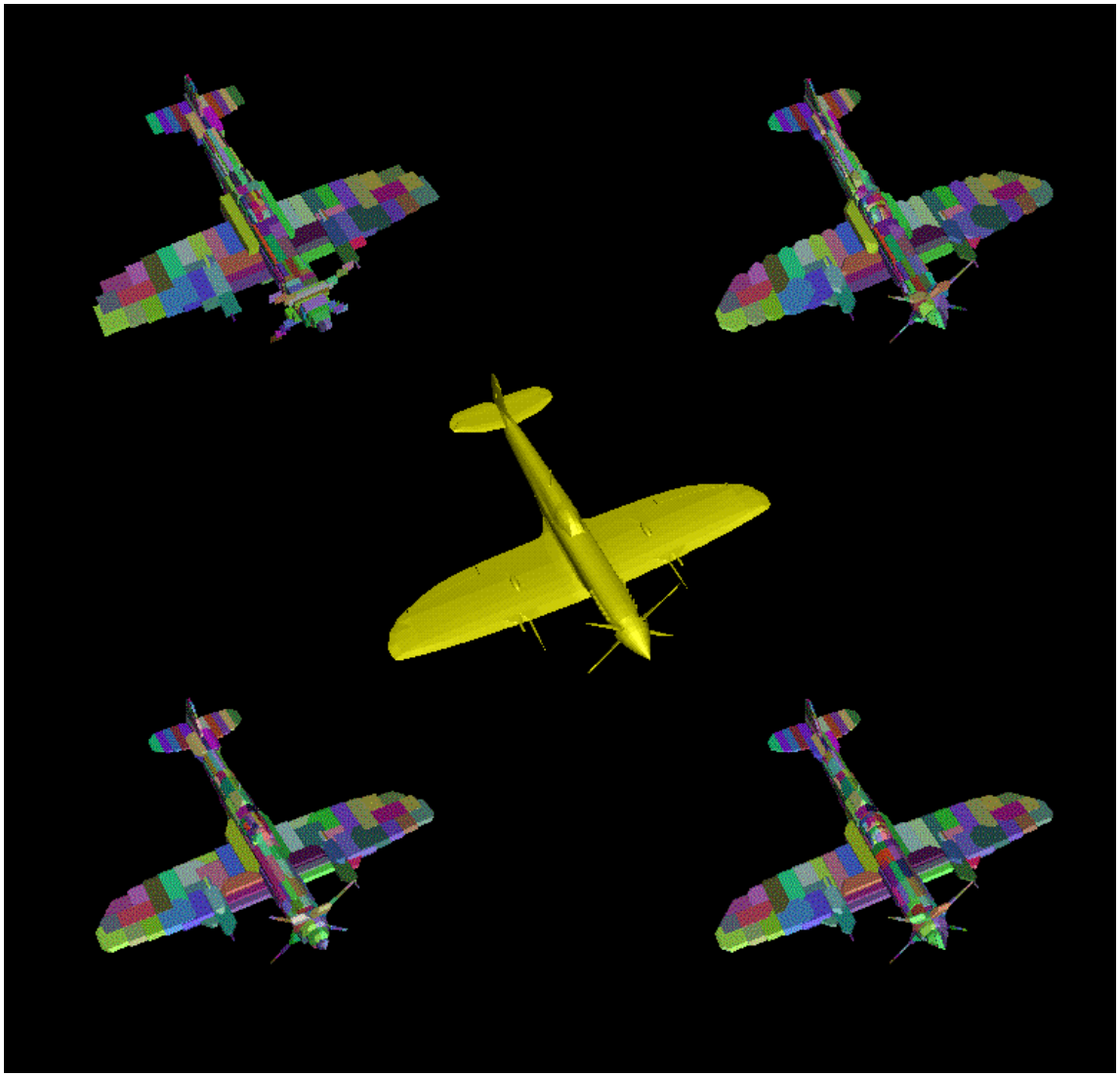
k-DOP hierarchies



Level 5 bounding volume trees for 6-DOPS, 14-DOPS, 18-DOPs and 26-DOPS

`www.ams.sunysb.edu/~jklosow/quickcd/QCD_kdops.html`

k-DOP hierarchies



Level 8 bounding volume trees for 6-DOPS, 14-DOPS, 18-DOPs and 26-DOPS

`www.ams.sunysb.edu/~jklosow/quickcd/QCD_kdops.html`

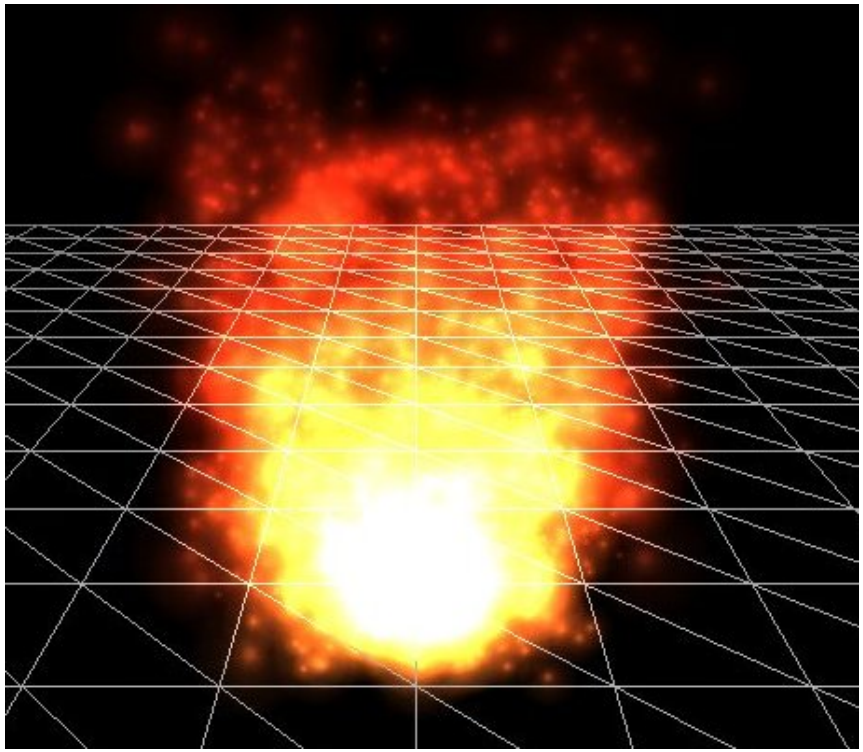
Particle systems

Objects diffuse and deformable objects: fire, smoke, clouds, water, etc.

Representation Object represented by a collection of particles

Dynamic Particles are not static. Particles change form and move. Particles may be destroyed and created.

Stochastic Usually evolution of particles incorporates a stochastic element.



Particle systems

Emitter Generates particles in with

- Location
- Velocity
- Lifetime
- Appearance

Evolution

1. New particles created
2. Aged particles destroyed
3. Particles evolve with rules for updating their positions: eg

$$\mathbf{x}_j(t+1) = \mathbf{x}_j(t) + \mathbf{v}_j(t) + \beta\bar{\mathbf{x}} + \eta$$

moves particle in direction determined by its velocity plus a component towards centre of particle cloud ($\bar{\mathbf{x}}$) plus random component

- Sophisticated interaction with other particles to simulate fluids etc.
- Collision detection for interaction with solids
- Constraints to control particle swarm; e.g. fixing ends of a chain

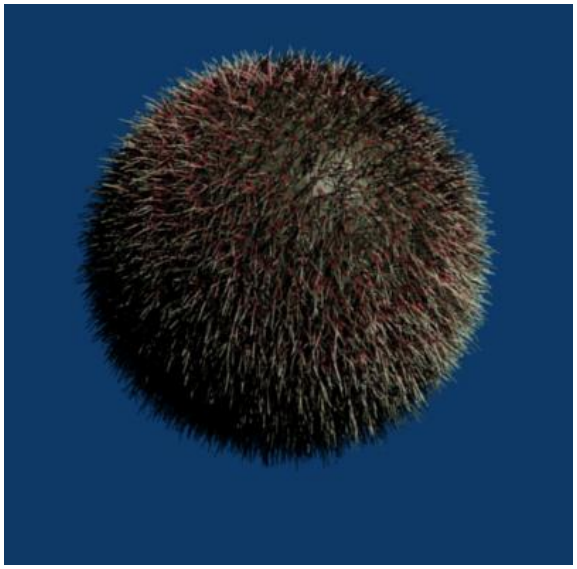
Particle systems

Rendering

- Particles usually rendered as *quad billboard textures*
  Single quadrilateral facets, always facing the viewer
- Single pixels
- Meta-balls: iso-surfaces from many particles for simulating free liquid surfaces
- Particles may also be rendered as complete objects with transparency, shape, colour, emission properties, etc.

Static particles

- Used for objects with stochastic elements
- Static, strand-like particles to simulate hair or fur



Examples

- Genesis effect from Star Trek II: The Wrath of Khan by William Reeves `http://www.siggraph.org/education/materials/HyperGraph/animation/movies/genesisp.mpg`
- Particle dreams by Karl Sims: `http://www.siggraph.org/education/materials/HyperGraph/animation/movies/particle75_1_89.mov`

- William T. Reeves, "Particle Systems - A Technique for Modeling a Class of Fuzzy Objects", Computer Graphics 17:3 pp. 359-376, 1983

Fabrics: free form surfaces

Catenaries Approximate cloth by a collection of cables, each hanging in a catenary (Weil, 1986).

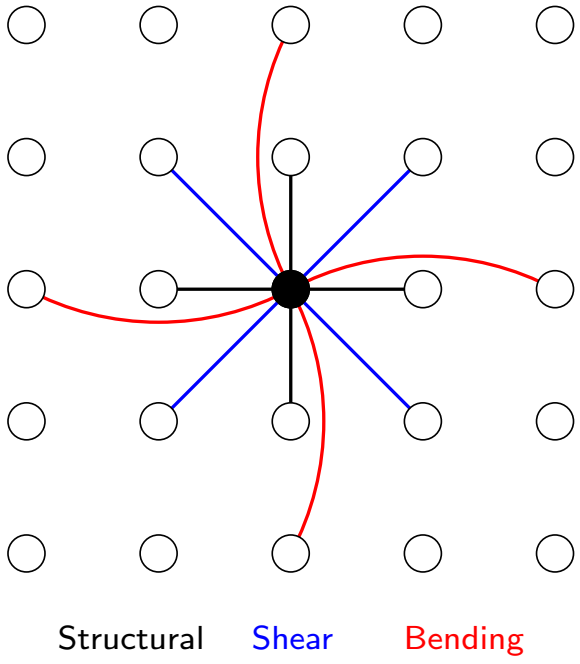
Spring mesh Mesh of particles, with forces derived from

- Elasticity:

$$F_{ij} = k_{ij}[\|\mathbf{x}_i - \mathbf{x}_j\| - l_{ij}]\frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|}$$

k_{ij} elastic modulus
l_{ij} rest length of spring
$\mathbf{x}_i, \mathbf{x}_j$ location of spring ends

- Bending (endows stiffness)
- Weight
- External bodies



Texture from texture maps or specialised bidirectional reflection distribution functions (BRDF).

Fabrics

Advanced methods

- Fabric has thickness.
- Individual yarn loops interconnected by loosely connected springs.
- Modelling of yarn cross-section to describe radiance from a cross section and interaction with other yarn fibres.
- `http://research.microsoft.com/~yqxu/papers/tvcg.pdf`