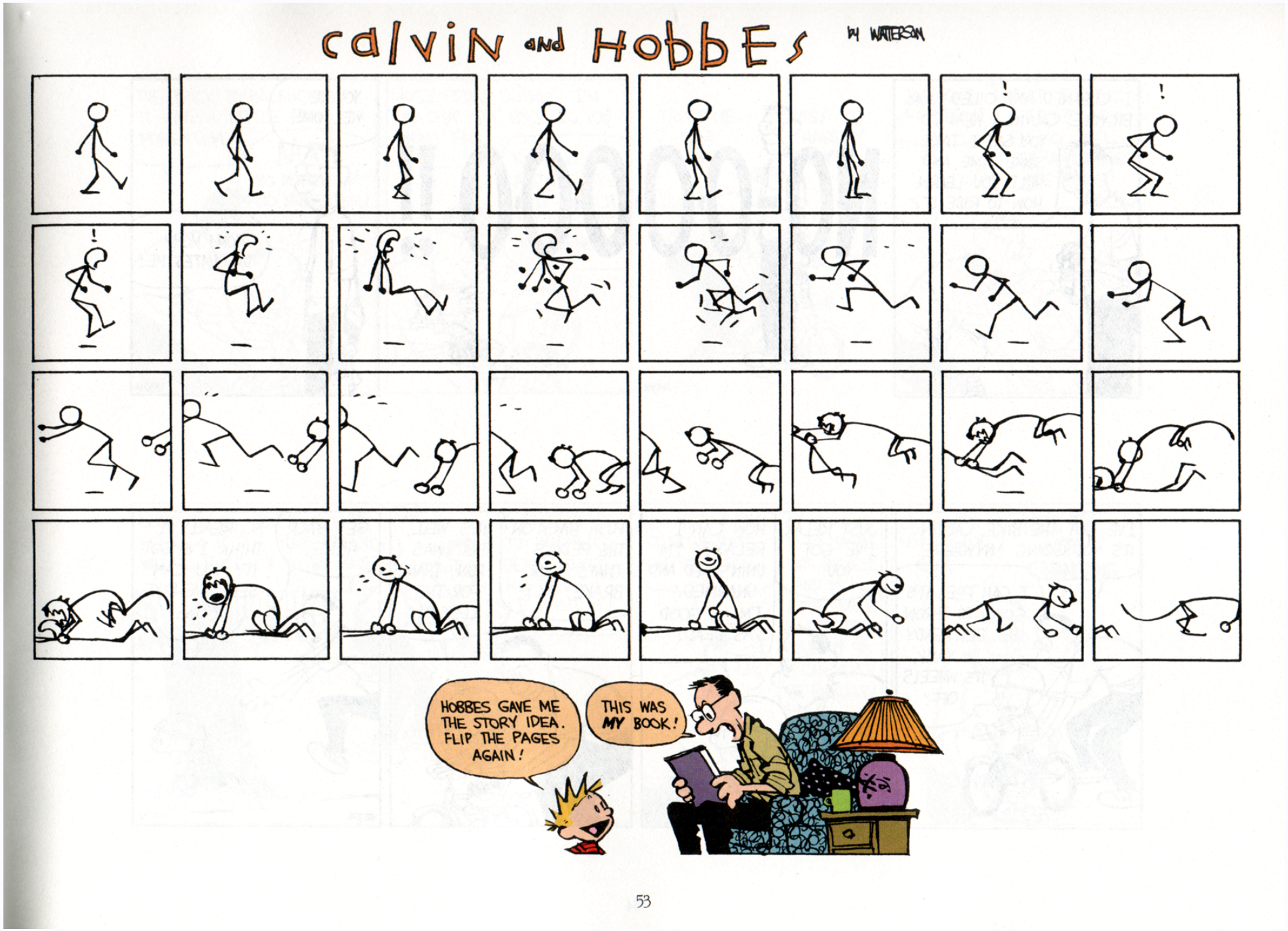
Animation

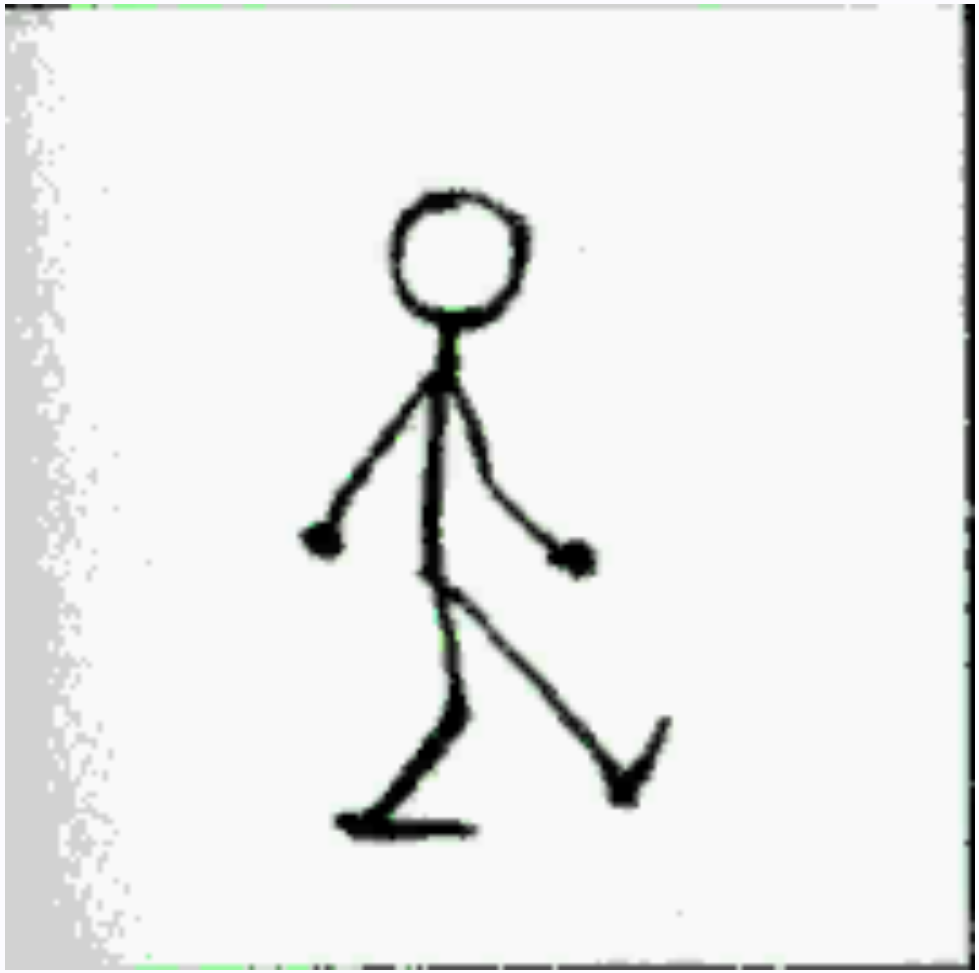
D.A. Forsyth

Animation

- Persistence of vision:
  - The visual system smoothes in time. This means that images presented to the eye are perceived by the visual system for a short time after they are presented. In turn, this means that if images are shown at the right rate (about 20-30 Hz will do it), the next image replaces the last one without any perceived blank space between them.
- Visual closure:
  - a sequence of still images is seen as a motion sequence if they are shown quickly enough - i.e. smooth motion between positions is inferred



From “It’s a magical world”, Watterson, 1996



Basic techniques

- Keyframing:
  - generate frames by drawings, interpolate between drawings
- Stop motion:
  - put model in position, photograph, move, photograph, etc.
- Compositing:
  - generate frames as mixtures of video sequences
- Morphing:
  - mix video sequences while modifying shapes
- Procedural animation:
  - use some form of procedural description to move object
- Motion capture
  - observe motions and generate new ones

Keyframing - issues

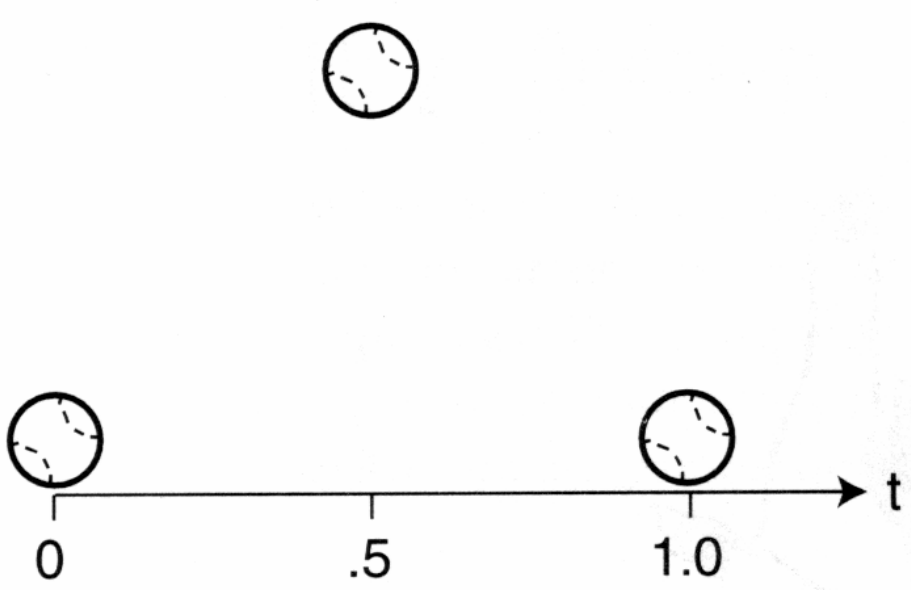


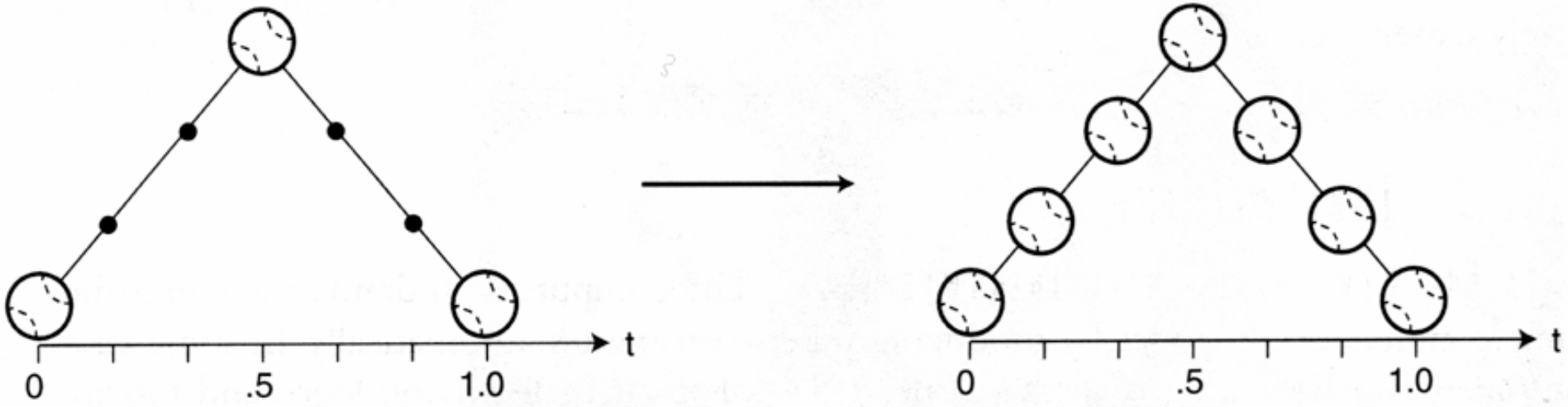
Figure 10.4 Three keyframes. Three keyframes representing a ball on the ground, at its highest point, and back on the ground.

- Generating frames by hand is a huge burden -- 1hr of film is 3600x24 frames
- Skilled artists generate key frames, inbetweeners generate inbetween frames
- Changes are hideously expensive
- Natural interpolation problem -- interpolate various variables describing position, orientation, configuration of objects

From "The computer in the visual arts", Spalter, 1999

Linear interpolation

Figure 10.5 Inbetweening with linear interpolation. Linear interpolation creates inbetween frames at equal intervals along straight lines. The ball moves at a constant speed. Ticks indicate the locations of inbetween frames at regular time intervals (determined by the number of frames per second chosen by the user).



From “The computer in the visual arts”, Spalter, 1999

More complex interpolation

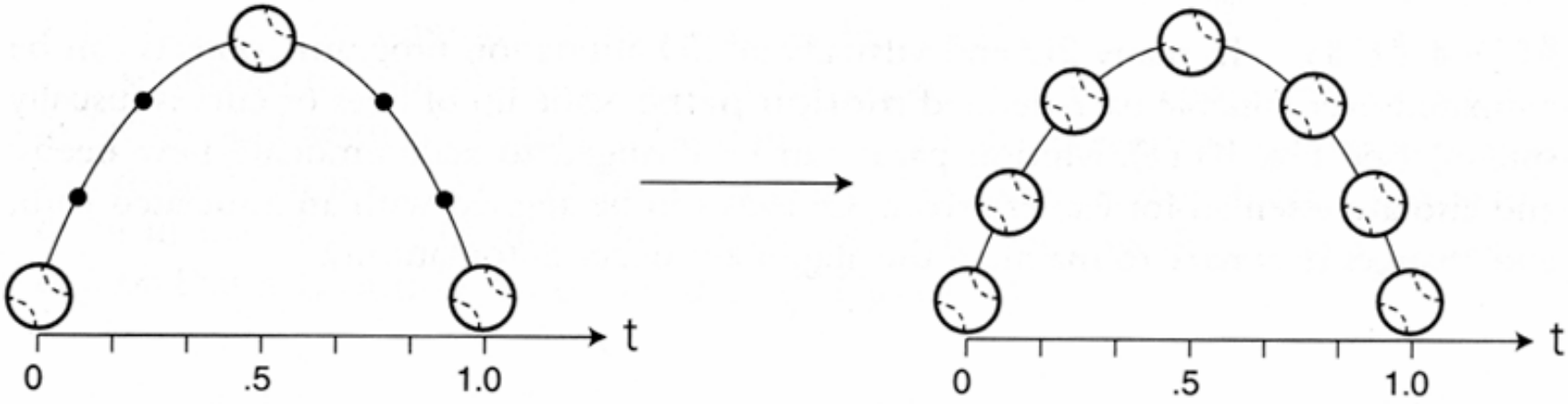


Figure 10.9 Inbetweening with nonlinear interpolation. Nonlinear interpolation can create equally spaced inbetween frames along curved paths. The ball still moves at a constant speed. (Note that the three keyframes used here and in Fig. 10.10 are the same as in Fig. 10.4.)

From "The computer in the visual arts", Spalter, 1999

Modify the parameter, too

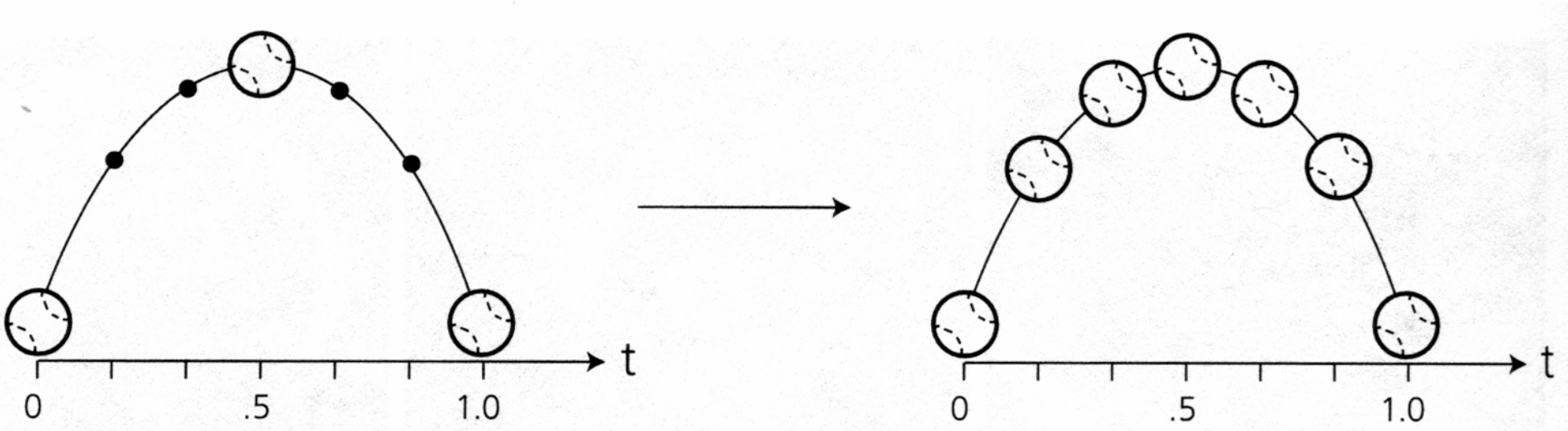
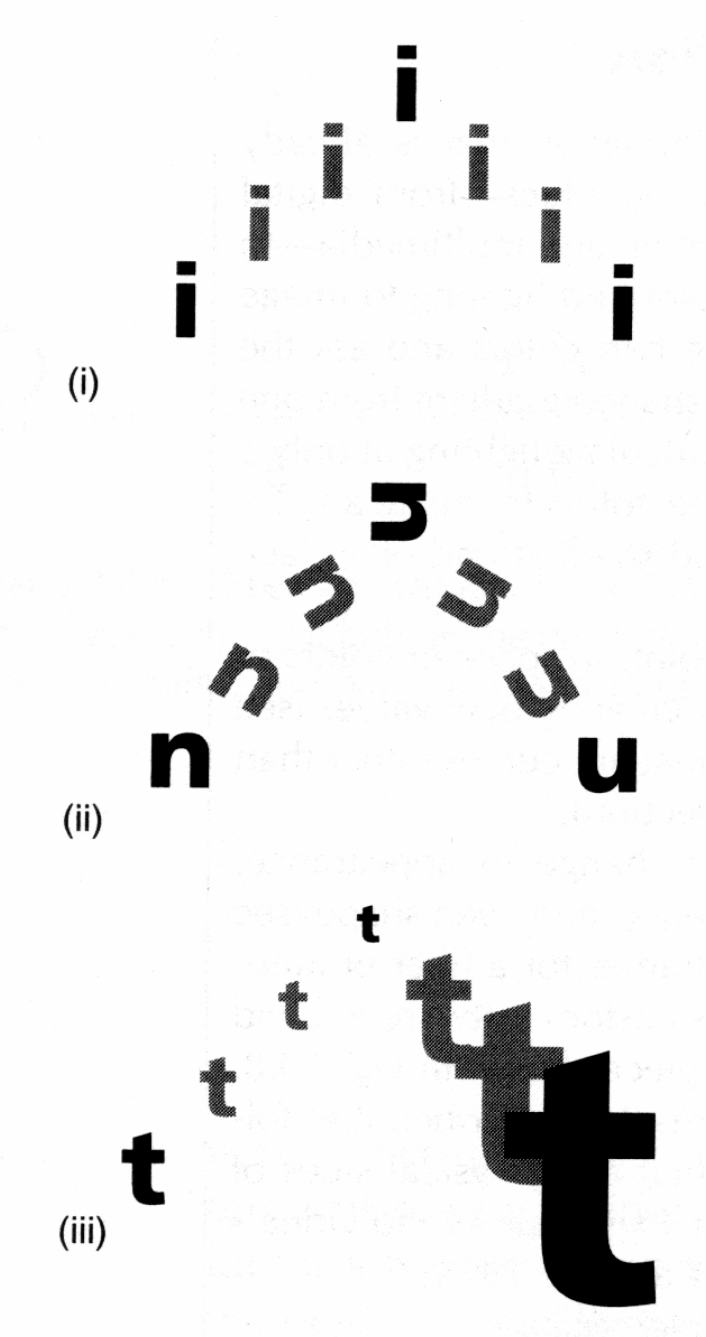


Figure 10.10 Inbetweening with nonlinear interpolation and easing. The ball changes speed as it approaches and leaves keyframes, so the dots indicating calculations made at equal time intervals are no longer equidistant along the path.

A use for parameter continuous interpolates here.
Notice that we don't necessarily need a physical ball.

From "The computer in the visual arts", Spalter, 1999

A variety of variables can be interpolated

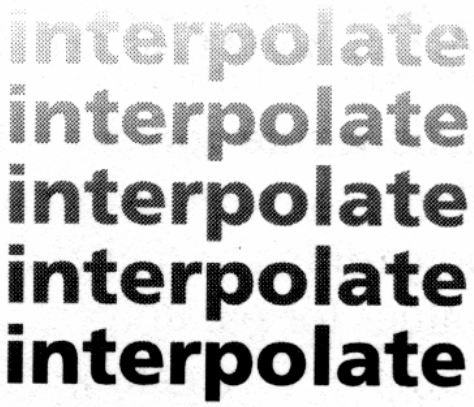


Position

Position and orientation

Position and scale

From “The computer in the visual arts”, Spalter, 1999



Grey-level

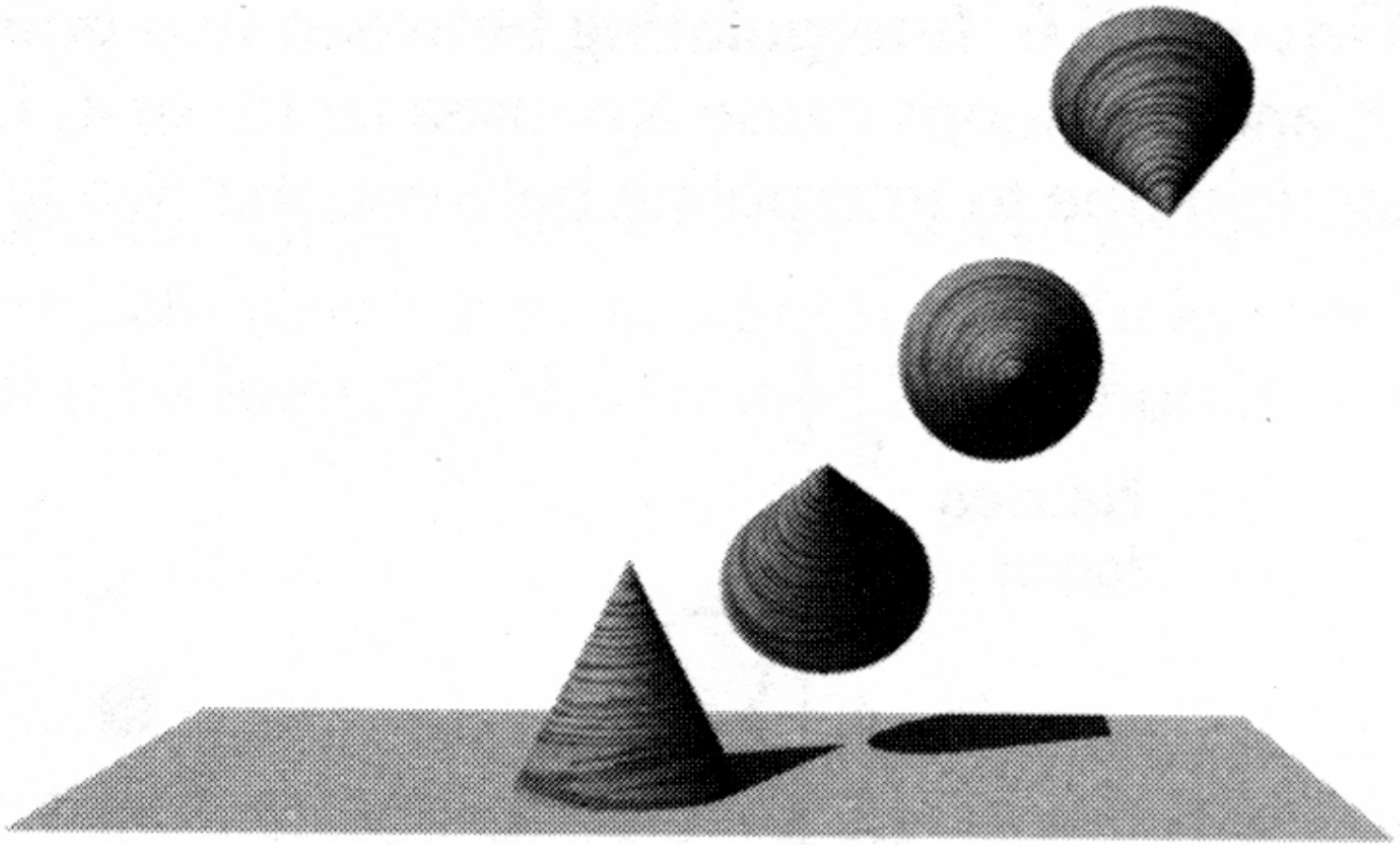
(iv)

Shear

(v)

Shape

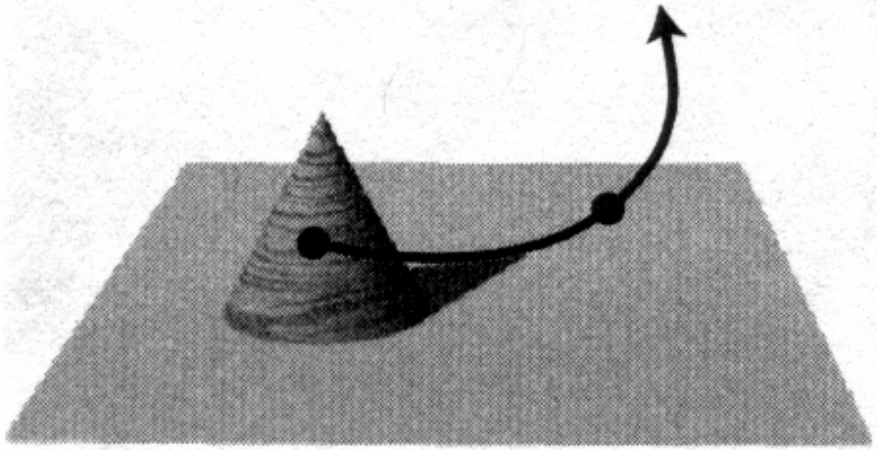
(vi) From “The computer in the visual arts”, Spalter, 1999

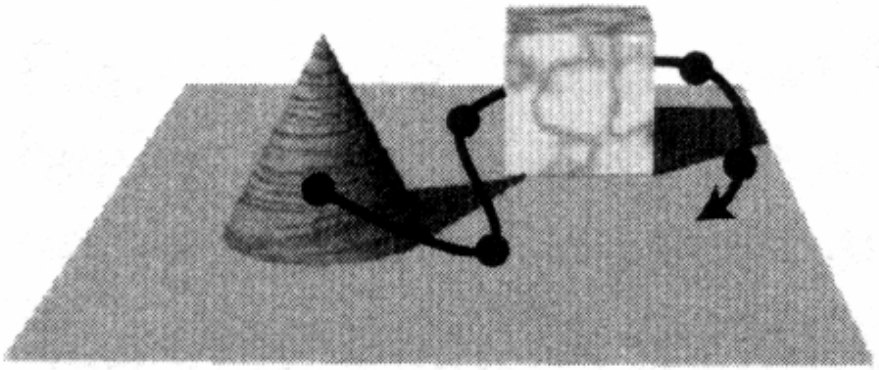


Position and orientation:

note that the position travels along a motion path

From “The computer in the visual arts”, Spalter, 1999

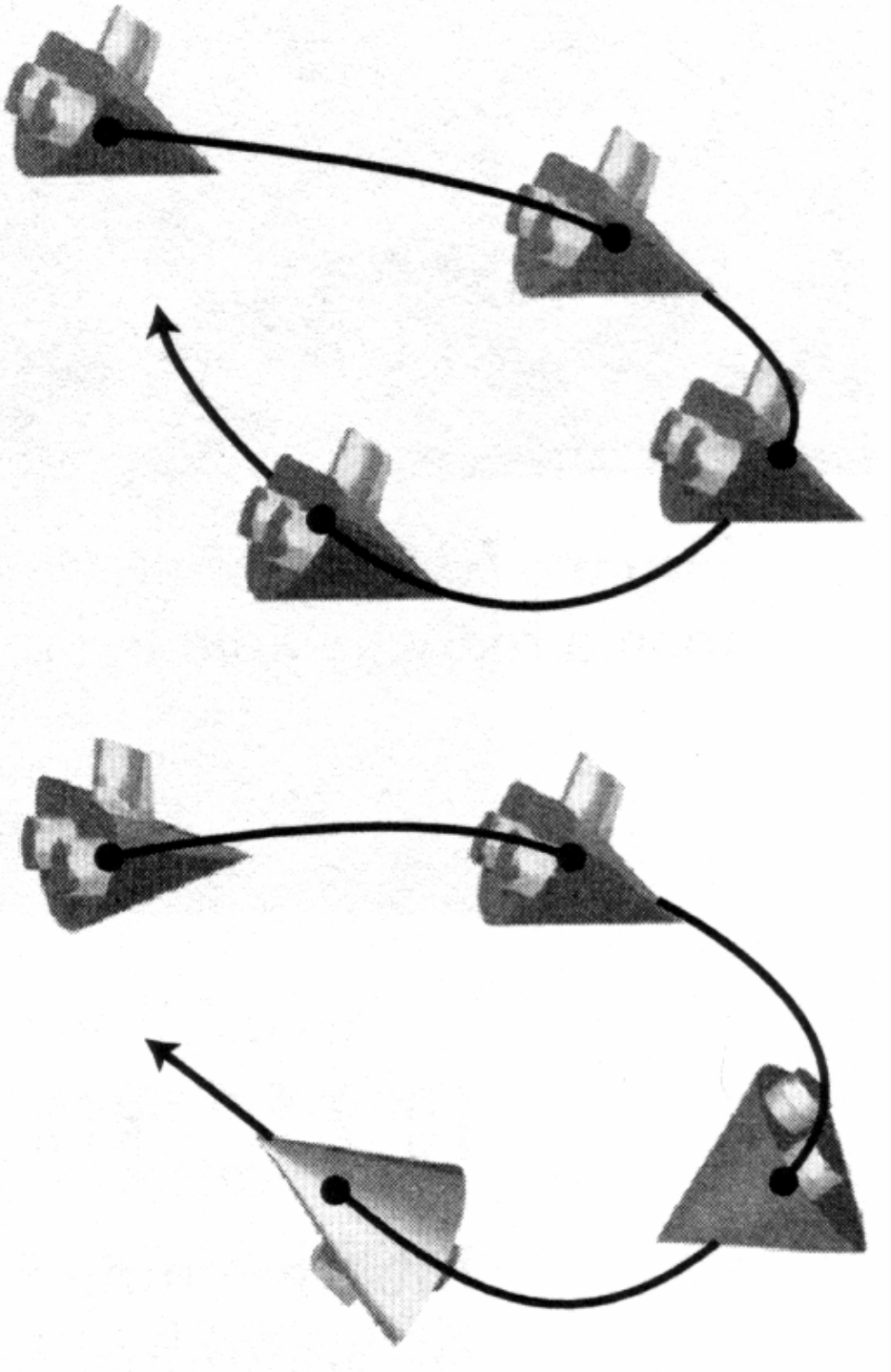




Various path specifications:

perhaps by interactive process;
two issues:
building the path
where are the keyframes?

From “The computer in the visual arts”, Spalter, 1999

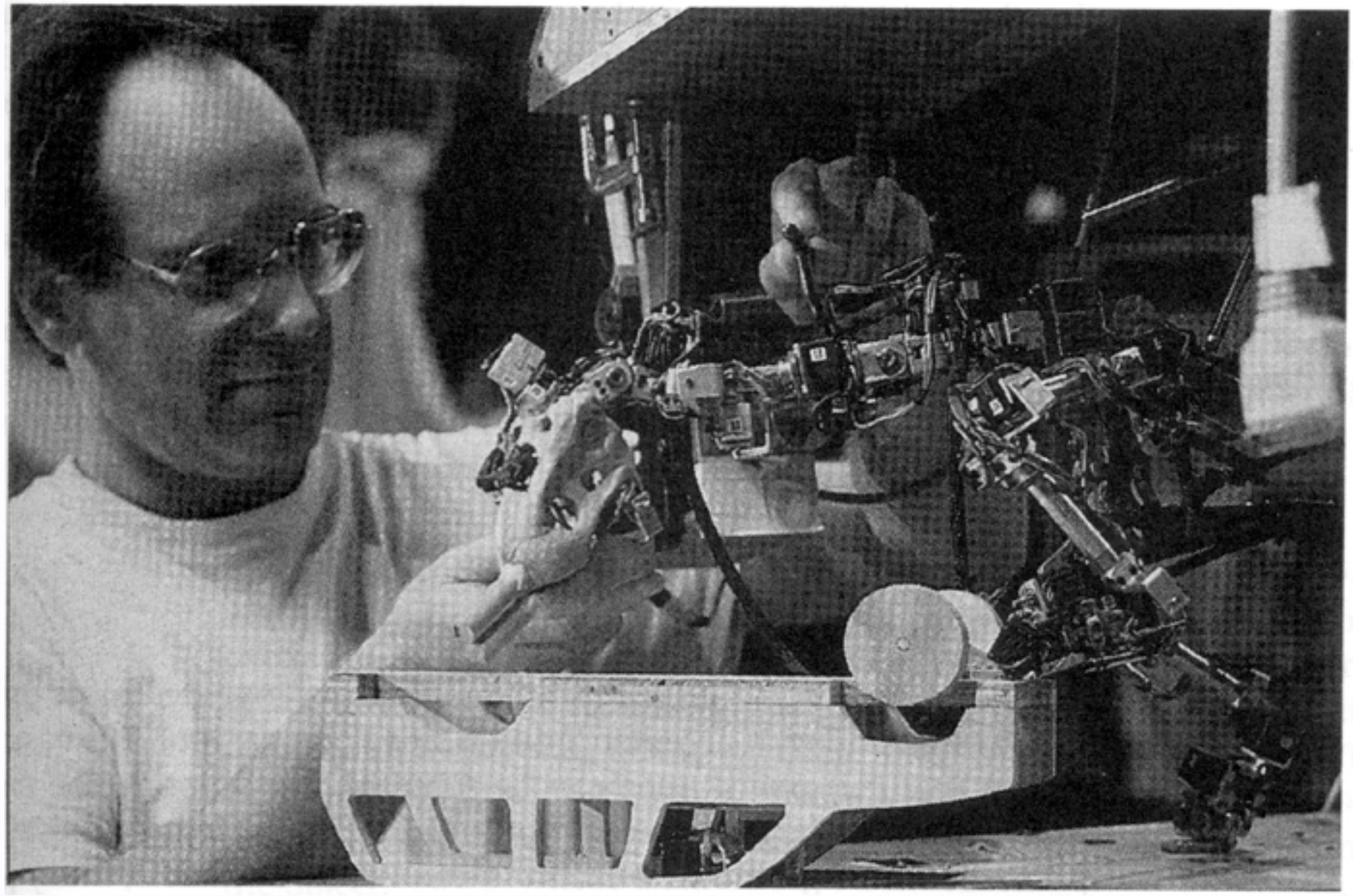


Interpolating orientation gives greater realism. Notice that the tangent to the motion path gives a great cue to the orientation of the object.

From “The computer in the visual arts”, Spalter, 1999

Stop motion

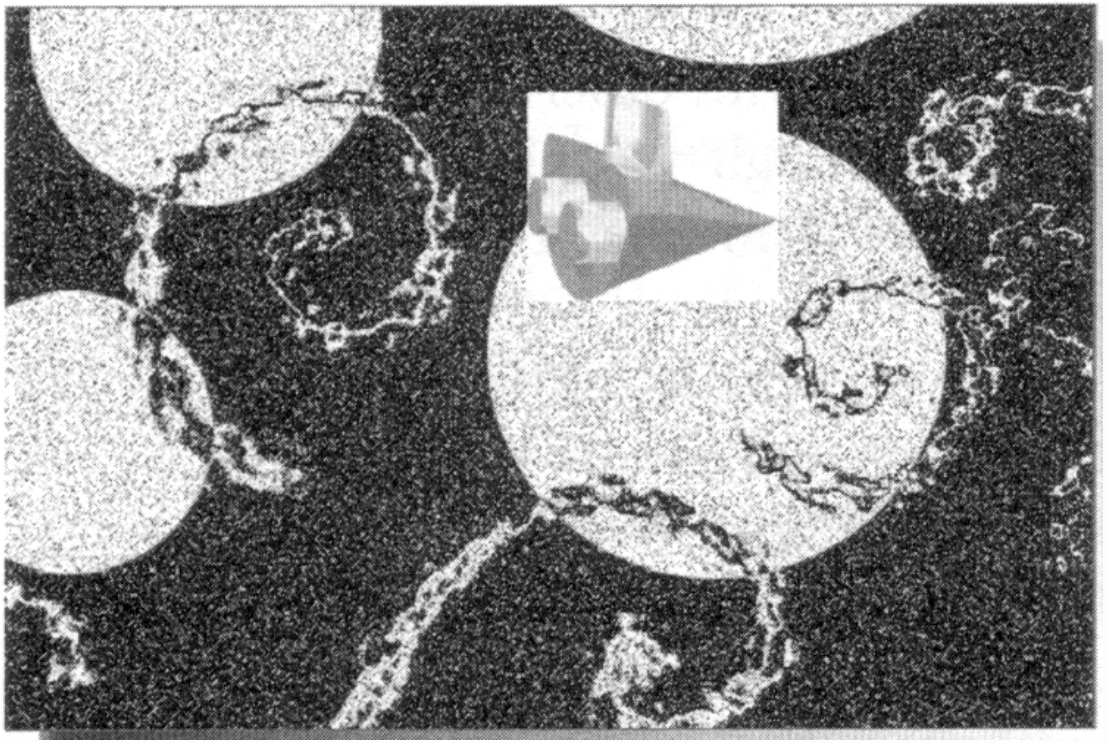
- Very important traditional animation technique
- Put model in position, photograph, move, photograph, etc. e.g. “Seven voyages of Sinbad”, “Clash of the titans”, etc.
  - Model could be
    - plastic
    - linkage
    - clay, etc.
- Model work is still very important e.g. “Men in Black”
- Computerizing model work is increasingly important
  - issue: where does configuration of computer model come from?



From “The computer Image”,Watt and Policarpo, 1998

Compositing

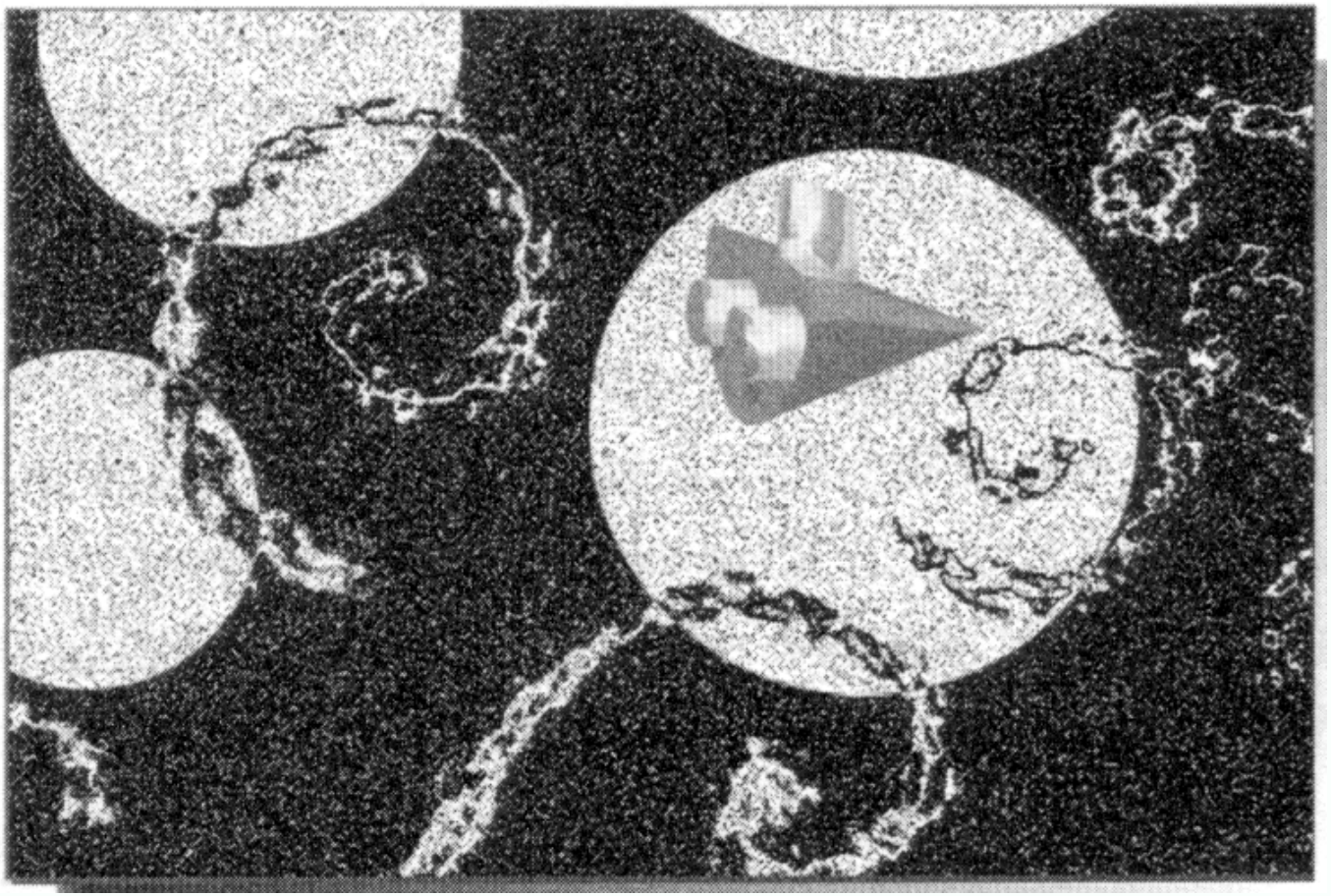
- Overlay one image/film on another
  - variety of types of overlay



Simple overlay - spaceship pixels replace background pixels

From “The computer in the visual arts”, Spalter, 1999

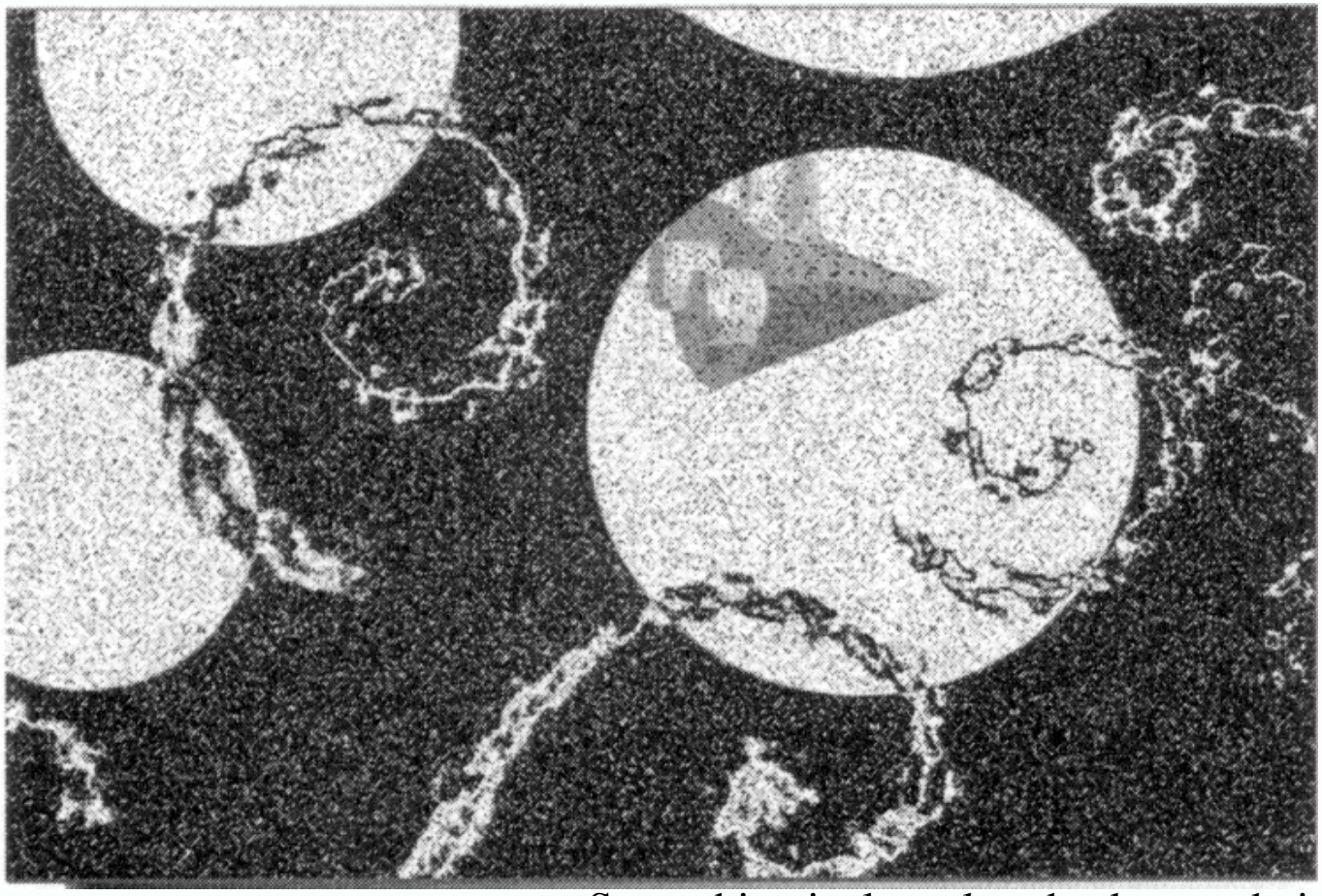
Compositing



Spaceship pixels replace background pixels if they are not white (white is “dropped out”)

From “The computer in the visual arts”, Spalter, 1999

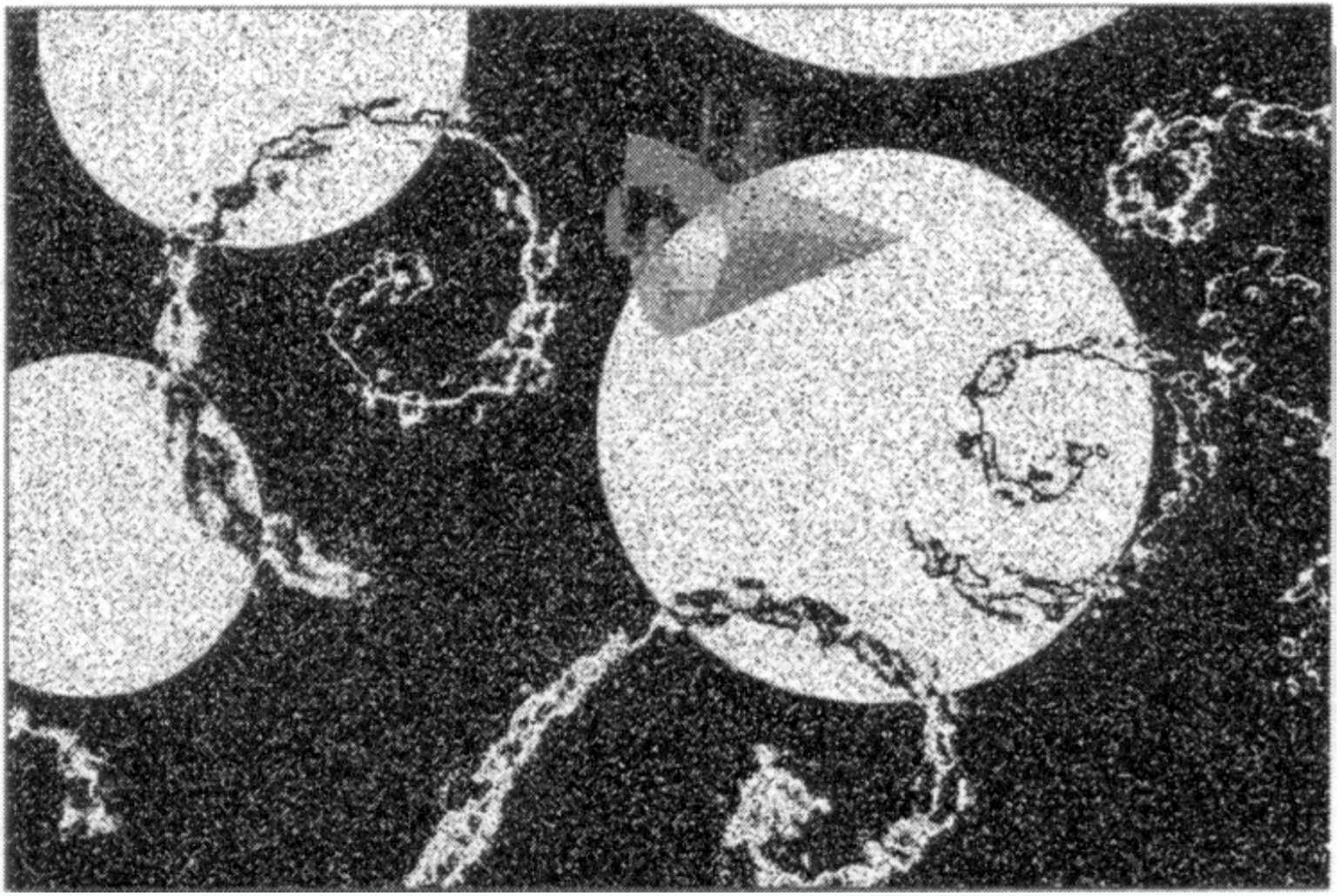
Compositing



Spaceship pixels replace background pixels if they are darker

From “The computer in the visual arts”, Spalter, 1999

Compositing

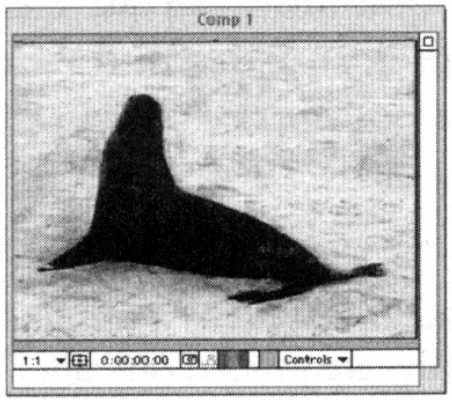


Light areas are more transparent - blending

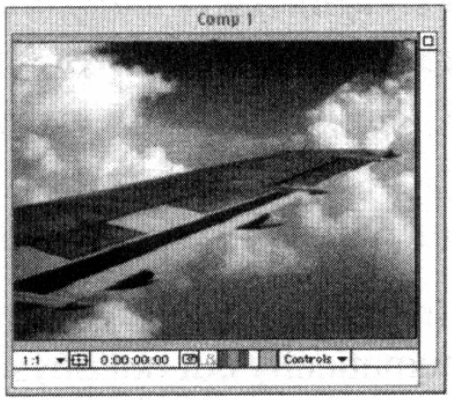
From “The computer in the visual arts”, Spalter, 1999

Compositing

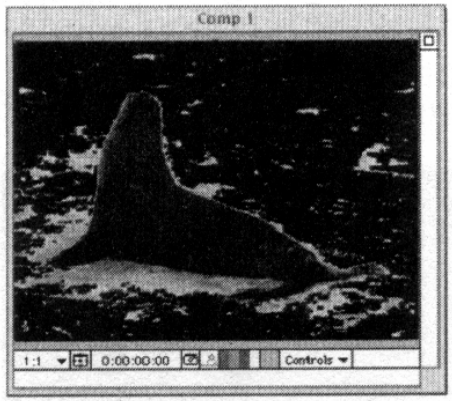
(a)



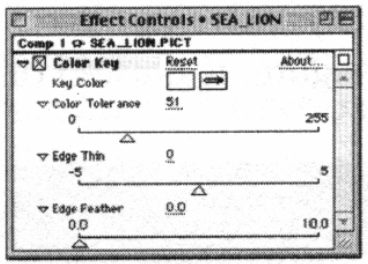
Original image



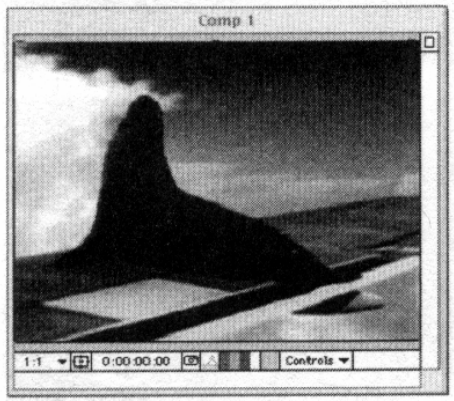
Underlying image



Background dropped out



Color key controls



Final effect

- Note that human intervention might be required to remove odd pixels, if the background doesn't have a distinctive colour
- One can buy sets of images which have been segmented by hand.

From "The computer in the visual arts", Spalter, 1999

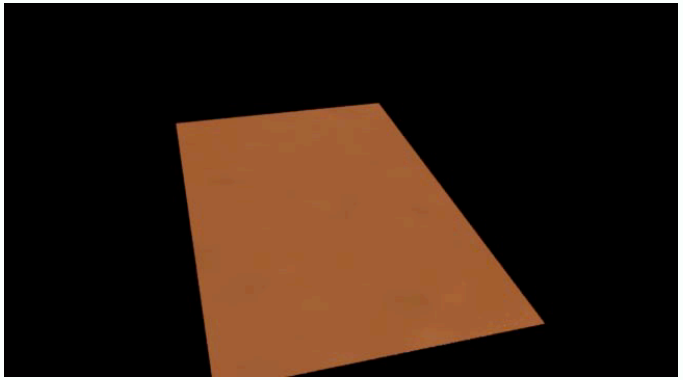
Compositing

- Insertion
  - we want to insert an object into a scene
  - we have
    - background scene — image is: B
    - model of background scene — image is: Mn
    - model of object in background scene — image is: Mo
- Composite by:
  - at model pixels
    - B+(Mo-Mn)
  - at object pixels
    - Mo
  - at background pixels
    - B

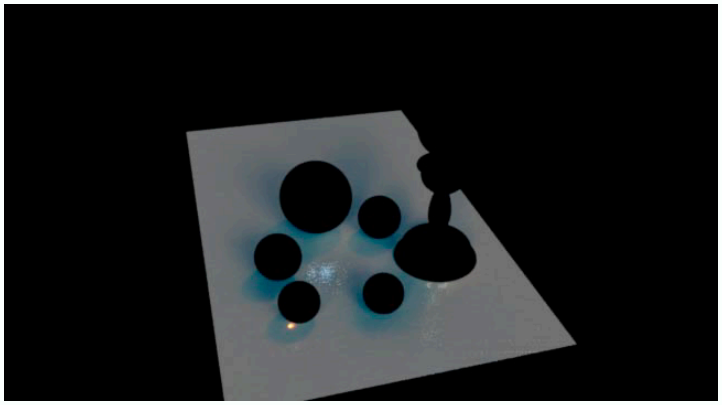
Compositing

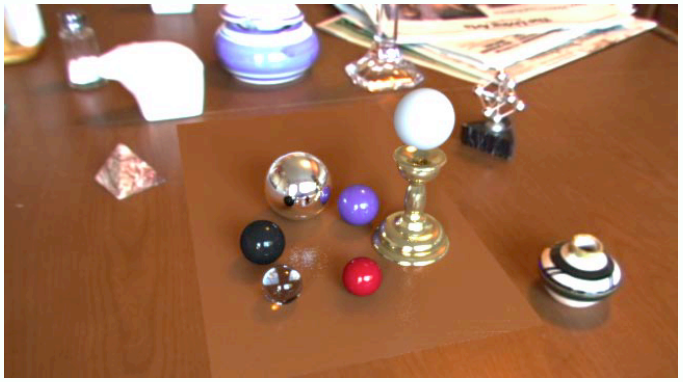
Background image B

Background model Mn



Mo-Mn
in non-object, non-
background pixels

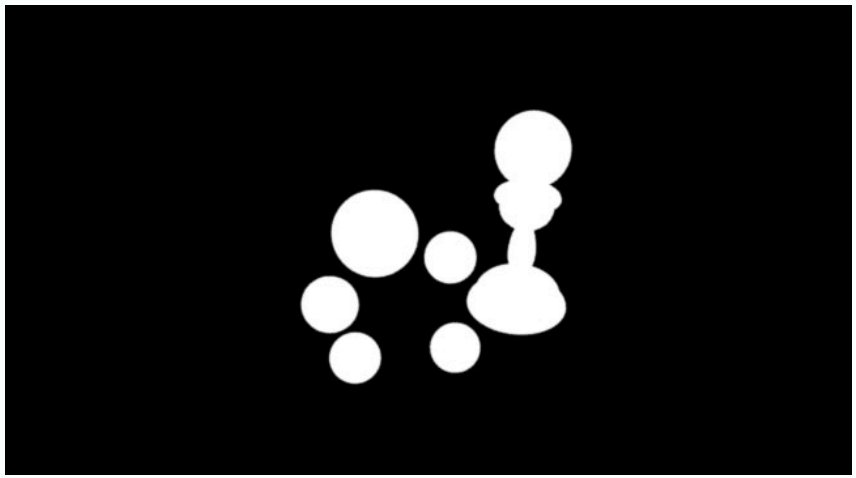




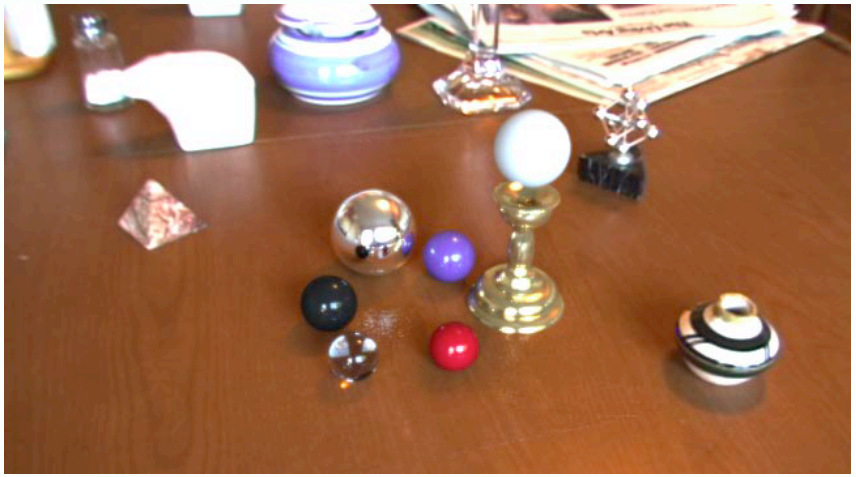
Background model,
rendered with objects Mo,
superimposed on B

Figures from Debevec,
Rendering Synthetic Objects
into Real Scenes:
Bridging Traditional and
Image-based Graphics with
Global Illumination
and High Dynamic Range
Photography1998

Compositing



Object mask



Final composite

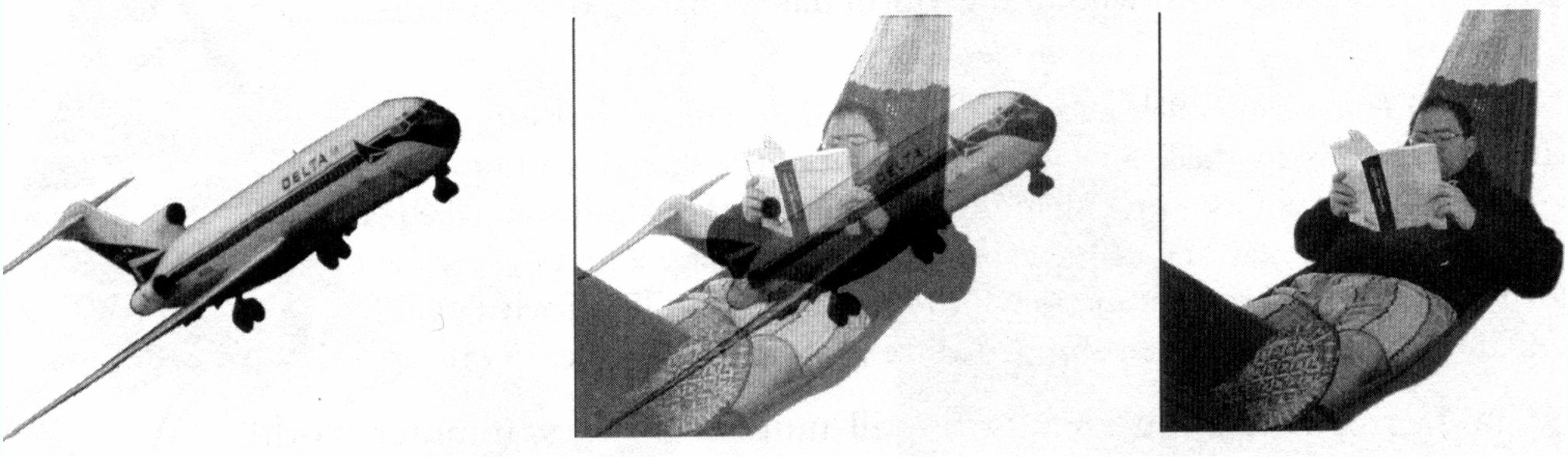
Figures from Debevec, Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-based Graphics with Global Illumination and High Dynamic Range Photography1998

More interesting compositing problems

(a) Foreground scene (b) Background scene (c) Blue screen composite (d) Our method (e) Reference photograph

Figure from Shadow matting and compositing, Chuang et al 2002

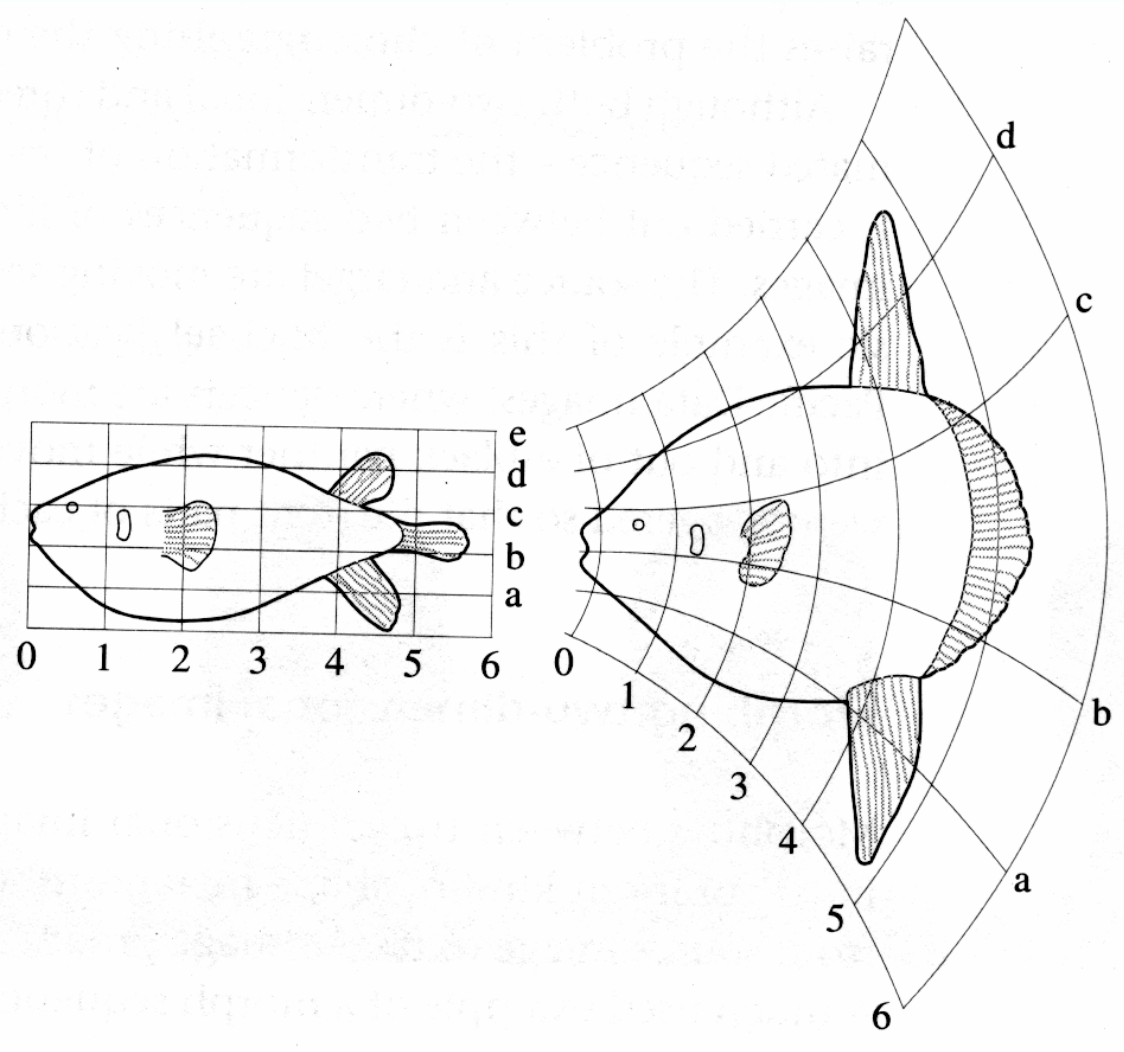
Morphing



- Simple blending doesn't work terribly well for distinct shapes
- Idea: map the one shape to the other, while blending

From "The computer Image",
Watt and Policarpo, 1998

Morphing

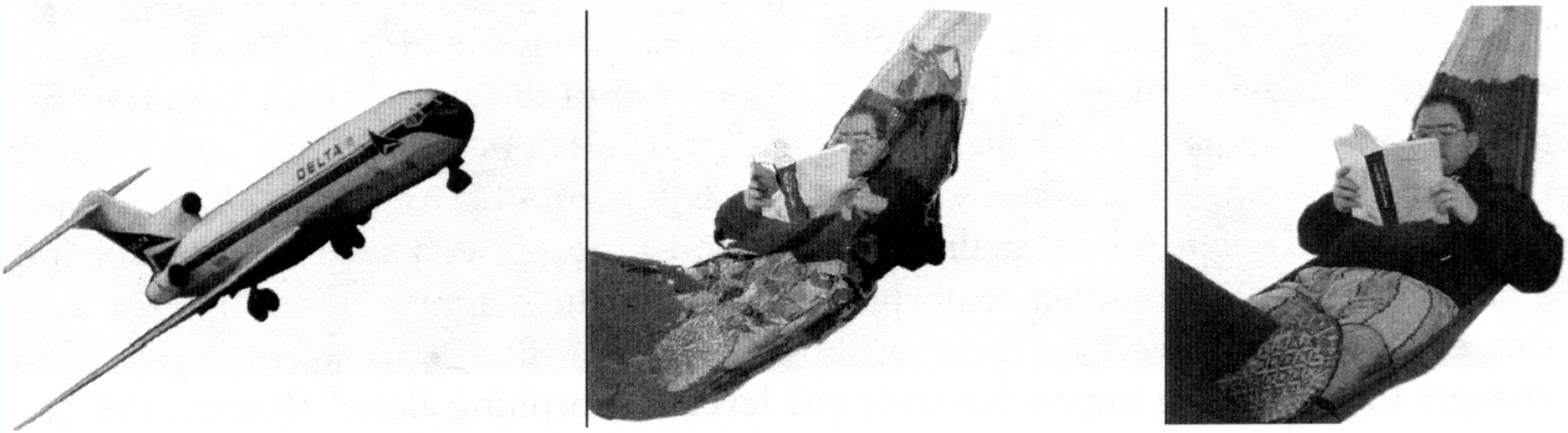


From “On growth and Form”, D’Arcy Thompson

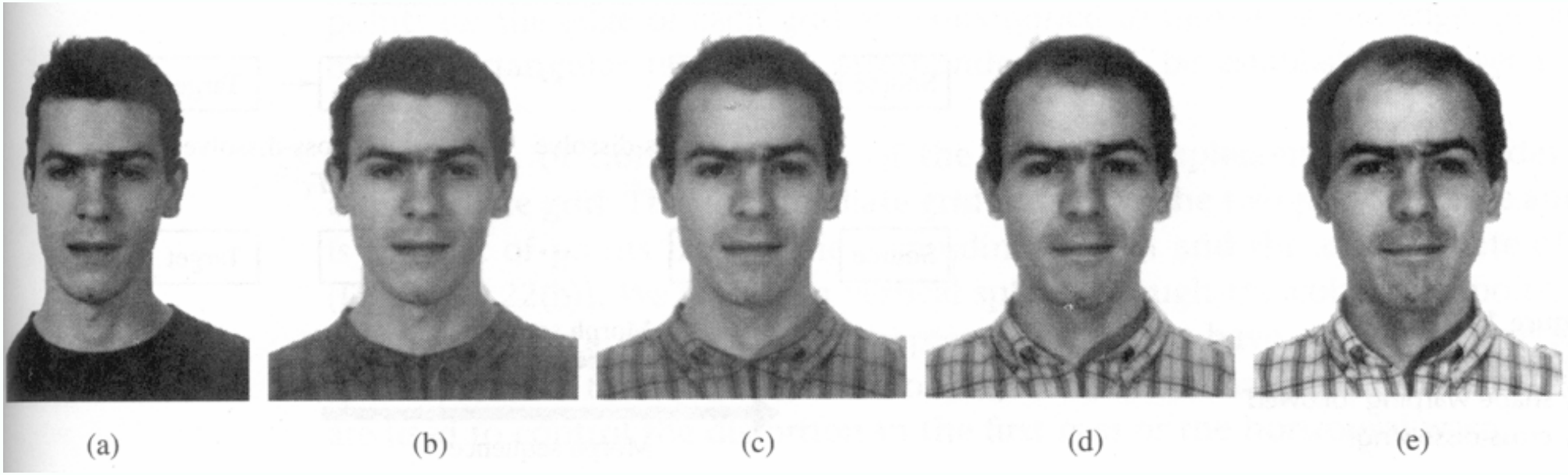
Morphing

- Build a spatial deformation

- From “The computer Image”,
- Watt and Policarpo, 1998



Morphing



From “The computer Image”,
Watt and Policarpo, 1998

Procedural Animation

- Key idea:
  - Algorithms yield motion
    - Inspirations:
      - insight
      - physics
      - simplified physics
    - Easily guessed algorithm gives good results
      - waves
      - terrain
      - l-systems (for plants)
      - finite state machines (for character control)
- And the winners are particles
  - because they're easy to model
  - and they don't interact, as we shall see

A single particle under gravity

- State:
  - position x, velocity v
  - gravitational acceleration is a
    - (all vectors)
- Motion is governed by two differential equations:

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}$$

$$\frac{d\mathbf{v}}{dt} = \mathbf{a}$$

- subject to initial conditions
  - x(0)=x0; v(0)=v0

Particle in a potential field

- Imagine there is some potential energy
  - usually depends on position only
  - gravity can be represented like this (remember gravitational potential?)
  - write potential as:

$$\phi(\mathbf{x})$$

- Now:
  - the particle has mass m
  - experiences force:

$$-\nabla\phi(\mathbf{x}) = -\begin{pmatrix} \frac{\partial\phi}{\partial x} \\ \frac{\partial\phi}{\partial y} \\ \frac{\partial\phi}{\partial z} \end{pmatrix}$$

Particle in a potential field

- We get equations:

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}$$

$$\frac{d\mathbf{v}}{dt} = \frac{-\nabla\phi(\mathbf{x})}{m}$$

  - subject to the same i.c.'s

Particle in a potential field

- Assume
  - we know position, velocity at time: t

$$\mathbf{x}_t, \mathbf{v}_t$$

  - want position, velocity at time: $t + \Delta t$

$$\mathbf{x}_{t+\Delta t}, \mathbf{v}_{t+\Delta t}$$

- Have:

$$\mathbf{x}_{t+\Delta t} \approx \mathbf{x}_t + \Delta t \mathbf{v}_t$$

$$\mathbf{v}_{t+\Delta t} \approx \mathbf{v}_t - \Delta t \frac{\nabla \phi(\mathbf{x}_t)}{m}$$

Particle in a potential field

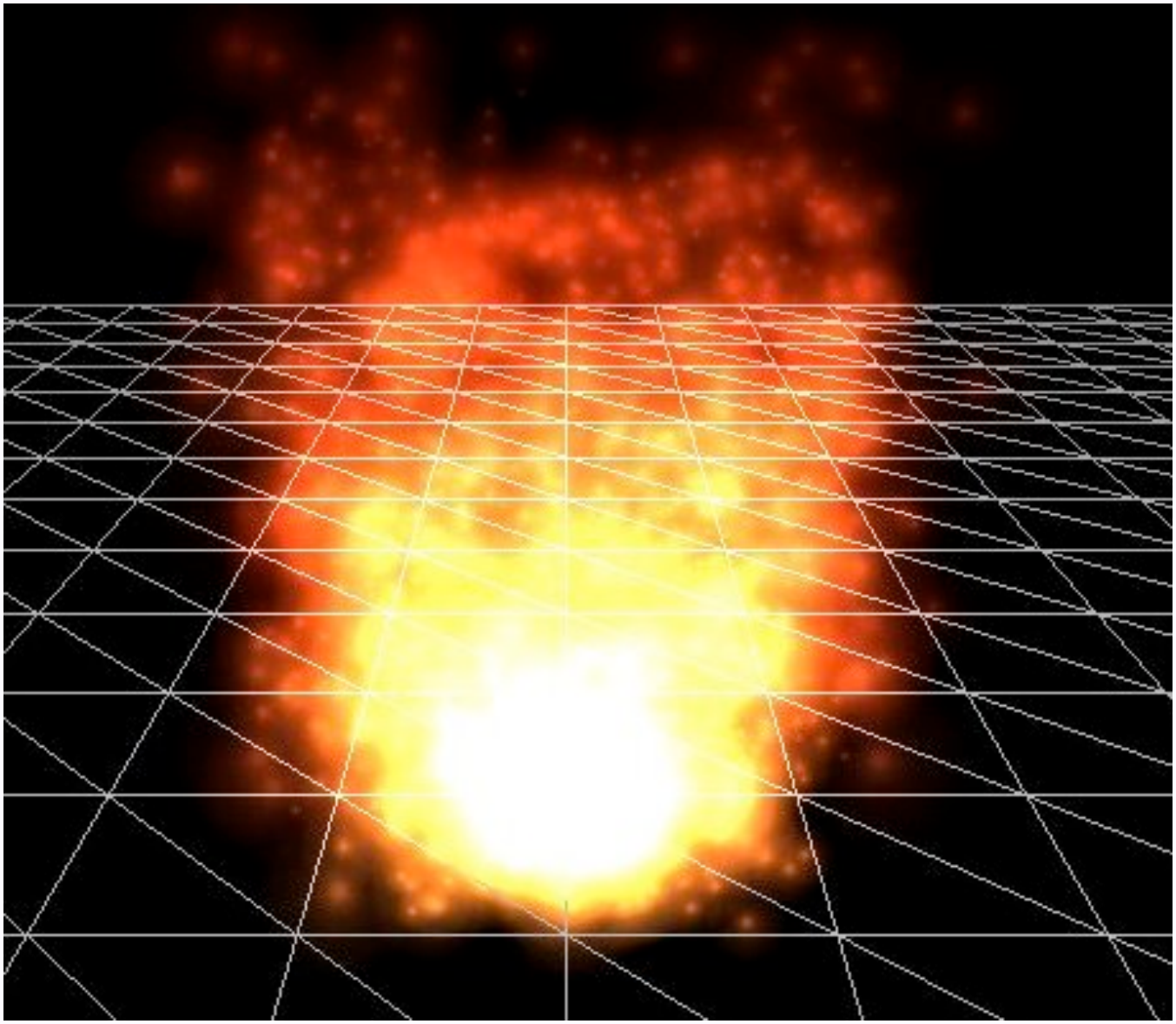
- Previous slide is forward Euler method
- Any other method for solving ODE's applies
  - and there are lots of better methods
  - which you learned in numerical analysis

Procedural Dynamics - Particle systems

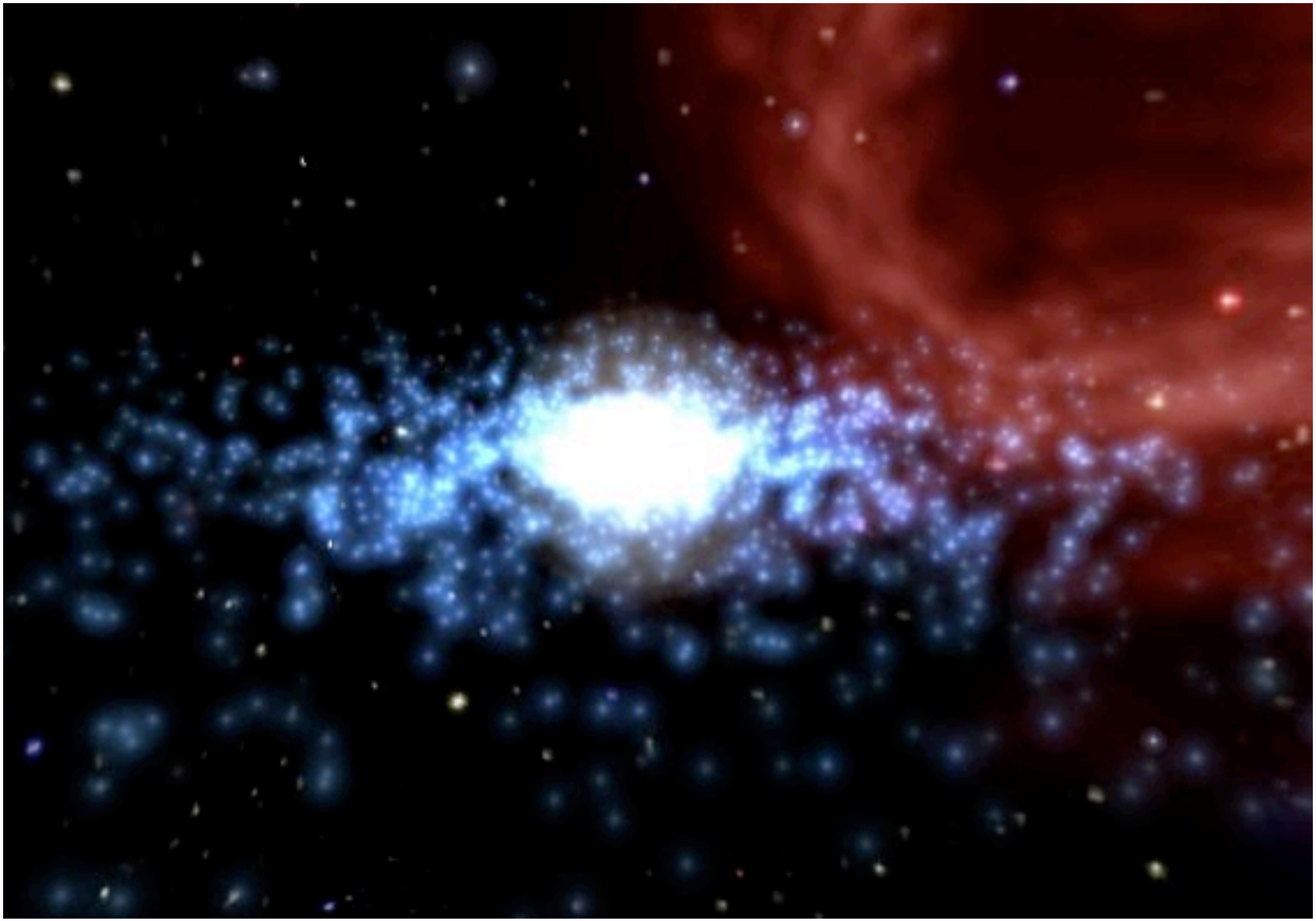
- There is a source of particles
  - move under gravity, sometimes collisions
  - control with potential fields
    - notice it is pretty straightforward to do time-varying potentials
- Example: fireworks
  - particles chosen with random colour, originating randomly within a region, fired out with random direction and lasting for a random period of time before they expire
    - or explode, generating another collection of particles,etc
- Example: water
  - very large stream of particles, large enough that one doesn't see the gap
- Example: grass
  - fire particles up within a tapered cylinder, let them fall under gravity, keep a record of the particle's trail.

Now replace particle centers with small blobs of colour in the image plane

http://www.arch.columbia.edu/manuals/Softimage/3d_learn/GUIDED/PARTICLES/p_first.htm



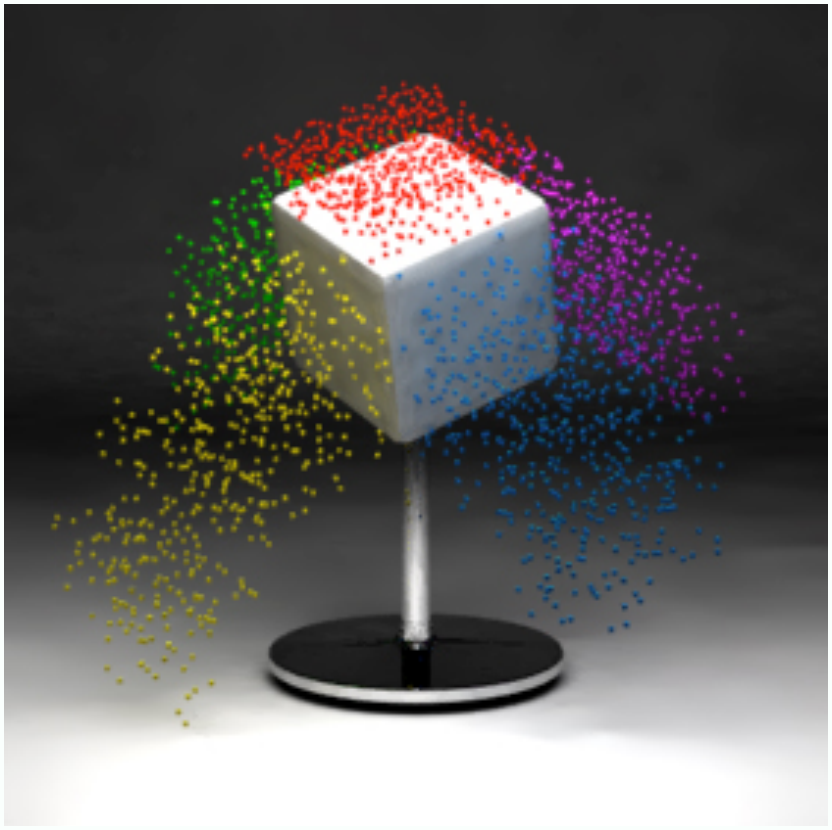
By John Tsiombikas from Wikipedia



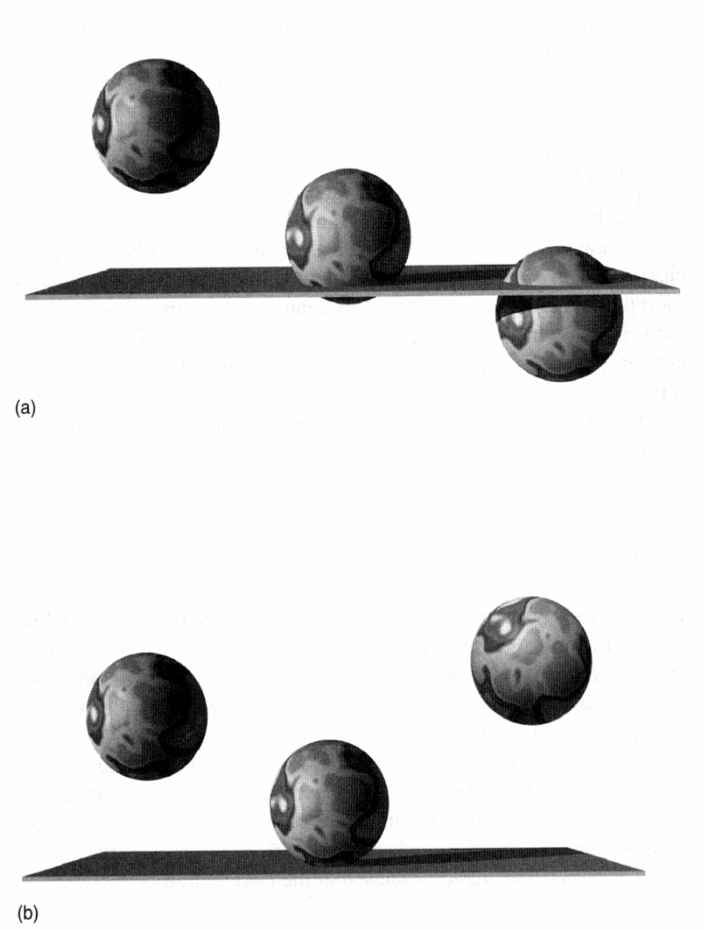
By John Tsiombikas from Wikipedia

Particle system videos

Strands



Ballistic + Collision



- Objects move freely under gravity until they collide.
- For accurate physical models, order in which collisions occur is important.

Collision detection

- Particles are straightforward
  - -ish (geometry is easy)
  - issues: undetected collisions
    - strategies:
      - take fixed time steps, fixup collision
        - but we may not be able to tell a collision has occurred!
      - potential barrier
        - but this may force quite small time steps; stiffness
        - backward Euler helps, but only within limits
      - identify safe bounds within which to advance time, search
        - use priority queue
        - but this may force quite small time steps

Collision detection

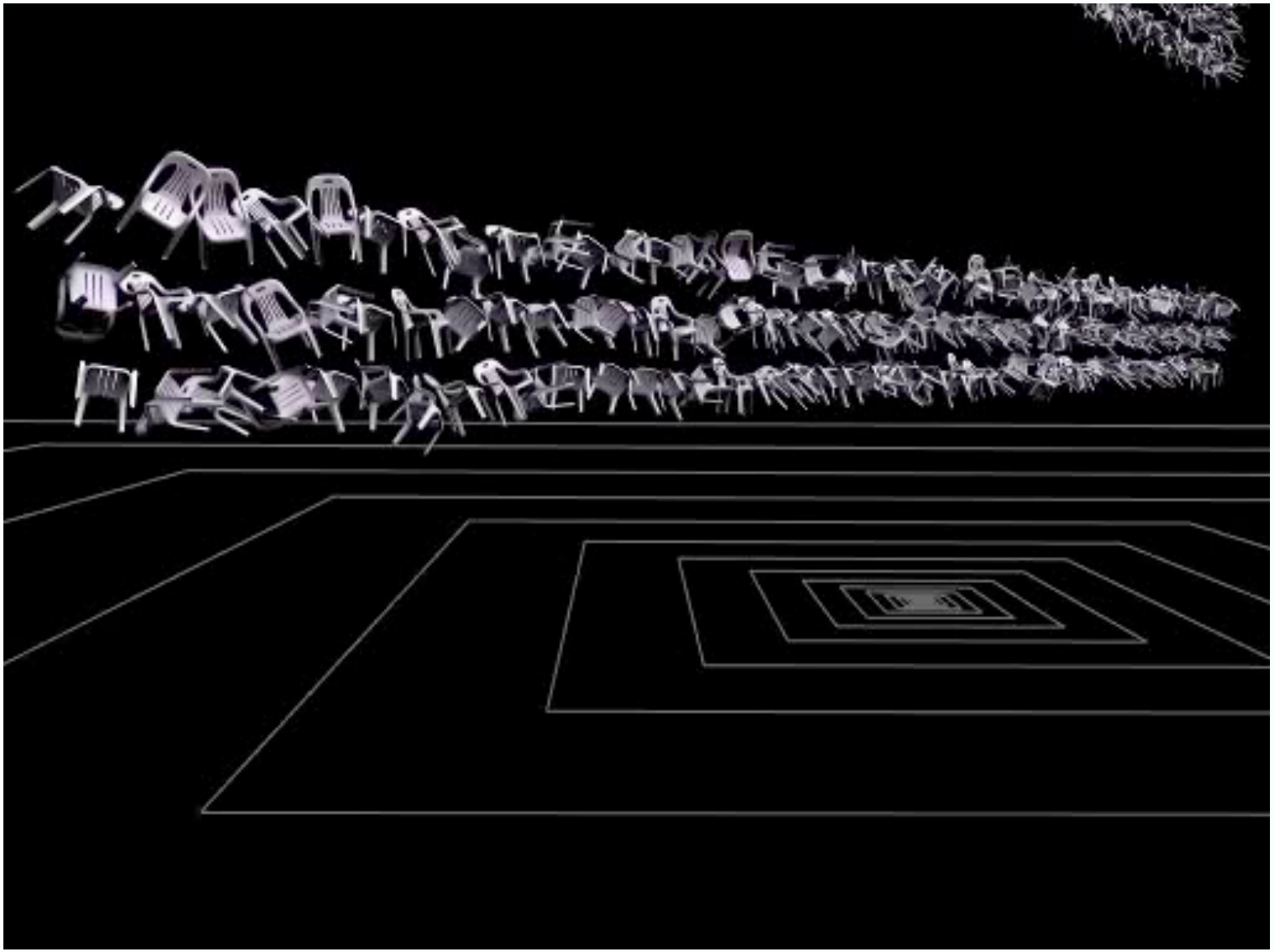
- Rigid objects
  - (safe bounds strategy)
  - We decide that objects closer than some small distance have collided
  - Problem:
    - we have a geometric representation
    - which faces/vertices are closer than epsilon?
  - Strategy:
    - prune with spatial data structures
      - axis aligned bounding boxes, BSP trees, etc.
    - test results exhaustively
      - triangle test is easiest case

Collision detection

- Two non-intersecting triangles can be separated by a plane
  - assume they're not coplanar
    - then can look at 6 choose 3 = 20 planes obtained by choosing 3 verts
    - extra work if they're coplanar
      - separating plane is normal to triangle plane
    - appropriate choice of plane yields distance between triangles
  - Improvement
    - Gilbert-Johnson-Keerthi algorithm, using support functions
    - open source version due to Stephen Cameron
      - http://www.comlab.ox.ac.uk/stephen.cameron/distances/

12,201 chairs;
218,568,714 triangles

Collision is solved



Doug L. James and Dinesh K. Pai, BD-Tree: Output-Sensitive Collision Detection for Reduced Deformable Models, ACM Transactions on Graphics (ACM SIGGRAPH 2004), 23(3), 2004.

Collisions - resolution

- Strategies:
  - potential field
  - explicit collision model
    - state_out=F(state_in, physical parameters)
    - typical physical parameters:
      - friction, coefficient of restitution
  - data driven
    - match inputs to data, read off outputs
- Collisions
  - produce randomness in motion
  - are a mechanism to control the motion

12 principles of cartoon animation

1. Squash and stretch
2. Anticipation
3. Staging
4. Straight Ahead Action and Pose to Pose
5. Follow Through and Overlapping Action
6. Slow In and Slow Out
7. Arcs
8. Secondary Action
9. Timing
10. Exaggeration
11. Solid Drawing
12. Appeal

Due originally to Frank Thomas and
Ollie Johnston, famous book "The Illusion of Life"
useful discussion at
http://www.animationtoolworks.com/library/article9.html

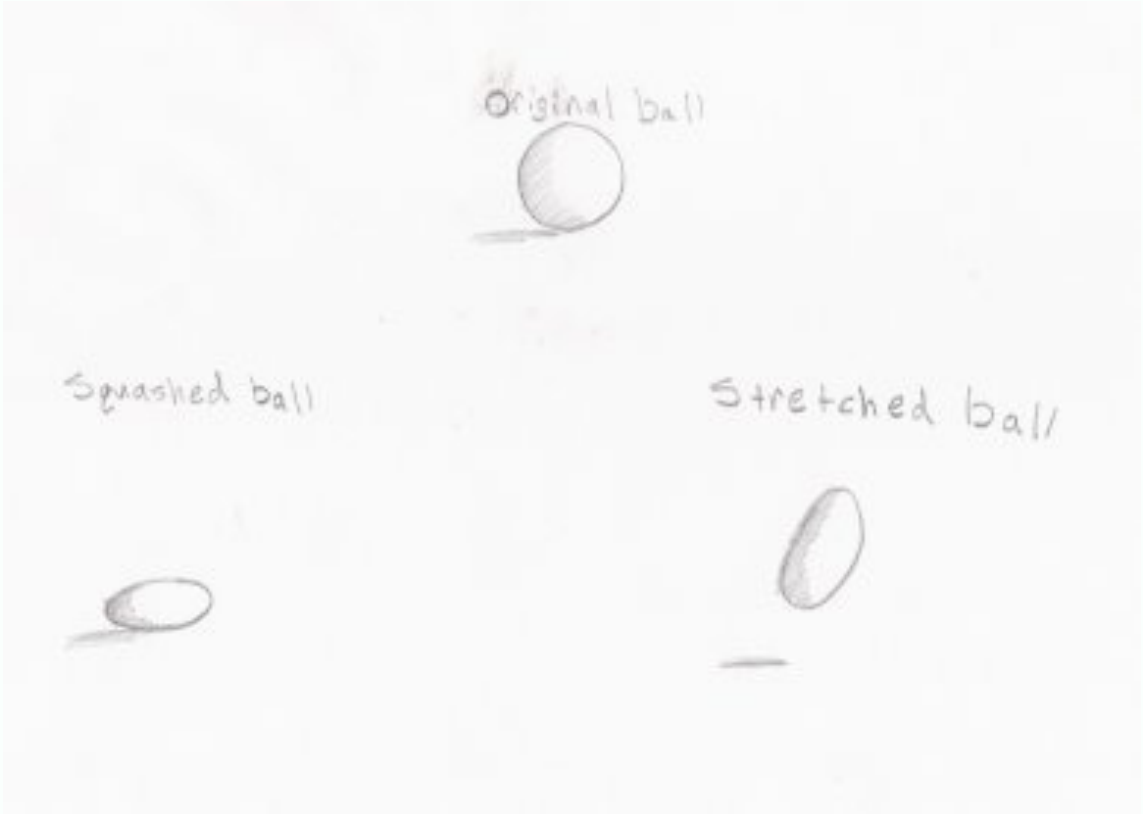
There are some nice youtube movies illustrating these; look at

https://www.youtube.com/watch?v=uDqjIdI4bF4

(or 12 principles of animation)

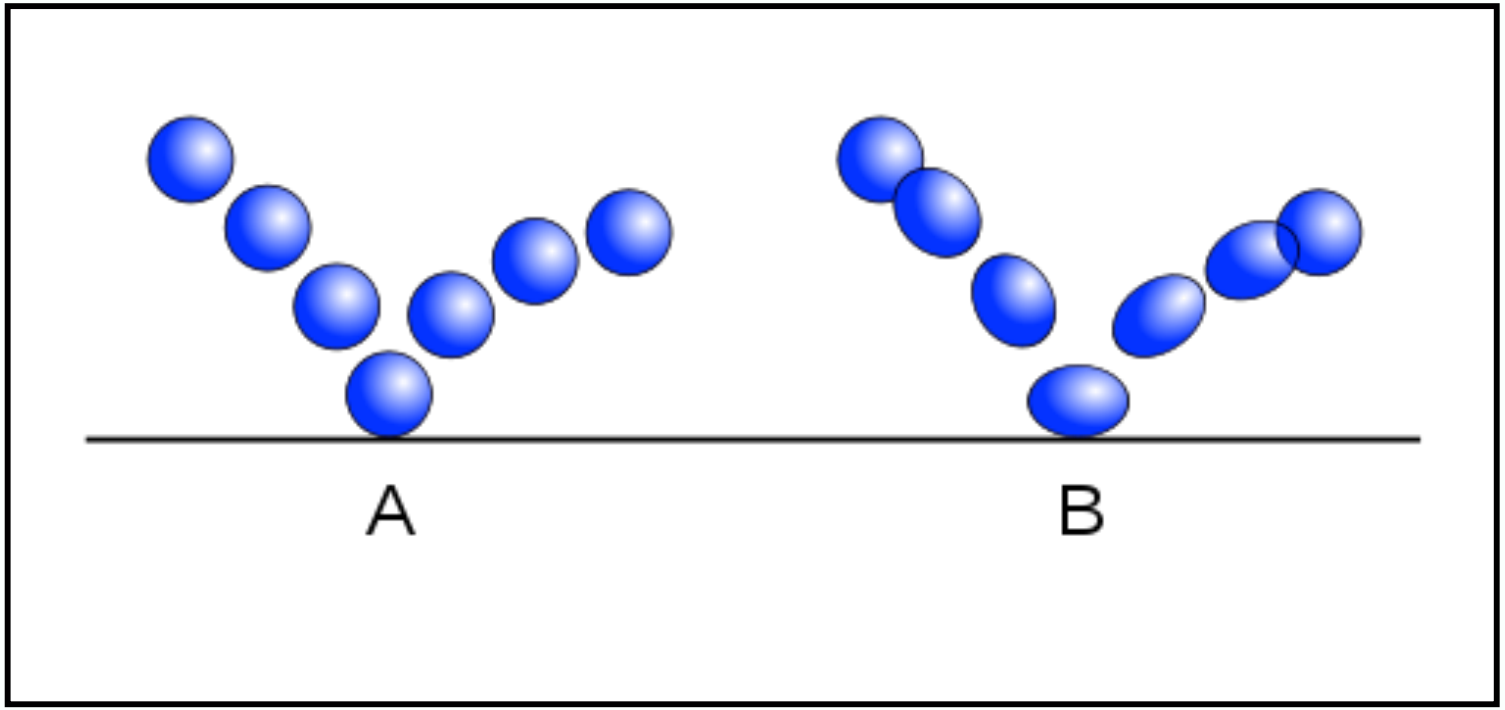
Making collisions more “cartoony”

- Good cartoon animators anticipate and follow through
  - eg a ball hesitates and stretches before it starts moving
  - squashes and overshoots when it finishes



http://en.wikipedia.org/wiki/File:Squash_and_Stretch.jpg

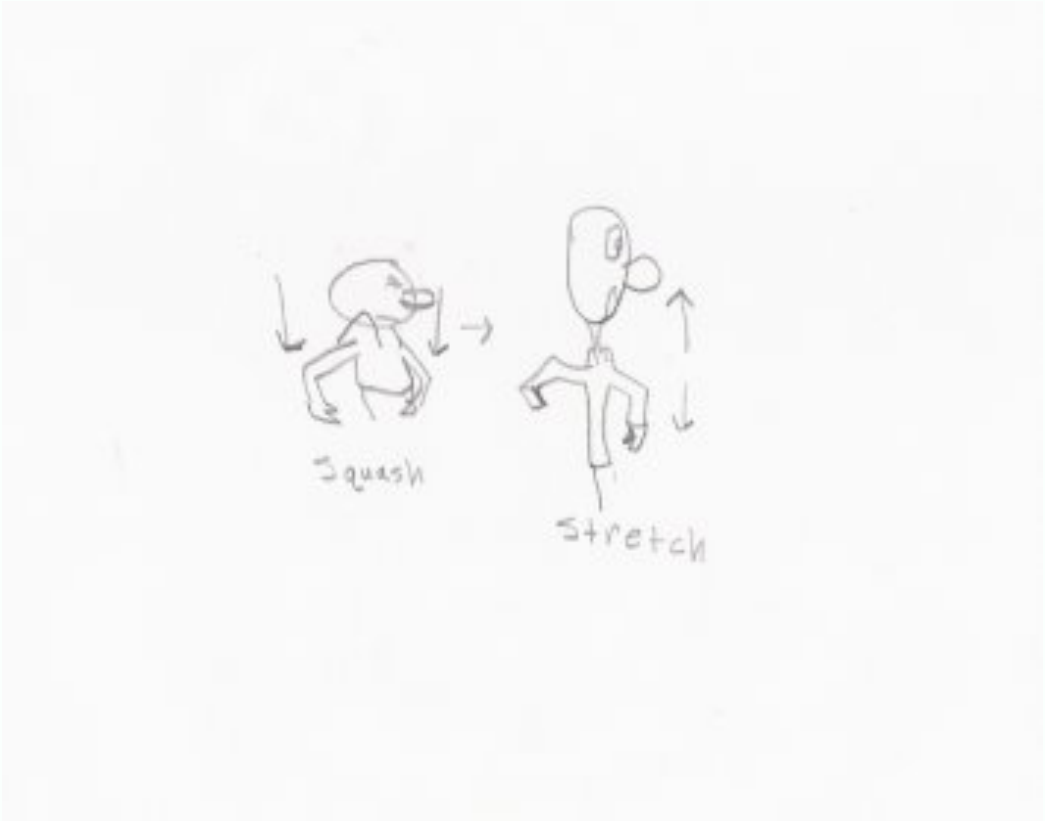
Making collisions more “cartoony”



http://en.wikipedia.org/wiki/File:Squash_and_Stretch.svg

Making collisions more “cartoony”

- and you can apply this to characters, too...

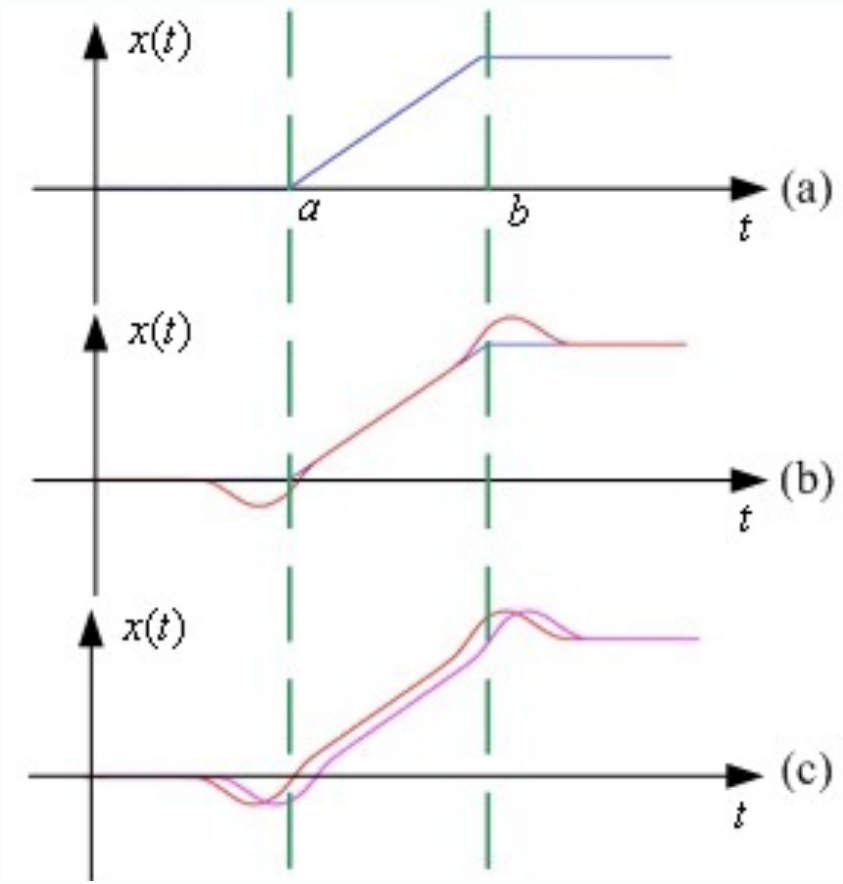


http://en.wikipedia.org/wiki/File:Squash_and_Stretch.jpg

Automatic anticipation

- Subtract a small amount of second derivative from motion
  - works well for lots of cases
  - Wang ea 2006

http://vis.berkeley.edu/papers/animfilter/



Secondary motion

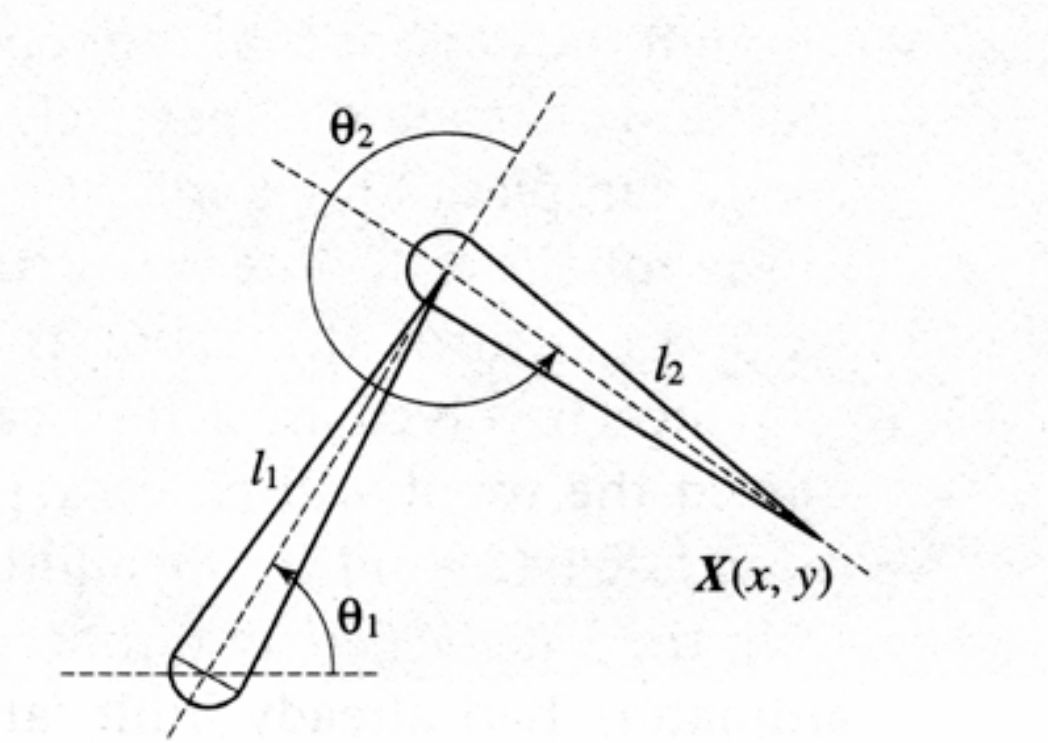
https://www.schoolofmotion.com/blog/secondary-animation

Procedural ideas

- Easily guessed algorithm gives good results
  - waves
  - terrain
  - l-systems (for plants)
  - finite state machines (for character control)

Procedural animation

- Kinematics
  - the configuration of a chain given its state variables
  - e.g. where is the end of the arm if angles are given?
- Inverse kinematics
  - the state variables that yield the configuration
  - e.g. what angles put the end of the arm here?

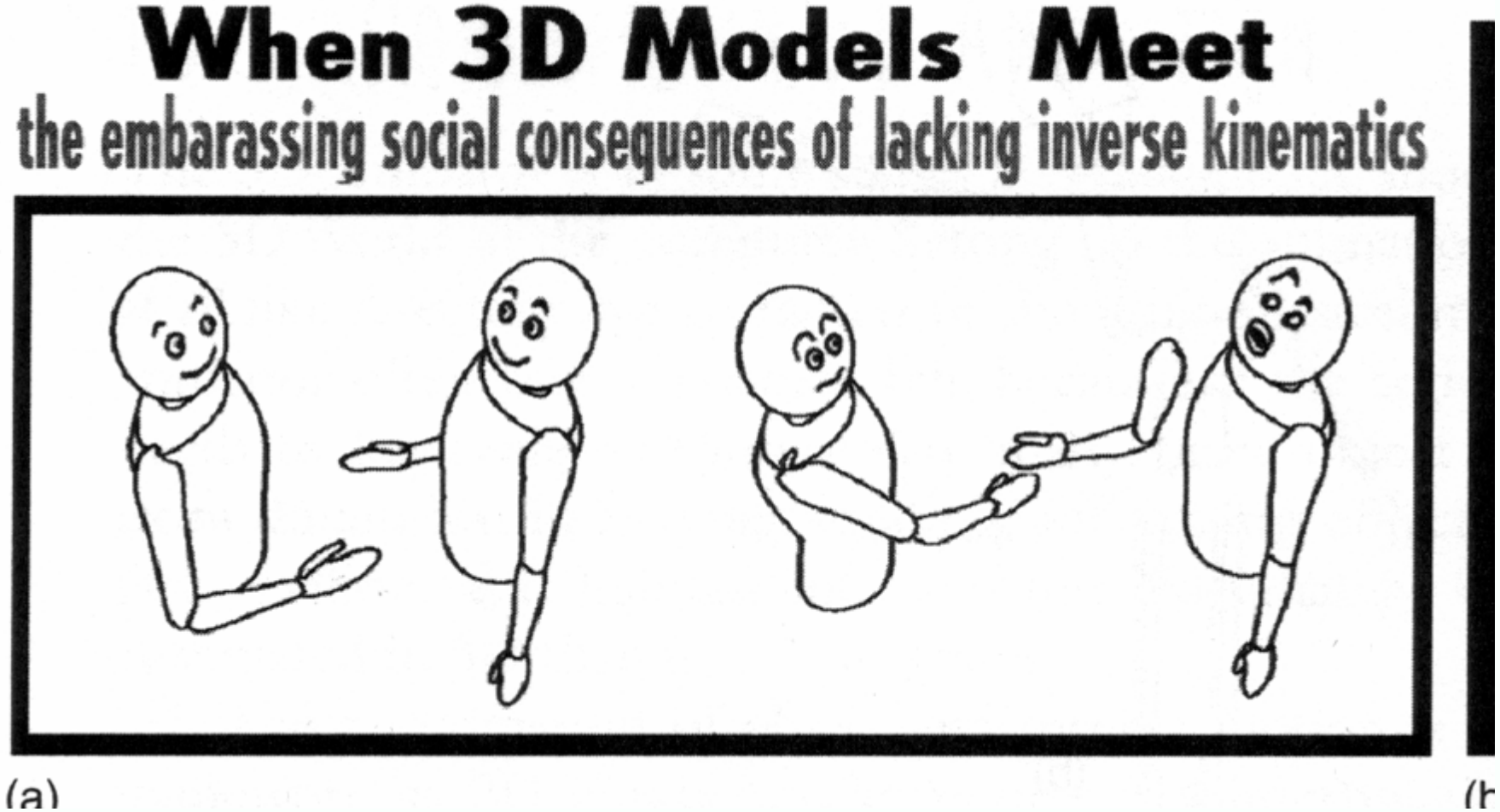


From “The computer Image”,
Watt and Policarpo, 1998

Inverse Kinematics

From “The computer Image”,
Watt and Policarpo, 1998

Inverse Kinematics



From “The computer in the visual arts”, Spalter, 1999

Inverse kinematics

- Endpoint position and orientation is: $\underline{e}(\underline{\theta})$

- Central Question: how do I modify the configuration variables to move the endpoint in a particular direction?

$$\delta \underline{e} = \begin{pmatrix} \frac{\partial e_1}{\partial \theta_1} & .. & \frac{\partial e_1}{\partial \theta_k} \\ .. & .. & .. \\ \frac{\partial e_6}{\partial \theta_1} & .. & \frac{\partial e_6}{\partial \theta_k} \end{pmatrix} \delta \underline{\theta} = J \delta \underline{\theta}$$

Inverse kinematics

- J is the Jacobian
  - If rank(J) < 6, then
    - some movements aren't possible
    - or more than one movement results in the same effect
  - If k>6 then the chain is redundant
    - more than one set of variables will lead to the same configuration

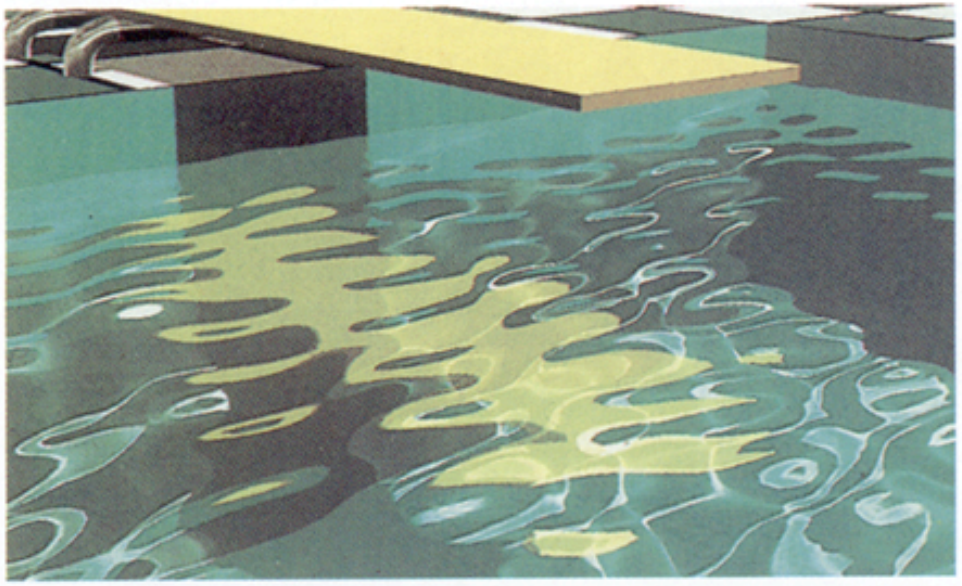
$$\delta \underline{e} = \begin{pmatrix} \frac{\partial e_1}{\partial \theta_1} & .. & \frac{\partial e_1}{\partial \theta_k} \\ .. & .. & .. \\ \frac{\partial e_6}{\partial \theta_1} & .. & \frac{\partial e_6}{\partial \theta_k} \end{pmatrix} \delta \underline{\theta} = J \delta \underline{\theta}$$

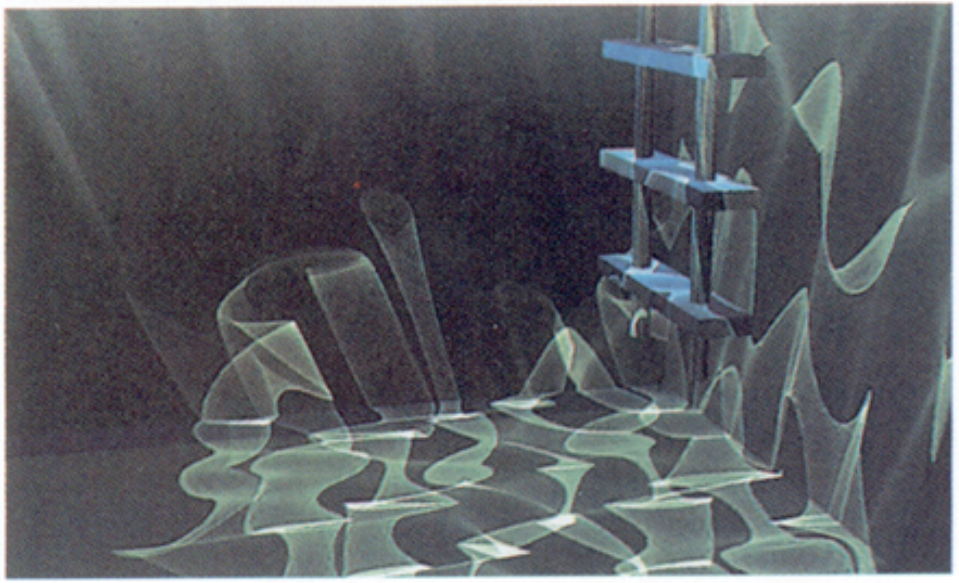
Procedural animation

- Generate animations using procedural approach
  - e.g. “Slice and dice” existing animations to produce a more complex animation
  - e.g. use forward kinematics and a hierarchical model (doors swinging in our original hierarchical model)
  - e.g. construct a set of forces, etc. and allow objects to move under their effects.
    - particle models
    - waves
    - collision and ballistic models
    - spring mass models
    - control - flocking, etc.

Procedural waves

- Sum weighted sinusoids
  - weights change by frequency
  - weights go down as frequency goes up



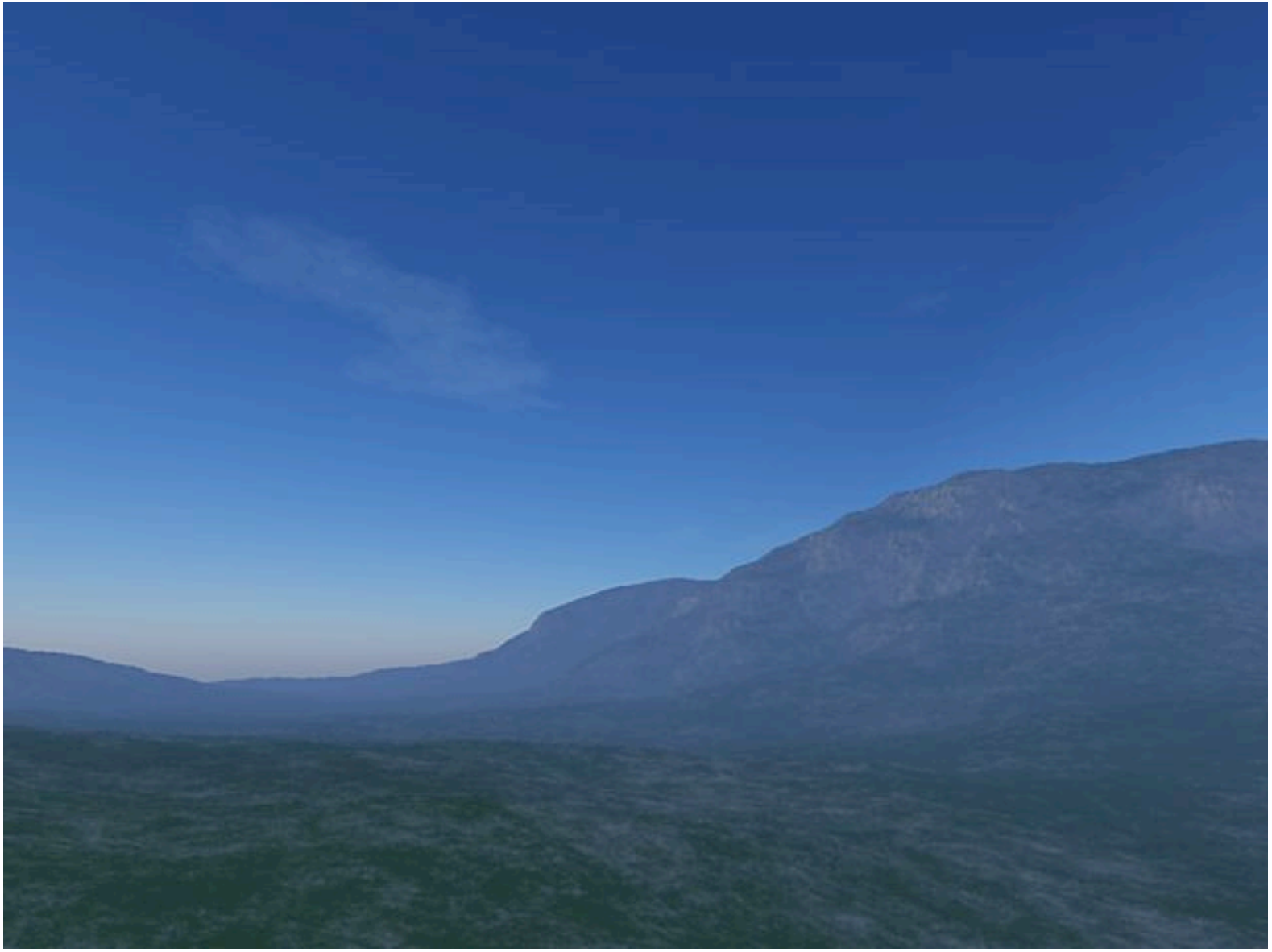


Turbulence/Perlin noise

- Many natural textures look like noise or “smoothed” noise
  - (marble, flames, clouds, terrain, etc.)
- Issue:
  - obtain the right kind of smoothing
- Strategy:
  - construct noise functions at a variety of scales
    - do this by drawing samples from a random number generator at different spacings
  - form a weighted sum

Turbulence/Perlin noise

- Typically,
  - spacing is in octaves
    - number of samples at i'th level is 2^i
  - weights
    - w(i)=p^i
    - p is persistence
- 1D turbulence yields natural head motions
- 2D turbulence yields marble, natural textures, terrains
- 3D turbulence yields animations for clouds, fog, flames



Terrain, clouds generated using procedural textures and Perlin noise
http://www.planetside.co.uk/ -- tool is called Terragen

Terrain, clouds generated using procedural textures and Perlin noise
http://www.planetside.co.uk/ -- tool is called Terragen

Terrain, clouds generated using procedural textures and Perlin noise
http://www.planetside.co.uk/ -- tool is called Terragen

Procedural Animation: L-systems

- Formal grammar, originally due to Lindenmayer
  - {Variables, Constants, Initial state, Rules}
  - Plants by:
    - Constants are bits of geometry,
    - rules appropriately chosen

Figure from wikipedia entry

SIGGRAPH 2016

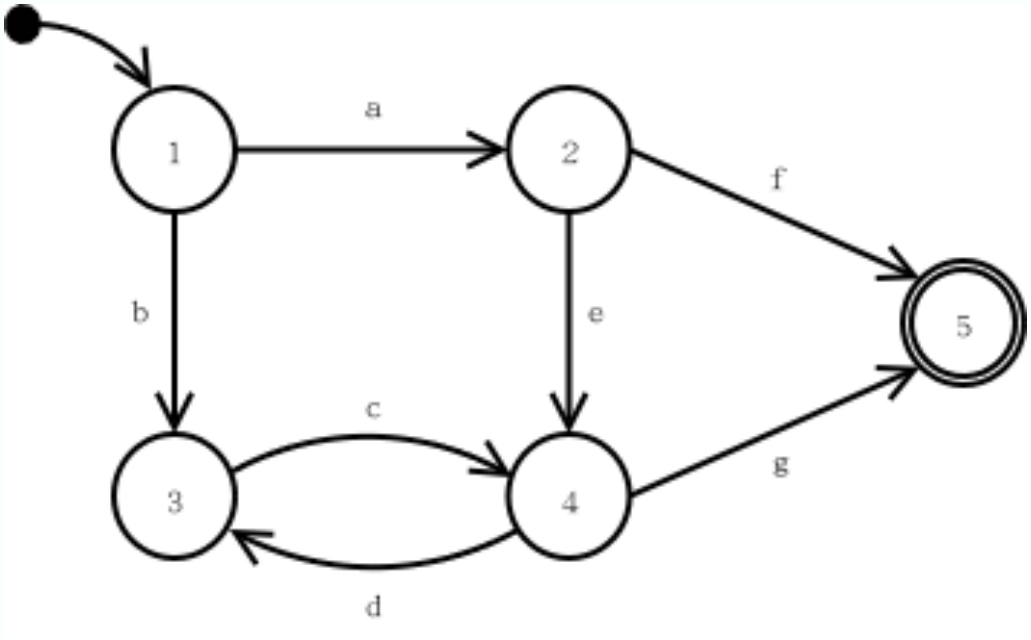
Modeling Dense Inflorescences

Andrew Owens[1]
Mikolaj Cieslak[1]
Jeremy Hart[1]
Regine Classen-Bockhoff[2]
Przemyslaw Prusinkiewicz[1]

[1]University of Calgary
[2]Johannes Gutenberg-Universität Mainz

Finite state machines

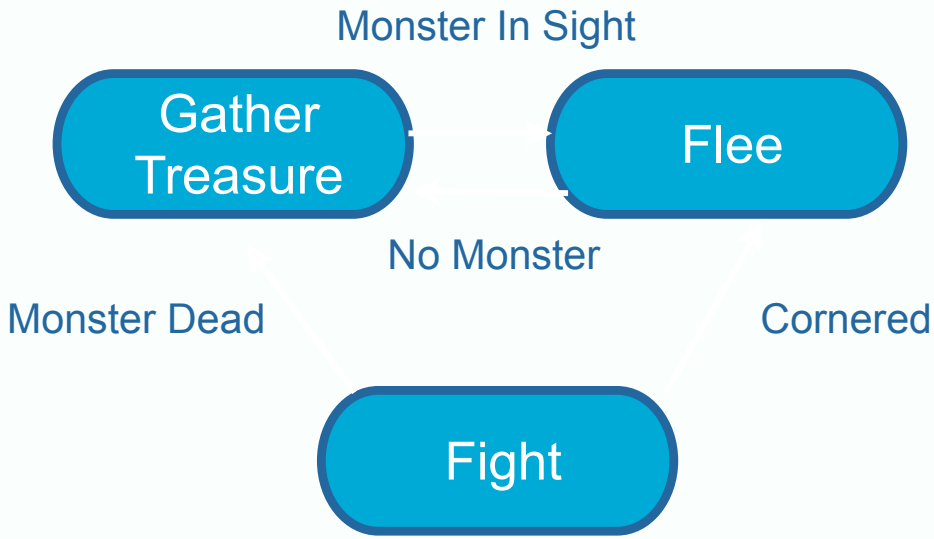
- FSM
  - set of states
    - special start, end state
  - input vocabulary
  - transition function
    - state change on receiving



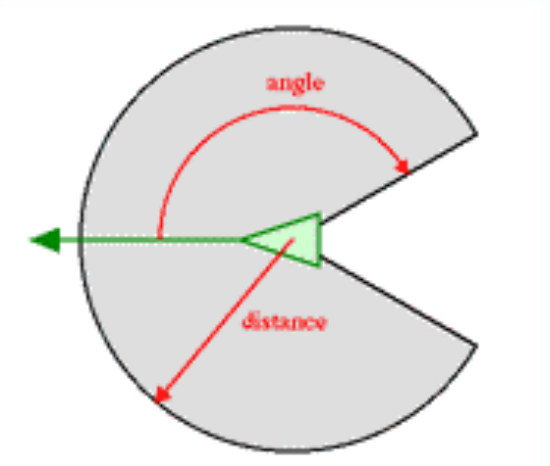
Slides after slides by Jarret Raim, LeHigh

Map to character

- AI modelled as a set of mental states
- State=desired behaviour mode
- Events trigger transition
- Input to the FSM continues as long as the game continues.

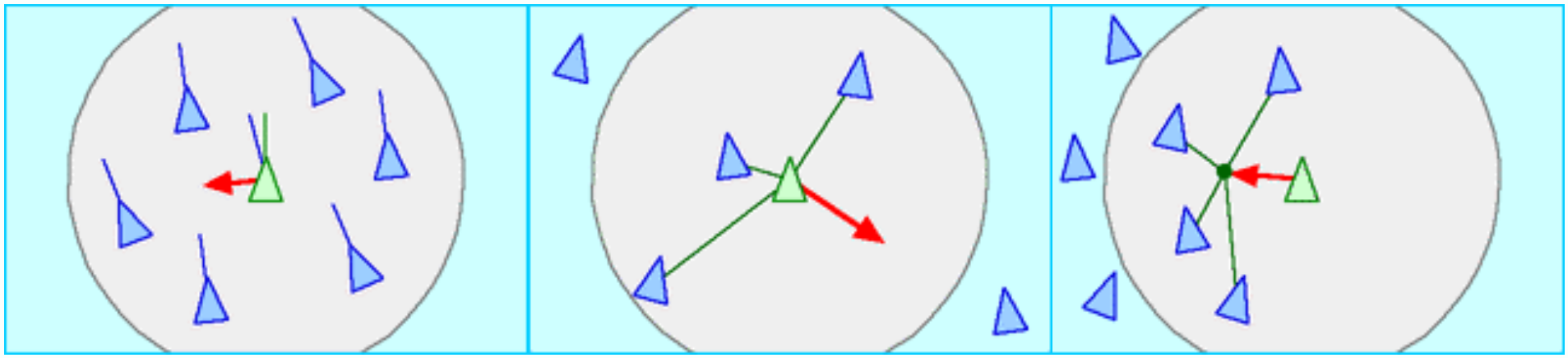


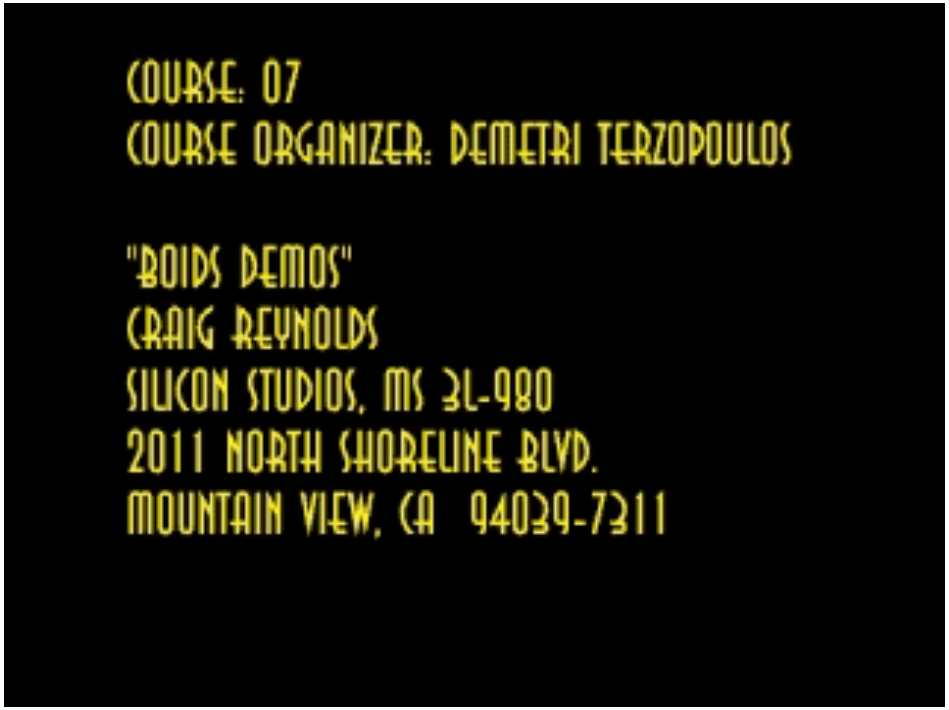
Flocking - Boids



- We'd like things to move in schools
  - and not hit each other, objects
  - abstraction: particle with rocket with maximum force
- 3 goals
  - How to accelerate?
    - each goal gives an acceleration; weighted sum
    - accumulate in priority order until acceleration exceeds threshold,
      - then cut back last

Alignment Separation Cohesion

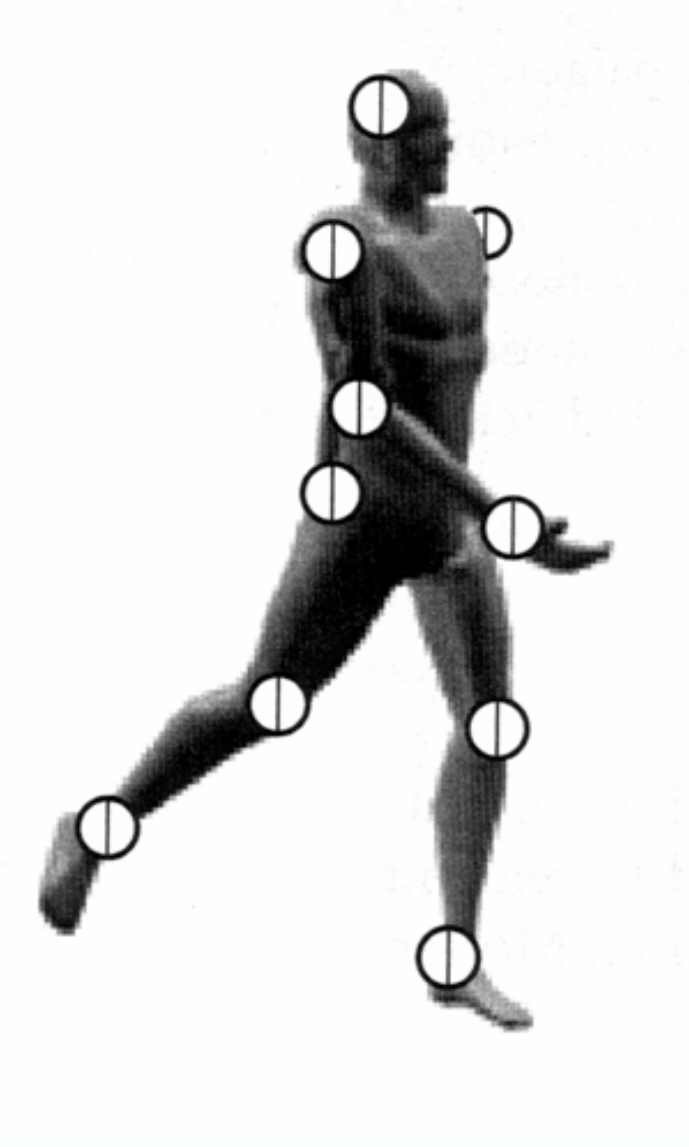




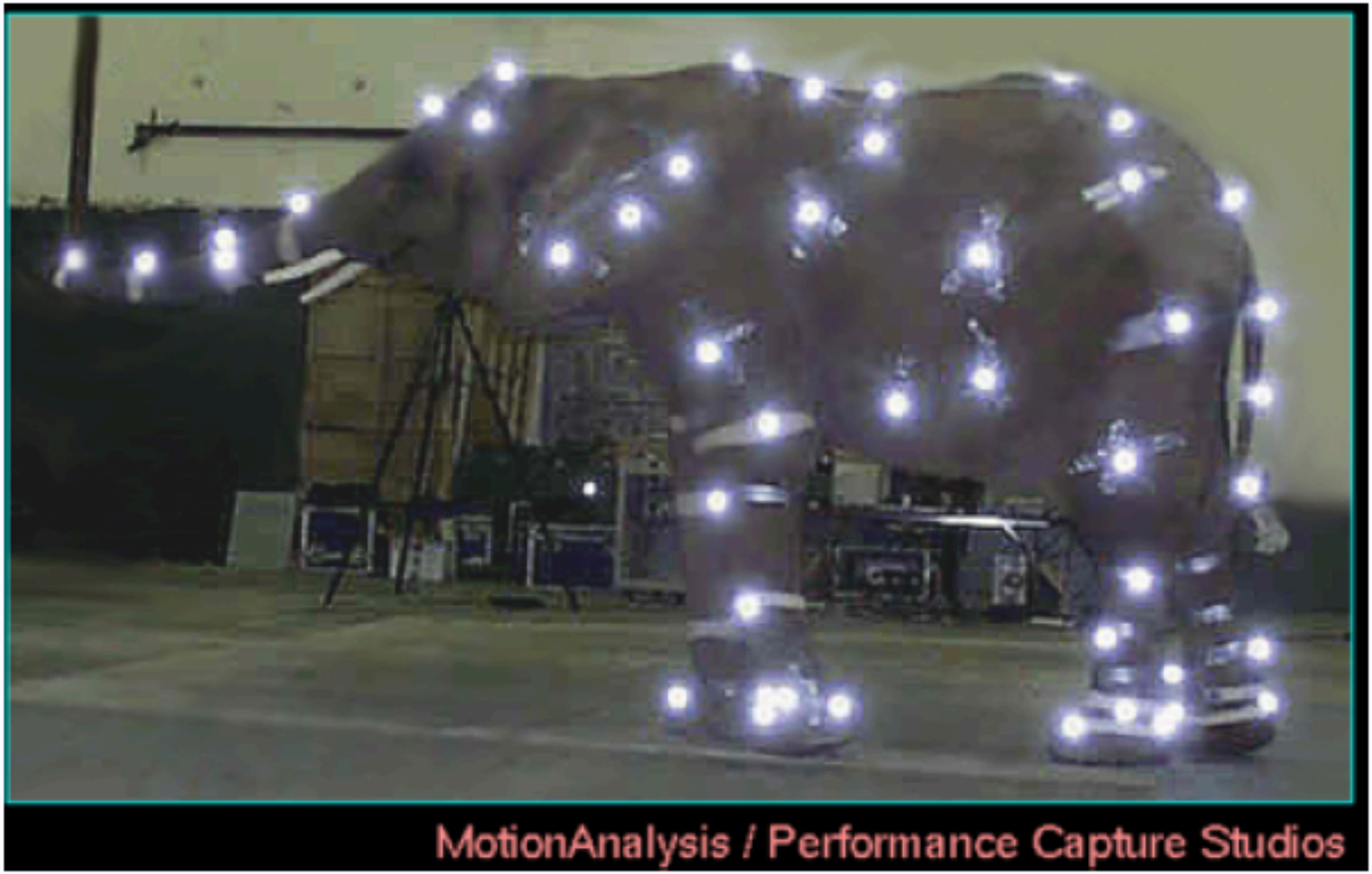
http://www.red.com/cwr/boids.html

http://www.red.com/cwr/boids.html

Motion capture



- Instrument a person or something else, perhaps by attaching sensors
- Measure their motion
- Link variables that give their configuration to variables that give configuration of a computer model


MotionAnalysis / Performance Capture Studios

Key motion capture question

- How to generalize?
  - use motion capture (or other) data of moving person to generate new motions
  - that are:
    - goal directed
    - genuinely new (?)
    - high quality
- Obstacles (at least)
  - complex covariances across the body
  - contact and collision
    - footskate
  - pronounced human sensitivity to motion problems

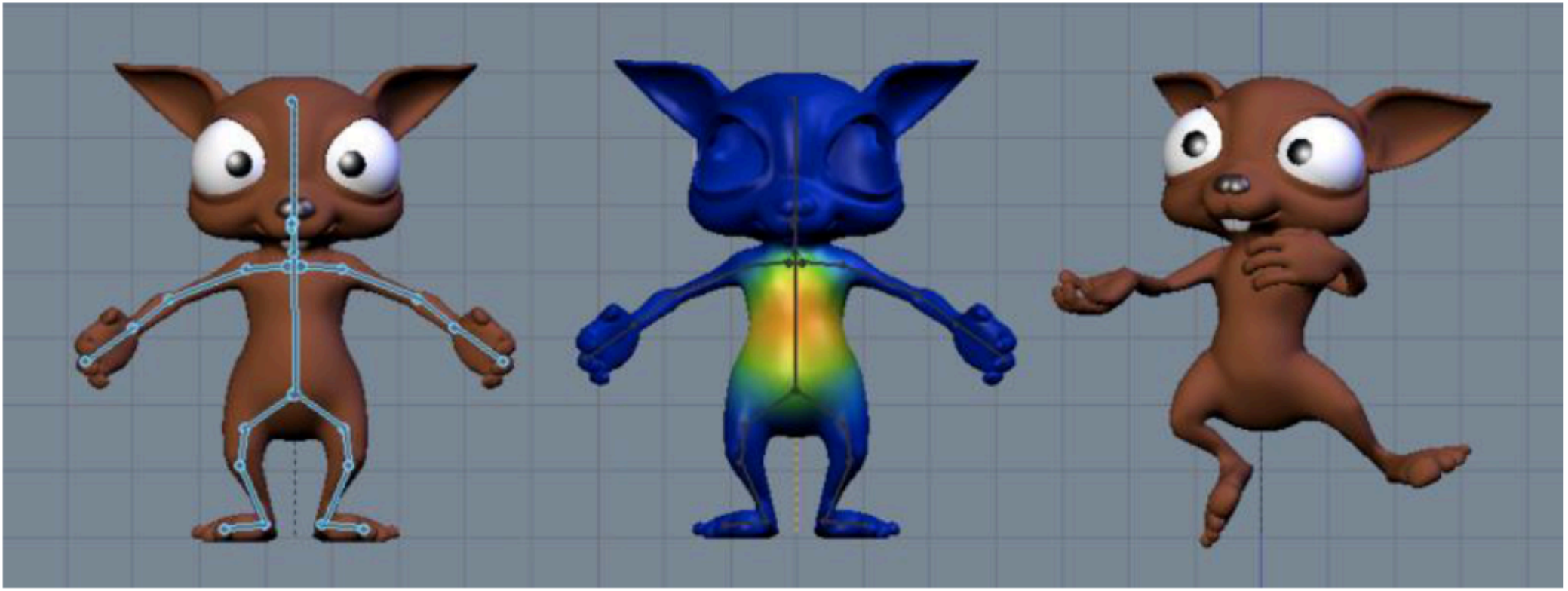
Some pragmatics

- Two major representations
  - keypoint position
  - joint angles
- Each implies a skeleton and a root coordinate system
  - keypoint position directly - where are the keypoints in rest position?
  - joint angles indirectly
- Transferring from skeleton to skeleton is not easy
  - for example, contacts can fail quite badly
    - interaction between root and limb lengths
  - for example, longer heavier limbs change timing
    - model the leg as a pendulum
  - often (usually) done by hand

More pragmatics

- Mostly, we don't want to see moving points
  - must link motion capture data to control of a mesh (etc.)
- Generically known as "rigging"
  - mostly done by hand
- Skinning
  - link joint positions to mesh vertex positions
  - part of rigging

Some skinning



Slides from Stanford CS248 2017, I *think* Fedkiw

Well understood case: Faces

- Measure neutral (rest) position
- Then measure various key poses
  - either by deforming face and measuring
  - or modeller deforms meshes
- Use the same mesh topology for each case, so we have

$$\begin{bmatrix} x_{1i} \\ x_{2i} \\ \vdots \\ x_{mi} \end{bmatrix}$$

i'th shape, m vertices

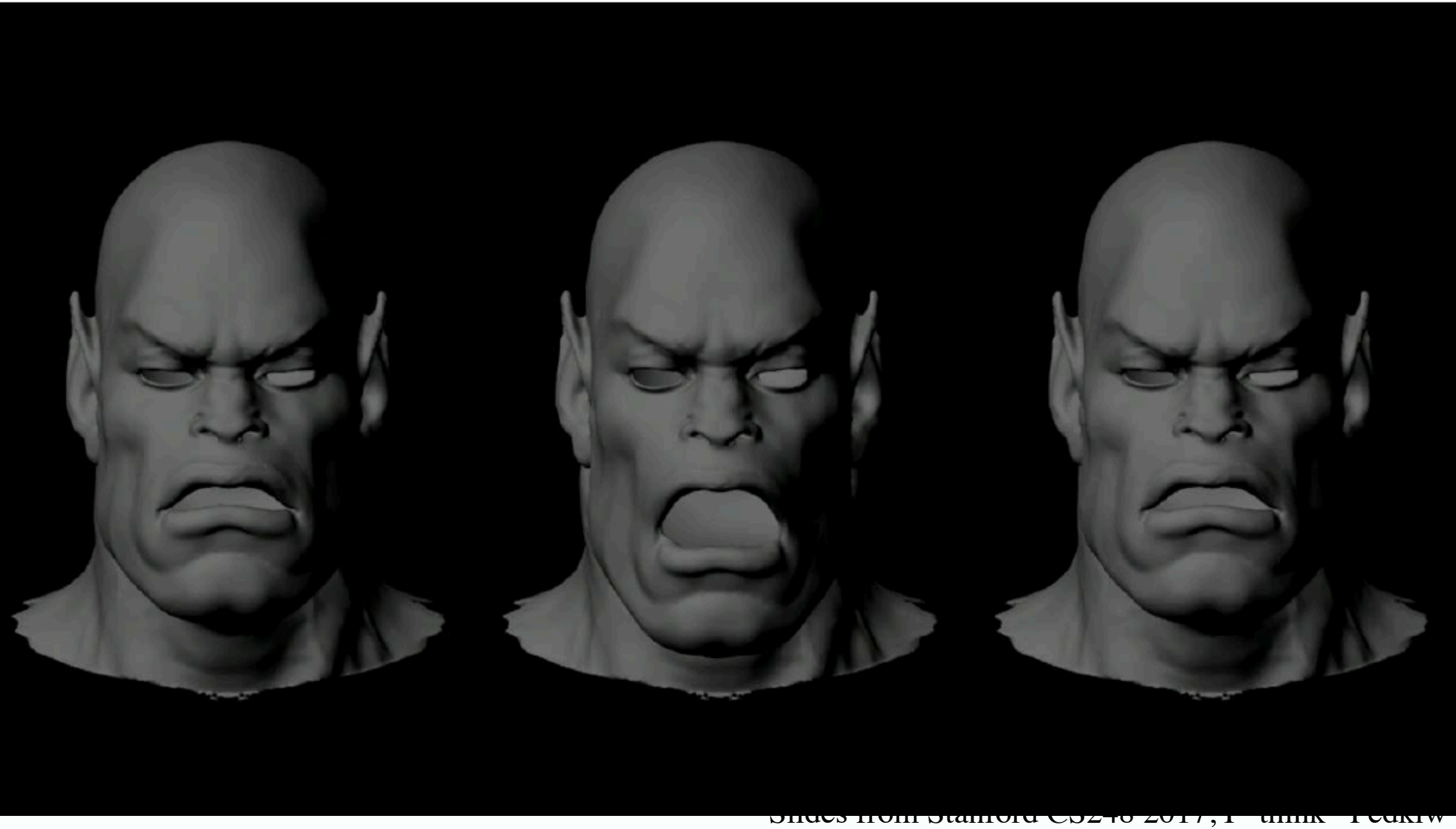
- Obtain a new shape by linearly interpolating between two key shapes

$$.28\begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{m1} \end{bmatrix} + .72\begin{bmatrix} x_{12} \\ x_{22} \\ \vdots \\ x_{m2} \end{bmatrix} = \text{resulting shape}$$

Slides from Stanford CS248 2017, I *think* Fedkiw

Animation

- Vary the interpolation weights (α,1-α) over time



Shape Matrix

- Consider the case of n key shapes (with m vertices in each)
- Concatenate the n column vectors to form a shape matrix:

$$\begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{m1} \end{bmatrix}, \begin{bmatrix} x_{12} \\ x_{22} \\ \vdots \\ x_{m2} \end{bmatrix} \cdots \begin{bmatrix} x_{1n} \\ x_{2n} \\ \vdots \\ x_{mn} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

- Note that one of the key shapes needs to be the face in a neutral/rest pose

Slides from Stanford CS248 2017, I *think* Fedkiw

Interpolation

- A new shape is computed by multiplying the shape matrix with a vector of interpolation weights:

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

- Every vector of interpolation weights $\vec{\alpha}$ gives a new set of vertex positions (i.e., a new shape) $\vec{x}$
- Animate the vector of interpolation weights $\vec{\alpha}$ in order to animate the shape of the face

Slides from Stanford CS248 2017, I *think* Fedkiw

Displacements

- Alternatively, one could construct a displacement matrix consisting of displacements from the neutral/rest pose

$$\begin{bmatrix} \delta x_{11} & \delta x_{12} & \cdots & \delta x_{1n-1} \\ \delta x_{21} & \delta x_{22} & \cdots & \delta x_{2n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \delta x_{m1} & \delta x_{m2} & \cdots & \delta x_{mn-1} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{n-1} \end{bmatrix} = \begin{bmatrix} \delta x_1 \\ \delta x_2 \\ \vdots \\ \delta x_m \end{bmatrix}$$

- In this case, the neutral shape $\overrightarrow{x_0}$ is not a column in the matrix (it would be a column of all zeroes)
- The result of the matrix multiplication is added to the neutral shape to obtain the new shape:

$$\vec{x} = \overrightarrow{x_0} + \overrightarrow{\delta x}$$

- The two approaches can be shown to be equivalent, if the weights have the property: $\sum_1^n \alpha_i = 1$

Slides from Stanford CS248 2017, I *think* Fedkiw

A Different Approach

- A similar process could be carried out for the body
  - i.e. create a shape matrix and interpolate
- But, the shape of the body is highly dependent on the angles of joints, so one can bootstrap the interpolation weights $\vec{\alpha}$ from the joint angles
  - The joint angles do miss some shape information such as whether a muscle is being intentionally flexed
  - Note: the $\vec{\alpha}$ in facial animation can be bootstrapped in a similar fashion using the angle of the jaw joint and contractions of various facial muscles
- Many parts of the body are relatively disjoint from each other, so we expect the displacement matrix to be sparse (but the shape matrix is not sparse)
- Because of these considerations, we approach skinning the body in a slightly different manner
  - While noting that it still highly depends on shapes and interpolation

Slides from Stanford CS248 2017, I *think* Fedkiw

Skinning joint angles

- Decompose the entire skin (for the whole character) into smaller pieces, and place a portion of the skin into the object space of each bone
  - The pieces may overlap, i.e. multiple bones may share the same skin vertices
- Given a set of joint parameters θ
- Let $T_i(\theta)$ represent the transformation that moves bone i from its object space to world space
- As the joint parameters change and the bones move in world space, calculate where the skin vertices are located in world space as well using $T_i(\theta)$
- Skin vertices which exist in the object space of multiple bones require some sort of interpolation or averaging

Slides from Stanford CS248 2017, I *think* Fedkiw

Rigid Skinning

- Each skin vertex is assigned to exactly one bone
  - For example, the skin for the upper arm would be assigned to a different bone than the skin for the forearm
- Use the transform of the associated bone to position each vertex of the skin in world space:
  - Consider a vertex j with position v_j in the object space of the ith bone with transformation T_i
  - Then, the world space position of vertex j is given by

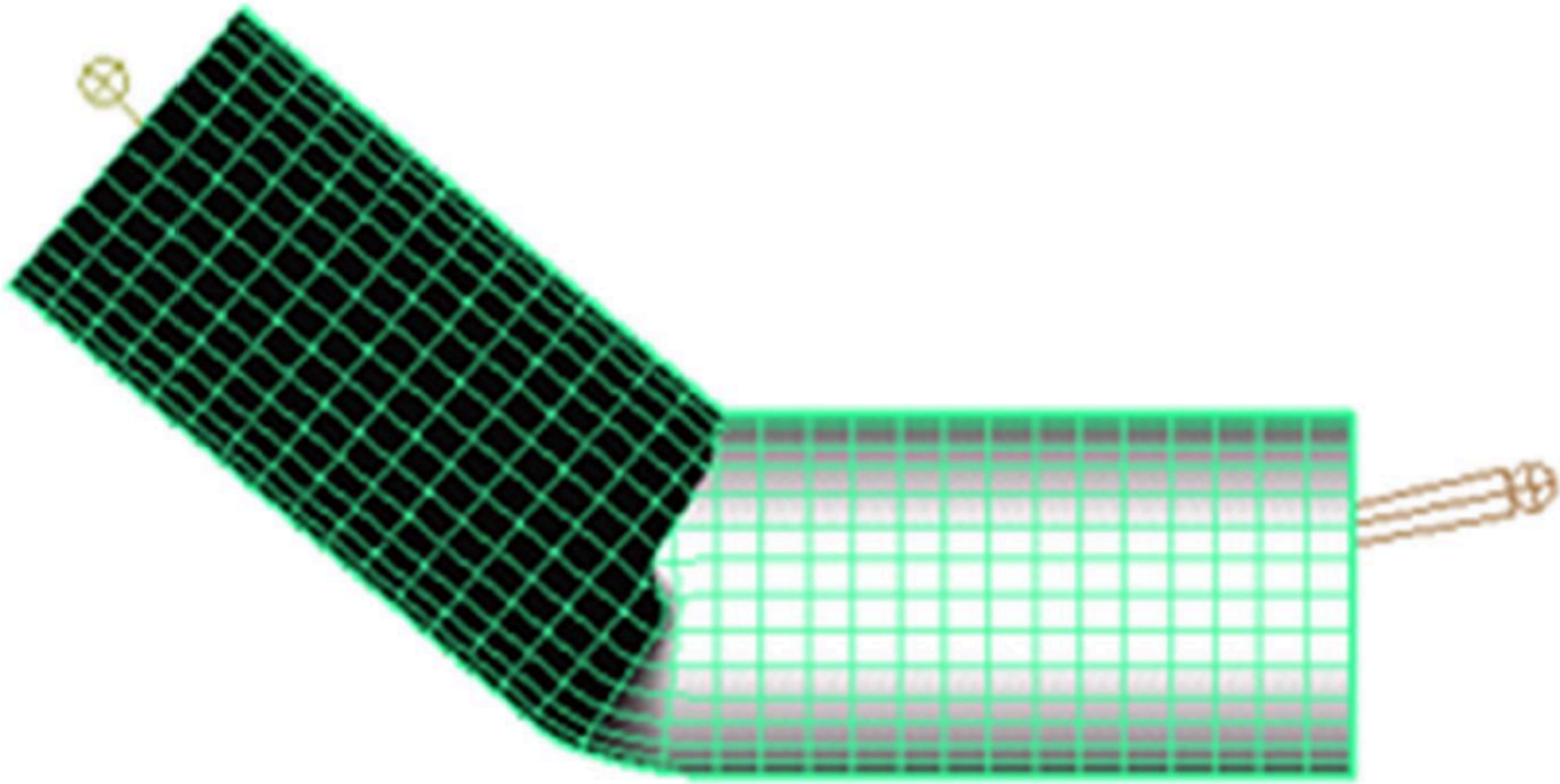
$$v_j' = T_i v_j$$

- As the skeleton moves, T_i changes and the vertex positions of the skin change as well

Slides from Stanford CS248 2017, I *think* Fedkiw

Rigid Skinning

- Unwanted discontinuities form along the boundaries where neighboring skin vertices are assigned to different bones



Slides from Stanford CS248 2017, I *think* Fedkiw

Linear Blend Skinning

- Remove the discontinuity by linearly blending vertices near the joint
- Assign each skin vertex to more than one bone
  - Note: v_j^i will have different coordinates in different rigid body object spaces
- Each bone i to which vertex v_j belongs to is assigned a nonzero weight w_{ij}
- The world space position of the vertex is computed as the weighted average of the world space positions obtained from each bone via rigid skinning:

$$v_j' = \sum_i w_{ij} T_i v_j^i$$

Slides from Stanford CS248 2017, I *think* Fedkiw

Normals & Tangents

- Normal and tangent vectors of the surface mesh (important for rendering/collisions) are blended as well:

$$n_j' = \sum_i w_{ij} T_i^{-T} n_j^i$$

$$t_j' = \sum_i w_{ij} T_i t_j^i$$

- Normalize n_j' and t_j' if unit length is required

Weights

- Weights for a vertex should be sparse
  - E.g., if the angle of the elbow joint is changed, the skin for the leg shouldn't deform
  - Nonzero weights should be localized to nearby bones
- Sparse weights allow for fast evaluation
  - Typically at most four non-zero weights per vertex (at most four bones can deform a vertex)
- Weights should be smooth to avoid discontinuities
  - Often chosen with a smooth falloff based on distance to a particular bone
- Weights should be independent of mesh resolution
  - So that subdividing the mesh doesn't require recomputing weights
- Constrain the weights to be convex (i.e. $\sum_i w_{ij} = 1, w_{ij} \geq 0$) to avoid undesired scaling and extrapolation artifacts

Slides from Stanford CS248 2017, *think* Fedkiw

Specifying Weights

- Manual Approach:
  - Hand-tune weights in order to obtain the best look
  - Intractable to individually modify the weights for each vertex in a large mesh
  - Various painting tools facilitate weight specification

- Automatic Approach:
  - Use an algorithm to calculate weights for each vertex and all its associated bones
    - E.g., based on a "distance" metric from vertices to bones
  - Automatically generated weights are often additionally modified by an artist for higher visual fidelity

Slides from Stanford CS248 2017, I *think* Fedkiw

Specifying Weights: Pinocchio

- System for automatically rigging and animating 3D characters
- Solves a Poisson equation (PDE!) for each bone with appropriate boundary conditions to obtain smoothly varying weights
- Can be used to rig and skin your own characters

- Available from MIT:

http://www.mit.edu/~ibaran/autorig/pinocchio.html

Slides from Stanford CS248 2017, I *think* Fedkiw

Artifacts…

- Linear blend skinning has issues when the joint angles are large or when a bone undergoes a twisting motion
  - "bow tie" or "candy wrapper" effect
  - mesh loses volume
- Linearly blending the matrix representations of rigid body transformations does not (in general) result in a matrix that represents a rigid body transformation

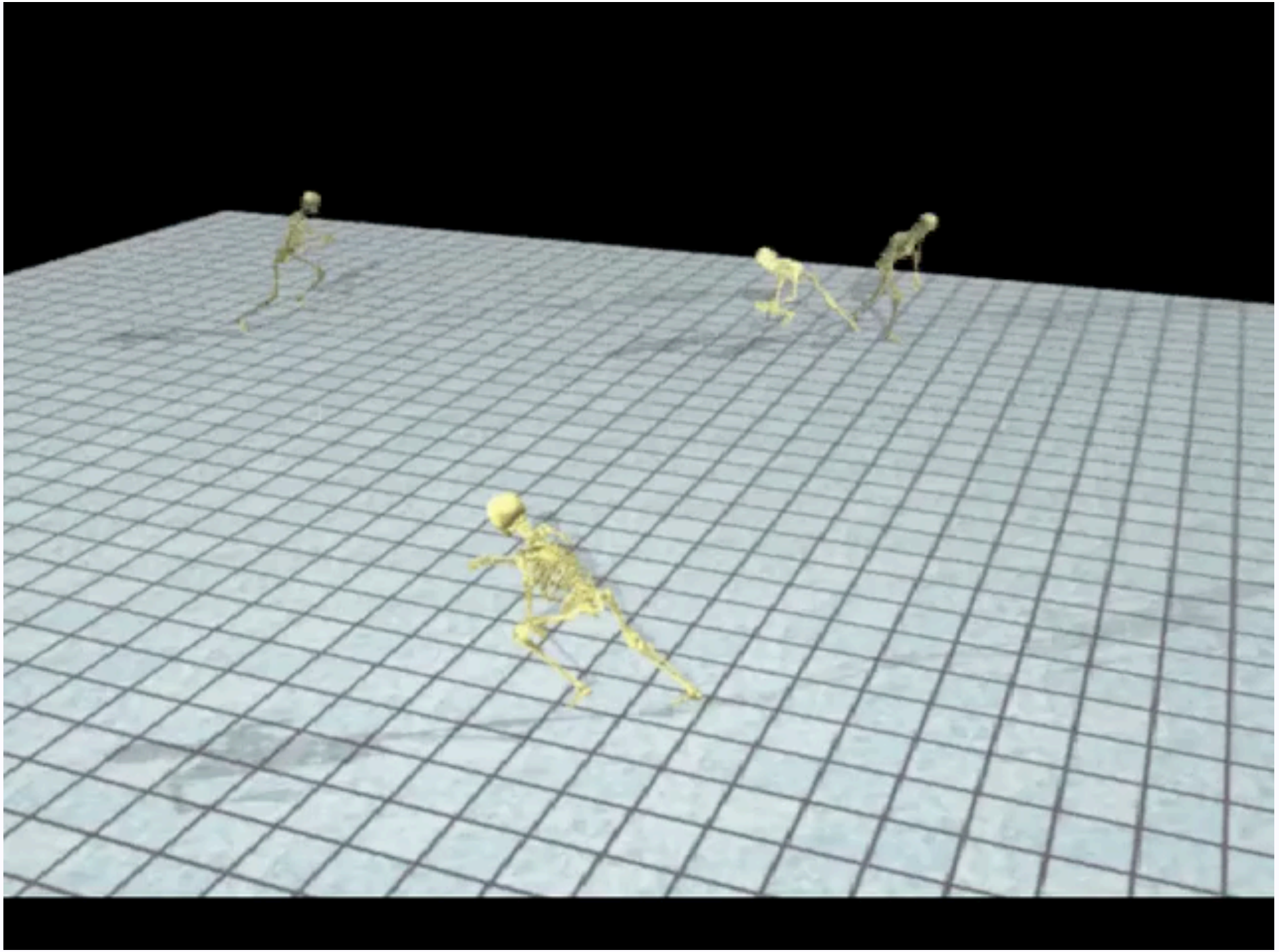
Slides from Stanford CS248 2017, I *think* Fedkiw

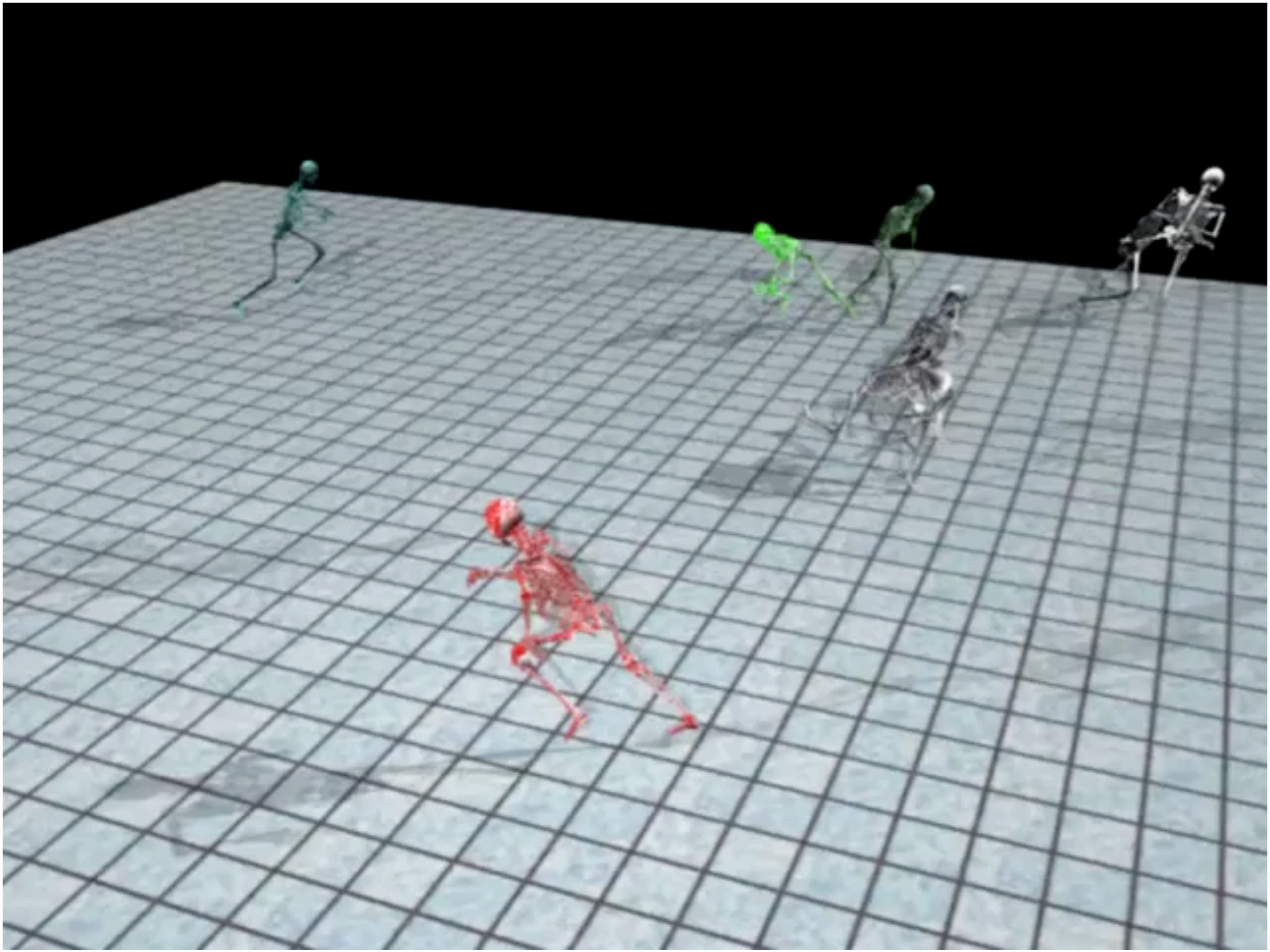
Much more skinning material in notes…

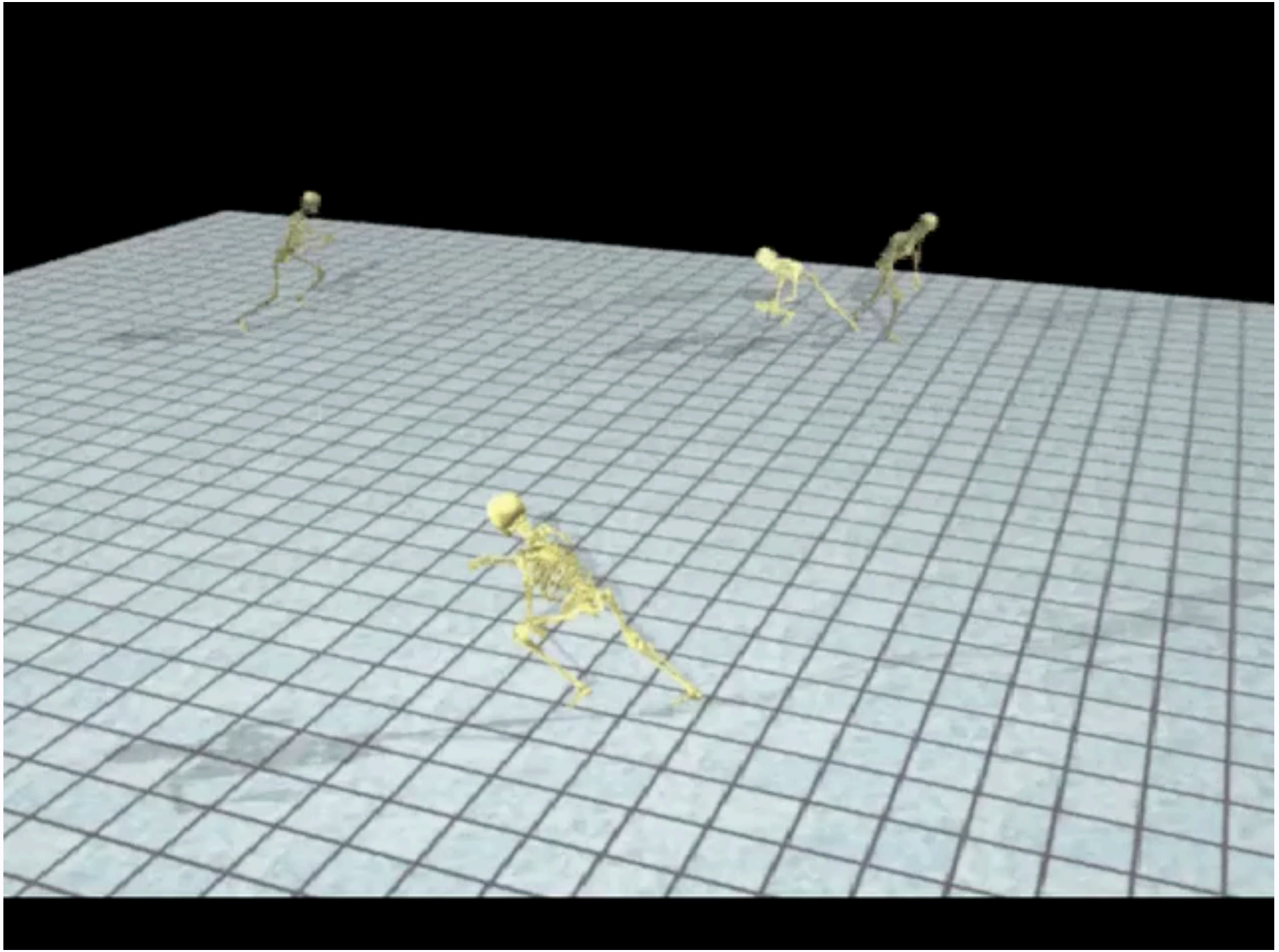
Slides from Stanford CS248 2017, I *think* Fedkiw

Simplest idea: the motion graph

- Three papers (web page) with essentially the same idea
- Each frame of motion capture is a vertex
- Directed edge between observed transitions
- Matching (say) to induce more edges
  - eg if V_a similar to V_i, V_b similar to V_i+1 insert edge V_a->V_b
- Now search this
- Can do:
  - start, end, keyframe constraints
  - interactive control
  - control from video (just!)
  - control by annotation







Various procedures

- Path of fixed length with start, end, keyframe constraints
  - randomized search (details in Arikan et al)
- Interactive control
  - clean up graph so that there are no dead ends
  - use interactive device to choose next frame (Kovar et al)
- Control from video
  - as above (Lee et al)
- Control by annotation
  - incorporate a version of dynamic programming into search
  -

Motion Synthesis from Annotations

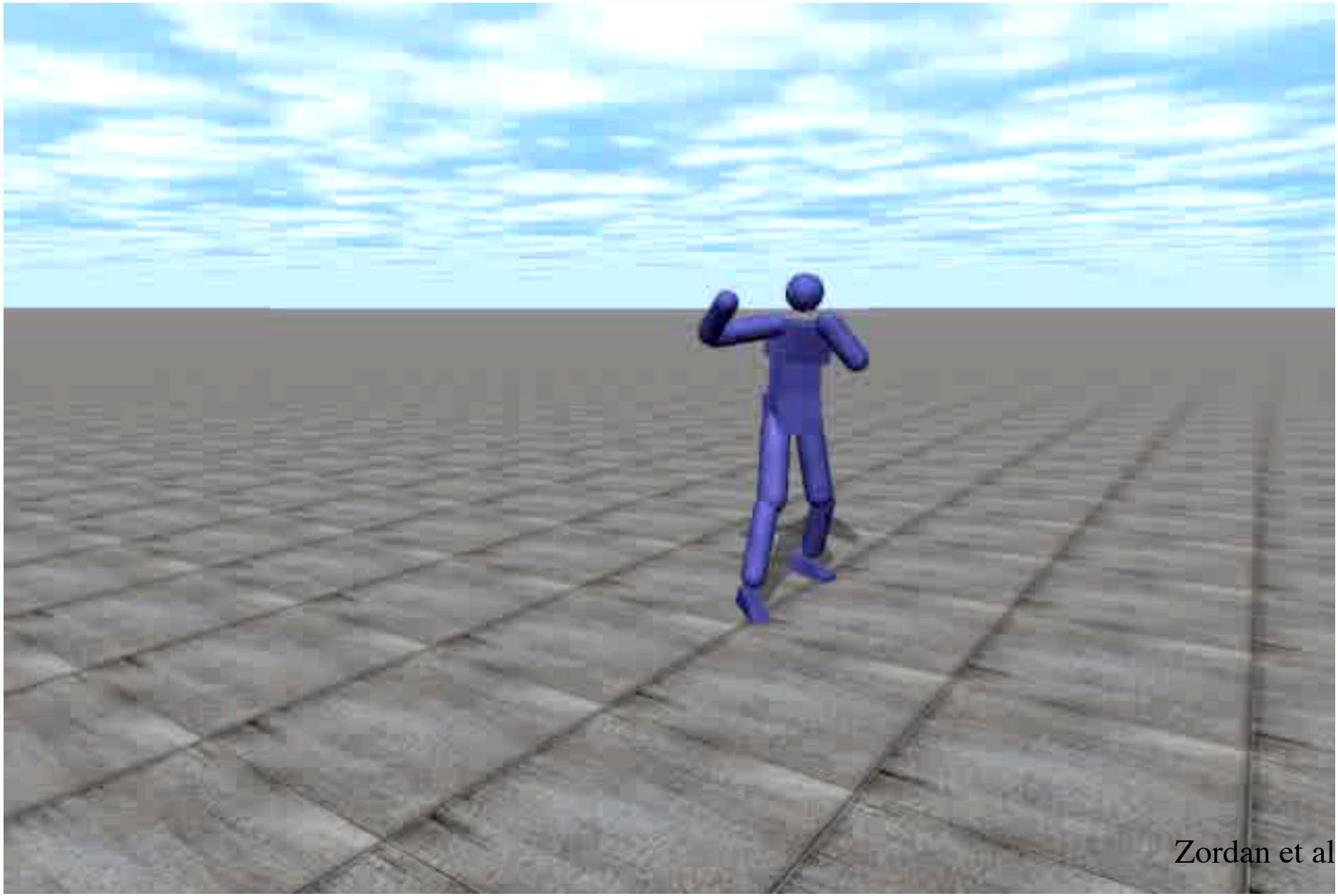
Okan Arikan
David Forsyth
James O'Brien

U.C. Berkeley

Motions can be blended and deformed

Arikan ea 06

However, modifying motion is dangerous




Replay

Motion data is low dimensional

- Kinematic:
  - actual human poses are wildly redundant
  - there are invariants in joint position data
  - there is quite good evidence that “snapping” to a body manifold is helpful
    - eg 3D pose recovery
    - styleIK
- Dynamic:
  - There are strong correlations across the body
    - eg Pullen+Bregler 04

StyleIK

- Build a model of P(kinematic configurations)
  - by augmenting observations with (unknown) latent variables
    - fit gaussian process model to this
- Use model to
  - produce posterior estimates of pose
    - meeting imposed constraints
      - (i.e. IK, but with a strong bias to good poses)
  - blend between sequences
    - i.e. find blend that is (a) roughly original and (b) biased to good
- Good
  - very effective, uses relatively little data
- Bad
  - one model per motion style

Q: Could we apply modern machinery?

- Likely A:
  - Yes
    - eg Victor's stuff
    - eg papers to follow

Motion data is low dimensional

- Dynamic:
  - There are strong correlations across the body
    - eg Pullen+Bregler 04
    - eg Safonova et al 04
    - important failures in transplantation (Ikemoto et al 04)
  - Physical constraints are important
    - obvious fact! you can't break the laws of physics
    - particularly for extreme motions, ballistic motions
  - but not dispositive
    - remains very hard to (say) synthesize locomotion
      - out of purely physical considerations

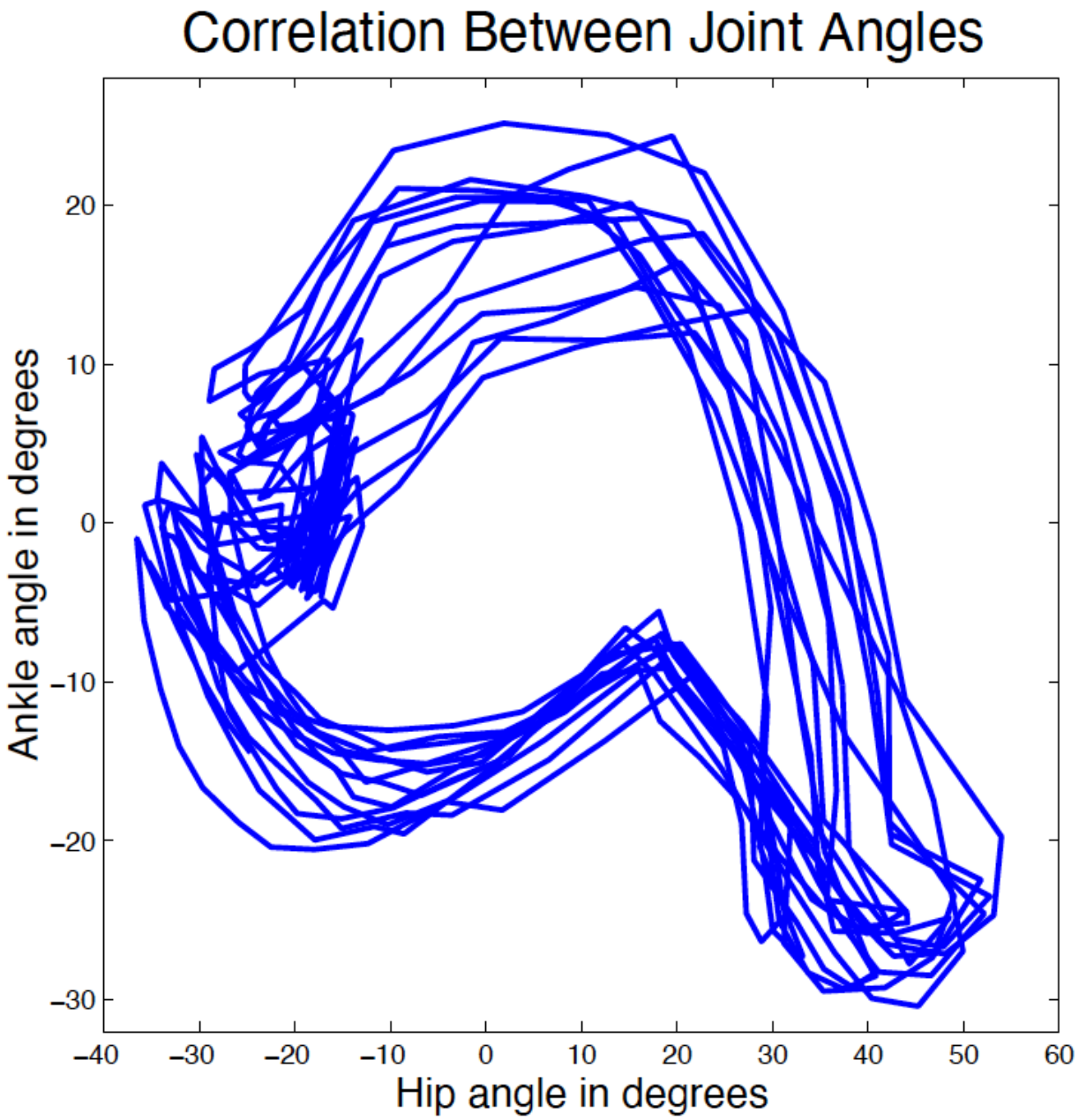


Figure 2: Correlation between joint angles. Shown is the ankle angle versus the hip angle for human walking data. The fact that this plot has a definite form demonstrates that the angles are related to each other.

Pullen+Bregler 04

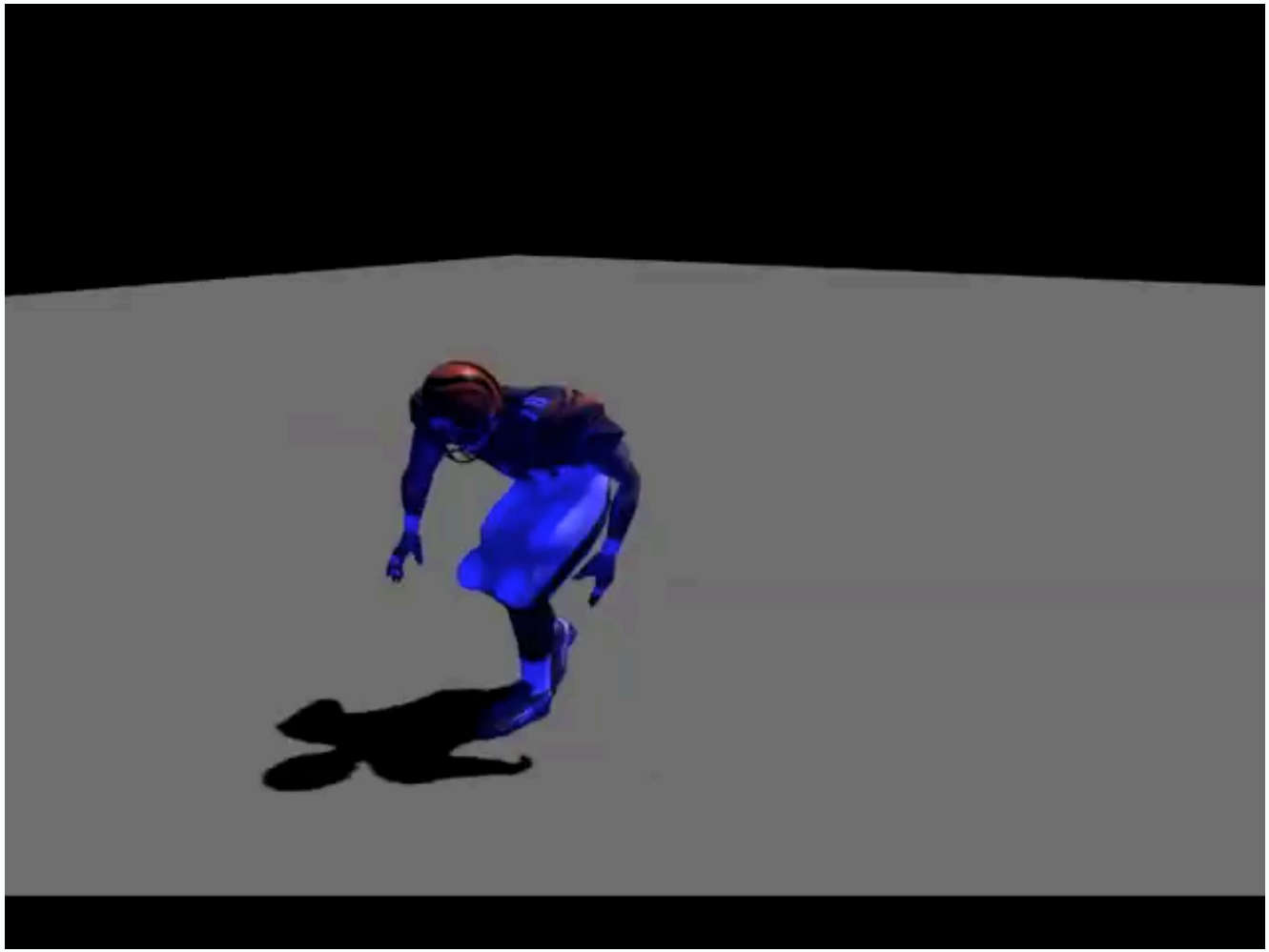
Motion capture data is “low-dimensional”

Safonova et al 04

Transplantation

- Motions clearly have a compositional character
  - Why not cut limbs off some motions and attach to others?
    - we get some bad motions
    - caused by cross-body correlations
  - build a classifier to tell good from bad
    - avoid foot slide by leaving lower body alone

Ikemoto+Forsyth 04

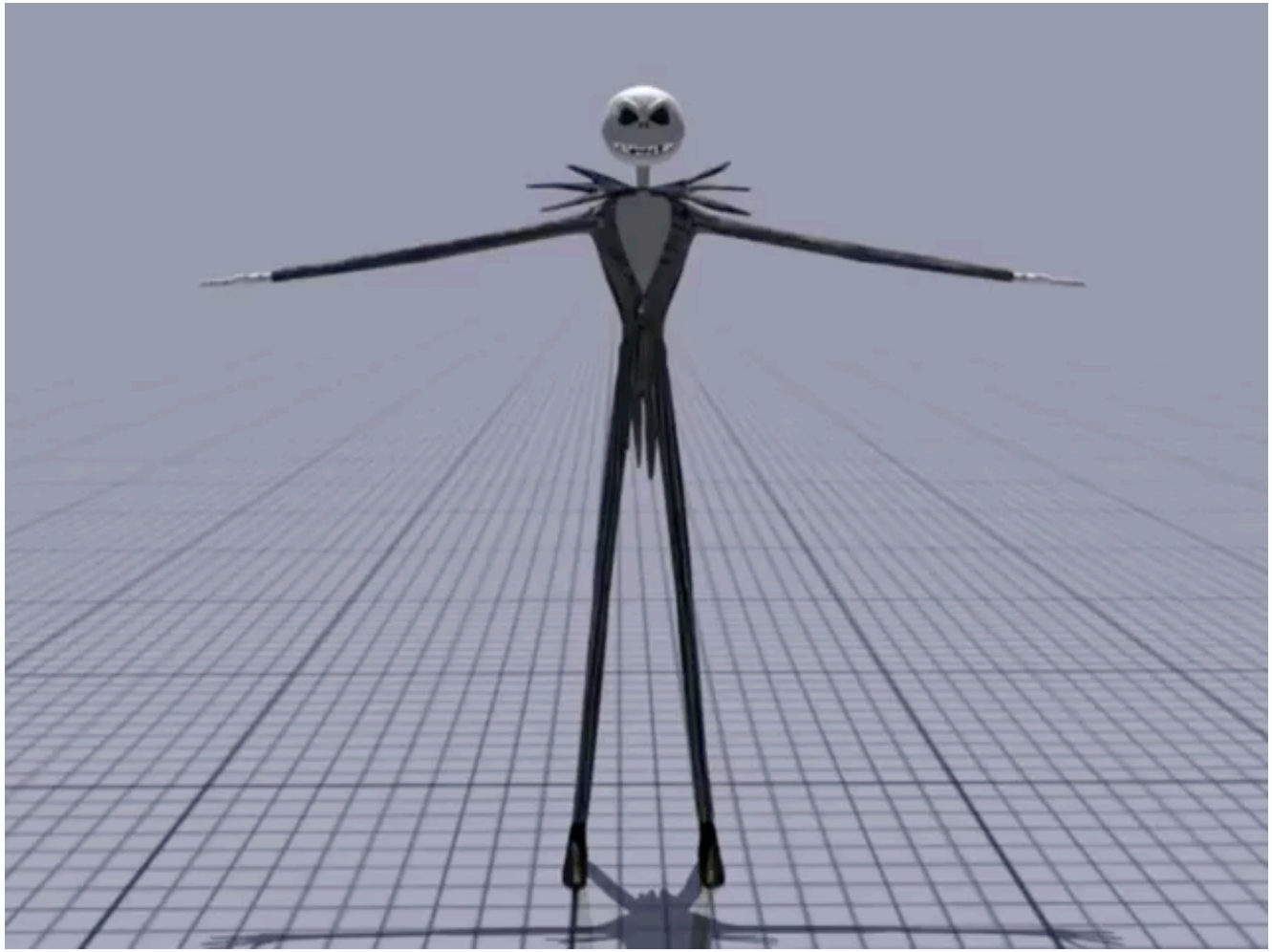


Ikemoto+Forsyth 04

Style

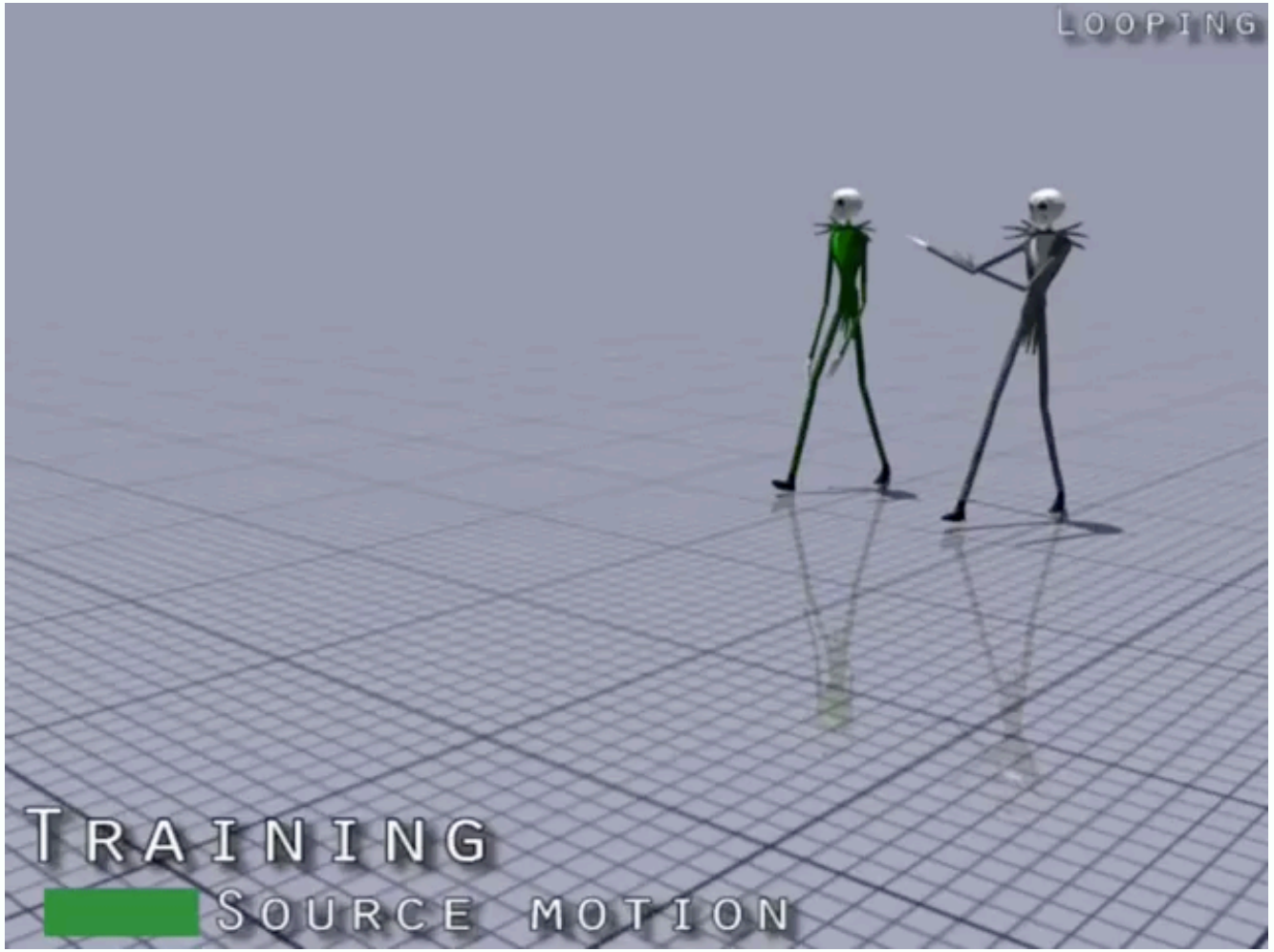
- Qualitative properties of motion, including
  - individual characteristics
  - modifiers, eg: clumsy, fast, heavy, forceful, graceful
- Animation problem:
  - Control new character with old motion, preserving new character's style
- Vision problem:
  - infer style descriptors, identity from observed motion

Kinematic style transfer



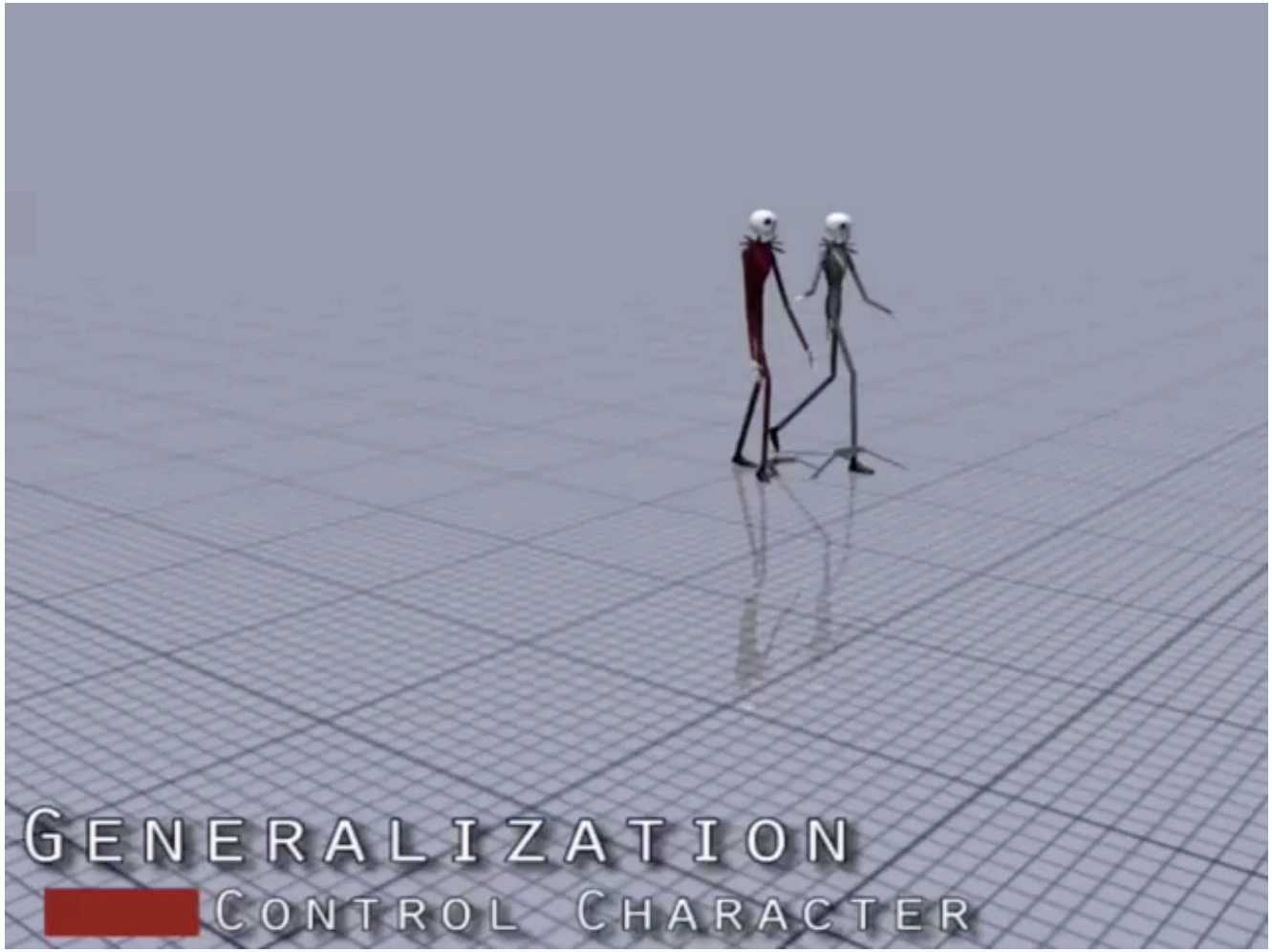
Ikemoto ea 09

Kinematic style transfer



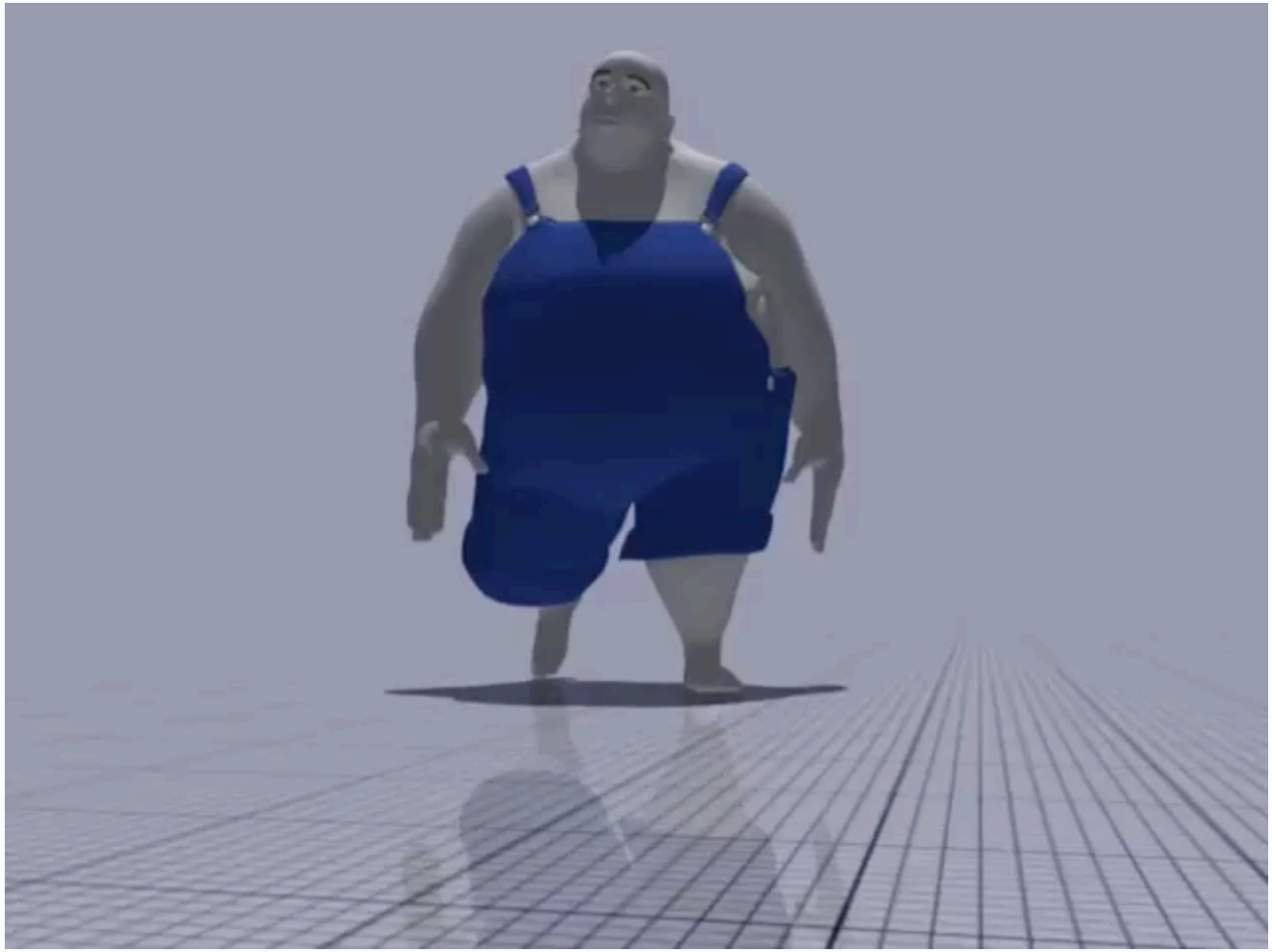
Ikemoto ea 09

Kinematic style transfer



Ikemoto ea 09

Kinematic style transfer



Ikemoto ea 09

The Question: Generalization

- It's hard because:
  - People seem very aware of detail in other peoples motion
    - footplants, contacts, etc.
- Temporal composition rules!
  - because nothing else looks natural
  - very hard to escape at present
  - consequence: major shortage of motion capture data

The question, equivalent

- How do we come up with an embedding/LD model
  - that gets details right
    - doesn't "smear" contacts, dynamics
    - respects kinematic constraints
    - respects contact
    - has temporal correlation right
  - that generalizes aggressively
    - many motions are (should be?) members of parametric families
      - reaching, striking, kicking, etc.
  - and doesn't require unreasonable quantities of data

Motion VAE

- Synthesize x_i+1 conditioned on x_i
  - which yields an autoregressive model

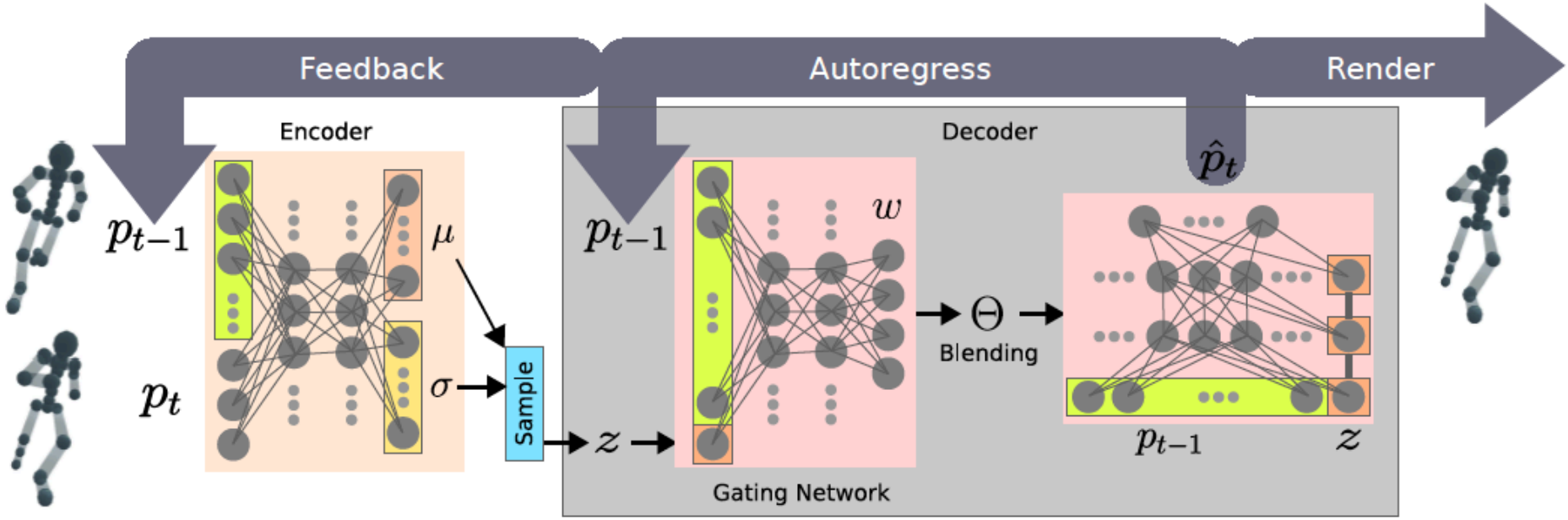


Fig. 2. The conditional VAE has two parts. The encoder takes past (p_{t-1}) and current (p_t) pose as input and outputs both μ and σ, which is then used t sample a latent variable z. The decoder uses p_{t-1} and z to reconstruct $\hat{p}_t$. For the decoder, we use a MANN-style mixture-of-expert neural network. Whe using scheduled sampling during training or at run-time, the decoder output, $\hat{p}_t$, is fed back as input for generating the next prediction.

Ling et al 20

Motion VAE - making long time motions

- Autoregressive model on its own isn't much good
  - drift
- Use RL to produce latent variables
- Thoroughly enjoyable demo at
  - https://belinghy.github.io/projects/MVAE/
- Note important point:
  - locomotion has really quite simple structure, viewed right

Motion VAE - N+Q

- Q: Could you use this to smooth parametric motions?
  - eg use VAE to encode many different reaches
  - latent variable explains reach strategy
  - now sample/control for different reaches?
- Q: Build new/better motion graph links?
- Q: Impose long term good behavior
  - with motion graph/some other structure