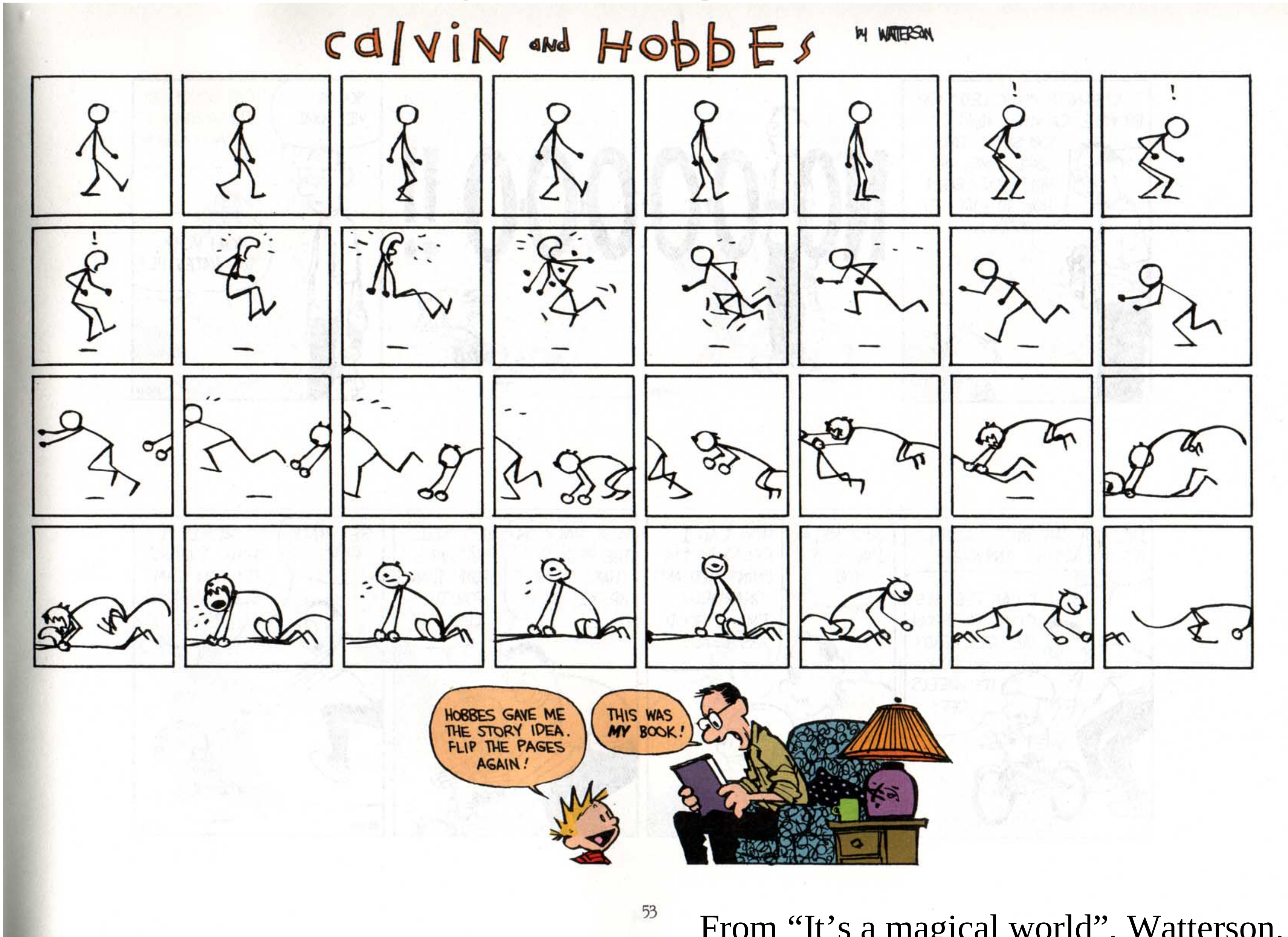
# Animation

- 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

# 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

# Keyframing




From "It's a magical world", Watterson, 1996

# Keyframing - issues

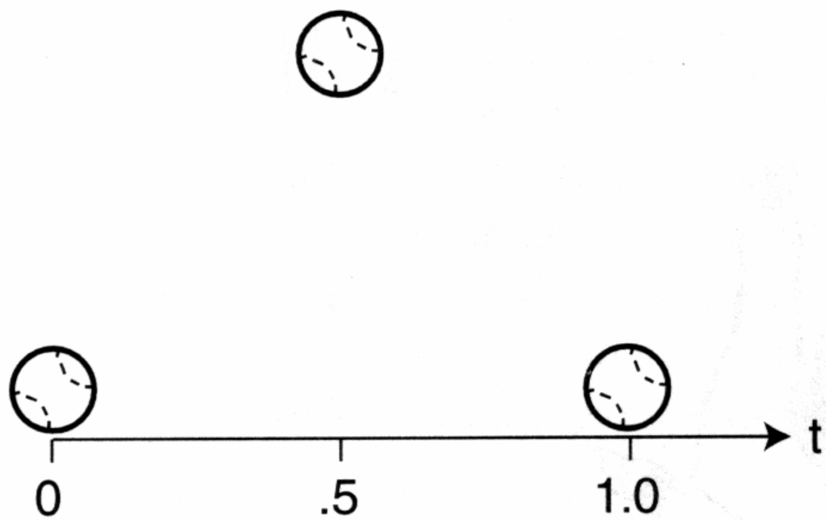


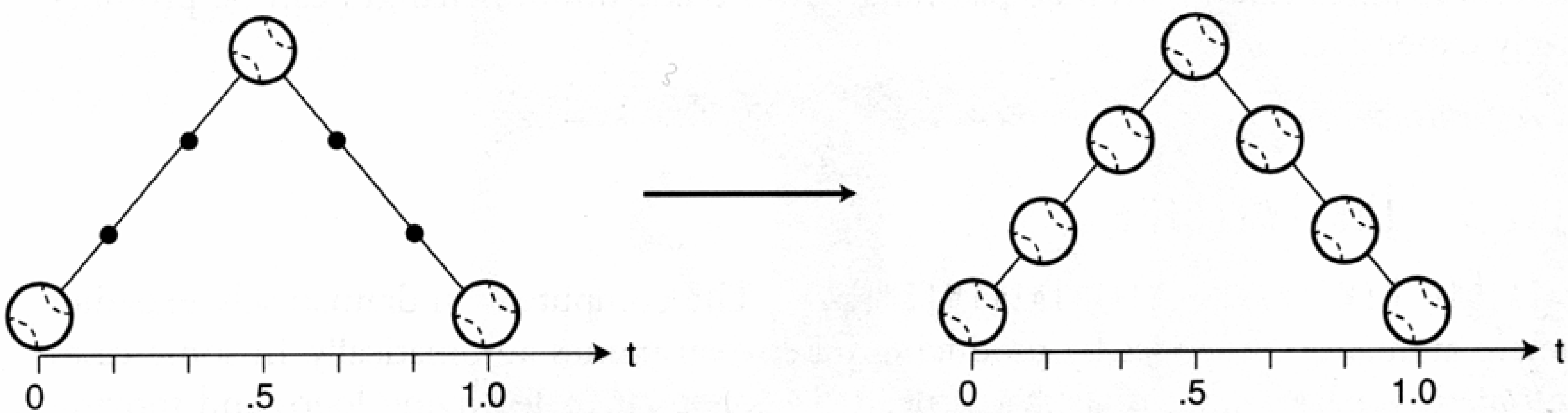

**Figure 10.4 Three keyframes.** Three keyframes representing a ball on the ground, at its highest point, and back on the ground.

- 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

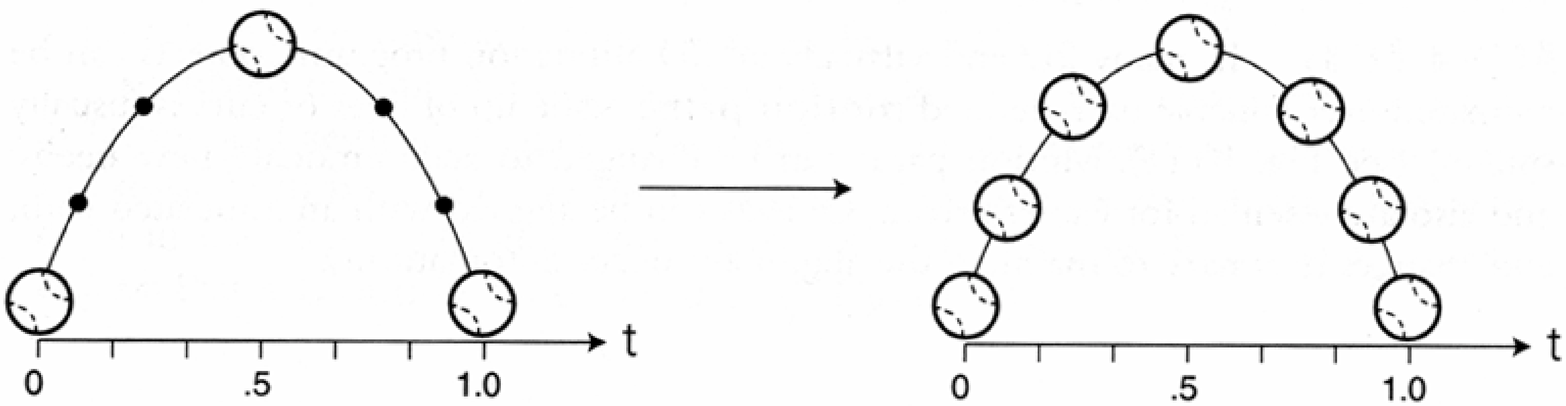



*Figure 10.9* **Inbetweening with nonlinear interpolation.** Nonlinear interpolation can create equally spaced inbetween frames along curved paths. The ball still moves at a constant speed. (Note that the three keyframes used here and in Fig. 10.10 are the same as in Fig. 10.4.)

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

# Modify the parameter, too

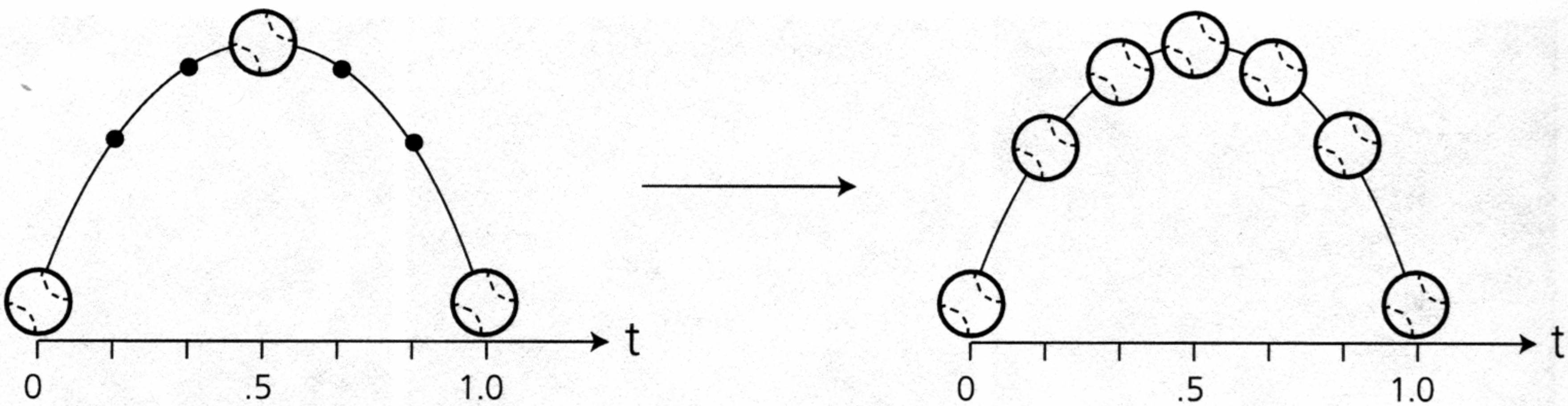
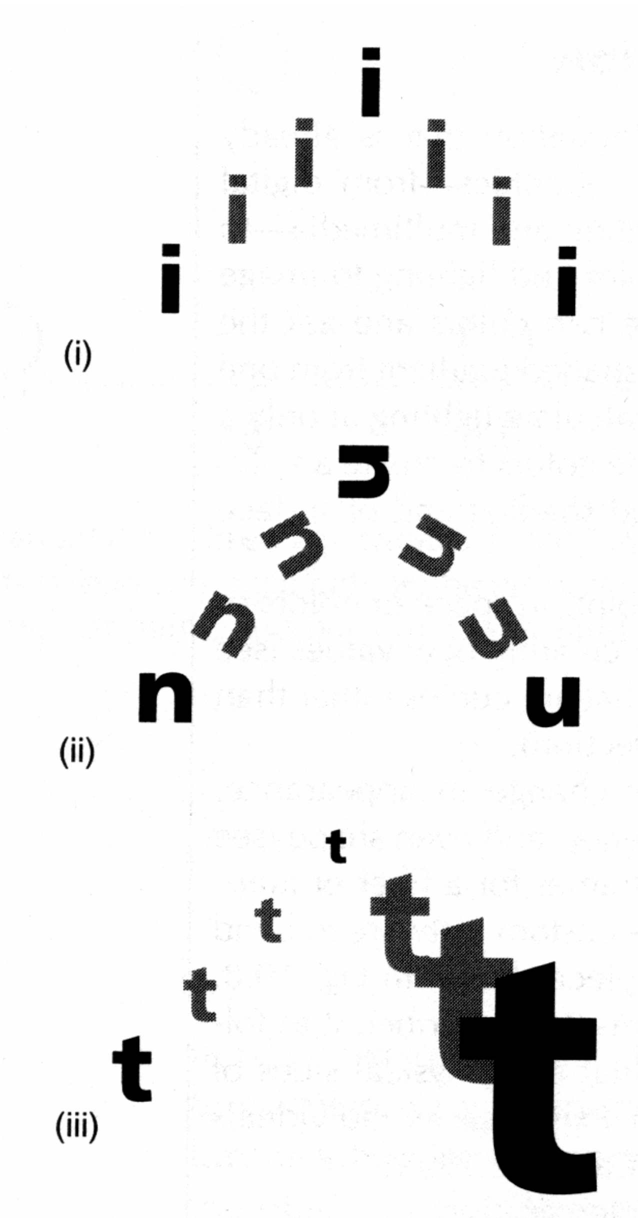



**Figure 10.10 Inbetweening with nonlinear interpolation and easing.** The ball changes speed as it approaches and leaves keyframes, so the dots indicating calculations made at equal time intervals are no longer equidistant along the path.

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

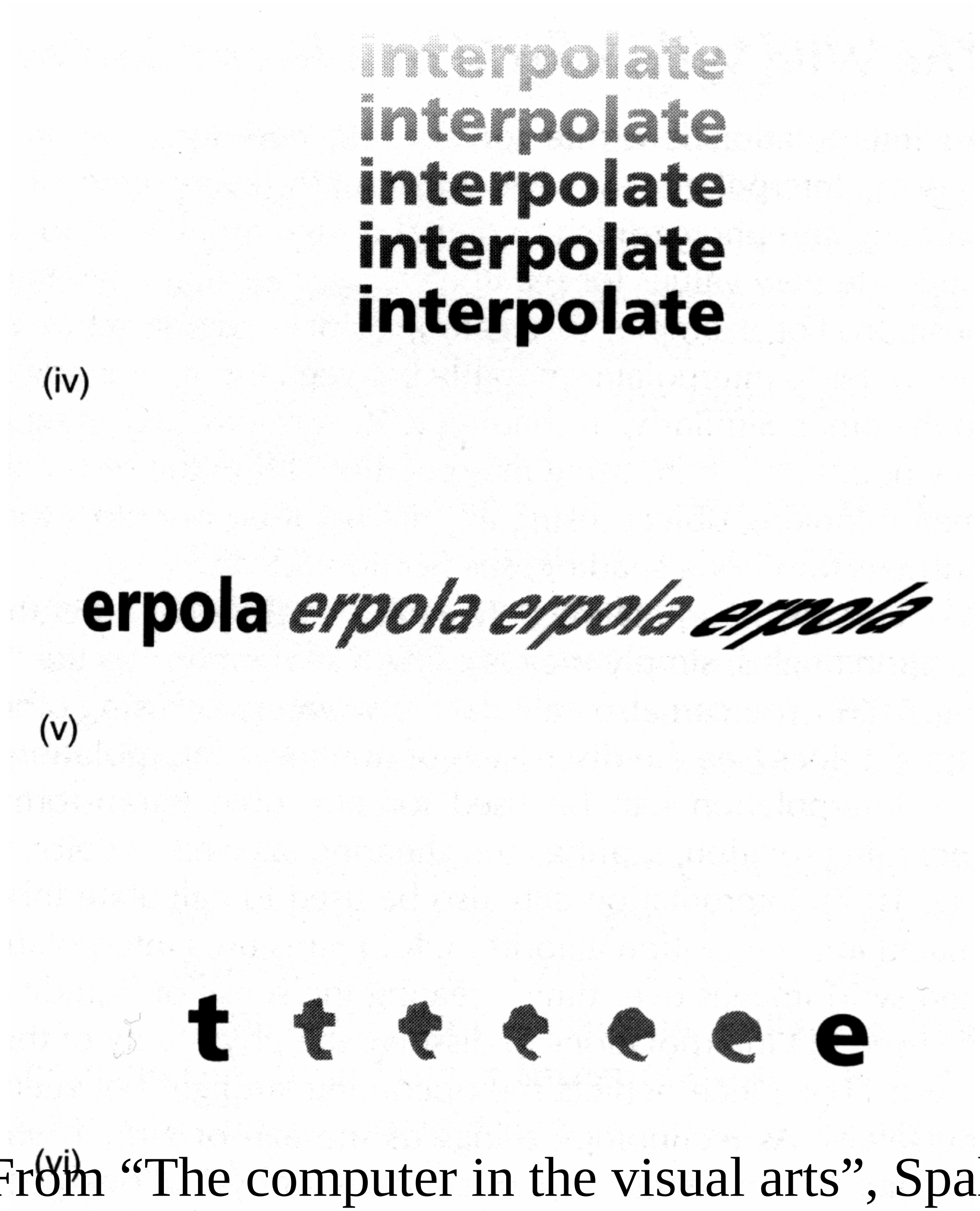



Position

Position and orientation

Position and scale

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

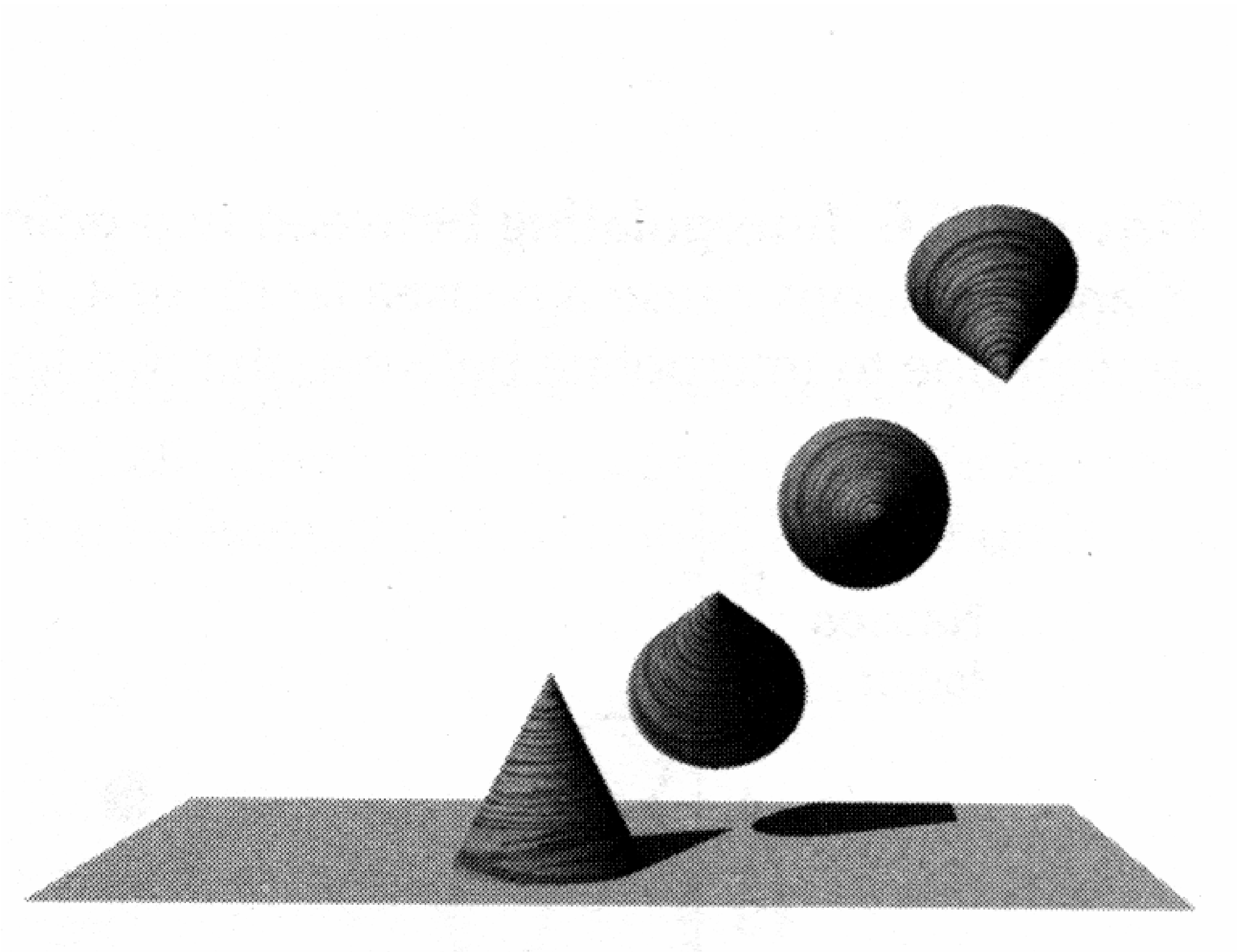



Grey-level

Shear

Shape

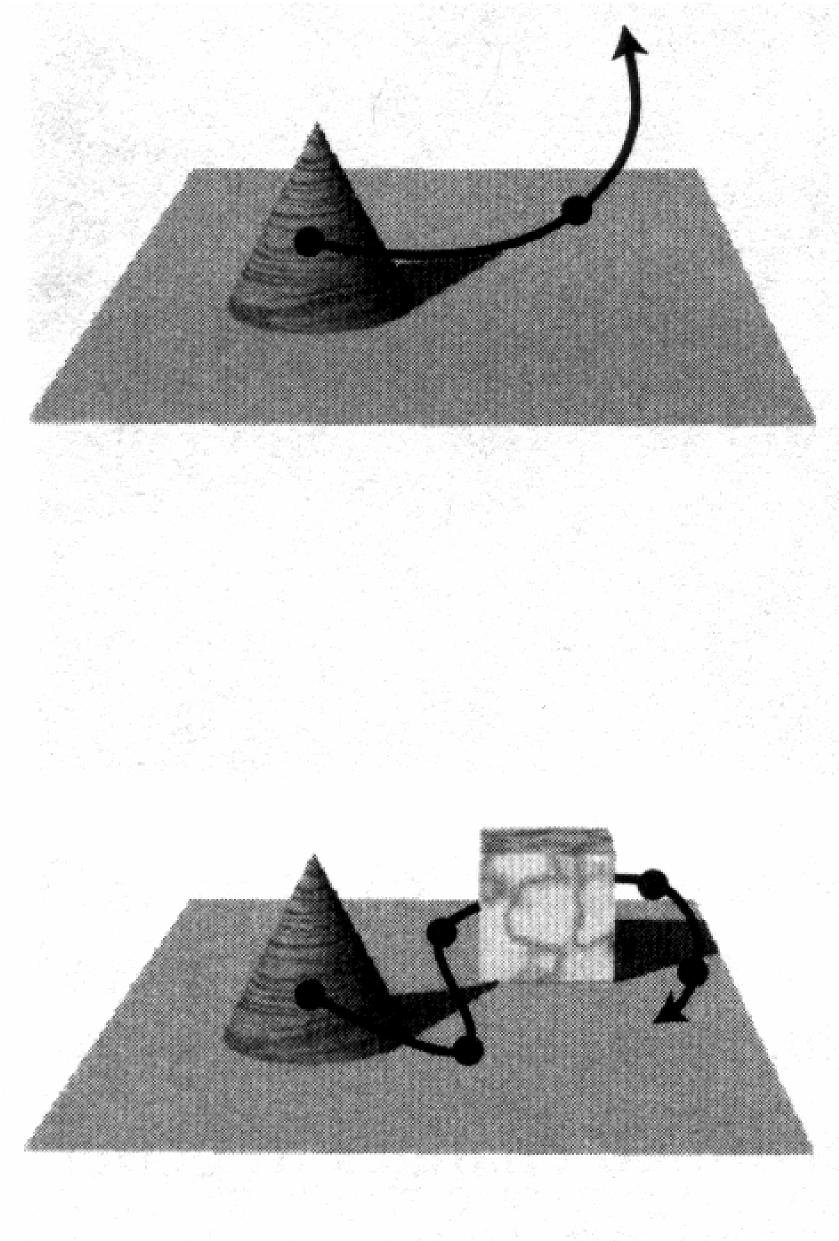
From “The computer in the visual arts”, Spalter, 1999



Position and orientation:

note that the position travels along a motion path

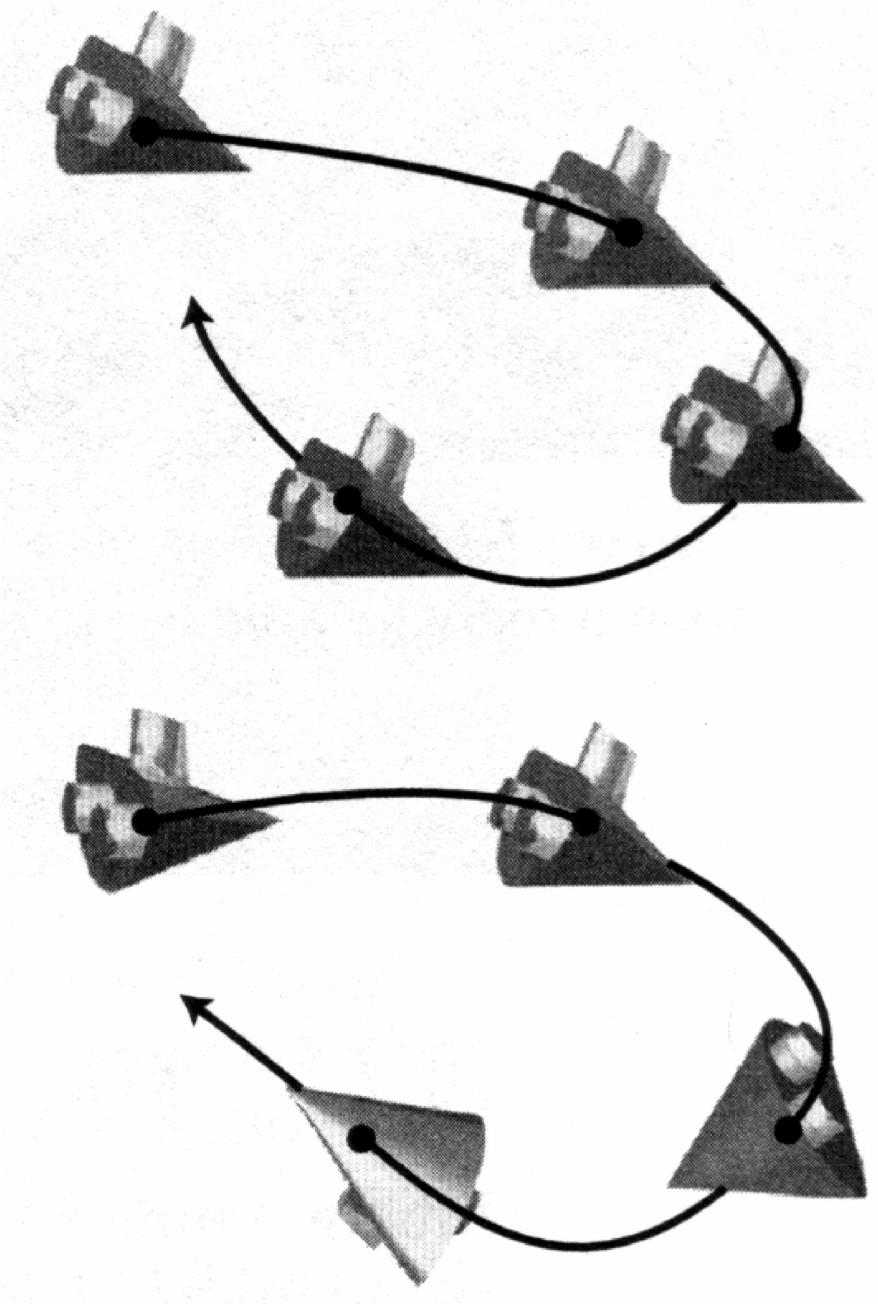
From “The computer in the visual arts”, Spalter, 1999



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

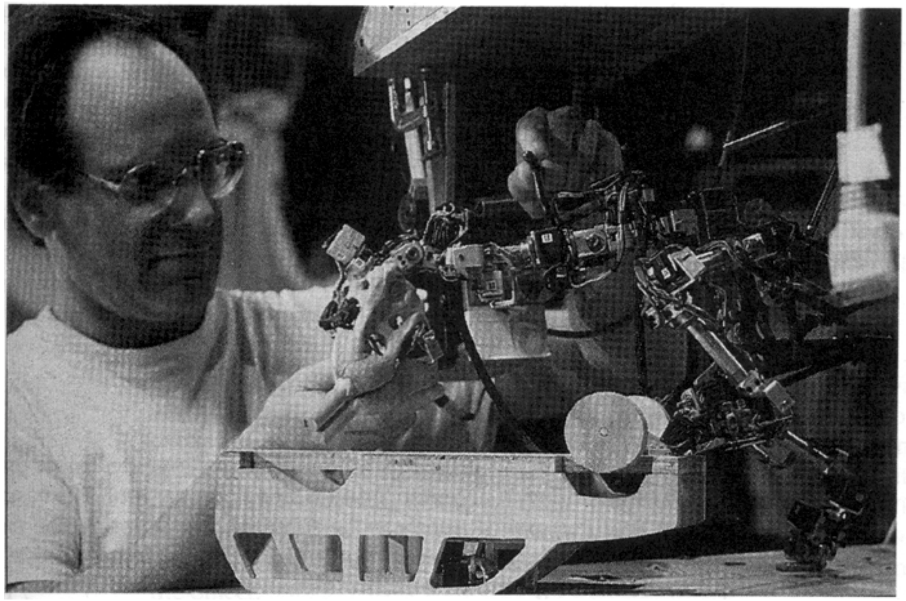


Interpolating orientation gives greater realism. Notice that the tangent to the motion path gives a great cue to the orientation of the object.

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

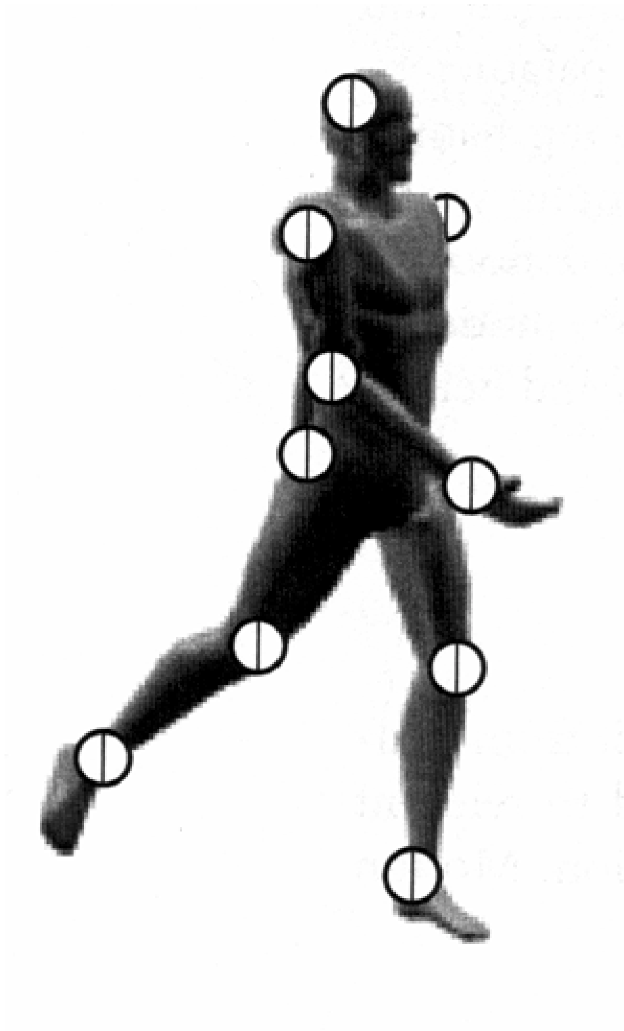
# Stop motion

- 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 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

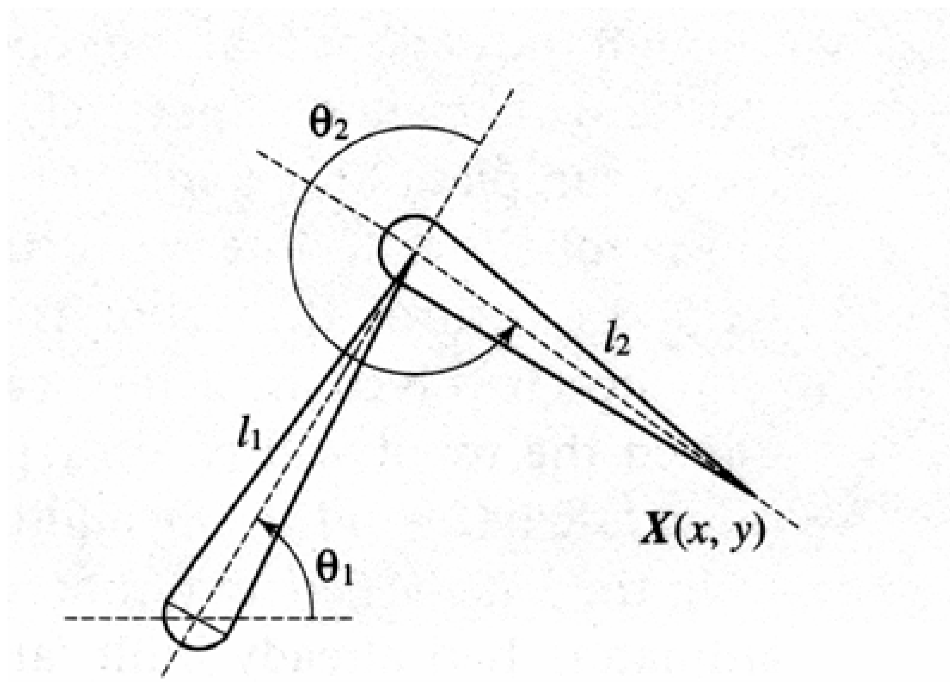
# Motion capture



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

# 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?

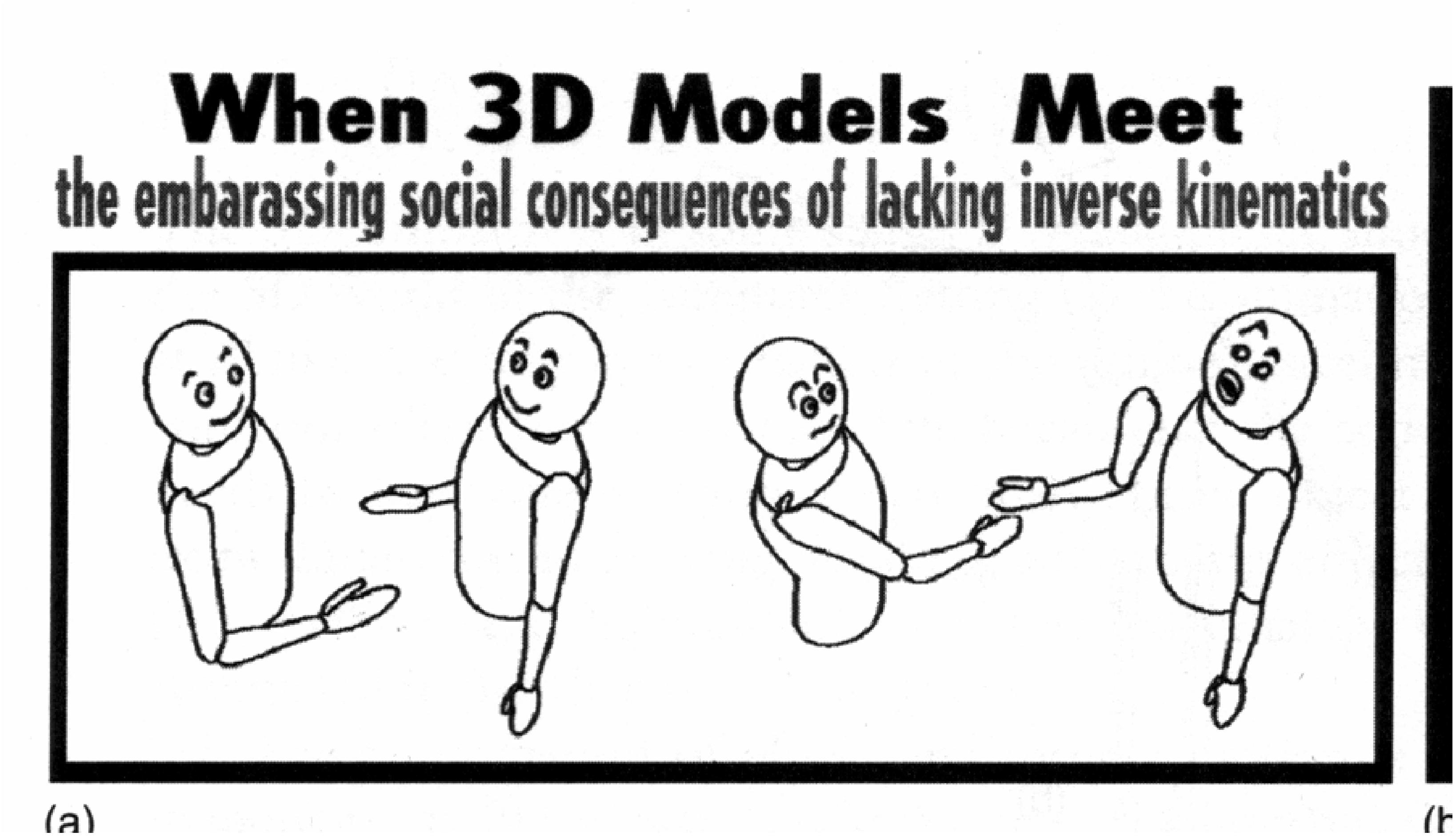



From “The computer Image”, Watt and Policarpo, 1998

# Inverse Kinematics

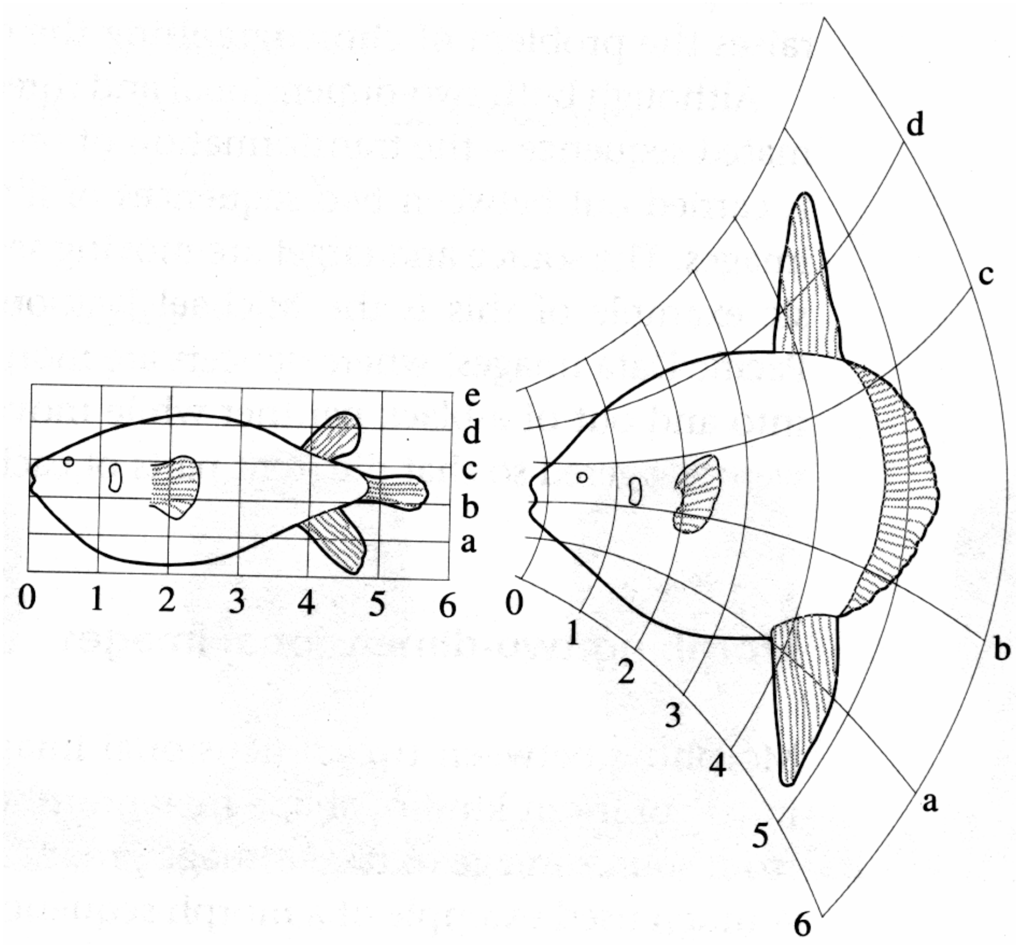

From “The computer Image”,
Watt and Policarpo, 1998

# Inverse Kinematics



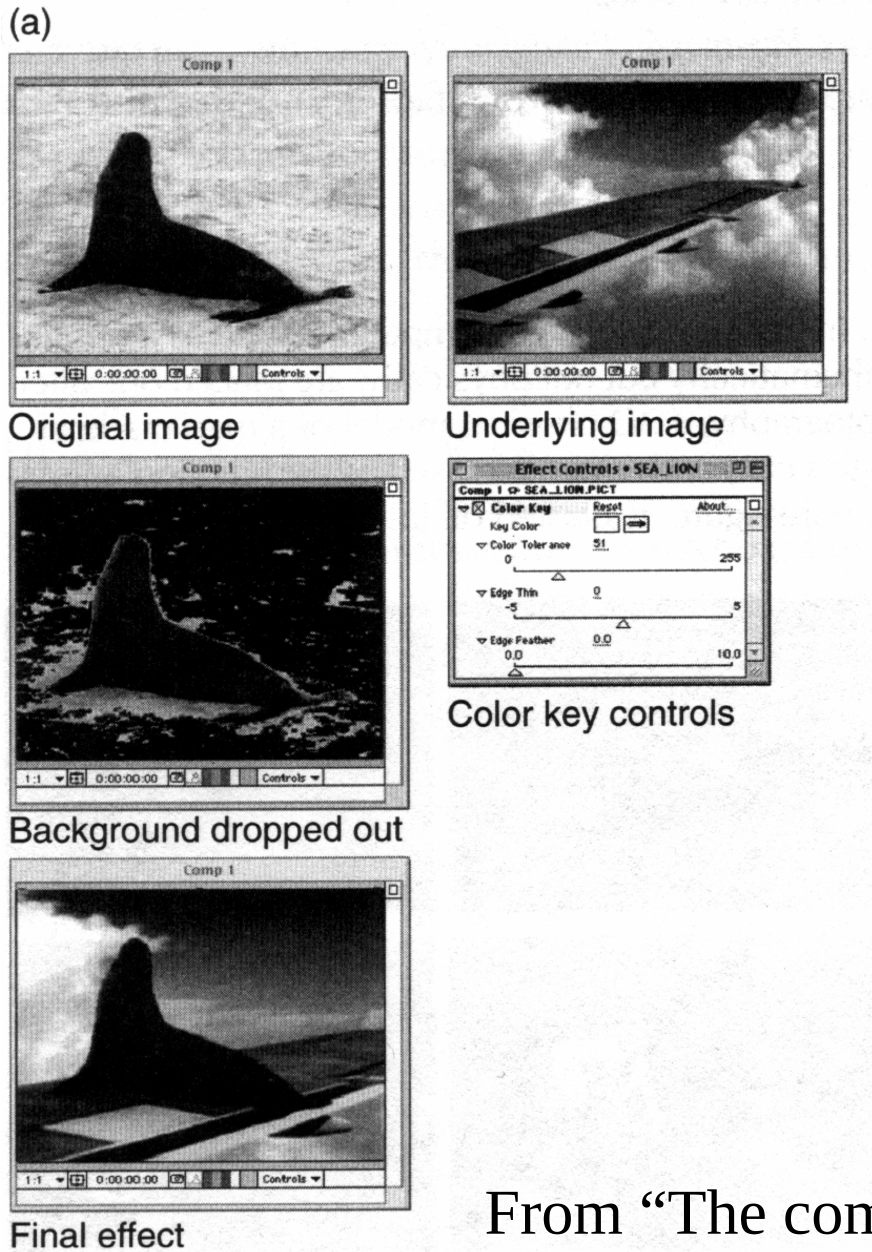

From "The computer in the visual arts", Spalter, 1999

# Morphing




From "On growth and Form", D'Arcy Thompson

# Compositing




- 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

# 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?

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

- 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

# 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.

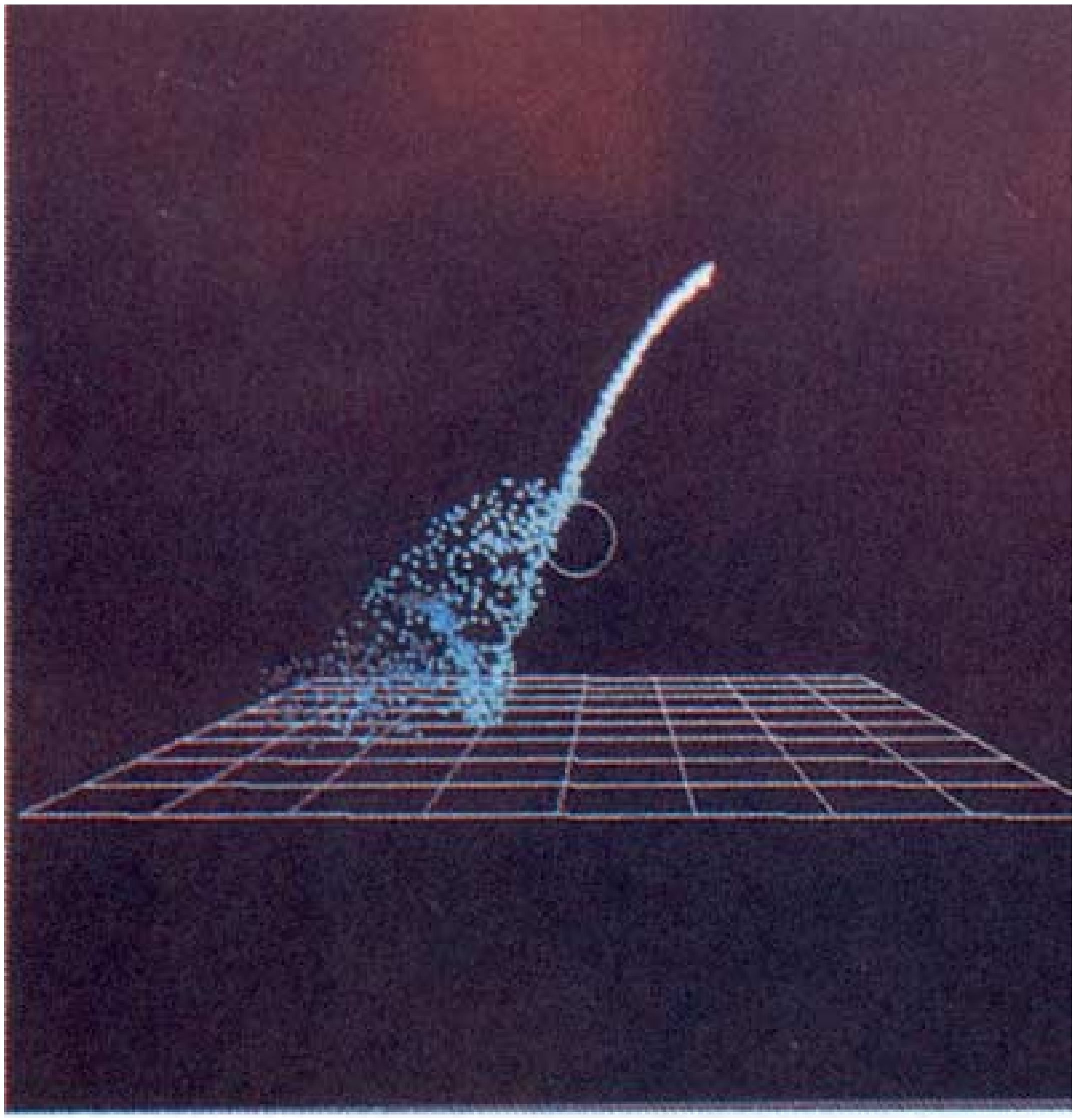
# Dynamics - Particle systems

- Objects move due to the effects of forces
  - usually gravity and collisions
- There is a source of particles
- 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.

# Particle explosion

Animation science:  http://www.anisci.com/main.html

# Particle water



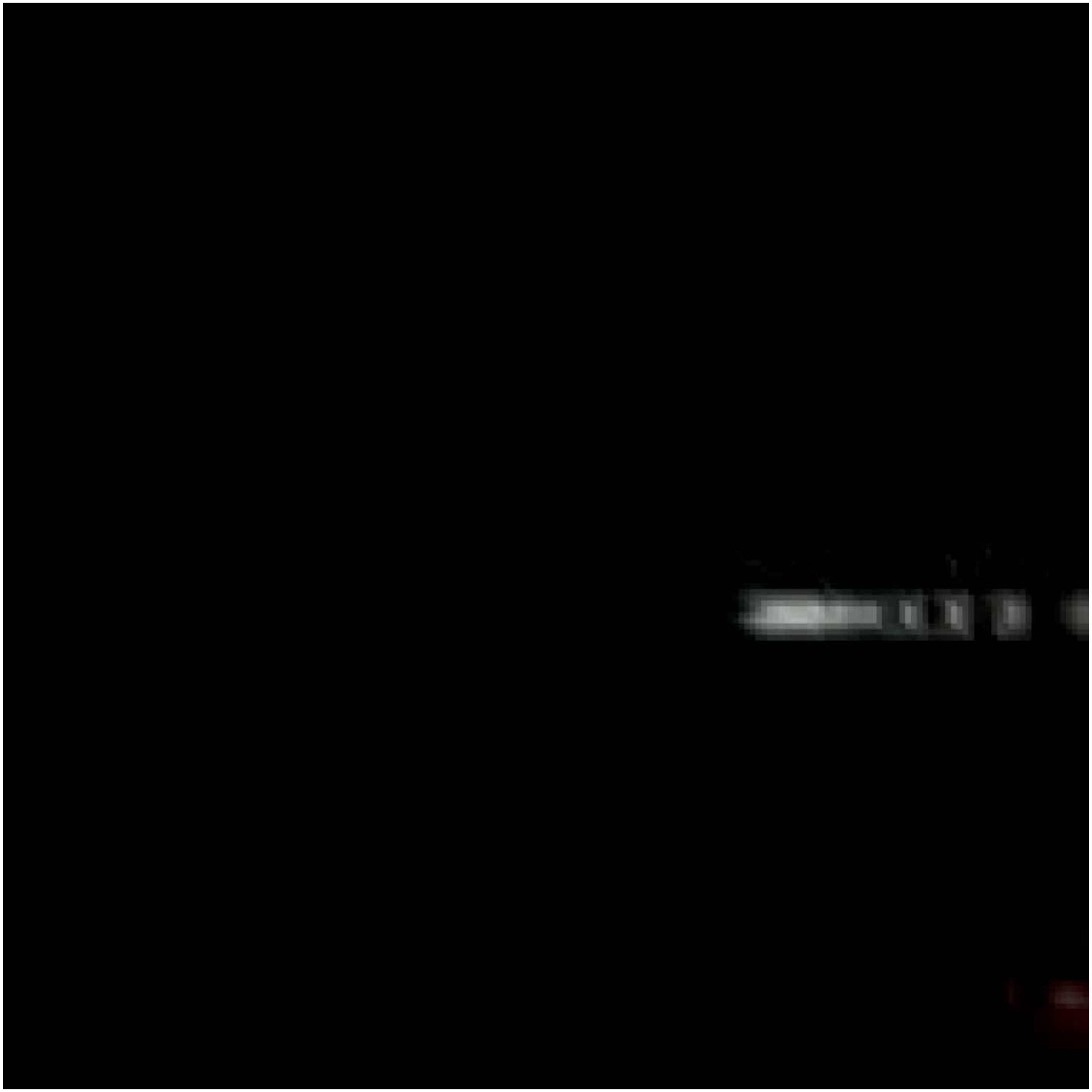
# Particle Torch

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

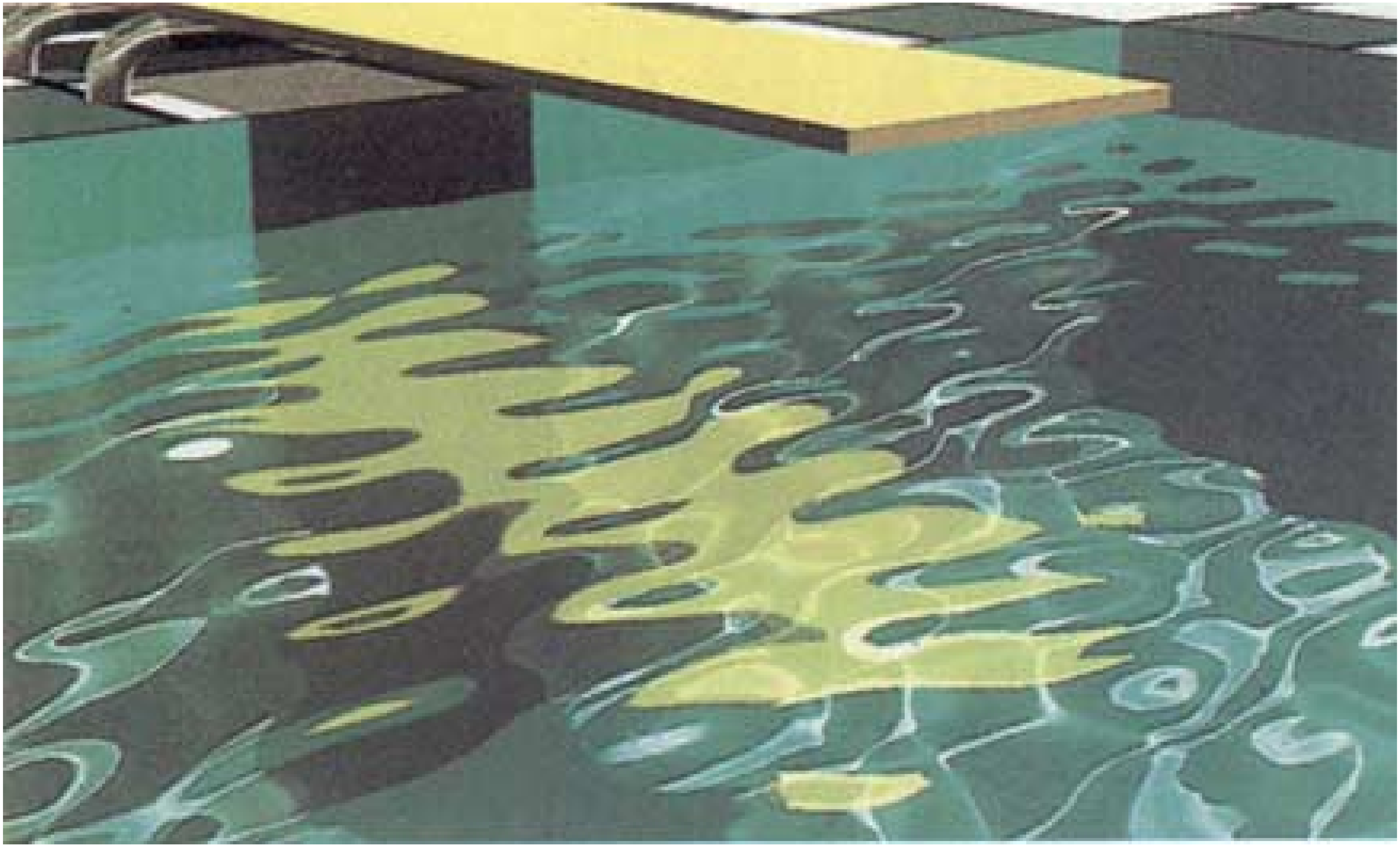
http://www.arch.columbia.edu/manuals/Softimage/3d_learn/GUIDED/PARTICLES/p_first.htm

# Particle surface

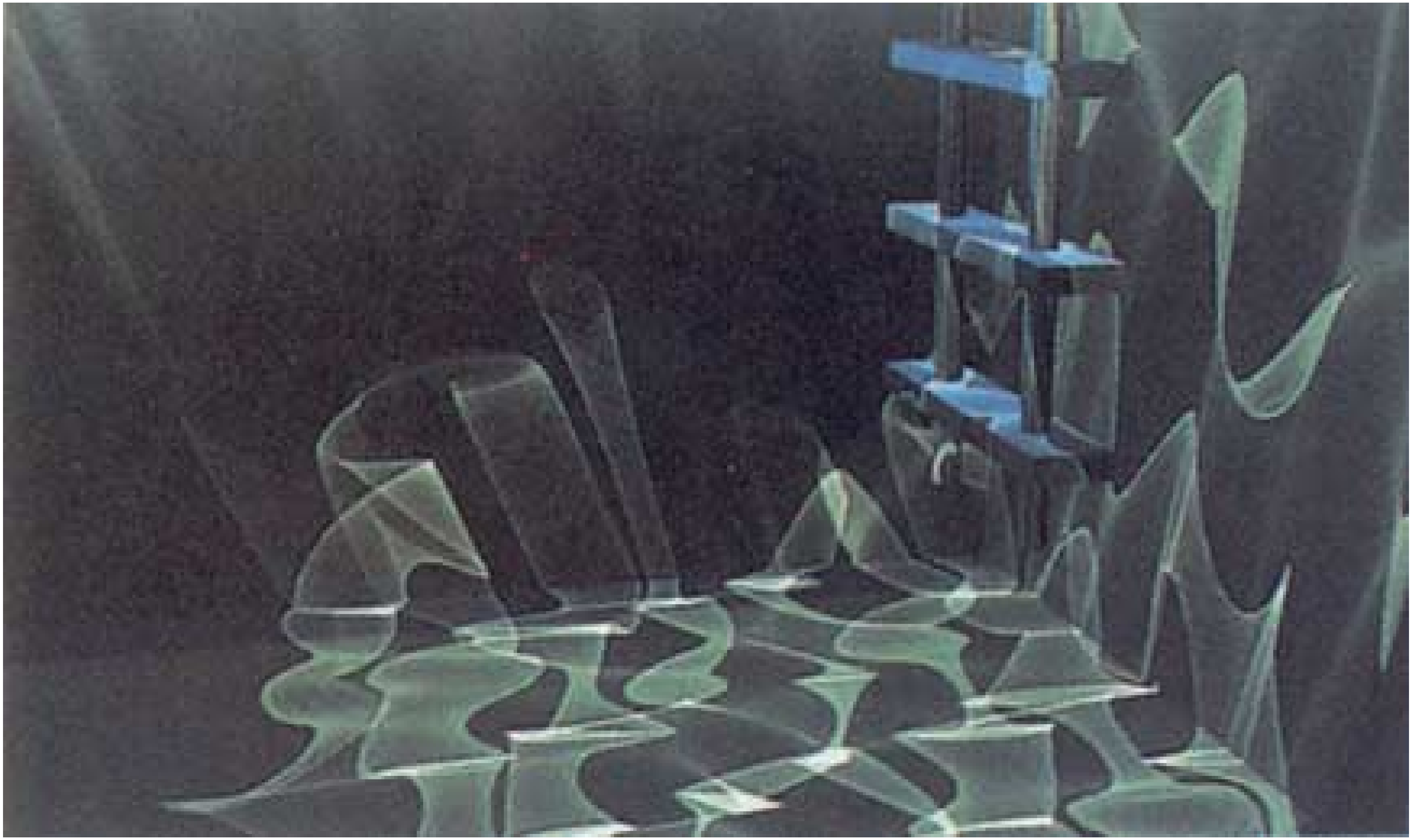
Particles stick to an implicit surface and repel one another
Andrew Witkin,
http://www.cs.cmu.edu/afs/cs/user/aw/www/gallery.html



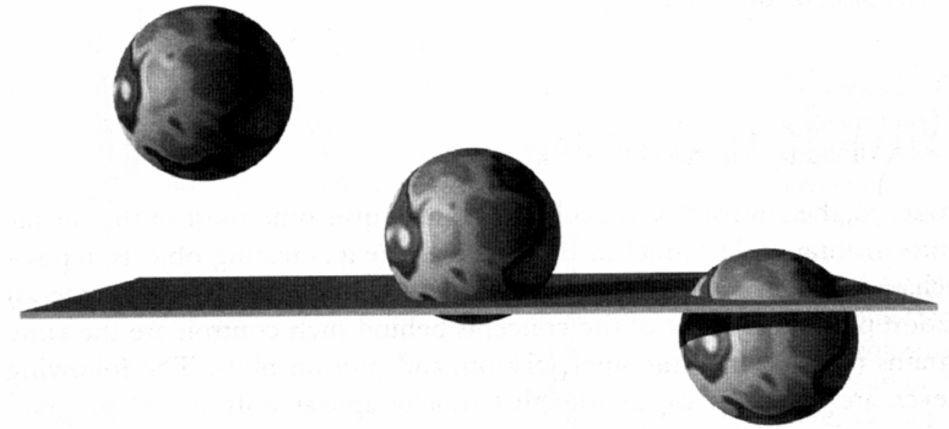
# Procedural waves

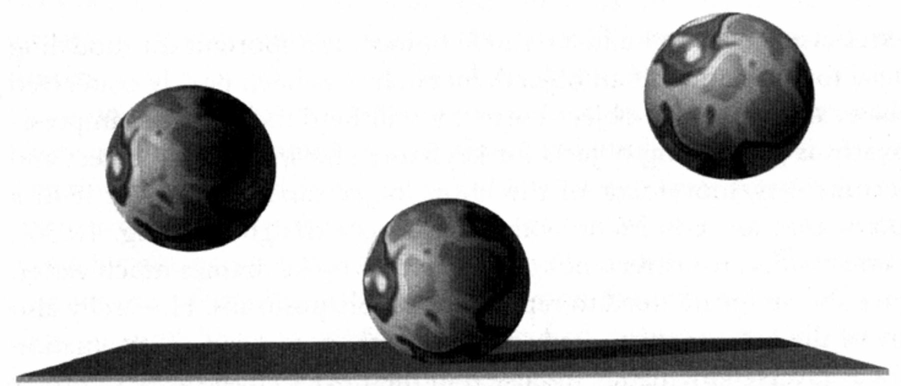


# Procedural waves

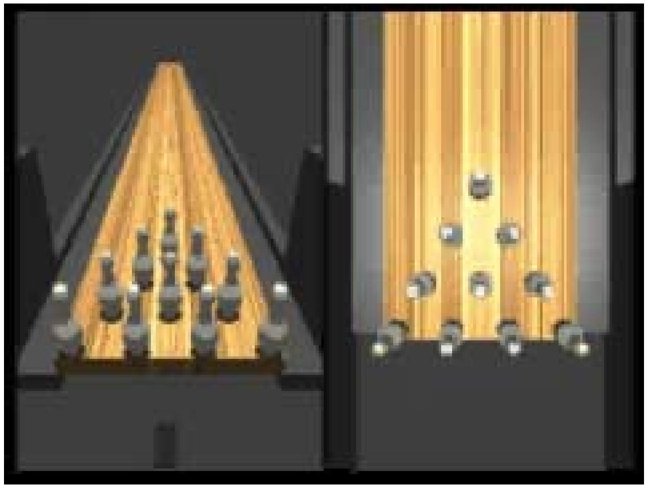


# Collisions

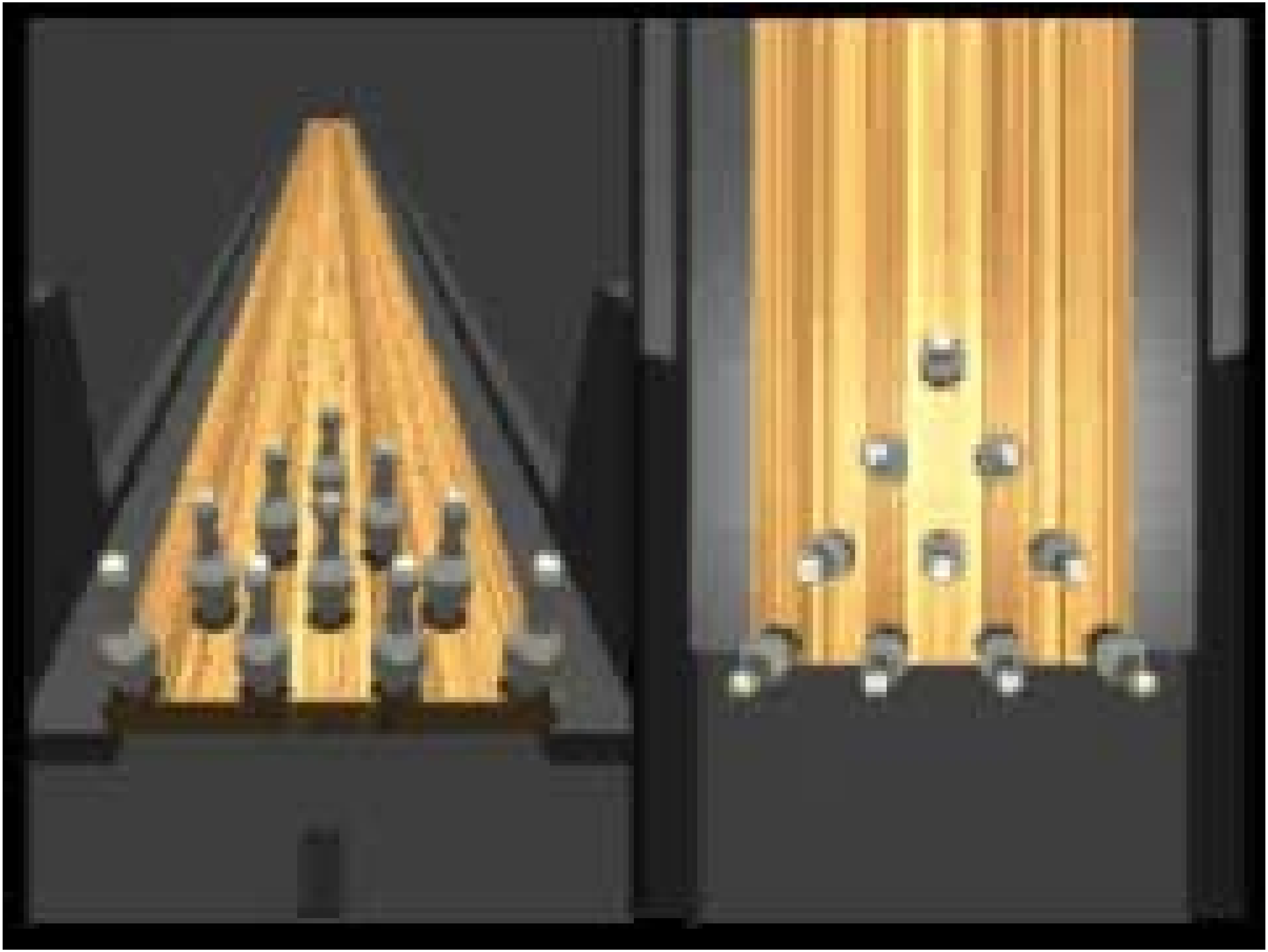

(a)

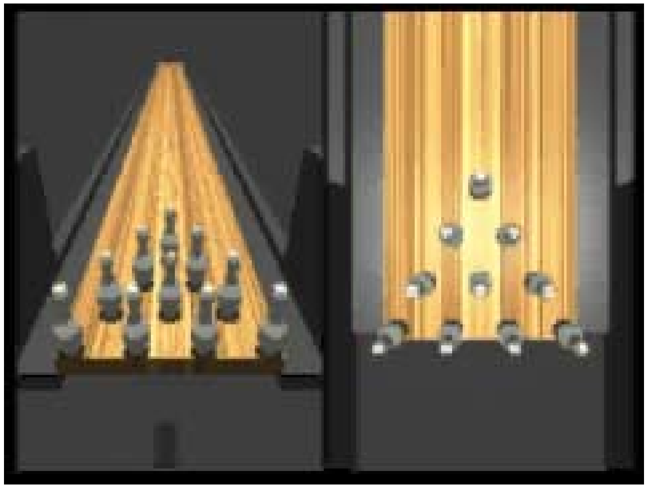

(b)

- Natural dynamic model - objects move freely under gravity till they collide
- Collisions with point particles are easy
- Collisions with more complex shapes are not
  - spheres are an exception
  - hierarchy helps
- For accurate simulation of physical dynamics, it is essential to identify the first collision.

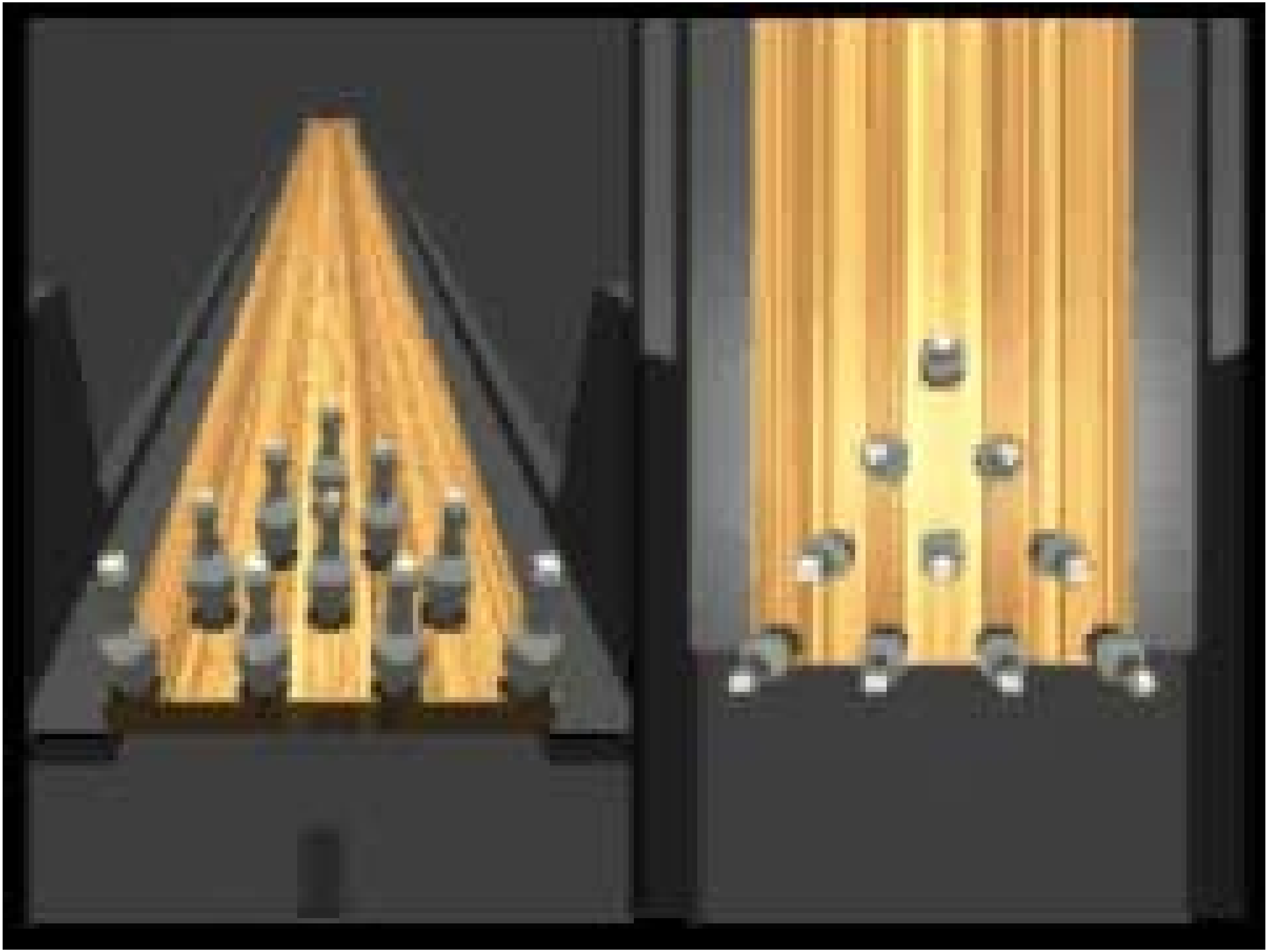


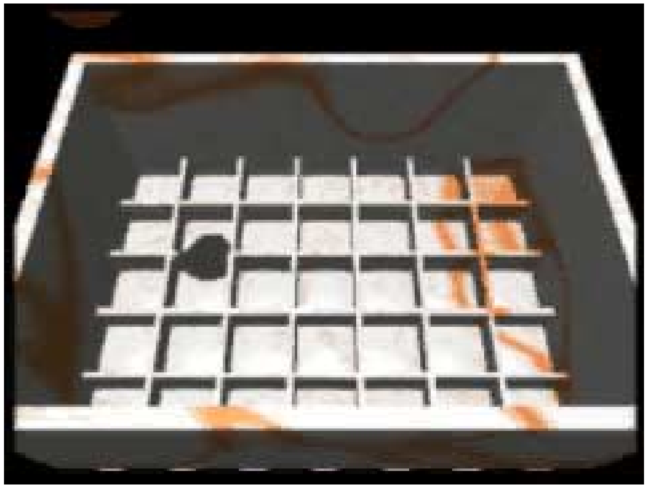
http://www.cs.wisc.edu/~schenney/research/directing/mcmc/web-video.html



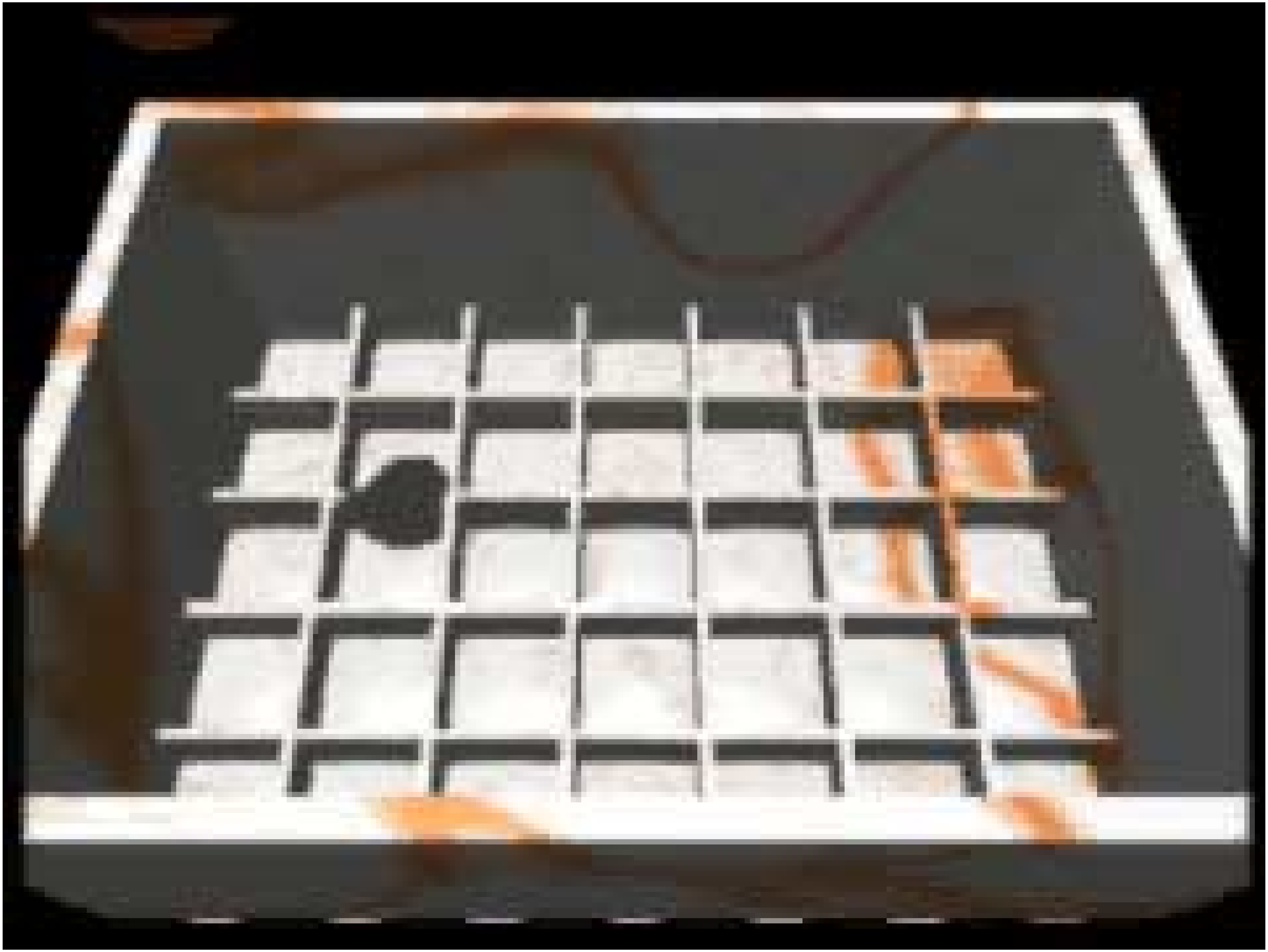


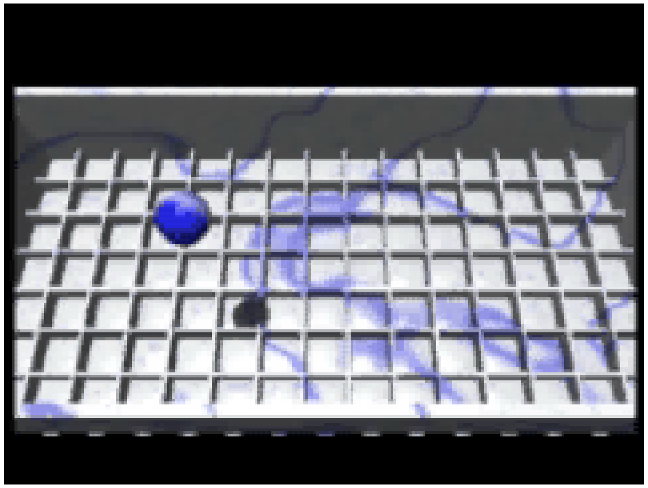
http://www.cs.wisc.edu/~schenney/research/directing/mcmc/web-video.html



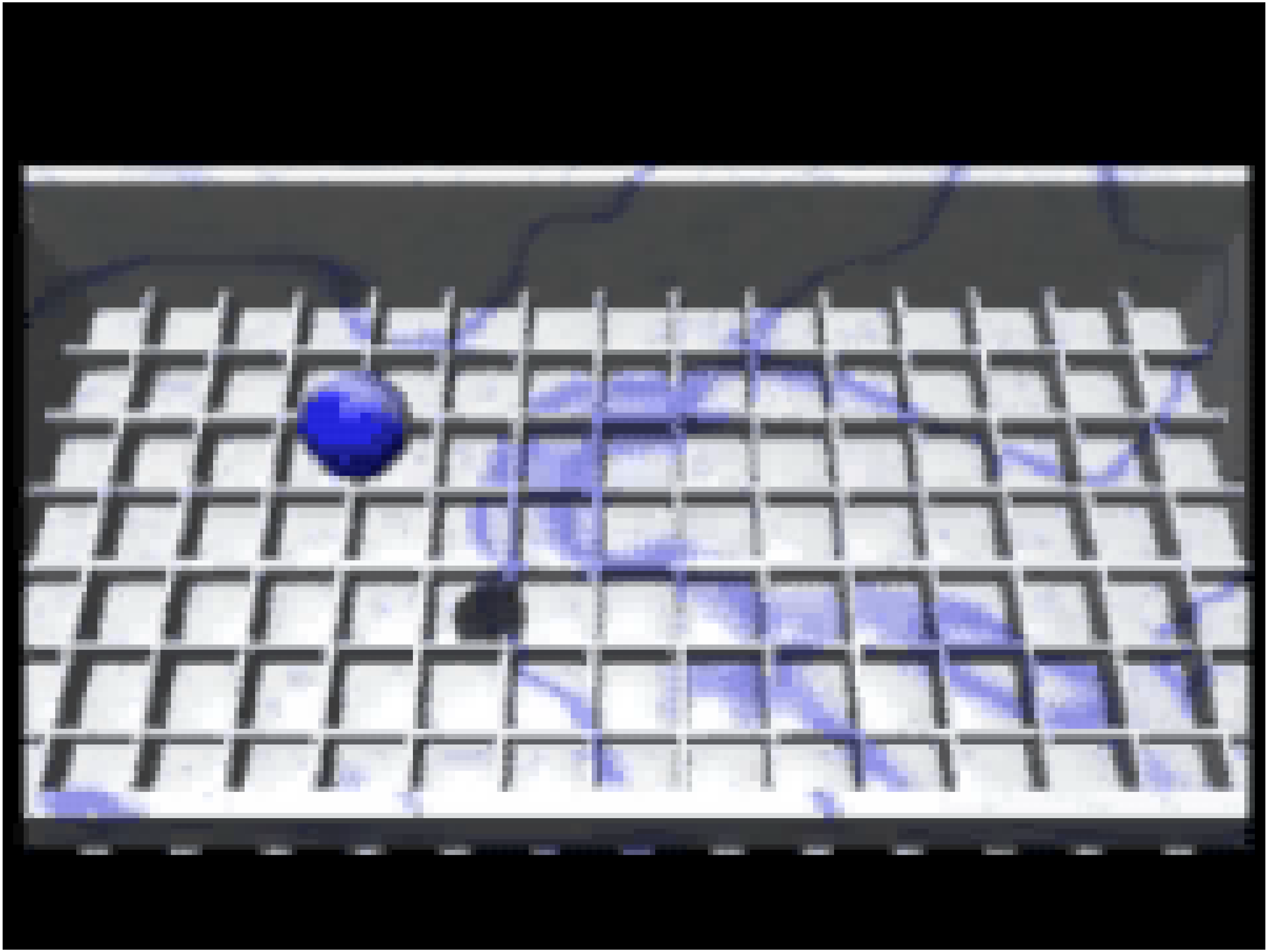


http://www.cs.wisc.edu/~schenney/research/directing/mcmc/web-video.html





http://www.cs.wisc.edu/~schenney/research/directing/mcmc/web-video.html



http://www.cs.wisc.edu/~schenney/research/directing/mcmc/web-video.html

# Dynamics - Springs and Masses

- Objects are modelled as a line/grid/lattice of masses
- Masses are connected by linear springs
- Energy in a linear spring is $k(l-l_r)^2$
  - here k is the spring constant and $l_r$ is the rest length
- This yields system of differential equations for the state of the object (position and velocity of the masses)
- Objects can be controlled by changing rest lengths of springs

# Spring mass fish

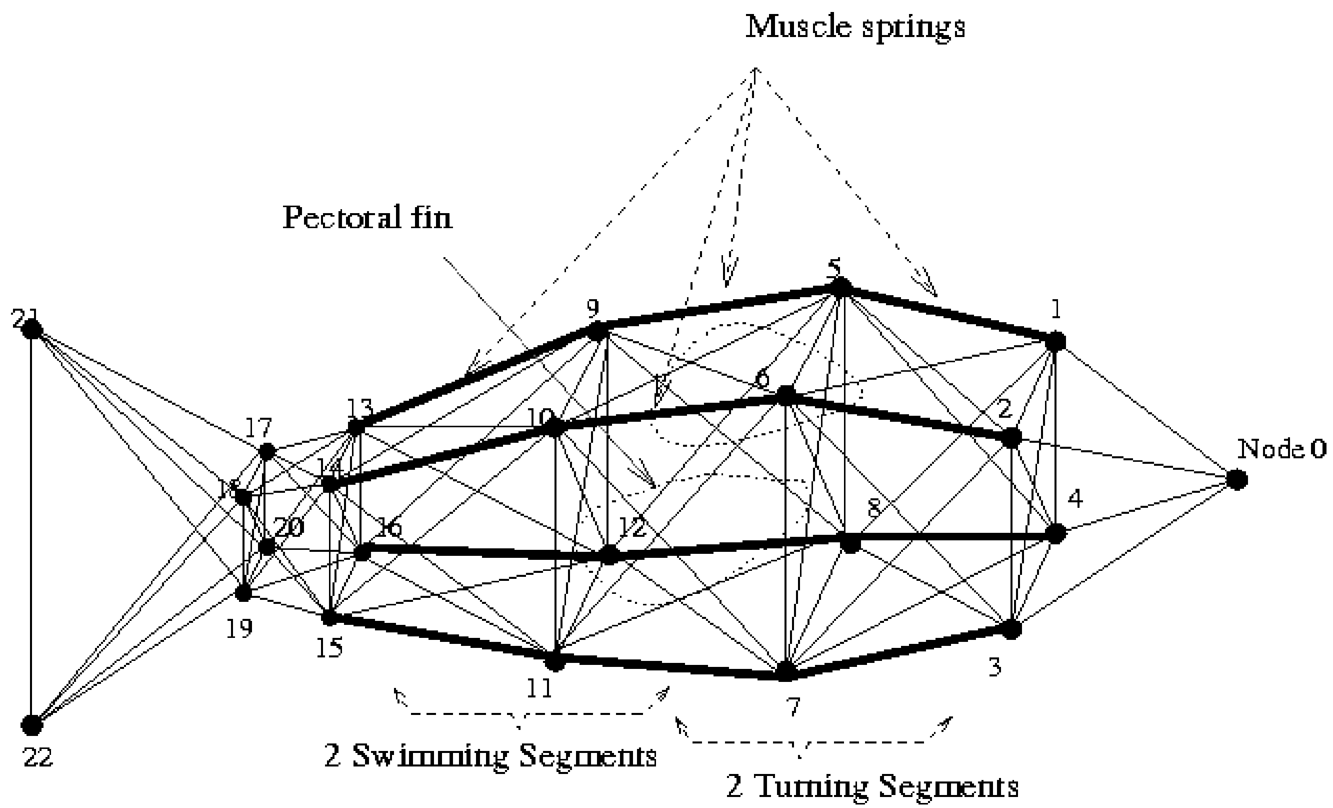



Figure 4.1: The biomechanical fish model. Black dots indicate lumped masses. Lines indicate deformable elements at their natural, rest lengths.

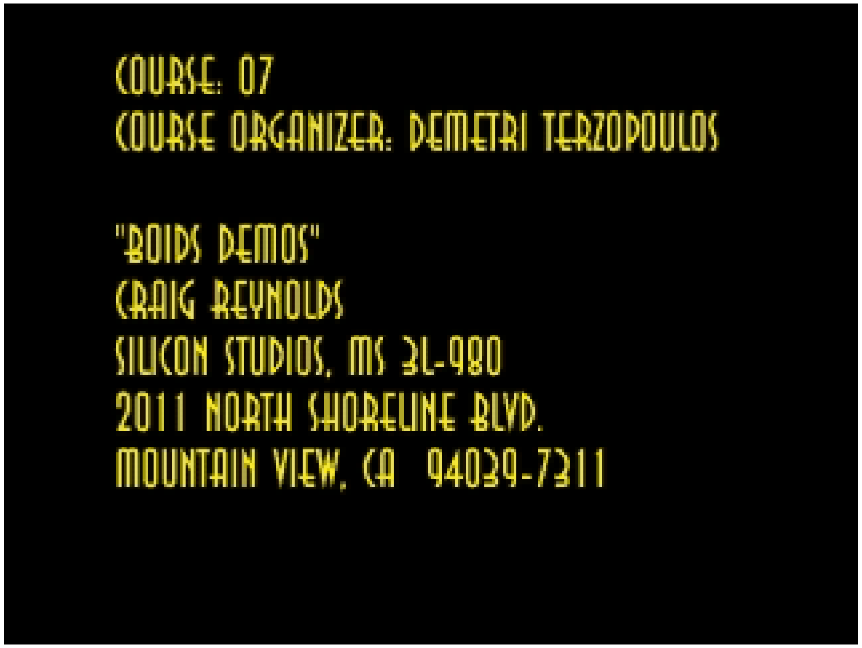
Due to Xiaoyuan Tu, http://www.dgp.toronto.edu/people/tu

# Spring Mass fish swimming

# Procedural animation - flocking

- We’d like objects to move in schools and not hit things.
- Abstraction - particle with a steerable rocket.
- 3 goals
  - Separation: steer to avoid crowding local flockmates.
  - Alignment: steer towards the average heading of local flockmates.
  - Cohesion: steer to move toward the average position of local flockmates.
- Each generates a separate acceleration request
- How to accelerate?
  - weighted sum
    - but there is a limited amount of acceleration available
  - accumulate acceleration in priority order until vector is too long, then trim back the last component.

# Boids




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

COURSE: 07
COURSE ORGANIZER: DEMETRI TERZOPOULOS

"BOIDS DEMOS"
CRAIG REYNOLDS
SILICON STUDIOS, MS 3L-980
2011 NORTH SHORELINE BLVD.
MOUNTAIN VIEW, CA  94039-7311

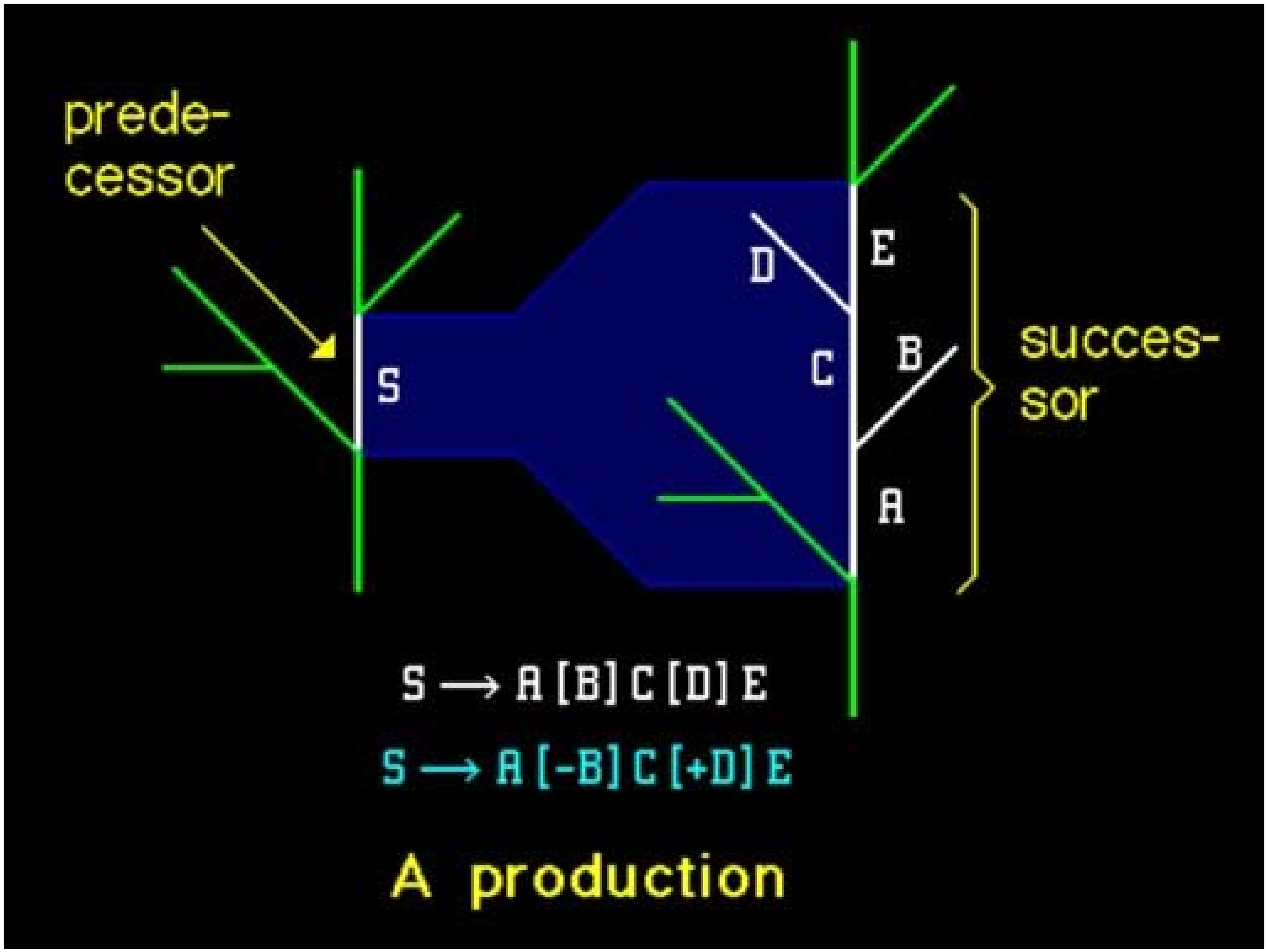
# Random offsets and subdivisions

- Subdivision using random offsets gives quite good terrain models
- Trick:
  - mesh rough model of terrain
  - subdivide, applying random offsets, but limiting the distance between the offset and the original model
  - Gives a terrain that “looks familiar”

# Fractal Terrains

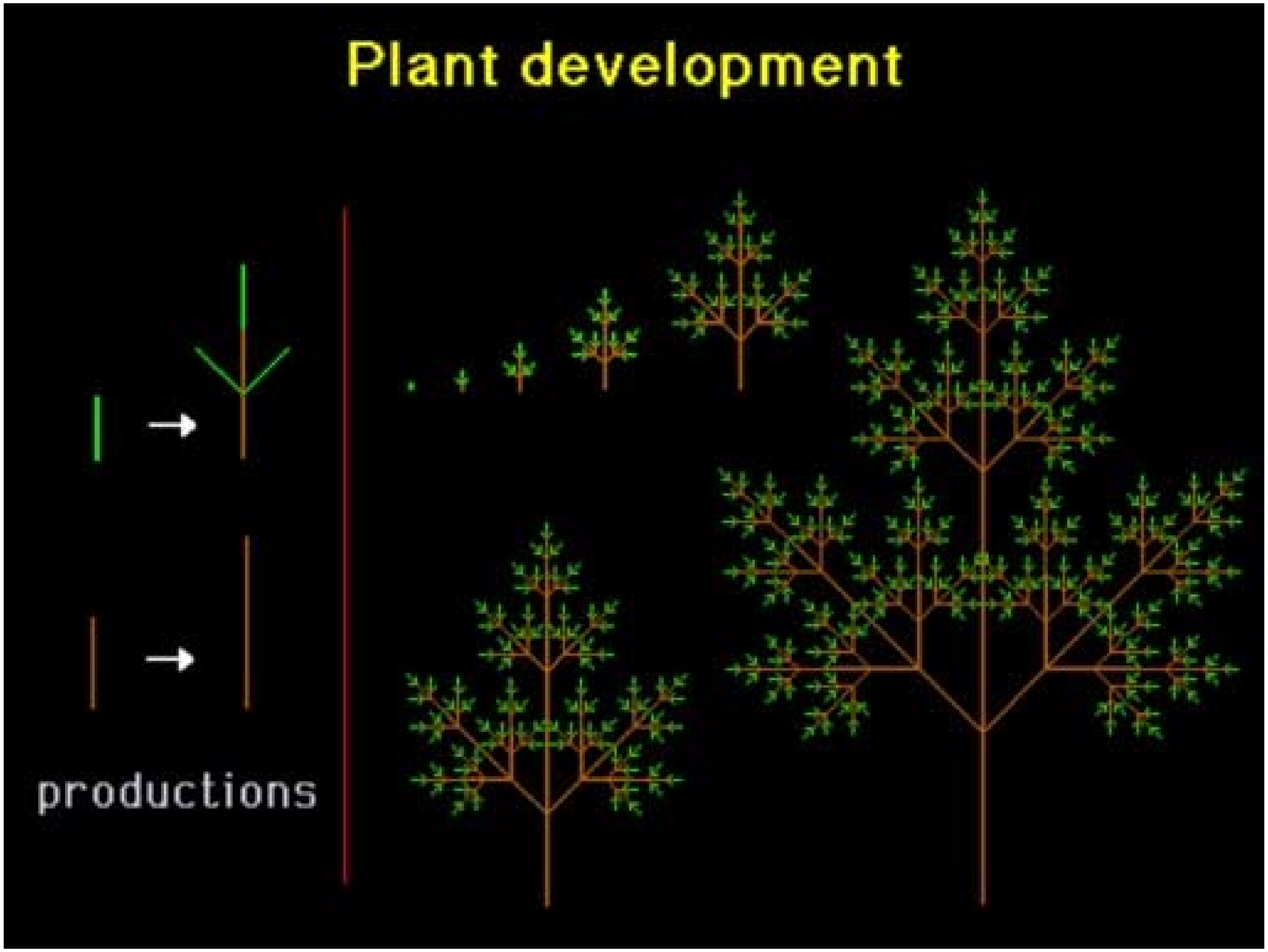

http://members.aol.com/maksoy/vistfrac/sunset.htm

# L-Systems




Prusinkiewicz, Hammel, Mech http://www.cpsc.ucalgary.ca/projects/bmv/vmm/title.html

# L-Systems

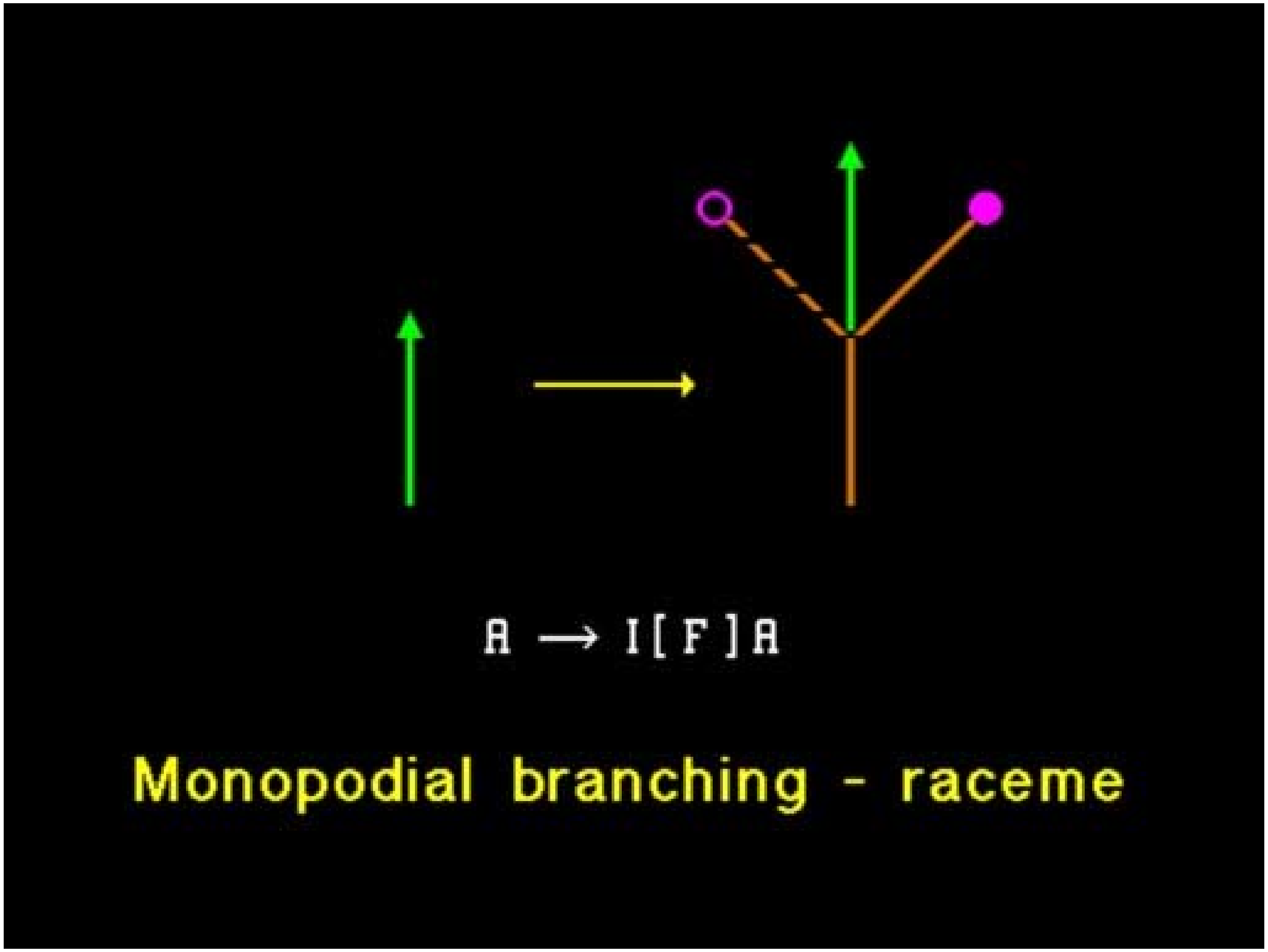



Prusinkiewicz, Hammel, Mech http://www.cpsc.ucalgary.ca/projects/bmv/vmm/title.html

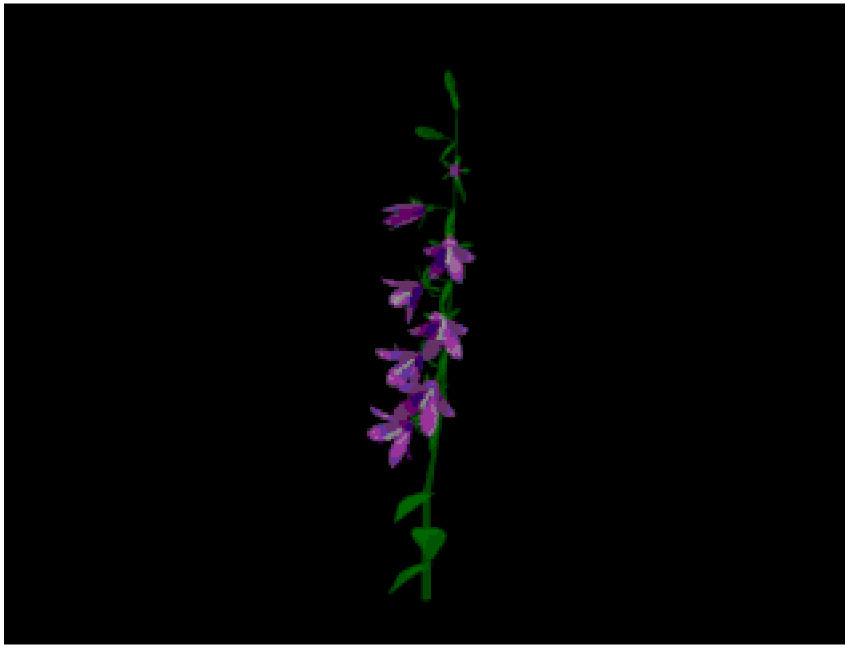
# L-System plant growing

Prusinkiewicz, Hammel, Mech http://www.cpsc.ucalgary.ca/projects/bmv/vmm/title.html

# Flowers at side branches

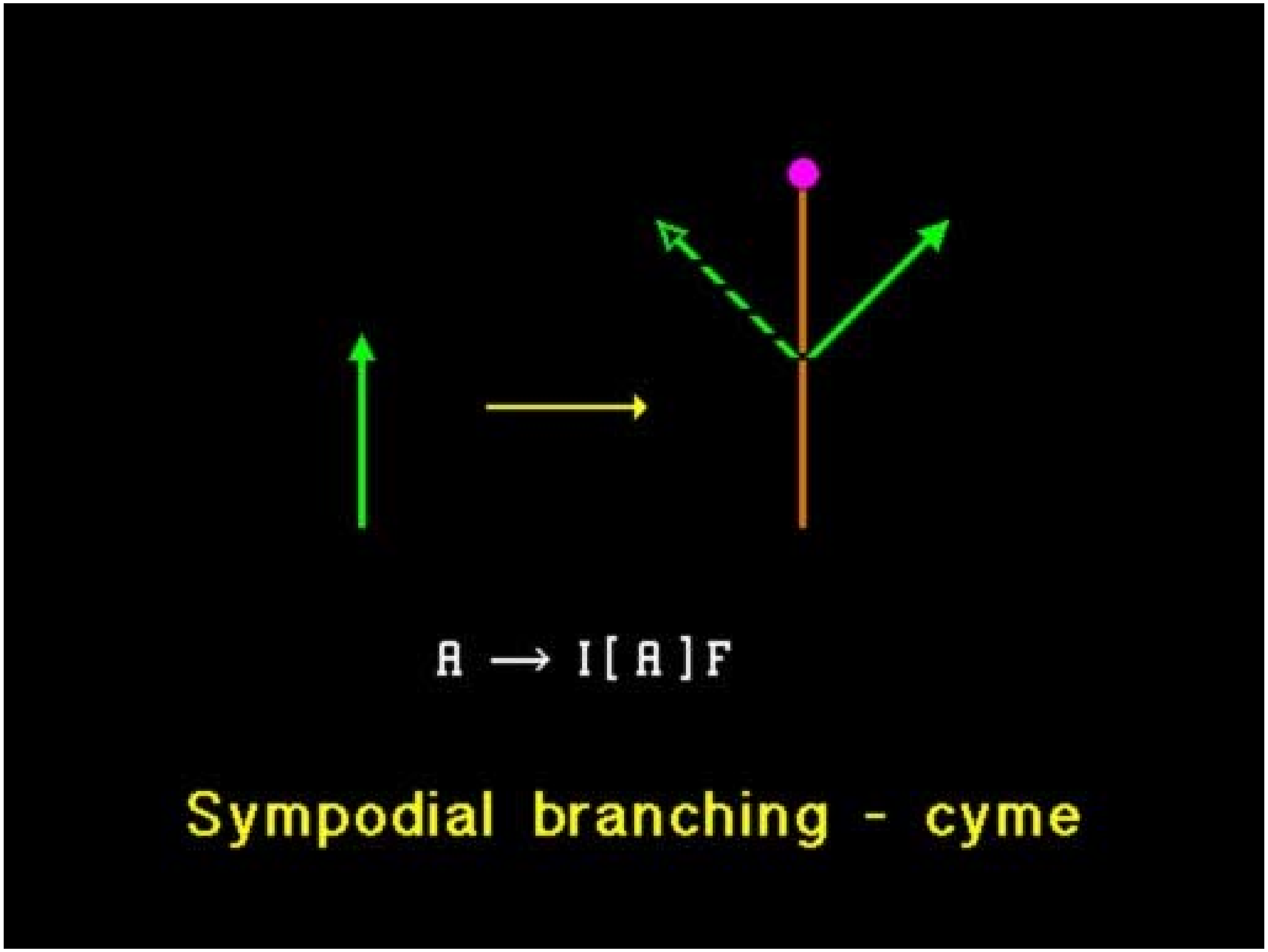



Prusinkiewicz, Hammel, Mech http://www.cpsc.ucalgary.ca/projects/bmv/vmm/title.html

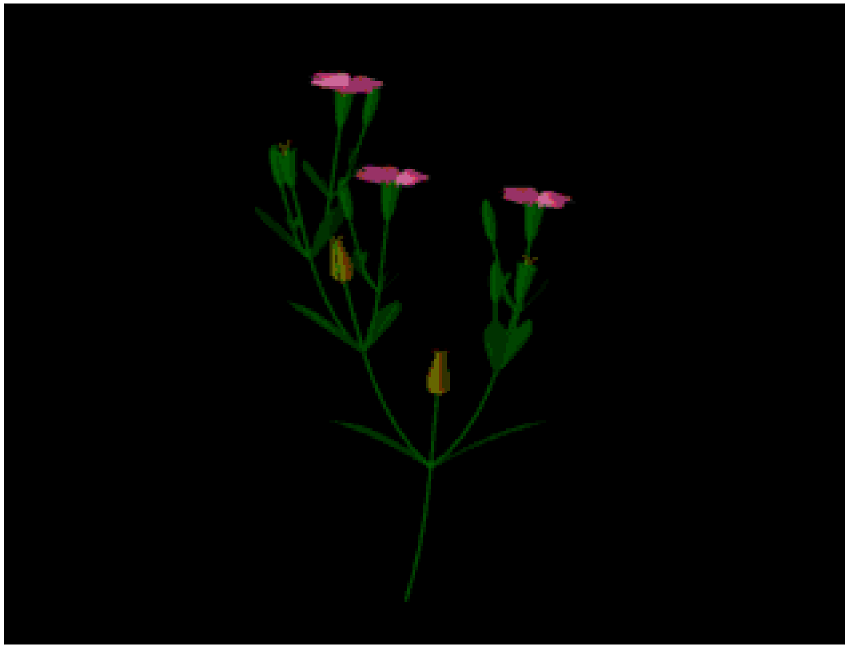


Prusinkiewicz, Hammel, Mech http://www.cpsc.ucalgary.ca/projects/bmv/vmm/title.html

# Flowers at the apex



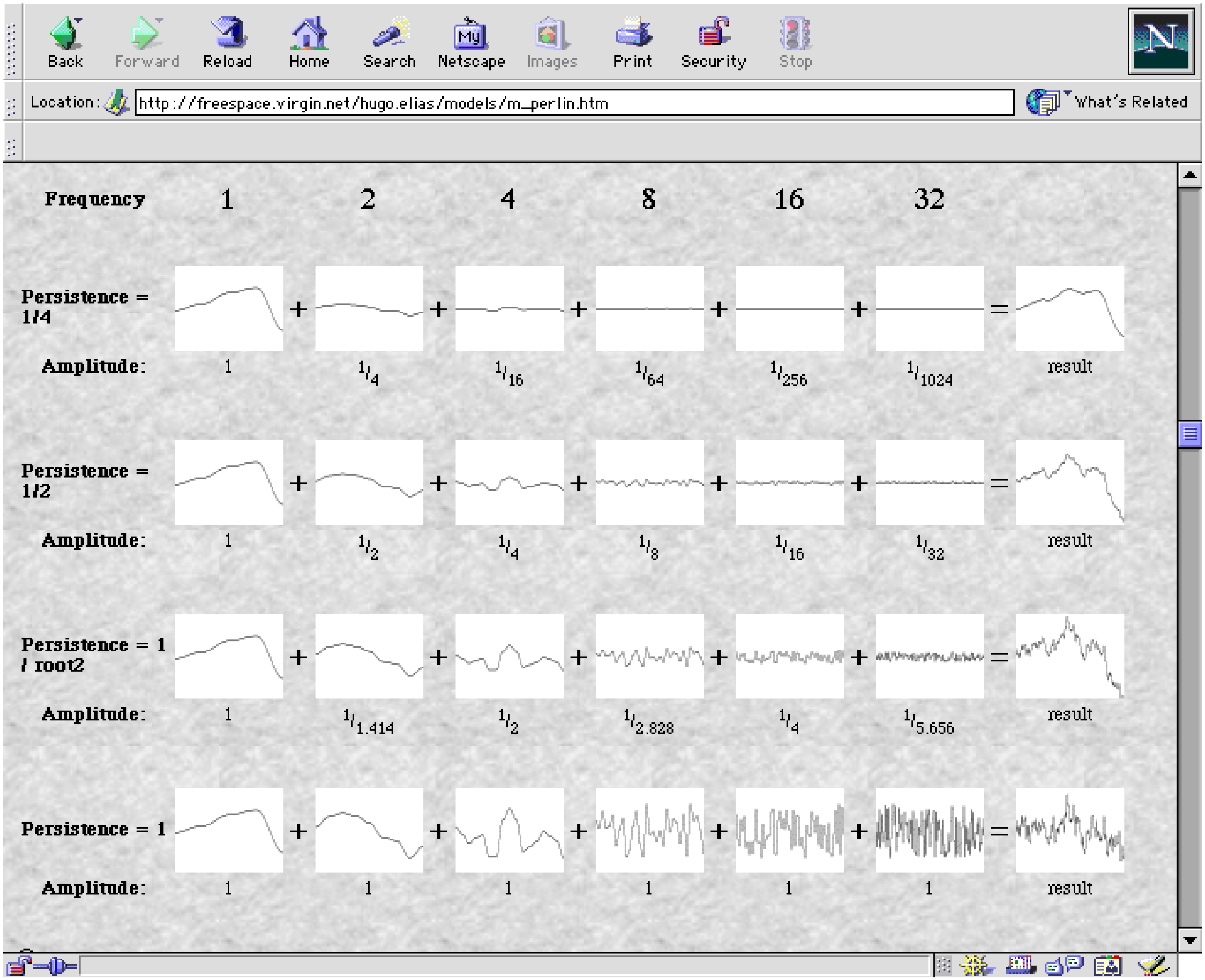

Prusinkiewicz, Hammel, Mech http://www.cpsc.ucalgary.ca/projects/bmv/vmm/title.html

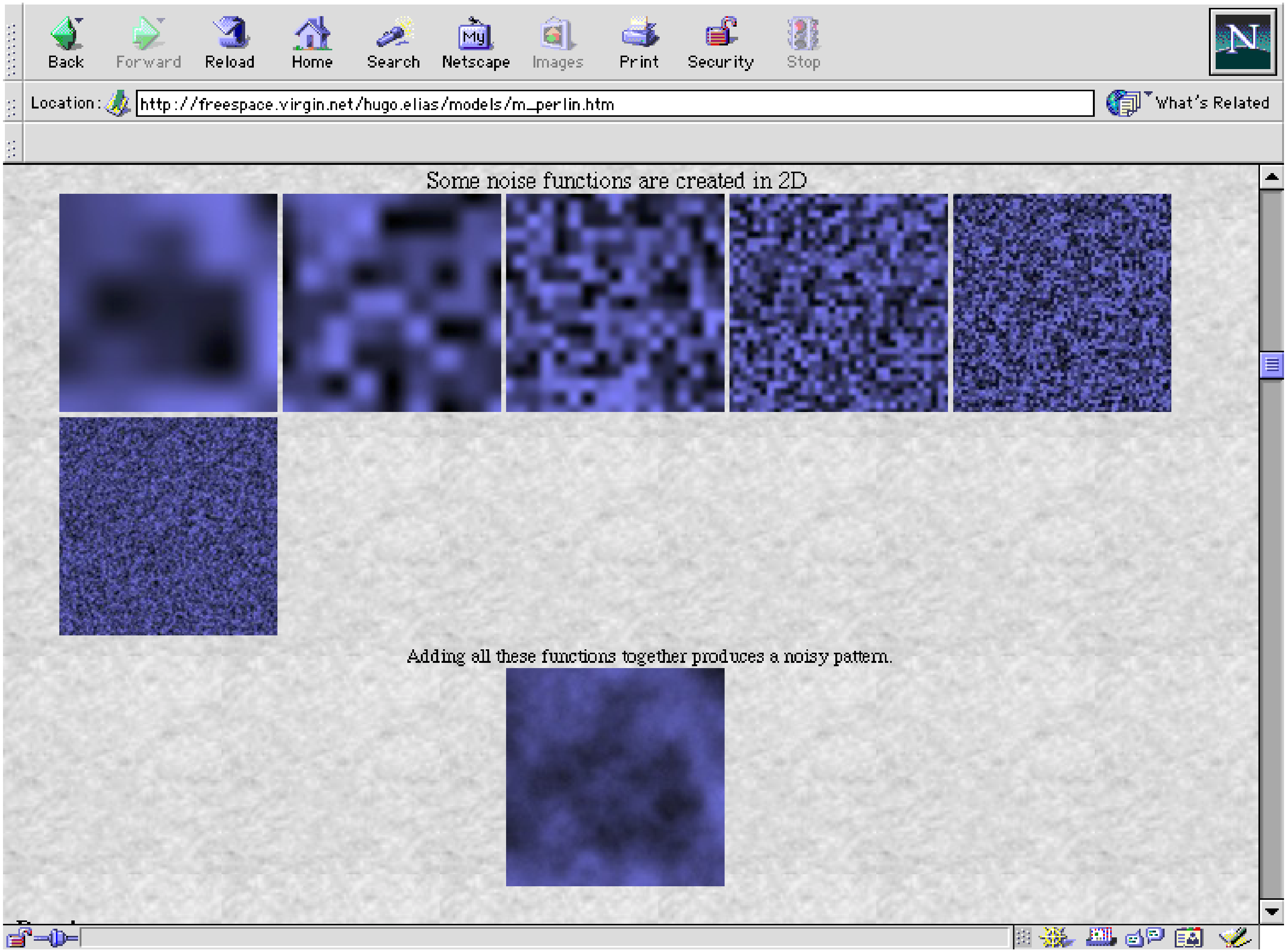


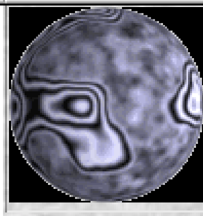
Prusinkiewicz, Hammel, Mech http://www.cpsc.ucalgary.ca/projects/bmv/vmm/title.html

# Turbulence/Perlin noise

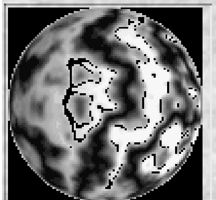
- Many natural textures look like noise or “smoothed” noise (marble, flames, clouds, terrain, etc.)
- Issue:
  - obtain the right kind of smoothing
- Strategy:
  - construct noise functions at a variety of scales
    - do this by drawing samples from a random number generator at different spacings
  - form a weighted sum
- Usually:
  - space the noise in octaves (i.e. interelement spacing goes as $(1/2)^i$ - this means frequency goes as $2^i$)
  - amplitude in sum goes as $p^i$, where p is a parameter called persistence.
  - persistence is commonly $2^i$
- 3D Turbulence yields animations for clouds, fog, flames


Back
Forward
Reload
Home
Search
Netscape
Images
Print
Security
Stop
Location:
http://freespace.virgin.net/hugo.elias/models/m_perlin.htm
What's Related
Frequency
1
2
4
8
16
32
Persistence = 1/4
Amplitude:
1
1/4
1/16
1/64
1/256
1/1024
result
Persistence = 1/2
Amplitude:
1
1/2
1/4
1/8
1/16
1/32
result
Persistence = 1 / root2
Amplitude:
1
1/1.414
1/2
1/2.828
1/4
1/5.656
result
Persistence = 1
Amplitude:
1
1
1
1
1
1
result


Back
Forward
Reload
Home
Search
Netscape
Images
Print
Security
Stop
Location: http://freespace.virgin.net/hugo.elias/models/m_perlin.htm
What's Related
Some noise functions are created in 2D
Adding all these functions together produces a noisy pattern.

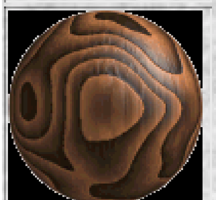


To create more interesting and complicated textures, you should try mixing several Perlin functions. This texture was created in two parts. Firstly a Perlin function with low persistence was used to define the shape of the blobs. The value of this function was used to select from two other functions, one of which defined the stripes, the other defined the blotchy pattern. A high value chose more of the former, a low value more of the latter. The stripes were defined by multiplying the first Perlin Function by some number (about 20) then taking the cosine.



A marbly texture can be made by using a Perlin function as an offset to a cosine function.

```
texture = cosine( x + perlin(x,y,z) )
```



Very nice wood textures can be defined. The grain is defined with a low persistence function like this:

```
g = perlin(x,y,z) * 20
grain = g - int(g)
```

The very fine bumps you can see on the wood are high frequency noise that has been stretched in one dimension.

```
bumps = perlin(x*50, y*50, z*20)
if bumps < .5 then bumps = 0  else bumps = 1t
```

## References

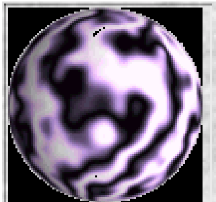
**Procedural textures:** http://developer.intel.com/drg/mmx/appnotes/proctex.htm
Intel Developer Site article about using the new MMX technology to render Perlin Noise in real time.

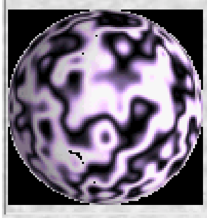
**Ken Perlin's Homepage:** http://mrl.nyu.edu/perlin/
I assume the person responsable for Perlin Noise. He has an interesting page with lots of useful links to texturing and modeling

Back Forward Reload Home Search Netscape Images Print Security Stop

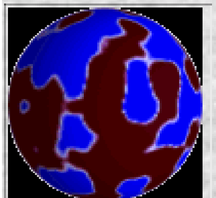
Location: http://freespace.virgin.net/hugo.elias/models/m_perlin.htm What's Related

The following textures were made with 3D Perlin Noise

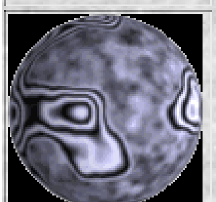




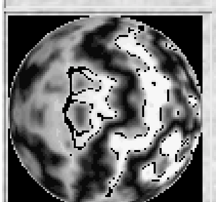
Standard 3 dimensional perlin noise. 4 octaves, persistence 0.25 and 0.5



Low persistence. You can create harder edges to the perlin noise by applying a function to the output.



To create more interesting and complicated textures, you should try mixing several Perlin functions. This texture was created in two parts. Firstly a Perlin function with low persistence was used to define the shape of the blobs. The value of this function was used to select from two other functions, one of which defined the stripes, the other defined the blotchy pattern. A high value chose more of the former, a low value more of the latter. The stripes were defined by multiplying the first Perlin Function by some number (about 20) then taking the cosine.



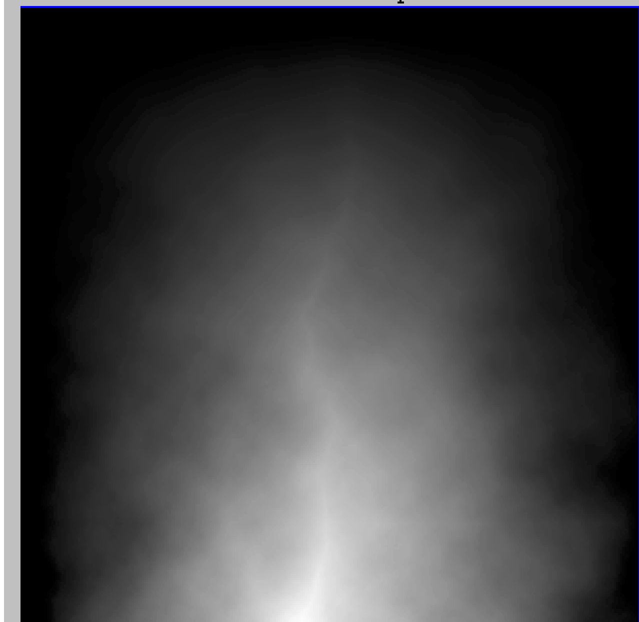
A marbly texture can be made by using a Perlin function as an offset to a cosine function.

```
texture = cosine( x + perlin(x,y,z) )
```

Back Forward Reload Home Search Netscape Images Print Security Stop

Location: http://www.cs.wpi.edu/~matt/courses/cs563/talks/noise/noise.html What's Related

*Flames:* Compute intensity of point based on distance from center in x. Scale it based on distance in y. Add turbulence. Use 3-D turbulence to animate. Here is an example of a flame with added turbulence.



**Conclusions**

- Lots of interesting effects can be gained by adding turbulence
- Need to play with degree and scale to get most realistic images
- Ties together a lot of topics in graphics (fractals, texture, color, curves)