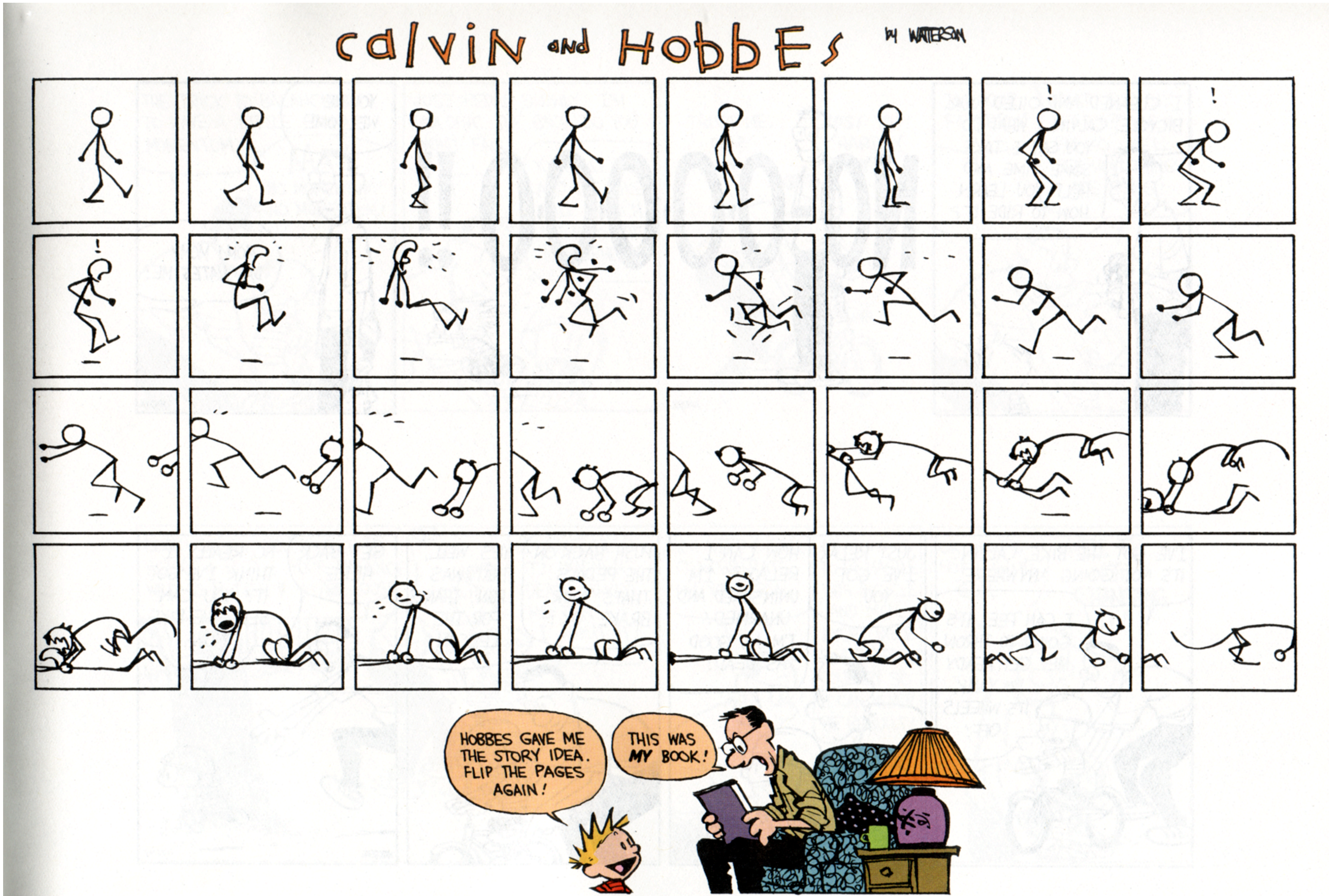
Animation

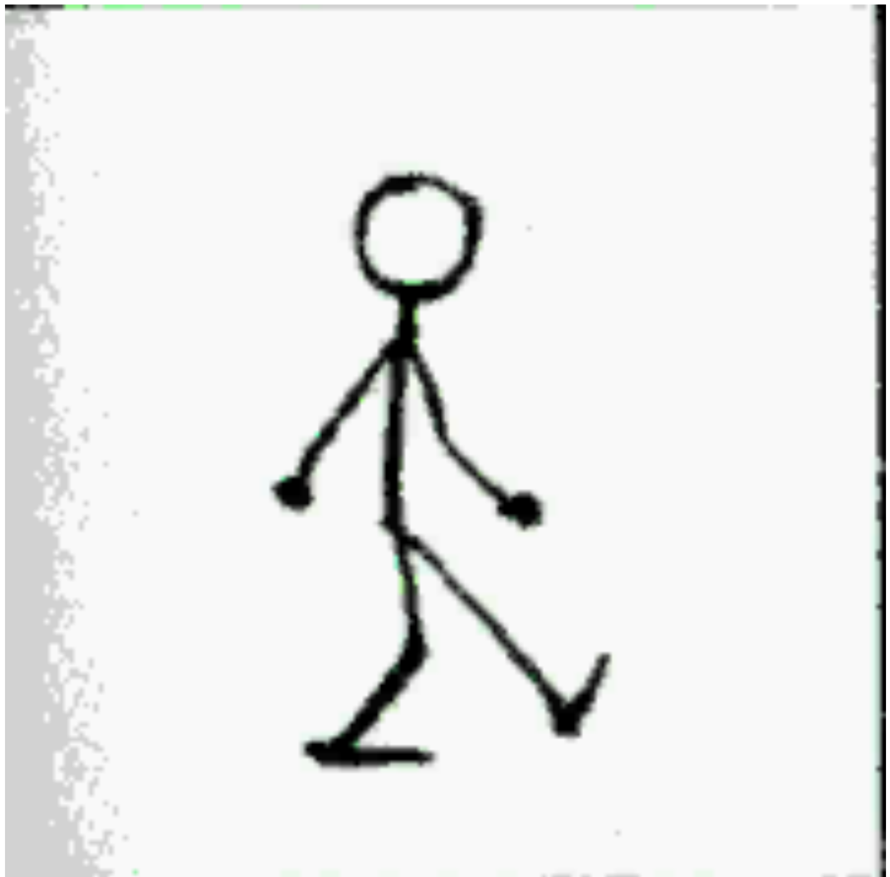
- Persistence of vision:
  - The visual system smoothes in time. This means that images presented to the eye are perceived by the visual system for a short time after they are presented. In turn, this means that if images are shown at the right rate (about 20-30 Hz will do it), the next image replaces 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

Keyframing



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

Result



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 - issues

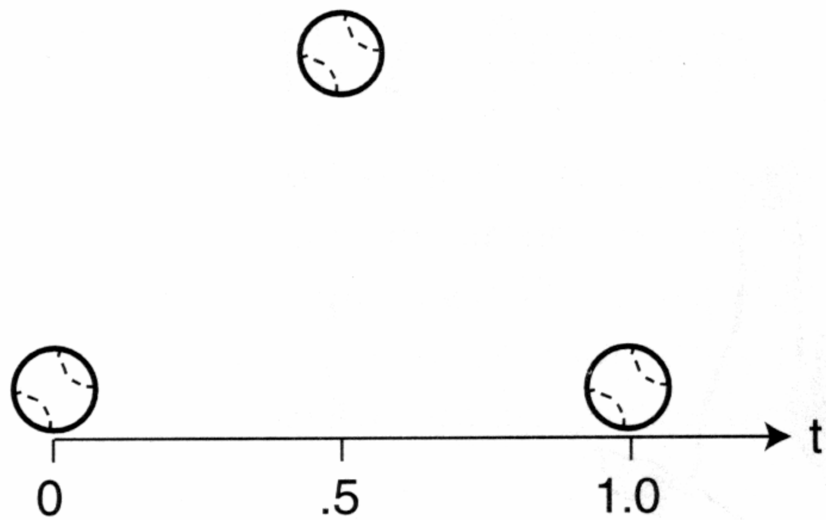


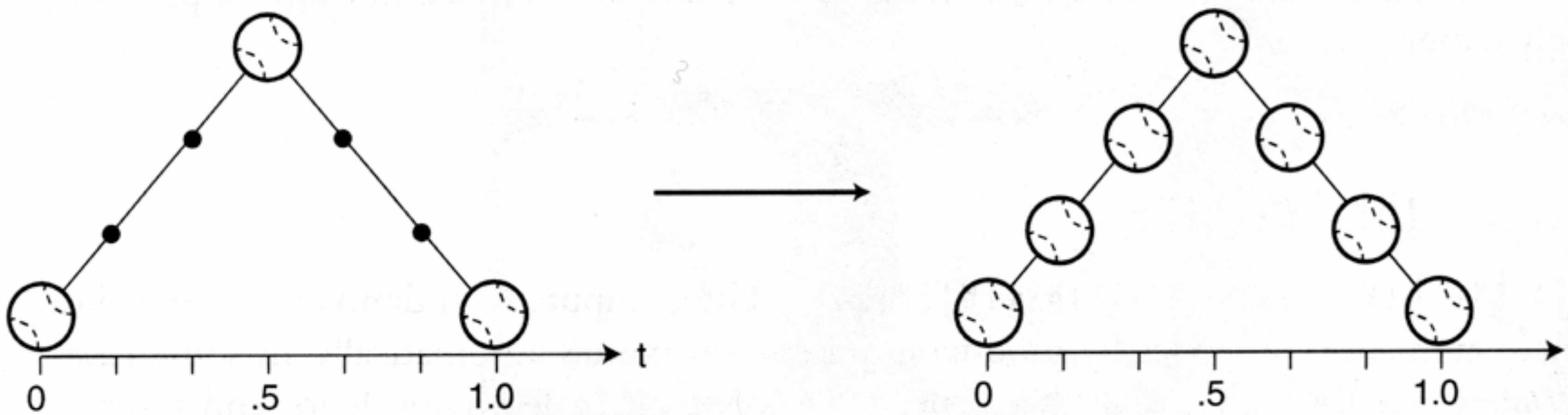
Figure 10.4 **Three keyframes.** Three keyframes representing a ball on the ground, at its highest point, and back on the ground.

- 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

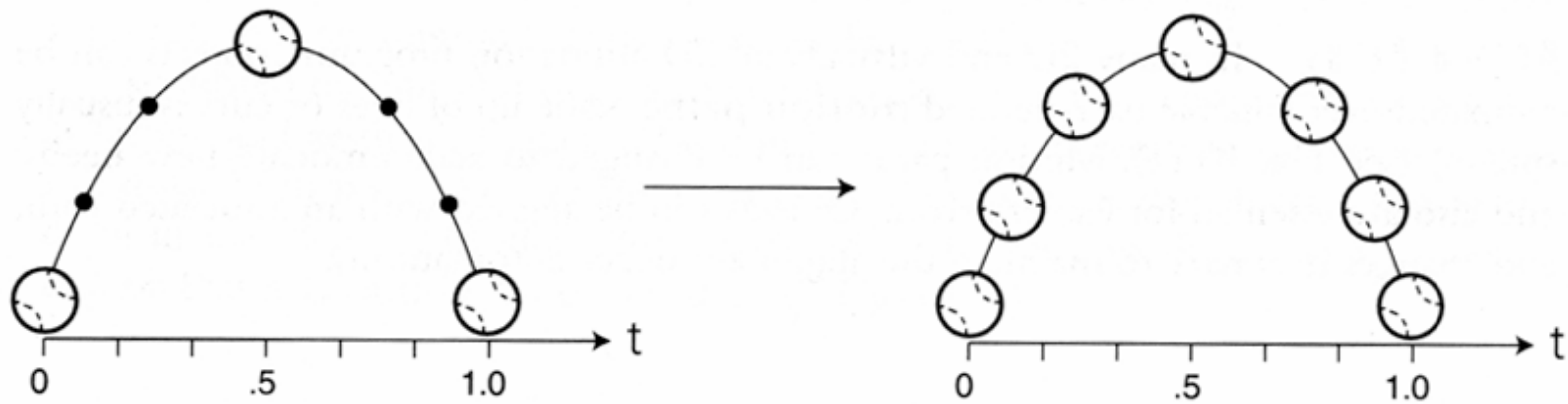


Figure 10.9 **Inbetweening with nonlinear interpolation.** Nonlinear interpolation can create equally spaced inbetween frames along curved paths. The ball still moves at a constant speed. (Note that the three keyframes used here and in Fig. 10.10 are the same as in Fig. 10.4.)

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

Modify the parameter, too

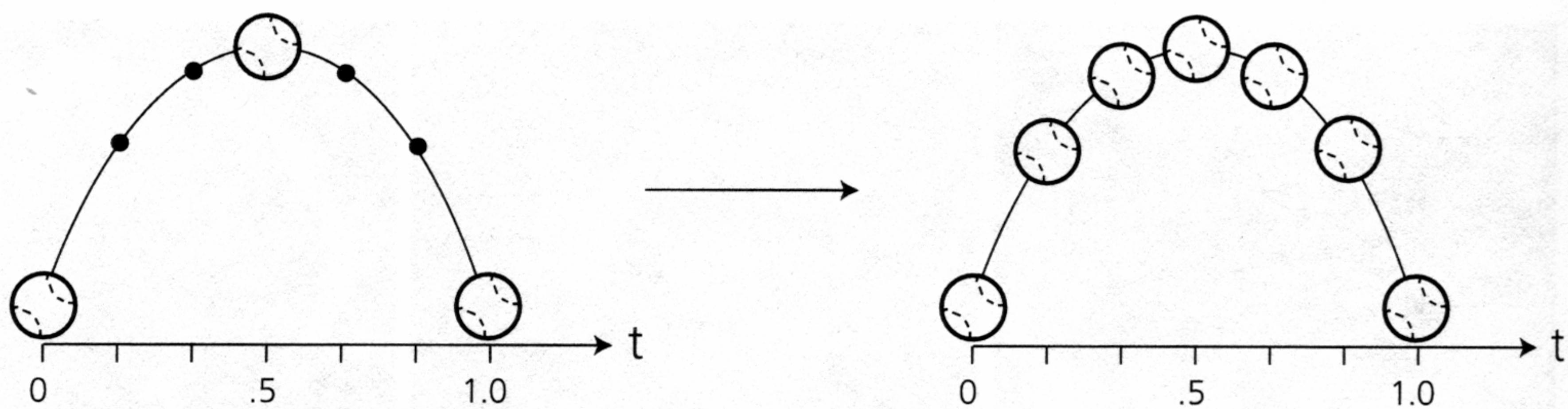
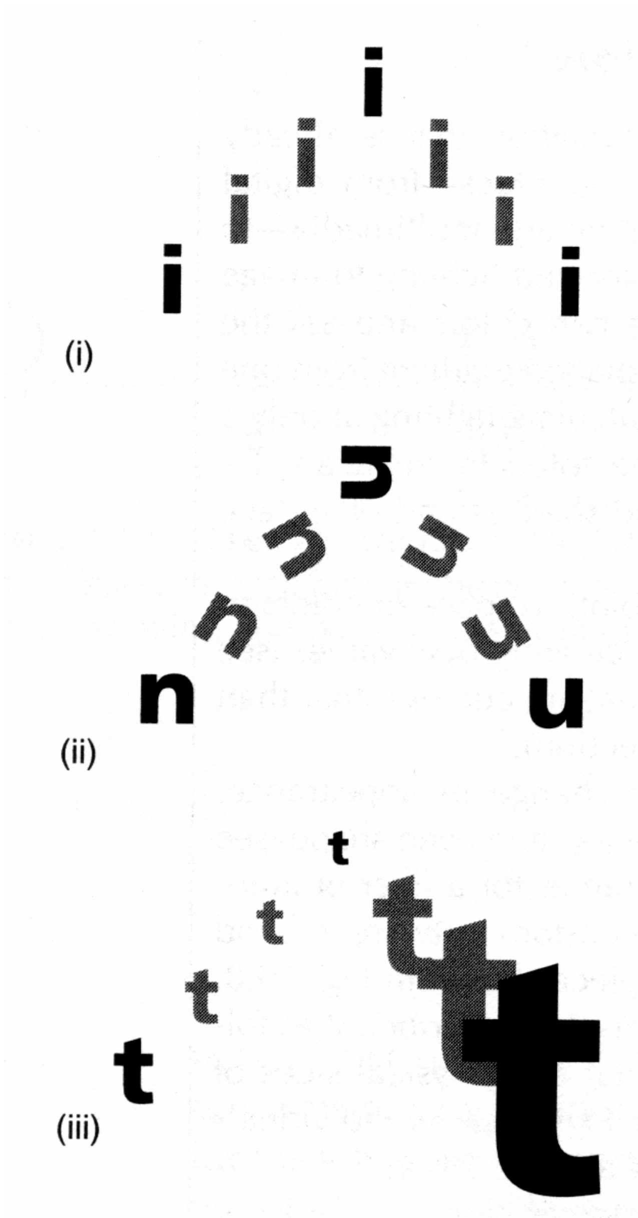


Figure 10.10 Inbetweening with nonlinear interpolation and easing. The ball changes speed as it approaches and leaves keyframes, so the dots indicating calculations made at equal time intervals are no longer equidistant along the path.

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

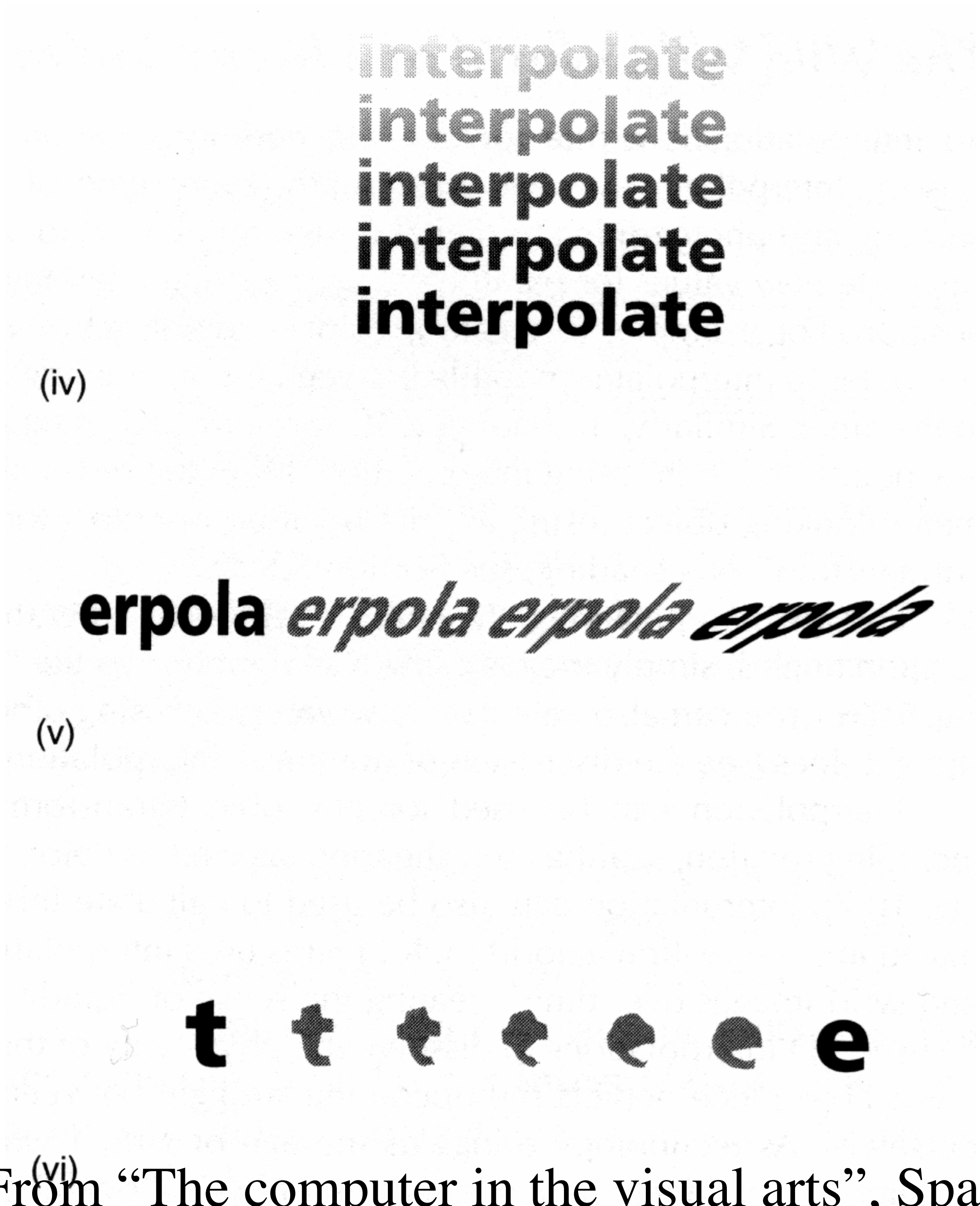


Position

Position and orientation

Position and scale

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

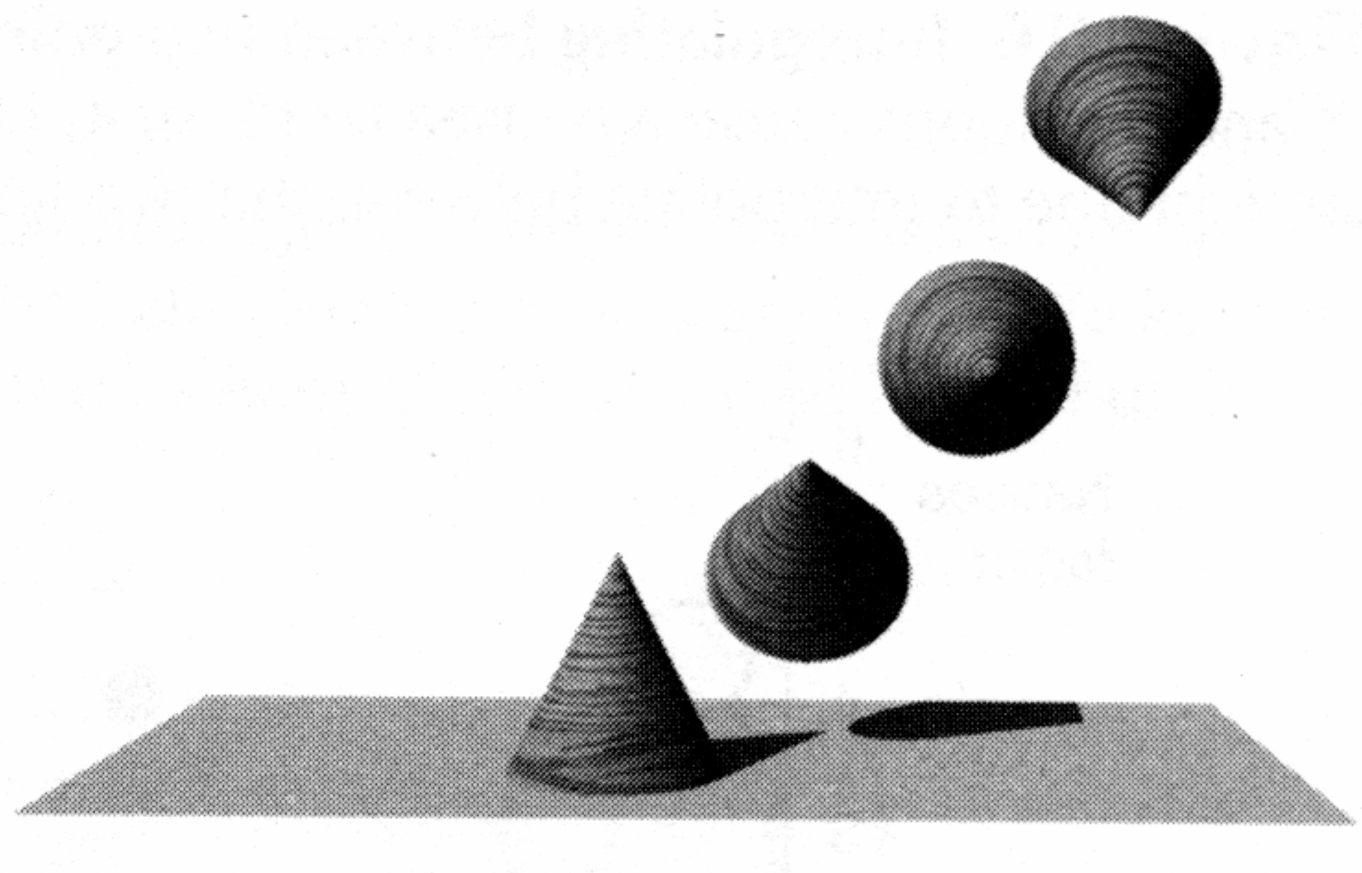


Grey-level

Shear

Shape

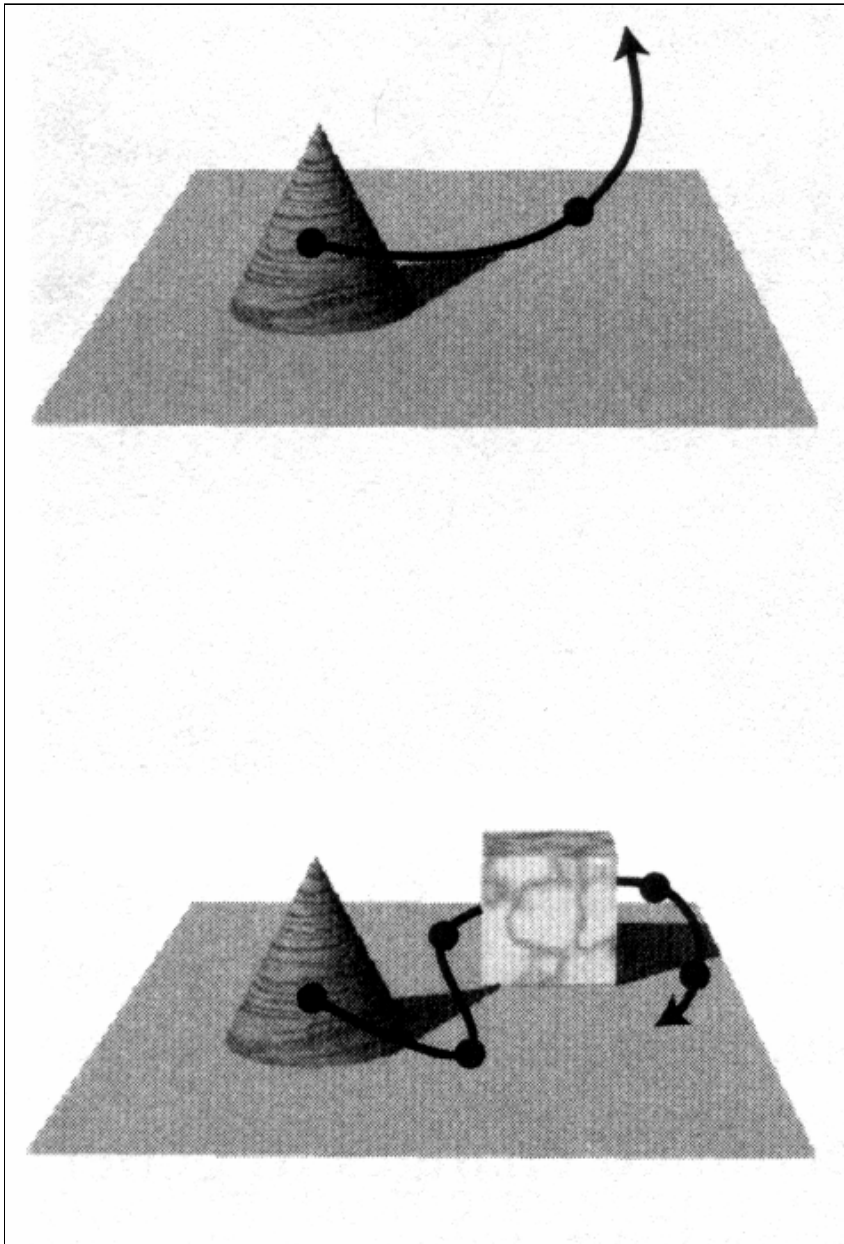
From “The computer in the visual arts”, Spalter, 1999



Position and orientation:

note that the position travels along a motion path

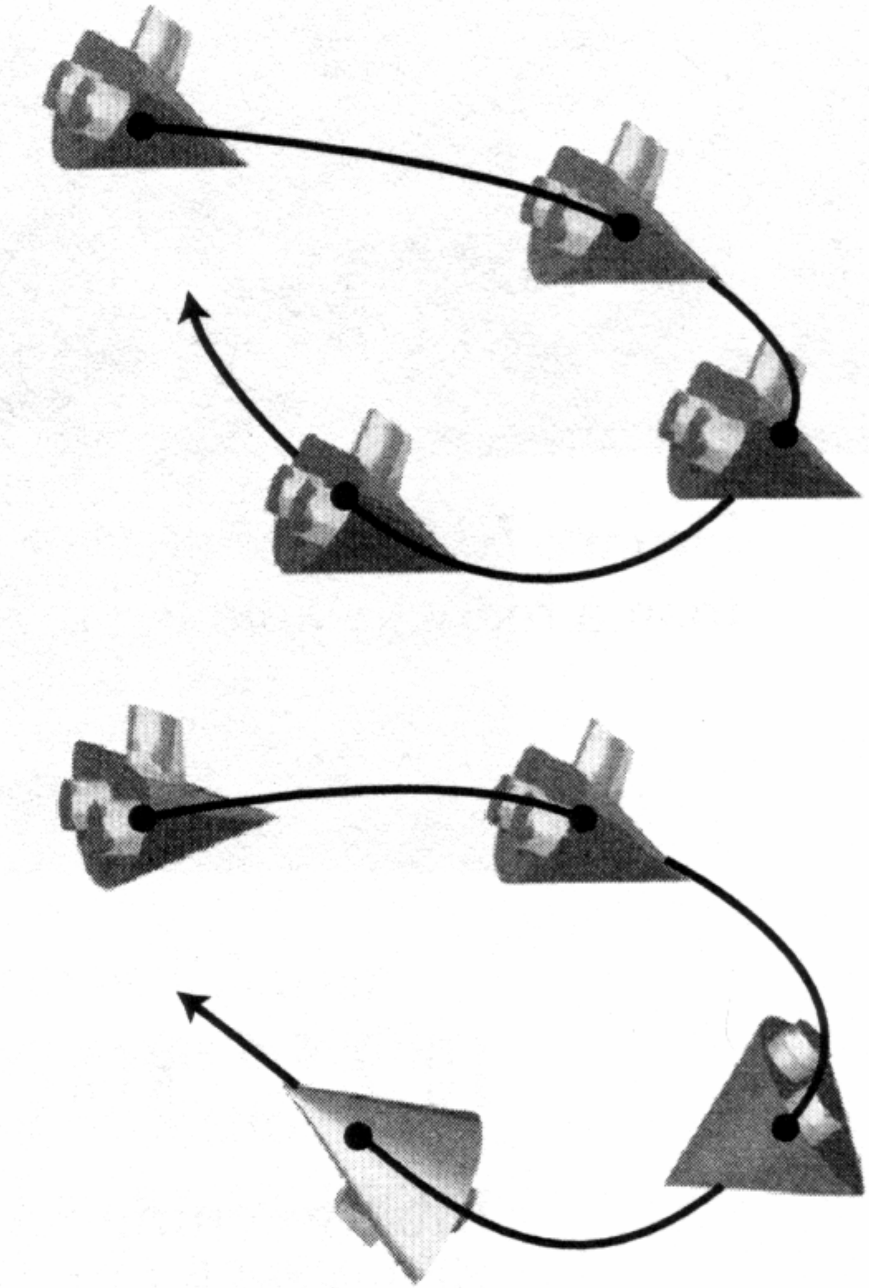
From “The computer in the visual arts”, Spalter, 1999



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

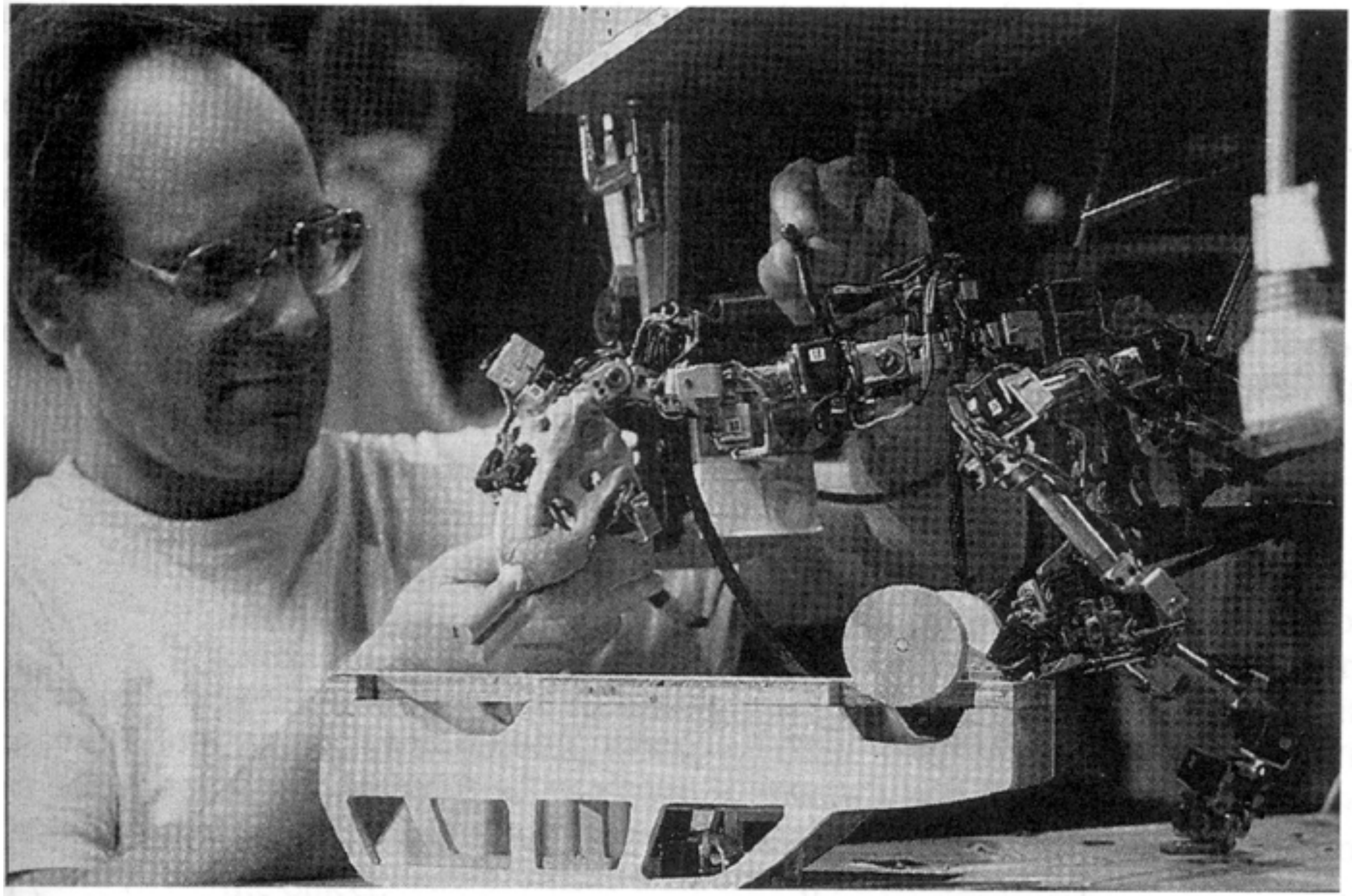


Interpolating orientation gives greater realism. Notice that the tangent to the motion path gives a great cue to the orientation of the object.

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

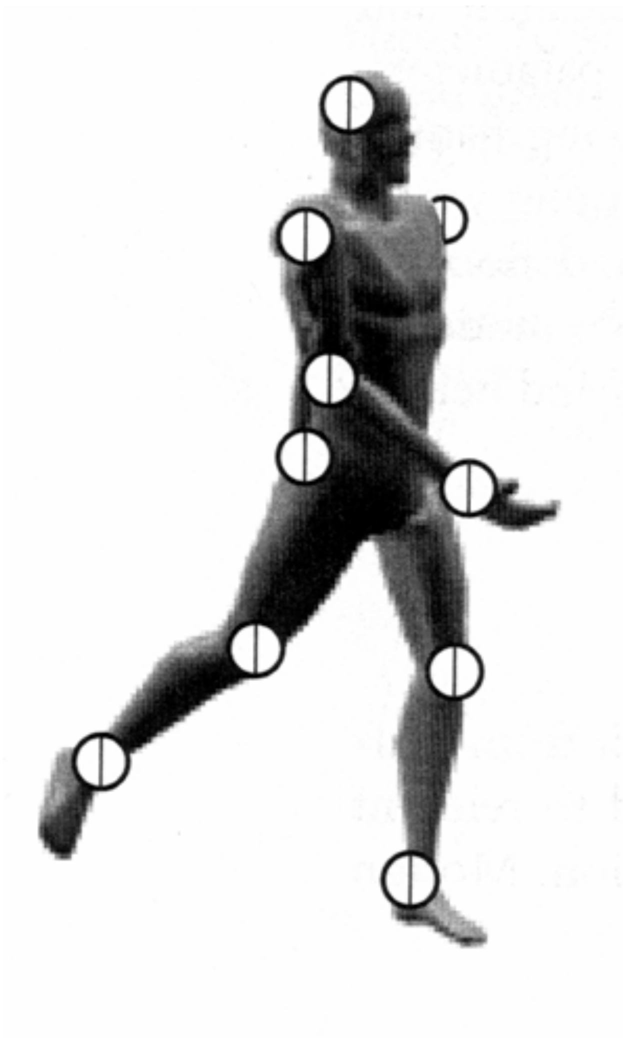
Stop motion

- 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 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?



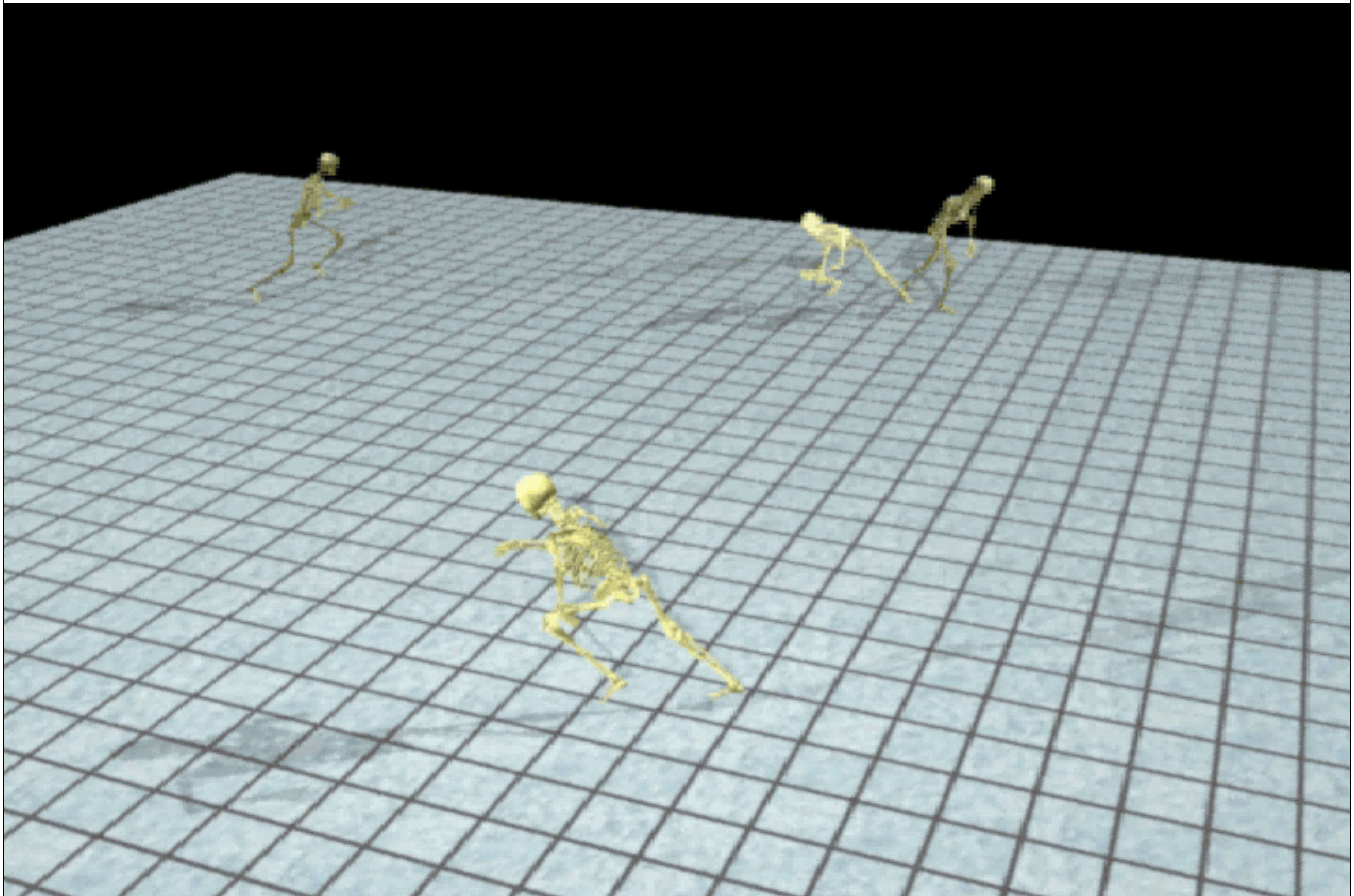
From “The computer Image”,Watt and Policarpo, 1998

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

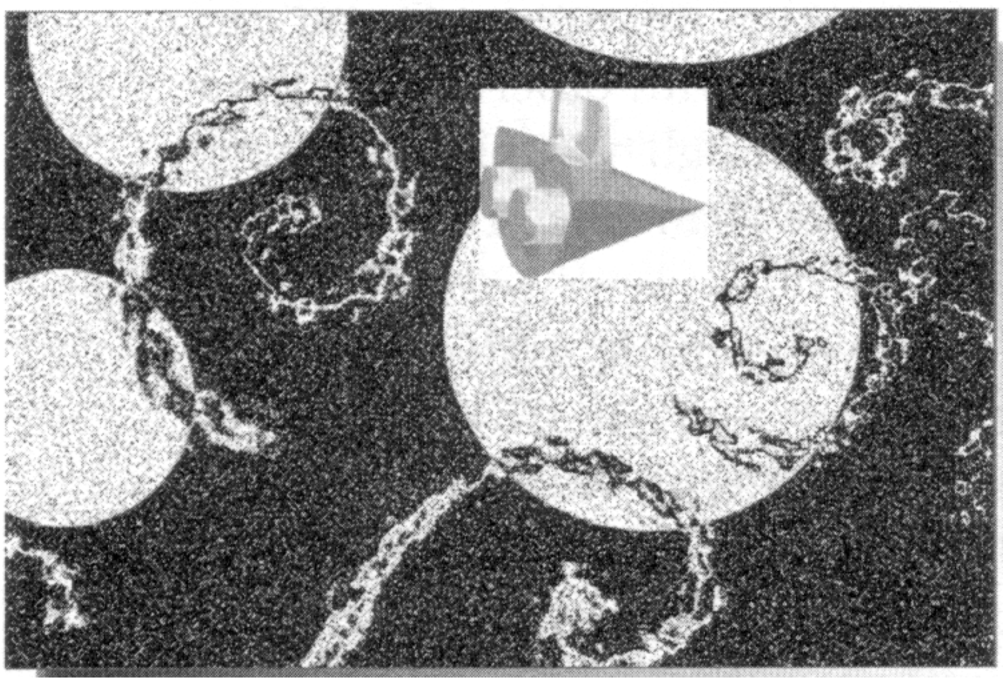
MotionAnalysis / Performance Capture Studios



[redacted]

Compositing

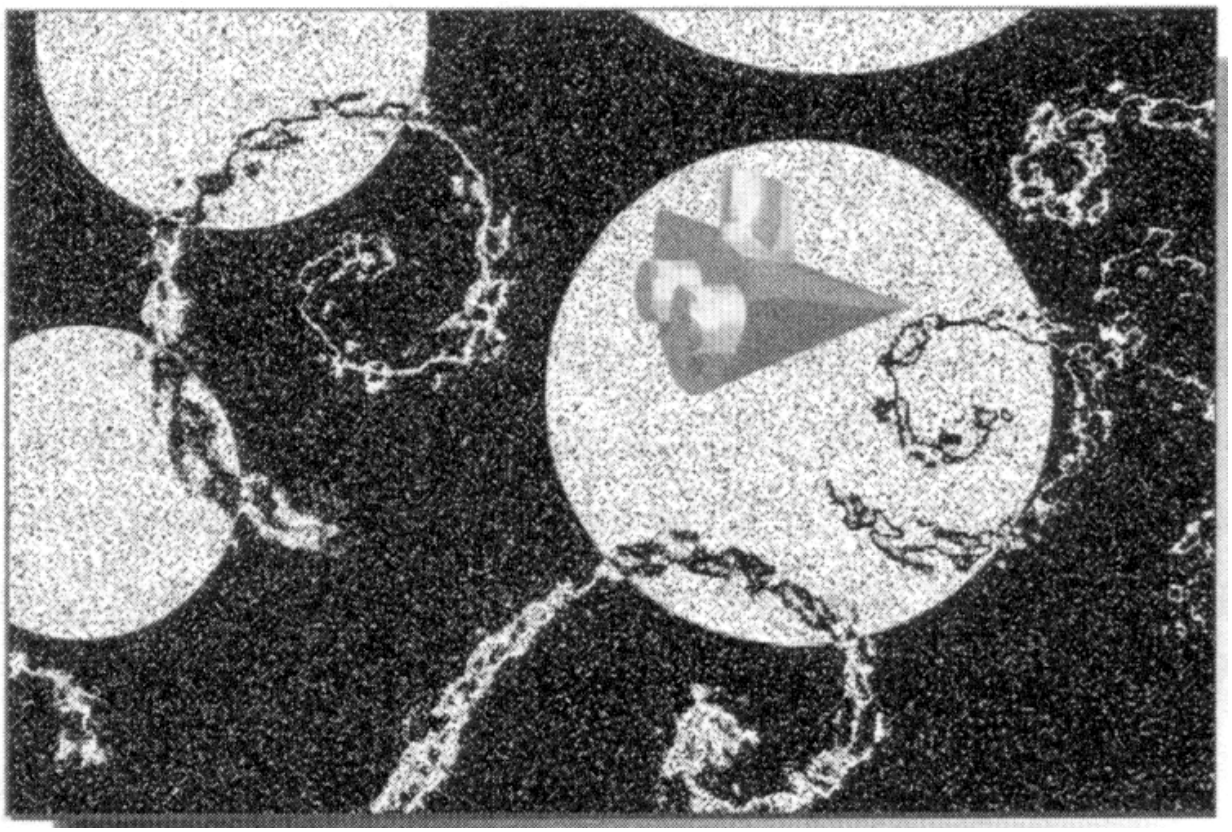
- Overlay one image/film on another
  – variety of types of overlay



Simple overlay - spaceship pixels replace background pixels

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

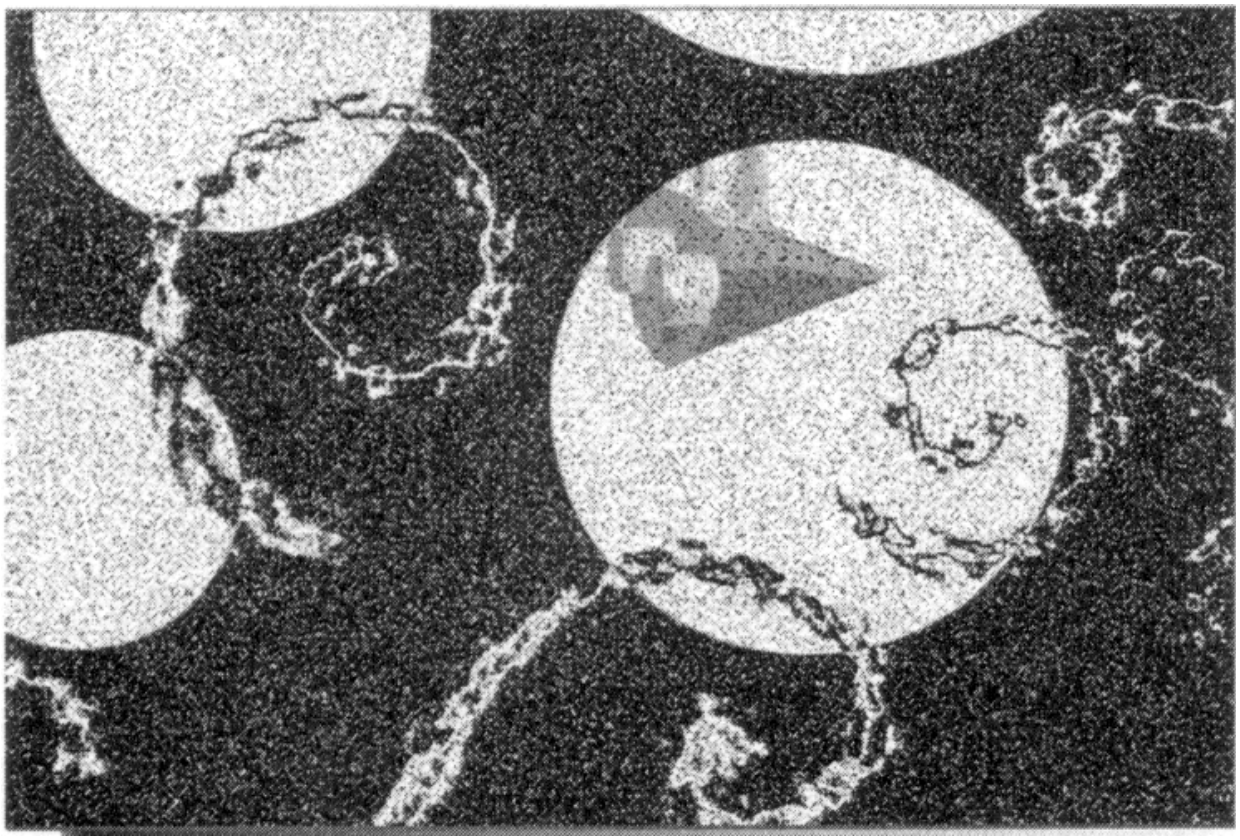
Compositing



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

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

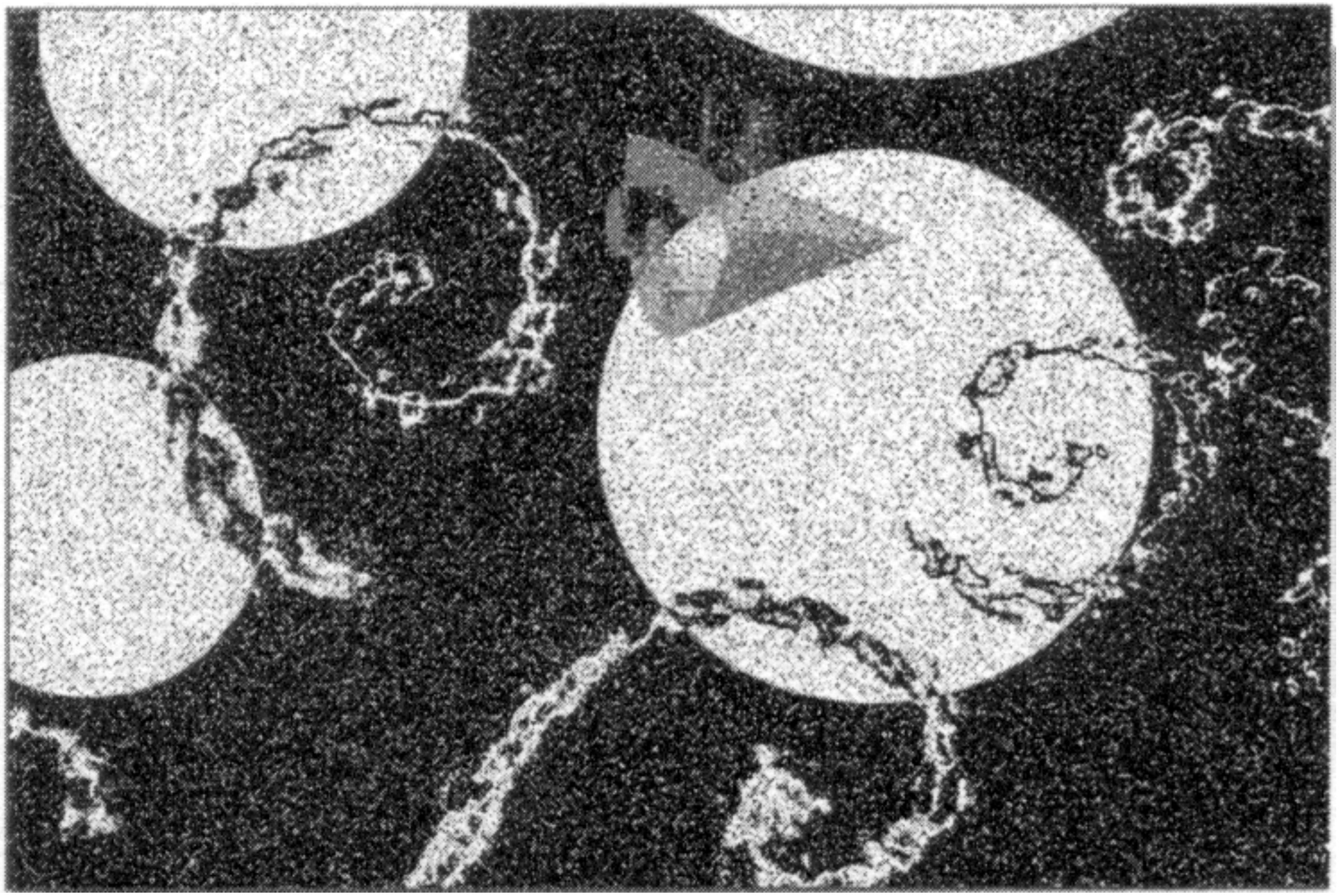
Compositing



Spaceship pixels replace background pixels if they are darker

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

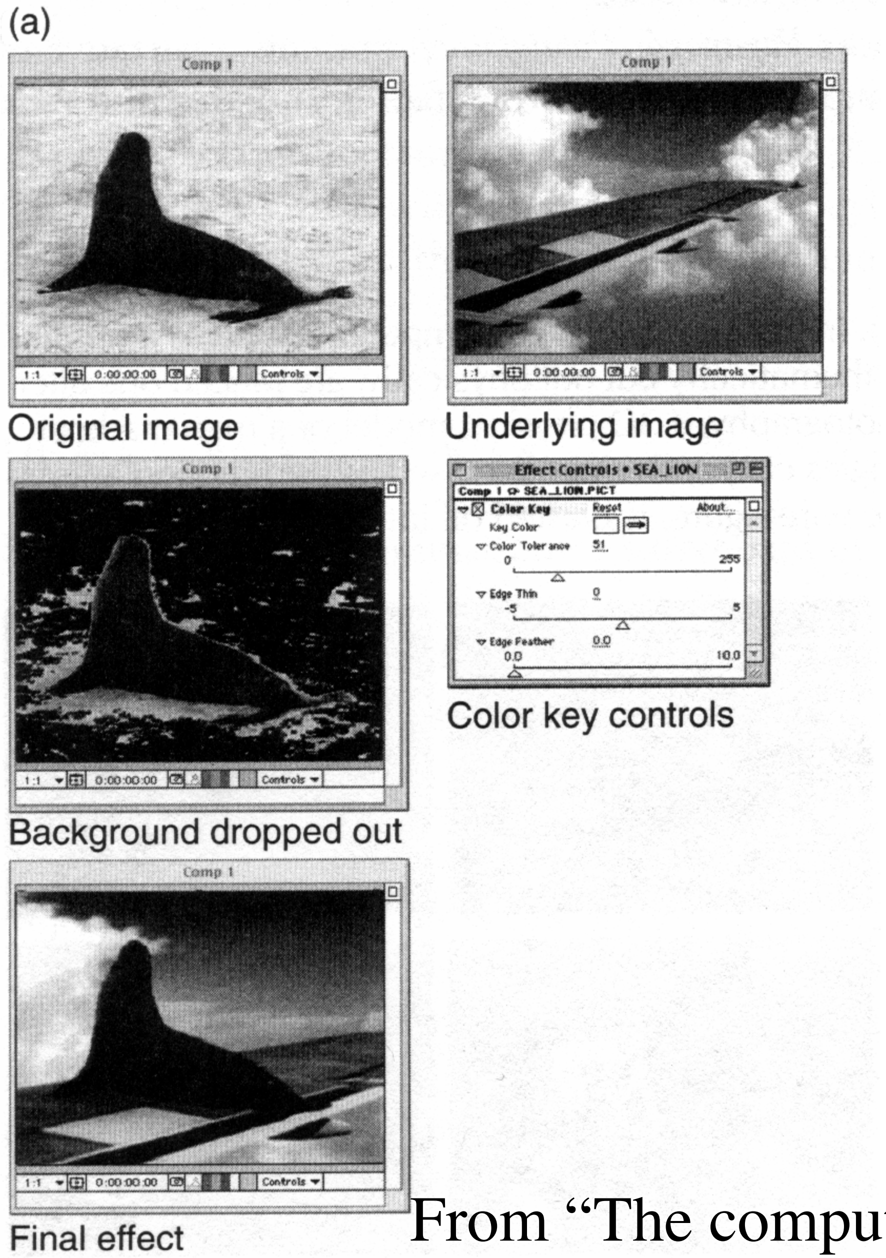
Compositing



Light areas are more transparent - blending

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

Compositing

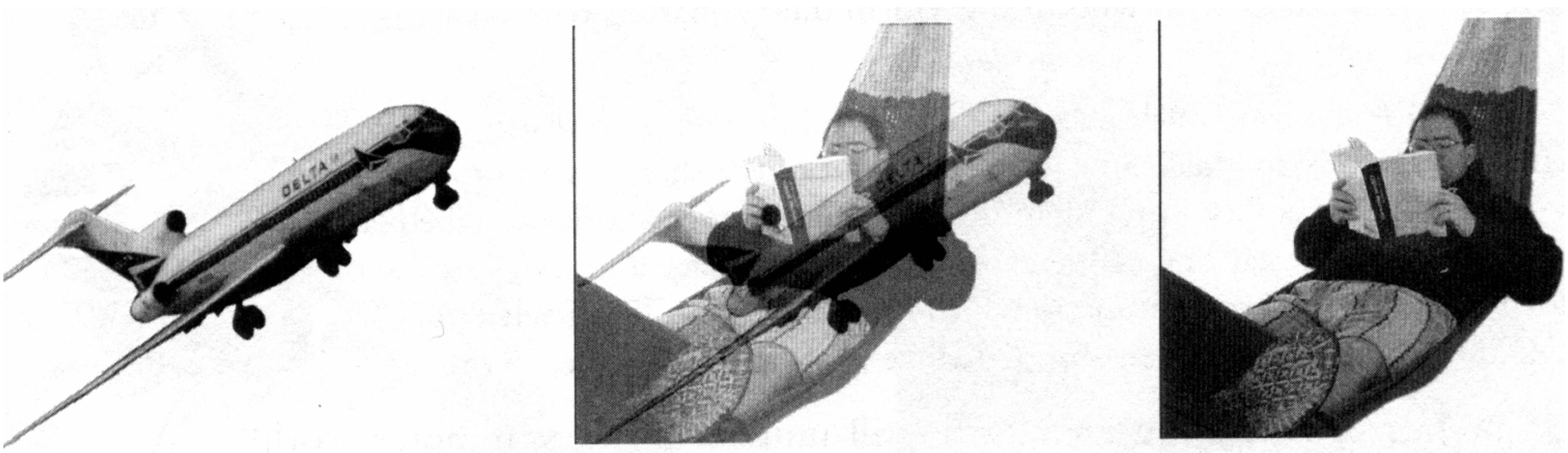


- Note that human intervention might be required to remove odd pixels, if the background doesn’t have a distinctive colour
- One can buy sets of images which have been segmented by hand.

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

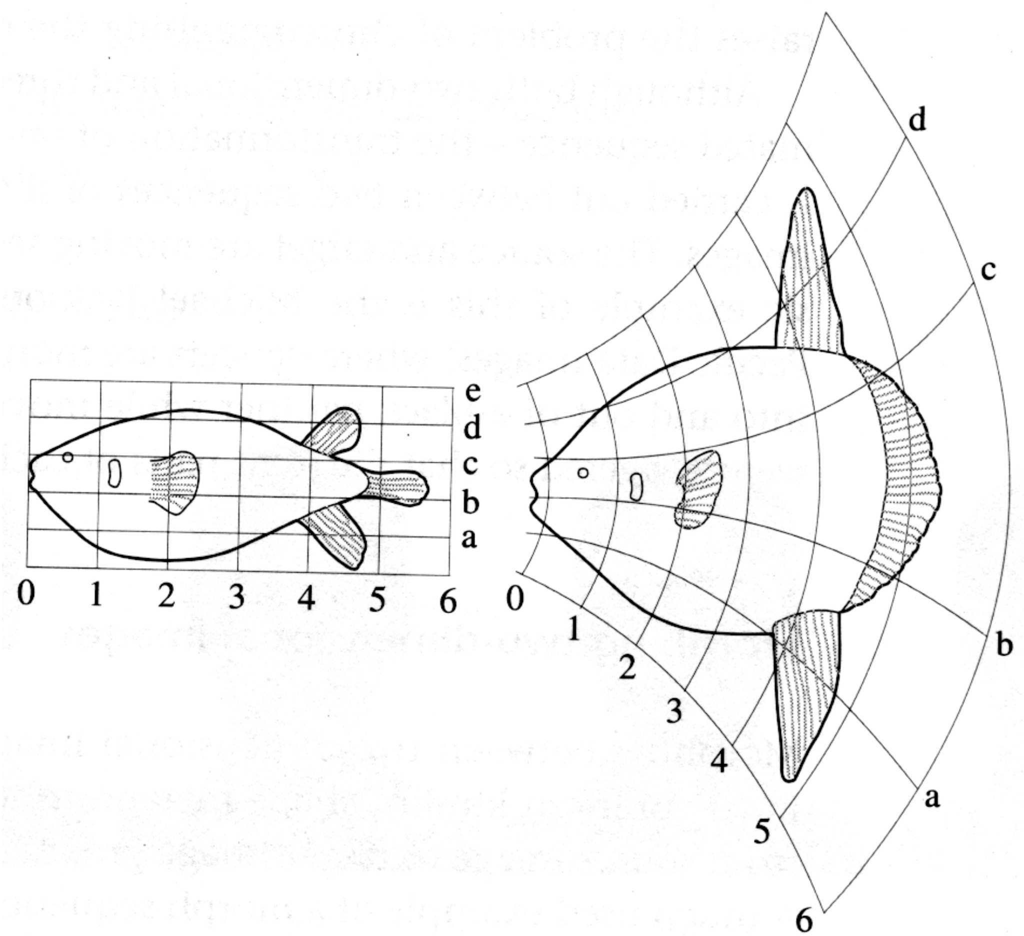
Morphing

- 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

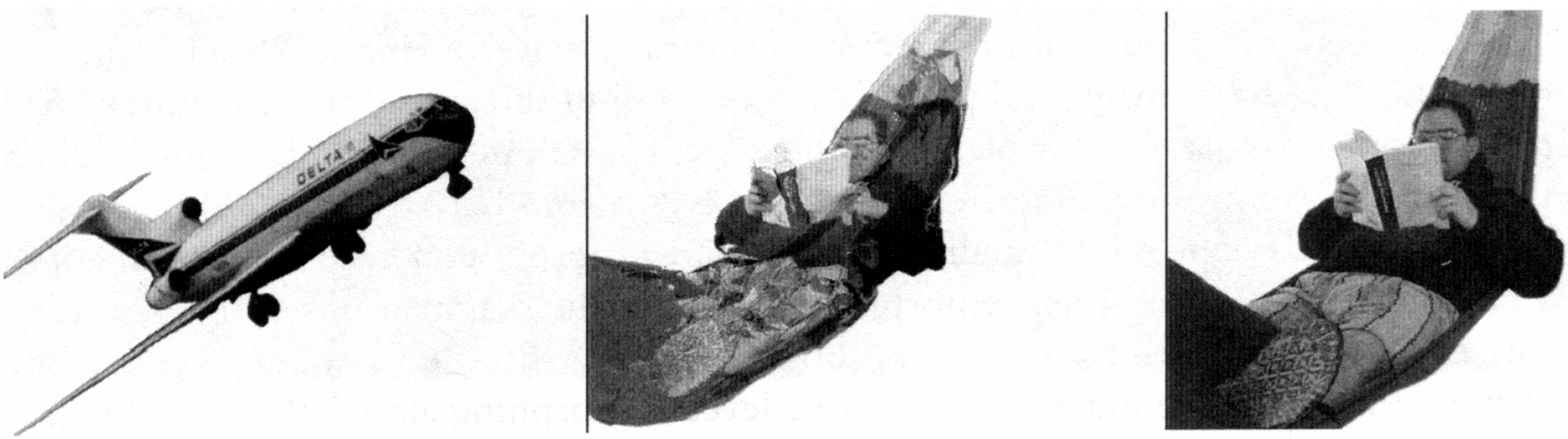
Morphing



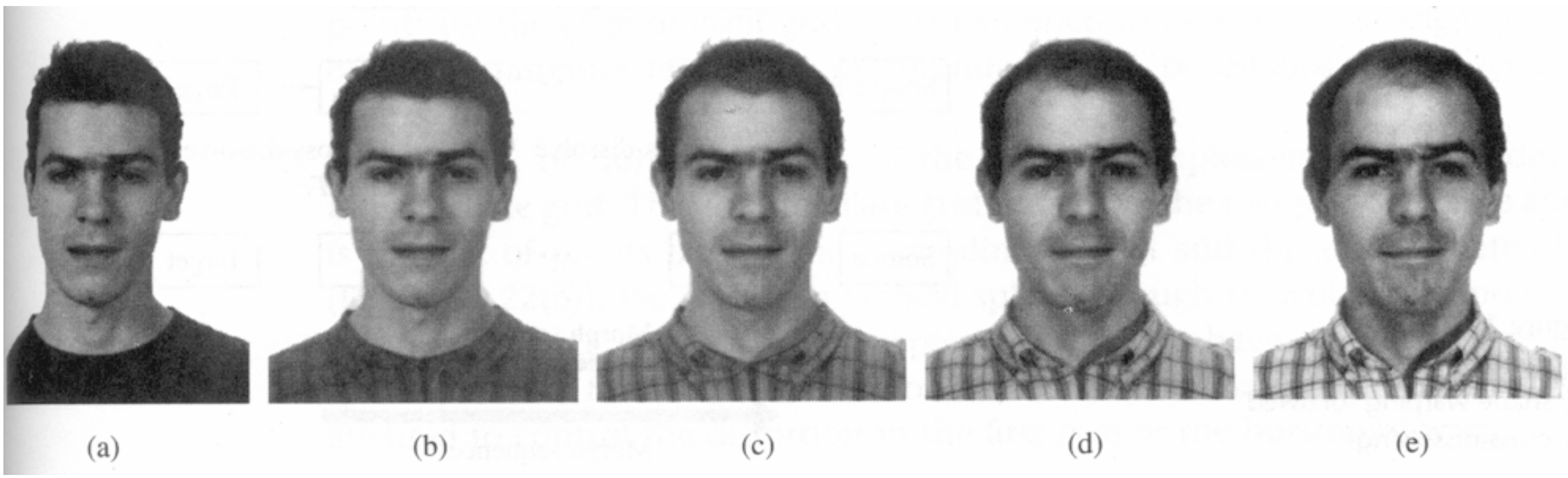
From "On growth and Form", D'Arcy Thompson

Morphing

- Another use for the deformation encoding shown earlier
- From “The computer Image”,
- Watt and Policarpo, 1998



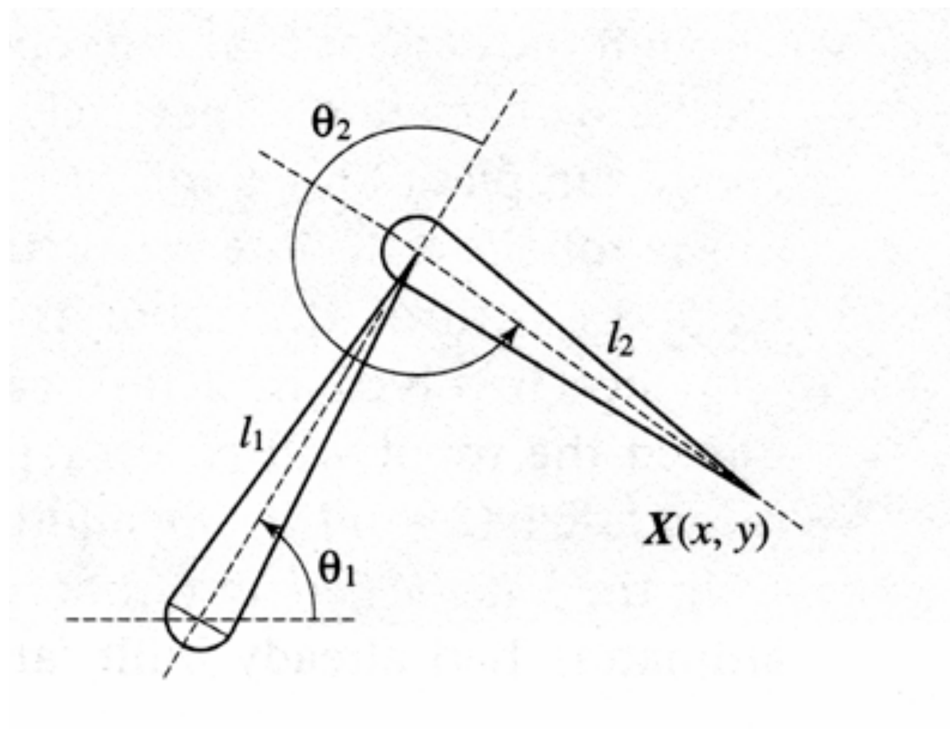
Morphing



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

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?

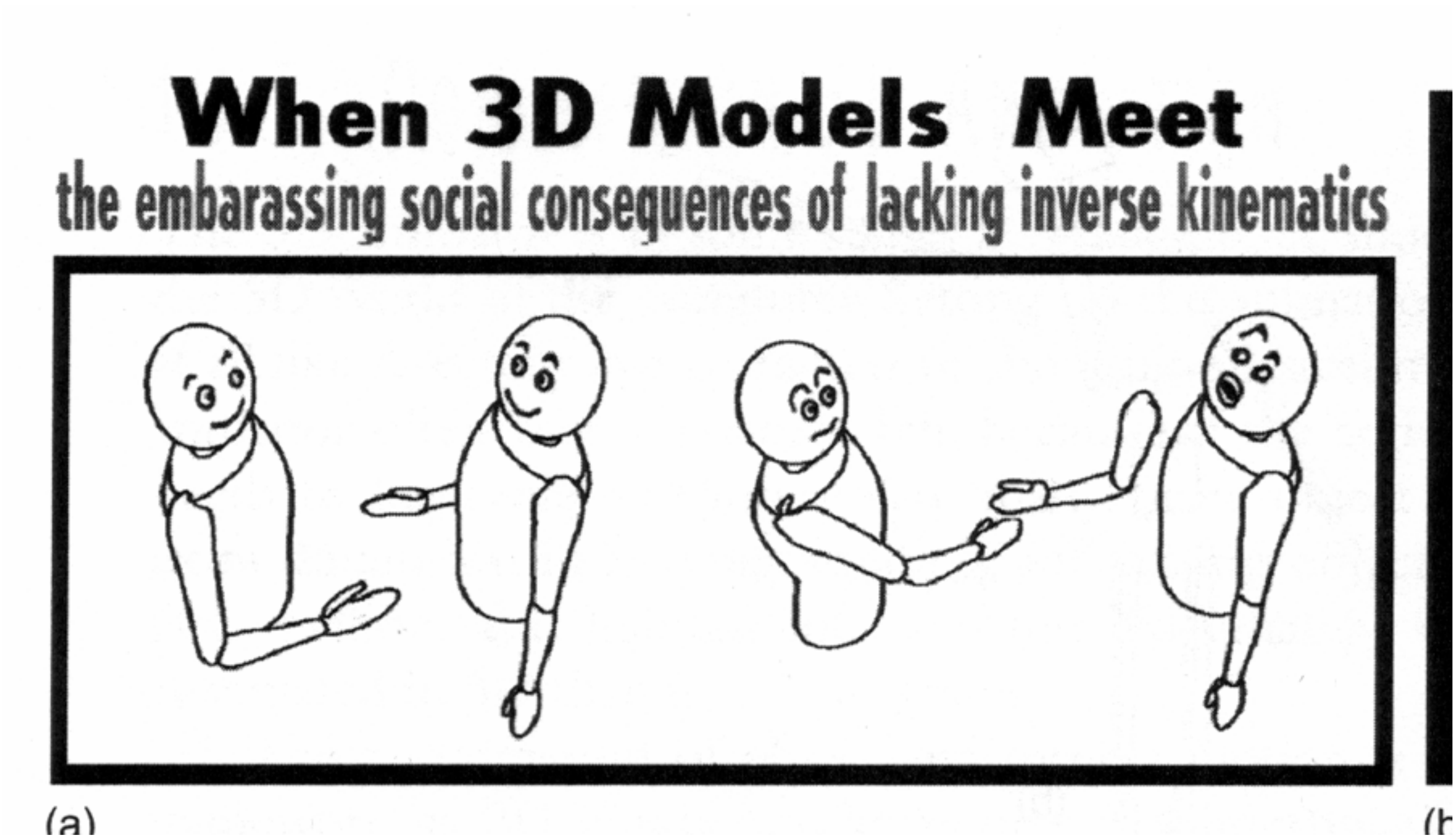


From "The computer Image",

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?

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

- 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.

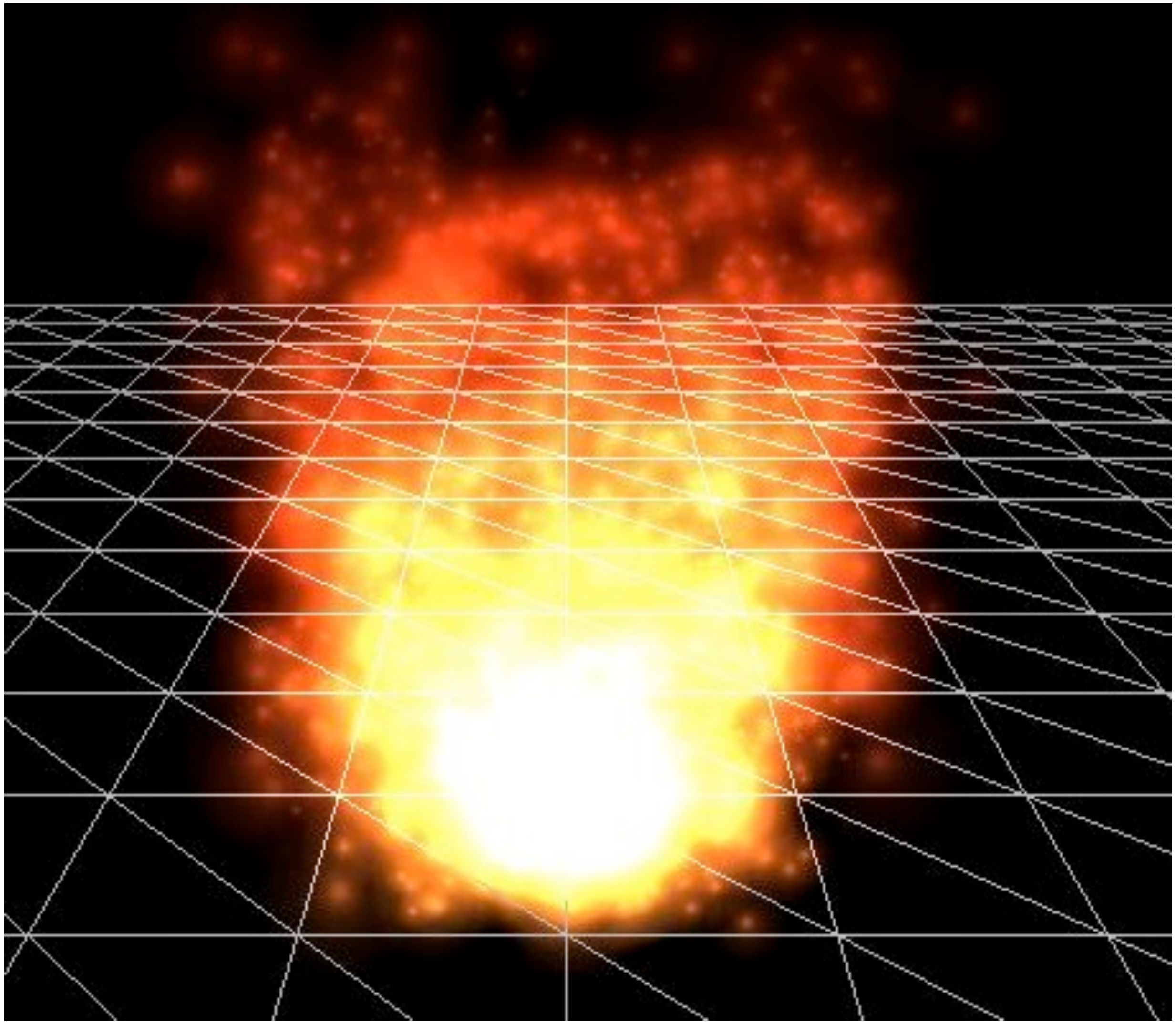
Procedural Dynamics - Particle systems

- There is a source of particles
  - move under gravity, sometimes collisions
  - sometimes other reactions
- 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.

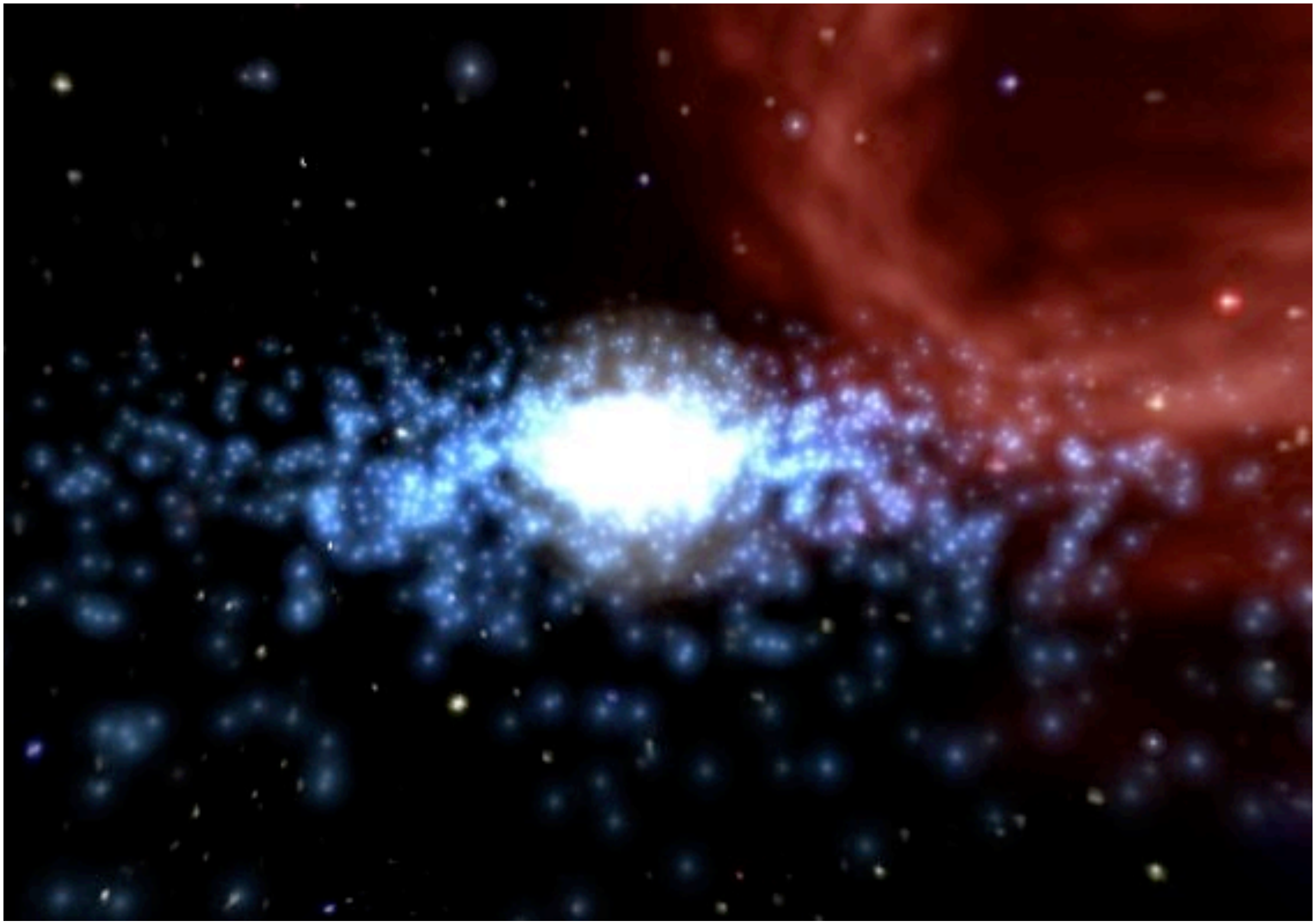
Particle Torch

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



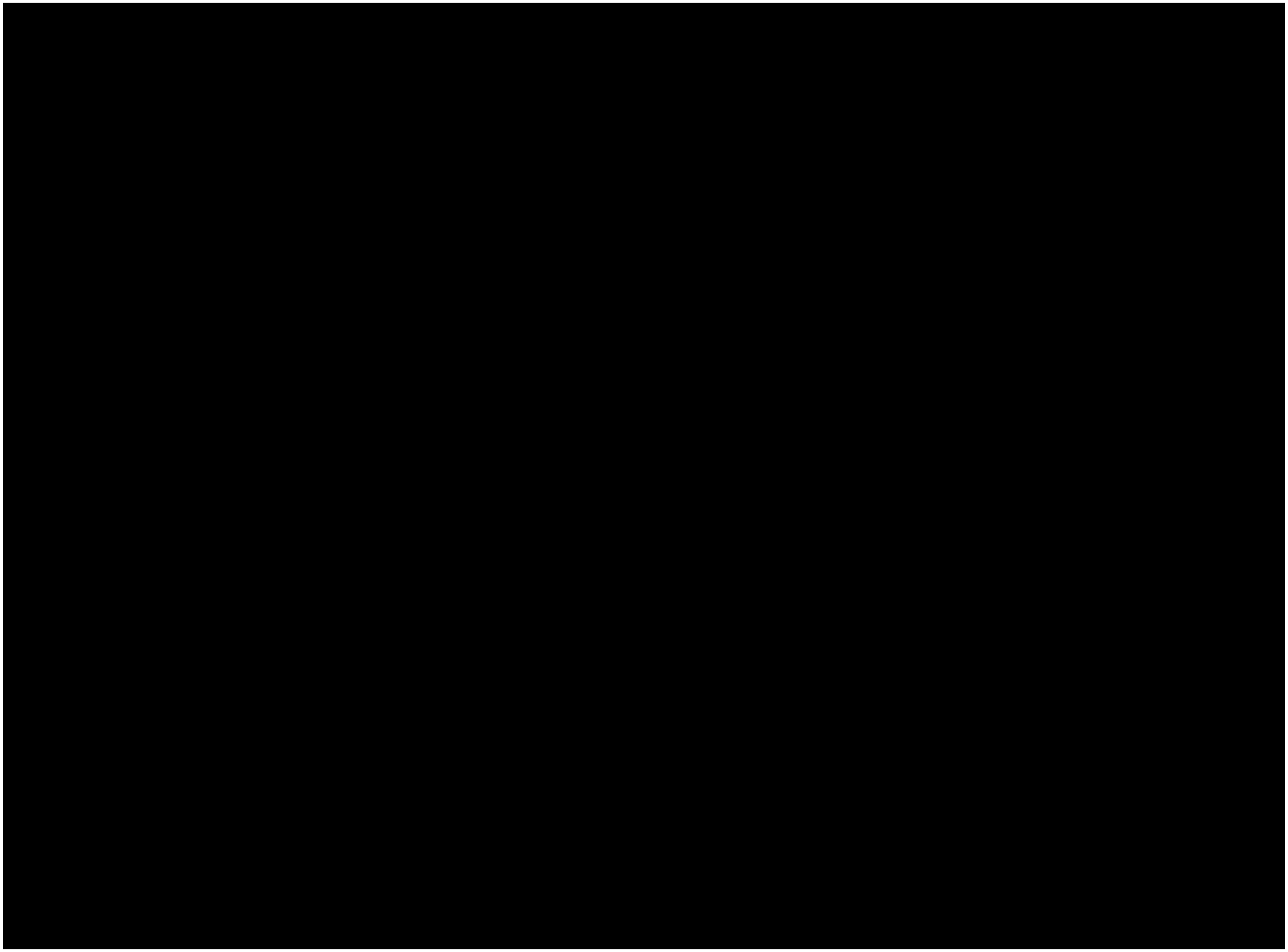
By John Tsiombikas from Wikipedia



By John Tsiombikas from Wikipedia

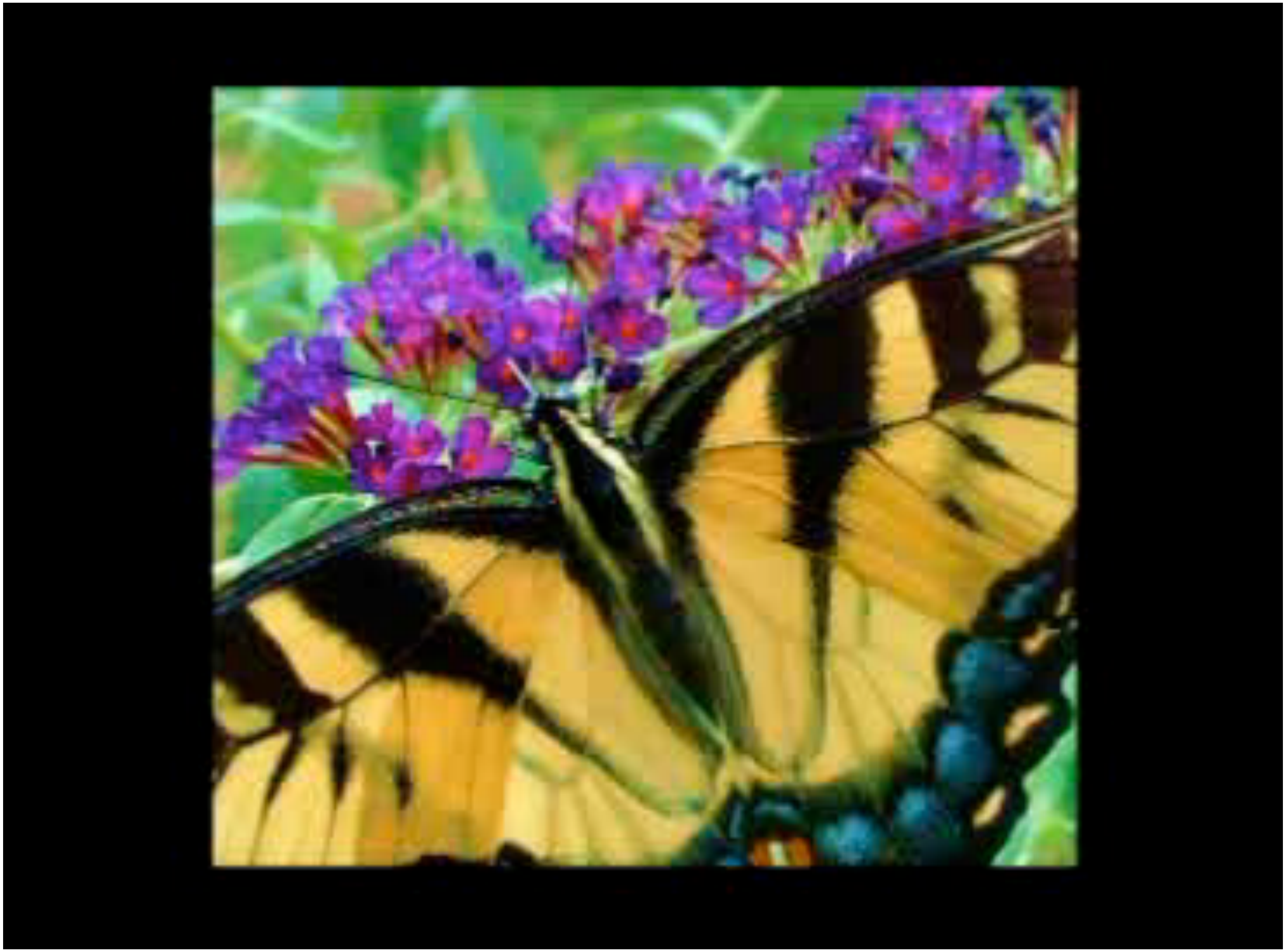
Commercial particle systems (wondertouch)

Commercial particle systems (wondertouch explosions)

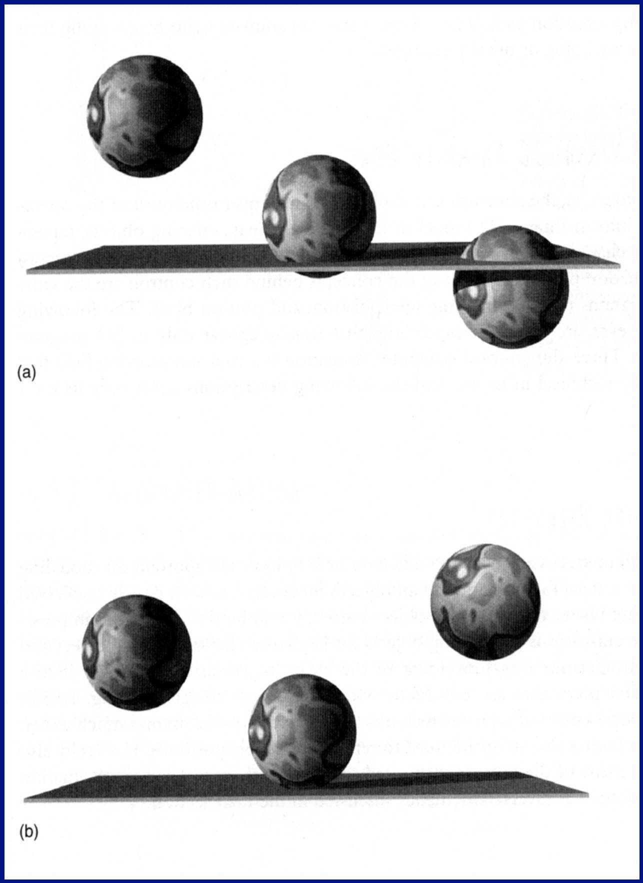


Commercial particle systems (wondertouch water)

Commercial particle systems (wondertouch distortions)



Ballistic + Collision



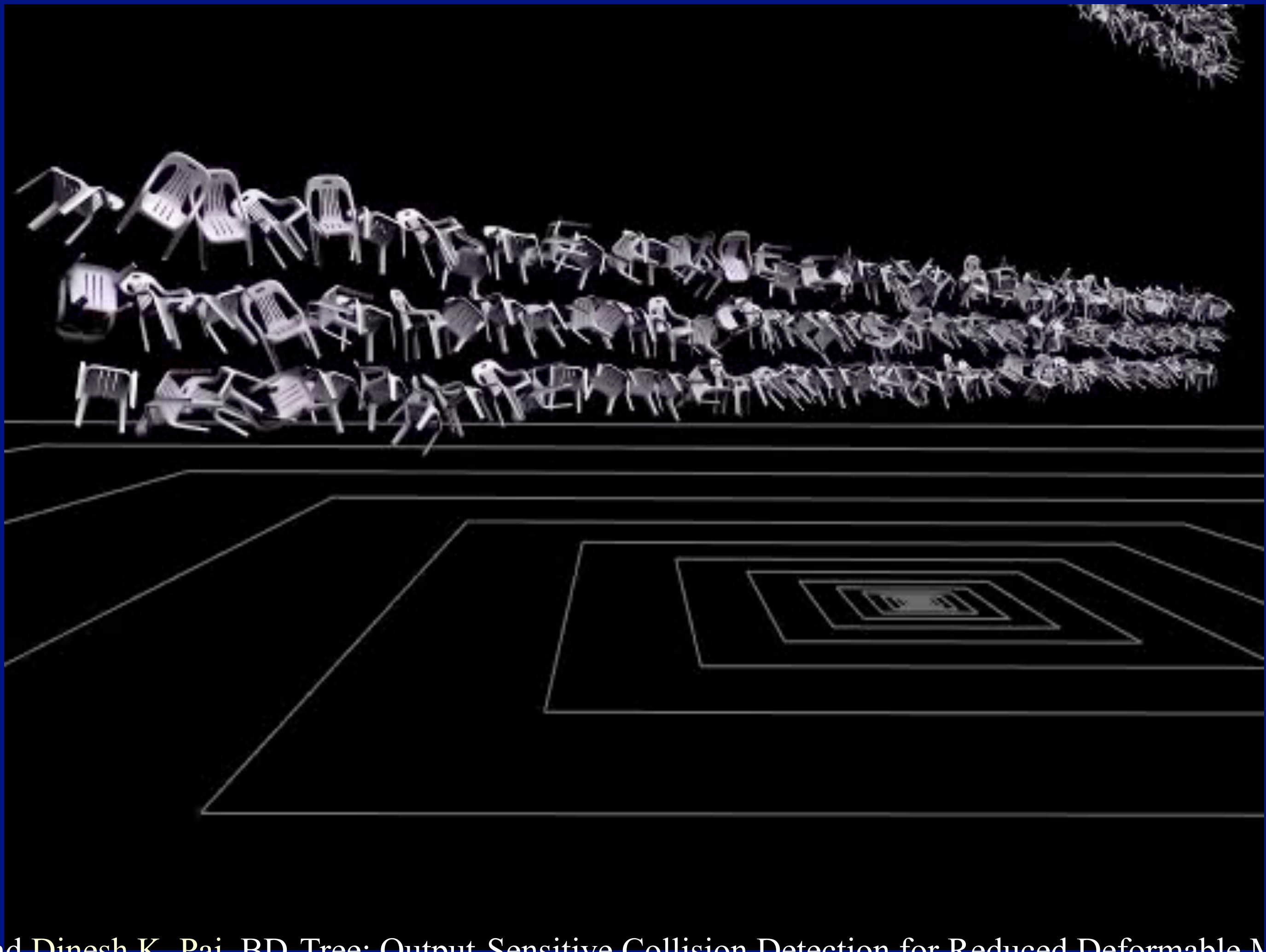
- Objects move freely under gravity until they collide.
- For accurate physical models, order in which collisions occur is important.

Collisions - detection

- Particles are straightforward
  - -ish (geometry is easy)
  - issues: undetected collisions
    - strategies:
      - identify safe bounds within which to advance time, search
        - use priority queue
        - but this may force quite small time steps
      - potential barrier
        - but this may force quite small time steps; stiffness
        - backward Euler helps, but only within limits
- Rigid objects more difficult
  - geometry: hierarchy of bounding spheres
  - strategies remain the same

12,201 chairs;
218,568,714 triangles

Collision: SOA

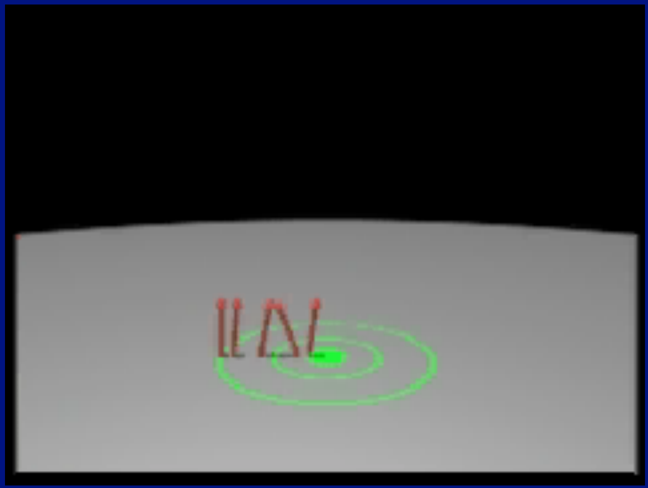


Doug L. James and Dinesh K. Pai, BD-Tree: Output-Sensitive Collision Detection for Reduced Deformable Models, ACM Transactions on Graphics (ACM SIGGRAPH 2004), 23(3), 2004.

Collisions - resolution

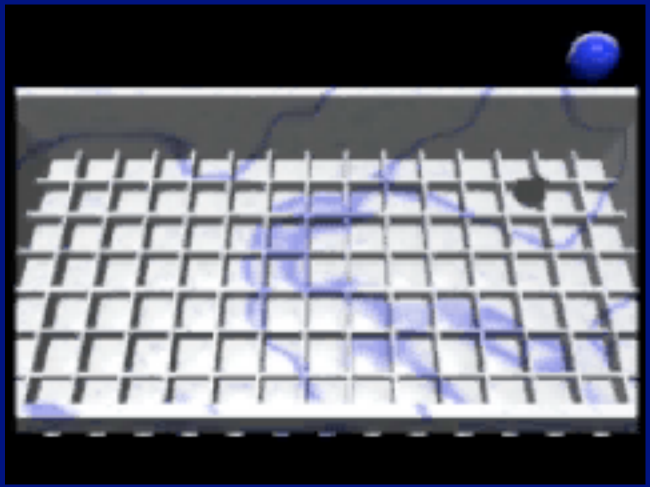
- Strategies:
  - potential field
  - explicit collision model
    - state_out=F(state_in, physical parameters)
    - typical physical parameters:
      - friction, coefficient of restitution
  - data driven
    - match inputs to data, read off outputs
- Collisions
  - produce randomness in motion
 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

Control via collisions

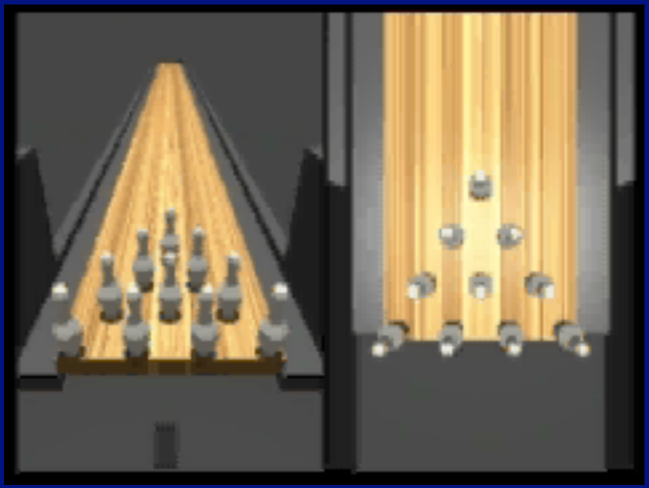


- Collisions are an important source of randomness
  - particularly in the case of sharp edges, rotation -> dice
  - physical parameters typically vary over space
    - Idea: modify physical parameters at collisions to produce desired outcome
      - Issues:
        - extremely complex search
        - requires very fast simulation
          - Notice: each object can be advanced different timesteps

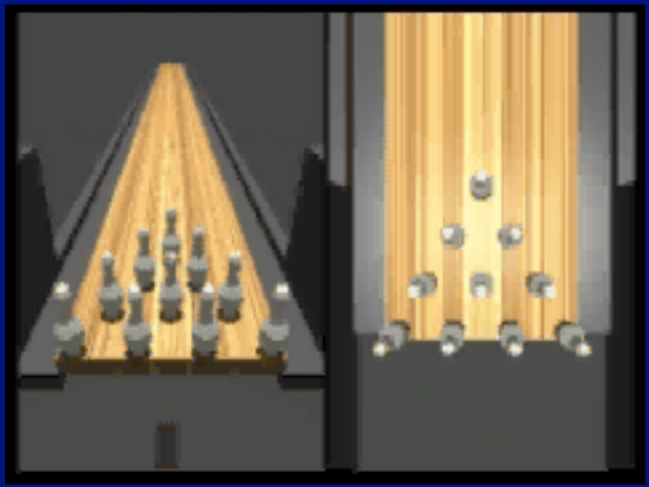
Stephen Chenney and D.A.Forsyth, "Sampling Plausible Solutions to Multi-Body Constraint Problems". SIGGRAPH 2000 Conference Proceedings, pages 219-228, July 2000.



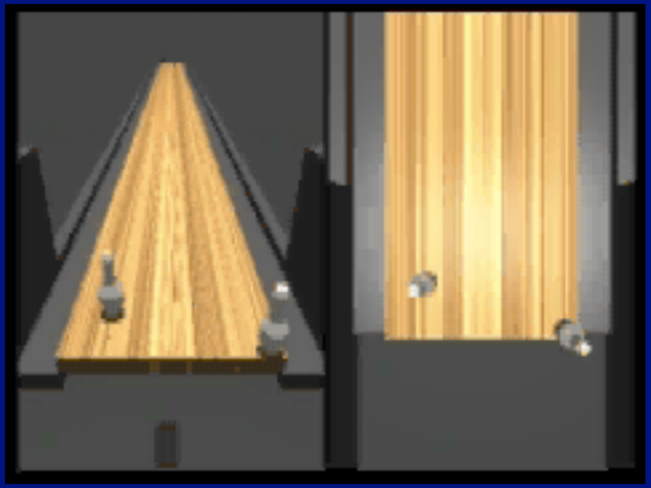
Stephen Chenney and D.A.Forsyth, "Sampling Plausible Solutions to Multi-Body Constraint Problems". SIGGRAPH 2000 Conference Proceedings, pages 219-228, July 2000.



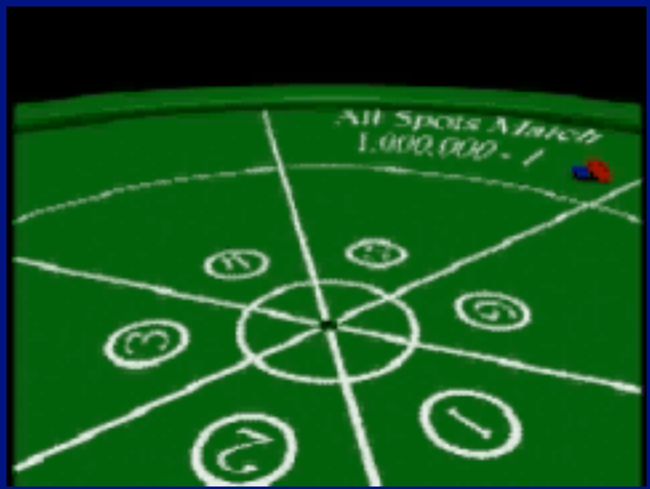
Stephen Chenney and D.A.Forsyth, "Sampling Plausible Solutions to Multi-Body Constraint Problems". SIGGRAPH 2000 Conference Proceedings, pages 219-228, July 2000.



Stephen Chenney and D.A.Forsyth, "Sampling Plausible Solutions to Multi-Body Constraint Problems". SIGGRAPH 2000 Conference Proceedings, pages 219-228, July 2000.



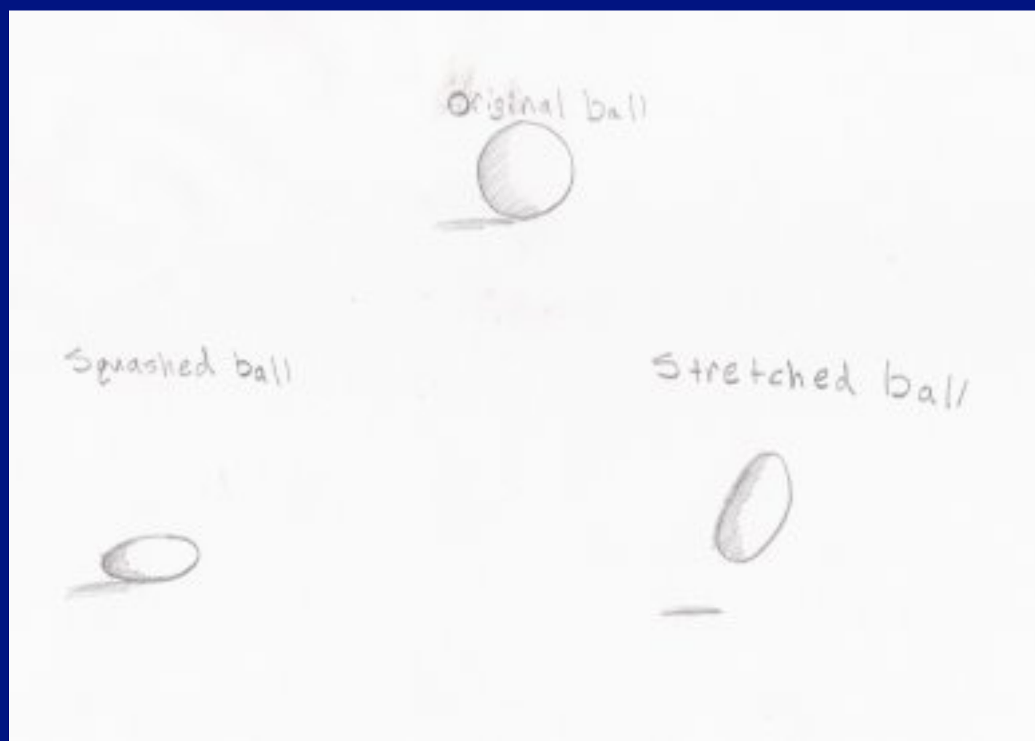
Stephen Chenney and D.A.Forsyth, "Sampling Plausible Solutions to Multi-Body Constraint Problems". SIGGRAPH 2000 Conference Proceedings, pages 219-228, July 2000.



Stephen Chenney and D.A.Forsyth, "Sampling Plausible Solutions to Multi-Body Constraint Problems". SIGGRAPH 2000 Conference Proceedings, pages 219-228, July 2000.

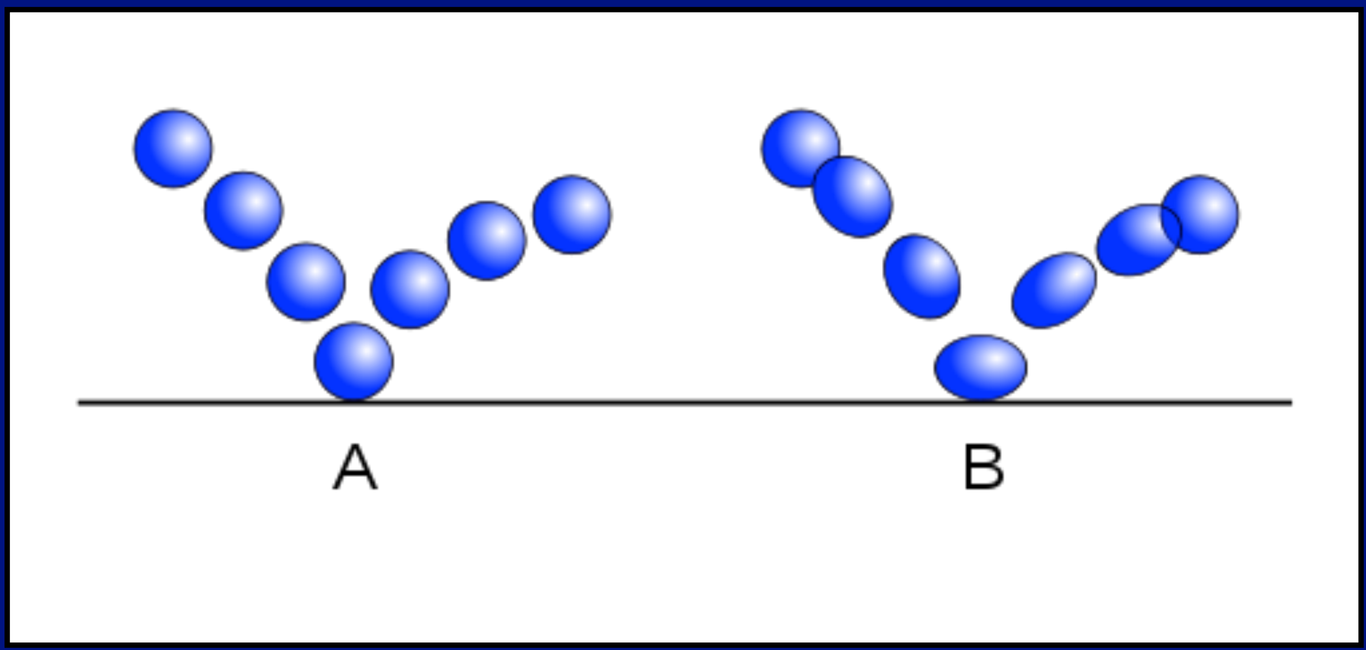
Making collisions more “cartoony”

- 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

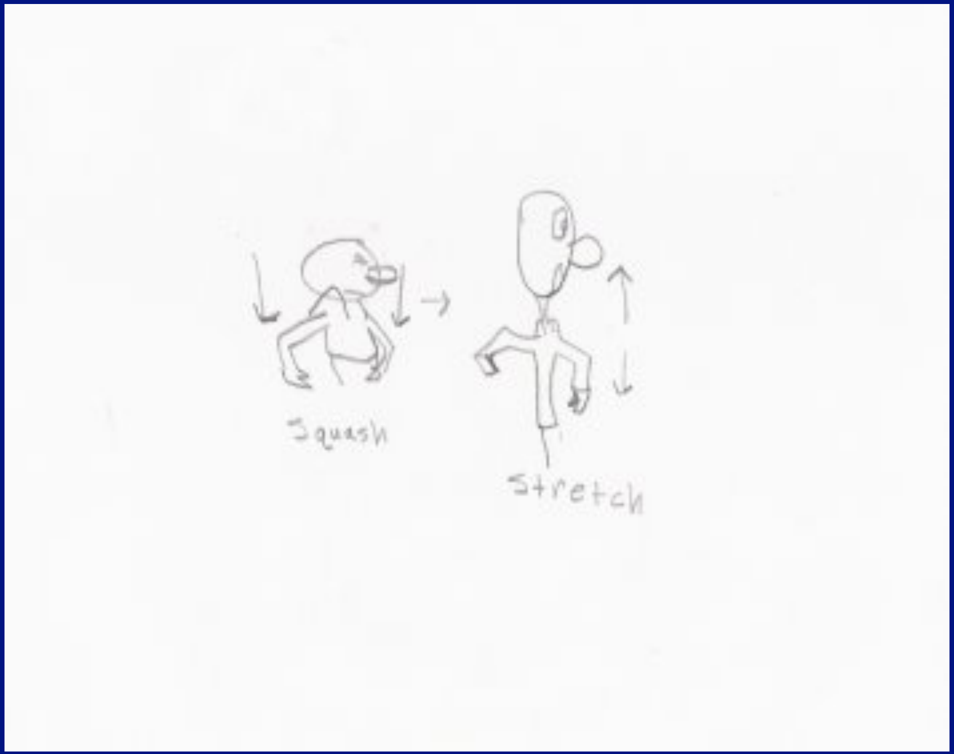
Making collisions more "cartoony"



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

Making collisions more "cartoony"

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



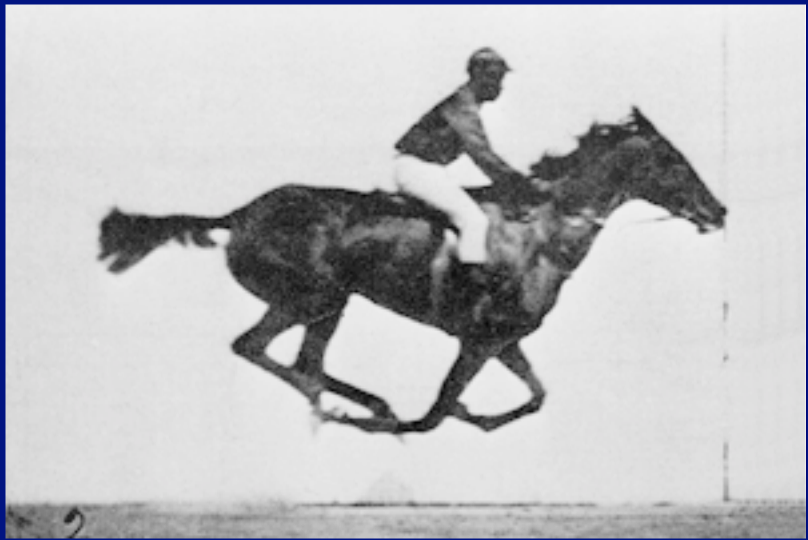
http://en.wikipedia.org/wiki/File:Squash_and_Stretch.jpg

12 principles of cartoon animation

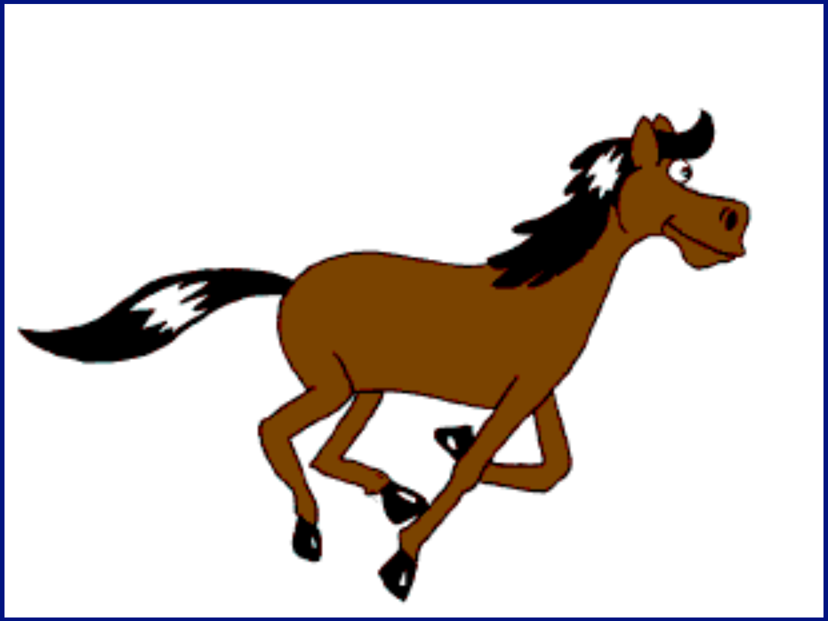
1. Squash and stretch
2. Anticipation
3. Staging
4. Straight Ahead Action and Pose to Pose
5. Follow Through and Overlapping Action
6. Slow In and Slow Out
7. Arcs
8. Secondary Action
9. Timing
10. Exaggeration
11. Solid Drawing
12. Appeal

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

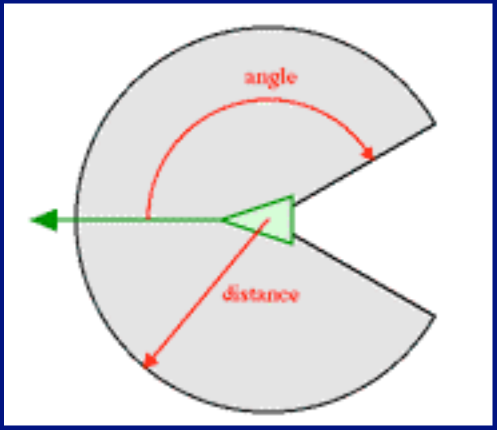
Secondary motion



Secondary motion

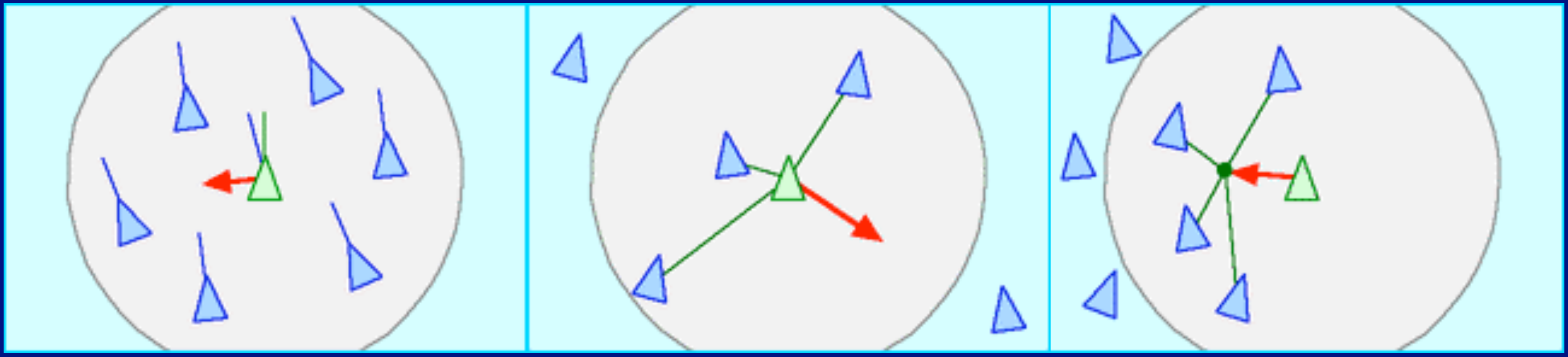


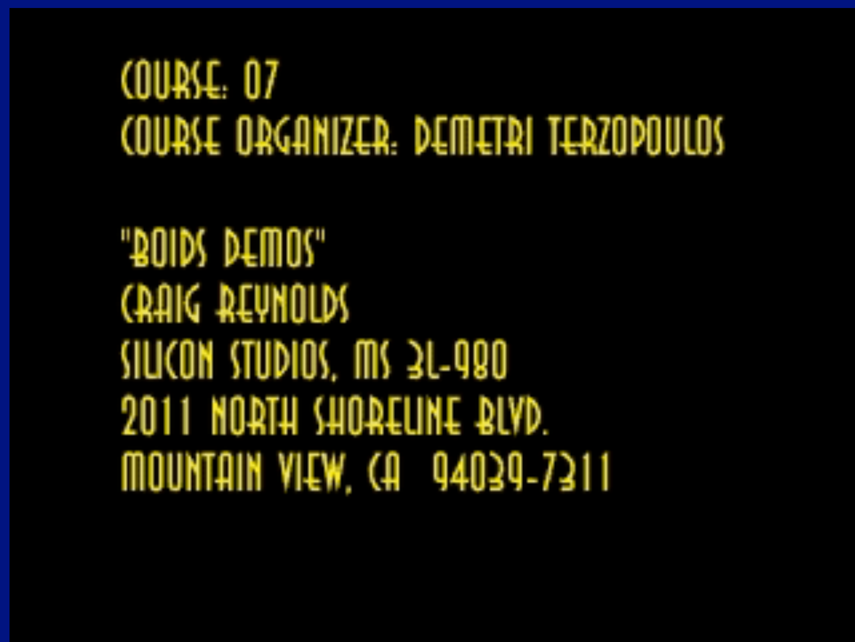
Flocking - Boids



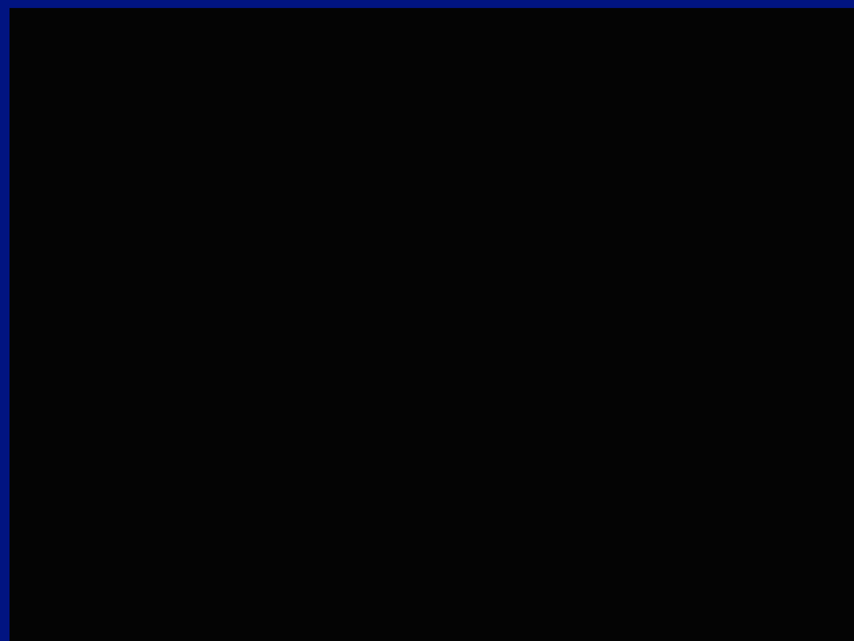
- We'd like things to move in schools
  - and not hit each other, objects
  - abstraction: particle with rocket with maximum force
- 3 goals
  - How to accelerate?
    - each goal gives an acceleration; weighted sum
    - accumulate in priority order until acceleration exceeds threshold,
      - then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

Alignment Separation Cohesion





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



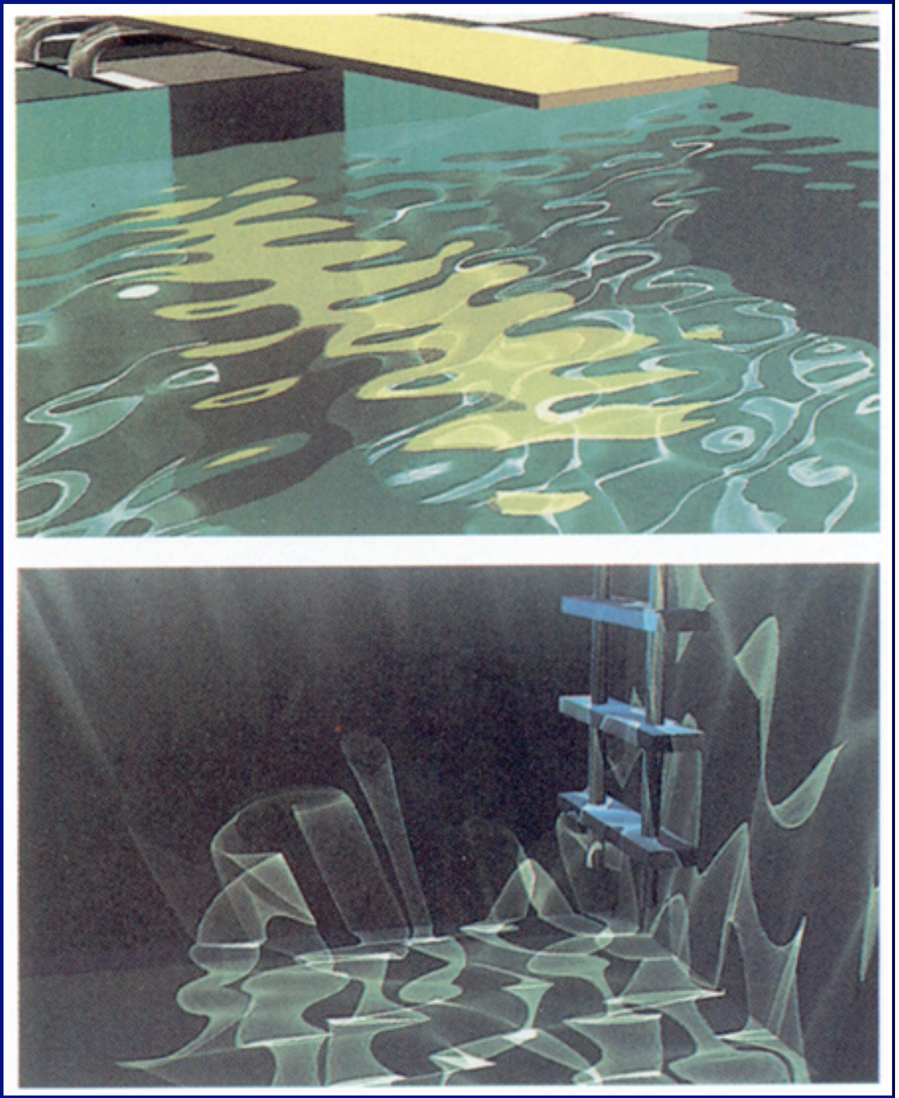
http://www.red.com/cwr/boids.html

Procedural ideas

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

Procedural waves

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

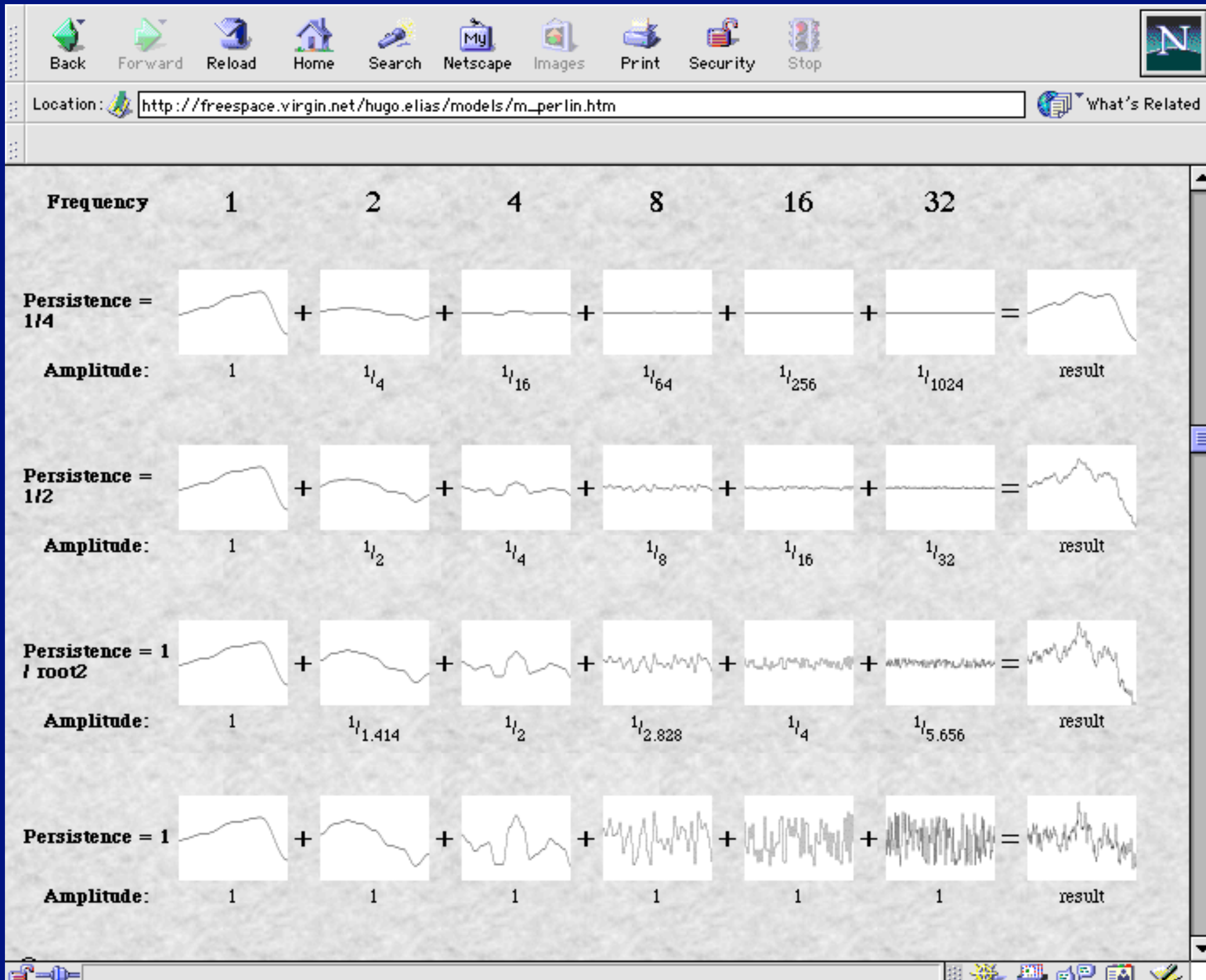


Turbulence/Perlin noise

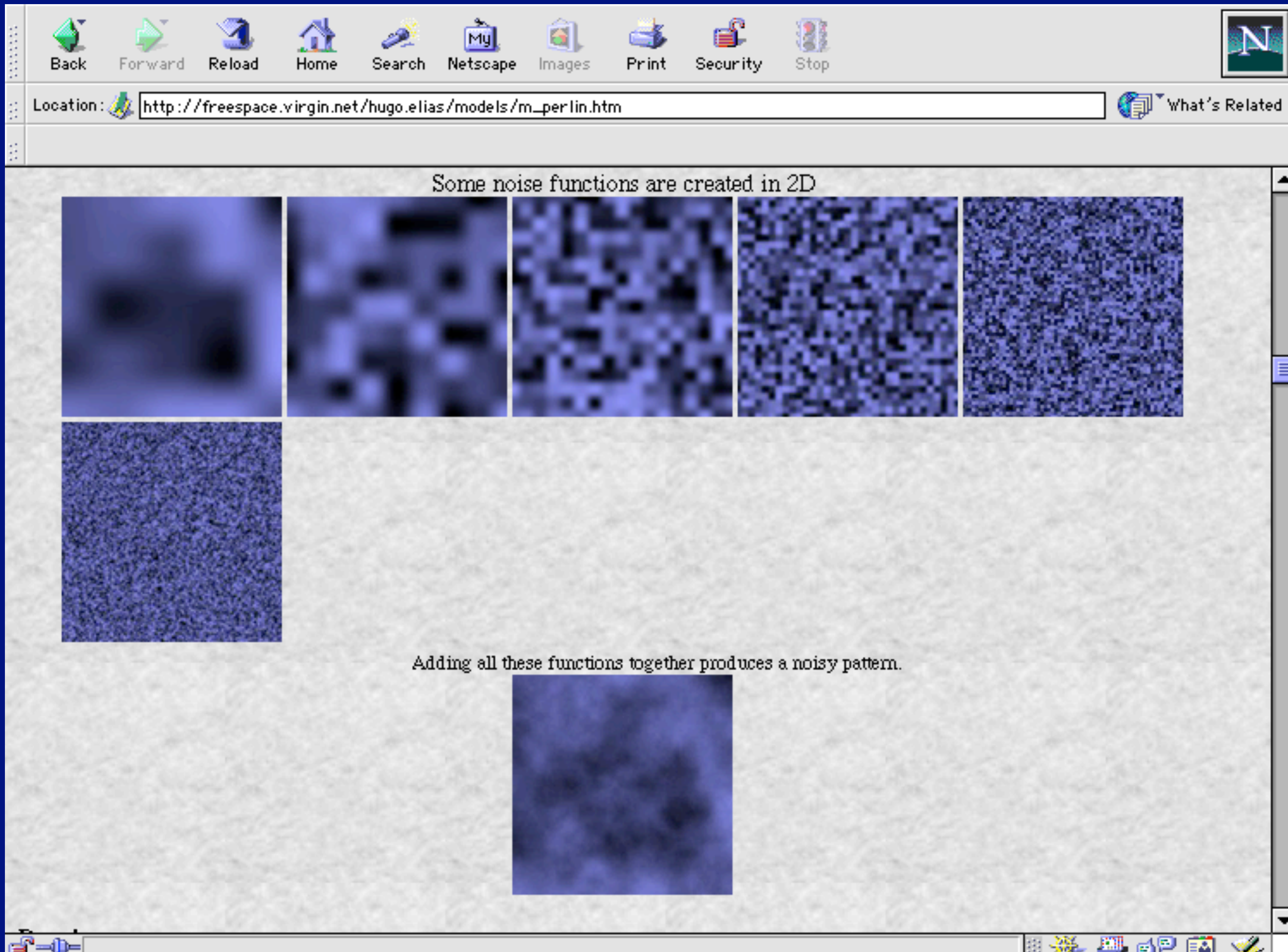
- Many natural textures look like noise or “smoothed” noise
  - (marble, flames, clouds, terrain, etc.)
- Issue:
  - obtain the right kind of smoothing
- Strategy:
  - construct noise functions at a variety of scales
    - do this by drawing samples from a random number generator at different spacings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



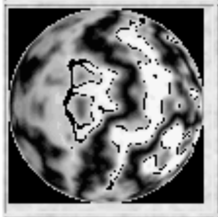
Back
Forward
Reload
Home
Search
Netscape
Images
Print
Security
Stop
Location: http://freespace.virgin.net/hugo.elias/models/m_perlin.htm
What's Related
Frequency 1 2 4 8 16 32
Persistence = 1/4
Amplitude: 1 1/4 1/16 1/64 1/256 1/1024 result
Persistence = 1/2
Amplitude: 1 1/2 1/4 1/8 1/16 1/32 result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

Back Forward Reload Home Search Netscape Images Print Security Stop

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

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
```

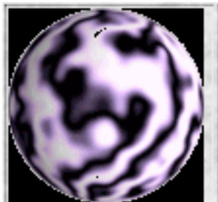
References

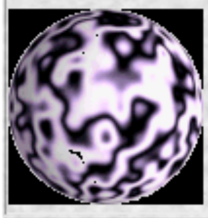
Procedural textures: **http://developer.intel.com/drg/mmx/appnotes/proctex.htm**
Intel Developer Site article about using the new MMX technology to render Perlin Noise in real time.

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

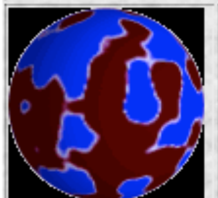
Location: http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

The following textures were made with 3D Perlin Noise

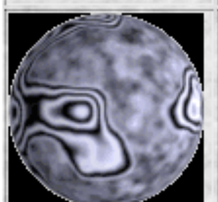




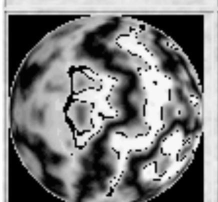
Standard 3 dimensional perlin noise. 4 octaves, persistence 0.25 and 0.5



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.

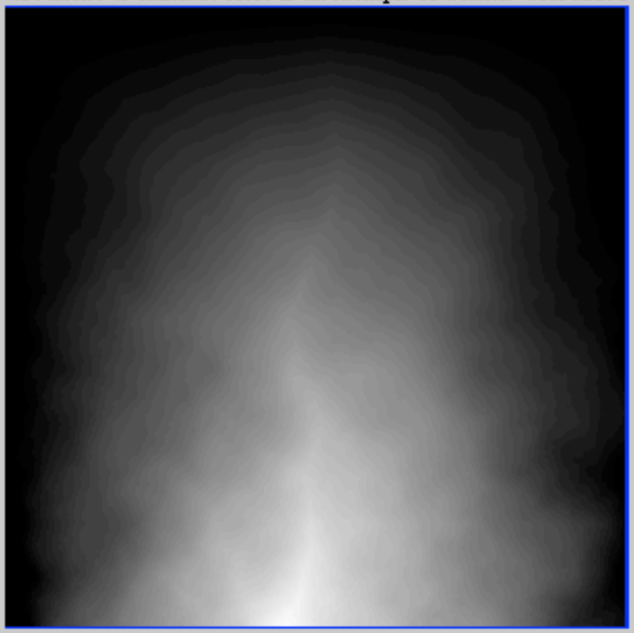


A marbly texture can be made by using a Perlin function as an offset to a cosine function.

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

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

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.

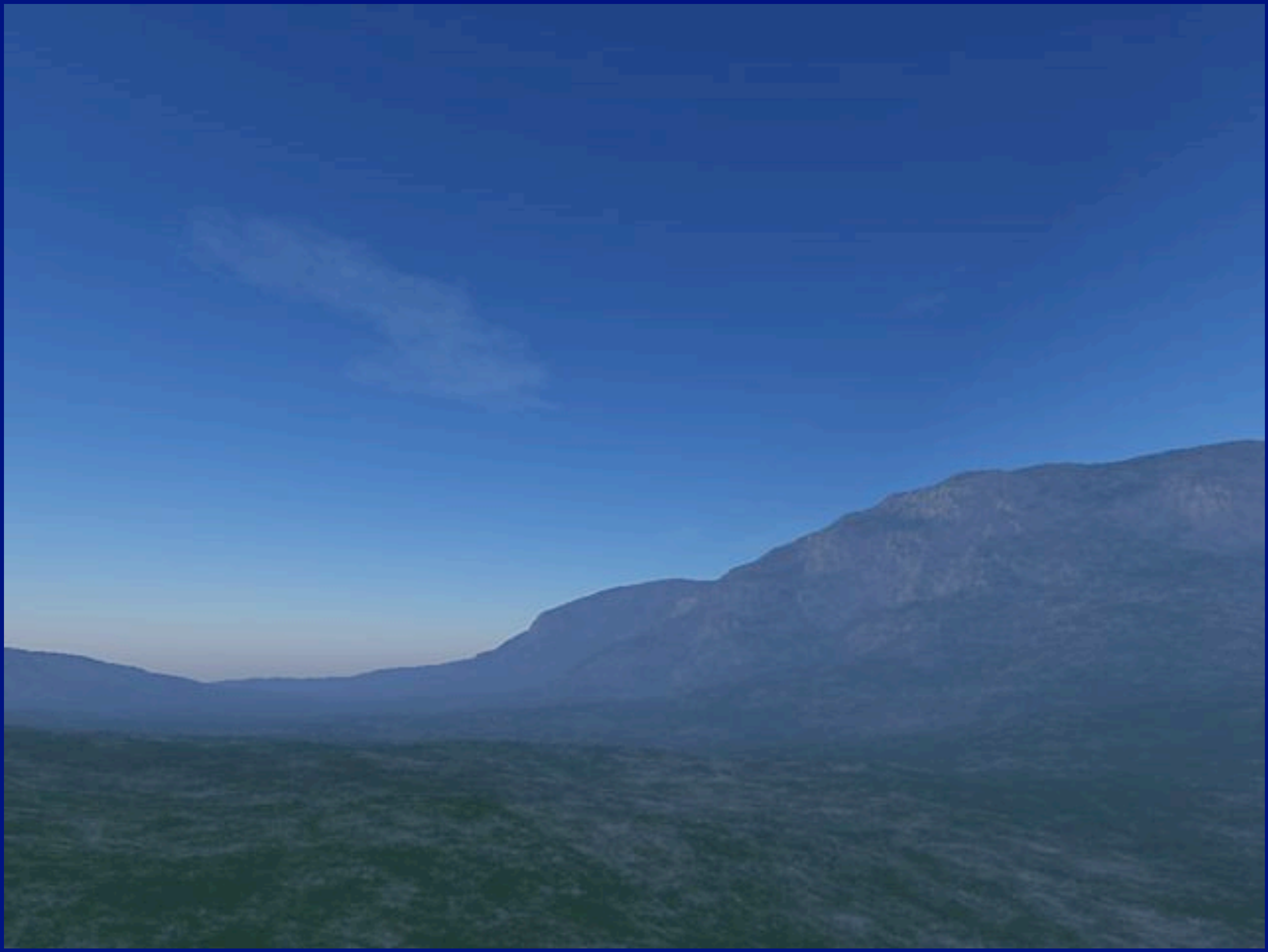


Conclusions

- 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)

Back

matt@owl.WPI.EDU
Tue Apr 25 11:51:57 EDT 1995



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

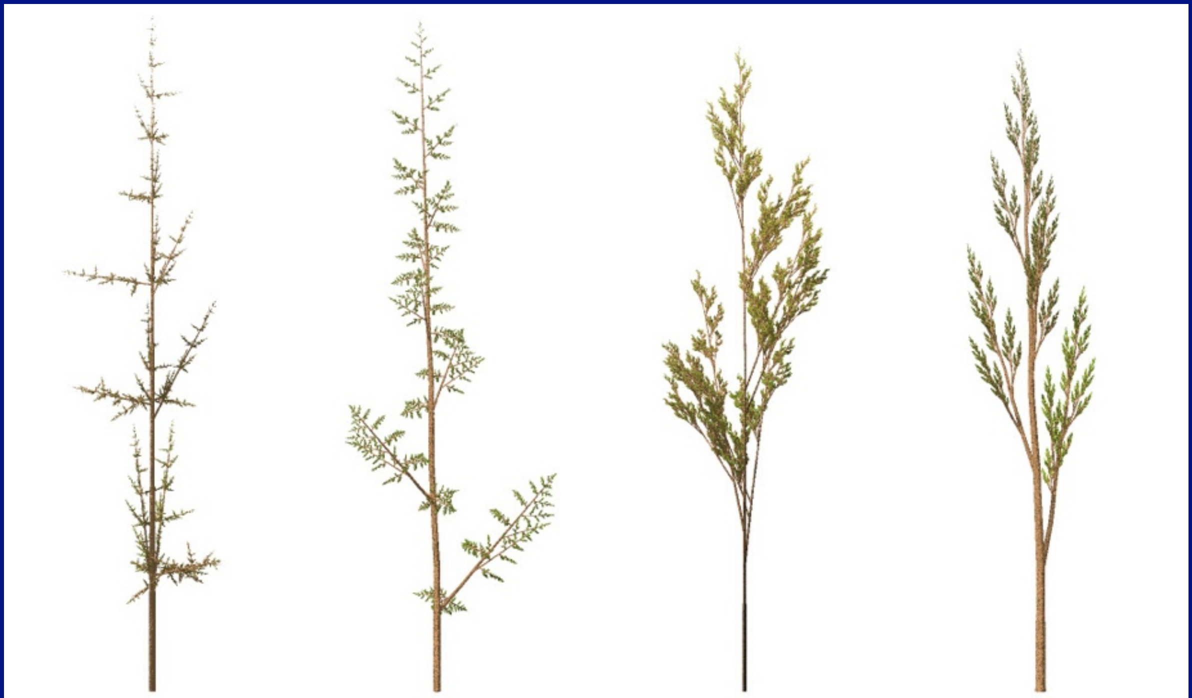
Terrain, clouds generated using procedural textures and Perlin noise
http://www.planetside.co.uk/ -- tool is called Terragen

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

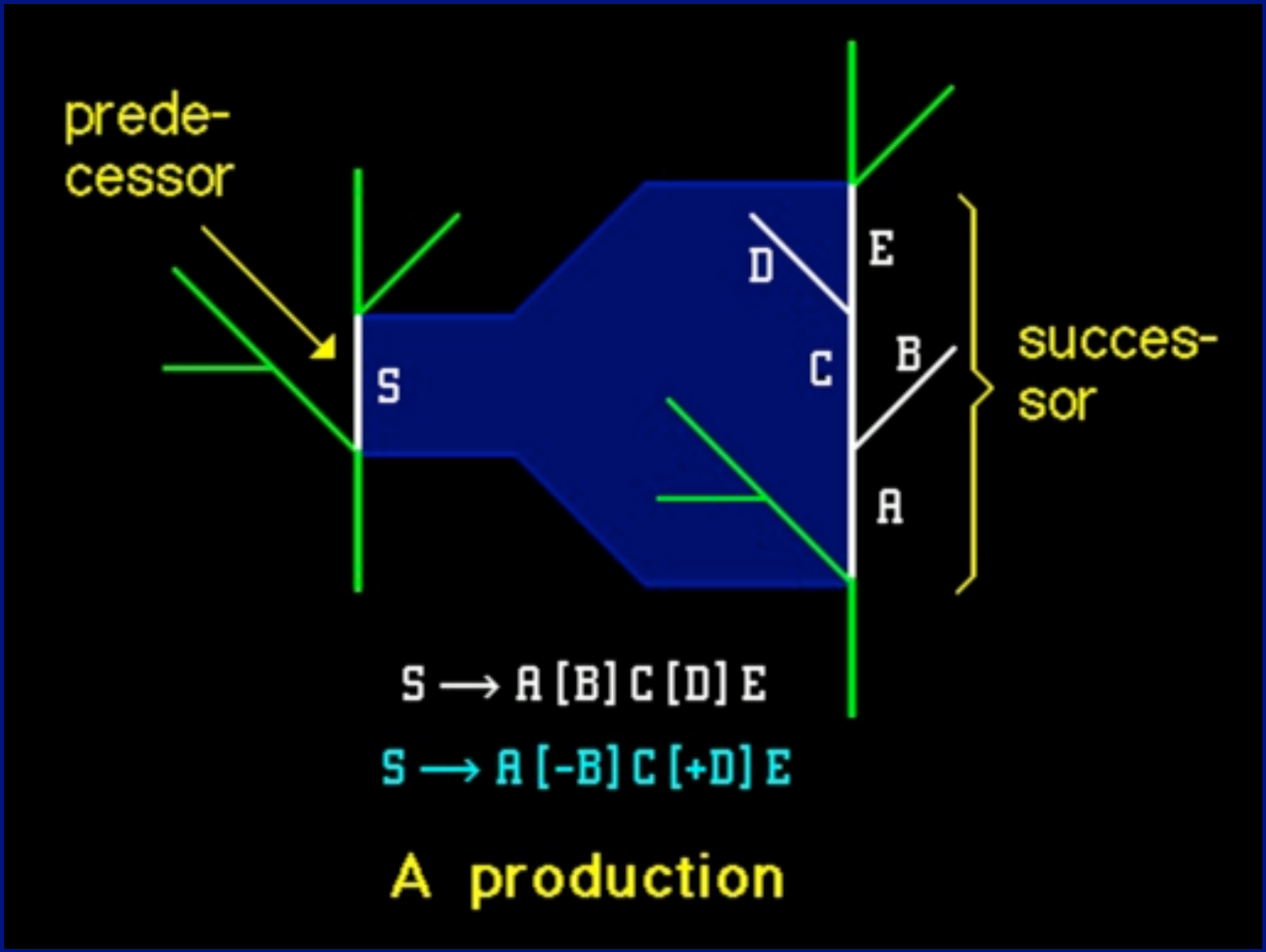
Procedural Animation: L-systems

- 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

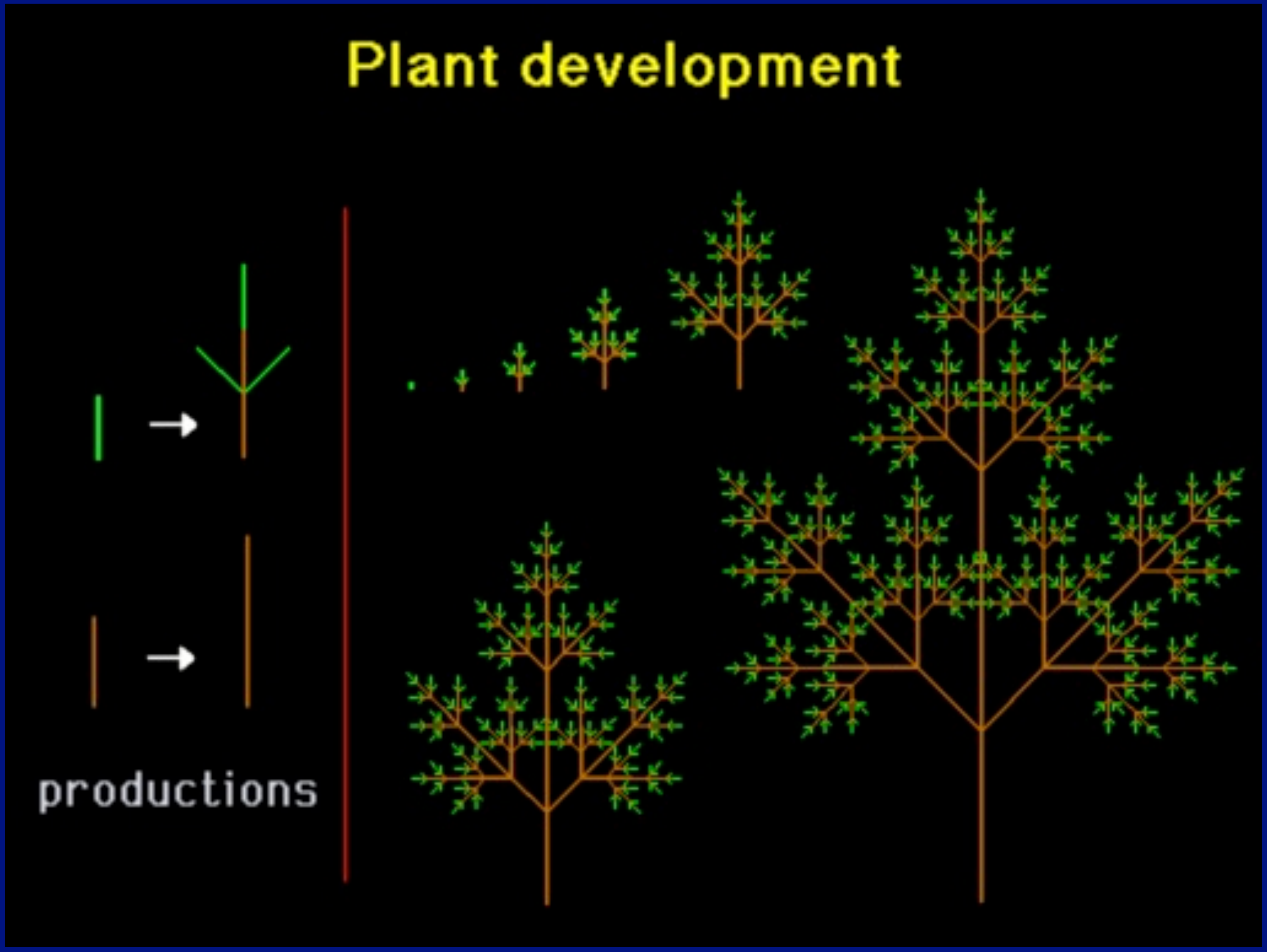


L-Systems



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

L-Systems

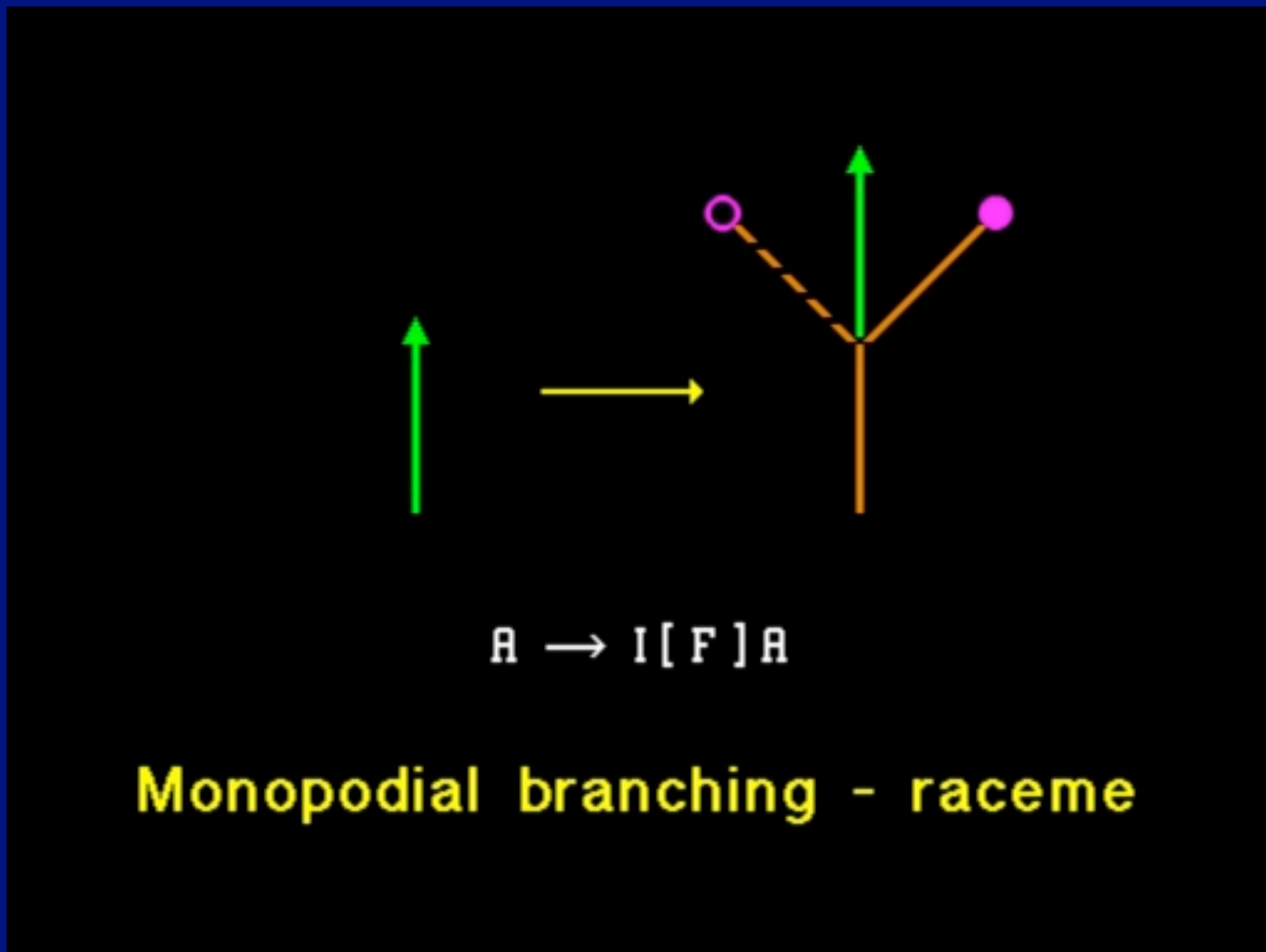


Prusinkiewicz, Hammel, Mech http://www.cpsc.ucalgary.ca/projects/bmv/vmm/title.html

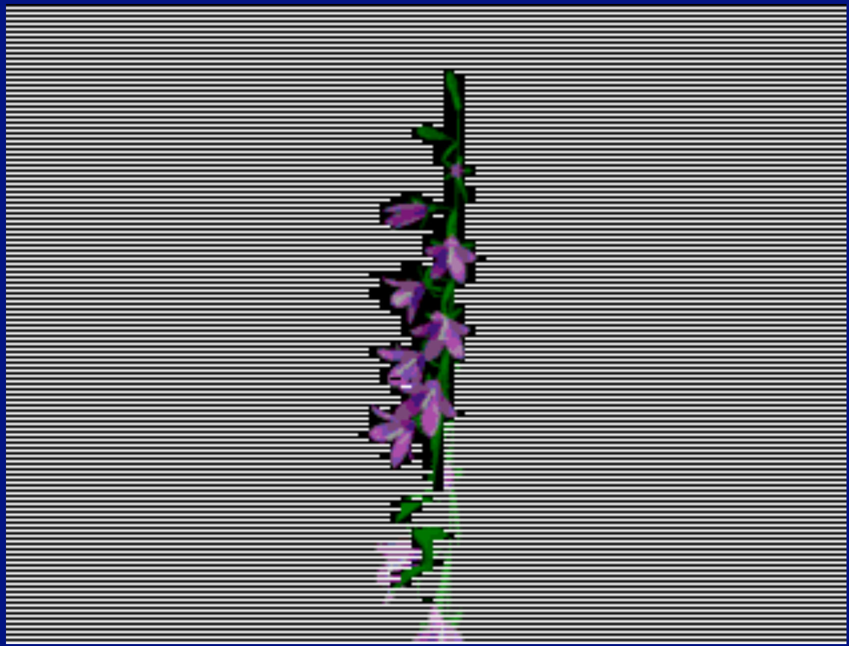
L-System plant growing

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

Flowers at side branches

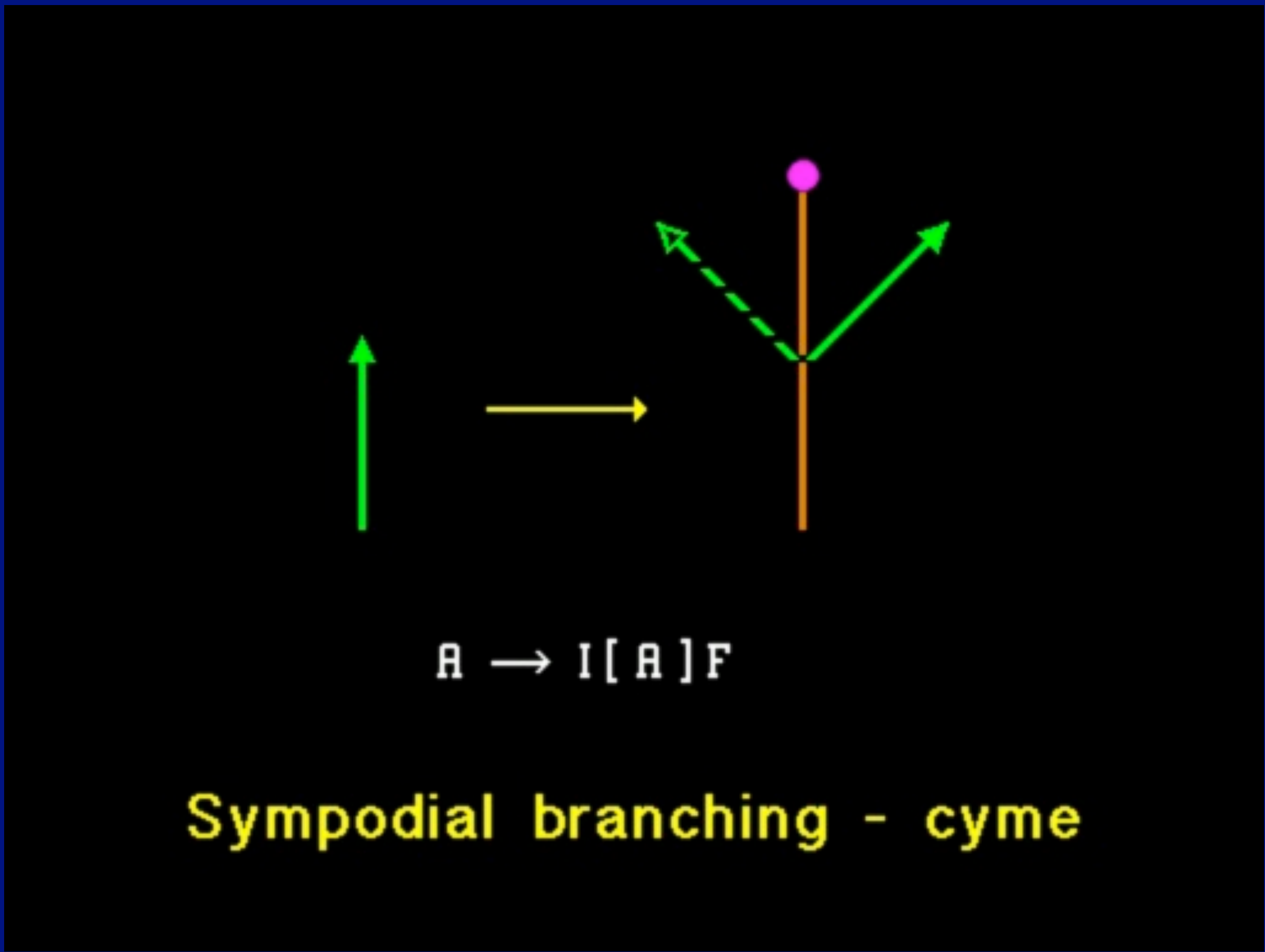


Prusinkiewicz, Hammel, Mech http://www.cpsc.ucalgary.ca/projects/bmv/vmm/title.html



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

Flowers at the apex

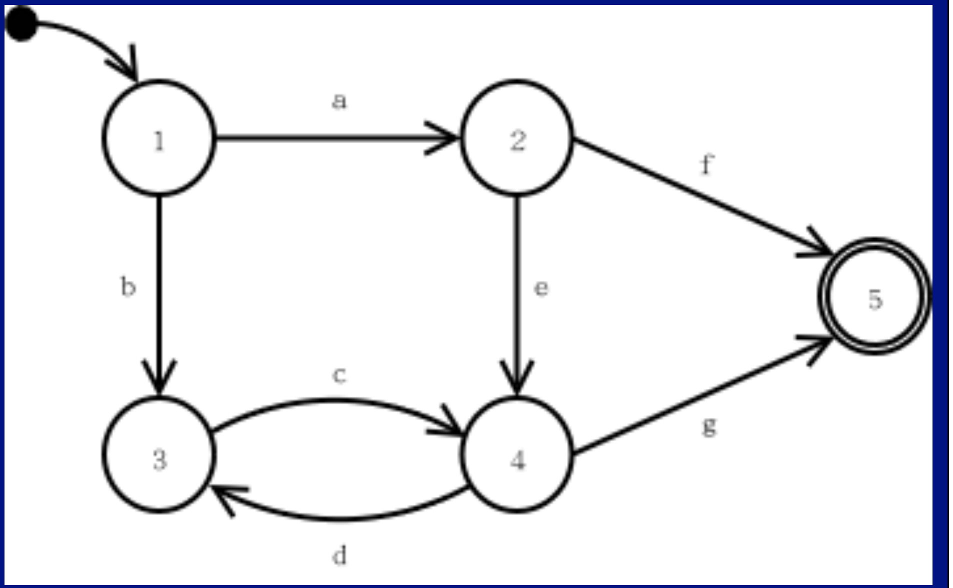


Prusinkiewicz, Hammel, Mech http://www.cpsc.ucalgary.ca/projects/bmv/vmm/title.html

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

Finite state machines

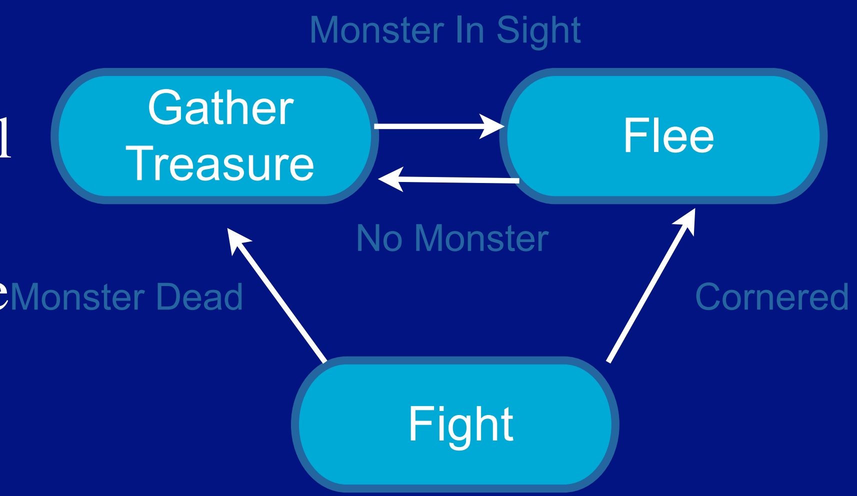
- FSM
  - set of states
    - special start, end state
  - input vocabulary
  - transition function
    - state change on receiving input



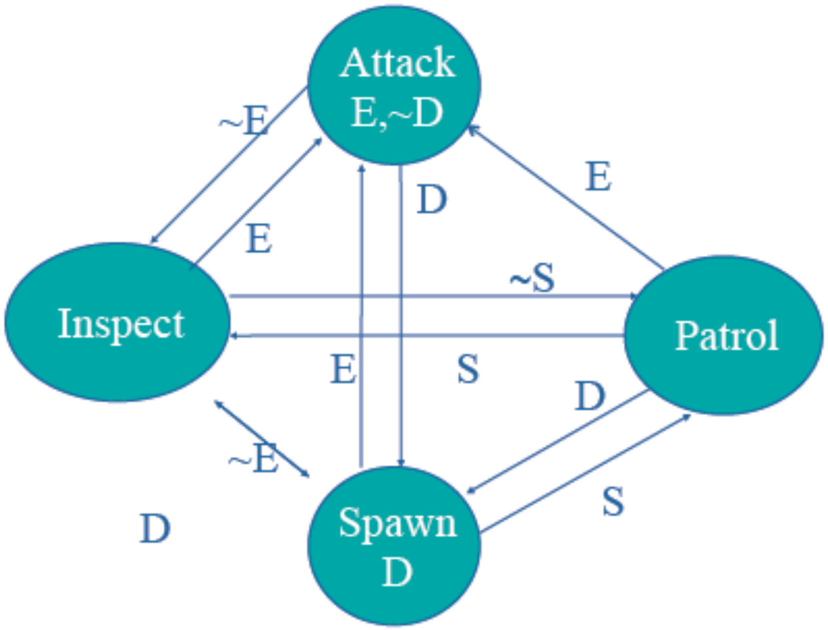
Slides after slides by Jarret Raim, LeHigh

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.



FSM Authoring

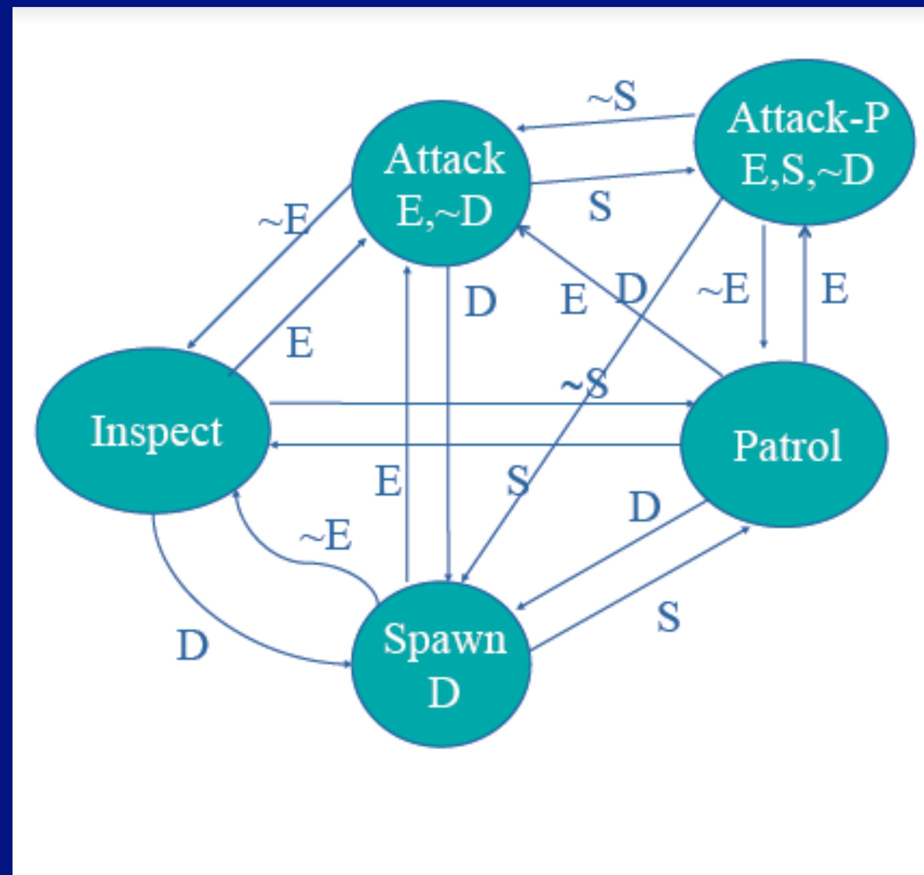


Problem: Can't go directly from attack to patrol. We'll fix this later.

- States
  - E: enemy in sight
  - S: hear a sound
  - D: dead
- Events
  - E: see an enemy
  - S: hear a sound
  - D: die
- Action performed
  - On each transition
  - On each update in some states (e.g. attack)

FSM Authoring

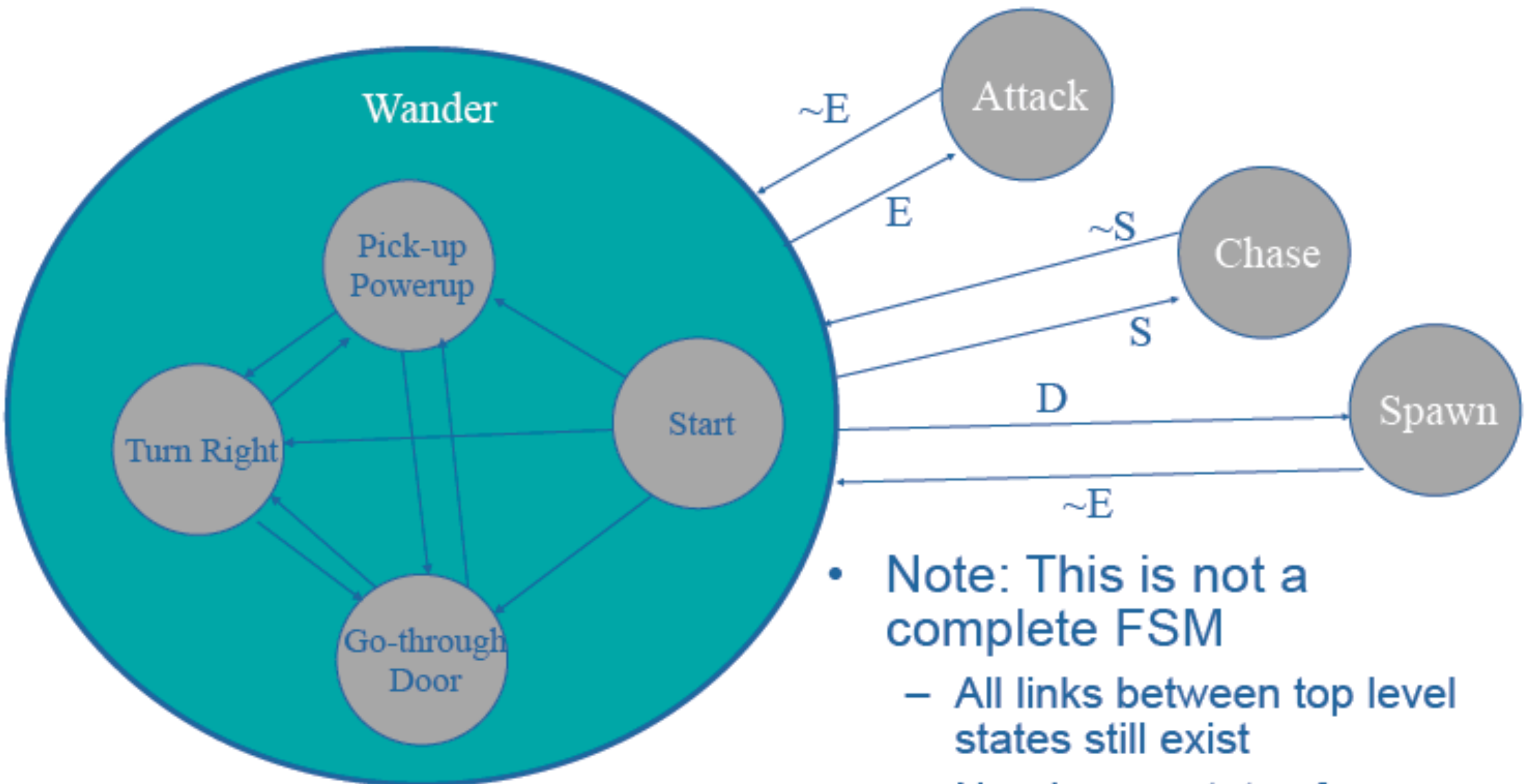
- Original FSM doesn't
  - remember previous state
  - so attack doesn't know if it was triggered from patrol or inspect (sound heard?)
- Could add a state
  - but this gets messy



Hierarchy

- Each state is an FSM
- Some events move within a level, some trigger a transition at higher levels
- When we enter a state, we need an initial state for that FSM
  - Default
  - Random
  - Depends on behaviour, etc.
- This is all just a (much bigger) FSM, but easier to author

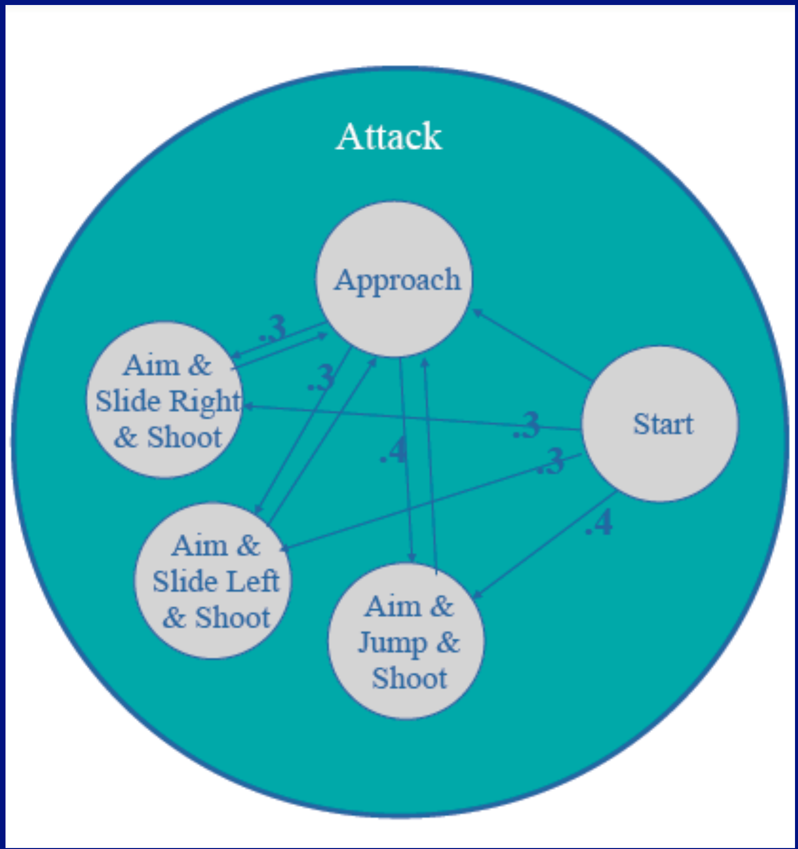
Hierarchy



- Note: This is not a complete FSM
  - All links between top level states still exist
  - Need more states for wander

Non-determinism

- Randomness easy to add, makes behaviour richer



Programming issues

- Typically work in a scripting language
  - Game engine, environment deal with details
  - e.g. events by polling? register/dispatch?
- Debugging
  - hard in rich environments
  - hard with multiple interacting state machines