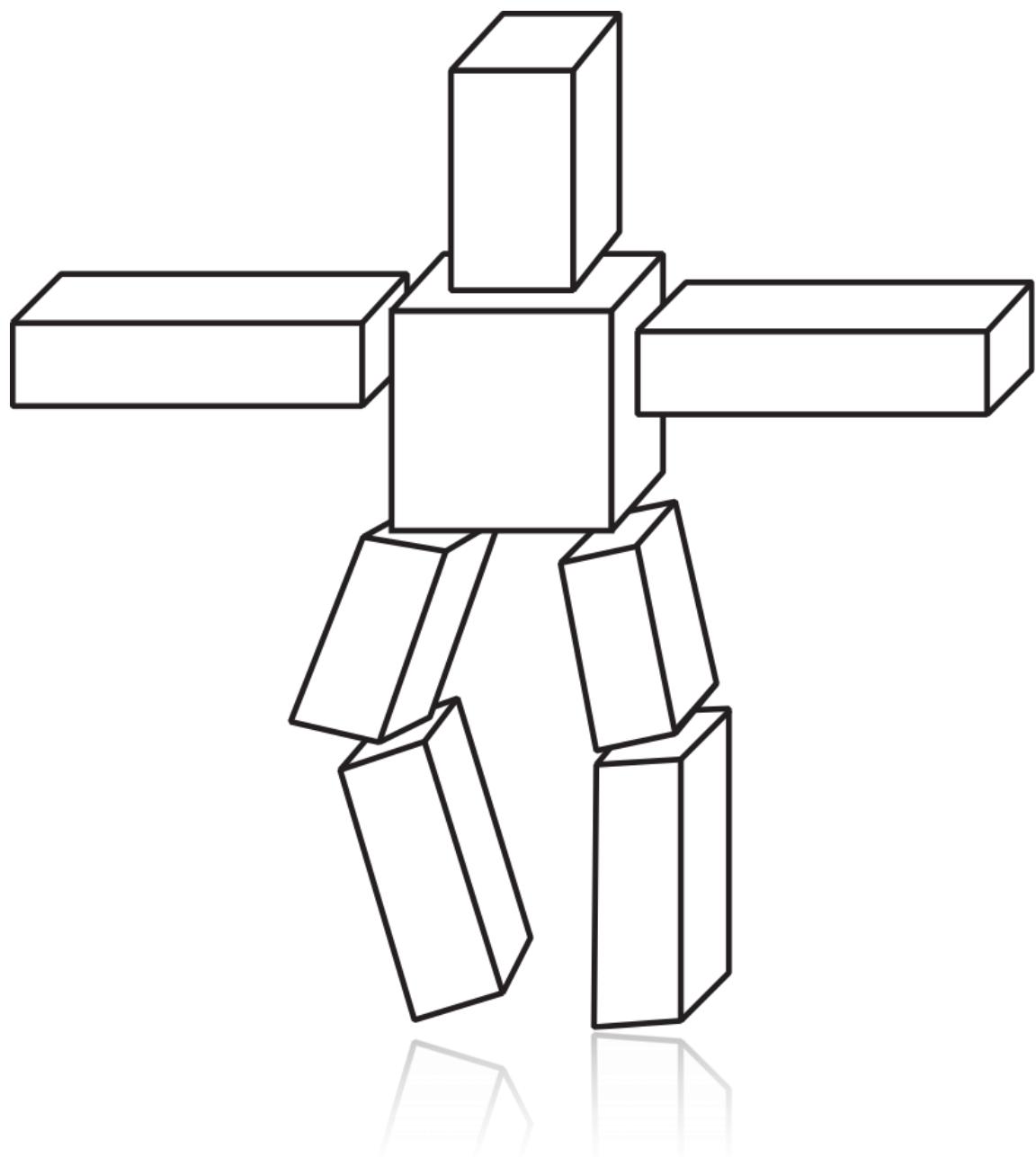


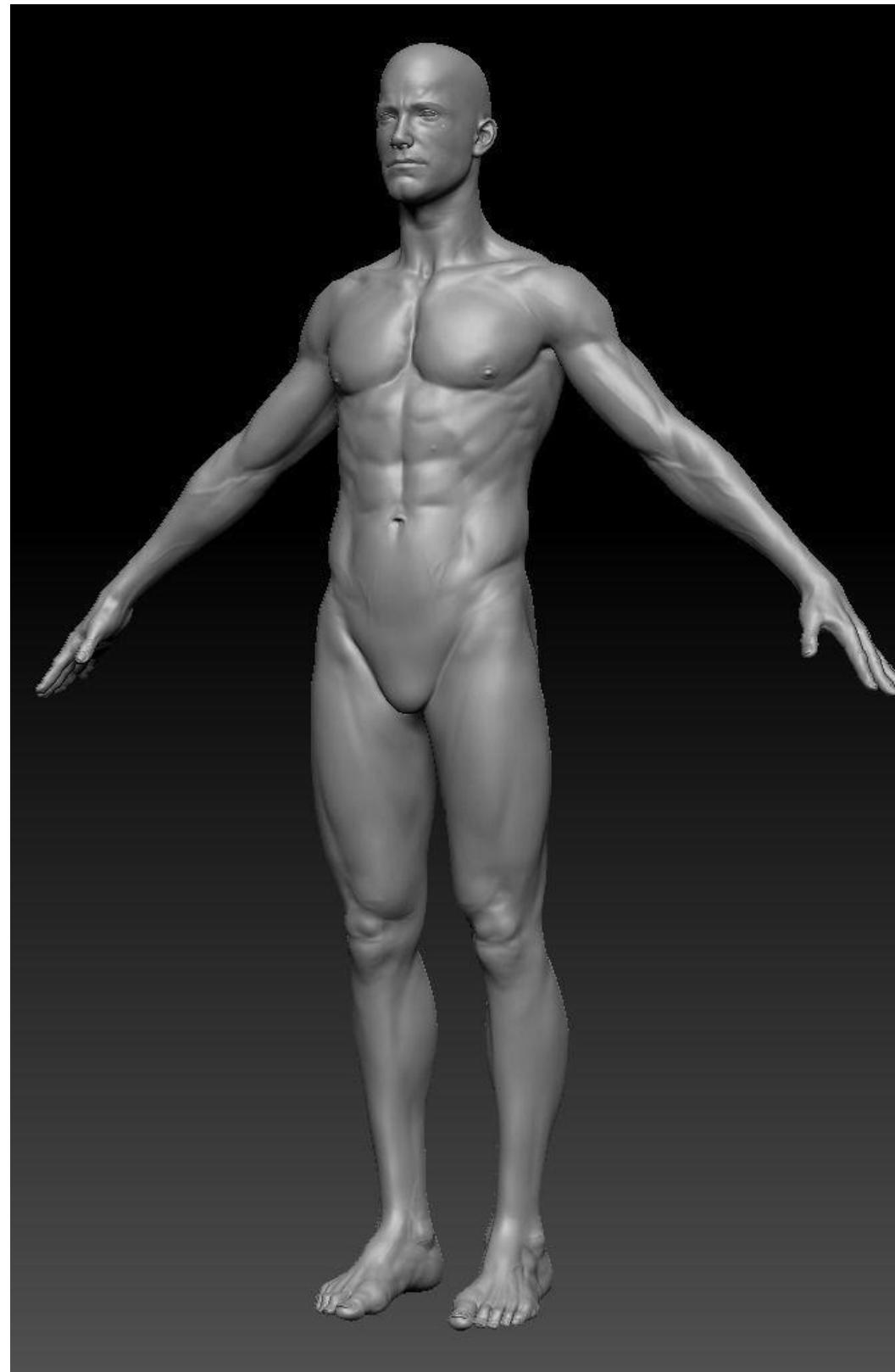
# **Intro to Computer Animation**

# Increasing the complexity of our models

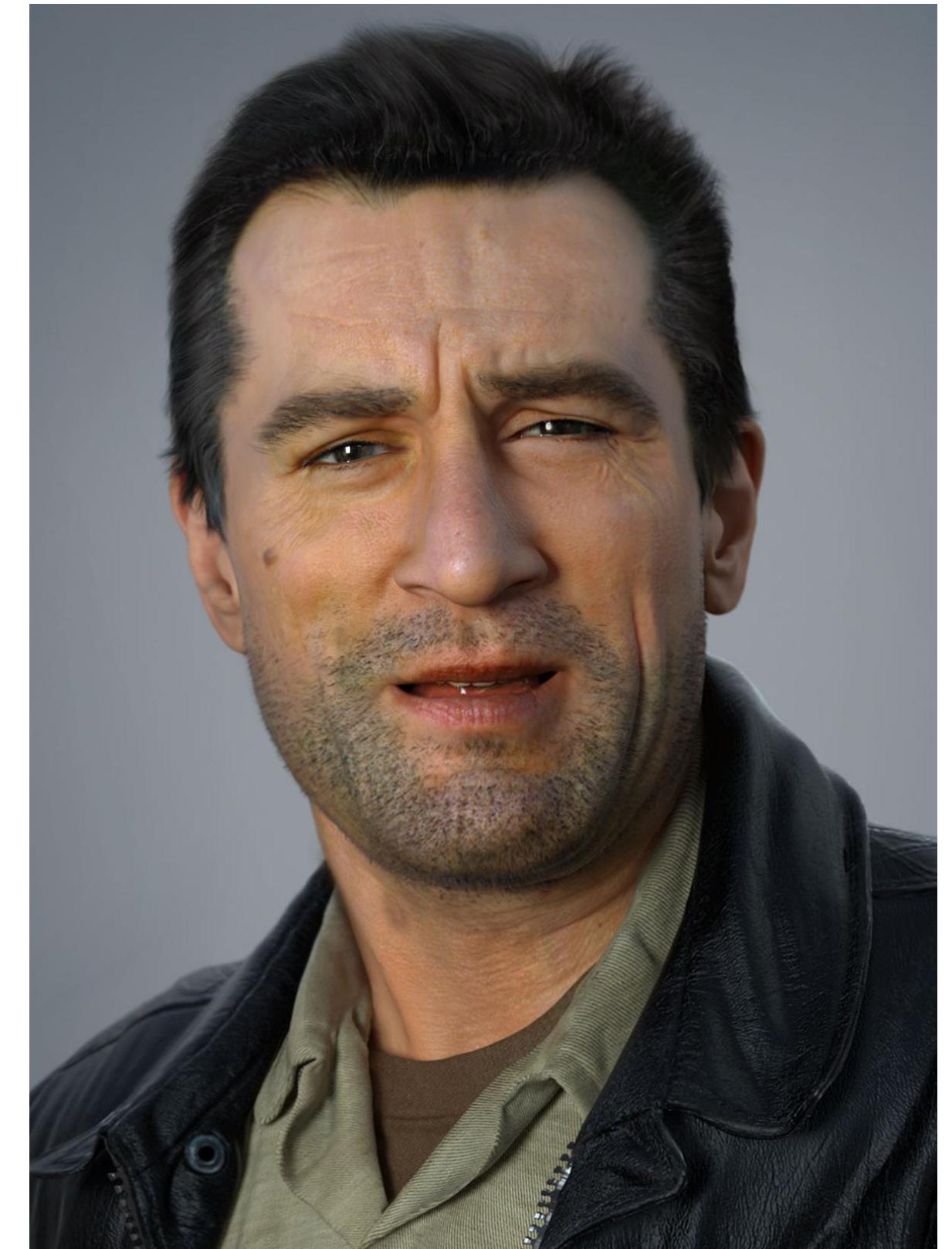
Transformations



Geometry



Materials, lighting



# Increasing the complexity of our models

...but what about *motion*?



(Misty Copeland, American Ballet Theatre)

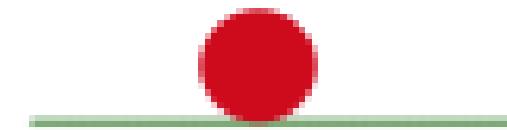
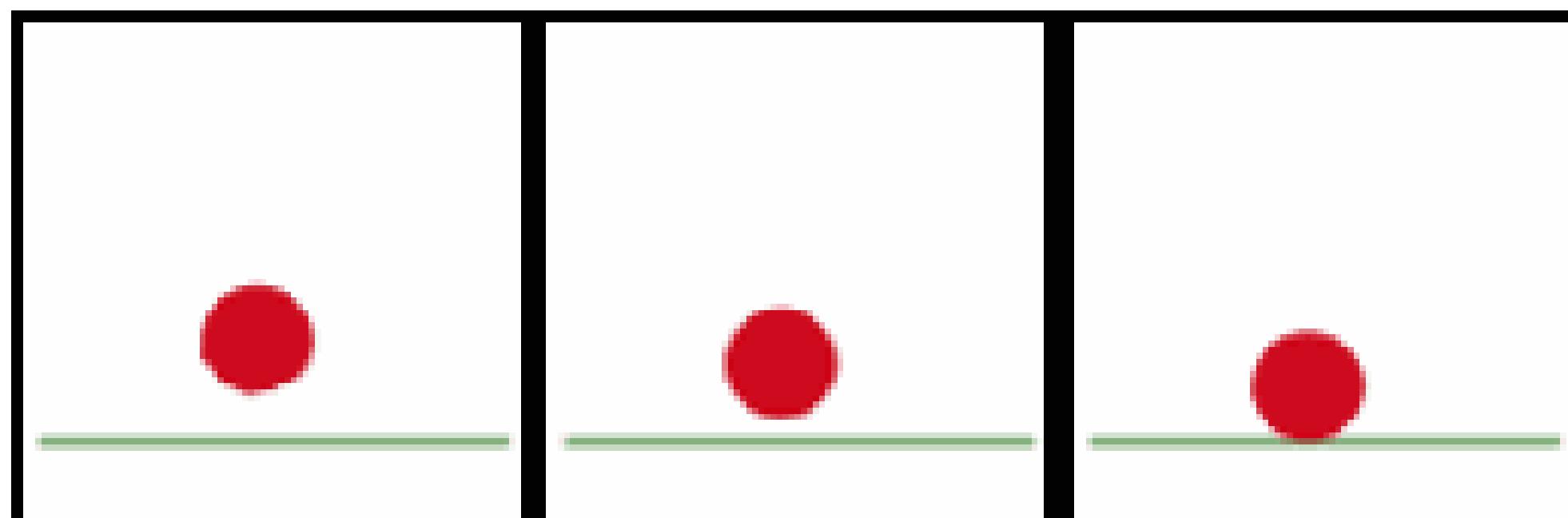
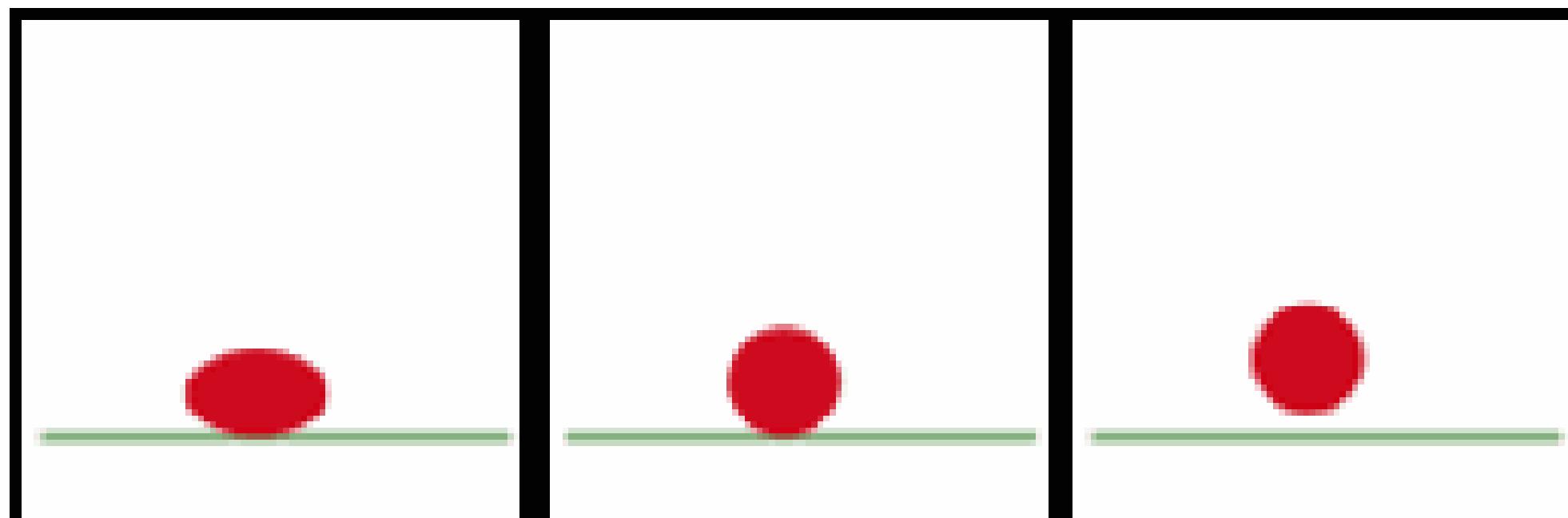
# Increasing the complexity of our models

Motion: an essential ingredient in making virtual worlds come to life!



# Animation

- **Creating the illusion of motion**
  - e.g. by rapidly displaying of a sequence of static images that minimally differ from each other.



# Early Animation

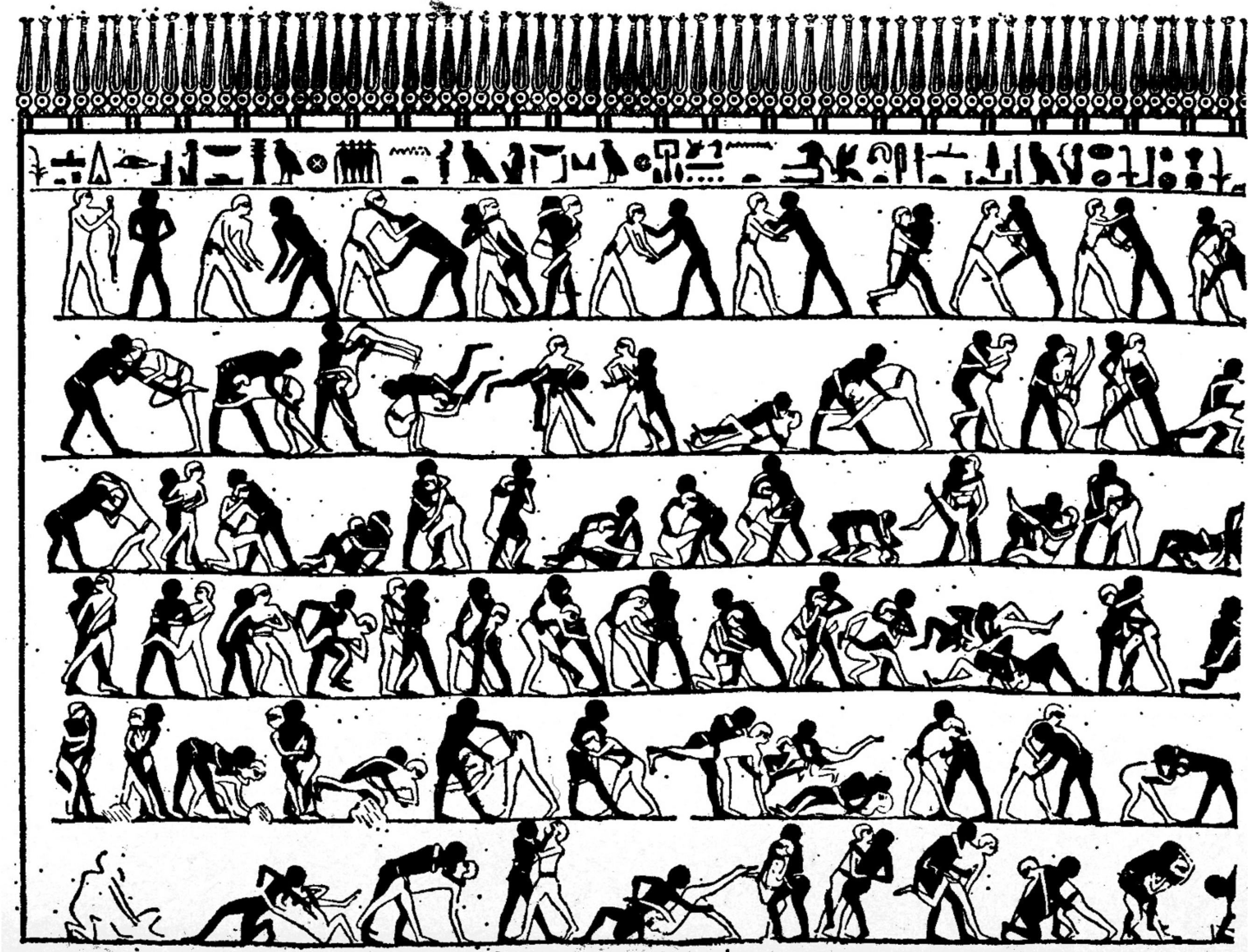


# Early Animation



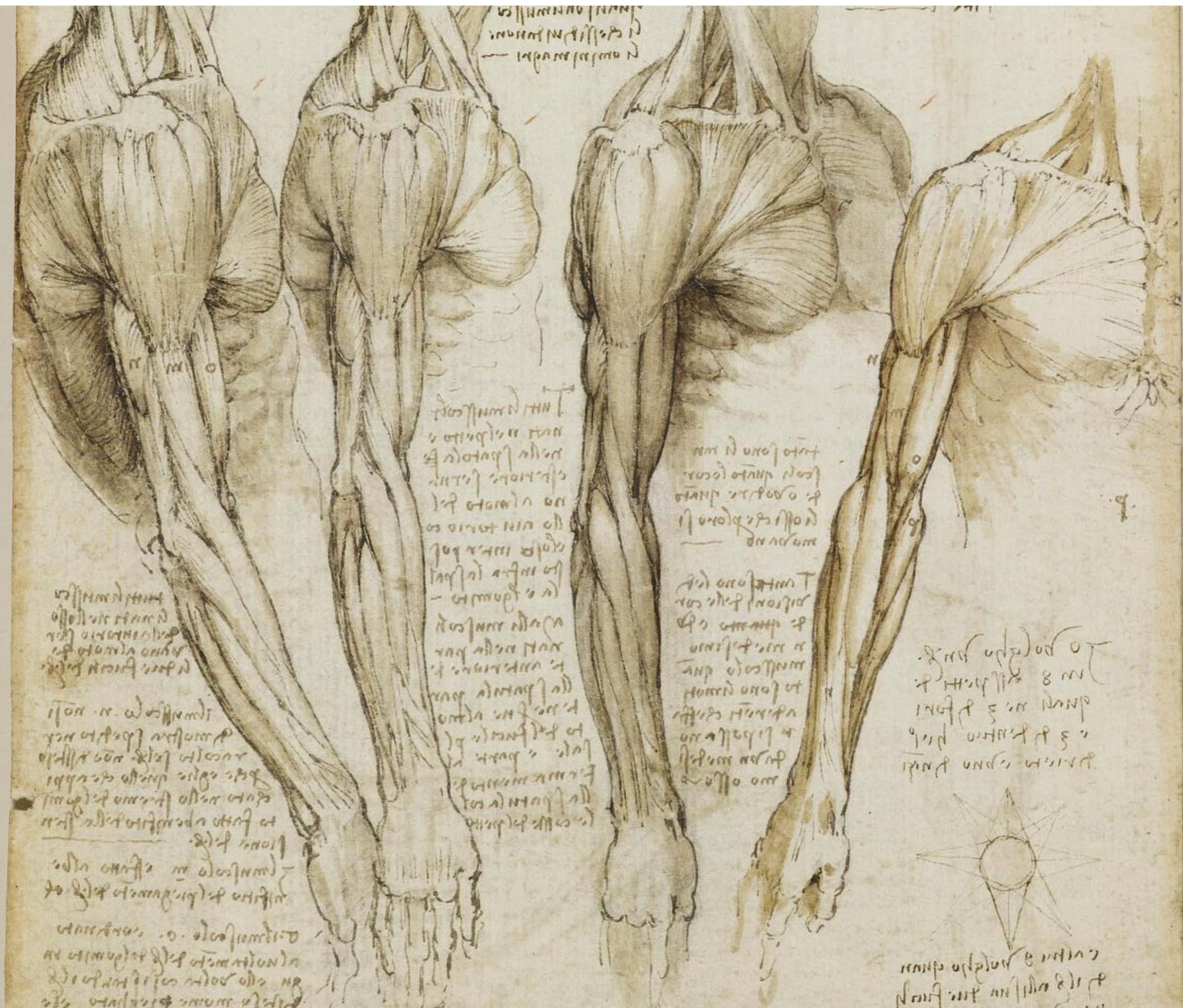
(Shahr-e Sukteh, Iran 3200 BCE)

# Early Animation



(tomb of Khnumhotep, Egypt 2400 BCE)

# Early Animation



Leonardo da Vinci (1510)

# Early Animation

- **16<sup>th</sup> century Mechanical Monk**



<https://www.youtube.com/watch?v=kie96iTq5M>

# Modern day automatons



<https://www.youtube.com/watch?v=B4J9TBIFxAg>

# Early Animation



(Phenakistoscope, 1831)

# Zoetrope - 3D Printed Animation



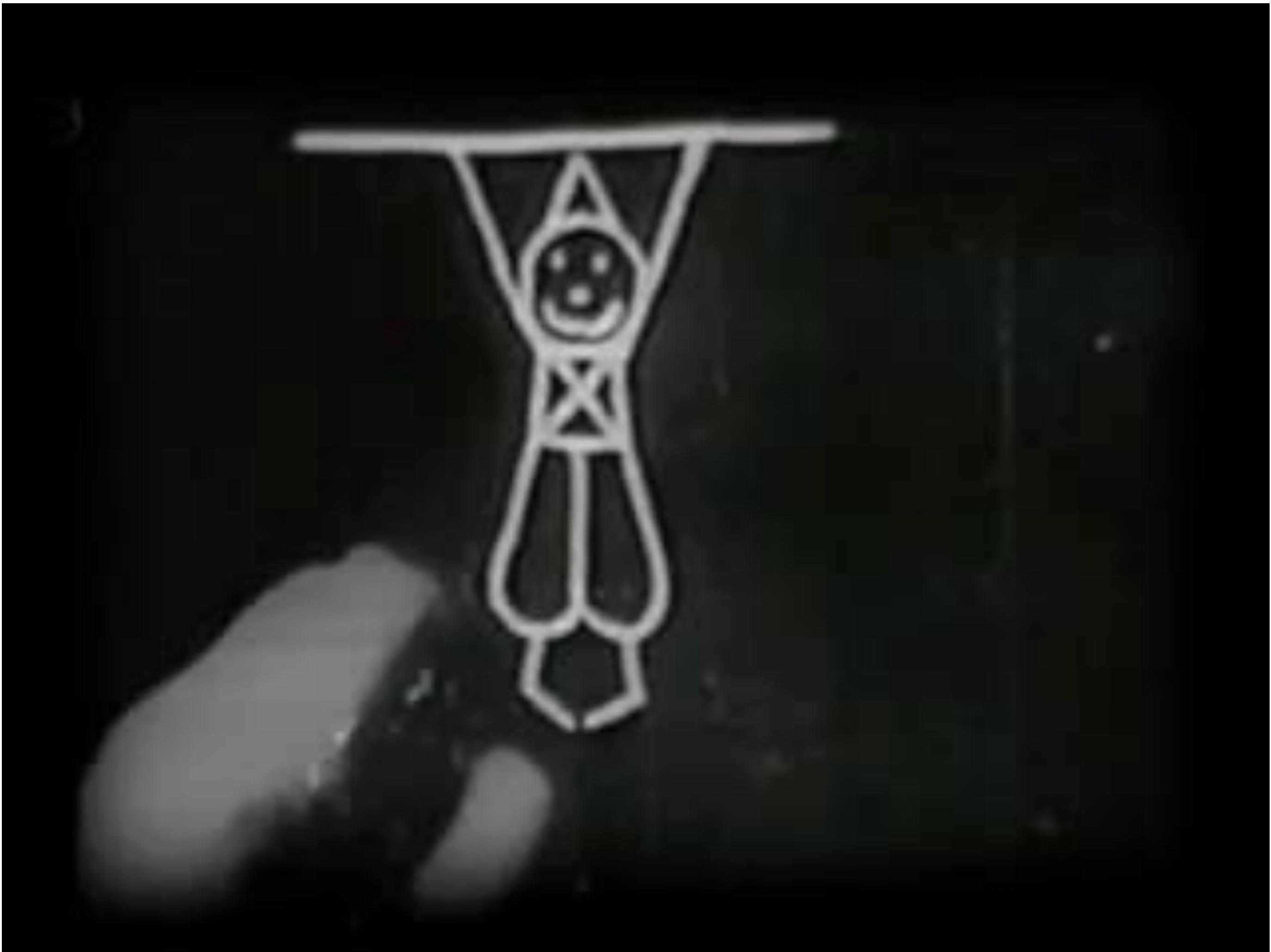
# First Film

- Originally used as scientific tool rather than for entertainment
- Critical *technology* that accelerated development of animation



Eadweard Muybridge, “*Sallie Gardner*” (1878)

# First Animation on Film

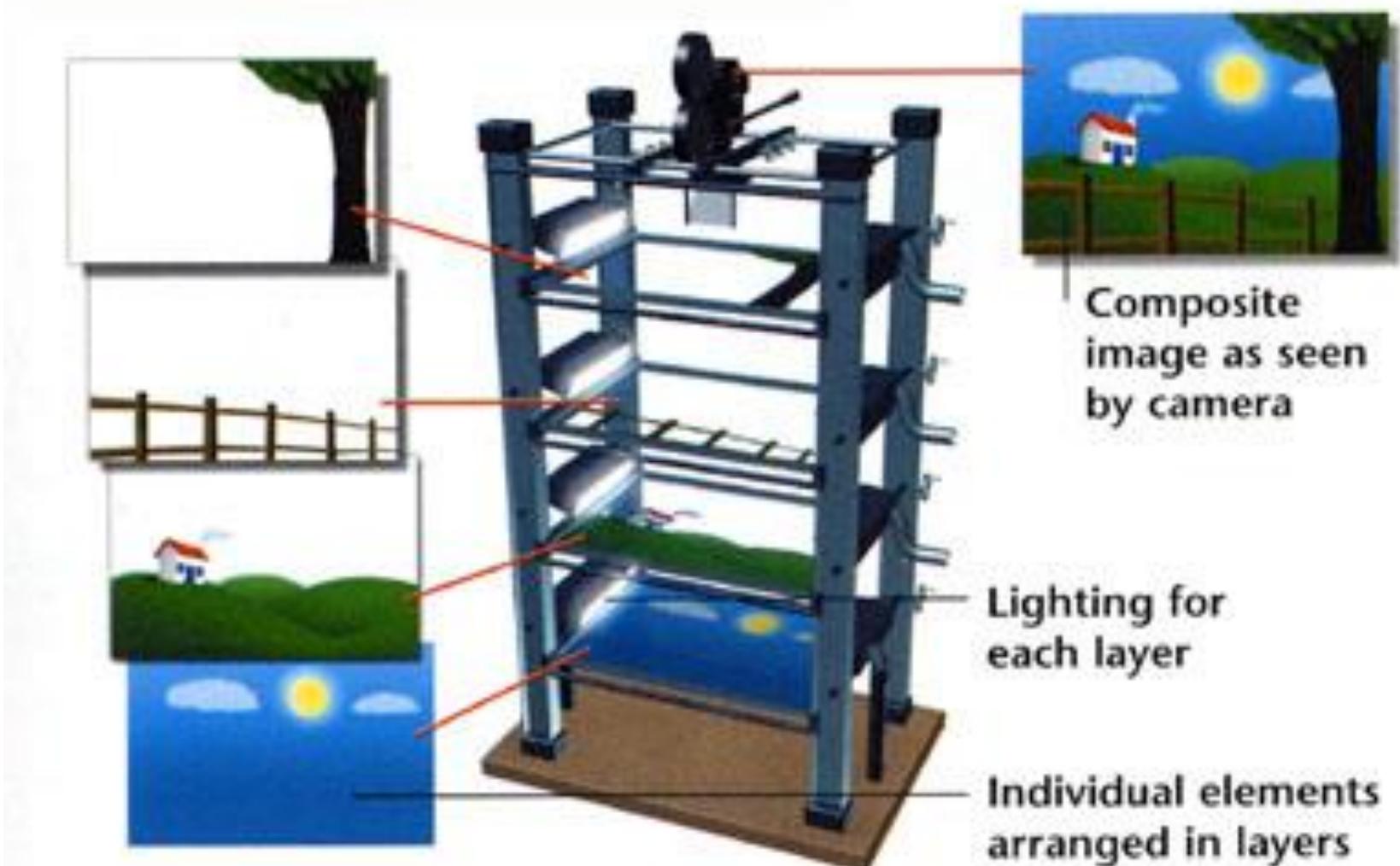


Emile Cohl, “Fantasmagorie” (1908)

# First Hand-Drawn Feature-Length Animation



Disney, “Snow White and the Seven Dwarves” (1937)

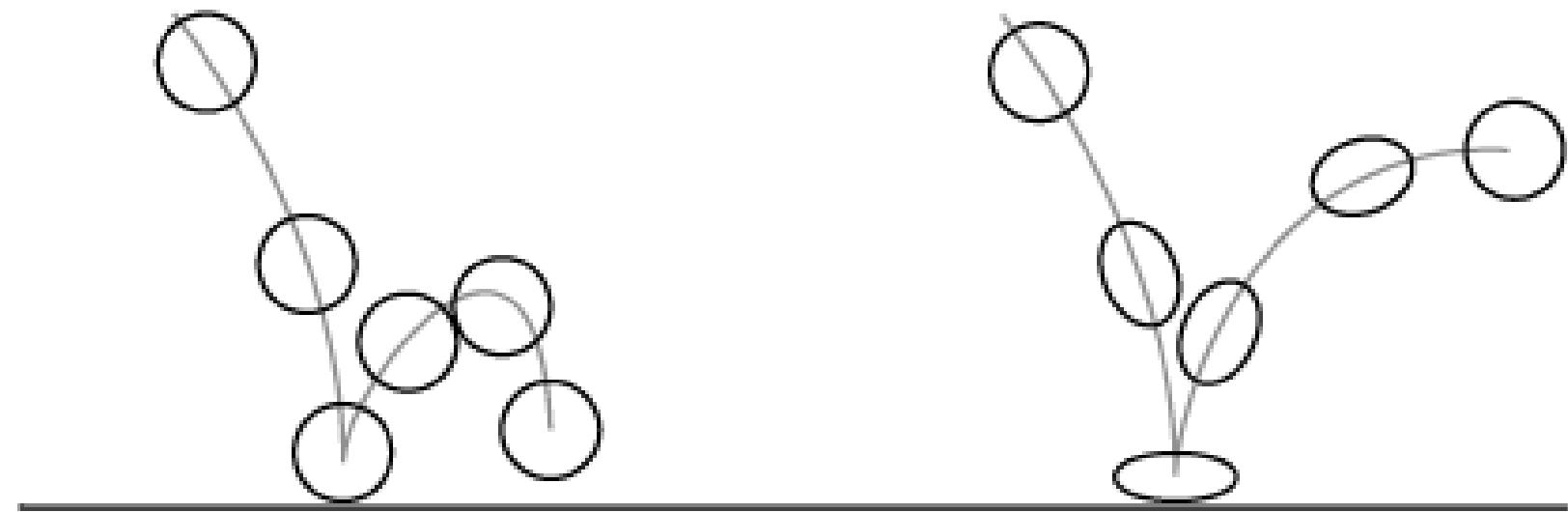


[https://www.youtube.com  
/watch?v=YdHTIUGN1zw](https://www.youtube.com/watch?v=YdHTIUGN1zw)

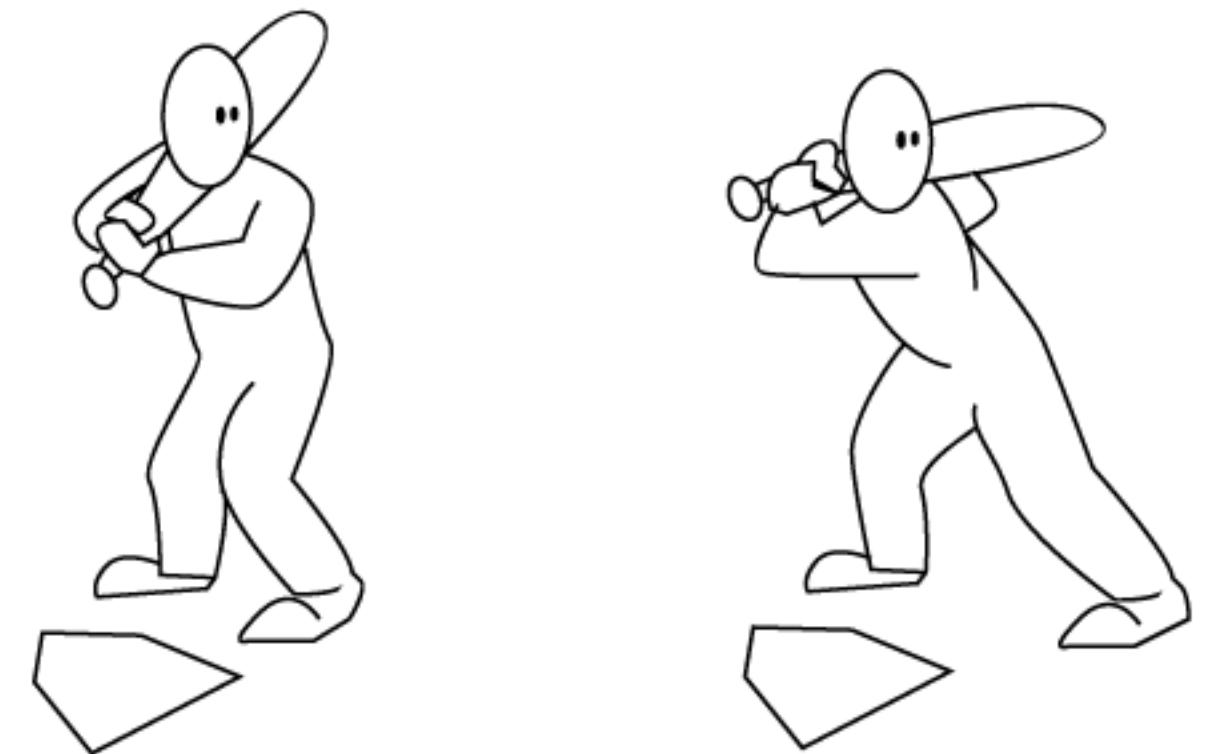
# 12 Principles of animation

- 1) **Squash and Stretch** - defining the rigidity & mass of an object by distorting its shape during an action.
- 2) **Timing** - spacing actions to define the weight & size of objects & the personality of characters.
- 3) **Anticipation** - The preparation for an action.
- 4) **Staging** - Presenting an idea so that it is unmistakably clear.
- 5) **Follow Through & Overlapping Action** - The termination of an action & establishing its relationship to the next action.
- 6) **Straight Ahead Action & Pose-To-Pose Action** - The two contrasting approaches to the creation of movement.
- 7) **Slow In and Out** - The spacing of in-between frames to achieve subtlety of timing & movements.
- 8) **Arcs** - The visual path of action for natural movement.
- 9) **Exaggeration** - Accentuating the essence of an idea via the design & the action.
- 10) **Secondary Action** - The Action of an object resulting from another action
- 11) **Appeal** - Creating a design or an action that the audience enjoys watching.
- 12) **Solid Drawing** - Knowing them can dramatically improve one's ability to create good, strong poses and compose them with well crafted environments.

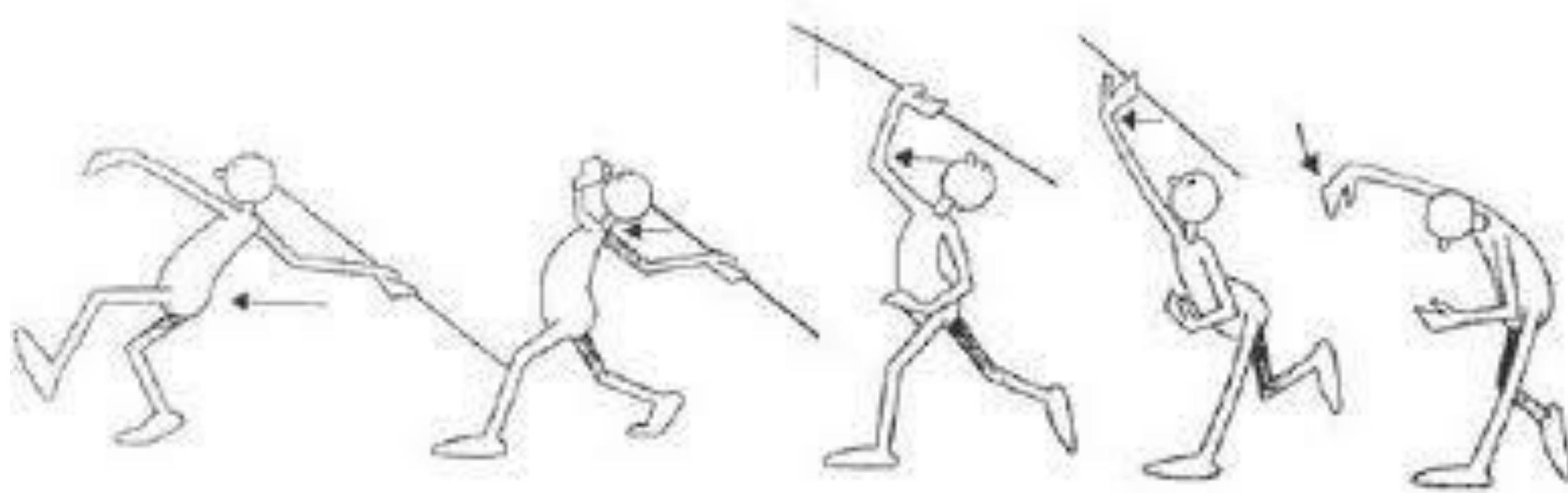
# 12 Principles of animation



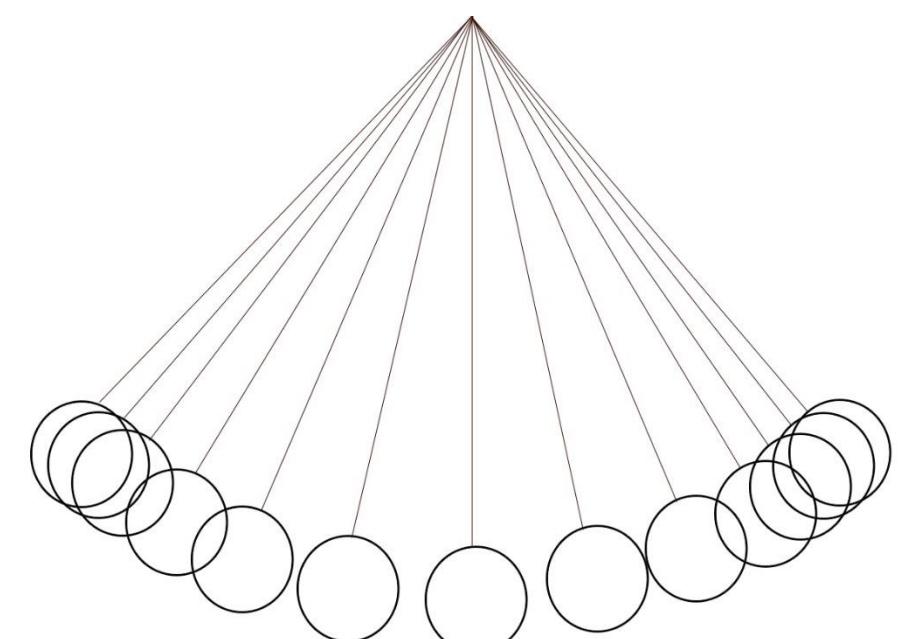
Squash and Stretch



Anticipation



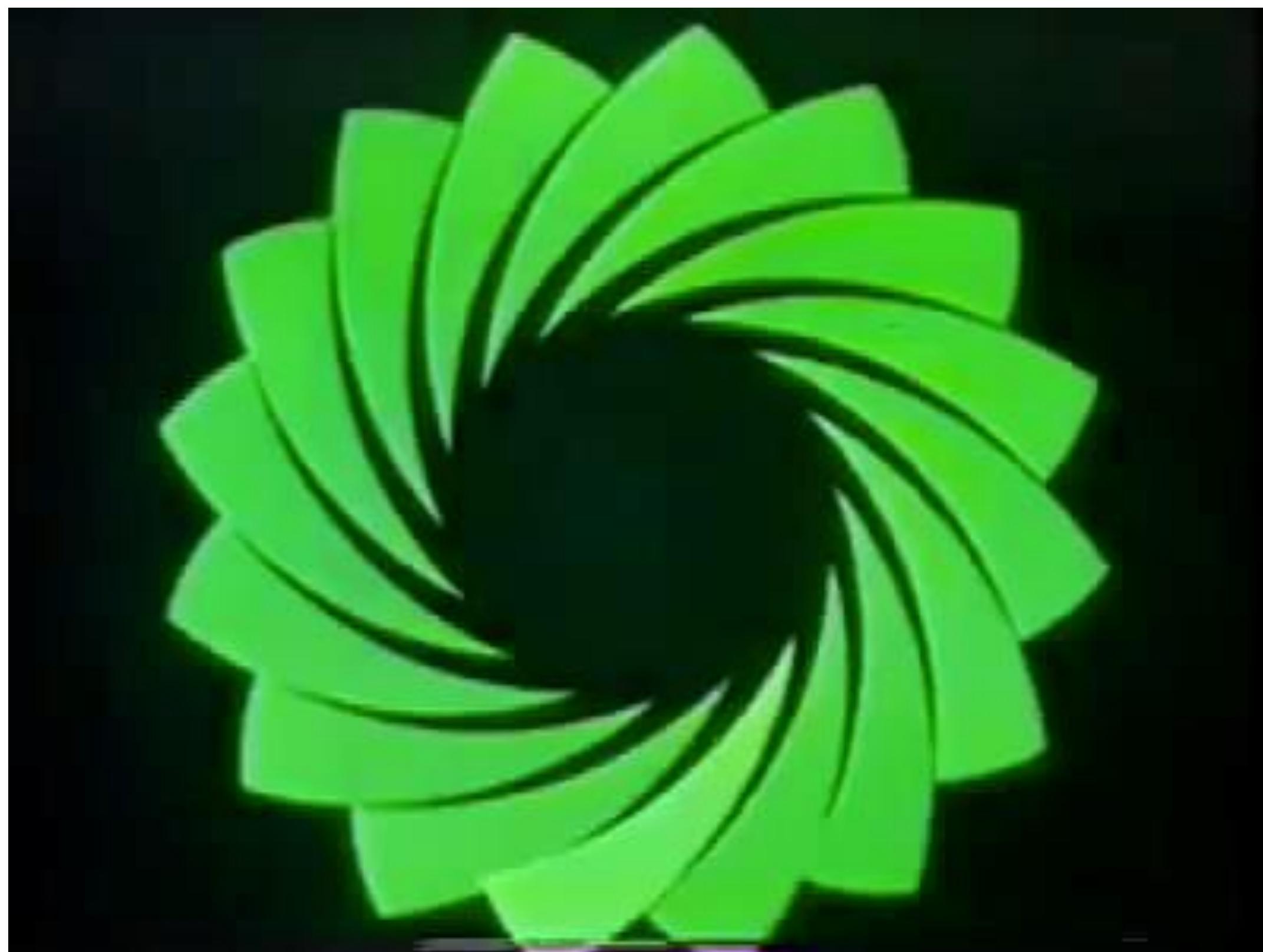
Follow through



Slow In, Slow Out

# First Computer-Generated Animation

- New *technology*, also developed as a scientific tool
- Again turbo-charged the development of animation



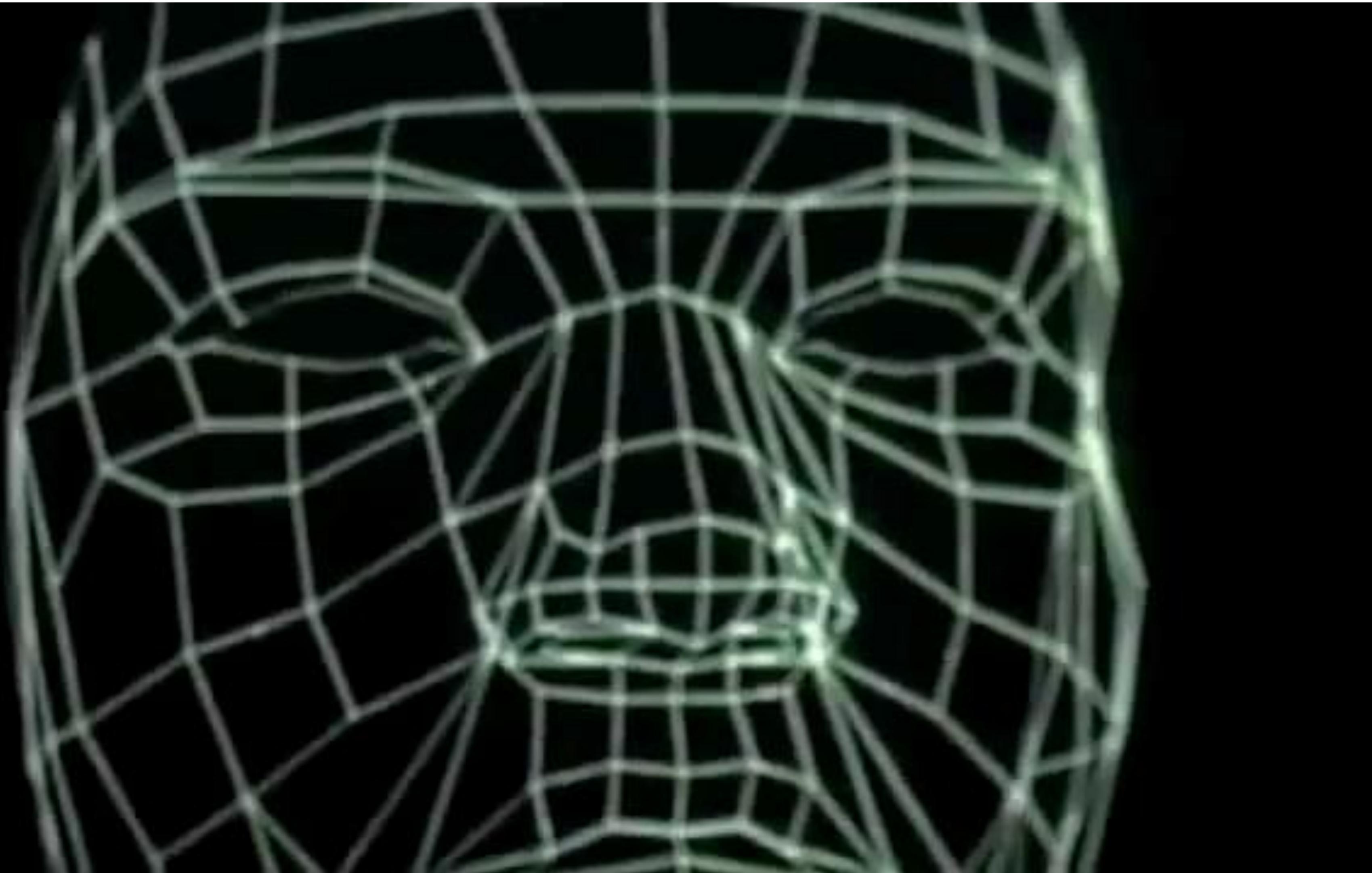
John Whitney, “Catalog” (1961)

# First Digital-Computer-Generated Animation



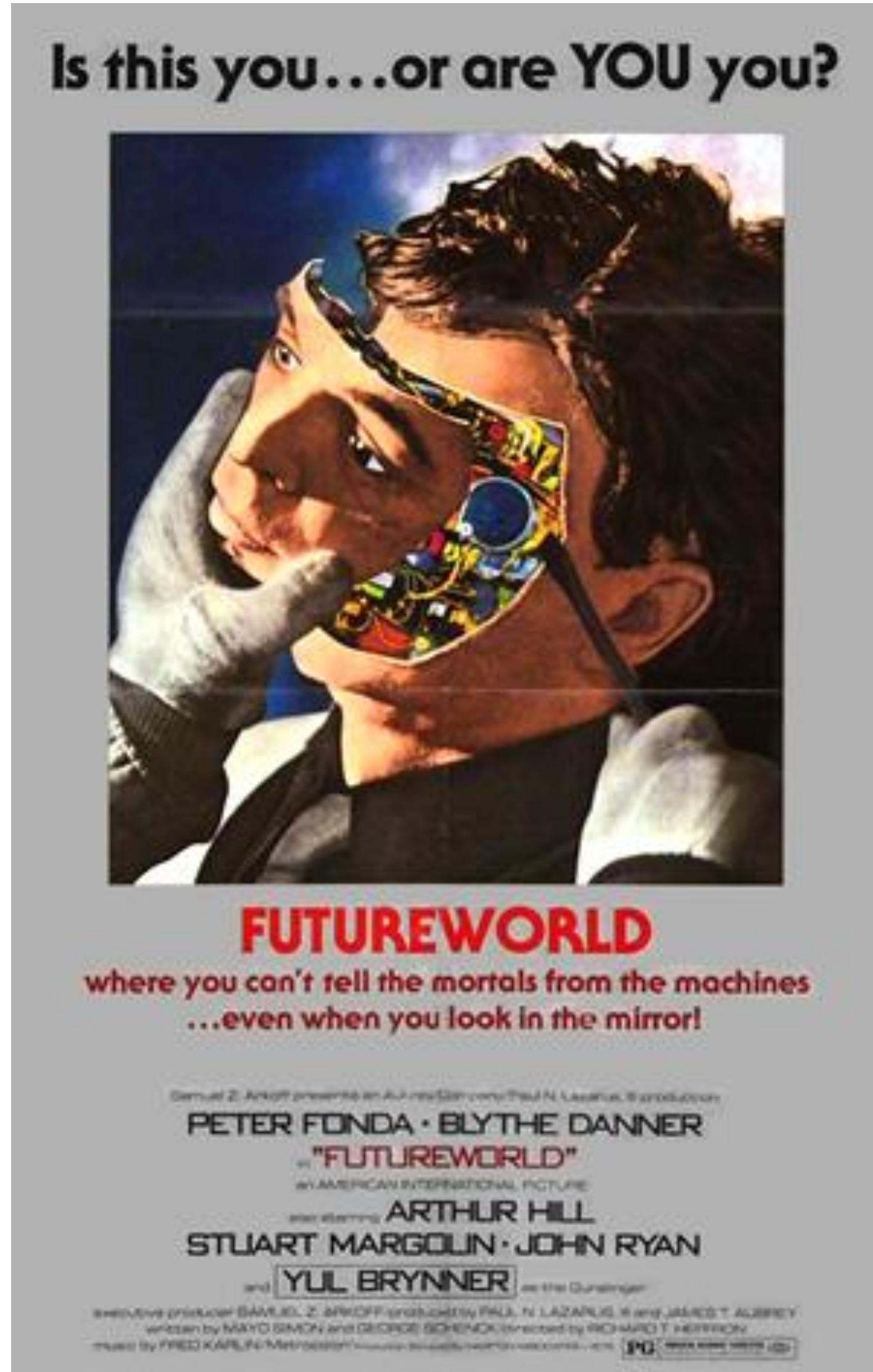
Ivan Sutherland, “Sketchpad” (1963)

# Early Computer Animation



Ed Catmull & Fred Park, “Computer Animated Faces” (1972)

# First major feature film to use CGI



Ed Catmull & Fred Park worked on “Futureworld” (1976)

# First Attempted CG Feature Film



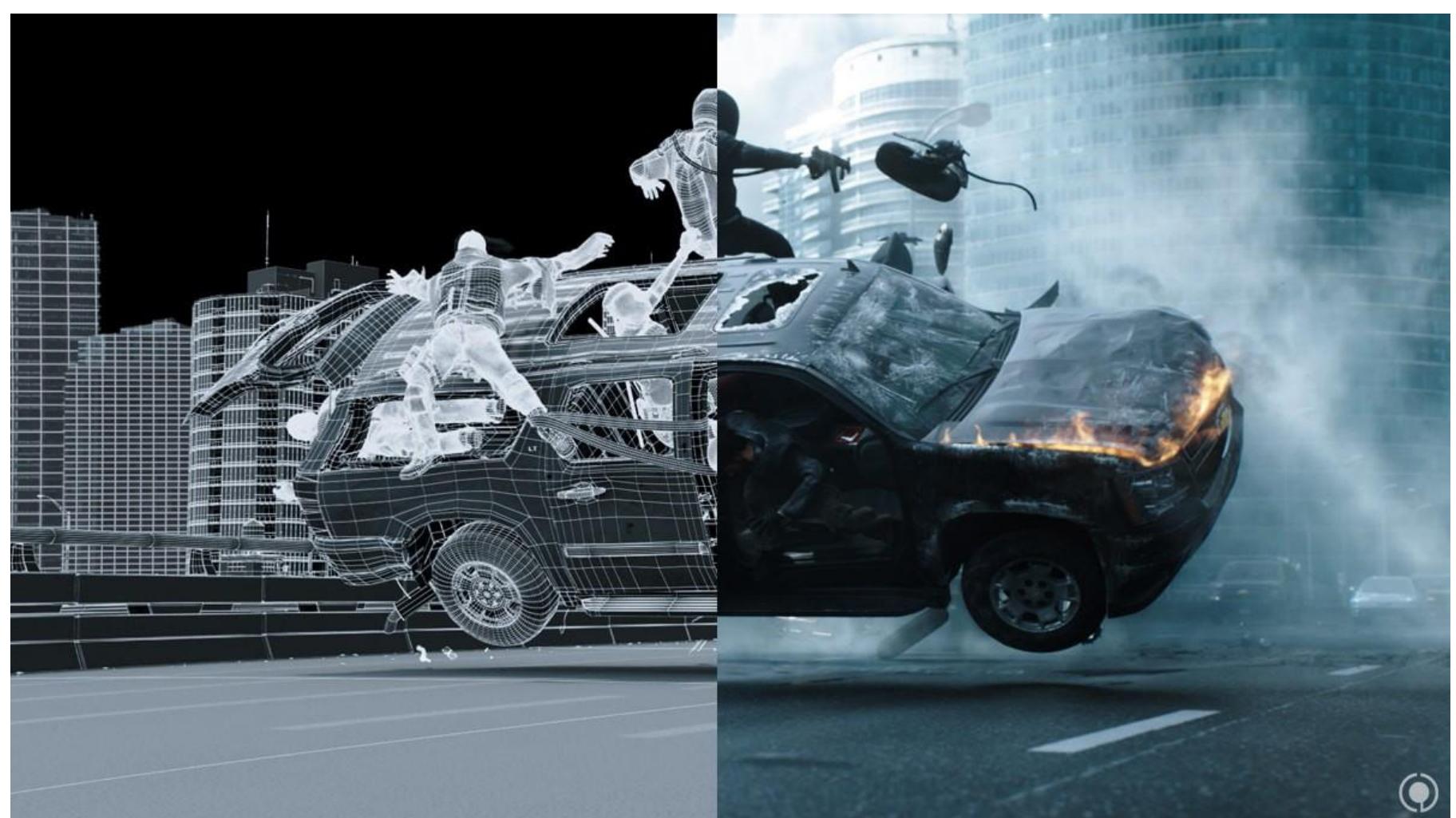
NYIT [Williams, Heckbert, Catmull, ...], "The Works" (1984)

# First CG Feature Film



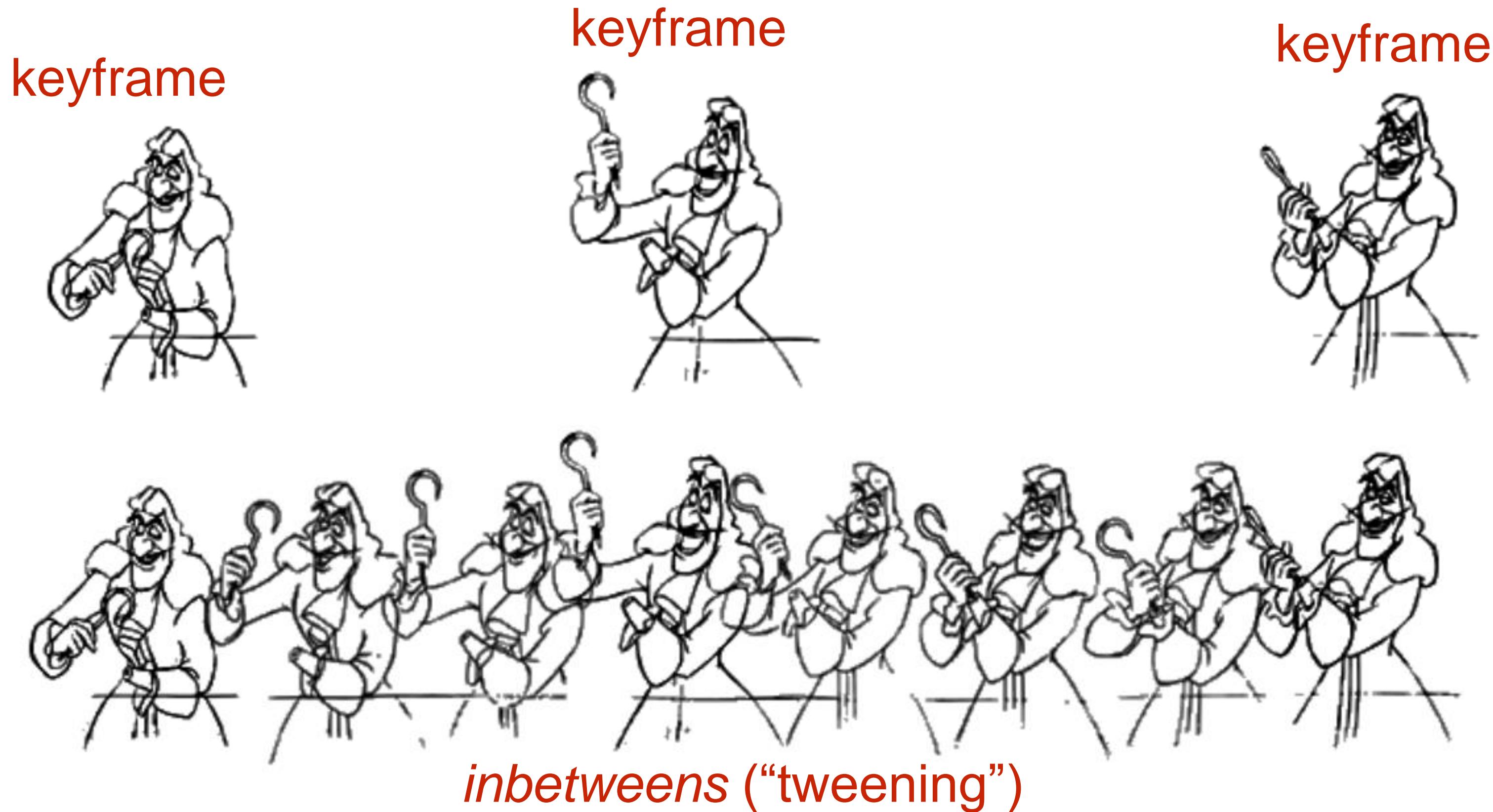
Pixar, “Toy Story” (1995)

# Computer Animation - Present Day



# Generating Motion (Hand-Drawn)

- Senior artist draws *keyframes*
- Assistant draws *inbetweens*
- Tedious / labor intensive (opportunity for technology!)



# Generating Motion (Stop-motion animation)

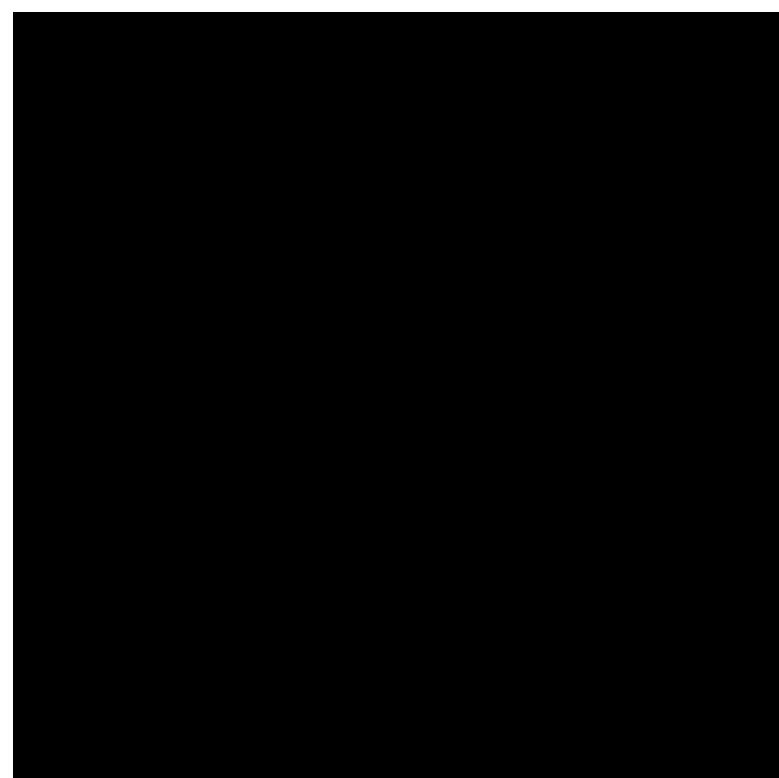
Same idea, but with physical props

- *armatures, clay models, etc...*



# Perception of Motion

- Original (but debunked) theory: *persistence of vision* (“streaking”)
- The eye is not a camera! More modern explanation:
  - *beta/phi phenomenon*: visual memory/anticipation built in brain—not eyeball

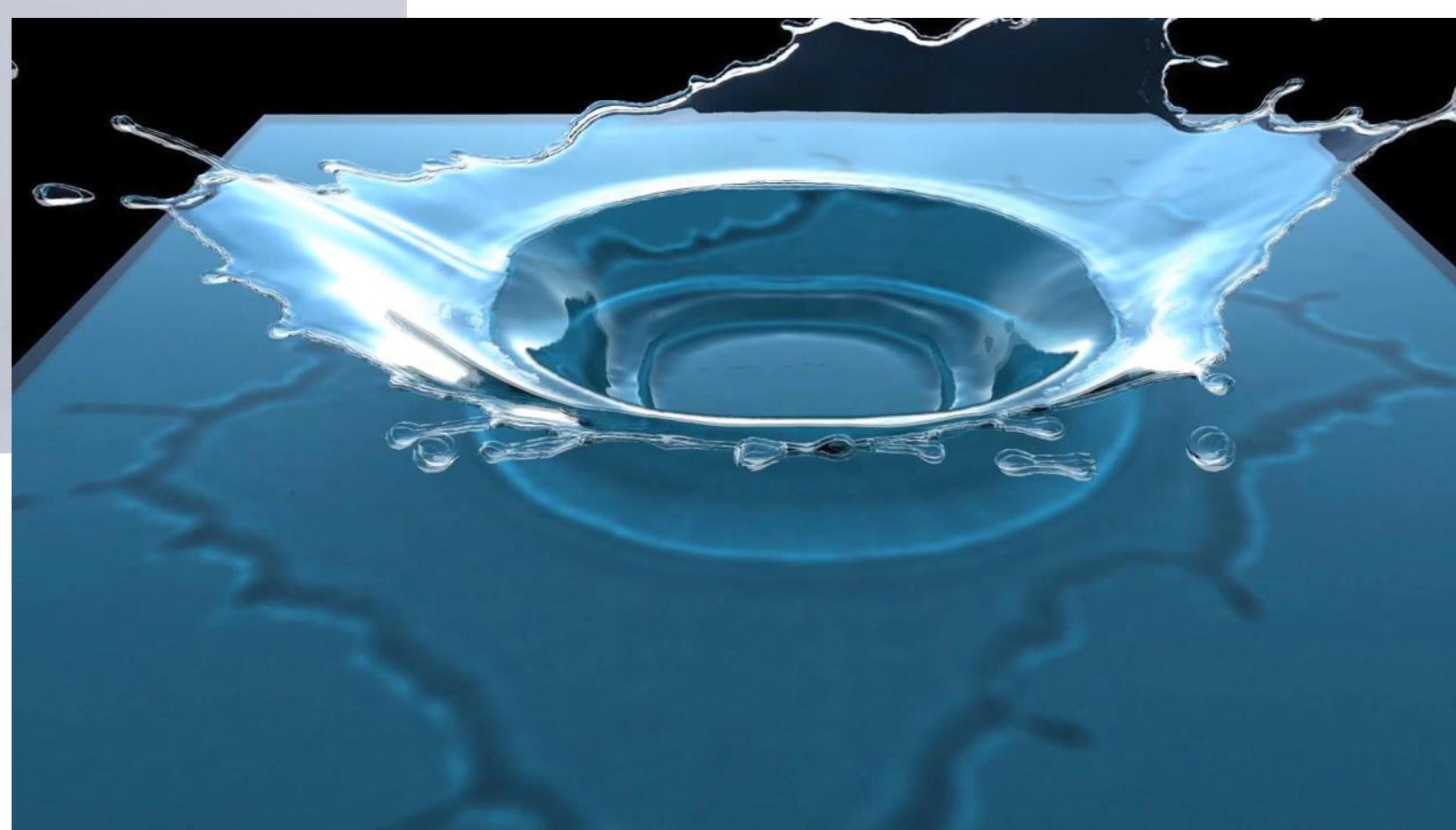
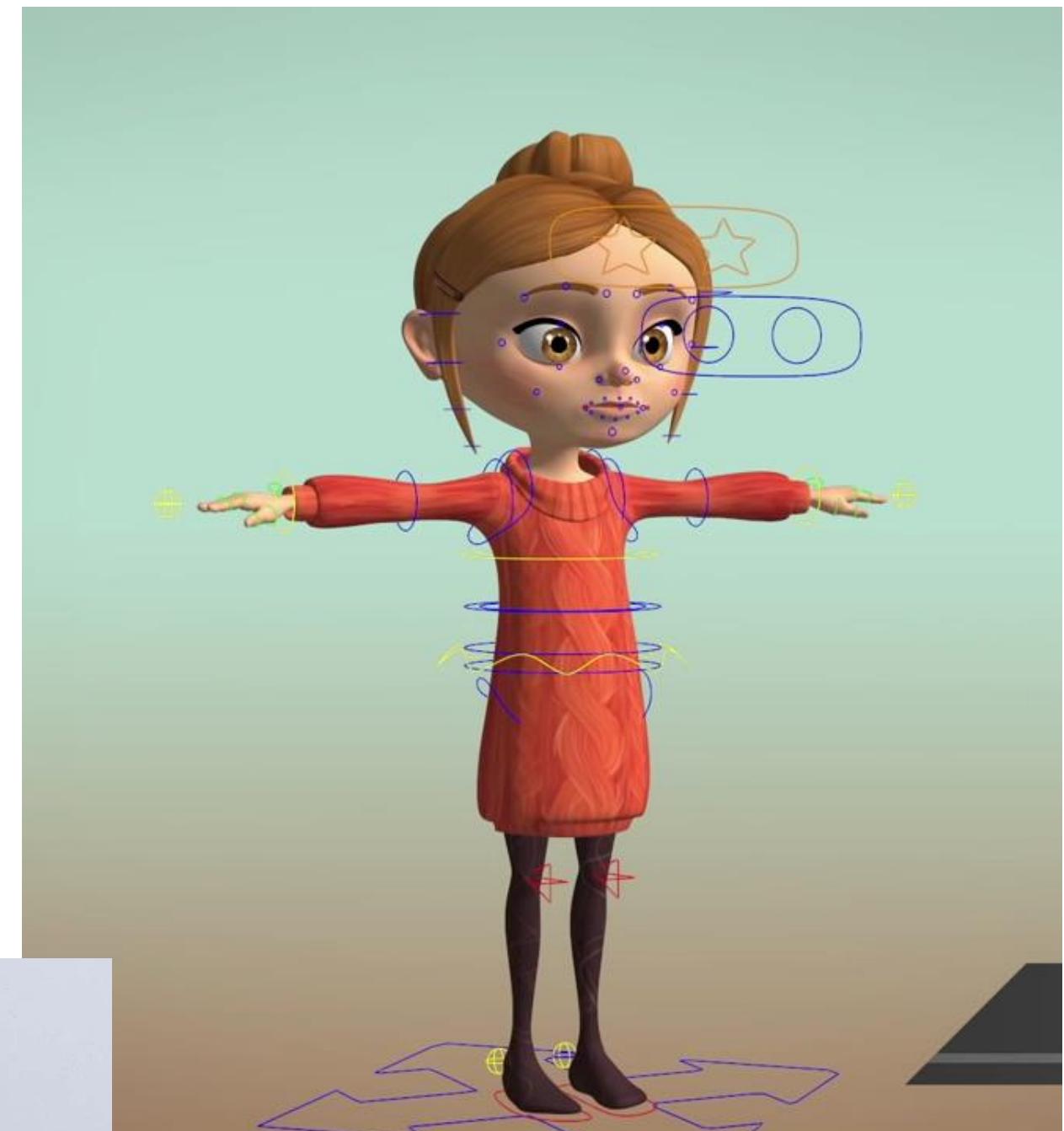


beta

**How do we describe motion on a computer?**

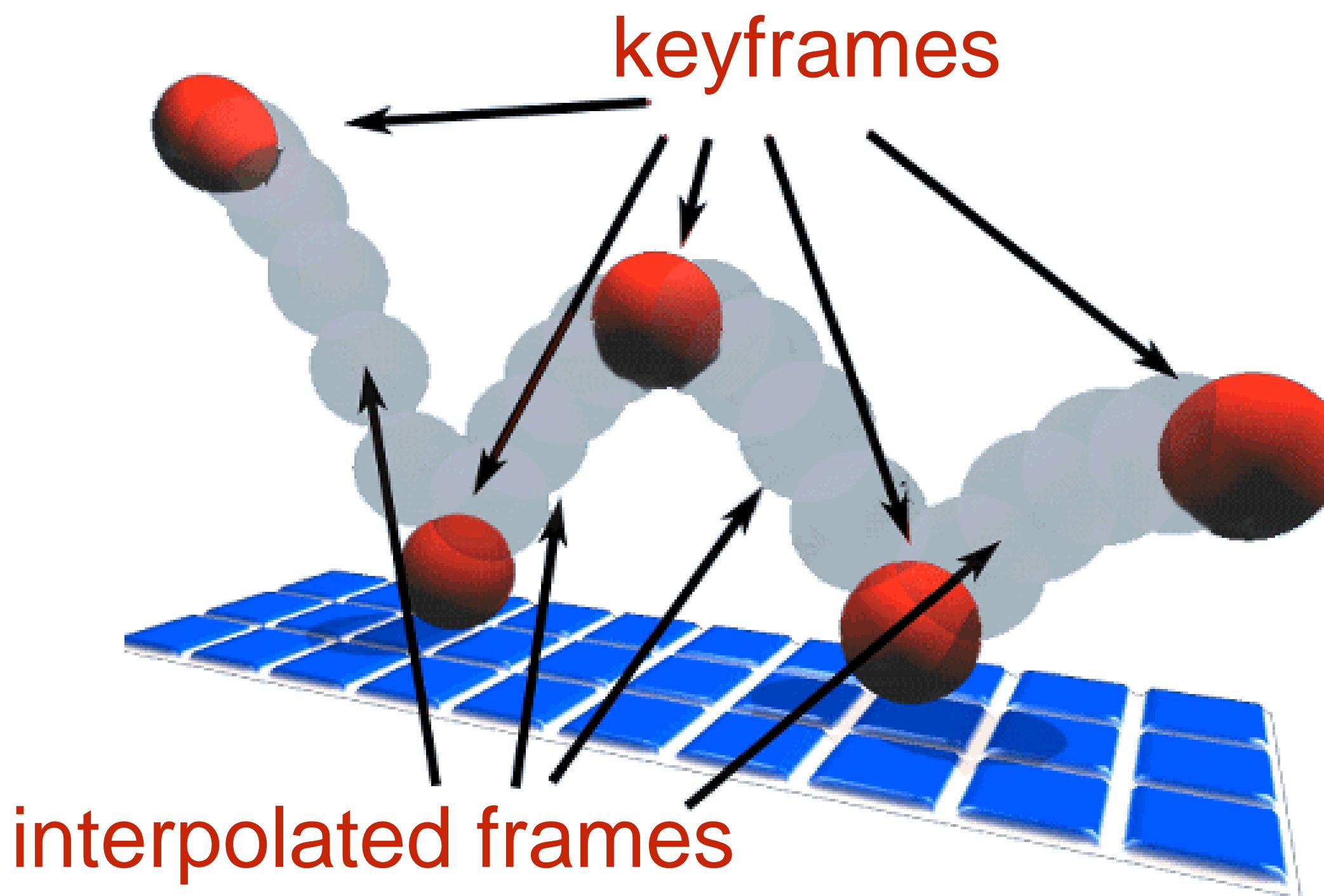
# Basic Techniques in Computer Animation

- Artist-directed (e.g., keyframing)
- Data-driven (e.g., motion capture)
- Procedural (e.g., simulation)



# Keyframing

- Basic idea:
  - specify important events only, computer fills in the rest via interpolation/approximation
- “Events” don’t have to be position
- Could be color, light intensity, camera configuration, ...

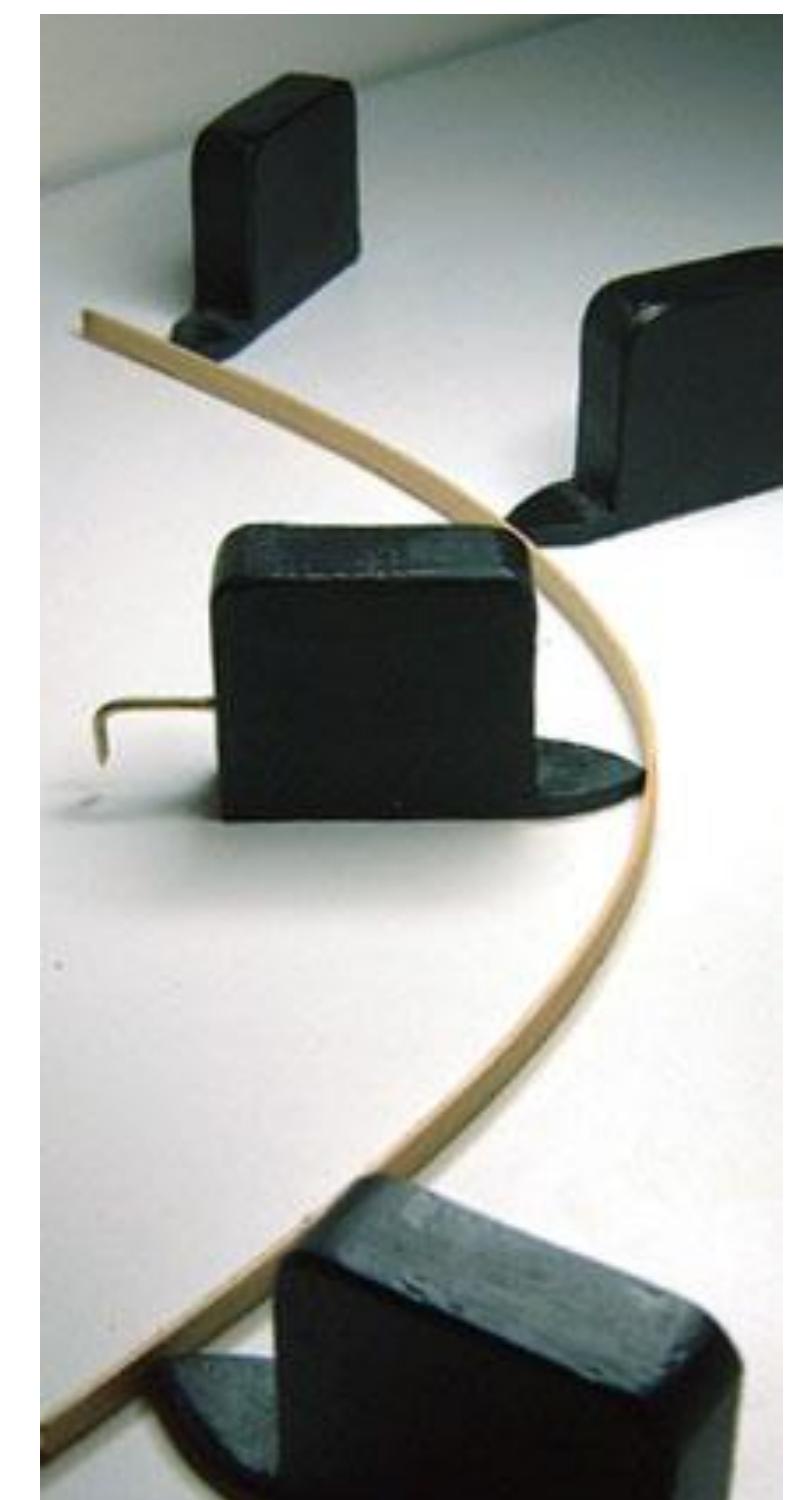
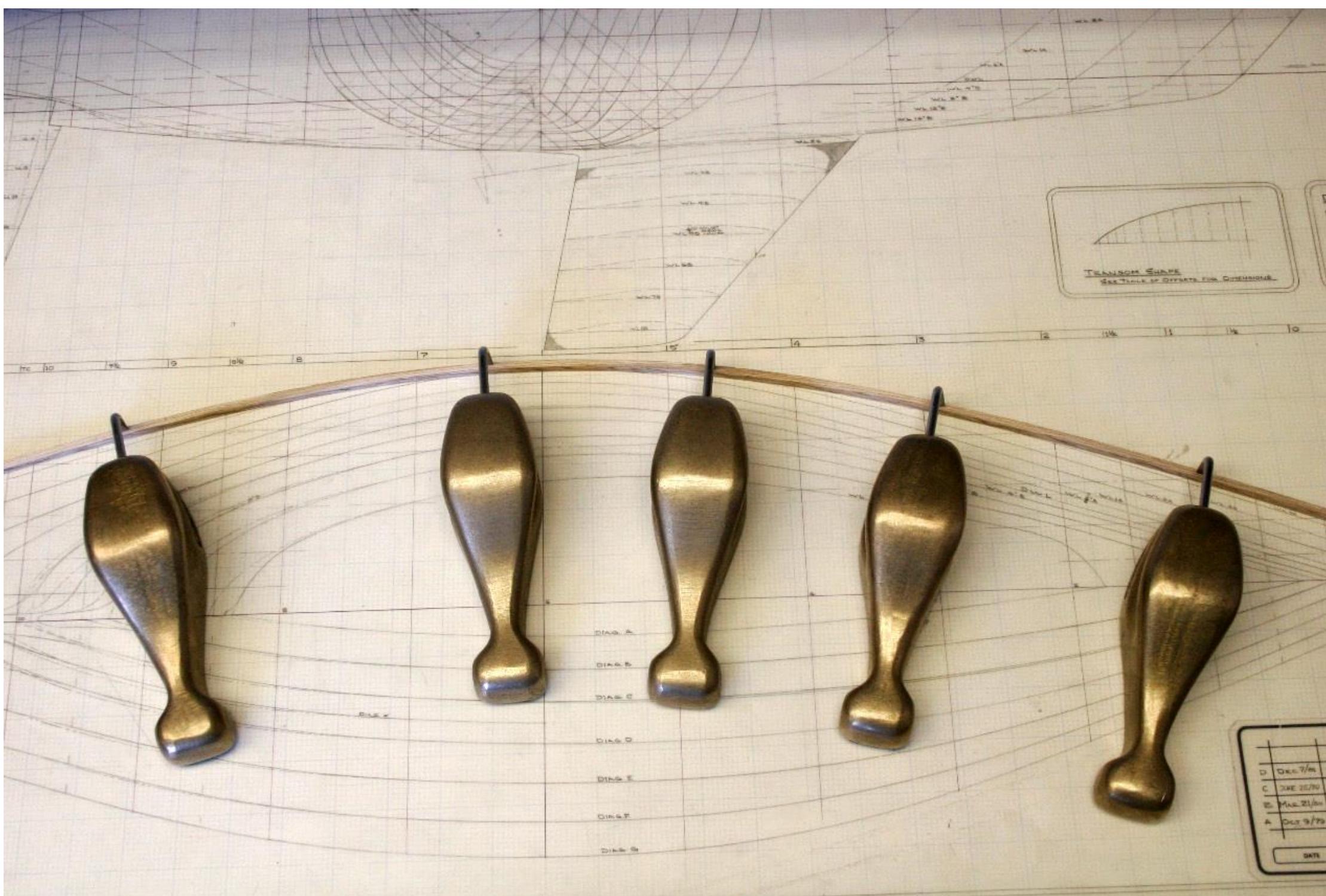


# **How do we interpolate data?**

# Spline Interpolation

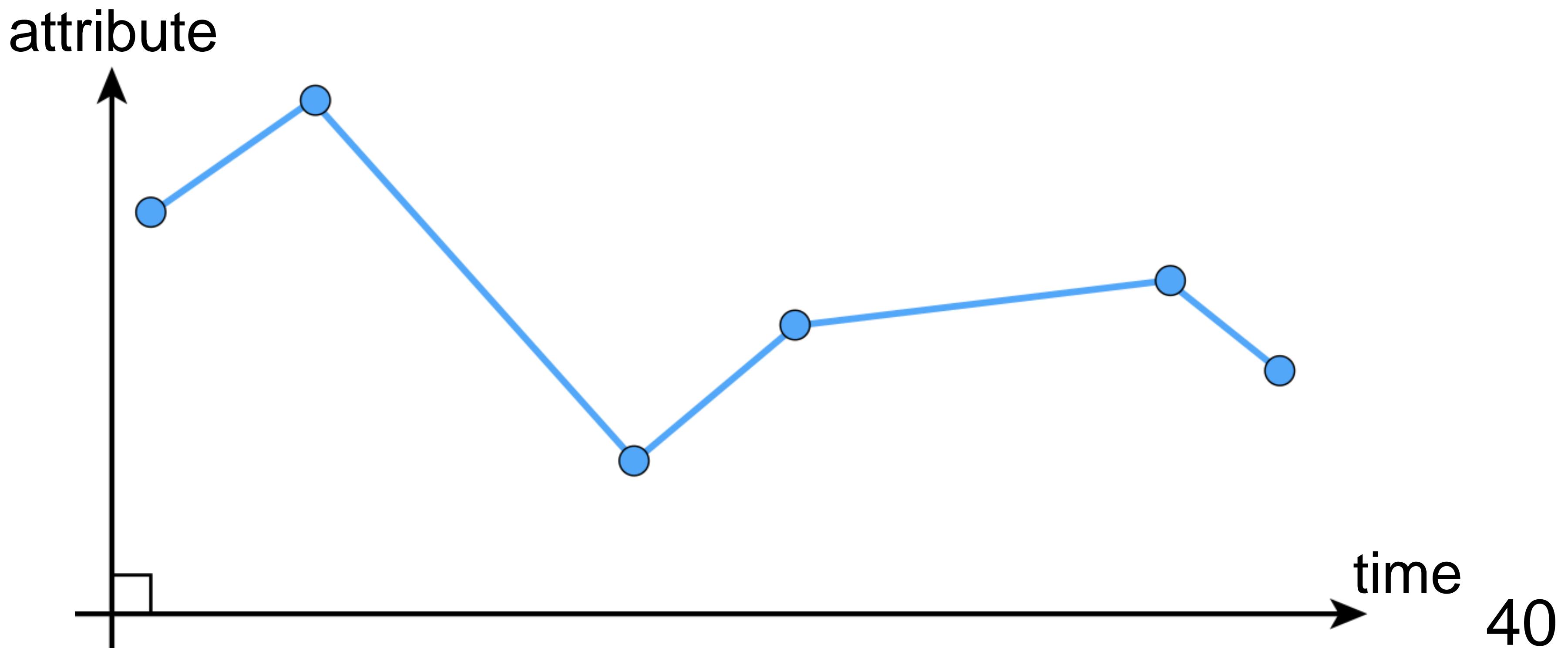
Mathematical theory of interpolation arose from study of thin strips of wood or metal (“splines”) under various forces

Good summary in Levin, “The Elastica: A Mathematical History”



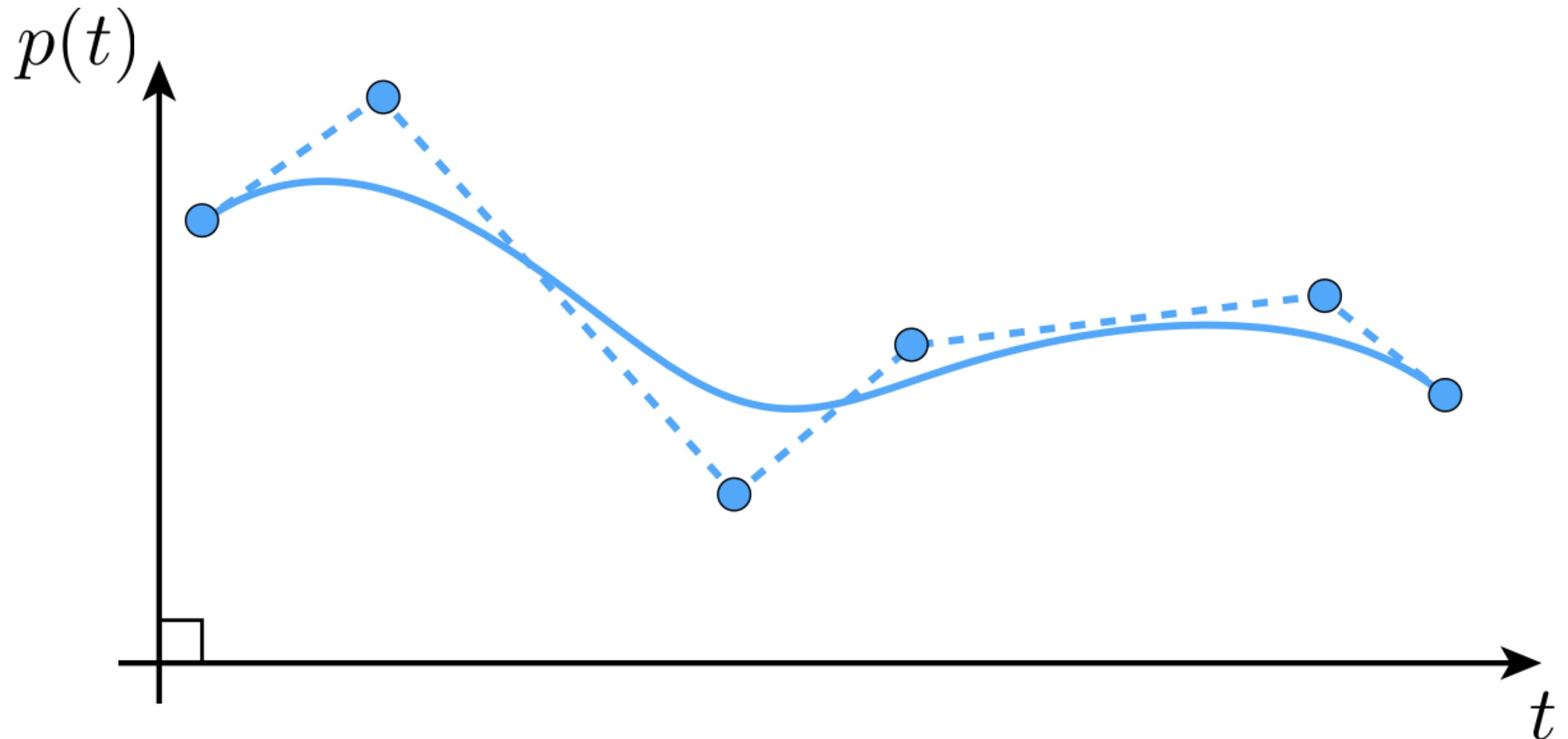
# Interpolation

- Basic idea: “connect the dots”
- E.g., *piecewise linear interpolation*
- Simple, but yields rather rough motion
  - infinite acceleration – why is this far from ideal?



# Piecewise Polynomial Interpolation

- Common interpolant: piecewise polynomial “spline”

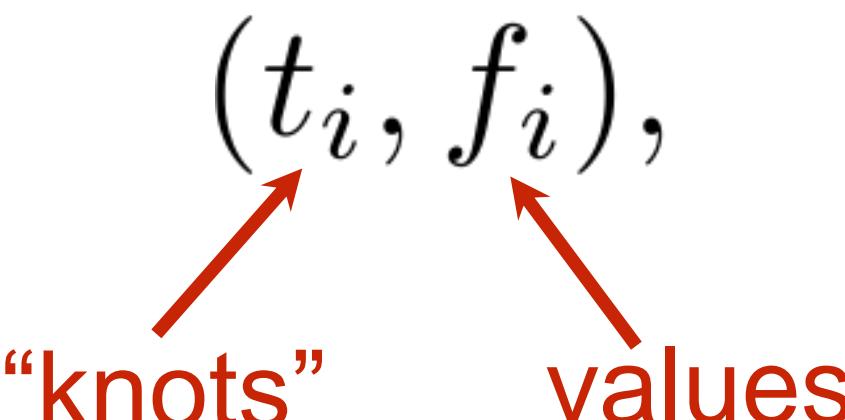


Basic motivation: get better continuity than piecewise linear!

# Splines

- In general, a *spline* is any piecewise polynomial function
- In 1D, spline interpolates data over the real line:

$$(t_i, f_i), \quad i = 0, \dots, n$$

  
“knots”                  values

$$t_i < t_{i+1}$$

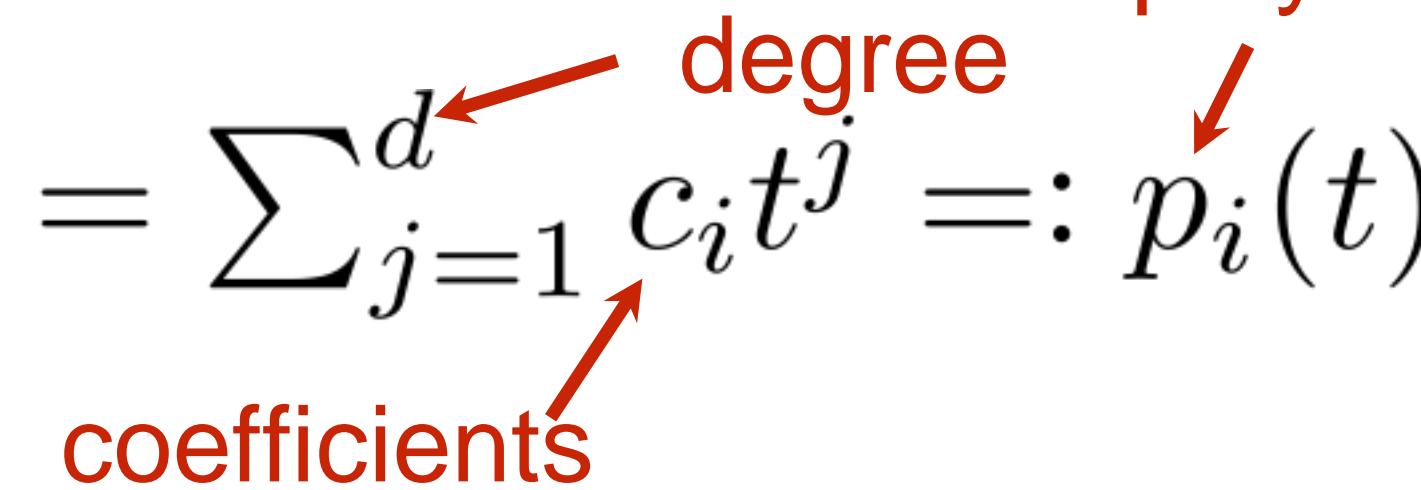
- “Interpolates” just means that the function *exactly* passes through those values:

$$f(t_i) = f_i \quad \forall i$$

- The only other condition is that the function is a *polynomial* when restricted to any interval between knots:

$$\text{for } t_i \leq t \leq t_{i+1}, f(t) = \sum_{j=1}^d c_i t^j =: p_i(t)$$

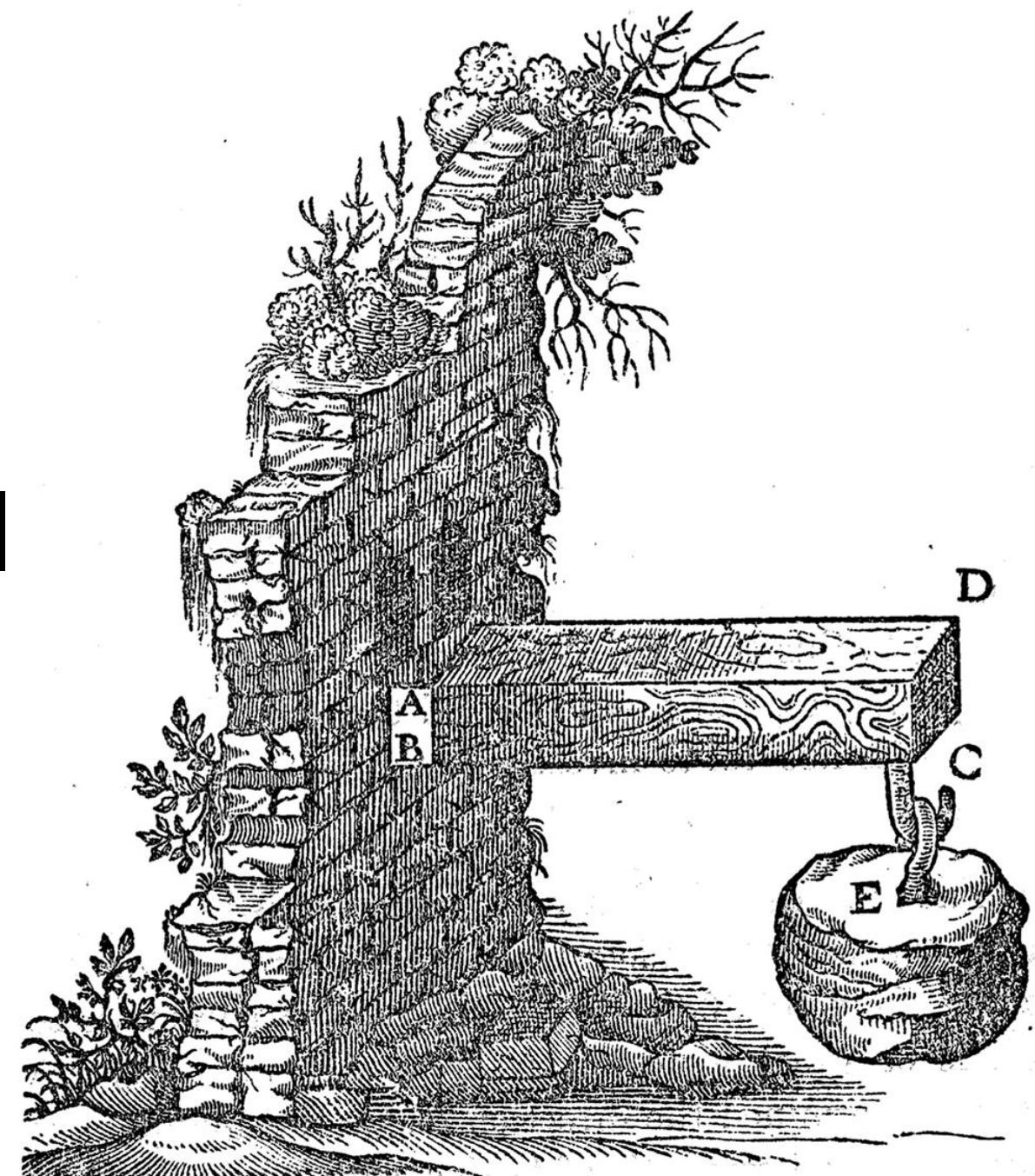
polynomial



degree  
coefficients  
polynomial

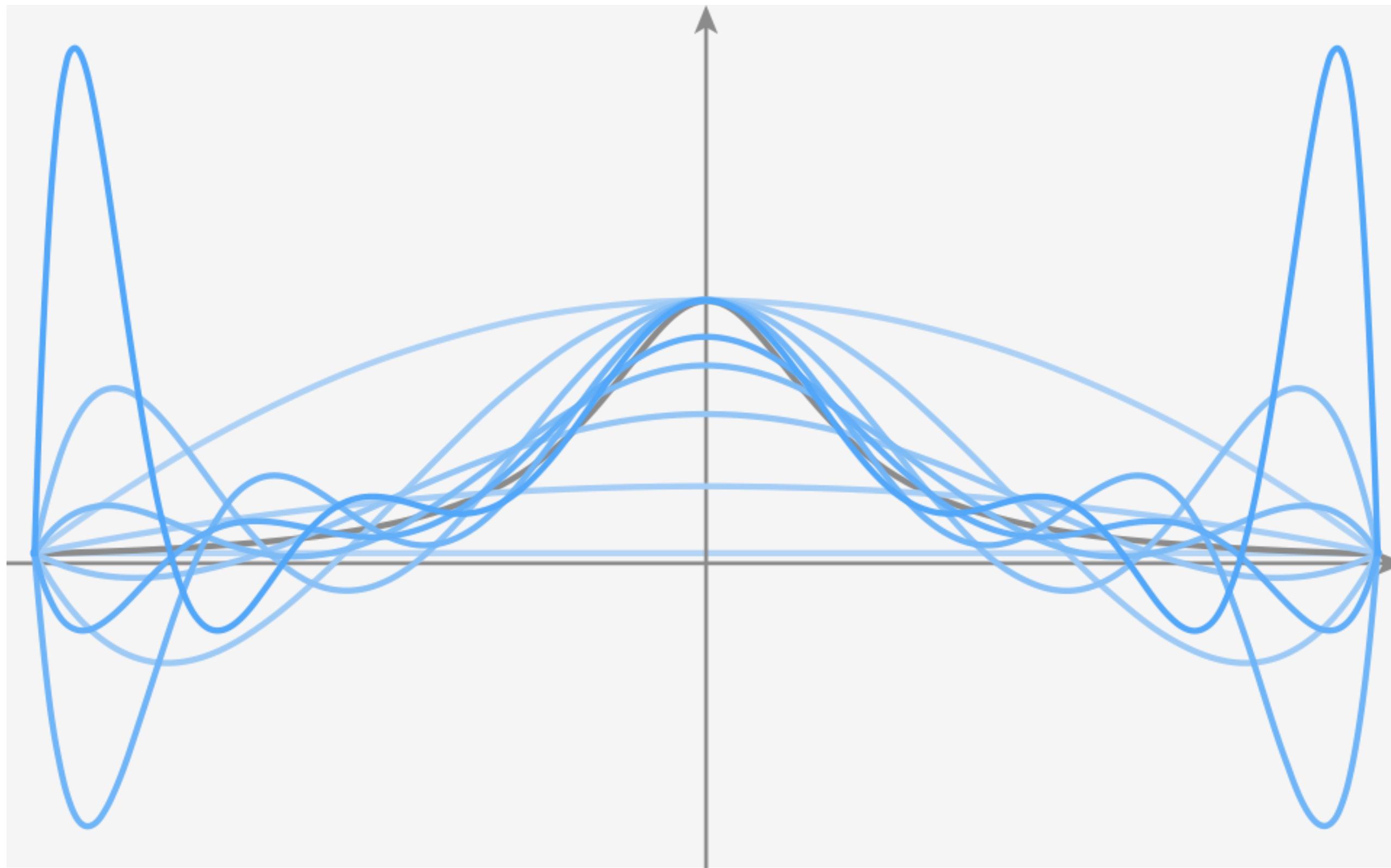
# Splines

- Splines most commonly used for interpolation are *cubic* ( $d=3$ )
- Schoenberg: piecewise cubics give exact solution to elastic spline problem under assumption of small displacements
- More precisely: among all curves interpolating set of data points, minimizes norm of second derivative
- Q: Why use model based on physical splines?



# Runge's Phenomenon

- Tempting to use higher-degree polynomials, in order to get higher-order continuity
- Can lead to oscillation, ultimately **worse approximation**:

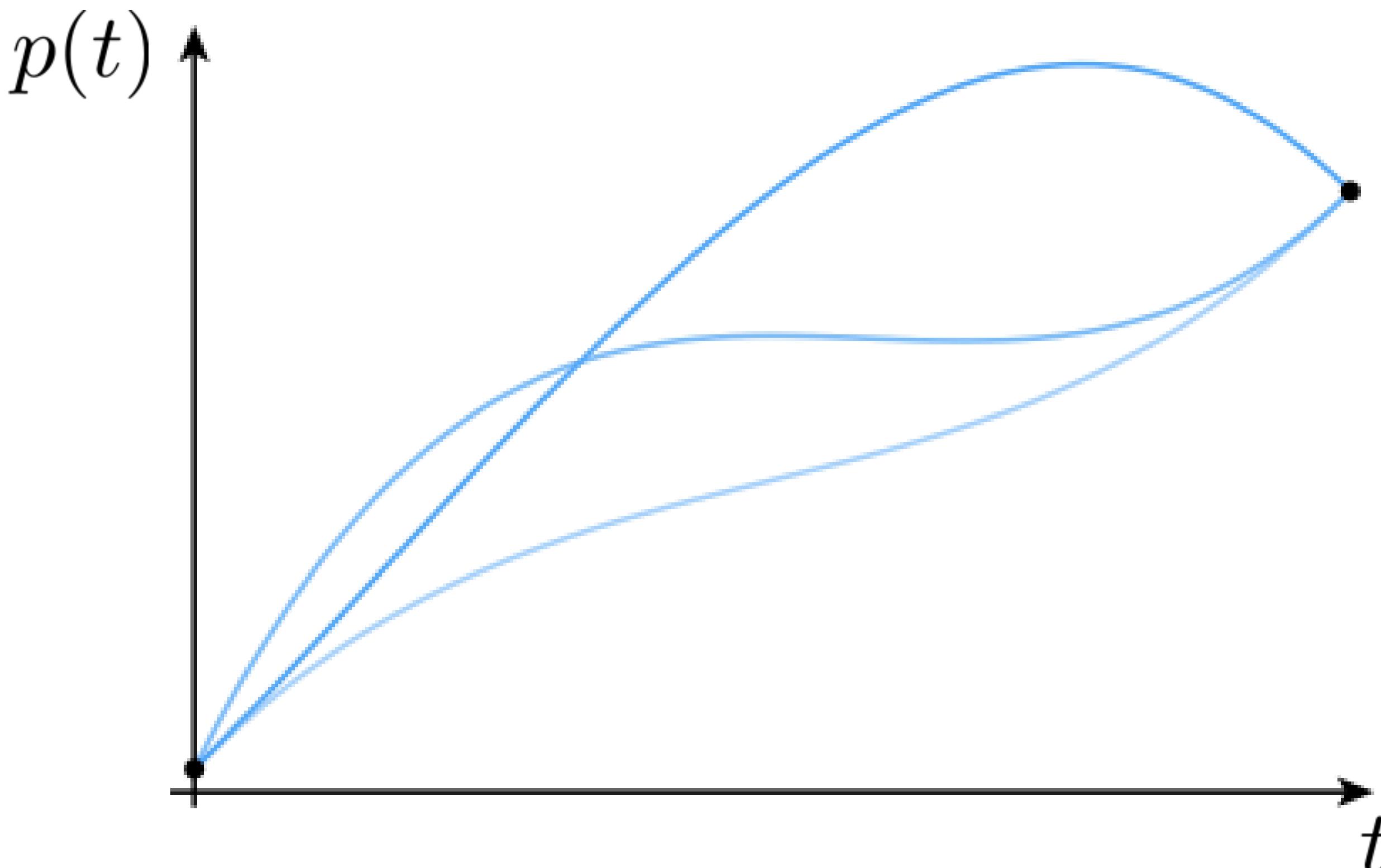


# Fitting a Cubic Polynomial to Endpoints

- Consider a *single* cubic polynomial

$$p(t) = at^3 + bt^2 + ct + d$$

- Suppose we want it to match given endpoints:



Many solutions!

# Cubic Polynomial - Degrees of Freedom

- Why are there so many different solutions?
- Cubic polynomial has four *degrees of freedom (DOFs)*, namely four coefficients (a,b,c,d) that we can manipulate/control
- Only need *two degrees of freedom to specify endpoints:*

$$p(t) = at^3 + bt^2 + ct + d$$

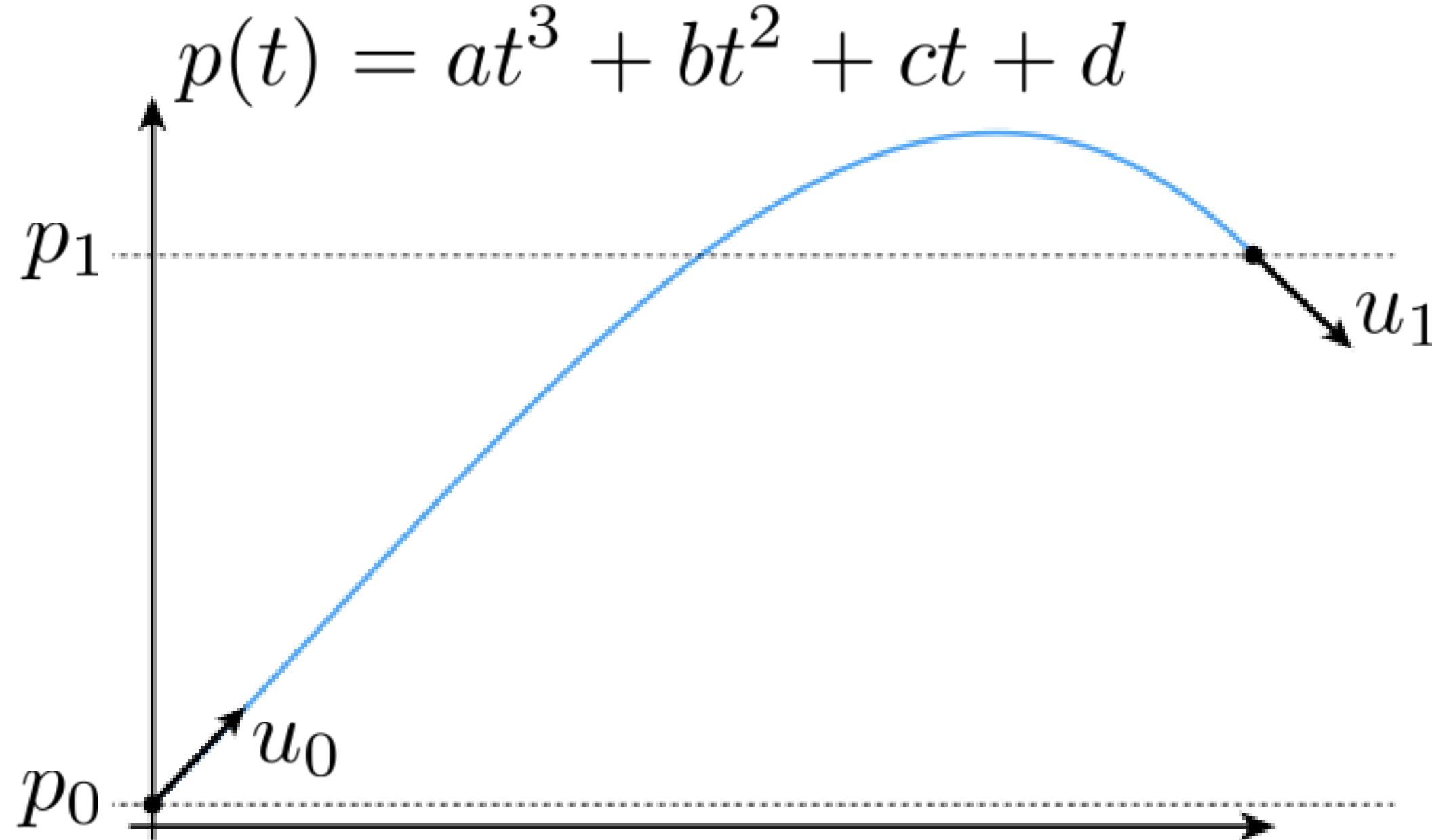
$$p(0) = p_0 \qquad \Rightarrow d = p_0$$

$$p(1) = p_1 \qquad \Rightarrow a + b + c + d = p_1$$

- Overall, four unknowns but only two equations
- Not enough to uniquely determine the curve!

# Fitting Cubic to Endpoints and Derivatives

- What if we also match *derivatives* at endpoints?



$$p(0) = p_0 \quad \Rightarrow d = p_0$$

$$p(1) = p_1 \quad \Rightarrow a + b + c + d = p_1$$

$$p'(0) = u_0 \quad \Rightarrow c = u_0$$

$$p'(1) = u_1 \quad \Rightarrow 3a + 2b + c = u_1$$

# Splines as Linear Systems

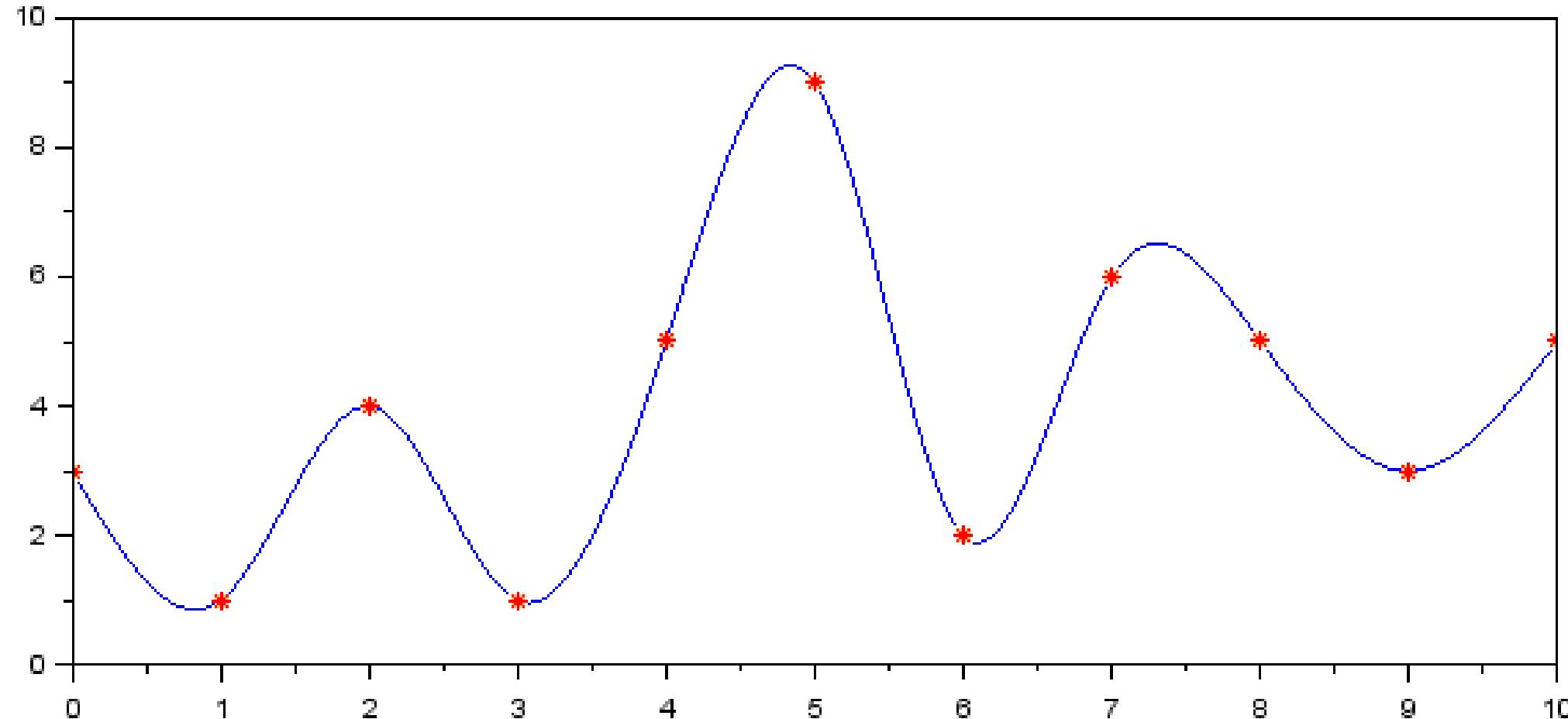
- This time, we have four equations in four unknowns
- Could also express as a matrix equation:

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} p_0 \\ p_1 \\ u_0 \\ u_1 \end{bmatrix}$$

- Often, this is the game we will play:
  - each condition on values or derivatives leads to a linear equality
  - if we have  $m$  degrees of freedom, we need  $m$  conditions to determine spline

# Natural Splines

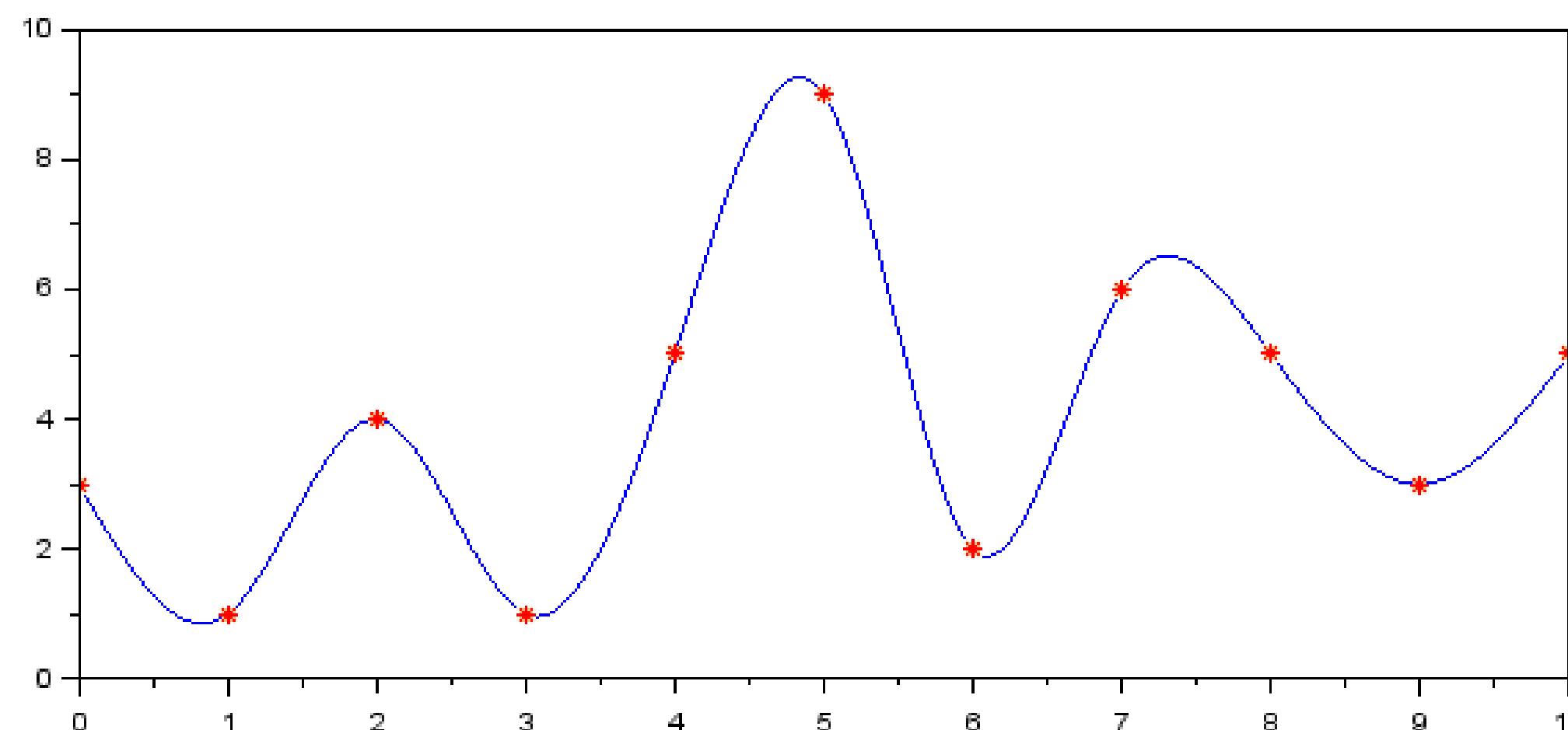
- *Piecewise spline made of cubic polynomials  $p_i$*



- **Want: interpolation at both endpoints:**
$$p_i(t_i) = f_i, \quad p_i(t_{i+1}) = f_{i+1}, \quad i = 0, \dots, n - 1$$
- **Want tangents to agree at endpoints (“ $C^1$  continuity”):**
$$p'(t_{i+1}) = p'_{i+1}(t_{i+1}), \quad i = 0, \dots, n - 2$$
- **Want curvature to agree at endpoints (“ $C^2$  continuity”):**
$$p''(t_{i+1}) = p''_{i+1}(t_{i+1}), \quad i = 0, \dots, n - 2$$

# Natural Splines

- How many degrees of freedom? How many equations?
  - **4n DOFs,  $2n+(n-1)+(n-1) = 4n-2$  equations**
- Pin down remaining DOFs by setting curvature to zero at endpoints (this is what makes the curve “natural”)

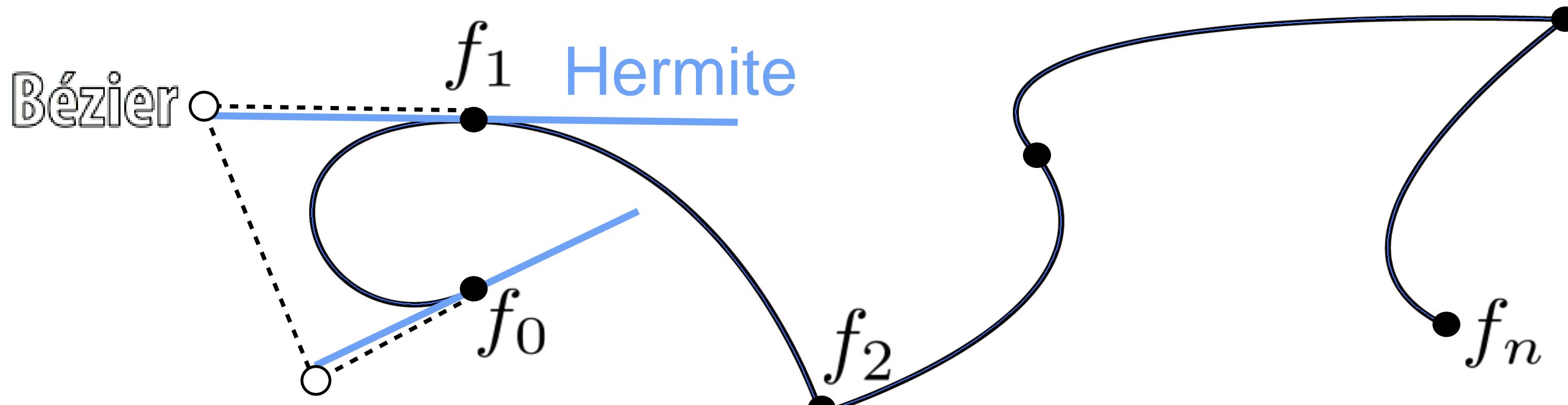


# Spline Desiderata

- In general, what are some properties of a “good” spline?
  - INTERPOLATION: spline passes *exactly* through data points
  - CONTINUITY: at least *twice* differentiable everywhere
  - LOCALITY: moving one control point doesn’t affect whole curve
- How does our natural spline do?
  - INTERPOLATION: yes, by construction
  - CONTINUITY:  $C^2$  everywhere
  - LOCALITY: no, coefficients depend on global linear system
- Many other types of splines we can consider
- Spoiler: there is “no free lunch” with cubic splines (can’t simultaneously get all three properties)

# Hermite/Bézier Splines

- Each cubic “piece” specified by endpoints and tangents:



- Equivalently: by four points (Bézier form); just take difference!
- Commonly used for 2D vector art (Illustrator, SVG, ...)
- Can we get tangent continuity?
- Sure: set tangents to same value on both sides of knot!
  - E.g.,  $f_1$  above, but not  $f_2$

# Properties of Hermite/Bézier Spline

- More precisely, want endpoints to interpolate data:

$$p_i(t_i) = f_i, \quad p_i(t_{i+1}) = f_{i+1}, \quad i = 0, \dots, n - 1$$

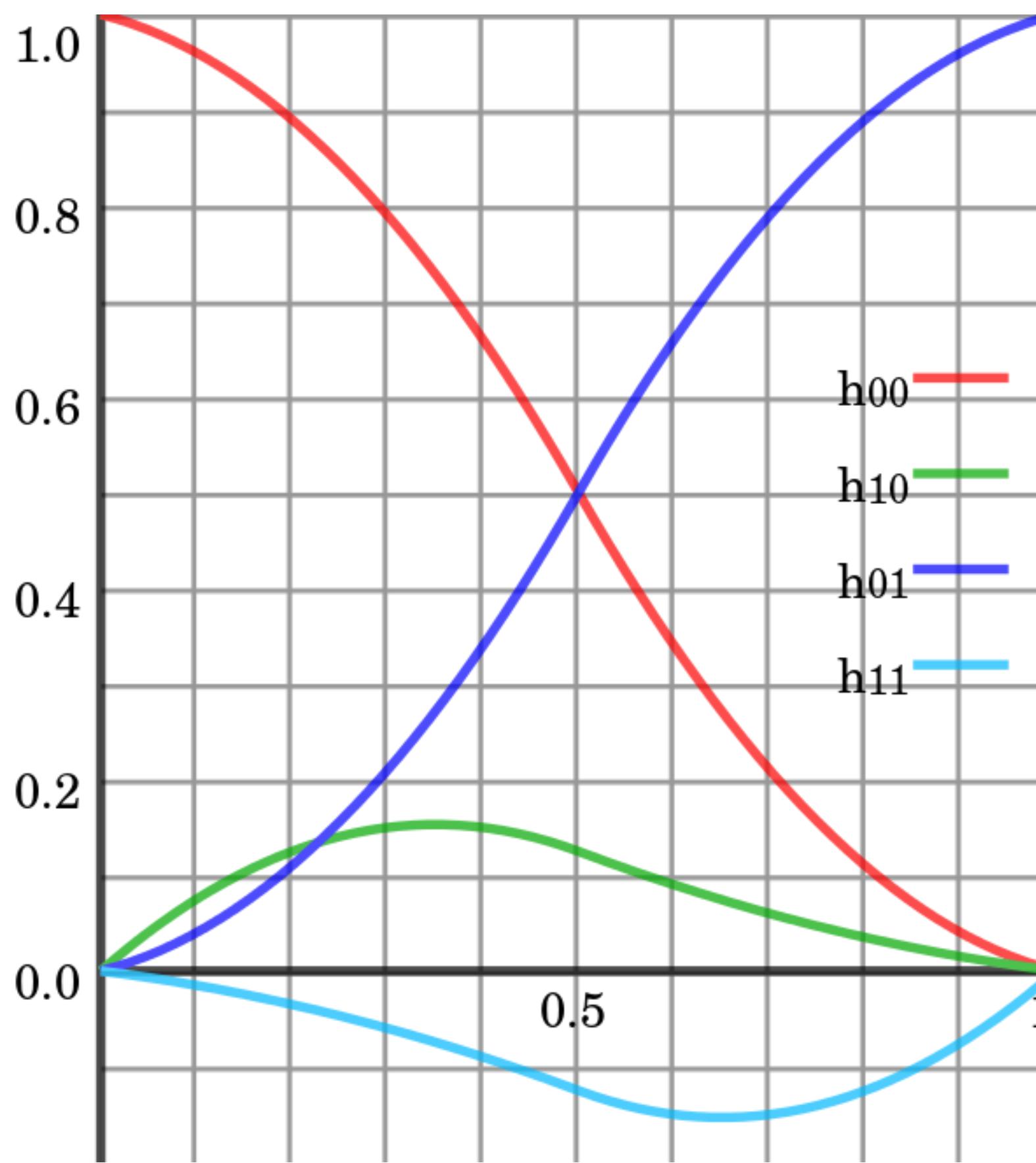
- Also want tangents to interpolate some given data:

$$p'_i(t_i) = u_i, \quad p'(i)_{t_{i+1}} = u_{i+1}, \quad i = 0, \dots, n - 1$$

- How is this *different* from our natural spline's tangent condition?
- There, tangents didn't have to match any prescribed value—they merely had to be the same. Here, they are given.
- How many conditions overall?
  - $2n + 2n = 4n$
- What properties does this curve have?
  - INTERPOLATION and LOCALITY, but not  $C^2$  CONTINUITY

# Hermite/Bézier Splines as sums of basis functions

$$\mathbf{p}(t) = (2t^3 - 3t^2 + 1)\mathbf{p}_0 + (t^3 - 2t^2 + t)\mathbf{m}_0 + (-2t^3 + 3t^2)\mathbf{p}_1 + (t^3 - t^2)\mathbf{m}_1$$

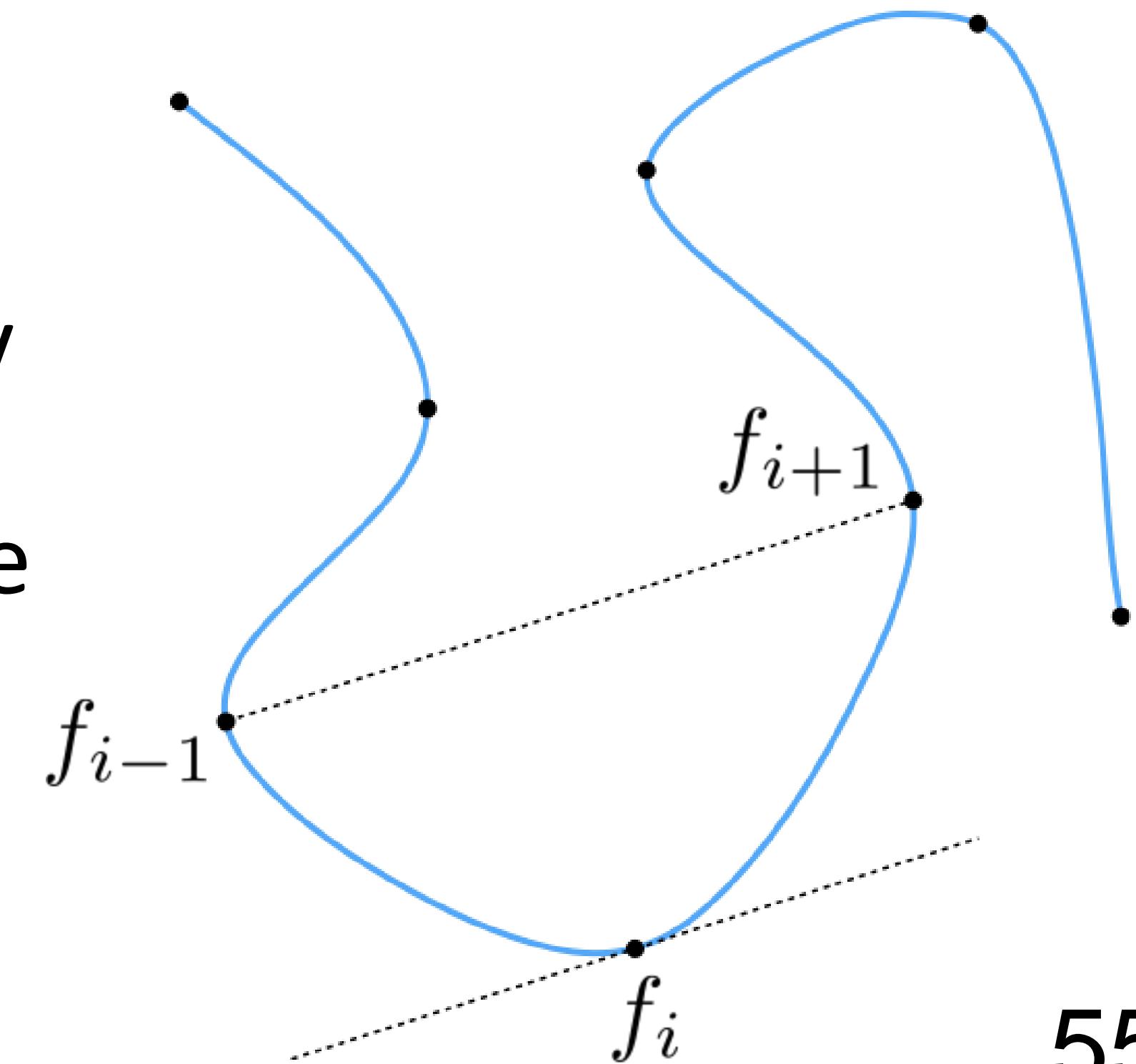


# Catmull-Rom Splines

- Sometimes makes sense to specify *tangents*, but often more convenient to just specify *values*
- Catmull-Rom: specialization of Hermite spline, determined by values alone
- Basic idea: use difference of neighbors to define tangent

$$u_i := \frac{f_{i+1} - f_{i-1}}{t_{i+1} - t_{i-1}}$$

- All the same properties as any other Hermite spline
- Commonly used to interpolate motion in computer animation.
- Many, many variants, but Catmull-Rom is usually good starting point



# Spline Desiderata, Revisited

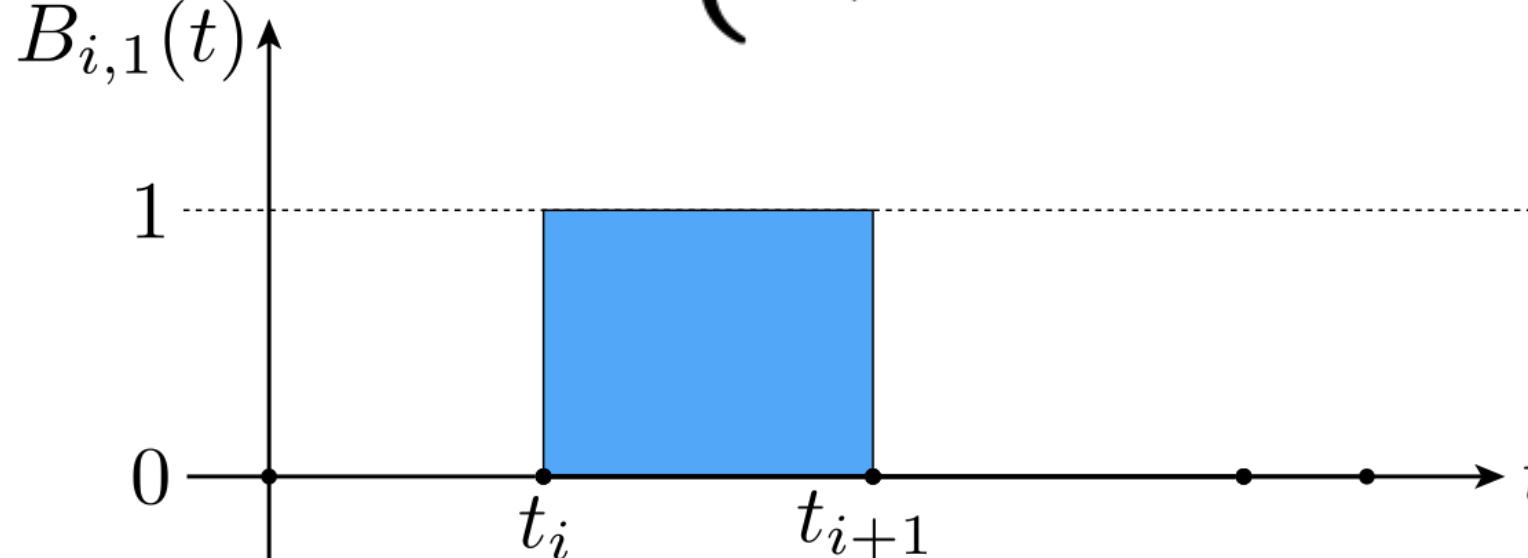
	INTERPOLATION	CONTINUITY	LOCALITY
natural	YES	YES	NO
Hermite	YES	NO	YES
???	NO	YES	YES

# B-Splines

- Get better continuity *and* local control! Sacrifice interpolation ☹

- B-spline *basis* defined recursively:

$$B_{i,1}(t) := \begin{cases} 1, & \text{if } t_i \leq t < t_{i+1} \\ 0, & \text{otherwise} \end{cases}$$



linear interpolation

$$B_{i,k}(t) := \frac{t-t_i}{t_{i+k-1}-t_i} B_{i,k-1}(t) + \frac{t_{i+k}-t}{t_{i+k}-t_{i+1}} B_{i+1,k-1}(t)$$

- B-spline itself is then a linear combination of bases:

$$f(t) := \sum_i a_i B_{i,d}$$

degree

# Spline Desiderata, Revisited

	INTERPOLATION	CONTINUITY	LOCALITY
natural	YES	YES	NO
Hermite	YES	NO	YES
B-Splines	NO	YES	YES

**Ok, I get it: splines are great.  
But what exactly are we  
interpolating?**

# Animation rigs...

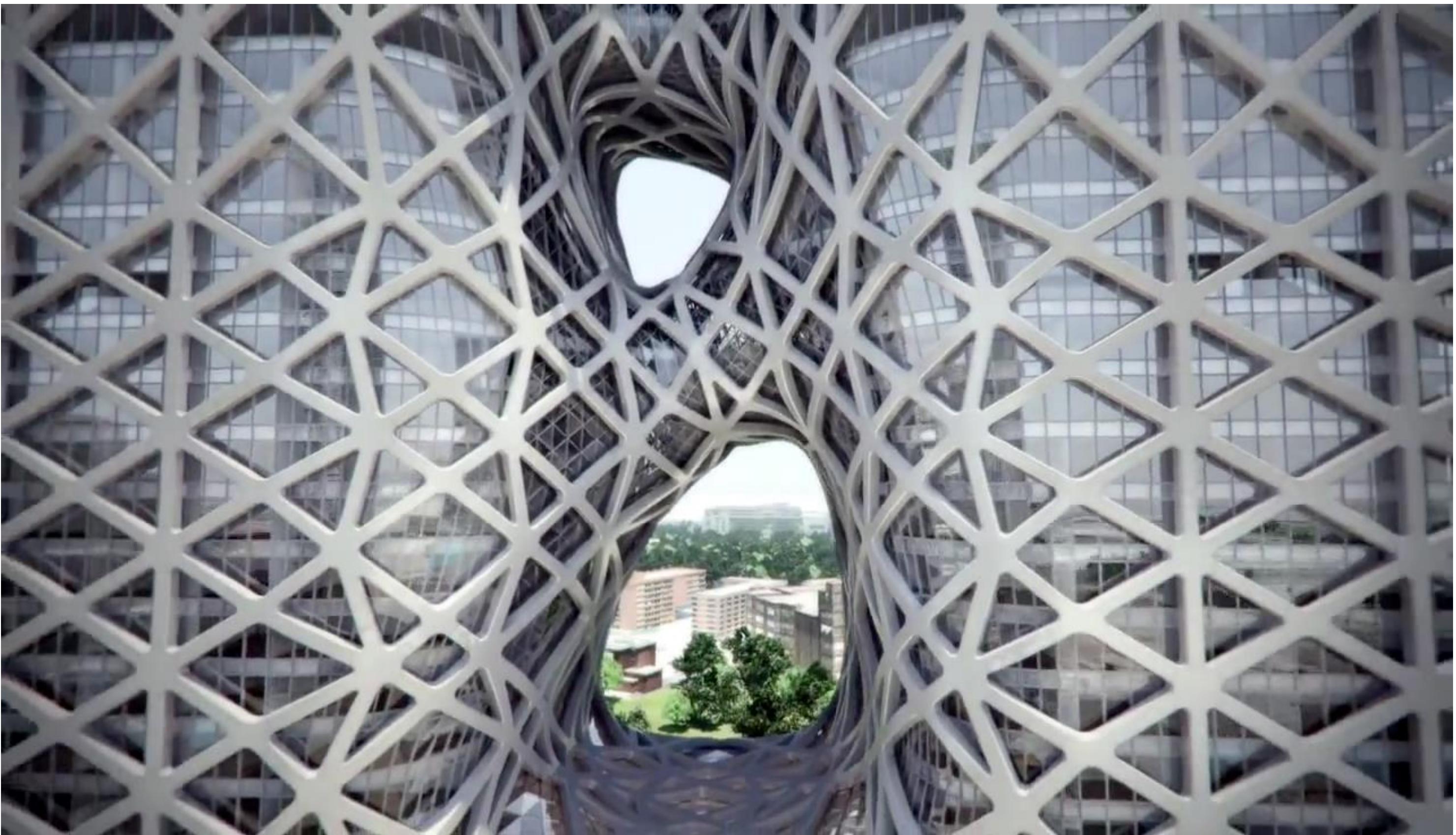
- In the hands of a skilled animator...



# Another example: camera path

## ■ Animate position, view direction, “up” direction

- each path is a function  $f(t) = (x(t), y(t), z(t))$
- each component (x,y,z) is a spline



Zaha Hadid Architects—City of Dreams Hotel Tower