CSC418/2504 Computer Graphics

Some Slides/Images adapted from Marschner and Shirley

Some slides courtesy of Karan Singh, Videos courtesy of AlanBeckerTutorials

Announcements

- Assignment 6 due on March 15th, Assignment 7 due on March 29th

- Office hours *should* be in Dave's office from now on, in BA 5268

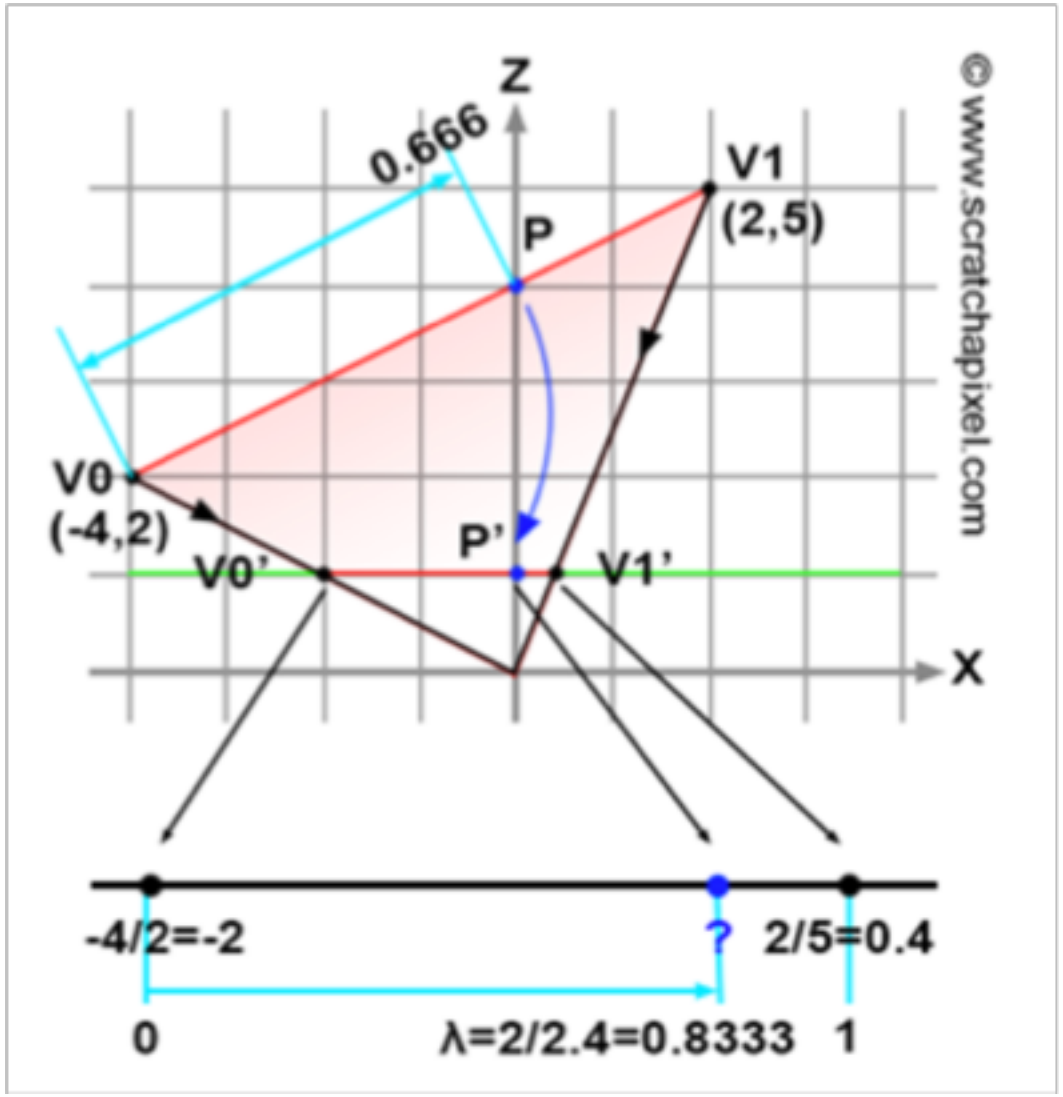
- Undergraduate drop date: March 17th

- No lecture March 19th or 20th. We'll still have tutorial that week, and you can ask questions in preparation for Midterm 2

- Midterm remark request office hour on Monday the 11th (Time TBA, keep an eye out on the discussion board)

Any Questions ?

Note on last time, calculating pseudo-depth



$$\frac{1}{P.z} = \frac{1}{V0.z} * (1 - \lambda) + \frac{1}{V1.z} * \lambda.$$

What didn’t you cover in terms of Ray tracing? (aka lecture 9 preview)

- Super-sampling/Anti-aliasing
- Depth of field
- Glossy reflections
- Path tracing/Caustics
- Motion blur

I’ll also cover artistic shading such as:
Cell shading, comic-book shading.

These topics will help you win the competition!

Topic 1: Animation

Lecture 8: Topic 1: Animation

- Motivation
- Key principles of animation
- Elements of CG
  - Keyframing
  - Keyframing: Character Animation
    - Rig (Skeleton)
    - Forward Kinematics
    - Inverse Kinematics
  - Physical Simulation
  - Motion Capture

Lecture 8: Topic 1: Animation

- Motivation
- Key principles of animation
- Elements of CG
  - Keyframing
  - Keyframing: Character Animation
    - Rig (Skeleton)
    - Forward Kinematics
    - Inverse Kinematics
  - Physical Simulation
  - Motion Capture

Motivation

You’ve mastered making an image, now let’s make a bunch of em’!

Animation Timeline

- 1908: Emile Cohl (1857-1938) France, makes his first film, FANTASMAGORIE, arguably the first animated film (running time ~1min 20sec)

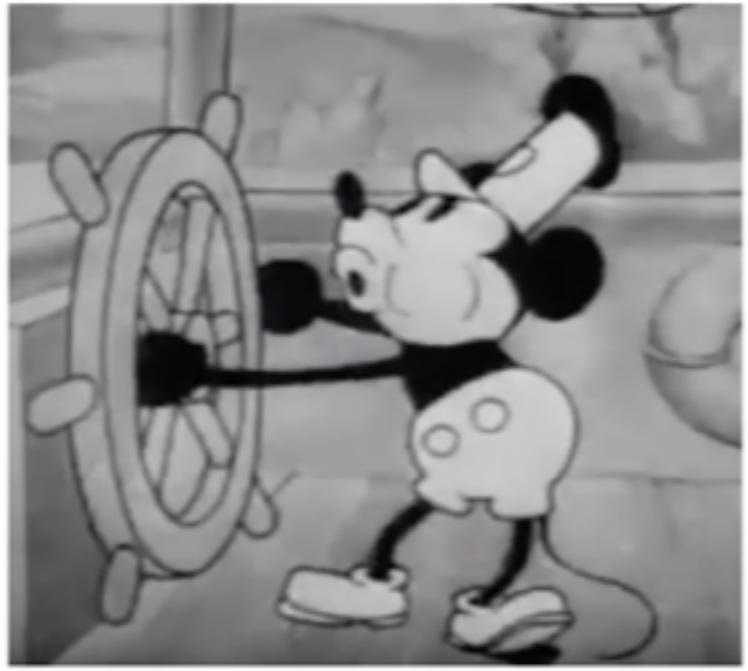
Animation Timeline

- 1911: Winsor McCay (1867-1934) makes his first film, LITTLE NEMO. McCay, already famous for comic strips, used the film in his vaudeville act. Pioneered keyframe animation, open about it and refused to patent his technique saying:

Any idiot that wants to make a couple of thousand drawings for a hundred feet of film is welcome to join the club.

Animation Timeline

- 1928: Walter Disney (1901-1966) working at the Kansas City Slide Company creates Mickey Mouse.



Animation Timeline

- 1974: First Computer animated film "Faim" from NFB nominated for an Oscar.

Lecture 8: Topic 1: Animation

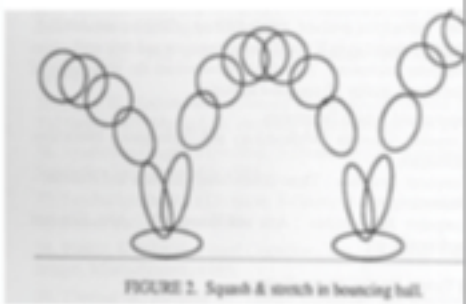
- Motivation
- Key principles of animation
- Elements of CG
  - Keyframing
  - Keyframing: Character Animation
    - Rig (Skeleton)
    - Forward Kinematics
    - Inverse Kinematics
  - Physical Simulation
  - Motion Capture

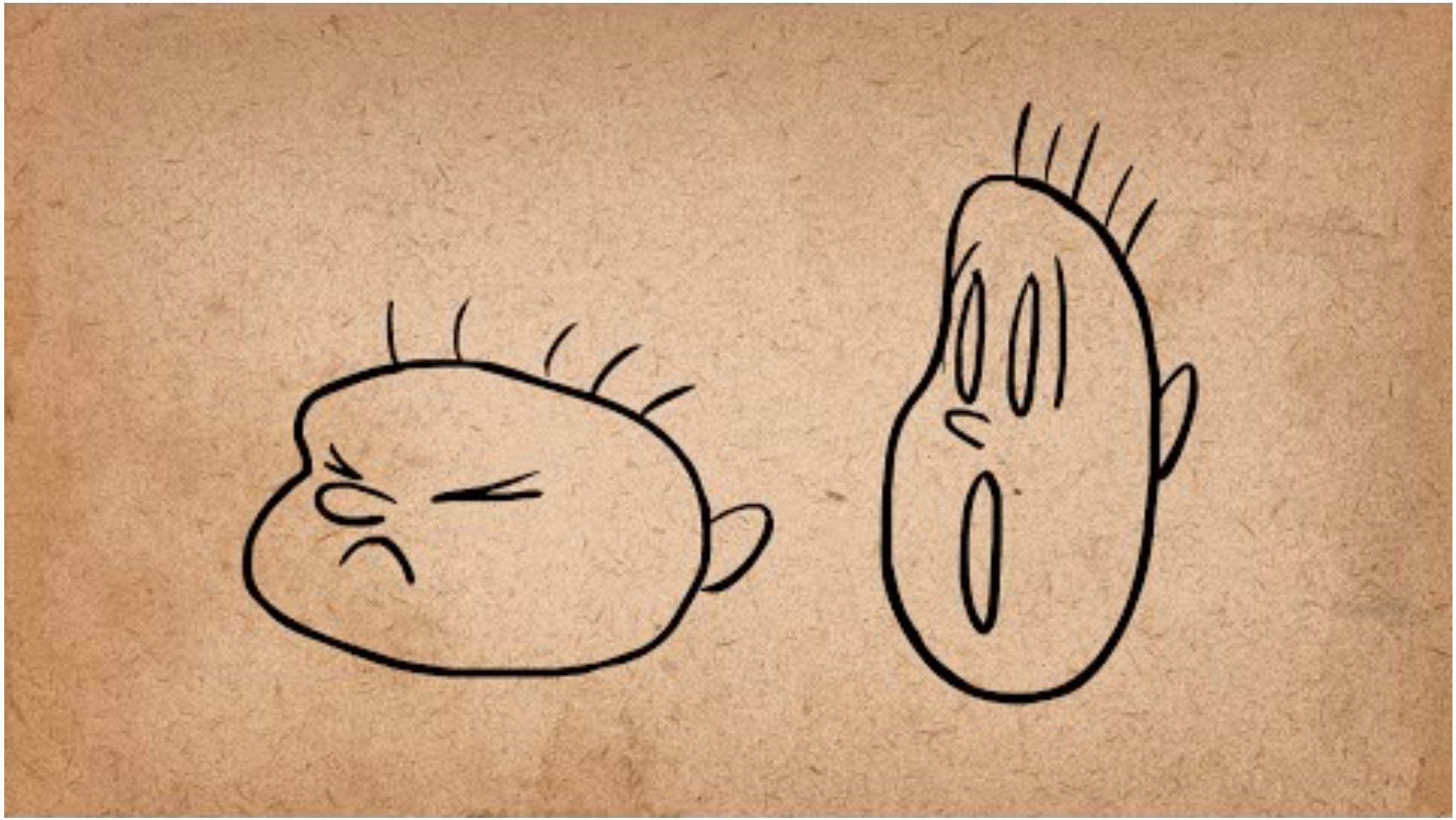
Animation Principles

- 12 basic principles of animation
- Deals with emotional timing and laws of physics
- Disney book “The Illusion of Life: Disney Animation”

Squash and Stretch

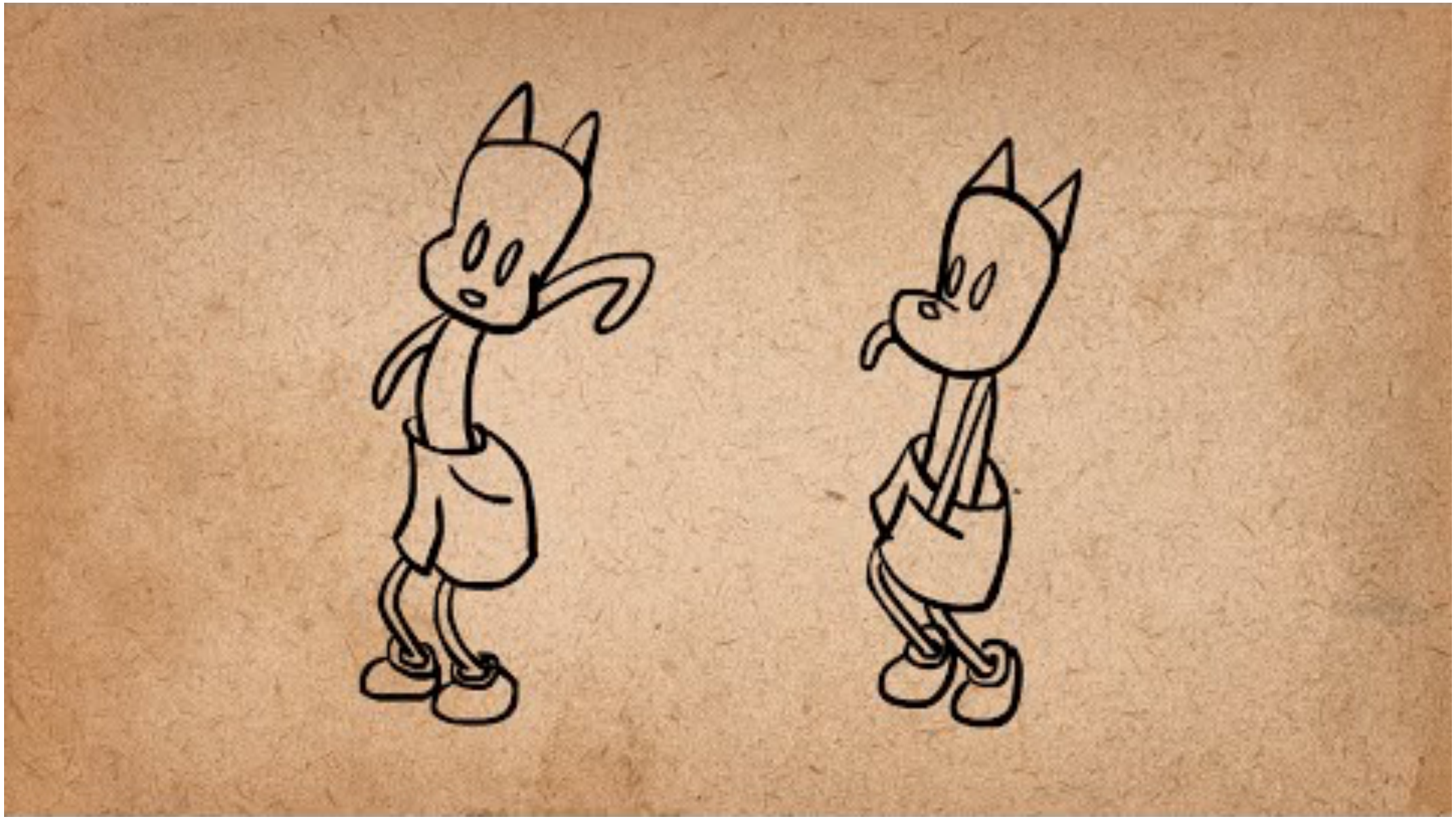
- Rigid objects look robotic: ***deformations*** make motion natural
- Accounts for physics of deformation
  - Think squishy ball…
  - Communicates to viewer object's composition, its weight,…
  - Usually large deformations conserve volume:
    Squash in one dimension, stretch in another to keep mass constant
- Also accounts for persistence of vision
  - Fast moving objects leave an elongated streak on our retinas





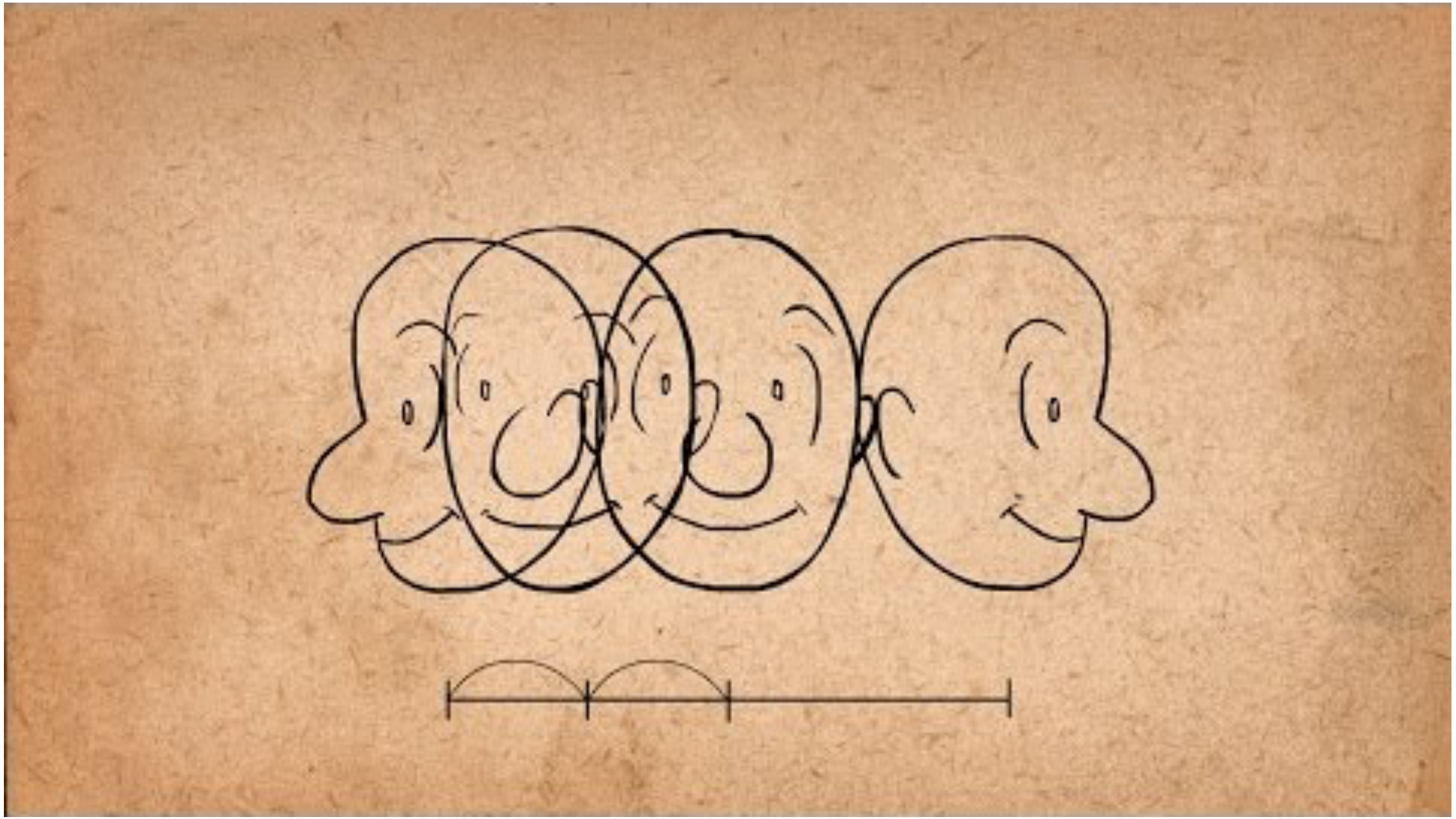
Anticipation

- The preparation before a motion
  - e.g. crouching before jumping, pitcher winding up to throw a ball
- Often physically necessary, and indicates how much effort a character is making
- Also essential for controlling the audience's attention, to make sure they don't miss the action
  - Signals something is about to happen, and where it is going to happen.



Ease-In & Ease-Out

- Objects don't have immediate speed and don't stop immediately
- Objects accelerate and decelerate, even if quickly
- Add more drawings near the beginning and end
- Emphasize extreme poses
- Can be controlled with spline interpolation



Staging

- Similar concept in film and theatre
- Direct attention to most important point
- Remove ambiguity

Straight Ahead and Pose to Pose

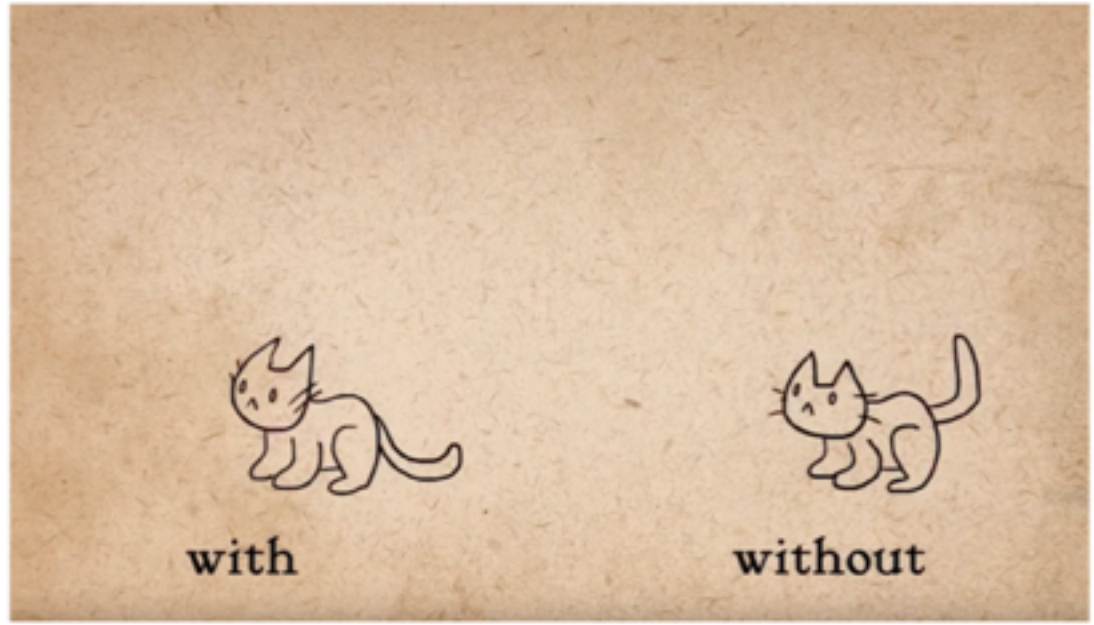
- Straight ahead is to draw the scene frame by frame from beginning to end in a linear fashion
- Pose to pose draws important key moments (key frames) and fills in between the frames (inbetweening), to give the illusion of motion.

(more later)

(more on this later when we do keyframing)

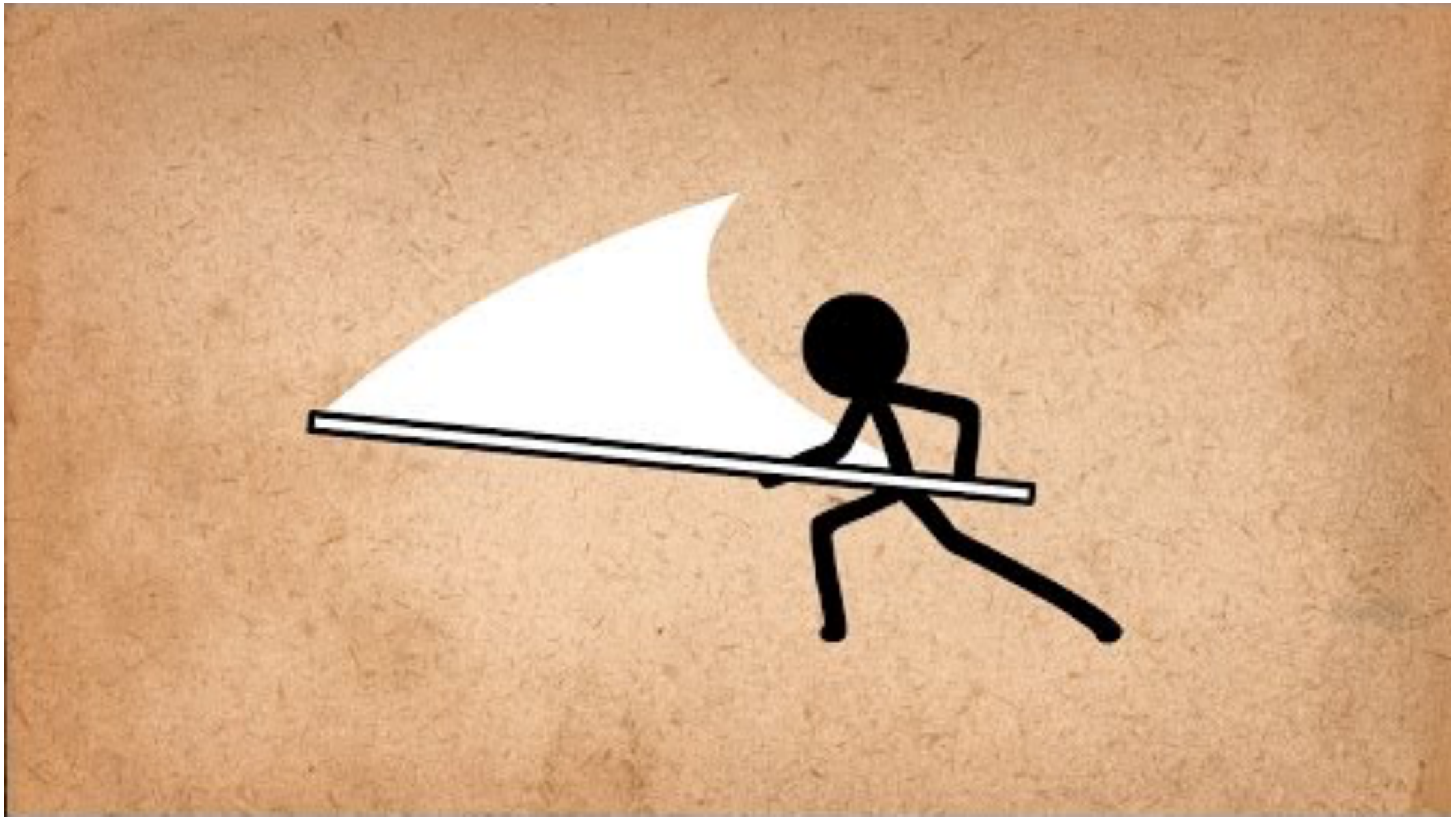
Follow-through and Overlapping Action

- Uses the principle of inertia and laws of physics
- Follow-through: adds realism to animation through continued motion of loosely held parts on the body (hair, tail, clothes, fatty tissue, etc…)
- Overlapping action: different parts of body move at different rates



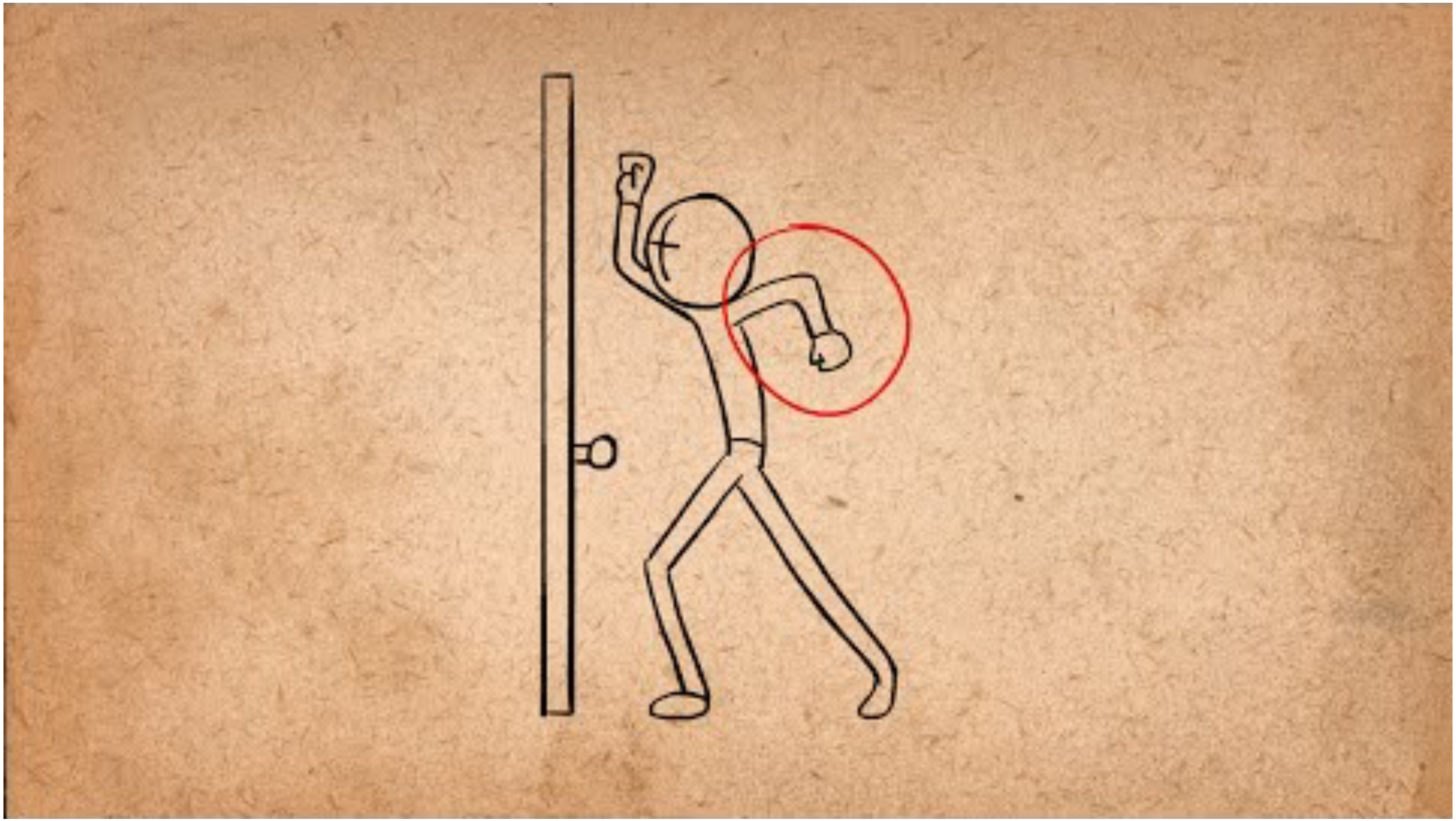
Arcs

- Things naturally move in arcs (baseball pitchers arms motion, swinging of a sword, etc...)
- Balls and objects follow a parabolic trajectory
- Follow the arc when inbetweening



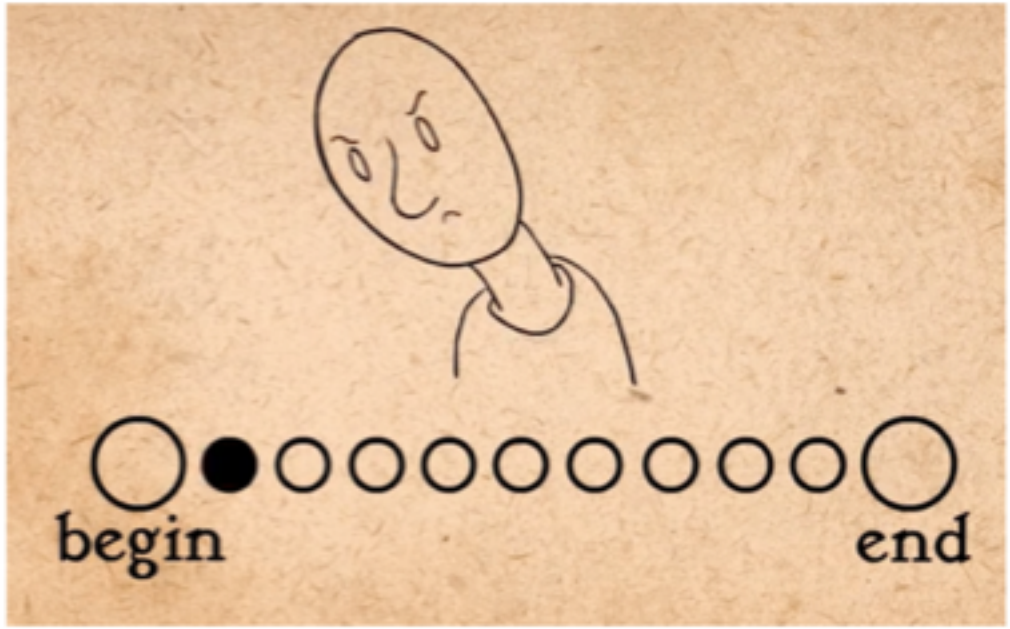
Secondary Action

- Coupling a primary action with natural secondary actions
- e.g. Person walking swinging their arms, wiping a tear while crying, facial expression while eating



Timing

- The number of drawings for a given action.
- More drawings equates to slower motion
- Less drawings equates to faster motion
- Can convey significantly different messages

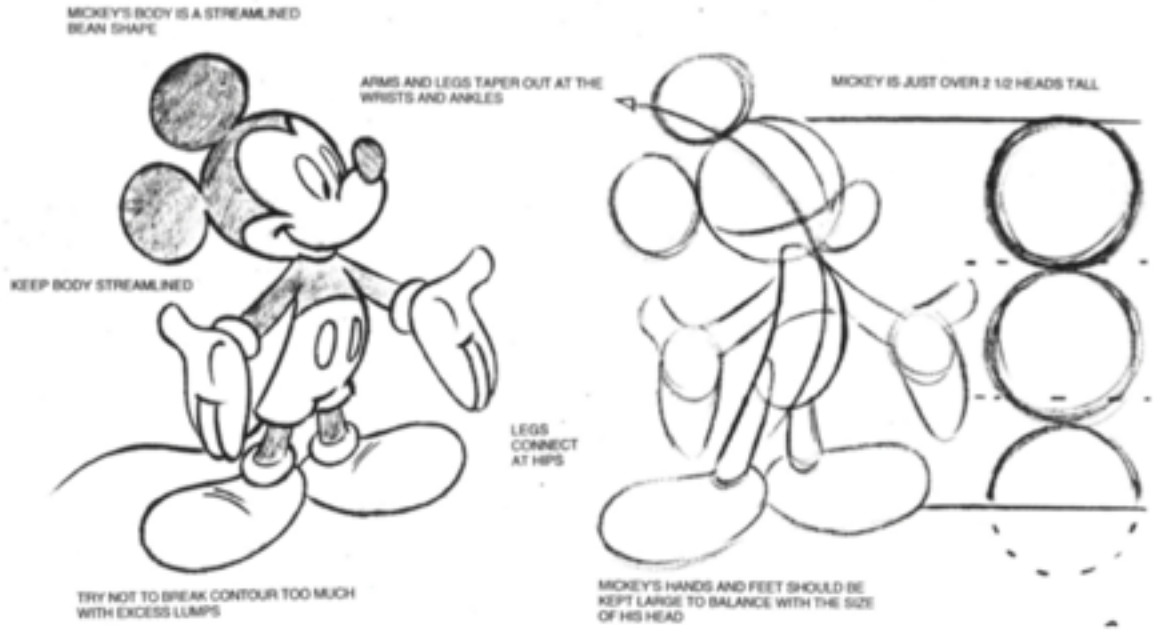


Exaggeration

- Can accentuate certain features
- Adds a degree of style
- Things that are “perfectly” real can come off as dull

Solid Drawing

- Able to draw things that are 3D
- Follow the contours of an object (e.g. sphere)
- Draw in perspective toward vanishing points
- Use solid shapes like cubes and circles to give dimension



Appeal

- Make the character real and interesting
- Pleasing to look at
- Charismatic aspect
- Interesting to look at, not necessarily “good-looking”
- Avoid symmetry

Animation Principles

- **Squash and Stretch**
- **Anticipation**
- **Ease-In & Ease-Out**
- Staging
- Straight Ahead and Pose to Pose
- Follow-through and Overlapping Action
- Arcs
- Secondary Action
- Timing
- Exaggeration
- Solid Drawing
- Appeal

What can be animated?

Lights

Camera

Jointed figures

Deformable objects

Clothing

Skin/muscles

Wind/water/fire/smoke

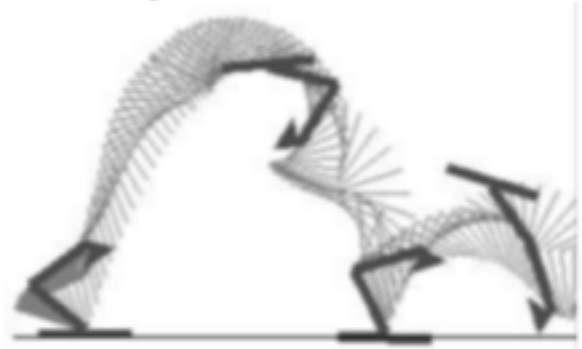
Hair

…any variable, Given the right time scale, almost anything…

Elements of CG (animation)

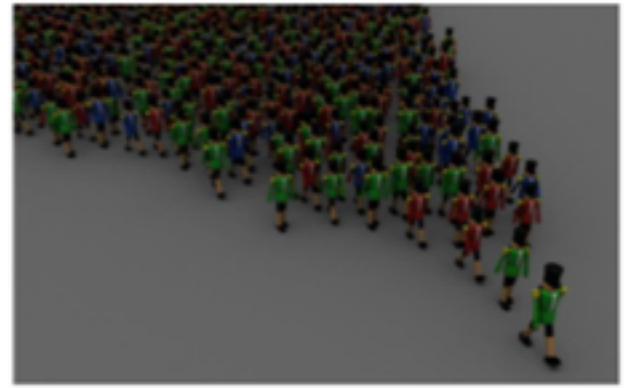
- How does one make digital models move?
  - Keyframing

  - Physical simulation



  - Motion capture

  - Behaviour rules



Lecture 8: Topic 1: Animation

- Motivation
- Key principles of animation
- Elements of CG
  - Keyframing
  - Keyframing: Character Animation
    - Rig (Skeleton)
    - Forward Kinematics
    - Inverse Kinematics
  - Physical Simulation
  - Motion Capture

Keyframes

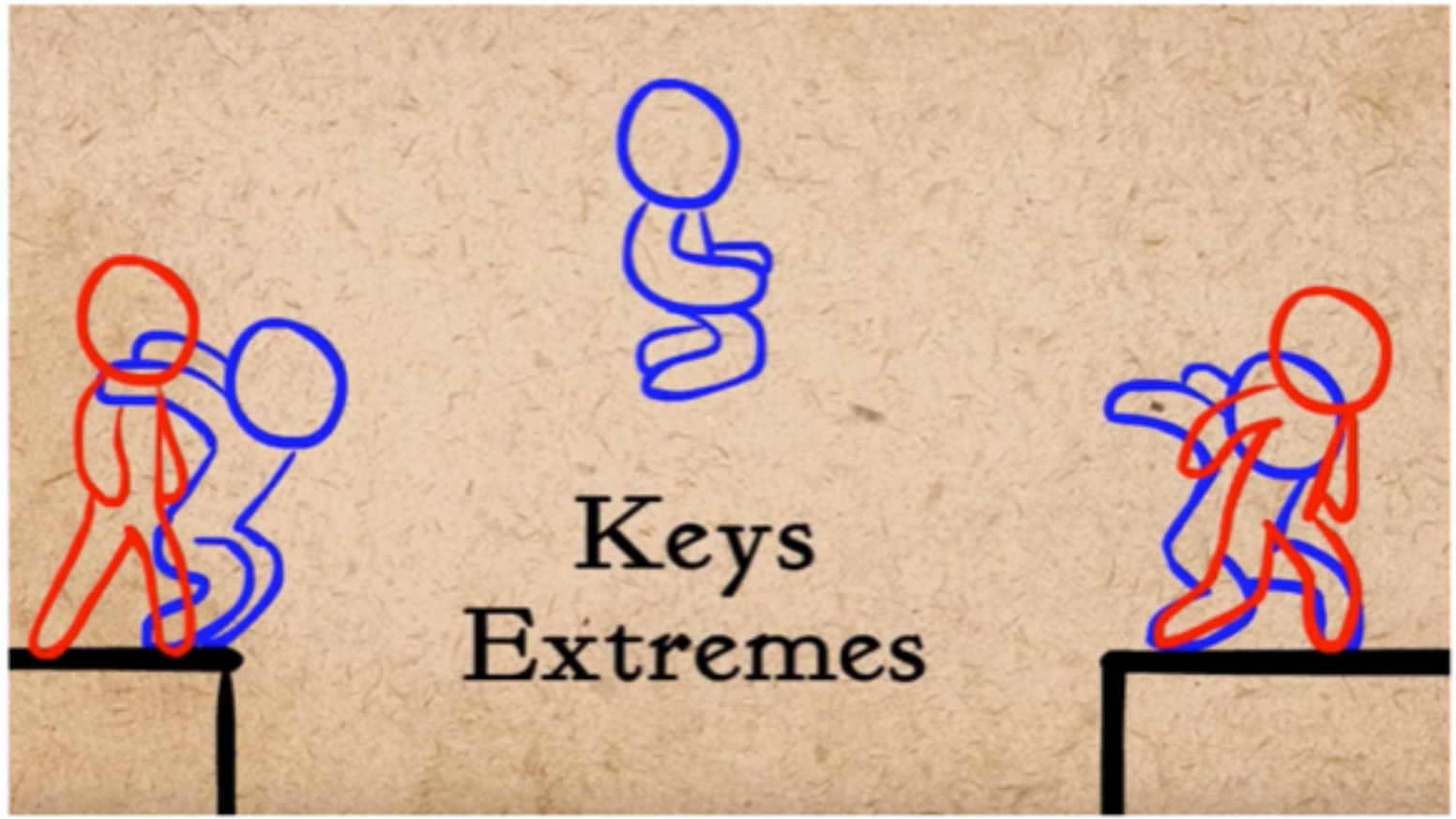
Keyframes, also called extremes, define important poses of a character:
Jump example:

- the start
- the lowest crouch
- the lift-off
- the highest part
- the touch-down
- the lowest follow-through

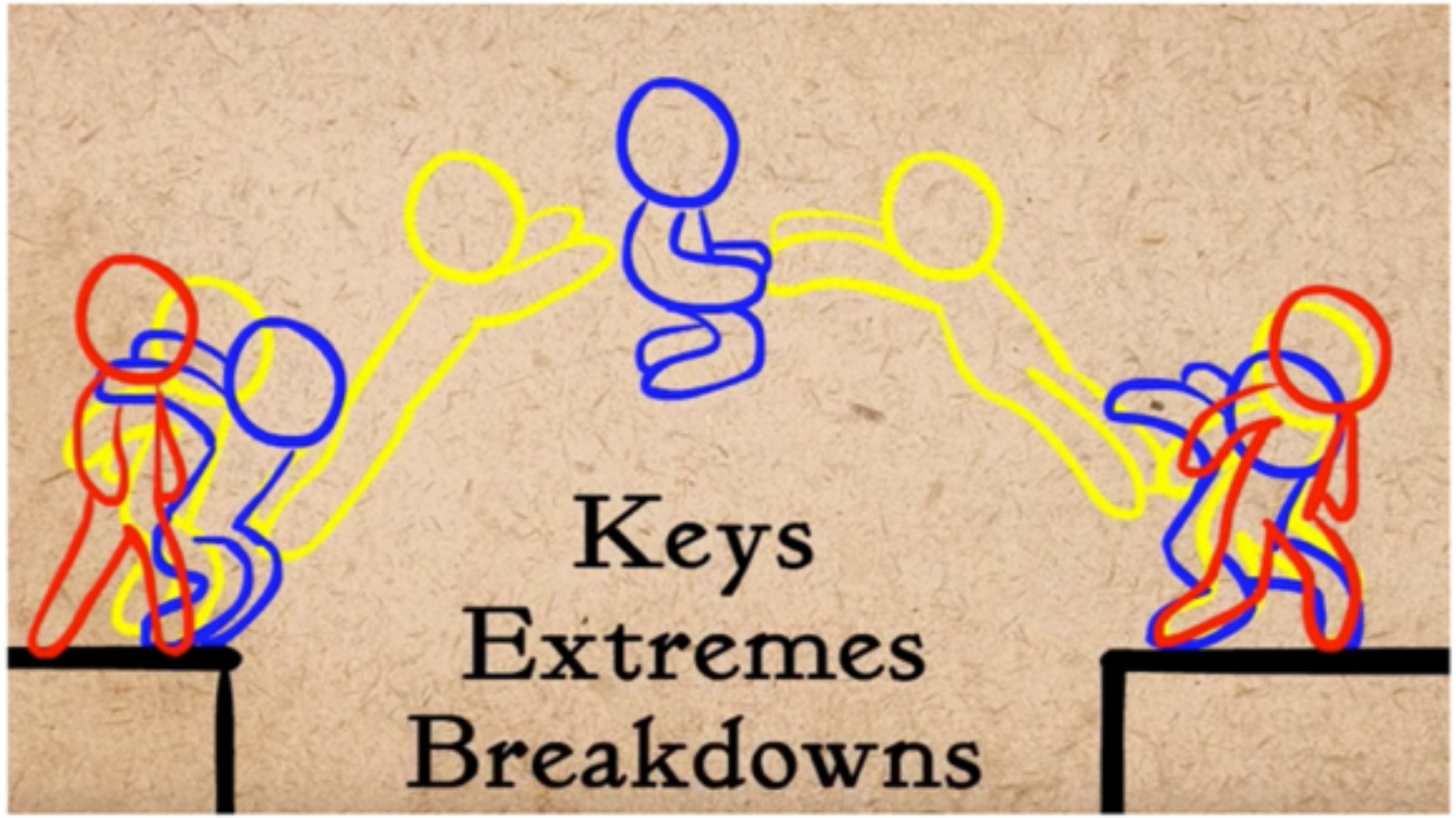
- Frames in between (“inbetweens”) introduce nothing new to the motion.
- May add additional keyframes to add some interest, better control the interpolated motion.

Keyframes

Keyframes



Keyframes

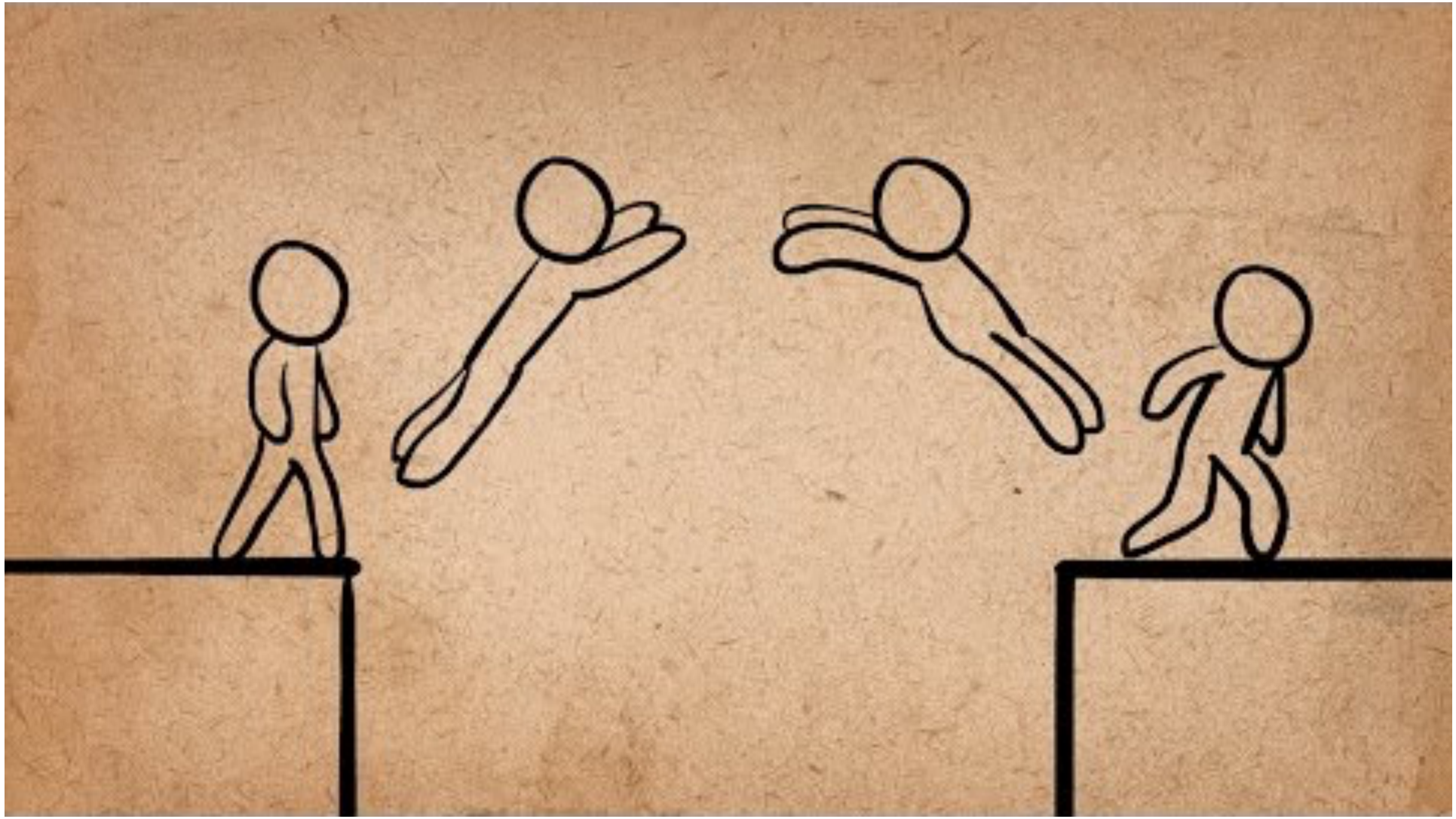


Keyframe Animation

The task boils down to setting animated variables (e.g. positions, angles, sizes, …) at each frame.

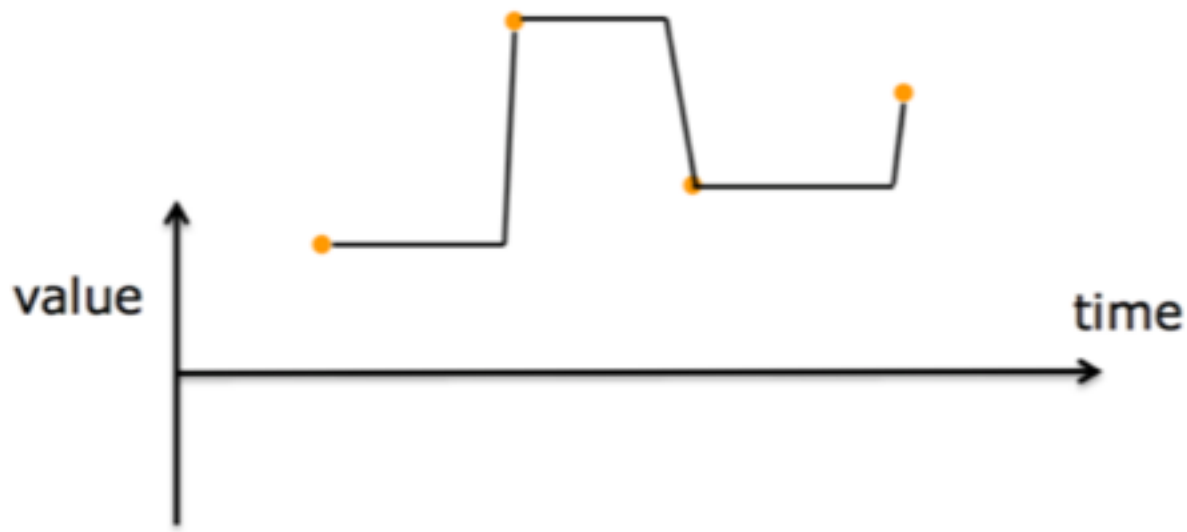
Straight-ahead: set variables in frame 0, then frame 1, frame 2, … forward in time.

Pose-to-pose: set the variables at keyframes, let the computer smoothly interpolate values for frames in between.



Interpolation

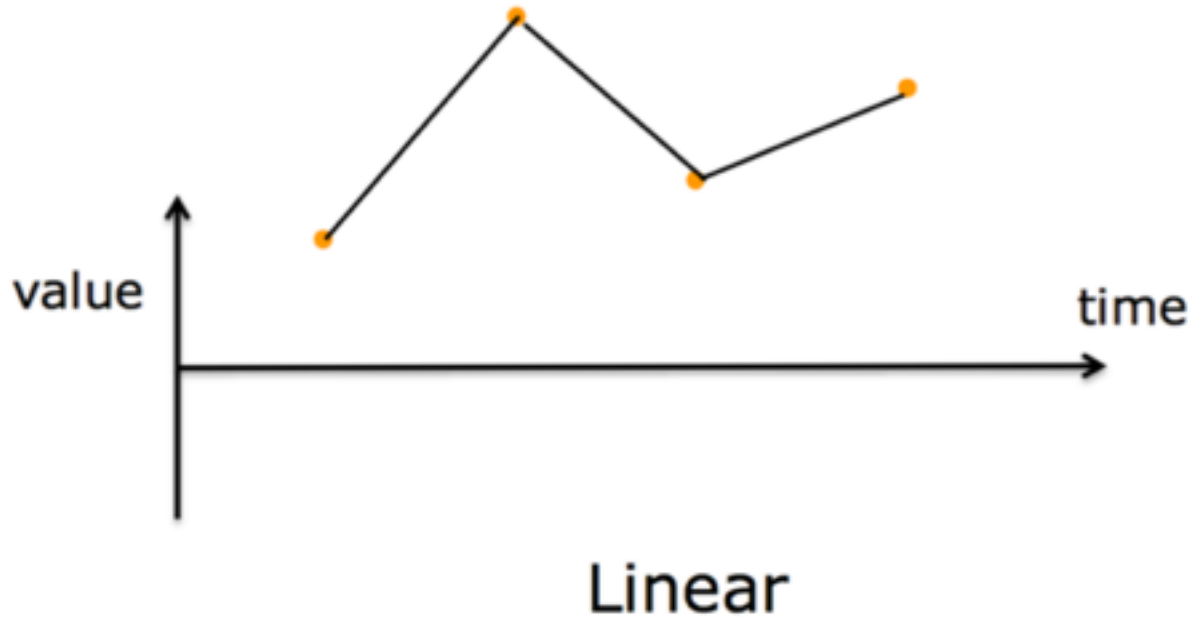
How do we interpolate between two values?



Hold

Interpolation

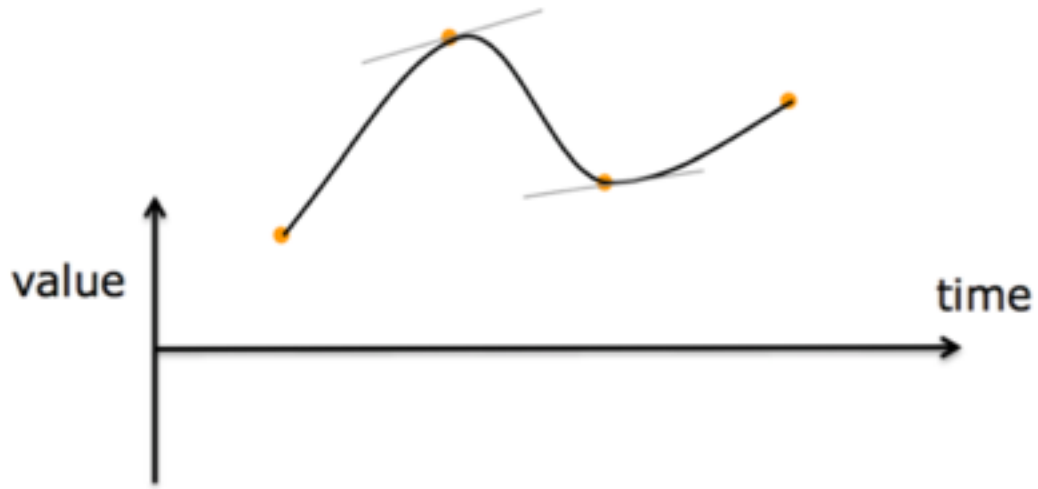
How do we interpolate between two values?



Linear

Interpolation

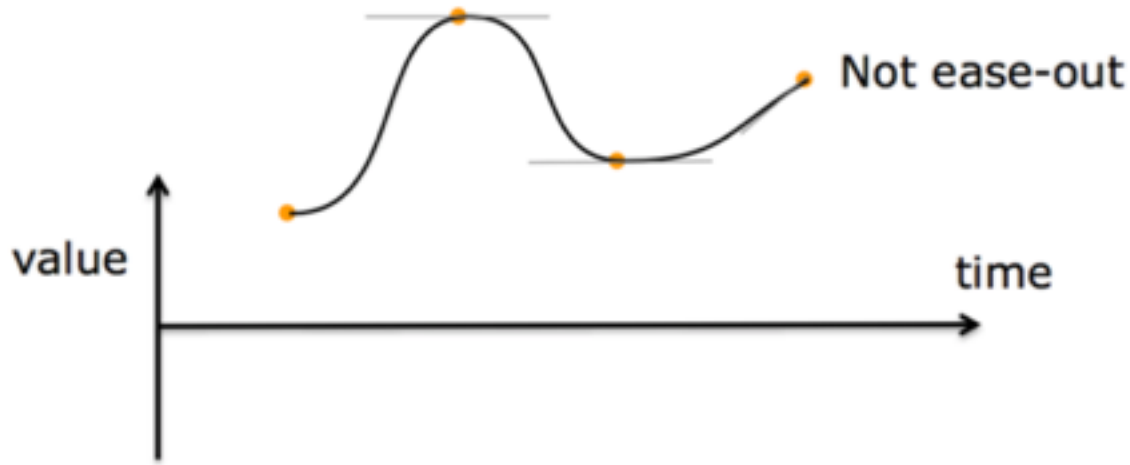
How do we interpolate between two values?



Spline

Interpolation

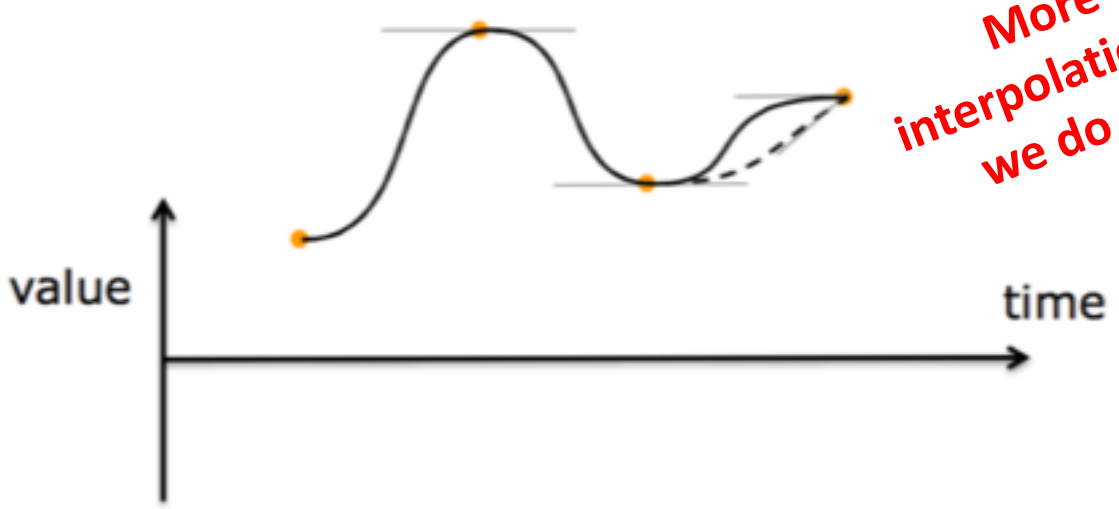
How do we interpolate between two values?



Ease-in Ease-out

Interpolation

How do we interpolate between two values?



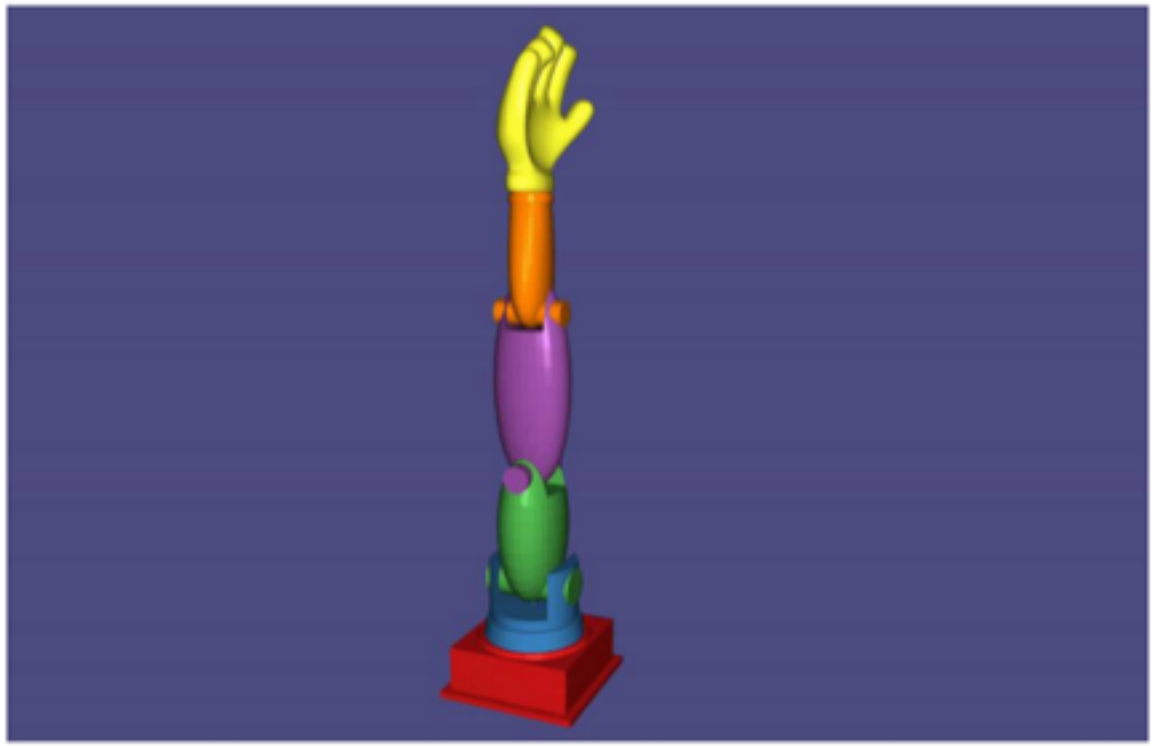
More on interpolation when we do curves

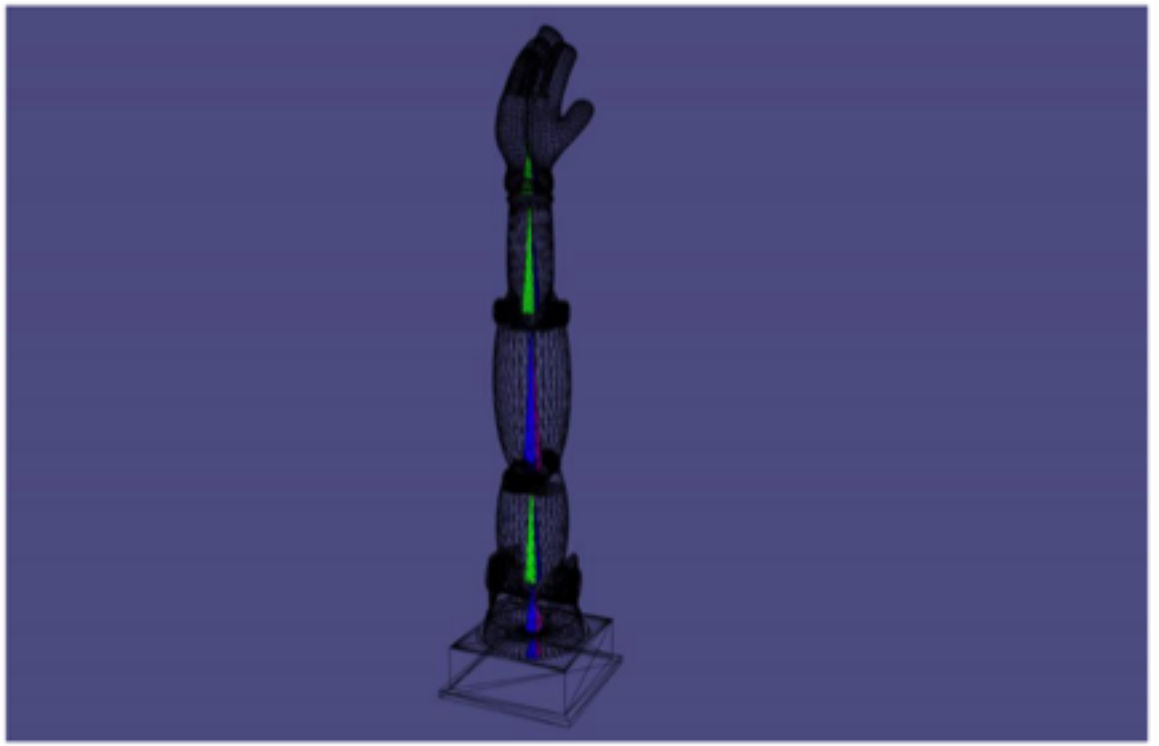
Ease-in Ease-out

Lecture 8: Topic 1: Animation

- Motivation
- Key principles of animation
- Elements of CG
  - Keyframing
  - Keyframing: Character Animation
    - Rig (Skeleton)
    - Forward Kinematics
    - Inverse Kinematics
  - Physical Simulation
  - Motion Capture

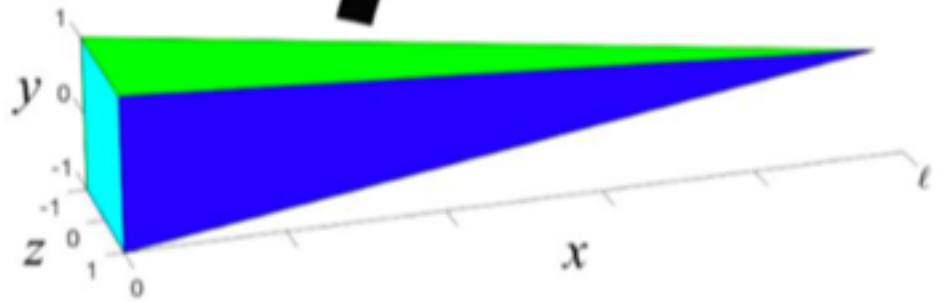
Skeletons

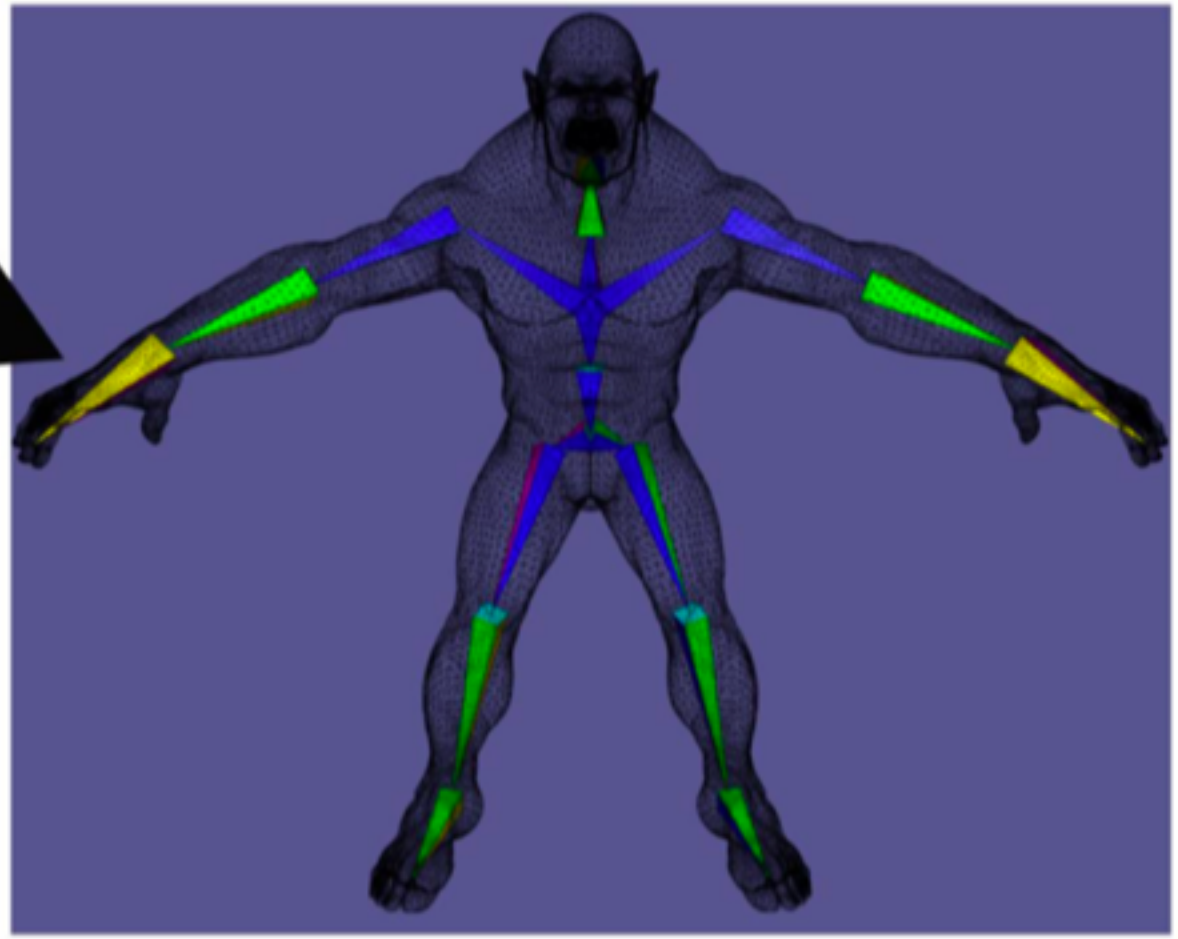




Skeletons: Rest Bone

$$\widehat{\mathbf{T}} = (\widehat{\mathbf{R}} \quad \hat{\mathbf{t}}) \in \mathbb{R}^{3\times4}$$

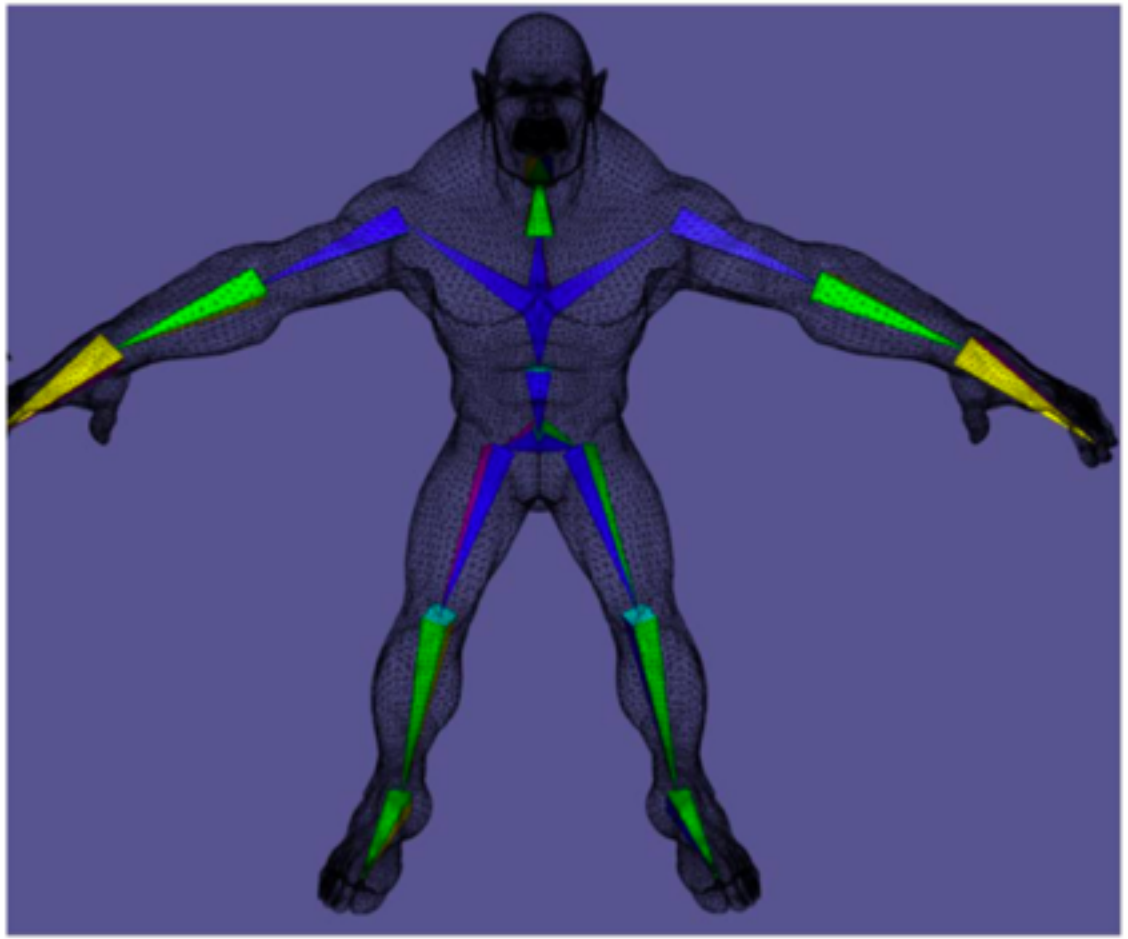




Skeletons: Rest Bone

rest position of *tail*= *tip* of parent bone *p*.

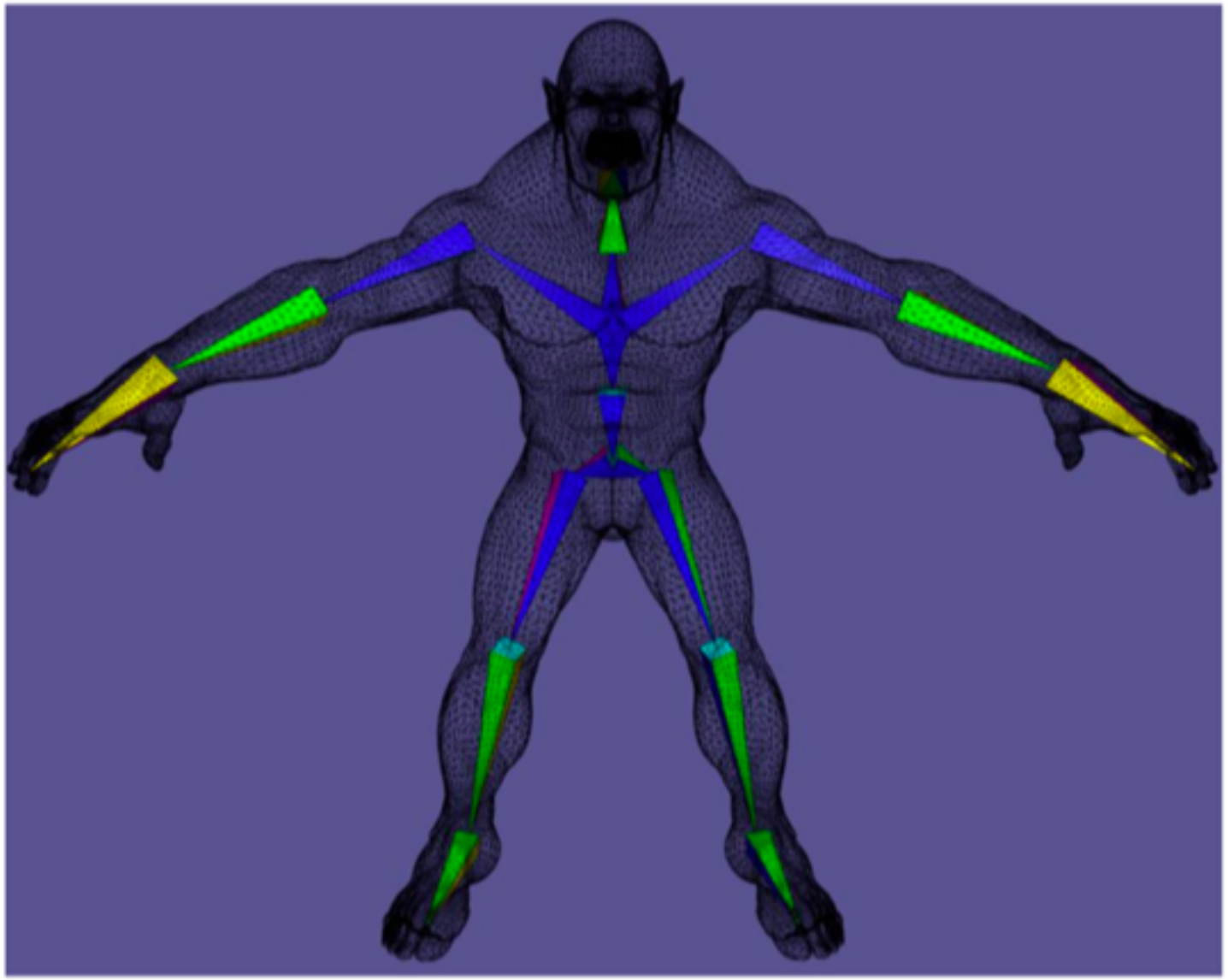
$$\hat{t} = \hat{T}_p [l,0,0]^T$$

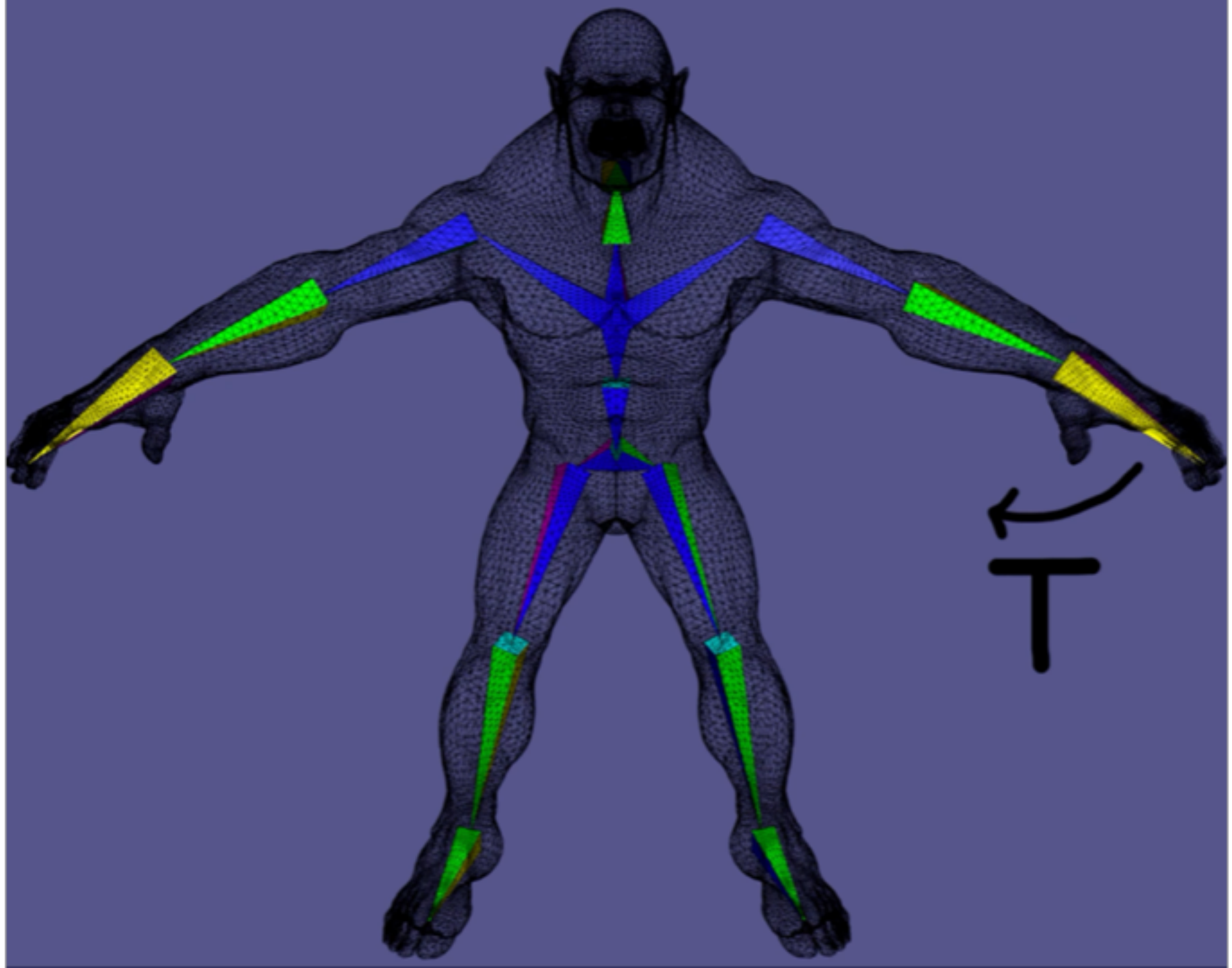


Skeletons: Pose Bone

Move points to x from rest $\hat{x}$ based on bone rotation transform T:

$$x = T\hat{x}$$




T

Forward Kinematics

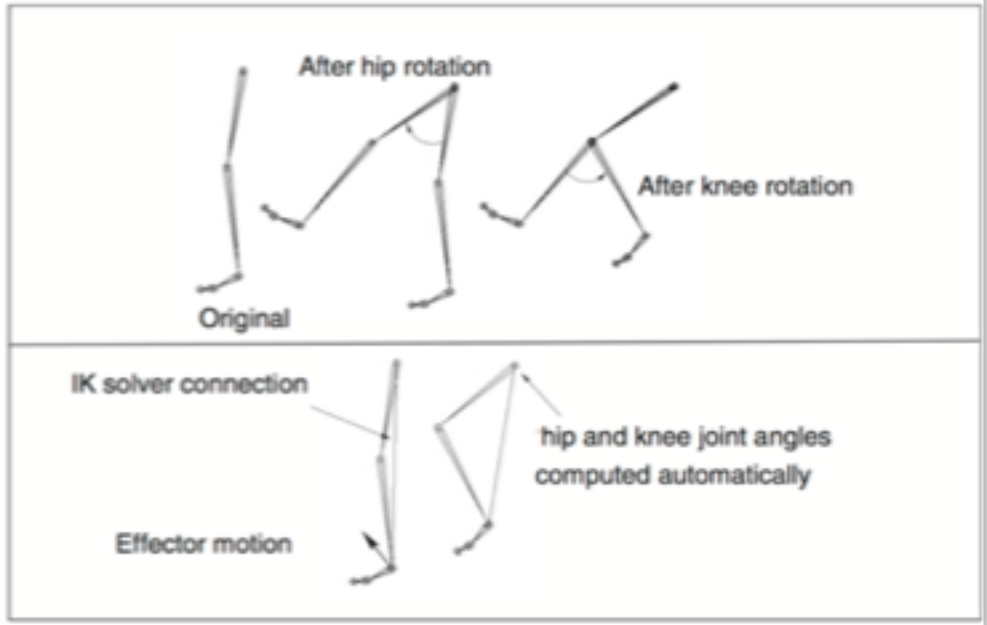
- Specify how joints should move
- Determine the space of possible motion
- Parametrize it
- Establish a mapping from joint angles to positions

Pros:

- Very easy to specify and implement

Cons:

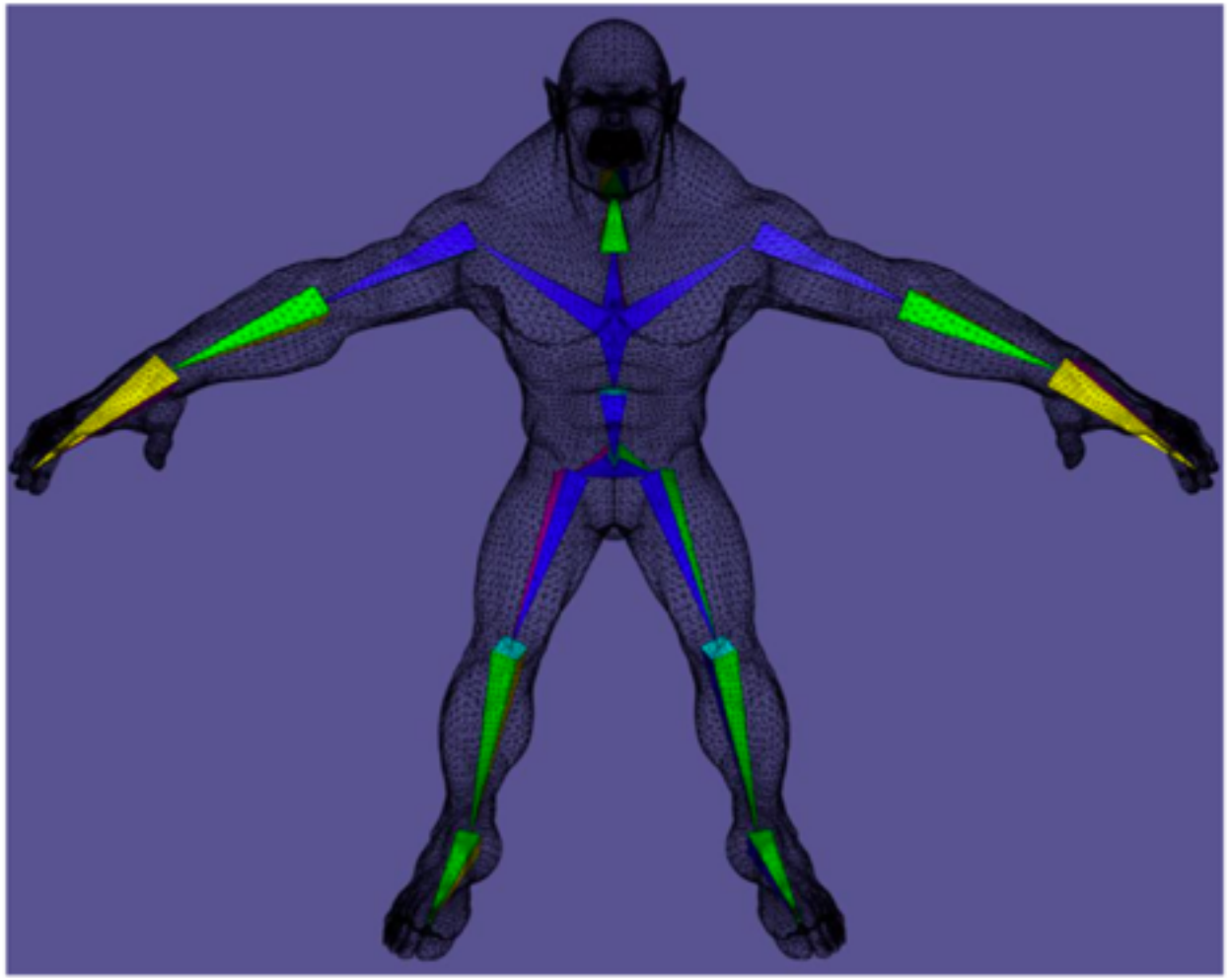
- Often we care about <u>where</u> the character should go, not <u>how</u> to get there
- Very hard to know how to move joints of a complicated figure in order to get the desired pose (esp. in presence of obstacles)



Skeletons: Forward Kinematics

Let the relative rotation at a bone joint *i* be R_i:

$$\mathbf{T}_i = \mathbf{T}_{p_i} \begin{pmatrix} \widehat{\mathbf{T}}_i \\ 0\,0\,0\,1 \end{pmatrix} \begin{pmatrix} \overline{\mathbf{R}}_i & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0\,0\,0 \end{matrix} & 1 \end{pmatrix} \begin{pmatrix} \widehat{\mathbf{T}}_i \\ 0\,0\,0\,1 \end{pmatrix}^{-1}$$

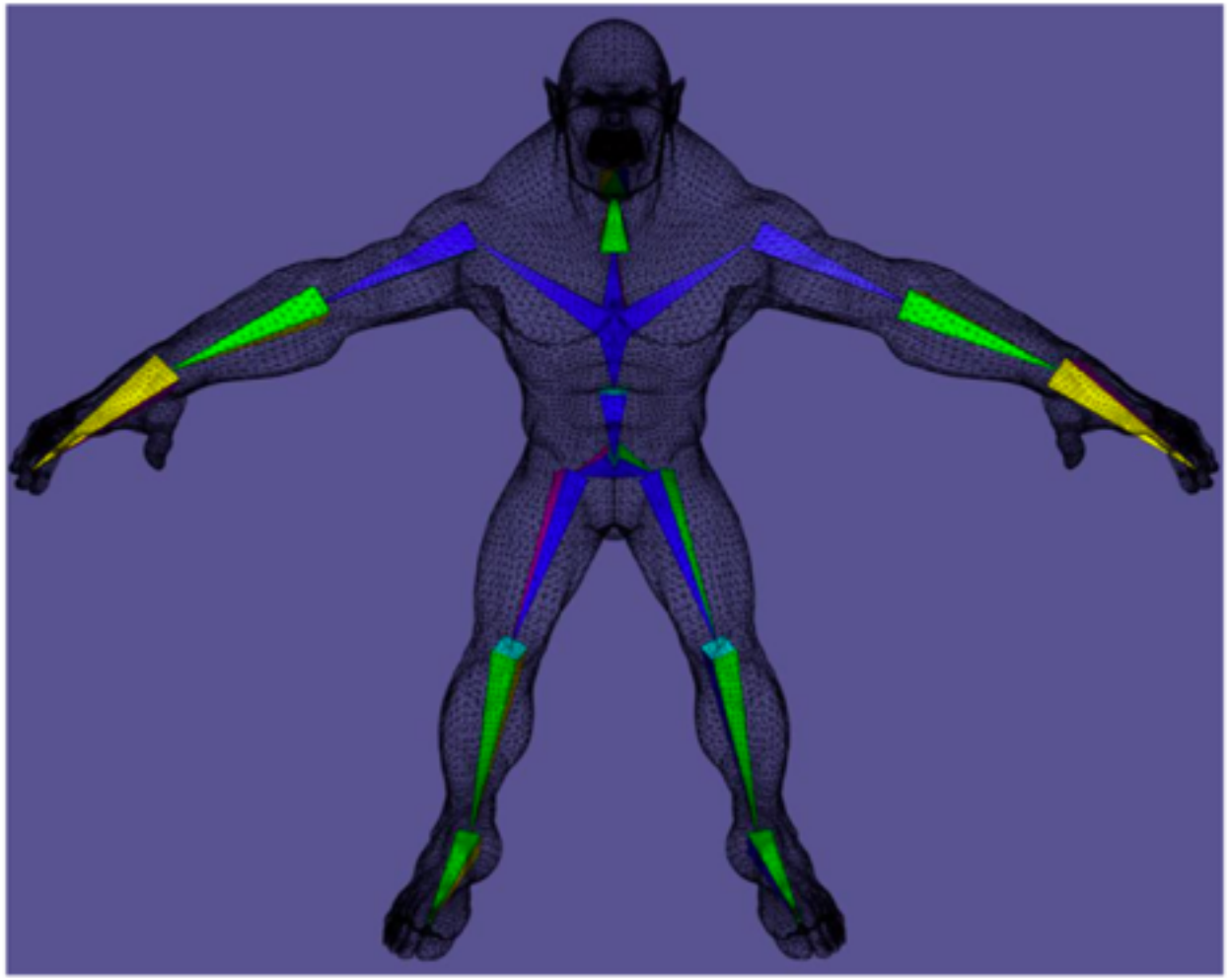


Skeletons: Forward Kinematics

Let the relative rotation at a bone joint *i* be R_i:

$$\mathbf{T}_i = \mathbf{T}_{p_i} \begin{pmatrix} \widehat{\mathbf{T}}_i \\ 0\,0\,0\,1 \end{pmatrix} \begin{pmatrix} \overline{\mathbf{R}}_i & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0\,0\,0 \end{matrix} & 1 \end{pmatrix} \begin{pmatrix} \widehat{\mathbf{T}}_i \\ 0\,0\,0\,1 \end{pmatrix}^{-1}$$

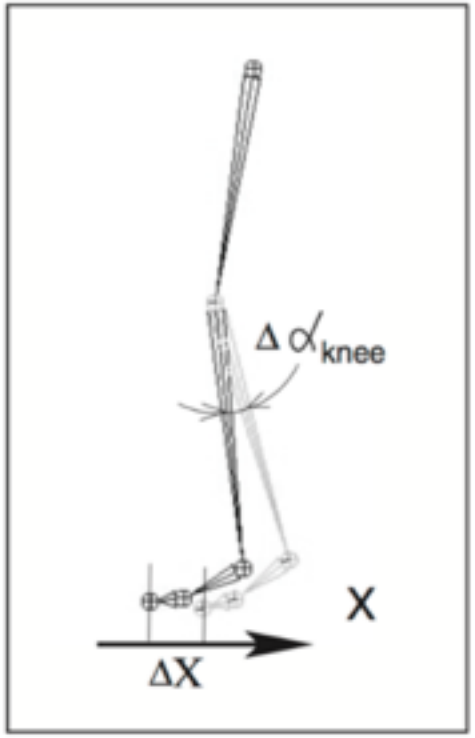
$$\mathbf{T}_i = \mathbf{T}_{p_i} \widehat{\mathbf{T}}_i \begin{pmatrix} \mathbf{R}_z(\theta_{i3}) & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0\,0\,0 \end{matrix} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R}_z(\theta_{i2}) & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0\,0\,0 \end{matrix} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{R}_z(\theta_{i1}) & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \begin{matrix} 0\,0\,0 \end{matrix} & 1 \end{pmatrix} \widehat{\mathbf{T}}_i^{-1}$$



utilize canonical twist-bend-twist rotations (three Euler angles, θ_1,θ_2,θ_3).

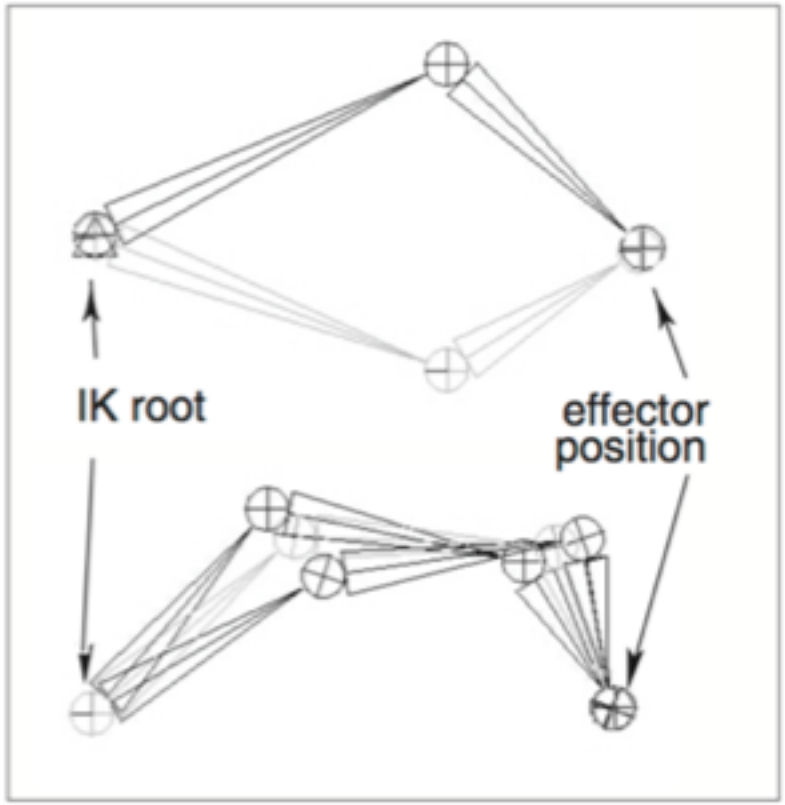
Inverse Kinematics

- Specify where character should go, then deduce join motion
- Position of the end effector is set by the animator.
- Automatically solve for joint angles that will result in that effector position.
- Solution is not usually “closed form”, iteratively solved by optimizer.



Inverse Kinematics

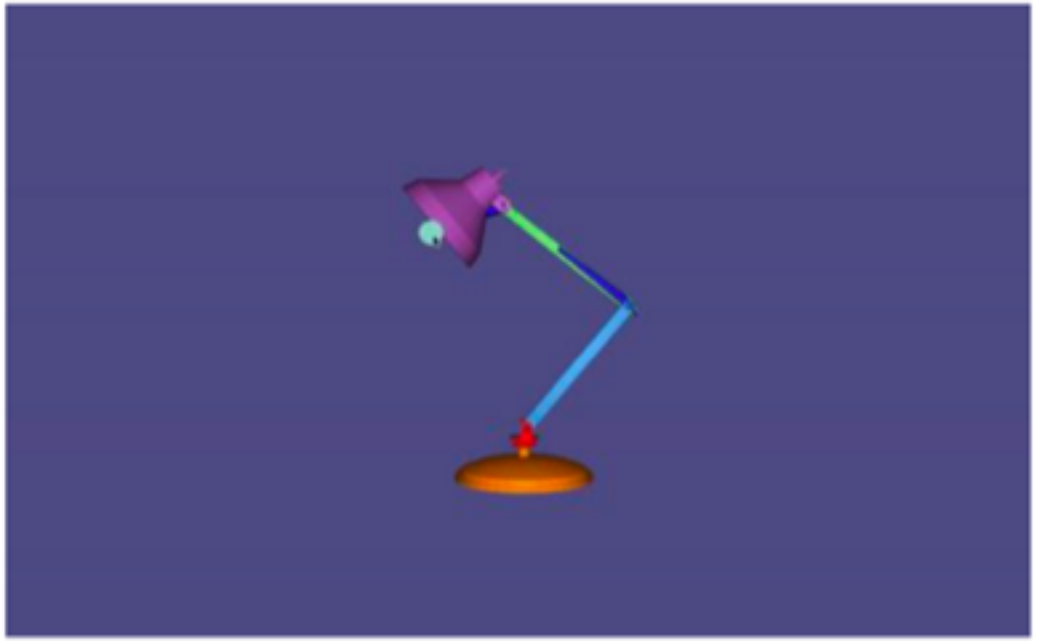
- Typically underconstrained: Multiple configurations of internal joints can result in the same effector position



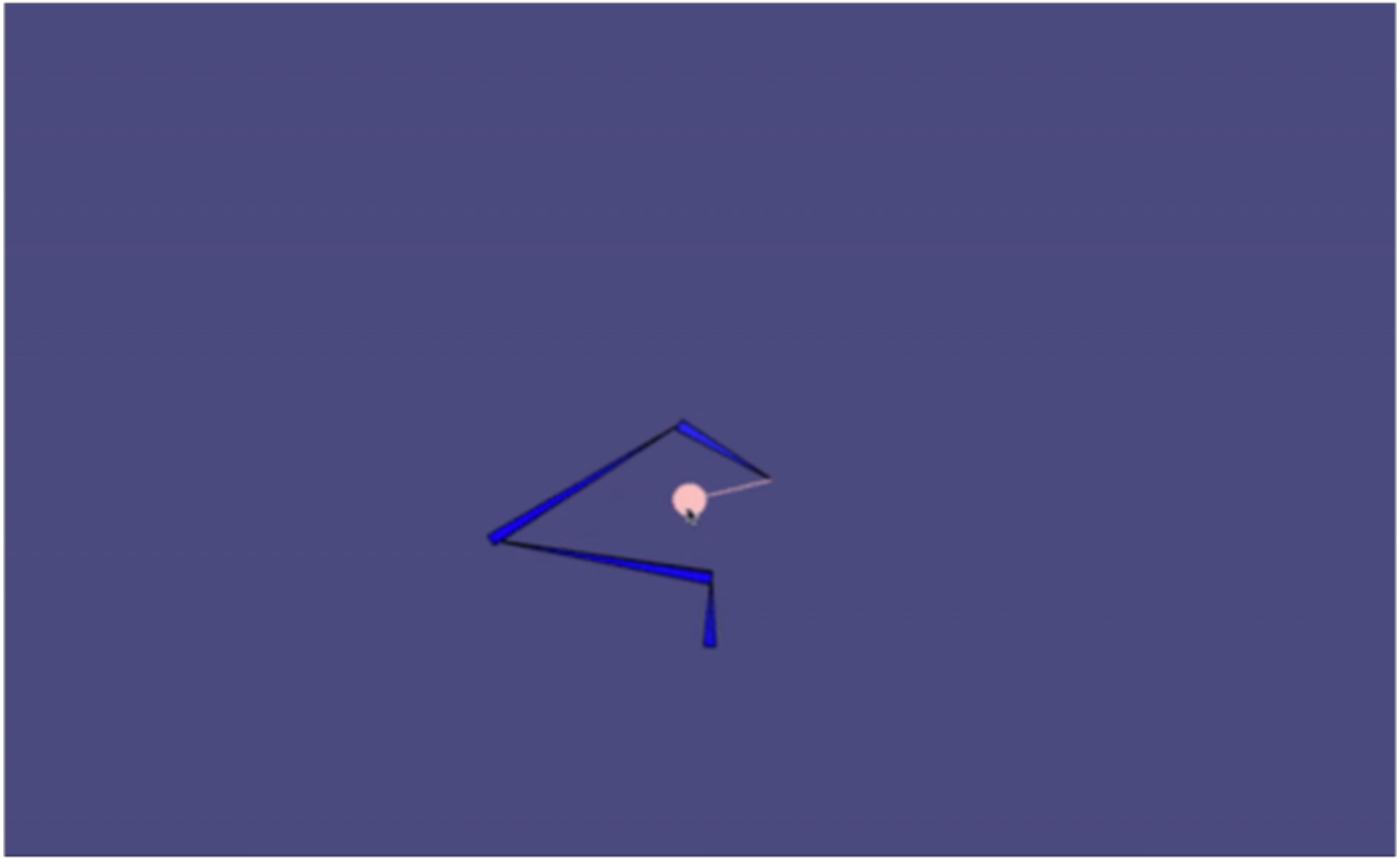
Skeletons: Inverse Kinematics

What is the pose (set of joint angles *a*) that lets us reach a given point (end-effector position).

$$\mathbf{a} = \begin{pmatrix} \theta_{11} \\ \theta_{12} \\ \theta_{13} \\ \theta_{21} \\ \theta_{22} \\ \theta_{23} \\ \vdots \\ \theta_{m1} \\ \theta_{m2} \\ \theta_{m3} \end{pmatrix}$$



What does it mean to reach (get as close as possible) to a point?



Skeletons: Inverse Kinematics

Closeness energy can be measured the squared distance between the pose tip x_b of some bone b and a desired goal location $q \in R^3$.

$$E(\mathbf{x}_b) = \|\mathbf{x}_b - \mathbf{q}\|^2.$$

Skeletons: Inverse Kinematics

Closeness energy can be measured the squared distance between the pose tip x_b of some bone b and a desired goal location $q \in R^3$.

$$E(\mathbf{x}_b) = \|\mathbf{x}_b - \mathbf{q}\|^2.$$

Given pose vector a, the bone tip x_b is:

$$\mathbf{x}_b(\mathbf{a}) = \mathbf{T}_b \widehat{\mathbf{d}}_b$$

Skeletons: Inverse Kinematics

Closeness energy can be measured the squared distance between the pose tip x_b of some bone b and a desired goal location $q \in R^3$.

$$E(\mathbf{x}_b) = \|\mathbf{x}_b - \mathbf{q}\|^2.$$

Given pose vector a, the bone tip x_b is:

$$\mathbf{x}_b(\mathbf{a}) = \mathbf{T}_b \widehat{\mathbf{d}}_b$$

Now given any number of end-effectors $b_1 .. b_k$:

$$\min_{\mathbf{a}} \underbrace{\sum_{i=1}^{k} \|\mathbf{x}_{b_i}(\mathbf{a}) - \widehat{\mathbf{x}}_{b_i}\|^2}_{E(\mathbf{x}_b(\mathbf{a}))}$$

Skeletons: Inverse Kinematics

Closeness energy can be measured the squared distance between the pose tip x_b of some bone b and a desired goal location $q \in R^3$.

$$E(\mathbf{x}_b) = \|\mathbf{x}_b - \mathbf{q}\|^2.$$

Given pose vector a, the bone tip x_b is:

$$\mathbf{x}_b(\mathbf{a}) = \mathbf{T}_b \widehat{\mathbf{d}}_b$$

Now given any number of end-effectors $b_1 .. b_k$:

$$\min_{\mathbf{a}} \underbrace{\sum_{i=1}^{k} \|\mathbf{x}_{b_i}(\mathbf{a}) - \widehat{\mathbf{x}}_{b_i}\|^2}_{E(\mathbf{x}_b(\mathbf{a}))}$$

And we impose some joint angle limits: $\min_{\mathbf{a}^{\min} \le \mathbf{a} \le \mathbf{a}^{\max}} E(\mathbf{x}_b(\mathbf{a}))$

Minimizing this energy is a non-linear least squares problem.

Skeletons: Inverse Kinematics Minimization

Projected Gradient Descent:

Start with an initial pose *a*, and move in direction of decrease in *E*,
project the pose to stay within limits and iterate towards solution.

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma \left(\frac{dE(\mathbf{x}(\mathbf{a}))}{d\mathbf{a}} \right)^T$$

Skeletons: Inverse Kinematics Minimization

Projected Gradient Descent:

Start with an initial pose *a*, and move in direction of decrease in *E*,
project the pose to stay within limits and iterate towards solution.

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma \left(\frac{dE(\mathbf{x}(\mathbf{a}))}{d\mathbf{a}} \right)^T$$

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma \left(\frac{d\mathbf{x}(\mathbf{a})}{d\mathbf{a}} \right)^T \left(\frac{dE(\mathbf{x})}{d\mathbf{x}} \right)$$ chain rule

$\frac{dE}{d\mathbf{a}} \in \mathbb{R}^{|\mathbf{a}|}$, $\frac{dE}{d\mathbf{x}} \in \mathbb{R}^{|\mathbf{x}|}$, and $\frac{d\mathbf{x}}{d\mathbf{a}} \in \mathbb{R}^{|\mathbf{x}| \times |\mathbf{a}|}$

Skeletons: Inverse Kinematics Minimization

Projected Gradient Descent:

Start with an initial pose *a*, and move in direction of decrease in *E*,
project the pose to stay within limits and iterate towards solution.

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma\left(\frac{dE(\mathbf{x}(\mathbf{a}))}{d\mathbf{a}}\right)^T$$

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma\left(\frac{d\mathbf{x}(\mathbf{a})}{d\mathbf{a}}\right)^T\left(\frac{dE(\mathbf{x})}{d\mathbf{x}}\right)$$ chain rule

$\frac{dE}{d\mathbf{a}} \in \mathbb{R}^{|\mathbf{a}|}$, $\frac{dE}{d\mathbf{x}} \in \mathbb{R}^{|\mathbf{x}|}$, and $\frac{d\mathbf{x}}{d\mathbf{a}} \in \mathbb{R}^{|\mathbf{x}|\times|\mathbf{a}|}$

$$\mathbf{J} = \frac{d\mathbf{x}}{d\mathbf{a}}.$$ also known as Jacobian measures the change in *x* for changes in joint angles *a*,

J can be computed using Finite Differences: $$\mathbf{J}_{i,j} \approx \frac{\mathbf{x}_i(\mathbf{a} + h\delta_j) - \mathbf{x}_i(\mathbf{a})}{h}.$$ $h = 10^{-7}$

Skeletons: Inverse Kinematics Minimization

Projected Gradient Descent:

Start with an initial pose *a*, and move in direction of decrease in *E*,
project the pose to stay within limits and iterate towards solution.

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma\left(\frac{dE(\mathbf{x}(\mathbf{a}))}{d\mathbf{a}}\right)^T$$

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma\left(\frac{d\mathbf{x}(\mathbf{a})}{d\mathbf{a}}\right)^T\left(\frac{dE(\mathbf{x})}{d\mathbf{x}}\right)$$ chain rule

$\frac{dE}{d\mathbf{a}} \in \mathbb{R}^{|\mathbf{a}|}$, $\frac{dE}{d\mathbf{x}} \in \mathbb{R}^{|\mathbf{x}|}$, and $\frac{d\mathbf{x}}{d\mathbf{a}} \in \mathbb{R}^{|\mathbf{x}|\times|\mathbf{a}|}$

$$\mathbf{J} = \frac{d\mathbf{x}}{d\mathbf{a}}.$$ also known as Jacobian measures the change in *x* for changes in joint angles *a*,

J can be computed using Finite Differences: $$\mathbf{J}_{i,j} \approx \frac{\mathbf{x}_i(\mathbf{a} + h\delta_j) - \mathbf{x}_i(\mathbf{a})}{h}.$$ $h = 10^{-7}$

$\left(\frac{dE(\mathbf{x})}{d\mathbf{x}}\right)$ is gradient of $\sum_{i=1}^{k} \|\mathbf{x}_{b_i}(\mathbf{a}) - \widehat{\mathbf{x}}_{b_i}\|^2$

Skeletons: Inverse Kinematics Minimization

Projected Gradient Descent:

Start with an initial pose *a*, and move in direction of decrease in *E*,
project the pose to stay within limits and iterate towards solution.

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma\left(\frac{dE(\mathbf{x}(\mathbf{a}))}{d\mathbf{a}}\right)^T$$

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma\left(\frac{d\mathbf{x}(\mathbf{a})}{d\mathbf{a}}\right)^T\left(\frac{dE(\mathbf{x})}{d\mathbf{x}}\right)$$ chain rule

$\frac{dE}{d\mathbf{a}} \in \mathbb{R}^{|\mathbf{a}|}$, $\frac{dE}{d\mathbf{x}} \in \mathbb{R}^{|\mathbf{x}|}$, and $\frac{d\mathbf{x}}{d\mathbf{a}} \in \mathbb{R}^{|\mathbf{x}|\times|\mathbf{a}|}$

$$\mathbf{J} = \frac{d\mathbf{x}}{d\mathbf{a}}.$$ also known as Jacobian measures the change in *x* for changes in joint angles *a*,

J can be computed using Finite Differences: $$\mathbf{J}_{i,j} \approx \frac{\mathbf{x}_i(\mathbf{a} + h\delta_j) - \mathbf{x}_i(\mathbf{a})}{h}.$$ $h = 10^{-7}$

$\left(\frac{dE(\mathbf{x})}{d\mathbf{x}}\right)$ is gradient of $\sum_{i=1}^{k} \|\mathbf{x}_{b_i}(\mathbf{a}) - \widehat{\mathbf{x}}_{b_i}\|^2$

Project to within limits: $\mathbf{a}_i \leftarrow \max[\mathbf{a}_i^{\min}, \min[\mathbf{a}_i^{\max}, \mathbf{a}_i]]$.

Skeletons: Inverse Kinematics Minimization

Projected Gradient Descent:
Start with an initial pose *a*, and move in direction of decrease in *E*,
project the pose to stay within limits and iterate towards solution.

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma\left(\frac{dE(\mathbf{x}(\mathbf{a}))}{d\mathbf{a}}\right)^T$$

$$\mathbf{a} \leftarrow \mathbf{a} - \sigma\left(\frac{d\mathbf{x}(\mathbf{a})}{d\mathbf{a}}\right)^T\left(\frac{dE(\mathbf{x})}{d\mathbf{x}}\right)$$ chain rule

$\frac{dE}{d\mathbf{a}} \in \mathbb{R}^{|\mathbf{a}|}$, $\frac{dE}{d\mathbf{x}} \in \mathbb{R}^{|\mathbf{x}|}$, and $\frac{d\mathbf{x}}{d\mathbf{a}} \in \mathbb{R}^{|\mathbf{x}|\times|\mathbf{a}|}$

$$\mathbf{J} = \frac{d\mathbf{x}}{d\mathbf{a}}.$$ also known as Jacobian measures the change in *x* for changes in joint angles *a*,

J can be computed using Finite Differences: $$\mathbf{J}_{i,j} \approx \frac{\mathbf{x}_i(\mathbf{a} + h\delta_j) - \mathbf{x}_i(\mathbf{a})}{h}.$$ $h = 10^{-7}$

$\left(\frac{dE(\mathbf{x})}{d\mathbf{x}}\right)$ is gradient of $\sum_{i=1}^{k} \|\mathbf{x}_{b_i}(\mathbf{a}) - \widehat{\mathbf{x}}_{b_i}\|^2$

Project to within limits: $\mathbf{a}_i \leftarrow \max[\mathbf{a}_i^{\min}, \min[\mathbf{a}_i^{\max}, \mathbf{a}_i]]$.

Find a good step that lowers energy: $E(\text{proj}(\mathbf{a} + \sigma\,\Delta\,\mathbf{a})) < E(\mathbf{a})$.

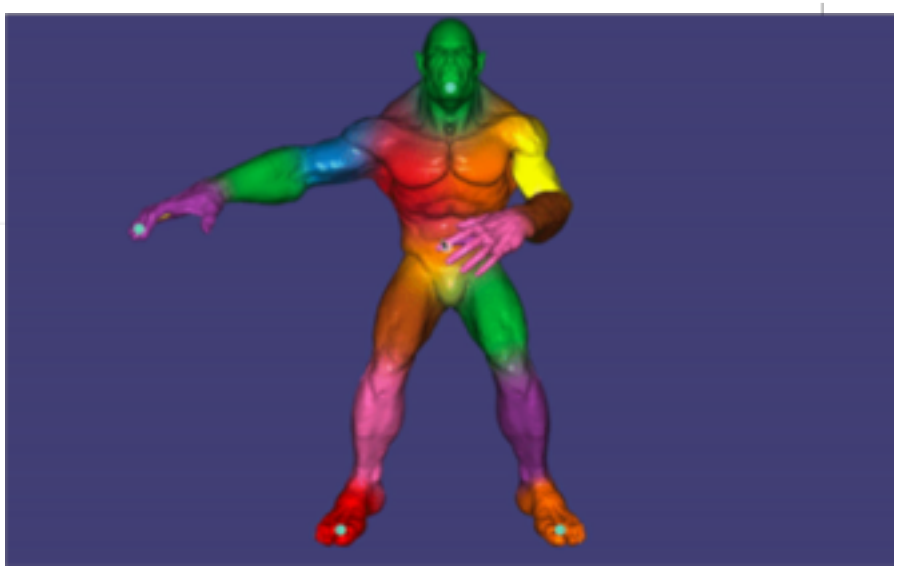
Linear Blend Skinning

Rigid Skinning: objects/points are controlled by a single bone.

Smooth Skinning: weights for multiple bones influence a point *v*.

$$\mathbf{v}_i = \sum_{j=1}^{m} w_{i,j} \mathbf{T}_j \begin{pmatrix} \hat{\mathbf{v}}_i \\ 1 \end{pmatrix}.$$

The final position is a weighted average of the position that each bone wants



Lecture 8: Topic 1: Animation

- Motivation
- Key principles of animation
- Elements of CG
  - Keyframing
  - Keyframing: Character Animation
    - Rig (Skeleton)
    - Forward Kinematics
    - Inverse Kinematics
  - Physical Simulation
  - Motion Capture

Physical Simulation

- Mathematically model real-world motion
- Animate via simulation
- Smoke, fire, clouds, fluids, cloth, rigid bodies, elastic objects.

Pros:

- Once implemented, easy to specify the state of the system at a particular time.
- Model can be easily changed via parameter settings
- High degree of physical realism

Cons:

- Complex systems used to model natural phenomena
- Computationally intensive

Physical Simulation

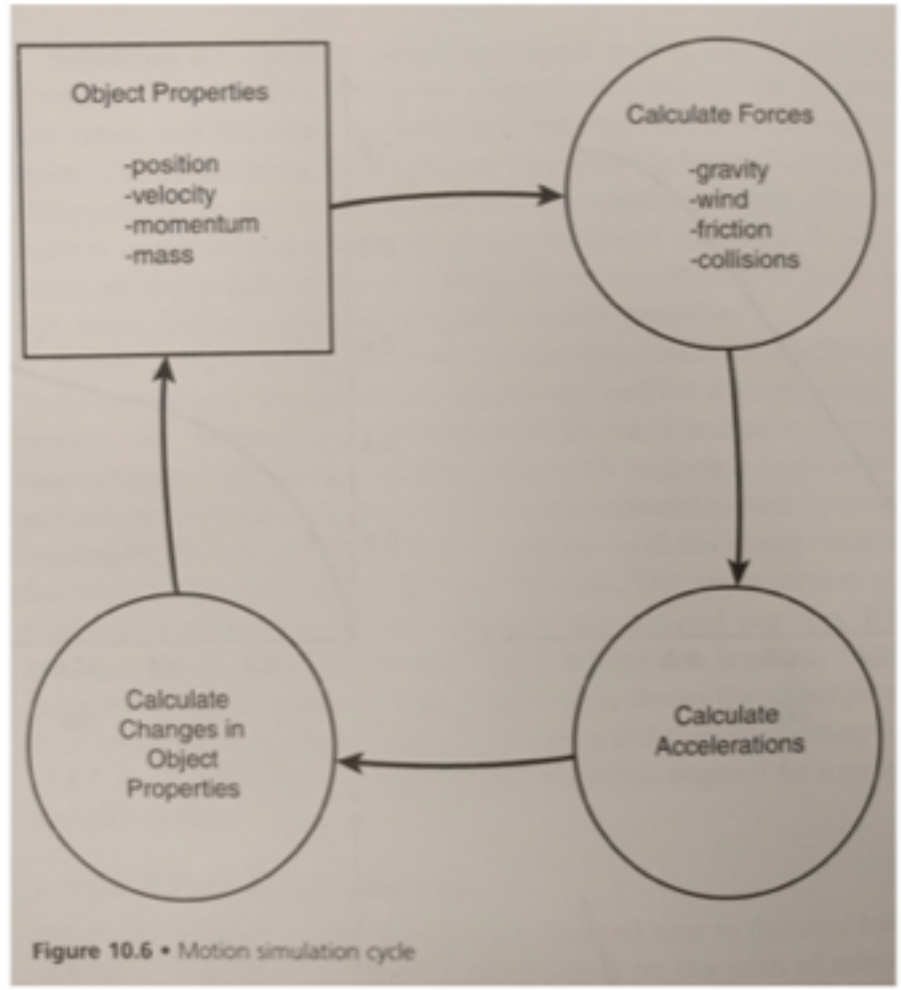
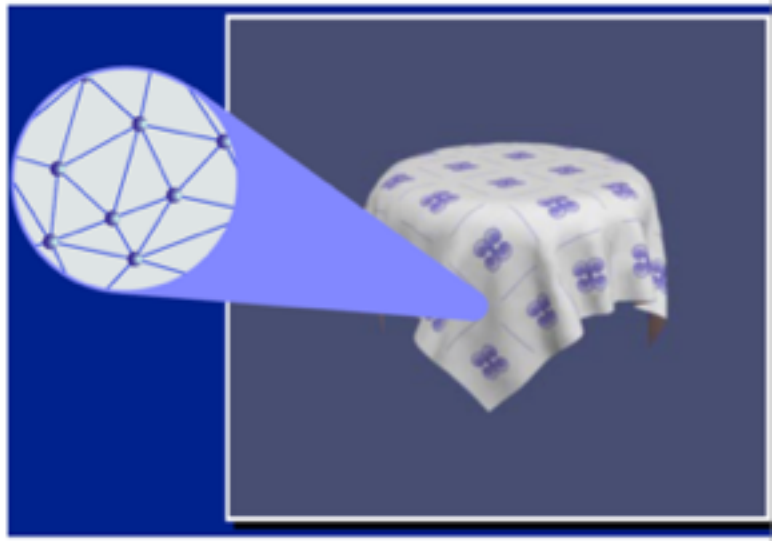
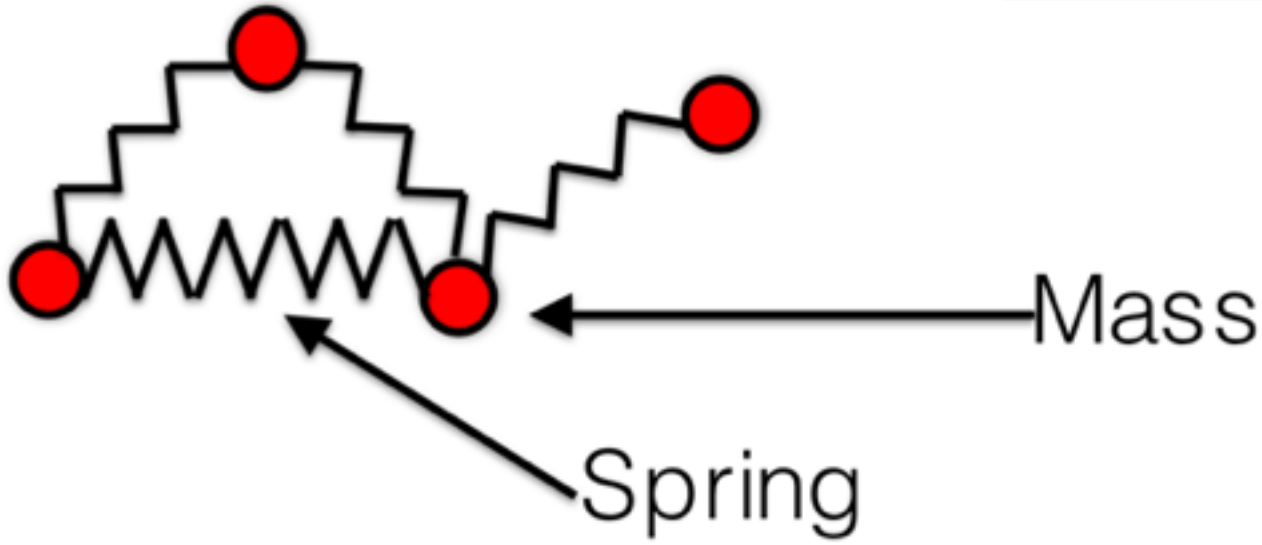


Figure 10.6 • Motion simulation cycle

Mass Spring Systems

- One way of modeling deformable objects is as a network of masses and springs





The Motion of a Mass Spring Systems

- The acceleration of a point mass is given by:

$$m\mathbf{a} = \mathbf{f}$$

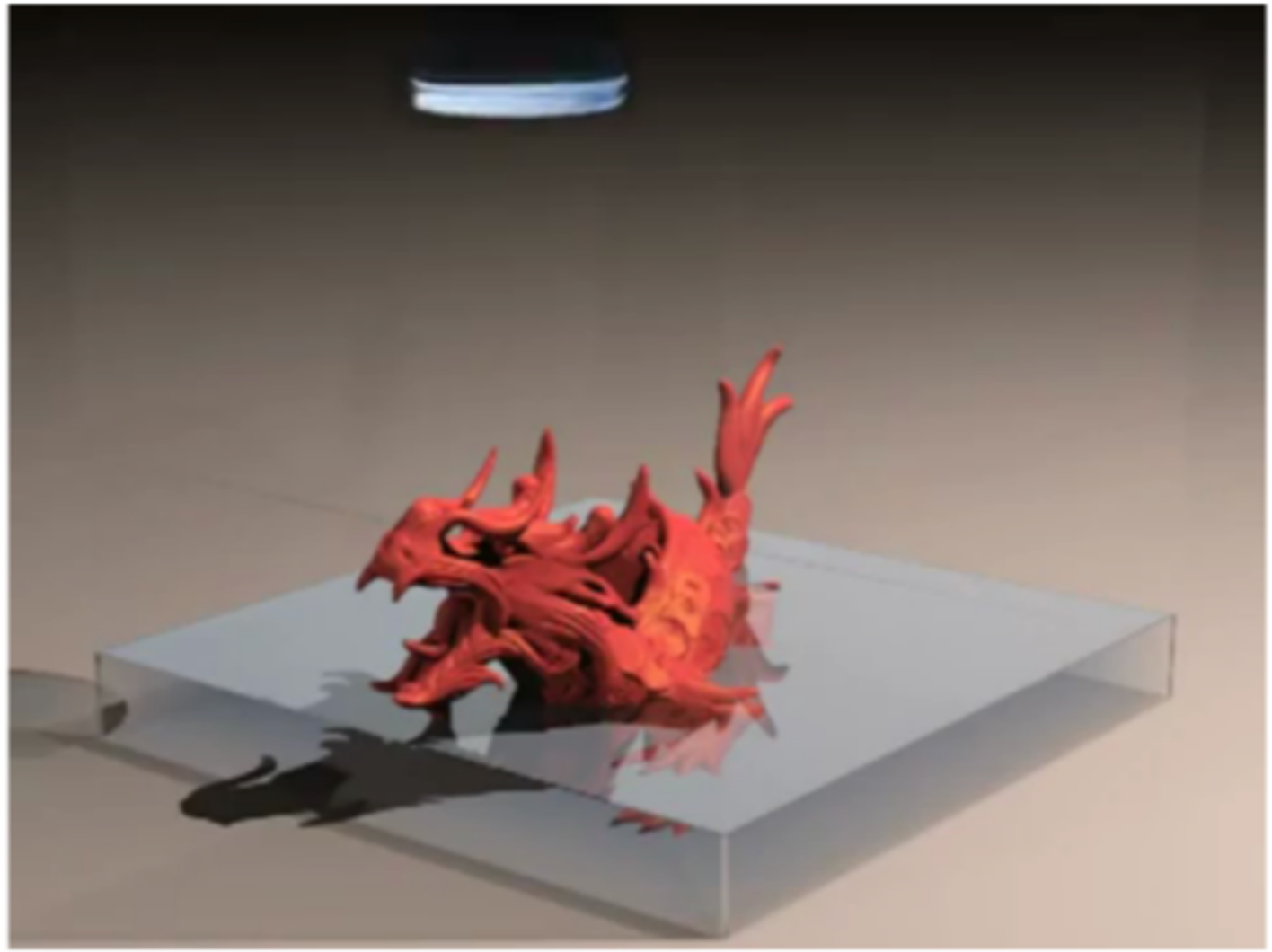
- We can use standard methods to integrate this system forward in time in order to compute the velocities and displacements of each point in the mass spring system

Simit *A Language for Physical Simulation*



Fredrik Kjolstad, Shoaib Kamil, Jonathan Ragan-Kelley, David Levin, Shinjiro Sueda
Desai Chen, Etienne Vouga, Danny Kaufman, Gurtej Kanwar
Wojciech Matusik and Saman Amarasinghe

Physical Simulation: Fluids



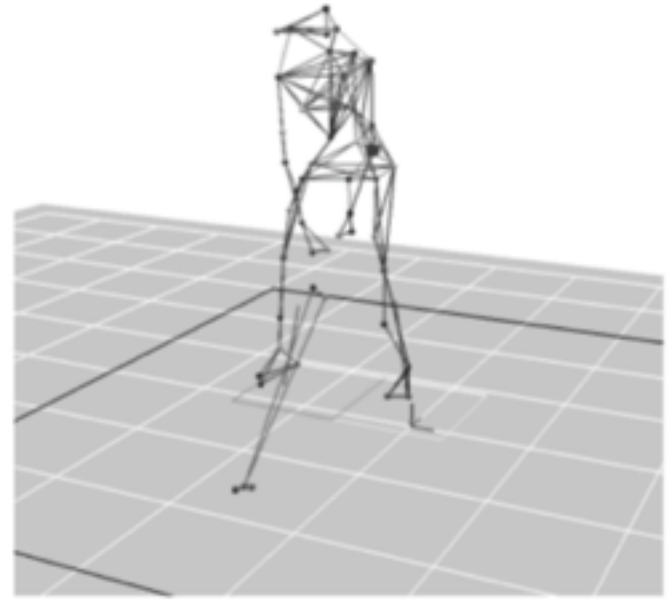
Lecture 8: Topic 1: Animation

- Motivation
- Key principles of animation
- Elements of CG
  - Keyframing
  - Keyframing: Character Animation
    - Rig (Skeleton)
    - Forward Kinematics
    - Inverse Kinematics
  - Physical Simulation
  - Motion Capture

Motion Capture

- Record motions of real people/objects then transfer to digital characters.
- Markers can be occluded. Use multiple cameras and interpolate where needed.
- Noise can cause limbs to loose contact with ground or other objects. Use redundant markers to reduce noise. Correct with Inverse Kinematics.
- Retargeting: applying recorded motion to different characters.

Motion Capture

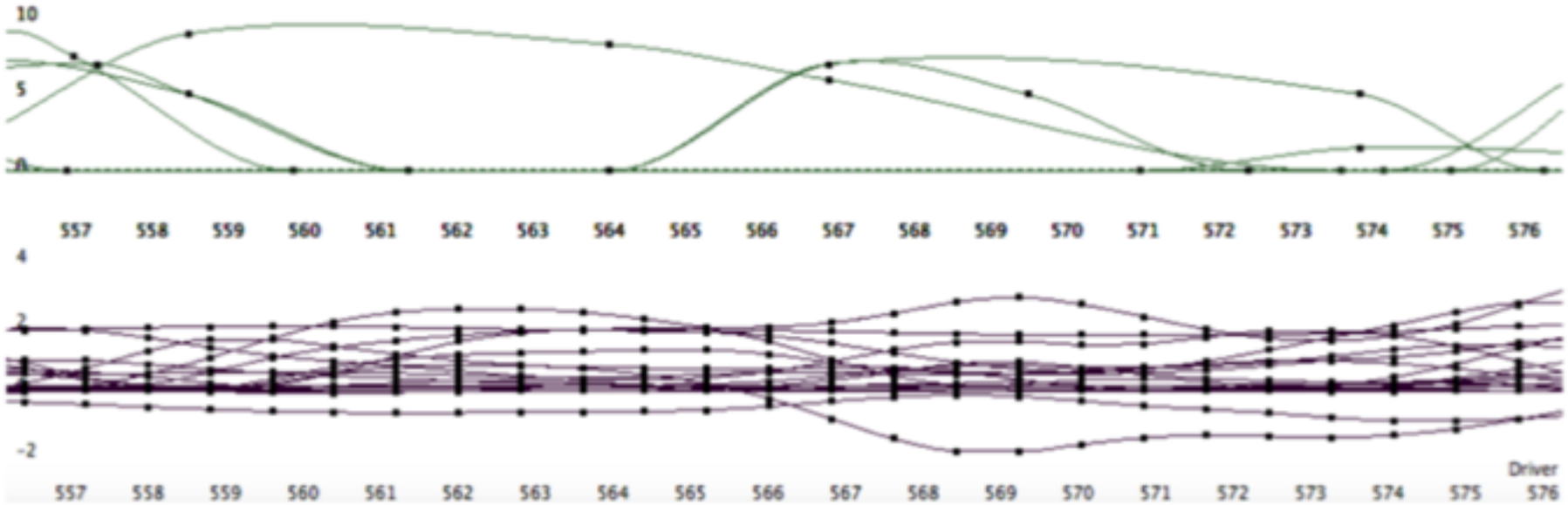



AA 06
15:22:31:18

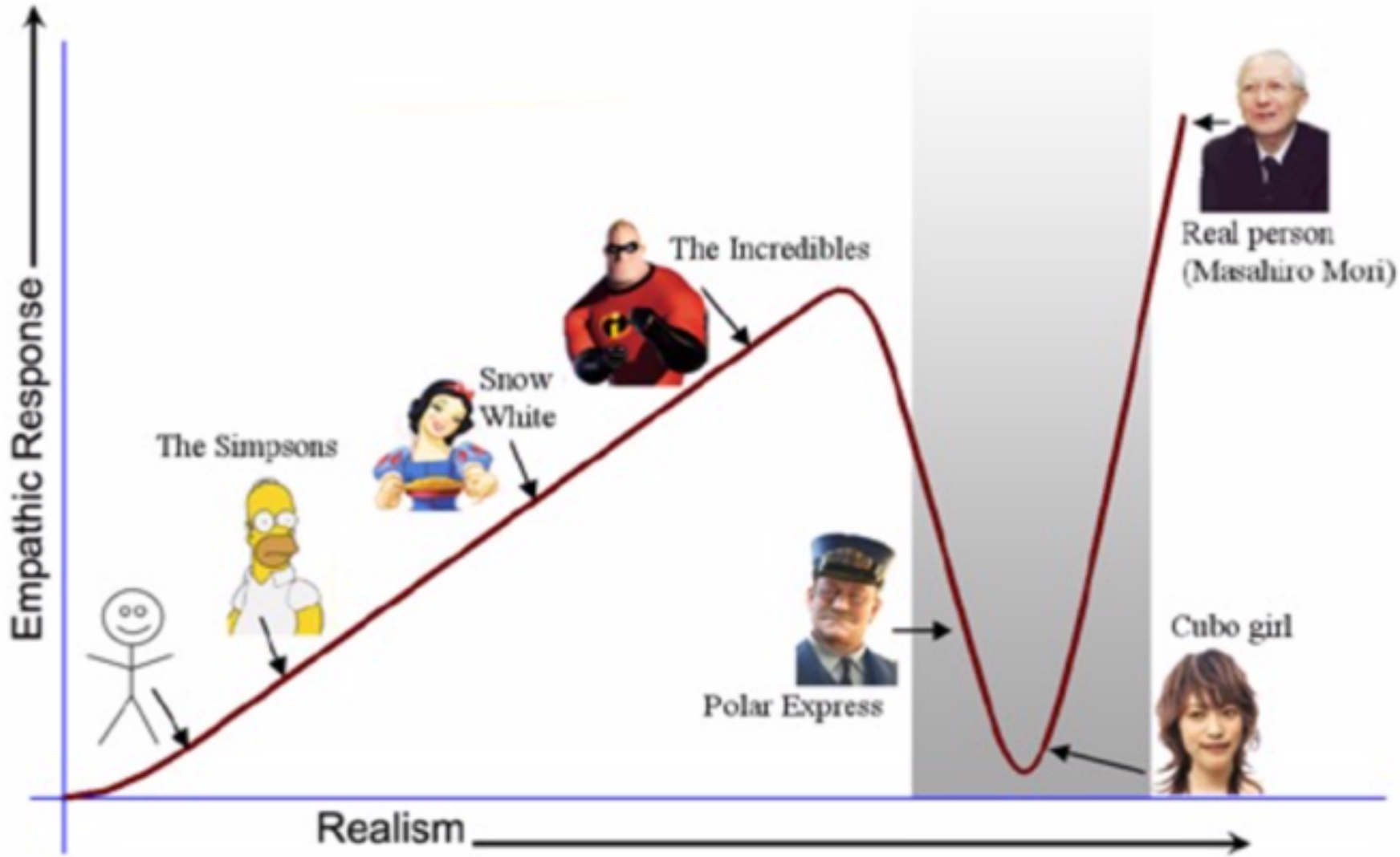
Motion Capture

Motion Capture

- Sparse = Easier to: Edit, Evaluate, Understand
- Dense = Realism, complexity, accuracy.



Uncanny Valley



Uncanny Valley

Topic 2: Curves

Lecture 8: Topic 2: Curves

- Motivation
- Interpolating vs approximating
- Properties and continuity
- Polynomial interpolation
- Bezier curves
- Splines

Lecture 8: Topic 2: Curves

- Motivation
- Interpolating vs approximating
- Properties and continuity
- Polynomial interpolation
- Bezier curves
- Splines

Motivation

Animation is the perfect motivation!

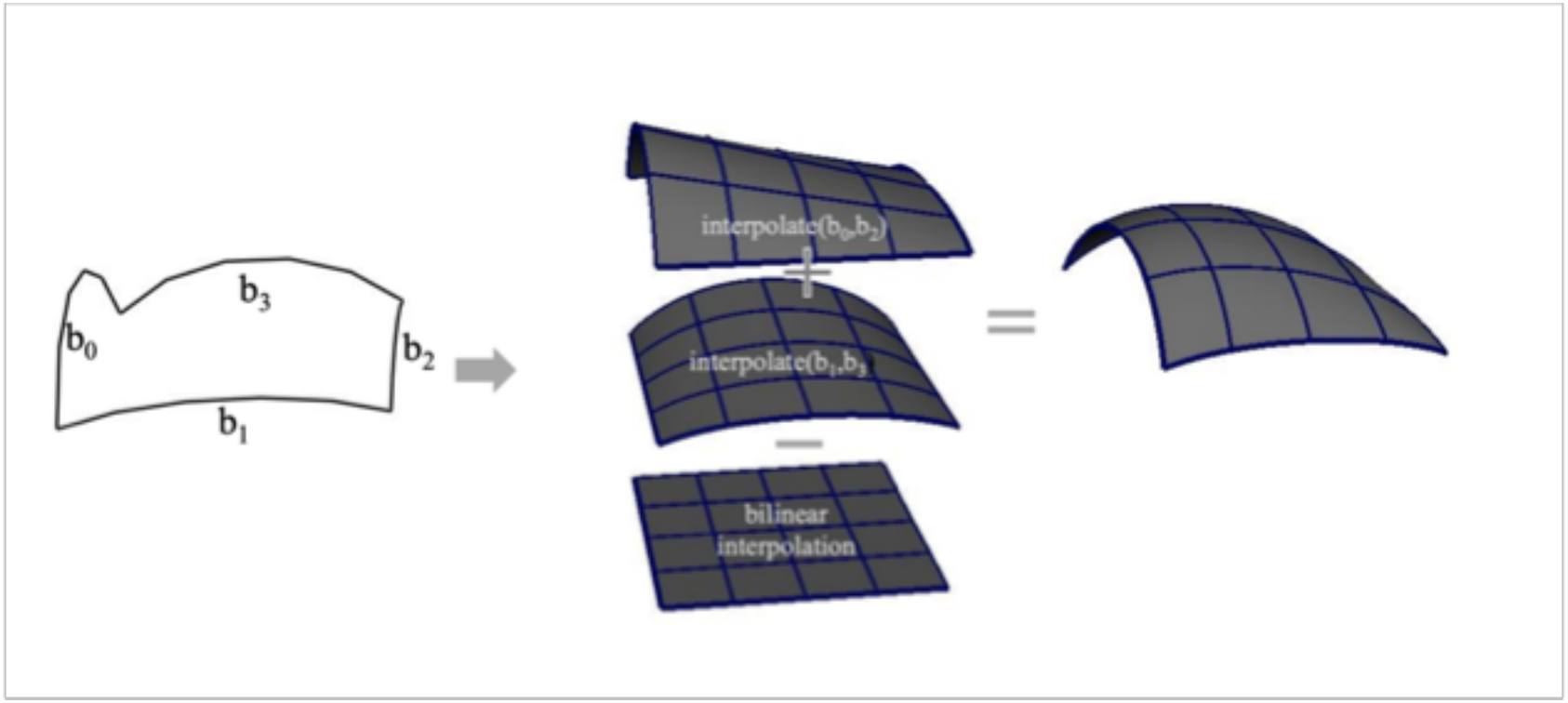
To interpolate keyframes we need curves!

Any animation parameter can be a curve over time:

- Position (eg. x(t), y(t), z(t))
- Rotation (eg. Θ(t))
- Color (r(t), g(t), b(t))
- … etc.

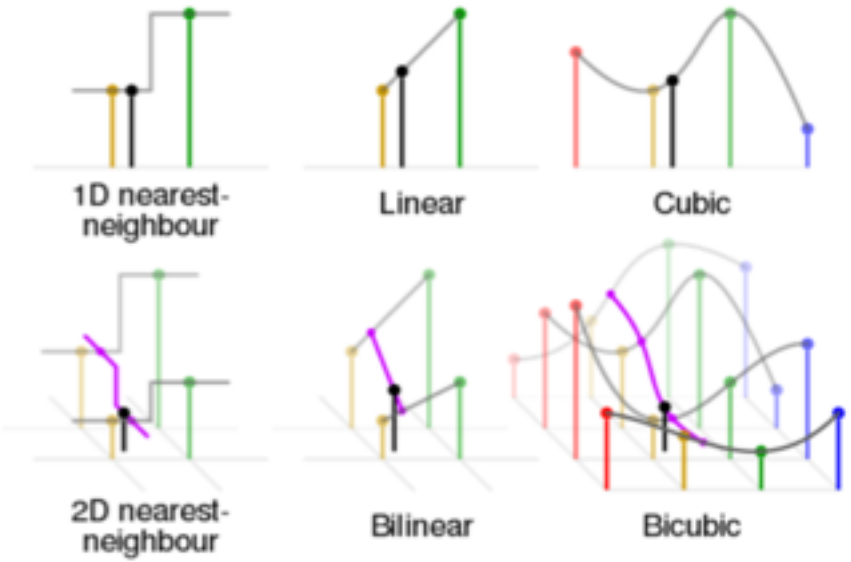
Motivation

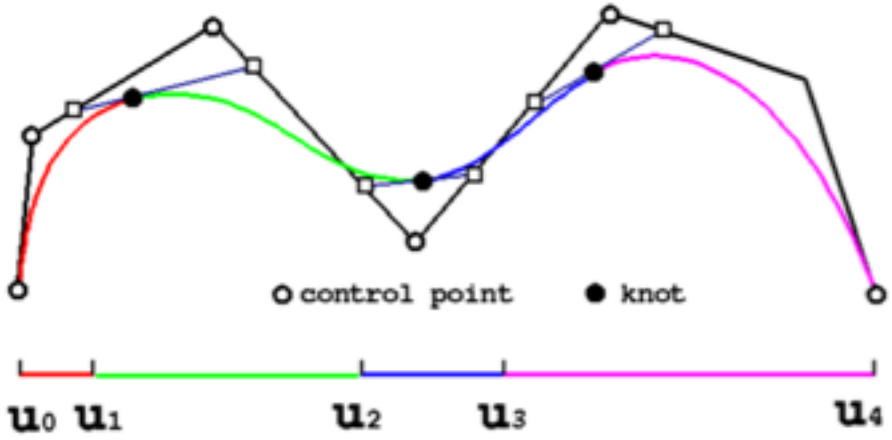
- We can also make shapes with curves



What are Splines?

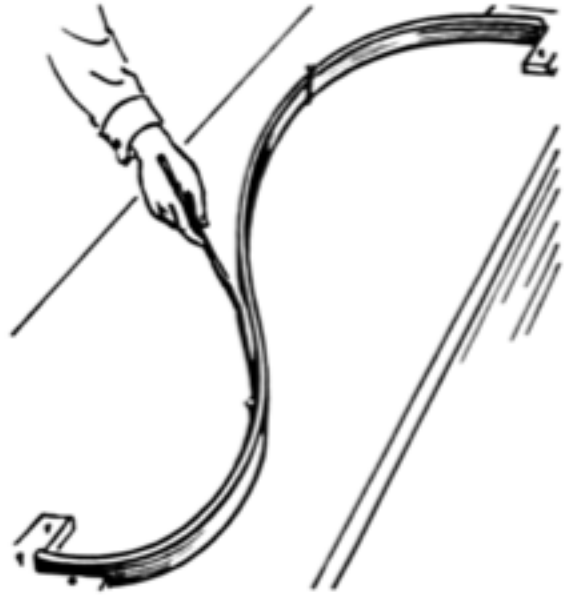
- Numeric function that is piecewise-defined by polynomial functions
- Possesses a high degree of smoothness where pieces connect
- These are intuitively called “knots”





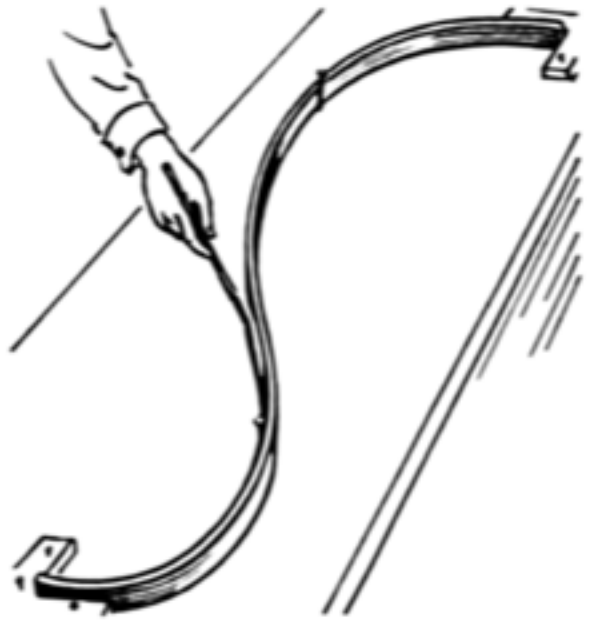
History

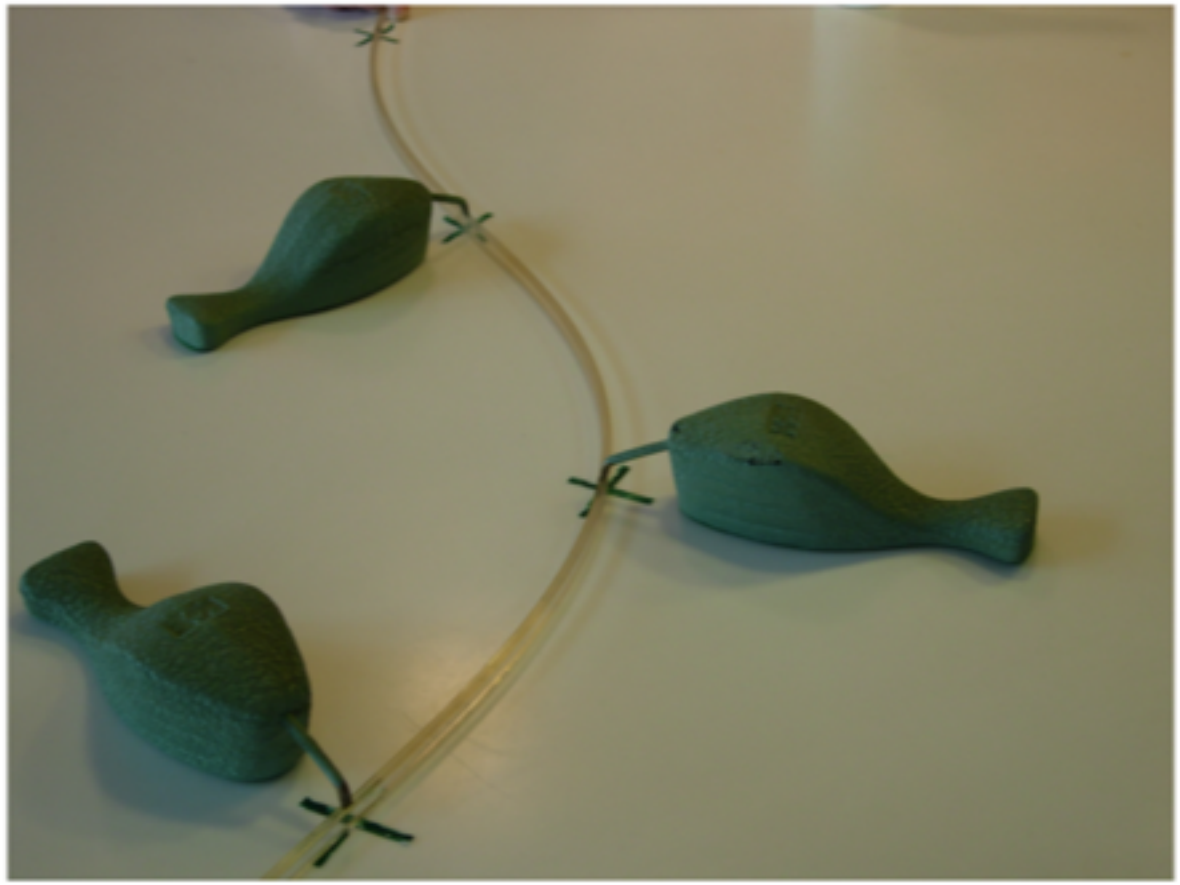
- Used by engineers in ship building and airplane design before computers were around
- Used to create smoothly varying curves
- Variations in curve achieved by the use of weights (like control points)



History

- Used by engineers in ship building and airplane design before computers were around
- Used to create smoothly varying curves
- Variations in curve achieved by the use of weights (like control points)





Applications

- Specify smooth camera path in scene along spline curve
- Rollercoaster tracks
- Curved smooth bodies and shells (planes, boats, etc)

Design issues

- Create curves that can have constraints specified
- Have natural and intuitive interaction
- Controllable smoothness
- Control (local vs global)
- Analytic derivatives that are easy to compute
- Compactly represented
- Other geometric properties (planarity, tangent/curvature control)

Lecture 8: Topic 2: Curves

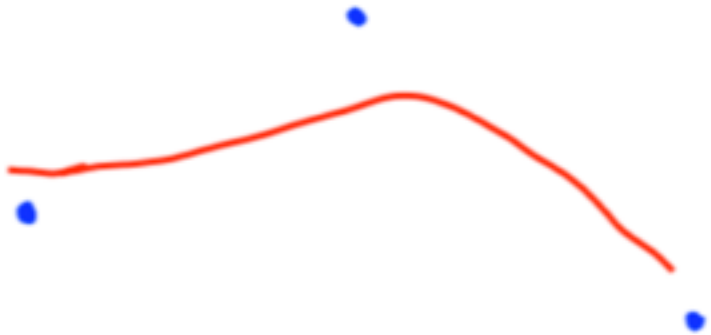
- Motivation
- Interpolating vs approximating
- Properties and continuity
- Polynomial interpolation
- Bezier curves
- Splines

Interpolation

- Interpolating splines: pass through all the data points (control points).
  Example: Hermite splines

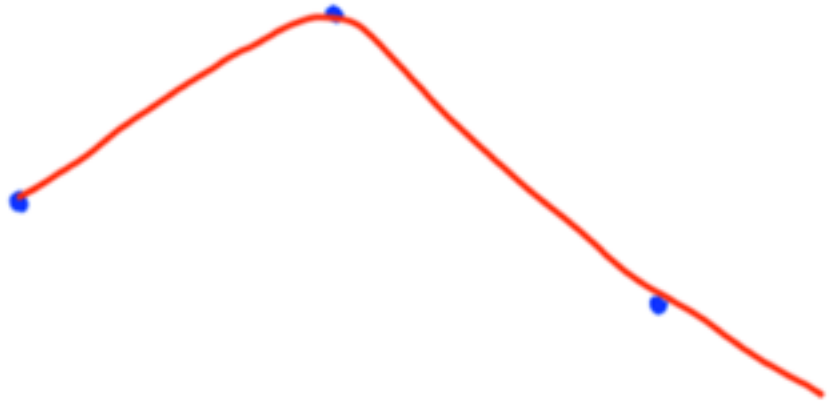
Approximation

- Curve approximates but does not go through all of the control points.
- Comes close to them.



Extrapolation

- Extend the curve beyond the domain of the control points



Lecture 8: Topic 2: Curves

- Motivation
- Interpolating vs approximating
- Properties and continuity
- Polynomial interpolation
- Bezier curves
- Splines

Local properties

- Continuity
- Position at a specific place on the curve
- Direction at a specific place on the curve
- Curvature

Global properties

- Closed or open curve
- Self intersection
- Length

Local vs Global Control

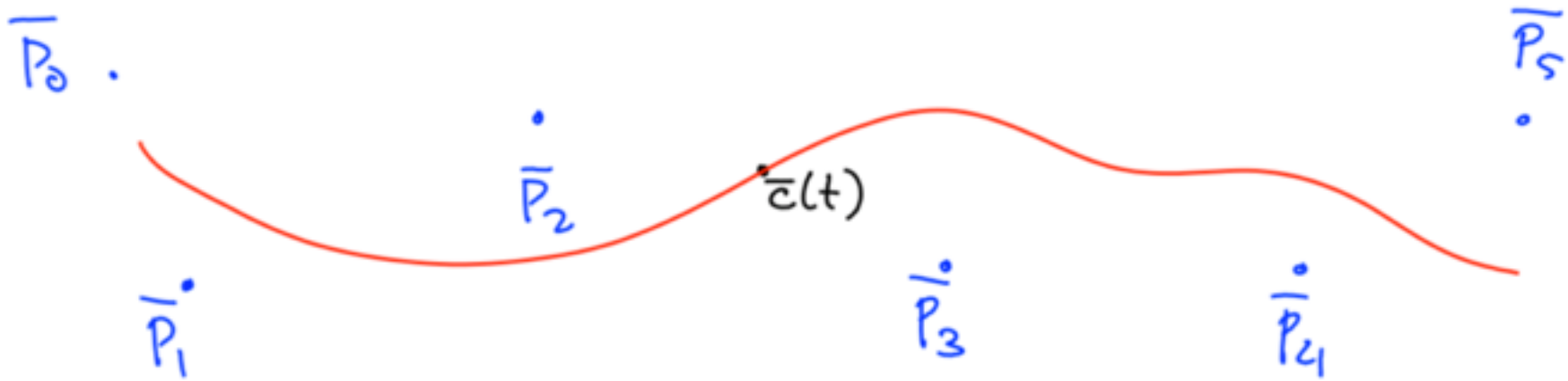
- Local control changes curve only locally while maintaining some constraints
- Modifying point on curve affects local part of curve or entire curve

Parametric and Geometric Continuity

- When piecing together smooth curves, consider the degrees of smoothness at the joints.
- Parametric Continuity: differentiability of the parametric representation (C^0, C^1, C^2, ...)
- Geometric Continuity: smoothness of the resulting displayed shape ($G^0=C^0$, G^1=tangent-cont., G^2=curvature-cont.)

2D Curve Design: General Problem Statement

- Given N control points, P_i, $i = 0 \ldots n - 1$, $t \in [0, 1]$ (by convention)
- Define a curve c(t) that interpolates / approximates them
- Compute its derivatives (and tangents, normals etc)



Linear Interpolation

- The simplest possible interpolation technique
- Create a piecewise linear curve that connects the control points

- Q: What is the disadvantage of such a technique?

Linear Interpolation

- The simplest possible interpolation technique
- Create a piecewise linear curve that connects the control points

- Q: What is the disadvantage of such a technique?
- A: The curves may be continuous but its derivatives are not…

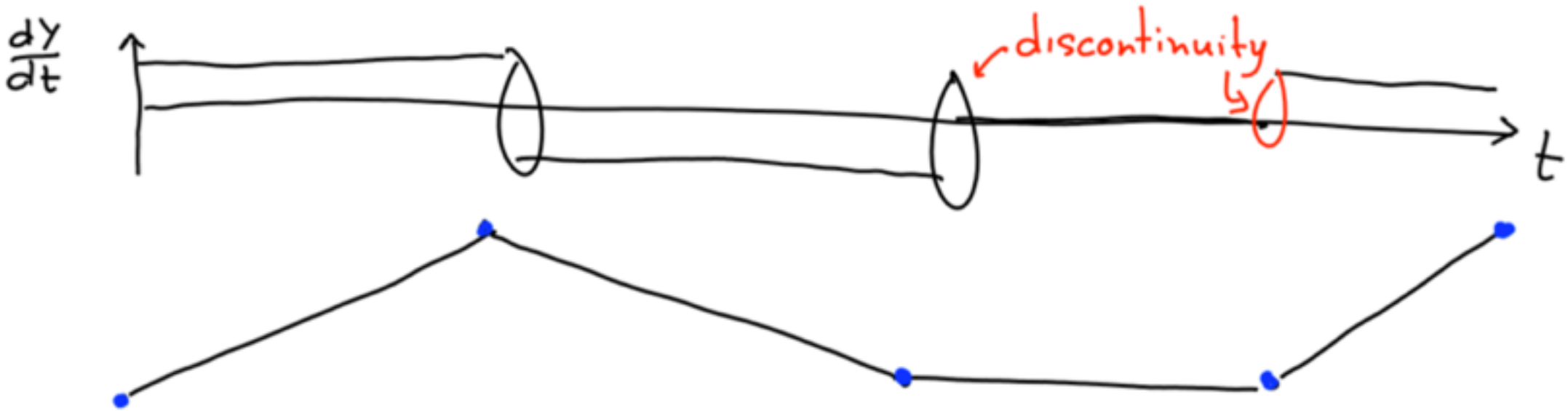
Linear Interpolation

- The simplest possible interpolation technique
- Create a piecewise linear curve that connects the control points

- Q: What is the disadvantage of such a technique?
- A: The curves may be continuous but its derivatives are not…

Linear Interpolation

- The simplest possible interpolation technique
- Create a piecewise linear curve that connects the control points



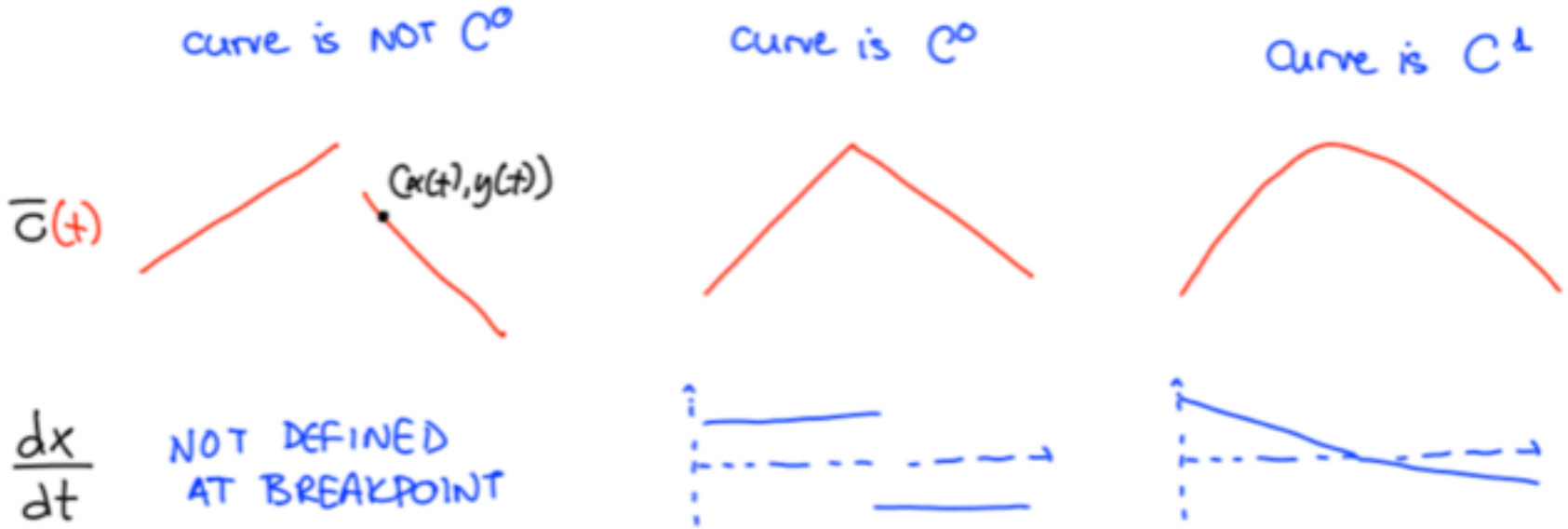
- Q: What is the disadvantage of such a technique?
- A: The curves may be continuous but its derivatives are not…

C^n continuity

- Definition: a function is called C^n if it's n^{th} order derivative is continuous everywhere

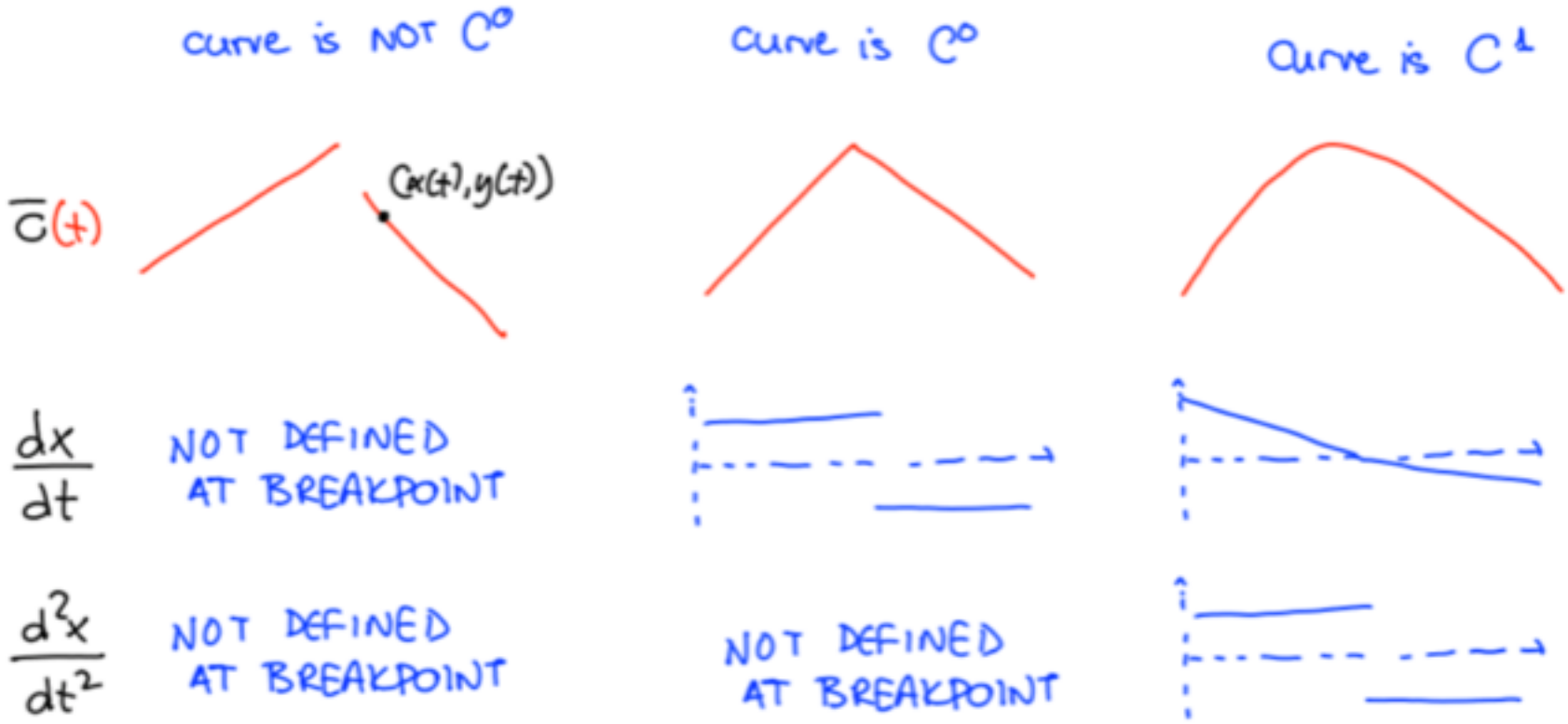
C^n continuity

- Definition: a function is called C^n if it's n^{th} order derivative is continuous everywhere



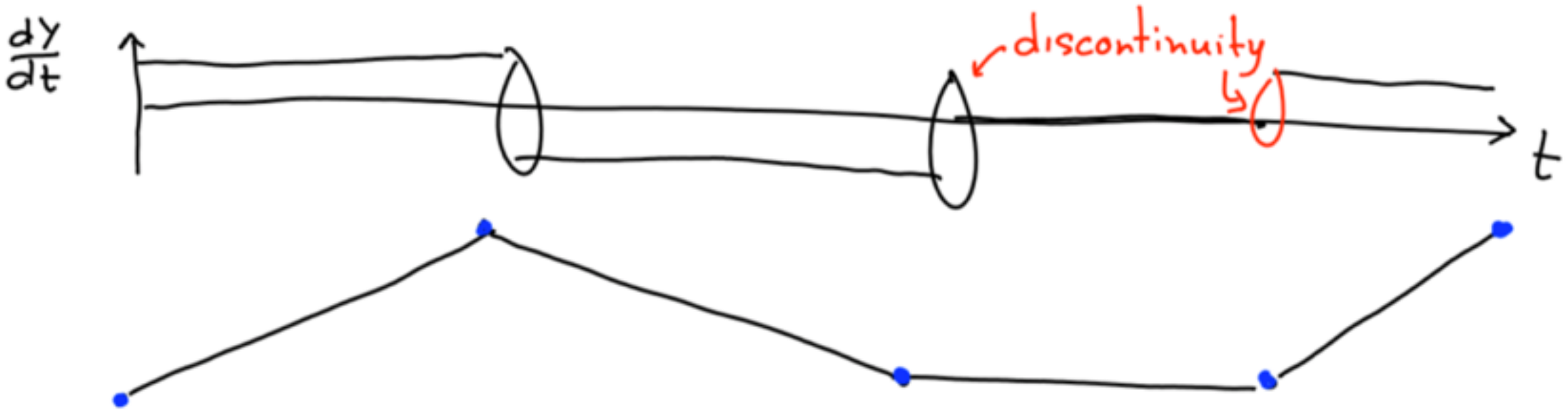
C^n continuity

- Definition: a function is called C^n if it's n^{th} order derivative is continuous everywhere



Linear Interpolation

- The simplest possible interpolation technique
- Create a piecewise linear curve that connects the control points

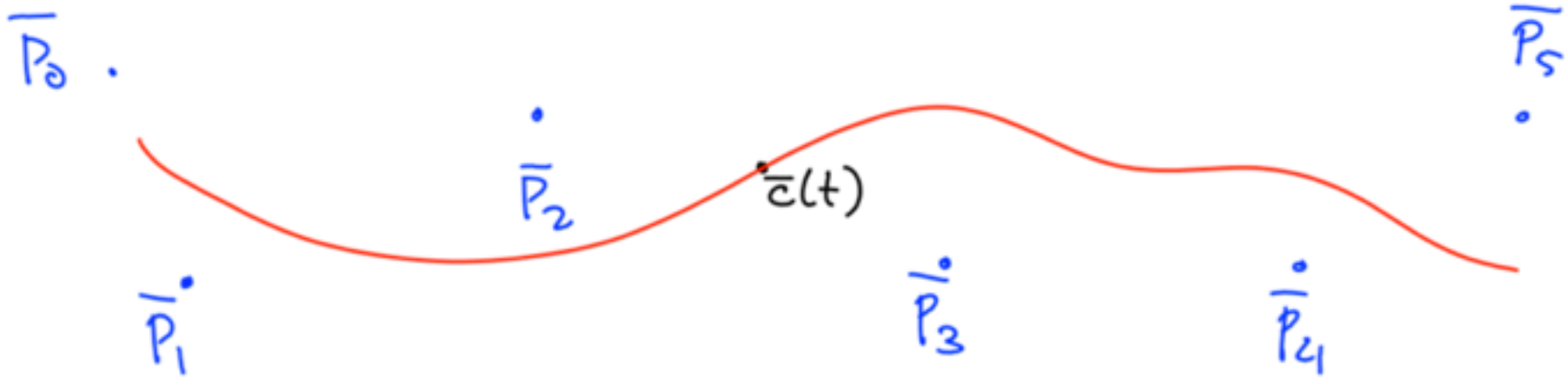


- Q: What is the disadvantage of such a technique?
- Curve has only C^0 continuity

Board Example!

2D Curve Design: General Problem Statement

- Given N control points, P_i, $i = 0 \ldots n-1$, $t \in [0, 1]$ (by convention)
- Define a curve c(t) that interpolates / approximates them
- Compute its derivatives (and tangents, normals etc)
- We will seek functions that are at least C^1

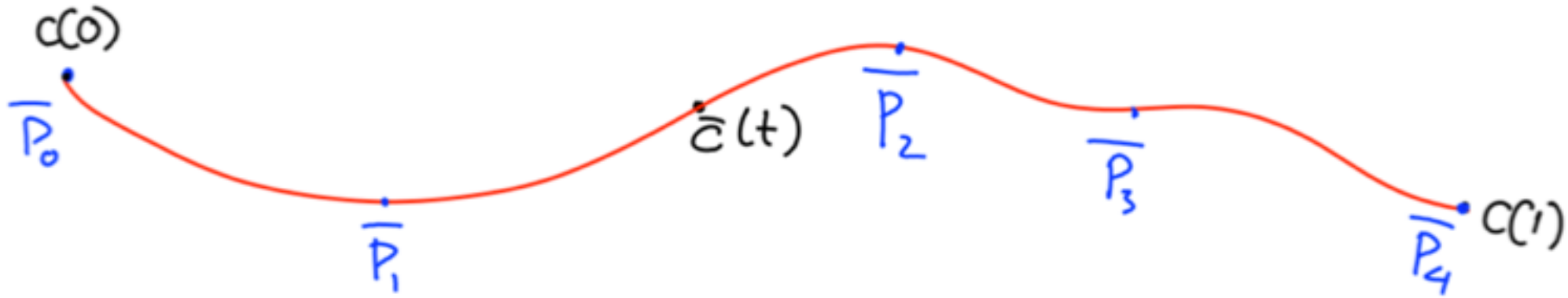


Lecture 8: Topic 2: Curves

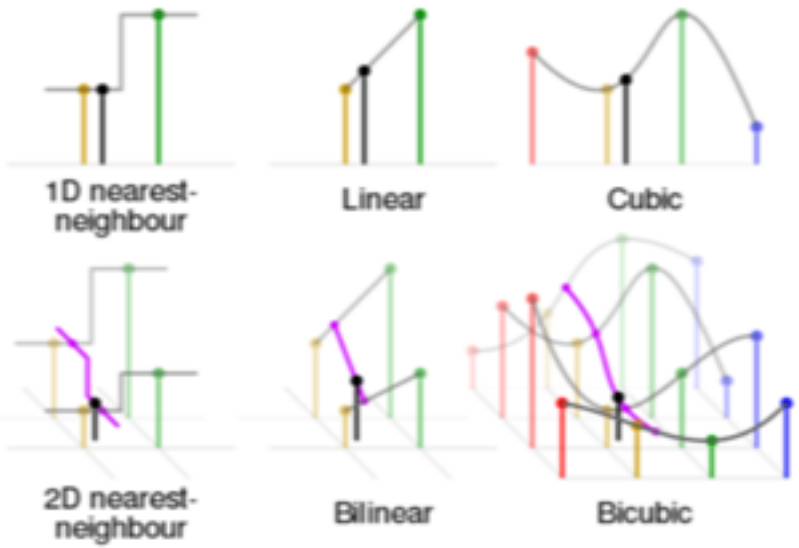
- Motivation
- Interpolating vs approximating
- Properties and continuity
- Polynomial interpolation
- Bezier curves
- Splines

Polynomial Interpolation

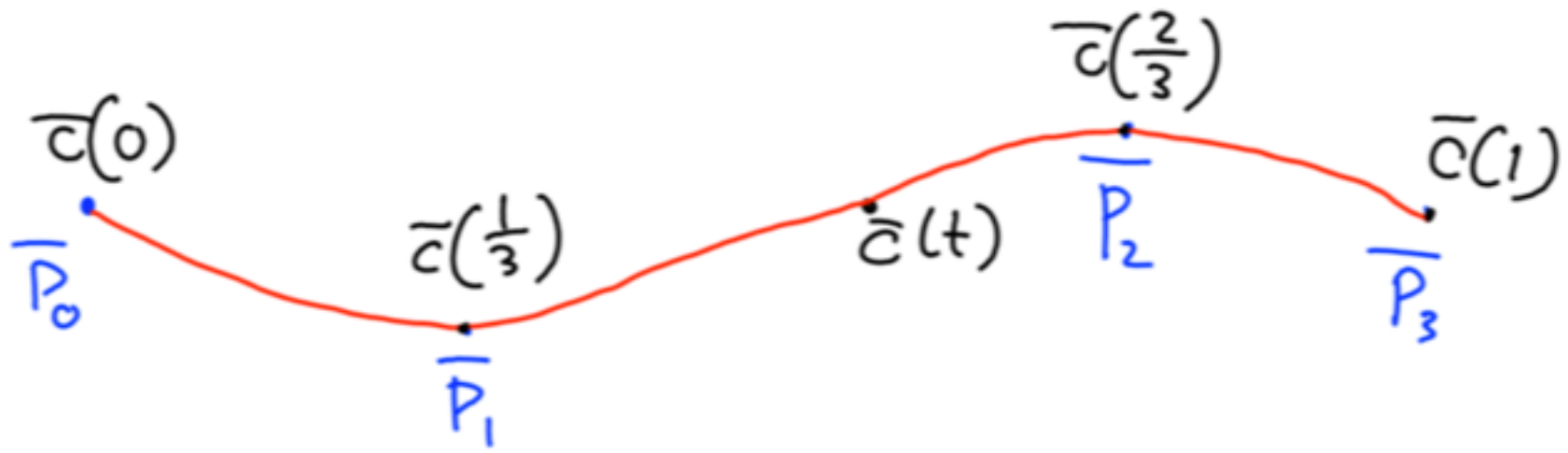
- Given N control points, P_i, $i = 0 \ldots n-1$, $t \in [0, 1]$ (by convention)
  - Define (N-1)-order polynomial x(t), y(t) such that $x(i/(N-1) = x_i$, $y(i/(N-1) = y_i$ for $i = 0, \ldots, N-1$
- Compute its derivatives (and tangents, normals etc)



Cubic Interpolation



- Given 4 control points, P_i, $i = (x_i, y_i)$, for $i = 0, \ldots, 3$
- Define 3rd-order polynomial $x(t)$, $y(t)$ such that $x(i/3) = x_i$, $y(i/3) = y_i$
- Compute its derivatives (and tangents, normals etc)



Cubic Interpolation: Basic Equations

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$y(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^2$$

given $\overline{P_1}, \overline{P_2}, \overline{P_3}, \overline{P_4}$

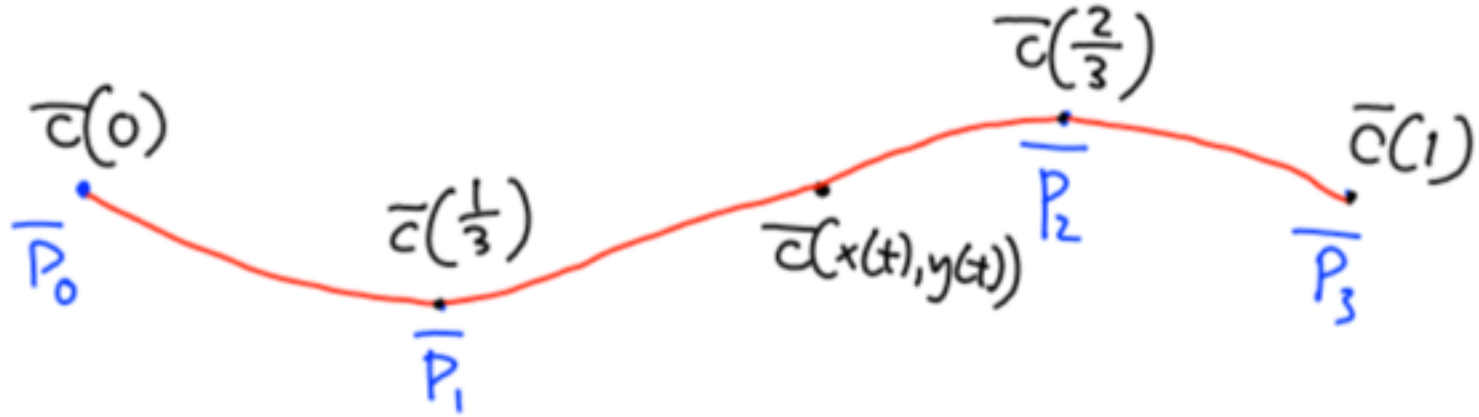
compute a_i, b_i

Equations for one control point:

$$x_1 = a_0 + a_1 \cdot \tfrac{1}{3} + a_2 (\tfrac{1}{3})^2 + a_3 (\tfrac{1}{3})^3$$
$$y_1 = b_0 + b_1 \cdot \tfrac{1}{3} + b_2 (\tfrac{1}{3})^2 + b_3 (\tfrac{1}{3})^3$$

Equations in matrix form:

$$\begin{bmatrix} x_1 & y_1 \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{3} & (\frac{1}{3})^2 & (\frac{1}{3})^3 \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$



Cubic Interpolation: Computing Coeffs

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$y(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^2$$

given $\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4$

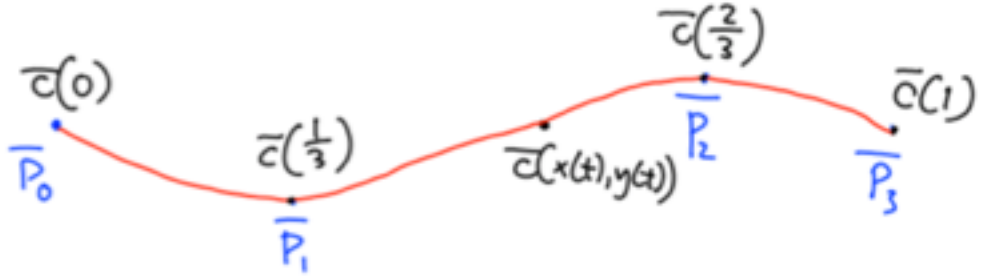
compute a_i, b_i

$$[x_i \; y_i] = \underbrace{[\,1 \quad t_i \quad (t_i)^2 \quad (t_i)^3\,]}_{\text{known } (t_i = i/N-1)} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix} \leftarrow \text{unknown}$$

$[x_i \; y_i]$: known

Equations in matrix form:

$$[x_1 \; y_1] = \left[1 \quad \tfrac{1}{3} \quad \left(\tfrac{1}{3}\right)^2 \quad \left(\tfrac{1}{3}\right)^3\right] \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$



Cubic Interpolation: Computing Coeffs

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$y(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^2$$

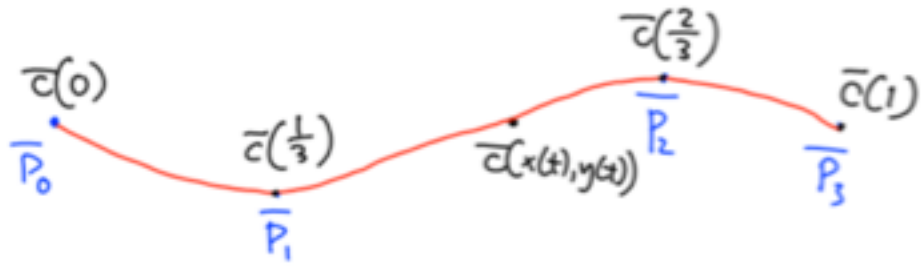
given $\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4$

compute a_i, b_i

$$\underbrace{\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}}_{C\ \text{known}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1/3 & (1/3)^2 & (1/3)^3 \\ 1 & 2/3 & (2/3)^2 & (2/3)^3 \\ 1 & 1 & 1 & 1 \end{bmatrix}}_{A\ \text{known}} \underbrace{\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ a_4 & b_4 \end{bmatrix}}_{X\ \text{unknown}}$$

$\Rightarrow$ solve system in terms of unknown matrix

$$X = A^{-1} C$$



Equations in matrix form:

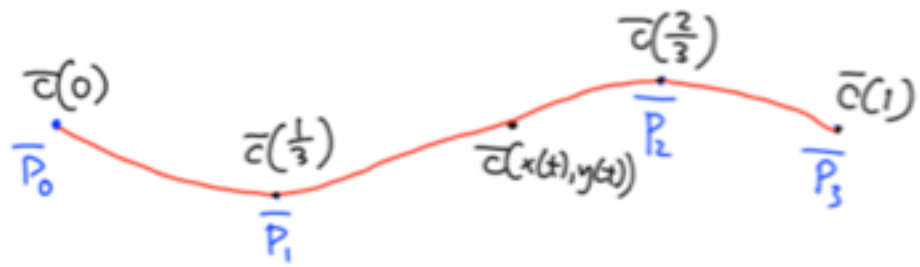
$$\begin{bmatrix} x_1 & y_1 \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{3} & \left(\frac{1}{3}\right)^2 & \left(\frac{1}{3}\right)^3 \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$

Cubic Interpolation: Computing Coeffs

$$\left.\begin{aligned} x(t) &= a_0 + a_1 t + a_2 t^2 + a_3 t^3 \\ y(t) &= b_0 + b_1 t + b_2 t^2 + b_3 t^2 \end{aligned}\right\} \quad \text{given } \overline{P_1}, \overline{P_2}, \overline{P_3}, \overline{P_4}$$
$$\text{compute } a_i, b_i$$

Coefficients of interpolating polynomial computed by:

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \\ a_4 & b_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1/3 & (1/3)^2 & (1/3)^3 \\ 1 & 2/3 & (2/3)^2 & (2/3)^3 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}$$



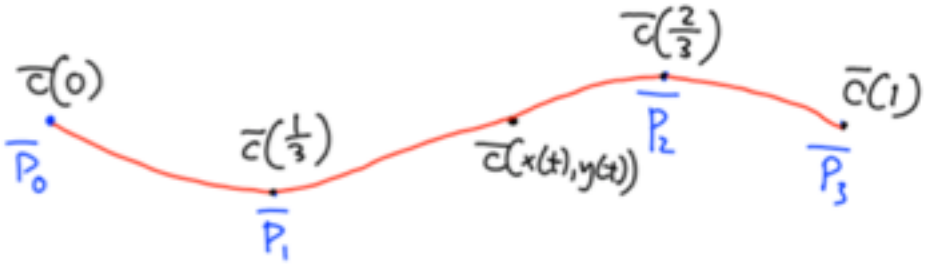
Equations in matrix form:

$$\begin{bmatrix} x_1 & y_1 \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{3} & \left(\frac{1}{3}\right)^2 & \left(\frac{1}{3}\right)^3 \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$

Cubic Interpolation: Evaluating the Polynomial

$$\left.\begin{aligned} x(t) &= a_0 + a_1 t + a_2 t^2 + a_3 t^3 \\ y(t) &= b_0 + b_1 t + b_2 t^2 + b_3 t^2 \end{aligned}\right\} \quad \begin{array}{l} \text{given } \bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4 \\ \text{compute } a_i, b_i \end{array}$$

$$\begin{bmatrix} x(t) & y(t) \end{bmatrix} = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$



Equations in matrix form:

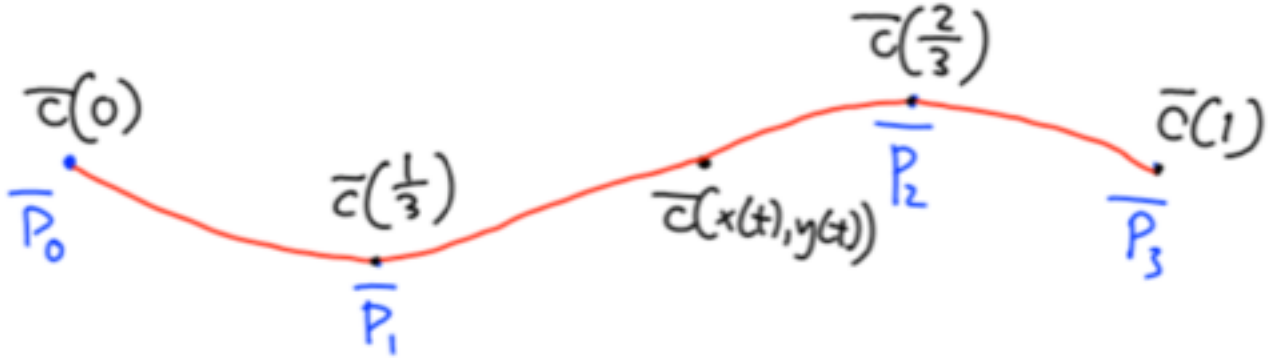
$$\begin{bmatrix} x_1 & y_1 \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{3} & \left(\frac{1}{3}\right)^2 & \left(\frac{1}{3}\right)^3 \end{bmatrix} \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$

Cubic Interpolation: Evaluating Derivatives

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

$$\frac{dx}{dt}(t) = a_1 + 2a_2 t + 3a_3 t^2$$

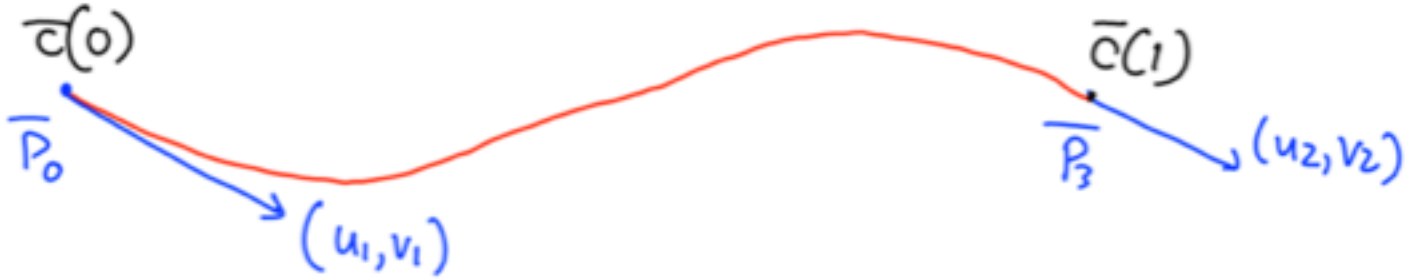
$$\left[\frac{dx}{dt}(t) \quad \frac{dy}{dt}(t)\right] = \begin{bmatrix} 1 & 2t & 3t^2 \end{bmatrix} \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$



Specifying the Poly via Tangent Constraints

- Instead of specifying 4 control points, we could specify 2 points and 2 derivatives.

$$\left[\frac{dx}{dt}(t) \quad \frac{dy}{dt}(t)\right] = \begin{bmatrix} 1 & 2t & 3t^2 \end{bmatrix} \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$

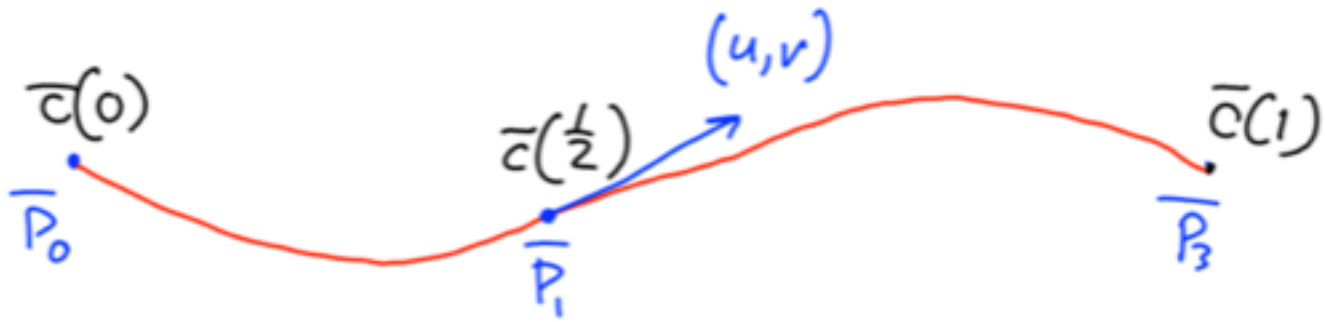


Specifying the Poly via Tangent Constraints

- Instead of specifying 4 control points, we could specify 3 points and a derivative.
- Replace the 4th pair of equations with

$$\begin{bmatrix} u & v \end{bmatrix} = \begin{bmatrix} 1 & 1 & 3\left(\frac{1}{2}\right)^2 \end{bmatrix} \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$

$$\begin{bmatrix} \frac{dx}{dt}(t) & \frac{dy}{dt}(t) \end{bmatrix} = \begin{bmatrix} 1 & 2t & 3t^2 \end{bmatrix} \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$

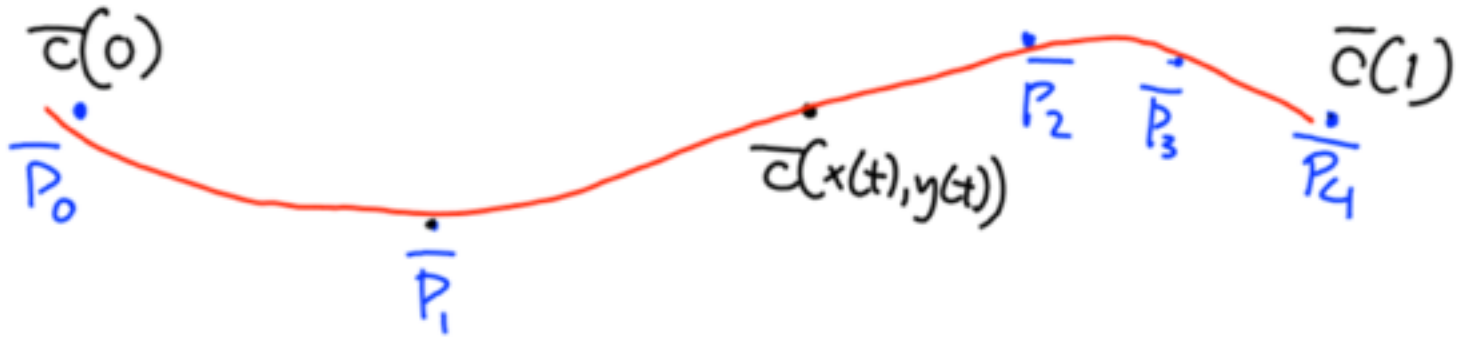


Degree-N Poly Interpolation: Major Drawback

To interpolate N points perfectly with a single polynomial, we need a polynomial of degree N-1

Major drawback: it is a global interpolation scheme

i.e. moving one control point changes the interpolation of all points, often in unexpected, unintuitive and undesirable ways

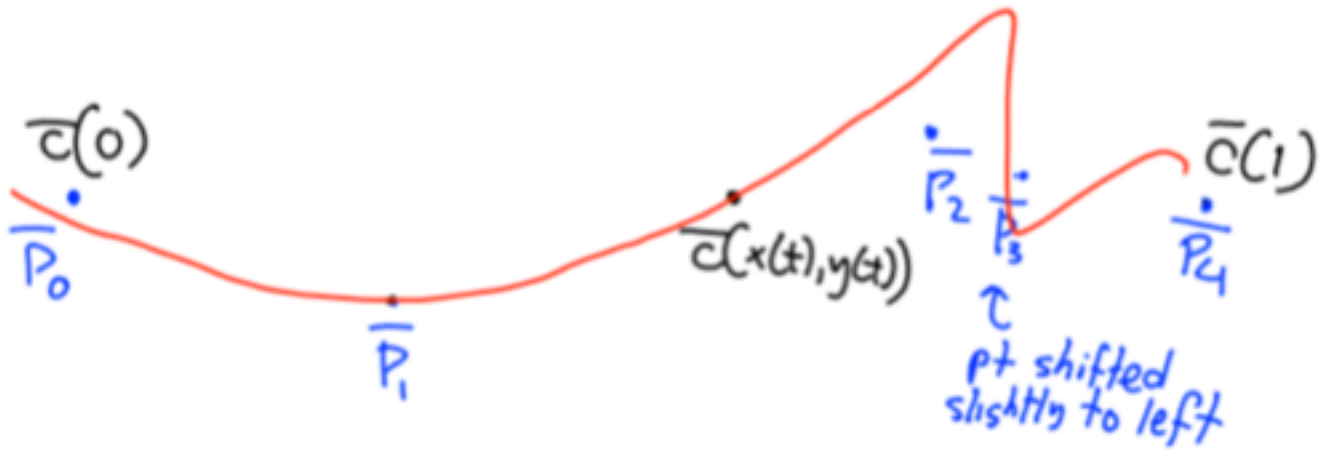


Degree-N Poly Interpolation: Major Drawback

To interpolate N points perfectly with a single polynomial, we need a polynomial of degree N-1

Major drawback: it is a global interpolation scheme

i.e. moving one control point changes the interpolation of all points, often in unexpected, unintuitive and undesirable ways



Lecture 8: Topic 2: Curves

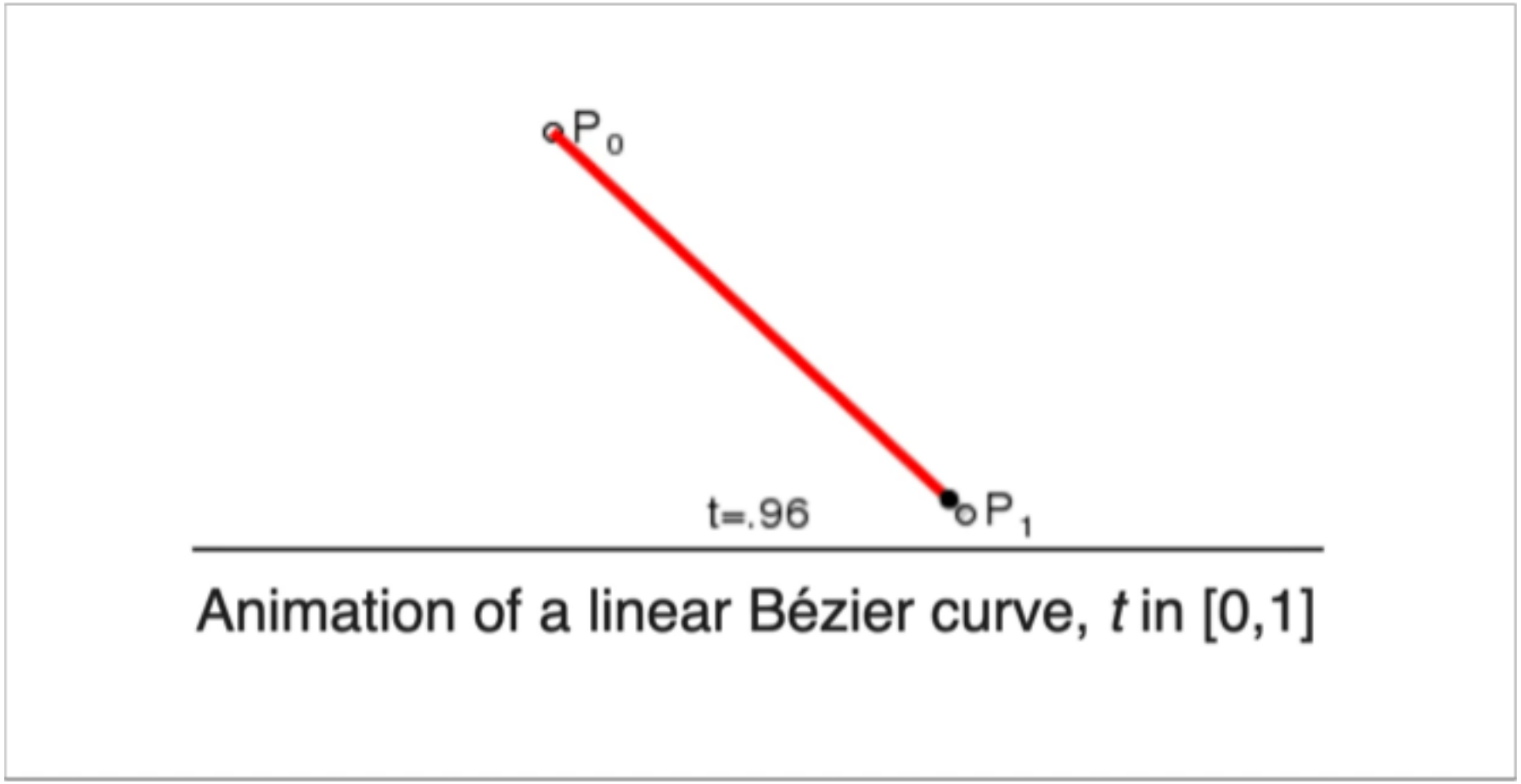
- Motivation
- Interpolating vs approximating
- Properties and continuity
- Polynomial interpolation
- Bezier curves
- Splines

Bézier Curves

Properties:

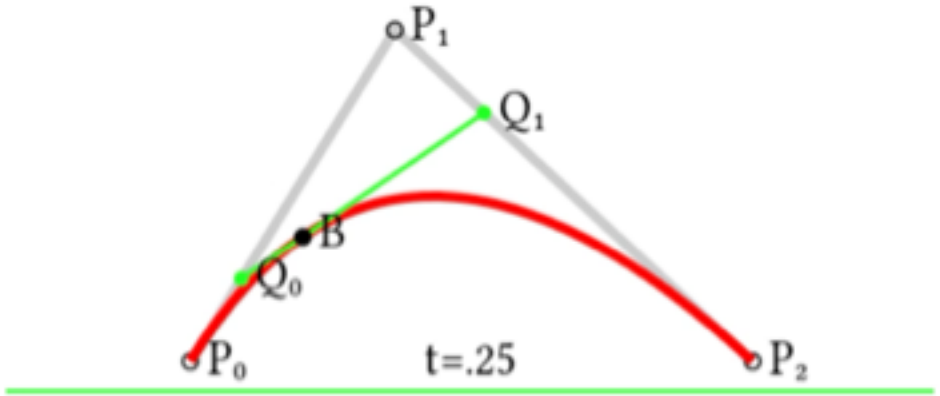
- Polynomial curves defined via endpoints and derivative constraints
- Derivative constraints defined implicitly through extra control points (that are not interpolated)
- They are approximating curves, not interpolating curves

Linear Bezier

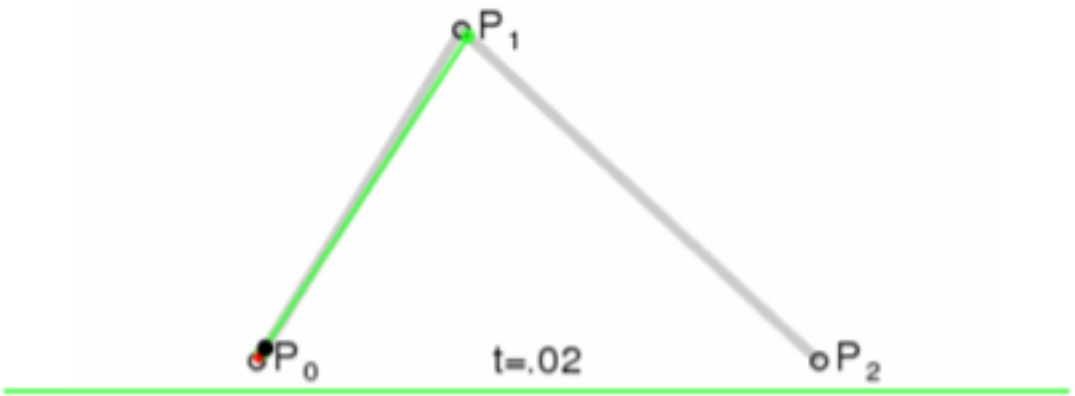


From Wikipedia

Quadratic Bezier (note it's approximating)



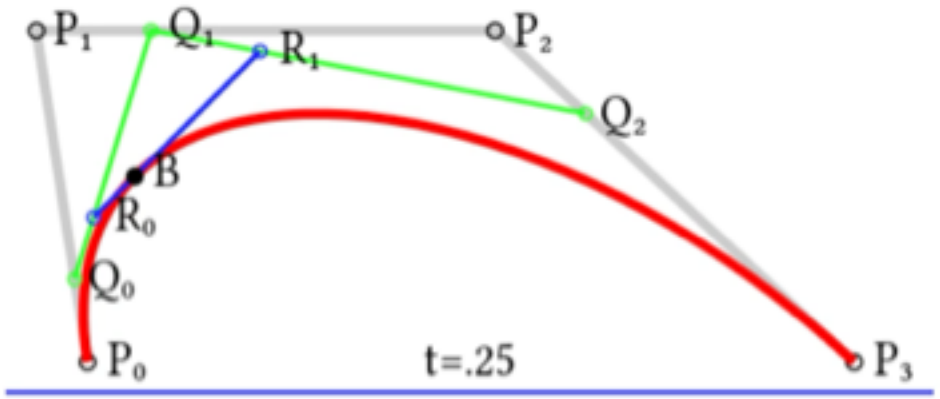
Construction of a quadratic Bézier curve



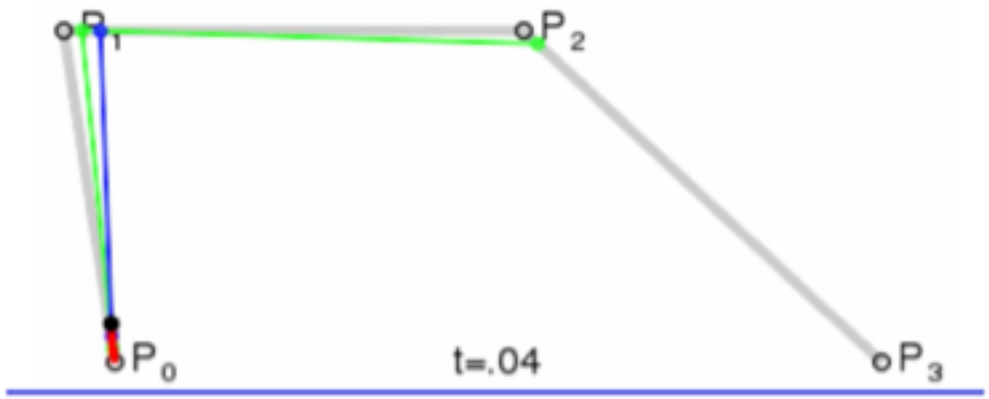
Animation of a quadratic Bézier curve, *t* in [0,1]

From Wikipedia

Cubic Bezier



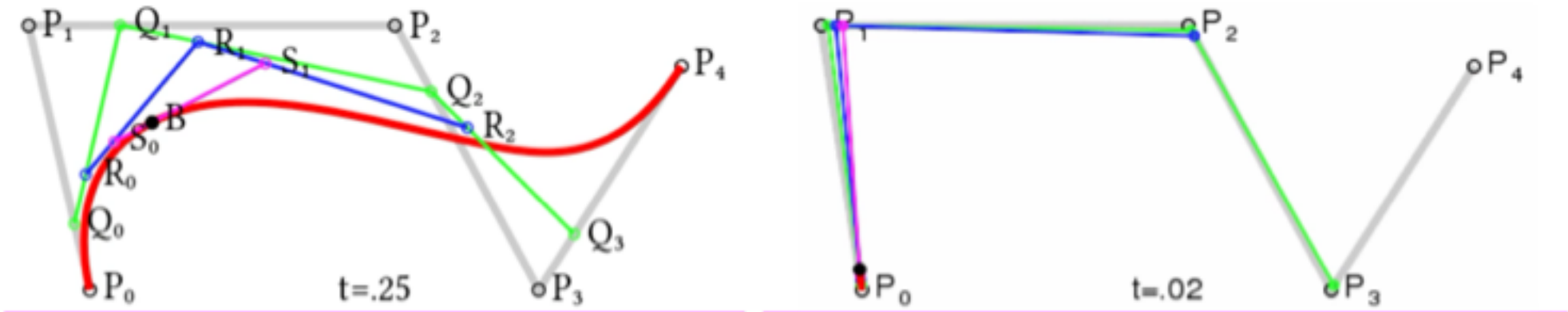
Construction of a cubic Bézier curve



Animation of a cubic Bézier curve, *t* in [0,1]

From Wikipedia

Higher-Order



Construction of a quartic Bézier curve

Animation of a quartic Bézier curve, *t* in [0,1]

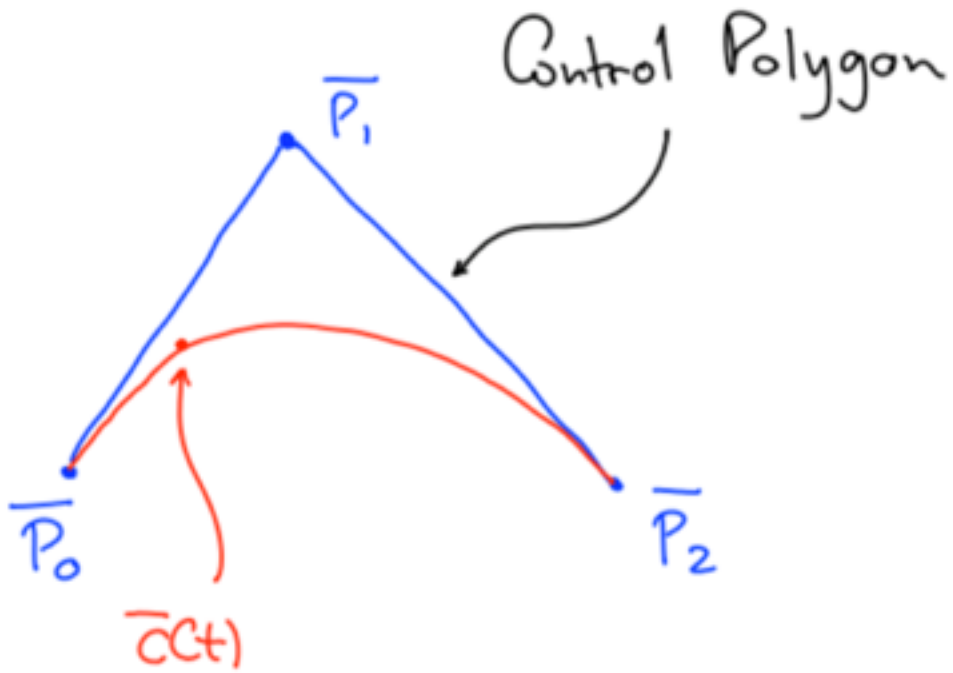
From Wikipedia

Bézier Curves: Control Polygon

A Bézier curve is completely determined by its control polygon

We manipulate the curve by manipulating its polygon

Example: a double cascade



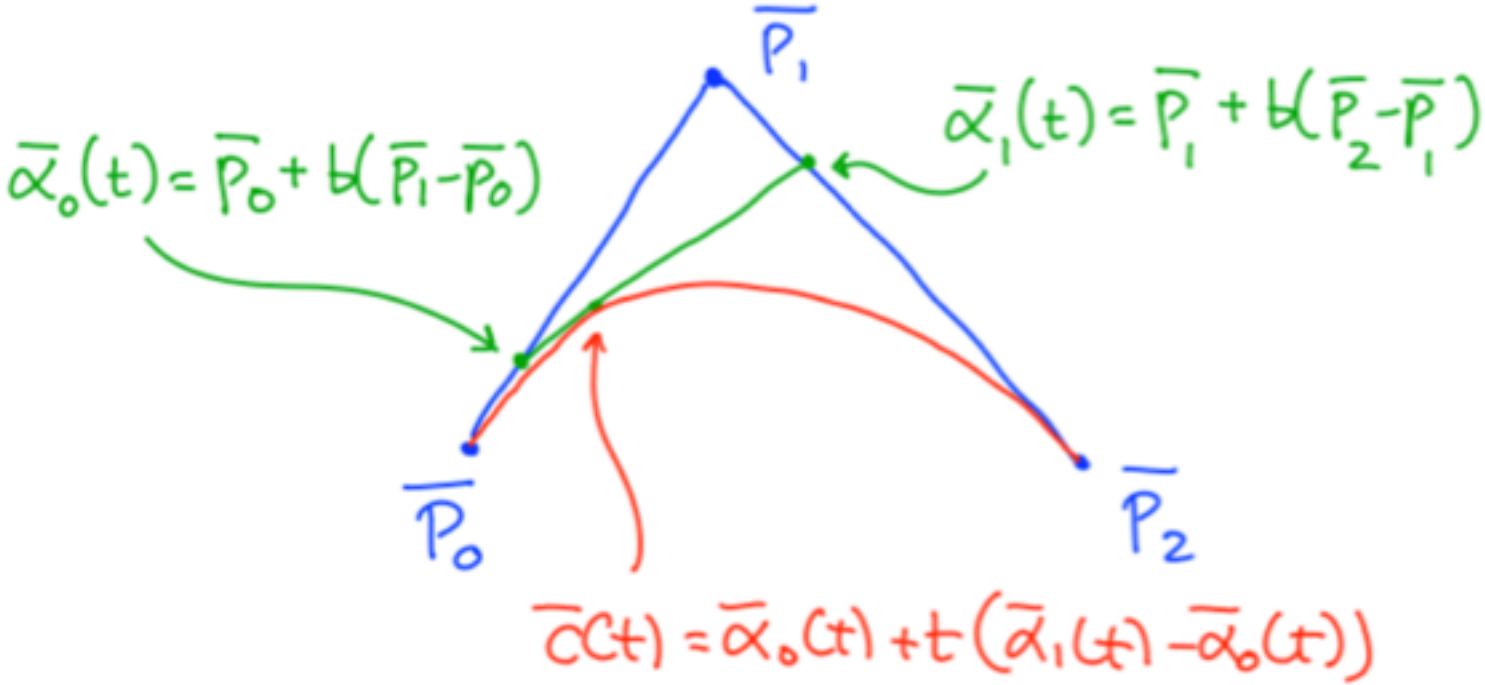
algorithm:
given $\overline{P}_0, \overline{P}_1, \overline{P}_2$ and t

1. linearly interpolate $\overline{P}_0, \overline{P}_1$ to get $\overline{\alpha}_0(t)$
2. linearly interpolate $\overline{P}_1, \overline{P}_2$ to get $\overline{\alpha}_1(t)$
3. linearly interpolate $\overline{\alpha}_0(t), \overline{\alpha}_1(t)$ to get $\overline{c}(t)$

Expressing the Bézier Curve as a Polynomial

Computing the polynomial

$$\begin{aligned} c(t) &= [P_0 + t(P_1 - P_0)] + t[P_1 + t(P_2 - P_1) - P_0 - t(P_1 - P_0)] \\ &= P_0(1 - t - t + t^2) + P_1(t + t - t^2 - t^2) + P_2 t^2 \\ &= P_0(1-t)^2 + 2P_1 t(1-t) + P_2 t^2 \end{aligned}$$



algorithm:

given $\bar{P}_0, \bar{P}_1, \bar{P}_2$ and t

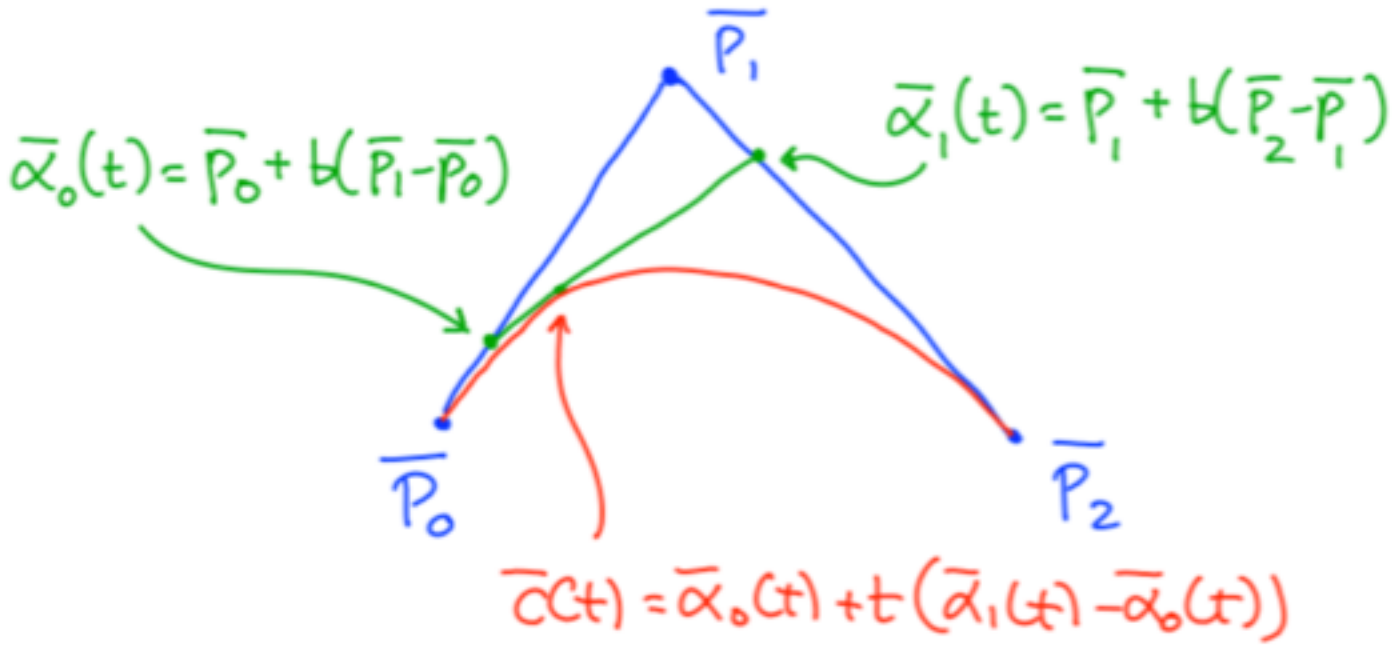
1. linearly interpolate $\bar{P}_0, \bar{P}_1$ to get $\bar{\alpha}_0(t)$
2. linearly interpolate $\bar{P}_1, \bar{P}_2$ to get $\bar{\alpha}_1(t)$
3. linearly interpolate $\bar{\alpha}_0(t), \bar{\alpha}_1(t)$ to get $\bar{c}(t)$

Derivatives of the Bézier Curve

Computing the polynomial's derivatives:

$$\frac{d}{dt}c(t) = -2(1-t)P_0 + 2P_1(1-t) + P_2 2t \quad \begin{array}{l} = 2(P_1-P_0) \text{ at } t=0 \\ = 2(P_2-P_1) \text{ at } t=1 \end{array}$$

$$c(t) = \overline{P}_0(1-t)^2 + 2\overline{P}_1 t(1-t) + \overline{P}_2 t^2$$



algorithm:

given $\overline{P}_0, \overline{P}_1, \overline{P}_2$ and t

1. linearly interpolate $\overline{P}_0, \overline{P}_1$ to get $\bar{\alpha}_0(t)$
2. linearly interpolate $\overline{P}_1, \overline{P}_2$ to get $\bar{\alpha}_1(t)$
3. linearly interpolate $\bar{\alpha}_0(t), \bar{\alpha}_1(t)$ to get $\overline{C}(t)$

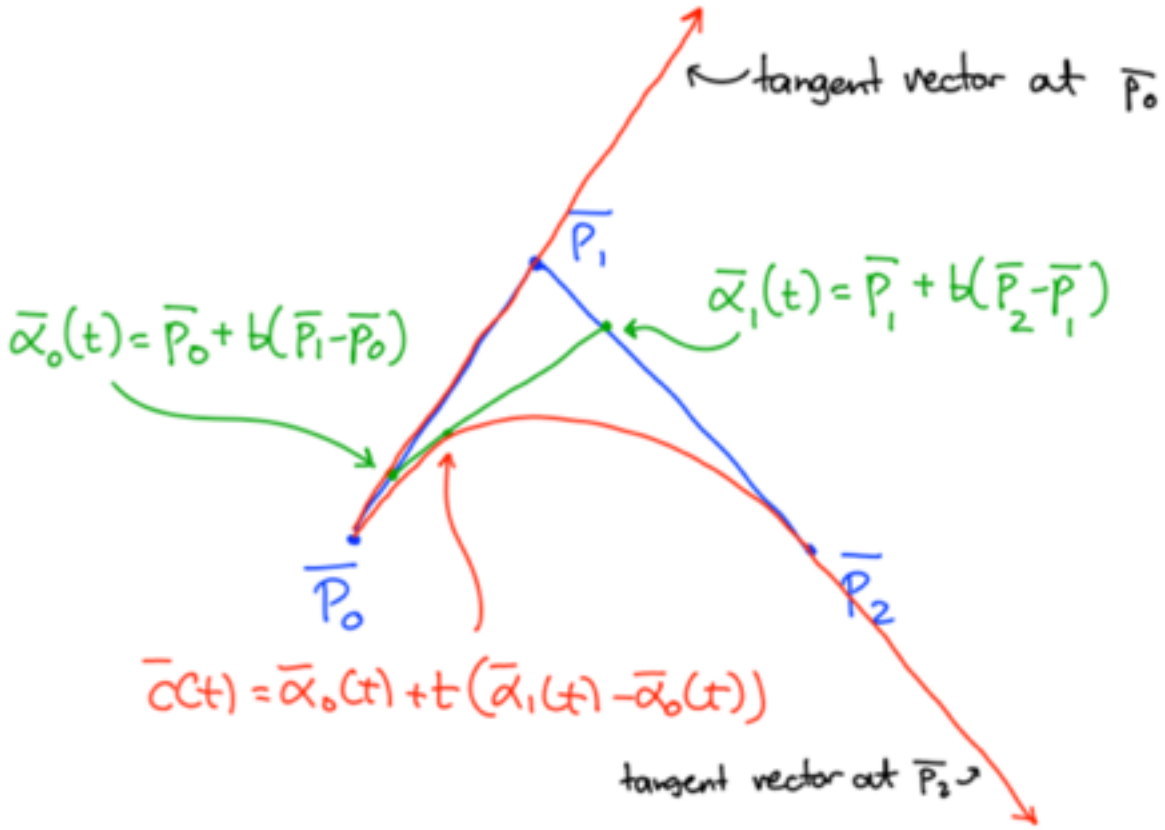
Bézier Curves: Endpoints and Tangent Constraints

Computing the polynomial's derivatives:

$$\frac{d}{dt}C(t) = -2(1-t)P_0 + 2P_1(1-2t) + P_2 2t$$

$$= 2(P_1 - P_0) \text{ at } t=0$$

$$= 2(P_2 - P_1) \text{ at } t=1$$



General Behaviour

- 1st and 3rd control points define the endpoints.
- 2nd control point defines the tangent vector at the endpoints.

Bézier Curves: Generalization to N+1 points

Expression in compact form:

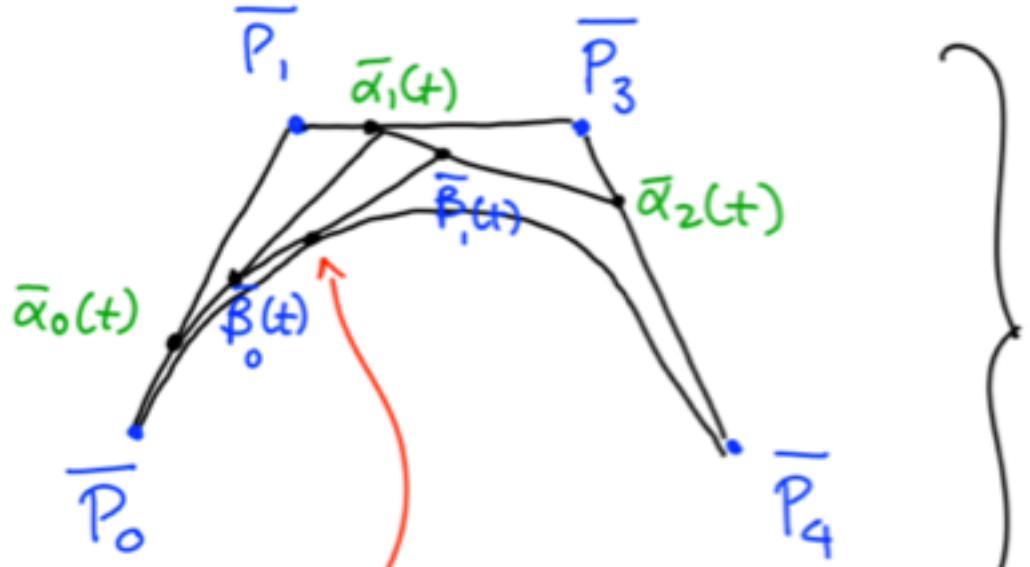
$$\bar{C}(t) = \sum_{i=0}^{N} \bar{P}_i B_i^N(t)$$ Where:

$\bar{C}(t)$: curve; $\bar{P}_i$: control pt

called the Bernstein Polynomials of degree N

$$B_i^N(t) = \binom{N}{i}(1-t)^{N-i}t^i = \frac{N!}{(N-i)!\,i!}(1-t)^{N-i}t^i$$

Curve defined by N linear interpolation cascades (De Casteljau's algorithm):



$$\bar{C}(t) = \bar{\beta}_0(t) + t\left(\bar{\beta}_1(t) - \bar{\beta}_0(t)\right)$$

Example for 4 control points and 3 cascades

Bézier Curves: A Different Perspective

Expression in compact form:

$$\bar{c}(t) = \sum_{i=0}^{N} \bar{P}_i B_i^N(t)$$

Where:

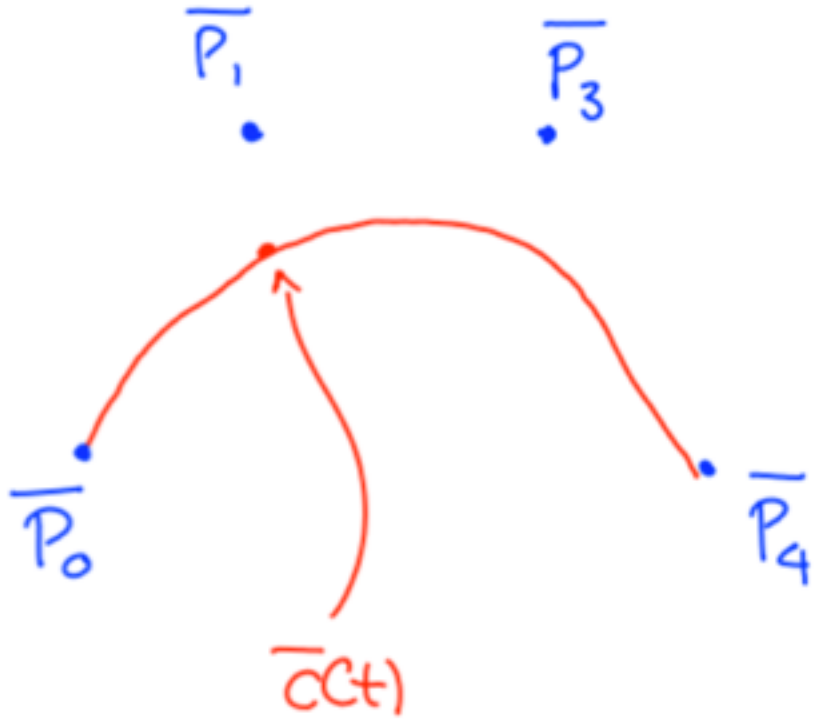
curve

control pt

called the Bernstein Polynomials of degree N

$$B_i^N(t) = \binom{N}{i}(1-t)^{N-i}t^i$$

$$= \frac{N!}{(N-i)!\,i!}(1-t)^{N-i}t^i$$



- Each curve point c(t) is a "blend" of the 4 control points.
- The blend coefficients depend on t
- They are Bernstein polynomials

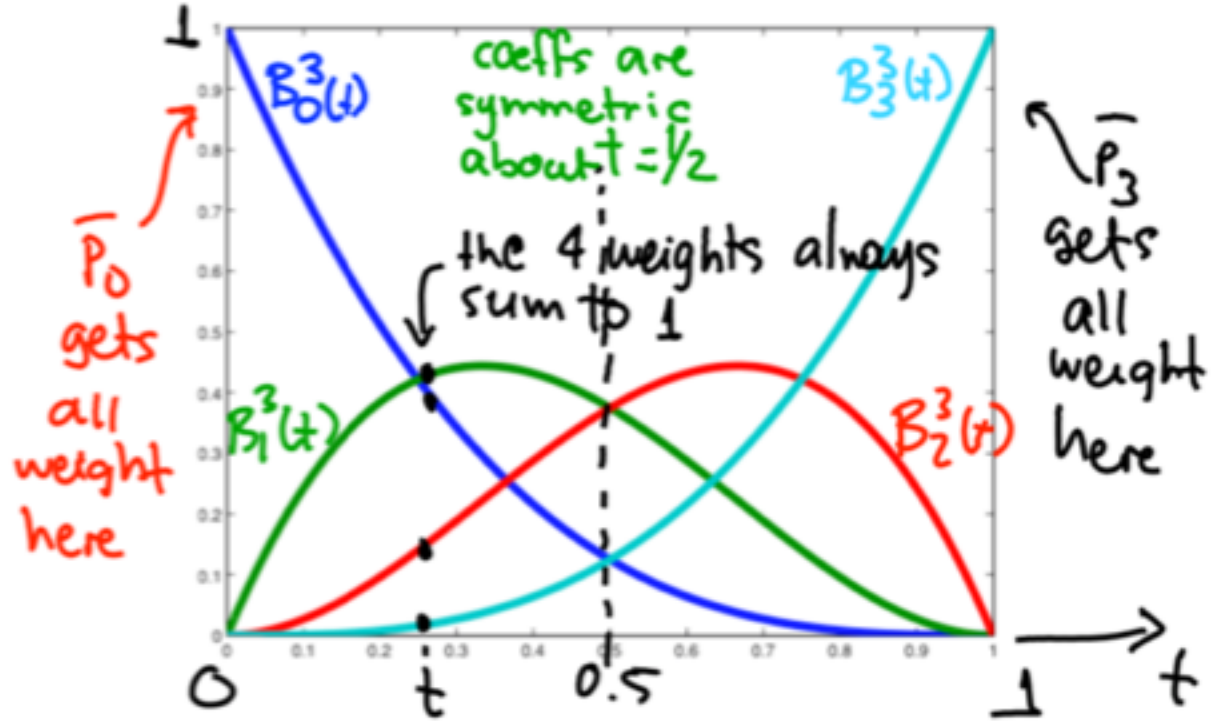
Bézier Curves as "blends" of the Control Points

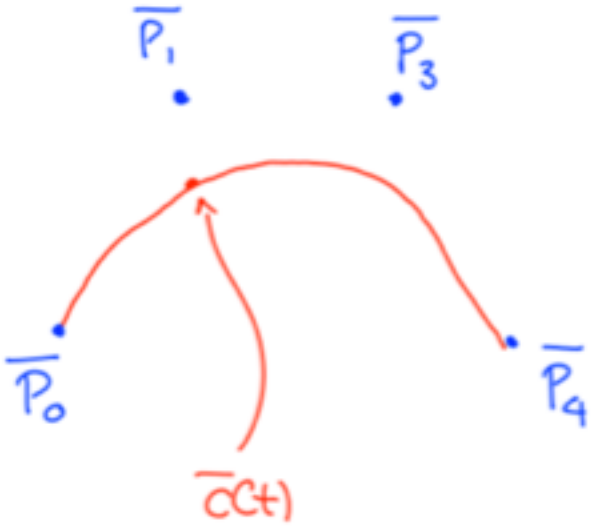
Expression in compact form:

$$\bar{c}(t) = \sum_{i=0}^{N} \bar{P}_i B_i^N(t)$$

curve ↑ $\bar{c}(t)$; control pt ↑ $\bar{P}_i$

with $\sum_{i=0}^{N} B_i^N(t) = 1$ for all t





- Each curve point c(t) is a "blend" of the 4 control points.
- The blend coefficients depend on t
- They are Bernstein polynomials

Bézier Curves: Useful Properties

Expression in compact form:

$$\bar{C}(t) = \sum_{i=0}^{N} \bar{P}_i B_i^N(t)$$

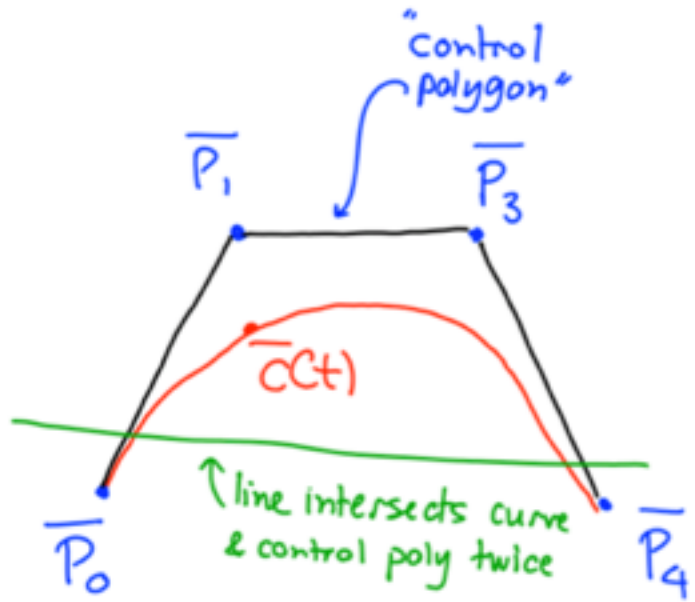
Where:

called the Bernstein Polynomials of degree N

$$B_i^N(t) = \binom{N}{i}(1-t)^{N-i}t^i$$

$$= \frac{N!}{(N-i)!\,i!}(1-t)^{N-i}t^i$$

1. Affine Invariance
   - Transforming a Bézier curve by an affine transform T is equivalent to transforming its control points by T
2. Diminishing Variation
   - No line will intersect the curve at more points than the control polygon
     - curve cannot exhibit "excessive fluctuations"
3. Linear Precision
   - If control poly approximates a line, so will the curve



Bézier Curves: Useful Properties

Expression in compact form:

$$\bar{c}(t) = \sum_{i=0}^{N} \bar{P}_i B_i^N(t)$$

Where:

called the Bernstein Polynomials of degree N

$$B_i^N(t) = \binom{N}{i}(1-t)^{N-i}t^i$$

$$= \frac{N!}{(N-i)!\,i!}(1-t)^{N-i}t^i$$

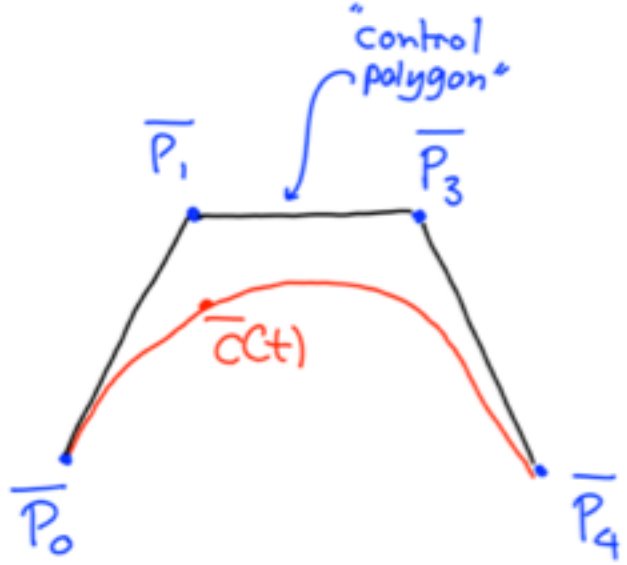
4. Tangents at endpoints are along the 1st and last edges of control polygon:

$$\frac{d}{dt}\bar{c}(t) = \sum_{i=1}^{N} \bar{P}_i \frac{d}{dt} B_i^N(t)$$

w/ some work $= N \sum_{i=0}^{N-1} (\bar{P}_{i+1} - \bar{P}_i) B_i^{N-1}(t)$

$= N(\bar{P}_1 - \bar{P}_0)$ for $t=0$

$= N(\bar{P}_N - \bar{P}_{N-1})$ for $t=1$



Bézier Curves: Pros and Cons

Expression in compact form:

$$\bar{c}(t) = \sum_{i=0}^{N} \bar{P}_i B_i^N(t)$$

Where:

called the Bernstein Polynomials of degree N

$$B_i^N(t) = \binom{N}{i}(1-t)^{N-i}t^i$$

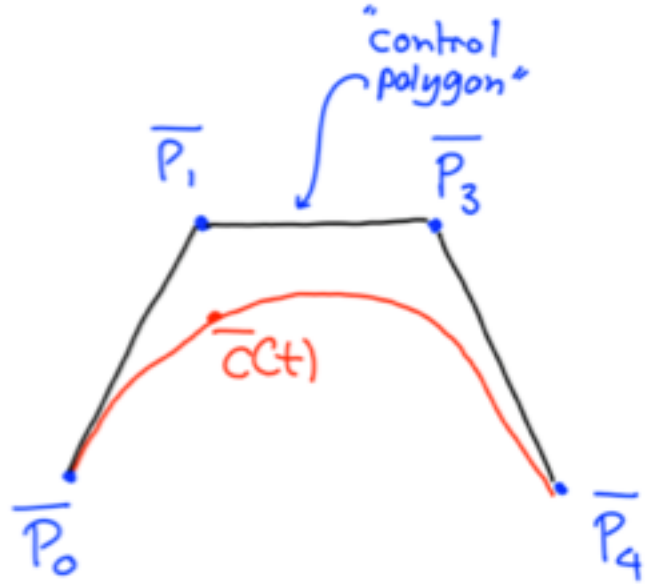
$$= \frac{N!}{(N-i)!\,i!}(1-t)^{N-i}t^i$$

Advantages:

- Intuitive control for $N \leq 3$
- Derivatives easy to compute
- Nice properties (affine invariance, diminishing variation)

Disadvantages:

- Scheme is still global (curve is function of all control points)

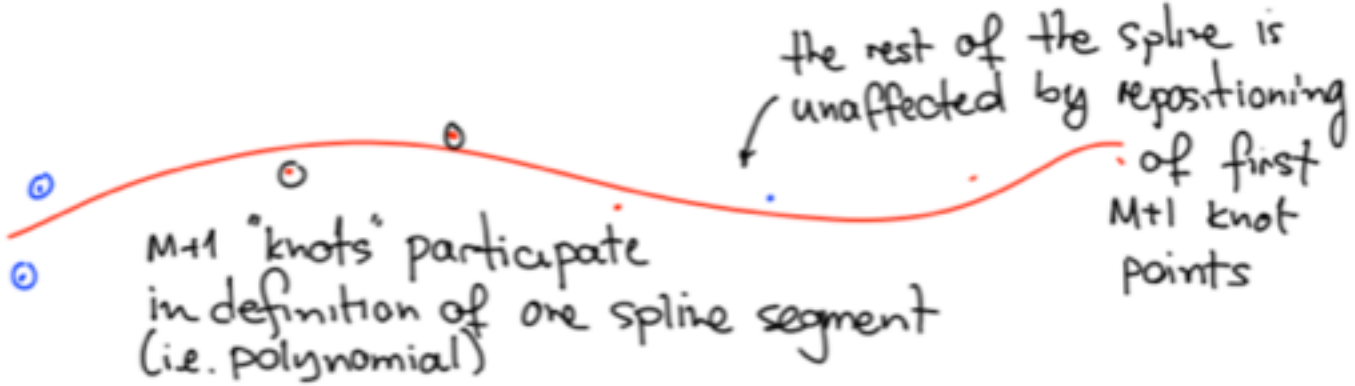


Lecture 8: Topic 2: Curves

- Motivation
- Interpolating vs approximating
- Properties and continuity
- Polynomial interpolation
- Bezier curves
- Splines

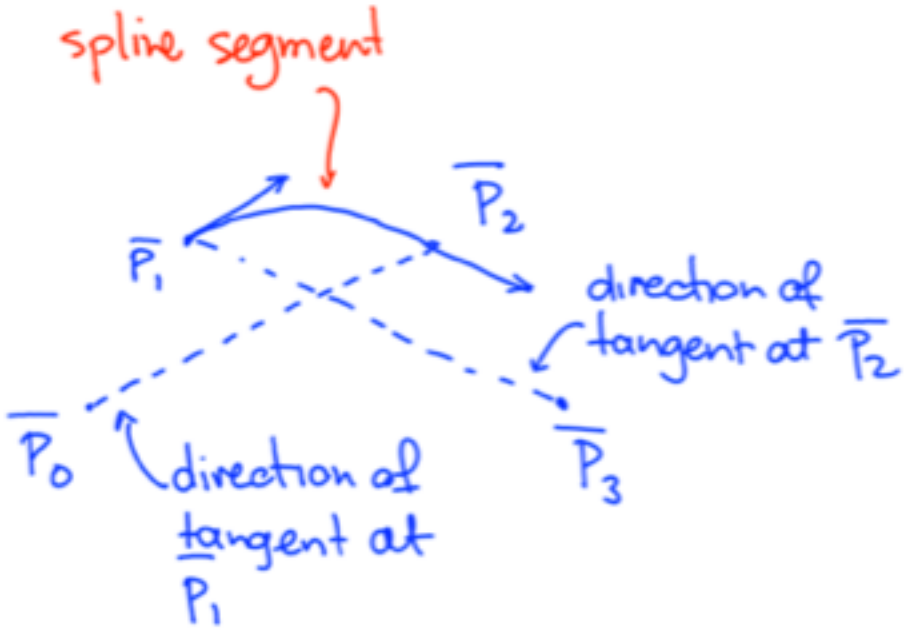
Splines: Local Control + Reduced Continuity

- Idea: provide local control of curve approximating N points by sacrificing C^N
- M-degree spline: a piecewise-polynomial curve of degree M
- Splines often defined to have C^{M-2} continuity at the "knot" points (aka control points)



Cubic Cardinal Splines: Defining 1st Segment

- Approach:
  1. A user only specifies points p_0, p_1, …
  2. Tangent at p_i set to be parallel to vector connecting p_{i-1} and p_{i+1}



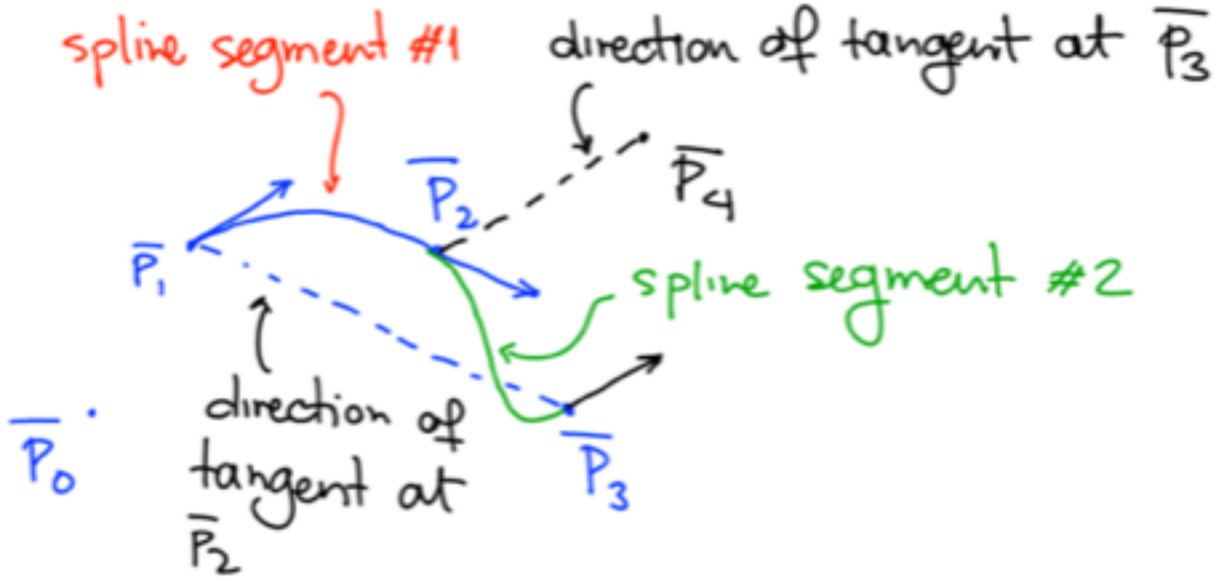
$\bar{P}_1, \bar{P}_2$ act as endpoint constraints

$\bar{P}_0, \bar{P}_3$ define derivative constraints

Cubic Cardinal Splines: Defining 2nd Segment

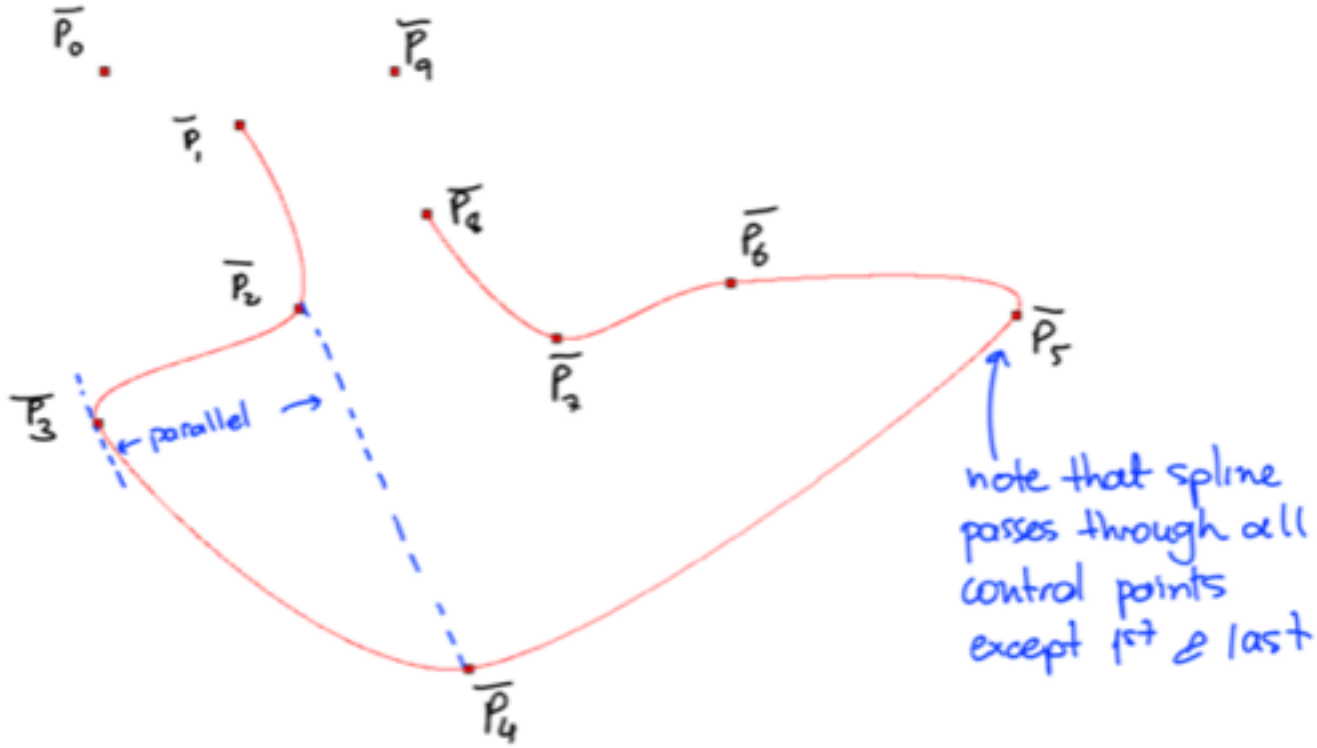
- Approach:
  1. A user only specifies points p_0, p_1, …
  2. Tangent at p_i set to be parallel to vector connecting p_{i-1} and p_{i+1}

Example: Adding a fifth point adds a new segment



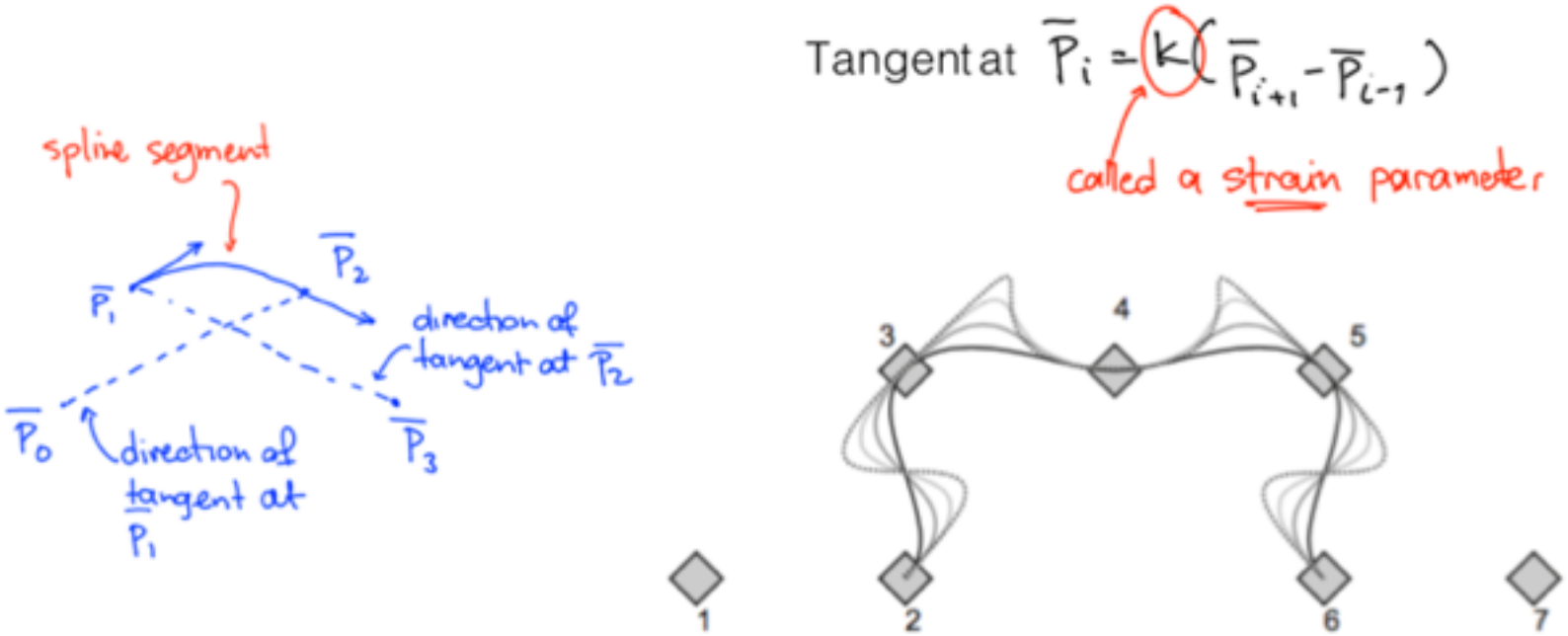
Cubic Cardinal Splines: General Case

- Approach:
  1. A user only specifies points p_0, p_1, …
  2. Tangent at p_i set to be parallel to vector connecting p_{i-1} and p_{i+1}



Cubic Cardinal Splines: The Strain Parameter

- Approach:
  1. A user only specifies points p_0, p_1, …
  2. Tangent at p_i set to be parallel to vector connecting p_{i-1} and p_{i+1}

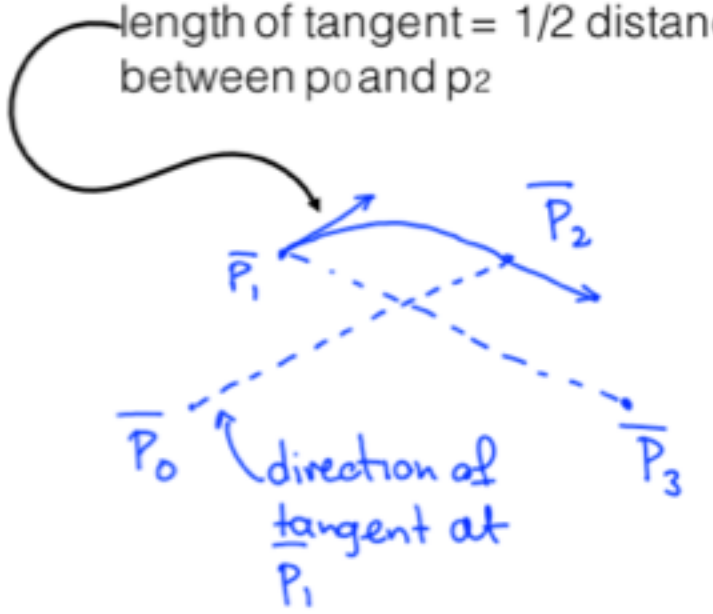


Catmull-Rom Splines

- Approach:
  1. A user only specifies points p_0, p_1, …
  2. Tangent at p_i set to be parallel to vector connecting p_{i-1} and p_{i+1}

Tangent at $\bar{P}_i = k(\bar{P}_{i+1} - \bar{P}_{i-1})$

called a strain parameter

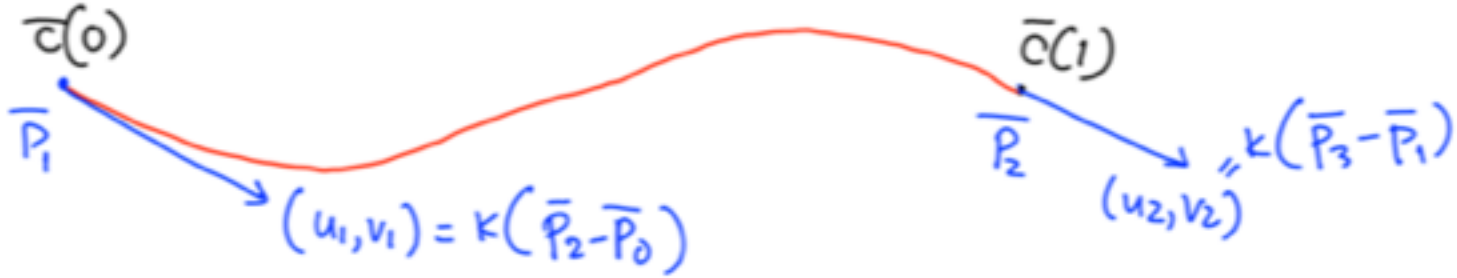


Note: If k = 0.5, the spline is called a Catmull-Rom Spline

Specifying the Poly via Tangent Constraints

- Instead of specifying 4 control points, we could specify 2 points and 2 derivatives

$$\left[\frac{dx}{dt}(t) \quad \frac{dy}{dt}(t)\right] = \begin{bmatrix} 1 & 2t & 3t^2 \end{bmatrix} \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$



Cardinal Splines: Solving for the Segment Coeffs

for t = 0

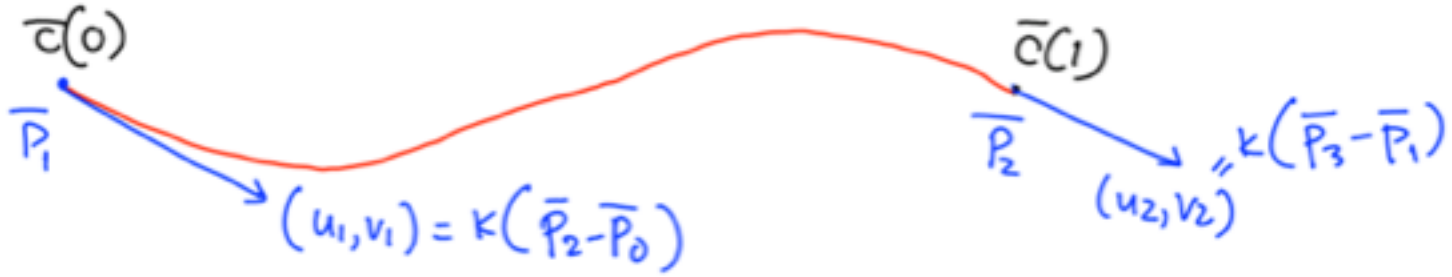
$$[x_1 \; y_1] = [1 \; \cancel{t_0} \; \cancel{t_0^2} \; \cancel{t_0^3}] \begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$

endpoint constraint

$$[k(x_2 - x_0) \; k(y_2 - y_0)] = [1 \; \cancel{2t_0} \; \cancel{3t_0^2}] \begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}$$

derivative constraint

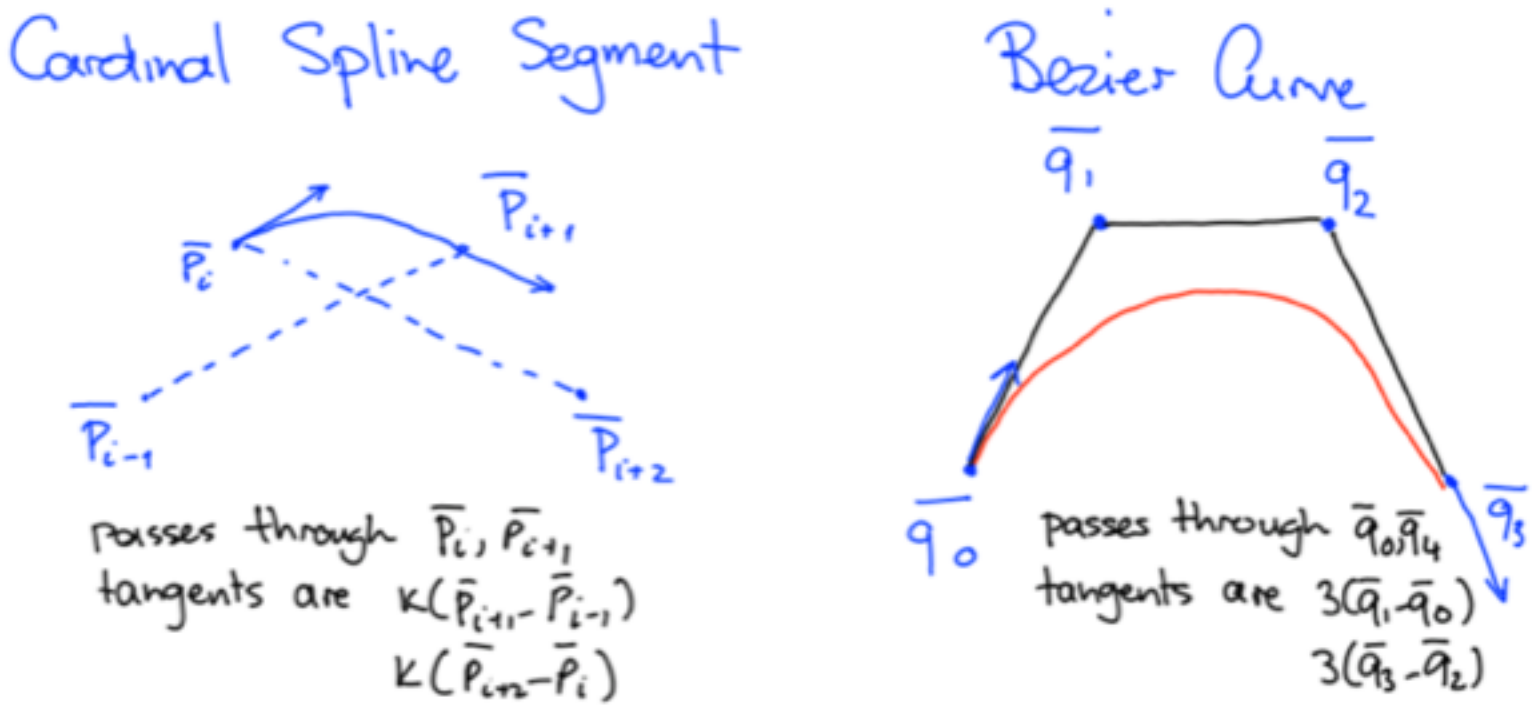
+ 2 more equations for other endpoint (t = 1)



Cubic Cardinal Spline Segment vs Bézier Curve

The two curves are actually equivalent:

given a cardinal spline, we can compute the control polygon of the equivalent Bézier curve



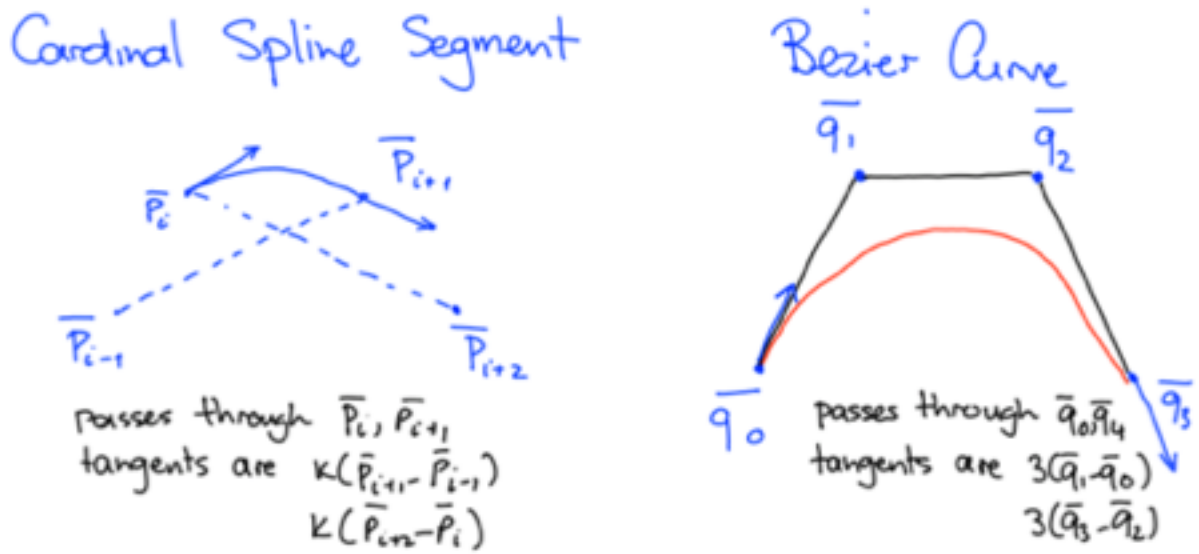
Cubic Cardinal Spline Segment vs Bézier Curve

In order to have c(t) = r(t) for all t, it must be:

$$\bar{q}_0 = \bar{P}_i \quad , \quad \bar{q}_4 = \bar{P}_{i+1}$$

$$k(\bar{P}_{i+1} - \bar{P}_{i-1}) = 3(\bar{q}_1 - \bar{q}_0) \Rightarrow k(\bar{P}_{i+1} - \bar{P}_{i-1}) = 3\bar{q}_1 - 3\bar{q}_0$$

$$\Rightarrow \bar{q}_1 = \frac{k}{3}(\bar{P}_{i+1} - \bar{P}_{i-1}) + \bar{P}_i$$



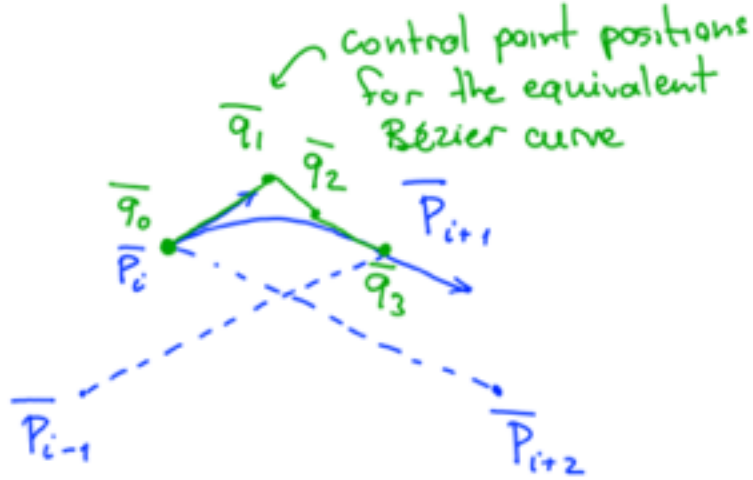
Cubic Cardinal Spline Segment vs Bézier Curve

In order to have c(t) = r(t) for all t, it must be:

$$\bar{q}_0 = \bar{P}_i \quad , \quad \bar{q}_4 = \bar{P}_{i+1}$$

$$\bar{q}_1 = \frac{k}{3}(\bar{P}_{i+1} - \bar{P}_{i-1}) + \bar{P}_i \quad , \quad \bar{q}_2 = \bar{P}_{i+1} - \frac{k}{3}(\bar{P}_{i+2} - \bar{P}_i)$$

Cardinal Spline Segment



Bezier Curve