Computer Animation

Sylvia Pan

March 2013

Lecture slides heavily based on previous versions produced by Marco Gillies

Course Outline

- Physical systems
  - Physics simulation
  - Integration techniques
  - Particle systems
- Traditional animation
- Key frame and interpolation
- Character animation
  - Body and face
  - Behaviour simulation

Computer Animation: Categorises

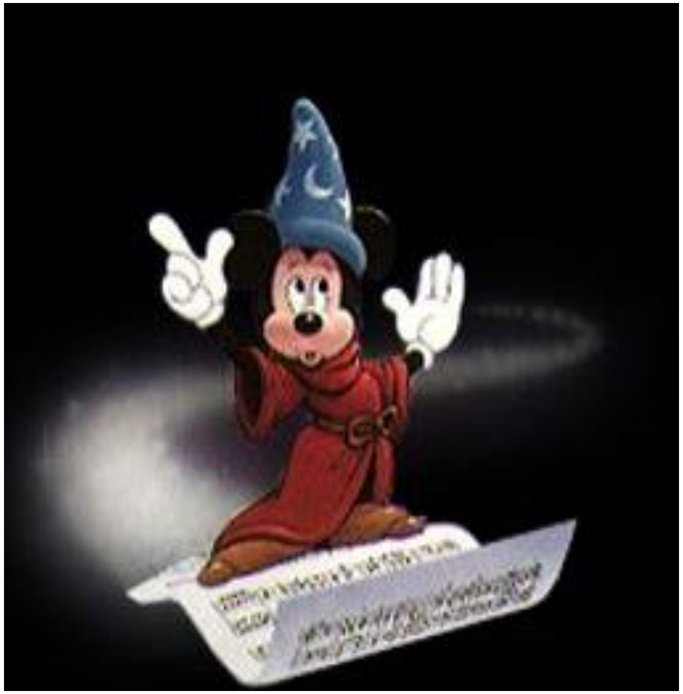
Three approaches to motion control:

- Artistic animation
  - Hand Animation (traditional animation)
  - Key frame and interpolation
- Data-driven animation
  - Motion capture
- Procedure animation
  - Simulations, artificial lives
  - AI

Traditional Animation:
Overview and some techniques

Traditional Animation

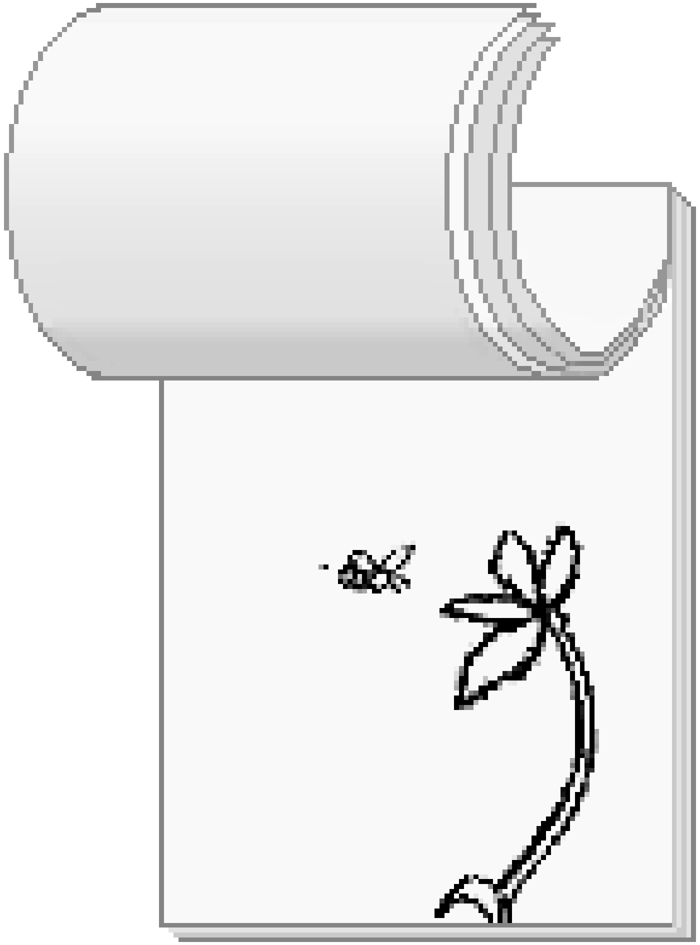
- Aims: More realistic and expressive, less labour intensive
- Methods and animation principles developed in traditional animation also applies in computer animation



Flip Books

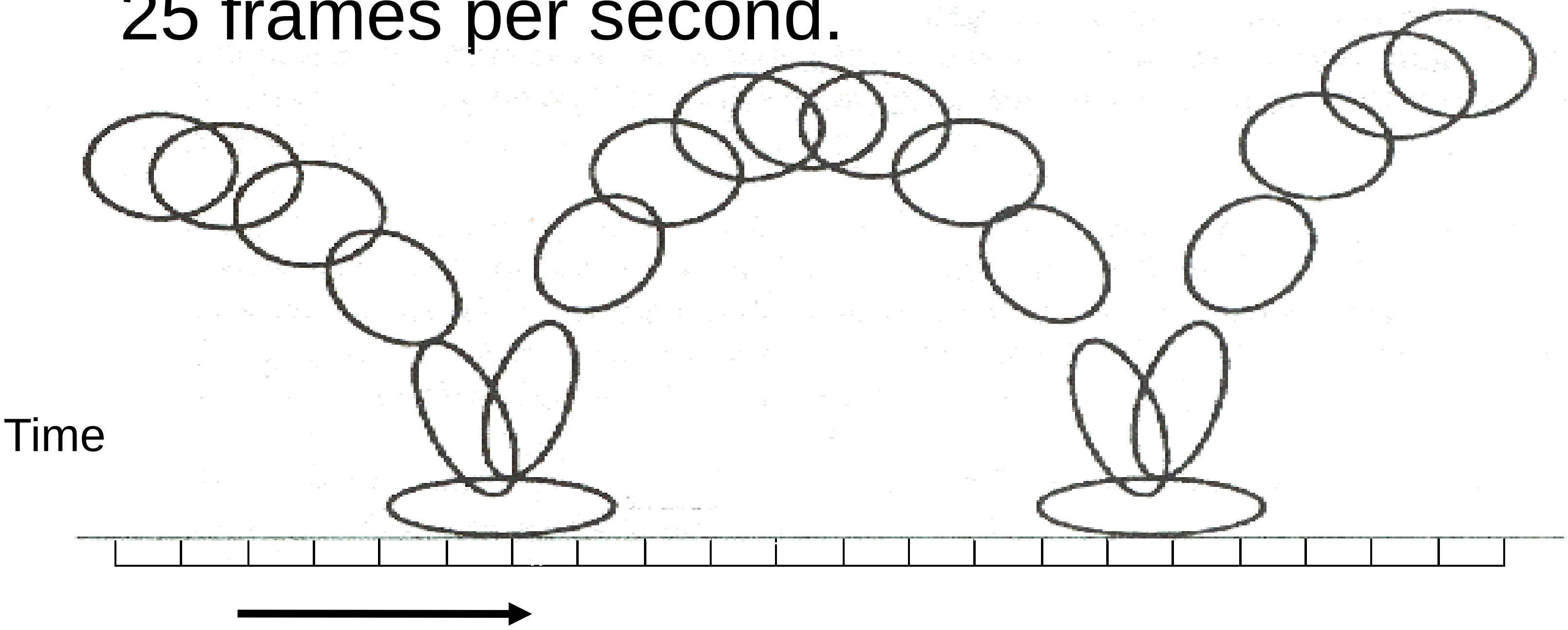
- The most basic form of animation is the flip book
- Presents a sequence of images in quick succession

Flip book Animation.pptx



The Time Line

- Animation is a sequence of frames (images) arranged along a time-line
- In films a sequence of images is displayed at 25 frames per second.



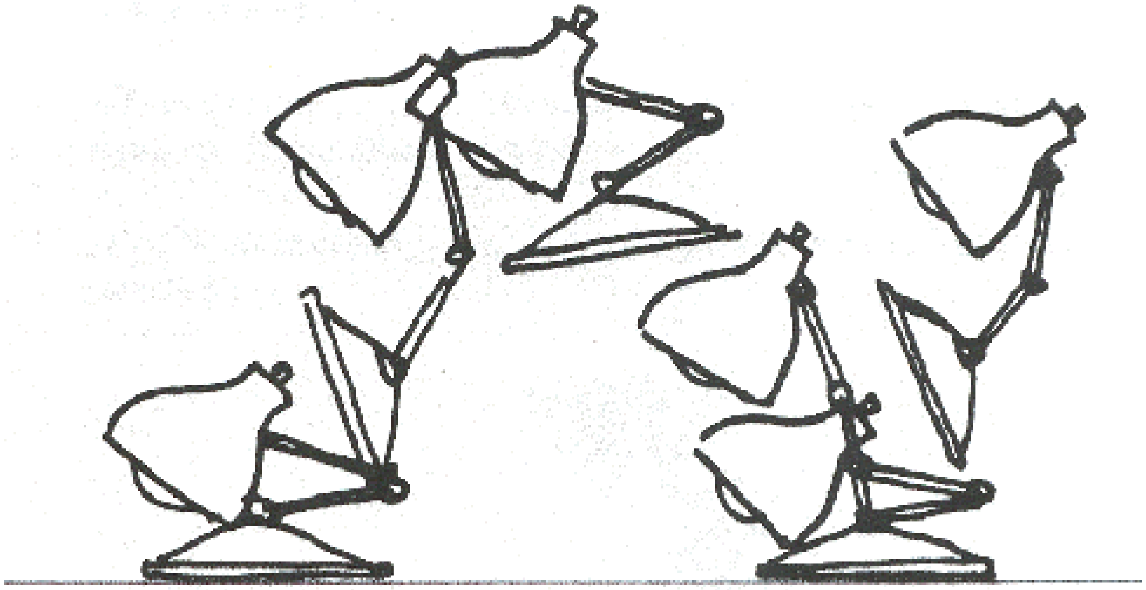
Lasseter '87

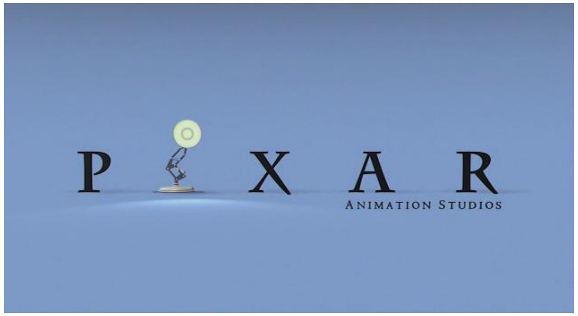
Frames

- Each frame is an image
- Traditionally each image had to be hand drawn individually
- This potentially requires vast amounts of work from a highly skilled animator

Key Frame System

- The head animator draws the most important frames (Keyframes)
- An assistant draws the in-between frames (inbetweens)





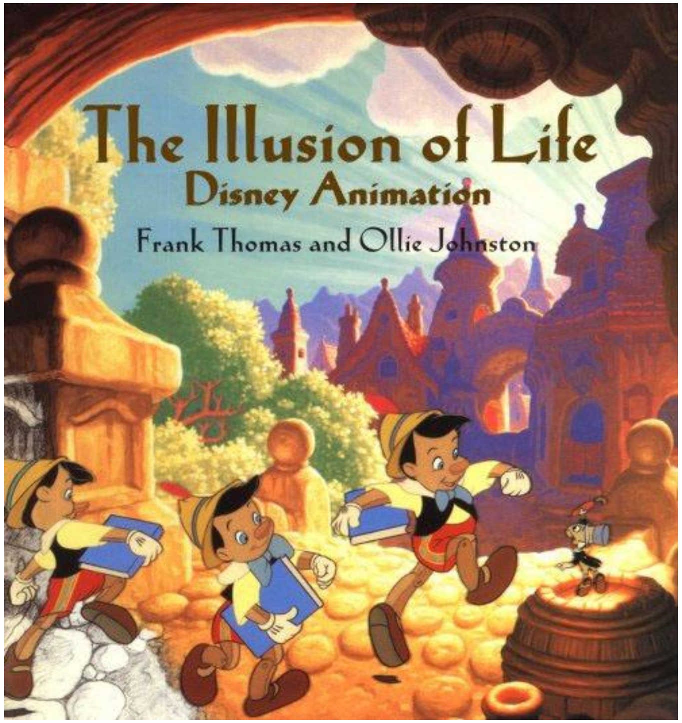
Luxo Jr. by Lasseter 1986

Layers

- Have a background image that does not move
- Put foreground images on a transparent slide in front of it
- Only have to animate bits that move
- Next time you watch an animation notice that the background is always more detailed than the characters

Animation Principles

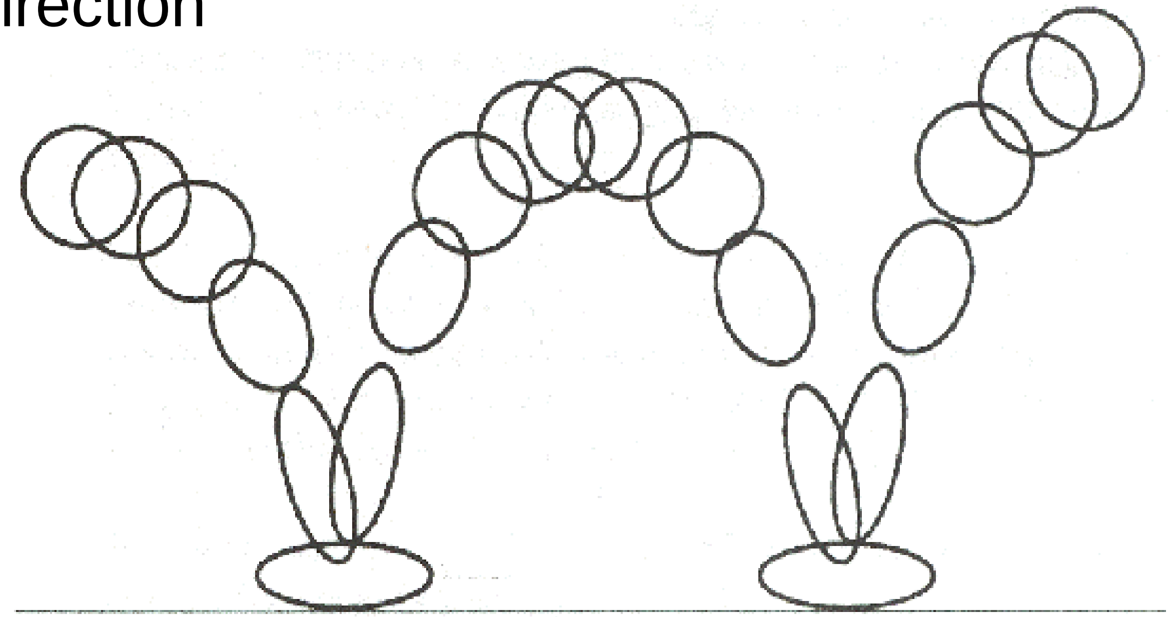
- *“The illusion of Life: Disney Animation”*



Ollie Johnston and Frank Thomas,1981

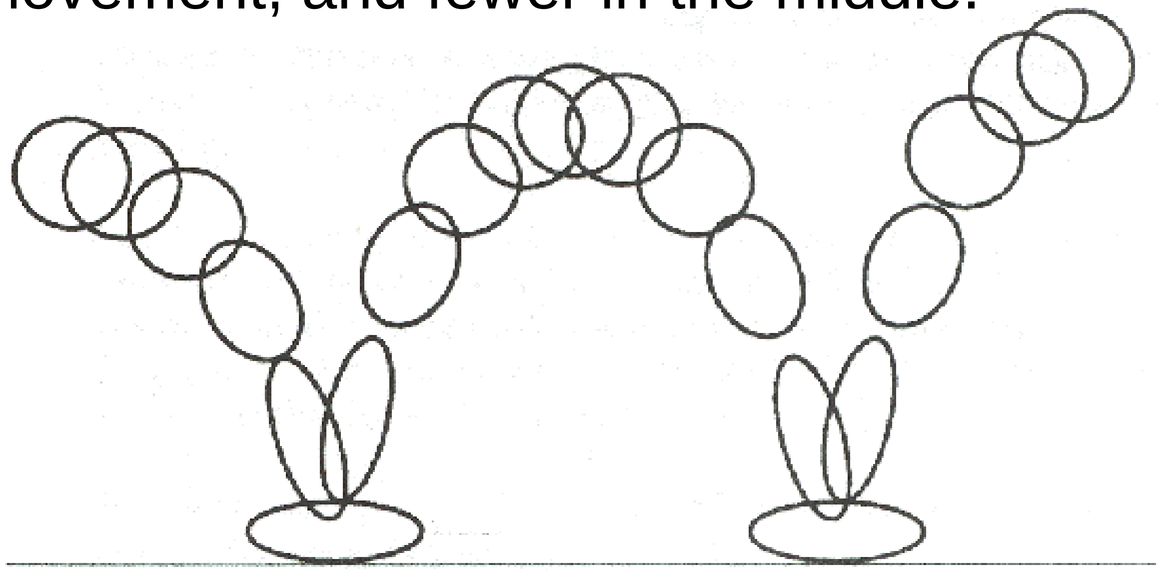
Animation Principles

- Squash and stretch
  - Change the shape of an object to emphasise its motion
  - In particular stretch then squash when changing direction



Animation Principles

- Slow in slow out
  - In real life an object needs time to accelerate and slow down.
  - An animation looks more smooth and realistic with more frames in the beginning and end of a movement, and fewer in the middle.



Stop Motion Animation

- Create models of all your characters
- Pose them
- Take a photo
- Move them slightly
- Take another photo

Stop Motion Animation

- More effort on Creating Characters
- A lot of detail
- Each individual frame is less work

Computer Animation

Computer Animation

- Similar to Stop Motion Animation
  - First to create 3D computer graphics models (some static, some can be animated!)
  - Create the animation frame by frame (pose)
  - Finally render the images considering camera position and lighting (take a photo)

Key Frame animation and Interpolation

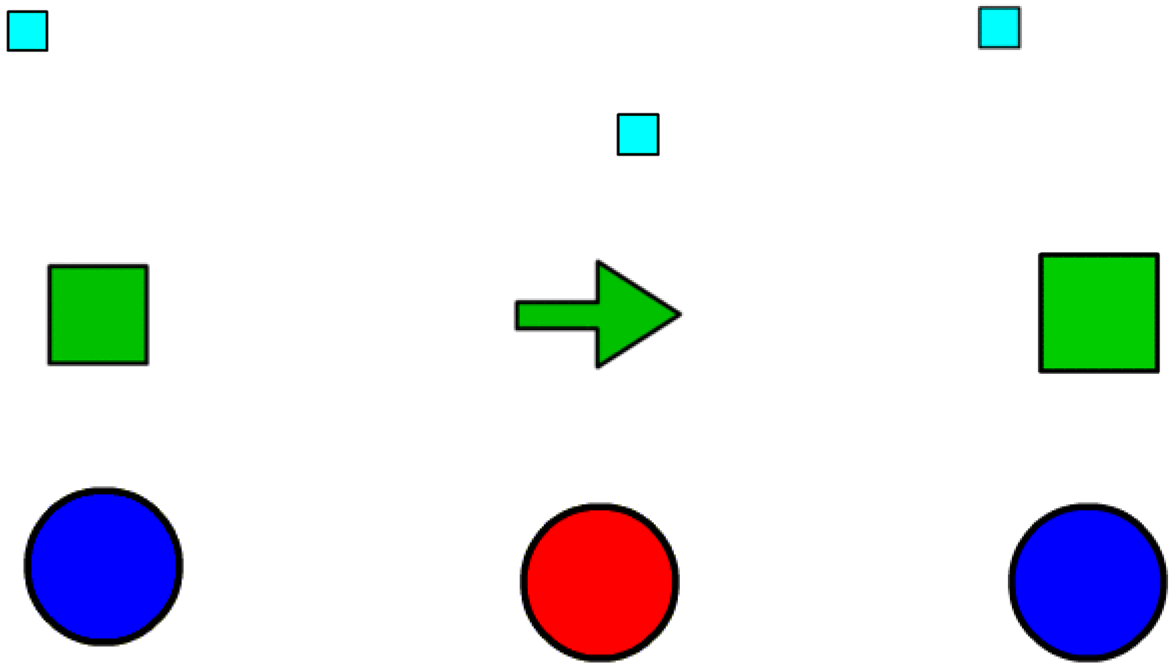
- Computer animation basics
- Computer based key frame system
- Interpolations methods
- Rotations and Quaternions

Key Frame Animation

- The starting point for computer animation is the automation of many of the techniques of traditional animation
- The labour savings can be greatly increased

Key Frame Animation

Key frame: Start | **Key frame: End** | **Animation**



Key Frame Animation

- Normally in computer animation objects are 3D models rather than images
- We can animate one property of the object or a few properties at the same time
  - e.g. position, rotation, normal map, …
- Only changing properties need animation
  - e.g. you can rotate an object without having to do anything to the texture

Key Frame Animation

- Keyframes are “key poses” of the animated model
- Keyframe is defined as (a tuple):

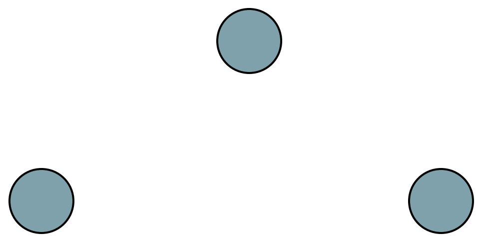
$$< time, value >$$

- The computer can do the inbetweening

Key Frame Animation

- Example 1: simple object movement

<0,[0,0]>,<1,[1,1]>,<2,[2,0]>

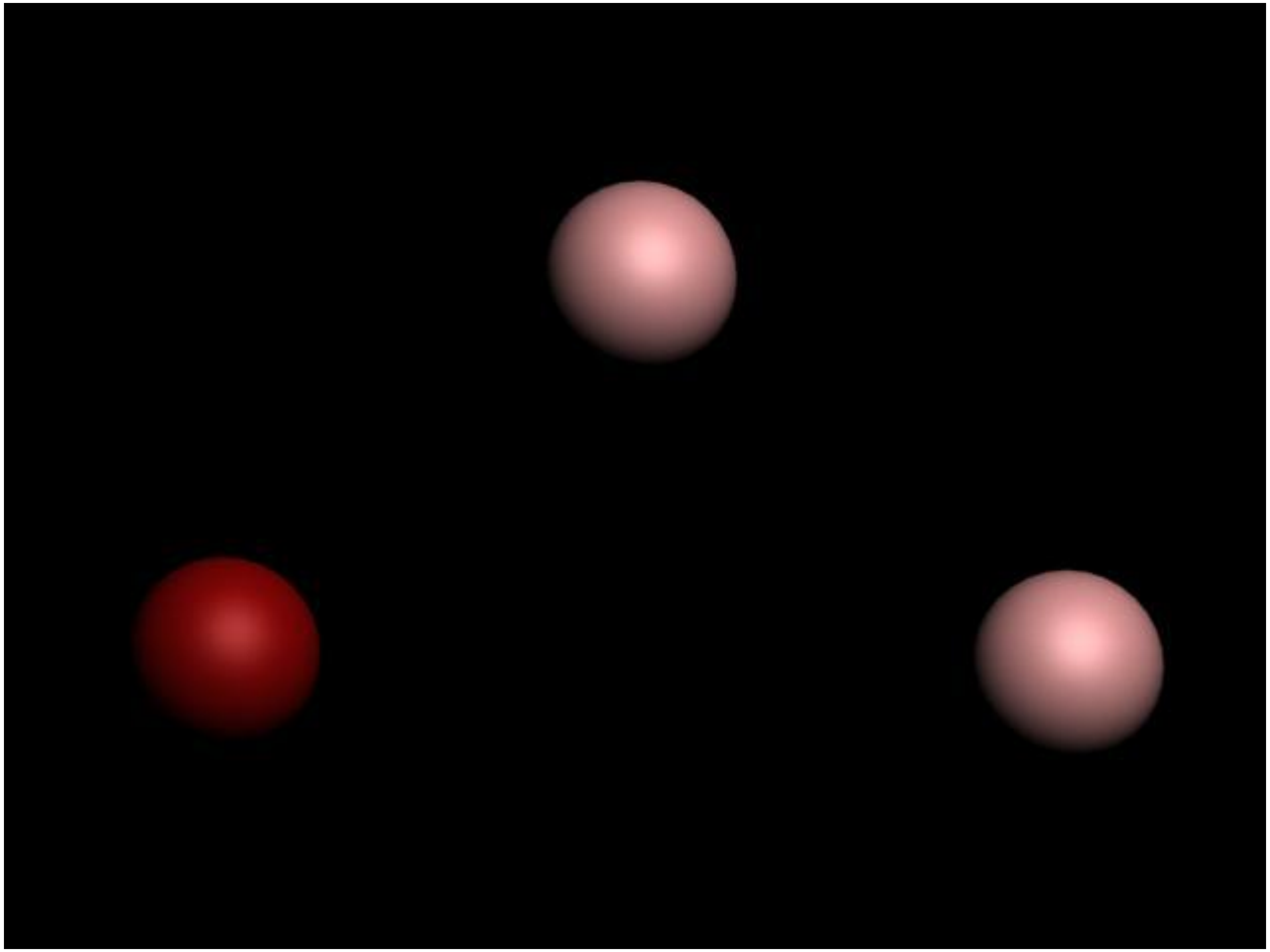


- Example 2: head movements: nod slowly and then shake quickly

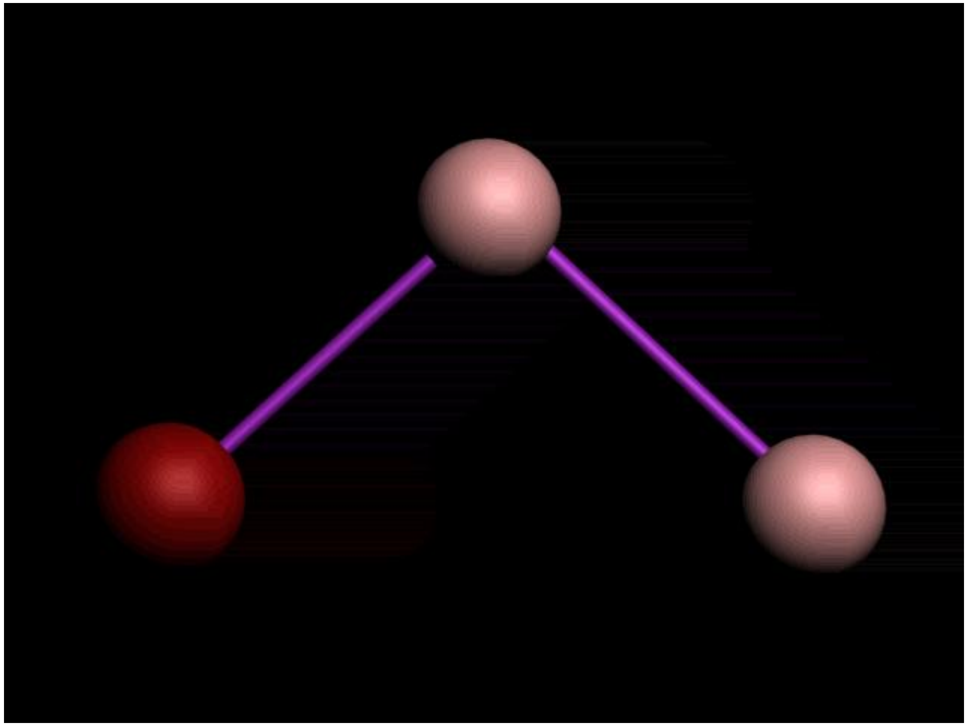
<0,up><1,down>,<3,up>,<5,down><7,up>

<8,up><8.5,left><9,right><9.5 ,left><10,right>…

Key Frame Animation (positions)



Linear Interpolation



- videos\linear.mov

Linear Interpolation

- The position is interpolated linearly between keyframes

$$\mathbf{P}(t) = \frac{t - t_{k-1}}{t_k - t_{k-1}} \mathbf{P}(t_k) + \left(1 - \frac{t - t_{k-1}}{t_k - t_{k-1}}\right) \mathbf{P}(t_{k-1})$$

When t goes from 0 to 1 we have:

$$\mathbf{P}_t = t\mathbf{P}_1 + (1 - t)\mathbf{P}_0$$

Linear Interpolation

$$\mathbf{P}_t = t\mathbf{P}_1 + (1-t)\mathbf{P}_0$$

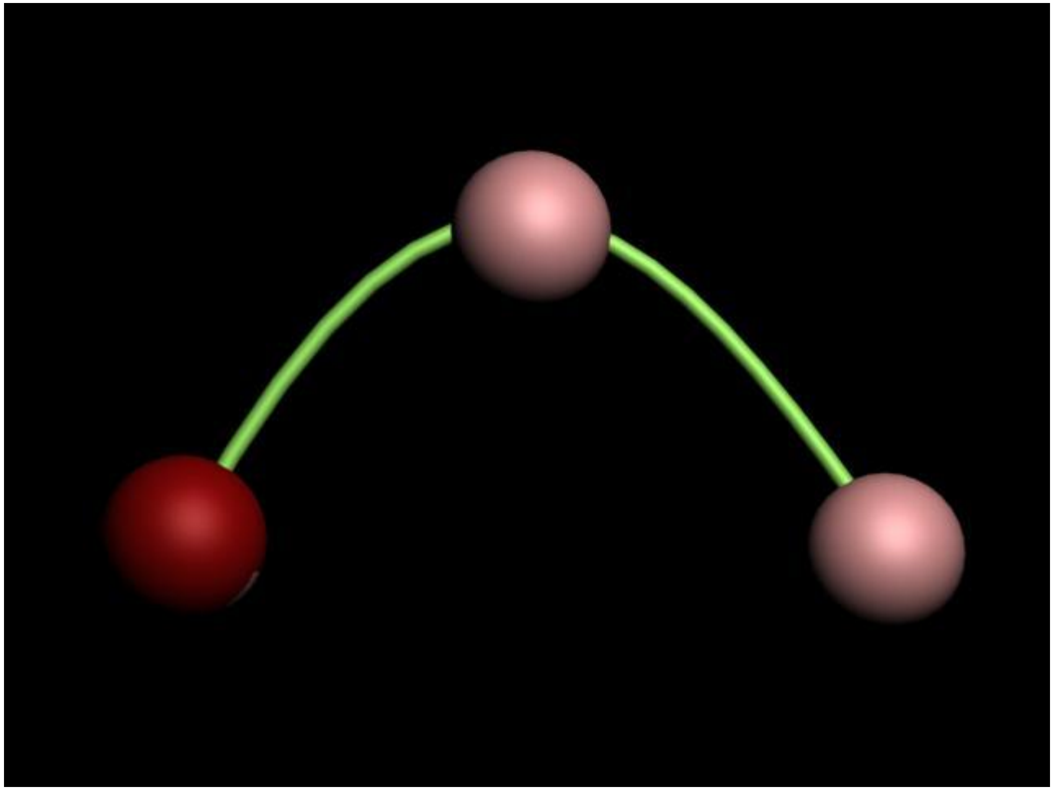
- Returning an interpolation between two inputs (p0,p1) for a parameter (t) in the range [0, 1]:

```
float lerp(float p0, float p1, float t) {
    return v1*t+v0*(1-t);
}
```

Linear Interpolation

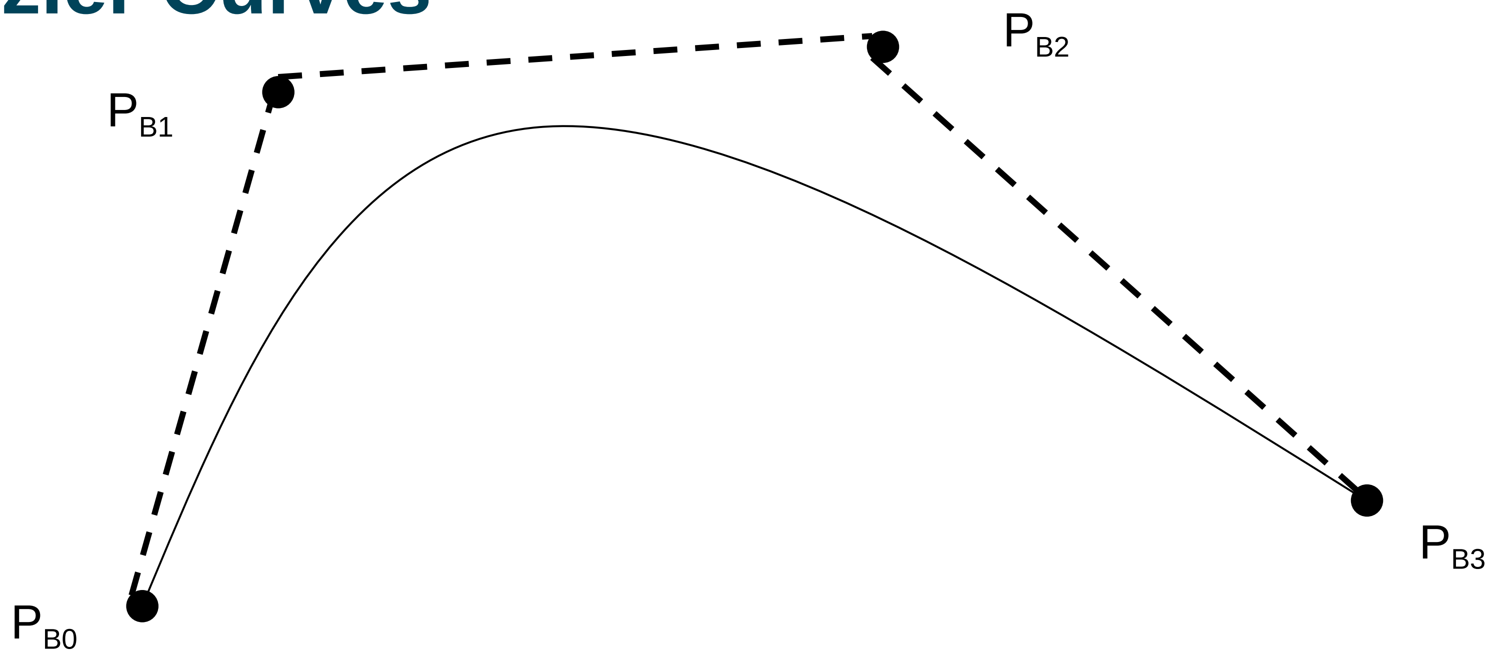
- The animation can be jerky
- Use smooth curves similar to Bezier instead

Spline Interpolation



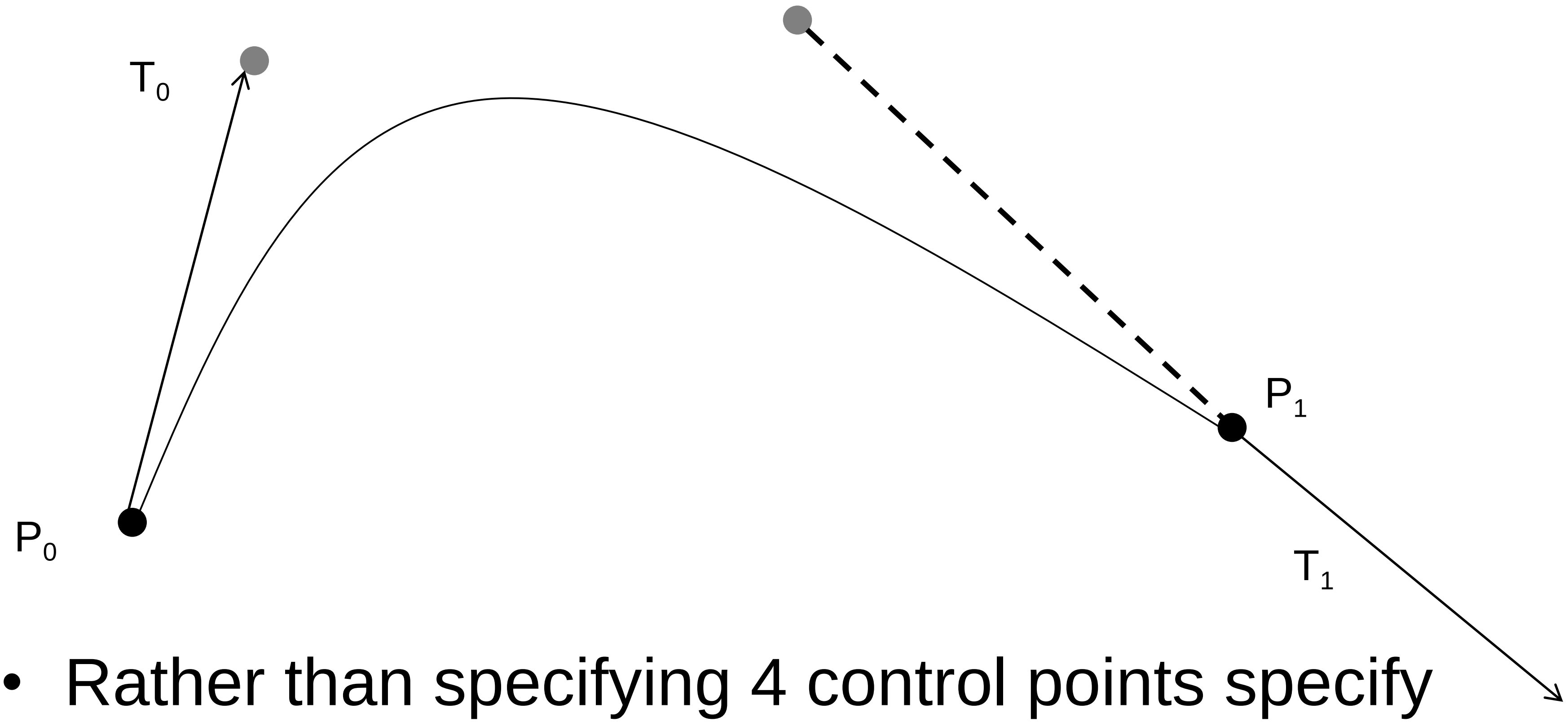
videos\Spline.mov

Bezier Curves



- Smooth but don't go through all the control points, we need to go through all the keyframes

Hermite Curves



- Rather than specifying 4 control points specify 2 end points and tangents at these end points
- In the case of interpolating positions the tangents are velocities

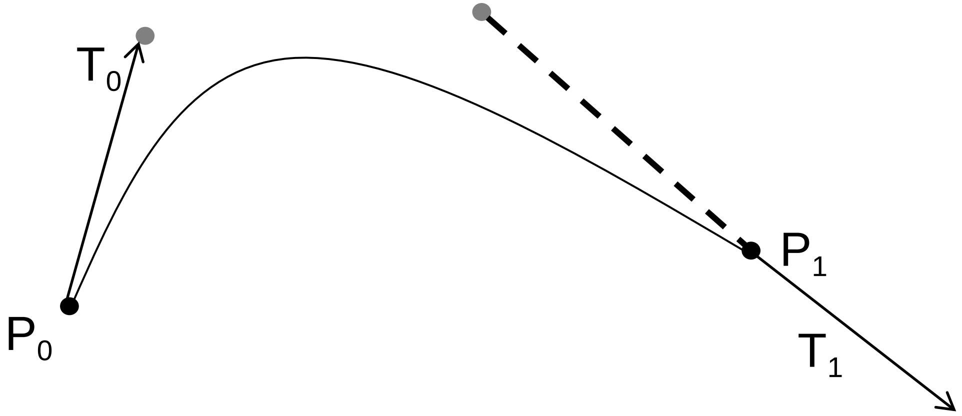
Hermite Curves

$C(0) = P_0$

$C(1) = P_1$

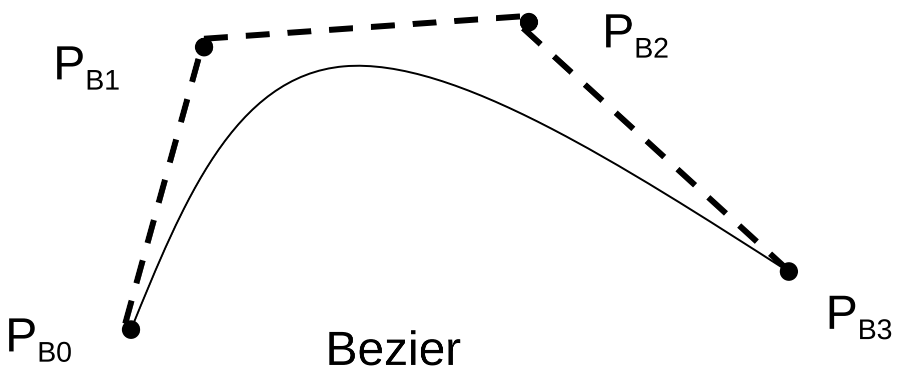
$C'(0) = T_0$

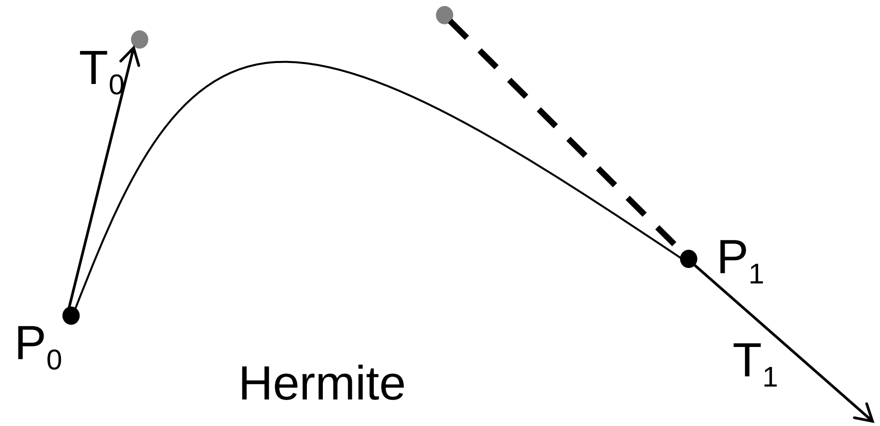
$C'(1) = T_1$



$$C(t) = (2t^3 - 3t^2 + 1)\mathbf{P}_0 + (t^3 + 2t^2 + t)\mathbf{T}_0$$
$$+ (-2t^3 + 3t^2)\mathbf{P}_1 + (t^3 - t^2)\mathbf{T}_1$$

Hermite Curves and Bezier Curves





$$C(t) = (1-t)^3\mathbf{P}_{B0} + 3t(1-t)^2\mathbf{P}_{B1} + 3t^2(1-t)\mathbf{P}_{B2} + t^3\mathbf{P}_{B3}$$

$$C(t) = (2t^3 - 3t^2 + 1)\mathbf{P}_0 + (t^3 + 2t^2 + t)\mathbf{T}_0 + (-2t^3 + 3t^2)\mathbf{P}_1 + (t^3 - t^2)\mathbf{T}_1$$

Bezier to Hermite:

$$P_{B0} = P_0$$

$$P_{B1} = \frac{1}{3}T_0 + P_0$$

$$P_{B3} = P_1$$

$$P_{B2} = P_1 - \frac{1}{3}T_1$$

Hermite to Bezier:

$$P_0 = P_{B0}$$

$$T_0 = 3(P_{B1} - P_{B0})$$

$$P_1 = P_{B3}$$

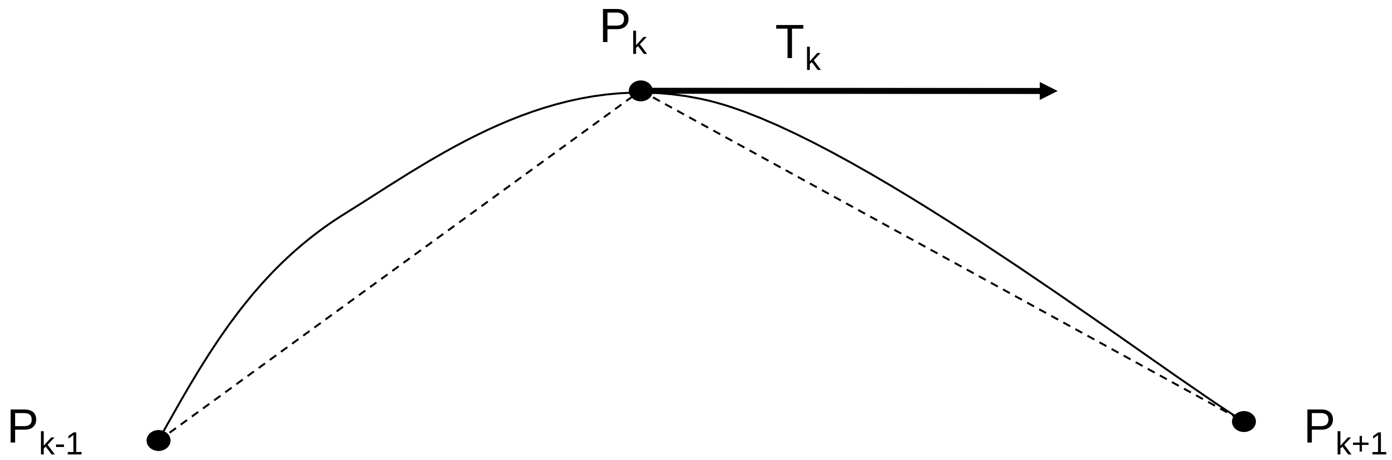
$$T_1 = 3(P_{B3} - P_{B2})$$

Tangents

- Now given two control points as well as the tangents at these points, we can interpolate the position at a given time.
- Where do we get the tangents (velocities) from?
- We could directly set them, they act as an extra control on the behaviour
- However often we want to generate them automatically

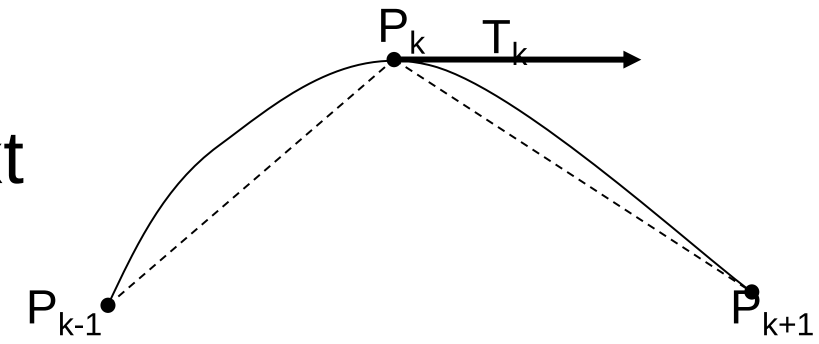
Tangents

- Base the tangent as a keyframe on the previous and next keyframe
- Obtained the tangent from the pervious keyframe and to the next one



Tangents

- Average the distance from the previous keyframe and to the next one



$$\mathbf{T}_k = \frac{1}{2}\left(\mathbf{P}_{k+1} - \mathbf{P}_{k-1}\right)$$

- If you set the tangents at the first and last frame to zero you get slow in slow out

Almost perfect…

- That's pretty much it on keyframe animation
- But there's one last problem: Rotations
- Rotations are used a lot on animation
- In fact human body animation is largely based on animating rotations rather than positions

Rotations

- Rotations are very different from positions
- They are essentially spherical rather than linear
- You can split them into rotations about the X,Y & Z axis (Euler angles), but:
  - Then the order in which you do them changes in final rotation
  - If you rotate about Y so that the Z axis is rotated onto the X axis you get stuck (Gimbal lock) and are in trouble

videos\gimbal-minus3.flv

Quarternions

- We need a representation of rotations that doesn’t suffer these problems
- We use Quaternions
- Invented by William Rowan Hamilton in 1843
- Introduced into computer animation by Ken Shoemake
  - K. Shoemake, “Animating rotations with quaternion curves”, ACM SIGGRAPH 1985 pp245-254

Quaternions

- Quaternions are a 4D generalisation of complex numbers:

$$\mathbf{q} = w + v_x i + v_y j + v_z k$$

- The last three terms are the imaginary part and are often written as a vector:

$$\mathbf{q} = [w, \mathbf{v}]$$

Quaternions Properties

- The conjugate of a quaternion is defined as:

$$\overline{\mathbf{q}} = [w, -\mathbf{v}]$$

- And multiplication is defined as:

$$\mathbf{q}_1\mathbf{q}_2 = [w_1 w_2 - \mathbf{v}_1 \bullet \mathbf{v}_2,\ w_1\mathbf{v}_2 + w_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2]$$

(Non commutative)

Quaternions Properties

- The inverse of a quaternion is defined as:

$$\mathbf{q}^{-1} = \frac{1}{|\mathbf{q}|^2}[w, -\mathbf{v}]$$

- For a unit quaternion (magnitude = 1) we have:

$$\mathbf{q}^{-1} = \overline{\mathbf{q}}$$

Quaternion Rotations

- A rotation of angle θ about an axis **u** is represented as a quaternion with (u is a unit vector):

$$w = \cos\left(\frac{\theta}{2}\right), \mathbf{v} = \vec{u}\sin\left(\frac{\theta}{2}\right)$$

- Now we have: $q = [\cos\left(\frac{\theta}{2}\right), \vec{u}\sin\left(\frac{\theta}{2}\right)]$

- All rotations are represented by unit quaternions (norm1)

Quaternion Rotations

$$w = \cos\left(\frac{\theta}{2}\right), \mathbf{v} = \vec{u}\sin\left(\frac{\theta}{2}\right)$$

$$q = [\cos\left(\frac{\theta}{2}\right), \vec{u}\sin\left(\frac{\theta}{2}\right)]$$

- [*w*, ***v***] and [-*w*, -***v***] specify the same rotation

[w,v] = [cos(θ/2), usin(θ/2)]

[-w,-v] = [-cos(θ/2), -usin(θ/2)]

= [cos((2**π**- θ)/2),-usin((2 **π**- θ)/2)]

Quaternion Rotations

- A vector (**V**) is rotated by first converting it to a quaternion:

$$\mathbf{v} = \left[0, \mathbf{V}\right]$$

- Premultiplying by the rotation and postmultiplying by its inverse

$$\mathbf{v}_R = \mathbf{q}\mathbf{v}\mathbf{q}^{-1} \qquad (\mathbf{q}^{-1} = \bar{\mathbf{q}})$$

- And transforming back to a vector

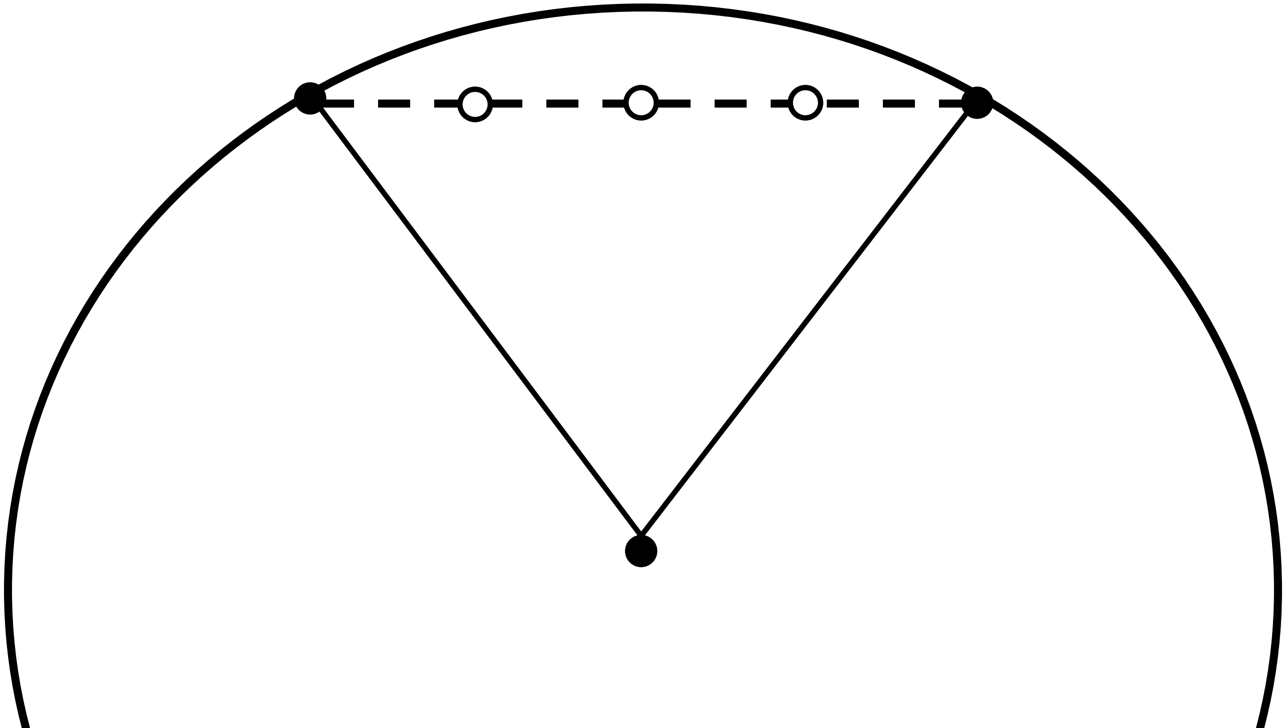
Quaternion Rotations

- A series of rotations can be concatenated into a single representation by a quaternion multiplication.
- A rotation by a quaternion p followed by a rotation by a quaternion q on a vector v:

$$v_R = q(pvp^{-1})q^{-1} = (qp)v(qp)^{-1}$$

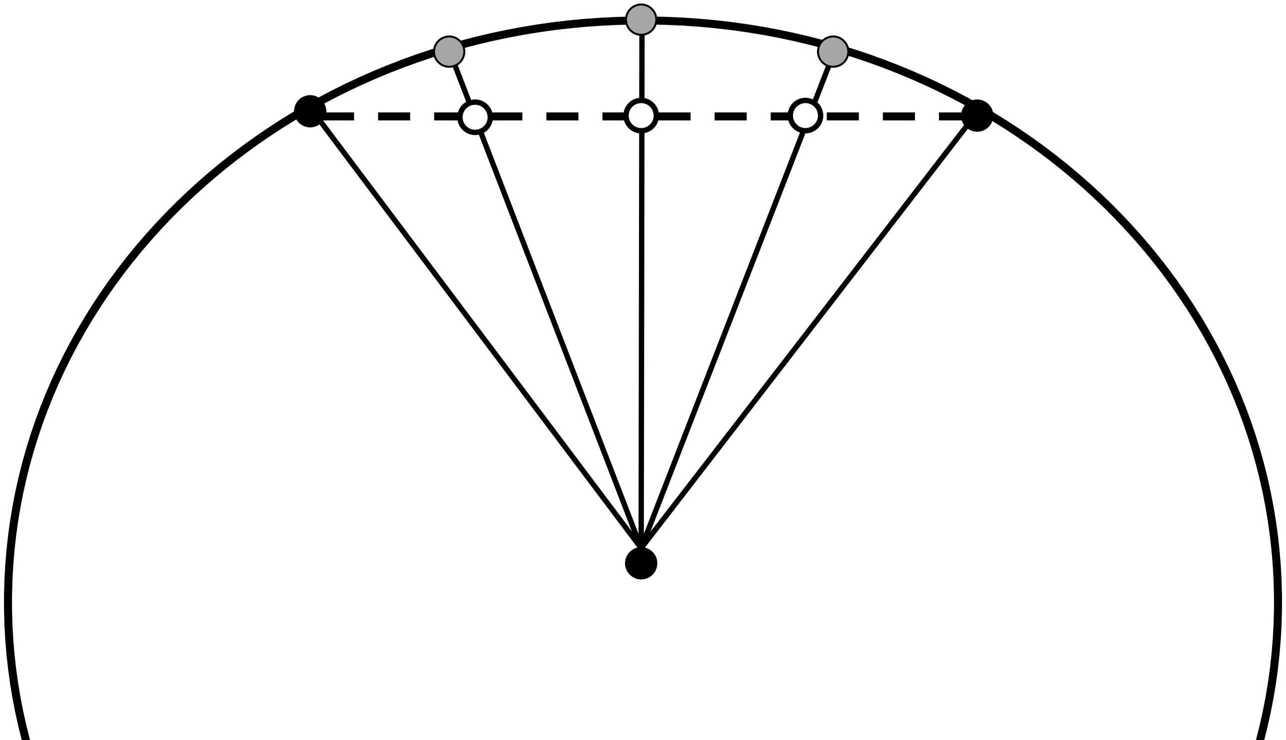
Interpolating Quaternions

- As quaternions have unit length, they all lie on a sphere with centre on the origin
- Interpolating normally will result in a quaternion that is not unit length



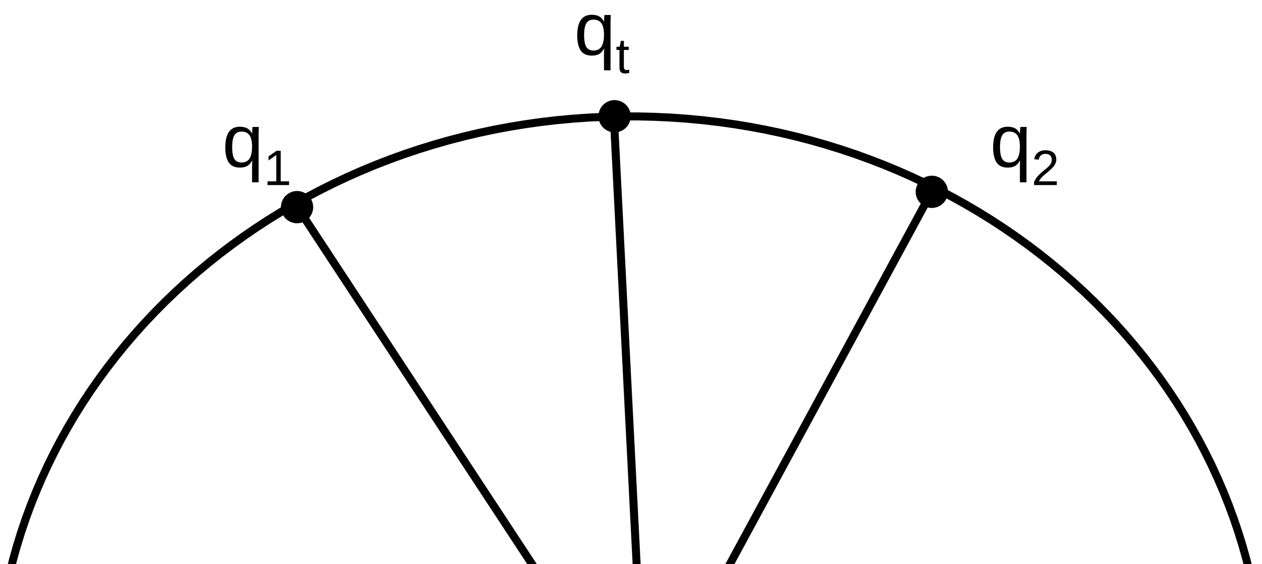
Interpolating Quaternions

- You can renormalise
- But it will not maintain constant speed along the surface of the sphere



Interpolating Quaternions

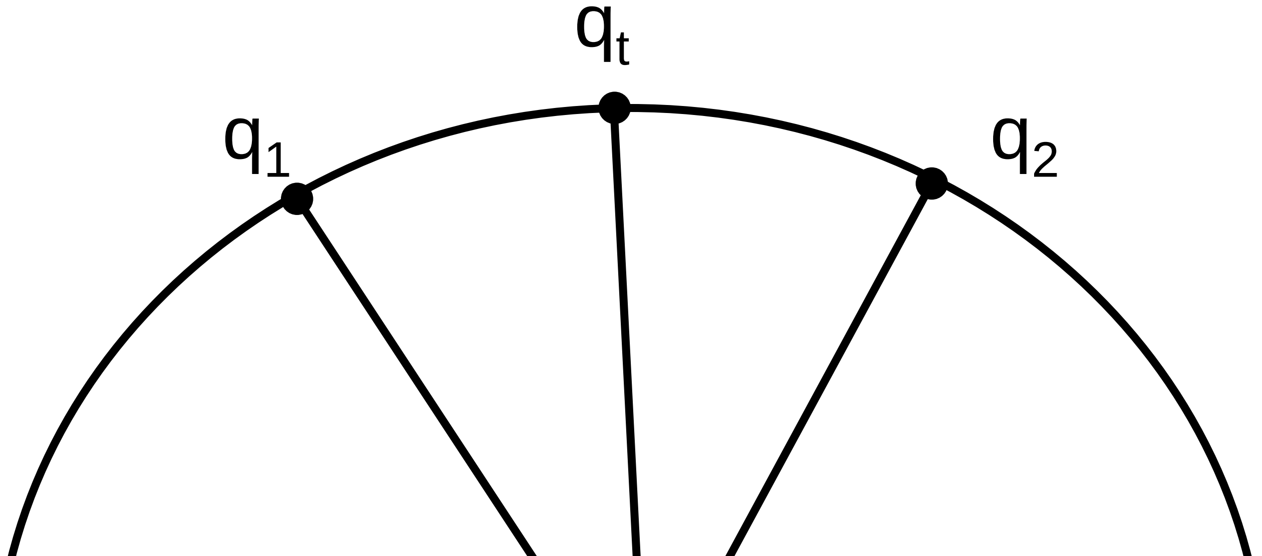
- Shoemake introduced Spherical Linear Interpolation (SLERP) which interpolates based on the angle at the centre



SLERP

- Interpolate using the sin of θ:

$$q_1 \frac{\sin((1-t)\theta)}{\sin\theta} + q_2 \frac{\sin(t\theta)}{\sin\theta}$$



SLERP

- [*w*, ***v***] and [-*w*, -***v***] specify the same rotation
- So 2 quaternions on the opposite sides of the hypersphere are the same rotation
- Before doing SLERP we project the 2 quaternions onto the same side
- If $\cos\theta < 0$ negate q_2

$$\cos(\theta) = q_1 \bullet q_2$$

$$= s_1 s_2 + v_1 \bullet v_2$$