Computer Graphics

P. Healy

CS1-08
Computer Science Bldg.
tel: 202727
patrick.healy@ul.ie

Spring 2021–2022

Outline

Surface Detail: §18

Basic stages of modelling and rendering an object to make it "realistic":

Produced with – what else! – POV-Ray. Original here.

- wire-frame display of object: position, lighting calcs, etc.
- surface layers fitted over object that gives a smooth-surface view
- surface details then added: clothing patterns, cloth textures, skin features, such as, moles, freckles, pores, etc.

Surface Detail (contd.)

Methods to add surface detail:

- Paste small objects, such as flowers, spines, on to a larger surface
- Model surface patterns with small polygon areas
- Map texture arrays or intensity-modifying procedures onto a surface
- Modify surface normal vector to create localized bumps
- Modify both surface normal vector and surface tangent vector to display directional patterns on wood and other materials

Texture Mapping; §18-2

- Idea of texture mapping is to add detail to an object in order to make it look more realistic
- The texture pattern may be defined either in terms of an array of colour values or a procedure that modifies object colours
- Any method of generating textures gives rise to a **texture space** and it may be referenced with **texture coordinates**
- example

Linear Textures

- We set up a one-dimensional array of colours
- The texture space is referenced with a single s-coordinate where
  - $s = 0.0$ designates the first colour in the array
  - $s = 1.0$ designates the last colour
  - $s = 0.5$ designates the middle colour of the array
- To map a linear texture into a scene we assign one s-value, say, s_1, to one position in space and another s-coordinate, s_2, to a second spatial position
- The colour of the line between the two points in space is determined by the section of the colour array between s_1 and s_2
- The colour of the pixels on the line are **linearly interpolated**; distribution of colours on line depends on ratio of pixels to colours
- If necess. round off coords to get an integer position in array

Surface Texture Patterns

- The linear texture space idea can be generalised to more than one dimension
- We now have a 2-D array of colours that are accessed with two-dimensional (s, t) coordinate values
- For example a texture pattern could be defined with 16×16 colours
- Given 3 points in space these can be mapped to 3 coordinates of the texture space by a linear transformation and all points within the triangle in space can get mapped to a colour point (an array element), also
- This idea can be extended further to **volume texture patterns** in a 3-D texture space with coordinates (s, t, r)

Texture Patterns: Other Issues

- A processing-saving idea is to introduce **mip maps** or multiple scaled versions of a texture
- In animations as an object gets scaled instead of applying the full-size texture map, scaled versions of it (*at lower resolutions*) can be applied when viewing the object at different sizes
- Problems:
  - Not good for rough surface appearances *e.g.*, raisins, oranges
  - Light-intensity tends to be made uniform across texture: also a problem
- **Bump Mapping** (§18-3) details

Introduction to Computer Animation – §12 **HBC**

- **Computer Animation:** any time sequence of visual changes in a picture
- Change in object position, size, colour, transparency surface texture
- In films / advertising often have case of transforming one object in to another; and morphing
- Animation can also include variations in lighting effects or camera parameters such as position, orientation, focal length
- Animations often require realism e.g. natural phenomena or flight simulators
- Two basic methods for constructing a motion sequence:
  - real-time animation
  - frame-by-frame animation

Introduction to Computer Animation (contd.)

A Common Approach

- **Storyboard Layout**: the plan; the basic design
- **Object Definition**: shapes for each characters
- **Key Frame**: detailed drawing of the scene at a point in time with every object (character) in a specified position; development of key frames is generally responsibility of senior animator
- **In-betweens**: intermediate frames between key frames; "stepping stones" between key frames; may be generated automatically
- Some figures:
  - Film requires 24 frames / sec, so a 1-minute film requires 1440 frames; if 4 in-betweens are required for each pair of key frames, this requires 288 key frames

Introduction to Computer Animation (contd.)

A Common Approach

- **Storyboard Layout**: the plan; the basic design
- **Object Definition**: shapes for each characters
- **Key Frame**: detailed drawing of the scene at a point in time with every object (character) in a specified position; development of key frames is generally responsibility of senior animator
- **In-betweens**: intermediate frames between key frames; “stepping stones” between key frames; may be generated automatically
- Some figures:
  - Film requires 24 frames / sec, so a 1-minute film requires 1440 frames; if 4 in-betweens are required for each pair of key frames, this requires 288 key frames

Introduction to Computer Animation (contd.)

A Common Approach

- **Storyboard Layout**: the plan; the basic design
- **Object Definition**: shapes for each characters
- **Key Frame**: detailed drawing of the scene at a point in time with every object (character) in a specified position; development of key frames is generally responsibility of senior animator

"Tin Toy" / RenderMan

- **In-betweens**: intermediate frames between key frames;

Introduction to Computer Animation (contd.)

A Common Approach

- **Storyboard Layout**: the plan; the basic design
- **Object Definition**: shapes for each characters
- **Key Frame**: detailed drawing of the scene at a point in time with every object (character) in a specified position; development of key frames is generally responsibility of senior animator
- **In-betweens**: intermediate frames between key frames; “stepping stones” between key frames; may be generated automatically
- Some figures:
  - Film requires 24 frames / sec, so a 1-minute film requires 1440 frames; if 4 in-betweens are required for each pair of key frames, this requires 288 key frames

Introduction to Computer Animation (contd.)

A Common Approach

- **Storyboard Layout**: the plan; the basic design
- **Object Definition**: shapes for each characters
- **Key Frame**: detailed drawing of the scene at a point in time with every object (character) in a specified position; development of key frames is generally responsibility of senior animator
- **In-betweens**: intermediate frames between key frames; “stepping stones” between key frames; may be generated automatically
- Some figures:
  - Film requires 24 frames / sec, so a 1-minute film requires 1440 frames; if 4 in-betweens are required for each pair of key frames, this requires 288 key frames

Introduction to Computer Animation (contd.)

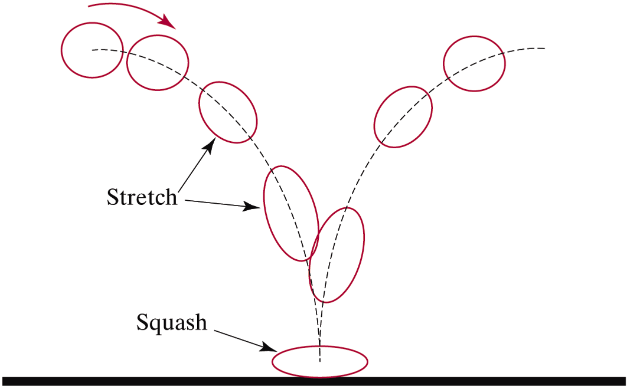
A Common Approach

- **Storyboard Layout**: the plan; the basic design
- **Object Definition**: shapes for each characters
- **Key Frame**: detailed drawing of the scene at a point in time with every object (character) in a specified position; development of key frames is generally responsibility of senior animator
- **In-betweens**: intermediate frames between key frames; “stepping stones” between key frames; may be generated automatically
- Some figures:
  - Film requires 24 frames / sec, so a 1-minute film requires 1440 frames; if 4 in-betweens are required for each pair of key frames, this requires 288 key frames

Introduction to Computer Animation (contd.)

Traditional Animation

- Classical motion trick:



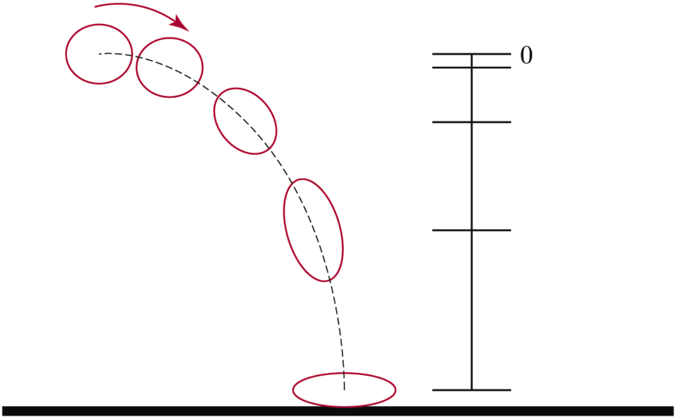
- Giving impression of acceleration:

Introduction to Computer Animation (contd.)

Traditional Animation

- Classical motion trick:

- Giving impression of acceleration:



Key-Frame Systems

- Once details of key frames have been decided the in-betweens bow need to be generated
- Motion paths can be specified with a *kinematic description* as a set of spline curves or they can be *physically based* by specifying the forces acting on the objects
- For complex object transformations shapes of objects will likely change over time
- If surface has been described using a polygon mesh then it is likely that more than simply positioning of vertices will change viz. no. of vertices and edges will also change
- This is related to the **morphing** (metamorphosis)

Morphing

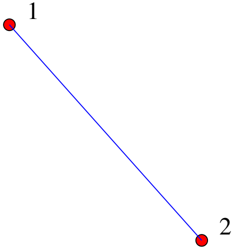
- In order to morph from one key frame to another we insert (as needed) dummy vertices in the first (or second) so that a correspondence can be made

- A linear interpolation is then made to calculate the in-betweens

- Another example:

Morphing

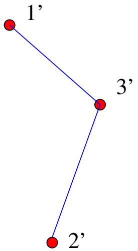
- In order to morph from one key frame to another we insert (as needed) dummy vertices in the first (or second) so that a correspondence can be made



- A linear interpolation is then made to calculate the in-betweens
- Another example:

Morphing

- In order to morph from one key frame to another we insert (as needed) dummy vertices in the first (or second) so that a correspondence can be made

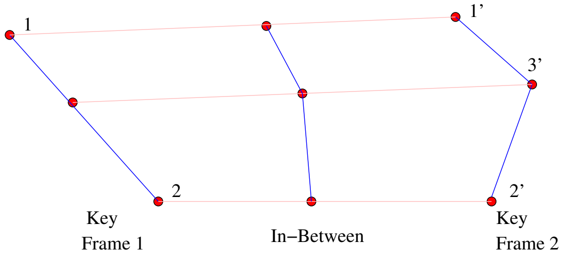


- A linear interpolation is then made to calculate the in-betweens
- Another example:

Morphing

- In order to morph from one key frame to another we insert (as needed) dummy vertices in the first (or second) so that a correspondence can be made

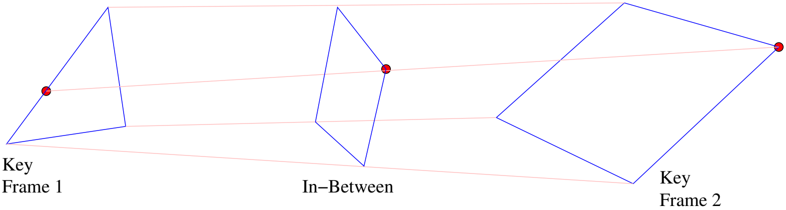
- A linear interpolation is then made to calculate the in-betweens



- Another example:

Morphing

- In order to morph from one key frame to another we insert (as needed) dummy vertices in the first (or second) so that a correspondence can be made
- A linear interpolation is then made to calculate the in-betweens
- Another example:



Frame Transitions

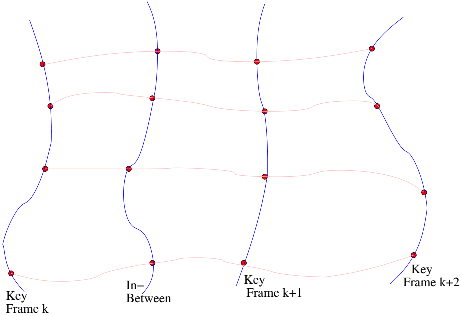
- We are not bound to use *linear* interpolation
- Sometimes we may wish to use **nonlinear** paths
  for the transitioning of a point from one key frame to another
- Similarly we may not wish for all in-betweens to be equally spaced viz. acceleration
- For case of **constant** motion if there are n in-betweens separating key frame k (at time t_k) and key frame $k+1$ (at time t_{k+1}) then the time spent on each in-between is

$$\Delta t = \frac{t_{k+1} - t_k}{n+1}$$

  Then in-between B_j occurs at time $t_{B_j} = t_k + j\Delta t$.
  This time value is used in lin. interpolation formula to calculate transitions of colour, position, etc.

Frame Transitions

- We are not bound to use *linear* interpolation
- Sometimes we may wish to use **nonlinear** paths
  for the transitioning of a point from one key frame to another



- Similarly we may not wish for all in-betweens to be equally spaced viz. acceleration
- For case of **constant** motion if there are n in-betweens separating key frame k (at time t_k) and key frame $k+1$ (at

Frame Transitions

- We are not bound to use *linear* interpolation
- Sometimes we may wish to use **nonlinear** paths
  for the transitioning of a point from one key frame to another

- Similarly we may not wish for all in-betweens to be equally spaced viz. acceleration
- For case of **constant** motion if there are n in-betweens separating key frame k (at time t_k) and key frame $k+1$ (at time t_{k+1}) then the time spent on each in-between is

$$\Delta t = \frac{t_{k+1} - t_k}{n+1}$$

  Then in-between B_j occurs at time $t_{B_j} = t_k + j\Delta t$.
  This time value is used in lin. interpolation formula to calculate transitions of colour, position, etc.

Frame Transitions

- We are not bound to use *linear* interpolation
- Sometimes we may wish to use **nonlinear** paths
  for the transitioning of a point from one key frame to another

- Similarly we may not wish for all in-betweens to be equally spaced viz. acceleration
- For case of **constant** motion if there are n in-betweens separating key frame k (at time t_k) and key frame $k+1$ (at time t_{k+1}) then the time spent on each in-between is

$$\Delta t = \frac{t_{k+1} - t_k}{n+1}$$

  Then in-between B_j occurs at time $t_{B_j} = t_k + j\Delta t$.
  This time value is used in lin. interpolation formula to calculate transitions of colour, position, etc.

Frame Transitions (contd.)

- In order to simulate **acceleration** we can use a non-linear function
- What we want is something that has small slope initially and increasing slope later
- A function such as $1 - \cos\theta, 0 < \theta < \pi/2$ achieves this; see also.
- In this case
$$t_{B_j} = t_k + \Delta t\left(1 - \cos\frac{j\pi}{2(n+1)}\right), \quad j = 1, 2, \ldots, n$$
- Similarly a sin function gives effect of deceleration:
$$t_{B_j} = t_k + \Delta t \sin\frac{j\pi}{2(n+1)}, \quad j = 1, 2, \ldots, n$$
- Speed-up and slow-down can also be achieved with a function such as $\frac{1}{2}(1 - \cos\theta)$

Frame Transitions (contd.)

- In order to simulate **acceleration** we can use a non-linear function
- What we want is something that has small slope initially and increasing slope later
- A function such as $1 - \cos\theta, 0 < \theta < \pi/2$ achieves this; see also.
- In this case

$$t_{B_j} = t_k + \Delta t\left(1 - \cos\frac{j\pi}{2(n+1)}\right), \quad j = 1, 2, \ldots, n$$

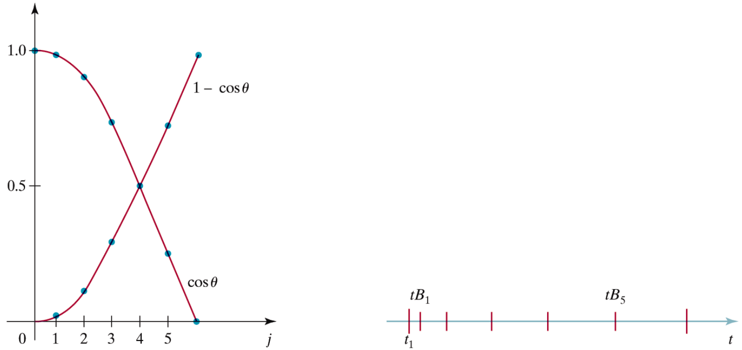
- Similarly a sin function gives effect of deceleration:

$$t_{B_j} = t_k + \Delta t \sin\frac{j\pi}{2(n+1)}, \quad j = 1, 2, \ldots, n$$

- Speed-up and slow-down can also be achieved with a function such as $\frac{1}{2}(1 - \cos\theta)$

Frame Transitions (contd.)

- In order to simulate **acceleration** we can use a non-linear function
- What we want is something that has small slope initially and increasing slope later
- A function such as $1 - \cos\theta, 0 < \theta < \pi/2$ achieves this; see also.



Frame Transitions (contd.)

- In order to simulate **acceleration** we can use a non-linear function
- What we want is something that has small slope initially and increasing slope later
- A function such as $1 - \cos\theta, 0 < \theta < \pi/2$ achieves this; see also.
- In this case

$$t_{B_j} = t_k + \Delta t\left(1 - \cos\frac{j\pi}{2(n+1)}\right), \quad j = 1, 2, \ldots, n$$

- Similarly a $\sin$ function gives effect of deceleration:

$$t_{B_j} = t_k + \Delta t \sin\frac{j\pi}{2(n+1)}, \quad j = 1, 2, \ldots, n$$

- Speed-up and slow-down can also be achieved with a function such as $\frac{1}{2}(1 - \cos\theta)$

Frame Transitions (contd.)

- In order to simulate **acceleration** we can use a non-linear function
- What we want is something that has small slope initially and increasing slope later
- A function such as $1 - \cos\theta, 0 < \theta < \pi/2$ achieves this; see also.
- In this case

$$t_{B_j} = t_k + \Delta t\left(1 - \cos\frac{j\pi}{2(n+1)}\right), \quad j = 1, 2, \ldots, n$$

- Similarly a sin function gives effect of deceleration:

$$t_{B_j} = t_k + \Delta t \sin\frac{j\pi}{2(n+1)}, \quad j = 1, 2, \ldots, n$$

- Speed-up and slow-down can also be achieved with a function such as $\frac{1}{2}(1 - \cos\theta)$

Frame Transitions (contd.)

- In order to simulate **acceleration** we can use a non-linear function
- What we want is something that has small slope initially and increasing slope later
- A function such as $1 - \cos\theta, 0 < \theta < \pi/2$ achieves this; see also.

- In this case

$$t_{B_j} = t_k + \Delta t\left(1 - \cos\frac{j\pi}{2(n+1)}\right), \quad j = 1, 2, \ldots, n$$

- Similarly a sin function gives effect of deceleration:

$$t_{B_j} = t_k + \Delta t \sin\frac{j\pi}{2(n+1)}, \quad j = 1, 2, \ldots, n$$

- Speed-up and slow-down can also be achieved with a function such as $\frac{1}{2}(1 - \cos\theta)$