Animation

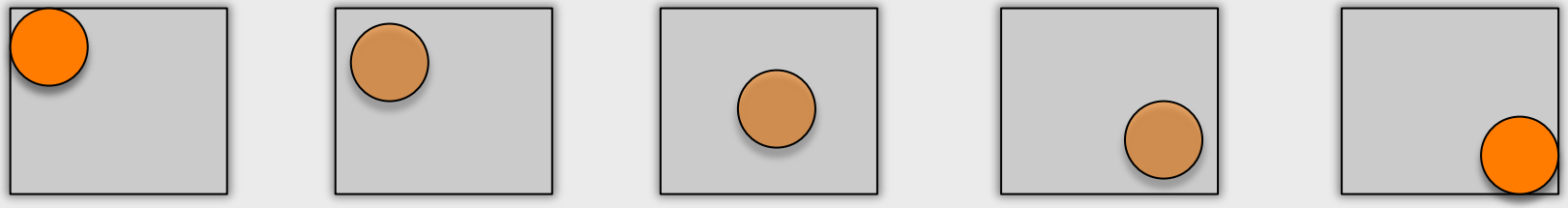
An introduction to real-time
3D animation concepts

Contents

- Keyframe animation
- Interpolators
- Hierarchical animation
- Forward and inverse kinematics
- Humanoid animation and motion capture
- Motion dynamics
- Animating cameras, lights, materials
- Animating geometry deformation
- Animation tricks

Keyframe Animation

- Most animation systems are based on the *key frame animation* technique
- Developed by Walt Disney and his contemporaries when animation was a young art form
- Master animator would draw important images ("keyframes") and junior animator would draw images to fill gaps between these key images



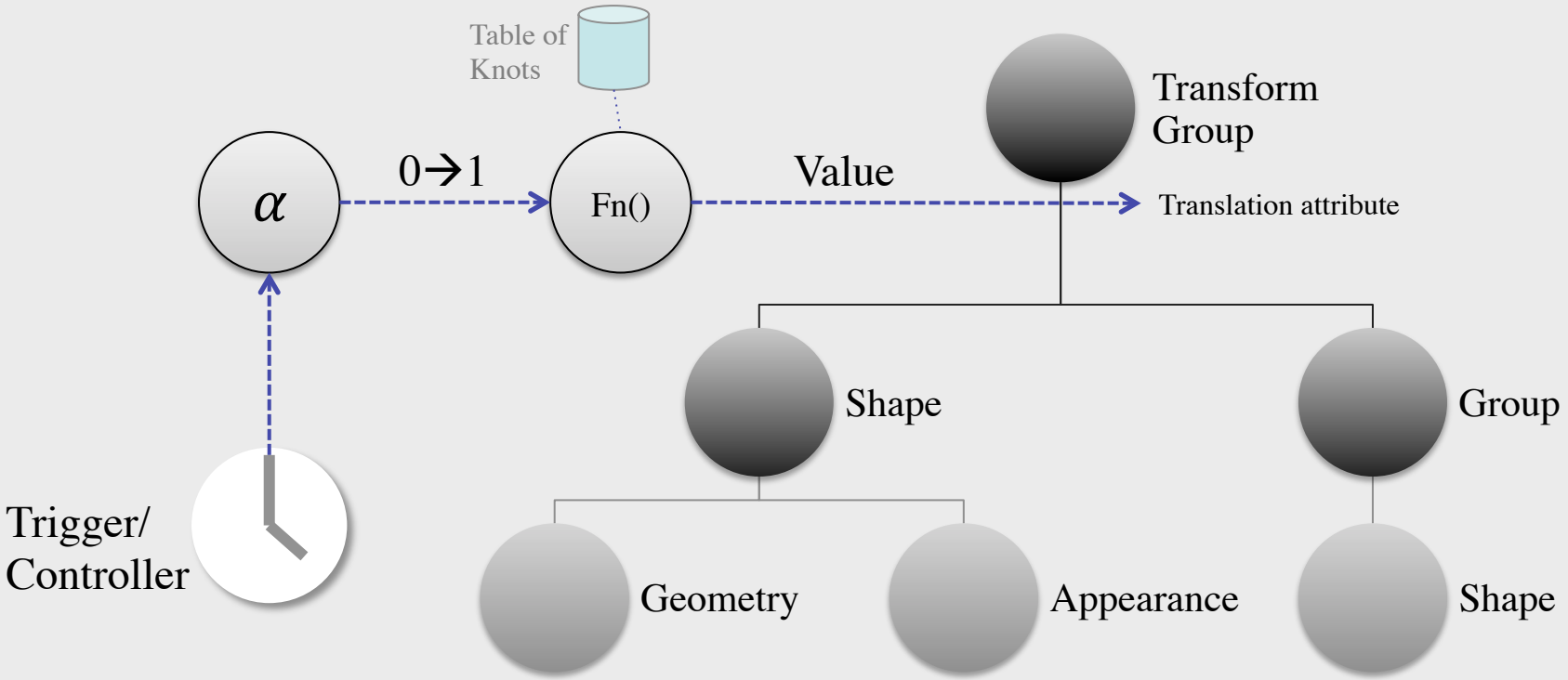
Keyframe Animation

- Objects in a scene graph are transformed relative to their parents
  - We can use this ability to animate a scene graph
- For each *keyframe*, can store corresponding values of a group object's transformation attributes
- Then, can calculate in-between transformation values required so that the object can be transformed smoothly between keyframes

Interpolators

- An *interpolator* is an object that varies the value of a parameter between a start value and an end value over time
- Rate at which variance takes place is controlled by an *Alpha* object
  - Alpha value is between 0 and 1
  - Given an alpha value, an interpolator object returns a corresponding parameter value
  - Smooth animation achieved using a timer/clock to generate alpha values, where the result of each tick is used to update nodes
- Interpolator data typically comprises of two or more values ("*knots*") indexed by a corresponding alpha-value

Interpolating Position Example



Interpolators

- Linear interpolation between two knot values is the simplest method of interpolation
- Can also use Bezier, B-spine, cardinal curves, etc.
  - Can give smooth interpolation with fewer knots than linear interpolation for complex paths
  - But need to be used with care to avoid problems such as *overshooting*
- Interpolators do not necessarily need to create in-between values at a linear speed
  - Sometimes it is desirable for the speed to vary, accelerating or decelerating towards a knot, for example

Hierarchical Animation

- Structure is crucial for animating complex objects
- To animate a hierarchy of objects, we can apply an interpolator to each node in the hierarchy that we want to animate
- Typically use one timer object per set of interpolators representing an animation sequence
- Conceptually simple
  - but it can quickly become complicated if there are lots of nodes that need to transformed

Forward Kinematics

- Most real-time 3D systems support *forward kinematics*
- Forward Kinematics:
  - Transformations propagate *down* through a hierarchy
    - When a group node is transformed then all of it's children are transformed too in absolute/global terms
      - Children's local/relative transformation attributes unchanged
    - If you transform a child group then the parent group is not affected by the change

Forward Kinematics

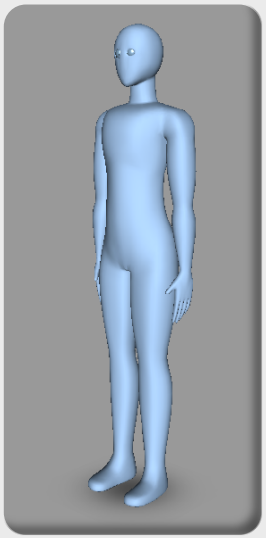
- Forward kinematics are often sufficient
  - Easy to implement
  - Conceptually simple
- But difficult to use in some cases
  - If hierarchy is deep then many objects need to be animated individually…
- Tip: Design & implement from top down
  - Animate objects at the top of the part of the scene graph that is to animated first and then animate child nodes to add detail

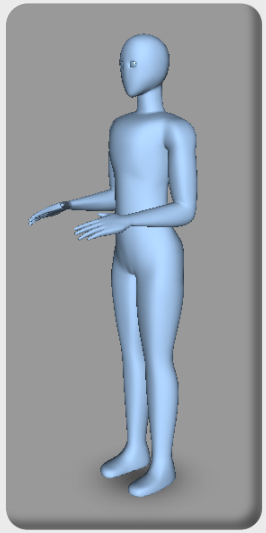
Inverse Kinematics

- Inverse kinematics (IK) enable us to do some things more easily than with forward kinematics
- But take longer to set up
- Inverse kinematics:
  - Transformations are propagated *up* a hierarchy
  - If a child object moves then the system attempts to determine the effect on of the movement on it's parent

Inverse Kinematics

- IK Example
  - If you raise the hand of a humanoid then the lower and upper arm should move too
    - Taking into account that the whole arm is fixed at the shoulder





Inverse Kinematics

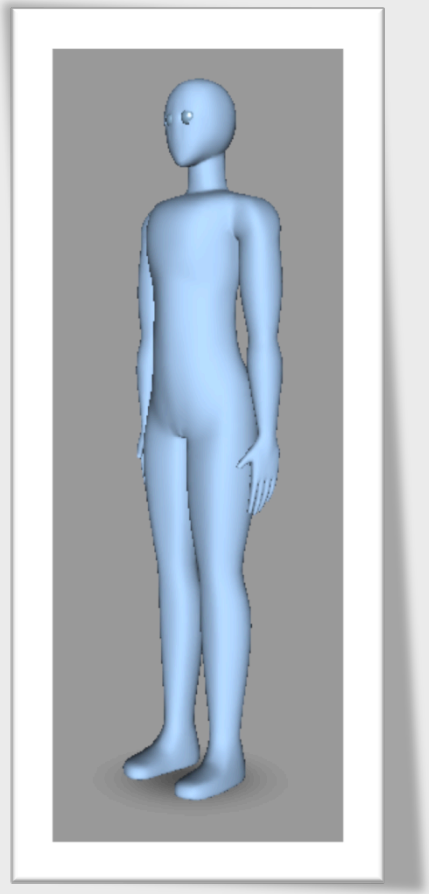
- Object hierarchy of IK model behaves like a chain
- IK hierarchy often called a *skeleton* and the joints *bones*
  - End of a branch of the hierarchy is called the *effector*
  - A fixed node (end of chain) higher up in the hierarchy is the *root*
  - Each bone has *constraints* that affect how it can be rotated if the effector moves
    - Some systems restrict rotations to one plane, since less mathematically complex and so more efficient to calculate in real time
- *"Rigging a mesh"* is task of setting up IK for a mesh
  - Mesh itself is wrapped around the skeleton and thus is deformed relative to the skeleton when the bones are later animated

IK for Animation

- In real-time 3D software systems, IK is mostly used to support dynamic simulation
  - Industrial robots, cranes, humanoids, …
- But, 3D animation packages typically support the use of IK to simplify the creation of certain types of animation (e.g. humans)
  - Using a modelling tool that supports IK, you can use IK to assist in producing the key knots needed to animate a hierarchy using FK
    - i.e. exporting the results as interpolator data for FK animation

Humanoid Animation

- Typically use motion capture data to animate the structure
  - e.g. by concatenating pre-recorded movements on-the-fly to give smooth human animation on-demand (e.g. for avatars)
- Requires that the hierarchical structure of the humanoid and naming scheme for nodes in the scene graph match the structure of the mocap data that is available
- Game engines generally provide a skeleton and a sub-system for handling animation



Motion Capture

- Can give very good results quickly compared to animating using 3D modelling and animation tools
- Set of sensors (or markers) typically used to register data
  - Registered data is then processed
    - Noise filtering important
    - Too few sensors, may provide too little data for good result
    - Too many sensors, may take too long to process
  - Processed data used to transform specific nodes in an animation hierarchy to animate the hierarchy (FK/IK) in 3D modelling tools
- BIP or BVH-format mocap data files increasingly available over the internet (commercially or free, e.g. mocapdata.com)

Motion Dynamics

- Physics-based simulation techniques can be used to pre-compute realistic animation paths and behaviour for playback using interpolators later
  - Take into account mass, density, friction, gravity, etc.
- When we do this on-the-fly, we call it simulation rather than animation
  - Tom-Robert will show some examples of this in the next lecture, where we will demonstrate how to use motion dynamics to simulate physics in real time

Implementation Strategies

- For in real-time virtual environments we typically use a combination of strategies to animate the response to user input
  - Predefined ("traditional") keyframe animation
    - Use 3D modelling and animation tools to animate geometry and then import the animation into run-time environment
    - *load then play*
  - Generated at run-time keyframe animation
    - Construct an animation at run-time from immediately before using them, e.g. to tell an object to move from position *a* to position *b* on demand
    - *generate then play*
  - Dynamic animation ("simulation")
    - Animate by generating values on the fly to animate objects based on (for example) motion dynamics equations
    - *calculate while playing*

Camera Animation

- Can animate movement of a camera/view by using interpolators to specify a camera motion path
- Use both position and orientation interpolators
  - Adjust the orientation so the camera follows the direction of a path of positions
  - Tilt the camera as it turns (banking effect)
- Useful effects can be created by animating other properties of the camera too, such as field of view

Animating Lights

- Animated lights can be used to achieve special effects by varying position, intensity, colour, and other properties
- Lots of possibilities — not all of which are obvious!
  - Dimming lights to “fade” a scene away before replacing it and then increasing light intensity to show the new scene
  - Changing light colours to alter mood or time of day
  - Using a spotlight to highlight an object
  - “Flickering” light to simulate fire/candle lighting

Animating Materials

- Can use interpolators to animate colours, transparency, texture settings, and other appearance attributes
- For example:
  - Use to animate an object fading away, getting hot or cold, or the sky getting dark
  - Texture animations (e.g. in combination with geometry deformation) can be used to animate water surfaces

Geometry Deformation

- Interpolate geometry data values to deform objects by dynamically modifying geometry
  - Keyframes are target shapes
  - Amount of geometry data remains constant (but can use multiple steps to increase/decrease this if necessary)
  - e.g. waves on water or an object getting crushed
- Simple alternative: Use a scale interpolator to animate deformation of an object
  - E.g. a ball hitting the ground and bouncing off

Animation Theory

- Animating things (creatures or objects) in a communicative or realistic manner is challenging!
- How can you convey the weight and size of an object?
- Is the object soft or hard?
  - Soft objects distort when they bounds, whereas hard objects distort the objects they hit or may simply vibrate
- Timing is very important when telling a story
  - Viewer should anticipate actions, view them, and react to them
  - But not too quickly (confusing) or too slow (boring)…
- We can apply some tricks from movie animators!

Animation Theory

- Attraction Tricks
  - Attract the users attention so that he/she does not miss something important!
  - Give the user a hint that something is about to happen and where
  - For example, before an action takes place the object that that will perform the action can attract attention by
    - blinking
    - rocking back and forth
    - backing up a little

Animation Theory

- Timing Tricks
  - It is sometimes useful to use unrealistic timing to achieve a desired communicative effect
  - For example
    - Slow the speed of animation up or down
      - E.g. if the real action would be too fast or slow to view effectively
    - Use a logarithmic scale for timing rather than linear
      - E.g if flying towards objects that are vastly different in size

Animation Theory

- Motion enhancement tricks
  - Sometimes it is necessary to make it more obvious that a motion is taking place
  - For example, a rolling ball may need some markings on it so that the viewer can clearly see that is is rotating

Contributing to Sense of Presence

- Virtual environments seem more alive and believable if several subtle things are happening simultaneously
- A stationary humanoid will look more alive than a statue if it does one or more of the following
  - breath
  - blink
  - look around
- Clouds, weather, birds, etc. enhance outdoor scenes
- Audio sources can also play a useful part in an animation by moving with object that represent audio emitters

Finally…

- Consider using storyboards to plan any complex animations before attempting to implement them
  - Ensure that your geometry hierarchy can support the animations you want to implement
- Animation is useful not only for implementing the behaviour of a virtual environment but also to enhance the user interface for interacting with it