Lecture 18:

Intro to Animation

Computer Graphics and Imaging
UC Berkeley CS184/284A
Principles of Animation
Rigging & Skinning
Motion Capture
Physical Simulation: Cloth
Animation

“Bring things to life”

• Communication tool

• Aesthetic issues often dominate technical issues

An extension of modeling

• Represent scene models as a function of space

Output: sequence of images that when viewed sequentially provide a sense of motion

• Film: 24 frames per second

• Video: 30 fps

• Virtual reality: 90 fps
Historical Points in Animation
(slides courtesy Keenan Crane)
First Animation

(Shahr-e Sukhteh, Iran 3200 BCE)
History of Animation

(tomb of Khnumhotep, Egypt 2400 BCE)
History of Animation

(Phenakistoscope, 1831)
First Film

Originally used as scientific tool rather than for entertainment

Critical technology that accelerated development of animation

Edward Muybridge, “Sallie Gardner” (1878)
First Hand-Drawn Feature-Length Animation

Disney, “Snow White and the Seven Dwarfs” (1937)
First Digital-Computer-Generated Animation

Ivan Sutherland, “Sketchpad” (1963) – Light pen, vector display
Early Computer Animation

Nikolay Konstantinov, "Kitty" (1968)
Early Computer Animation

Ed Catmull & Frederick Parke, “Computer Animated Faces” (1972)
Digital Dinosaurs!

Jurassic Park (1993)
First CG Feature Film

Computer Animation - Present Day

Disney/Pixar Soul (2020)
Animation Principles
(slides courtesy Mark Pauly)
Animation Principles

From


In turn from

- “The Illusion of Life”
  Frank Thomas and Ollie Johnston

Same for 2D and 3D

http://www.siggraph.org/education/materials/HyperGraph/animation/character_animation/principles/prin_trad_anim.htm
Squash and Stretch

Refers to defining the rigidity and mass of an object by distorting its shape during an action.

Shape of object changes during movement, but not its volume.
Anticipation

Prepare for each movement
For physical realism
To direct audience’s attention
Staging

Picture is 2D
Make situation clear
Audience looking in right place
Action clear in silhouette

Disney Animation: The Illusion of Life
Follow Through

Overlapping motion

Motion doesn’t stop suddenly

Pieces continue at different rates

One motion starts while previous is finishing, keeps animation smooth
Ease-In and Ease-Out

Movement doesn’t start & stop abruptly.
Also contributes to weight and emotion
Arrows

Move in curves, not in straight lines

This is how living creatures move

Disney Animation: The Illusion of Life
Secondary Action

Motion that results from some other action

Needed for interest and realism

Shouldn’t distract from primary motion

Cartoon Animation, Preston Blair
Timing

Rate of acceleration conveys weight

Speed and acceleration of character’s movements convey emotion

Timing for Animation, Whitaker & Halas
Exaggeration

Helps make actions clear
Helps emphasize story points and emotion
Must balance with non-exaggerated parts

Timing for Animation, Whitaker & Halas
Appeal

Attractive to the eye, strong design

Avoid symmetries

Disney Animation: The Illusion of Life
Personality

Action of character is result of its thoughts
Know purpose & mood before animating each action
No two characters move the same way
Further Reading
12 Animation Principles

1. Squash and stretch
2. Anticipation
3. Staging
4. Straight ahead and pose-to-pose
5. Follow through
6. Ease-in and ease-out
7. Arcs
8. Secondary action
9. Timing
10. Exaggeration
11. Solid drawings
12. Appeal
12 Animation Principles

THE ILLUSION OF LIFE

Cento Lodgiani, https://vimeo.com/93206523
12 Animation Principles

Applications:
- Movies
- Games
- User interfaces
- ...

CS184/284A
Ren Ng
Computer Animation
Keyframe Animation

Keyframes

"Tweens"

Animator (e.g. lead animator) creates keyframes
Assistant (person or computer) creates in-between frames ("tweening")
Keyframe Interpolation

Think of each frame as a vector of parameter values
Keyframe Interpolation of Each Parameter

Linear interpolation usually not good enough

Recall splines for smooth / controllable interpolation
Forward Kinematics
Forward Kinematics
Recall this skeleton from Transforms lecture

torso
  head
  right arm
    upper arm
    lower arm
    hand
  left arm
    upper arm
    lower arm
    hand
right leg
  upper leg
  lower leg
  foot
left leg
  upper leg
  lower leg
  foot
translate(0, 10);
drawTorso();

pushmatrix(); // push a copy of transform onto stack
translate(0, 5); // right-multiply onto current transform
rotate(headRotation); // right-multiply onto current transform
drawHead();
popmatrix(); // pop current transform off stack
pushmatrix();

translate(-2, 3);
rotate(rightShoulderRotation);
drawUpperArm();
pushmatrix();

translate(0, -3);
rotate(elbowRotation);
drawLowerArm();
pushmatrix();

translate(0, -3);
rotate(wristRotation);
drawHand();
popmatrix();

popmatrix();
popmatrix();
Forward Kinematics

Articulated skeleton

• Topology (what’s connected to what)
• Geometric relations from joints
• Tree structure (in absence of loops)

Joint types

• Pin (1D rotation)
• Ball (2D rotation)
• Prismatic joint (translation)
Forward Kinematics

Example: simple two segment arm in 2D
Forward Kinematics

Animator provides angles, and computer determines position $p$ of end-effector.

$$p_z = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$
$$p_x = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2)$$

Warning: Z-up Coordinate System
Forward Kinematics

Animation is described as angle parameter values as a function of time

$$p_z = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)$$

$$p_x = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2)$$
Example Walk Cycle
Inverse Kinematics
Inverse Kinematics

Given the end effector position, find the joint angles.

Goals

• Keep end of limb fixed while body moves
• Position end of limb by direct manipulation
• (More general: arbitrary constraints)
Inverse Kinematics

Egon Pasztor
Inverse Kinematics

Animator provides position of end-effector, and computer must determine joint angles that satisfy constraints
Inverse Kinematics

Direct inverse kinematics: for two-segment arm, can solve for parameters analytically

\[
\theta_2 = \cos^{-1}\left(\frac{p_z^2 + p_x^2 - l_1^2 - l_2^2}{2l_1l_2}\right)
\]

\[
\theta_1 = \frac{-p_z l_2 \sin(\theta_2) + p_x (l_1 + l_2 \cos(\theta_2))}{p_x l_2 \sin(\theta_2) + p_z (l_1 + l_2 \cos(\theta_2))}
\]
Inverse Kinematics

Why is the problem hard?

- Multiple solutions separated in configuration space
Inverse Kinematics

Why is the problem hard?

- Multiple solutions connected in configuration space
Inverse Kinematics

Why is the problem hard?

- Solutions may not always exist
Inverse Kinematics

Numerical solution to general N-link IK problem

• Choose an initial configuration
• Define an error metric (e.g. square of distance between goal and current position)
• Compute gradient of error as function of configuration
• Apply gradient descent (or Newton’s method, or other optimization procedure)
Style-Based IK

Grochow et al., Style Based Inverse Kinematics
Kinematics Pros and Cons

Strengths

• Direct control is convenient
• Implementation is straightforward

Weaknesses

• Animation may be inconsistent with physics
• Time consuming for artists
Skinning
Skinning

Goal: move the surface along with assigned bones or "handles"
Basic Idea

1. Transform each vertex with each bone rigidly
2. Blend the results using weights, or assignments
Common Approach: Linear Blend Skinning (LBS)

Blend contribution linearly.
Super simple to implement. Great for real time.

How much influence this bone has on \( v \) (often sparse)

\[
\mathbf{v}' = \sum_{j \in H} w_j(v) T_j \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix}
\]

New vertex

Bone \( j \) transformation

Original vertex

Ren Ng
Illustration of Rig & Skinning Weights

Bone transformations

Skinning Weights
Blend Shapes
Blend Shapes

Not all deformation is from bones.
Interpolate surfaces between key shapes.
Blend Shapes

• A set of vertex offsets to neutral shape
• Linearly interpolate these key blend shapes for control
• Often used for expressions
• Works for deformations that are linear, i.e. the average of two shapes is a valid shape

\[ B = \text{vec}( \begin{bmatrix} \Delta x_1 & \Delta y_1 & \Delta z_1 \\ . & . & . \\ \Delta x_N & \Delta y_N & \Delta z_N \end{bmatrix} ) \]

\[ V = \sum_i \beta_i B_i \]
Blend Shapes

Courtesy Félix Ferrand
Rigging
Rigging

Augment character with controls to easily change its pose, create facial expressions, bulge muscles, etc.

Rigging is like the strings on a marionette.
Capture space of meaningful deformations.
Varies from character to character.

Skeleton is ONE type of rigging
Example of A Diverse Set of Sophisticated Rigs

Courtesy of Matthew Lailler
Motion Capture
Motion Capture

Data-driven approach to creating animation sequences

- Record real-world performances
- Extract pose as a function of time from raw data
Motion Capture Equipment

**Optical**
(More on following slides)

**Magnetic**
Sense magnetic fields to infer position / orientation.
Tethered.

**Mechanical**
Measure joint angles directly.
Restricts motion.
Optical Motion Capture

- Markers on subject
- Positions by triangulation from multiple cameras
- 8+ cameras, 240 Hz, occlusions are difficult

Slide credit: Steve Marschner
Motion Capture
Motion Data

Subset of motion curves from captured walking motion.

From Witkin and Popovic, 1995
Motion Capture Pros and Cons

Strengths

- Can capture large amounts of real data quickly
- Realism can be high

Weaknesses

- Complex and costly set-ups
- Captured animation may not meet artistic needs, requiring alterations
Markerless Motion Capture

Kanazawa et al. 2018

Kanazawa et al. 2019
Acknowledgments

Art Competition #2 Results
Art Competition #2 – 3rd Place Winner

Capybaras
Raine Koizumi & Arjun Palkhade

Caption: capybaras sitting in a hot spring, zero feet apart cuz they all happy :D

I shaped out cubes for the capybara, and icosahedrons for the rocks, and a plane for the water. Additionally, I made a fragment shader that represents the normal of each vertex as RGB.

1.5 hours
Minuteman
Olivia Xie

This is a character that I had modeled for a multiplayer game I helped create back when I took the Game Design & Development DeCal.
Here's a link to the game: https://minutemen.itch.io/rotor
(multiplayer does not work in this version)
Modeled everything in Autodesk Maya, 4 hours.
Rainbow Toon-shaded Catfish Fakemon!
Rebecca Feng & Mahum Khan

The Fakemon was box-modeled in Autodesk Maya, exported as a DAE file, and rendered in the viewer. We made the background change color by keeping a time variable, and using the glClearColor method. In order to create the toon-shader, we took the default Phong shader and made it a peach color if outputted color by the Phong shader was above a certain brightness, and a blue color if it was less than that. Additionally, we added a rainbow-effect by assigning an RGB value as its position times a small factor of 0.05.

~20 hours for the model, ~4 hours for the shader