PA199
Advanced Game Design

Lecture 12

Character Modeling

Dr. Fotis Liarokapis

18th May 2017

3D Animation

- Rendering
  - 3D Scene and Motion
  - Sequence of Frames
    - Rates: Video 30fps, Film 24fps
  - Persistence of Vision
- Animator must create
  - Illusion of Life
  - Weight

Animation

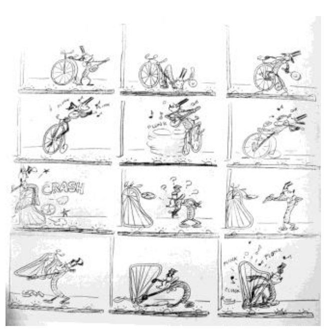
- Almost every property of every object in the scene can be animated changed through time
  - Models, cameras
  - Transformations (Move, Rotate, Scale)
- Modifications/Deformation:
  - Edits, bends, twists, manipulating a skeleton
- Materials, colors, textures

Animation .

- 3D Scene does not have
  - Gravity
  - Weight
  - Force
  - Complete interactions between objects
    - Sometimes it has…
- You must make it seem so

Preproduction Phases

- Screen-play
- Storyboards
- Character development



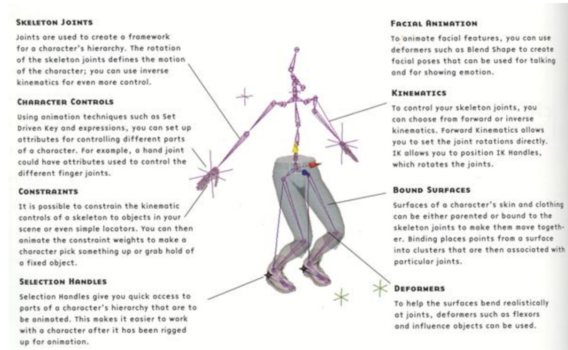
3D Characters

- Digital actor
  - Tin can
  - Sack of flower
  - Butterfly, beetle
  - Bird
  - Flower
  - Robot
  - Humanoid
  - Etc…

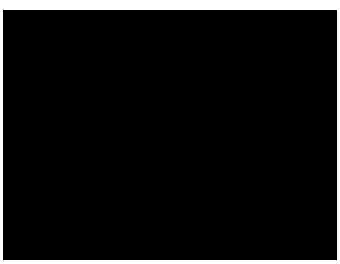
Typical Character

- Mechanics of movement must be convincing
- Skin and clothing moves & bends appropriately
- This process of preparing character controls is called rigging (see next slides)
  - Fully rigged character has:
    - Skeleton joints, surfaces, deformers, expressions, Set Driven Key, constraints, etc

Typical Character .



Facial Animation Video



https://www.youtube.com/watch?v=z86YsS-pVsQ

Character Resolution

- Use low resolution character that has surfaces 'parented' to skeleton
  - Allows interactive animations
  - Switch to full resolution character later

Typical Character Animation Workflow

- Character Design
- Model
- Skeleton Rigging
- Binding
- Animation
- Integration
- Rendering

Rigging

- Rigging refers to the construction and setup of an animatable character
  - Similar to the idea of building a puppet
- A 'rig' has numerous degrees of freedom (DOFs) that can be used to control various properties

Primary Methods of Animation

- Keyframe
- Procedural
  - Expressions
  - Scripting
- Dynamics/Simulation
  - Physics
- Motion Capture
- Combinations of the above

Keyframe Workflow

- Set Keys
  - Usually extreme positions
  - Less is more: Keys only the properties being animated
- Set Interpolation
  - Specify how to get from one key to another
  - Secondary, but a necessary step
- Scrub Time slider and refine motion curve

Setting Keys

- Start with extreme positions
- Add intermediate positions
  - Secondary motion
- Less is more
  - Don't add keys for properties that you are not animating
  - Easier to manage/edit fewer keys

Skeletal Hierarchy

- The Skeleton is a tree of bones
  - Often flattened to an array in practice
- Top bone in tree is the "root bone"
  - May have multiple trees, so multiple roots
- Each bone has a transform
  - Stored relative to its parent's transform
- Transforms are animated over time
- Tree structure is often called a "rig"

Bone Masks

- Some animations only affect some bones
  - Wave animation only affects arm
  - Walk affects legs strongly, arms weakly
    - Arms swing unless waving or holding something
- Bone mask stores weight for each bone
  - Multiplied by animation's overall weight
  - Each bone has a different effective weight
  - Each bone must be blended separately
- Bone weights are usually static
  - Overall weight changes as character changes animations

Variable Delta Extraction

- Uses root bone motion directly
- Sample root bone motion each frame
- Find delta from last frame
- Apply to instance pos+orn
  - Instance pos+orn is the root bone!
- Root bone is ignored when rendering

Animation Storage Problem

- 4x3 matrices, 60 per second is huge
  - 200 bone character = 0.5Mb/sec
- Consoles have around 32-64Mb
- Animation system gets maybe 25%
- PC has more memory
  - But also higher quality requirements

Decomposition

- Decompose 4x3 into components
  - Translation (3 values)
  - Rotation (4 values - quaternion)
  - Scale (3 values)
  - Skew (3 values)
- Most bones never scale & shear
- Many only have constant translation
- Don't store constant values every frame

Interpolation

- Specify how to get from one key to the other (in between)
- Common types
  - Step: stay at the same value, then suddenly switch
  - Linear: change at constant rate
  - Spline/Smooth: make it smooth
- All of these (and more) are useful and appropriate in the right circumstance

Mesh Deformation

- Find Bones in World Space
- Find Delta from Rest Pose
- Deform Vertex Positions
- Deform Vertex Normals

Find Bones in World Space

- Animation generates a "local pose"
  - Hierarchy of bones
  - Each relative to immediate parent
- Start at root
- Transform each bone by parent bone's world-space transform
- Descend tree recursively
- Now all bones have transforms in world space
  - "World pose"

Find Delta from Rest Pose

- Mesh is created in a pose
  - Called the "rest pose"
- Must un-transform by that pose first
- Then transform by new pose
  - Multiply new pose transforms by inverse of rest pose transforms
  - Inverse of rest pose calculated at mesh load time
- Gives "delta" transform for each bone

Deform Vertex Positions

- Deformation usually performed on GPU
- Delta transforms fed to GPU
  - Usually stored in "constant" space
- Vertices each have n bones
- n is usually 4
  - 4 bone indices
  - 4 bone weights 0-1
  - Weights must sum to 1

Deform Vertex Normals

- Normals are done similarly to positions but use inverse transpose of delta transforms
  - Translations are ignored
  - For pure rotations, inverse(A)=transpose(A)
  - So inverse(transpose(A)) = A
  - For scale or shear, they are different
- Normals can use fewer bones per vertex
  - Just one or two is common

Motion of characters

- Along with key frame animation we can use kinematics
  - Kinematics = study of motion without regard to the forces that cause it

Forward: $A = f(\alpha,\beta)$ **Inverse:** $\alpha,\beta = f^{-1}(A)$

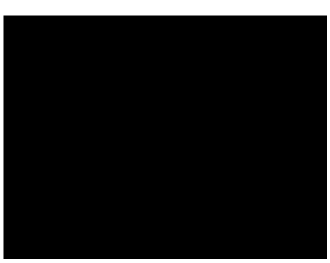
Specify fewer degrees of freedom

More intuitive control

FK & IK

- Most animation is "forward kinematics"
  - Motion moves down skeletal hierarchy
- But there are feedback mechanisms
  - Eyes track a fixed object while body moves
  - Foot stays still on ground while walking
  - Hand picks up cup from table
- This is "inverse kinematics"
  - Motion moves back up skeletal hierarchy

Skeleton Puppet Game Video (FK & IK)



https://www.youtube.com/watch?v=QDwo9d8Fa5M

User Control of Kinematic Characters

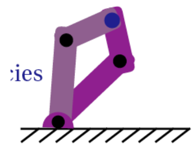
- Joint Space
  - Position all joints
    - Fine level of control
- Cartesian Space
  - Specify environmental interactions easily
    - Most DOF computed automatically

Inverse Kinematics

- Balance = keep center-of-mass over support polygon
- Control
  - i.e. position vaulter's hands on line between shoulder and vault
  - i.e. Compute knee angles that will give runner the right leg length

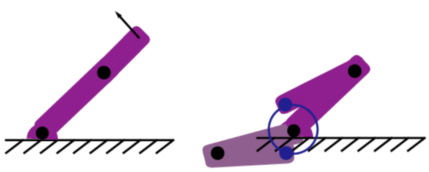

Inverse Kinematics is Hard

- Redundancy

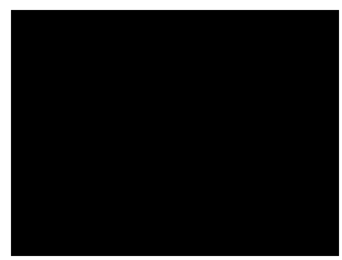


Inverse Kinematics is Hard .

- Singularities



Momentum-based Inverse Kinematics with Motion Capture Video



https://www.youtube.com/watch?v=FJTBMnP6oCM

Single Bone IK

- Orient a bone in given direction
  - Eyeballs
  - Cameras
- Find desired aim vector
- Find current aim vector
- Find rotation from one to the other
  - Cross-product gives axis
  - Dot-product gives angle
- Transform object by that rotation

Multi-Bone IK

- One bone must get to a target position
  - Bone is called the "end effector"
- Can move some or all of its parents
- May be told which it should move first
  - Move elbow before moving shoulders
- May be given joint constraints
  - Cannot bend elbow backwards

Cyclic Coordinate Descent

- Simple type of multi-bone IK
- Iterative
  - Can be slow
- May not find best solution
  - May not find any solution in complex cases
- But it is simple and versatile
  - No pre-calculation or pre-processing needed

Cyclic Coordinate Descent .

- Start at end effector
- Go up skeleton to next joint
- Move (usually rotate) joint to minimize distance between end effector and target
- Continue up skeleton one joint at a time
- If at root bone, start at end effector again
- Stop when end effector is “close enough”
- Or hit iteration count limit

Cyclic Coordinate Descent ..

- May take a lot of iterations
- Especially when joints are nearly straight and solution needs them bent
  - e.g. a walking leg bending to go up a step
  - 50 iterations is not uncommon!
- May not find the “right” answer
  - Knee can try to bend in strange directions

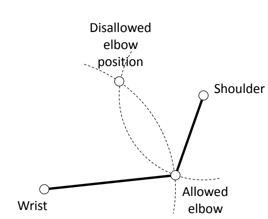
Two-Bone IK

- Direct method, not iterative
- Always finds correct solution
  - If one exists
- Allows simple constraints
  - Knees, elbows
- Restricted to two rigid bones with a rotation joint between them
  - Knees, elbows!
- Can be used in a cyclic coordinate descent

Two-Bone IK .

- Three joints must stay in user-specified plane
  - e.g. knee may not move sideways
- Reduces 3D problem to a 2D one
- Both bones must remain same length
- Therefore, middle joint is at intersection of two circles
- Pick nearest solution to current pose
- Or one solution is disallowed
  - Knees or elbows cannot bend backwards

Two-Bone IK ..



IK by Interpolation

- Animator supplies multiple poses
- Each pose has a reference direction
  - e.g. direction of aim of gun
- Game has a direction to aim in
- Blend poses together to achieve it
- Source poses can be realistic
  - As long as interpolation makes sense
  - Result looks far better than algorithmic IK with simple joint limits

IK by Interpolation .

- Result aim point is inexact
  - Blending two poses on complex skeletons does not give linear blend result
- Can iterate towards correct aim
- Can tweak aim with algorithmic IK
  - But then need to fix up hands, eyes, head
  - Can get rifle moving through body

Attachments

- For example character holding a gun
- Gun is a separate mesh
- Attachment is bone in character's skeleton
  - Represents root bone of gun
- Animate character
- Transform attachment bone to world space
  - Move gun mesh to that pos+orn

Attachments .

- For example person is hanging off bridge
- Attachment point is a bone in hand
  - As with the gun example
- But here the person moves, not the bridge
- Find delta from root bone to attachment bone
- Find world transform of grip point on bridge
- Multiply by inverse of delta
  - Finds position of root to keep hand gripping

Collision Detection

- Most games just use bounding volume
- Some need perfect triangle collision
  - Slow to test every triangle every frame
- Pre-calculate bounding box of each bone
  - Transform by world pose transform
  - Finds world-space bounding box
- Test to see if bounding box was hit
  - If it did, test the this bone influences

NCCA Video 1

NCCA Video 2

Our Work

Questions