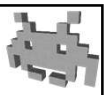
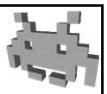
Course Plan



lec. 1: Introduction
lec. 2: Mathematics for 3D Games
lec. 3: Scene Graph
lec. 4: Game 3D Physics +
lec. 5: Game Particle Systems
lec. 6: Game 3D Models
lec. 7: Game Textures
lec. 8: Game 3D Animations
lec. 9: Game 3D Audio
lec. 10: Networking for 3D Games
lec. 11: Artificial Intelligence for 3D Games
lec. 12: Game 3D Rendering Techniques

120

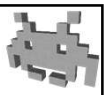
Mixing keyframes and entire animations (notes)



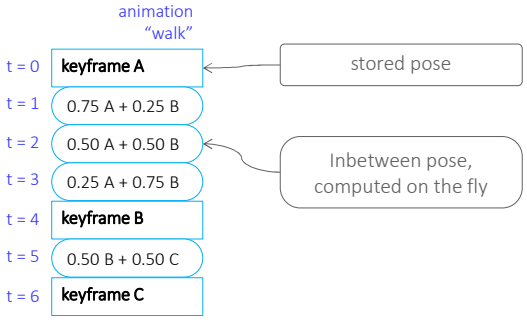
- Poses in a skeletal animations can be easily blended (blending local per-bone transforms)
  - This interpolation is very expressive: very different frames can be blended with good results
  - much more than with blend-shapes!
  - Keyframes can be very far apart
  - E.g.: decent walk-cycles with just 4 key-frames! (2 per step)
  - E.g.: decent attack animations with just 2 key-frames!
  - (but better results are always obtained inserting new key-frames)
- Entire animations can be mixed. Two ways:
  - Transitions between two animations (or more)
  - Compositing (layering) two animations (or more)

121

Pose = keyframe

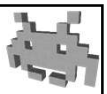


- Compress animations

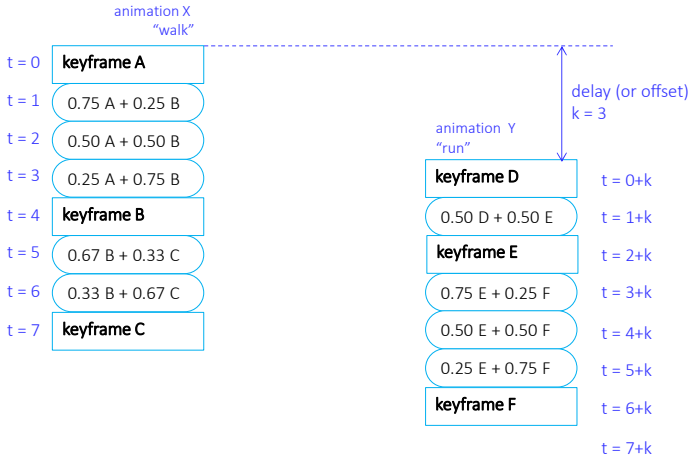


122

Interpolation of poses (at runtime):
transition between animations

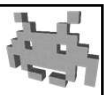


- Eg: from *stance* to *run*

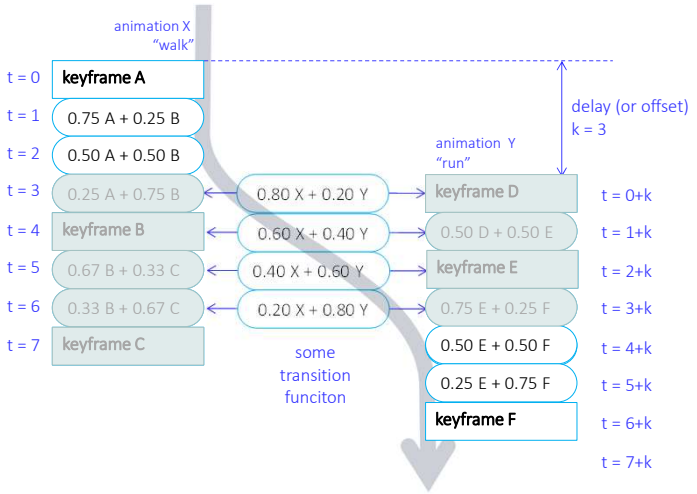


123

Interpolation of poses (at runtime):
transition between animations

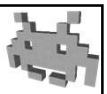


- Eg: from *stance* to *run*



124

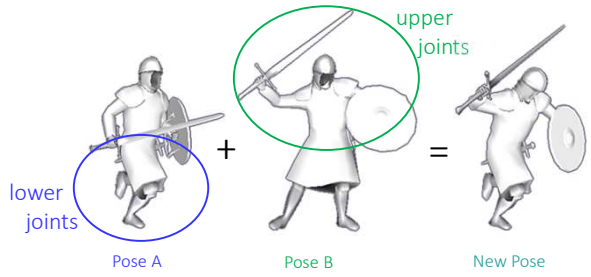
Interpolation of poses (at runtime):
transition between animations (notes)

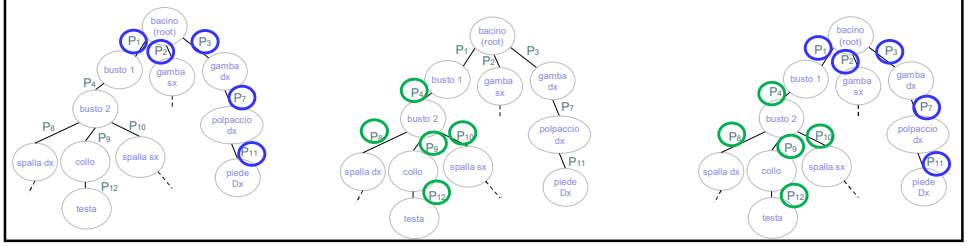


- Transitions between animations
  - E.g., from stance (aka "idle") animation to walk
  - E.g., from attack to parry
  - Most typical example:
    from a gait (run, trot, walk) to another
- The quality of the resulting transition varies
  - E.g., on how similar the two interpolated frames are (e.g.: do they both have the left leg in front?)
  - Determining the good offsets / speed / transition function can be a matter of trial and error
  - E.g.: Unity has a WYSIWYG interface for that!

125

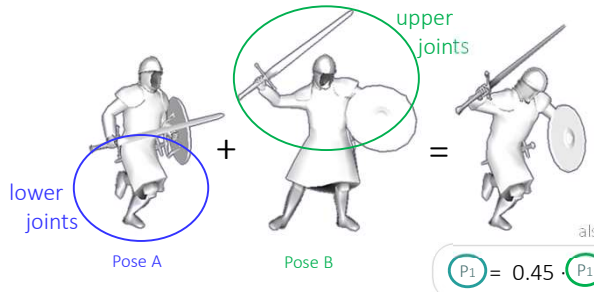
Compositing (layering) poses
(➔ and animations)





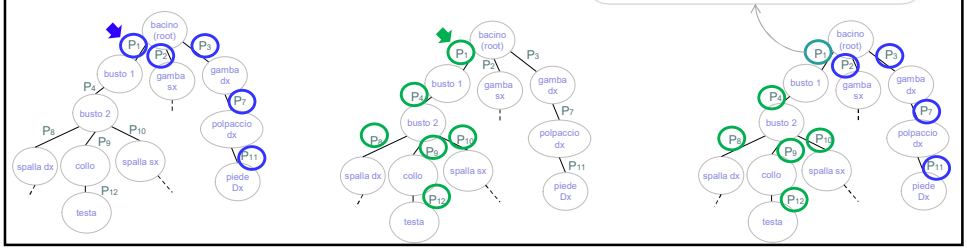
126

Compositing (layering) poses
(➔ and animations)



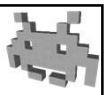
also, interpolating, e.g.:

$P_1 = 0.45 \cdot P_1 + 0.55 \cdot P_1$



127

Compositing (layering) poses
(notes)

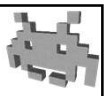


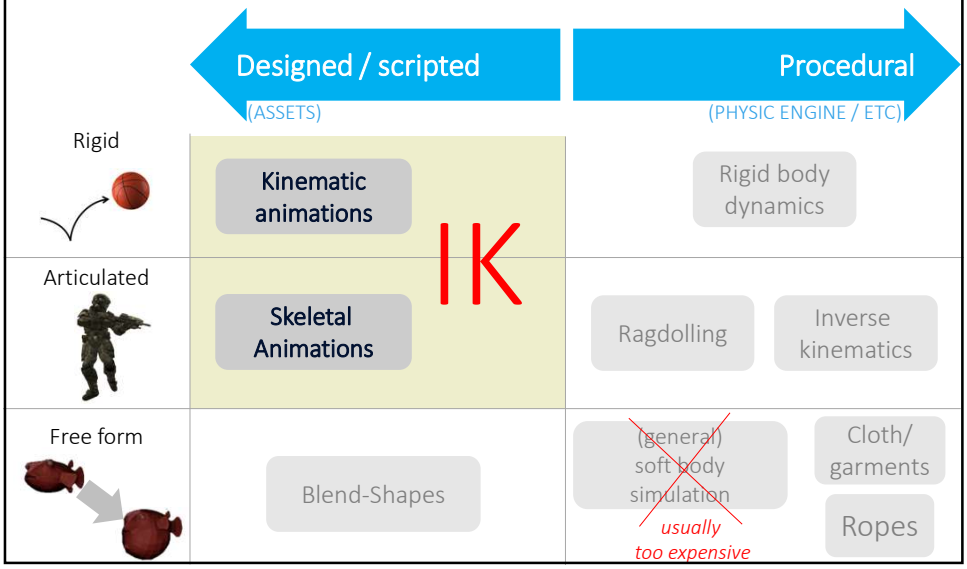
- Useful in different contexts:
  - e.g., different character parts following different anims
  - e.g., lower body: run. Upper body: aims/shoots/reload
- Note:
  local transformations are mixed (as usual).
  Final transformations need be updated after mixing
  - (after changing the local ones)
- Unity has an interface for this:
  - Per-bone Layer = a mask of per-bone Booleans:
    is the local animation of this bones "overwritten" by this animation?

128

Inverse Kinematics (notes)

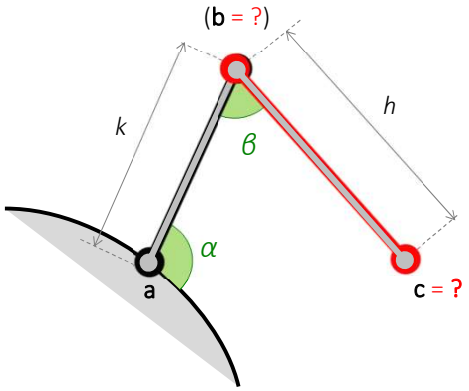
to learn more on IK, see course on → Virtual Reality





129

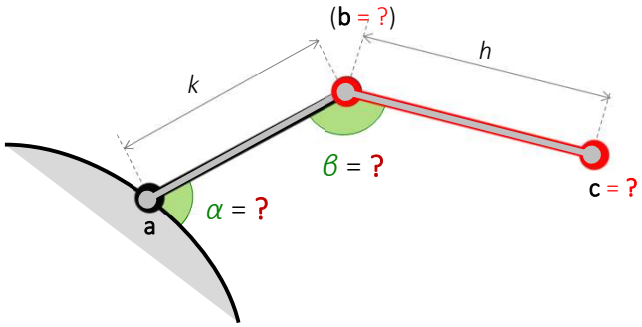
Forward Kinematic



find **c** (and **b**), given **a**, α, β, k, h

131

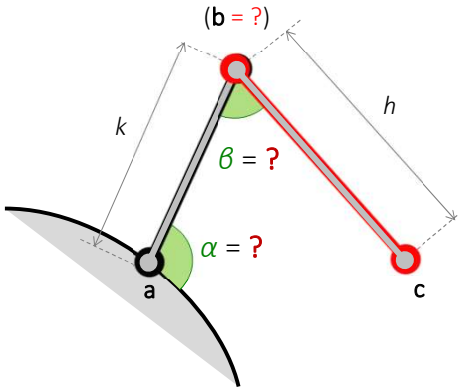
Forward Kinematic



find **c** (and **b**), given **a**, α, β, k, h

133

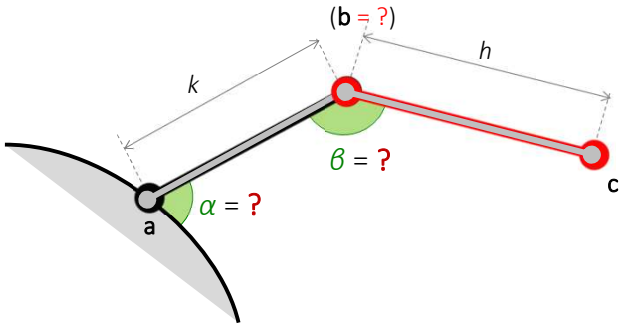
Inverse Kinematic (IK)



find α, β (and **b**), given **a**, **c**, k, h

134

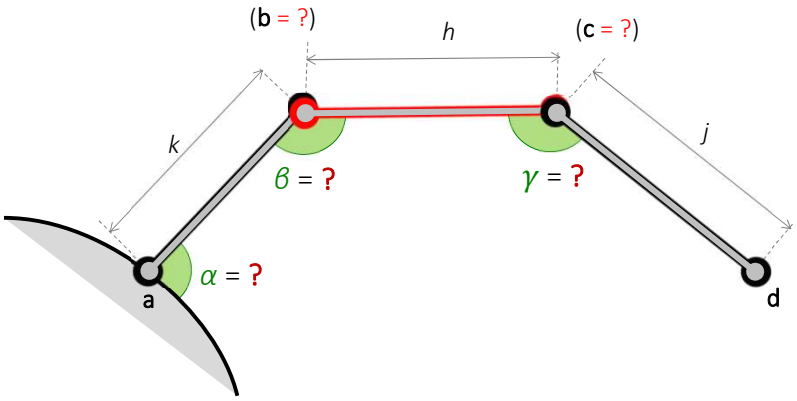
Inverse Kinematic (IK)



find α, β (and **b**), given **a**, **c**, k, h

136

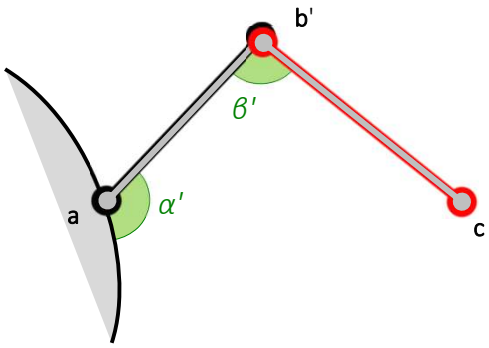
Inverse Kinematic (IK):
for any number of joints



solve for α , β , γ (and **b**, **c**), given **a**, **d**, k, h, j

137

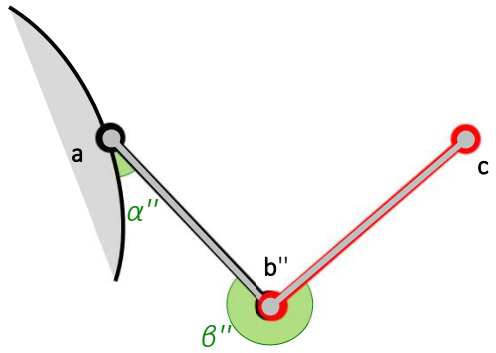
Inverse Kinematic (IK):
an ambiguity



solve for α, β (and **b**), given **a**, **c**, k, h

138

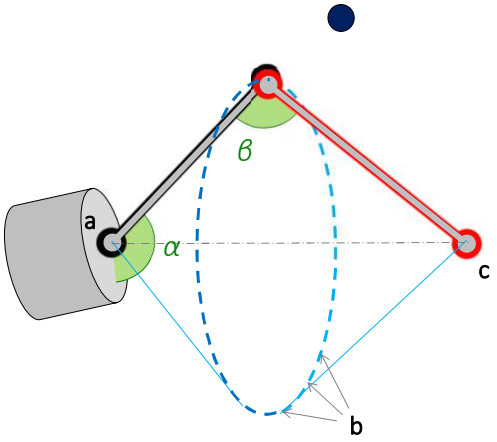
Inverse Kinematic (IK): an ambiguity



solve for α, β (and **b**), given **a**, **c**, *k*, *h*

139

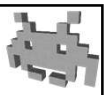
Inverse Kinematic (IK) in 3D: more ambiguities



solve for α, β (and **b**), given **a**, **c**, *k*, *h*

140

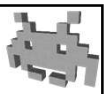
Inverse Kinematic (IK): useful in editing, and at run-time



- Direct kinematics
  - Single solution exists
  - An example is going from Local Transform to Final Transform
- Inverse Kinematics
  - it's more difficult to solve (as it's often the case with inverse problems!)
  - Often, trivial solutions are all that we need in Games: e.g., just two bones (for articulated legs, or arms)
  - Multiple solutions exists: which one to pick?
  - Disambiguate with additional constraint, such as: minimize the distance from the intermediate bone to a given attractor position

141

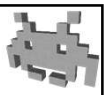
Inverse Kinematic (IK): useful in editing, and at run-time



- Many uses:
  - in preprocessing (helping the task of the animator)
  - in real time (performed by the game engine)
- Examples of real-time uses:
  - Exact positioning of feet on ground
  - Exact positioning of hand to object to be grabbed
  - Hands need to be joined (e.g. 2-handed weapon wielding)
    - (e.g., making the system auto-correct for small changes in bone lengths – helps animation retargeting)
    - (e.g., during interpolated keyframes)
  - Helps attack animation "connect" with target

143

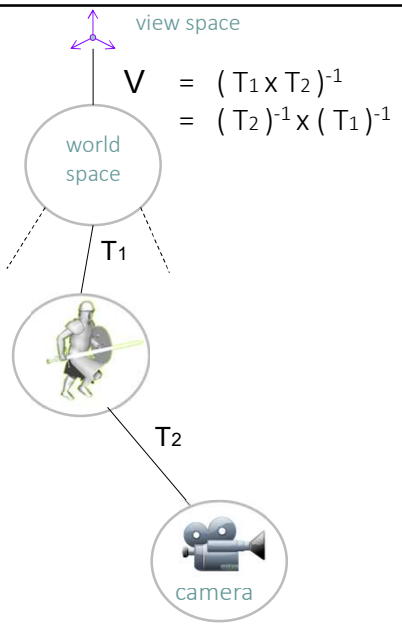
Skeletons as part of the scenegraph (summary)



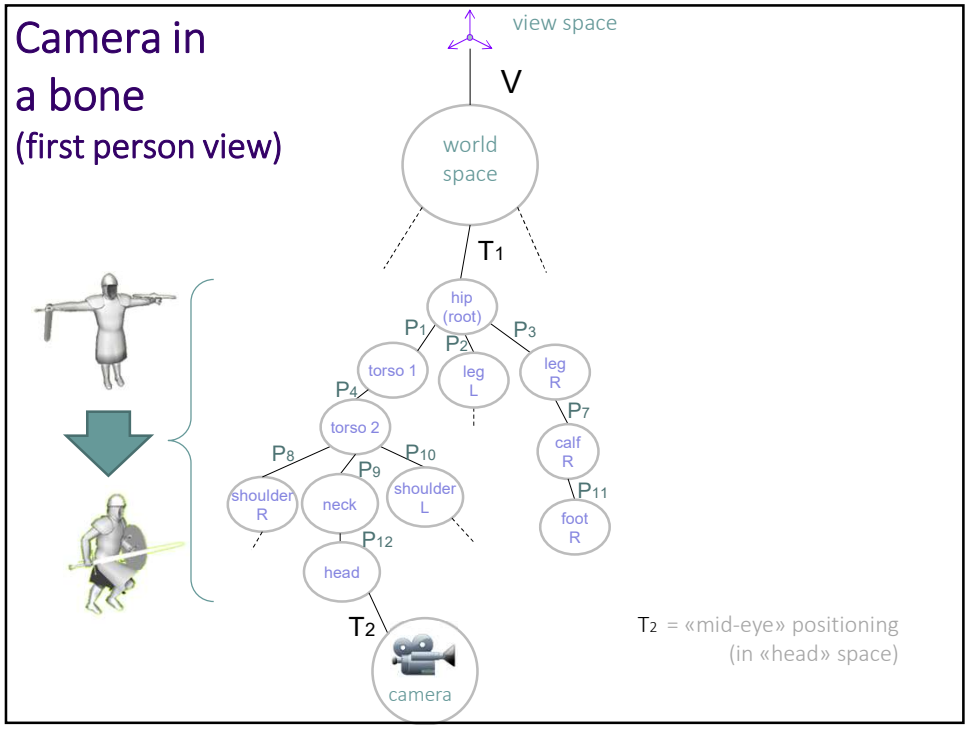
- A skeleton (rig) can be considered a subtree in the scene graph
  - It's local transforms are defined by the skeletal animation animations being rub
- Examples of techniques using the rig as just a part of the scene graph:
  - Placement of cameras in a node of the rig
  - Geometry proxies (hit boxes) for collision detection in nodes of the rig

144

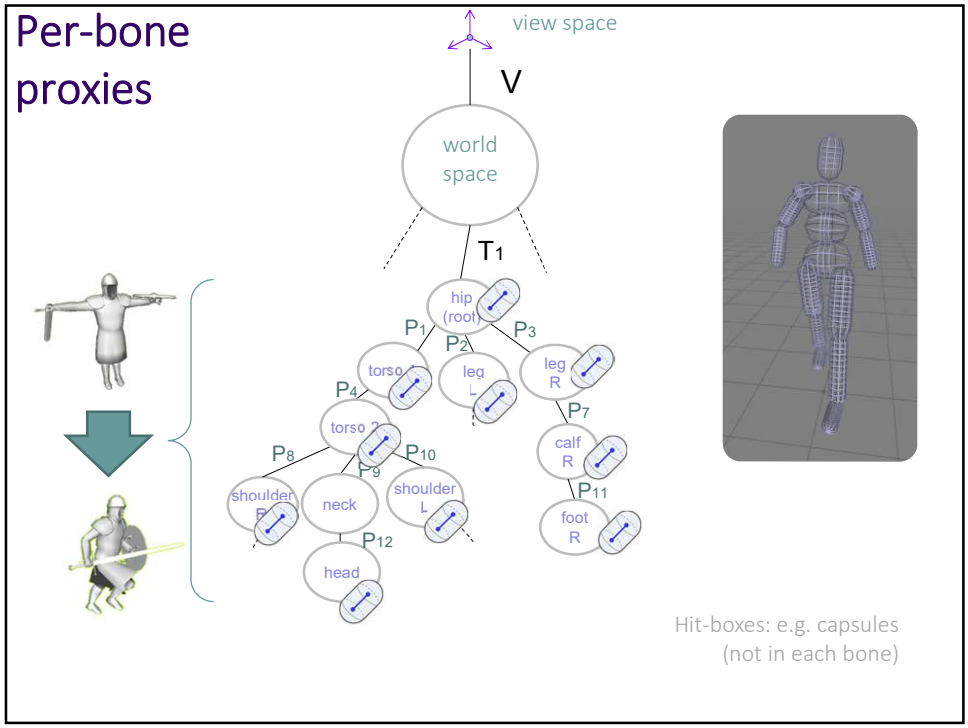
Third person view



145

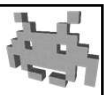

Camera in
a bone
(first person view)
view space
V
world
space
T1
hip
(root)
P1
P2
P3
torso 1
leg
L
leg
R
P4
torso 2
P7
calf
R
P8
P10
P9
shoulder
R
neck
shoulder
L
P11
foot
R
P12
head
T2
camera
T2 = «mid-eye» positioning
(in «head» space)

146


Per-bone
proxies
view space
V
world
space
T1
hip
(root)
P1
P2
P3
P4
P7
P8
P10
P9
shoulder
R
neck
shoulder
L
P11
foot
R
P12
head
Hit-boxes: e.g. capsules
(not in each bone)

147

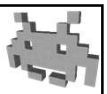
Ragdolling



- Idea: let a physical simulation simulate the evolution of the rigid-bodies defined as proxies in the bones
  - Including: gravity, reaction to forces, collisions with environment, self collisions
    (i.e., collision with other proxies in the same skeleton)
- Ingredients:
  - Per-bone proxies (at least a subset of the bones)
  - Constraints, such as…
    attachments of bones, constraint on rotations (“knees don’t bend backward or sideways”)
  - They can be expressed as positional constraint in a Position based Dynamics simulation
- Result: procedural skeletal animation

148

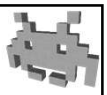
A few (pre)processing tasks for skeletal animations



- Compression
  - input: ani with N keyframes
  - output: ani with M<N keyframes
- Retargeting
  - input: Rig1 + (Skel animat for Rig1) + Rig2
  - output: (Skel animat for Rig 2)
- Building from a blend-shape animation
  - input: Blend-shape
  - output: Rig + Skinned Mesh + Anim
  - note: the opposite is a trivial («baking»)

149

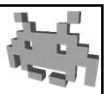
Compression of skeletal animations



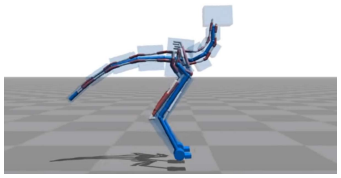
- Objective: remove keyframes
  - the "redundant" ones
  - preprocessing task (e.g., as a game tool)
- Basic algorithm concept:
  - for each keyframe $P\mathrm{x}$
    - tentatively remove $P\mathrm{x}$
    - compute interpolated version $P\mathrm{i}$ from remaining keyframes
      - (the prev. and next ones)
    - if *distance*($P\mathrm{i}$, $P\mathrm{x}$) > MAX_ERR then reinsert keyframe $P\mathrm{x}$

150

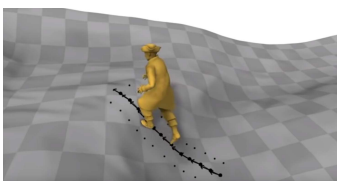
Research topic: apply ML to skeletal animations



- A very active area of research…



Flexible Muscle-Based Locomotion for Bipedal Creatures
Thomas Geijtenbeek, Michiel van de Panne, A. Frank van der Stappen
SIGGRAPH 2013

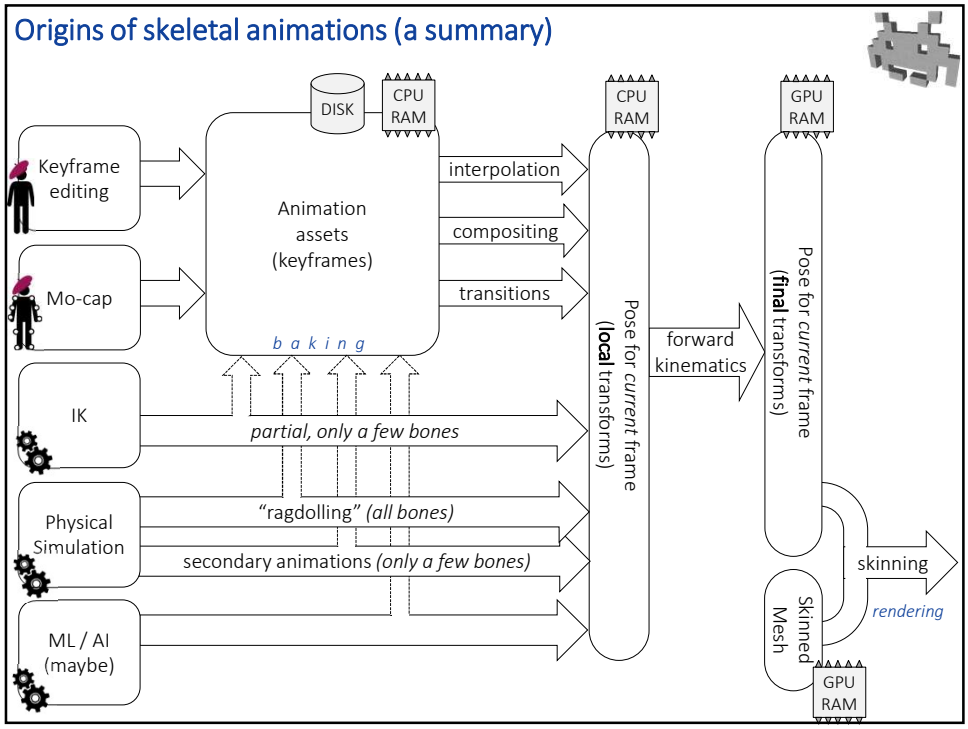


Phase-Functioned Neural Networks for Character Control
Daniel Holden, Taku Komara, Jun Saito SIGGRAPH 2017

(among MANY others)

151

Origins of skeletal animations (a summary)



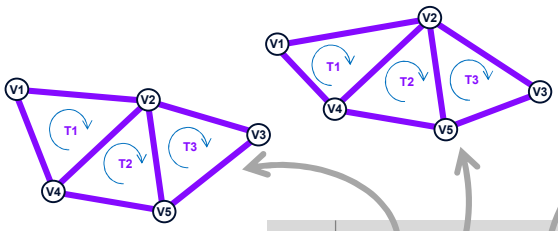
152

Per-vertex animations VS Skeletal-Animations

- Per Vertex animations
  - can interpolate keyframes (but linear trajectories)
  - heavy in RAM
    - replications of normals / positions
  - light to render / compute
- Skeletal animations
  - can interpolate keyframes *better* (curved trajectories)
  - light in RAM
    - animations / models orthogonality
  - minor overheads
    - transform interpolation (x vert!)
    - updates final transoform before (unless can be baked)

153

Observation: blandshapes + skeletal anims *can* be used togheter



Connectivity (shared):			
Tri:	Wedge 1:	Wedge 2:	Wedge 3:
T1	4	1	2
T2	4	2	5
T3	5	2	3

Vert:	*geometries:*				*shared attributes:*		
	Base Shape	Shape 1	Shape 2	Shape ...	UV	Bone links (skinning)	...
V1	(x, y, z)	(x, y, z)	(x, y, z)	...	(u, v)	$(b_0, w_0, b_1, w_1, b_2, w_2)$	
V2	(x, y, z)	(x, y, z)	(x, y, z)	...	(u, v)	$(b_0, w_0, b_1, w_1, b_2, w_2)$	
V3	(x, y, z)	(x, y, z)	(x, y, z)	...	(u, v)	$(b_0, w_0, b_1, w_1, b_2, w_2)$	
V4	(x, y, z)	(x, y, z)	(x, y, z)	...	(u, v)	$(b_0, w_0, b_1, w_1, b_2, w_2)$	
V5	(x, y, z)	(x, y, z)	(x, y, z)	...	(u, v)	$(b_0, w_0, b_1, w_1, b_2, w_2)$	

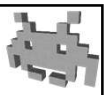
154

Observation: blandshapes + skeletal anims *can* be used togheter

- A blend shape can be skinned!
- Both animations can be combined
  - frame of the blend shape
    $\neq$
    frame of the skeletal animation
- Examples:
  - Breathing animations = blend shape,
    + Idle animation = skeletal anim
  - Cheeks puffing = blend shape (face morph)
    + mandible bone = skeletal animations
  - Blend shapes correctives (see later)

155

Example: LBS + Blend shapes with relative encoding



$$\mathbf{p}_P = \left(\sum_{i=0}^{3} w_i \, \mathrm{T}[b_i] \right) \left(\mathbf{p}_R + \vec{d} \right)$$

bone links (skinning)

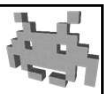
relative blend shape (note: it's applied in rest-pose, so, *before* the skinning)

rest pose

deformed mesh

156

Limits of skinning (both LBS and DQS)



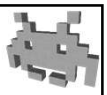
The bar for 3D game quality has gone up, but skeletal animations + skinning stayed the same for almost 10 years.

Deformations of the rest pose mesh is simple!

- Does not account for Dynamic effects:
  e.g. a fat belly jiggling up and down during a run
  - Solution 1: use blend shapes (e.g. blend shape correctives)
  - Solution 2: add new bones (belly bone), make a dynamic simulatin control the bone (aka in animation slang as a "secondary motion")
- Does not account for collision/contact
- Does not account for volume preservation
  - E.g., no muscle bulging
  - Can be in part compensated with skillful edit of bone weights
- Etc

157

Limits of skeletal animations notes:



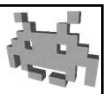
The bar for 3D game quality has gone up, but skeletal animations + skinning stayed the same for almost 10 years.

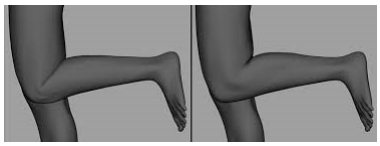
With skeletal animations:

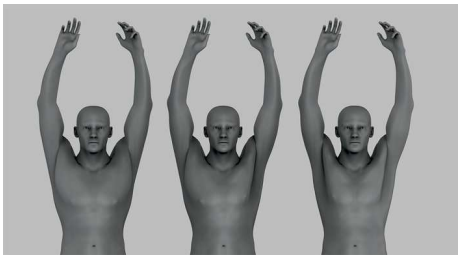
- Transitions between animation is crude, can look robotic
  - Possible solution: use ad-hoc animations for transitions
- Ragdolling = completely death "sack of potatoes"
  Authored Animation = character completely alive and in control, irresponsive to actual forces / dynamics
  - What about intermediate situations?
- IK not necessarily realistic
  - E.g.: feet are placed on the ground (not into it),
    but this is not how you would walk over a rugged terrain
- Animation is not physically based

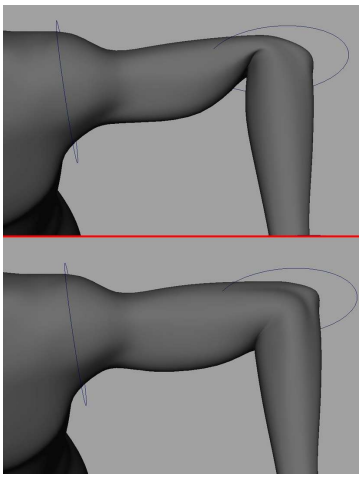
158

Blend Shape Correctives



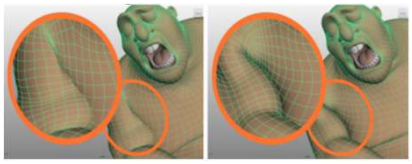






159

Research topic: Deformation beyond standard skinning

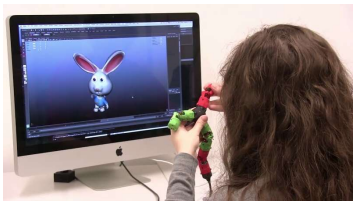


Efficient Elasticity for Character Skinning with Contact and Collisions
Aleka McAdams et al (Disney animation)
SIGGRAPH 11

Note: usually way more complex than direct methods (LBS / DQS).
More offline animation oriented than videogames

160

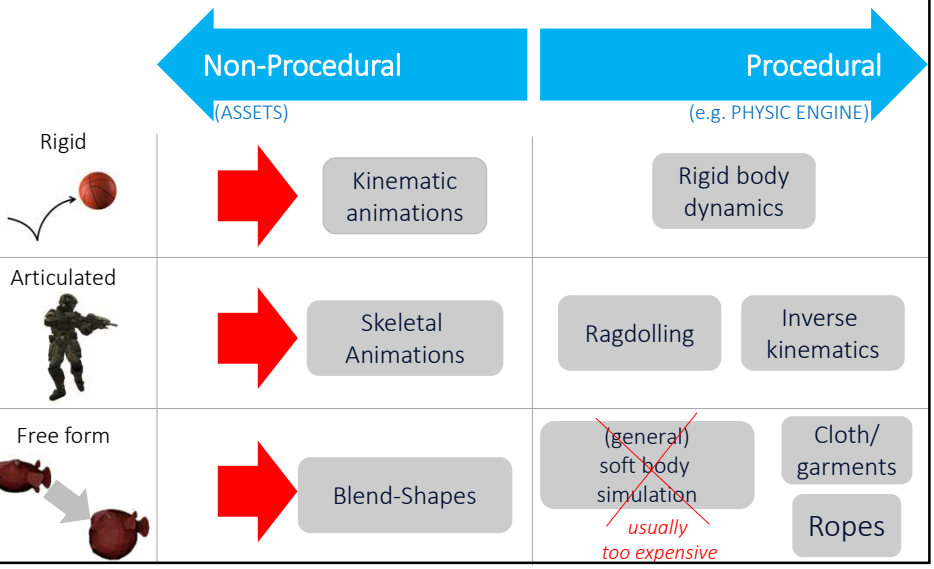
Research topic: better interfaces to author animations



Tangible and Modular Input Device for Character Articulation
Alec Jacobson, Daniele Panozzo, Oliver Glauser, Cedric Pradalier, Otmar Hilliges, Olga Sorkine-Hornung
SIGGRAPH 2014

161

Animations in games
(of 3D Solid Objects)

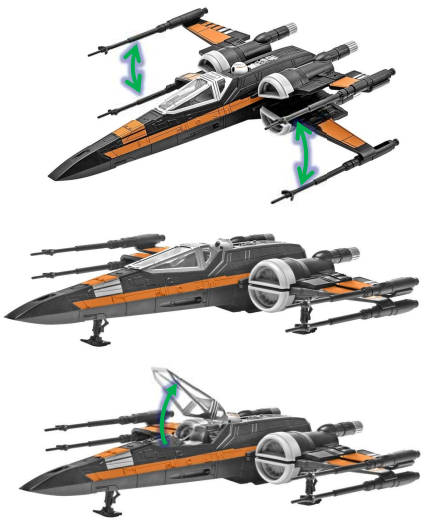


163

Non-procedural Animations: which one to pick?

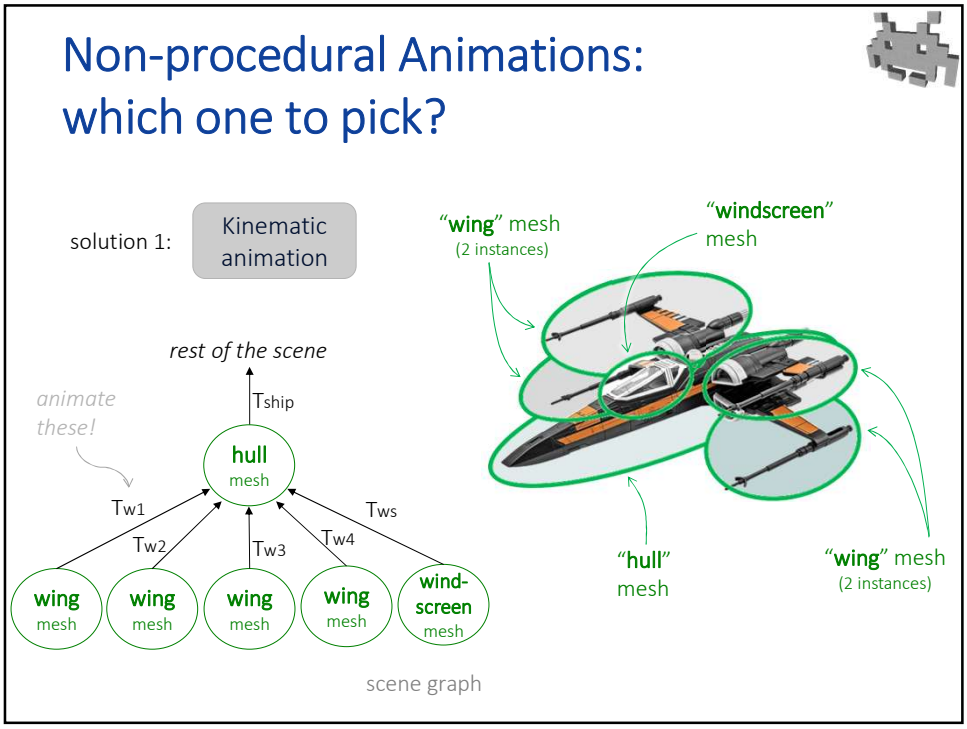
- Which format to pick?

EXAMPLE:
say we want
a model capable of
doing this:



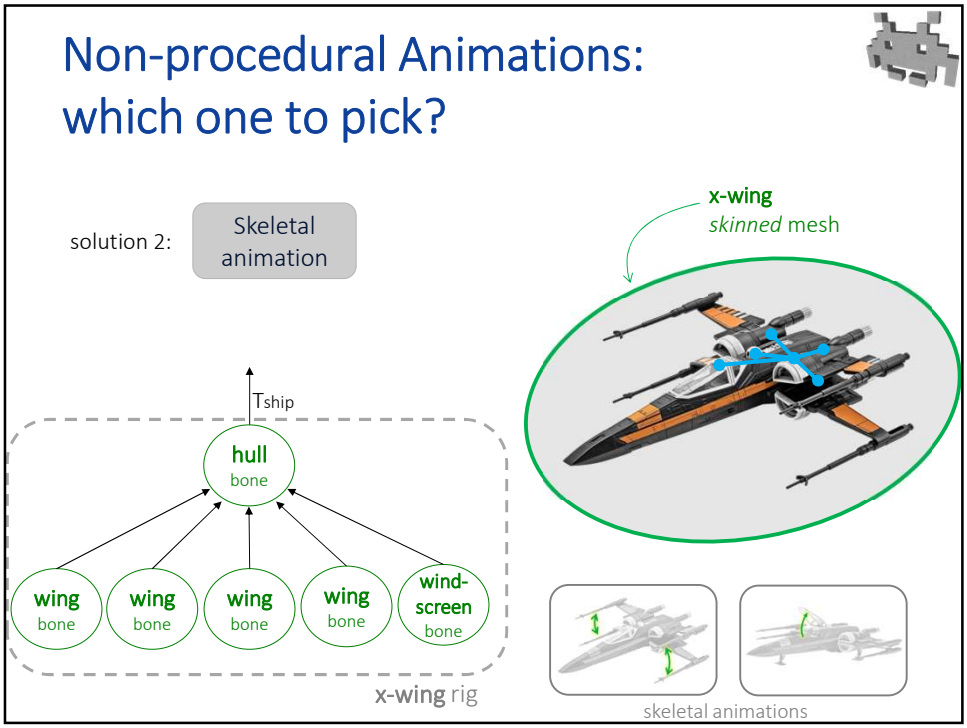
164

Non-procedural Animations: which one to pick?



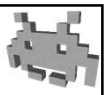
165

Non-procedural Animations: which one to pick?



166

Non-procedural Animations: which one to pick?

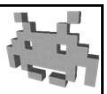


solution 3: Blend-shape

"x-wing" blend-shape

167

Non-procedural Animations: which one to pick?

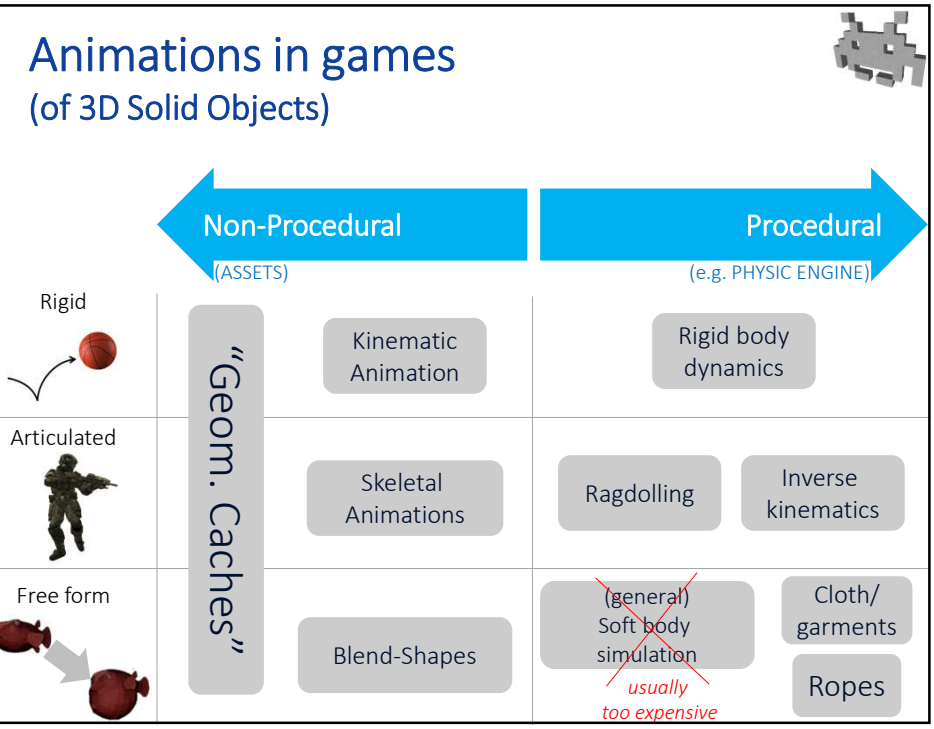


In this example:

- Animation of transforms (of the scenegraph):
  - *how:* 3 (rigid) meshes, 5 instances, animate scenegraph transforms
  - can reuse geometry for all wings: most compact on RAM ☺
  - simpler rendering
  - 5 separated draw calls! ☹
- Skeletal animation:
  - *how:* one rig + one skinned mesh + few skeletal animations
    - mesh skinning: single bone enough in this case
  - if very low poly mesh (few polys): a waste?
  - more taxing rendering (a bit) ☹
    - real time skinning on vertex any
  - single draw call! ☺
- Blend shapes:
  - *how:* blend shape with one base shape + 2 morphs
  - minimal impact
  - worst quality interpolation: linear → straight (non curved) paths
    - vertices on straight paths (unless, intermediate shapes are added)
  - heaviest on RAM ☹
    - (a waste of DoF!)
    - not important, if very low res
  - single draw call! ☺
    - but to different buffers each frame / or to a larger buffer

168

Animations in games
(of 3D Solid Objects)



169

Geometry Caches
(for lack of a better name)

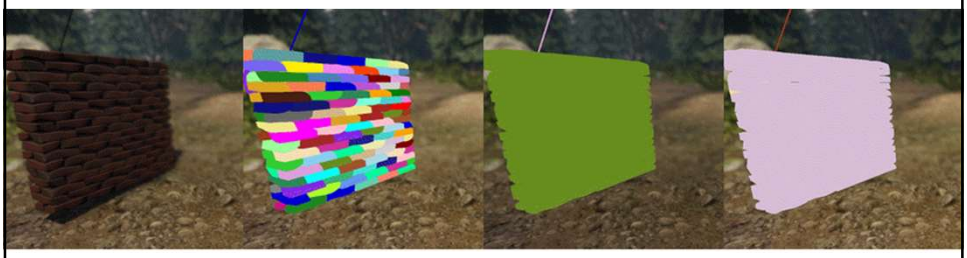
- Baked, optimized animations
  - of a mixture of types e.g.
    - blend shapes
    - kinematic animations
      - (approximated)
    - skinned animations
      - (typically, no scene graph, just final transf)
  - optimized
    - compressed, streamed…
  - Can be used to bake results of a physical simulation
    - i.e., convert it from physical to kinematic

one used file format: .abc

ALEMBIC by imageworks

170

Geom. Caches
(for lack of a better name)



- Baked, optimized animations
  - of the appropriate types including mixtures

Input:
170 Meshes
88400 Verts

as Pre-made Tansforms:
Meshes: 170
Data rate: 0.13 MB/s
Draw calls: 170
(same ones each frame)

as a Blend Shape:
Meshes: 1, with N shapes
Data rate: 4.3 MB/s
Draw calls: 1
(different one each frame)

as a Skeletal Animation
Meshes: 1, w skinning (*)
Data rate: 0.13 MB/s
Draw calls: 1
(same one each frame)
(*) just 1 bone per vertex

Geometry Caches
(a subset of Alembic)

by CRYENGINE

171

Animations
in Mecanim (Unity) (notes)

- Assets (models, animation, skeletons) imported as formats:
  - fbx, collada
- Animation compression
  - available during import / builds
  - auto reduction of: num of links per vertex, num of keyframes … :
- «Animator Controller» module → deals with:
  - blending between animations: «transitions»
  - compositing animations: «layers»
    - e.g.: a layer overwrites upper body bones
  - and is nicely WYSIWYG (graph visualization)
- Inverse Kinematic: with scripts (`Avatar.SetIKPoistion`)
- Skeletons:
  - way 1: custom (imported as assets)
  - way 2: built-in standard humanoid skeleton provided
    - (~21 ossa)
    - simplified: rigging (predefined constrains), layers (predef. labelling)

172