

Lezione 10 10/04/2025

Visual effects e Animations

Sono tutti gli elementi dinamici che sono usati per mostrare fenomeni. Possono essere creati come effetti particellari che si muovono nello spazio, con shaders, luci...



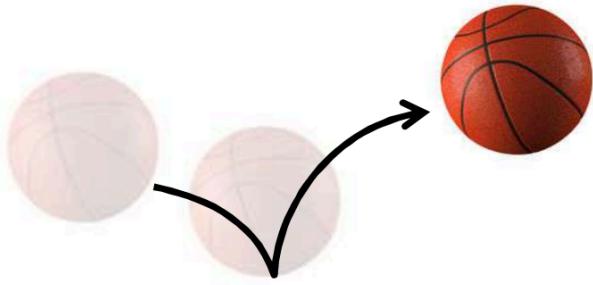
Migliorano il coinvolgimento, l'atmosfera.

Non sono da confondere con le **animazioni**. Le animazioni sono processi di creare movimenti per i personaggi, seguono una struttura preimpostata.

Feature	Animation	VFX (Visual Effects)
Purpose	Movement of characters/objects	Enhance visuals with dynamic effects
Control	Keyframe-based, rigged, or procedural	Procedural or physics-based
Typical Use	Walk cycles, UI transitions, facial movement	Explosions, spells, smoke, hit impacts
Driven By	Animator, gameplay systems	Particle systems, shaders, events
Style	Deliberate, repeatable motion	Reactive, dynamic, and often randomized
Tools	Animation rigs, timeline editors	Particle editors, shader graphs, VFX editors

1. of rigid objects

- animate scene transformations

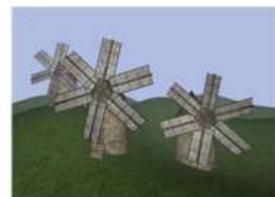


6 gradi di libertà perchè si può fare una roto-traslazione.

Non per forza definiremo l'animazione per ogni frame, le trasformazioni nel mezzo di due keyframes vengono fatti **interpolando**.

1. of rigid objects

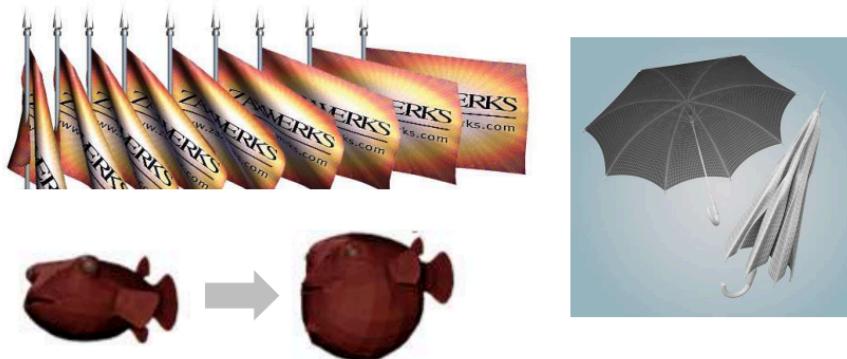
- or objects made of rigid sub-parts



Possiamo animare anche oggetti che sono **composti da una collezione di parti rigide**.

2. Free-Form deformations

- generic transformations of the object



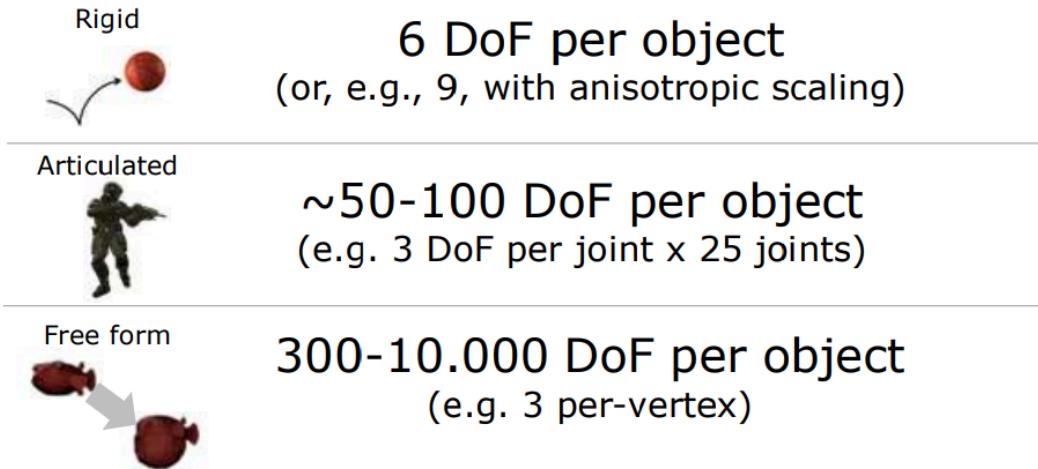
Possiamo anche animare **oggetti non rigidi**, la geometria dell'oggetto cambia.

3. of articulated models

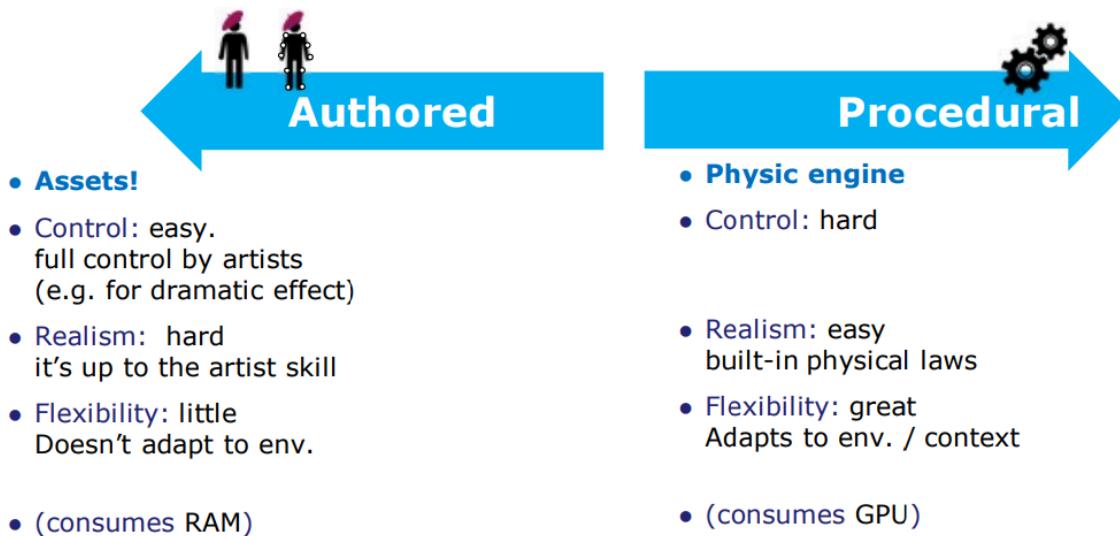
- internal skeleton
- most virtual characters!
- “skinning”



L'animazione dei **modelli articolati** viene fatta tramite lo scheletro interno. Lo skinning lo vediamo dopo.



Ci sono diversi tipi di animazioni: quelle che vanno fatte da un'**artista** (che consideriamo **asset**) e quelle che si possono ottenere tramite **pipelines procedurali** (realizzate dal **motore fisico**).

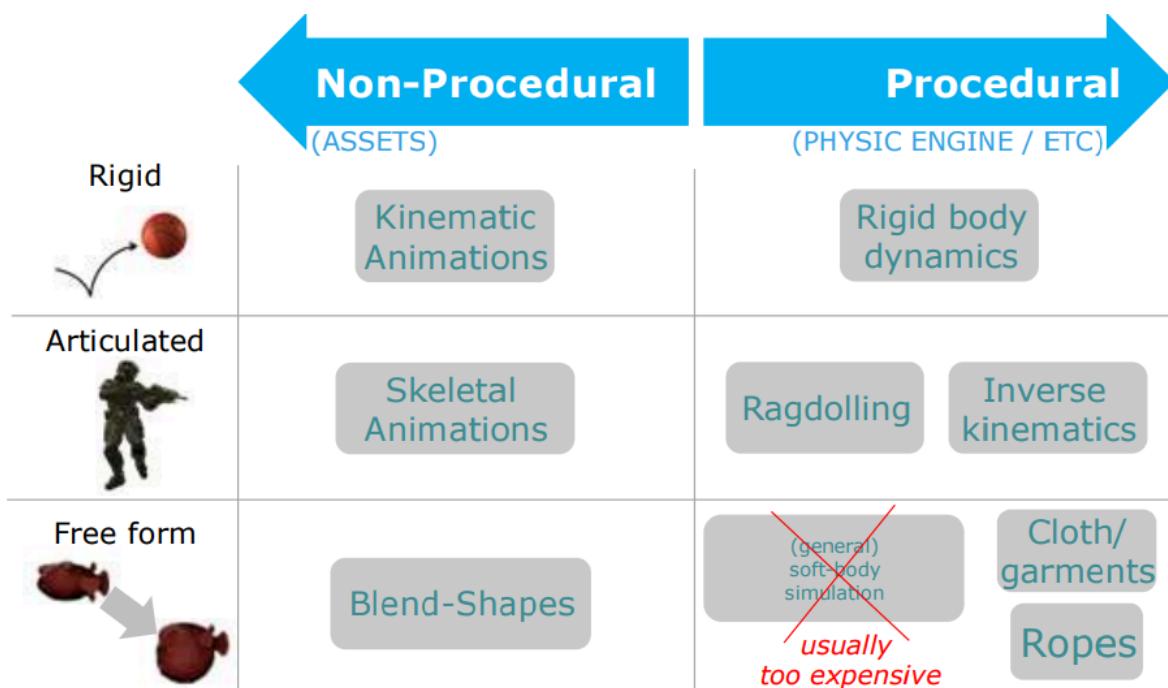


Si fa un mix delle due. Possiamo avere per esempio:

- 1: "primary" animations: authored
"secondary" animations: physically generated
 - 2: alive characters: authored
dead characters: physically generated ("ragdolls")
 - 3: walk cycle: authored (skeletal animation)
exact feet placement: procedural (inverse kinematic)
 - 4: normal "behavior", such as sparring: authored
gaze control during sparring: procedural
 - 5: normal "behaviors" such as jumping, running: authored
modifications / transitions: AI generated
- and more!

Volendo aggiungendo anche animazioni fatte tramite AI.

C'è così tanta attenzione a mixare le animazioni perchè da un lato bisogna mantenere basso il costo ma dall'altro si vuole avere un effetto realistico, coinvolgente.



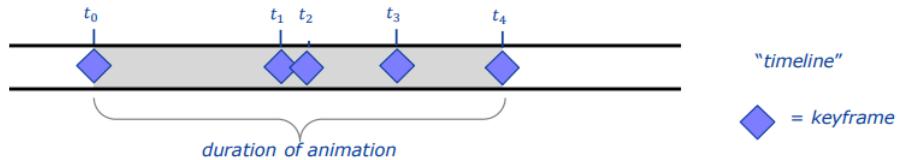
Per gli oggetti rigidi, quando vogliamo un meccanismo non procedurale allora entriamo nella **cinematica**, ovvero una serie di operazioni definite a meno su oggetti rigidi.

Noi ci focalizziamo sulla colonna delle animazioni non procedurali.

Le animazioni vengono definite come un insieme di **keyframes** per i quali definiamo la trasformazione, ciascuno con un istante temporale.

- ◆ The animation asset stores only a subset of the frames ("key"-frames)

- each with its own associated *time*



- ◆ Other frames ("in-betweens") are interpolated **keyframes**

- saves storage RAM (only keyframes are stored)
 - saves artist work (only keyframes are constructed)
 - animation can be very smooth (avoids temporal aliasing)
e.g., even when played at extreme slow-motion
 - this implies the ability to *interpolate* key-frames!

- ◆ keyframes distribution can be *adaptive*
 - more keyframes only where needed
- ◆ in-betweening happens automatically on demand, in real time
 - e.g., at each refresh of video
- ◆ times associated to keyframe are chosen arbitrarily
 - not necessarily as an integer number, of video frames
 - all frames shown on screen will be in-betweens
- ◆ the better the interpolation schema
→ the better in-betweens → the fewer keyframes are needed
- ◆ editing the animation:
 - editing individual keyframes
 - editing keyframe *times* (e.g., to achieve non-linearity of moment, vary speed)
 - 1. pick a new time t_i (not a keyframe)
2. **bake** the in-between at t as a new keyframe
3. edit it!

Blend shapes

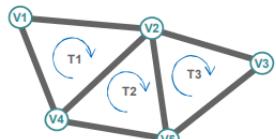
Le **blend-shapes** sono quelle più difficili. Queste sono usate per free-form animations. Sono anche conosciute come:

- Morph-targets (animations)
- Face-morphs
- Shape-keys
- Per-vertex animations
- Vertex-animations
- ...

L'idea arriva dal 2D, dove per simulare l'animazione di un oggetto si faceva una collezione di sprites differenti. Come possiamo ottenere lo stesso effetto nel 3D?



L'idea è quella di modificare le coordinate dei vertici, al posto di cambiare l'intera mesh.



connectivity (indexed)

Tri:	Wedge 1:	Wedge 2:	Wedge 3:
T1	V4	V1	V2
T2	V4	V2	V5
T3	V5	V2	V3

geometry:

Vert :	Pos
V1	(x,y,z)
V2	(x,y,z)
V3	(x,y,z)
V4	(x,y,z)
V5	(x,y,z)

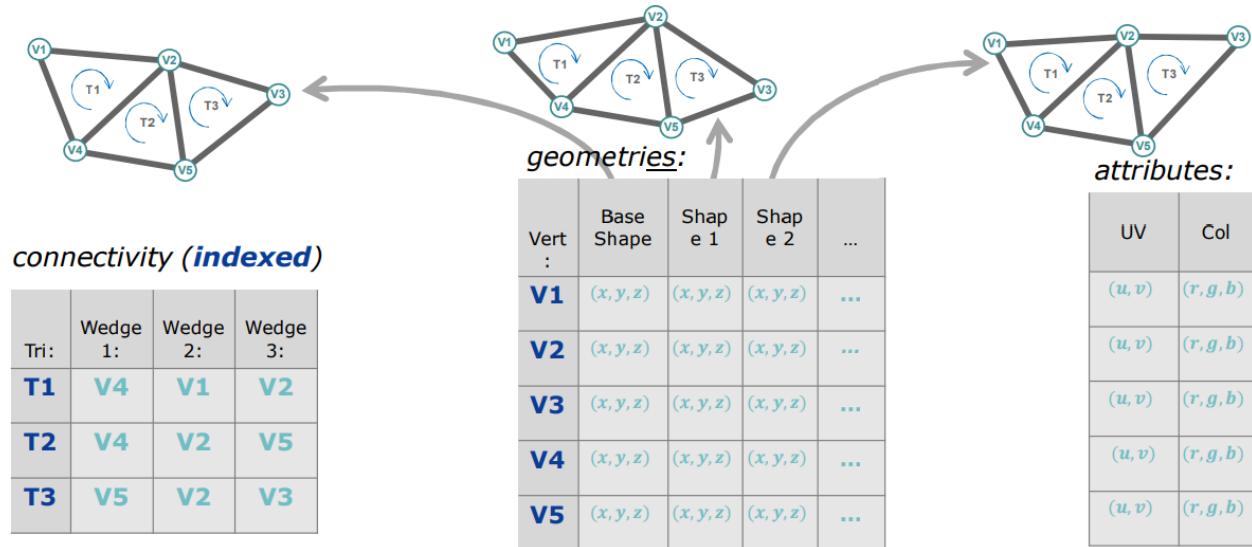
attributes:

UV	Col
(u,v)	(r,g,b)

Le mesh sono salvate in memoria con la vertex list e la face list (tabella a sinistra), e poi abbiamo la geometria (tabella al centro) che ci dà l'ordinamento dei vertici.

Poi si può salvare anche l'UV mapping, o altre informazioni come textures, o anche le normali per non doverle ricalcolare.

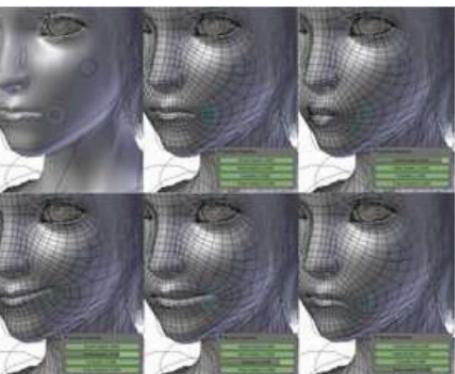
Quindi per le blend shapes, salvo queste informazioni per mostrare le deformazioni:



Cambia la tabella della geometria, le facce sono le stesse e l'UV mapping è lo stesso.

Quindi fisso una mesh e gli associo una sequenza ordinata di geometrie.

- ◆ A mesh with several associated **geometries**
- ◆ I.e. a sequence of meshes ('**shapes**') with
 - shared connectivity
 - many shared attributes
 - except normals / tangents dirs
 - shared UV-map, per vertex colors...
 - different geometries
 - (and **shared textures** as well)
- ◆ Encoding (they are equivalent):
 - **Relative mode:**
 - *base shape*: stored as per-vertex positions (points)
 - any other *shape*: stored as difference with *base shape* (vectors)
 - **Absolute mode:**
 - each *shape* stored as per-vertex positions (points)



La connettività (face list) è sempre la stessa. Le normali invece per esempio cambiano perchè dipendono dalla geometria.

Possiamo rappresentare le blend shapes in due modi (equivalenti):

- **encoding relativo:** salvo la shape di base, e poi salvo la differenza rispetto alla base shape per fare l'animazione.
- **encoding assoluto:** per ciascuna shape si salvano le posizioni.

Tutte le shapes di una blend shape **condivideranno**:

- la **connettività**
- la **topologia della superficie** (quello che non è separato rimarrà non separato, quello che è separato rimarrà separato)
- **attributi** come il colore, UV-map... ad eccezione delle posizioni e delle normali.
- le **texture**.

Ci sono diversi formati:

- ◆ Formats supporting blend-shapes include:
 - **.GLTF** (Khronos)
 - **.DAE** (Collada)
 - **.FBX** (Autodesk)
- ◆ Older / simpler alternatives:
 - **.MD5** ("quake", valve)
 - just store a sequence of meshes (es **.OBJ**)
 - making sure connectivity is coherent!
(vertex, face ordering must be the same - can be tricky)

noi useremo FBX penso.

- ◆ shapes = keyframes



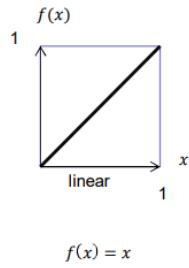
Blending keyframes

- ◆ shapes = keyframes of the animation
 - shape_A with time t_A
 - shape_B with time t_B
 - shape_C with time t_C
 - shape_D with time t_D
- ◆ given current time t with $t_B \leq t \leq t_C$
- ◆ then...
 - which shapes to blend? shape_B, shape_C
 - weights? $w_B = f\left(\frac{t - t_C}{t_B - t_C}\right)$ *transition function*
 $w_C = (1 - w_B)$

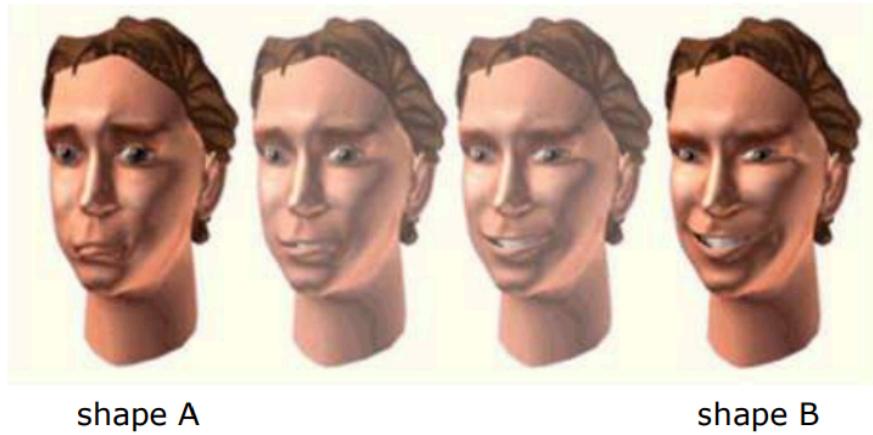
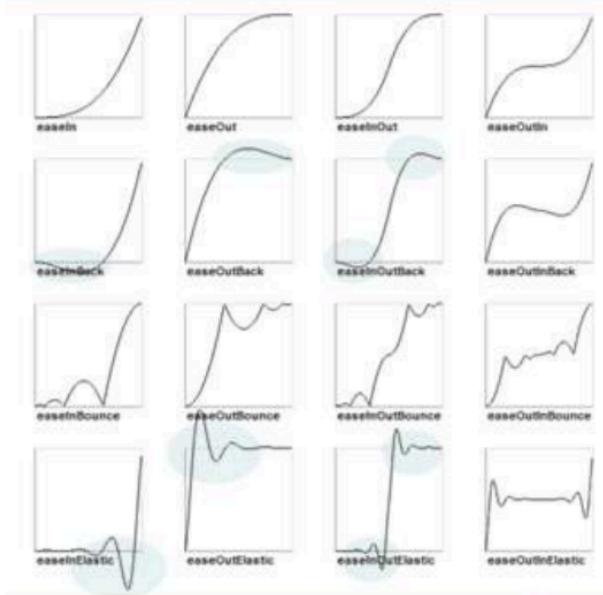
Se l'istante di tempo sta tra il tempo b e il tempo c, allora vado a mischiare la forma b e c facendo per esempio un'interpolazione lineare.

Altrimenti si può pensare anche di usare una funzione per stare più vicino con l'animazione a b per più tempo e meno per c. Questa è detta **funzione di transizione**, dice come si passa da una shape all'altra.

- ◆ Not necessarily the Linear one



NB:  = extrapolation !
i.e. exaggeration



here: shapes = facial expressions
(typical use; that's why they are also called "face morphs")

Le blend shapes vengono usate per le espressioni facciali.

- ◆ The **absolute mode** is natural when shapes are designed to be used as *alternatives*.

- keyframes of an animation sequence.

shape₁ = keyframe of time 5

shape₂ = keyframe of time 9

at time 6, we use

$$0.75 \text{ shape}_1 + 0.25 \text{ shape}_2$$

w_1

w_2

- ◆ The **relative mode** is natural when shapes are designed to be *superimposed*.

- shape₁ = left-eye closed

- shape₂ = smile

- base + shape₁ + shape₂ = wink

- ◆ But the two ways are equivalently expressive = They can achieve the same set of shapes

- In **absolute mode**, $\sum w_i = 1$

- In **relative mode**, there is no such condition, but

it's equivalent to use the absolute mode, with $w_{base} = 1 - \sum w_i$

- when $\sum w_i > 1$, this means that we are implicitly using an **extrapolation**

Il modo assoluto ha più senso quando le singole shapes potrebbero anche essere usate come alternative, cioè se non si ha per forza una shape base. Quindi per comporle vado a mettere un peso a ciascuna shape.

Si può anche fare **estrapolazione**, esagerando in una delle due direzioni.

	using Absolute Encoding	using Relative Encoding
What is stored	<p>base shape (positions)</p> <p>$S_b, S_0, S_1, S_2 \dots$</p> <p>$S_b + R_0$ $S_b + R_1$</p>	<p>base shape (positions)</p> <p>$S_b, R_0, R_1, R_2 \dots$</p> <p>$S_0 - S_b$ $S_1 - S_b$</p>
Equivalent ways to blend...	<p>two shapes i and j</p> $w_i S_i + w_j S_j$ <p>three shapes i, j and k</p> $w_i S_i + w_j S_j + w_k S_k$ <p>More than three</p> <p style="text-align: right;">$\text{with } \sum w = 1$</p>	$S_b + w_i R_i + w_j R_j$ <p>$S_b + w_i R_i + w_j R_j + w_k R_k$</p>

	using Absolute Encoding	using Relative Encoding
What is stored	<p>base shape (positions)</p> <p>shapes (positions)</p> $S_b, S_0, S_1, S_2 \dots$ <p>$\underbrace{S_b + R_0}_{S_b + R_1}$</p>	<p>base shape (positions)</p> <p>shapes (vectors)</p> $S_b, R_0, R_1, R_2 \dots$ <p>$\underbrace{S_0 - S_b}_{S_1 - S_b}$</p>
Equivalent ways to blend...	<p>base shape with one shape i</p> $(1 - w)S_b + w S_i$ <p>base shape with two shapes (i, j)</p> $(1 - w_i - w_j)S_b + w_i S_i + w_j S_j$ <p>base shape with three shapes</p> $(1 - w_i - w_j - w_k)S_b + w_i S_i + w_j S_j + w_k S_k$ $\sum w = 1$	$S_b + w R_i$ $S_b + w_i R_i + w_j R_j$ $S_b + w_i R_i + w_j R_j + w_k R_k$

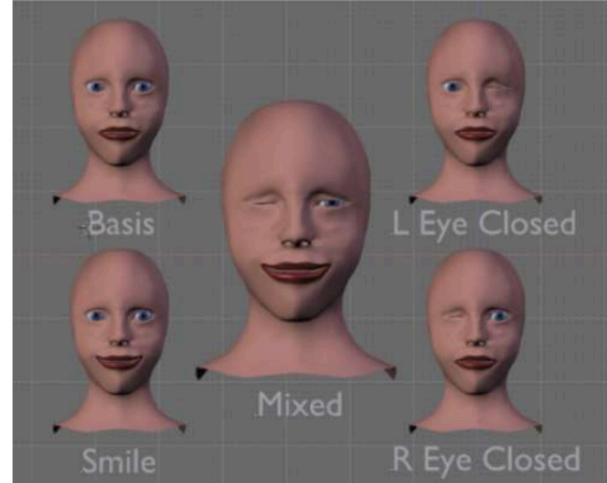
- ◆ Shape 0 = base shape
- ◆ Shape 1 = smile
- ◆ Shape 2 = left eye closed
- ◆ Shape 3 = right eye closed

«wink» = 75% smile + right eye fully closed

How to blend the shapes to get a wink
(which weight to use),
if the blend-shape is encoded in...

A: Relative mode?

B: Absolute mode?



Blend shapes: authoring

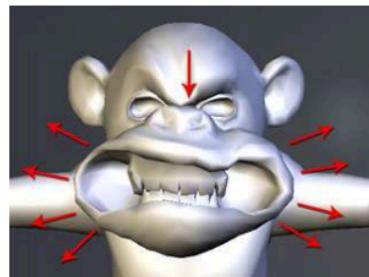
1. Editing base shape

- including:
uv-mapping, texturing, etc.



2. Re-edit it for each shape-key! ...while preserving: connectivity, textures, etc:

- with low poly editing
- or with subdivision surfaces...
- or with parametric surfaces...
- or with sculpting.



Le blend shapes hanno il vantaggio che se andiamo a fare editing della base shape, per esempio mettendo una texture, UV mapping... poi non bisogna farlo per gli altri keyframes perché la connettività è preservata (anche se a volte può non uscire bene).

Pros and cons

◆ During authoring:

- 👉 flexible, expressive, huge number of DOF (too many?)
- 👉 work intensive to construct
- 👉 expensive to store

◆ During use (by animators)

- 👉 easy to use (just define global weights)
- 👉 RAM cost
- 👉 very little degree of freedoms the number of weights (too few?)

but, not as bad as old sprites,

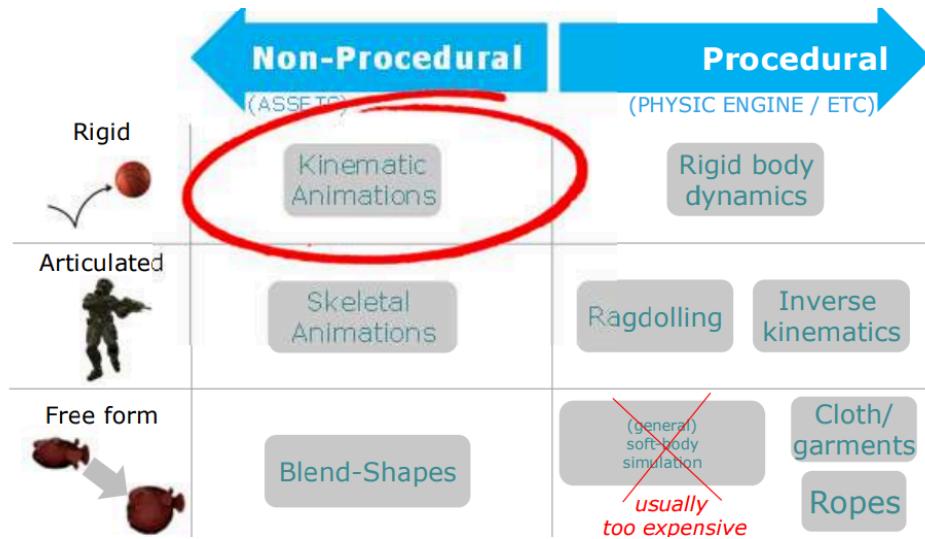


Because:

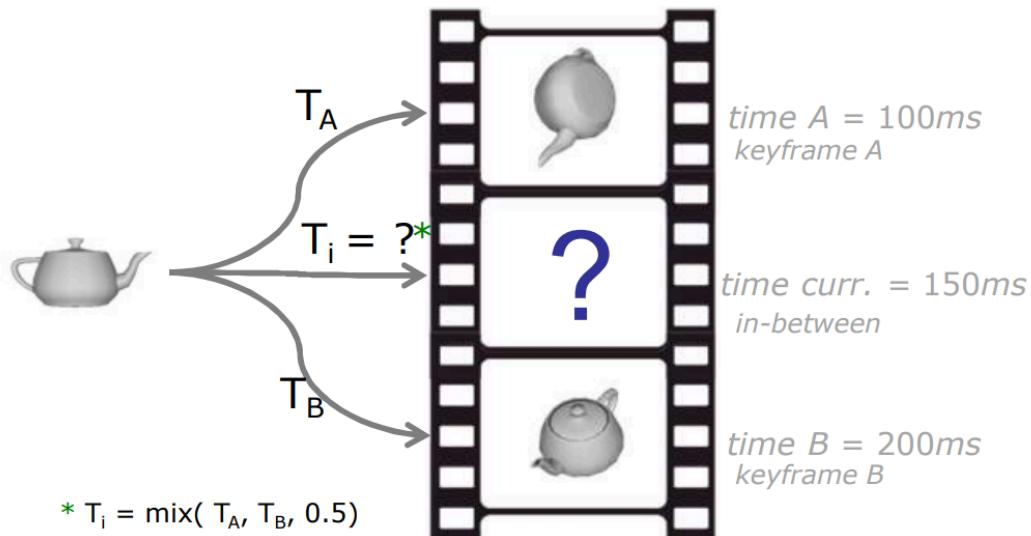
- (1) shared of connectivity, textures, attributes
- (2) keyframes / inbetweens!

DOF sono 3 per numero di vertici.

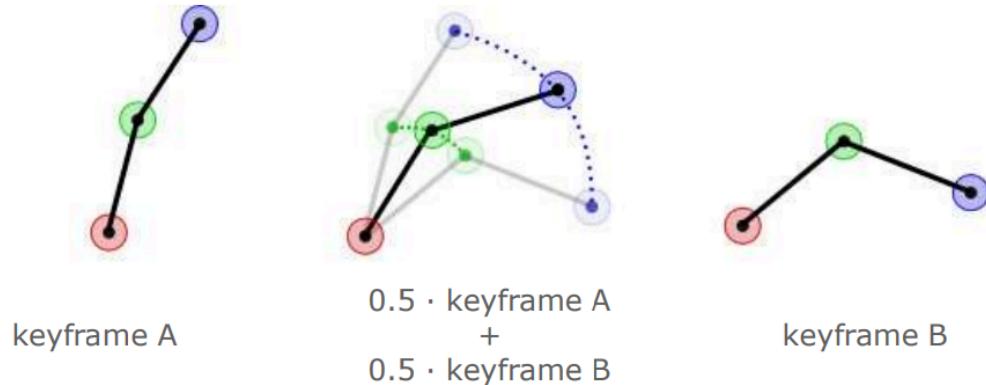
Cinematica



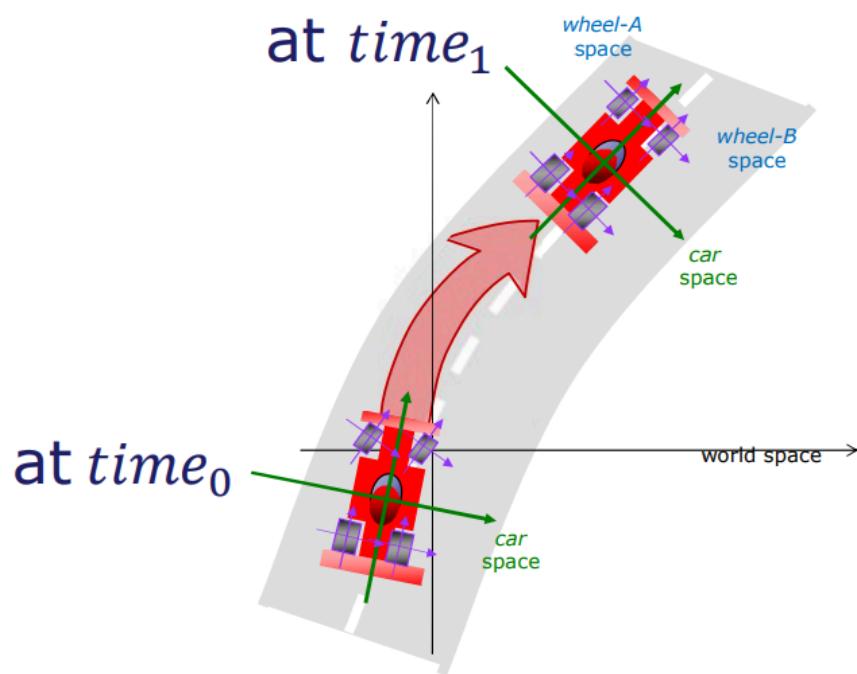
Se conosco la posizione dell'oggetto rigido all'istante a e all'istante b, posso trovare la posizione dell'istante nel mezzo facendo interpolazione:



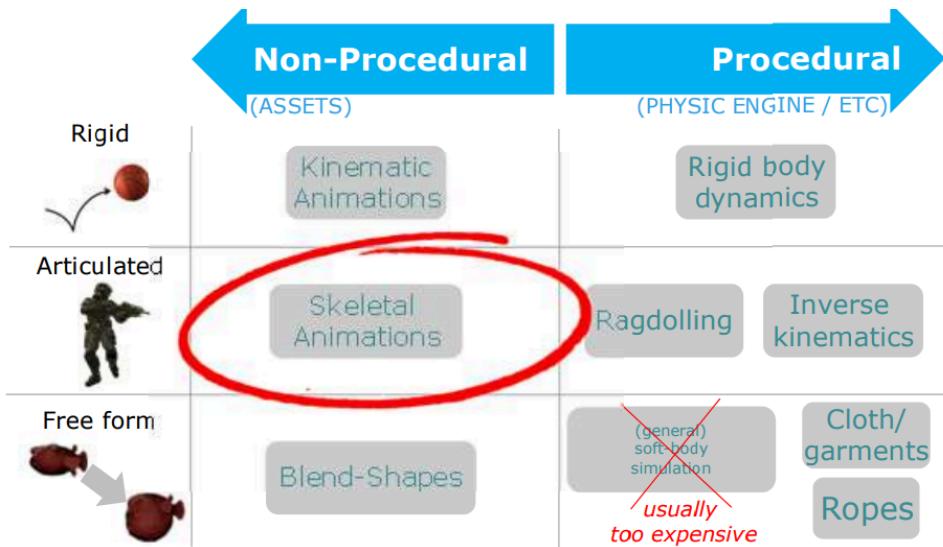
- ◆ Keyframes + in-betweens (interpolation)



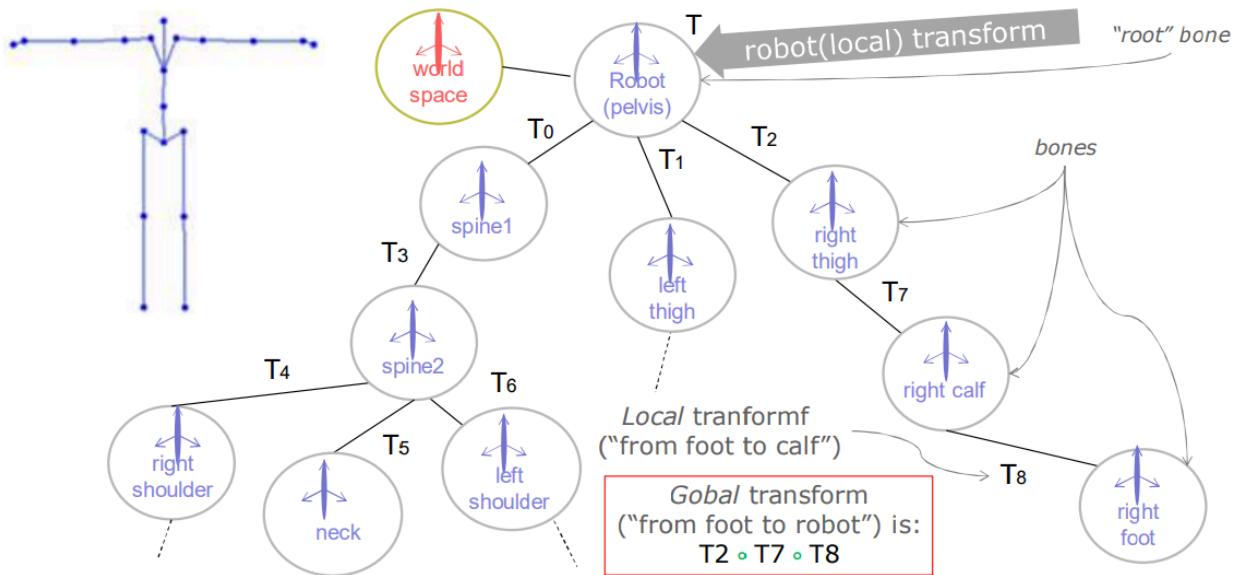
Per oggetti complessi composti da più oggetti rigidi separati, devo fare interpolazione tenendo conto della gerarchia delle trasformazioni (lo spostamento della macchina induce uno spostamento delle ruote) e della trasformazione che c'è solo sulle ruote e non sulla macchina.



Oggetti articolati: scheletri



L'esempio standard è quello di un corpo umano. Si definisce una **struttura ad albero** dove c'è una trasformazione sul nodo base, che ci dice la posizione nello spazio ricevendo la trasformazione dal world space. Tutte le trasformazioni dei nodi che seguono sono descrivibili come composizioni di trasformazioni gerarchiche applicate una all'altra.



Quindi come posso usare questa struttura su una mesh?

L'idea è quella di utilizzare una sola mesh, ma **skinned**.

◆ Idea: one **mesh**, but **skinned**



- 1 mesh per the entire character
- a new attribute per vertex: *index of bone*
- the 3D model can now be animated!

◆ Orthogonality models / animations!

- one skinned mesh: runs with any animations
- one skeletal animation: can be applicable to any model
 - (as long as they use the same skeleton)
- → 500 models + 500 animations = 1000 things in GPU RAM
 - not: 500x500 combinations

"*Skinning*"
of the mesh
(1st version).

Quindi dichiariamo che i vertici della mesh che rappresenta il personaggio hanno un'appartenenza alle varie ossa dello scheletro.

In questo modo se ho uno scheletro uguale/simile, posso trasferire le animazioni.

◆ The tasks required from digital artists:

- "**rigging**": define the **skeleton** (the rig) inside the mesh (by riggers)
- "**skinning**": define vertex-to-bone links, i.e. the skinning (by skinners)
- "**animation**": define the actual animations (by animators)

Si parte da uno scheletro, è un albero perchè devo avere un nodo padre. Ogni osso è rappresentato da un vettore nello spazio (vettore perchè ha anche una lunghezza), il verso ci descrive la gerarchia.

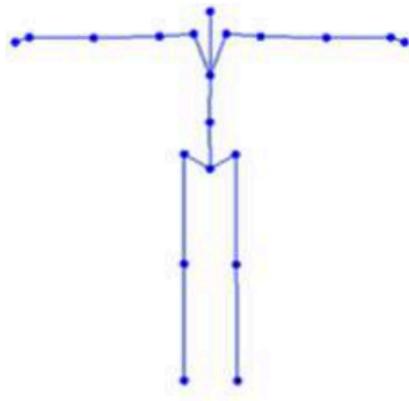
◆ A tree of **bones**

◆ **bone**:

- Vectorial frame (space) used to express pieces of the animated model
- eg, for a humanoid: *forearm, calf, pelvis, ...*
- (animation bones != biological bones)

◆ Space of the *root bone* =(def)= **object space** (of the entire character)

◆ How many bones in a skeleton of a humanoid:
at least: 22-24 (typically)
reasonable: ~40 bones.
very high: few 100s

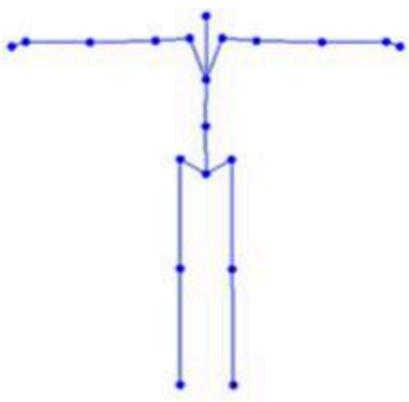


1. **Hierarchy (tree) of bones**

- a **root bone** on top

2. A special **pose «rest pose»**

- 3D models are to be modelled in this pose
- also: «T-pose», «T-stance»,
 - same reason why T-shirts are called T-shirts ;)
- also: «A-pose», when arms are bent down



Poi abbiamo una trasformazione per ogni osso, rispetto al nodo che gli fa da padre.

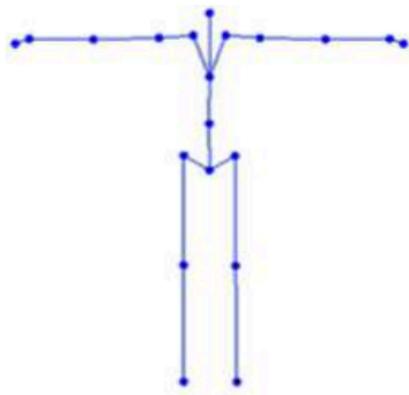
One transformation for each bone i

◆ Local transform: (of bone i)

- from: space of one i
- to: space of bone father of i

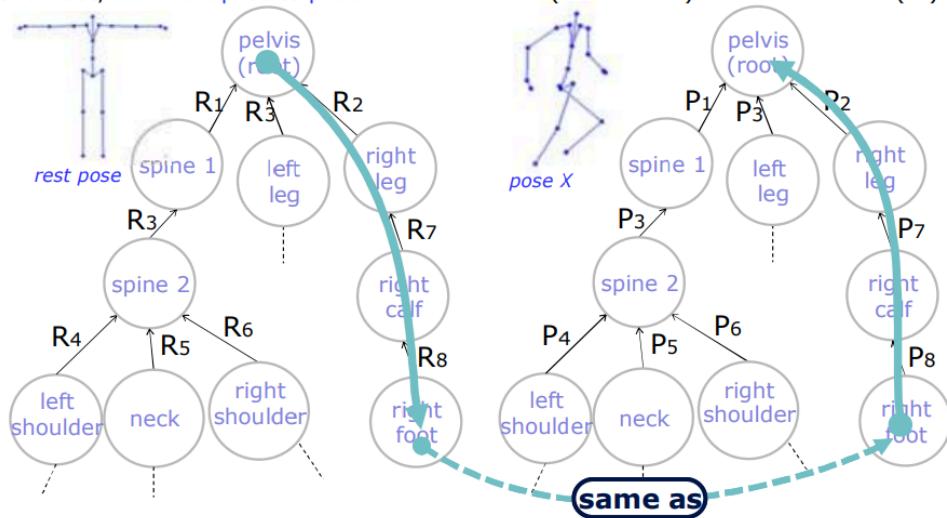
often, only the rotation component

("fixed length bones":
translations defined once and for all by the skeleton)



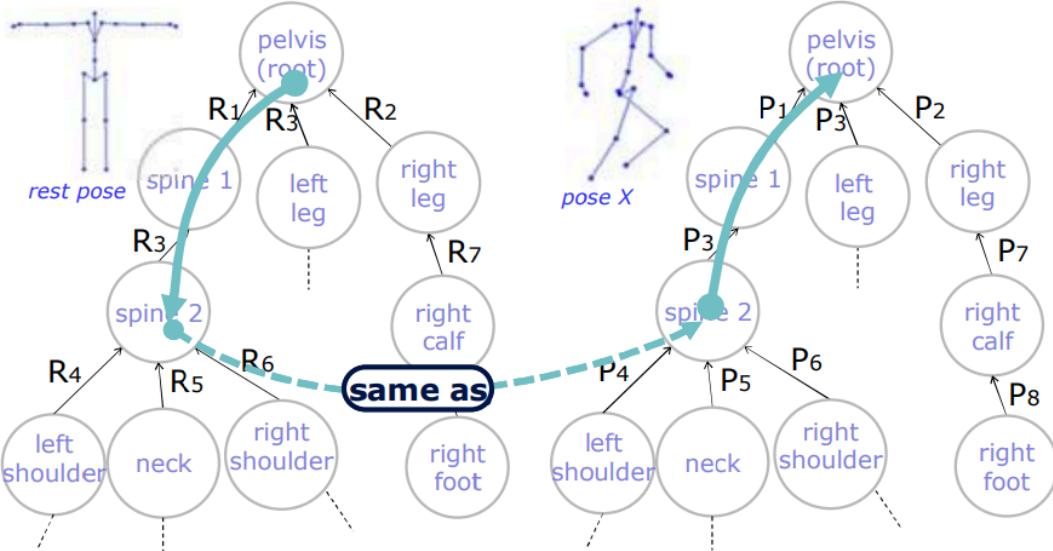
Per passare da una posa all'altra, si ottiene la trasformazione che è l'inversa della composizione delle trasformazioni sul nodo per la rest pose e la composizione delle trasformazioni sulla posa che voglio andare ad ottenere.

final trans for foot, from **rest pose to pose X** = $P_2 \circ P_7 \circ P_8 \circ (R_2 \circ R_7 \circ R_8)^{-1} = P_2 \circ P_7 \circ P_8 \circ (R_8)^{-1} \circ (R_7)^{-1} \circ (R_2)^{-1}$



Questo vale per ogni nodo.

final trans for the spine, from **rest pose** to **pose X** = $P_1 \circ P_3 \circ (R_1 \circ R_3)^{-1} = P_1 \circ P_3 \circ (R_3)^{-1} \circ (R_1)^{-1}$



Qui abbiamo un esempio per il piede, la **local transform** è data da P_8 , che era la trasformazione che andava dal figlio piede al padre. La **global transform** è la composizione di tutte quelle che arrivano fino al padre, definite dalla gerarchia e dal nodo radice. La **composizione finale** è la composizione di questa trasformazione con l'inversa della trasformazione definita sulla rest pose.

◆ Local Transform: P_8

- from «right foot» to «right lower leg»

the local frame
of the character,
which is the frame of
the **root bone**

◆ Global Transform: $P_2 P_7 P_8$

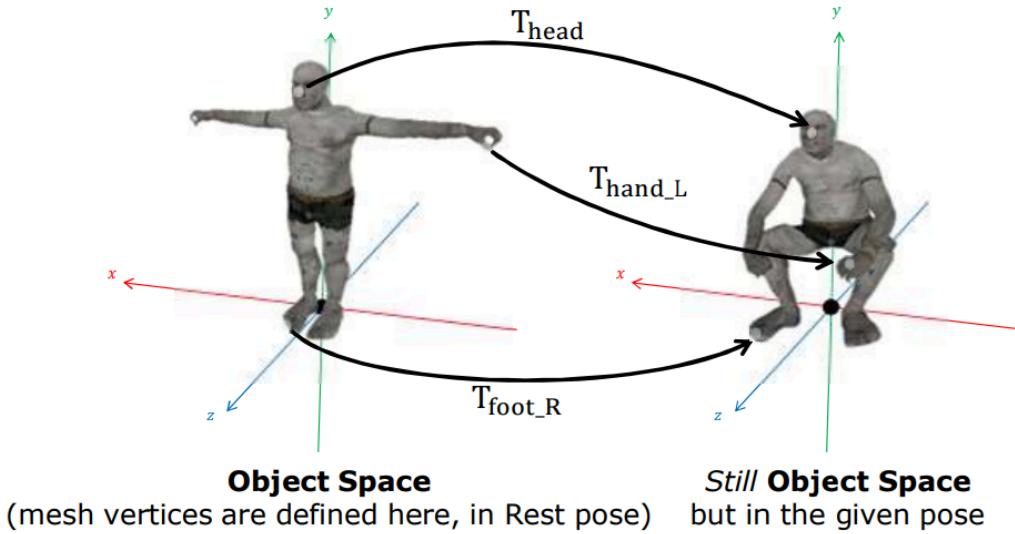
- from «right foot» space to «character» space
- uses the **Hierarchy** of the **Skeleton**
 - once computed, skeleton **hierarchy** no longer needed

◆ Final Transform: $P_2 P_7 P_8 R_8^{-1} R_7^{-1} R_2^{-1}$

- from «character» space in rest pose to «character» space in dest. pose
 - uses the **Rest Pose** of the **Skeleton** ($R_1 \dots R_N$)
 - once this is computed, **Rest Pose** is no longer needed
- mesh (vertices normals...) is defined in this space!

La trasformazione finale è data dalla composizione di tutte queste trasformazioni, trasferita sulla superficie vedremo in che modo. Se prendiamo un punto sulla

faccia, abbiamo la trasformazione che porta il nodo dalla posizione originale a quella dopo la trasformazione.



Per salvare in memoria questa cosa andiamo a salvare per ogni osso le trasformazioni che lo portano nella configurazione dello scheletro in quella posa, quindi abbiamo una trasformazione per ogni osso.

- ◆ **pose** = array of (local) transforms
 - it's defined for one given skeleton
 - RAM cost: **n_bones x bytes_for_a_transform**



Bone <i>i</i>	Trasform[<i>i</i>]
#0 (pelvis) [root]	L[0]
#1 (spine)	L[1]
#2 (chest)	L[2]
#3 (shoulder sx)	L[3]
...	...
#10 (calf)	L[10]
...	...

Local Transform

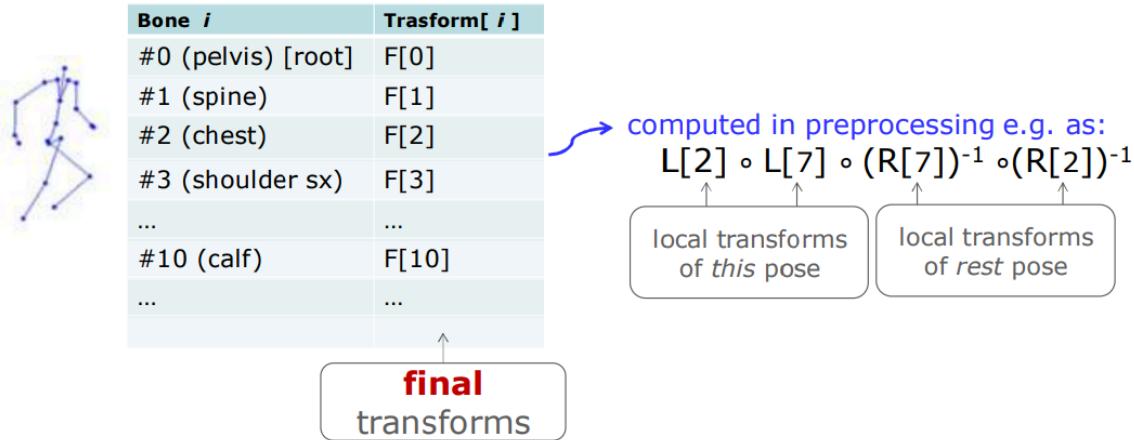
It includes:

- a **Rotation**: always!
- a **Translation**: maybe
If not, use the one defined in the **rest pose** of the skeleton.
==> a pose cannot redefine bone *lengths*.
- a **Scaling**: usually not
A joint cannot enlarge a part of the character

Potrebbe essere utile, al posto di salvare la trasformazione singola di quella posa, salvare direttamente la composizione con l'inversa della trasformazione della rest

pose.

- ◆ pose = array of **final** transforms
 - it's defined for one given skeleton (RAM cost: $n_{\text{bones}} \times \text{bytes_for_a_transform}$)



Se voglio salvare un'animazione, avrò un vettore di pose, dove ogni posa costituisce un keyframe.

- ◆ 1D Array of **poses** (1 pose = 1 keyframe)

- RAM cost:
 $(\text{num keyframes}) \times (\text{num bones}) \times (\text{transform size})$
- Each pose assigned to time dt
 - delta from start of animation t_0
- Sometime, looped
 - interpolation 1st keyframe with last

