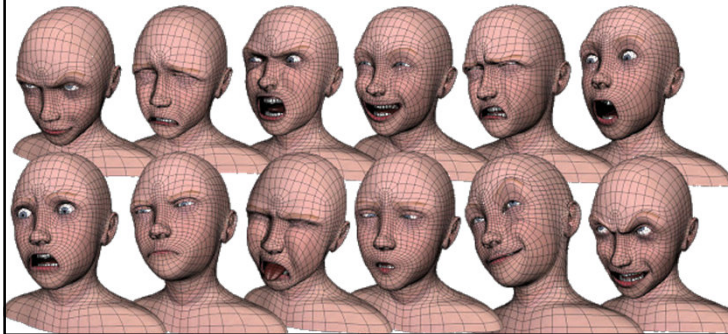


Master Game Dev 2014/2015

Game Engines

## Animations

Marco Tarini



## Animazioni nei games

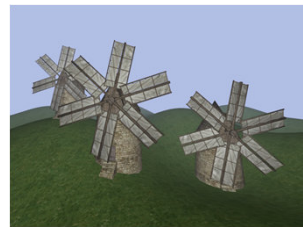
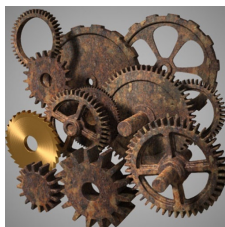
- Scripted
  - **Un assets!**
  - Controllo da parte degli artisti / creatori (dramatic effects!)
  - Non interattiva
  - Realismo... dipende dall'artista
  - Poca customizzabilità
- Computed / procedural
  - **Physic engine**
  - Poco controllo
  - Interattiva
  - Realismo come prodotto collaterale del rispetto leggi fisiche
  - Si autoadatta

## Animazioni nei games

- O... miste
  - primary animations: scripted  
secondary animations: computed
  - alive characters: scripted  
dead characters: computed (ragdolls)
  - ...

## Tipi di animazioni scripted

- 1. di oggetti composti di parti rigide
  - [animazione di trasformaz di modellazione](#)
  - (anche con giunti: robot, macchine...)



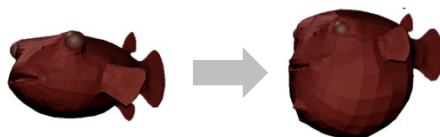
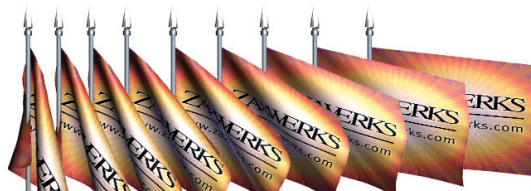
## Tipi di animazioni scripted

- 2. di oggetti deformabili articolati
  - con scheletro interno
  - es: esseri umani
  - o la maggior parte dei virtual characters!  
e.g. games
  - “skinning” / o “rigging”



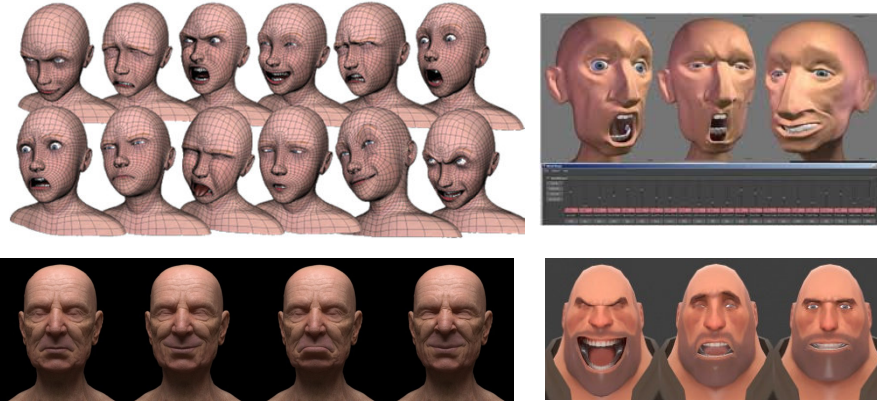
## Tipi di animazioni scripted

- 3. di oggetti deformabili generici
  - “per-vertex animations” / “blend shapes” / “morph targets”
  - es...



## Tipi di animazioni scripted

- 3. di oggetti deformabili generici
  - “per-vertex animations” / “blend shapes” / “morph targets”
  - ma anche: volti



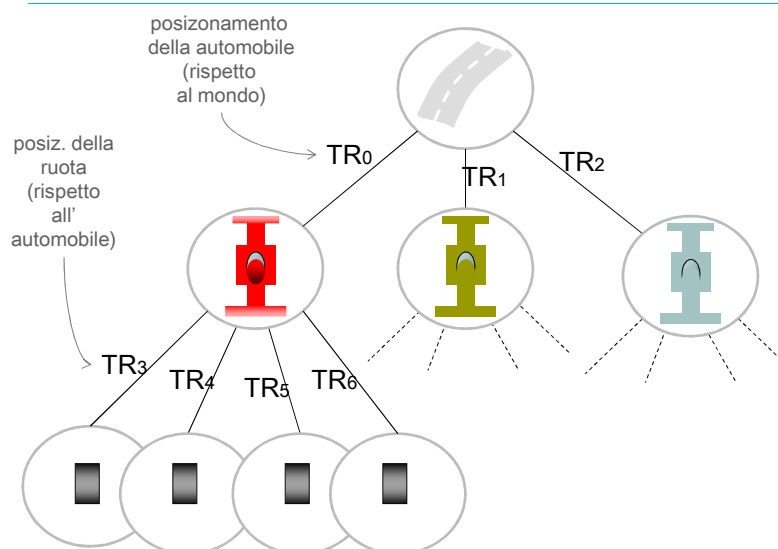
## Tipi di animazioni scripted

- di oggetti composti di parti rigide
  - anche con giunti: robot, macchine...
  - → animazioni “cinematiche” / “forward kinematics animations” (mutamenti delle trasformaz di modellazione)
- di oggetti deformabili articolati
  - con scheletro interno
  - es: esseri umani, virtual characters, animali...
  - (la maggior parte dei virtual characters! e.g. games)
  - → “skinning” / “rigging”
- di oggetti deformabili generici
  - es: volti, un ombrello che si apre, cose con membrane...
  - “per-vertex animations” / “blend shapes” / “morph targets”

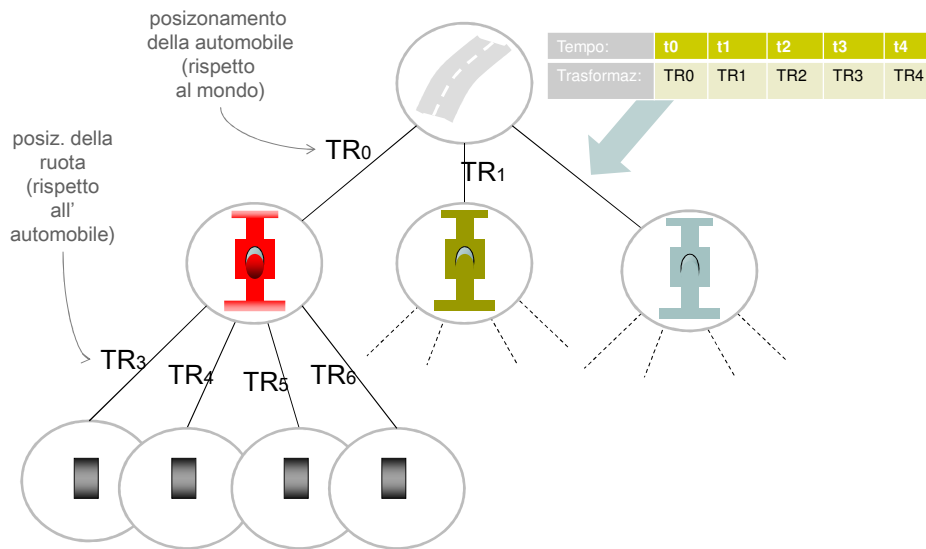
## Tipi di animazioni scripted

- di oggetti composti di parti rigide
  - anche con giunti: robot, macchine...
  - → animazioni “cinematiche” / “forward kinematics animations” (mutamenti delle trasformaz di modellazione)
- di oggetti deformabili articolati
  - con scheletro interno
  - es: esseri umani, virtual characters, animali...
  - (la maggior parte dei virtual characters! e.g. games)
  - → “skinning” / “rigging”
- di oggetti deformabili generici
  - es: volti, un ombrello che si apre, cose con membrane...
  - “per-vertex animations” / “blend shapes” / “morph targets”

## Scene graph

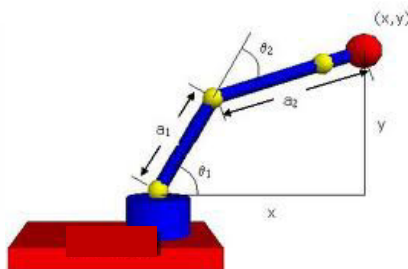


## Scene graph... animato (animazioni “cinematiche”)



## Cinematica diretta Cinematica inversa

- Tipici task della robotica



### Forward kinematics:

«da angoli (e lunghezze) a  $(x,y)$ »  
banale!  
soluz univoca

VS

### Inverse kinematics:

«da  $(x,y)$  (e lunghezze), a angoli»  
non banale!  
soluzioni multiple  
(task: scegliere quella “migliore”)

## Animazioni cinematiche: come

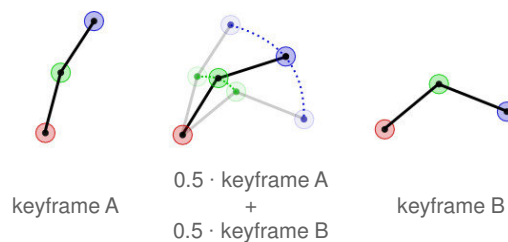
- modo 1:
  - scripting
- modo 2:
  - editing is software di animaz.
    - cinema 4D, blender, 3D max, ...
    - (possono usare I.K.)
  - esportare animaz
    - come sequenza di **keyframes**
    - Files: collada, fbx, ...

Tempo:	t0	t1	t2	t3	t4
Arco A:	TR0	TR1	TR2	TR3	TR4
Arco B:	TR0	TR1	TR2	TR3	TR4



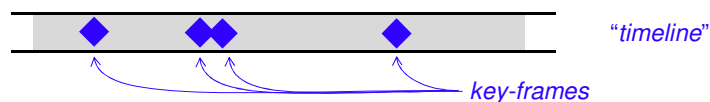
## Interpolazione di keyframes (concetto generico)

- Keyframes  
+  
inerpolazione



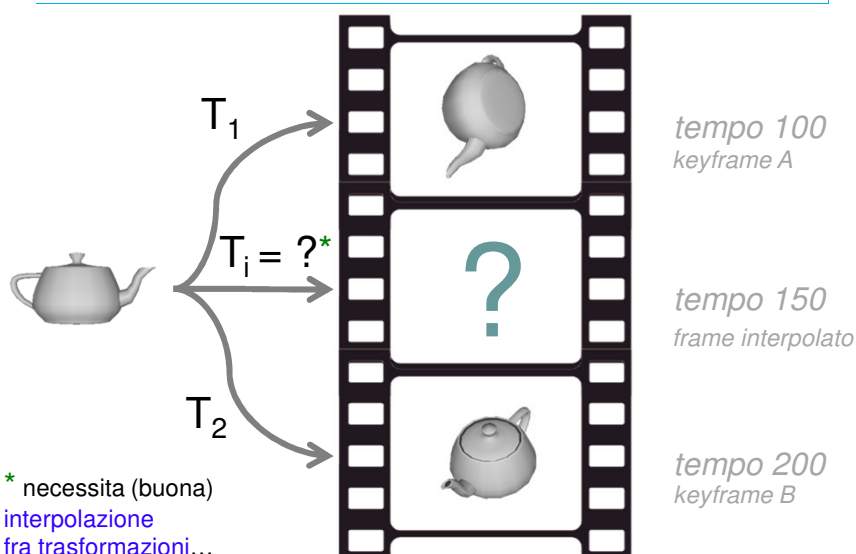
## Interpolazione di keyframes (concetto generico)

- Modellatore / animatore decide:
  - set di keyframes
  - con tempi associati



- La maggior parte **frames** effettivi: **interpolazioni** fra **keyframes**
  - risparmio di workload dell'artista
  - risparmio di storage
- nota: distribuzione keyframes *adattiva*
  - più keyframes dove necessario

## Interpolazione di keyframes (caso animazioni cinematografiche)





## Tipi di animazioni scripted

- di oggetti composti di parti rigide
  - anche con giunti: robot, macchine...
  - → animazioni “cinematiche” / “forward kinematics animations”  
(mutamenti delle trasformaz di modellazione)
- di oggetti deformabili articolati
  - con scheletro interno
  - es: esseri umani, virtual characters, animali...
  - (la maggior parte dei virtual characters! e.g. games)
  - → “skinning” / “rigging”
- di oggetti deformabili generici
  - es: volti, un ombrello che si apre, cose con membrane...
  - “per-vertex animations” / “blend shapes” / “morph targets”



## Animaz. oggetti deformabili: Blend shapes

- A.K.A:
  - Blend shapes
  - Per-vertex animations
  - Face morphs
  - Shape keys
  - Morph targets
  - ...

## Blend shapes: concept



Walk cycle  
(Monkey Island  
LucasArt 1991)

- Animaz 2D (vecchia scuola):  
sequenza di sprites
- Animaz 3D:  
sequenza di mesh?

## Come rappresento una mesh? (quali strutture dati)

- Modo **indexed**:
  - Geometria: array di vertici
    - in ogni vertice, posizione e attributi
  - Attributi:
    - coi vertici
      - (e.g. campi della classe "vertice")
  - Connettività:
    - Array di triangoli
    - Per ogni triangolo:
      - tripletta di **indici** a vertice

## Blend shapes

### (struttura dati)

connettività (es: *indexed*)

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

geometria:

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

attributi:

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

## Blend shapes

### (struttura dati)

connettività (es: *indexed*)

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

geometrie:

Vert:	Base Shape	Shape 1	Shape 2	...
V1	(x,y,z)	(x,y,z)	(x,y,z)	...
V2	(x,y,z)	(x,y,z)	(x,y,z)	...
V3	(x,y,z)	(x,y,z)	(x,y,z)	...
V4	(x,y,z)	(x,y,z)	(x,y,z)	...
V5	(x,y,z)	(x,y,z)	(x,y,z)	...

attributi:

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

## Blend shapes

- Una mesh con **geometria** ripetuta varie volte
  - Cioé sequenza di mesh ('**shape**') con
    - **connettività**: condivisa
    - **attributi**: condivisi
      - (eccetto normali / direz tangenti)
    - **geometria**: ripetuta x ogni 'shape'
    - (e **tessiture**: condivise)
  - Varianti (equivalenti):
    - Modo **relativo**:
      - *base shape*: memorizzato esplicitamente
      - ogni altro *shape*: come differenza (x,y,z) con *base shape*
    - Modo **assoluto**:
      - ogni *shape* memorizzato indipendentemente
- o '**morph**'  
o (key)-'**frame**'  
o '**shape-key**'

## Blend shapes (struttura dati)

- Mesh **indexed**:

```
class Vertex {
    vec3 pos;
    rgb color;
    vec3 normal;
};

class Face{
    int vertexIndex[3];
};

class Mesh{
    vector<Vertex> vert; /* geom + attr */
    vector<Face> tris; /* connettivita' */
};
```

## Blend shapes (struttura dati)

- Mesh **indexed**:

```
class Vertex {  
    vec3 pos [ N_SHAPES ] ;  
    rgb color;  
    vec3 normal [ N_SHAPES ] ;  
};  
  
class Face{  
    int vertexIndex[3];  
};  
  
class Mesh{  
    vector<Vertex> vert; /* geom + attr */  
    vector<Face> tris; /* connettivita' */  
};
```

## Blend-shapes: file formats più diffusi

- Semplici:
  - **.MD5** ("quake", valve)
  - o, sequenza di mesh separate (es **.OBJ**)
    - occhio! connettività coerente?  
(vertex ordering, face ordering, wedge ordering)
- Complessi:
  - **.DAE** (Collada)
  - **.FBX** (Autodesk)

## Interpolazione fra 2 shapes

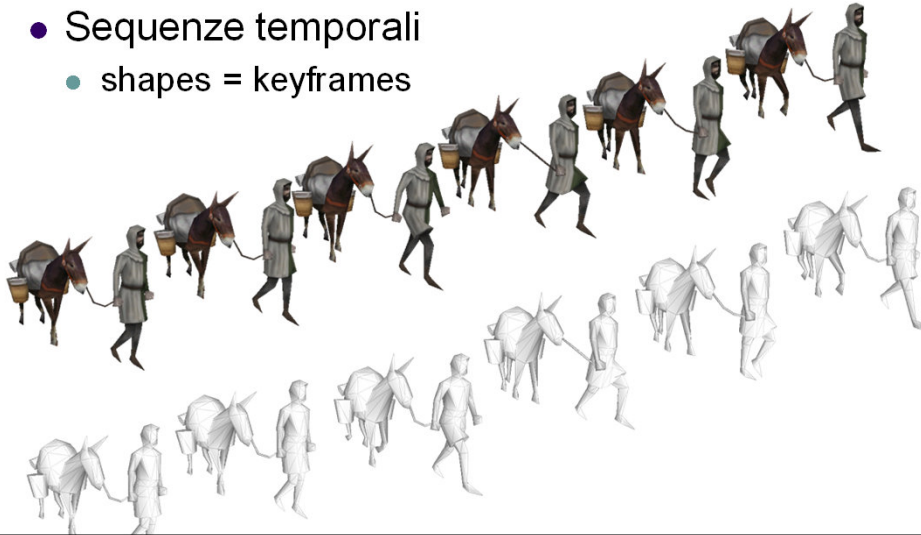
- Geometria interpolata:
  - Assoluto:  
 $\text{shape}_A \cdot w_A + \text{shape}_B \cdot w_B$
  - Relativo:  
 $\text{shape}_{base} + \text{shape}_A \cdot w_A + \text{shape}_B \cdot w_B$
- con:  $0 \leq w_A \leq 1$   
 $0 \leq w_B \leq 1$   
 $w_A + w_B = 1$

## Interpolazione fra shapes



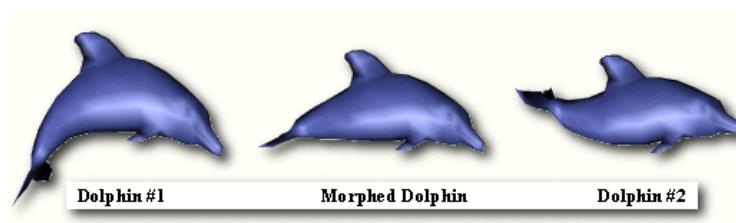
## Usi delle Blend shapes

- Sequenze temporali
  - shapes = keyframes







## Usi delle Blend shapes

- Sequenze temporali
  - blends = keyframes



## Usi delle Blend shapes: sequenze temporali

- Blend shape per animaz

- $\text{shape}_A$   – tempo  $t_A$
- $\text{shape}_B$   – tempo  $t_B$
- $\text{shape}_C$   – tempo  $t_C$
- $\text{shape}_D$   – tempo  $t_D$





- attimo attuale:  $t$  con  $t_B < t < t_C$

- interpolare fra le shape:  $\text{shape}_B$  e  $\text{shape}_C$

- con pesi  $w_B = \frac{t-t_C}{t_B-t_C}$   $w_C = (1 - w_B) = \frac{t-t_B}{t_C-t_B}$

## Usi delle Blend shapes: sequenze temporali

- Blend shape per animaz

- $\text{shape}_A$   – tempo  $t_A$
- $\text{shape}_B$   – tempo  $t_B$
- $\text{shape}_C$   – tempo  $t_C$
- $\text{shape}_D$   – tempo  $t_D$

- attimo attuale:  $t$  con  $t_B < t < t_C$

- interpolare fra le shape:  $\text{shape}_B$  e  $\text{shape}_C$

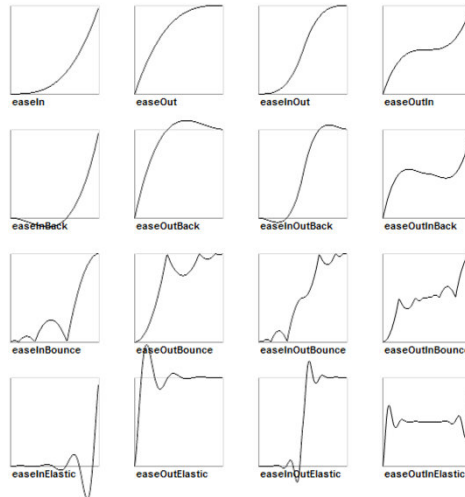
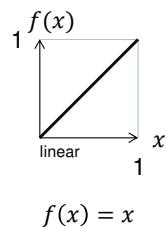
- con pesi  $w_B = f\left(\frac{t-t_C}{t_B-t_C}\right)$   $w_C = f\left(\frac{t-t_B}{t_C-t_B}\right)$

↑  
transition function



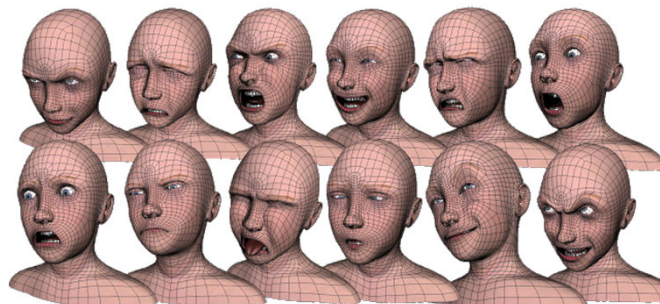
## Transition functions

- Concetto generale nelle animaz
- Non solo Linear



## Usi delle Blend shapes

- Animazioni facciali
  - (forse l'uso più comune)



Spesso congiunta con animaz skeletal  
(mandibola, collo, rotaz bulbi oculari)

## Interpolazione fra più shapes

- Geometria interpolata:

- Assoluto:

$$\text{shape}_A \cdot w_A + \text{shape}_B \cdot w_B + \text{shape}_C \cdot w_C + \dots$$

- Relativo:

$$\text{shape}_{base} + \text{shape}_A \cdot w_A + \text{shape}_B \cdot w_B + \text{shape}_C \cdot w_C + \dots$$

- con:

$$0 \leq w_{A,B,C, \dots} \leq 1$$

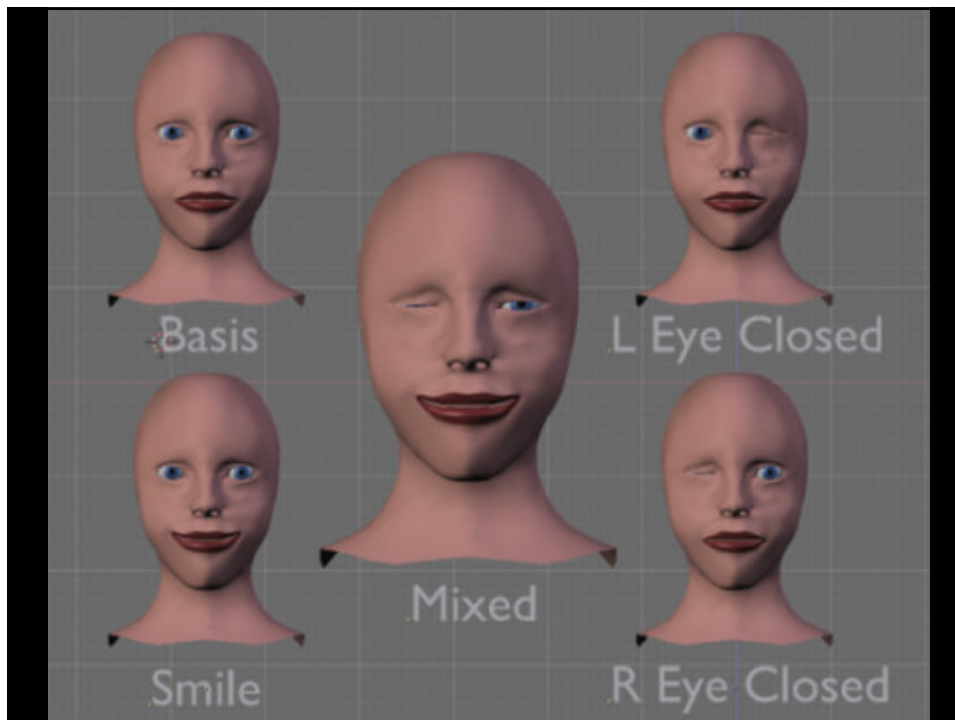


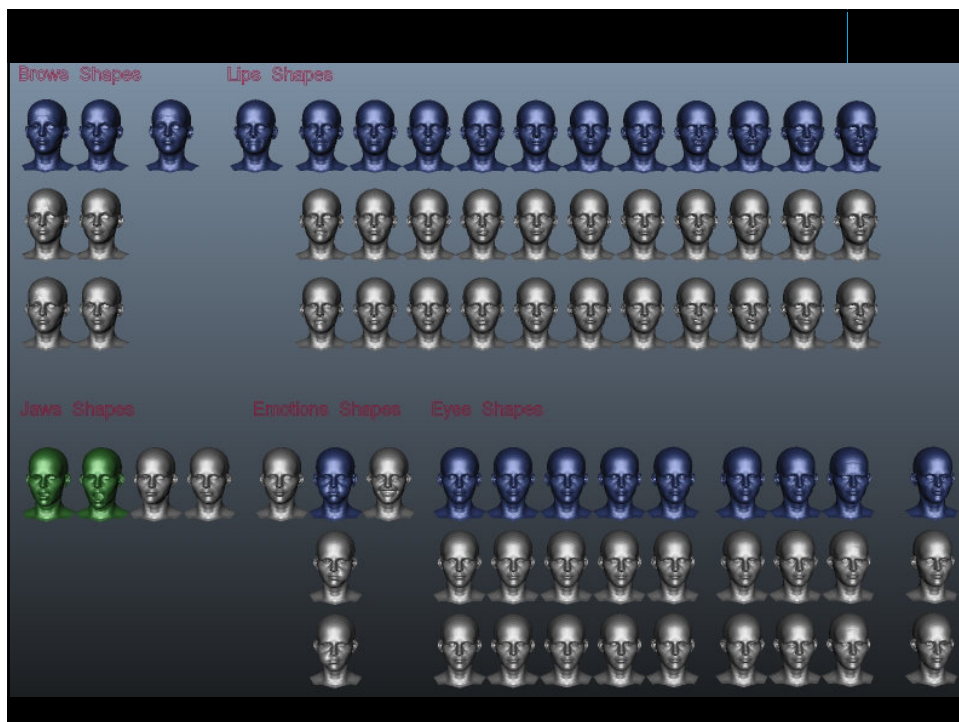
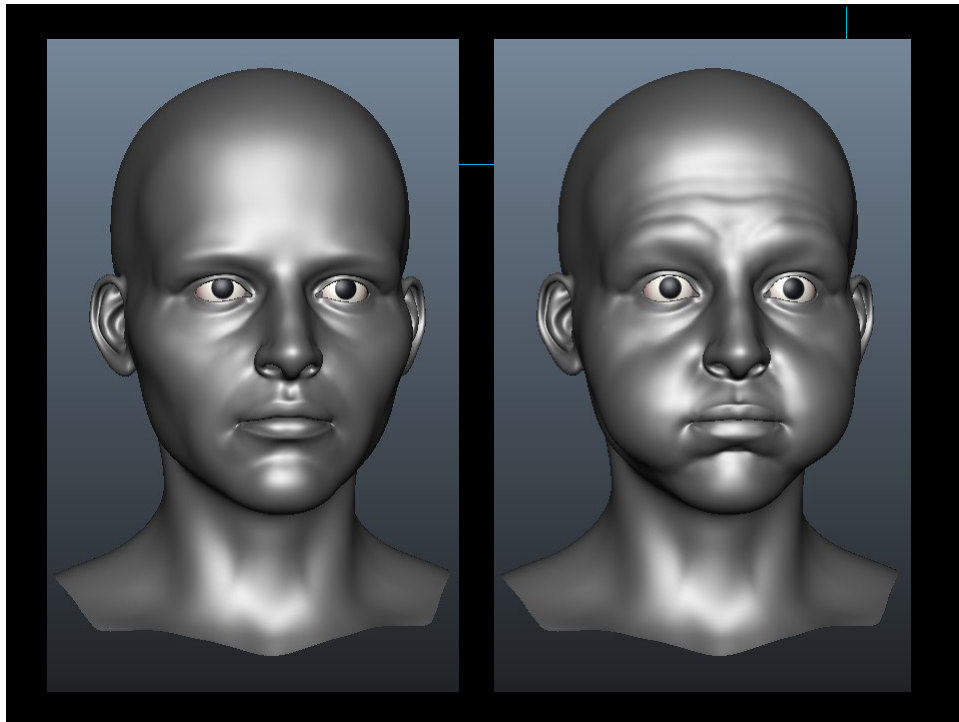
o anche no (estrapolazione).  
Utile per...

$$w_A + w_B + w_C + \dots = 1$$



o anche no (**solo in relativo**)  
Utile per...





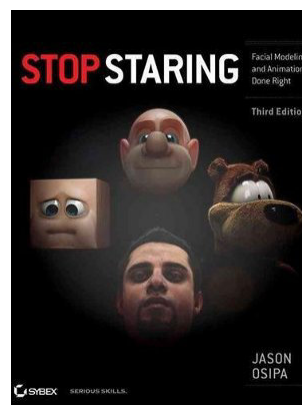
## Blend shapes per animazione facciale: workflow

- Modellatore 3D fa:  
il set di blend-keys
- Animatore (delle espressioni) decide:  
i pesi
  - tipicamente: attraverso slider
  - aiutato / sostituito da software
    - lip sync
    - espressioni dinamiche e.g. in games
- Blending dei keys: in tempo reale

[VIDEO]

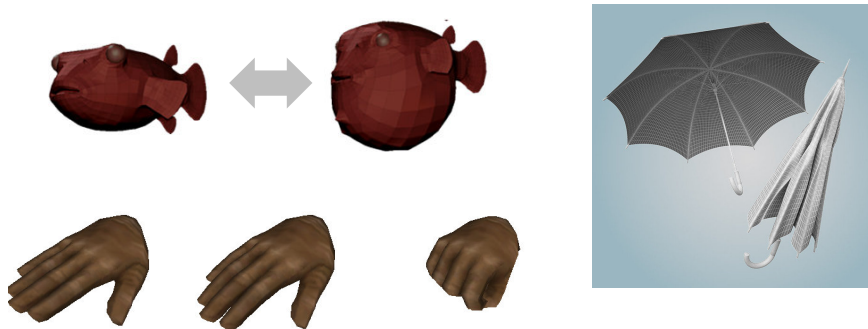
## Come si ottengono

- Manuale per animazione blend-shape di volti:
  - “Stop Staring” (3d edition)  
Jason Osipa
  - Stile, espressione...
  - Non tecnico (alto livello)
  - No strumenti specifici e.g. Blender, Maya



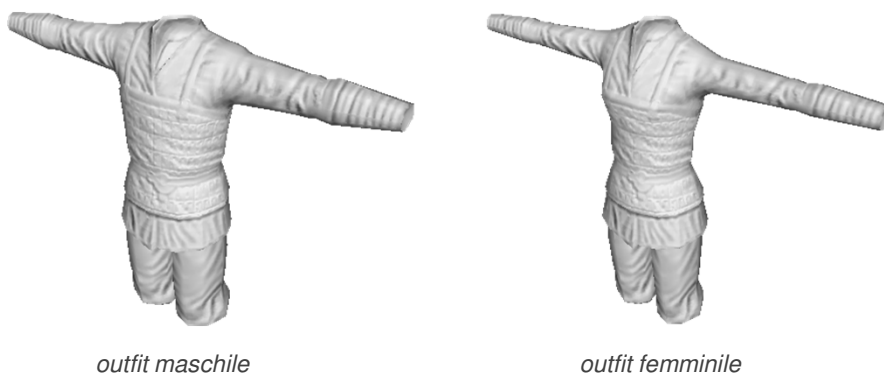
## Usi delle Blend shapes

- Configurazioni di oggetti deformabili
- Pose «precotte»



## Usi delle Blend shapes

- Varianti di uno stesso oggetto
  - (miscelabili!)



## Usi delle Blend shapes

- Varianti di uno stesso oggetto
  - (miscelabili!)



umano



orco



goblin



nano

## Usi delle Blend shapes

- Definire morfologia di una classe di oggetti
  - definire una forma = specificare i pesi
    - modellazione 3D “ad alto livello”
  - i pesi “spannano” uno **spazio delle forme**
    - una forma = un punto dello spazio
    - pesi = coordinate
  - uno spazio è tanto più utile quanto più :
    - *tutte e sole* le forme “ragionevoli” sono punti dello spazio
- Es: (ricorrente): morfologie facciali
  - “face-space”
  - nota: morfologia facciale ≠ espressione facciale

## Usi delle Blend shapes

- Una **blend shape** per un **face space** ("face-morphs")



## Cosa una blend shape **non** fa

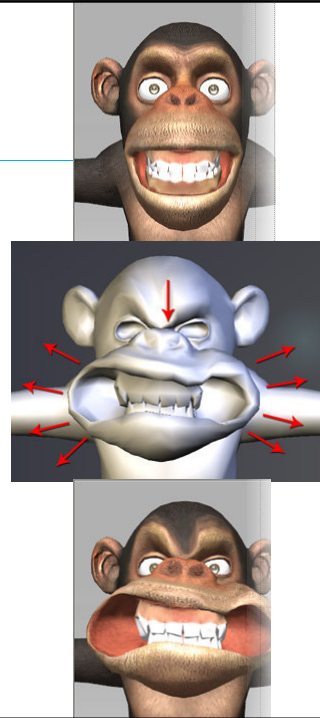
- Cambi di connettività
  - es di risoluzione, remeshing
- Cambi topologici
  - rottura, separazione, fusione di pezzi
- Cambi di attributi
  - (es mutamenti di colore...)
- Cambi di textures
  - usare invece una **texture animation**, maybe?



## Blend shapes: come si ottengono

Manualmente:

- 1. Editare la base shape
  - compreso:  
uv-mapping, texturing, etc.
- 2. Ri-editare la base shape  
x ogni shape-key!  
...ma preservando:  
connettività,  
textures, etc
  - low poly editing
  - subdivision surfaces
  - parametric surfaces...
  - sculpting: difficilmente.




## Blend shapes: come si ottengono

- Cattura:
  - acquisizione della base shape
  - (opz: semplificaz remeshing uv-mapping etc)
  - cattura dei frame successivi
    - e.g. kinect, o scansione di ogni frame
  - morphing dei frame successivi
    - “non rigid mesh alignment”

[VIDEO]

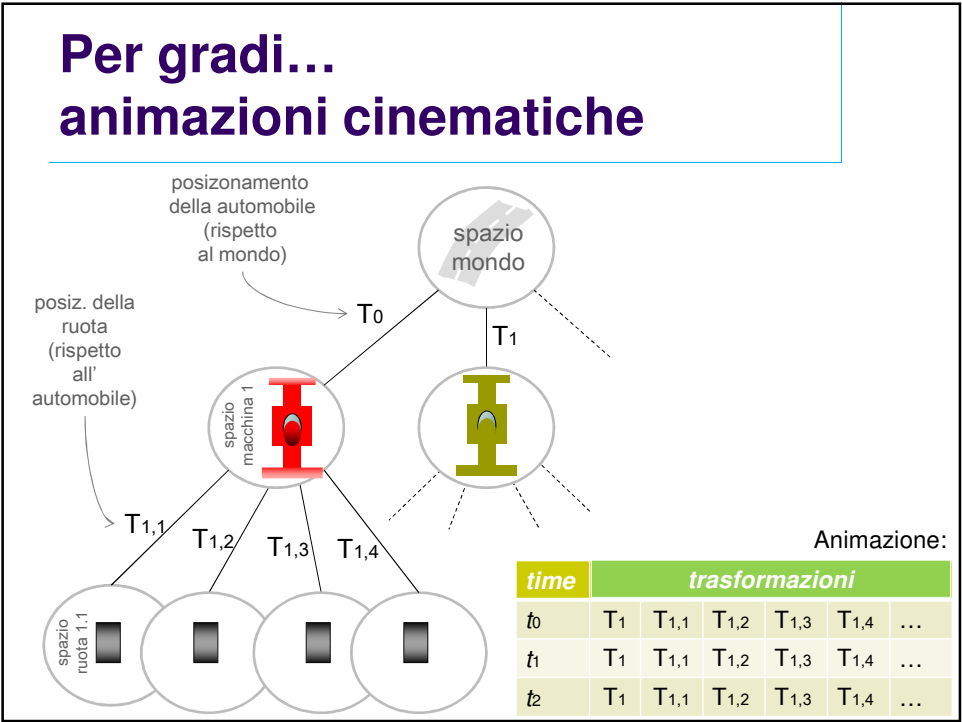
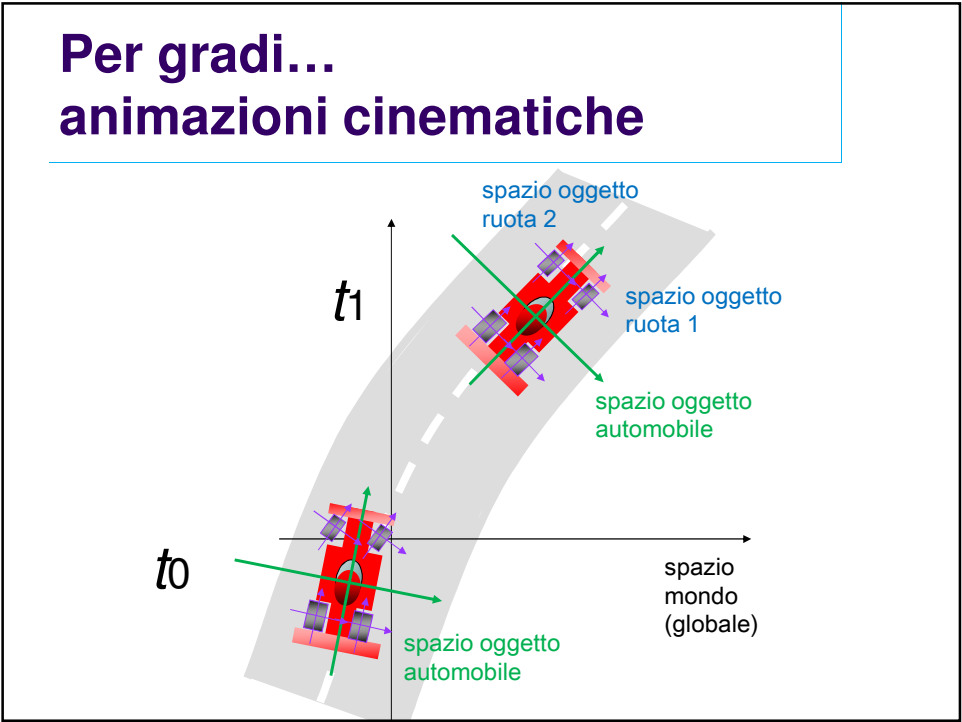


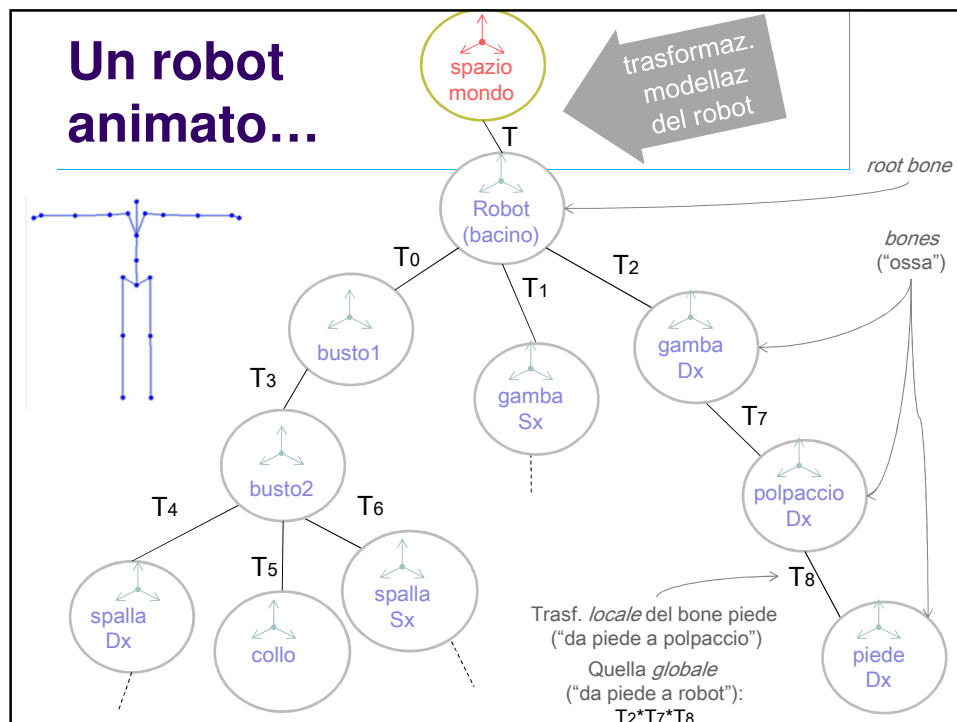
## Blend shapes: vantaggi e svangaggi

- ☺ Flessibili, espressive, molta libertà... troppa?
    - ☹ Consumo memoria
    - ☹ Work intensive da costruire
- (anche se non tanto quanto, e.g., sprites,
- 
- xchè frames riutilizzano: connettività attributi tessiture...)
- ☺ Facili da usare una volta costruite
    - solo definiz pesi (e.g. sliders)

## Tipi di animazioni scripted

- di oggetti composti di parti rigide
  - anche con giunti: robot, macchine...
  - → animazioni “cinematiche” / “forward kinematics animations” (mutamenti delle trasformaz di modellazione)
- di oggetti deformabili articolati
  - con scheletro interno
  - es: esseri umani, virtual characters, animali...
  - (la maggior parte dei virtual characters! e.g. games)
  - → “skinning” / “rigging”
- di oggetti deformabili generici
  - es: volti, un ombrello che si apre, cose con membrane...
  - “per-vertex animations” / “blend shapes” / “morph targets”





## Scheletro (o rig)

- Struttura gerarchica di **ossa**
- **Ossso**:
  - **Spazio vettoriale** in cui sono espressi alcuni pezzi del personaggio (l'oggetto animato)
  - Es, in un umanoide: *braccio, avambraccio, bacino, ...*
  - (non centra il significato biologico di "osso")
- Spazio vettoriale del **root bone** = **spazio oggetto** (del personaggio)
- Scheletro di un personaggio umanoide medio: 20-40 ossa (tipicamente)

## Da assemblaggio di pezzi...

- Fin qui: una mesh per ogni osso
  - (es, carlinga, ruota)
- Ok per oggetti meccanici con strutture semplici
  - (macchina, braccio meccanico con 2 giunti...)
- Ma, per un “robot” di ~20-40 ossa?
  - Mesh separate per braccia, avambraccia, falangi, falangine, falangette...
  - Assemblaggio con le matrici (dopo modellazione 3D)
  - → molto scomodo.



## ... a skinning di mesh

- Idea: *mesh skinned*
  - 1 sola mesh per tutto il personaggio
  - attributo per ogni vertice: indice di osso
  - un modello 3D animabile!
- Ortogonalità modelli / animazioni!
  - cioè:
    - ogni modello skinned: va su tutte le animazioni
    - ogni animazione: applicabile a tutti i modelli
  - (basta che si riferiscano ad uno stesso scheletro)
  - → 500 modelli e 500 animazioni = 1000 oggetti in RAM
    - invece di 500x500 combinazioni
- I task dell'artista digitale:
  - “*rigging*”: definizione dello *scheletro* (riggers)
  - definizione dello *skinning* su un modello (skinners)
  - “*animation*”: definizione delle animazioni (animators)

“*Skinning*”  
della mesh  
(qui, ad 1  
osso solo).

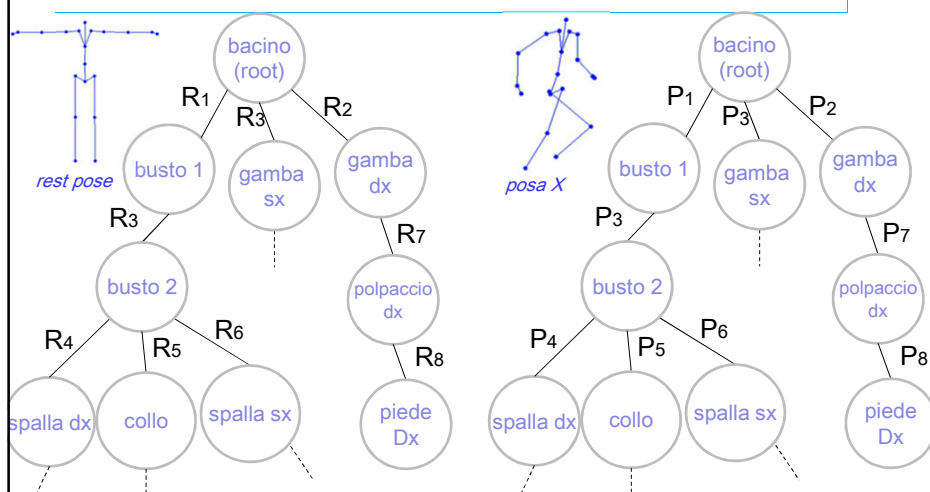
## Posa (frame di un'animazione scheletale)

- Definizione di una **trasformazione** per ogni **bone**
- Trasformazione locale**: (del **bone i**)
  - da**: spazio osso padre nella posa data
  - a**: spazio osso figlio nella posa data
  - usate per costruire la posa
- Trasformazione globale**: (del **bone i**)
  - da**: spazio osso **i** nella rest pose
  - a**: spazio osso **i** nella posa data
  - usate per memorizzare/applicare la posa

a volte, solo la componente "rotazione"

(allora, "ossa rigide": estensione osso costante per tutte le pose)

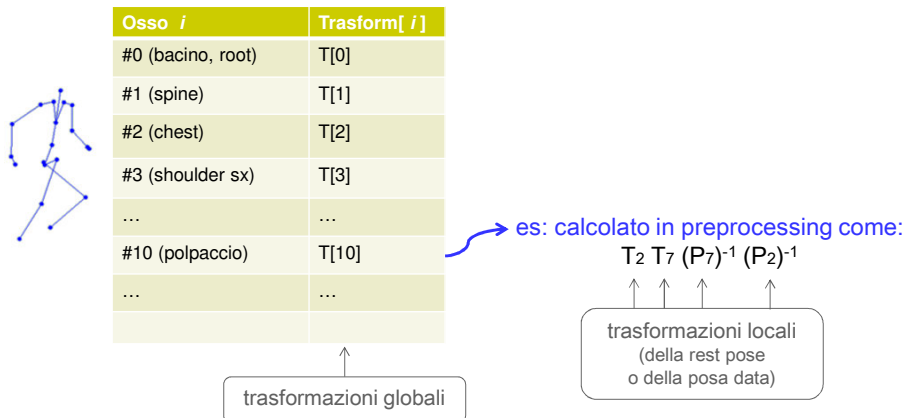
## Da rest pose a una posa data



trasf. globale polpaccio, da **rest pose** a **posa X** =  $P_2 P_7 (R_2 R_7)^{-1} = P_2 P_7 (R_7)^{-1} (R_2)^{-1}$

## Posa (per uno scheletro dato) struttura dati

- **posa** = semplice array di trasformazioni globali
  - costo in ram:  $n\_ossa \times bytes\_per\_trasf$
  - per 1 scheletro dato!



## Animazione skeletale

- Array 1D di keyframes (pose)
  - Costo RAM:  
(num key-frames) x (num ossa) x ( bytes x trasf)
  - Spesso tenuto in RAM scheda video
    - applicato da GPU direttamente a modelli skinned

## Da robot di pezzi, a oggetti deformabili



- Idea: più ossa per 1 vertice
- Trasformazione del vertice:
  - interpolazione delle trasformazioni associate alle ossa scelte
  - pesi interpolaz fissi (per quel vertice)
- Strutture dati: attributi X vertice
  - Per ogni vertice:
    - [ indice osso , peso ] x  $N_{max}$
    - (Tipicamente,  $N_{max} = 4$  o  $2$ , vedi dopo)

*"Skinning"*  
della mesh  
(a  $N_{max}$  ossa)

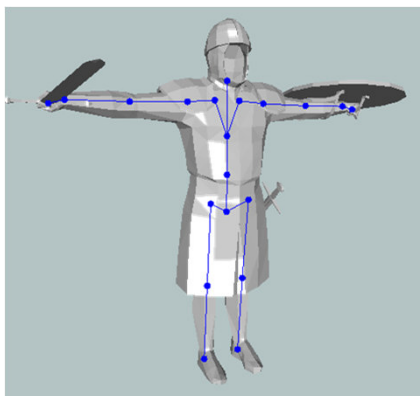
## Quanti link ad osso per vertice

- Dipende dal game engine!
- $N_{max}$  tipicamente:
  - 1 (sottopezzi rigidi)
  - 2 (cheap, e.g. su dispositivi mobili)
  - 4 (top quality – standard)
- Decrementare  $N_{max}$ 
  - in preprocessing  
(task per un game tool!)

## (perchè limite superiore a num link x verice?)

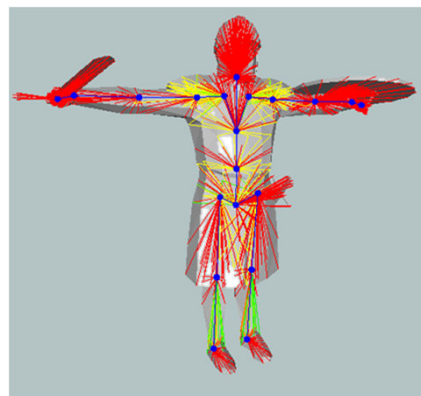
- Costo in performance
  - $N_{\max}$  trasformazioni da interpolare in GPU (x vert)
  - GPU = poco controllo:
    - interpolaz fra  $N_{\max}$  trasf (costante)
    - bones non utilizzati: peso 0
- Costo in RAM (della scheda video)
  - strutture dati semplici per GPU-RAM
  - array a lunghezza fissa:
    - $N_{\max}$  coppie (indice , peso)
    - anche dove, localmente, ne basterebbero meno (pesi 0)

## Rigging e skinning



### Rigging

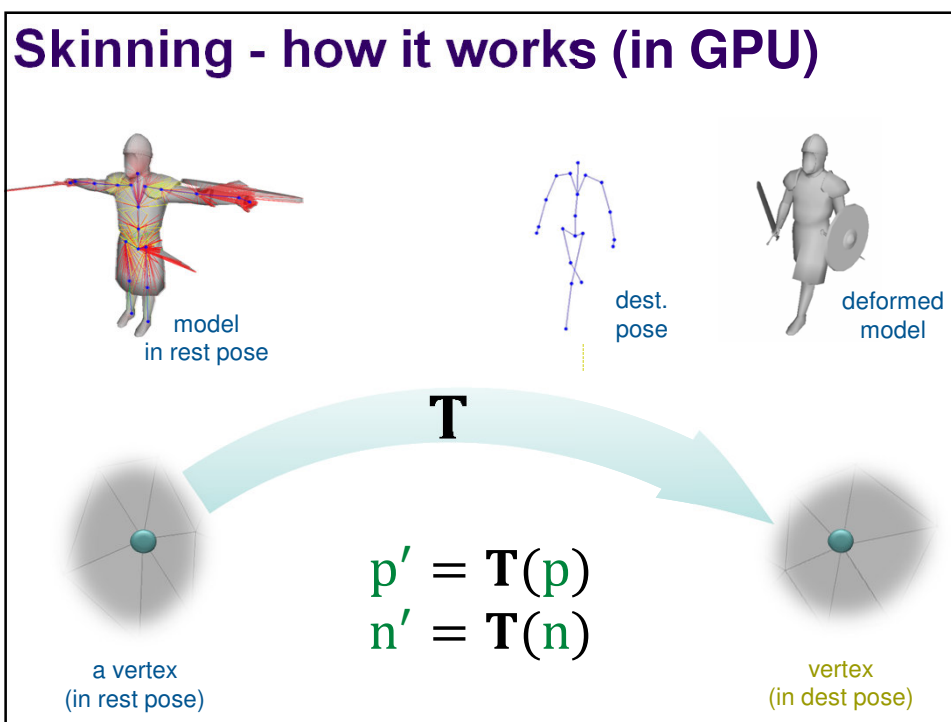
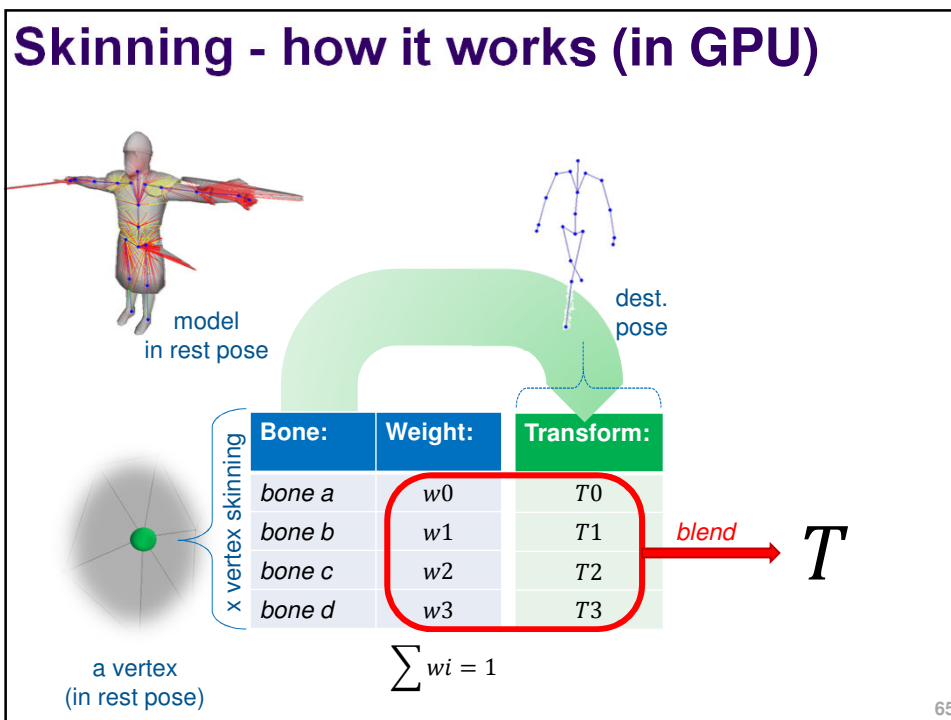
definizione dello scheletro  
(e spesso  
anche dei controlli usati per manipolarlo)

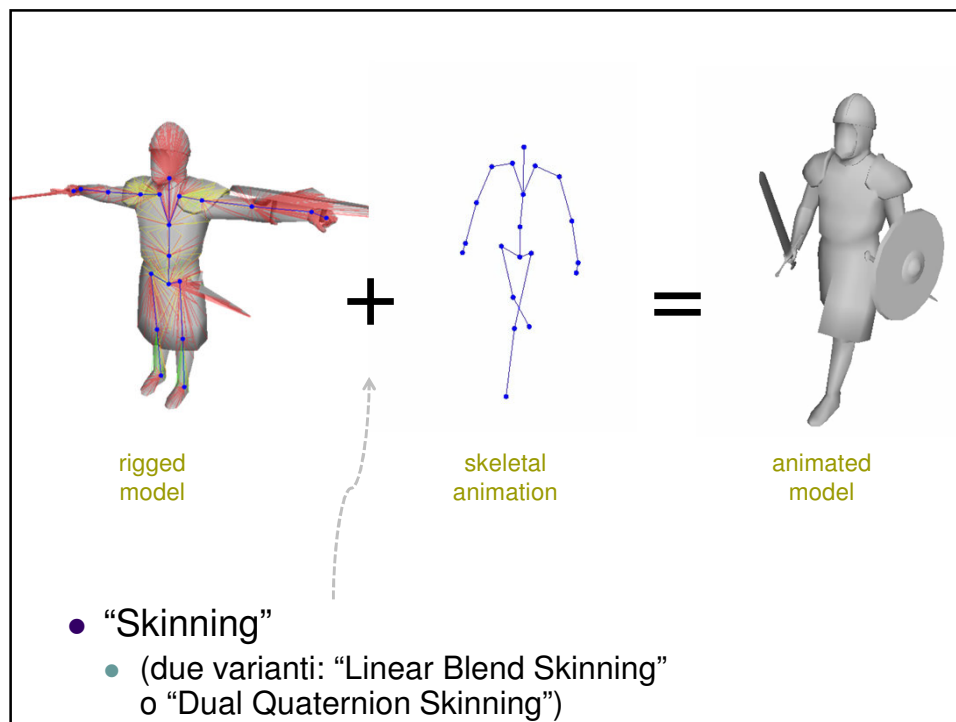


### Skinning

definizione delle link (pesati)  
fra vertici e ossa







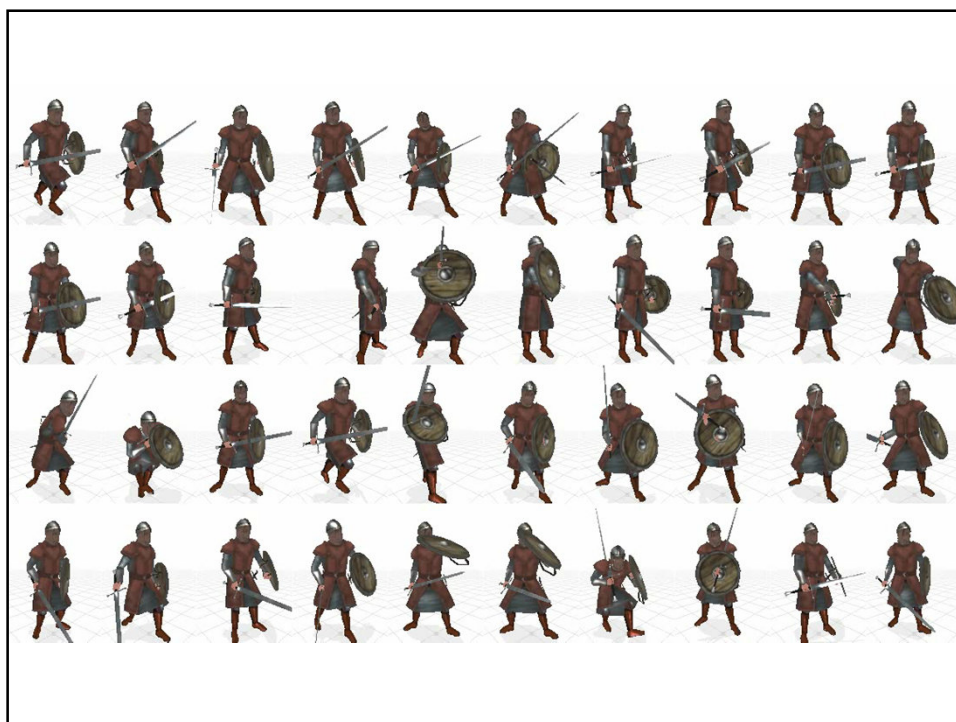
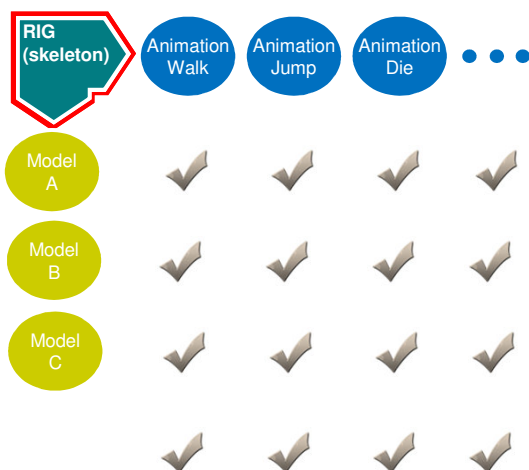
## Skeletal animations: tre strutture dati (riassunto)

- Scheletro (o rig)
  - Tree di **bones** (ossa)
  - Ogni **bone** => sistema di riferimento (in rest pose)
    - (sistema dell'osso root = sistema oggetto)
- Modelli 3D skinned
  - Mesh con associazione **vertici** => **bones**
  - Per vertice: [ **indice osso** , **peso** ] x N<sub>max</sub>
- Animazioni scheletrali
  - Sequenza di **pose**
  - Posa = trasformazione globale  $\forall$  **bone**

possibile formati file (per tutti e tre i dati):

- **.SMD** (Valve), **.FBX** (Autodesk), **.BVH** (Biovision)

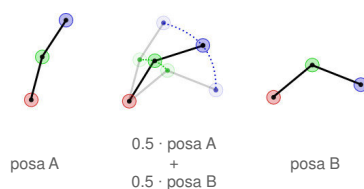
## Ortogonalità animazioni / modelli





## Interpolazione pose

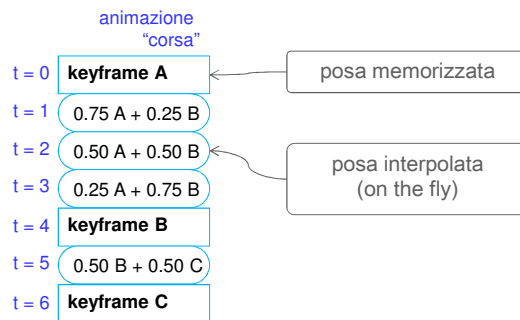
- Due pose si possono interpolare!



- basta interpolare le trasformazioni che le compongono

## Interpolazione pose (a runtime): keyframes

- Per comprimere animazioni

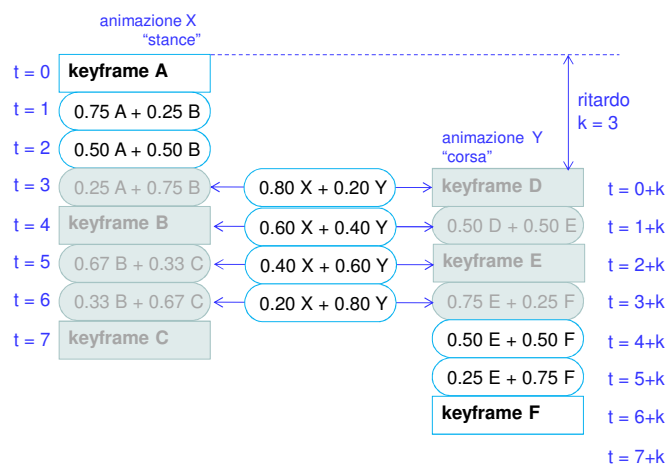


## Compressione animazioni

- Obiettivo: rimuovere keyframes
  - quelli "inutili"
  - in preprocessing ([game tools!](#))
- Versione 0.0 di un algoritmo:
  - per ogni keyframe
    - rimuovi keyframe  $P_x$
    - computa versione interpolata  $P_i$  dai keyframe rimanenti
      - (il precedente e il successivo)
    - se  $distanza(P_i, P_x) > ErrMax$  allora reinserisci keyframe  $P_x$

## Interpolazione pose (a runtime): transizioni fra animazioni

- Es: da stance a corsa



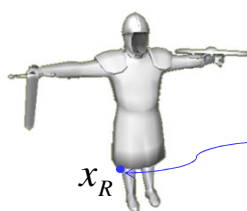
## Applicare le pose: *Linear Blend Skinning*

- Per ogni vertice della mesh:

$$x_P = \left( \sum_{i=1}^{N_{\max}} w_i T[b_i] \right) (x_R)$$

interpolaz. delle  
trasformaz.  
legate dal rigging  
al vertice

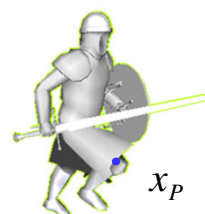
computaz a  
bordo della  
scheda video  
(in GPU)



posiz del vertice  
definito nella rest pose.

Skinning  
del vertice  
(con  $N_{\max} = 4$ ):

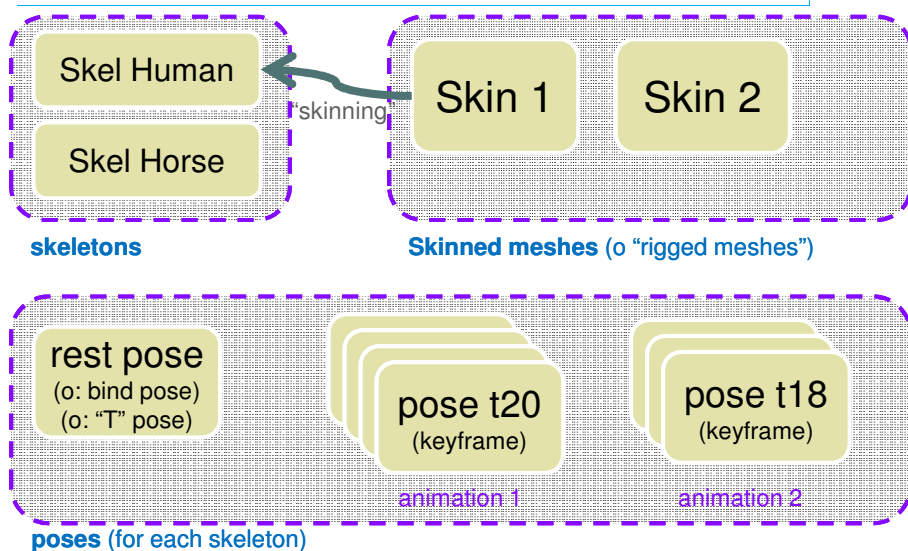
- $(b_1, w_1)$
- $(b_2, w_2)$
- $(b_3, w_3)$
- $(b_4, w_4)$



## Applicare le pose: *Dual Quaternion Skinning*

- Variante in cui le trasformazioni sono isometrie storate e interpolate come **dual quaternion**
  - in teoria, migliore qualità
    - (quaternioni interpolano meglio le rotazioni)
  - costo in GPU maggiore
    - (operazioni necessarie per vertice ~ x3)
- LBS o DQS?
  - scelta del game engine

## Assets di un modello animato: (riassunto)

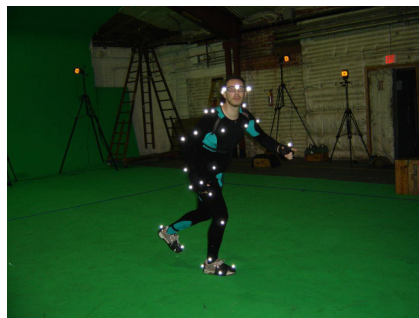


## Rigging e skinning di modelli 3D

- **rigging** e **skinning** di una mesh:
  - **rigging**: definizione di uno scheletro per quella mesh
  - **skinning**: definizione associazione vertice ossa
  - Due dei task dei modellatori digitali!
    - "skimmers" e "riggers"
    - aiutati (o sostituito) da strumenti automatici
- Animatori digitali
  - Definizione delle **animazioni** (scheletrali)
- Strutture dati: attributi X vertice
  - Per ogni vertice:
    - [ indice osso , peso ] x  $N_{max}$
    - (Tipicamente,  $N_{max} = 4$  o  $2$ )

## Animazioni scheletrali (scripted): come si ottengono

- Editing manuale
  - Animatori digitali
  - help da: IK, simulazioni fisiche ...
- Motion capture:





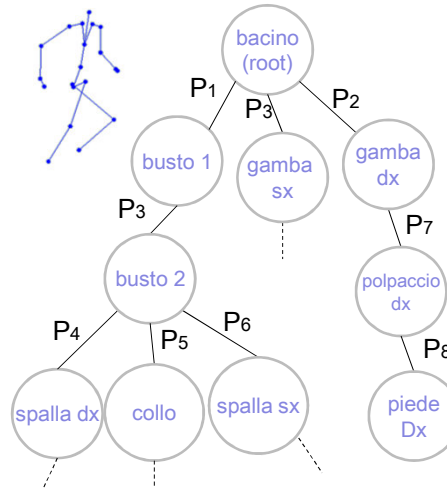
## Inverse Kinematic (IK): un tool utile

- Cinematica **diretta**:

- “date le trasf locali  $P_1, P_2 \dots P_N$ , (es, come angoli) dove finisce il piede?”
- (univoco, banale)

- Cinematica **inversa**:

- “se voglio mettere il piede nella pos p, quali  $P_1, P_2 \dots P_N, ?$ ”
- (+ vincoli, es. DoF giunti)
- (non univoco, non banale)



## Inverse Kinematic (IK): un tool utile

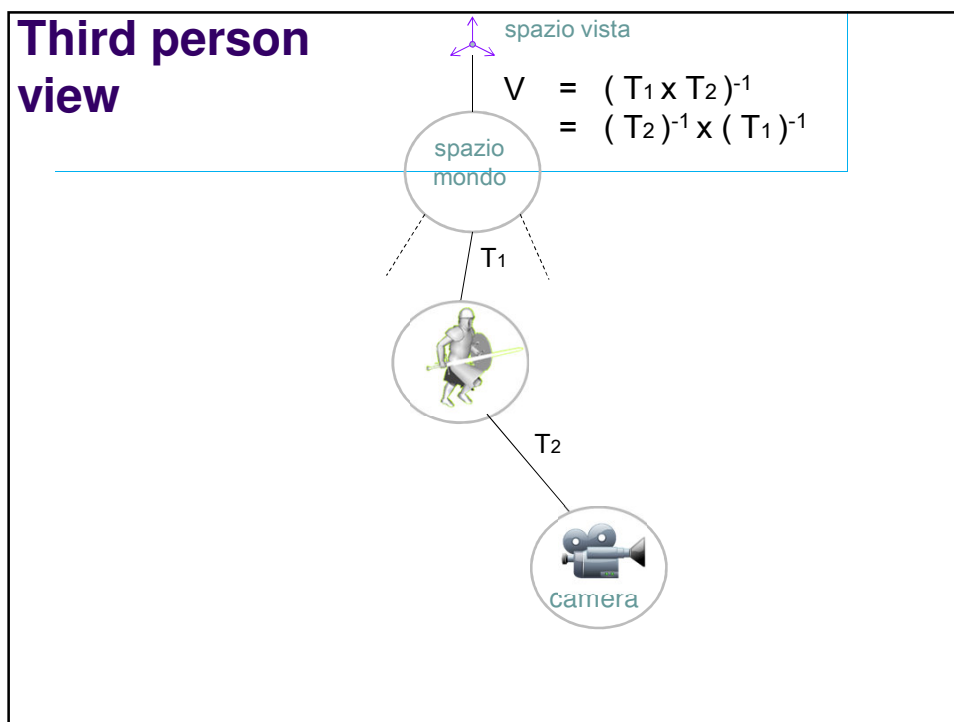
- Due tipi di utilizzo:

- in preprocessing (durante il task dell'animatore)
- in real time (task da **game engine**)

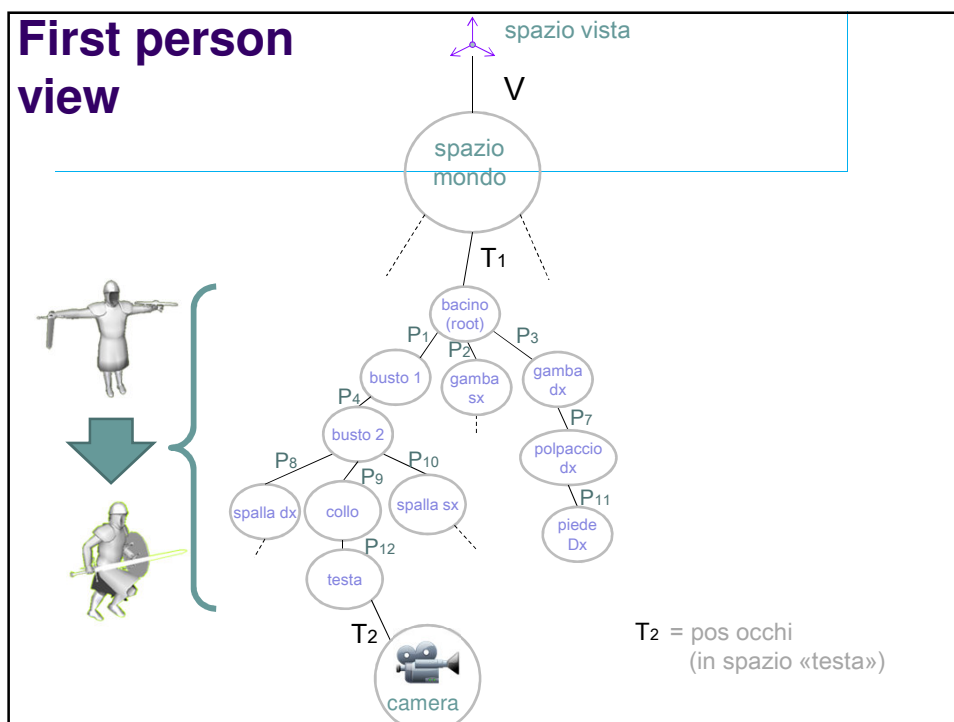
- Esempi di utilizzo in real time:

- Posizionamento esatto piedi su terreno irregolare
- Mano che raggiunge esattamente un oggetto da afferrare
- Mani che si congiungono esattamente in un'arma a due mani
  - (es. nonostante piccole discrepanze nella forma dello scheletro)
  - (es. nelle trasizioni)

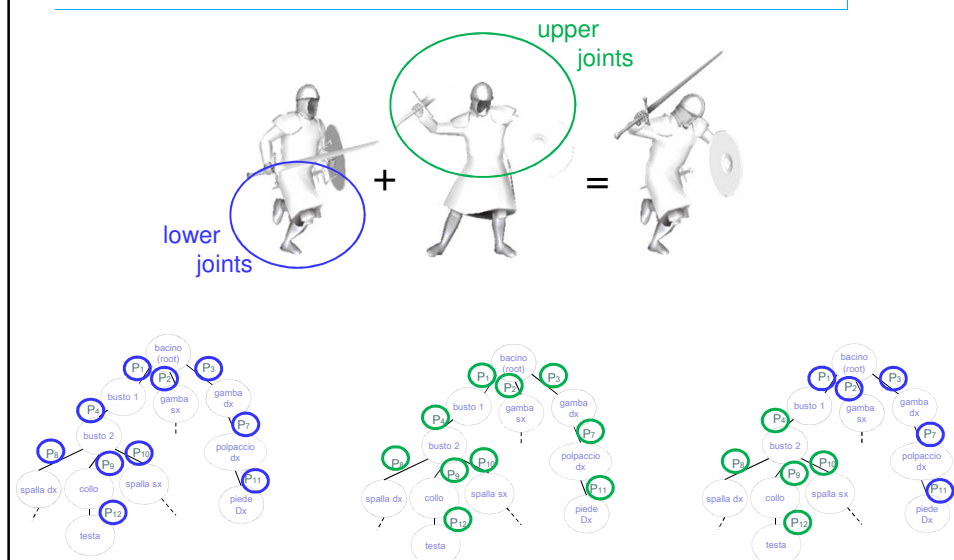
## Third person view



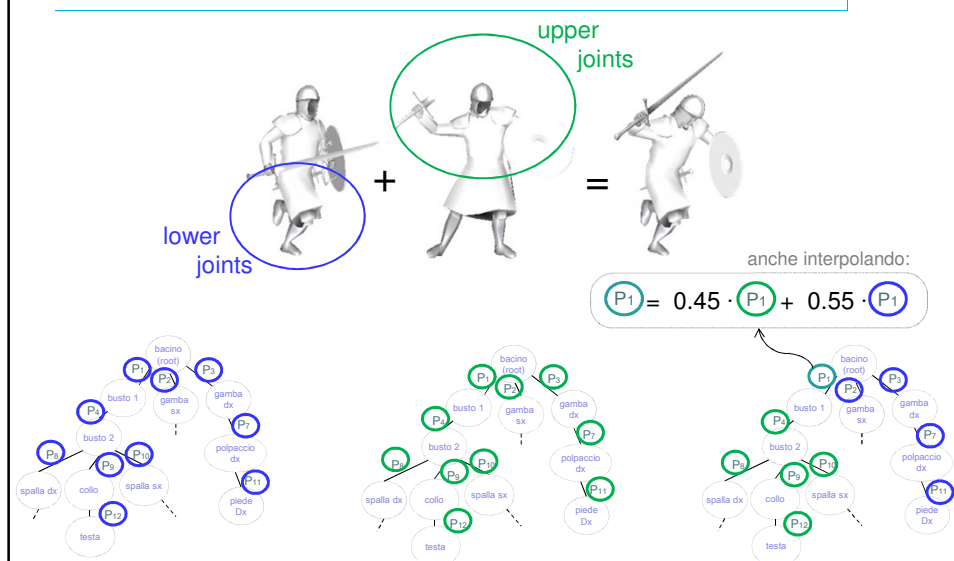
## First person view



## Composizione di pose (→ di animazioni)



## Composizione di pose (→ di animazioni)



## Per-vertex animation VS Rigging

- |  |  |
|--|--|
| <ul style="list-style-type: none"> <li>• Per Vertex animations           <ul style="list-style-type: none"> <li>• possibilità interpolazione</li> <li>• pesante in RAM               <ul style="list-style-type: none"> <li>• replicazione esplicita posizioni + normali</li> </ul> </li> <li>• gratis in GPU</li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li>• Rigged animations           <ul style="list-style-type: none"> <li>• possibilità interpolazione</li> <li>• leggero in RAM               <ul style="list-style-type: none"> <li>• ortogonalità modelli / animazioni</li> </ul> </li> <li>• pesante in GPU               <ul style="list-style-type: none"> <li>• interpolaz transf. x vertice</li> </ul> </li> </ul> </li> </ul> |
|--|--|

## in Unity (cioè Mecanim) (note)



- Assets (modelli, animazioni, scheletri) importabili in formati:
  - fbx, collada
- Compressione animazioni
  - durante import o builds
  - riduz num link per vertice, num keyframes:
- Modulo «Animator Controller» gestisce:
  - transizioni fra animazioni
  - composizione di animazioni (layers)
- Inverse kinematic: da script ( `Avatar.SetIKPosition` )
- Scheletri:
  - custom (importati da file)
  - standard per umanoidi
    - «scheletro per bipedi di unity (di mecanim)» (~21 ossa)
    - interfaccia semplificata