Animazioni

Parte1:
scrpited
animations

Animazioni nei games

- Scripted
  - Parte degli Assets!
  - Controllo da parte degli artisti (dramatic effects!)
  - Non interattiva
  - Realismo… dipende dall'artista
  - Poca customizzabilità
- Computed
  - Physic engine
  - Poco controllo
  - Interattiva
  - Realismo come prodotto collaterale del rispetto leggi fisiche
  - Si autoadatta

Tipi di animazioni scripted

- Di oggetti rigidi
  - animazione di trasformaz di modellazione
  - (anche con giunti: robot, macchine…)
- Di oggetti deformabili con scheletro interno
  - "skinning" / o "rigging"
  - (es: esseri umani)

- Di oggetti deformabili generici
  - "per-vertex animations"
  - (es: volti)

Animazioni scripted

- Keyframes
  +
  inerpolazione

Per-vertex animations

"morph target animation",
"shape interpolation", "blend shapes" ...

[DEMO]

- Mesh con N frames
  - Connettività x 1
  - Geometria x N
  - Attributi:
    - alcuni x 1 (es: UV map)
    - alcuni alcuni x N (es: normals)
- Possibilità di interpolare fra keyframes
  - Interpolaz lineare

Un possibile formato file: (Quake) MD3

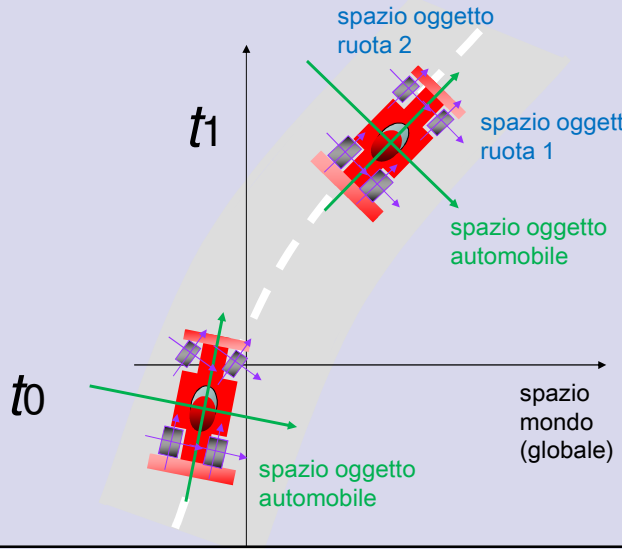
Vertex animations

[DEMO]

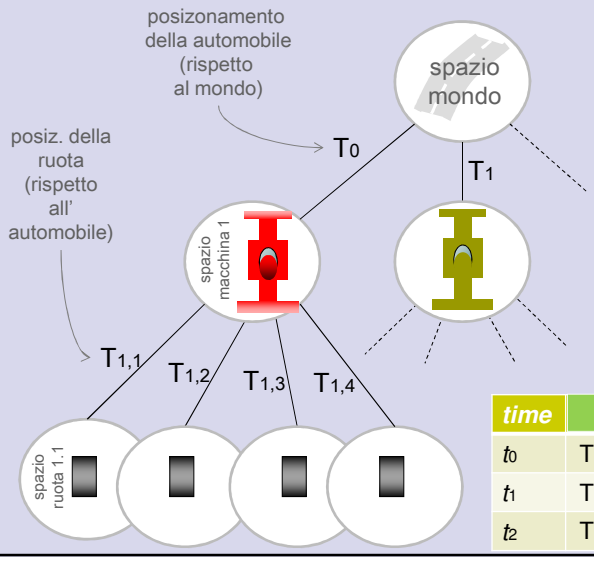
- Es, face morphs

Un possibile formato file: (Quake) MD3

Animazioni (scripted) di trasformazioni di modellazione

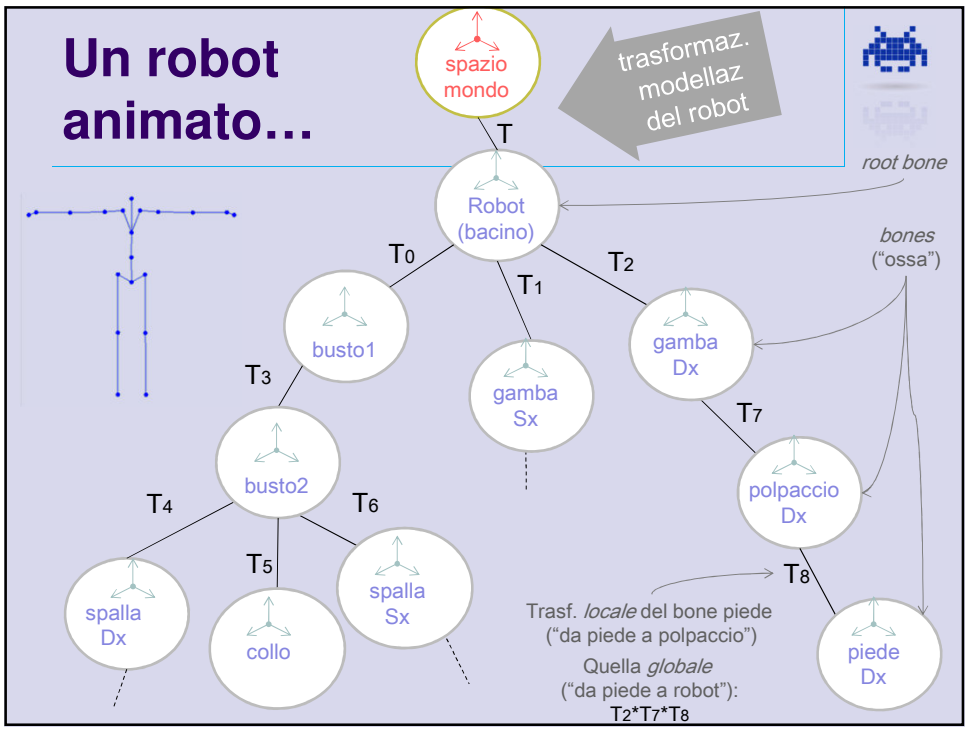


Animaz di matrici di modellazione



Animazione:

time	trasformazioni					
t_0	T_1	$T_{1,1}$	$T_{1,2}$	$T_{1,3}$	$T_{1,4}$	…
t_1	T_1	$T_{1,1}$	$T_{1,2}$	$T_{1,3}$	$T_{1,4}$	…
t_2	T_1	$T_{1,1}$	$T_{1,2}$	$T_{1,3}$	$T_{1,4}$	…



Scheletro

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

Da assemblaggio di pezzi…

- Fin qui: una mesh per ogni osso
  - (es, carlinga, ruota)
- Ok per oggetti meccanici struttre semplici
  - (macchina, torretta con 2 giunti …)
- Ma, per un "robot" di 40 ossa?
  - Mesh 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
  - per ogni vertice, attributo: indice di osso
  - un modello 3D animabile!

"Skinning" della mesh (qui, ad 1 osso solo).

- Ortogonalità modelli / animazioni!
  - ogni modello skinned: va su tutte le animazioni
  - ogni animazione: applicabile a tutti i modelli
    - Basta che si riferiscano ad una stesso scheletro
  - →500 modelli e 500 animazioni = 1000 oggetti in RAM
    - invece di 500x500 combinazioni
- I task dell'artista digitale:
  - Definizione dello skinning (skinners)
  - e "rigging": definizione delo scheletro (riggers)
  - e "animators": definizione delle animazioni

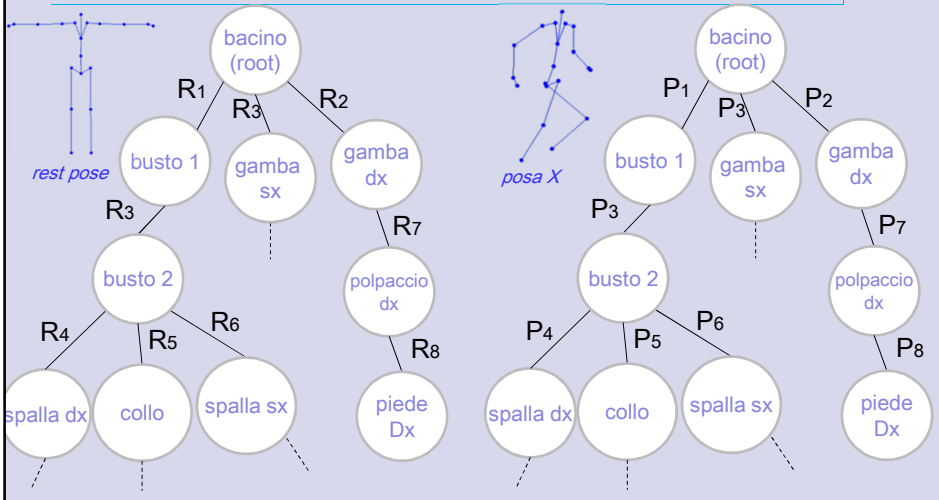
Posa
(frame di un'animazione scheletale)

- 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 *i* costante per tutte le pose)

Da rest pose a una posa data



trasf. globale olpaccio, 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 x bytes_per_trasf
  - per 1 scheletro dato!

Osso *i*	Trasform[*i*]
#0 (bacino, root)	T[0]
#1 (spine)	T[1]
#2 (chest)	T[2]
#3 (shoulder sx)	T[3]
…	…
#10 (polpaccio)	T[10]
…	…

trasformazioni globali

es., se isometrie:
angoli di eulero + trasl (6 floats),
quaternions + trasl (7 floats),
dual quaternion (8 floats)
…
di solito: matrici 4x3 (12 floats)

es: calcolato in preprocessing come:
$T_2\ T_7\ (P_7)^{-1}\ (P_2)^{-1}$

trasformazioni locali
(della rest pose
o della posa data)

Animazione

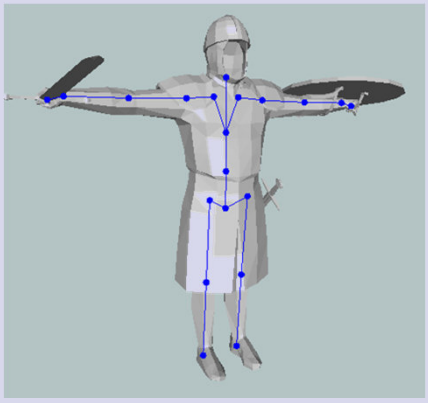
- Array 1D di pose
  - Costo RAM:
    (num frames) x (num ossa) x (bytes x trasf)
  - Spesso tenuto in RAM scheda viseo
  - Compressione:
    pose keyframes (vedi dopo)

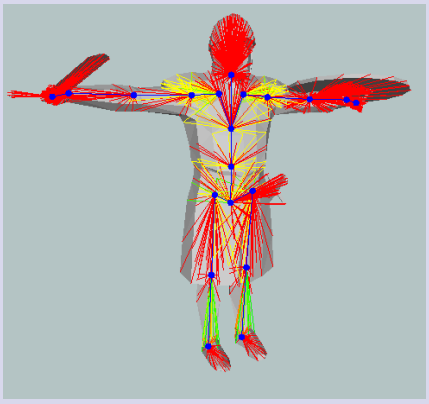
Da robot di pezzi, a oggetti deformabili

- 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)

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

Mesh skinned





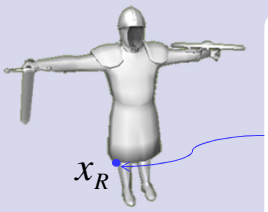
Applicare le pose: *Linear Blend Skinning*

- Per ogni vertice della mesh:

interpolaz. delle trasformaz. legate dal rigging al vertice

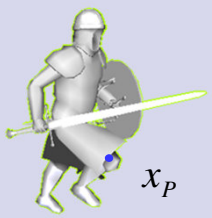
$$x_P = \left(\sum_{i=1}^{N_{\max}} w_i T[b_i] \right) (x_R)$$

computaz a bordo della scheda video (in GPU)



posiz del vertice definito nella rest pose.

Skinning del vertice (con Nmax = 4): (b_1, w_1) (b_2, w_2) (b_3, w_3) (b_4, w_4)



Dual Quaternion Skinning

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

Quante link ad osso per vertice

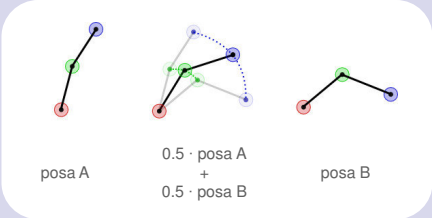
- Dipende dal game engine!
- N_{max} tipicamente:
  - 1 (sottopezzi rigidi)
  - 2 (cheap, e.g. su dispositivi mobili)
  - 4 (top quality)
- Decrementare N_{max}
  - in preprocessing
    (task per un game tool!)

(perchè limite sup 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 lungezza fissa:
    - N_{max} coppie (indice , peso)
    - anche dove, localmente, ne basterebbero meno (pesi 0)

Interpolazione pose

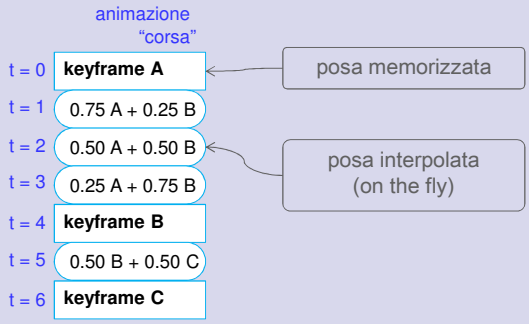
- Due pose si possono interpolare!



  - basta interpolare le trasformazioni che le compongono
  - (vedi lez rappres. rotazioni)

Interpolazione pose (a runtime): keyframes

- Per comprimere animazioni

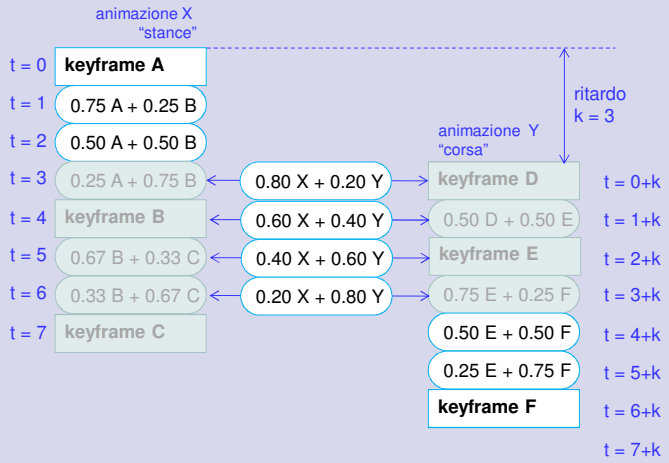


Compressione animazioni

- Obiettivo: rimuovere keyframes
  - quelli “inutili”
  - in preprocessing (game tools!)
- Versione 0.0 di un algoritmo:
  - per ogni keyframe
    - rimuovi keyframe Px
    - computa versione interpolata Pi dai keyframe rimanenti
      - (il precedente e il successivo)
    - se distanza(Pi , Px) > ErrMax allora reinserisci keyframe Px

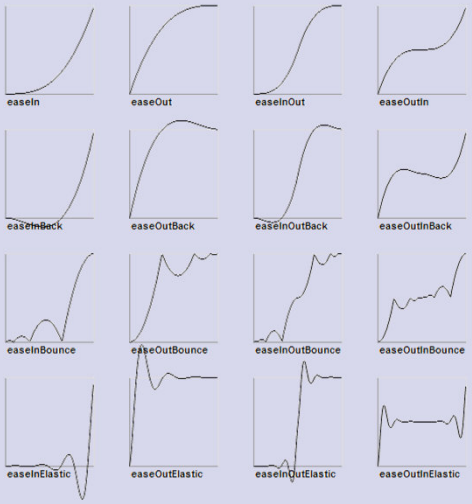
Interpolazione pose (a runtime): transizioni fra animazioni

- Es: da stance a corsa



digression:
Transition functions

- Concetto generale nelle animaz

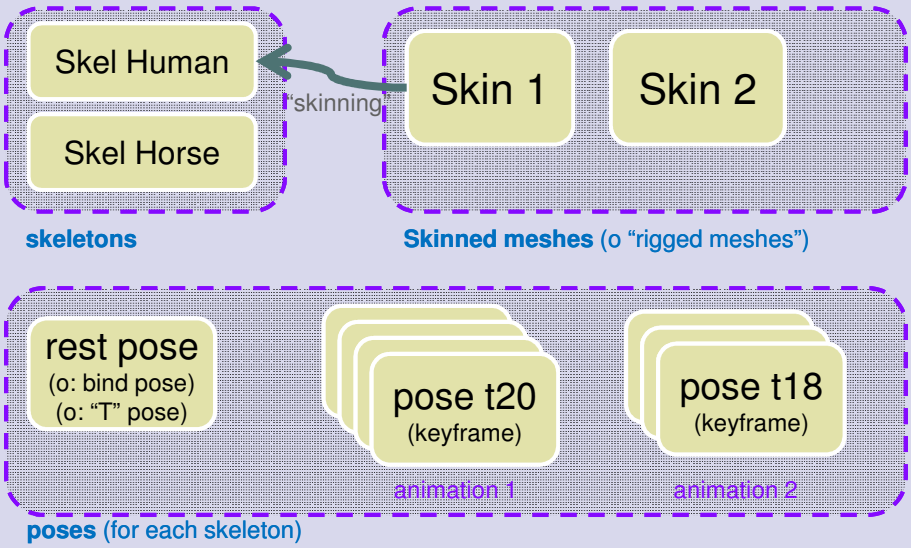


Skeletal animations: strutture dati (riassunto)

- Scheletri
  - Tree di bones (ossa)
  - Ogni bone => sistema di riferimento (in rest pose)
    - (sistama dell'osso root = sistema oggetto)
- Modelli 3D skinned
  - Mesh con associazione vertici => bones
  - Per vertice: [indice osso , peso] x N_{max}
- Animazioni scheletali
  - Sequenza di pose
  - Posa = trasformazione globale ∀ bone

possibile formati file (per tutti e tre i dati):
- **.SMD** (Valve), **.FBX** (Autodesk), **.BVH** (Biovision)

Assets di un modello animato: (riassunto)



Rigging e skinning di modelli 3D

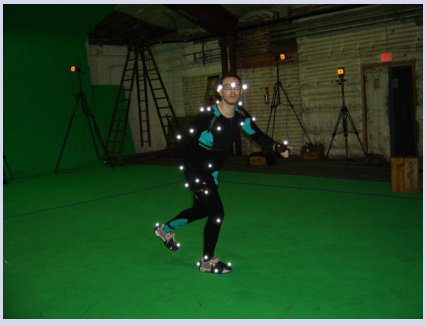
- rigging e skinning di una mesh:
  - rigging: definizione di uno scheletro per quella mesh
  - skinning: definizione associazione vertice ossa
  - Due dei task dei modellatori digitali!
    - "skinners" e "riggers"
    - aiutati (o sostituito) da strumenti automatici
- Animatori digitali
  - Definizione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

Animazione skeletali

- DEMO

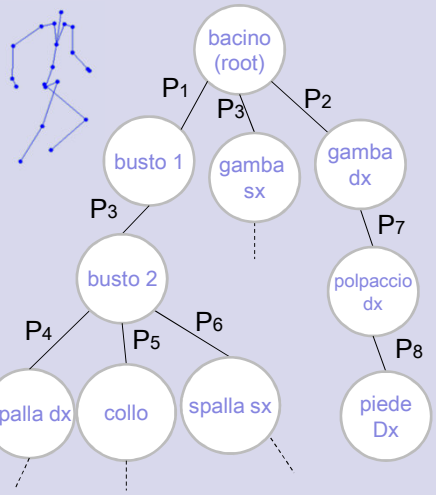
Animazioni scheletali (scripted): come si ottengono

- Editing manuale
  - Animatori digitali
- Motion capture:



Inverse Kinematic: un tool utile

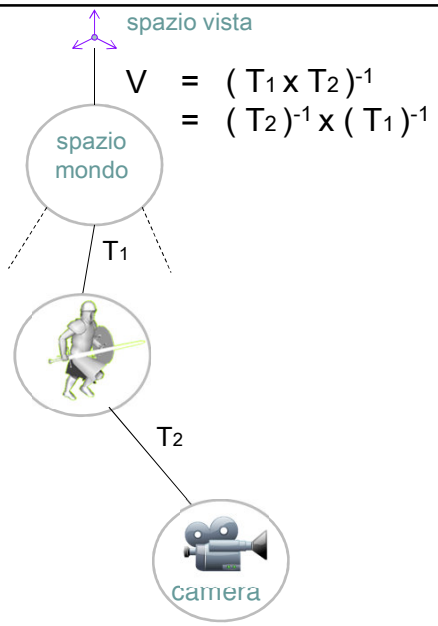
- Cinematica diretta:
  - "date le trasf locali P_1, P_2… P_N, (es, come angoli) dove finisce il piede?"
  - (univoco, banale)
- Cinematica inversa:
  - "se voglio mettere il piede nella pos p, quali P_1, P_2… P_N, ?"
    - (+ vincoli, es. DoF giunti)
  - (non univoco, non banale)



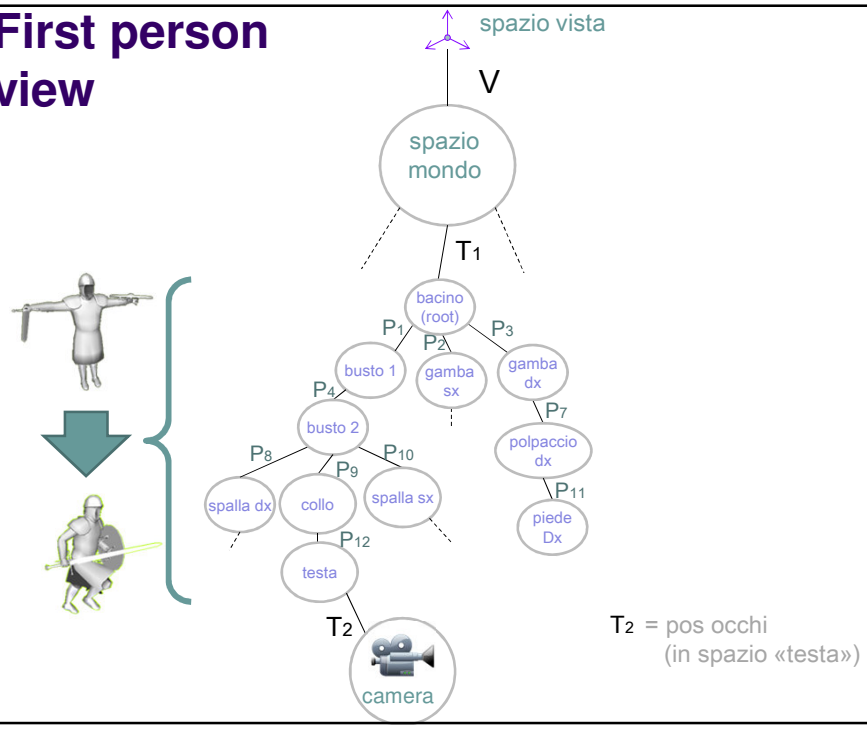
Inverse Kinematic: un tool utile

- Quando:
  - in preprocessing (task da game tools)
  - in real time (task da game engine)
- Esempi di utilizzo:
  - 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)

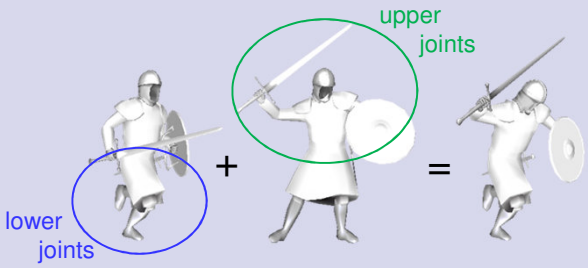
Third person view

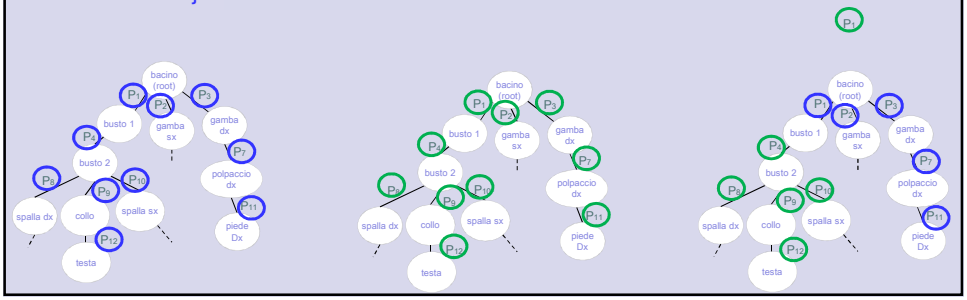


First person view

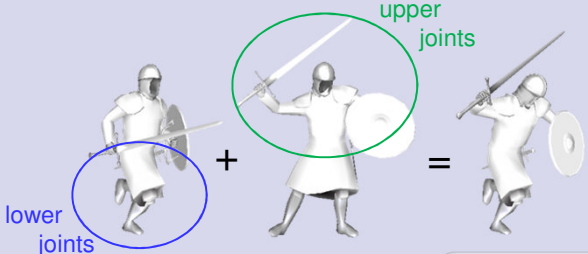


Composizione di pose
(→ di animazioni)



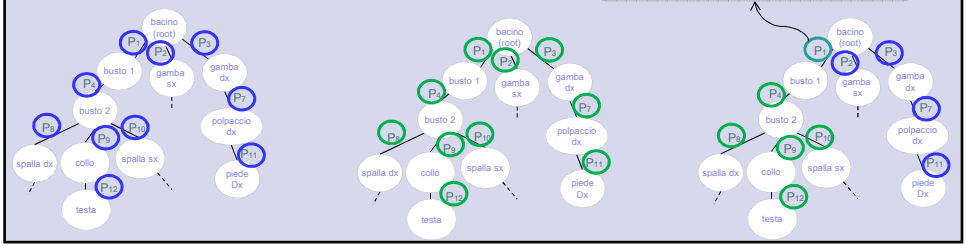


Composizione di pose
(→ di animazioni)



anche interpolando:

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



Per-vertex animation VS Rigging

- Per Vertex animations
  - possibilità interpolazione
  - pesante in RAM
    - replicazione esplicita posizioni + normali
  - gratis in GPU
- Rigged animations
  - possibilità interpolazione
  - leggero in RAM
    - ortogonalità modelli / animazioni
  - pesante in GPU
    - interpolaz trasf. x vertice

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.SetIKPoistion`)
- Scheletri:
  - custom (importati da file)
  - standard per umanoidi
    - «scheletro per bipedi di unity (di mecanim)» (~21 ossa)
    - interfaccia semplificata