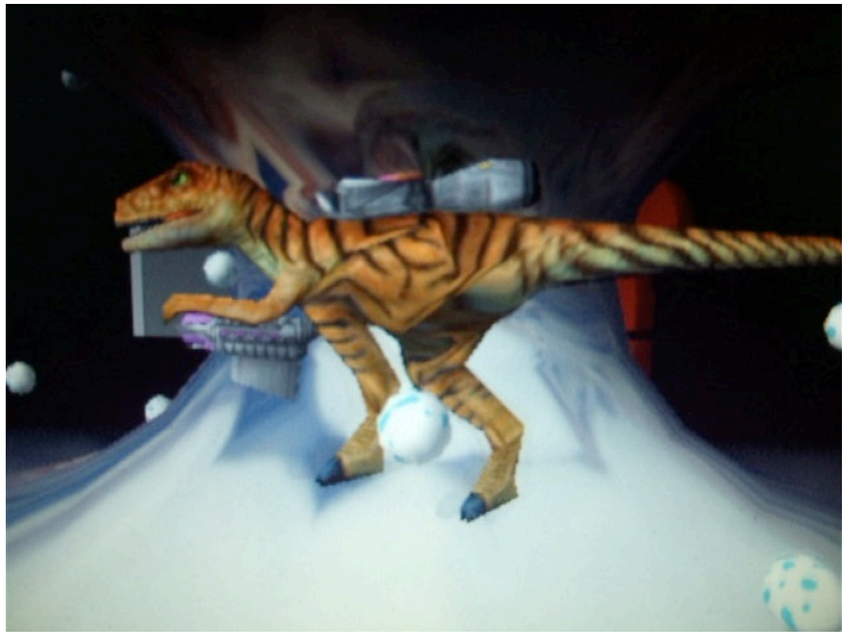
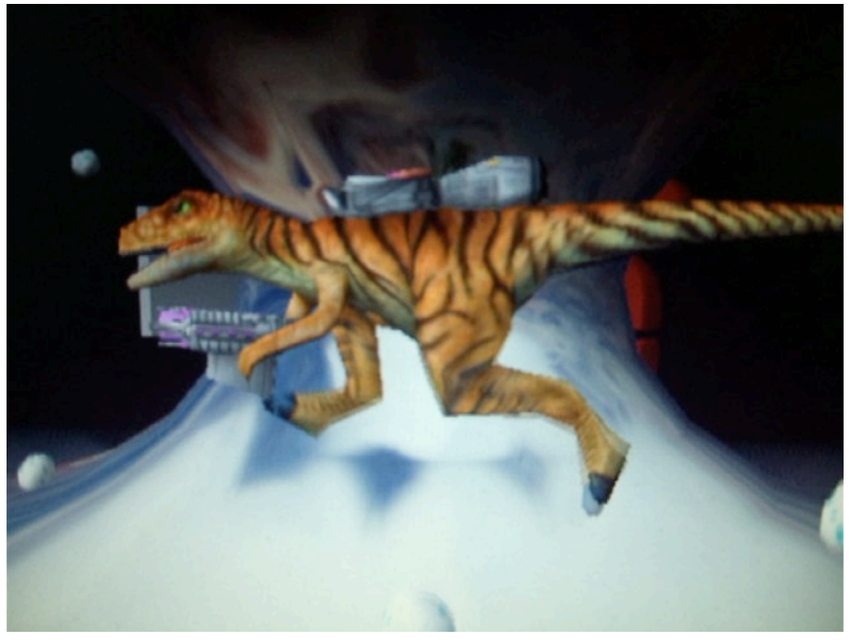
Skinning

Animation baserat på en "benstruktur" som "huden", 3D-modellen, deformeras efter.





Koordinatsystem!

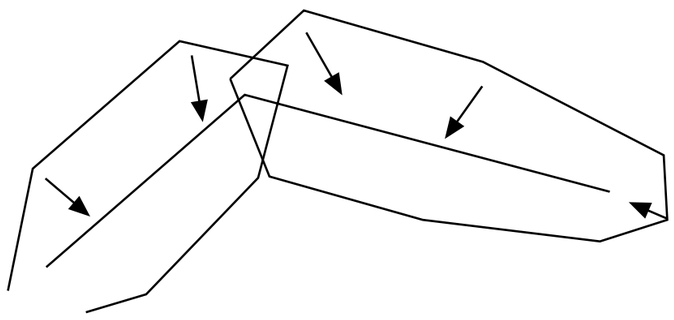
Samma grundproblem som bump mapping: En fråga om koordinater och transformationer

…fast lite fler.

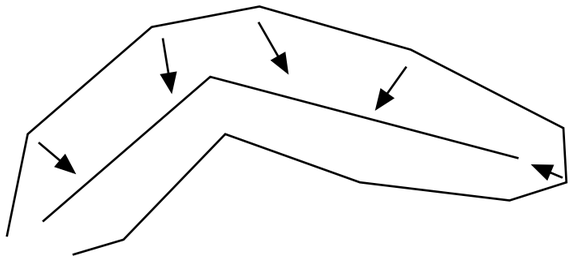
Tre sätt att animera en modell

- Parenting
- Stitching
- Skinning

Parenting: Separata stela meshar följer varje ben



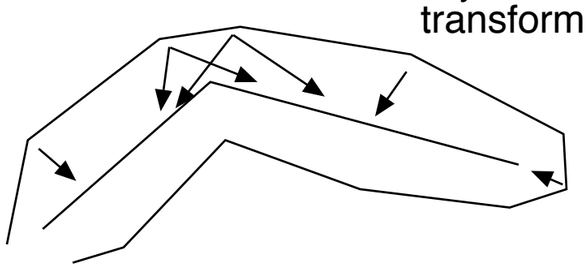
Stitching: Varje vertex följer ett ben



Skinning: En vertex kan viktas efter flera ben

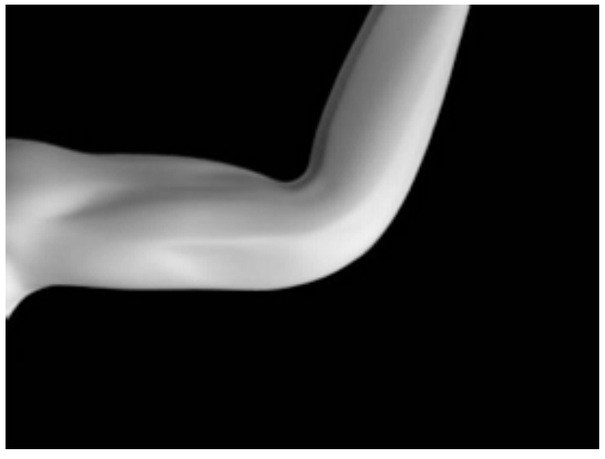
Två närliggande ben, följ båda med lämplig viktning

En närliggande, följ benets transform



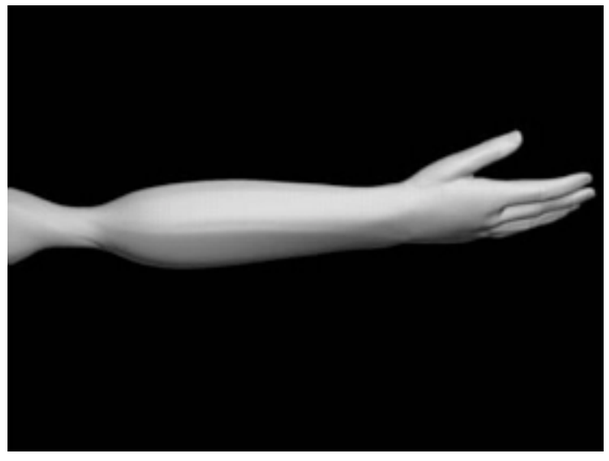
Avancerad skinning

Grundläggande algoritmen har vissa svagheter



<- Collapsing elbow

Twist ->



Kan åtgärdas med

- Flera ben i en och samma led
- Shape blending - flera modeller som man morphar mellan
- Volymbevarande tilläggsvillkor

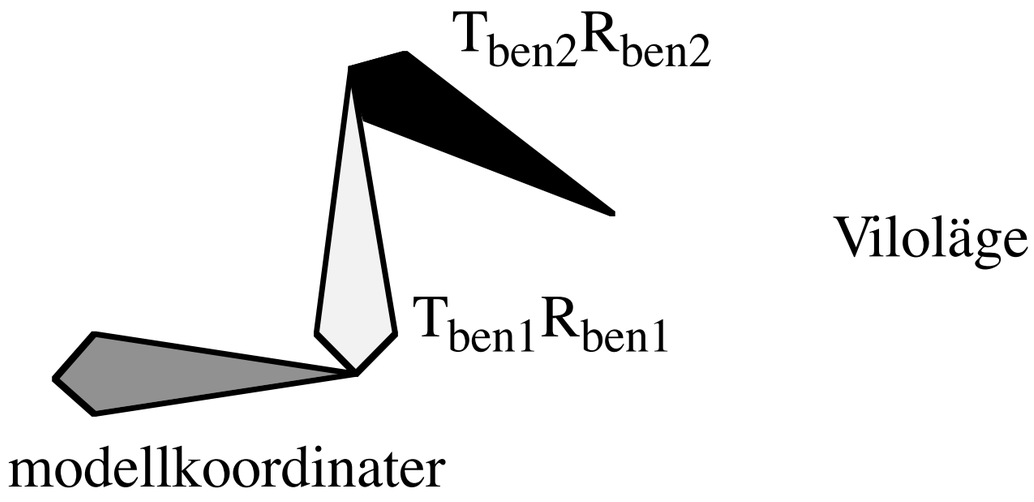
Koordinatsystemen

Mesh given i modellkoordinater

Mesh och skelett givna i viloläge

Hur påförs animation?

Hur beräknar man en vertex modifierade position?



Varje bens position definieras av en transformation, en translation och en rotation:

$$M_{ben2} = T_{ben2}R_{ben2}$$

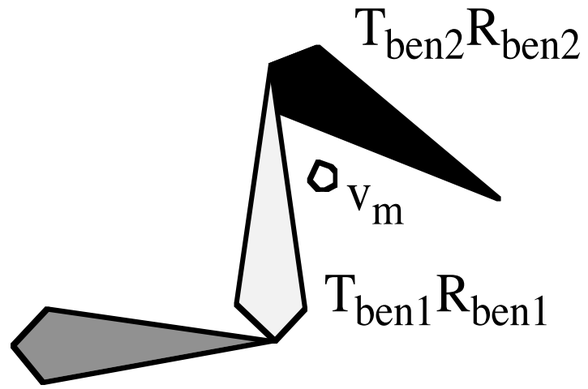
Transformation av vertex från modellkoordinater till benkoordinater:

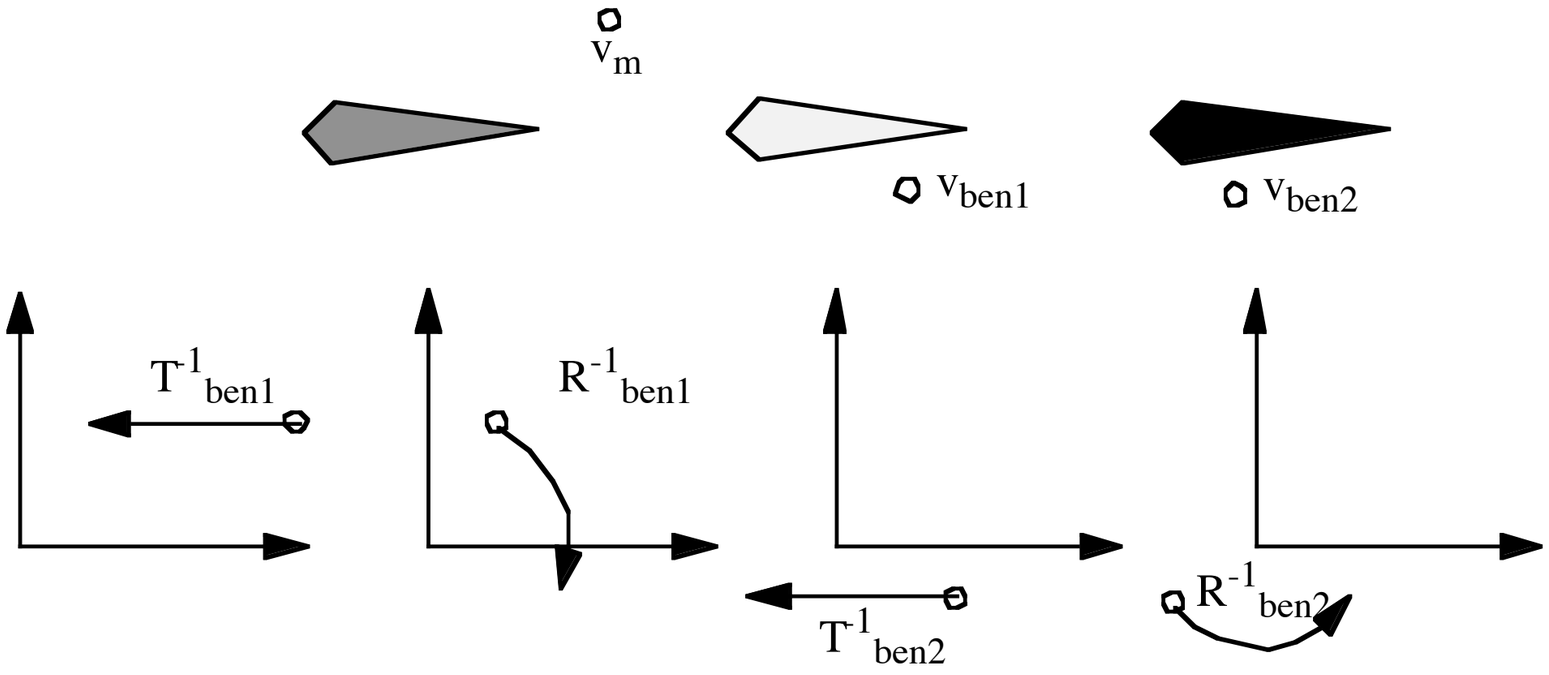
$$v_{ben2} = M^{-1}{}_{ben2}M^{-1}{}_{ben1}v_m$$

Transformation av vertex från benkoordinater till modellkoordinater:

$$v_m = M_{ben1}M_{ben2}v_{ben2}$$

Vi följer en vertex genom koordinatsystemen:





Animation:

Transformera till benkoordinat med viloläget.

Transformera tillbaka till modellkoordinater med modifierade benpositioner

$$v'_m = M'_{ben1} M'_{ben2} M^{-1}_{ben2} M^{-1}_{ben1} v_m$$

Animation oftast rotationer:

Vilolägestransformation är vilolägets rotation och translation

$$M_{ben} = T_{vila}R_{vila}$$

På denna tillkommer sedan animationens rotation

$$M'_{ben} = M_{ben}R_{anim} = T_{vila}R_{vila}R_{anim}$$

Traversera skelettet för resulterande transformation:

Modellkoordinater till benkoordinater:

$$M_{mb} = \prod M^{-1}_{ben,i}$$

Benkoordinater till modellkoordinater:

$$M_{bm} = \prod M_{ben,i} \cdot R_{anim,i}$$

och hela transformationen kan skrivas

$$v' = M_{bm} M_{mb} \cdot v$$

Viktning av vertexar

$$v' = \sum_{i=1}^{n} w_i M_i v$$

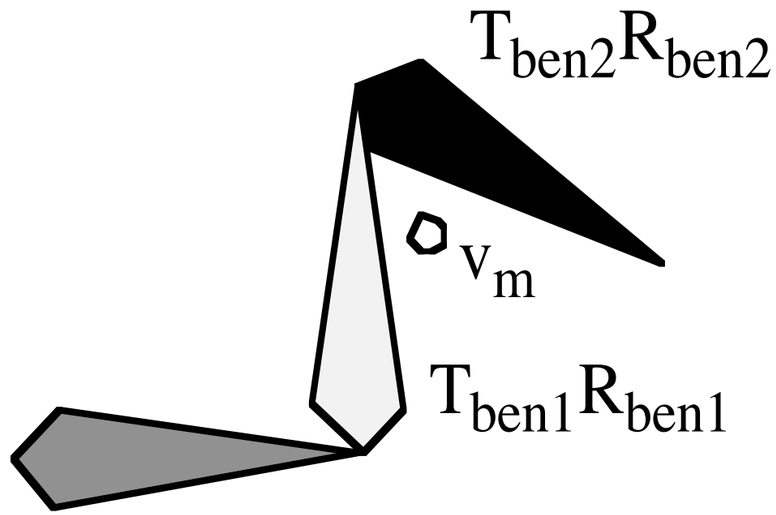
M_i beräknas, enligt ovan, som

$$M_{mbi} = \prod_{j=1}^{i} M^{-1}_{ben,j}$$

$$M_{bmi} = \prod_{j=1}^{i} M^{-1}_{ben,j} \cdot R_{anim,j}$$

$$M_i = M_{bmi} M_{mbi}$$

Exemplet med två ben ovan. Antag att v_m påverkas av ben1 och ben2.



$$M_1 = M'_{ben1}M^{-1}_{ben1}$$
$$M_2 = M'_{ben1}M'_{ben2}M^{-1}_{ben2}M^{-1}_{ben1}$$

$$v'_m = \sum_{j=1}^{2} w_i M_i v = w_1 M_1 v_m + w_2 M_2 v_m$$

Beräkna allt i rätt domän

Matriserna kan beräknas per ben.

$$M_{mb} = \prod M^{-1}_{ben,i}$$
$$M_{bm} = \prod M_{ben,i} R_{anim,i}$$
$$M_{tot} = M_{bm} M_{mb}$$

Transformationen av vertex görs per vertex.

$$v' = M_{tot} \cdot v$$

I shader

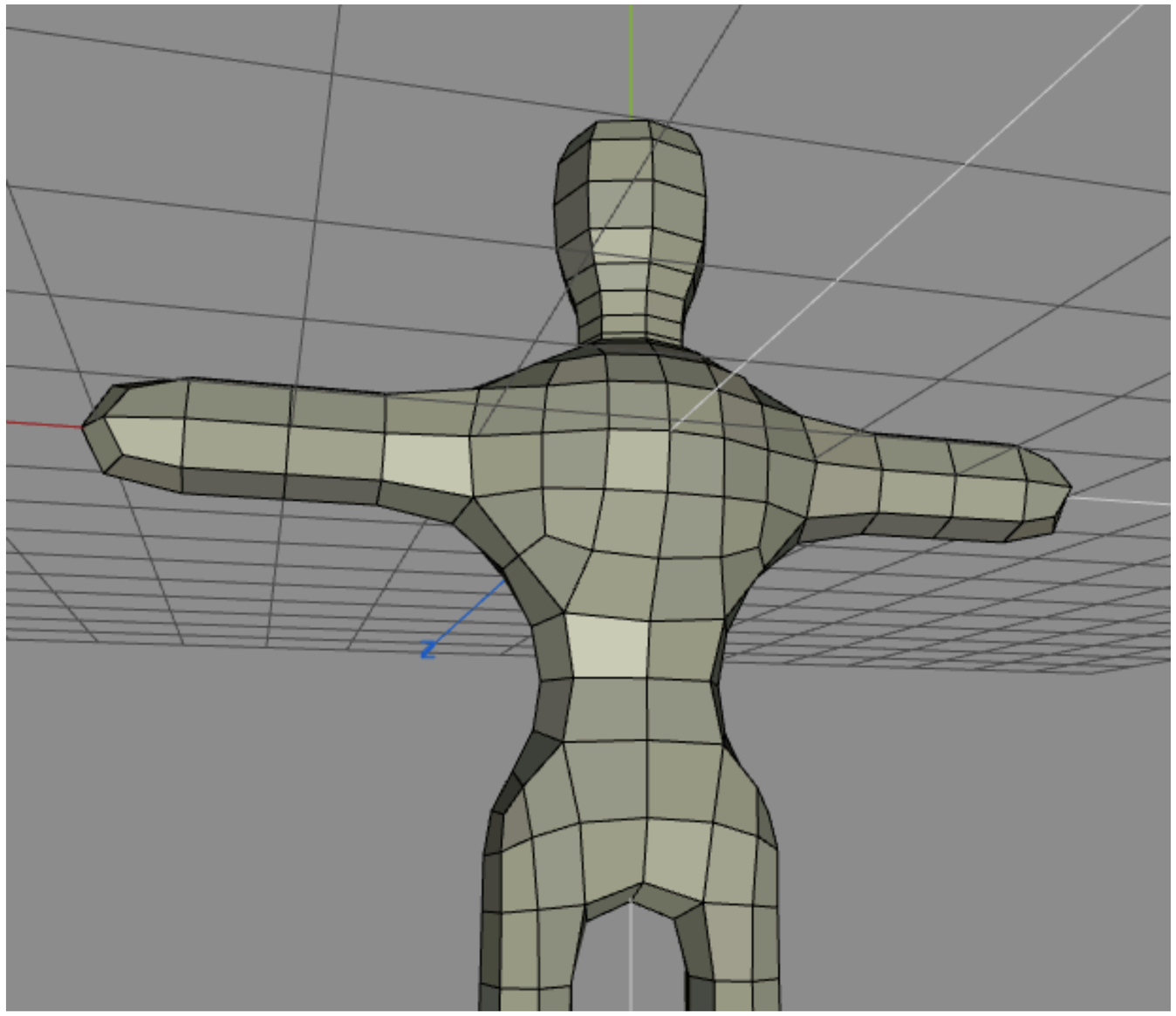
Per ben-operationer: Görs på CPU

Per vertex-operationer: Görs i vertex shader

Matriserna kan skickas som uniform

Vikterna w_i är per vertex, skickas som attribute arrays

Undvik att deformera hela objektet på CPU. Då måste hela modellen laddas om varje frame. Vitalt för stora modeller.



Modeller skapas typiskt i ett lämpligt "viloläge"

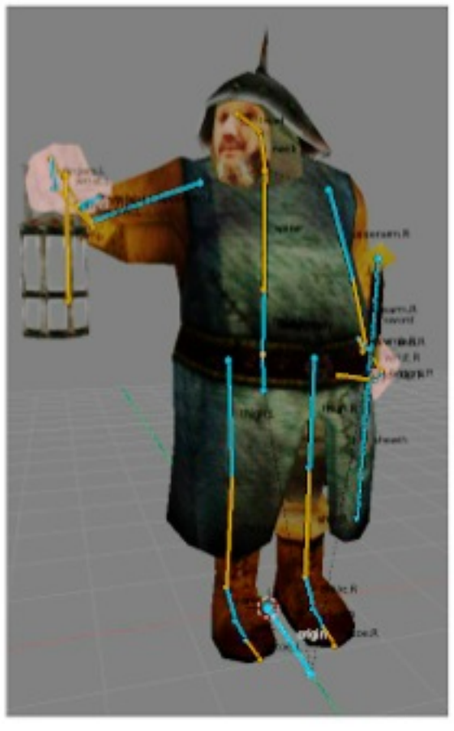
"Reference pose", även känt som "T-pose" eller "A-pose"

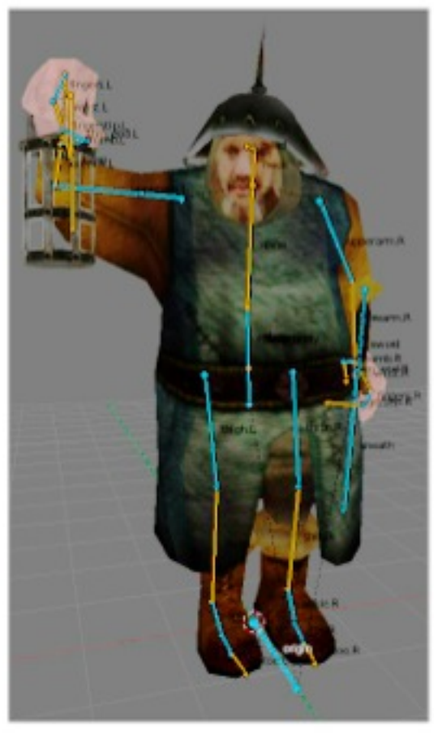
Liten anekdot: "Civilian pose" i TF2 kommer från en tidig karaktär, "the civilian"

Refererar idag till att hitta buggar som får karaktärer att låsa sig i referensläget.

Var kommer vikterna ifrån?
Animationsparametrar - hur ändras rotationerna?

Inbyggt i 3D-program, t.ex Blender



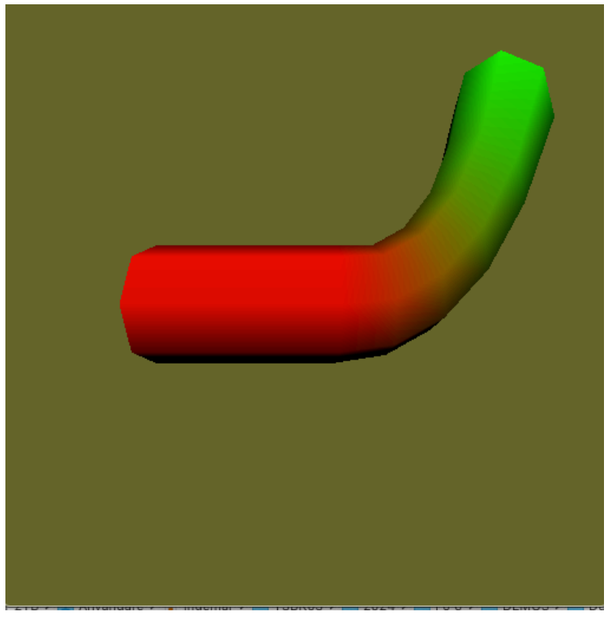


Exempel från atspaces skinning tutorial

Labb 2

Enkla fall: Slang

Inga grenar. Enkla vikter längs slangen.



Projekt på skinning?

Massor av möjligheter!

Skinning på mer komplex modell än labben

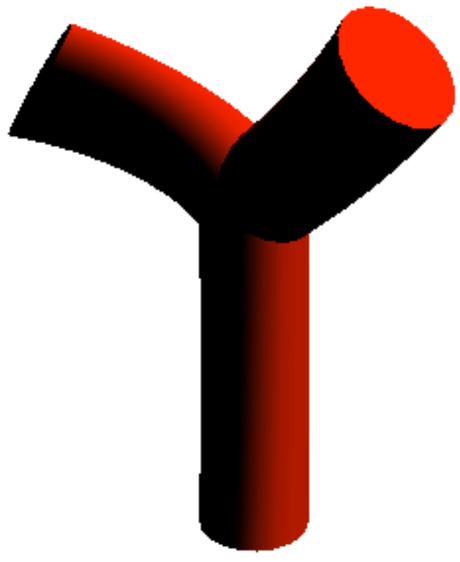
Skinning + animationsdata

Förbättrad skinning, undersök någon metod eller jämför flera

GLUGGW möjligt verktyg?

GLUGG med stöd för skinningvikter.

Prototyp! Osäker mark. Men mycket hjälp från mig på grund av detta.



Vad är målet? GLUGGW är i grunden till för att experimentera med procedurell generering av geometri och skinningvikter. Men om du vill jobba från färdig riggning då...?

Preliminär ASSIMP-laddare

OBJ-filer är bra... men kan inte innehålla skinningvikter!

Assimp (asset importer) kan ladda t.ex. Collada.

MEN Assimp är ett elände att använda!

Jag har preliminära Assimp-laddare, inklusive laddare med vikter.