

Skinning

Ännu fler koordinatsystem att hålla reda på

SUPPLEMENT 2 till "So How Can We Make Them Scream?"

Version 1.1, 08-09-25. Vissa rättelser i formlerna.

Så här under kursens gång har jag hittat en lucka till i materialet, nämligen att skinning inte är komplett och väl beskrivet. När jag skrev boken kom skinning-delen in mycket sent och blev ett hastverk baserat på Johans föreläsningssanteckningar, och allt jag ville haft med hans inte med. Därför behövs detta supplement.

1. Olika metoder för kroppsanimation

Det klassiska sättet att animera en människokropp eller liknande är hierarkisk modellering, där varje kroppsdel är en separat stelkropp. Kroppen organiseras hierarkiskt, typiskt med någon punkt på ryggraden som rot, och animeringarna beskrivs med transformationer från varje nod till de underliggande. Detta kallas för *parenting* [3].

För robotar och insekter är detta utmärkt, men för däggdjur och annat med mjuk hud är det förfärligt dåligt. Därför tog man snart steget till skinning.

Skinning är ett lite knepigt ämne att läsa om, eftersom det är så spretigt i terminologin. Enligt Lewis [1] är grundalgoritmen för skinning aldrig publicerad. Därför finns det ingen given huvudreferens och vi får olika namn beroende på källorna. Lewis kallar metoden "Skeleton-Subspace Deformation". Ett annat namn är *enveloping*. Jag väljer att använda det vanligare *skinning*, som bl.a. används av Maya [2]. Om du har andra åsikter än jag om vad saker borde heta så är jag öppen för förslag, men jag försöker välja termer som både känns rätt och jag upplever är populära.

Att skinning har stökig bakgrund kan också förklara varför litteraturen ofta hanterar begreppet mycket styvmoderligt. Flera av de starkaste böckerna i ämnet ger inkompleta eller luddiga beskrivningar. Ändå är metoden allmänt känd. Det finns säkert böcker med bra beskrivningar, men de är inte så lätta att hitta som man skulle tro.

Den enklaste formen av skinning kallas *stitching*. [4] I stitching binds varje vertex till ett enda ben.

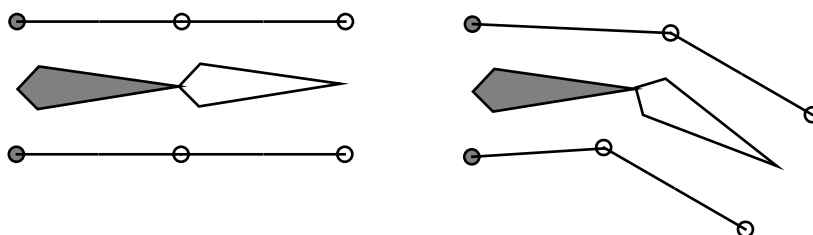


FIGURE 1. Stitching - varje vertex följer ett ben

Stitching är förvånansvärt bra i många fall, när rotationsleder och vertexar passar bra för metoden, med relativt få vertexar och relativt små rotationer. Det räcker dock bara till en viss gräns, och med dagens allt större detaljnivå är det ofta en otillräcklig metod.

Lösningen är att låta mer än ett ben påverka vertexar som är i närheten av leder. Varje vertex har därmed ett antal vikter, vars summa är 1, varav varje vikt refererar till ett ben. Antalet ben som påverkar en viss vertex är oftast upp till 4.

Även detta är inte en perfekt metod. Den har flera svagheter, som beskrivs av Lewis [1]. Därför behövs mer avancerade skinningmetoder, som till exempel att använda flera modeller som man tonar mellan, eller lägga till villkor som ger volymbevarande. Den enklaste lösningen är nog att använda flera ben i varje led. Det har naturligtvis ingen fysiologisk motsvarighet, men det har sällan animationsskelett över huvud taget. Är ni intresserade av skinning så kan det vara intressant att utforska någon mer avancerad metod.

2. Koordinatsystem och transformationer

Det mest grundläggande problemet i skinning är hur man finner en vertex modifierade läge när benen flyttas. Detta viktiga och inte helt triviala problem är mycket ofta ignorerat i texter om skinning (tyvärr inklusive skinningkapitlet i “So How Can We Make Them Scream”). Detta är därför det problem som jag känner är allra viktigast att reda ut.

Alla vertex ges i en enda polyedermodell (mesh) i ett viloläge för vilket vi också känner alla benens lägen. Detta är *modellkoordinater*, från vilket rotbenets transformation till världen kan föra oss till världskoordinater.

Varje ben definierar ett eget koordinatsystem. Dessa koordinatsystem ligger längre “till höger” än modellkoordinater i transformationskedjan (som den beskrivs i PFNP). Därför anser vi att transformationerna bör ges från dessa koordinater strävande mot modellkoordinater (och därifrån vidare till världskoordinater, vykoordinater etc).

Låt oss för enkelhetens skull ta ett exempel, där vi har en rotnod följd av två ben:

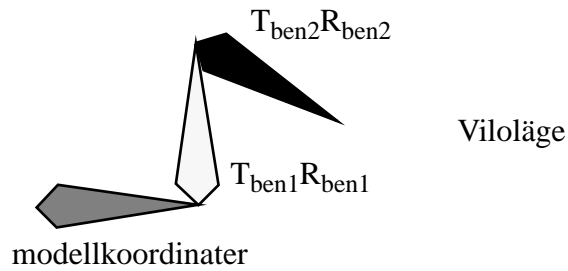


FIGURE 2. Exempelmodell, rotnod (grå) plus två ben (ljusgrå, svart) i sitt viloläge.

Benens vilolägen definieras av var sin transformation, $M_{ben2} = T_{ben2}R_{ben2}$.

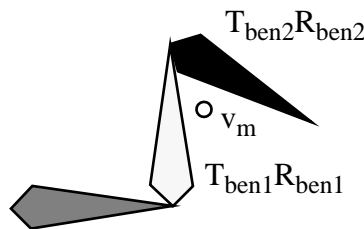


FIGURE 3. En vertex v_m (vit cirkel) angiven i modellkoordinater.

Om vi har en vertex v_m definierad i modellkoordinater, och vill ha den i koordinatsystemet för ett ben så måste vi multiplicera transformationerna för samtliga ben mellan roten och benet. Eftersom dessa transformationer är givna åt andra hållet så är det *inverserna* av dessa vi behöver.

$$v_{ben2} = M_{ben2}^{-1}M_{ben1}^{-1}v_m$$

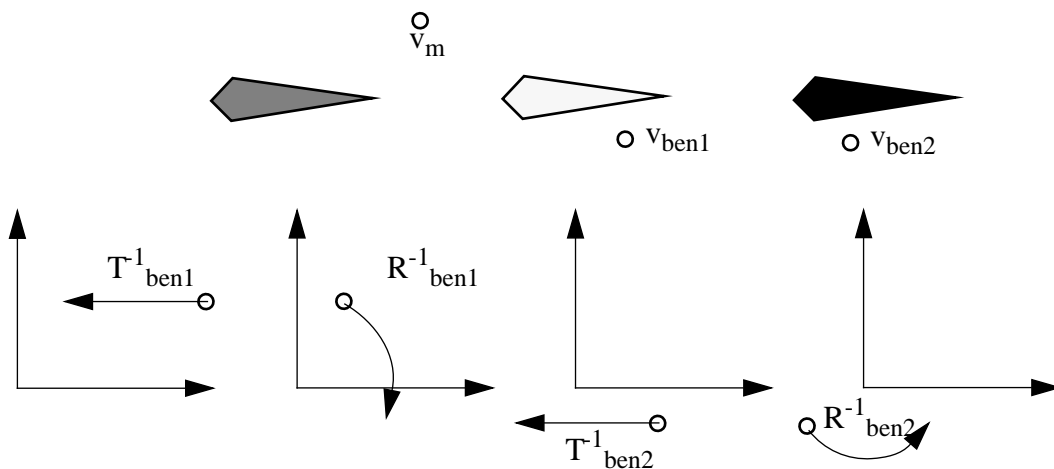


FIGURE 4. Denna vertex transformeras mellan de olika koordinatsystemen.

Om en vertex \mathbf{v} å andra sidan är definierad i benlokala koordinater så kan vi transformera det till modellkoordinater med:

$$\mathbf{v}_{\mathbf{m}} = M_{\text{ben1}} M_{\text{ben2}} \mathbf{v}_{\text{ben2}}$$

När vi animerar skelettet så modifieras transformationerna, så det definieras av de justerade transformationerna M'_{ben1} och M'_{ben2} . För att gå från en vertex $\mathbf{v}_{\mathbf{m}}$ i modellkoordinater till dess justerade position $\mathbf{v}'_{\mathbf{m}}$, också i modellkoordinater, blir därmed hela transformationen

$$\mathbf{v}'_{\mathbf{m}} = M'_{\text{ben1}} M'_{\text{ben2}} M^{-1}_{\text{ben2}} M^{-1}_{\text{ben1}} \mathbf{v}_{\mathbf{m}}$$

Ersätt ben1/ben2 med längre sekvenser efter behov.

Så hela processen blir:

- Transformera till benets lokala koordinatsystem med inverserna av bentransformationerna för viloläget.
- Transformera tillbaka till modellkoordinater med transformationerna för ändrat läge.

Så precis som i bump mapping (och många andra situationer) så är det viktigaste att hålla bra koll på vilket koordinatsystem man är i, så blir problemet plötsligt lätt.

Ofta framgår denna process bara i koden till skinning-demos, medan artiklar kring det inte diskuterar hela transformationskedjan. Till och med i labben är denna process mer antydd än explicit.

Beräkningsordningen har betydelse. Multiplikationen av matriserna görs per *ben*, och benets resulterande matris multipliceras sedan per *vertex*.

En möjlig optimering är att utföra inverstransformerna i förväg, så att alla vertexar finns i benlokala koordinater. [4] För skinning på CPU är detta inget problem. Skinning på GPU är dock mer minneskänslig.

3. Viktning av vertexar

Det bör nämnas att matrissekvensen $M'_{\text{ben1}} M'_{\text{ben2}} M^{-1}_{\text{ben2}} M^{-1}_{\text{ben1}}$ är detsamma som matrisen R_i formeln

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

som kommer från SHCWMTS sid 133.

Vi påstod att man ignorerar translationsdelen i transformationen. Detta är tyvärr *felaktigt*. I en typisk animation ändras rotationerna, vilket roterar translationerna, varvid transformationerna naturligtvis måste appliceras i rätt ordning. Därför skall det stå:

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

För exemplet ovan, antag att en vertex påverkas av ben1 och ben2. Då blir

$$M_1 = M'_{\text{ben1}} M_{\text{ben1}}^{-1}$$

$$M_2 = M'_{\text{ben1}} M'_{\text{ben2}} M_{\text{ben2}}^{-1} M_{\text{ben1}}^{-1}$$

och viktningen utförs som

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

4. Animationsparametrar

För varje ben finns alltså en transformationsmatris M_{ben} som definierar transformationen från benets koordinatsystem till ovanliggande nivå. Vid animation skall denna modifieras. Normalfallet för skinning är att vi påför *rotationer*. Vi kan skriva benets vilolägen som en rotation och en translation:

$$M_{\text{ben}} = T_{\text{vila}} \cdot R_{\text{vila}}$$

Rotation av benet med R_{anim} ger då

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

För beräkningen av en vertexposition gör vi så här: Matrisen för att transformera modellkoordinater till benkoordinater är

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

och transformationen tillbaka till modellkoordinater är

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

och vi får

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

för en sekvens ben indexerade med i .

5. Implementation i shader

För att implementera skinning i en shader så måste vi meddela shadern en relativt stor mängd variabla data. Såväl transformationer som vertexvikter måste levereras. Om möjligt bör dessa skickas som attribute- eller uniform-variabler, då dessa ger högre prestanda än data i texturer. Observera att man kan skicka hela arrayer av data till attribute och uniform, men minnesmängden är begränsad så vi kan inte skicka hur mycket som helst.

Vi är tillbaka på den här viktningsformeln:

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

Denna formel är uttrycket vi får efter att ha multiplicerat samman alla transformationer. Hela kedjan av transformationer ovan görs per *ben*. Detta görs på CPU'n, inte i shadern. Formeln ovan uttrycker därmed det som görs i shadern.

De resulterande matriserna M_i skickas som *uniform*-variabler. Vikterna w_i varierar per vertex och skickas därför som *attribute*.

6. Representation av rotation

Ovan har jag antagit att rotationer representeras med matriser. Det kan givetvis vara lämpligt att representera animationsrotationerna (de variabla) med kvaternjoner eller rena vektorer, i syftet att interpolera dessa. Detta är dock ett annat problem, vars lösning beror på hur animationerna påförs. Det vore direkt olämpligt att blanda in den frågan här. I stället antar vi att R_{anim} kommer från en extern källa, som till exempel kan vara konvertering från en annan representation.

7. Slutsatser

Detta beskriver, i lite mer detalj, den grundläggande skinning-algoritmen. Bortom denna finns många förbättringar som vi inte tar upp här.

8. Referenser

Jag har konsulterat följande källor:

- [1] Lewis, Cordner, Fong: "Pose Space Deformation: A New Unified Approach to Shape Interpolation and Skeleton-Driven Deformation", SIGGRAPH, 2000
- [2] Gould, "Complete Maya Programming", Morgan Kaufmann 2003
- [3] "The Art Of Maya", Alias learning tools, 2005
- [4] Game Programming Gems 1, Charles River Media, 2000

Tack till Johan Hedborg för diskussioner och synpunkter!