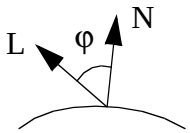
Lösningsförslag till omtentamen

TNM077 samt TNM008

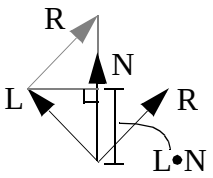
2005-06-10 kl 8-12

Uppgift 1 (7 p)

a) Se figur. Skalärprodukten $N \bullet L$ är för normerade vektorer med $|N| = |L| = 1$ lika med cosinus för vinkeln mellan vektorerna, $N \bullet L = |N||L|\cos\varphi$. Intensiteten för det infallande ljuset mot en yta är proportionellt mot ljusets intensitet och cosinus för ljusets infallsvinkel mot ytan, och därför är Phongs ekvation en bra modell här. (2 p)



b) Med hjälp av figuren bredvid fås sambandet $R + L = 2(L \bullet N)N$. Man utnyttjar att infallsvinkeln för L är lika med reflexionsvinkeln för R, vilket medför att $R + L$ är parallell med N, och projektionen av L på N och projektionen av R på N är lika. Enkel omskrivning ger uttrycket $R = 2(L \bullet N)N - L$. (2 p)



c) Den diffusa reflexion som beskrivs av k_d har att göra med materialets egenskaper även en liten bit ner under ytan, men k_s beskriver den spekulära reflexionen precis i ytan. Dessa reflexioner har olika orsaker, olika egenskaper, olika intensitet och oftast olika färg. Att alltid sätta dem lika skulle vara alltför begränsande. (1 p)

d) Man kan tolka alla intensiteter och reflexionskoefficienter som vektorer med samplad spektral färginformation, till exempel tre tal för färgkanalerna R, G och B. När man multiplicerar och adderar intensiteterna och reflexionskoefficienterna gör man det komponentvis för varje färgkanal. Ekvationen kan även tolkas som den reflekterade intensiteten för en enstaka färgkanal, varvid samma beräkning upprepas för varje färgkanal. (2 p)

Uppgift 2 (8 p)

a) Tangentriktningen $\bar{v}(u)$ ges av derivatan med avseende på parametern u:

$$\bar{v}(u) = \frac{d}{du}\sum_{i=0}^{3} B_i(u)\bar{p}_i = \sum_{i=0}^{3} B_i'(u)\bar{p}_i, \text{ där } \quad B_0'(u) = -3 + 6u - 3u^2 \quad B_2'(u) = 6u - 9u^2$$
$$B_1'(u) = 3 - 12u + 9u^2 \qquad B_3'(u) = 3u^2$$

Béziér-kurvor kan definieras i godtyckligt många dimensioner. För att uttrycket ovan skall avse en 3D-vektor krävs att kontrollpunkterna $\bar{p}_i$ är tredimensionella ortsvektorer. (3 p)

b) Det som saknas är en definition av vad som skall vara "upp" i bilden. Kameran kan roteras fritt runt sin riktningsvektor men fortfarande peka åt samma håll från samma punkt. Ett vanligt sätt att ange "upp" i bilden är att ange en vektor vars projektion i bilden skall vara vertikal, en så kallad "up-vector". Detta kan vara en av koordinataxlarna, ofta världens "upp". (2 p)

c) Tangentvektorn som tecknades ovan har ett belopp som uppenbart beror av parametern u, det är inte konstant längs kurvan. Ett tydligt exempel är att derivatan i punkterna $u = 0$ och $u = 1$ beror av helt olika kontrollpunkter och alltså kan varieras oberoende av varandra. Kamerans momentana hastighet längs kurvan blir alltså inte konstant om man varierar parametern u med konstant hastighet. (För att få en konstant banhastighet måste man animera u med en varierande hastighet som är omvänt proportionell mot tangentvektorns belopp.) (2 p)

Uppgift 3 (4 p)

Himlen är stor och finns i alla riktningar. Om kameran skall panorera i scenen så behövs en väldigt stor texturbild för himlen, som dessutom måste mappas korrekt på insidan av en sfär eller något annat virtuellt objekt som täcker alla synliga riktningar. Detta är inte helt lätt, och om vanliga foton skall användas som grund behöver man ta många bilder åt alla håll och på något sätt klistra ihop dem till en. Det duger inte alls med en bild av ett enda moln åt ett enda håll, och det innebär ett avsevärt manuellt arbete att skapa en tillräckligt bra texturbild. En procedurell textur kan däremot genereras helt automatiskt, och är förhållandevis lätt att mappa på en sfär eftersom den kan definieras i tre dimensioner i stället för som en platt bild två.

En typisk bild visar bara några få procent av hela himlen. Om man vill ha en högupplöst himmel måste man lagra väldigt mycket data varav man inte utnyttjar mer än ett litet stycke i taget. En procedurell bild lagras inte alls, utan beräknas bara där man behöver den.

Om man vill zooma in på himlen måste man ha en högupplöst textur. En lagrad textur har alltid en begränsad upplösning och sätter gränser för hur nära man kan zooma in på den utan att förlora skärpa och kvalitet. En procedurell textur kan däremot beräknas i godtycklig upplösning när man renderar bilden, och har därför inga principiella problem att anpassa sig efter en godtycklig inzoomning.

Om man vill variera himlens utseende kan man enkelt ändra parametrarna i den procedurella texturen och få större eller mindre moln, tätare eller glesare molntäcke och andra färger. En ritad bild skulle däremot behöva ritas om från början. En procedurell bild kan även animeras så att molnen rör sig på himlen och förändras över tiden, genom att helt enkelt animera parametrarna. En lagrad animerad, högupplöst filmsekvens av himlen skulle kräva ett jämförelsevis enormt arbete och en mycket stor lagringskapacitet.

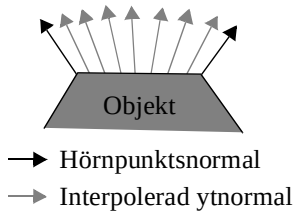
Den procedurella bilden av himlen kanske inte är så fantastiskt naturtrogen, men eftersom en kompetent person anser att den duger så är den förmodligen tillräckligt bra för den scen där den skall användas. Det är inte alltid man behöver eller ens vill ha extrem realism. Om resten av scenen är helt syntetisk och inte har en mycket hög grad av realism kan det till och med vara störande att ha en helt fotorealistisk himmel. (4 p)

Uppgift 4 (3 p)

a) Det som krävs för ljussättning är *normaler* till ytan (lagras oftast som normalvektorer i hörnpunkterna) och information om ytans *reflexionsegenskaper* (åtminstone en färg, men oftast flera parametrar som beskriver en reflexionsmodell för ytan). (2 p)

b) För att lägga en textur på objektet behöver man dessutom *texturkoordinater*, som oftast lagras som en extra uppsättning 2D-koordinater i varje hörnpunkt. (1 p)

Uppgift 5 (6 p)



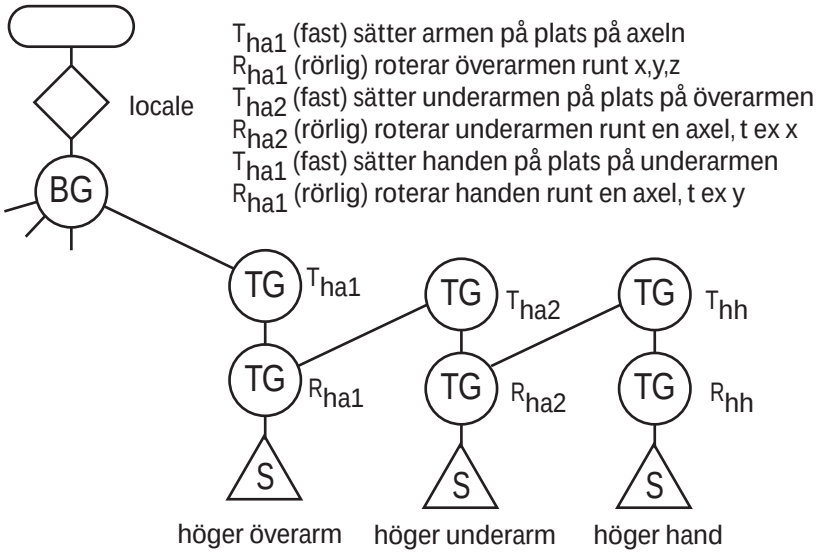
a) Mekanisk konstruktion med CAD avser att modellera ett objekts exakta form för att sedan kunna tillverka det med stor precision efter digitala ritningar. Datorgrafik däremot syftar till att efterlikna det mer övergripande utseendet hos objekt, och då behöver man inte alls vara lika noga. Krökta ytor kan utan problem approximeras med ett antal plana polygoner, förutsatt att det ser OK ut. En förutsättning för att det skall se OK ut är att man i renderingen lyckas dölja de skarpa kanterna som uppstår mellan polygonerna. Detta görs med så kallad interpolerad shading, där man på något sätt låter ytnormalerna följa det verkliga objektets form snarare än den kantiga approximationen. Phong shading (se figur) lagrar till exempel normaler för ytan i varje hörnpunkt, och ytnormalen inom varje polygon interpoleras fram baserat på de normalerna i de närliggande hörnpunkterma.På detta sätt kommer ljusreflexionen att se ut som om den kom från en mjukt krökt yta, det är bara silhuetten av objektet som kan se litet kantig ut. (4 p)

b) Polygonmodeller är en enklare men ändå generell representation av ytor, som är avsevärt enklare och snabbare att lagra, animera, redigera och rendera än olika sorters parametriska ytor. I stort sett all rendering av datorgrafik sker dessutom genom att allt konverteras till polygoner, speciellt för rendering i realtid med modern grafikhårdvara, så polygoner är en representation som är mer direkt användbar för rendering. Det är dessutom inte alltid som verktygen i 3D-modelleringsprogram utanför CAD-området är särskilt lämpade för att skapa och hantera parametriska ytor. Polygonmodeller används därför ofta där parametriska ytor egentligen skulle vara mer geometriskt korrekta. Det väsentliga i datorgrafik är att det ser tillräckligt bra ut, inte att det är 100% korrekt. (2 p)

Uppgift 6 (5 p)

a) Transformationerna utförs i den ordning TG-noderna länkas till varandra i grafen, från den nedersta transformationen (närmast objektet) till den översta (närmast scenens rot). De slutliga transformerade koordinaterna för höger hand blir därför $T_{ha}R_{ha}T_{hh}R_{hh}\bar{x}_i$. (2 p)

b) En riktig mänsklig axelled är en kulled som kan rotera fritt (inom vissa gränser) kring tre axlar. Armbågen är en enkel gångjärnsled som kan rotera kring en enda rotationsaxel. Scengrafen för en sådan mer ledad arm skulle se ut enligt figuren nedan. Exakt vilka koordinataxlar som rotationerna sker kring beror på hur man väljer de lokala koordinaterna för Shape-noderna. (3 p)



Uppgift 7 (8 p)

a) Radiosity kan tillföra diffusa interreflexioner mellan objekt och ge mycket mjuka, naturtrogna skuggor i en diffust belyst scen.

b) Man börjar med att på något sätt dela upp alla ytor i scenen i ett stort antal små ytelement, ofta baserat på polygonerna i scenen. Därefter räknar man ut (en approximation av) den relativa graden av synlighet mellan alla ytor i scenen. Den rymdvinkel som upptas av yta j sett från yta i räknas om till en formfaktor b_{ij}. Reflexionskoefficienten för den diffusa reflexionen hos varje yta i modelleras som R_i, och den eventuella egenemissionen hos yta i (vilka ytelement som helst kan vara ljuskällor) betecknas ε_i. Radiosityekvationen tecknas då

$$I_i = \varepsilon_i + R_i \sum_j I_j b_{ij}$$

En ekvation enligt ovan för varje ytelement ger ett ekvationssystem med lika många kopplade ekvationer som man har ytor i scenen. Med matrisnotation för formfaktorerna så att b_{ij} är elementet i B på rad i, kolumn j, I_i och ε_i är elementen på rad i i kolumnvektorerna I respektive ε och R är en diagonalmatris där R_i är elementet på rad i, kolumn i så blir detta helt enkelt:

$$I = \varepsilon + RBI$$

(4 p)

c) Förutom att algoritmen är beräkningstung så finns till exempel följande nackdelar:

Radiosityalgoritmen är en global metod som beräknar belysningen för samtliga ytor i hela scenen, även om kanske bara en liten del av dessa är synliga och har någon effekt på den slutliga bilden. Mycket av beräkningsarbetet utförs alltså i onödan.

Radiosity har ingen möjlighet att modellera något annat än perfekt diffust ljus och diffus reflexion, så eventuella blänk och speglingar, samt skarpa skuggor och andra effekter av riktat ljus måste särbehandlas och beräknas på annat sätt.

Formfaktorerna är starkt beroende av scenens geometri, så om något objekt rör sig i scenen måste i princip hela beräkningsarbetet med formfaktorerna göras om. Radiosity är därför inte särskilt lämpad för att rendera animerade scener där mycket rör sig.

För att kunna rendera en scen med radiosity måste scenen vara noggrant modellerad så att alla objekt har en väldefinierad insida och utsida sedda från alla möjliga håll, inte bara från kameran.

Det blir oproportionerligt mycket svårare att rendera scener med många polygoner eftersom antalet formfaktorer växer med kvadraten på antalet polygoner. Man kan behöva ta till komplicerade algoritmer för att förenkla geometrin i scenen inför en radiosityrendering.

Radiosityalgoritmen kan inte hantera detaljerade skuggor och andra snabba variationer i belysning inom en stor polygon. Där det händer mycket i belysningshänseende i scenen måste stora polygoner delas upp i mindre delar. Hur denna uppdelning skall ske är inte känt förrän man påbörjat renderingen. Den måste alltså ske dynamiskt under renderingen, och behöver göras annorlunda om ljussättningen i scenen ändras. Algoritmen för detta är ganska komplicerad. (2 p)