Lösningsförslag till
Omtentamen TNM077, 3D datorgrafik och animering
(samt även TNM008, 3D datorgrafik och VR)
2004-04-23

Lösningsförslag, inte facit

Dessa lösningsförslag är just förslag till lösningar, inte något facit. Det finns på många av de beskrivande frågorna flera olika svar som ger full poäng, och på alla frågor finns olika grader av nästan rätta eller inte helt fullständiga svar som fortfarande ger hyggligt många poäng. Dessa svar är ibland tämligen kortfattade, men tillräckligt utförliga för att de skulle ha gett full poäng i bedömningen.

Uppgift 1 (10 p)

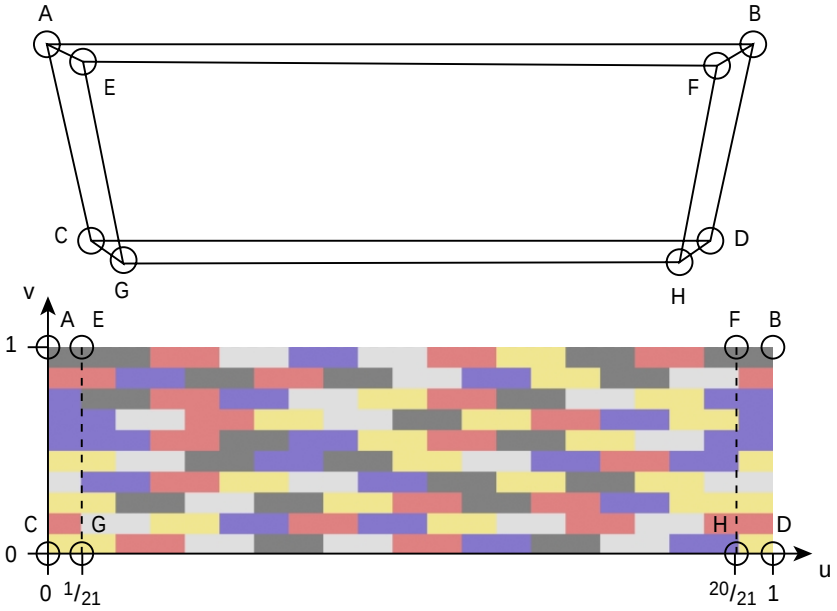
a) Bitens fyrkantiga bas är modellerad med åtta hörnpunkter och sex rektanglar som vardera består av två trianglar. För de cylindriska knopparna består mantelytan av 16 rektanglar om vardera två trianglar, som ritas mellan 16+16=32 hörnpunkter. Om man sedan antar att det inte finns någon bottenyta på knopparna (botten syns ju ändå inte) och att den 16-sidiga polygonen på toppen består av 16 trianglar utladga som tårtbitar med en gemensam extra hörnpunkt i mitten, så blir det totala antalet hörnpunkter 8 + 8*(32+1) = 272 och antalet trianglar 2*6 + 8*(2*16+16) = 396. (2 p)

b) Hela muren består av 400 bitar * 396 trianglar per bit = 158400 trianglar. De allra flesta av dessa syns dock aldrig, oavsett från vilken vinkel man tittar. Den första och mest radikala förenklingen är att ta bort alla knoppar utom de som verkligen syns, alltså de 4*10*8 = 320 knopparna på det översta lagret bitar. Då blir man av med nio av tio lager med knoppar, vilket innebär 9*320 = 2880 knoppar, motsvarande 2880*48 = 138240 trianglar. Detta räcker säkerligen i sig för att göra scenen hanterlig. Om man vill förenkla modellen ytterligare kan man dessutom ta bort alla ytor som ligger platt emot varandra. De rektangulära topp- och bottenytor som inte syns utgör 9 lager * 2 ytor * 4 väggar * 10 bitar (endast det översta och nedersta lagrets ytor syns fortfarande), vilket motsvarar 9*2*4*10*2 = 1440 trianglar. De kortsidor som inte syns utgör 9 längdskarvar * 2 ytor * 4 väggar * 10 lager, *plus* 2*5*4 kortsidor som döljs av långsidorna på andra bitar i hörnen. Detta uppgår totalt till 9*2*4*10 + 10*4 = 760 kortsidor, alltså 2*760 = 1520 trianglar. Kvar blir endast 158400 - 138240 - 1440 - 1520 = 17200 trianglar, alltså bara drygt en tiondel av det ursprungliga antalet. (3 p)

c) Texturer kan användas till att sätta de olika färgerna på muren, så att varje plan yta på murens väggar kan modelleras med en enda rektangel som bara byter färg där skarvarna mellan bitarna fanns i ursprungsmodellen. Man behöver då bara 4*4*2 = 32 trianglar för samtlga väggar, oavsett hur stora de är. En på detta sätt fullt förenklad modell för en av de fyra väggarna ses i figuren nedan. (De tre andra väggarna blir likadana.) Knopparna behöver tyvärr fortfarande 320*48 = 15360 trianglar, och de blir fler om väggarna görs bredare, men det blir i alla fall betydligt enklare att göra muren *högre* om man så vill. (2 p)

d) Texturbilden ges lämpligen samma utseende som väggen sedd rakt från sidan. Det räcker faktiskt med en enda bild för texturen. Eftersom bitarna har samma färg på insidan och utsidan av väggarna kan man använda samma bild för båda sidorna, spegelvänd på baksidan. Om man tittar noga på bilden ser man dessutom att alla fyra väggarna faktiskt har exakt samma mönster av bitar. Man kan ana att den som modellerade muren var litet lagom lat och bara byggde en vägg, och sedan skapade fyra roterade kopior. Texturbild och texturkoordinater för en vägg visas i figuren nedan. Alla fyra väggarna har samma textur och texturkoordinater. Om väggarna hade varit olika hade det fortfarande räckt med en

texturbild, men den hade då fått göras fyra gånger så bred, så att alla fyra väggarna hade kunnat läggas på bredden i samma bild, och texturkoordinaterna hade blivit olika i u-led för de fyra väggarna. (3 p)



Uppgift 2 (8 p)

a) Raytracing. (1 p)

b) Se bokens framställning och föreläsningsanteckningar. “Ray tree” är en väsentlig figur. (4 p)

c) X anger det maximala rekursionsdjupet för den ursprungliga strålens förgreningar i reflexion och brytning. (2 p)

d) Ett alltför stort rekursionsdjup kan ge väldigt långa renderingstider, utan att bilden bli så mycket bättre. Hur stort djup man behöver beror till stor del på den scen man renderar, så det finns inget allmängiltigt svar på vilket värde som är lämpligt eller lagom att använda. (1 p)

Uppgift 3 (9 p)

a) Scengrafen är mycket rättfram och består av en rak kedja av noder helt utan förgreningar. Att rita den lämnas som övning åt läsaren. (Jag hinner tyvärr inte rita rent figuren för detta dokument.) (4 p)

b) Man kan i Java3D koppla transformationer och andra dynamiska delar av scengrafen till bivillkor genom att använda så kallade *behaviors*. Båda klons griparmar skulle kunna roteras via var sitt behavior beroende på en och samma underliggande styrbara parameter som kontrollerar öppning och slutning, där deras transformationer skulle fås att rotera åt olika håll när parametern ändrades. Bivillkoret att klon alltid skall hänga rakt ner kan med fördel också uttryckas som ett behavior: man tillåter animering av själva armen till godtycklig position, men den transformationsgrupp som bestämmer hur klon skall vara orienterad beräknas alltid automatiskt via ett behavior så att klon alltid pekar rakt nedåt. (2 p)

c) Man kan enkelt lägga in en transformationsgrupp ovanför hela Lego-modellen i scengrafen som utför en likformig skalning med faktorn 0.008. Inget annat i grafen behöver ändras (1 p)

d) Invers kinematik medger att hierarkiska ledade strukturer animeras genom att bestämma rörelsen för deras ändpunkter i stället för att uttryckligen ange den exakta rotationen för varje led. Eftersom kranens griparm har ett verktyg med vilket man förmodligen vill kunna manipulera objekt (timmerstockar) så vore det prektiskt om man direkt kunde positionera gripklon och låta matematiska beräkningar (invers kinematik) göra att resten av armen automatiskt följer med i rörelsen. (2 p)

Uppgift 4 (13 p)

a) En normalvektor N beräknas smidigt genom en kryssprodukt mellan vektorer längs två av triangelns kanter, förslagsvis $N = (P_2 - P_1) \times (P_3 - P_1)$. Normering till en vektor $\hat{N}$ av längden 1 görs genom att dividera N med dess belopp: $\hat{N} = N / |N|$. (1 p)

b) Genom att införa ljuskällans intensitet I_0 och triangelns diffusa ytreflektans R_d, $0 \le R_d \le 1$, kan man teckna den diffusa reflexionens intensitet I_d enligt: $I_d = I_0 R_d (\hat{N} \bullet \hat{L})$, där $\hat{L} = (P_L - P) / |P_L - P|$. (2 p)

c) Skalärprodukten $\hat{N} \bullet \hat{L}$ blir i detta fall negativ, och så även den reflekterade ljusintensiteten. Negativa ljusintensiteter finns inte i verkligheten, så detta är inte fysikaliskt korrekt. Om skalärprodukten blir negativ skall den reflekterade intensiteten i stället sättas till noll i beräkningen. (2 p)

d) Eftersom riktningen till ljuskällan varierar med positionen i världen så kommer punkterna inom triangelns yta inte att ha samma vektor $\hat{L}$ i beräkningen. Intensiteten blir därför olika i olika punkter. (2 p)

e) Man kan låta ljuskällans intensitet I_0 bero av avståndet d från ljuskällan till punkten: $d = |P_L - P|$. Detta avstånd beräknas redan när man normerar vektorn $\hat{L}$. Det enda som behöver utföras ytterligare är att multiplicera intensiteten med en extra faktor som beror av d, t ex enligt $I_d = I_0 \frac{1}{d^2} R_d (\hat{N} \bullet \hat{L})$ (2 p)

f) Den speglande reflexionen kan beräknas som en egen term I_s enligt $I = I_d + I_s$, där $I_s = I_0 R_s (\hat{V} \bullet \hat{R})^n$, R_s är den speglande reflektansen hos ytan (som kan vara annorlunda än den diffusa reflektansen R_d), n är en parameter som bestämmer hur stort det speglande blänket blir, $\hat{V} = (P_V - P) / |P_V - P|$ och $\hat{R}$ kan beräknas enligt $\hat{R} = 2(\hat{N} \bullet \hat{V})\hat{N} - \hat{V}$ (se härledning i boken, i föreläsningsanteckningar eller i svaren till tentan i TNM008 från 2003-04-23). (4 p)