Tentamen

TNM077 3D-datorgrafik och animering

(även omtentamen TNM008 3D-datorgrafik och VR)

2005-03-17 kl 14-18

Inga hjälpmedel

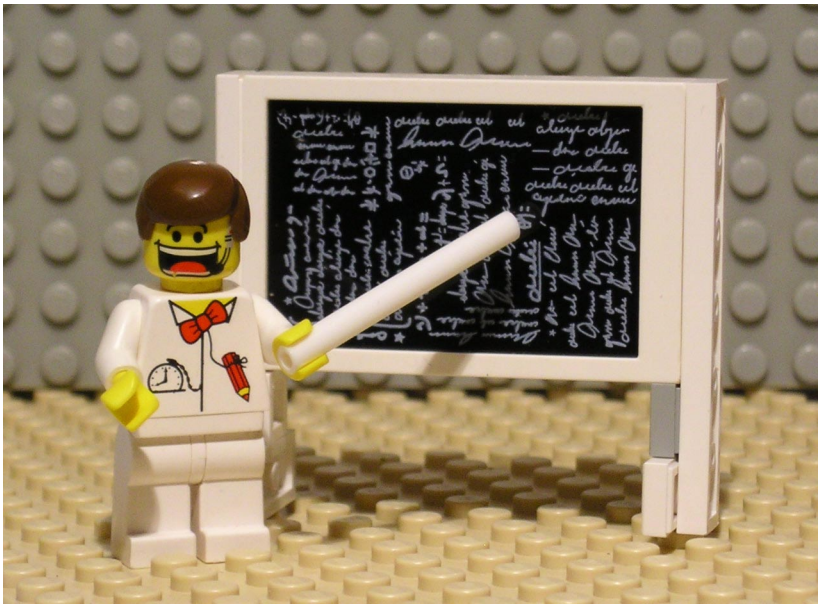
Denna tentamen innehåller fem uppgifter som tillsammans kan ge maximalt 40 poäng.
För betyget G (registreras som sifferbetyg 3) krävs minst 21 poäng.
För betyget VG (registreras som sifferbetyg 5) krävs minst 31 poäng.

Uppgifterna står inte i någon särskild ordning, så läs igenom samtliga uppgifter innan du börjar. Svara kortfattat men rimligt uttömmande på frågorna. Rita tydliga figurer där det gör framställningen klarare. Förklara införda beteckningar och motivera eventuella beräkningar så att de är lätta att följa.

Flera uppgifter kräver beräkningar. Någon uppgift eller två kräver dessutom ett visst mått av eftertanke och egna idéer med teorin som grund. Samtliga uppgifter kan lösas med endast de grundkunskaper som kursen avsett att förmedla. Vissa uppgifter kan faktiskt lösas helt med endast förkunskaperna från grundkursen.

Om något är oklart, fråga mig. Jag kommer in vid flera tillfällen under tentamenstiden för att svara på eventuella frågor.

Lycka till!



Stefan Gustavson

Uppgift 1 (9 p)

Phongs reflexionsmodell är en enkel men ofta använd modell för ljusreflexion. En vanlig form för denna modell är:

$$I = I_a k_a + I_d k_d (N \bullet L) + I_s k_s (R \bullet V)^n$$

a) Förklara vad samtliga tre termer i ekvationen avser att modellera, och hur de ingående materialparametrarna inverkar på det visuella resultatet. (4 p)

b) Samtliga termer innebär mer eller mindre grova förenklingar av verkligheten. Förklara vilka förenklingarna är! (3 p)

c) Ekvationen ovan gäller *en* ljuskälla. Hur hanteras *flera* ljuskällor? Visa med ekvation! (1 p)

d) Ekvationen ovan är egentligen inte korrekt. De två sista termerna kan bli positiva eller negativa, men det finns inget negativt ljus, och potenser av negativa tal blir komplexa tal om man inte har heltalsexponenter. Hur räknar man *egentligen* ut ljusintensiteten I? (1 p)

Uppgift 2 (8 p)

Ett segment av en kubisk Béziérkurva har ekvationen

$$\bar{p}(u) = \sum_{i=0}^{3} B_i(u)\bar{p}_i, \qquad 0 \le u < 1$$

$$B_0(u) = (1-u)^3,\ B_1(u) = 3u(1-u)^2$$
$$B_2(u) = 3u^2(1-u),\ B_3(u) = u^3$$

Kurvans riktning (dess tangent) i varje punkt längs kurvan ges av derivatan med avseende på u:

$$\bar{v}(u) = \frac{d}{du}\bar{p}(u) = \frac{d}{du}\sum_{i=0}^{3} B_i(u)\bar{p}_i$$

Man vill oftast ange en sammanhängande kurva som en följd av flera kurvsegment.

a) För att de två kurvsegmenten skall hänga ihop, alltså att kurvan är kontinuerlig precis i skarven mellan två segment, måste vissa villkor för kontrollpunkterna vara uppfyllda! Härled villkoren utifrån kontinuiteten, eller ansätt villkoren och visa att de ger kontinuitet. (2 p)

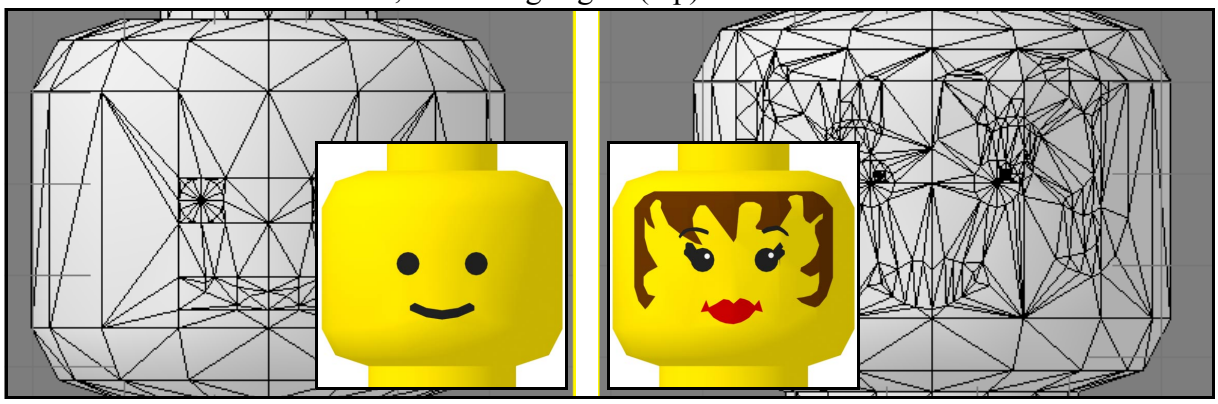
b) För att det inte skall vara något hörn på kurvan och den skall se mjuk ut skall även derivatan vara kontinuerlig i skarven mellan två segment. Ange vilka ytterligare villkor för kontrollpunkterna som i så fall skall vara uppfyllda! Härled villkoren utifrån kravet på kontinuitet hos derivatan, eller ansätt villkoren och visa att kravet är uppfyllt. (4 p)

c) Ett vanligt sätt att göra skarven “hyfsat mjuk” rent visuellt är att kräva att derivatan i skarven har samma riktning men olika belopp (längd) för de två segmenten. Man tillåter alltså en viss begränsad diskontinuitet hos derivatan i skarven. Vilka villkor gäller då för kontrollpunkterna? Visa med ekvation eller en tydlig och förklarad figur. (2 p)

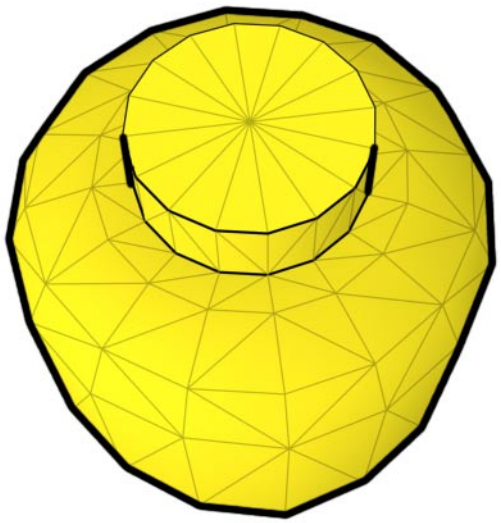
Uppgift 3 (8 p)

En ganska stor grupp löst organiserade Lego-entusiaster över hela världen har skapat ett programvarupaket kallat "LDraw" för att bygga med Lego virtuellt. En stor del av alla de tusentals olika Legobitar som tillverkats har modellerats som polygonmodeller i 3D, och det finns några enkla CAD-liknande program för att bygga samman bitarna till modeller och rendera bilder av dem. Projektet har pågått sedan 1990-talets mitt, då 3D-grafik fortfarande var svårt för persondatorer att klara av, och filformatet "LDR" som skapats för att representera Legobitarna har vissa egenheter.

a) Alla polygoner i LDR-formatet är enfärgade. Legobitar med tryckta mönster definieras med hjälp av extra polygoner för mönstret, till exempel som visas i figuren nedan. För enkla mönster som det till vänster är det en rimlig metod, men för mer komplicerade mönster i flera färger som det till höger blir det väldigt många polygoner i modellen. Vilken metod skulle kunna ha använts i stället om man hade tänkt på det när man definierade LDR-formatet? Kan du ändå se någon fördel med den valda metoden, även i dagsläget? (3 p)



b) Eftersom de flesta LDraw-program är CAD-verktyg där bitarna endast ritas ut schematiskt i så kallat "wire-frame"-läge finns det i filformatet ett särskilt attribut för alla kanter mellan polygoner. Många av kanterna är så kallade "mjuka kanter", som ligger inom en polygonapproximation av en verklig mjuk yta (de grå linjerna i figuren till höger), och i normala fall ritas inte dessa kanter ut. Övriga kanter är "hårda kanter" som är skarpa i verkligheten (de tunna svarta linjerna i figuren), och dessa ritas alltid ut som streck. *Om en "mjuk kant" däremot hamnar på silhuetten av objektet skall den ritas som ett streck* (de tjocka svarta linjerna i figuren). Exakt vilka kanter som hamnar på silhuetten varierar förstås med betraktningsriktningen. Föreslå en fullständig algoritm för att beräkna om en viss kant ligger på silhuetten av ett objekt! Algoritmen skall utgå från polygonrepresentationen av Legobiten och en punkt som anger betraktarens position. Den skall inte förutsätta någon rendering, utan endast använda geometriska beräkningar i 3D. Polygonmodellen består av en lista över alla hörnpunkter, en lista över alla kanter och en lista över alla trianglar, med hörnpunkterna för trianglarna i moturs ordning sett från triangelns framsida. Eftersom modellen läses in och lagras som en datastruktur i ett program kan godtyckliga sökningar göras i dessa listor. (5 p)



Verkliga sådana här bitar finns i tentasalen

Uppgift 4 (8 p)

En av 2005 års nyheter i Lego-sortimentet var en brandbil som köpts även av väldigt stora barn, däribland er examinator. Modellen ser ut enligt bilden nedan, och finns också i tentamenslokalen. De rörliga delarna är ganska många, men modellen har ändå en enkel struktur som lämpar sig ypperligt att modellera med en scengraf. Vi vill kunna animera en modell av bilen med Java3D. Vi begränsar rörligheten något, och nöjer oss med följande funktioner:

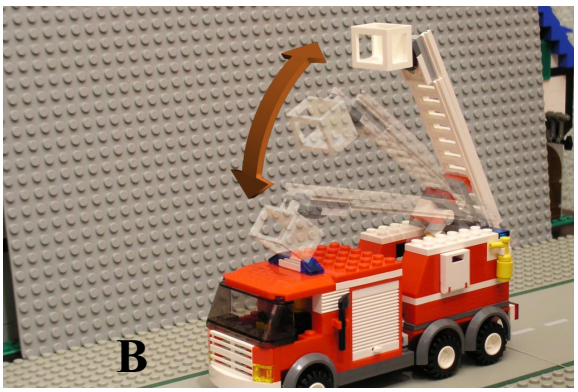
- De två luckorna bakom förarhytten skall kunna öppnas och stängas. (Bild A nedan)
- Stegen skall kunna resas upp och fällas ner. (Bild B)
- Basen för stegen skall kunna rotera åt vänster och höger i horisontalplanet. (Bild C)
- Stegen skall kunna dras ut och skjutas ihop genom att dess övre del förskjuts. (Bild D)

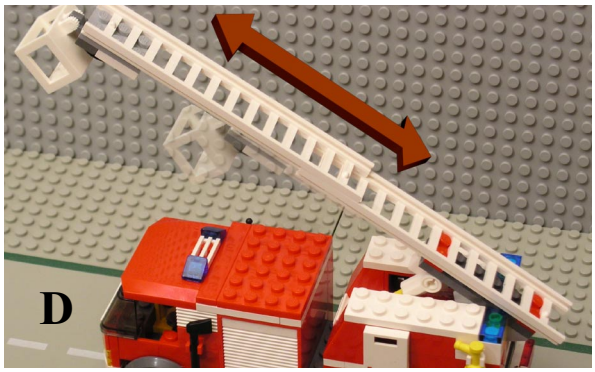
a) Rita en scengraf för Java3D som enkelt möjliggör alla de ovanstående animeringarna! Rita hur du väljer ditt koordinatsystem, visa schematiskt hur dina Shape-noder ser ut och var de har sitt lokala origo, och rita hela scengrafen ända från roten. Förklara för varje rörlig transformation vad den styr och hur den varierar, och förklara för varje fast transformation varför den finns där och vad den utför. Tag med samtliga dina definierade Shape-noder i figuren, och förklara tydligt vilka verkliga objekt de motsvarar. (5 p)

b) Antag att man förutom de ovanstående animeringarna även vill kunna låta bilen åka framåt och bakåt. Visa hur grafen ovan skall ändras för att medge en sådan transformation! Du behöver inte ta hänsyn till att hjulen egentligen skall snurra när bilen rör sig. (1 p)

c) När bilen rör sig skall hjulen egentligen snurra runt sina respektive axlar, samtidigt och lika mycket. Rotationsvinklar anges i Java3D i radianer. Hur stor vinkel φ i radianer skall ett hjul med radien r rotera när det rullar sträckan x om det inte skall slira mot underlaget? (1 p)

d) Korgen längst upp på stegen skall egentligen kunna rotera i vertikalplanet, och alltid peka rakt nedåt så att minifiguren i den kan stå säkert. Ange hur scengrafen skulle förändras för att tillåta denna rörelse, och hur den extra transformationen skall beräknas för att korgen alltid skall vända samma sida nedåt oavsett stegens vinkel. (1 p)





Uppgift 5 (7 p)

De enkla lokala belysningsmodellerna från datorgrafikens tidiga historia på 1970- och 1980-talen används fortfarande ofta för realtidstillämpningar, men i sammanhang där man har längre tid på sig för renderingen tar man numera nästan alltid hjälp av någon eller några globala belysningsmodeller (global illumination models), exempelvis "strålföljning" (raytracing).

a) Förklara vad raytracing kan tillföra i en renderad bild som är svårt eller omöjligt att göra med lokala ljusmodeller liknande Phongs belysningsmodell från uppgift 1. (2 p)

b) Förklara på ett tydligt sätt (självklart med en eller flera figurer) hur den klassiska raytracing-algoritmen ("Whitted raytracing", invers raytracing) gör för att beräkna dessa effekter! (4 p)

c) Vilken väg en enstaka stråle tar efter en brytning eller en reflexion är egentligen inte särskilt mycket krångligare att beräkna än en lokal ljusmodell. I en enkel scen är det heller inga större problem med att räkna på vilka objekt som strålen kolliderar med. Förklara varför det trots detta kan ta extremt lång tid att rendera till exempel glas med raytracing, även om scenen är väldigt enkel och bara innehåller ett enda glasobjekt mot en enkel bakgrund. (1 p)