Tentamen TNM061, 3D-grafik och animering för MT2

Tisdag 3/6 2014 kl 8-12

TP51, TP52, TP54, TP56, TP41, TP43

Inga hjälpmedel

Tentamen innehåller 8 uppgifter, vilka tillsammans kan ge maximalt 50 poäng.

För betyg G (registreras som sifferbetyg 3) erfordras minst 24 poäng.
För betyg VG (registreras som sifferbetyg 5) erfordras minst 35 poäng.

Deluppgifterna bygger inte alltid direkt på varandra, så läs t ex b) även om du inte svarar på a).

Svara kortfattat men rimligt fullständigt på uppgifterna. Förutsätt inte att läsaren redan kan svaret, utan visa med ditt svar att *du* kan ämnet och att du förmår förklara det för andra. Förklara införda beteckningar och ekvationer, och redovisa eventuella beräkningar. Rita figurer där det gör svaret tydligare. Glöm inte heller bort att förklara dina figurer i ord eller i ekvationer.

För uppgifter som ger många poäng vill jag se rimligt utförliga svar för full poäng, men för uppgifter med enstaka poäng räcker det med korta svar. Även korta svar skall vara tydliga.

Jag kommer att befinna mig i närheten av tentamenslokalen och besöka den flera gånger under tentamenstiden för att svara på eventuella frågor. Kontakta tentamensvakten om du behöver få tag i mig mellan dessa tillfällen.

Lycka till!

Stefan Gustavson

Uppgift 1 (9 p). Phongs klassiska lokala reflexionsmodell med tre termer för allmänljus, diffus reflexion och spekulär reflexion brukar traditionellt presenteras med följande förenklade ekvation:

$$I = I_a k_a + I_d k_d (\hat{N} \cdot \hat{L}) + I_s k_s (\hat{R} \cdot \hat{V})^n$$

Om man verkligen ska implementera modellen i programkod i t.ex. en shader måste man tänka på flera saker som inte framgår av den förenklade ekvationen:

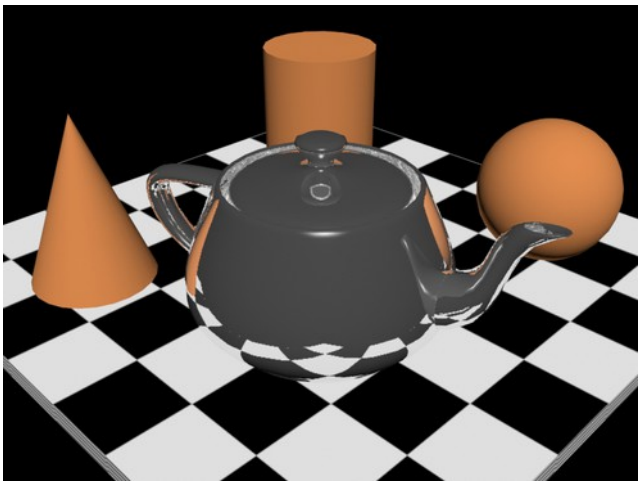
a) Ekvationen beskriver reflexionen som en skalär intensitet, I . Hur hanterar man färgat ljus och färgade ytor? Visa tydligt med en förklarad ekvation för åtminstone en av de tre termerna. (2 p)

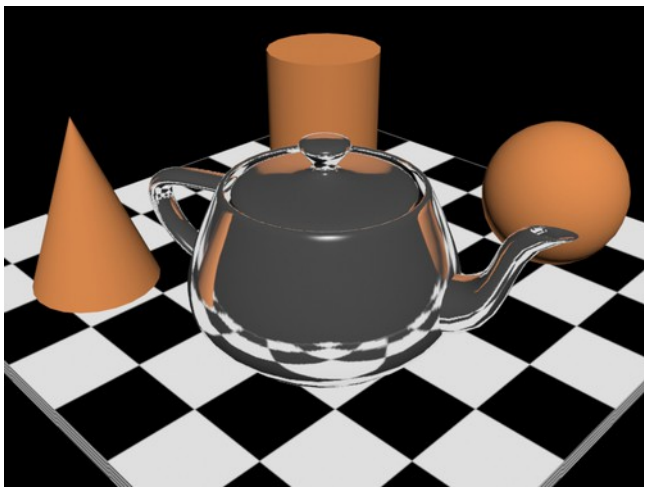
b) Vinkelberoendet för den diffusa termen beräknas med en skalärprodukt. Vad händer om vinkeln mellan de båda vektorerna är större än 90 grader? Hur hanterar man det? Hur väl motsvarar det vad som händer i verkligheten? (2 p)

c) Ekvationen beskriver inverkan av en enda ljuskälla. Hur hanterar man flera ljuskällor? Svara med en modifierad ekvation. (2 p)

d) Nästan samtliga ingående parametrar beror på olika sätt av vilken punkt $\vec{p} = (p_x, p_y, p_z)$ på ytan av ett visst objekt som man räknar på, men det framgår inte av ekvationen. För normalriktningen $\hat{N}$ är det ganska självklart att den beror på $\vec{p}$, men förklara varför var och en av I_d , k_d och $\hat{L}$ i den diffusa termen också bör kunna bero av $\vec{p}$ för att få god realism! (3 p)

Uppgift 2 (5 p). Nedanstående två bilder visar ett spegelblankt objekt i en enkel miljö. Den ena bilden har renderats med raytracing, den andra med en så kallad ”environment map”, även kallad ”reflection map”.





a) Vilken är vilken? Motivera ditt svar. (1 p)

b) En av metoderna är lämplig för realtidsrendering. Vilken, och varför? (2 p)

c) Den andra metoden är inte lämplig för realtidsrendering. Varför inte? (2 p)

Uppgift 3 (7 p). Raytracing (”strålföljning”) är en vanlig renderingsmetod.

a) Förklara den grundläggande principen för hur metoden fungerar. Förklara särskilt hur man beräknar reflexionen från speglande, genomskinliga och diffusa ytor . (5 p)

b) Förklara hur skuggor beräknas med raytracing. (2 p)

Uppgift 4 (6 p).

Spelet ”Elder Scrolls Online” som släpptes i april 2014 är ett så kallat MMO, massive multi-player online-spel. Hundratals spelare ska kunna delta i samma scen, där alla deras olika karaktärer visas samtidigt på skärmen. Modellerna för varje spelare behöver därför vara förhållandevis enkla. En bild av en sådan modell visas i bilden här bredvid.

De visuella effekterna av förenklingarna yttrar sig som följer (a, b, c). Beskriv i tekniska termer vad det är som förenklats, och varför just den förenklingen sparar arbete för datorn.

a) Modellerna ser litet kantiga ut. Inte så att det direkt stör, men i alla fall så att det syns. (2 p)

b) De olika kläderna och rustningarna som en spelare kan ha ser på nära håll ut som om de målats på modellerna. Det finns några få grundvarianter av själva modellen, och sedan är det bara mönstret som skiljer. (2 p)

c) När det är väldigt många spelare i bild blir de ännu kantigare, mönstret på ytan blir pixligt och suddigt, och blanka detaljer som t ex polerad metall på rustningar och spännen blir matta. (2 p)

Uppgift 5 (4 p). I bilden i föregående uppgift ser det på ljusreflexionen ut som om den blanka rustningen är skrovlig, men modellen är i själva verket slät. Det ser också ut som om det går vågor på vattnet i bakgrunden, men vattenytan är egentligen bara ett enkelt plan. För dessa effekter har man använt en metod som är väldigt vanlig inom datorgrafik. Vilken? Förklara någorlunda i detalj hur den fungerar. Rita figur. Du behöver inte gå in på några ekvationer.

Uppgift 6 (4 p). Videospelet ”Lego Marvel Super Heroes” släpptes under 2013. Några av de många spelbara figurerna visas i bilden. Mönstret på kroppen på figurerna är gjorda med bildbaserade texturer, men ansiktsdragen är gjorda med en ganska ansenlig mängd polygoner, trots att de flesta av ansiktena är tvådimensionella mönster på en helt slät yta. Det blir mer uppenbart när man ser figurerna animeras, men även i en stillbild på nära håll i hög upplösning syns det tydligt att mönstret för ansiktsdragen är gjort med polygoner och att mönstret på kroppen är gjort som en bildbaserad textur. Hur ser man det? Varför har man valt att göra så?

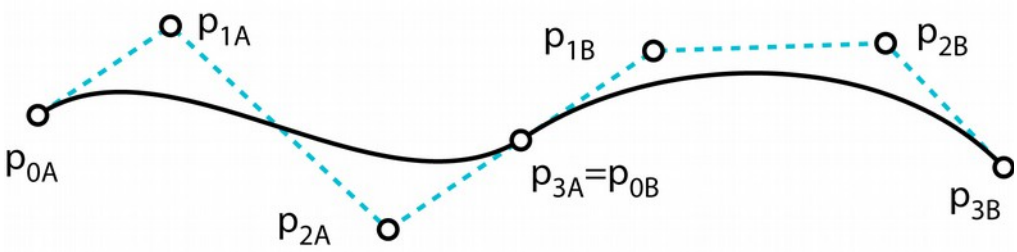
Uppgift 7 (6 p). En kubisk Béziérkurva definieras av ekvationen:

$$p(t)=(1-t)^3 p_0+3t(1-t)^2 p_1+3t^2(1-t)p_2+t^3 p_3\,,\ 0\le t\le 1$$

där p_i är kurvans kontrollpunkter, $i=0,1,2,3$.

Två kurvsegment A och B med kontrollpunkter p_{iA} och p_{iB} skall skarvas ihop till en sammanhängande bana så att segment B tar vid med $t=0$ där segment A slutar med $t=1$, och de kännetecknas av följande:

Segment A har sin sista kontrollpunkt på samma ställe som kurva B har sin första kontrollpunkt: $p_{3A}=p_{0B}$. Den tredje kontrollpunkten för segment A (p_{2A}) och den andra kontrollpunkten för segment B (p_{1B}) ligger lika långt åt var sitt håll längs en rät linje som går igenom $p_{3A}=p_{0B}$. (Se figur.)



a) Visa att den ihopskarvade kurvan är kontinuerlig i skarven. (2 p)

b) Visa att även derivatan för den ihopskarvade kurvan är kontinuerlig i skarven. (2 p)

c) Är dessutom andraderivatan kontinuerlig i skarven? Motivera ditt svar med en beräkning. (2 p)

Uppgift 8 (9 p). Nedanstående programkod ritar en teddybjörn med OpenGL och C++. Teddybjörnen har rörliga armar, ben och huvud. En verklig teddybjörn med samma rörelsemöjligheter finns i tentamenslokalen om du vill testa. (Det är helt OK att klappa honom också.) Rotationsaxlarna har förenklats något i koden för att göra den kortare.

Klassen Matrix implementerar en matrisstack enligt följande:

`Matrix.pop()`	Tag bort den översta matrisen från stacken.
`Matrix.push()`	Skapa en kopia av den översta matrisen och lägg den överst på stacken.
`Matrix.init()`	Tag bort alla matriser från stacken och lägg en enhetsmatris överst.
`Matrix.rotateX(phi)`	Multiplicera den översta matrisen med en rotationsmatris som roterar vinkeln phi runt x-axeln.
`Matrix.rotateY(phi)`	Multiplicera den översta matrisen med en rotationsmatris som roterar vinkeln phi runt y-axeln.
`Matrix.rotateZ(phi)`	Multiplicera den översta matrisen med en rotationsmatris som roterar vinkeln phi runt z-axeln.
`Matrix.translate(tx, ty, tz)`	Mutliplicera den översta matrisen med en translationsmatris.

För att rita objekt har man implementerat en enkel klass Mesh som renderas med metoden:

`Mesh.render(Matrix M);`

Matrisen högst upp på matrisstacken `M` används för att transformera objektet vid renderingen.

Arrayen `float pose[5]` innehåller de vinklar som beskriver rotationen för figurens leder.

```
Matrix M;
Mesh TeddyBody, TeddyHead,
 TeddyLeftArm, TeddyRightArm,
 TeddyLeftLeg, TeddyRightLeg;
float pose[5];
// (Code to create meshes left out)
M.init();
TeddyBody.render(M);
M.push();
M.translate(0.0, 0.0, 0.3);
M.rotateZ(pose[0]);
TeddyHead.render(M);
M.pop();
M.push();
M.translate(0.1, 0.0, 0.2);
M.rotateX(pose[1]);
TeddyRightArm.render(M);
M.pop();
M.push();
M.translate(-0.1, 0.0, 0.2);
M.rotateX(pose[2]);
TeddyLeftArm.render(M);
M.pop();
M.push();
M.translate(0.1, 0.0, 0.0);
M.rotateX(pose[3]);
TeddyRightLeg.render(M);
M.pop();
M.push();
M.translate(-0.1, 0.0, 0.0);
M.rotateX(pose[4]);
TeddyLeftLeg.render(M);
M.pop();
```

a) Rita en scengraf för att beskriva samma scen i ett scengrafsbaserat API, exempelvis Java3D. Det är inte så noga med exakt vilken notation du använder, men du skall visa tydligt hur scengrafen ser ut. (5 p)

b) När man multiplicerar en ny matris T med en existerande matris M på stacken, ska den nya matrisen då bli $T \cdot M$ eller $M \cdot T$? Motivera ditt svar! (2 p)

c) Med koden ovan kommer teddybjörnen att kunna röra på huvud, armar, ben, men den kommer att ”sitta fast” på en fix position i scenen. Visa vilka ändringar som behöver göras i koden för att även kunna flytta och rotera teddybjörnen. (2 p)