Tentamen TNM061, 3D-grafik och animering för MT2

Onsdag 20/8 2014 kl 14-18

SP71

Inga hjälpmedel

Tentamen innehåller 7 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 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:

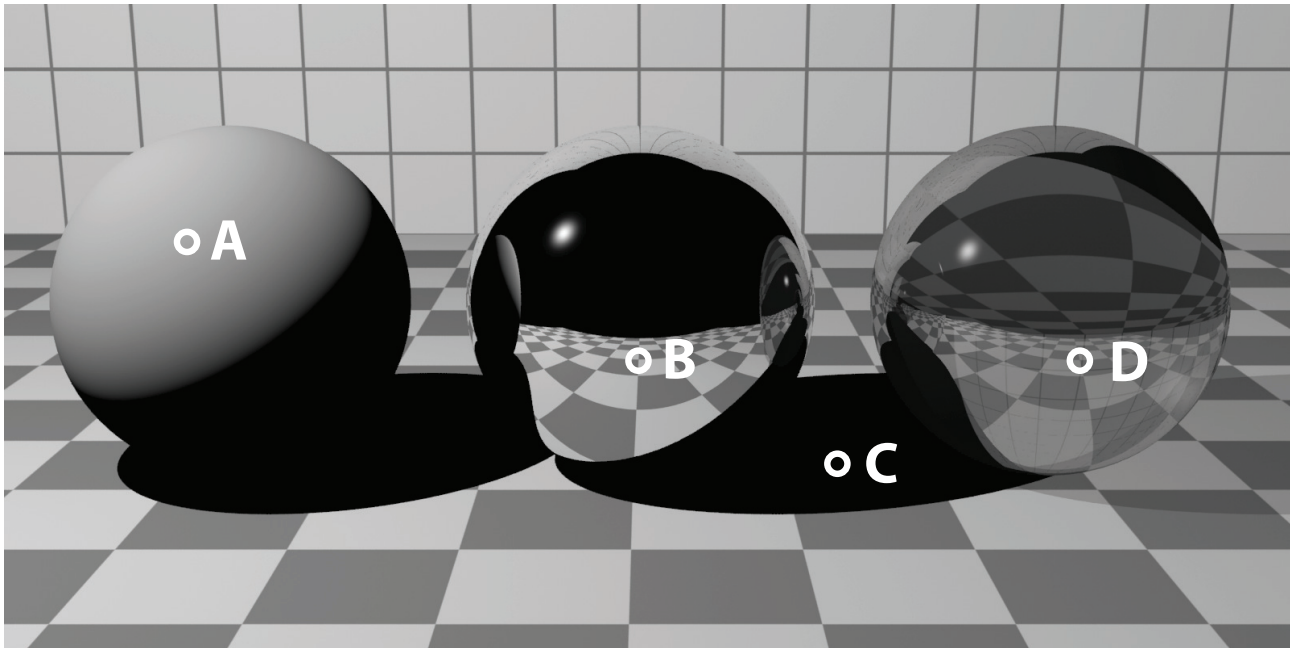
$$I = I_a k_a + I_d k_d (\hat{N} \cdot \hat{L}) + I_s k_s (\hat{R} \cdot \hat{V})^n$$

a) Förklara vad de fyra vektorerna $\hat{N}$, $\hat{L}$, $\hat{R}$ och $\hat{V}$ betecknar. Rita en tydlig figur. (3 p)

b) Flera av de ingående parametrarna beskriver ytans egenskaper, och dessa kan fås att variera över ytan med hjälp av en textur (*texture map*). Beskriv tydligt vilken visuell effekt man får av att koppla en textur till var och en av de tre parametrarna k_d, k_s och N. (6 p)

Uppgift 2 (9 p). Nedanstående bild visar en mycket enkel scen som renderats med raytracing. Scenen innehåller tre sfärer på ett rutigt golv, belysta av en enda ljuskälla. Sfären till vänster har en diffust reflekterande yta, sfären i mitten har en rent speglande yta och sfären till höger är avsedd att simulera glas.

Fyra ringar (A, B, C, D) markerar positionen för fyra pixels i bilden. För var och en av dessa pixels, beskriv vilka strålar som måste ha varit inblandade i renderingen av dem för att bilden skall ha kunnat få det utseende den har! Använd figurer i din förklaring. (A, B, C 2 p vardera, D 3 p)



Uppgift 3 (4 p).

Ange varför följande effekter är förhållandevis svåra och tidskrävande att beräkna med raytracing.

a) Skuggor med mjuka kanter. (2 p)

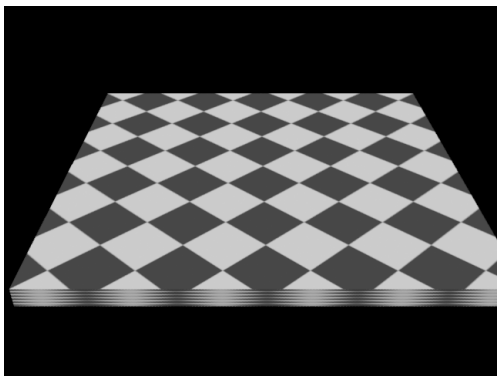
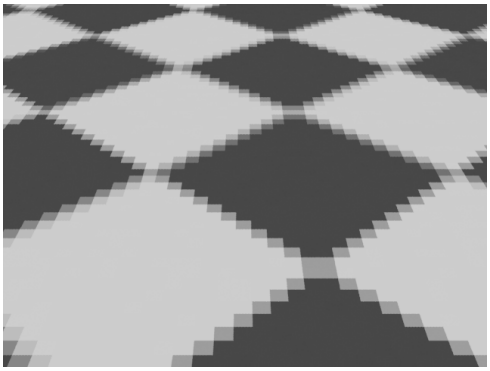
b) Diffusa reflexioner mellan objekt. (2 p)

Uppgift 4 (6 p).

Det finns flera väsentliga skillnader mellan 3D-grafik i spel, där man måste räkna ut omkring 60 bilder per sekund, och 3D-grafik i filmeffekter, där man oftast har flera minuter eller timmar på sig per bild. Ange minst tre fundamentala tekniska skillnader i hur renderingen görs, och förklara vilken inverkan de har på det visuella resultatet!

Uppgift 5 (6 p).

En 3D-modellerare gör en textur för ett golv med kvadratiska plattor som skall ligga på diagonalen i 45 graders vinkel mot väggarna. (Se bild nedan till vänster.) Golvet ska kunna visas i närbild. Den bildbaserade texturen blir tyvärr taggig i kanterna när man zoomar in. Det syns väldigt tydligt att den består av pixels. (Se bild nedan i mitten.) Bilden som modelleraren ritat och sparat som en PNG-fil med storleken 128x128 pixels ser ut som visas i bilden nedan till höger.


a) Föreslå en lösning för att skapa det här mönstret som fortfarande använder en bildbaserad textur men utan att ge samma extremt pixliga intryck på nära håll, och som ändå inte kräver att texturbilden görs större! Det finns minst två sätt att lösa problemet. (2 p)

b) En annan lösning är att använda en så kallad procedurell textur (*procedural texture*). Förklara vad som menas med detta, och ange de väsentliga för- och nackdelarna med metoden jämfört med bildbaserade texturer. (4 p)

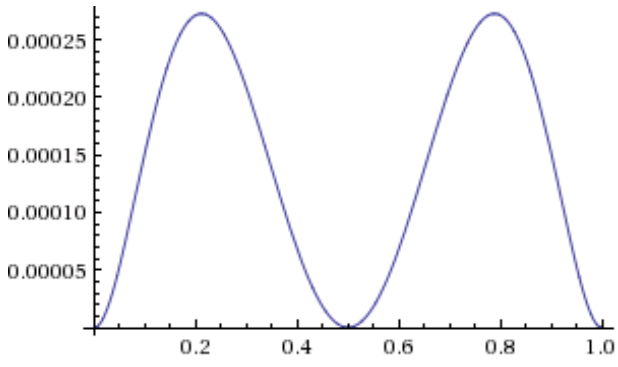
Uppgift 6 (6 p). En kubisk Béziérkurva definieras av ekvationen:

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

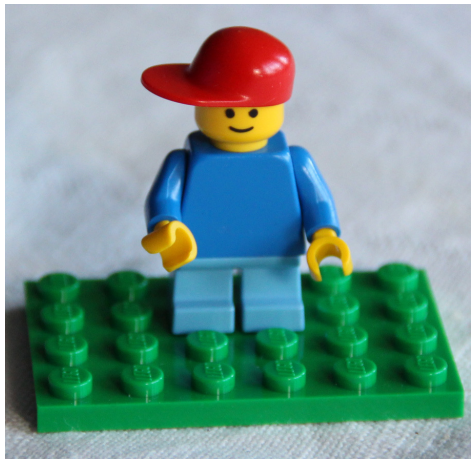
där p_i är kurvans kontrollpunkter, $i=0,1,2,3$.

a) En trevlig egenskap för Béziér-kurvor är att punkterna på kurvan håller sig inom det konvexa höljet (*convex hull*) av kontrollpunkterna. Med detta menas att alla punkter på kurvan kan beskrivas som en linjärkombination av kontrollpunkternas vektorer, där ingen av linjärkombinationens vikter är negativ, och där summan av alla vikterna är 1. Visa detta! (4 p)

b) Béziér-kurvsegment används ofta för att rita cirklar, men en sådan ”cirkel” blir inte exakt. Ett kurvsegment som approximerar en kvartscirkel i första kvadranten ($x\ge 0$, $y\ge 0$) med centrum i origo och radien 1 kan anges med kontrollpunkterna $p_0=(0,1)$, $p_1=(a,1)$, $p_2=(1,a)$, $p_3=(1,0)$, med $a=(4/3)(\sqrt{(2)}-1)\approx 0.552$. Avvikelsen mellan den exakta cirkelns radie och radien för approximationen ser då ut som i grafen här bredvid. Ange hur man räknar ut den grafen! Du behöver inte förenkla slututtrycket, men ange alla steg som behövs för beräkningen. (4 p)



Uppgift 7 (10 p). Programkoden längst ner på sidan ritar en Lego-figur med OpenGL och C++. Figuren kan rotera armarna vid axlarna, vrida händerna vid handlederna och vrida på huvudet, men benen är orörliga och sitter fast i kroppen. En verklig figur med samma rörelsemöjligheter visas i bilden här bredvid, och den finns också fysiskt i tentamenslokalen om du vill testa. (Vinklar och avstånd har förenklats i koden, och armarna antas för enkelhets skull vara raka och inte krökta.)



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 FigBody, FigHead,
 FigLeftArm, FigRightArm,
 FigLeftHand, FigRightHand;
float pose[5];
// (Code to create meshes left out)
M.init();
FigBody.render(M);
M.push();
M.translate(0.0, 0.0, 2.0);
M.rotateZ(pose[0]);
FigHead.render(M);
M.pop();
M.push();
M.translate(0.7, 0.0, 1.5);
M.rotateY(0.1); // 6 degrees or so
M.rotateX(pose[1]);
FigRightArm.render(M);
M.push();
M.translate(0.0, 0.0, -2.0);
M.rotateZ(pose[2]);
FigRightHand.render(M);
M.pop();
M.pop();
M.push();
M.translate(-0.1, 0.0, 0.2);
M.rotateY(-0.1); // -6 degrees or so
M.rotateX(pose[3]);
FigLeftArm.render(M);
M.push();
M.translate(0.0, 0.0, -2.0);
M.rotateZ(pose[4]);
FigLeftHand.render(M);
M.pop();
M.pop();
```

(uppgiften fortsätter på nästa sida)

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, och det ska framgå vilken funktion de olika noderna har. (5 p)

b) Man vill sätta en mössa på figuren. (Den verkliga figuren på bilden och i tentamenslokalen har en sådan mössa.) Mössan skall sitta fast på huvudet och rotera med huvudet, men den skall också kunna rotera för sig själv runt z-axeln. Visa hur detta skulle återspeglas i scengrafen, och visa också hur det skulle kunna göras i OpenGL-koden. (3 p)

c) Antag nu att man skulle vilja göra så att figuren i stället håller mössan i handen. Hur skulle scengrafen se ut då? Hur skulle OpenGL-koden behöva förändras? (2 p)