UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i	INF3320 — Metoder i grafisk databehandling og diskret geometri
Eksamensdag:	3. desember 2008
Tid for eksamen:	14.30 – 17.30
Oppgavesettet er på 6 sider.	
Vedlegg:	Ingen
Tillatte hjelpemidler:	Ingen

Kontroller at oppgavesettet er komplett før
du begynner å besvare spørsmålene.

Oppgave 1 Barysentriske koordinater

1. Forklar hva barysentriske koordinater i 2D er, hvordan de kan defineres og hvorfor de er viktige i data-grafikk.

2. Tegn opp en trekant og marker hvor punktene med barysentriske koordinater $(1, 0, 0)$, $(1/2, 1/2, 0)$ og $(1/3, 1/3, 1/3)$ ligger. Anta du har gitt RGB fargene $(1, 0, 0)$, $(0, 1, 0)$ og $(0, 1, 1)$ i de tre hjørnene. Hvilken RGB farge ville et fragment med barysentriske koordinater $(1/2, 1/4, 1/4)$ få med smooth shading?

3. Forklar prinsippet bak perspektiv-korrekt interpolasjon.

Løsning.

1. *Barysentriske koordinater mhp. en trekant $[\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2]$ er koordinatfunksjoner $\alpha_0, \alpha_1, \alpha_2$ slik at ethvert punkt i planet kan uttrykkes som*

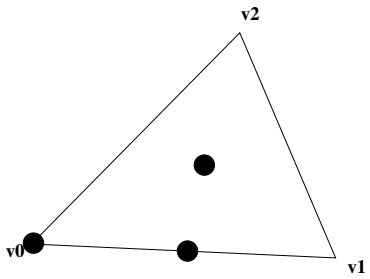
$$\mathbf{p} = \sum \alpha_i(\mathbf{p})\mathbf{v}_i,$$

der $\sum \alpha_i(\mathbf{p}) = 1$ og $\alpha_i(\mathbf{p}) \geq 0$ når $\mathbf{p}$ er i det indre av T. Disse egenskapene definerer barysentriske koordinater. Alternativt kan de defineres som forholdstall mellom areal, f.eks er

$$\alpha_0 = \frac{areal(\mathbf{p}, \mathbf{p}_1, \mathbf{p}_2)}{areal(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)}.$$

Barysentriske koordinater brukes til lineær interpolasjon over en trekant, f.eks farger, normaler eller andre attributter.

(Fortsettes på side 2.)



2. *Fargen blir* $(1/2, 1/2, 1/4)$.

3. *En teknikk som korrigerer for perspektiv-effekter når en interpolerer mhp. skjerm-koordinater. Resultatet blir som om man interpolerte mhp. rom-koordinater.*

Oppgave 2 Shading

Gjør rede for sammenhengen mellom avstand til en punkt-lyskilde og lys-intensitet. Hvordan utvides Phongs lys-modell slik at den inkluderer denne effekten og hvordan gjøres dette i OpenGL?

1. Skriv ned Phongs lys-modell (lighting model) og illustrer størrelsene som er involvert med en tegning.

2. Forklar hva flat, smooth (Gouraud) og Phong shading er.

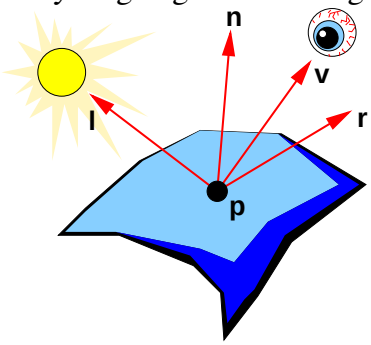
3. Gjør rede for sammenhengen mellom avstand til en punkt-lyskilde og lys-intensitet. Hvordan utvides Phongs lys-modell slik at den inkluderer denne effekten og hvordan gjøres dette i OpenGL?

Løsning.

1. *Phongs lys-modell:*

$$\mathbf{c} = \mathbf{c}_a + \mathbf{c}_d \max(\mathbf{n} \cdot \mathbf{l}, 0) + \mathbf{c}_s \max(\mathbf{n} \cdot \mathbf{h}, 0)^s$$

der fargene $\mathbf{c}$ *er resultatet av komponentvis multiplikasjon av lysfarge og material-farge.*



Her er $h = (\mathbf{l} + \mathbf{v})/||\mathbf{l} + \mathbf{v}||$ *og* s *er shininess parameteren.*

(Fortsettes på side 3.)

2. *Flat = konstant farge pr primitiv, typisk bestemt fra normalen i ett av hjørnene eller fra hele primitivet, smooth = fargen i et pixel finnes ved lineær interpolasjon av fargene i hjørnene, Phong = Phongs lysmodell regnes ut pr pixel, med vertex normaler interpolert over primitivet.*

3. *Lys-intensitet avtar som $1/r^2$ der r er avstanden til kilden. Phongs lys-modell med attenuation:*

$$\mathbf{c} = \mathbf{c}_a + \frac{1}{f(r)}\big(\mathbf{c}_d \max(\mathbf{n}\cdot\mathbf{l},0) + \mathbf{c}_s \max(\mathbf{n}\cdot\mathbf{h},0)^s\big),$$

der f er på formen $f(r) = ar^2 + br + c$ med parametere a, b, c som styres i OpenGL med glLightf. F.eks settes parameteren c med glLightf(GL_LIGHT0,GL_CONSTANT_ATTENUATION,2.0)

Oppgave 3 Dybdebufferet og skygger

1. Hva er dybdebufferet (også kalt z-buffer) og hvordan kan det brukes til å tegne ugjennomsiktige objekter? Forklar kort hvordan bruk av dybdebufferet styres i OpenGL?

2. Forklar prinsippet bak shadow map algoritmen og hvordan dybdebufferet brukes i denne.

Løsning.

1. *Dybdebufferet = en del av et framebuffer som kan lagre info om dybde (avstand fra kamera) for hver pixel. En kan utføre dybde-testing på fragmenter ved å sammenligne deres dybde med verdien i dybdebufferet.*

   Ved tegning av ugjennomsiktige objekter kan en sette opp OpenGL slik at fragmenter som kommer ned pipelinen blir forkastet med mindre de har mindre dybde enn verdien i dybdebufferet. Dersom verdien er mindre erstatter den verdien i dybdebufferet, og fargen som er satt i pixelet blir oppdatert med den nye verdien. På denne måten står man igjen med de fragmentene som ligger nærmest kamera til slutt, alt bakenfor blir forkastet.

   Dybdebufferet må aktiveres med glutInitDisplayMode(GLU_DEPTH) i intialiseringen av OpenGL. Dybde-testing aktiveres med glEnable(GL_DEPTH_TEST). Hvilken test som skal gjøres kan bestemmes med glDepthFunc.

2. *En gjør første et pass med minimum-dybdetest og kamera i lyskilden og genererer en "depth map" (et dybde-buffer) mhp lyskilden, dvs en lagrer for hvert pixel i en tenkt skjerm dybden fra lyskilden til nærmeste punkt på en ugjennomsiktige flate, akkurat som ved "hidden line removal".*

   Deretter rendrer man scenen fra en vilkårlig kameraposisjon og tester hvert fragment for om det har en dybde som er større en verdien i "depth mapen" mhp lyskilden- i såfall er fragmentet i skygge.

(Fortsettes på side 4.)

Oppgave 4 Transformasjoner

Forklar hvilke rolle modelview- og projeksjons-matrisene spiller i OpenGL. I resten av oppgaven skal vi sette opp følgende scene:

1. OpenGL kameraet (øye-punktet) skal plasseres i punktet $(1, 1, 1)$, se i retningen $(0, 0, -1)$ og ha opp-vektor $(1, 0, 0)$.
2. Du skal ved hjelp av funksjonene `glutSolidCube(1.0)` og `glScale` tegne en fysisk modell av et kamera som er formet som en avlang boks med fysiske dimensjoner 1, 2 og 3. Modellen plasseres med midtpunktet i punktet $(1, 1, -1)$, pekende i retning $(0, 0, -1)$ og med opp-vektor $(1, 1, 0)$.
3. View frustumet skal ha kvadratisk frontside, inneholde hele modellen og være minst mulig.

Lag en figur som illustrerer hva du ser på skjermen og skriv ned OpenGL kode med tranformasjoner som realiserer denne scenen, med og uten bruk av funksjonen `gluLookAt`. **Hint**: del opp i view- og model-transformer og sett dem i korrekt rekkefølge.

Løsning. *Her er transformasjoner, delt i blokker:*

Perspective transform:
```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(-x,x,-x,x,1.5,2.5);
```

View transform:
```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glRotatef(90.0,0.0,0.0,1.0);
glTranslatef(-1.0,-1.0,-1.0);
```
eller `gluLookAt(1.0,1.0,1.0, 1.0,1.0,-1.0, 1.0,0.0,0.0);`

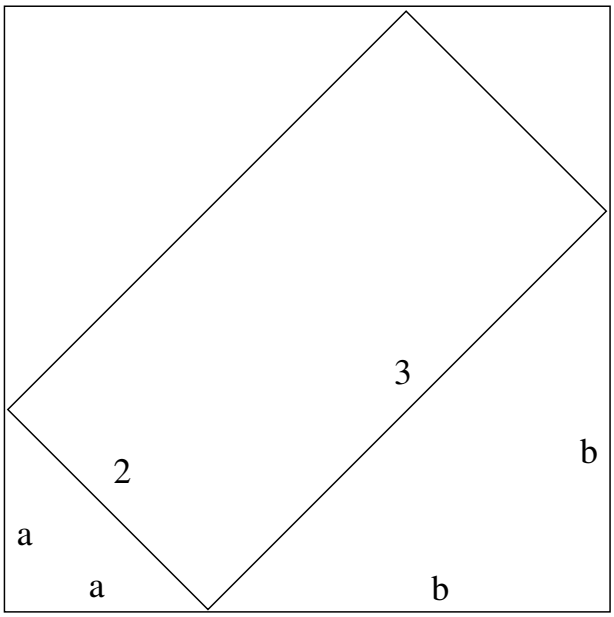
Model transform:
```
glTranslatef(1.0,1.0,-1.0);
glRotatef(-45.0,0.0,0.0,1.0);
glScalef(3.0,2.0,1.0);
glutSolidCube(1.0);
```

Kan også forenkle dette ved å slå sammen til ModelView transform:
```
glTranslatef(0.0,0.0,-2.0);
glRotatef(45.0,0.0,0.0,1.0);
glScalef(3.0,2.0,1.0);
```

(Fortsettes på side 5.)

```
glutSolidCube(1.0);
```

For å finne x (se tegning): Finner at $2x = a+b$, der $a^2+a^2 = 2^2$ og $b^2+b^2 = 3^2$ (Pythagoras). Dvs $a = \sqrt{2}$ og $b = 3/\sqrt{2}$. Da er $x = (a+b)/2 = 5/\sqrt{8}$



Oppgave 5 Subdivision

1. Du har gitt et polygon som inneholder følgende sekvens av kontroll-punkter

$$\mathbf{p}_0^0 = \begin{bmatrix}0\\0\end{bmatrix} \qquad \mathbf{p}_1^0 = \begin{bmatrix}1\\1\end{bmatrix} \qquad \mathbf{p}_2^0 = \begin{bmatrix}2\\-1\end{bmatrix} \qquad \mathbf{p}_3^0 = \begin{bmatrix}3\\1\end{bmatrix}$$

   som en delmengde. Bruk kubisk B-spline subdivisjon regler og regn ut alle kontroll-punkter på neste nivå som lar seg beregne fra de gitte kontroll-punktene.

2. Hva menes med grense-kurven ("limit curve") og hvor glatt er den i dette tilfellet? Kan grensekurven uttrykkes på en annen og mer direkte måte?

Løsning.

1. *Kontroll punkter:*

$$\begin{bmatrix}1/2\\1/2\end{bmatrix} \begin{bmatrix}1\\5/8\end{bmatrix} \begin{bmatrix}3/2\\0\end{bmatrix} \begin{bmatrix}2\\-1/2\end{bmatrix} \begin{bmatrix}5/2\\0\end{bmatrix}$$

2. *Grense-kurven er kurven som subdivisjonsprosessen konvergerer til. En kan tenke seg denne som resultatet av uendelg mange subdivisjons-steg. I dette tilfellet er grensekurven en uniform kubisk B-spline kurve som er C^2.*

(Fortsettes på side 6.)

Oppgave 6 Teksturer

1. Forklar relativt kort hva mipmapping er.
2. Skisser hvordan en kan bruke OpenGL/GLU til å finne gjennomsnittsfargen i en tekstur med dimensjon $2^m \times 2^m$.

Løsning.

1. *Mipmapping er en teknikk som går på å generere fra et bilde/tekstur en sekvens av bilder i stadig grovere oppløsninger, gjerne helt ned til et bilde med kun ett pixel. Ett pixel er gjerne gjennomsnitt av fire pixler på det finere nivået over. Mippmapping brukes for å reduserer artefakter som aliasing og gjør dessuten at rendringen går raskere siden en ofte kan bruke grovere teksturer til oppslag.*
2. *En kan bruke funksjonalitet i glu-biblioteket til å generere en mipmap fra bildet, f.eks* `gluBuild2DMipmaps`*. Gjennomsnittet finnes da på toppen av mipmap-sekvensen, dvs. fargen i bildet som er kun ett pixel stort.*