UNIVERSITETET I AGDER

Emnekode : DAT200
Kandidatnr. : 312
Dato : 04.12-13
Ark nr. : 1 av 12

Oppgave 1

a) Usant
b) Sant
c) Sant (hvis du bruker den for å fastslå hvor de andre aksene peker, og derifra se hvilken vei positiv rotasjon er)
d) Usant
e) Sant
f) Usant
g) Usant
h) Usant
i) Sant
j) Sant
k) Usant
l) Sant

Oppgave 2

a)

a)

(1) er en translasjonsmatrise der vi translaterer et element 4 enheter langs x-aksen, 2 enheter langs y-aksen og 2 enheter langs z-aksen.

(2) en rotasjonsmatrise der vi roterer 270° (-90°) om z-aksen.

(3) En skaleringsmatrise med skaleringsfaktor 6 i x-, y-, og z-retning. Denne har også homogenkoordinat H=2. Og siden $x = \frac{x_h}{H}$ betyr dette at $x = \frac{x_h}{2}$ i dette tilfellet

$y = \frac{y_h}{H}$ $\quad y = \frac{y_h}{2}$

$z = \frac{z_h}{H}$ $\quad z = \frac{z_h}{2}$

Oppgave 2 forts.

b) De tre basis transformasjonene vi har er Translasjon, Skalering, og Rotasjon.

c) Dette kan vi se ut ifra hvem koordinat som forblir uendret. Feks i en rotasjon om Z-aksen kan vi se at z-koordinaten forblir det samme.

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Dette gir ligningene:

$$x' = x \cdot \cos\theta - y \cdot \sin\theta$$

$$y' = x \cdot \sin\theta + y \cdot \cos\theta$$

$$z' = z$$

Oppgave 3

a) 1. Grønt
2. Rødt
3. Blått
4. Magenta
5. Cyan
6. Yellow
7. Black
8. White

b) Z-buffer algoritmen fungerer på den måten At den har to buffere som initialiseres når algoritmen starter opp. Et dybdebuffer og et ramme buffer. Så begynner den å sjekke pikslene til en flate. Har pikselen større dybde enn det som finnes i dybdebufferet, så oppdateres begge bufferene. Så går den til neste piksel. Slik gjør den med alle pikslene på flaten før den går til neste flate. En ulempe med z-buffer algoritmen er at den ikke kan behandle transparens (gjennomsiktig flater).

Oppgave 3 forts.

c) 1. Dette er gouraud. Her mister vi noe av spekuler refleksjonen og den blir ikke like blank/ fin.

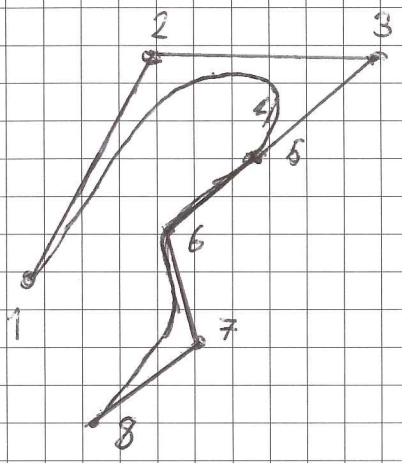
2. Dette er phong. Blankest og finest.

3. Dette er konstant intensitet. Blir ikke veldig realistisk å se på.

Oppgave 4

3-4-5-6 på rett linje gir G_1 kontinuitet.

a)



b) G_1-kontinuitet er lik retning på 1. derivant mens C_1 er lik 1. derivant.

Resultatet på G_1 og C_1-kontinuitet er da like bra

Oppgave 4 forts.

d) På figuren til venstre har pikslene blitt nødt til å ha blitt regnet ut på nytt siden den ser så trang ut i forhold. (vektor grafikk som .eps f.eks ctc.)

e) Isometrisk projeksjon er en ortografisk parallell projeksjon. Og blir kjennetegnet med at projeksjons-planet skjærer de tre hoveddaksene like langt vnna Origo.

Dette er kjekt f.eks når vi jobber med 3D-modeller så vil slippe å ta hensyn til de perspektiviske forkortningene. Også anvendbart innenfor noen enkelte dataspill.

Oppgave 4 forts.

a) For å projisere punktet (20, 40, 50) ned i x-y-planet når projeksjonspunktet er (0, 0, -50) setter vi opp en linje med parameter p.

$$x = (1-p)\cdot 0 + 20\cdot p = 20p$$
$$y = (1-p)\cdot 0 + 40p = 40p$$
$$z = (1-p)\cdot(-50) + 50p = -50 + 100p$$

Siden dette er xy-planet vet vi at $z = 0$.
Som vi så vil si at:

$$p = \frac{50}{100} = \frac{1}{2}$$

Dette gir

$$x = 20 \cdot \frac{1}{2} = 10$$

$$y = 40 \cdot \frac{1}{2} = 20$$

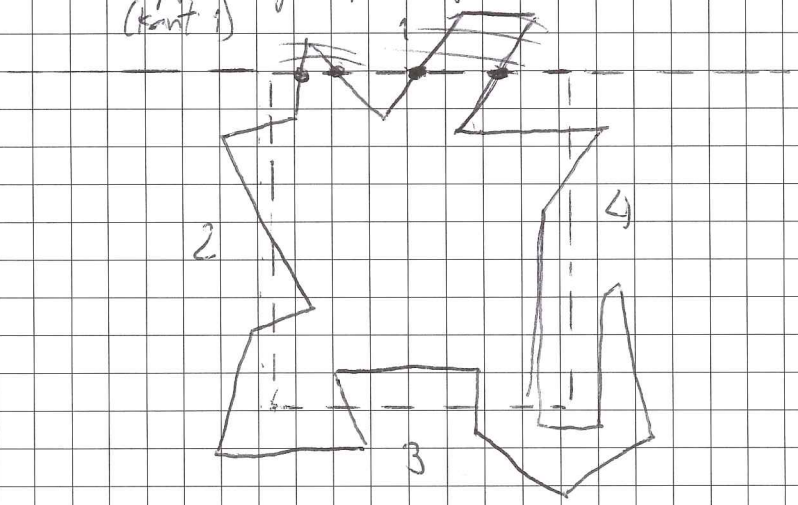
$$z = 0$$

Derfor er den perspektiviske projeksjonen lik $\underline{(10, 20, 0)}$

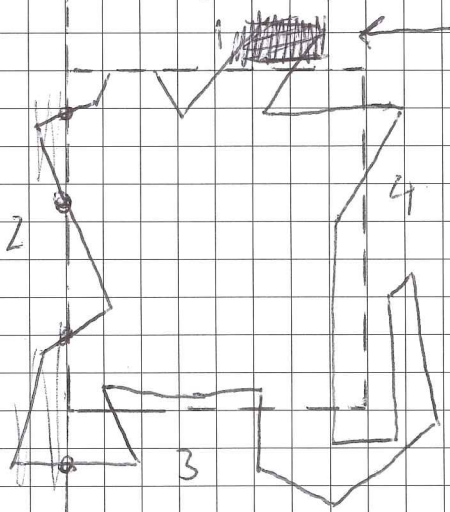
Oppgave 4 forts.

g). Steg 1.

Forleng kant 1 i det vendelinje og klipp mot toppen. (spor på skjæringen)
(kant 1)



Steg 2. forleng kant 2 i det vendelinje og klipp mot venstre (kant 2)

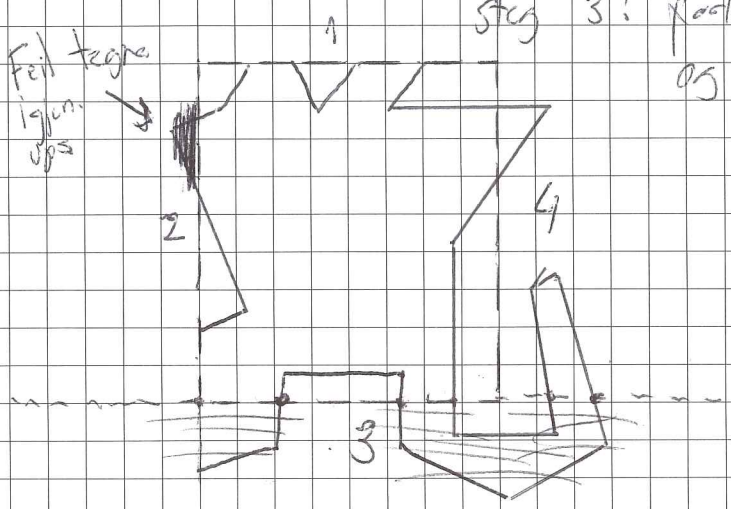


UNIVERSITETET I AGDER

Emnekode : DAT200
Kandidatnr. : 312
Dato : 04.12-13
Ark nr. : 9 av 12

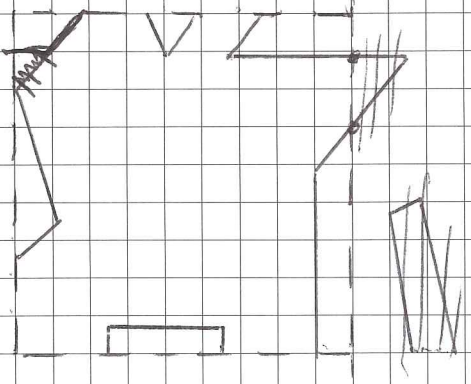
Oppgave 4 g) forts.

Feil tegne igjen. Ups

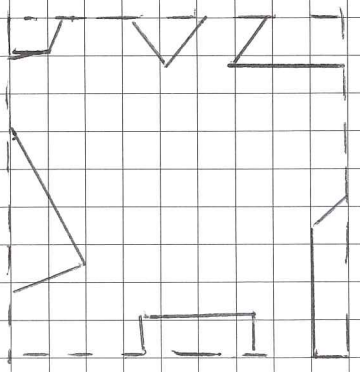


Steg 3: forlang kant 3: det vendelige og klipp mot kant (kant 3)

Steg 4 forlang kant 4: det vendelige og klipp mot høyre (kant 4)



Resultat



UNIVERSITETET I AGDER

Emnekode : DAT 200
Kandidatnr. : 312
Dato : 04.12-13
Ark nr. : 10 av 12

Oppgave 5

a) Ville først tegnet et snitt av selve lageret (ikke kulen) ~~så dreid det (omkring sentrum) litt utenfor for å gå jor å få det~~

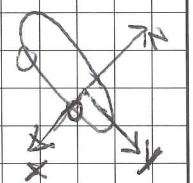
Så en sirkel. Så ville jeg lofted snittet rundt sirkelen. Deretter ville jeg laget en kule og plassert den inn i "lageret". Så ville jeg kopiert kulen, men brukt "loft-objektet" sitt koordinatsystem slik at det er bare å klikke shift + dra.

Oppgave 6

a)

1. Da sier vi at "denne" klassen skal lytte etter "muse-hendelser". Denne måten gjør at vi blir nødt til å skrive alle metodene som blir implementert selv om vi ikke skal bruke dem.
2. Denne måten sier vi at et MouseAdapter objekt skal lytte etter musehendelser. Når vi gjør det slik kan vi unngå å skrive alle metodene som vi ikke skal bruke.
3. Dette blir det samme som den første bortsett fra at metodene og klassen som skal lytte blir i musObjektet sin klasse.

Oppgave 6 b

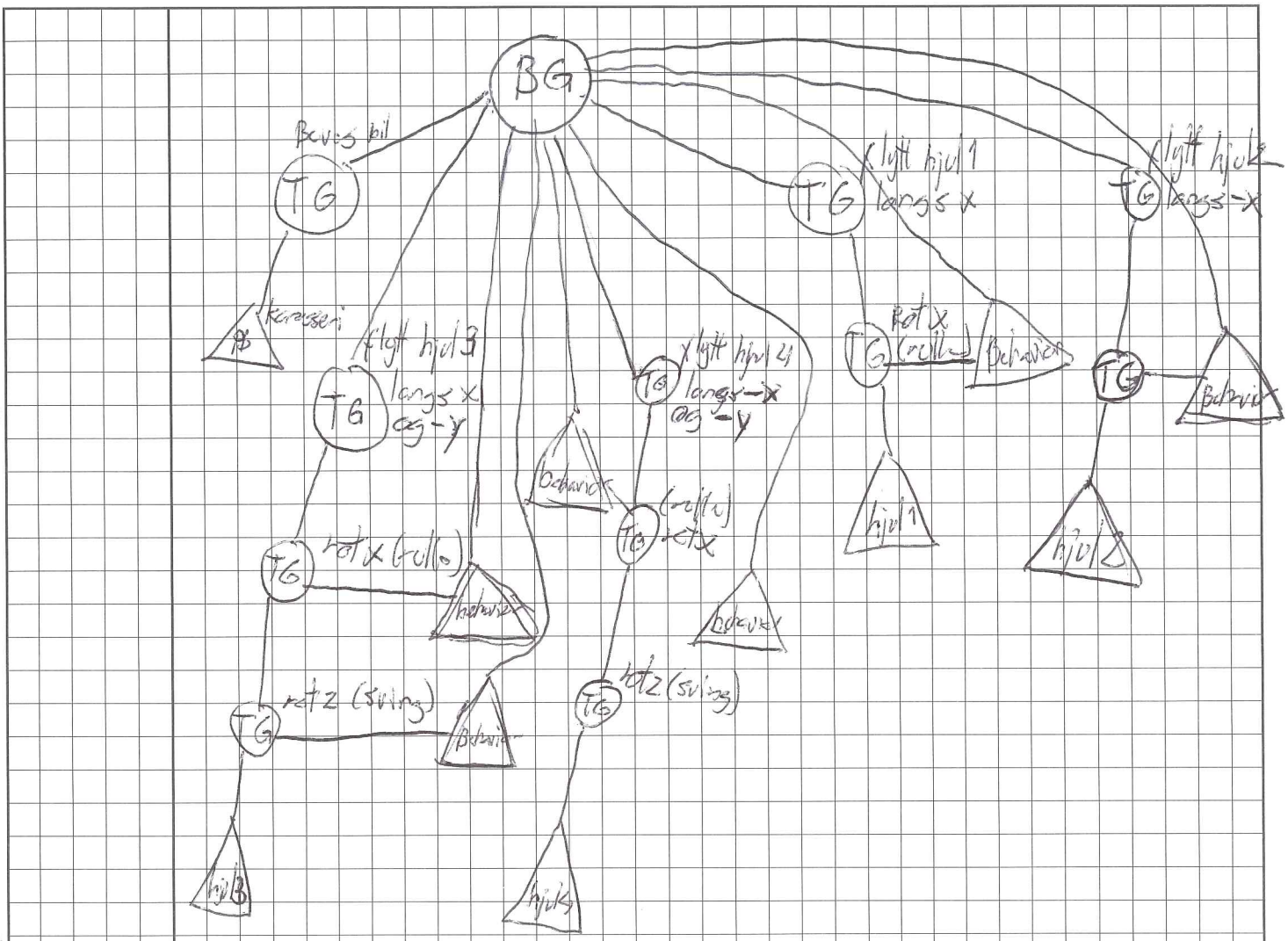


1)

```
{
BranchGroup bilRoot = new BranchGroup();
TransformGroup tH1 = new TransformGroup (10.0f, 0.0f, 0.0f);
TransformGroup tH2 = new TransformGroup(-10,0f, 0.0f,10.0f);
TransformGroup tH3 = new TransformGroup (10.0f,-100.0f,0f);
TransformGroup tH4 = new TransformGroup (-10.0f,-100.0f, 0.0f);
InspectorTDS loader = new inspectorTDS ("karosseri.3ds");
loader.parseIt();
TransformGroup karosseri = loader.getModel();
bilRoot.addChild(karosseri);
InspectorTDS loader2 = new InspectorTDS("hjul.3ds");
loader2.parseIt();
TransformGroup hjul1 = loader2.getModel();
TransformGroup hjul2 = loader2.getModel();
TransformGroup hjul3 = loader2.getModel();
TransformGroup hjul4 = loader2.getModel();
tH1.addChild(hjul4);
tH2.addChild(hjul2);
tH3.addChild(hjul3);
tH4.addChild(hjul4);
bilRoot.addChild(tH1);
bilRoot.addChild(tH2);
bilRoot.addChild(tH3);
bilRoot.addChild(tH4);

return bilRoot;
}
```

X - langs bakaksel på bilen
Y - tvers bilen (positiv mot bakenden)
Z - oppover



X = langs bak aksel på bilen.
Y - langs bilen (positiv mot bakgrunn)
Z - oppover