NTNU
Kunnskap for en bedre verden

Øving 14: Mer om transformasjoner

Tomas Holt 9.11.2016.
Instituttt for informatikk og e-læring ved NTNU.

Oppgave
I denne oppgaven skal du bygge opp en kube. Den skal bestå av sideflater med forskjellige farger. Anta at sentrum i kuben er i origo i et rettvinklet høyrehandskoordinatsystem. La lengden på sidekantene være lik 2.

Tegn tilslutt opp kuben i 4 forskjellige viewports, hvor du ser kuben fra forskjellige vinkler.

Under finner du tips på hvordan oppgaven kan løses, merk at beskrivelsene bruker norske navn.

I stedet for å benytte **glBegin(GL_POLYGON)** og skrive inn alle hjørnepunktene for hver av de seks sideflatene, skal du nå hente disse koordinatverdiene fra tabeller, når du bygger opp modellen av kuben.

Start med å tegne en skisse for hand av kuben. Angi koordinater og nummer på de ulike hjørnepunktene på skissen. Angi også flatenummer på de ulike sideflatene på kuben (0--5).

Lag en tabell **cornerPositions[][]** som inneholder alle hjørnepunktene i kuben og en tabell som inneholder seks ulike farger, (R, G, B)-verdier, colors**[][]**. Hjørnepunktnummer på kuben bør være det samme som indeksnummer for linker i tabellen **cornerPositions[indeksnummer][]**. På samme måte bør flatenummer på kuben ha samme nummer som indeksnummeret i tabellen **colors[indeksnummer][].** La tabellene **cornerPositions[][]** og **colors[][]** være klassekonstanter i tegneklassen.

Hint: Tabellene kan være på formen:
static final float cornerPositions [][] = {{ x0, y0, z0},{ x1, y1, z1}, ……};
static final float colors[][] = {{r0, g0, b0}, {r1, g1, b1},……..};

Lag følgende to metoder i tegneklassen: drawSide() og drawCube()

drawSide(): en generell metode som tegner opp et polygon og som har input-parametre lik hjørnepunktnummerne på en kubeflate. Fargen på kubeflaten skal være lik fargen i tabellen **colors[][],** som har indeks lik hjørnepunktnummeret til det første punktet som angis i argumentlista, når **drawSide()** kalles. Se eksemplet under:

```
public void drawSide( int a, int b, int c, int d){
        ……
        gl.glColor3fv(color[a],0);
        gl.glBegin(GL2.GL_POLYGON);
```

```
            gl.glVertex3fv(cornerPositions[a],0);
            ……..

        gl.glEnd();
}
```

drawCube(): Metoden **drawCube()** skal kalle metoden drawSide**()** flere ganger og tegne opp hele kuben (alle seks flatene).

Rotasjonsrekkefølgen på punktnumrene som gis inn når hvert enkelt polygon skal tegnes, bør være den samme for alle sideflatene. Dersom en benytter høyrehandsregelen og legger handbaken på sideflaten og fingrene i samme rotasjonsretning som punktene er gitt inn i, så skal tommelen peke ut av kuben (rotasjon mot urviseren). En grunn til dette er at en da i ettertid, systematisk skal kunne lage flatenormaler for alle sideflatene til kuben, som peker ut av kuben. På den måten vil en kunne skille mellom inn- og utside av kuben. Dersom flatenormalen peker inn i kuben vil denne flaten ikke bli synlig når vi tegner ut flaten.

For å tegne ut til ulike viewports bruk noe sånt

```
//første viewport
gl.glViewport(0,0 ,width/2, height/2);
 .......  //transformasjoner?
drawCube(gl);

//andre viewport
gl.glViewport(width/2, 0 ,width/2, height/2);
....//transformasjoner og/eller bruk gluLookAt() for å endre kameraposisjon
drawCube(gl);
```

Husk at transformasjonene påvirker hverandre. Det *kan* være lurt å vurdere å kalle glLoadIdentity() for å nullstille MODEL_VIEW-matrisa.