Øving 12: Intro til grafisk databehandling med OpenGL

Jan Nilsen, redigert av Tomas Holt 26.10.2016.
Instituttt for informatikk og e-læring ved NTNU.

Oppgave 1

Les Børre Stendsvold sine lenker til forelesning 1.

* Flater: http://borres.hiof.no/gb/exp-surface/p-surface.html

* Model til skjerm: http://borres.hiof.no/gb/exp-pipe/p-pipe.html

Les kap. 2 fram til og med «Reversing and Culling Polygon Faces» i «Red Book». Du kan hoppe over det som står under «Advanced».

Oppgave 2

1) Installer JOGL på maskina di. Følg installasjonsveiledningen og vedlegg om miljøvariabler.

Oppgave 3

Kjør JOGL2-eksemplene til denne øvingen/forelesningen (altså java-filene).

Gå igjennom koden og sørg for å skjønne hvordan ting henger sammen. Du skal være istand til å skjønne/forklare hva som bør ligge i main(), init(), reshape() og display(). Du kan gjerne også prøve å gjøre endringer i koden og se hva som skjer...

Merk at du kan finne forklaring på C versjonen av JOGL2Nehe02Basic2D.java på http://nehe.gamedev.net/tutorial/your_first_polygon/13002/.

Du finner også forklaring til de andre Nehe eksemplene på http://nehe.gamedev.net/ under «Legacy Tutorials». Nederst på hver lesson finner du kode for ulike språk/plattformer. Merk dog at JOGL2 som vi bruker ikke er dekket.

Oppgave 4 – frivillig men anbefales!

De aller fleste eksempler man finner av OpenGL er skrevet i språket C. Det kan derfor være lurt å vite hvordan man kan konvertere eksempler fra C til Java!

Sammenlikn eksempelkoden i JOGL2Nehe02Basic2D.java med lesson2.c som er vedlagt øvingen. Hva er ulikhetene?

Gjør om eksempel 2-5 i «Red Book» til JOGL2. Det kan da være lurt å ta utgangspunkt i eksempelkoden fra JOGL2Nehe02Basic2D.java og å endre denne.

***** TIPS 1**

I toppen av eksemplet finner du #define drawLine(x1,y1,x2,y2) osv.....

I Java så kan dette skrives som følger:

```
  private void drawOneLine(float x1, float y1, float x2, float y2, GL2 gl){
     gl.glBegin(GL2.GL_LINES);
        gl.glVertex2f(x1, y1);
        gl.glVertex2f(x2,y2);
     gl.glEnd();
  }
```

Merk spesielt at vi tar med GL2-objektet vårt inn i metodekallet. Alternativt kan man ha dette som en objektvariabel, men husk da å alltid oppdatere denne i starten av display-metoden.

***** TIPS 2**

gluOrtho2d() fungerer ikke (i min) JOGL2 implementasjon. Får bare NullpointerException.

Bruk heller 3d versjonen – slik gl.glOrtho (0.0, width, 0.0, height, 0.0, 1.0);

***** TIPS 3**

Tall kan ha forskjellig (standard) datatype i C og Java. Eksempel:

Om vi skriver 0.0 i Java så vil dette oppfattes som en double. Dette kan gi problemer ved f.eks. oversettelse av glClearColor(0.0, 0.0, 0.0, 0.0) i init-metoden. Her vil glClearColor()-metoden forvente at tallene er float, se Java-doc for GL-interfacet på http://jogamp.org/deployment/jogamp-next/javadoc/jogl/javadoc/ for mer om dette. For å forklare Java at vi ønsker float i stedet for double kan man legge på en f rett etter tallet. Her blir det da glClearColor(0.0f, 0.0f, 0.0f, 0.0f). Du kan derfor anta at der det står et desimaltall så må du legge på en f etter.

Om en metode som brukes krever short i stedet for int så må man bruke en cast. Det er f.eks. aktuelt på linjen glLineStipple(1, 0x0101) i diplay-metoden. Her vil 0x0101 oppfattes som en heksadeximal (0x) **int** mens metoden forventer **short**. Vi må dermed skrive om til ...glLineStipple(1, **(short)**0x0101).

Merk også at GLfloat ikke finnes i Java. Her brukes float i stedet. GLdouble blir double, og GLsizei blir int.

Oppgave 5

Kjør Multicolor_Cube_With_Keyboard_Listener. Denne løsningen benytter seg av tabeller for å tegne ut figuren (se i koden).

Løsningen har også en lytter for tastaturtrykk. Trykk med musa inne i vinduet og trykk så på en av knappene på tastaturet. Du vil da se at figuren begynner å rotere. Forhåpentligvis ser alt nå fint ut. Endre så koden slik at dybdetesting (GL_DEPT_TEST) ikke slås på. Kjør koden på nytt og roter kuben. Du vil da kunne se at opptegningen ikke fungerer som ønsket. Hvorfor skjer dette?

Frivillig. Denne blir første oppgave på neste øving, men kan godt gjøres nå.

Lag en JOGL2-løsning (start f.eks. med det første eksemplet) slik at det tegner ut figurer basert på følgende 8 punkter i vinduet:

P0 = (0.0,2.0,0.0), P1 = (1.5,1.5,0.0), P2 = (2.0,0.0,0.0), P3 = (1.5,-1.5,0.0),

P4 = (0.0,-2.0,0.0), P5 = (-1.5,-1.5,0.0), P6 = (-2.0,0.0,0.0), P7 = (-1.5,1.5,0.0)

Figurene skal tegnes ved hjelp av alle de ti grafiske OpenGL-primitivene:

GL_POINT, GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP

GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN

GL_QUADS, GL_QUAD_STRIP, GL_POLYGON

Alle figurene kan plasseres i samme vindu. Bruk glTranslate3f(x,y,z) eller glTranslate3d() til å forskyve figurene slik at du får plass til alle på skjermen samtidig.

Tips 1:

For å få plass til mye samtidig på skjermen så er det en fordel å translere i negativ z-retning (prøv f.eks. med -30). Husk også at etterfølgende transleringer vil være relativ til de før (dette gjelder så lenge man ikke kaller glLoadIdentity() på nytt – da dette vil gjøre at man starter «fra null igjen». Mer om dette når vi kommer til transformasjoner) så du vil nok kun translere i negativ z-retning ved første translering.

Tips 2:

Man glVertex3dv(**tabell**,**indeks**) kan være aktuell (i stedet for glVertex3f()/glVertex3d()). Denne er slik at man sender inn en **tabell** eller vektor som man sier i OpenGL. **Indeks** angir hvor langt ut i denne tabellen, man skal starte. De tre neste tallene vil så brukes som x,y,z verdier. Her kan man altså vurdere å ha en

* 10 endimensjonal tabell med 3 verdier for hvert punkt
* 1 endimensjonal tabell med med 30 rader
    * deklarering kan være slik: double[] tabell = {1,2,3.....,30};
    * glVertex3dv(tabell,**3**) vil her lage en vertex av tallene **3**,4 og 5 i tabellen.
* 1 todimensjonal tabell med 10 rader og 3 tall (x,y,z) på hver rad
    * deklarering kan være slik: double[][] tabell = {{1,2,3},{4,5,6},{..},{28,29,30}}

* glVertex3dv(tabell[1],0) vil gi de tre første tallene (4,5,6) på rad 1 i tabellen.

De to siste løsningene er de som gir mulighet for mest mulig kompakt kode – da vi kan gå i løkke å skrive ut alle punktene/vertexene. Den siste er den som blir enklest/mest logisk å forholde seg til.