NTNU
Kunnskap for en bedre verden

# *Øving: Transformasjoner*

*Jan Nilsen, redigert av Tomas Holt 28.10.2016.*
Instituttt for informatikk og e-læring ved NTNU.

---

**Oppgave 1**

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:**

Metoden 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

* 8 **en**dimensjonale tabeller med 3 verdier for hvert punkt (P0, P1,..., P7)
    * **OBS!** Denne løsningen er tungvindt - da tabellene må ha ulike navn!
* 1 **en**dimensjonal tabell med med 24 (8 x 3) rader
    * deklarering kan være slik: double[] tabell = {1,2,3,**4,5,6**,....,24};
    * glVertex3dv(tabell,**3**) vil her lage en vertex av tallene **4,5** og **6** i tabell.
* 1 **to**dimensjonal tabell med 8 rader og 3 tall (x,y,z) på hver rad
    * deklarering kan være slik: double[][] tabell = {{1,2,3},{4,5,6},{..},{22,23,24}}
    * 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.

---

**Oppgave 2**
Nå skal du lage kode (skill det gjerne ut i en egen metode) for å tegne opp en kube/terning som en trådmodell på skjermen på to ulike måter. Den første måten er beskrevet i oppgave 2a), den andre måten i oppgave 2b):

**2a)** Gi inn koordinater til hjørnepunktene til kuben/terningen. Bruk gl.glBegin(GL_LINE_LOOP) til å tegne linjer mellom hjørnepunktene. Anta at sentrum i kuben er plassert i origo i modellkoordinatsystemet (World Coordinate System) og at lengden på sidekantene i kuben/terningen = 2.0.

Bruk metoden glTranslated() til å forskyve terningen i negativ z-retning slik at den faller innenfor synspyramiden definert ved det parametervalget du har benyttet i glu.gluPerspective().

**2b**) Benytt den ferdiglagde metoden glut.glutWireCube() til å tegne en ny kube/terning. Hint. Før du kaller denne metoden må du lage et objekt av klassen GLUT. Du må importere klassen GLUT. Bruk: import com.jogamp.opengl.util.gl2.GLUT; glutWireCube() trenger ett argument! **Finn ut hva dette argumentet skal være og hva det faktisk betyr!**

---

**Oppgave 3**
Benytt OpenGL metodene for modelltransformasjoner skalering, translering og rotasjon til å forandre på posisjon, orientering på kuben/terningen. Lag to uttegninger hvor du roterer endrer rekkefølgen på rotasjon og translering.

Tegn ut minst to eksempler på sammensatt bruk av de tre transformasjonene.

Vis ved **uttegning** at rekkefølgen transformasjonene utføres i får betydning for uttegningene.

---

**Oppgave 4**
Bruk metoden glu.gluLookAt() til å betrakte terningen/kuben fra ulike øyeposisjoner.