PG430 – Innføring i grafikkprogrammering

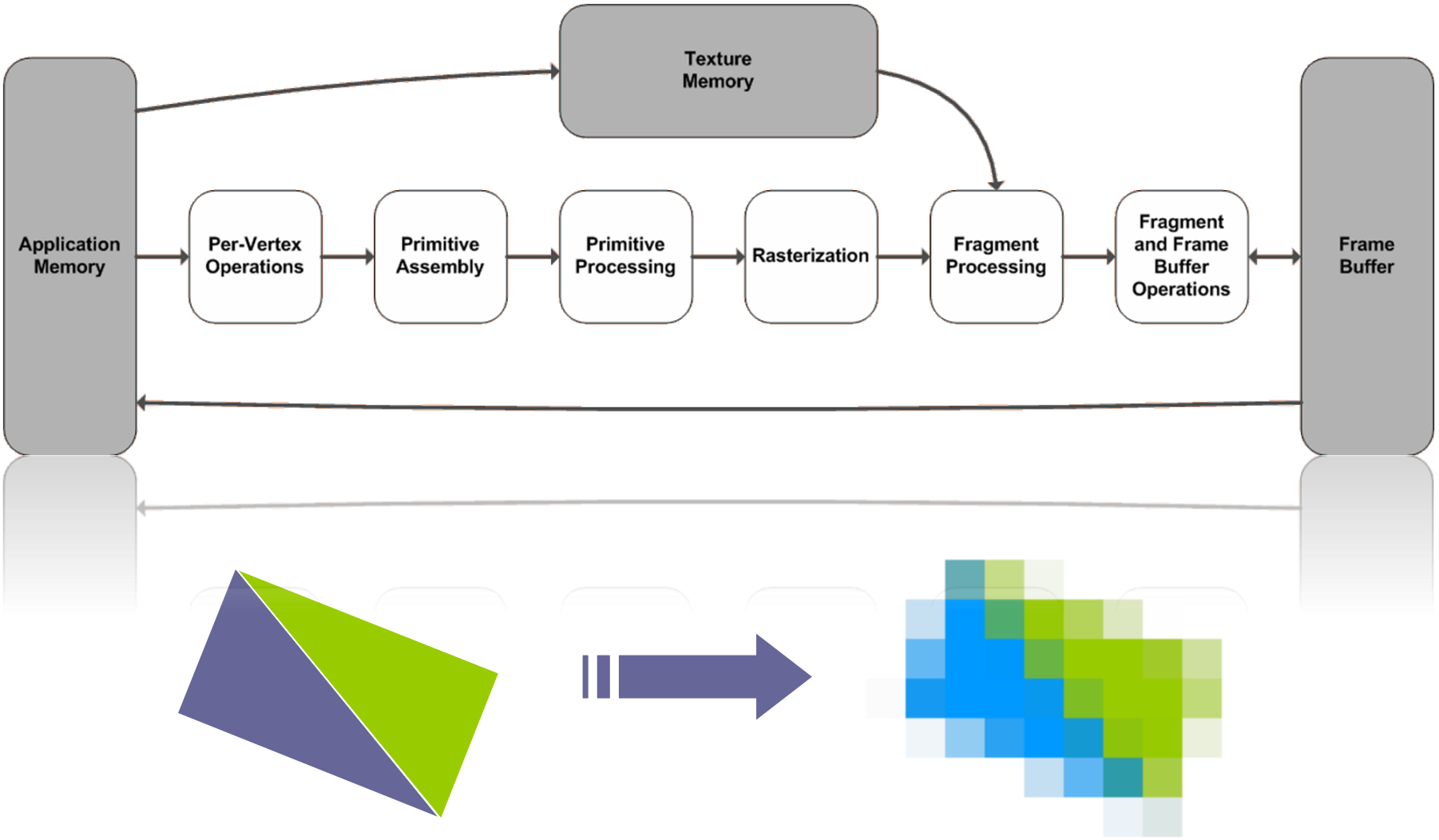
Forelesning 2

André R. Brodtkorb
Andre.Brodtkorb@nith.no

Oversikt dagens forelesning

- Repetisjon fra forrige time
- Normaler
- Rendring av geometri
- OpenGL og vindusystemer
- SDL
- Lab

OpenGL Pipeline



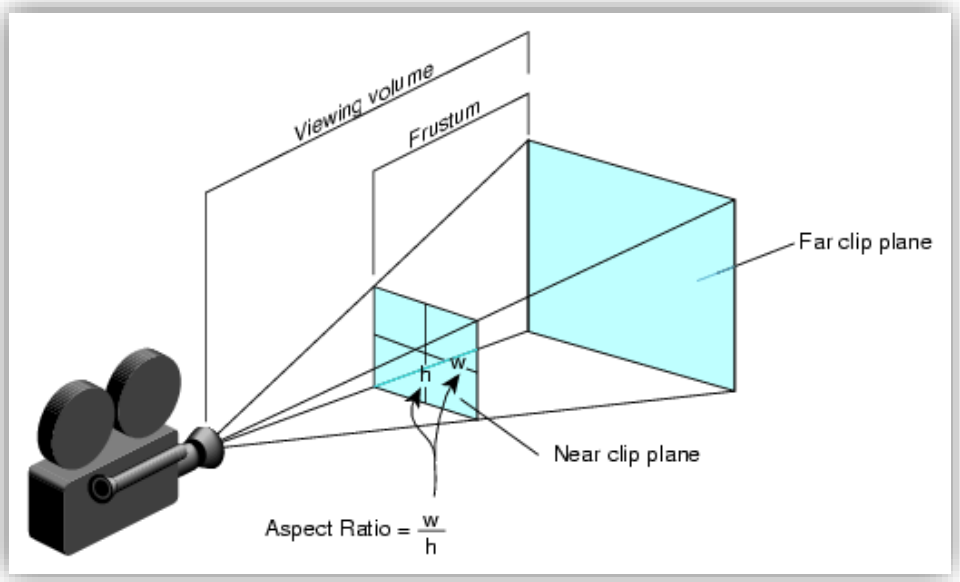
OpenGL - State maskin

- OpenGL er en state machine

- Current state avgjør hvordan ting rendres
  - glEnable(GL_CULL_FACE);
  - glEnable(GL_LIGHTING);
  - glEnable(GL_DEPTH_TEST);
  - glCullFace(GL_BACK);
  - glFrontFace(GL_CCW);
  - glBegin(GL_POINTS);

OpenGL matriser

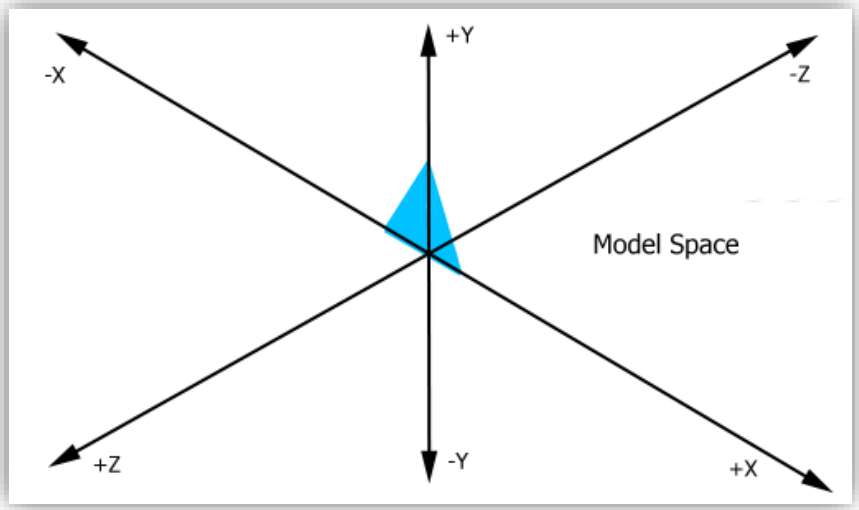
- *Modelview*-matrisen beskriver retning og hvor kameraet er: transformerer fra *world space* til *eye space*

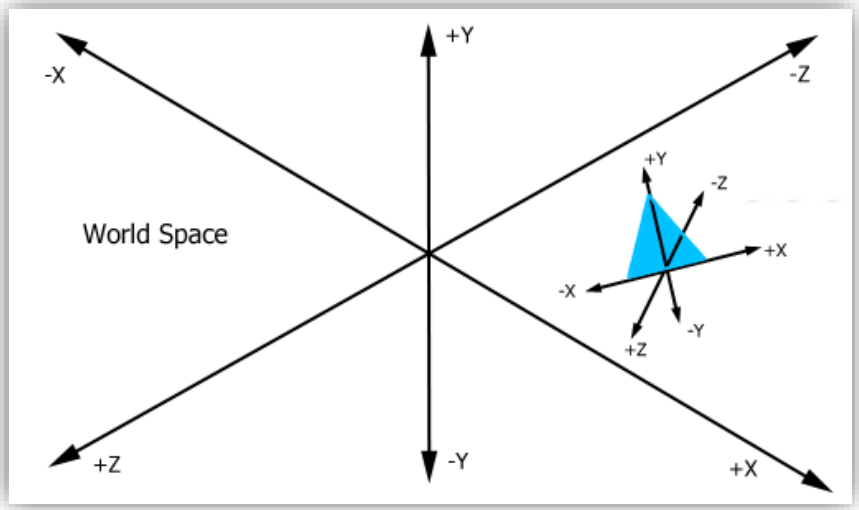


- *Projection*-matrisen beskriver hvordan kameraet ser (”linsen”) transformerer mellom *eye space* og *clip space*

Modelview matrisen

- Modelview matrisen plasserer objekter i scenen
  - glRotate
  - glTranslate

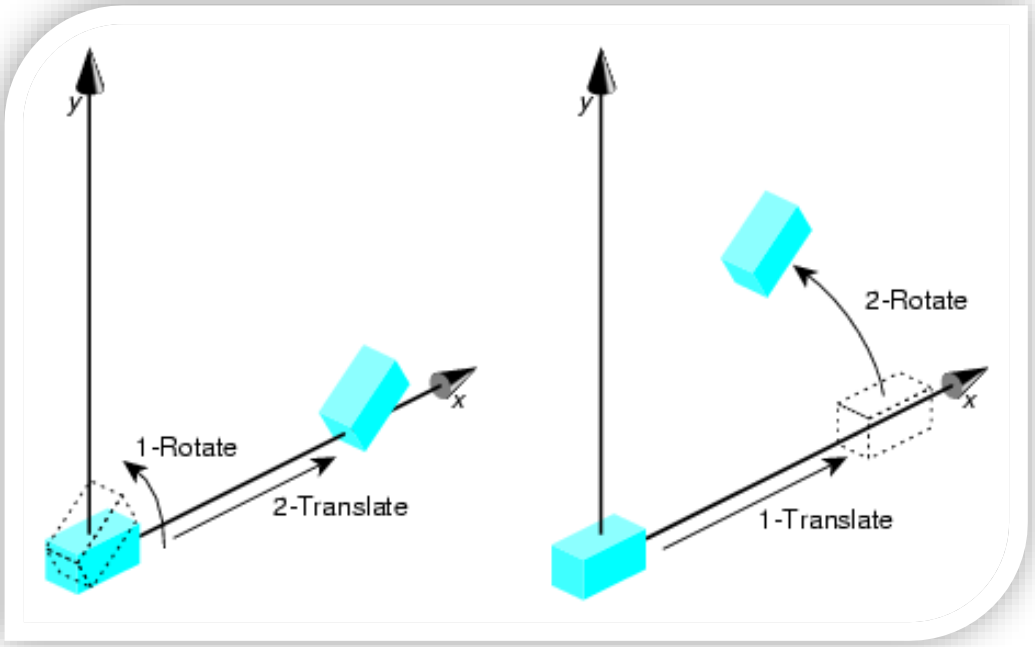




Rekkefølge på operasjoner

- Ved matrise multiplikasjon er rekkefølgen viktig
- Merk: Operasjonene gjøres i "motsatt" rekkefølge
  - Siste angitte transformasjon er den første som utføres

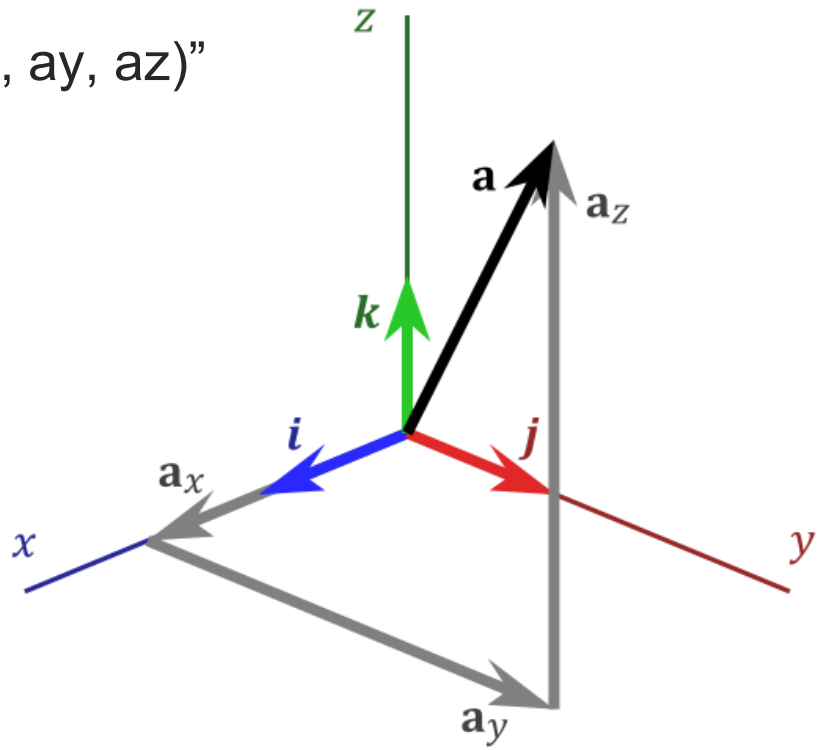
  - Venstre:
    - glTranslatef(...)
    - glRotatef(...)
  - Høyre:
    - glRotatef(...)
    - glTranslatef(...)



DAGENS FORELESNING

Vektorer

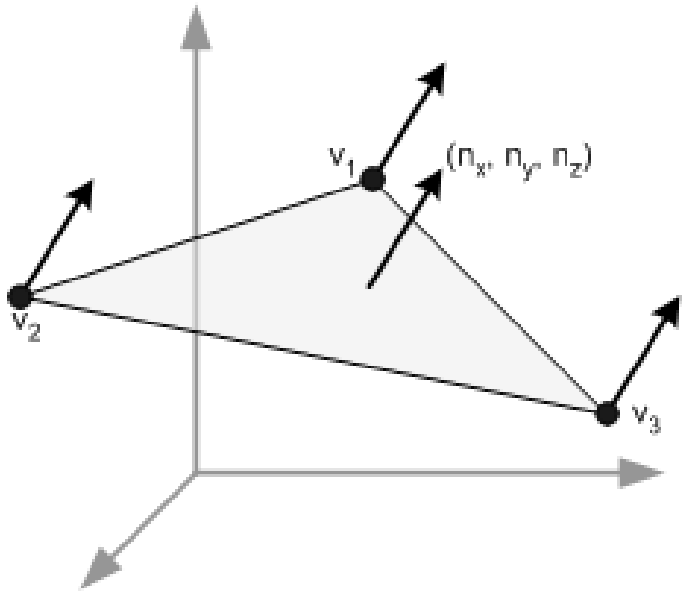
- En vektor
  - Beskriver en retning
  - ”Pilen fra punktet (0, 0, 0) til (ax, ay, az)”
  - Angis typisk som [ax, ay, az]



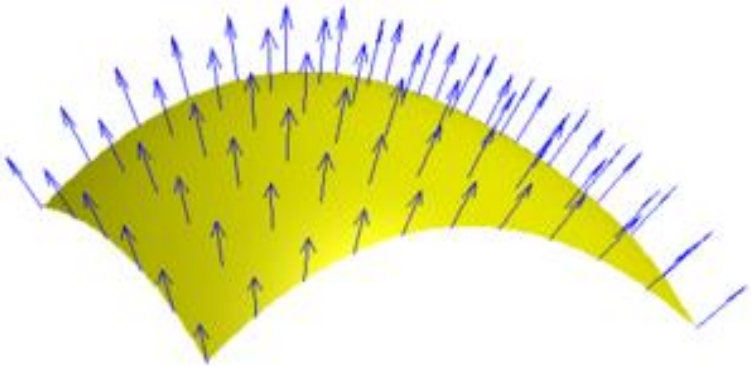
- Normaler har gjerne lengde 1 (normaliserte)

Normaler

- En normal er en vektor som angir retningen som er vinkelrett på en overflate.

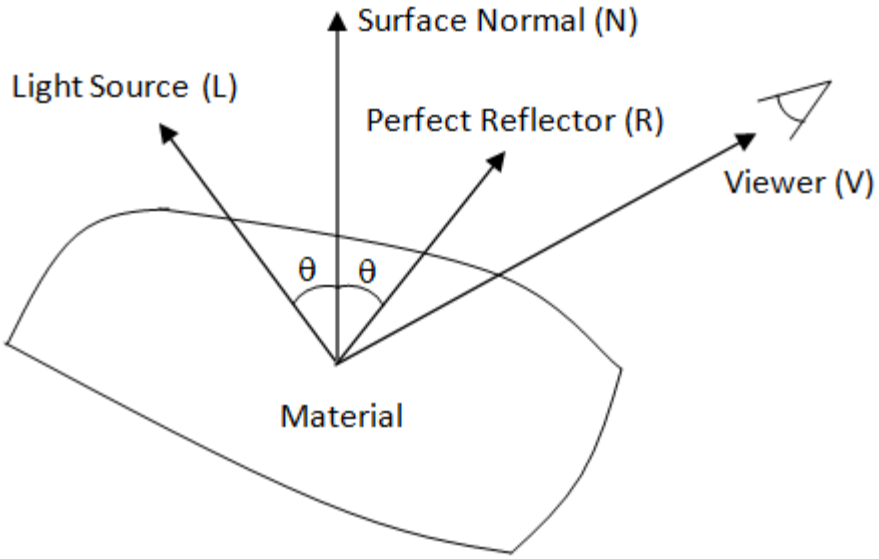


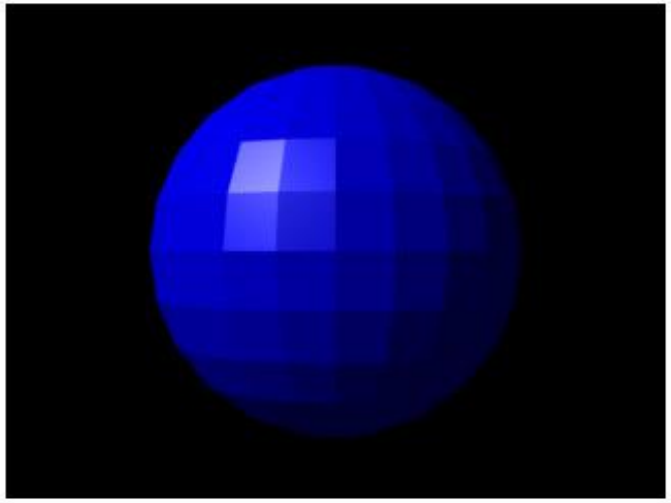
Normalvektorer



- Normalen brukes til lyssetting i OpenGL

- glNormal3f(n0, n1, n2);



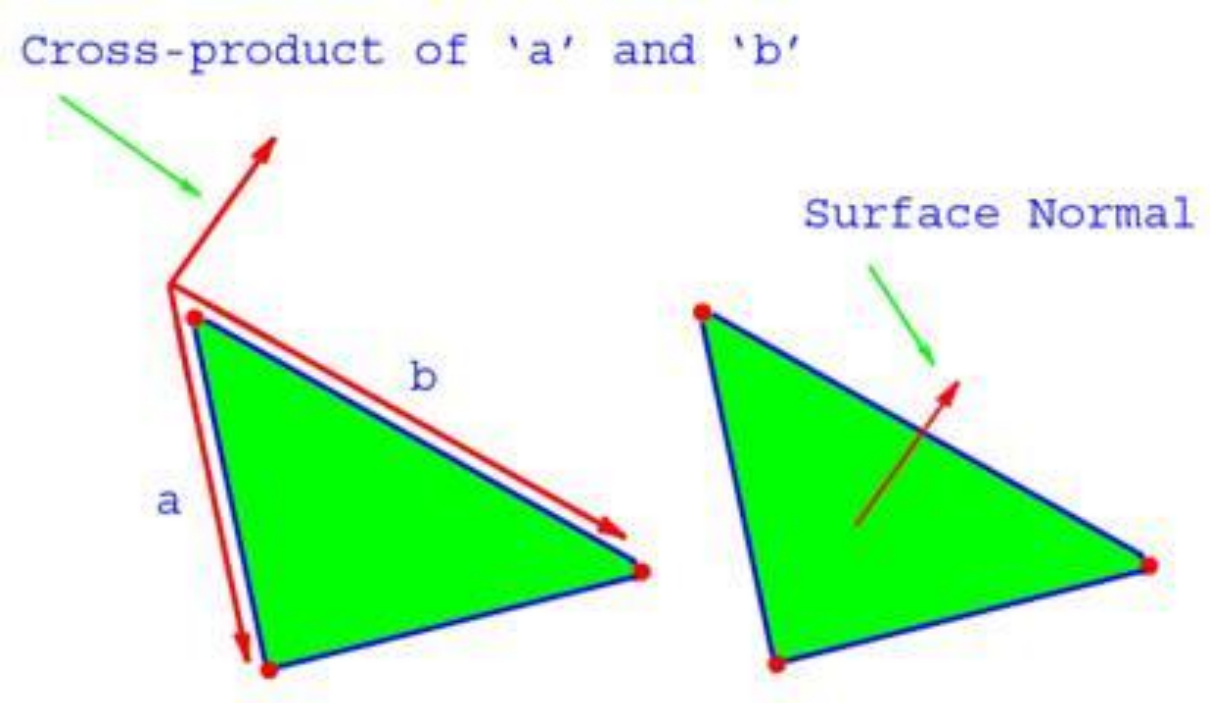


Normalvektorer - Demo

- Wireframe
  - Viser antall trekanter
  - Hidden line
- Flat shading
  - Hver trekant har én normal
- Goraud Shading
  - Hver vertex har én normal som er gjennomsnitt av omliggende trekanter
- Jo fler trekanter, jo glattere shading.

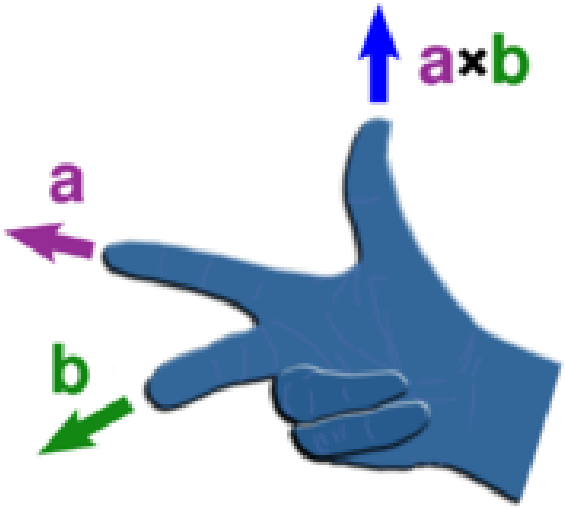
Normalvektorer

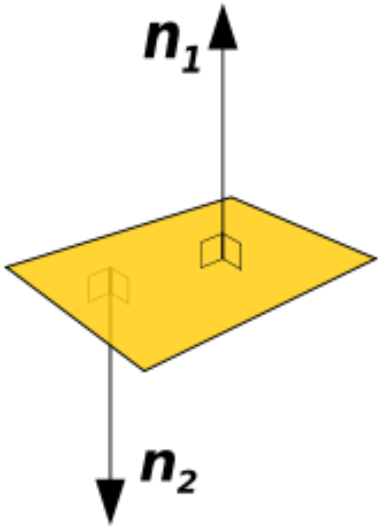
- Normalen til en flate kan regnes ut med å ta kryssproduktet mellom to vektorer i flaten.



Normalvektorer

- Rekkefølgen man krysser vektorene på bestemmer retningen.





Kryssprodukt

- To vektorer i R^3
  - $a = [a_0, a_1, a_2]$
  - $b = [b_0, b_1, b_2]$

- Kryssproduktet a x b:

$$a \times b = [(a_1 \cdot b_2 - a_2 \cdot b_1), -(a_0 \cdot b_2 - a_2 \cdot b_0), (a_0 \cdot b_1 - a_1 \cdot b_0)]$$

- Eks: a = (1, 0, 0) og b = (0, 1, 0)
  - a x b = ((0-0, 0-0, 1-0)) = (0, 0, 1)

Normalisering

- For korrekte lysberegninger krever OpenGL at lengden på normalvektorene er 1 (normalisert).

- For å normalisere en normal deler man alle komponentene med lengden av normalen.

$$|a| = \sqrt{x^2 + y^2 + z^2}$$

- OpenGL kan også normalisere automatisk
  - glEnable(GL_NORMALIZE);
  - Vil typisk redusere ytelsen

Input til OpenGL

- Immediate mode:
  - Enkelt men lite effektivt
- Display lists:
  - Caching av geometri og states; kan gi bedre effektivitet.
- Vertex arrays:
  - Kompakt representasjon.

Immediate mode

- En kommando kan kjøres umiddelbart når den kalles.
- Kommandoer for å tegne geometri, forandre stater, sette transformasjonen osv. blir sendt en om gangen til OpenGL for rendering.
- Dette er den enkleste måten å rendre geometri på i OpenGL; det er denne metoden vi har brukt så langt.
- Dette regnes som den minst effektive metoden man kan bruke for rendering.

Immediate mode

- Typisk kode:

```
glBegin(GL_TRIANGLES);
    glNormal3f(0.0f, 0.0f, 1.0f);
    glVertex3f(0.0f, 0.0f, 0.0f);
    glVertex3f(1.0f, 0.0f, 0.0f);
    glVertex3f(1.0f, 1.0f, 0.0f);
glEnd();
```

Display-lister

- OpenGL-kommandoer som har blitt lagret og kompilert
- Data kan bli lagret på en måte som er tilpasset den hardware som brukes.
- Kan ikke forandres når den er laget.
- Godt egnet for å rendre samme geometri flere ganger:
  - Eksmpel bil: Lag geometrien for et hjul i en dislay-liste. Kall denne display-lista fire ganger med forskjellig modelview matrise.

Display-lister

- I funksjonen init()

```
list_id = glGenLists(1);                    // list_id er en integer

glNewList(list_id, GL_COMPILE);
   glBegin(GL_TRIANGLES);
      glNormal3f(0.0f, 0.0f, 1.0f);
      glVertex3f(0.0f, 0.0f, 0.0f);
      glVertex3f(1.0f, 0.0f, 0.0f);
      glVertex3f(1.0f, 1.0f, 0.0f);
   glEnd();
glEndList();
```

- I funksjonen render()

```
glCallList(list_id)
```

Vertex Arrays

- Reduserer antall funksjonskall i forhold til display-lister og immediate mode.
- All vertex data lagres i arrays.
  - Vertex array, normal array, color array osv ...
- Vertexer som brukes av forskjellige primitiver må ikke spesifiseres mer enn en gang.
- Kan lett gjøres om til vertex buffer objects!

Vertex Arrays

```
//Hint: Bruk std::vector<GLfloat> i din kode!
GLfloat vertices[ ] = {0,0,1,  0,0,1,  0,0,1,  0,0,1,
                       1,0,0,  1,0,0,  1,0,0, 1,0,0,
                         0,1,0,  0,1,0,  0,1,0, 0,1,0,
                         -1,0,0,  -1,0,0, -1,0,0,  -1,0,0,
                         0,-1,0,  0,-1,0,  0,-1,0,  0,-1,0,
                         0,0,-1,  0,0,-1,  0,0,-1,  0,0,-1   };

glEnableClientState(GL_VERTEX_ARRAY);    // enable vertex arrays
glVertexPointer(3, GL_FLOAT, 0, vertices);     // sette vertex pointer
glDrawArrays (GL_QUADS, 0, 24);                // tegne 6 quads
glDisableClientState(GL_VERTEX_ARRAY);  // disable vertex arrays
```

Vindusystemer for OpenGL

- Vindussystemer
  - Håndterer brukergrensesnittet,
  - Lager OpenGL contexter, etc.
- På windows kan man bruke:
  - Glut http://www.opengl.org/resources/libraries/glut/
  - Qt http://www.trolltech.com/products/qt
  - SDL
  - Microsoft (Win32, Microsoft .NET Framework)
  - FreeGlut (Open source Glut)
  - wxWidgets http://www.wxwidgets.org/
  - Fast Light Toolkit (FLTK) http://www.fltk.org/
  - Etc...
- Vi kommer til å bruke SDL.

Win32

- Mye brukt for Window applikasjoner
- Ikke plattform uavhengig
- Må lage OpenGL kontekst selv
- Skriv egen ”eventhandler”
- Ferdig installert

GLUT / freeglut

- Standard rammeverk for OpenGL applikasjoner
- Windows, Linux, Mac ++
- Godt egnet for små prosjekter
- Bruker callbacks
- FreeGLUT erstatter GLUT (siste versjon 2001)

Simple DirectMedia Layer

- Støtter Windows, Linux Mac++
- Kan lage OpenGL kontekst
- Skriv egen ”eventhandler”
- Støtte for lyd
- http://www.libsdl.org/

Vindusystemer

- Håndterer input fra bruker
  - Resize, Keyboard, Mouse, etc.

- Typisk bruk med OpenGL:
  - init() – initialisere OpenGL states
  - reshape() – forandre størrelsen på vindu
  - display() – tegne scenen
  - (+ andre funksjoner for mus, tastatur, etc)

OpenGL, GLU og GLUT

- OpenGL funksjoner har prefix gl (glRotate)
  - definert i gl.h.
  - Implementert i grafikkdriveren.
- OpenGL Utility funksjoner har prefix glu (gluOrtho2D)
  - definert i glu.h
  - inneholder funksjoner som kan forenkle oppsett av bl.a. projeksjonsmatriser, etc.
- OpenGL Utility Toolkit funksjoner har prefix glut (glutCreateWindow)
  - Definert i glut.h
  - Definere vinduer og interaksjon med brukeren.

OpenGL applikasjon – init()

- Initialisering av OpenGL context
- Sette default states som sjelden forandres runtime.
  - Clear color, enable dybde test, etc.
- Innlasting av data som trengs før render.
  - Teksturer, modeller, etc
- +++ annen initialisering

Eksempel – init()

```
void init(void) {
    createContext();
    glClearColor (0.0, 0.0, 0.0, 1.0);
    glShadeModel (GL_SMOOTH);
    glEnable(GL_LIGHTING);
}
```

OpenGL applikasjon – reshape()

- Sette ny viewport
  - Viewporten beskriver vinduet man rendrer til i pixler
  - glViewport(0, 0, width, height);

- Sette ny projeksjonsmatrise
  - gluPerspective(.....) , gluOrtho(...)
  - Gjøres som regel for å beholde aspect ratio

Eksempel – reshape()

```
void reshape(int w, int h) {
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-50.0, 50.0, -50.0, 50.0, -1.0, 1.0);
    glMatrixMode(GL_MODELVIEW);
}
```

OpenGL applikasjon – display()

- Overskriv innholdet i framebuffer med bakgrunns-farge og nullstill dybdeverdiene.
  - glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
- Forandre eventuelle states.
- Tegne geometri.
  - glBegin(…) ….
- Swappe buffere ved dobbeltbuffering

Eksempel – Display()

```
void display(void) {
    glClear(GL_COLOR_BUFFER_BIT);
    glBegin(GL_TRIANGLES);
    glVertex3f(  0.0f, 25.0f, 0.0f);
    glVertex3f(  0.0f, 0.0f, 0.0f);
    glVertex3f(  25.0f, 0.0f, 0.0f);
    glEnd();
    SDL_GL_SwapBuffers();
}
```

MAPPE 1

Mappe 1 – Space Shooter

Oppgavetekst

1a) Trekant (romskip) som styres på skjermen med piltaster.

1b) Romskipet skyter kuler eller laser.

- Man får kun bestått eller ikke bestått på denne oppgaven. Dette betyr at man får 100% uttelling på oppgaven hvis man løser den tilfredsstillende.

- Oppgavetekst og startkode blir lagt ut på It's learning

Mappe 1 – Space Shooter

- Oppgaven skal leveres innen tidsfristen på It's learning
- Hjelp til mappe i labtimen før innlevering
- Oppgaven skal løses med C++ **inkludert byggesystem**
  - Microsoft Visual Studio 2010
- Innleveringen skal **IKKE** inneholde kompilerte filer, intellisence-filer, etc. Kun kildekode og byggefiler.
  - Sjekk selv: Zip opp filene, pakk ut til en annen mappe, og forsøk å kompilere opp og kjøre. Skal kompilere og kjøre "out-of-the-box"

Tips til løsning av oppgaven

- Ta utgangspunkt kildekoden på it's learning
- Lag egne klasser for romskip og skudd, tenk objektorientert!
- Ta i bruk Timer.h slik at det er enklere å bestemme hastigheten til skudd og romskip
- Bruk glTranslatef() for å flytte romskip og skudd.
- Pass på at romskipet ikke kan bevege seg utenfor frustum (det man ser på skjermen).

Vurdering av mapper

- 40% Utfør oppgaven, f.eks:
  - Har du utført alle oppgavene og skrevet koden selv?
- 30% Kodekvalitet, f.eks:
  - Er løsningen enkel, elegant, og riktig? OpenGL effektivitet, vel kommentert kode, kompileringsfeil og warnings.Se video om kodekvalitet på http://www.youtube.com/watch?v=mslMLp5bQD0 "**RailsConf 2010: Robert Martin"**
- 20% Visuelt tiltalende og naturlig feeling, f.eks:
  - Virker løsningen "naturlig" i interaksjonen, er grafikken pen, etc.
- 10% Annet, f.eks:
  - En snedig løsning som ligger utenfor oppgaven, men som fortjener ekstra poeng.

Hint til mappe: Rare feil

- Inkluder windows.h før gl.h

LAB

Dagens Lab

- Sett opp Visual Studio for bruk med SDL
- Rendering
- Rendermetoder

1 – Last ned og installer SDL

- Last ned SDL-devel-1.2.14-VC8.zip fra http://www.libsdl.org/download-1.2.php

- Unzip mappen til C:\libraries\i386\SDL-1.2.14 slik at SDL.h ligger i C:\libraries\i386\SDL-1.2.14\include\SDL.h
  - Viktig at det er nøyaktig path, vil bli brukt i mappene!

Sette opp prosjekt for Visual Studio

1. Velg fra menyen File → New → Project

2. Velg “Win32 Console Application”.

3. Angi navn til prosjektet ”simpleGL” og hvor det skal lagres.

4. Velg next.

5. Velg “Console Application” og “Empty Project”. Trykk på “Finish”.

En enkel OpenGL applikasjon?

- Enkelt rammeverk for en OpenGL applikasjon
  - Last ned zip-filen med kildekode fra “It’s learning”
- Det er to kataloger i mappen:
  - include – prosjektets include-filer (headere typisk)
  - src – prosjektets kildekode-filer
- Lag tilsvarende kataloger i prosjektet og legg til filene
- Sett include katalog til "include" og "C:\libraries\i386\SDL-1.2.14\include\"

OpenGL Bibliotek

- Før linking må man angi hvilke OpenGL-bibliotek man skal bruke i prosjektet:
  1. Gå til Project → Property Pages
  2. Velg Linker → Additional Dependencies
  3. Legg til lib-filene:

```
SDL.lib          // SDL
SDLmain.lib      // SDL
opengl32.lib     // OpenGL
glu32.lib        // OpenGL Utility Library
```

Timer-klassen

- En klasse som gir informasjon om tidsforbruk. Brukes til:
  - Beregne framerate
  - Benchmarking av kode
  - Sørge for konstant hastighet ved animasjoner og lignende

- Tre metoder
  - restart() // Starte timer på nytt
  - elapsed() // Returner tid siden timer ble startet
  - elapsedAndRestart() //gjett☺

Modifisering av kode

- Eksempelet tegner en trekant ved bruk av immediate mode.
- Generer en display liste som gir det samme resultatet.
- Lag et vertex array som gir det samme resultatet.

Lab fortsetter:

- Tegn en roterende kube
  - Det hjelper å tegne på papir først
- Gi hver flate forskjellig farge
  - Bruk flat shading
- Gi hver vertex forskjellig farge
  - Bruk smooth shading
- Legg til lys!
  - Legg til normal for hver flate i kuben
  - Eksperimenter med ambient, diffuse, specular og posisjonen
- Definer et objekt med mange vertices
  - test ut immediate mode, display list og vertex array for rendering

Lys

- Lys i OpenGL er en state
  - glEnable(GL_LIGHTING);
  - glEnable(GL_LIGHT<X>);
- Bruker normalen til å beregne lyssetting
  - Pass på at du har riktige normaler
- Farger på lyset settes også som state:
  - glLightfv(GL_LIGHT0, GL_AMBIENT, ambient);
  - glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse);
  - glLightfv(GL_LIGHT0, GL_SPECULAR, specular);
  - glLightfv(GL_LIGHT0, GL_POSITION, position);