PG430 – Innføring i grafikkprogrammering

Forelesning 1

André R. Brodtkorb
broand@nith.no

Oversikt

- Om kurset, om foreleser
- Informasjon om mappeevaluering / eksamen
- OpenGL demoer
- Introduksjon til OpenGL
- Lab

Om foreleser

- André Rigland Brodtkorb, broand@nith.no
- Utdannet ved institutt for informatikk ved Universitetet i Oslo
- Mastergrad fra 2007, Doktorgrad fra 2010
- Jobbet med grafikkprogrammering siden 2005
- 80% stilling som forsker ved SINTEF
- 20% stilling som foreleser ved NITH

Om kurset (og neste kurs…)

- PG430 «Innføring in Grafikkprogrammering»
  - I dette kurset lærer dere om grunnleggende spill-grafikk via enkel OpenGL
    - Rendering, Lyssetting, effektivitet, etc.
  - Dere skal også lære praktisk bruk av OpenGL
    - Fire mappeinnleveringer som til sammen danner et spill

- PG612 «Avansert grafikkprogrammering
  - I neste kurs lærer dere teknikker for avansert shading, og aller nyeste OpenGL.

Hvorfor OpenGL og ikke DX?

- Teknikkene og metodene dere lærer i dette kurset gjelder både for OpenGL og for DirectX
  - Moderne OpenGL og moderne DirectX er to «dialekter»
- DirectX er windows/xbox only, OpenGL finnes på alt fra iPhone/Android til PS3 og Linux
- Det er enklere å starte med OpenGL enn med DirectX
  - Mye oppstartskode ol. som kreves i DirectX slipper man med OpenGL
  - Fokus i kurset er på grafikkforståelse

Hva skal vi lære i dette kurset?

- Klassisk OpenGL – fixed functionality pipeline.
- Introduksjon til nyeste OpenGL – programmerbar pipeline
- Tegne primitiver (punkt, linjer, triangler osv)
- Transformasjoner (flytte objekter i rommet)
- Farger, materialegenskaper og lyssetting
- Teksturering
- Interaksjon med grafikk

Hva skal vi lære i neste kurs?

- Avansert grafikk – Programmerbar pipeline.
- Bruk av shadere.
- Skygger og avansert lyssetting.
- Avansert teksturering.
- Scenegraf-teknologi.
- Ray-tracing
- Etc.

Mappeinnleveringer - Info

- 4 innleveringer fordelt på semesteret (Se ukeplan)
- Siste forelesning blir muntlig eksamen
- Hjelp med mappe-problemer i siste lab før innlevering
- Krav:
  - Gruppeinnlevering ikke tillatt
  - Kopiering av kode fra andre / internett / etc. ikke tillatt
  - Visual Studio 2010 prosjekt
  - Readme-fil som beskriver løsningen

Mappe – Space shooter

- Implementer “Space Shooter”-type spill

- Konsept:
  - Ødelegg alle fiendtlige romskip
  - Unngår å kollidere med fiender og skudd
  - Romskipene kommer mot deg fra toppen av skjermen i forskjellige formasjoner og skyter med forskjellige type våpen.
  - Når alle romskipene er ødelagt avanserer man til neste nivå.
  - Siste romskip på hvert nivå er en ekstra sterk fiende (level boss) som er vanskelig å ødelegge.

Mappe – Space shooter

- Eksempler:
  - http://www.smiliegames.com/galaga/
- Demo: Besvarelser fra i fjor.

Mappe – Space shooter

1a) Trekant som styres på skjermen med piltaster.

1b) Trekanten skyter kuler eller laser.

2a) Fienderomskip beveger seg rundt på skjermen.

2b) Skuddene kan treffe romskipene som da slettes.

3a) Forskjellige typer romskip og bevegelsesmønstre.

3b) Romskipene slipper bomber og skyter mot spilleren.

3c) Spilleren skades og destrueres hvis han treffes av tilstrekkelig mange romskip eller skudd fra romskip.

Mappe – Space shooter

4a) Partikkeleffekter i forbindelse med våpenoppgradering og eksplosjoner legges til.

4b) Bakgrunnslandskap og andre objekter som for eksempel asteroider legges til.

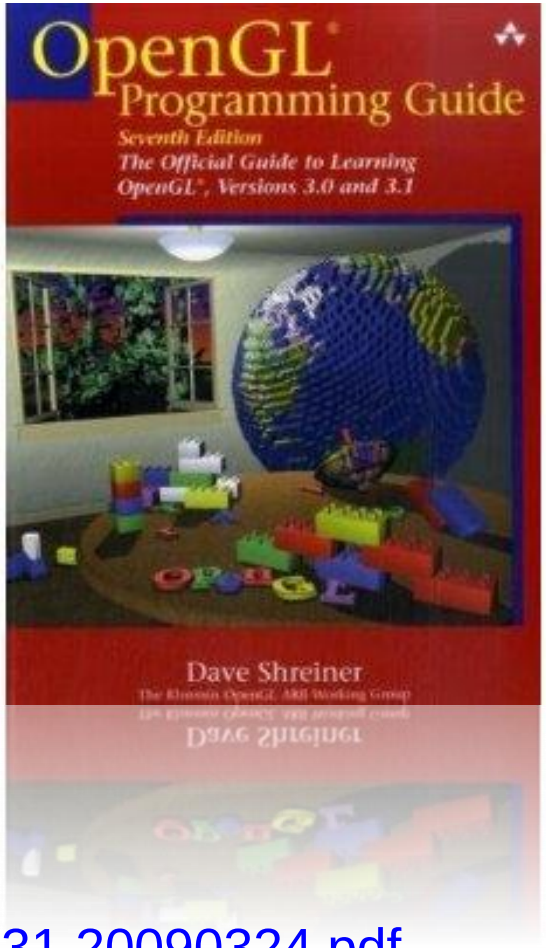
4c) Tekst på skjem (poengsystem, framerate counter, etc)

Karakter på mappe 2, 3 og 4, samt muntlig høring/eksamen.

OpenGL

Faglitteratur

- OpenGL Programming Guide,
  Seventh Edition,
  OpenGL Versions 3 and 3.1



- OpenGL spesifikasjon / reference
  - http://www.opengl.org/registry/doc/glspec31.20090324.pdf
  - http://www.khronos.org/files/opengl-quick-reference-card.pdf

Hva er OpenGL?

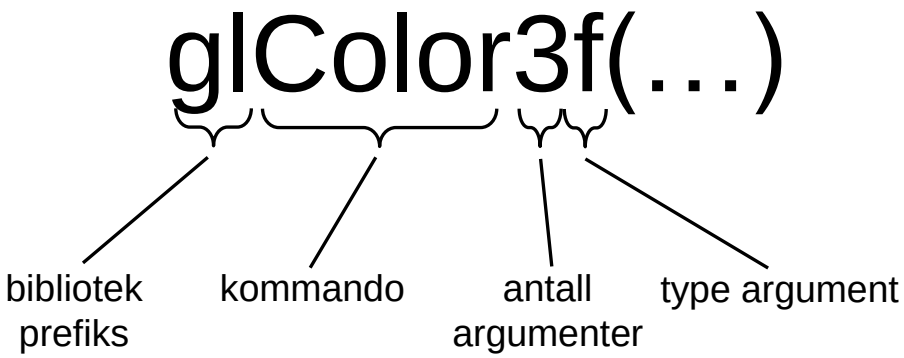
- OpenGL er et API (Application Programming Interface) for grafikkprogrammering.

- OpenGL er ikke et programmeringsspråk men et bibliotek skrevet i C.

Eksempel bruk i C/C++:

```
#include <GL/gl.h>              //Inkluder headerfil
…
glColor3f(1.0f, 0.0f, 0.0f);    //Kall en OpenGL-funksjon
```

OpenGL funksjonsnavn



Eksempel:

glColor3f(1.0f, 0.0f, 0.0f); //sett gjeldende farge til rød.

Grafikk-Demoer

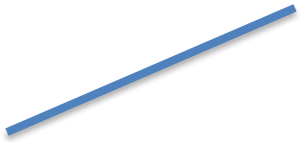
- NVIDIA OpenGL SDK 10
  - Dual Depth Peeling
  - Cascaded Shadow Maps
  - HDR
  - CG Isosurf
  - CG Geometry Program

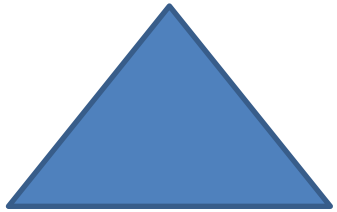
Grafikk-Demoer

- NVIDIA DirectX SDK 11
  - Diffuse Global Illumination
  - Hair
  - Island 11
  - Multiview Soft Shadows
  - OceanCS
  - SSAO11
  - Stochastic Transparency
  - Terrain Tessellation

Hvordan funker OpenGL

- Input til OpenGL er geometri i 3D
  - Punkter, linjer, trekanter, polygoner

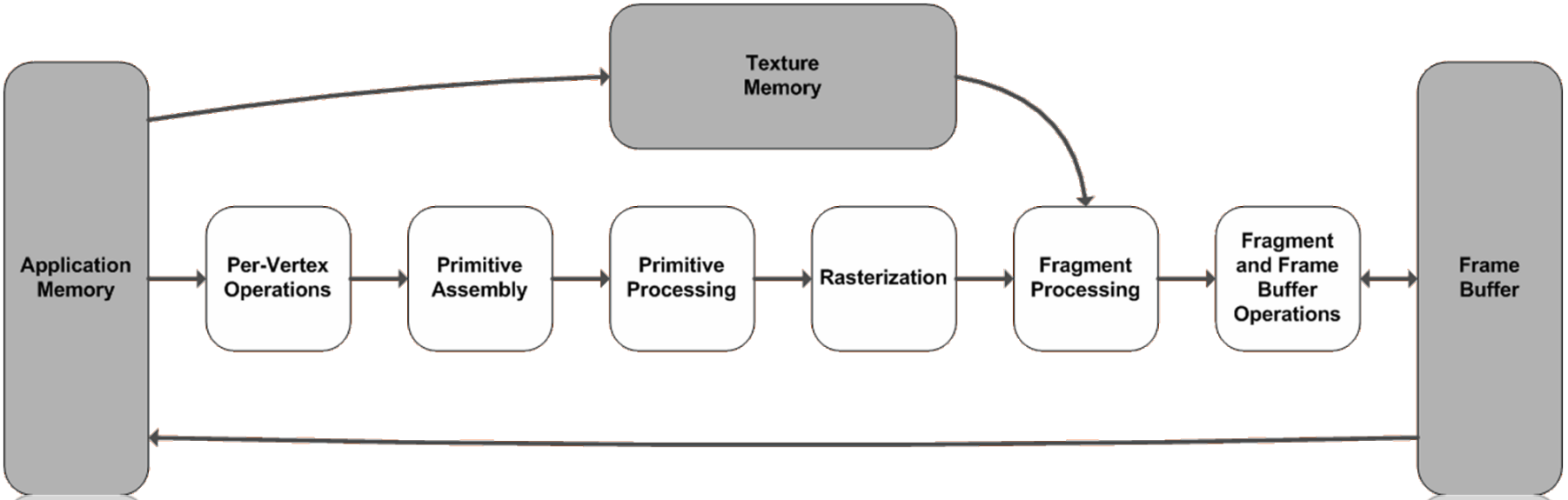




- Output fra OpenGL er bilder
  - Bildene består av pixler

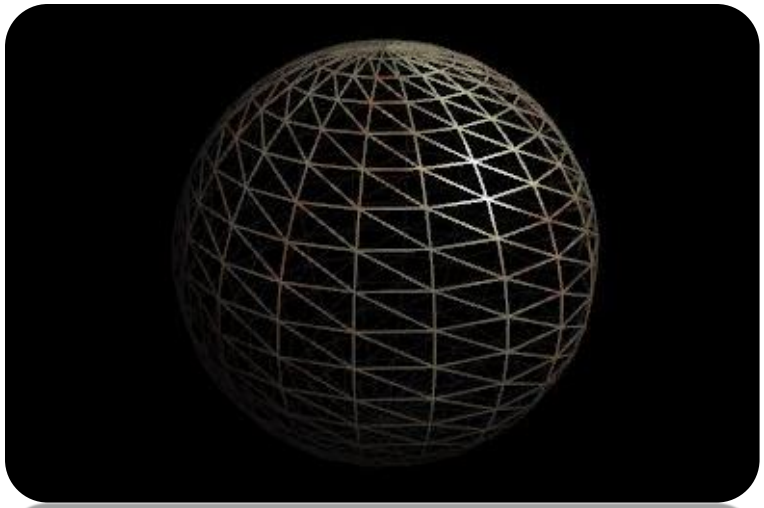
OpenGL Pipeline

- For å gjøre om trekanter til bilder bruker OpenGL en grafikk-prosesserings pipeline.



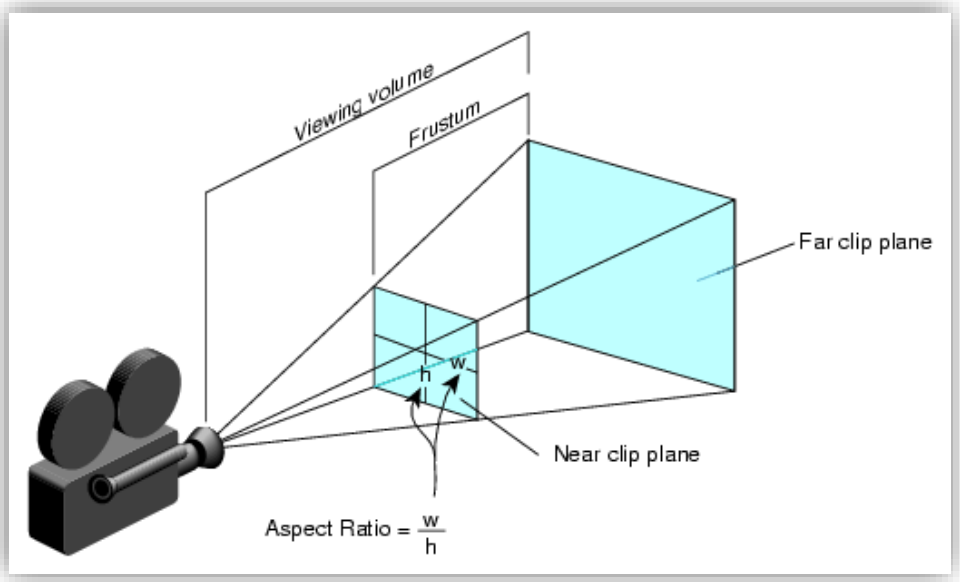
1. Per-Vertex operations

- Transformere vertexer, normaler og teksturkoordinater fra world space til eye space.
  - Vertexer er definert i *object space.*
  - Et felles koordinatsystem for alle objekter er *world space*.
  - Koordinatsystemet til kameraet kalles *eye space*.
- Utfører per-vertex lysberegninger.



1. Per-Vertex operations

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



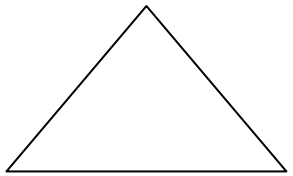
- *Projection*-matrisen beskriver hvordan kameraet ser (”linsen”) transformerer mellom *eye space* og *clip space*

2. Primitive Assembly

- Vertex data blir brukt for å generere primitiver.
  - Punkt, linjer, triangler, polygoner, etc.

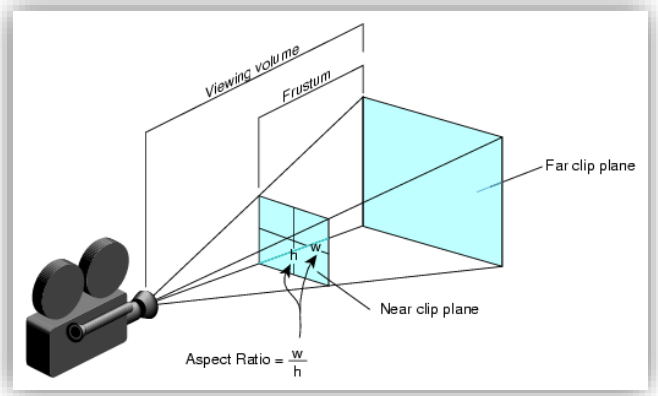
- Hvert primitiv blir laget fra ett sett med vertexer som blir spesifisert i applikasjonen.

```
glBegin(GL_TRIANGLES);
  glVertex3f(0.0, 0.0, 0.0);
  glVertex3f(1.0, 0.0, 0.0);
  glVertex3f(0.5, 0.5, 0.0);
glEnd();
```



3. Primitive Processing

- Clipping
  - Geometri som er utenfor *viewing volume* blir fjernet.

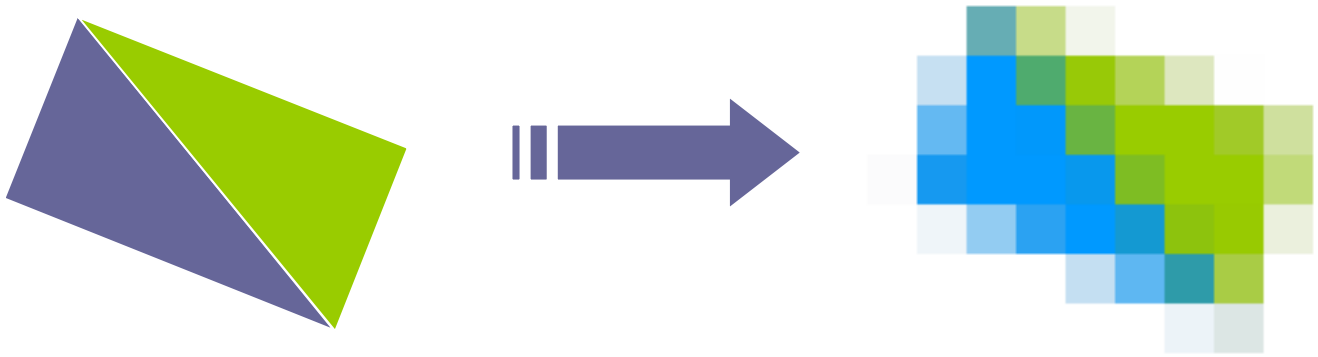


- Culling
  - Flater som vender vekk fra kamera kan fjernes.

- Objekter blir transformert fra *clip space* til *window space*.
  - glViewport(...);

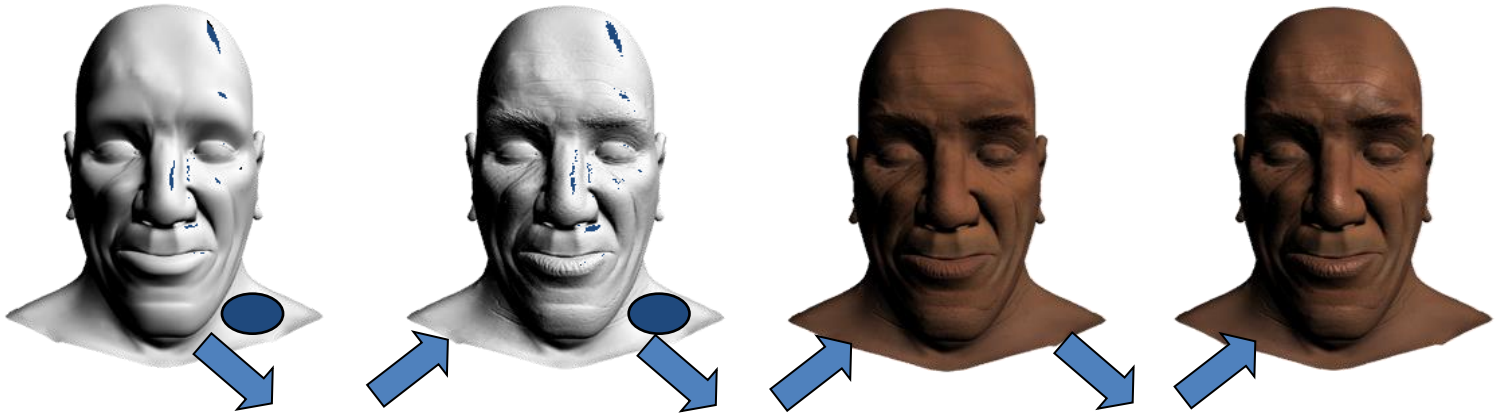
4. Rasterization

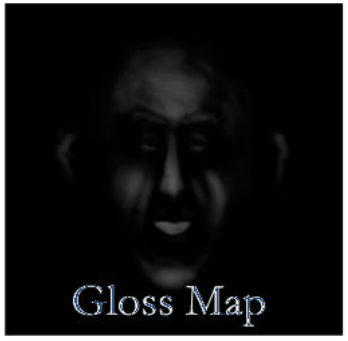
- Konvertering av primitiver til fragmenter



5. Fragment Processing

- Spesifisering av farge til fragmentet.
- Texture mapping, normal mapping, etc.





6. Fragment and Frame-Buffer Operations

- Alpha test
  - Forkast eller behold fragment basert på alfaverdi.
- Depth test
  - Bestemme om et fragment skal tegnes basert på dybde. Hvis et annet fragment er tegnet tidligere og ligger nærmere kamera skal fragmentet forkastes.
- Blending
  - Brukes for å blande fargeverdien på det fragmentet som blir prosessert med den fargeverdien som ligger i frame-buffer.

OpenGL State Machine

- Det er mye som påvirker hvordan en gitt geometri skal tegnes
  - lysegenskaper,
  - materialegenskaper,
  - teksturer, etc.

- Samlingen av disse variablene kalles *staten* (tilstanden)

- OpenGL er en *state machine*
  - Tar vare på alle *state variables* (tilstandsvariabler)

- En *state variable* forblir uforandret inntil vi selv endrer den
  - Dette er mye mer effektivt enn å sette alle *stater* hver gang man skal tegne

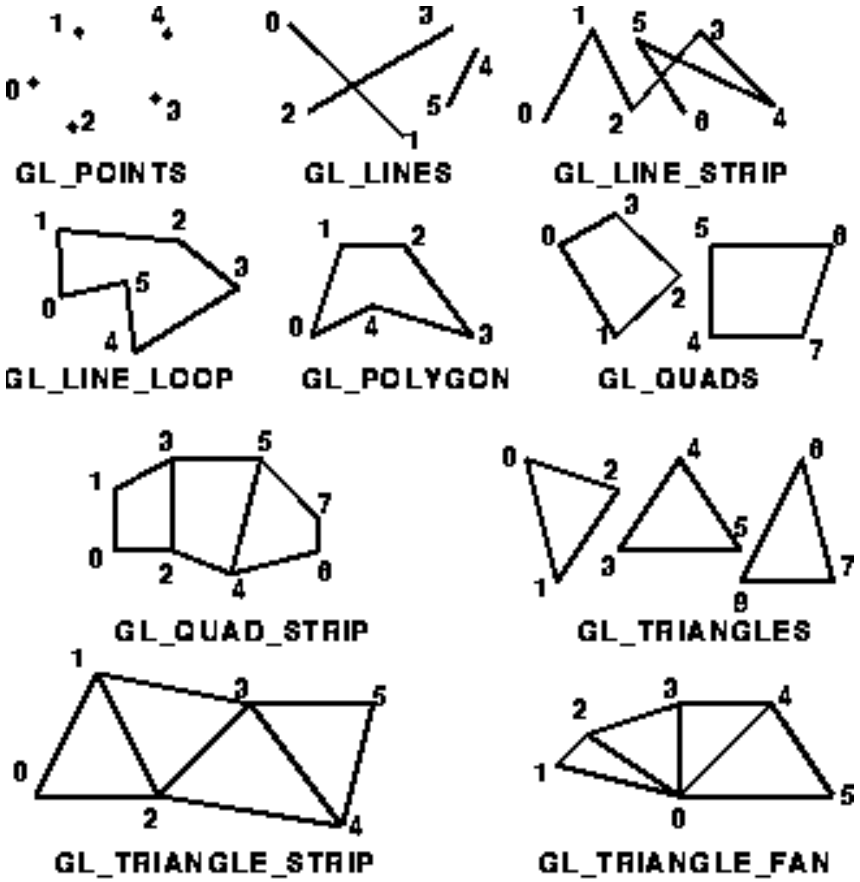
Grafikkdriveren

- Data blir sendt til OpenGL og grafikkortet via grafikkdriverne.
  - Selve “implementasjonen” av OpenGL ligger i grafikkdriverne.

- Data blir så sendt gjennom OpenGL pipelinen
  - Current state avgjør hvordan resultatet blir vist på skjermen.

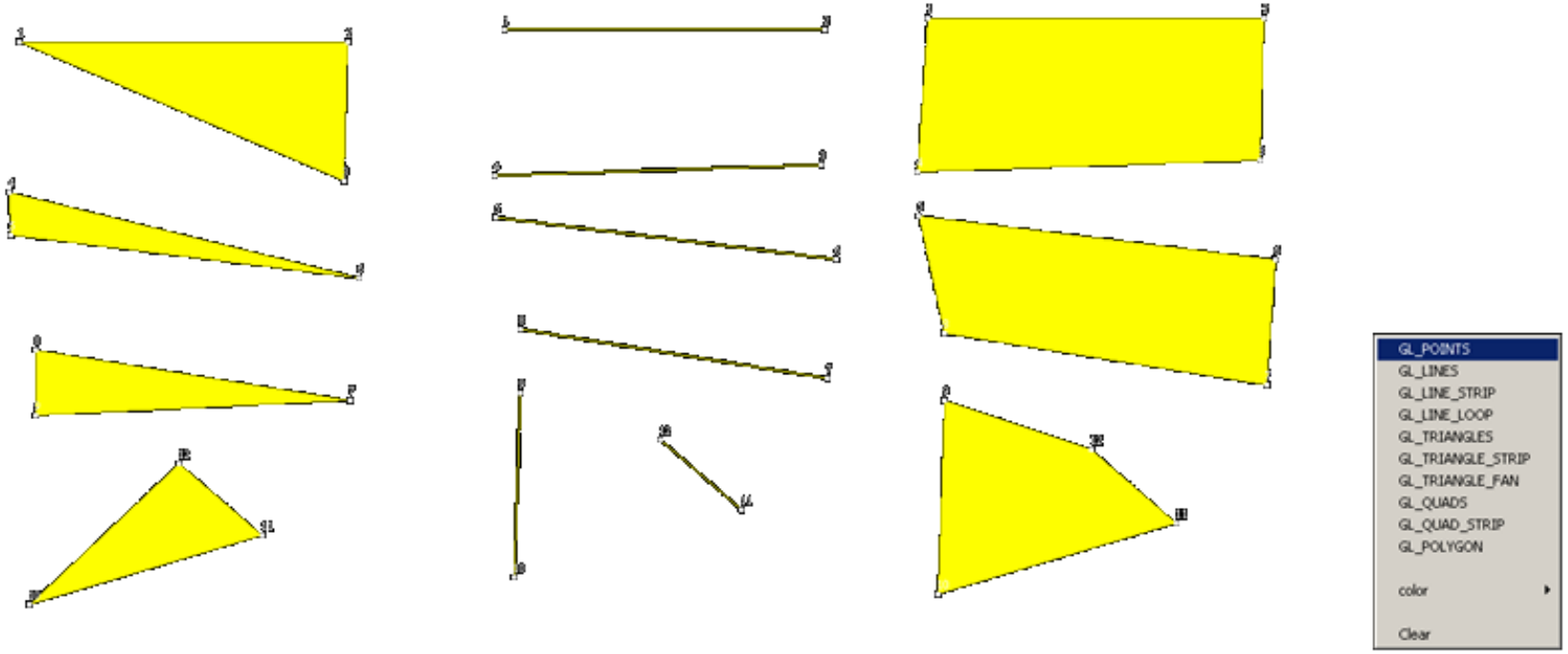
OpenGL Data

- Vertices (Punkter)
  - XYZ koordinater
  - Normal,
  - Farge,
  - ...
- Primitiver
  - Punkter
  - Linjer
  - Polygoner



Primitive Assembly

- Samme vertexer kan gi ulike primitiver.



Primitive Assembly – Demo

- Last ned **Making Primitives**
- **Demo:** Plasser vertexer og skift mellom primitiver som skal tegnes.
- **Demo:** Alle triangler har en forside og bakside. Måten disse tegnes på er avhengig av hvordan vertexene plasseres. Eksperimenter med denne versjonen som ikke viser baksiden av trianglene, og observer hvordan primitivene tegnes avhengig av plasseringen til vertexene.

States

Drawing states

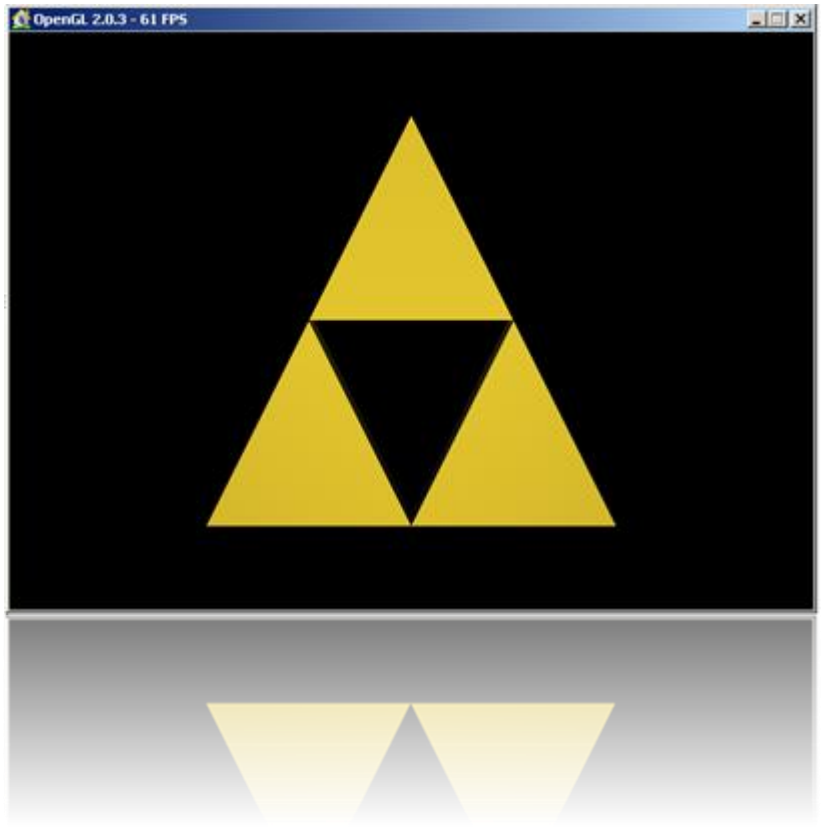
- glEnable(GL_LINE_SMOOTH)
- glEnable(GL_LINE_STIPPLE)

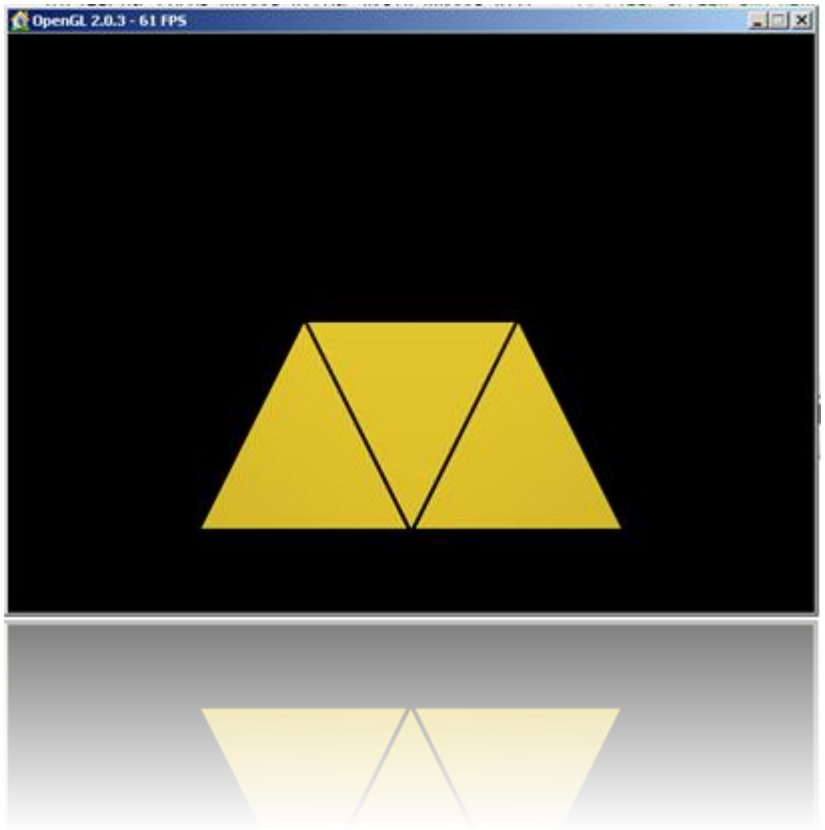
Pause – 15 min

Lab

- I dagens lab skal vi lære litt bruk av OpenGL
  - Definering av geometri
  - Rotasjoner
  - Translasjoner
  - Farger

Lab: Modifisering av kode





Lab-oppgave forts.

- Demo:
  - Program før modifikasjon.
  - Program etter modifikasjon.

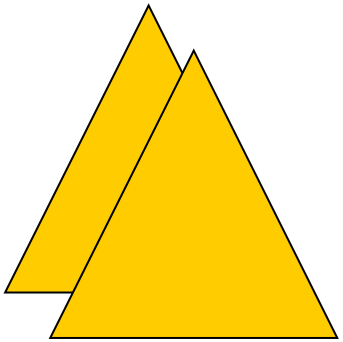
- Kildekode før modifikasjon.
  - Vi skal kun gjøre endringer i RenderFrame(...) i DrawMain.c:326
  - Ignorer resten av koden for nå

Klargjør OpenGL for tegning:

- Nullstill skjerm- og dybdebuffer:
  glClear(..);

- Sett modelview matrix til identitet.
  glLoadIdentity();

Tegne to triangles (trekanter)



```
// tegn trianglene
glBegin(GL_TRIANGLES);
      glNormal3f(0, 0, -1);// innerste triangel
      glVertex3f(-5, -5, -1);      // left
      glVertex3f(0, 5, -1);  // top
      glVertex3f(5, -5, -1);// right

      glNormal3f(0, 0, 1);  // ytterste triangel
      glVertex3f(-5, -5, 1);// left
      glVertex3f(0, 5, 1);   // top
      glVertex3f(5, -5, 1);  // right
glEnd();
```

Tegne to quads (firkanter)

```
glBegin(GL_QUADS);
        glNormal3f(-1, 0, 0);
        glVertex3f(-5, -5, -1);   // bottom left
        glVertex3f(0, 5, -1);     // top left
        glVertex3f(0, 5, 1);      // top right
        glVertex3f(-5, -5, 1);    // bottom right

        glNormal3f(1, 0, 0);
        glVertex3f(5, -5, 1);     // bottom left
        glVertex3f(0, 5, 1);      // top left
        glVertex3f(0, 5, -1);     // top right
        glVertex3f(5, -5, -1);    // bottom right
glEnd();
```

Transformasjoner av trianglene

- Roter triangelet i midten 180 grader
  - glRotatef(vinkel, x, y, z)
- Posisjoner de to andre trianglene litt ut til siden
  - glTranslatef(x, y, z)
- Animer trianglene ved å rotere dem rundt x,y,z aksene.
- Forandre fargen på trianglene.

Posisjoner og roter triangelene

- Legg til følgede under hver glPushMatrix:

```
glPushMatrix();
glTranslatef(-5.2, -5, -35);
glRotatef(x, 1.0f, 1.0f, 1.0f); //360 degrees
x += 45.0 * dElapsed;
```

- Roter det ene triangelet:

```
glRotatef(180, 1.0f, 0.0f, 0.0f);
glRotatef(x, 1.0f, 1.0f, 1.0f);
```

Rotasjonshastighet

```
x += 45.0 * dElapsed;
```

dElapsed bruker QueryPerformanceFrequency fra winbase.h til å ta tiden for hvert sekund som går ved å sjekke CPU klokkefrekvens og tics.

Farge

- Endre farge:

```
const static GLfloat MatYellowDiffuse[ ] =
  { 0.99f, 0.2f, 0.0f, 1.0f };
```