OpenGL



Gustav Taxén
Research Lead
gustav.taxen@avalanchestudios.se
www.avalanchestudios.se

Avalanche Studios

- Sveriges ledande oberoende spelutvecklare
- Fokus på egenutvecklade IPn
- Finns på Söder i Stockholm
- ~160 anställda
- *Just Cause* för PS2, PC, XBox, och XBox 360 släpptes 2006
- Utvecklar *Just Cause 2* samt tre andra projekt


Våra spel

- Stora, öppna spelvärldar
- Sandbox-gameplay
- Hög audiovisuell standard
- Multiplatform
  - PlayStation 3
  - XBox 360
  - PC

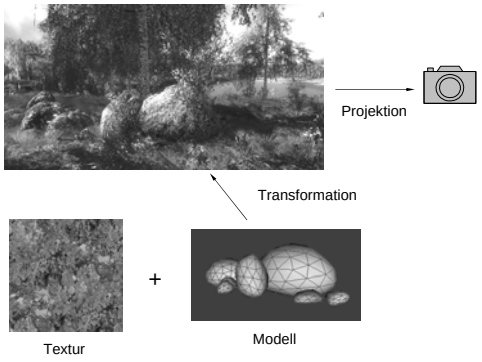

Vår teknik

- Egenutvecklad spelmotor
- R&D genomsyrar hela företaget
- Task force-team
- Nära forskningsfronten
- Samarbeten med universitet och högskolor

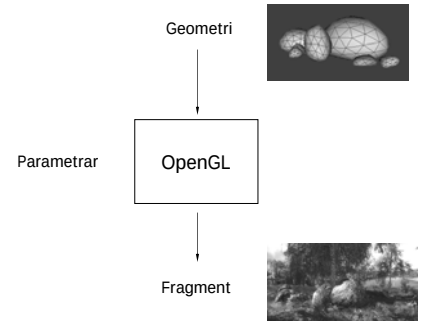
- ***Sök exjobb och jobb hos oss!***
  jobs@avalanchestudios.se


Lite inspiration...

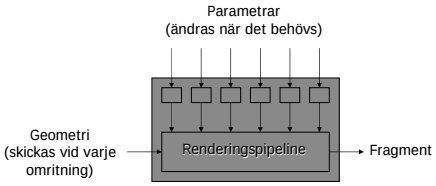
3D-grafiksystem



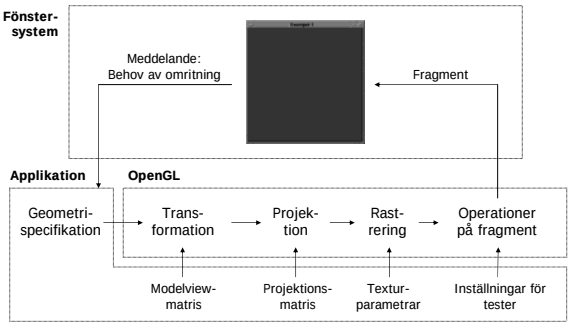
Översikt över OpenGL



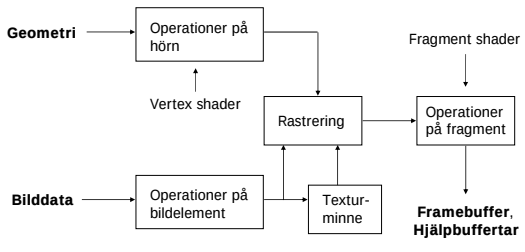
Översikt över OpenGL



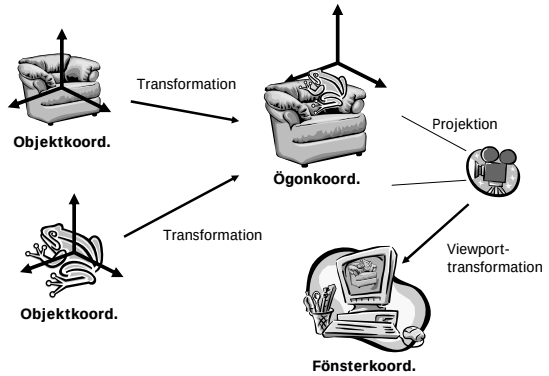
Översikt över OpenGL



Översikt över OpenGL

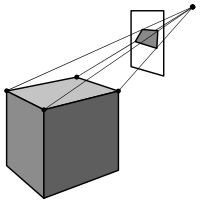


Transformationer

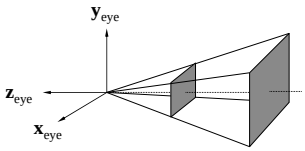


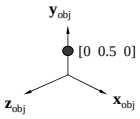
Enpunktsperspektiv

Brook Taylor, **New Principles of Linear Perspective**, 1719.



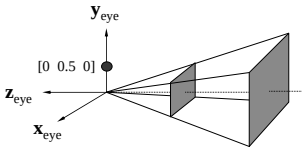
Kameran i OpenGL

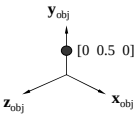




$\mathbf{p}_{eye} = \mathbf{M}\,\mathbf{p}_{obj}$

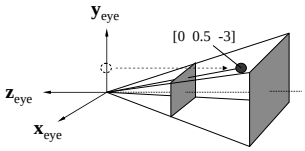
Om $\mathbf{M}$ är enhetsmatrisen...

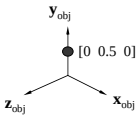




$\mathbf{p}_{eye} = \mathbf{I}\,\mathbf{p}_{obj} = \mathbf{p}_{obj}$

Om $\mathbf{C}$ är en translationsmatris...

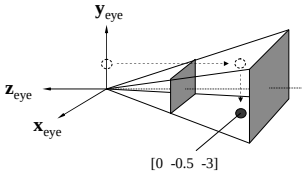




$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

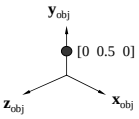
$\mathbf{p}_{eye} = \mathbf{T}\,\mathbf{p}_{obj}$

Sammansatta transformationer

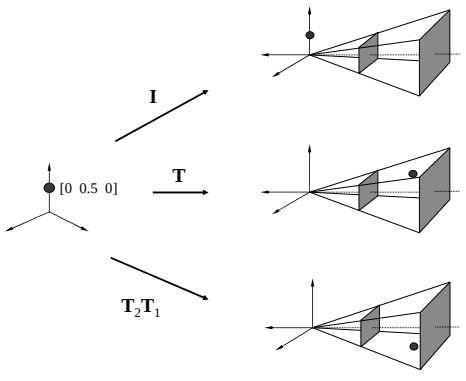


$$\mathbf{T}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$\mathbf{p}_{eye} = \mathbf{T}_2\,(\mathbf{T}_1\,\mathbf{p}_{obj}) = (\mathbf{T}_2\,\mathbf{T}_1)\,\mathbf{p}_{obj}$



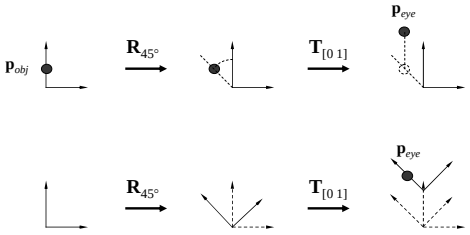
$$\mathbf{p}_{eye} = \mathbf{M}\,\mathbf{p}_{obj}$$

$\mathbf{M}$ styr var punkten $\mathbf{p}_{obj}$ hamnar i ”världen”

Hamnar den ”framför” kameran
(i klippvolymen) syns den

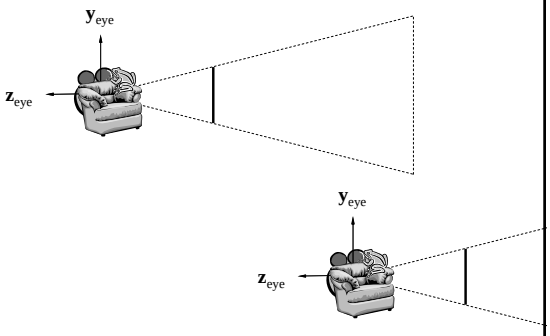
I OpenGL kallas $\mathbf{M}$ ”modelview matrix”

$$\mathbf{p}_{eye} = \mathbf{T}_{[0\ 1]}\,(\mathbf{R}_{45^\circ}\,\mathbf{p}_{obj})$$



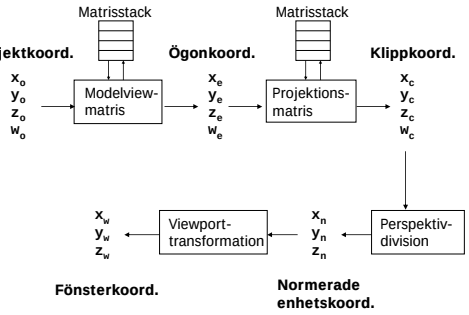
Transformation är ekvivalent med byte av koordinatsystem!

Kameraplacering



Flytta kameran är som att flytta världen fast med en invers transformationsmatris!

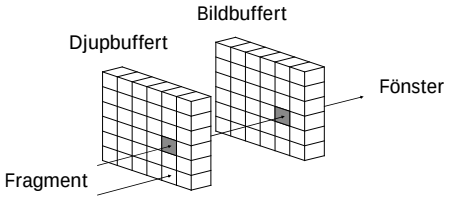
Transformationer i OpenGL



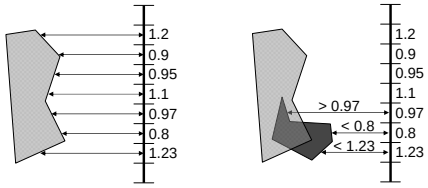
Operationer på fragment

- Ett antal **tester**
- Avgör om fragmentet ska skickas till bildbufferten
- Genomförs alltid i samma ordning
- Ordningen är definierad i OpenGL-specifikationen
- Borttagning av skymda ytor, dynamiska skuggor, specialeffekter, m.m.

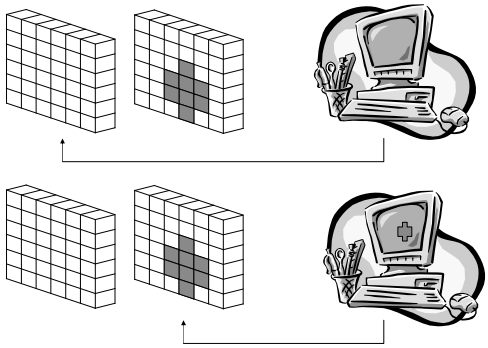
Djuptest



Djuptest (Z-buffring)

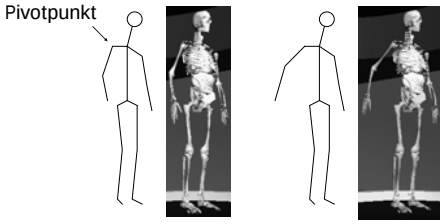


Animation och dubbelbuffring

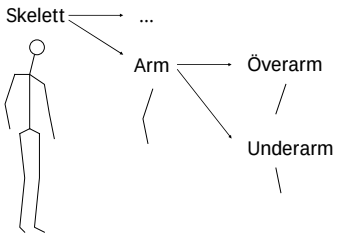


DEMO

Hierarkiska modeller

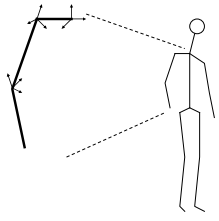


Består-av-hierarki

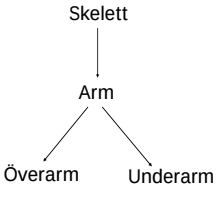


Lokala koordinatsystem

Subnodens origo positioneras m.a.p. **förälderns koordinatsystem**.

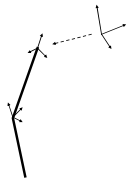


Från hierarki till kod



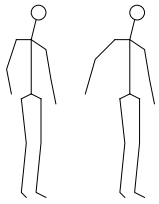

```
void drawSkeleton(void) {
   drawArm();
}

void drawArm(void) {
   drawUpperArm();
   drawLowerArm();
}

void drawUpperArm(void) {
}

void drawLowerArm(void) {
}
```

Från hierarki till kod



```
void drawSkeleton(void) {
   glPushMatrix();
   positionera armens origo;
   drawArm();
   glPopMatrix();
}

void drawArm(void) {
   glPushMatrix();
   positionera överarmens origo;
   drawUpperArm();
   glPopMatrix();

   glPushMatrix();
   positionera underarmens origo;
   drawLowerArm();
   glPopMatrix();
}
```

Från hierarki till kod



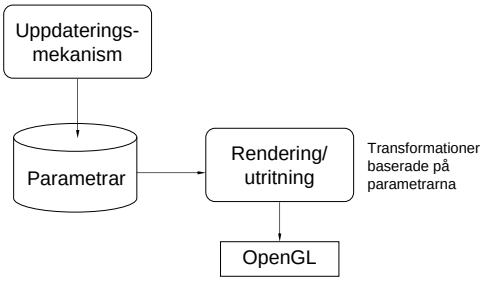
Ändra rotation här!

```
void drawSkeleton(void) {
   glPushMatrix ();
   positionera armens origo;
   drawArm ();
   glPopMatrix ();
}
```

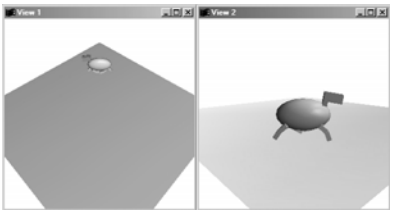
Från hierarki till kod

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();

glPushMatrix();
glTranslatef(modellens position);
rita torso;

  glPushMatrix();
  glTranslatef(pivotpunkt för överarmen);
  glRotatef(grader, rotationsaxel);
  rita överarmen;

    glPushMatrix();
    glTranslatef(pivotpunkt för underarmen);
    glRotatef(grader, rotationsaxel);
    rita underarmen;
    glPopMatrix();

  glPopMatrix();

glPopMatrix();
```

Animation



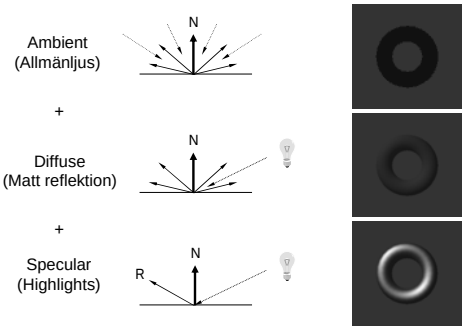
Demo - forward kinematics



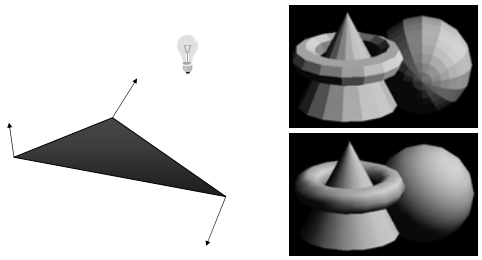
$param_0 = f_0(t)$
$param_1 = f_1(t)$
...

Kända

Ljussättning – Phongs reflektionsmodell

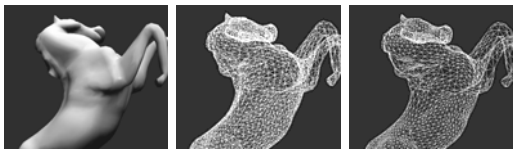


Goraud shading

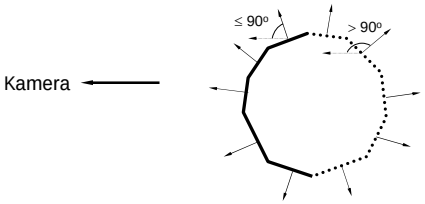


Färg beräknas i varje hörn och
interpoleras sedan linjärt över polygonen.
Normaler måste anges före varje hörn!

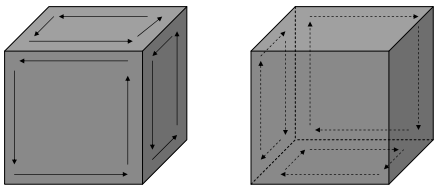
Back face culling



Back face culling



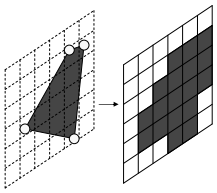
Fram- eller baksida?



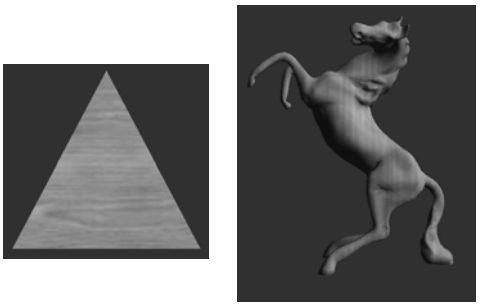
Definieras av hörnens ”ordning” sett från kameran.
Konventionen är att **motsols är riktat mot kameran**.

Rastrering

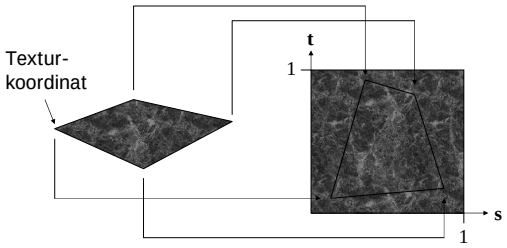
- "Översätter" från hörn till pixelpunkter
- Ett fragment består av
  - **Position**: (x, y, z) i normerade enhetskoordinater
  - **Färg**: (R, G, B, A)
  - **Texturkoordinater**: (s, t, q, r)



Texturer

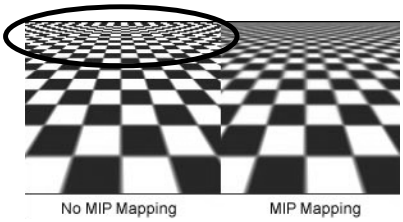


Texturkoordinater

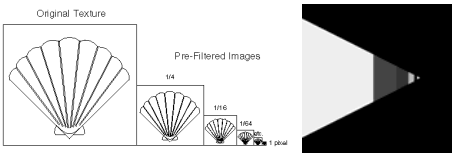


MIP-mapping

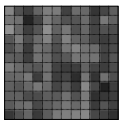
Teknik för att undvika aliasing-problem.



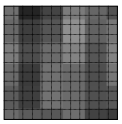
MIP-mapping



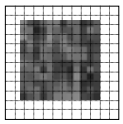
Förstoring och förminskning



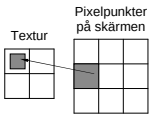
Texelstorlek = pixelstorlek
(inträffar i princip aldrig)

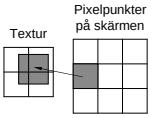


Förstoring

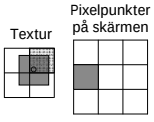


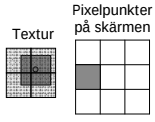
Förminskning



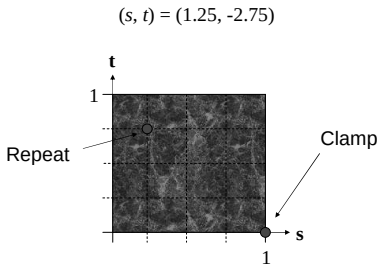


Filtreringsalternativ

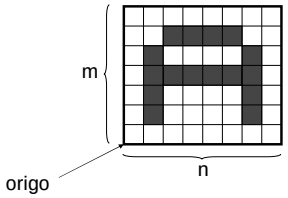




Clamp / repeat

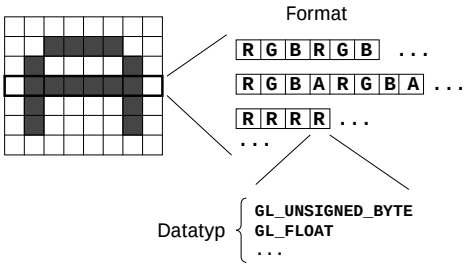


Bilddata: Texels



(**`width`**, **`height`**) = (2^n, 2^m)
där m, n är heltal ≥ 0

Bilddata: Format och datatyper



DEMO

gustav.taxen@avalanchestudios.se
jobs@avalanchestudios.se
www.avalanchestudios.se