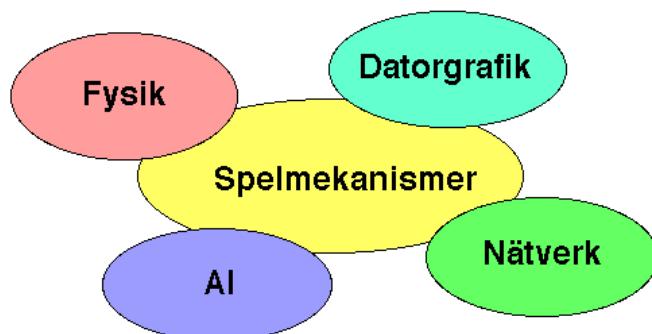


TSBK 10

Teknik för avancerade datorspel

Ingemar Ragnemalm, ISY



Ingemar
Ragnemalm
ingis@isy.liu.se

Föreläsning 4

- En ny blick på 3D-pipelinen
- Allmänt om programmerbara shaders
- Introduktion till GLSL

Ingemar
Ragnemalm
ingis@isy.liu.se

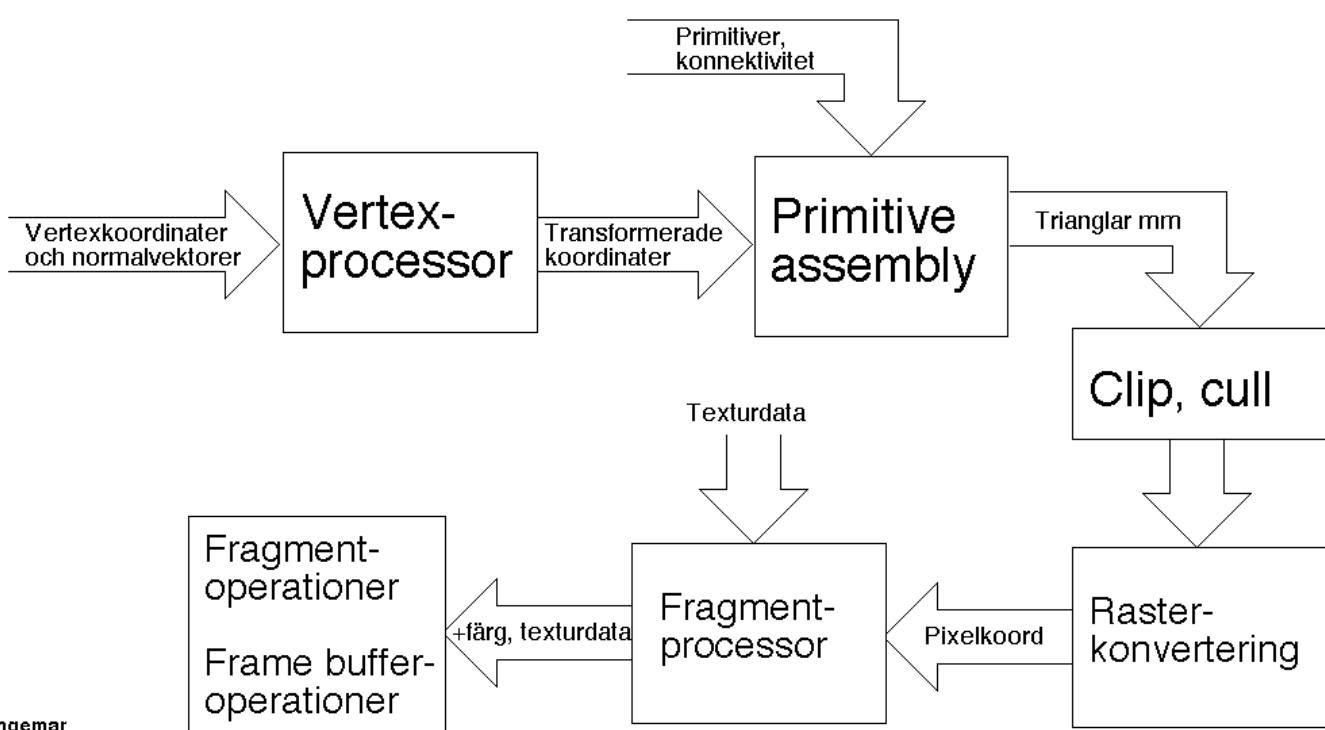
***Now is an excellent time to be working in
the field of computer graphics***

David Kirk, NVidia

Ingemar
Ragnemalm
ingis@isy.liu.se

3D-pipelinen i GPU'n

Lågnivåoperationer på vägen till pixeldata

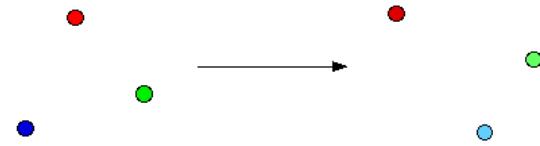


Ingemar
Ragnemalm
ingis@isy.liu.se

Vertexprocessorn

Vertexprocessorn utför följande uppgifter:

- Vertextransformation (från modellkoordinater till skärmkoordinater)
- Transformation av normalvektorer
- Generering av texturkoordinater
- Transformation av texturkoordinater
- Ljussättning
- Materialparametrar



Ingemar
Ragnemalm
ingis@isy.liu.se

Primitive assembly

Sammansättning av primitiver

“Primitiv” inte som i simpel utan som i geometriska primitiver

De transformerade koordinaterna samlas i strukturer som beskriver enskilda trianglar mm.
För varje vertex i ett primitiv läses skärmkoordinater ut från vertextabellen, indexerat med triangeltabellen.



Ingemar
Ragnemalm
ingis@isy.liu.se

Clipping och culling

Primitiverna klipps till skärmens gränser och backface culling utförs.

Notera att även texturkoordinater behöver klippas (och eventuella ytterliggare data som interpoleras mellan vertexar).

Ingemar
Ragnemalm
ingis@isy.liu.se

Rasterkonvertering

Polygonrendering, konvertera polygoner till pixelkoordinater

Skapar “fragment”. Observera att de än så länge inte har några färgvärden!



Ingemar
Ragnemalm
ingis@isy.liu.se

Fragmentprocessorn

Givet pixelkoordinater och interpolerade data för färg, textur mm, beräkna ett färgvärde för fragmentet.

- Texturering
- Dimma
- Färgsummering



Ingemar
Ragnemalm
ingis@isy.liu.se

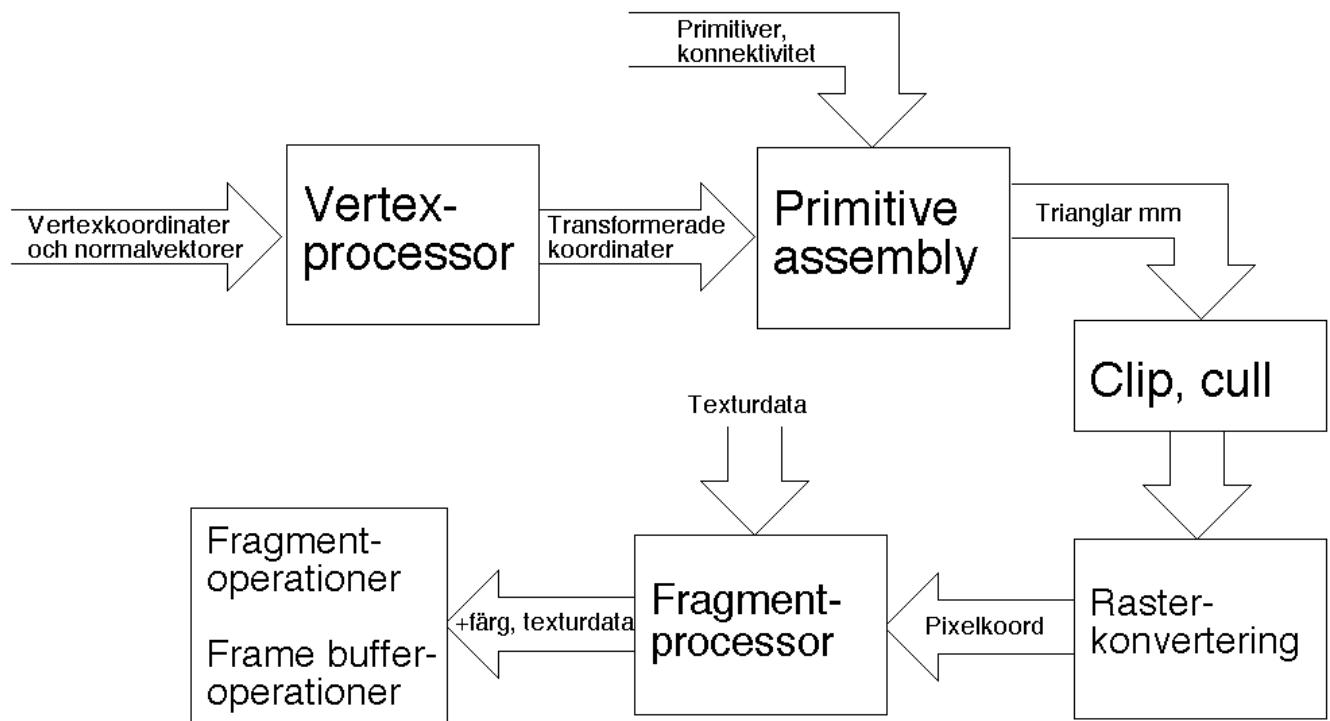
Fragmentoperationer

Avslutande operationer innan skrivning i framebuffern

- **Stenciltest**
- **Z-buffertest**
- **Blend-funktion (`glBlendFunc` mm)**
- **Alpha-funktion (`glAlphaFunc`)**

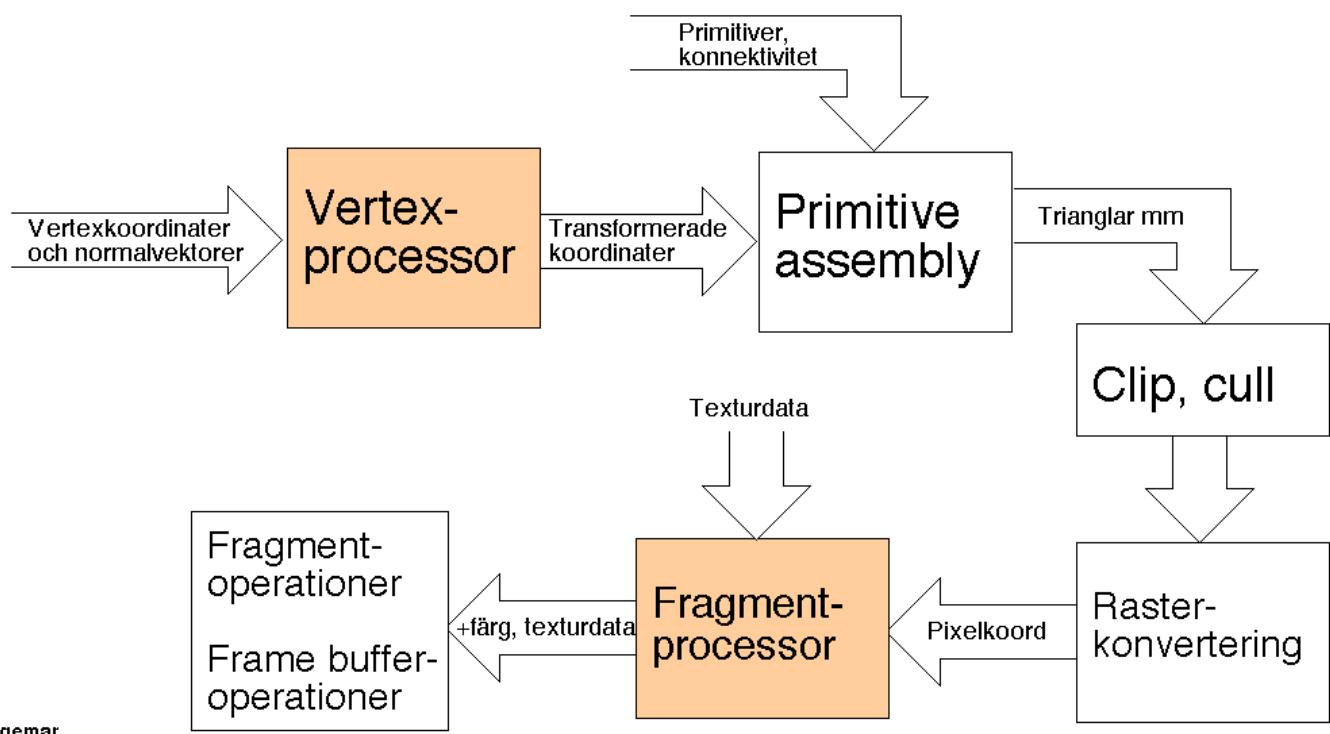
Ingemar
Ragnemalm
ingis@isy.liu.se

Två av dessa är programmerbara!



Ingemar
Ragnemalm
ingis@isy.liu.se

Två av dessa är programmerbara!



Ingemar
Ragnemalm
ingis@isy.liu.se

Shaderprogram

Programsnuttar som exekveras per vertex eller per fragment (pixel), på GPU'n!

Två program samverkar, ett vertexprogram och ett fragmentprogram.

“Shader” betyder inte att det enbart är till för ljussättning. Ljussättning är dock ett av målen.

Vertextransform

Vertexfärg, vertexljussättning

} Kan göras med vertex-shader

Texturering

Färgsättning, ljussättning per pixel

} Kan göras med fragment-shader

Ingemar
Ragnemalm
ingis@isy.liu.se

Vertex-shader

Ersätter fasta funktionaliteten i vertexprocessorn

Kan:

- Transformera vertexar, normaler, texturkoordinater
- Generera texturkoordinater
- Ljussätta per vertex
- Sätta värden som skall interpoleras för fragment shader

Vet inget om:

- Perspektiv, viewport
- Frustum
- Primitiver (!)
- Culling

...

Ingemar
Ragnemalm
ingis@isy.liu.se

Fragment-shader

(=pixel shader)

Ersätter fragmentprocessorns fasta funktionalitet

Kan:

- Sätta fragmentfärgen
- Hämta pixelvärdet från texturer
- Beräkna dimma, färgsummering
- Använda interpolerade värden från vertexar

Kan inte:

- Ändra fragmentets position
- Skriva i texturer
- Påverka stencil, scissor, alpha, depth...

Ingemar
Ragnemalm
ingis@isy.liu.se

Shaderspråk

Fyra olika:

Assemblerspråk: Användes i början, på väg ut, stödet för det uppdateras inte längre.

Cg: “C for graphics”, NVidia

HLSL: “High-level shading language”, Microsoft

GLSL: “OpenGL shading language”

Assemblerlösningar på väg ut.

Valet mellan de andra tre beror på plattform och behov (och smak?).

Ingemar
Ragnemalm
ingis@isy.liu.se

Shaderspråk

Cg:

- + Fungerar både i OpenGL och DirectX
- + Så gott som källkodskompatibel med HLSL
- + Starkaste GPU-tillverkaren bakom
- Ej öppen standard

HLSL:

- + Så gott som källkodskompatibel med Cg
- + Stora pengar bakom
- Ej öppen standard
- Fungerar inte i OpenGL (än?)

GLSL: “OpenGL shading language”

- + Öppen standard
- + En del av OpenGL 2.0.
- Fungerar inte i DirectX (än?)

Ingemar
Ragnemalm
ingis@isy.liu.se

Intressant citat

For a cross-platform game targeting both DX and OGL with a single shader solution, Cg is a godsend. Cg is also great for learning shaders.

ARB_VERTEX/FRAGMENT_PROGRAM may be better as a shipping solution.... And obviously GLSL will be what you want to use in the future.

(Från ett spelutvecklingsforum)

Observera att tre av de fyra språken nämns och anses värdiga att överväga, och HLSL är mellan raderna eftersom det är så gott som identiskt med Cg.

Valet är inte självklart, men GLSL är ett starkt val.

Ingemar
Ragnemalm
ingis@isy.liu.se