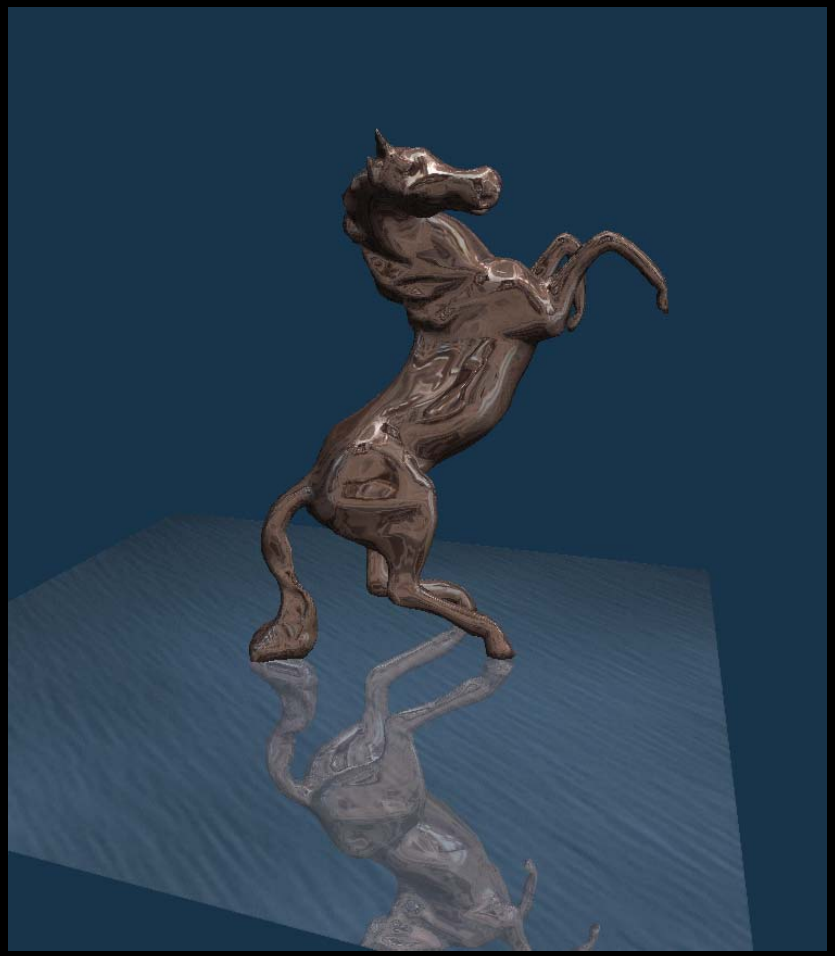
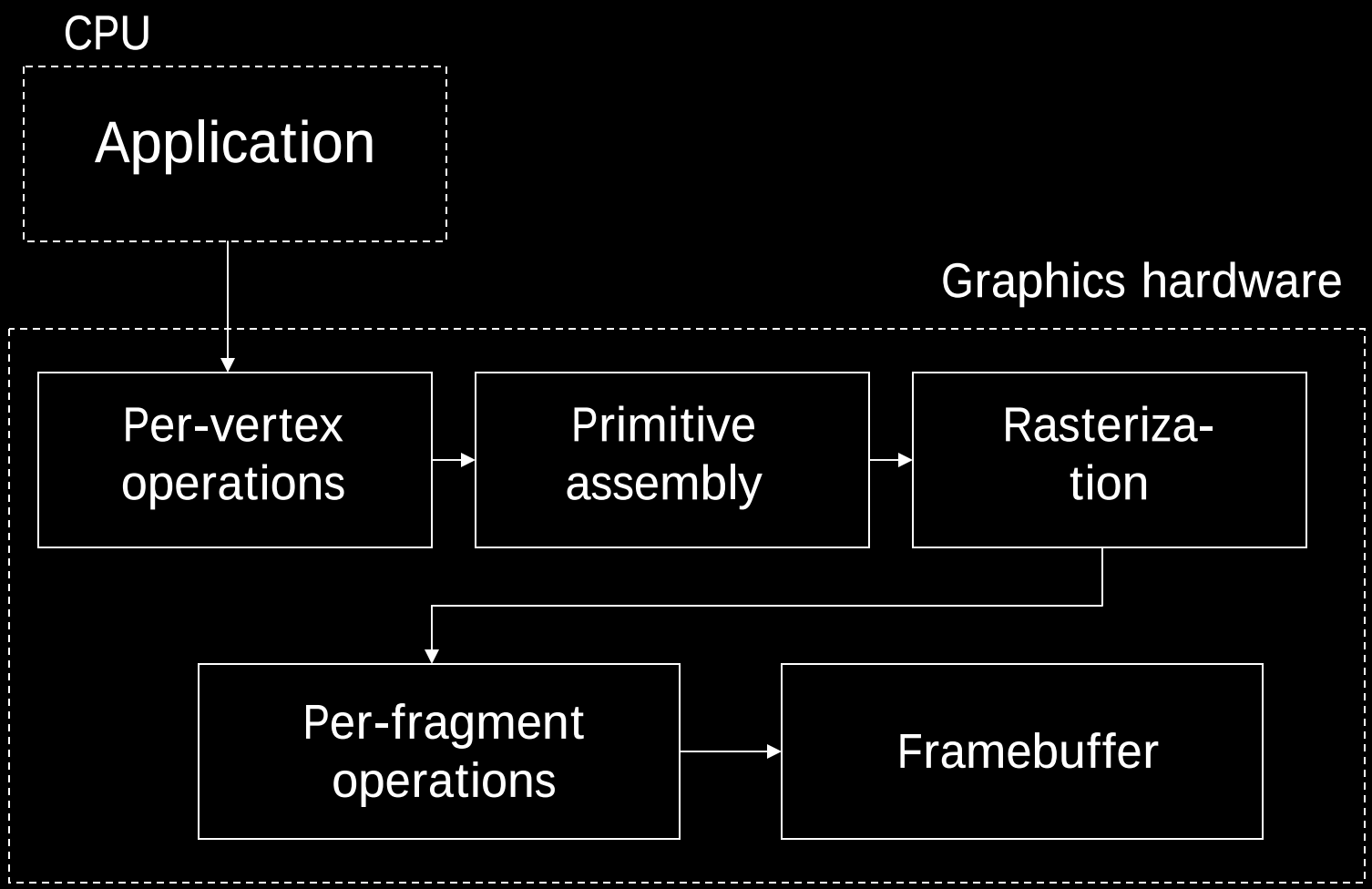


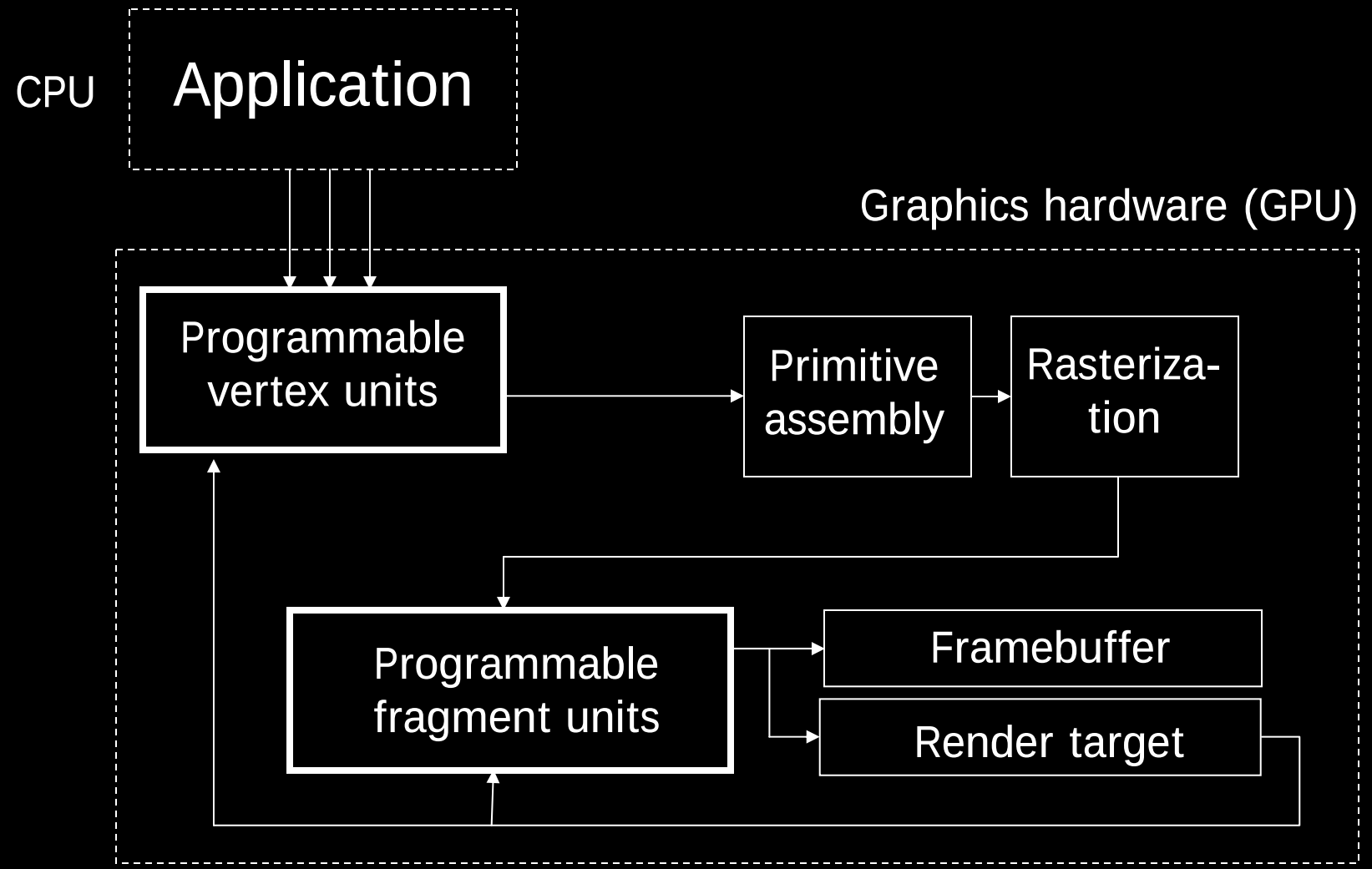
OpenGL

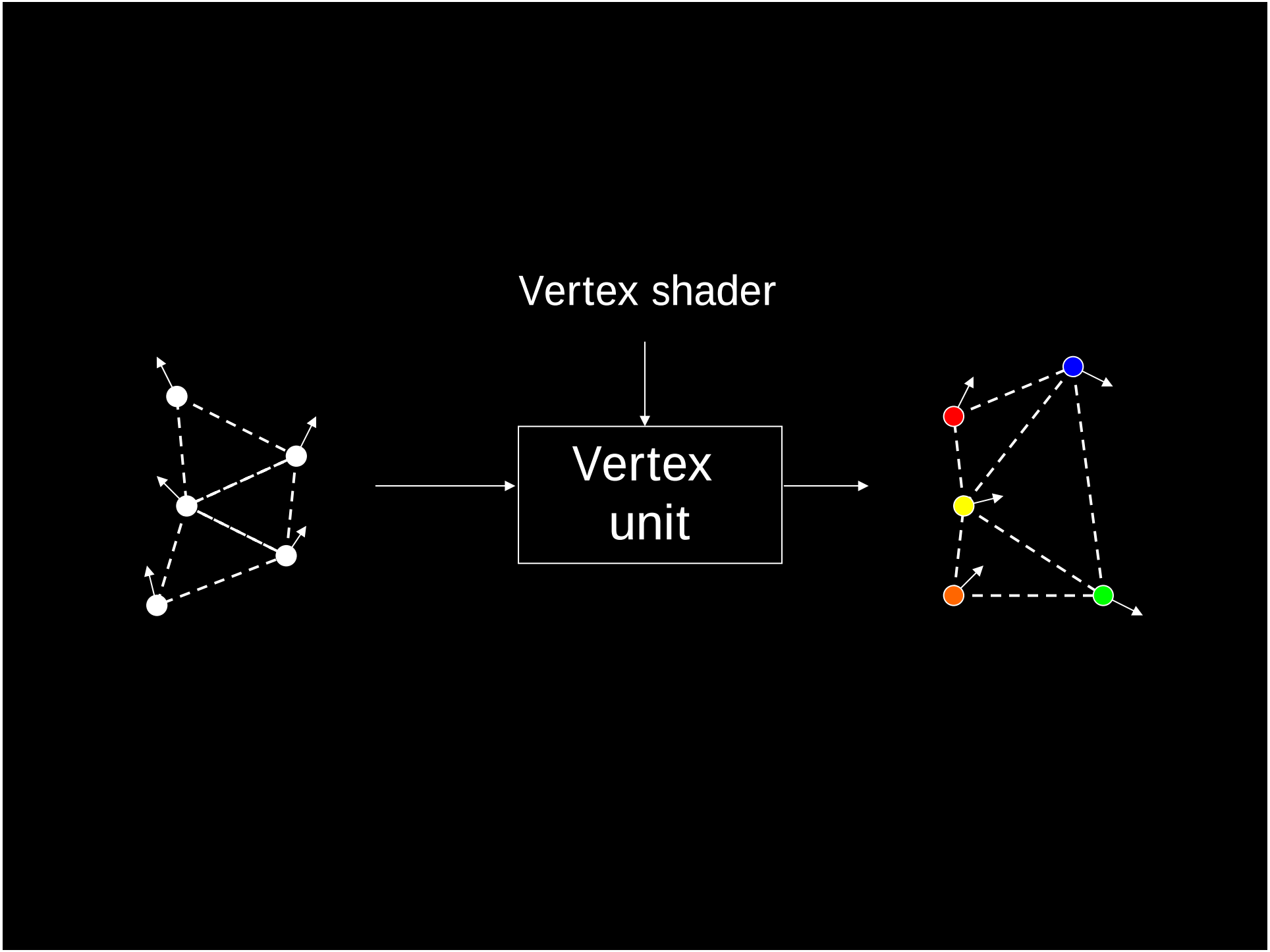
Gustav Taxén
gustavt@csc.kth.se

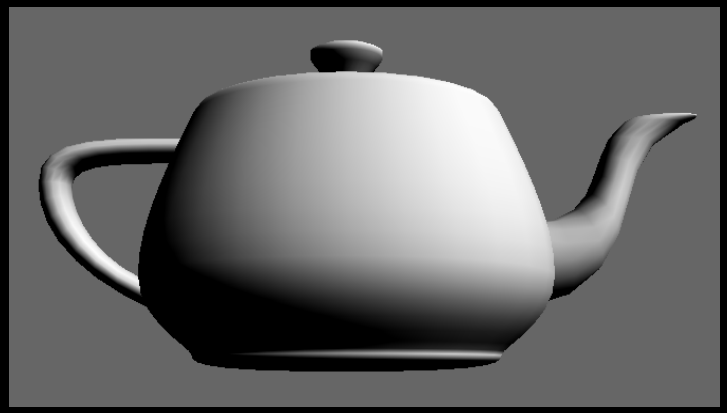
Fixed-function pipeline

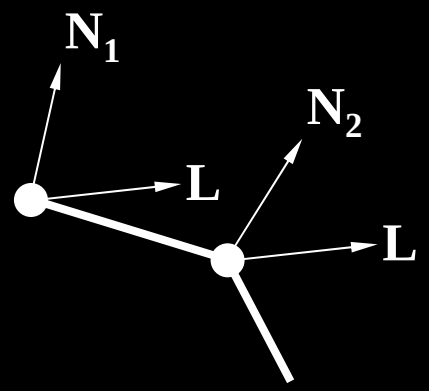


Programmerbar pipeline

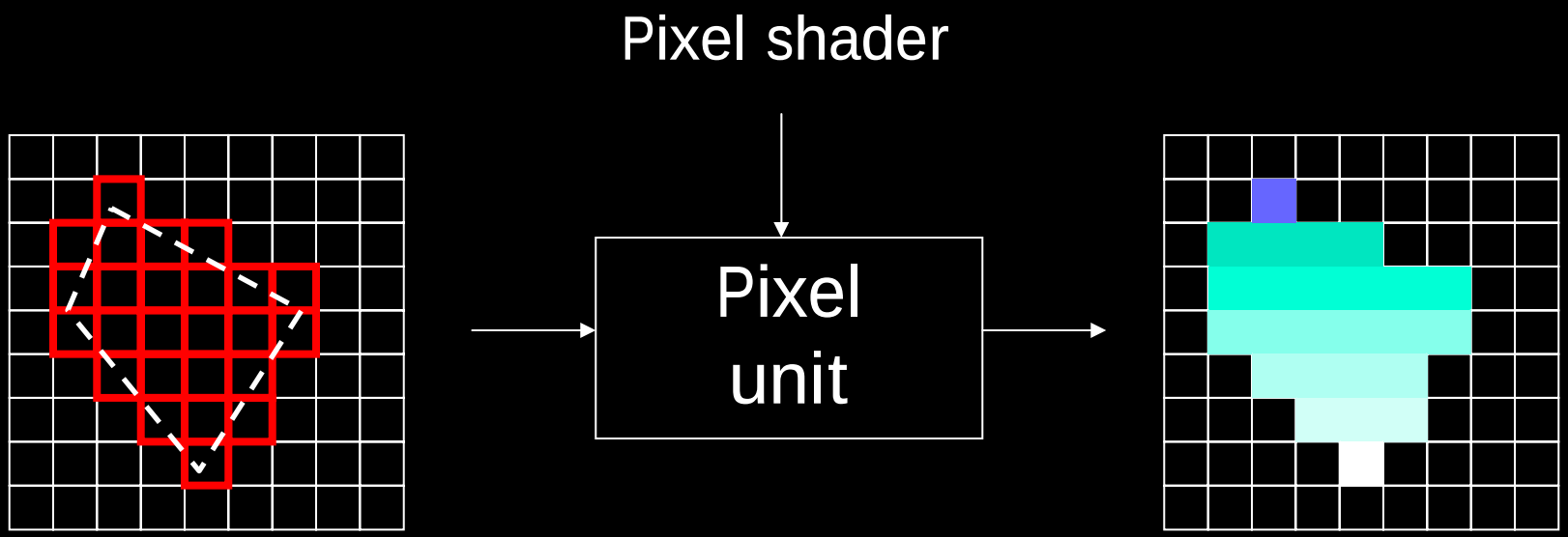


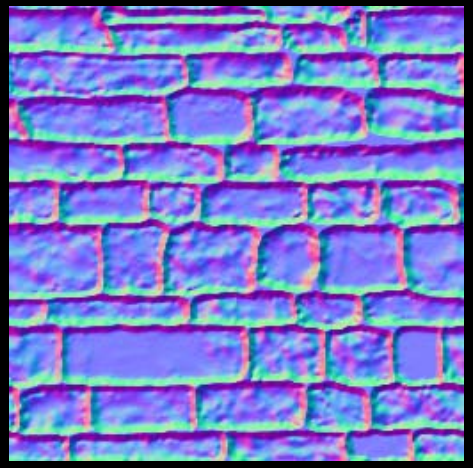

Vertex shader
Vertex
unit






```
float3 main(float3 N : NORMAL,
            uniform float3 L)
{
  float d = dot(N, L);
  return float3(d, d, d);
}
```

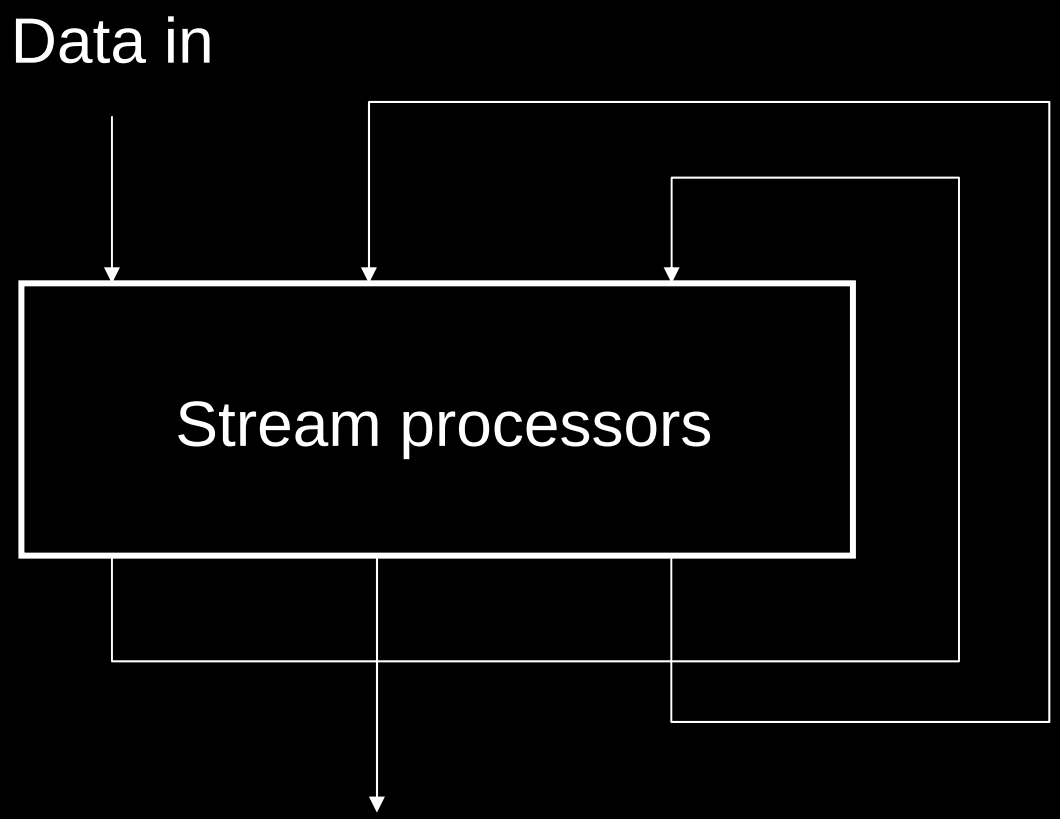

Pixel shader
Pixel
unit



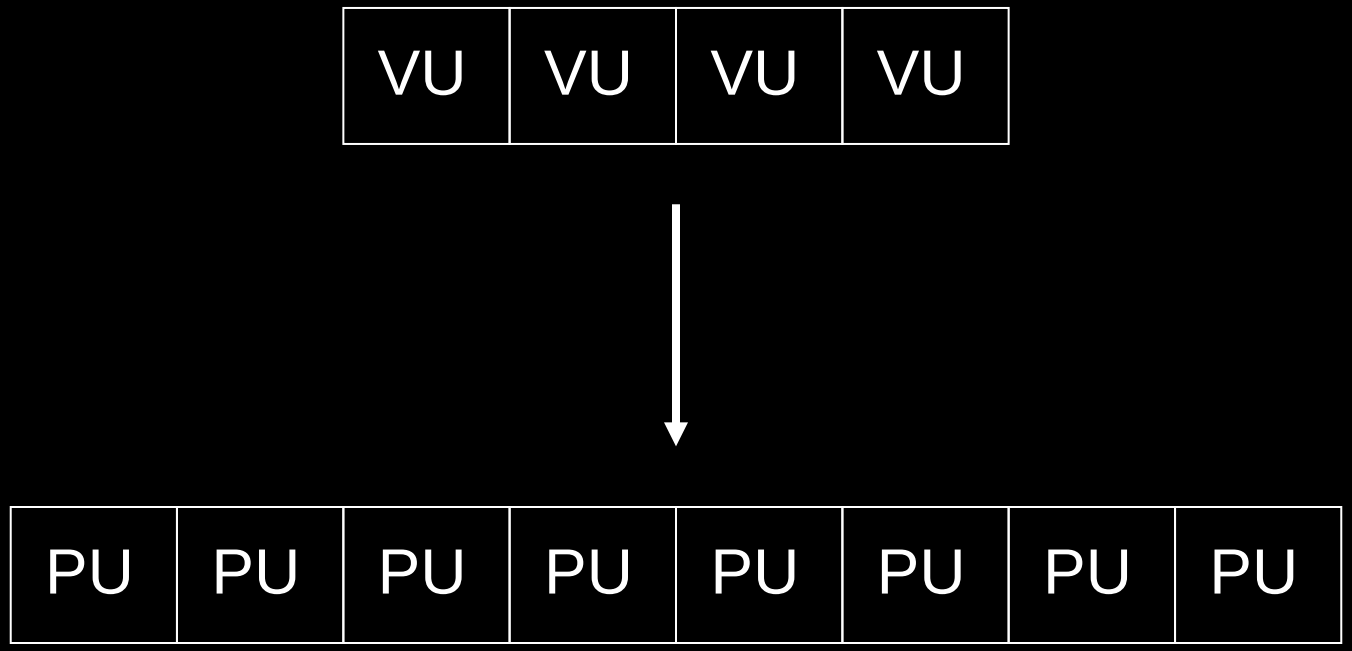
http://nehe.gamedev.net

```
float3 main(float2 uv : TEXCOORD0,
            sampler2D normalMap,
            uniform float3 L)
{
  float3 N = tex2D(normalMap, uv);
  float d = dot(N, L);
  return float3(d, d, d);
}
```

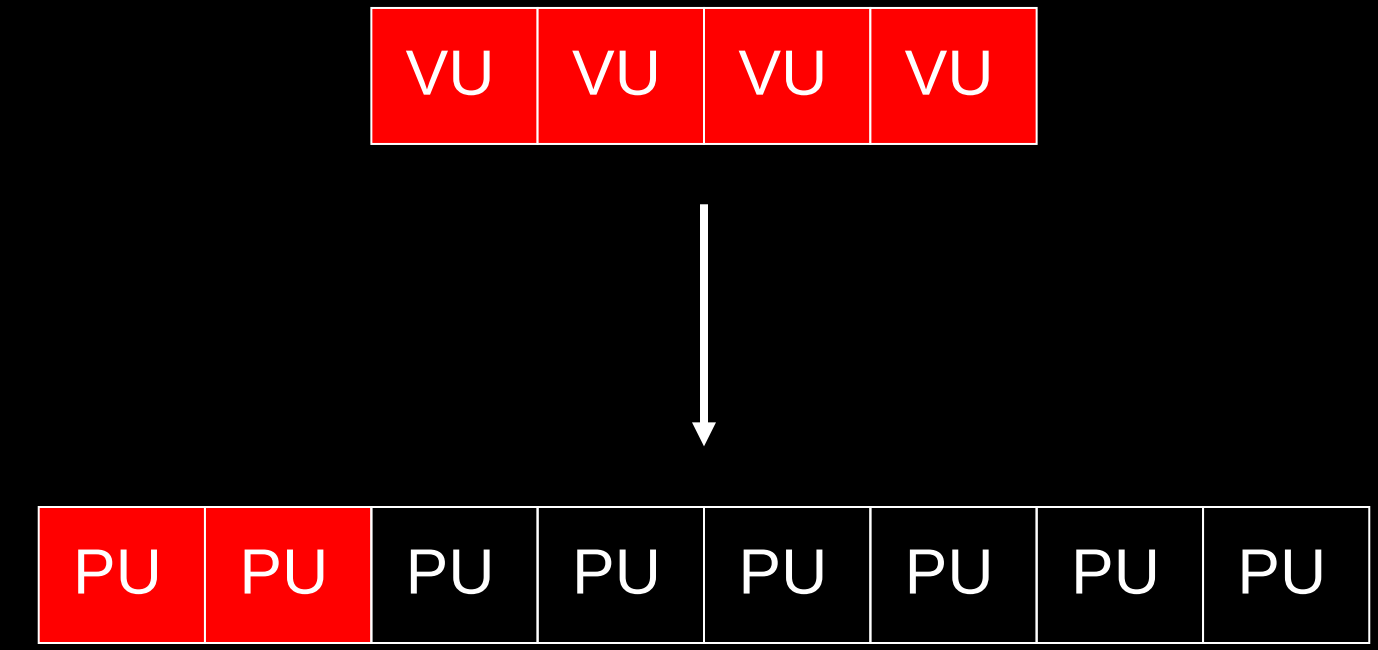
Stream processors



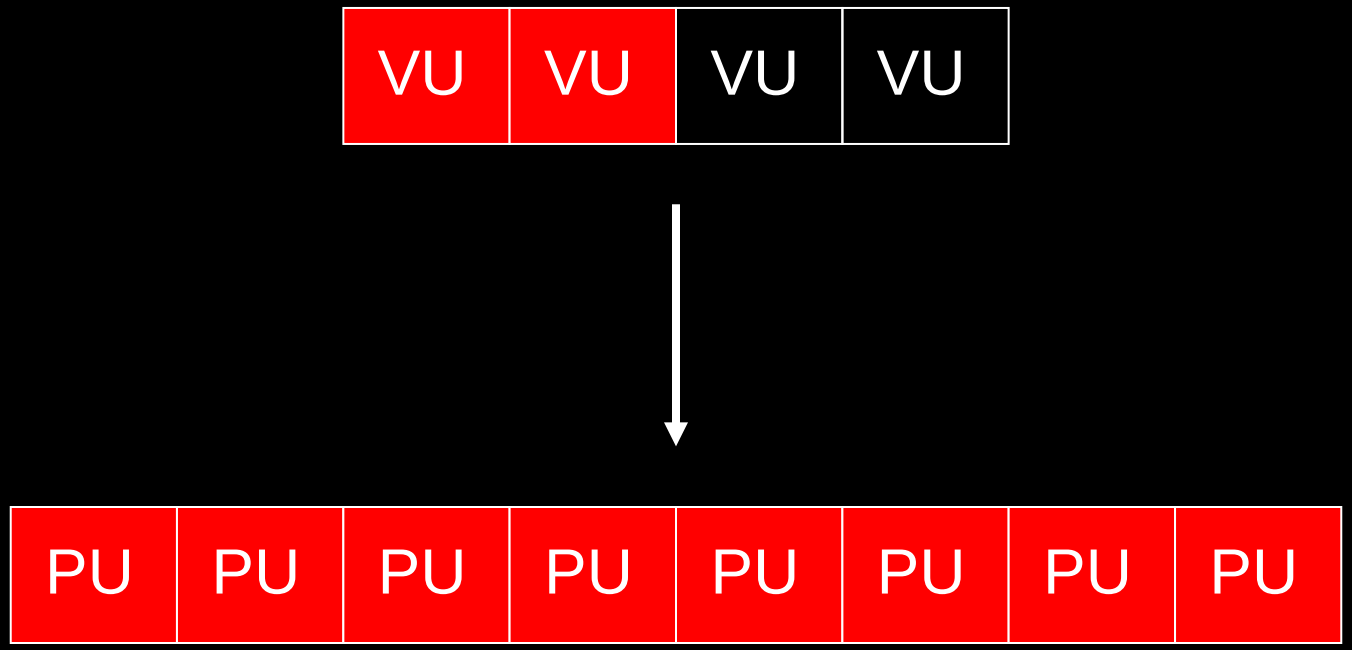
4 Vertex Units + 8 Pixel Units



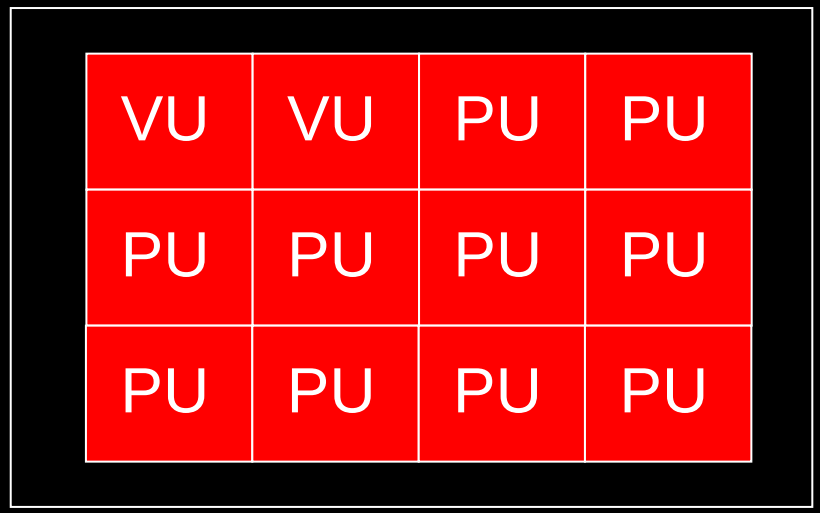
Vertex-intensive application

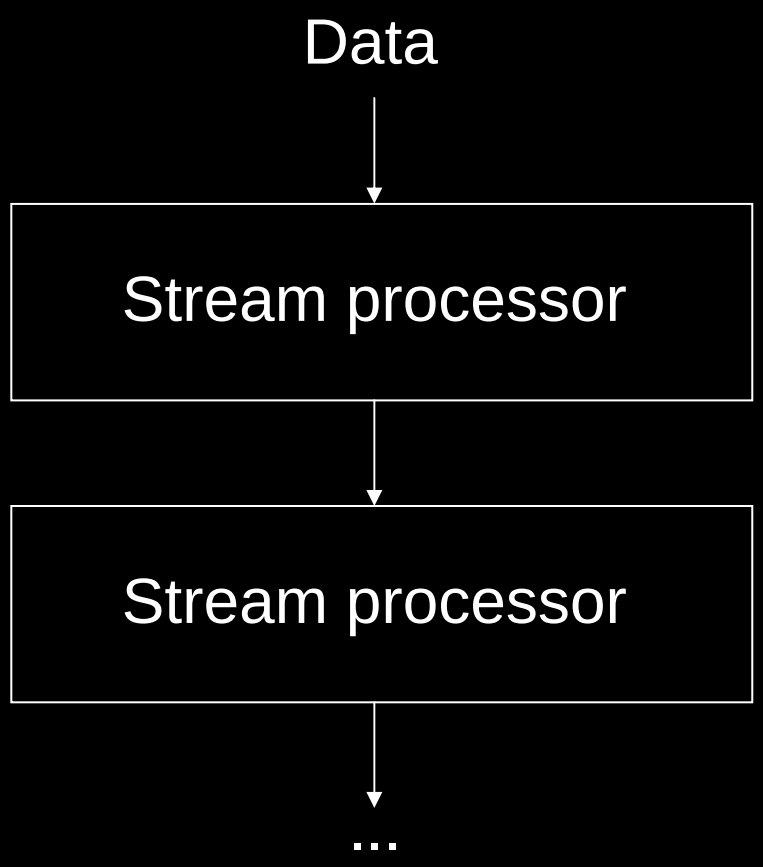


Pixel-intensive application



12 stream processors

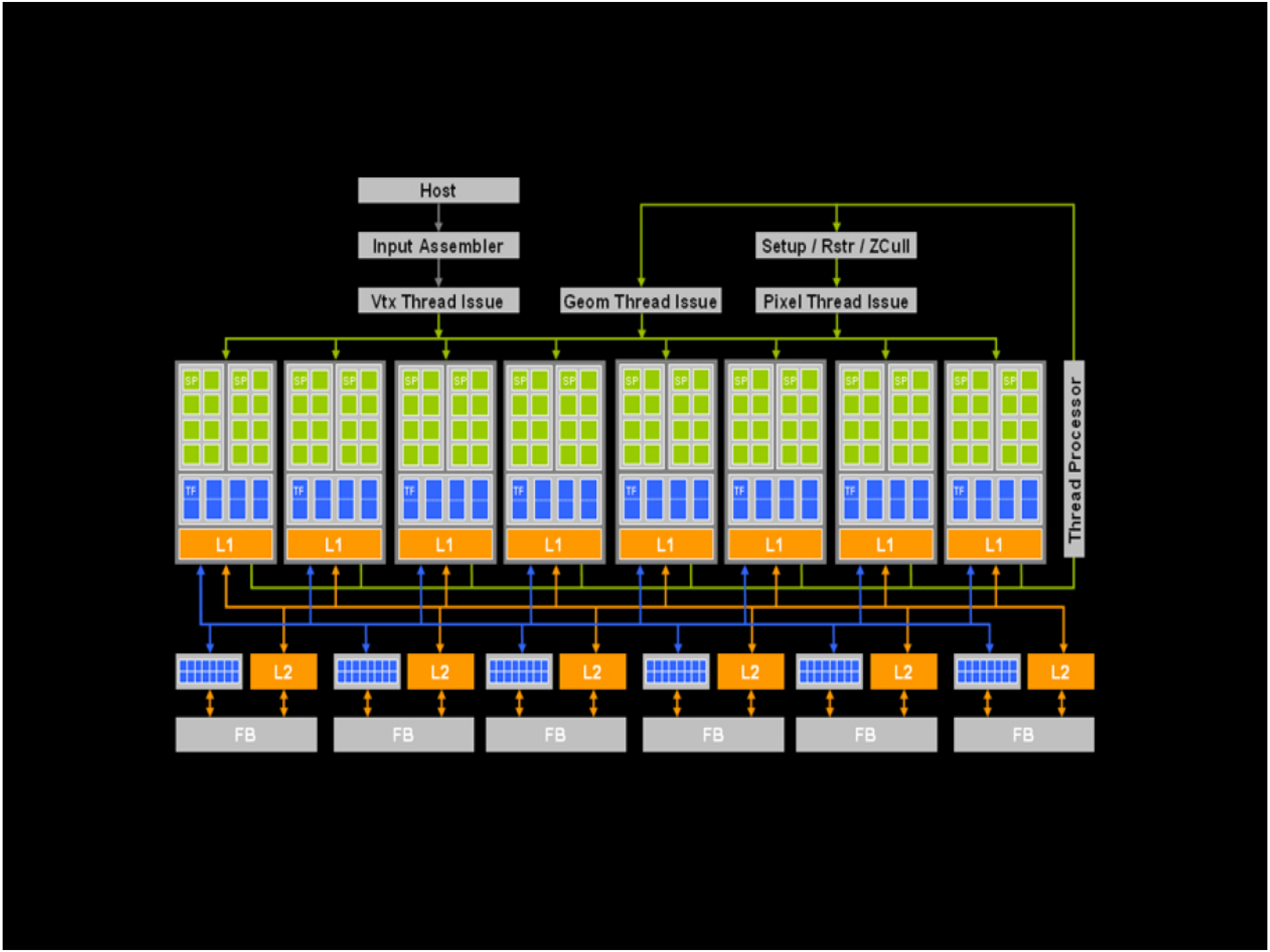


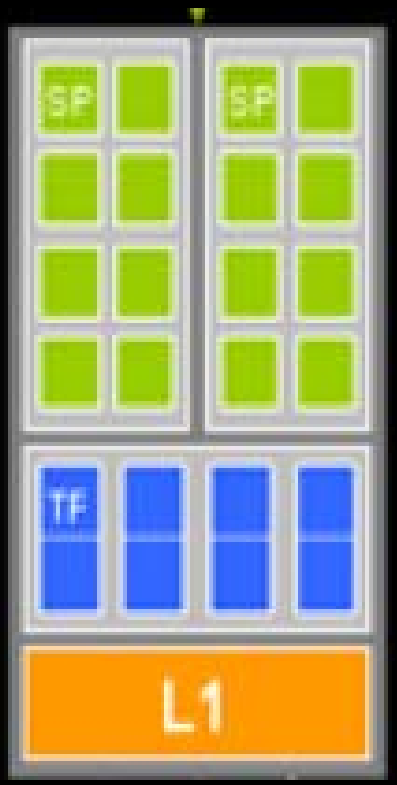


- Limited instruction set
  - Optimized for 4D float vectors
- Inefficient branching
- Small, fast RAM
- Small, fast cache
- Excellent for SIMD ops

nVidia GeForce 8800

- 128 stream processors @ 1.35 GHz
- 2560 x 1600
- 16x antialiasing using multisampling
- Up to 8 render targets, up to FP32 (128 bit) formats
- SLI (x2)
- Full support for Direct3D 10, OpenGL 2.0 and CUDA
- Blue-Ray-, HD-DVD-codecs
- API for physics (Quantum Effects)

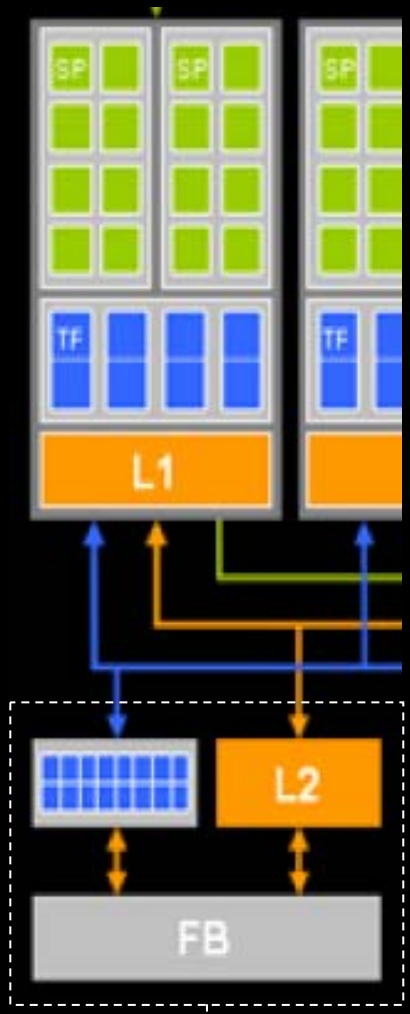

Host
Input Assembler
Vtx Thread Issue
Geom Thread Issue
Setup / Rstr / ZCull
Pixel Thread Issue
Thread Processor
SP
TF
L1
L2
FB



Stream processors

Texture lookup and filtering

Level 1 cache

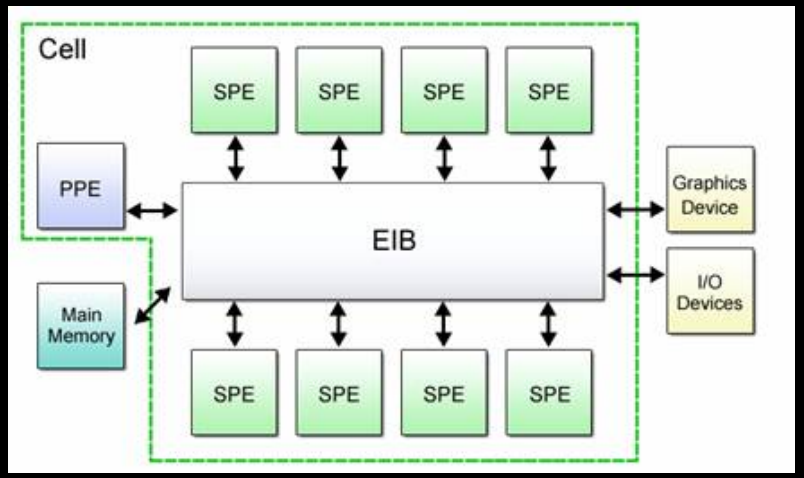


General purpose registers

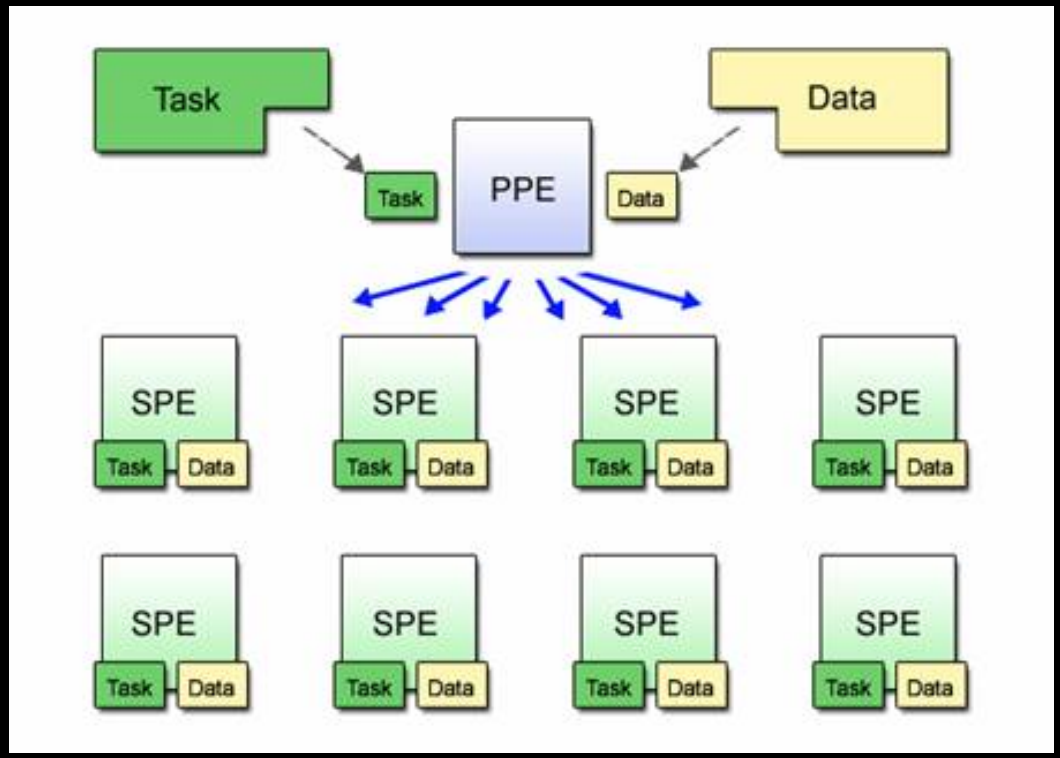
Level 2 cache

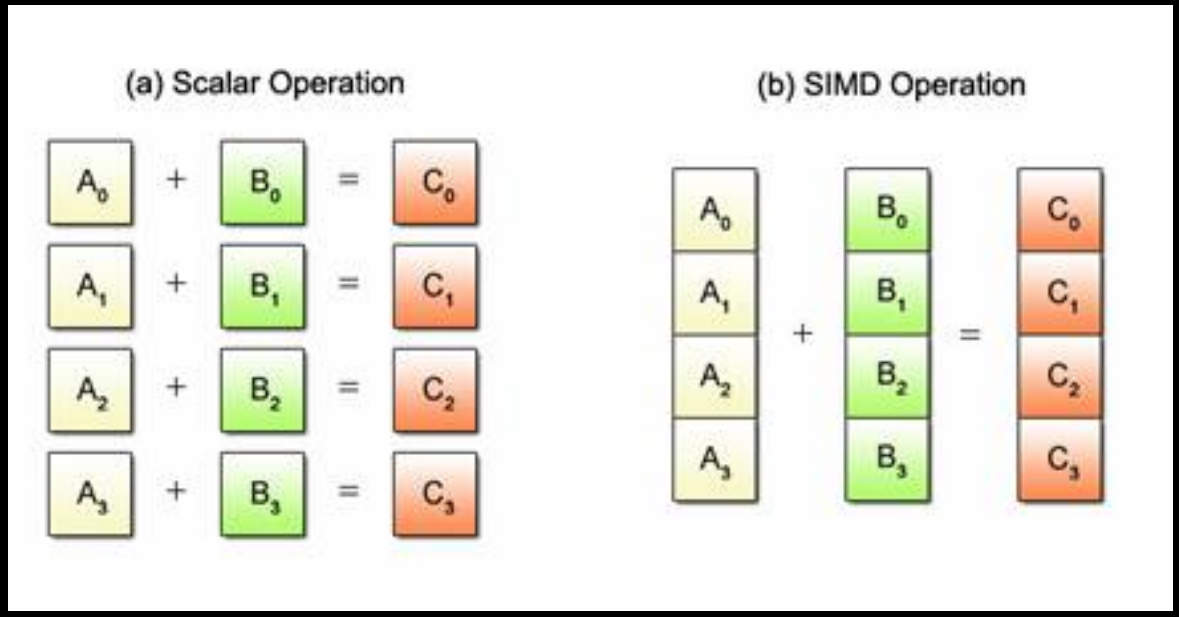
Frame buffer

Raster ops

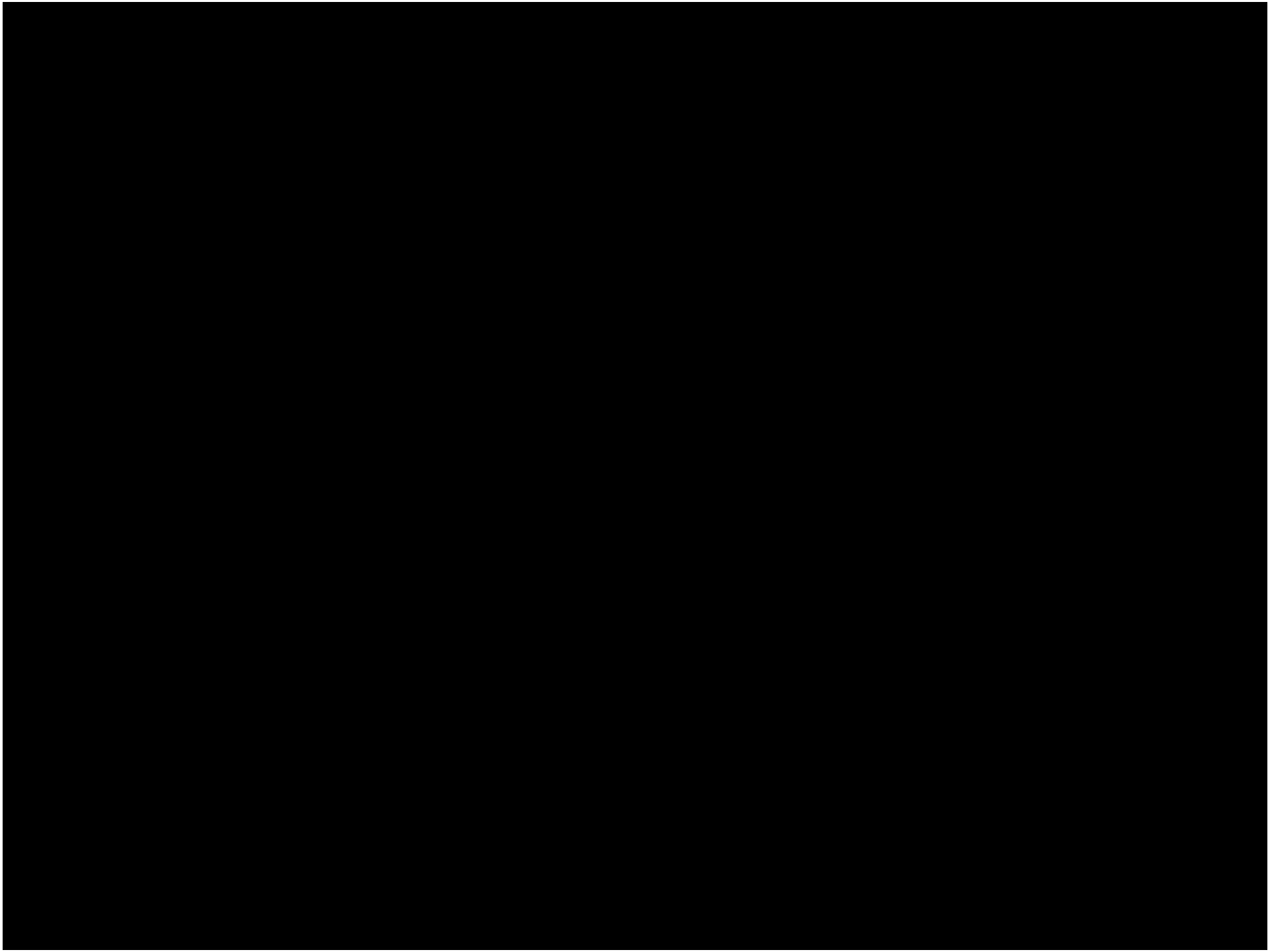


- PowerPC Processor Element (PPE)
  - Memory management, task management
- Synergistic Processor Unit (SPU)
  - Special instruction set
  - 128 bytes memory (prog + data)
  - 8 Synergistic Processing Elements (SPE)
  - Designed for SIMD-algorithms
- Element Interconnect Bus (EIC)
  - Primary memory bus

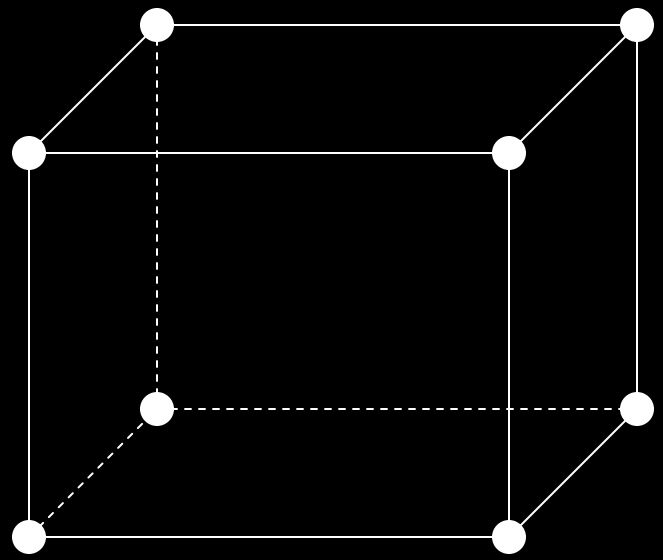

Task
Data
PPE
Task
Data
SPE
Task
Data
SPE
Task
Data
SPE
Task
Data
SPE
Task
Data
SPE
Task
Data
SPE
Task
Data
SPE
Task
Data
SPE
Task
Data



Cell BE uses a vector datatype with 16 bytes
Extended C/C++ provides access



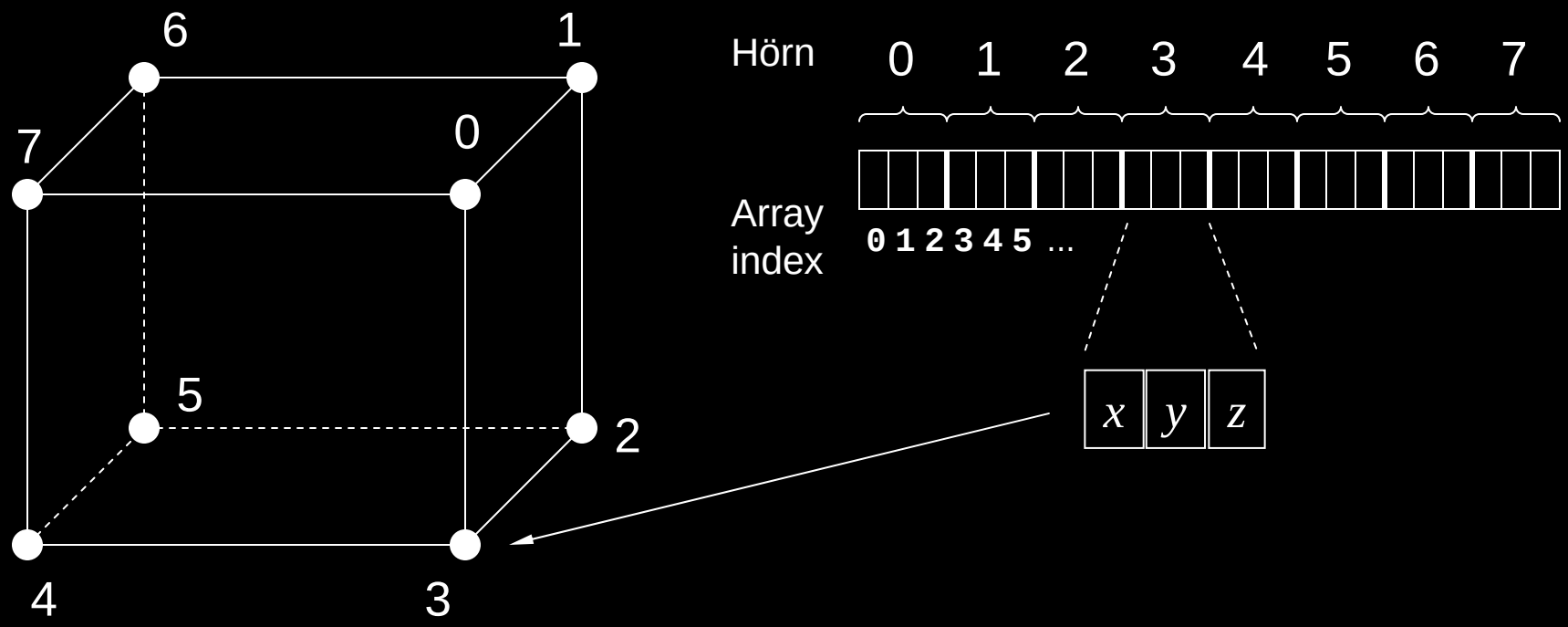
Geometrispecifikation



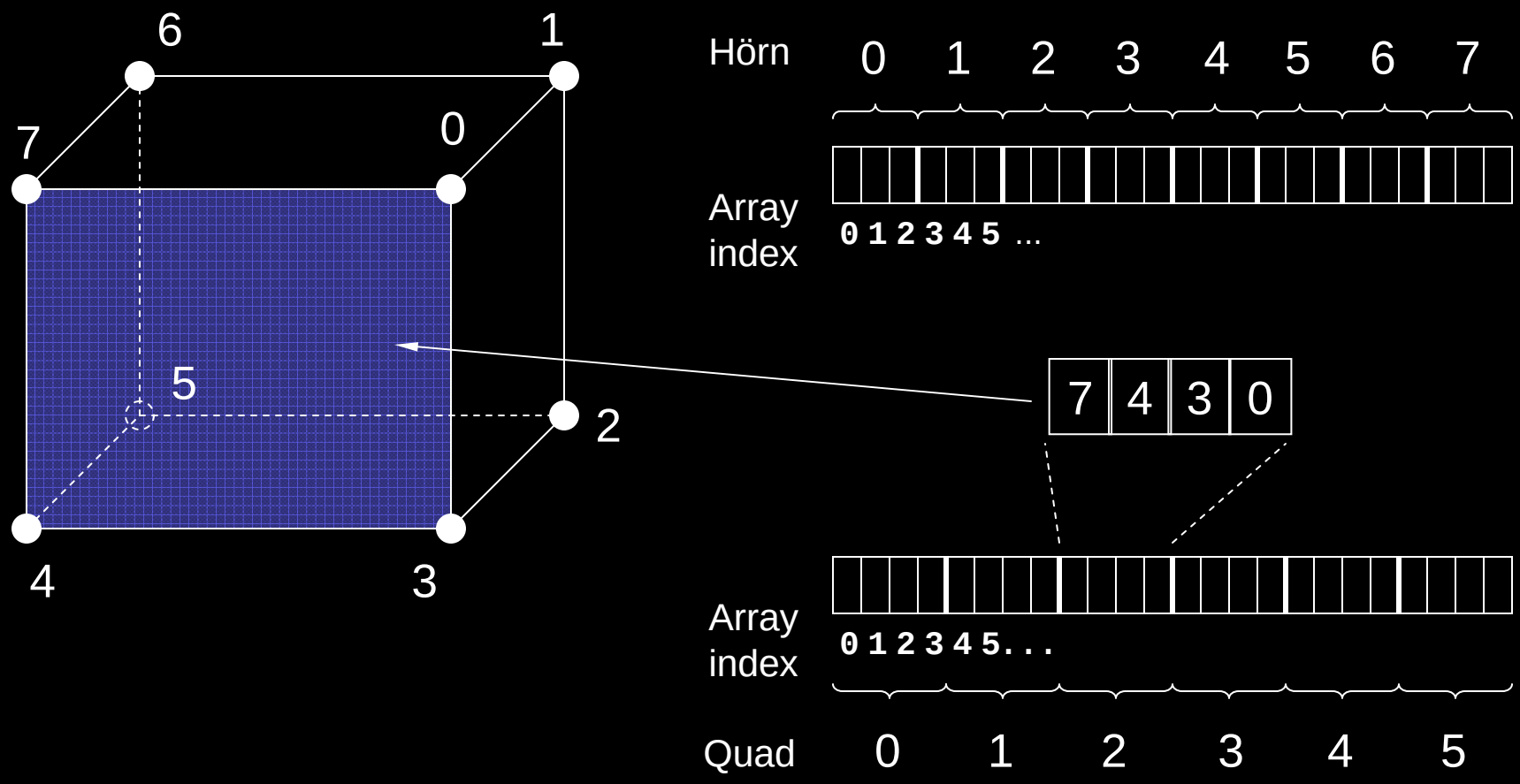
```
glBegin(GL_QUADS);
glVertex3f(...);
...
glEnd();
```

6 sidor x 4 hörn per sida: 24 hörn + 24 funktionsanrop

Vertex arrays

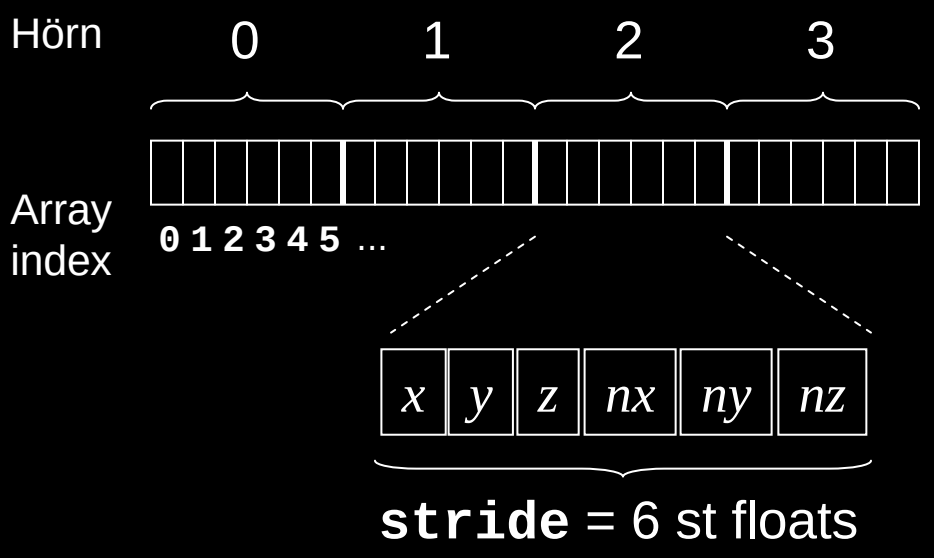


Indexerade primitiver

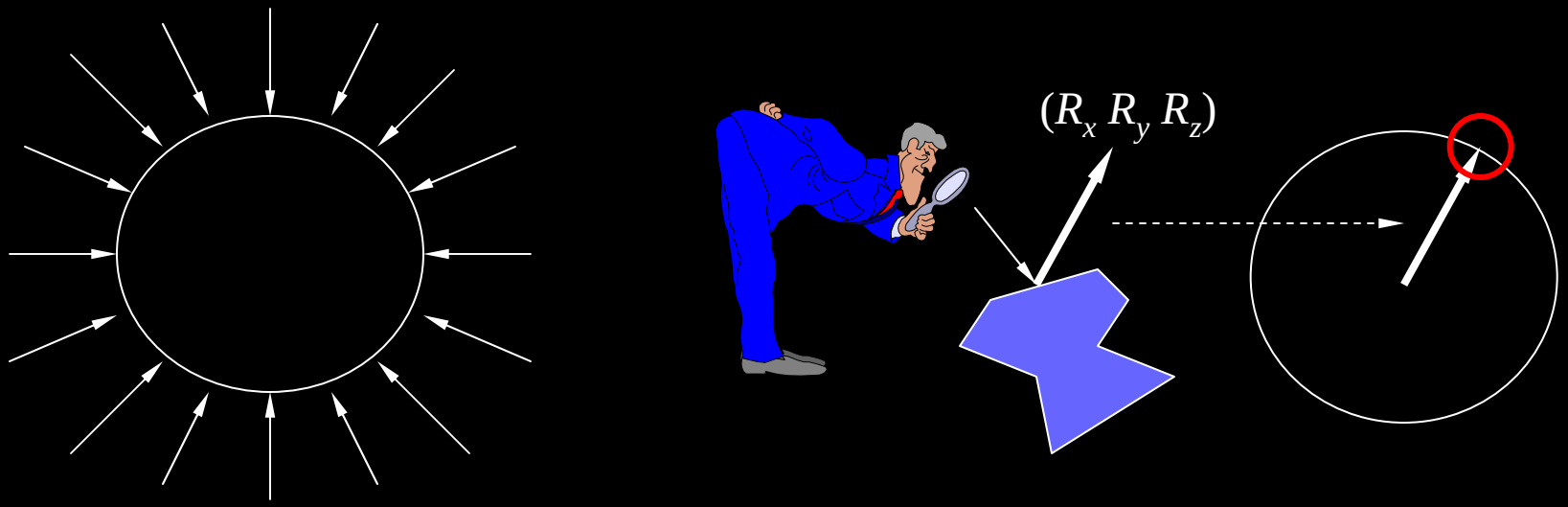


8 hörn, **2** funktionsanrop!

Interleaved vertex arrays

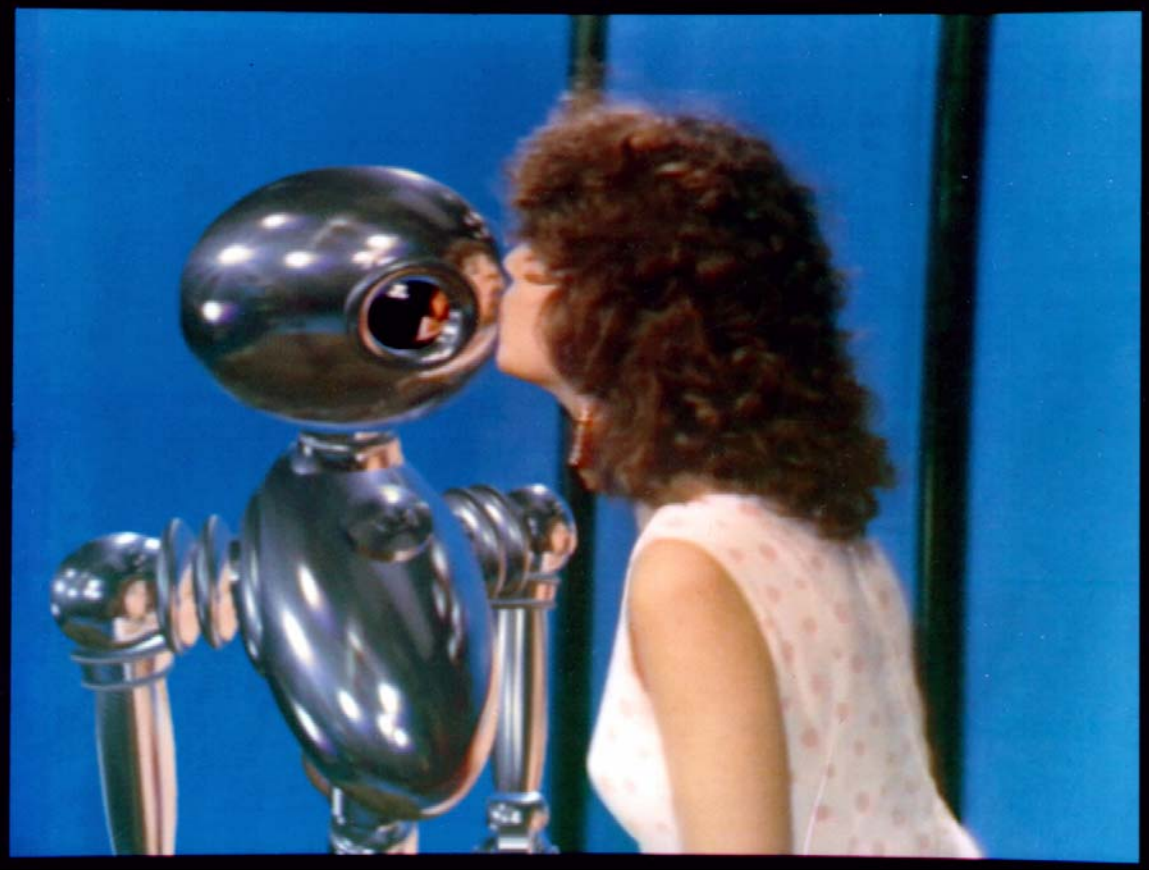


Reflection mapping



Jim Blinn, 1976

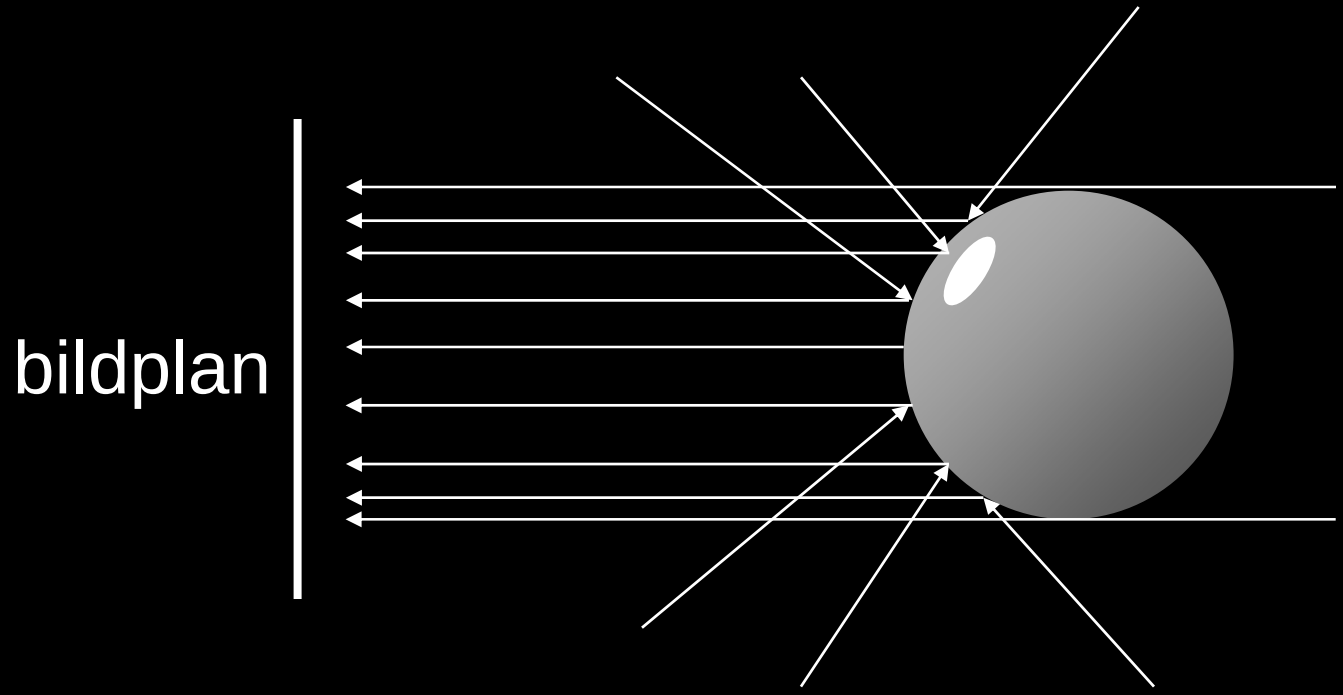
Gene Miller, Michael Shou m.fl. 1982-1983



Interface, Lance Williams, 1985

Terminator 2, 1991

Sphere mapping



Sphere mapping

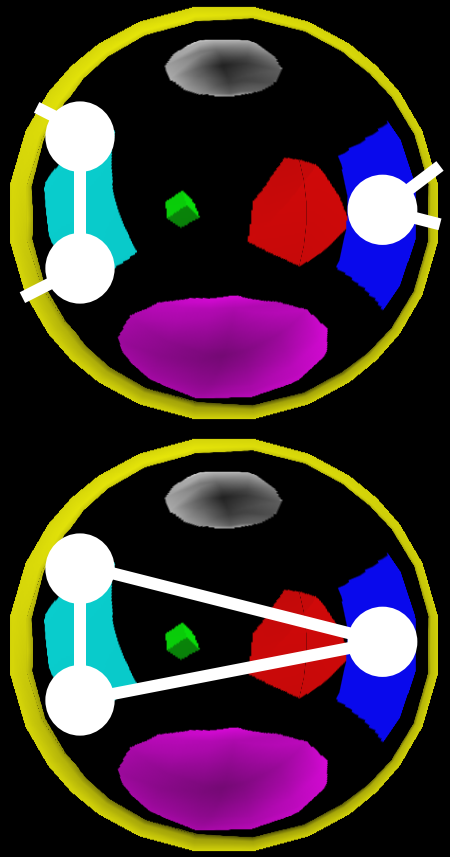
Från reflektionsvektor $\boldsymbol{R} = (R_x\ R_y\ R_z)$ till texturkoordinater (s, t):

$$s = \frac{R_x}{m} + \frac{1}{2}, \quad t = \frac{R_y}{m} + \frac{1}{2}$$

$$m = 2\left(R_x^2 + R_y^2 + \left(R_z + 1\right)^2\right)^{1/2}$$

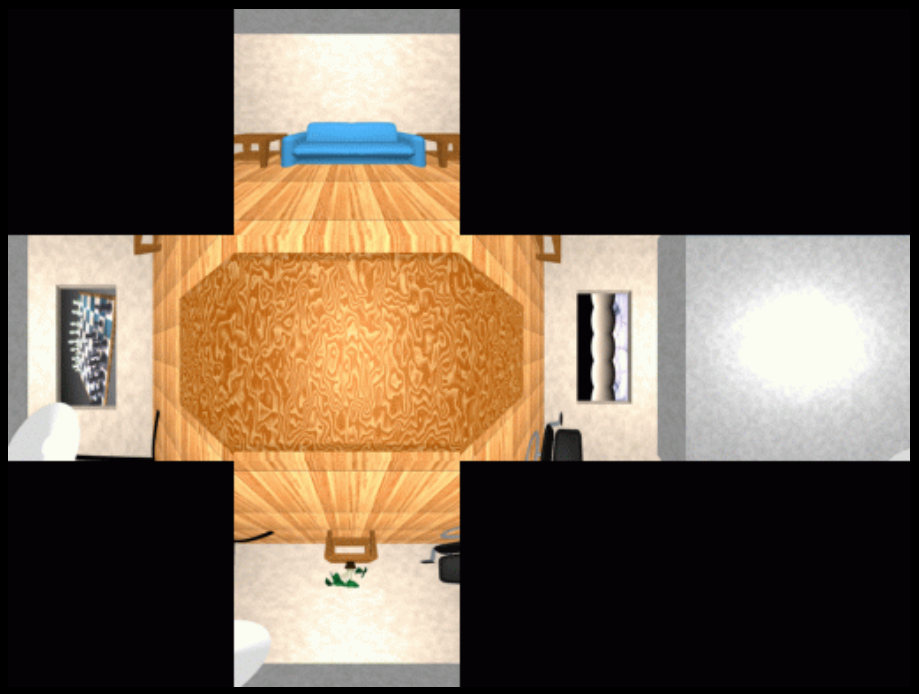
Sphere mapping - problem

Kraftigt varierande upplösning beroende på riktning.

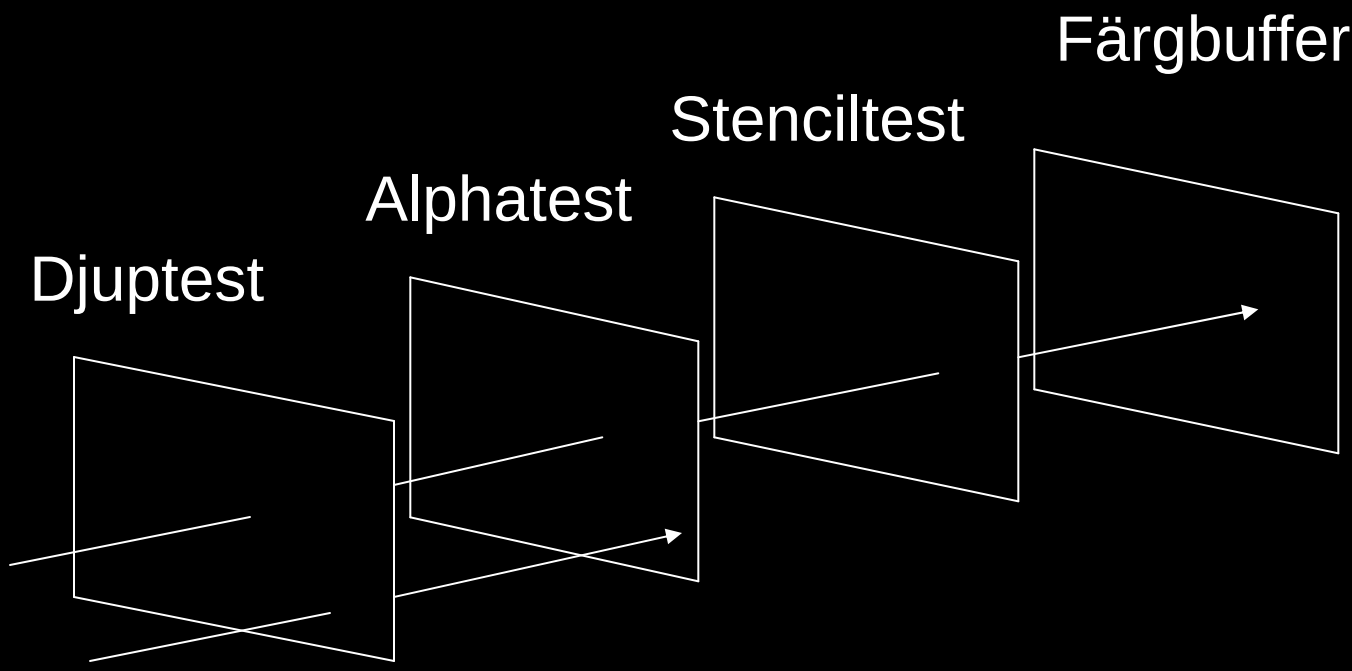


Problem med textur-interpolation för riktningar “bakåt”.

Cube mapping

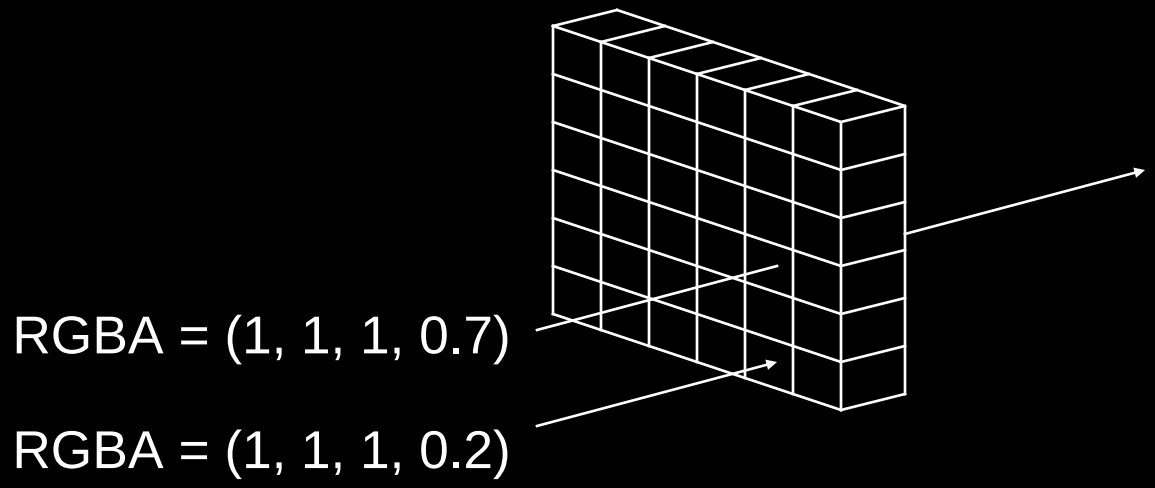


Buffertester i OpenGL

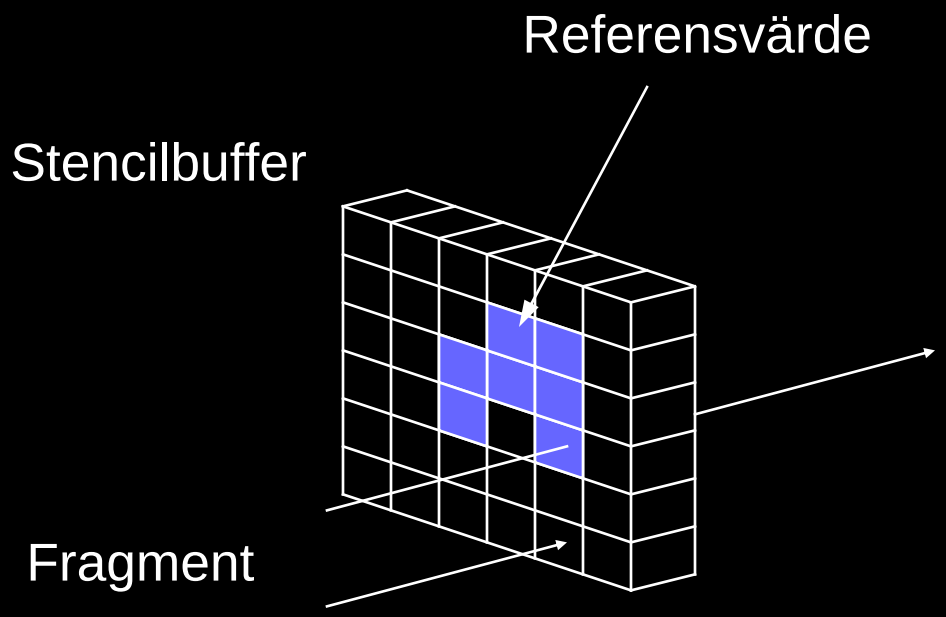


Alphatest

```
glAlphaFunc(GL_GREATER,0.5);
glEnable(GL_ALPHA_TEST);
```



Stenciltest



Stenciltest

- **`glStencilFunc()`** - konfigurerar stenciltestet
- **`glStencilOp()`** - bestämmer vad som ska hända med referensvärdet när testet utförs

```
glStencilFunc(GL_EQUAL, 1, ~0);
```

Stenciltestet ska returnera
TRUE om referensvärdet är lika med 1

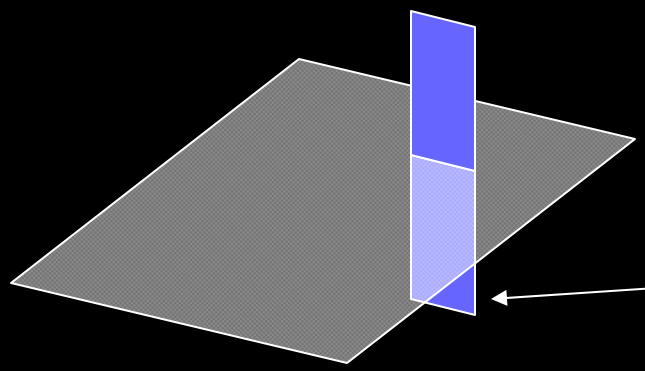
~0 är en *bitmask* och anger vilka
bitar i referensvärdet som ska beaktas
(**~0 = 0xFF** för 8 bitar)

```
glClear(GL_STENCIL_BUFFER_BIT);

glEnable(GL_STENCIL_TEST);
glStencilFunc(GL_ALWAYS, 1, ~0);
glStencilOp(GL_REPLACE, GL_REPLACE, GL_REPLACE);
glColorMask(GL_FALSE, GL_FALSE,
            GL_FALSE, GL_FALSE);

/* draw something here */

glColorMask(GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE);
glStencilFunc(GL_EQUAL, 1, ~0);
glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
glColorMask(GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE);

/* draw something else here */
```

Fyll i stencilbuffern utan att uppdatera färg-buffern

Rita scenen utan att uppdatera stencil-buffern

Stenciltest och labben



Denna del
måste klippas
bort...

Färgblandning

$$R_{out} = R_{in} \cdot S_R + R_{bildbuffert} \cdot D_R$$
$$G_{out} = G_{in} \cdot S_G + G_{bildbuffert} \cdot D_G$$
$$B_{out} = B_{in} \cdot S_B + B_{bildbuffert} \cdot D_B$$

där (S_R, D_R), (S_G, D_G) och (S_B, D_B) kallas *blandningsfaktorer*.

```
glBlendFunc(GL_SRC_ALPHA,
            GL_ONE_MINUS_SRC_ALPHA);
```

S_R ←

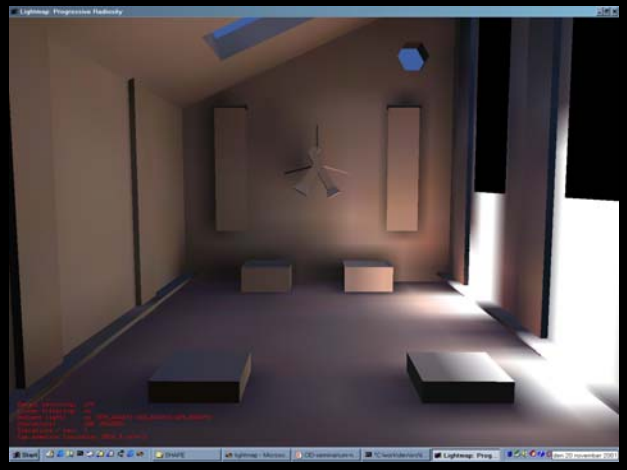
D_R ←

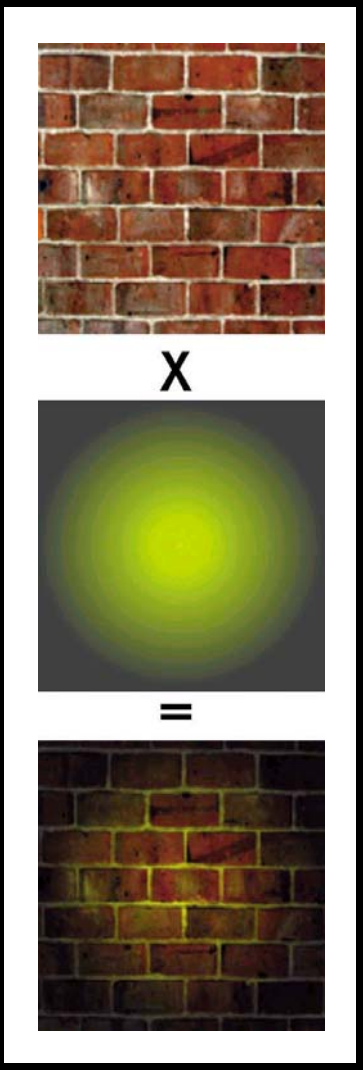
$$R_{out} = R_{in} \cdot A_{in} + R_{bildbuffert} \cdot (1 - A_{in})$$

Steg 1:
Rita gul triangel.

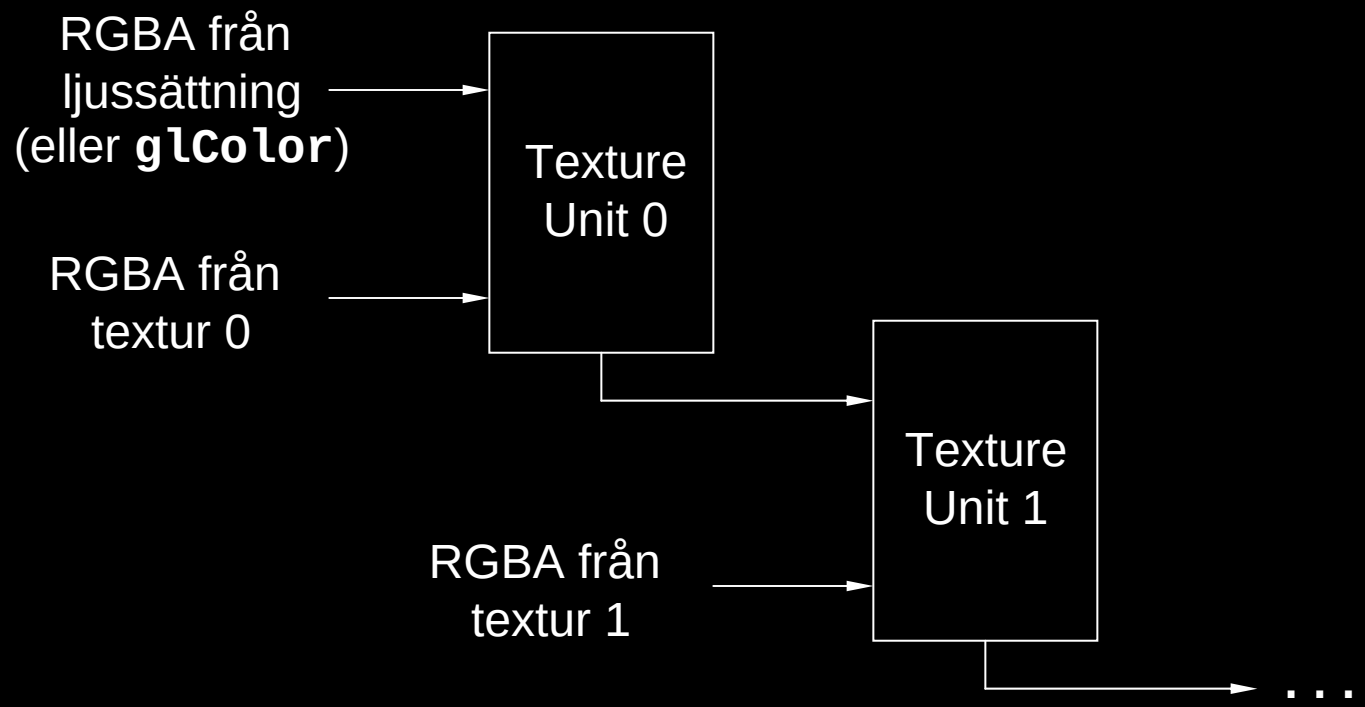
Steg 2:
`glEnable(GL_BLEND);`
Rita blå triangel
med A < 1.

Multitexturing





Multitexturing



```
#include <glew.h>



/* Aktivera enhet 0 och ladda textur */

glActiveTextureARB(GL_TEXTURE0_ARB);
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, ...);



/* Aktivera enhet 1 och ladda textur */

glActiveTextureARB(GL_TEXTURE1_ARB);
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D, ...);
```

```
glBegin(GL_TRIANGLES);


glMultiTexCoord2fARB(GL_TEXTURE0_ARB, 0.0, 0.0);
glMultiTexCoord2fARB(GL_TEXTURE1_ARB, 0.1, 0.2);
glVertex3f(0.0, 0.0, 0.0);


glMultiTexCoord2fARB(GL_TEXTURE0_ARB, 1.0, 0.0);
glMultiTexCoord2fARB(GL_TEXTURE1_ARB, 0.6, 0.2);
glVertex3f(1.0, 0.0, 0.0);


glMultiTexCoord2fARB(GL_TEXTURE0_ARB, 0.5, 1.0);
glMultiTexCoord2fARB(GL_TEXTURE1_ARB, 0.3, 0.5);
glVertex3f(0.5, 1.0, 0.0);


glEnd();
```

Debugging

```
if (glGetError() != GL_NO_ERROR)
{
   /* Något är fel! */
}
```

Labb - FAQ

- Maya lagrar absoluta sökvägar till texturfiler – ändra i RTG-filen om det behövs
- Se till att dina texturer är 2^n x 2^m