

# 3D Graphics Hardware

Hannes Kaufmann

Interactive Media Systems Group (IMS)  
Institute of Software Technology and  
Interactive Systems

Thanks to Dieter Schmalstieg and Michael Wimmer for providing some slides/images/diagrams



---

# Motivation

- Last week: Various VR Application Areas
  - VR/AR environment = Hardware setup + VR Software Framework + Application
- Detailed knowledge is needed about
- Hardware: **3D Graphics**, Input Devices & Tracking, Output Devices
  - Software: Standards, Toolkits, VR frameworks
  - Human Factors: Usability, Evaluations, Psychological Factors (Perception,...)

---

# 3D Graphics Hardware - Development

- Incredible development boost of consumer cards in previous ~10 years
- PC graphics surpassed workstations (~2001)
- Development driven by Game Industry

---

# Consumer Graphics – Major Points

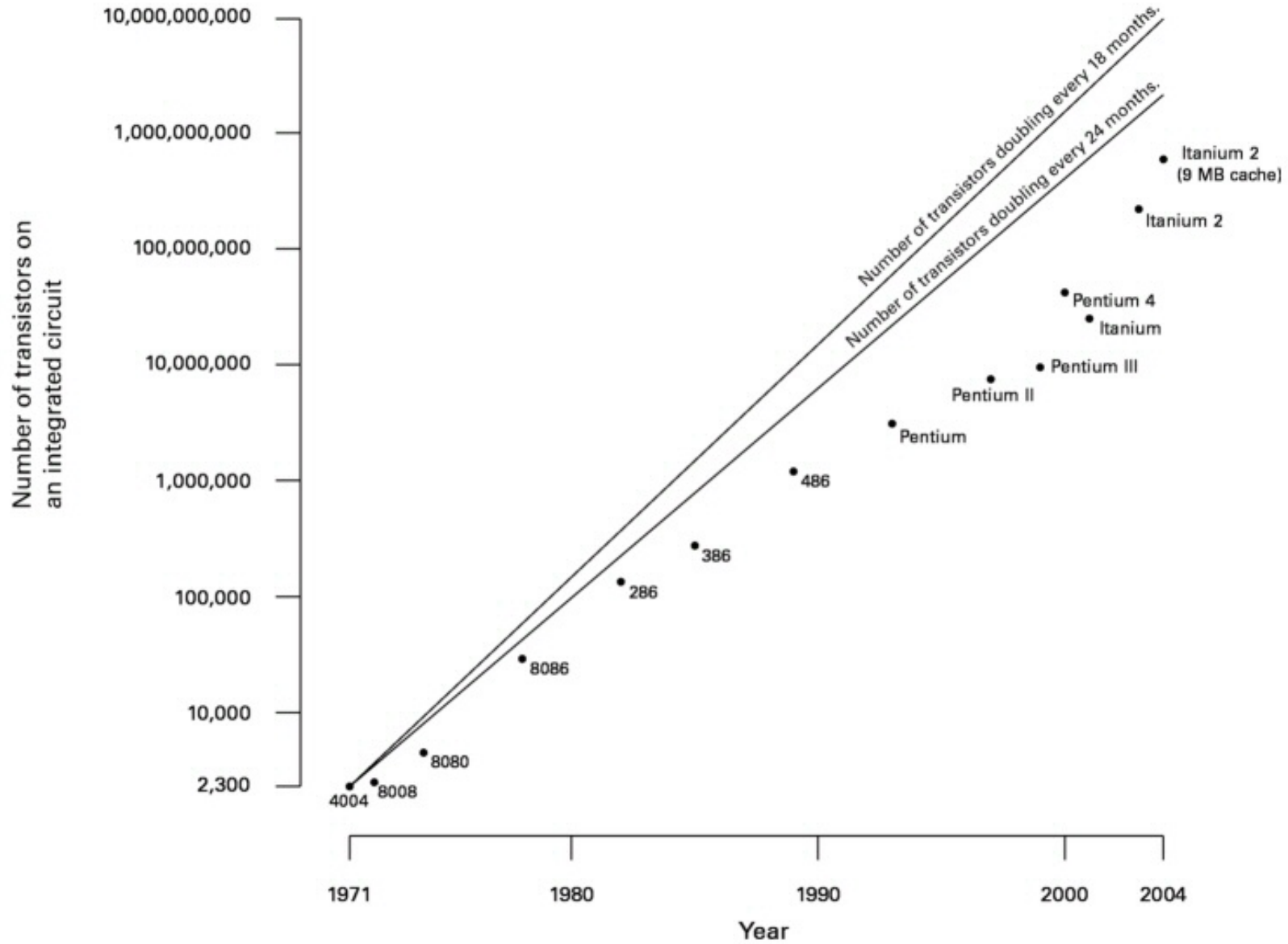
- Up to 1995
  - 2D only (S3, Cirrus Logic, Tseng Labs, Trident)
- 1995 Scanlines (Proprietary APIs)
- 1996 **3DFX Voodoo** (first real 3D card); Introduction of DX3
- 1997 Triangle rendering (... DX5)
- 1998 Triangle setup (...DX6)
- 1999 Multi-Pipe, Multitexture (...DX7)
- 2000 Transform and lighting (...DX8)
- 2001 Programmable shaders
  - **PCs surpass workstations**
- 2002 Full floating point
- 2004 Full looping and conditionals
- 2006/07 Geometry/Primitive shaders (DX10, OpenGL 3.0)

---

# Moore's Law

- Gordon Moore, Intel co-founder, 1965
- Exponential growth in number of transistors
- Doubles every 18 months (holds for CPUs)
  - yearly growth: factor 1.6
  - But: stagnation lately (2.8GHz available since December 2002);
  - Now: Dual Core / Quad Core CPUs

# Moore's Law



# SGI Development (Workstations)

Gen	Year	Product	Fill rate (no Z)	Yr rate	Tri rate (no Z)	Yr rate
1st	1984	Iris 2000	100K (46M)	-	0.8K (10K)	-
2nd	1988	GTX	40M (80M)	1.2	135K	1.9
3rd	1992	RealityEngine	380M	1.5	2M	2.0
		(Cost: 1,000,000\$, size of a domestic fridge!)				
3rd	1996	InfiniteReality	1000M	1.3	12M	1.6
				1.3		1.8

- Yearly tri rate growth above Moore's law!

# Nvidia Development 1/2

Season	Product	Proces	# Trans	MHz	32-bit AA <sup>1)</sup> Fill	MPolys	New Features
2H97	Riva 128	.35	3M	?	20M	3M	Integrated 2D/3D
1H98	Riva ZX	.25	5M	?	31M	3M	AGP 2x
2H98	Riva TNT	.25	7M	100	50M <sup>2)</sup>	6M	32-bit, Tri setup
1H99	TNT2 (Ultra)	.22	9M	150	75M	9M	AGP4x
2H99	GeForce256	.22	23M	120	120M <sup>3)</sup>	15M	HW T&L
1H00	GeForce 2 GTS	.18	25M	200	200M <sup>4)</sup>	25M	Per-Pixel Shading
2H00	Geforce 2 Ultra	.18	25M	250	250M	31M	
1H01	GeForce 3	.15	57M	200	417M <sup>5)</sup>	25M	Vertex Shading
2H01	GeForce 3 Ti500	.15	57M	240	500M	30M	
1H02	GeForce 4	.15	63M	300	625M	75M <sup>6)</sup>	(Dual display)
1H03	GeForce FX 5800	.13	125M	500	1041M	375M	Floating Point
2H03	GeForce FX 5900	.13	130M	450	938M	338M	(faster shading)
2H04	GeForce FX 6800	.13	222M	400	~2000M	600M	looping

# Nvidia Development 2/2

Season	Product	32-bit AA Fill	Yr rate	MPolys	Yr rate
2H97	Riva 128	20M	-	3M	-
1H98	Riva ZX	31M	2.4	3M	1.0
2H98	Riva TNT	50M	2.6	6M	4.0
1H99	TNT 2	75M	2.3	9M	2.3
2H99	GeForce256	120M	2.6	15M	2.8
1H00	GeForce 2 GTS	200M	2.6	25M	2.8
2H00	Geforce 2 Ultra	250M	1.6	31M	1.5
1H01	GeForce 3	416M	2.5	25M	0.6
2H01	GeForce 3 Ti500	500M	1.4	30M	1.4
1H02	GeForce 4	625M	1.6	75M	6.3
1H03	GeForceFX 5800	1041M	1.7	375M	5
2H03	GeForceFX 5900	938M	0.8	338M	0.8
2H04	GeForceFX 6800	~2000M	2.1	600M	1.8
	(Cost: 500 Euro)		2.1		2.5

2H05 GeForce 7800

860M

Almost Moore's law squared ( $\wedge 1.5-2.0$ )

Performance doubles every 9-12 months!

# Mobile Graphics

	GeForce Go 6800 Ultra (10 months ago)	GeForce Go 7800 GTX (Today)
<b>Shader Performance (Pixels per Clock)</b>	1X	2X
<b>Geometry Performance</b>	1X	1.6X
<b>Transistors</b>	202 Million	302 Million
<b>Power Consumption</b>	1X	1X
<b>Package</b>	Identical	Identical
<b>Memory Interface</b>	256-Bit	256-Bit
<b>Clock Frequencies (Core/Memory)</b>	450MHz/550MHz	400MHz/550MHz
<b>DirectX 9.0</b>	Shader Model 3.0	Shader Model 3.0
<b>Transparency AA</b>	No	Yes
<b>Programmable Video Processor</b>	PureVideo	PureVideo
<b>Power Management</b>	PowerMizer 5.0	PowerMizer 6.0
<b>Host Interface</b>	PCI-Express	PCI-Express

Current model:  
GeForce Go 7950 GTX

Complies to Moore's Law too  
Similar performance increase

# And it goes on and on....

- Performance increase expected to continue within the next few years
  - Smaller chip production processes possible (currently 90nm)
  - More pixel pipelines
  - Multi-core GPUs
  - Multiple graphics cards or GPUs in a PC
- General purpose computing on GPUs
  - [www.gpgpu.org](http://www.gpgpu.org)



---

# What are the benefits in VR/AR?

---

# 3D Card High End Model

- nVidia Quadro FX 5500 (~ € 2900.- ), 1GB
- Based on G71 chip (such as GeForce 7900)
- 24 Pixel Pipelines, 8 Vertex Pipelines
- 33.6 GB/s Bandwidth
- 10.32 billion pix/sec.
- Vertices: 860 M/s
- optimized OpenGL drivers (comp. to consumer card)



---

# Some Relevant Features (for VR)

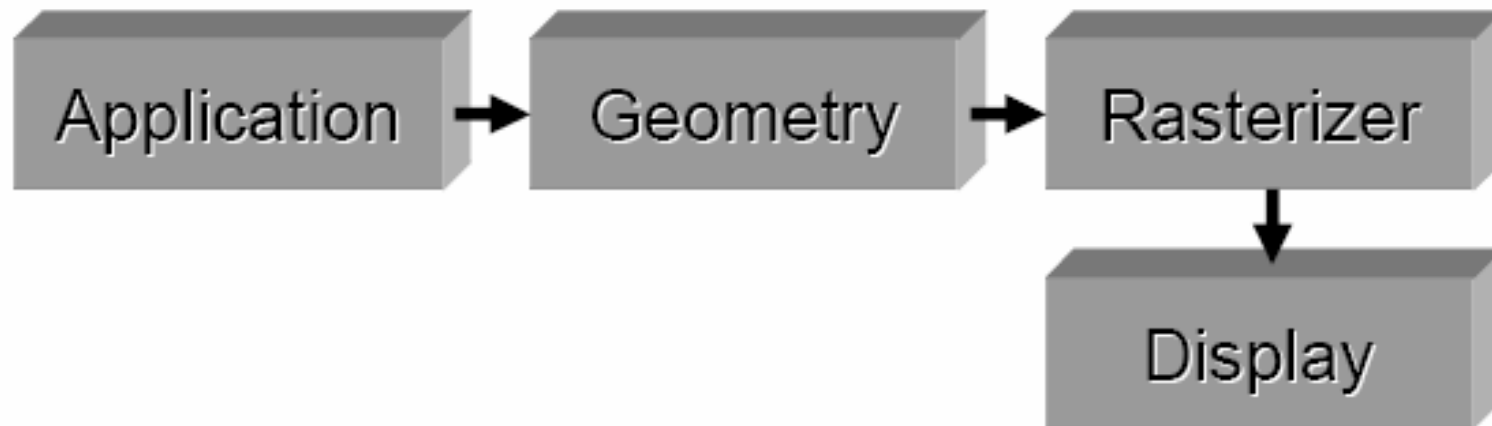
- Memory size: 1 GB
- Dual DVI / VGA Output (up to 40x2400), Dual 400Mhz RAMDACs
- Multi-Display support
- OpenGL quad-buffered stereo (3-pin sync connector)
- Scalable Link Interface (SLI™) Technology
- Edge Blending for Powerwalls (supported by driver)
- G-Sync Option Board with Framelock and Genlock
- Quality: Full-Scene Antialiasing (FSAA); OpenGL 2.0; Shader Model 3.0,...

---

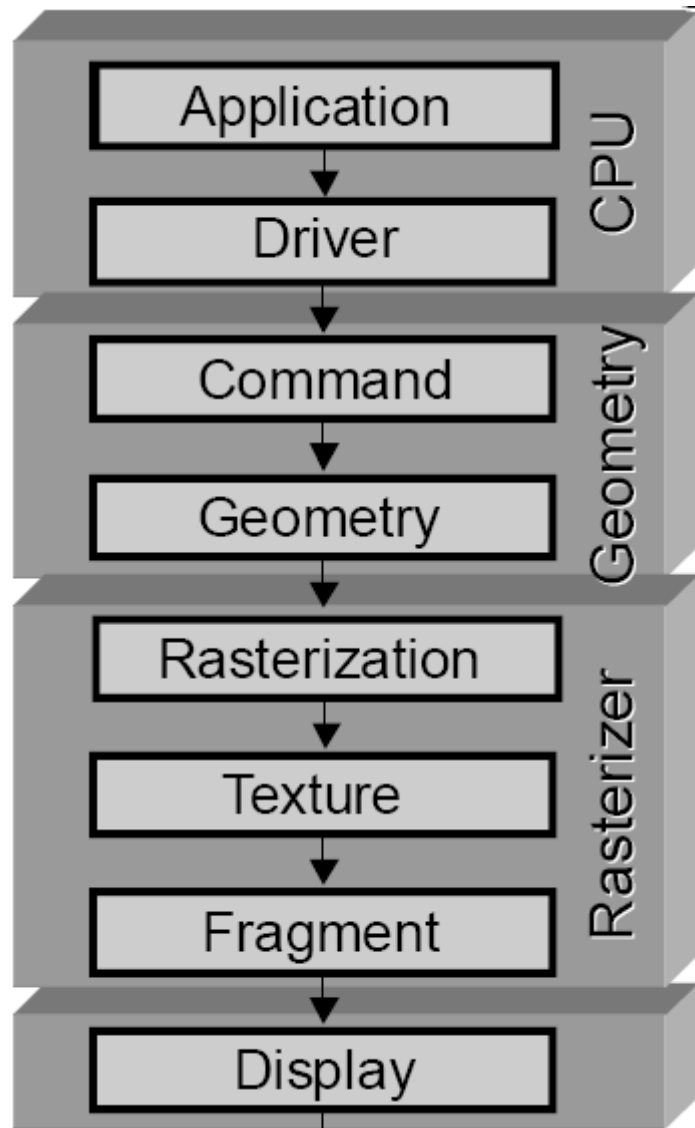
# Explanations & Back to the Basics

# 3D Graphics Basics

## The Graphics Pipeline



# „Old“ Fixed Graphics Pipeline



Purpose: Convert Scene to Pixel Data  
Fixed processing of scene

Geometry Stage:

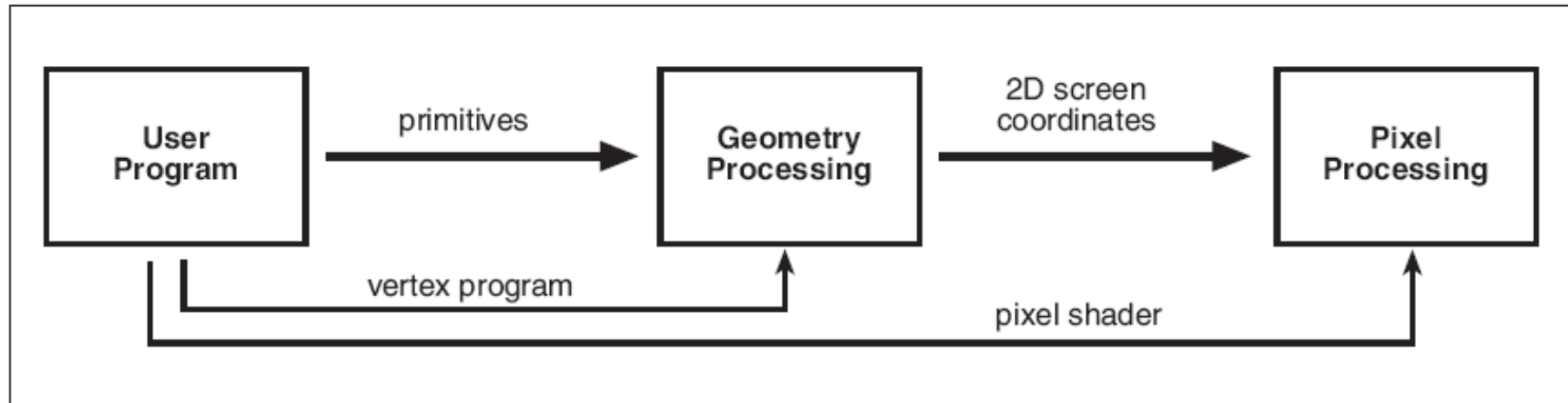
- Input: Primitives
- Output: 2D window coordinates

Rasterization Stage:

- Input: 2D window coordinates
- Output: Pixels

- Nowadays Geometry and Rasterizer Stage completely HW accelerated

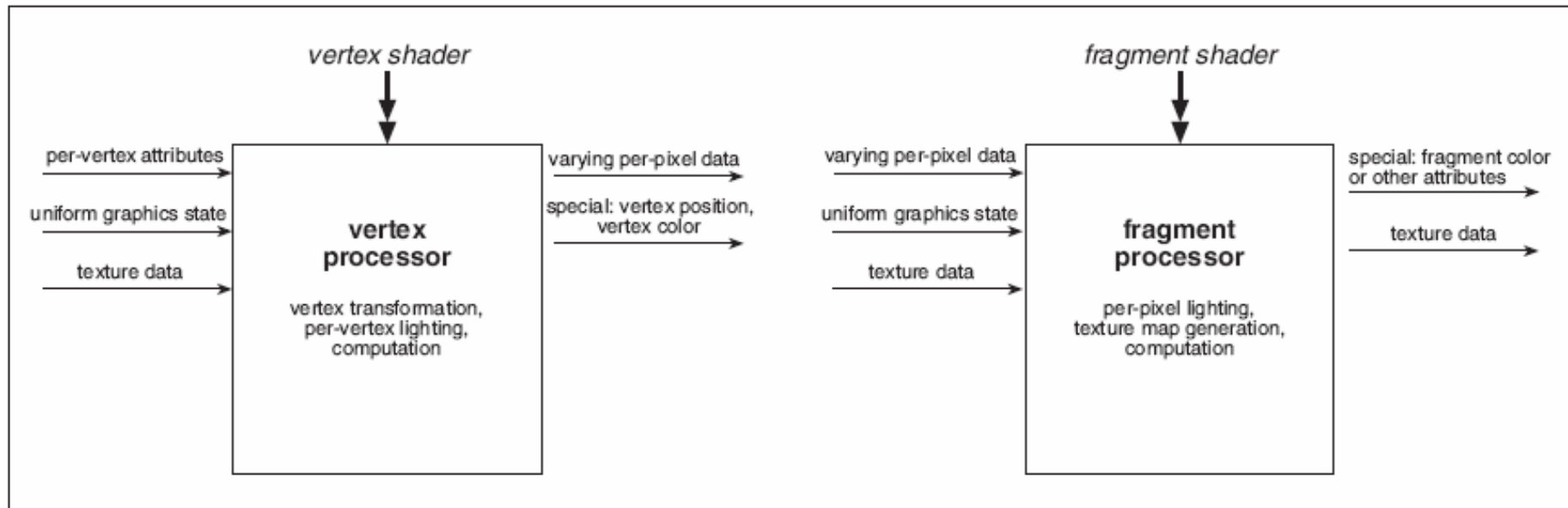
# Modern Programmable Graphics Pipeline



**Figure 17.5.** The programmable graphics hardware pipeline. The user program supplies primitives, vertex programs, and fragment programs to the hardware.

- Vertex Shader before „old“ Geometry Stage
  - Allows per vertex transformations e.g. warping
- Fragment/Pixel Shader integrated in „old“ Rasterization Stage
  - Fragment: „pixel“ with additional information (alpha, depth, stencil,...)
  - Allows e.g. per pixel lightning,.....

# Vertex and Fragment Shaders



**Figure 17.6.** The execution model for shader programs. Input, such as per-vertex attributes, graphics state-related uniform variables, varying data, and texture maps are provided to vertex and fragment programs within the shader processor. Shaders output special variables used in later parts of the graphics pipeline.

- **Various Shading Languages**
  - GLSL (Open GL Shading Language – in OpenGL 2.0)
  - HLSL (High Level Shading Language – Microsoft)
  - CG (NVIDIA)
  - Assembler (optimized)

---

# Not yet in Hardware

- Create vertices (tessellation/triangulation)
- Evaluation of polynomials for curved surfaces

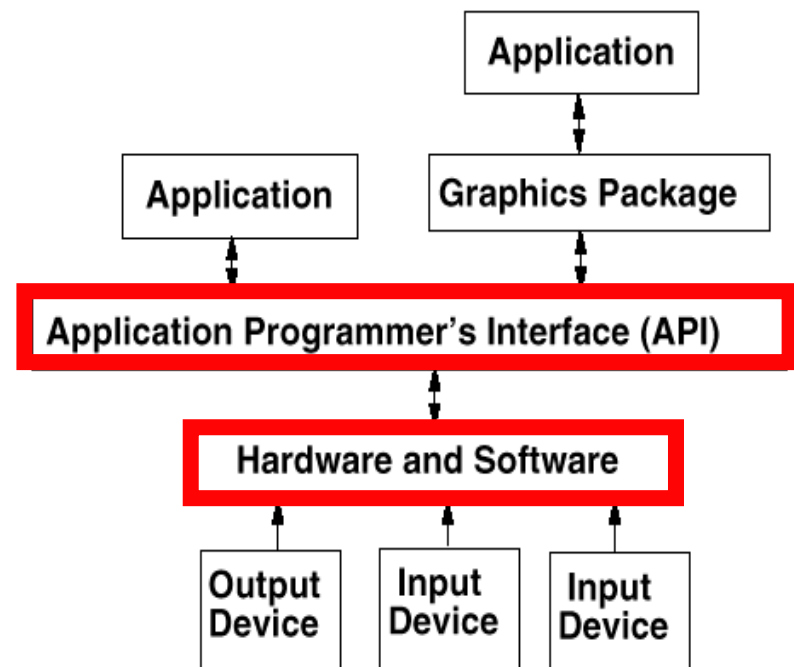
## → Solution: Geometry Shaders

- Integrated in DirectX 10 (Nov. 2006)
- Will be included in OpenGL 3.0 (Summer 2007)
- First graphics card with Geometry Shader support by NVidia in Nov. 2006
- High Level Object Data can be sent to HW

# (1) Application Stage: 3D Graphics Programming

## 3D Application Programmer's Interfaces (APIs)

- Access to Hardware
- Standards:
  - OpenGL, Direct3D
- Language: C (mostly)
- Higher Level APIs based on OpenGL, Direct3D
  - Game Engines
  - Scene Graph APIs:
    - OpenInventor, Java3D
    - Performer,...



# OpenGL – Hello World

```
#include <GL/glut.h>

void display(void) {
    glClear (GL_COLOR_BUFFER_BIT);
    /* draw white polygon (rectangle) with
    corners at (0.25, 0.25, 0.0)
    and (0.75, 0.75, 0.0) */
    glColor3f (1.0, 1.0, 1.0);
    glBegin(GL_POLYGON);
    glVertex3f (0.25, 0.25, 0.0);
    glVertex3f (0.75, 0.25, 0.0);
    glVertex3f (0.75, 0.75, 0.0);
    glVertex3f (0.25, 0.75, 0.0);
    glEnd();
    glFlush ();
}

void init (void) {
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE
    GLUT_RGB);
    glutInitWindowSize (250, 250);
    glutInitWindowPosition (100, 100);
    glutCreateWindow ("hello");
    init ();
    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```

# OpenGL Geometric Primitives




*All geometric primitives are specified by vertices*

  
GL\_POINTS

  
GL\_LINES

  
GL\_LINE\_STRIP


  
GL\_LINE\_LOOP

  
GL\_POLYGON

  
GL\_TRIANGLES

  
GL\_TRIANGLE\_STRIP

  
GL\_TRIANGLE\_FAN

  
GL\_QUADS

  
GL\_QUAD\_STRIP

---

# (1) Application Stage

- Generate geometric primitives, camera properties, lightning, object properties (color, texture,...)
- Input event handling
- Optimizations possible: Build hierarchy, Level of Details, Culling Techniques, Impostors, Triangulation,...

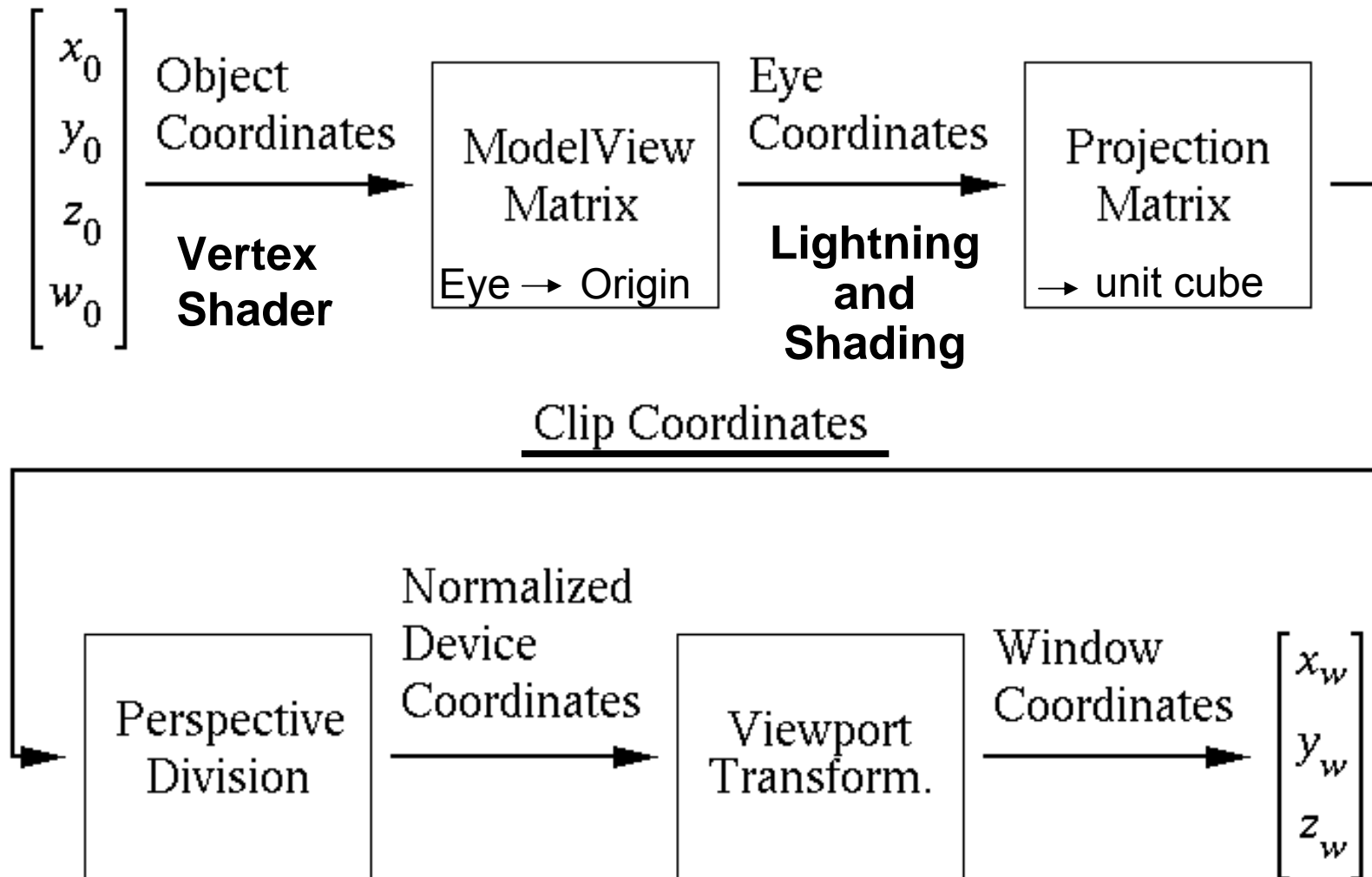
---

# Graphics Driver

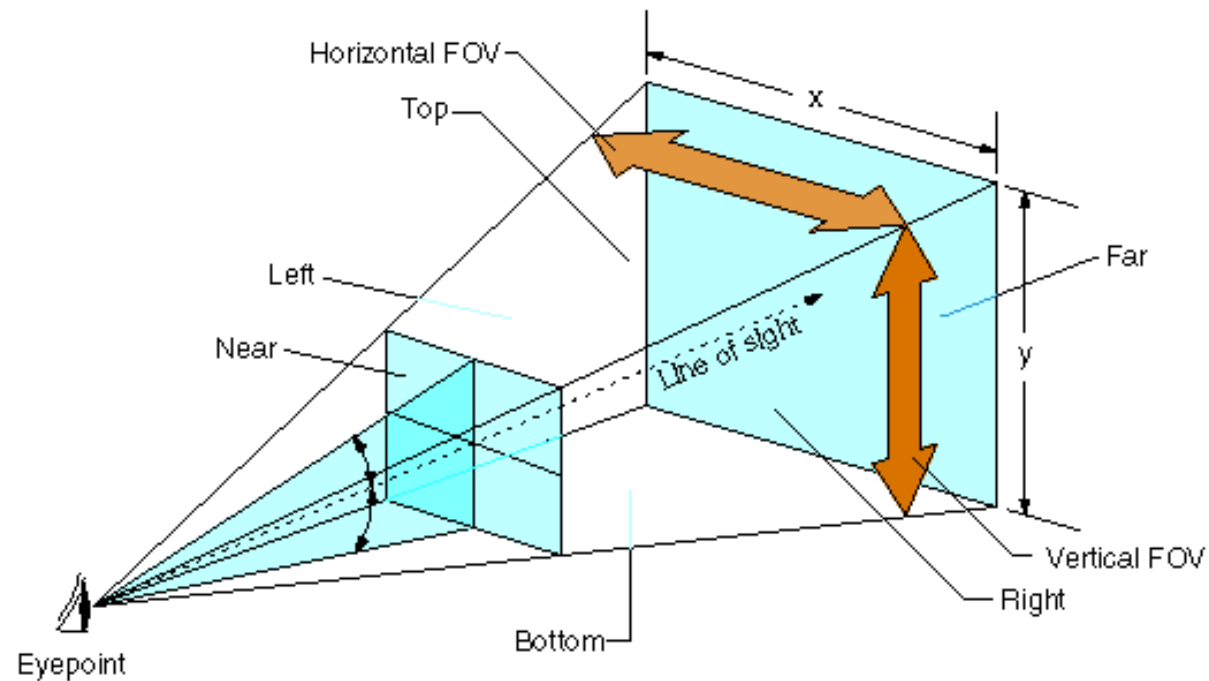
- Command interpretation/translation
  - Host commands      GPU commands
- Handle data transfer
- Memory management
- Emulation of missing features (e.g. full OpenGL 2.0 support)



## (2) The Geometry Stage (Simple)



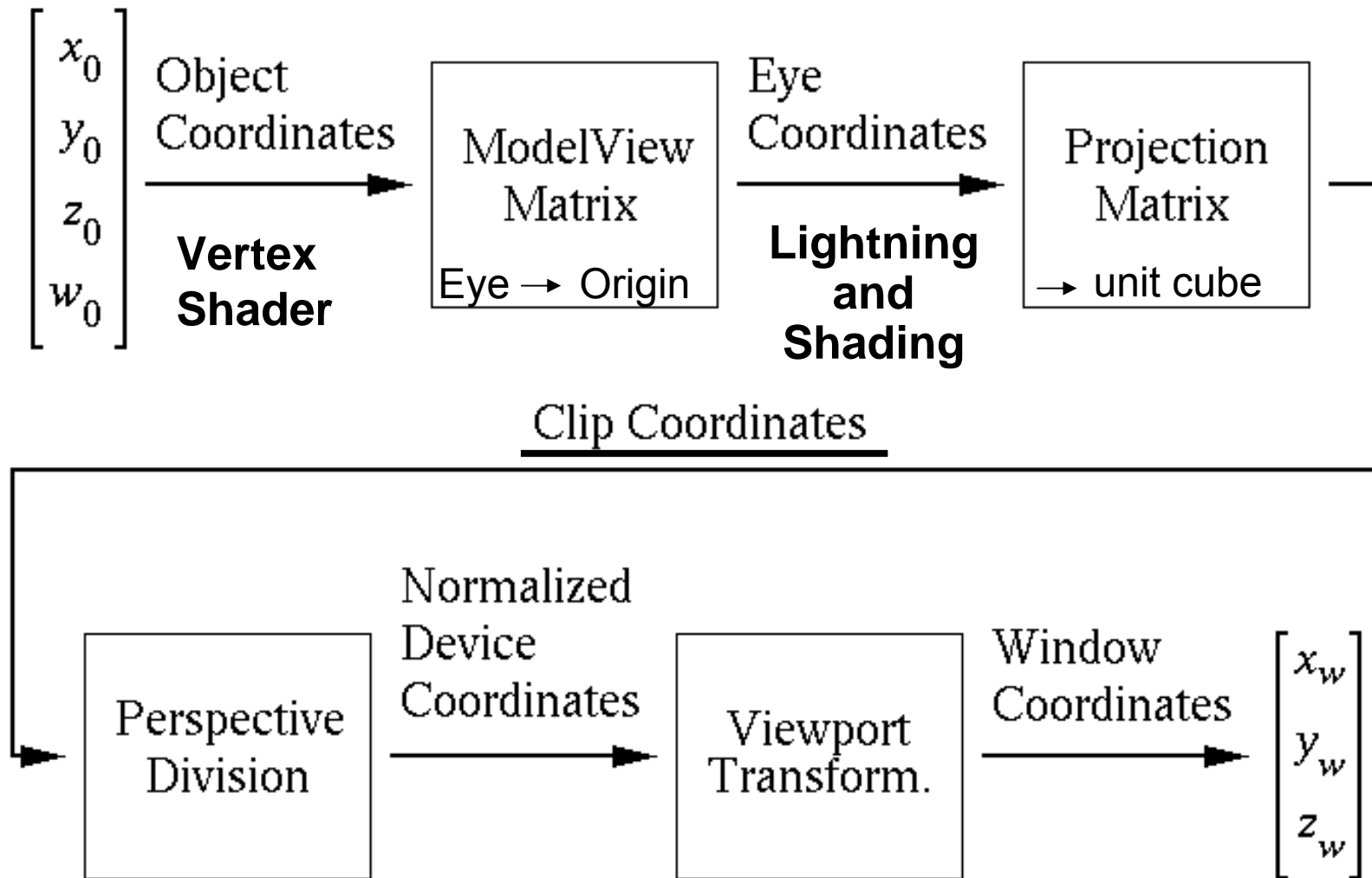
# Viewing Frustum



$$\text{Aspect Ratio} = \frac{y}{x} = \frac{\tan(\text{vertical FOV}/2)}{\tan(\text{horizontal FOV}/2)}$$

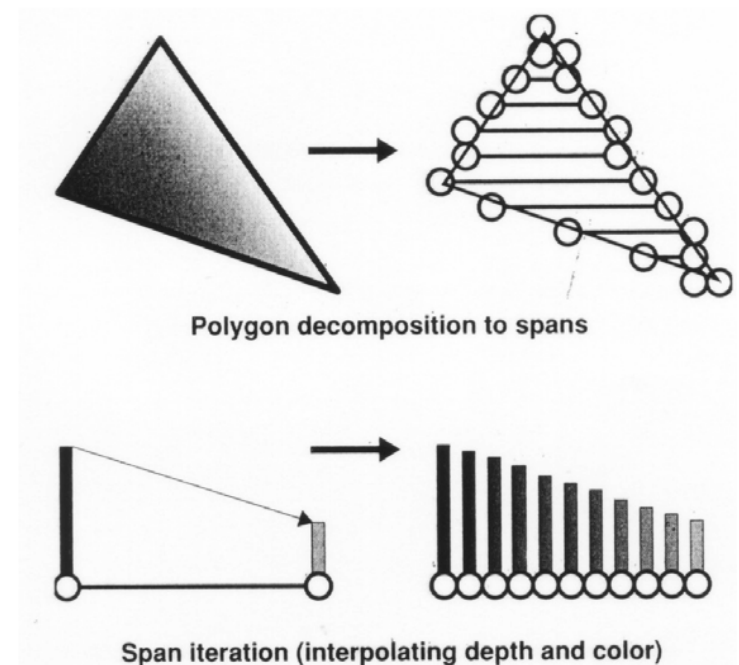
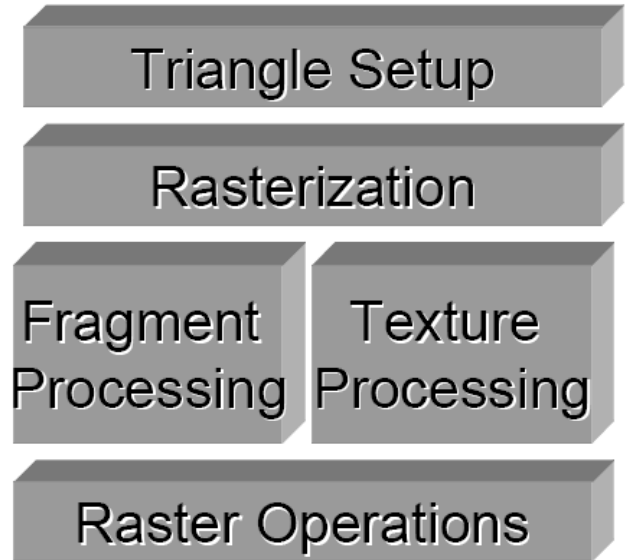


## (2) The Geometry Stage (Simple)



# (3) Rasterization Stage

- Input: 2D Geometric Primitives (Points, Lines, Polys, Bitmaps)
- **Primitives** needed!
- 1st step output: **Fragments** (Pixel-Coord. + Color + Depth + Texture-Coord.)
- Polygons are decomposed (various methods)



---

## (3) Rasterization Stage

- Per-Fragment Operations
- Pixel Ownership Test (Window visible?)

### Buffers:

- Frame Buffer (Color + Alpha channel)
- Depth Buffer Test (z-Buffer)
- Stencil Buffer
- Accumulation Buffer

---

# Rasterizer/Display Stage

- Framebuffer pixel format: RGBA vs. indexed (colormap)
- Bits: 32, 24 (true color) 16, 15 (high color), 8
- Double buffering, Triple Buffering
- For Stereo: **Quad buffer**
- Per-window video mode (e.g. stereo, mono)
- **Display:** frame buffer -> screen

---

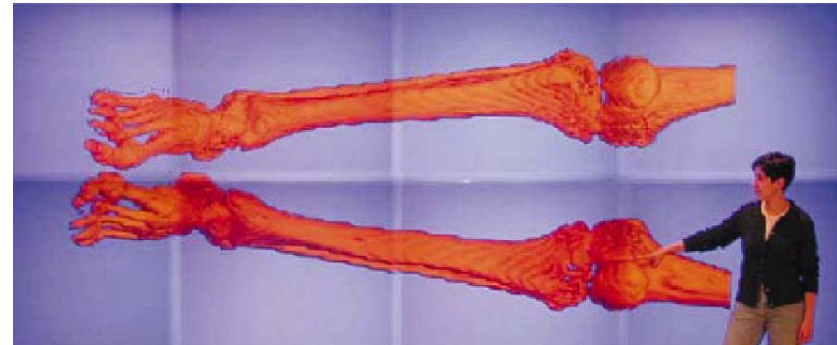
# Bottlenecks

- **Application** limited: App cannot create drawing primitives fast enough
- **Interface** limited: Not enough bandwidth CPU->graphics card (e.g. AGP)
- **Geometry** limited: Not enough vertex transform capacity (Test: remove lights)
- **Pixel** fill limited: Not enough pixel fill capacity (Test: decrease resolution)
- **Memory** limited: Not enough (texture) memory bandwidth

# VR/AR and the Need for Extreme Graphics Power: Examples



Mechanical visualization  
CAVE, SGI Onyx  
(8 CPUs, 6 outputs)



Princeton Display Wall  
3x8 projectors, 24 PC cluster



HMD setups for  
larger groups

---

# Parallel Graphics Hardware

Overcome bottleneck by parallel computation

Types of parallel graphics:

1. On-chip / on a graphics board (soon standard)
2. Multiple boards (former: graphics supercomputer)
3. Multiple boards with multi GPUs (1+2)
4. PC „cluster“ with standard network –  
Distributed Environment
5. PC cluster with special hardware

---

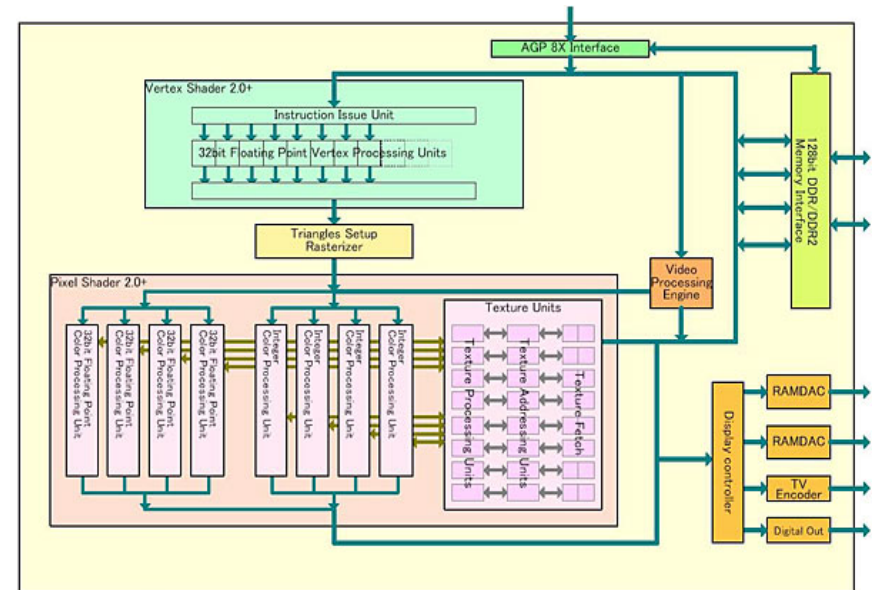
# Parallel Graphics Hardware

- (A) Computing the same (high resolution) image
- (B) Computing multiple images –  
Multiple outputs



# Multiple Graphic Pipelines

- Pipelines fully in HW
- Multiple independant pipelines can be parallelized

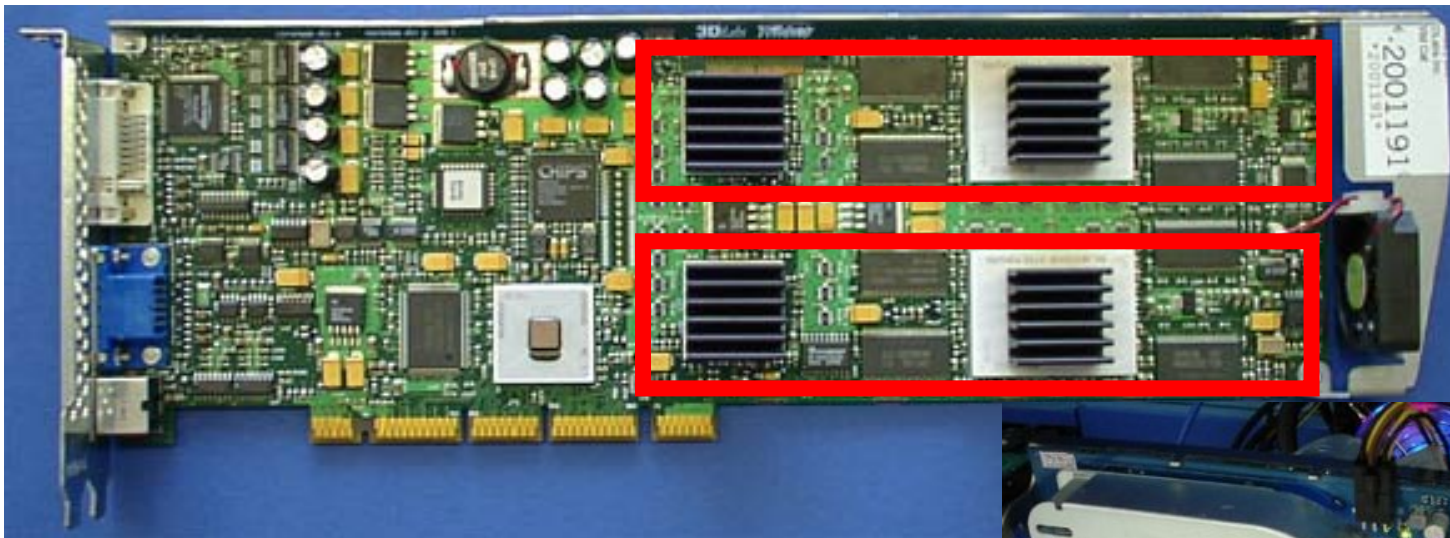


- Modern GPUs process up to 24 Pixel-Pipelines (Rasterizer) and 8 Vertex-Pipelines (Geometry) in parallel

NVidia  
NV30

# Parallel On-Board

- Examples:
- 3DLabs Wildcat III: 2 Pipes



- GeForce 7950 GX2  
(Sept. 06)
  - 48 pixels/clock



---

# Parallel Graphics Hardware

Types of parallel graphics:

1. On-chip / on a graphics board
2. **Multiple boards (former: graphics supercomputer)**
3. Multiple boards with multi GPUs (1+2)
4. PC cluster with standard network
5. PC cluster with special hardware

# Multiple Graphics Boards

Parallel graphics rendering:

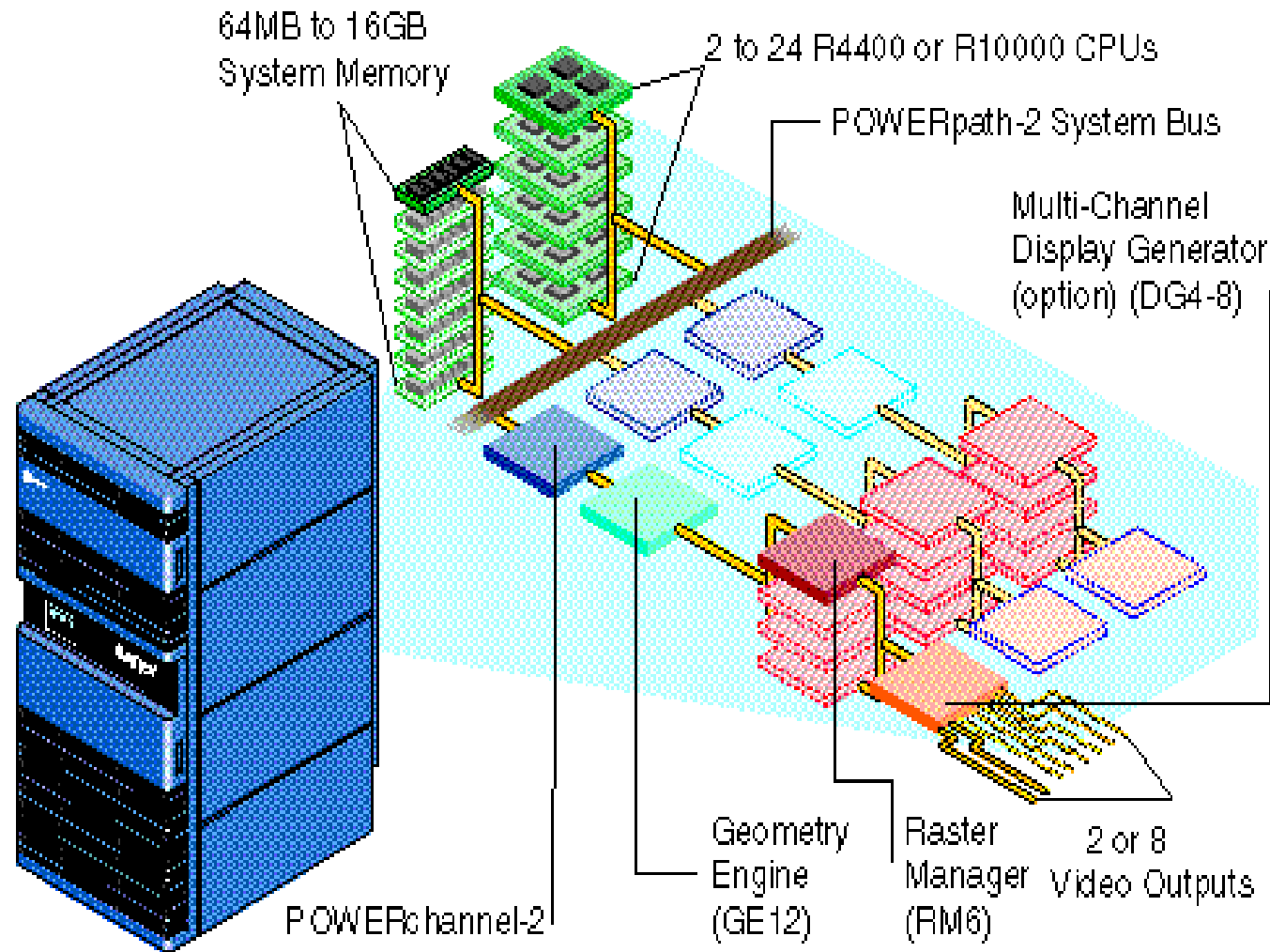
- Graphics Supercomputer
- PC with SLI or CrossFire

Different:

Multiple display support (not synchronized):

- PC with multiple unconnected cards (via PCIe or APG+PCI)

# Graphics Supercomputer



SGI Onyx with  
Infinite Reality 3

# SGI Onyx 3000 & Infinite Reality 4

## G-Brick:

- 4 RasterManager Boards
- 1.3 Gpixel/s/Pipeline
  - (8 subsample/full scene/AA)
- 1 GB Texturspeicher
- 10 GB Framebuffer
- 192 GB/s Bandbreite
- Kombination bis zu 16 IR4





---

# Nvidia Quadro Plex (2006)



- Connected via special PCIe cable to PC
- 2-4 GPUs
- 1GB Frame Buffer per GPU
- 2 to 8 Dual-Link Digital Display Connectors
- Genlock/Frame Lock
- Price: 17.5K – 24.5K USD

# Supercomputer – Application Areas

- Theme Parks  
(DisneyQuest –  
CyberSpace Mountain)
- Flight Simulators
- Military Applications
- CAVEs / Large setups



# Graphics Supercomputer

- Multiple CPUs
- Multiple Geometry Engines
- Multiple Rasterization Engines
- Genlocking
- Multiple Pipes (=graphics cards)
- Multiple Channels (=display outputs)
- Highly configurable
- Now used: standard Nvidia/ATI graphics chips
- Now on PC: Scalable Link Interface (Nvidia) or CrossFire (ATI) for PCI Express

---

# Parallel Graphics Hardware

(A) Computing the same (high resolution) image

(B) Computing multiple images –  
Multiple outputs

---

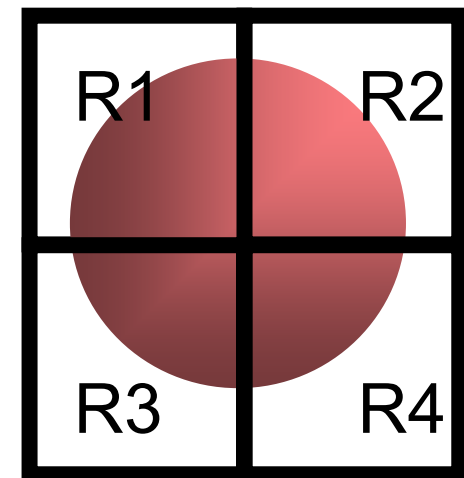
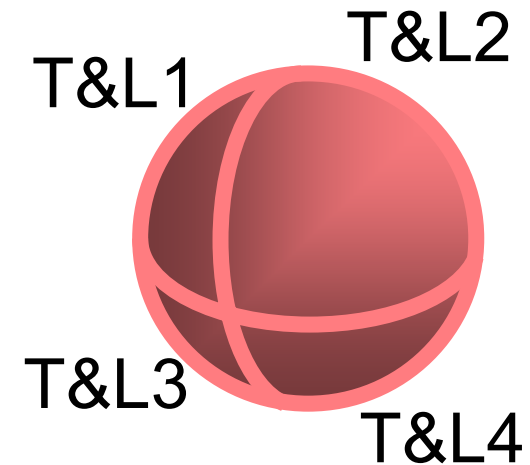
# Basic Problems of Parallel Rendering

## Vertex and Pixel Load Balancing:

- Problem with parallel rendering
  - Load balancing of vertices
    - 3D (object space) problem
  - Load balancing of pixel (rasterizers)
    - 2D (screen space) problem

# Parallel Rendering as Sorting (1)

- Parallel Geometry Stage
    - Cut 3D model into pieces with equal number of vertices
    - Assign one piece to one T&L unit
  - Parallel Rasterization
    - Cut destination image into tiles
    - Assign (triangles contained in) one tile to one rasterizer
- Need to SORT transformed 2D triangles
- Shared common memory

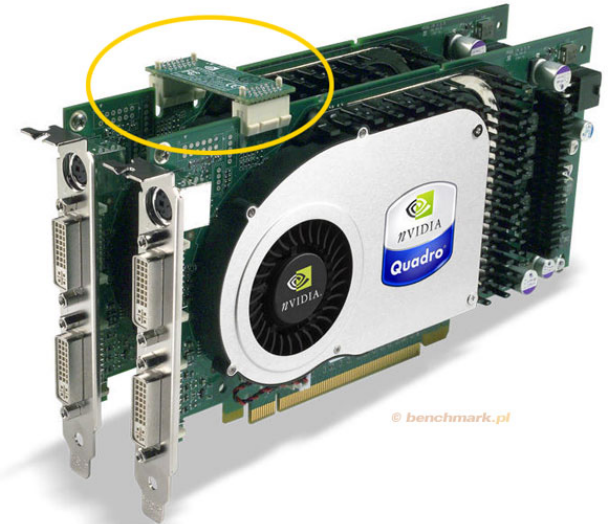


# SLI™ (Nvidia)

- originally Scan Line Interleave (3Dfx Voodoo 2) – odd/even lines
- Nvidia calls it **Scalable Link Interface**

## 2 Modes:

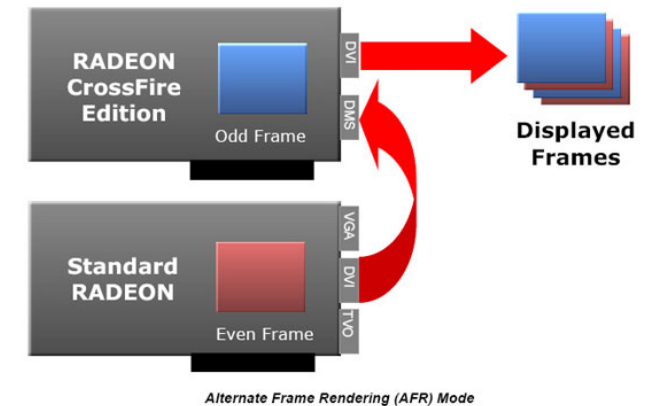
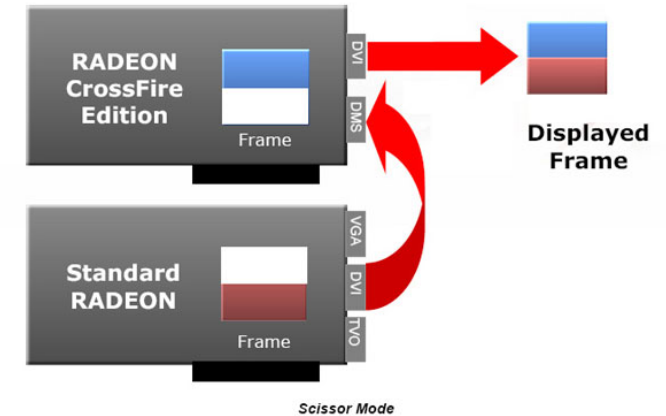
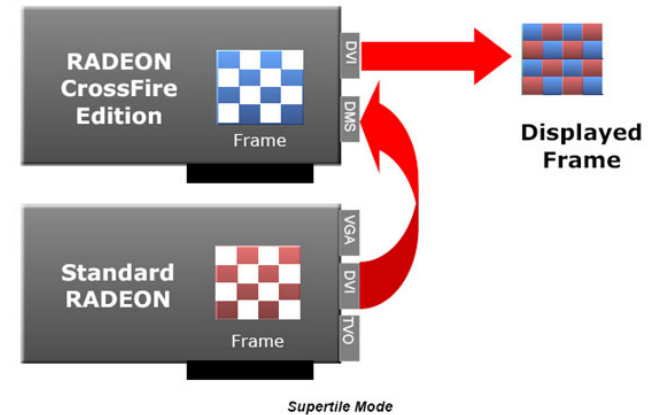
- Split Frame Rendering (SFR) - Scissors: Splits each frame and sends half the load to each of the graphics cards
- Alternate Frame Rendering (AFR):  
Frame 1 – Card 1, Frame 2 – Card 2, alternating
- PCIe cards are connected by a bridge
- Optimal performance increase: 1,8 max.



# CrossFire (ATI)

3 Modes:

- Supertiling
  - Scissors
  - Alternate Frame Rendering
- Additional AA Mode

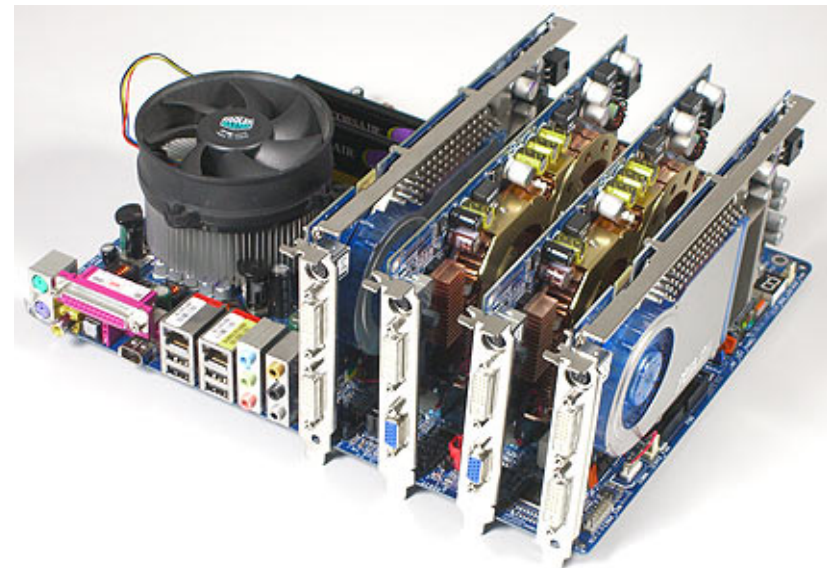
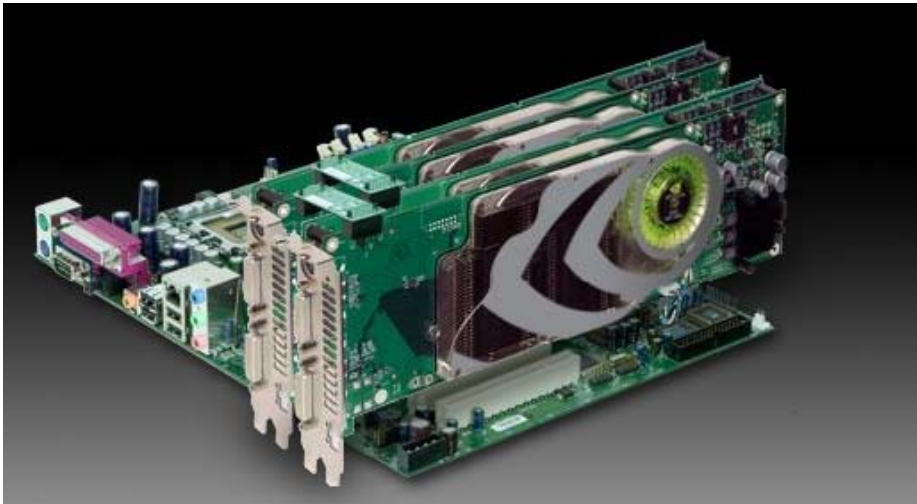


# SLI / CrossFire

- Mainboards with SLI or CrossFire support needed
- Master/slave setup
- SLI: Connection via separate bridge (PCIe communication)
- CrossFire: Inter-GPU connector, compositing on Master board
- CrossFire SuperTiling efficient
- Only 2 cards can be connected (yet)
- Stereo SLI Modes: Alternate Left/Right (?)

# Multiple Boards with Multi GPUs

- Quad SLI by Nvidia
- More coming



---

# Parallel Graphics Hardware

Types of parallel graphics:

1. On-chip / on a graphics board
2. Multiple boards (former: graphics supercomputer)
3. Multiple boards with multi GPUs (1+2)
4. **PC cluster with standard network**
5. PC cluster with special hardware

# Parallel Cluster Rendering (1)

- PC Cluster
  - Off-the-shelf hardware
  - Network (LAN)
  - Cheap
  - Scalable
- Distributed Software System



# Parallel Cluster Rendering (2)

- power of cluster  $\geq$  power of supercomputer
  - Price of cluster  $\ll$  price of supercomputer
  - BUT: problems of cluster
    - How to make cluster PCs work together
    - On a single image  
(or consistent set of images)
- Parallel Execution of Rendering !
- Cluster **synchronisation** (genlocking) !

# Cluster Synchronisation

4

Q: How to  
**synchronize**  
multiple  
displays?

(1) Simple: PC +  
Multiple  
graphic  
outputs

(2) Not so simple:  
Multiple  
workstations



---

# Parallel Graphics Hardware

Types of parallel graphics:

1. On-chip / on a graphics board
2. Multiple boards (former: graphics supercomputer)
3. Multiple boards with multi GPUs (1+2)
4. PC cluster with standard network
5. **PC cluster with special hardware**

# Example: CAVE

“Computer Assisted Virtual Environment”™

- Has 3 to 6 large screens
- Puts user in a room for visual immersion
- Usually driven by a single or group of powerful graphics engines – nowadays PC cluster



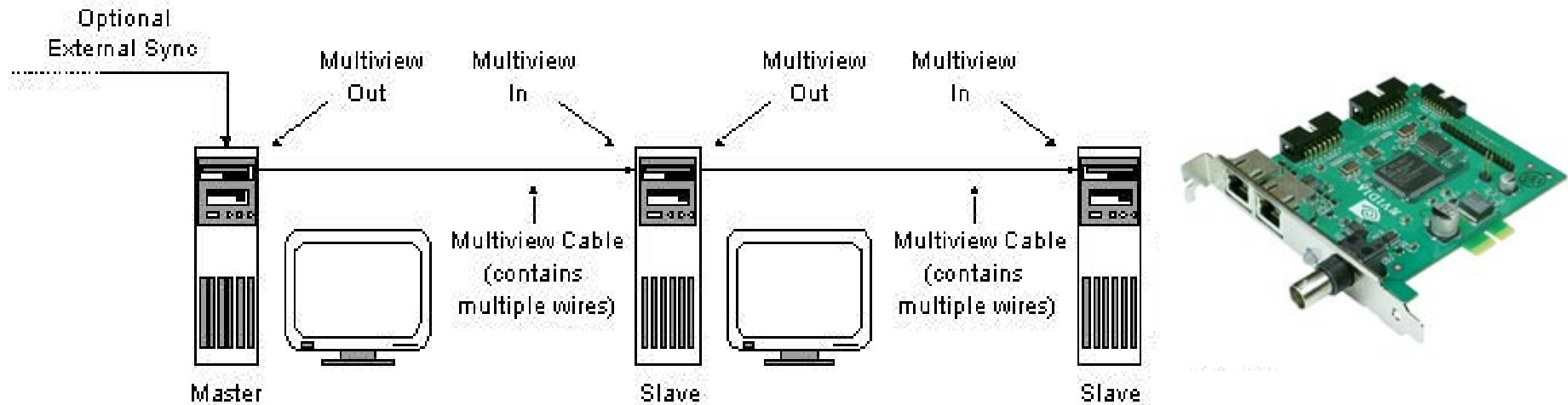
---

# Example: CAVE & Shuttering



# Hardware Synchronisation

Synchronizing multiple displays/workstations



## Frame Lock:

Synchronizing frame buffer swap

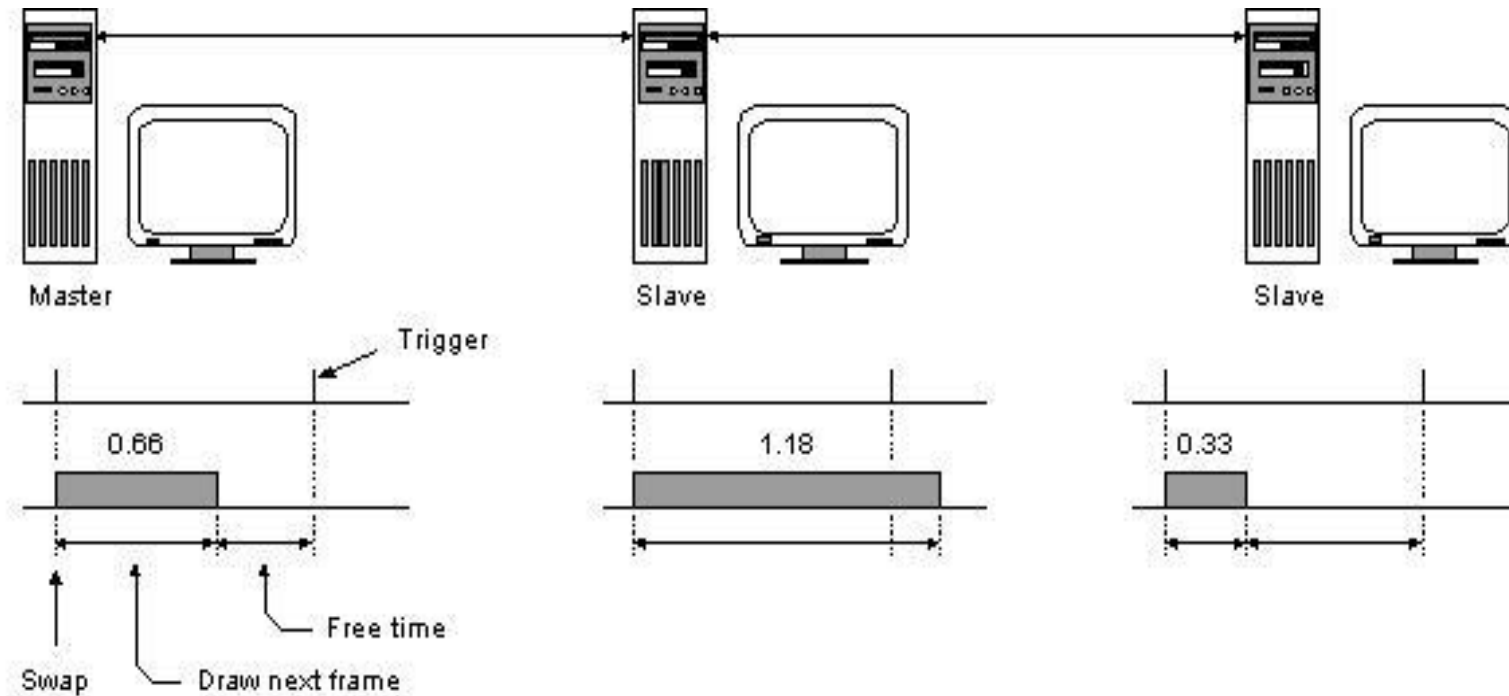
- Begins redrawing at the same time

## Genlock:

**Exact** synchronization of vertical synch (electron beam of CRT)

- Refreshes each pixel synchronously

# RateLock



Mechanism to ensure equal rendering rate of multiple applications

- minimum time per frame (hardware skips frame if too slow)

---

# Example: Blue-C



---

# 3D Card High End Model

- nVidia Quadro FX 5500 (~ € 2900.- ), 1GB
- Based on G71 chip (such as GeForce 7900)
- **24 Pixel Pipelines, 8 Vertex Pipelines**
- 33.6 GB/s Bandwidth
- 10.32 billion pix/sec.
- Vertices: 860 M/s
- optimized OpenGL drivers (comp. to consumer card)



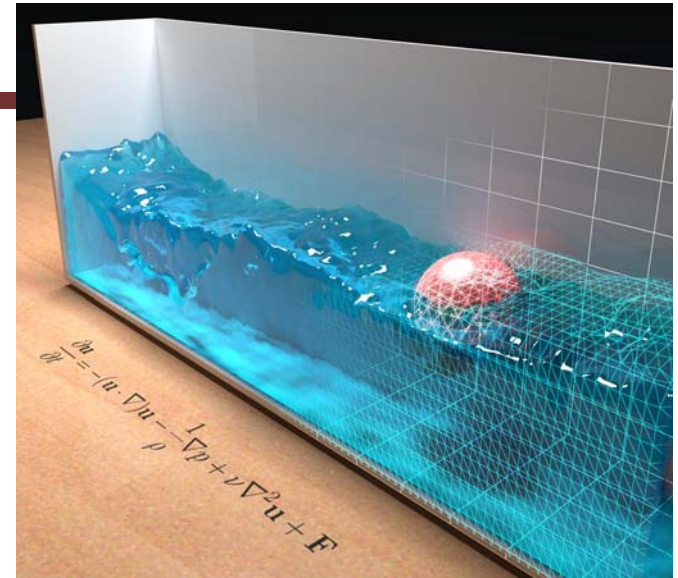
---

# Some Relevant Features (for VR)

- Memory size: 512 MB - 1 GB
- Dual DVI / VGA Output (up to 3840x2400)
- Multi-Display support
- **OpenGL quad-buffered stereo** (3-pin sync connector)
- **Scalable Link Interface** (SLI™) Technology
- Edge Blending for Powerwalls (supported by driver)
- G-Sync Option Board with **FrameLock** and **Genlock**
- Quality: Full-Scene Antialiasing (FSAA); OpenGL 2.0; Shader Model 3.0,....

# Physics Hardware

- Ageia Physics Card
  - Cloth Simulation
  - Particles, Fire, Fluids
- HavocFX: calculation on GPU (not released yet)
- Only computes collisions/deformations/explosions ...
- Puts more rendering load on graphics card



---

# Nächster VO Termin

25. Oktober '06

(10:30-12:15)

Zemanek HS