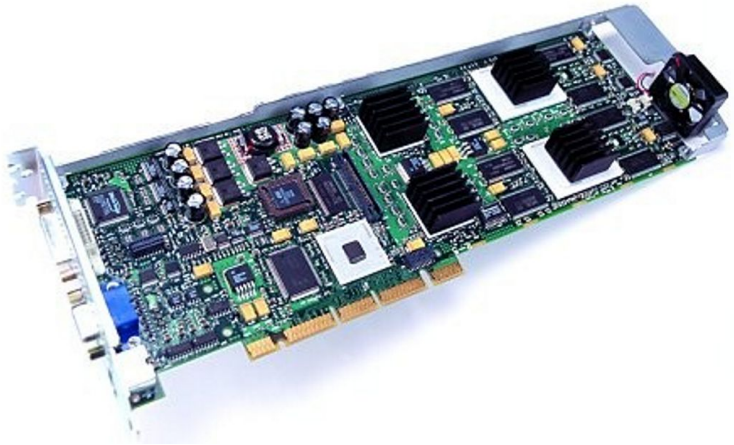
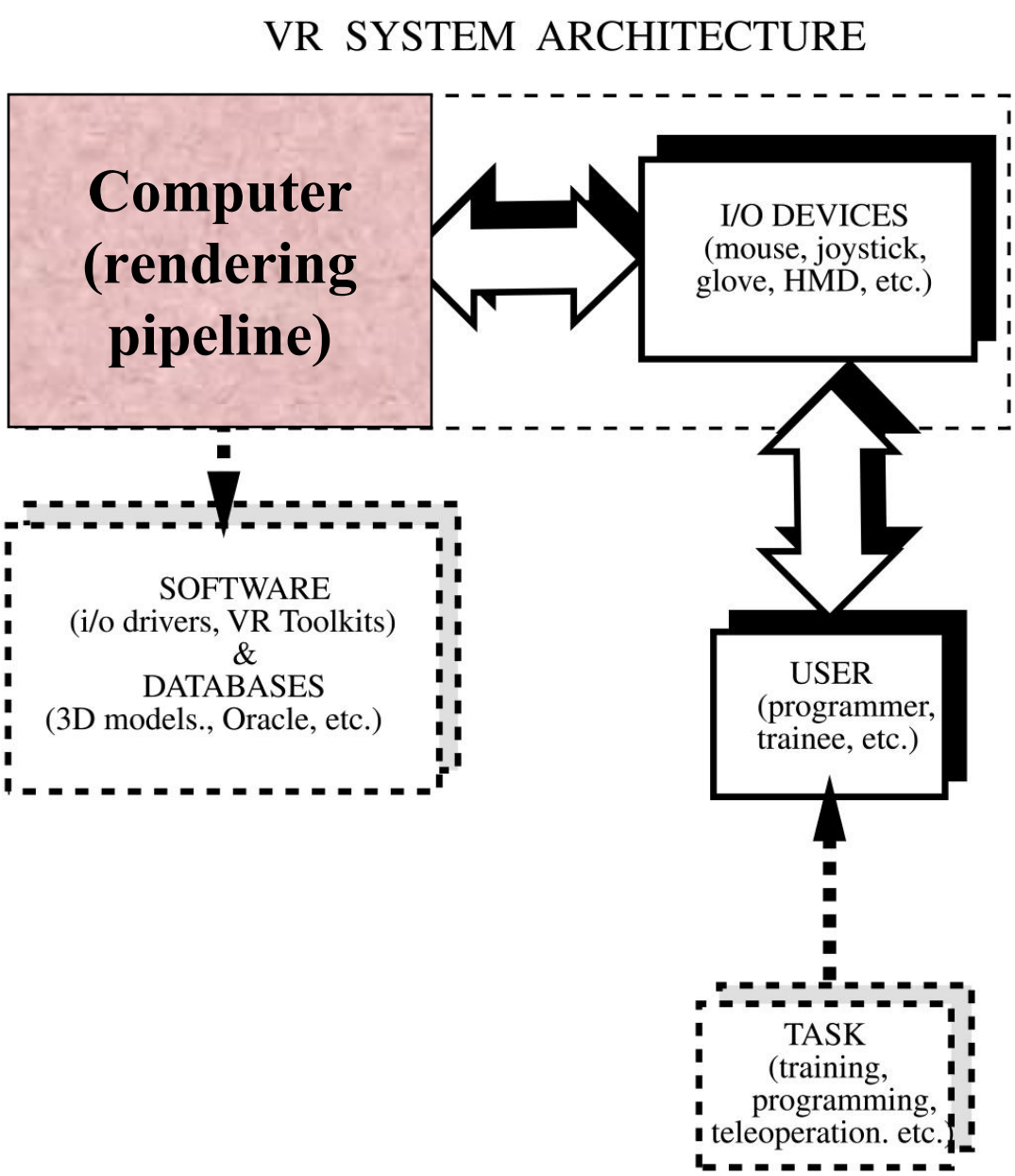
Computing Architectures for Virtual Reality





System architecture

The VR Engine

Definition:

A key component of the VR system which reads its input devices, accesses task-dependent databases, updates the state of the virtual world and feeds the results to the output displays.

It is an ***abstraction*** – it can mean one computer, several co-located cores in one computer, several co-located computers, or many remote computers collaborating in a distribute simulation

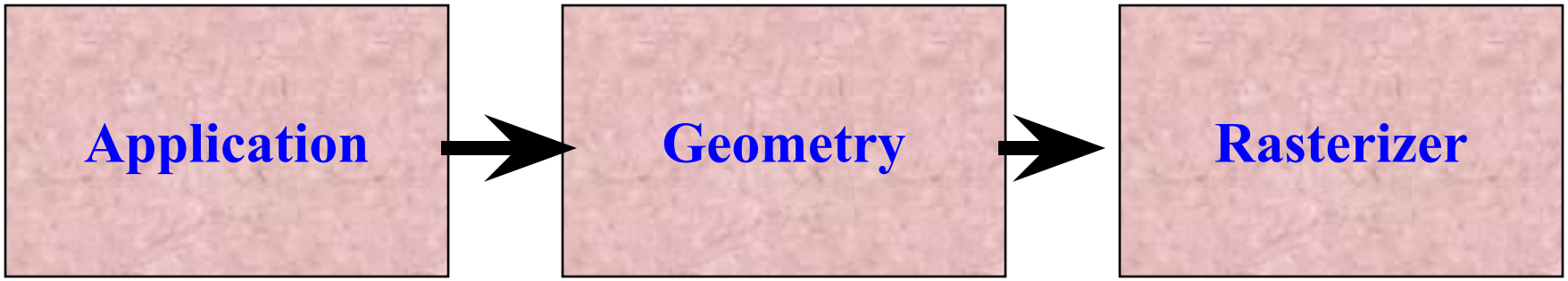
Computing Architectures

The real-time characteristic of VR requires a VR engine which is powerful in order to assure:

- fast graphics and haptics refresh rates (30 fps for graphics and hundreds of Hz for haptics);
- low latencies (<100 ms to avoid simulation sickness);
- at the core of such architecture is the ***rendering pipeline***.
- within the scope of this course rendering is extended to include haptics

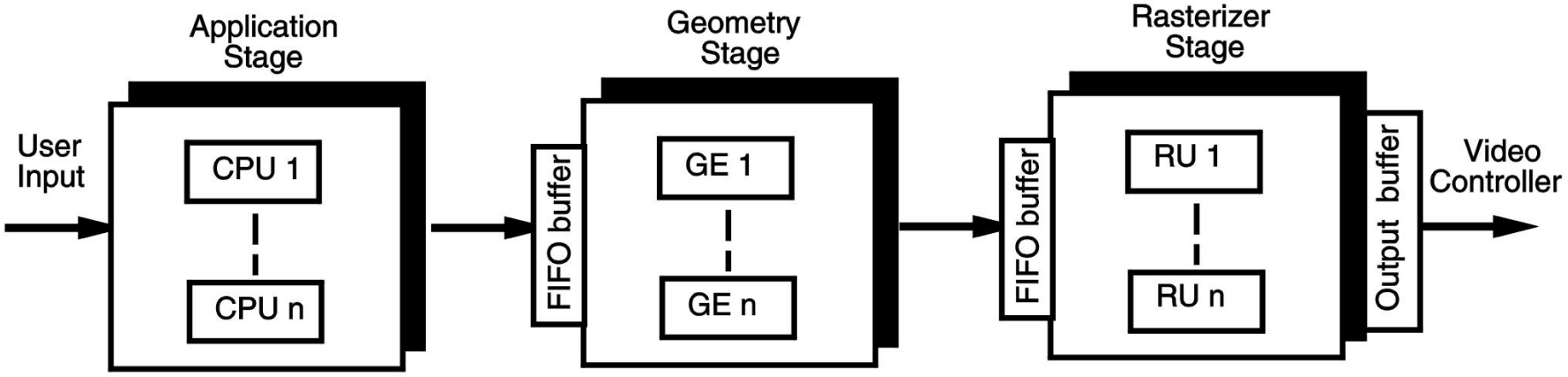
The Graphics Rendering Pipeline

The process of creating a 2-D scene from a 3-D model is called "rendering." The rendering pipeline has ***three functional stages***. The speed of the pipeline is that of its slowest stage.



The Graphics Rendering Pipeline

Old rendering pipelines were done in software (slow)
Modern pipeline architecture uses parallelism
and buffers. The application stage is implemented in software,
while the other stages are *hardware-accelerated*.

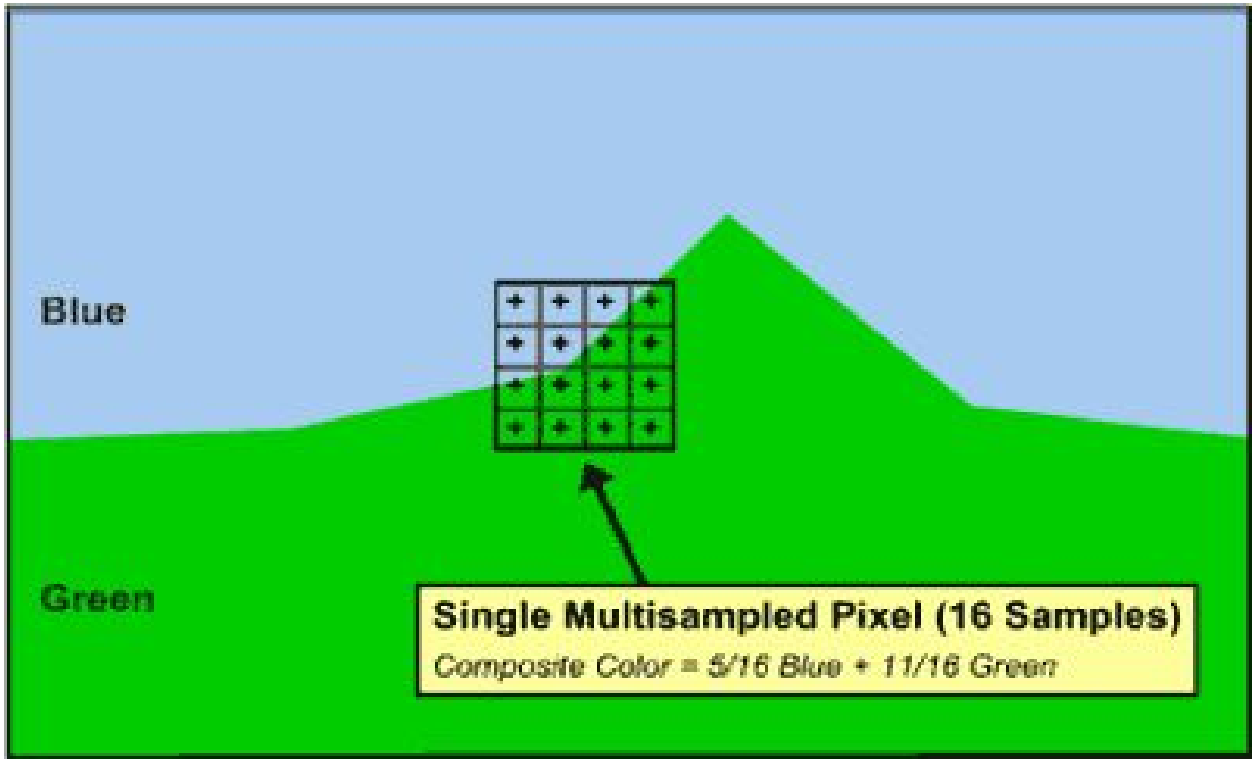


✓ Modern pipelines also do ***anti-aliasing*** for points, lines or the whole scene;

Aliased polygons (jagged edges)

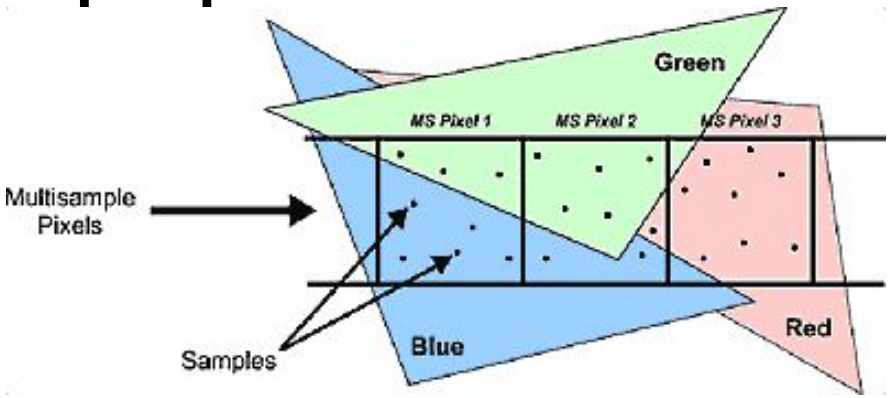
Anti-aliased polygons

✓ **How is *anti-aliasing* done? Each pixel is subdivided (sub-sampled) in n regions, and each sub-pixel has a color;**

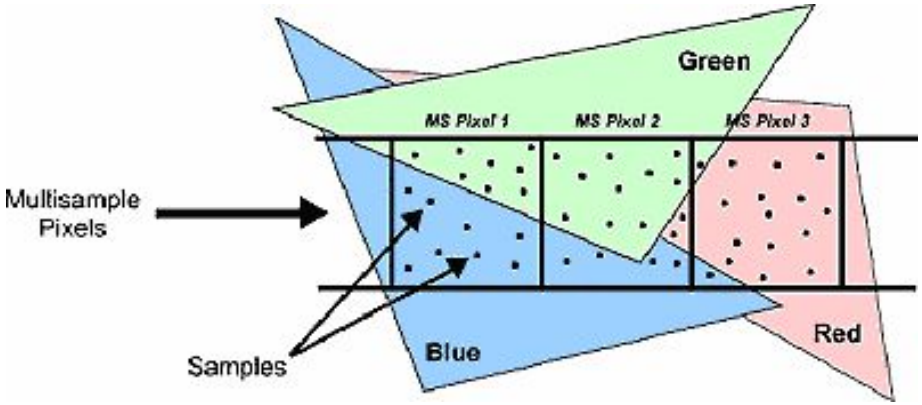


The anti-aliased pixel is given a shade of green-blue (5/16 blue + 11/16 green). Without sub-sampling the pixel would have been entirely green – the color of the center of the pixel (from Wildcat manual)

✓ **More samples produce better *anti-aliasing*;**



8 sub-samples/pixel

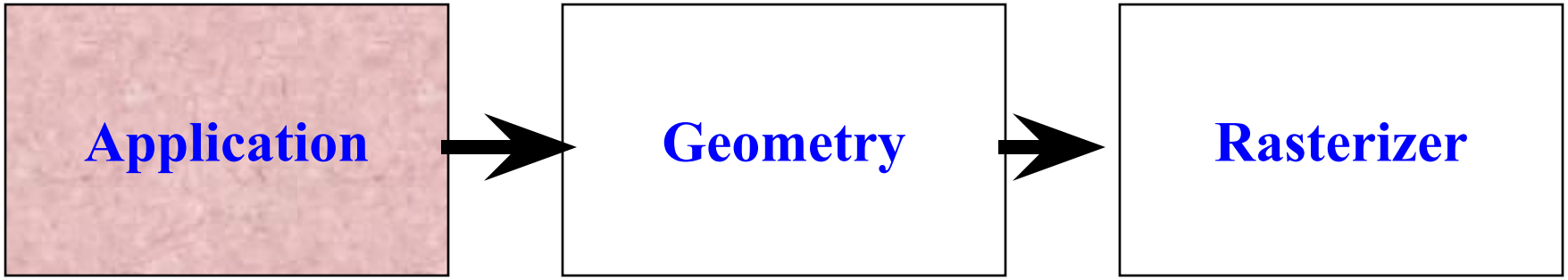


16 sub-samples/pixel

Multisample Pixel	Actual Color Makeup			16 Sample Final Color			8 Sample Final Color		
	Red	Green	Blue	Red	Green	Blue	Red	Green	Blue
MS Pixel 1	0%	47%	53%	0%	50%	50%	0%	37%	62%
MS Pixel 2	17%	59%	24%	12%	62%	25%	12%	62%	25%
MS Pixel 3	86%	10%	4%	87%	6%	6%	100%	0%	0%

From Wildcat "SuperScene" manual
http://62.189.42.82/product/technology/superscene_antialiasing.htm

The Rendering Pipeline

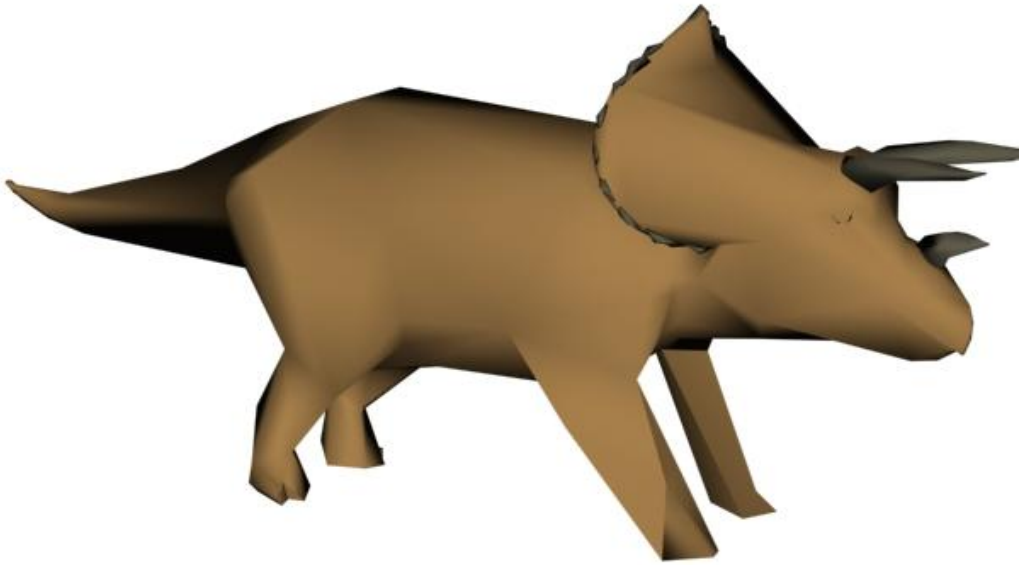


The application stage

- ✓ Is done entirely in software by the CPU;
- ✓ It reads Input devices (such as gloves, mouse);
- ✓ It changes the coordinates of the virtual camera;
- ✓ It performs collision detection and collision response (based on object properties) for haptics;
- ✓ One form of collision response if force feedback.

Application stage optimization…

✓ Reduce model complexity (models with less polygons – less to feed down the pipe);



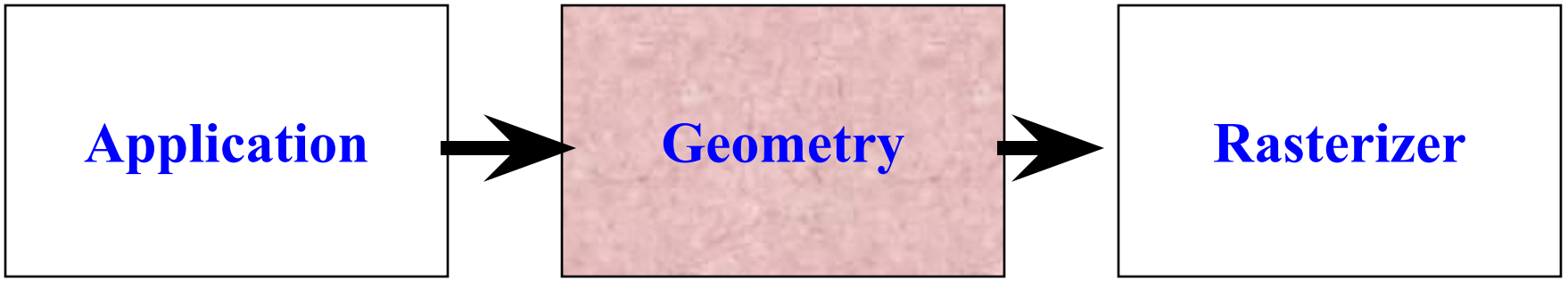
Low res. Model
~ 600 polygons

Higher resolution model
134,754 polygons.

Application stage optimization…

- ✓ Reduce floating point precision (single precision instead of double precision)
- ✓ minimize number of divisions
- ✓ Since all is done by the CPU, to increase speed a dual-processor (super-scalar) architecture is recommended.

The Rendering Pipeline



The geometry stage

- ✓ Is done in hardware;
- ✓ Consists first of model and view transforms
- ✓ Next the scene is shaded based on light models;
- ✓ Finally the scene is projected, clipped, and mapped to the screen coordinates.

The lighting sub-stage

- ✓ It calculates the surface color based on:
- ✓ type and number of simulated light sources;
- ✓ the lighting model;
- ✓ the reflective surface properties;
- ✓ atmospheric effects such as fog or smoke.
- ✓ Lighting results in object shading which makes the scene more realistic.

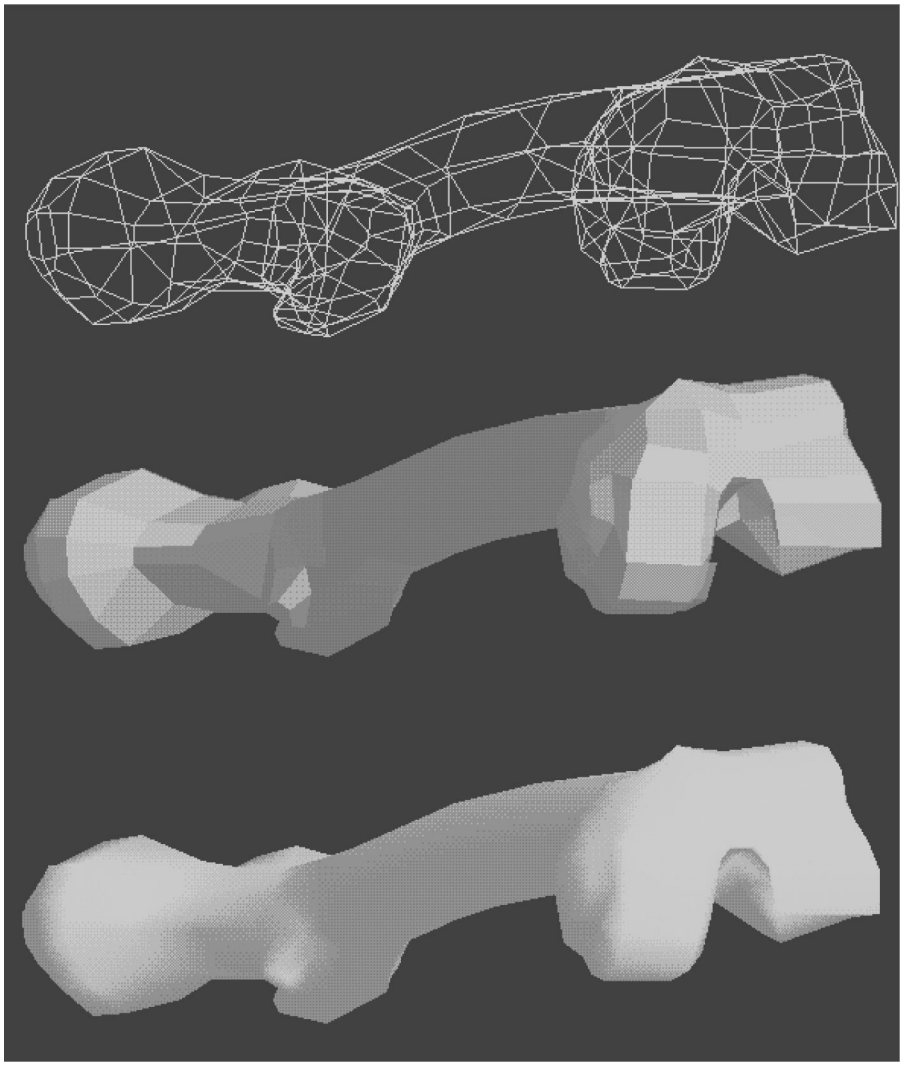
The lighting sub-stage optimization…

- ✓ It takes less computation for fewer lights in the scene;
- ✓ The simpler the shading model, the less computations (and less realism):
- ✓ Wire-frame models;
- ✓ Flat shaded models;
- ✓ Gouraud shaded;
- ✓ Phong shaded.

The lighting models

- ***Wire-frame*** is simplest – only shows polygon visible edges;
- The ***flat shaded*** model assigns same color to all pixels on a polygon (or side) of the object;
- ***Gouraud*** or smooth shading interpolates colors Inside the polygons based on the color of the edges;
- ***Phong shading*** interpolates the vertex normals before calculating the light intensity based on the model described – most realistic shading model.

Computing architectures



Wire-frame model

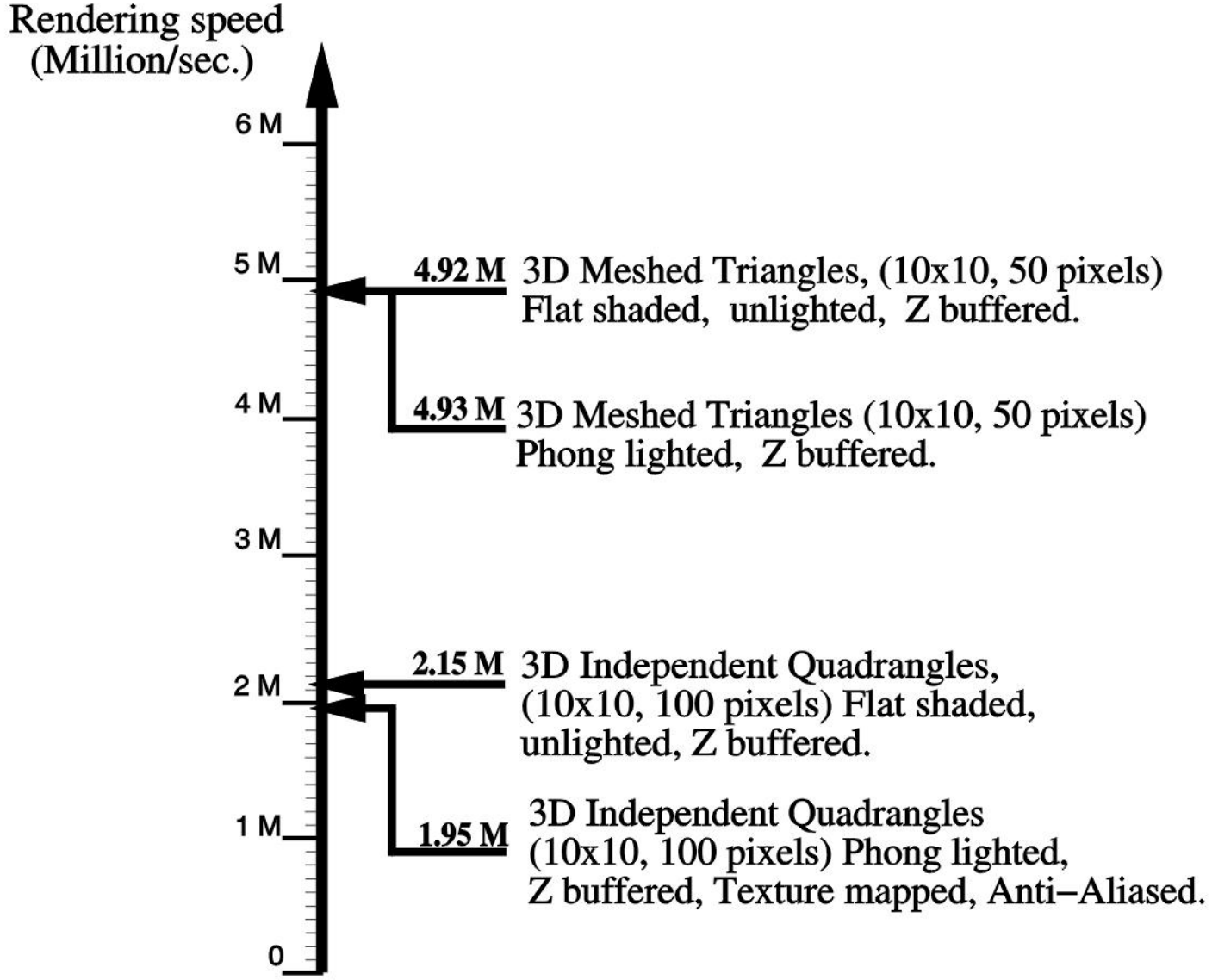
Flat shading model

Gouraud shading model

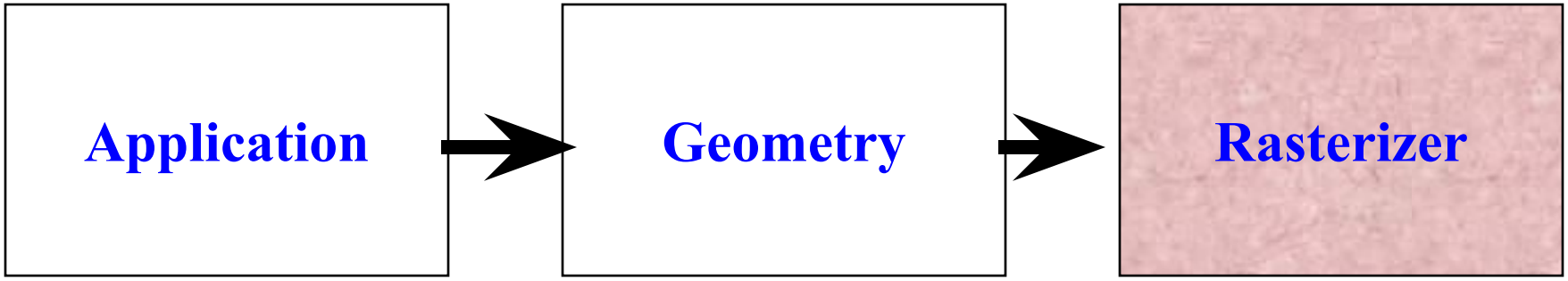
The rendering speed vs. surface polygon type

- ✓ The way surfaces are described influences rendering speed.
- ✓ If surfaces are described by triangle meshes, the rendering will be faster than for the same object described by independent quadrangles or higher-order polygons. This is due to the graphics board architecture which may be optimized to render triangles.
- ✓ Example the rendering speed of SGI Reality Engine.

SGI Onyx 2 with Infinite Reality



The Rendering Pipeline



The Rasterizer Stage

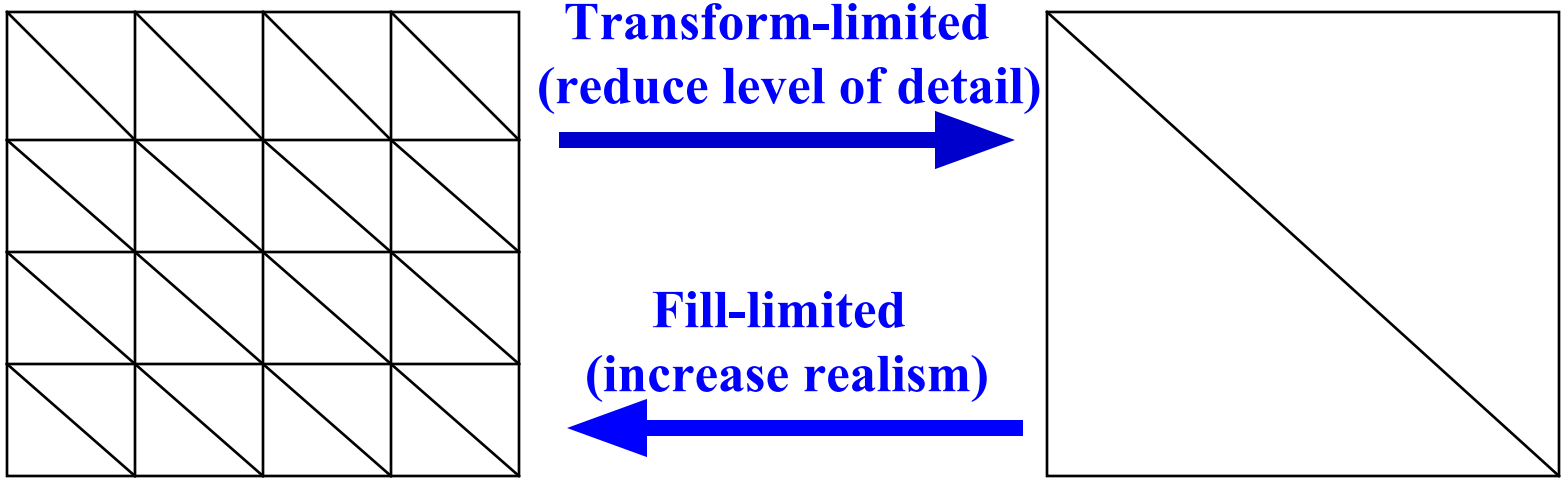
- ✓ Performs operations in hardware for speed;
- ✓ Converts 2-D vertices information from the geometry stage (x,y,z, color, texture) into pixel information on the screen;
- ✓ The pixel color information is in ***color buffer***;
- ✓ The pixel z-value is stored in the ***Z-buffer*** (has same size as color buffer);
- ✓ Assures that the primitives that are visible from the point of view of the camera are displayed.

The Rasterizer Stage - continued

- ✓ The scene is rendered in the ***back buffer;***
- ✓ It is then swapped with the ***front buffer*** which stores the current image being displayed;
- ✓ This process eliminates flicker and is called "double buffering";
- ✓ All the buffers on the system are grouped into the ***frame buffer.***

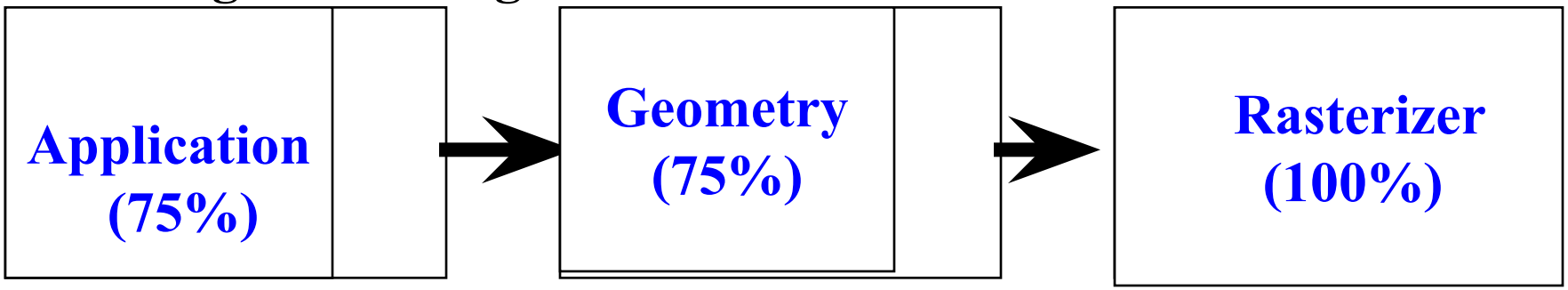
Testing for pipeline bottlenecks

- ✓ If CPU operates at 100% – then the pipeline is ***"CPU-limited"*** (bottleneck in application stage);
- ✓ If the performance increases when all light sources are removed, then the pipeline is ***"transform-limited"*** (bottleneck in geometry stage);
- ✓ If the performance increases when the resolution of the display window, or its size are reduced then the pipeline is ***"fill-limited"*** (bottleneck in rasterizer stage).

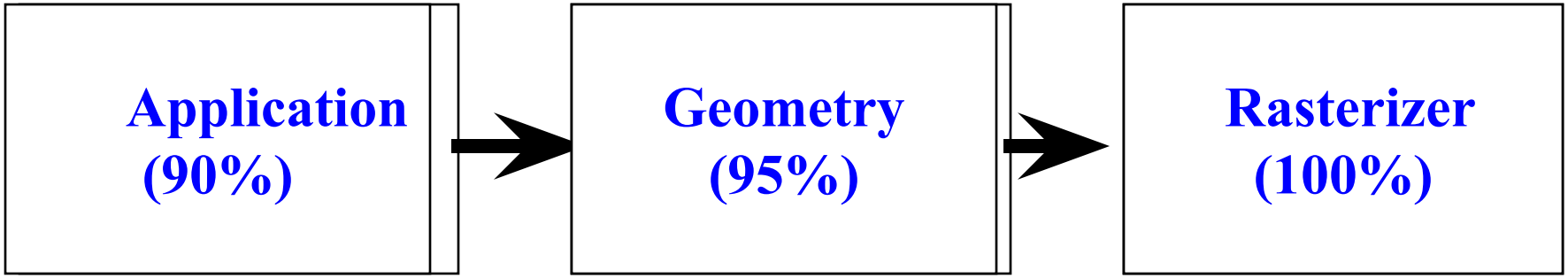

Transform-limited
(reduce level of detail)
Fill-limited
(increase realism)

The Pipeline Balancing

Single buffering

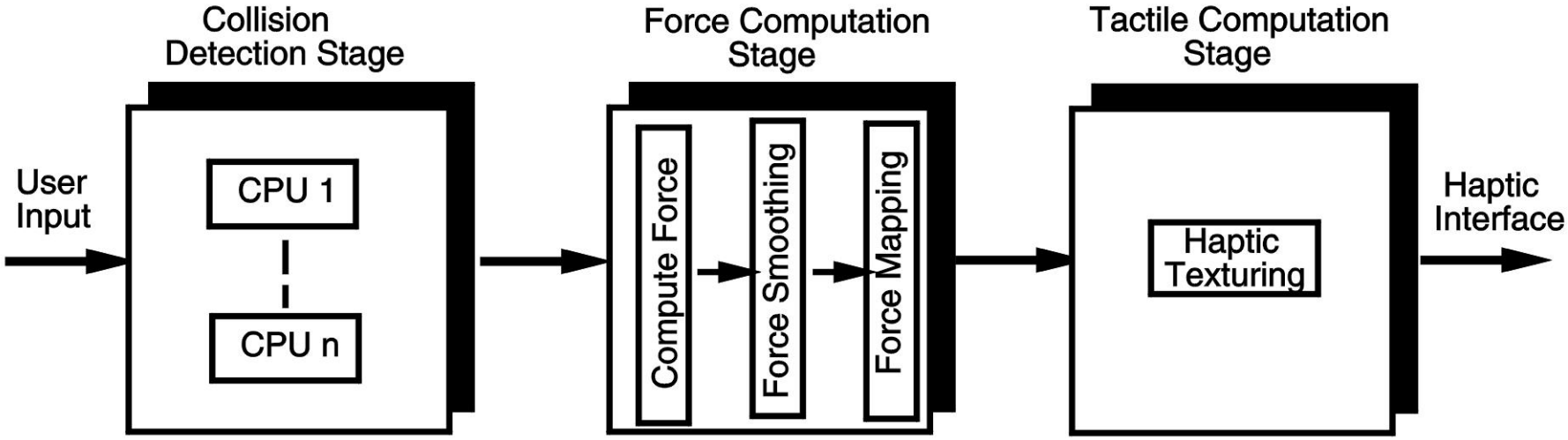


Double buffering, balanced pipeline



The Haptics Rendering Pipeline

The process of computing the forces and mechanical textures Associated with haptic feedback. Is done is software and in hardware. Has three stages too.



PC graphics architecture – PC is King!

- ✓ Went from 66 MHz Intel 486 in 1994 to 3.6 GHz Pentium IV today;
- ✓ Newer PC CPUs are dual (or quad) core – improves performance by 50%

- ✓ Went from 7,000 G-shaded poly./sec (Spea Fire board) in 1994 to 27 Mil G-shaded poly/sec.

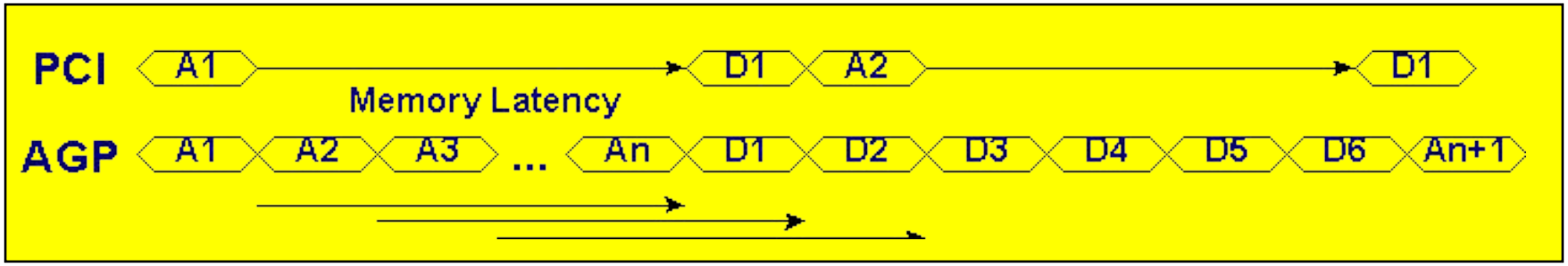
- ✓ Today PCs are used for single or multiple users, single or tiled displays;

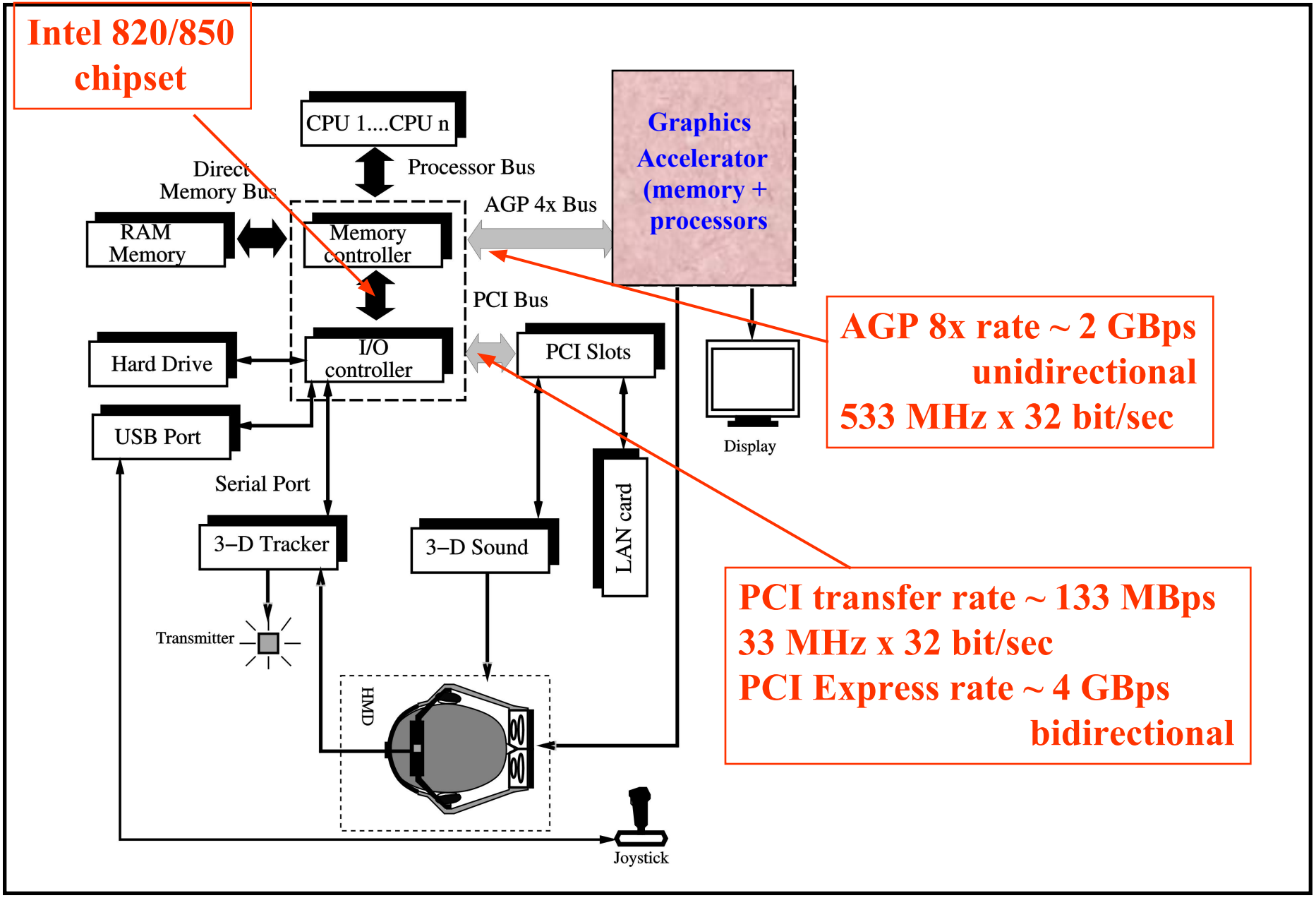
- ✓ Intensely *competitive* industry.

PC bus architecture – just as important

✓ Went from 33 MHz “Peripheral Component Interface” (PCI) bus to 264 MHz “Accelerated Graphics Port” (AGP4x) bus, and doubled again in the AGP8x;

✓ Larger throughput and lower latency since address bus lines decoupled from data lines. AGP uses “sideband” lines



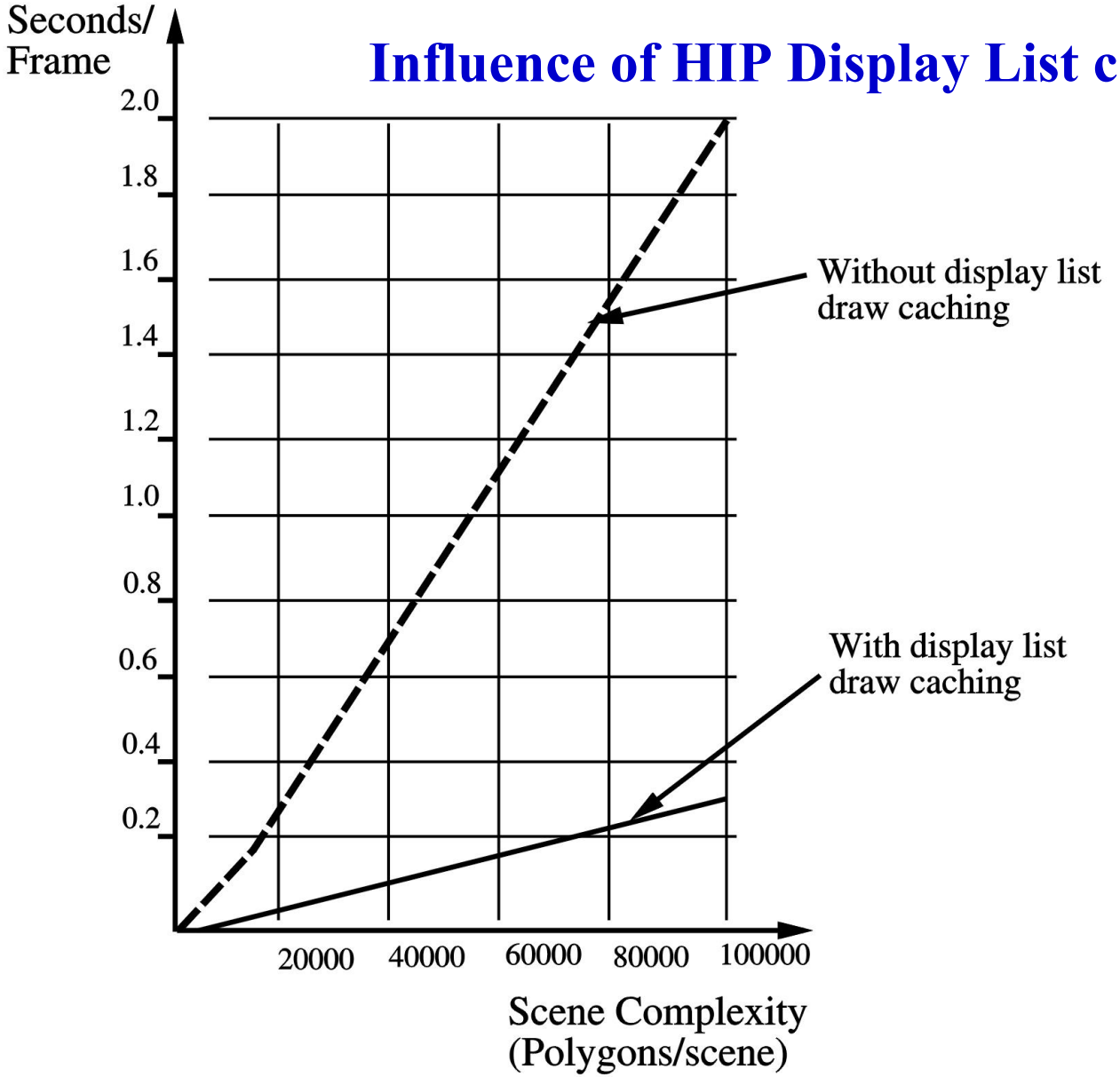


Today's PC system architecture

The SGI InfiniteReality computer:

- A *massively parallel* architecture based on proprietary ASIC technology; Was considered for a long time the "crème-de-la-crème" in VR computers.
- Can have up to 10,000 CPUs in the application stage,
- The geometry board consists of a "*host interface processor*" (HIP), a "geometry distributor" and geometry engines (with a FIFO queue);
- The HIP task is to pull data from main memory (using DMA); it also has its own 16 MB cache, such that the need to pull data is reduced.

Influence of HIP Display List caching



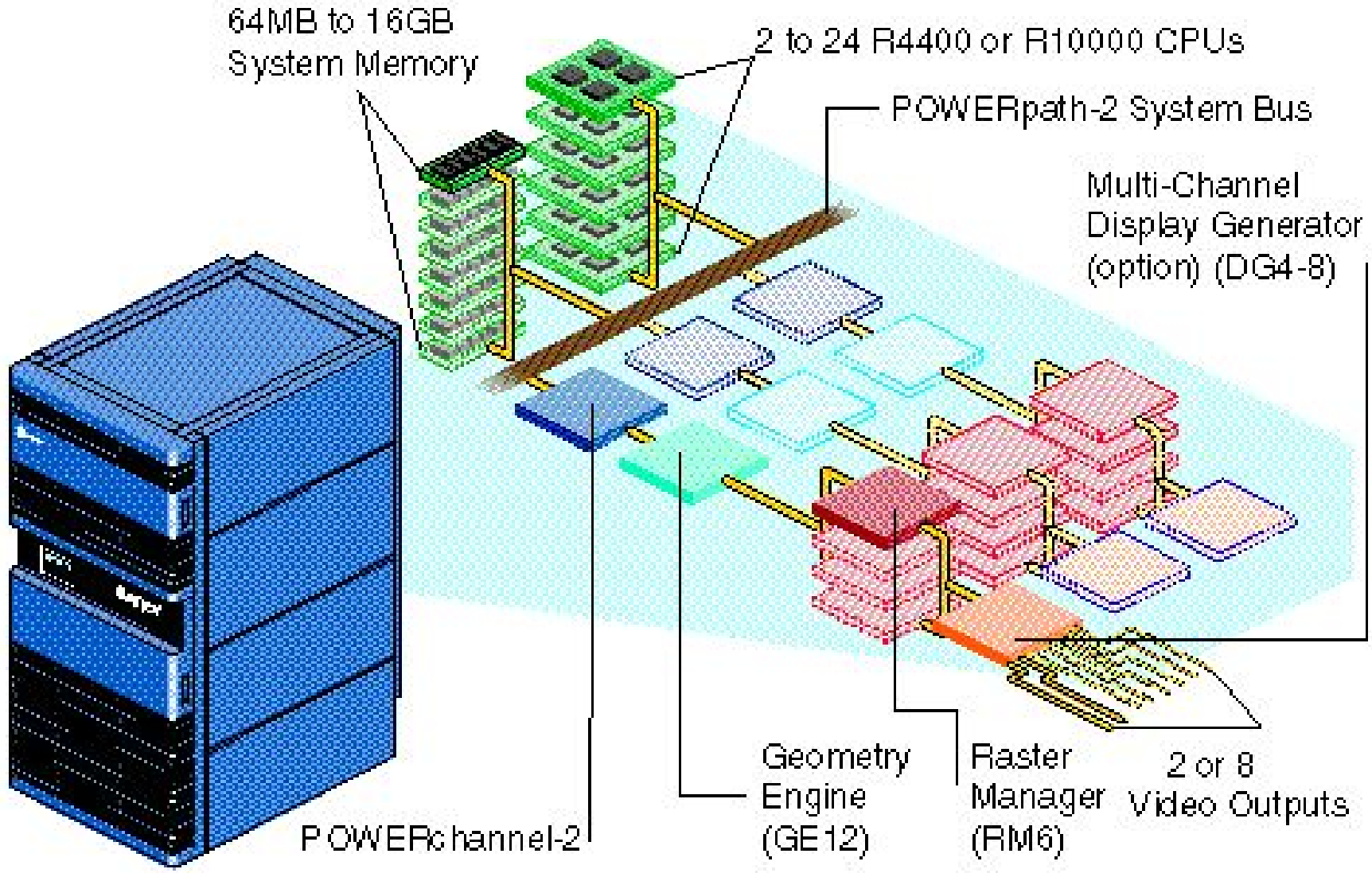


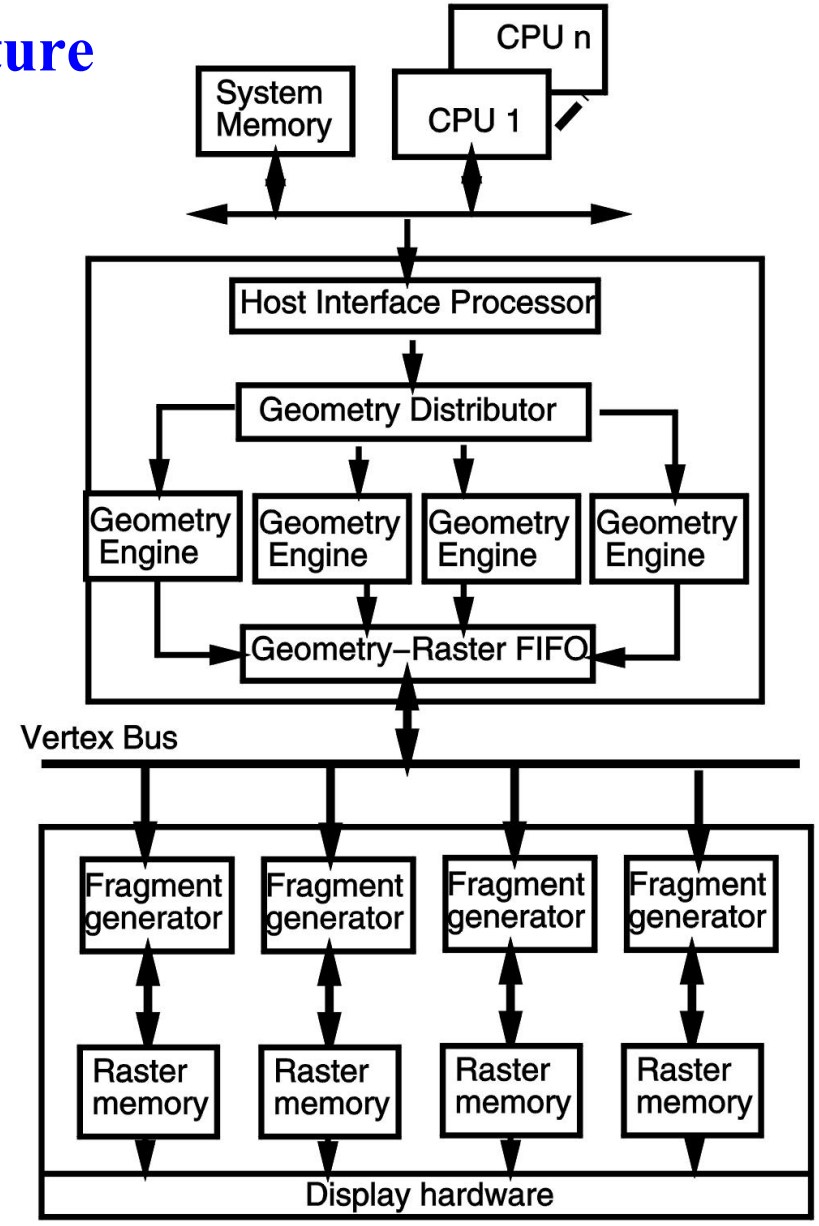
Figure 2. InfiniteReality Rack System Block Diagram

The SGI InfiniteReality - continued:

- ✓ The HIP sends data to the *geometry distributor* which distributes the load to the geometry engines on a "least busy" fashion (with a FIFO queue);
- ✓ Each Geometry Engine uses SIMD (single-instruction-multiple-data) by processing the three coordinates of the vertex in parallel on three "floating-point cores".
- ✓ The GE floating point core has its own ALU, multiplier and 32-word register in a four-stage pipeline;
- ✓ The FIFO holds the results of the GEs output and writes the merged stream to the *vertex bus;*

SGI Infinite Reality system architecture

- ✓ Data from the vertex bus are received by the "*fragment generators*" on the raster memory board;
- ✓ The fragment generator performs the texturing, color, depth pixel interpolation and anti-aliasing (4 to 8 sub-samples/pixel)
- ✓ Their output is then distributed equally among 80 image engines on the raster board;
- ✓ The image engine tiling pattern is 320x80 pixels;
- ✓ The *display hardware* has dynamic video resize, video timing and D/A conversion;

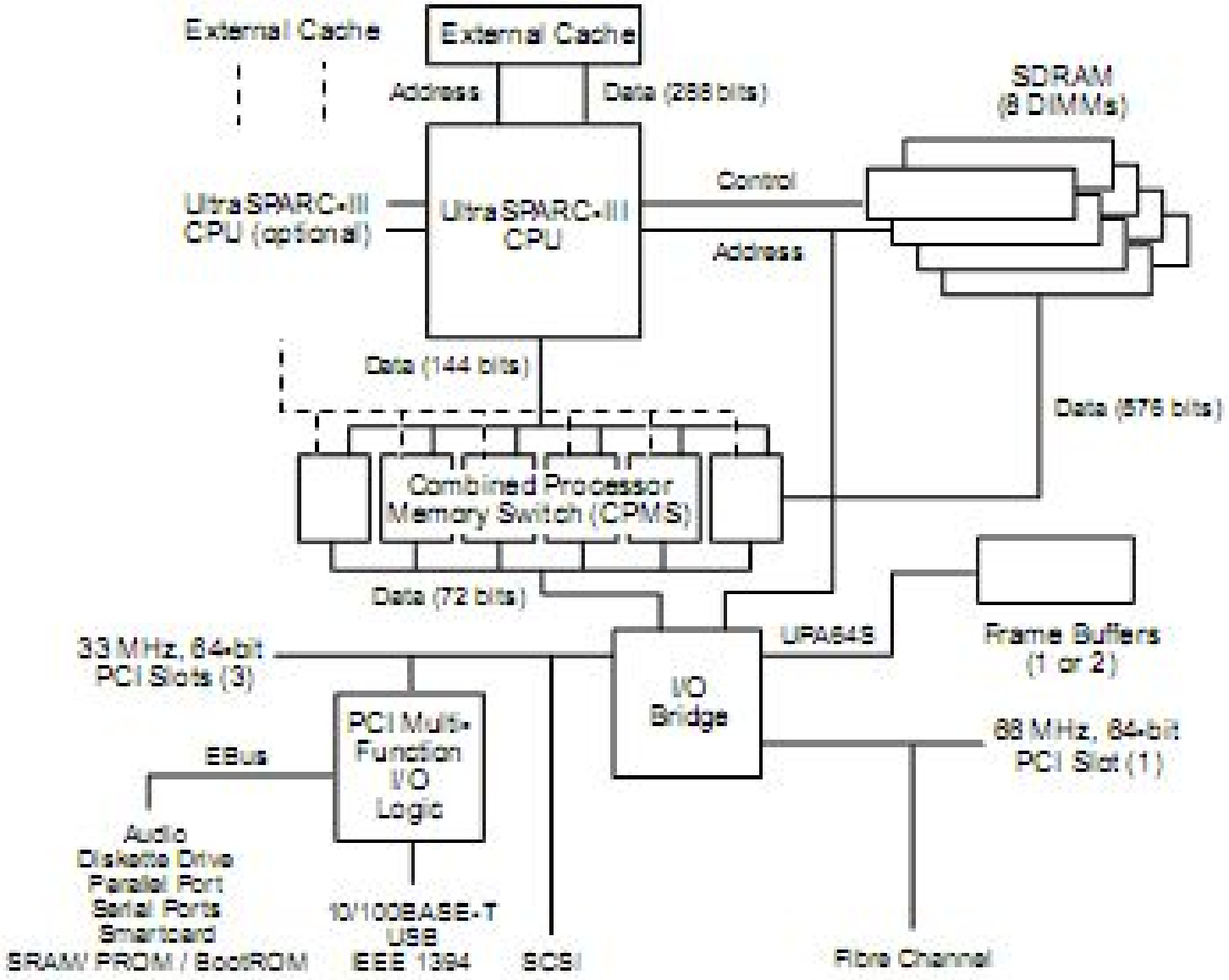


Sun Blade 1000 systems

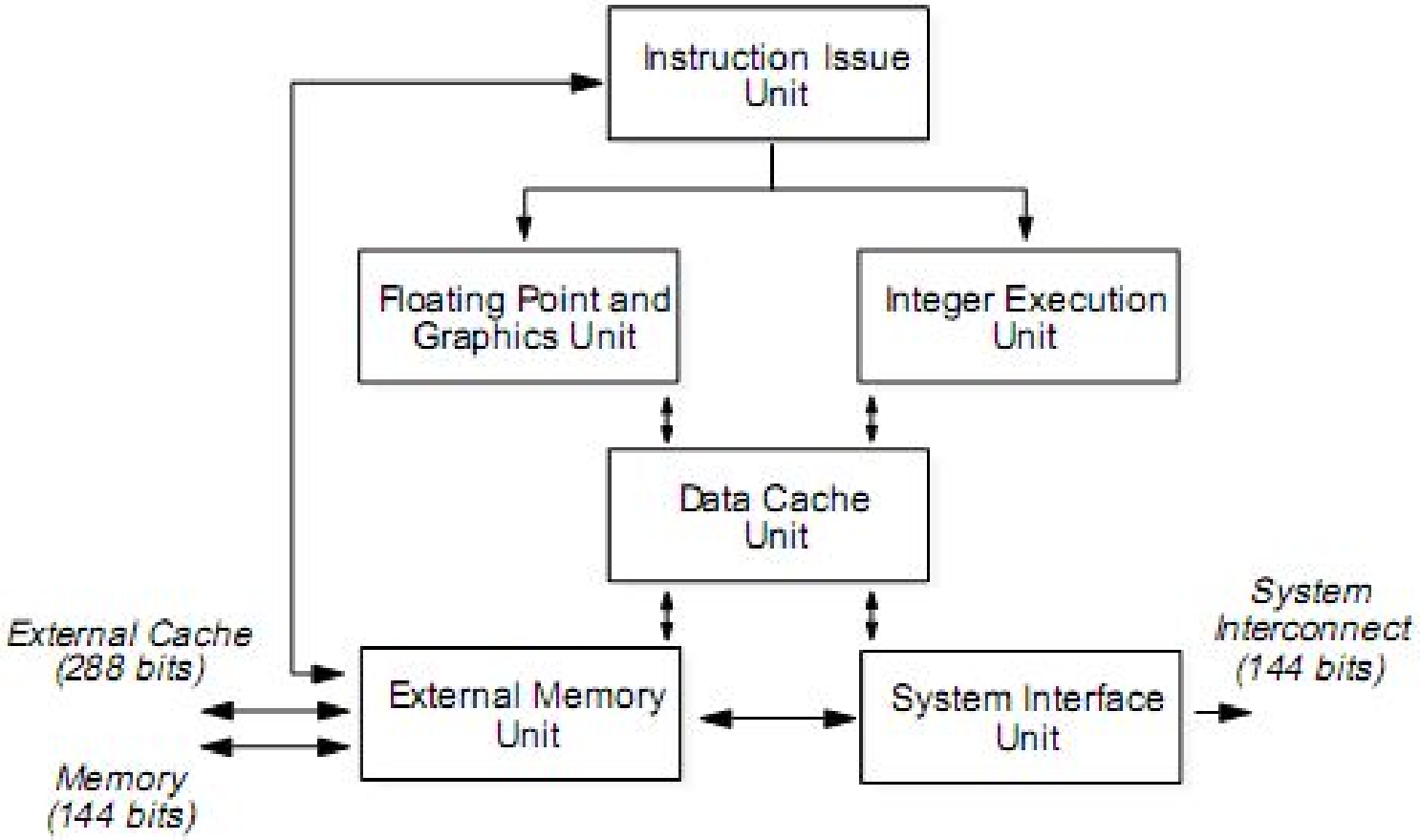
Represents a significant step in multiprocessor desktop computing by integrating new technologies into a workstation package.
Consists of UltraSPARC-III microprocessors tightly integrated into it
UltraSPARC-III microprocessors is implemented in 14 stage pipeline
UltraSPARC-III has 6 key components

- Instruction Issue unit
- Floating point unit
- Integer execution unit
- Data cache unit
- External memory unit

High-level architecture of the Sun Blade 1000 workstation



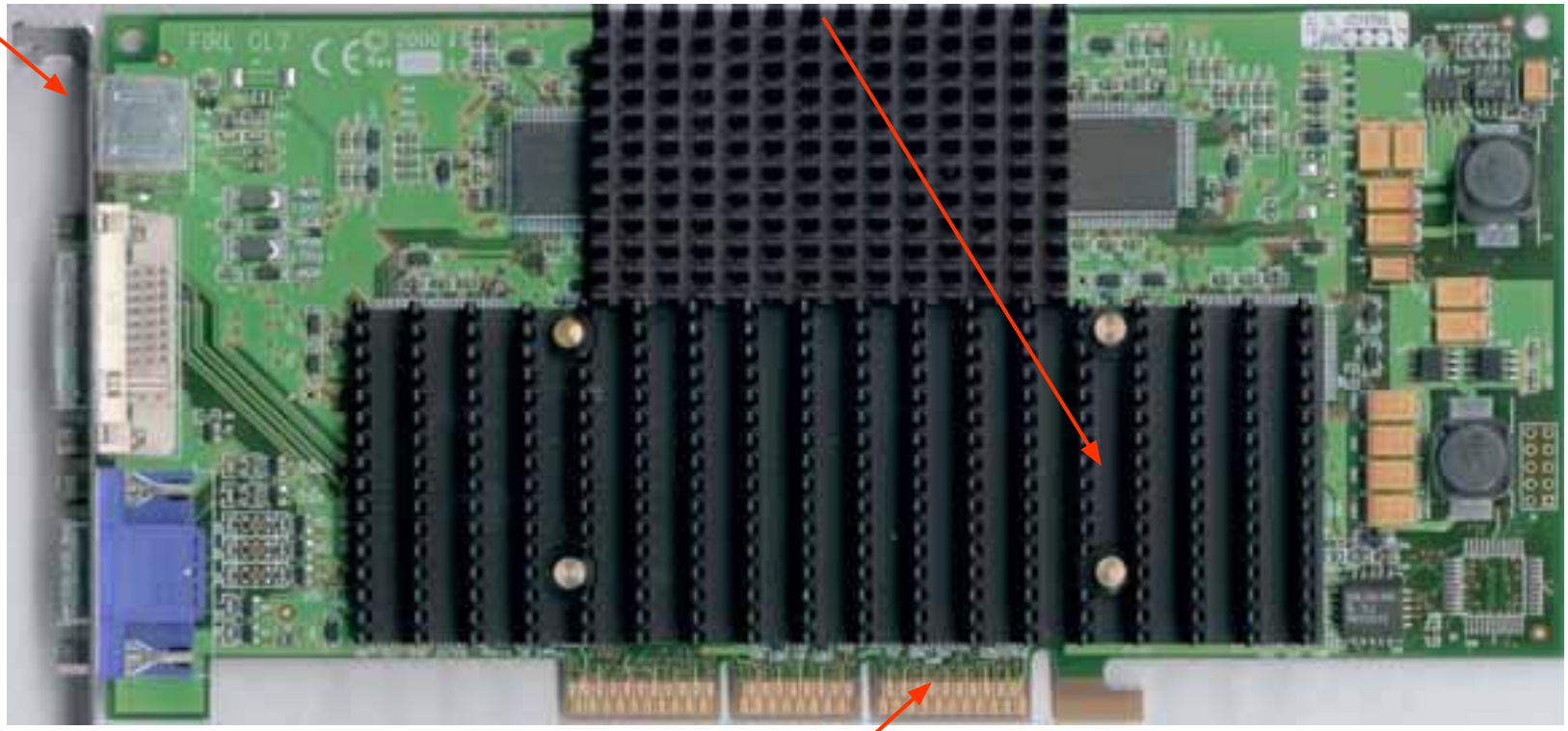
UltraSPARC-III CPU functional block diagram



Fire GL 2

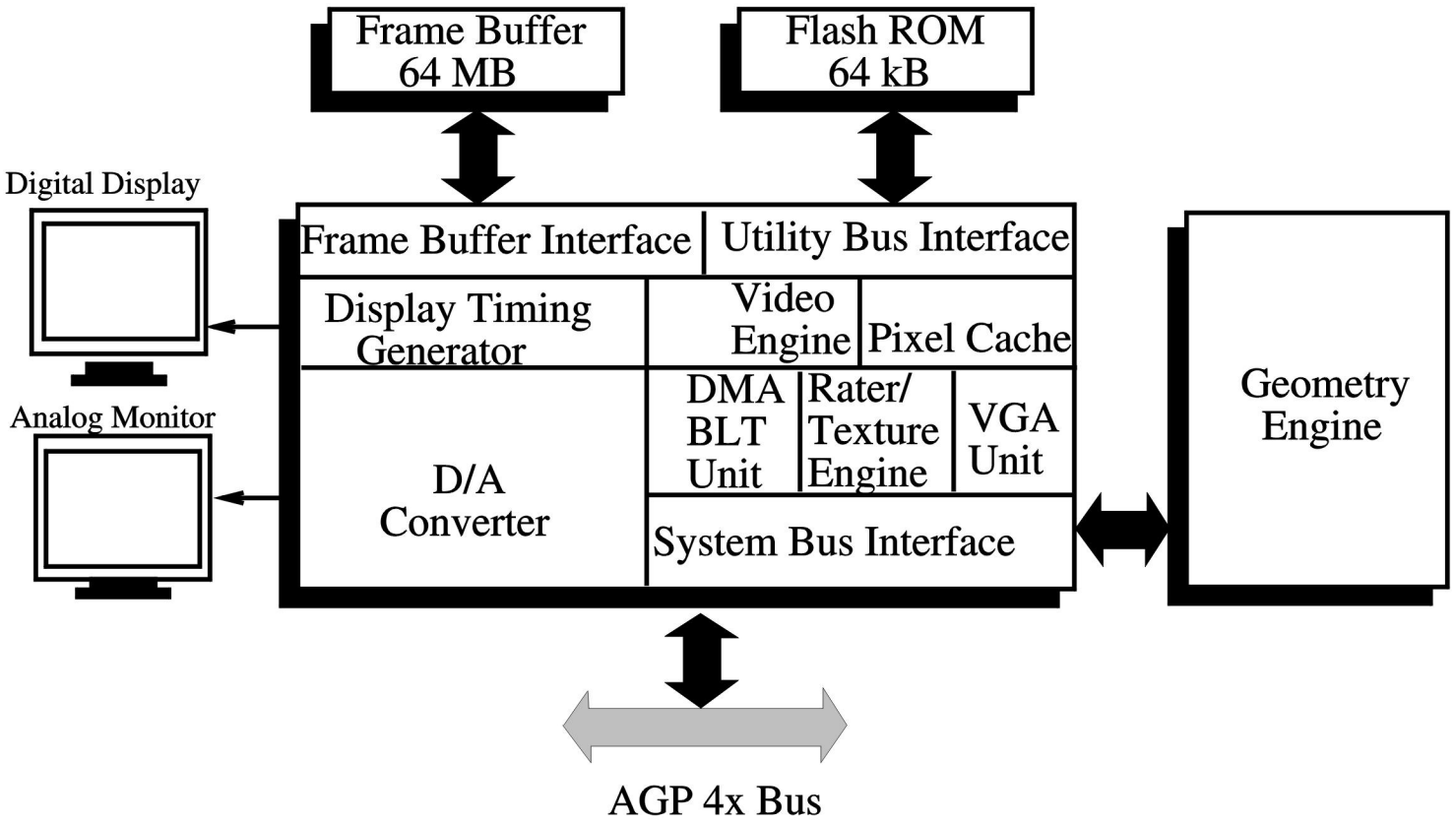
Stereo glasses connector

Passive coolers



AGP bus connector

Fire GL 2 architecture



Fire GL 2 features:

- ✓ 27 Million G-shaded/sec., non-textured polygons/sec;
- ✓ Fill rate is 410 M Pixels/sec.;
- ✓ supports up to 16 light sources;
- ✓ has a 300 MHz D/A converter

Distributed VR architectures

- ✓ *Single-user systems*:
- ✓ multiple side-by-side displays;
- ✓ multiple LAN-networked computers;
- ✓ *Multi-user systems*:
- ✓ client-server systems;
- ✓ peer-to-peer systems
- ✓ hybrid systems;

Improving scalability of DVE

Communication architecture
Interest management
Concurrency control
Data replication
Load distribution

Communication architecture

Can be implemented as

- Client server
- Peer –to – Peer
- Peer –to – Server

Interest management

These strategies exploit the fact that users do not need to receive all update messages related to teh whole world.

They receive only those messages in which they are interested.

Methods used

- Region based
- Interest based

Concurrency control

Replication allows local access and updates to the data

Mechanishms needed to syncronize these replicas

Approaches

- Pessimistic
- Optimistic
- Prediction schemes

Data replication

Should address the issue

How efficient should the replication be

Replication implemented via 2 techiniques

- Prioritized transfer of objects
- Caching and Prefretching

Load distribution

To maintain interactive performance , dymanic load distribution schemes have been introduced where overloaded servers transfer their excessive workloads to less loaded servers.

3 approaches

- Local
- Global
- Adapative dynamic load

Genlock..

- ✓ If the output of two or more graphics pipes is used to drive monitors placed side-by-side, then the display channels need to be synchronized pixel-by-pixel;
- ✓ Moreover, the edges have to be blended, by creating a region of overlap.

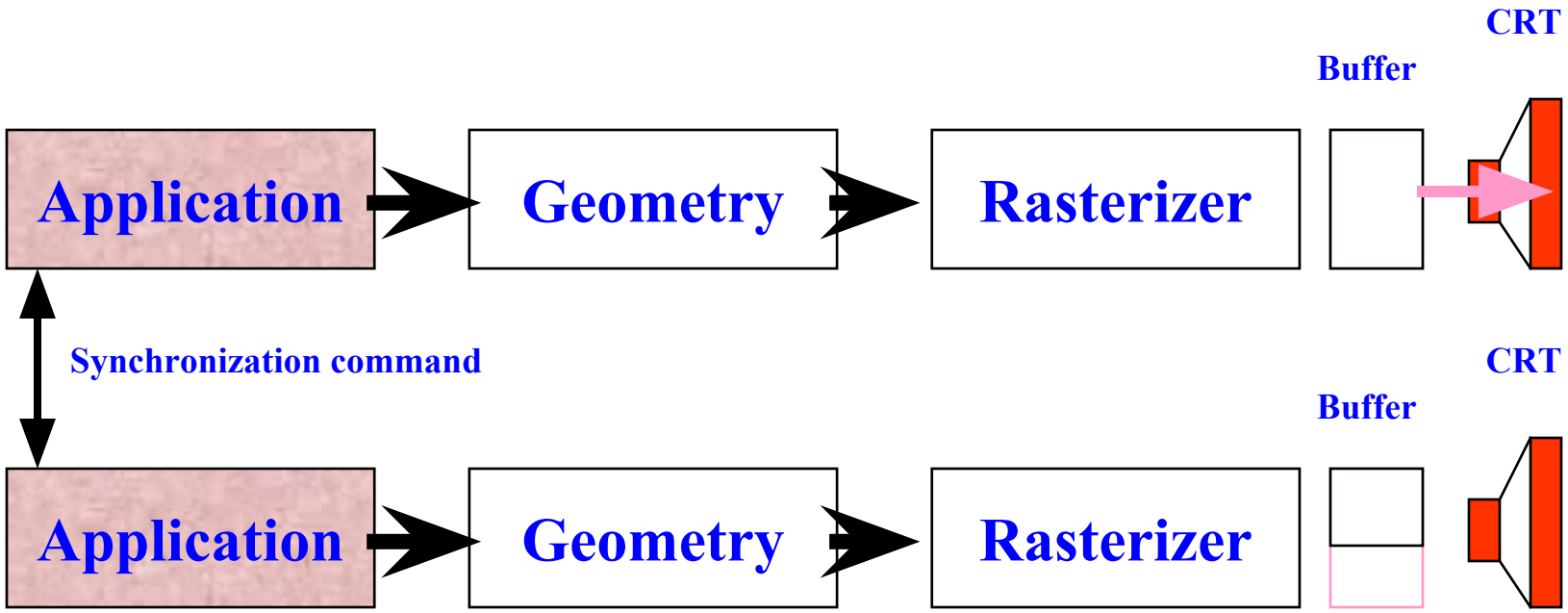
A large screen visual channel is created by using multiple visual channels adjacent to each other.

A three-channel out-the-window display where the channels are not synchronized correctly, resulting in the images being misaligned.

(Courtesy of Quantum3D Inc.)

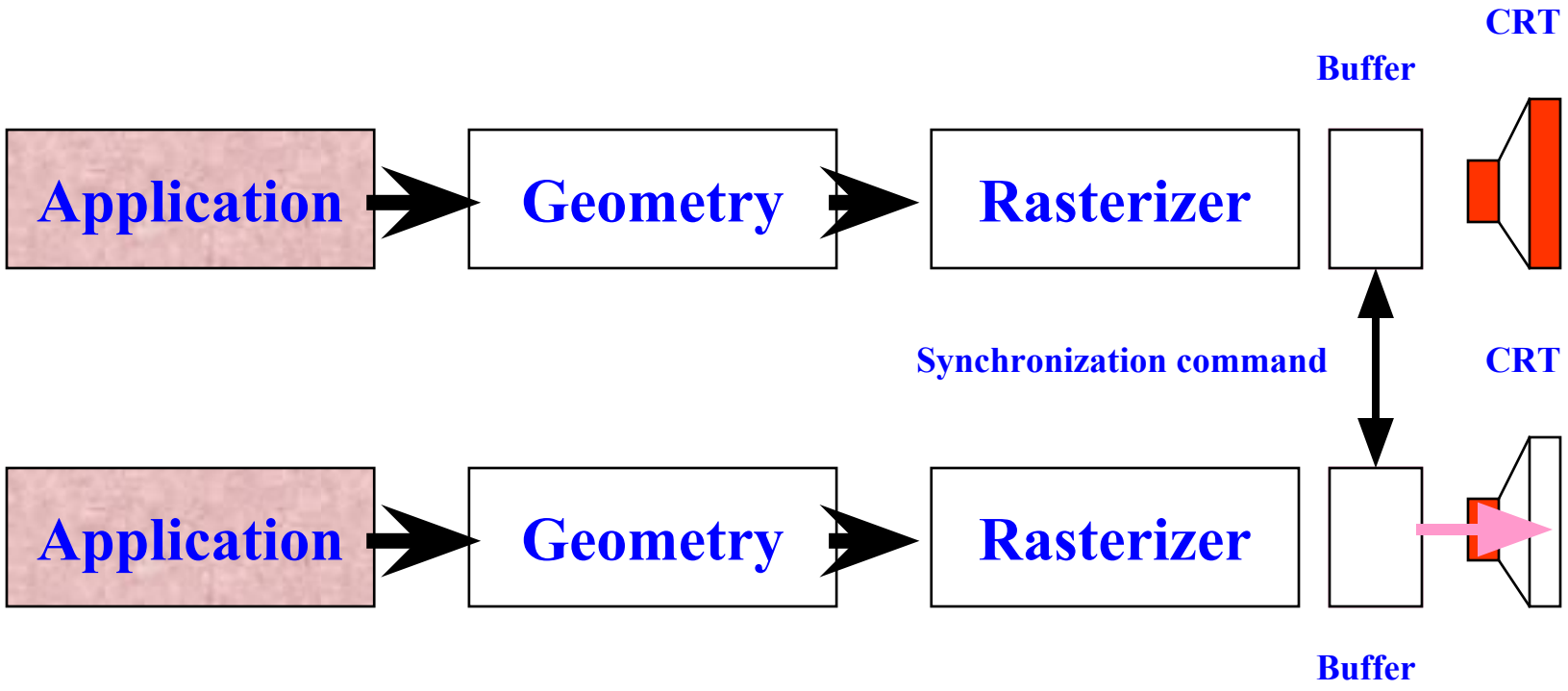
Synchronization of displays:

- ✓ ***software synchronized*** – system commands that frame processing start at same time on different rendering pipes;
- ✓ does not work if one pipe is overloaded – one image finishes first



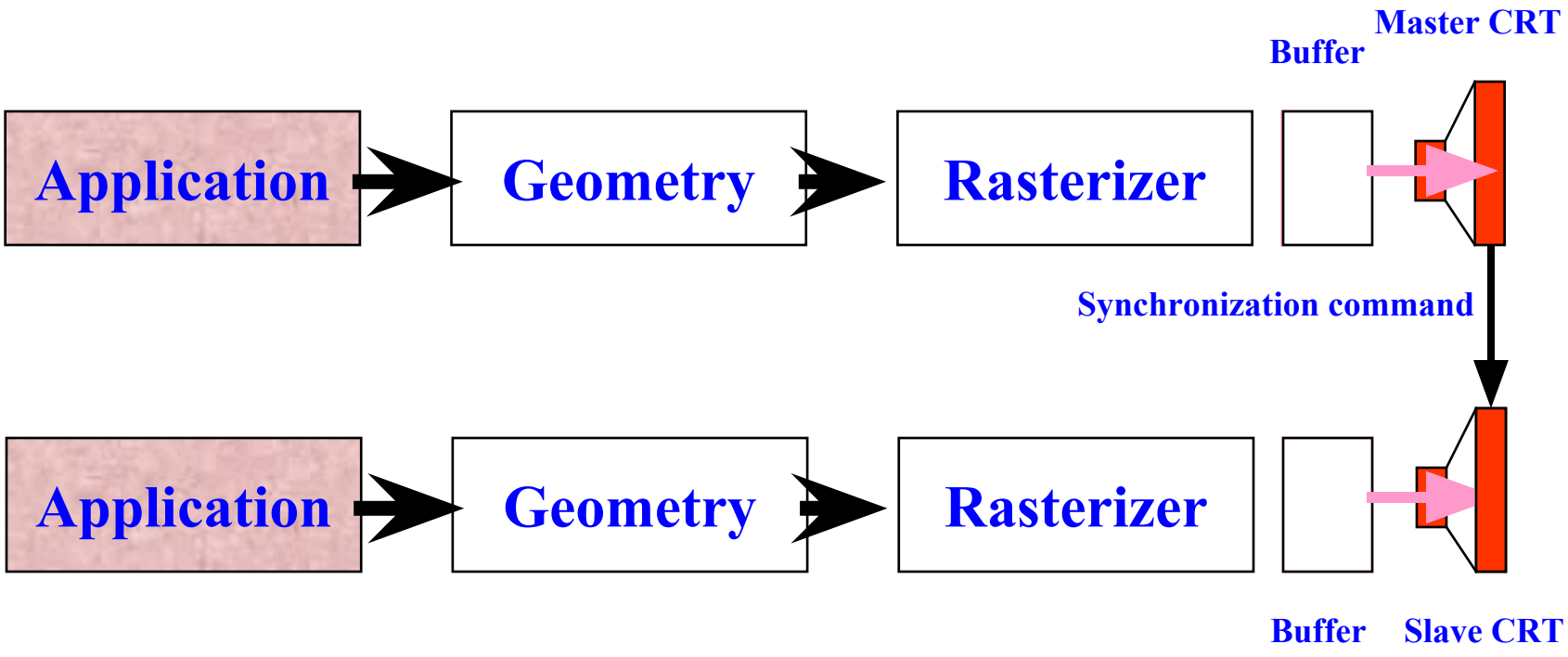
Synchronization of displays:

- ✓ ***frame buffer synchronized*** – system commands that frame buffer swapping starts at same time on different rendering pipes;
- ✓ does not work because swapping depends on electronic gun refresh - one buffer will swap up to 1/72 sec before the other.



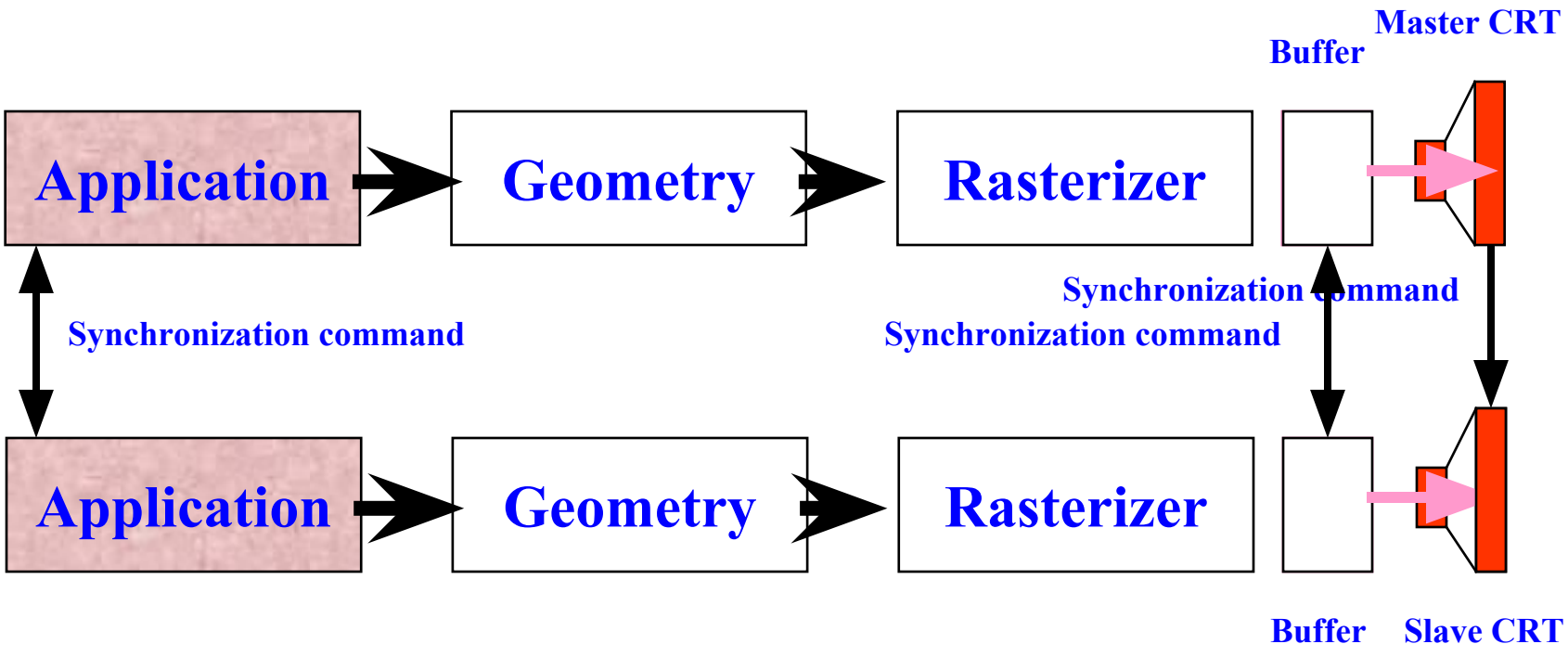
Synchronization of displays:

- ✓ ***video synchronized*** – system commands that CRT vertical beam starts at same time; one CRT becomes the "master"
- ✓ does not work if horizontal beam is not synchronized too (one line too many or too few).

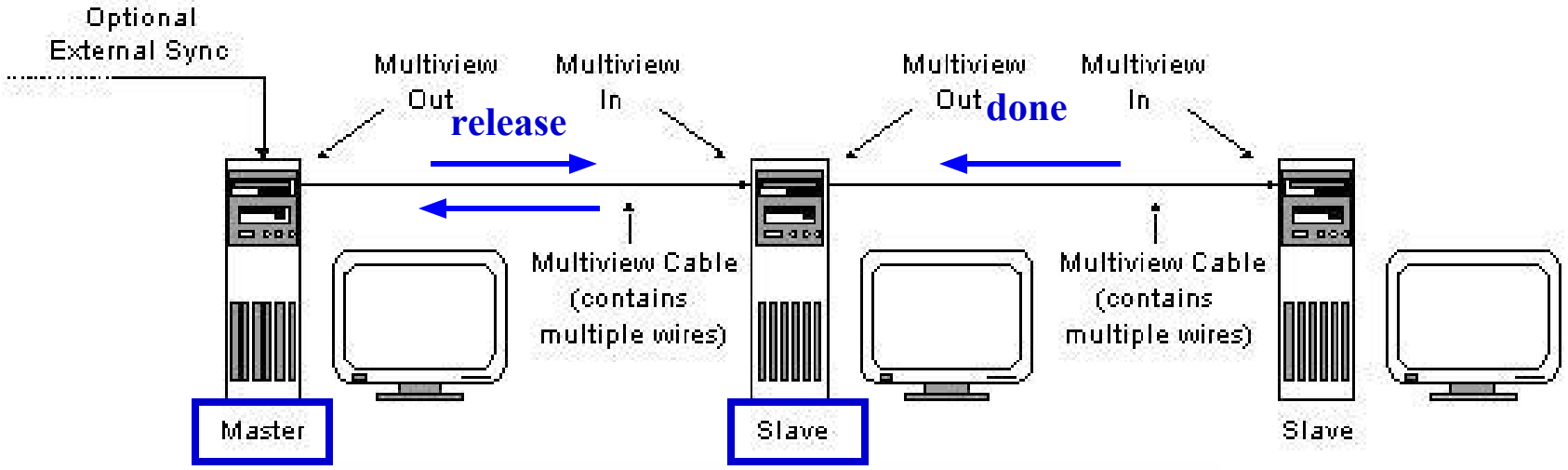


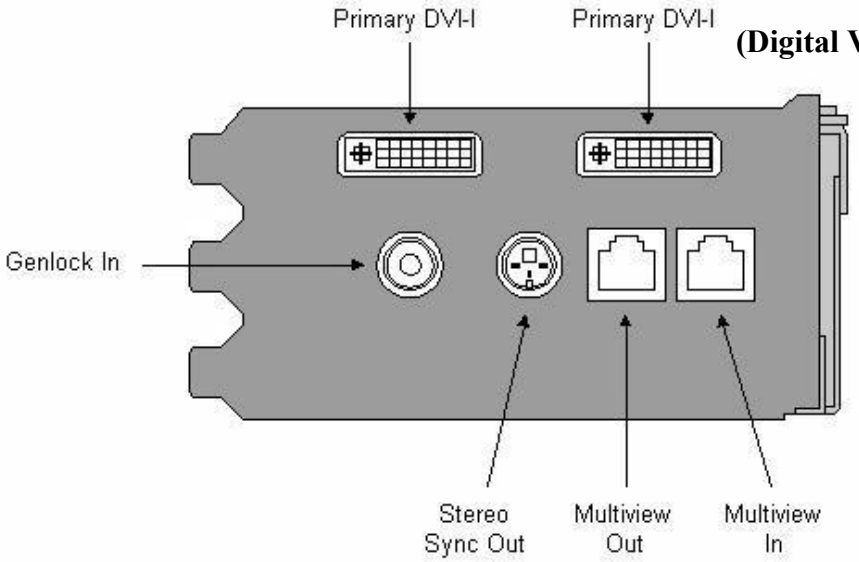
Synchronization of displays:

✓ Best method is to have software + buffer + video synchronization of the two (or more) rendering pipes



Video synchronized displays (three PCs)





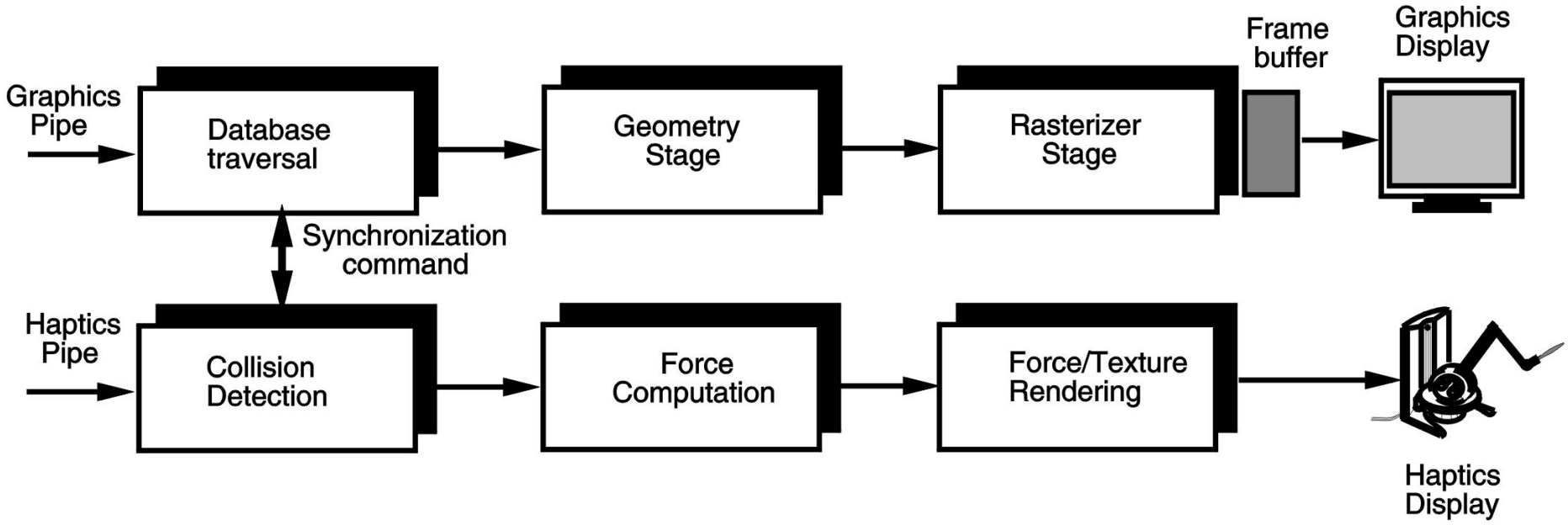
(Digital Video Interface- Video out)

Wildcat 4210

The ship's bridge simulator –shown here with eight of 12 out-the-window video channels all synchronized precisely using SyncLock™ and all swapping on the same frame using the Quantum3D™ Swaplock™.

Graphics and Haptics Pipeline Synchronization:

✓ Has to be done at the application stage to allow decoupling of the rendering stages (have vastly different output rates)



Graphics pipe and Haptics pipe