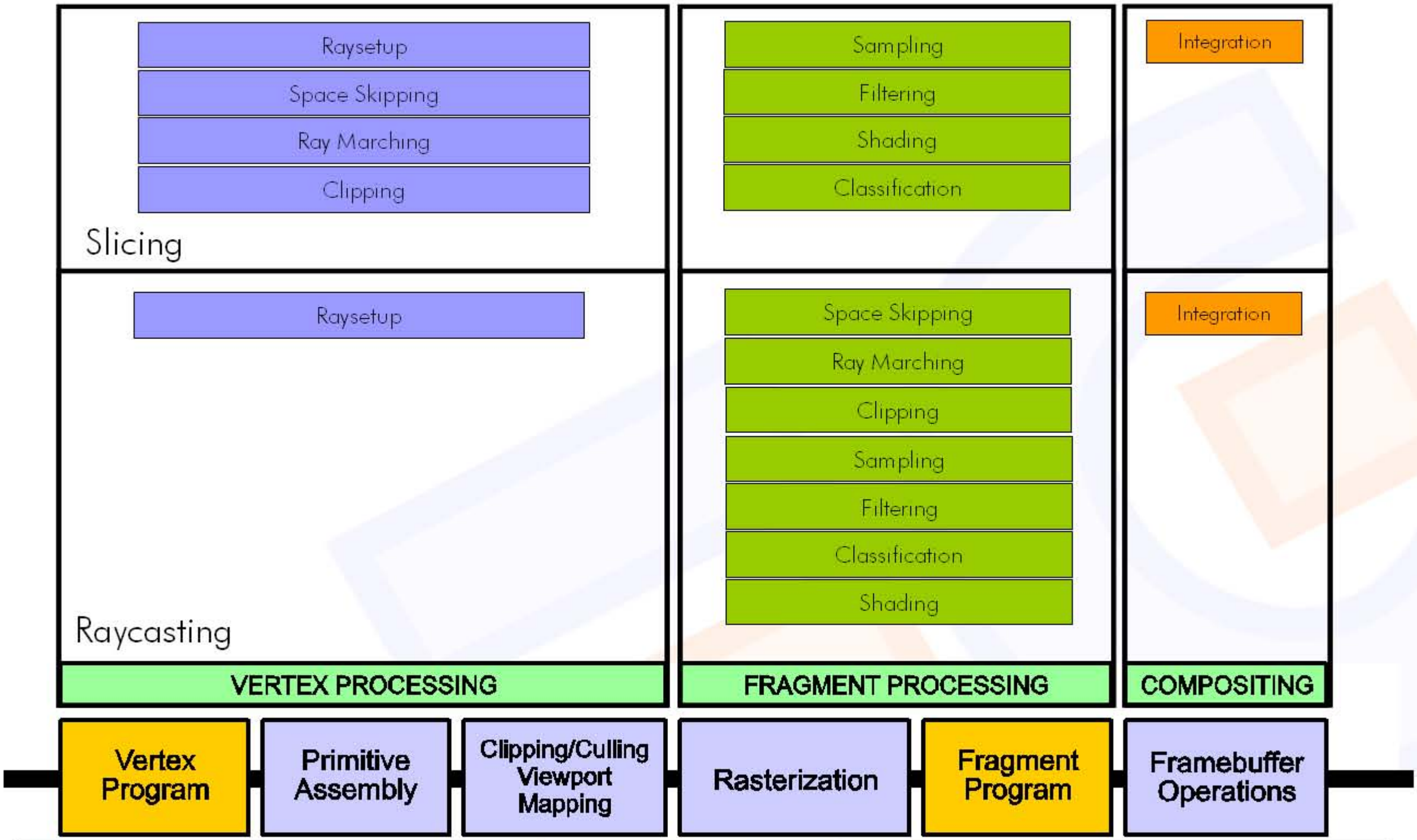
Real-Time Volume Graphics

[08] Improving Performance

GPU Pipeline Load



Fragment Processing Bound

Volume Rendering is usually fragment processing bound:

- Simple Example:
  - 1024x1024 Viewport
  - 512x512x512 Volume
  - Orthographic Projection, full zoom
  - 512 Samples along each ray, 512 slices

8 vertices (bounding box)	= 8 Vertices

or

512 x 4 vertices (quads)	= 1024 Vertices
1024 x 1024 x 512 Samples	= 512 MSamples

Fragment Processing Power

Single cycle fragment program performance:

- NVIDIA GeForce 7900 GTX
  - 24 (pipelines) x 650 MHz = 15.6 GPix/s

- ATI Radeon 1900 XTX
  - 16 (pipelines) x 650 MHz = 10.4 GPix/s

- NVShaderPerf:

No Shading, Post-Interpolative classification:
Target: GeForce 7800 GT (G70) :: Unified Compiler: v81.95
Cycles: 2.00 :: R Regs Used: 1 :: R Regs Max Index (0 based): 0
Pixel throughput (assuming 1 cycle texture lookup) 4.80 GP/s

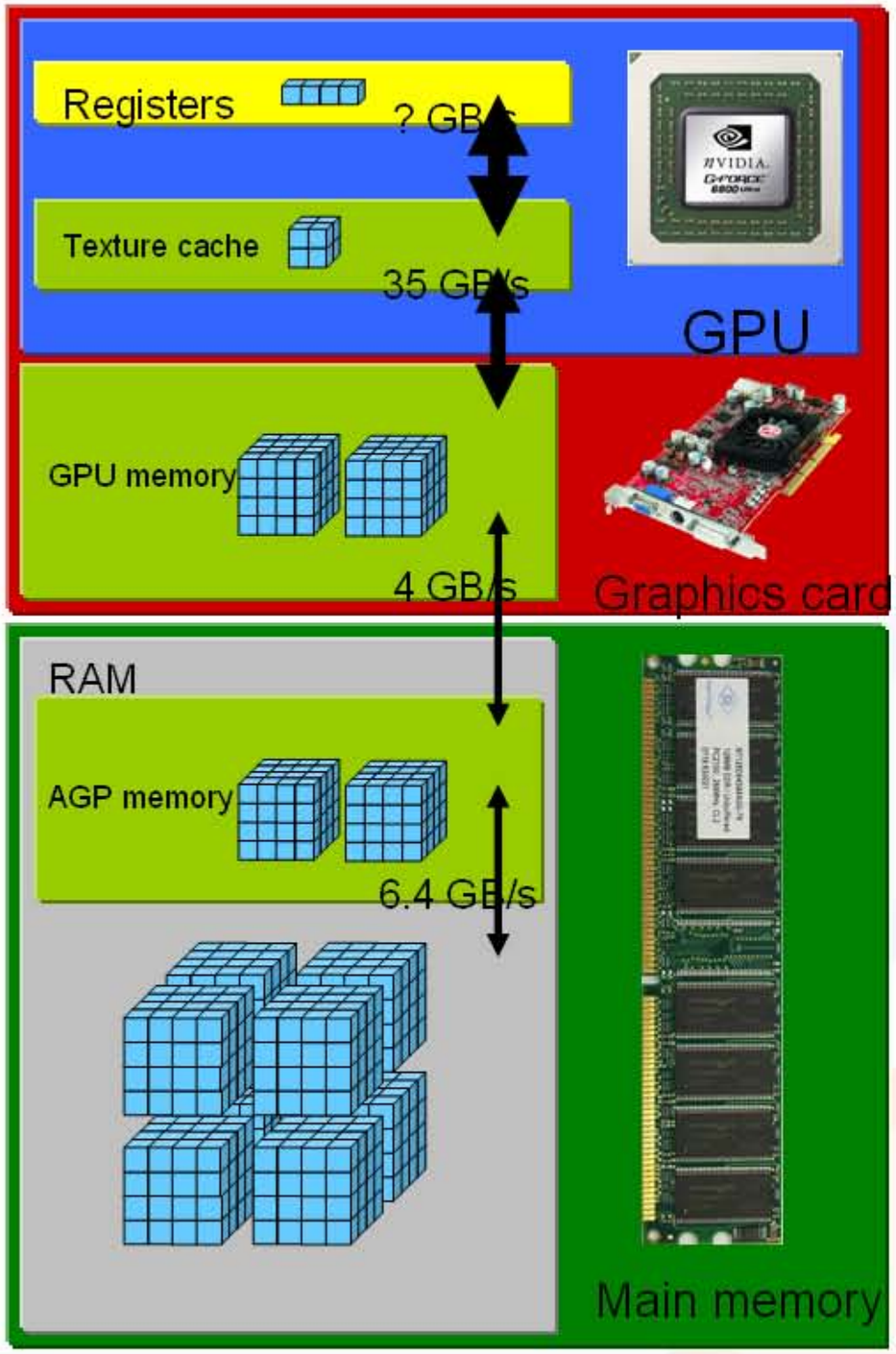
With Shading, Pre-computed Gradients, Post-Interpolative classification :
Target: GeForce 7800 GT (G70) :: Unified Compiler: v81.95
Cycles: 7.00 :: R Regs Used: 2 :: R Regs Max Index (0 based): 1
Pixel throughput (assuming 1 cycle texture lookup) 1.37 GP/s

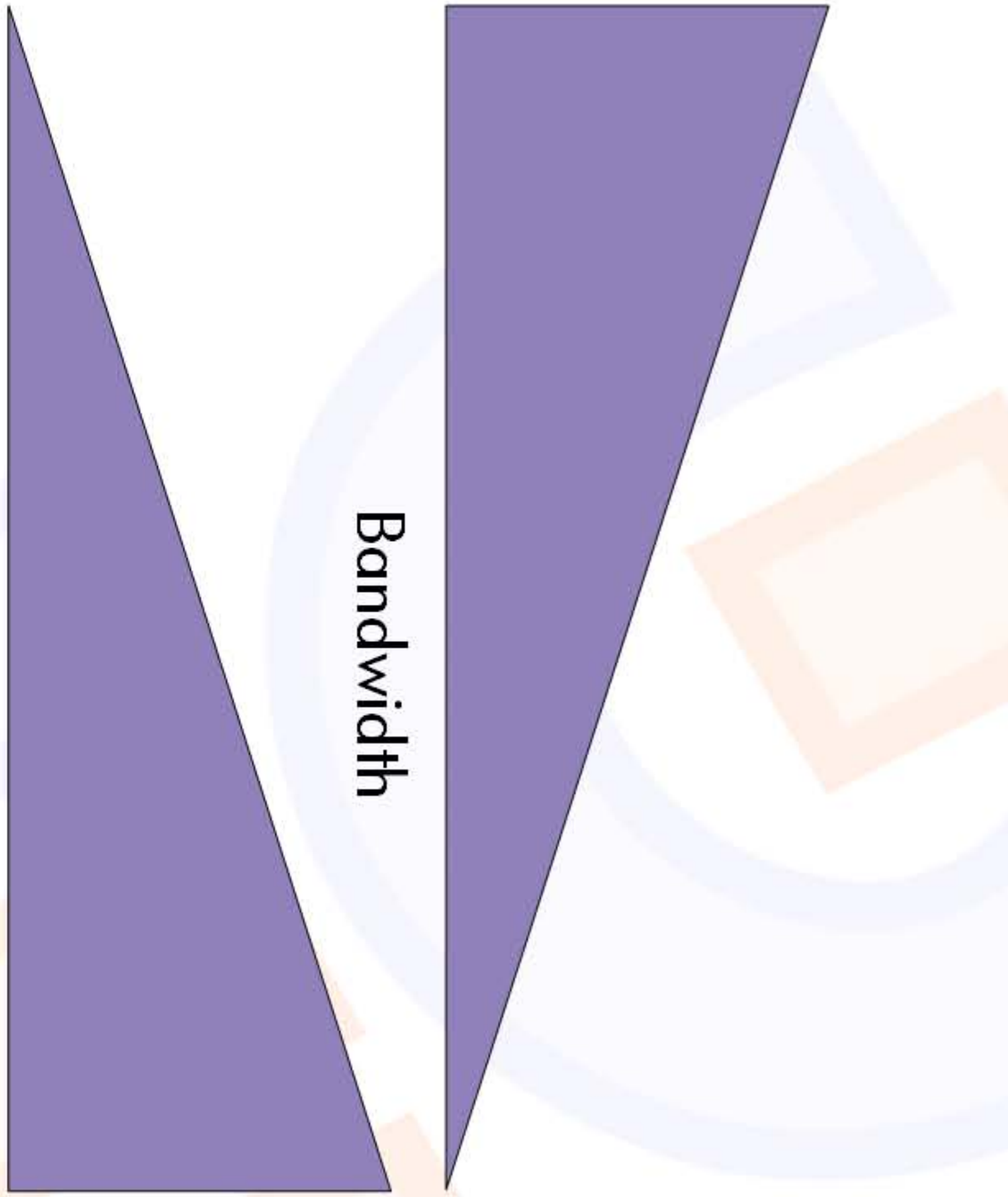
With Shading, On-the-fly Gradients, Post-Interpolative classification:
Target: GeForce 7800 GT (G70) :: Unified Compiler: v81.95
Cycles: 13.00 :: R Regs Used: 3 :: R Regs Max Index (0 based): 2
Pixel throughput (assuming 1 cycle texture lookup) 738.46 MP/s

Memory Bandwidth

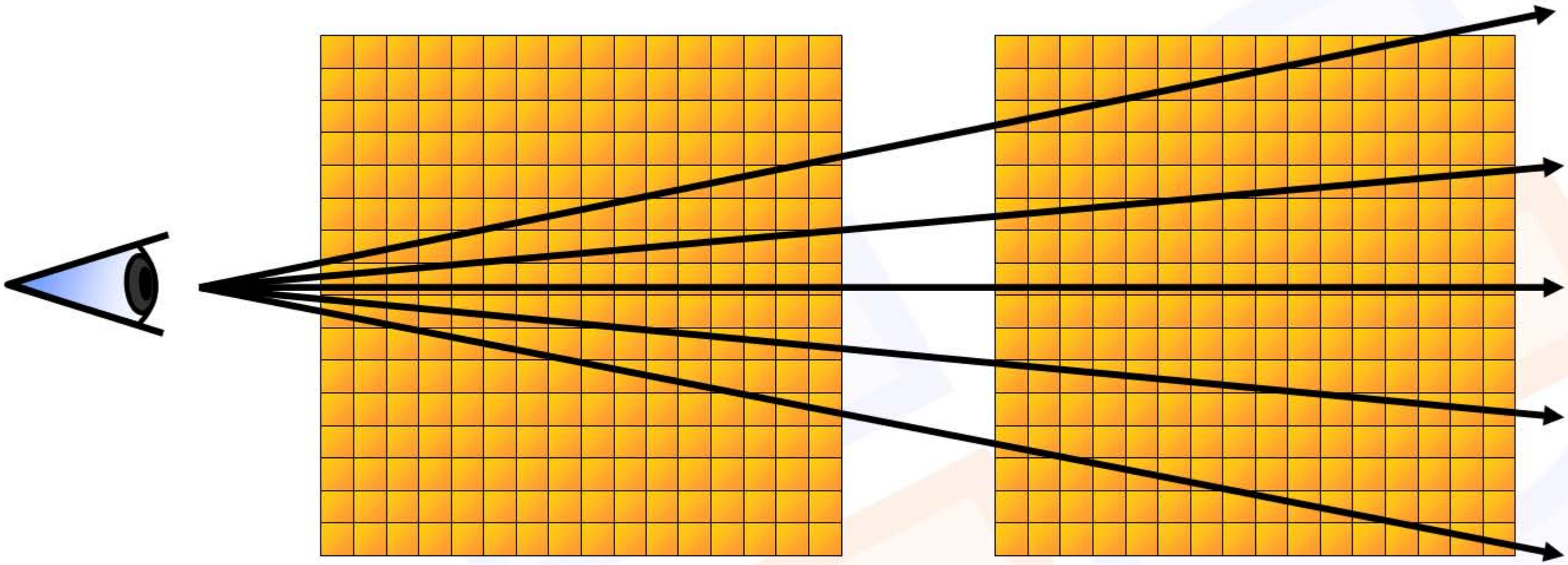
- NVIDIA GeForce 7900 GTX
  - 32 byte (256 bit) x 2 (DDR) x 800 MHz = 51.2 Gbyte/s
- ATI Radeon 1900 XTX
  - 32 byte (256 bit) x 2 (DDR) x 775 MHz = 49.6 Gbyte/s
- But:
  - Peak rate when accessing memory linearly (dependent texture operations are bad)
  - Multiple data values for filtering required (8 for trilinear)
  - Many data values are fetched multiple times (cache miss)
  - On-the-fly gradients require neighbor information

Memory Latency

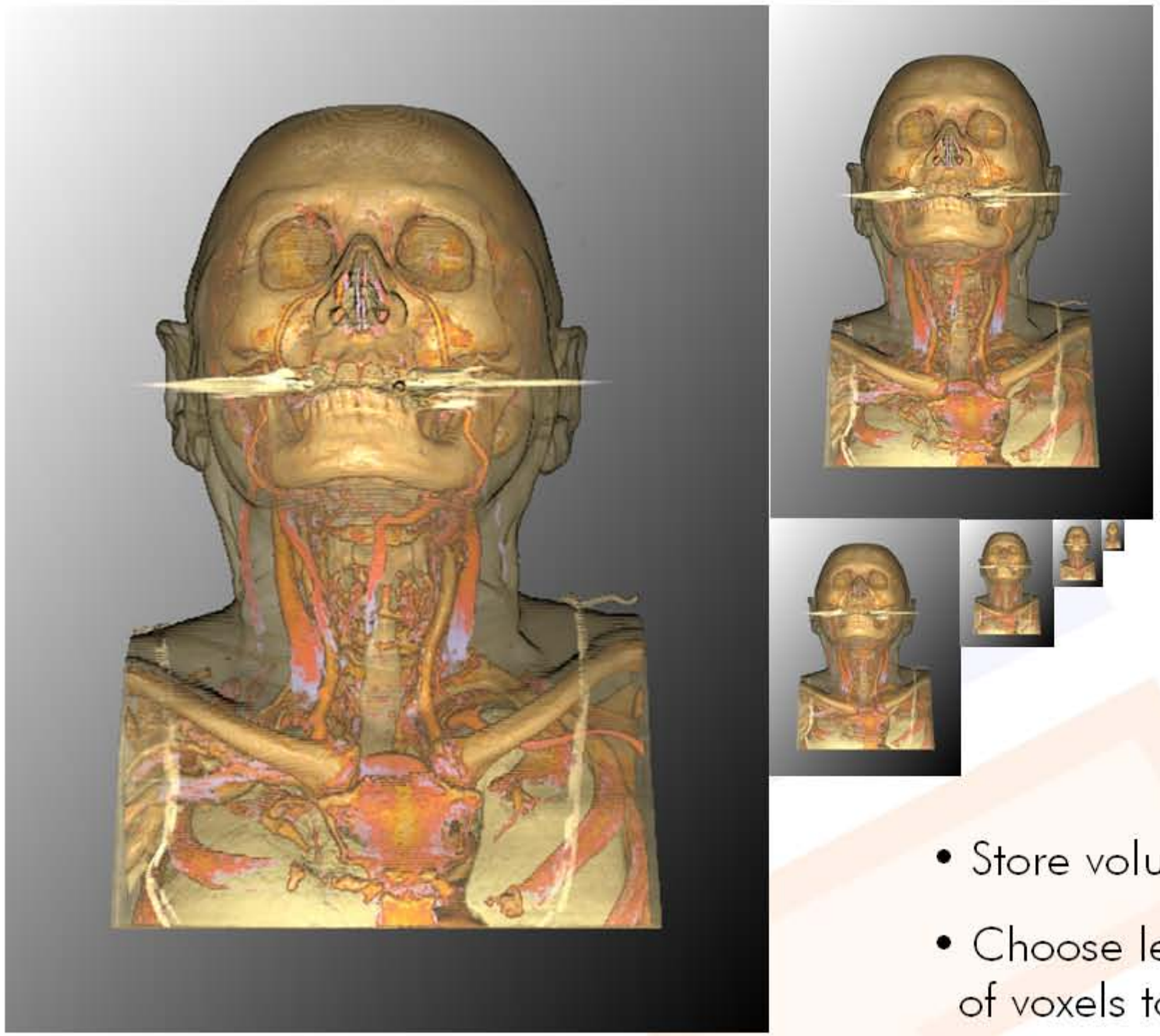




Mipmapping

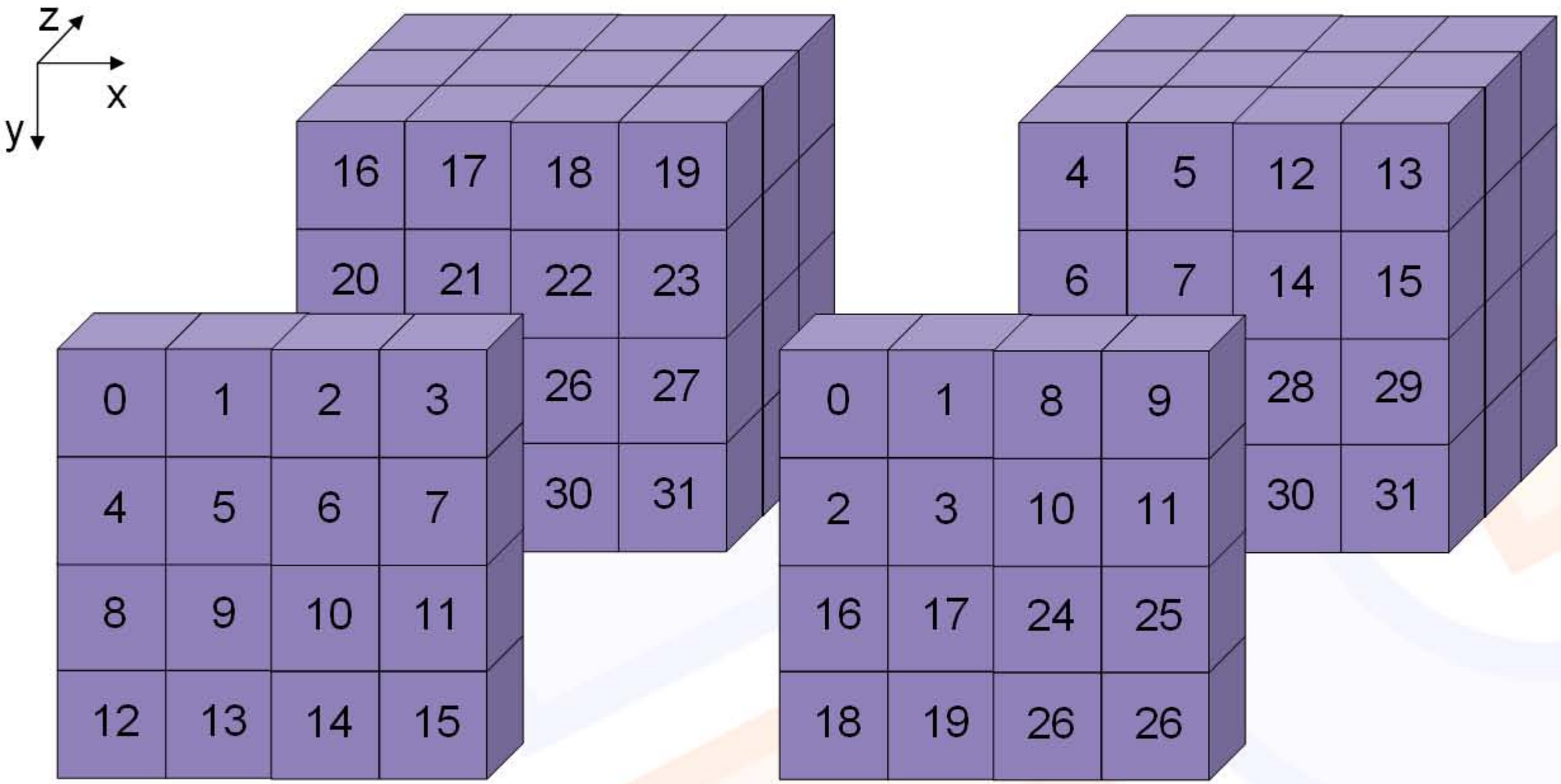


Mipmapping



- Store volume at multiple resolutions
- Choose level dependent on projection of voxels to pixels

Block-based Volume Swizzling

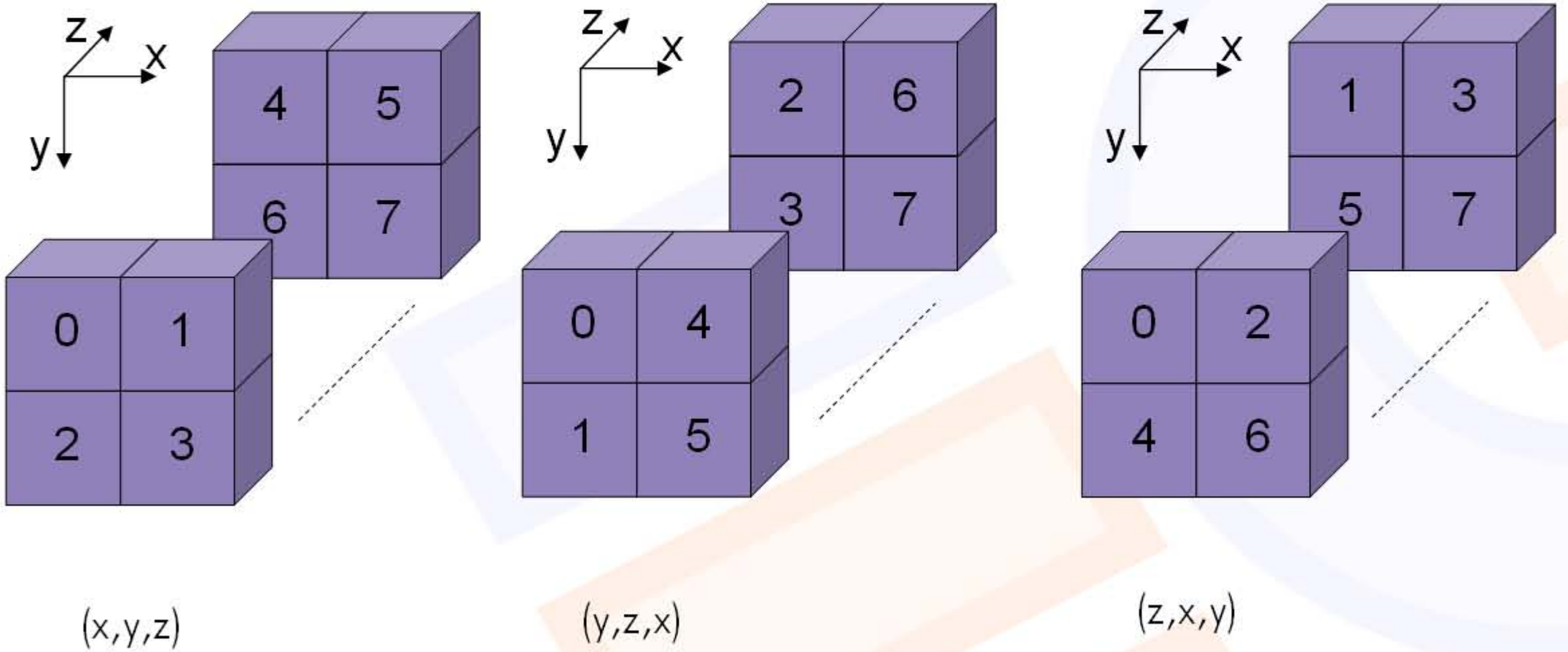


linear

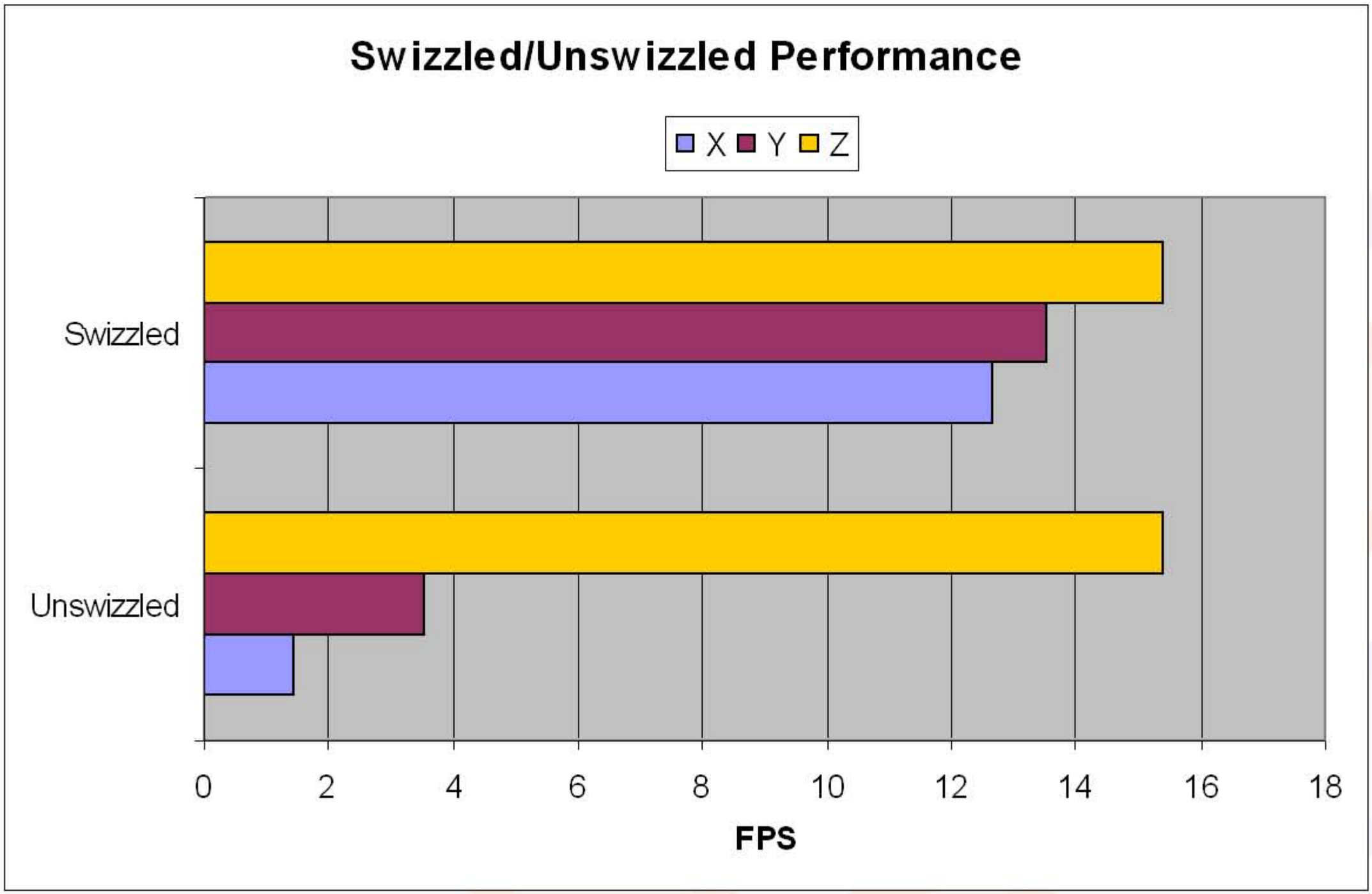
swizzled

Multioriented Volume Swizzling

Weiskopf et al., *"Maintaining Constant Frame Rates in 3D Texture-based Volume Rendering"*, CGI 2004



Volume Swizzling

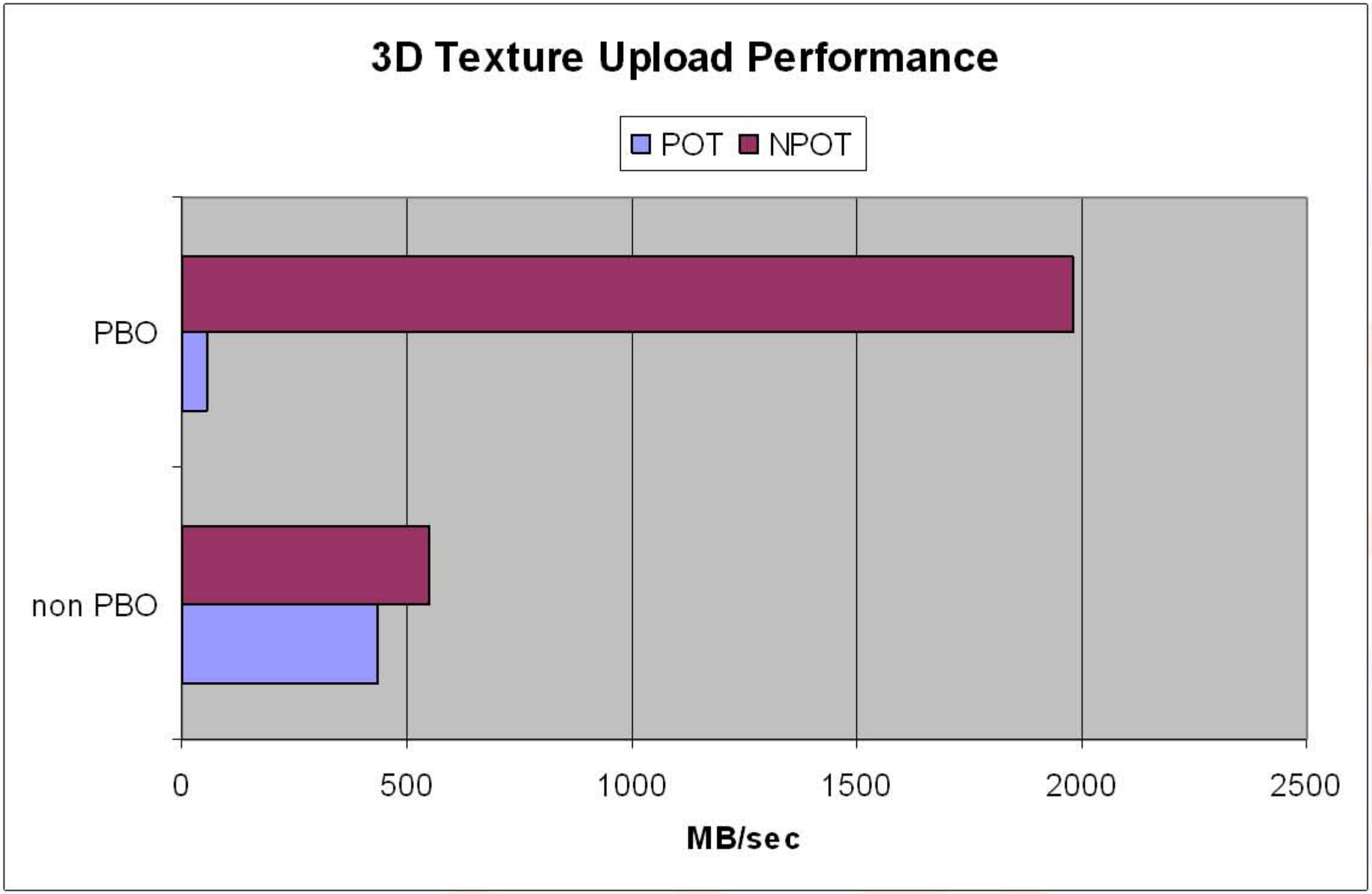


Asynchronous Data Upload

- Volume data size > GPU memory size
  - data stored in main memory
  - transfer per frame via PCIe to GPU (4 GB/sec)

- Pixel buffer objects (PBO)
  - From AGP/PCIe memory
  - Asynchronous(CPU does not block, GPU does block)
  - Data must be in GPU-native format
  - NPOT 3D textures are not swizzled on NVIDIA GPUs

Asynchronous Data Upload

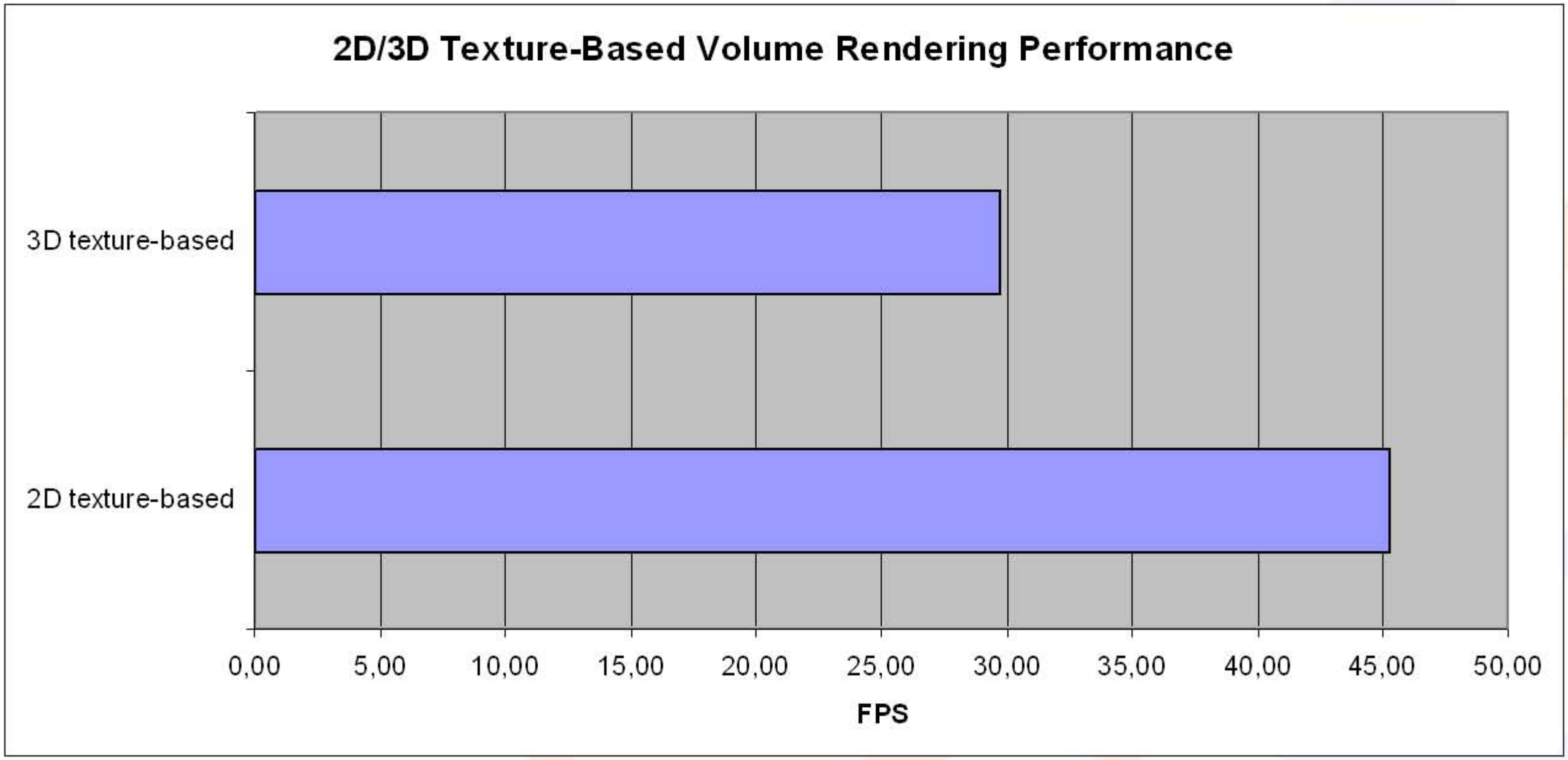


Bilinear Filtering

Use 2D textures instead of 3D textures:

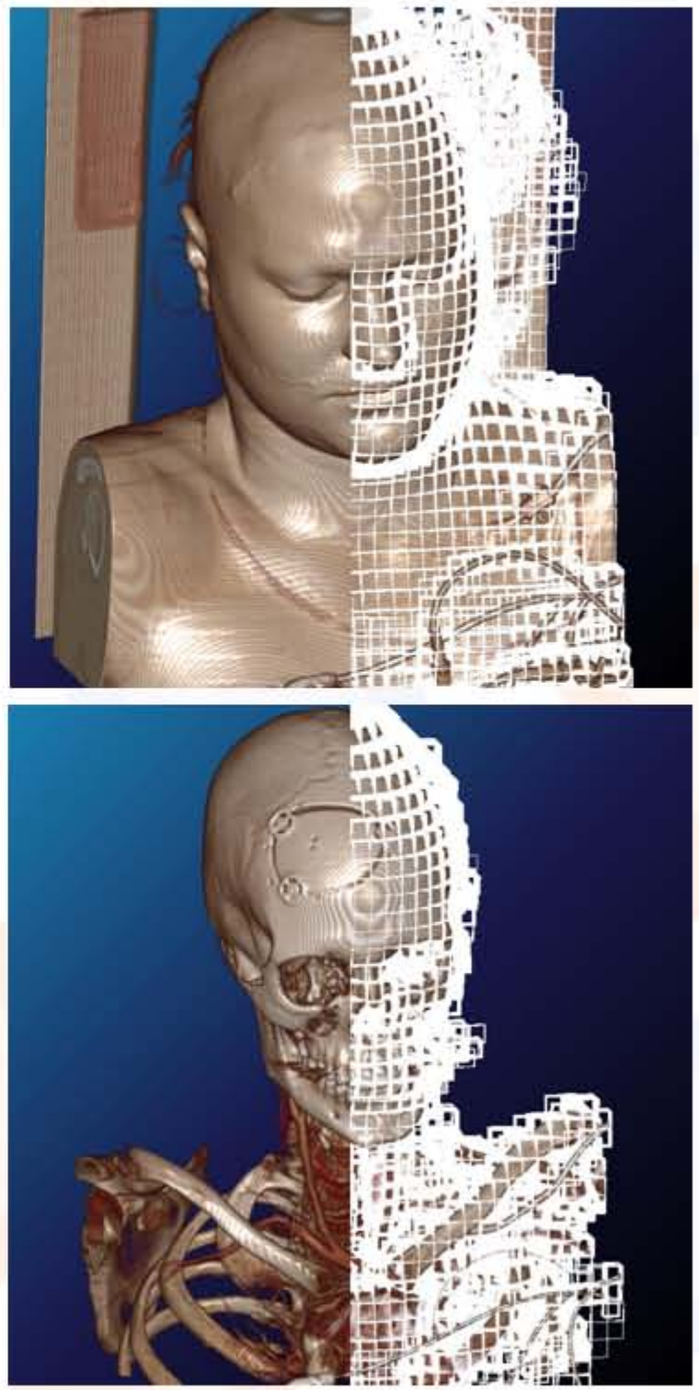
- Only bilinear filtering
  - 4 instead of 8 data values required for filtering
  - less memory bandwidth
- Trilinear filtering only for intermediate slices (see Part 2, 2D Multi-Texture-based Approch)
- Better cache utilization
  - GPUs better optimized for 2D textures
  - Smaller working set

Bilinear Filtering

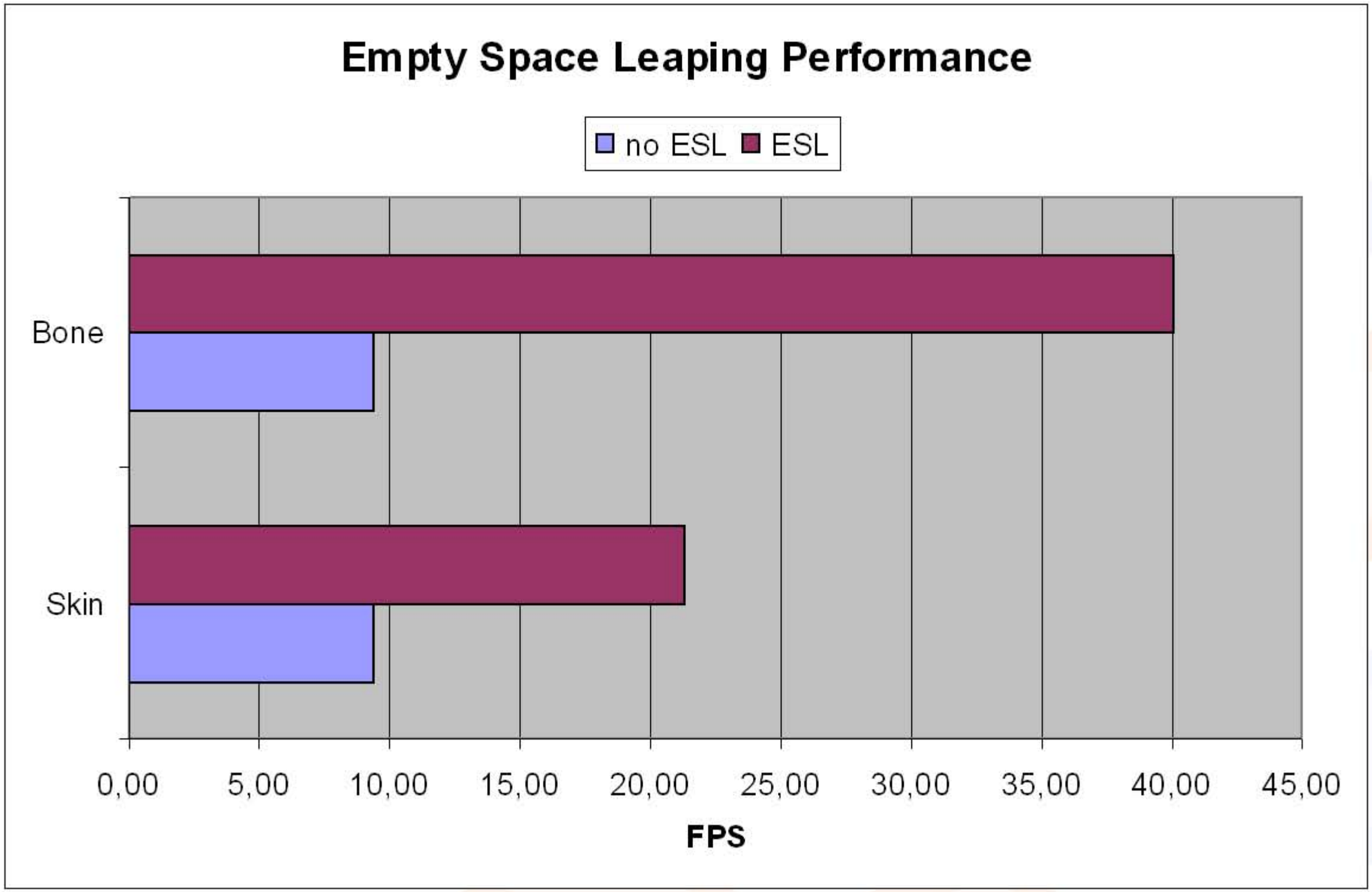


Empty Space Leaping

- Don't access memory that contains no data
  - Subdivide volume into blocks
  - Store minimum/maximum value per block
  - Check with transfer function and min/max if block is non-empty
- Render block only if not empty



Empty Space Leaping



Occlusion Culling

Block-based culling:

- Before slicing or raycasting each block
  - Disable color and depth writes
  - Render front faces of block with framebuffer texture
  - Discard fragments with alpha larger than threshold (alpha test)
  - Use ARB_occlusion_query to count fragments that pass the test
- Slice or raycast block only if fragment count > 0
  - else all pixels in block are occluded and block can be culled

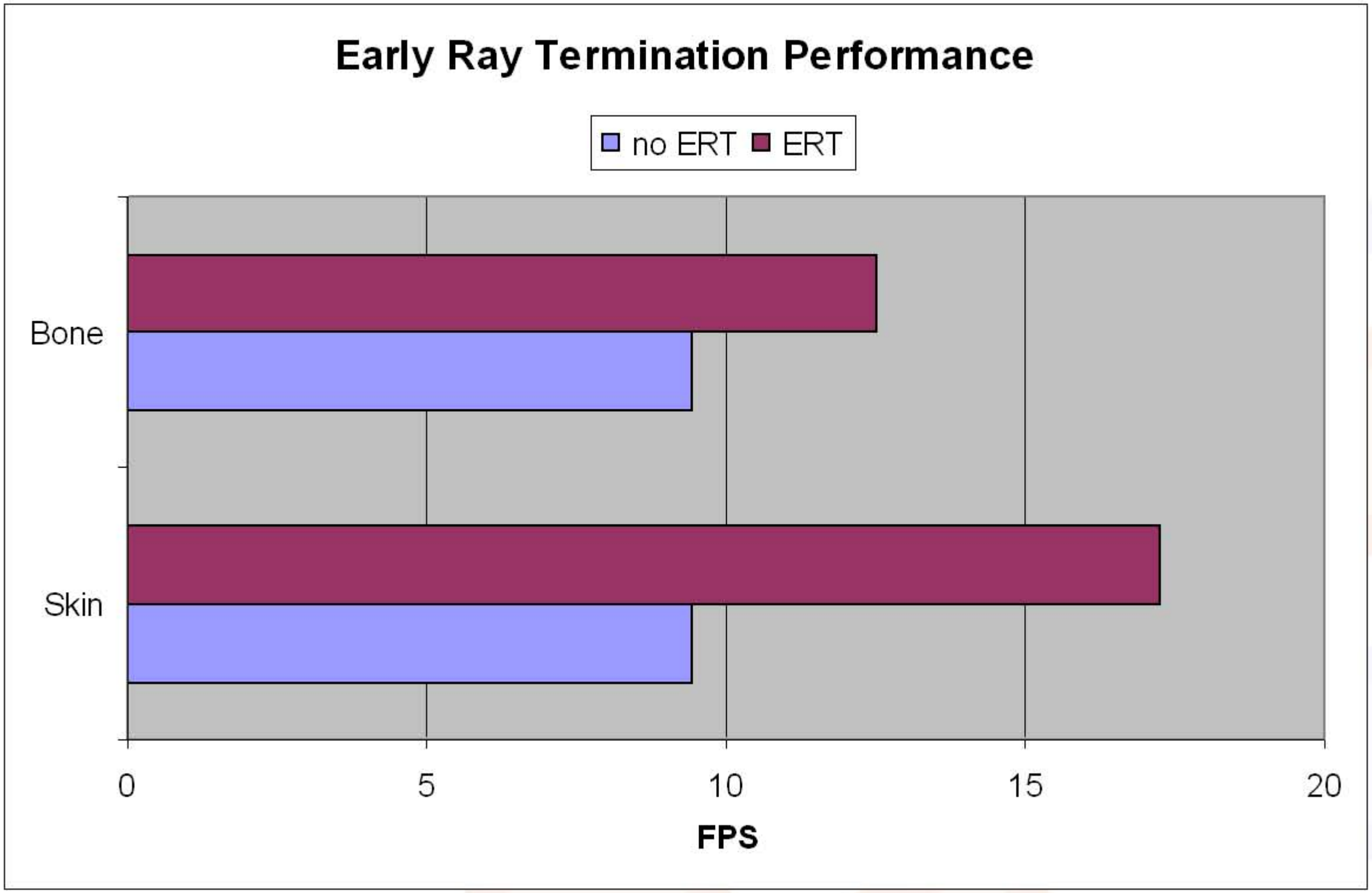
Ray Termination

Krueger/Westermann – Acceleration Techniques for GPU-based Volume Rendering, IEEE Visualization 2003

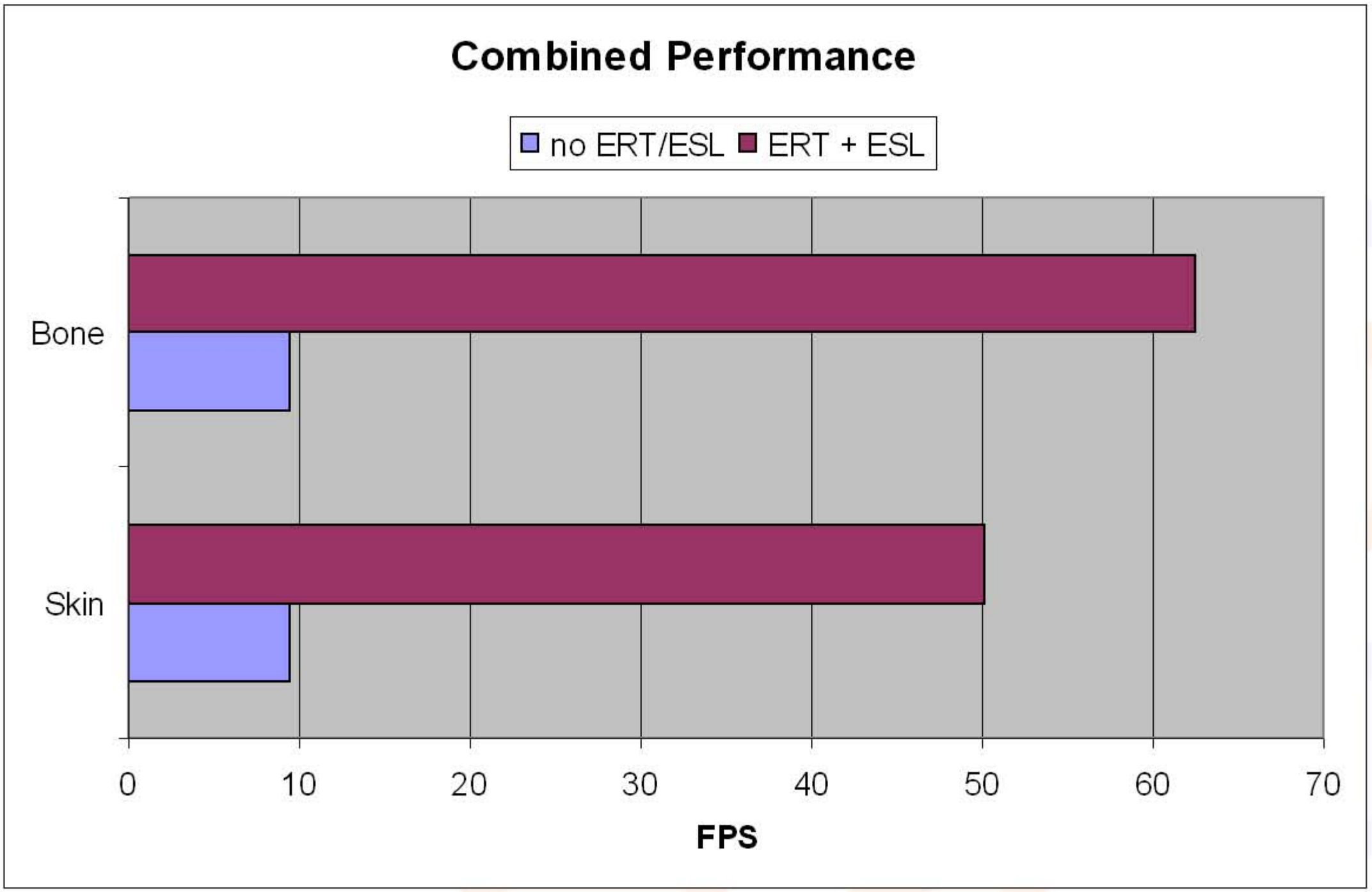
Pixel-based culling:

- Terminate rays(pixels) that have accumulated maximum opacity
- Termination is done in a separate pass
- Render bounding box with framebuffer as texture
- Check for each pixel if alpha above threshold (alpha test, branching disables early-z)
- Set z value if above threshold
- Requires early-z test

Ray Termination



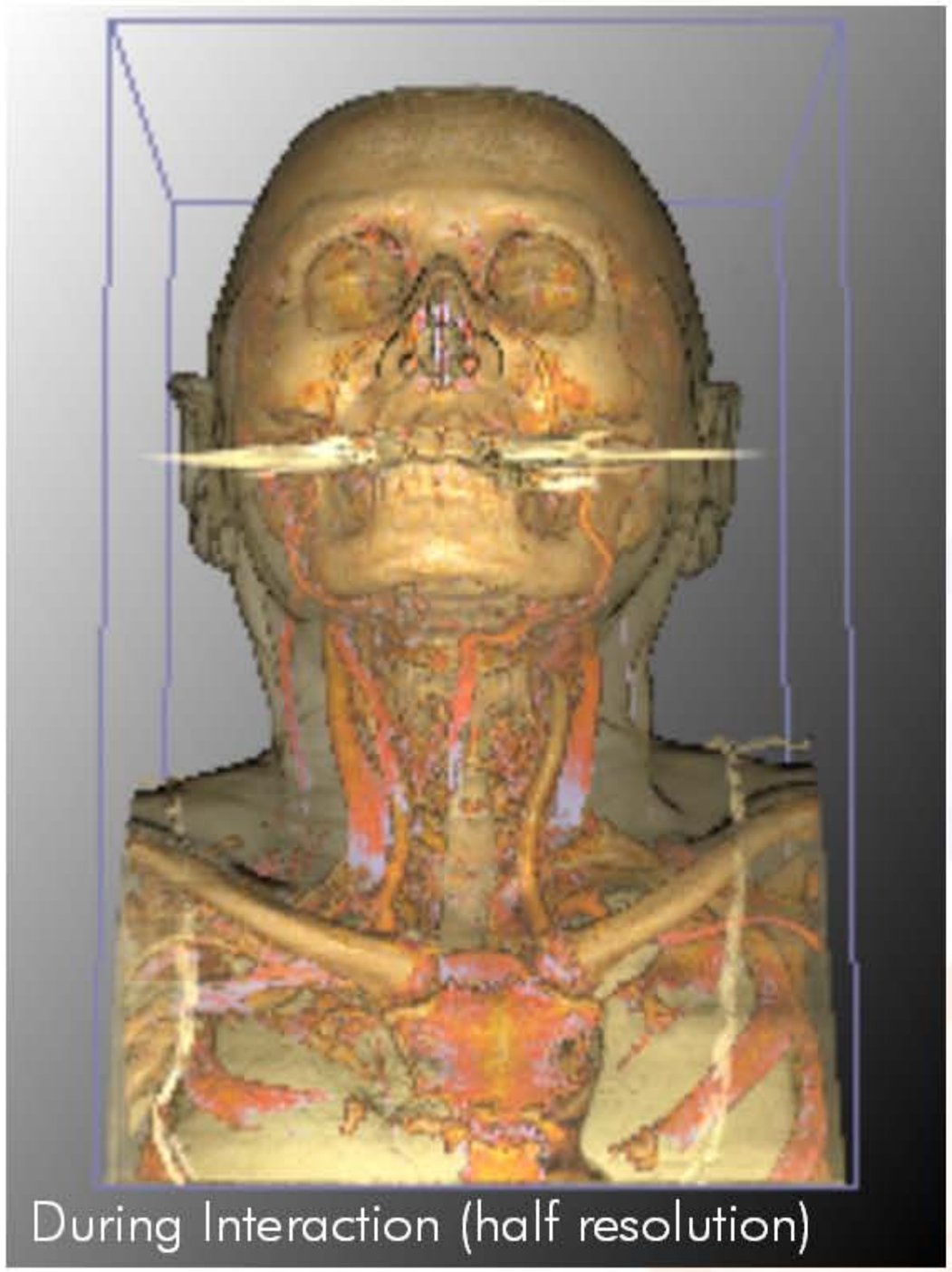
ERT + ESL


Combined Performance
no ERT/ESL
ERT + ESL
Bone
Skin
0
10
20
30
40
50
60
70
FPS

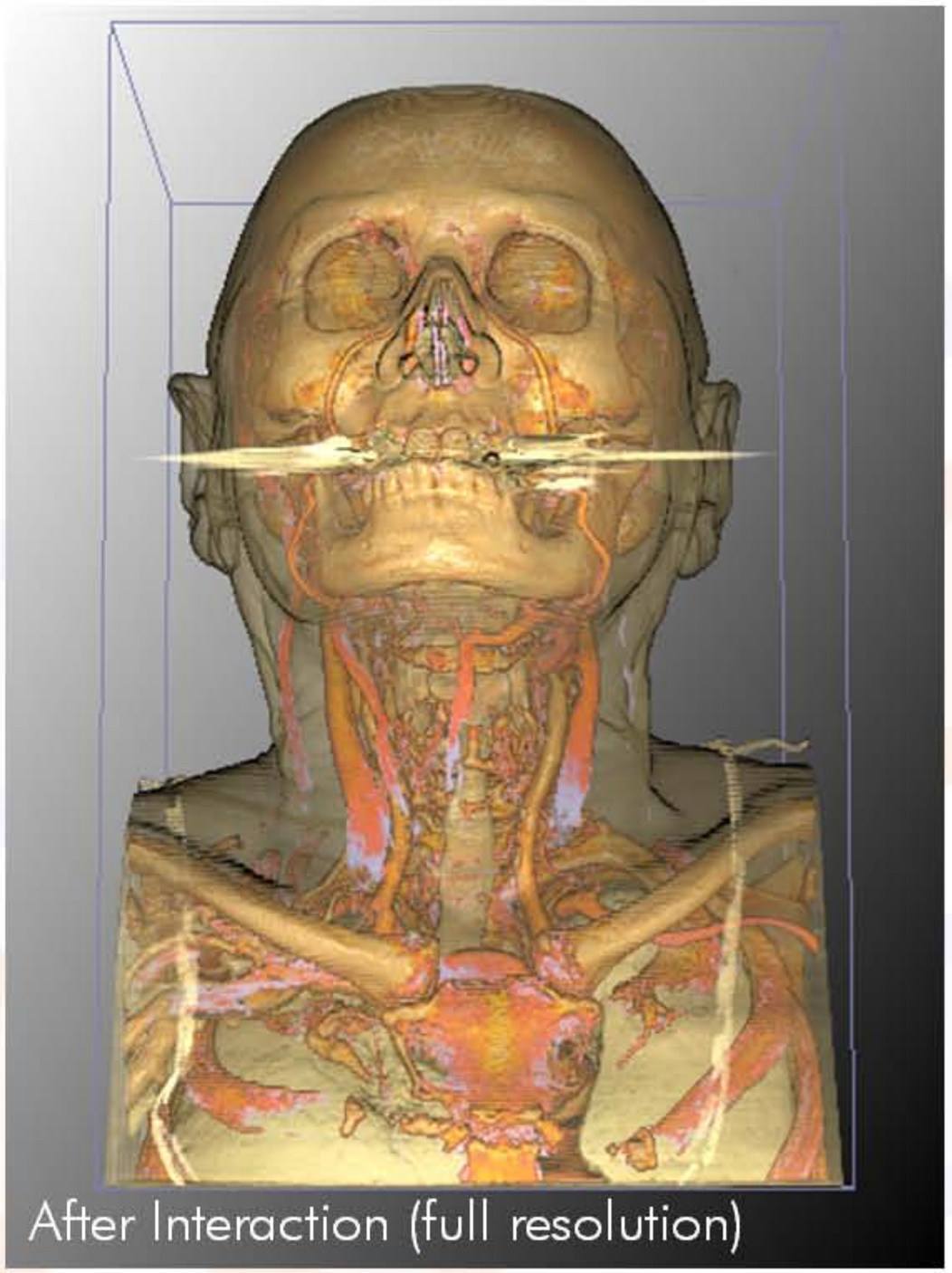
Deferred Shading

- Shade selectively
  - Shade only first volume boundary
    - Two passes: volume pass + image space pass
    - 1st Pass: Render unshaded + depth
    - 2nd Pass: Compute volume coordinates from depth and shade
  - Shade only if alpha is above a threshold
    - Two passes for each slice
    - 1st Pass: Render unshaded in first pass
    - 1st Pass: Set z/alpha where alpha is above threshold
    - 2nd Pass: Use early-z/stencil test
    - 2nd Pass: shade where z/alpha test succeed
    - Requires early-z/stencil test

Image Downscaling

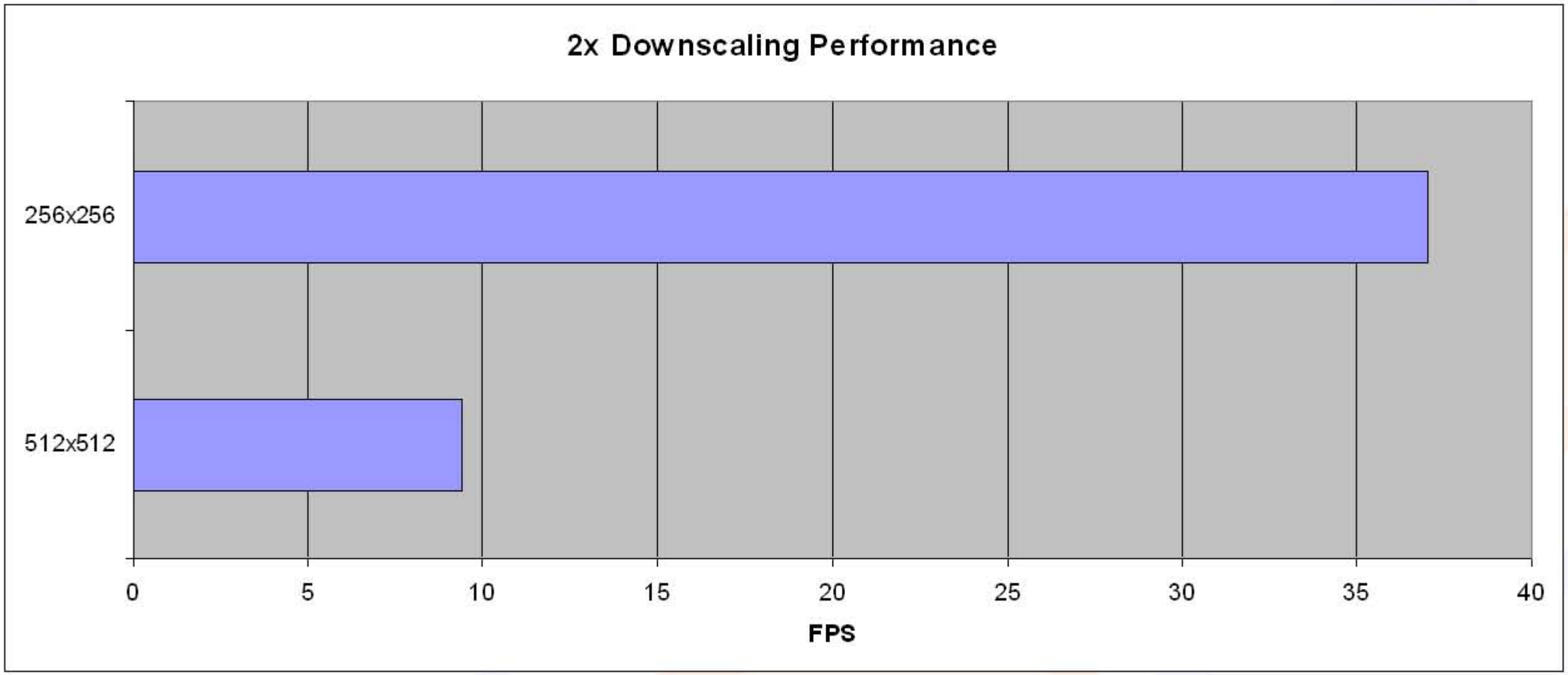


During Interaction (half resolution)



After Interaction (full resolution)

Image Downscaling



Guidelines

- Balance the pipeline
  - Slicing better than raycasting
  - Might change with unified shaders in future GPUs
- Cull, cull, cull
- Keep data close to the GPU, Improve memory access
- Benchmark

Tools from GPU vendors

- NVIDIA
  - NVShaderPerf: shader performance metrics
  - NVPerfKit: instrumentation driver
  - NVPerfHUD: Real-Time statistics on top of DX Appl.
- ATI
  - Plugin for MS PIX: Performance Investigator for DirectX