Workload Characterization of 3D Games

Jordi Roca, Victor Moya, Carlos González, Chema Solis, Agustín Fernandez and Roger Espasa (Intel DEG Barcelona)

Computer Architecture Department

Outline

- Introduction
- Game selection & stats gathering
- Game analysis
  - System → GPU traffic
  - Primitive culling efficiency
  - Rasterization pipeline
  - Fragment shading & texturing
  - Memory usage
- Conclusions

Introduction

- Games and GPU evolve fast
- GPUs cater for game demands:
  - Better effects (flexible programming models)
  - Higher fill-rate (more processing power)
  - Higher quality (HDR, MSAA, AF)
- Games highly tuned to released GPUs
- New characterization needed for every Game and GPU generation.

Outline

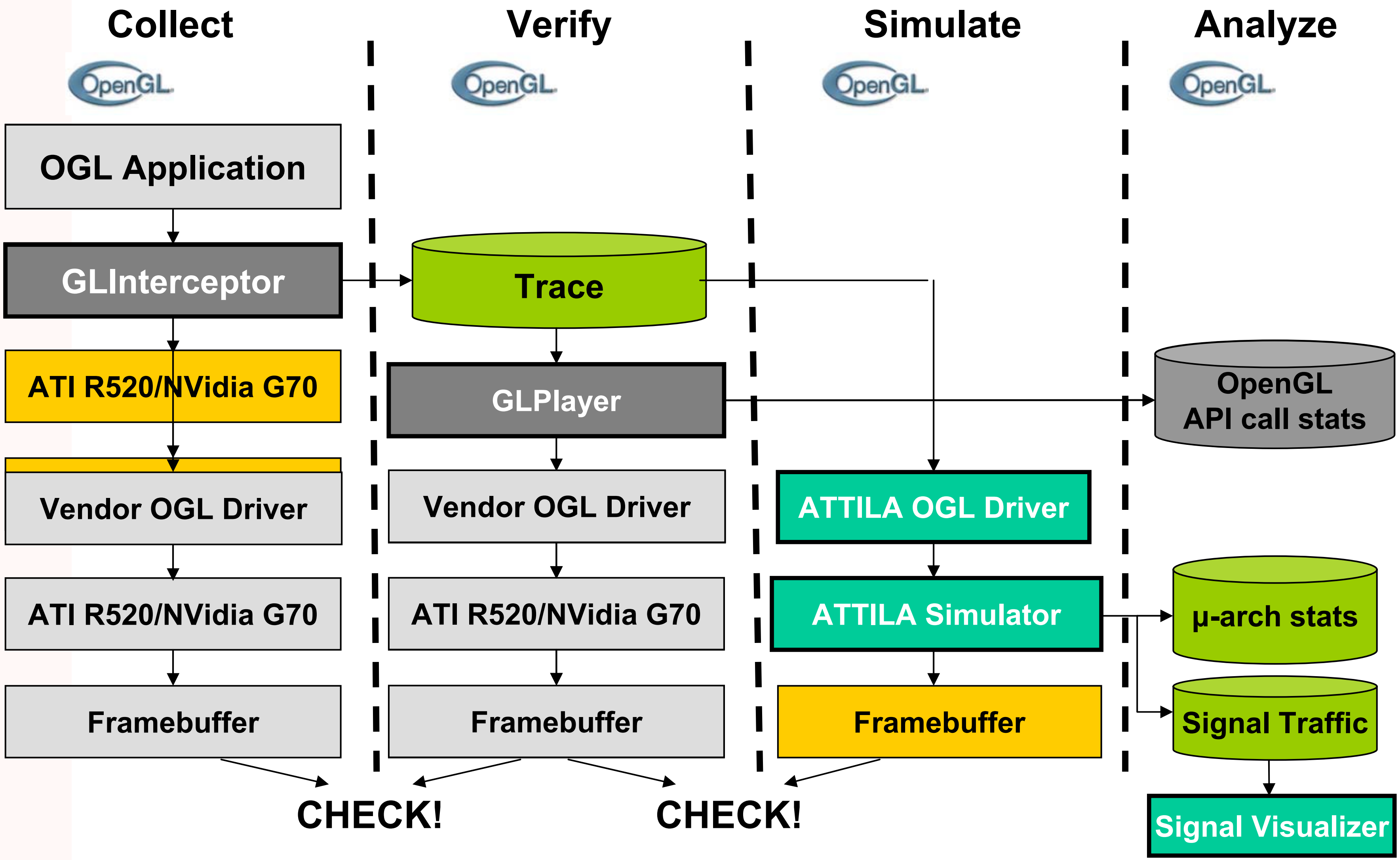
- Introduction
- **Game selection & stats gathering**
- Game analysis
  - System → GPU traffic
  - Primitive culling efficiency
  - Rasterization pipeline
  - Fragment shading & texturing
  - Memory usage
- Conclusions

Game workload selection

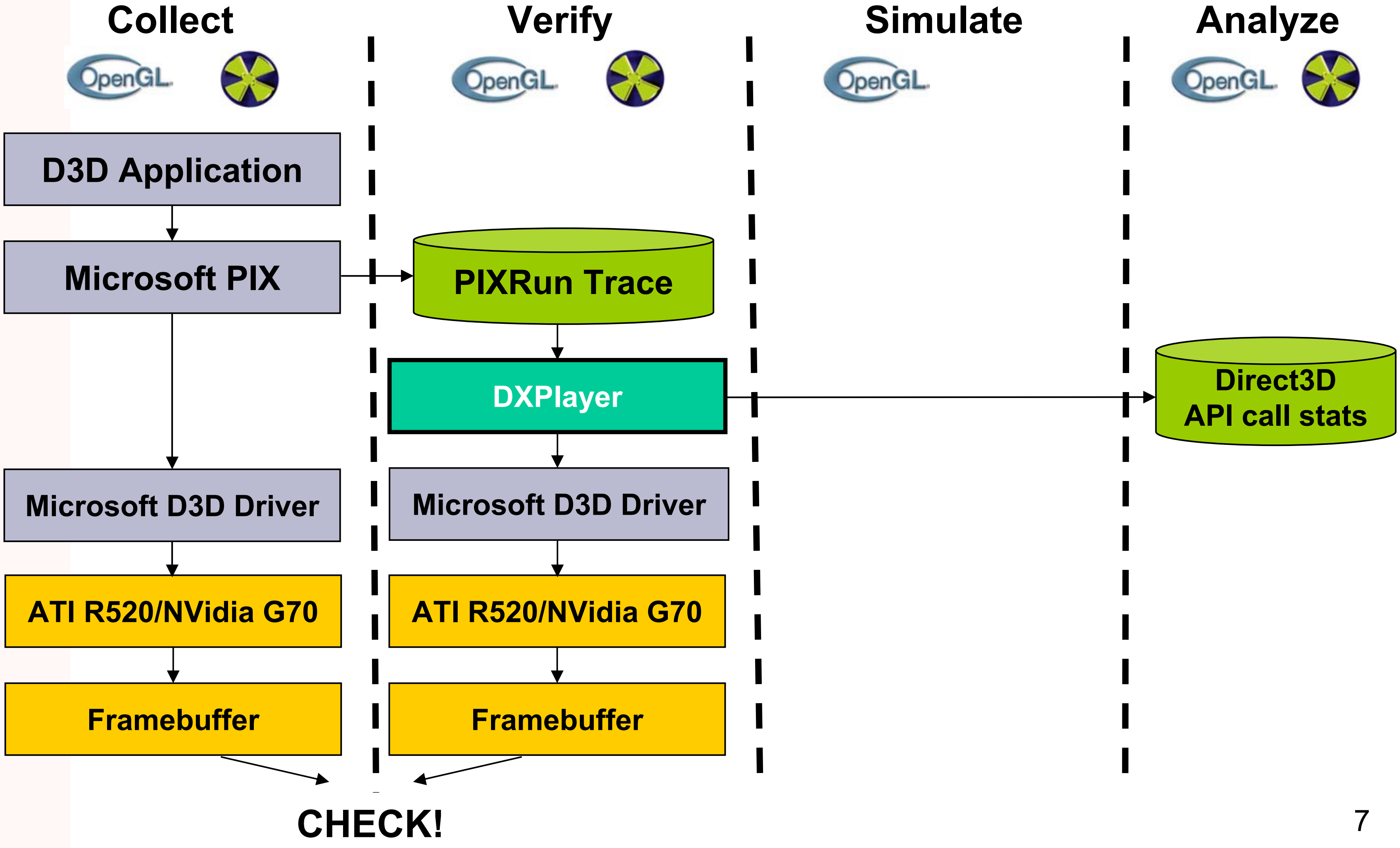
Game/Timedemo	Frames	Duration at 30 fps	Texture Quality	Aniso Level	Shaders	Graphics API	Engine	Release Date
UT2004/Primeval	1992	1’ 06”	High/Aniso	16X	NO	OpenGL	Unreal 2.5	Mar 2004
Doom3/trdemo1	3464	1’ 55”	High/Aniso	16X	YES	OpenGL	Doom3	Aug 2004
Doom3/trdemo2	3990	2’ 13”	High/Aniso	16X	YES			
Quake4/demo4	2976	1’ 39”	High/Aniso	16X	YES	OpenGL	Doom3	Oct 2005
Quake4/guru5	3081	1’ 43”	High/Aniso	16X	YES			
Riddick/MainFrame	1629	0’ 54”	High/Trilinear	-	YES	OpenGL	Starbreeze	Dec 2004
Riddick/PrisonArea	2310	1’ 17”	High/Trilinear	-	YES			
FEAR/built-in demo	576	0’ 19”	High/Aniso	16X	YES	Direct3D	Monolith	Oct 2005
FEAR/interval2	2102	1’ 10”	High/Aniso	16X	YES			
Half Life 2 LC/built-in	1805	1’ 00”	High/Aniso	16X	YES	Direct3D	Valve Source	Oct 2005
Oblivion/Anvil Castle	2620	1’ 27”	High/Trilinear	-	YES	Direct3D	Gamebryo	Mar 2006
Splinter Cell 3/first level	2970	1’ 39”	High/Aniso	16X	YES	Direct3D	Unreal 2.5++	Mar 2005

- Resolution: 1024x768

Statistics environment (OpenGL)



Statistics environment (Direct3D)



Outline

- Introduction
- Game selection & stats gathering
- **Game analysis**
  - **System → GPU traffic**
  - Primitive culling efficiency
  - Rasterization pipeline
  - Fragment shading & texturing
  - Memory usage
- Conclusions

System → GPU traffic

	Old games (Voodoo)	New games (GeForce)
Vertex processing	Done in CPU	Done In GPU (T&L)
Vertex data communication	Every frame	At startup
Vertex data storage	System memory	Local GDDR memory
Rendering action	Sends transformed data	Sends indices to data to transform
Proper analysis	**Vertex data BW ***	**Index data BW**

* T. Mitra. T. Chiueh, “*Dynamic 3D Graphics Workload Characterization and the architectural implications*”, MICRO ‘99

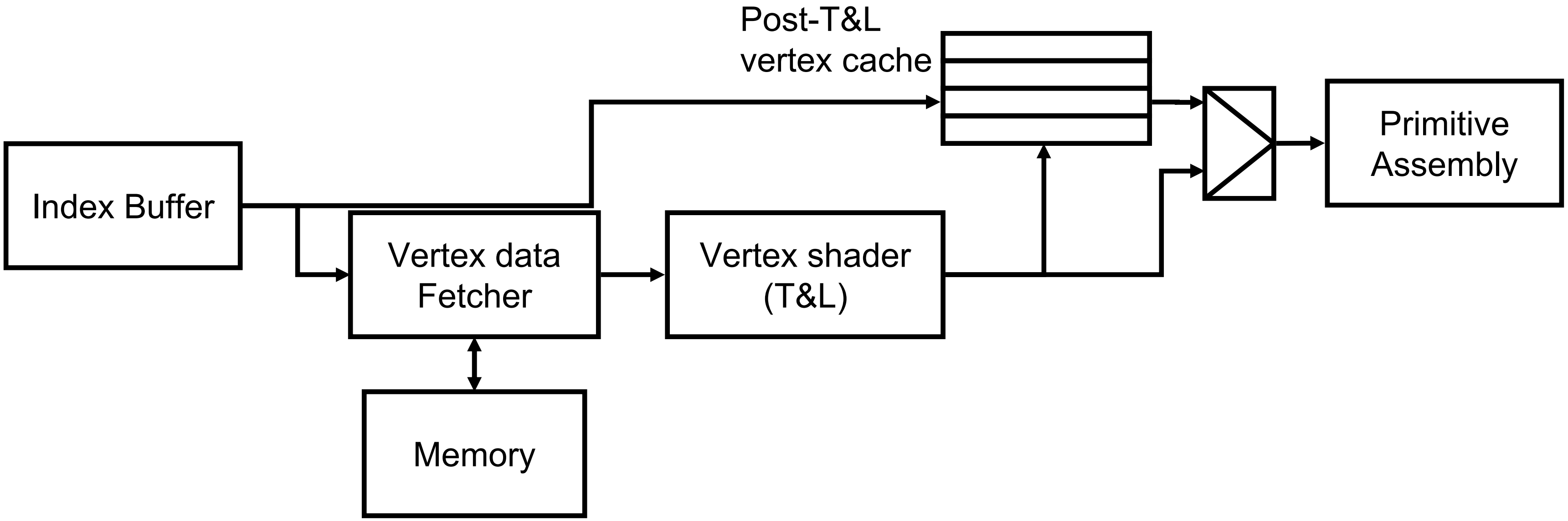
System → GPU traffic

Index BW

Game/Timedemo	Avg. batches per frame	Avg. indexes per batch	Avg. indexes per frame	Bytes per index	**Index BW at 100fps**	**PCIExpress x16 usage (4 Gb/s)**	**Triangle List**	**Triangle Strip**	**Triangle Fan**
UT2004/Primeval	229	1110	249285	2	**50 MB/s**	**1.3%**	**99.9%**		**0.1%**
Doom3/trdemo1	776	275	196416	4	**79 MB/s**	**2.0%**	**100%**		
Doom3/trdemo2	483	304	136548	4	**55 MB/s**	**1.4%**	**100%**		
Quake4/demo4	423	405	172330	4	**69 MB/s**	**1.7%**	**100%**		
Quake4/guru5	834	166	135051	4	**54 MB/s**	**1.4%**	**100%**		
Riddick/MainFrame	676	356	214965	2	**43 MB/s**	**1.1%**	**100%**		
Riddick/PrisonArea	363	658	239425	2	**48 MB/s**	**1.2%**	**100%**		
FEAR/built-in demo	488	641	331374	2	**66 MB/s**	**1.7%**	**100%**		
FEAR/interval2	294	1085	307202	2	**61 MB/s**	**1.5%**	**96.7%**	**3.3%**	
Half Life 2 LC/built-in	441	736	328919	2	**66 MB/s**	**1.7%**	**100%**		
Oblivion/Anvil Castle	564	998	711196	2	**142 MB/s**	**3.4%**	**46.3%**	**53.7%**	
Splinter Cell 3/first level	563	308	177300	2	**35 MB/s**	**0.9%**	**69.1%**	**26.7%**	**4.2%**

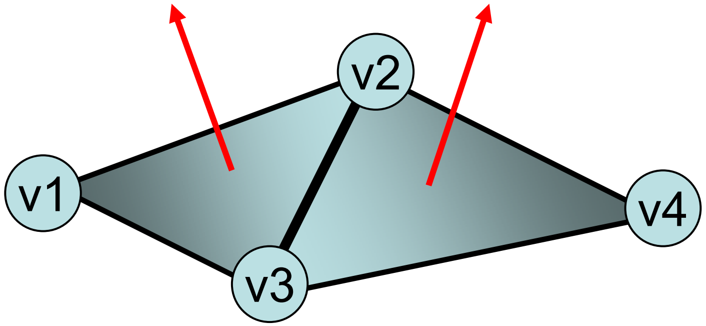
System → GPU traffic

Post-T&L vertex cache



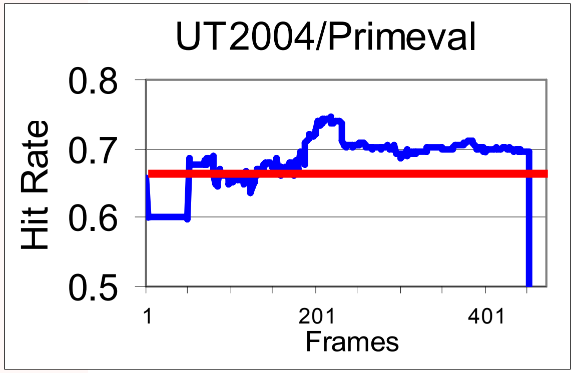
- For adjacent triangles lists:
  - 2/3 of referenced vertexes already computed :

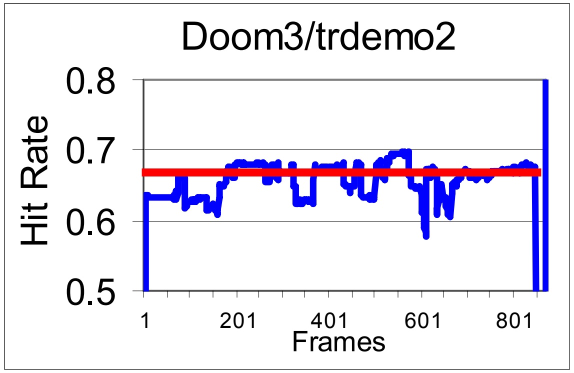
    66% hit rate

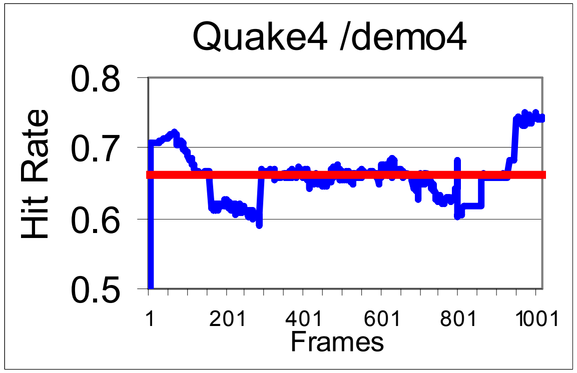


System → GPU traffic

Post-T&L vertex cache experiments





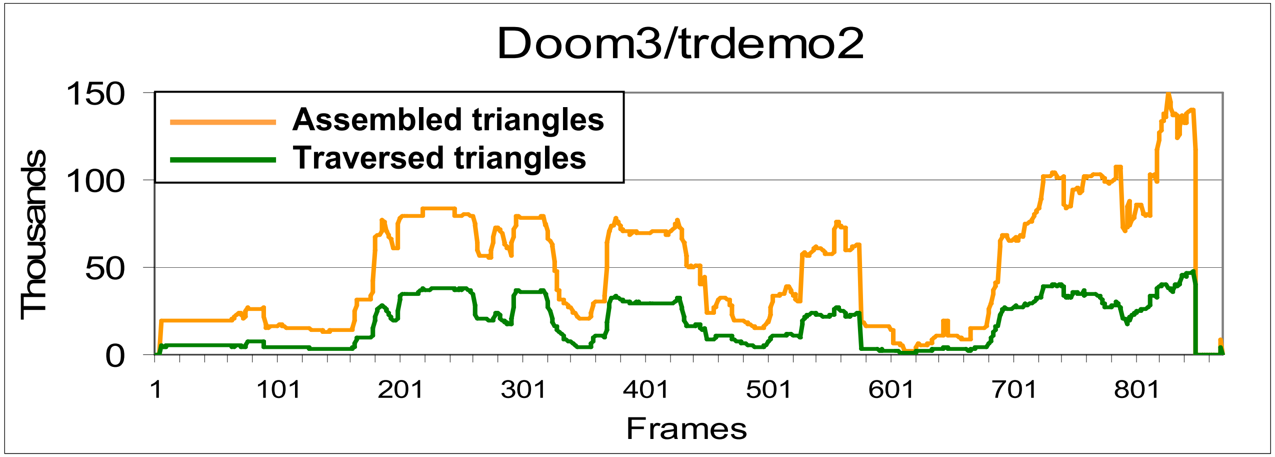


- Results show expected hit rate
- Game preference for triangle lists:
  - Low Bus BW usage related to index sent
  - Same vertex computation work as with strips or fans using a Post-T&L vertex cache
  - Triangle lists are easier managed by modeling tools.

Outline

- Introduction
- Game selection & stats gathering
- **Game analysis**
  - System → GPU traffic
  - **Primitive culling efficiency**
  - Rasterization pipeline
  - Fragment shading & texturing
  - Memory usage
- Conclusions

Primitive culling efficiency



Game/timedemo	%rejected		%traversed
	%clipped	%culled	
UT2004/Primeval	30%	21%	49%
Doom3/trdemo2	37%	28%	35%
Quake4/demo4	**51%**	21%	28%

- Clipping/Culling intensively used by our games.
- Quake4: half of the polygons lie out of the view volume.

- Game renderer engines let GPU do the important clipping/culling work:
  - Easier and cheaper in GPU Hardware.

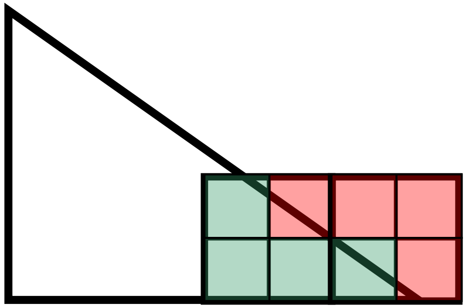
Outline

- Introduction
- Game selection & stats gathering
- **Game analysis**
  - System → GPU traffic
  - Primitive culling efficiency
  - **Rasterization pipeline**
  - Fragment shading & texturing
  - Memory usage
- Conclusions

Rasterization pipeline

The Basics

- Triangles are broken into quads (2x2 fragments)



- Boundaries generate non-full quads

- Quad frags are tested individually in different stages:
  - Z test (hidden surfaces),Stencil test, Alpha Test (transparency), Color Mask.
- Finally alive frags update framebuffer
- Empty quads are not further processed

Rasterization pipeline

Experimentation

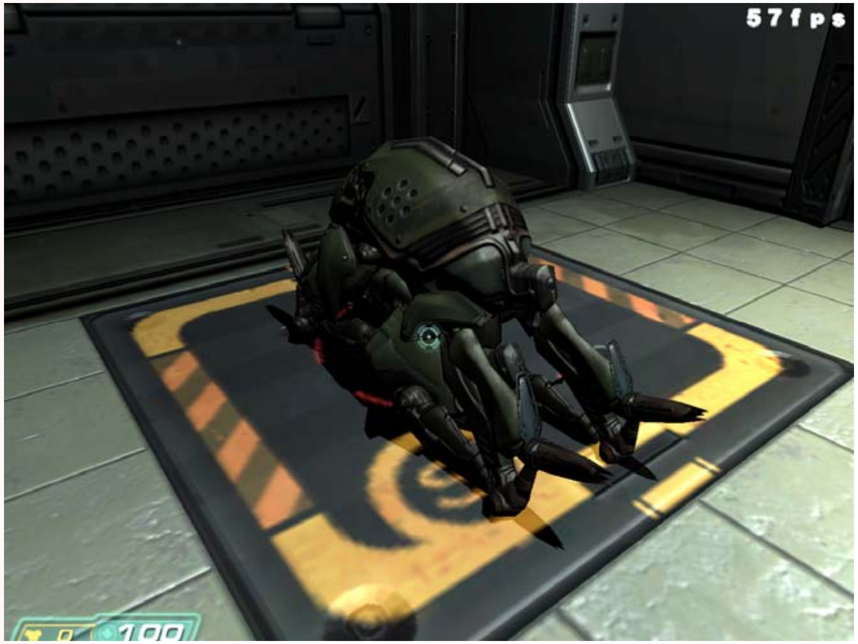
- Quad generation efficiency:

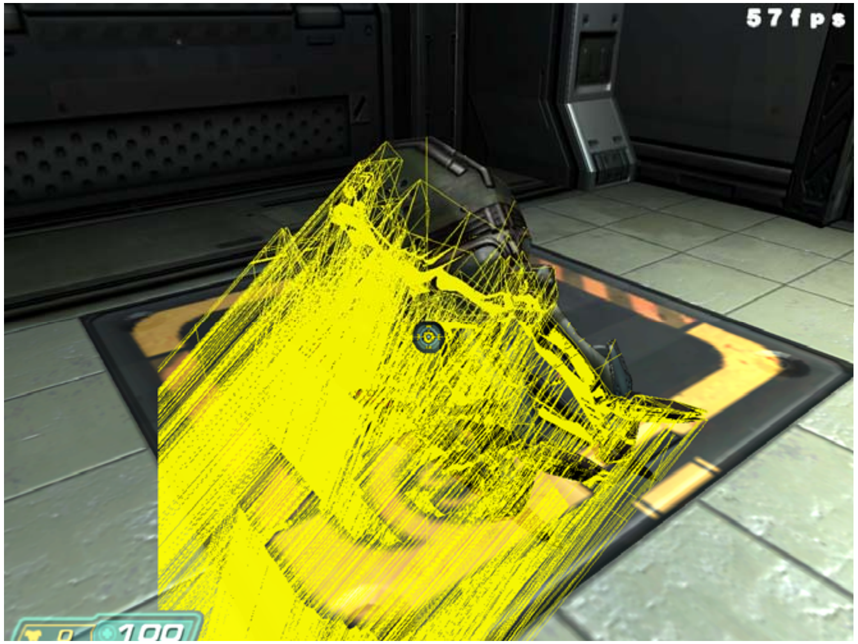
Game/timedemo	Avg Triangle Size	Avg Quad Efficiency
UT2004/Primeval	652	92%
Doom3/trdemo2	2117	93%
Quake4/demo4	1232	92%

- Higher efficiency than reported in [Mitra 99]
  - Results show between 40 and 60% efficiencies.
  - Interactive 3D games use less detailed 3D models (larger triangles).

Rasterization pipeline

- Doom3 and Quake4
  - Polygon rasterization overhead due to stencil shadow volumes (SSV)





Rasterization pipeline

- Fragment rejection breakdown:

Game/timedemo	Rejected Fragments				Blended Fragments
	HZ	Z&Stencil	Alpha	Color Mask = FALSE	
UT2004/Primeval	38%	2%	4.15%	0%	56%
Doom3/trdemo2	34%	14%	0.03%	34%	18%
Quake4/demo4	42%	21%	0.32%	19%	18%

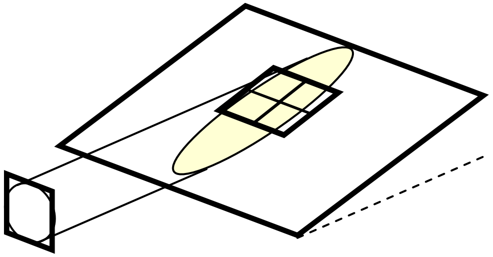
- On-die HZ greatly reduces GDDR BW avoiding Z&Stencil buffer accesses.
- In SSV games: Still room for higher BW reduction with HZ performing also Stencil test

Outline

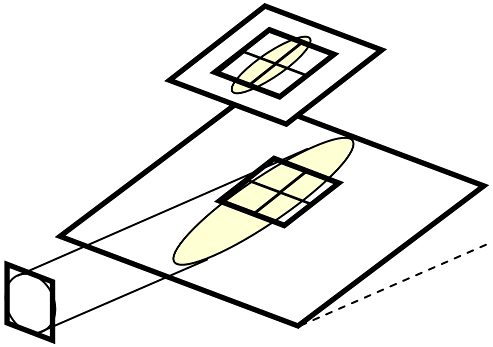
- Introduction
- Game selection & stats gathering
- **Game analysis**
  - System → GPU traffic
  - Primitive culling efficiency
  - Rasterization pipeline
  - **Fragment shading & texturing**
  - Memory usage
- Conclusions

Fragment shading & texturing

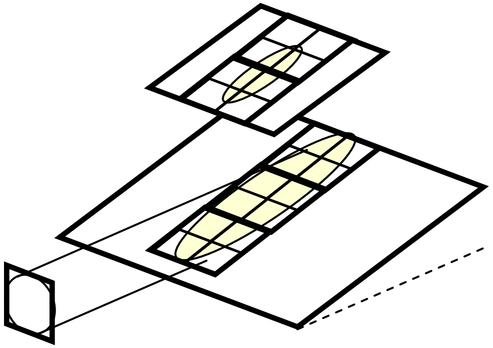
- Texture filtering cost measured in bilinears:



Bilinear filtering:
1 bilinear
(constant)



Trilinear filtering:
2 bilinears
(constant)



Anisotropic filtering:
from 2 up to 32 bilinears
(variable)

- Texture pipelines can usually execute 1 bilinear/cycle

Fragment shading & texturing

- ALU to Texture Ratio

Game/Timedemo	Instructions	Texture requests	**ALU to Texture Ratio**
UT2004/Primeval	4.6	1.5	**2.0**
Doom3/trdemo1	12.9	4.0	**2.2**
Doom3/trdemo2	13.0	4.0	**2.3**
Quake4/demo4	16.3	4.3	**2.8**
Quake4/guru5	17.2	4.5	**2.8**
Riddick/MainFrame	14.6	1.9	**6.6**
Riddick/PrisonArea	13.6	1.8	**6.4**
FEAR/built-in demo	21.3	2.8	**6.6**
FEAR/interval2	19.3	2.7	**6.1**
Half Life 2 LC/built-in	19.9	3.9	**4.1**
Oblivion/Anvil Castle	15.5	1.4	**10.4**
Splinter Cell 3/first level	4.6	2.1	**1.2**

Game/timedemo	Bilinear samples per tex. request
UT2004/Primeval	5.2
Doom3/trdemo2	4.4
Quake4/demo4	4.7

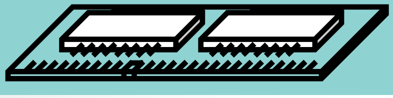
Game/timedemo	ALU instructions per bilinear request
UT2004/Primeval	**0.4**
Doom3/trdemo2	**0.5**
Quake4/demo4	**0.6**

- ATI Xenos, RV530, R580 peak performance:
  - Up to 3 ALU instructions per bilinear
  - 80% ALU power not used

Outline

- Introduction
- Game selection & stats gathering
- **Game analysis**
  - System → GPU traffic
  - Primitive culling efficiency
  - Rasterization pipeline
  - Fragment shading & texturing
  - **Memory usage**
- Conclusions

Memory usage



- Memory Hierarchy:

Cache	Size	Way	Line Size
Z&Stencil	16Kb	64	256B
Texture L0/L1	4Kb/16Kb	16/16	64B/64B
Color	16Kb	64	256B

- Specialized features:
  - Fast clears
  - Transparent compression

- Hit rate and miss BW:

Game/ timedemo	Z&Stencil		Texture		Color		% Read	% Write	BW@ 100fps
	% BW	Hit rate	% BW	Hit rate	% BW	Hit rate			
UT2004/Primeval	15%	94.9%	**42%**	97.7%	**35%**	93.7%	73%	27%	**8 GB/s**
Doom3/trdemo2	**54%**	91.0%	26%	99.2%	15%	93.2%	63%	37%	**11 GB/s**
Quake4/demo4	**51%**	93.4%	23%	99.3%	17%	93.2%	62%	38%	**10 GB/s**

- In non-SSV games (UT2004):
  - Most demanding stages: Texture, Color.
- In SSV games (Doom3, Quake4)
  - The most demanding stage: Z&Stencil (50%!!)

Conclusions

Conclusions

- Do our 3D games use GPU resources efficiently?

The results	The numbers
Low CPU ↔ GPU traffic when carrying idx data	1.5% PCIE x16 BW
Effective Post-T&L vtx cache with TLs.	66% hit rate
Clipping/Culling stages are shown very effective	51% to 72% of polygon reduction
On-die HZ greatly reduce GDDR BW because Z&Stencil is the most demanding stage	53% of total BW in Doom3
High quad efficiency	91% to 93%
ALU processing power is underutilised in fragment processing	80% ALU power unused

Conclusions

- Some inferred implications

Experimental Observations	Implications/Solutions
Games using SSV stress Z&Stencil the most (becomes the most GDDR BW demanding stage)	Improving HZ (i.e: supporting also stencil) would reduce even more total GDDR BW
Fragment processing does not exploit ALU processing power	• Increase ALU to Texture ratio in fragment programs (newer games tend to it) or • Reduce bilinears cost in anisotropic sampling.