Introduction to the Performance Analyzer For PlayStation®2

Kirk Bender & Geoff Audy
Developer Support Engineers
Sony Computer Entertainment America

Agenda

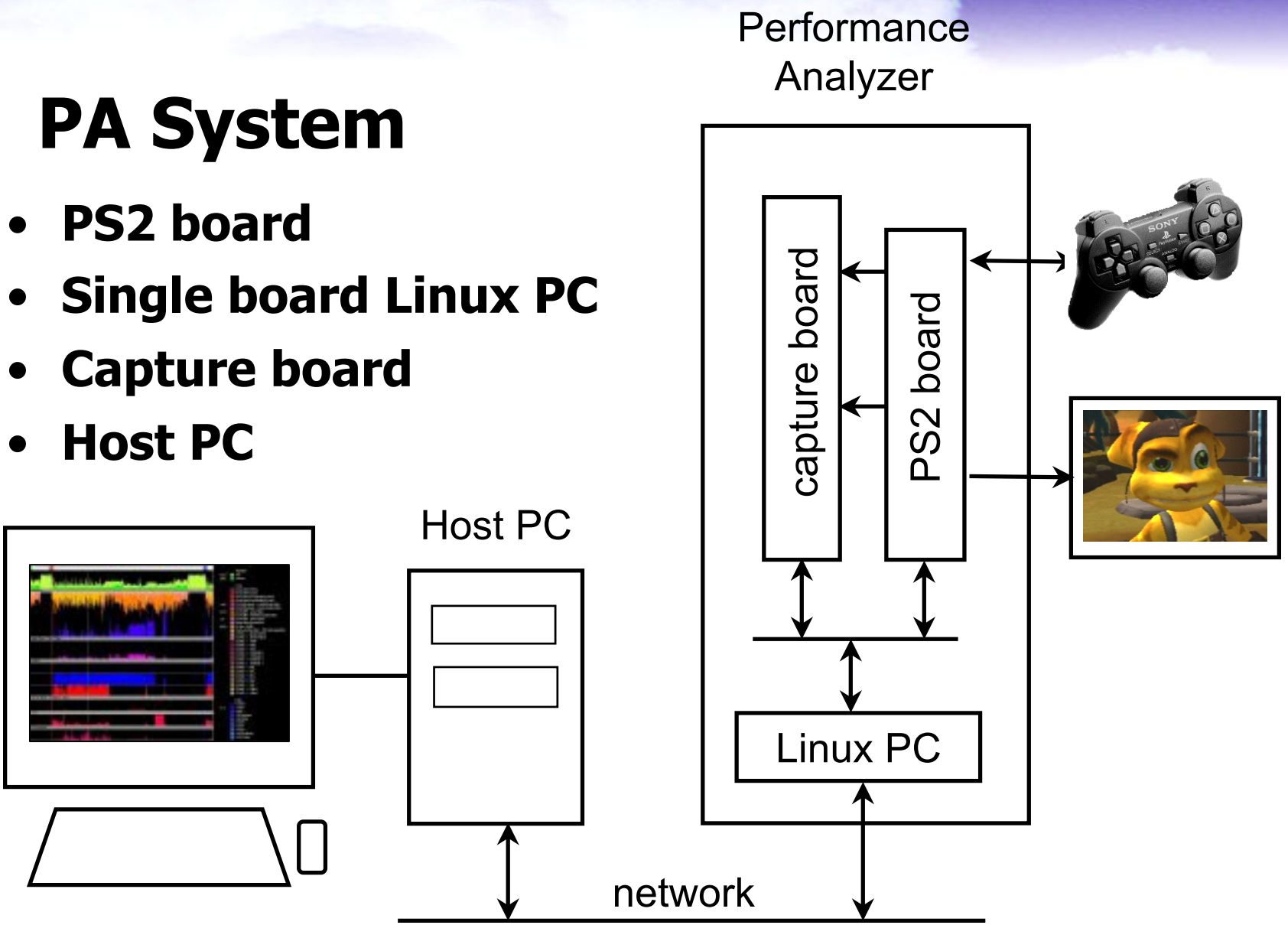
- Overview
- Tour of Features
- Demonstration
- Optimization Example
- Case Studies
- Packet Viewer

The Performance Analyzer

- **Hardware: captures a snapshot of PS2 processors, bus activity**
  - Development Tool + internal capture hardware
  - Samples at main bus clock- 150mhz
  - Three 256MB ring buffers
  - Up to 11 frames @ 60Hz
- **Software: captures & displays the data**
  - Shows how the PS2 is being utilized
- **Indispensable tool for optimization**
  - non-intrusive
  - Visualize and quantify efficiency

PA System

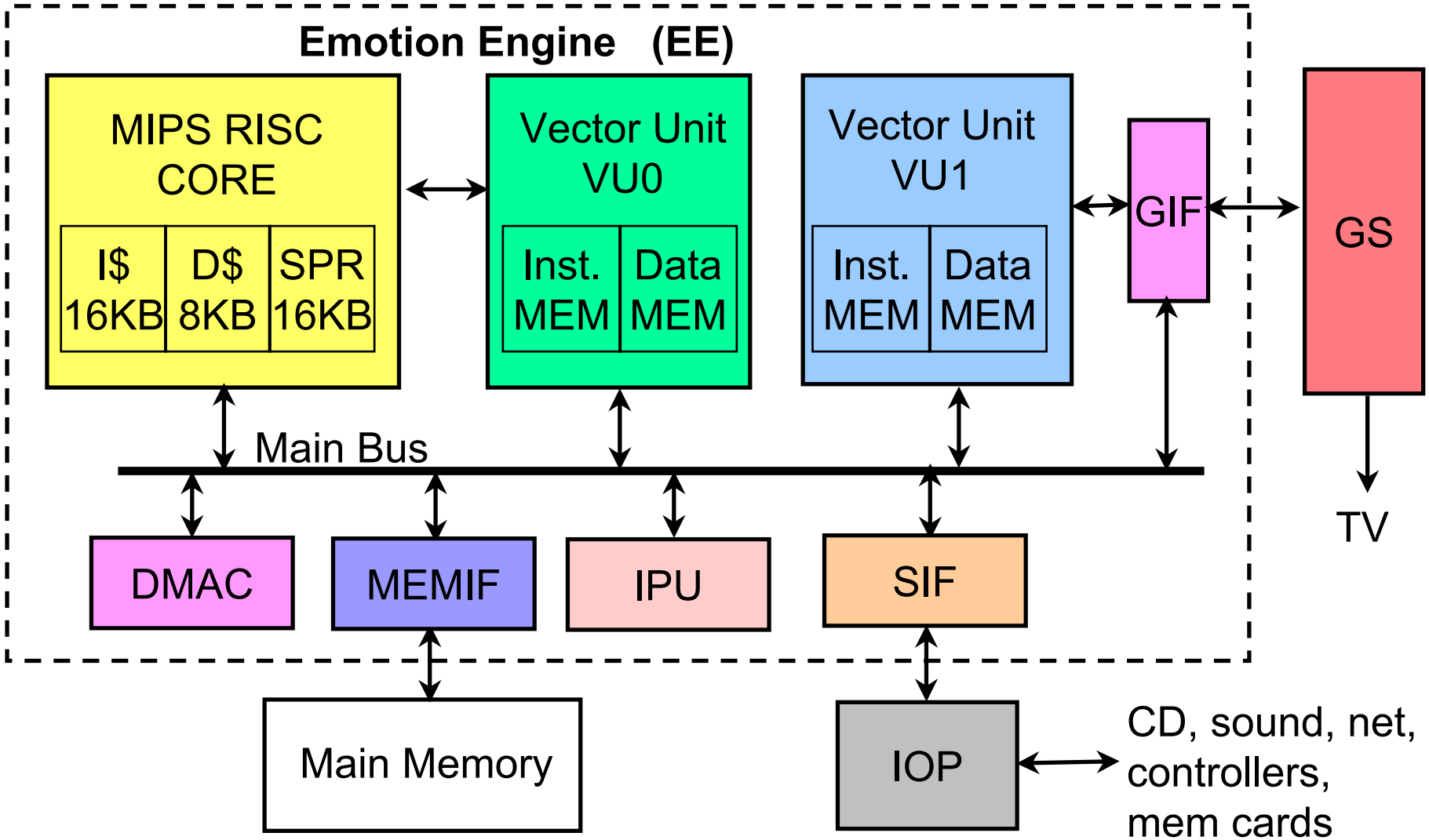
- **PS2 board**
- **Single board Linux PC**
- **Capture board**
- **Host PC**



How to Use

- **Set the game to scene of interest**
  - Boot from disc or run from host
- **Set Trigger** -many types available:
  - Manually, on GS register read/write, vblank, breakpoint, foot switch, or within code
- **Start Capture**
  - Data captured in ring buffers until trigger
- **Transfer data to PC**
  - About 24 MB/frame
- **View captures on Windows or Linux**
  - Graphs, statistics, etc.

PS2 Architecture Review



PA Software

- Using graphics & statistics, shows:
  - EE core pipeline
  - DMA: EE cache misses, DMA channels
  - VUs: Micro mode run, XGKICK blocking
  - GIF texture transfers, primitive packets
  - GS DDA: Pixels, primitives, texturing
  - GS VRAM: Host-local, page misses
  - IOP: I-cache misses, DMA, interrupts

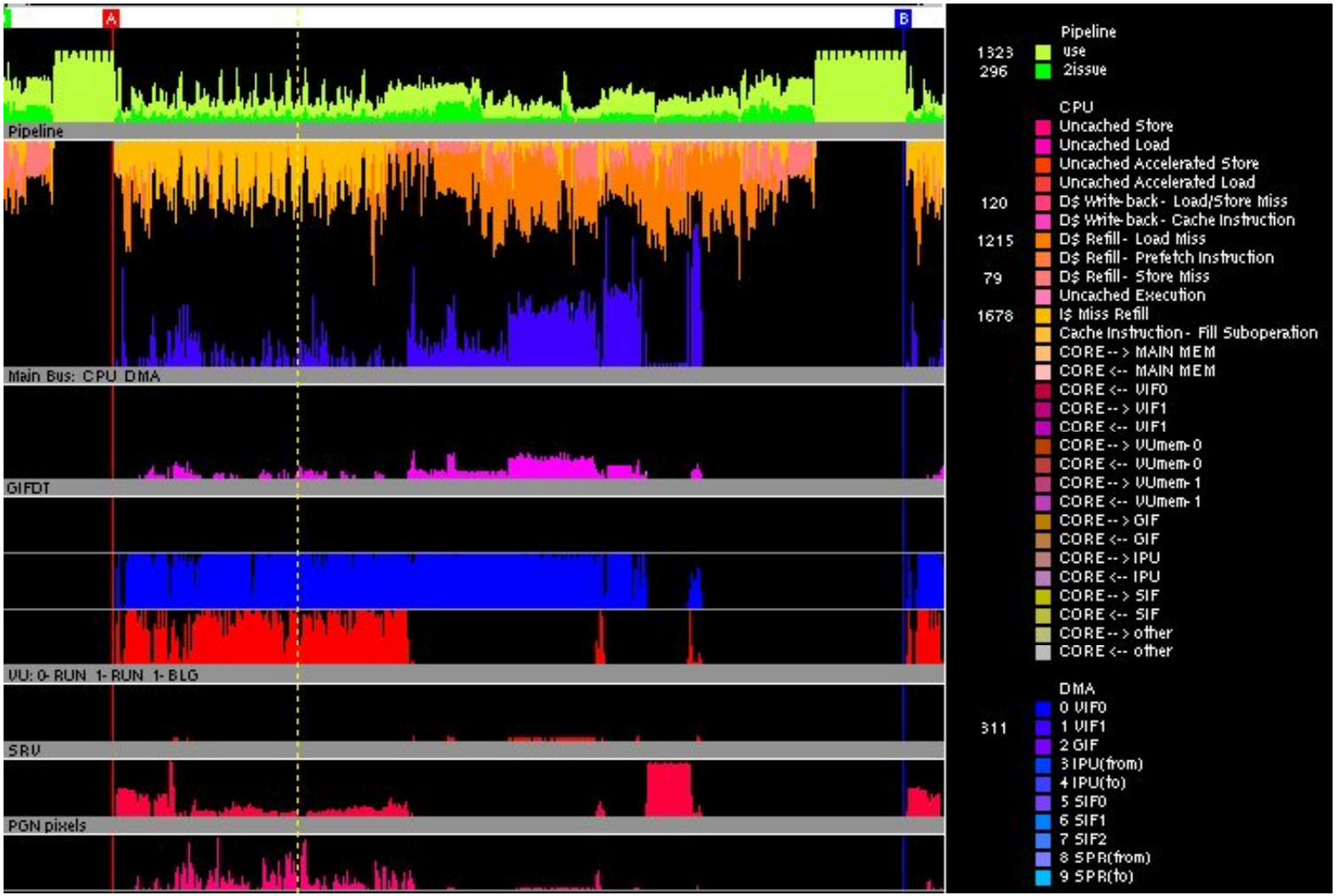
What the PA Can Do

- Separates processes into their parts
- Shows how busy the hardware is
- Shows bottlenecks
- Shows parallelism or lack thereof
- Gives facts and figures

What the PA Can't Do

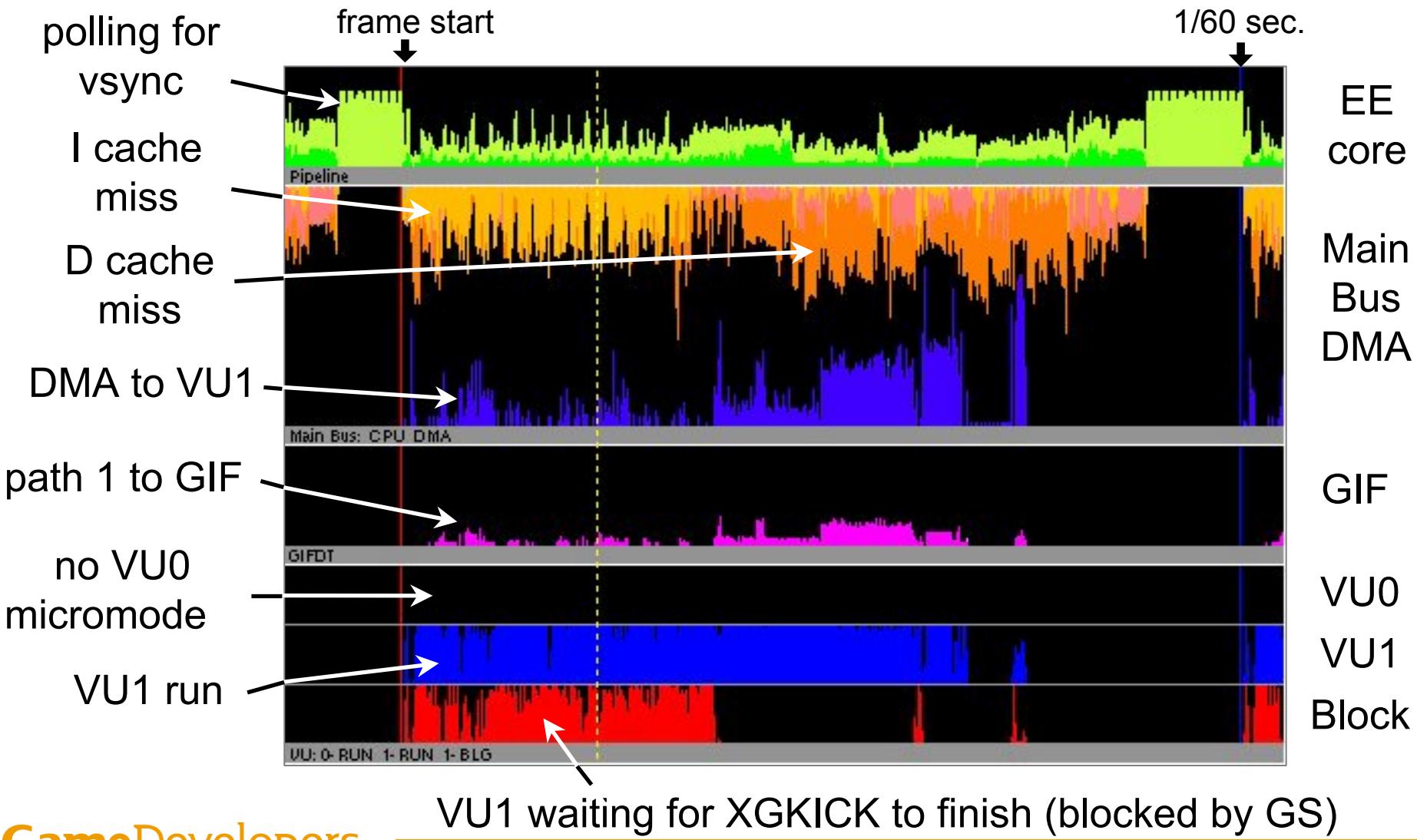
- Monitor realtime
- Capture program counter
  - Not a profiler: use SN Systems, Metrowerks, your own
- Capture actual DMA data
- Capture actual VRAM
  - But can capture and display GIF packets
  - User can add VRAM dump to trigger code
- Interpret
  - You have to do the analysis and interpretation
  - Need to know your goals and your code

Typical Capture

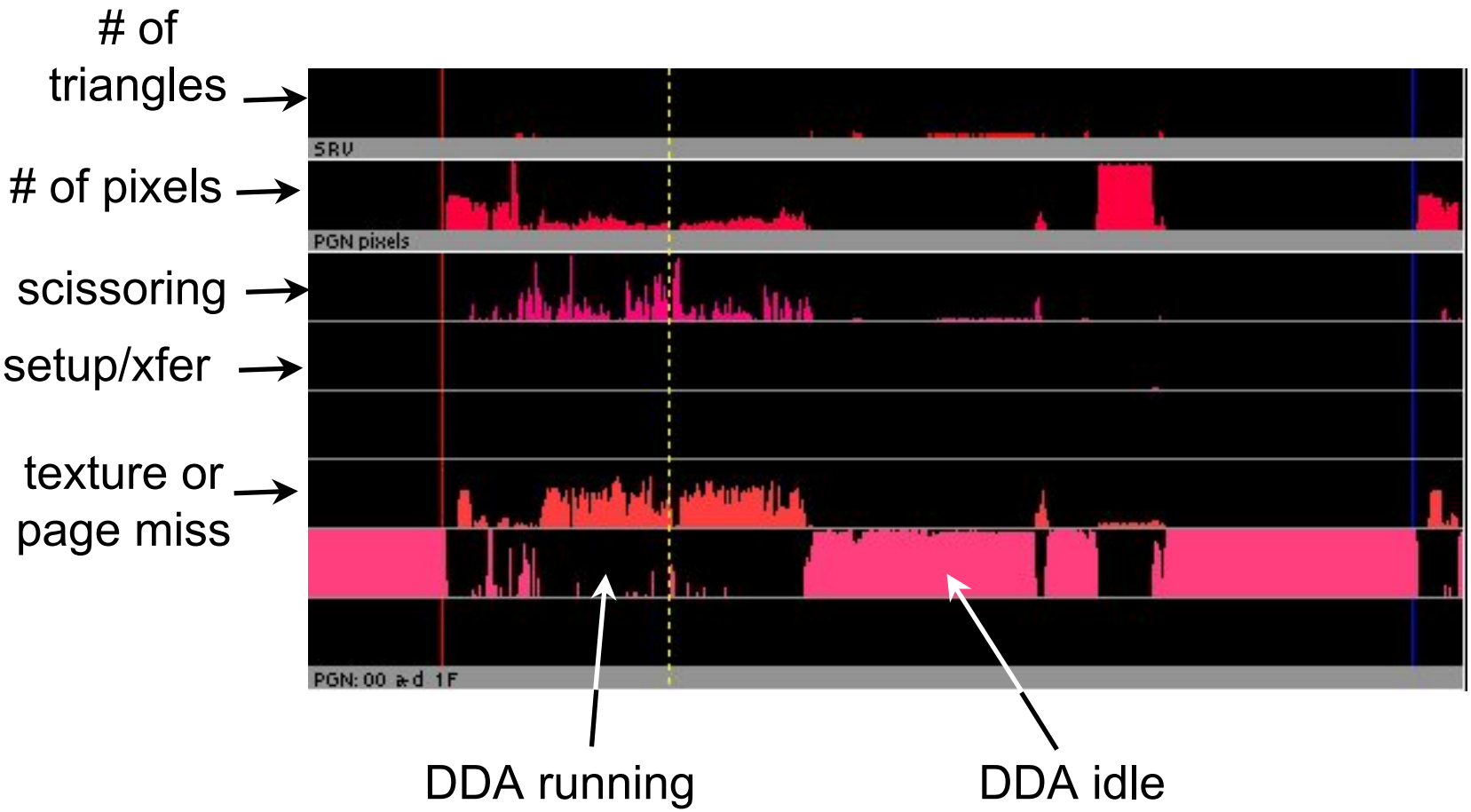


what does it mean...??

Graph: EE, DMA, VU



Graph: GS Status



Graph: GS Mem.

memory page misses

memory reads

memory writes

texture buffer misses

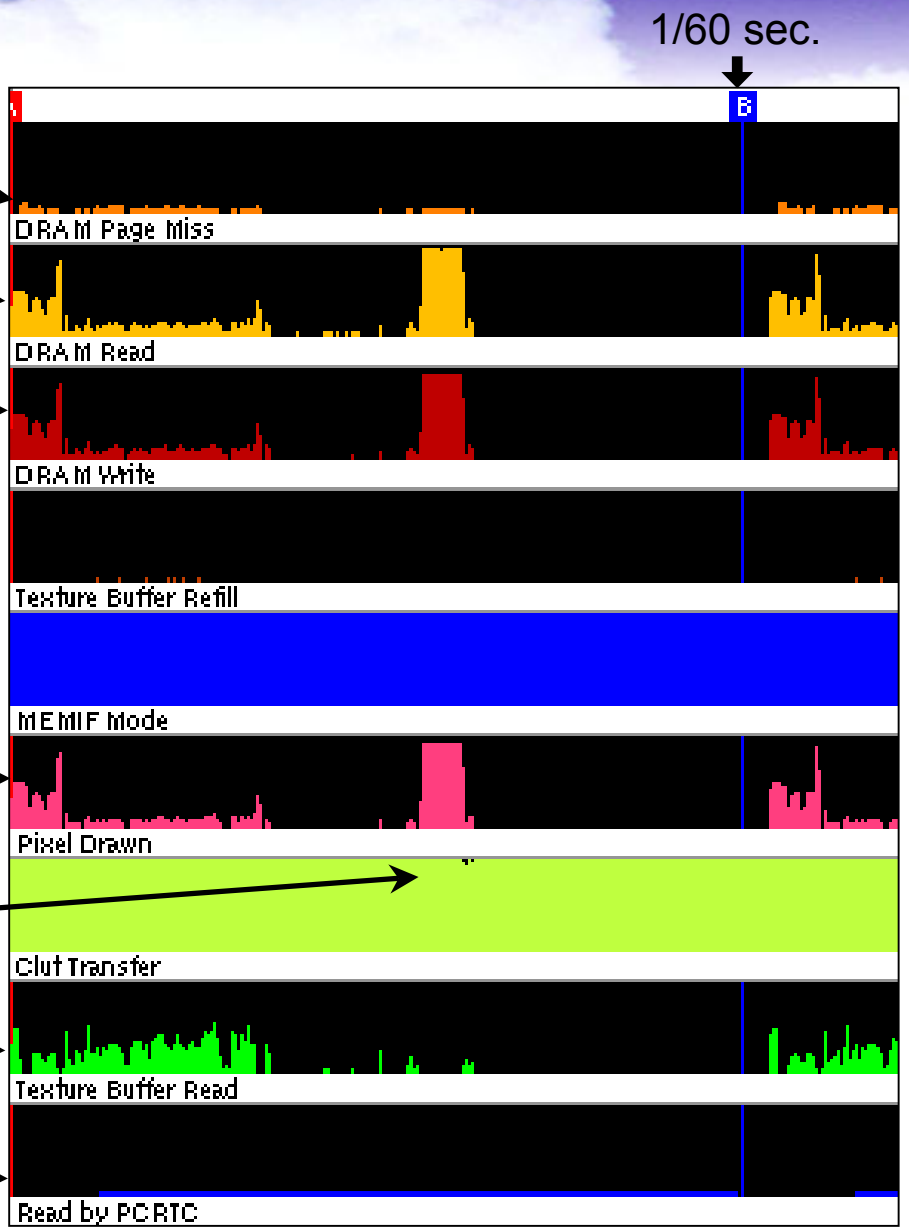
host-local transfers

pixels written

palettized textures

reads to SRAM buffer

vblanks



Graph: IOP

I cache misses →

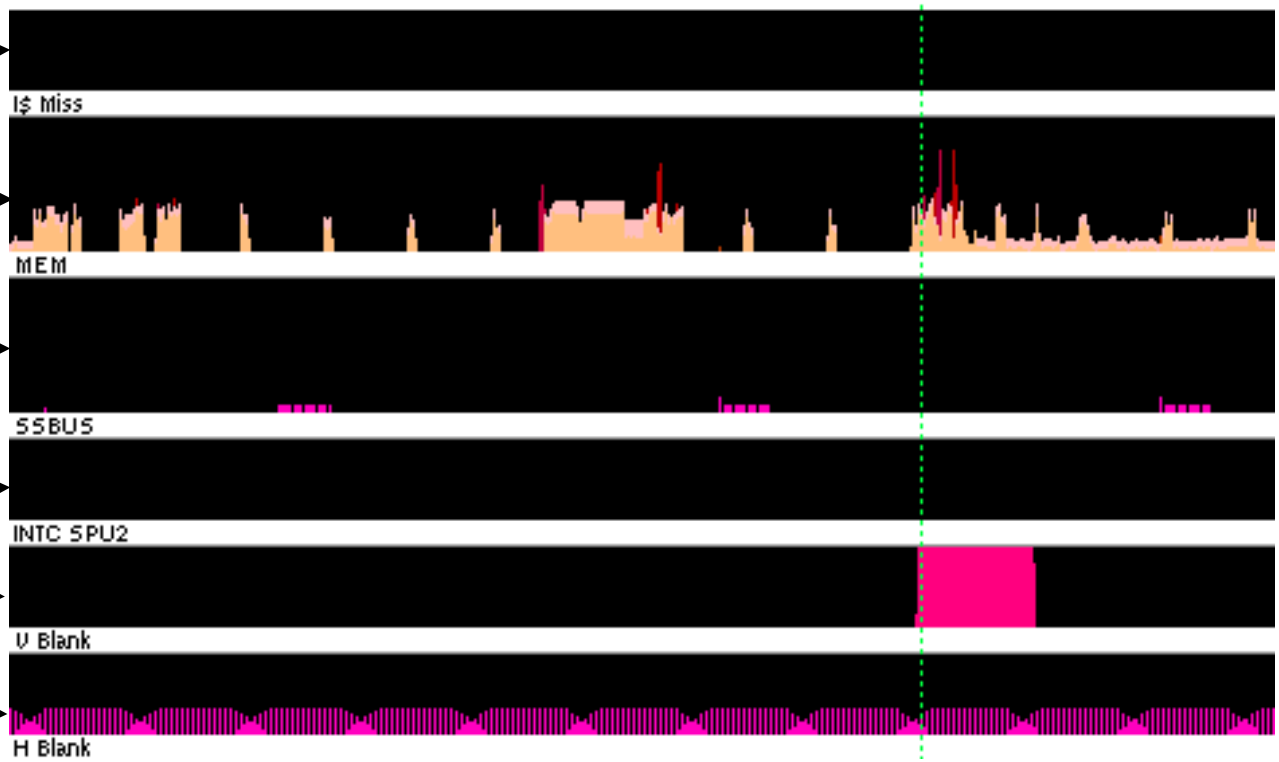
mem. loads/stores →

CD, SPU, net, xfers →

SPU interrupts →

vblanks →

hblanks →



Statistics

- Use the markers
  - Get a total of polygons, etc. in low-performance sections
- DMA statistics:
  - Occupation: total time it occupies the bus
  - Send: time actually sending the data
  - Occupation time >= Send time
- What are "good" numbers?
  - Depends

Statistics: DMA

send is 35% of occupation

Name	Occupy (cycles)	Send (cycles)	Eff. (%)	MB/s
VIF 0				
VIF 1	246243	86529	35.14%	78.89
GIF	24	15	62.50%	0.01
from IPU				
to IPU				
SIF 0	33	33	100.00%	0.03
SIF 1	236	81	34.32%	0.07
SIF 2				
from SPR				
to SPR				

Statistics: GS

DDA

Primitives Kicked	35365	(2.14M polys/s)
Pixels Generated	4201714	
Discarded	139119	5.71%
Stall	349411	14.35%
Non-polygonal Data	2333	0.10%
DDA Waiting	3264	0.13%
Idle	1489841	61.20%
DDACHK	0	0.00%

Memory

Page Misses	230872
Memory Read	1311326
Memory Write	1139303
Texture Read	54748
Pixel Write	4201714
CLUT Read	2864
Texture Buffer Read	3752839
Display Fetch	78395

Memory Mode

Normal	2434551
Buffer Clear	0
Host -> Local	0
Local -> Host	0
Local -> Local	0
EXT Video	0

2.14M polys/s:
not bad,
but could
do a lot more

TPC

- Target Program Counter
  - Relates graph, waveform to your code
  - Captures exceptions and jump targets
- Shows symbols for JALR jumps
  - compiler option -mlong-calls forces JALR
- Load executable to show symbols
  - Use unstripped .elf with symbols
- See which functions cause I$ misses
  - Combine functions or relocate

TPC (cont.)

Time	TPC	Trace	Address	Symbol
00000000		marker <A>		
00000012	E52234000	Jump A	10C894	VSync +0x64
00000023	EB4304000	Jump A	100D2C	sceGsSyncV + 0x2C
00000073	B8E0	Jump B	???3A0	
00000077	B8230400	Jump B	???100CA0	sceGsSwapDBuff +0x00
00000085	B57404000	Jump B	1011D4	sceGsPutDispEnv +0x14
00000100	E23304000	Jump A	100CC8	sceGsSwapDBuff +0x28
00000292	BA3	Jump B	???E8	
00000295	E1	Jump A	???4	
00000297	B41104000	Jump B	100450	killtime +0x00
00000317	E6F	Jump A	???3D8	
00000320	E41104000	Jump A	100450	killtime +0x00

Demonstration

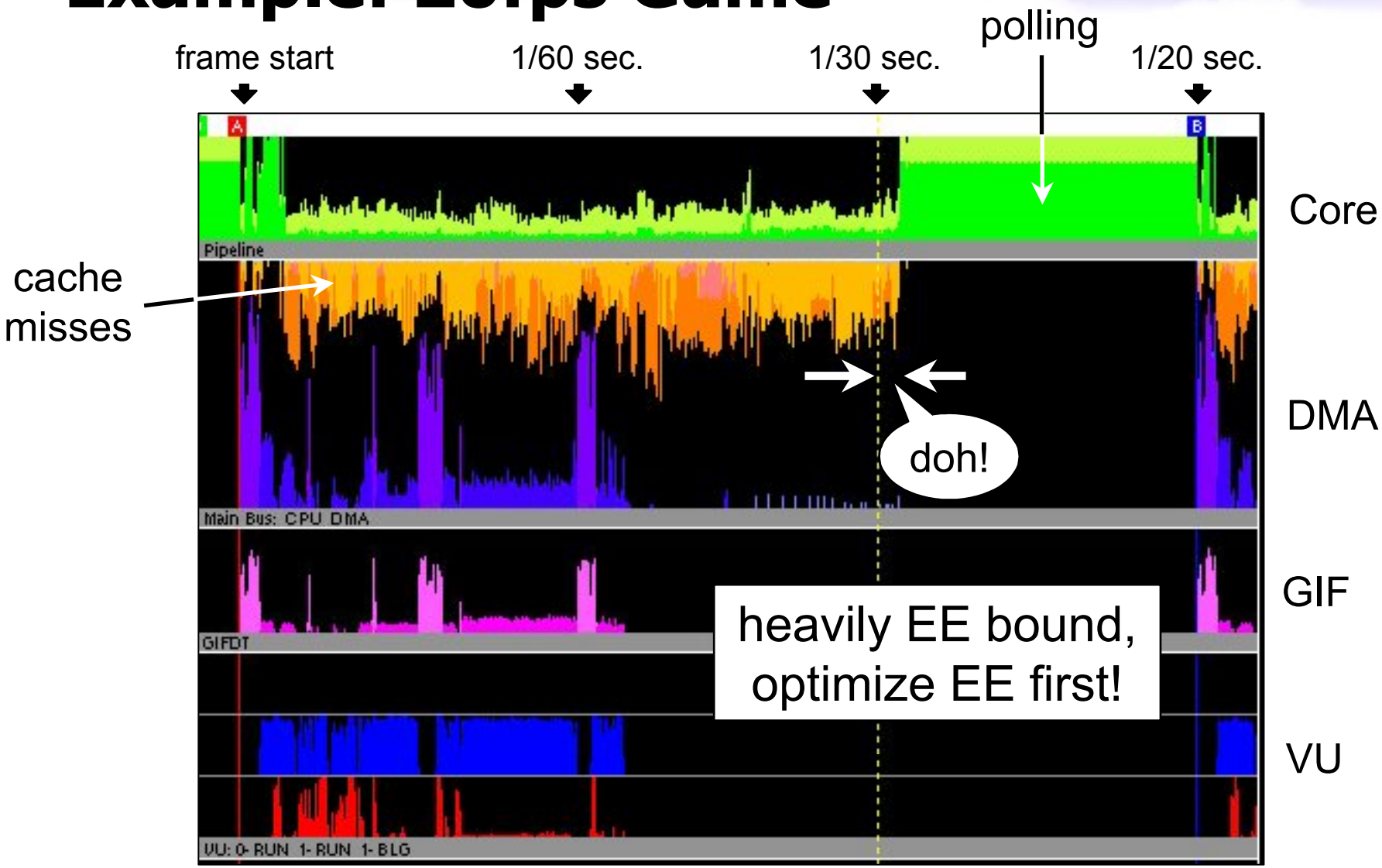
Using the PA: Goals

- Increase the frame rate
- Increase content for a given frame rate
- Refine engine designs
  - Choose a design, implement, optimize
- Fix bugs
  - Set trigger near hang point

Using the PA: Bottlenecks

- Prioritize: Is game EE or GS bound ?
  - Most games we've seen are EE bound
  - EE bottleneck can be memory bottleneck: use cache and scratchpad effectively
- Reduce processing for unseen polygons
- Increase parallelism among core, VUs

Example: 20fps Game



Example: Fluid Simulation

- Calculate & render fluid
  - 100 x 100 float height matrix
- Original: physics in EE core
  - Heavy data cache misses
- Optimizations:
  - 1: transpose height matrix in SPR
  - 2: memcpy using DMA via SPR
  - 3: calculate surface normals in SPR
  - 4: inline VU0 macros on SPR

Example: Fluid Simulation (cont.)

Original

data cache misses

VU1 DMA for vertices

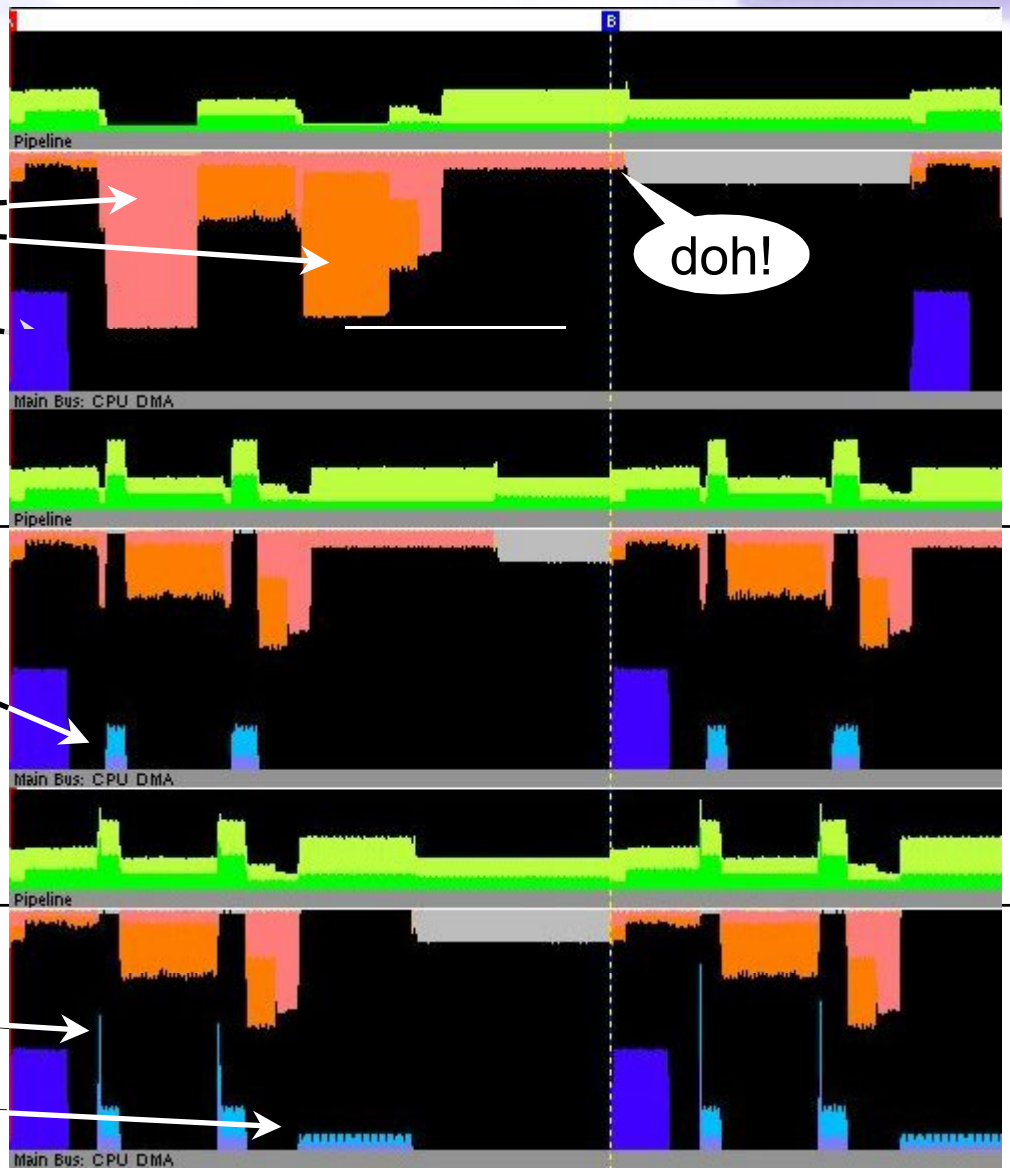
Optimized Version 1

transpose on SPR

Optimized Version 4

memcpy via SPR

normals on VU0, SPR



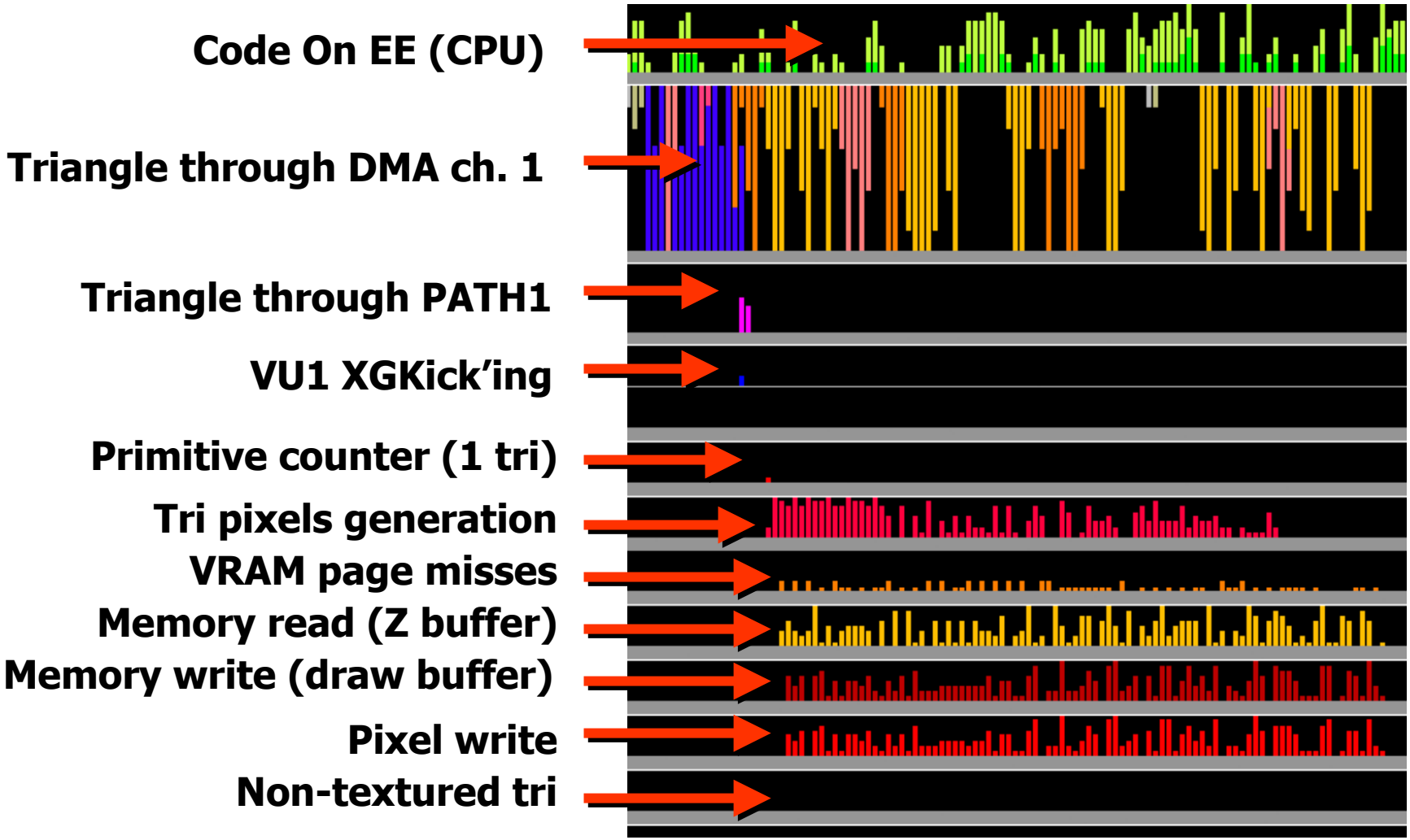
Using the PA: Tips

- Put markers in code to show in graph
  - e.g. VU0 micro mode
- printfs cause cache misses
- Save region of interest to a smaller file
- Detect frame rate drop and trigger in code
- Write code to capture VRAM to a file at trigger point (or use packet viewer)

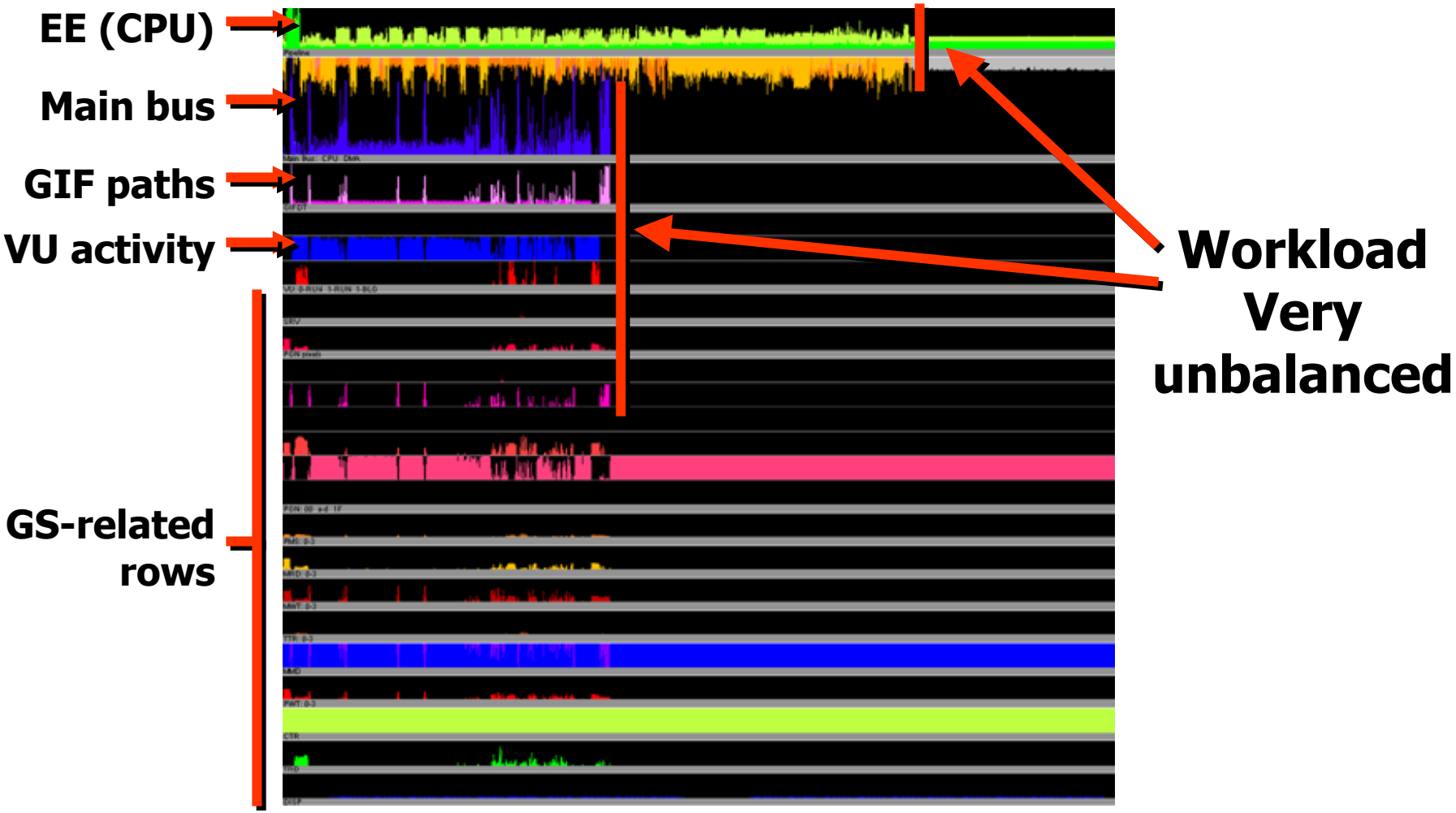
Part 2:

Case Studies
and
Packet Viewer

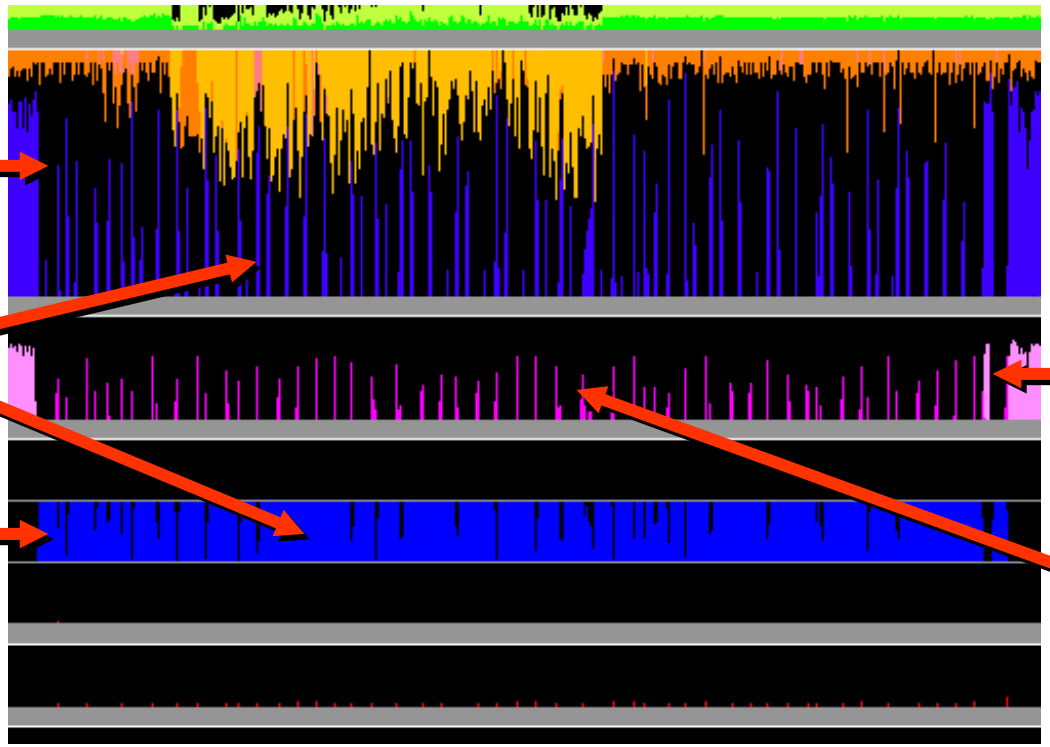
A Triangle Going Through



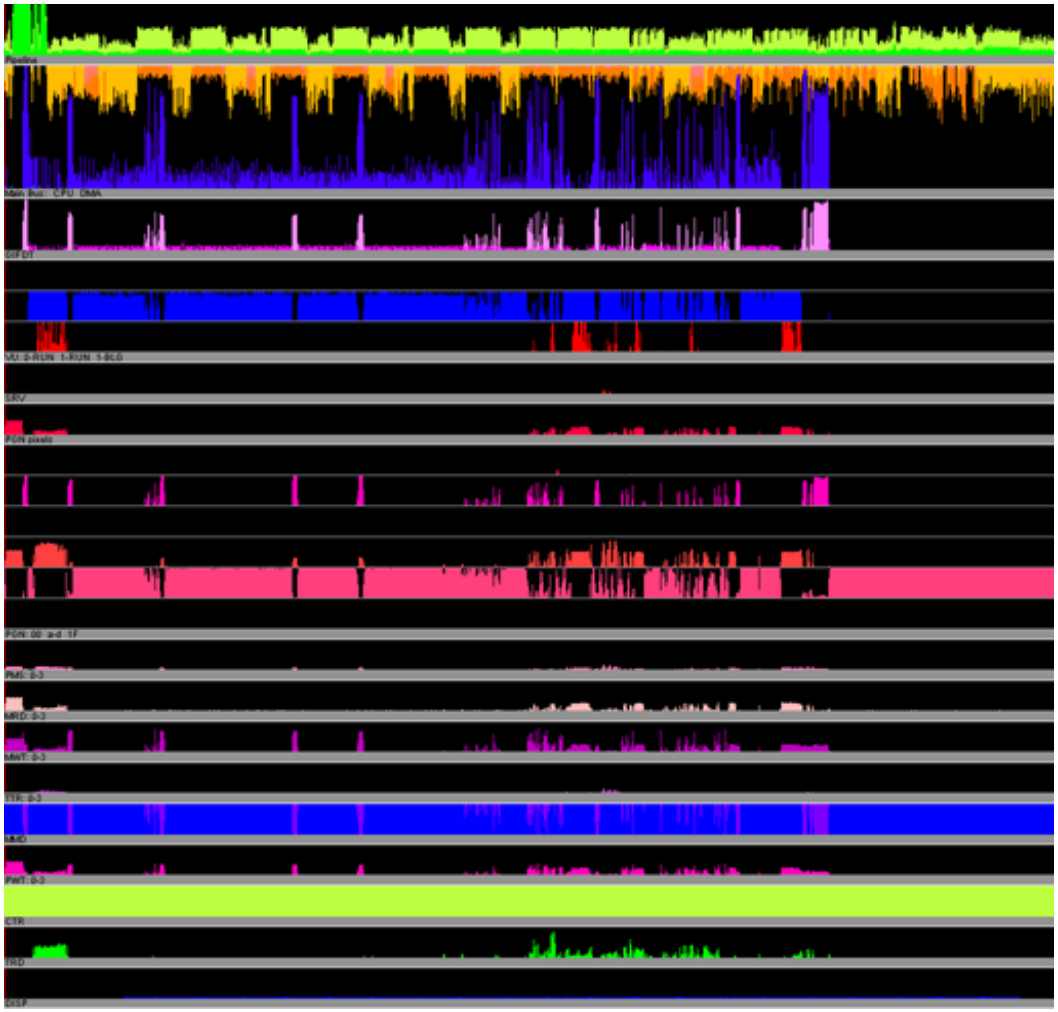
A Sub-Optimal Example



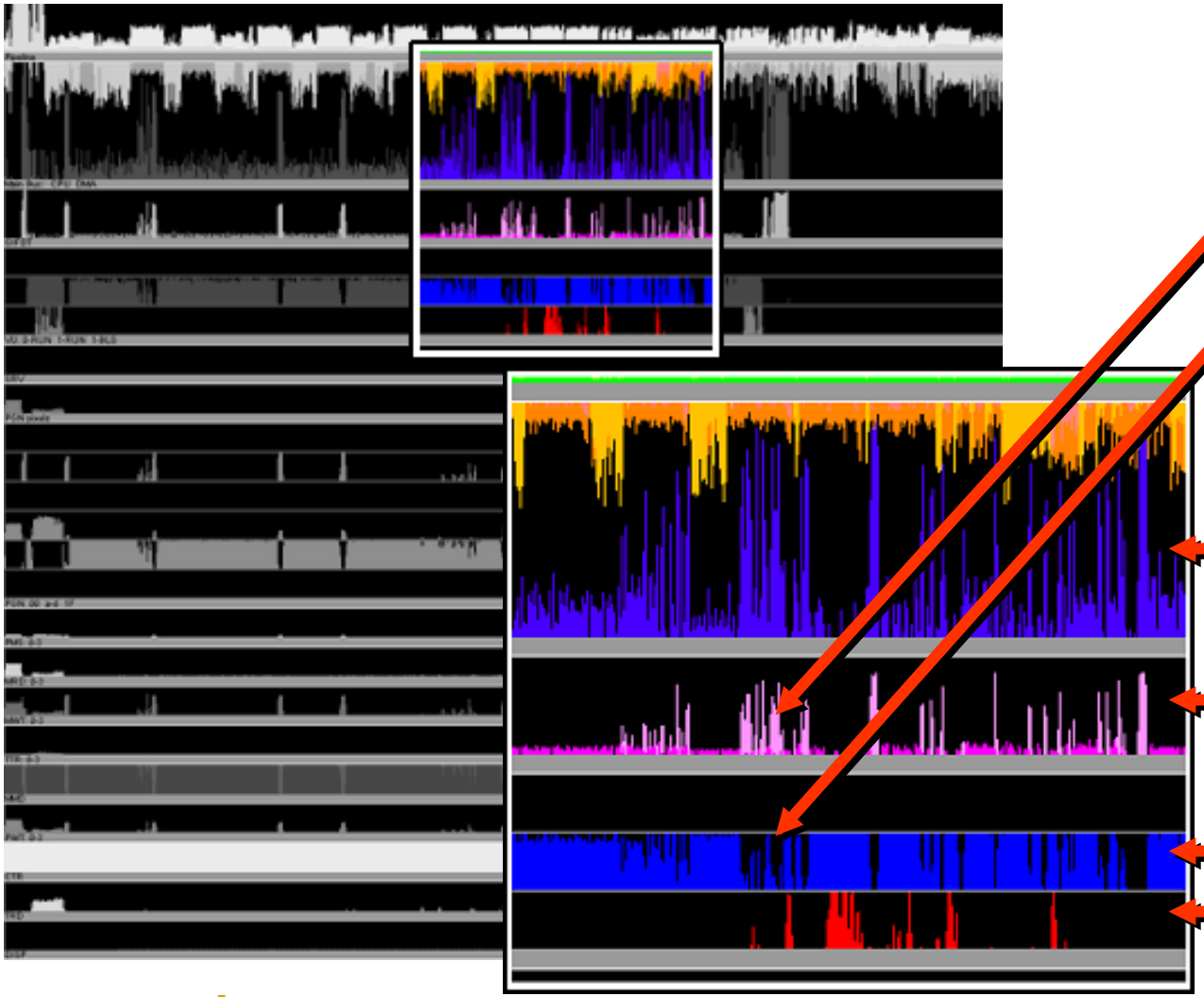
A Sub-Optimal Example (cont.)



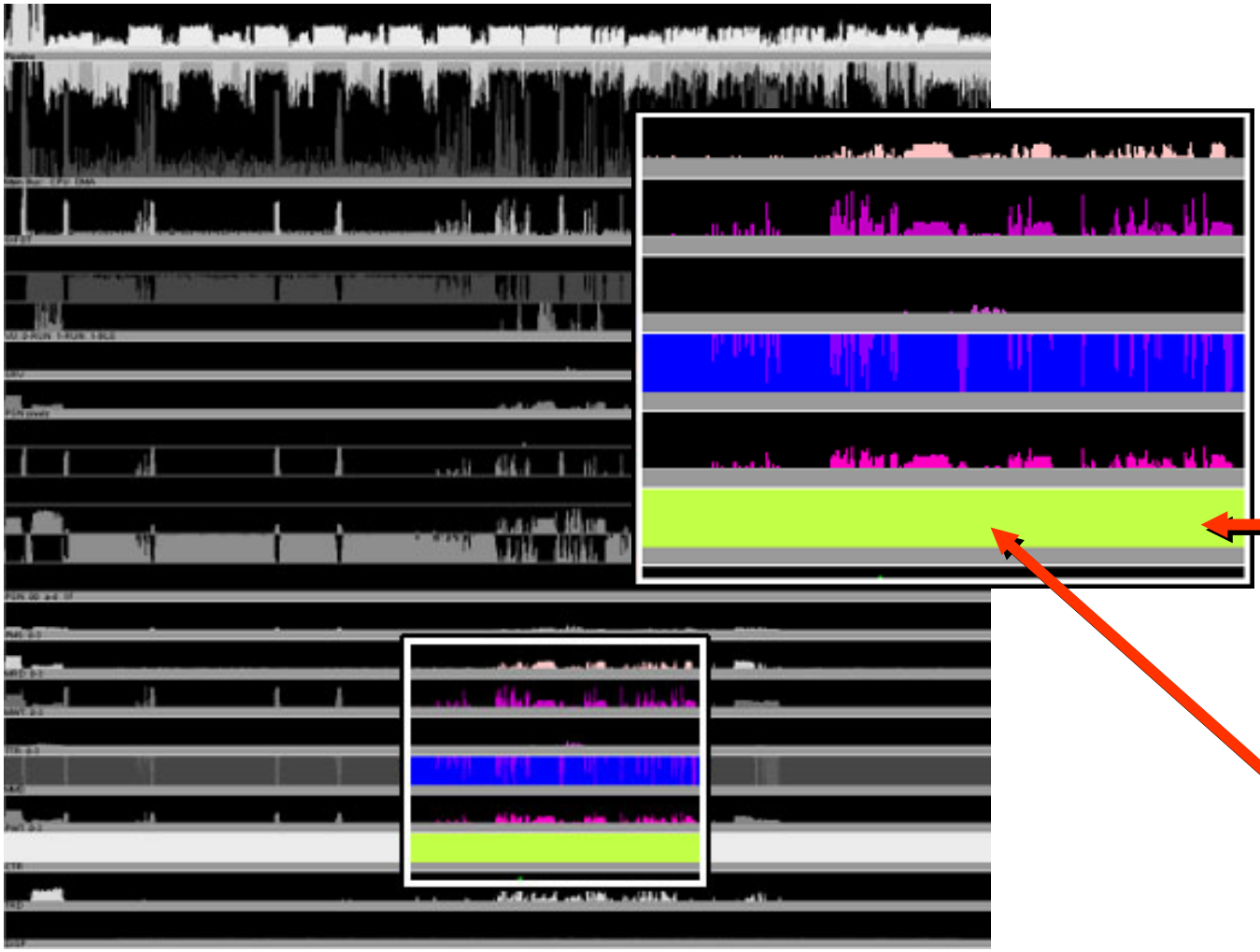
A Sub-Optimal Example (cont.)



A Sub-Optimal Example (cont.)



A Sub-Optimal Example (cont.)



CLUT reads

Palettized-textures not used

A Sub-Optimal Example (cont.)

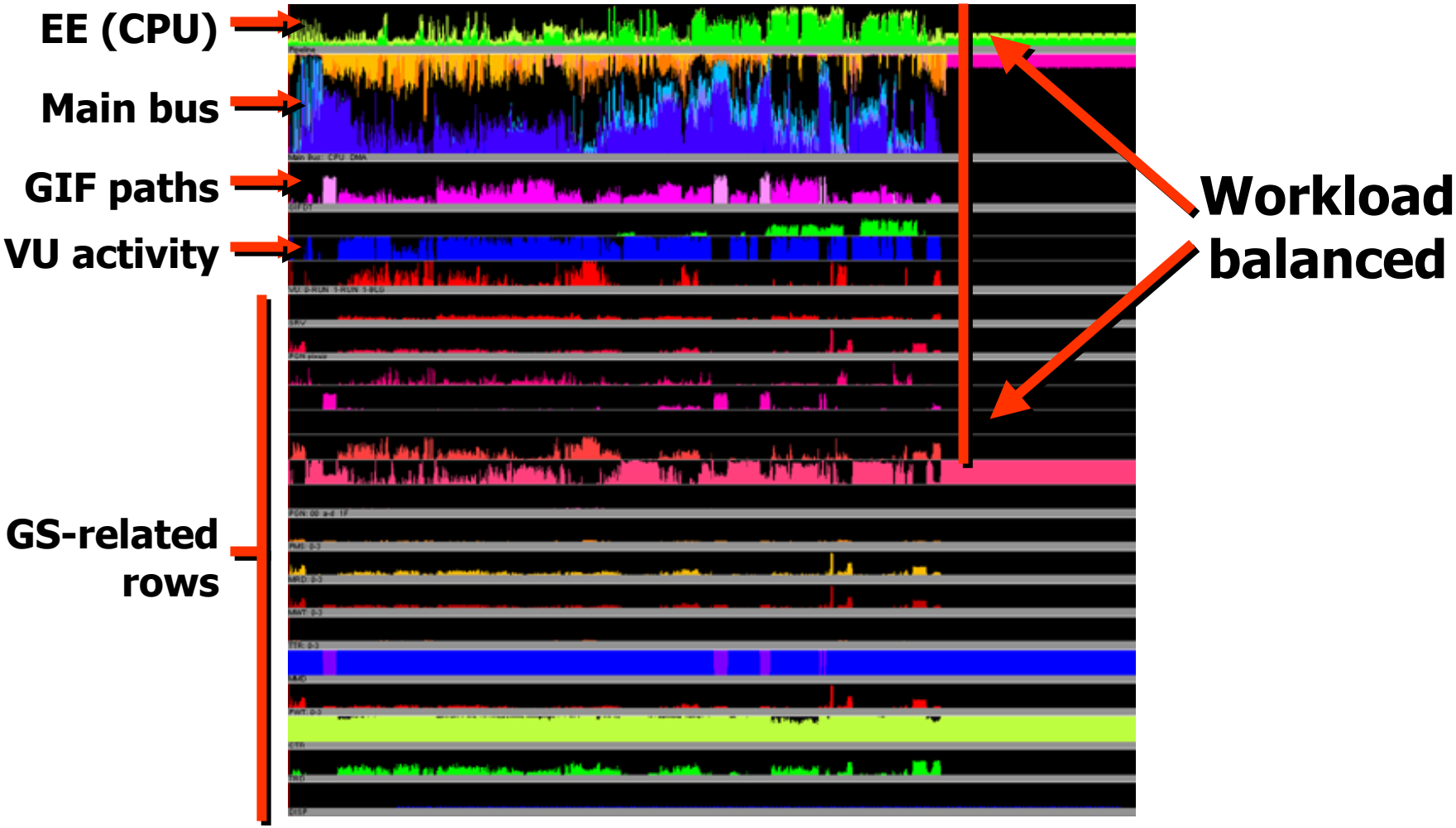
- ~10000 primitives / frame (300k/s)
- ~230k GS page misses / frame
- Average 23 page misses / primitive
- ~700k GS mem reads, ~800k GS mem writes

- Typical of low-polycount games using big triangles to make up for the quantity

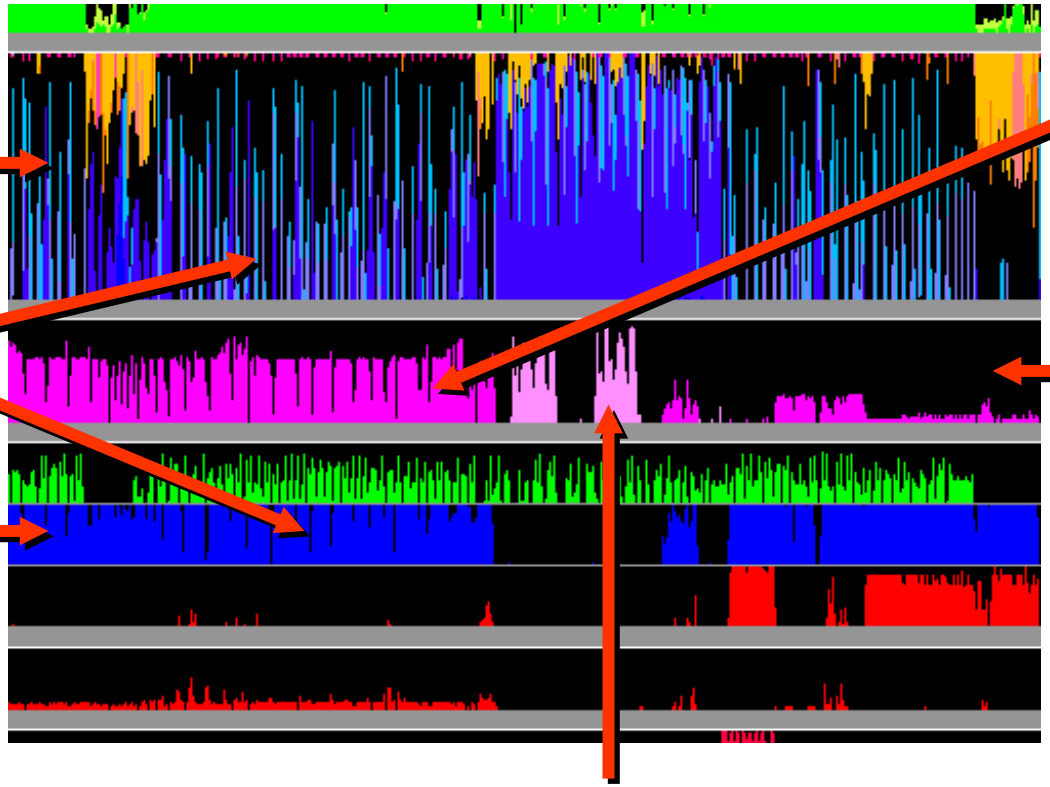
So What Can Be Done?

- Use PS2 preformatted data = Leave the CPU for game processing (ai, physics)
- Use strips
- Use specialized VU1 code for each case
- Use VU0 in micro-mode, and transfer data out to scratchpad
- Use 4- and 8-bit textures whenever possible
- Bottom line: Need to use parallelism as much as possible (PS2's strength!)

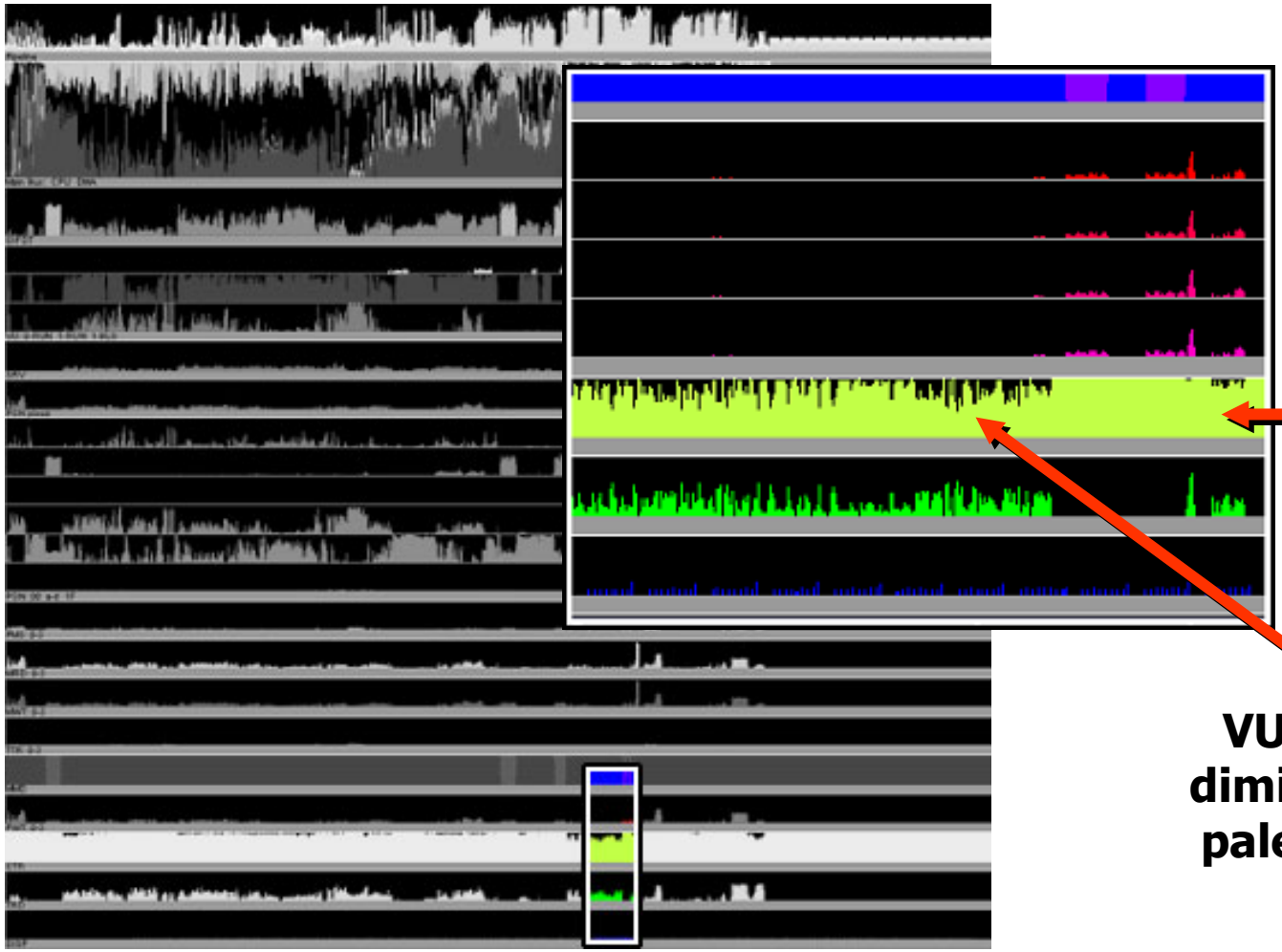
An Optimal Example



An Optimal Example (cont.)



An Optimal Example (cont.)



CLUT reads

VU1 starvation is diminished by using palettized textures

An Optimal Example (cont.)

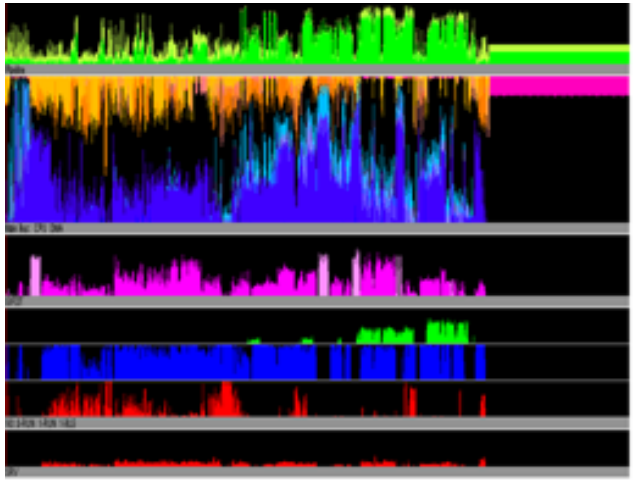
- ~120000 primitives / frame (7.2M / s)
- ~190k GS page misses / frame
- Average of 1.58 page misses / primitive
- ~720k GS mem reads, ~650k GS mem writes

- Typical of high-polycount games using small triangles

So What Have They Done Right?

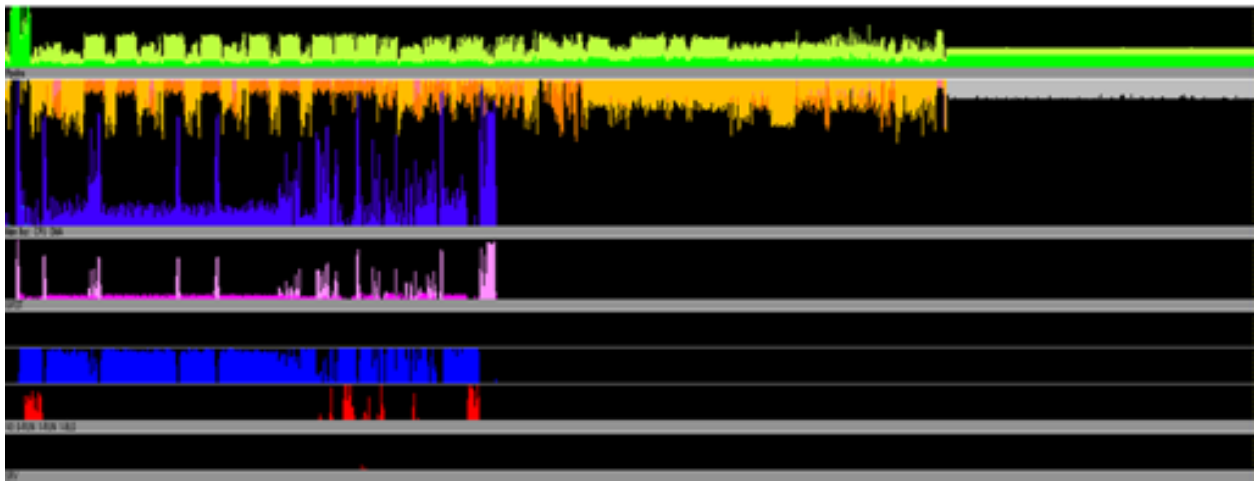
- Tailored their data for PS2 at tool time
  - Ready for VIF compression (possibly!)
  - Smaller and denser geometry (better for GS)
- Made good use of VU0 and scratchpad
- Wrote efficient and specialized VU1 code
- To avoid being EE-bound: Transferred workload to other processors (VU0 & VU1)
- Balance, balance, balance

A Side-By-Side Comparison



Optimal example
(60fps)

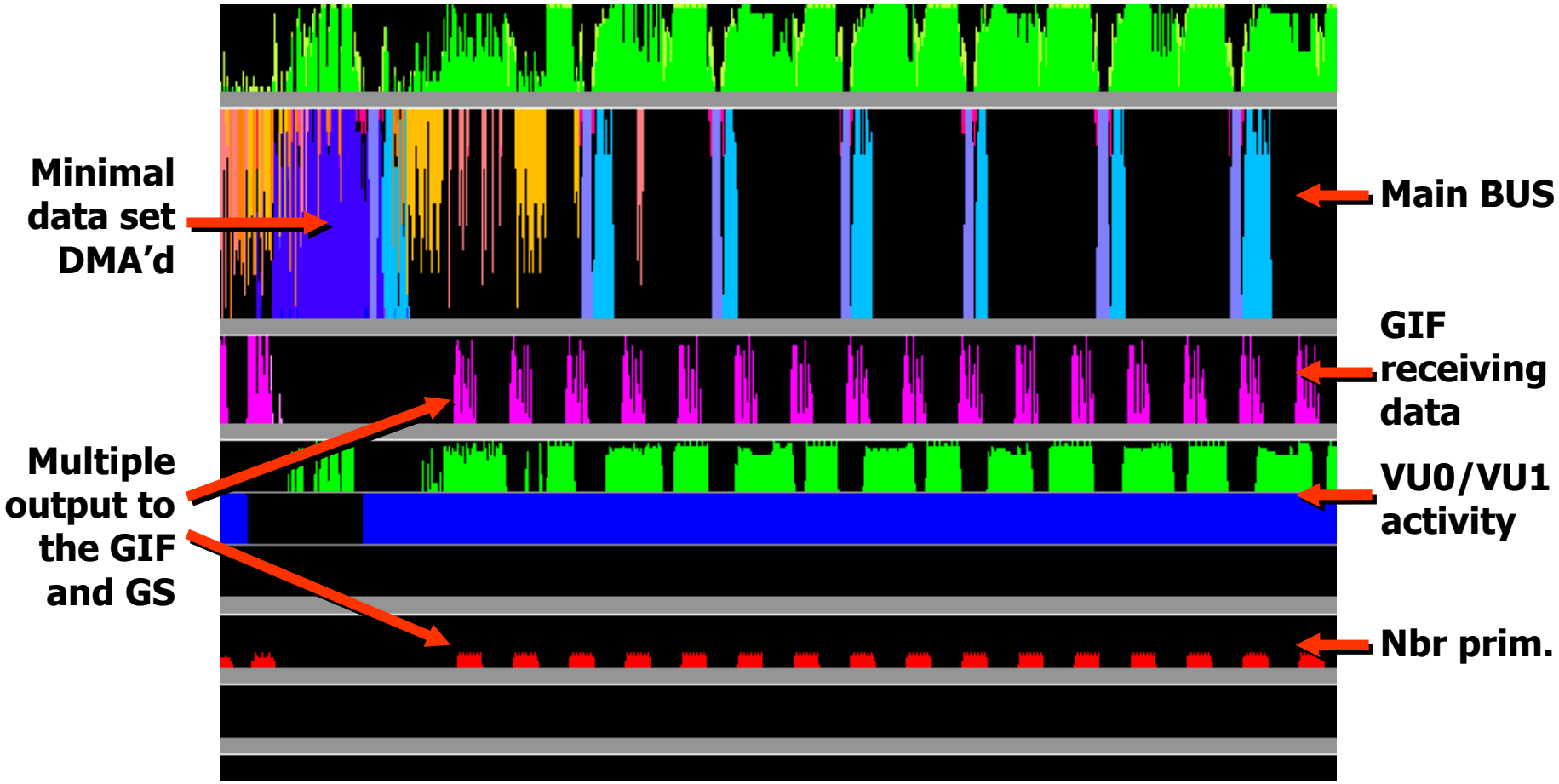
Sub-optimal example
(30fps)



A Suggestion If You're EE-Bound: Dump work on VU1!

- Contrary to usual saying, but…
- Send minimal data set
- Have VU1 generate the rest
- Must be creative (must be for rendering)
  - Particle system?
  - Building generation?
  - Crowd generation, for stadium-based games
  - Older ideas, like NURBS

A Suggestion If You're EE-Bound: Dump work on VU1!(cont.)



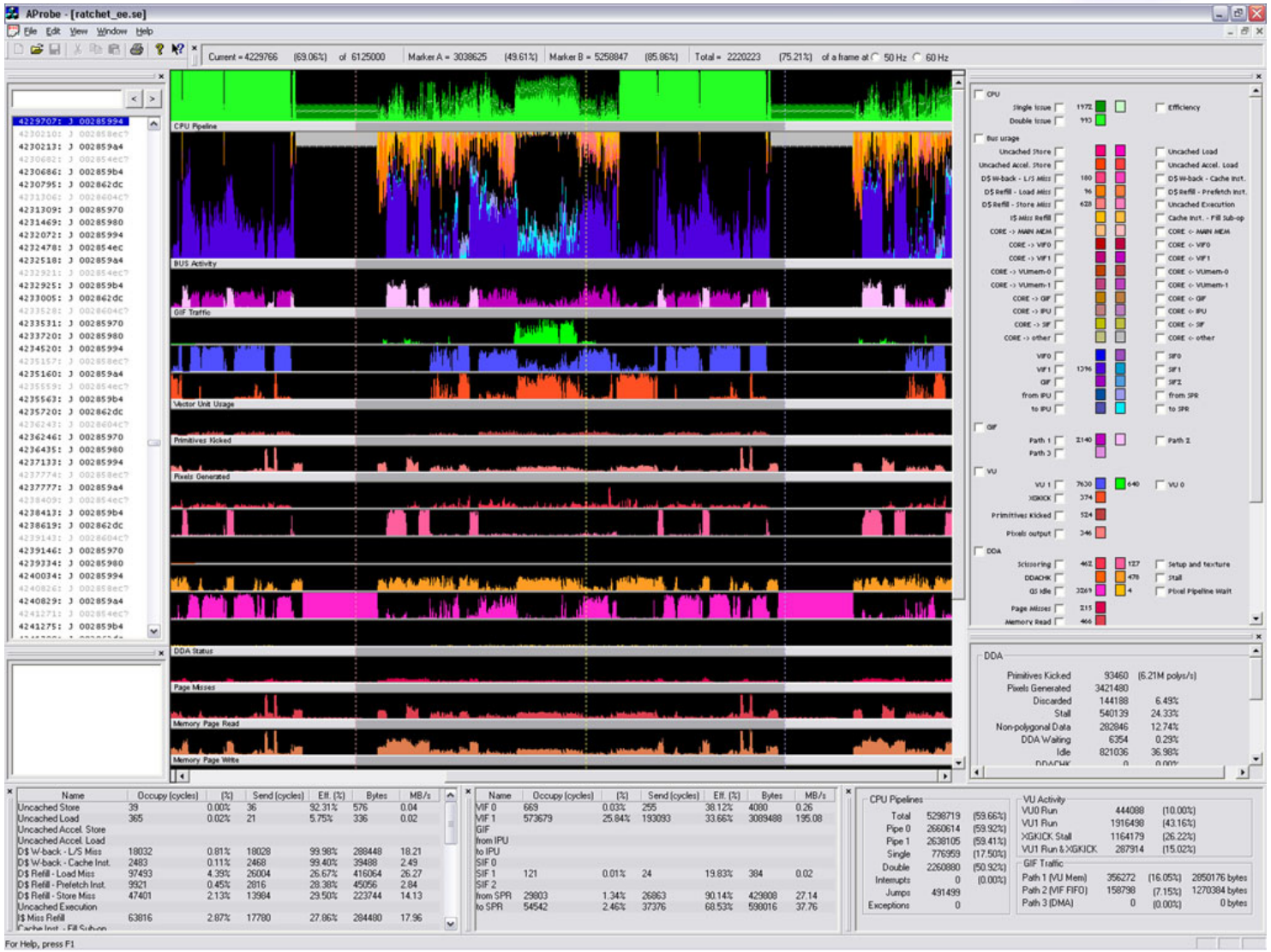
What You Ideally Want To Get

- Over 10M polygons per second, in-game
- More than 50% CPU usage
- More than 80% dual-issue

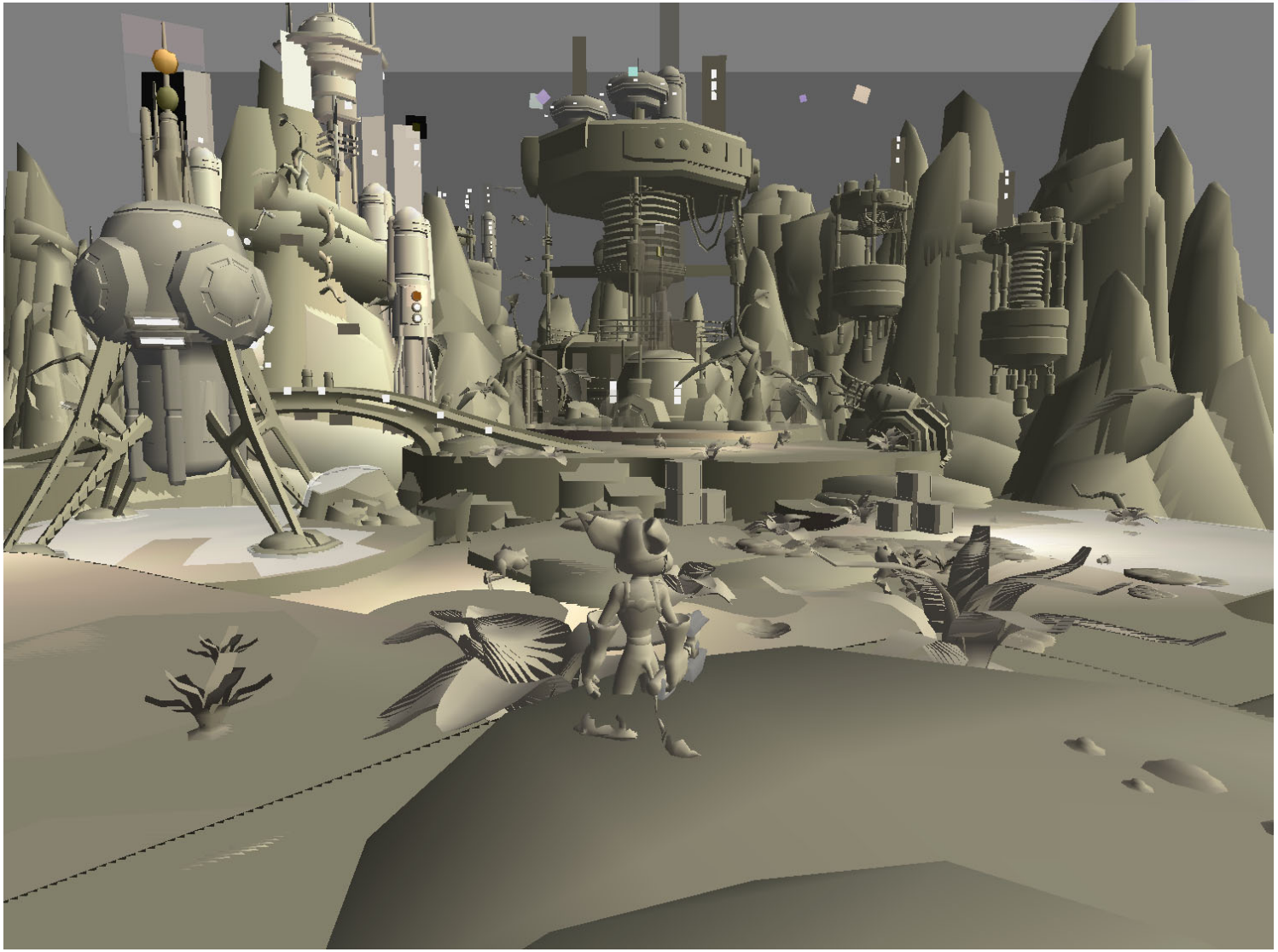
GIF Packet Viewer (OpenGL)

- We know exactly what went through the GIF (GS register settings)
- We're able to rebuild a scene drawn or any part of it, closely simulating the GS
- The cool part: We're able to get more out of it than simply redrawing the scene!

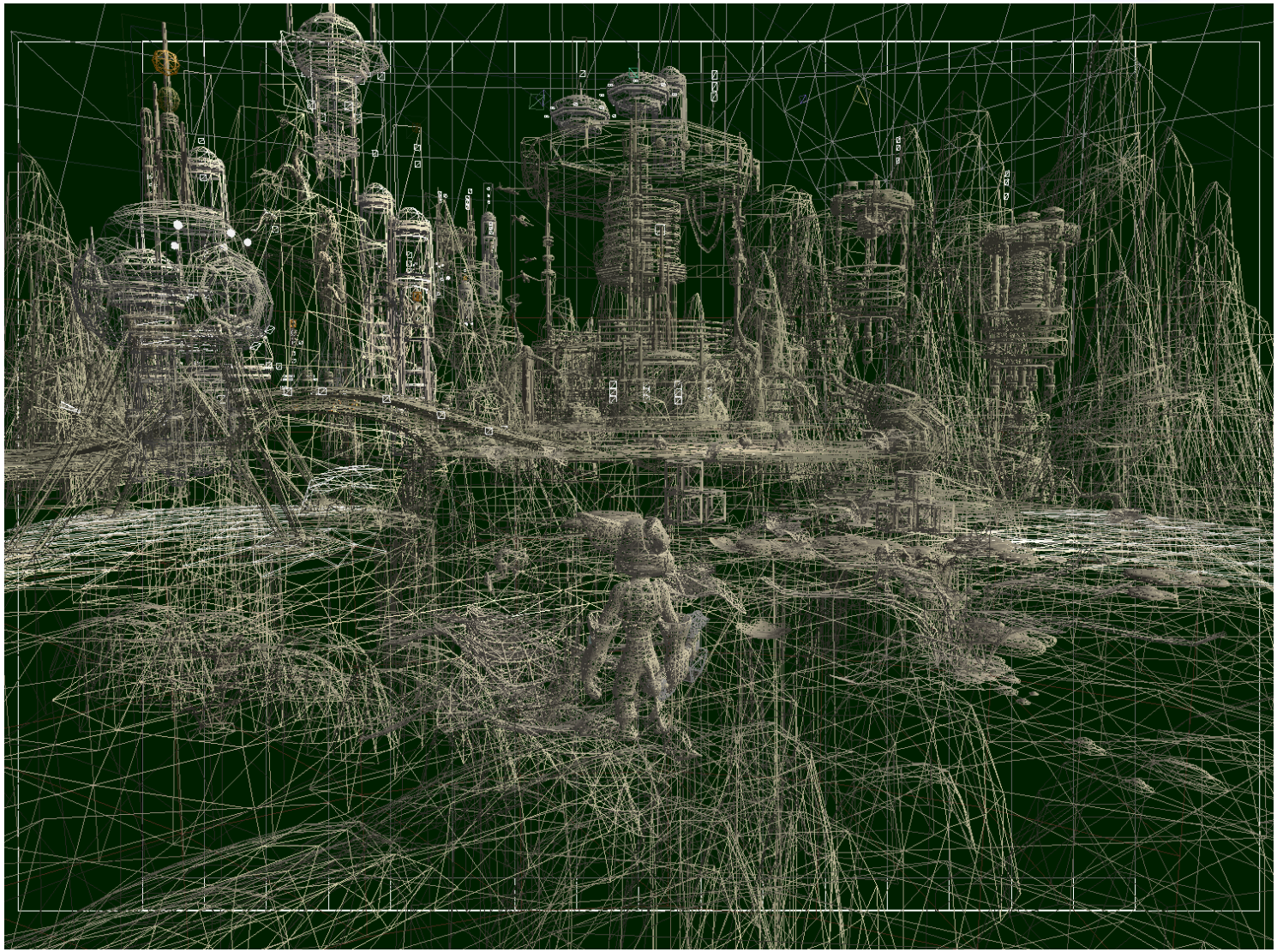
A Look at a PA Snapshot



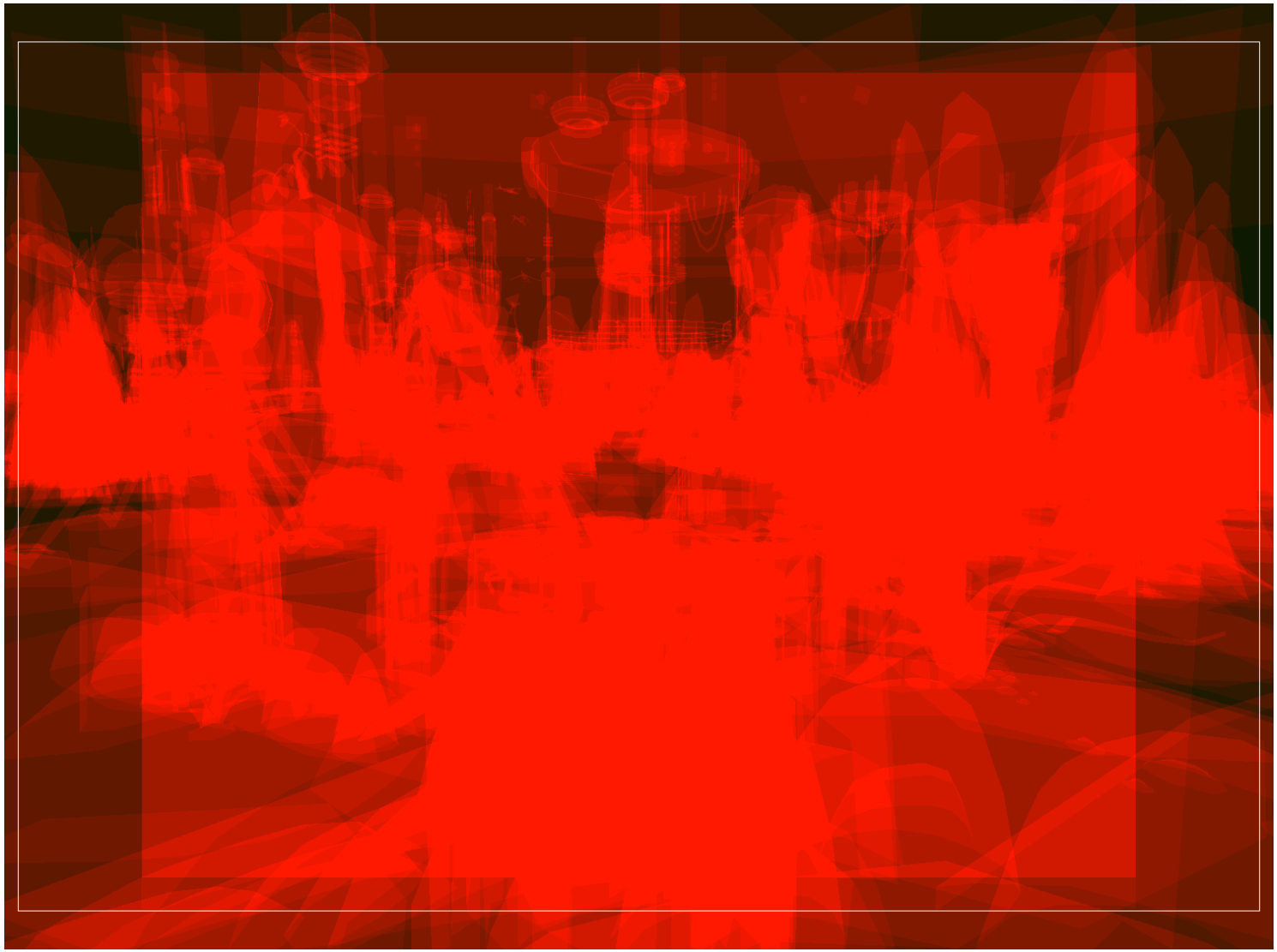
Example 1: Normal Mode



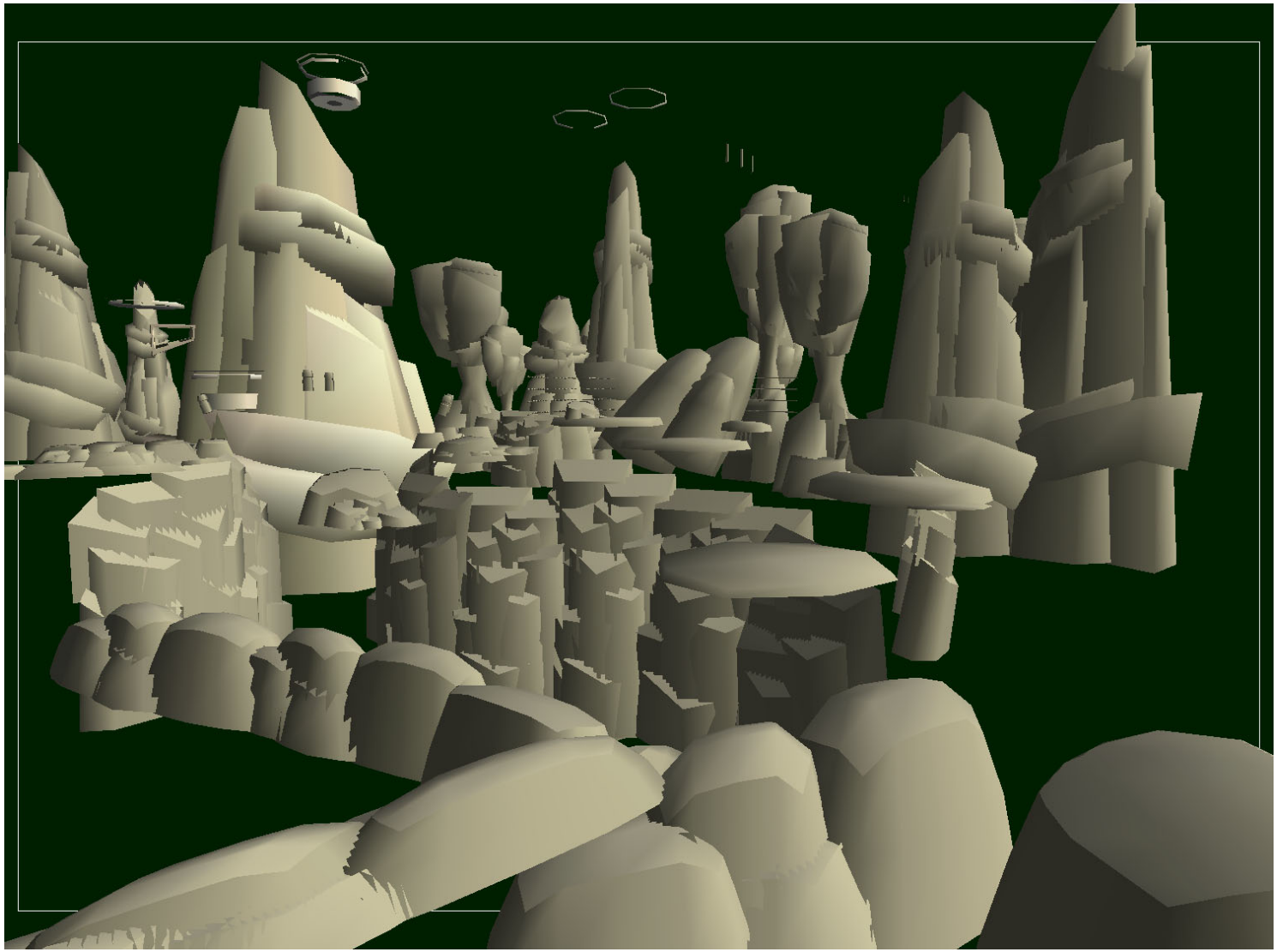
Example 2: Wire Frame Mode



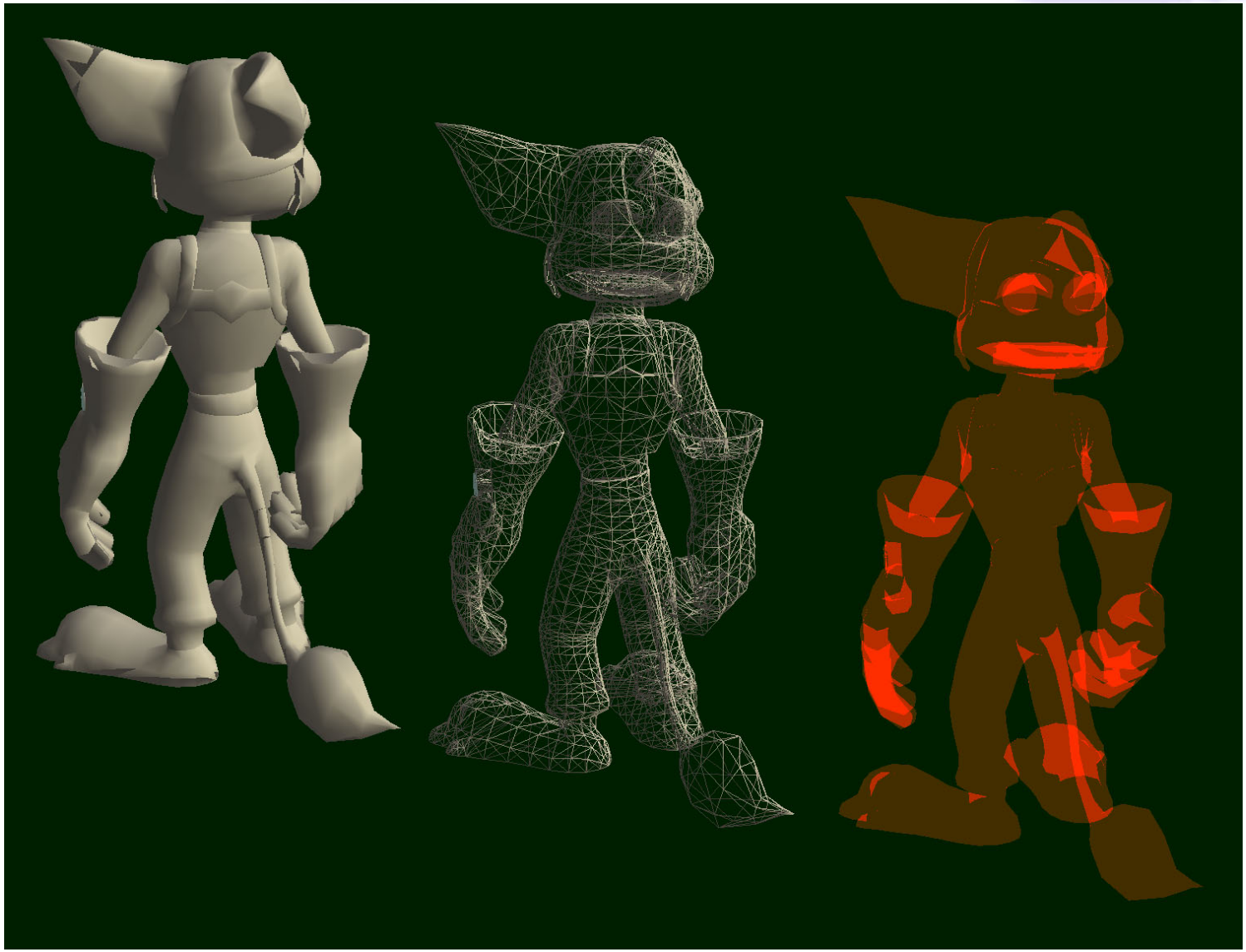
Example 3: Overdraw Mode



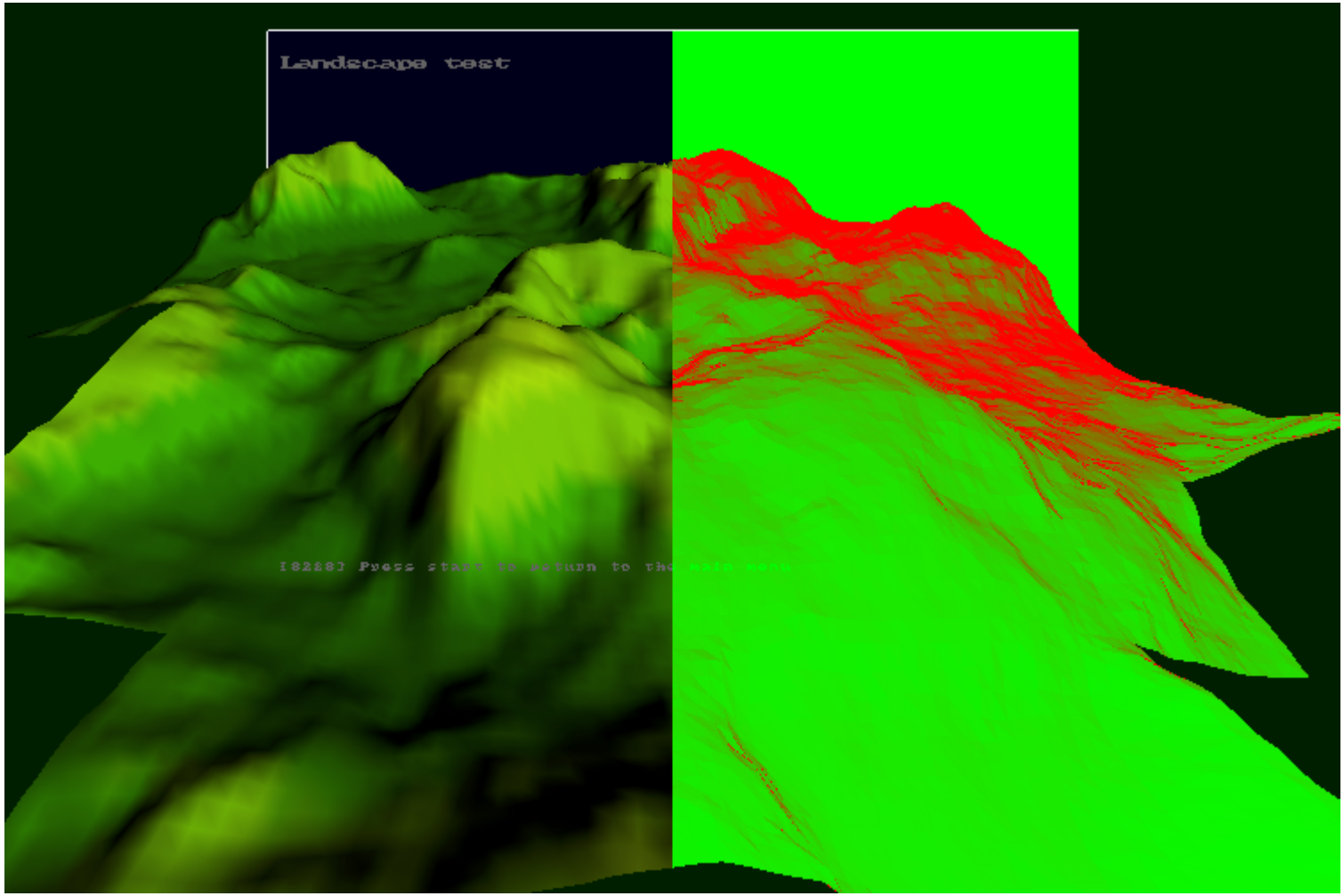
Example 4: Bracketing VU0 Activity



Example 5: Main Character



Example 6: Texel-to-Pixel Ratio Mode



Much More is Possible!

- Drawing order
- Primitive size
- Number of page misses generated
- Number of texture read generated
- 3D view of a scene
- Etc…

GIF Packet Viewer (PS2 GS)

- Sends captured GIF data to an actual tool's GS via DECI2
- Exactly recreates rendering, assuming texture uploads were dynamic
- Support textures (unlike the OpenGL one)
- Drawing may be done in slow motion

How to get a hold of a PA

- **MUST be a licensed PlayStation®2 developer!**
- **Contact SCEA/SCEE/SCEI developer support group**
- **Price and availability: TBA!**

Conclusion

- PlayStation®2 Performance Analyzer is useful at every stage of development
- But use it as early as possible!
  - Starting as early as engine design stage
- It helps you make full use of the hardware
- Make an appointment or send in a disc
  - Form on the developer support website
  - Sessions at SCEA office, PS2 DevCon and GDC

Questions?

- By email
  - geoff_audy@playstation.sony.com
  - kirk_bender@playstation.sony.com
- Pass by our booth and talk to us!
  - "The big Sony booth"
- This presentation available at:
  - http://research.scea.com