Sony Computer Entertainment US Research and Development
http://www.research.scea.com

Introduction to the graphics pipeline of the PS3

: : Cedric Perthuis

Introduction

- An overview of the hardware architecture with a focus on the graphics pipeline, and an introduction to the related software APIs
- Aimed to be a high level overview for academics and game developers
- No announcement and no sneak previews of PS3 games in this presentation

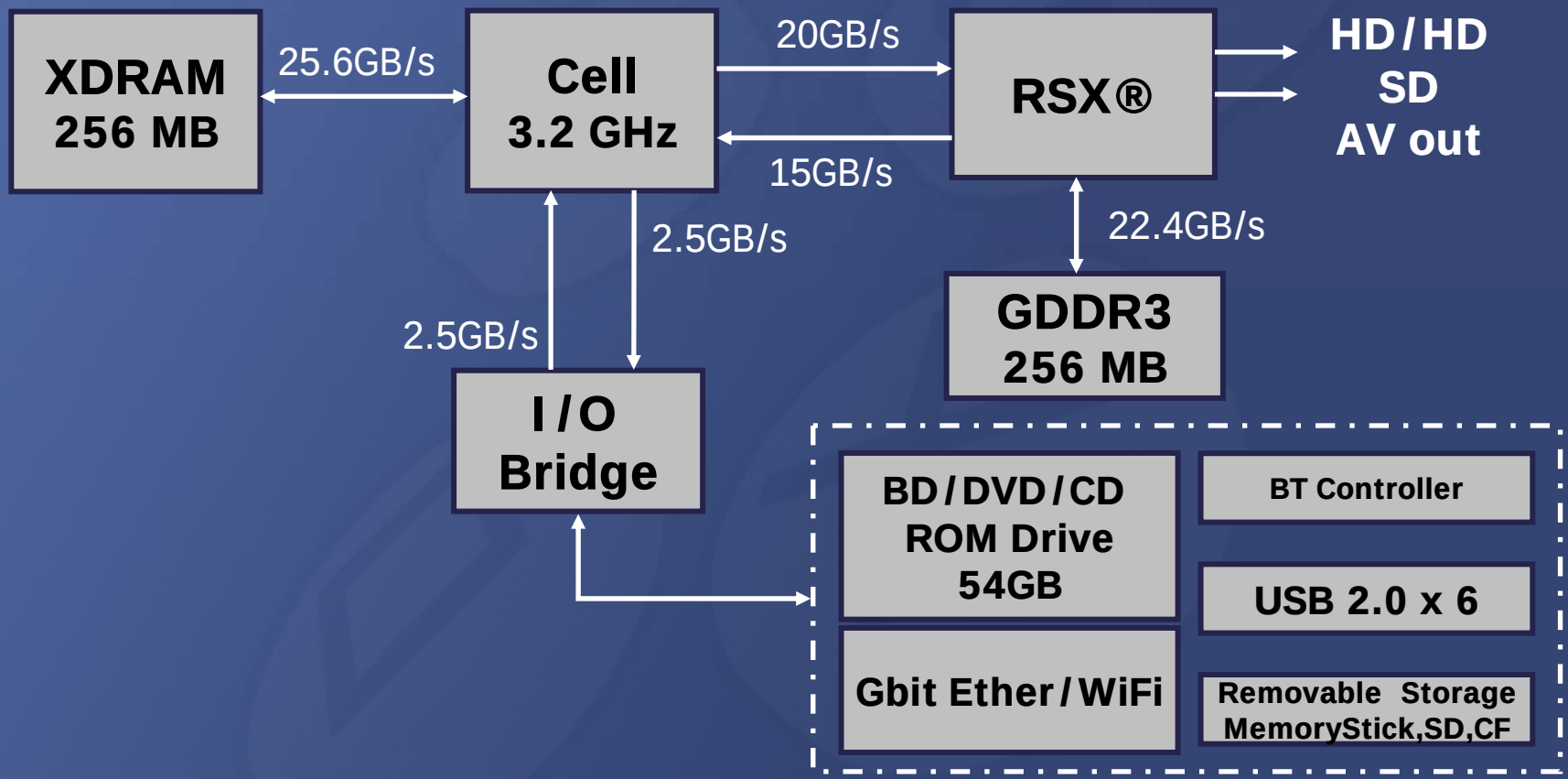
Outline

- Platform Overview
- Graphics Pipeline
- APIs and tools
- Cell Computing example
- Conclusion

Platform overview

- Processing
  - 3.2Ghz Cell: PPU and 7 SPUs
    - PPU: PowerPC based, 2 hardware threads
    - SPUs: dedicated vector processing units
  - RSX®: high end GPU
- Data flow
  - IO: BluRay, HDD, USB, Memory Cards, GigaBit ethernet
  - Memory: main 256 MB, video 256 MB
  - SPUs, PPU and RSX® access main via shared bus
  - RSX® pulls from main to video

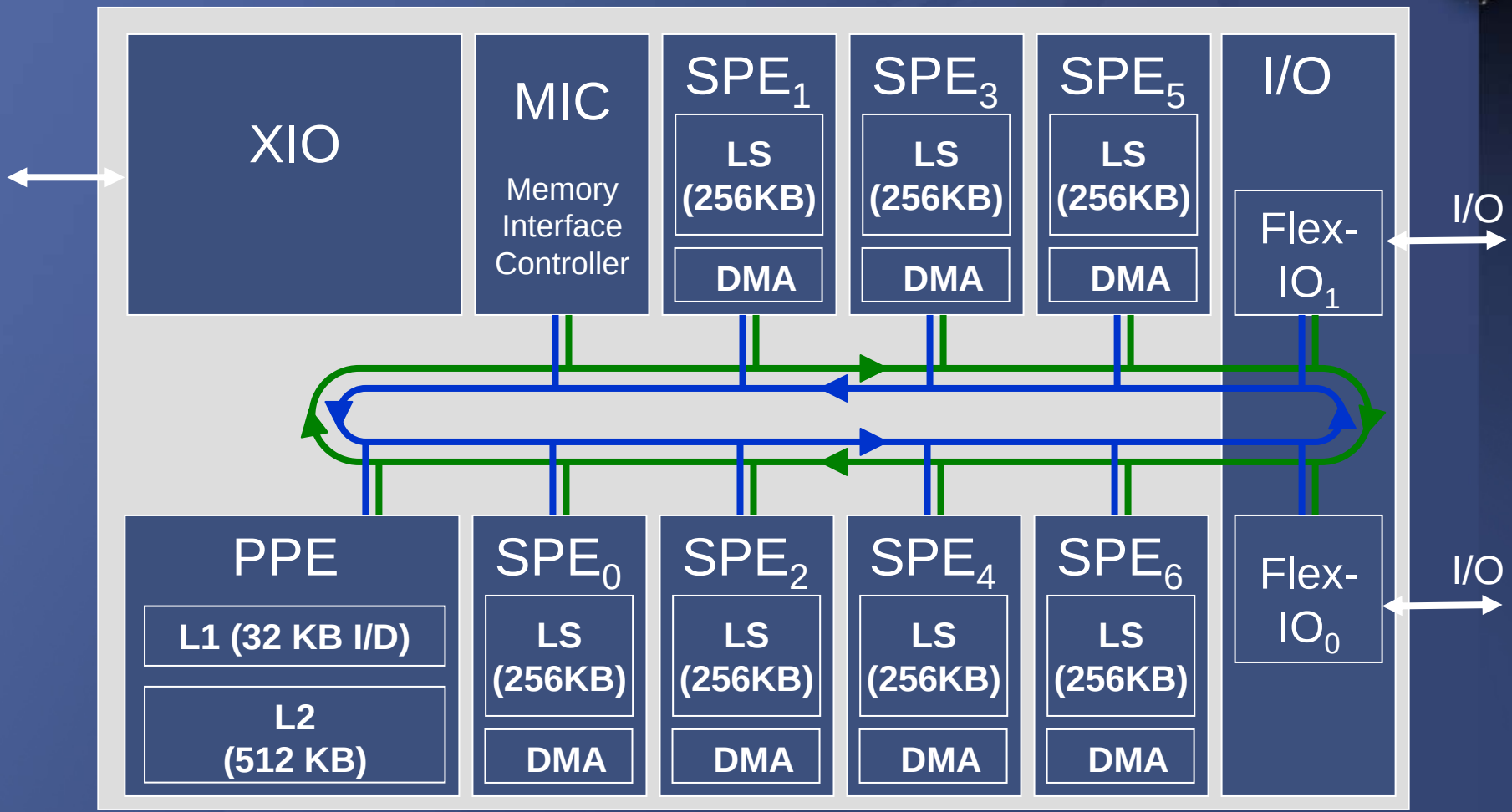
PS3 Architecture



Focus on the Cell SPUs

- The key strength of the PS3
  - Similar to PS2 Vector Units, but order of magnitude more powerful
  - Main Memory Access via DMA: needs software cache to do generic processing
  - Programmable in C/C++ or assembly
  - Programs: standalone executables or jobs
- Ideal for sound, physics, graphics data preprocessing, or simply to offload the PPU

The Cell Processor



The RSX® Graphics Processor

- Based on a high end NVidia chip
  - Fully programmable pipeline: shader model 3.0
  - Floating point render targets
  - Hardware anti-aliasing (2x, 4x)
  - 256 MB of dedicated video memory
- PULL from the main memory at 20 GB/s
- HD Ready (720p/1080p)
  - 720p = 921 600 pixels
  - 1080p = 2 073 600 pixels

  ➔ a high end GPU adapted to work with the Cell Processor and HD displays

The RSX® parallel pipeline

- Command processing
  - Fifo of commands, flip and sync
- Texture management
  - System or video memory
  - storage mode, compression
- Vertex Processing
  - Attribute fetch, vertex program
- Fragment Processing
  - Zcull, Fragment program, ROP

Particle system example on PS3 Hardware

- Objective: to update a particle system
  - The PPU prepares the rendering
    - schedule SPU jobs to compute batches of particles
    - push RSX® commands to pull the VBO from the main memory
    - make the render call
  - The SPUs fill a VBO with positions, normals, etc
    - receive a job
    - compute particles properties
    - DMA the result directly to VBO
    - release RSX® semaphore

➔ fundamental hardware difference with other platforms: the SPUs are part of the pipeline

API differences with the PC approach

- Pass-through driver
  - no driver level optimization, no batching, no shader modification
- direct access to RSX® via memory mapped “registers”
  - restricted to the system
- deferred access to RSX® via a fifo of commands
  - system and user

PSGL: the high level graphics API

- Needed a standard: practical and extensible
  → the choice was OpenGL ES 1.0
- Why not a subset of OpenGL ?
  - Mainly needed conformance tests
- Benefits:
  - pipeline state management
  - Vertex arrays
  - Texture management
  - Bonus: Fixed pipeline
  - Only ~20 entry points for fixed pipeline
  - Fog, light, material, texenv
- Inconvenience:
  - Fixed point functions
  - No shaders: needed to be added

PSGL: modern GPU extensions

- OpenGL ES 1.1
  - VBO
  - FBO
  - PBO
  - Cube Map, texgen
- Primitives:
  - Quads, Quads_strips
  - primitive restart
  - Instancing
- Queries and Conditional Rendering
- More data types
  - ex: half_float
- Textures:
  - Floating point textures
  - DXT
  - 3D
  - non power of 2
  - Anisotropic filtering, Min/Max LOD, LOD Bias
  - Depth textures
  - Gamma correction
  - Vertex Texture

PSGL: PS3 specific extensions

- Synchronizations:
  - Wait on or check GPU progress
  - Make the GPU wait on another GPU event or on PPU
  - Provide sync APIs for PPU and for SPU
- Memory usage hints
  - For texture, VBO, PBO, render-targets
- PPU specific extensions:
  - Embedded system: PPU usage needs to be limited, some extensions are added to decrease the PPU load for some existing features:
    - Ex: Attribute set

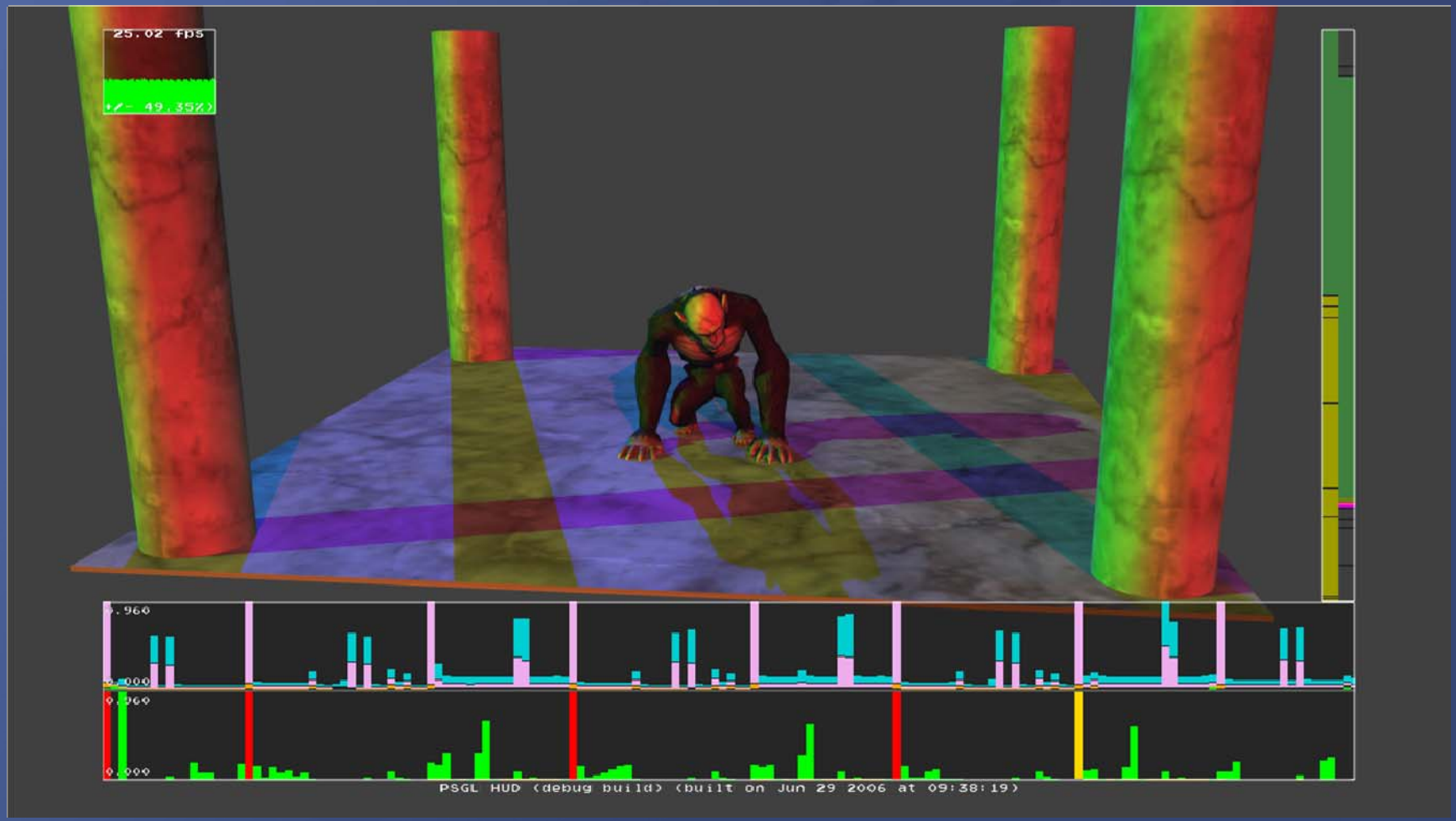
Shading language

- CG: high level shader language
  - Support Cg 1.5
  - PS3 specific compiler
  - Mostly compatible with other languages like HLSL
  - Tools: FX composer for PS3
- CG: runtime
  - Direct access to shader engine registers or via CG parameter
  - shared and unshared parameters
  - CG FX runtime: techniques, render states, textures

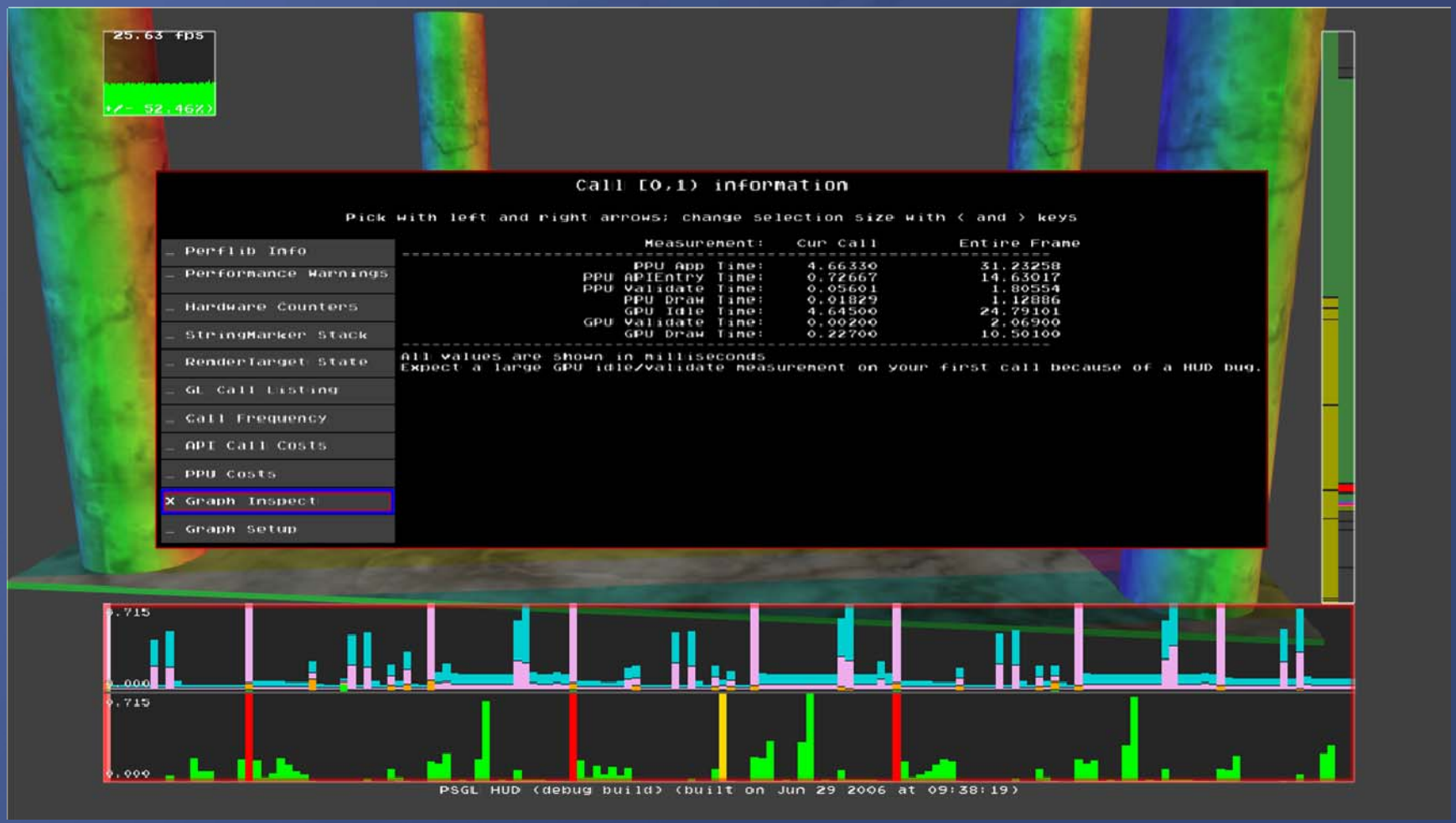
Performance analysis

- PSGL HUD: runtime performance analyzer
  - display global statistics and hardware counters
  - explore objects in video and main memory
  - explore individual draw calls
  - profile graphics API calls

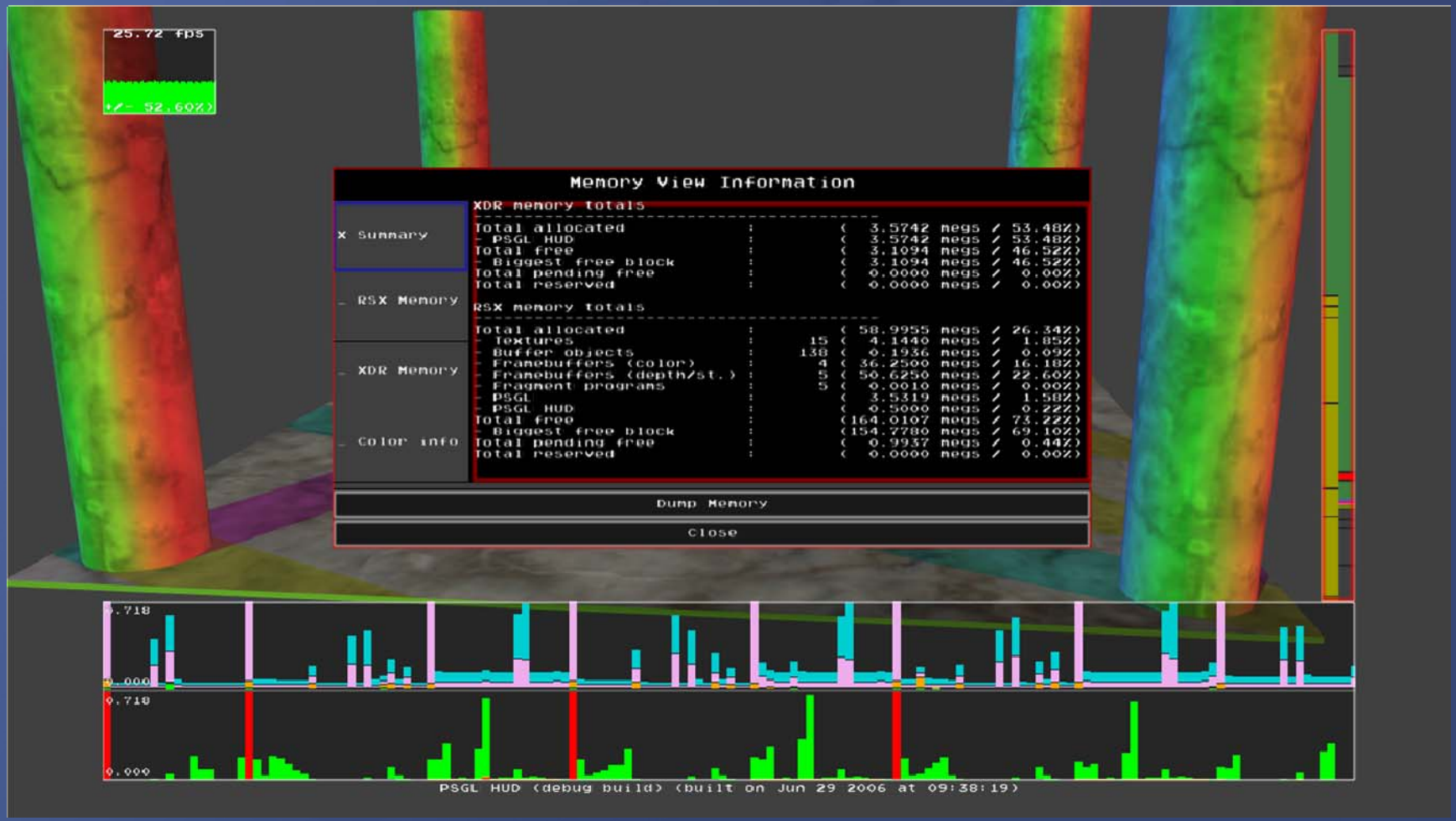
PSGL HUD


25.02 fps
+/- 49.35%)
PSGL HUD (debug build) (built on Jun 29 2006 at 09:38:19)

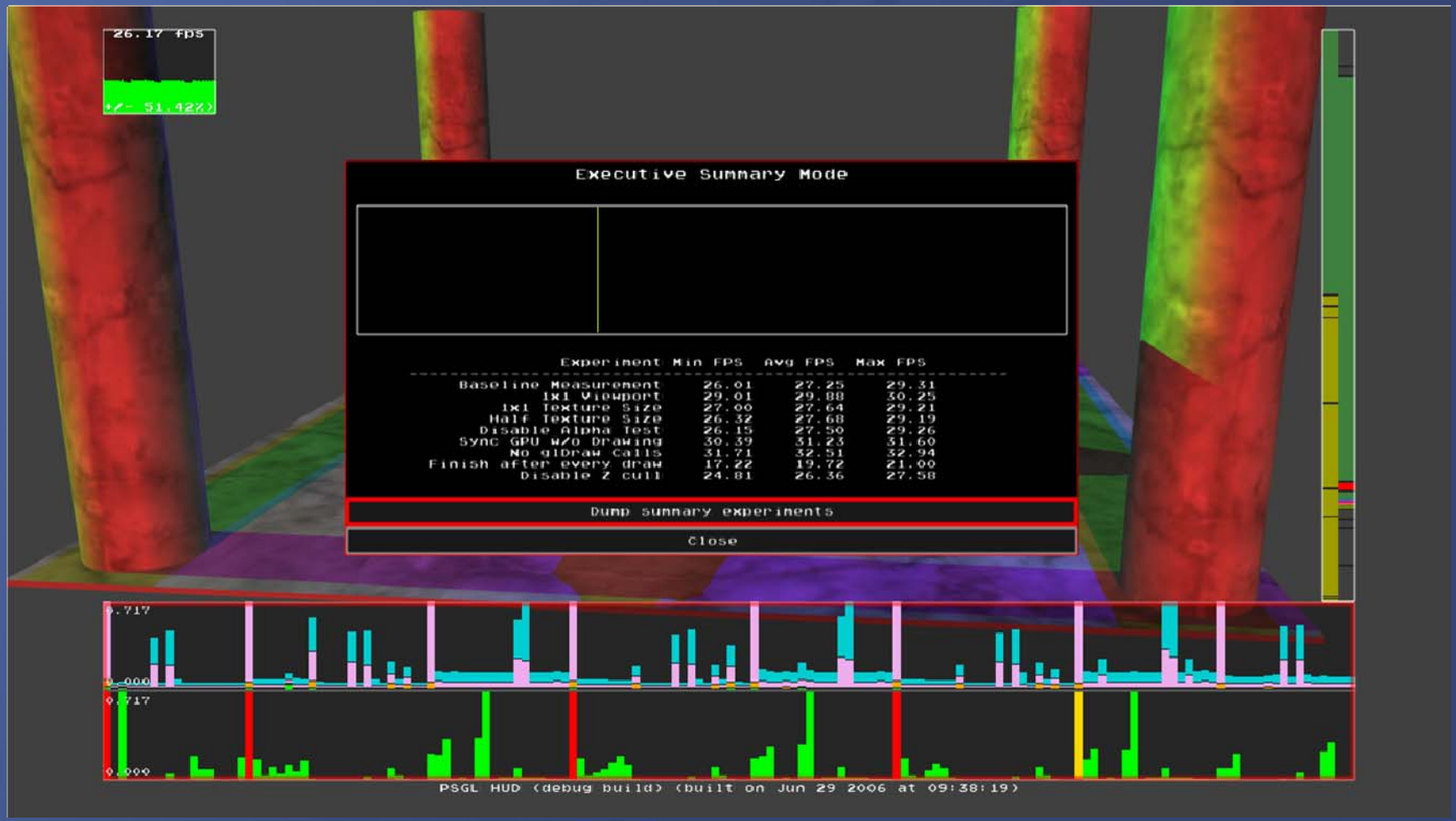
Call View


25.63 fps
+/- 52.46%
Call [0,1) information
Pick with left and right arrows; change selection size with < and > keys
Perflib Info
Performance Warnings
Hardware Counters
StringMarker Stack
RenderTarget State
GL Call Listing
Call Frequency
API Call Costs
PPU Costs
X Graph Inspect
Graph Setup
Measurement: Cur Call Entire Frame
PPU App Time: 4.66330 31.23258
PPU APIEntry Time: 0.72667 14.63017
PPU Validate Time: 0.05601 1.80554
PPU Draw Time: 0.01829 1.12886
GPU Idle Time: 4.64500 24.79101
GPU Validate Time: 0.00200 2.06900
GPU Draw Time: 0.22700 10.50100
All values are shown in milliseconds
Expect a large GPU idle/validate measurement on your first call because of a HUD bug.
PSGL HUD (debug build) (built on Jun 29 2006 at 09:38:19)

Memory view


25.72 fps
+/- 52.60%)
Memory View Information
x Summary
RSX Memory
XDR Memory
Color info
XDR memory totals
Total allocated : (3.5742 megs / 53.48%)
- PSGL HUD : (3.5742 megs / 53.48%)
Total free : (3.1094 megs / 46.52%)
- Biggest free block : (3.1094 megs / 46.52%)
Total pending free : (0.0000 megs / 0.00%)
Total reserved : (0.0000 megs / 0.00%)
RSX memory totals
Total allocated : (58.9955 megs / 26.34%)
- Textures : 15 (4.1440 megs / 1.85%)
- Buffer objects : 138 (0.1936 megs / 0.09%)
- Framebuffers (color) : 4 (36.2500 megs / 16.18%)
- Framebuffers (depth/st.) : 5 (50.6250 megs / 22.60%)
- Fragment programs : 5 (0.0010 megs / 0.00%)
- PSGL : (3.5319 megs / 1.58%)
- PSGL HUD : (0.5000 megs / 0.22%)
Total free : (164.0107 megs / 73.22%)
- Biggest free block : (154.7780 megs / 69.10%)
Total pending free : (0.9937 megs / 0.44%)
Total reserved : (0.0000 megs / 0.00%)
Dump Memory
Close
PSGL HUD (debug build) (built on Jun 29 2006 at 09:38:19)

Executive summary



Beyond High Level APIs

- A low level graphics API exists:
  - proprietary
  - small and simple
  - let the user create and send command buffers
  - deep knowledge of the RSX® internals needed to really take full advantage of it

A leap forward in graphics

- Gamer expectations have changed:
  - Higher resolutions
  - Deeper colors
  - Larger and deeper environment
  - More environmental and lighting effects
- Game console developer expectations have changed too

Typical PS2 title graphics budget

- Assets
  - 60 000 polygons
    - 5 years old HW, at that time PC games were around 30 000 polys, it's only with GF3 that gamers started seeing 100 000 polys in games.
    - compare to 480p FB: 1 poly for 4 pixels
  - 10 MB of 8 bits or 4 bits textures
- Rendering
  - Multi pass for lightmaps
  - Multi pass for specular
  - Projected shadow

Typical Next Gen graphics budget

- Assets
  - 800 000 polygons : compare to 720p FB
  - 150 MB of textures in video memory
- Rendering
  - Z pass
  - 2 shadow maps 1024x1024: blur
  - color and lighting pass: diffuse, normal, specular, 4xAA
  - Post effects: blooming, tone mapping,…

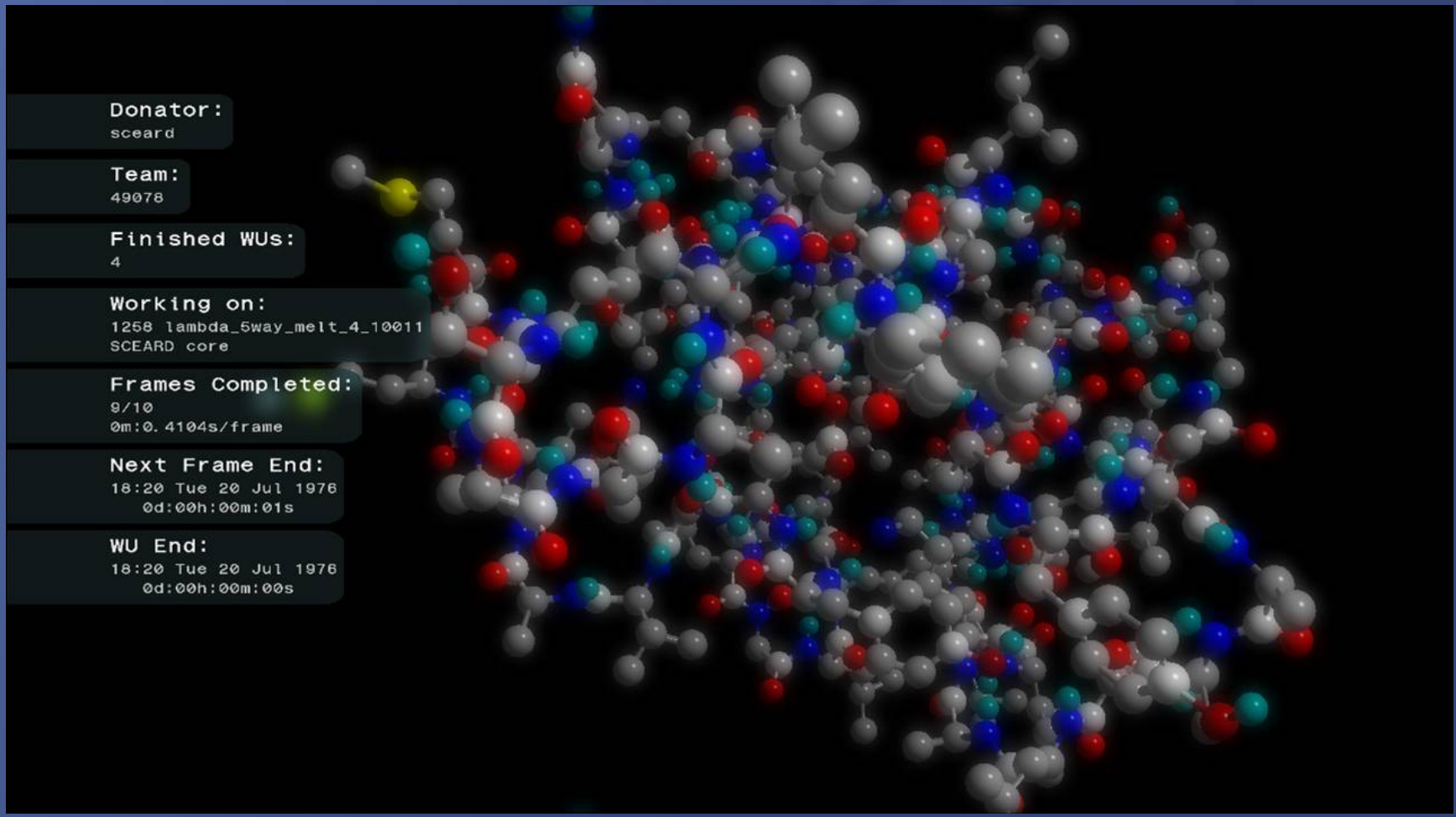
→ Maximized Framebuffer Read/Write bandwidth

→ 20 millions+ rasterized pixels

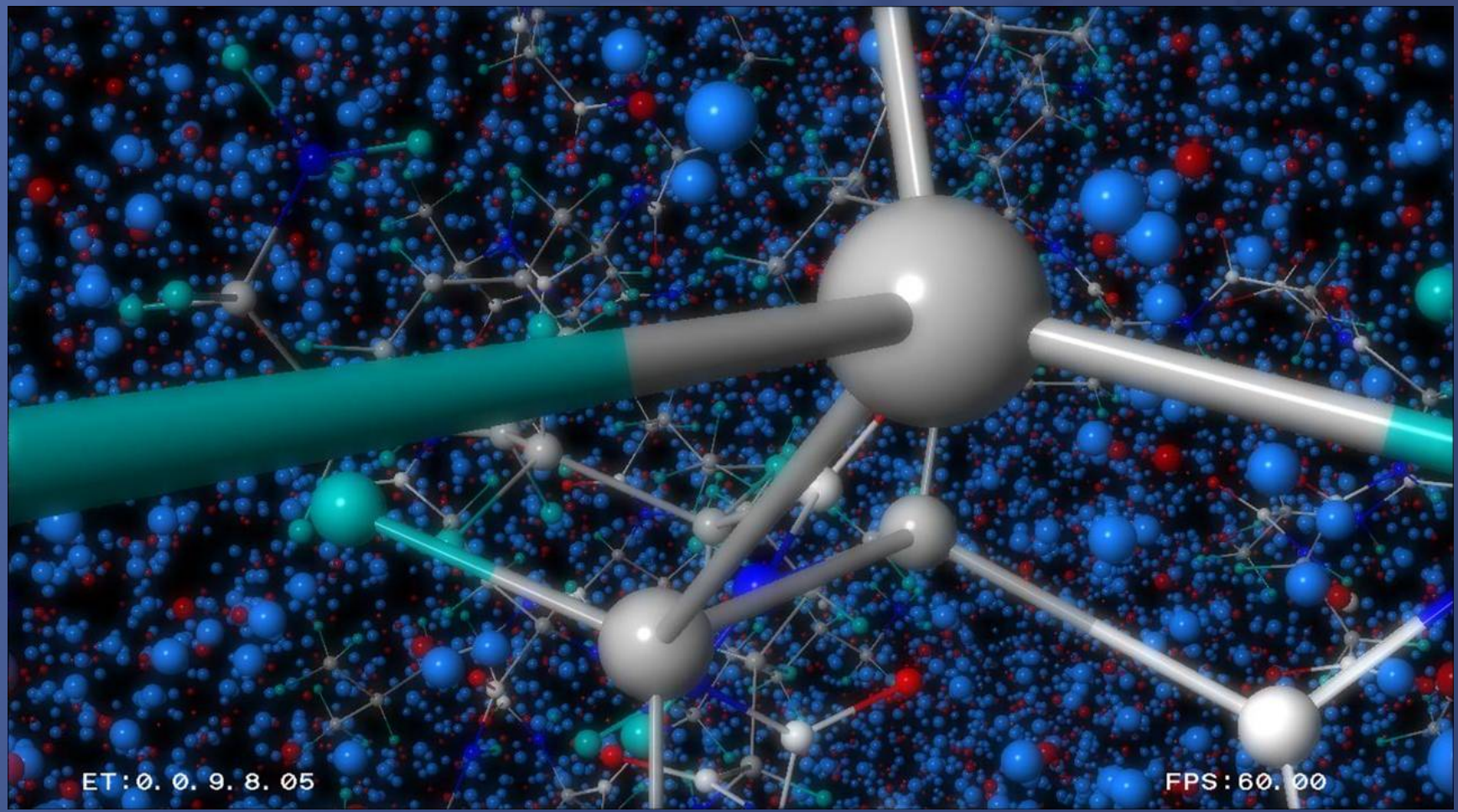
Example of intensive computing and visualization on PS3

- Cure@PS3
  - Project Folding @ home : provides a PC client
  - PS3 client created in few months by SCE
  - presented at the Game Convention 2006 in Leipzig
  - intensive computing application for PS3
    - maximize SPU processing
    - PPU schedules jobs
  - visualization on PS3
    - Arbitrary complex molecule rendering challenge
    - Geometries generated in the fragment program
    - PSGL MRTs

Cure@PS3: protein



Cure@PS3: protein + water



Cure@PS3 : what if...

- What if it became a PS3 screensaver ?
- Running on 1% of the PS3 sold during the 1st month

→ Estimation: x2 the current Folding @ home computing power of 210 T flops

→ Up to 20 times faster than a PC

Conclusion

- Thank you for attending
- Questions ?

- Cedric_Perthuis @ playstation.sony.com