idc15

Imagination Developers Connection

PowerVR Graphics - Latest Developments and Future Plans

Latest Developments and Future Plans

A brief introduction

- **Joe Davis**
  - Lead Developer Support Engineer, PowerVR Graphics
  - With Imagination's PowerVR Developer Technology team for ~6 years

- **PowerVR Developer Technology**
  - SDK, tools, documentation and developer support/relations (e.g. this session ☺)

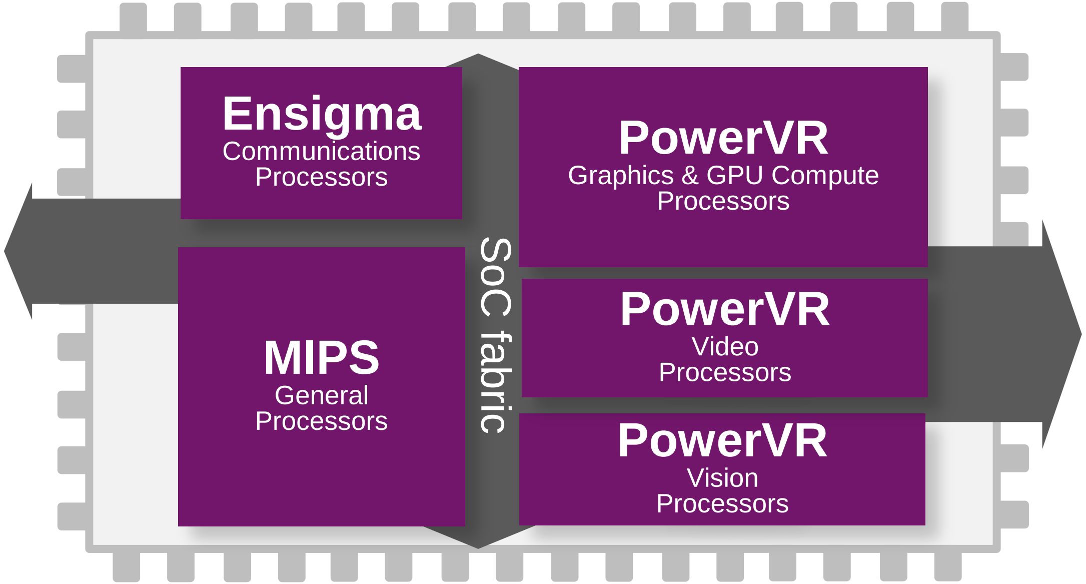
Company overview

About Imagination

Multimedia, processors, communications and cloud IP

Driving IP innovation with unrivalled portfolio

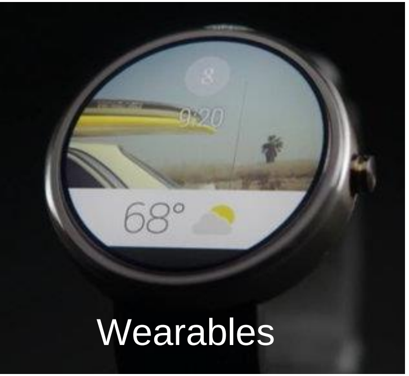
- Recognised leader in graphics, GPU compute and video IP
- #3 design IP company world-wide*



* source: Gartner

About Imagination

Our IP plus our partners' know-how combine to drive and disrupt


Wearables

Advanced Automotive

Smart Security

Gaming & VR/AR

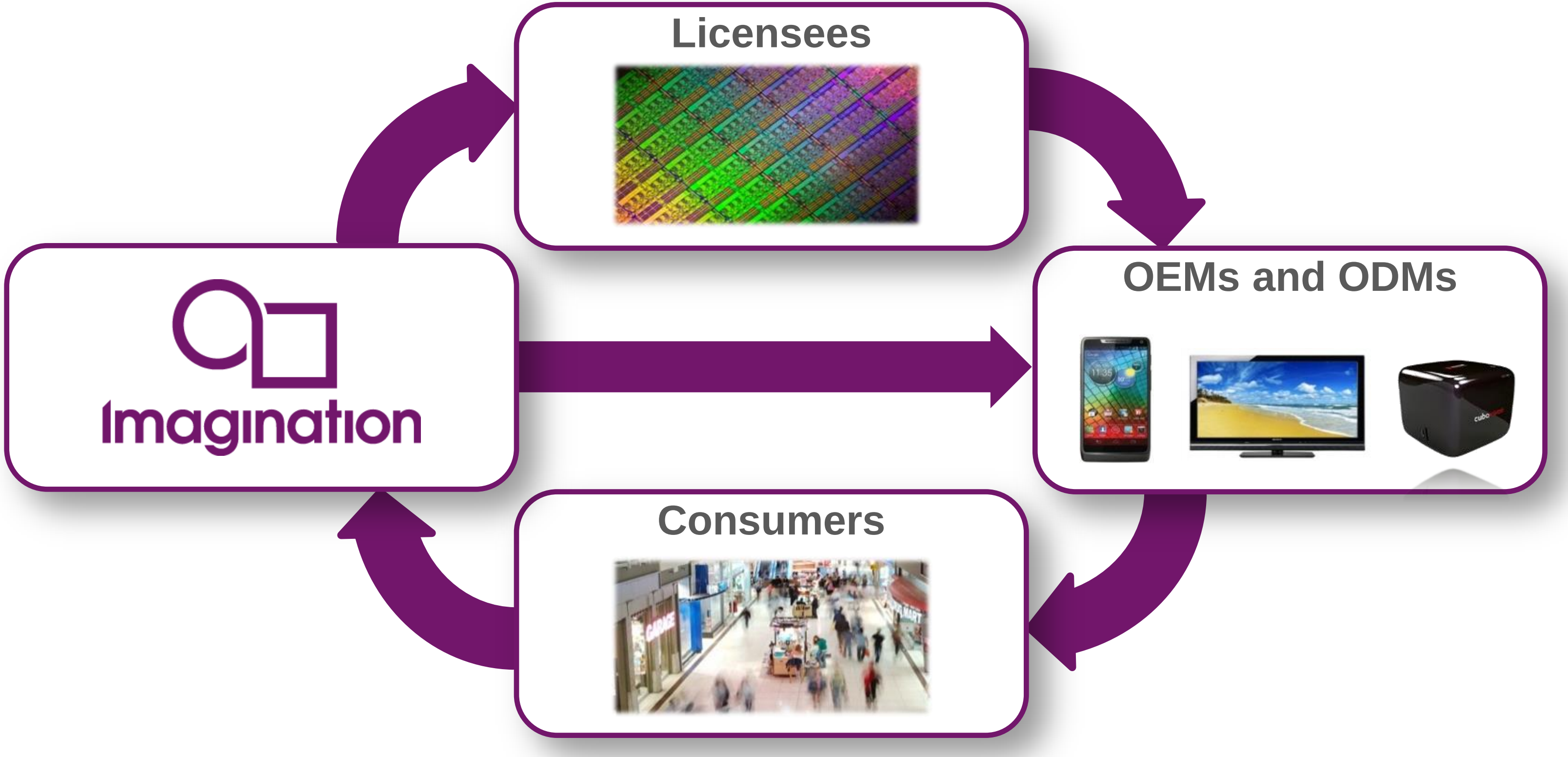
Retail

Smart homes

eHealth

About Imagination

Business model



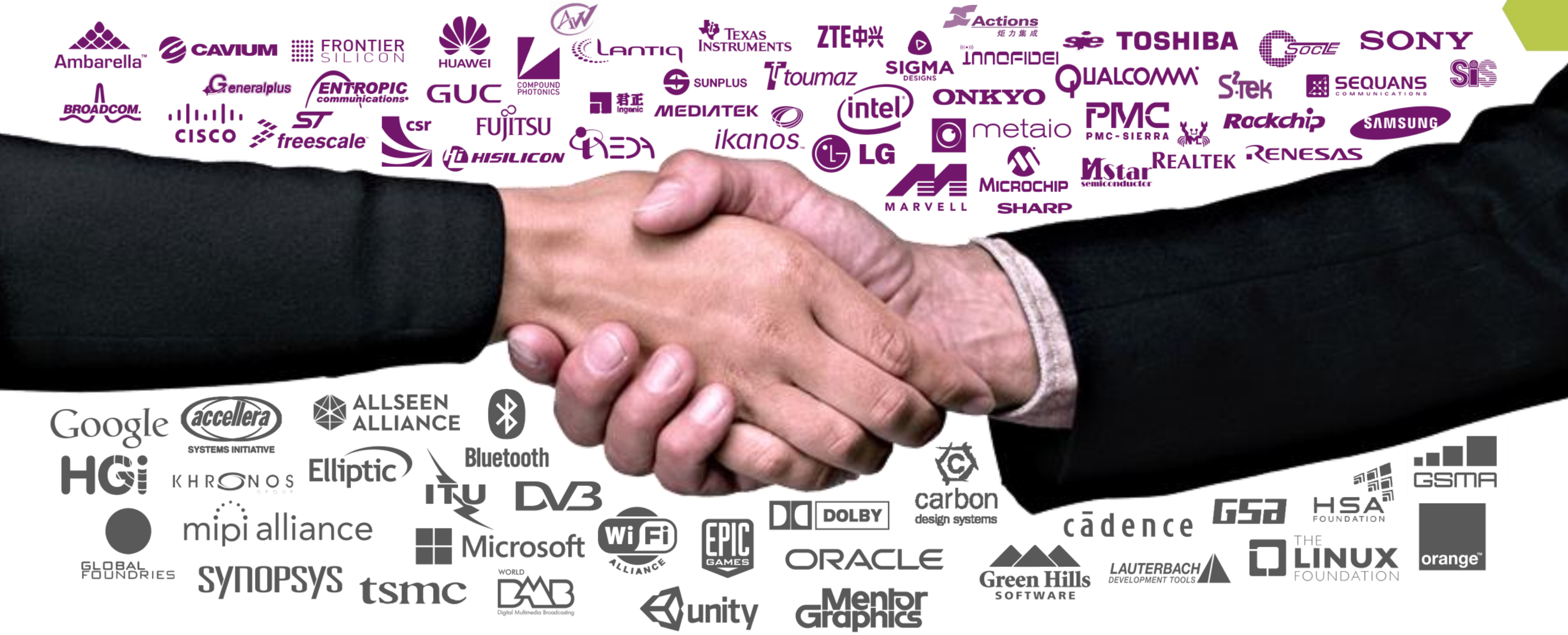
Licensees

OEMs and ODMs

Consumers

About Imagination

Our licensees and partners drive our business



PowerVR Rogue Hardware

PowerVR Rogue

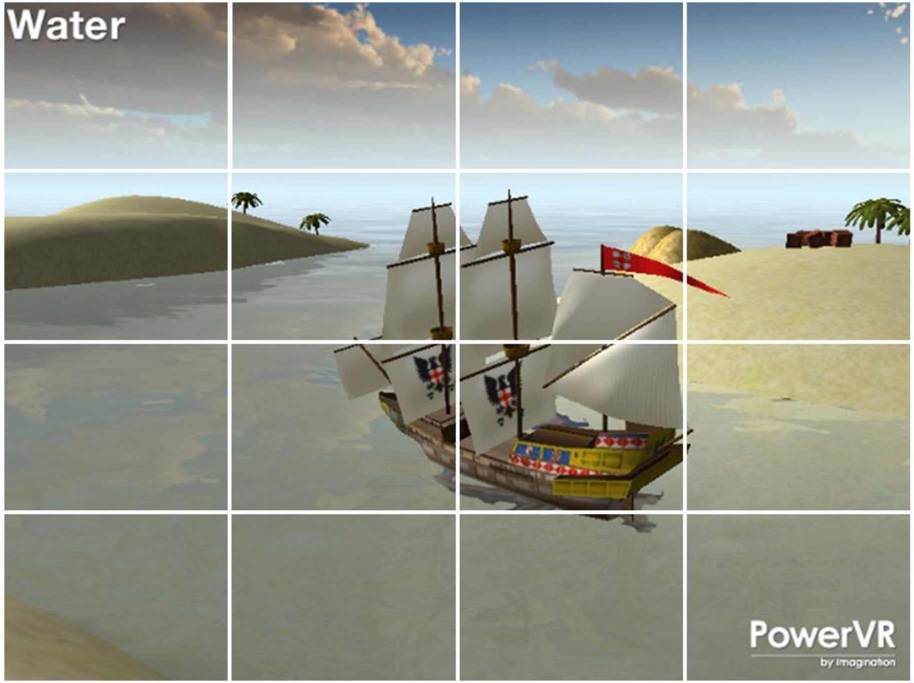
Recap

- **Tile-based deferred renderer**
  - Building on technology proven over 5 previous generations
- **Formally announced at CES 2012**
- **USC - Universal Shading Cluster**
  - New scalar SIMD shader core
  - General purpose compute is a first class citizen in the core …
  - … while not forgetting what makes a shader core great for graphics

TBDR

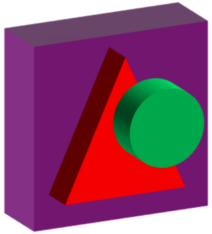
Tile-based

- **Tile-based**
  - Split each render up into small tiles (32x32 for the most part)
  - Bin geometry after vertex shading into those tiles
  - Tile-based rasterisation and pixel shading
  - Keep all data access for pixel shading on chip



TBDR

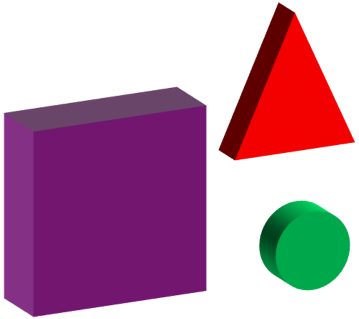
Deferred



- **Deferred rasterisation**
  - Don’t actually get the GPU to do any pixel shading straight away
  - HW support for fully deferred rasterisation and then pixel shading
  - Rasterisation is pixel accurate

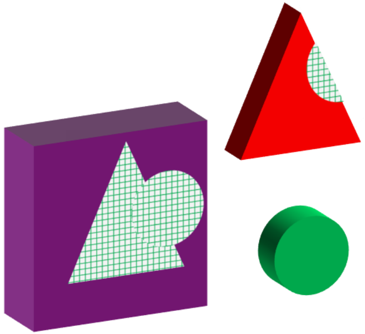
Conventional GPUs

All surfaces filled



PowerVR GPUs

Only visible surfaces filled



TBDR

Bandwidth savings

- **Bandwidth savings across all phases of rendering**
  - Only fetch the geometry needed for the tile
  - Only process the visible pixels in the tile

- **Efficient processing**
  - Maximize available computational resources
  - Do the best the hardware can with bandwidth

TBDR

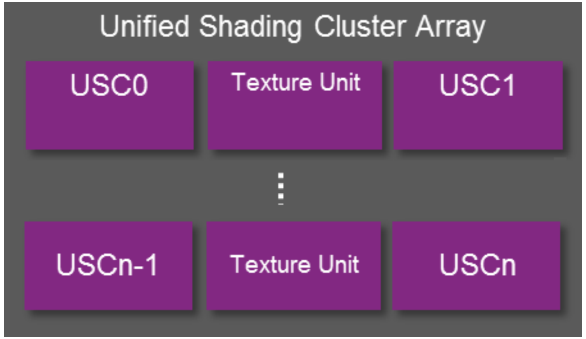
Power savings

- **Maximizing core efficiency**
  - Lighting up the USC less often is always going to be a saving
- **Minimizing bandwidth**
  - Texturing less is a fantastic way to save power
  - Geometry fetch and binning is often more than 10% of per-frame bandwidth
  - Saves bandwidth for other parts of your render

Rogue USC

Architectural Building Block

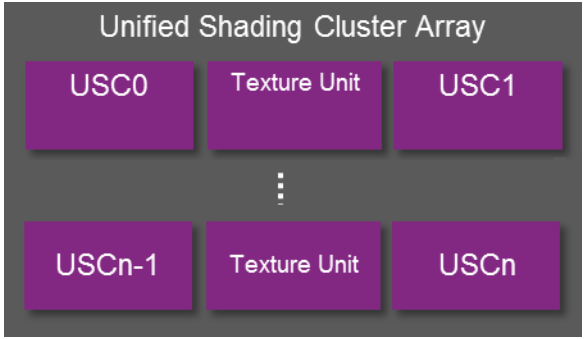
- **Unified Shading Cluster**
  - Basic building block of the Rogue architecture
  - Laid out in pairs, with a shared TPU
- **1, 0.5 and 0.25 USC designs are special**
  - Different balance in the design
  - Tend to find their way into non-gaming applications



Rogue USC

Shader Architecture

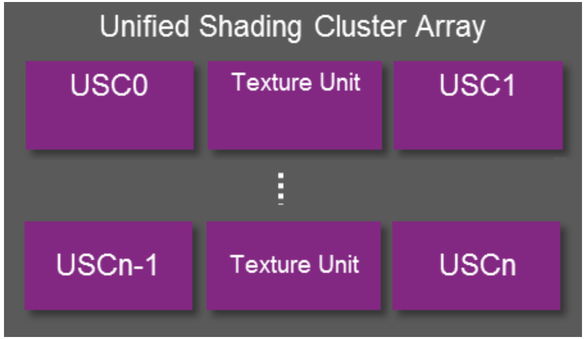
- **16-wide in hardware**
- **32-wide branch granularity**
    - We run half a task/warp per clock
- **Scalar SIMD**
- **Optimized ALU pipeline**
    - Mix of F32, F16, integer, floating point specials, logic ops



Rogue USC

Pipeline datapaths

- **Configurable in the IP core**
  - F16 paths were sometimes optional, thankfully not any more
  - F16 paths performance increased significantly after the first generation
- **Performance in your shader**
  - F32 paths are dual FMAD
  - F16 paths can do different things per cycle depending on shader
  - ISA is available for you to interrogate though, with disassembling compilers



Rogue USC

Scalar

- **Scalar ALUs**
  - Hard to understate what a benefit this is
  - Seems obvious to do, right?
  - Vector architectures are just hard to program well
  - Scalar isn't a free lunch
  - Makes performance a lot more predictable for you

Rogue USC

Programmable output registers

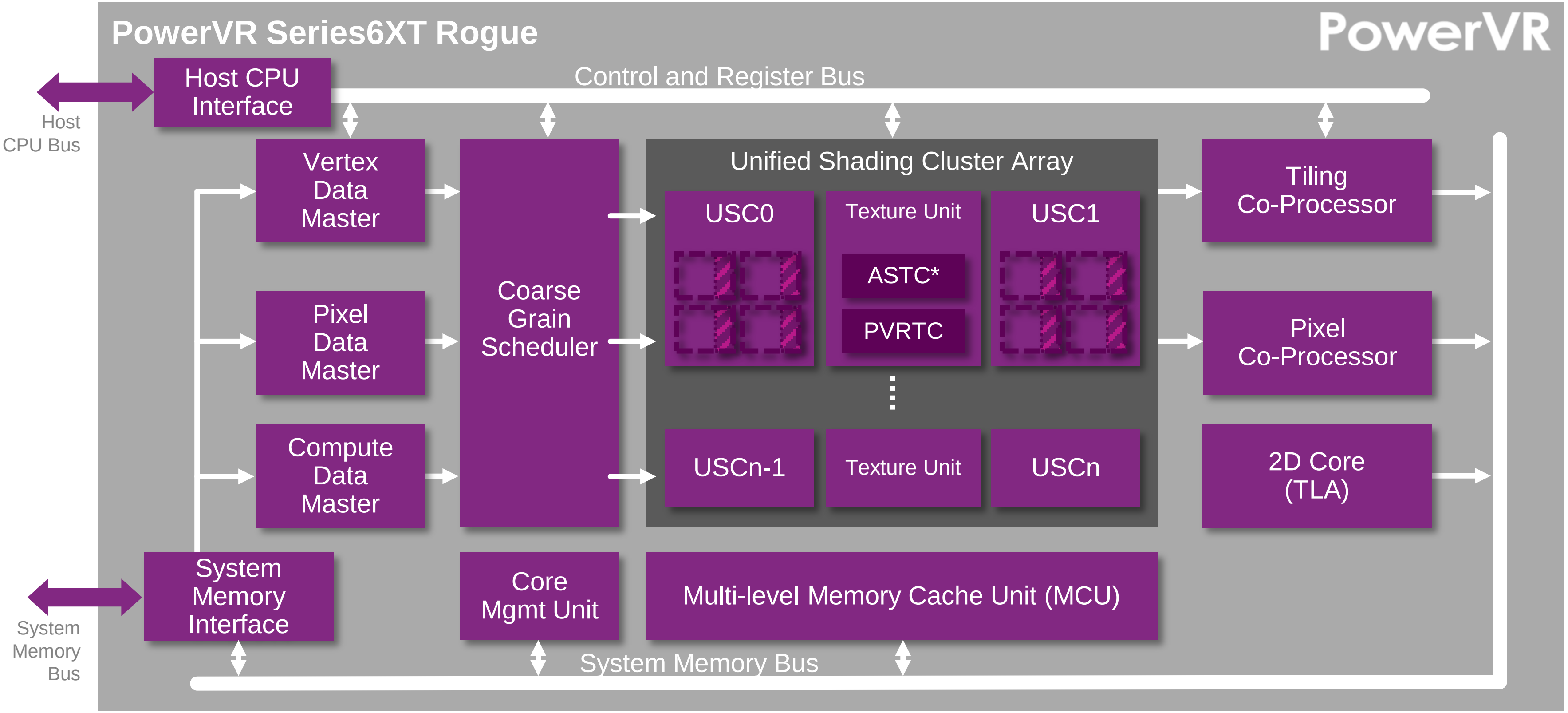
- **The pixel output registers in the ISA are read/write**
- **One per pixel**
- **Width depends on IP core**
- **We expose it programmatically with Pixel Local Storage**
  - Worked closely with ARM (thanks, Jan-Harald!)

Evolution

Health Warning: Really Bad Diagrams™

Rogue Evolution

- Architecture has changed quite a bit over time
- Rogue in 2010 still mostly looks like a Rogue today
- Significant evolutionary changes across the architecture
- Lots of it driven by developers before the IP is baked
- Lots of it driven by also analysing your stuff anyway



PowerVR Series6XT Rogue

PowerVR

Host CPU Bus

Host CPU Interface

Control and Register Bus

Vertex Data Master

Pixel Data Master

Compute Data Master

Coarse Grain Scheduler

Unified Shading Cluster Array

USC0

Texture Unit

ASTC*

PVRTC

USC1

USCn-1

Texture Unit

USCn

Tiling Co-Processor

Pixel Co-Processor

2D Core (TLA)

System Memory Bus

System Memory Interface

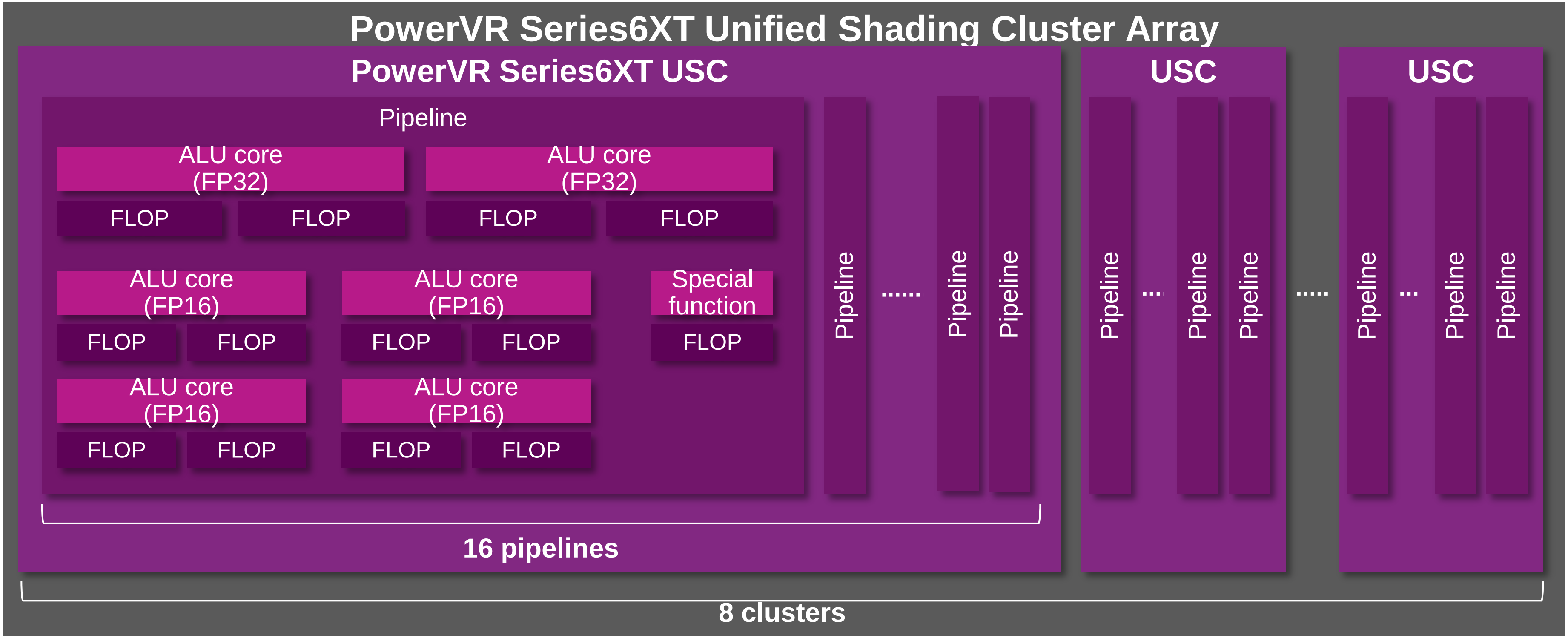
Core Mgmt Unit

Multi-level Memory Cache Unit (MCU)

System Memory Bus

▨ *Extra low power GFLOPS*

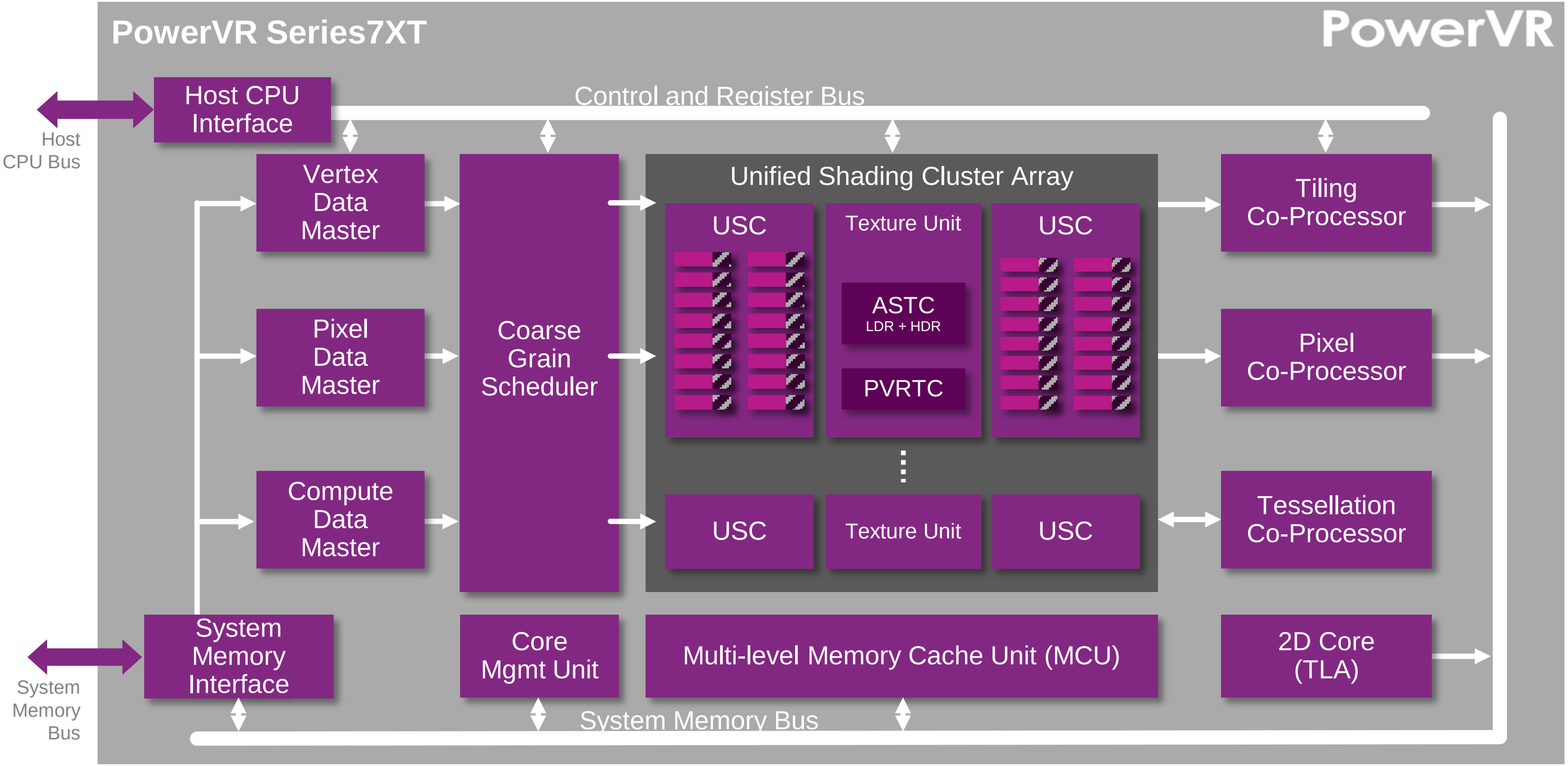
* *Supports both LDR and HDR ASTC formats*


PowerVR Series6XT Unified Shading Cluster Array
PowerVR Series6XT USC
Pipeline
ALU core (FP32)
FLOP
FLOP
ALU core (FP32)
FLOP
FLOP
ALU core (FP16)
FLOP
FLOP
ALU core (FP16)
FLOP
FLOP
Special function
FLOP
ALU core (FP16)
FLOP
FLOP
ALU core (FP16)
FLOP
FLOP
Pipeline
......
Pipeline
Pipeline
16 pipelines
USC
Pipeline
...
Pipeline
Pipeline
.....
USC
Pipeline
...
Pipeline
Pipeline
8 clusters

Series6 to Series6XT

Lots of lessons learned

- Improved scheduler
- Streamlined ISA
- Improved compute task efficiency
- Completely new F16 datapath
- Improved front-end for sustained geometry performance
- ASTC



PowerVR Series7XT

PowerVR

Host CPU Bus

Host CPU Interface

Control and Register Bus

Vertex Data Master

Pixel Data Master

Compute Data Master

Coarse Grain Scheduler

Unified Shading Cluster Array

USC

Texture Unit

ASTC LDR + HDR

PVRTC

USC

USC

Texture Unit

USC

Tiling Co-Processor

Pixel Co-Processor

Tessellation Co-Processor

System Memory Bus

System Memory Interface

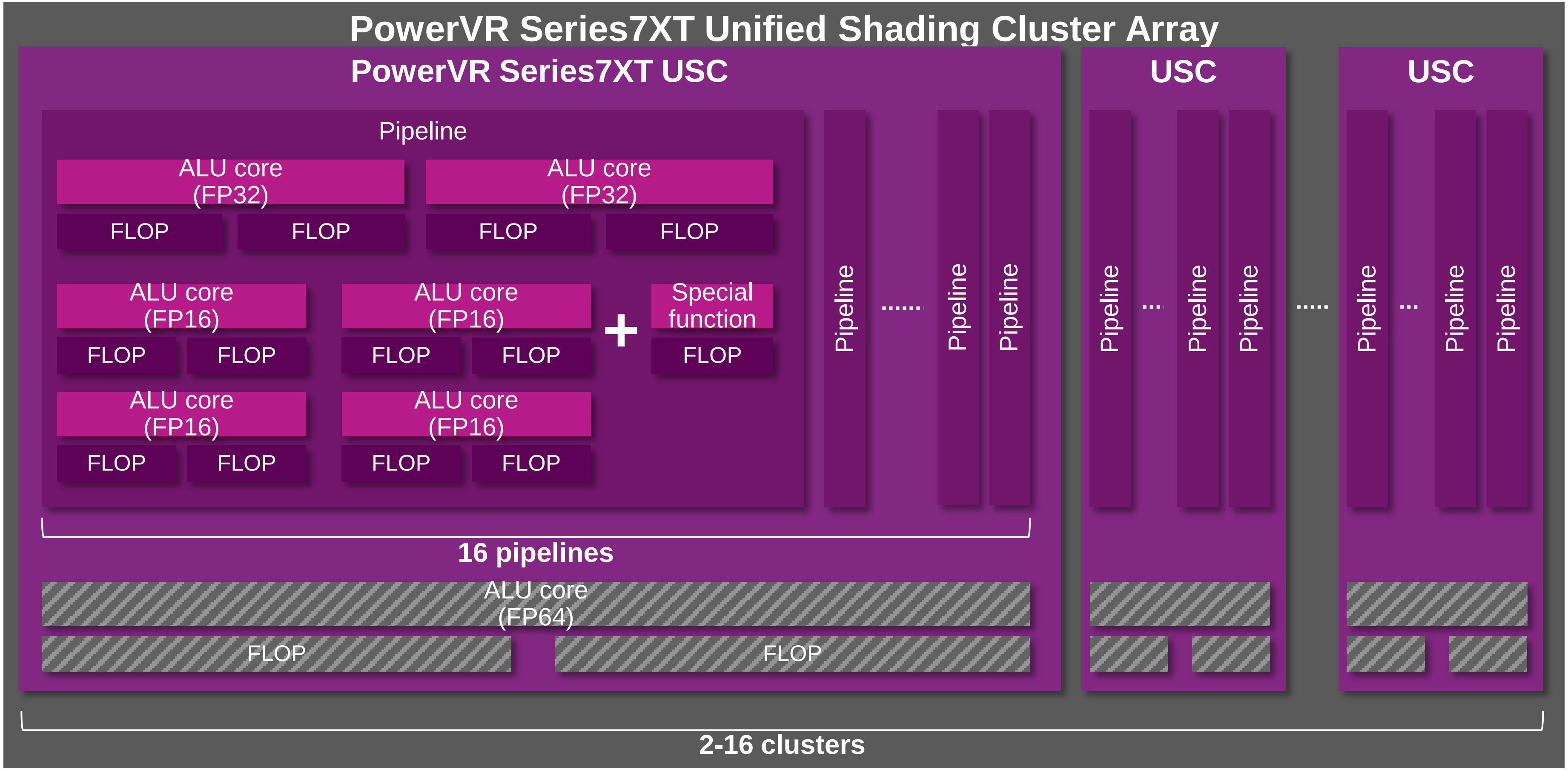
Core Mgmt Unit

Multi-level Memory Cache Unit (MCU)

2D Core (TLA)

System Memory Bus

▨ ***Extra low power GFLOPS***



PowerVR Series7XT Unified Shading Cluster Array

PowerVR Series7XT USC

Pipeline

ALU core (FP32)

FLOP FLOP

ALU core (FP32)

FLOP FLOP

ALU core (FP16)

FLOP FLOP

ALU core (FP16)

FLOP FLOP

+

Special function

FLOP

ALU core (FP16)

FLOP FLOP

ALU core (FP16)

FLOP FLOP

Pipeline .... Pipeline Pipeline

16 pipelines

ALU core (FP64)

FLOP FLOP

USC

Pipeline ... Pipeline Pipeline

.....

USC

Pipeline ... Pipeline Pipeline

2-16 clusters

▨ *Optional*

Series6XT to Series7XT

Adding features and smoothing off rough edges

- **Changed how the architecture scales**
- **Improved USC**
- **Streamlined ISA**
- **Features**
  - Hardware tessellation
  - DX11-compliant USC (precision mainly)
  - FP64

Into the future

- **Exciting changes being worked on across the architecture**
  - USC
  - Front-end
  - Scaling
  - Stuff you want!

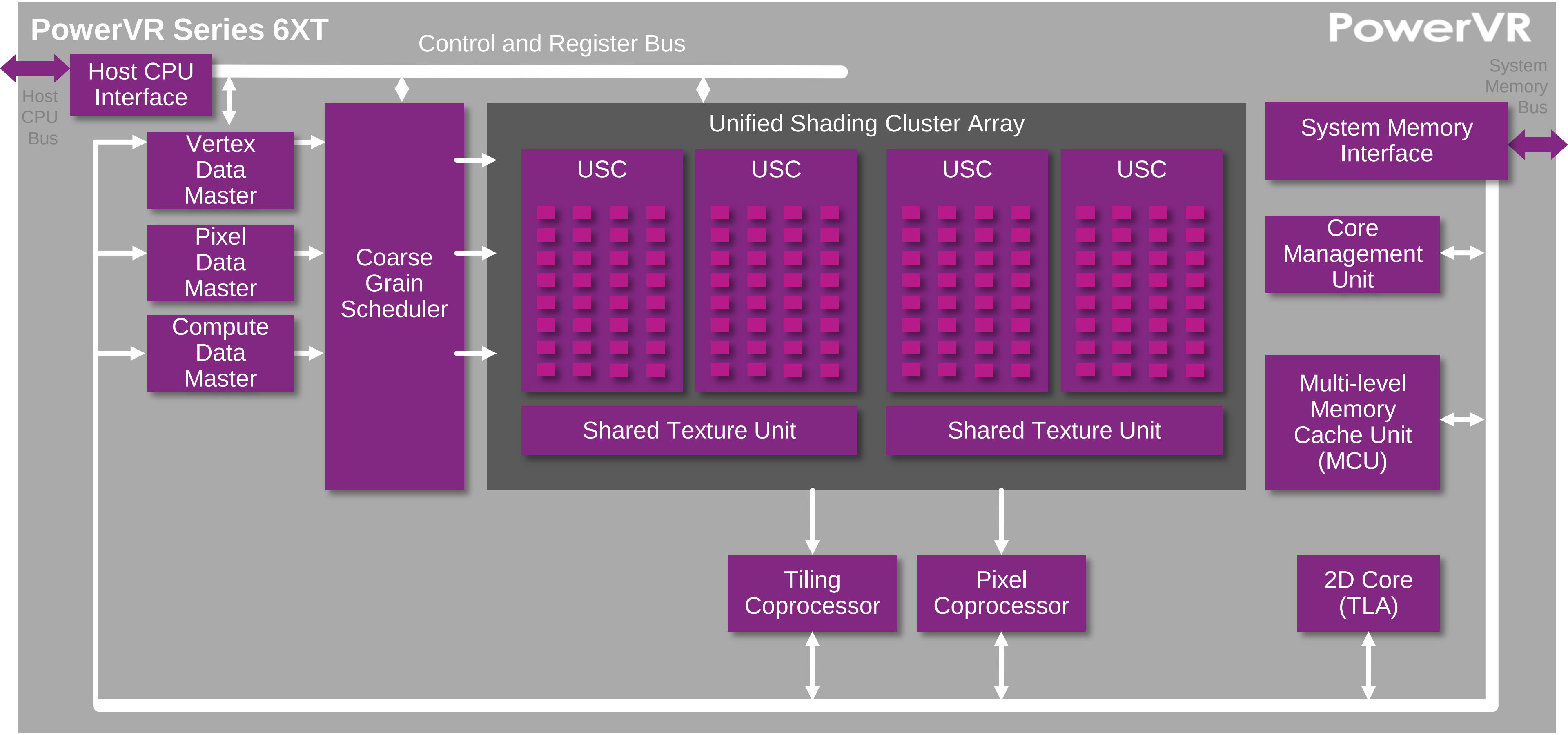
- **You can help**
  - We love feedback about the architecture and how it could best fit what you’re doing
  - Don’t be shy

PowerVR Wizard
Ray Tracing Update

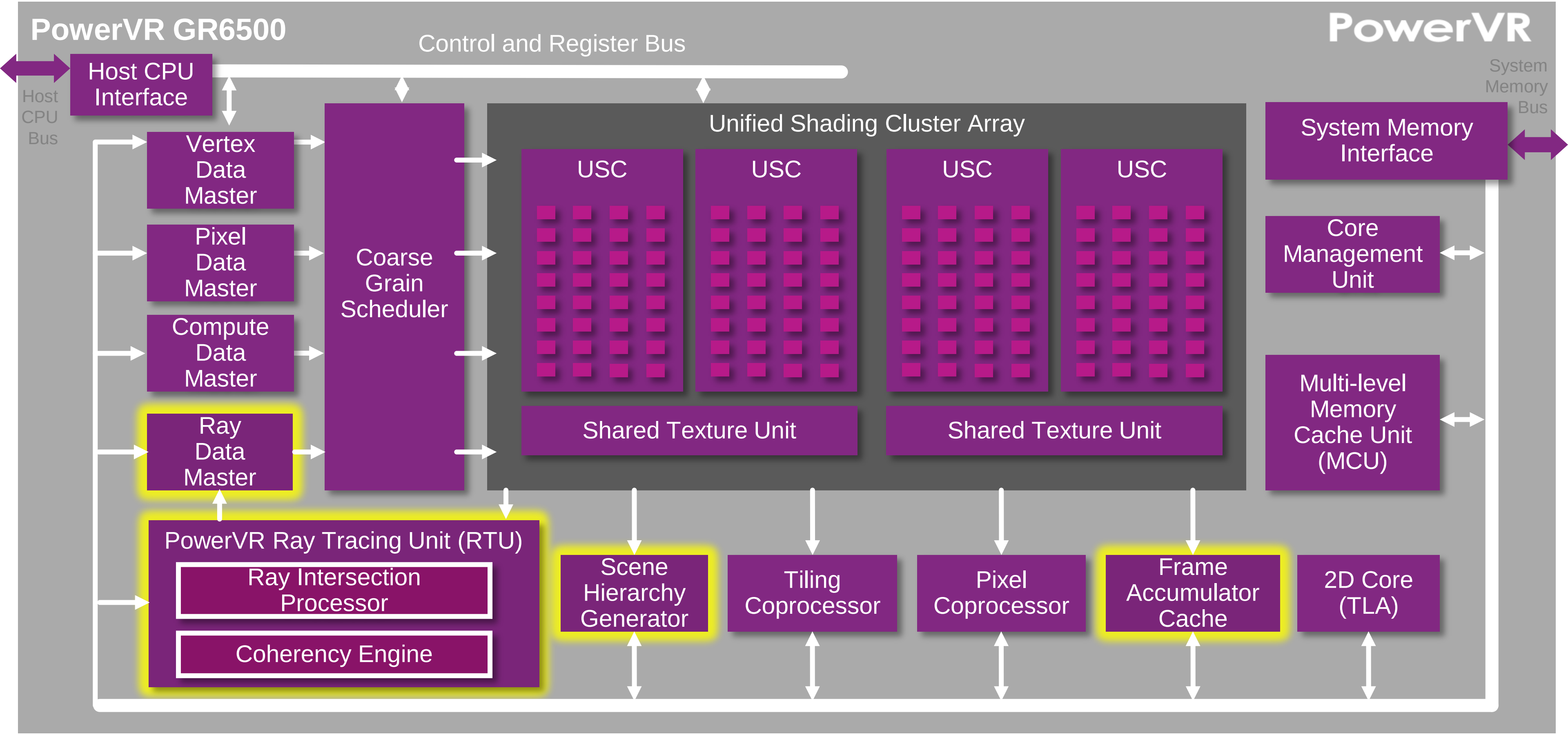
What is Ray Tracing?

Ray tracing is the ability for the shader program for one object to be aware of the geometry of other objects.

PowerVR Architecture



PowerVR Graphics Wizard Architecture

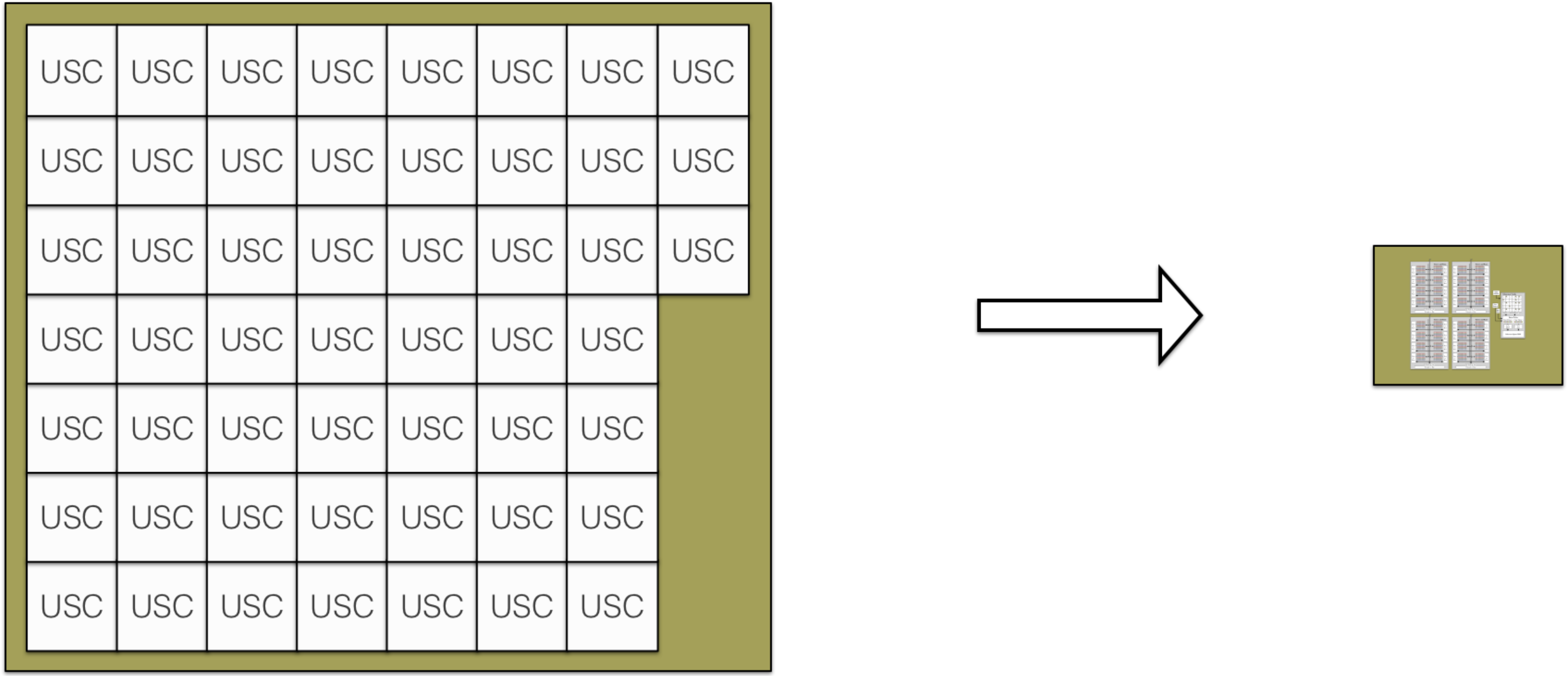


3 Unique Features of Wizard

- Fixed-function Ray-Box and Ray-Triangle testers
- Coherence-Driven Task-Forming and Scheduling
- Streaming Scene Hierarchy Generator

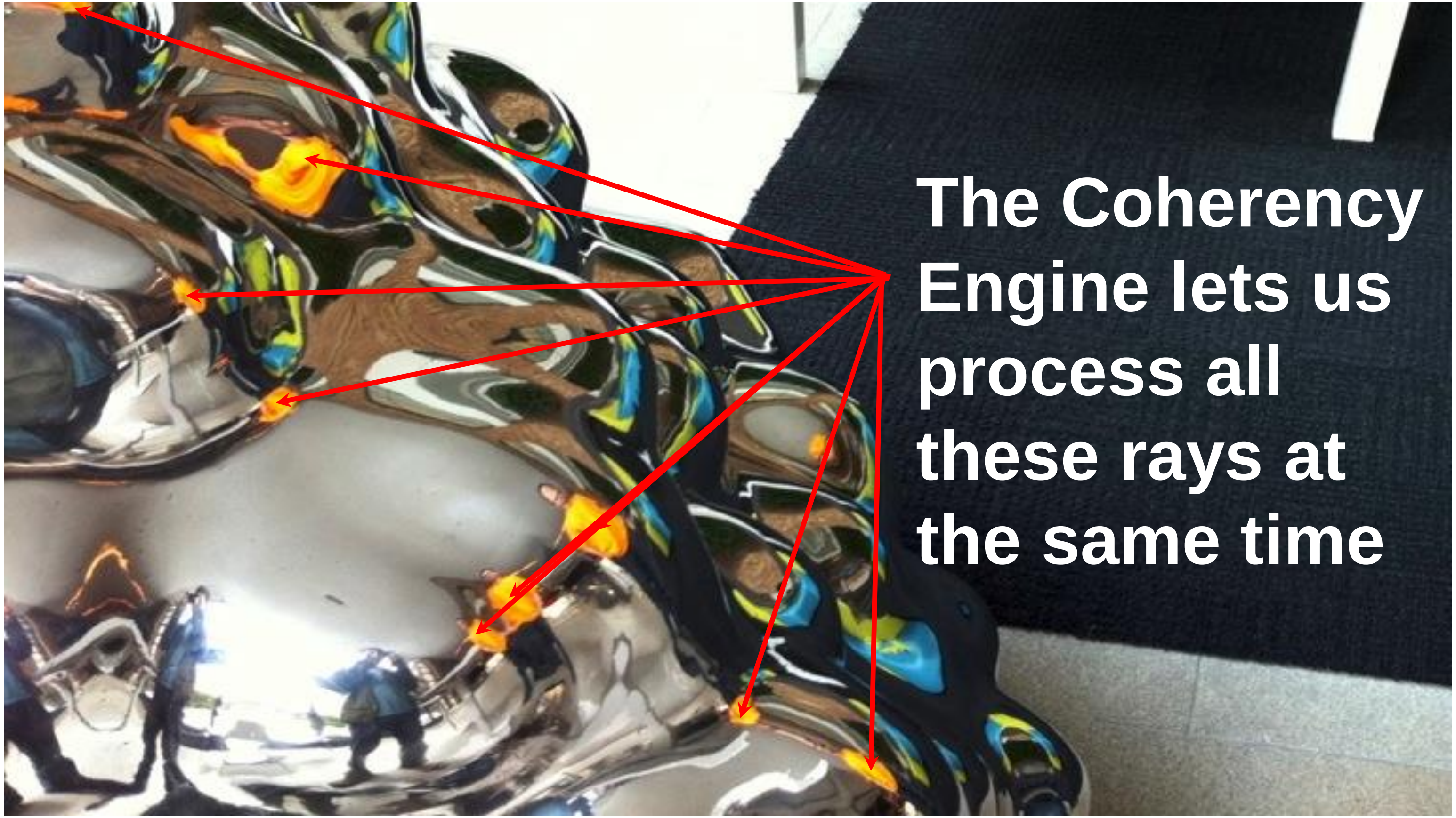
Fixed-Function Ray-Box and Ray-Triangle Testers

44x Less Area for Box Testing



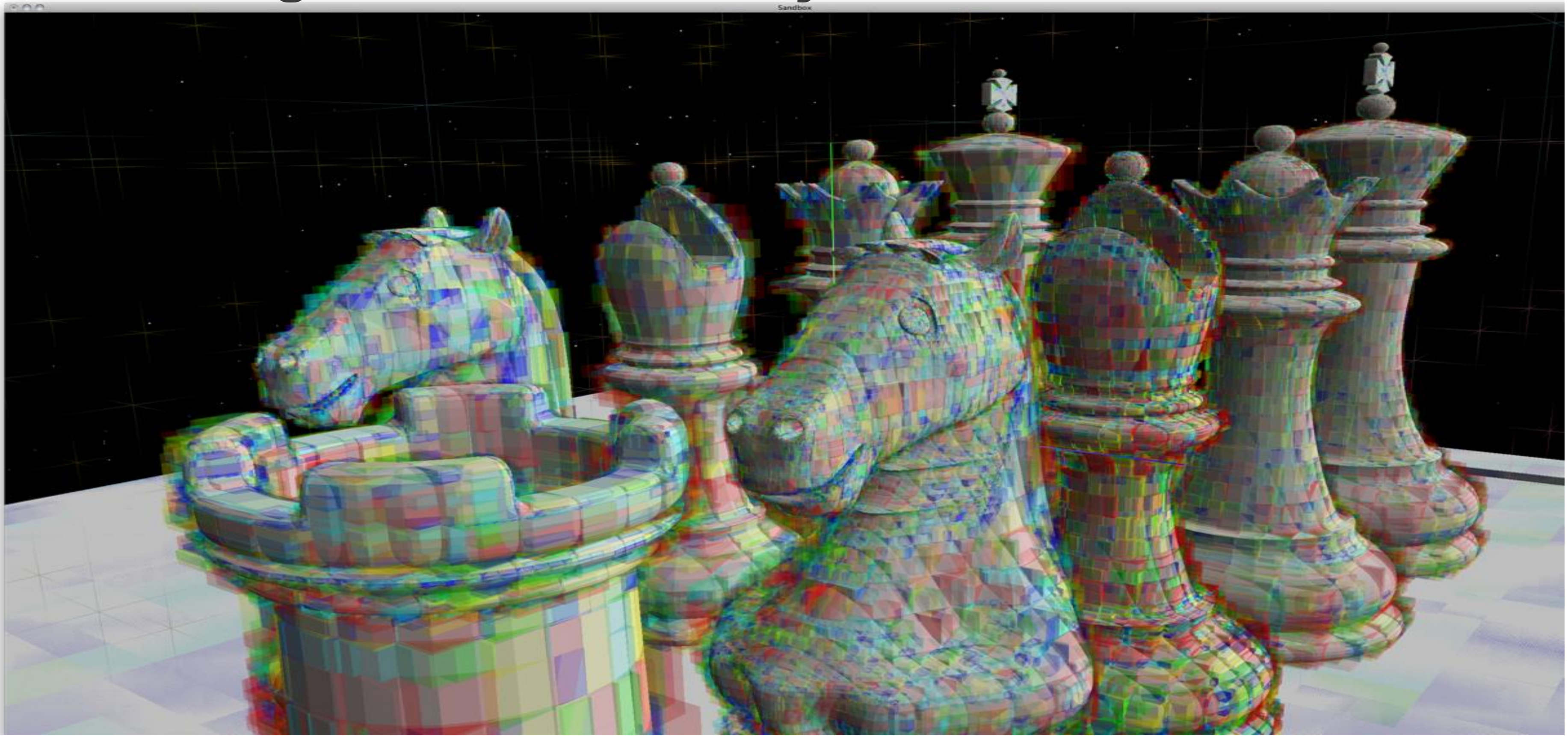
USC USC USC USC USC USC USC USC
USC USC USC USC USC USC USC USC
USC USC USC USC USC USC USC USC
USC USC USC USC USC USC USC
USC USC USC USC USC USC USC
USC USC USC USC USC USC USC
USC USC USC USC USC USC USC

Coherence-Gathering



The Coherency Engine lets us process all these rays at the same time

Streaming Scene Hierarchy Generator



What is Ray Tracing?

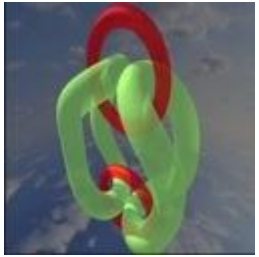
Ray tracing is the ability for the shader program for one object to be aware of the geometry of other objects.

Just a few use cases

Hybrid Shadows, Reflections, etc.

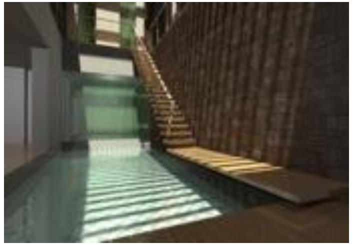
Augmented Reality

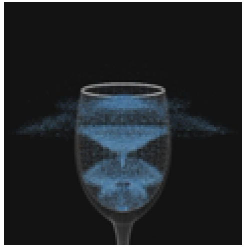
Production-Quality Renders

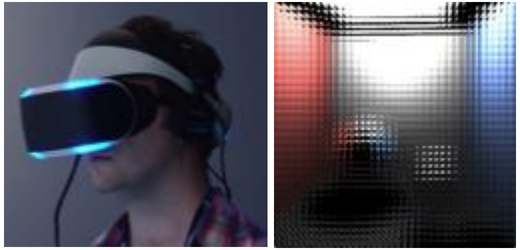

Order-Independent Transparency

Ambient Occlusion

Asset creation / compression


Global Illumination


Physics & Collision Detection

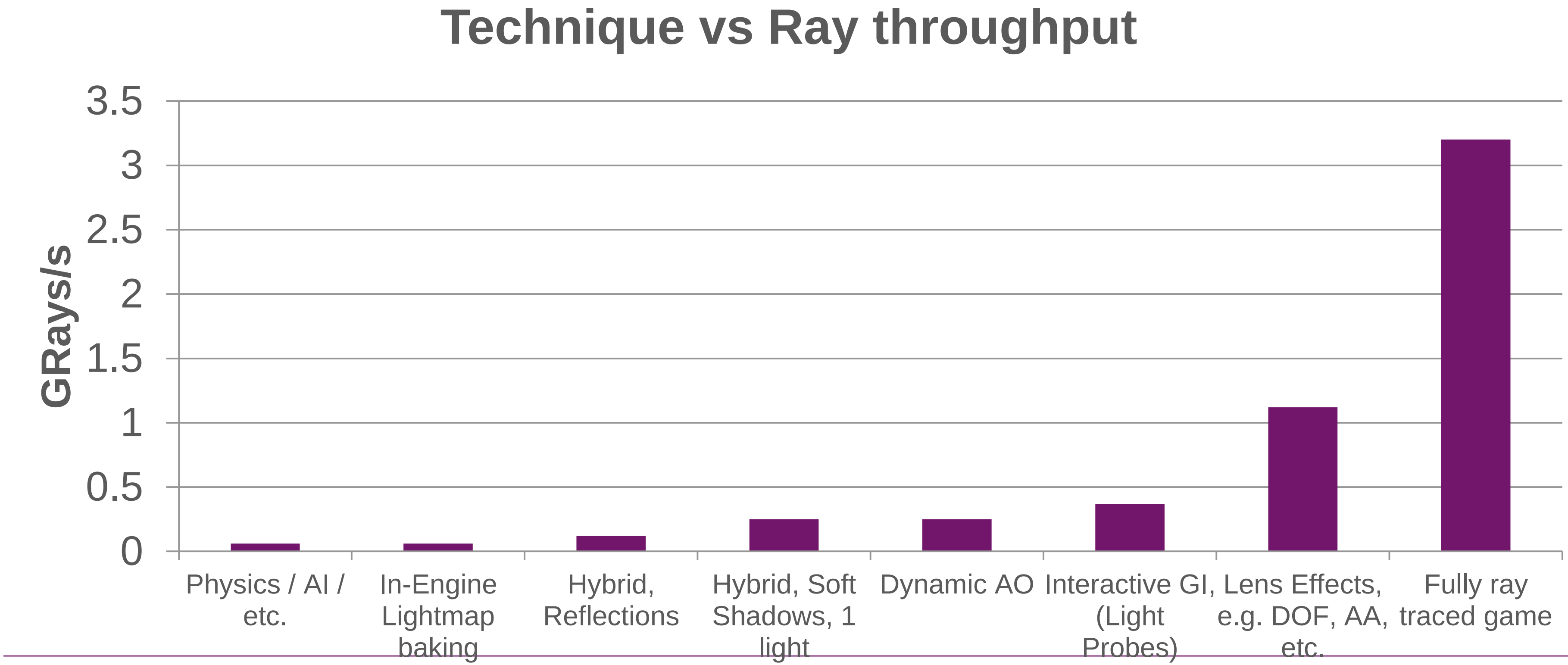

Virtual Reality
Lens correction, Ultra-low latency rendering, Lenticular Displays

A.I. & Line of Sight Calculations

Rapid photo-quality output

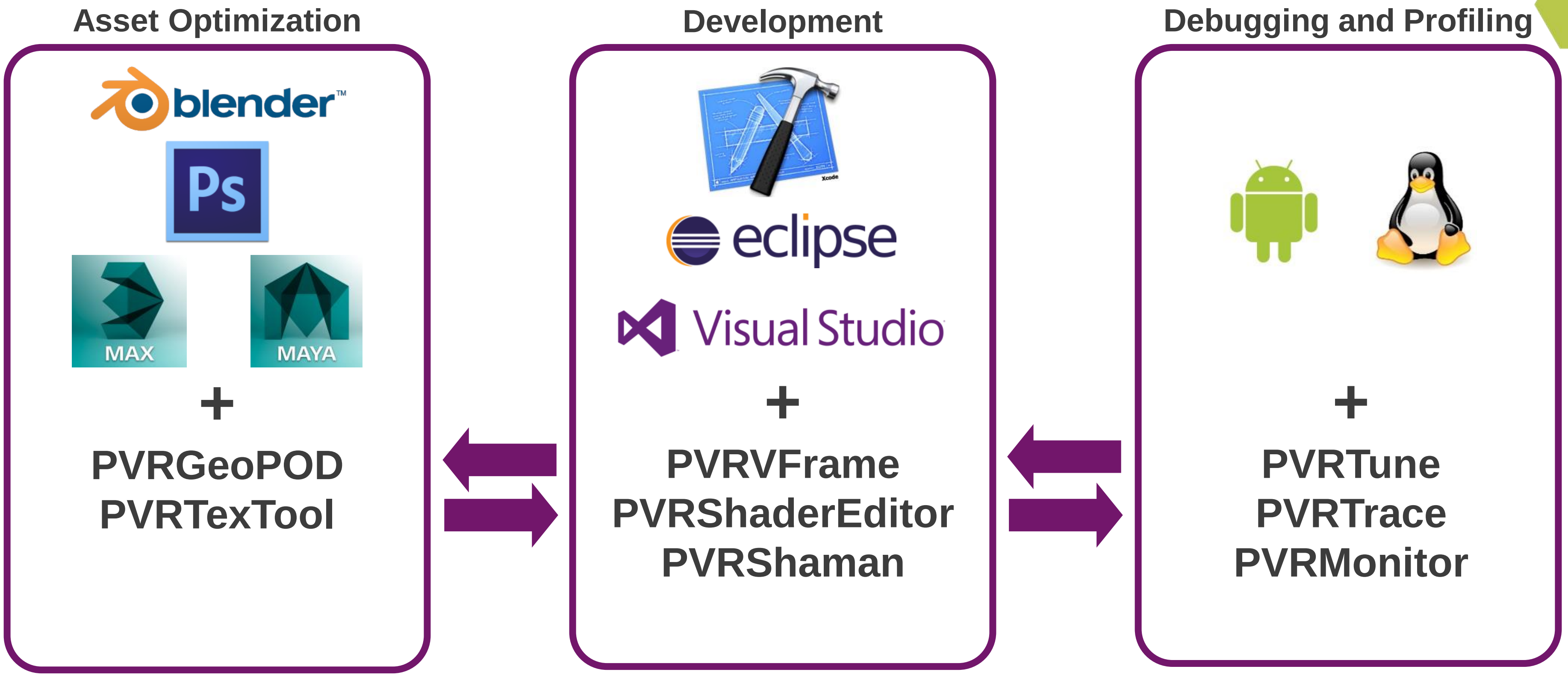
Ray Tracing Requirements

Sustained Ray Throughput at 1080p, 60fps



PowerVR developer tools

PowerVR Tools



Asset Optimization

+

PVRGeoPOD
PVRTexTool

Development

+

PVRVFrame
PVRShaderEditor
PVRShaman

Debugging and Profiling

+

PVRTune
PVRTrace
PVRMonitor

PowerVR Tools

Release schedule

- **PowerVR Tools release process**
  - Minor revision roughly every 6 months

- **Recent/upcoming releases**
  - 3.5 SDK (April 2015)
  - 4.0 SDK (due September 2015)

PVRTrace

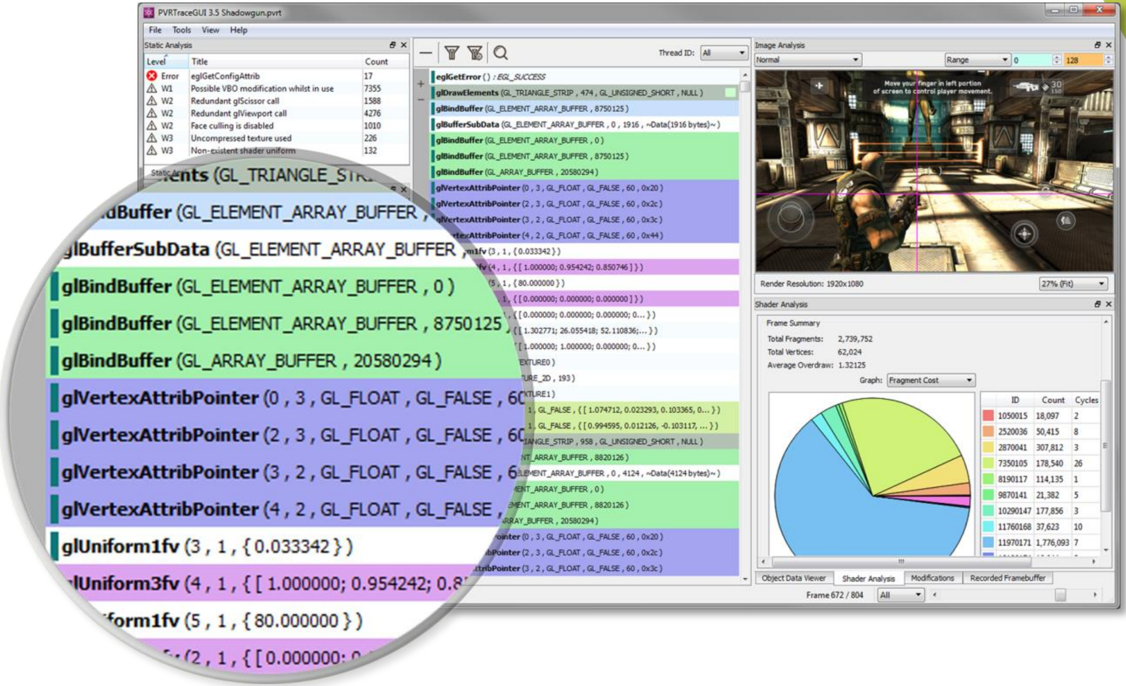
What is PVRTrace?

OpenGL ES API tracer

- OpenGL ES 1.x, 2.0 and 3.x recording libraries
- GUI for analysis

Features

- Inspect, analyse and playback captured data



Level	Title	Count
Error	eglGetConfigAttrib	17
W1	Possible VBO modification whilst in use	7355
W2	Redundant glScissor call	1588
W2	Redundant glViewport call	4276
W2	Face culling is disabled	1010
W3	Uncompressed texture used	226
W3	Non-existent shader uniform	132

```
eglGetError ( ) : EGL_SUCCESS
glDrawElements (GL_TRIANGLE_STRIP , 474 , GL_UNSIGNED_SHORT , NULL )
glBindBuffer (GL_ELEMENT_ARRAY_BUFFER , 8750125 )
glBufferSubData (GL_ELEMENT_ARRAY_BUFFER , 0 , 1916 , ~Data(1916 bytes)~ )
glBindBuffer (GL_ELEMENT_ARRAY_BUFFER , 0 )
glBindBuffer (GL_ELEMENT_ARRAY_BUFFER , 8750125 )
glBindBuffer (GL_ARRAY_BUFFER , 20580294 )
glVertexAttribPointer (0 , 3 , GL_FLOAT , GL_FALSE , 60 , 0x20 )
glVertexAttribPointer (2 , 3 , GL_FLOAT , GL_FALSE , 60 , 0x2c )
glVertexAttribPointer (3 , 2 , GL_FLOAT , GL_FALSE , 60 , 0x3c )
rtexAttribPointer (4 , 2 , GL_FLOAT , GL_FALSE , 60 , 0x44 )
m1fv (3 , 1 , { 0.033342 } )
fv (4 , 1 , { [ 1.000000; 0.954242; 0.850746 ] } )
(5 , 1 , { 80.000000 } )
1 , { [ 0.000000; 0.000000; 0.000000 ] } )
, { [ 0.000000; 0.000000; 0.000000; 0... } )
{ [ 1.302771; 26.055418; 52.110836;... } )
[ 1.000000; 1.000000; 0.000000; 0... } )
EXTURE0 )
URE_2D , 193 )
XTURE1 )
1 , GL_FALSE , { [ 1.074712, 0.023293, 0.103365, 0... } )
1 , GL_FALSE , { [ 0.994595, 0.012126, -0.103117, ... } )
IANGLE_STRIP , 958 , GL_UNSIGNED_SHORT , NULL )
NT_ARRAY_BUFFER , 8820126 )
ELEMENT_ARRAY_BUFFER , 0 , 4124 , ~Data(4124 bytes)~ )
MENT_ARRAY_BUFFER , 0 )
EMENT_ARRAY_BUFFER , 8820126 )
ARRAY_BUFFER , 20580294 )
ointer (0 , 3 , GL_FLOAT , GL_FALSE , 60 , 0x20 )
bPointer (2 , 3 , GL_FLOAT , GL_FALSE , 60 , 0x2c )
ttribPointer (3 , 2 , GL_FLOAT , GL_FALSE , 60 , 0x3c )
```

```
ents (GL_TRIANGLE_STR
dBuffer (GL_ELEMENT_ARRAY_BUFFER ,
glBufferSubData (GL_ELEMENT_ARRAY_BUFFER ,
glBindBuffer (GL_ELEMENT_ARRAY_BUFFER , 0 )
glBindBuffer (GL_ELEMENT_ARRAY_BUFFER , 8750125 )
glBindBuffer (GL_ARRAY_BUFFER , 20580294 )
glVertexAttribPointer (0 , 3 , GL_FLOAT , GL_FALSE , 60
glVertexAttribPointer (2 , 3 , GL_FLOAT , GL_FALSE , 60
glVertexAttribPointer (3 , 2 , GL_FLOAT , GL_FALSE , 6
glVertexAttribPointer (4 , 2 , GL_FLOAT , GL_FALSE ,
glUniform1fv (3 , 1 , { 0.033342 } )
lUniform3fv (4 , 1 , { [ 1.000000; 0.954242; 0.8
form1fv (5 , 1 , { 80.000000 } )
(2 , 1 , { [ 0.000000; 0.
```

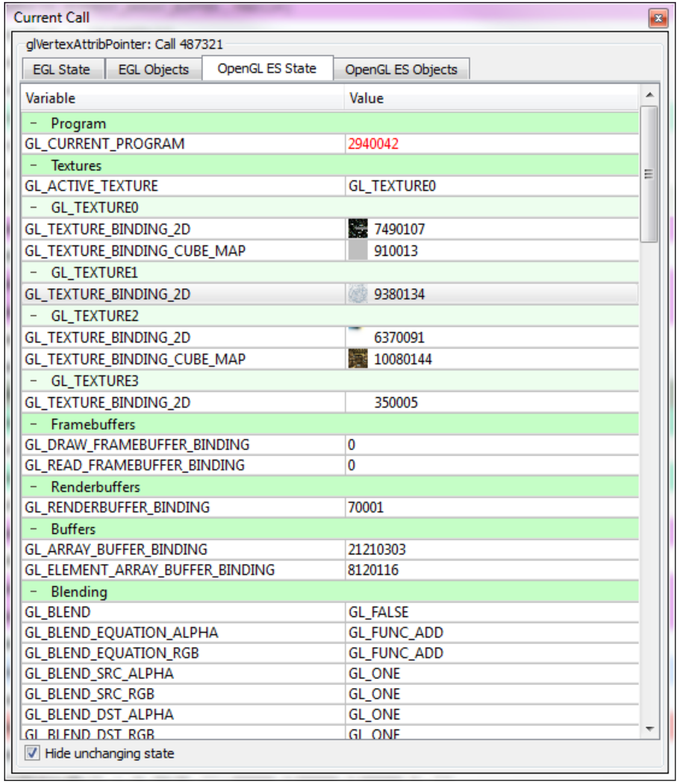
Frame Summary

Total Fragments: 2,739,752
Total Vertices: 62,024
Average Overdraw: 1.32125

	ID	Count	Cycles
🟥	1050015	18,097	2
🟧	2520036	50,415	8
🟨	2870041	307,812	3
🟩	7350105	178,540	26
🟩	8190117	114,135	1
🟩	9870141	21,382	5
🟩	10290147	177,856	3
🟦	11760168	37,623	10
🟦	11970171	1,776,093	7

PVRTrace

New render state & data inspectors

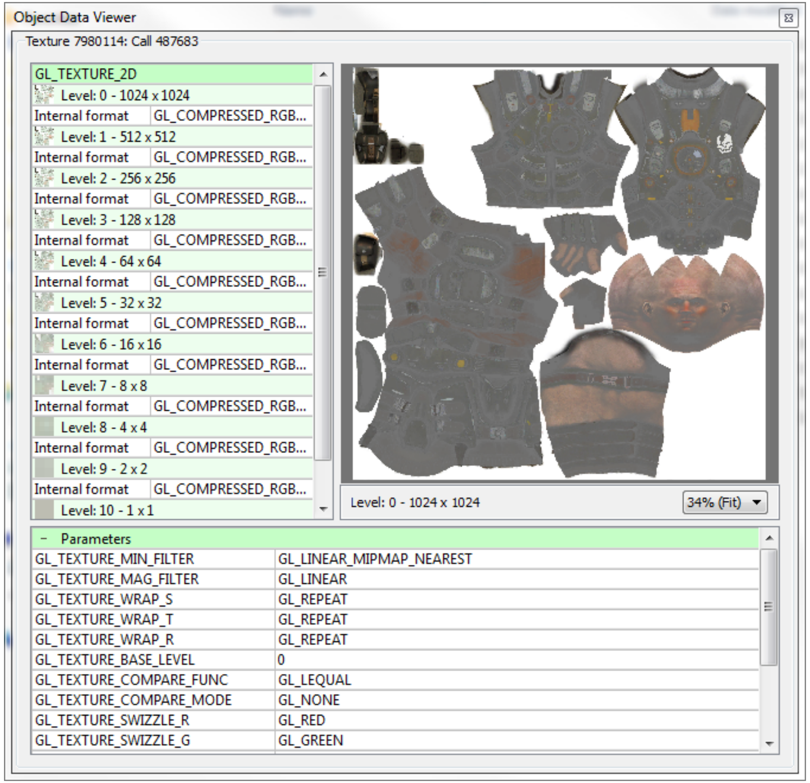


Current Call

glVertexAttribPointer: Call 487321

Variable	Value
− Program	
GL_CURRENT_PROGRAM	2940042
− Textures	
GL_ACTIVE_TEXTURE	GL_TEXTURE0
− GL_TEXTURE0	
GL_TEXTURE_BINDING_2D	7490107
GL_TEXTURE_BINDING_CUBE_MAP	910013
− GL_TEXTURE1	
GL_TEXTURE_BINDING_2D	9380134
− GL_TEXTURE2	
GL_TEXTURE_BINDING_2D	6370091
GL_TEXTURE_BINDING_CUBE_MAP	10080144
− GL_TEXTURE3	
GL_TEXTURE_BINDING_2D	350005
− Framebuffers	
GL_DRAW_FRAMEBUFFER_BINDING	0
GL_READ_FRAMEBUFFER_BINDING	0
− Renderbuffers	
GL_RENDERBUFFER_BINDING	70001
− Buffers	
GL_ARRAY_BUFFER_BINDING	21210303
GL_ELEMENT_ARRAY_BUFFER_BINDING	8120116
− Blending	
GL_BLEND	GL_FALSE
GL_BLEND_EQUATION_ALPHA	GL_FUNC_ADD
GL_BLEND_EQUATION_RGB	GL_FUNC_ADD
GL_BLEND_SRC_ALPHA	GL_ONE
GL_BLEND_SRC_RGB	GL_ONE
GL_BLEND_DST_ALPHA	GL_ONE
GL_BLEND_DST_RGB	GL_ONE

☑ Hide unchanging state



Object Data Viewer

Texture 7980114: Call 487683

GL_TEXTURE_2D	
Level: 0 - 1024 x 1024	
Internal format	GL_COMPRESSED_RGB...
Level: 1 - 512 x 512	
Internal format	GL_COMPRESSED_RGB...
Level: 2 - 256 x 256	
Internal format	GL_COMPRESSED_RGB...
Level: 3 - 128 x 128	
Internal format	GL_COMPRESSED_RGB...
Level: 4 - 64 x 64	
Internal format	GL_COMPRESSED_RGB...
Level: 5 - 32 x 32	
Internal format	GL_COMPRESSED_RGB...
Level: 6 - 16 x 16	
Internal format	GL_COMPRESSED_RGB...
Level: 7 - 8 x 8	
Internal format	GL_COMPRESSED_RGB...
Level: 8 - 4 x 4	
Internal format	GL_COMPRESSED_RGB...
Level: 9 - 2 x 2	
Internal format	GL_COMPRESSED_RGB...
Level: 10 - 1 x 1	

Level: 0 - 1024 x 1024

− Parameters	
GL_TEXTURE_MIN_FILTER	GL_LINEAR_MIPMAP_NEAREST
GL_TEXTURE_MAG_FILTER	GL_LINEAR
GL_TEXTURE_WRAP_S	GL_REPEAT
GL_TEXTURE_WRAP_T	GL_REPEAT
GL_TEXTURE_WRAP_R	GL_REPEAT
GL_TEXTURE_BASE_LEVEL	0
GL_TEXTURE_COMPARE_FUNC	GL_LEQUAL
GL_TEXTURE_COMPARE_MODE	GL_NONE
GL_TEXTURE_SWIZZLE_R	GL_RED
GL_TEXTURE_SWIZZLE_G	GL_GREEN

PVRTune

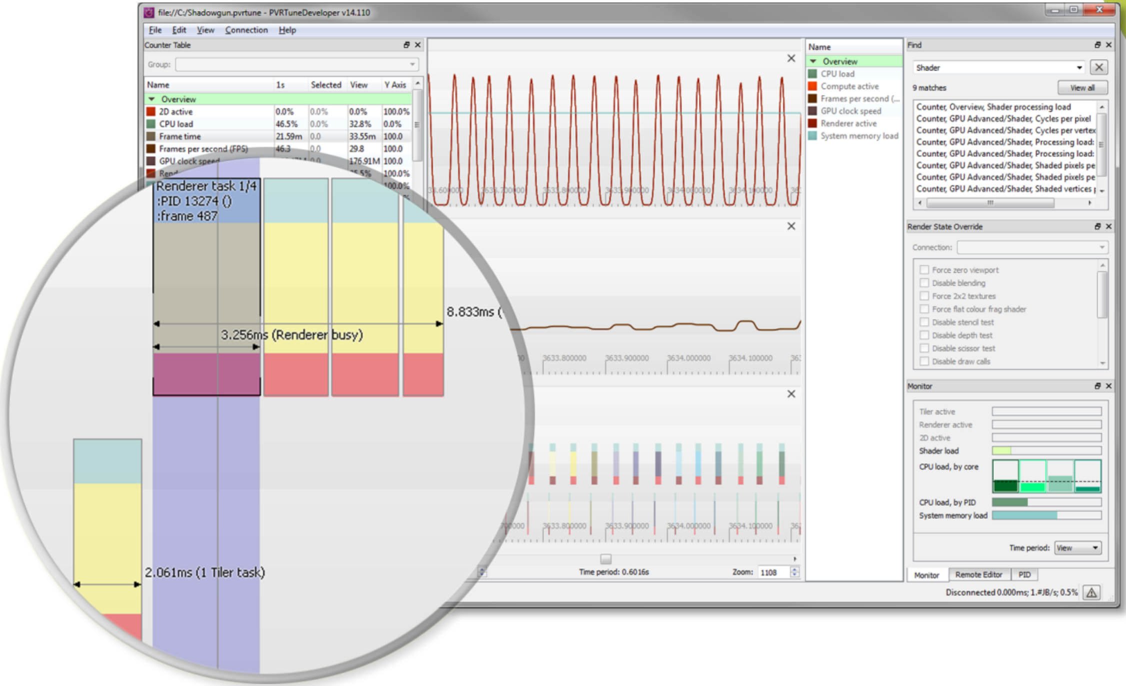
What is PVRTune?

PowerVR graphics core performance analyser

- GUI for analysis
- On-device server

Features

- Real-time performance data



file://C:/Shadowgun.pvrtune - PVRTuneDeveloper v14.110

Counter Table

Group:

Name	1s	Selected	View	Y Axis
Overview				
2D active	0.0%	0.0%	0.0%	100.0%
CPU load	46.5%	0.0%	32.8%	0.0%
Frame time	21.59m	0.0	33.55m	100.0
Frames per second (FPS)	46.3	0.0	29.8	100.0
GPU clock speed	[illegible]	0.0	176.91M	100.0
Rend				100.0%

Renderer task 1/4
:PID 13274 ()
:frame 487

8.833ms (

3.256ms (Renderer busy)

2.061ms (1 Tiler task)

Name
- Overview
- CPU load
- Compute active
- Frames per second (...
- GPU clock speed
- Renderer active
- System memory load

Find

Shader

9 matches

Counter, Overview, Shader processing load
Counter, GPU Advanced/Shader, Cycles per pixel
Counter, GPU Advanced/Shader, Cycles per vertex
Counter, GPU Advanced/Shader, Processing load:
Counter, GPU Advanced/Shader, Processing load:
Counter, GPU Advanced/Shader, Shaded pixels pe
Counter, GPU Advanced/Shader, Shaded pixels pe
Counter, GPU Advanced/Shader, Shaded vertices

Render State Override

Connection:

□ Force zero viewport
□ Disable blending
□ Force 2x2 textures
□ Force flat colour frag shader
□ Disable stencil test
□ Disable depth test
□ Disable scissor test
□ Disable draw calls

Monitor

Tiler active
Renderer active
2D active
Shader load
CPU load, by core
CPU load, by PID
System memory load

Time period: View

Time period: 0.6016s

Zoom: 1108

Disconnected 0.000ms; 1.#JB/s; 0.5%

PVRTune

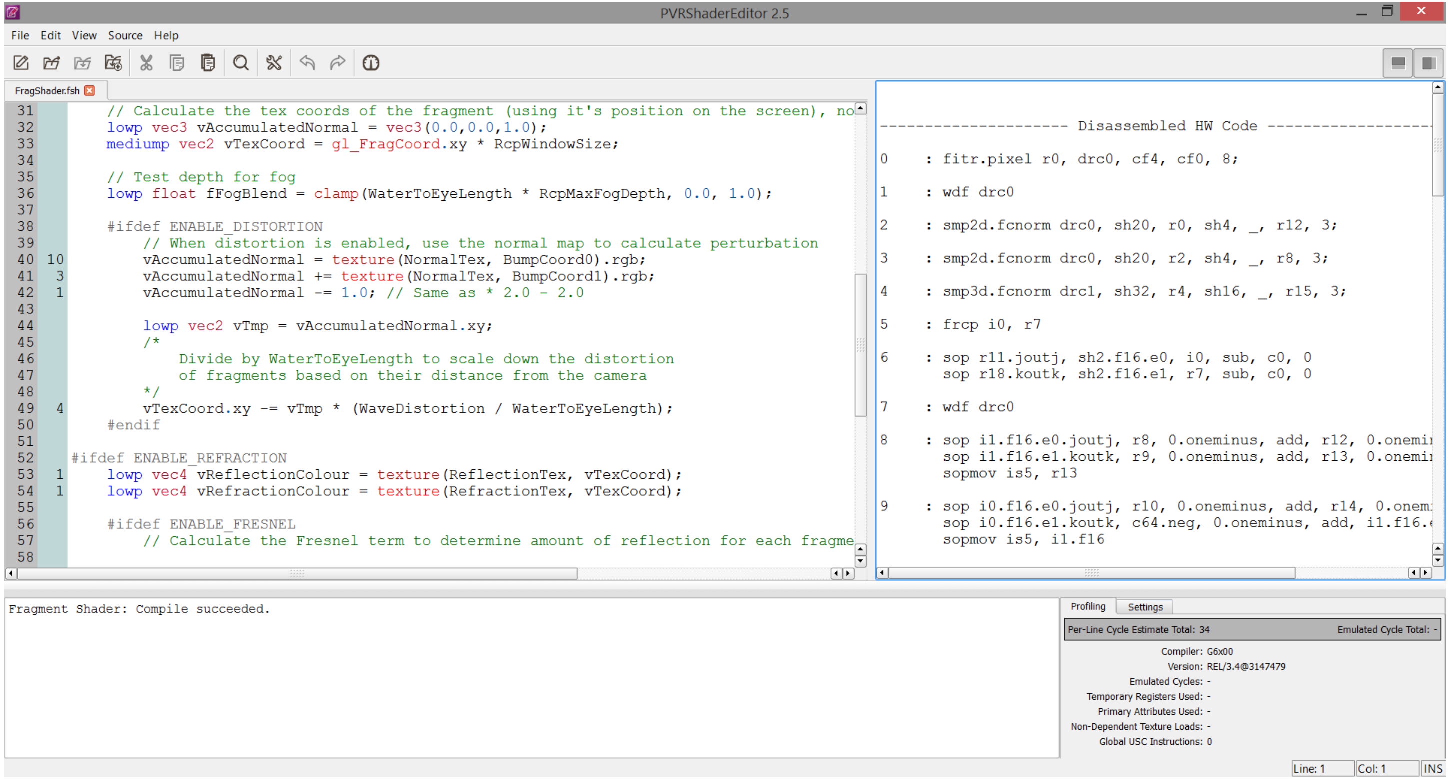
Real-time GPU profiler

- **New counters**
  - GPU clock speed, triangles culled, Hidden Surface Removal efficiency, SLC memory reads/writes and more

- **GUI changes**
  - Simplified setup and navigation
  - Graphics and Compute modes
  - Tree view for counters (Overview, Tiler, Renderer etc.)

PVRShaderEditor

Shader editor & offline profiler (with disassembly!)



```
            // Calculate the tex coords of the fragment (using it's position on the screen), no
            lowp vec3 vAccumulatedNormal = vec3(0.0,0.0,1.0);
            mediump vec2 vTexCoord = gl_FragCoord.xy * RcpWindowSize;

            // Test depth for fog
            lowp float fFogBlend = clamp(WaterToEyeLength * RcpMaxFogDepth, 0.0, 1.0);

            #ifdef ENABLE_DISTORTION
                // When distortion is enabled, use the normal map to calculate perturbation
10              vAccumulatedNormal = texture(NormalTex, BumpCoord0).rgb;
 3              vAccumulatedNormal += texture(NormalTex, BumpCoord1).rgb;
 1              vAccumulatedNormal -= 1.0; // Same as * 2.0 - 2.0

                lowp vec2 vTmp = vAccumulatedNormal.xy;
                /*
                    Divide by WaterToEyeLength to scale down the distortion
                    of fragments based on their distance from the camera
                */
 4              vTexCoord.xy -= vTmp * (WaveDistortion / WaterToEyeLength);
            #endif

        #ifdef ENABLE_REFRACTION
 1          lowp vec4 vReflectionColour = texture(ReflectionTex, vTexCoord);
 1          lowp vec4 vRefractionColour = texture(RefractionTex, vTexCoord);

            #ifdef ENABLE_FRESNEL
                // Calculate the Fresnel term to determine amount of reflection for each fragme
```

```
--------------------- Disassembled HW Code -------------------

0    : fitr.pixel r0, drc0, cf4, cf0, 8;

1    : wdf drc0

2    : smp2d.fcnorm drc0, sh20, r0, sh4, _, r12, 3;

3    : smp2d.fcnorm drc0, sh20, r2, sh4, _, r8, 3;

4    : smp3d.fcnorm drc1, sh32, r4, sh16, _, r15, 3;

5    : frcp i0, r7

6    : sop r11.joutj, sh2.f16.e0, i0, sub, c0, 0
       sop r18.koutk, sh2.f16.e1, r7, sub, c0, 0

7    : wdf drc0

8    : sop i1.f16.e0.joutj, r8, 0.oneminus, add, r12, 0.onemi
       sop i1.f16.e1.koutk, r9, 0.oneminus, add, r13, 0.onemi
       sopmov is5, r13

9    : sop i0.f16.e0.joutj, r10, 0.oneminus, add, r14, 0.onem
       sop i0.f16.e1.koutk, c64.neg, 0.oneminus, add, i1.f16.e
       sopmov is5, i1.f16
```

```
Fragment Shader: Compile succeeded.
```

Per-Line Cycle Estimate Total: 34 | Emulated Cycle Total: -

Compiler: G6x00
Version: REL/3.4@3147479
Emulated Cycles: -
Temporary Registers Used: -
Primary Attributes Used: -
Non-Dependent Texture Loads: -
Global USC Instructions: 0

Line: 1 | Col: 1 | INS

Rogue graphics driver

Rogue graphics driver

Release schedule

- **DDK (Driver Development Kit) release process**
  - Reference driver source code released to PowerVR IP licensees
  - Minor revision roughly every 6 months
  - Top-tier customers engage early. Drivers in products shortly after official DDK release

Rogue graphics driver

1.4 DDK

- **Release date**
  - Q4 2014 (release 1)
  - Q1 2015 (release 2)
- **OpenGL ES: Key features (release 1)**
  - OpenGL ES 3.1
    - Compute shaders, shader storage buffer objects, draw indirect and more
- **OpenGL ES: Key features (release 2)**
  - Android Lollipop support

Rogue graphics driver

1.5 DDK

- **Release date**
  - Q2/Q3 2015
- **OpenGL ES: Key features**
  - Android Extension Pack (AEP)
    - ASTC, blend equation advanced, GPU shader model 5 and more
  - sRGB PVRTC
  - Pixel local storage
    - 128/256 bits per-pixel on-chip

Rogue graphics driver

1.6 DDK

- **Release date**
  - Q4 2015

- **OpenGL ES: Key features**
  - Bicubic texture filtering
  - Shader group vote
  - Polygon offset clamp
  - Pixel local storage 2
    - Simultaneously write to pixel local storage and a framebuffer attachment

Vulkan
TM

Vulkan

About

- ## What is Vulkan?
  - New open standard API developed by the Khronos group
  - Designed for high-efficiency access to graphics and compute on modern GPUs

- ## Key features
  - Minimizes driver overhead and enables multi-threaded GPU command preparation
  - Designed for mobile, desktop, console and embedded platforms
  - Designed for all GPUs - tile based GPUs are first-class citizens!
  - SPIR-V – binary intermediate language for shaders

Vulkan

PowerVR driver status

- **PowerVR Vulkan driver**
  - Driver development on-going
  - Working with key partners on initial content bring up
  - More details at SIGGRAPH 2015
    - Khronos BoF: Vulkan, OpenGL, OpenGL ES - 5:30 PM - 7:30 PM

PowerVR Graphics

Future roadmaps

- **What drives our roadmaps?**
  - Market analysis
  - Customer feedback
  - Developer feedback

Upcoming events

idc-UK

- **Imagination Developers Connection 2015 UK**
  - 1st October, SOHO Hotel, London UK
  - Register here: http://imgtec.com/idc/idc15-uk/

- **Agenda**
  - A full developer day including optimization tips, how to use ray tracing with raster graphics and more
  - Also includes guest talks from Google and Digital Legends

Questions?

Imagination

www.imgtec.com/idc