GPU TECHNOLOGY CONFERENCE

ZeroLight

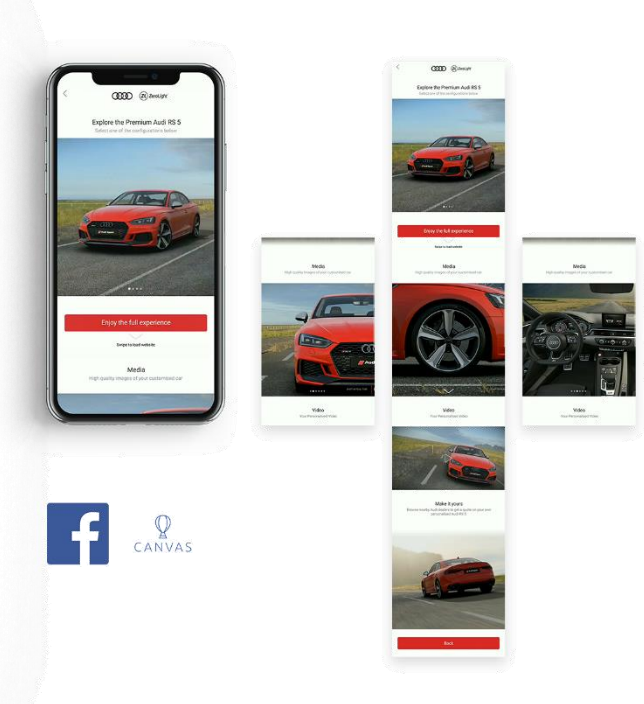
ZeroLight at GTC presents

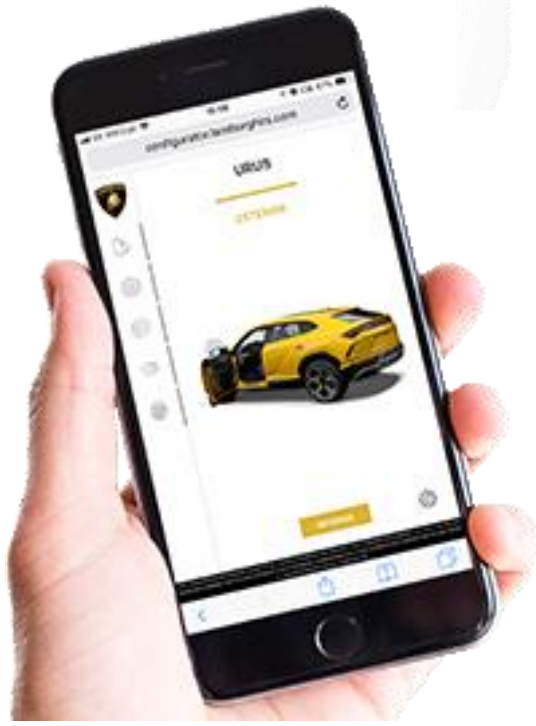
Advances in Real-Time Automotive Visualisation

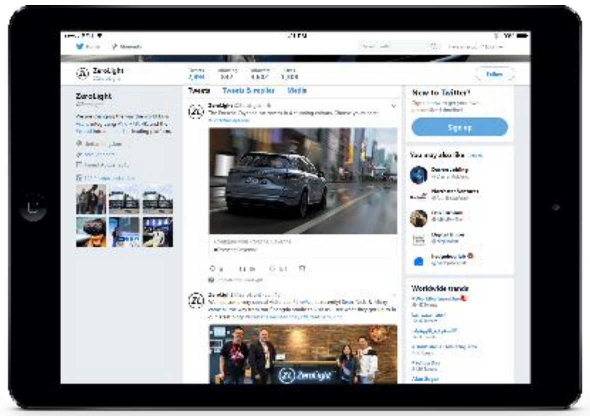
Chris O'Connor

At ZeroLight, we've created the market-leading visualisation and data platform for the **automotive industry**

Our Content Maintains the Same High Quality and **Real-time** Configurability Across All Platforms







Amplifying personalisation

through machine learning

Ray Tracing and Next-gen VR

This Section Will Cover

- Why ray tracing is important
- Details of custom tech first shown at GTC18
- Rasterisation and ray tracing
- DXR and RTX

What’s Missing from Automotive Real-time Rendering?

- Audi asked us what's next to improve graphics quality?
- Started development in 2017
- Content already had passed look and quality control
- We had to find a solution that adds quality to the current look

Ray Tracing

Reference image from Audi.de

Self-reflection Options

- SSR (Screen Space Reflections) – lots of issues
- Localised Planar Reflections – great for some areas
- Compute ray tracing – too slow for real-time
- Voxels reflections – perfect for GPU acceleration

Voxel Ray-traced Reflections

- GPU can be used to generate voxels
- Quick ray cast and look up in Pixel Shader
- Uses a lot of memory
- Sparse voxels helps a lot
- Difficult to update dynamically

Voxel Ray-traced Reflections

Ray Tracing Demo GTC 2018

- We showed our solution at GTC 2018 on the NVIDIA booth
- Showing the Audi A3 with accessories
- Rendering 4K at 60FPS
- 500 million rays per second (0.5 billion)
- Used 2x GV100 (Volta GPU)

Voxel Ray-traced Reflections

Ray Tracing Demo GTC 2018


REAL

DXR + RTX

- DXR and RTX hardware accelerated ray tracing
- Up to 10 billion rays per second
- Resources all managed by DXR
- Native support: rays and pixels can be dispatched in parallel
- Denoiser = fewer rays required per pixel

Rasterisation and Ray Tracing

- Key to maintaining the look across devices that can’t ray trace
- Let each part of the GPU do what it does best
- Ray tracing can be used for shadows, AO, GI and reflections
- Stage 1: graphical improvement without changing the look
- Stage 2: runs at high quality in real-time (4K @ 60FPS)

Rasterisation and Ray Tracing

Ray Tracing

Ray Tracing

Ray Tracing

Ray Tracing

Ray Tracing

Ray Tracing

Next-gen VR

This Section Will Cover

- Current VR headsets
- Next-gen VR headsets
- Optimising for next-gen VR
- Quality and results

Current VR Headsets

Windows Mixed Reality

oculus

VIVE

Current VR Headsets

- 1080x1200 / 1440x1440 resolution per eye
- 90 FPS
- 95˚ - 110˚ FOV
- 2 rendered viewports, one per eye

Features of New Headsets

- Higher resolution up to 2560x1440 per eye
- 90 FPS
- Up to 210˚ FOV
- Multiple rendered viewports per eye (wide FOV, foveated rendering)

VIVE Pro and VIVE Pro Eye

VIVE PRO

StarVR One, Pimax 8K, XTAL

STARVR

PIMAX

vrg. vrgineers

Varjo VR-1

What's the Challenge?

- If you just brute force render viewports and pixels, even with the best GPUs, it will be a challenge to hit performance
- You must take advantage of GPU software-activated hardware features
- NVIDIA VR Works with NVAPI

VR SLI

- Previously, we used AFR (Alternate Frame Rendering)
- Great for GPU Utilisation (80%+)
- AFR adds additional 11ms of latency (bad for VR)
- VR SLI renders one eye per GPU
- Copy of buffer back to GPU0 is the bottleneck (PCI Express)
- Easy to instance per eye, reducing CPU overhead

VR SLI

GPUs: 2 NVIDIA
GPU-0: 90%
View 0 SinglePassRenderView: 9.896927 [17.51731]
View 0 RenderUI: 0.03110934 [0.0616]
RealtimeShadowBlur: 0.6090316 [0.60512]

Drivers: 416.81
GPU-1: 84%
View 0 RenderView: 5.668022 [11.16374]
View 0 RenderFlares: 6.4E-06 [3.2E-05]
Total frame time: 9.896927 [17.51731]

VRAM: 7208/12288

View 0 PostPass: 0.8722379 [1.727328]
RealtimeShadowRender: 3.098377 [4.358176]
Miliseconds Waited For Result: 2

GPUs: 2 NVIDIA
GPU-0: 90%

Drivers: 416.81
GPU-1: 84%

RealtimeShadowRender: 3.098377 [4.358176]
Miliseconds Waited For Result: 2

VR SLI

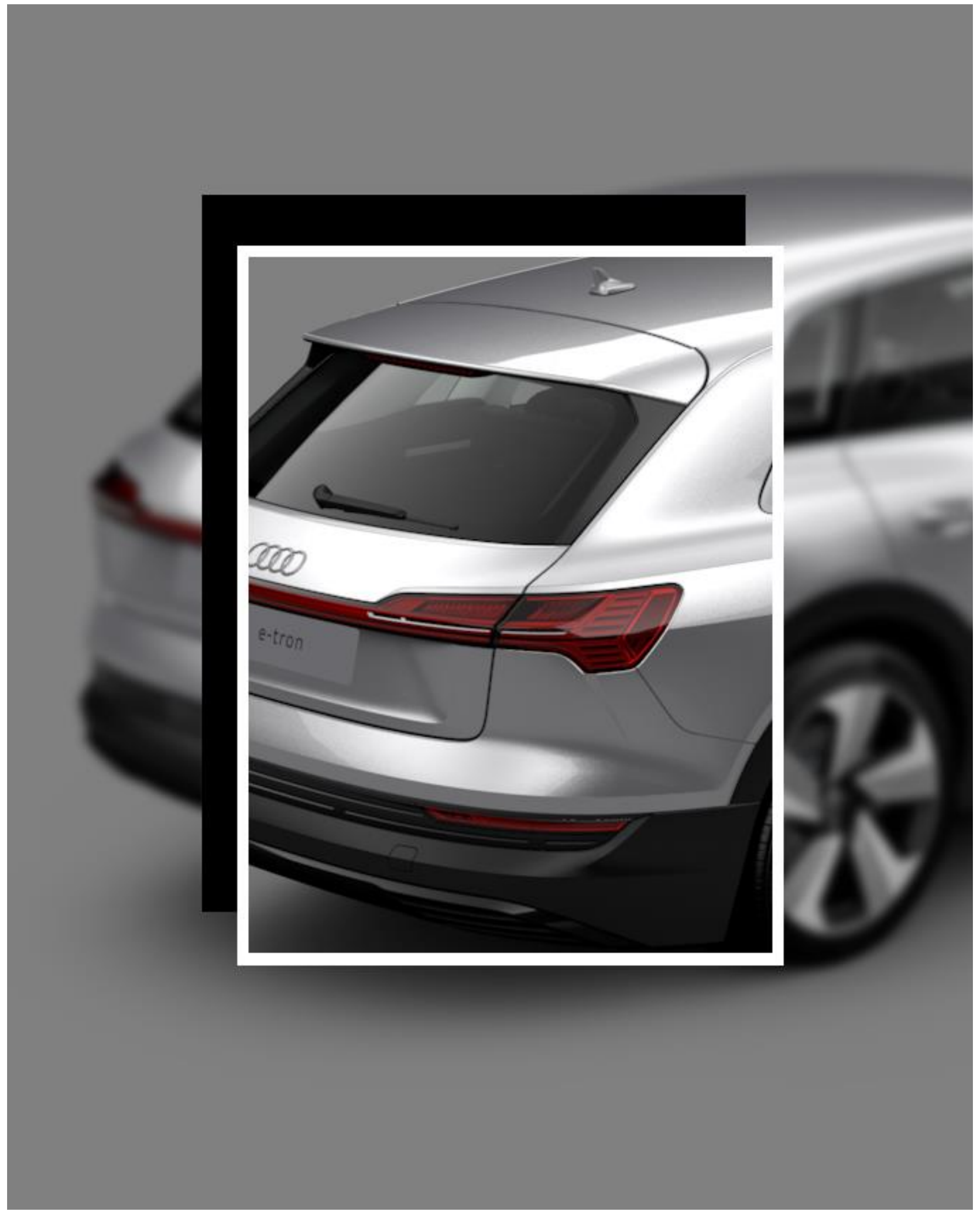
- We have to use tricks to copy less data and prevent GPUs waiting for copies

Copy Data Async

```
m_pMultiGPUDevice->CopySubresourceRegion(deviceContext,
rtcd.m_RenderTargetTexture, 0, dstGPU, rtcd.m_Left, rtcd.m_Top, 0,
rtcd.m_RenderTargetTexture, 0, srcGPU, &srcBox, NVAPI_COPY_ASYNCHRONOUSLY);
```

VR SLI

Viewport cut and copy



Single Pass Stereo

- Reduce CPU overhead and render in a single pass for both eyes
- Requires shader modifications

```
//Standard
float4 standardPos = mul(MATRIX_MVP, vertex);
outPos.position = standardPos;


//Single Pass Stereo
float4 eyePos0 = mul(g_ViewProjMatrix[1], worldPos);
float4 eyePos1 = mul(g_ViewProjMatrix[2], worldPos);

outPos.position = eyePos0;
outPos.posClipRight = eyePos1;
outPos.ViewportMask = NV_SINGLERT_VIEWPORT_MASK;
```

- Mid-frame effects become more complex

New with Turing GPUs

VARIABLE RATE SHADING

MULTI-VIEW RENDERING

NVLINK2

- Reduces copy time from 4ms to 0.1ms (StarVR One)
- We can now share additional data between the GPUs

Multi-view Rendering

- Very similar to SPS but supports 4 views
- Big CPU optimisation on HMDs that require >2 Viewports

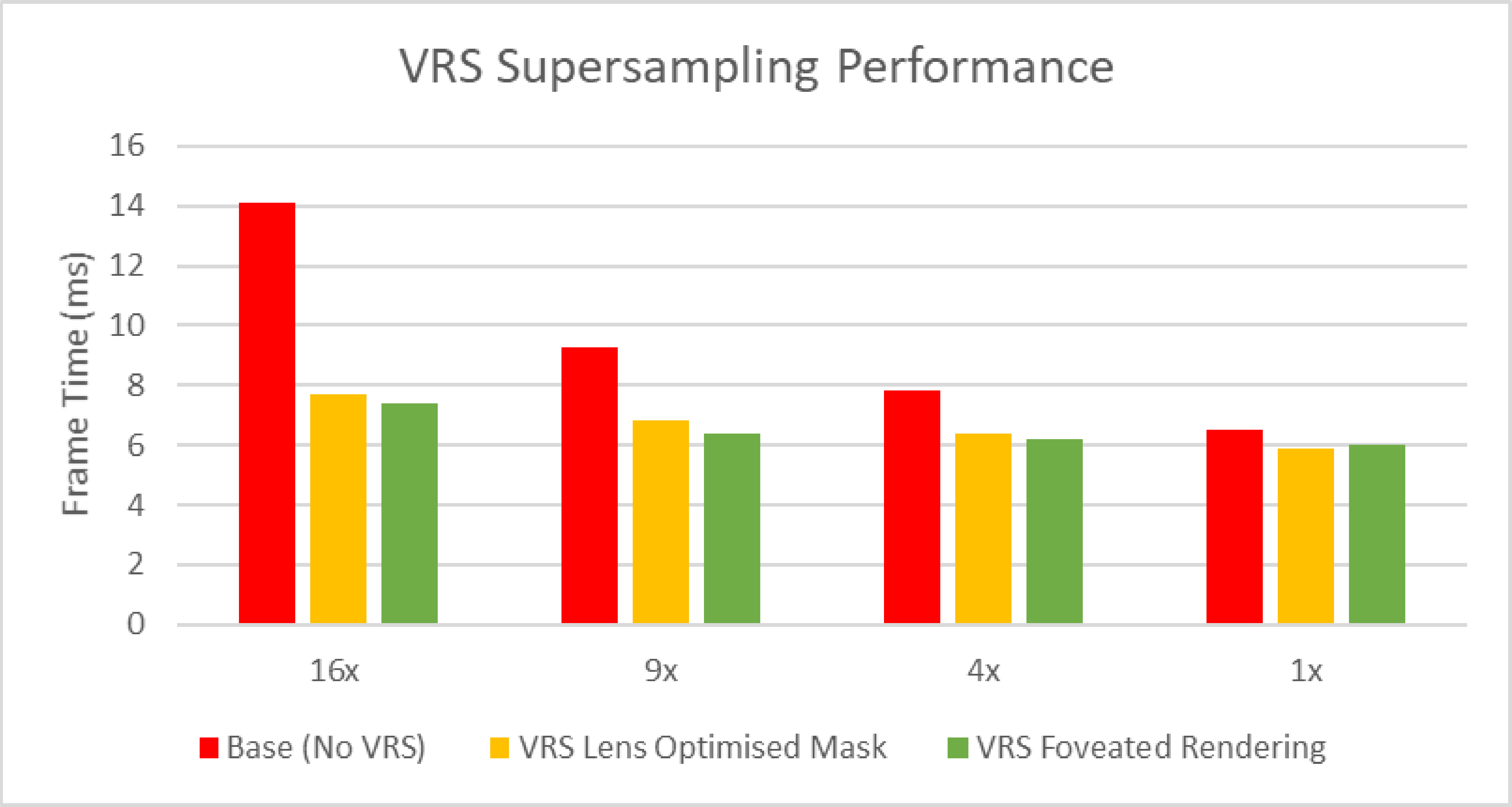
```
//Standard
float4 standardPos = mul(MATRIX_MVP, vertex);
outPos.position = standardPos;


//Single Pass Stereo
float4 eyePos0 = mul(g_ViewProjMatrix[1], worldPos);
float4 eyePos1 = mul(g_ViewProjMatrix[2], worldPos);

outPos.position = eyePos0;
outPos.posClipRight = eyePos1;
outPos.ViewportMask = NV_SINGLERT_VIEWPORT_MASK;
```

```
//Multi-View Rendering
float4 eyePos0 = mul(g_ViewProjMatrix[1], worldPos);
float4 eyePos1 = mul(g_ViewProjMatrix[2], worldPos);
float4 eyePos2 = mul(g_ViewProjMatrix[3], worldPos);
float4 eyePos3 = mul(g_ViewProjMatrix[4], worldPos);

outPos.position = eyePos0;
outPos.position_v1 = eyePos1;
outPos.position_v2 = eyePos2;
outPos.position_v3 = eyePos3;

outPos.ViewportMask = NV_SINGLERT_VIEWPORT_MASK;
outPos.ViewportMask2 = NV_SINGLERT_VIEWPORT_MASK_2;
```

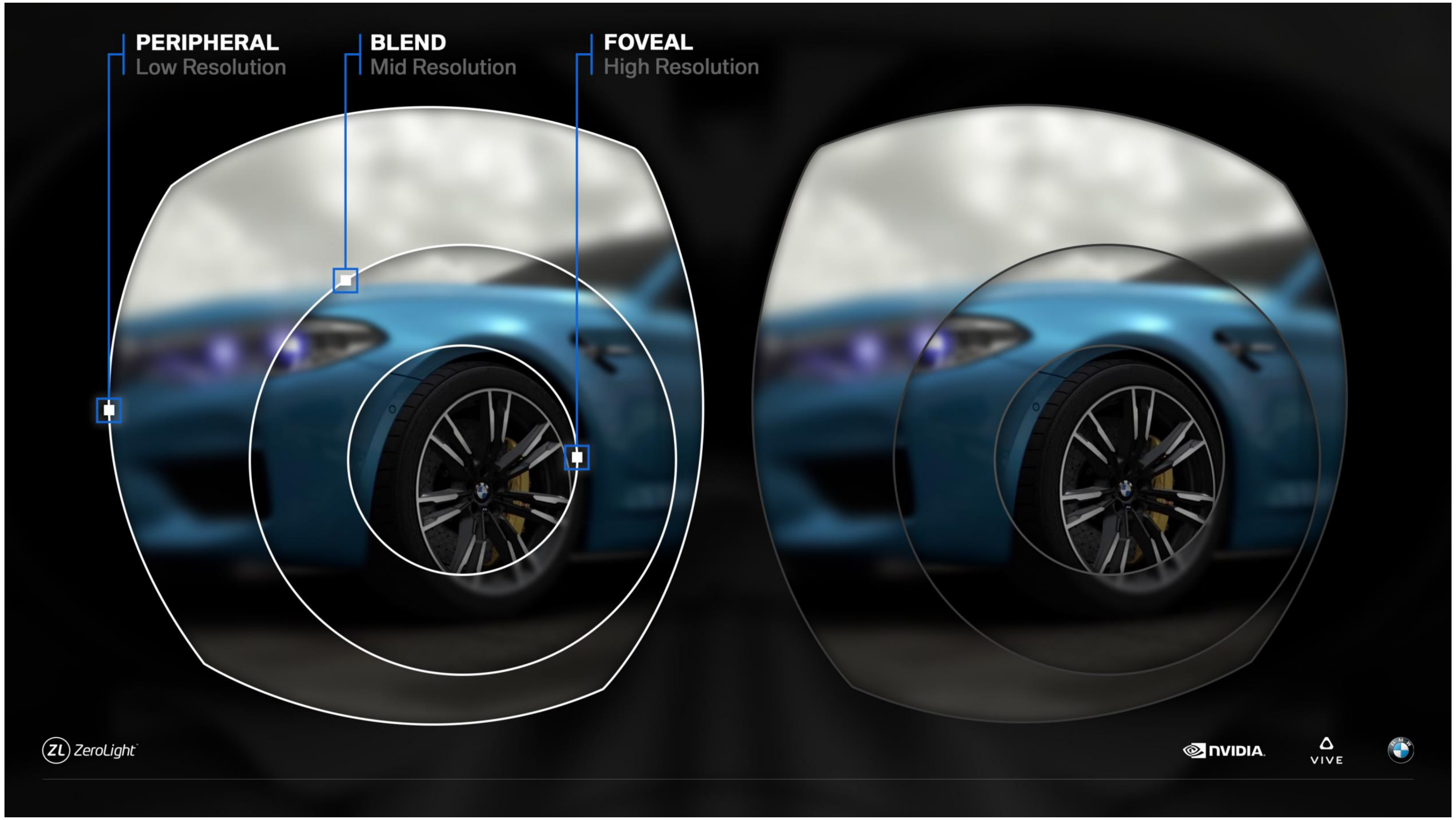
Variable Rate Shading

- Vary the pixel density of your render target
- Big fill rate improvements possible
- Lens-optimised shading and foveated rendering
- The larger the render target the more that can be saved
- Supersampling antialiasing makes VR look great

Variable Rate Shading



Foveated Rendering



Variable Rate Shading

Variable Rate Shading

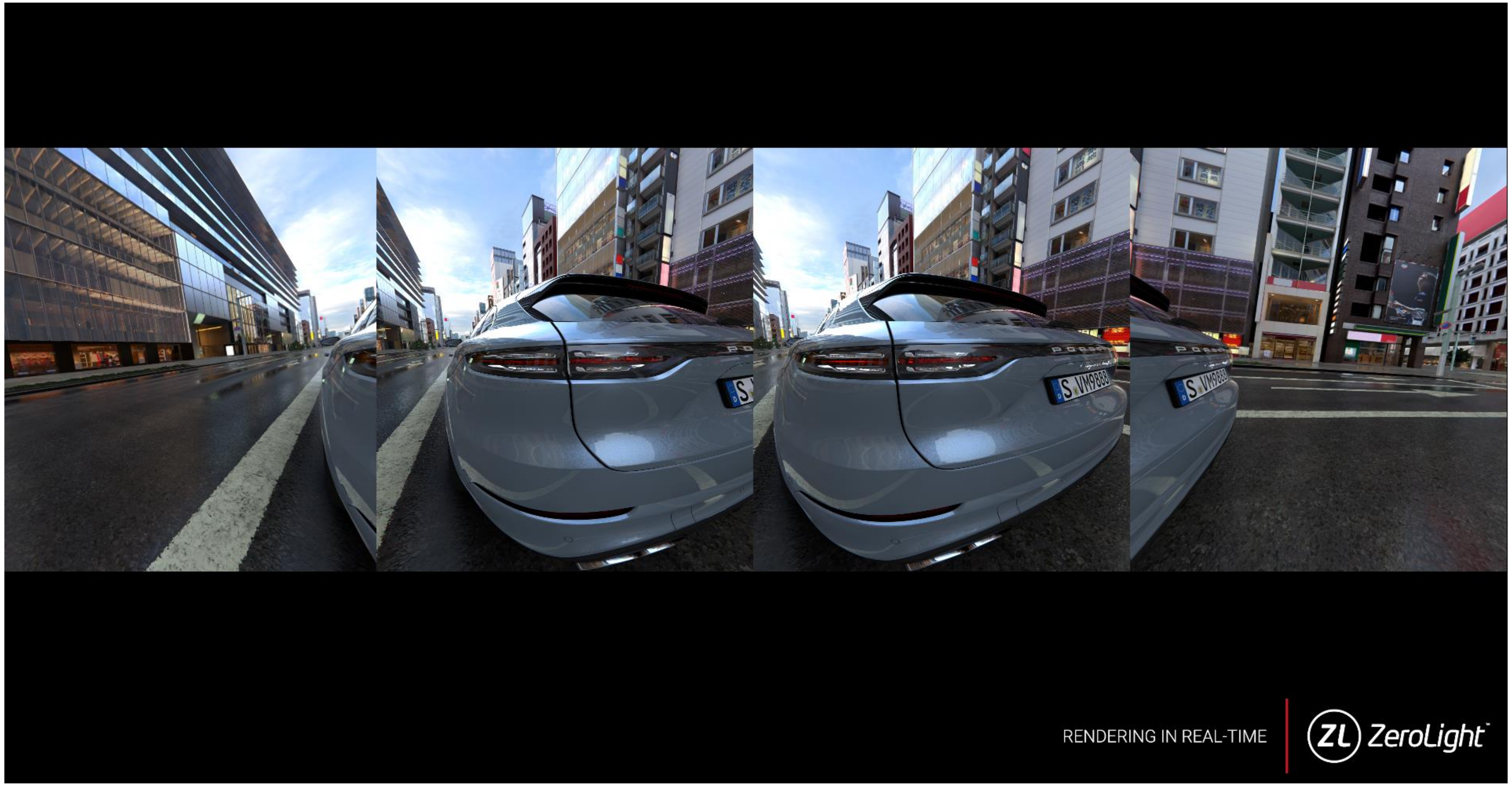
```
void Plugin_Initialize()
{
    //Initialize NVAPI
    NvAPI_Initialize();

    //Check VRS Support
    NV_D3D1x_GRAPHICS_CAPS caps = {};
    if (NvAPI_D3D1x_GetGraphicsCapabilities(d3d11NativeDevice,
NV_D3D1x_GRAPHICS_CAPS_VER, &caps) == NVAPI_OK
        && caps.bVariablePixelRateShadingSupported)
    {
        isTuringCard = true;
    }
}
```

```
//Update texture using compute shader or render
RenderFullScreenWithShader(VRSTexture)
```

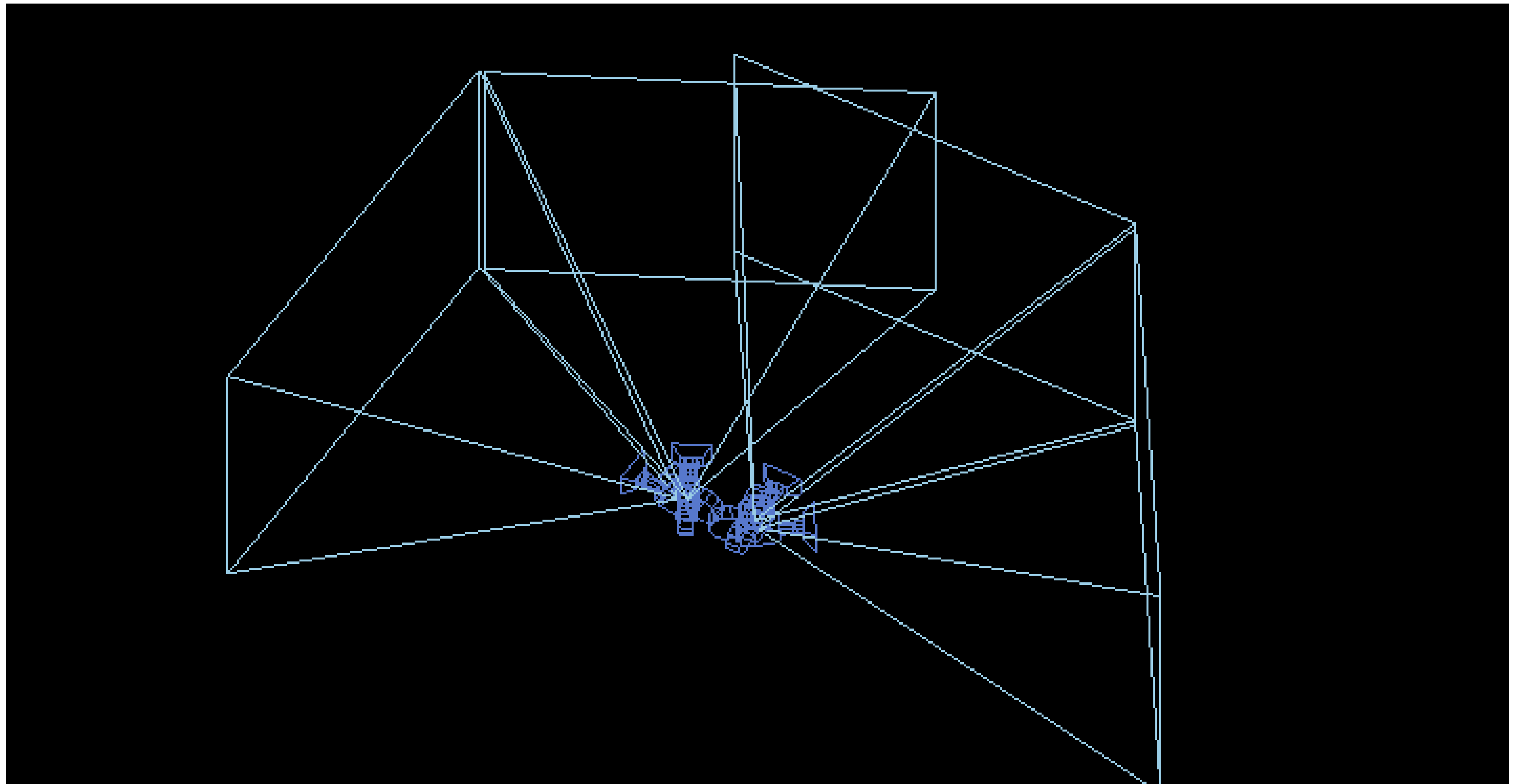
```
//Pre main camera Render
void Plugin_EnableVRS(void* VRStextureResourceView)
{
    //Create look up table with shading rates and set
    vrsViewportDescription.numViewports = 2;
    NvAPI_D3D11_RSSetViewportsPixelShadingRates(deviceContext,
&vrsViewportDescription);

    //Set texture to be used as index into shading rate table
    NvAPI_D3D11_RSSetShadingRateResourceView(deviceContext,
VRStextureResourceView);
}
//Engine main camera render
GameEngineCamera.Render();

//Post main camera render
void Plugin_DisableVRS()
{
    //Create look up table with shading rates and set
    vrsViewportDescription.numViewports = 0;
    NvAPI_D3D11_RSSetViewportsPixelShadingRates(deviceContext,
&vrsViewportDescription);
}
```

Porsche StarVR One

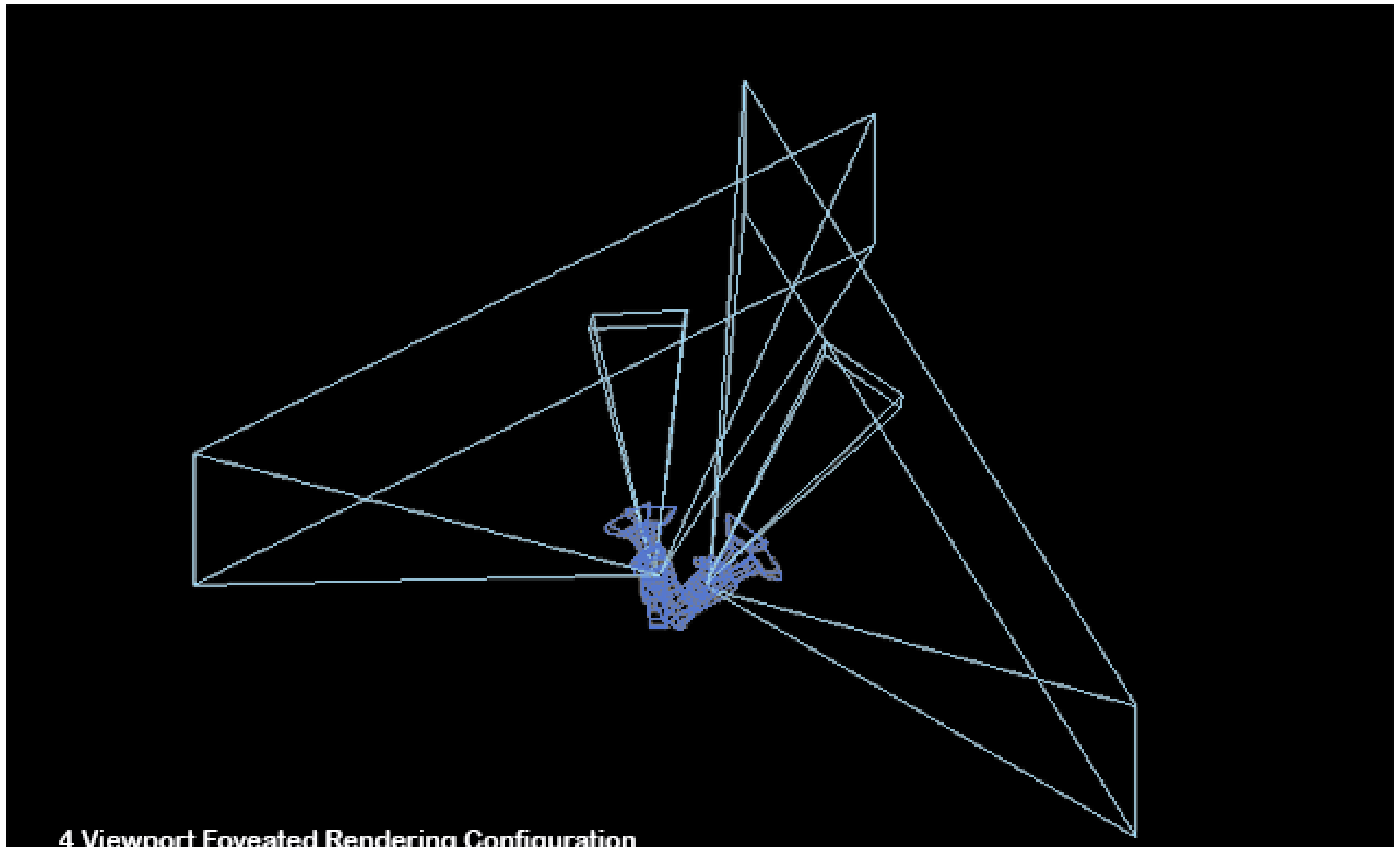
Porsche StarVR One



Porsche StarVR One

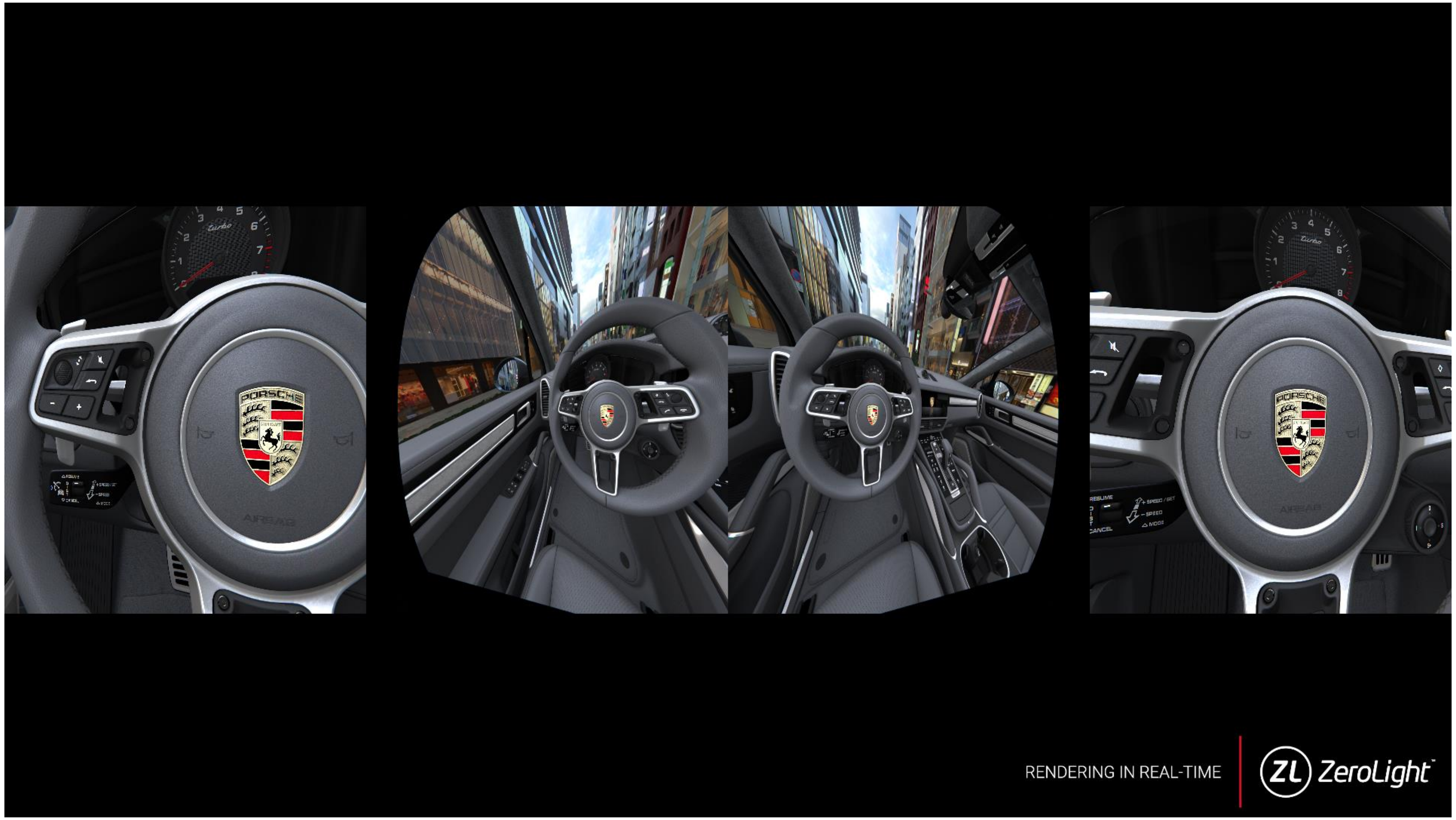


4 Viewport Split over wide FOV configuration



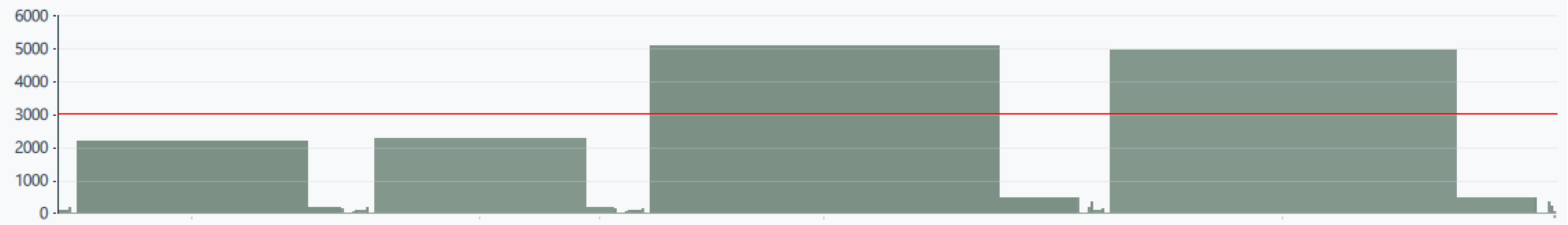
4 Viewport Foveated Rendering Configuration

Porsche StarVR One

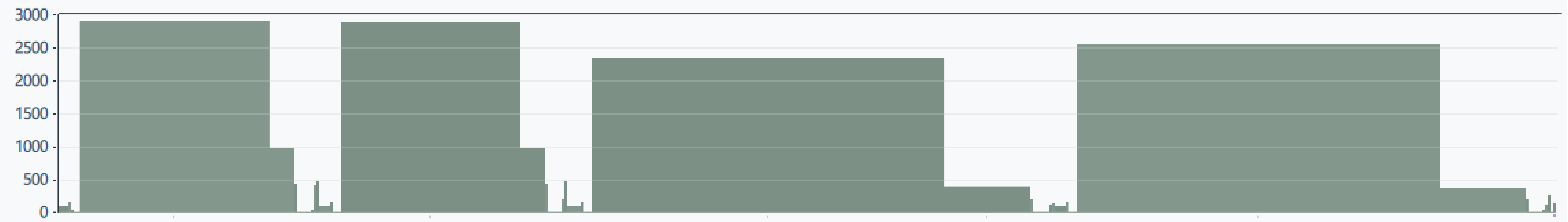


Porsche StarVR One

GPU Profile Standard 4 Viewport wide FOV rendering



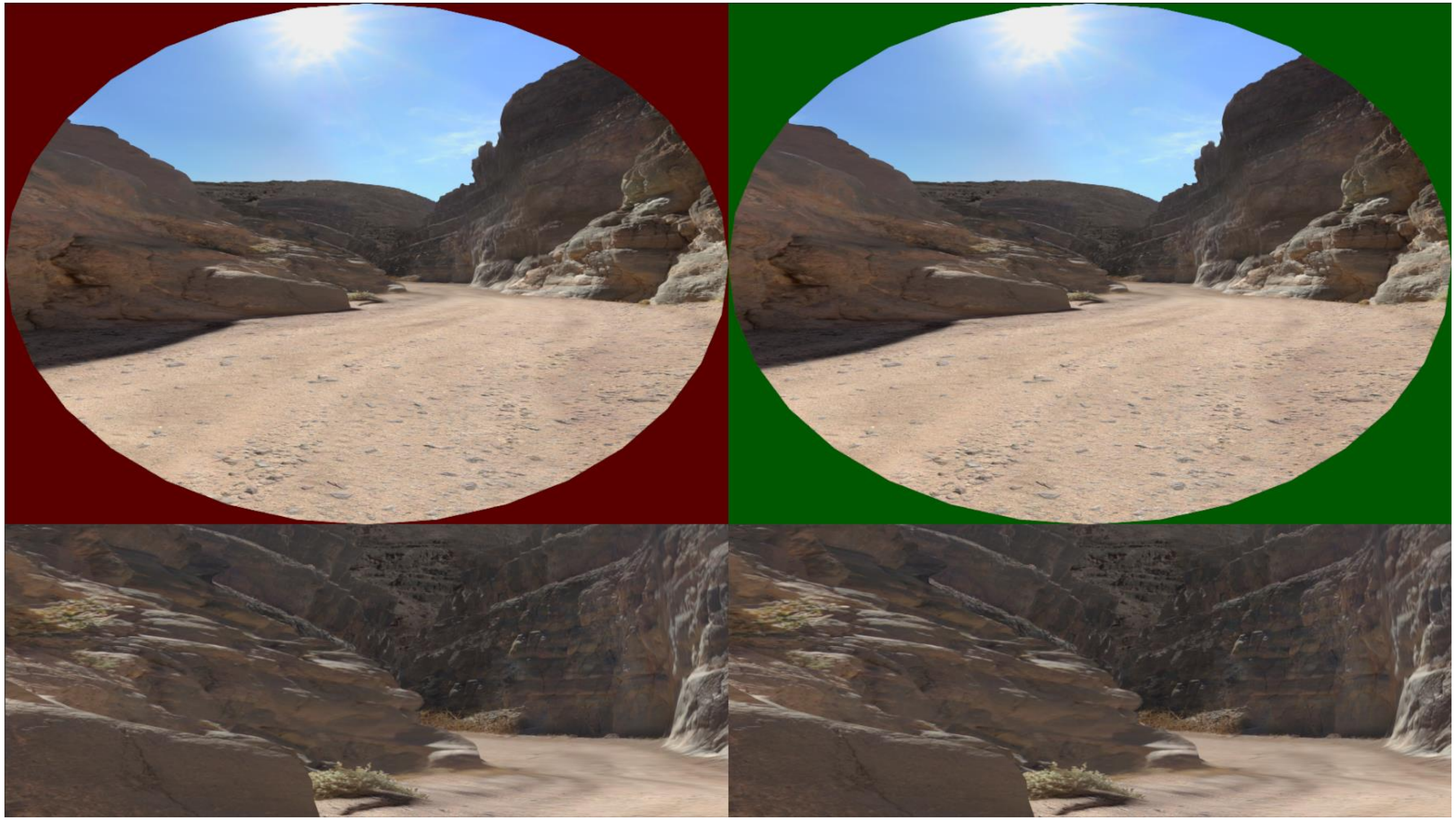
GPU Profile Foveated Rendering 4 Viewports




PORSCHE
SELECT A PAINT
BACK

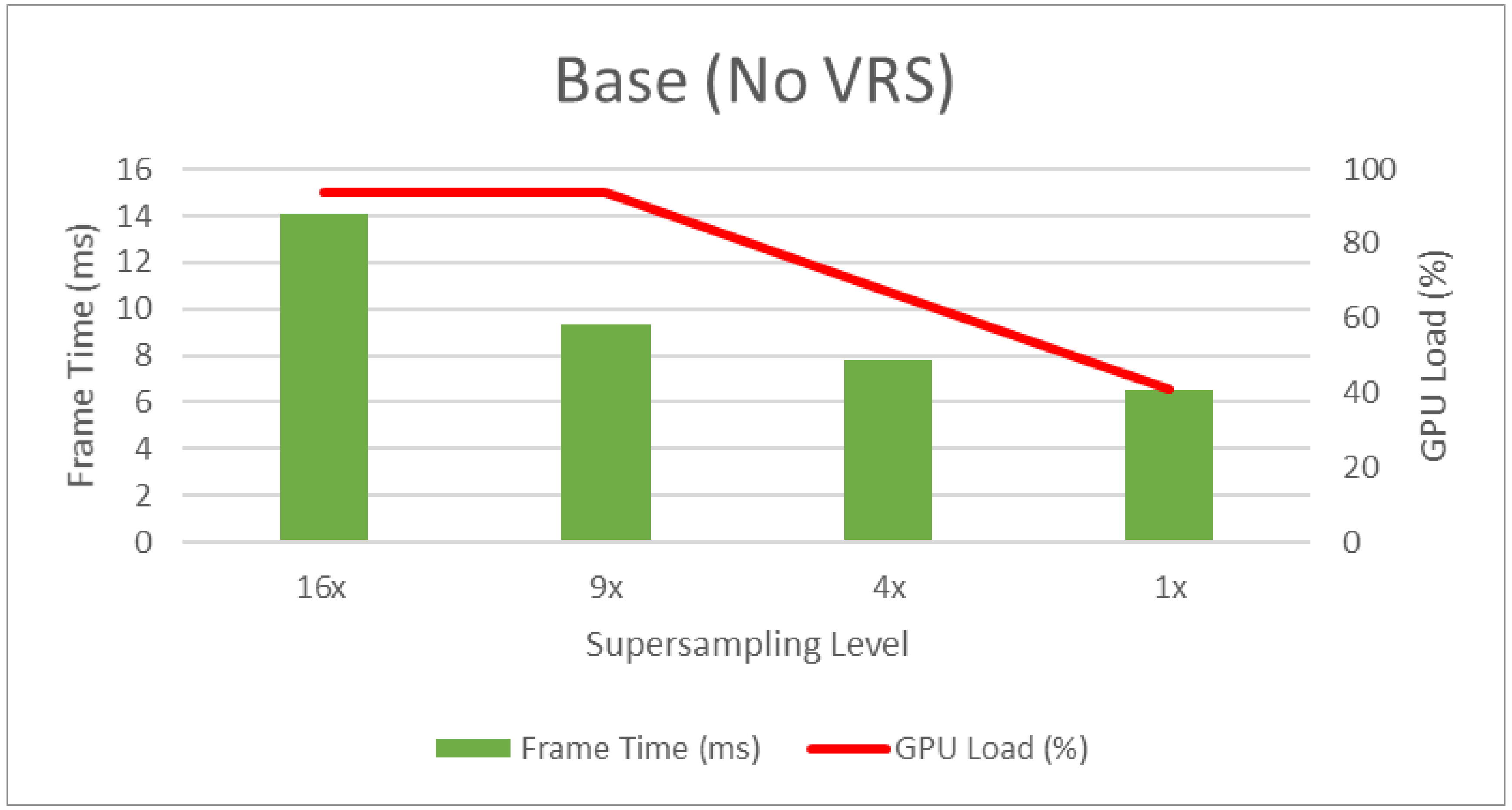
Audi Varjo VR-1

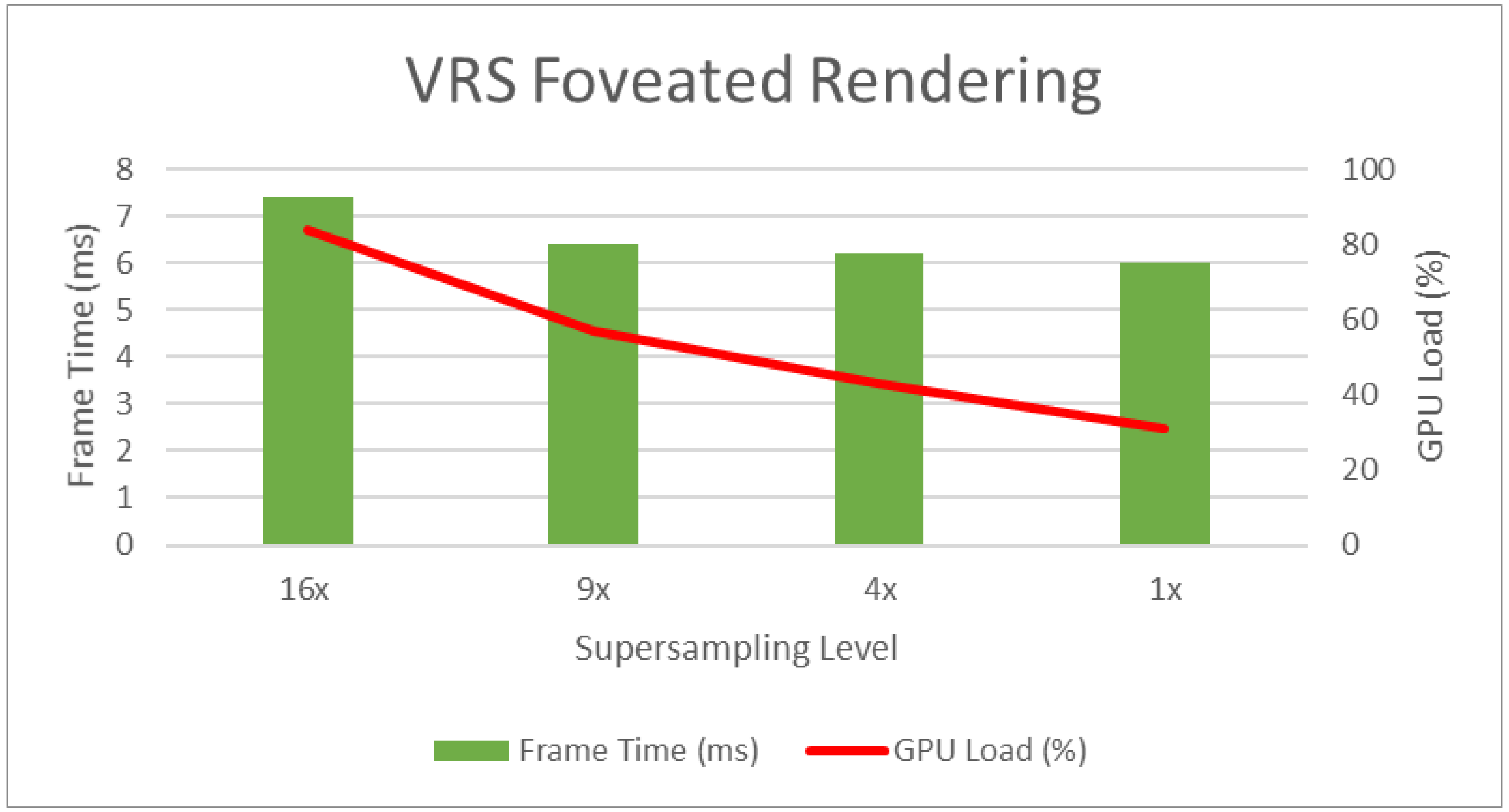
Audi Varjo VR-1



BMW VIVE Pro Eye

BMW VIVE Pro Eye





BMW VIVE Pro Eye

Find Out More

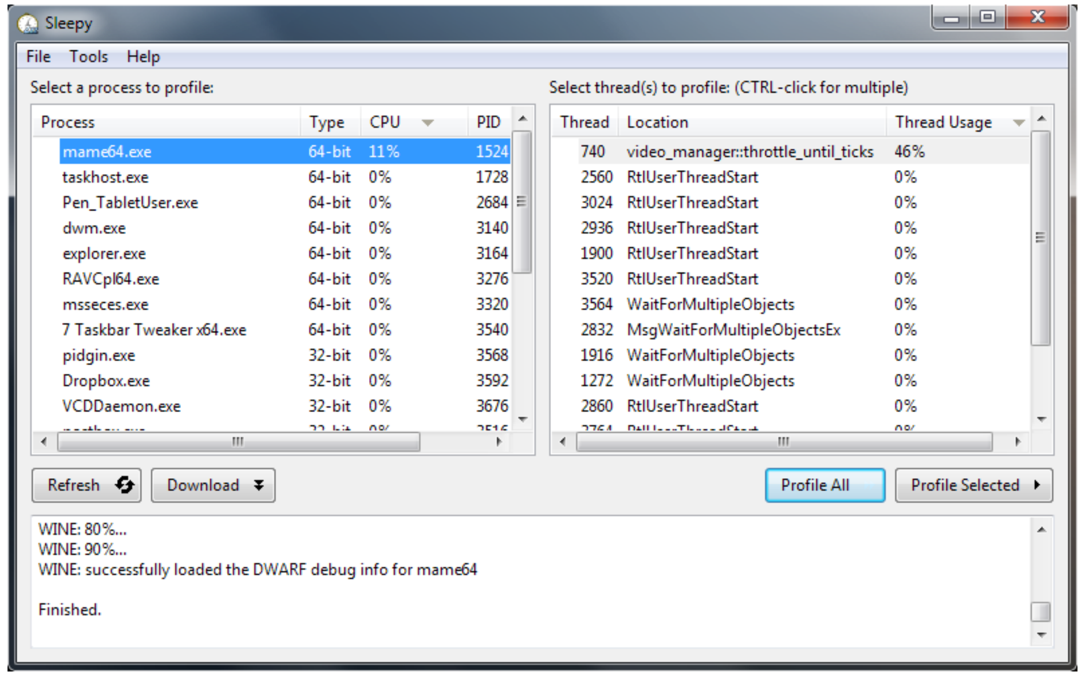
NVIDIA DEVELOPER

NVIDIA Developer Blog

https://devblogs.nvidia.com/

Profiling

NVIDIA® Nsight™



Follow Our Tech Updates on Twitter

@TechZeroLight

Questions?

chris.oconnor@zerolight.com

www.linkedin.com/in/chris-zerolight