March 17-21, 2019 | Silicon Valley

S9717

REAL-TIME RAY TRACING

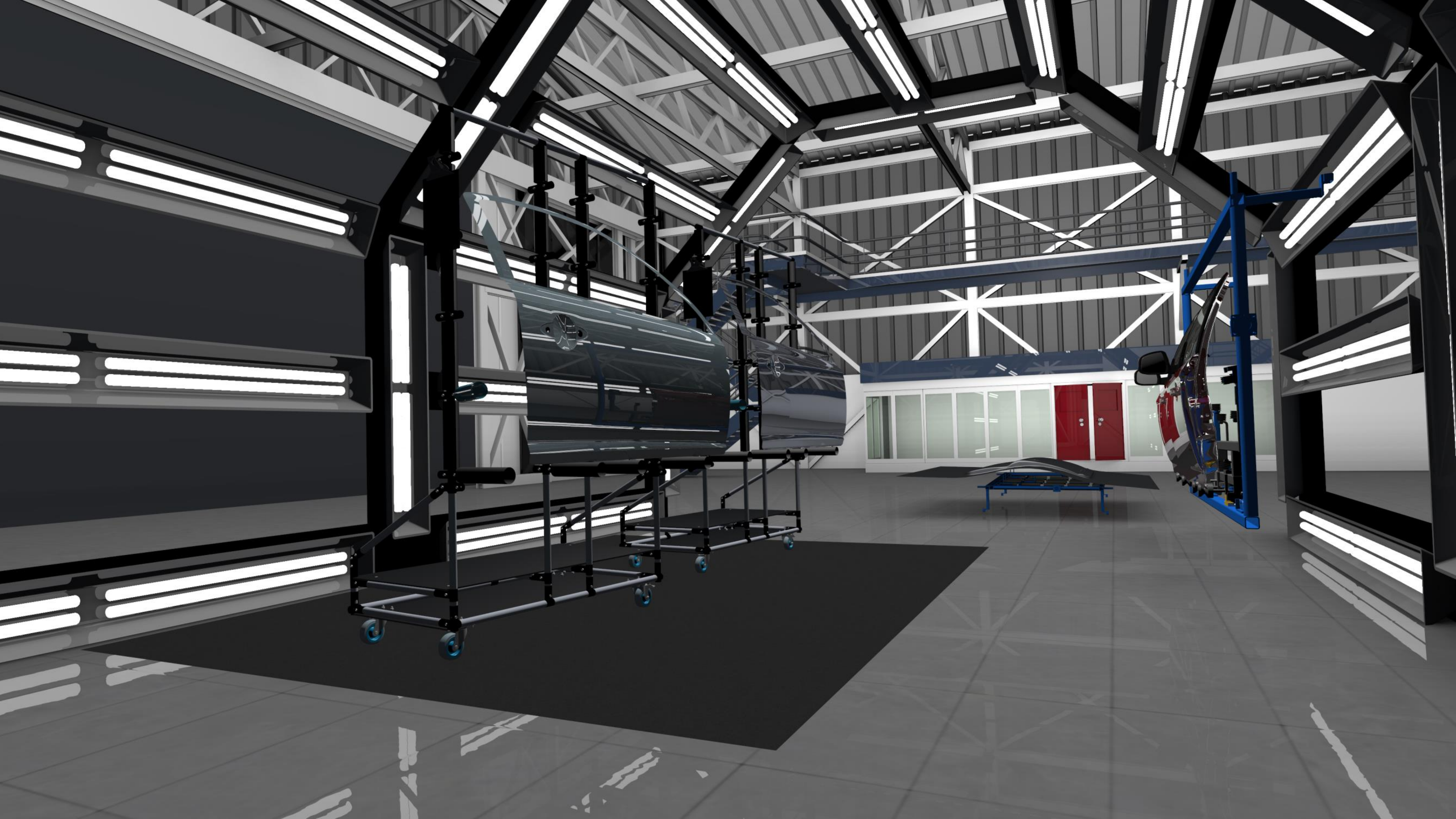
ON PROFESSIONAL HEAD-MOUNTED DISPLAYS WITH **NVIDIA RTX**

Andreas Dietrich, Jan Wurster, ESI Group

March 18, 2019

get it right®

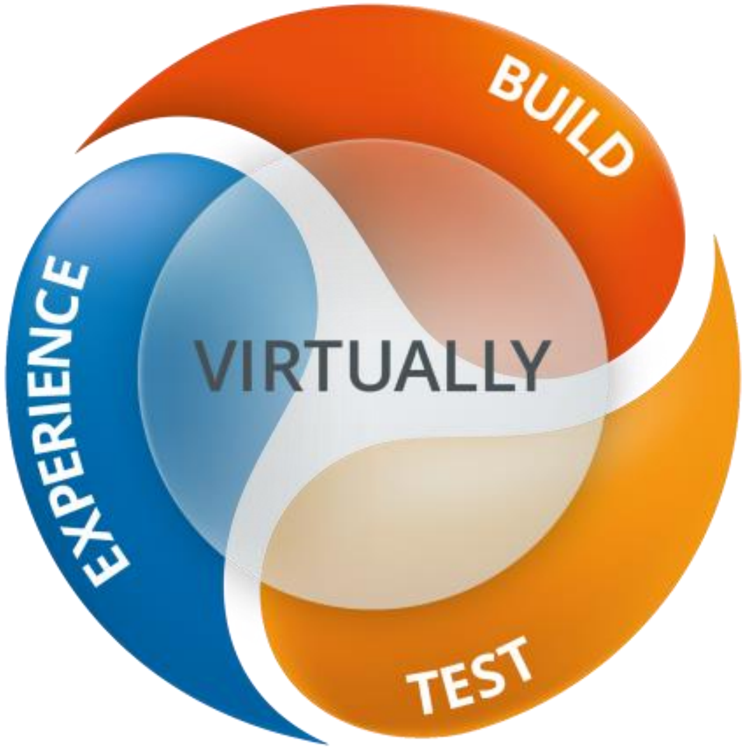
www.esi-group.com



**“Rasterization is fast,
but needs cleverness to support complex visual effects.**

**Ray tracing supports complex visual effects,
but needs cleverness to be fast.“**

„Interactive Ray Tracing with CUDA“ –
David Luebke, Steven Parker, NVIDIA, 2008



"A pioneer and world-leading provider in Virtual Prototyping."

— www.esi-group.com

AGENDA

Introduction

Part 1 : ESI Ray Tracing Technology

- Helios Rendering Architecture
- OptiX Backend
- Ray Tracing on HMDs

Part 2 : Application: Visualization of Stamping Defects

- Sheet Metal Forming
- Cosmetic Defect Prediction
- Immersive Virtual Reality Performance

Introduction

Ray Tracing in VR

- „Real-time“ ray tracing around for decades
  - Mostly used for fast (GI) previews
  - More „interactive“ than „real-time“

- New developments
  - GPU technology: RTX, RT Cores / Turing
  - APIs: DXR, Vulkan extensions

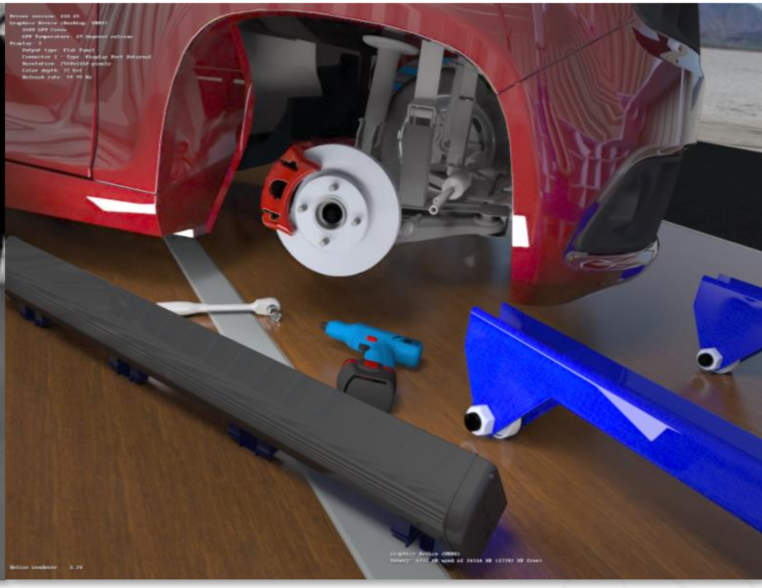
- Enables use in VR
  - Simulate accurate reflections
  - Recreate physical workflows

GPU TECHNOLOGY CONFERENCE

March 17-21, 2019 | Silicon Valley

ESI Ray Tracing Technology





Data Copyright Volkswagen AG

esi get it right®

Andreas Dietrich

www.esi-group.com

Helios Rendering Architecture

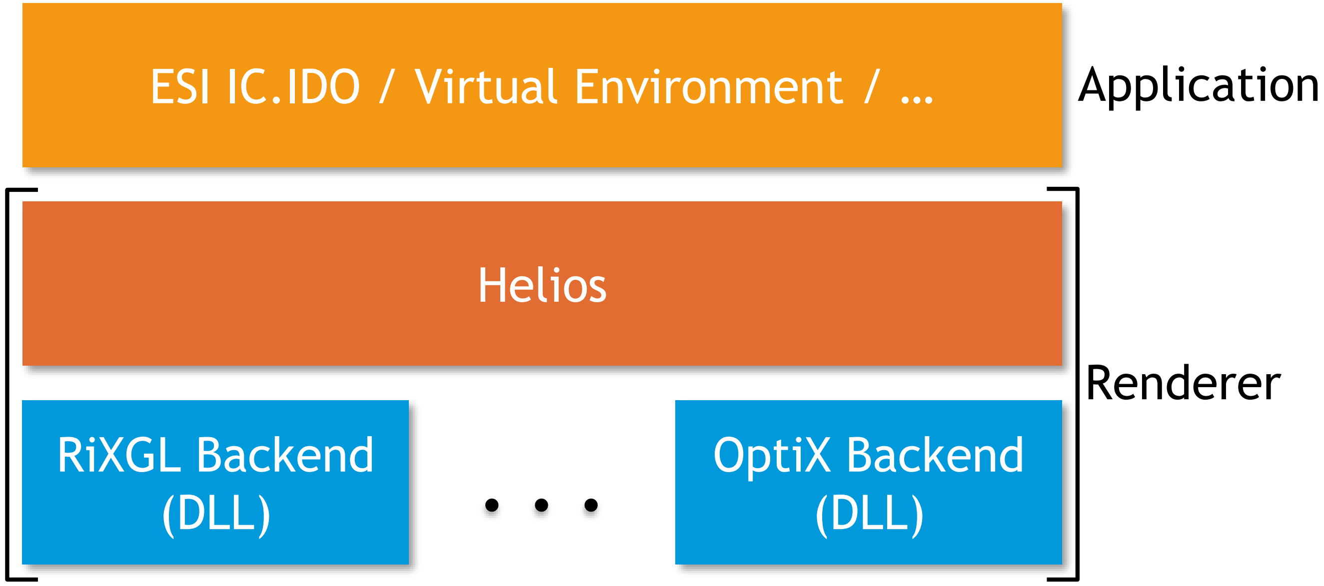
Requirements

- Lightweight
  - Not a full-blown scene graph, let application handle this
  - Replicate as few data as possible
- High performance
  - Exploit specific/low-level GPU features
  - Must be able to get close to the metal
- Extensible
  - Leverage external rendering frameworks, such a NVIDIA RiX or OptiX
- Support for various rendering algorithms
  - Rasterization, ray tracing, hybrid modes
  - CAD rendering, photo-realistic rendering, scientific visualization, …

Helios Rendering Architecture
Overview

"Challenges in Real-Time Rendering and Software Design for Interactive Immersive Visualization"
Andreas Dietrich, ESI Group,
Galen Faidley, Caterpillar Inc.
— GTC 2018, S8433

- Helios renderer supports a variety of rendering technologies
- Encapsulated in Backends, e.g.,
  - RiXGL: rasterization
  - OptiX: ray tracing
  - Coming soon: Vulkan
- Backends as dynamic libraries (DLLs)
  - Loaded and unloaded at runtime
  - Can be switched arbitrarily
- Helios controls render graph, e.g.,
  - Hybrid rendering
  - Postprocessing (e.g. anti-aliasing filter)



Helios OptiX Backend

Overview

- Based on NVIDIA OptiX:
    - Programmable GPU ray tracing pipeline
    - Single-ray programming model using C++
    - AI accelerated rendering

- Implements a range of physically based rendering algorithms
    - Whitted-style ray tracing, Ambient Occlusion, Global Illumination
    - Generates precomputed lighting data (e.g., texture baking)
    - Separation of BSDF and integrator code (surface shading / light transport)
    - Supports Material Definition Language (MDL)

OptiX

NVIDIA RTX

- RTX
  - Ray tracing functionality in the GPU driver
  - Use of RT Cores on Turing GPUs
    - Traversal and intersection in hardware

- Support in OptiX
  - Available since OptiX 5 (optional)
  - Per default active in OptiX 6

```
bool RTXEnabled = true;
RTresult code = rtGlobalSetAttribute(RT_GLOBAL_ATTRIBUTE_ENABLE_RTX, sizeof(RTXEnabled), (void*)&RTXEnabled));
```

- Performance improvement also for pre-Turing GPUs

OptiX

GeometryTriangles

- OptiX GeometryTriangles node
  - New in OptiX 6.0
  - Complements Geometry node
  - Built-in support for triangles
  - OptiX provides intersection program and triangle-aware BVH build
  - RT Core accelerated on Turing

- Application provides
  - Index and vertex buffers
  - Attribute program

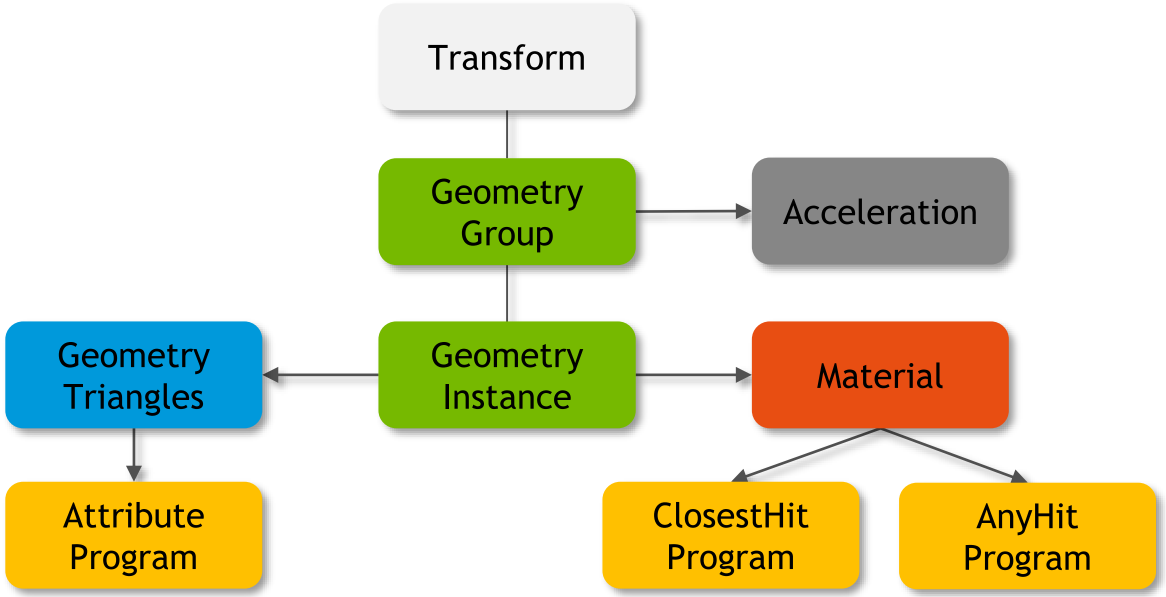
OptiX

GeometryTriangles Example

```
// Create geometry triangles node
optix::GeometryTriangles geometryTriangles = context->createGeometryTriangles();

// Setup triangle buffers
optix::Buffer indexBuffer  = context->createBuffer(RT_BUFFER_INPUT, RT_FORMAT_UNSIGNED_INT3, 0);
optix::Buffer vertexBuffer = context->createBuffer(RT_BUFFER_INPUT, RT_FORMAT_FLOAT, 0);

fillBuffers(indexBuffer, vertexBuffer);

unsigned vertexCount   = ... // number of vertices of the geometry
RTsize   vertexOffset  = ... // offset in bytes to the first vertex in buffer vertexBuffer
RTsize   vertexStride  = ... // stride in bytes between vertices
unsigned primtiveCount = ... // number of triangles

// Set triangle data
geometryTriangles->setTriangleIndices(indexBuffer, /*index format*/ RT_FORMAT_UNSIGNED_INT3);
geometryTriangles->setVertices(vertexCount, vertexBuffer, vertexOffset, vertexStride, /*vertex format*/ RT_FORMAT_FLOAT3);
geometryTriangles->setBuildFlags(RT_GEOMETRY_BUILD_FLAG_NONE);
geometryTriangles->setPrimitiveCount(primitiveCount);

// Set attribute program
geometryTriangles->setAttributeProgram(context->createProgramFromPTXFile(filepath, "attributes"));

// Set buffers for attribute programs
geometryTriangles["indexBuffer"]->setBuffer(indexBuffer);
geometryTriangles["vertexBuffer"]->setBuffer(vertexBuffer);
```

OptiX Node Graph

Object Nodes and Programs

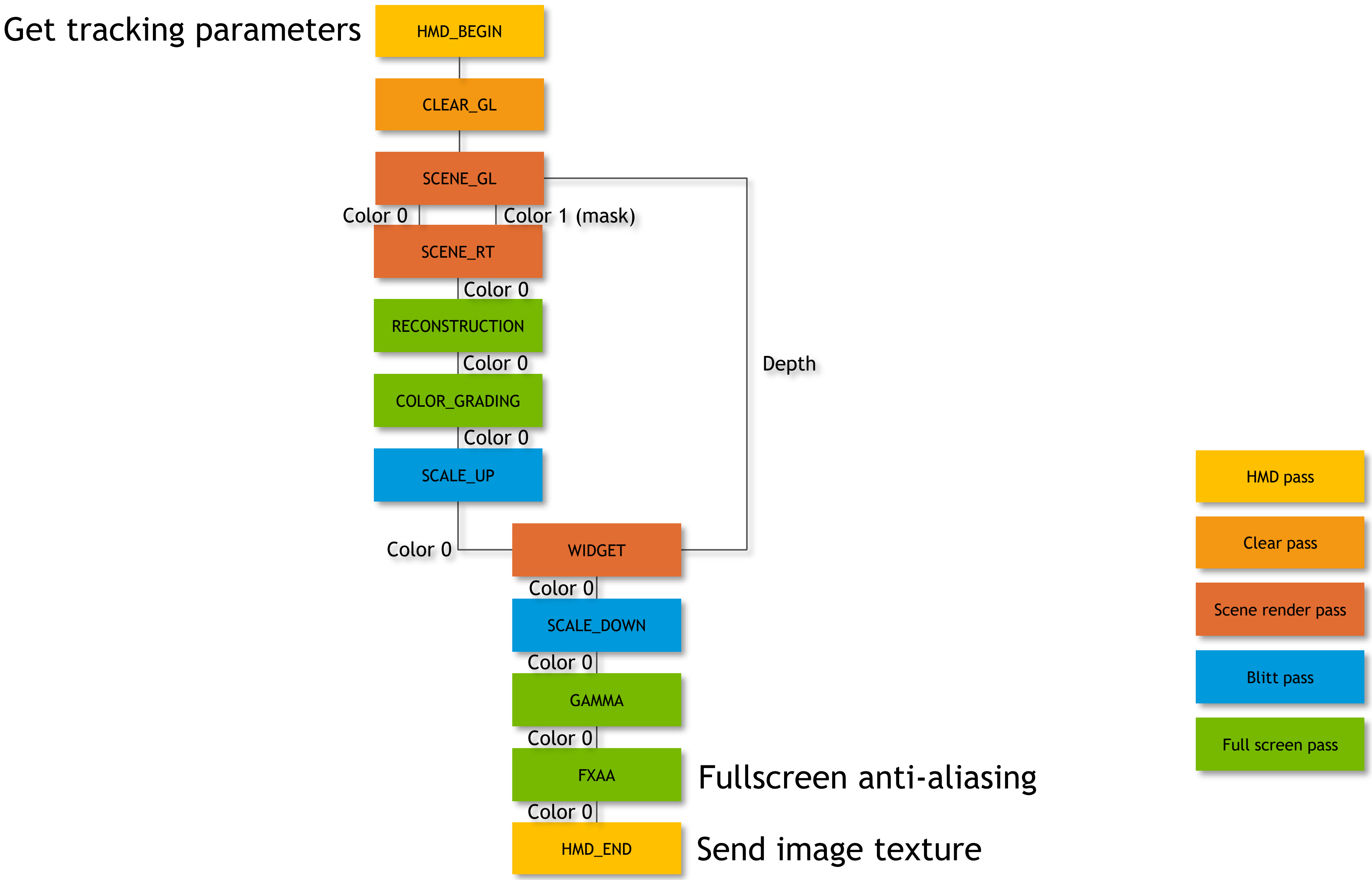


OptiX

Attribute Program Example

```
rtDeclareVariable(float3, normal, attribute NORMAL, );

rtBuffer<int3>  indexBuffer;
rtBuffer<float> vertexBuffer;

RT_PROGRAM void attributes()
{
    // Get vertex indices
    const int  primitiveIndex = rtGetPrimitiveIndex();
    const int3 vertexIndex    = indexBuffer[primitiveIndex];

    // Get barycentric coordinates
    const float2 barycentrics = rtGetTriangleBarycentrics();
    const float  beta  = barycentrics.x;
    const float  gamma = barycentrics.y;
    const float  alpha = 1.0f - beta - gamma;

    // Read vertex normals
    const float3 n0 = getNormal(vertexBuffer, vertexIndex.x);
    const float3 n1 = getNormal(vertexBuffer, vertexIndex.y);
    const float3 n2 = getNormal(vertexBuffer, vertexIndex.z);

    // Interpolate shading normal
    normal = normalize(n0*alpha + n1*beta + n2*gamma);
}
```

HMD Ray Tracing Rendergraph

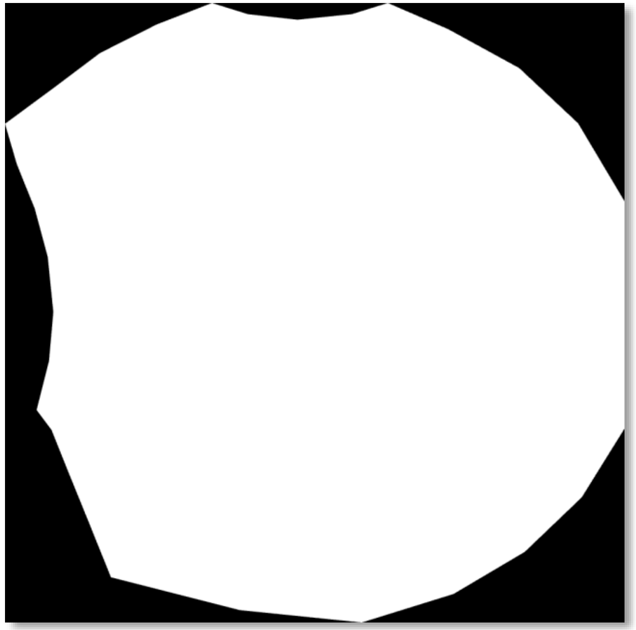


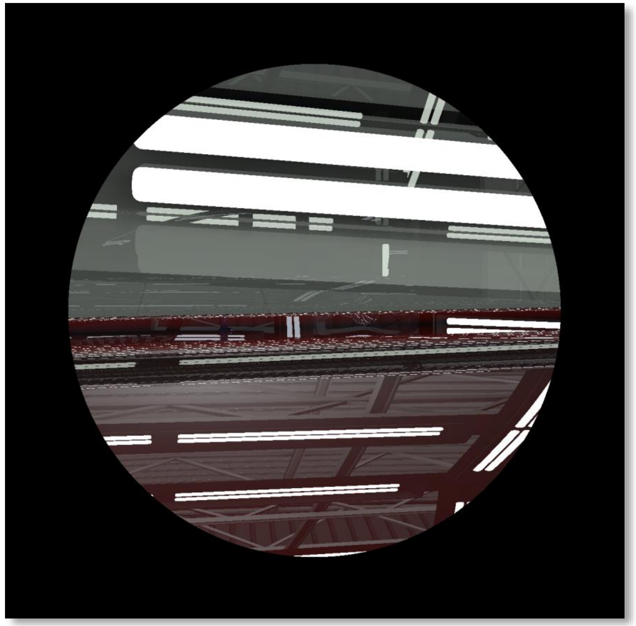
Foveated Rendering

Skipping invisible pixels

- Hidden Area Mesh
  - Provided by OpenVR SDK
  - Defines visible area within image
  - Depends on HMD optics (e.g., lens distortion)
  - Roughly circular on Vive Pro

- Exploit in OptiX ray generation program
  - Skip computing pixels outside disc
  - Disc diameter about 80% of box width
  - Avoid visibility computation **and** shading

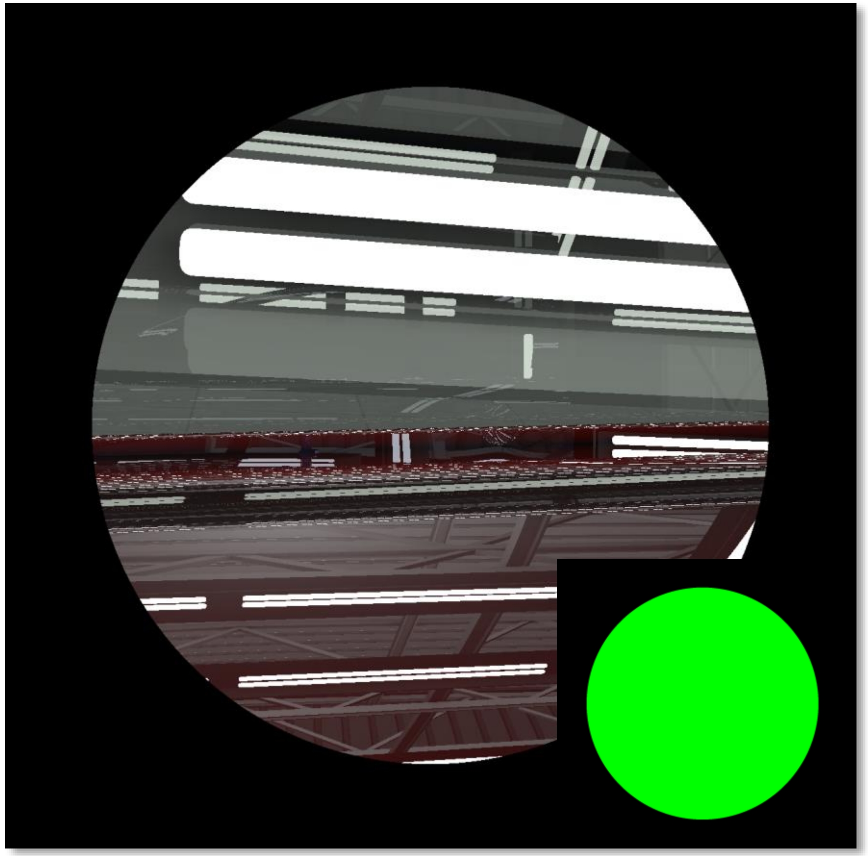


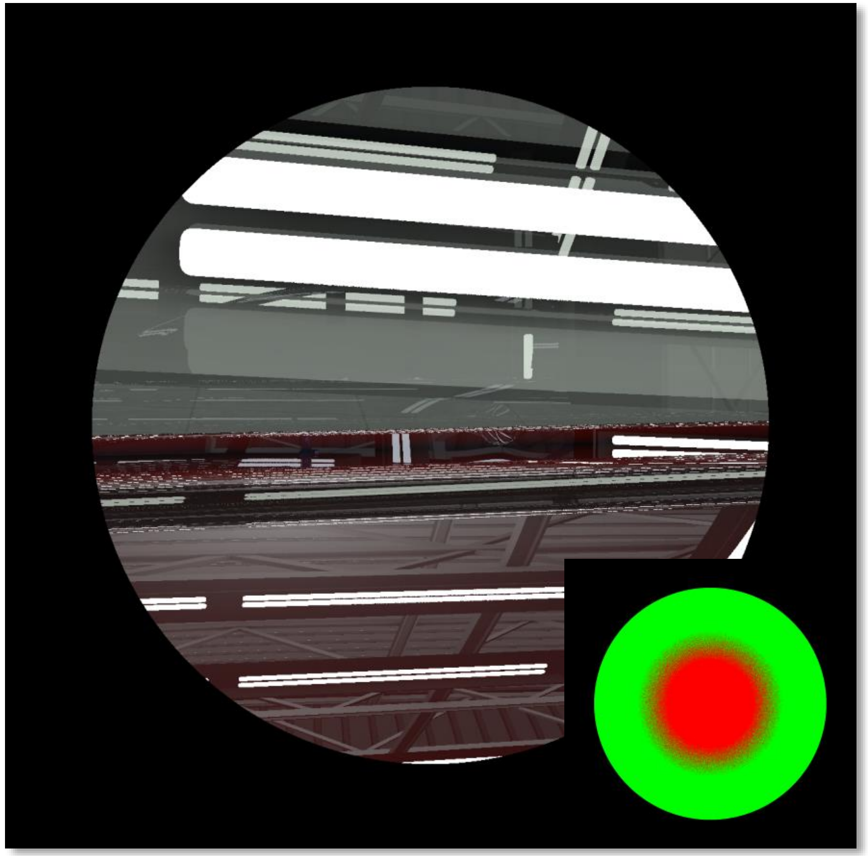


get it right®

Foveated Rendering

Experimental Variable Rate Sampling





Green: 1 sample / pixel
Red: 4 samples / pixel

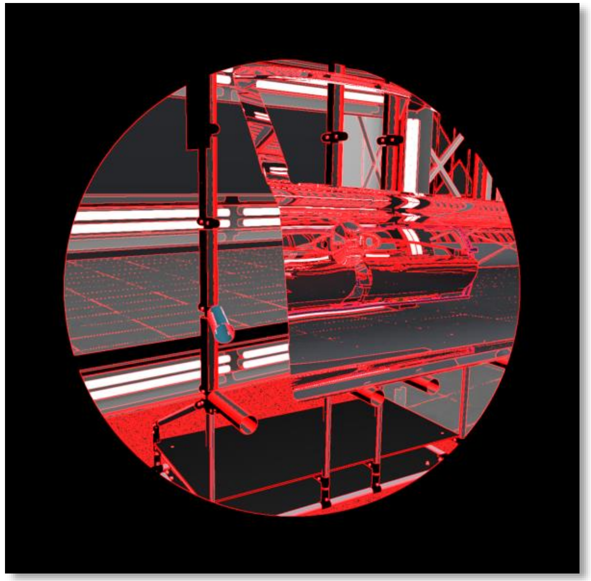
Anti-Aliasing

Fast ApproXimate Anti-Aliasing (FXAA)

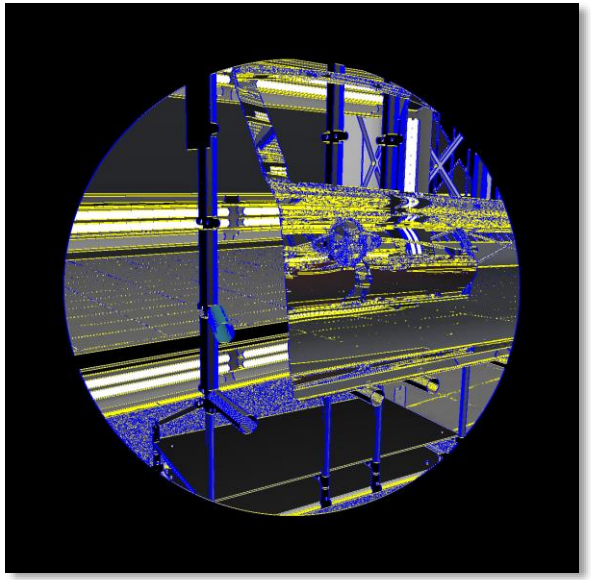
- FXAA
  - Fullscreen post process (GLSL)
  - Edge-aware low-pass filter

- Basic algorithm
  - Detect edges based on contrast difference
  - Approximate luminance gradient
  - Filter along axis perpendicular to gradient

> "FXAA"
>
> Timothy Lottes, NVIDIA
>
> — NVIDIA White Paper, 2009



Edge detection

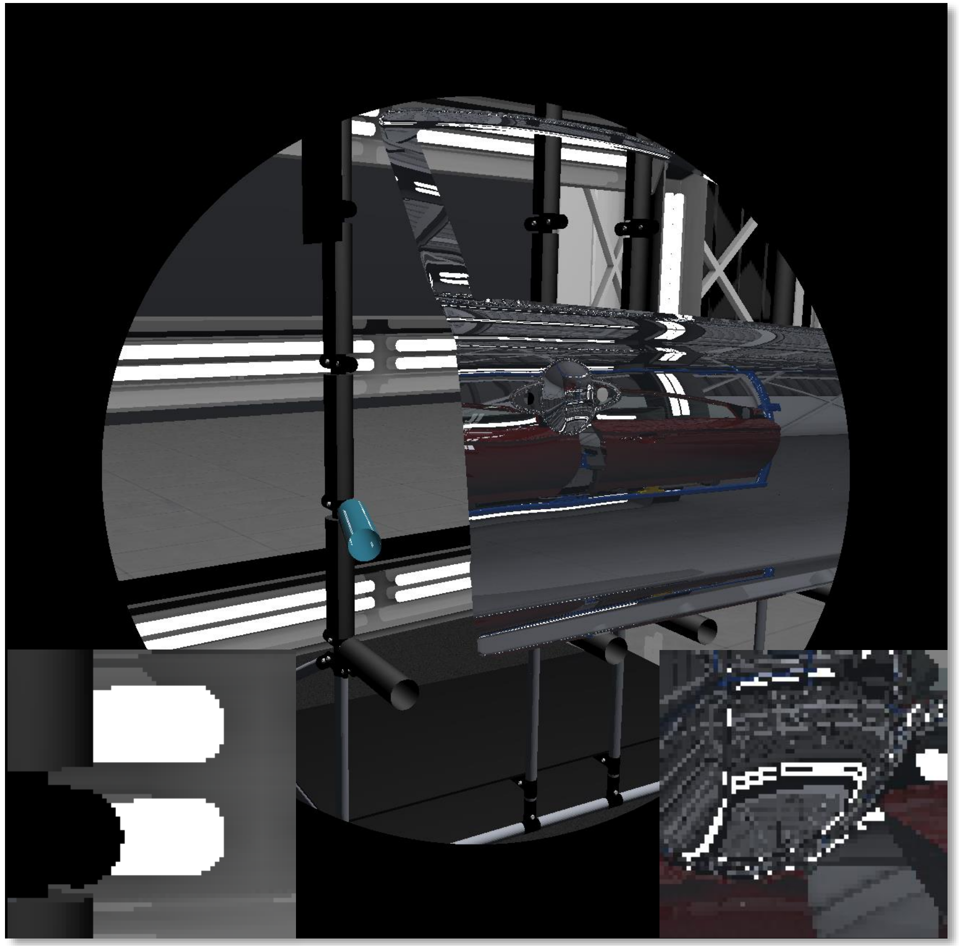


Main filtering direction

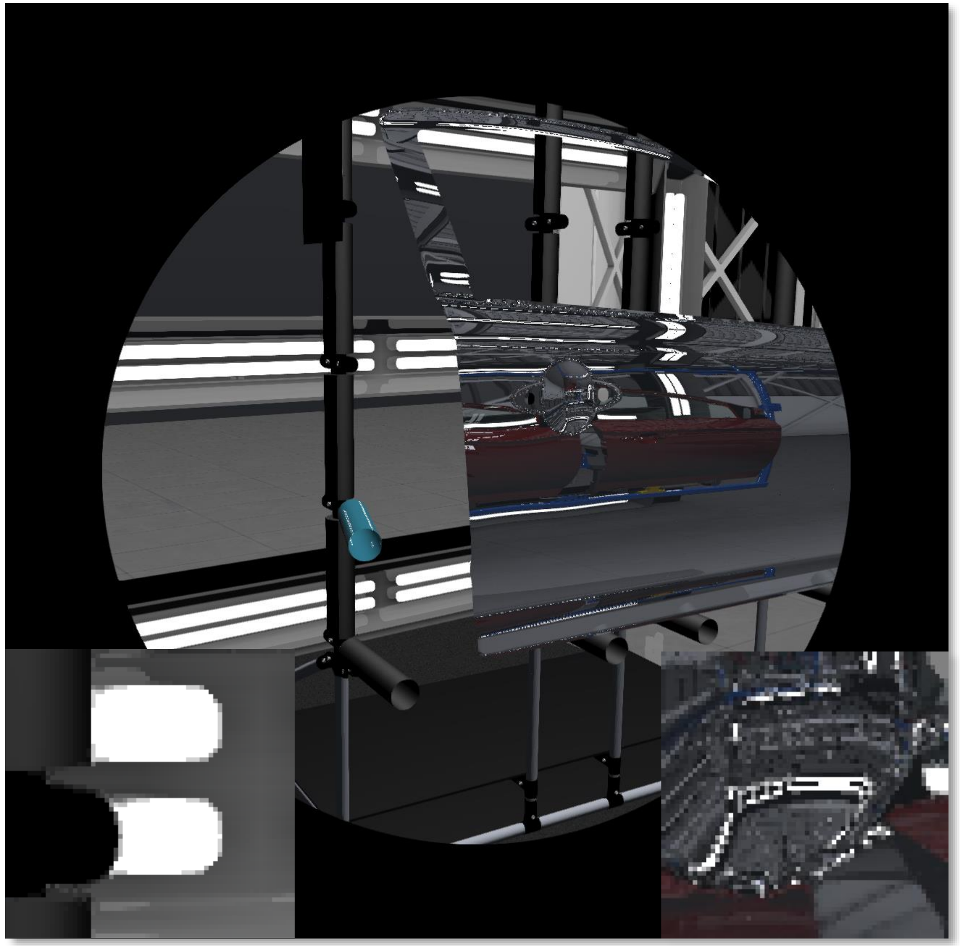
blue: vertical
yellow: horizontal

Anti-Aliasing

Fast ApproXimate Anti-Aliasing (FXAA)



Without FXAA



With FXAA

Hardware Setup

Graphics Processing Unit

- NVIDIA Quadro RTX 8000
  - Turing architecture
  - 4608 CUDA cores
  - 576 Tensor cores
  - 72 RT cores
  - 48 GB device memory

www.nvidia.com

Hardware Setup

Head-Mounted Display

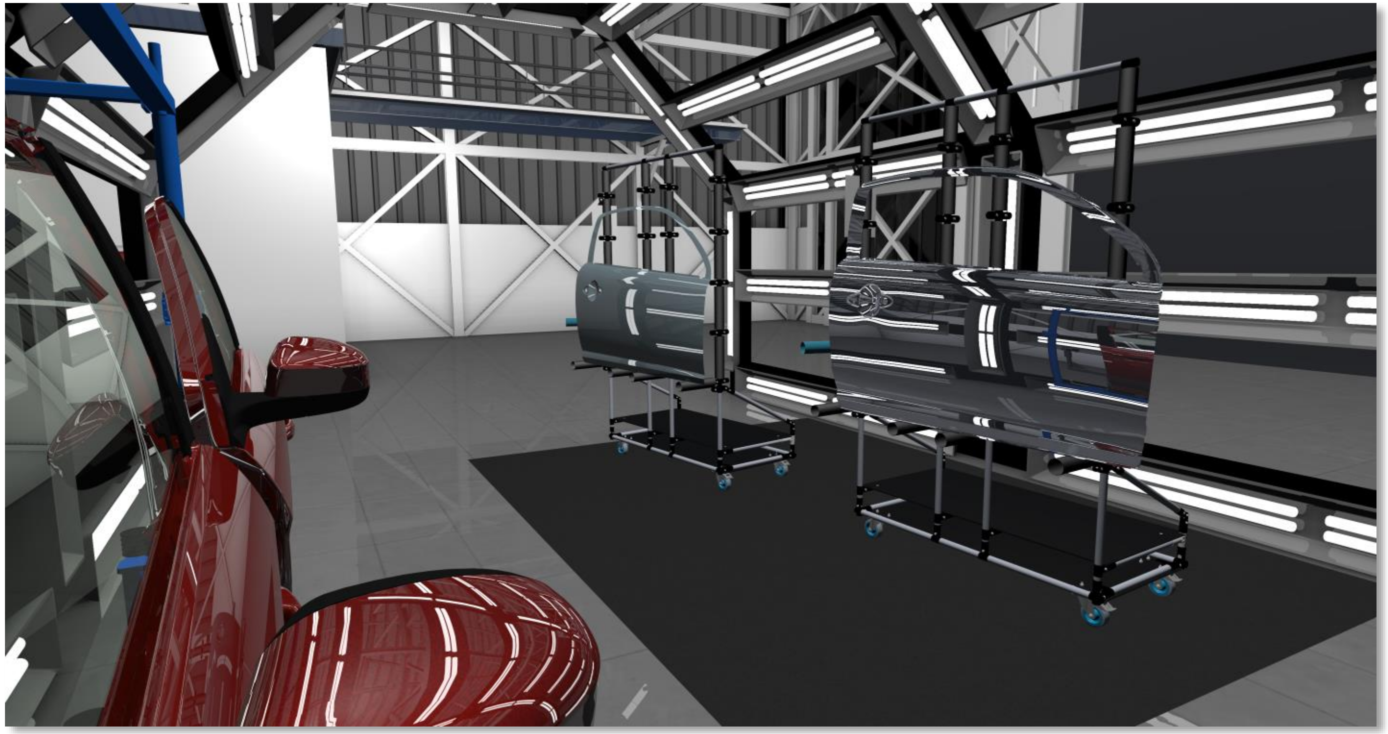
- HTC Vive Pro
  - 1440 x 1600 pixels per eye
  - 90 Hz refresh rate
  - 110 degrees field of view

www.vive.com

Performance

Sheet Metal Forming – Cosmetic Defect Prediction

- Demo Scene
  - 3.9 Mio triangles
  - ~1200 individual objects
  - 3 levels of reflection
  - Point and directional lighting
  - Baked ambient occlusion



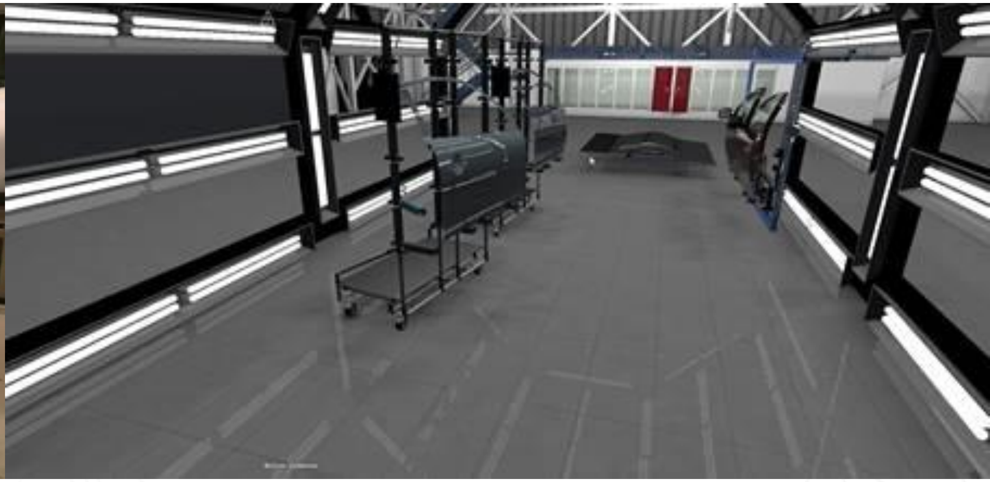
- Ray tracing backend provides
  - 2016 x 2240 pixels per eye
    - Super-sampled anti-aliasing
    - 100% resolution in SteamVR settings
    - Filtering done by HMD
  - 30 – 45 frames per second
    - HMD performs asynchronous reprojection to reduce judder

GPU TECHNOLOGY CONFERENCE

March 17-21, 2019 | Silicon Valley

Advanced Visualization of Stamping Defects

ESI – Sheet Metal Forming – Cosmetic Defect Prediction

Jan Wurster

get it right®


2534 LBS

Sheet Metal Forming - Cosmetic Defects Prediction

Applications in Manufacturing

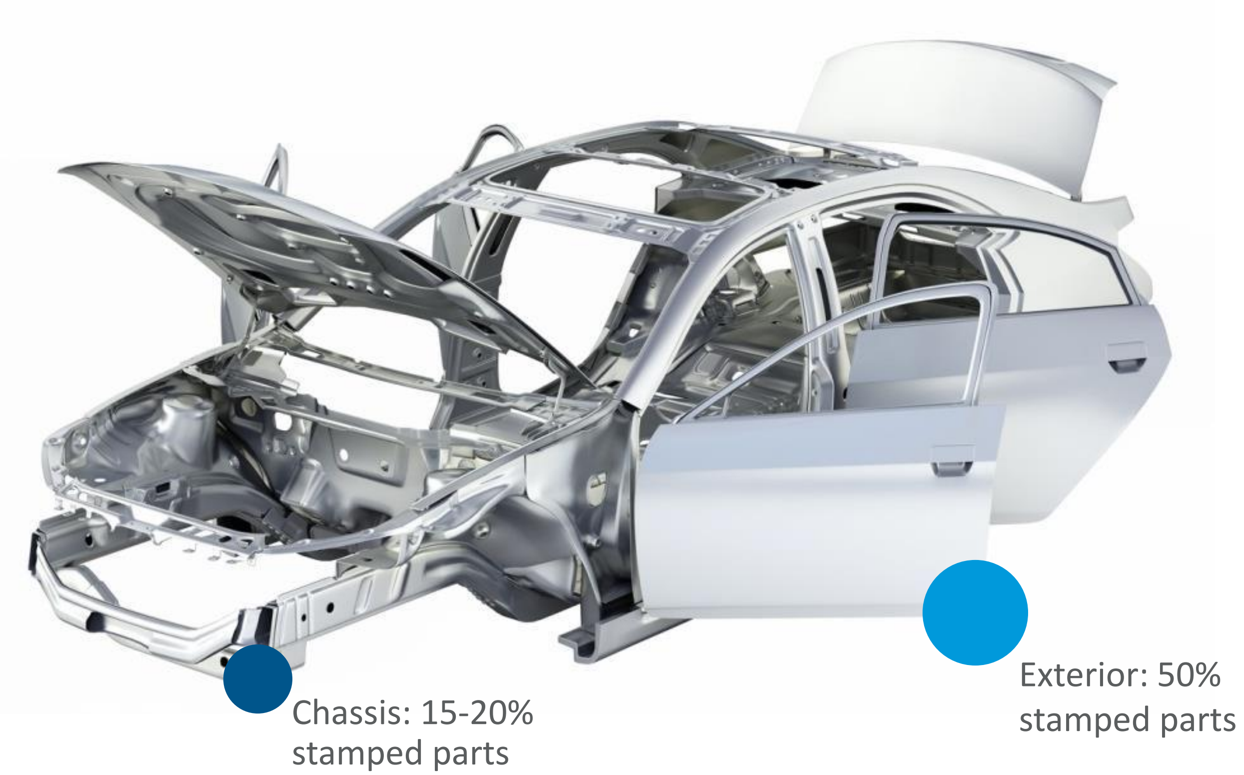
Sheet Metal Forming summarizes a number of metal forming techniques in mass manufacturing:

- Stamping
- Punching
- Blanking
- Embossing
- Bending
- Coining

Most common raw materials to form are sheet metal, other applications include materials such as polysterene.

Sheet Metal Forming - Cosmetic Defects Prediction

Manufacturing Industry Applications



- Introduced in **bicycle manufacturing** around 1880 to replace forged parts
- Rapidly taken up by other industries for efficiency and scalability of the process
- Major pillar in Manufacturing: Aerospace, Appliances, Construction, Automotive, Appliances, …
- Automotive Industry
  - Exterior Surfaces (Class A)
  - Structural Components

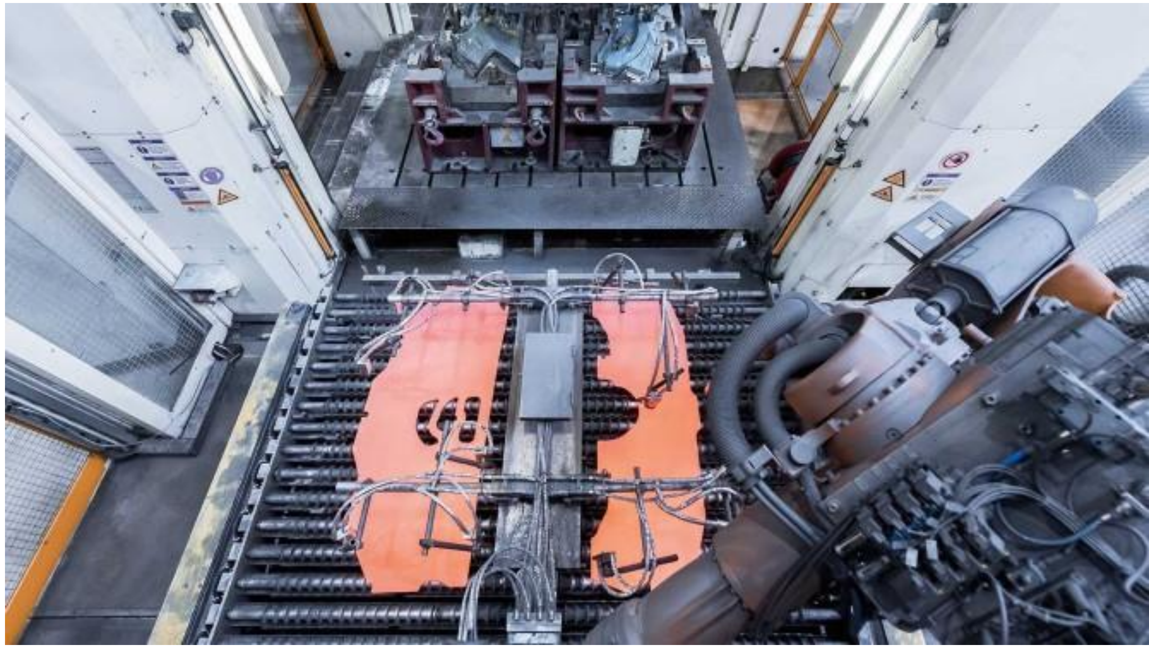
Source: Roland Berger

Sheet Metal Forming - Cosmetic Defects Prediction

Sheet Metal Forming - Manufacturing Process

Trends and **Challenges** in Sheet Metal Forming

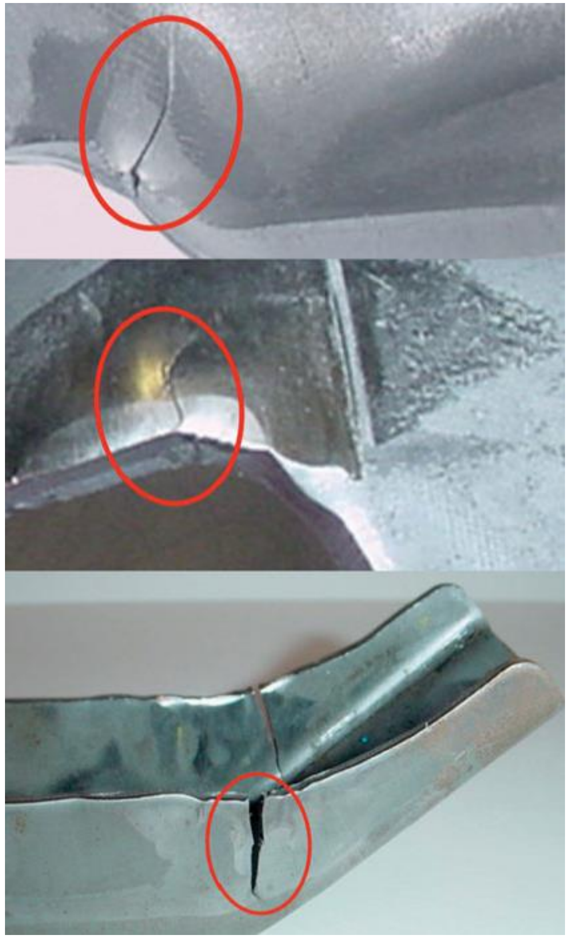
- **New Techniques** - Hot Forming / Press Hardening
  - Complex Shapes
  - Higher Tensile Strength – key in EV and lightweight vehicle initiatives
  - Reduction of **Springback** and **Warping** in panels (by relieving internal stress)
- **Quality and Yield in Manufacturing**
  - ‘As Designed’ vs ‘As Manufactured’
  - Die Design to counter Process Defects
  - Virtual Prototypes and Validation



Sheet Metal Forming - Cosmetic Defects Prediction

Virtual Prototyping Process: Reference Design to Manufactured Part to Product

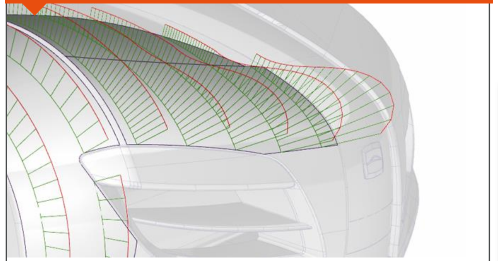
- Typical Issues in Sheet Metal Forming Manufacturing Processes:
  - Cracking, Splitting, ..
  - Springback (deviation from reference)
  - Wrinkles
  - Thinning / Thickening
- The reference design is exceedingly complex to match
  - **Material Physics:** achieving design strength
  - **Aesthetics:** matching reference design appearance expectations
- Continuous Validation against target objectives **as early** in the process **as possible** is key in reducing cost and time to market



Sheet Metal Forming - Cosmetic Defects Prediction

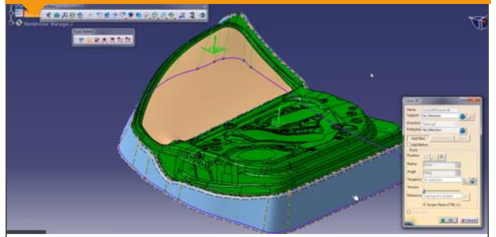
Virtual Prototyping Process: Reference Design to Manufactured Part to Product

REFERENCE CLASS A SURFACES



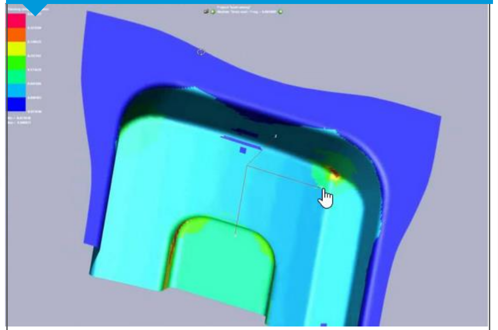
Automotive Design & Styling
e.g. Autodesk Alias

TOOL DESIGN DIE FACE, PUNCH



CAD
e.g. ESI Die Face Design Solution for Catia

SHEET METAL FORMING SIMULATION



Virtual Prototyping Process Simulation
e.g. ESI PAM-STAMP

PRODUCT ASSEMBLY

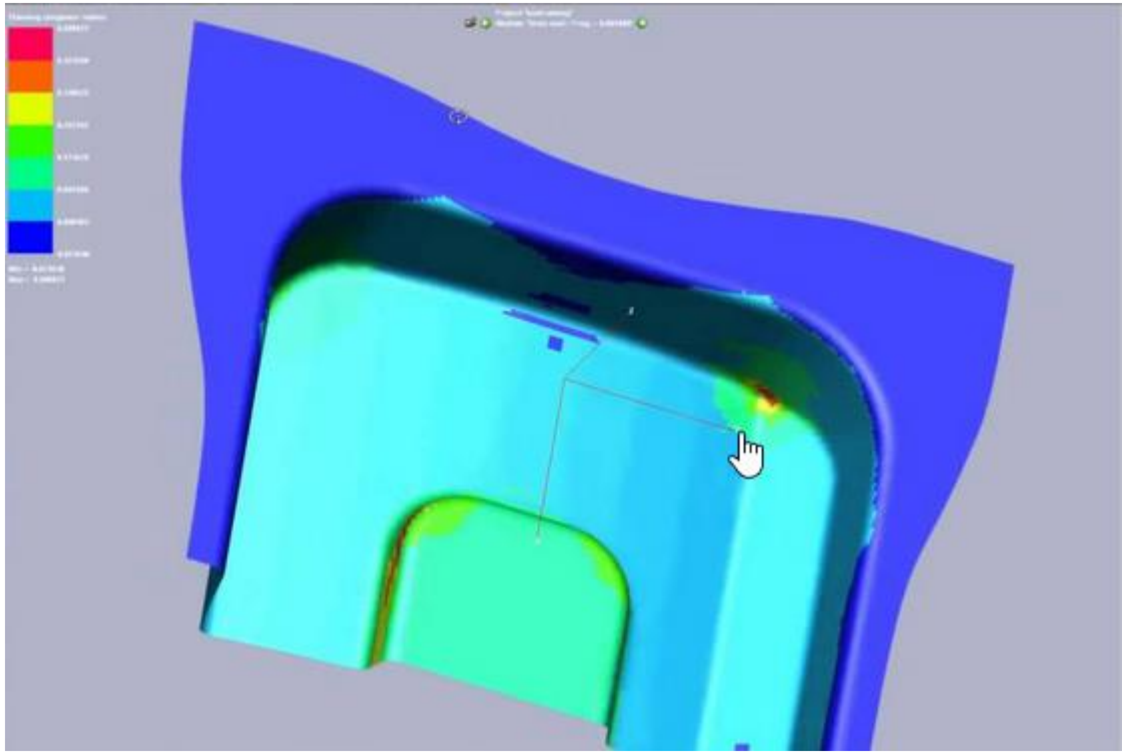
Virtual Prototyping Assembly Simulation
e.g. ESI SYSWELD

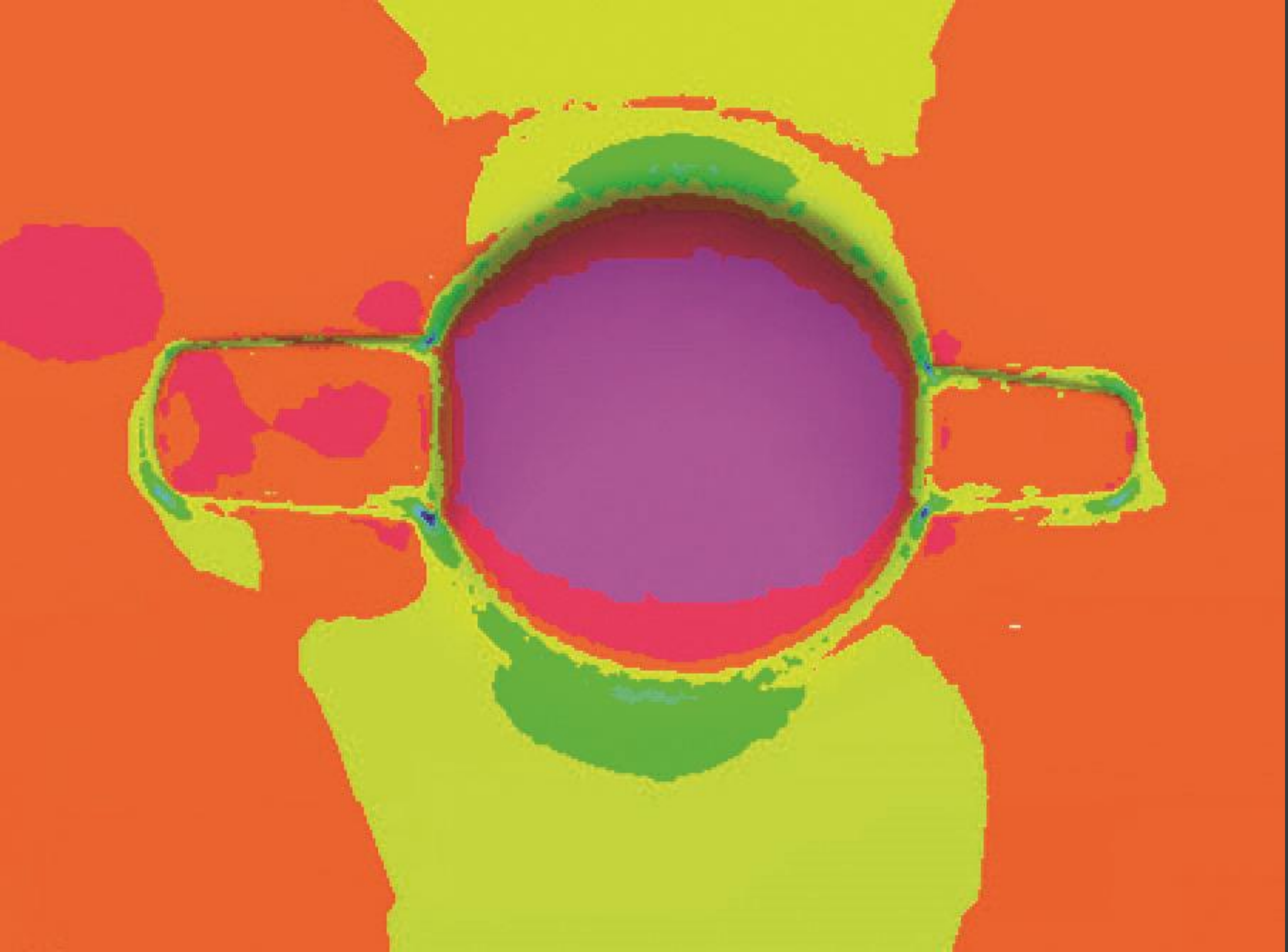
Virtual Prototyping today is the key to all but eliminate costly iterations in between design and final product.

Advanced Visualization of Stamping Defects

Sheet Metal Forming Simulation

- Numerical solver solutions such as ESI PAM-STAMP predict defects in stamped parts.
- Finite Element Analysis approaches allow highly accurate simulation of the stamping process, the die setup and quantification of relevant phenomena in manufacturing such as spring-back.
- Structural Defects such as splitting, wrinkling, cracking are clearly quantifiable against the reference surface design – no physical prototype required.





Is this a ‘good’ panel?

What about now?

GPU TECHNOLOGY CONFERENCE

Advanced Visualization of Stamping Defects

As Designed vs As Manufactured – The Quest for the Magic Contour

The Magic Contour Challenge

- Aesthetics and visual impact of remaining cosmetic defects **cannot be estimated from numerical analysis**.

- There is no general rule to automatically qualify based on numerics:
  - Acceptance criteria vary from manufacturer to manufacturer as well as from model to model.
  - The process builds on engineer’s expertise and years of anecdotal manufacturing experience

- Interpretation of the visual impact of a defect is highly subjective.

- Every further step in manufacturing – such as assembly, catophoresis, paint – will further affect the visual impact of a defect.

GPU TECHNOLOGY CONFERENCE

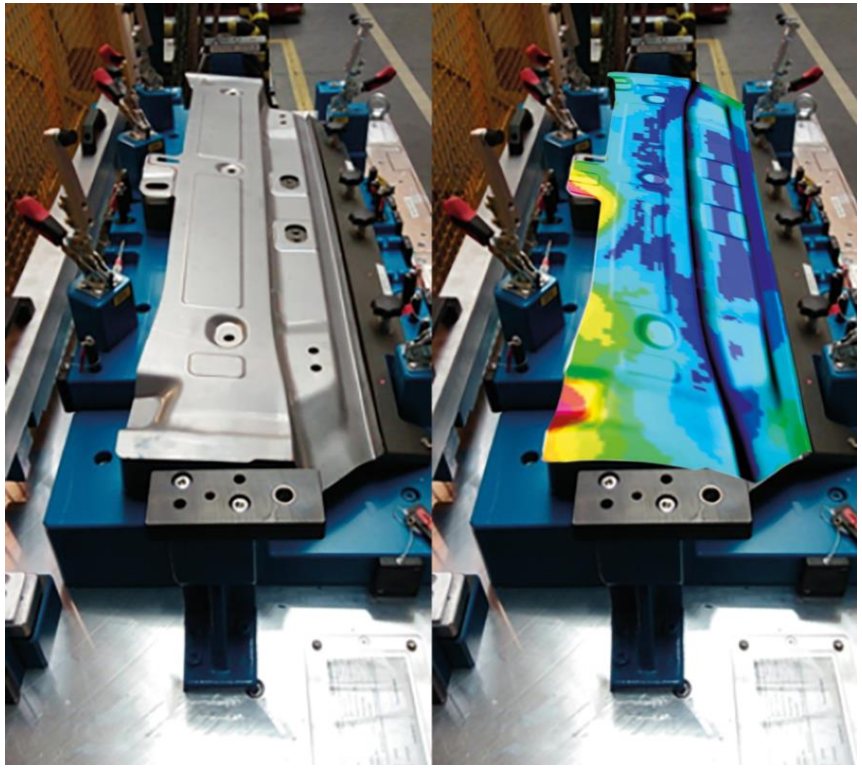
Advanced Visualization of Stamping Defects

As Designed vs As Manufactured – The Quest for the Magic Contour

- Prototype parts are verified using a manual review process with experts.
- The stamped part is removed from the die, trimmed, put up on a holder, brought to a mirror-like finish using a special formula.
- The engineer will then use special lighting and a combination of viewpoint and interaction with the part to evaluate visual defects caused by the stamping process.

Advanced Visualization of Stamping Defects

As Designed vs As Manufactured – The Quest for the Magic Contour



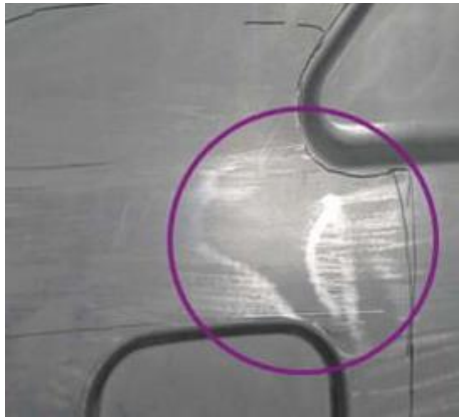
- While **structural defects** are well covered by today's Virtual Prototyping solutions, the **aesthetic impact** of how acceptable a surface defect is **remains unquantified** throughout the virtual prototyping process.
- Real World Evaluation based on physical prototypes are only possible in Try Out Phases, a time at which late **changes** to the die and process are **complex and costly**.

Advanced Visualization of Stamping Defects

As Designed vs As Manufactured – The Quest for the Magic Contour

Use Case Challenge

- Interpretation of the visual impact of a cosmetic defect is highly subjective.
- The process is not reproducible – results could vary by engineer, a lighting situation change or setup modification.
- Every further step in manufacturing – such as assembly, catophoresis, paint – will further affect the visual impact of a defect.
- Accurate impression and final verdict today are only possible using the real-life, physical part.
- No relation in between the reference **design**, the original **quotation** and the **manufactured part** – there is **no magic contour**.



Sheet Metal Forming - Cosmetic Defects Prediction

Human-Centric Quantification of Cosmetic Defects

ACCURATE VISUALIZATION

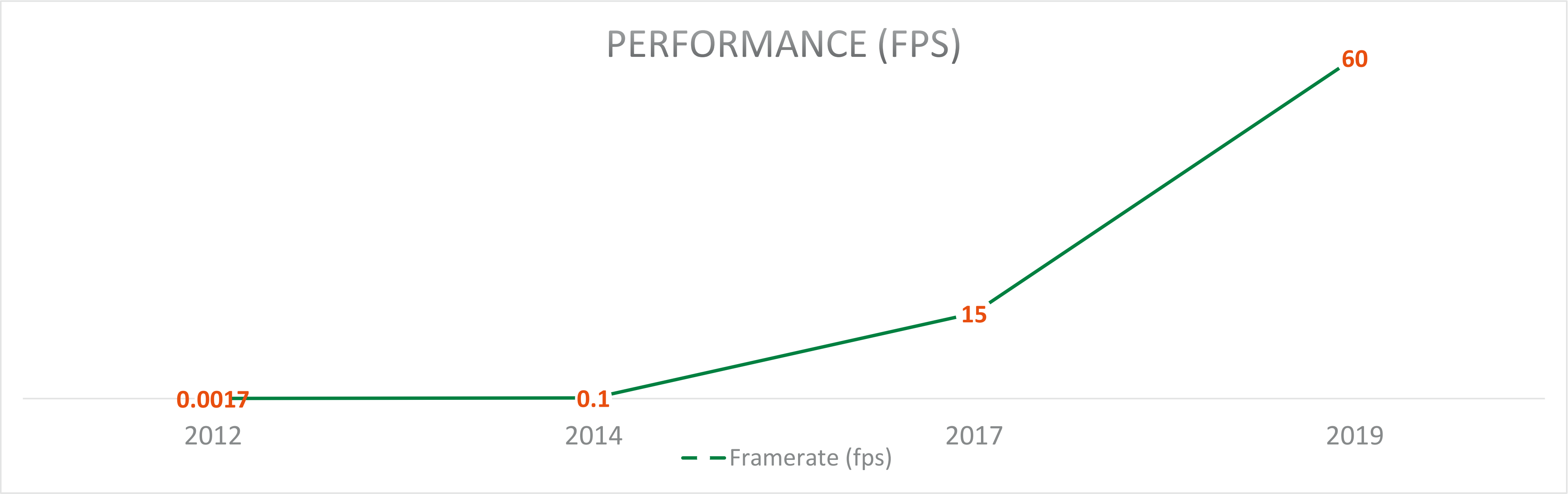
- Raytraced Reflections
- Physically-based Appearances
  - will benefit from MDL Support
  - Reproducible per assembly / manufacturing step (raw, treated, painted)
- **Performance!**

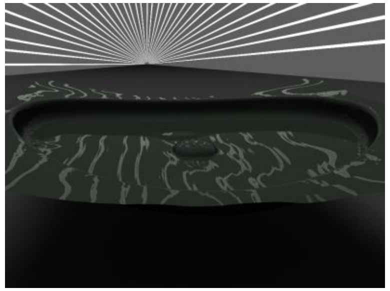
EXPERIENTIAL EVALUATION

- Low ramp-up time: Simulation to Visualization in seconds
  - Change Management on daily data
  - Reproducible Lighting and Scenario Environments
- Interactivity:
  - First person view with HMD
  - Kinematics and physics for direct manipulation

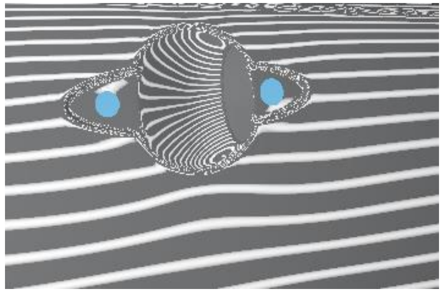
Advanced Visualization of Stamping Defects

Human-Centric Quantification of Cosmetic Defects

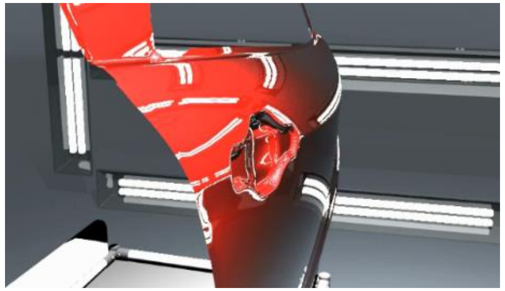




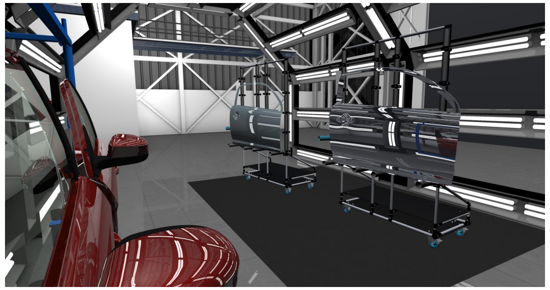
1 min
(Fermi)



1-5 sec
(Kepler)



15 fps
(Maxwell)



~40-60 fps
(Turing - Quadro RTX 8000)

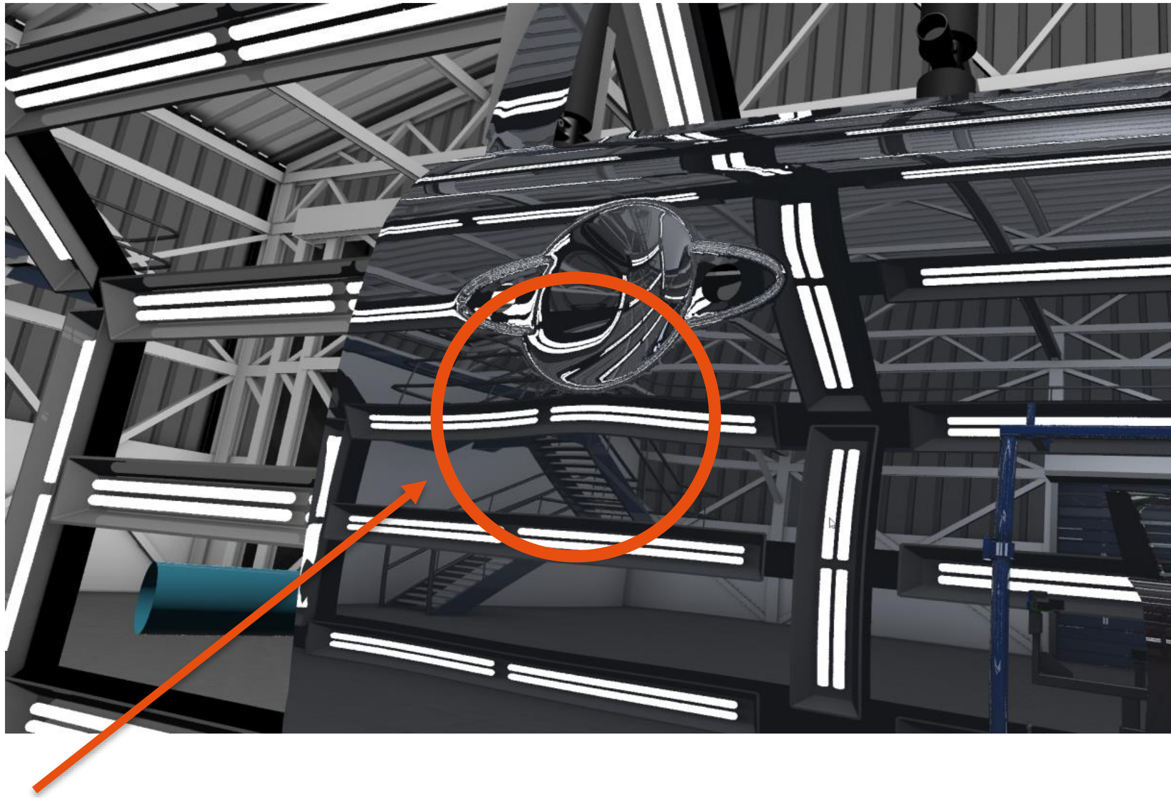
Advanced Visualization of Stamping Defects

Human-Centric Quantification of Cosmetic Defects

NVIDIA RTX technology on NVIDIA Quadro RTX series with Turing generation GPUs enables real-time ray tracing in an immersive HMD scenario for the first time.

This provides **true to life experiences** to engineers and decision makers, allowing them to predict cosmetic behaviour, discuss and decide on visual acceptance of the manufactured part versus the designed part.

Reproducible at any time during the design, testing and manufacturing planning cycles **issues** can be **resolved earlier** and with **increased confidence**.



Come see our narrated demo

VR Theater: Real-time Ray Tracing on Professional Head-Mounted Displays with NVIDIA RTX

at 5:00 PM - 06:00 PM
on Tuesday, Mar 19

VR Theater
SJCC Expo Hall 3, Concourse Level

Tuesday: 12:00pm - 6:00pm
Wednesday: 12:00pm - 6:00pm
Thursday: 11:00am - 1:00pm

QUESTIONS?

andreas.dietrich@esi-group.com

jan.wurster@esi-group.com

http://esi-group.com