Path Tracing in ParaView-OptiX: RTX for Scientific Visualization

Tim Biedert Senior Developer Technology Engineer
Mathias Hummel Senior Developer Technology Engineer

March 21, GTC San Jose 2019

Agenda

Introduction to RTX

Pathtracing in ParaView/VTK

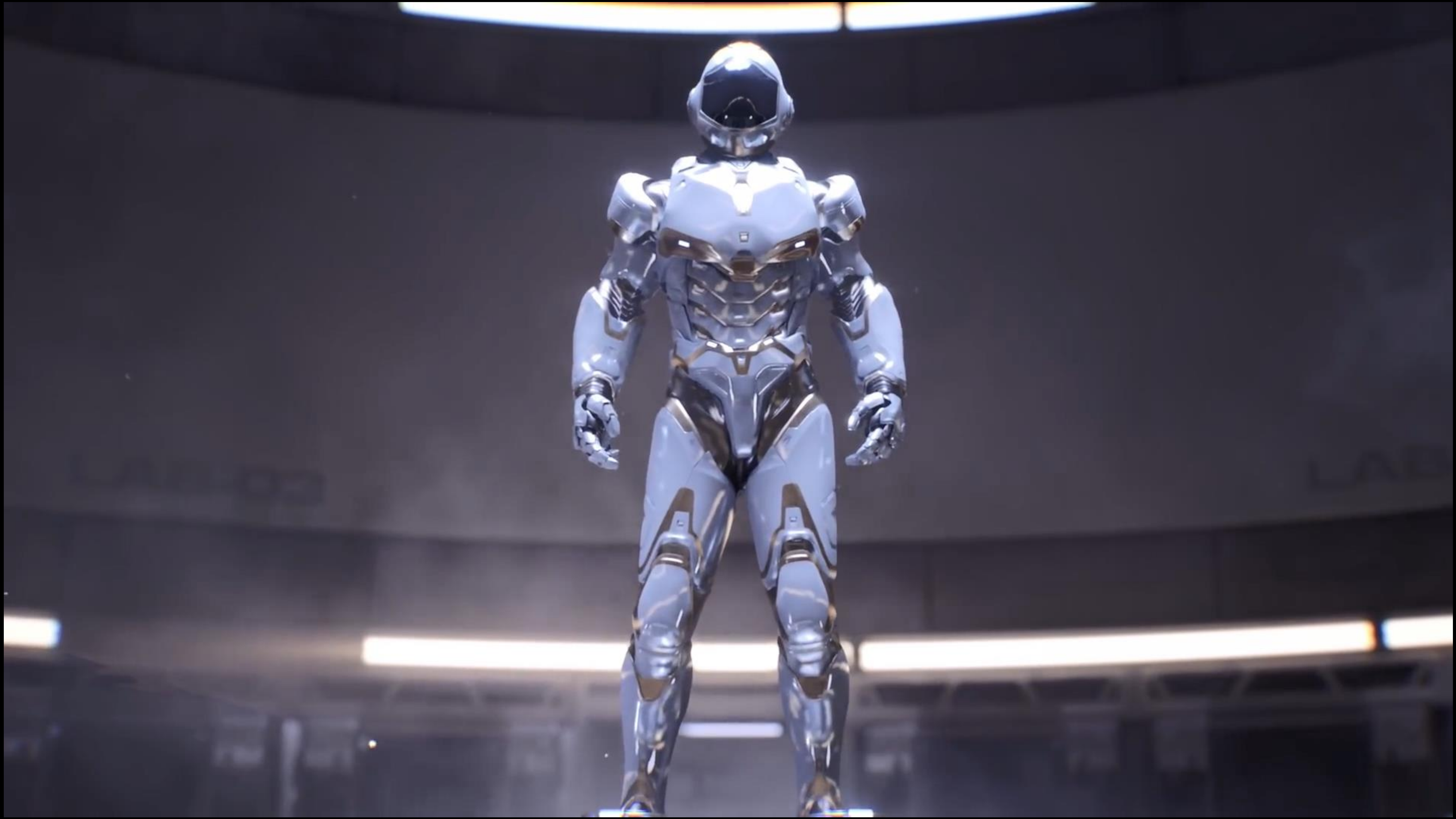
Physically-Based Materials

Denoisers

Remote Visualization

RTX ON

Unreal Engine "Reflections" Tech Demo (Star Wars) - GDC 2018



„Project Sol" Tech Demo - SIGGRAPH 2018

„The Speed of Light" Tech Demo - SIGGRAPH 2018

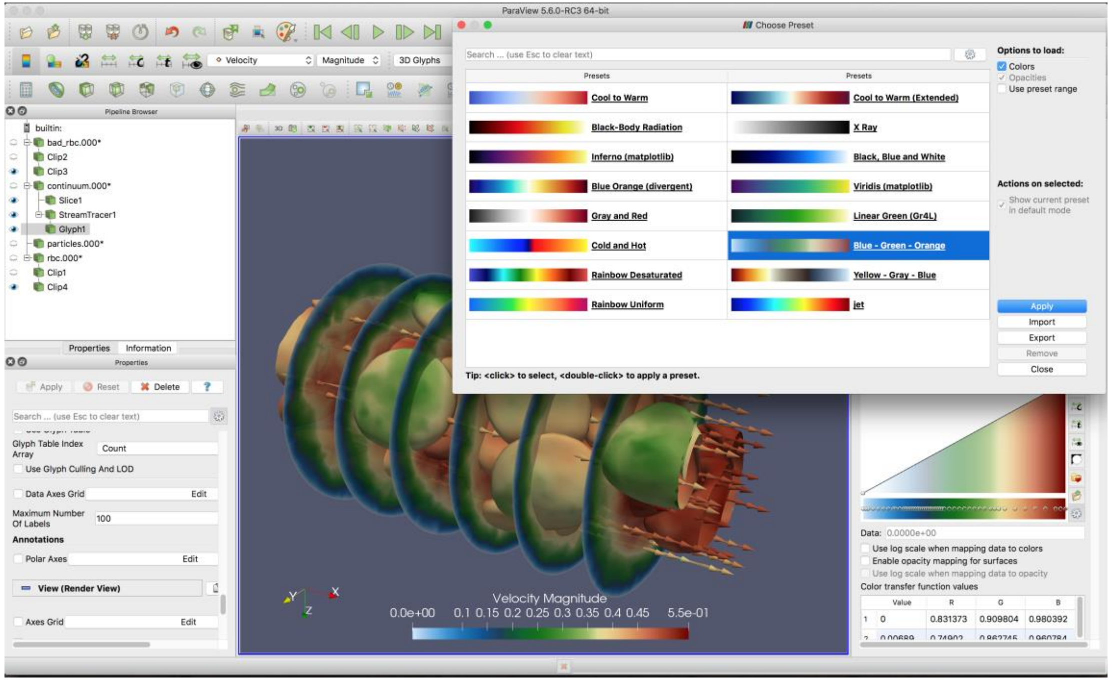
Battlefield V


GEFORCE®
RTX
METRO
EXODUS
NVIDIA®

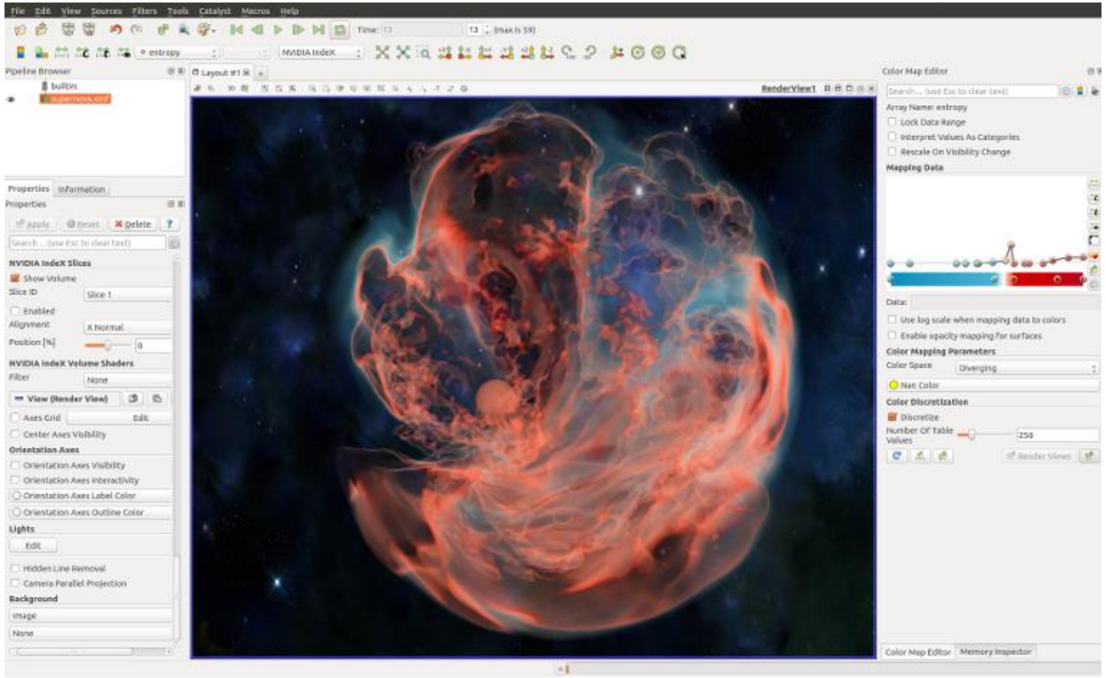
RAYTRACING IN PARAVIEW

KITWARE PARAVIEW

Open-Source (Distributed) Visualization Package



OpenGL



NVIDIA IndeX Plugin

ParaView

VTK: VISUALIZATION TOOLKIT

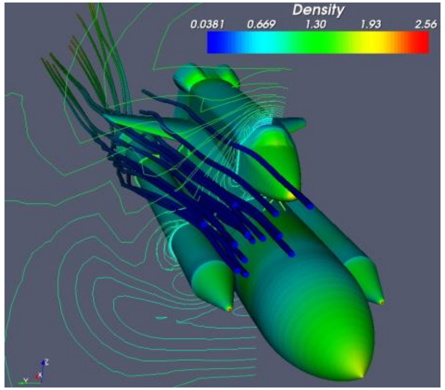
Open Source Scientific Visualization Toolbox

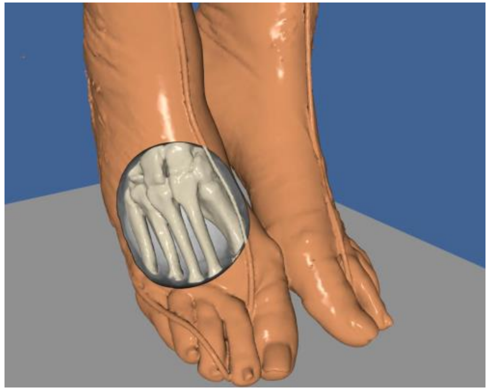
Process data using pipelines made up of filters

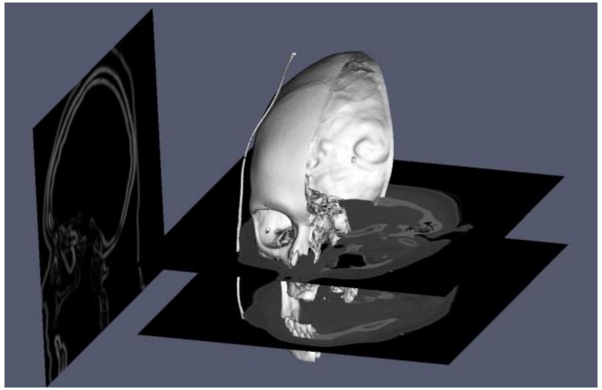
Forms the foundation of ParaView

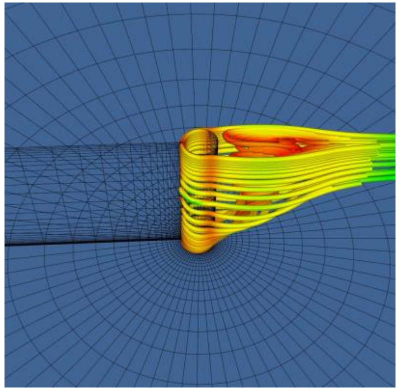
OpenGL

Software raytracing

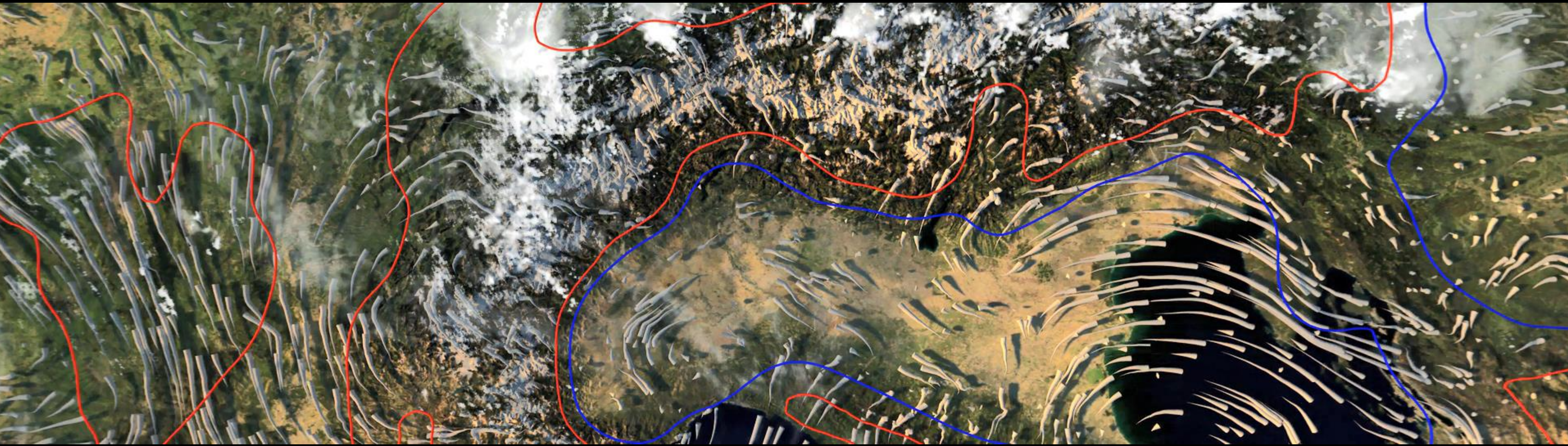








ParaView „Urban Airflow" Tech Demo - SIGGRAPH 2018



ParaView „Weather Simulation in a Box" Tech Demo – SC 2018

VISRTX + MDL

VISRTX

Visualization Framework Powered by NVIDIA RTX Technology

Progressive forward pathtracer with NEE/MIS

Hardware-acceleration through OptiX

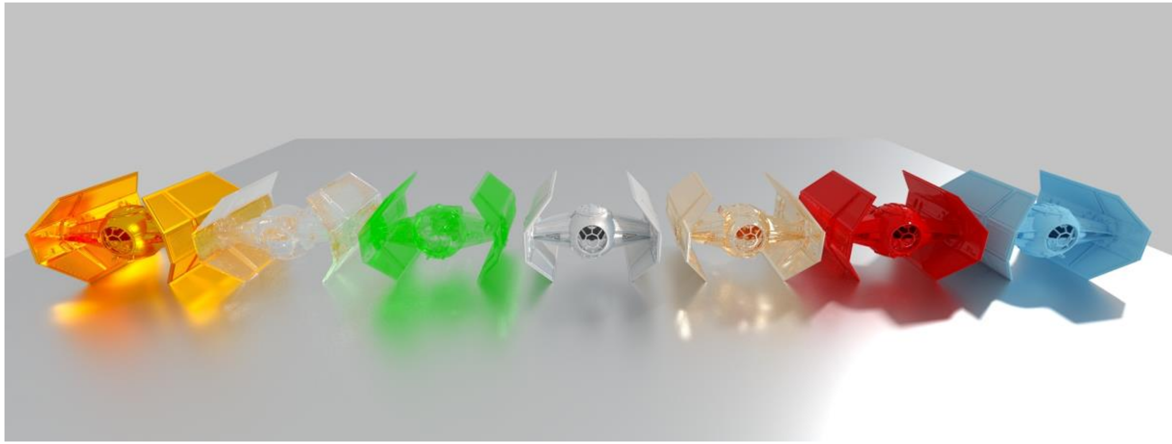
MDL for physically-based materials

AI denoiser

Area lights, Depth of Field, Tone mapping, etc.

Open-source C++ library

Feedback welcome (issues, PRs, e-mail)!



https://github.com/NVIDIA/VisRTX

```cpp
#define VISRTX_DYNLOAD
#include <VisRTX.h>

int main(int argc, char **argv)
{
    // Load shared VisRTX library
    if (!VisRTX_LoadLibrary())
        return 1;

    // Get factory instance
    VisRTX::Context* ctx = VisRTX_GetContext();

    // Basic material
    VisRTX::BasicMaterial* basic = ctx->CreateBasicMaterial();
    basic->Diffuse(VisRTX::Vec3f(1.0f, 0.0f, 0.0f));
    basic->SetSpecular(VisRTX::Vec3f(1.0f, 1.0f, 1.0f));
    basic->SetShininess(10.0f);
    basic->SetEmissive(VisRTX::Vec3f(0.0f, 0.0f, 1.0f));
    basic->SetLuminosity(5.0f);
    basic->SetOpacity(0.5f);

    // Textures
    VisRTX::Texture* texture = ctx->CreateTexture(VisRTX::Vec2ui(512, 512), VisRTX::TextureFormat::RGBA8, texels);
    texture->SetFiltering(...)
    texture->SetWrapMode(...)
    texture->SetMaxAnisotropy(...)

    basic->SetSpecularTexture(...)
    basic->SetShininessTexture(...)
    basic->SetEmissiveTexture(...)
    basic->SetOpacityTexture(...)
    basic->SetBumpMapTexture(...)
```

```cpp
// MDL material
VisRTX::MDLMaterial* mdl = ctx->CreateMDLMaterial("::Materials::Metal", source.c_str(), source.size(), 0, nullptr,
                                                  VisRTX::CompilationType::INSTANCE);
mdl->SetParameterFloat("roughness", 0.2f);
mdl->Compile(); // Instance compilation only

// Query available parameters
for (uint32_t i = 0; i < mdl->GetParameterCount(); ++i)
{
    const char* name = mdl->GetParameterName(i);
    VisRTX::ParameterType type = mdl->GetParameterType(name);

    if (type == VisRTX::ParameterType::COLOR)
        VisRTX::Vec3f defaultValue = mdl->GetParameterColor(name);
    // ...
}

// Geometry
VisRTX::TriangleGeometry* triangles = ctx->CreateTriangleGeometry(3, indices, 6, vertices, normals);
triangles->SetMaterial(mdl);
// triangles->SetTexCoords(...);

VisRTX::SphereGeometry* spheres = ctx->CreateSphereGeometry(2, centers, radii);
spheres->SetMaterial(basic);
// spheres->SetMaterials(...)

VisRTX::Model* model = ctx->CreateModel();
model->AddGeometry(triangles);
model->AddGeometry(spheres);
```

```
// Lights
VisRTX::DirectionalLight* light = ctx->CreateDirectionalLight();
light->SetDirection(VisRTX::Vec3f(-1.0f, -1.0f, -1.0f));
light->SetColor(VisRTX::Vec3f(1.0f, 0.0f, 0.0f));
light->SetIntensity(0.7);
light->SetAngularDiameter(2.0f);
light->SetVisible(true);

// Camera
VisRTX::PerspectiveCamera* camera = ctx->CreatePerspectiveCamera();
camera->SetPosition(VisRTX::Vec3f(0.0f, 0.0f, 5.0f));
camera->SetDirection(VisRTX::Vec3f(0.0f, 0.0f, -1.0f));
camera->SetAspect(width / (float)height);
camera->SetFocalDistance(5.0f);
camera->SetApertureRadius(0.1f);

// Renderer
VisRTX::Renderer* renderer = ctx->CreateRenderer();
renderer->SetNumBounces(2, 8);
renderer->SetSampleAllLights(true);
renderer->SetDenoiser(VisRTX::DenoiserType::AI);
// renderer->SetSamplesPerPixel(1);
// renderer->SetToneMapping(..)
// renderer->SetFireflyClamping(..)
renderer->SetCamera(camera);
renderer->SetModel(model);
renderer->AddLight(light);

// Framebuffer
VisRTX::FrameBuffer* frameBuffer = ctx->CreateFrameBuffer(VisRTX::FrameBufferFormat::RGBA32F);
```


NVIDIA.

```
while (!done)
{
    // ... app logic

    // Resize framebuffer (if necessary)
    frameBuffer->Resize(VisRTX::Vec2ui(width, height));

    // Reset progressive rendering (if necessary)
    if (sizeChanged || interacted)
        frameBuffer->Clear();

    // Render
    renderer->Render(frameBuffer);

    // Display in OpenGL
    glBindFramebuffer(GL_DRAW_FRAMEBUFFER, 0);
    glUseProgram(fullscreenQuadProgram);
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D, frameBuffer->GetColorTextureGL());
    glUniform1i(fullscreenTextureLocation, 0);
    glBindVertexArray(fullscreenVAO);
    glDrawArrays(GL_POINTS, 0, 1);
}

// Clean up
renderer->Release();
frameBuffer->Release();
// etc.
```

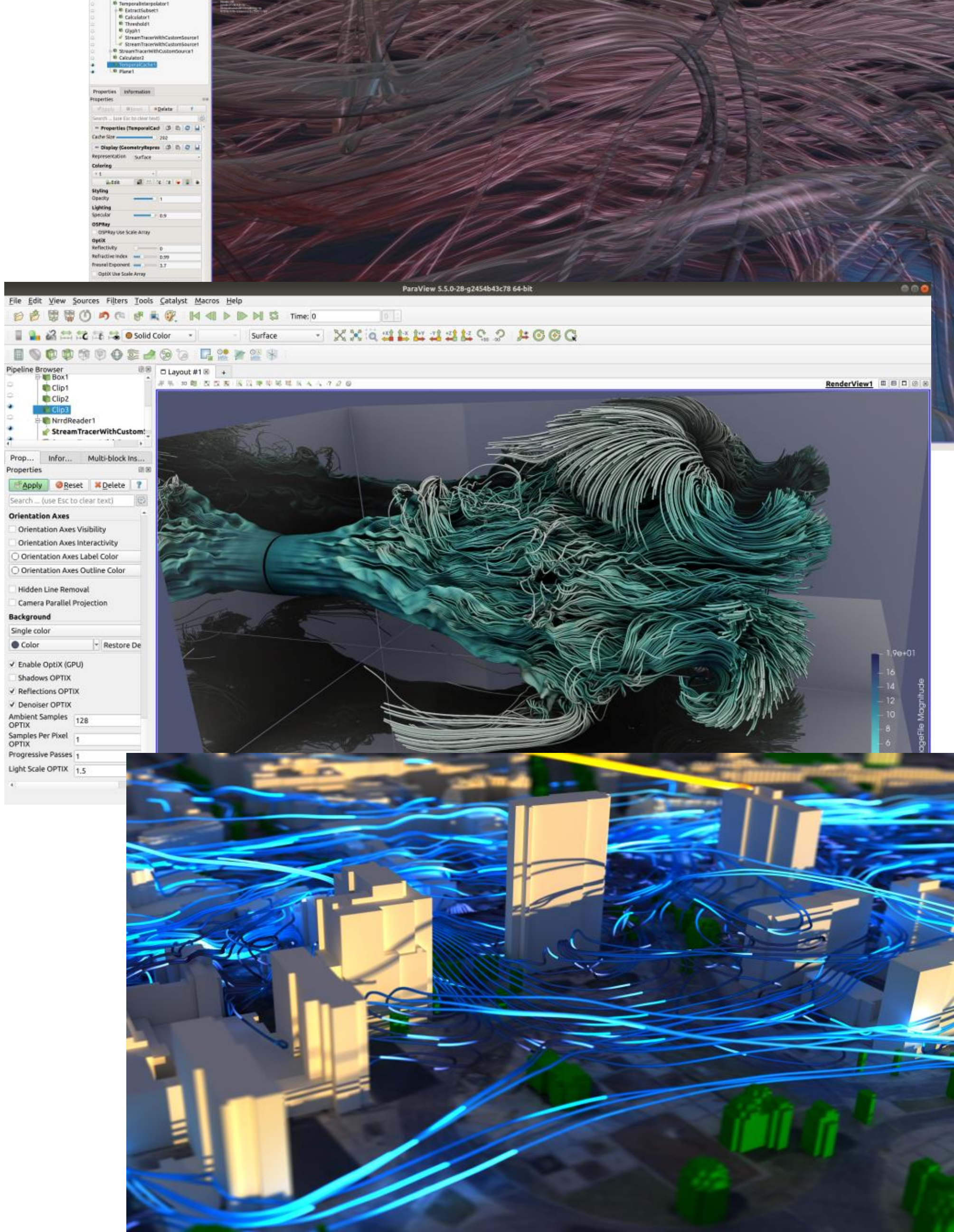
VISRTX + PARAVIEW

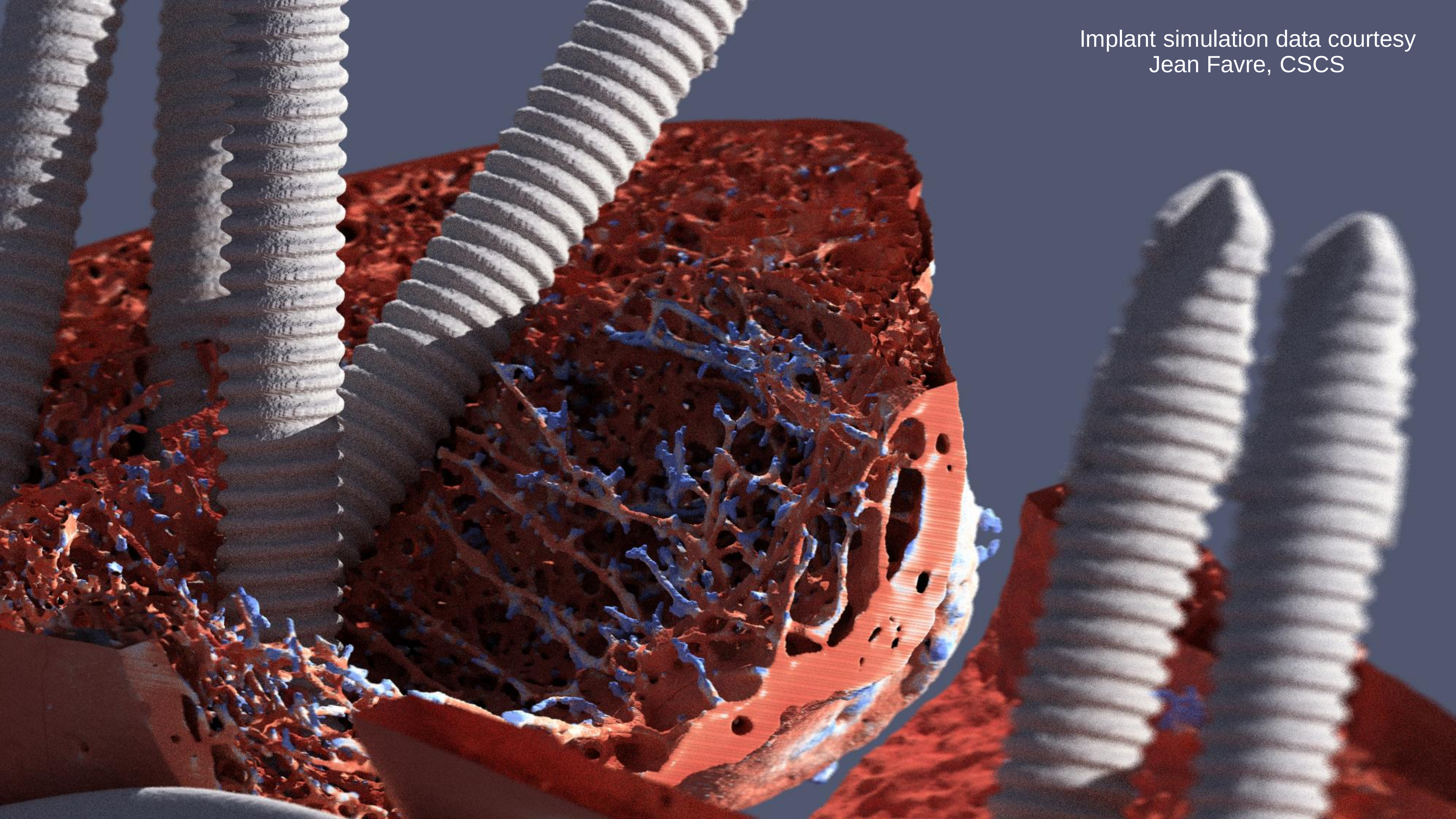
VisRTX open-source on GitHub

Shipped with upcoming ParaView release

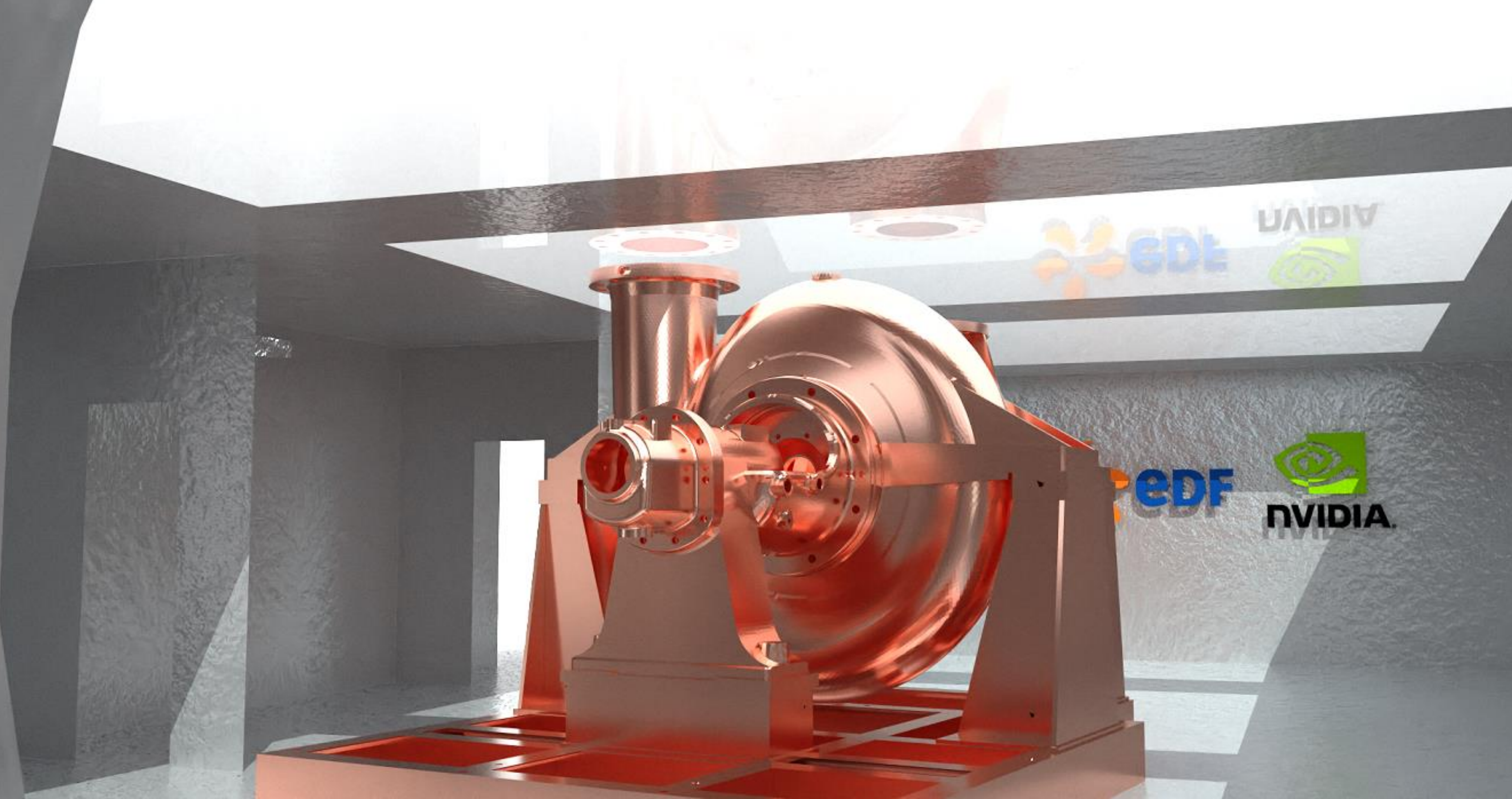
- **No additional steps necessary!**

Kitware NVIDIA

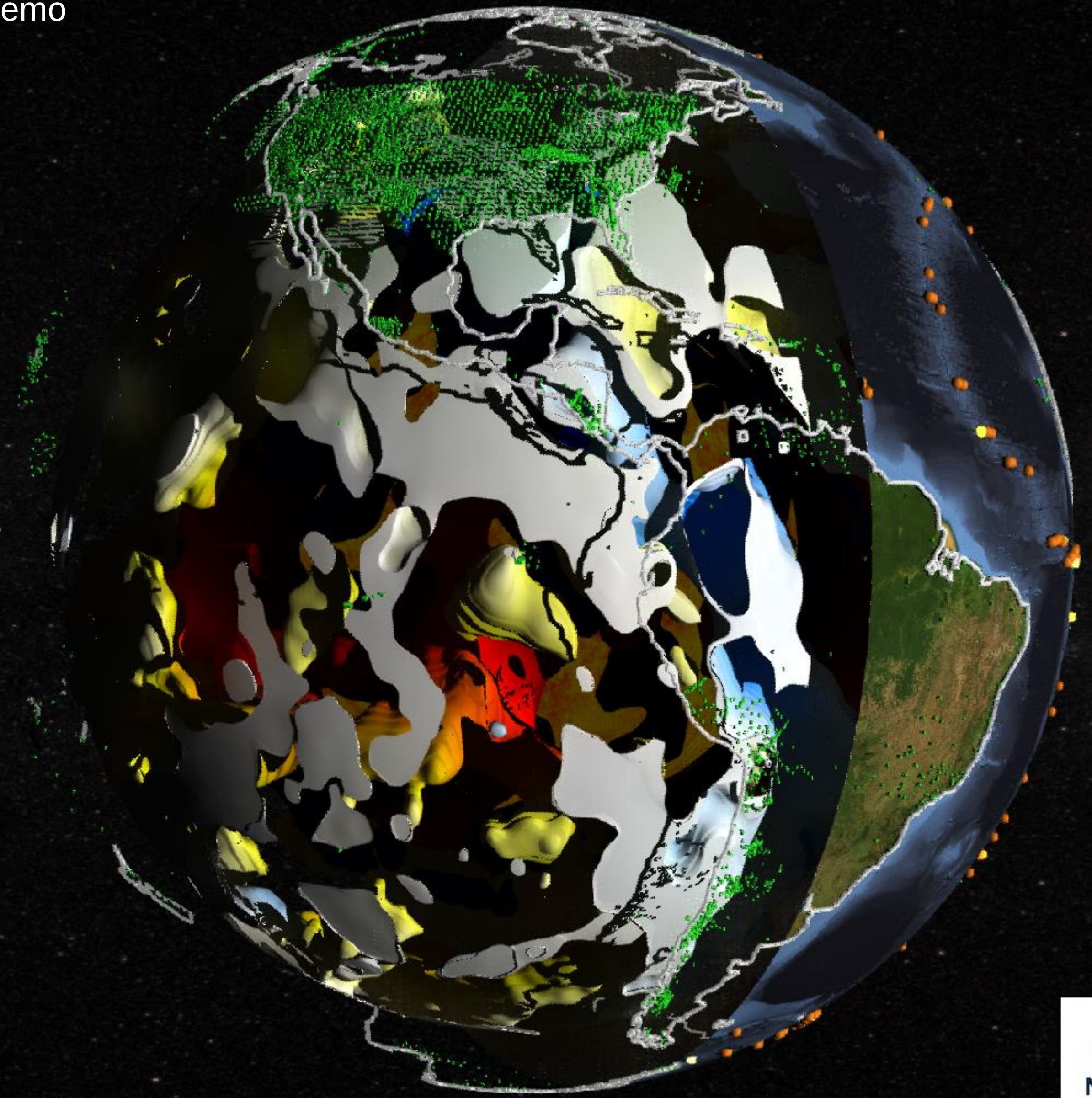



Implant simulation data courtesy
Jean Favre, CSCS

Data courtesy of Électricité de France / Kitware SAS

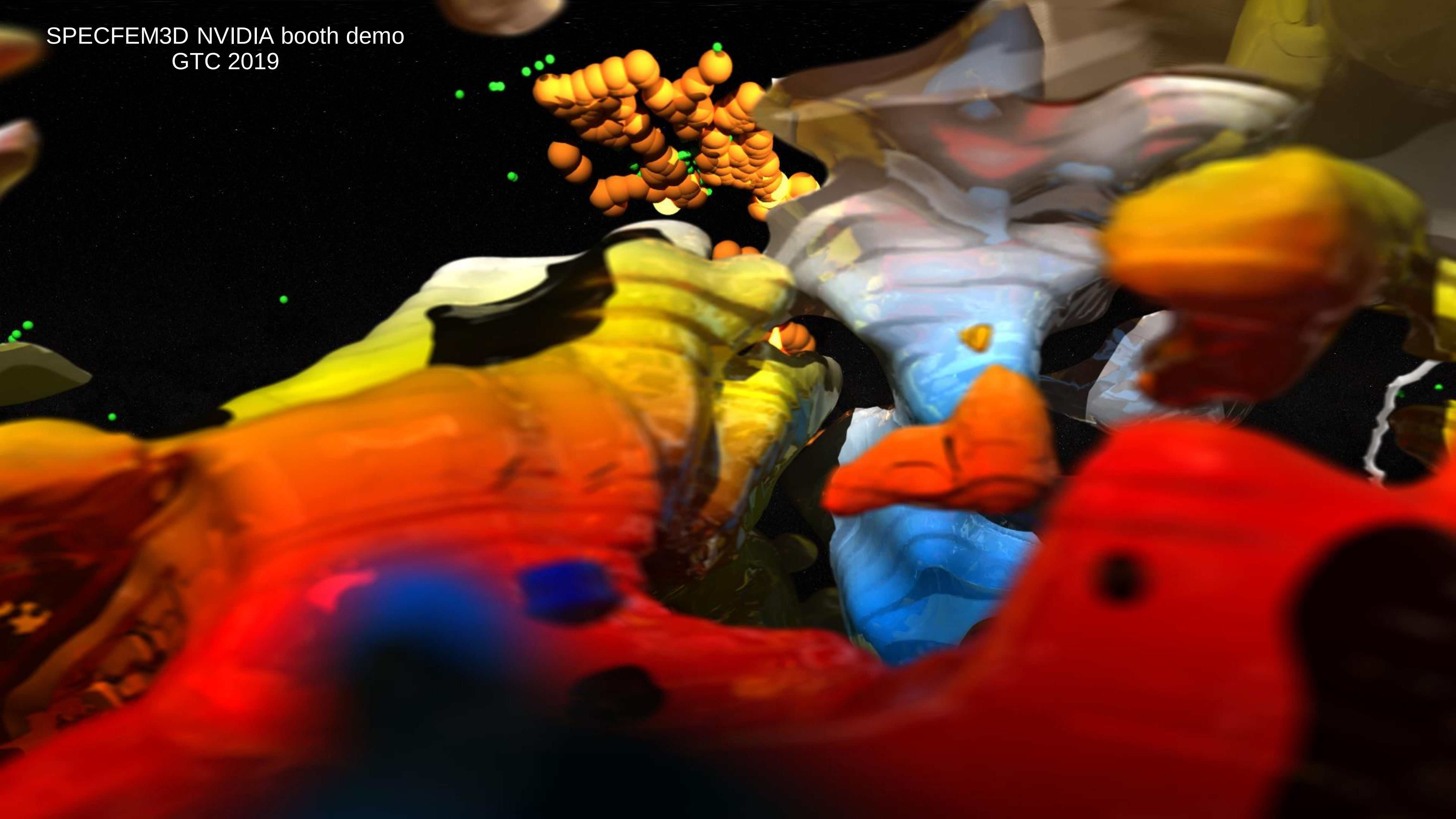


SPECFEM3D NVIDIA booth demo
GTC 2019

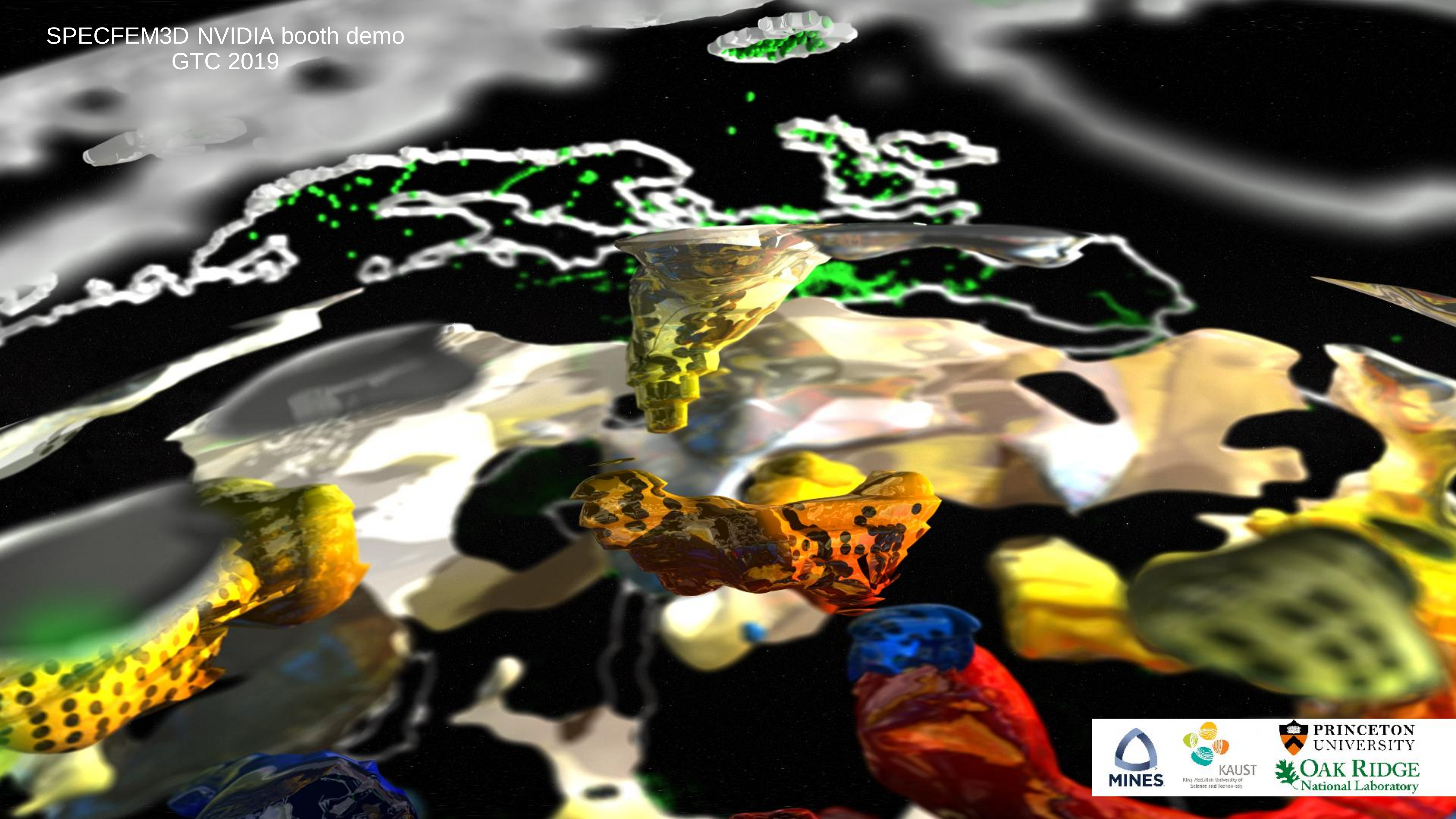


MINES
KAUST
King Abdullah University of Science and Technology
PRINCETON UNIVERSITY
OAK RIDGE National Laboratory

SPECFEM3D NVIDIA booth demo
GTC 2019



SPECFEM3D NVIDIA booth demo
GTC 2019



MINES
KAUST
King Abdullah University of Science and Technology
PRINCETON UNIVERSITY
OAK RIDGE National Laboratory

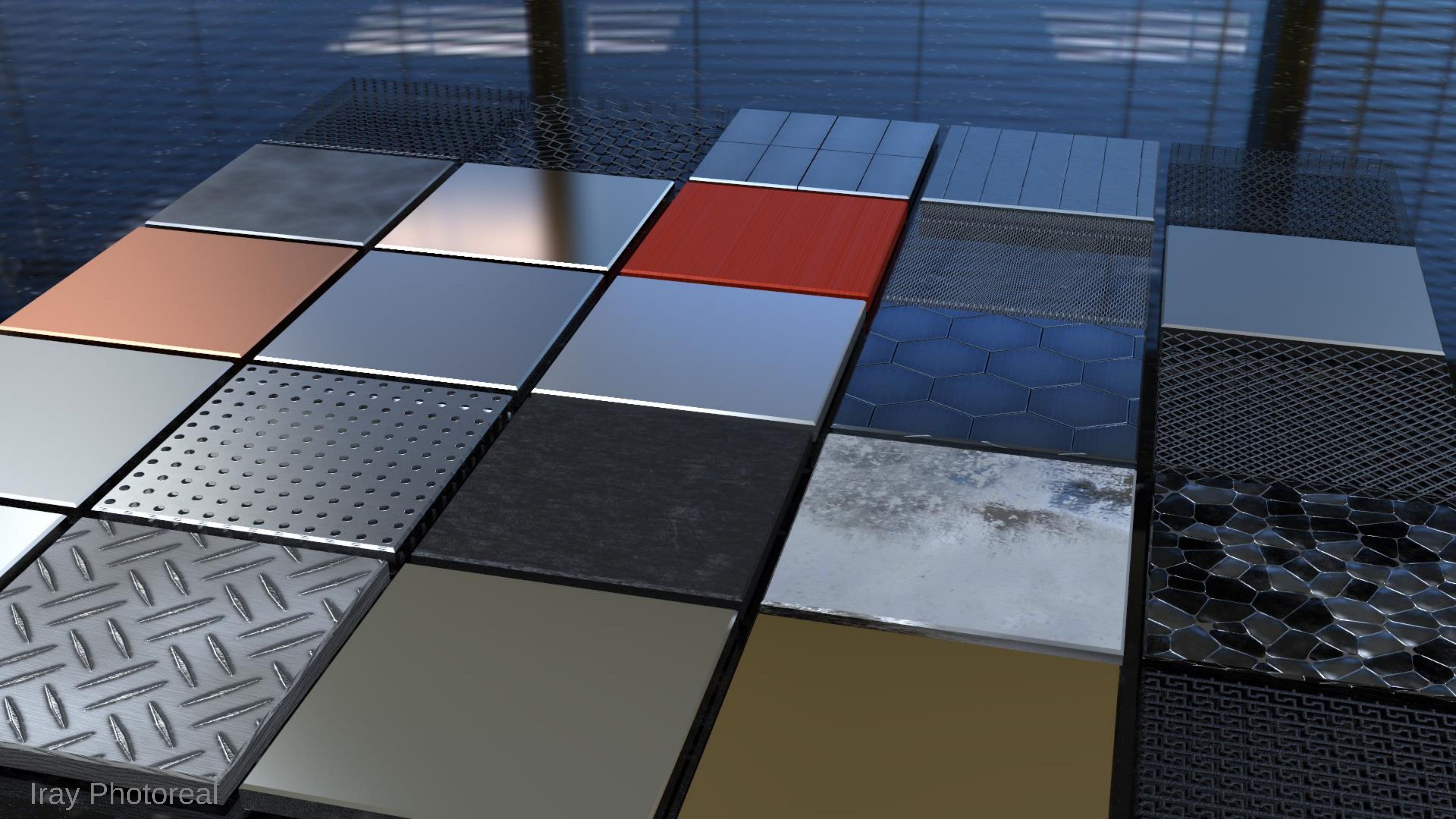
NVIDIA MDL

The NVIDIA Material Definition Language (MDL)

is technology developed by NVIDIA

to define physically-based materials

for physically-based rendering solutions.


Iray Photoreal


Iray Photoreal

Iray Photoreal

courtesy Harley Davidson

Photographer: Simon Puschmann/Rendering and Retouching:[zerone]

vMaterials

Free Catalog of Real-World Materials

Described in **MDL**

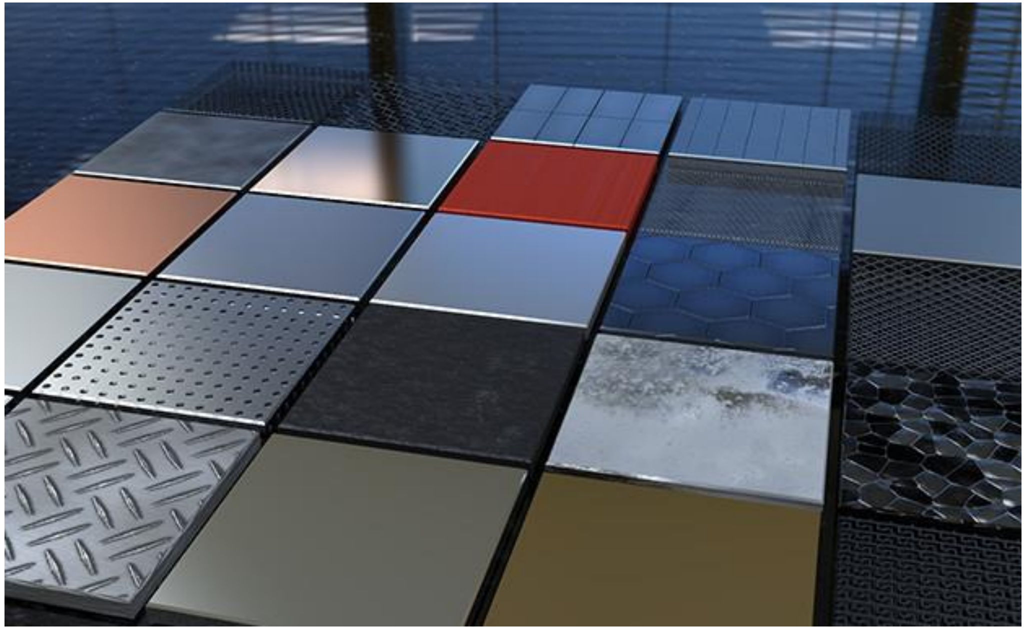
Designed and **verified** by NVIDIA material specialists

Can be used **as-is** …

… or **modified** and **layered** to create custom materials

Example: Dust layer -> Scratch layer -> Metal layer

https://developer.nvidia.com/vmaterials



NVIDIA vMaterials with Iray Photoreal

Matching the Appearance of a Single Material Within Different Rendering Techniques

One Scene for Different Renderers

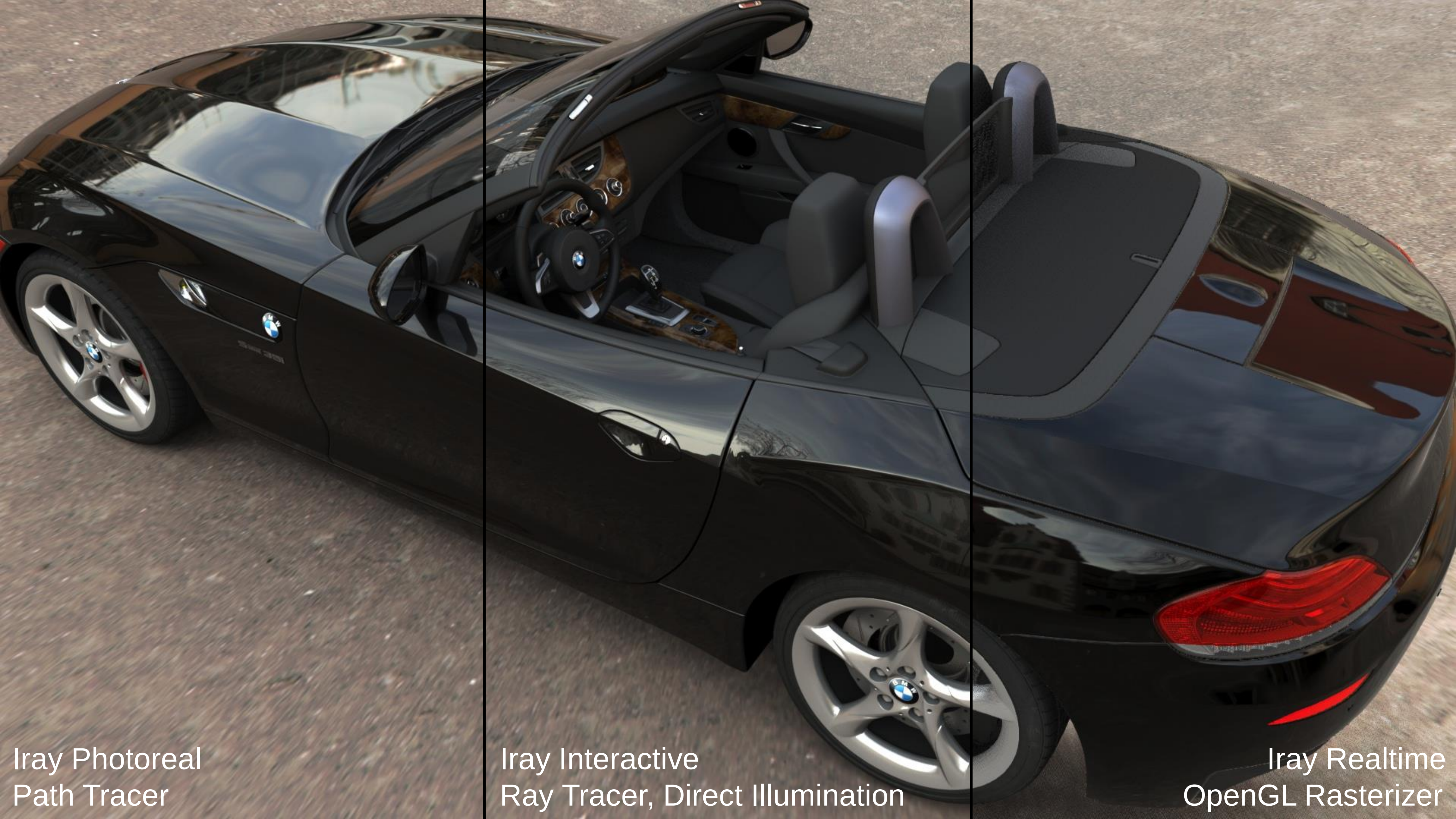
Realtime Rasterizer

Interactive Raytracer

Pathtracer

Share scene and MDL materials for a consistent look

Switching renderers with no scene modifications


Iray Photoreal
Path Tracer
Iray Interactive
Ray Tracer, Direct Illumination
Iray Realtime
OpenGL Rasterizer

DENOISERS / REALTIME RAYTRACING

OPTIX AI DENOISER

Recurrent Denoising Autoencoder

GPU-accelerated artificial intelligence approach

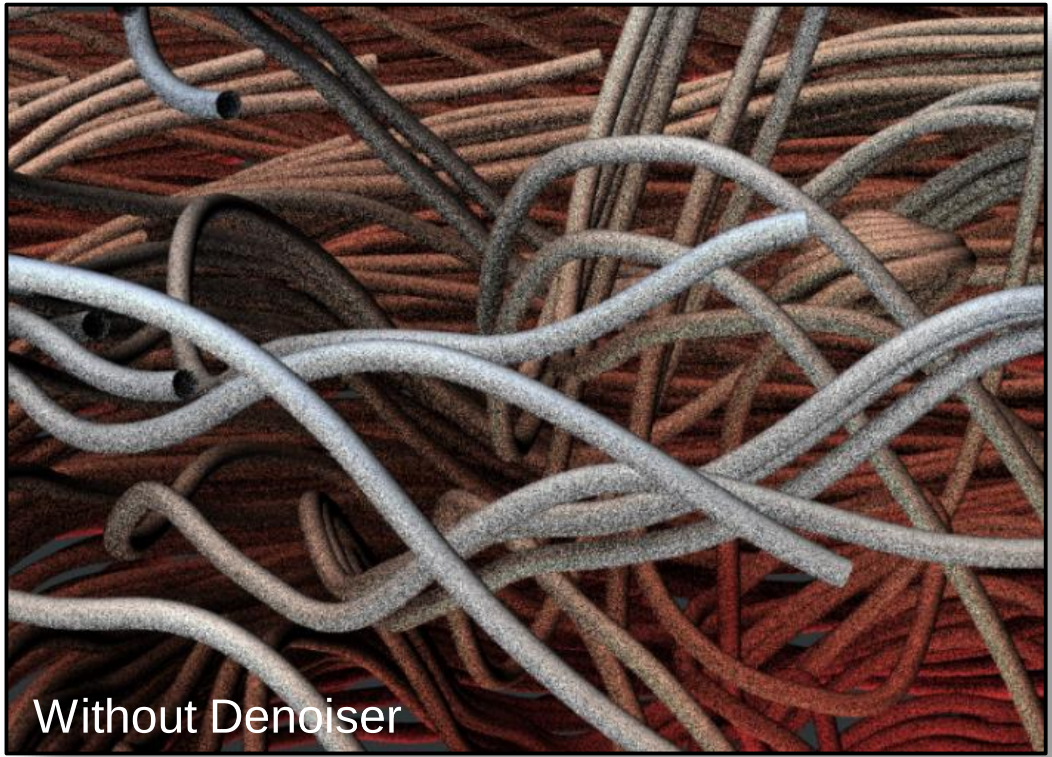
Ships with OptiX

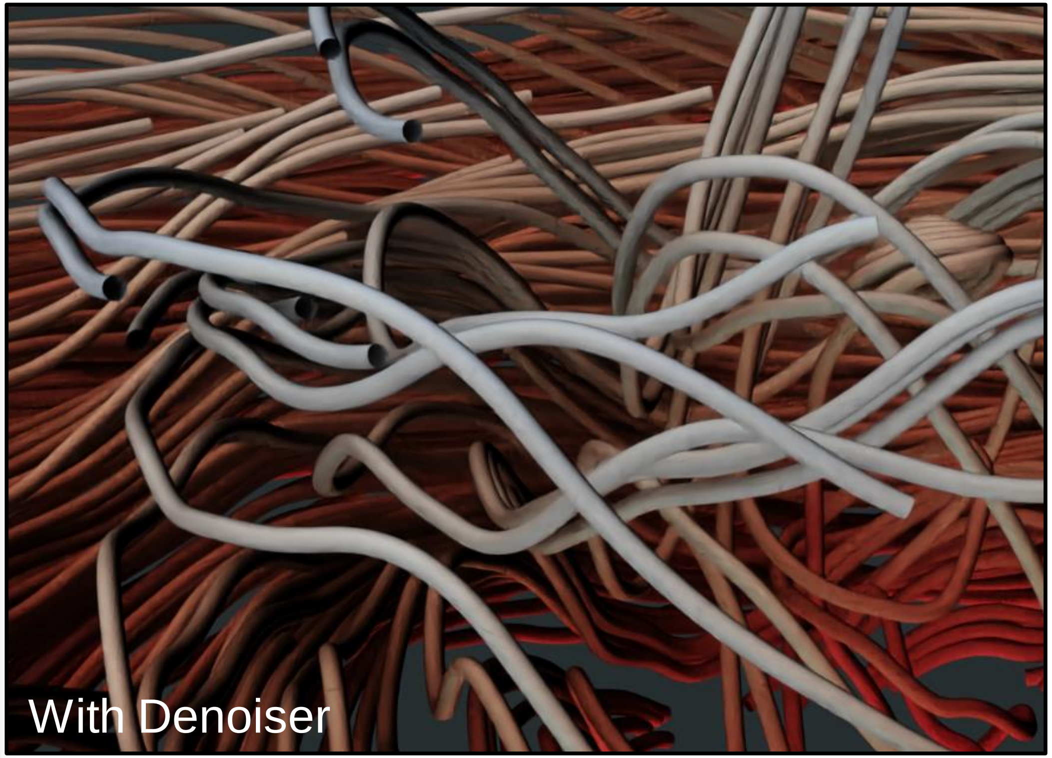
Last-frame denoiser after 10+ samples

Publication:
Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder - SIGGRAPH 2017

OPTIX AI DENOISER

In VisRTX / ParaView





FUN IMAGE ON TWITTER

NOISE IN RAY TRACING RENDERING

Where do the fireflies come from?

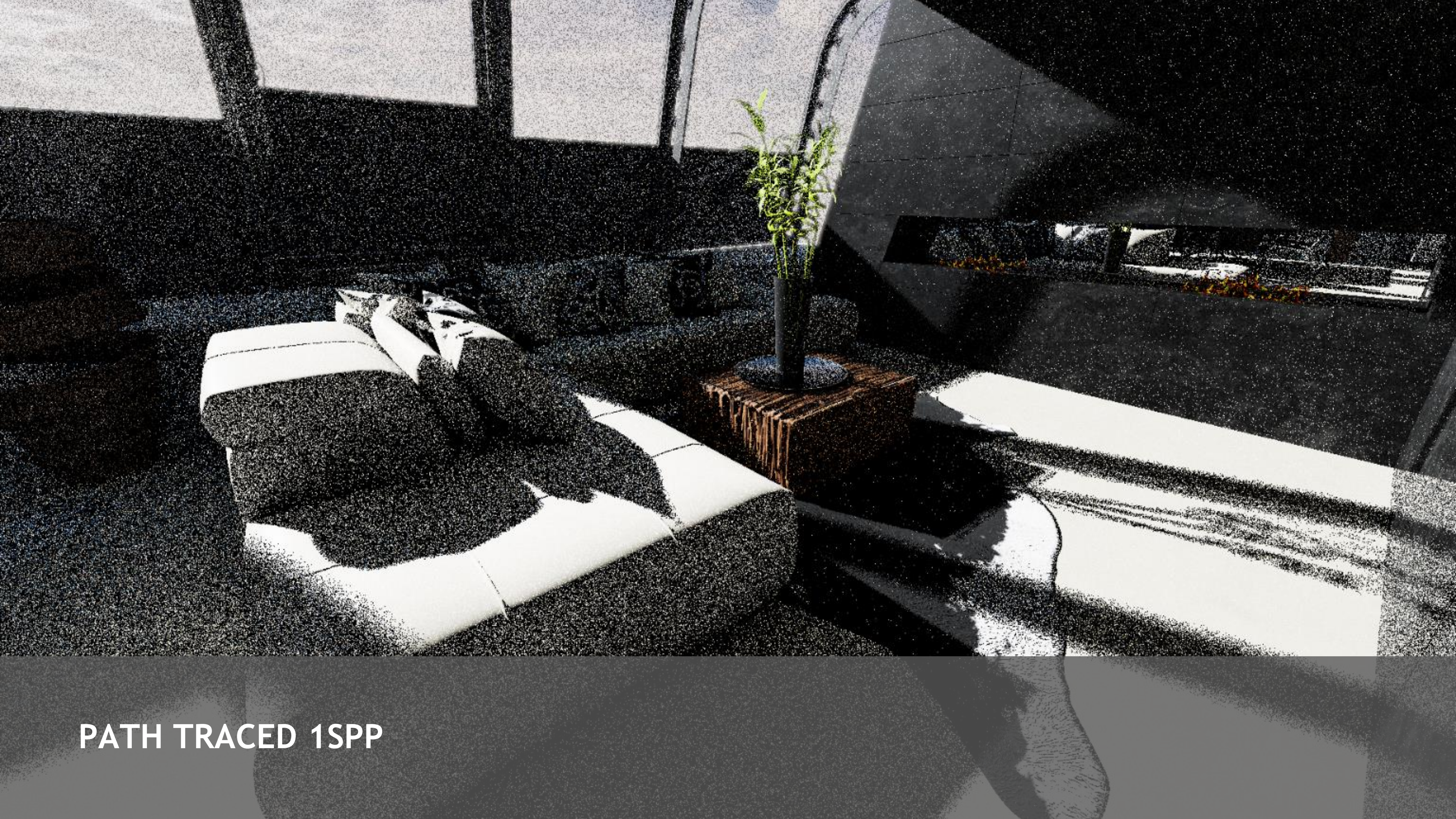
The rendering equation is solved with Monte Carlo sampling

$$L(\omega_o) = \int_{\delta} L(\omega_i) f(\omega_o, \omega_i) |\omega_i \cdot n| d\omega_i \approx \sum_{i=0}^{n} L(\omega_i) f(\omega_o, \omega_i) |\omega_i \cdot n| / p(\omega_i)$$

Every term in the estimator is a complicated function over the hemisphere

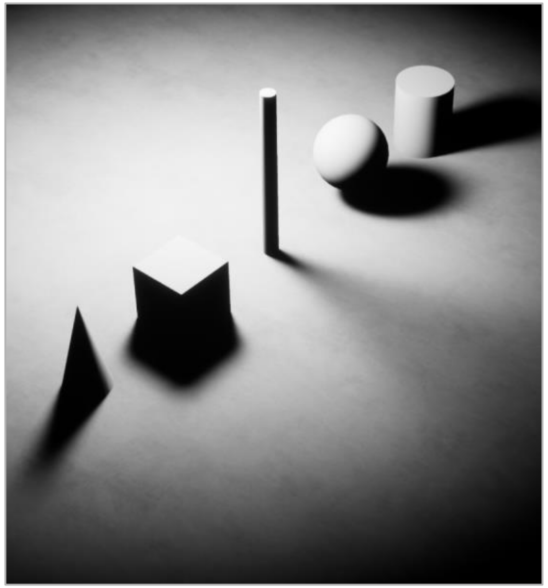
- Incoming radiance, visibility, BRDF, and sampling Pdf

Insufficient sampling leads to high variance in the estimator



PATH TRACED 1SPP

RAY TRACING WITH 1SPP (OR LESS)



Shadows

Reflections & Specular

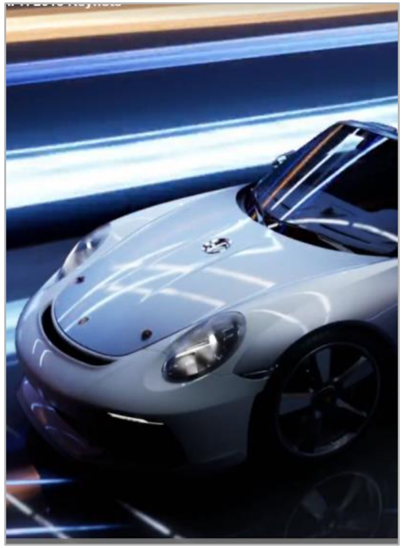
Ambient Occlusion

Global Illumination

USED IN MULTIPLE DEMOS

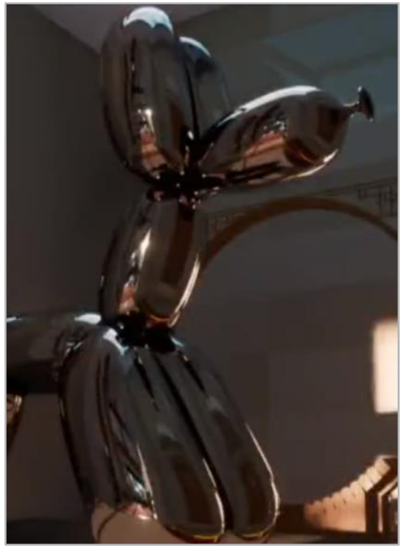
Star Wars
Reflections

RTX Demo



Porsche 70 Trailer

SOL



Rosewood Bangkok

GROUND TRUTH

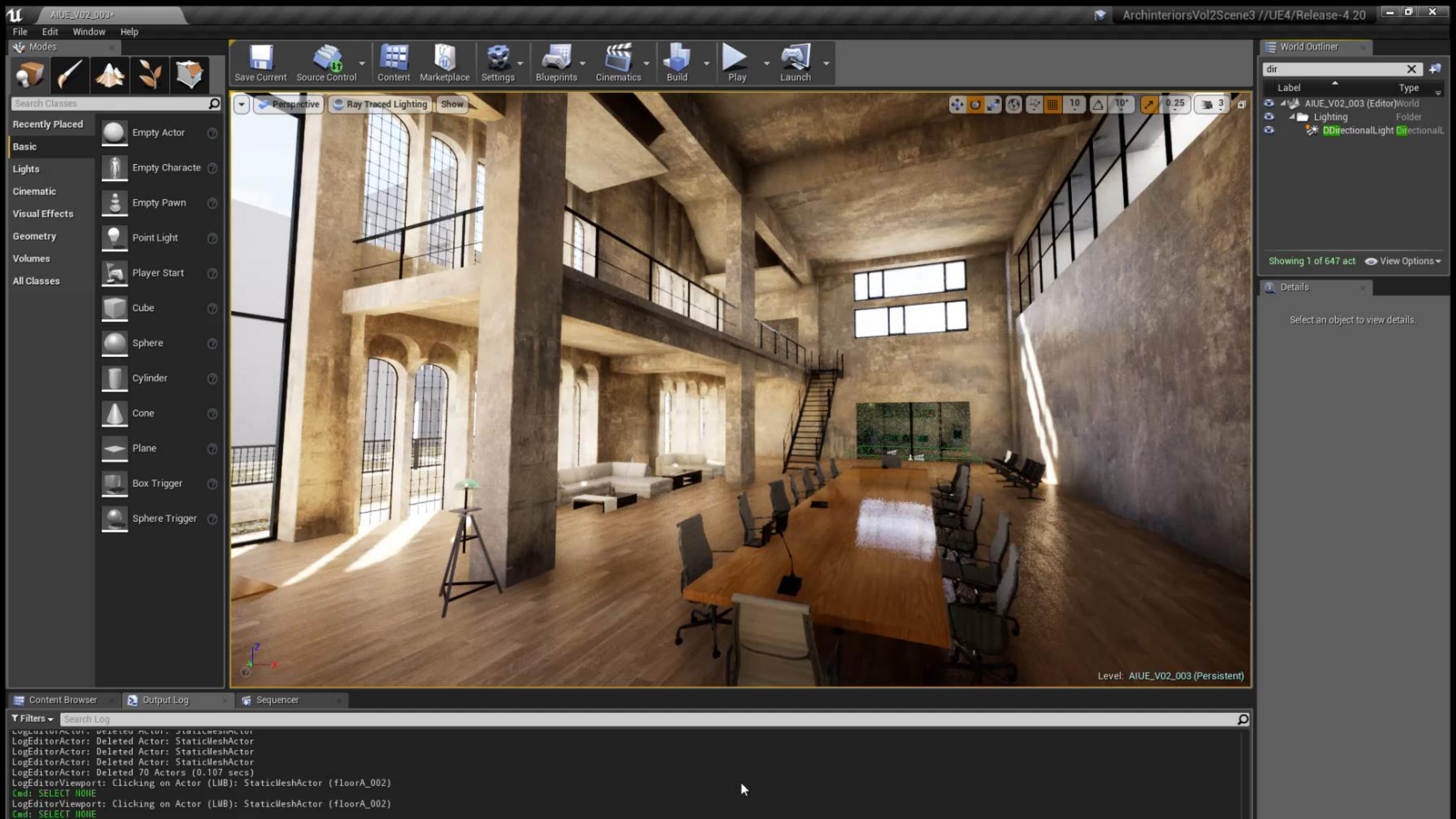
1SPP RAY TRACED REFLECTIONS

1SPP RAY TRACED REFLECTIONS + DENOISING

Ground Truth

1spp Ray Traced Global Illumination

1spp Ray Traced Global Illumination + Denoising



AIUE_V02_003*
ArchinteriorsVol2Scene3 //UE4/Release-4.20
File
Edit
Window
Help
Modes
Search Classes
Recently Placed
Basic
Lights
Cinematic
Visual Effects
Geometry
Volumes
All Classes
Empty Actor
Empty Characte
Empty Pawn
Point Light
Player Start
Cube
Sphere
Cylinder
Cone
Plane
Box Trigger
Sphere Trigger
Save Current
Source Control
Content
Marketplace
Settings
Blueprints
Cinematics
Build
Play
Launch
Perspective
Ray Traced Lighting
Show
10
10°
0.25
3
Level: AIUE_V02_003 (Persistent)
World Outliner
dir
Label
Type
AIUE_V02_003 (Editor) World
Lighting Folder
DDirectionalLight DirectionalL
Showing 1 of 647 act
View Options
Details
Select an object to view details.
Content Browser
Output Log
Sequencer
Filters
Search Log
LogEditorActor: Deleted Actor: StaticMeshActor
LogEditorActor: Deleted Actor: StaticMeshActor
LogEditorActor: Deleted Actor: StaticMeshActor
LogEditorActor: Deleted Actor: StaticMeshActor
LogEditorActor: Deleted 70 Actors (0.107 secs)
LogEditorViewport: Clicking on Actor (LMB): StaticMeshActor (floorA_002)
Cmd: SELECT NONE
LogEditorViewport: Clicking on Actor (LMB): StaticMeshActor (floorA_002)
Cmd: SELECT NONE

Indirect Diffuse in Glossy Reflections

GAMEWORKS FOR RAY TRACING

Denoiser Module

Area Shadows

Spherical/Rect./Directional Lights, Soft Shadows

Glossy Reflections

Inter-Object Reflections, Mirror to Glossy

Ambient Occlusion

High Quality Contact Hardening, Support for off-screen objects

Early Access Program:

https://developer.nvidia.com/gameworks-ray-tracing


Ray Traced Shadows


Ray Traced Reflections


Ray Traced Ambient Occlusion

REMOTE VISUALIZATION

VISUALIZATION TRENDS

New Approaches Required to Solve the Remoting Challenge

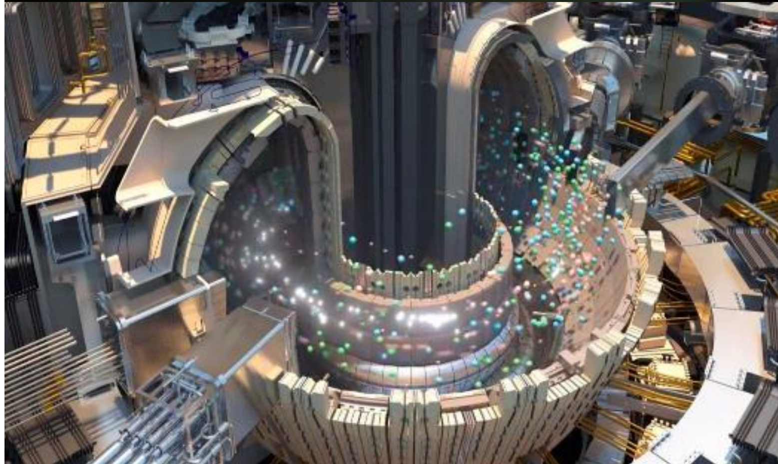
Increasing data set sizes

In-situ scenarios

Interactive workflows

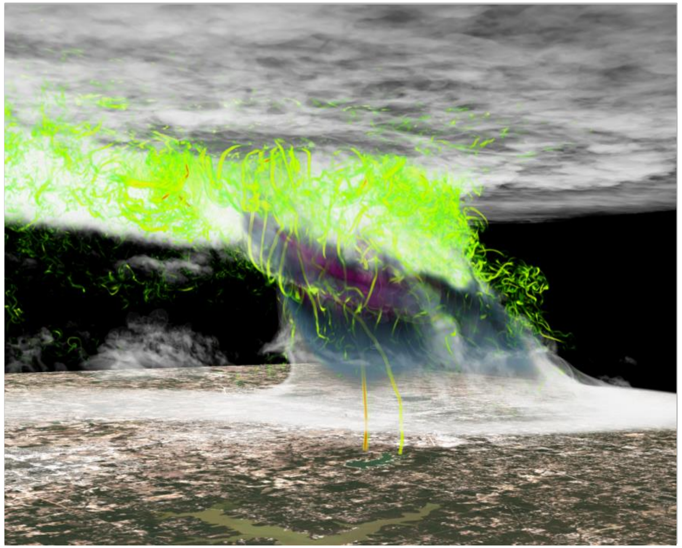
New display technologies

Globally distributed user bases

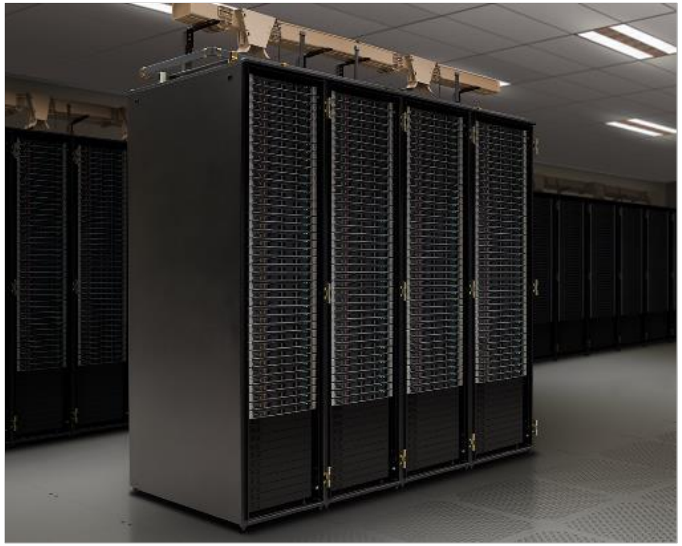


STREAMING

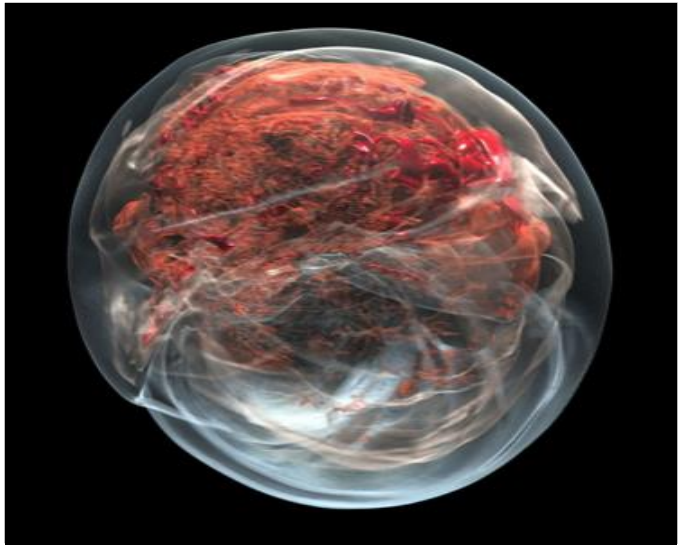
Benefits of Rendering on Supercomputer



Scale with Simulation
No Need to Scale Separate Vis Cluster



Cheaper Infrastructure
All Heavy Lifting Performed on the Server



Interactive High-Fidelity Rendering
Improves Perception and Scientific Insight

FLEXIBLE GPU ACCELERATION ARCHITECTURE

Independent CUDA Cores & Video Engines

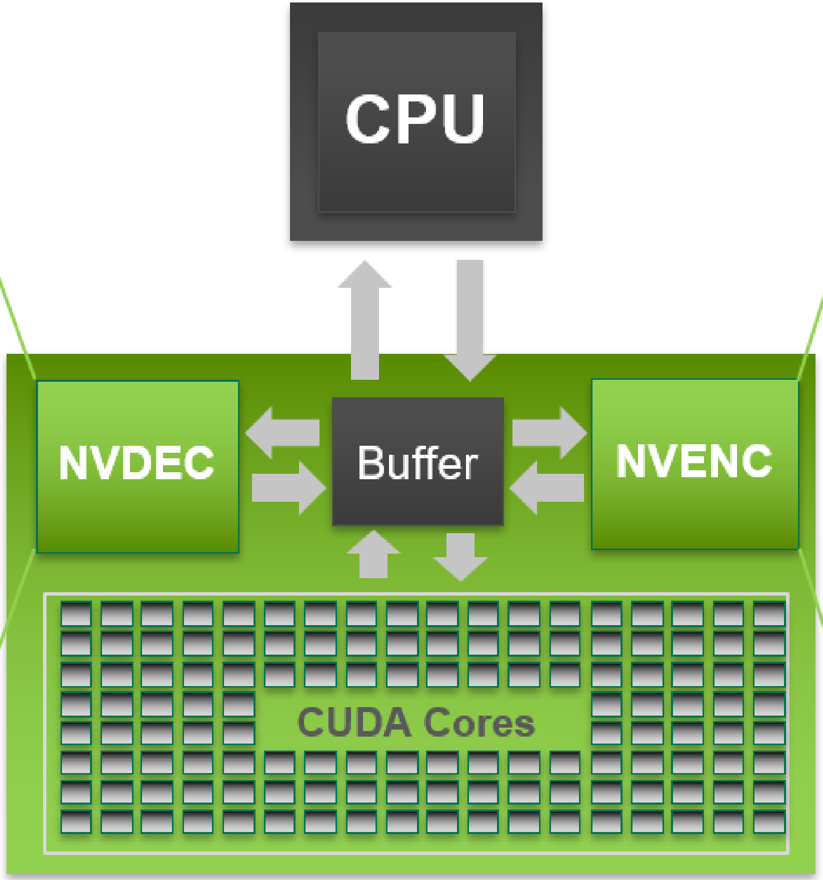
Decode HW*

Formats:
- MPEG-2
- VC1
- VP8
- VP9
- H.264
- H.265
- Lossless

Bit depth:
- 8 bit
- 10 bit
- 12 bit

Color**
- YUV 4:2:0
- YUV 4:4:4

Resolution
- Up to 8K***



Encode HW*

Formats:
- H.264
- H.265
- Lossless

Bit depth:
- 8 bit
- 10 bit

Color**
- YUV 4:4:4
- YUV 4:2:0

Resolution
- Up to 8K***

** Diagram represents support for the NVIDIA Turing GPU family*
*** 4:2:2 is not natively supported on HW*
**** Support is codec dependent*

VIDEO CODEC SDK

APIs For Hardware Accelerated Video Encode/Decode

What's New with Turing GPUs and Video Codec SDK 9.0

- Up to 3x decode throughput with multiple decoders on professional cards (Quadro & Tesla)
- Higher quality encoding - H.264 & H.265
- Higher encoding efficiency (15% lower bitrate than Pascal)
- HEVC B-frames support
- HEVC 4:4:4 decoding support

NVIDIA GeForce Now is made possible by leveraging **NVENC** in the datacenter and streaming the result to end clients

https://developer.nvidia.com/nvidia-video-codec-sdk

NVPIPE

A Lightweight Video Codec SDK Wrapper

Simple C API

H.264, HEVC

RGBA32, uint4, uint8, uint16

Lossy, Lossless

Host/Device memory, OpenGL textures/PBOs

https://github.com/NVIDIA/NvPipe

Issues? Suggestions? Feedback welcome!

```
#include <NvPipe.h>


// Encode
NvPipe* encoder = NvPipe_CreateEncoder(NVPIPE_RGBA32,
    NVPIPE_HEVC, NVPIPE_LOSSY, 32 * 1000 * 1000, 90);

while (...)
{
    uint64_t compressedSize = NvPipe_Encode(encoder,
        rgba, buffer, bufferSize, width, height);
    ...
}

NvPipe_Destroy(encoder);


// Decode
NvPipe* decoder = NvPipe_CreateDecoder(NVPIPE_RGBA32,
    NVPIPE_HEVC);

while (...)
{
    NvPipe_Decode(decoder, buffer, compressedSize,
        rgba, width, height);
    ...
}

NvPipe_Destroy(decoder);
```

PARAVIEW WEB

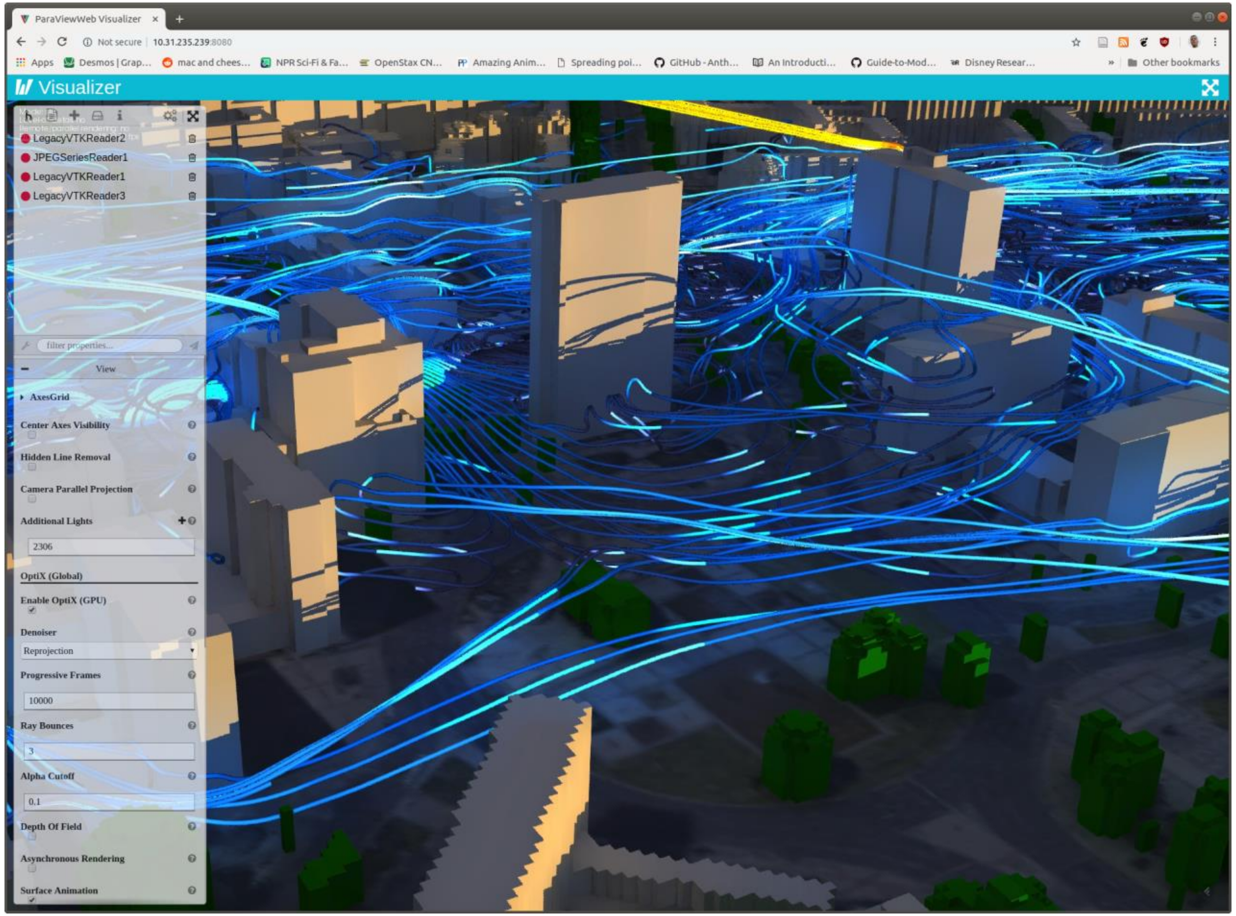
High Performance Visualization in the Browser

Render remotely on high-performance hardware

Supports thin clients, even without discrete GPUs

High-quality, interactive to real-time visualizations

Works with RTX backend out of the box!



SUMMARY

"Ray tracing is the future ~~and ever will be.~~"

CONCLUSION

RTX path tracing in ParaView/VTK available soon!

Physically-based and interchangeable materials via MDL

Stream interactively from your supercomputer!

VisRTX

https://github.com/NVIDIA/VisRTX

NvPipe

https://github.com/NVIDIA/NvPipe

We want to help you solve your large-scale vis problems on NVIDIA!

Tim Biedert
tbiedert@nvidia.com

Mathias Hummel
mathiash@nvidia.com

NVIDIA