ANARI BIRDS OF A FEATHER: WELCOME AND INTRO

John E. Stone,
NVIDIA Distinguished Engineer,
Scientific Visualization Devtech

SC24
Atlanta, GA | hpc creates.

AGENDA:

LIGHTNING TALKS WITH OPEN DISCUSSION

- **Kevin Griffin (NVIDIA)**, "Plans for leveraging ANARI via VTK in the VisIt Visualization Tool"
- **Sankhesh Jhaveri (Kitware)**, "The benefits, challenges, and future of portable rendering in VTK + ParaView"
- **Nicole Marsaglia and Cyrus Harrison (LLNL)**, "Possibilities to leverage ANARI inside, outside, and sideways with Ascent"
- **Victor A. Mateevitsi (ANL)**, "Rendering at Warp Speed: OSPray + ANARI on Aurora"
- **Ken Moreland and David Pugmire (ORNL)**, "Using ANARI to provide and supply rendering in VTK-m"
- **Bill Sherman (NIST)**, "Integrating ANARI into Virtual Reality"

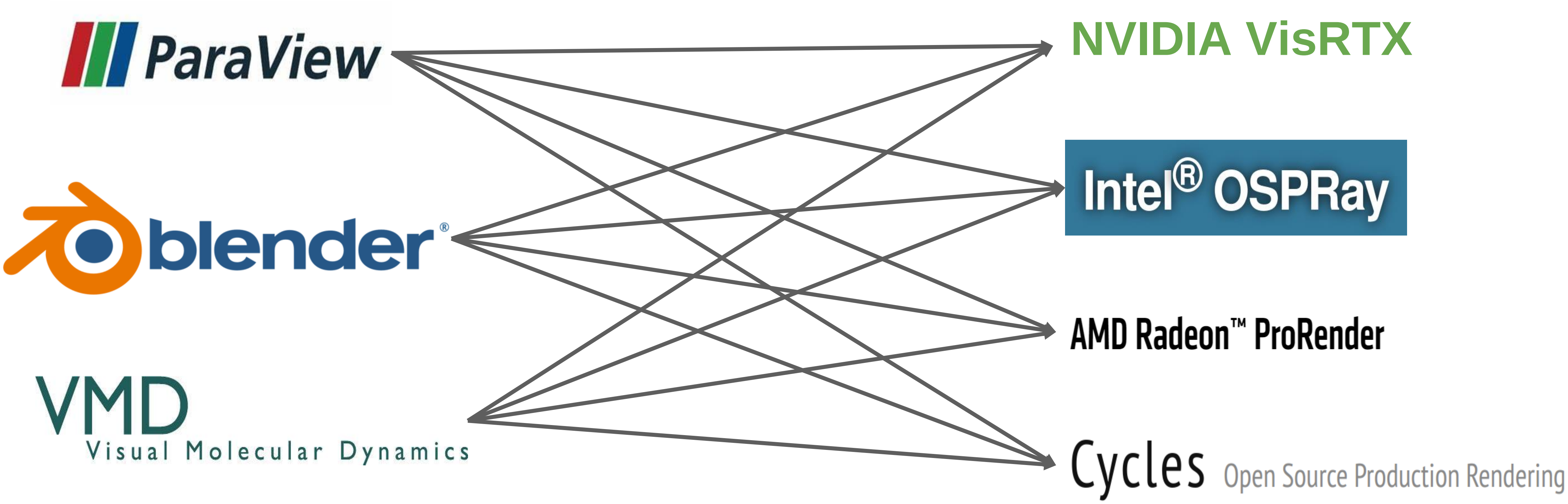
ANARI

HTTPS://WWW.KHRONOS.ORG/ANARI/

- A Khronos Group open standard 3-D rendering API
- Abstracts state-of-the-art renderers and advanced rendering algorithms such as path tracing, for use by diverse applications
- Well suited to the needs of technical and scientific visualization and HPC since it initially developed from within our community
- ANARI Working Group develops the API specification with Advisory Panel
- Participants from industry, national laboratories, academic research
- Ongoing efforts to broaden scope and connect with other standards
- First ANARI hackathon took place in October '24!

3D APPLICATIONS

RENDERING ENGINES



…

…

3D APPLICATIONS

ParaView

blender®

VMD
Visual Molecular Dynamics

...

ANARI™

RENDERING ENGINES

NVIDIA VisRTX

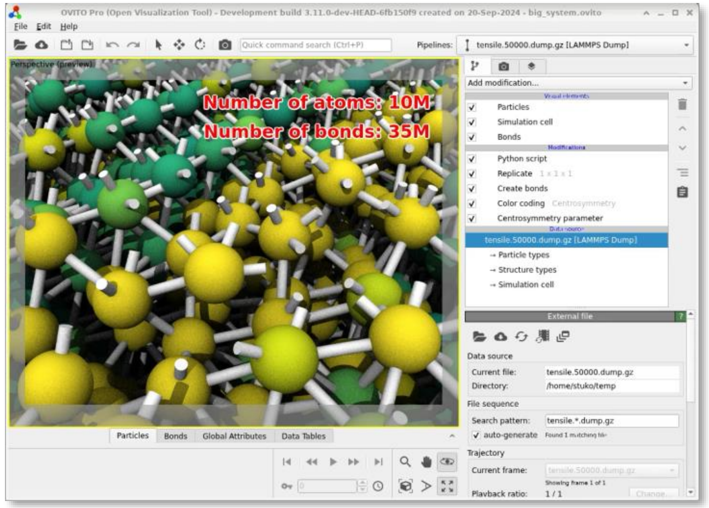
Intel® OSPRay

AMD Radeon™ ProRender

Cycles Open Source Production Rendering

...

ANARI is *Portable* (API Uniformity)



OVITO Pro (Open Visualization Tool) - Development build 3.11.0-dev-HEAD-6fb150f9 created on 20-Sep-2024 - big_system.ovito

Pipelines: tensile.50000.dump.gz [LAMMPS Dump]

Visual elements
- ☑ Particles
- ☑ Simulation cell
- ☑ Bonds

Modifications
- ☑ Python script
- ☑ Replicate 1 x 1 x 1
- ☑ Create bonds
- ☑ Color coding Centrosymmetry
- ☑ Centrosymmetry parameter

Data source
- tensile.50000.dump.gz [LAMMPS Dump]
  - → Particle types
  - → Structure types
  - → Simulation cell

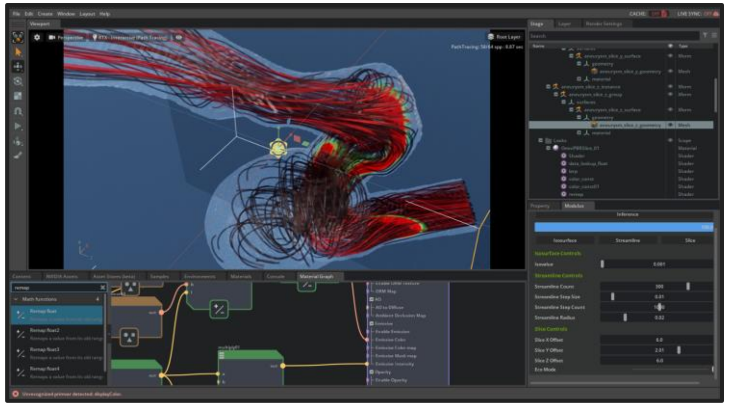
External file

Data source
- Current file: tensile.50000.dump.gz
- Directory: /home/stuko/temp

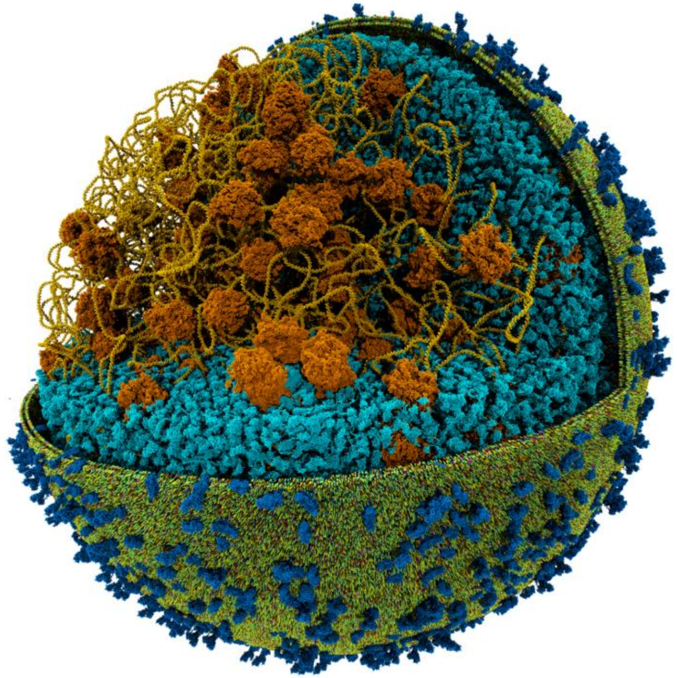
File sequence
- Search pattern: tensile.*.dump.gz
- ☑ auto-generate Found 1 matching file

Trajectory
- Current frame: tensile.50000.dump.gz
- Showing frame 1 of 1
- Playback ratio: 1 / 1

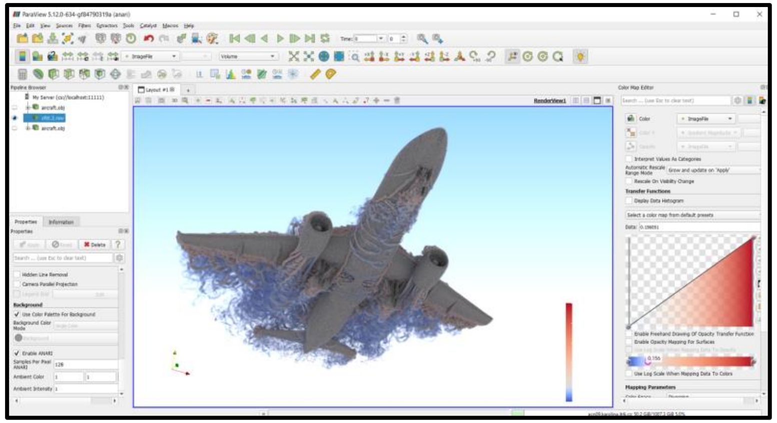
OVITO



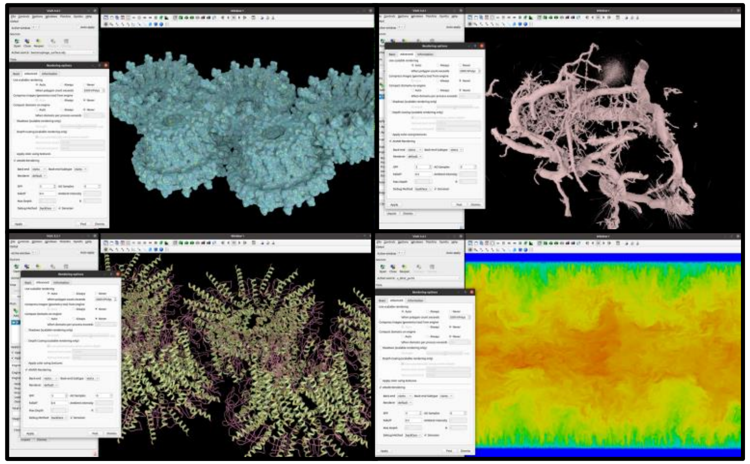
NVIDIA OMNIVERSE



VMD
Visual Molecular Dynamics



ParaView
Parallel Visualization Application



VisIt

ANARI LINKS:

- Home page: https://www.khronos.org/anari/
- Specification: https://registry.khronos.org/ANARI/
- SDK: https://github.com/KhronosGroup/ANARI-SDK

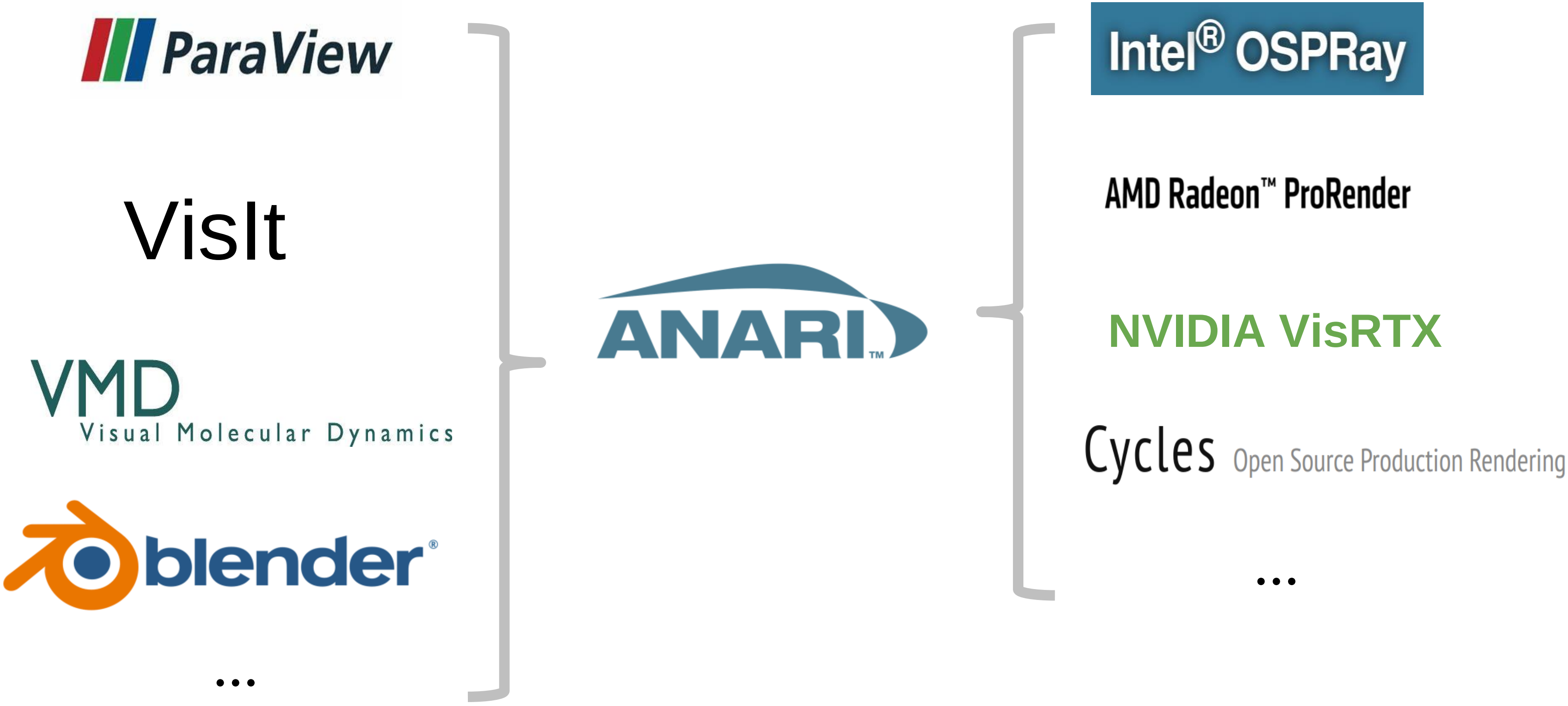
- **Join the ANARI Community:** https://www.khronos.org/anari/#ANARICommunity

Leveraging ANARI via VTK in VisIt

Kevin S. Griffin, Senior Developer Technology Engineer
NVIDIA Corporation / ANARI Working Group Member

3D APPLICATIONS

RENDERING ENGINES



ParaView

VisIt

VMD Visual Molecular Dynamics

blender®

...

ANARI™

Intel® OSPRay

AMD Radeon™ ProRender

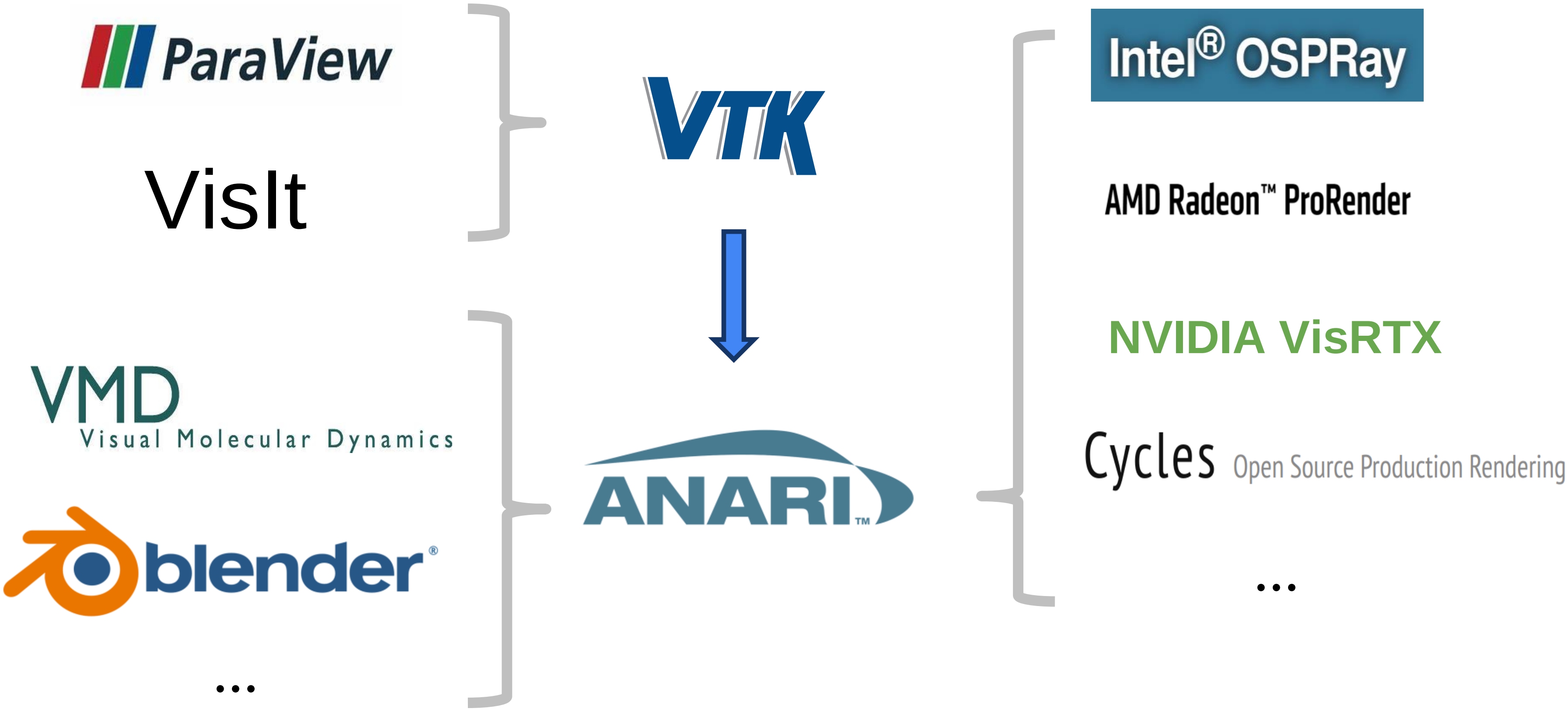
NVIDIA VisRTX

Cycles Open Source Production Rendering

...

3D APPLICATIONS

RENDERING ENGINES



ParaView

VisIt

VTK

VMD Visual Molecular Dynamics

blender®

...

ANARI™

Intel® OSPRay

AMD Radeon™ ProRender

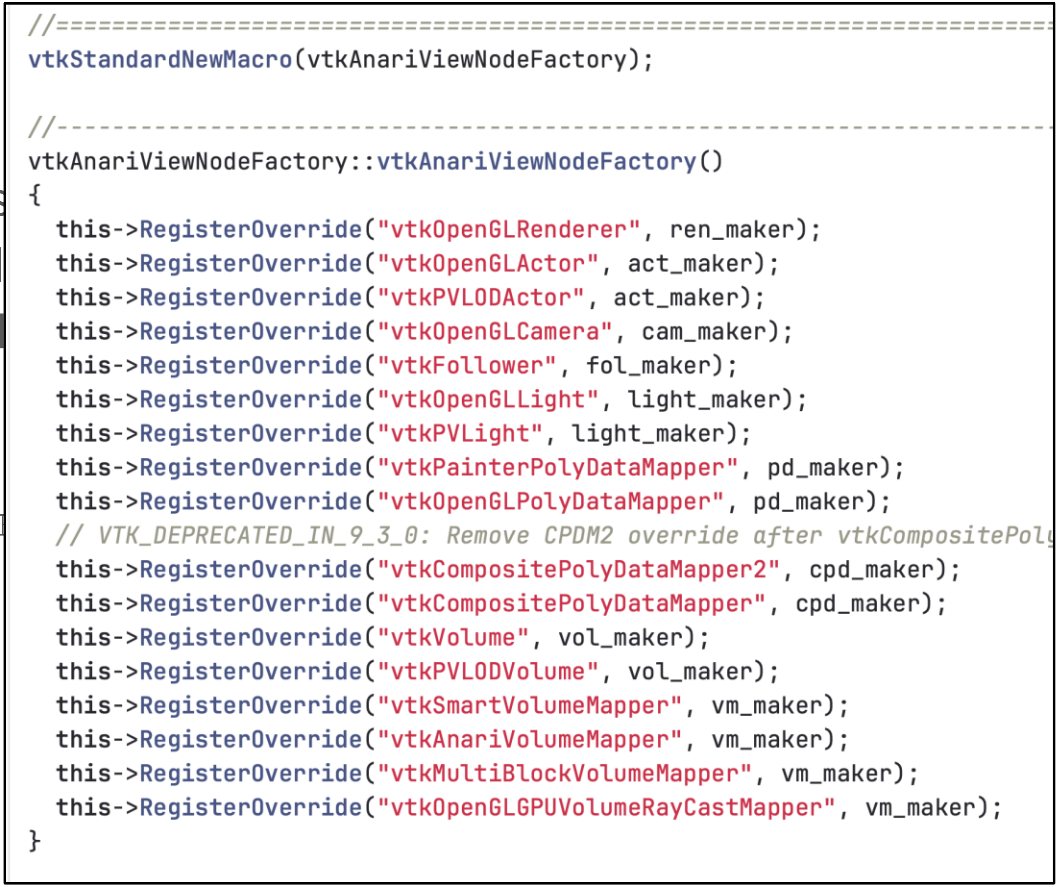
NVIDIA VisRTX

Cycles Open Source Production Rendering

...

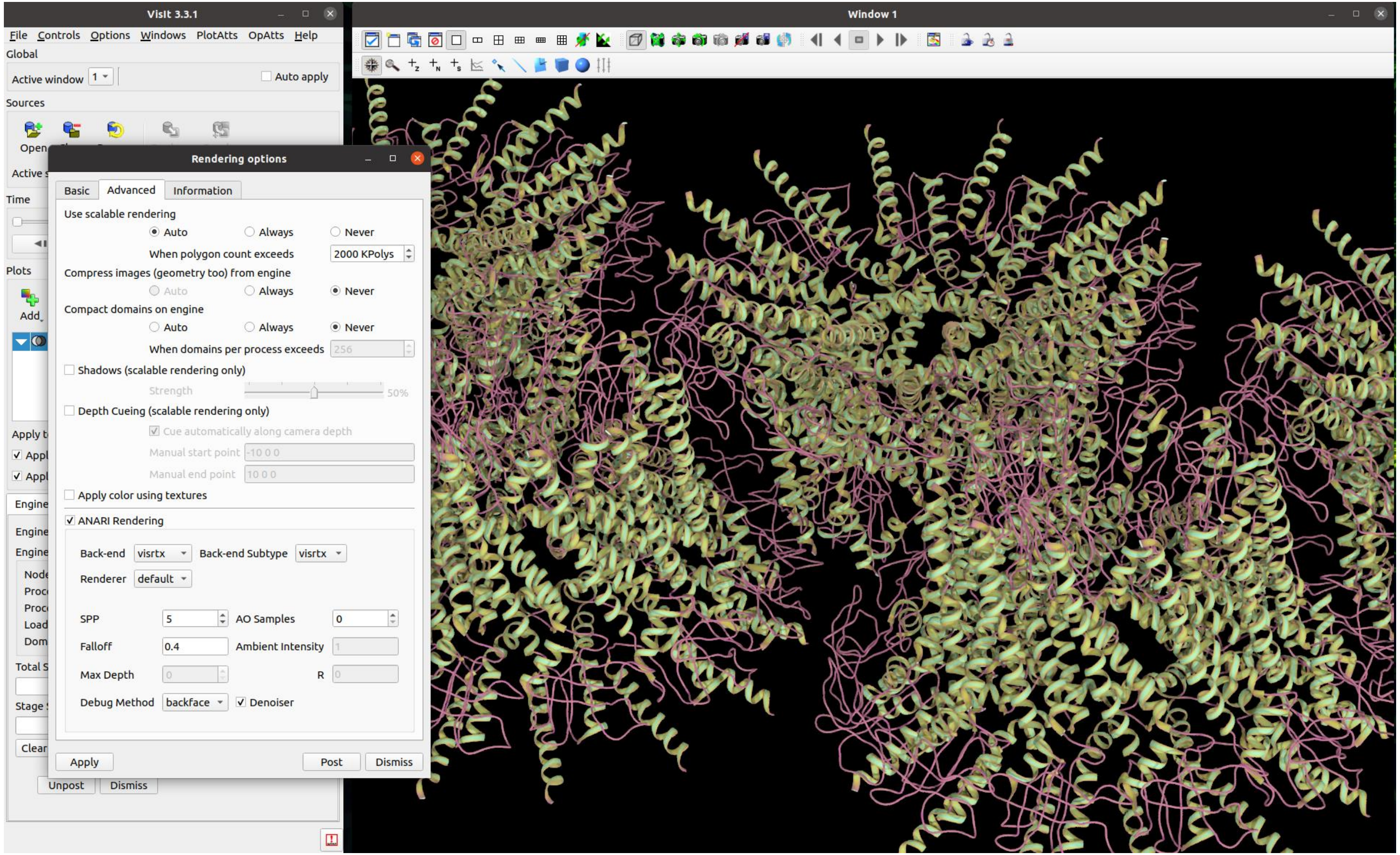
VTK

- VTK Master - https
- Code [VTK->Rend
- Tests [VTK->Rend
- Factory Overrides
- CMake Options
  - -DVTK_MODULE_
  - -Danari_DIR



```
//==========================================================================
vtkStandardNewMacro(vtkAnariViewNodeFactory);

//--------------------------------------------------------------------------
vtkAnariViewNodeFactory::vtkAnariViewNodeFactory()
{
  this->RegisterOverride("vtkOpenGLRenderer", ren_maker);
  this->RegisterOverride("vtkOpenGLActor", act_maker);
  this->RegisterOverride("vtkPVLODActor", act_maker);
  this->RegisterOverride("vtkOpenGLCamera", cam_maker);
  this->RegisterOverride("vtkFollower", fol_maker);
  this->RegisterOverride("vtkOpenGLLight", light_maker);
  this->RegisterOverride("vtkPVLight", light_maker);
  this->RegisterOverride("vtkPainterPolyDataMapper", pd_maker);
  this->RegisterOverride("vtkOpenGLPolyDataMapper", pd_maker);
  // VTK_DEPRECATED_IN_9_3_0: Remove CPDM2 override after vtkCompositePol
  this->RegisterOverride("vtkCompositePolyDataMapper2", cpd_maker);
  this->RegisterOverride("vtkCompositePolyDataMapper", cpd_maker);
  this->RegisterOverride("vtkVolume", vol_maker);
  this->RegisterOverride("vtkPVLODVolume", vol_maker);
  this->RegisterOverride("vtkSmartVolumeMapper", vm_maker);
  this->RegisterOverride("vtkAnariVolumeMapper", vm_maker);
  this->RegisterOverride("vtkMultiBlockVolumeMapper", vm_maker);
  this->RegisterOverride("vtkOpenGLGPUVolumeRayCastMapper", vm_maker);
}
```

VisIt – Workflow 1 [Surface Rendering]



Use scalable rendering
● Auto ○ Always ○ Never
When polygon count exceeds 2000 KPolys

Compress images (geometry too) from engine
○ Auto ○ Always ● Never

Compact domains on engine
○ Auto ○ Always ● Never
When domains per process exceeds 256

□ Shadows (scalable rendering only)
Strength 50%

□ Depth Cueing (scalable rendering only)
☑ Cue automatically along camera depth
Manual start point -10 0 0
Manual end point 10 0 0

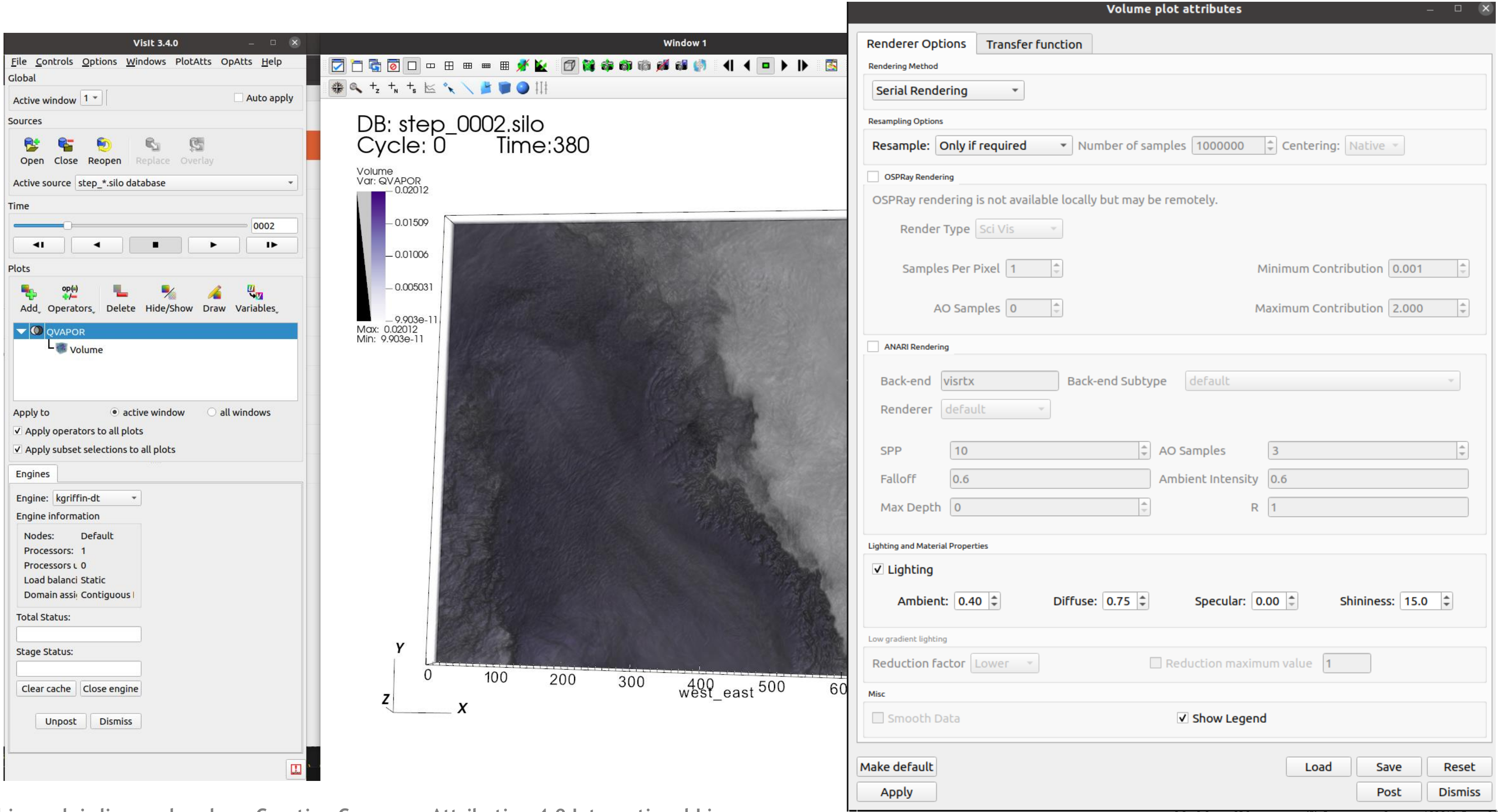
□ Apply color using textures

☑ ANARI Rendering

Back-end visrtx Back-end Subtype visrtx
Renderer default

SPP 5 AO Samples 0
Falloff 0.4 Ambient Intensity 1
Max Depth 0 R 0
Debug Method backface ☑ Denoiser

VisIt – Workflow 2 [Volume Rendering]



Global

Active window 1

□ Auto apply

Sources

Active source step_*.silo database

Time 0002

Plots

QVAPOR
Volume

Apply to ● active window ○ all windows

☑ Apply operators to all plots

☑ Apply subset selections to all plots

Engines

Engine: kgriffin-dt

Engine information

Nodes: Default
Processors: 1
Processors u 0
Load balanci Static
Domain assi Contiguous

Total Status:

Stage Status:

DB: step_0002.silo
Cycle: 0 Time:380

Rendering Method

Serial Rendering

Resampling Options

Resample: Only if required Number of samples 1000000 Centering: Native

□ OSPRay Rendering

OSPRay rendering is not available locally but may be remotely.

Render Type Sci Vis

Samples Per Pixel 1 Minimum Contribution 0.001

AO Samples 0 Maximum Contribution 2.000

□ ANARI Rendering

Back-end visrtx Back-end Subtype default

Renderer default

SPP 10 AO Samples 3

Falloff 0.6 Ambient Intensity 0.6

Max Depth 0 R 1

Lighting and Material Properties

☑ Lighting

Ambient: 0.40 Diffuse: 0.75 Specular: 0.00 Shininess: 15.0

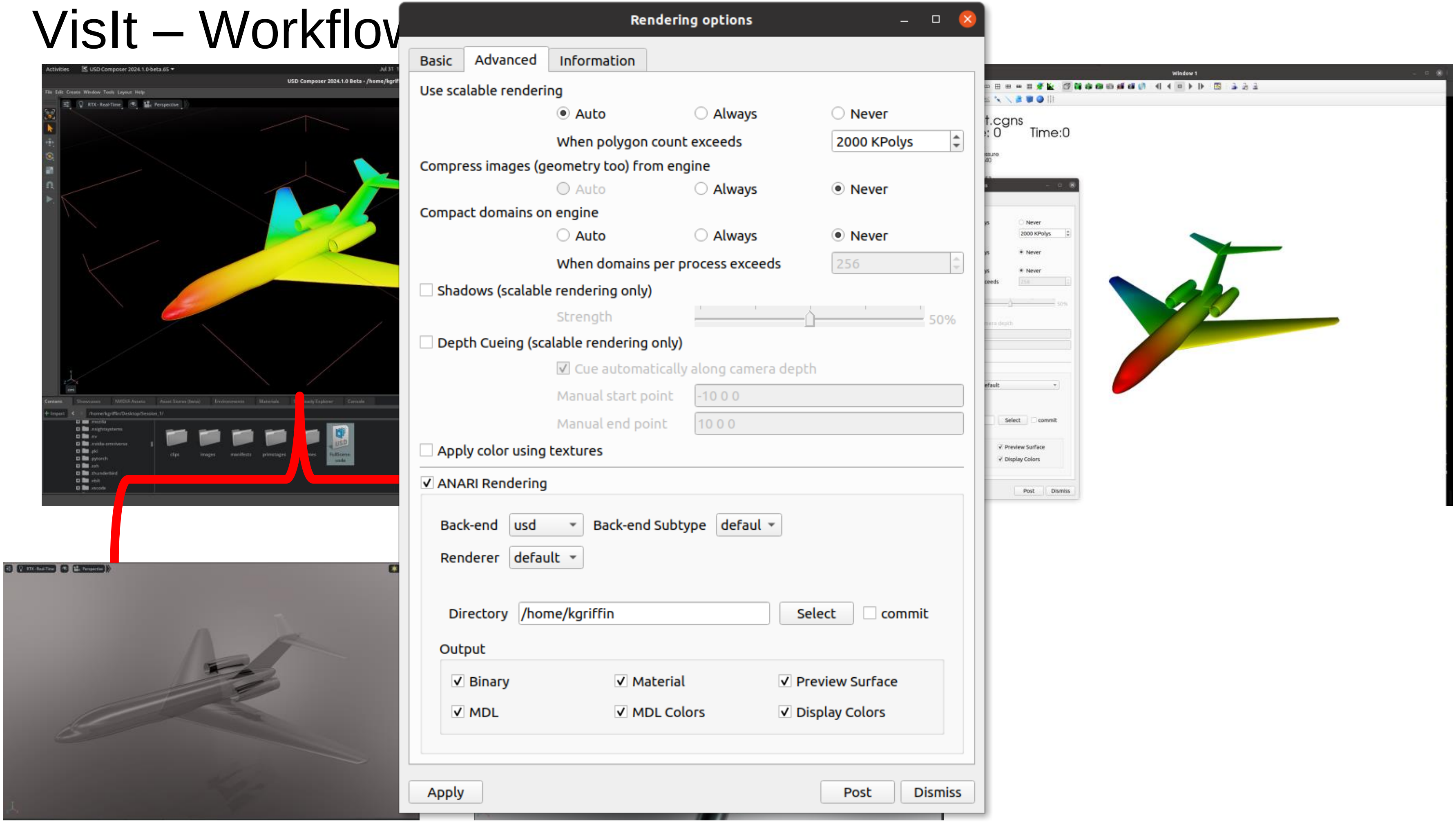
Low gradient lighting

Reduction factor Lower □ Reduction maximum value 1

Misc

□ Smooth Data ☑ Show Legend

VisIt – Workflov



Rendering options

Basic | Advanced | Information

Use scalable rendering

● Auto ○ Always ○ Never

When polygon count exceeds 2000 KPolys

Compress images (geometry too) from engine

○ Auto ○ Always ● Never

Compact domains on engine

○ Auto ○ Always ● Never

When domains per process exceeds 256

□ Shadows (scalable rendering only)

Strength 50%

□ Depth Cueing (scalable rendering only)

☑ Cue automatically along camera depth

Manual start point -10 0 0

Manual end point 10 0 0

□ Apply color using textures

☑ ANARI Rendering

Back-end usd Back-end Subtype defaul

Renderer default

Directory /home/kgriffin Select □ commit

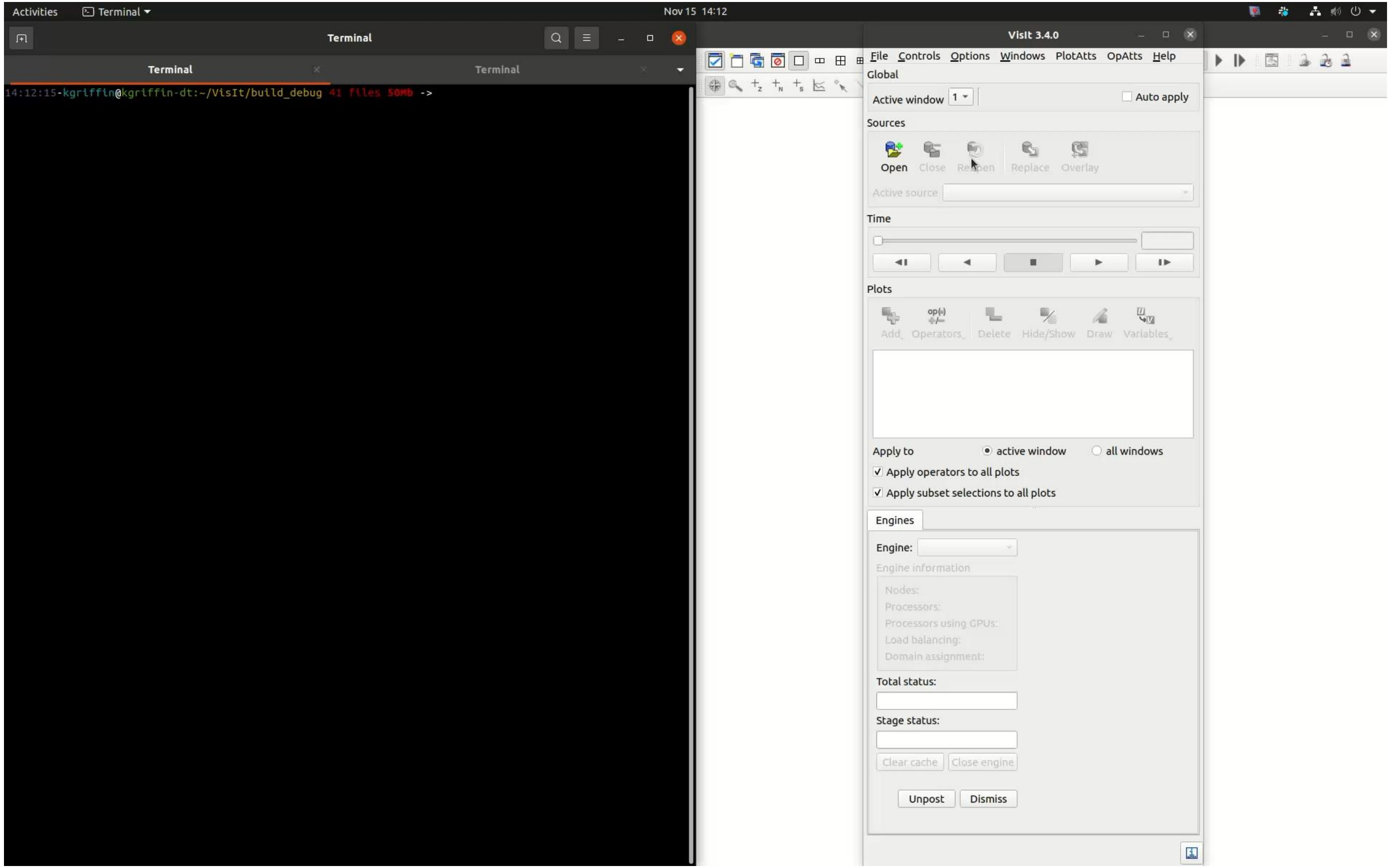
Output

☑ Binary ☑ Material ☑ Preview Surface

☑ MDL ☑ MDL Colors ☑ Display Colors

Apply | Post | Dismiss

VisIt – Dynamic UI



```
14:12:15-kgriffin@kgriffin-dt:~/VisIt/build_debug 41 files 50Mb ->
```

Active window 1

□ Auto apply

Apply to ● active window ○ all windows

☑ Apply operators to all plots

☑ Apply subset selections to all plots

Engine:

The benefits, challenges, and future of portable rendering in VTK + ParaView

Sankhesh Jhaveri, Kitware, Inc.

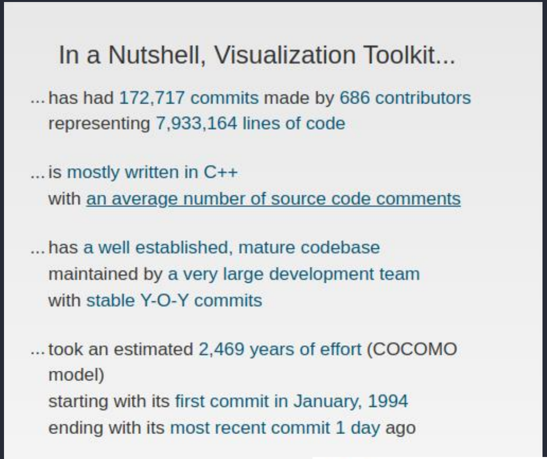
kitware

What are VTK and ParaView?

- **VTK is an object-oriented** approach to scientific high performance visualization
- **ParaView** is a client-server post-processing architecture that uses VTK for data processing and rendering
- **C++** class library
- Automated Java, Tcl, **Python** bindings - .NET/C# through ActiViz
- **Portable** across Unix, Windows, MacOS
- Supports 2D/3D graphics, visualization, image processing, volume rendering, infoviz, geoviz
- Active **discourse** forum and 100+ active developers
- World-wide academic, commercial, government users
- **Free** (OSI-approved BSD 3-clause License)

Three decades of VTK

https://www.kitware.com/happy-birthday-vtk-30-years-of-innovation/



In a Nutshell, Visualization Toolkit...

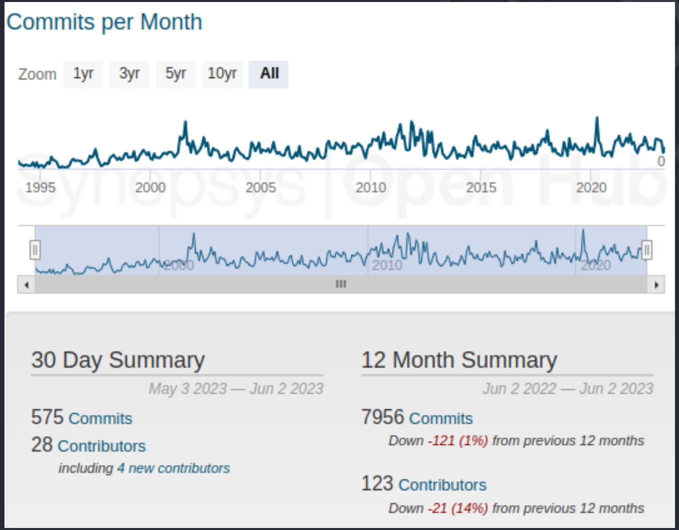
... has had 172,717 commits made by 686 contributors representing 7,933,164 lines of code

... is mostly written in C++ with an average number of source code comments

... has a well established, mature codebase maintained by a very large development team with stable Y-O-Y commits

... took an estimated 2,469 years of effort (COCOMO model) starting with its first commit in January, 1994 ending with its most recent commit 1 day ago

Ref: https://openhub.net/p/vtk



Commits per Month

30 Day Summary

May 3 2023 — Jun 2 2023

575 Commits

28 Contributors

including 4 new contributors

12 Month Summary

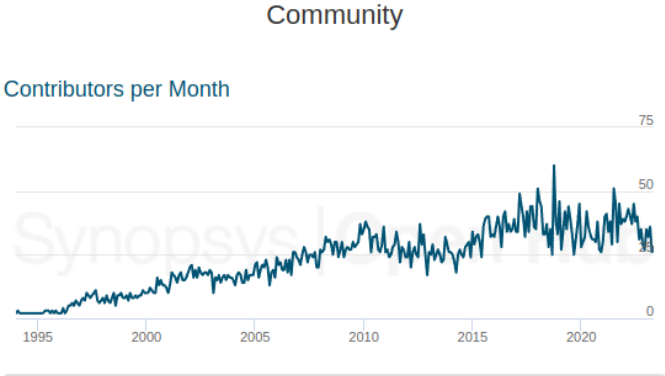
Jun 2 2022 — Jun 2 2023

7956 Commits

Down -121 (1%) from previous 12 months

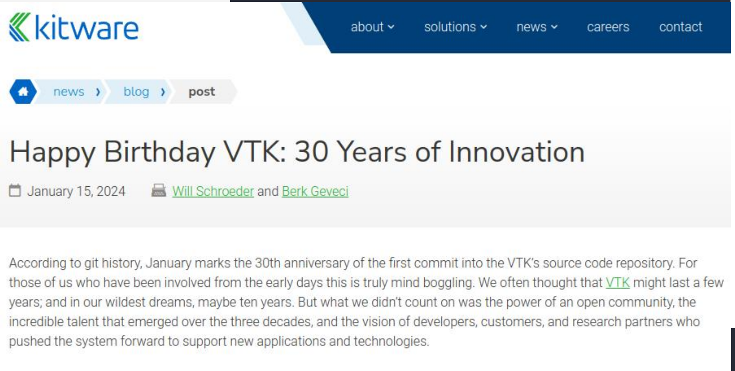
123 Contributors

Down -21 (14%) from previous 12 months



Community

Contributors per Month



Happy Birthday VTK: 30 Years of Innovation

January 15, 2024 Will Schroeder and Berk Geveci

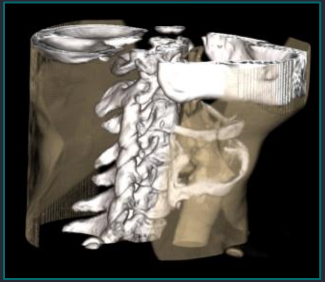
According to git history, January marks the 30th anniversary of the first commit into the VTK's source code repository. For those of us who have been involved from the early days this is truly mind boggling. We often thought that VTK might last a few years; and in our wildest dreams, maybe ten years. But what we didn't count on was the power of an open community, the incredible talent that emerged over the three decades, and the vision of developers, customers, and research partners who pushed the system forward to support new applications and technologies.

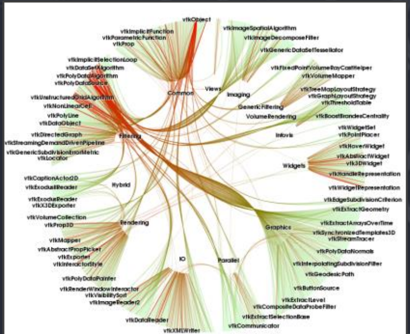
Design Philosophy

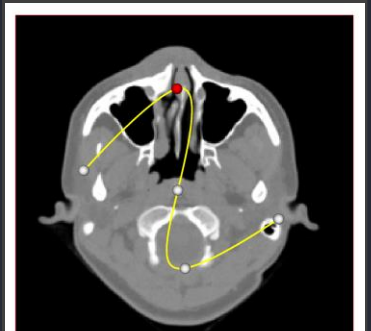
- **Open ended architecture that you use to construct programs**
- **Modular architecture: each module does one thing well**
  - Modules implemented in Object Oriented Classes
  - Pipeline: data flows through modules in a pipeline
- **Underlying themes**
  - Process data
    - To find the salient features
    - To produce imagery that conveys meaning
  - Interact with data
    - Give interactive controls to the user
    - Let the end user do the searching, visually
  - Large data
    - Parallel processing and rendering with MPI
    - Lazy evaluation: only process what is changed

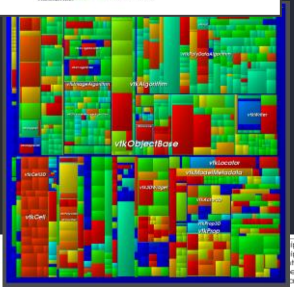
What can VTK do for me?

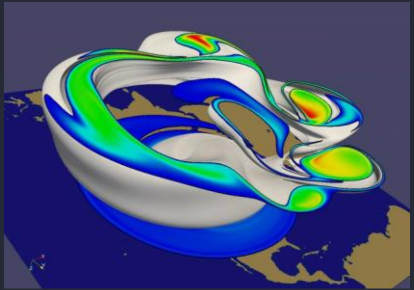
- **Scientific Visualization**
  - 0D: Information visualization
  - 2D: Charting/plotting
  - 3D / 4D: data processing and rendering
  - Image processing
  - Volume rendering
- **Application support**
  - Cross-platform UI
  - Interactive widgets

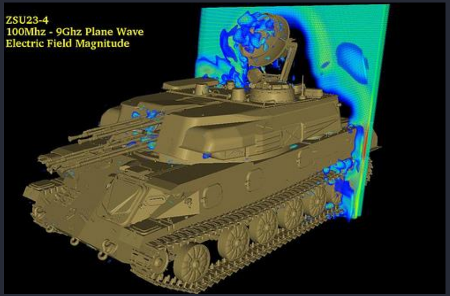


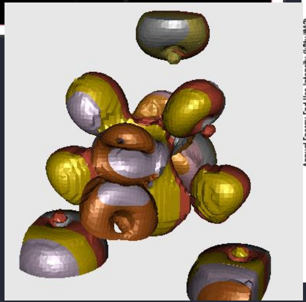


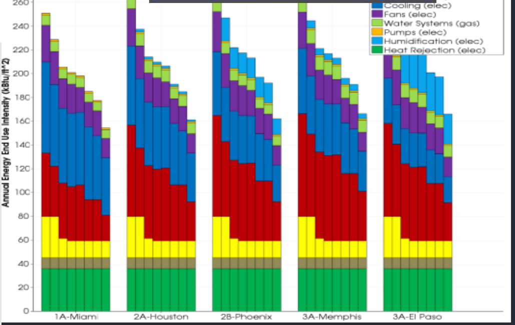




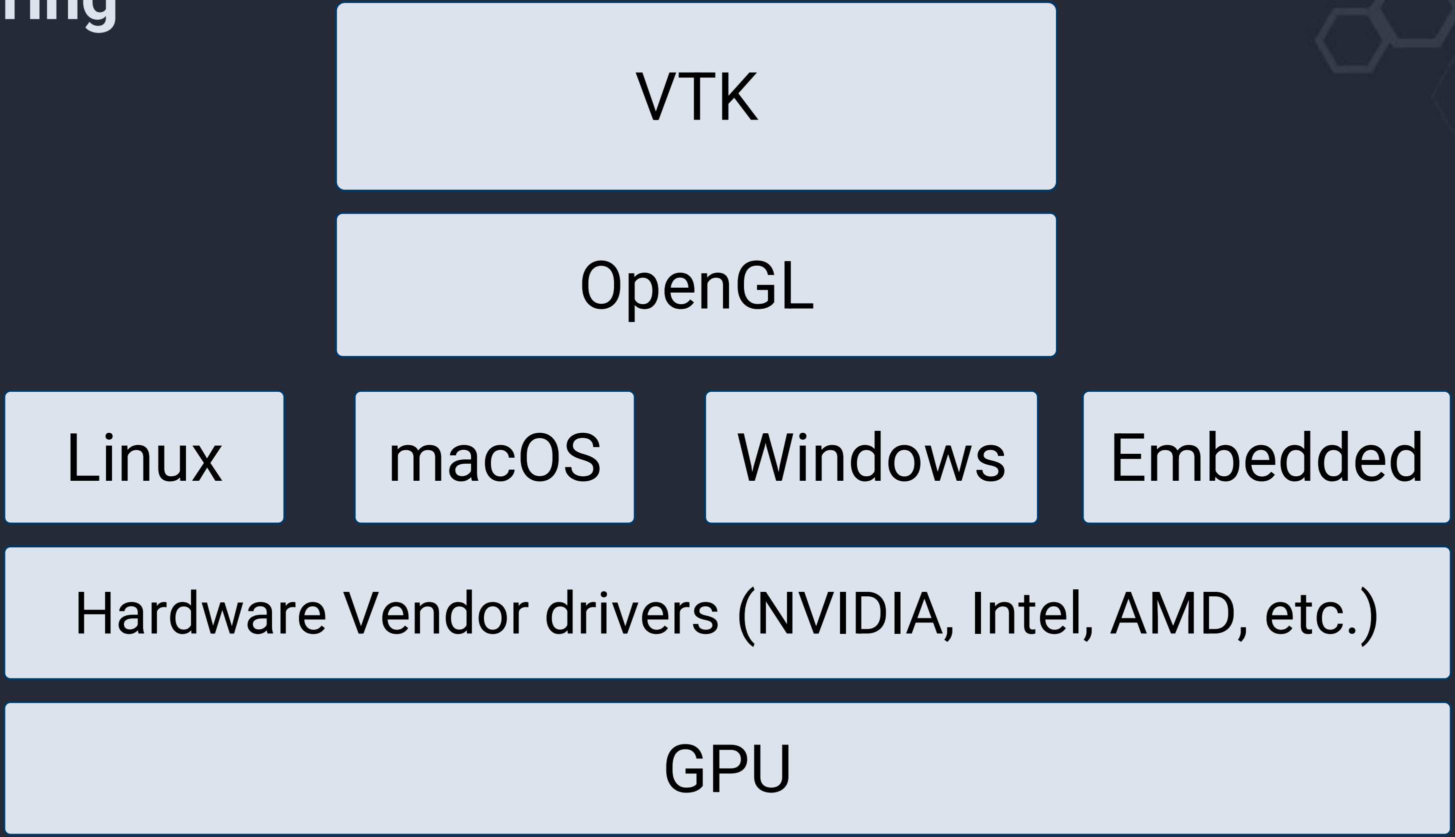








Rendering



VTK

OpenGL

Linux

macOS

Windows

Embedded

Hardware Vendor drivers (NVIDIA, Intel, AMD, etc.)

GPU

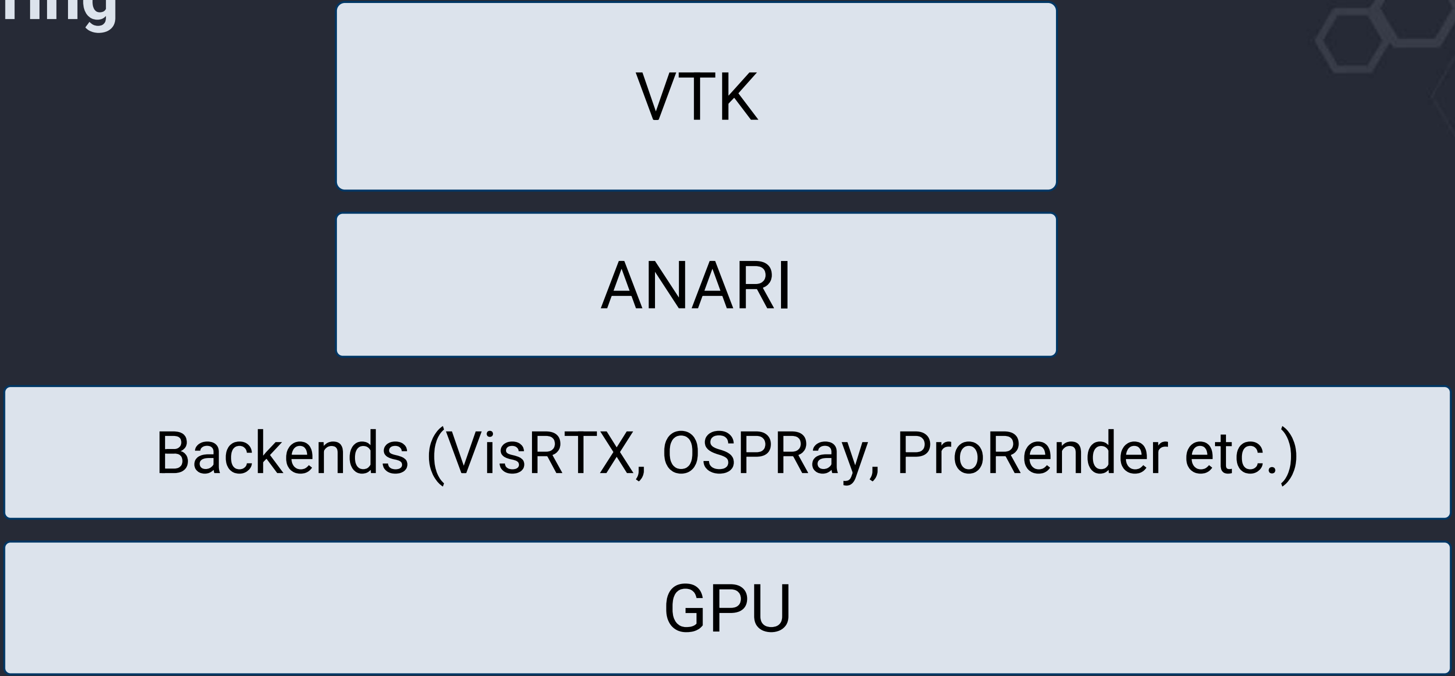
OpenGL departure

- OpenGL 4.6 released in July 2017
- Core architecture was not sufficient for modern graphics hardware
- Dependencies on extensions
- Multithreading and asynchronous processing
- Vulkan
- Apple -> Metal
- Microsoft -> Direct3D

Khronos® ANARI™

- **High-level functionality**
- **Global illumination and ray-tracing**
- **Advanced rendering for scientific visualization**
- **Implementations/backends**
  - VisRTX
  - OSPRay
  - Blenders Cycles

Rendering



VTK

ANARI

Backends (VisRTX, OSPRay, ProRender etc.)

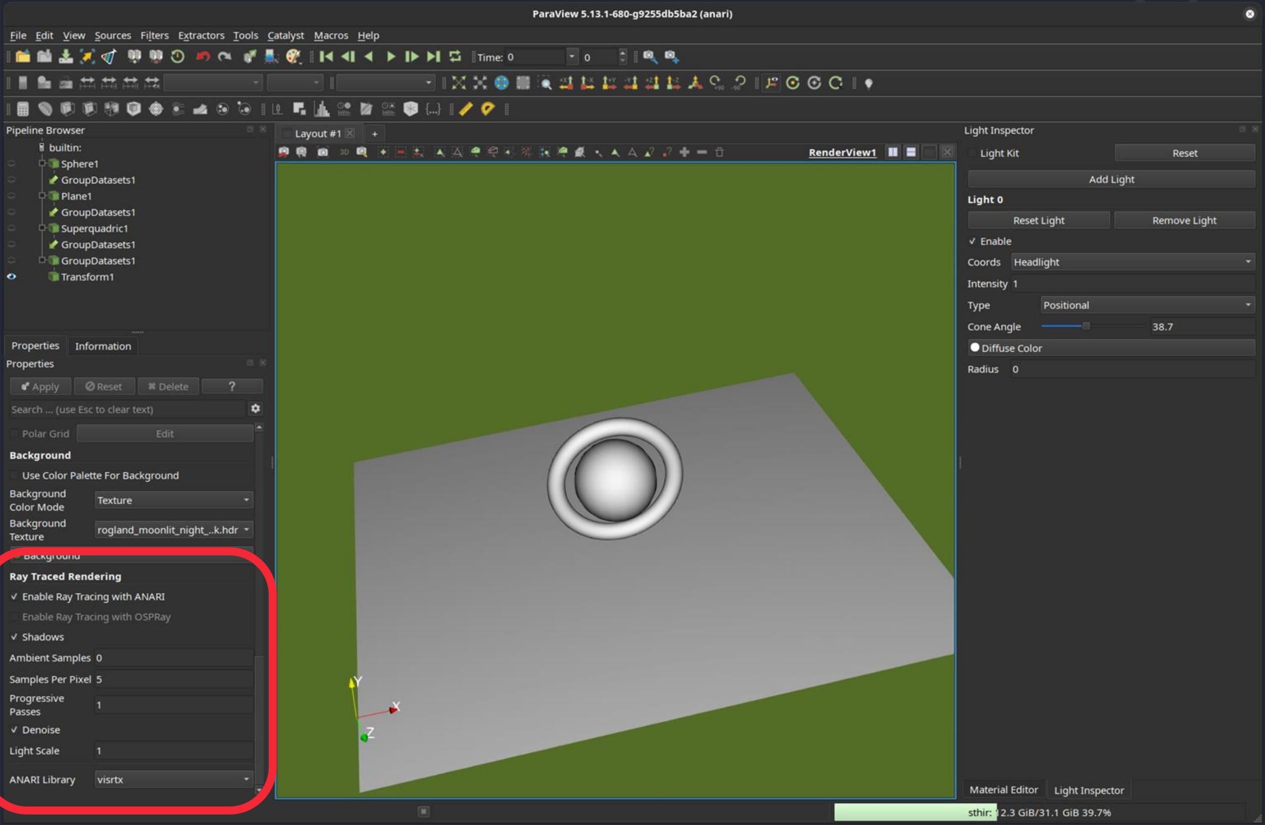
GPU

Khronos® ANARI™ in VTK / ParaView / VTK-m

- **vtkRenderingANARI module**
- **ParaView UI (WIP)**
- **vtk-m ANARI backend (WIP)**

```
// enable ANARI with render passes
vtkNew<vtkANARIPass> anariP;
vtkOpenGLRenderer::SafeDownCast(renderer)->SetPass(anariP);

// ANARI configuration
export ANARI_LIBRARY=visrtx;
vtkANARIRendererNode::SetRendererType("scivis", renderer);
```

Khronos® ANARI™ in VTK / ParaView / VTK-m



Properties

Polar Grid

Background

Use Color Palette For Background

Background Color Mode: Texture

Background Texture: rogland_moonlit_night_..k.hdr

Ray Traced Rendering

✓ Enable Ray Tracing with ANARI

Enable Ray Tracing with OSPRay

✓ Shadows

Ambient Samples: 0

Samples Per Pixel: 5

Progressive Passes: 1

✓ Denoise

Light Scale: 1

ANARI Library: visrtx

Light Inspector

Light Kit

Light 0

✓ Enable

Coords: Headlight

Intensity: 1

Type: Positional

Cone Angle: 38.7

● Diffuse Color

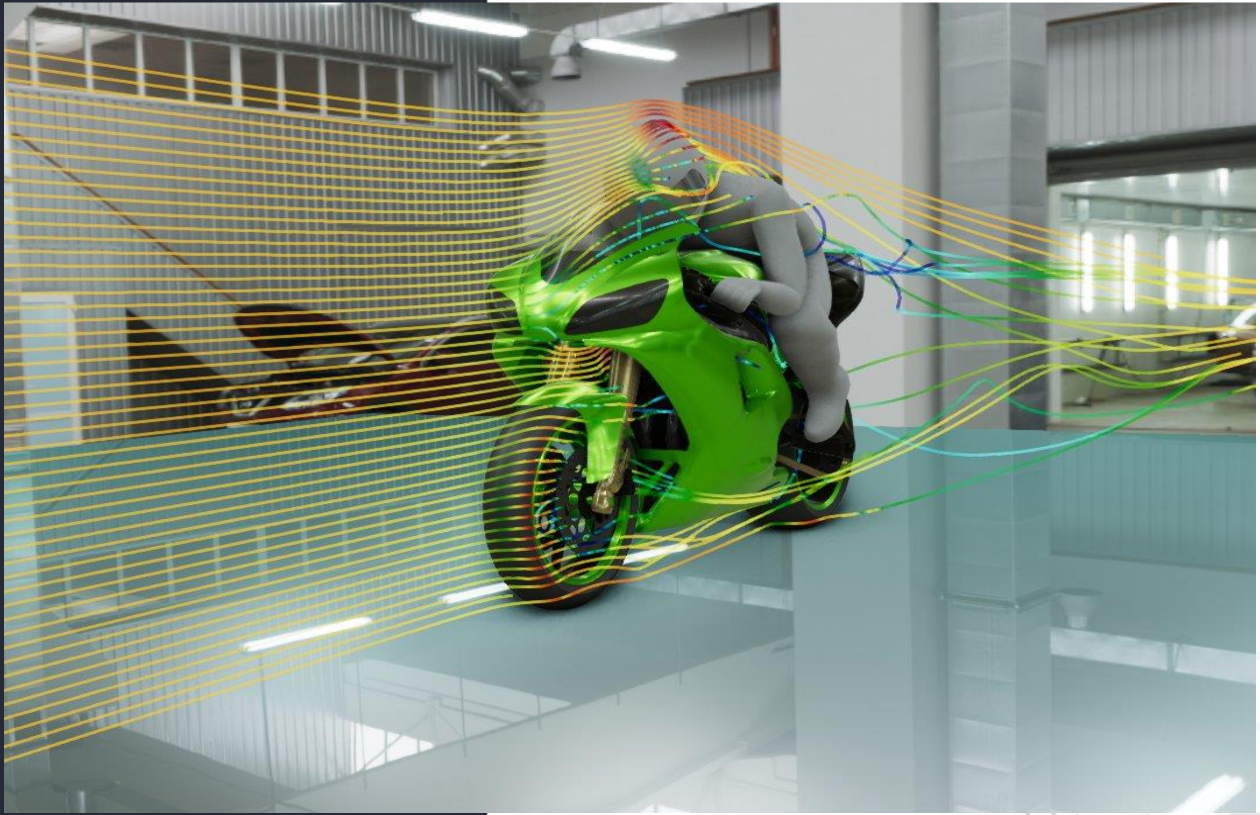
Radius: 0

Yet to come

- AMR grids
- Unstructured grid volume rendering
- Higher order elements
- GPU zero-copy rendering
- Uniformity between backends

Thank you

Questions?



Possibilities to leverage ANARI inside, outside, and sideways with the Ascent In Situ Library

SC24 ANARI BOF

Tuesday November 19th, 2024

Cyrus Harrison (LLNL),
Nicole Marsaglia (LLNL)

LLNL-PRES-2001039
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

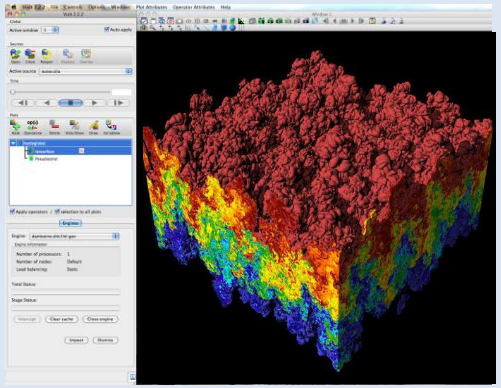
Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344.
Lawrence Livermore National Security, LLC

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

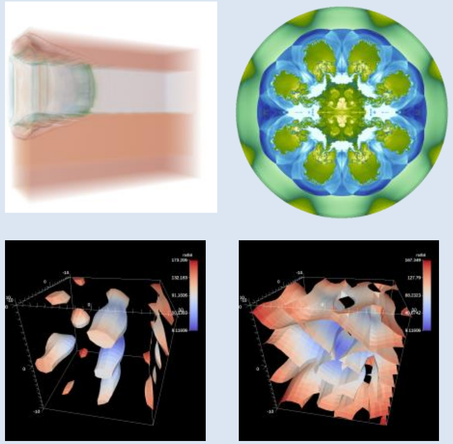
The LLNL VisIt team develops open-source Visualization, Analysis, and I/O tools.

VisIt



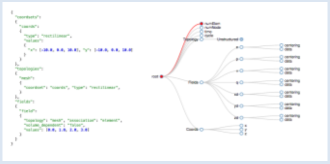
Turnkey HPC application for visualization and analysis of simulation data

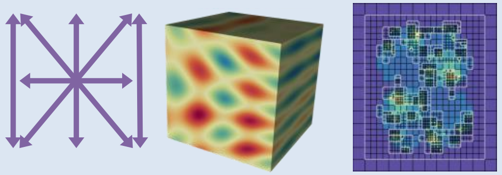
Ascent



Easy-to-use flyweight in situ visualization and analysis library for HPC simulations

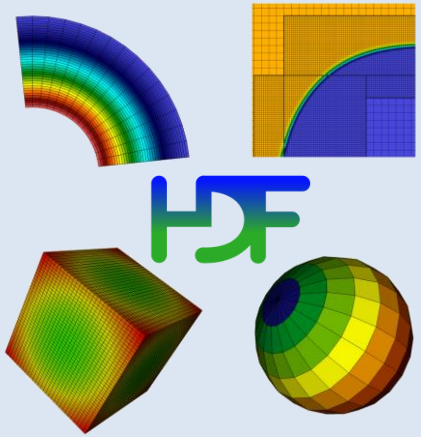
CONDUIT





In-memory data description, HPC I/O, and shared schemas for simulation data exchange

Silo

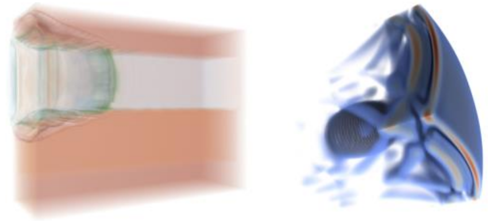


File-based, scientific data exchange library for checkpoint restart and visualization

Ascent is an easy-to-use flyweight in situ visualization and analysis library for HPC simulations

- **Easy to use in-memory visualization and analysis**
  - Use cases: ***Making Pictures, Transforming Data,*** and ***Capturing Data***
  - Young effort, yet already supports most common visualization operations
  - Provides a simple infrastructure to integrate custom analysis
  - Provides C++, C, Python, and Fortran APIs
- **Uses a flyweight design targeted at next-generation HPC platforms**
  - Efficient distributed-memory (MPI) and many-core (CUDA, HIP, OpenMP) execution
    - Demonstrated scaling: In situ filtering and ray tracing across ***16,384 GPUs*** on LLNL's Sierra Cluster
  - Has lower memory requirements than current tools
  - Requires less dependencies than current tools (ex: no OpenGL)

Ascent



Visualizations created using Ascent

python™ HDF CINEMA

Extracts supported by Ascent

http://ascent-dav.org

https://github.com/Alpine-DAV/ascent

Website and GitHub Repo

Why are we interested in ANARI?

One API to a leverage diverse set of runtimes:

- It is not possible for small software development teams to support the full crop of rendering APIs and runtimes in our products.

  Examples from VisIt's past:

  GL Variants (GLX, OSMesa, EGL, OpenSWR), SLIVR, Manta, OSPRay, Index, etc.

Less direct software dependences:

- Ascent is a library directly linked into simulations, build/deployment pose even bigger barriers compared a standalone application like VisIt
  - ANARI provides run time loading of backends, and paths to tools via Universal Scene Description (USD)

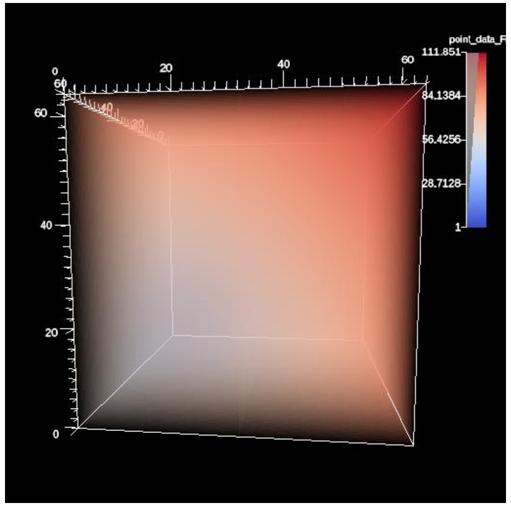
Ascent plus ANARI: Inside, Outside, Sideways?

Possibilities:

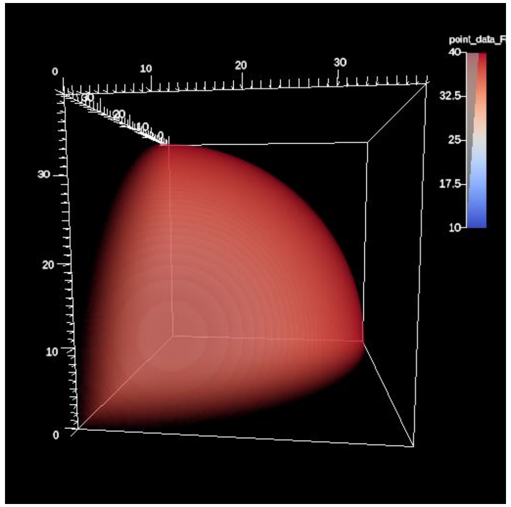
- Ascent Rendering using ANARI
  - — Provide Ascent users access to a wide range of ANARI-based Rendering backends
    - OSPRay, VisRTX, Radeon ProRender, Cycles
  - — *We are exploring this path via current Viskores (VTK-m) ANARI support*
- Ascent as a frontend to Universal Scene Description (USD) Ecosystem
- Ascent rendering as ANARI Backend
  - — Provide access Ascent’s two GPU + MPI Distributed Memory Renderers via an ANARI Interface
- ANARI implementation to select between Ascent’s two internal renderers
- Conduit Blueprint as both an ANARI frontend or backend (data sink)

WIP: Ascent Rendering using ANARI

- Viskores (VTK-m) now has ANARI-compatible filters, connecting Viskores datasets to the ANARI frontend
- Our prototype uses ANARI's built-in CPU ray tracer, Helide, for single node rendering and Ascent's MPI compositing
- Initial Single Node Volume Rendering Comparisons:

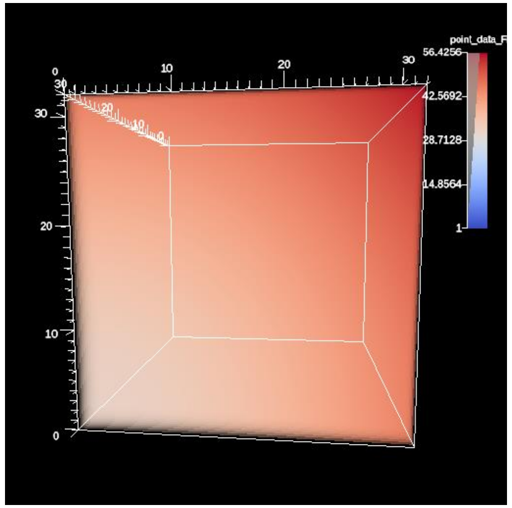


Structured

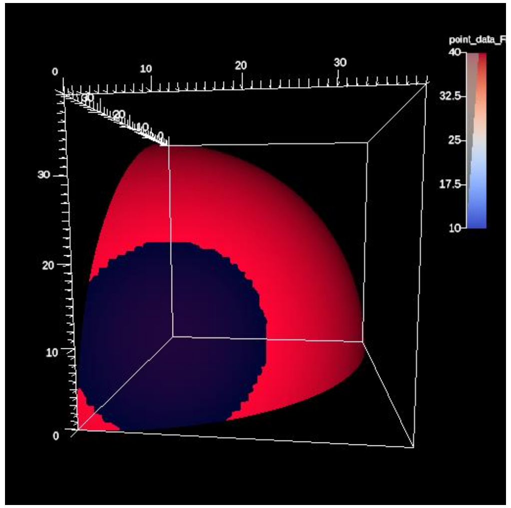


Unstructured

Viskores



Structured

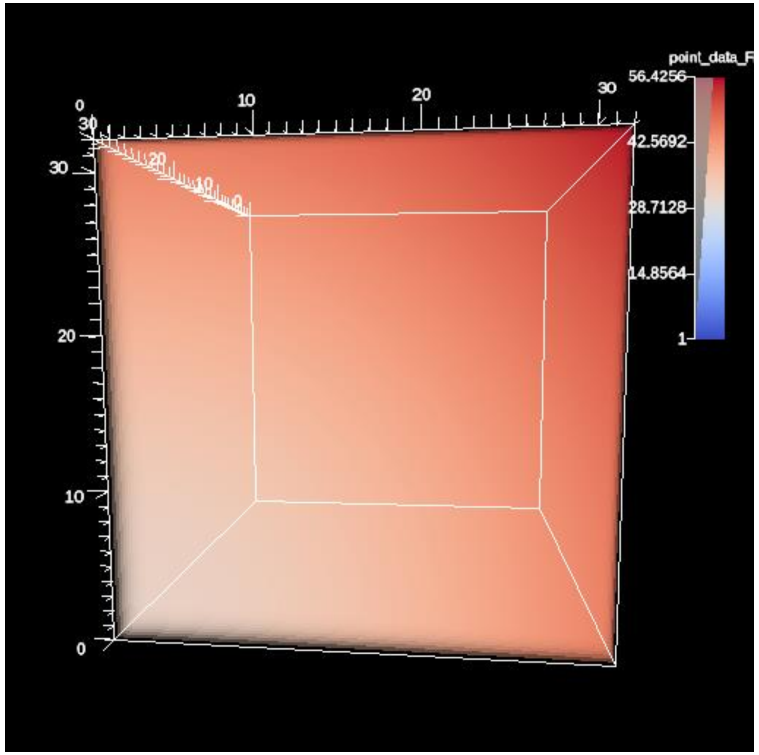


Unstructured

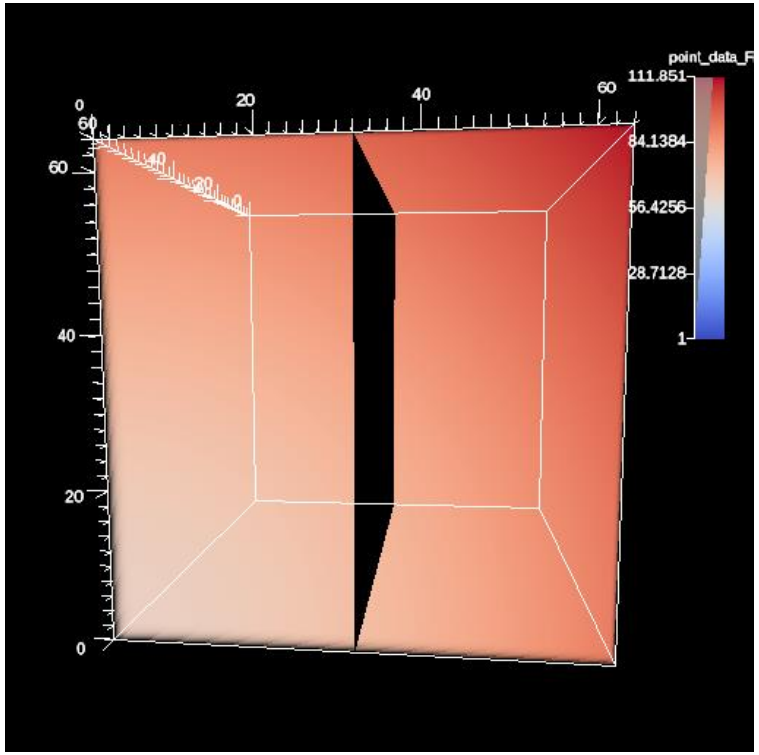
ANARI + Helide

WIP: Ascent Rendering using ANARI

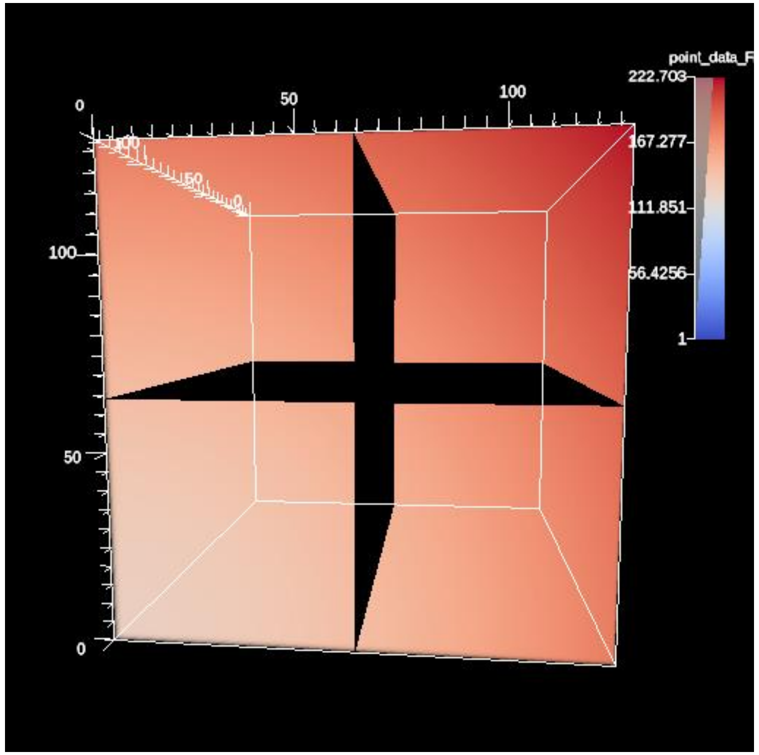
- We are working multi-domain data



1 Domain



2 Domains



4 Domains

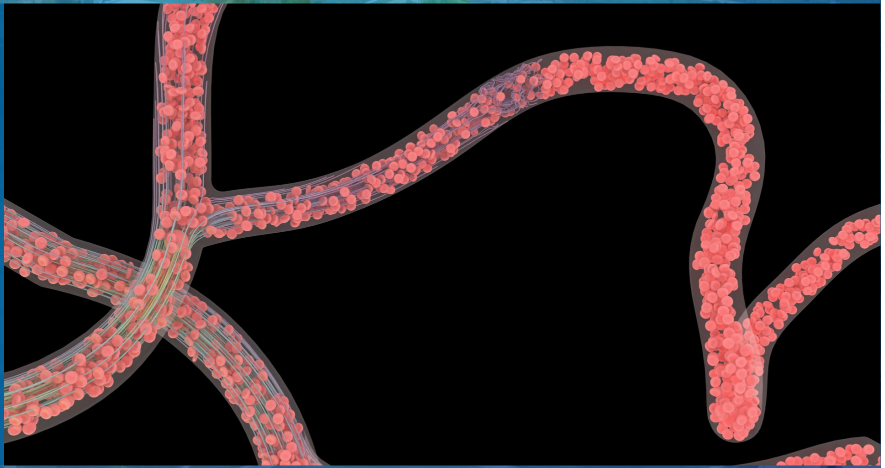
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344.
Lawrence Livermore National Security, LLC

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

Lawrence Livermore
National Laboratory

SC24 ANARI BOF- NOVEMBER 19, 2024

RENDERING AT WARP SPEED: OSPRAY + ANARI ON AURORA

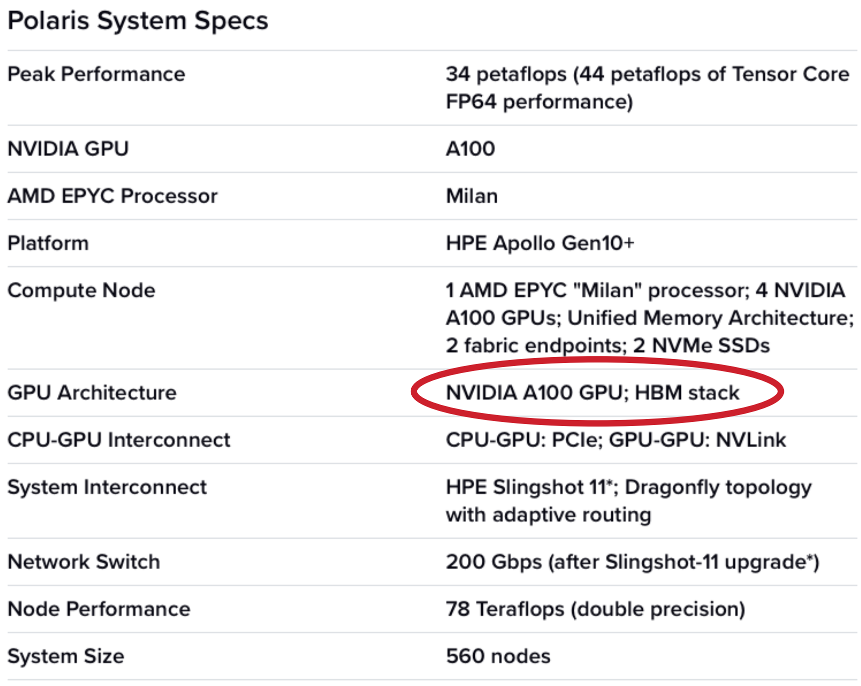


VICTOR MATEEVITSI
Assistant Computer Scientist
vmateevitsi@anl.gov

U.S. DEPARTMENT OF ENERGY
Argonne National Laboratory is a
U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC.

Argonne NATIONAL LABORATORY

POLARIS



Polaris System Specs

Peak Performance	34 petaflops (44 petaflops of Tensor Core FP64 performance)
NVIDIA GPU	A100
AMD EPYC Processor	Milan
Platform	HPE Apollo Gen10+
Compute Node	1 AMD EPYC "Milan" processor; 4 NVIDIA A100 GPUs; Unified Memory Architecture; 2 fabric endpoints; 2 NVMe SSDs
GPU Architecture	NVIDIA A100 GPU; HBM stack
CPU-GPU Interconnect	CPU-GPU: PCIe; GPU-GPU: NVLink
System Interconnect	HPE Slingshot 11*; Dragonfly topology with adaptive routing
Network Switch	200 Gbps (after Slingshot-11 upgrade*)
Node Performance	78 Teraflops (double precision)
System Size	560 nodes

AURORA

Top 500: #3 (1.02 ExaFlops)
AI Top 500: #1

Aurora System Specifications

Compute Node
2 Intel Xeon CPU Max Series processors: 64GB HBM on each, 512GB DDR5 each; 6 Intel Data Center GPU Max Series, 128GB HBM on each, RAMBO cache on each; Unified Memory Architecture; 8 SlingShot 11 fabric endpoints

Software Stack
HPE Cray EX supercomputer software stack + Intel enhancements + data and learning

GPU Architecture
6 Intel Data Center GPU Max Series; Tile-based chiplets, HBM stack, Foveros 3D integration, 7nm

CPU-GPU Interconnect
CPU-GPU: PCIe; GPU-GPU: Xe Link

System Interconnect
Slingshot 11; Dragonfly topology with adaptive routing; Peak Injection bandwidth 2.12 PB/s; Peak Bisection bandwidth 0.69 PB/s

Network Switch
25.6 Tb/s per switch, from 64–200 Gbs ports (25 GB/s per direction)

System Performance
Exascale

High-Performance Storage
230 PB, 31 TB/s, 1024 Nodes (DAOS)

Programming Models
Intel oneAPI, MPI, OpenMP, C/C++, Fortran, SYCL/DPC++

Platform
HPE Cray EX supercomputer

Aggregate System Memory
20.4 PB

System Size
10,624 nodes

CRUX

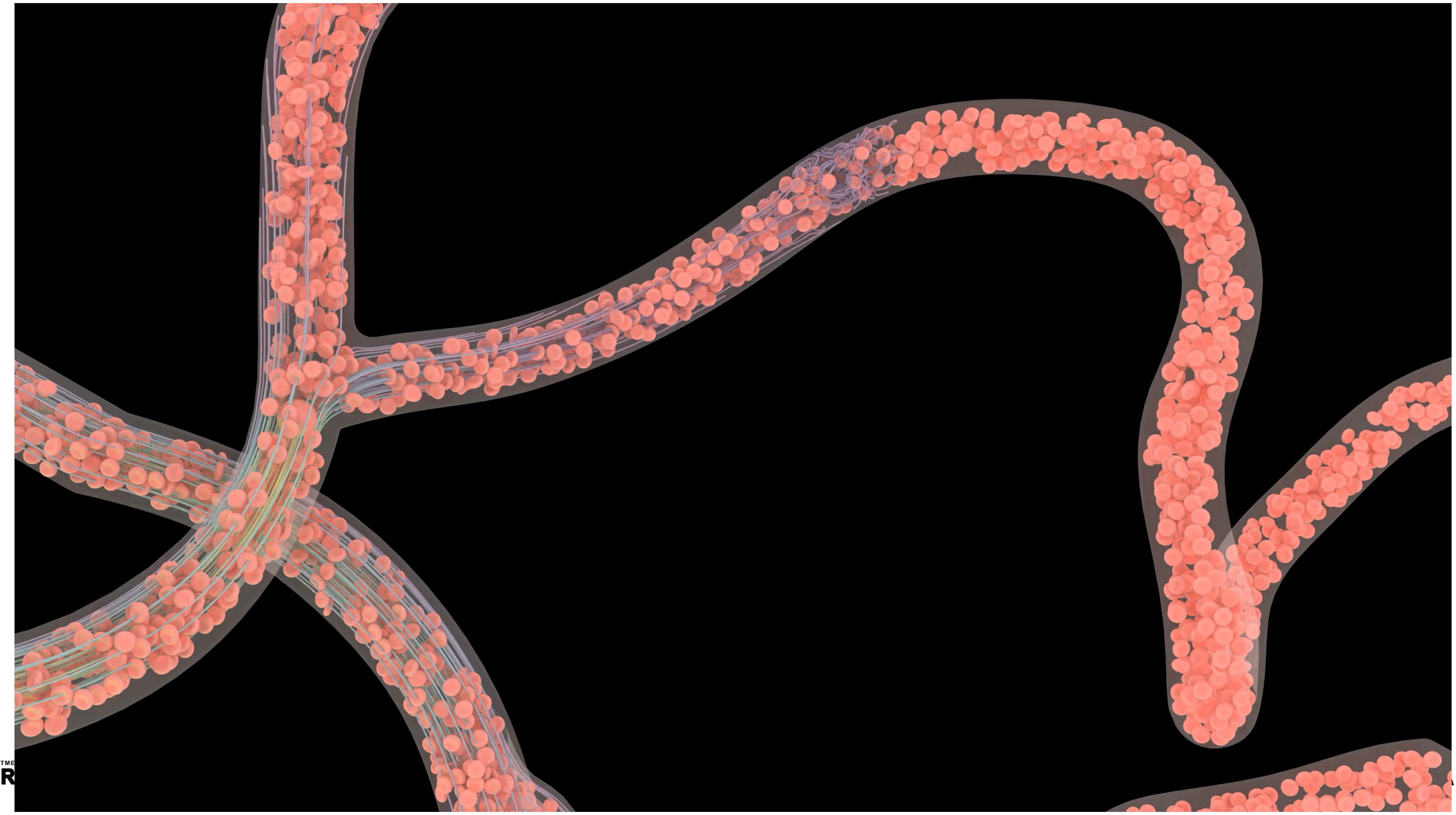
Crux, a CPU only resource (no GPUs or other accelerators), will be coming online soon.

WHY ANARI?

- Support for heterogeneous resources
- Same visualization/rendering code works across resources
- Users can select their preferred rendered
- Users are not “locked” with a specific renderer

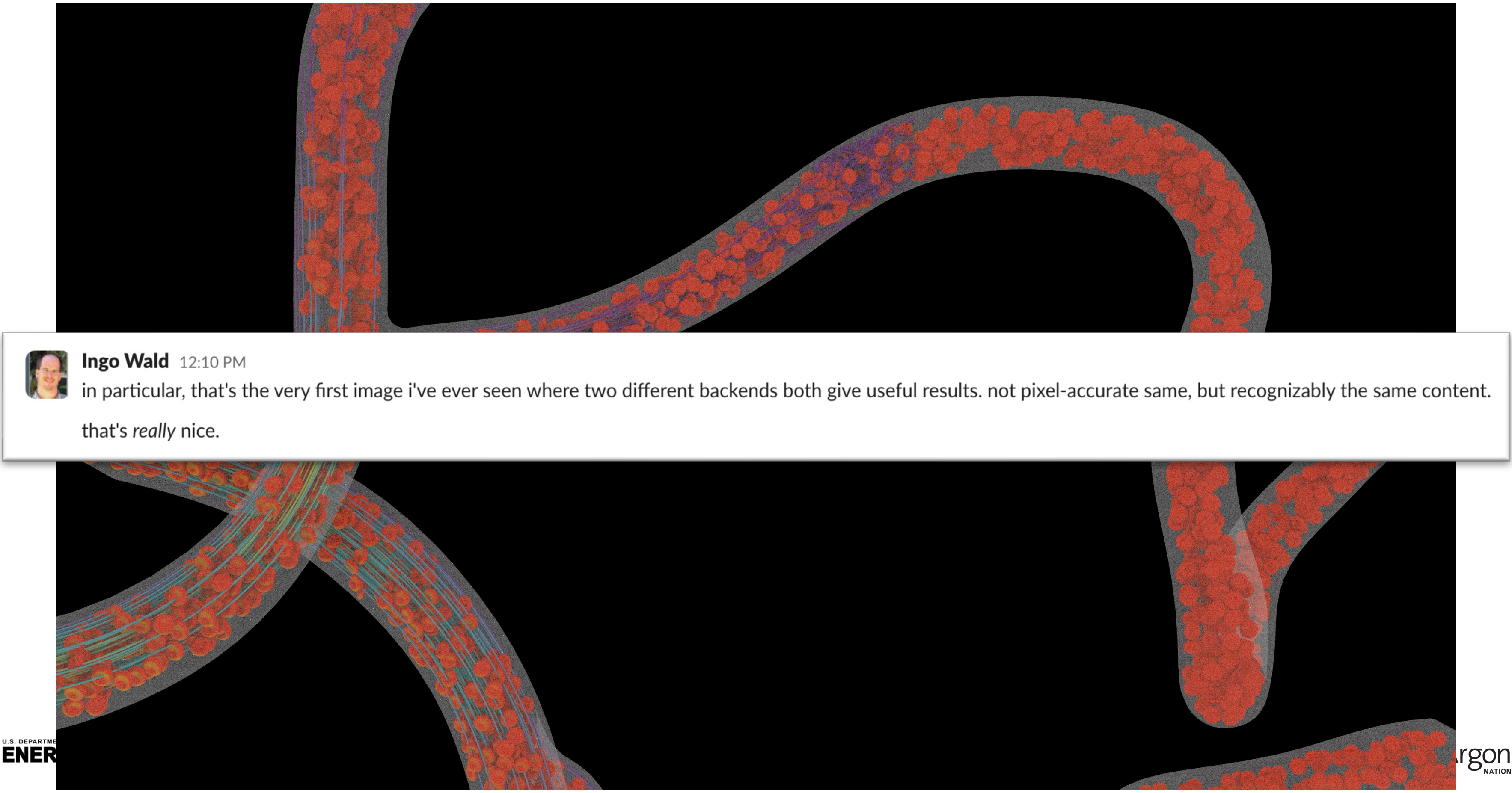
DEMO TIME!

Red blood cell transport visualized in complex human vasculature using HARVEY, with velocity streamlines illustrating flow dynamics. This simulation, conducted by Ayman Yousef, highlights advances in personalized blood flow modeling. From the Randles Lab at Duke University as a part of the Argonne AESP program. Rendered with OSPRay via ANARI on Aurora.



ONE MORE THING!

Red blood cell transport visualized in complex human vasculature using HARVEY, with velocity streamlines illustrating flow dynamics. This simulation, conducted by Ayman Yousef, highlights advances in personalized blood flow modeling. From the Randles Lab at Duke University as a part of the Argonne AESP program. Rendered with Barney via ANARI on Polaris.



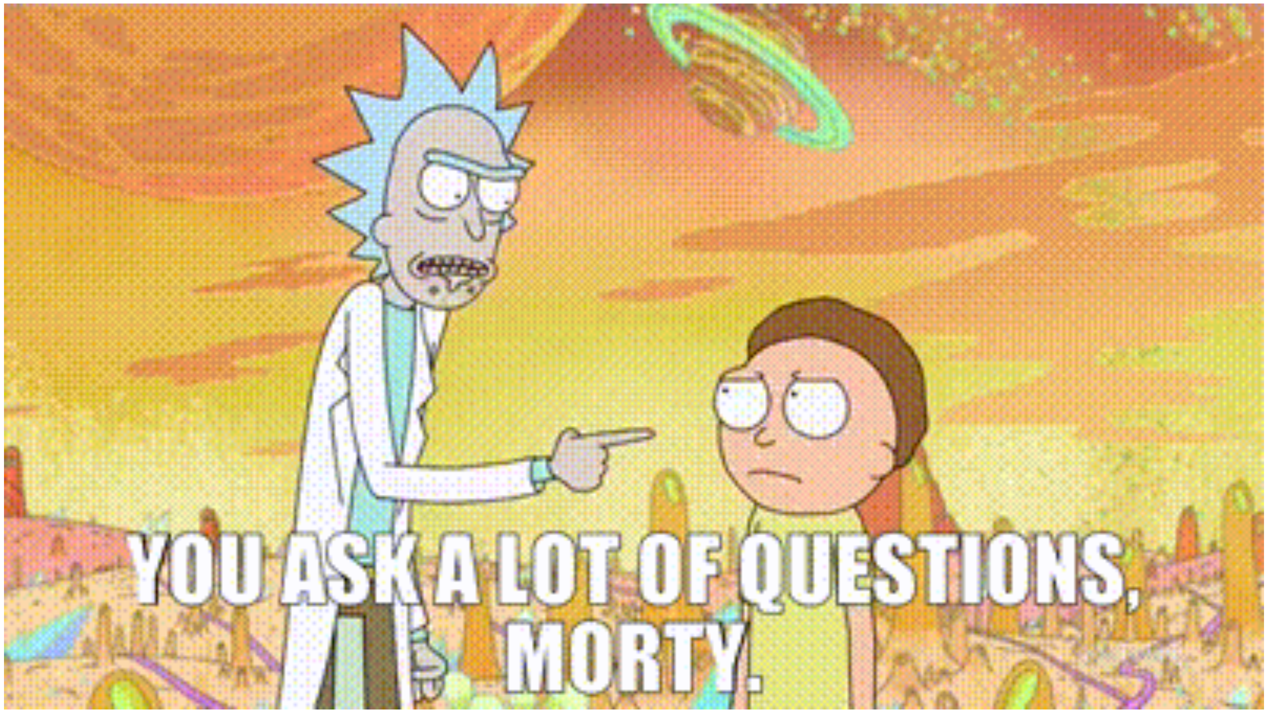
Ingo Wald 12:10 PM

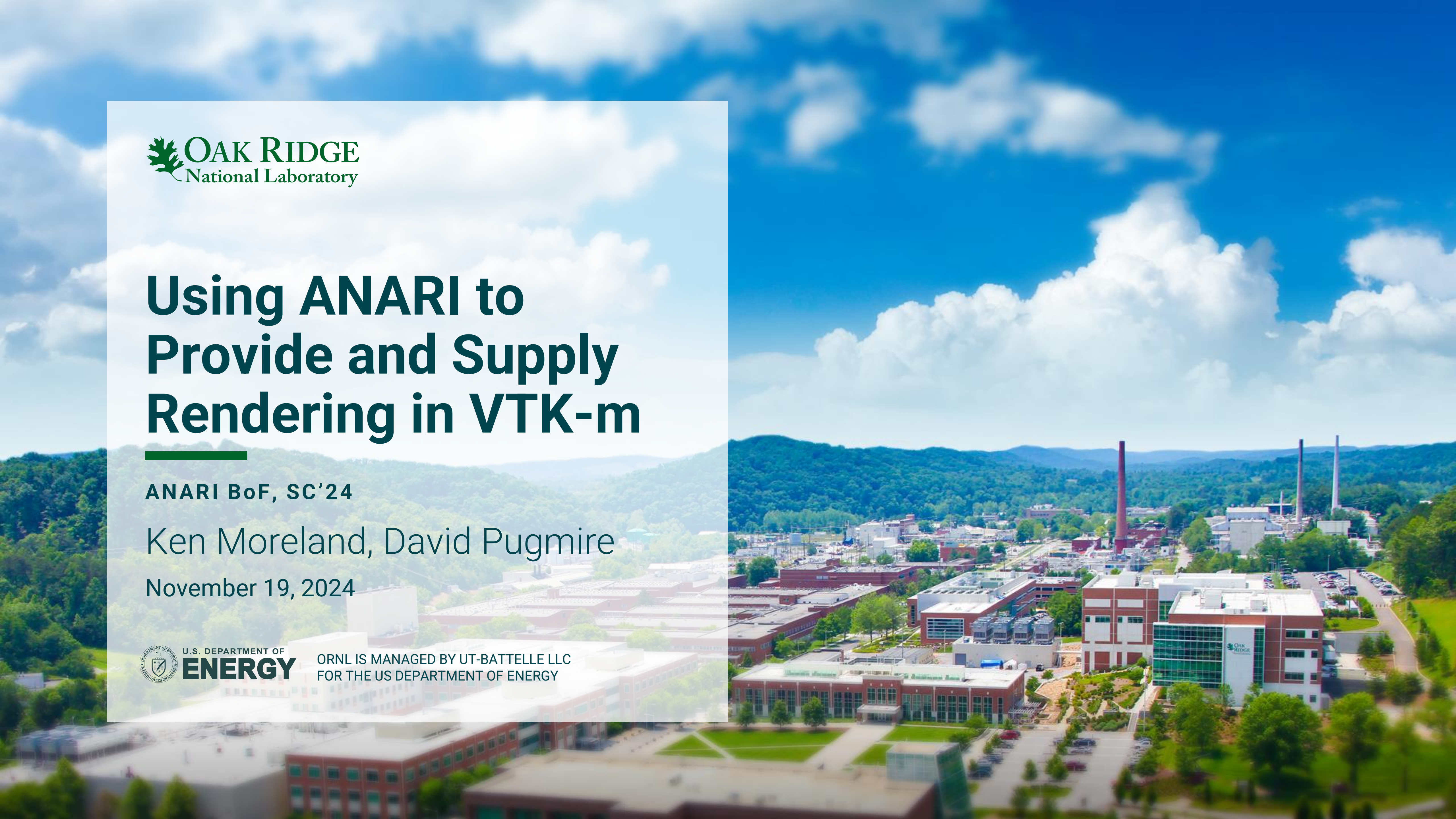
in particular, that's the very first image i've ever seen where two different backends both give useful results. not pixel-accurate same, but recognizably the same content.

that's *really* nice.

QUESTIONS?

Thank you!





OAK RIDGE
National Laboratory

Using ANARI to Provide and Supply Rendering in VTK-m

ANARI BoF, SC’24

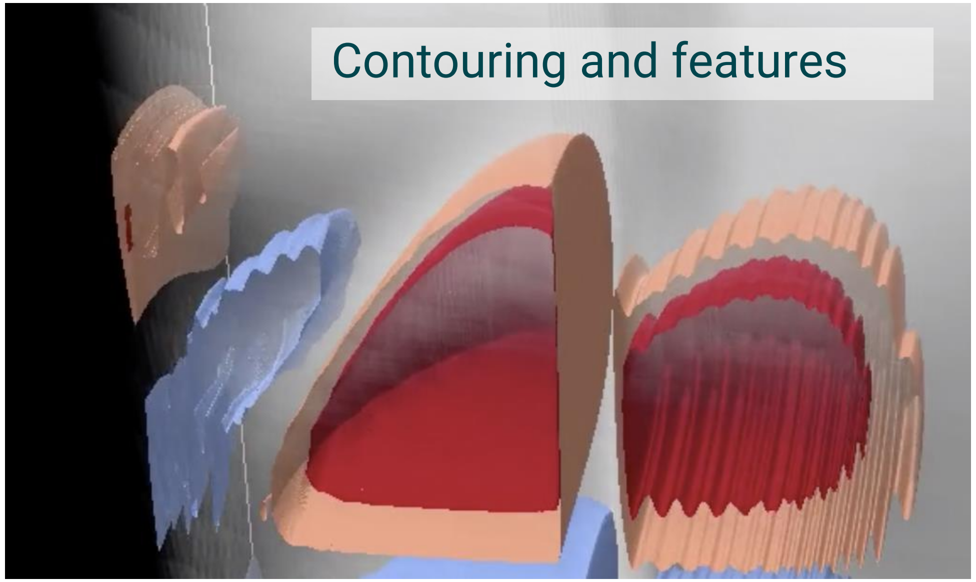
Ken Moreland, David Pugmire

November 19, 2024

U.S. DEPARTMENT OF ENERGY

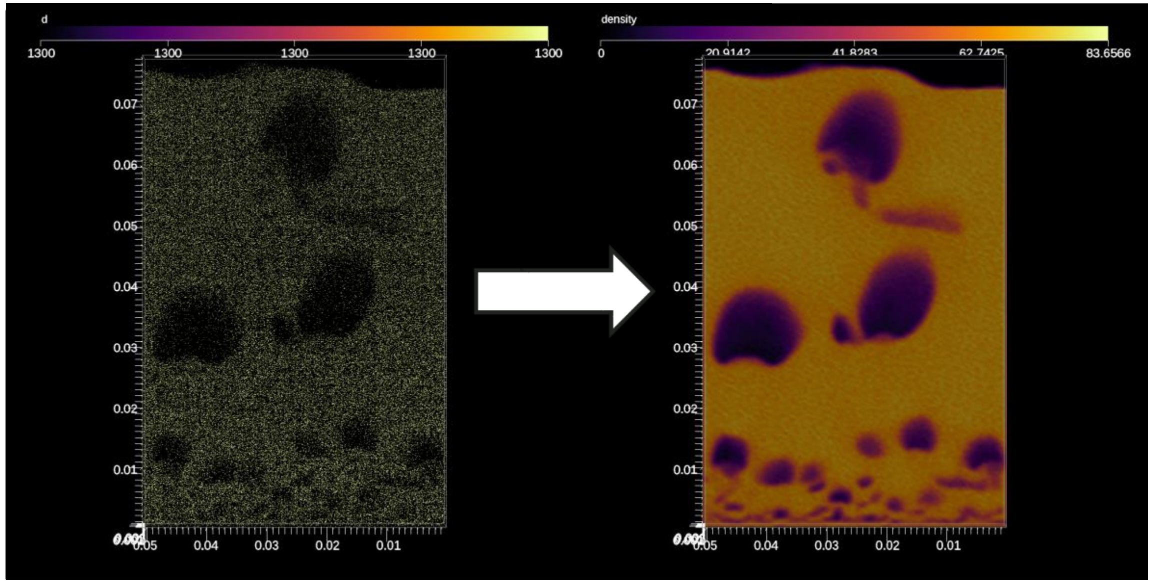
ORNL IS MANAGED BY UT-BATTELLE LLC
FOR THE US DEPARTMENT OF ENERGY

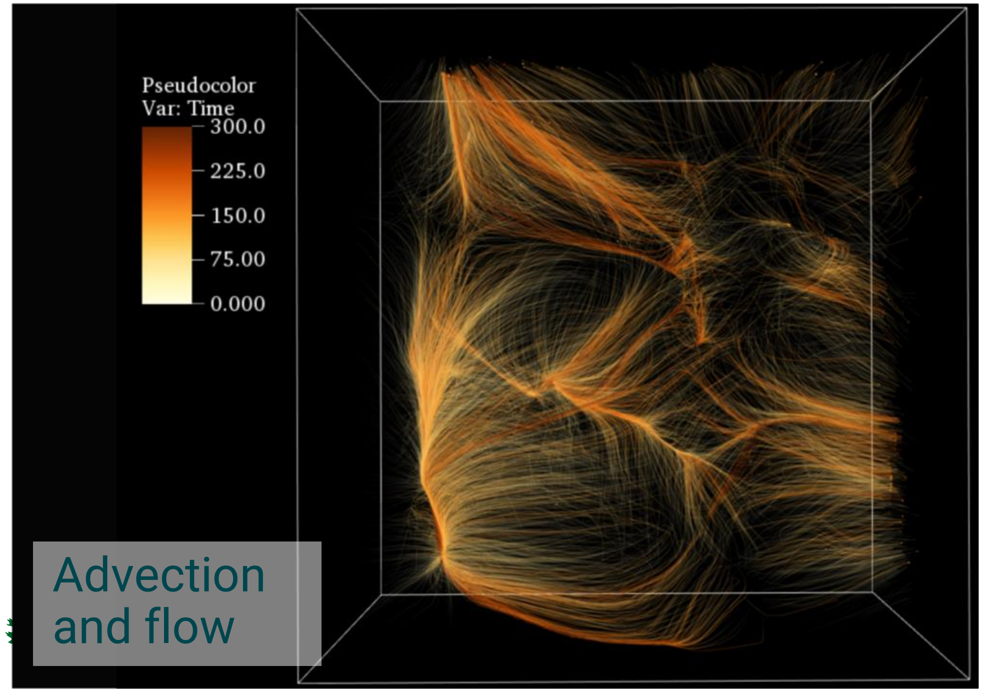
VTK-m: Visualization on Accelerators



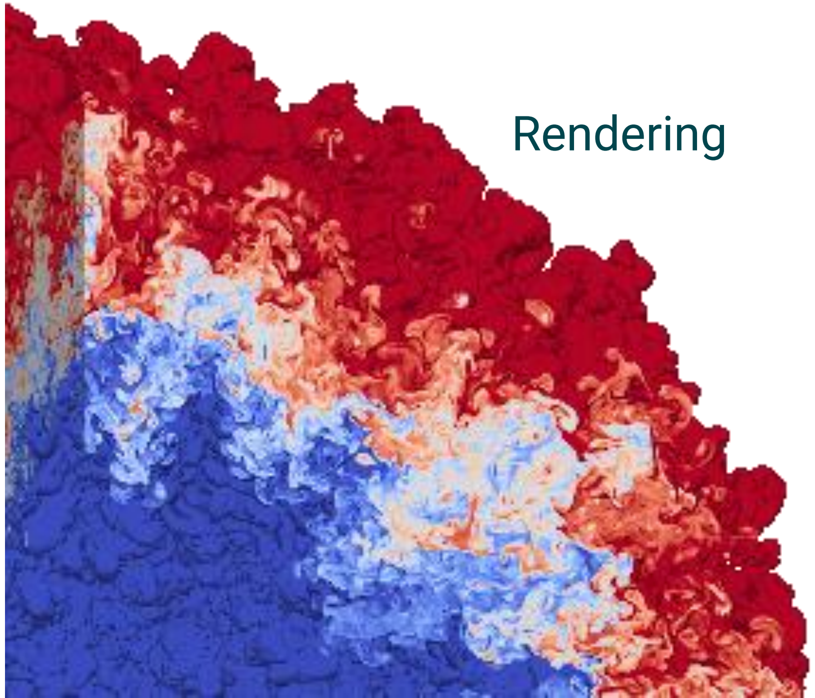
Contouring and features

Particle Density



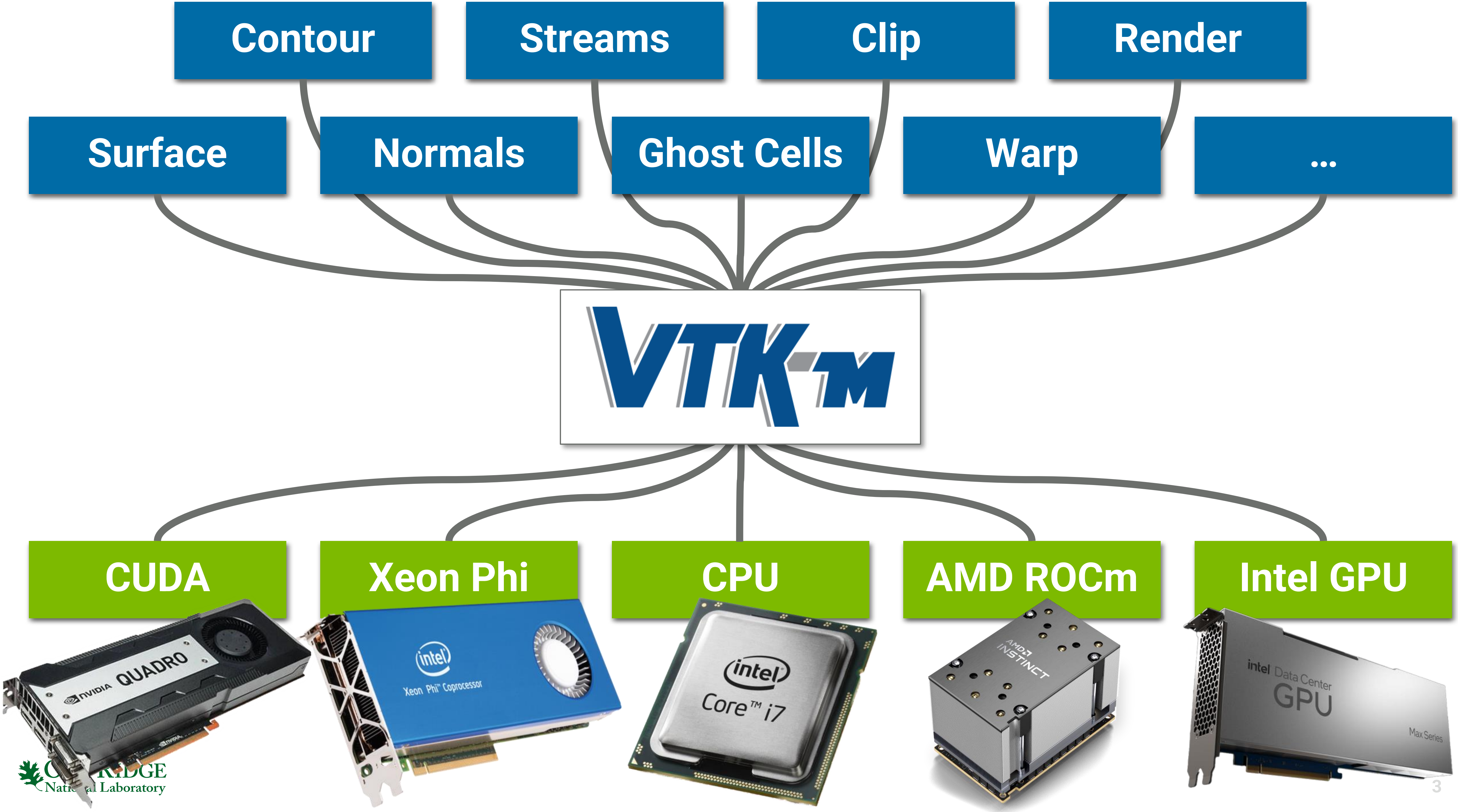


Advection and flow



Rendering

And much more…



Contour

Streams

Clip

Render

Surface

Normals

Ghost Cells

Warp

...

VTK-m

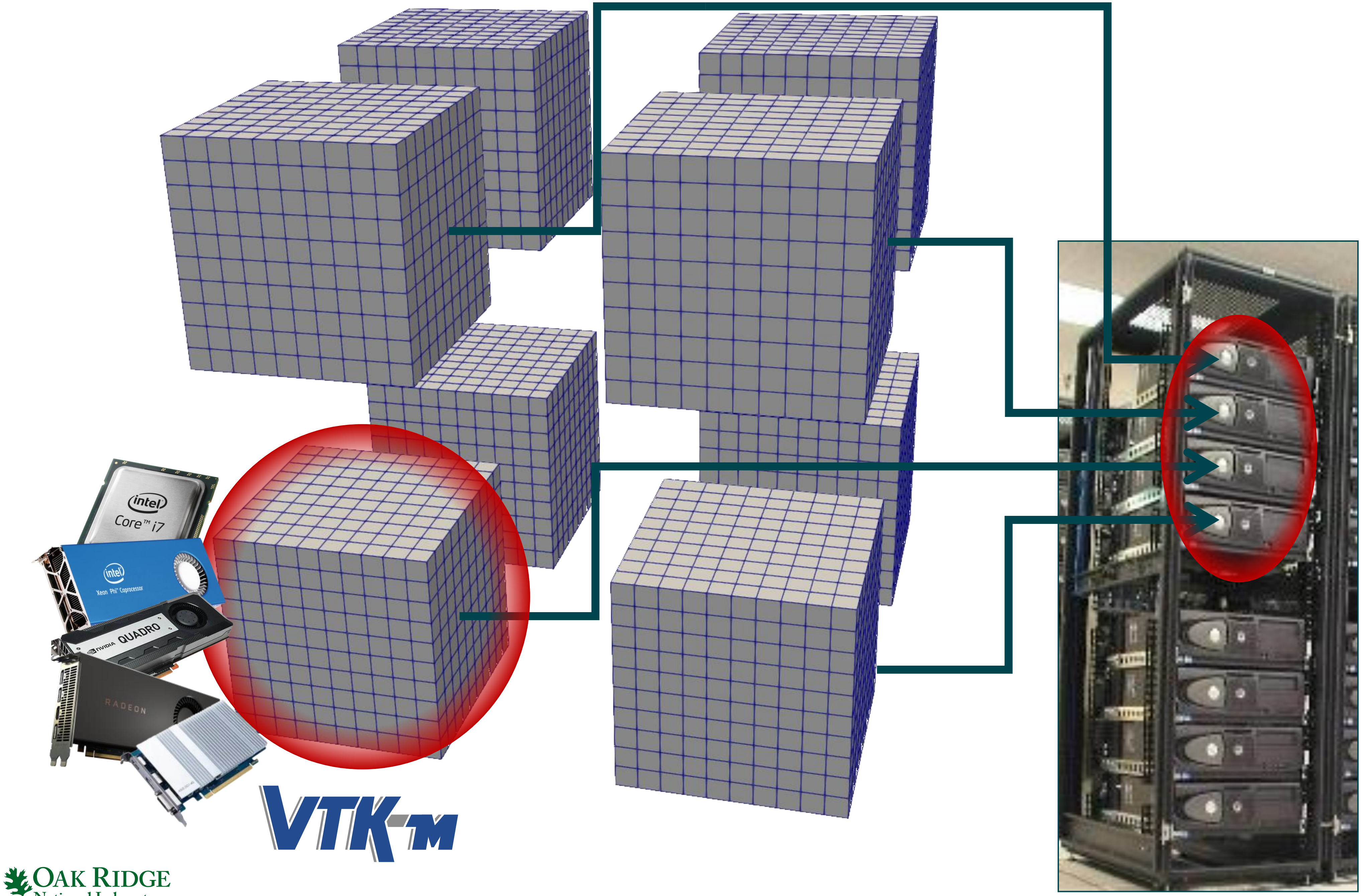
CUDA

Xeon Phi

CPU

AMD ROCm

Intel GPU



Distributed Parallelism

ParaView

VisIt

Ascent

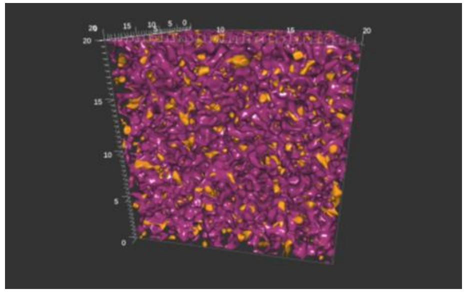
Oak Ridge National Laboratory

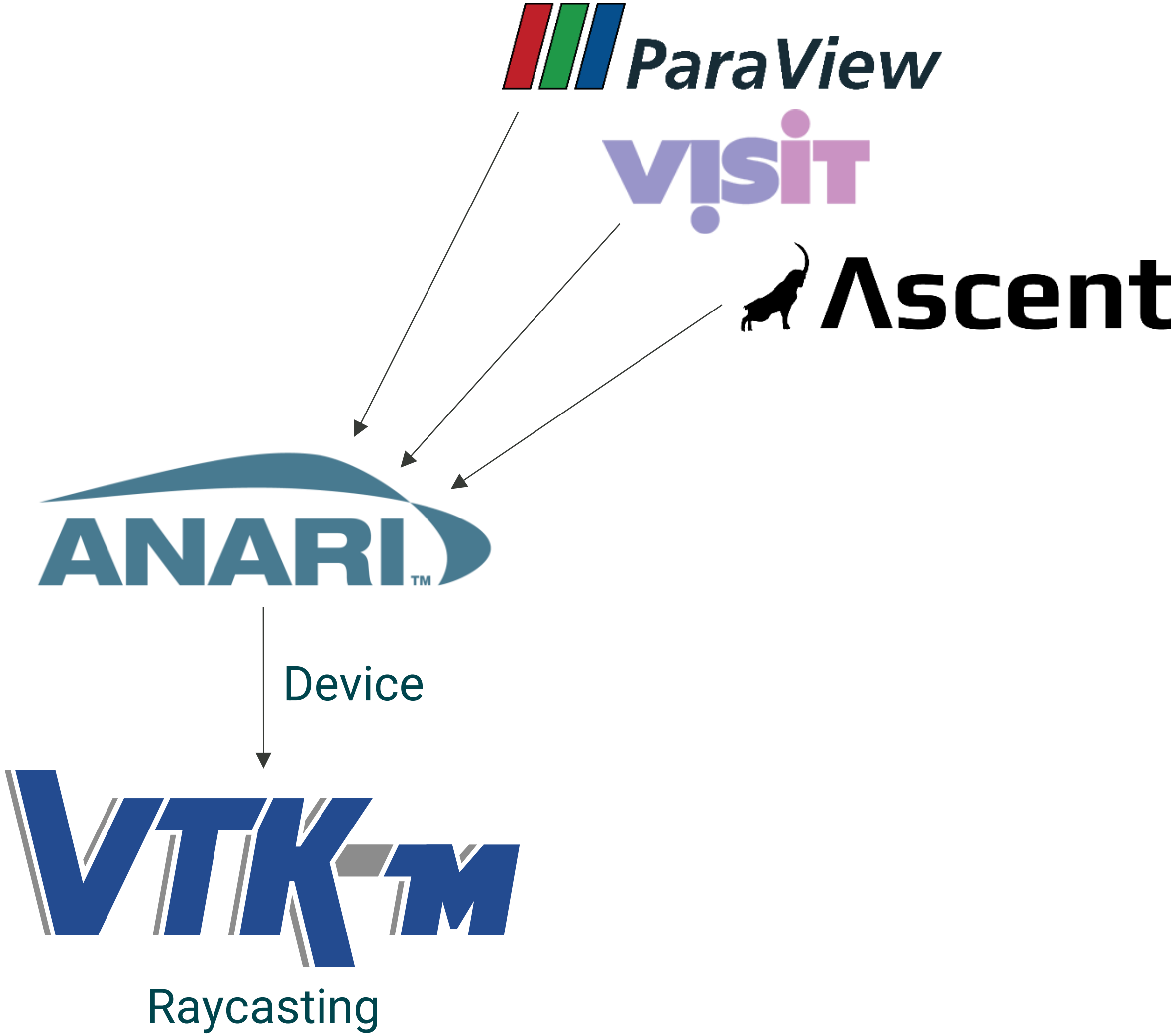
ANARI™

Device

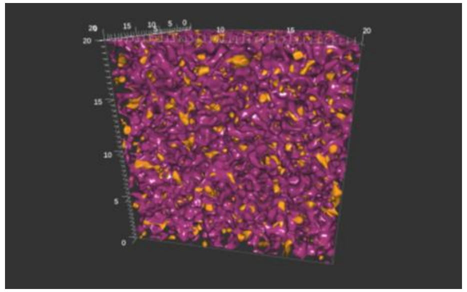
VTK-m

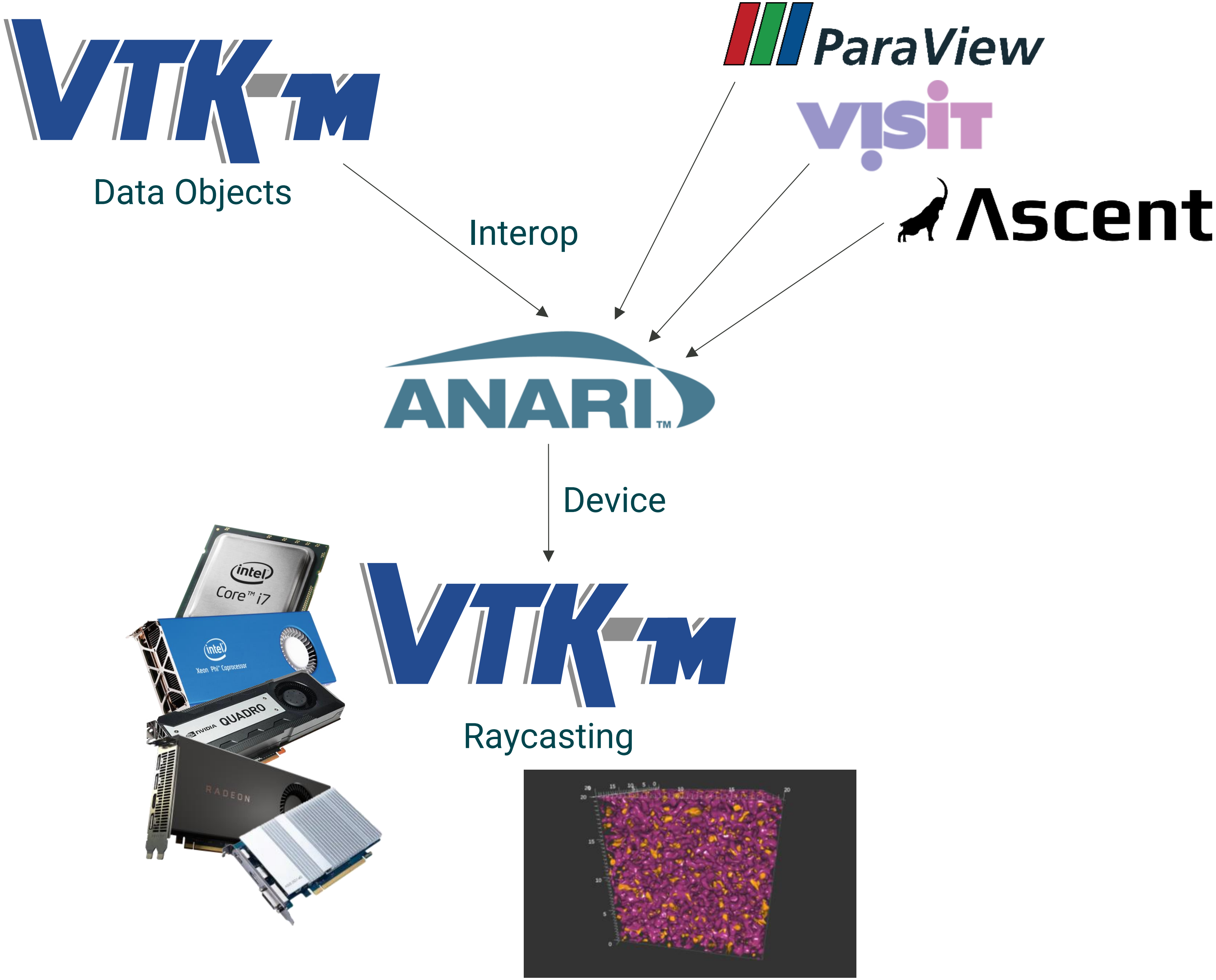
Raycasting




ParaView
VisIt
Ascent
ANARI™
Device
VTK-m
Raycasting

intel
Core™ i7
intel
Xeon Phi™ Coprocessor
NVIDIA QUADRO
RADEON



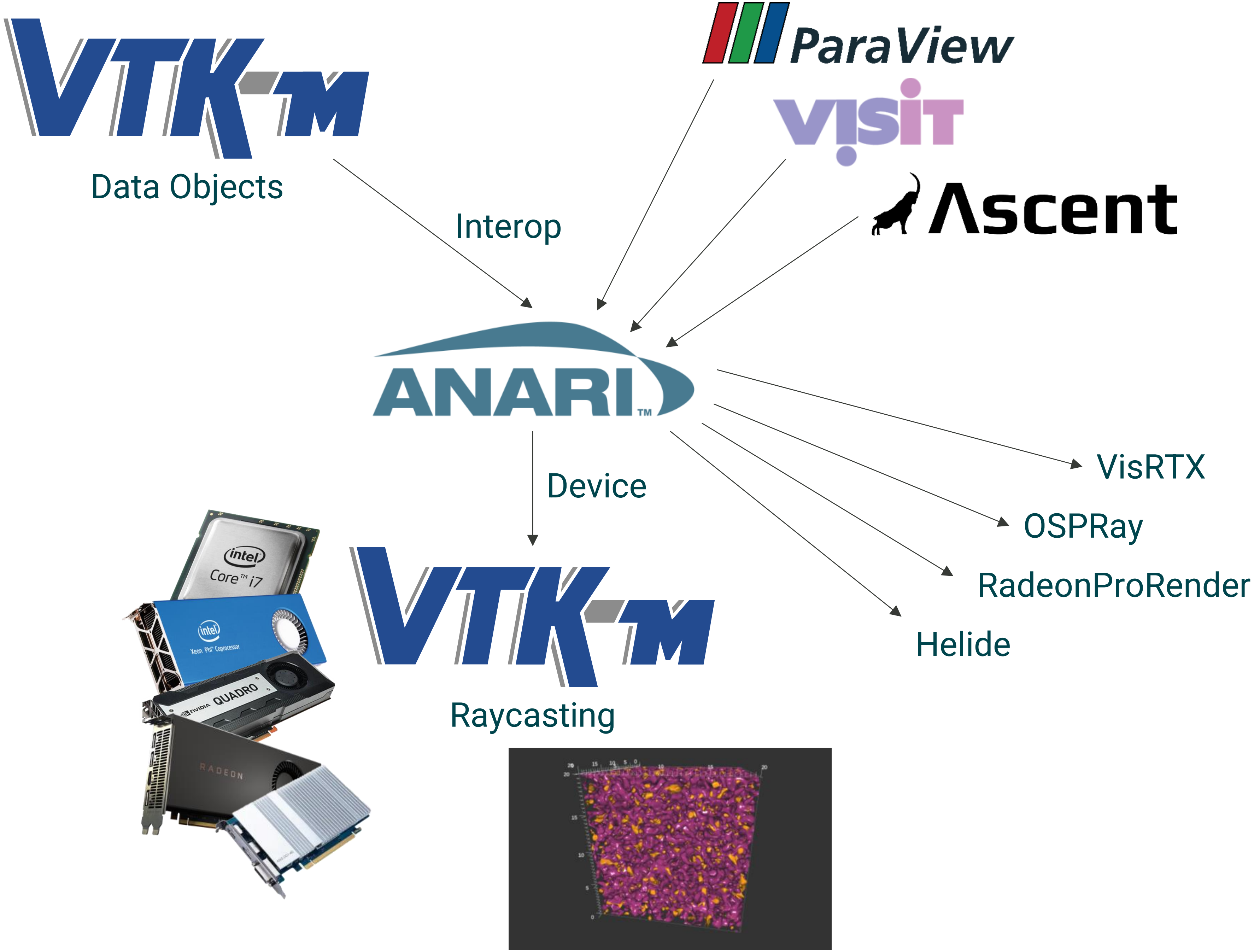


Data Objects

Interop

Device

Raycasting



Data Objects

Interop

Device

Raycasting

VisRTX

OSPRay

RadeonProRender

Helide

NATIONAL INSTITUTE OF
STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

Integrating ANARI into Virtual Reality

William Sherman

SC 2024: ANARI BOF

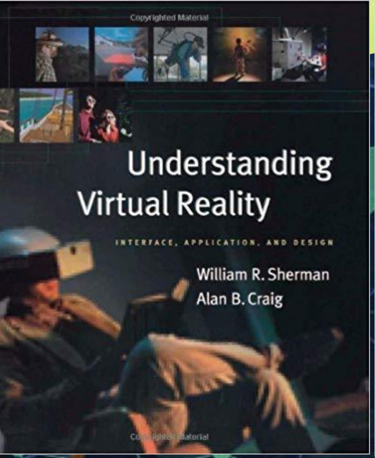
November 19, 2024

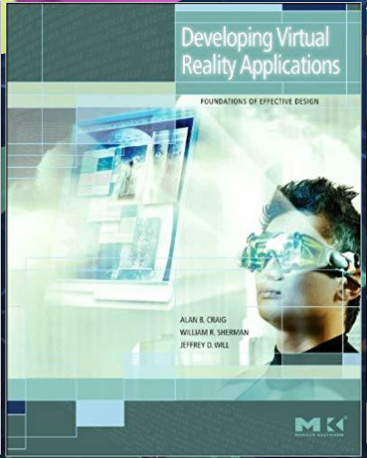
William Sherman

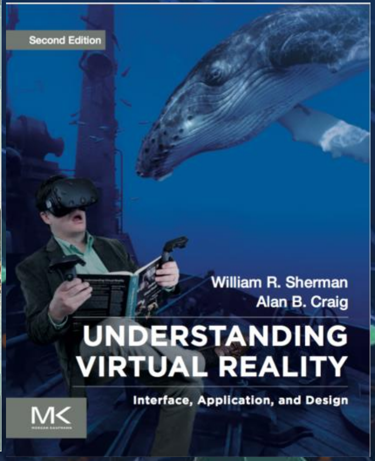
NIST

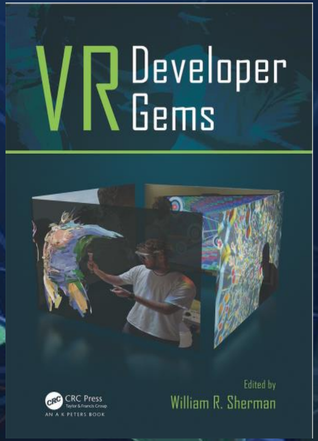
- 33 years in virtual reality

  (plus 6 more in CG)
- M.S. UIUC, 1989
- NIST Computer Scientist (presently)
- Indiana University (previously)
- Desert Research Institute
- National Center for Supercomputing Appl.
- Teaching “Introduction to Virtual Reality” off and on since 2000 (about 10 times)
- Co-author/Editor of 4 books on VR
- PhD UIUC, underway




Developing Virtual Reality Applications
FOUNDATIONS OF EFFECTIVE DESIGN
ALAN B. CRAIG
WILLIAM R. SHERMAN
JEFFREY D. WILL
MK

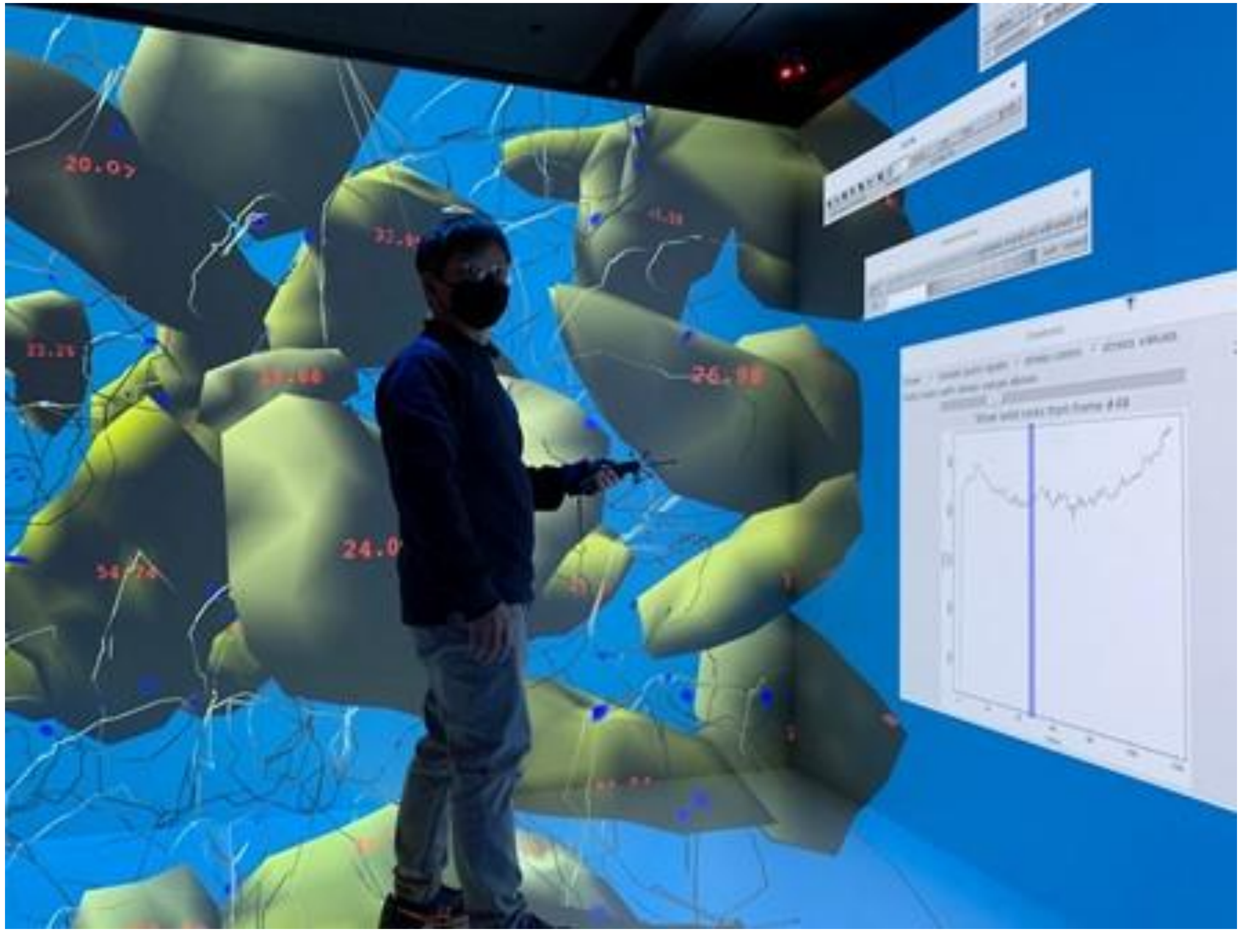

Second Edition
William R. Sherman
Alan B. Craig
UNDERSTANDING VIRTUAL REALITY
Interface, Application, and Design
MK



High Performance Computing and **Visualization Group**

Immersive Technologies to advance measurement science, standards, and technology.

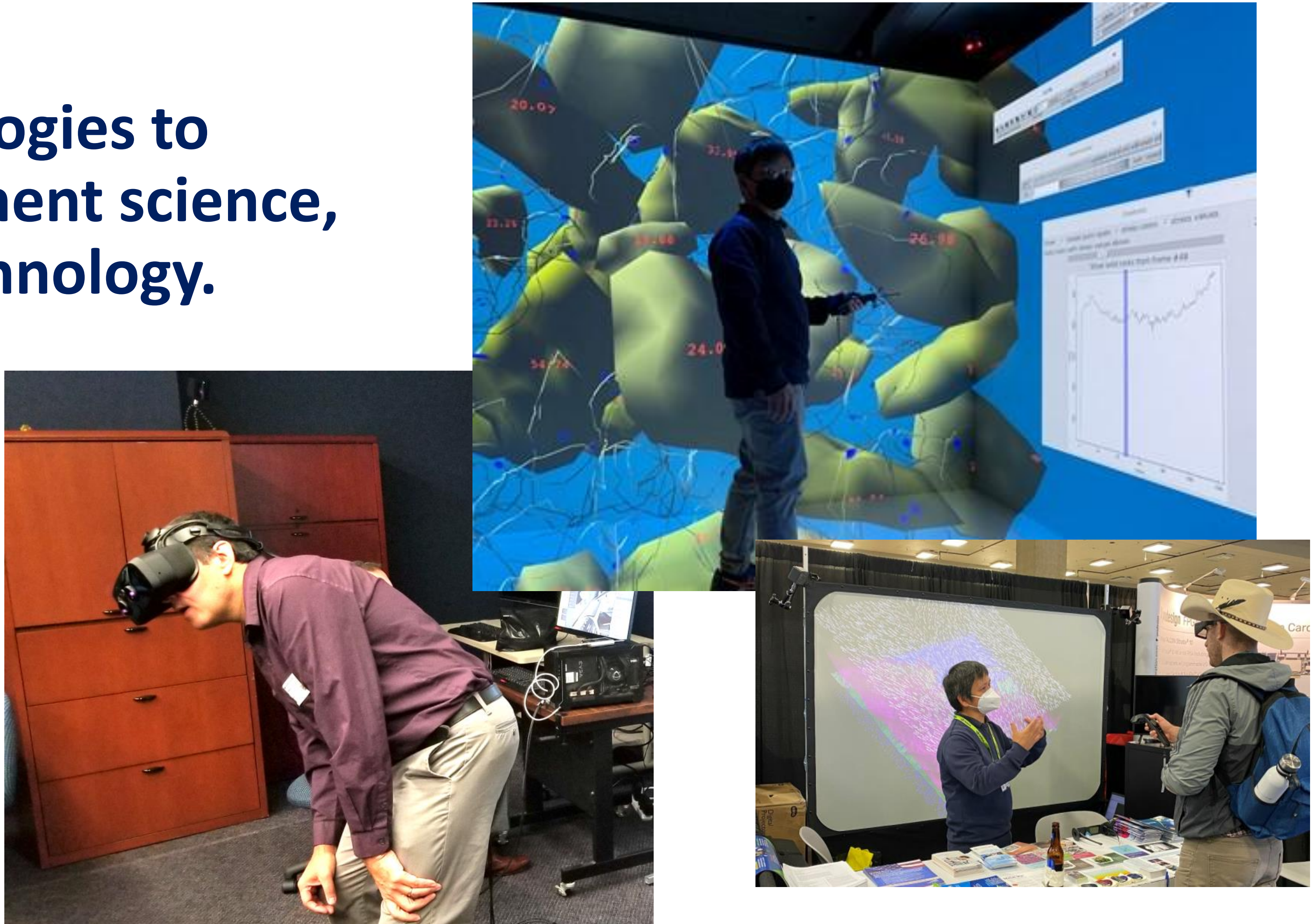
Software standards to broaden the impact of visualization and immersive systems.



Immersive Technologies to advance measurement science, standards, and technology.

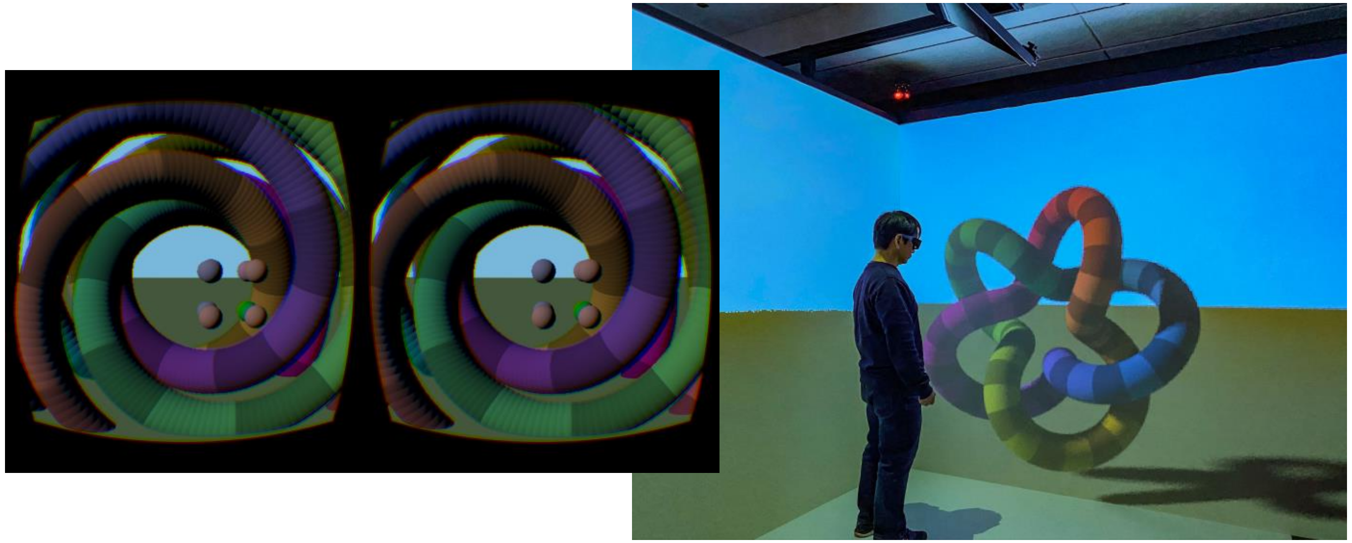
Visualization Displays:

- CAVE (3-sided)
- HMDs (several)
  - Vives
  - Quests
  - Index
  - Vario X3
- IQ-station (mini-CAVE)



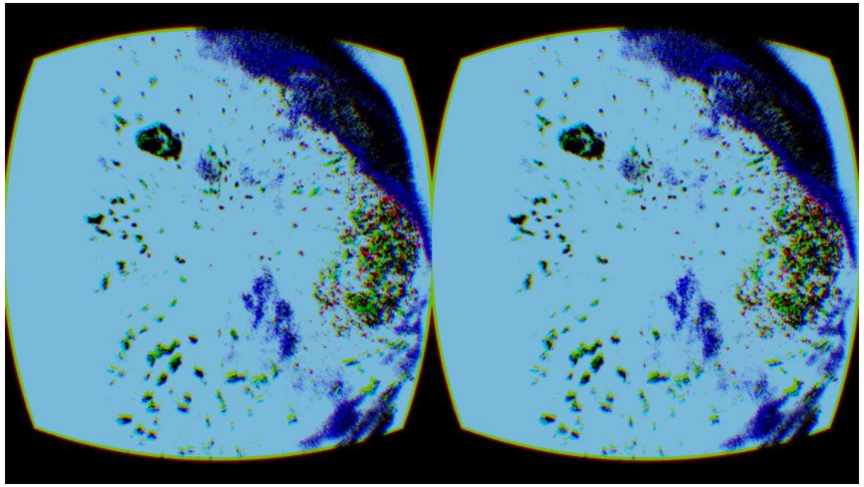
ANARI and VR

ANARI VR Implementations



My Implementations:

- CAVE-style with FreeVR (freevr.org)
- HMD with ILLIXR (illixr.org)
- HMD directly to OpenXR
- HMD to OpenXR/OpenVR via VTK

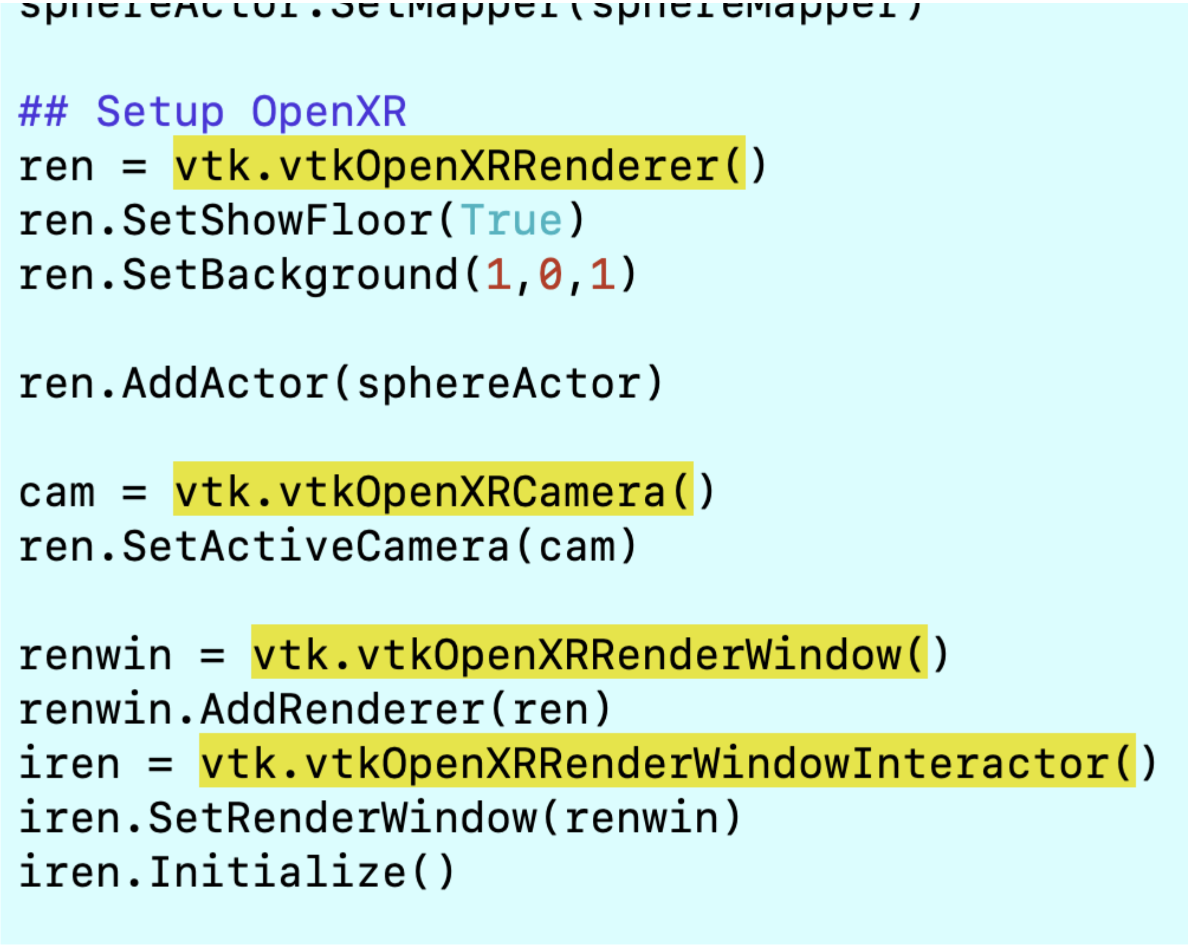


Other Implementations

- CAVE-style using COVISE (Zellmann & Wössner)

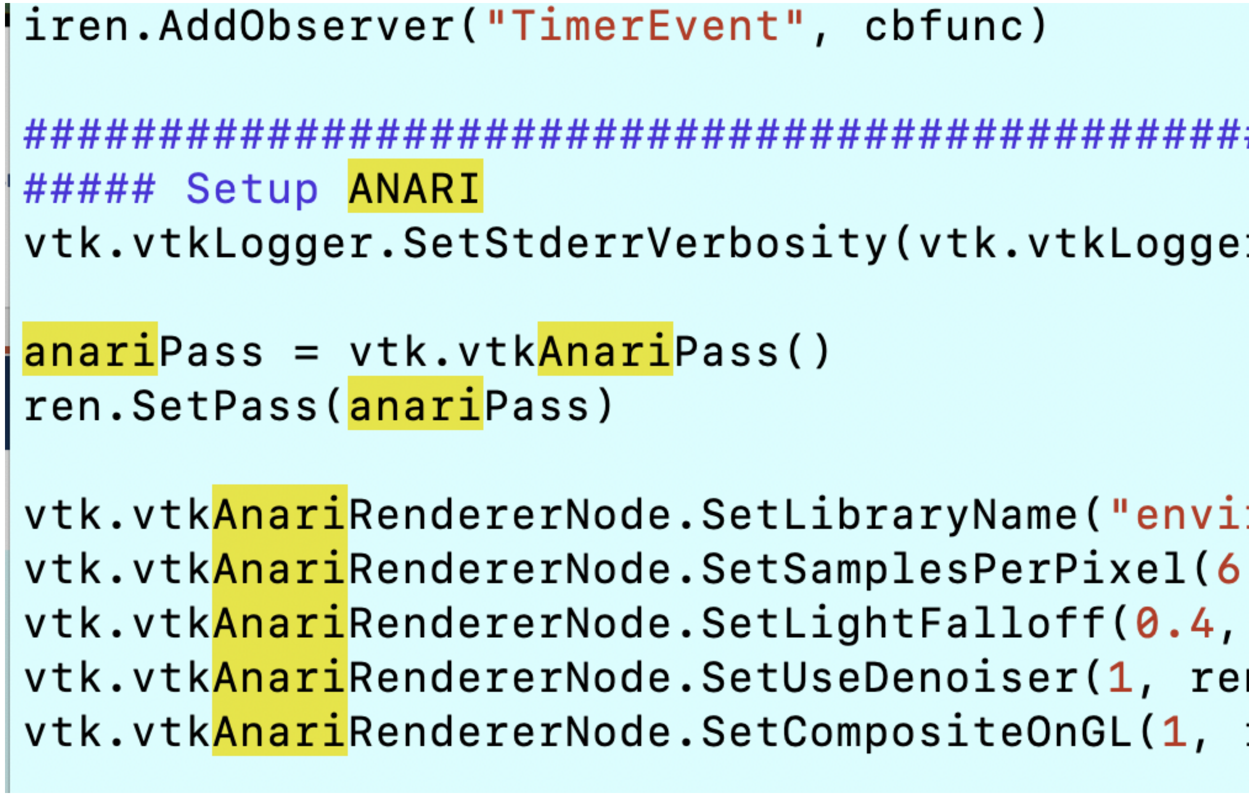
ANARI VR Implementations

OpenXR in VTK



```
sphereActor.SetMapper(sphereMapper)

## Setup OpenXR
ren = vtk.vtkOpenXRRenderer()
ren.SetShowFloor(True)
ren.SetBackground(1,0,1)

ren.AddActor(sphereActor)

cam = vtk.vtkOpenXRCamera()
ren.SetActiveCamera(cam)

renwin = vtk.vtkOpenXRRenderWindow()
renwin.AddRenderer(ren)
iren = vtk.vtkOpenXRRenderWindowInteractor()
iren.SetRenderWindow(renwin)
iren.Initialize()
```

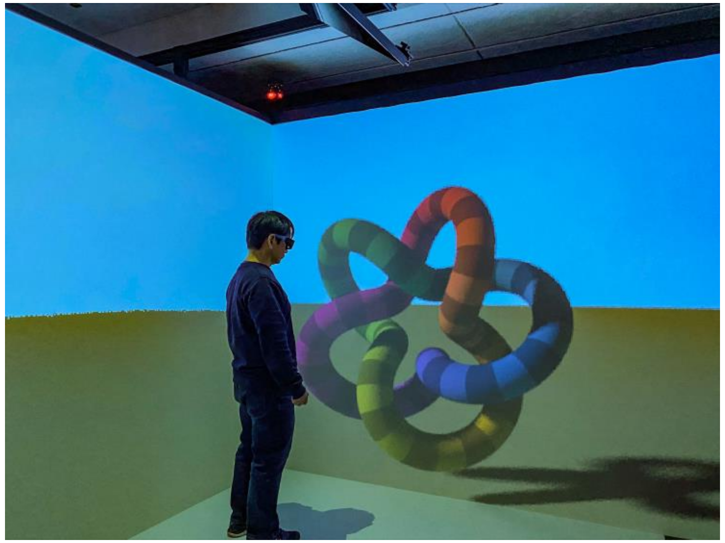
ANARI in VTK

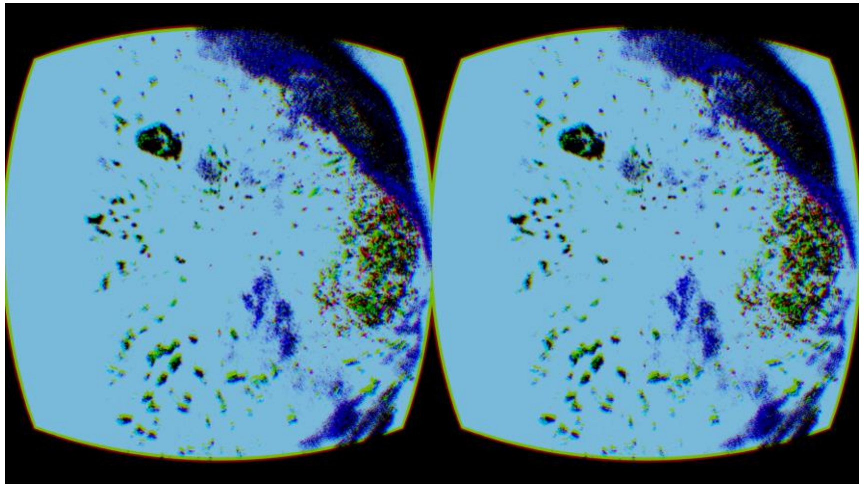


```
iren.AddObserver("TimerEvent", cbfunc)

##############################################
##### Setup ANARI
vtk.vtkLogger.SetStderrVerbosity(vtk.vtkLogge

anariPass = vtk.vtkAnariPass()
ren.SetPass(anariPass)

vtk.vtkAnariRendererNode.SetLibraryName("envi
vtk.vtkAnariRendererNode.SetSamplesPerPixel(6
vtk.vtkAnariRendererNode.SetLightFalloff(0.4,
vtk.vtkAnariRendererNode.SetUseDenoiser(1, re
vtk.vtkAnariRendererNode.SetCompositeOnGL(1, 
```

ANARI Spec and VR

- Spec permits a stereoscopic camera
  - ➢ perhaps only OSPRay backend presently provides one
- Renders to a rectangular frame buffer
- Doesn’t specify memory management
  - ➢ Watch for simultaneously rendering the scene to multiple views
  - ➢ (Thus far my implementations render sequentially)

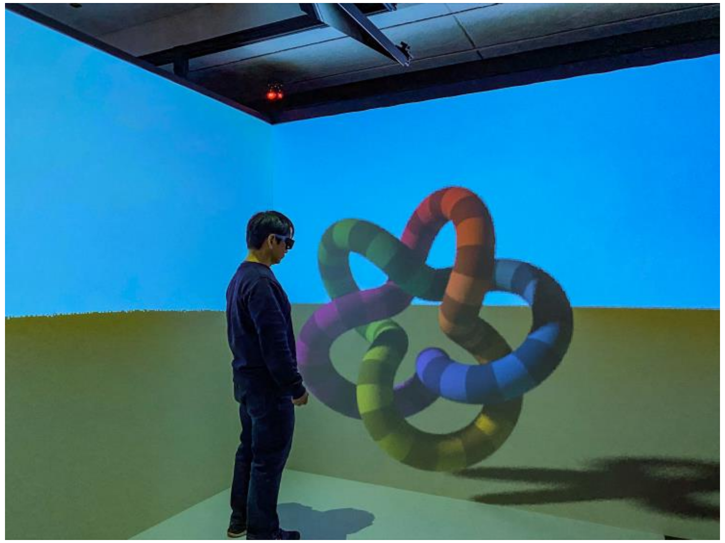


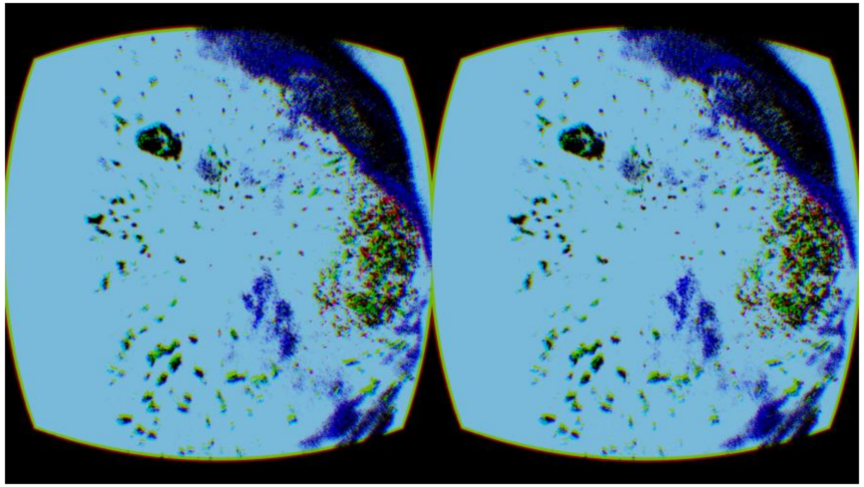


ANARI Spec and VR

ANARI rendering to a frame buffer

- Thus any warping would still be handled by the runtime
- Rendering to CAVE-style displays can use the tiled-window feature to render an off-axis view that can be copied directly to a screen's surface

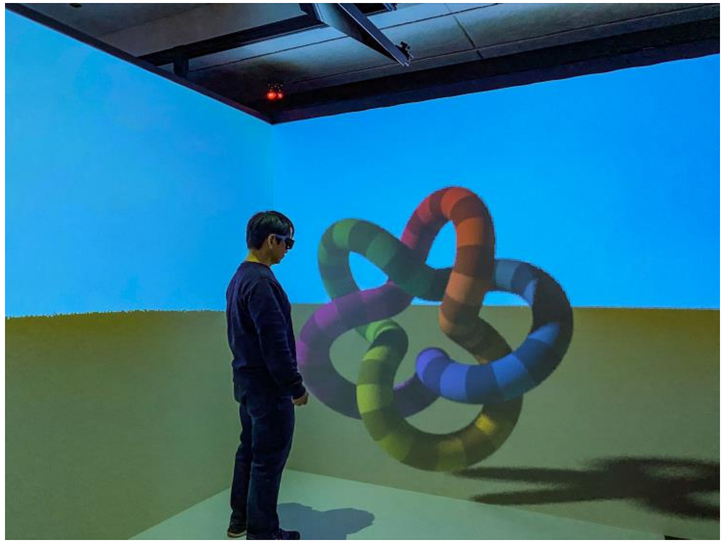


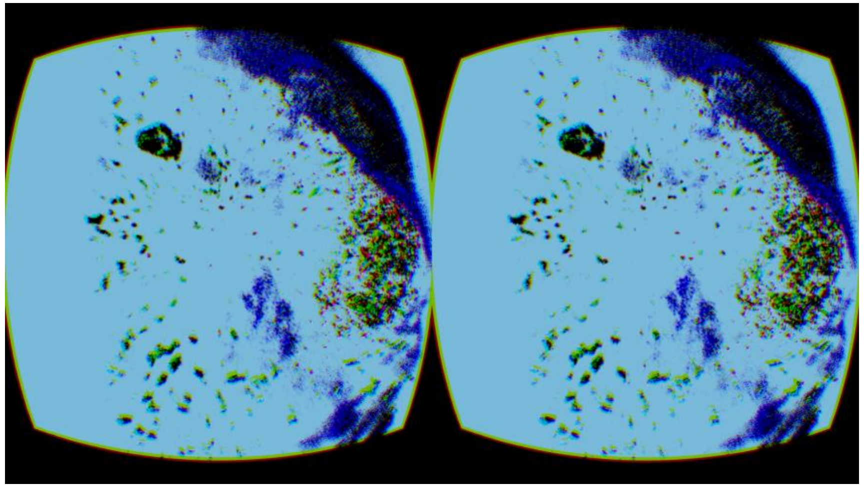


Future plans

ANARI rendering to a frame buffer

- Implement a parallel rendering mechanism
- Explore the use of stereoscopic camera implementations
  - ➢ (for the backends that do implement it)
- Investigate the possibility of Omni-directional cameras

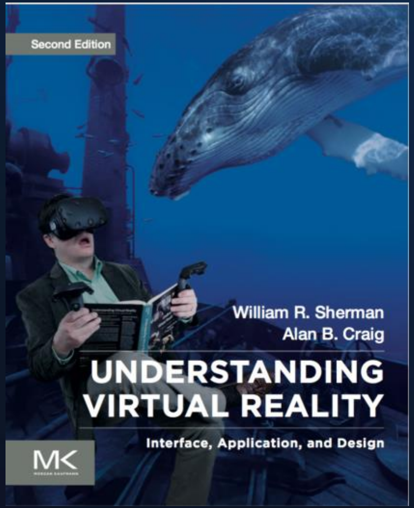




Thank you

Integrating ANARI into Virtual Reality

William Sherman
National Institute of Standards and Technology


Second Edition
William R. Sherman
Alan B. Craig
UNDERSTANDING VIRTUAL REALITY
Interface, Application, and Design
MK