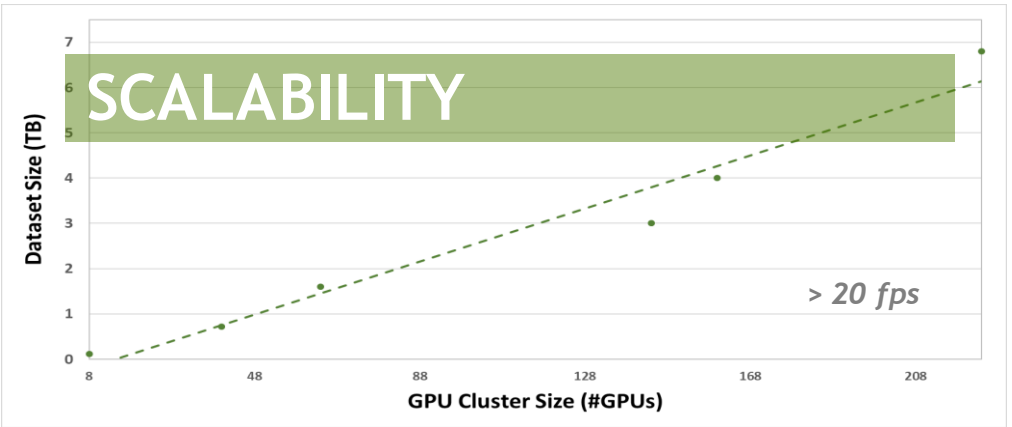
NVIDIA INDEX

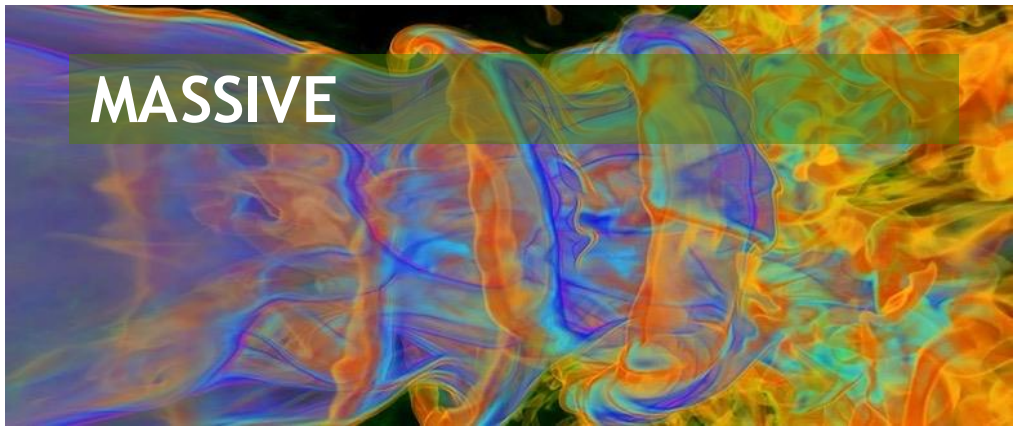
IMPLEMENTING CLOUD SERVICES FOR MASSIVE DATA VISUALIZATION

Marc Nienhaus (NVIDIA), Tom-Michael Thamm (NVIDIA), and Brant Robertson (IAS, UC Santa Cruz)

NVIDIA INDEX

Analyze Large-Scale Data for Faster Discoveries





NVIDIA INDEX IS FOR …

DIRECT CUSTOMERS	INDEPENDENT SOFTWARE VENDORS (ISVs)	CLOUDS AND DATACENTERS	HPC RESEARCH COMMUNITY
NVIDIA IndeX SDK to power in-house software solutions with specific workflows. NVIDIA IndeX plugin for ParaView, cluster version. Commercial license.	NVIDIA IndeX SDK to power enterprise products. Commercial license.	NVIDIA IndeX SDK to power new cloud services and datacenter solutions. Commercial license.	NVIDIA IndeX plugin for ParaView. Accelerate researchers' science and discovery processes. Non-commercial use.

NVIDIA INDEX IS AVAILABLE …

NGC CONTAINER	NVONLINE	KITWARE
NVIDIA IndeX SDK and NVIDIA IndeX for ParaView Plugin.	NVIDIA IndeX SDK.	NVIDIA IndeX plugin for ParaView.
https://ngc.nvidia.com	*http://partners.nvidia.com* *http://nvdeveloper.nvidia.com*	*http://www.paraview.org*

GALACTIC WINDS

NVIDIA IndeX and Cholla

Supercomputing 2018

7 TB volume data

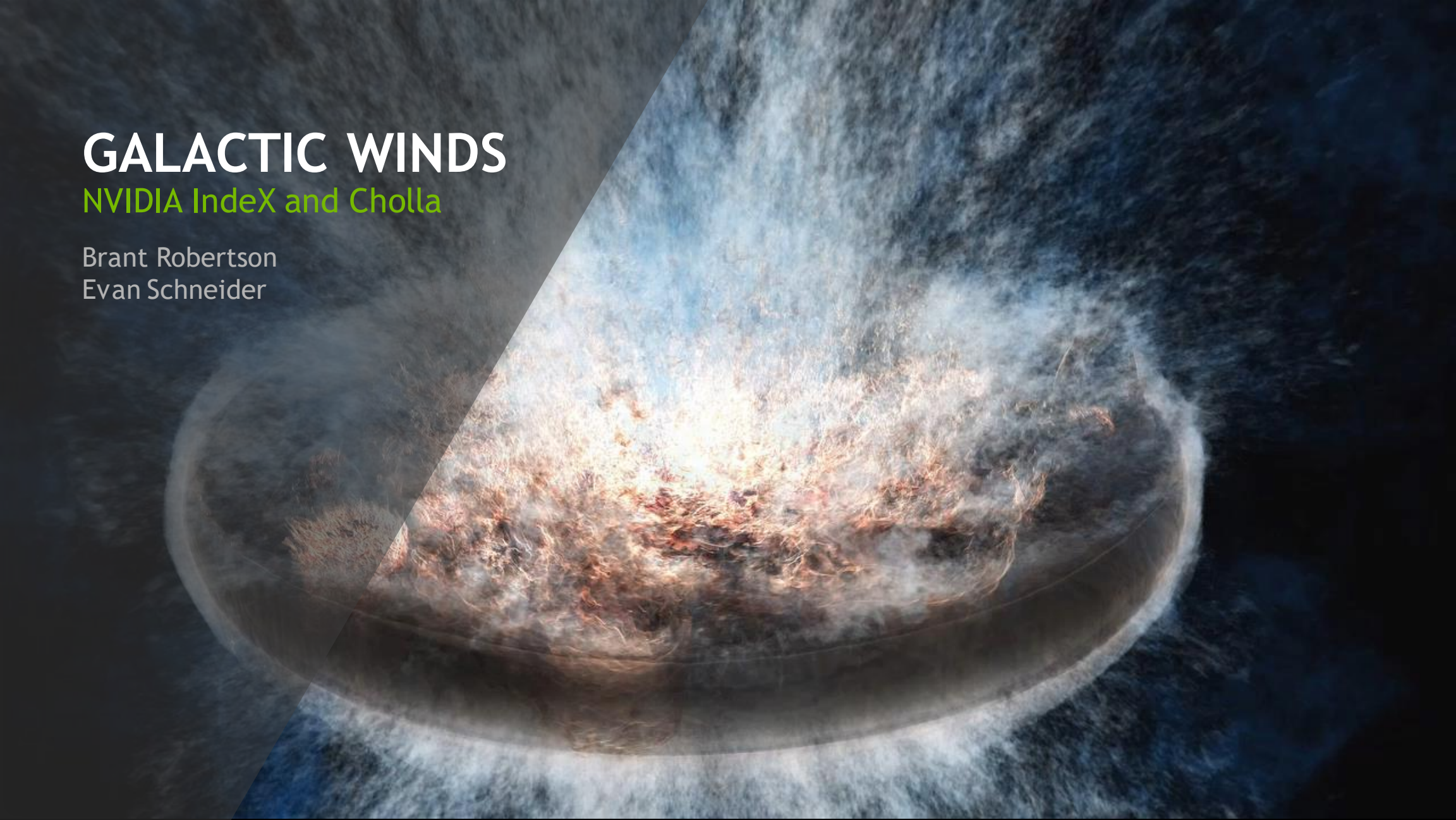
28 DGX-1 with 8 V100/32GB each

Acknowledgments

Brant Robertson (University of California, Santa Cruz)

Evan Schneider (Princeton Univerity)

Cholla - GPU-Accelerated Hydrodynamics Solver



GALACTIC WINDS

NVIDIA IndeX and Cholla

Brant Robertson
Evan Schneider

“ NVIDIA IndeX has been a revolutionary resource for us. The data volumes for our largest simulations are hundreds of gigabytes each, and even our moderate size simulations can be tens of gigabytes in size.
With NVIDIA IndeX we can explore the data in a way not previously possible, which is allowing us to understand the complicated physical structure of the galactic outflows like never before. ”

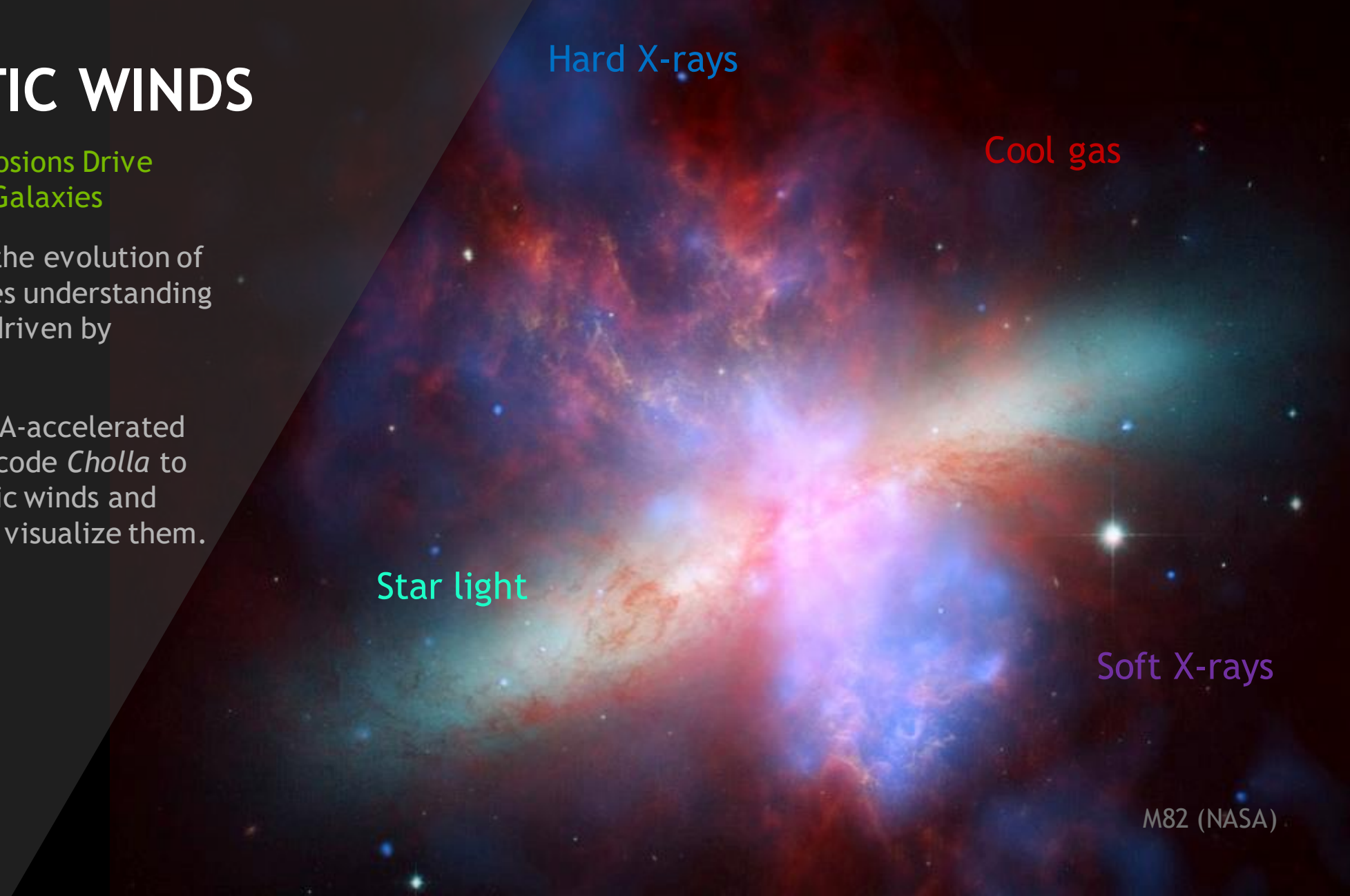
Brant Robertson
Maureen and John Hendricks Visiting Professor, Institute for Advanced Study and
Associate Professor, University of California, Santa Cruz

GALACTIC WINDS

Supernova Explosions Drive Outflows from Galaxies

Understanding the evolution of galaxies requires understanding galactic winds driven by supernovae.

We use the CUDA-accelerated hydrodynamics code *Cholla* to simulate galactic winds and NVIDIA IndeX to visualize them.



CHOLLA

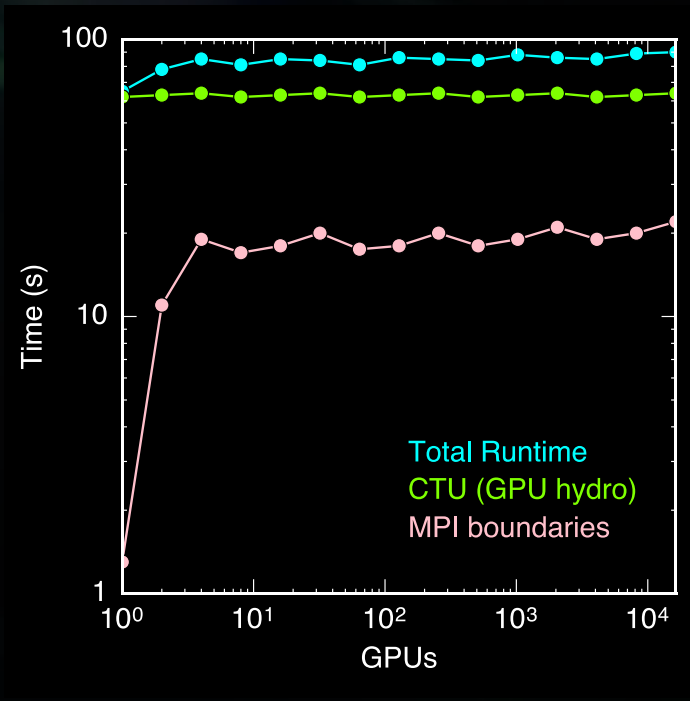
Computational Hydrodynamics On *II Architectures*

A massively parallel, GPU-native hydrodynamic simulation code written by Dr. Evan Schneider (Princeton) for her PhD thesis.

See Dr. Schneider's talk on *Cholla* and galactic winds: S9728, Wednesday 1:00pm, Hilton Hotel Market Room

- Available publicly at github.com/cholla-hydro/cholla
- Incorporates state-of-the-art hydrodynamics algorithms (unsplit integrators, 3rd order spatial reconstruction, precise Riemann solvers, dual energy formulation, etc.)
- Includes GPU-accelerated radiative cooling based on Cloudy tables.

Roughly perfect weak scaling to >10,000 NVIDIA GPUs

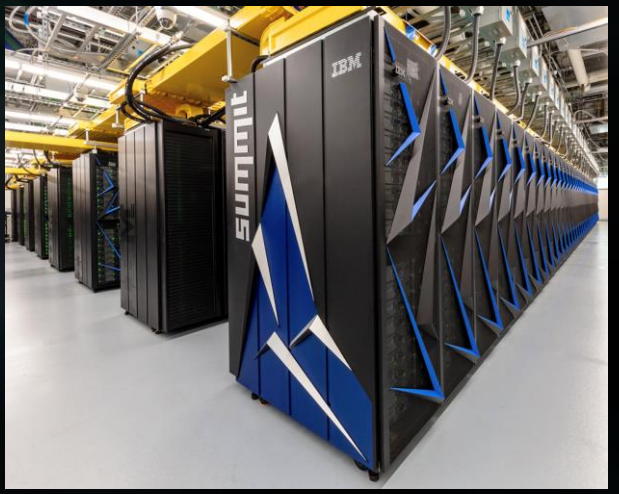


Schneider & Robertson (2015, 2017)

CHOLLA: BY THE NUMBERS

Petascale Simulations Powered by NVIDIA GPUs

Largest production simulations run to date: 17.2 billion cells, 824GB per snapshot
Largest test calculations run to date: 547 billion cells, 21.9 TB per snapshot
Largest number of NVIDIA K20X GPUs used simultaneously: 16,348 (OLCF Titan)
Largest number of NVIDIA V100 GPUs used simultaneously: 5,472 (OLCF Summit)
Currently running on Summit (Project ID AST149)



Summit (OLCF)

Our simulations run for tens of thousands of timesteps, which would result in ~**100 petabytes** of information if we saved every snapshot for a visualization.

NVIDIA IndeX is an enabling technology for us, as it will allow in-situ visualization of petascale simulations that would be impossible in post processing.

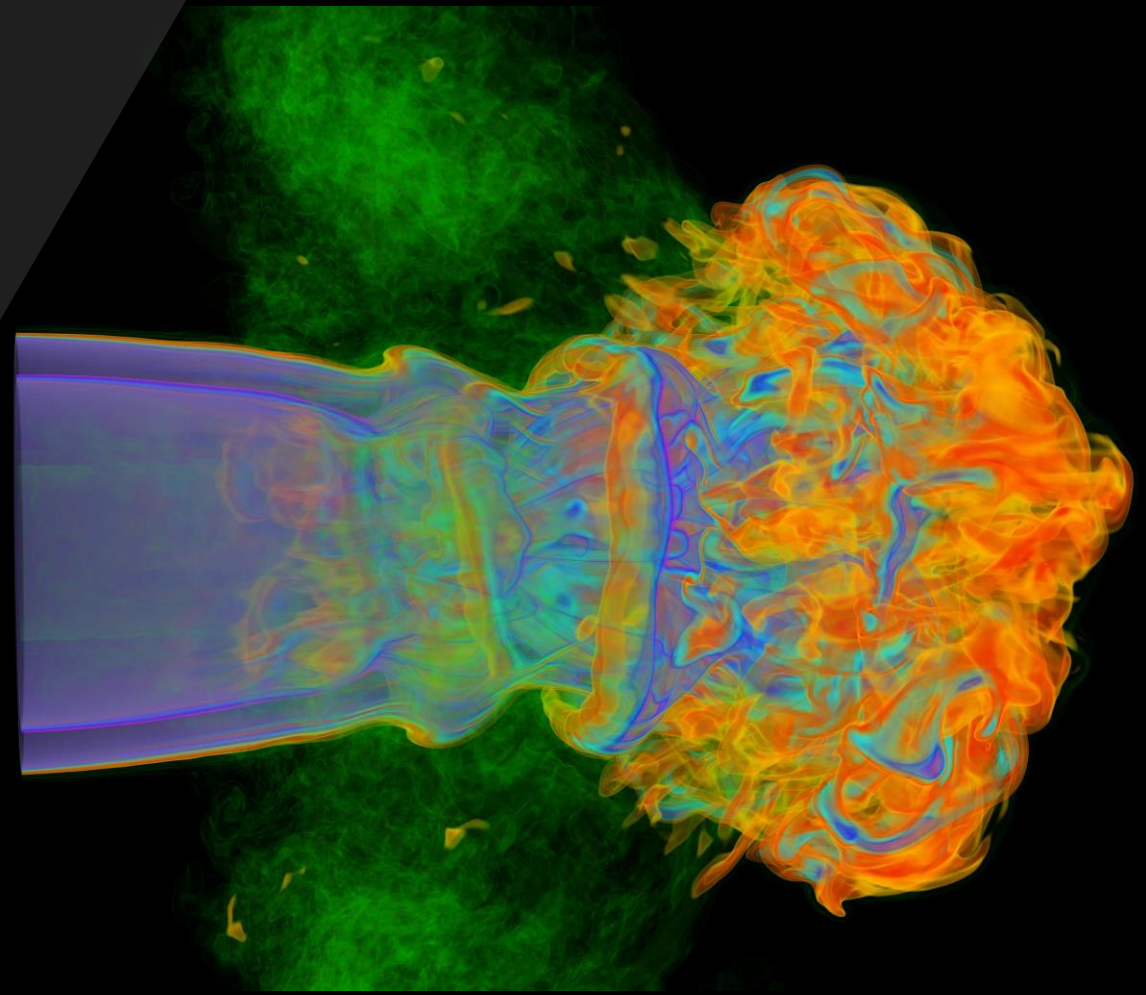
ENGINEERING SCIENCE

In-Situ Visualization

Creating new workflows

Faster discoveries

Faster decision making



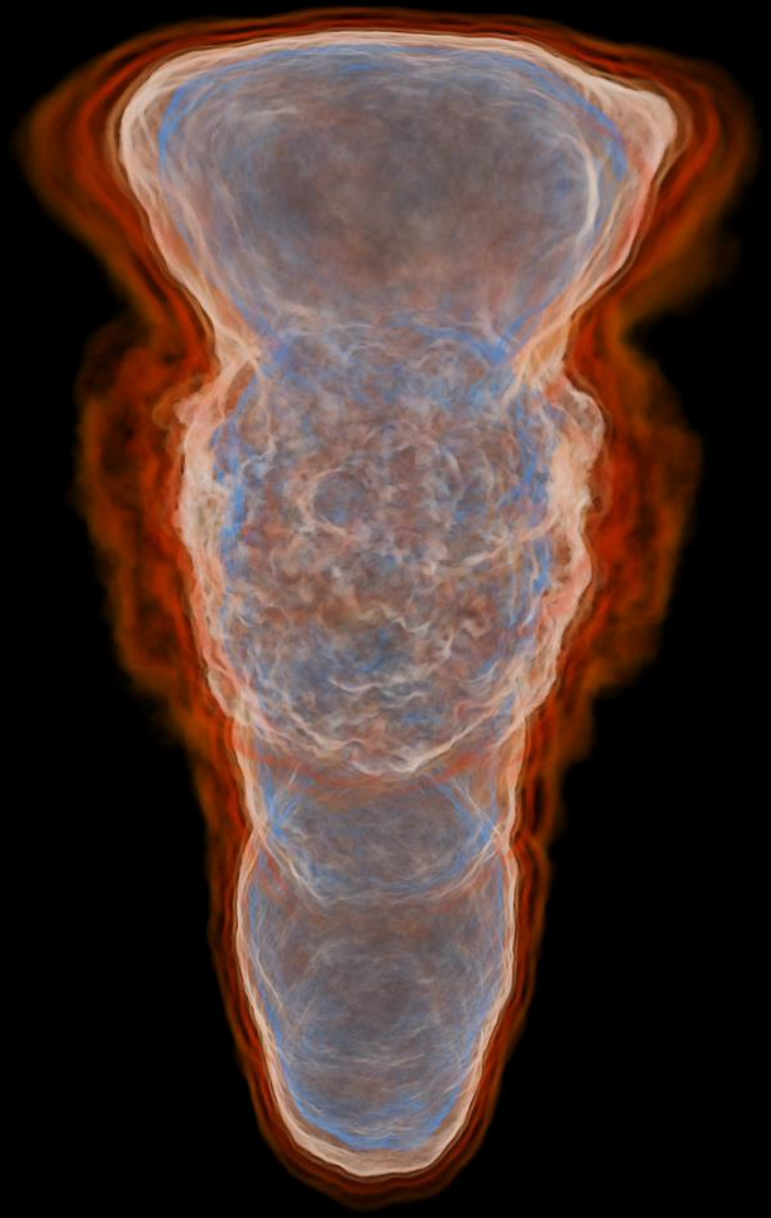
NEURON SCIENCE

HPC and Supercomputers

Containerization

Multi-node deployment

Singularity

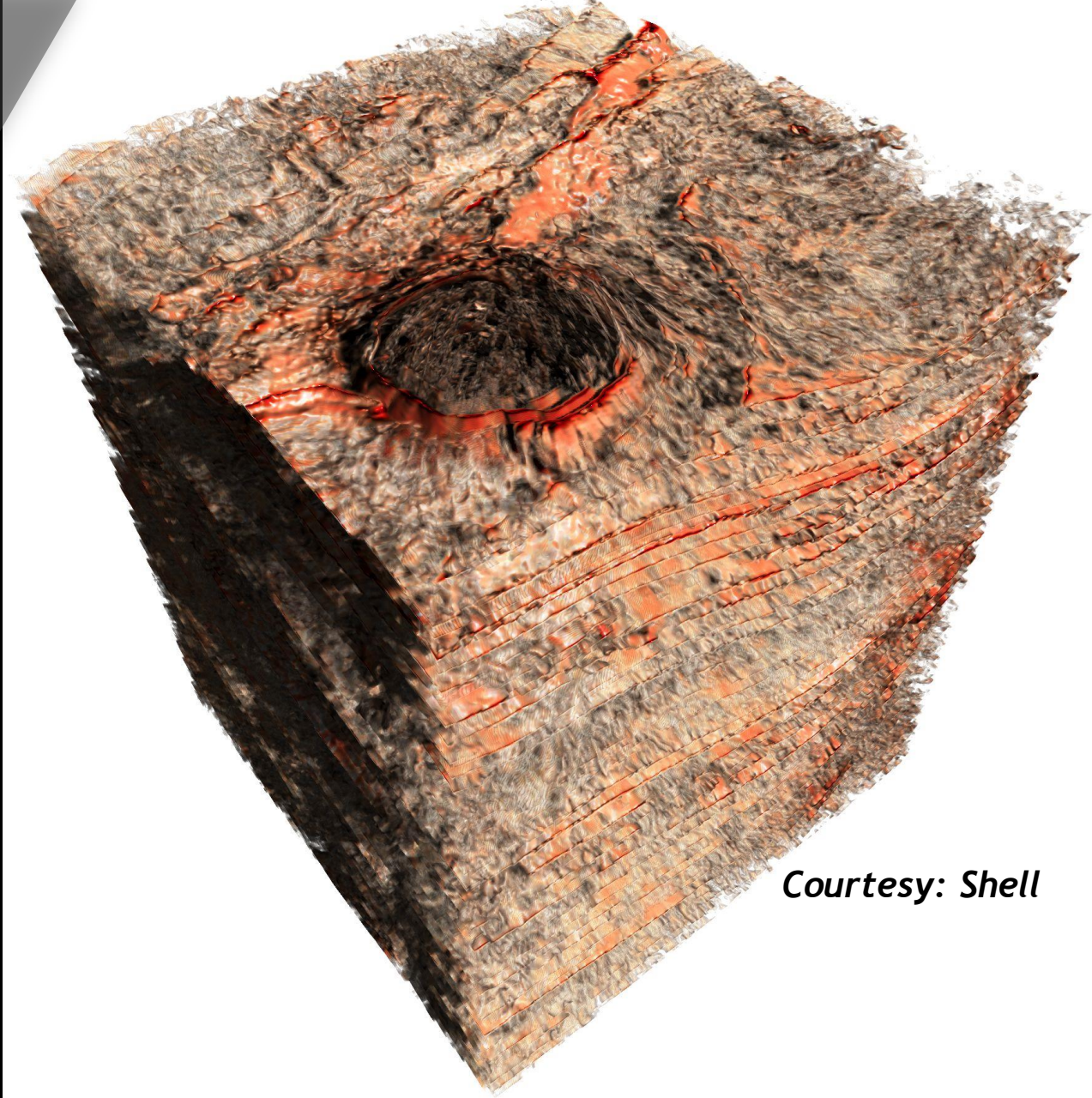


SEISMIC DATA

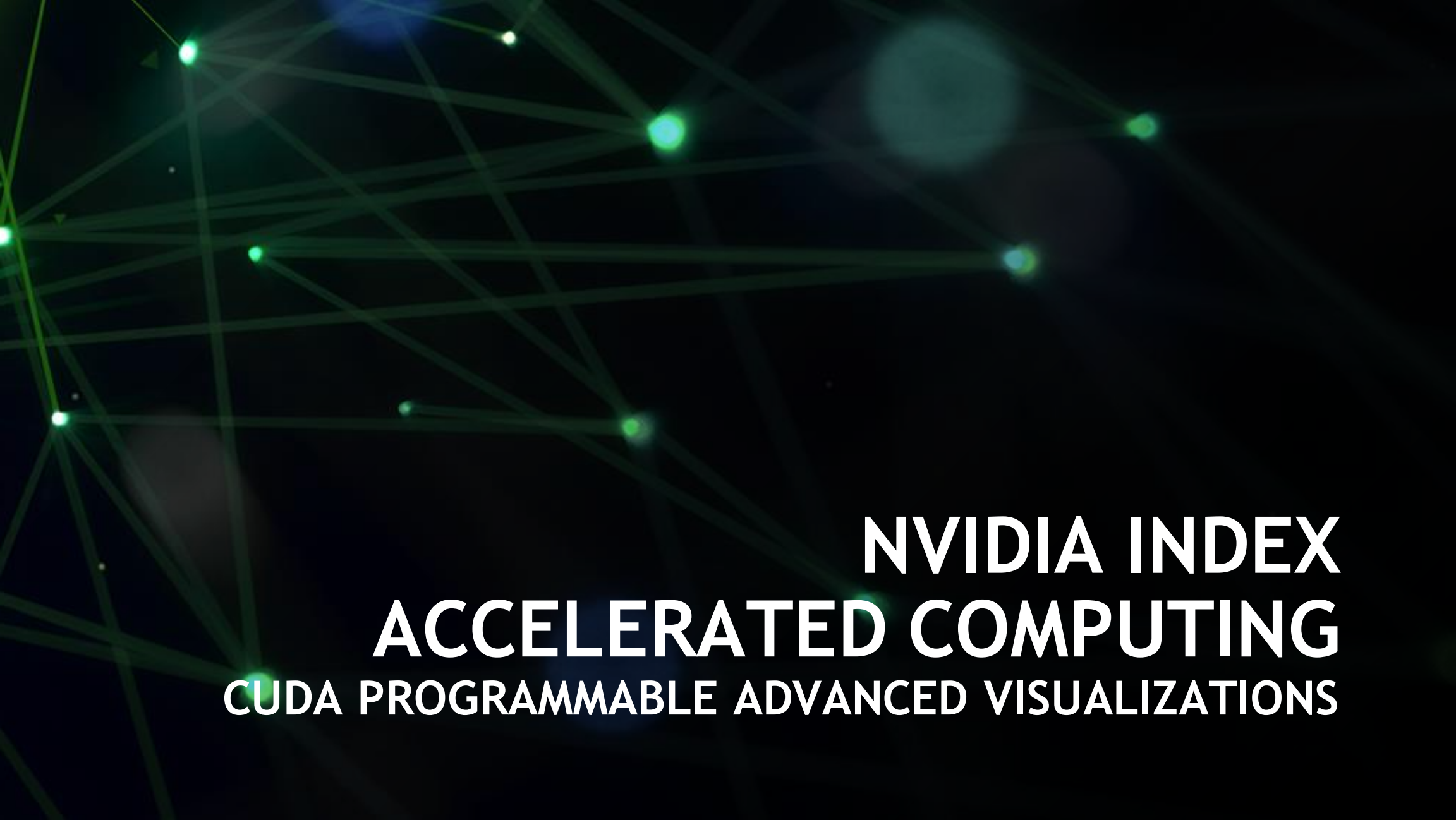
Data Interpretation in the Cloud

Google Cloud Platform & AWS Datacenters

Elasticity
Fault Tolerance
Data sharing
Headless Setups

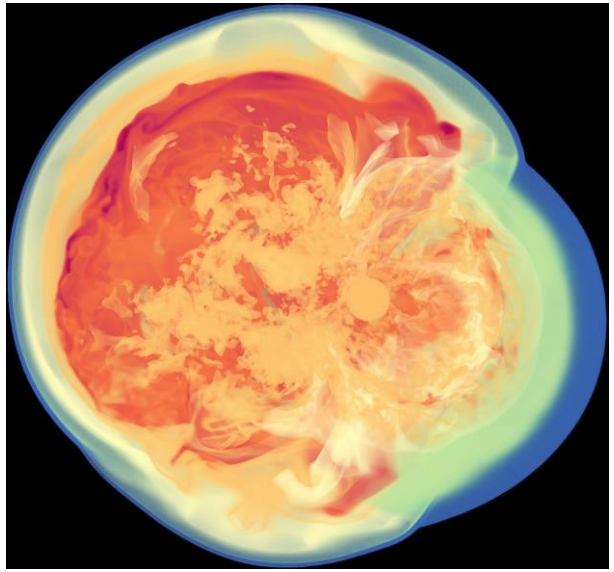


Courtesy: Shell


NVIDIA INDEX
ACCELERATED COMPUTING
CUDA PROGRAMMABLE ADVANCED VISUALIZATIONS

NVIDIA INDEX ACCELERATED COMPUTING (XAC)

Program, Compile and Execute CUDA Code at Runtime

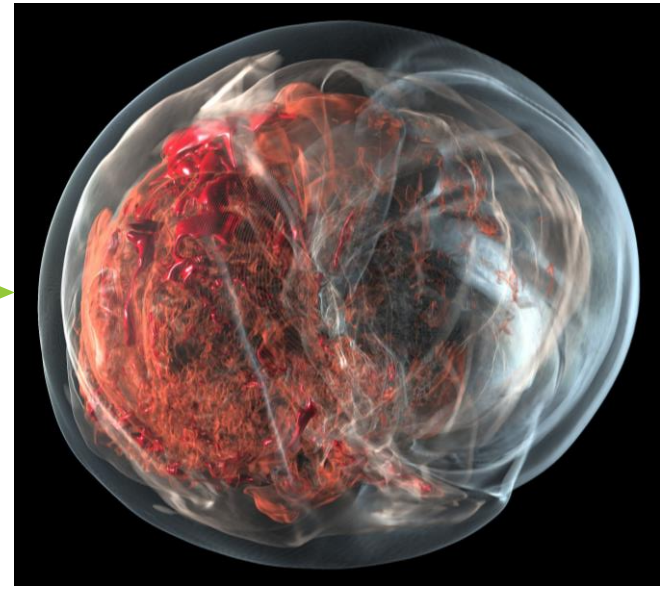


3D volume input
(Supernova)

pseudocode

```
class NV_IDX_volume_program
{
public:

int execute(
    const float3& sample_pos,
    float4&       color)
{
  float vol_sample =
    this.volume.sample(sample_pos);
  float4 sample_color =
    colormap.lookup(vol_sample);
  color = sample_color;

  return NV_IDX_PROG_OK;
}};
```

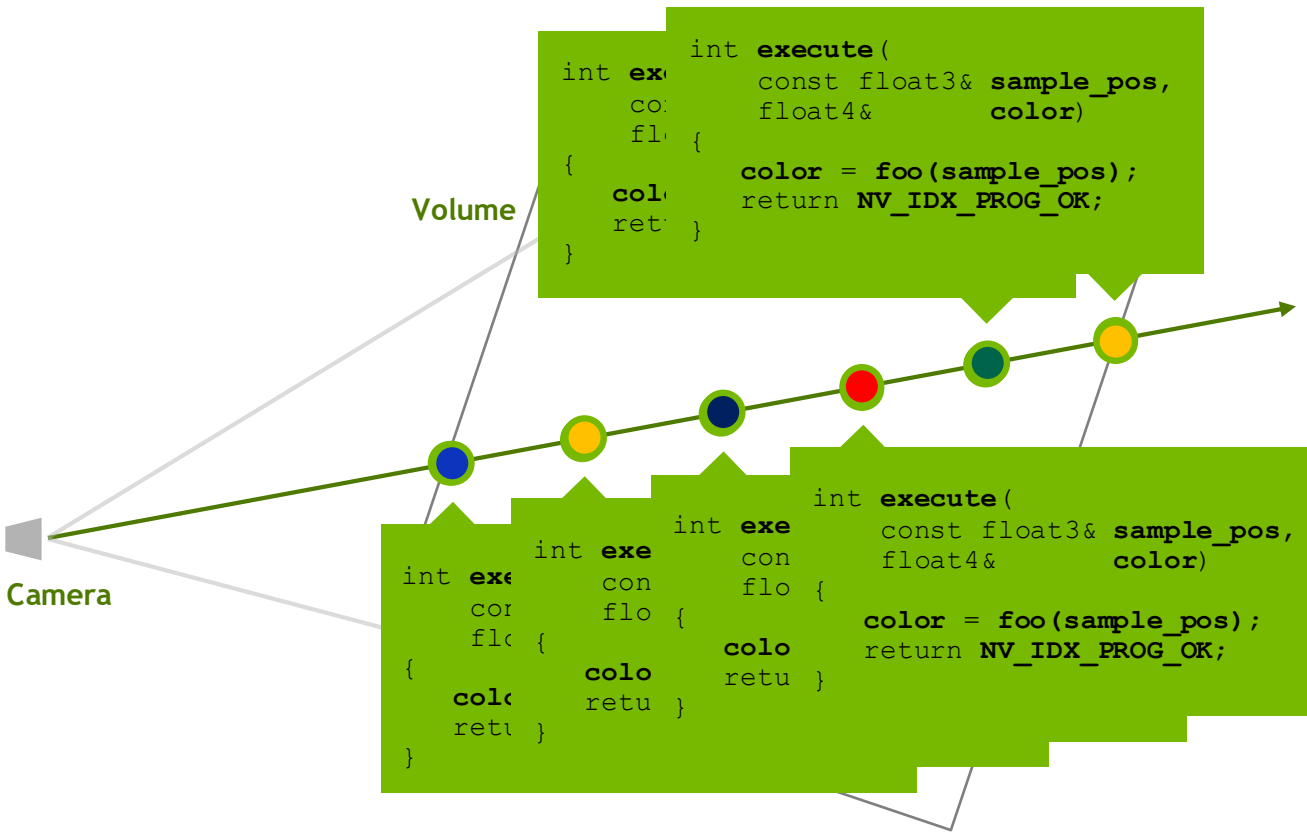
Your CUDA Source Code



High-Fidelity
3D Volume Visualization

CODE INJECTION TO INNER-LOOP OF RAY-CASTER

Distributed Cluster-Wide Sampling Along Rays Transparent to User

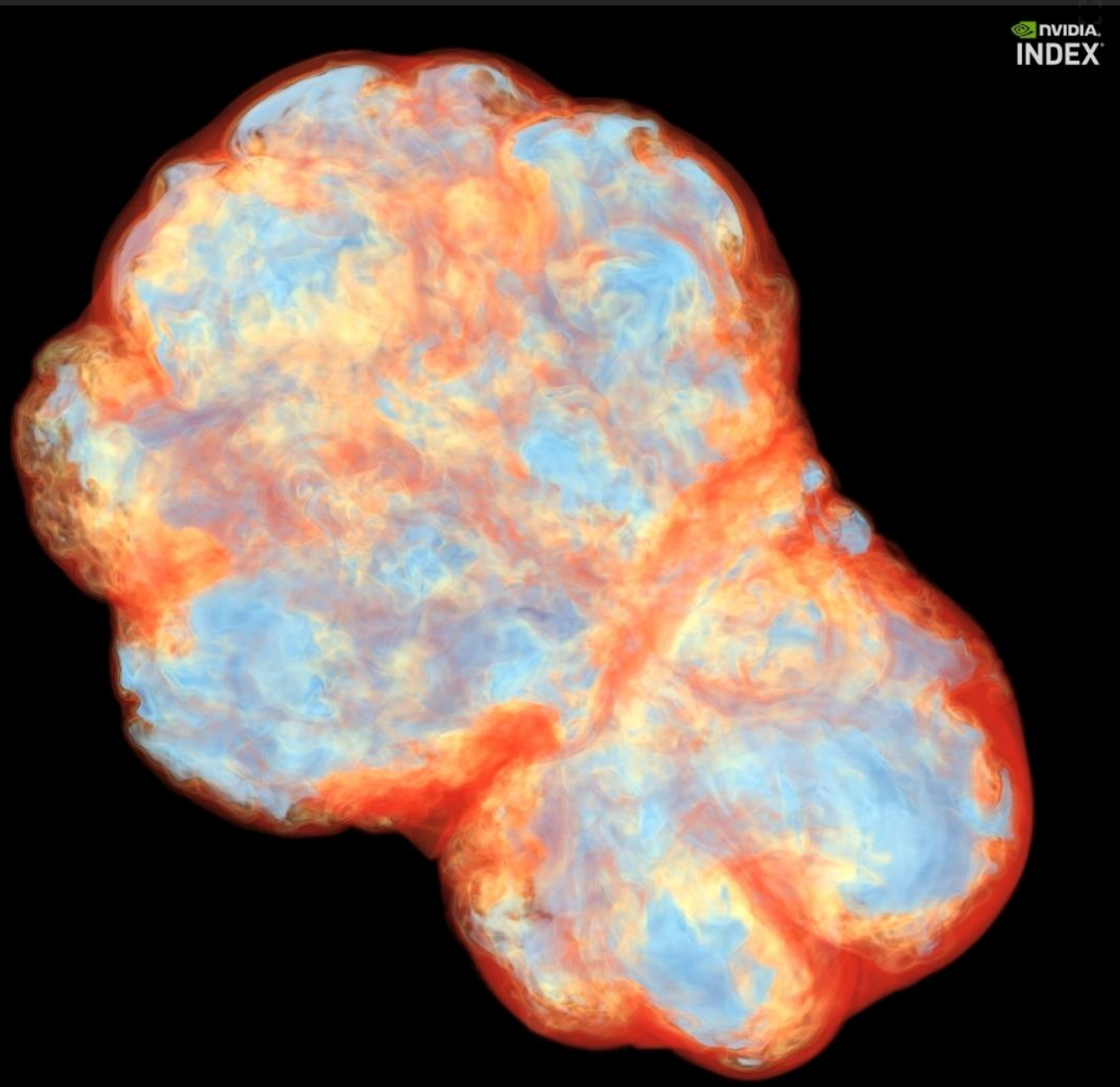


CUDA Code Editor (1)

editor target: rtc_volume Compile

code preset: Basic Example

```
using namespace nv::index;
using namespace nv::index::xac;
using namespace nv::index::xaclib;

class Volume_sample_program
{
    NV_IDX_VOLUME_SAMPLE_PROGRAM

    public:
    NV_IDX_DEVICE_INLINE_MEMBER void initialize() {}

    NV_IDX_DEVICE_INLINE_MEMBER
    int execute(
        const Sample_info_self& sample_info,
              Sample_output& sample_output)
    {
        const float sample = state.self.sample<float>(sample_info.sample_position);
        const float4 color = state.self.get_colormap().lookup(sample);

        sample_output.set_color(gamma_correct(color, 0.8f));

        return NV_IDX_PROG_OK;
    }};
```

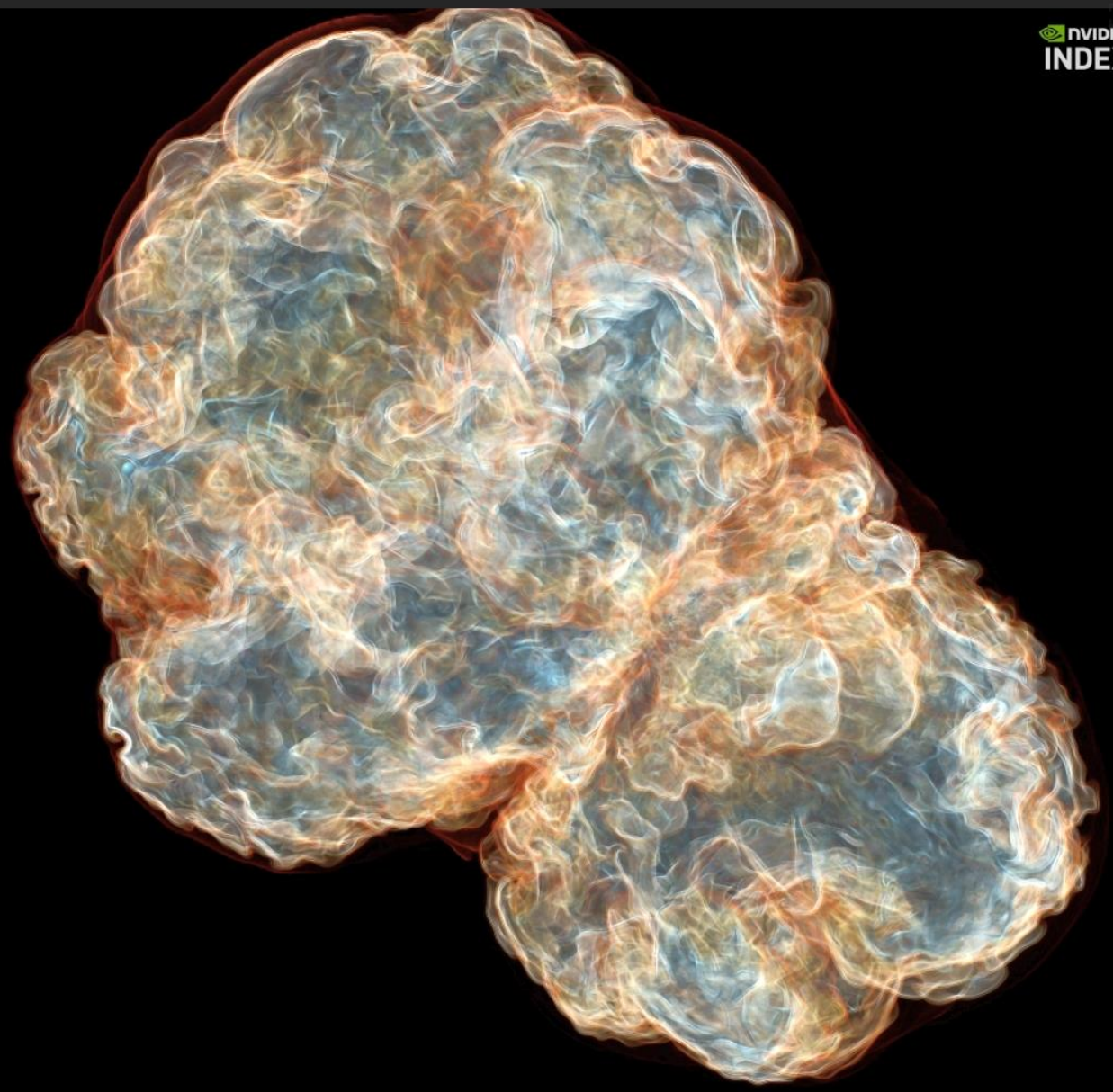


CUDA Code Editor (1)

editor target: rtc_volume | Compile

code preset: Custom Code

```
NV_IDX_XAC_VERSION_1_0

using namespace nv::index;

class Volume_sample_program
{
    NV_IDX_VOLUME_SAMPLE_PROGRAM

    public:
    NV_IDX_DEVICE_INLINE_MEMBER
    void initialize() {}

    NV_IDX_DEVICE_INLINE_MEMBER
    int execute(
        const Sample_info_self &sample_info,
              Sample_output &sample_output)
    {
        // get position & references
        const float3 &sample_position = sample_info.sample_position;
        const xac::Colormap colormap = state.self.get_colormap();

        // sample volume and color
        const float volume_sample = state.self.sample<float>(sample_position);
        const float4 sample_color = colormap.lookup(volume_sample);

        // compute volume gradient
        const float3 vol_grad = state.self.get_gradient(sample_position);

        // shade color by gradient magnitude
        const float grad_fac = length(vol_grad) * 200.0f;

        // scale color and apply gamma correction
        float4 out_color = clamp(sample_color * grad_fac, 0.0f, 1.0f);
        out_color = xaclib::gamma_correct(out_color, 0.6f);

        // set output color
        sample_output.set_color(out_color);

        return NV_IDX_PROG_OK;
    }
};
```



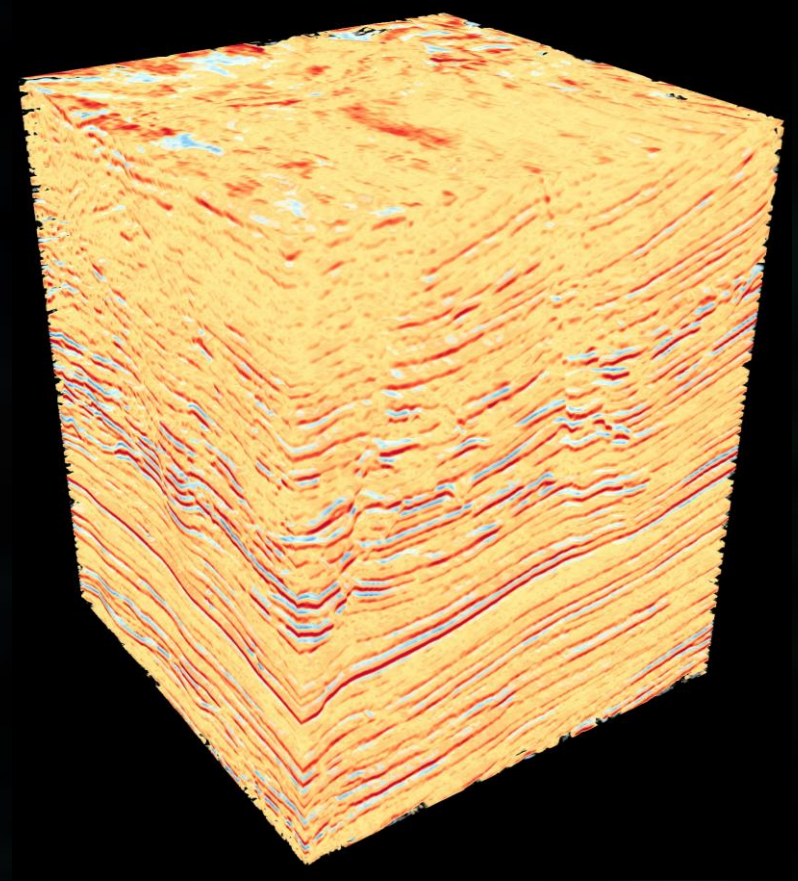
NVIDIA INDEX ACCELERATED COMPUTING (XAC) ON DISTRIBUTED DATA

Spatial Subdivision for Scalable Rendering

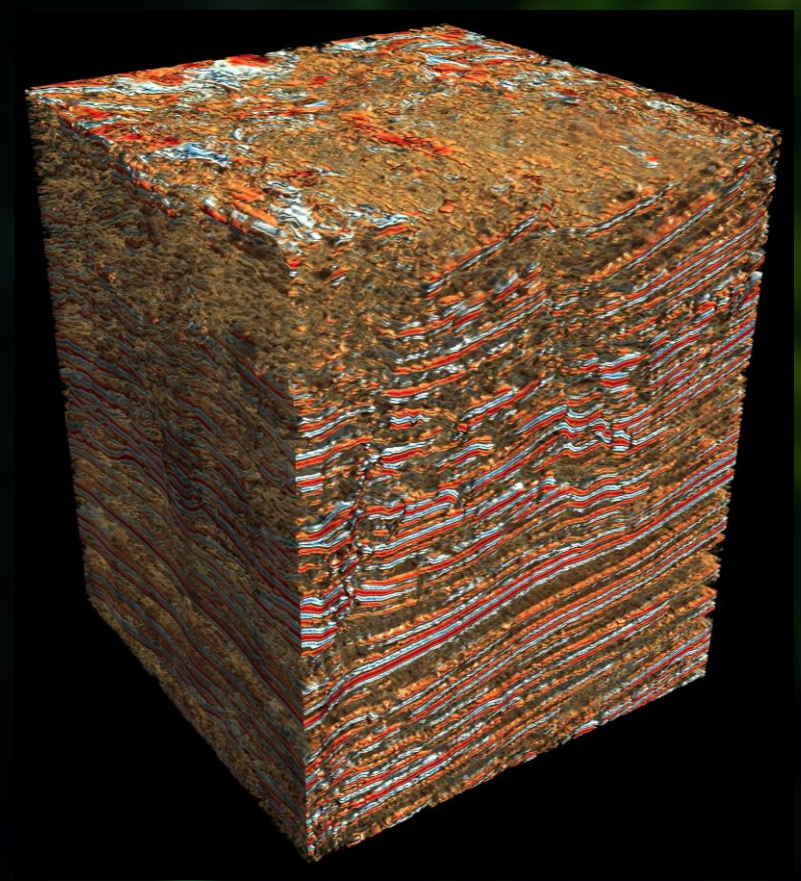
3D Seismic Volume

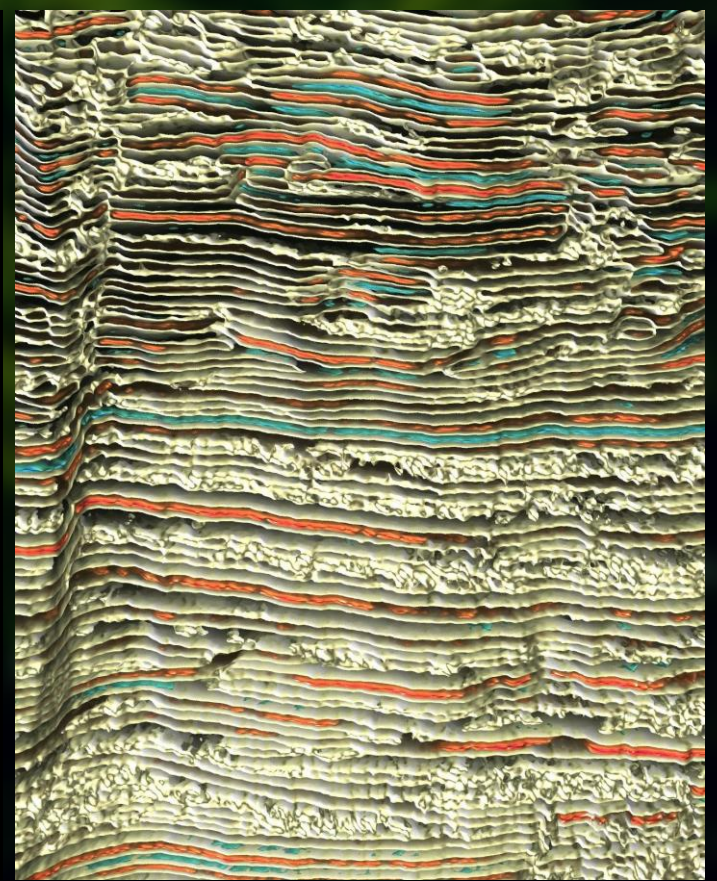
Visualization using XAC

SEISMIC DATA VISUALIZATION



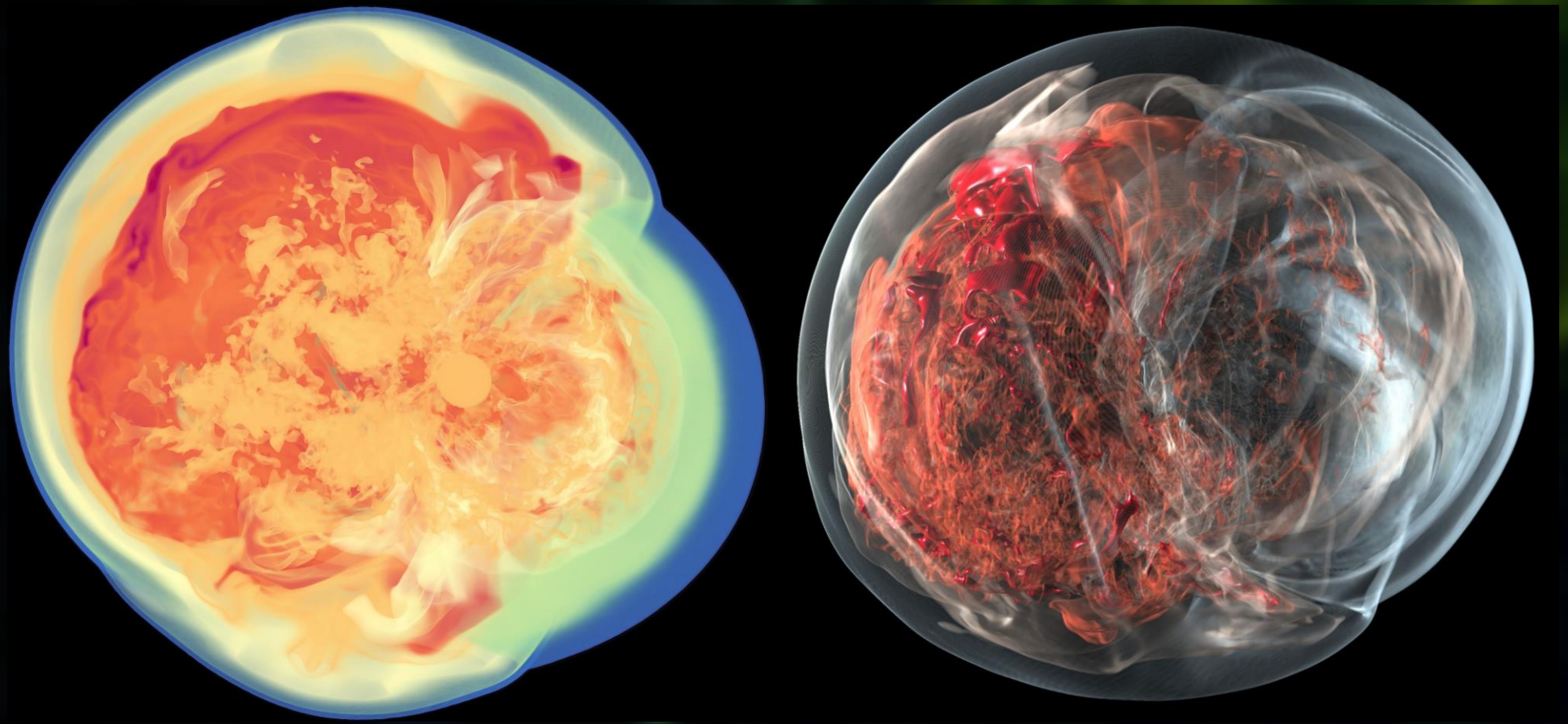
No XAC Shading





XAC Shading

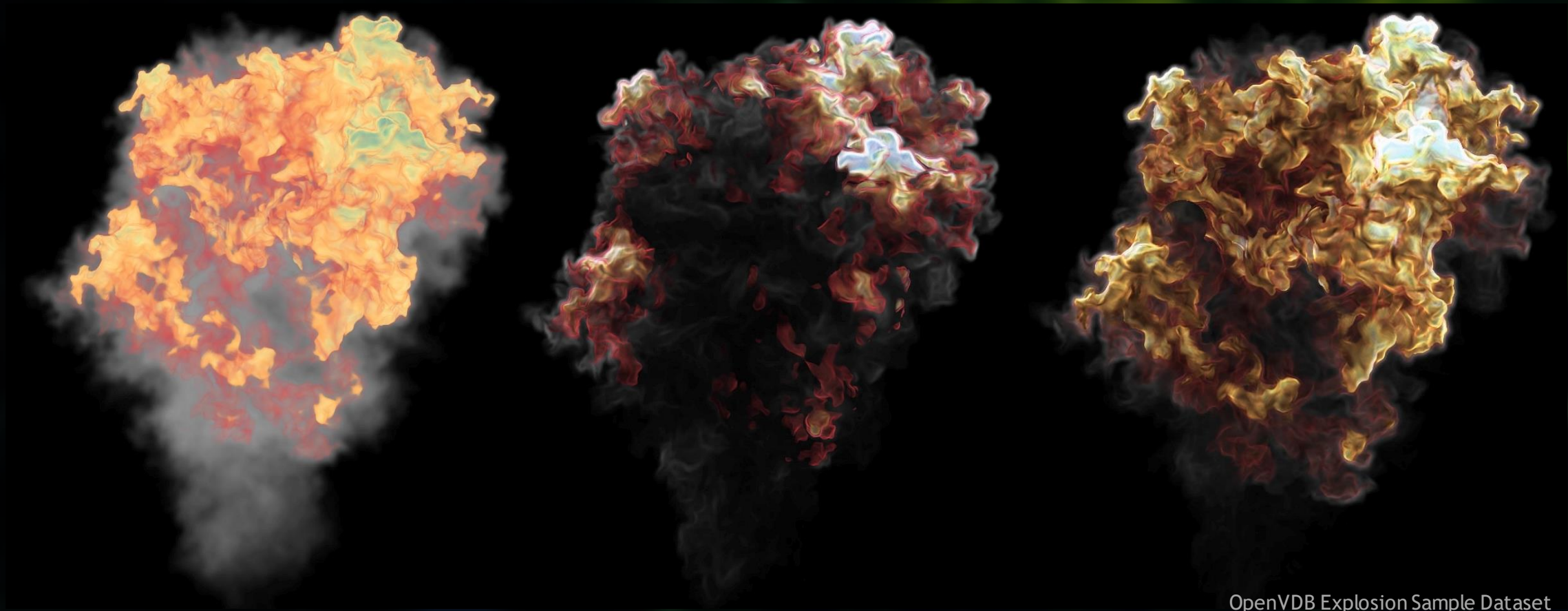
SUPERNOVA VISUALIZATION



No XAC Shading

XAC Shading

EXPLOSION EFFECTS

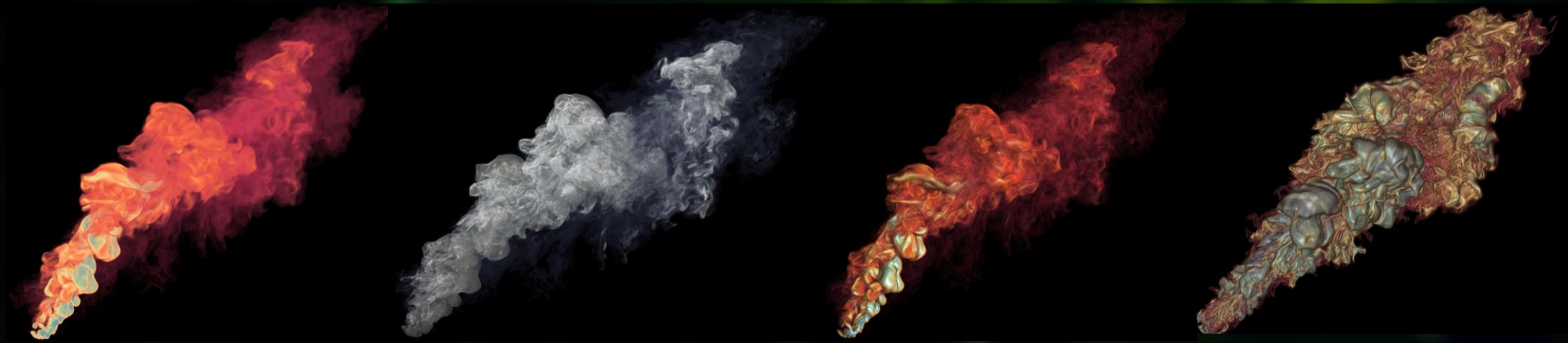


OpenVDB Explosion Sample Dataset
Voxel Resolution: 201x272x230
www.openvdb.org/download/

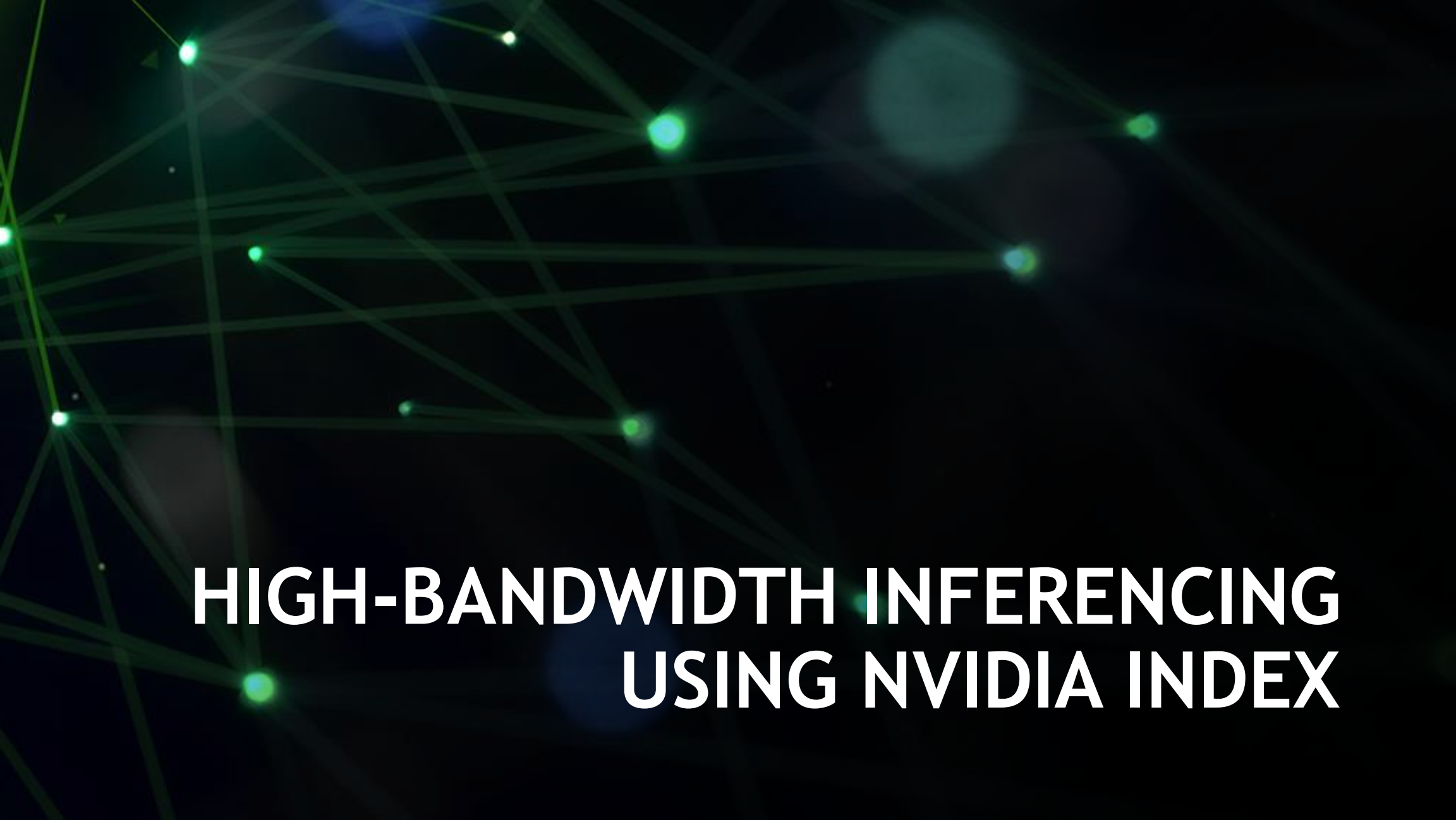
No XAC Shading

XAC Shading

SMOKE EFFECTS USING XAC



OpenVDB Smoke2 Sample Data Set
Voxel Resolution: 191x610x178
www.openvdb.org/download/

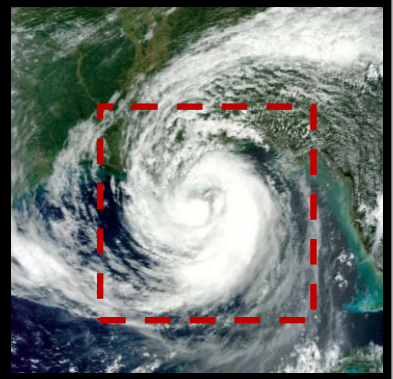

HIGH-BANDWIDTH INFERENCING
USING NVIDIA INDEX

AI FOR SCIENCE THROUGH NVIDIA INDEX

David Hall – Numerical Weather Prediction – Overview
(Tuesday morning, GTC Recordings)

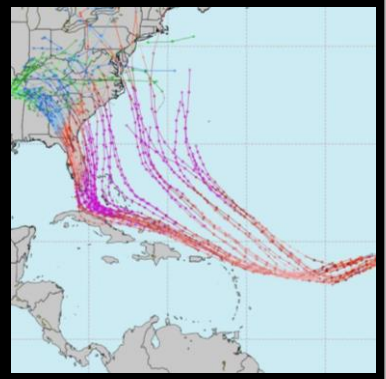
DETECTION

Tropical storms
Extra-tropical cyclones
Atmospheric rivers
Cyclogenesis events
Convection initiation
Change detection



PREDICTION

Uncertainty prediction
Storm track
Storm intensity
Fluid motion
Now casting
Satellite frame prediction



Create Solutions

NVIDIA INDEX®

AI-Guided Data Analysis and Interpretation

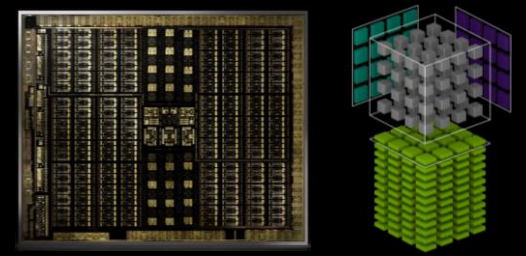
Leverage NVIDIA Platforms

cuDNN

TensorRT

GPU Accelerated AI Frameworks

DGX Systems & Clusters for AI



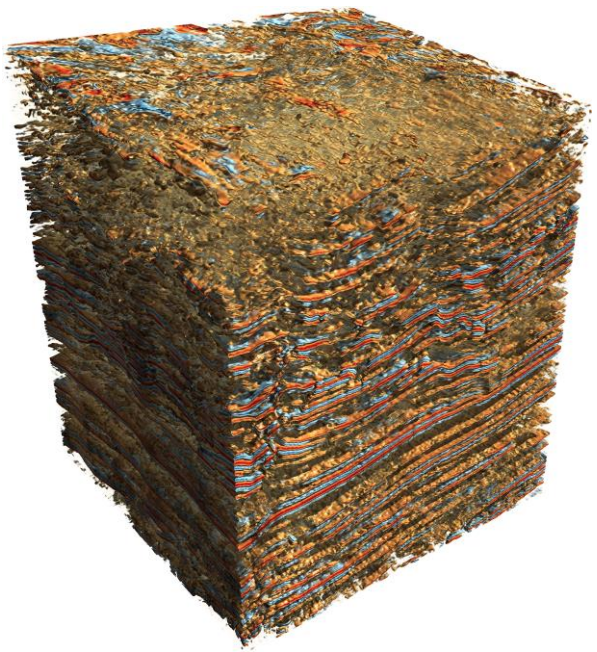
Volta & Turing & T4 with Tensor Cores

AI-GUIDED VISUAL DATA ANALYSIS AND INTERPRETATION

Idea: Combining AI-Frameworks with XAC Program Evaluation

Trained Network

Seismic (3D Volume)



pseudocode

```
#include cuDNN

class NV_IDX_volume_program
{
public:

int execute(
    const NV_IDX_cnn& cnn,
    NV_IDX_volume&    feature)
{
    /* Run cnn against volume */
    feature = detect(cnn, self);

    return NV_IDX_PROG_OK;
}};
```

Visualization

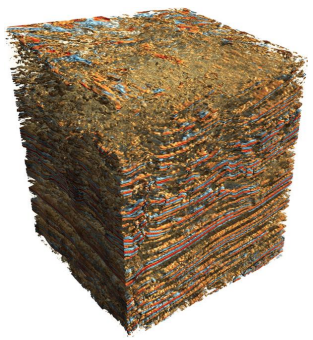
Input → **Inference** → **Segmentation**

AI-GUIDED VISUAL DATA ANALYSIS

Visualizing Inference Results in Combination with the Source Datasets

Trained Network

Seismic (3D Volume)



C++ Callback

pseudocode

```
#include TensorRT

class Compute_callback_host
{
public:

int execute(
    const Network& dnn,
    Volume         volume,
    Data&          output)
{
    /* Run TensorRT */
    output = tensor_rt->apply(
                dnn,
                volume);
}};
```

XAC Program

pseudocode

```
class NV_IDX_volume_program
{
public:

int execute(
    const Data&        in,
    NV_IDX_volume&     output)
{
    /* Apply inference results
       with volume data */
    output = combine(in, self);

    return NV_IDX_PROG_OK;
}};
```

Visualization

CUDA-based workflow

Input → ***CUDA device*** **buffers share inference results with XAC shader/compute program evaluation** → **Segmentation**

INFERFACE FOR USER-DEFINED INFERENCING ON THE GPU

Launched by NVIDIA IndeX Giving Access to Distributed Data

```
void My_tensorrt_inference_technique::inference(
    const IInference_source_data*           source_data,
    IMemory_manager*                        memory_manager,
    IInference_result*                      result_data_assignment) const
{
    // Access volume data resident on the CUDA device:
    const mi::math::Bbox_struct<mi::Float32, 3> brick_bbox;
    char* brick_data = source_data->get_brick(brick_bbox);  // CUDA device pointer.

    // Request CUDA device buffer, here: linear device buffer.
    nv::index::ICuda_memory_buffer* buffer = memory_manager->get_cuda_memory_manager()->
        request_linear_device_memory<mi::math::Bbox_struct<mi::Float32, 3>>(1);
    char* device_ptr = buffer->get();

    // Implement inference using TensorRT (pseudocode). Free to use other AI Frameworks.
    tensorRT->inference(
        cnn,          /* trained network */
        brick_data,   /* source data */
        device_ptr,   /* inference results */);

    // Bind inference results to a buffer/slot to make use in a subsequent XAC shader.
    const mi::Uint32 slot_id = 0;
    result_data_assignment->bind_inference_result(slot_id, buffer);
}
```

inference()-Callback-Invocation

(1)	C++-callback issues by the NVIDIA IndeX rendering system.
(2)	Request a CUDA device pointer to a volume brick or slice.
(3)	Request CUDA device memory from NVIDIA IndeX to store inference results.
(4)	Run inference ***here*** using TensorRT ***(pseudocode)***.
(5)	Write inference results to requested device memory.
(6)	Bind device memory buffer containing inference result to a XAC shader.

SUBSEQUENT XAC SHADER EXECUTION

Using Inference Results

```
class Volume_sample_program
{
public:
    int execute(const Sample_info_self& sample_info,
                      Sample_output&    sample_output)
    {
        const auto   svol_sampler = state.self.generate_sampler<float, xac::Volume_filter_mode::TRILINEAR>(0, sample_info.sample_context);
        const float3 sample_pos   = sample_info.sample_position_object_space;

        const int slot_id  = 0;
        const Bbox* buffer = state.self.bind_device_buffer<Bbox>(slot_id);

        if( buffer->is_inside(sample_pos) )
        {
            const float v = svol_sampler.fetch_sample(sample_info.sample_position_object_space);
            const nv::index::xac::Colormap m_colormap = state.self.get_colormap();
            sample_output.set_color(m_colormap.lookup(v));
        }
        else
        {
            const float4 default_color = make_float4(0.1, 0.1, 0.1, 0.2);  // translucent and greyed out.
            sample_output.set_color(default_color);
        }

        return NV_IDX_PROG_OK;
    }
};
```

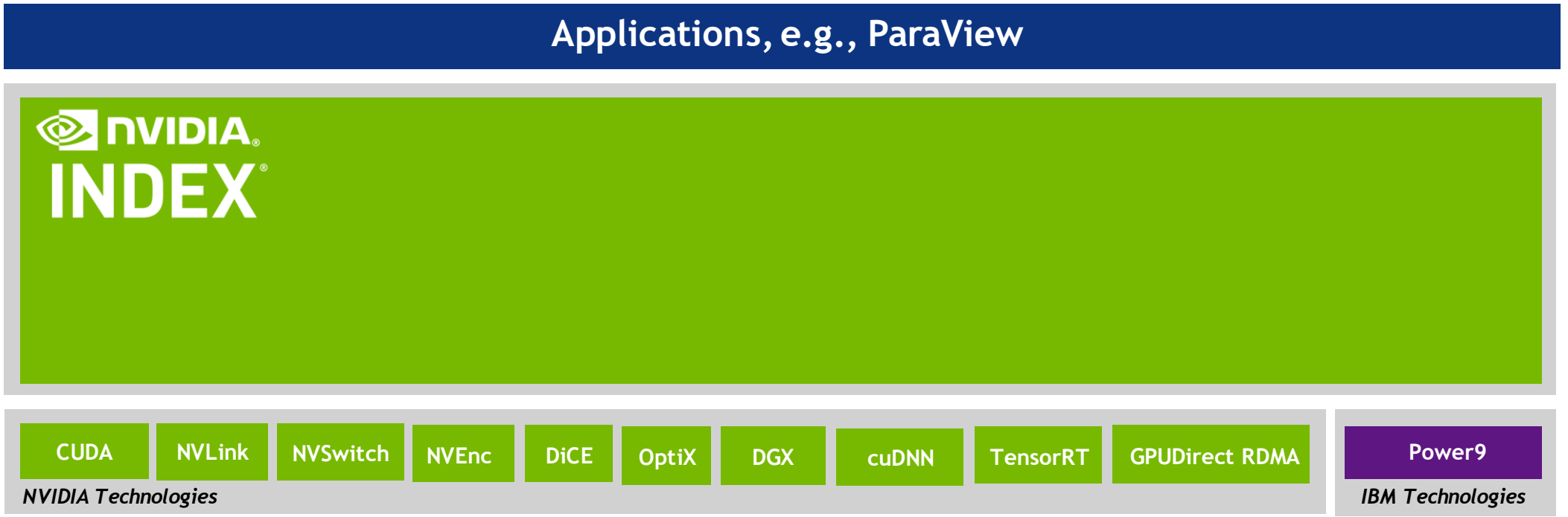
GPU-ACCELERATED HIGH-BANDWIDTH INFERENCING

Spatial Subdivision for Scalable Inference Algorithms

NVIDIA INDEX ENTERPRISE LAYER AND PLUGIN ARCHITECTURE

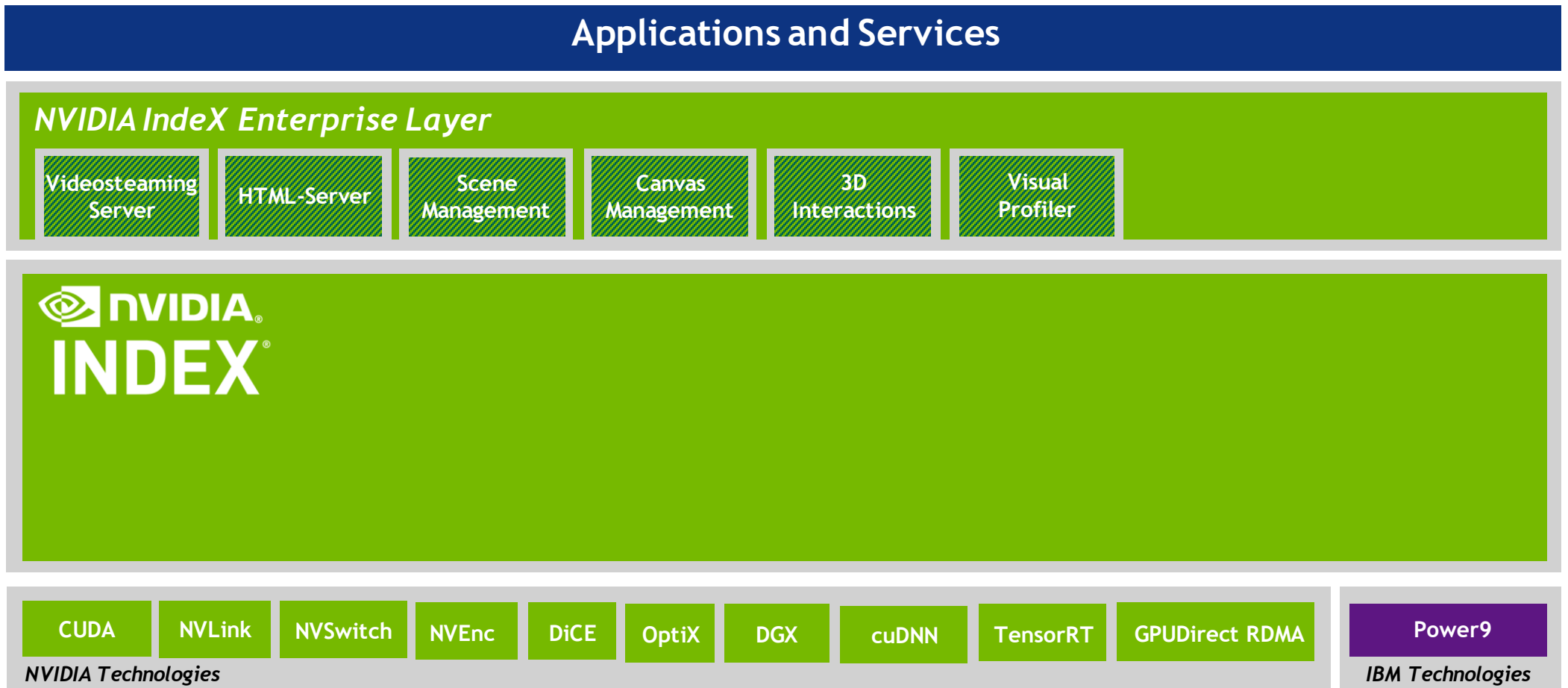
MAKING USE OF NVIDIA TECHNLOGIES

NVIDIA IndeX SDK – C++ API



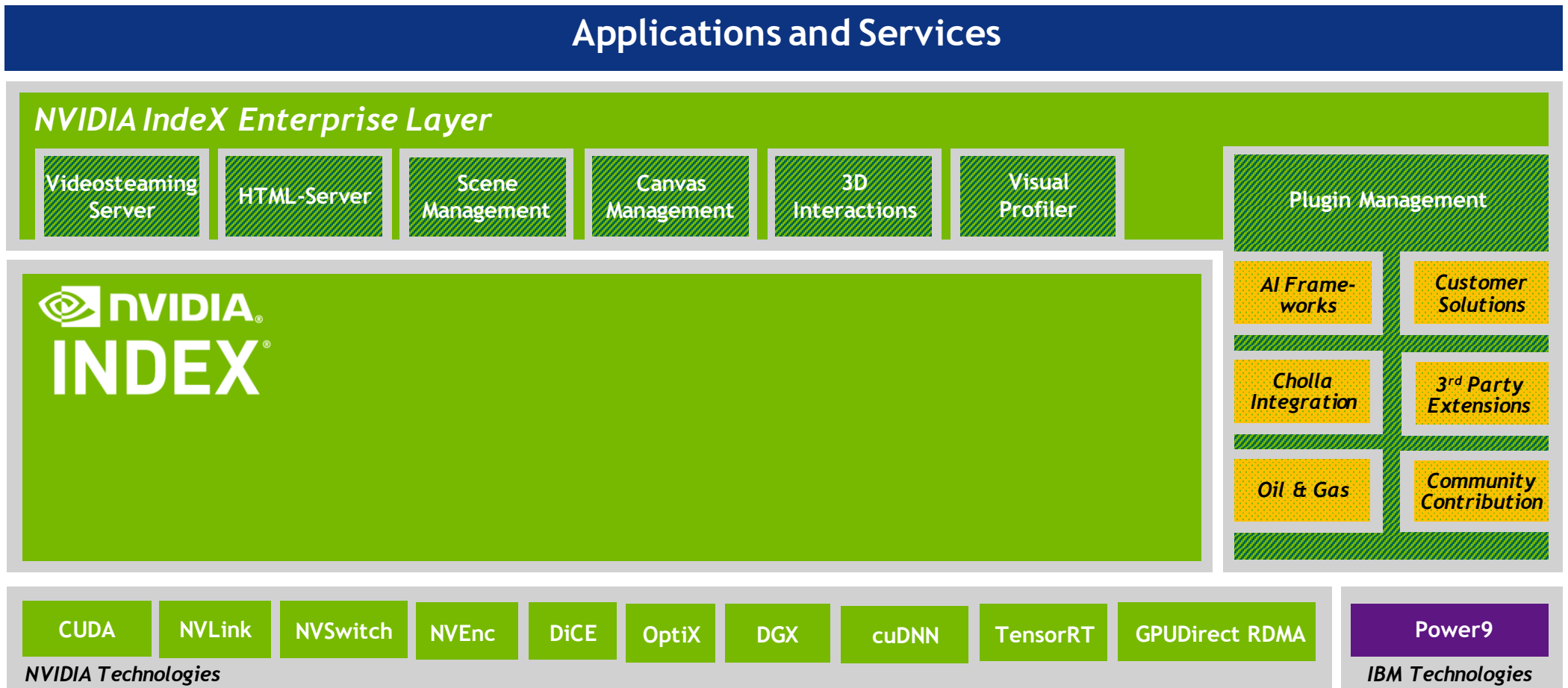
MAKING NVIDIA INDEX INTEGRATION EASY

NVIDIA IndeX Enterprise Layer exposing Convenience Functionalities



PLUGINS LOWERING THE BAR FURTHER

NVIDIA Extensions or External Contributions



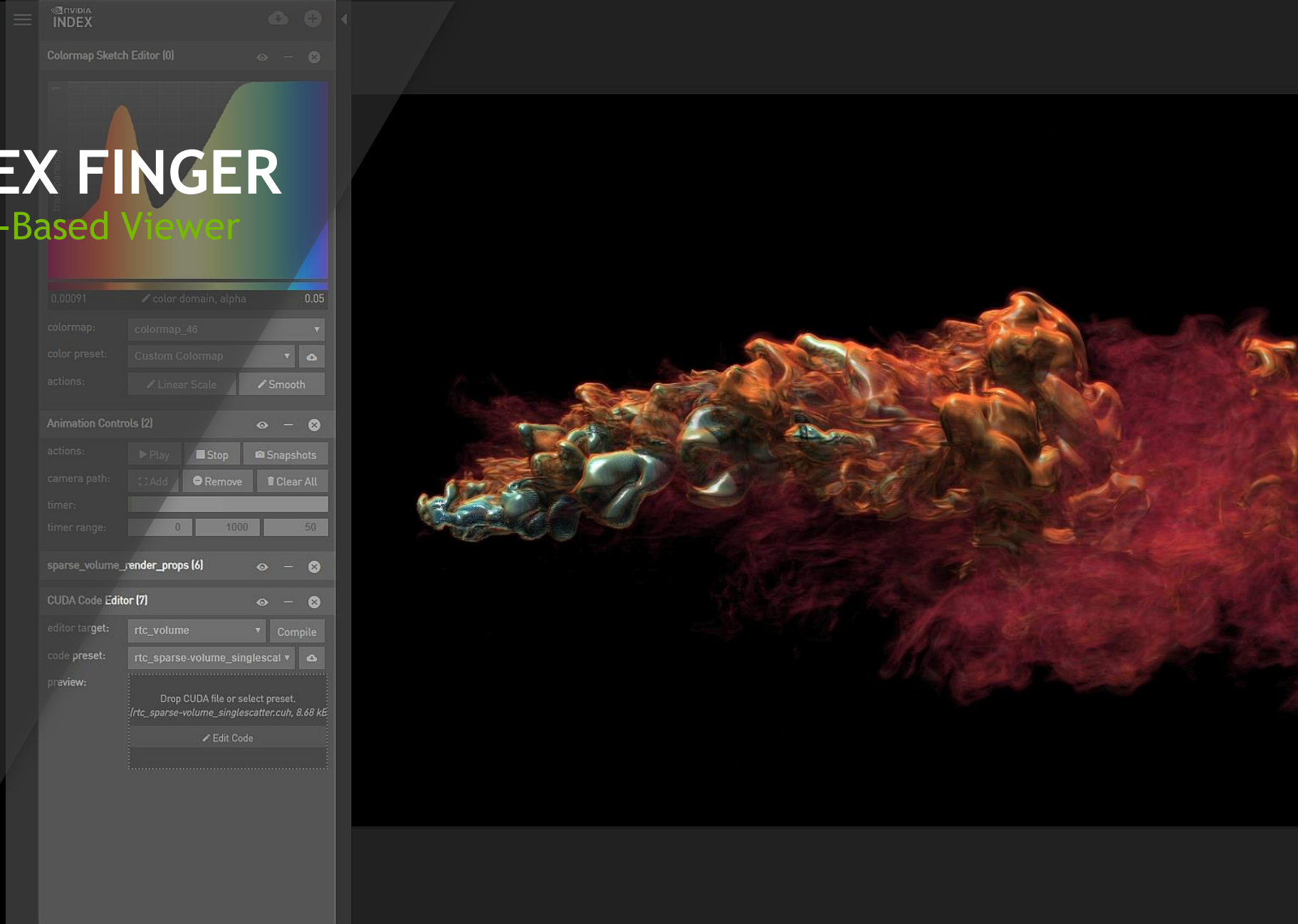
PLUGIN EXAMPLES

O&G	AI	OPENVDB	CHOLLA INTEGRATION	MEDICAL
Importers Compute techniques Inference technique Common workflows	AI Framework integration ▪ cuDNN ▪ TensorRT	OpenVDB Importer Compute integration In-Situ workflow	Cholla-NVIDIA IndeX synchronization In-situ workflow Run on Summit	Nifti importer DiCOM importer Inference techniques


INTRODUCTION OF
NVIDIA INDEX FINGER

NVIDIA INDEX FINGER

Browser & HTML-5-Based Viewer



FEATURES

Configurable Panels for:

Scene Elements

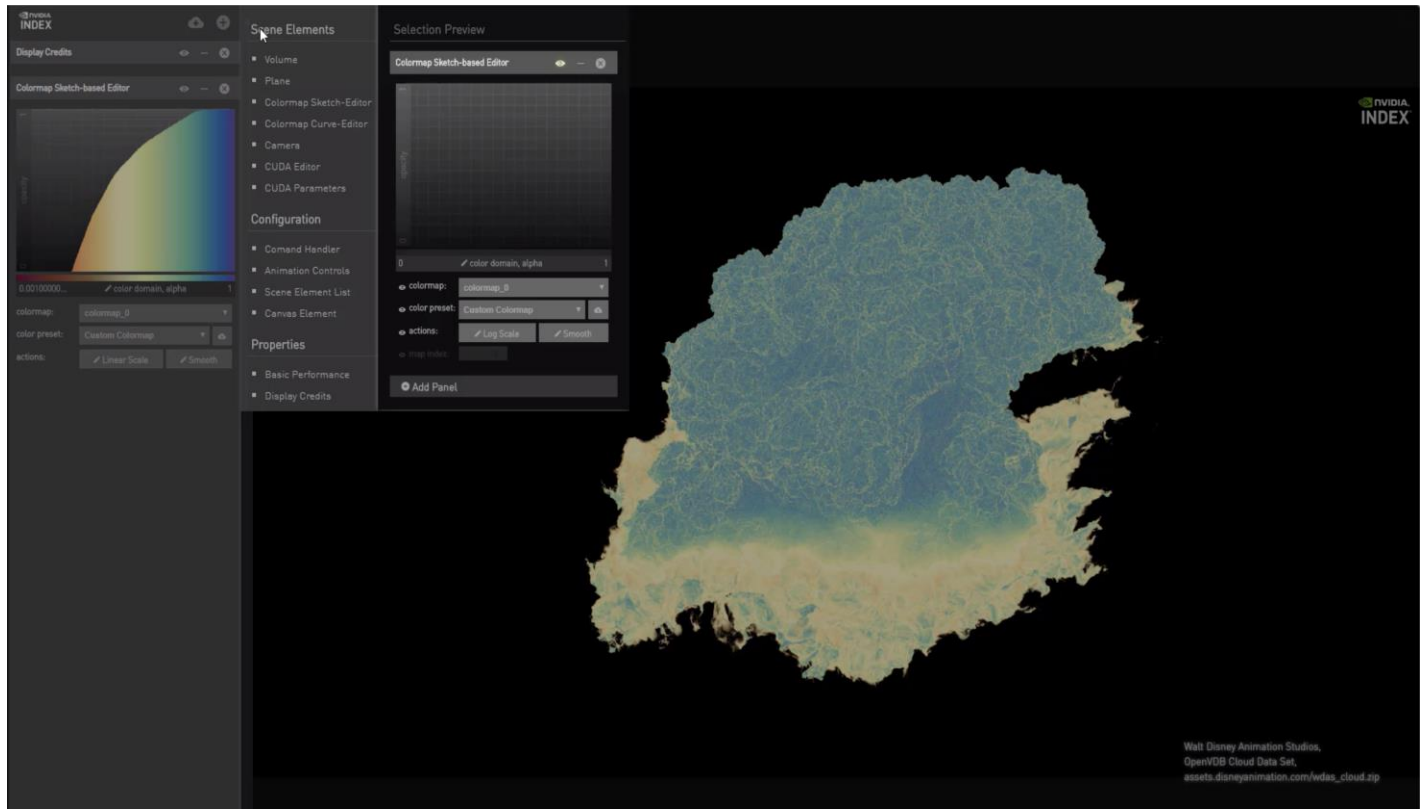
- Volume
- Plane
- Colormap Sketch-Editor
- Colormap Curve-Editor
- Camera
- CUDA Editor
- CUDA Parameters

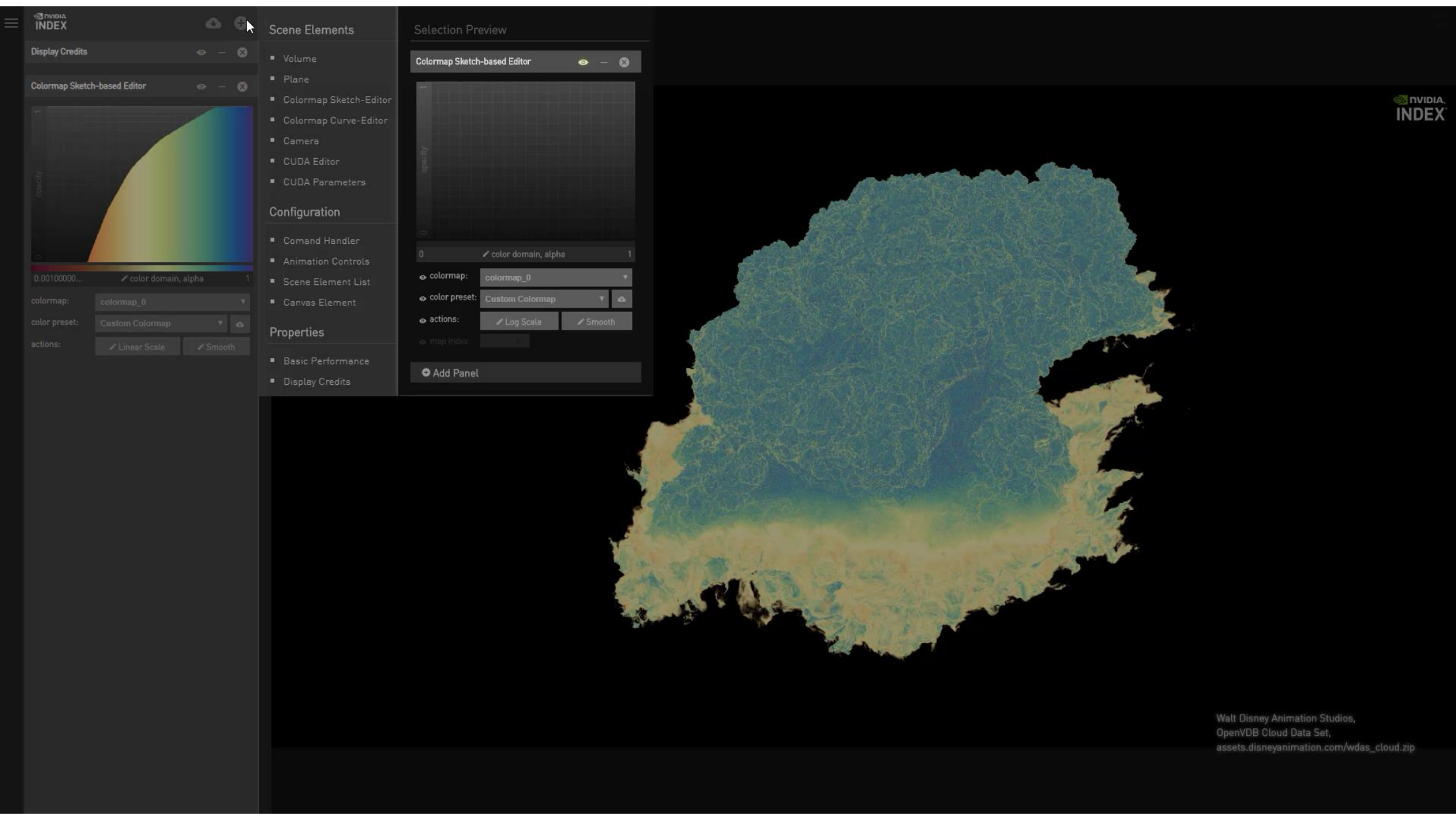
Configurations

- Command Handler
- Animation
- Scene Element List
- Canvas Element

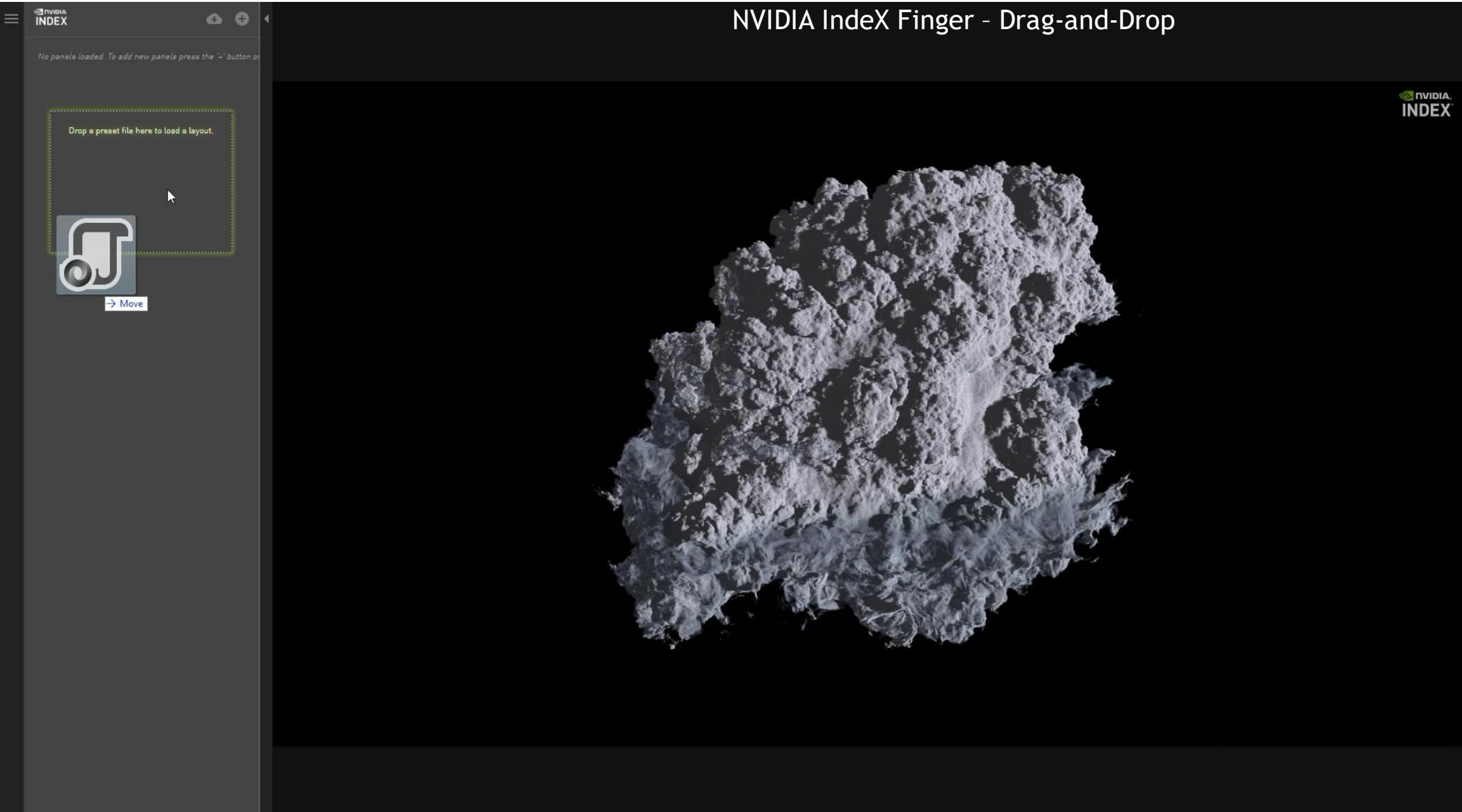
Properties

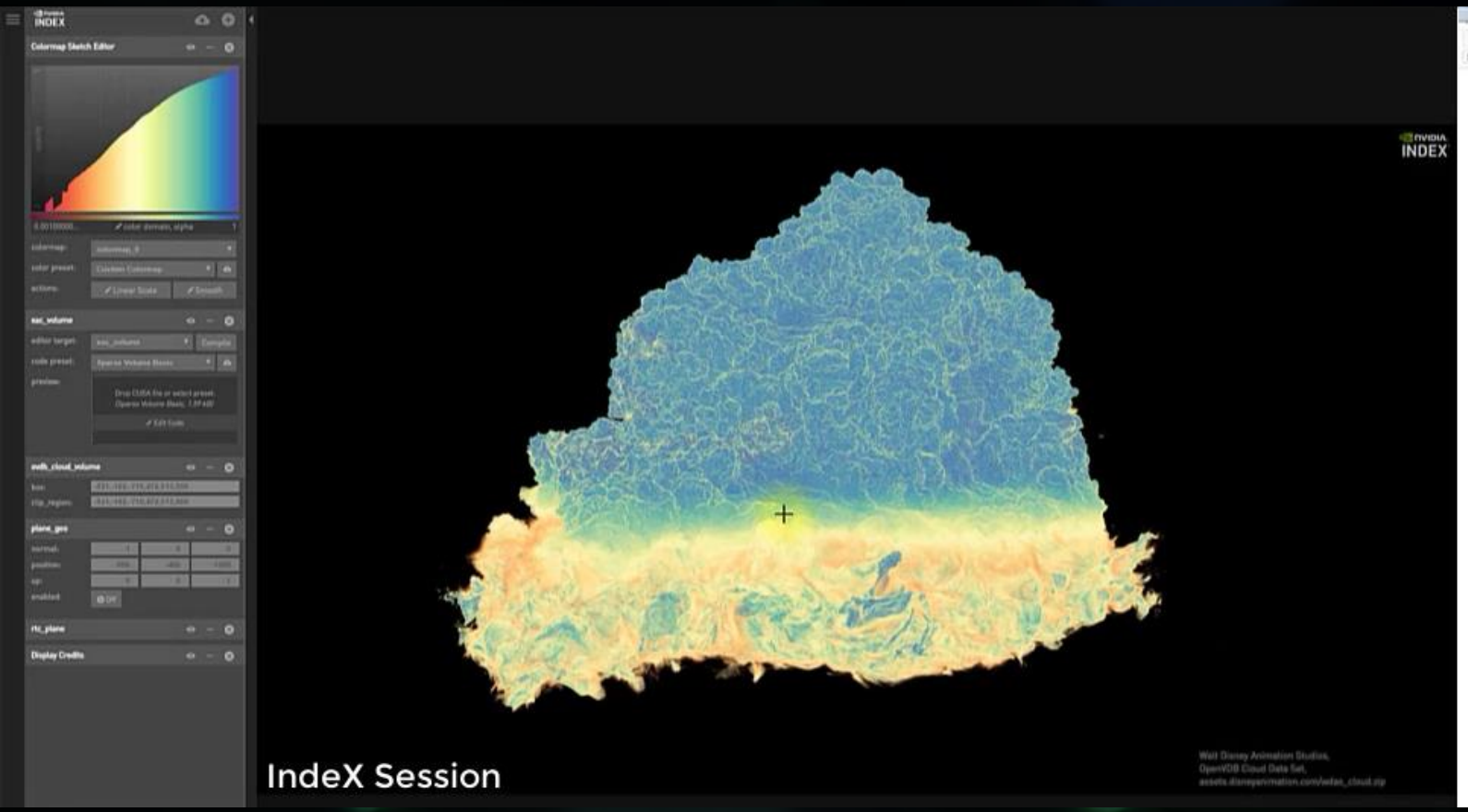
- Basic Performance
- Display Credits



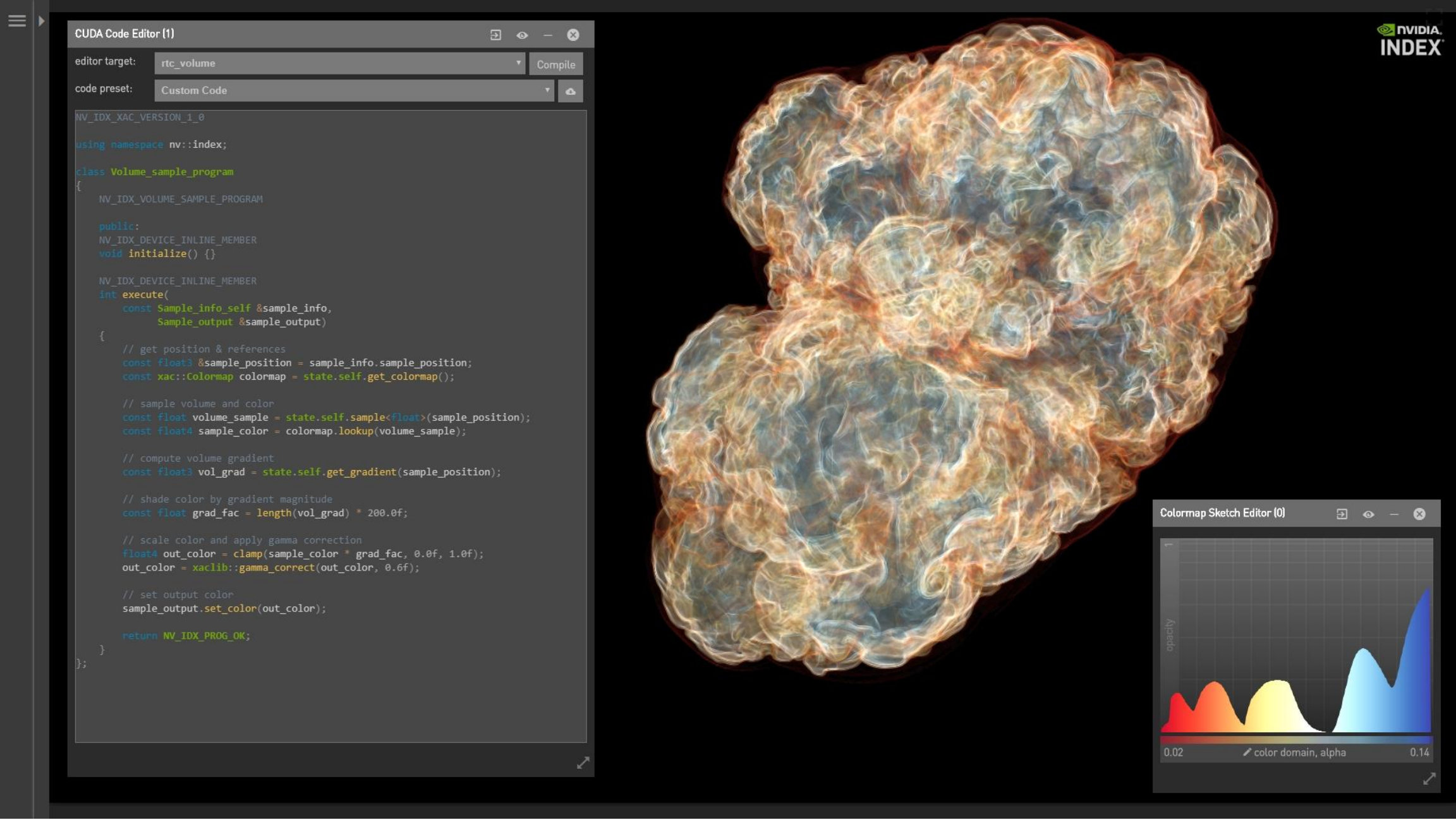

NVIDIA INDEX
Display Credits
Colormap Sketch-based Editor
opacity
0.00100000...
color domain, alpha
1
colormap:
colormap_0
color preset:
Custom Colormap
actions:
Linear Scale
Smooth
Scene Elements
Volume
Plane
Colormap Sketch-Editor
Colormap Curve-Editor
Camera
CUDA Editor
CUDA Parameters
Configuration
Comand Handler
Animation Controls
Scene Element List
Canvas Element
Properties
Basic Performance
Display Credits
Selection Preview
Colormap Sketch-based Editor
opacity
0
color domain, alpha
1
colormap:
colormap_0
color preset:
Custom Colormap
actions:
Log Scale
Smooth
map index:
Add Panel
NVIDIA INDEX
Walt Disney Animation Studios,
OpenVDB Cloud Data Set,
assets.disneyanimation.com/wdas_cloud.zip

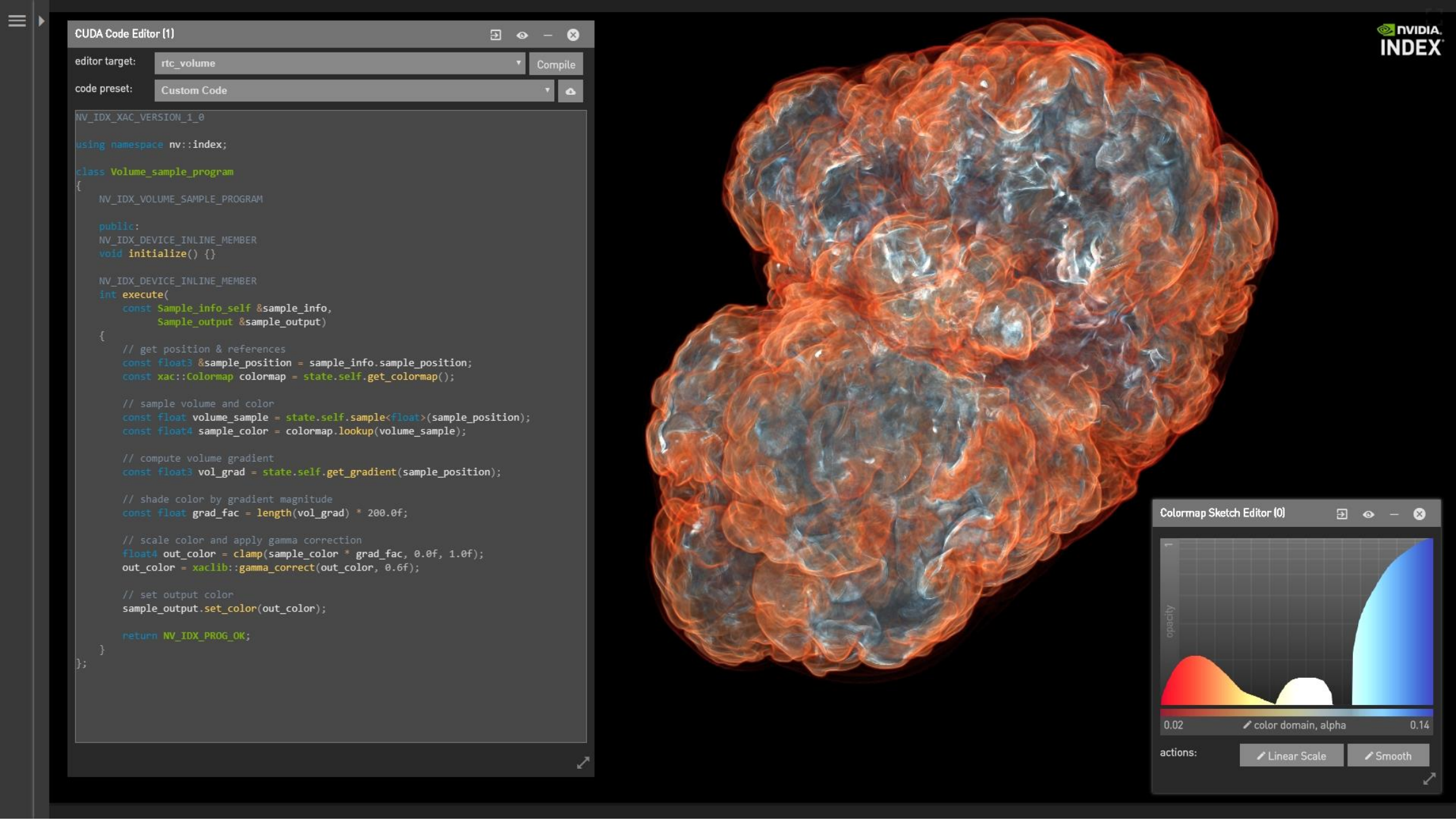
NVIDIA IndeX Finger - Drag-and-Drop

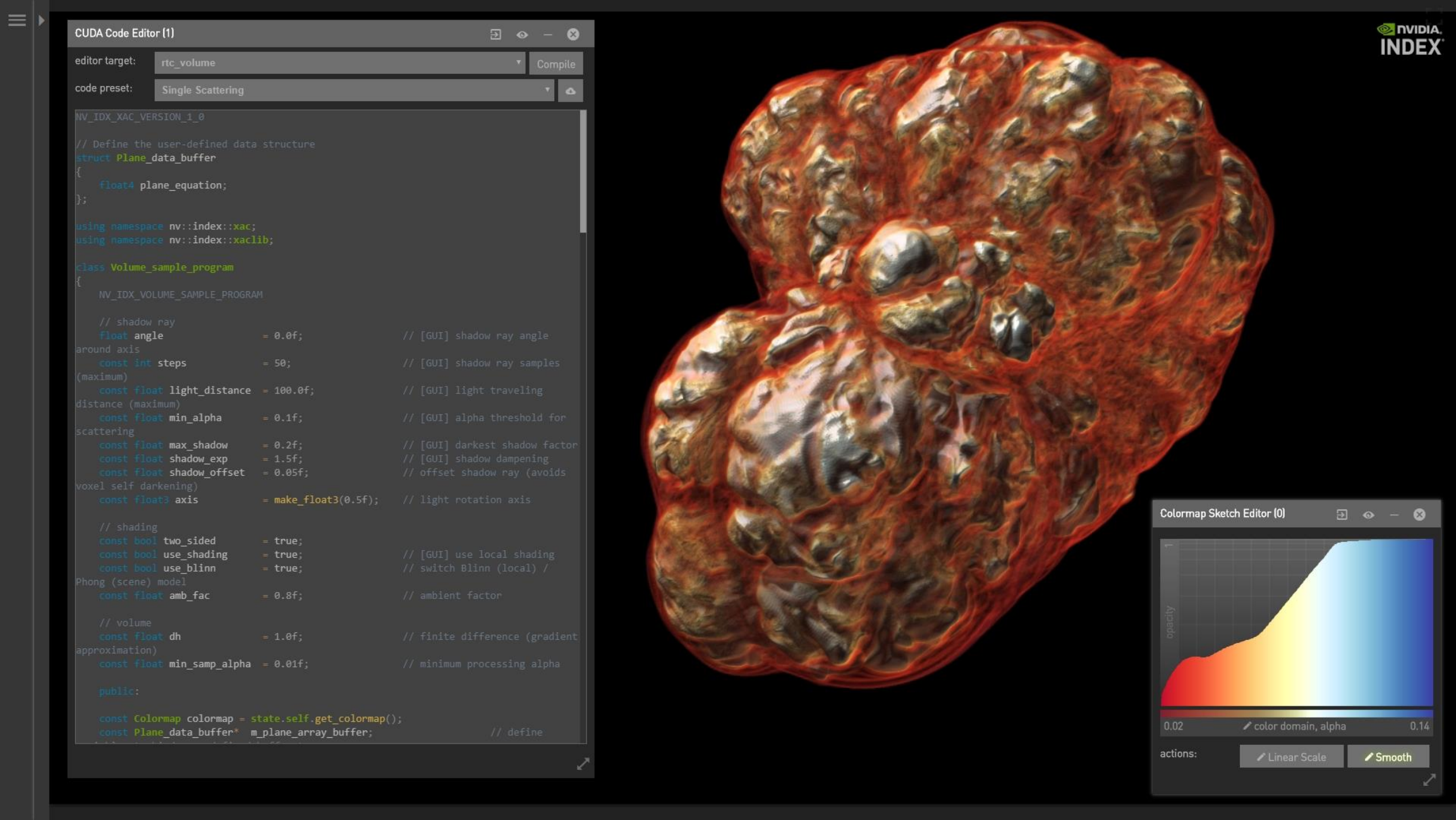


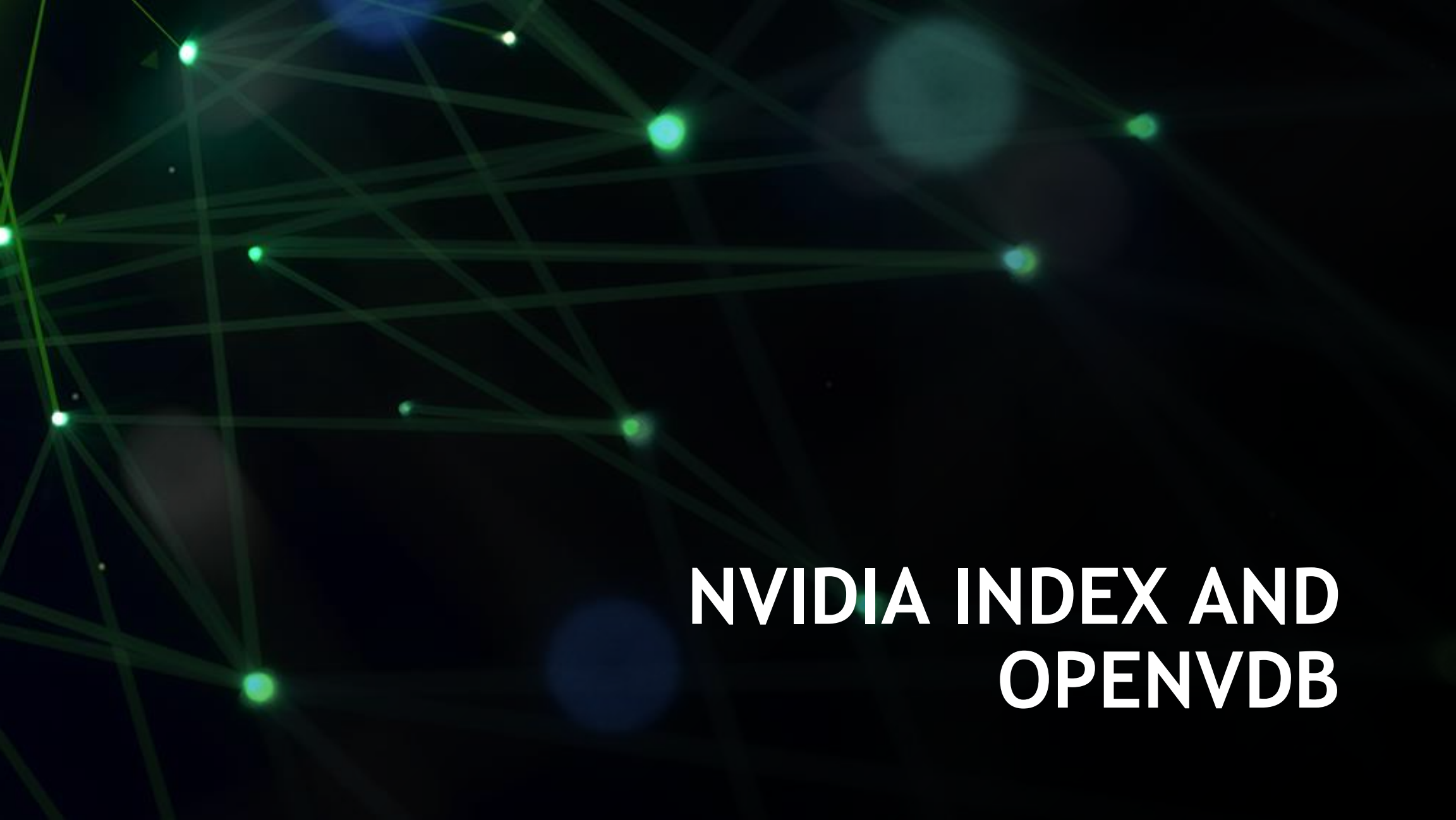

INDEX
Colormap Sketch Editor
NVIDIA
INDEX
IndeX Session


import threading
# websocket setup
ws_url = "ws://127.0.0.1:3333/INDEX_REQ"
step = 0
element_id = "plane_geo"
time_max = 150 # maximum number of time steps
time_step = 0.01 # duration between time step in seconds
h_min = -600
h_max = 500
print(" - websocket: connecting to "+ws_url+" ...")
ws = create_connection(ws_url)
# timer function
def update_plane():
global step
step = step + 1
height = (step / time_max) * (h_max - h_min) + h_min
# assemble websocket command
ws_cmd_02 = """{
'command': 'edit_scene_element',
'edit_scene_element': {
'id': '"""+element_id+"""',
'property':'position',
'value': """+str(height)+""" -400 -1000'
}}"""
# send websocket command
ws_cmd_02 = ws_cmd_02.replace("'","\"")
ws.send(ws_cmd_02)
print(" - step "+str(step)+" of "+str(time_max)+") websocket sending command "+element_id+"...")
if (step <= time_max):
threading.Timer(time_step, update_plane).start()
else:
print(" - websocket: closing websocket connection.")
ws.close()
# create timer object
timer = threading.Timer(time_step, update_plane)
timer.start()
websocket: connecting to ws://127.0.0.1:3333/INDEX_REQ ...
step 1 of 150: websocket sending command plane_geo...
step 2 of 150: websocket sending command plane_geo...
step 3 of 150: websocket sending command plane_geo...
step 4 of 150: websocket sending command plane_geo...
step 5 of 150: websocket sending command plane_geo...
step 6 of 150: websocket sending command plane_geo...
step 7 of 150: websocket sending command plane_geo...
step 8 of 150: websocket sending command plane_geo...
step 9 of 150: websocket sending command plane_geo...
step 10 of 150: websocket sending command plane_geo...
step 11 of 150: websocket sending command plane_geo...
step 12 of 150: websocket sending command plane_geo...
step 13 of 150: websocket sending command plane_geo...
step 14 of 150: websocket sending command plane_geo...
step 15 of 150: websocket sending command plane_geo...
step 16 of 150: websocket sending command plane_geo...
step 17 of 150: websocket sending command plane_geo...
step 18 of 150: websocket sending command plane_geo...
step 19 of 150: websocket sending command plane_geo...
step 20 of 150: websocket sending command plane_geo...
step 21 of 150: websocket sending command plane_geo...
step 22 of 150: websocket sending command plane_geo...
step 23 of 150: websocket sending command plane_geo...
step 24 of 150: websocket sending command plane_geo...
step 25 of 150: websocket sending command plane_geo...
step 26 of 150: websocket sending command plane_geo...
step 27 of 150: websocket sending command plane_geo...
step 28 of 150: websocket sending command plane_geo...
step 29 of 150: websocket sending command plane_geo...
step 30 of 150: websocket sending command plane_geo...
step 31 of 150: websocket sending command plane_geo...
Jupyter Notebook Session

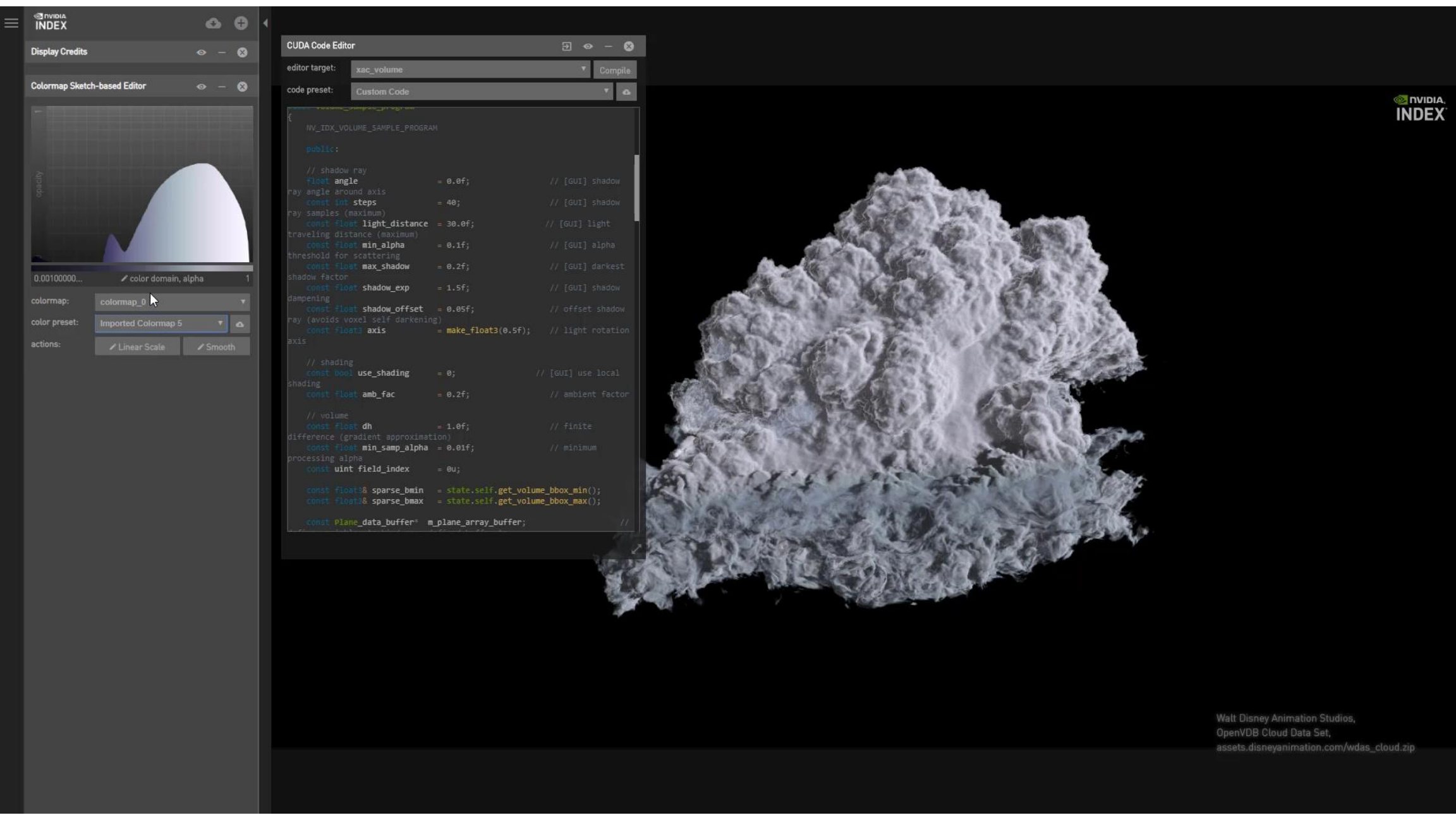

CUDA Code Editor (1)
editor target:
rtc_volume
Compile
code preset:
Custom Code
NV_IDX_XAC_VERSION_1_0
using namespace nv::index;
class Volume_sample_program
{
NV_IDX_VOLUME_SAMPLE_PROGRAM
public:
NV_IDX_DEVICE_INLINE_MEMBER
void initialize() {}
NV_IDX_DEVICE_INLINE_MEMBER
int execute(
const Sample_info_self &sample_info,
Sample_output &sample_output)
{
// get position & references
const float3 &sample_position = sample_info.sample_position;
const xac::Colormap colormap = state.self.get_colormap();
// sample volume and color
const float volume_sample = state.self.sample<float>(sample_position);
const float4 sample_color = colormap.lookup(volume_sample);
// compute volume gradient
const float3 vol_grad = state.self.get_gradient(sample_position);
// shade color by gradient magnitude
const float grad_fac = length(vol_grad) * 200.0f;
// scale color and apply gamma correction
float4 out_color = clamp(sample_color * grad_fac, 0.0f, 1.0f);
out_color = xaclib::gamma_correct(out_color, 0.6f);
// set output color
sample_output.set_color(out_color);
return NV_IDX_PROG_OK;
}
};
NVIDIA INDEX
Colormap Sketch Editor (0)
opacity
0.02
color domain, alpha
0.14

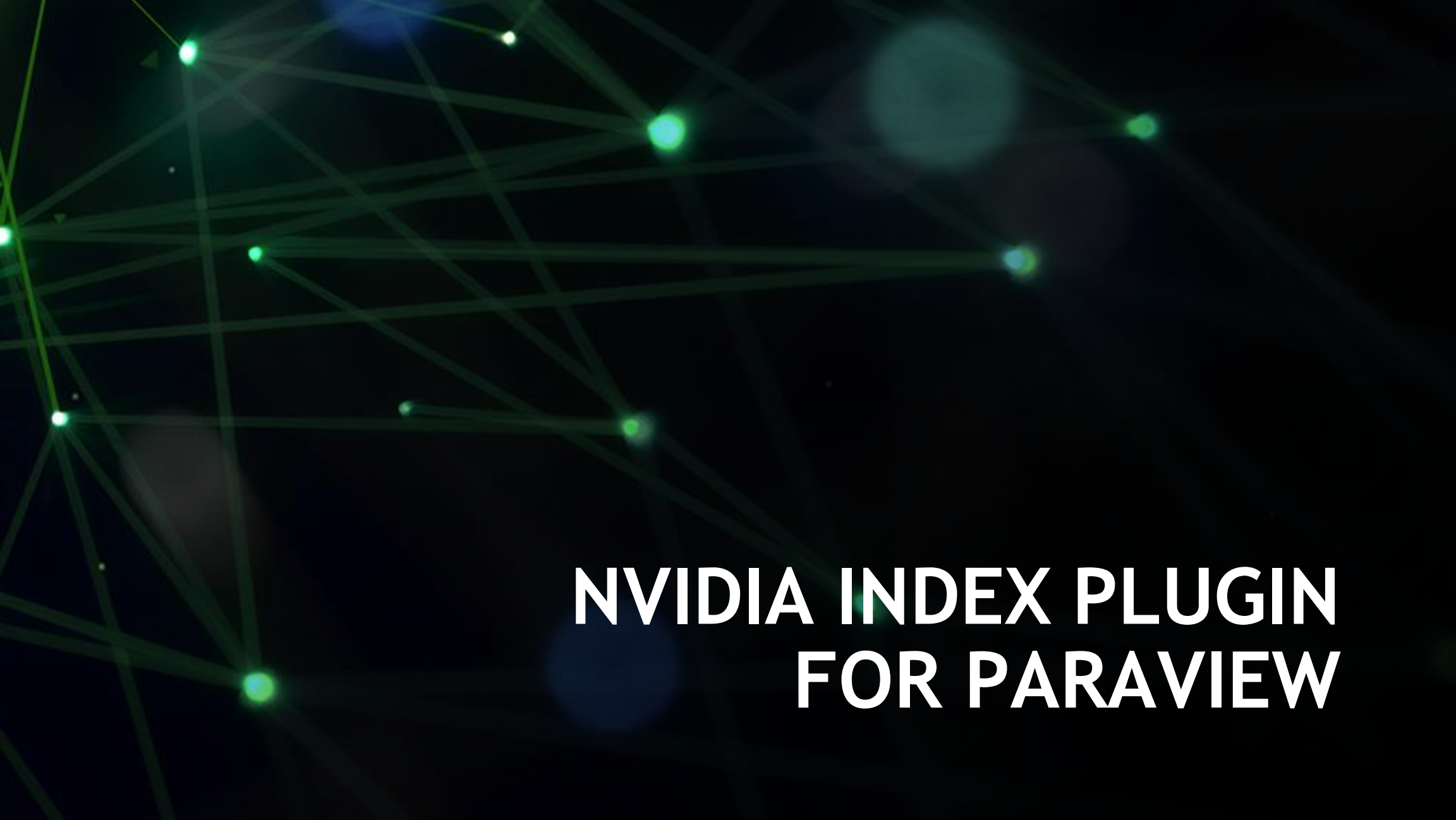

CUDA Code Editor (1)
editor target:
rtc_volume
Compile
code preset:
Custom Code
NV_IDX_XAC_VERSION_1_0
using namespace nv::index;
class Volume_sample_program
{
NV_IDX_VOLUME_SAMPLE_PROGRAM
public:
NV_IDX_DEVICE_INLINE_MEMBER
void initialize() {}
NV_IDX_DEVICE_INLINE_MEMBER
int execute(
const Sample_info_self &sample_info,
Sample_output &sample_output)
{
// get position & references
const float3 &sample_position = sample_info.sample_position;
const xac::Colormap colormap = state.self.get_colormap();
// sample volume and color
const float volume_sample = state.self.sample<float>(sample_position);
const float4 sample_color = colormap.lookup(volume_sample);
// compute volume gradient
const float3 vol_grad = state.self.get_gradient(sample_position);
// shade color by gradient magnitude
const float grad_fac = length(vol_grad) * 200.0f;
// scale color and apply gamma correction
float4 out_color = clamp(sample_color * grad_fac, 0.0f, 1.0f);
out_color = xaclib::gamma_correct(out_color, 0.6f);
// set output color
sample_output.set_color(out_color);
return NV_IDX_PROG_OK;
}
};
NVIDIA INDEX
Colormap Sketch Editor (0)
opacity
0.02
color domain, alpha
0.14
actions:
Linear Scale
Smooth


CUDA Code Editor (1)
editor target:
rtc_volume
Compile
code preset:
Single Scattering
NV_IDX_XAC_VERSION_1_0
// Define the user-defined data structure
struct Plane_data_buffer
{
float4 plane_equation;
};
using namespace nv::index::xac;
using namespace nv::index::xaclib;
class Volume_sample_program
{
NV_IDX_VOLUME_SAMPLE_PROGRAM
// shadow ray
float angle = 0.0f; // [GUI] shadow ray angle around axis
const int steps = 50; // [GUI] shadow ray samples (maximum)
const float light_distance = 100.0f; // [GUI] light traveling distance (maximum)
const float min_alpha = 0.1f; // [GUI] alpha threshold for scattering
const float max_shadow = 0.2f; // [GUI] darkest shadow factor
const float shadow_exp = 1.5f; // [GUI] shadow dampening
const float shadow_offset = 0.05f; // offset shadow ray (avoids voxel self darkening)
const float3 axis = make_float3(0.5f); // light rotation axis
// shading
const bool two_sided = true;
const bool use_shading = true; // [GUI] use local shading
const bool use_blinn = true; // switch Blinn (local) / Phong (scene) model
const float amb_fac = 0.8f; // ambient factor
// volume
const float dh = 1.0f; // finite difference (gradient approximation)
const float min_samp_alpha = 0.01f; // minimum processing alpha
public:
const Colormap colormap = state.self.get_colormap();
const Plane_data_buffer* m_plane_array_buffer; // define
NVIDIA INDEX
Colormap Sketch Editor (0)
opacity
0.02
color domain, alpha
0.14
actions:
Linear Scale
Smooth



NVIDIA INDEX AND OPENVDB


NVIDIA INDEX
Display Credits
Colormap Sketch-based Editor
opacity
0.00100000...
color domain, alpha
1
colormap:
colormap_0
color preset:
Imported Colormap 5
actions:
Linear Scale
Smooth
CUDA Code Editor
editor target:
xac_volume
Compile
code preset:
Custom Code
{
NV_IDX_VOLUME_SAMPLE_PROGRAM
public:
// shadow ray
float angle = 0.0f; // [GUI] shadow ray angle around axis
const int steps = 40; // [GUI] shadow ray samples (maximum)
const float light_distance = 30.0f; // [GUI] light traveling distance (maximum)
const float min_alpha = 0.1f; // [GUI] alpha threshold for scattering
const float max_shadow = 0.2f; // [GUI] darkest shadow factor
const float shadow_exp = 1.5f; // [GUI] shadow dampening
const float shadow_offset = 0.05f; // offset shadow ray (avoids voxel self darkening)
const float3 axis = make_float3(0.5f); // light rotation axis
// shading
const bool use_shading = 0; // [GUI] use local shading
const float amb_fac = 0.2f; // ambient factor
// volume
const float dh = 1.0f; // finite difference (gradient approximation)
const float min_samp_alpha = 0.01f; // minimum processing alpha
const uint field_index = 0u;
const float3& sparse_bmin = state.self.get_volume_bbox_min();
const float3& sparse_bmax = state.self.get_volume_bbox_max();
const Plane_data_buffer* m_plane_array_buffer; //
NVIDIA INDEX
Walt Disney Animation Studios,
OpenVDB Cloud Data Set,
assets.disneyanimation.com/wdas_cloud.zip


NVIDIA INDEX PLUGIN
FOR PARAVIEW

NVIDIA INDEX PLUGIN FOR PARAVIEW

New Release is comming

Updates and new features:

Improved upload time and memory management

Implemented new IndeX instance class as one global instance.

Schedule:

The IndeX plugin release will be synchronized with the new ParaView 5.7 release

The ParaView Release is scheduled for April 2019.

NVIDIA INDEX PLUGIN FOR PARAVIEW

Performance Comparison

NASA Fun3D data

30 million cells, unstructured tetrahedrons
Using ParaView 5.6 & NVIDIA IndeX plugin 2.3

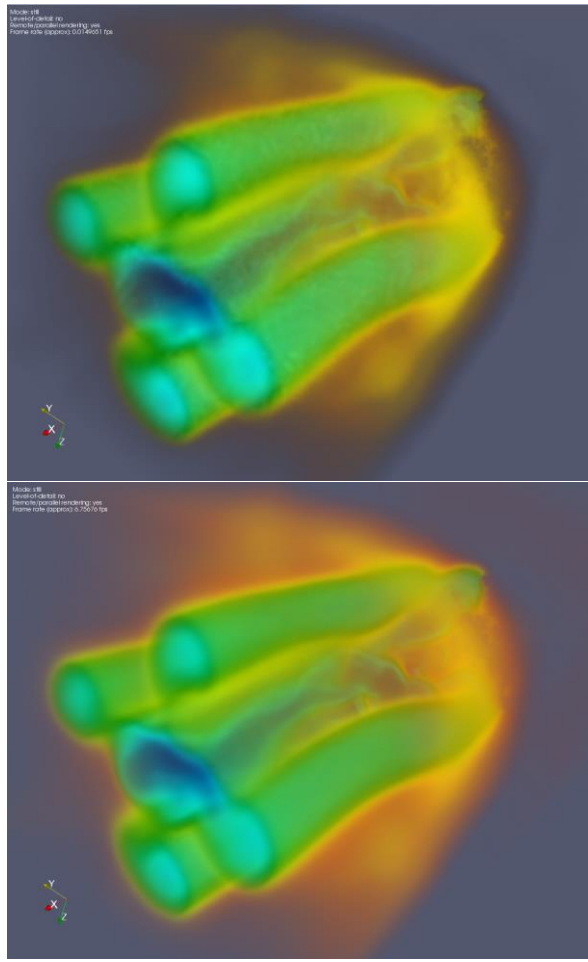
Performance Results:

Colormap change:

ParaView native: 66.38 [sec]
NVIDIA IndeX: 0.051 [sec]

Moving camera:

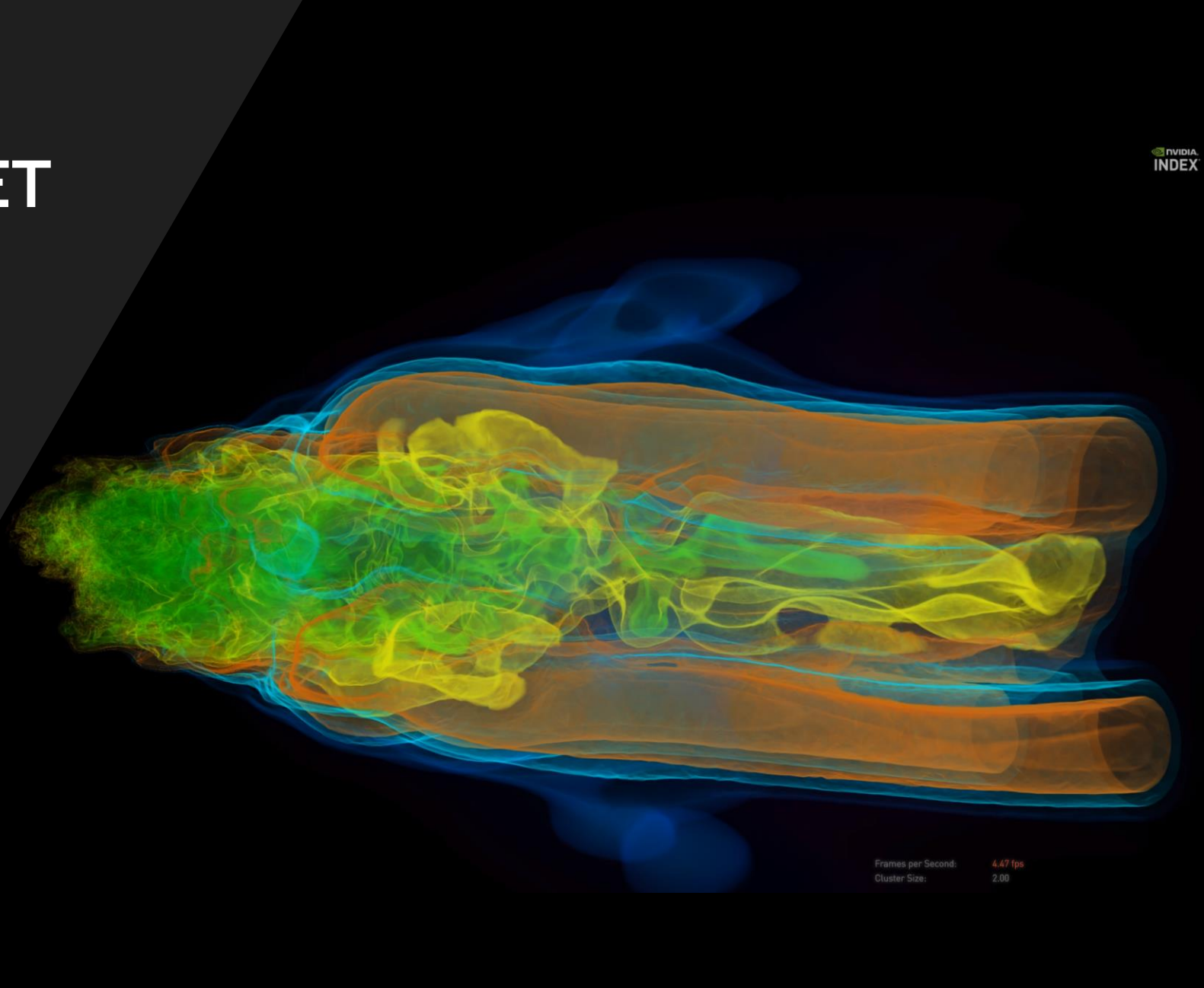
ParaView: 25.01 [sec]
IndeX: 0.044 [sec]



FUN 3D DATASET

NASA

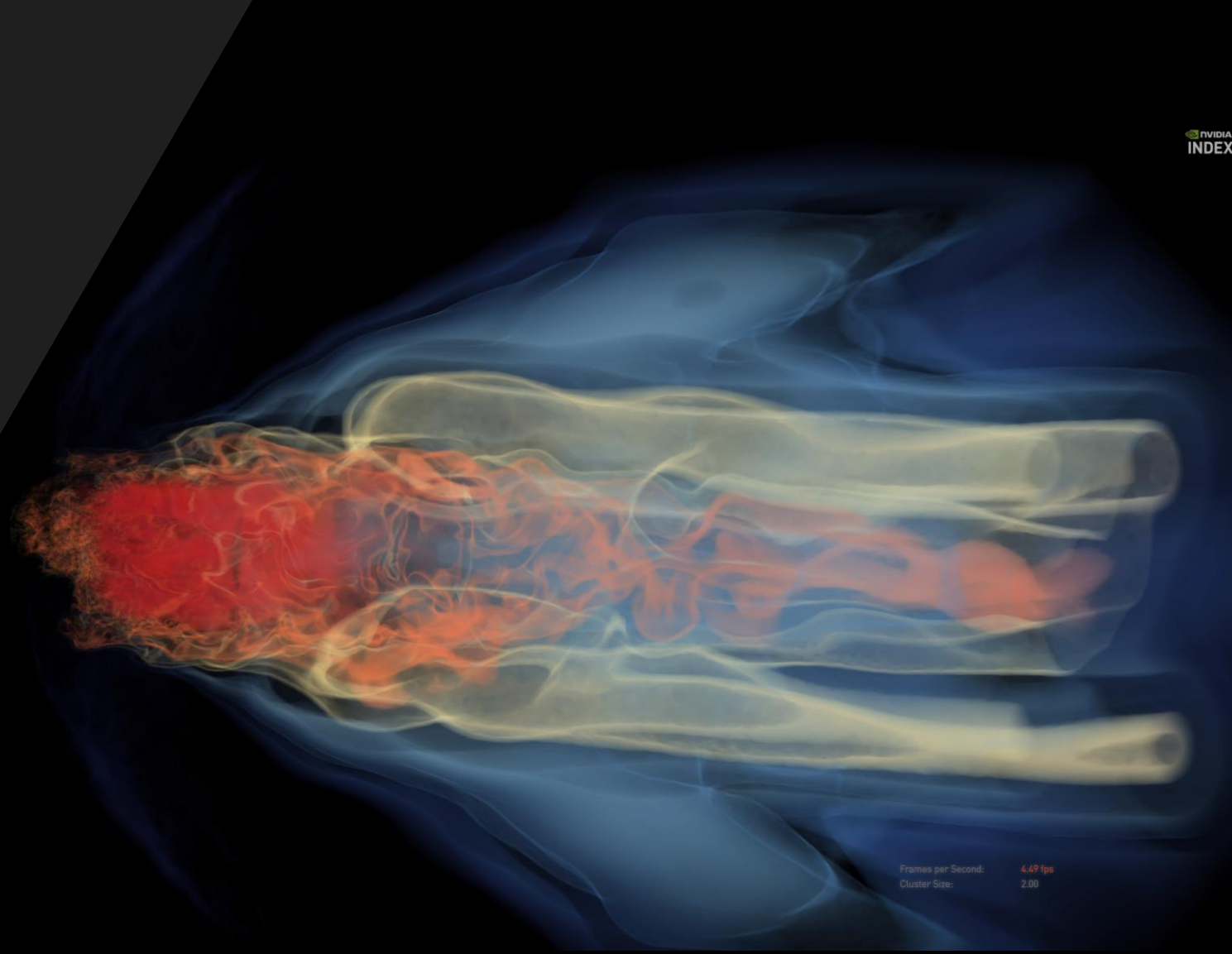
2.5 Giga Cells



FUN 3D DATASET

NASA

2.5 Giga Cells



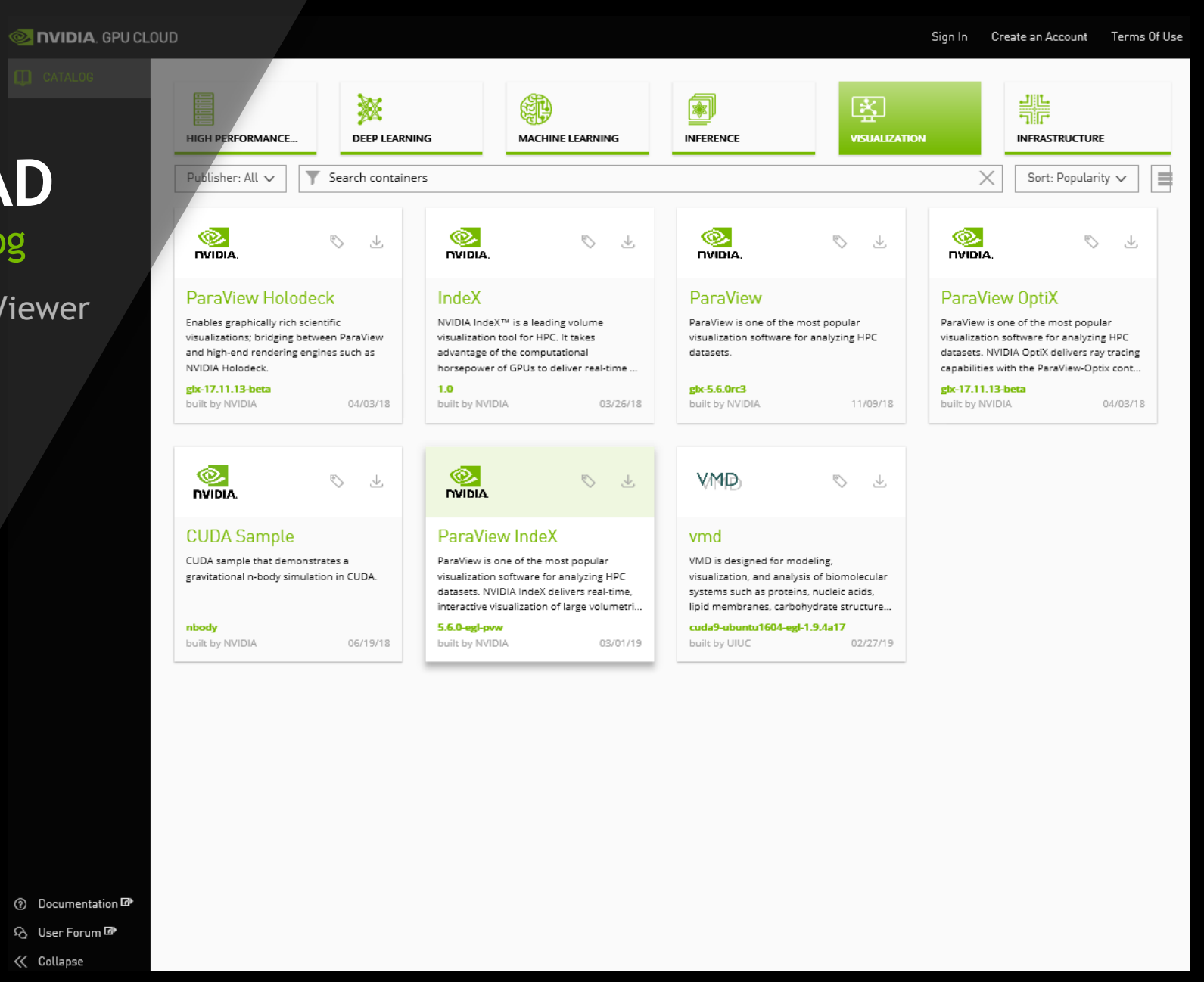

NVIDIA INDEX

NGC DOWNLOAD

NVIDIA GPU Cloud Catalog

NVIDIA IndeX & HTML5-based Viewer

NVIDIA IndeX Paraview Plugin



LATEST NVIDIA INDEX KEY FEATURES

Available Summer 2019

XAC Compute using CUDA programming
Speed up the discovery process

High-Bandwidth Inferencing
AI-guided visual discovery process

IBM Power9 Software Package
Broad platform availability

Corner-Point Grid Datastructure
Oil & Gas reservoir modelling

Enterprise Layer with Plugin Infrastructure
Simplify creation of tailor-made cloud services and solutions

NVIDIA IndeX Finger – HTML5-based Viewer
Enabling tailor-made cloud and enduser solutions

OpenVDB Support
Extending data format support in Media and Entertainment

GALACTIC WINDS WITH NVIDIA INDEX

@ GOOGLE BOOTH
@ NVIDIA BOOTH

CONNECT WITH THE EXPERTS

NVIDIA INDEX TECHNOLOGY FOR HPC VISUALIZATION
TODAY | 5PM – 6PM, HALL 3 POD E (CONCOURSE LEVEL)

PARTNER TALK

ADVANCING ASTROPHYSICS WITH THE GPU-NATIVE, MASSIVELY PARALLEL CODE, CHOLLA
EVAN SCHNEIDER, PRINCETON UNIVERSITY
WEDNESDAY | MAR 20, 1PM - 01:50PM - HILTON HOTEL MARKET ROOM

Marc Nienhaus, NVIDIA
Product Technology Lead NVIDIA IndeX
Sr. Manager Engineering

Tom-Michael Thamm, NVIDIA
Product Manager NVIDIA IndeX
Director Product Management

Brant Robertson
Maureen and John Hendricks Visiting Professor, Institute for Advanced Study
Associate Professor, University of California, Santa Cruz

NVIDIA®


NVIDIA