NVIDIA IndeX

Unboxing NVIDIA IndeX

19 July 2024
Version 2.3

NVIDIA IndeX – Unboxing NVIDIA IndeX

 Document build number rev376353

Contents

Preface

Purpose of this document

The NVIDIA IndeX® framework is intended for large-scale and high-quality volume data visualization. This document describes how to unbox NVIDIA IndeX and render a volumetric dataset.

Audience

This document is intended for scientists, engineers, and other professionals who require high-quality visualization of volumetric datasets of any size for interactive, real-time rendering, exploration, and analysis.

Prerequisites

Unpackaging the NVIDIA IndeX release and rendering a volumetric dataset requires no special skills. A description of the hardware and software prerequisites for running NVIDIA IndeX are included in this document.

Integrating NVIDIA IndeX into your visualization pipeline requires C++ programming skills. This task is covered in the developer documentation shipped with NVIDIA IndeX.

How this document is organized

This document is organized as follows:

- Accelerating scientific discovery (page 2) provides a brief introduction to NVIDIA IndeX.
- System requirements (page 3) describes the hardware and software requirements for running NVIDIA IndeX.
- Installing NVIDIA IndeX (page 4) guides you through download and installation of NVIDIA IndeX.
- Testing your installation (page 5) describes how to load and render an example dataset.
- Importing a dataset (page 7) explains how to load and render your own data.
- Next steps (page 8) offers suggestions for further exploration.
- Frequently asked questions (page 9) provides immediate answers to commonly asked questions.
- This appendix (page 10) provides a listing of a simple scene that you can edit to load your own data.

1 Accelerating scientific discovery

The NVIDIA IndeX® framework sets a new standard for interactive and collaborative exploration and analysis of large-scale scientific visualizations. This document is intended to guide you through the steps for unboxing NVIDIA IndeX and start visualizing and exploring your scientific data.

For additional information about NVIDIA IndeX, see the:

- NVIDIA IndeX 3D Volumetric Visualization Framework[1] page, which provides a brief introduction to NVIDIA IndeX and how teams are using it to solve large-scale visualization challenges and advance scientific understanding.
- NVIDIA Ray Tracing Documentation[2] page, where you can link to or download NVIDIA IndeX documentation.

Note: Setting up the development environment is not part of the scope of this document. For a description of the NVIDIA IndeX SDK and setting up the development environment, see the *IndeX Programmer's Manual*.

1. `https://developer.nvidia.com/nvidia-index`
2. `https://raytracing-docs.nvidia.com`

2 System requirements

Following is a list of system requirements for NVIDIA IndeX:

GPU and driver

At least one CUDA-capable GPU and a driver that is compatible with the CUDA version required by NVIDIA IndeX. See the NVIDIA IndeX release notes for details.

Operating systems

NVIDIA IndeX runs under the following Linux and Windows systems:

- Red Hat Enterprise Linux (RHEL) or CentOS version 7 or newer. Typically, NVIDIA IndeX will run on other Linux distributions.
- Microsoft Windows 10.

3 Installing NVIDIA IndeX

To install NVIDIA IndeX:

1. Fill in and submit the IndeX contact[1] form.

   You will be given instructions for downloading NVIDIA IndeX.

2. After downloading NVIDIA IndeX, unzip the contents to the appropriate directory on your machine.

   For a description of the directories, see the README (`readme.md` or `readme.html`).

1. `https://developer.nvidia.com/index-contact`

4 Testing your installation

The following procedure explains how to test your installation by running the rendering service included with your NVIDIA IndeX download. The rendering service initializes NVIDIA IndeX, loads the test dataset, and renders it. An HTTP server is automatically started and a viewing URL is specified in the log. Multiple people can view and explore the visualization from multiple browser instances using the same URL to enable a collaborative experience.

To test your installation:

1. Open a terminal session if you have not already done so.
2. From the machine where you installed NVIDIA IndeX, switch to the `demo` directory.

   > **Note:** You cannot run the demo from elsewhere because all child scripts and files have relative paths from the `demo` directory.

3. To start the demo, enter the following command at the prompt:
   - Windows: `nvindex-viewer.cmd`
   - Linux: `./nvindex-viewer.sh`

   When rendering is successfuly completed, a viewing URL is displayed in the log.
4. If the process fails, check the log for error messages. Ensure that all errors are fixed. If you are unable to resolve an error, contact NVIDIA for support at `nvidia-index@nvidia.com`.
5. To display a successful rendering result, open a browser session and load the URL displayed in the log. As mentioned already, multiple people can interact with the visualization simultaneously by starting separate browser instances and loading the same URL.

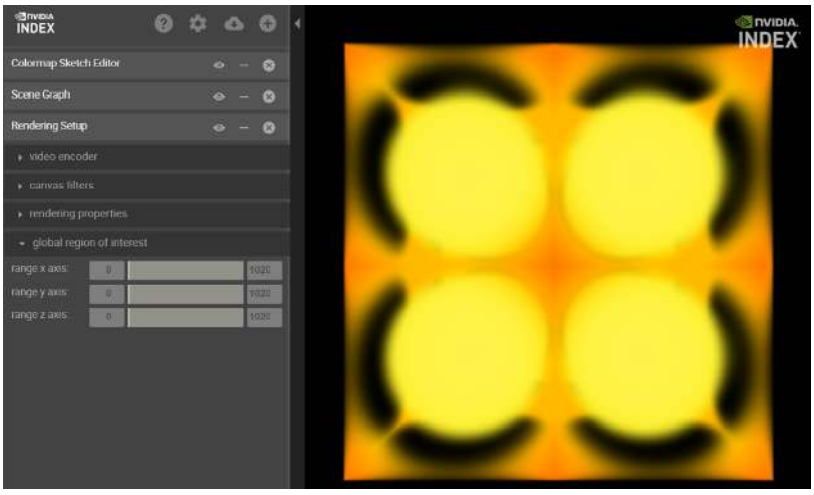

Fig. 4.1 - Initial image rendered from demo dataset.

6. To explore the visualization, manipulate it directly and use the supported functionality displayed to the left of the visualization. For example:
   - *Panning* — Hold down the Ctrl key while dragging the mouse
   - *Zooming* — Hold down the Shift key while dragging the mouse
   - *Rotating* — Drag the mouse

   To add a panel, click the + button at the top of the window.
7. To close the visualization and release the memory, open the [Server] menu (top-left) and click [File ▸ Shutdown server].

5 Importing a dataset

This chapter describes how to import your own data using a simple scene file and use the render service to visualize it. The scene file you create is used by the system at runtime to upload your data and override related settings in the default scene file.

To import and render your own data:

1. Create a simple scene file to import your data:
   1.1. Create an empty text file and name it `svol-simple.prj`.
   1.2. Copy the contents of the scene file listing in the appendix (page 10) into the empty text file.
2. Edit the following settings in `svol-simple.prj` as needed:
   - A volume shader used for shading a volume dataset called `svol_prog`
   - A color map applied to the volume dataset called `svol_cmap`
   - Rendering properties applied to the dataset called `svol_render_props`
   - The dataset with importer information called `seismic_uint8` In particular, you need to specify the following information about the dataset:
     - Location
     - Size
     - Voxel type
3. Save the edited `svol-simple.prj` scene file to the `[main_distribution_directory]/demo` directory.
4. Open a command line session, if you have not already done so.
5. At the command prompt, enter:

   ```
   ./nvindex-viewer.sh --add svol-simple.prj
   ```

   The `--add` option enables you to specify the scene file that you want the system to use to override settings in the default scene file.
6. When rendering is successful, enter the viewing URL from the log into the location field of your browser to view the result.

6 Next steps

This chapter offers some suggestions to expand your hands-on exploration of NVIDIA IndeX capabilities.

6.1 Running NVIDIA IndeX from a cloud service

NVIDIA IndeX is currently available as a cloud service from AWS.[1] The accompanying documentation[2] guides you through the process of setting up a cluster, uploading data, rendering, and viewing results. Example datasets that you can use are available from the Sample Dataset Information[3] page. Instructions are also provided for uploading and rendering your own datasets.

6.2 Exploring the NVIDIA IndeX tutorials

Be sure to take a look at the tutorials, which explore some of the key capabilities of NVIDIA IndeX. You can start the tutorials from the `[main_distribution_directory]/tutorial` using the `./nvindex-tutorial.sh` shell script. Be sure to take a look at the `../tutorial/README` for additional detail before you start.

6.3 Integrating NVIDIA IndeX into your application

For detailed information about integrating NVIDIA IndeX into your application or application pipeline, refer to the NVIDIA IndeX documentation accessible from the NVIDIA Ray Tracing Documentation[4] page.

1. `https://aws.amazon.com/marketplace/pp/prodview-jungamkavzpw2`
2. `https://github.com/NVIDIA/nvindex-cloud/blob/master/doc/eks.md`
3. `https://github.com/NVIDIA/nvindex-cloud/blob/master/doc/datasets.md`
4. `https://raytracing-docs.nvidia.com`

7 Frequently asked questions

Q: Do I need to install the CUDA SDK or any other libraries to use the plugin?

A: The CUDA SDK[1] is required for building example programs or plugins that make use of CUDA. You also need to install the appropriate NVIDIA display driver for your GPU.

Q: Can I render multiple volumes?

A: Yes, the NVIDIA IndeX SDK supports multi-volume rendering.

1. `https://developer.nvidia.com/cuda-zone`

Appendix A Example scene file svol-simple.prj

The following listing is the simple scene file named `svol-simple.prj`. You can use this file to override the settings of the default scene file to load your own data. For more information, see Importing a dataset (page 7).

In this example, long lines are broken with the "\" character to improve formatting, but should be on a single line when used by IndeX as .prj file input.

Words in italics should be replaced by the appropriate value for your installation.

```
#! index_app_project 0
# -*- mode: Conf; -*-

index::region_of_interest = 0 0 0 500 500 1500
app::scene::root::children = sparse_volume_data

app::scene::sparse_volume_data::type = static_scene_group
app::scene::sparse_volume_data::children = \
   svol_render_props svol_cmap rtc_volume volume_dataset

app::scene::rtc_volume::type = rendering_kernel_program
app::scene::rtc_volume::target = volume_sample_program
app::scene::rtc_volume::enabled = true
app::scene::rtc_volume::source_file = sparse_volume_basic.cu

# Setup rendering properties
app::scene::svol_render_props::type = sparse_volume_rendering_properties
app::scene::svol_render_props::filter_mode = trilinear
app::scene::svol_render_props::sampling_distance = 0.5

# The map_type: either procedural or lookup_table
app::scene::svol_cmap::type = colormap
app::scene::svol_cmap::map_type = lookup_table
app::scene::svol_cmap::domain = 0.0 1.0
app::scene::svol_cmap::domain_boundary_mode = clamp_to_edge

# The volume type. A sparse volume is able to manage dense and sparse volume
# datasets as well as multi-resolution data.
app::scene::volume_dataset::type = sparse_volume

# This option selects a specific data importer. The importer reads raw voxel
# data in a given order (see below).
app::scene::volume_dataset::importer = \
```

```
   nv::index::plugin::base_importer.Sparse_volume_importer_raw

# The voxel format. The present dataset's voxels are of type uint8.  Valid types
# are: uint8, uint16, sint16, rgba8, float32.
app::scene::volume_dataset::voxel_format = uint8

# In some cases, volume data is stored in z-first/x-last order. In such cases,
# the option 'app::scene::volume_dataset::convert_zyx_to_xyz' needs to be set to
# 'true' because NVIDIA IndeX sores the volume data in x-first/z-last order in
# memory.
# The present dataset is assumed to be in x-first/z-last order, i.e., no
# conversion required: app::scene::volume_dataset::convert_zyx_to_xyz = false

# The size of the dataset in the datasets local space
app::scene::volume_dataset::size = 500 500 1500

# The bounding box defines the space within which the volume is defined
app::scene::volume_dataset::bbox = 0 0 0 500 500 1500

# Import directory
app::scene::volume_dataset::input_directory = your-directory

# Name of the file
app::scene::volume_dataset::input_file_base_name = your-filename-without-extension

# File extension (including the initial dot character)
app::scene::volume_dataset::input_file_extension = .your-filename-extension

# Cache data on disk for future accelerated data imports
app::scene::volume_dataset::cache = false
```

The value of `app::scene::rtc_volume::source_file` is the filename `sparse_volume_basic.cu`. This file contains a generic sparse volume CUDA program, displayed in Listing A.1:

Listing A.1: CUDA file sparse_volume_basic.cu

```
class Volume_sample_program
{
   NV_IDX_VOLUME_SAMPLE_PROGRAM
   const nv::index::xac::Colormap colormap = state.self.get_colormap();

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)
      {
         using namespace nv::index;

         const auto& svol = state.self;
         const auto svol_sampler =
            svol.generate_sampler<float,xac::Volume_filter_mode::TRILINEAR>(
               0u,
               sample_info.sample_context);

         const float v = svol_sampler.fetch_sample(
            sample_info.sample_position_object_space);

         sample_output.set_color(colormap.lookup(v));

         return NV_IDX_PROG_OK;
      }
};
```