Large-Scale Cinematic Visualization Using Universal Scene Description

Mark A. Bolstad*
Sandia National Laboratories

ABSTRACT

Universal Scene Description (USD) is the core of Pixar Animation Studios graphics and rendering pipeline. Released as open source in 2016, USD provides for the interchange of elemental assets (e.g. models), but additionally enables assembly and organization of any number of assets into higher level concepts, such as scenes or shots. These assets can be non-destructively edited or modified through layering that can then be transmitted between applications. Exporting scientific simulations through the USD API allows access to a wide range of tools for creating cinematic visualizations or for viewing scientific simulations in an augmented/virtual reality environment.

Index Terms: Human-centered computing—Visualization—Visualization systems and tools—Visualization toolkits; Computing methodologies—Computer graphics—Graphics systems and interfaces—Graphics file formats

1 INTRODUCTION

Traditionally, scientific visualization uses simple rendering techniques such as Gouraud or Phong shaded polygons or ray sampled transfer functions for volume rendering to reduce complex numerical data into images or animations. Newer hardware has enabled the use of computationally more expensive rendering techniques such as ambient occlusion, but the goal of these systems and techniques is high interactivity to speed insight into complex data. Cinematic Visualization is the process of applying rendering techniques developed for film and entertainment to enhance visualizations or to provide a context for communication to a wider audience, e.g., a globe model for climate data. One of the difficulties in creating cinematic visualizations is bridging the divide between scientific formats and those formats that evolved for Digital Content Creation (DCC) tools such as Maya or Houdini. Universal Scene Description (USD) [6] from Pixar Animation Studios is one possible bridge.

This paper presents a framework for generating USD files by post-processing scientific data using the Visualization Toolkit (VTK) [4], and ParaView [1]. Using USD's "composition engine" we create USD layers that seamlessly author scenes from multi-threaded output such as that generated in-situ using ParaView Catalyst [2], and layer additional information such as data ranges and color-mapped data values in a compact disk footprint. A ParaView plugin is under development that will allow for the generation of in-situ USD files through Catalyst.

2 UNIVERSAL SCENE DESCRIPTION

Universal Scene Description (USD), released to the open source community in 2016, is a core component of Pixar Animation Studio's graphics and rendering pipeline. An overview of USD [6] describes its capabilities as:

> USD provides for interchange of elemental assets (e.g. models) or animations. But unlike other interchange packages, USD also enables assembly and organization of any number of assets into virtual sets, scenes, and shots, transmit them from application to application, and non-destructively edit them (as overrides), with a single, consistent API, in a single scenegraph. USD provides a rich toolset for reading, writing, editing, and rapidly previewing 3D geometry and shading. In addition, because USD's core scenegraph and "composition engine" are agnostic of 3D, USD can be extended in a maintainable way to encode and compose data in other domains.

*e-mail: mbolsta@sandia.gov

IEEE LDAV 2019
21 October, Vancouver, BC, Canada
978-1-7281-2605-0/19/$31.00 ©2019 IEEE

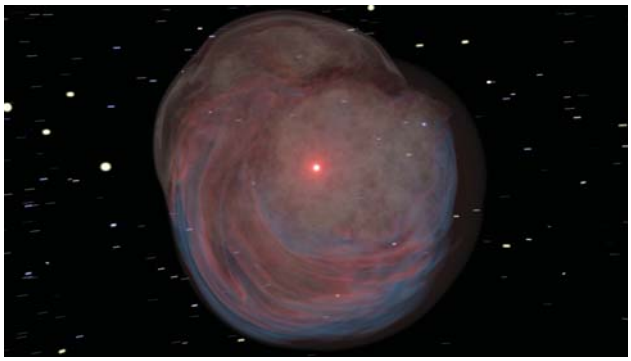

Figure 1: A visualization of a supernova explosion [3]. The 430^3 data was converted to USD using the python API and OpenVDB on a single core of a MacBook Pro in 99 secs and then animated and rendered using Houdini.

USD provides an API for creating scenes and geometry, and provides a high-performance OpenGL based application for previewing the generated files. The generated files can be an easy to read and edit in their ASCII format (see figure 2, for an example file and the geometry it describes), and a binary format that is compact and allows for non-linear access.

USD has been widely adopted across a wide variety of DCC tools (Maya, Houdini, and Katana with Blender and 3DMax support under development) and other entertainment tools such as Unity and Unreal Engine. As such, it makes an attractive target to bridge the gap between scientific data and tools for cinematic visualization.

3 METHODOLOGY

In the post-processing method, datasets are converted into USD files through either the C++ or Python API. Since our data will generally fall under the category of write-once, read-many, we generate an initial base-layer USD file consisting of the initial geometry (points, connectivity, and other attributes such as normals) and a "variant set"[1] of raw scalar and vector values. Additionally, a separate entity is added to the USD file that holds the minimum and maximum of each of the data values. If desired, a second script can be run that generates a new USD file that references the original file and turns the scalar values into colors by mapping them through a VTK or ParaView colormap. This step is usually performed when the

[1] A USD variant set is a collection of attributes of which only one is loaded and active at runtime

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. Approved for unlimited release: SAND2019-10380 C.

```
def Xform "World
{
   def Mesh "mesh_0"
   {
      float3[] extent.timeSamples = {
         1: [(-0.5, -0.5, -0.5), (0.5, 0.5, 0.5)],
      }
      int[] faceVertexCounts.timeSamples = {
         1: [4, 4, 4, 4, 4, 4],
      }
      int[] faceVertexIndices.timeSamples = {
         1: [1, 5, 4, 0, 2, 6, 5, 1, 3, 7, 6, 2, 0, 4, 7, 3, 2, 1, 0, 3, 5, 6, 7, 4],
      }
      point3f[] points.timeSamples = {
         1: [(-0.5, -0.5, -0.5), (0.5, -0.5, -0.5), (0.5, -0.5, 0.5), (-0.5, -0.5, 0.5),
             (-0.5, 0.5, -0.5), (0.5, 0.5, -0.5), (0.5, 0.5, 0.5), (-0.5, 0.5, 0.5)],
      }
      color3f[] primvars:displayColor (
         interpolation = "vertex"
      )
      color3f[] primvars:displayColor.timeSamples = {
         1: [(1, 0, 0), (1, 0.5, 0), (1, 1, 0), (0, 1, 0),
             (0, 1, 1), (0, 0, 1), (1, 0, 1), (1, 0.5, 1)],
      }
   }
}
```

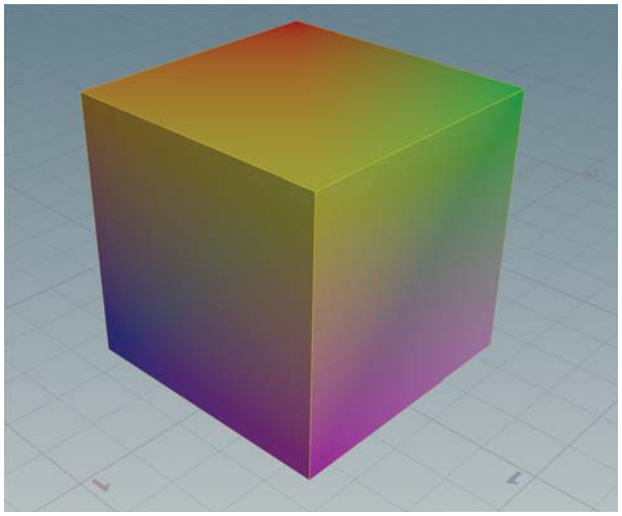

Figure 2: A simple ASCII USD file and the geometry it describes.

user wants the data mapped through a colormap that is difficult to reproduce in a DCC tool, e.g., ParaView's default red, white, and blue divergent colormap.

3.1 Geometry Types

We support all of the VTK surface element types (point, line, triangle, quad, ...) and natively support uniform[2] and unstructured grids and volumes. Unstructured grids are written to USD by traversing the VTK data cell by cell and exporting the faces of each cell. An auxiliary structure is used during traversal to prevent duplicating faces that are common to two cells. As duplicate cell faces are found they are removed from the list. By traversing the grid using an expanding wavefront, only the cells on the active wavefront boundary needed to be stored in the auxiliary list reducing the amount of memory needed.

Uniform grids, or VTK Image data, are natively supported as they can be directly converted into OpenVDB [5] files and referenced from within the USD file. Since OpenVDB data is voxel centric, cell-based VTK scalars convert directly. To accommodate point-based data we convert the data to cell-based by shifting the OpenVDB voxels one-half a cell and pad the volume by one additional voxel in each dimension. Since OpenVDB only allocates cells for non-empty voxels ("empty" is a user-defined value), OpenVDB files are typically smaller than the original data. In figure 1, the original VTK data is 430^3 voxels and consumes 322MB of disk, while the OpenVDB file has the same number of voxels and is 263MB.

Currently, we can support other grid and volume types, but only through conversion to one of the previously listed types. Each conversion has its own major pitfalls:

- Converting from an implicit format to an explicit format (unstructured) results in a huge increase in memory and storage requirements

- Converting to a uniform grid will typically involve undersampling the original data missing potentially important data features, or having to use a small enough voxel size to capture the important features resulting in the explosion in storage requirements mentioned above

[2]uniform = VTK Image Data

3.2 Time-Series Data

USD encodes time-series data by explicitly enumerating the values for each element. In general, values are held at one time value until the next value is encountered. If only one time value is specified, that value is held for the entire animation sequence. Not only does this allow for time varying scalars on a static mesh, it allows for the mixing of datasets and grids that have different time stepping requirements and start and stop values. In addition, we have the ability to specify velocities for each grid position, which allows USD to interpolate a deforming grid between time steps.

4 Conclusion

USD is an effective data format for encoding scientific data in preparation for cinematic visualization. Using the USD composition engine we can create a seamless whole dataset from small structured parts. As such, we are working on a ParaView plugin that will allow Catalyst to create USD files in-situ from simulations. In addition, we are working with the USD development team to enable a more seamless integration of the other volumetric grids that are common in simulation.

References

[1] J. P. Ahrens, B. Geveci, and C. C. Law. Paraview: An end-user tool for large-data visualization. In *The Visualization Handbook*, pp. 717–731. Academic Press / Elsevier, 2005. doi: 10.1016/B978-012387582-2/50038-1

[2] U. Ayachit, A. C. Bauer, B. Geveci, P. O'Leary, K. Moreland, N. Fabian, and J. Mauldin. Paraview catalyst: Enabling in situ data analysis and visualization. In *ISAV@SC*, pp. 25–29. ACM, 2015. doi: 10.1145/2828612.2828624

[3] J. M. Blondin, A. Mezzacappa, and C. DeMarino. Stability of standing accretion shocks, with an eye toward core-collapse supernovae. *The Astrophysical Journal*, 584(2):971–980, Feb 2003. doi: 10.1086/345812

[4] Kitware, Inc. *The Visualization Toolkit User's Guide*, January 2003.

[5] K. Museth, J. Lait, J. Johanson, J. Budsberg, R. Henderson, M. Alden, P. Cucka, D. Hill, and A. Pearce. Openvdb. *ACM SIGGRAPH 2013 Courses on - SIGGRAPH '13*, 2013. doi: 10.1145/2504435.2504454

[6] Pixar Animation Studios. *Introduction to USD*, August 2019.