



Leveraging NVIDIA Omniverse for In Situ Visualization

Mathias Hummel^(✉) and Kees van Kooten

NVIDIA, Santa Clara, USA
{mathiash,kvankooten}@nvidia.com

Abstract. Typical in situ visualization approaches involve rendering images of the simulation data in step with the simulation itself, using in situ visualization tools such as ParaView Catalyst, VisIt libsim, or SENSEI. For these approaches, one has to determine visualization parameters such as camera perspective, color maps, or scene properties in advance. The resulting frames can later be combined to produce animations, but leave little room for improving the presentation.

Alternatively, simulation data can be distilled in situ to a geometric representation, which is then transferred to a workstation and used for live exploration and visualization.

Producing high-quality animations, for example for outreach purposes, typically requires a somewhat tedious process of exporting the geometry to different formats and postprocessing using dedicated modelling, rendering, or compositing software.

We propose a method that allows interactive, high-quality visualization of distilled simulation geometry. Omniverse is NVIDIA's collaboration platform for 3D production pipelines. It is integrated with a number of freely and commercially available 3D software packages and game engines and enables content creators to work on different aspects of models or entire scenes simultaneously.

By integrating ParaView and Catalyst with the Omniverse, the visualization geometry becomes immediately accessible to a number of 3D content authoring and rendering tools without the requirement of invasive software changes in situ or tedious postprocessing and conversion workflows.

Since Omniverse can directly communicate with game engines, the visualization can be augmented using advanced features such as game physics simulation to improve insight and enhance intuition. We demonstrate this by placing architectural building models on a surface that is deformed by an earthquake-like wave. This makes it possible to immediately assess the impact of the earthquake on buildings in a live, interactive way, while leveraging the advanced rendering capabilities of the game engine.

Keywords: In situ visualization · NVIDIA Omniverse · ParaView · Catalyst

1 Introduction

In the past years, the available computing capability has been outpacing the capacity for I/O. This leaves scientists no choice but to severely limit the amount of data stored as the simulation runs. In situ visualization is a broad term for approaches that do not rely on writing simulation data to persistent storage and then transferring them elsewhere (e.g. a dedicated visualization cluster or even a workstation), but instead perform visualization tasks *in situ*, alongside the simulation.

One approach to in situ visualization is to render images as simulation data are produced, alongside with the simulation. Once an image (or a set of images) is available, the original data can be discarded. The produced images can then be collected and combined into animated videos. With such an approach, fundamental parameters such as camera positions and angles, color coding of physical quantities, and the visualization pipeline itself, have to be set in advance of the simulation. Since the full data are no longer available, there is little opportunity for improving the visualization after the fact.

A different approach consists of setting up a visualization pipeline, and then storing the output of this pipeline. The result is a time series of geometries, which are typically considerably smaller in size than the original simulation output, and can, therefore, be produced at a higher temporal frequency. These geometries are then available to be used with established visualization tools.

Offline visualization purposes including outreach, presentation, and collaborative exploration in virtual reality, are typically facilitated by 3D authoring tools and rendering engines, which are not themselves geared towards scientific visualization. Therefore, they typically require a sequence of file transfers and format conversions before the data can be rendered. Especially in the presence of time-varying data, this process can be tedious and time-consuming.

In the area of media and entertainment, the NVIDIA Omniverse collaboration platform aims to solve similar issues by integrating with a number of established tools, game engines, and renderers [13]. We have implemented the Omniverse bridge for in situ visualization, an adapter that integrates ParaView Catalyst with the Omniverse.

Section 2 describes the in situ visualization challenges that we are addressing in this paper, and lists some relevant previous efforts at NVIDIA. In Sect. 3, we provide a short overview of the NVIDIA Omniverse collaboration platform. The main contribution of this paper, the Omniverse bridge, is introduced in Sect. 4. We present two examples of applying the Omniverse bridge to visualization data in Sect. 5, before concluding our results in Sect. 6.

2 In Situ Frameworks and Workflows

A number of in situ frameworks are available to researchers that make it possible to attach widely used visualization tools to simulation codes with minimal manual effort [5]. Typically, a small piece of adapter code has to be written that

runs in step with the simulation and, for each time step (or with a predefined frequency), converts output of the simulation to a format that can be ingested by the visualization software and fed into a visualization pipeline. Examples for such tools include *libSim* [10], which interfaces with *VisIt* [7], *Catalyst* [4] for *ParaView* [3], and *SENSEI* [2], which provides a more general approach that can be used in conjunction with a number of existing in situ frameworks. For each time step ingested by the in situ framework, a visualization pipeline is executed that applies a range of filters, and produces distilled geometry for the purpose of visualization. One option is to render this geometry from a number of predefined view angles, and for a variety of visualization parameters such as color maps. The resulting images can be assembled into image databases for interactive exploration, for example using *ParaView Cinema* [1]. Alternatively, the distilled geometry can be streamed directly to a workstation for live viewing. Also, it is often saved to persistent storage for later retrieval and processing.

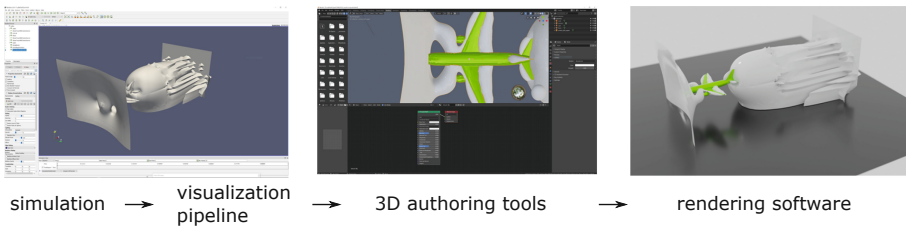


Fig. 1. For outreach and presentation, the output of the visualization pipeline (left) is often converted and imported into 3D authoring applications (center), where a scene with context geometry is put together. Finally, this scene is then rendered using cinematic rendering software. This example uses *Blender* [6] for scene setup and rendering.

For outreach and presentation purposes, where high visual quality is desired, the distilled geometry is often converted to formats ingestible by established 3D authoring tools from the media and entertainment world. These authoring tools are then used to set up scenes and animations, together with background and context geometry. From this, high-quality images and videos are produced using cinematic rendering software. Such an approach requires a number of data transfer and format conversion steps, which can consume considerable amounts of time and effort. These steps have to be repeated whenever changes are made to the underlying data or visualization pipeline.

For interactive and collaborative exploration in virtual reality (VR), game engines are often used to ensure sufficient performance and appealing visual quality. These applications are fundamentally subject to the same challenges as outreach and presentation, in that visualization geometry has to be transferred and converted before being added to a VR scene. Achieving the desired interactivity often requires even more work on the conversion side, by having to break up a geometry into logical pieces and attaching certain physics properties.

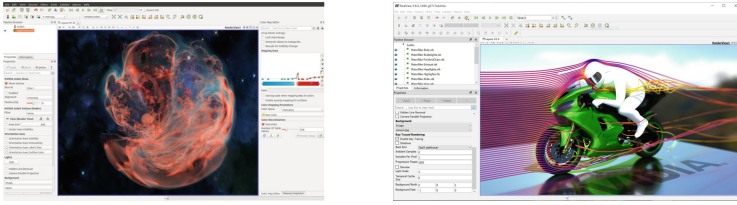


Fig. 2. Left: NVIDIA IndeX can be used inside ParaView for scalable, high-quality, GPU-accelerated volume rendering. Right: VisRTX provides GPU-accelerated path tracing inside ParaView, enabling advanced rendering with reflections, global illumination, and physically-based MDL materials.

At NVIDIA, several efforts have been made previously that make high-quality visualizations more accessible to researchers. Several technologies have been integrated with ParaView, and can already aid in situ visualization. The NVIDIA IndeX volume rendering framework [12] for scalable, GPU-accelerated volume rendering can be directly used in conjunction with ParaView through the NVIDIA IndeX Plugin (Fig. 2, left). VisRTX [15], an open source library for scientific visualization rendering, leverages the NVIDIA OptiX hardware-accelerated ray tracing engine [14] to produce high-quality, path-traced images. The ParaView integration of VisRTX (Fig. 2, left) is scheduled to be publically available as part of an upcoming release of ParaView. In addition, the Unreal game engine was coupled with ParaView to support virtual reality applications with visualization data without the need for separate transfer and format conversion [9].

3 NVIDIA Omniverse

In the field of media and entertainment, artists often work independently on different parts of a scene within a movie or video game. They use a variety of industry-standard tools to create and modify assets. Before such changes can be collectively visualized, they have to be reconciled through a tedious process consisting of data transfer, format conversions, imports, and compositing. NVIDIA Omniverse, which was announced in March 2019, is a collaboration platform for 3D production pipelines that makes it possible for artists to work independently on different parts of a scene, using a wide range of tools, while supporting live updates and changes across those tools¹. Assets such as geometry and materials are accessed centrally through the Omniverse server, which can be run locally or hosted externally. Content authoring and manipulation applications are directly integrated with the Omniverse, so that changes are immediately reflected on all other connected clients. In addition to content authoring tools, the Omniverse is also integrated with game engines and cinematic rendering software, which virtually eliminates the need for tedious data transfer and format conversion.

¹ Further information about NVIDIA Omniverse is available at <https://developer.nvidia.com/nvidia-omniverse>.

```
#usda 1.0
(
  upAxis = "Y"
  defaultPrim = "mesh"
)

def Mesh "mesh"
{
  uniform bool doubleSided = 1
  int[] faceVertexCounts = [3, 3]
  int[] faceVertexIndices = [0, 1, 2, 0, 2, 3]
  normal3f[] normals = [(0, 0, 1), (0, 0, 1), (0, 0, 1), (0, 0, 1)]
  point3f[] points = [(-1, 1, 0), (-1, -1, 0), (1, -1, 0), (1, 1, 0)]
}
```

Fig. 3. A minimal example of a USD mesh in the human-readable, text-based `usda` format.

To achieve this, assets and scenes are represented using an extended version of Pixar’s Universal Scene Description (USD) [17]. Figure 3 shows a minimal example of the human-readable `usda` format; typically, assets are stored and transferred in binary or compressed form. Changes are pushed to the Omniverse through incremental updates. In addition, the Omniverse supports material descriptions written in the Material Definition Language (MDL) [16] for physically based materials. The Omniverse comes with a reference viewing and editing application, *Omniverse Kit*, which utilizes the newest features of current GPUs to produce high-quality visuals at maximum performance.

4 The Omniverse Bridge

Some of the challenges addressed by the Omniverse in media and entertainment also apply to scientific visualization, especially when data transfer and format conversions are required. We propose to utilize the Omniverse as a ‘Rosetta Stone’ for in situ visualization geometry, to allow the direct combination of

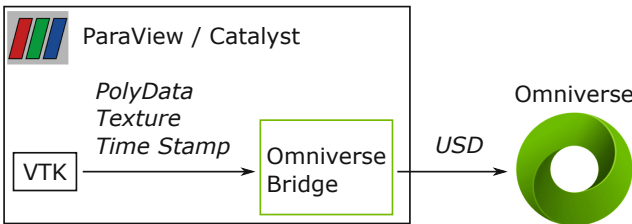


Fig. 4. The Omniverse bridge converts distilled visualization geometry from the in situ visualization framework to USD, and pushes the results into the Omniverse.

scientific visualization data with creative tools, assets, and engines. To this end, we have implemented the Omniverse bridge, an adapter that connects in situ visualization tools with the Omniverse. Distilled geometry, textures, and color maps are converted on-the-fly to USD and pushed to the Omniverse as they become available (Fig. 4). If the visualization geometry is annotated with time step information, time steps are represented as time samples in USD.

Currently, the implementation is integrated with ParaView, where the Omniverse bridge acts as a specialized render view. Any geometry that is flagged to be ‘rendered’ using the Omniverse bridge render view is automatically converted to USD and transferred into the Omniverse. In a Catalyst script, this view can be registered with the coprocessor in the same way as a regular render view would be used to render and save images.

Even though the current implementation supports ParaView and Catalyst, the concept of the Omniverse bridge itself is sufficiently generic to make integration with other in situ visualization frameworks straight-forward.

5 Examples

To provide a first impression of how the Omniverse can be used to address the mentioned challenges of in situ visualization, we have applied the Omniverse bridge to two example scenarios.

Airliner Flow Simulation: The Airliner example consists of an OpenFOAM [8, 18] simulation of flow around an A320 jet aircraft. ParaView Catalyst was used to compute an isosurface of the pressure field in situ, which was then pushed into the Omniverse using the Omniverse bridge together with the aircraft boundary mesh. The visualization geometries from the Omniverse were combined with some planes serving as contextual background in Omniverse Kit. A polished aluminum material, described in MDL, was applied to the aircraft mesh. When the visualization geometry that is fed into the Omniverse bridge is modified, for example by choosing a different isovalue, the changes are automatically reflected in Omniverse Kit. Advanced rendering methods such as ray traced reflections, ray traced ambient occlusion, and ray traced shadows provide high-quality visuals at real-time frame rates.

“Earthquake:” The “Earthquake” sample is not based on an actual simulation, but instead employs a programmable filter inside ParaView to apply a time-varying, wave-shaped distortion to a planar surface, which is then pushed into the Omniverse (Fig. 5, bottom row). As the filter produces time samples of the surface, the USD representation is updated by the Omniverse bridge, and differential updates are transferred to the Omniverse server. A client, such as Omniverse Kit, can play back all currently available time samples of the surface. As additional time samples arrive, these are incorporated into the animation.

In Omniverse Kit, a number of buildings are placed on the ground surface. Both the building models and the ground surface are registered with the particle-based NVIDIA FleX engine for real time physics simulation [11]. Playing back

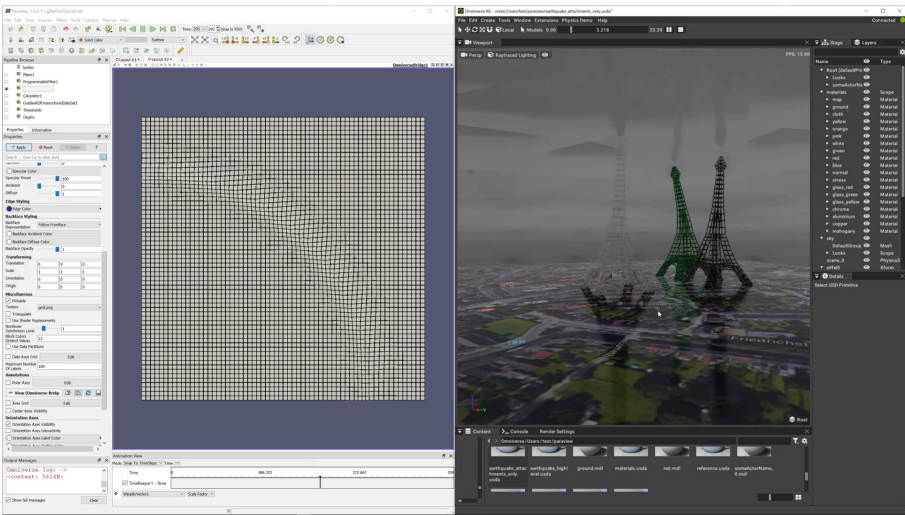
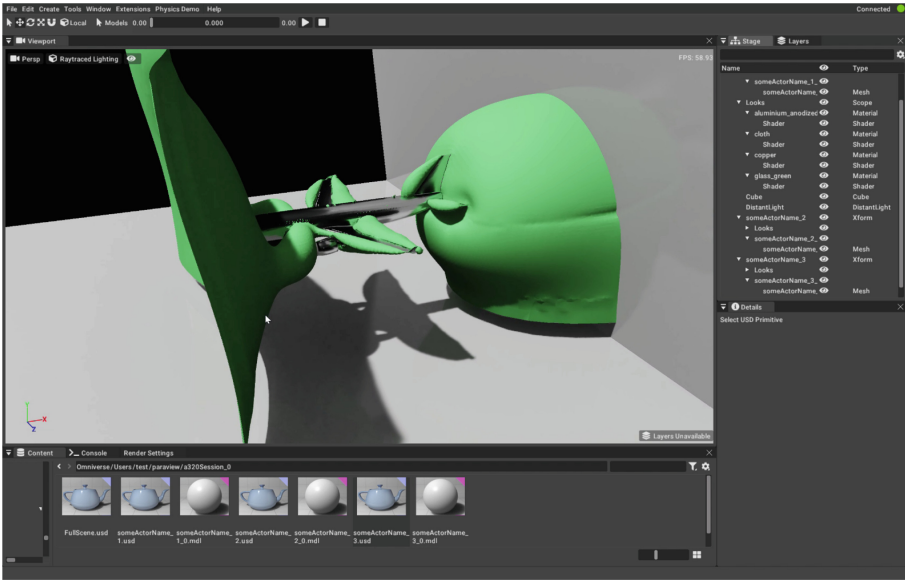


Fig. 5. Two examples making use of the Omniverse bridge. Top: An isosurface of the pressure volume and the airliner boundary mesh, both pushed into the Omniverse by the Omniverse bridge, are visualized using the Omniverse Kit application. Materials and background geometry were added in the Omniverse Kit; the underlying visualization geometry is updated when it changes in ParaView/Catalyst. Bottom row: A programmable filter is used in ParaView to generate a time-varying, wave-shaped distortion on a planar surface (left). The resulting, time-varying mesh is pushed into the Omniverse using the Omniverse bridge. The mesh is used as an animated ground model in Omniverse Kit (right), where a number of physics-enabled building models are attached to it (Map Data: Google, GeoBasis-DE/BKG). As the animation is played back, the wave passes underneath the buildings and causes them to sway.

the surface animation results in the base of the buildings being deformed as the ground wave passes underneath them. The physics simulation model then causes the buildings to sway and wobble, providing an immediate, visual impression of how the ground wave's impact.

6 Conclusion and Future Work

We have discussed some of the challenges associated with in situ visualization, specifically when it comes to data transfer and format conversion for live visualization, outreach, and presentation. We have shown how the NVIDIA Omniverse collaboration platform can help address these challenges by acting as a 'rosetta stone' that removes the need for tedious transfers and conversions. As a proof of concept, we have introduced the Omniverse bridge, which interfaces with in situ visualization frameworks such as ParaView Catalyst. We have provided two examples to show that, using the Omniverse bridge, visualization geometry can be pushed into the Omniverse with minimal additional effort. We have shown that this geometry can be immediately accessed, combined with additional context geometry and materials, and rendered in another Omniverse client. In addition, we have shown how time-varying geometry produced in situ and pushed into the Omniverse can serve as input for real time game physics simulation.

In the future, we plan to apply the Omniverse bridge to more elaborate settings, for example involving live collaboration, large geometries, and virtual reality. Further, we would like to discover applications of game physics engines, involving features such as rigid bodies and fluid simulation, in real-world applications.

Acknowledgments. The authors would like to thank CFD SUPPORT LTD. for providing the Airliner OpenFOAM case: <https://www.cfdsupport.com/download-cases-a320.html>.

References

1. Ahrens, J., Jourdain, S., OLeary, P., Patchett, J., Rogers, D.H., Petersen, M.: An image-based approach to extreme scale in situ visualization and analysis. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2014, pp. 424–434, November 2014. <https://doi.org/10.1109/SC.2014.40>
2. Ayachit, U., Whitlock, B., Wolf, M., Loring, B., Geveci, B., Lonie, D., Bethel, E.W.: The SENSEI generic in situ interface. In: 2016 Second Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV), pp. 40–44, November 2016. <https://doi.org/10.1109/ISAV.2016.013>
3. Ayachit, U.: The ParaView Guide: A Parallel Visualization Application. Kitware Inc., New York (2015)
4. Ayachit, U., et al.: ParaView catalyst: enabling in situ data analysis and visualization. In: Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization, ISAV 2015, pp. 25–29. ACM, New York (2015). <http://doi.acm.org/10.1145/2828612.2828624>

5. Bauer, A.C., et al.: In situ methods, infrastructures, and applications on high performance computing platforms. *Comput. Graph. Forum* **35**(3), 577–597 (2016). <https://doi.org/10.1111/cgf.12930>
6. Blender Online Community: Blender - a 3D modelling and rendering package. Blender Foundation (2019). <http://www.blender.org>
7. Childs, H., et al.: VisIt: an end-user tool for visualizing and analyzing very large data. In: *High Performance Visualization-Enabling Extreme-Scale Scientific Insight*, pp. 357–372, October 2012
8. Jasak, H., Jemcov, A., Tukovic, Z., et al.: OpenFOAM: a C++ library for complex physics simulations. In: *International Workshop on Coupled Methods in Numerical Dynamics*, vol. 1000, pp. 1–20. IUC Dubrovnik Croatia (2007)
9. van Kooten, K.: Bridging scientific visualization and unreal VR. In: Sherman, W.R. (ed.) *VR Developer Gems*, chap. 9. AK Peters/CRC Press (2019)
10. Kuhlen, T., Pajarola, R., Zhou, K.: Parallel in situ coupling of simulation with a fully featured visualization system. In: *Proceedings of the 11th Eurographics Conference on Parallel Graphics and Visualization (EGPGV)* (2011)
11. NVIDIA: FleX. <https://developer.nvidia.com/flex>. Accessed 19 July 2019
12. NVIDIA: IndeX. <https://developer.nvidia.com/index>. Accessed 19 July 2019
13. NVIDIA: Omniverse. <https://developer.nvidia.com/nvidia-omniverse>. Accessed 19 July 2019
14. NVIDIA: OptiX. <https://developer.nvidia.com/optix>. Accessed 19 July 2019
15. NVIDIA: VisRTX. <https://github.com/NVIDIA/VisRTX>. Accessed 19 July 2019
16. NVIDIA: NVIDIA material definition language 1.3 (2016)
17. Pixar: Pixar animation studios open sources Universal Scene Description (2016). <https://graphics.pixar.com/usd/docs/Open-Source-Release.html>. Accessed 19 July 2019
18. Weller, H.G., Tabor, G., Jasak, H., Fureby, C.: A tensorial approach to computational continuum mechanics using object-oriented techniques. *Comput. Phys.* **12**(6), 620–631 (1998)