Blender on Frontier

Michael Sandoval

HPC Engineer - User Assistance Group
Oak Ridge Leadership Computing Facility (OLCF)
Oak Ridge National Laboratory (ORNL)

June 28, 2023

ORNL is managed by UT-Battelle LLC for the US Department of Energy

Overview

- What is Blender?
- It can be used for science?
- Frontier usage
  - Benchmark comparisons
  - Limitations
  - Potential workflows
  - How to install
  - Visualizations produced on Frontier
- Conclusions and Key Takeaways
- Additional Resources

Blender

https://www.blender.org/

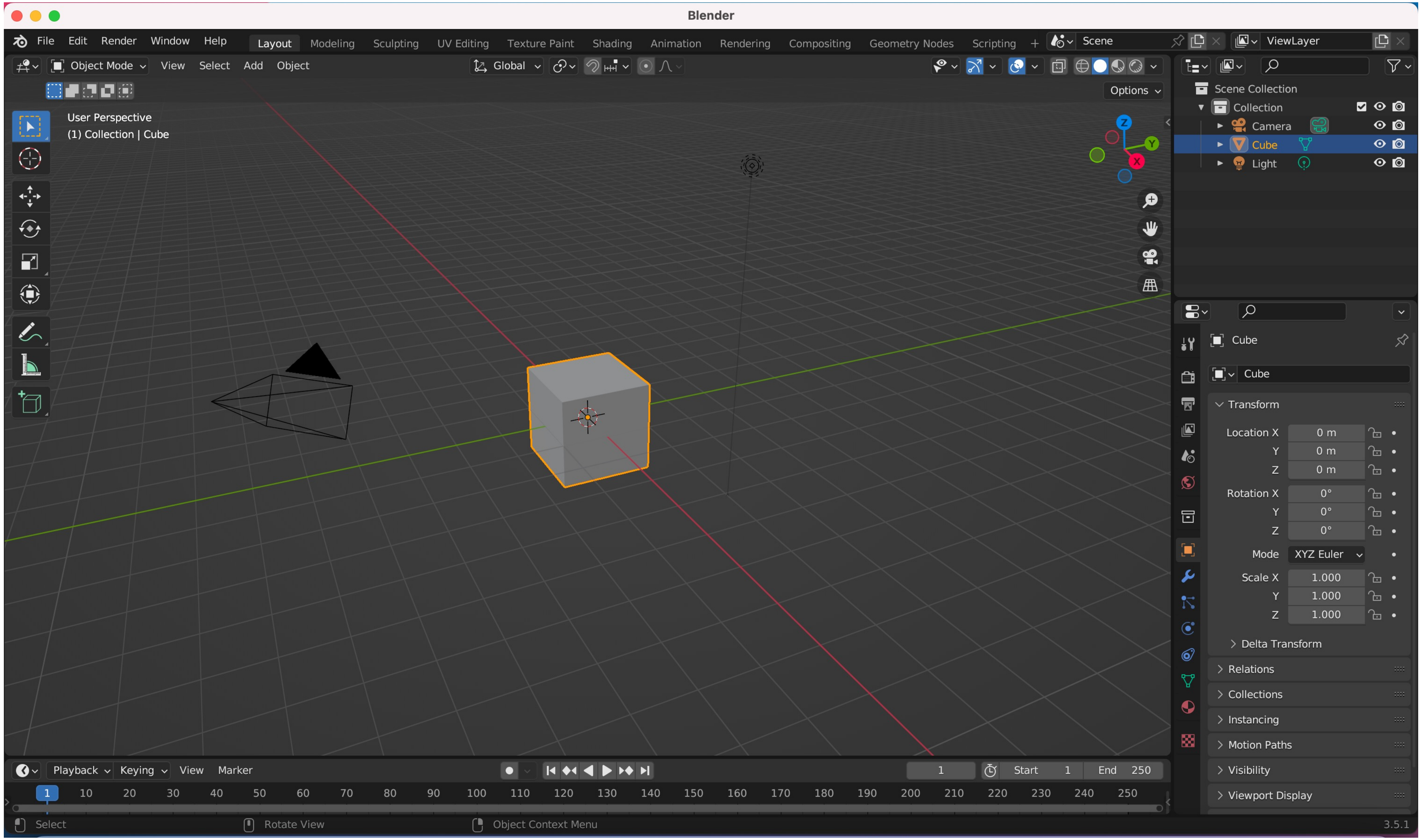
- Free, open-source software for 3D visualization
  - Animation
  - Modeling
  - Video games
  - **Rendering**
  - **Scripting**
  - ***Simulations***
- Python based API
- Cross-platform for Linux, Windows, and Mac
- Supports AMD (HIP) and NVIDIA (CUDA, OptiX)

Image credit: studio.blender.org, Oksana Dobrovolska

What does it look like?

- Although using the GUI on Frontier isn’t applicable, it’s still worth showing you what Blender looks like
  - Mainly just want to quickly highlight scripting mode

GUI



Cube

Cube

Transform

Location X	0 m
Y	0 m
Z	0 m
Rotation X	0°
Y	0°
Z	0°
Mode	XYZ Euler
Scale X	1.000
Y	1.000
Z	1.000

Delta Transform

Relations

Collections

Instancing

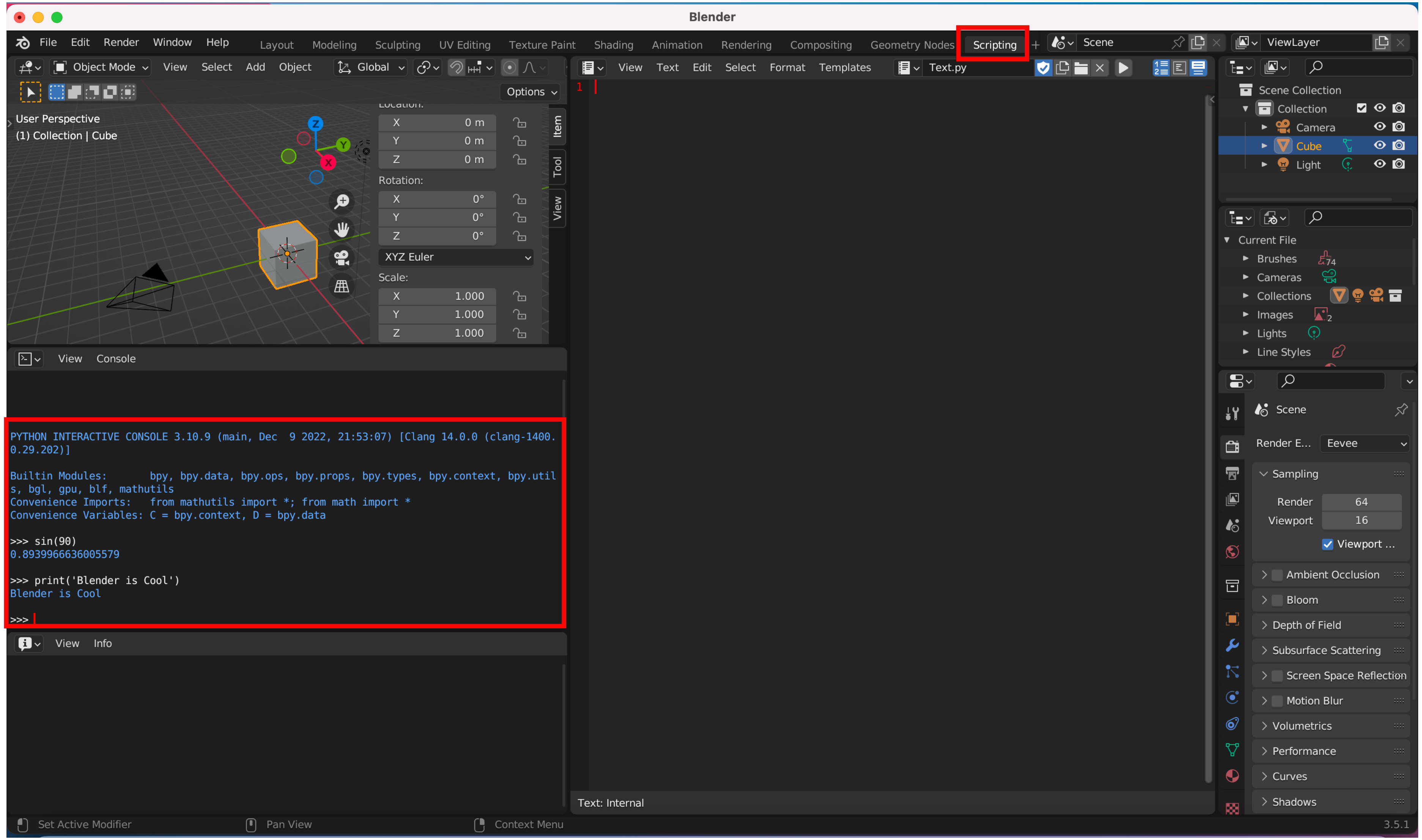
Motion Paths

Visibility

Viewport Display

Current frame: 1 · Start 1 · End 250

GUI



Location:

X	0 m
Y	0 m
Z	0 m

Rotation:

X	0°
Y	0°
Z	0°

XYZ Euler

Scale:

X	1.000
Y	1.000
Z	1.000

Scene

Render E... Eevee

Sampling

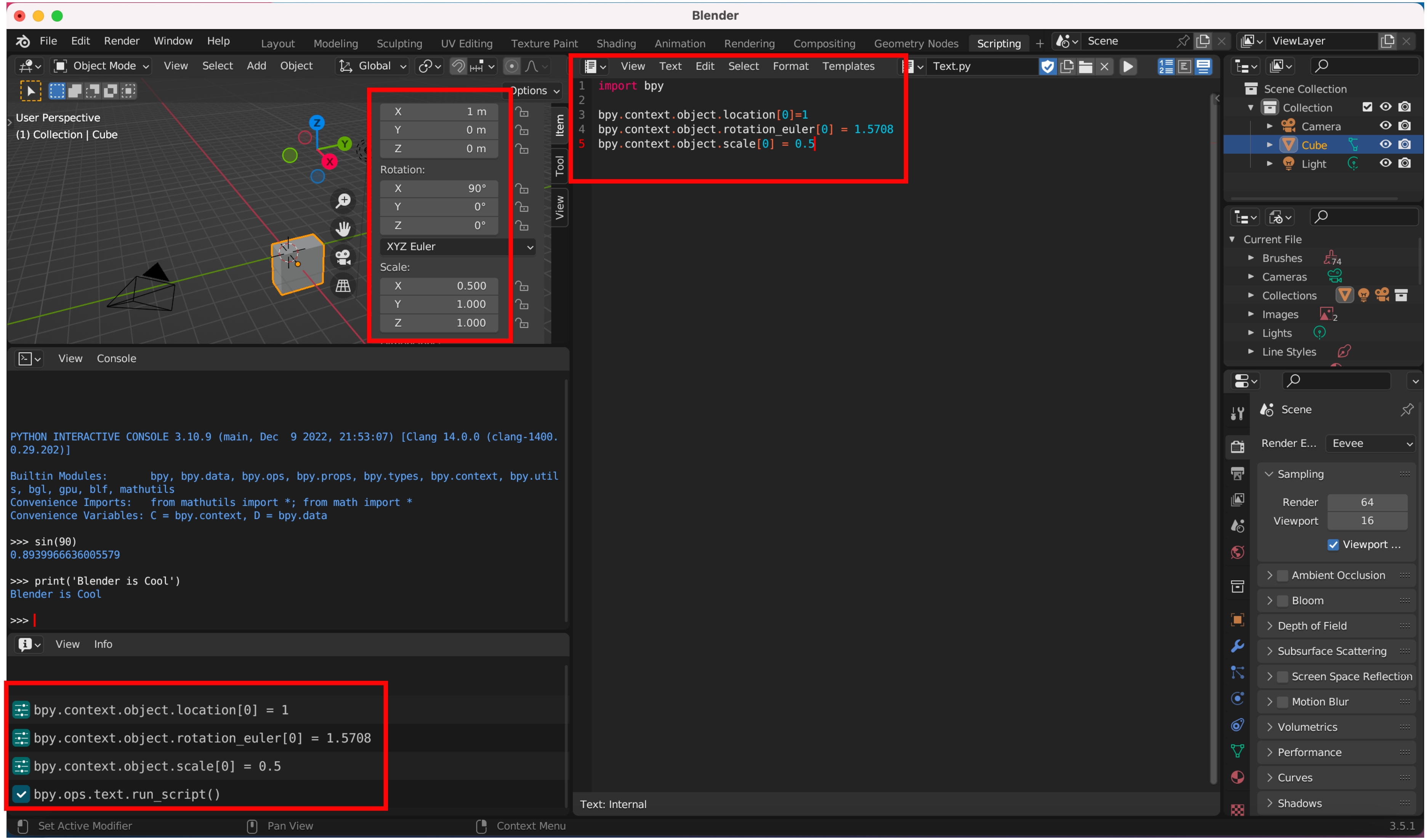
Render	64
Viewport	16

☑ Viewport ...

- □ Ambient Occlusion
- □ Bloom
- Depth of Field
- Subsurface Scattering
- □ Screen Space Reflection
- □ Motion Blur
- Volumetrics
- Performance
- Curves
- Shadows

```
PYTHON INTERACTIVE CONSOLE 3.10.9 (main, Dec  9 2022, 21:53:07) [Clang 14.0.0 (clang-1400.0.29.202)]

Builtin Modules:       bpy, bpy.data, bpy.ops, bpy.props, bpy.types, bpy.context, bpy.utils, bgl, gpu, blf, mathutils
Convenience Imports:   from mathutils import *; from math import *
Convenience Variables: C = bpy.context, D = bpy.data

>>> sin(90)
0.8939966636005579

>>> print('Blender is Cool')
Blender is Cool

>>> 
```

GUI



Location:

X	1 m
Y	0 m
Z	0 m

Rotation:

X	90°
Y	0°
Z	0°

XYZ Euler

Scale:

X	0.500
Y	1.000
Z	1.000

```python
import bpy

bpy.context.object.location[0]=1
bpy.context.object.rotation_euler[0] = 1.5708
bpy.context.object.scale[0] = 0.5
```

```
PYTHON INTERACTIVE CONSOLE 3.10.9 (main, Dec  9 2022, 21:53:07) [Clang 14.0.0 (clang-1400.
0.29.202)]

Builtin Modules:       bpy, bpy.data, bpy.ops, bpy.props, bpy.types, bpy.context, bpy.util
s, bgl, gpu, blf, mathutils
Convenience Imports:   from mathutils import *; from math import *
Convenience Variables: C = bpy.context, D = bpy.data

>>> sin(90)
0.8939966636005579

>>> print('Blender is Cool')
Blender is Cool

>>> 
```

```
bpy.context.object.location[0] = 1
bpy.context.object.rotation_euler[0] = 1.5708
bpy.context.object.scale[0] = 0.5
bpy.ops.text.run_script()
```

Render Engine: Eevee

Sampling

Render	64
Viewport	16

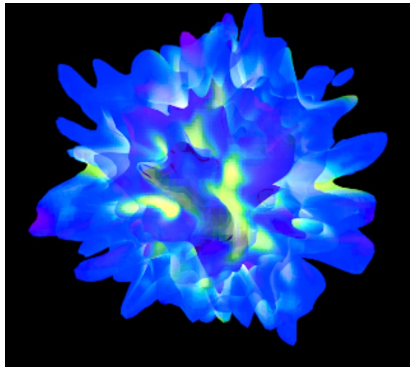
☑ Viewport ...

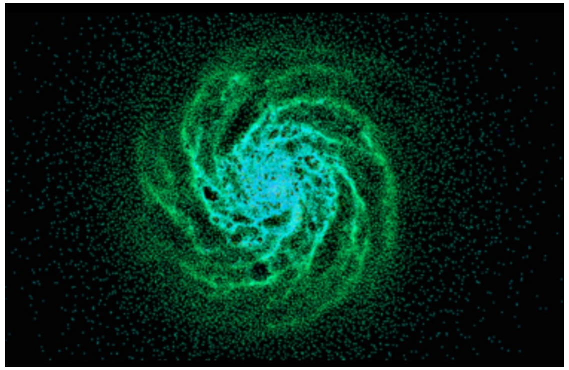
But…what about science?

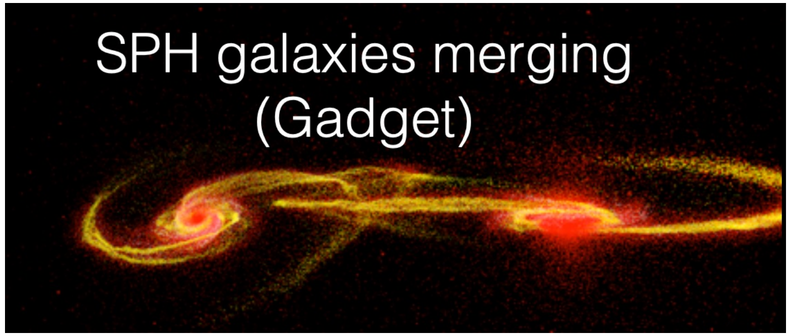
- Rendering, Scripting, Simulations all apply to science applications
  - **Rendering**: Allows you to produce high-quality images within a dynamic 3D environment
    - Matters for outreach, advertising your science, and publications
    - Arguably can produce higher quality visualizations than typical viz tools
    - More lighting and camera control ability than something like VisIt or ParaView
    - Can easily export renders to interactive 3D formats (either web-based or VR)
  - **Scripting**: The Python API works well with typical analysis and viz workflows
    - Allows importing external science packages (numpy, scipy, h5py, etc.)
    - Records and displays any actions you make in Blender as Python code
  - **Simulations**: Can run Python scripts *within* Blender itself to simulate things
    - Allows you to simulating objects in a scaled 3D environment
    - Jargon: Blender has its own “simulation” unit and solvers, but you can make your own

What can it handle? (Science-wise)

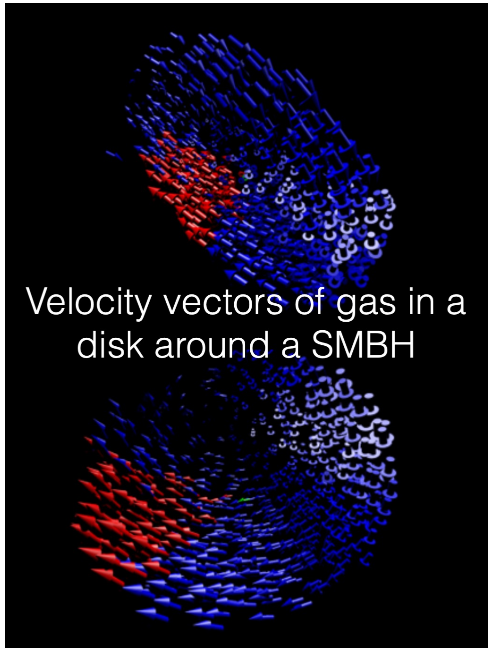
- Currently built to handle data that can be represented by 3D objects or by points in 3D space
  - Isosurfaces / contours
  - Point cloud / particle data
- The astronomy community is one of the only groups exploring this
  - AstroBlend (Naiman 2016)
    - Integrates with yt
    - R.I.P ~2021







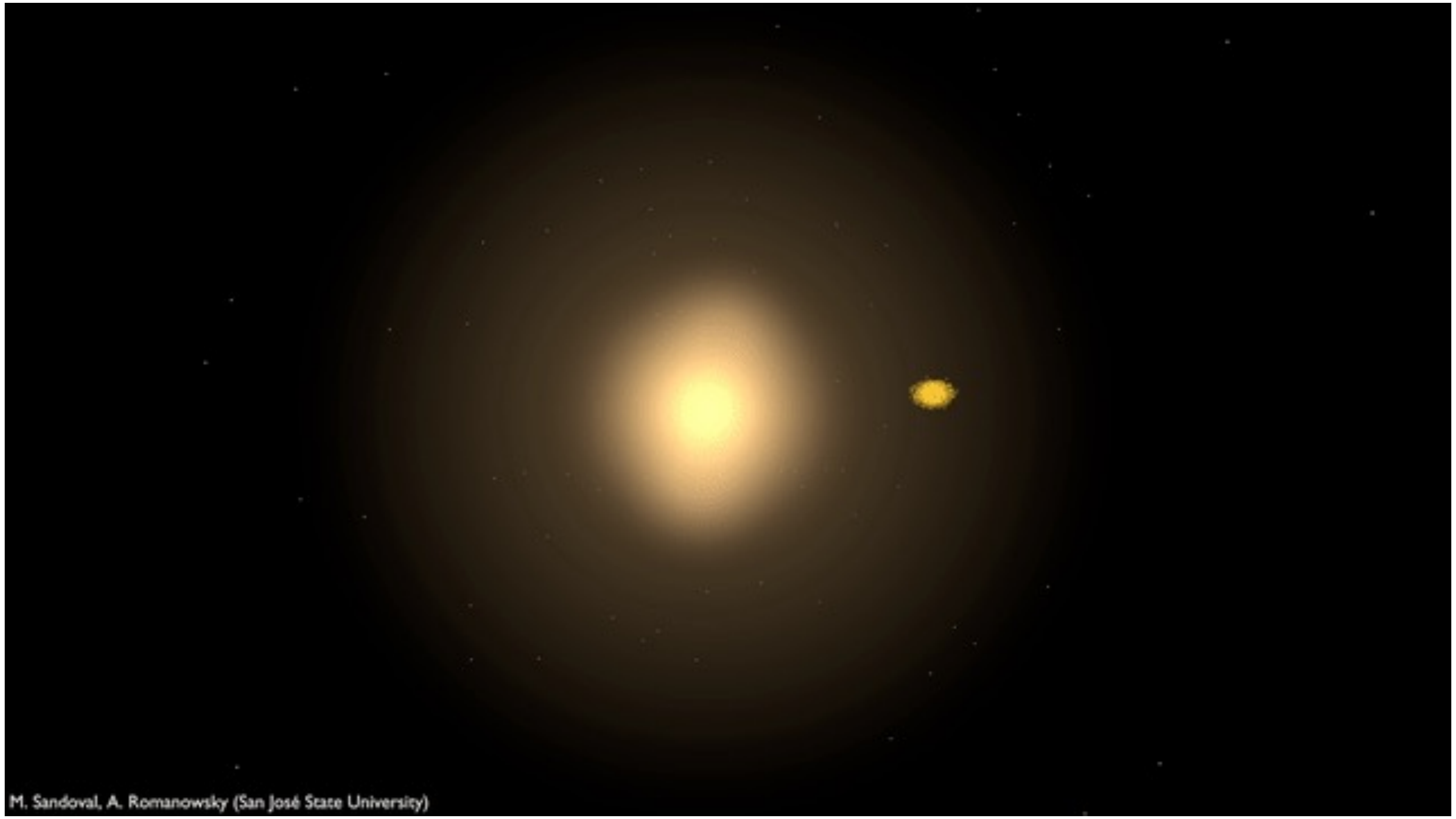
SPH galaxies merging (Gadget)

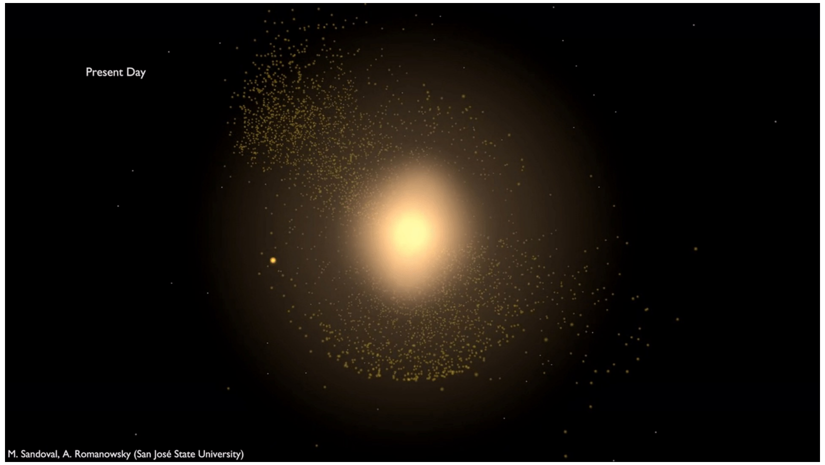


Velocity vectors of gas in a disk around a SMBH

Image credit: J.P. Naiman 2016, DOI: 10.1016/j.ascom.2016.02.002

Example Science Applications I

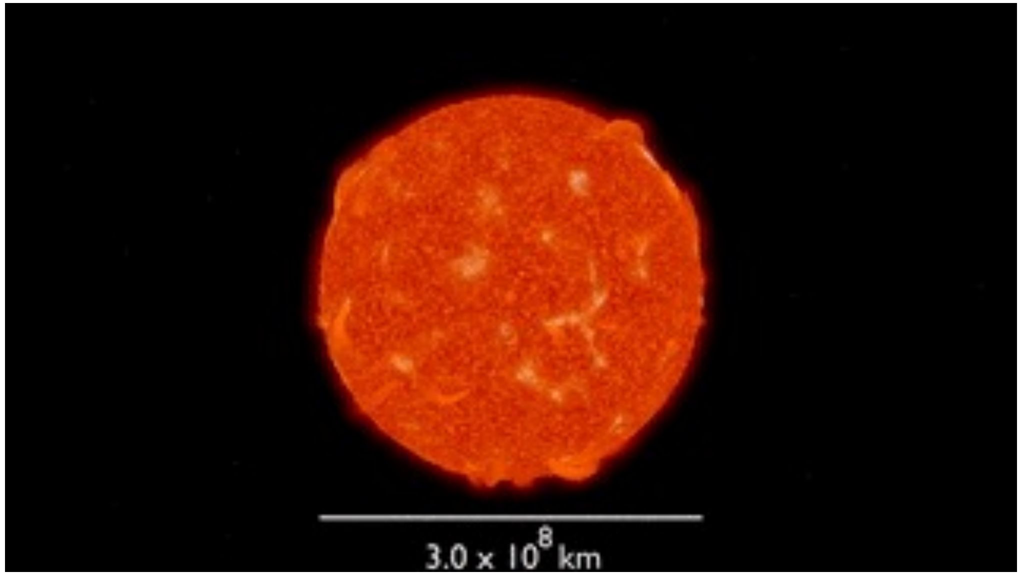




Galaxy evolution (tidal stripping simulation)

Credit: Michael A. Sandoval

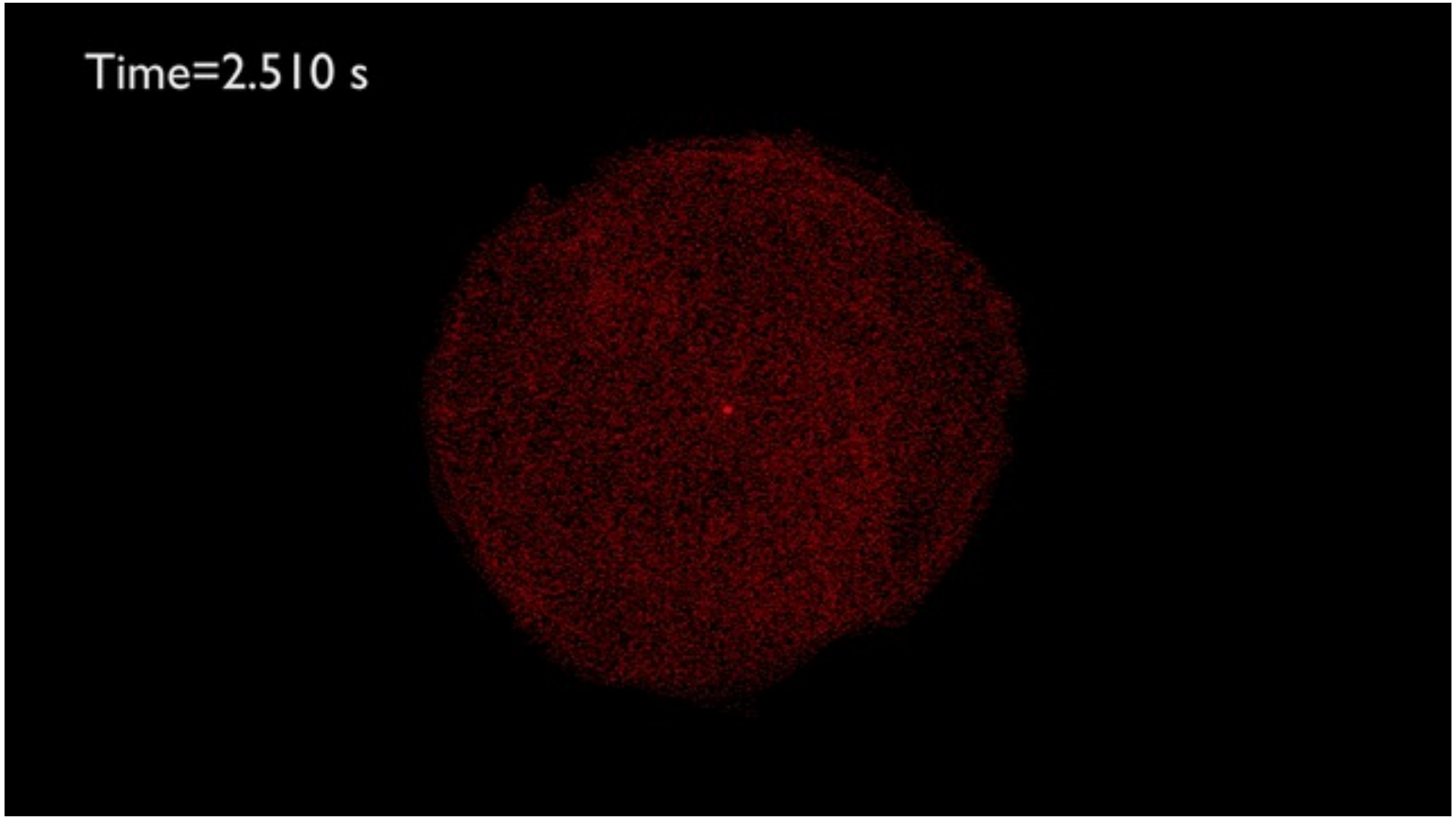
Example Science Applications II



Star Modeling (exploring a star during a CCSN)

Credit: Michael A. Sandoval

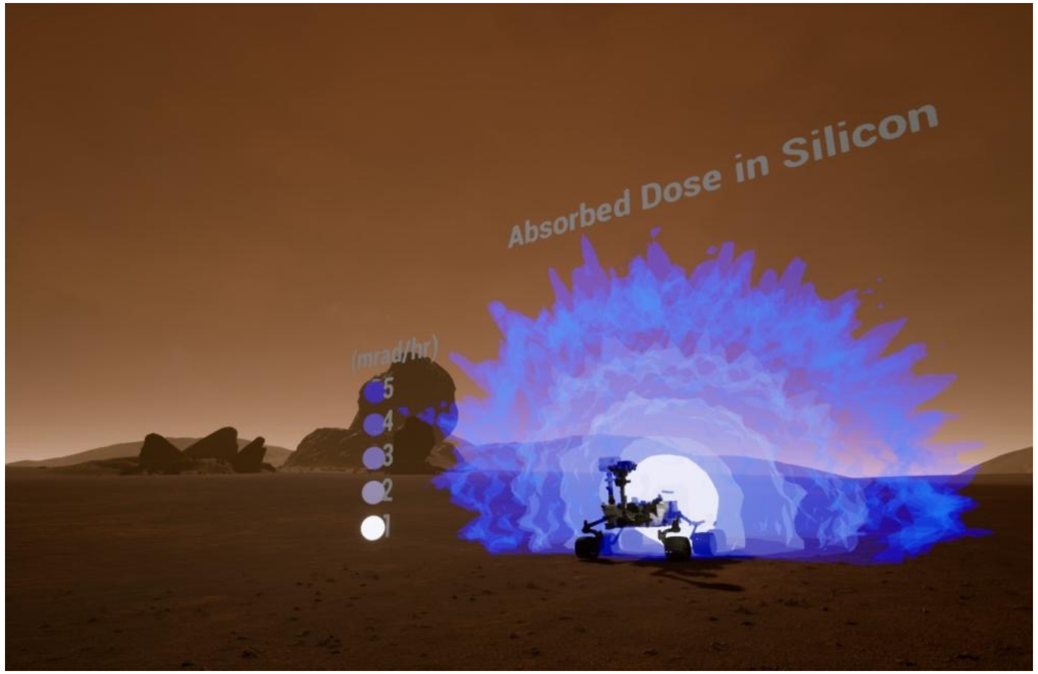
Example Science Applications III



Particle Data (CCSN simulation)

Example Science Applications IV

- Radiation transport on Mars
- Helps visualize transport geometries with 3D results
- Data pre-processed in ParaView before being imported into Blender+UnrealEngine4



Reference: M. Scott Greenwood and Michael B. R. Smith 2020, https://www.ans.org/pubs/transactions/article-48919/

Drawbacks

- Exciting language aside, it is far from easy
  - Scientific use is certainly a minority of the community, so it doesn't drive how the code is developed in general
    - Use in an HPC environment is even more of a minority
- Ultimately, more of an art than a science because of how it's built
- Most use cases would require pre-processing externally before porting to Blender
- Not the tool for someone that wants to quickly viz something
  - This is more of a tool to elevate your viz to the next level and/or for interactivity
- Steep learning curve in general (even more than ParaView!)
  - And an even steeper curve for using it for science

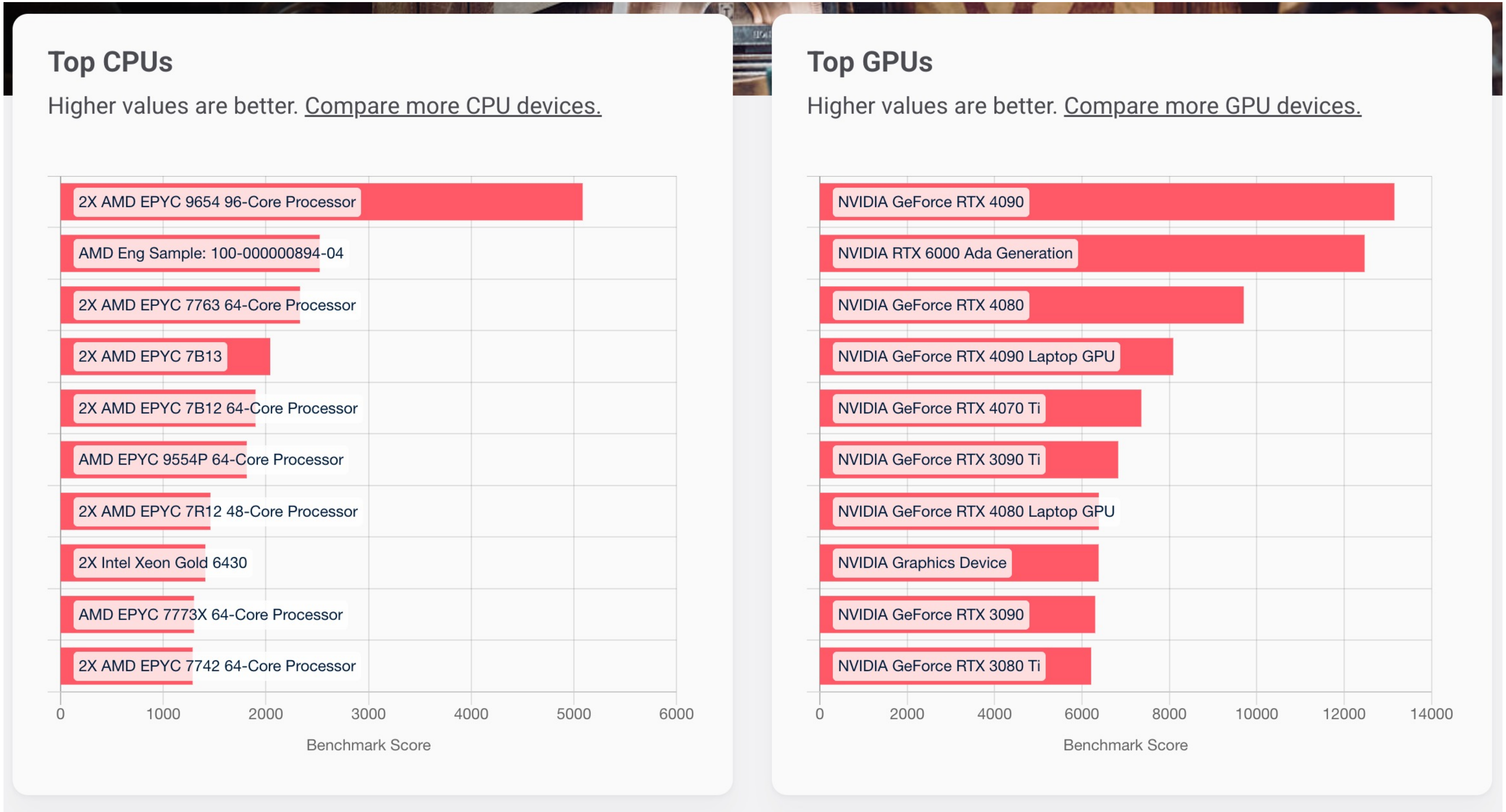
Looking to Frontier

- Not the viz cluster, so why Frontier?
  - It actually installs! (sorry Andes/Summit!)
  - GPU Power
    - Not just a single GPU per node (Andes), but multiple (this will matter)
- Can strongly utilize Blender’s “Cycles” render engine
  - High-quality ray tracing
  - GPU accelerated rendering (a big thing in Blender)
    - Supports HIP
  - Designed to provide physically based results
    - As opposed to rasterization algorithms (EEVEE render engine)
- Stumbled into the Blender Open Data Benchmark…

Open Data Benchmark

- Blender Open Data is a platform to collect the results of Blender performance tests across different hardware
- Provide benchmark “scenes” that calculate the rendering samples per minute over a given time period
  - Acts as your “benchmark score”
- Essentially the “Top 500”, but for Blender
- https://opendata.blender.org/

Open Data Benchmark



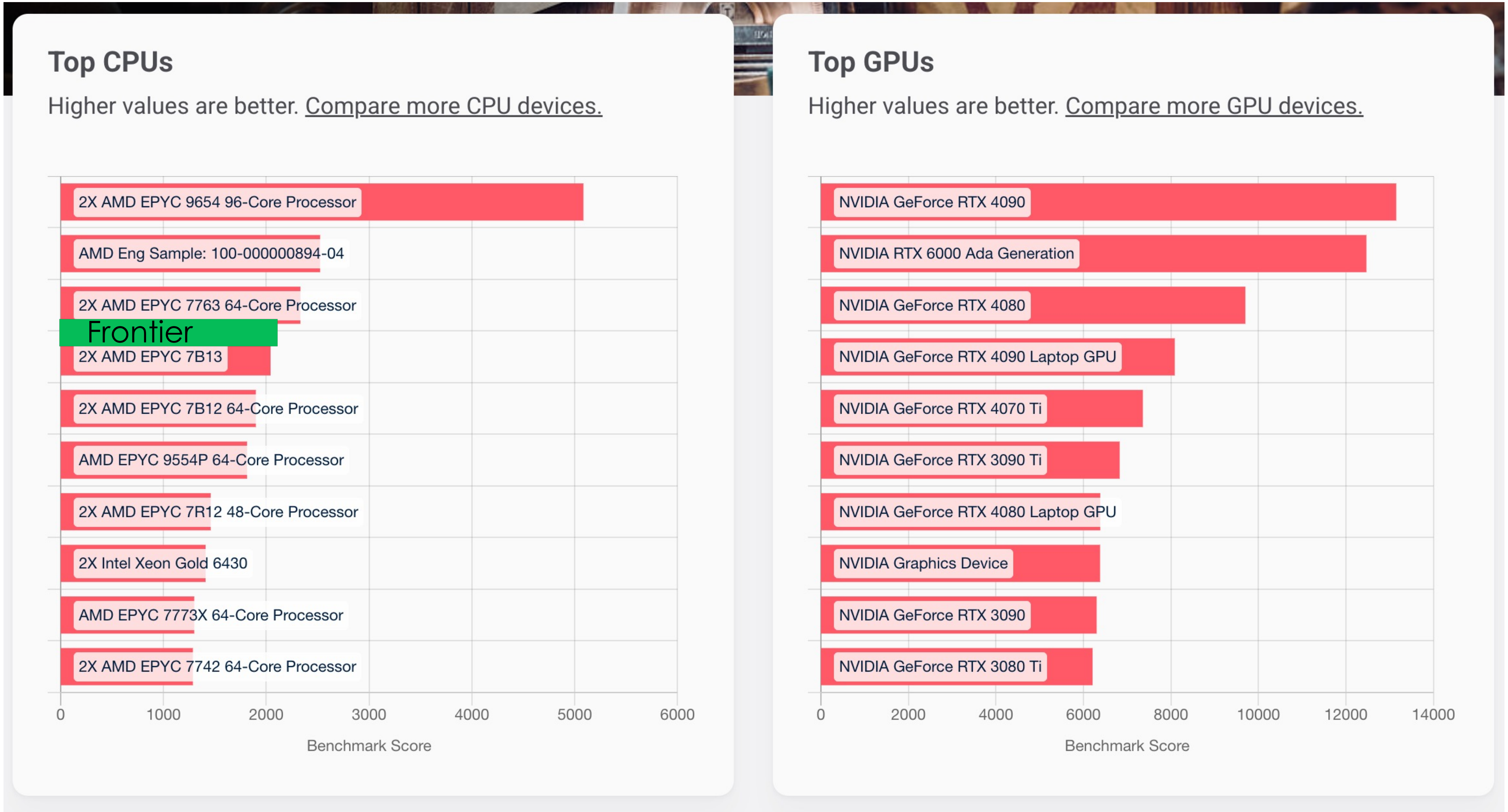
Top CPUs

Higher values are better. Compare more CPU devices.

Top GPUs

Higher values are better. Compare more GPU devices.

Open Data Benchmark



Top CPUs

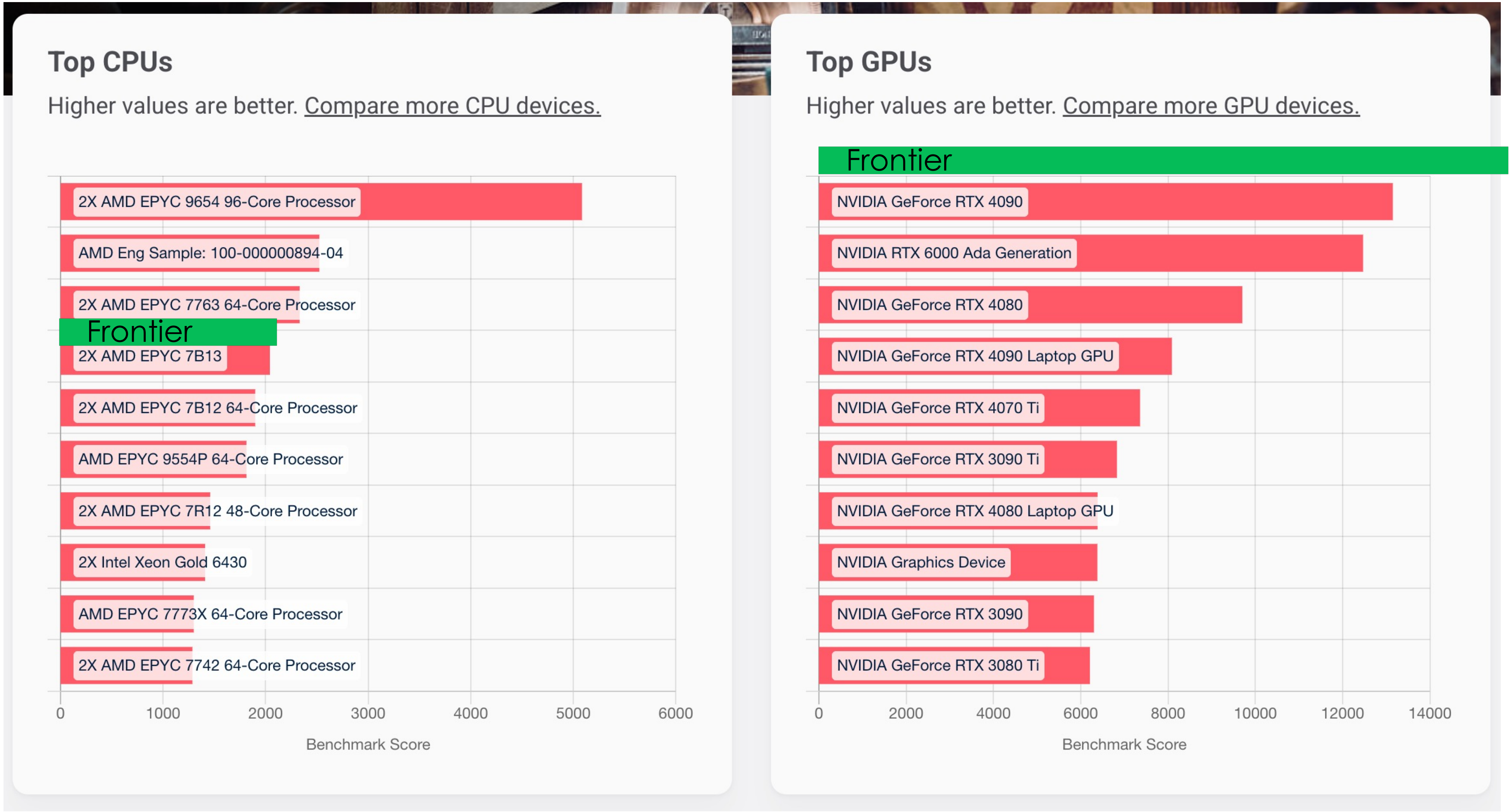
Higher values are better. Compare more CPU devices.

Top GPUs

Higher values are better. Compare more GPU devices.

Frontier

Open Data Benchmark



Top CPUs

Higher values are better. Compare more CPU devices.

Frontier
2X AMD EPYC 9654 96-Core Processor
AMD Eng Sample: 100-000000894-04
2X AMD EPYC 7763 64-Core Processor
2X AMD EPYC 7B13
2X AMD EPYC 7B12 64-Core Processor
AMD EPYC 9554P 64-Core Processor
2X AMD EPYC 7R12 48-Core Processor
2X Intel Xeon Gold 6430
AMD EPYC 7773X 64-Core Processor
2X AMD EPYC 7742 64-Core Processor
0
1000
2000
3000
4000
5000
6000
Benchmark Score

Top GPUs

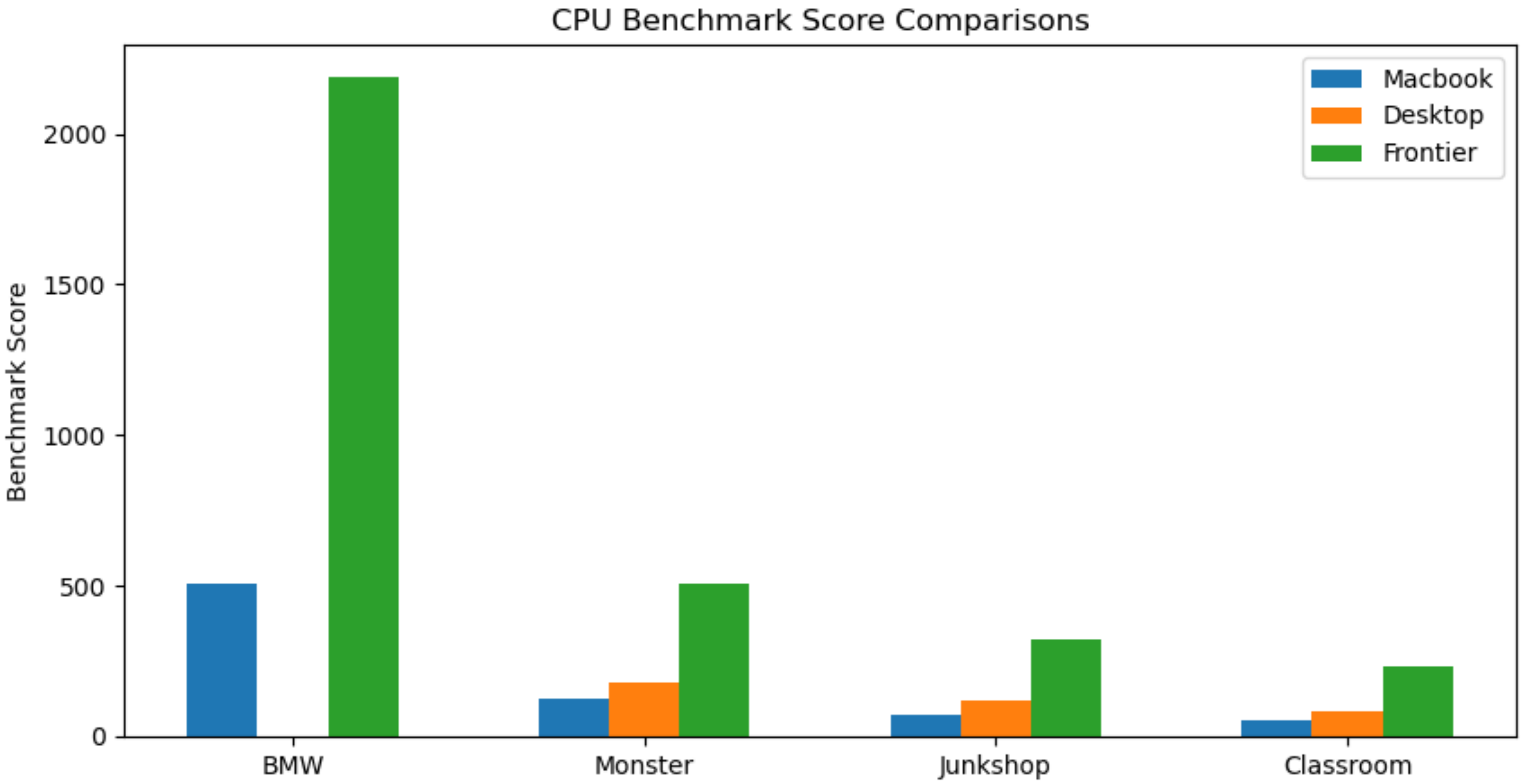
Higher values are better. Compare more GPU devices.

Frontier
NVIDIA GeForce RTX 4090
NVIDIA RTX 6000 Ada Generation
NVIDIA GeForce RTX 4080
NVIDIA GeForce RTX 4090 Laptop GPU
NVIDIA GeForce RTX 4070 Ti
NVIDIA GeForce RTX 3090 Ti
NVIDIA GeForce RTX 4080 Laptop GPU
NVIDIA Graphics Device
NVIDIA GeForce RTX 3090
NVIDIA GeForce RTX 3080 Ti
0
2000
4000
6000
8000
10000
12000
14000
Benchmark Score

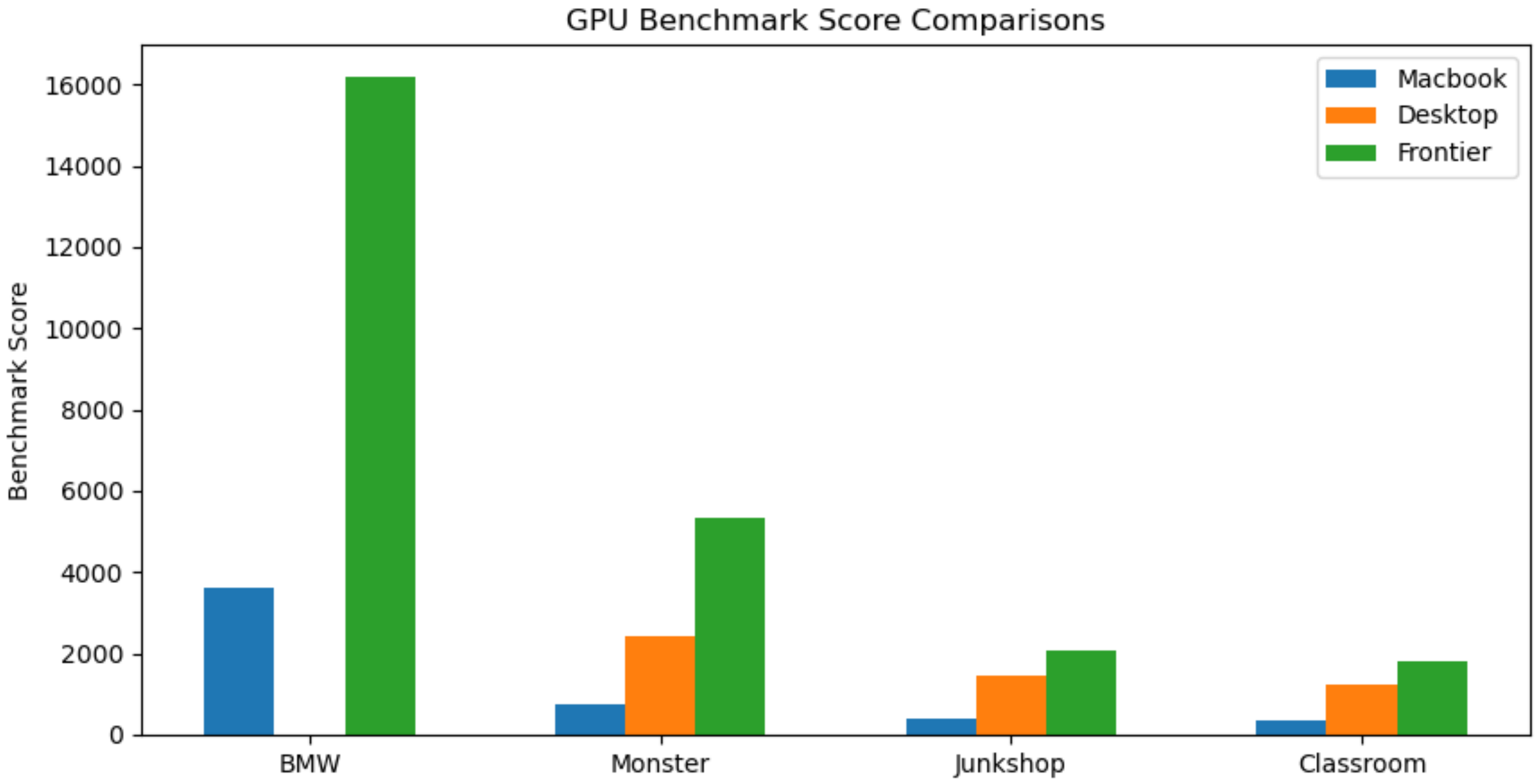
Comparing Across Devices

- Just for a sense of scale, let’s make some comparisons
- **My work laptop:** 2023 Apple M2 Max w/ 30 Core GPU
- **My rendering/gaming desktop:**
  - **CPU:** AMD Ryzen 9 5950X 16 core
  - **GPU:** NVIDIA GeForce RTX 3080
- **Frontier:**
  - **CPU:** AMD EPYC 7A53 64-Core Processor
  - **GPU:** AMD MI250X (x4) (Blender sees all 8 GCDs per node)
- Blender will use max threads on all devices by default

Benchmarks Across the CPUs



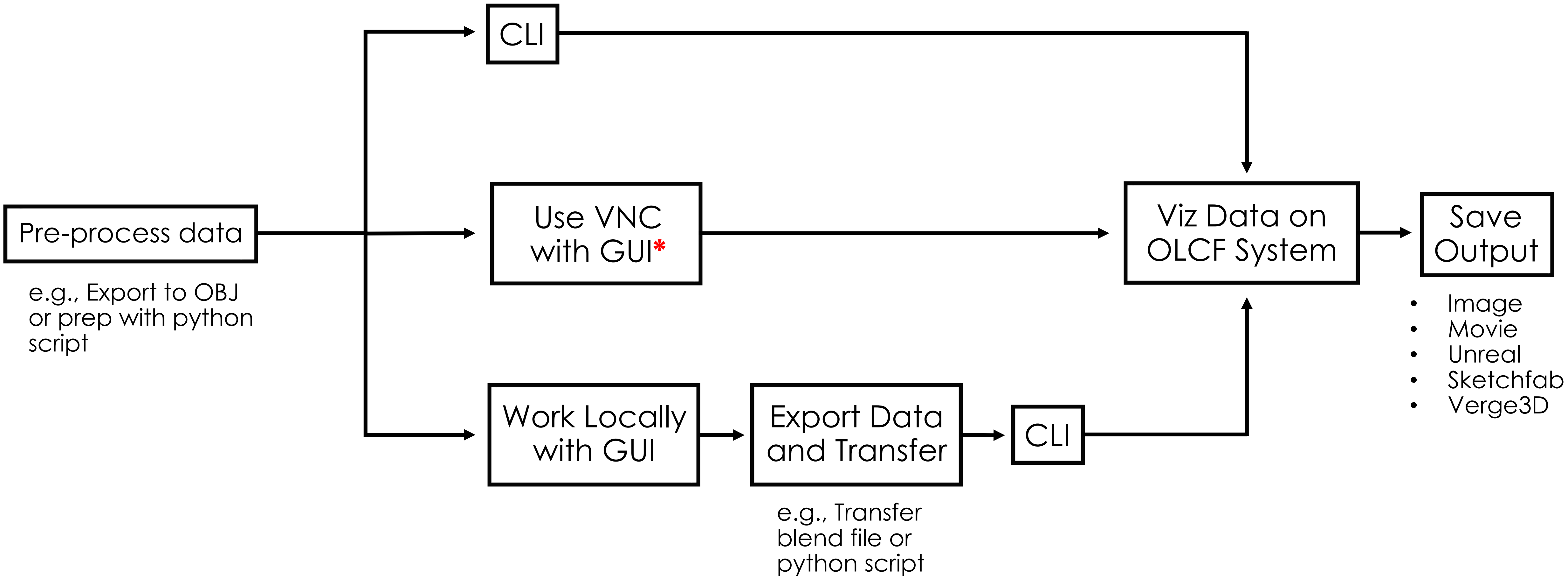
Benchmarks Across the GPUs



Limitations

- If that’s how it performs on 1 node, what about multi-node?
  - Only able to be run on 1 node, infrastructure / code isn’t built to run across multiple nodes
    - Good thing is 1 node is great
    - Most “normal” viz tools don’t necessarily scale when throwing multiple nodes anyway
- Limited to “headless mode” on Frontier (i.e., command-line usage only)
  - Make things hard to viz through trial and error, especially as a new user
    - Needs a working python script or blend file
  - Pushes you toward viz-ing locally first which isn’t always possible
    - Not everyone has this luxury
    - Can’t always reduce a dataset
- Exploring VNC usage currently, should help with above

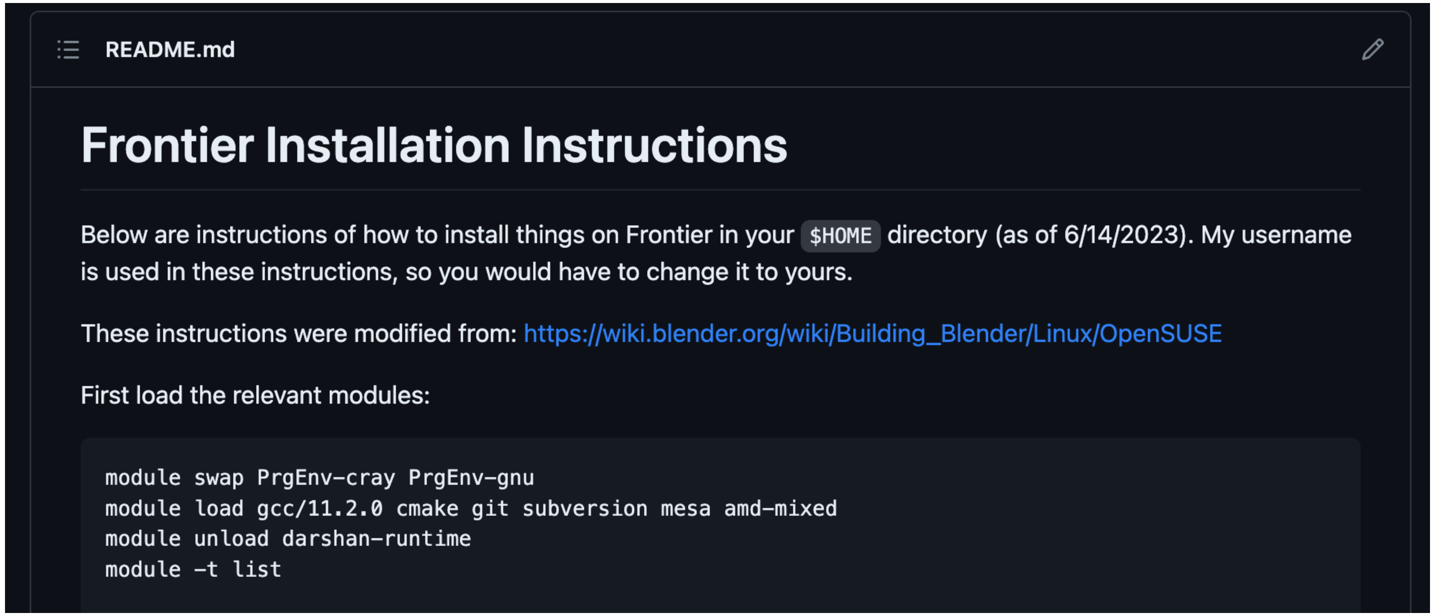
Potential Workflows



* Currently not an option on Frontier

Installing for CLI Usage on Frontier

- An involved process with some hacking necessary, but a slightly modified version of these OpenSUSE instructions: https://wiki.blender.org/wiki/Building_Blender/Linux/OpenSUSE
- Documented what I did on a fork here: https://github.com/michael-sandoval/blender/tree/3.5_frontier



README.md

Frontier Installation Instructions

Below are instructions of how to install things on Frontier in your `$HOME` directory (as of 6/14/2023). My username is used in these instructions, so you would have to change it to yours.

These instructions were modified from: https://wiki.blender.org/wiki/Building_Blender/Linux/OpenSUSE

First load the relevant modules:

```
module swap PrgEnv-cray PrgEnv-gnu
module load gcc/11.2.0 cmake git subversion mesa amd-mixed
module unload darshan-runtime
module -t list
```

Results

- So…what did the pictures look like?!
- Saved fun stuff for last!

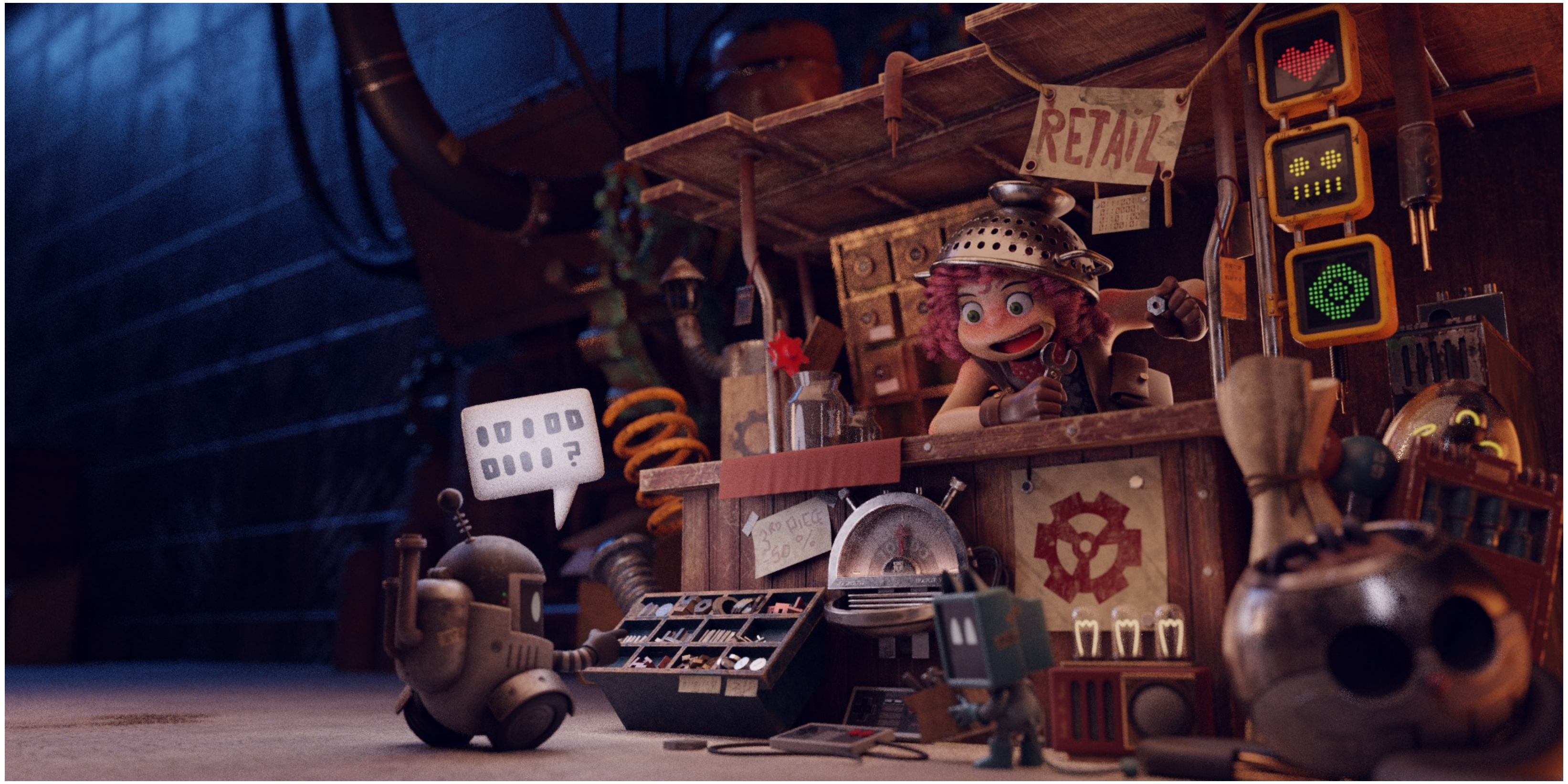
Benchmark Visualizations Produced on Frontier I

BMW benchmark

Benchmark Visualizations Produced on Frontier II

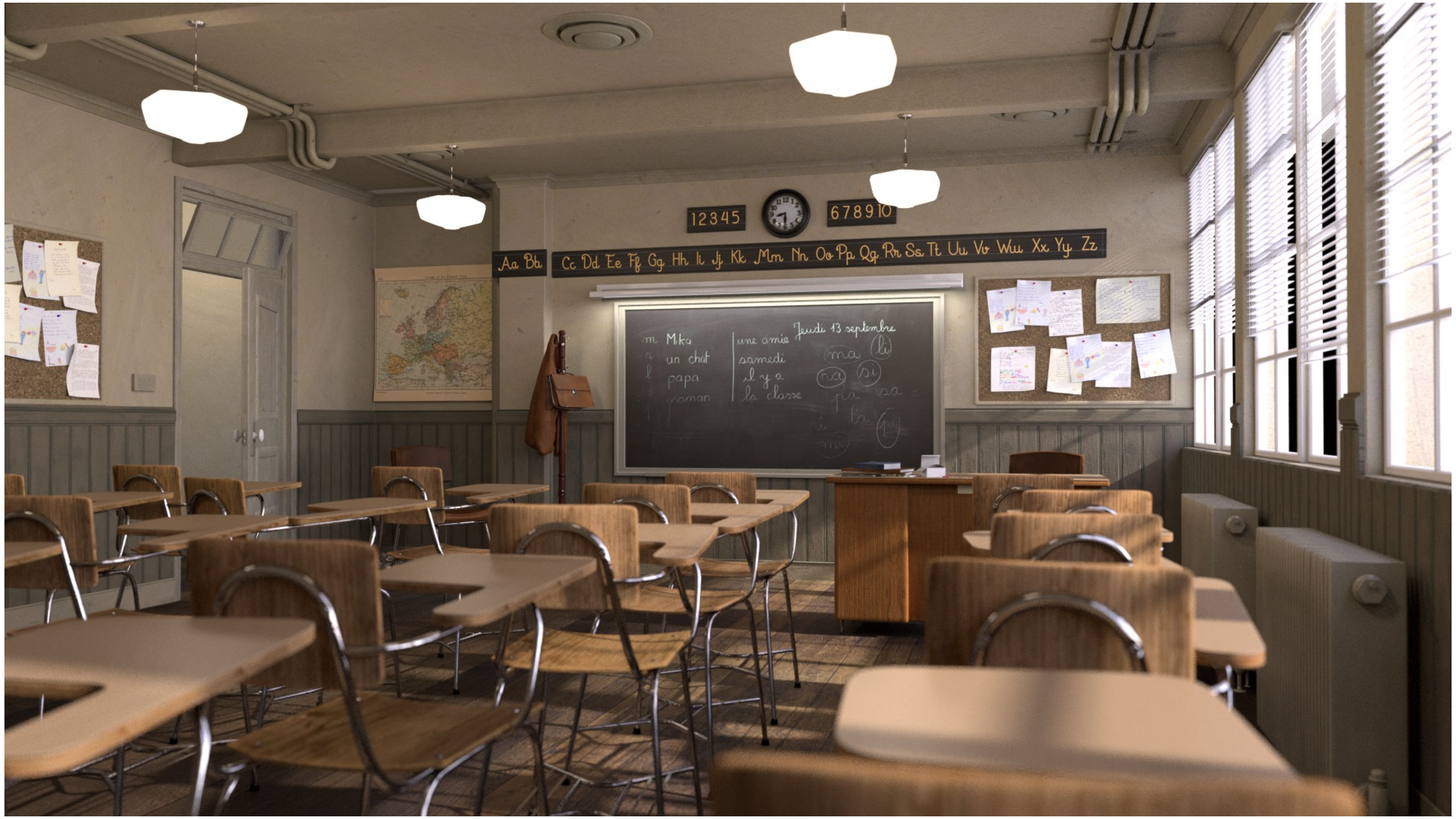
Monster benchmark

Benchmark Visualizations Produced on Frontier III



Junkshop benchmark

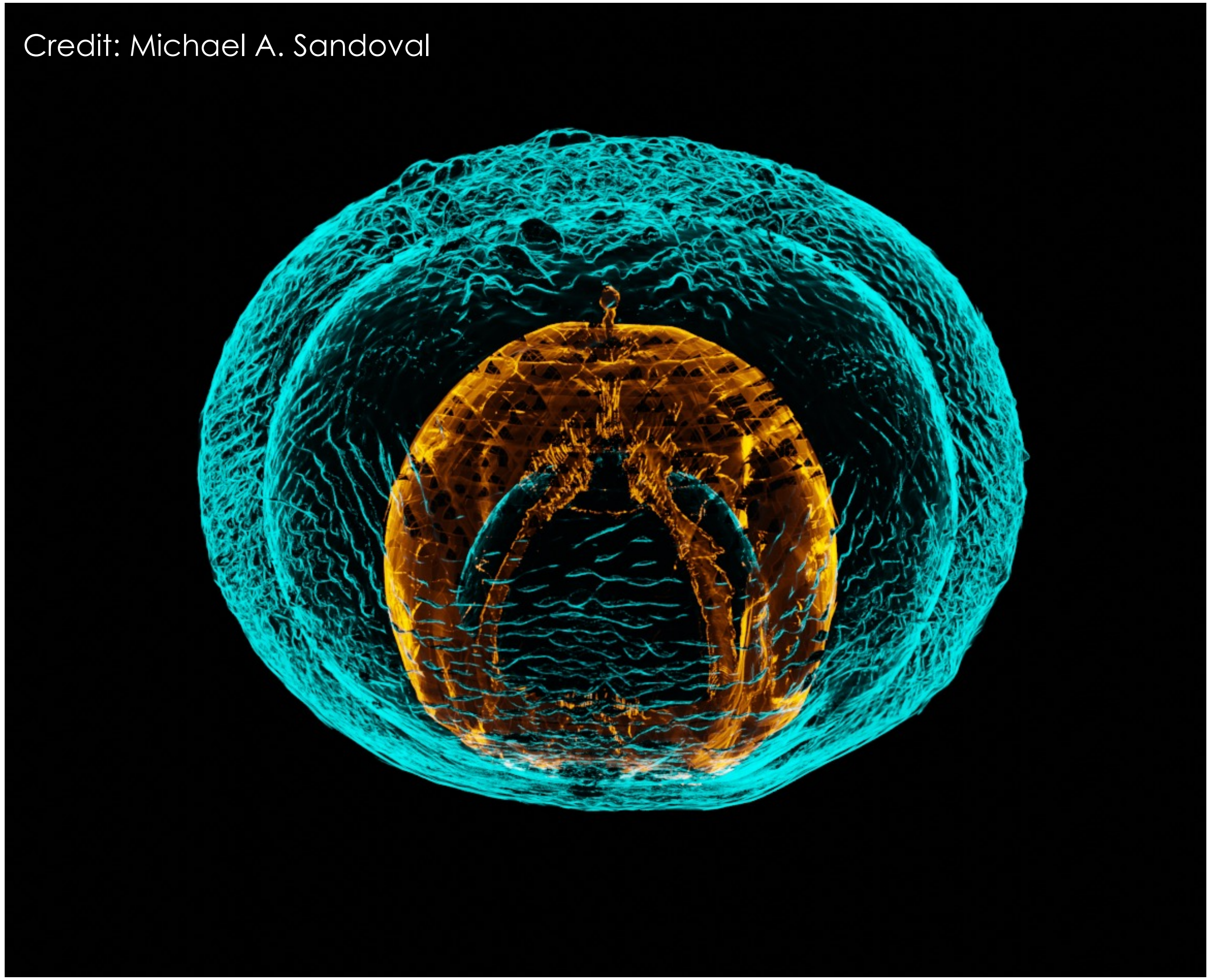
Benchmark Visualizations Produced on Frontier IV



Classroom benchmark

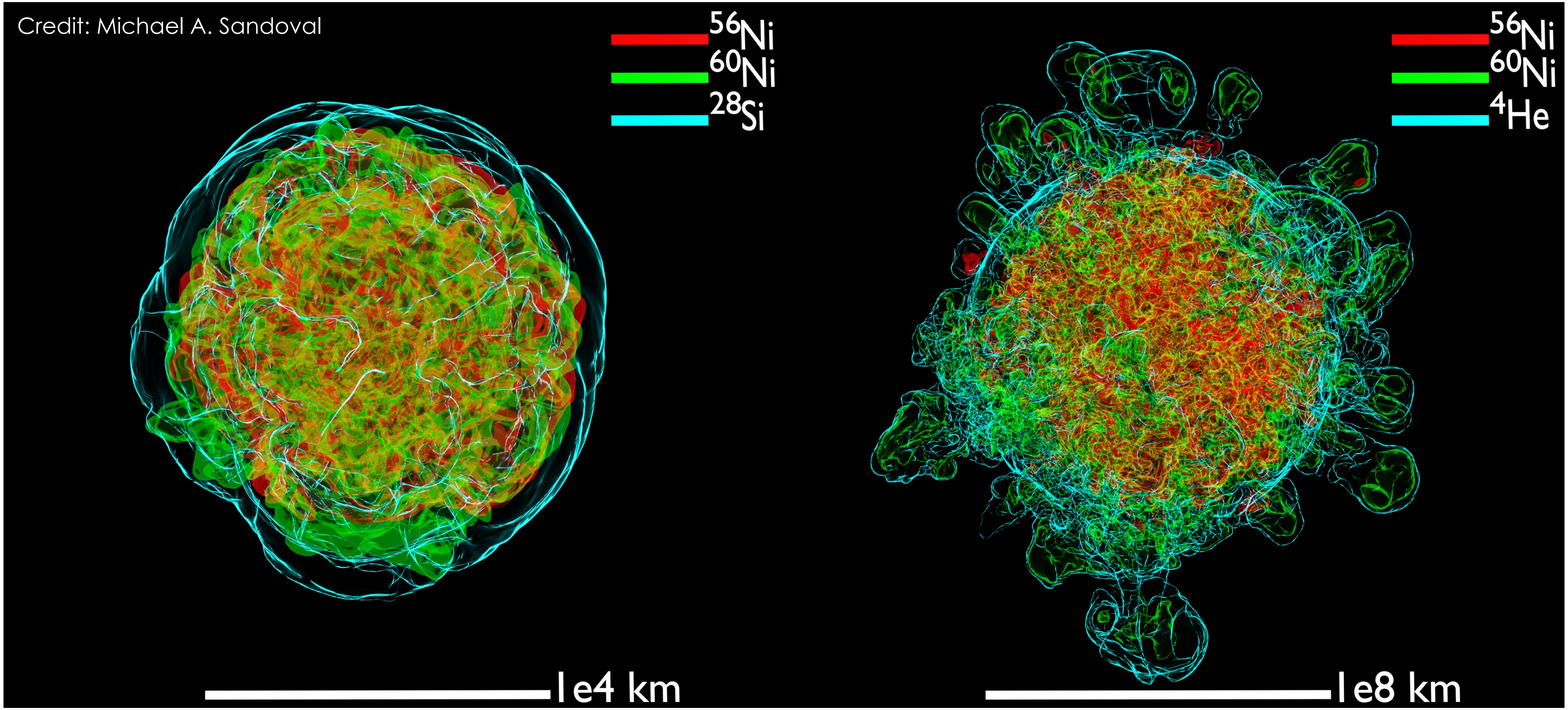
Yeah yeah, but what about the science viz?

Science Visualizations Produced on Frontier I


Credit: Michael A. Sandoval

Type IA Double Detonation

Science Visualizations Produced on Frontier II



Credit: Michael A. Sandoval

CCSN “before” / “after” shock breakout

Conclusions and Takeaways

- Frontier can make some pretty nice viz! (poor Andes ☹)
- Tuned desktop rigs have potential to compete on paper or “score”, but don’t have the luxury of RAM and filesystem
- The community is *slowly* growing – HPC extensions exist!
  - Essentially pioneering the use case for science+hpc and let’s make it known because the potential is there!
- Won’t be easy
  - Not the viz tool for you if you want a quick in/out (Mainly for “polish” right now)
  - Can only handle certain datasets
- My recommendation: play around with it, but start small!
  - Play with things locally and not necessarily for science purposes

Additional Resources

- Non-HPC related resources that helped me:
  - 3D Scientific Visualization Blender (Brian Kent):
    - Website: https://www.cv.nrao.edu/~bkent/blender/
    - Book: https://iopscience.iop.org/book/mono/978-1-6270-5612-0
  - www.astroblend.com