The Journal of Open Source Software

BlenDaViz, scientific visualization library for Blender in Python

Simon Candelaresi [1*¶] **and Christopher Berg Smiet** [2*]

1 High-Performance Scientific Computing, University of Augsburg, Universitätsstraße 12a, Augsburg, 86159, Germany **2** BergTop Fusion, Rochussenstraat 281B, 3023DE Rotterdam, Netherlands ¶ Corresponding author * These authors contributed equally.

DOI: 10.21105/joss.11181

Software

- Review
- Repository
- Archive

Editor: Johan Larsson

Reviewers:

- @Miko997
- @Chilipp

Submitted: 20 May 2026
Published: 03 November 2026

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

BlenDaViz (Candelaresi & Smiet, 2026) is a 3D plotting library for the visualization tool Blender (Blender Development Team, 2025), written in Python. It is able to visualize line plots, scatter plots, surfaces, iso-surfaces, glyphs (arrows) and streamlines from NumPy (Harris et al., 2020) arrays. The user can load BlenDaViz within a Blender Python console as any other library. Its design is close to other plotting libraries, especially Matplotlib (Hunter, 2007; Matplotlib Development Team, 2026), which means that every part of a plot is treated as an object that can be changed when updating the plot with new parameters or data. Compared to other 3D visualization software, we achieve higher image quality, thanks to Blender's ray tracing capabilities.

Statement of need

In scientific fields, like structural mechanics, computational fluid dynamics, astrophysics, or molecular dynamics, it is common to use 3D visualization tools like ParaView (Ahrens et al., 2005), VAPOR (Gruchalla et al., 2009), Mayavi (Ramachandran & Varoquaux, 2011) or VisIt (Childs et al., 2012). They can load a variety of data formats and visualize different aspects of the data using filters. However, they often lack the image quality needed to make the data easier to understand. Their limited shaders can create optical illusions where the depth of the geometry is not clear, or the colors are off due to unrealistic image generation.

Realistically rendered or ray-traced images, like those generated by 3D modelling tools like Blender, not only improve the image quality, but make it easier to understand the three-dimensional structure and colors. For instance, contextually implied scattered light can have a significant impact on the perceived color of objects, as exemplified by *The Dress* controversy (Chetverikov & Ivanchei, 2016). Lighting, shadow, depth and scattering can be used to generate photorealistic images that are easier to interpret.

State of the field

So far Blender has seen only limited use in 3D data visualization. BioBlender (Andrei et al., 2012) has been developed specifically for biological data and requires significant Blender knowledge. Similarly, suggestions to use Blender in astrophysics (Kent, 2015) require the generation of the geometry by hand. High-quality plots by the Oak Ridge National Laboratory show that to achieve high quality a collaboration between Blender experts and scientists is sometimes required.

Attempts at creating libraries were made with AstroBlend (Naiman, 2016) and BlenderSciViz,

but their applications are relatively limited and, as of November 2026, neither appears to be maintained: the last commit to BlenderSciViz dates from 24 December 2015 (commit 51535b3), while the AstroBlend documentation covers Blender versions up to 2.76b (2015) and its source repository is no longer available. Perhaps BVtkNodes and SciBlend (Marín et al., 2025) stand out, as they are able to load VTK data and apply filters to it.

With BlenDaViz we give the users all the rendering and ray tracing capabilities of Blender, combined with the ease of use and modularity we know from libraries like Matplotlib. It is general purpose, thanks to the implementation of some of the most used 3D plotting methods.

Software design

With BlenDaViz we keep the design very close to Matplotlib. The user supplies the input data in the form of NumPy arrays and the resulting plots are objects that can be freely modified. These objects contain members, such as the data and parameters (e.g. color), and methods, like the plotting function. That way, the user can modify their plots and replot them. An example is shown here where we generate a simple line plot, change the data of the resulting plot object and replot it:

```
import blendaviz as blt
import numpy as np

# Generate the data.
y = np.linspace(0, 6*np.pi, 400)
x = 2*np.cos(y)
z = 2*np.sin(y)

# Generate the line plot.
pl = blt.plot(x, y, z, radius=0.5)

# Change the data.
pl.x = 2*pl.x

# Replot the data.
pl.plot()
```

Each of our plotting functions follows a similar pipeline. As the plotting function is called, the input is checked for consistency. Then any existing geometry or materials for this plot are deleted. Depending on the plotting function, different kinds of meshes are generated (lines, surfaces, arrows, etc.). To finalize the plot object a material is added. Geometry and material are then part of the plotting object and can, in principle, be accessed. After the geometry generation we add lights and a camera, if not already present.

An example is shown in Figure 1 where we plot a two-dimensional array as a surface with the values used as z-coordinates and as colors. To highlight its geometry we added a plane with a checkerboard texture in the background and readjusted the camera. Such manual adjustments can be done directly in Blender and can greatly improve the plot quality.

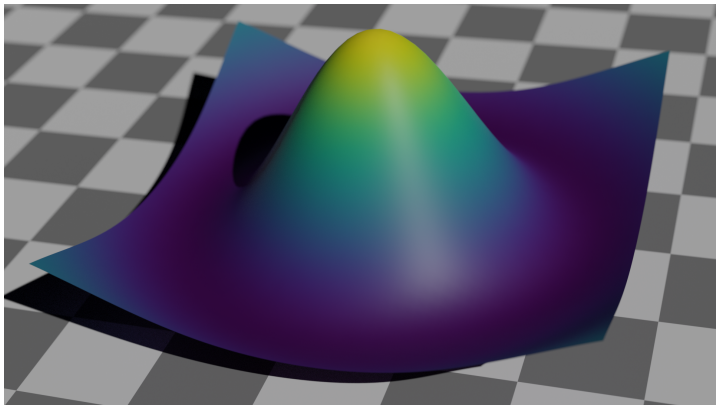


Figure 1: Mesh plot of a Gaussian using a colormap.

Handling of time

We make handling time-dependent data easy in BlenDaViz. All that is required is that the data arrays contain one more dimension. For instance, if we want to plot the position of n particles at p different times, our arrays x, y and z would have the shape $[n, p]$. Parameters that take the same shape, like colors, follow the same structure.

BlenDaViz recognizes the data and input parameters as time dependent. It then adds the plotting method of this plot object to the `bpy.app.handlers.frame_change_pre` handler of Blender. That way, any change in the time via, e.g. the slider, results in an update of all present plots.

Research impact statement

In Candelaresi et al. (2021) BlenDaViz was used to visualize the vortex field of a turbulent finite-difference simulation using the Pencil Code (The Pencil Code Collaboration et al., 2021). Later Israeli et al. (2023) and Candelaresi & Beck (2023) used it to visualize magnetic field configurations obtained from simulations. The former also uses BlenDaViz to present particle trajectories.

AI usage disclosure

We used Claude Sonnet 4.6 for some code debugging and to help write the majority of tests. All changes made by Claude were verified by hand.

Acknowledgements

This work was supported by a grant from the Simons Foundation (1013657, JL). This work has been carried out within the framework of the EUROfusion Consortium, via the Euratom Research and Training Programme (Grant Agreement No 101052200 — EUROfusion) and funded by the Swiss State Secretariat for Education, Research and Innovation (SERI). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union, the European Commission, or SERI. Neither the European Union nor the European Commission nor SERI can be held responsible for them. This project has benefited from funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) through the research unit FOR 5409 "Structure-Preserving Numerical Methods for Bulk- and Interface Coupling of Heterogeneous Models (SNuBIC)" (project number 463312734).

References

Ahrens, J., Geveci, B., & Law, C. (2005). ParaView: An end-user tool for large-data visualization. In C. D. Hansen & C. R. Johnson (Eds.), *The visualization handbook* (pp. 717–731). Elsevier. https://doi.org/10.1016/B978-012387582-2/50038-1

Andrei, R. M., Callieri, M., Zini, M. F., Loni, T., Maraziti, G., Pan, M. C., & Zoppè, M. (2012). Intuitive representation of surface properties of biomolecules using BioBlender. *BMC Bioinformatics*, *13*, S16. https://doi.org/10.1186/1471-2105-13-S4-S16

Blender Development Team. (2025, July 15). *Blender*. https://www.blender.org/

Candelaresi, S., & Beck, C. (2023). Twisted magnetic knots and links. *Physics of Plasmas*, *30*(8), 082102. https://doi.org/10.1063/5.0148156

Candelaresi, S., Hornig, G., Podger, B., & Pontin, D. I. (2021). Topological constraints in the reconnection of vortex braids. *Physics of Fluids*, *33*(5), 056101. https://doi.org/10.1063/5.0047033

Candelaresi, S., & Smiet, C. B. (2026). *BlenDaViz, scientific visualization library for Blender in Python* (Version v1.0.0). Zenodo. https://doi.org/10.5281/zenodo.23810340

Chetverikov, A., & Ivanchei, I. (2016). Seeing "the Dress" in the right light: Perceived colors and inferred light sources. *Perception*, *45*(8), 910–930. https://doi.org/10.1177/0301006616643664

Childs, H., Brugger, E., Whitlock, B., Meredith, J., Ahern, S., Pugmire, D., Biagas, K., Miller, M., Harrison, C., Weber, G. H., Krishnan, H., Fogal, T., Sanderson, A., Garth, C., Bethel, E. W., Camp, D., Rübel, O., Durant, M., Favre, J. M., & Navrátil, P. (2012). VisIt: An end-user tool for visualizing and analyzing very large data. In E. W. Bethel, H. Childs, & C. Hansen (Eds.), *High performance visualization–enabling extreme-scale scientific insight* (pp. 357–372). https://doi.org/10.1201/b12985

Gruchalla, K., Rast, M., Bradley, E., Clyne, J., & Mininni, P. (2009). Visualization-driven structural and statistical analysis of turbulent flows. In N. M. Adams, C. Robardet, A. Siebes, & J.-F. Boulicaut (Eds.), *Advances in intelligent data analysis VIII* (pp. 321–332). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-03915-7_28

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Israeli, B. Y., Smiet, C. B., Simeni Simeni, M., & Diallo, A. (2023). EUV debris mitigation using magnetic nulls. *Applied Physics Letters*, *123*(4), 043507. https://doi.org/10.1063/5.0152083

Kent, B. R. (2015). *3D scientific visualization with Blender*. Morgan & Claypool Publishers. https://doi.org/10.1088/978-1-6270-5612-0

Marín, J., Baptiste, T. M. G., Rodero, C., Williams, S. E., Niederer, S. A., & García-Fernández, I. (2025). SciBlend: Advanced data visualization workflows within Blender. *Computers & Graphics*, *130*, 104264. https://doi.org/10.1016/j.cag.2025.104264

Matplotlib Development Team. (2026). *matplotlib/matplotlib: REL: v3.11.2* (Version v3.11.2). Zenodo. https://doi.org/10.5281/zenodo.22700776

Naiman, J. P. (2016). AstroBlend: An astrophysical visualization package for Blender.

Astronomy and Computing, *15*, 50–60. https://doi.org/10.1016/j.ascom.2016.02.002

Ramachandran, P., & Varoquaux, G. (2011). Mayavi: 3D visualization of scientific data. *Computing in Science & Engineering*, *13*(2), 40–51. https://doi.org/10.1109/mcse.2011.35

The Pencil Code Collaboration, Brandenburg, A., Johansen, A., Bourdin, P. A., Dobler, W., Lyra, W., Rheinhardt, M., Bingert, S., Haugen, N. E. L., Mee, A., Gent, F., Babkovskaia, N., Yang, C.-C., Heinemann, T., Dintrans, B., Mitra, D., Candelaresi, S., Warnecke, J., Käpylä, P. J., … Qian, C. (2021). The Pencil Code, a modular MPI code for partial differential equations and particles: multipurpose and multiuser-maintained. *Journal of Open Source Software*, *6*(58), 2807. https://doi.org/10.21105/joss.02807