

# <sup>1</sup> openpv/simshady: A Javascript Package for Photovoltaic Yield Estimation Based on 3D Meshes

<sup>3</sup> **Florian Kotthoff**  <sup>1,2</sup>¶, **Konrad Heidler**  <sup>1</sup>, **Martin Großhauser**  <sup>1</sup>, **Mino Estrella**  <sup>3</sup>, and **Korbinian Pöppel**  <sup>1</sup>

<sup>5</sup> 1 OpenPV GbR, Munich, Germany 2 OFFIS Institute for Information Technology, Escherweg 2, 26121 Oldenburg, Germany 3 Chair of Renewable and Sustainable Energy Systems, Technical University of Munich, Germany ¶ Corresponding author

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Editor: [Open Journals](#) ↗

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)). To meet global climate targets, solar photovoltaic (PV) capacity must expand significantly. Tripling renewable energy capacity by 2030 is essential to limit global warming to 1.5°C ([International Energy Agency, 2023](#)). The expansion of PV plays a crucial role, and PV systems offer an additional benefit: small-scale house-mounted PV systems enable public participation and legitimize the energy transition.

## <sup>16</sup> Statement of need

<sup>22</sup> For calculating the yield of PV systems, various factors are important, including the location of the planned installation, local climate, surrounding objects such as houses or trees, and terrain. To provide accurate estimates of expected yields, simulation tools are essential in both research and practical PV system planning.

<sup>26</sup> For these reasons, a variety of software tools for simulating photovoltaic systems already exist ([Holmgren et al., 2018](#); [Jakica, 2018](#)). One widely used software is the Python package pvlib ([Anderson et al., 2023](#)), which offers a range of functionalities. However, the rather niche topic of shading simulation with 3D objects is not included in this package. Another Python-based software that enables irradiance modeling in two dimensions is pvfactors ([Anoma et al., 2017](#); [Pvfactors, 2022](#)).

<sup>32</sup> Web-based tools for solar panel simulations, such as PVGIS, PVWatts, and RETScreen, provide an accessible means for non-technical individuals to estimate energy yields based on geographic location and building geometry ([Psomopoulos et al., 2015](#)). However, these tools lack the capability to perform shading simulations using 3D geometries.

## <sup>36</sup> Software design

<sup>37</sup> openpv/simshady simulates the yield of photovoltaic (PV) systems by considering weather/climate data and shading from local 3D geometry. It is build around three core principles: (1) To

39 run the core simulation code efficiently on the GPU, (2) to expose two different interfaces for  
 40 both browser-based applications and server-based simulation, and (3) to integrate standardized  
 41 data models, namely from Three.js and from the Wavefront OBJ file format.  
 42 In simshady, a 3D scene is built that represents the environment, comprising primary objects  
 43 for simulation (e.g., PV panels or target buildings) and surrounding objects that may cast  
 44 shadows (e.g., neighboring buildings, trees). Weather and climate data are integrated using  
 45 Global Horizontal Irradiance (GHI) and Direct Normal Irradiance (DNI) datasets, which are  
 46 reconstructed to include directional irradiance information using the HEALPix framework  
 47 ([Górski et al., 2005](#); [Zonca et al., 2019](#)).  
 48 The simulation utilizes the Möller-Trumbore intersection algorithm ([Möller & Trumbore, 1997](#))  
 49 to determine if any shading objects obstruct the view between a sky pixel and the main  
 50 simulation geometry. For each triangle in the simulation geometry, a shading mask is generated,  
 51 indicating whether an object blocks the line of sight from the sky pixel to the triangle. The  
 52 shading mask values range from 0 to 1, where 0 indicates that an object shades the triangle,  
 53 1 signifies that there is no obstruction and the line of sight is perpendicular to the triangle,  
 54 and values between 0 and 1 represent cases where there is no obstruction but the angle of  
 55 incidence is not perpendicular. The aggregated radiance values from all sky dome pixels  
 56 are then multiplied by the corresponding shading mask values and summed to calculate the  
 57 total energy received by each triangle. This computation is fully parallelizable and has been  
 58 implemented using WebGL, allowing for GPU acceleration.

### 59 **Simshady in the browser**

60 Simshady reuses the data model and objects of Three.js, a common package for 3D web  
 61 applications. The package processes Three.js meshes and adds the simulated PV yield as a  
 62 color to the mesh, as shown in [Figure 1](#). Additionally, each triangle of the simulated buildings  
 63 has its solar yield assigned as an attribute for further processing.



**Figure 1:** A simulated building with its solar yield, where dark purple represents low yields and light yellow represents high yields. The simulated shading from neighboring buildings is visualized through darker colors. Screenshot taken from ([Openpv.de](#))

## 64    Simshady in the terminal

65    The simshady CLI is a wrapper around the core WebGL-based simulation engine that enables  
66    batch processing of photovoltaic-yield analyses on a server. It first parses a small set of required  
67    arguments (the simulation geometry and the irradiance data). The supplied geometry files,  
68    either JSON objects or Wavefront OBJ files, are handed to a headless Chromium instance  
69    launched via Puppeteer ([Google Chrome Team, 2025](#)).

70    Inside the browser context the full simshady package is injected, the scene is reconstructed, and  
71    the GPU-accelerated Möller-Trumbore ray-tracing routine is executed. When the calculation  
72    finishes, the CLI extracts the mesh data from the browser, writes binary artefacts (positions.bin,  
73    colors.bin, intensities.bin), a colour-coded OBJ file, a top-down snapshot, and the simulation  
74    results into the user-specified output directory.

## 75    Research impact statement

- 76    ▪ First open source, free software of its kind  
77    ▪ Multi usage approach: Server and Client

## 78    Conclusion

79    The openpv/simshady package serves two primary purposes: it provides a solution for scientific  
80    calculations of PV yield, while also facilitating science communication through interactive and  
81    user-friendly simulations. This eliminates the need for specialized software or programming  
82    knowledge, making it accessible to a broader range of users. Furthermore, by implementing the  
83    main algorithm in WebGL, the package achieves higher performance than a pure Javascript  
84    implementation, and it offers a JavaScript wrapper around PV simulation in WebGL. This  
85    is particularly beneficial because WebGL is a language that is not widely known among  
86    scientists, and thus can be challenging for them to implement their own code, making the  
87    openpv/simshady package a valuable tool for simplifying this process.

## 88    Credit Authorship Statement

- 89    FK: Conceptualization, Software, Funding acquisition, Writing – original draft  
90    MG: Conceptualization, Software, Funding acquisition, Writing – review & editing  
91    KH: Conceptualization, Software, Funding acquisition, Writing – review & editing  
92    ME: Software, Writing – review & editing  
93    KP: Conceptualization, Software, Funding acquisition, Writing – review & editing

## 94    AI usage disclosure

95    After designing the software architecture and writing core functionalities, both open-weight and  
96    properitarian LLM-based chatbots where used to implement or debug some of the methods in  
97    simshady. Generative agentic AI tools that can write and edit code autonomously were not  
98    used. The open-weight models gpt-oss:120b and deepseek-r1:70b were used for reviewing the  
99    written text of this paper.

## 100 Acknowledgements

101 The development of this software was funded by the German Federal Ministry of Research,  
102 Technology and Space within the “Software Sprint - Support for Open Source Developers”  
103 program by Prototype Fund.

## 104 References

- 105 Anderson, K. S., Hansen, C. W., Holmgren, W. F., Jensen, A. R., Mikofski, M. A., & Driesse,  
106 A. (2023). Pvlib python: 2023 project update. *Journal of Open Source Software*, 8(92),  
107 5994. <https://doi.org/10.21105/joss.05994>
- 108 Anoma, M. A., Jacob, D., Bourne, B. C., Scholl, J. A., Riley, D. M., & Hansen, C. W.  
109 (2017). View factor model and validation for bifacial PV and diffuse shade on single-  
110 axis trackers. *2017 IEEE 44th Photovoltaic Specialist Conference (PVSC)*, 1549–1554.  
111 <https://doi.org/10.1109/PVSC.2017.8366704>
- 112 Google Chrome Team. (2025). *Puppeteer: Headless chrome node.js API* (Version v24.31.0).  
113 Google. <https://github.com/puppeteer/puppeteer>
- 114 Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., &  
115 Bartelmann, M. (2005). HEALPix: A framework for high-resolution discretization and  
116 fast analysis of data distributed on the sphere. *The Astrophysical Journal*, 622(2), 759.  
117 <https://doi.org/10.1086/427976>
- 118 Holmgren, W. F., Hansen, C. W., Stein, J. S., & Mikofski, M. A. (2018). Review of open  
119 source tools for PV modeling. *2018 IEEE 7th World Conference on Photovoltaic Energy  
120 Conversion (WCPEC)(a Joint Conference of 45th IEEE PVSC, 28th PVSEC & 34th EU  
121 PVSEC)*, 2557–2560. <https://doi.org/10.1109/PVSC.2018.8548231>
- 122 International Energy Agency. (2023). *Tripling renewable power capacity by 2030  
123 is vital to keep the 1.5°C goal within reach*. [https://www.iea.org/commentaries/  
tripling-renewable-power-capacity-by-2030-is-vital-to-keep-the-150c-goal-within-reach](https://www.iea.org/commentaries/<br/>124 tripling-renewable-power-capacity-by-2030-is-vital-to-keep-the-150c-goal-within-reach).
- 125 Jakica, N. (2018). State-of-the-art review of solar design tools and methods for assessing  
126 daylighting and solar potential for building-integrated photovoltaics. *Renewable and  
127 Sustainable Energy Reviews*, 81, 1296–1328. <https://doi.org/10.1016/j.rser.2017.05.080>
- 128 Möller, T., & Trumbore, B. (1997). Fast, minimum storage ray-triangle intersection. *Journal  
129 of Graphics Tools*, 2(1), 21–28. <https://doi.org/10.1080/10867651.1997.10487468>
- 130 *Openpv.de*. [www.openpv.de](http://www.openpv.de).
- 131 Psomopoulos, C. S., Ioannidis, G. C., Kaminaris, S. D., Mardikis, K. D., & Katsikas, N.  
132 G. (2015). A comparative evaluation of photovoltaic electricity production assessment  
133 software (PVGIS, PVWatts and RETScreen). *Environmental Processes*, 2, 175–189.  
134 <https://doi.org/10.1007/s40710-015-0092-4>
- 135 Pvfactors: Open-source view-factor model for diffuse shading and bifacial PV modeling (Version  
136 1.5.2). (2022). <https://github.com/SunPower/pvfactors>
- 137 Zonca, A., Singer, L., Lenz, D., Reinecke, M., Rosset, C., Hivon, E., & Gorski, K. (2019).  
138 Healpy: Equal area pixelization and spherical harmonics transforms for data on the sphere  
139 in python. *Journal of Open Source Software*, 4(35), 1298. <https://doi.org/10.21105/joss.01298>