The Journal of Open Source Software

GroLink: A general application programming interface for the plant-modeling platform GroIMP

Tim Oberländer [1*], **Gaëtan Heidsieck** [1*], **Thomas Hay** [1], **and Winfried Kurth**[1]

1 Georg-August-Universität Göttingen, Lower Saxony, Germany ***** These authors contributed equally.

DOI: 10.21105/joss.08343

Software

- Review
- Repository
- Archive

Editor: Rachel Wegener

Reviewers:

- @egonw
- @gaurav

Submitted: 20 February 2025
Published: 18 November 2025

Summary

The plant modeling platform GroIMP (Kniemeyer, 2008), is broadly used by students and researchers to develop and run functional structural plant models and analyze their behavior and environmental interactions. Its integrated design, which keeps modeling, simulation, and analysis in the same graphical user interface, makes installation, learning, and modeling easy. However, as data size and computing power increased, so did the complexity of models, driving the need for integration into external tools and remote execution. The GroLink project (Oberländer, 2023; Oberländer et al., 2025) created a general HTTP API with additional client libraries to answer this need. Using this approach, it is possible to run and interact with GroIMP models from other software or devices using an extendable set of generalized commands.

Statement of need

The GroIMP ecosystem has expanded a lot since the first release 20 years ago. In some cases, the focus shifted towards creating bigger pipelines, or, due to size and complexity, more performant executions were needed. From our experience with model developers, modelers, educators and their projects, we derived four main directions in automatization and integration to focus on.

3D model provider

Creating 3D models as (intermediate) results of simulations is a key feature of GroIMP. Yet direct export to another software is currently only possible after the simulation ended (Lanwert, 2008) or via non-interactive file-based approaches. Enabling other software to **execute simulation steps** of the model and to **export the scene at any time**, would support live integration for external interactive visualization (as seen on other platforms (Fabrika, 2021)) as well as forwarding structural information (Zhu et al., 2021). That kind of visualization would allow modelers and educators to present interactive results without going through the whole modeling environment.

Custom model interfaces

There are cases where simpler or more model-specific interfaces are needed. Yet creating a new interface for a model is currently limited to either an implementation inside the platform (Smoleňová et al., 2012, 2013) or an external interface that restarts a non-interactive simulation in the background after each input (Spehle, 2023). To enable real-time external model interfaces, a **software independent interactive connection** would be required. Therefore, model developers

could create custom user interfaces, providing a simpler and clearer environment for modelers and external users.

GroIMP as a service

The idea of providing a file containing a model and set of instructions to GroIMP from another software was already introduced by the work of Qinqin Long (Long et al., 2018) on the co-simulation with the plant modeling platform OpenAlea (Pradal et al., 2008). Yet this work focused more on the theory of co-simulation and model transfer, therefore the integration follows a quite strict procedure. A more general integration as a service would need an approach that can work **independently from specific file formats or scenarios**. This would allow model developers to use GroIMP components in models defined on other platforms.

Orchestration of multiple simulations

To automatically execute multiple models, modelers commonly run GroIMP headless, i.e. without a graphical interface nor possibility of interactions, for each simulation using external scripts (Streit et al., 2016). However, the headless mode has three main limitations. First, it requires additional manual configuration. Second, every model execution deploys, configures, and starts the platform. Finally, it is non-interactive. Thus it is more complex to adapt the orchestration based on intermediate feedback (having a "user in the loop").

An HTTP server (Kniemeyer, 2008; Lanwert, 2008) was added to GroIMP to resolve the second limitation. However, it still does not provide means of interacting with the model and requires not only additional configuration in the model but also technical expertise.

Therefore, an improved solution needs to allow modelers to easily execute several simulations on a running instance **in parallel while being able to interact** with them **without additional implementation** on the simulation. The main audience for this would be modelers, running parameterized experiments such as sensitivity analysis.

Related projects

When considering other software projects with a similar focus on developing functional structural plant models, it becomes clear that different software architectures present different challenges. Most other approaches, such as OpenAlea (Pradal et al., 2008) or Virtual plant lab (Morales et al., 2025), are created as libraries rather than as standalone platforms. With this design, part of the library is considered the "API," enabling integration into larger projects, provided the programming languages are compatible. In order to link these projects to incompatible platforms or remote systems and additional layer would be required.

GroLink

The GroLink project (Oberländer et al., 2025), as presented in Tim Oberländer's Master's thesis (Oberländer, 2023), adds a non-blocking HTTP server to GroIMP, which provides a stateless API. The design uses the fact that each project (workbench) runs independently on its thread and that any interaction is done through predefined commands. For each request, the API server pushes the referenced command to the specified workbench for execution.

This design was chosen based on the discussion above:

- Software independence is given by the usage of HTTP.
- Id-based addressing of workbenches allows for parallel interactive simulation.
- By using commands, quite general access to GroIMP's functionalities is provided.
- As the interaction mimics the GUI, changes to the model are rarely necessary.

The API server is an independent implementation that should not be confused with the embedded HTTP server. The predefined commands are divided into two categories: application commands for managing workbenches and workbench commands for interacting with loaded projects. With a few exceptions, input parameters are provided in the HTTP query, and the results are JSON-formatted text. For workbench commands, the result includes the current content of the project's logs and console. The server's openAPI specification can be found in the repository and accessed through the API.

We created additional libraries in Python (GroPy) and R (GroR) to simplify interacting with the server and individual projects.

Installation and usage

GroLink can be installed directly from within GroIMP through the plugin manager. To start the software as an API server the additional command-line argument "`-a api`" is required, as explained in the repository (docs/starting_the_api.md).

To send requests to the running API server a web browser or a client library can be used, tutorials and install instructions can be found in the repository (docs/Getting_started_with_HTTP.md).

Examples

We created three different client projects aiming in the directions we considered above. All three projects and more examples are available on Gitlab: https://gitlab.com/groimp-api-examples.

1. To demonstrate automation and management of external resources, we created a simple light simulation pipeline. This pipeline consists of importing a 3D model into GroIMP; running a light simulation; coloring model parts based on the absorbed radiation; and exporting the resulting 3D model. It was implemented as a GroPy-based script that can be executed from commandline.
2. To highlight simulation-level interactions between GroIMP and other commonly used software, we implemented two interactive RShiny web applications. The first is a generalized interactive viewer for any GroIMP model. The second is an interface to manipulate the environment of a specific plant model and observe its reactions. These examples were implemented using GroR and also shows the possibility of using custom UI for GroIMP.
3. To test live client side manipulations of the 3D scene, we linked three different forest models to a Godot game (Godot, n.d.). In this game a user can walk through the forest, cut or plant trees, and trigger growth steps which are then simulated in GroIMP and pushed back to the game. This example also shows that the API can be used by any kind of http client.

Discussion

The presented project shows potential to improve GroIMP usage in all four discussed areas, as demonstrated by the examples. The first three improvements are significant because such usage was nearly impossible before. Regarding orchestration, even though it was previously possible, two improvements can be noted: first, interaction with an automatically called model is now possible; and second, the client libraries and API structure ease the usage for modelers with little coding experience.

References

Fabrika, M. (2021). Interactive procedural forest in game engine environment as background for forest modelling. *DVFFA – Sektion Ertragskunde, Beiträge zur Jahrestagung 2021*, 86–95. https://sektionertragskunde.nw-fva.de/2021/06_Fabrika.pdf

Godot. (n.d.). *Godot Engine - Free and open source 2D and 3D game engine — godotengine.org*. https://godotengine.org/

Kniemeyer, O. (2008). *Design and implementation of a graph grammar based language for functional-structural plant modelling* [Doctoralthesis, BTU Cottbus]. https://nbn-resolving.org/urn:nbn:de:kobv:co1-opus-5937

Lanwert, D. (2008). *Funktions-/Strukturorientierte Pflanzenmodellierung in E-Learning-Szenarien*. [Doctoralthesis, University of Göttingen]. https://hdl.handle.net/11858/00-1735-0000-0006-B10B-A

Long, Q., Kurth, W., Pradal, C., Migault, V., & Pallas, B. (2018). An architecture for the integration of different functional and structural plant models. *Proceedings of the 7th International Conference on Informatics, Environment, Energy and Applications*, 107–113. https://doi.org/10.1145/3208854.3208875

Morales, A., Kottelenberg, D. B., Ernst, A., Vezy, R., & Evers, J. B. (2025). The virtual plant laboratory: A modern plant modelling framework in julia. *In Silico Plants*, *7*(1), diaf005. https://doi.org/10.1093/insilicoplants/diaf005

Oberländer, T. (2023). *GroLink: Implementing and testing a general application programming interface for the plant-modelling platform GroIMP* [Masterthesis, University of Göttingen]. https://wkurth.grogra.de/oberlaender_msc.pdf

Oberländer, T., Heidsieck, G., Hay, T., & Kurth, W. (2025). *GroLink: A general application programming interface for the plant-modeling platform GroIMP*. Zenodo. https://doi.org/10.5281/zenodo.17091817

Pradal, C., Dufour-Kowalski, S., Boudon, F., Fournier, C., & Godin, C. (2008). OpenAlea: A visual programming and component-based software platform for plant modelling. *Functional Plant Biology*, *35*(10), 751–760. https://doi.org/10.1071/FP08084

Smoleňová, K., Henke, M., & Kurth, W. (2012). Rule-based integration of GreenLab into GroIMP with GUI aided parameter input. *2012 IEEE 4th International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications*, 347–354. https://doi.org/10.1109/PMA.2012.6524856

Smoleňová, K., Henke, M., Ong, Y., & Kurth, W. (2013). Rule-based integration of LIGNUM into GroIMP. *Proceedings of the 7th International Conference on Functional-Structural Plant Models, Saariselkä, Finland, Eds.: Risto Sievänen Et Al.*, 214. https://ojs.silvafennica.fi/index.php/fspm2013/article/view/850

Spehle, B. (2023). Conveying the Effects of Climate Change. In *Book of abstracts of the 10th international conference on functional-structural plant models: FSPM2023, 27- 31 March 2023* (p. 151). https://hi.converia.de/custom/media/FSPM2023/FSPM2_2ss24_BOA_2023_v1.pdf

Streit, K., Henke, M., Bayol, B., Cournède, P.-H., Sievänen, R., & Kurth, W. (2016). Impact of geometrical traits on light interception in conifers: Analysis using an FSPM for Scots pine. *2016 IEEE International Conference on Functional-Structural Plant Growth Modeling, Simulation, Visualization and Applications (FSPMA)*, 194–203. https://doi.org/10.1109/FSPMA.2016.7818307

Zhu, J., Gou, F., Rossouw, G., Begum, F., Henke, M., Johnson, E., Holzapfel, B., Field, S., & Seleznyova, A. (2021). Simulating organ biomass variability and carbohydrate

distribution in perennial fruit crops: A comparison between the common assimilate pool and phloem carbohydrate transport models. *In Silico Plants*, *3*(2), diab024. https://doi.org/10.1093/insilicoplants/diab024