

# 1 Introduction

This is a user’s manual for Palander, the Blender add-on for running the Palabos simulation engine. It provides instructions for how to install and use the program, as well as a simple tutorial to get the reader started on how to use it as well as Blender in general. Some knowledge of Linux basics like compiling and `ssh` is recommended to increase the likelihood of a successful installation.

Both Palabos and Blender are open-source tools. In the same spirit, the files of this program are also released as open-source under the conditions of the following licenses:

**GNU GPLv3** for file `palander_addon.py`;

**GNU AGPLv3** for file `palander_engine.cpp`; and

**Beerware License rev. 42** for files `palander_remoterun.sh` and `palander_scriptbase`.

Palander is provided “as is”, and it will most likely not work perfectly for all possible scene setups. Palabos is more of a scientific/engineering tool than a content creation engine. Its main mode of usage is to optimize an executable *for each use case separately*. This means that this program, which is intended as a general tool, must make compromises as to speed, stability, etc. in order to work sufficiently well for as many use cases as possible. If you think you know what you’re doing, you’re invited to try to optimize the executable for your own specific setups; however, control of most of the relevant parameters is already offered through the add-on.

Palander is Copyright © 2017 Animationsinstitut, Filmakademie Baden-Wuerttemberg. It has been realized in the scope of the Media Solution Center project, funded by the state of Baden-Wuerttemberg, Germany.

## 2 Installing Palander

Palander is not a single piece of code, but rather an entity based on the established open-source programs Palabos and Blender. This entity consists of the following five parts:

1. a copy of Blender,
2. `palander_addon.py`, a Python add-on to be installed on top of Blender,
3. `palander_engine.cpp`, a Palabos executable,
4. `palander_remoterun.sh`, a remote running script accessed on Hazel Hen by the add-on, and
5. `palander_scriptbase`, a submission script base for the Hazel Hen batch system.

It is recommended to compile the Palabos executable yourself for your desired platform; a Makefile is provided for compilation on Linux computers. Palabos is primarily intended for parallel use on clusters, which is why its primary platform is Linux. However, it has also been tested and found to work on Windows, albeit not without difficulties in compiling. Pre-compiled executables may be available (without guarantee!) for specific platforms — please inquire the authors.

The instructions below are first for desktop usage; subsection 2.7 is necessary only if you wish to run your simulations on the Hazel Hen supercomputer. Likewise, compiling and installing the Palabos executable on your desktop is not necessary if you only wish to run your (Palabos) simulations on the supercomputer. In this case, you can make test simulations using Blender’s native fluid simulation engine, but please keep in mind that the results might differ considerably from the Palabos simulations.

More in-depth instructions for installing Palabos can be found at the Palabos website under <http://www.palabos.org/documentation/userguide/getting-started.html>.

### 2.1 Installing Blender

Installing Blender is very straightforward: go to <https://www.blender.org/download>, choose your platform, and click on a link to download. Install using the instructions provided on the website.

## 2.2 Installing the Blender Add-On

Palander’s Blender add-on is named `palander_addon.py`. To install it, follow these steps:

1. Start Blender.
2. Go to **File** → **User Preferences** → **Add-ons**.
3. Make sure that the level “Community” is supported.
4. Click on “Install from File” and locate the file in question.
5. In section “Import-Export”, locate “Palander” and checkmark it.
6. Click on “Save User Settings”.

The last two steps are necessary for the add-on to be loaded upon each startup of Blender. To verify that installation was successful, make sure that there is a new tab named “Palander” at the bottom of the tab column on the left-hand side of the Blender window. As with any modifications to the add-on, you might need to restart Blender for the changes to take effect.

If you wish to modify the python add-on yourself after installation, you can simply copy the new version into Blender’s add-on directory. For Linux, this should be something like `~/.config/blender/2.78/scripts/addons`.

## 2.3 Installing the Palabos Library

If you don’t have a pre-compiled Palabos executable, you need to compile one yourself. For this, you first need to install the Palabos library. Officially, you can download this library at <https://www.palabos.org/download-ql>. However, the current (as of April, 2017) newest official release version, v1.5r1, is incompatible with Palander; improvements have since then been made to the library without its being released, and a newer version that is actually compatible is available from the authors of Palander, and is distributed with this installation. Future official releases of Palabos should also be compatible with Palander, and their use is recommended once they become available.

Either way, you should obtain a ZIP file of the Palabos library. Simply unzip it and make sure that you are satisfied with the location into which the library directory is uncompressed.

## 2.4 Compiling and Installing the Palabos Executable (Linux)

Palander comes with the Palabos program file `palander_engine.cpp` and its Makefile. Place these in a directory that you wish to use to store your simulation output. Then, modify the Makefile by changing the following settings as required:

- `palabosRoot`: set this to the root directory where you installed the Palabos library
- `MPIparallel` (default **true**): change this to **false** if you intend to run Palabos on your local computer without installing MPI
- `libraryPaths`: set this to the `externalLibraries` subdirectory of your Palabos installation directory
- `serialCXX` (default `g++`): change this to your serial compiler of choice
- `parallelCXX` (default `mpicxx`): change this to your parallel compiler of choice

In addition, you may want to change the number of compiler threads in the `SCons` definition (parameter `-j`, default `4`).

After making these modifications, you are ready to compile the Palander engine. In Linux, you can do this with the command `make` in the directory where you have the files `palander_engine.cpp` and `Makefile`. It will take a few minutes to compile all of the necessary library functions for the first time, but after this, re-compiling after any modifications to `palander_engine.cpp` will take a much shorter amount of time. After compiling, you will have an executable called `palander_engine`, which the add-on will call from inside Blender when requested to perform a simulation using Palabos.

## 2.5 Compiling the Palabos Executable (Mac OS X)

According to the makers of Palabos, compiling a Palabos executable should work the same way under Mac OS X as under Linux. You simply need to find a `gcc`-compatible compiler such as `xcode` and use the compiler flag `-DPLB_MAC_OS_X`. Then follow the instruction given in subsection 2.4. However, the authors have not tested this and will therefore not be able to provide any support.

## 2.6 Compiling the Palabos Executable (Windows)

If despite all warnings you wish to compile the Palabos executable in Windows, there are several considerations that you have to keep in mind:

1. Please read the Windows installing instructions on the Palabos website.
2. You may need to install new programs and make sure that Code::Blocks (or whichever IDE you use) can reference them. These are
  - an MPI program and
  - zlib.

## 2.7 Installing Palander on Hazel Hen

Since Palander can send a simulation job to be performed on the Hazel Hen supercomputer in HLRS, it is crucial that the user have an account on Hazel Hen. Palander assumes that this is the case, and that it is being run on a computer that is allowed to access the supercomputer, and can do so without needing to prompt for a password. To be granted access, your IP address first needs to be cleared by the HLRS administration, so that you can get through their firewall; then, you need to set up password-free access yourself on your computer. If you don't know how to do this, you can find instructions on the Internet, *e.g.* here: <http://www.rebol.com/docs/ssh-auto-login.html>.

After completing these two steps, you should be able to log in to Hazel Hen using `ssh` (*e.g.* `ssh username@hazelhen.hww.de`). While logged on, you can set up Palander's supercomputer end with the following steps:

1. In your home directory, create the working directory for Palander with the command `mkdir palander`. (NOTE: This directory name is hardcoded into the Python add-on script. If you wish to use another name, make sure to change `palander_addon.py` accordingly!)
2. Copy the files `palander_engine.cpp`, `Makefile`, `palander_remoterun.sh` and `palander_scriptbase` into this directory.
3. Change both compiler flags (`serialCXX` and `parallelCXX`) in `Makefile` to `CC`.
4. Set up the Palabos library and compile the Palabos executable as directed in subsections 2.3 and 2.4.

## 3 Using Palander

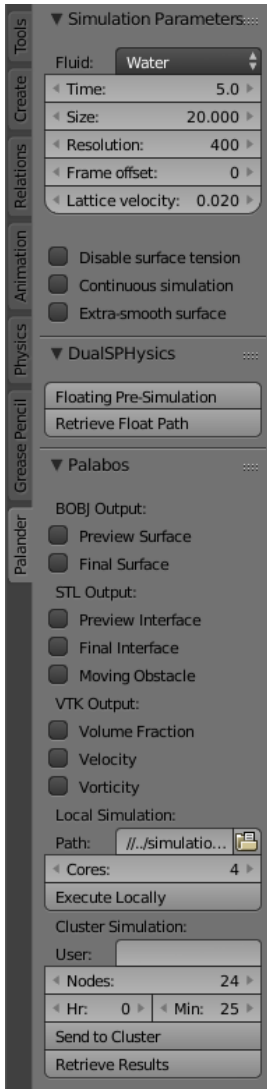
For more in-depth instructions on how to use Blender itself, the reader is encouraged to search for the Blender official online manual, as well as subject-specific third-party tutorials *e.g.* on YouTube. What follows is a cursory set of instructions concentrating on those features of Blender that are relevant to using Palander.

### 3.1 Building a Scene

To make use of Palander, first you need to build a scene as you normally would when using Blender: first add and customize a number of objects, then apply a fluid simulation modifier to each relevant object, and finally launch Palander through the add-on. New objects are available *e.g.* on the “Create” tab on the left-hand side; the fluid simulation modifiers can be found on the “Physics” tab.

For a successful simulation, you need one cuboid domain object, inside of which all of the water simulation happens. The sides of this cuboid act as walls for the fluid, but any non-fluid objects can exist partially or completely outside of the domain — the domain walls won’t affect them. For there to be any fluid inside of the domain, you need to put in one or more fluid or inflow objects. Then, you can use static or dynamic obstacle objects to influence the motion of the fluid during the simulation, as long as this happens within the confines of the domain.

## 3.2 Explicit Palander Parameters



When activated, the add-on is visible as an extra tab entitled “Palander” on the left-hand side of the Blender window. If you click on it, you will see a menu like the one on the left. The available options are the following:

**Fluid:** This controls the fluid viscosity using several given presets (Blood, Gasoline, Honey, Oil, and Water).

**Time:** This is the simulation length in seconds. Palabos outputs 25 frames per second, and so the total number of output frames is 25 times this number.

**Size:** This is the real-world size of the simulation box. If set to zero, it will use Blender’s own value (**Fluid World** → **Real World Size** on the “Physics” tab of the domain object of your fluid simulation). The limit of 10 meters can be bypassed by setting a value on the “Palander” tab.

**Resolution:** The resolution of the simulation, as divisions along the longest edge of the domain box.

**Frame offset:** The frame at which the fluid simulation will start (positive or negative).

**Lattice velocity:** This is the lattice velocity of the simulation. A higher velocity will result in faster simulations, but at the cost of stability. If your simulation keeps crashing, lower this value, but otherwise leave it untouched — the default value is optimal for most uses.

**Disable surface tension:** Disable Palabos’s surface tension algorithm for extra splashiness.

**Continuous simulation:** When enabled, this will save the simulation at the end of the allotted time, and try to load a previously saved state when starting.

**Extra-smooth surface:** This will run a smoothing algorithm on the final mesh twice and on the preview mesh once, as opposed to just the final mesh once when unchecked. Please note that smoothing will cause a loss of detail.

**DualSPHysics & Tangaroa:** These menus are for running external SPH simulations. They serve as an example of how to incorporate other third-party engines, but they will not work without installing the engines in question. Please contact the authors for further details.

**Use Mask:** Use an integer mask to set up the simulation. If the mask file doesn't exist, it will be created and saved for use in further simulations. This can drastically speed up the initiation of simulations containing non-moving obstacles with high vertex counts.

**(BOBJ) Preview Surface:** Output the simulated preview surface directly into Blender's cache folder (local run) or into the remote output directory (remote run).

**(BOBJ) Final Surface:** Output the smoothed final surface directly into Blender's cache folder (local run) or into the remote output directory (remote run).

**(STL) Preview Interface:** Output an STL file of the preview surface into your local simulation directory (see below).

**(STL) Final Interface:** Output an STL file of the final surface into your local simulation directory (see below).

**(STL) Moving Obstacle:** Output an STL file of the moving obstacle into your local simulation directory (see below).

**(VTK) Volume Fraction:** Write the volume fraction into a VTK output file.

**(VTK) Velocity:** Write the velocities into a VTK output file.

**(VTK) Speeds:** Write the velocity components (x, y, z) into three separate VTK files.

**(VTK) Vorticity:** Write the vorticities into a VTK output file. All VTK output will be written in the same files by frame, which will not be created at all if none of the VTK flags are checked.

**Path:** Set the path to your local simulation directory here (the one where you compiled the Palabos executable).

**Cores:** The number of cores (threads) to use for simulating locally with Palabos. Please don't set this higher than the number of cores available on your machine. (NOTE: If you don't have MPI installed on your machine and/or you have compiled your Palabos executable with the `MPIparallel` flag set to `false`, then leave this setting alone!)

**Execute Locally:** Click on this to run the simulation locally using Palabos. Simulation output will be visible in the System Console — please check this to see if there are any problems.



**User:** Insert your Hazel Hen user name here. This is necessary to connect to the right account automatically.

**Nodes:** The number of Hazel Hen nodes to use for the simulation. The simulation will automatically run on 24 cores per node.

**Hr:/Min:** The wall time reserved for the simulation.

**Send to Cluster:** Click on this to run the simulation on the Hazel Hen supercomputer. The simulation progress cannot be monitored locally; you must log in to Hazel Hen and view the progress there, or try to retrieve the results with the button below.

**Retrieve Results:** Click on this to retrieve the simulation results from Hazel Hen.

### 3.3 Implicit Interface Parameters

In addition to the explicit parameters, Palander also uses some of Blender’s own simulation parameters. These of course include all objects from the scene that have been flagged for fluid simulation, such as one domain, any number of obstacles, fluids or inflows, and one outflow. Furthermore, the following parameters are implicitly passed on to the Palabos simulation:

**Fluid cache:** From the directory path in **Fluid** on the “Physics” tab of the domain.

**Real World Size:** From **Fluid World** → **Real World Size** on the “Physics” tab of the domain.

**Inflow Velocity:** From **Fluid** on the “Physics” tab of each inflow.




Also, Palander supports a certain number of different keyframe actions. These are

**Animated obstacles:** Objects that are flagged as obstacles for the fluid simulation and that are animated using keyframes will have their motion paths exported to Palabos for the simulation.

**On/off inflows:** Objects that are flagged as inflows for the fluid simulation and that are enabled/disabled using keyframes will be activated/deactivated accordingly in the Palabos simulation.

## 4 Tutorial: Dam Break

This tutorial will show how to build and simulate a simple dam break scenario.

1. Open Blender.
2. You should see the startup scene, which contains a single cube. Click on the “Object” tab () . Under **Transform** → **Scale**, change the scale along the  $y$ -axis to 3.
3. To make viewing inside the domain easier, click on “Viewport Shading” () and select “Wireframe”.
4. Create a new cube from the “Create” tab.
5. Move it to one end of the previous cube and scale it down a bit, *e.g.* in **Transform** → **Location**  $Y \rightarrow 2.0$  and  $Z \rightarrow -0.2$  and in **Transform** → **Scale**  $Z \rightarrow 0.8$ .
6. Create a third cube, scale it down, and move it downstream, *e.g.* in **Transform** → **Location**  $Y \rightarrow -2.0$  and  $Z \rightarrow -0.8$  and in **Transform** → **Scale**  $X \rightarrow 0.4$ ,  $Y \rightarrow 0.2$ , and  $Z \rightarrow 0.2$ .
7. Rename the cubes to make it easier to tell them apart. In the “Outliner” tab (top right corner, contains a list of all objects), right-click on each cube and select “Rename”: *e.g.* “Cube” → “Domain”, “Cube.001” → “Water”, and “Cube.002” → “Block”.
8. Select the domain by either clicking on it in the “Outliner” tab or right-clicking on it in the viewport. Click on the “Physics” tab () , then click on “Fluid”. From the “Type” dropdown menu, select “Domain”. From the new options that now appear below, change **Fluid World** → **Real World Size** to 3.0.
9. Select the water, click on “Fluid”, and select “Fluid” from the menu.
10. Select the block, click on “Fluid”, and select “Obstacle” from the menu. Below, select “Shell” from “Volume Initialization”.
11. Go back to “Solid” in the “Viewport Shading” menu. Select the domain again.
12. Click on “Bake” on the “Physics” tab. There! Now you have started your first simulation using Blender’s native engine. You can view its progress from the bar under the text “Fluid Simulation” on the top of the screen.

13. To view the results, first click on the eye symbol on the “Water” row in the “Outliner” tab. This will remove the water object from view, but the results are actually contained in the domain object. Now, pressing the play button at the bottom of the screen will show a preview of what the results look like.

You should realize that the resolution of the simulation is quite low by default (65/45). This means that when you preview the results in the viewport, there are only 45 subdivisions along the long edge of the domain! To view the higher-resolution results (65), select “Final” from **Fluid** → **Viewport Display** just below the resolution settings. This isn’t much better — to see a clear difference, set the resolutions to 200/100 and bake the simulation again. You can see that the simulation is much slower, but if you have the patience to wait, you can again compare the Preview and Final resolutions afterwards. They should now look more like an actual fluid instead of a moving collection of blocks.

Now you can try to run your simulation using Palabos. In many cases, Palabos can be slower than Blender’s native engine, so you might want to start with a resolution of 100 to see how fast the simulation can progress. This time, you need to change the resolution from the “Palander” tab on the left-hand side of the viewport. Change any other parameters that you wish (*e.g.* click on “Disable surface tension” to try to mimic the Blender results), including the path, which you should set to the directory where you have installed Palander (if you’re running Palabos locally), or the user name if you want to run on Hazel Hen. Before clicking on “Execute Locally” or “Send to Cluster”, make sure that your water object is active in the viewport — you may have deactivated it by clicking on its eye symbol in order to better see the results of the previous simulation. Inactive objects will not be passed on to Palabos!

## 5 Tips and Best Practice

When you run Palabos locally, the simulation results are automatically output into the same directory where Blender places its own fluid simulation results. Consequently, you can preview the results as they come in, and you don't need to import anything into Blender for further processing and rendering. However, if you run Palabos on Hazel Hen, you will need to transfer the results into Blender's cache directory before being able to view them. You can do this easily by clicking on the "Retrieve Results" button on the "Palander" tab. Please note that there is currently no algorithm to check what you have previously downloaded; this button will retrieve all the results, starting from the first frame, each time that it is pressed.

Also note that there is no explicit connection between your Blender setup and the retrieved results; rather, the algorithm retrieves the results from the newest workspace, *i.e.* from the most recently started simulation job. For example, this means that if you are working on two projects, and you run a fluid simulation in each of them, you can only use the "Retrieve Results" button to download the results from the simulation that started to process later on the supercomputer. To retrieve any other results, you must log in to Hazel Hen yourself and download the desired files manually. This will not be a problem if you run and process one simulation at a time.

If the simulation doesn't finish in the time allotted, the result files are not transferred from the workspace to your home directory on Hazel Hen. This means that when the workspace expires, your results will be lost! By default, Palander reserves workspaces for 10 days, so you have plenty of time to retrieve your results before that happens. Simulations that have time to finish will be copied to your home directory, into a sub-directory denoted by `run` and the date and time of the start of the simulation. These files are safe even after the corresponding workspace expires, but you cannot use the "Retrieve Results" button to download them anymore.