

PhysiCell Simulation Prep

Integration is achieved through PhysiCellDataLoader

https://github.com/elmbeech/physicelldataloader/blob/master/man/TUTORIAL_blender.md

Elmar has done a great job on this documentation.

Start by setting up pcdl

<https://github.com/elmbeech/physicelldataloader/blob/master/man/HOWTO.md> (4/7/26)

How to install pcdl?

```
pip3 install pcdl
```

How to load the pcdl library?

```
import pcdl
```

How to check for the current installed pcdl version?

```
import pcdl
pcdl.__version__
```

How to test if the command line interface pcdl functions work?

Execute the following command on the command line.

Note: depending on your operating system and command line shell, tab completion might or might not work.

```
pcdl_get_version path/to/PhysiCell/output
```

If you get an error like: `pcdl_get_version: command not found`, please, follow the troubleshooting guide on the github.

Generate vtk files from PhysiCell output

Generate vtk files from the command line

```
pcdl_make_cell_vtk output --ext .vtp # blender bvtk nodes compatible filename and extension.
```

Generate vtk files from within python

```
import pcdl
```

```
mcdsts = pcdl.TimeSeries('output')
mcdsts.make_conc_vtk()
```

```
mcdsts.make_cell_vtk(ext='.vtp') # blender bvtk nodes compatible filename and extension.
```

1.

Blender setup

Intergration is achieved through PhysiCellDataLoader

https://github.com/elmbeech/physicelldataloader/blob/master/man/TUTORIAL_blender.md

Elmar has done a great job on this documentation.

Blender vtk nodes plugin installation

This installation is a little tricky, but doable. Please follow the bvtk nodes installation instructions, including workspace setup.

- <https://bvtknodes.readthedocs.io/en/latest/BVTKNodes.html#installation>

Installation

Install Blender (if needed, see [instructions](#)). - Can use Blender v5.1

Install VTK to Blender Python as instructed above in [Installation of VTK for Blender](#).

- Run commands on normal terminal (**This must work for BVTK to be functional**)
 - `cd /path/to/blender-5.1/5.1/python/bin`
 - `./python3.10 -m ensurepip`
 - `./python3.10 -m pip install vtk (optional: ==9.2.6)`

Testing

- You can test if VTK is found in [Blender Python Console](#) (by default located in the Scripting [workspace](#)) with commands (**This must work for BVTK to be functional**)
 - `import vtk`
 - `vtk.vtkVersion().GetVTKVersion()`

Download appropriate BVTKNodes add-on ZIP file (see options in [Available Versions of BVTKNodes addon](#) - use [tkeskita/BVtkNodes](#)). To download add-on from Github, Select “Code”, then “Download ZIP”.

- <https://github.com/tkeskita/BVtkNodes>

Using BVTK in Blender

- Start Blender, go to “Edit” → “Preferences” → “Add-ons” → “Install” → open the add-on zip file.
- Activate the “BVTKNodes” add-on in Preferences by clicking on the checkbox. Add-on is located in Node category, “Community” level of Blender add-ons.
- Follow the workspace setup (Option: add Blender Text Editor as described)

- <https://bvtknodes.readthedocs.io/en/latest/BVTkNodes.html#workspace-setup>

Load vtk polynomial data vtk files

1. In the BVTk Node Tree Workspace, click New to add a Node Tree
2. In the BVTk Node Tree Workspace, right click Add / Reader / vtkXMLPolyDataReader
3. In the BVTk Node Tree Workspace, right click Add / Converters / VTKtoBlenderMesh
4. Connect vtkXMLPolyDataReader output with input VTKtoBlenderMesh
5. vtkXMLPolyDataReader FileName: path/to/output000000000.vtp
6. VTKtoBlenderMesh: click Update Node

Load vtk polynomial data vtk files as a time course

1. In the BVTk Node Tree Workspace from the previous section, right click Add / Custom / TimeSelector
2. Connect vtkXMLPolyDataReader output with input TimeSelector
3. Connect TimeSelector output with input VTKtoBlenderMesh
4. In the BVTk Node Editor, press **N** to open the sidebar
5. Select the **Inspect** tab and change: **Update Mode** → **Update All Automatically**
6. Click update node on each element in the BVTk Node Editor and force update upstream on VTKtoBlenderMesh
7. Set your scene frame range to:
 - a. **Start = 1** (this selects output000000000.vtp)
 - **End = # of .vtp files in the sequence** (the last output file number plus 1)
8. Press play on the animation.

ChatGPT walkthrough of Blender setup

Prompt

Prompt: You are an expert in Blender, VTK, the BVTK blender addon, PhysiCell, and PhysiCellDataLoader (pcdl).

I am trying to integrate VTK and Blender on a set of files that comprises a time course of simulated data. Using pcdl I generated .vtp xml files for each time point. I can get the readers in Blender using BVTK to show one file at a time, but now I need to play the full animation from start to finish by looking at each of the time point files. Make a step by step walk through with details for how to setup my Blender environment so it will play the time course of my simulation.

ChatGPT 5.2 Pro Response

Below is the workflow I would use for a PhysiCell/pcdl .vtp time series (*can be generated with 9.5.1 in pcdl*) in current *tkeskita/BVtkNodes*, which is the BVTK line documented for **Blender 3.6 LTS or 4.2** (*can use 5.1*) and **VTK 9.2.6** (*can use 9.6.2*). I'm assuming that version of the add-on, not the old Blender 2.79 branches, because the node names and update behavior differ there. Also, PyPI's current latest *vtk* is newer than the version the add-on docs target, so pinning *vtk==9.2.6* (*can use 9.6.2*) is the safer match for this setup. ([GitHub](#))

The key idea is this: for a folder of **one .vtp per time point**, BVTK does **not** magically animate the reader by itself. The normal pattern is:

vtkXMLPolyDataReader -> Time Selector -> (your existing pipeline) -> VTK To Blender Mesh

With **Use Scene Time** enabled, BVTK copies Blender's current frame into the **Time Selector** on every frame change. For readers that are not exposing internal time steps, BVTK then swaps the **numeric suffix at the end of the filename** so the reader points at the matching file in the sequence. For the downstream geometry to actually rebuild every frame, the global BVTK update mode must be **Update All Automatically**. ([GitHub](#))

1) Install a version combination that BVTK actually expects

Use Blender **3.6 LTS** or **4.2** (*can use 5.1*), then install the *tkeskita/BVtkNodes* ZIP from GitHub as an add-on. BVTK's docs explicitly say VTK must be available **inside Blender's Python environment**, not just in your normal system Python. ([GitHub](#))

Run these commands with **Blender's bundled Python executable**:

```
python -m ensurepip --upgrade
python -m pip install --upgrade pip
```

```
python -m pip install vtk==9.2.6 (can use 9.6.2)
```

`ensurepip` is the standard way to bootstrap `pip`, and `python -m pip` is the recommended invocation pattern. Using `vtk==9.2.6` (*can use 9.6.2*) matches the version the current BVTk docs target. ([Python documentation](#))

Then in Blender:

1. **Edit** → **Preferences** → **Add-ons** → **Install...**
2. Select the downloaded **BVTkNodes ZIP**
3. Enable **BVTkNodes**

A quick verification inside Blender's Python Console is:

```
import vtk
vtk.vtkVersion().GetVTKVersion()
```

That is also the version check the BVTk docs recommend. ([GitHub](#))

2) Make sure your **.vtp** files follow BVTk's filename rule

This is the part that matters most for a pcdl-generated time series. The **Time Selector** fallback for time-unaware readers only works when the reader filename ends with **digits immediately before the extension**. Good examples:

```
cells_0000.vtp
cells_0001.vtp
cells_0002.vtp
...
```

Also good:

```
snapshot_100.vtp
snapshot_110.vtp
snapshot_120.vtp
```

Bad examples for automatic stepping:

```
cells_t0000_final.vtp # digits are not the last thing before .vtp
frame_01_cells.vtp   # same problem
```

BVTK searches the data directory for files with the **same basename pattern + trailing integer + same extension**, sorts them by that integer, and then picks files by sequence position. The integer values do **not** have to be continuous. ([GitHub](#))

Two important consequences:

- **Frame 1 maps to the first file in the sorted list**, not to the numeric value in the filename.
- If you go past the last file, BVTK **loops back to the start** using modulo logic.

So if your first file is `cells_0100.vtp`, Blender frame **1** should still load that file. Do **not** set Blender's start frame to 100 for this workflow. Also avoid starting at frame 0, because the code maps with $(\text{time_index} - 1) \% n$, so frame 0 wraps to the last file. ([GitHub](#))

3) Build a BVTK workspace once, then reuse it

BVTK's docs recommend creating a dedicated workspace:

1. Start a new Blender file.
2. Duplicate the Layout workspace and rename it to **BVTK**.
3. Change one area to **BVTK Node Tree**.
4. Change another area to **Text Editor**.
5. In the BVTK Node Editor, click **New** to create a new BVTK node tree. ([GitHub](#))

If you are on Blender 3.6 and the normal Add-menu search does not find VTK nodes, the BVTK docs specifically say to use **F3 Menu Search** instead. ([GitHub](#))

4) Start from the single-file pipeline that already works

Since you can already display one `.vtp` file, keep that pipeline and make the smallest possible change:

before:

vtkXMLPolyDataReader -> (filters/color/glyphs/etc.) -> VTK To Blender Mesh

after:

vtkXMLPolyDataReader -> Time Selector -> (same filters/color/glyphs/etc.) -> VTK To Blender Mesh

The **Time Selector** is meant to be connected **immediately after a VTK reader**. On the reader node, set **FileName** to the **first file in the sequence**. On the Time Selector node, leave **Use Scene Time = ON**. With that setting enabled, Blender's timeline frame controls the Time Selector automatically. ([GitHub](#))

If you are following old tutorials, note that the current [tkeskita/BVtkNodes](#) branch uses **VTK To Blender Mesh** as the main export node rather than the legacy **VTK To Blender** node. ([GitHub](#))

5) This is the setting people usually miss: switch BVTK to full automatic updates

In the BVTK Node Editor, hover the mouse there and press **N** to open the sidebar. Go to the **Inspect** tab and change:

Update Mode → **Update All Automatically**

This is the critical switch. The add-on's settings default to **Update Current Automatically**, not **update-all**. On a frame change, the add-on updates Time Selector / Global Time Keeper values from `scene.frame_current`, but it only calls the full downstream rebuild automatically when the update mode is **update-all**. If you leave the default, the frame number may change while the mesh appears stuck. ([GitHub](#))

6) Initialize the pipeline once

Now click the final **VTK To Blender Mesh** node and run **Force Update Upstream** once. BVTK documents that operator as the “make sure everything upstream really refreshes” option for cases where the update system does not detect the need on its own. ([GitHub](#))

At this point, you should be able to:

1. Scrub the timeline
2. Watch the reader switch files through the Time Selector
3. See the Blender object update

If you want an easy debug view, insert an **Info** node right after the Time Selector. The docs recommend it whenever you want to inspect what is currently flowing through the pipeline: data type, point/cell counts, bounds, and available arrays. ([GitHub](#))

7) Set the frame range to match the number of files

Set your scene frame range to:

- **Start = 1**
- **End = number of .vtp files in the sequence**

Blender's Frame Range controls how long the animation lasts in frames. For your BVTk file series, setting `End = Nfiles` gives you one clean pass through the sequence without wrapping. ([Blender Documentation](#))

Example:

If the folder contains:

```
cells_0000.vtp
cells_0001.vtp
...
cells_0359.vtp
```

set:

- Start = 1
- End = 360

Then the mapping is:

- Frame 1 → `cells_0000.vtp`
- Frame 2 → `cells_0001.vtp`
- ...
- Frame 360 → `cells_0359.vtp`

If you set End higher than 360, BVTk will loop back to the first file. ([GitHub](#))

8) Press play; if needed, render instead of relying on real-time playback

Once the frame range and update mode are set, press Play in the Timeline. The add-on registers frame-change handlers and updates Time Selectors that use Scene Time from Blender's current frame, which is what makes the sequence follow the timeline. ([GitHub](#))

For final output, set the output path and file type in Blender's Output settings, confirm the frame range, and use **Render Animation**. Blender's manual describes the frame range as controlling animation length and the Output panel as the place to choose where rendered animation frames go. ([Blender Documentation](#))

9) A few PhysiCell/pcdl-specific practical notes

If your `.vtp` data is already working for one frame, do **not** redesign the whole pipeline just for animation. Keep the same downstream nodes you already use for a single time point and insert only the **Time Selector** after the reader. That is the least fragile way to convert a working single-frame setup into a time-series setup. The Time Selector is exactly the node BVTk provides for transient data in a reader-driven pipeline. ([GitHub](#))

If you are coloring the mesh with **Color Mapper**, place it after the Time Selector and before **VTK To Blender Mesh**. BVTk's docs note that Color Mapper colors show in Blender when the viewport shading is **Material Preview** or **Rendered**. ([GitHub](#))

If your current visualization is built on **VTK To Blender Particles** rather than Mesh, there is an extra caveat: BVTk's docs say particle data may require **Update Node** after frame changes and the 3D Viewport may need a click to refresh, although rendering should still be correct. ([GitHub](#))

10) Optional advanced setups

If you have **multiple synchronized file series** — for example one `.vtp` sequence for cells and another for a vessel network or some derived surface — create a separate

reader -> Time Selector

branch for each sequence. Because BVTk updates all Time Selector nodes with **Use Scene Time** from the same `scene.frame_current`, they stay synchronized automatically. ([GitHub](#))

If you want non-1:1 time mapping — pauses, slow motion, skipping, or custom time warping — then disable **Use Scene Time** and use the **Global Time Keeper** route instead. BVTk documents that as the alternative path when Scene Time is off. ([GitHub](#))

Fast troubleshooting

If the animation still does not advance, check these in order:

- **The mesh never changes:** Update Mode is probably not **Update All Automatically**. Switch it, then run **Force Update Upstream** once. ([GitHub](#))
- **The frame number changes but the reader stays on one file:** your filenames likely do not end in a trailing integer right before `.vtp`, or the sequence files do not share the same basename pattern in one folder. ([GitHub](#))
- **Frame 0 shows the wrong file:** that is expected from the modulo logic. Start the scene at **frame 1**. ([GitHub](#))
- **Your files start at 100 or 1000:** that is fine. Keep Blender start frame at **1** anyway; BVTk maps by sorted sequence position, not the literal filename number. ([GitHub](#))

- **You cannot find BVTk nodes in Blender 3.6:** use **F3** instead of the Add-menu search bar. ([GitHub](#))

If you want, I can turn this into a node-by-node recipe for a typical PhysiCell cell-scatter **.vtp** pipeline, for example using **points** → **glyphs/particles** versus **surface mesh**.