Quickstart Guide to PhysiCell

Download

We currently provide 2 options for downloading the PhysiCell source code:

- https://sourceforge.net/projects/physicell/.
- https://github.com/MathCancer/PhysiCell/releases.

For more detailed information, see Section 3 of the User Guide (in `/documentation`) and/or also http://www.mathcancer.org/blog/physicell-tutorials/.

Build: the basics

PhysiCell is written in C++ and should build on any of the three major operating systems. The one **requirement is that your compiler support OpenMP**. If, during your build (make) process, you get a message like: `error: unsupported option '-fopenmp'`, you'll know your Makefile is trying to use a compiler that doesn't support OpenMP. You may need to install an OpenMP-supported compiler and/or edit the Makefile to use it.

Windows

The currently preferred way to use PhysiCell on Windows is to install MinGW-w64 which will let you use a version of GCC that supports OpenMP. For more detailed information, see our blog post.

(Sometime in the near future, we plan to provide an option to use native Windows and a recent Microsoft Visual C++ compiler, using CMake to build PhysiCell. If you want to be a beta-tester, contact us).

OSX

Unfortunately, the C++ compiler provided by the latest version of XCode on OSX does not support OpenMP. To resolve this, we recommend using the `brew` package manager to install a recent version of `gcc`. Follow the brew installation instructions.

After installing brew, type `brew install gcc` from a Terminal command line. This should install a recent version of gcc/g++ (supporting OpenMP) into `/usr/local/bin`. You can verify this with (note the g++ version # will change over time):

```
$ ls -l /usr/local/bin/g++*
   ...             /usr/local/bin/g++-10@ -> ../Cellar/gcc/10.2.0/bin/g++-10
```

Set the following environment variable in your Terminal's shell, e.g., in the bash shell:

```
$ export PHYSICELL_CPP=/usr/local/bin/g++-10
```

and the Makefile will use it. You should permanently set this in your environment via:

```
$ echo export PHYSICELL_CPP=g++-10 >> ~/.bash_profile.
```

Linux

If you're a Linux user, you probably already have or know how to install/use a proper g++ environment for building PhysiCell. If not, contact us!

Build: sample projects

We provide several sample projects to help you get started. Most of the projects are 2D models (*biorobots, anti-cancer biorobots, cancer heterogeneity, virus-macrophage, predator-prey-farmer, and worms*); but one project is a 3D model (*cancer immunology*). The procedure to build and execute each of the sample projects follows the same pattern. For example, from your Terminal, in the root PhysiCell directory/folder:

```
$ make biorobots-sample     # populate the biorobots sample project
$ make                      # compile the project
```

Assuming the project builds without errors, you should now have an executable called `biorobots` which you can run, e.g.:

```
$ ./biorobots
```

This will begin the simulation, write information to your terminal, and generate output files of types `.svg`, `.xml`, and `.mat`. More about those below. You can `Control-c` to kill the simulation early, if you want.

For the remaining example projects provided with PhysiCell, you would follow similar steps, but first, you may want to clean out the previous simulation's output and prepare for the new one:

```
$ make data-cleanup   # Delete output data. (Optionally, manually move it to another
directory to keep it)
$ make reset          # clear out the sample project / clean slate
$ make cancer-biorobots-sample    # populate the new project
$ make                            # compile the project
$ ./cancer_biorobots
```

and

```
$ make data-cleanup
$ make reset
$ make heterogeneity-sample
$ make
$ ./heterogeneity
```

and

```
$ make data-cleanup
$ make reset
$ make virus-macrophage-sample
$ make
$ ./virus-sample
```

and

```
$ make data-cleanup
$ make reset
$ make pred-prey-farmer
$ make
$ ./pred_prey
```

and

```
$ make data-cleanup
$ make reset
```

```
$ make worm-sample
$ make
$ ./worm
```

and the 3-D model (more computationally intensive):

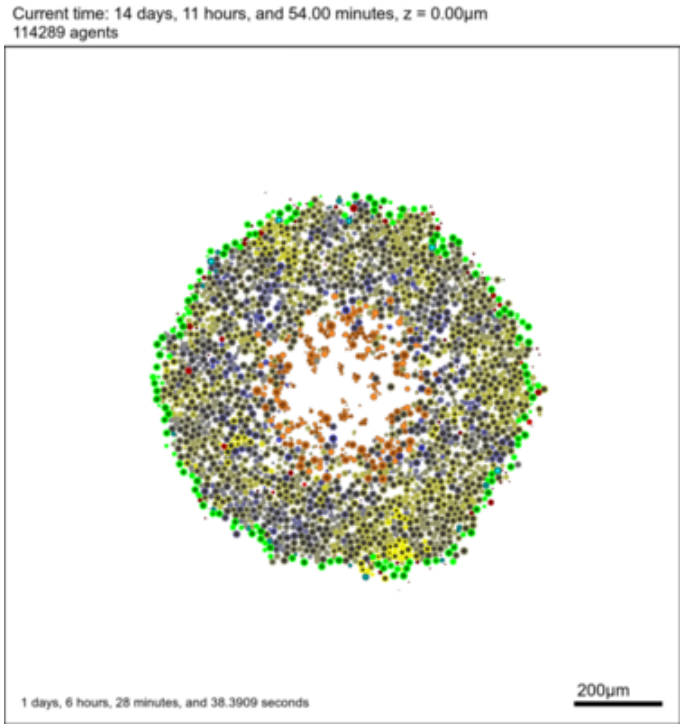
```
$ make data-cleanup
$ make reset
$ make cancer-immune-sample
$ make
$ ./cancer_immune_3D
```

Visualizing Output

PhysiCell does not currently provide a GUI for visualizing output results. Our approach, for now, is to suggest and offer guidance on using other tools, e.g. your browser, ImageMagick, MATLAB, Octave, Python, and ParaView.

Browser

At a bare minimum, you should be able to use your browser to `File -> Open` any .svg (scalable vector graphics) file that your simulation generates. PhysiCell simulates transmitted light microscopy to create virtual pathology images for the .svg files. Even for 3D models, 2D cross-section images (.svg files) are generated, by default, using a slice through the Z=0 plane, as depicted in the following image (from the cancer-immune-sample project).



MATLAB/Octave

If you have access to MATLAB (or Octave), we have a detailed tutorial on how to visualize the MultiCellDS digital snapshots (.xml and .mat files).

Python

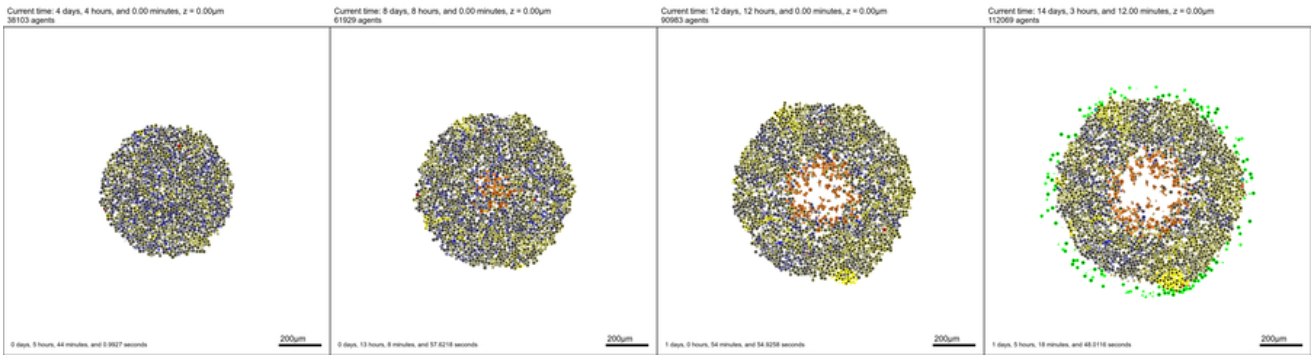
If you want to use Python to visualize results, see this blog post http://www.mathcancer.org/blog/python-loader/ and also see various scripts in the `/beta` directory, e.g., `anim_svg.py` .

ImageMagick

If you are able to install ImageMagick (with SVG support) on your computer, you will have access to several image processing command line tools that will let you filter and arrange images. For example, the following commands:

```
$ montage -geometry +0+0 -tile 4x1 snapshot00000100.svg snapshot00000200.svg
snapshot00000300.svg snapshot00000400.svg tmp.png
$ convert -resize 15% tmp.png out.png
```

will generate a tiled horizontal sequence of images:



ImageMagick will also let you generate an animated gif of your results, e.g.:

```
$ convert snapshot000034*.svg foo.gif
$ magick animate foo.gif    # may be huge, if original SVGs were; downsize in
following steps
$ convert foo.gif -coalesce tmp.gif
$ identify snapshot00003471.svg    # get size of a single image (e.g. 1500x1605)
$ convert -size 1500x1605 tmp.gif -resize 20% small.gif
$ magick animate small.gif
```

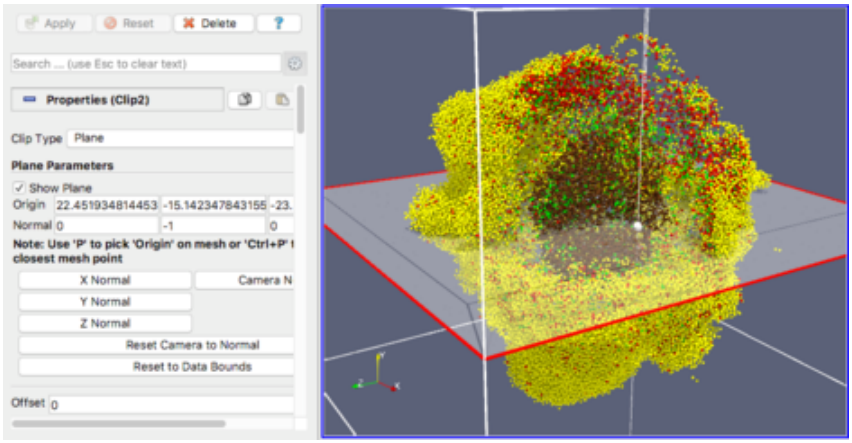
Since PhysiCell 1.8.0, there are helper targets in the Makefile that will perform various functions, e.g.:

```
$ make jpeg   # convert all output/snapshot*.svg to .jpg
$ make gif    # create out.gif from output/snapshot*.svg
```

```
$ make movie  # assuming you have ffmpeg, create out.mp4 from output/snapshot*.jpg
  You can also specify the name of the output directory, e.g.:
$ make jpeg OUTPUT=out1
```

ParaView

If you install ParaView, you can visualize and interact with output from 3D models (the `.mat` files). Refer to our blog post to get started.



Support

Please submit questions and report problems at https://sourceforge.net/p/physicell/tickets/ and follow us on Twitter (https://twitter.com/PhysiCell and https://twitter.com/MathCancer).

References

- http://physicell.mathcancer.org/
- http://www.mathcancer.org/blog/setting-up-a-64-bit-gcc-environment-on-windows
- http://www.mathcancer.org/blog/setting-up-gcc-openmp-on-osx-homebrew-edition/
- http://www.mathcancer.org/blog/physicell-tutorials/
- http://www.mathcancer.org/blog/working-with-physicell-snapshots-in-matlab/
- http://www.mathcancer.org/blog/python-loader/
- https://github.com/MathCancer/PhysiCell/blob/master/documentation/User_Guide.pdf