Slides, videos, links and more:
https://github.com/physicell-training/ws2023

Setting up MacOS for PhysiCell

Randy Heiland and John Metzcar

@PhysiCell

PhysiCell Project

July 2023

Presentation overview

- OpenMP-enabled g++ (using Homebrew)
- Test building the default model ("heterogeneity")

Minimum setup

- Python 3 (using Anaconda distribution)
- Test building an intracellular model
- ImageMagick
- PhysiCell Studio

Studio setup

- pcDataLoader
- COPASI
- C++ code editor
- Git and GitHub

Hackathon setup + additional options

Brief notes on this install guide

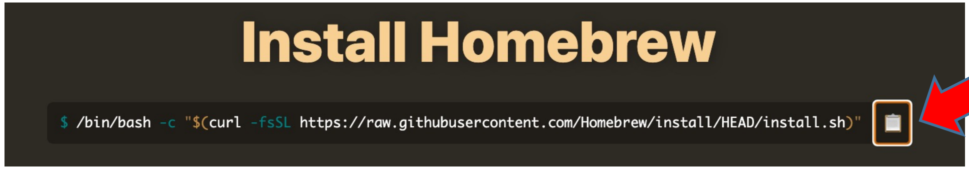
- We tried to make all the Terminal commands **bold face** and able to be directly copied (command + c) and pasted (command + v) directly into the Terminal
- Note that this is a static document – it is possible that the commands could vary slightly as version numbers change.
- Apple Intel CPU vs. Silicon (M1/M2) CPU
  - ♦ You may experience some problems with our setup instructions if you have the newer Apple Silicon CPU. If so, please contact us (see Support links at end).

Overview

- OpenMP-enabled g++ (using Homebrew)
- Test building the default model (“heterogeneity”)
- Python 3 (using Anaconda distribution)
- Test building an intracellular model
- ImageMagick
- PhysiCell Studio
- pcDataLoader
- COPASI
- C++ code editor
- Git and GitHub

OpenMP-enabled g++

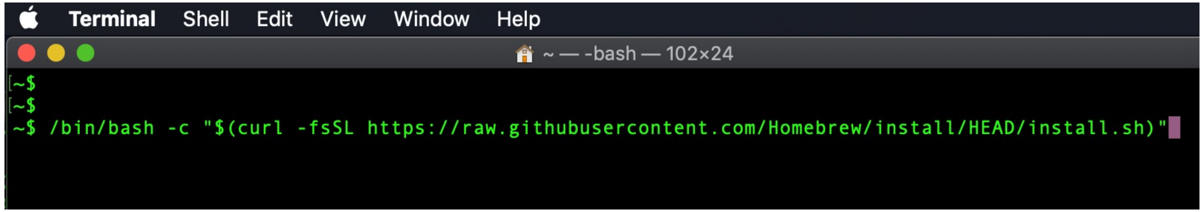
- The default /usr/bin/g++ (clang) that comes with macOS is not OpenMP-enabled. You need to install one that is.
- Homebrew (a package-manager for macOS) will let you do this.
- https://brew.sh/



```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Click to copy the bash command

- Open a new 'Terminal' window and paste the copied command there:



```
~$
~$
~$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Continue to next slide →

https://docs.brew.sh/Installation - additional useful information if needed

Press ‘return’ to execute the command you copied into the Terminal window. Then ‘return’ again to continue the installation of Homebrew.

This will take a few minutes, depending on your network speed.

When installation is finished, you can see where it got installed using the “which” command:

which brew

This may result in:
/usr/local/bin/brew
or, possibly here, on the Mac M1/M2:
/opt/homebrew/bin/brew
or possibly a different path, but it’s not that important, except for some of the following instructions.

Continue to next slide →

Once you've completed installing the basic Homebrew package manager, proceed to install an OpenMP-enabled g++ using the Terminal command:

```
brew install gcc
```

Again, this will take a few minutes. This page https://formulae.brew.sh/formula/gcc will tell you the current stable gcc/g++ version, e.g., 13.1.0

When it completes, you can again use "which" to see where it is installed:

which g++-13

Again, this may be:
/usr/local/bin/g++-13
or,
/opt/homebrew/bin/g++-13
or whatever path homebrew used as its default directory.

Continue to next slide →

Once you've completed installing the basic Homebrew package manager, proceed to install an OpenMP-enabled g++ using the Terminal command:

```
brew install gcc
```

Again, this will take a few minutes. This page https://formulae.brew.sh/formula/gcc will tell you the current stable gcc/g++ version, e.g., 13.1.0

When it completes, you can again use "which" to see where it is installed:

which g++-13

Again, this may be:
/usr/local/bin/g++-13
or,
/opt/homebrew/bin/g++-13
or whatever path homebrew used as its default directory.

If you use this install guide later (much after July 2023), this is an example of a command that may need updated. Homebrew will, at some point, switch to g++-14. There are other similar points later in the presentation – pretty much any time there is a version number – that version number might have changed by the time you use the guide!!!

Continue to next slide →

PHYSICELL_CPP

- As described in the Quickstart guide: Quickstart (*note – this is deprecated), but may be helpful if you have trouble – hopefully the video will get you where you need to be!*

You need to define an environment variable that will point to this g++ so that a PhysiCell Makefile will know to use it.

```
export PHYSICELL_CPP=g++-13
```

(Your **PATH** env var should have the full path to homebrew's /bin in it and therefore be able to find it)

- Furthermore, we recommend that you make this a permanent feature of any new Terminal Shell window that you open. To do this, you want to copy/paste the above **export** command into a special, existing configuration file. This file will be in your HOME directory (type: **echo $HOME**) and the name of the config file will depend on the type of shell that you are using – most likely either "bash" or "zsh". To find out which, run:

```
echo $SHELL
```

It should print out either: **/bin/bash** or **/bin/zsh**
If you are using "bash", you should have a **.bash_profile** file (has a preceding "."); if "zsh" then a **.zshenv** file in your home directory. If this file does not exist, you will need to create it. From a Terminal shell do:

```
cd ~      # go to your home directory
touch .bash_profile      # or for zsh,  touch .zshenv
```

https://support.apple.com/guide/terminal/use-environment-variables-apd382cc5fa-4f58-4449-b20a-41c53c006f8f/mac for more about env vars

PHYSICELL_CPP (cont'd)

- To permanently put the previous **`export`** command into your shell's configuration file, so that it is executed each time a new shell is opened, run one of the following in your Terminal (again, depending on which shell you are using):

```
echo export PHYSICELL_CPP=g++-13 >> ~/.bash_profile
```

or,

```
echo export PHYSICELL_CPP=g++-13 >> ~/.zshenv
```

- When you open a *new* Terminal shell, you can verify that this is defined:

```
echo $PHYSICELL_CPP
```

You should see this printed out:

```
g++-13
```

PHYSICELL_CPP (cont'd)

- To permanently put the previous **`export`** command into your shell's configuration file, so that it is executed each time a new shell is opened, run one of the following in your Terminal (again, depending on which shell you are using):

```
echo export PHYSICELL_CPP=g++-13 >> ~/.bash_profile
```

or,

```
echo export PHYSICELL_CPP=g++-13 >> ~/.zshenv
```

- When you open a *new* Terminal shell, you can verify that this is defined:

```
echo $PHYSICELL_CPP
```

You should see this printed out:

```
g++-13
```

This instruction appears several times. We really mean it. When you start a new terminal, that causes multiple programs to relaunch. In this case, you will "source" your bash_profile/zshenv.

Overview

- OpenMP-enabled g++ (using Homebrew)
- Test building the default model (“heterogeneity”)
- Python 3 (using Anaconda distribution)
- Test building an intracellular model
- ImageMagick
- PhysiCell Studio
- pcDataLoader
- COPASI
- C++ code editor
- Git and GitHub

Test build/run: PhysiCell model (1)

- Get PhysiCell
  - Open browser and navigate to https://raw.githubusercontent.com/physicell-training/ws2023/main/setup/get_physicell.py
  - Using the menu, go to File → Save Page As → **get_physicell.py** in **Downloads**
    - Can view other releases and release notes at: https://github.com/MathCancer/PhysiCell/releases
- Unzip PhysiCell

Assuming the get_physicell.py is downloaded to your ~/Downloads directory:

```
~/Downloads$ python get_physicell.py
~/Downloads$ mv PhysiCell.zip ~     # move this .zip file to your home directory
~/Downloads$ cd ~                   # change to home directory
~$ unzip -q PhysiCell.zip
~$ cd PhysiCell
~/PhysiCell$
```

Test build/run: PhysiCell model (2)

~/PhysiCell$ **make**   # from this directory, just run ‘make’ (or **make -j 2** to use 2 cores and speed up compilation)

→ You will see the following output:

```
make heterogeneity-sample
cp ./sample_projects/heterogeneity/custom_modules/* ./custom_modules/
touch main.cpp && cp main.cpp main-backup.cpp
cp ./sample_projects/heterogeneity/main-heterogeneity.cpp ./main.cpp
cp Makefile Makefile-backup
cp ./sample_projects/heterogeneity/Makefile .
cp ./config/PhysiCell_settings.xml ./config/PhysiCell_settings-backup.xml
cp ./sample_projects/heterogeneity/config/* ./config/
make
g++-13 -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./BioFVM/BioFVM_vector.cpp
    ... (continues to compile files)...
g++-13 -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -o heterogeneity BioFVM_vector.o BioFVM_mesh.o
BioFVM_microenvironment.o BioFVM_solvers.o BioFVM_matlab.o BioFVM_utilities.o BioFVM_basic_agent.o BioFVM_MultiCellDS.o
BioFVM_agent_container.o   pugixml.o PhysiCell_phenotype.o PhysiCell_cell_container.o PhysiCell_standard_models.o PhysiCell_cell.o
PhysiCell_custom.o PhysiCell_utilities.o PhysiCell_constants.o PhysiCell_basic_signaling.o  PhysiCell_SVG.o PhysiCell_pathology.o
PhysiCell_MultiCellDS.o PhysiCell_various_outputs.o PhysiCell_pugixml.o PhysiCell_settings.o PhysiCell_geometry.o heterogeneity.o main.cpp

Executable name is heterogeneity

~/PhysiCell$
```

Test build/run: PhysiCell model (3)

~/PhysiCell$ **./heterogeneity**   # run the model

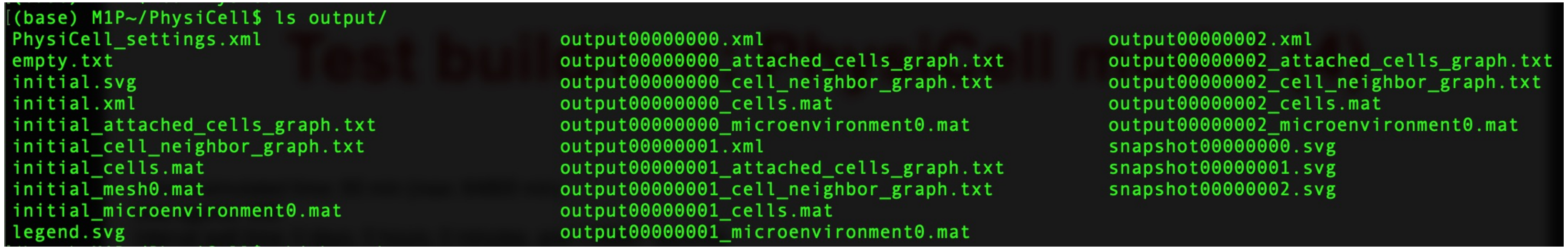
...   → lots of model configuration info will be printed out, and then info at each specified output interval:

```
Oncoprotein summary:
===================
mean: 1.00687
standard deviation: 0.250737
[min max]: [0.205535 1.71906]

Using PhysiCell version 1.12.0
        Please cite DOI: 10.1371/journal.pcbi.1005991
        Project website: http://PhysiCell.MathCancer.org

See ALL_CITATIONS.txt for this list.
current simulated time: 0 min (max: 64800 min)
total agents: 890
interval wall time: 0 days, 0 hours, 0 minutes, and 2.1e-05 seconds
total wall time: 0 days, 0 hours, 0 minutes, and 2.4e-05 seconds

Using method diffusion_decay_solver__constant_coefficients_LOD_2D (2D LOD with Thomas Algorithm) ...

current simulated time: 60 min (max: 64800 min)
total agents: 896
interval wall time: 0 days, 0 hours, 0 minutes, and 1.89867 seconds
total wall time: 0 days, 0 hours, 0 minutes, and 1.8987 seconds
```

Test build/run: PhysiCell model (4)

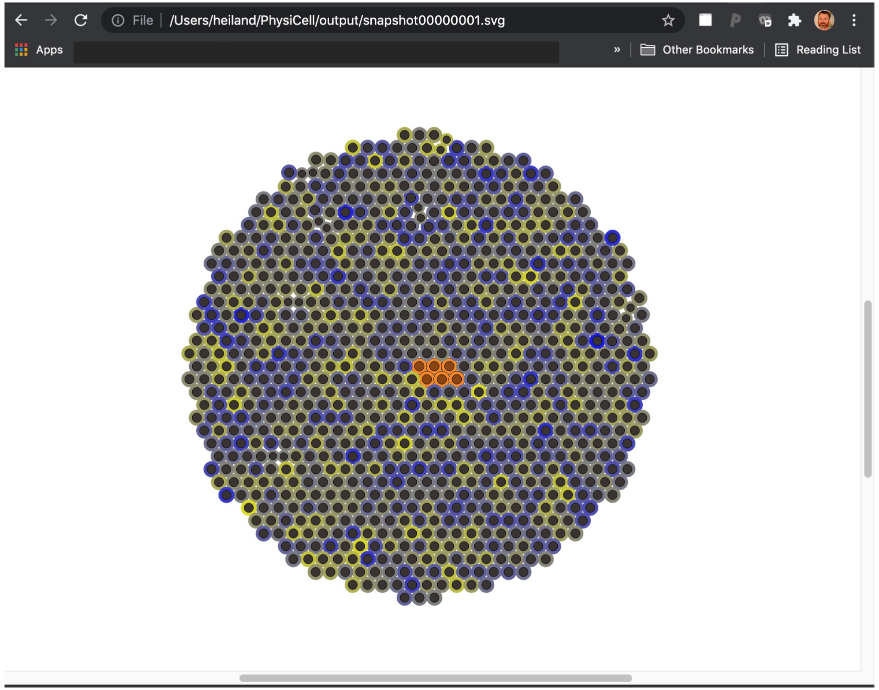
```
current simulated time: 60 min (max: 64800 min)
total agents: 896
interval wall time: 0 days, 0 hours, 0 minutes, and 1.89867 seconds
total wall time: 0 days, 0 hours, 0 minutes, and 1.8987 seconds
```

You can press “control-c” to cancel the simulation and then type: **ls output** # list files created in the /output directory



```
(base) M1P~/PhysiCell$ ls output/
PhysiCell_settings.xml                 output00000000.xml                          output00000002.xml
empty.txt                              output00000000_attached_cells_graph.txt     output00000002_attached_cells_graph.txt
initial.svg                            output00000000_cell_neighbor_graph.txt      output00000002_cell_neighbor_graph.txt
initial.xml                            output00000000_cells.mat                    output00000002_cells.mat
initial_attached_cells_graph.txt       output00000000_microenvironment0.mat        output00000002_microenvironment0.mat
initial_cell_neighbor_graph.txt        output00000001.xml                          snapshot00000000.svg
initial_cells.mat                      output00000001_attached_cells_graph.txt     snapshot00000001.svg
initial_mesh0.mat                      output00000001_cell_neighbor_graph.txt      snapshot00000002.svg
initial_microenvironment0.mat          output00000001_cells.mat
legend.svg                             output00000001_microenvironment0.mat
```

Test build/run: PhysiCell model (5)



To easily visualize the cells at a particular output interval, you can simply open one of the .svg files in your Web browser. Beware that it will be rather large, but you can use the scrollbars to find the heterogeneous tumor of cells at the center of the domain.

Overview

- OpenMP-enabled g++ (using Homebrew)
- Test building the default model (“heterogeneity”)
- Python 3 (using Anaconda distribution)
- Test building an intracellular model
- ImageMagick
- PhysiCell Studio
- pcDataLoader
- COPASI
- C++ code editor
- Git and GitHub

Python

Python is a requirement for certain aspects of PhysiCell:

- It is used for PhysiCell Studio.
- It is needed to install certain libraries for the intracellular models.
- It can be used for visualization and data analysis scripts.
- It can be used for parameter explorations of models.

Python 3 (not Python 2)

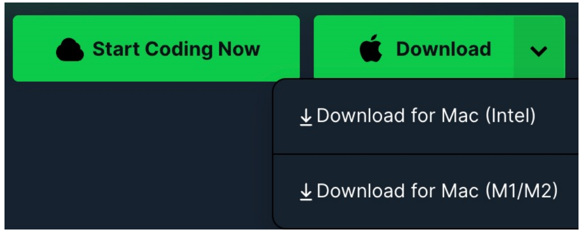
- Note that your Mac may have Python 2 installed, by default:

```
~$ which python
/usr/bin/python

~$ python
Python 2.7.16 (default, Jan 27 2020, 04:46:15)
[GCC 4.2.1 Compatible Apple LLVM 10.0.1 (clang-1001.0.37.14)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- We want Python 3. (And even if the newer Macs come bundled with Python 3, it will not contain all the modules that we want).

Anaconda Python 3.x

- Download a (free) Python distribution that comes bundled with lots of useful modules that are not in the standard Python library.
- https://www.anaconda.com/products/individual#Downloads
- https://docs.anaconda.com/anaconda/install/mac-os/

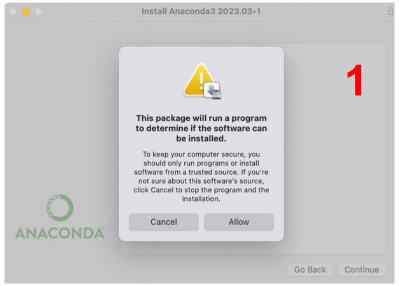


Download the appropriate one - Intel or M1/M2(Silicon) for your Mac. If you're unsure, clicking on the Apple logo in your menu bar and "About This Mac" will tell you which "Chip" it has.

Anaconda Graphical Installer

```
~$ open Anaconda3-2023.03-1-MacOSX-arm64.pkg
```

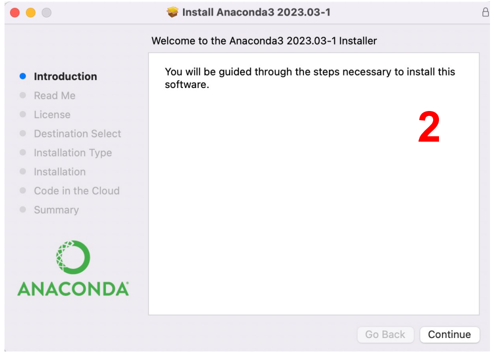
Your installation should resemble these screenshots:


Install Anaconda3 2023.03-1

This package will run a program to determine if the software can be installed.

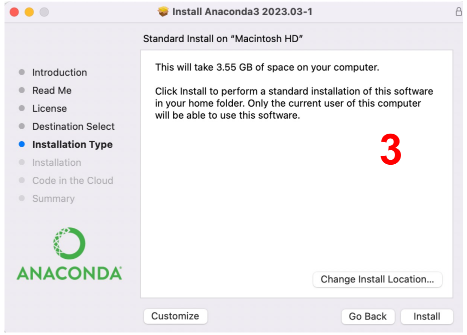
To keep your computer secure, you should only run programs or install software from a trusted source. If you're not sure about this software's source, click Cancel to stop the program and the installation.

Click 'Allow'


Install Anaconda3 2023.03-1

Welcome to the Anaconda3 2023.03-1 Installer

You will be guided through the steps necessary to install this software.

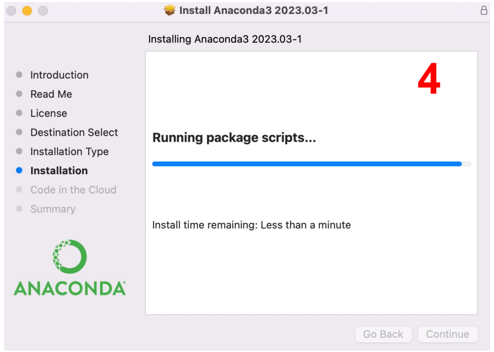

Install Anaconda3 2023.03-1

Standard Install on "Macintosh HD"

This will take 3.55 GB of space on your computer.

Click Install to perform a standard installation of this software in your home folder. Only the current user of this computer will be able to use this software.

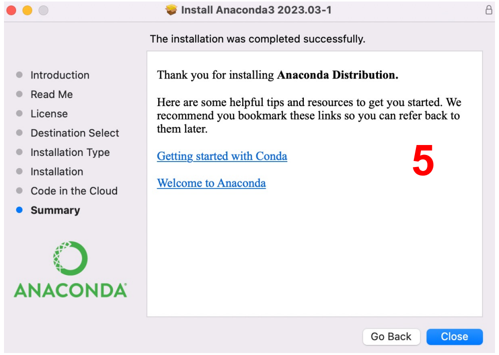
Click 'Continue', 'Install', agree to their license, …, 'Close'.


Install Anaconda3 2023.03-1

Installing Anaconda3 2023.03-1

Running package scripts...

Install time remaining: Less than a minute


Install Anaconda3 2023.03-1

The installation was completed successfully.

Thank you for installing **Anaconda Distribution.**

Here are some helpful tips and resources to get you started. We recommend you bookmark these links so you can refer back to them later.

Getting started with Conda

Welcome to Anaconda

Confirm Installation

Open a New Terminal shell to verify you are using the Anaconda Python as your default:

```
~$ which python
/Users/heiland/anaconda3/bin/python

~$ python
Python 3.10.9 (main, Mar  1 2023, 12:20:14) [Clang 14.0.6 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Overview

- OpenMP-enabled g++ (using Homebrew)
- Test building the default model (“heterogeneity”)
- Python 3 (using Anaconda distribution)
- Test building an intracellular model
- ImageMagick
- PhysiCell Studio
- pcDataLoader
- COPASI
- C++ code editor
- Git and GitHub

Advanced topic: Intracellular sample project

```
~/PhysiCell$ make reset

rm -f *.cpp
cp ./sample_projects/Makefile-default Makefile
rm -f ./custom_modules/*
touch ./custom_modules/empty.txt
touch ALL_CITATIONS.txt
touch ./core/PhysiCell_cell.cpp
rm ALL_CITATIONS.txt
cp ./config/PhysiCell_settings-backup.xml ./config/PhysiCell_settings.xml
touch ./config/empty.csv
rm -f ./config/*.csv

~/PhysiCell$ make list-projects
Sample projects: template biorobots-sample cancer-biorobots-sample cancer-immune-sample
        celltypes3-sample heterogeneity-sample pred-prey-farmer virus-macrophage-sample
        worm-sample interaction-sample mechano-sample rules-sample

Sample intracellular projects: ode-energy-sample physiboss-cell-lines-sample cancer-metabolism-sample

~/PhysiCell$
```

ODE intracellular model (1)

```
~/PhysiCell$ make ode-energy-sample

cp ./sample_projects_intracellular/ode/ode_energy/custom_modules/* ./custom_modules/
touch main.cpp && cp main.cpp main-backup.cpp
cp ./sample_projects_intracellular/ode/ode_energy/main.cpp ./main.cpp
cp Makefile Makefile-backup
cp ./sample_projects_intracellular/ode/ode_energy/Makefile .
cp ./config/PhysiCell_settings.xml ./config/PhysiCell_settings-backup.xml
cp ./sample_projects_intracellular/ode/ode_energy/config/* ./config/

~/PhysiCell$ make
python3 beta/setup_libroadrunner.py


This model requires the libRoadrunner libraries which will now be downloaded.
(for your  Darwin  operating system)
libRoadRunner will now be installed into this location:
addons/libRoadrunner

Beginning download of libroadrunner into addons/libRoadrunner ...
https://sourceforge.net/projects/libroadrunner/files/libroadrunner-1.4.18/roadrunner-osx-10.9-cp36m.tar.gz/download
my_file =  addons/libRoadrunner/roadrunner-osx-10.9-cp36m.tar.gz
rrlib_dir =  addons/libRoadrunner/roadrunner-osx-10.9-cp36m
100.0% 96092160 / 96087190
installing (uncompressing) the file...
Done.
```

$\}$ The ODE solver library is downloaded

Or, for M1/M2:
https://github.com/PhysiCell-Tools/intracellular_libs/raw/main/ode/roadrunner_macos_arm64.tar.gz

ODE intracellular model (2)

(from previous 'make')

```
g++-13 -march=native  -O3 -fomit-frame-pointer -fopenmp -m64 -std=c++11 -D ADDON_ROADRUNNER -I./addons/libRoadrunner/roadrunner/include/rr/C  -c
./core/PhysiCell_cell.cpp
g++-13 -march=native  -O3 -fomit-frame-pointer -fopenmp -m64 -std=c++11 -D ADDON_ROADRUNNER -I./addons/libRoadrunner/roadrunner/include/rr/C  -c
./custom_modules/custom.cpp
g++-13 -march=native  -O3 -fomit-frame-pointer -fopenmp -m64 -std=c++11 -D ADDON_ROADRUNNER -I./addons/libRoadrunner/roadrunner/include/rr/C  -c
./addons/libRoadrunner/src/librr_intracellular.cpp
Your OS= -D OSX
LIBRR_CFLAGS= -I./addons/libRoadrunner/roadrunner/include/rr/C
LIBRR_LIBS= ./addons/libRoadrunner/roadrunner/lib

g++-13 -march=native  -O3 -fomit-frame-pointer -fopenmp -m64 -std=c++11 -D ADDON_ROADRUNNER -I./addons/libRoadrunner/roadrunner/include/rr/C   -o ode_energy
BioFVM_vector.o BioFVM_mesh.o BioFVM_microenvironment.o BioFVM_solvers.o BioFVM_matlab.o BioFVM_utilities.o BioFVM_basic_agent.o BioFVM_MultiCellDS.o
BioFVM_agent_container.o   pugixml.o PhysiCell_phenotype.o PhysiCell_cell_container.o PhysiCell_standard_models.o PhysiCell_cell.o PhysiCell_custom.o
PhysiCell_utilities.o PhysiCell_constants.o  PhysiCell_SVG.o PhysiCell_pathology.o PhysiCell_MultiCellDS.o PhysiCell_various_outputs.o PhysiCell_pugixml.o
PhysiCell_settings.o custom.o  librr_intracellular.o  main.cpp -L./addons/libRoadrunner/roadrunner/lib -lroadrunner_c_api

created ode_energy

~/PhysiCell$ ./ode_energy

dyld: Library not loaded: @rpath/libroadrunner_c_api.dylib
  Referenced from: /Users/heiland/PhysiCell/./ode_energy
  Reason: image not found
Abort trap: 6
~/PhysiCell$
```

You will see this output (but not highlighted …) if the executable was successfully made

When we try to run the model, we get an error, but it was expected and serves as a reminder if/when you ever see it again. See next slide.

ODE intracellular model (3)

```
~/PhysiCell$ export DYLD_LIBRARY_PATH=$DYLD_LIBRARY_PATH:./addons/libRoadrunner/roadrunner/lib

~/PhysiCell$ ./ode_energy

        ... model info output...
current simulated time: 30 min (max: 1440 min)
total agents: 144
interval wall time: 0 days, 0 hours, 0 minutes, and 4.27858 seconds
total wall time: 0 days, 0 hours, 0 minutes, and 4.27861 seconds

current simulated time: 60 min (max: 1440 min)
total agents: 144
interval wall time: 0 days, 0 hours, 0 minutes, and 4.33063 seconds
total wall time: 0 days, 0 hours, 0 minutes, and 8.60924 seconds

------------ start: librr_intracellular.cpp: start() called

... (lots more output)...
```

To avoid the previous runtime error, define another environment variable. The model should then run OK.

And once again, you could use your browser to open one of the .svg files that are created in `/output`

ODE intracellular model (4)

As before, permanently put this environment variable in your (bash or zsh) shell's config startup file:

```
export DYLD_LIBRARY_PATH=$DYLD_LIBRARY_PATH:./addons/libRoadrunner/roadrunner/lib >> ~/.bash_profile
```

or,

```
export DYLD_LIBRARY_PATH=$DYLD_LIBRARY_PATH:./addons/libRoadrunner/roadrunner/lib >> ~/.zshenv
```

Then when you start a *new* Terminal Shell window, this environment variable will be defined.

Overview

- OpenMP-enabled g++ (using Homebrew)
- Test building the default model ("heterogeneity")
- Python 3 (using Anaconda distribution)
- Test building an intracellular model
- ImageMagick
- PhysiCell Studio
- pcDataLoader
- COPASI
- C++ code editor
- Git and GitHub

ImageMagick (1)

https://imagemagick.org/ - free, powerful image conversion, composition, editing software.

```
$ brew install imagemagick
```

(probably will install lots of dependencies)

```
...
==> Installing imagemagick dependency: openexr
...
```

You should then have access to various ImageMagick commands, for example:

```
$ which convert
/opt/homebrew/bin/convert
Or, depending on your system, it may have this path (or other):
/usr/local/bin/convert
```

ImageMagick (2)

Refer to the Quickstart guide for helpful ImageMagick commands, (*note – this is deprecated but may be helpful)* including Makefile targets:

As an example, if you have generated some .svg files (in /output), you should be able to generate an animation, using something like the following set of commands in your shell:

```
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
```

Overview

- OpenMP-enabled g++ (using Homebrew)
- Test building the default model (“heterogeneity”)
- Python 3 (using Anaconda distribution)
- Test building an intracellular model
- ImageMagick
- PhysiCell Studio
- pcDataLoader
- COPASI
- C++ code editor
- Git and GitHub

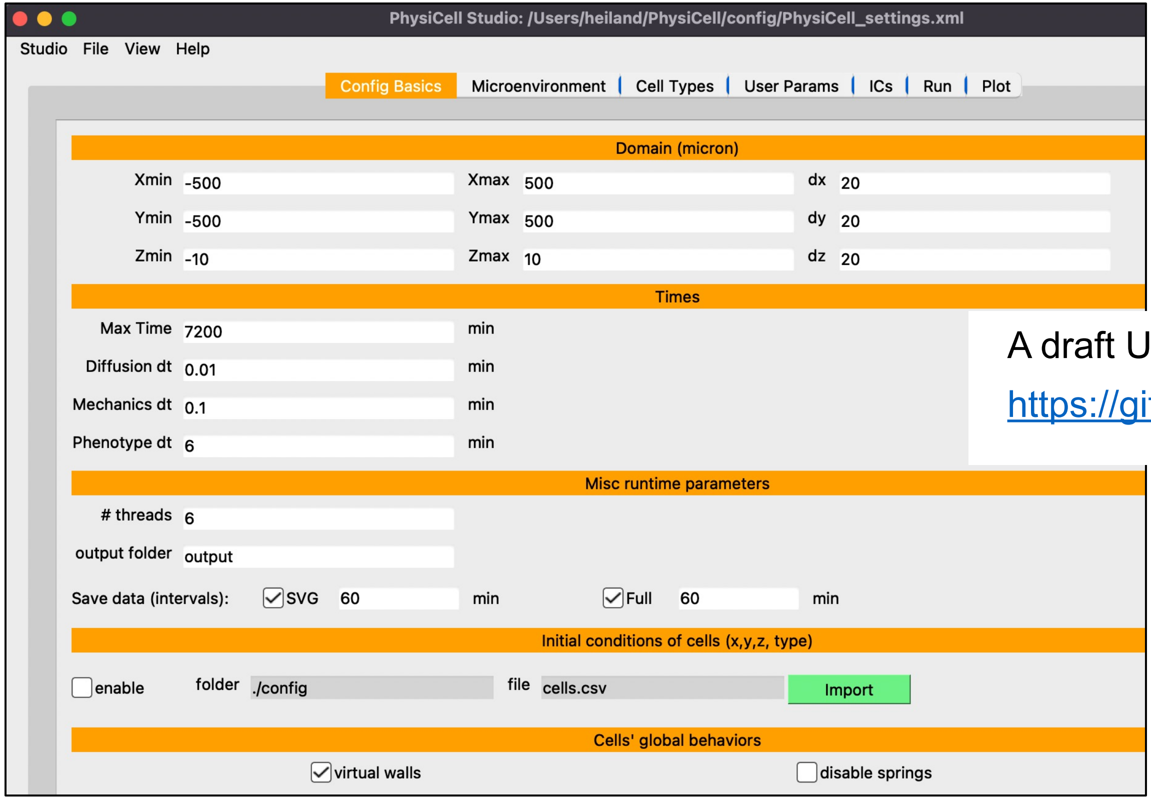
PhysiCell Studio (1)

- PhysiCell Studio is a desktop graphical tool to let you create/edit a .xml configuration file that defines a PhysiCell model. The Studio also lets you run a simulation, plot results, and more.
- Download the latest release at:
  https://github.com/PhysiCell-Tools/PhysiCell-Studio/releases
- Copy or move the .zip to the same level as your PhysiCell directory, unzip it, and run the Studio, e.g.:

```
~$ unzip PhysiCell-Studio-2.26.7.zip
~$ python ~/PhysiCell-Studio-2.26.7/bin/studio.py
```

This should display the Studio (next page):

PhysiCell Studio (2)


PhysiCell Studio: /Users/heiland/PhysiCell/config/PhysiCell_settings.xml

Studio File View Help
Config Basics | Microenvironment | Cell Types | User Params | ICs | Run | Plot

Domain (micron)

Xmin	-500	Xmax	500	dx	20
Ymin	-500	Ymax	500	dy	20
Zmin	-10	Zmax	10	dz	20

Times

Max Time	7200	min
Diffusion dt	0.01	min
Mechanics dt	0.1	min
Phenotype dt	6	min

Misc runtime parameters

threads 6

output folder output

Save data (intervals): [✓] SVG 60 min [✓] Full 60 min

Initial conditions of cells (x,y,z, type)

[ ] enable folder ./config file cells.csv

Cells' global behaviors

[✓] virtual walls [ ] disable springs

A draft User Guide for the Studio is at

https://github.com/PhysiCell-Tools/Studio-Guide

Overview

- OpenMP-enabled g++ (using Homebrew)
- Test building the default model (“heterogeneity”)
- Python 3 (using Anaconda distribution)
- Test building an intracellular model
- ImageMagick
- PhysiCell Studio
- pcDataLoader
- COPASI
- C++ code editor
- Git and GitHub

pcDataLoader

- Used to load PhysiCell data
  - Python based data loader
  - Facilitates post-simulation analysis
- Additional information (tutorials, reference guide etc) and latest releases at: https://github.com/PhysiCell-Tools/python-loader
- From any running terminal:
  ~/PhysiCell$ **pip3 install pcdl**
- pcDataloader should now be available for import at any python prompt

Overview

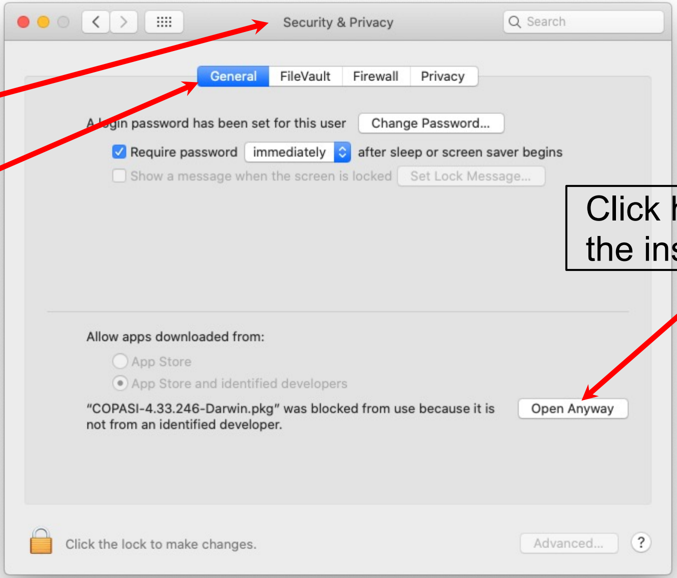
- OpenMP-enabled g++ (using Homebrew)
- Test building the default model (“heterogeneity”)
- Python 3 (using Anaconda distribution)
- Test building an intracellular model
- ImageMagick
- PhysiCell Studio
- pcDataLoader
- COPASI
- C++ code editor
- Git and GitHub

SBML editors: COPASI (1)

- COPASI, SBML, and SBML editors:
  - COPASI can simulate some categories of mathematical models (ordinary and stochastic differential equations) among other features
  - COPASI Provides a graphical interface for editing Systems Biology Markup Language (SBML)
    - SBML is a language used to encode biological models, often intracellular models
      - The ode-sample-model is written in SBML as is the FBA example
  - Note: there are other SBML editors (search "SBML editors" for other options)
- For the PhysiCell workshop, SBML model creation and editing will be demonstrated with COPASI

SBML editors: COPASI (2)

- Navigate to http://copasi.org/Download/ and download COPASI for Mac OS X
- Follow instructions here: http://copasi.org/Support/Installation/Mac_OS_X/
- Once downloaded and activated, you may be challenged by Apple security as this app isn't from the App Store
  - To get around this:
    - ♦ Go to **Settings → Security & Privacy,** then the **General** tab in **Security & Privacy**
    - ♦ If you recently ran the package file, there will be something like " "COPASI" was blocked from use because it is not from an identified developer." and an option "Open Anyway" → **Click** "Open Anyway"
- Complete installation via the COPASI Installer
  - It should be fine to accept the defaults…



A login password has been set for this user

[✓] Require password immediately after sleep or screen saver begins

[ ] Show a message when the screen is locked

Allow apps downloaded from:

( ) App Store

(●) App Store and identified developers

"COPASI-4.33.246-Darwin.pkg" was blocked from use because it is not from an identified developer.

Click the lock to make changes.

Click here to allow the installer to run

Overview

- OpenMP-enabled g++ (using Homebrew)
- Test building the default model (“heterogeneity”)
- Python 3 (using Anaconda distribution)
- Test building an intracellular model
- ImageMagick
- PhysiCell Studio
- pcDataLoader
- COPASI
- C++ code editor
- Git and GitHub

C++ Code editor

When you get to the point of editing the custom C++ code for your model, you will want a decent code editor. If you're already using one (for C or C++), great! - keep using it. But if you are new to programming, we recommend keeping it pretty simple. If you just search "C++ code editor macOS", you'll find some good suggestions.

One popular, free integrated development environment (IDE) that can be used in a minimal fashion for editing is VSCode (https://code.visualstudio.com/).

Version control

When you get to the point of editing the custom C++ code, python scripts for analysis, etc, it is common to use version control for your code and to share with collaborators. If you are already using version control great! - keep using it. If you are new to programming, we recommend using git. A search for “git mac install” will yield helpful results. (You may even already have it – it is included in XCode command line tools).

Once you have git, github.com is a common place to share code. There are also many graphical interfaces for git (GitHub has one for example.)

Support

- We encourage you to join and actively use the PhysiCell community Slack channel. There, you can post questions (#troubleshooting), answer questions, and (hopefully) share successful modeling stories.
- Alternatively, you can submit problem tickets at https://sourceforge.net/p/physicell/tickets/
- Finally, please follow us on Twitter @PhysiCell and @MathCancer.

Funding Acknowledgements

PhysiCell Development:

- Breast Cancer Research Foundation
- Jayne Koskinas Ted Giovanis Foundation for Health and Policy
- National Cancer Institute (U01CA232137)
- National Science Foundation (1720625, 1818187)

Training Materials:

- Administrative supplement to NCI U01CA232137 (Year 2)

Other Funding:

- NCI / DOE / Frederick National Lab for Cancer Research (21X126F)
- DOD / Defense Threat Reduction Agency (HDTRA12110015)
- NIH Common Fund (3OT2OD026671-01S4)