

Slides, videos, links and more:

<https://github.com/physicell-training/ws2021>

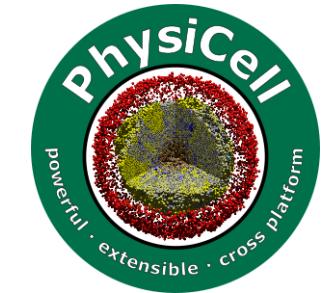
Preparing your PhysiCell model to share on nanoHUB (or, just run the Jupyter notebook GUI locally)



Randy Heiland
 [@rheiland](https://twitter.com/rheiland) [@PhysiCell](https://twitter.com/PhysiCell)

PhysiCell Project

July 25-31, 2021



Overview

- nanoHUB (<https://nanohub.org/>) is an open and free platform for computational research, education, and collaboration in nanotechnology, materials science, and related fields.
- nanoHUB lets users run interactive apps from a browser.
- This tutorial will explain how you port an existing PhysiCell (2-D) model to run on nanoHUB.

Some Limitations (currently)

- Only 2-D models, not 3-D (but customization is possible, with our help)
- Intracellular models are not supported on nanoHUB
 - We need to work with nanoHUB to pre-install the necessary libraries
- The “Cell Types” tab will represent the actual cell type hierarchy found in your PhysiCell_settings.xml, e.g.,

```
<cell_definition name="A" ID="1" parent_type="default">
```

(this could be considered a limitation because it doesn't expose every cell_definition parameter for every cell type)

Assumptions

- You have installed the Anaconda Python distribution:
 - <https://github.com/MathCancer/PhysiCell/blob/master/documentation/Quickstart.md#python>
 - If you need more setup details: <https://github.com/physicell-training/ws2021#pre-workshop-materials>
- You have a working PhysiCell 2-D model that adheres to the default directory structure, file naming scheme (`Makefile`, `main.cpp`, `config/PhysiCell_settings.xml`), and contains the `output/initial.xml` file from a simulation.
- You have downloaded and installed (unzipped) the latest release from <https://github.com/PhysiCell-Tools/PhysiCell-Jupyter-GUI/releases>

Detailed instructions

- Follow the steps here:
 - <https://github.com/PhysiCell-Tools/PhysiCell-Jupyter-GUI/blob/master/README.md>
 - <https://github.com/PhysiCell-Tools/PhysiCell-Jupyter-GUI/blob/master/README.md#steps-to-follow>
 - ◆ Instead of using “ise_proj1”, you may want to use: “pc4ws21teamN” (PhysiCell for Workshop 2021, where N=1,2,etc (whatever your team # is)) – see next slide
 - These instructions should let you create a Jupyter notebook that you can run locally (on your computer) before installing it on nanoHUB.
 - Creating a nanoHUB app is considered optional (<https://github.com/PhysiCell-Tools/PhysiCell-Jupyter-GUI/blob/master/README.md#create-a-nanohub-tool-optional>), but we are assuming you will do it in this tutorial. Let us know if you need help.

The following slides try to illustrate these steps.

Create a new GitHub repo for your nanoHUB project. E.g., I created “pc4ws21team42”

The repo needs to be “public” for nanoHUB.

For a team project, one person can create it on their personal account and add others as collaborators.

github.com/new

Imported IPy IUCAT sovf vtk-py gists PCUsers Other Boo

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner * / Repository name *

Great repository names are short and memorable. Need inspiration? How about [jubilant-octo-dollop](#)?

Description (optional)

 Public Anyone on the internet can see this repository. You choose who can commit.

 Private You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you’re importing an existing repository.

Add a README file This is where you can write a long description for your project. [Learn more](#).

Add .gitignore Choose which files not to track from a list of templates. [Learn more](#).

Choose a license A license tells others what they can and can’t do with your code. [Learn more](#).

This will set  main as the default branch. Change the default name in your [settings](#).

[Create repository](#)

Using celltypes3 as an example (1)

```
~$ unzip PhysiCell_V.1.9.0.zip
```

```
...  
~$ cd PhysiCell
```

```
~/PhysiCell$ make celltypes3-sample  
... (copies over this sample's files) ...
```

```
~/PhysiCell$ make
```

```
...  
g++-11 -march=native -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11 -o celltypes3  
BioFVM_vector.o BioFVM_mesh.o BioFVM_microenvironment.o BioFVM_solvers.o BioFVM_matlab.o  
BioFVM_utilities.o BioFVM_basic_agent.o BioFVM_MultiCellIDS.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_MultiCellIDS.o PhysiCell_various_outputs.o PhysiCell_pugixml.o  
PhysiCell_settings.o PhysiCell_geometry.o custom.o main.cpp
```

```
~/PhysiCell$ ./celltypes3
```

```
... Let it run just for a very brief time (to generate the /output/initial.xml) and then ctl-c to kill it
```

Note: we use MacOS/Linux commands in these slides. In a Windows shell, your syntax will likely vary, e.g., “\” instead of “/”, using “.exe” suffix on a model’s executable, “dir” instead of “ls”, etc.

Using celltypes3 as an example (2)

- Verify you have these 4 critical files:

```
~/PhysiCell$ ls Makefile main.cpp config/PhysiCell_settings.xml output/initial.xml
Makefile                               main.cpp
config/PhysiCell_settings.xml          output/initial.xml
```

Clone your GitHub repo

```
~$ git clone git@github.com:rheiland/pc4ws21team42.git  
Cloning into 'pc4ws21team42'...  
remote: Enumerating objects: 3, done.  
remote: Counting objects: 100% (3/3), done.  
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0  
Receiving objects: 100% (3/3), done.  
~$
```

```
~/PhysiCell-Jupyter-GUI-1.4$ python setup_new_proj.py ~/pc4ws21team42 ~/PhysiCell pc4ws21team42  
...
```

This will generate these files in your cloned nanoHUB project:

```
~/pc4ws21team42$ ls  
LICENSE examples/ rappture/  
README.md make_my_tool.py src/  
bin/ middleware/ tmpdir/  
data/ mod_makefile.py  
doc/ pc4ws21team42.ipynb ← This will be your project's name
```

Rf. <https://github.com/PhysiCell-Tools/PhysiCell-Jupyter-GUI/blob/master/README.md#steps-to-follow> for options and for Windows command syntax.



Note: if you need to re-run the `setup_new_proj.py` script for any reason, it is probably best to first delete all the files & directories that were previously created. HOWEVER, if you made edits to your `doc/about.html`, make a backup copy!

FYI: `setup_new_proj.py`
is an extension of an earlier
project: `xml2jupyter`

<https://doi.org/10.21105/joss.01408>



The Journal of Open Source Software

xml2jupyter: Mapping parameters between XML and Jupyter widgets

Randy Heiland¹, Daniel Mishler¹, Tyler Zhang¹, Eric Bower¹, and Paul Macklin¹

¹ Intelligent Systems Engineering, Indiana University

DOI: [10.21105/joss.01408](https://doi.org/10.21105/joss.01408)

Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Submitted: 08 April 2019
Published: 01 July 2019

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC-BY](#)).

Summary

Jupyter Notebooks (Kluyver et al., 2016, Perkel (2018)) provide executable documents (in a variety of programming languages) that can be run in a web browser. When a notebook contains graphical widgets, it becomes an easy-to-use graphical user interface (GUI). Many scientific simulation packages use text-based configuration files to provide parameter values and run at the command line without a graphical interface. Manually editing these files to explore how different values affect a simulation can be burdensome for technical users, and impossible to use for those with other scientific backgrounds. `xml2jupyter` is a Python package that addresses these scientific bottlenecks. It provides a mapping between configuration files, formatted in the Extensible Markup Language (XML), and Jupyter widgets. Widgets are automatically generated from the XML file and these can, optionally, be incorporated into a larger GUI for a simulation package, and optionally hosted on cloud resources. Users modify parameter values via the widgets, and the values are written to the XML configuration file which is input to the simulation's command-line interface. `xml2jupyter` has been tested using PhysiCell (Ghaffarizadeh, Heiland, Friedman, Mumenthaler, & Macklin, 2018), an open source, agent-based simulator for biology, and it is being used by students for classroom and research projects. In addition, we use `xml2jupyter` to help create Jupyter GUIs for PhysiCell-related applications running on nanoHUB (Madhavan et al., 2013).

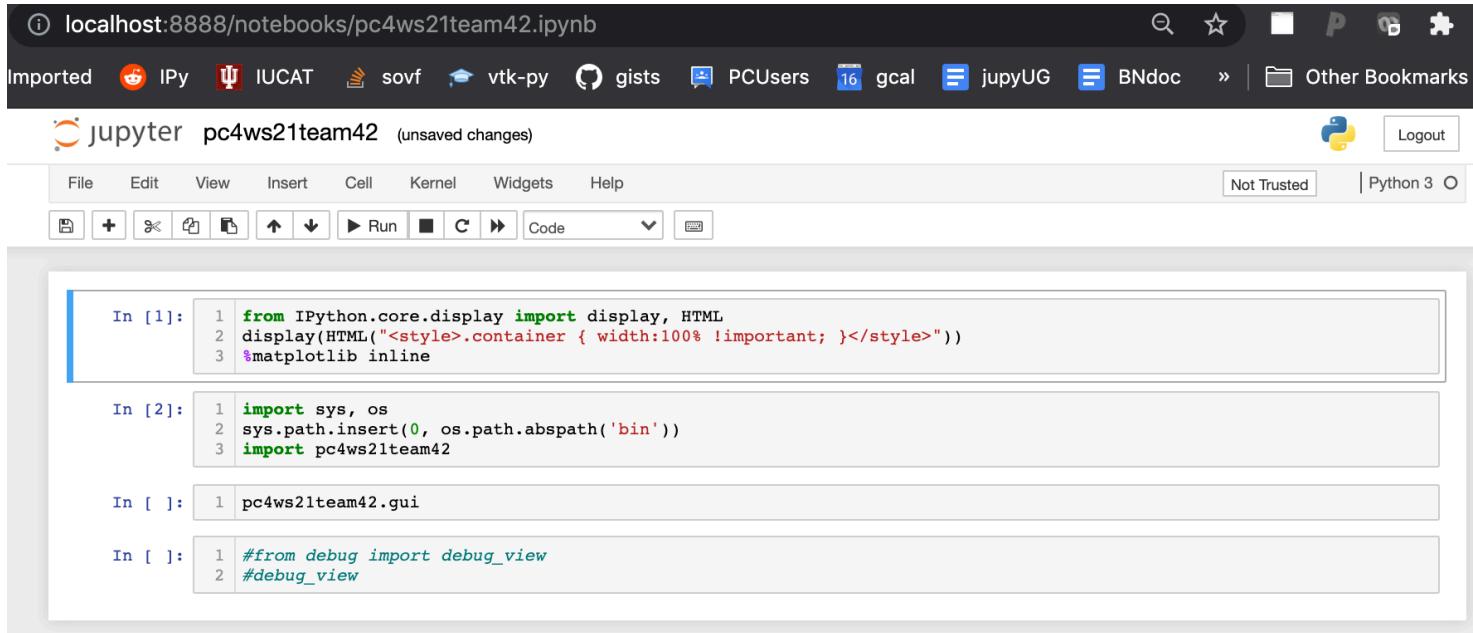
A PhysiCell configuration file defines model-specific `<user_parameters>` in XML. Each parameter element consists of its name with attributes, defining its data type, units (optional),

Test your notebook on your computer

Have a Web browser running, then run the following command from a shell window:

```
~/pc4ws21team42$ jupyter notebook pc4ws21team42.ipynb
```

...



The screenshot shows a Jupyter Notebook interface running in a web browser. The title bar indicates the URL is `localhost:8888/notebooks/pc4ws21team42.ipynb`. The browser's address bar also shows `Imported`, `IPy`, `IUCAT`, `sofv`, `vtk-py`, `gists`, `PCUsers`, `gcal`, `jupyUG`, `BNdoc`, and `Other Bookmarks`. The main window displays a Jupyter logo and the title "jupyter pc4ws21team42 (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and various icons for file operations like Open, Save, and Run. A status bar at the bottom right shows "Not Trusted" and "Python 3". The notebook content consists of several code cells:

- In [1]:**

```
1 from IPython.core.display import display, HTML
2 display(HTML("<style>.container { width:100% !important; }</style>"))
3 %matplotlib inline
```
- In [2]:**

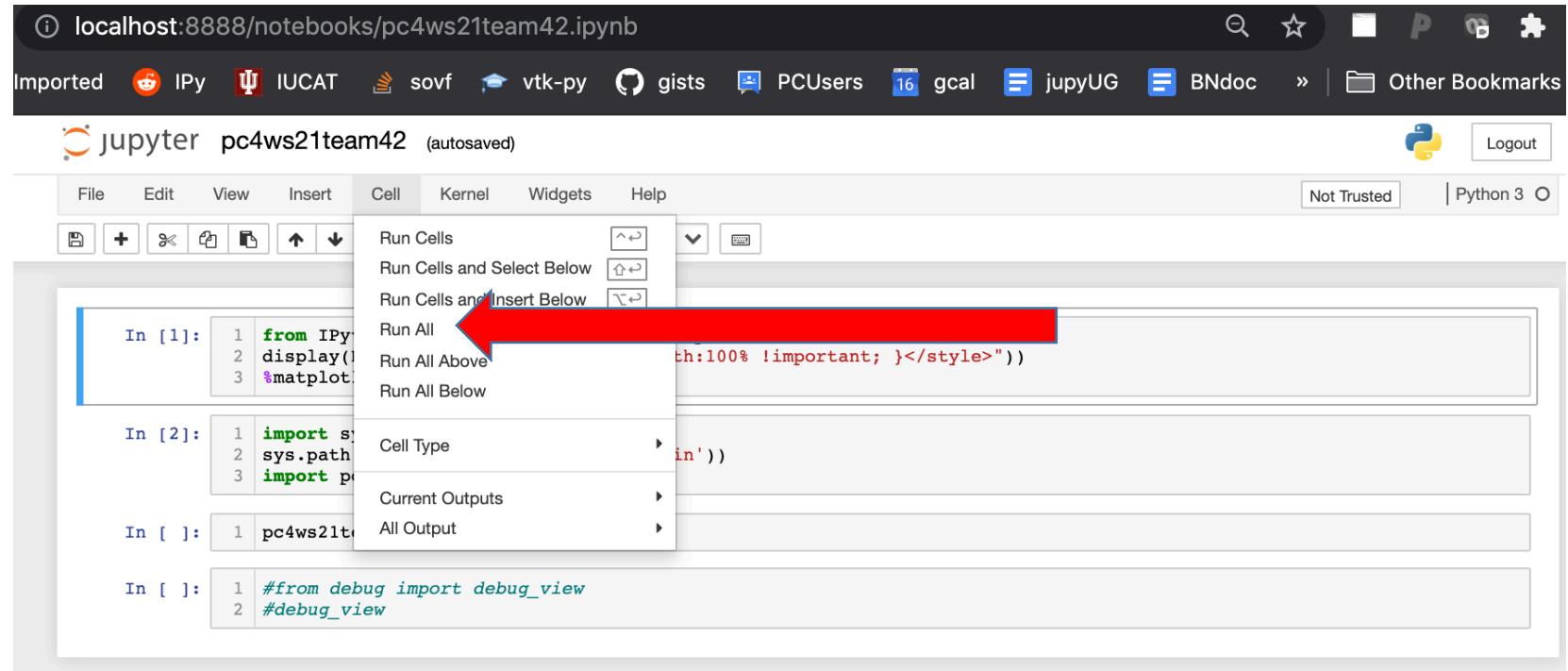
```
1 import sys, os
2 sys.path.insert(0, os.path.abspath('bin'))
3 import pc4ws21team42
```
- In []:**

```
1 pc4ws21team42.gui
```
- In []:**

```
1 #from debug import debug_view
2 #debug_view
```

You should see this
Jupyter notebook
displayed in your
browser:

Click the ‘Cell’ menu, ‘Run All’, wait for each notebook “cell” to execute...



You should see a similar looking notebook GUI for your project.

Now or later:
You can leave this running, then edit the doc/about.html file to update this page's contents and (see next page).

The screenshot shows a Jupyter Notebook interface with three code cells and a generated GUI overview page.

In [1]:

```
1 from IPython.core.display import display, HTML
2 display(HTML("<style>.container { width:100% !important; }</style>"))
3 %matplotlib inline
```

In [2]:

```
1 import sys, os
2 sys.path.insert(0, os.path.abspath('bin'))
3 import pc4ws21team42
```

In [3]:

```
1 pc4ws21team42.gui
```

Below the code cells, there is a configuration dropdown set to "pc4ws21team42".

The main content area displays a "GUI Overview" page with the following tabs:

- About
- Config Basics
- Microenvironment
- User Params
- Cell Types
- Out: Plots
- Animate

GUI Overview

- Config Basics tab:** input parameters common to all models (e.g., domain grid, simulation time, choice/frequency of outputs)
- Microenvironment tab:** microenvironment parameters that are model-specific
- User Params tab:** user parameters that are model-specific
- Cell Types tab:** parameters for cell types that are model-specific
- Out: Plots tab:** output display of cells and substrates
- Animate tab:** generate an animation of cells

Clicking the 'Run' button will use the specified parameters and start a simulation. When clicked, it creates an "Output" widget that can be clicked/expanded to reveal the progress (text) of the simulation. When the simulation generates output files, they can be visualized in the "Out: Plots" tab. The "# cell frames" will be dynamically updated as those output files are generated by the running simulation. When the "Run" button is clicked, it toggles to a "Cancel" button that will terminate (not pause) the simulation.

Introduction

This app demonstrates ...

This model and cloud-hosted demo are part of a course on computational multicellular systems biology created and taught by Dr. Paul Macklin in the Department of Intelligent Systems Engineering at Indiana University. It is also part of the education and outreach for the IU Engineered nanoBIO Node and the NCI-funded cancer systems biology grant U01CA232137. The models are built using PhysiCell: a C++ framework for multicellular systems biology [1].

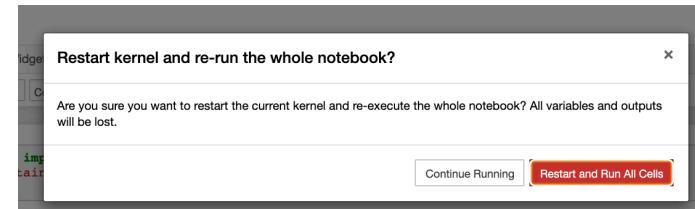
Basic instructions

After editing the “doc/about.html”, Click the ‘Kernel’ menu, ‘Restart & Run All’, then click the pop-up button to confirm:

The screenshot shows a Jupyter Notebook window titled "jupyter pc4ws21team42 (autosaved)". The "Kernel" menu is open, displaying options: Interrupt, Restart, Restart & Clear Output, **Restart & Run All**, Reconnect, and Shutdown. A red arrow points to the "Restart & Run All" option. Below the menu, code cells are visible:

```
In [1]: 1 from IPython.co  
2 display(HTML(""  
3 %matplotlib in  
  
In [2]: 1 import sys, os  
2 sys.path.insert(0, os.path.abspath('bin'))  
3 import pc4ws21team42  
  
In [3]: 1 pc4ws21team42.gui
```

At the bottom, there is a "Load Config" dropdown set to "DEFAULT" and a "pc4ws21team42" button. Below the dropdown are tabs: About (highlighted), Config Basics, Microenvironment, and User Params.



The notebook will refresh,
showing your edited text:

Clicking the 'Run' button will use the specified parameters and start a simulation. When cli simulation. When the simulation generates output files, they can be visualized in the "Out": simulation. When the "Run" button is clicked, it toggles to a "Cancel" button that will termin

Introduction
This app demonstrates my app!!! **← Edited text**
This model and cloud-hosted demo are part of a course on computational multicellular sys

Explore the other tabs of the GUI (but disregard the Animate tab for now)

The screenshot shows the PhysiCell software interface. At the top, there is a navigation bar with several tabs: About, Config Basics, Microenvironment, User Params, Cell Types, Out: Plots, and Animate. The 'Config Basics' tab is currently active, indicated by a blue background. Red circles highlight the 'Microenvironment', 'User Params', 'Cell Types', and 'Out: Plots' tabs. A large red 'X' is drawn over the 'Animate' tab. Below the tabs, there are various configuration settings:

- Domain (micron):**
 - Xmin: -1000
 - Xmax: 1000
 - dx: 20
 - Ymin: -1000
 - Ymax: 1000
 - dy: 20
- Max Time:** 2880 min
- # threads:** 4
- Plots:** Cells every 2 min Substrates every 2 min

A green 'Run' button is located at the bottom left.

Final steps for nanoHUB app

- If your local Jupyter notebook looks correct, you have the option of compiling your project and making sure you can Run it from the notebook - rf. Steps 4 and 5 in:
 - <https://github.com/PhysiCell-Tools/PhysiCell-Jupyter-GUI/blob/master/README.md#steps-to-follow>
- However, you can probably skip straight to getting it ready to test on nanoHUB:
 - [\(not “optional” if you plan to do it ☺\)](https://github.com/PhysiCell-Tools/PhysiCell-Jupyter-GUI/blob/master/README.md#create-a-nanohub-tool-optional)
 - ◆ <https://nanohub.org/register/> – register for a nanoHUB account if you haven’t
 - ◆ <https://nanohub.org/tools/create> – create your app’s basic info
 - ◆ Make sure your middleware/invoke script is “executable” (rf. github link above)

This is our previously published **pc3types** app on nanoHUB.

The screenshot shows the nanoHUB.org website with the URL nanohub.org/tools/pc3types. The page title is "Three-Type Multicellular Simulation Lab". Below the title, it says "By Paul Macklin, Randy Heiland". A summary text reads: "An analog of the 3-body problem: modeling 3 interacting cell types in a dynamic environment". To the right, there is a "Launch Tool" button and a sidebar with user statistics and sharing options. At the bottom, there is a navigation bar with tabs like About, Usage, Citations, Questions, Reviews, Wishlist, Versions, and Supporting Docs.

The screenshot shows the same page as above, focusing on the abstract and metadata. The abstract is titled "The 3-Type Problem" and states: "Inspired by the [3-body problem](#) in physics, this virtual cell laboratory explores a tunable multicellular system with 3 interacting cell types. Similarly to the 3-body problem, simple cell-cell interactions can lead to complex emergent dynamics, including". To the right, there is a "Watch resource" section and a logo for "nanoBIO ENGINEERED nanoBIO AN INDIANA UNIVERSITY RESEARCH NODE".

Abstract
The 3-Type Problem

Inspired by the [3-body problem](#) in physics, this virtual cell laboratory explores a tunable multicellular system with 3 interacting cell types. Similarly to the 3-body problem, simple cell-cell interactions can lead to complex emergent dynamics, including

Category: Tools
Published on: 04 Jan 2021

Watch resource
When watching a resource, you will be notified of changes made. You may stop watching at any time.

<https://nanohub.org/tools/pc3types>

Load Config **DEFAULT** ▾ **pc3types**

About Config Basics Microenvironment User Params Cell Types Out: Plots Animate

GUI Overview

- **Config Basics** tab: input parameters common to all models (e.g., domain grid, simulation time, choice/frequency of outputs)
- **Microenvironment** tab: microenvironment parameters that are model-specific
- **User Params** tab: user parameters that are model-specific
- **Cell Types** tab: parameters for cell types that are model-specific
- **Out: Plots** tab: output display of cells and substrates
- **Animate** tab: generate an animation of cells

Clicking the 'Run' button will use the specified parameters and start a simulation. When clicked, it creates an "Output" widget that can be clicked/expanded to reveal the progress (text) of the simulation. When the simulation generates output files, they can be visualized in the "Out: Plots" tab. The "# cell frames" will be dynamically updated as those output files are generated by the running simulation. When the "Run" button is clicked, it toggles to a "Cancel" button that will terminate (not pause) the simulation.

The 3-Type Problem

Inspired by the [3-body problem](#) in physics, this virtual cell laboratory explores a tunable multicellular system with 3 interacting cell types. Similarly to the 3-body problem, simple cell-cell

Run

Load Config [DEFAULT](#)[▼ pc3types](#)[About](#)[Config Basics](#)[Microenvironment](#)[User Params](#)[Cell Types](#)[Out: Plots](#)[Animate](#)

Domain (micron):

Xmin Xmax dx Ymin Ymax dy Max Time min# threads

Plots:

 Cellsevery min Substratesevery min[Run](#)

Load Config

About	Config Basics	Microenvironment	User Params	Cell Types	Out: Plots	Animate
resource (none)						
diffusion_coefficient	<input type="text" value="100000"/>	micron ² /min				
decay_rate	<input type="text" value="0"/>	1/min				
initial_condition	<input type="text" value="1"/>					
Dirichlet_boundary_condition	<input type="text" value="1"/>					<input checked="" type="checkbox"/> on/off
signal_A (none)						
diffusion_coefficient	<input type="text" value="1000"/>	micron ² /min				
decay_rate	<input type="text" value="0.1"/>	1/min				
initial_condition	<input type="text" value="0"/>					
Dirichlet_boundary_condition	<input type="text" value="0"/>					<input checked="" type="checkbox"/> on/off
signal_B (none)						
diffusion_coefficient	<input type="text" value="1000"/>	micron ² /min				
decay_rate	<input type="text" value="0.1"/>	1/min				
initial_condition	<input type="text" value="0"/>					
Dirichlet_boundary_condition	<input type="text" value="0"/>					<input checked="" type="checkbox"/> on/off
signal_C (none)						
diffusion_coefficient	<input type="text" value="1000"/>	micron ² /min				
decay_rate	<input type="text" value="0.1"/>	1/min				
initial_condition	<input type="text" value="0"/>					
Dirichlet_boundary_condition	<input type="text" value="0"/>					<input checked="" type="checkbox"/> on/off
<input checked="" type="checkbox"/> calculate_gradients <input checked="" type="checkbox"/> track_in_agents						
<input type="button" value="Run"/>						

Load Config **DEFAULT** **pc3types**

About	Config Basics	Microenvironment	User Params	Cell Types	Out: Plots	Animate
random_seed	0					
---Initialization settings---						
number_of_A	25				initial number of A	
number_of_B	25				initial number of B	
number_of_C	25				initial number of C	
max_distance_from_origin	150	micron			max initial distance of cells from (0,0,0)	
---Coloring settings---						
A_color	magenta				color for A cells	
B_color	green				color for B cells	
C_color	cyan				color for C cells	
standard_plots	<input type="checkbox"/>				Enable additional standard plots?	
---Overall signaling settings---						
hill_power	5				Hill power for signal responses	
half_max	0.1				half max for signal responses	
---cell type A settings---						
A_base_cycle	0.00072	1/min			base cell cycling rate for type A	
A_max_cycle	0.0072	1/min			maximum cell cycling rate for type A	
A_cycle_A	neutral				impact of signal A on A cycling (promote, inhibit, or neutral)	
A_cycle_B	neutral				impact of signal B on A cycling (promote, inhibit, or neutral)	
A_cycle_C	neutral				impact of signal C on A cycling (promote, inhibit, or neutral)	
A_cycle_pressure_threshold	2				pressure above threshold inhibits A cycling	
A_base_death	0.0000531667	1/min			base cell death rate for type A	
A_max_death	0.000531667	1/min			maximum cell death rate for type A	
A_death_A	neutral				impact of signal A on A apoptosis (promote, inhibit, or neutral)	

Load Config **DEFAULT** **▼ pc3types**

About Config Basics Microenvironment User Params **Cell Types** Out: Plots Animate

Cell type: default **▼** inherits properties from parent type: None

This cell line inherits its properties from its parent type. Any settings below override those inherited properties.

phenotype:cycle (model: live; code=5)		
Phase 0 -> Phase 0 transition rate	0.00072	1/min
phenotype:death		
model: apoptosis		
death rate	0.0000531667	1/min
unlysed_fluid_change_rate	0.05	1/min
lysed_fluid_change_rate	0	1/min
cytoplasmic_biomass_change_rate	0.0166667	1/min
nuclear_biomass_change_rate	0.00583333	1/min
calcification_rate	0	1/min
relative_rupture_volume	2	1/min
model: necrosis		
death rate	0	1/min
unlysed_fluid_change_rate	0.05	1/min
lysed_fluid_change_rate	0	1/min
cytoplasmic_biomass_change_rate	0.0166667	1/min
nuclear_biomass_change_rate	0.00583333	1/min
calcification_rate	0	1/min
relative_rupture_volume	2	1/min
phenotype:volume		
total	2494	1/min
fluid_fraction	0.75	1/min

Load Config 2021-06-24 13:58:12.193933 ▾ pc3types

About

Config Basics

Microenvironment

User Params

Cell Types

Out: Plots

Animate

frames

240

resource

YlOrRd

...

Cells

edges

transparency

 Fixed substrate range?

Min

0

Max

1

...

 Substrates dark mode

frame

217



cells: 4d, 12h, 30m (518 agents)

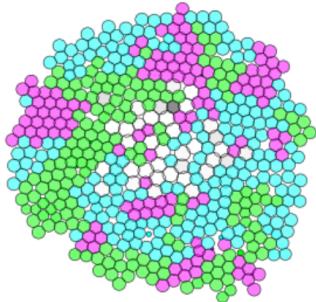
400

200

0

-200

-400



Click/drag the slider knob
to step through output files

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



JAYNE KOSKINAS
TED GIOVANIS

Foundation for
Health and Policy



NATIONAL
CANCER
INSTITUTE



BCRF
BREAST
CANCER
RESEARCH
FOUNDATION

PhysiCell Development:

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

Training Materials:

- Administrative supplement to NCI U01CA232137 (Year 2)