PhysiCell: Quorum Sensing Demo

Furkan Kurtoglu
@FKurtogluSysBio

Aneequa Sundus
@AneequaSundus

PhysiCell Project

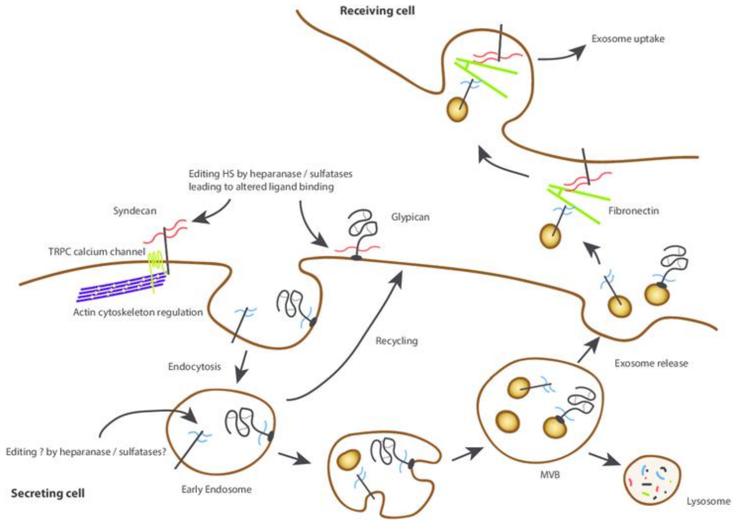
February 16, 2022

Goals

- Build a model from start to finish
  - State the problem
  - Plan the model
  - Adapt the "template" project with Model Builder
  - Change the custom module
  - Run and view results
- How to model basic quorum sensing

Chemical communication

- Cells can communicate by secreting chemical factors:
  - ✦Some stick to the extracellular matrix
    - ♦ Proteoglycans, Some forms of VEGF
    - ♦ Good for leaving “memory”
  - ✦Many diffuse out into the surrounding tissue
    - ♦ IL6, VEGF, CXCL12, CXC/SDF1, Estrogen
    - ♦ Good for long distance communication
  - ✦Some factors never leave the cell surface
    - ♦ Delta-like proteins (e.g., Dll4)
    - ♦ Adhesion molecules
    - ♦ Good for contact signals



Couchman, John & Multhaupt, Hinke & Sanderson, Ralph. (2016). Recent Insights into Cell Surface Heparan Sulphate Proteoglycans and Cancer. F1000Research. 5. 1541. 10.12688/f1000research.8543.1.

Indirect chemical communication

- Single-cell processes can deplete chemical resources
  - Consume resource during metabolism
  - Consume glucose in metabolism
  - Consume growth factors during cycling
- Gradients emerge due to non-homogeneous distribution of cells
- Amounts of chemical substrates has information on local conditions
- Substrate gradients have information on the nearby environment
- Thus, even when cells don't "intend" to communicate, they send information just by altering the chemical environment. This is an indirect communication

Uptake mathematics

Uptake is defined as,

$$\frac{\partial q}{\partial t} = \nabla \cdot D \nabla q - \lambda q - \sum_i \delta\left(\mathbf{x} - \mathbf{x}_i\right) V_i U_i q$$

We then expect uptake dependent on the rate constant U

Secretion mathematics

- You could also use regular secretion:

$$\frac{\partial q}{\partial t} = \nabla \cdot D\nabla q - \lambda q + \sum_i \delta(\mathbf{x} - \mathbf{x}_i)V_i\, S_i(q^* - q)$$

What will happen:

1. Cells will tend secrete until the nearby q density reaches q^*
2. q will have higher values near larger concentrations of cells
3. ∇q will point towards cells.

BUT:

1. $\int_\Omega q\, dV$ will ***not*** be proportional to the total cell count
2. Even a small population can drive q towards q^* for sufficiently large S_i

Export mathematics

Suppose each cell exports q at a constant rate E.

$$\frac{\partial q}{\partial t} = \nabla \cdot D \nabla q - \lambda q + \sum_i \delta(x - x_i) E$$

Now, define $Q(t) = \int_\Omega q \, dV$ to be the total q in the domain.

We can then find,

Q is proportional to the number of cells that secrete q.

Implementation Toolkits: PhysiCell and PhysiCell Model Builder

- We assume you have setup your system
  - Also, for some commands to run directly as written in the slides, the **PhysiCell** and **PhysiCell-model-builder** directories need to be at the same level/within the same directory
- Below are resources to obtain and install all necessary software for the rest of the tutorial:
  - Download PhysiCell 1.9.0 or later
    - http://PhysiCell.org/download
  - Download the PhysiCell model-builder GUI:
    - https://github.com/PhysiCell-Tools/PhysiCell-model-builder
  - Follow the setup tutorials
    - **QuickStart**: https://github.com/MathCancer/PhysiCell/blob/master/documentation/Quickstart.md
    - **MacOS**
      - Slides: https://github.com/physicell-training/ws2021/blob/main/pdfs/PhysiCell_ws2021_macOS_setup.pdf
      - Video: https://www.youtube.com/watch?v=mv_phTdanws
    - **Windows**
      - Slides: https://github.com/physicell-training/ws2021/blob/main/pdfs/PhysiCell_ws2021_Windows_setup.pdf
      - Video: https://www.youtube.com/watch?v=Jp3ZOMt761M

Implementation: Open the toolkits and start up!

- Setup and build the template project
  - Open a terminal/shell and navigate to your **PhysiCell** directory
  - Enter the following
    - `make reset`
    - `make data-cleanup #Note this will remove everything in /output`
    - `make template`
    - `make`
- Open Model Builder GUI
  - Open a second terminal/shell and navigate to the **PhysiCell/config**
  - `python ../../PhysiCell-model-builder/bin/gui4xml.py` *

*This assumes 1) that the Model Builder and PhysiCell folders are within the same directory at the same level in the file tree. 2) You have downloaded Model Builder Release 1.1. *Modify this command as needed to start the Model Builder.*

Memory Refresher…

Quorum sensing

- How can cells "see" or "count" how many cells are nearby?
- How can cells find nearby cells?
- How can cells build an army before attacking?

Quorum factors!

- Many cells (particularly bacteria) secrete a diffusible factor to help it communicate with others of its kind.
- Quorum factors communicate two key pieces of information:
  - ✦They accumulate in regions of high cell density
  - ✦The gradient points towards regions of higher cell density

Using what we know (1)

- Let's create a new project:
  - Diffusing substrate: resource
    - initial value: 0.25 (dimensionless)
    - D = 100000 μm^2/min
    - λ = 0.1 1/min
    - boundary value: 0.25 on $x_{\min}$,
    - zero flux on all other boundaries.
  - Cells of type "bacteria":
    - Uptake resource at rate chosen to get 100 micron length scale
    - They move by chemotaxis up resource gradients.
    - Place in a circle in the center
  - Cells of type "supplier":
    - Release resource at a high rate towards saturation value of 1
    - No birth or death
    - Can't be moved
    - Place randomly in the domain

Using what we know (2) (round 1)

- Let's start with one diffusing substrate:
  - ✦resource (Dirichlet condition 1 mmHg)
- **Cell type "bacteria":**
  - ✦**Proliferate proportional to resource**
  - ✦**Die if resource is below a threshold**
- **Agent type "supplier":**
  - ✦Don't proliferate or die.
  - ✦Can't be moved
  - ✦Release resource

Approach

- Change Dirichlet condition for resource in PhysiCell_settings.xml

- Add custom variables (to default definition):
  - ❿R_necrosis, R_max_growth, necrosis_rate

- Create and use phenotype function for bacteria:
  - ✦bacteria_phenotype

First, let's get a clean template project

- make data-cleanup
- make clean
- make reset
- make template

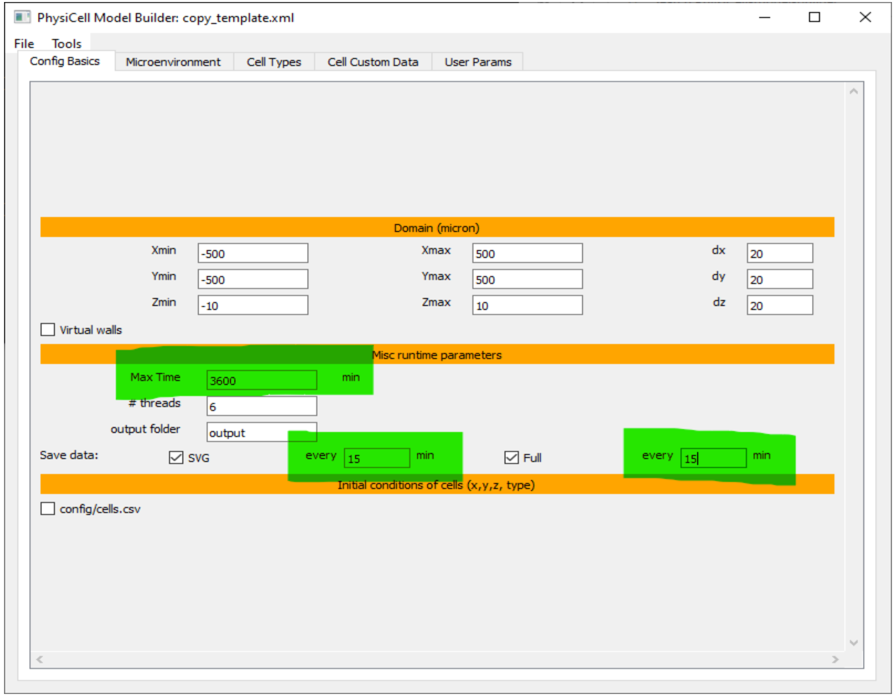
Simulation time and interval for outputs

From `PhysiCell\config` (anaconda)

```
python ..\..\PhysiCell-model-builder\bin\gui4xml.py
```

Set to 3600 minutes of simulation

Output SVG files every 15 minutes



Domain (micron)

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

□ Virtual walls

Misc runtime parameters

Max Time 3600 min

threads 6

output folder output

Save data: ☑ SVG every 15 min ☑ Full every 15 min

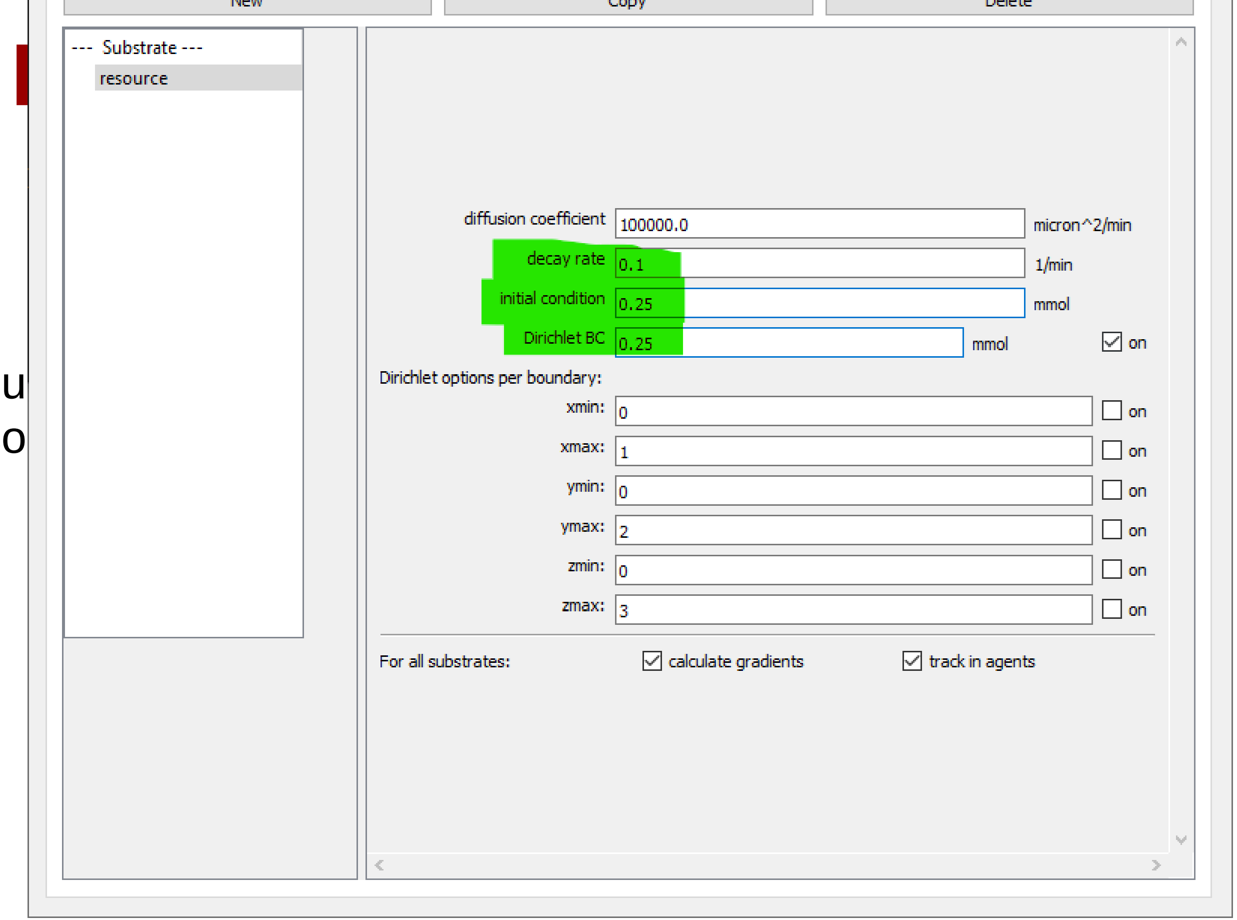
Initial conditions of cells (x,y,z, type)

□ config/cells.csv

Changes to I

Set up the substrate

- Rename substrate to resou
- Change decay rate, initial co boundarycondition values.



--- Substrate ---

resource

diffusion coefficient: 100000.0 micron^2/min

decay rate: 0.1 1/min

initial condition: 0.25 mmol

Dirichlet BC: 0.25 mmol ☑ on

Dirichlet options per boundary:

xmin: 0 □ on

xmax: 1 □ on

ymin: 0 □ on

ymax: 2 □ on

zmin: 0 □ on

zmax: 3 □ on

For all substrates: ☑ calculate gradients ☑ track in agents

Create the new cell definitions

Create a definition to bacteria (math)

- We want a diffusion length of 100 μm, and D = 100000 μm²/min. We need the uptake rate *U*:

$$100\ \mu m = L = \sqrt{\frac{D}{U}} = \sqrt{\frac{100000\ \mu m^2/\text{min}}{U}}$$

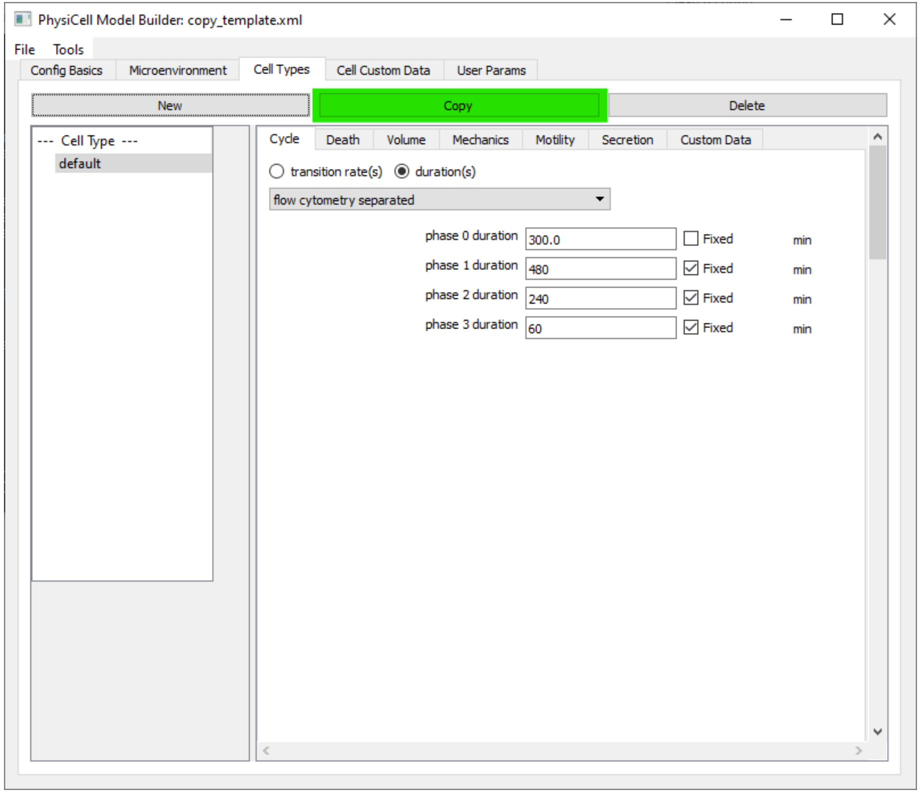
- Solving for *U*, we have:

$$U = \frac{D}{L^2} = \frac{100000\ \mu m^2/\text{min}}{10000\ \mu m^2} = 10\ \text{min}^{-1}$$

Changes to PhysiCell_settings.xml (2)

Create bacteria definition from default type

- Click on copy
- Rename bacteria
- Activate chemotaxis on resource gradient direction
- Set the resource update rate



○ transition rate(s) ● duration(s)

flow cytometry separated

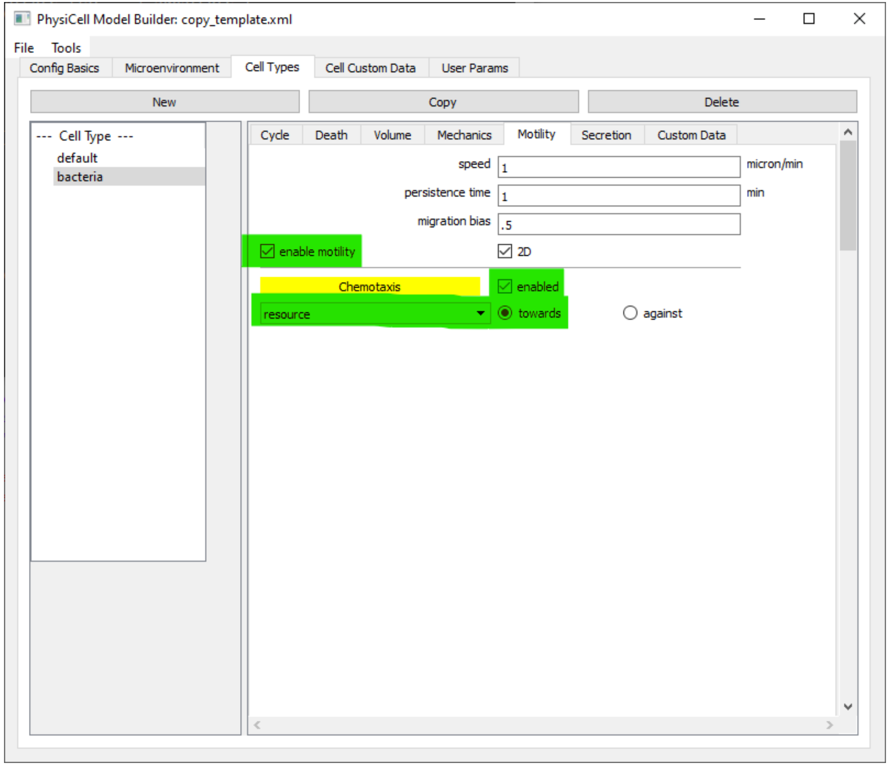
phase 0 duration 300.0 □ Fixed min

phase 1 duration 480 ☑ Fixed min

phase 2 duration 240 ☑ Fixed min

phase 3 duration 60 ☑ Fixed min

Changes to PhysiCell_settings.xml (3)



speed 1 micron/min

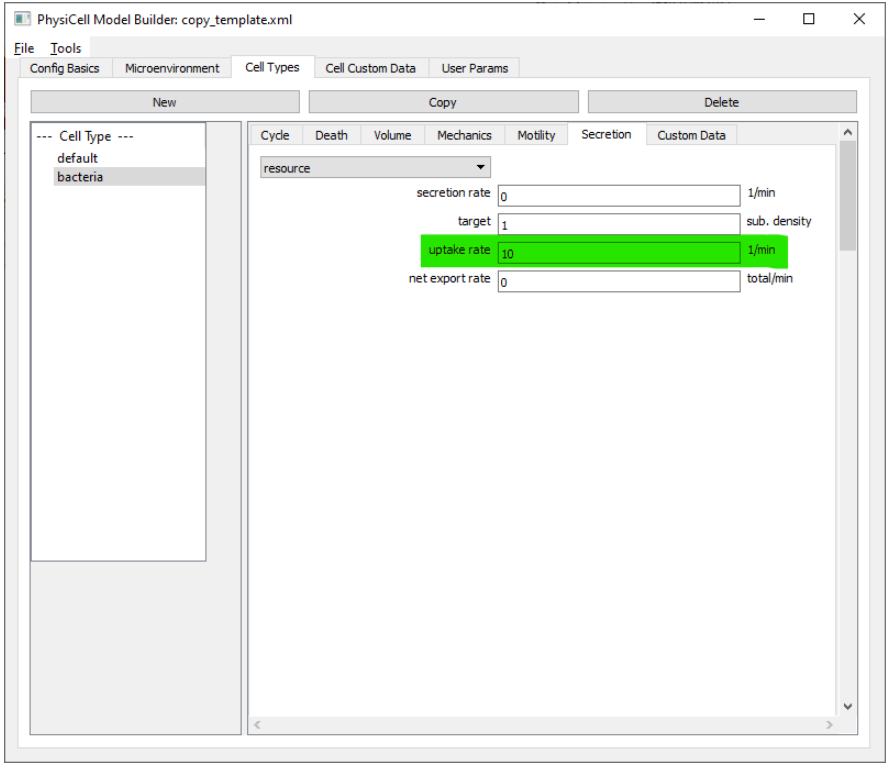
persistence time 1 min

migration bias .5

☑ enable motility ☑ 2D

Chemotaxis ☑ enabled

resource ● towards ○ against



resource

secretion rate 0 1/min

target 1 sub. density

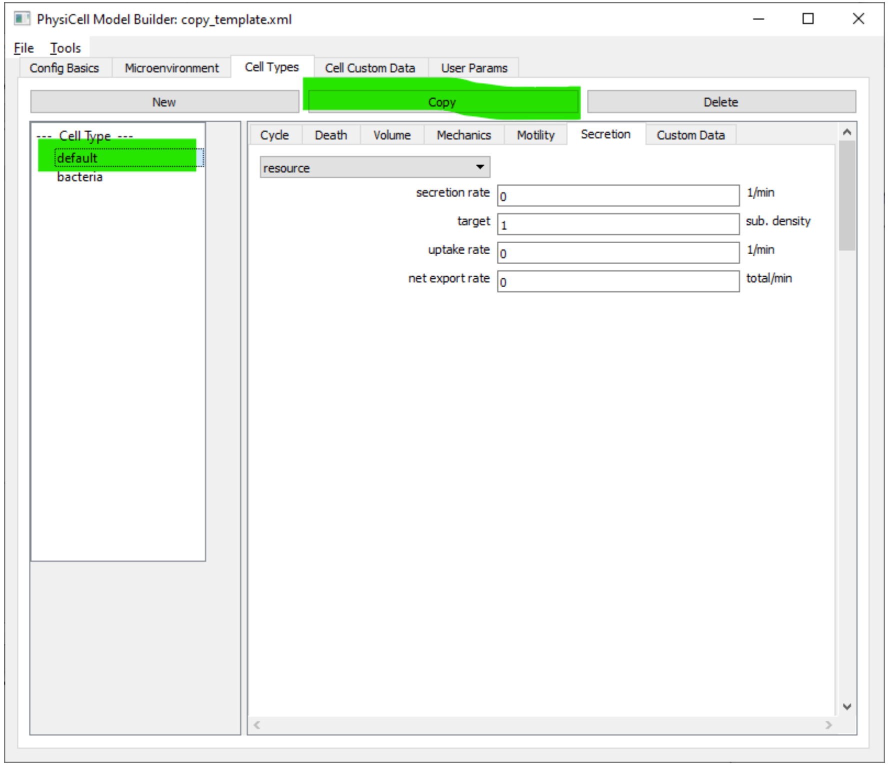
uptake rate 10 1/min

net export rate 0 total/min

Changes to PhysiCell_settings.xml (4)

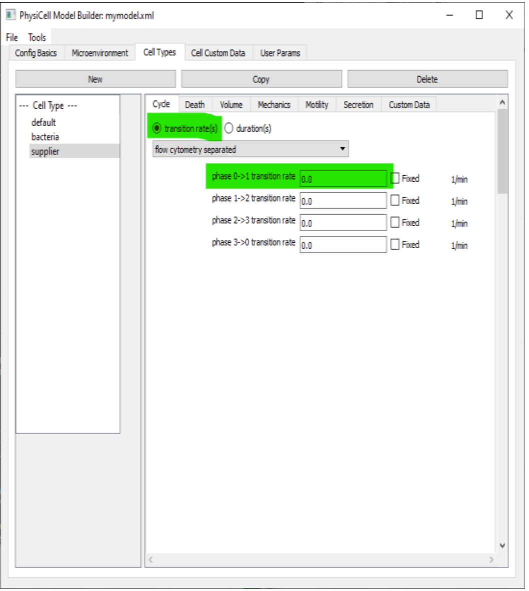
Create supplier definition from default type

- Select default type and click on copy
- Set proliferation off
- Set dead off
- Set the resource secretion



secretion rate 0 1/min
target 1 sub. density
uptake rate 0 1/min
net export rate 0 total/min

Changes to PhysiCell_settings.xml (5)



● transition rate(s) ○ duration(s)

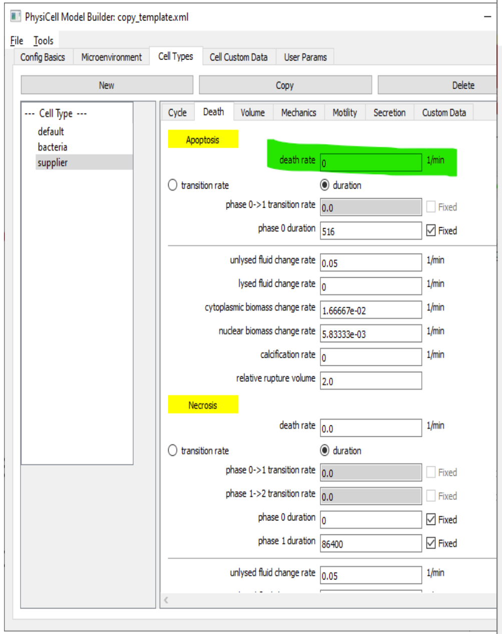
flow cytometry separated

phase 0->1 transition rate 0.0 □ Fixed 1/min

phase 1->2 transition rate 0.0 □ Fixed 1/min

phase 2->3 transition rate 0.0 □ Fixed 1/min

phase 3->0 transition rate 0.0 □ Fixed 1/min



Apoptosis

death rate 0 1/min

○ transition rate ● duration

phase 0->1 transition rate 0.0 □ Fixed

phase 0 duration 516 ☑ Fixed

unlysed fluid change rate 0.05 1/min

lysed fluid change rate 0 1/min

cytoplasmic biomass change rate 1.66667e-02 1/min

nuclear biomass change rate 5.83333e-03 1/min

calcification rate 0 1/min

relative rupture volume 2.0

Necrosis

death rate 0.0 1/min

○ transition rate ● duration

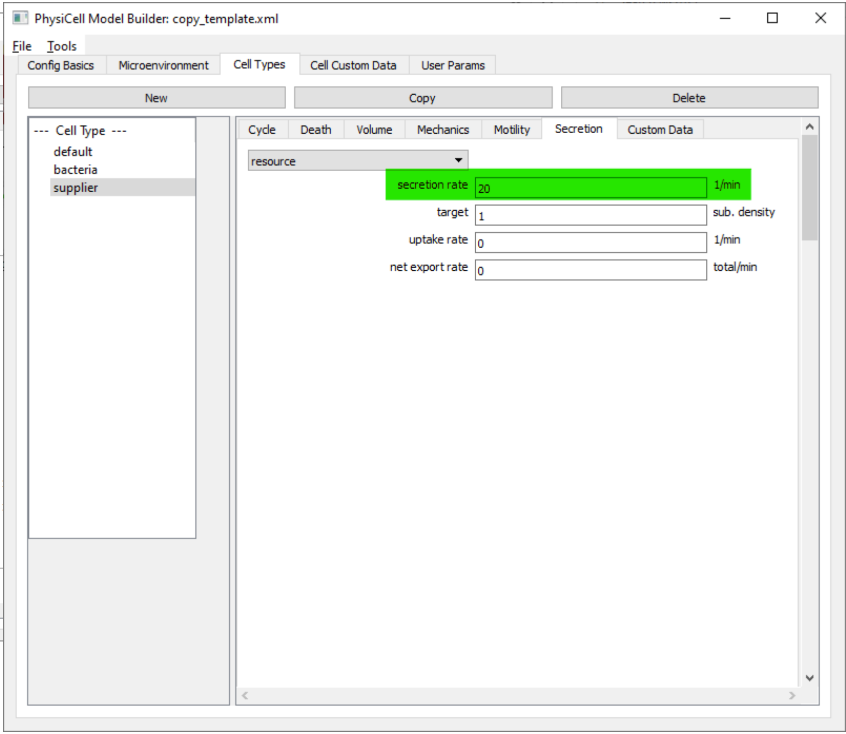
phase 0->1 transition rate 0.0 □ Fixed

phase 1->2 transition rate 0.0 □ Fixed

phase 0 duration 0 ☑ Fixed

phase 1 duration 86400 ☑ Fixed

unlysed fluid change rate 0.05 1/min



resource

secretion rate 20 1/min

target 1 sub. density

uptake rate 0 1/min

net export rate 0 total/min

Changes in custom.cpp

Change to create_cell_types()

```
  /*
     Put any modifications to individual cell definitions here.

     This is a good place to set custom functions.
  */

  cell_defaults.functions.update_phenotype = phenotype_function;
  cell_defaults.functions.custom_cell_rule = custom_function;
     cell_defaults.functions.contact_function = contact_function;

  Cell_Definition* pBacteria = find_cell_definition( "bacteria" );
  pBacteria->functions.update_phenotype = bacteria_phenotype;

  /*
     This builds the map of cell definitions and summarizes the setup.
  */

  build_cell_definitions_maps();
  display_cell_definitions( std::cout );

  return;
}
```

Modify cell placement

Changed to custom.cpp (setup_tissue)

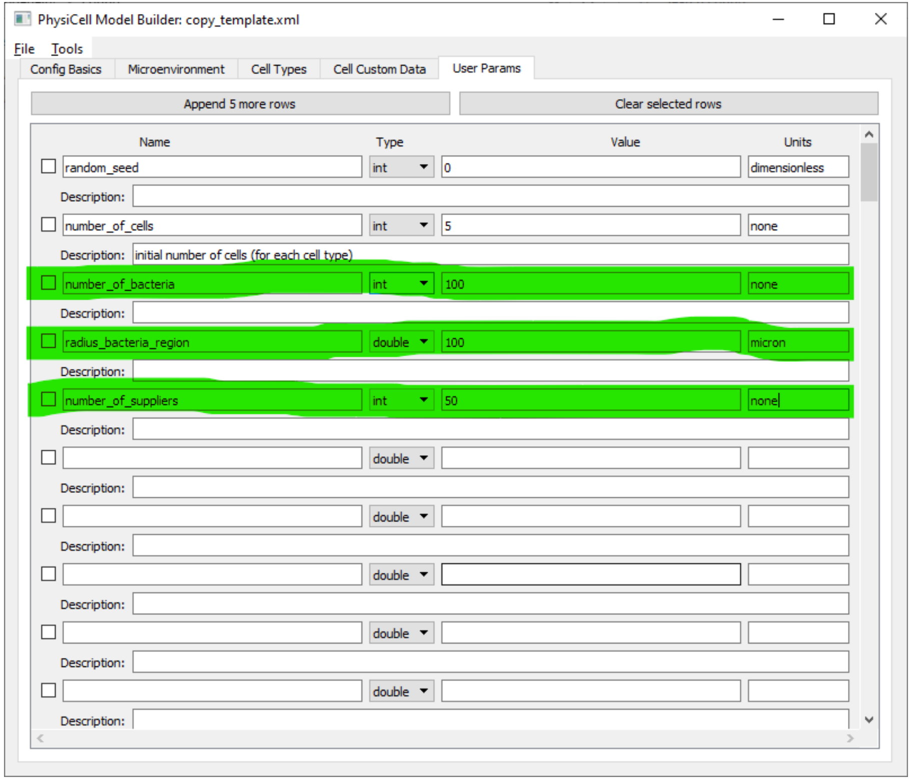
```
void setup_tissue( void )
{
   double Xmin = microenvironment.mesh.bounding_box[0];
   double Ymin = microenvironment.mesh.bounding_box[1];
   double Zmin = microenvironment.mesh.bounding_box[2];

   double Xmax = microenvironment.mesh.bounding_box[3];
   double Ymax = microenvironment.mesh.bounding_box[4];
   double Zmax = microenvironment.mesh.bounding_box[5];

   if( default_microenvironment_options.simulate_2D == true )
   {
      Zmin = 0.0;
      Zmax = 0.0;
   }

   double Xrange = Xmax - Xmin;
   double Yrange = Ymax - Ymin;
   double Zrange = Zmax - Zmin;

   double center_x = 0.5*(Xmin+Xmax);
   double center_y = 0.5*(Ymin+Ymax);
   double center_z = 0.5*(Zmin+Zmax);

   // create some of each type of cell

   Cell* pC;

   // find cell definitions
   Cell_Definition* pBacteria = find_cell_definition( "bacteria" );


...
```

```
...

   // find cell definitions
   Cell_Definition* pBacteria = find_cell_definition( "bacteria" );

   Cell_Definition* pSupplier = find_cell_definition( "supplier" );

   for( int k=0; k<parameters.ints("number_of_bacteria"); k++ )
   {
       std::vector<double> position = {0,0,0};
       double r = NormalRandom(0,1) *
       parameters.doubles( "radius_bacteria_region" );
       double theta = 6.28318530718 * UniformRandom();

       position[0] = center_x + r*cos(theta);
       position[1] = center_y + r*sin(theta);
       position[2] = center_z;

       pC = create_cell( *pBacteria );
       pC->assign_position( position );
   }
   for( int k=0; k<parameters.ints("number_of_suppliers"); k++ )
   {
       std::vector<double> position = {0,0,0};
       position[0] = Xmin + UniformRandom()*Xrange;
       position[1] = Ymin + UniformRandom()*Yrange;
       position[2] = Zmin + UniformRandom()*Zrange;

       pC = create_cell( *pSupplier );
       pC->assign_position( position );

       pC->is_movable = false;
    }

    return;
}
```

Changes to PhysiCell_settings.xml (6)

Define parameters on xml file

- Include to the parameters: `number_of_bacteria`, `radius_bacteria_region`, and `number_of_suppliers`



	Name	Type	Value	Units	Description
□	random_seed	int	0	dimensionless	
□	number_of_cells	int	5	none	initial number of cells (for each cell type)
□	number_of_bacteria	int	100	none	
□	radius_bacteria_region	double	100	micron	
□	number_of_suppliers	int	50	none	
□		double			
□		double			
□		double			
□		double			
□		double			

Bacteria phenotype (1)

```
// declare function in custom.h

void bacteria_phenotype( Cell* pCell, Phenotype& phenotype , double dt );

// create it in custom.cpp

void bacteria_phenotype( Cell* pCell, Phenotype& phenotype , double dt )
{
    if( phenotype.death.dead == true )
    {
        pCell->functions.update_phenotype = NULL; // don't bother doing this function again!
        return;
    }

    // find my cell definition
    // don't use static if you plan to use this for more than one cell type
    static Cell_Definition* pCD = find_cell_definition( pCell->type_name );

    // find the index of resource
    static int nR = microenvironment.find_density_index( "resource" );

    // index of necrotic death model
    static int nNecrosis = 1; // PhysiCell_constants::necrosis_death_model;

    // sample microenvironment at cell position to get resource
    double R = pCell->nearest_density_vector()[nR];

    // check for necrotic death
    if( R < pCell->custom_data["R_necrosis"] )
    { phenotype.death.rates[nNecrosis] = pCell->custom_data["necrosis_rate"]; }
    else
    { phenotype.death.rates[nNecrosis] = 0.0; }

    // ...
```

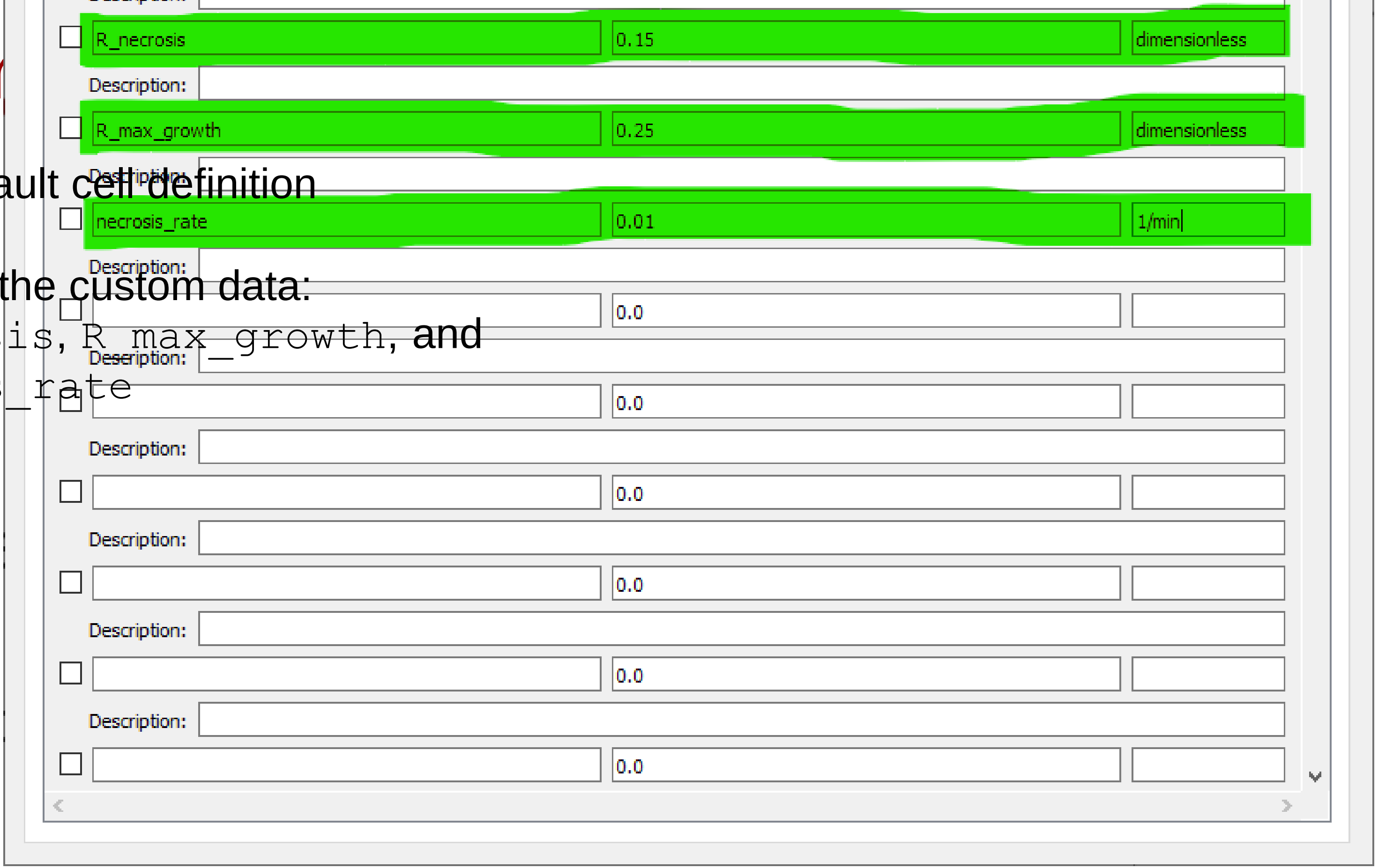
Bacteria phenotype (2)

```
  // ...

  // set birth rate

  // set proliferation
  // first, set to the cell line rate
  phenotype.cycle.data.transition_rate(0,1) =
     pCD->phenotype.cycle.data.transition_rate(0,1);

  // scale with R
  double scaling_factor = (R - pCell->custom_data["R_necrosis"])
  / (pCell->custom_data["R_max_growth"] - pCell->custom_data["R_necrosis"]);
  if( scaling_factor > 1 )
  { scaling_factor = 1.0; }
  if( scaling_factor < 0 )
  { scaling_factor = 0.0; }

  // multiply by scaling factor
  phenotype.cycle.data.transition_rate(0,1) *= scaling_factor;

  return;
}
```

Char

Update the default cell definition

- Include to the custom data: `R_necrosis`, `R_max_growth`, and `necrosis_rate`



Description:

□ R_necrosis | 0.15 | dimensionless

Description:

□ R_max_growth | 0.25 | dimensionless

Description:

□ necrosis_rate | 0.01 | 1/min

Description:

□ | 0.0 |

Description:

□ | 0.0 |

Description:

□ | 0.0 |

Description:

□ | 0.0 |

Description:

□ | 0.0 |

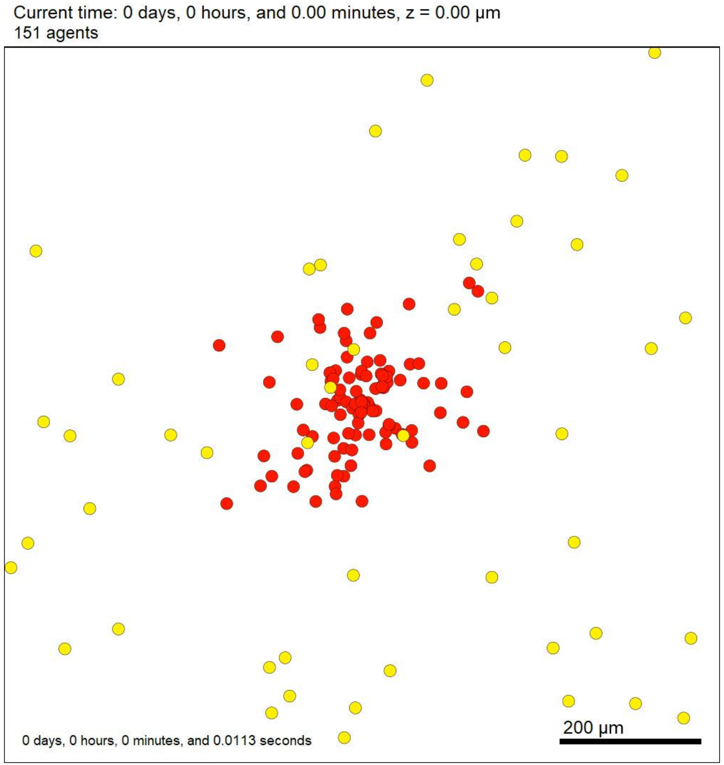
Description:

□ | 0.0 |

Give it a try!

```
make data-cleanup
make
.\project .\config\mymodel.xml
```

```
python beta/plot_cells.py
```



Link video

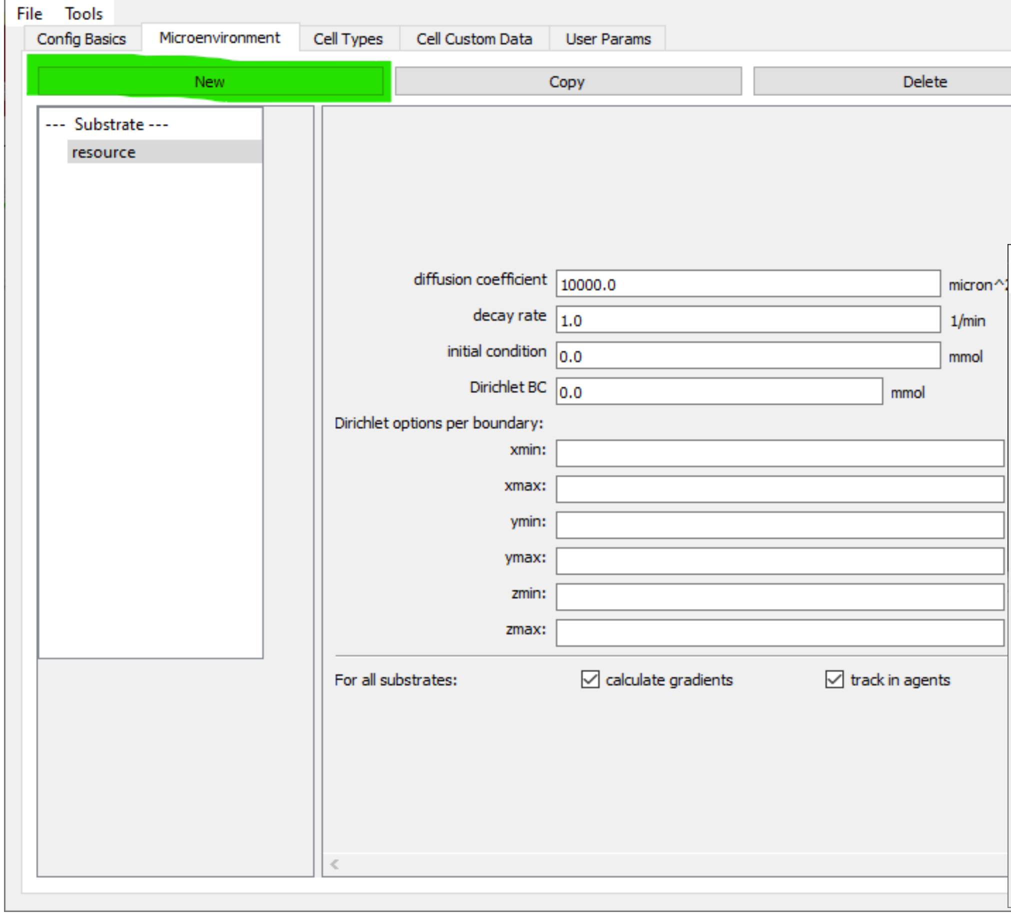
Using what we know (2) (round 2)

- Let's use two diffusing substrates:
  - ✦resource (Dirichlet condition 1 mmHg)
  - ✦**quorum factor (Neumann condition)**
- **Cell type "bacteria":**
  - ✦Proliferate proportional to resource
  - ✦Die if resource is below a threshold
  - ✦**Secrete *q***
  - ✦**Chemotax towards regions of high *q***
- **Agent type "supplier":**
  - ✦Don't proliferate or die.
  - ✦Can't be moved
  - ✦Release resource

Approach

- In PhysiCell_settings.xml
  - ✦Add "quorum factor" to microenvironment
    - ♦ Let's use a diffusion coefficient of 10000, decay rate of 1
    - ♦ Neumann conditions!
  - ✦Add corresponding secretion / uptake to default cell definition
  - ✦make sure bacteria export quorum factor
    - ♦ Let's use a rate of 1 for now.
  - ✦change bacteria chemotaxis to quorum factor

tings.xml (1)



File Tools

Config Basics | Microenvironment | Cell Types | Cell Custom Data | User Params

New | Copy | Delete

--- Substrate ---
resource

diffusion coefficient: 10000.0 micron^2

decay rate: 1.0 1/min

initial condition: 0.0 mmol

Dirichlet BC: 0.0 mmol

Dirichlet options per boundary:

xmin:

xmax:

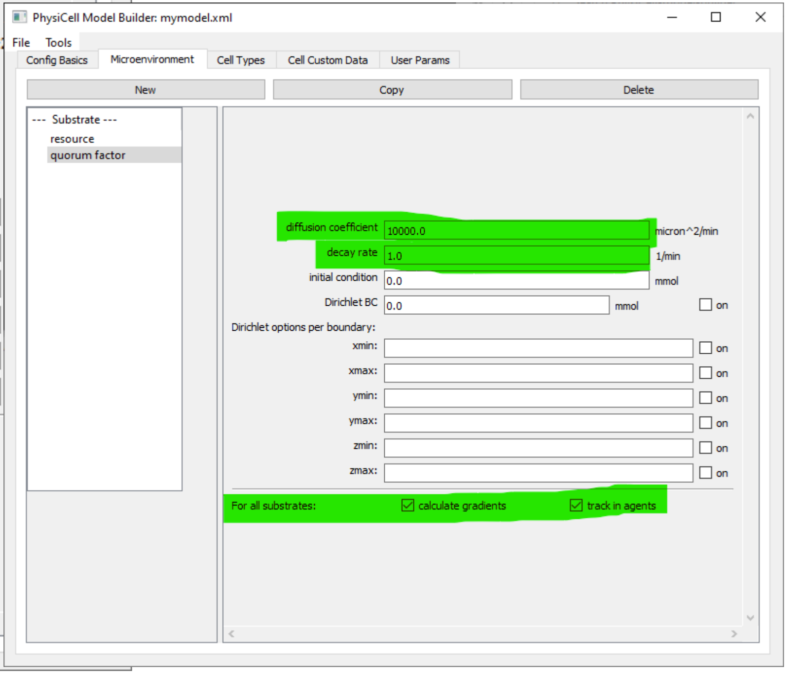
ymin:

ymax:

zmin:

zmax:

For all substrates: ☑ calculate gradients ☑ track in agents



PhysiCell Model Builder: mymodel.xml

File Tools

Config Basics | Microenvironment | Cell Types | Cell Custom Data | User Params

New | Copy | Delete

--- Substrate ---
resource
quorum factor

diffusion coefficient: 10000.0 micron^2/min

decay rate: 1.0 1/min

initial condition: 0.0 mmol

Dirichlet BC: 0.0 mmol □ on

Dirichlet options per boundary:

xmin: □ on

xmax: □ on

ymin: □ on

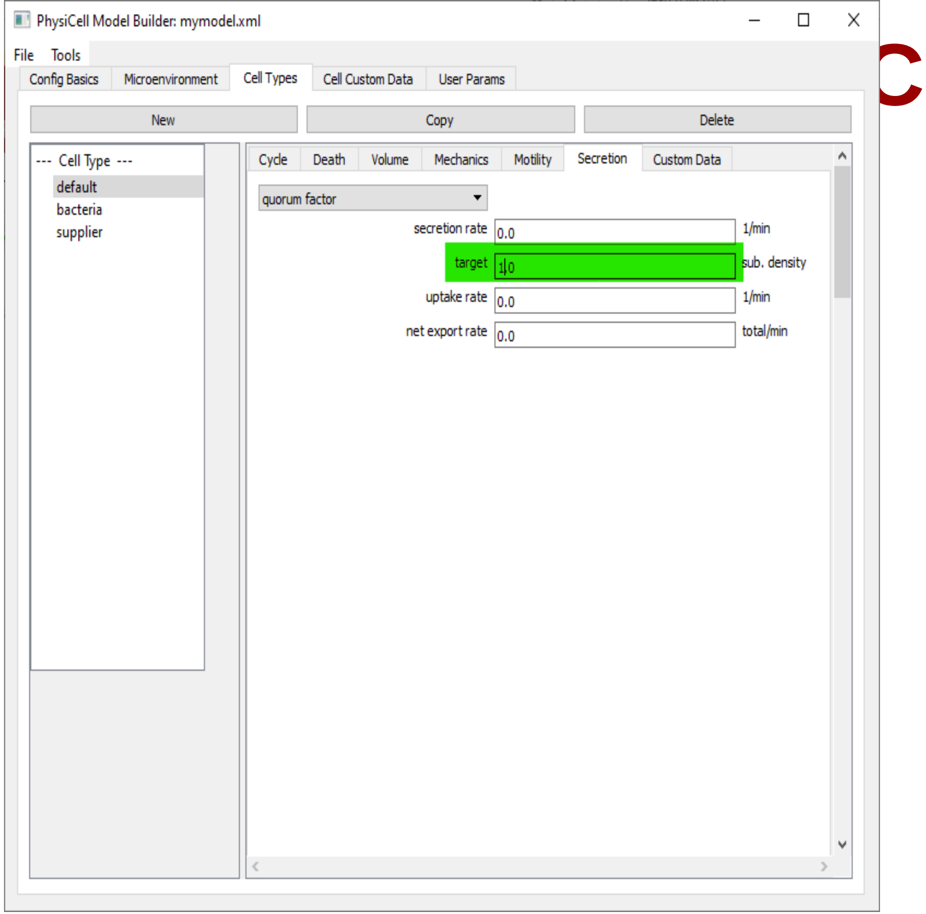
ymax: □ on

zmin: □ on

zmax: □ on

For all substrates: ☑ calculate gradients ☑ track in agents

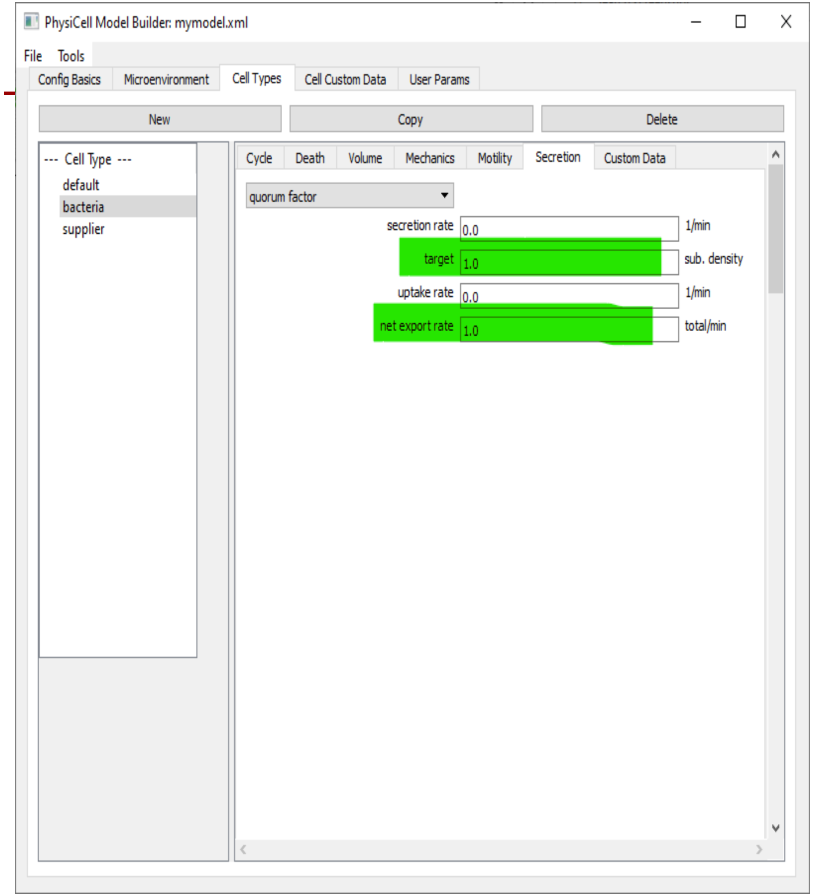
Cell_



PhysiCell Model Builder: mymodel.xml

quorum factor

secretion rate	0.0	1/min
target	1.0	sub. density
uptake rate	0.0	1/min
net export rate	0.0	total/min

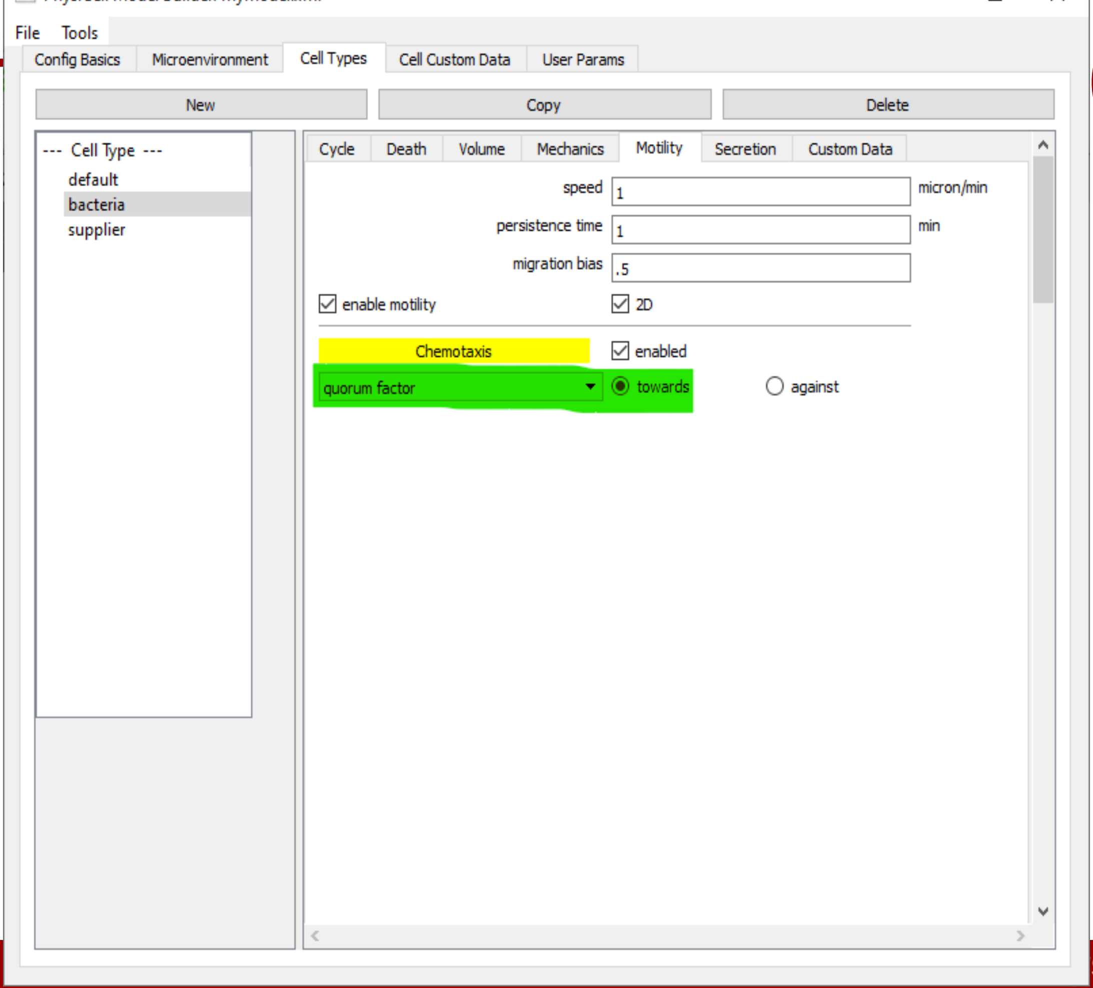


PhysiCell Model Builder: mymodel.xml

quorum factor

secretion rate	0.0	1/min
target	1.0	sub. density
uptake rate	0.0	1/min
net export rate	1.0	total/min

Changes (3)



--- Cell Type ---
default
bacteria
supplier

speed: 1 micron/min

persistence time: 1 min

migration bias: .5

☑ enable motility ☑ 2D

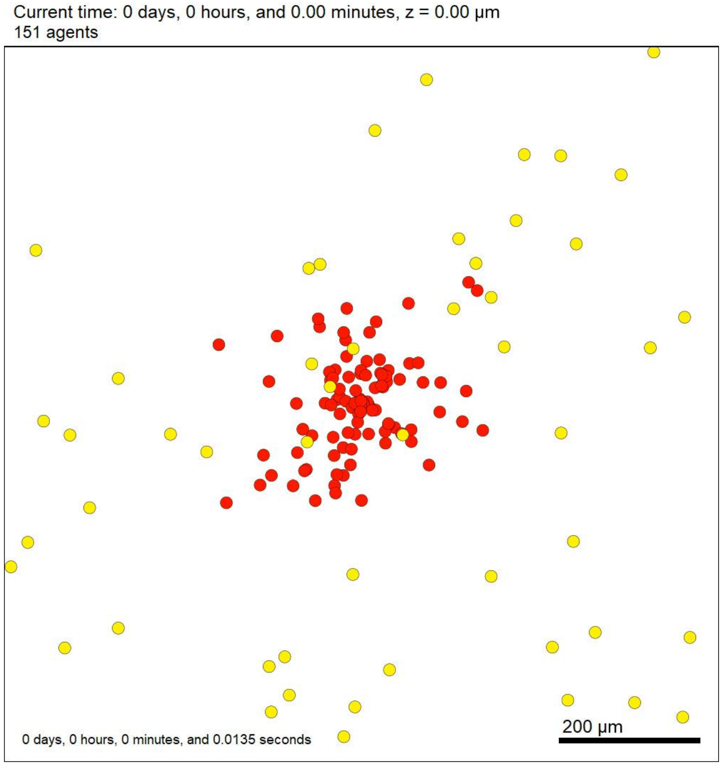
Chemotaxis ☑ enabled

quorum factor ● towards ○ against

Give it a try!

```
make data-cleanup
make
.\project .\config\mymodel.xml
```

```
Python beta/plot_cells.py
```



Link video

Using what we know (2) (round 3)

- Let's use two diffusing substrates:
  - ✦resource (Dirichlet condition 1 mmHg)
  - ✦quorum factor (Neumann condition)

- **Cell type "bacteria":**
  - ✦Proliferate proportional to resource
  - ✦Die if resource is below a threshold
  - ✦Secrete q
  - ✦Chemotax towards regions of high q
  - ✦**Slow down motility when *q* is high**

- **Agent type "supplier":**
  - ✦Don't proliferate or die.
  - ✦Can't be moved
  - ✦Release resource

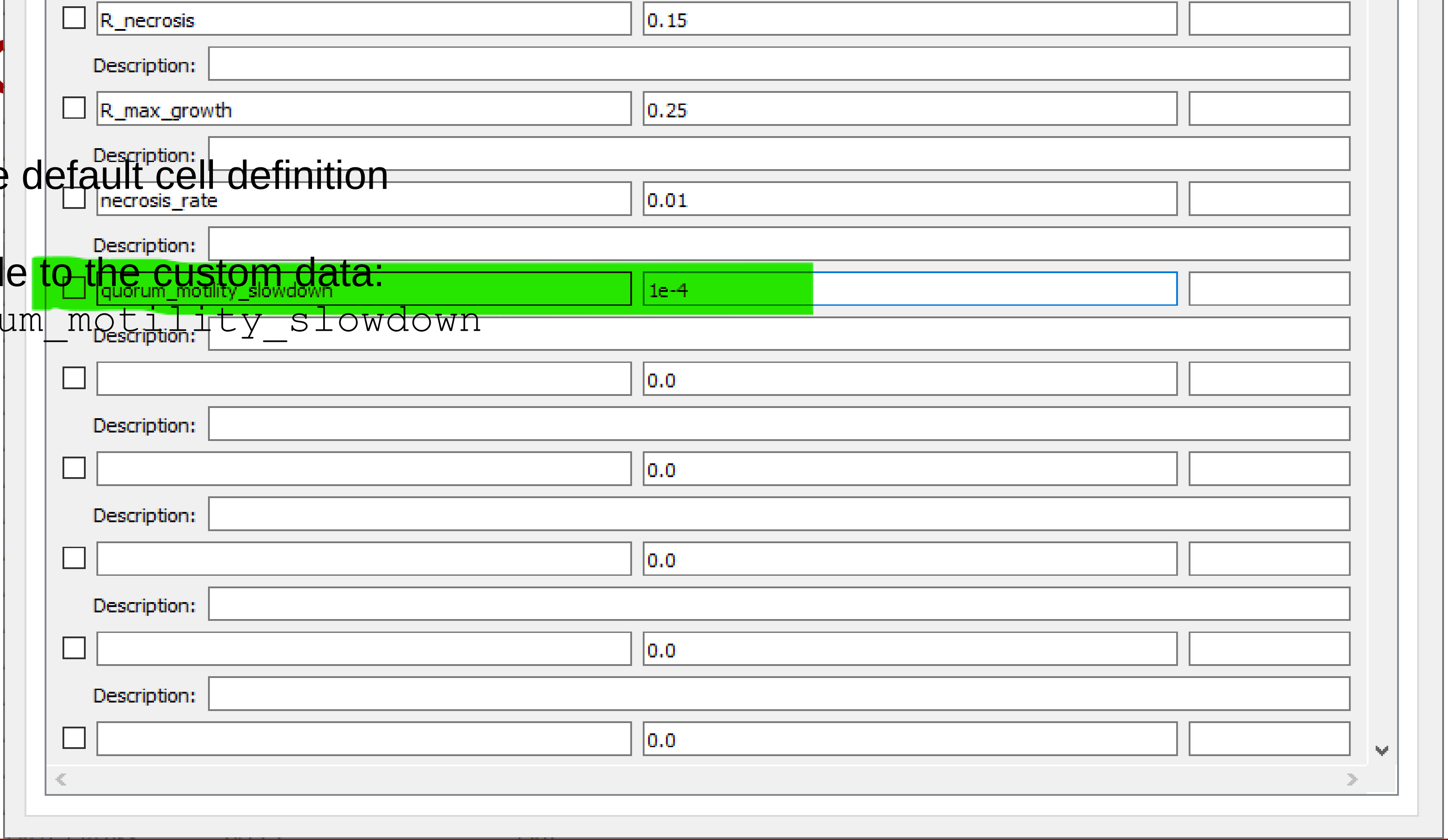
Approach

- In PhysiCell_settings.xml
  - Add custom data to default cell definition
    - quorum_motility_slowdown (we'll default to 1e-4)

- In custom.cpp
  - In bacteria_phenotype()
    - scale phenotype.motility.speed by $\max\left(0,1-\frac{q}{q_{mot}}\right)$

Update the default cell definition

- Include to the custom data: `quorum_motility_slowdown`



□	R_necrosis	0.15	
	Description:		
□	R_max_growth	0.25	
	Description:		
□	necrosis_rate	0.01	
	Description:		
□	quorum_motility_slowdown	1e-4	
	Description:		
□		0.0	
	Description:		
□		0.0	
	Description:		
□		0.0	
	Description:		
□		0.0	
	Description:		
□		0.0	

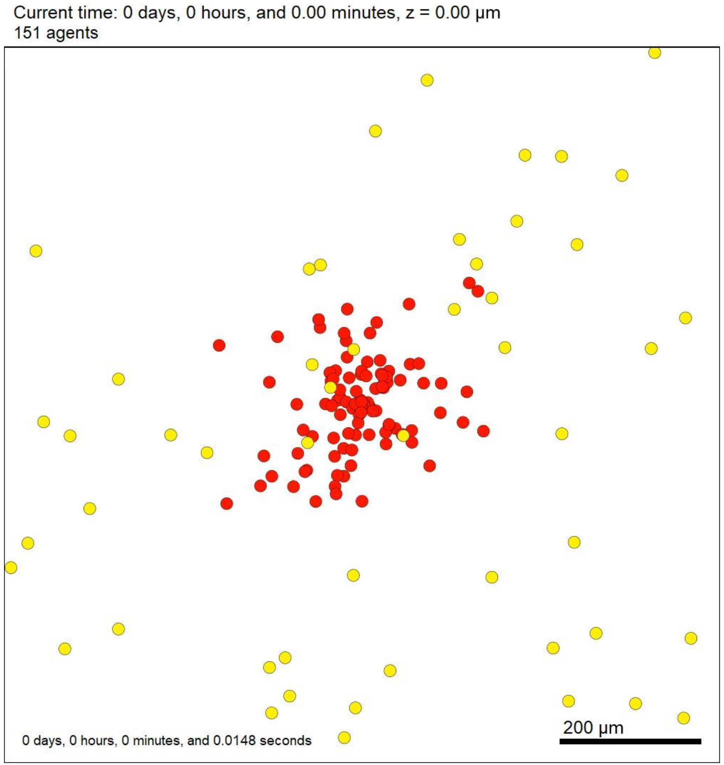
Changes to bacteria phenotype (2)

```
  // ...

 // sampling Quorum Factor
  double q = pCell->nearest_density_vector()[nQ];
  static int nQ = microenvironment.find_density_index( "quorum factor" );

  // get the cell line's motile speed
  phenotype.motility.migration_speed = pCD->phenotype.motility.migration_speed;

  // get a scaling factor
  scaling_factor = 1.0 - q / pCell->custom_data["quorum_motility_slowdown"];
  if( scaling_factor < 0.0 )
  { scaling_factor = 0.0; }

  // scale migration speed
  phenotype.motility.migration_speed *= scaling_factor;

  return;
}
```

Give it a try!

```
make data-cleanup
make
.\project .\config\mymodel.xml
```

```
Python beta/plot_cells.py
```

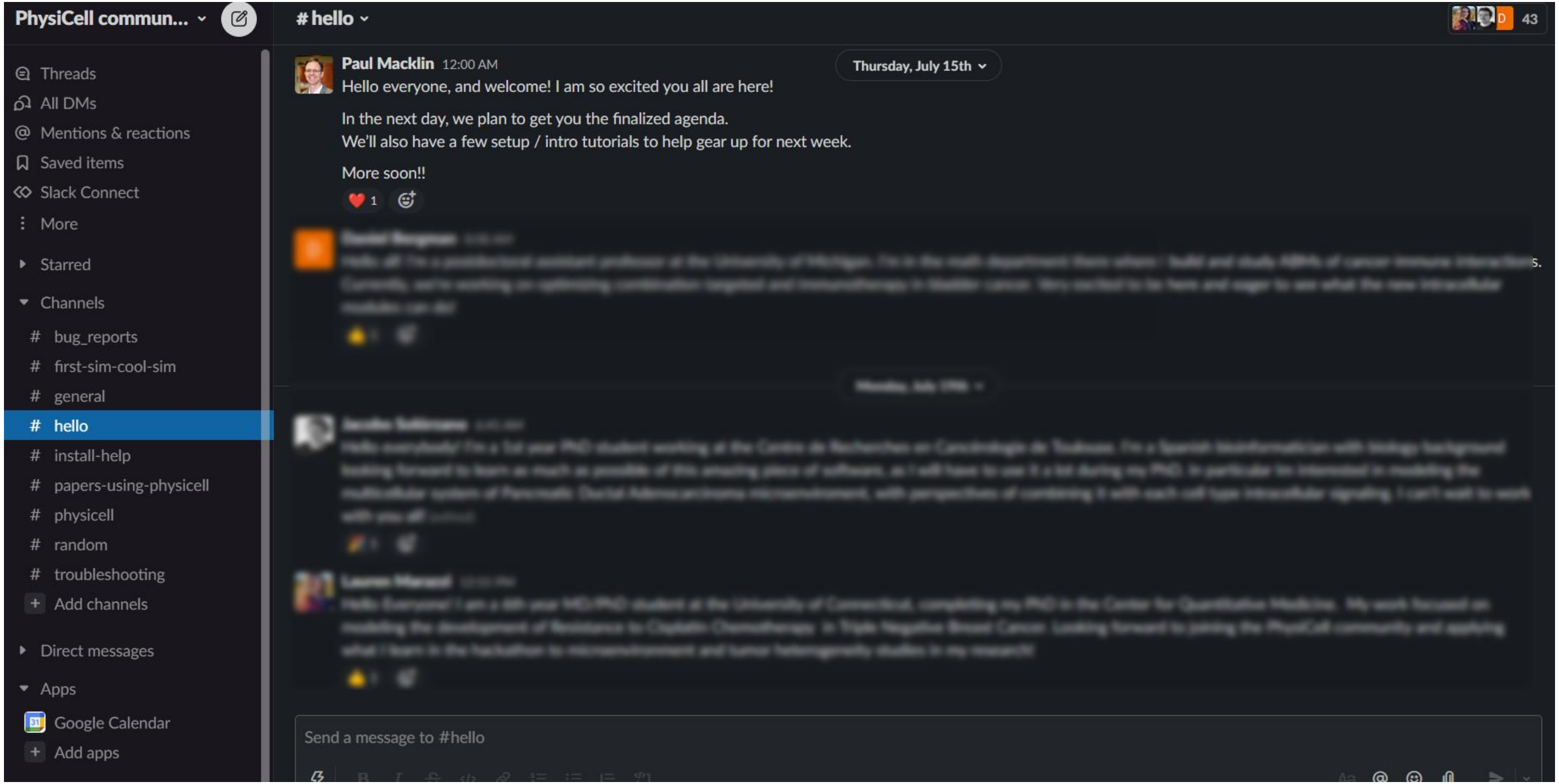


Link video

PhysiCell Slack

- Slack Workspace for PhysiCell Community: https://join.slack.com/t/physicellcomm-sf93727/shared_invite/zt-qj1av6yd-yVeer8VkQaNDjDz7fF00jA

Slack



Paul Macklin 12:00 AM

Hello everyone, and welcome! I am so excited you all are here!

In the next day, we plan to get you the finalized agenda.
We'll also have a few setup / intro tutorials to help gear up for next week.

More soon!!

❤️ 1

[redacted]

[redacted]

[redacted]

[redacted]

Acknowledgment

Funding

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

MathCancer Lab

Michael Getz

Randy Heiland

Kali Konstantinopoulos

Paul Macklin

John Metzcar

Heber Rocha

Aneequa Sundus

Yafei Wang

libroadrunner

Herbert Sauro

Andy Somogyi

University of Southern California

Stacey Finley

Shannon Mumenthaler

Credits

Funding:

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)