

Session 8: Chemical Communication in PhysiCell



Heber Rocha



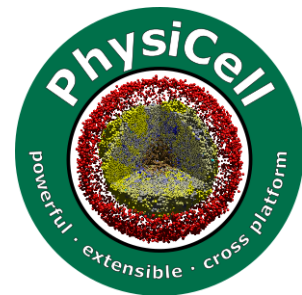
@HeberLRocha

Michael Getz

@mich_getz

Intelligent Systems Engineering
Indiana University

July 27, 2021



LUDDY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

PhysiCell Project

PhysiCell.org

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

What we learned last session

- PhysiCell full modeling workflow
- Handy C++ Tidbits for Cell Agents
- Typical form / syntax of PhysiCell functions
- The customizable functions in Cell.functions
- How to assign new functions to a cell definition
- Sampling the microenvironment at Cell locations
- Controlling initial cell placement
- Custom coloring functions

What we will learn in session 8

1. Secretion, uptake, export

- How do we expect cells to communicate
- How do we handle this process
 - ◆ Export/ Secretion/ Uptake definitions

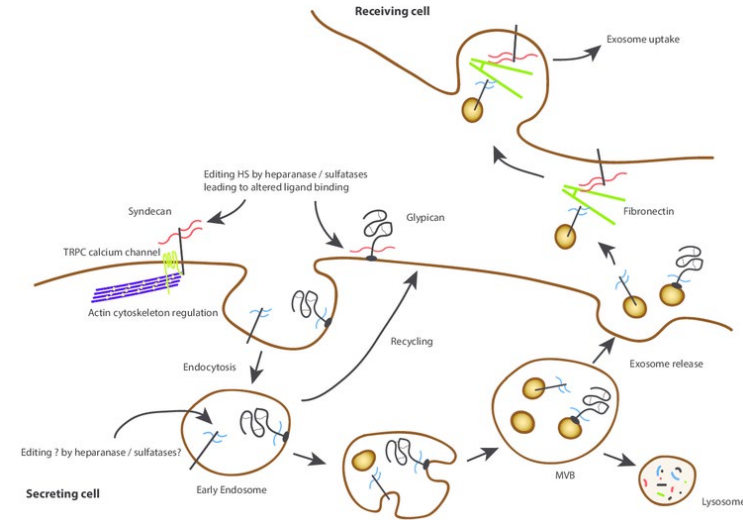
2. Example: Quorum sensing (advanced)

- Edit .cpp files to add custom functionality
- Edit rates
- Edit signal interactions

[github repository](#)

Chemical communication

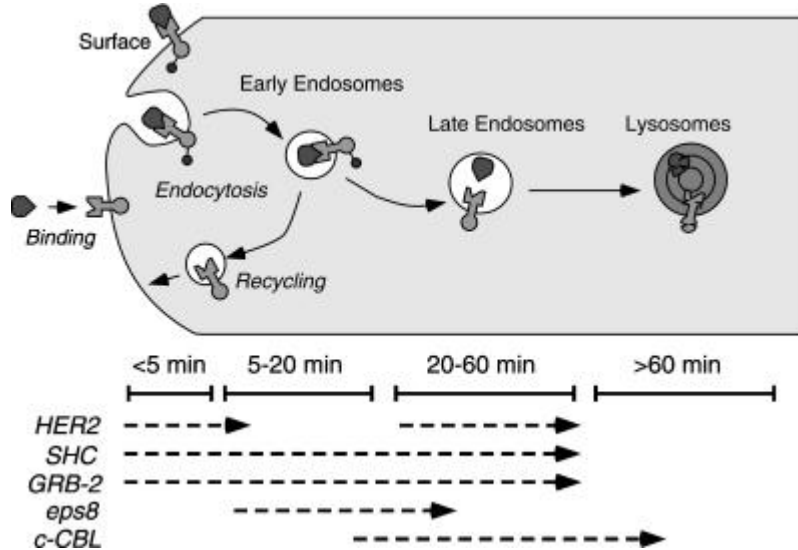
- Cells can communicate by secreting chemical factors:
 - Some stick to the extracellular matrix
 - Proteoglycans, Some forms of VEGF
 - Good for leaving “memory”
 - Many diffuse out into the surrounding tissue
 - IL6, VEGF, CXCL12, CXCL12/SDF1, Estrogen
 - Good for long distance communication
 - Some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



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

Chemical receptors

- Chemical signaling by receptors can be rate limited by **receptor trafficking**



Burke, P et al. "Regulation of epidermal growth factor receptor signaling by endocytosis and intracellular trafficking." *Molecular biology of the cell* vol. 12,6 (2001): 1897-910. doi:10.1091/mbc.12.6.1897

- If some steps are slower than others (e.g., slow recycling of receptors, or very stable receptor-ligand binding), then cells can use up all the receptors and lose response to further signal.

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

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 .

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

Uptake mathematics

Uptake is defined as,

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

We then expect uptake dependent on the rate constant U

Accessing internalized substrates

- By default, PhysiCell keeps track of the net amount of internalized substrates.
- Each environmental substrate has a corresponding internalized substrate with the same index.

```
phenotype.molecular.internalized_total_substrates[ index ]
```

- Cells can release their contents at death. Set this (on a per-substrate basis) via

```
phenotype.molecular.fraction_released_at_death[ index ]
```

- Similarly, if the cell is eaten, the attacking cell can acquire some or all of the contents

```
phenotype.molecular.fraction_transferred_when_ingested[ index ]
```

WARNING: If cells are secreting (or exporting), the internalized substrates can go to negative values unless you write code to internally generate this quantity. Use the "at death / when ingested" options with caution.

Biological example: Quorum sensing



LUDDY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

PhysiCell Project

PhysiCell.org

 @PhysiCell

Quorum sensing (Advanced)

- 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)
 - ♦ $D = 100000 \mu\text{m}^2/\text{min}$
 - ♦ $\lambda = 0.1 \text{ 1/min}$
 - ♦ 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):
 - R_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 reset
- make template

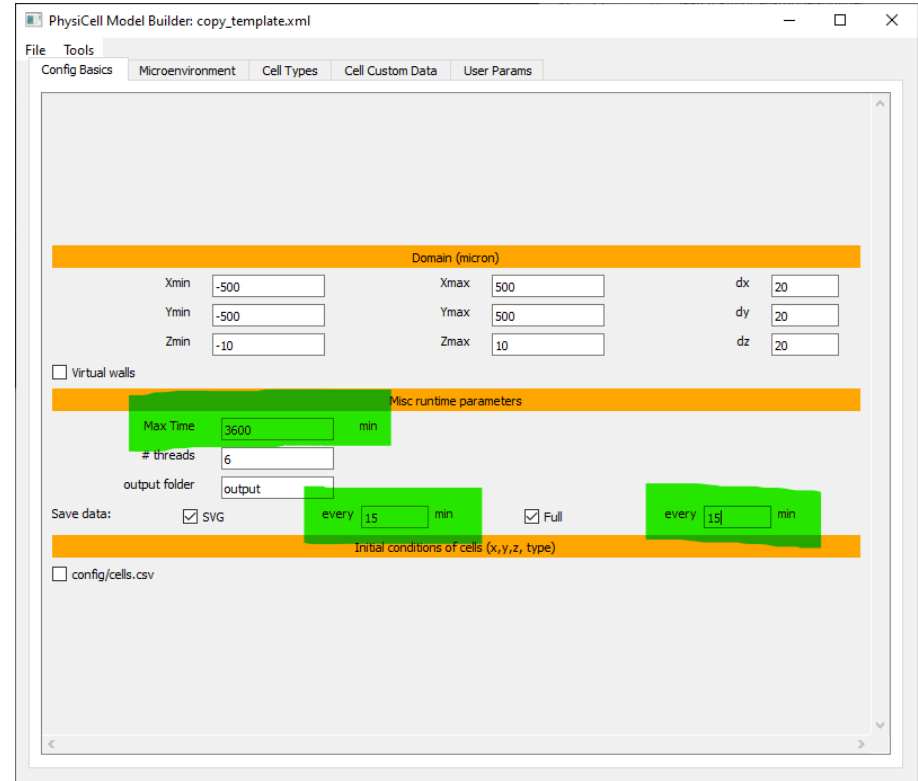
Simulation time and interval for outputs

From PhysiCell\config (anaconda)

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

Set to 3600 minutes of simulation

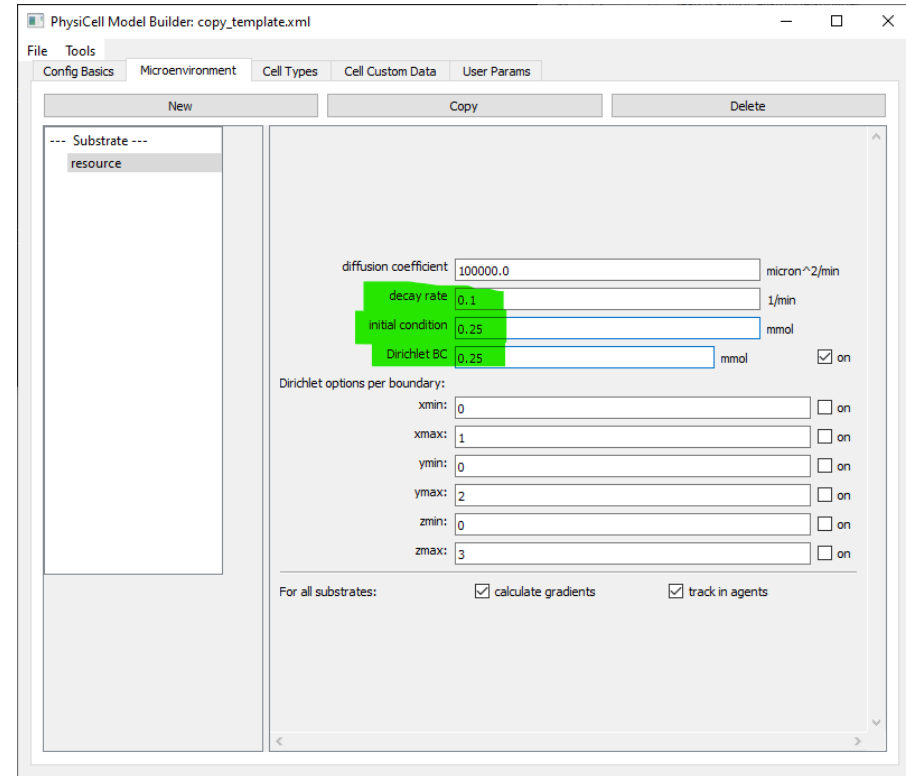
Output SVG files every 15 minutes



Changes to PhysiCell_settings.xml (1)

Set up the substrate

- Rename substrate to resource
- Change decay rate, initial condition, and boundary condition values.



Create the new cell definitions



LUDDY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

PhysiCell Project

PhysiCell.org

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

Create a definition to bacteria (math)

- We want a diffusion distance of 10, and $D = 100000 \mu m^2/min$. We need the uptake rate U :

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

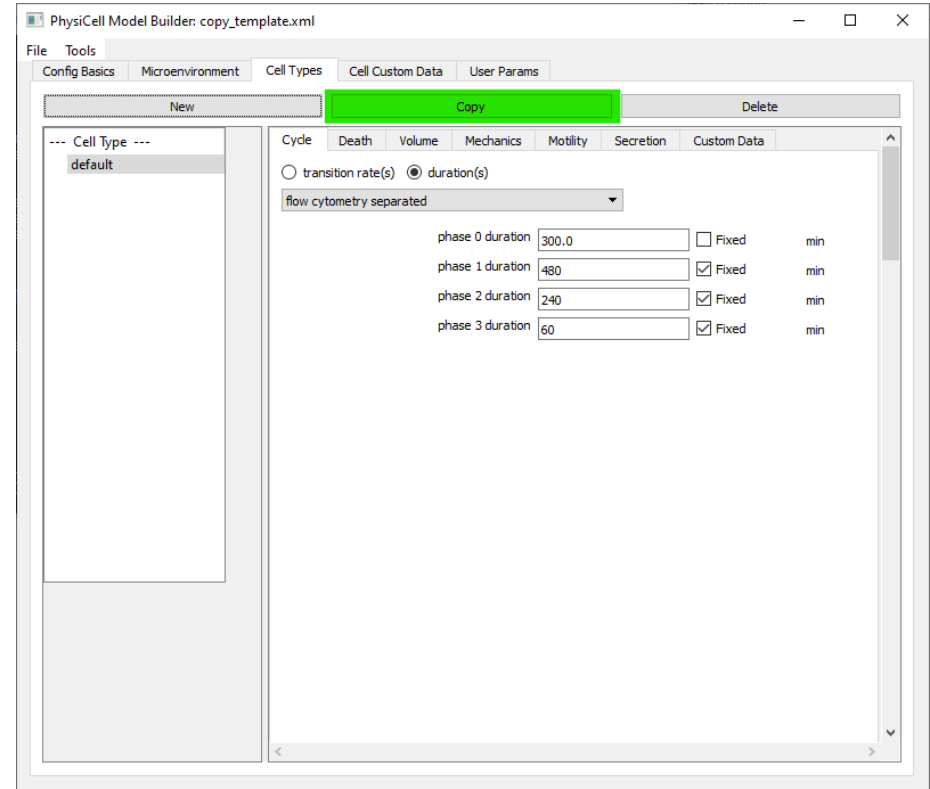
- Solving for U , we have:

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

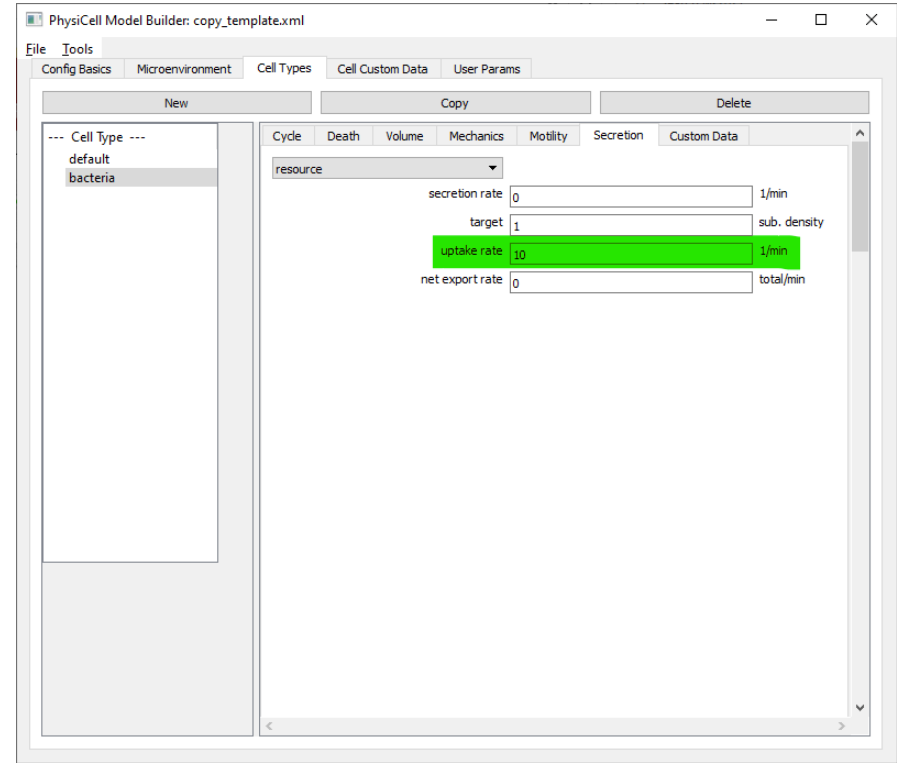
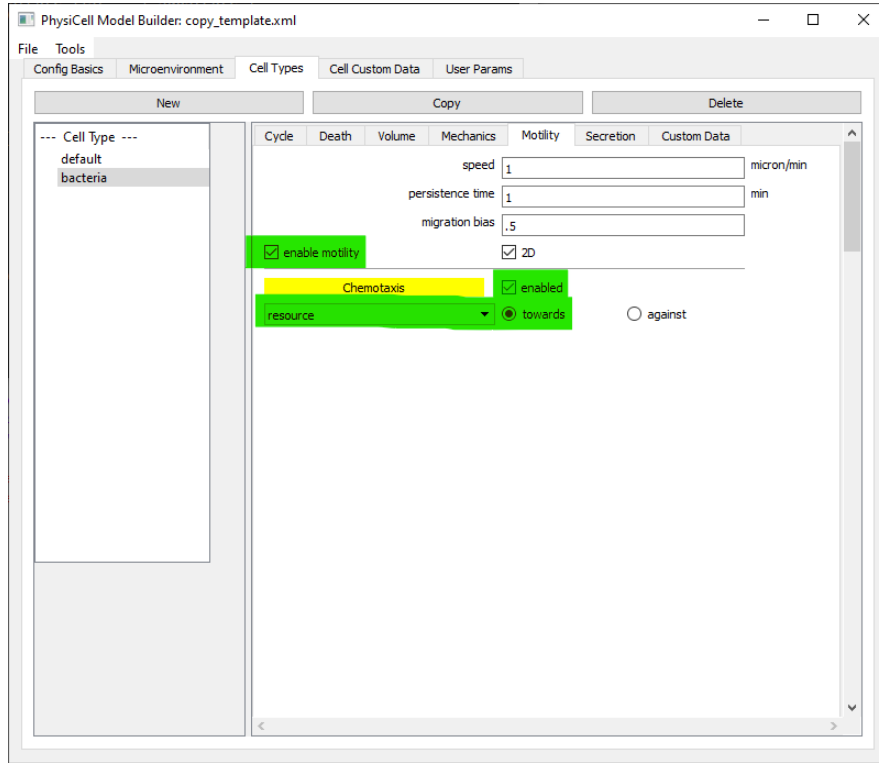
Changes to PhysiCell_settings.xml (2)

Create bacteria definition from default type

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



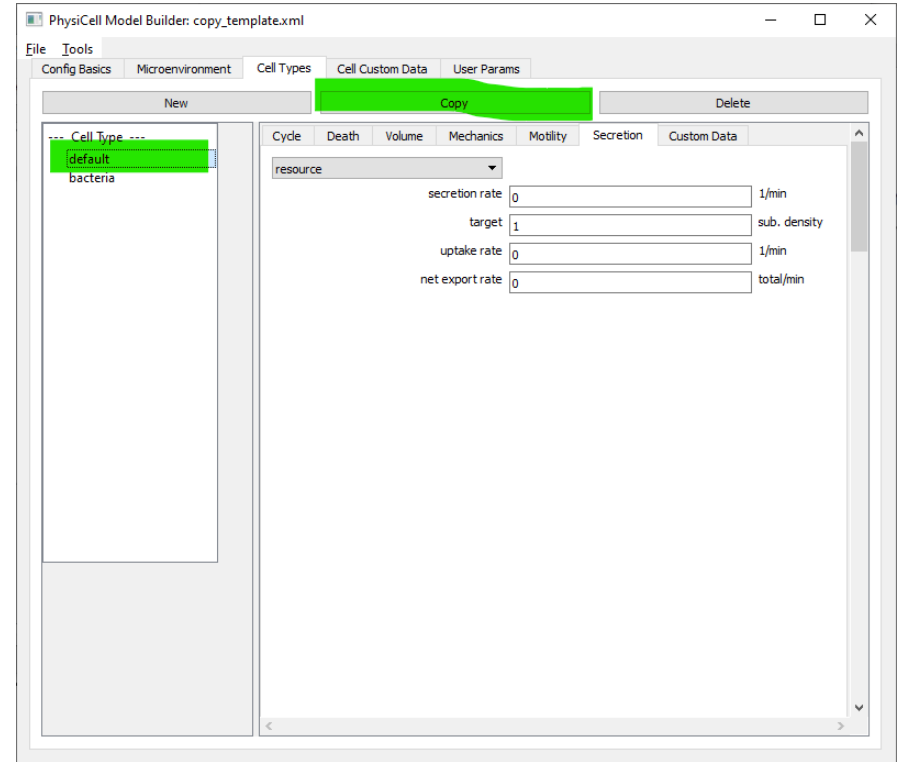
Changes to PhysiCell_settings.xml (3)



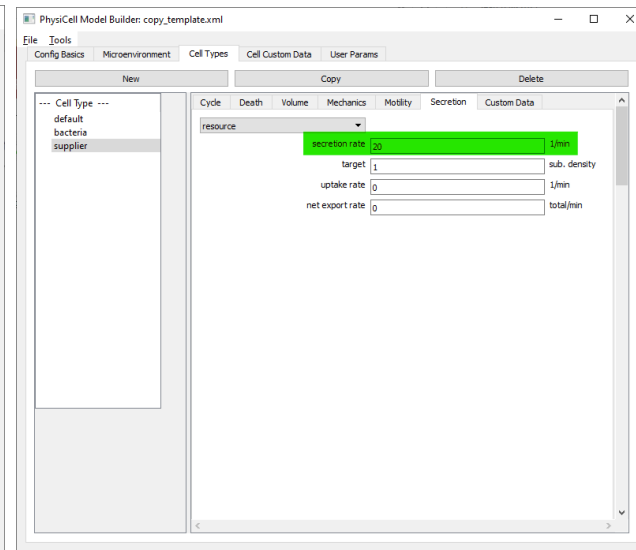
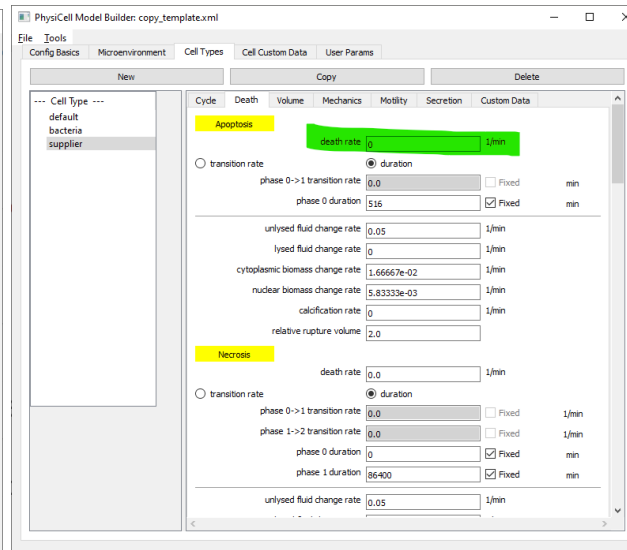
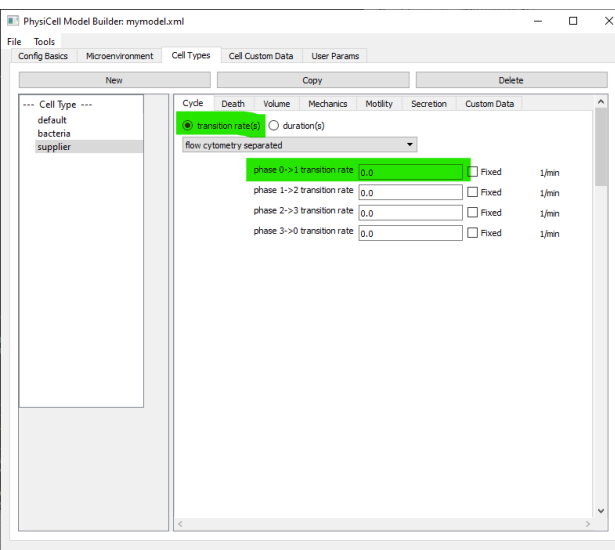
Changes to PhysiCell_settings.xml (4)

Create supplier definition from default type

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



Changes to PhysiCell_settings.xml (5)



Changes in custom.cpp



LUDDY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

PhysiCell Project

PhysiCell.org

 @PhysiCell

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



LUDDY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

PhysiCell Project

PhysiCell.org

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

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

PhysiCell Model Builder: copy_template.xml

File Tools

Config Basics Microenvironment Cell Types Cell Custom Data User Params

Append 5 more rows Clear selected rows

Name	Type	Value	Units
<input type="checkbox"/> random_seed	int	0	dimensionless
Description:			
<input type="checkbox"/> number_of_cells	int	5	none
Description: Initial number of cells (for each cell type)			
<input type="checkbox"/> number_of_bacteria	int	100	none
Description:			
<input type="checkbox"/> radius_bacteria_region	double	100	micron
Description:			
<input type="checkbox"/> number_of_suppliers	int	50	none
Description:			
<input type="checkbox"/>	double		
Description:			
<input type="checkbox"/>	double		
Description:			
<input type="checkbox"/>	double		
Description:			
<input type="checkbox"/>	double		
Description:			
<input type="checkbox"/>	double		
Description:			

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;

}
```



LUDDY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

PhysiCell Project

PhysiCell.org

 @PhysiCell

Changes to PhysiCell_settings.xml (7)

Update the default cell definition

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

PhysiCell Model Builder: copy_template.xml

File Tools

Config Basics Microenvironment Cell Types Cell Custom Data User Params

Append 5 more rows Clear selected rows

Note: changing a default value here will also change it in each Cell Type

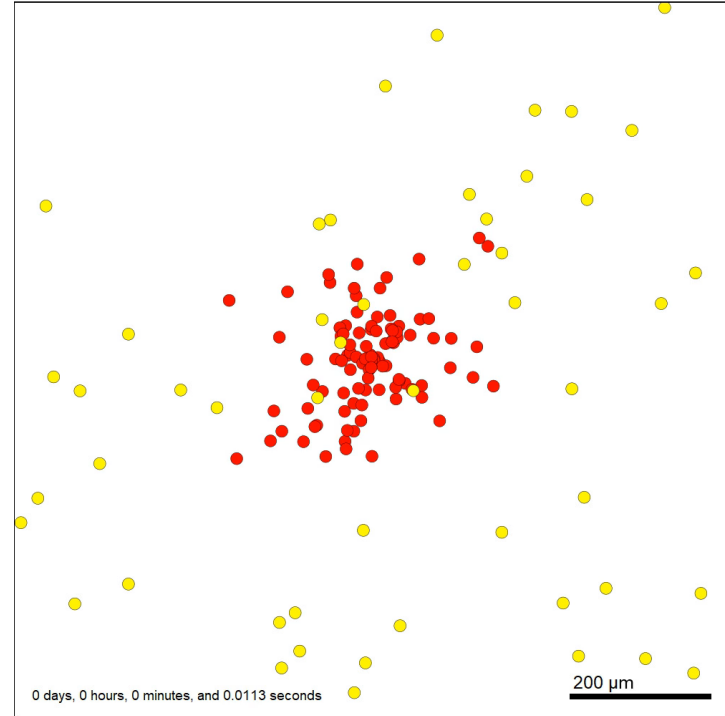
	Name (required)	Default Value (floating point)	Units
<input type="checkbox"/>	sample	1.0	dimensionless
Description:			
<input type="checkbox"/>	R_necrosis	0.15	dimensionless
Description:			
<input type="checkbox"/>	R_max_growth	0.25	dimensionless
Description:			
<input type="checkbox"/>	necrosis_rate	0.01	1/min
Description:			
<input type="checkbox"/>		0.0	
Description:			
<input type="checkbox"/>		0.0	
Description:			
<input type="checkbox"/>		0.0	
Description:			
<input type="checkbox"/>		0.0	
Description:			
<input type="checkbox"/>		0.0	
Description:			
<input type="checkbox"/>		0.0	
Description:			

Give it a try!

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

```
make jpeg
make movie
```

Current time: 0 days, 0 hours, and 0.00 minutes, z = 0.00 μm
151 agents



[Link video](#)

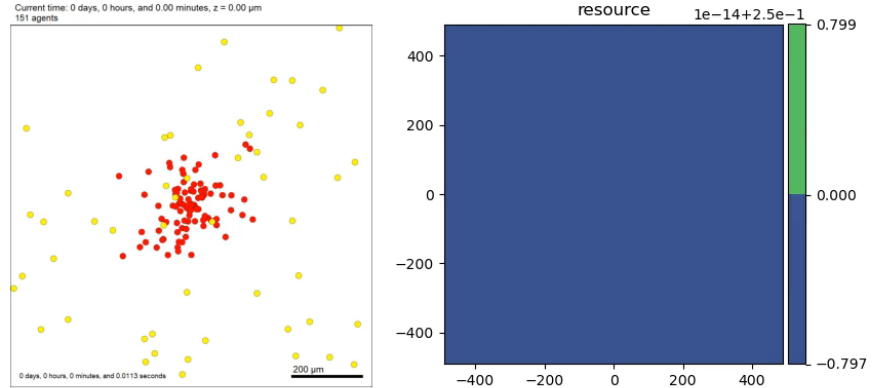
Optional visualization (advanced)

Load [plot_CellSubs.py](#) on \beta
\ script to plot cells + substrates using
pyMCDS.py

python beta\plot_CellSubs.py 0 240 1 output

Load [Makefile](#) on \PhysiCell
\ generate movie cells + substrates
make movie2

Warning: It's necessary to run 'make
jpeg' before that.



[Link video](#)

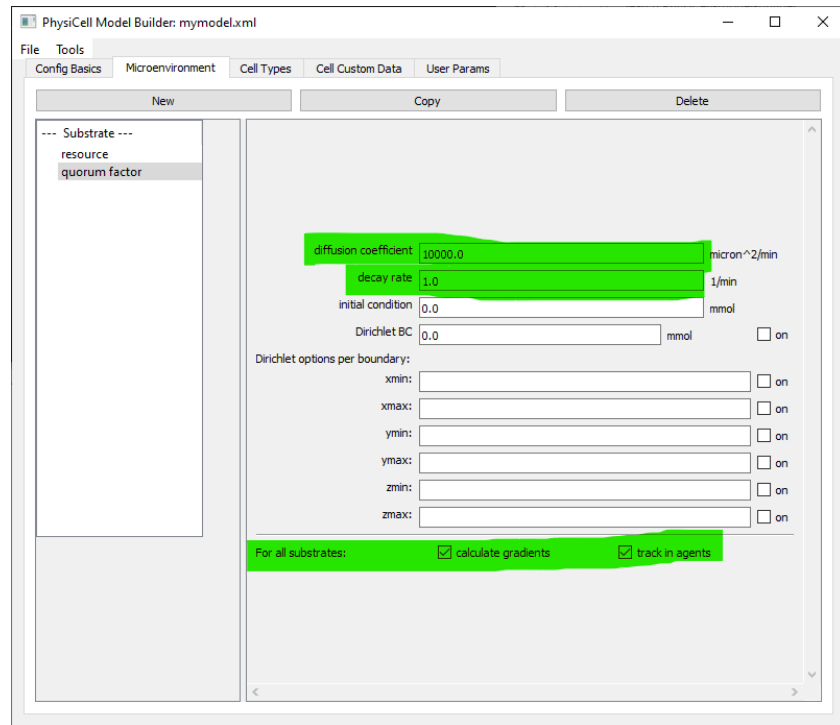
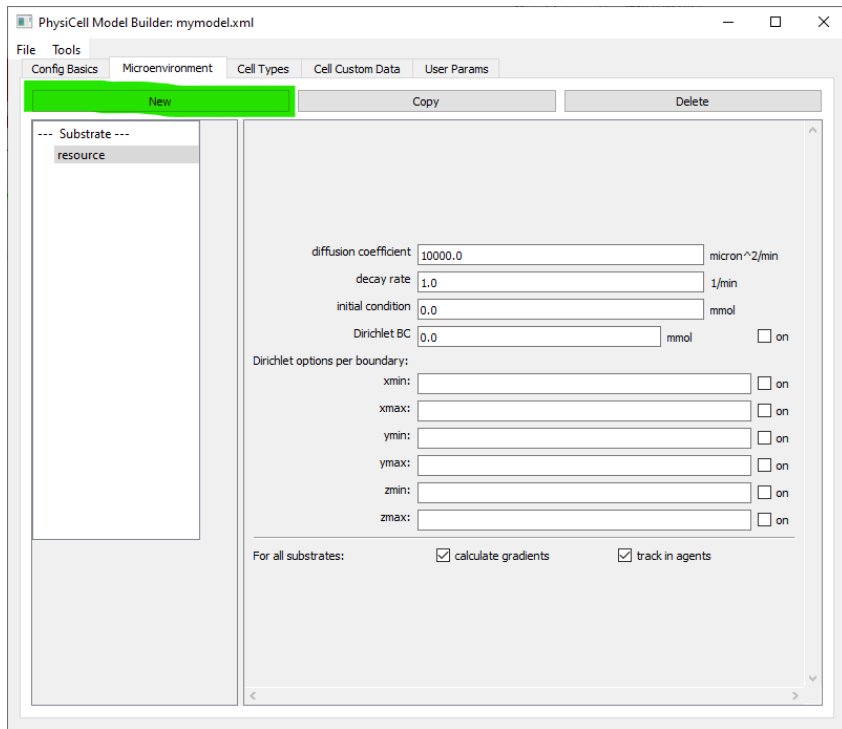
Using what we know (2) (round 2)

- 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

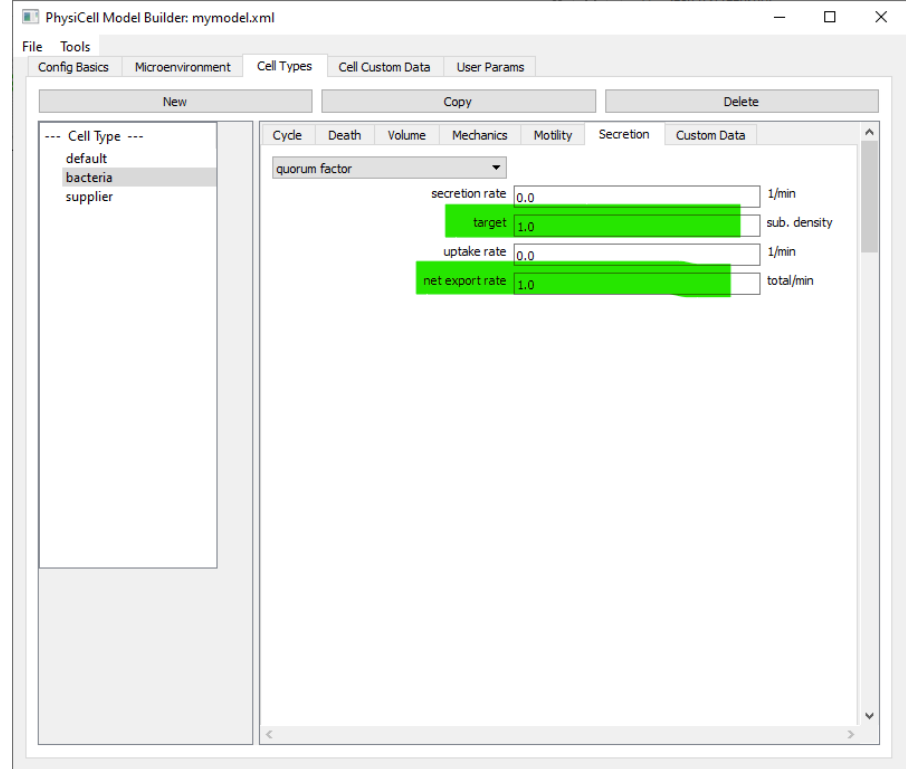
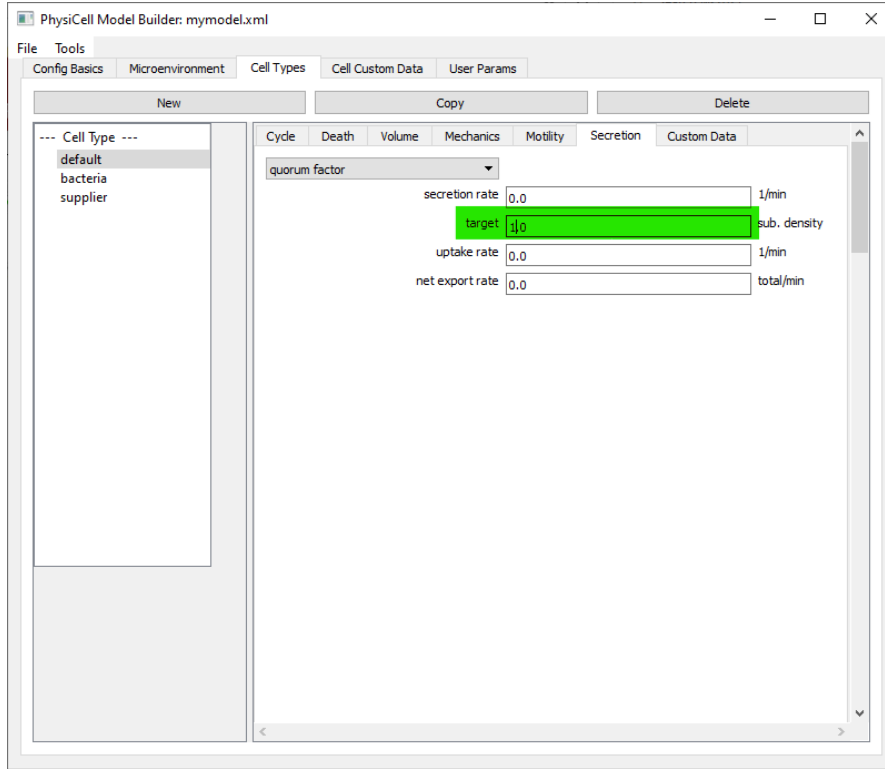
Approach

- 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

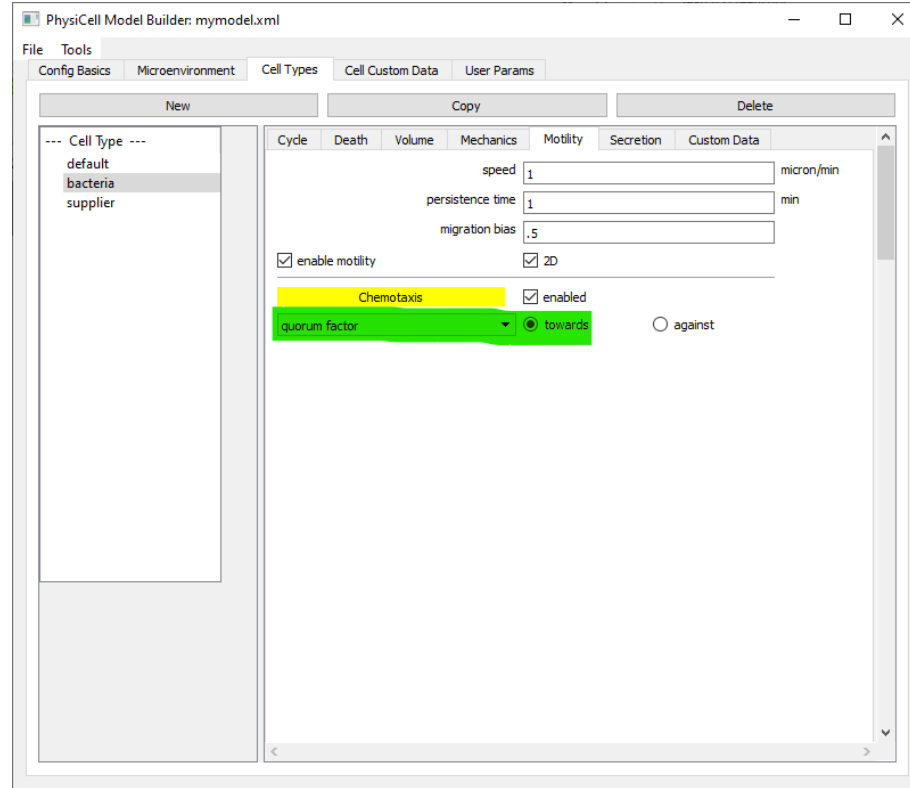
Changes to PhysiCell_settings.xml (1)



Changes to PhysiCell_settings.xml (2)



Changes to PhysiCell_settings.xml (3)

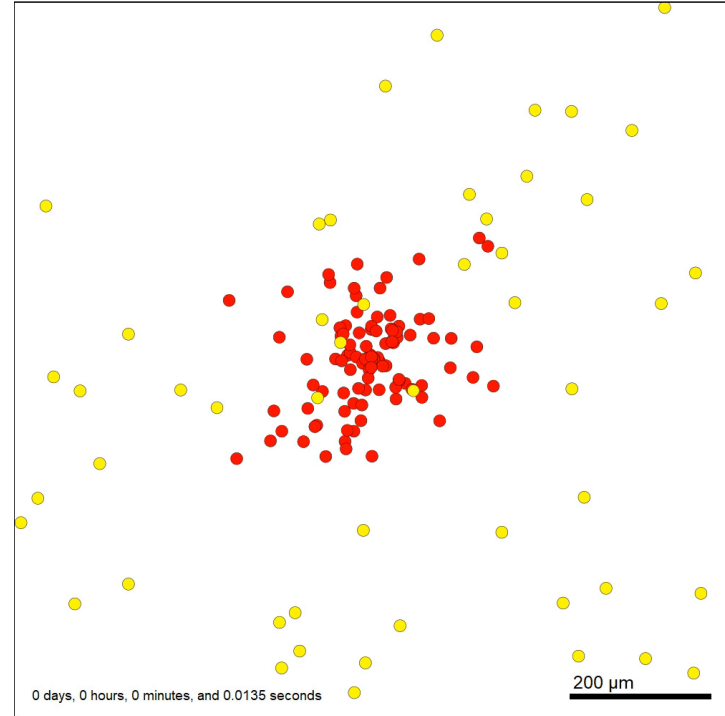


Give it a try!

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

```
make jpeg  
make movie
```

Current time: 0 days, 0 hours, and 0.00 minutes, $z = 0.00 \mu\text{m}$
151 agents



[Link video](#)

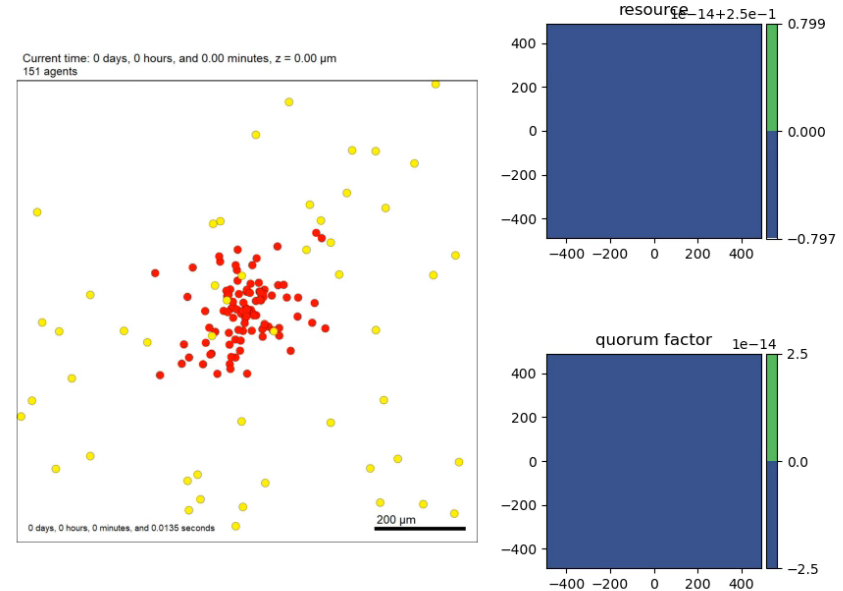
Optional visualization (advanced)

Load [plot_CellSubs.py](#) on \beta
\\ script to plot cells + substrates using
pyMCDS.py

python beta\plot_CellSubs.py 0 240 1 output

Load [Makefile](#) on \PhysiCell
\\ generate movie cells + substrates
make movie2

Warning: It's necessary to run 'make
jpeg' before that.



[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 $\max\left(0, 1 - \frac{q}{q_{mot}}\right)$

Changes to bacteria phenotype (1)

```
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" );

    // find the index of quorum factor
    static int nQ = microenvironment.find_density_index( "quorum factor" );

    // 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];
    double q = pCell->nearest_density_vector()[nQ];

    // ...
}
```

Changes to bacteria phenotype (2)

```
// ...

// 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;
}
```



LUDDY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

PhysiCell Project

PhysiCell.org

 @PhysiCell

Changes to PhysiCell_settings.xml

Update the default cell definition

- Include to the custom data:
quorum_motility_slowdown

The screenshot shows the 'PhysiCell Model Builder: mymodel.xml' window. The 'Cell Types' tab is selected. A table lists cell parameters with columns for Name, Default Value, and Units. The 'quorum_motility_slowdown' row is highlighted in green.

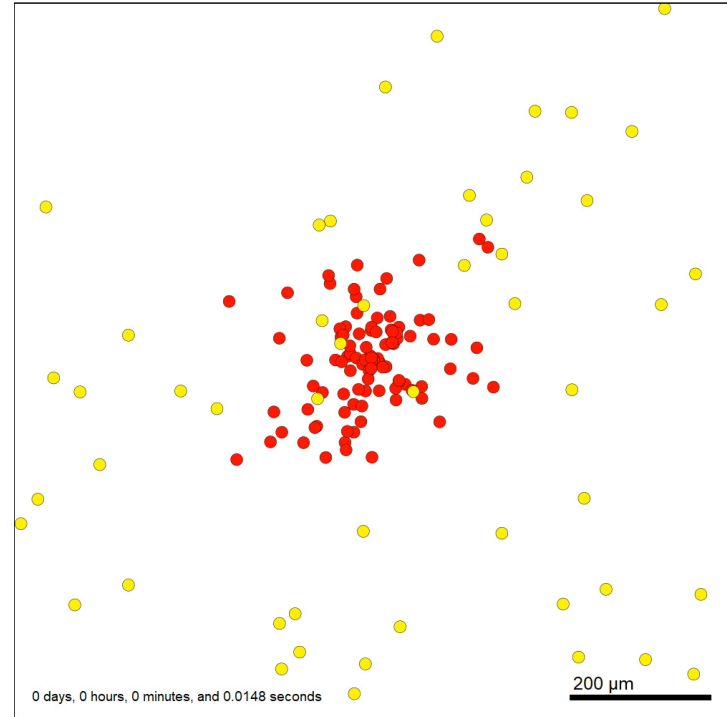
Name (required)	Default Value (floating point)	Units
<input type="checkbox"/> sample	1.0	
Description:		
<input type="checkbox"/> R_necrosis	0.15	
Description:		
<input type="checkbox"/> R_max_growth	0.25	
Description:		
<input type="checkbox"/> necrosis_rate	0.01	
Description:		
<input type="checkbox"/> quorum_motility_slowdown	1e-4	
Description:		
<input type="checkbox"/>	0.0	
Description:		
<input type="checkbox"/>	0.0	
Description:		
<input type="checkbox"/>	0.0	
Description:		
<input type="checkbox"/>	0.0	
Description:		
<input type="checkbox"/>	0.0	
Description:		

Give it a try!

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

```
make jpeg  
make movie
```

Current time: 0 days, 0 hours, and 0.00 minutes, z = 0.00 μm
151 agents



[Link video](#)

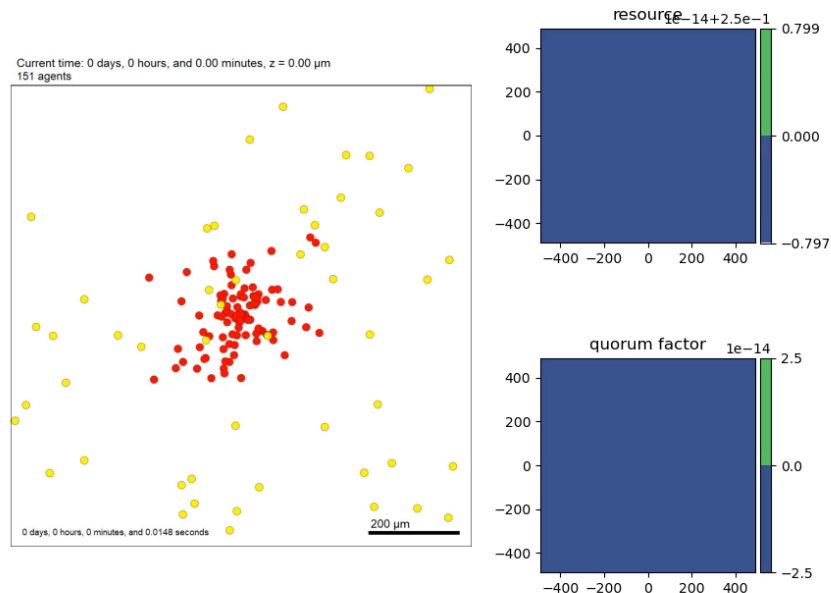
Optional visualization (advanced)

Load [plot_CellSubs.py](#) on \beta
\ script to plot cells + substrates using
pyMCDS.py

python beta\plot_CellSubs.py 0 240 1 output

Load [Makefile](#) on \PhysiCell
\ generate movie cells + substrates
make movie2

Warning: It's necessary to run 'make
jpeg' before that.



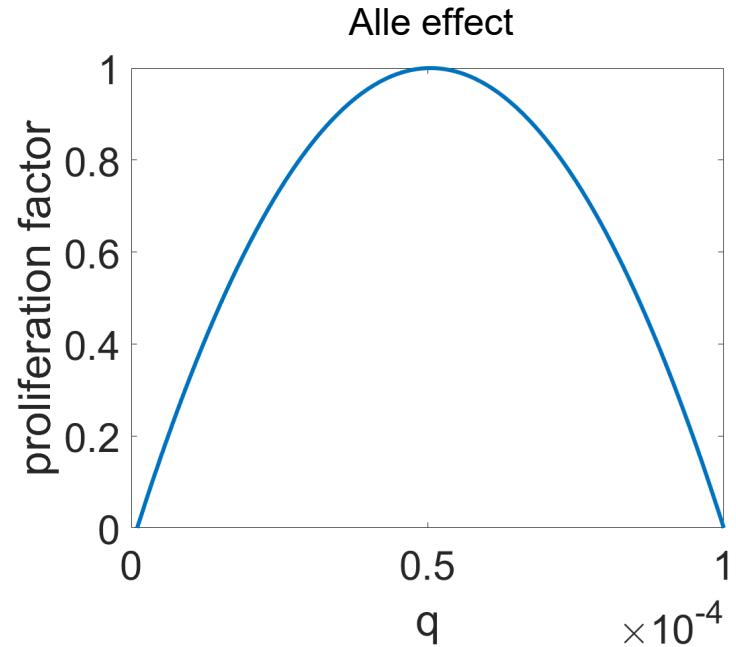
[Link video](#)

Using what we know (2) (round 4)

- Let's use two diffusing substrates:
 - resource (Dirichlet condition 0.25)
 - Quorum factor (Neumann condition)
- **Cell type "bacteria":**
 - Proliferate proportional to resource
 - **Low proliferation when q is low (too far from colony) or q is too high (overcrowding)**
 - 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_low_proliferation (default this to 1e-6)
 - ♦ quorum_high_proliferation(default this to 1e-4)
- In custom.cpp
 - In bacteria_phenotype()
 - ♦ proliferation is zero if $q < q_L$ or if $q > q_H$
 - ♦ scale proliferation by $\frac{4}{(q_H - q_L)^2} (q - q_L)(q_H - q)$



Changes to bacteria_phenotype()

```
// ...
// 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; }

// scale with quorum factor
double Qlow = pCell->custom_data["quorum_low_prolif"];
double Qhigh = pCell->custom_data["quorum_high_prolif"];
double constant = 4.0 / pow( Qhigh - Qlow , 2.0 );

// no proliferation if Q is low or high
if( q < Qlow || q > Qhigh )
{ scaling_factor = 0.0; }

scaling_factor *= ( constant*(q-Qlow)*(Qhigh-q) );

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



LUDDY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

PhysiCell Project

PhysiCell.org

 @PhysiCell

Changes to PhysiCell_settings.xml

Update the default cell definition

- Include to the custom data:
quorum_low_prolif and
quorum_high_prolif

PhysiCell Model Builder: mymodel.xml

File Tools

Config Basics Microenvironment Cell Types Cell Custom Data User Params

Append 5 more rows Clear selected rows

Note: changing a default value here will also change it in each Cell Type

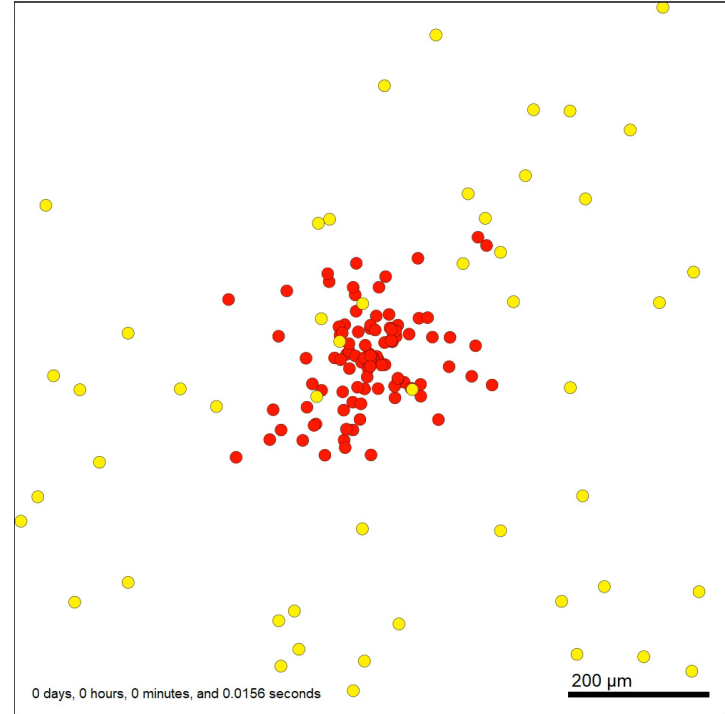
	Name (required)	Default Value (floating point)	Units
<input type="checkbox"/>	sample	1.0	
	Description:		
<input type="checkbox"/>	R_necrosis	0.15	
	Description:		
<input type="checkbox"/>	R_max_growth	0.25	
	Description:		
<input type="checkbox"/>	necrosis_rate	0.01	
	Description:		
<input type="checkbox"/>	quorum_motility_slowdown	1e-4	
	Description:		
<input type="checkbox"/>	quorum_low_prolif	1e-6	
	Description:		
<input type="checkbox"/>	quorum_high_prolif	1e-4	
	Description:		
<input type="checkbox"/>		0.0	
	Description:		
<input type="checkbox"/>		0.0	
	Description:		
<input type="checkbox"/>		0.0	
	Description:		

Give it a try!

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

```
make jpeg  
make movie
```

Current time: 0 days, 0 hours, and 0.00 minutes, z = 0.00 μm
151 agents



[Link video](#)

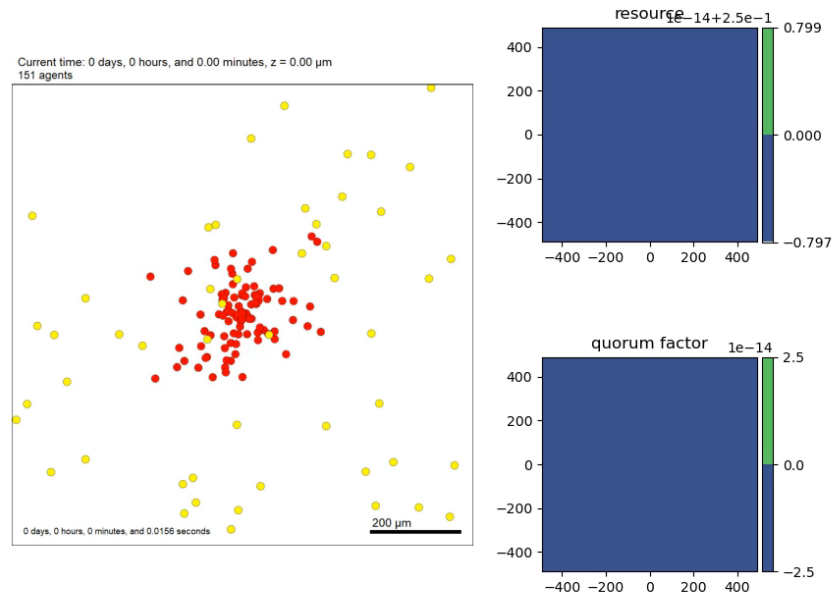
Optional visualization (advanced)

Load [plot_CellSubs.py](#) on \beta
\ script to plot cells + substrates using
pyMCDS.py

python beta\plot_CellSubs.py 0 240 1 output

Load [Makefile](#) on \PhysiCell
\ generate movie cells + substrates
make movie2

Warning: It's necessary to run 'make
jpeg' before that.



[Link video](#)

Using what we know (2) (round 5)

- Let's use two diffusing substrates:
 - resource (Dirichlet condition 1)
 - quorum factor (Neumann condition)
- **Cell type "bacteria":**
 - Proliferate proportional to resource
 - Low proliferation when q is low (too far from colony) or q is too high (overcrowding)
 - Die if resource is below a threshold
 - Secrete q
 - Chemotax towards regions of high q
 - Slow down motility when q is high
 - **Increase migration bias as q increases (random wandering when far from colonies)**
- **Agent type "supplier":**
 - Don't proliferate or die.
 - Can't be moved
 - Release resource

Approach

- In `custom.cpp`
 - In `bacteria_phenotype()`
 - ♦ scale motility bias by $\frac{q}{q_m}$

Changes to bacteria_phenotype()

```
// get the cell line's motile speed
phenotype.motility.migration_speed = pCD->phenotype.motility.migration_speed;
// get a scaling factor
scaling_factor = 1.0 - 1 / pCell->custom_data["quorum_motility_slowdown"];
if( scaling_factor < 0.0 )
{ scaling_factor = 0.0; }
// scale migration speed
phenotype.motility.migration_speed *= scaling_factor;

// scale migration bias by the quorum factor
scaling_factor = q / pCell->custom_data["quorum_motility_slowdown"];
if( scaling_factor > 1.0 )
{ scaling_factor = 1.0; }
phenotype.motility.migration_bias = pCD->phenotype.motility.migration_bias;
phenotype.motility.migration_bias *= scaling_factor;

return;
}
```



LUDDY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

PhysiCell Project

PhysiCell.org

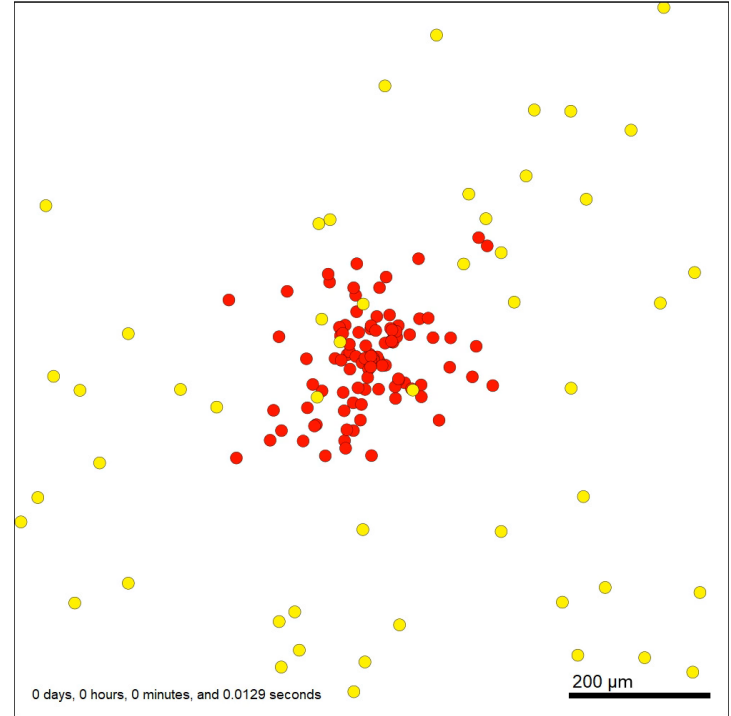
 @PhysiCell

Give it a try!

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

```
make jpeg  
make movie
```

Current time: 0 days, 0 hours, and 0.00 minutes, z = 0.00 μm
151 agents



[Link video](#)

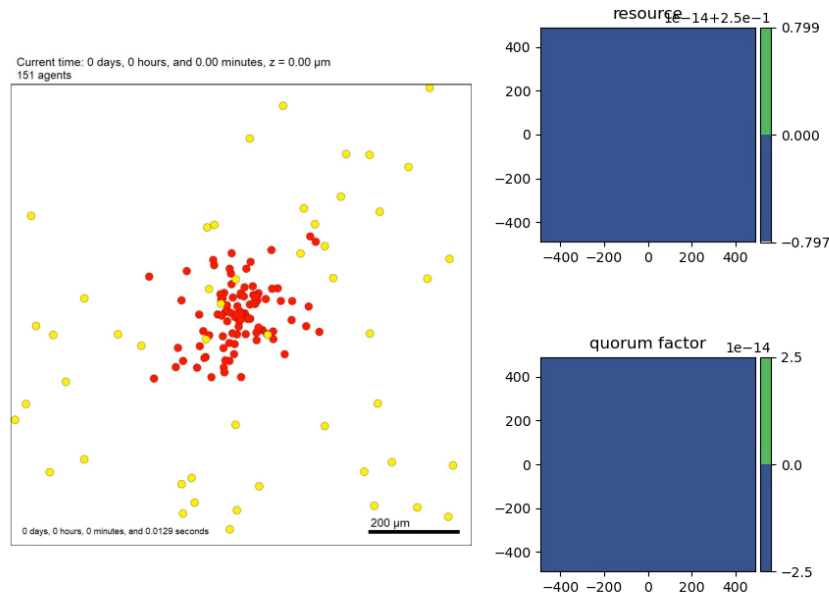
Optional visualization (advanced)

Load [plot_CellSubs.py](#) on \beta
\ script to plot cells + substrates using
pyMCDS.py

python beta\plot_CellSubs.py 0 240 1 output

Load [Makefile](#) on \PhysiCell
\ generate movie cells + substrates
make movie2

Warning: It's necessary to run 'make
jpeg' before that.



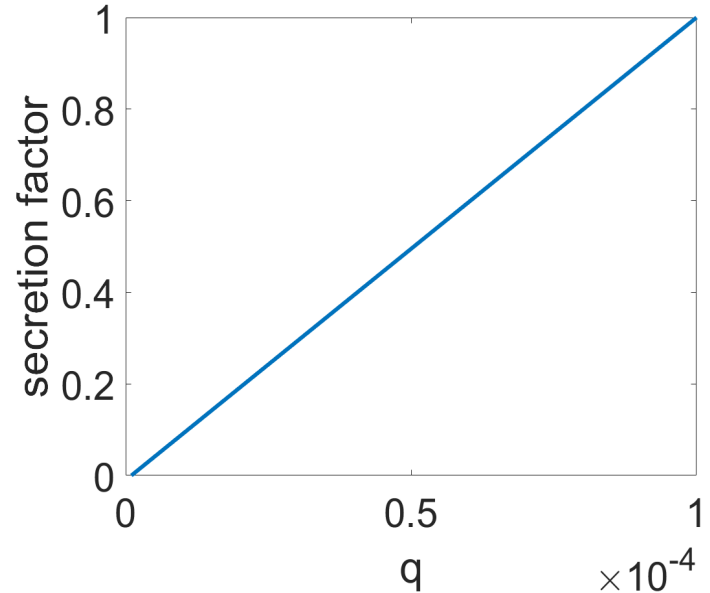
[Link video](#)

Using what we know (2) (round 6)

- Let's use two diffusing substrates:
 - resource (Dirichlet condition 1)
 - quorum factor (Neumann condition)
- **Cell type "bacteria":**
 - Proliferate proportional to resource
 - Low proliferation when q is low (too far from colony) or q is too high (overcrowding)
 - Die if resource is below a threshold
 - Secrete q
 - Chemotax towards regions of high q
 - Slow down motility when q is high
 - Increase migration bias as q increases (random wandering when far from colonies)
- **Agent type "supplier":**
 - Don't proliferate or die.
 - Can't be moved
 - Release resource
 - **resource release increases when q is high (dynamic response to "needs" of nearby colony)**

Approach

- In PhysiCell_settings.xml
 - Create new custom cell data:
 - ♦ max_R_release_rate, min_R_release_rate
 - ♦ q_max_R_release , q_min_R_release
- In custom.cpp / custom.h
 - Create supplier_phenotype()
 - Linear R release rate: $r_{\text{low}} + (r_{\text{hi}} - r_{\text{low}}) \frac{q - q_{\text{low}}}{q_{\text{hi}} - q_{\text{low}}}$
 - Make sure supplier use this function



Use the function (create_cell_types())

```
// ...

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

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

Cell_Definition* pSupplier = find_cell_definition( "supplier" );
pSupplier->functions.update_phenotype = supplier_phenotype;

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

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

return;

}
```



LUDDY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

Supplier phenotype

```
void supplier_phenotype( Cell* pCell, Phenotype& phenotype , double dt )
{
    // 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" );

    // find the index of quorum factor
    static int nQ = microenvironment.find_density_index( "quorum factor" );

    // sample microenvironment at cell position to get resource and quorum factor
    double q = pCell->nearest_density_vector()[nQ];

    double Qlow  = pCell->custom_data["quorum_low_R_release"];
    double Qhigh = pCell->custom_data["quorum_high_R_release"];
    double Rlow  = pCell->custom_data["low_R_release"];
    double Rhigh = pCell->custom_data["high_R_release"];

    if( q < Qlow )
    { phenotype.secretion.secretion_rates[nR] = Rlow; return; }

    if( q > Qhigh )
    { phenotype.secretion.secretion_rates[nR] = Rhigh; return; }

    double scaling_factor = (q-Qlow)/(Qhigh-Qlow);
    phenotype.secretion.secretion_rates[nR] = Rlow + (Rhigh-Rlow)*scaling_factor;

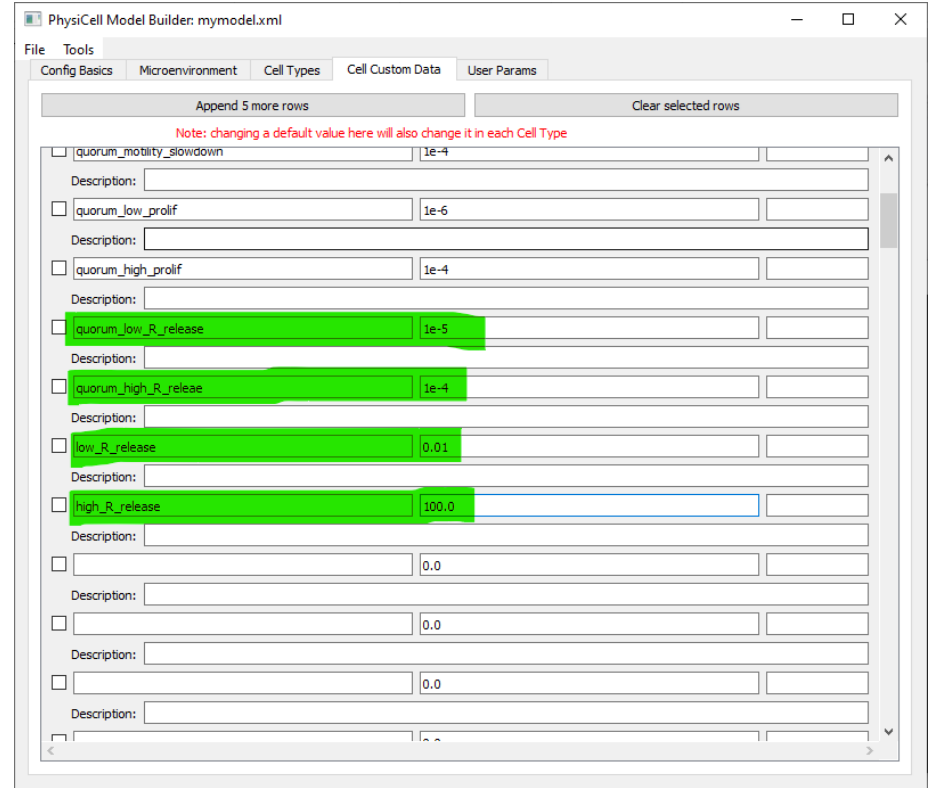
    return;
}

Add void supplier_phenotype( Cell* pCell, Phenotype& phenotype , double dt ) to custom.h
```

Changes in PhysiCell_settings.xml

Update the default cell definition

- Include to the custom data:
quorum_low_R_release,
quorum_high_R_release,
low_R_release, and
high_R_release



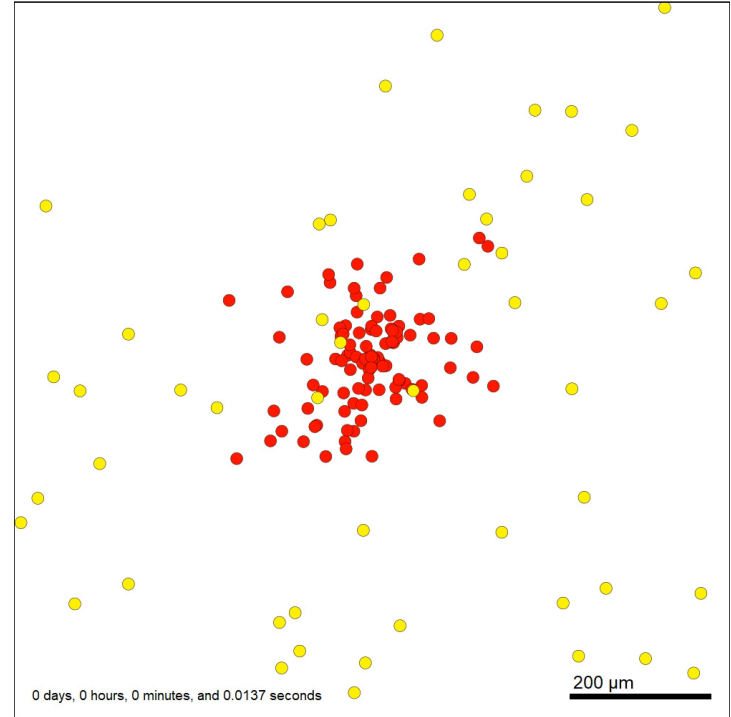
Give it a try!

```
cp ../PhysiCell-model-builder/mymodel.xml  
./config/mymodel_6.xml
```

```
make data-cleanup  
make  
./project ./config/mymodel_6.xml
```

```
make jpeg  
make movie
```

Current time: 0 days, 0 hours, and 0.00 minutes, z = 0.00 μm
151 agents



[Link video](#)

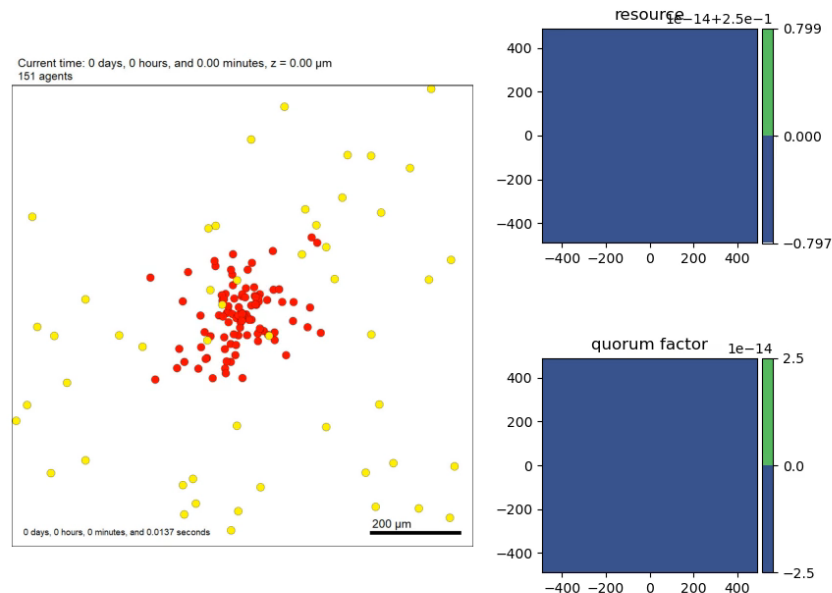
Optional visualization (advanced)

Load [plot_CellSubs.py](#) on \beta
\ script to plot cells + substrates using
pyMCDS.py

python beta\plot_CellSubs.py 0 240 1 output

Load [Makefile](#) on \PhysiCell
\ generate movie cells + substrates
make movie2

Warning: It's necessary to run 'make
jpeg' before that.



[Link video](#)

Mathematics: Hill Functions

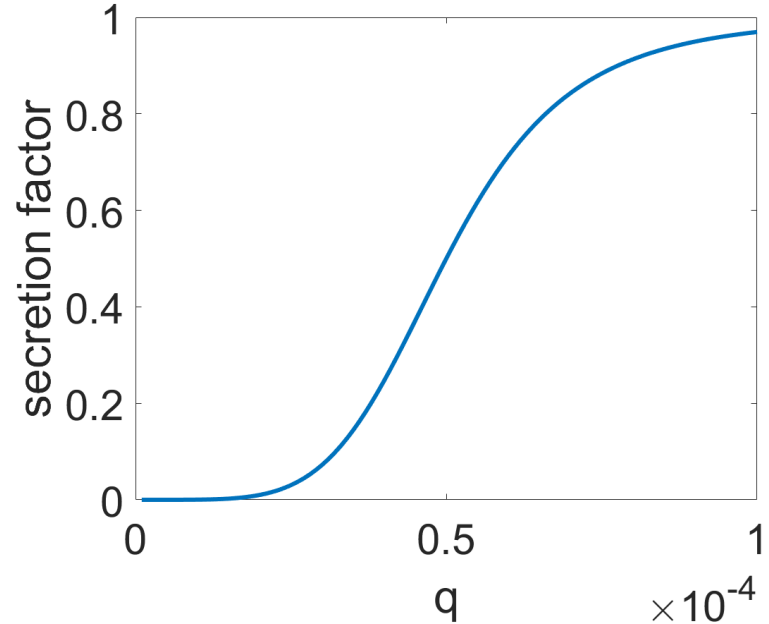
$$H(q) = \frac{q^n}{h^n + q^n}$$

Hill power

half-max

Changes to custom.cpp

```
if (q < Qlow)  
{  
phenotype.secretion.secretion_rates[nR] = Rlow;  
return;  
  
if (q > Qhigh)  
{  
phenotype.secretion.secretion_rates[nR] = Rhigh;  
return;  
    double h = Qlow/2  
    double scaling_factor =  
    pow(q, 5) / (pow(h, 5) + pow(q, 5));  
    phenotype.secretion.secretion_rates[nR]  
    = Rlow + (Rhigh - Rlow) * scaling_factor;  
    return;  
}
```



3-Types model (Intermediate homework [optional])

- In physics, the **3-body problem** shows how 3 objects with very simple interactions (gravitation) can demonstrate chaotic behavior.
- **Let's build a similar system for biology!**
- **3 cell types** (A,B,C) each secrete their own chemical factor
 - **visualization**: assume each cell fluoresces proportionally to its signal
- Each cell type can:
 - **divide** and **die** in response to resource (R), A, B, C, and pressure
 - **move** in response to A, B, C, and R
 - **secrete** (or not secrete) in response to A, B, C, and R
- *What can happen in this general system?*



Try this model yourself!

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

Mathematics: Hill Functions

- Let's define a **Hill function** $H(s)$:

$$H(s) = \frac{s^n}{s^n + h^n}$$

Hill power

- Let's write:

U = sum of promoter signals

D = sum of inhibitor signals

half-max

- We can use these signals to control a parameter p via a Hill function:

$$p = \left[p_0 + \overbrace{(p_{\max} - p_0)H(U)}^{p \rightarrow p_{\max} \text{ as } U \rightarrow \infty} \right] \left[\overbrace{1 - H(D)}^{p \rightarrow 0 \text{ as } D \rightarrow \infty} \right]$$

Exercise: tumor competition

- Let's look at 2 tumor sub-clones:
- Population A:
 - Regular proliferation and death
- Population B:
 - Increased proliferation
 - Cost: more sensitive to resource depletion

Build the model

- Cell Type A:
 - Use default parameters
- Cell Type B:
 - double B_base_cycle to 0.00144
 - double B_max_cycle to 0.0144
 - increase B_necrosis_threshold to 0.6
- set to no Type C cells
 - number_of_C = 0

Run the model

- How do pink (Type A) and green (Type B) cells compete in high-resource regions?
- How do pink (Type A) and green (Type B) cells compete in low-resource regions?

Exercise: tumor co-option of stromal cells

- Let's look at two interacting cell populations:
 - Tumor cells attract stromal cells and "convince" them to secrete a growth factor.
- Population A (tumor):
 - Secretes signal A
 - Signal B **promotes** proliferation
 - No proliferation without signal B
- Population B (stromal):
 - No proliferation without signal A
 - Chemotaxis towards Signal A (and stops in regions of high signal A)
 - Signal A **promotes** secretion of Signal B

Build the model (1)

- Cell Type A:
 - Cycling
 - ♦ $A_base_cycle = 0$ (no proliferation without signal B)
 - ♦ $A_max_cycle = 0.00072$ (slow the kinetics down a bit)
 - ♦ $A_cycle_B = promote$ (signal B enables proliferation)
- Cell Type B:
 - Cycling
 - ♦ $B_base_cycle = 0$
 - ♦ $B_max_cycle = 0.000072$ (less proliferation than tumor cells)
 - ♦ B_cycle_A (signal A promotes proliferation)
 - Secretion
 - ♦ $B_base_secretion = 0$ (no secretion without signal A)
 - ♦ $B_signal_A = promote$ (signal A stimulates secretion)
 - Motility
 - ♦ $B_speed_base = 0.5$ (increase the base speed a bit)
 - ♦ $B_speed_A = inhibit$ (slow down when you reach the tumor)

Build the model (2)

- Now, switch to the cell types tab, and select Type B
- Let's turn on chemotaxis
- Cell Type B:
 - phenotype: motility
 - ♦ enabled = on (turn on motility)
 - ♦ chemotaxis:
 - » enabled = on (use chemotaxis to guide migration)
 - » substrate = signal A (chemotaxis towards tumor cells)
 - » direction = 1 (move up the gradient towards stronger signal A)

Run the model

- Set 25 initial type A cells, and 1 initial type B cell. 0 Type C cells.
 - Where do green (type B) cells end up?
 - Where do you see the most pink cells (Type A)?
- Increase to 5 initial type B cells.
 - Where do green (type B) cells end up?
 - Where do you see the most pink cells (Type A)?
- Increase B_cycle_max to 0.00018. What happens?
- Set 0 initial Type B cells. What happens?

Exercise: a population with a quorum factor

- Cells can use ***quorum factors*** to find each other and sense population size.
- Population A (bacteria):
 - Secretes signal A
 - Signal A **promotes** proliferation
 - Movement towards signal A (help for aggregation)
 - Signal A **inhibits** migration (they stop moving when they find their home)

Build the model

- Cell Type A:
 - Cycling
 - ♦ $A_base_cycle = 0.0$ (no proliferation without signal B)
 - ♦ $A_max_cycle = 0.0072$ (fast birth once aggregated)
 - ♦ $A_cycle_A = promote$ (signal B enables proliferation)
 - secretion
 - ♦ $A_signal_A = promote$
 - motility
 - ♦ $A_base_speed = 1$
 - ♦ $A_max_speed = 1$
 - ♦ $A_speed_A = inhibit$
- Turn on chemotaxis towards signal A as before in the "cell types" tab

Run the model

- Set 25 initial type A cells, and 0 Type B cells. 0 Type C cells.
 - What happens?
- Set 5 initial type A cells, and 0 Type B cells. 0 Type C cells.
 - What happens?
- Set 1 initial type A cells, and 0 Type B cells. 0 Type C cells.
 - What happens?

Exercise: add attackers

- Cells can use **quorum factors** to find each other and sense population size.
- Population A (bacteria):
 - Secretes signal A
 - Signal A **promotes** proliferation
 - Signal B **promotes** death (it's a poison)
 - Movement towards signal A (help for aggregation)
 - Signal A **inhibits** migration (they stop moving when they find their home)
- Population B (attackers):
 - Migration towards signal A
 - No proliferation or death
 - Signal A **promotes** secretion of signal B
 - Signal A **inhibits** migration (they stop moving when they find their target)

Build the model

- Cell Type A:
 - death
 - ♦ A_death_B = promote (Type B signal kills it)
 - ♦ A_max_death 0.005 (B signal increases death)
- Cell Type B:
 - Cycling
 - ♦ B_base_cycle = 0.0
 - ♦ B_max_cycle = 0
 - death
 - ♦ B_base_death = 0
 - ♦ B_max_death = 0
 - secretion
 - ♦ B_signal_A = promote
 - motility
 - ♦ B_base_speed = 1
 - ♦ B_max_speed = 1
 - ♦ B_speed_A = inhibit
- Turn on chemotaxis towards signal A as before in the "cell types" tab

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)

Training Materials:

- Administrative supplement to NCI U01CA232137 (Year 2)