PhysiCell Tutorial: more stuff

Paul Macklin, Ph.D.

Intelligent Systems Engineering
Indiana University

March 6, 2023

Thank you to our sponsors!

- Jayne Koskinas Ted Giovanis Foundation for Health and Policy
- National Cancer Institute (U01CA232137)
- Administrative supplement to NCI U01CA232137 (Year 2)
- National Science Foundation (1720625, 1818187)
- NCI / DOE / Frederick National Lab for Cancer Research (21X126F)
- DOD / Defense Threat Reduction Agency (HDTRA12110015)
- NIH Common Fund (3OT2OD026671-01S4)

Goals

- Key background
  - signals
  - behaviors
  - PhysiCell functions
- Introduce full modeling workflow
- sample project: fibrosis

Key Background

Signal Dictionary

- [substrate X]
- intracellular [substrate X]
- [substrate X] gradient
- pressure
- volume
- contact with [cell type X]
- contact with live cell
- contact with dead cell
- contact with basement membrane (not yet implemented)
- damage
- dead
- total attack time
- time
- custom:[variable name]
- apoptotic
- necrotic

Accessing signals

- A simple API allows us to access the human-interpretable behaviors

```
get_single_signal( pCell , signal_name )
```

- Examples:

```
get_single_signal( pCell , "oxygen" );
get_single_signal( pCell , "pressure" );
get_single_signal( pCell , "contact with melanocyte" );
```

- We can set/get vectors of parameters with a similar API.

Dictionary of Behaviors

Behavior Dictionary

- [substrate X] secretion
- [substrate X] secretion target
- [substrate X] uptake
- [substrate X] export
- cycle entry
- exit from cycle phase [1 to 5]
- apoptosis
- necrosis
- migration speed
- migration bias
- migration persistence time
- chemotactic response to [substrate X]
- cell-cell adhesion
- cell-cell adhesion elastic constant
- adhesive affinity to cancer cell
- adhesive affinity to [cell typeX]
- relative maximum adhesion distance
- cell-cell repulsion
- cell-BM adhesion
- cell-BM repulsion
- phagocytose dead cell
- phagocytose [cell type X]
- attack [cell type X]
- immunogenicity to [cell type X]
- transform to [cell type X]
- custom:[variable V]
- is_movable. **warning! fragile**

Accessing behaviors

- A simple API allows us to access the human-interpretable behaviors

```
set_single_behavior( pCell , behavior_name , behavior_value )
```

•Examples:

```
set_single_behavior( pCell , "cycle entry" , 0.001);
set_single_behavior( pCell , "phagocytose dead cell" , 0.01);
set_single_behavior( pCell , "secrete TGB-beta" , 10);
```

- We can also access reference values (from a cell definition)

```
get_single_base_behavior( pCell , behavior_name );
```

- We can set/get vectors of parameters with a similar API.

Functions in PhysiCell

Functions in PhysiCell

- In PhysiCell, almost all cell functions have the following form:

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

- **`pCell`** : pointer to a cell. Can be NULL
- **`phenotype`**: a cell phenotype. Usually pCell->phenotype.
- **`dt`**: how far the function / model should be advanced in time.

- These functions can access:
  - Cell **state** : `pCell->state`
  - Cell **custom data** :
    ```
    get_single_behavior( pCell, "custom:data_name" );
    set_single_behavior( pCell, "custom:data_name" , new_value );
    ```
  - Cell **functions** : `pCell->functions`
  - Cell **phenotype** :
    ```
    get_single_behavior( pCell, "behavior_name" );
    set_single_behavior( pCell, "behavior_name" , new_value );
    ```
  - Reference **phenotype:** `get_single_base_behavior( pCell, "behavior_name" );`
  - nearby **microenvironment:**
    - `get_single_signal( pCell, "substrate_name");` extracellular value at cell location
    - `get_single_signal( pCell, "intracellular substrate_name");` intracellular value at cell location
    - `get_single_signal( pCell, "substrate_name gradient");` slope of substrate at cell location

Functions in PhysiCell

- Almost all functions in PhysiCell have this form:

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

All cells have the following key functions (`in pCell->functions`):

- `volume_update_function` (defaults to a built-in model)
- `update_migration_bias` (default NULL unless you enabled chemotaxis)
- `custom_cell_rule` (default NULL, evaluated at each mechanics time step)
- `update_phenotype` (default NULL, evaluated at each phenotype time step)
- `update_velocity` (defaults to a built-in model with potentials)
- `set_orientation` (automatically set as needed)
- `contact_function` (default NULL, evaluated at each mechanics time step)
  - We'll spend more time on this in Sessions 7 and 15

Purpose of the Functions

- **volume_update_function**
  - Dynamically grow / shrink cells towards "target" values
- **update_migration_bias**
  - Used whenever a cell chooses a new migration bias direction
- **custom_cell_rule**
  - A catch-all customization that's evaluated at each mechanics time step. (0.1 min)
  - Use this for rules that need frequent evaluation.
- **update_phenotype**
  - The general purpose rule to set phenotype parameters at each cell temp step. (6 min)
  - Generally where you spend the majority of your (implementation) time in a modeling project.
- **update_velocity**
  - Sets the cell velocity based on interaction potentials.
  - The custom rule and motility functions are automatically evaluated as well.
- **set_orientation**
  - Used during cell division to choose the division plane (a random plane through this vector).
  - We set this to (0,0,1) for 2-D simulations to ensure division in the xy-plane
- **contact_function**
  - A newer addition for cell-cell contact interactions such as adding/removing spring links. Evaluated at each mechanics step. More in Session 7.

A short example

- In custom.h, declare your new function;

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

- In custom.cpp, write the code:

```
void my_phenotype_function( Cell* pCell, Phenotype& phenotype, double dt )
{
	// get a rate from cell's custom data
	double rate = get_single_behavior( pCell, "custom:rate" );
	// change a cell's apoptosis rate
	set_single_behavior( pCell, "apoptosis", rate);
	return;
}
```

- Use the function:

```
cell_defaults.functions.update_phenotype = my_phenotype_function;
```

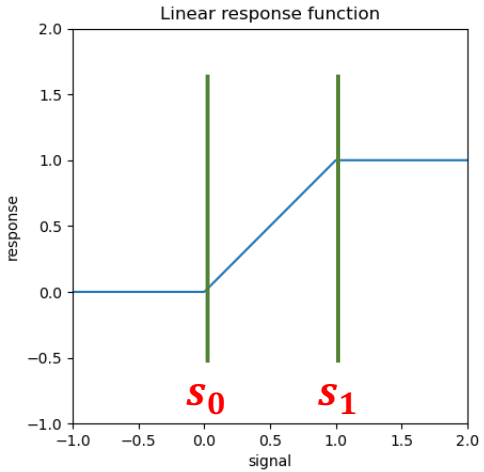
  - The best place to do this is in **`create_cell_types()`** in custom.cpp

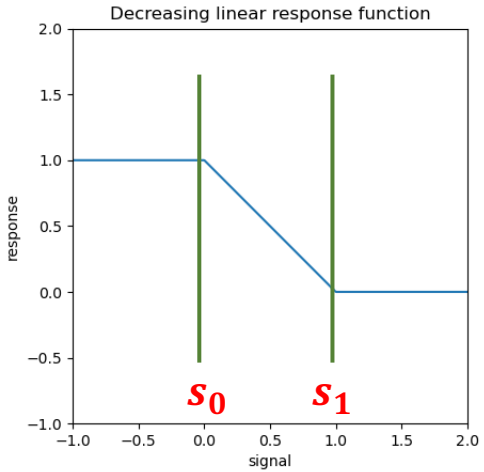
Handy C++ Functions

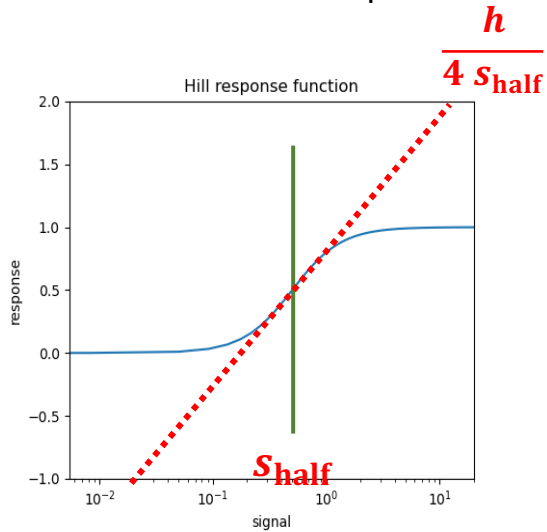
Built-in response functions

- **linear_response_function**(s, s0,s1)
  - Ramps from 0 to 1 as input increases from s0 to s1.
  - Outputs capped to [0,1]
- **decreasing_linear_response_function**(s,s0,s1)
  - Ramps from 1 to 0 as input increases from s0 to s1.
  - Outputs capped to [0,1]

- **Hill_response_function**(s, s_half, h);
  - Classical Hill function
  - s_half: half-max
  - h: Hill power
  - Tip: use integer powers for MUCH faster performance







Finding cell definitions

- `Cell_Definition* find_cell_definition( std::string )`
  - Get a pointer to a cell definition by searching for its name.

- `Cell_Definition* find_cell_definition( int )`
  - Get a pointer to a cell definition by searching for its integer type.
  - Since cells keep their `type_ID`, this can be quite handy for phenotype functions.

Assigning a function to a cell def

- In the create_cell_types function:
  - Find the cell definition for "my cell type"

```
Cell_Definiton* pCD = find_cell_definition( "my cell type" );
```

  - Assign your function(s) "my_phenotype"

```
pCD->functions.update_phenotype = my_phenotype;
```

Full modeling workflow

Suitable for creating a new PhysiCell model with custom C++ to drive dynamical phenotype changes

- Plan the model
- Populate a project
- Edit configuration Model Builder GUI
  - Edit domain
  - Edit microenvironment
  - Edit cell definitions
  - **Add custom variables**
  - **Add custom parameters**
- **Edit custom modules:**
  - **Declare functions in custom.h**
  - **Implement functions in custom.cpp**
  - **Assign functions to cell definitions**
- **Edit initial cell placement**
- **Edit cell coloring function**
- Build
- Run
- View results

Plan the model

A (simple) fibrosis model

- **Simple homeostatic model**
  - Epithelial cells proliferate
  - Mechanical pressure feedbacks reduce cycling
- **Necrotic damage**
  - Apoptotic cells release apoptotic debris
  - Necrotic cells release necrotic debris that increases epithelial death
  - We'll add an initial region of necrotic damage (e.g., a wound)
- **Macrophages and inflammation**
  - Chemotactically drawn to debris
  - Phagocytose dead cells (but with a volume feedback to prevent overgorging)
  - Necrotic debris triggers release of inflammatory factors
- **Fibrotic scarring**
  - Fibroblasts drawn to inflammation
  - Fibroblasts release collagen (we'll call it fibrosis)
  - Epithelial cells strongly adhere to fibrosis

Identifying the players

- Diffusing substrates / fields
  - apoptotic debris
  - necrotic debris
  - pro-inflammatory factor
  - fibrosis (non-diffusing)

- Cell types
  - epithelial
  - macrophage
  - fibroblast

Math: epithelial cells

- Proliferation with mechano feedback

$$b = b_0 = \left(1 - \frac{p}{p_m}\right)$$

- Apoptotic death driven by necrotic debris

$$d_A = d_{A0} + (d_{AM} - d_{A0}) \cdot Hill(n)$$

- Spring-like adhesion to fibrosis
  - If attached to $\mathbf{x}_A$, spring-like addition to cell velocity:

$$\boldsymbol{v} = \cdots + k\,(\boldsymbol{x}_A - \boldsymbol{x})$$

  - If not attached, attachment rate driven by presence of fibrosis

$$\text{Prob}(\text{attach to ECM in } [t, t + \Delta t]) = r_A \cdot Hill(f) \cdot \Delta t$$

  - Detachment: none for now

Math: macrophages

- Slow down phagocytosis if too big (need to digest materials)

$$r_{\text{phag}} = r_{P0} \cdot \left(1 - Hill(V)\right)$$

- Increase inflammatory secretion with necrotic debris

$$s = s_0 + (s_M - s_0) \cdot Hill(n)$$

Math: fibroblasts

- Increase fibrosis secretion with inflammatory signal

$$s = s_0 + (s_M - s_0) \cdot Hill(\alpha)$$

Start with template project

Prepare a clean slate

- In terminal window:
  - make data-cleanup // cleanup date a
  - make reset // return to blank slate
  - make template // populate the template project
  - make // compile
    - note: our executable name is project

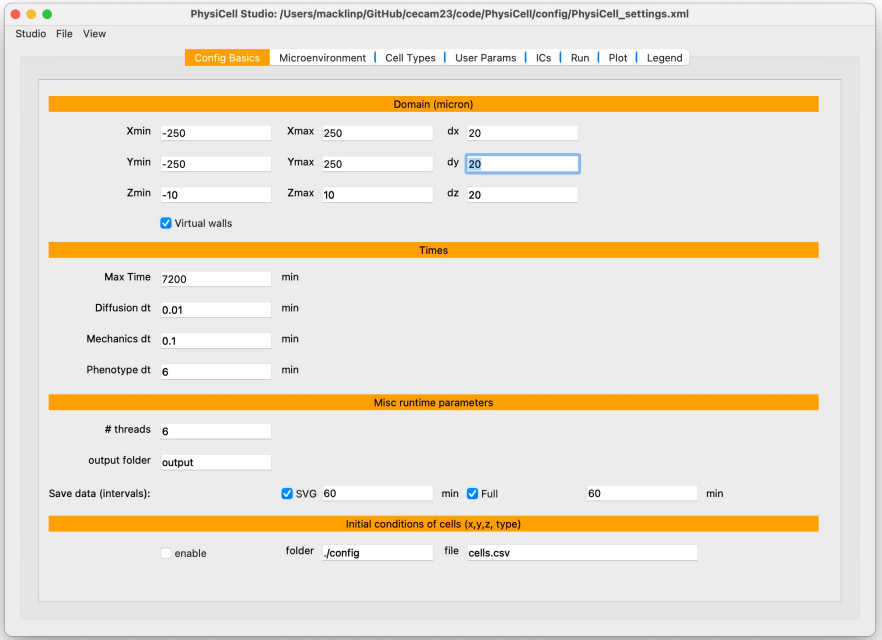
- In GUI:
  - Load the PhysiCell_settings.xml as before

Set up time and other parameters

- Reduce the domain to [-250,250] x [-250,250] x [-½ dz, ½ dz]

- Set the max simulation time to 5 days (7200 minutes)
  - Click the “config basics” tab
  - Set the “max time” to 7200

- Set the simulation outputs to every 60 minutes
  - Set SVG output interval to 60 min
    - This is how often the cell positions save
  - Set Full output interval to 60 min
    - This is how often the substrate data is saved



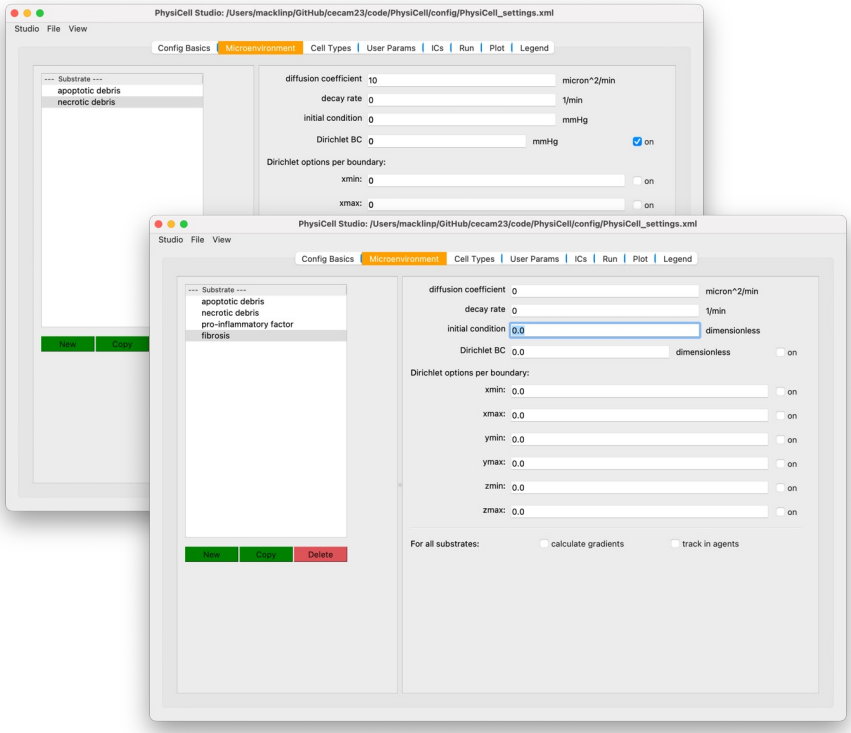
Prepare diffusing substrates

- Rename the first substrate to apoptotic debris
  - Diffusion coefficient small : 10
  - No decay
  - No initial or boundary conditions

- Copy the "apoptotic debris" to make necrotic debris, same properties

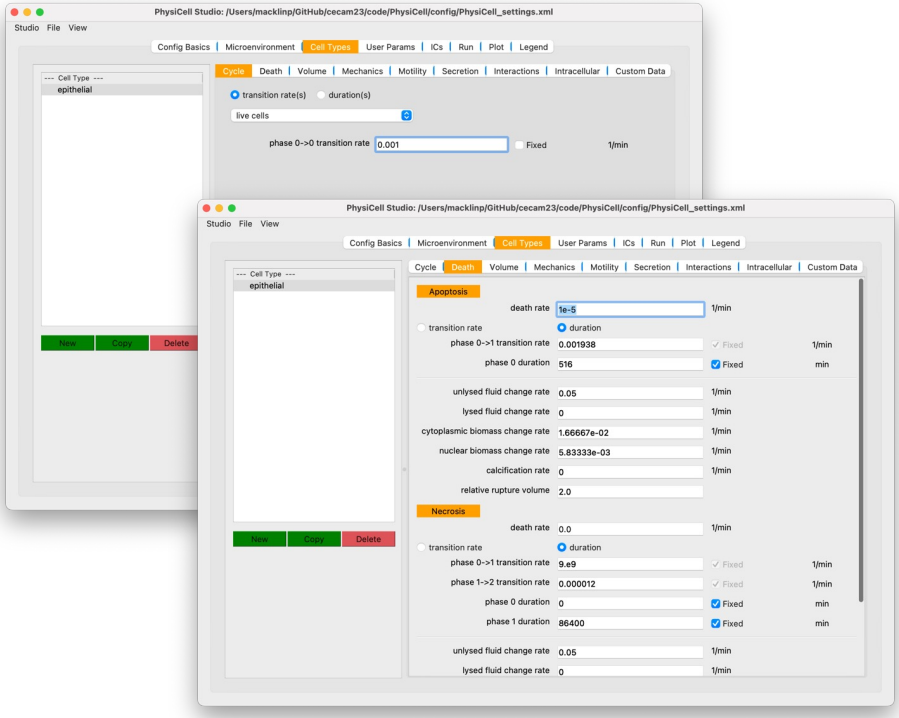
- Make a diffusing pro-inflammatory factor
  - Diffusion = 10000
  - Decay = 1

- Make one more called fibrosis
  - Diffusion = 0
  - Decay = 0



Create an epithelial type: birth/death

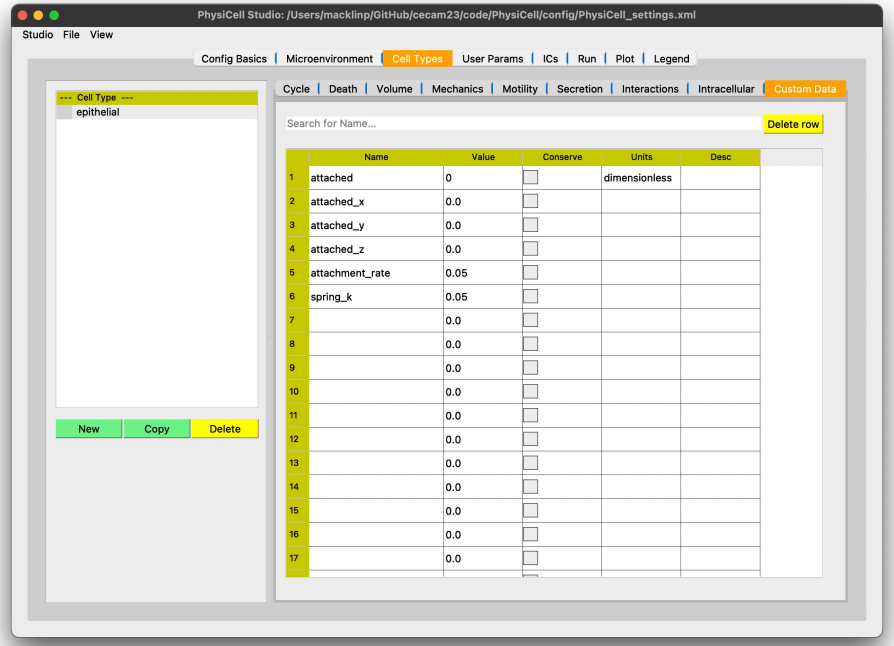
- Go to the “cell types” tab
- Rename to "epithelial"
- Set its cycling rate
  - Click on the “cycle” sub-tab
  - On the drop-down, select the “live cells” cycle model
  - For rate, chose 0.001 min^{-1}
- Set apoptotic death to 10^{-5} min^{-1}
  - Choose the “death” sub-tab
  - Under “Apoptosis”, set the death rate to 1e-5



Epithelial cells: custom

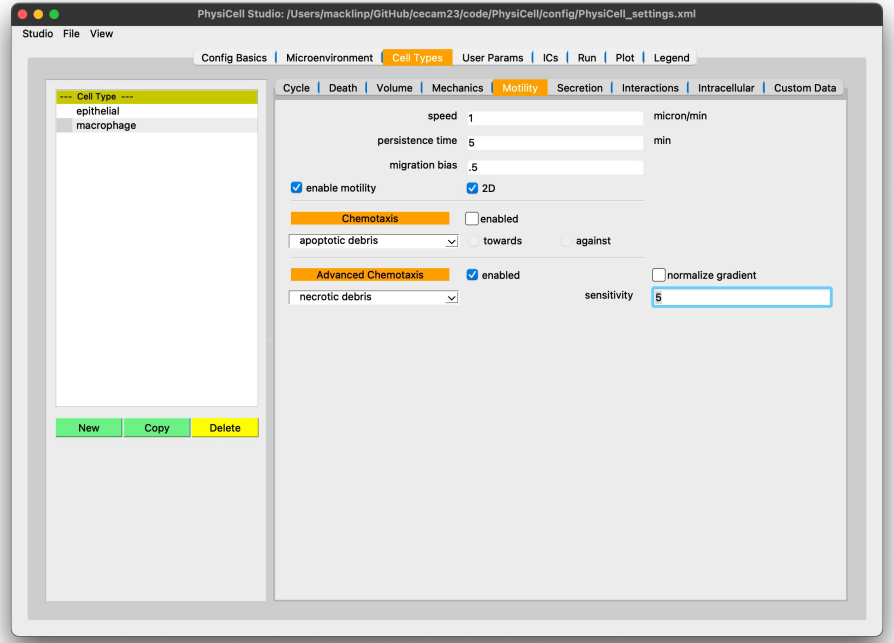
- Continue working on the “epithelial” type

- Go to the "custom data" sub-tab
  - Add "attached", initial value 0
  - Add "attached_x"
  - Add "attached_y"
  - Add "attached_z"
  - Add "attachment_rate", value 0.05
  - Add "spring_k", value 0.05



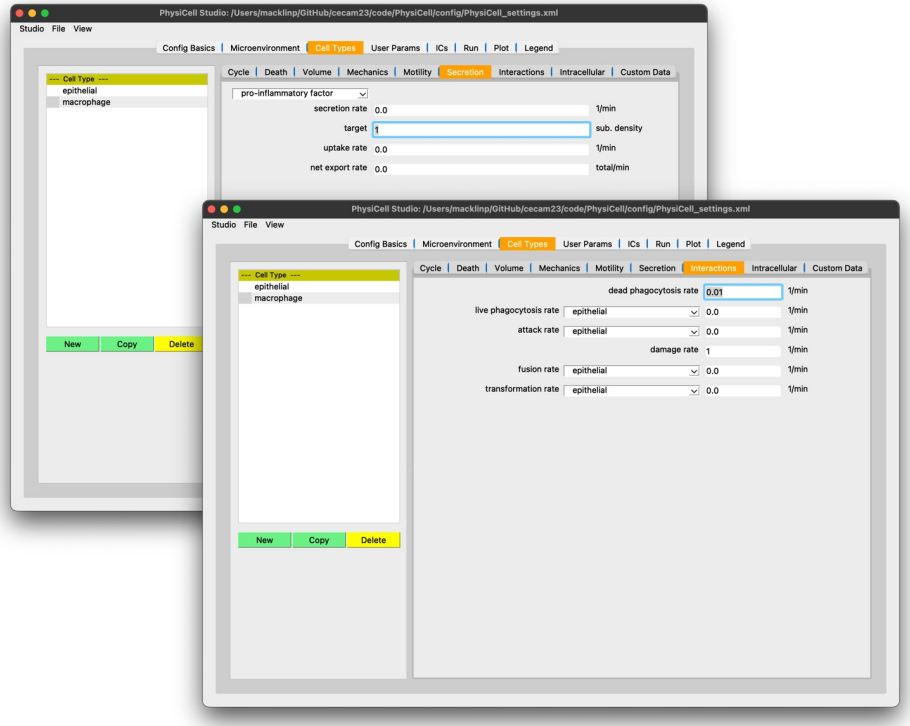
Create macrophages (1)

- Copy epithelial
- Cycle
  - rate = 0
- Apoptosis
  - rate = 0
- Mechanics
  - Cell-cell adhesion = 0
- Motility
  - enabled, 1 micron/min, 5 min persistence time, bias 0.5
  - Advanced chemotaxis
    - sensitivity to apoptotic debris = 1
    - sensitivity to necrotic debris = 10



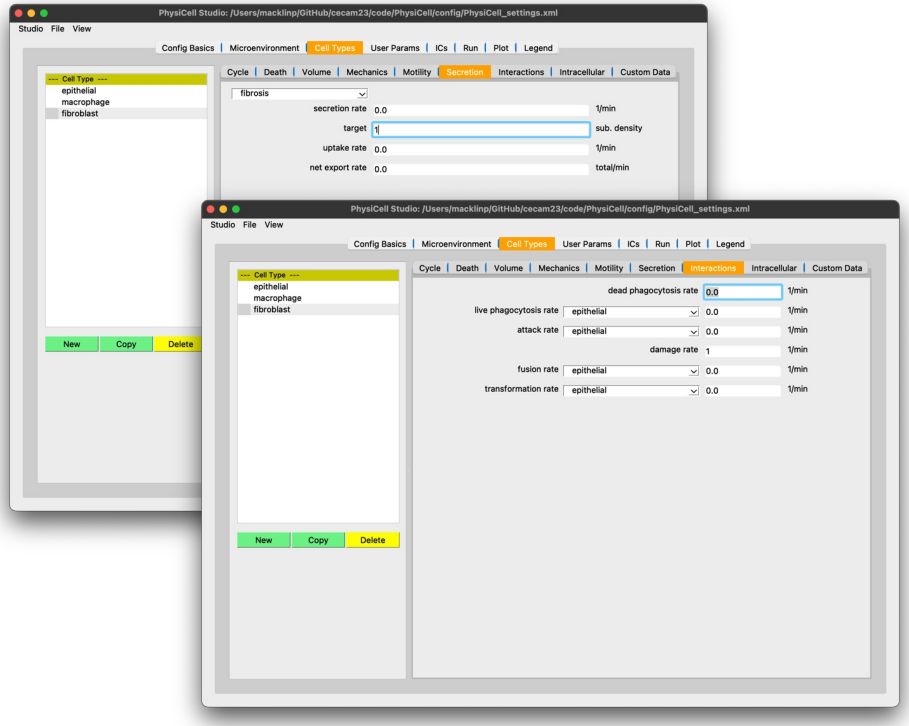
Create macrophages (2)

- Secretion
  - apoptotic debris
    - ♦ uptake rate = 1
  - necrotic debris
    - ♦ uptake rate = 1
  - pro-inflammatory factor
    - ♦ secretion rate = 0, target = 1
- interactions
  - dead cell phagocytosis = 0.01



Create fibroblasts

- Copy macrophage
  - Cell-cell adhesion = 0
- Motility
  - enabled, 1 micron/min, 5 min persistence time, bias 0.5
  - Regular chemotaxis
    - Follow pro-inflammatory factor
- Secretion
  - 0 uptake of debris
  - 1 uptake of pro-inflammatory factor
  - secretion target =1 for fibrosis
- Interactions
  - Disable dead cell phagocytosis



Place cells in the environment

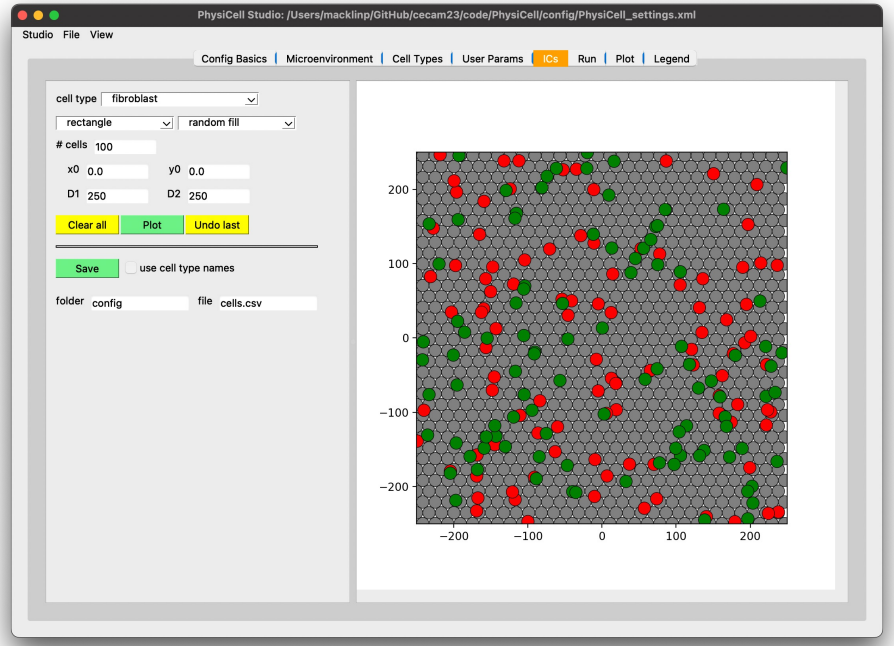
- Go to the ”ICs” (initial conditions) tab

- Epithelial cells
  - Choose “rectangle" , hex fill
  - Set (x0,y0) (center) to (0,0)
  - Set L1 , L2 ("radius" in x and y directions) to 250 , 250
  - Click “plot”

- Macrophage
  - Choose “predator” in the drop-down
  - Choose “rectangle" , random fill
  - Select 100 cells
  - Set (x0,y0) (center) to (0,0)
  - Set L1 , L2 ("radius" in x and y directions) to 250 , 250

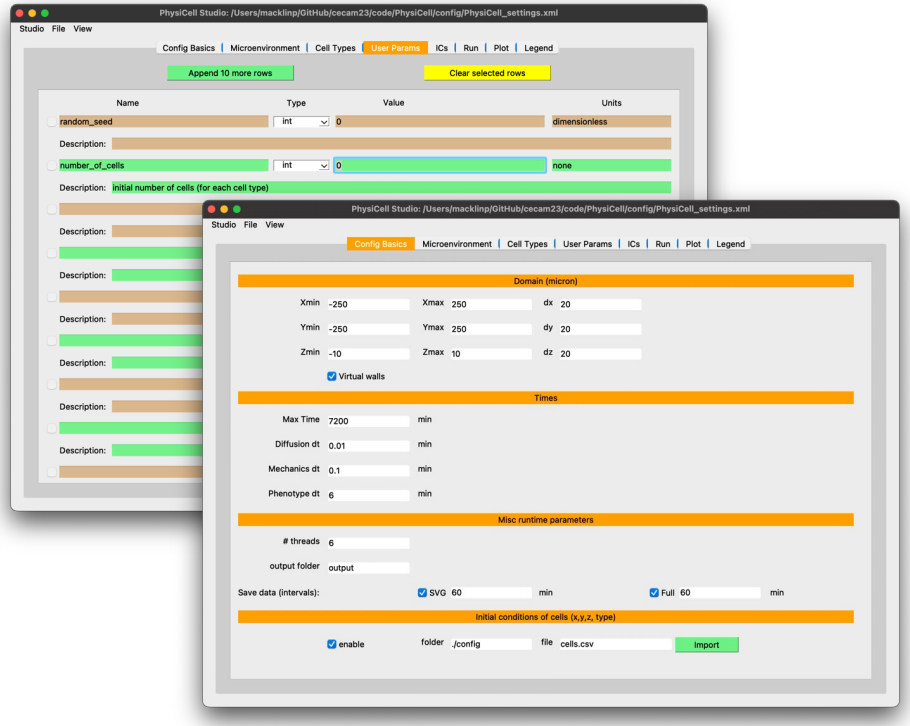
- Repeat for fibroblasts

- Export the cells
  - Click the "save" button to write this plot to a CSV file
    - Make sure it saves to "config" as "cells.csv"



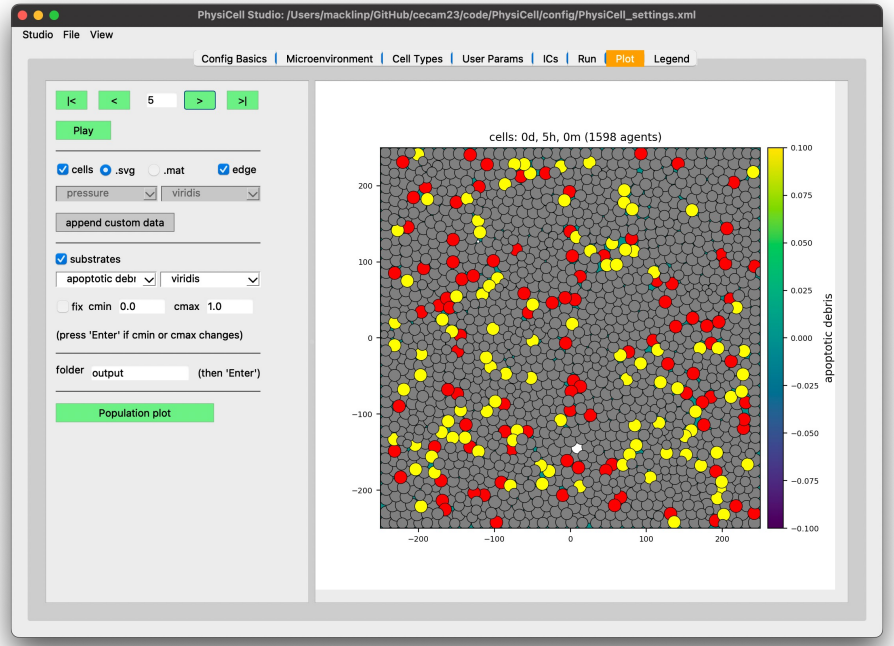
Use these cell positions

- Disable random placement of cells
  - Go to the ”User params” tab
  - Click on the “number_of_cells” variable
  - Set the value to 0
- Enable the code to read the initial cell positions in CSV
  - Go to the “Config basics” tab
  - In the bottom “initial conditions of cells” section, click “enable”



Visualize results

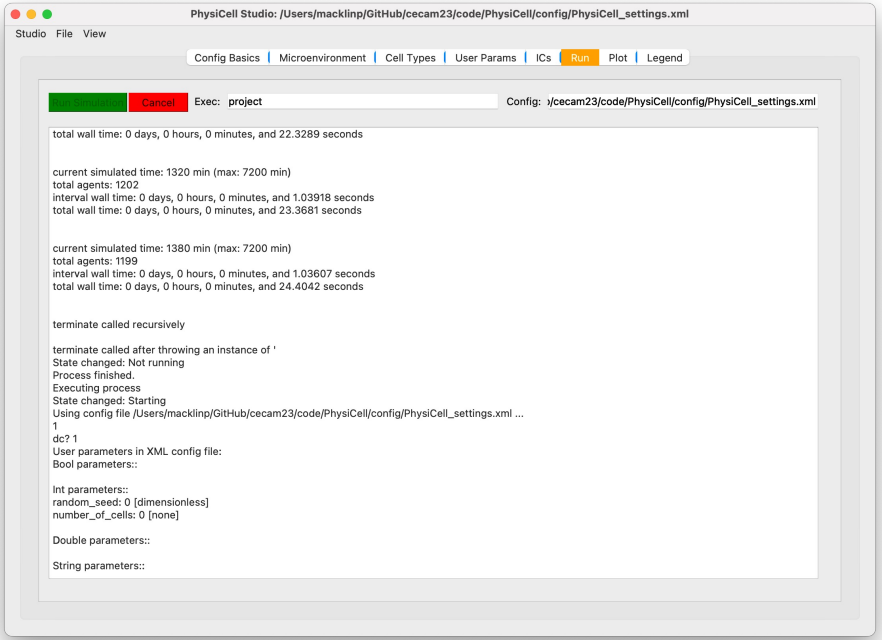
- Go to the “Plot” tab
- Click the Play and similar buttons to animate the plots
- Use the “substrates” to choose different substrates to plot



Create functions

Run the simulation

- Go to the “Run” tab
- Make sure the executable name is correct (in this case, “project”)
- Click “run simulation”



Overall custom function outline

- Check for dead cells
  - **idea:** make sure you don't process dead cells. Don’t' want non-zero birth rates for dead cells!
- Read necessary signals
- Get base / reference behavior values
- Use math to calculate new parameters
- Assign parameters

epithelial cells: starting

```
// in custom.h, declar the function

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


// in custom.cpp implemented it!

   // work on dead cells

void epithelial_phenotype( Cell* pCell , Phenotype& phenotype , double dt )
{
   // if apoptotic, release apoptotic debris and exit
   if( get_single_signal( pCell, "apoptotic") > 0.5 )
   {
      set_single_behavior( pCell, "apoptotic debris secretion" , 1);
      pCell->functions.update_phenotype = NULL;
      return;
   }

   // if necrotic, release necrotic debris and exit
   if( get_single_signal( pCell, "necrotic") > 0.5 )
   {
      set_single_behavior( pCell, "necrotic debris secretion" , 1);
      pCell->functions.update_phenotype = NULL;
      return;
   }
```

epithelial cells: signals, reference values

```
// get signals
double p = get_single_signal( pCell , "pressure");
double p_max = 1.0;
double nd = get_single_signal( pCell , "necrotic debris");
double f = get_single_signal( pCell , "fibrosis");

// get (nonzero) base values, and needed constants
double b0 = get_single_base_behavior( pCell , "cycle entry");

double dA0 = get_single_base_behavior( pCell , "apoptosis");
double dAM = 100 * dA0;

bool attached = (bool) get_single_signal( pCell , "custom:attached");
double rA = get_single_behavior( pCell ,"custom:attachment_rate");
```

epithelial cells: setting birth and death

```
// calculate and set values
   // birth
p /= p_max;
if( p > 1 )
{ p = 1; }
double b = b0 * (1-p);
set_single_behavior( pCell , "cycle entry" , b );

   // apoptosis
double dA = dA0 + (dAM-dA0) * Hill_response_function( nd , 0.1 , 4 );
set_single_behavior( pCell , "apoptosis" , dA );
```

epithelial cells: dynamic ECM attachment

```
  // dynamic attachment to ECM
if( attached == false )
{
  double prob_attach = rA * Hill_response_function( f , 0.5 , 4 ) * dt ;
  if( UniformRandom() <= prob_attach )
  {
    set_single_behavior( pCell , "custom:attached_x" , pCell->position[0] );
    set_single_behavior( pCell , "custom:attached_y" , pCell->position[1] );
    set_single_behavior( pCell , "custom:attached_z" , pCell->position[2] );

    set_single_behavior( pCell , "custom:attached" , 1.0 );
  }
}
```

custom function for the spring

```
// in custom.h
void epithelial_custom( Cell* pCell, Phenotype& phenotype, double dt );

// in custom.cpp

void epithelial_custom( Cell* pCell, Phenotype& phenotype, double dt )
{
   // exit if not attached
   if( get_single_signal( pCell , "custom:attached") < 0.5 )
   { return; }

   // get constants
   double k = get_single_behavior( pCell ,"custom:spring_k");
   double attached_x = get_single_signal( pCell , "custom:attached_x");
   double attached_y = get_single_signal( pCell , "custom:attached_y");
   double attached_z = get_single_signal( pCell , "custom:attached_z");

   // calculate stretch and dv
   std::vector<double> dv = { attached_x , attached_y , attached_z }; // (xA)
   dv -= pCell->position; // (xA-x)
   dv *= k; // k*(xA-x)

   pCell->velocity += dv;

   return;
}
```

macrophages

```
// in custom.h

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

// in custom.cpp

void macrophage_phenotype( Cell* pCell , Phenotype& phenotype , double dt )
{
   // get signals
   double nd = get_single_signal( pCell , "necrotic debris");

   // get ref values
   double s0 = get_single_base_behavior( pCell , "pro-inflammatory factor secretion");
   double sM = 10; //

   double ph0 = get_single_base_behavior( pCell , "phagocytose dead cell");
   double ph = ph0;
   ph = ph0;
   if( vol > 2500 )
   { ph = 0; }
   set_single_behavior( pCell , "phagocytose dead cell" , ph );

   // calculate and set
   double s = s0 + (sM-s0)*Hill_response_function( nd , 0.5 , 2 );
   set_single_behavior( pCell , "pro-inflammatory factor secretion" , s );


   return;
}
```

fibroblasts

```
// in custom.h

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

// in custom.cpp

void fibroblast_phenotype( Cell* pCell , Phenotype& phenotype , double dt )
{
   // get signals
   double pif = get_single_signal( pCell , "pro-inflammatory factor");

   // get ref values
   double s0 = get_single_base_behavior( pCell , "fibrosis secretion");
   double sM = 1; //

   // calculate and set
   double s = s0 + (sM-s0)*Hill_response_function( pif , 0.5 , 3 ); // 4
   set_single_behavior( pCell , "fibrosis secretion" , s );


   return;
}
```

Assign functions

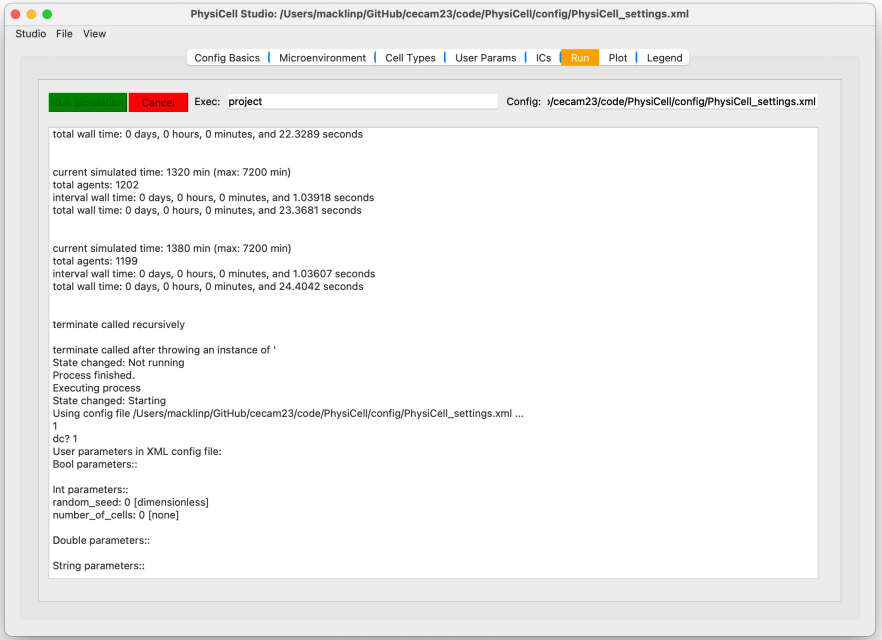
```
// in create_cell_types()

// ..
/*
   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* pCD = find_cell_definition( "epithelial");
pCD->functions.update_phenotype = epithelial_phenotype;
pCD->functions.custom_cell_rule = epithelial_custom;

pCD = find_cell_definition( "macrophage");
pCD->functions.update_phenotype = macrophage_phenotype;
pCD = find_cell_definition( "fibroblast");
pCD->functions.update_phenotype = fibroblast_phenotype;

/*
   This builds the map of cell definitions and summarizes the setup.
*/
display_cell_definitions( std::cout );
```

Let's nucleate a wound

```
// at endof setup_tissue()

// load cells from your CSV file (if enabled)
load_cells_from_pugixml();

for( int n =0; n < (*all_cells).size() ; n++ )
{
  Cell* pC = (*all_cells)[n];
  if( fabs( pC->position[0] ) < 50 && fabs( pC->position[1] ) < 50 )
  { set_single_behavior( pC , "necrosis" , 9e99 ); }
}

  return;
}
```

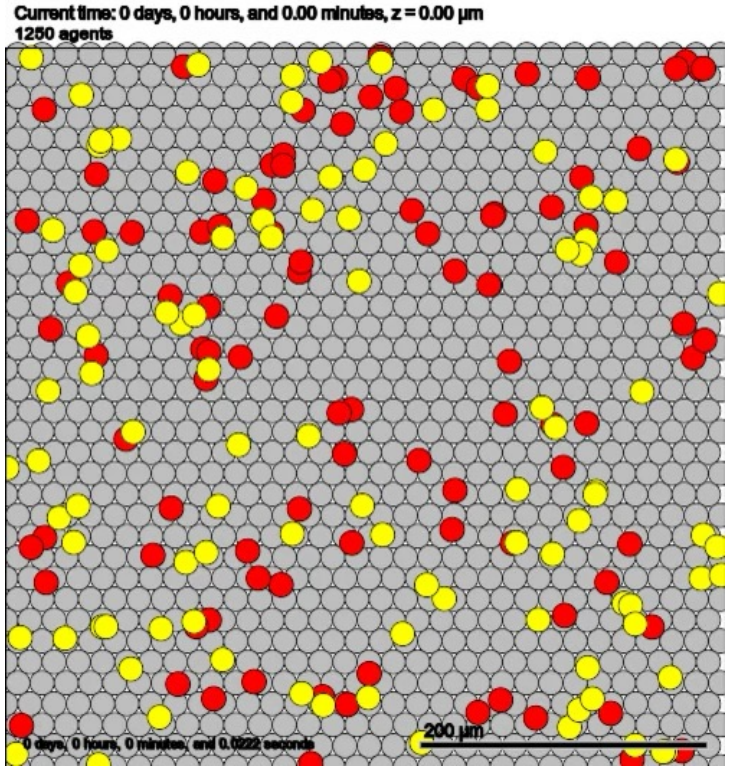
Run the simulation

- Go to the “Run” tab
- Make sure the executable name is correct (in this case, “project”)
- Click “run simulation”



Visualize results

- Go to the “Plot” tab
- Click the Play and similar buttons to animate the plots
- Use the “substrates” to choose different substrates to plot



Retrieving this project

- This project is saved as "demo2"
  - make load PROJ=demo2

Some other topics for further depth

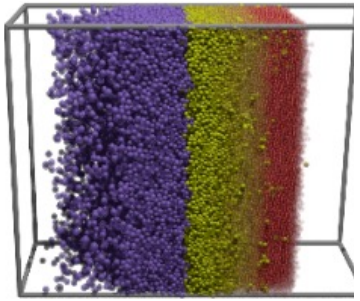
- **Initialization**
  - Detailed data structures for loading initial cell positions and states
  - Ability to read non-homogeneous initial substrate conditions
- **Mechanics**
  - Cell interactions with level surface functions
  - Strain
  - Plastic-elastic materials – dynamically form and break spring links
  - ECM microstructure
- Further depth (and active development) of **reference models**
  - Cell damage / integrity
  - Cell polarization
  - Immunogenicity
- **New (software / framework) architectures**
  - Subcellular element representations (for cell morphology)
  - Next-generation architectures (GPU + CPU + MPI)

Save the date!

2023 Virtual PhysiCell Workshop and Hackathon

July 23-29, 2023

- **Build and explore multicellular agent-based simulations of cancer and other systems**
- **Learn to share your models online**
- **Meet other modelers in the PhysiCell community**
- **Compete in an exclusive mentored hackathon**
- **PhysiCell swag available for accepted participants**



- **Application and full agenda coming soon at:** https://github.com/PhysiCell-Training/ws2023