Slides, videos, links and more:
https://github.com/physicell-training/ws2022

Session 6: Functions in PhysiCell

Paul Macklin, Ph.D.
@MathCancer

PhysiCell Project

July 26, 2022

Goals

- Introduced the full modeling workflow
- Typical form / syntax / purpose of PhysiCell functions
- Learn about the available customizable functions in cell.functions
- Learn how to assign new functions to a cell definition

- **Example:**
  - oxygen-based tumor cell birth and death
  - dead cells release debris
  - macrophages consume and chemotax towards debris
  - macrophages phagocytose dead cells

- **Stretch goal:** Controlling initial cell placement
- **Stretch goal:** Custom coloring functions

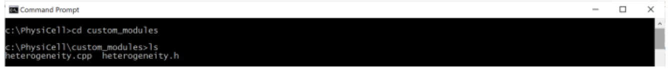
Full modeling workflow

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

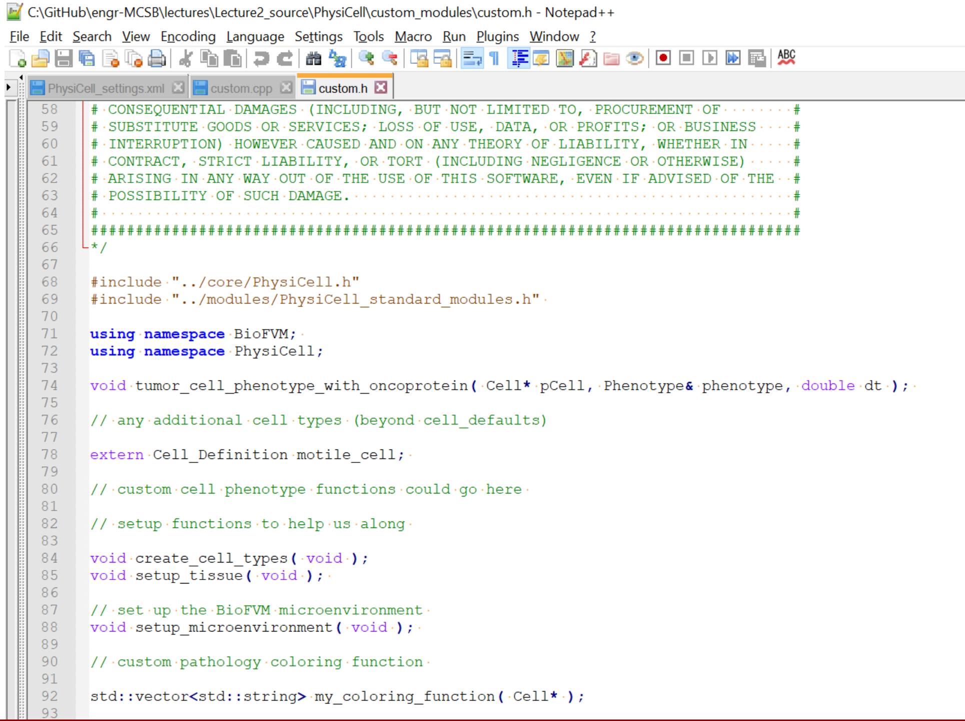
- Plan the model
- Populate a project
- Edit configuration Model Builder GUI
  - Edit domain
  - Edit microenvironment
  - Edit cell definitions
  - **Add custom variables**
  - **Add custom parameters**
- **Edit custom modules:**
  - **Declare functions in custom.h**
  - **Implement functions in custom.cpp**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

Project structure: custom modules

- Custom Modules
  - Setup functions
  - Cell definitions
  - Custom functions
  - any other modeling
  - Custom coloring functions



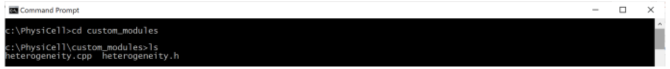
```
c:\PhysiCell>cd custom_modules

c:\PhysiCell\custom_modules>ls
heterogeneity.cpp  heterogeneity.h
```


C:\GitHub\engr-MCSB\lectures\Lecture2_source\PhysiCell\custom_modules\custom.h - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
PhysiCell_settings.xml
custom.cpp
custom.h

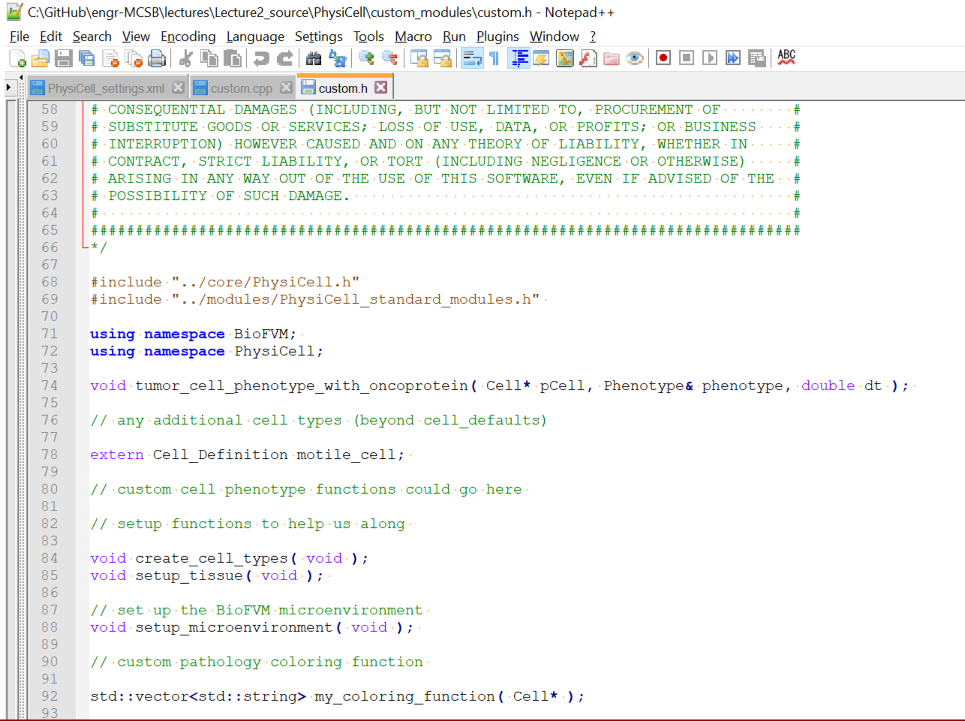
```
# CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF        #
# SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS    #
# INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN     #
# CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)     #
# ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE  #
# POSSIBILITY OF SUCH DAMAGE.                                                 #
#                                                                             #
###############################################################################
*/

#include "../core/PhysiCell.h"
#include "../modules/PhysiCell_standard_modules.h"

using namespace BioFVM;
using namespace PhysiCell;

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

// any additional cell types (beyond cell_defaults)

extern Cell_Definition motile_cell;

// custom cell phenotype functions could go here

// setup functions to help us along

void create_cell_types( void );
void setup_tissue( void );

// set up the BioFVM microenvironment
void setup_microenvironment( void );

// custom pathology coloring function

std::vector<std::string> my_coloring_function( Cell* );

```

Project structure: custom modules

- Custom Modules
  - Any user-defined globals (at top)
  - Setup functions
    - ♦ `create_cell_types()`
      - » Do all setup on all cell types
        - ○ Adjust phenotype
        - ○ Add / adjust custom data
        - ○ Assign functions
    - ♦ `setup_tissue()`
      - » Place initial cells in microenvironment
      - » Modify each cell as needed
  - Custom functions
  - any other modeling
  - Custom coloring functions



```
c:\PhysiCell>cd custom_modules

c:\PhysiCell\custom_modules>ls
heterogeneity.cpp  heterogeneity.h
```

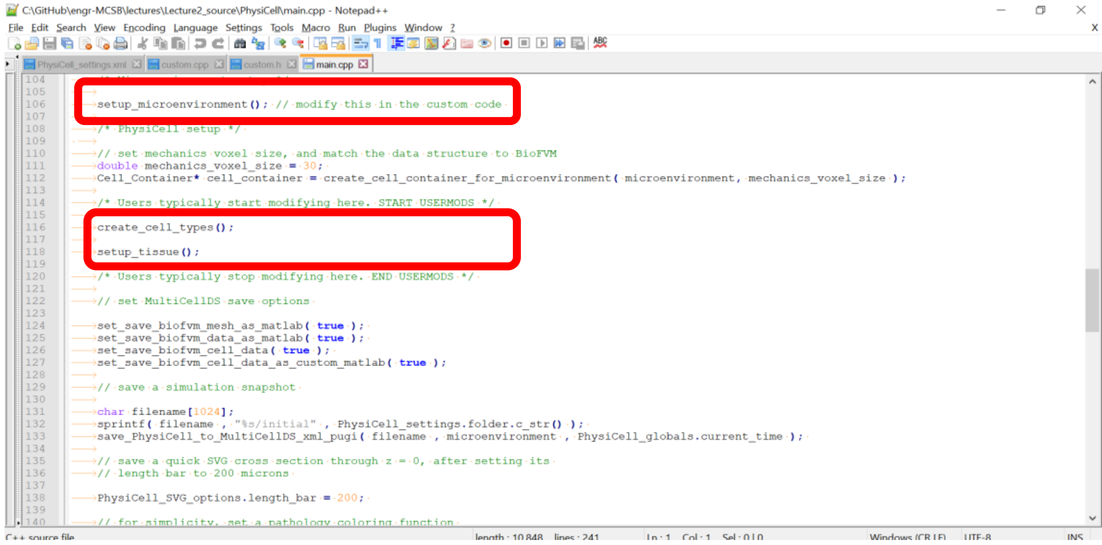


C:\GitHub\engr-MCSB\lectures\Lecture2_source\PhysiCell\custom_modules\custom.h - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
PhysiCell_settings.xml
custom.cpp
custom.h

```
# CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF       #
# SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS   #
# INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN    #
# CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)    #
# ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE #
# POSSIBILITY OF SUCH DAMAGE.                                                #
#                                                                            #
##############################################################################
*/

#include "../core/PhysiCell.h"
#include "../modules/PhysiCell_standard_modules.h"

using namespace BioFVM;
using namespace PhysiCell;

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

// any additional cell types (beyond cell_defaults)

extern Cell_Definition motile_cell;

// custom cell phenotype functions could go here

// setup functions to help us along

void create_cell_types( void );
void setup_tissue( void );

// set up the BioFVM microenvironment
void setup_microenvironment( void );

// custom pathology coloring function

std::vector<std::string> my_coloring_function( Cell* );

```

Project structure: main.cpp

- `main.cpp`
  - (in the root directory)
  - calls the setup functions

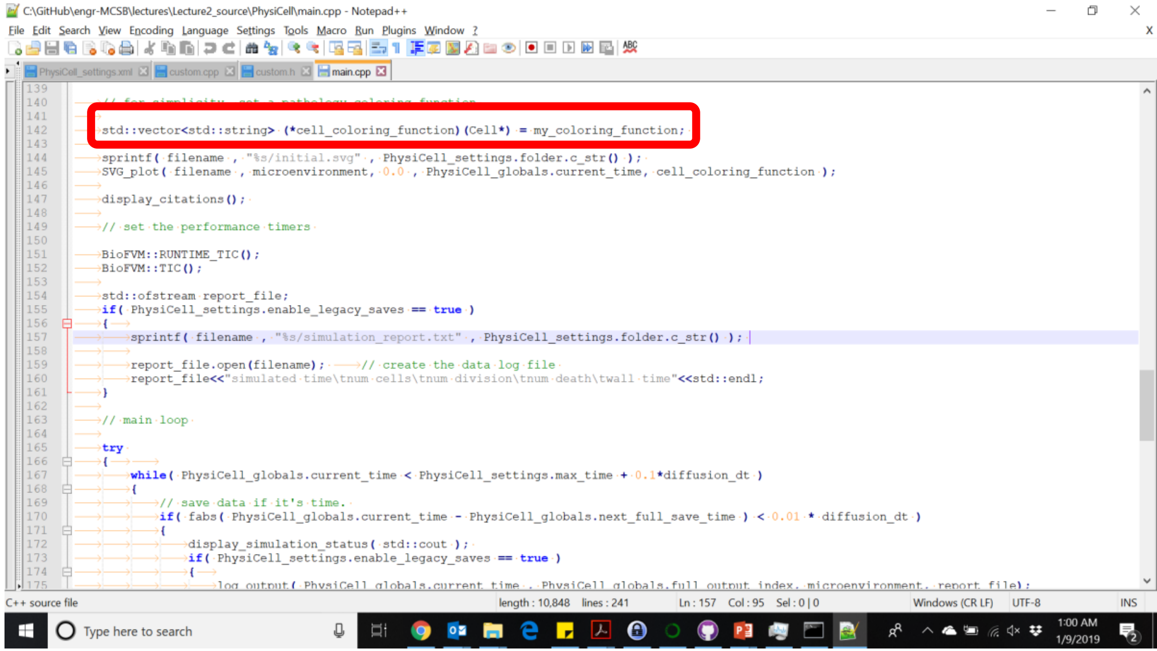

C:\GitHub\engr-MCSB\lectures\Lecture2_source\PhysiCell\main.cpp - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
PhysiCell_settings.xml
custom.cpp
custom.h
main.cpp

```
	setup_microenvironment(); // modify this in the custom code 
	/* PhysiCell setup */ 
	
	// set mechanics voxel size, and match the data structure to BioFVM
	double mechanics_voxel_size = 30; 
	Cell_Container* cell_container = create_cell_container_for_microenvironment( microenvironment, mechanics_voxel_size );
	
	/* Users typically start modifying here. START USERMODS */ 
	
	create_cell_types();
	
	setup_tissue();
	
	/* Users typically stop modifying here. END USERMODS */ 
	
	// set MultiCellDS save options 
	
	set_save_biofvm_mesh_as_matlab( true ); 
	set_save_biofvm_data_as_matlab( true ); 
	set_save_biofvm_cell_data( true ); 
	set_save_biofvm_cell_data_as_custom_matlab( true );
	
	// save a simulation snapshot 
	
	char filename[1024];
	sprintf( filename , "%s/initial" , PhysiCell_settings.folder.c_str() ); 
	save_PhysiCell_to_MultiCellDS_xml_pugi( filename , microenvironment , PhysiCell_globals.current_time ); 
	
	// save a quick SVG cross section through z = 0, after setting its 
	// length bar to 200 microns 
	
	PhysiCell_SVG_options.length_bar = 200; 
	
	// for simplicity, set a pathology coloring function 
```

C++ source file
length : 10,848 lines : 241
Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF) UTF-8 INS

Project structure: main.cpp (continued)

- **`main.cpp`**
    - set coloring function

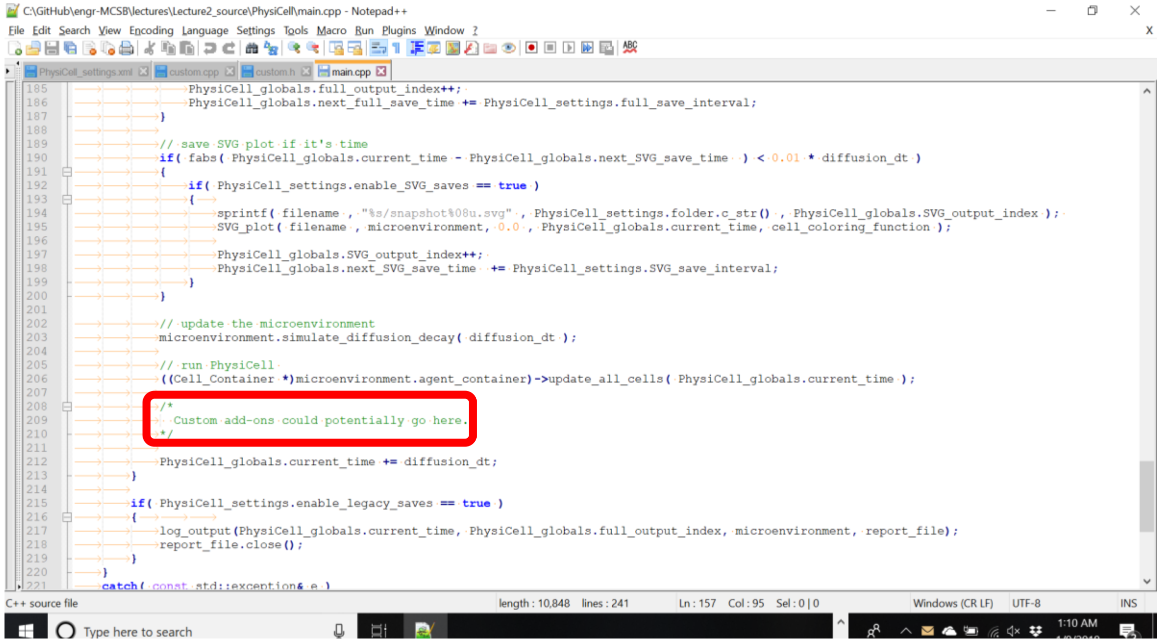

C:\GitHub\engr-MCSB\lectures\Lecture2_source\PhysiCell\main.cpp - Notepad++

```cpp
    // for simplicity, set a pathology coloring function

    std::vector<std::string> (*cell_coloring_function)(Cell*) = my_coloring_function;

    sprintf( filename , "%s/initial.svg" , PhysiCell_settings.folder.c_str() );
    SVG_plot( filename , microenvironment, 0.0 , PhysiCell_globals.current_time, cell_coloring_function );

    display_citations();

    // set the performance timers

    BioFVM::RUNTIME_TIC();
    BioFVM::TIC();

    std::ofstream report_file;
    if( PhysiCell_settings.enable_legacy_saves == true )
    {
        sprintf( filename , "%s/simulation_report.txt" , PhysiCell_settings.folder.c_str() );

        report_file.open(filename);     // create the data log file
        report_file<<"simulated time\tnum cells\tnum division\tnum death\twall time"<<std::endl;
    }

    // main loop

    try
    {
        while( PhysiCell_globals.current_time < PhysiCell_settings.max_time + 0.1*diffusion_dt )
        {
            // save data if it's time.
            if( fabs( PhysiCell_globals.current_time - PhysiCell_globals.next_full_save_time ) < 0.01 * diffusion_dt )
            {
                display_simulation_status( std::cout );
                if( PhysiCell_settings.enable_legacy_saves == true )
                {
                    log_output( PhysiCell_globals.current_time , PhysiCell_globals.full_output_index, microenvironment, report_file);
```

C++ source file   length : 10,848   lines : 241   Ln : 157   Col : 95   Sel : 0 | 0   Windows (CR LF)   UTF-8   INS

Project structure: main.cpp (continued)

- **`main.cpp`**
  - main loop:
    - ♦ **This would be a good place to put extensions.**


C:\GitHub\engr-MCSB\lectures\Lecture2_source\PhysiCell\main.cpp - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
PhysiCell_settings.xml
custom.cpp
custom.h
main.cpp

```cpp
                PhysiCell_globals.full_output_index++;
                PhysiCell_globals.next_full_save_time += PhysiCell_settings.full_save_interval;
            }

            // save SVG plot if it's time
            if( fabs( PhysiCell_globals.current_time - PhysiCell_globals.next_SVG_save_time  ) < 0.01 * diffusion_dt )
            {
                if( PhysiCell_settings.enable_SVG_saves == true )
                {
                    sprintf( filename , "%s/snapshot%08u.svg" , PhysiCell_settings.folder.c_str() , PhysiCell_globals.SVG_output_index );
                    SVG_plot( filename , microenvironment, 0.0 , PhysiCell_globals.current_time, cell_coloring_function );

                    PhysiCell_globals.SVG_output_index++;
                    PhysiCell_globals.next_SVG_save_time  += PhysiCell_settings.SVG_save_interval;
                }
            }

            // update the microenvironment
            microenvironment.simulate_diffusion_decay( diffusion_dt );

            // run PhysiCell
            ((Cell_Container *)microenvironment.agent_container)->update_all_cells( PhysiCell_globals.current_time );

            /*
              Custom add-ons could potentially go here.
            */

            PhysiCell_globals.current_time += diffusion_dt;
        }

        if( PhysiCell_settings.enable_legacy_saves == true )
        {
            log_output(PhysiCell_globals.current_time, PhysiCell_globals.full_output_index, microenvironment, report_file);
            report_file.close();
        }
    }
    catch( const std::exception& e )
```

Summary: Where things will go

- Declare custom functions in **./custom_modules/custom.h**
- Implement these functions in **./custom_modules/custom.cpp**
- Assign custom functions to cell definitions in custom.cpp in **create_cell_types()**;
- Declare any cell parameters needed for custom functions in the **custom_data** part of a cell definition in the XML configuration file
- Declare any parameters need to set up the simulation in the **user_parameters** part of the XML config file

PhysiCell Cell Functions

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" );
    ```
  - 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 I

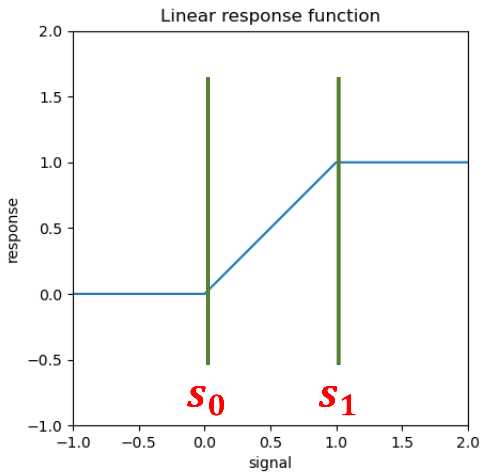
Reminder: finding cell definitions

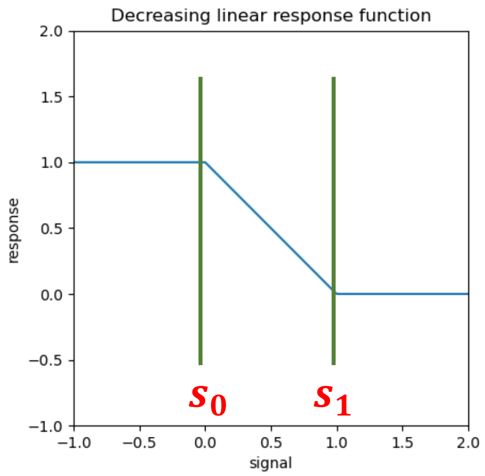
- `Cell_Definition*` **`find_cell_definition`**`( std::string )`
  - Get a pointer to a cell definition by searching for its name.

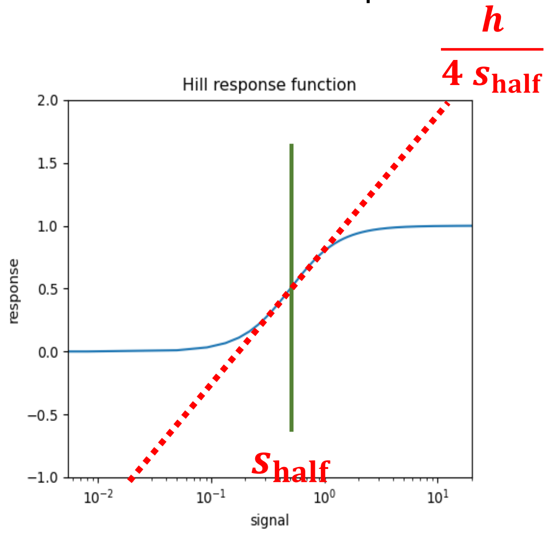
- `Cell_Definition*` **`find_cell_definition`**`( int )`
  - Get a pointer to a cell definition by searching for its integer type.
  - Since cells keep their `type_ID`, this can be quite handy for phenotype functions.

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







Full Model Workflow: Example

Scenario: simple tumor model

- Let's illustrate these with an example:
  - cancer cells:
    - Cycle entry proportional to local pO2
    - Necrosis probability increases below a pO2 threshold
    - Dead cells release debris (at rate proportional to cell volume)
  - macrophages:
    - chemotaxis towards debris
    - uptake debris
    - phagocytose dead cells

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

Checklist

- Plan ☐
- Build iteratively (in model builder):
  - Set the domain ☐
  - Add diffusing substrates ☐
  - Add cancer cells and test ☐
  - Add macrophages and test ☐
- Refinement (in C++):
  - Declare and write cancer cell phenotype ☐
  - Assign function ☐
  - Compile and test ☐

Planning (1)

- Microenvironment
  - [-400,400] x [-400,400], 2160 minutes max time.
  - Oxygen with default parameters, boundary and initial conditions to 38 mmHg
  - Debris with smaller diffusion coefficient, no decay, no-flux conditions
  - Enable virtual wall

- Custom cell data (known once you have planned your cell functions)

pO2_proliferation_saturation	(max proliferation rate above this value)
pO2_proliferation_threshold	(no proliferation below this value)
pO2_necrosis_threshold	(necrosis starts at this value)
pO2_necrosis_saturation	(necrosis at max value below this value)
max_necrosis_rate	(max necrotic death rate for very low pO2)

- Cell definitions
  - cancer
  - macrophage

Planning (2)

- cancer cell proliferation (σ = pO$_2$) with the simpler **live** cycle model.

$$r_{00} = \bar{r}_{00}\left(\frac{\sigma - \sigma_{\text{p\_threshold}}}{\sigma_{\text{p\_saturation}} - \sigma_{\text{p\_threshold}}}\right)$$

  - $\sigma_{\text{p\_saturation}} = 38\text{ mmHg }(5\%)$
  - $\sigma_{p\text{\_threshold}} = 5\text{ mmHg }(0.65\%)$
  - $\bar{r}_{00} = 7.2\text{e}{-4}\ \text{ min}^{-1}$

- cancer cell necrosis (σ = pO$_2$)

$$r_N = \bar{r}_N\left(\frac{\sigma_{\text{n\_threshold}} - \sigma}{\sigma_{\text{n\_threshold}} - \sigma_{\text{n\_saturation}}}\right)$$

  - $\sigma_{\text{n\_threshold}} = 5\text{ mmHg }(0.65\%)$
  - $\sigma_{\text{n\_saturation}} = 2.5\text{ mmHg }(0.32\%)$
  - $\bar{r}_N = 2.8\text{e}{-3}\ \text{ min}^{-1}$

Checklist

- Plan ☑
- Build iteratively (in model builder):
  - Set the domain ☐
  - Add diffusing substrates ☐
  - Add cancer cells and test ☐
  - Add macrophages and test ☐
- Refinement (in C++):
  - Declare and write cancer cell phenotype ☐
  - Assign function ☐
  - Compile and test ☐

Start modeling!

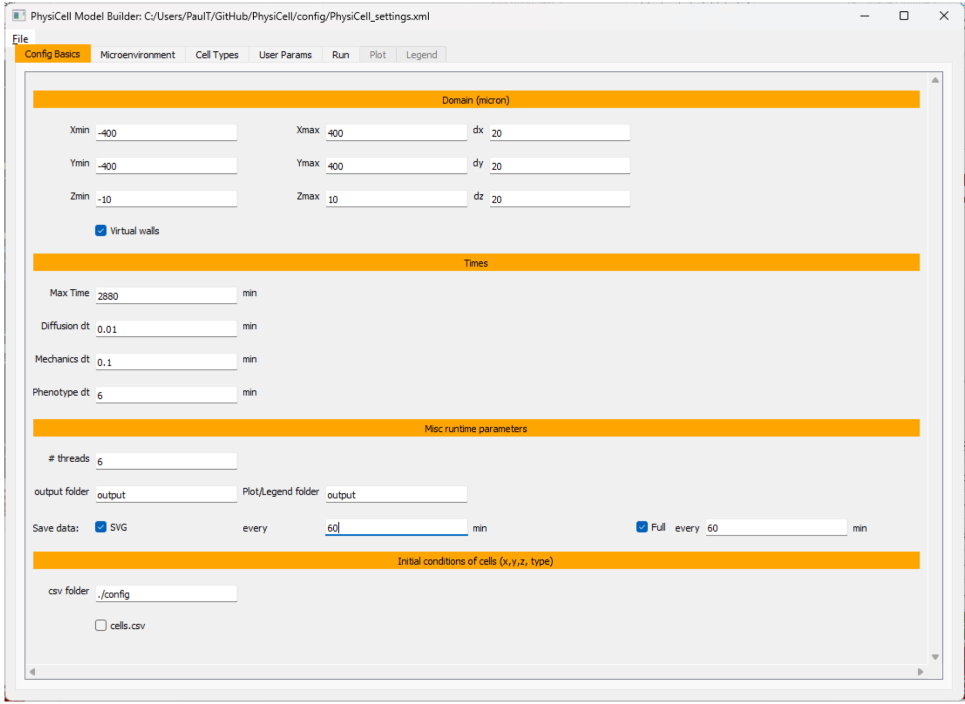
- populate and build the template project
  - `make template`
  - `make`

- Open Model Builder GUI
  - `python ../PhysiCell-model-builder/bin/pmb.py --studio`

- Open config/PhysiCell_settings.xml, and save.

Edit the model: domain

- Go to **config basics** tab
  - Xmin = -400, Xmax = 400
  - Ymin = -400, Ymax = 400
  - max time = 2880 (2 days)
  - full output every 30 min
  - SVG every 30 min
  - activate "virtual wall"
    - keep cells from leaving the domain


PhysiCell Model Builder: C:/Users/PaulT/GitHub/PhysiCell/config/PhysiCell_settings.xml
File
Config Basics
Microenvironment
Cell Types
User Params
Run
Plot
Legend

Domain (micron)

Xmin	-400	Xmax	400	dx	20
Ymin	-400	Ymax	400	dy	20
Zmin	-10	Zmax	10	dz	20

☑ Virtual walls

Times

Max Time	2880	min
Diffusion dt	0.01	min
Mechanics dt	0.1	min
Phenotype dt	6	min

Misc runtime parameters

threads 6

output folder output   Plot/Legend folder output

Save data: ☑ SVG every 60 min   ☑ Full every 60 min

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

csv folder ./config

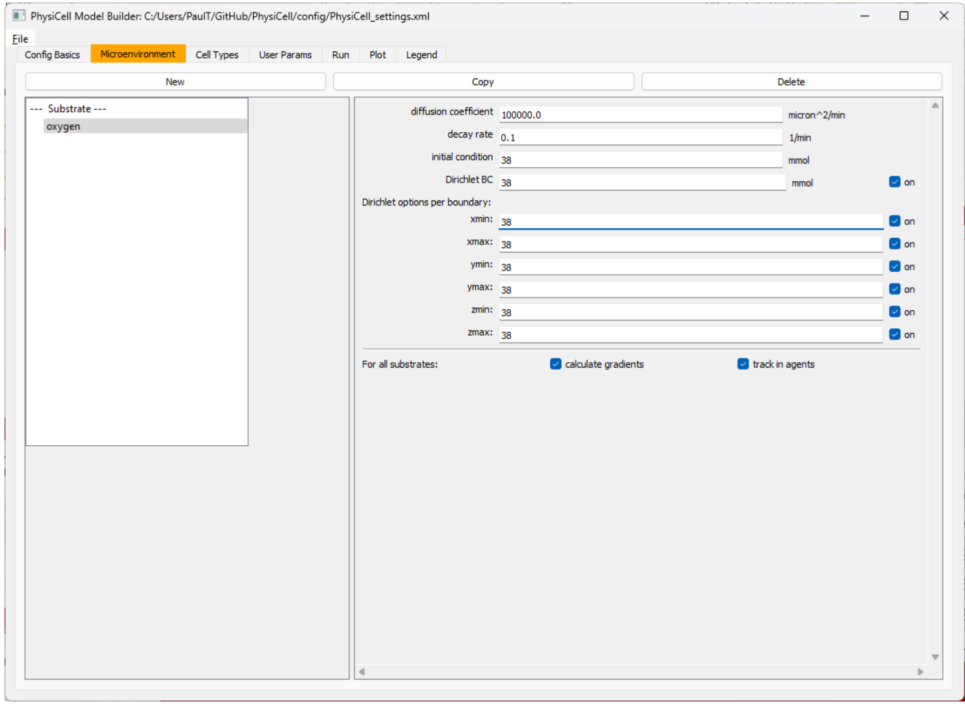
☐ cells.csv

Checklist

- Plan ☑
- Build iteratively (in model builder):
  - Set the domain ☑
  - Add diffusing substrates ☐
  - Add cancer cells and test ☐
  - Add macrophages and test ☐
- Refinement (in C++):
  - Declare and write cancer cell phenotype ☐
  - Assign function ☐
  - Compile and test ☐

Edit the model: microenvironment (1)

- Go to **microenvironment** tab
- double-click **substrate**
  - rename it **oxygen**, with units mmHg
  - reduce decay rate to 0.1
  - set Dirichlet BC to 38 (mmHg)
  - enable the Dirichlet BC
  - set initial value to 38 (mmHg)


PhysiCell Model Builder: C:/Users/PaulT/GitHub/PhysiCell/config/PhysiCell_settings.xml

File

Config Basics | Microenvironment | Cell Types | User Params | Run | Plot | Legend

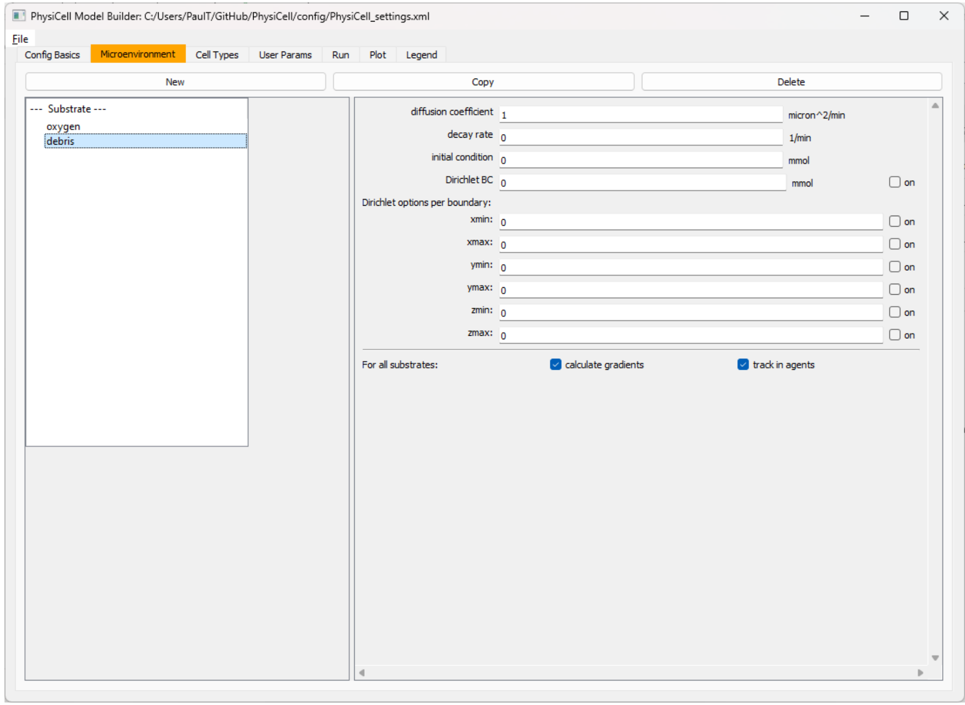
New | Copy | Delete

--- Substrate ---
oxygen

diffusion coefficient 100000.0 micron^2/min
decay rate 0.1 1/min
initial condition 38 mmol
Dirichlet BC 38 mmol ✓ on
Dirichlet options per boundary:
xmin: 38 ✓ on
xmax: 38 ✓ on
ymin: 38 ✓ on
ymax: 38 ✓ on
zmin: 38 ✓ on
zmax: 38 ✓ on
For all substrates: ✓ calculate gradients ✓ track in agents

Edit the model: microenvironment (2)

- select **oxygen** and copy
- double-click, rename to **debris**
  - diffusion 1
  - decay 0
  - initial condition: 0
  - No boundary condition



PhysiCell Model Builder: C:/Users/PaulT/GitHub/PhysiCell/config/PhysiCell_settings.xml

File

Config Basics | Microenvironment | Cell Types | User Params | Run | Plot | Legend

New | Copy | Delete

--- Substrate ---
oxygen
debris

diffusion coefficient	1	micron^2/min	
decay rate	0	1/min	
initial condition	0	mmol	
Dirichlet BC	0	mmol	☐ on

Dirichlet options per boundary:

xmin:	0	☐ on
xmax:	0	☐ on
ymin:	0	☐ on
ymax:	0	☐ on
zmin:	0	☐ on
zmax:	0	☐ on

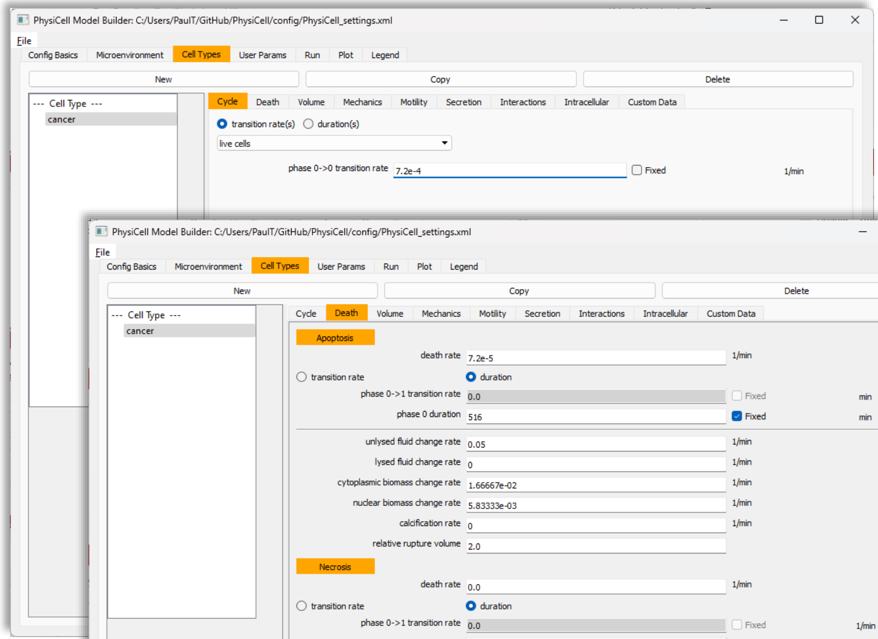
For all substrates: ☑ calculate gradients ☑ track in agents

Checklist

- Plan ✓
- Build iteratively (in model builder):
  - Set the domain ✓
  - Add diffusing substrates ✓
  - Add cancer cells and test ☐
  - Add macrophages and test ☐
- Refinement (in C++):
  - Declare and write cancer cell phenotype ☐
  - Assign function ☐
  - Compile and test ☐

Create cancer cells: birth and death

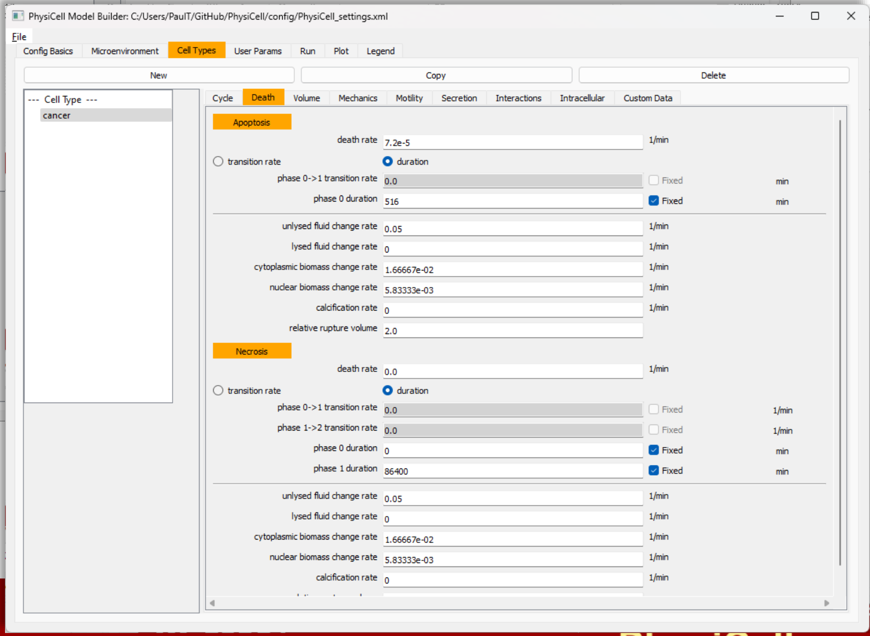
- Click the **cell types** tab
- double-click **default**
  - rename to **cancer**
  - go to **cycle**
    - ♦ Choose **live cells** from the drop-down menu of cycle models
    - ♦ use the **transition rates** form
    - ♦ Set the 0→0 transition rate to **7.2e-4**
  - Go to **death**
    - ♦ Set the **apoptosis** rate to **7.2e-5**



[✓ transition rate(s)] duration(s)

live cells

phase 0->0 transition rate: 7.2e-4 [ ] Fixed 1/min



Apoptosis

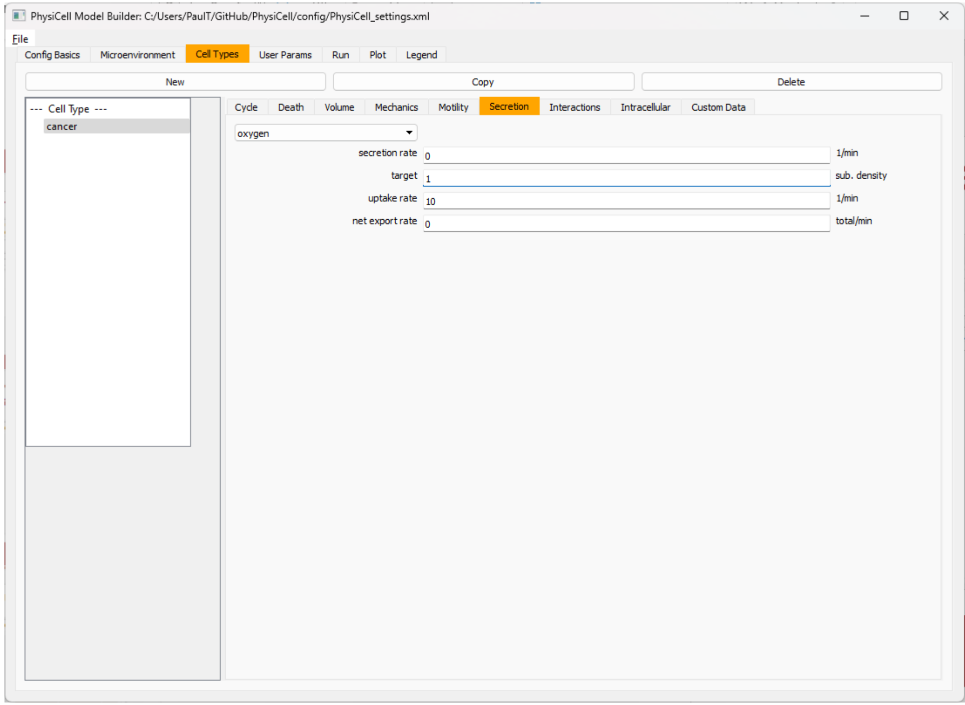
death rate	7.2e-5		1/min
transition rate [✓ duration]			
phase 0->1 transition rate	0.0	[ ] Fixed	min
phase 0 duration	516	[✓] Fixed	min
unlysed fluid change rate	0.05		1/min
lysed fluid change rate	0		1/min
cytoplasmic biomass change rate	1.66667e-02		1/min
nuclear biomass change rate	5.83333e-03		1/min
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	1/min
phase 1->2 transition rate	0.0	[ ] Fixed	1/min
phase 0 duration	0	[✓] Fixed	min
phase 1 duration	86400	[✓] Fixed	min
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

Create cancer cells: uptake

- Let's enable O2 uptake
  - Click **secretion**
  - Choose **oxygen** in the drop-down menu
  - Set the **uptake rate** to **10**

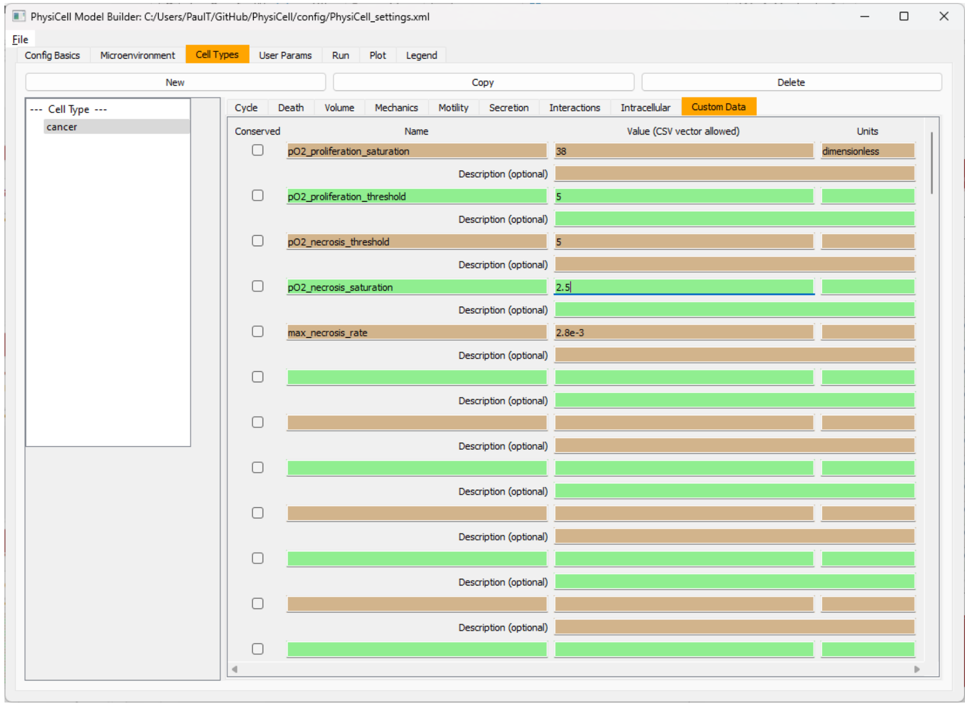


oxygen

secretion rate	0	1/min
target	1	sub. density
uptake rate	10	1/min
net export rate	0	total/min

Create cancer cells: custom data

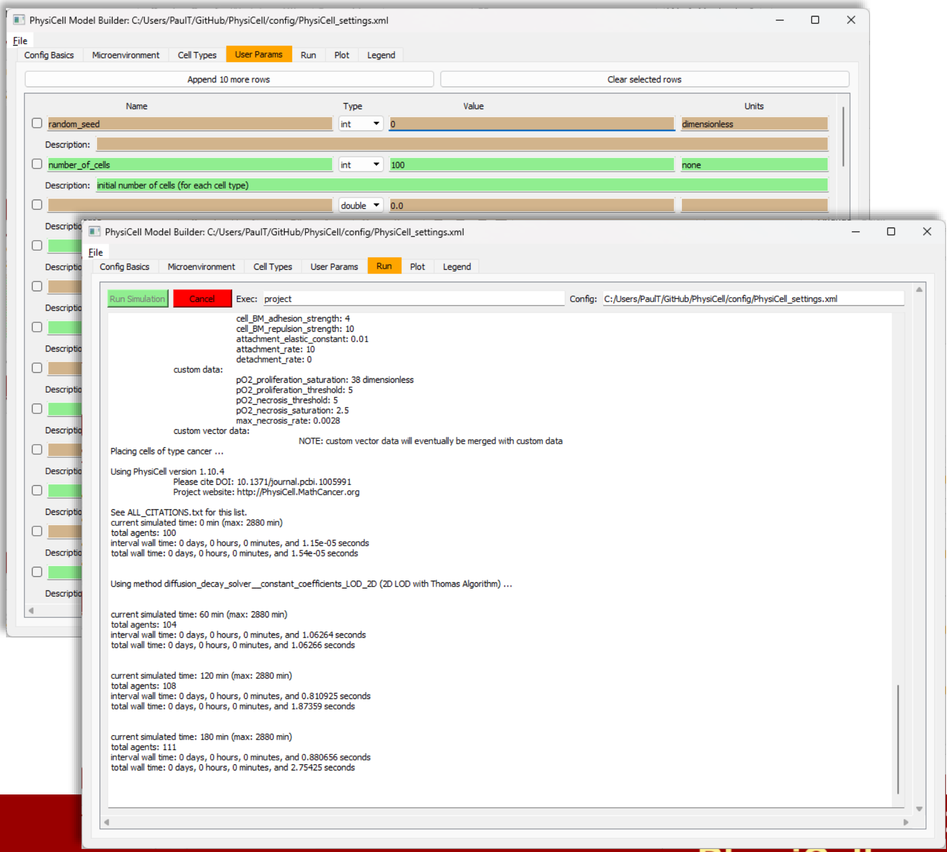
- Go to **custom data** tab
  - double-click **sample** and rename it to **pO2_proliferation_saturation**
    - ♦ Set it to **38**
  - Add **pO2_proliferation_threshold** = **5**
  - Add **pO2_necrosis_threshold** = **5**
  - Add **pO2_necrosis_saturation** = **2.5**
  - Add **max_necrosis_rate** = **2.8e-3**



Conserved	Name	Value (CSV vector allowed)	Units
☐	pO2_proliferation_saturation	38	dimensionless
	Description (optional)		
☐	pO2_proliferation_threshold	5	
	Description (optional)		
☐	pO2_necrosis_threshold	5	
	Description (optional)		
☐	pO2_necrosis_saturation	2.5	
	Description (optional)		
☐	max_necrosis_rate	2.8e-3	
	Description (optional)		

Test the model!

- Go to the **user params** tab
  - Set **number of cells** to **100**
    - Randomly place 100 of each cell type in the domain
- Save the config file
  - (overwrite PhysiCell_settings.xml)
- Go to the **run** tab
  - Set the **exec** name to **project**
- Click the **run** button



Name	Type	Value	Units
random_seed	int	0	dimensionless
Description:			
number_of_cells	int	100	none
Description: initial number of cells (for each cell type)			
	double	0.0	

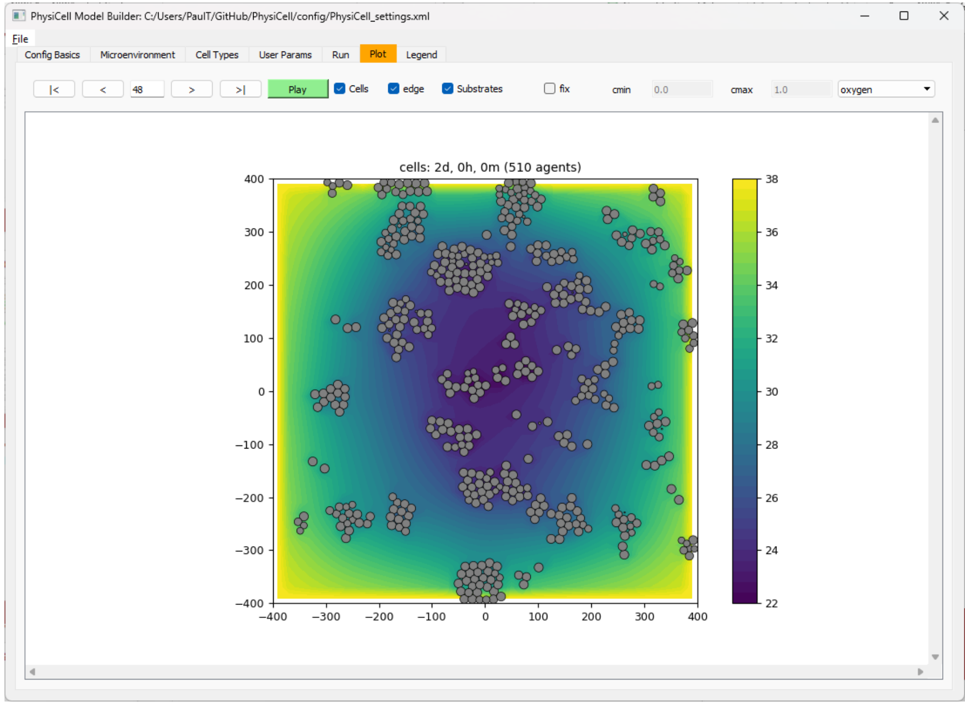
Exec: project Config: C:/Users/PaulT/GitHub/PhysiCell/config/PhysiCell_settings.xml

```
                        cell_BM_adhesion_strength: 4
                        cell_BM_repulsion_strength: 10
                        attachment_elastic_constant: 0.01
                        attachment_rate: 10
                        detachment_rate: 0
            custom data:
                        pO2_proliferation_saturation: 38 dimensionless
                        pO2_proliferation_threshold: 5
                        pO2_necrosis_threshold: 5
                        pO2_necrosis_saturation: 2.5
                        max_necrosis_rate: 0.0028
            custom vector data:
                                    NOTE: custom vector data will eventually be merged with custom data
Placing cells of type cancer ...

Using PhysiCell version 1.10.4
            Please cite DOI: 10.1371/journal.pcbi.1005991
            Project website: http://PhysiCell.MathCancer.org

See ALL_CITATIONS.txt for this list.
current simulated time: 0 min (max: 2880 min)
total agents: 100
interval wall time: 0 days, 0 hours, 0 minutes, and 1.15e-05 seconds
total wall time: 0 days, 0 hours, 0 minutes, and 1.54e-05 seconds


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


current simulated time: 60 min (max: 2880 min)
total agents: 104
interval wall time: 0 days, 0 hours, 0 minutes, and 1.06264 seconds
total wall time: 0 days, 0 hours, 0 minutes, and 1.06266 seconds


current simulated time: 120 min (max: 2880 min)
total agents: 108
interval wall time: 0 days, 0 hours, 0 minutes, and 0.810925 seconds
total wall time: 0 days, 0 hours, 0 minutes, and 1.87359 seconds


current simulated time: 180 min (max: 2880 min)
total agents: 111
interval wall time: 0 days, 0 hours, 0 minutes, and 0.880656 seconds
total wall time: 0 days, 0 hours, 0 minutes, and 2.75425 seconds
```

View results

- Go to the **plot** tab
  - click the **>** button to advance 1 frame
  - click **>|** to advance to the end
  - click **|<** to rewind to the start
  - click **play** to animate

- Check the **substrate** box to include a contour plot
  - choose **oxygen** in the drop-down menu

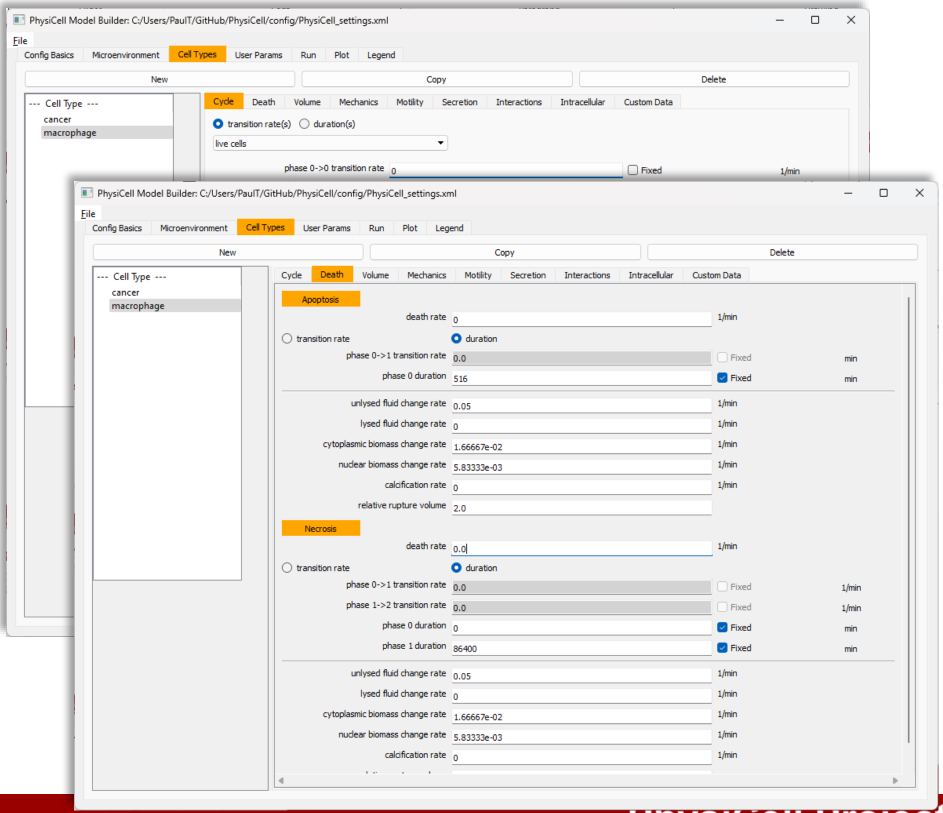

PhysiCell Model Builder: C:/Users/PaulT/GitHub/PhysiCell/config/PhysiCell_settings.xml

Checklist

- Plan ☑
- Build iteratively (in model builder):
  - Set the domain ☑
  - Add diffusing substrates ☑
  - Add cancer cells and test ☑
  - Add macrophages and test ☐
- Refinement (in C++):
  - Declare and write cancer cell phenotype ☐
  - Assign function ☐
  - Compile and test ☐

Create macrophages: birth and death

- Click the **cell types** tab
- Click **cancer**, copy
  - rename to **macrophage**
  - go to **cycle**
    - ♦ Set the 0→0 transition rate to **0**
  - Go to **death**
    - ♦ Set the **apoptosis** rate to **0**


PhysiCell Model Builder: C:/Users/PaulT/GitHub/PhysiCell/config/PhysiCell_settings.xml

(● transition rate(s) ○ duration(s))

live cells

phase 0->0 transition rate 0 ☐ Fixed 1/min

PhysiCell Model Builder: C:/Users/PaulT/GitHub/PhysiCell/config/PhysiCell_settings.xml

Apoptosis

death rate 0 1/min

○ transition rate ● duration

phase 0->1 transition rate 0.0 ☐ Fixed min

phase 0 duration 516 ☑ Fixed min

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 1/min

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

phase 0 duration 0 ☑ Fixed min

phase 1 duration 86400 ☑ Fixed min

unlysed fluid change rate 0.05 1/min

lysed fluid change rate 0 1/min

cytoplasmic biomass change rate 1.66667e-02 1/min

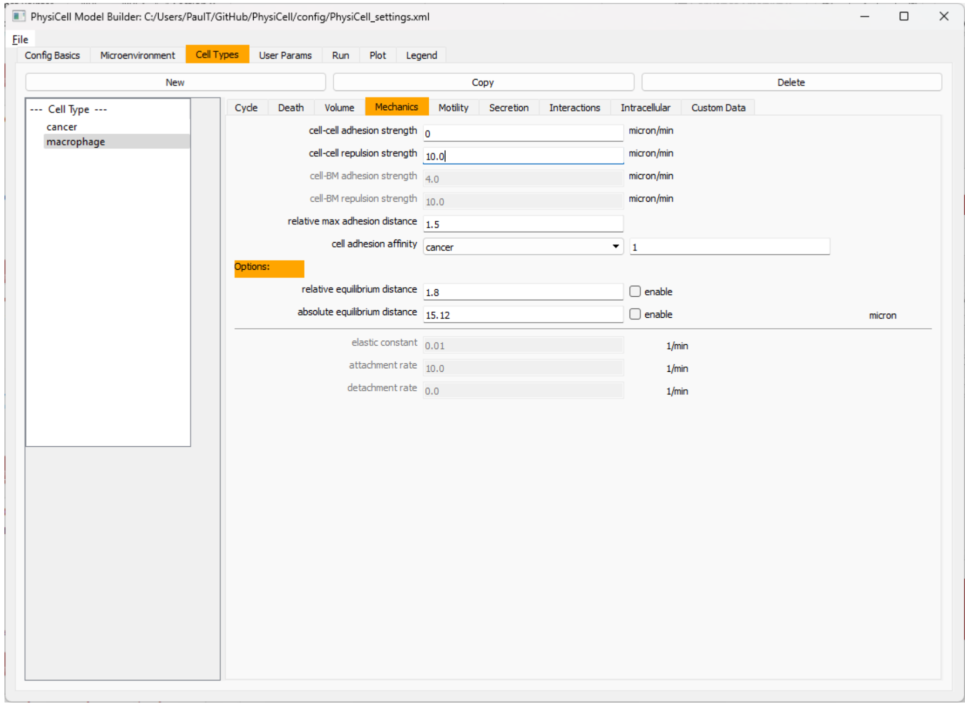
nuclear biomass change rate 5.83333e-03 1/min

calcification rate 0 1/min

Create macrophages: mechanics

- Let's adjust mechanics
  - No adhesion
  - Longer interaction distance

- Click the **mechanics** tab
  - Set **cell-cell adhesion strength** to 0
  - Set **relative max adhesion distance** to 1.5



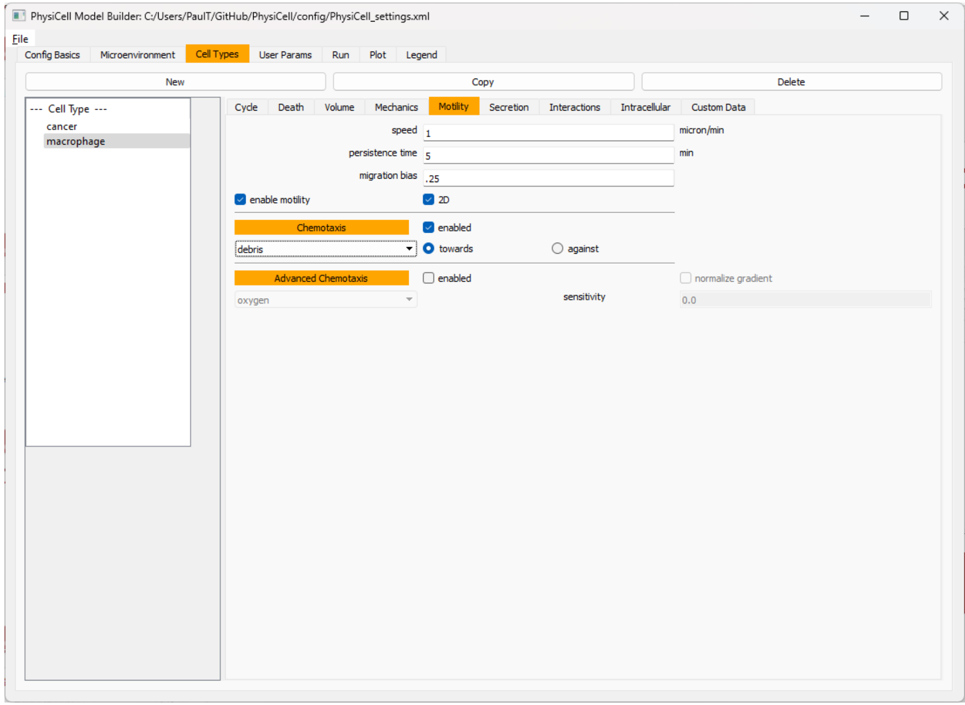
cell-cell adhesion strength	0	micron/min
cell-cell repulsion strength	10.0	micron/min
cell-BM adhesion strength	4.0	micron/min
cell-BM repulsion strength	10.0	micron/min
relative max adhesion distance	1.5	
cell adhesion affinity	cancer	1

Options:

relative equilibrium distance	1.8	☐ enable	
absolute equilibrium distance	15.12	☐ enable	micron
elastic constant	0.01	1/min	
attachment rate	10.0	1/min	
detachment rate	0.0	1/min	

Create macrophages: chemotaxis

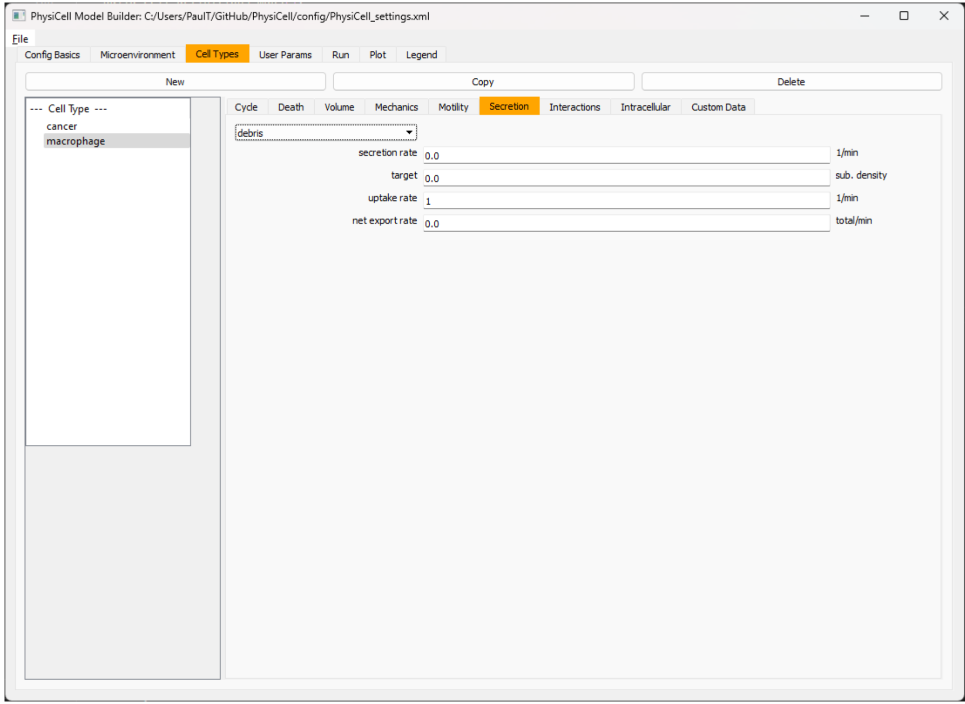
- Let's enable chemotaxis towards debris
- Click the **motility** tab
  - Check **enable motility**
  - Set **bias** to **0.25**
  - Set **persistence time** to **5**
  - Go to **chemotaxis**
    - Check **enabled**
    - Choose **debris** from the drop-down list



speed 1 micron/min
persistence time 5 min
migration bias .25
[✓] enable motility [✓] 2D
Chemotaxis: [✓] enabled
debris (● towards ○ against)
Advanced Chemotaxis: [ ] enabled [ ] normalize gradient
oxygen sensitivity 0.0

Create macrophages: secretion

- Go to the **secretion** tab
  - Consume cell debris
    - ♦ choose **debris** in the drop-down
    - ♦ set the **uptake rate** to 1

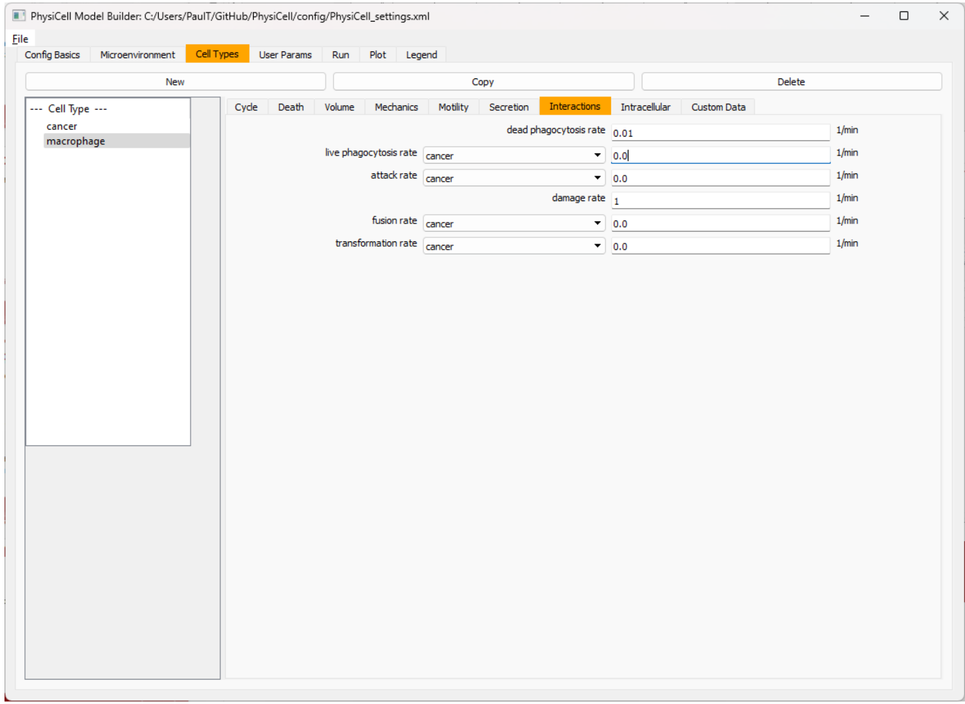


debris

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

Create macrophages: phagocytosis

- Let's enable phagocytosis of dead cells.
- Click the **interactions** tab
  - Set **dead phagocytosis rate** to **0.01**
- With this parameter, they will wait on average 100 minutes (1/rate) to phagocytose a dead cell in contact



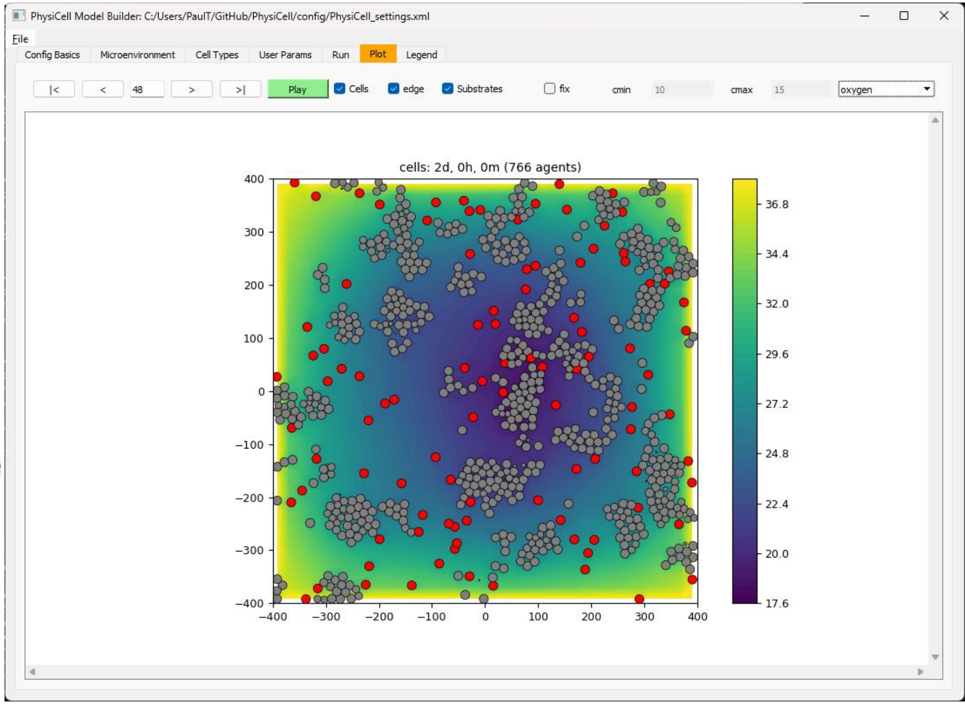
dead phagocytosis rate		0.01	1/min
live phagocytosis rate	cancer	0.0	1/min
attack rate	cancer	0.0	1/min
damage rate		1	1/min
fusion rate	cancer	0.0	1/min
transformation rate	cancer	0.0	1/min

Test the model

- Go to the **run** tab and click **run**
- Go to the **plot** tab
  - click **play** to animate
- View the **legend** tab to see the cell colors

●	cancer
🔴	M1 macrophage

- Expected behavior:
  - Tumor cell growth as before
  - Macrophages wander randomly
    - debris release not yet modeled!


PhysiCell Model Builder: C:/Users/PaulT/GitHub/PhysiCell/config/PhysiCell_settings.xml

Checklist

- Plan ✓
- Build iteratively (in model builder):
  - Set the domain ✓
  - Add diffusing substrates ✓
  - Add cancer cells and test ✓
  - Add macrophages and test ✓
- Refinement (in C++):
  - Declare and write cancer cell phenotype ☐
  - Assign function ☐
  - Compile and test ☐

Unzip Session06_checkpoint1.zip in ./PhysiCell to get this code.

Declare custom functions

- In **`./custom_modules/custom.h`**, declare:

```
void cancer_phenotype( Cell* pCell, Phenotype& p, double dt);
```

Custom phenotype rule (1)

```
void cancer_phenotype( Cell* pCell, Phenotype& p, double dt)
{
  // if dead, set secretion/uptake zero, release (export) debris
  // sample O2
  // set birth rate -- scale value from cell definition
  // set necrosis rate -- scale max value
}
```

Custom phenotype rule (2)

```
void cancer_phenotype( Cell* pCell, Phenotype& p, double dt)
{
   // if dead, set secretion/uptake zero, release (export) debris

   bool dead = (bool) get_single_signal( pCell, "dead");
   if( dead )
   {
      double volume = get_single_signal( pCell, "volume");

      set_single_behavior( pCell, "oxygen uptake" , 0.0 );
      set_single_behavior( pCell, "debris export" , 1*volume );

      return;
   }

   // sample O2
   // set birth rate -- scale value from cell definition
   // set necrosis rate -- scale max value
}
```

Custom phenotype rule (3)

```
void cancer_phenotype( Cell* pCell, Phenotype& p, double dt)
{
   // if dead, set secretion/uptake zero, release (export) debris

   // sample O2
   double o2 = get_single_signal( pCell, "oxygen");

   // set birth rate -- scale value from cell definition
   // set necrosis rate -- scale max value
}
```

Custom phenotype rule (4)

```
void cancer_phenotype( Cell* pCell, Phenotype& p, double dt)
{
   // if dead, set secretion/uptake zero, release (export) debris
   // sample O2

   // set birth rate -- scale value from cell definition

      // base birth rate
   double rate0 = get_single_base_behavior( pCell , "cycle entry");
      // scale based on o2
   double o2_sat = get_single_signal( pCell , "custom:pO2_proliferation_saturation");
   double o2_threshold =
      get_single_signal( pCell , "custom:pO2_proliferation_threshold");
   double rate = rate0 * linear_response_function( o2 , o2_threshold , o2_sat );
   set_single_behavior( pCell , "cycle entry" , rate );

   // set necrosis rate -- scale max value
}
```

Custom phenotype rule (5)

```
void cancer_phenotype( Cell* pCell, Phenotype& p, double dt)
{
   // if dead, set secretion/uptake zero, release debris
   // sample O2
   // set birth rate -- scale value from cell definition

   // set necrosis rate -- scale max value

      // get max rate
   double rateMax = get_single_behavior( pCell, "custom:max_necrosis_rate");
      // scale by O2
   o2_sat = get_single_behavior( pCell, "custom:pO2_necrosis_saturation");
   o2_threshold = get_single_behavior( pCell, "custom:pO2_necrosis_threshold");
   rate = rateMax * decreasing_linear_response_function( o2, o2_sat , o2_threshold );
   set_single_behavior( pCell, "necrosis" , rate );

   return;
}
```

Assign the 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( "cancer" );
   pCD->functions.update_phenotype = cancer_phenotype;

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

   // ...
```

Rebuild

- Open an additional terminal window in GitHub/PhysiCell

- Recompile:
  - **make**

Modify conditions and test the model

- In the **microenvironment** tab for **oxygen:**
  - Set the **initial condition** to **15**
  - Set the **Dirichlet BC** to **15**

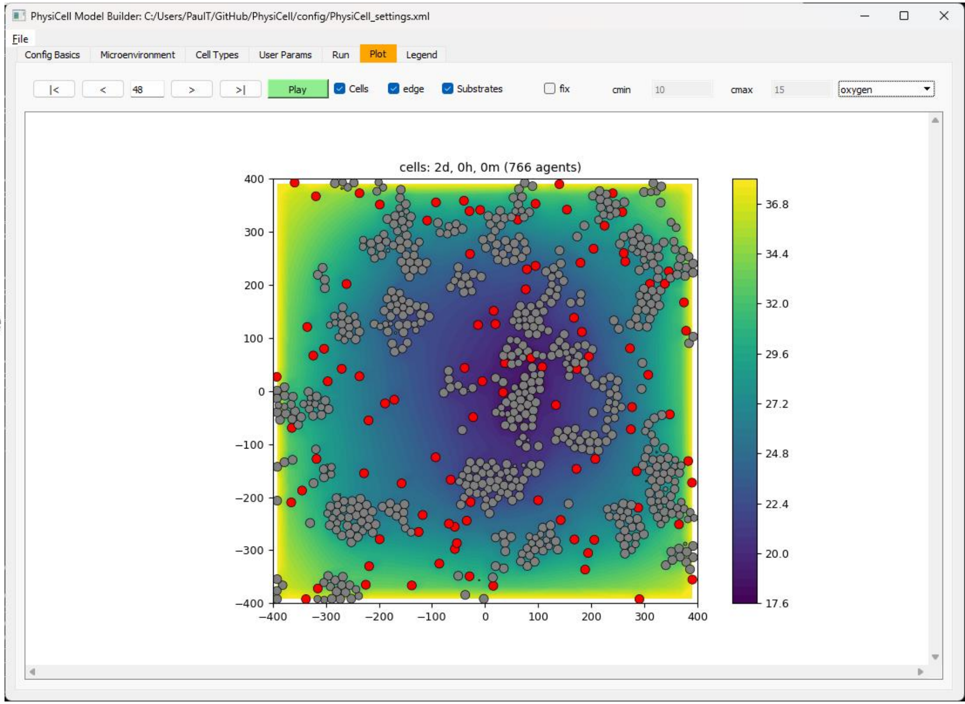
- Go to the **run** tab and click **run**

- Go to the **plot** tab
  - click **play** to animate

● cancer
● M1 macrophage

- View the **legend** tab to see the cell colors

- Expected behavior:
  - Tumor cell growth faster near outer edge
  - Macrophages wander towards dead cells
    - phagocytosis of dead cells (not yet visualized)



Unzip Session06_checkpoint2.zip
in ./PhysiCell to get this code.

Handy C++ Functions II

Handy C++ tidbits: creating cells

- Functions to help (properly) create and place new cells
  - `Cell* create_cell( void );`
    - Create a new **Cell** using the default cell definition (cell_defaults: has ID 0)
    - Returns a pointer to the cell, allowing you to further access and modify it
  - `Cell* create_cell( Cell_Definition& cd );`
    - Create a new **Cell** using supplied cell definition
    - Returns a pointer to the cell, allowing you to further access and modify it
  - `bool assign_position( std::vector<double> new_position);`
    - Use this if you want to manually set the cell's position.
    - Fully compatible with BioFVM and its data structures
    - Useful for initialization

Handy C++ Tidbits: Random Numbers

- `double UniformRandom( void );`
  - Get a uniformly distributed number in U(0,1)

- `double NormalRandom( double mean, double standard_deviation );`
  - Get a normally distributed number in N(mean, standard_deviation)

- `std::vector<double> UniformOnUnitCircle( void );`
  - Get a uniformly random point on the Unit Circle

- `std::vector<double> UniformOnUnitSphere( void );`
  - Get a uniformly random point on (not in!) the unit sphere.

- `int choose_event( std::vector<double>& probabilities );`
  - Given a vector of probabilities $(p_0, p_1, \dots, p_{n-1})$, choose an integer in [0,*n*-1] with the given probabilities.
  - The probabilities must sum to 1.

These use the STL 64-bit Mersenne Twister in C++11.

Refining our example: initial positions

- Tumor cells:
  - Randomly place *nC* cells within a disk of radius *radius_tumor*

- Macrophages:
  - Randomly place *nM* cells on a circle of radius *radius_macrophages*

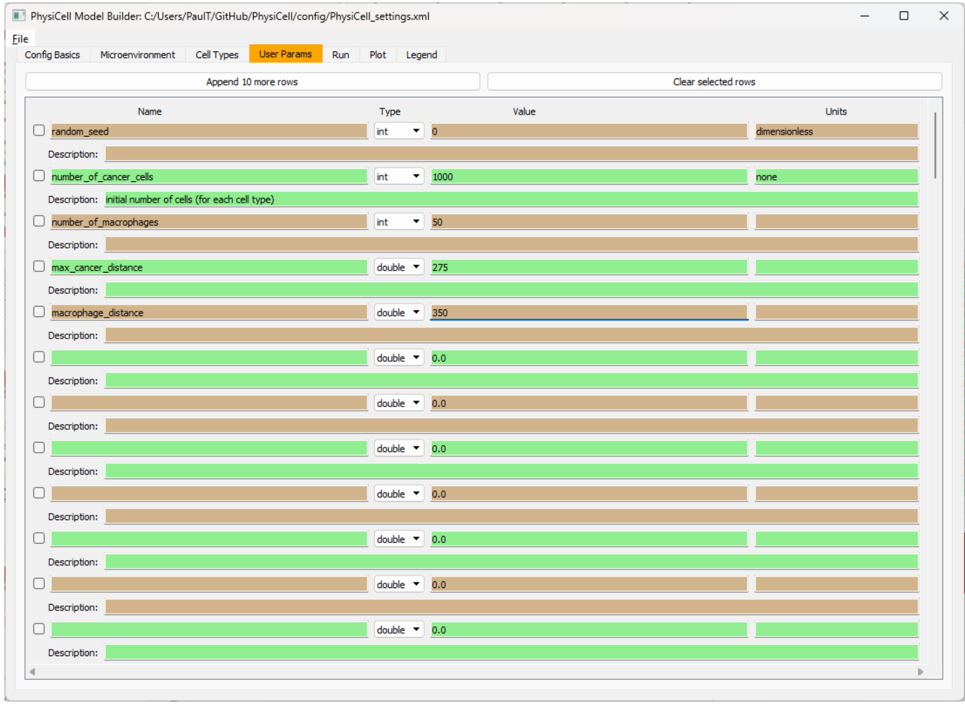
Full modeling workflow

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

Add new user parameters

- Go to **user parameters** in Model Builder
  - rename **number_of_cells** to **number_of_cancer_cells**
    - ♦ Set its value to 1000
    - ♦ Change its description to "initial number of tumor cells"
  - add another parameter called **number_of_macrophages**
    - ♦ set its type to int
    - ♦ set its value to 50
    - ♦ set units and description
  - add another parameter called **max_cancer_distance**
    - ♦ keep its type as double
    - ♦ set its value and units to 275 micron
    - ♦ set description to "max initial cell distance from origin"
  - Add one last parameter called **macrophage_distance**
    - ♦ Set its value at 350
- Resave



PhysiCell Model Builder: C:/Users/PaulT/GitHub/PhysiCell/config/PhysiCell_settings.xml

Name	Type	Value	Units	Description
random_seed	int	0	dimensionless	
number_of_cancer_cells	int	1000	none	initial number of cells (for each cell type)
number_of_macrophages	int	50		
max_cancer_distance	double	275		
macrophage_distance	double	350		
	double	0.0		
	double	0.0		
	double	0.0		
	double	0.0		
	double	0.0		
	double	0.0		
	double	0.0		

Edit setup_tissue (1)

- In **./custom_modules/custom.cpp** find **setup_tissue**
- Comment out the current placement code:

```
/*
   for( int k=0; k < cell_definitions_by_index.size() ; k++ )
   {
      Cell_Definition* pCD = cell_definitions_by_index[k];
      std::cout << "Placing cells of type " << pCD->name << " ... " << std::endl;
      for( int n = 0 ; n < parameters.ints("number_of_cells") ; n++ )
      {
         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( *pCD );
         pC->assign_position( position );
      }
   }
*/
```

Edit setup_tissue (2)

```
            pC->assign_position( position );
        }
    }
*/

    // place tumor cells
    double max_distance = parameters.doubles("max_initial_distance");

    Cell_Definition* pCD = find_cell_definition( "cancer" );
    std::cout << "Placing cells of type " << pCD->name << " ... " << std::endl;
    for( int k=0 ; k < parameters.ints( "number_of_cancer_cells" ); k++ )
    {
        std::vector<double> position = {0,0,0};
        double r = sqrt(UniformRandom())* max_distance;
        double theta = UniformRandom() * 6.2831853;
        position[0] = r*cos(theta);
        position[1] = r*sin(theta);

        pC = create_cell( *pCD );
        pC->assign_position( position );
    }
```

Edit setup_tissue (3)

```
   // place macrophages
   pCD = find_cell_definition( "macrophage" );

   std::cout << "Placing cells of type " << pCD->name << " ... " << std::endl;
   for( int k=0 ; k < parameters.ints( "number_of_macrophages" ); k++ )
   {
      std::vector<double> position = UniformOnUnitCircle();
      position *= parameters.doubles("macrophage_distance");

      pC = create_cell( *pCD );
      pC->assign_position( position );
   }

   std::cout << std::endl;

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

   return;
}
```

Rebuild and test the model

- In your terminal window, recompile:
  - **make**

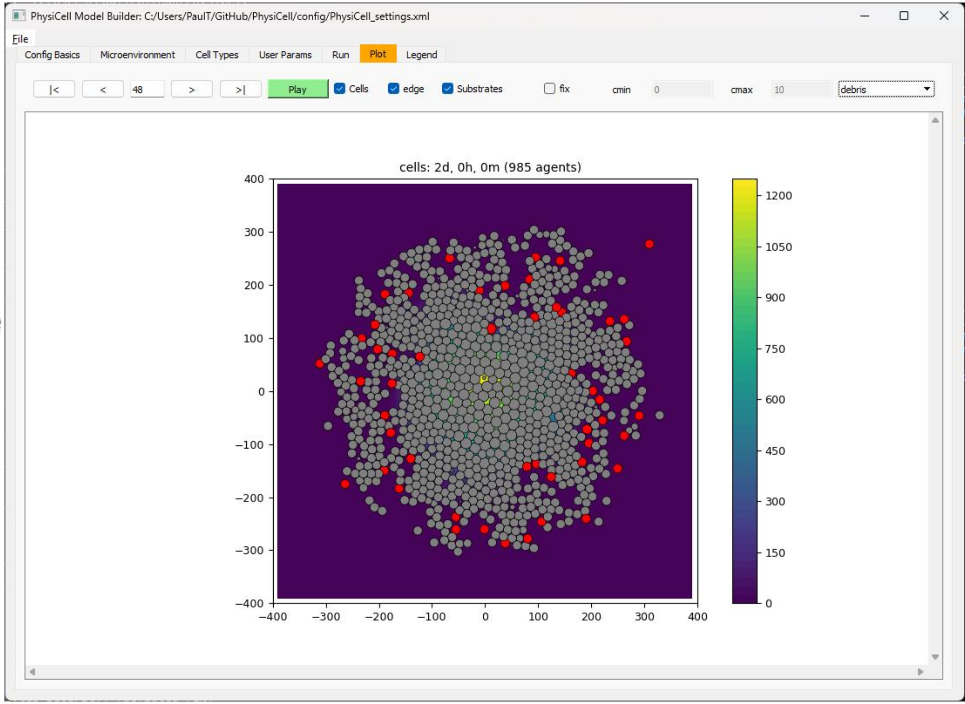
- Go to the **run** tab and click **run**

- Go to the **plot** tab
  - click **play** to animate

● cancer
🔴 M1 macrophage

- View the **legend** tab to see the cell colors

- Expected behavior:
  - Tumor cell growth faster near outer edge
  - Macrophages wander towards dead cells
    - phagocytosis of dead cells (not yet visualized) *if they can reach them*



Unzip Session06_checkpoint3.zip in ./PhysiCell to get this code.

Custom coloring functions for SVGs (1)

Declare the function in the custom header file

```
std::vector<std::string> custom_coloring_function( Cell* pCell );
```

Create it in the custom cpp file

```
std::vector<std::string> custom_coloring_function( Cell* pCell )
{
   // color 0: cytoplasm fill
   // color 1: outer outline
   // color 2: nuclear fill
   // color 3: nuclear outline

   // start with color-by-number
   // dead cells: black if apoptotic, brown if necrotic
   // live tumor cells: shade by proliferation rate
}
```

Coloring function (1)

```
std::vector<std::string> custom_coloring_function( Cell* pCell )
{
   // start with color-by-number

   std::vector<std::string> = paint_by_number_cell_coloring(pCell);

   // dead cells: black if apoptotic, brown if necrotic
   // live tumor cells: shade by proliferation rate
}
```

Coloring function (2)

```
std::vector<std::string> custom_coloring_function( Cell* pCell )
{
  // start with color-by-number

  // dead cells: black if apoptotic, brown if necrotic

  bool dead = (bool) get_single_signal( pCell, "dead");
  if( dead )
  {
    if( pCell->phenotype.cycle.model().name == "Apoptosis" )
    { output[0] = "rgb(0,0,0)"; }
    else
    { output[0] = "rgb(111,78,55)"; }
  }

  // live tumor cells: shade by proliferation rate
}
```

Coloring function (3)

```
std::vector<std::string> custom_coloring_function( Cell* pCell )
{
   // start with color-by-number
   // dead cells: black if apoptotic, brown if necrotic

   // live tumor cells: shade by proliferation rate

   if( pCell->type_name == "cancer" && dead == false )
   {
      // get relative birth rate
      double s = 10 * get_single_behavior( pCell, "cycle entry" )
         / get_single_base_behavior( pCell, "cycle entry" );
      if( s > 1 )
      { s = 1; }

      // make color
      int color = (int) round( 255.0 * s );
      char szColor [1024];
      // interpolate from blue to yellow
      sprintf( szColor, "rgb(%u,%u,%u)",color,color,255-color );

      // modify output
      output[0] = szColor;
      output[2] = szColor;
      output[3] = szColor;
   }

   return output;
}
```

Tell PhysiCell to use your coloring function

In main.cpp

```
std::vector<std::string> (*cell_coloring_function)(Cell*) =
   custom_coloring_function;
```

Colors follow the W3C standards for SVG files. Names, RGB values, etc.

https://www.w3.org/TR/SVG11/types.html#ColorKeywords

User Guide: Section 14.2

Unzip Session06_checkpoint4.zip in ./PhysiCell to get this code.

Rebuild and test the model

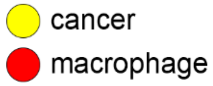
- In your terminal window, recompile:
  - **make**

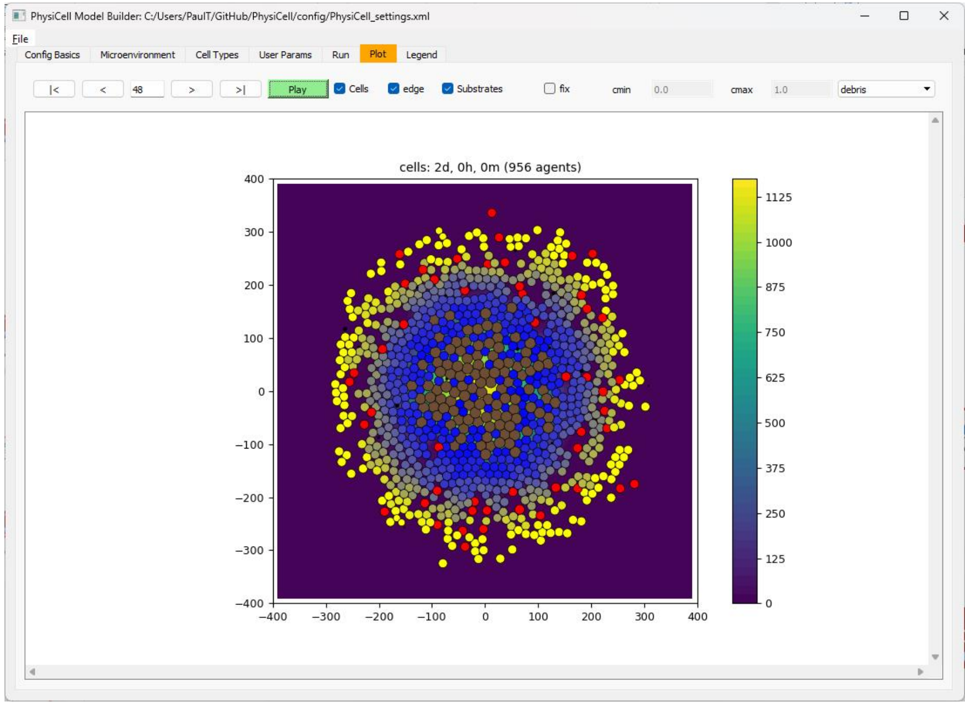
- Go to the **run** tab and click **run**

- Go to the **plot** tab
  - click **play** to animate

- View the **legend** tab to see the cell colors

- Expected behavior:
  - Tumor cell growth faster near outer edge
    - Bright yellow = rapidly proliferating
  - Macrophages wander towards dead cells
    - necrotic core (brown), sporadic apoptosis (black)
    - phagocytosis of dead cells *if they can reach them*





Unzip [Session06_checkpoint4.zip]
in ./PhysiCell to get this code.

Funding Acknowledgements

PhysiCell Development:

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

Training Materials:

- Administrative supplement to NCI U01CA232137 (Year 2)

Other Funding:

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