

Slides, videos, links and more:

<https://github.com/physicell-training/ws2022>

Session 1: Working with PhysiCell Projects

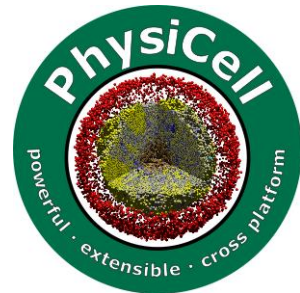


Paul Macklin, Ph.D.

 @MathCancer

PhysiCell Project

July 17, 2022



LUDDY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

PhysiCell Project

PhysiCell.org

 @PhysiCell

Goals

- Learn how to work with sample projects
 - Get a list of sample projects
 - Populate a project
 - Look at typical project structure
 - Modify settings
 - Compile and run a populated project
 - See typical model outputs
 - Clear out data and reset

Sample projects

- It's inefficient (and a little insane) to code new projects *entirely* from scratch.
- So, we provide sample projects:
 - 2D/3D template project
 - Cancer models
 - Synthetic multicellular systems
 - Viral dynamics in tissue
 - and more ...
- **make [project-name]**: populate a sample project (puts all the source files where they belong)
 - Then use **make** to compile it
- **make data-cleanup**: clean up the output data
- **make reset**: return to a "clean slate" (depopulate the project)
- **make list-projects**: display all available sample projects

Documentation: User Guide Sections 6, 7.

PhysiCell Project Essentials (1)

- Each PhysiCell release includes sample projects. To list them:

- **make list-projects**

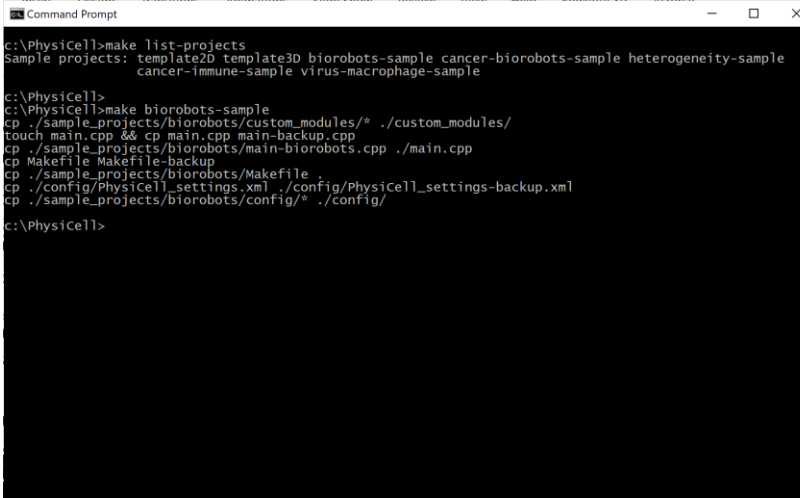
- Your first step is to **populate a project**.

- **make <project_name>**

- Let's use biorobots-sample:

- ♦ **make biorobots-sample**

- This copies source code, a tailored make file, and configuration files



```
Command Prompt
c:\PhysiCell>make list-projects
Sample projects: template2D template3D biorobots-sample cancer-biorobots-sample heterogeneity-sample
cancer-immune-sample virus-macrophage-sample

c:\PhysiCell>
c:\PhysiCell>make biorobots-sample
cp ./sample_projects/biorobots/custom_modules/* ./custom_modules/
touch main.cpp && cp main.cpp main-backup.cpp
cp ./sample_projects/biorobots/main-biorobots.cpp ./main.cpp
cp Makefile Makefile-backup
cp ./sample_projects/biorobots/Makefile .
cp ./config/PhysiCell_settings.xml ./config/PhysiCell_settings-backup.xml
cp ./sample_projects/biorobots/config/* ./config/

c:\PhysiCell>
```

Let's look at the project structure ...



LUDDY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

PhysiCell Project

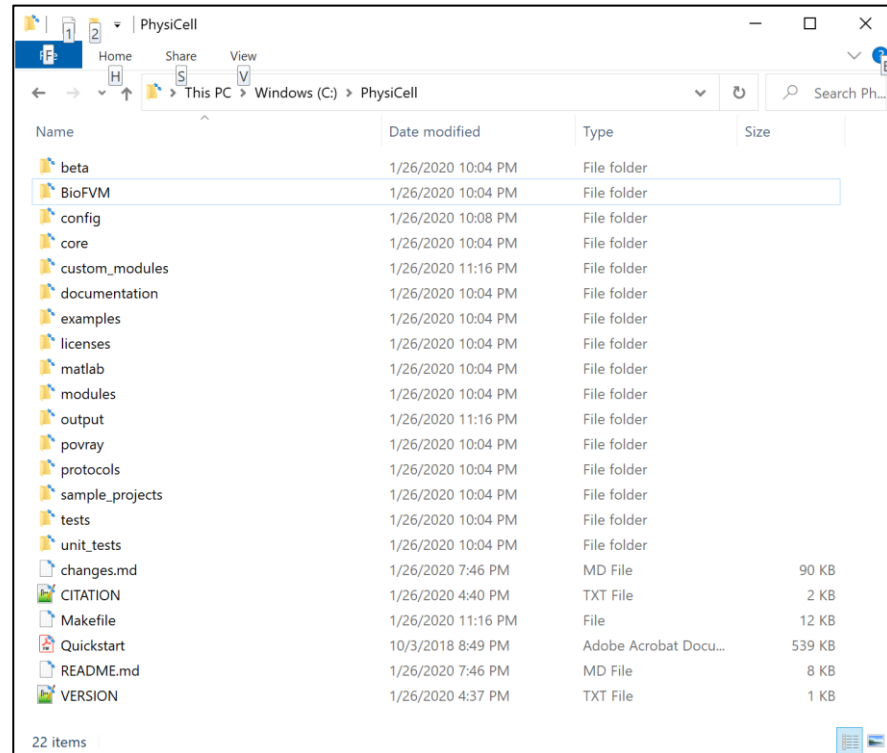
PhysiCell.org

 @PhysiCell

Project directory structure

- (key) directories:
 - **./ (root):** main source, Makefile, and executable go here
 - **./beta:** for beta-testing (don't use)
 - **./BioFVM:** diffusion solver
 - **./config:** configuration files
 - **./core:** PhysiCell core functions
 - **./custom_modules:** put custom code for your project here.
 - **./documentation:** user guide, etc.
 - **./examples:** deprecated
 - **./licenses:** yep
 - **./matlab:** scripts and functions to load data in matlab
 - **./modules:** standard add-ons for PhysiCell
 - **./addons:** officially supported addons like PhysiBoSS and libRoadrunner
 - **./output:** where data are stored (by default, but can be changed)
 - **./povray:** deprecated
 - **./protocols:** instructions mostly for maintainers (e.g., release protocols)
 - **./sample_projects:** where we add sample projects
 - **./tests:** for automated testing (WIP)
 - **./unit_tests:** for automated testing (WIP)

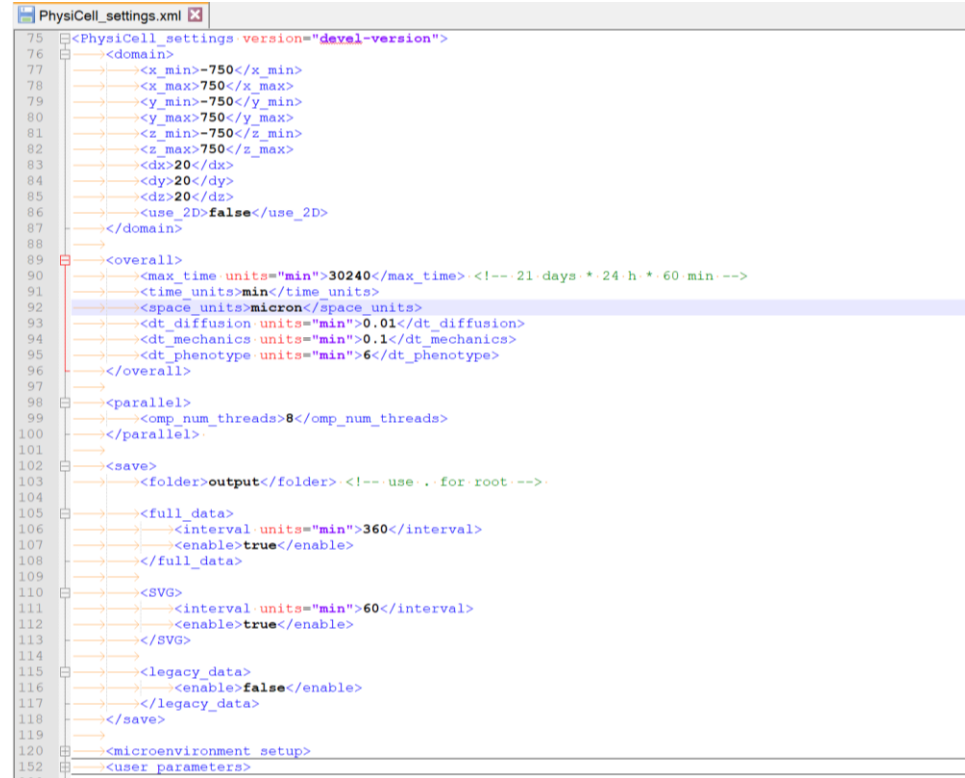
Most of your work will be in the **red directories**



Name	Date modified	Type	Size
beta	1/26/2020 10:04 PM	File folder	
BioFVM	1/26/2020 10:04 PM	File folder	
config	1/26/2020 10:08 PM	File folder	
core	1/26/2020 10:04 PM	File folder	
custom_modules	1/26/2020 11:16 PM	File folder	
documentation	1/26/2020 10:04 PM	File folder	
examples	1/26/2020 10:04 PM	File folder	
licenses	1/26/2020 10:04 PM	File folder	
matlab	1/26/2020 10:04 PM	File folder	
modules	1/26/2020 10:04 PM	File folder	
output	1/26/2020 11:16 PM	File folder	
povray	1/26/2020 10:04 PM	File folder	
protocols	1/26/2020 10:04 PM	File folder	
sample_projects	1/26/2020 10:04 PM	File folder	
tests	1/26/2020 10:04 PM	File folder	
unit_tests	1/26/2020 10:04 PM	File folder	
changes.md	1/26/2020 7:46 PM	MD File	90 KB
CITATION	1/26/2020 4:40 PM	TXT File	2 KB
Makefile	1/26/2020 11:16 PM	File	12 KB
Quickstart	10/3/2018 8:49 PM	Adobe Acrobat Docu...	539 KB
README.md	1/26/2020 7:46 PM	MD File	8 KB
VERSION	1/26/2020 4:37 PM	TXT File	1 KB

Project structure: config files

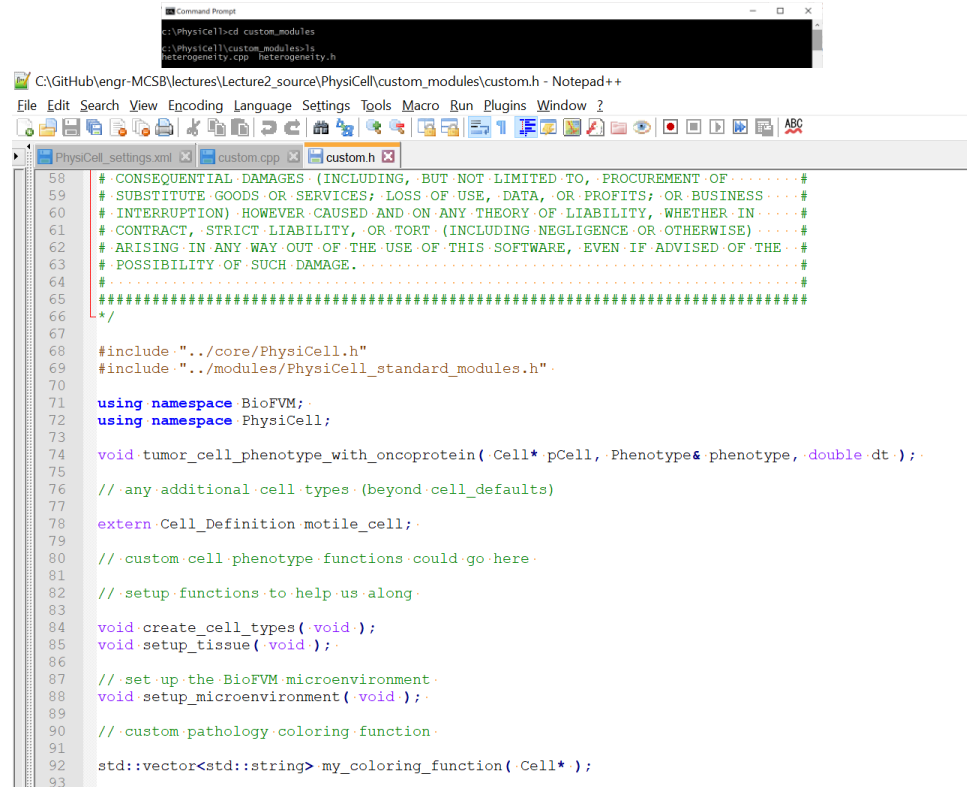
- Configuration files (XML)
 - **domain:** domain size and resolution
 - **overall:** general options
 - ◆ Final simulation time
 - ◆ Time step sizes
 - **parallel:** parallelization options
 - ◆ Number of threads
 - **save:** save options
 - ◆ Save where?
 - ◆ Save SVGs? (how often?)
 - ◆ Save full data? (how often?)
 - ◆ Save legacy data (don't)
 - **microenvironment_setup:** diffusion settings
 - ◆ more later
 - **cell_definitions:** define different cell types and starting parameters
 - ◆ more later
 - **user_parameters:** simulation-specific settings
 - ◆ more later



```
PhysiCell_settings.xml
75 <PhysiCell_settings version="devel-version">
76   <domain>
77     <x_min>-750</x_min>
78     <x_max>750</x_max>
79     <y_min>-750</y_min>
80     <y_max>750</y_max>
81     <z_min>-750</z_min>
82     <z_max>750</z_max>
83     <dx>20</dx>
84     <dy>20</dy>
85     <dz>20</dz>
86     <use_2D>false</use_2D>
87   </domain>
88
89   <overall>
90     <max_time units="min">30240</max_time> <!-- 21.days * 24.h * 60.min -->
91     <time_units>min</time_units>
92     <space_units>micron</space_units>
93     <dt_diffusion units="min">0.01</dt_diffusion>
94     <dt_mechanics units="min">0.1</dt_mechanics>
95     <dt_phenotype units="min">6</dt_phenotype>
96   </overall>
97
98   <parallel>
99     <comp_num_threads>8</comp_num_threads>
100   </parallel>
101
102   <save>
103     <folder>output</folder> <!-- use . for root -->
104
105     <full_data>
106       <interval units="min">360</interval>
107       <enable>true</enable>
108     </full_data>
109
110     <SVG>
111       <interval units="min">60</interval>
112       <enable>true</enable>
113     </SVG>
114
115     <legacy_data>
116       <enable>false</enable>
117     </legacy_data>
118   </save>
119
120   <microenvironment_setup>
121   </microenvironment_setup>
122
123   <user_parameters>
```

Project structure: custom modules

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



The screenshot shows a code editor window with the file `custom.h` open. The code defines a custom module for PhysiCell, including headers, namespace declarations, and function prototypes. A terminal window above shows the command `cd custom_modules` being executed.

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

C:\GitHub\enrg-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
58  /* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF .....#
59  /* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS .....#
60  /* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN .....#
61  /* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) .....#
62  /* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE .....#
63  /* POSSIBILITY OF SUCH DAMAGE. ....#
64  /* .....#
65  /* .....#
66  */
67
68  #include "../core/PhysiCell.h"
69  #include "../modules/PhysiCell_standard_modules.h"
70
71  using namespace BioFVM;
72  using namespace PhysiCell;
73
74  void tumor_cell_phenotype_with_oncoprotein( Cell* pCell, Phenotype& phenotype, double dt );
75
76  // any additional cell types (beyond cell_defaults)
77
78  extern Cell_Definition motile_cell;
79
80  // custom cell phenotype functions could go here
81
82  // setup functions to help us along
83
84  void create_cell_types( void );
85  void setup_tissue( void );
86
87  // set up the BioFVM microenvironment
88  void setup_microenvironment( void );
89
90  // custom pathology coloring function
91
92  std::vector<std::string> my_coloring_function( Cell* );
93
```


Project structure: custom modules

- Custom Modules

- Any user-defined globals (at top)

- ◆ Declared cell types

- Setup functions

- ◆ `create_cell_types()`

- » Do all setup on all cell types
 - Adjust phenotype
 - Add / adjust custom data
 - Set functions

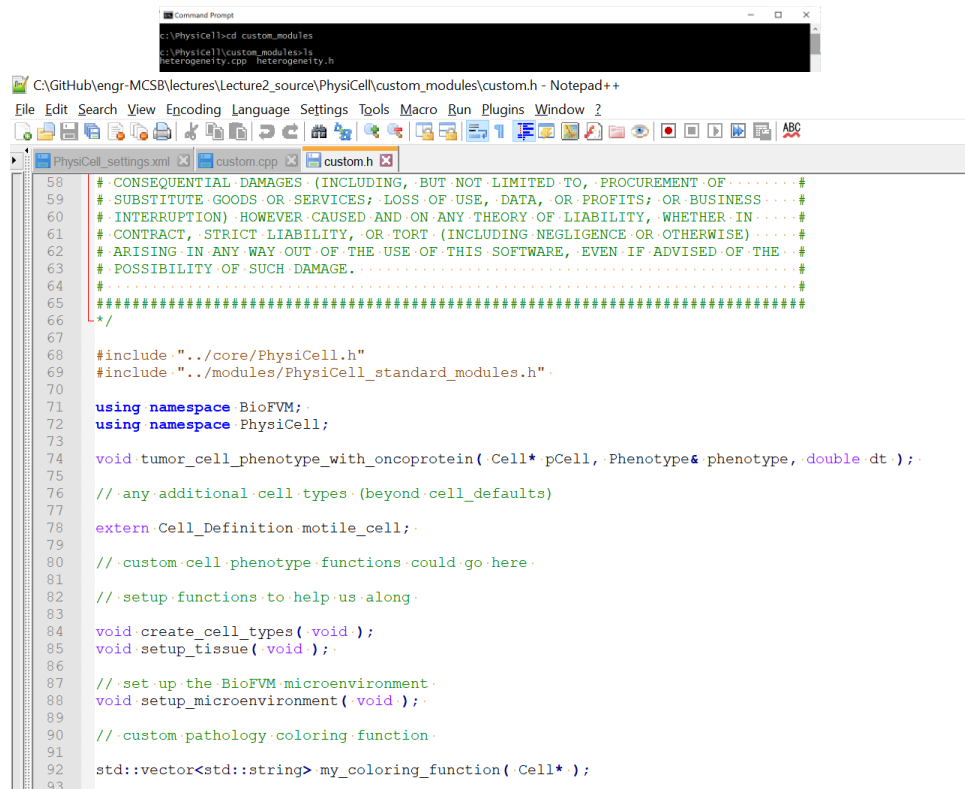
- ◆ `setup_tissue()`

- » Place initial cells in microenvironment
- » Modify each cell as needed

- Custom functions

- any other modeling

- Custom coloring functions

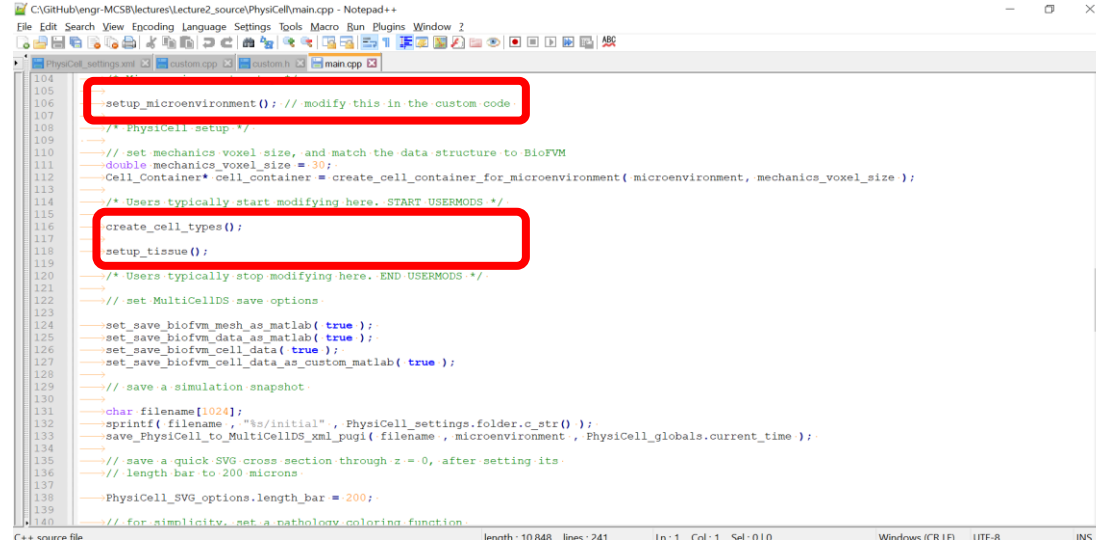


```
58  * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF ..... *
59  * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS ..... *
60  * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN ..... *
61  * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ..... *
62  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE ..... *
63  * POSSIBILITY OF SUCH DAMAGE. .... *
64  * ..... *
65  * ..... *
66  */
67
68  #include "../core/PhysiCell.h"
69  #include "../modules/PhysiCell_standard_modules.h"
70
71  using namespace BioFVM;
72  using namespace PhysiCell;
73
74  void tumor_cell_phenotype_with_oncoprotein( Cell* pCell, Phenotype& phenotype, double dt );
75
76  // any additional cell types (beyond cell_defaults)
77
78  extern Cell_Definition motile_cell;
79
80  // custom cell phenotype functions could go here
81
82  // setup functions to help us along
83
84  void create_cell_types( void );
85  void setup_tissue( void );
86
87  // set up the BioFVM microenvironment
88  void setup_microenvironment( void );
89
90  // custom pathology coloring function
91
92  std::vector<std::string> my_coloring_function( Cell* );
93
```

Project structure: main.cpp

- **main.cpp**

- (in the root directory)
- calls the setup functions

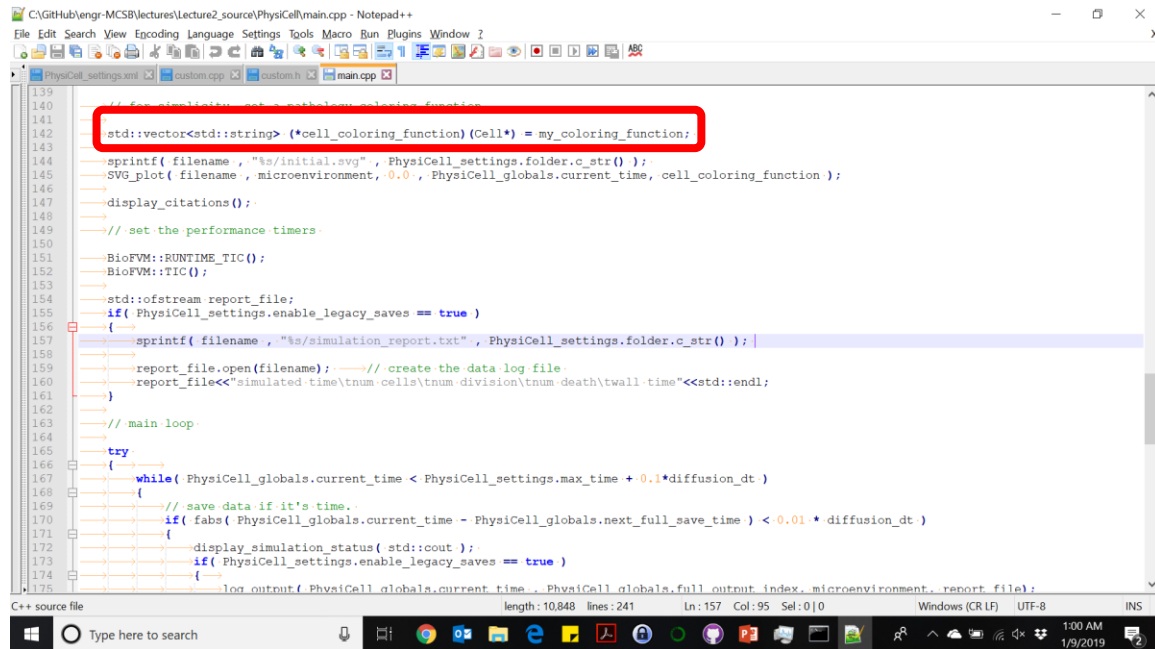


```
104  setup_microenvironment(); // modify this in the custom code
105
106  /* PhysiCell setup */
107
108  // set mechanics voxel size, and match the data structure to BioFVM
109  double mechanics_voxel_size = 30;
110  Cell_Container* Cell_container = create_cell_container_for_microenvironment( microenvironment, mechanics_voxel_size );
111
112  /* Users typically start modifying here. START USERMODS */
113
114  create_cell_types();
115
116  setup_tissue();
117
118  /* Users typically stop modifying here. END USERMODS */
119
120  // set MultiCellDS save options
121
122  set_save_biofvm_mesh_as_matlab( true );
123  set_save_biofvm_data_as_matlab( true );
124  set_save_biofvm_cell_data( true );
125  set_save_biofvm_cell_data_as_custom_matlab( true );
126
127  // save a simulation snapshot
128
129  char filename[1024];
130  sprintf( filename, "%s/initial", PhysiCell_settings.folder.c_str() );
131  save_PhysiCell_to_MultiCellDS_xml_pugi( filename, microenvironment, PhysiCell_globals.current_time );
132
133  // save a quick SVG cross section through z = 0, after setting its
134  // length bar to 200 microns
135
136  PhysiCell_SVG_options.length_bar = 200;
137
138  // for simplicity, set a pathology coloring function
139
140
```

Project structure: main.cpp (continued)

- **main.cpp**

- set coloring function

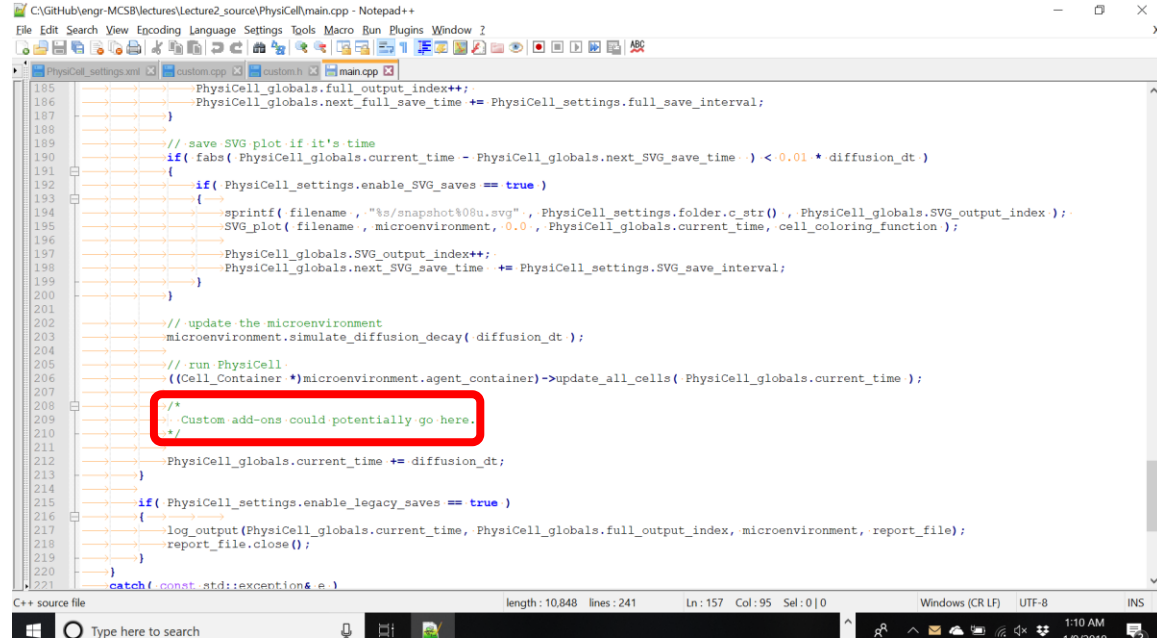


```
139 // for simplicity, set a pathless coloring function
140
141 std::vector<std::string> (*cell_coloring_function) (Cell*) = my_coloring_function;
142
143
144 sprintf( filename , "%s/initial.svg" , PhysiCell_settings.folder.c_str() );
145 SVG_plot( filename , microenvironment, 0.0 , PhysiCell_globals.current_time , cell_coloring_function );
146
147 display_citations();
148
149 // set the performance timers
150
151 BioFVM::RUNTIME_TIC();
152 BioFVM::TIC();
153
154 std::ofstream report_file;
155 if( PhysiCell_settings.enable_legacy_saves == true )
156 {
157     sprintf( filename , "%s/simulation_report.txt" , PhysiCell_settings.folder.c_str() );
158     report_file.open(filename); // create the data log file
159     report_file<<"simulated time\tnum. cells\tnum. division\tnum. death\twall time"<<std::endl;
160 }
161
162 // main loop
163
164 try
165 {
166     while( PhysiCell_globals.current_time < PhysiCell_settings.max_time + 0.1*diffusion_dt )
167     {
168         // save data if it's time
169         if( fabs( PhysiCell_globals.current_time - PhysiCell_globals.next_full_save_time ) < 0.01*diffusion_dt )
170         {
171             display_simulation_status( std::cout );
172             if( PhysiCell_settings.enable_legacy_saves == true )
173             {
174                 log_output( PhysiCell_globals.current_time , PhysiCell_globals.full_output_index , microenvironment , report_file );
175             }
176         }
177     }
178 }
```

Project structure: main.cpp (continued)

- **main.cpp**

- insert custom routines
- **This would be a good place to put extensions.**



```
C:\GitHub\enr-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
185 PhysiCell_globals.full_output_index++;
186 PhysiCell_globals.next_full_save_time += PhysiCell_settings.full_save_interval;
187 }
188
189 // save SVG plot if it's time
190 if ( fabs( PhysiCell_globals.current_time - PhysiCell_globals.next_SVG_save_time ) < 0.01 * diffusion_dt )
191 {
192     if ( PhysiCell_settings.enable_SVG_saves == true )
193     {
194         sprintf( filename , "%s/snapshot%08u.svg" , PhysiCell_settings.folder.c_str() , PhysiCell_globals.SVG_output_index );
195         SVG_plot( filename , microenvironment , 0.0 , PhysiCell_globals.current_time , cell_coloring_function );
196
197         PhysiCell_globals.SVG_output_index++;
198         PhysiCell_globals.next_SVG_save_time += PhysiCell_settings.SVG_save_interval;
199     }
200 }
201
202 // update the microenvironment
203 microenvironment.simulate_diffusion_decay( diffusion_dt );
204
205 // run PhysiCell
206 (Cell_Container *)microenvironment.agent_container->update_all_cells( PhysiCell_globals.current_time );
207
208 /*
209  Custom add-ons could potentially go here.
210 */
211
212 PhysiCell_globals.current_time += diffusion_dt;
213 }
214
215 if ( PhysiCell_settings.enable_legacy_saves == true )
216 {
217     log_output( PhysiCell_globals.current_time , PhysiCell_globals.full_output_index , microenvironment , report_file );
218     report_file.close();
219 }
220
221 catch( const std::exception& e )
```

**Now, let's get back to
working with sample
projects.**



LUDDY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

PhysiCell Project

PhysiCell.org

 **@PhysiCell**

- Now, **compile the project**

- Then, **run the project**

- **./biorobots** (Linux, MacOS)
- **biorobots.exe** (Windows)

- This should take about 5 minutes

```

C:\Command Prompt - biobots
current simulated time: 12 min (max: 2880 min)
total agents: 513
interval wall time: 0 days, 0 hours, 0 minutes, and 0.189681 seconds
total wall time: 0 days, 0 hours, 0 minutes, and 1.17758 seconds

current simulated time: 14 min (max: 2880 min)
total agents: 513
interval wall time: 0 days, 0 hours, 0 minutes, and 0.172898 seconds
total wall time: 0 days, 0 hours, 0 minutes, and 1.35119 seconds

current simulated time: 16 min (max: 2880 min)
total agents: 513
interval wall time: 0 days, 0 hours, 0 minutes, and 0.169814 seconds
total wall time: 0 days, 0 hours, 0 minutes, and 1.52185 seconds

current simulated time: 18 min (max: 2880 min)
total agents: 513
interval wall time: 0 days, 0 hours, 0 minutes, and 0.166861 seconds
total wall time: 0 days, 0 hours, 0 minutes, and 1.68967 seconds

current simulated time: 20 min (max: 2880 min)
total agents: 513
interval wall time: 0 days, 0 hours, 0 minutes, and 0.173556 seconds
total wall time: 0 days, 0 hours, 0 minutes, and 1.8642 seconds

current simulated time: 22 min (max: 2880 min)
total agents: 513
interval wall time: 0 days, 0 hours, 0 minutes, and 0.178207 seconds
total wall time: 0 days, 0 hours, 0 minutes, and 2.0432 seconds

current simulated time: 24 min (max: 2880 min)
total agents: 513
interval wall time: 0 days, 0 hours, 0 minutes, and 0.190704 seconds
total wall time: 0 days, 0 hours, 0 minutes, and 2.23486 seconds

```

PhysiCell Project Essentials (3)

- Look at saved data

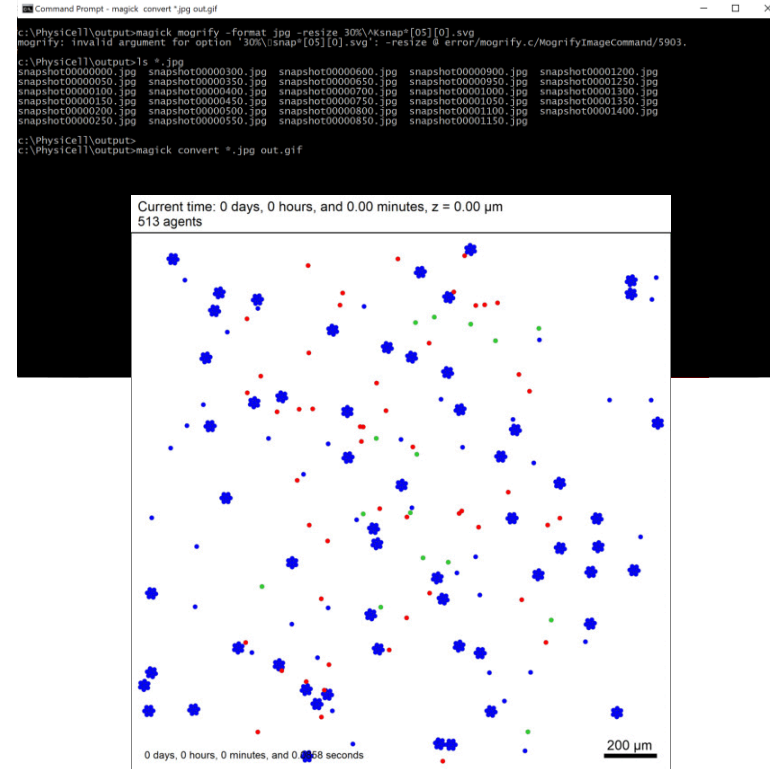
- Most projects save data to ./output
 - ♦ XML files give metadata, mesh, and substrate info
 - ♦ MAT file save (compressed) substrate and cell data
 - ♦ SVG files are visual quick snapshots
 - ♦ More on loading XML / MAT files in Python later

- Let's convert SVG to rescaled JPEG

- `magick mogrify -format jpg -resize 30% snap*.svg`
 - ♦ Convert `snapshot00000000.svg`, `snapshot00000001.svg`, ...
- `magick mogrify -format jpg -resize 30% snap*[05][0].svg`
 - ♦ Convert `snapshot00000000.svg`, `snapshot00000050.svg`, ...

- Now, let's create an animated GIF

- `magick convert *.jpg out.gif`



Working with the images

- To convert all the SVG files to PNG format

```
magick mogrify -format png snap*.svg
```

- To convert every SVG file ending in 0 or 5 to JPG format

```
magick mogrify -format jpg snap*[05].svg
```

- To convert the JPG files to an animated GIF

```
magick convert *.jpg out.gif
```

- To create an mp4 movie:

```
ffmpeg -r 24 -f image2 -i snapshot%08d.jpg -vcodec libx264 -pix_fmt yuv420p -strict -2 -tune animation -crf 15 -acodec aac out.mp4
```

Handy tricks!

Use `make jpeg` to create a full set of JPGs

Use `make movie` easily create the mp4.

PhysiCell Project Essentials (4)

- **Data cleanup**

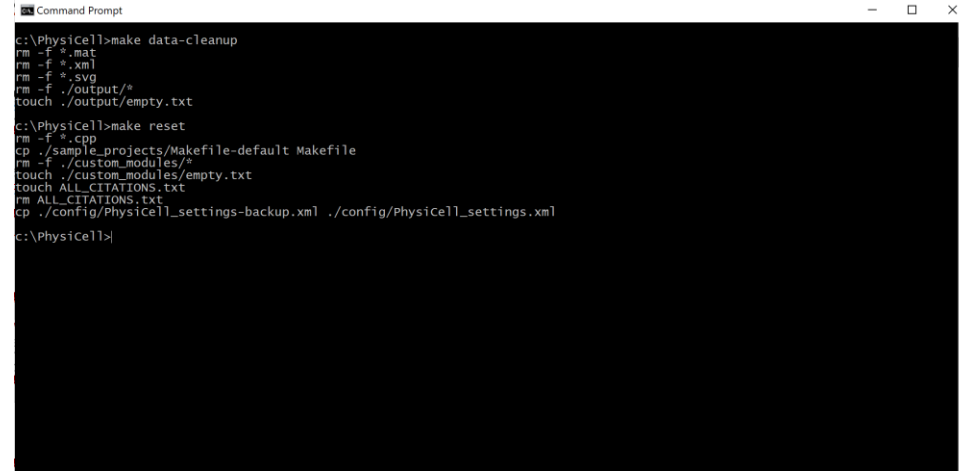
- Clean up data to get ready for another run

- **make data-cleanup**

- **Reset** to a clean slate

- De-populate the project
- Get ready for another project

- **make reset**



```
c:\PhysiCell>make data-cleanup
rm -f *.mat
rm -f *.xml
rm -f *.svg
rm -f ./output/*
touch ./output/empty.txt

c:\PhysiCell>make reset
rm -f *.cpp
cp ./sample_projects/Makefile-default Makefile
rm -f ./custom_modules/*
touch ./custom_modules/empty.txt
touch ALL_CITATIONS.txt
rm ALL_CITATIONS.txt
cp ./config/PhysiCell_settings-backup.xml ./config/PhysiCell_settings.xml

c:\PhysiCell>
```

Changing settings in a project



LUDDY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

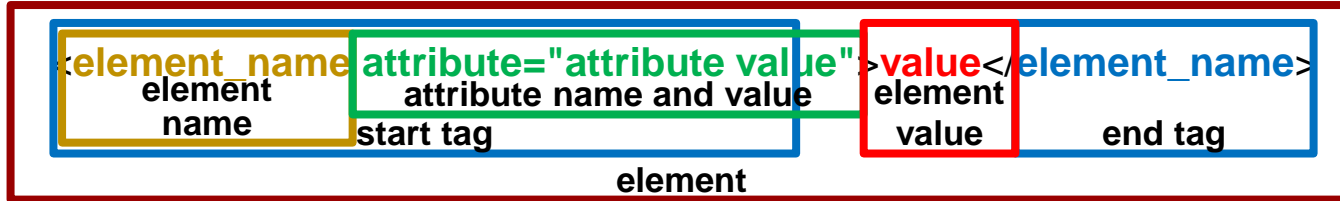
PhysiCell Project

PhysiCell.org

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

XML Refresher (1)

- XML stands for **eXtensible Markup Language**
 - (Think of it as a generalization of HTML.)
- Information in XML are stored in elements. Key elements are:
 - element name in a start tag
 - attributes and values
 - element value
- If an element has a value, it must have a matching end tag:



- If an element has attributes but no value, you can use a more compact form:

```
<element_name attribute1="attribute 1 value" attribute2="attribute 2 value" />
```

XML Refresher (2)

- Just like HTML, XML can have sub-elements:

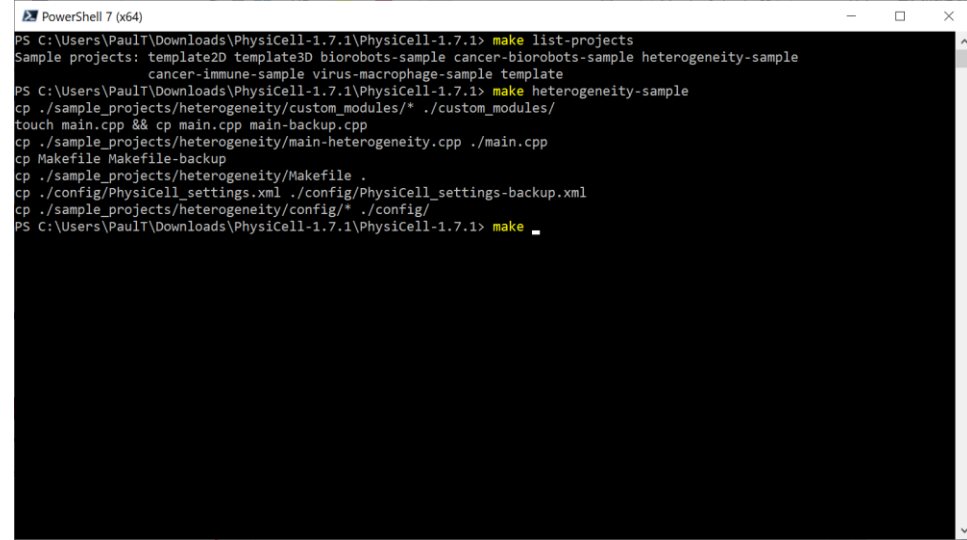
```
<element_name attribute1="attribute 1 value" attribute2="attribute 2 value">
  <subelement_name attribute="attribute value">subvalue1</subelement_name >
  <subelement_name attribute="attribute value">subvalue2</subelement_name >
</element_name >
```

- By convention:
 - the name of the element is a parameter name
 - the element's value is the parameter value
 - attributes are used to store metadata or other clarifications (e.g., units)

```
<diffusion_coefficient units="micron/min^2">1000</diffusion_coefficient>
```

First, populate the cancer heterogeneity project

- List all available sample projects
- Populate the cancer heterogeneity project
- Build the project
- Change some settings (next slide)



```
PowerShell 7 (x64)
PS C:\Users\PaulT\Downloads\PhysiCell-1.7.1\PhysiCell-1.7.1> make list-projects
Sample projects: template2D template3D biorobots-sample cancer-biorobots-sample heterogeneity-sample
                  cancer-immune-sample virus-macrophage-sample template
PS C:\Users\PaulT\Downloads\PhysiCell-1.7.1\PhysiCell-1.7.1> make heterogeneity-sample
cp ./sample_projects/heterogeneity/custom_modules/* ./custom_modules/
touch main.cpp && cp main.cpp main-backup.cpp
cp ./sample_projects/heterogeneity/main-heterogeneity.cpp ./main.cpp
cp Makefile Makefile-backup
cp ./sample_projects/heterogeneity/Makefile .
cp ./config/PhysiCell_settings.xml ./config/PhysiCell_settings-backup.xml
cp ./sample_projects/heterogeneity/config/* ./config/
PS C:\Users\PaulT\Downloads\PhysiCell-1.7.1\PhysiCell-1.7.1> make
```

How to change settings in XML

- Open config/PhysiCell_settings.xml
- Major sections:
 - **domain** -- how big of a region to simulate
 - **overall** -- how long to simulate, time step sizes
 - **parallel** -- OpenMP settings
 - **save** -- how often to save SVG images and full data
 - **microenvironment** -- settings on diffusing substrates
 - **user_parameters** -- model-specific settings
 - **cell_definitions** -- set baseline cell properties

Exercise: change settings and run

- Let's set the maximum simulation time to 2160 minutes
- Let's set the domain to $[-500,500] \times [-500,500]$ to speed it up
- Let's set the oncoprotein standard deviation to 3
- Let's set the max oncoprotein to 10 (mean + 3 standard deviations)
- Compile and run as before.

Let's set options and run (1)

- Open `./config/PhysiCell-settings.xml`
- Let's set the domain size in the **domain** block
 - Switch to `[-500,500] x [-500,500] x [-10,10]` to speed it up

```
<PhysiCell_settings version="devel-version">
  <domain>
    <x_min>-500</x_min>
    <x_max>500</x_max>
    <y_min>-500</y_min>
    <y_max>500</y_max>
    <z_min>-10</z_min>
    <z_max>10</z_max>
    <dx>20</dx>
    <dy>20</dy>
    <dz>20</dz>
    <use_2D>true</use_2D>
  </domain>
```


Let's set options and run (2)

- Let's also look at the **user_parameters** block
 - Let's change the oncoprotein standard deviation (**oncoprotein_sd**) to 3 (more variation)
 - Let's change the max oncoprotein (**oncoprotein_max**) to $\text{mean} + 3 \text{ sds} = 1 + 9 = 10$

```
<user_parameters>
  <tumor_radius type="double" units="micron">250.0</tumor_radius>
  <oncoprotein_mean type="double" units="dimensionless">
    1.0</oncoprotein_mean>
  <oncoprotein_sd type="double" units="dimensionless">3.0</oncoprotein_sd>
  <oncoprotein_min type="double" units="dimensionless">0.0</oncoprotein_min>
  <oncoprotein_max type="double" units="dimensionless">10</oncoprotein_max>
  <random_seed type="int" units="dimensionless">0</random_seed>
</user_parameters>
```

Let's set options and run (3)

- Let's look at the **overall** block
 - Set max time to 1.5 days = $1.5 \times 24 \times 60 = 2160$ minutes

```
<overall>
  <max_time units="min">2160</max_time> <!-- 36 h * 60 min -->
  <time_units>min</time_units>
  <space_units>micron</space_units>
```

- Let's look at the **save** block
 - Set the full save interval to 6 hours = 360 minutes

```
<save>
  <folder>output</folder> <!-- use . for root -->
  <full_data>
    <interval units="min">360</interval>
    <enable>true</enable>
  </full_data>
```

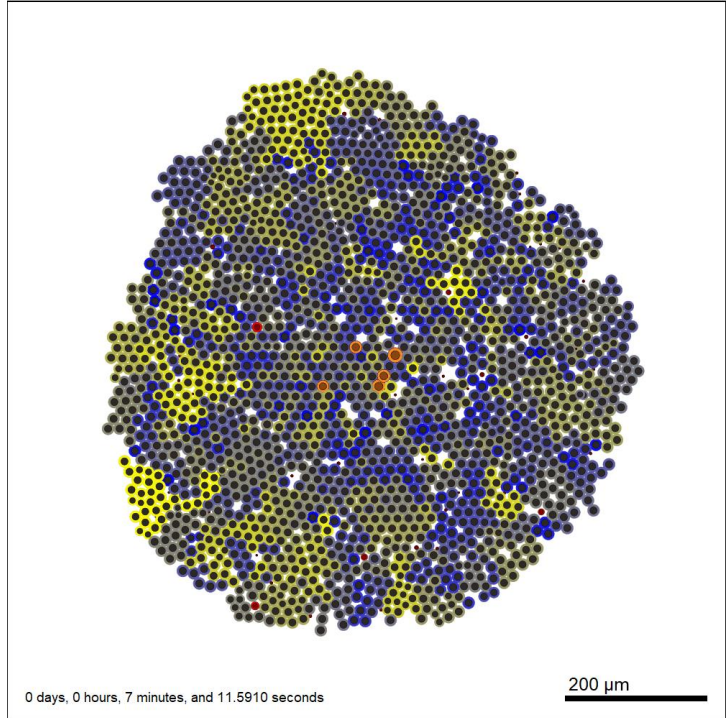
- Now, run! (`./heterogeneity`)

Let's do a quick visualization

- make gif

- We can see that the yellow cells eventually "win": they grow faster and form microcolonies within the tumor
- The effect is greatest on the outside edge: They have access to more O_2 here

Current time: 5 days, 0 hours, and 0.01 minutes, z = 0.00 μm
1996 agents



Funding Acknowledgements



NATIONAL
CANCER
INSTITUTE



leidos



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)