Slides, videos, links and more:
https://github.com/physicell-training/ws2023

Session 1: Working with PhysiCell Projects

Paul Macklin
@MathCancer

PhysiCell Project

July 31, 2023

Session 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

- Learn how to work with user projects
  - Save a user project
  - Get a list of user projects
  - Load (populate) and recompile a user project
  - Pack a user project (for sharing)

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)
  - 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

PhysiCell Project Essentials (1)

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

- The 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

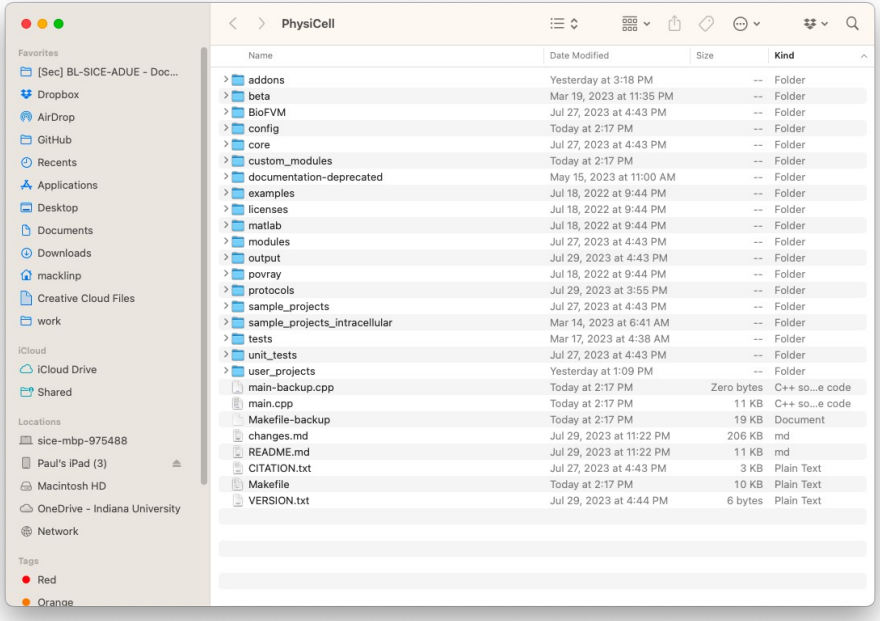
PhysiCell — -zsh — 102×39

```
(base) macklinp@sice-mbp-975488 PhysiCell % make list-projects
Sample projects: template biorobots-sample cancer-biorobots-sample cancer-immune-sample
                 celltypes3-sample heterogeneity-sample pred-prey-farmer virus-macrophage-sample
                 worm-sample interaction-sample mechano-sample rules-sample physimess-sample

Sample intracellular projects: ode-energy-sample physiboss-cell-lines-sample cancer-metabolism-sample

(base) macklinp@sice-mbp-975488 PhysiCell %
(base) macklinp@sice-mbp-975488 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.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/
(base) macklinp@sice-mbp-975488 PhysiCell %
(base) macklinp@sice-mbp-975488 PhysiCell %
```

Let's look at the project structure …

Project directory structure

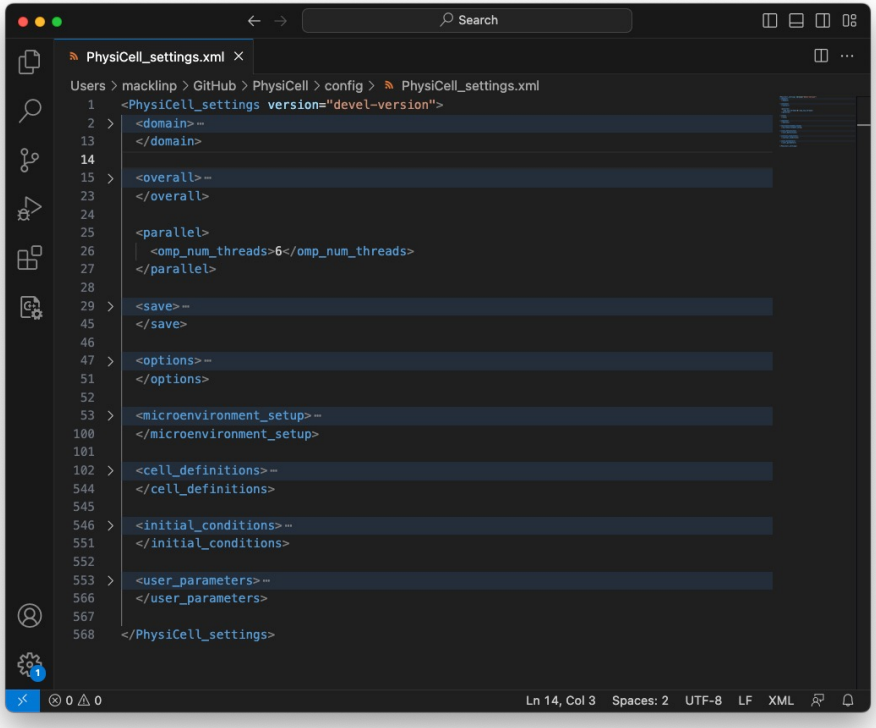
- (key) directories:
  - **./ (root):** main source, Makefile, and executable go here
  - **./addons:** officially supported addons, such as PhysiBoSS and libRoadrunner
  - **./beta**: for beta-testing (don't use)
  - **./BioFVM:** diffusion solver
  - **./config:** configuration files
  - **./core:** PhysiCell core functions
  - **./custom_modules:** put custom code for your project here.
  - **./documentation-deprecated:** old user guide, etc.
  - **./examples:** deprecated
  - **./licenses:** yep
  - **./matlab:** scripts and functions to load data in matlab
  - **./modules:** standard add-ons for PhysiCell
  - **./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
  - **./sample_projects_intracellular:** where we add intracellular sample projects
  - **./tests:** for automated testing (WIP)
  - **./unit_tests:** for automated testing (WIP)
  - **./user_projects:** where we save user-driven projects



Most of your work will be in the red directories

Project structure: main config file

- 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



Project structure: other config files

- As PhysiCell has evolved, it has migrated more setup to **./config**
  - **./config/cells.csv:**
    An optional comma-separated text file to specify initial cell positions
  - **./config/rules.csv:**
    An optional comma-separated text file to write cell rules (based on the cell behavior grammar)
  - **./config/model_n.bnd:**
    This type of file specifies Boolean network models for PhysiBoSS / MaBoSS
  - **Other:** libSBML, PhysiMeSS, dFBA can place additional config files here.

Project structure: custom modules

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

Command Prompt

```
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

```
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

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

```
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

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
    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++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

PhysiCell_settings.xml | custom.cpp | custom.h | main.cpp

```
139
140     // for simplicity, set a pathology coloring function
141
142     std::vector<std::string> (*cell_coloring_function)(Cell*) = my_coloring_function;
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
159         report_file.open(filename);    // create the data log file
160         report_file<<"simulated time\tnum cells\tnum division\tnum death\twall time"<<std::endl;
161     }
162
163     // main loop
164
165     try
166     {
167         while( PhysiCell_globals.current_time < PhysiCell_settings.max_time + 0.1*diffusion_dt )
168         {
169             // save data if it's time.
170             if( fabs( PhysiCell_globals.current_time - PhysiCell_globals.next_full_save_time ) < 0.01 * diffusion_dt )
171             {
172                 display_simulation_status( std::cout );
173                 if( PhysiCell_settings.enable_legacy_saves == true )
174                 {
175                     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

Type here to search | 1:00 AM 1/9/2019

Project structure: main.cpp (continued)

- **`main.cpp`**
  - insert custom routines
  - **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

```
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 )
```

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

Type here to search | 1:10 AM

A last word on C++ source files

- Most modelers will use the cell behavior grammar to write their models.
  - Such models will not require modifying the source files you saw.

- However, it's wise to understand the structure of PhysiCell:
  - Know which files are yours (specific to your project) and which belong to PhysiCell
  - Know where to make edits if you need to.

- PhysiCell ecosystem developers may need this technical knowledge.
  - Future workshops may split: one for users, and one for ecosystem contributors.
  - This year, developer-focused optional sessions are marked **"advanced."**

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

PhysiCell Project Essentials (2)

- Now, **compile the project**
  - `make`

- Then, **run the project**
  - `./biorobots` (Linux, MacOS)
  - `biorobots.exe` (Windows)

- This should take about 5 minutes

Command Prompt

```
c:\PhysiCell>make
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./BioFVM/BioFVM_vector.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./BioFVM/BioFVM_mesh.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./BioFVM/BioFVM_microenvironment.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./BioFVM/BioFVM_solvers.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./BioFVM/BioFVM_matlab.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./BioFVM/BioFVM_utilities.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./BioFVM/BioFVM_basic_agent.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./BioFVM/BioFVM_MultiCellDS.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./BioFVM/BioFVM_agent_container.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./BioFVM/pugixml.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./core/PhysiCell_phenotype.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./core/PhysiCell_cell_container.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./core/PhysiCell_standard_models.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./core/PhysiCell_cell.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./core/PhysiCell_custom.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./core/PhysiCell_utilities.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./core/PhysiCell_constants.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./modules/PhysiCell_SVG.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./modules/PhysiCell_pathology.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./modules/PhysiCell_MultiCellDS.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./modules/PhysiCell_various_outputs.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./modules/PhysiCell_pugixml.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./modules/PhysiCell_settings.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -c ./custom_modules/biorobots.cpp
g++ -march=native  -O3 -fomit-frame-pointer -mfpmath=both -fopenmp -m64 -std=c++11  -o biorobots BioFVM_vector.o BioFVM_mesh.o
BioFVM_microenvironment.o BioFVM_solvers.o BioFVM_matlab.o BioFVM_utilities.o BioFVM_basic_agent.o BioFVM_MultiCellDS.o BioFVM_
agent_container.o  pugixml.o PhysiCell_phenotype.o PhysiCell_cell_container.o PhysiCell_standard_models.o PhysiCell_cell.o Phy
siCell_custom.o PhysiCell_utilities.o PhysiCell_constants.o PhysiCell_SVG.o PhysiCell_pathology.o PhysiCell_MultiCellDS.o Physi
Cell_various_outputs.o PhysiCell_pugixml.o PhysiCell_settings.o biorobots.o main.cpp

c:\PhysiCell>
```

Command Prompt - biorobots

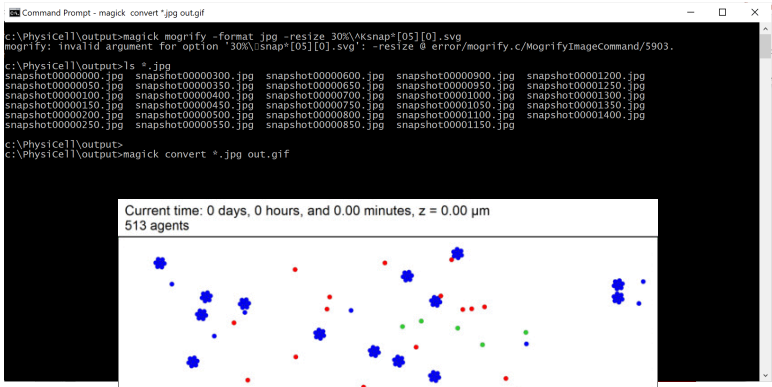
```
current simulated time: 12 min (max: 2880 min)
total agents: 513
interval wall time: 0 days, 0 hours, 0 minutes, and 0.189681 seconds
total wall time: 0 days, 0 hours, 0 minutes, and 1.17758 seconds

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
```

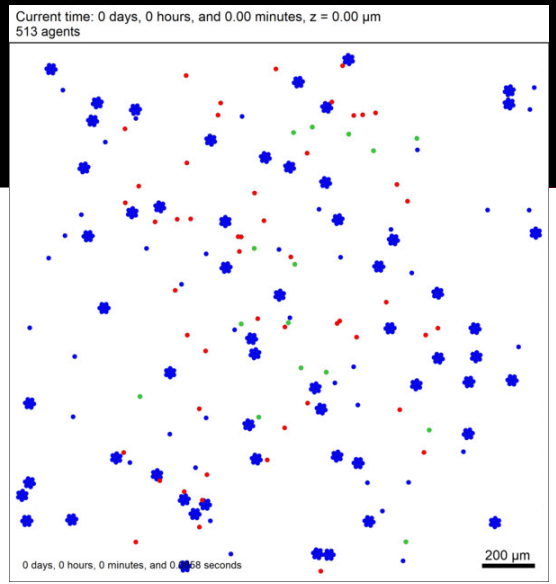
PhysiCell Project Essentials (3)

- **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!

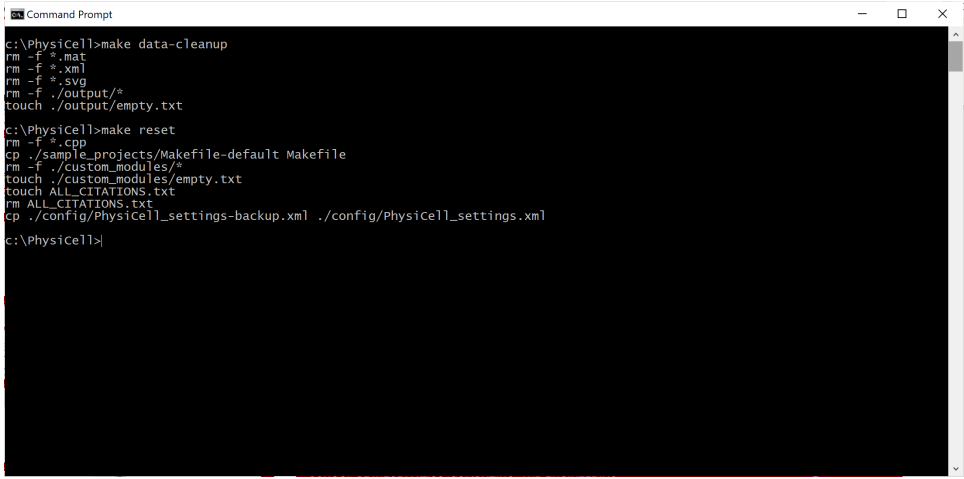
Use make jpeg to create a full set of JPGs

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`



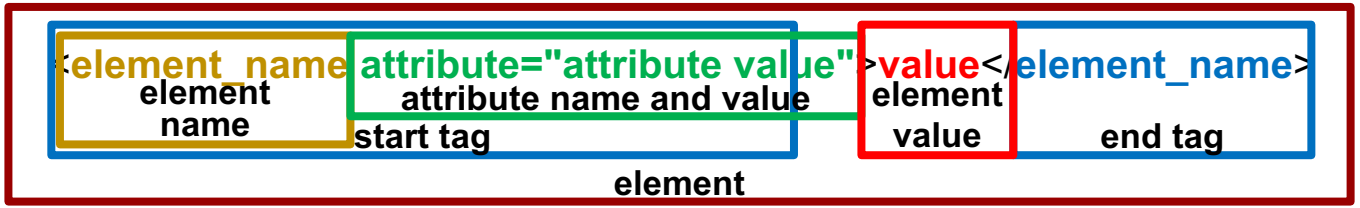
Changing settings in a project

XML Refresher (1)

- XML stands for e**X**tensible **M**arkup **L**anguage
  - (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] x [-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 mean + 3 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 x 24 x 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>
```

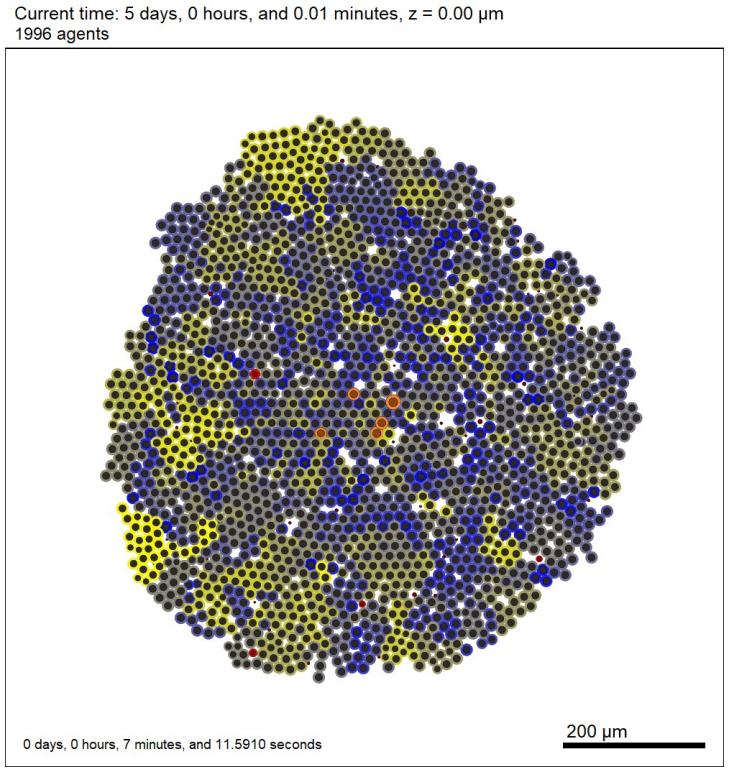
- Now, run! (`./heterogeneity`)

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



A last word on configuration files

- PhysiCell is migrating its model specification into markup and configuration files.

- Most of these will be edited graphically, as in the next session.

- However, it's wise to understand the structure of PhysiCell:
  - Know which files are yours (specific to your project) and which belong to PhysiCell
  - Know where to make edits if you need to.

- PhysiCell ecosystem developers may need this technical knowledge.
  - Future workshops may split: one for users, and one for ecosystem contributors.
  - This year, developer-focused optional sessions are marked **"advanced."**

User Projects

- We now support saving and loading user projects.
- **make save PROJ= [project-name]:** save a user project
  - saves user project files in **./user_projects/[project-name]**
  - files currently saved:
    ./config/* **main.cpp** **Makefile** **./custom_modules/***
- **make load PROJ=[project-name]:** load a user project
  - loads user project files from **./user_projects/[project-name]**
  - Use **make** to compile and run the project
- **make list-user-projects:** list all prior user projects

Packing and Sharing User Projects

- You may want ot share your PhysiCell project with another scientist.
  This is not yet developed, but here's a method:
  1. zip the contents of your project (stored in **./user_projects/[my-project]**)
     ```
     cd ./user_projects && zip -r $[my-project].zip [my-project]
     ```
  2. Share / email the zip file with another user
  3. Instruct them to unzip it in their **./user_projects** folder

- **Note:** PhysiCell 1.13.1 will probably have a **make pack PROJ=[my-project]** rule

- Future development direction:
  - Create a true XML format for PhysiCell projects
    - Provenance
    - Versioning (e.g., version of PhysiCell used)
    - Desired executable name
    - Project instructions / documentation …

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)