Slides, videos, links and more:
https://github.com/physicell-training/ws2021

Session 6: Custom Variables and Accessing parameters using C++

Aneequa Sundus
@AneequaSundus

PhysiCell Project

July 26, 2021

Agenda

- Working with C++ code
  - Defining Custom variables
  - Querying cell definitions
  - Querying for microenvironment
  - Boundary conditions
  - Cell secretion, uptake, export

Need and Files to edit

- Always add something later
- Custom Functions
- File to edit
  - custom.cpp
  - custom.h
  - PhysiCell_settings.xml

Assumptions

I assume that you have a basic Model built using quick builder and have following substrates and cell types.

Biofilm Model

- Substrates
  - Oxygen
  - Glucose
  - ECM
- Cell Types
  - Wound
  - Aerobic bacteria
  - Anaerobic bacteria
- Custom Variables
  - Energy
  - alpha

Custom Variables

- Two ways to add custom variables to Model
  - Quick builder
  - Add in C++ file

Defining and Initializing Custom Variables

- Where?

custom.cpp

- How?

- cell_defaults.custom_data.add_variable("energy", "dimensionless" , 0.5);

- cell_defaults.custom_data.add_variable("energy", "dimensionless" , parameters.doubles("cell_default_inital_energy"));

- cell_defaults.custom_data.add_variable("alpha", "none" , parameters.doubles("cell_default_aplha"));

Accessing custom Vriables

- Where ?
  - custom.cpp
- How?
  - static int nE = pCell->custom_data.find_variable_index("energy");
  - pCell->custom_data[nE]

Boundary conditions microenvironment

Sampling the microenvironment (1)

- There is a global microenvironment called **microenvironment.** You can access it anywhere from inside a PhysiCell model.

- Each cell is in some computational voxel in the microenvironment.
  - pCell->get_current_voxel_index(void);
    - Get the index of the voxel

- You can query the microenvironment to determine which index corresponds to a variable.
  - microenvironment.find_density_index("resource"); // Find the index of "resource".
    - This function returns -1 if it can't find your substrate.

- Each cell can access the vector of chemical substrates in its voxel
  - pCell->nearest_density_vector();
    - Vector of all the substrates
  - pCell->nearest_density_vector()[2]
    - substrate with index 2 in the cell's voxel
    - often, you'll want to use the search above to figure out which index

Sampling the microenvironment (2)

- Each cell can access the gradients of the substrates in its voxel
  - `pCell->nearest_gradient(2);`
    - ♦ gradient of substrate #2

- We can access the mesh
  - `microenvironment.mesh`
  - `microenvironment.mesh.voxels`

- We can iterate through all voxels

```
for( int i=0; i < microenvironment.mesh.voxels.size() ; i++ )
{ std::cout << microenvironment.mesh.voxels[i].center << std::endl; }
```

Dirichlet's nodes

- void Microenvironment::add dirichlet node(int voxel index, std::vector<double>& value)
- `microenvironment.add_dirichlet_node( n,bc_vector_air  );`

- `Where n is the voxel number and bc_vector_air is double vector of size 3 since we have 3 substrates in the environment.`

Cell secretion, uptake, export

Secretion

- Get Substrate index

```
int oxygen_substrate_index = microenvironment.find_density_index( "oxygen" );
```

Set Uptake Rate

```
cell_defaults.phenotype.secretion.uptake_rates[oxygen_substrate_index] =
parameters.doubles("cell_default_oxygen_uptake_rate");
```

Set Secretion Rate

```
cell_defaults.phenotype.secretion.secretion_rates[oxygen_substrate_index] =
parameters.doubles("cell_default_oxygen_secretion_rate");
```

Set Saturation Density

```
cell_defaults.phenotype.secretion.saturation_densities[oxygen_substrate_index] =
parameters.doubles("cell_default_oxygen_saturation_density");
```

Accessing internalized substrates

- By default, PhysiCell keeps track of the net amount of internalized substrates. (They have dimensions of "total stuff", not stuff/volume.)
- Each environmental substrate has a corresponding internalized substrate with the same index.
- Access via:

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

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

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

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

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

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

Best Practices

- In any customized cell function, you can access the microenvironment at its location.

- For best practices, you *don't* want to hard-code the index substrate. If somebody adds a substrate to the XML or reorders them, it could break your code.

- Instead, search for the index of the substrate and store the result in a static variable.
- Code Available:https://github.com/furkankurtoglu/Biofilm-Project

Funding Acknowledgements

PhysiCell Development:

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

Training Materials:

- Administrative supplement to NCI U01CA232137 (Year 2)