PhysiCell Scene Explorer v1.0 User Guide

Dylan Taylor

January 2020

Contents

1 Introduction

1.1 Purpose

The purpose of this application is to provide a simple way to view PhysiCell output files for users of all levels of familiarity with PhysiCell.
The purpose of this guide is to explain the key components of the application and how to use them.

1.2 What You Need

All you need is the PhysiCell Scene Explorer along with some PhysiCell output files (I have provided a couple in the examples folder)

1.3 Installation

You can download the files here.

2 User Actions

2.1 Key Bindings

Key	Action
w, a, s, d, Space, Left Shift, Left Control	movement
up, down, left, right, mouse	camera look
z + up, z + down, scroll wheel	camera zoom
p	disable mouse
r	switch cell data file
t	switch mod file
g	add modification zones
b	save modification zones
y	reset cells
u	apply modification zones
h	toggle coordinate display
k	take a screenshot
m	toggle modification zone visibility
Escape	close application

2.2 Actions

movement: WASD provides directional control, space ascends, left control descends, and holding shift increases movement speed.

switch cell data file: Cell data is stored in the "outputXXXXXXXX_cells_PhysiCell.mat" file. The user may select the file directly or the corresponding "outputXXXXXXXX.xml" file. If the XML file is selected, the variables defined in the XML file will be given to each cell. If the mat file is selected directly, then the default variables[1] will be used.

switch mod file: Overwrite the current modification zones with the ones serialized in the selected mod file. The mod file contains the modification zone data. More on this in section 4,

add modification zones: Similar to switching mod files, but instead of overwriting the current modification zones, it adds the ones from the selected mod file to the scene without getting rid of the old ones.

save modification zones: Save the current modification zones to a mod file to be reloaded later.

reset cells: reset all cells to their initial position and undo all modification zone transformations.

apply modification zones: Reapply modification zones to scene.

toggle coordinate display: Display the user's current (x, y, z) position. Meant to assist the user in writing modification zones to mod files.

toggle modification zone visibility: toggle the mesh renderers of the current modification zones.

[1]Default variables: ID, position (vector), total_volume, cell_type, cycle_model, current_phase, elapsed_time_in_phase, nuclear_volume, cytopasmic_volume, fluid_fraction, calcified_fraction, orientation (vector), polarity, migration_speed, motility_vector (vector), migration_bias, motility_bias_direction (vector), persistence_time, motility_reserved, oncoprotein, elastic_coefficient, kill_rate, attachment_lifetime, attachment_rate. All of the variables are floats or float vectors.

3 Modification Zones

3.1 What are Modification Zones?

Modification zones are the means by which we conduct transformations to our scene.
In the folder "Assets/Resources/Modification Zones", you will find the Modification Zone Prefab.

The significant components of a modification zone are:

- Transform
  - The modification zone's position, size, and rotation
- Mesh Renderer
  - Visual representation of the modification zone
  - Toggled on and off by pressing 'm'
- Box Collider
  - Used to detect if cells are within the modification zone
- Script (Modifier)
  - Describes the transformation to be applied to cells within the modification zone
  - Must implement the Modifier interface (more on this on the next page)

3.2 Available Modification Zones

The following modification zones are available as of v1.0:

- Colorer: changes a cells color
- ColorMap: color a cell based on its cell type
- Displacer: translate the cell's position
- Excluder: toggle the visibility of a cell
- Stratifier: split cells into a certain number of evenly spaced subsections along a provided direction

3.3 How to Use Modification Zones

If you are adding modification zones in the Unity editor, you must put them in the 'Controller' object's Controller.modificationZones array for them to be applied to cells in the scene.

If you are adding modification zones in the application, you must make a mod file. You can find an example of this in section 4.

3.4 The Modifier Interface

Each modification zone has a script which implements the modifier interface. The purpose of the script is to specify how a cell within a modification zone's bounds is transformed, how the modification zone is saved to a file, and how it is loaded.

The modifier interface has four functions:

- void init()
  - Initialize any non-public data needed for the modification zone
  - Often does nothing
- void modify(GameObject g)
  - Specify how to transform a given cell g
    - g is implicitly within the bounds of the modification zone
- string writeToString()
  - Return a string representation of the modification zone
  - The first line is always of the form:
    - "[Modification Zone Name] [n]"
    - where n is the number of lines after this line that belong to this particular modification zone in the mod file
  - The second line is always of the form:
    - "[x-coordinate] [y-coordinate] [z-coordinate]"
  - The third line is always of the form:
    - "[x-scale] [y-scale] [z-scale]"
  - All other lines depend on the particular modification zone
- void loadFromString(string serialization)
  - Initialize any public data from the string serialization
    - string serialization is of the form specified by writeToString()

4 Mod Files

4.1 What are Mod Files?

Mod files are the means by which we save, load, and edit modification zones for use within the application. The form of a mod file is:

```
n
[modification zone 1 writeToString()]
[modification zone 2 writeToString()]
...
[modification zone n writeToString()]
```

Where n is the number of modification zones in the file.

4.2 Modification Zone Serializations

Each modification zone has its own serialization speficied by its writeToString method. Examples of each of these serialization can be found in the "Examples/Mod Files" directory. Follow the example in section 4.3 to add a new modification zone to your mod file.

4.2.1 Colorer

```
Colorer 3
[x-coordinate] [y-coordinate] [z-coordinate]
[x-scale] [y-scale] [z-scale]
[R] [G] [B]
```

4.2.2 ColorMap

```
ColorMap [2 + number of rules]
[x-coordinate] [y-coordinate] [z-coordinate]
[x-scale] [y-scale] [z-scale]
[type_1] [color_1]
[type_2] [color_2]
...
[type_(number of rules)] [color_(number of rules)]
```

4.2.3 Displacer

```
Displacer 3
[x-coordinate] [y-coordinate] [z-coordinate]
[x-scale] [y-scale] [z-scale]
[offset-x] [offset-y] [offset-y]
```

4.2.4 Excluder

```
Excluder 2
[x-coordinate] [y-coordinate] [z-coordinate]
[x-scale] [y-scale] [z-scale]
```

4.2.5 Stratifier

```
Stratifier 5
[x-coordinate] [y-coordinate] [z-coordinate]
[x-scale] [y-scale] [z-scale]
[reference-x] [reference-y] [reference-z]
[direction-x] [direction-y] [direction-z]
[numSubsections] [spacing]
```

4.3 Example

The following is an example of how you can edit mod files to change how your scene is being transformed We start with a mod file which contains an excluder and stratifier modification zone:

```
2
Excluder 2
485 -341 -359
940.0 702.0 841.0
Stratifier 5
-113 0 0
1892.0 1747.0 1730.0
-500 0 0
1 0 0
20 100
```

Say we wish to change how our modification zones transform our scene. We can accomplish this by changing the relevant data in the mod file. For example, what if we want the stratifier to split our scene into even more subsections. All we have to do is change the 20 in the stratifier's serialization to a larger number, say 50. Doing this gives us:

```
2
Excluder 2
485 -341 -359
940.0 702.0 841.0
Stratifier 5
-113 0 0
1892.0 1747.0 1730.0
-500 0 0
1 0 0
50 100
```

Now say we wish to add an additional stratifer so that our scene is being stratifiied in two different directions. All we have to do is increment the number at the start of the file to a 3 and add the writeToString() for our new stratifer. Doing this gives us:

```
3
Excluder 2
485 -341 -359
940.4 702.85 841.0615
Stratifier 5
-113 0 0
1892.926 1747.376 1730.495
-500 0 0
1 0 0
50 100
Stratifier 5
-113 0 0
1892.926 1747.376 1730.495
0 -500 0
0 1 0
50 100
```

5 Additional Reading

5.1 Unity

Unity Technologies. (2019). Unity. https://unity.com/

5.2 Runtime File Browser

https://github.com/yasirkula/UnitySimpleFileBrowser

5.3 Fly Camera

https://forum.unity.com/threads/fly-cam-simple-cam-script.67042/

5.4 .mat Parser

adapted from: https://github.com/MathCancer/PhysiCell/blob/master/BioFVM/BioFVM_matlab.cpp

A. Ghaffarizadeh, S.H. Friedman, and P. Macklin, BioFVM: an efficient parallelized diffusive transport solver for 3-D biological simulations, Bioinformatics 32(8): 1256-8, 2016. DOI: 10.1093/bioinformatics/btv730