

# CompuCell 3D

# Quickstart Guide

**Version 0.5**

**Authors: Maciej Swat, Ariel Balter, Nikodem Poplawski, James A. Glazier**

**Please send corrections and Revisions to: [mawat@indiana.edu](mailto:mawat@indiana.edu)**

**Extra credit may be offered**

**Last Updated: Tuesday, March 7, 2006**

## Table of Contents

## 1. Developing a CompuCell3D Simulation—Overview

We hope that CompuCell is a user-friendly tool. Once you have CompuCell installed on your computer, you will be able to design, run and post-process your own biological simulations. In this guide we briefly outline the steps involved in developing a simulation. Currently, all CompuCell simulations are based on an .xml configuration file which the CompuCell Player reads and executes. This configuration file contains the basic parameters that define the cells and their interactions and directs CompuCell to other user-defined text files containing initial configurations of cells and chemical fields.

The first step in building a CompuCell simulation is to design the cells and their interactions. Properties that you can assign to cells include a target volume or surface area, a response to chemical gradients, secretion of chemicals, differential cell adhesion, *etc.*.... Since the Monte Carlo employed in the Glazier-Graner Cellular Potts Model (*CPM*) method is based on fluctuations, you must also specify the overall amplitude of fluctuations and the degree of deviation allowed from target values, *e.g.* how much the volume will fluctuate around the target volume.

CompuCell includes some standard initializers that create initial configurations of cells, but you will probably want to create your own. You specify the initial cell configuration in a data file with a simple format. You can create this file with a scripting language or just type it in using a word processor, working from a diagram on some graph paper. In the future, CompuCell will have graphical tools to help with this process.

Once you have written your configuration file, running CompuCell is as simple as playing a movie. You open the player, select the configuration file you want to use (using the command [Open](#) from the [File](#) drop-down menu) and click the [Play](#) button. You can select visualization options and customize the output you wish to store for later analysis. In addition to visualizing (and saving) the cell fields, you can look at chemical fields, pressure fields, velocity fields, *etc.*....

Finally, you can analyze your data with standard techniques for analyzing and visualizing spatial data, *e.g.* in MatLab or Mathematica. We are currently developing some basic analysis tools to add to the CompuCell player.

## 2. Writing Your Own Configuration Files

### 2.1 XML Structure and Syntax

*XML* (or *.xml*), which stands for "eXtensible Markup Language" is an extension (or superclass) of *HTML* (Hypertext Markup Language). If you are familiar with HTML then you should find using XML easy. If not, it may take some getting used to. We use XML to specify configuration files because it is easy for a computer to parse. In the .xml configuration file you will specify the general parameters of your simulation such as the types of cells, their properties, *etc.*....

A typical block of .xml you will use in CompuCell3D looks like this:

```
<SectionName Attribute1="attribute" Attribute2="attribute">
  <Variable1 VarAttribute1="attribute" VarAttribute2="attribute">
    ValueOfVariable
  </Variable1>
  <Subsection>
    <VarA>ValueA</VarA>
  </Subsection>
</SectionName>
```

#### Example 1.

If that looks like a lot of gobble-dee-gook, don't worry. I'll explain it. The first thing to notice is that in XML you

specify where each **statement** or **element** begins and ends like this:

```
<begin>
...
</begin>
```

`<begin>` is referred to as a **begin tag** and `</begin>` is called an **end tag**. Every xml element must have a begin tag and an end tag.

We can also use the following syntax: `<begin .... />`. In this case, `/>` marks the end of the element. This is an example of shortcut notation often becomes handy.

As a matter of style, defining characteristics, properties, or **attributes** of a **section** are specified using the syntax `PropertyValue="attribute"`, while specific **values** for a variable is usually placed between a `<begin>` and an `</begin>` for that variable. By section we mean everything that is contained between beginning tag and ending tag. For example:

```
<Sentence>
  <Text>This is nice example</Text>
  <Font>TimesNewRoman</Font>
</Sentence>
```

Above we can see a section called `Sentence` which consists of two elements `Text` and `Font`. An set of examples, each of which defines a variable and assigns it a style and value is:

```
<Age Style="numeric">38</Age>
or
<Age Style="InWords">"Thirty-Eight"</Age>
or
<Age Style="Roman">"XXXVIII"</Age>
```

I say it's a matter of style, because we could just use the syntax:

```
<Age>
  <Style>"Roman"</Style>
  <Number>XXXVIII</Number>
</Age>
```

or:

```
<Age Style="Roman" Value="XXXVIII"/>
```

In the last example, properties of an element are attributes. In the second to last they are specified as values of two sub-elements `Style` and `Number`. Most parts of a configuration file will use a mixture of properties and attributes. The reference section of this manual lays out explicitly the proper syntax for these definitions.

XML is hierarchical and nested. Thus sections can contain **sub-sections**, and the properties defined within sub-sections apply only within those subsections. The section called `Subsection` in Example 1 is nested within the section `SectionName`. So `VarA` only takes the value `ValueA` within the subsection.

## 2.2 Configuration File Contents

Any configuration file will contain two main types of **blocks**, called **plugins** and **steppables**. A **block** is an .xml element with non-trivial content. Plugins and steppables have different functions. A **plugin** is a routine that defines the global properties of your simulation such as cell properties. A plugin is called at every spin-flip attempt. A **steppable** is a routine that operates on the CompuCell lattice after each **Monte Carlo Step (MCS)** (*i.e.* after as many spin-flip attempts as the number of lattice sites).

**Note:** You must list all plugins first, then steppables. If you reverse the order or mix plugin and steppable declarations in the configuration file you will generate cryptic runtime errors.

**Note:** The difference between a spin flip and a Monte Carlo Step (MCS) is crucial. A Monte Carlo Step includes many spin flip attempts. Typically the number of spin-flip attempts equals the number of lattice sites, though you can change this relationship in the configuration file. Confusion between MCS and spin-flip attempts is common among novice CPM users.

### 3. Your First CompuCell Simulation—Foam Growth

I will keep things as simple as possible and instead of starting with an introduction, motivation, overview *etc.*..., I will show you complete configuration file for a foam-coarsening simulation. I found that looking at an example is the best way to grasp how to write CompuCell configuration files. Writing these files is really not magic, so let's start:

Our first simulation will model the growth of soap bubbles in a foam. Bubbles are compact domains of gas separated by thin films of liquid stabilized by surfactants. The boundaries are subject to surface tension, and rearrange to try to minimize their total boundary length, causing them to assume the shapes of circular arcs. Triples of boundaries meet at vertices and the minimization causes the vertex angles to be  $120^\circ$ . In our CPM simulation, domains with the same index will represent the gas and the boundaries between domains (links with mismatched indices) will represent the soap films. In this case we have only one type of cell (the bubbles), though we will reserve a **Medium** cell type for the background. The curvature of the bubble walls causes pressure differences between bubbles and this in turn results in diffusion of gas from high pressure bubbles to low pressure bubbles. Somewhat counter-intuitively, walls move towards their concave side. Eventually, some bubbles will disappear and the average length-scale of the pattern will grow.

Here is a typical configuration file:

```
<CompuCell3D>
<Potts>
<Dimensions x="101" y="101" z="1"/>
<Anneal>0</Anneal>
<Steps>1000</Steps>
<Temperature>5</Temperature>
<Flip2DimRatio>1.0</Flip2DimRatio>
<Boundary_y>Periodic</Boundary_y>
<Boundary_x>Periodic</Boundary_x>
<FlipNeighborMaxDistance>1.75</FlipNeighborMaxDistance>
</Potts>

<Plugin Name="CellType">
<CellType TypeName="Medium" TypeId="0"/>
<CellType TypeName="Foam" TypeId="1"/>
</Plugin>

<Plugin Name="Contact">
<Energy Type1="Foam" Type2="Foam">50</Energy>
<Depth>2.0</Depth>
</Plugin>

<Steppable Type="PIFInitializer">
<PIFName>foaminit2D.pif</PIFName>
</Steppable>
```

</CompuCell3D>

It wasn't that bad. In fact, I am sure, that without any explanation you could figure out what every symbol means in this file. Nevertheless let's go through it step by step to make sure we understand syntax of every section.

### 3.1 CPM Definitions

The first section of the .xml file defines the global parameters of the lattice and the simulation.

```
<Potts>
<Dimensions x="101" y="101" z="1"/>
<Anneal>0</Anneal>
<Steps>1000</Steps>
<Temperature>5</Temperature>
<Flip2DimRatio>1</Flip2DimRatio>
<Boundary_y>Periodic</Boundary_y>
<Boundary_x>Periodic</Boundary_x>
<FlipNeighborMaxDistance>1.75</FlipNeighborMaxDistance>
</Potts>
```

This section appears at the beginning of the configuration file. Line `<Dimensions x="101" y="101" z="1"/>` declares the dimensions of the lattice to be 101 x 101 x 1, *i.e.*, the lattice is two-dimensional and extends in the xy plane. The basis of the lattice is 0 in each direction, so the 101 lattice sites in the x and y directions have indices ranging from 0 to 100. `<Steps>1000</Steps>` tells CompuCell how long the simulation lasts in MCS. After executing this number of steps, CompuCell can run simulation at zero temperature for an additional period. In our case it will run for `<Anneal>10</Anneal>` extra steps. Setting the temperature is as easy as writing `<Temperature>5</Temperature>`. Now, as you remember from the discussion about the difference between spin-flip attempts and MCS we can specify how many spin flips should be attempted in every MCS. We specify this number indirectly by specifying the **Flip2DimRatio** - `<Flip2DimRatio>1</Flip2DimRatio>`, which tells CompuCell that it should make 1 x number of lattice sites attempts per MCS – in our case one MCS is 101x101x1 spin-flip attempts. To set 2.5x101x101x1 spin flip attempts per MCS you would write `<Flip2DimRatio>2.5</Flip2DimRatio>`.

The next line specifies the neighbor range of interactions (nearest neighbor, next-nearest neighbor, *etc....*), **FlipNeighborMaxDistance**, `<FlipNeighborMaxDistance>1.75</FlipNeighborMaxDistance>`. This line tells CompuCell to search for a trial spin among those pixels whose distance from the current pixel (the one that undergoes the flip) is less than 1.75 lattice constants from the site to be substituted. The simplest simulation would set this distance to be 1 (nearest neighbor interaction), but as discussed in class nearest neighbor interactions may cause artifacts due to lattice anisotropy. The longer the interaction range, the more isotropic the simulation and the slower it runs. In addition, if the interaction range is comparable to the cell size, you may generate unexpected effects, since non-adjacent cells will contact each other.

The Potts section also contains tags called `<Boundary_y>` and `<Boundary_x>`. These tags impose boundary conditions on the lattice. In this case the x and y axes are **periodic** (`<Boundary_x>Periodic</Boundary_x>`) so that *e.g.* the pixel with x=0, y=1, z=1 will neighbor the pixel with x=100, y=1, z=1. If you do not specify boundary conditions CompuCell will assume them to be of type **no-flux**, *i.e.* lattice will not be extended. The conditions are independent in each direction, so you can specify any combination of boundary conditions you like.

Once we have used the Potts section to create our lattice we can start listing plugins.

### 3.2 Cell Type Plugin

Let's start with the **CellType** plugin whose main purpose is to inform CompuCell what cell types you will be using in the simulation.

**Note:** In CompuCell, every cell has a unique index or **Id** which differentiates it from other cells and a non-unique **cell type** which identifies its class of behavior. Many distinct cells may have the same type and will appear painted with the same color when you visualize them.

```
<Plugin Name="CellType">
<CellType TypeName="Medium" TypeId="0"/>
<CellType TypeName="Foam" TypeId="1"/>
</Plugin>
```

The syntax here is quite straightforward. Each line contains the name of a type that the simulation uses and assigns it to an integer valued **TypeId**. We strongly recommend that **TypeIds** are consecutive positive integers (e.g. 0,1,2,3...). Medium is traditionally assigned a TypeId=0 but this assignment is not a requirement. In Example 1, we have created two cell types, Medium and Foam, with Medium assigned a **TypeId** of 0 and Foam a **TypeId** of 1.

### 3.3 Contact Energy Plugin

Energy calculations for the foam simulation are based on the boundary or contact energy between cells (or surface tension, if you prefer). The total energy of the foam is simply the total boundary length times the surface tension (here defined to be  $2J$ ).

The explicit formula for the energy is:

$$E_{\text{adhesion}} = \sum_{i,j,\text{neighbors}} \mathbf{J}(\tau_{\sigma(i)}, \tau_{\sigma(j)}) (1 - \delta_{\sigma(i),\sigma(j)})$$

where  $i$  and  $j$  label two neighboring lattice sites,  $\sigma$  s denote cell Ids,  $\tau$  s denote cell types. Once again you need to differentiate between cell types and cell Ids. This formula shows that cell types and cell Ids are not the same. The Contact plugin in the .xml file, defines the energy per unit area of contact between cells of different types ( $\mathbf{J}(\tau_{\sigma(i)}, \tau_{\sigma(j)})$ ) and the interaction range (**Depth**) of the contact:

```
<Plugin Name="Contact">
<Energy Type1="Foam" Type2="Foam">3</Energy>
<Energy Type1="Medium" Type2="Medium">0</Energy>
<Energy Type1="Medium" Type2="Foam">0</Energy>
<Depth>2.0</Depth>
</Plugin>
```

In this case, the interaction range is 2, fourth-nearest neighbor, foam cells have a contact energy per unit area of 3 and foam and medium and medium and medium have a contact energy of 0 per unit area.

### 3.4 PIF Initializer

To initialize the configuration of the simulation lattice you can use one of the built-in lattice **initializers** (we will show one in the next example), or you can write your own **lattice initialization file**. Our experience suggests that you will probably have to write your own initialization files rather than relying on built-in initializers. The reason is simple: the built-in initializers implement very simple cell layouts, and if you want to study more complicated cell arrangements, the built-in initializers will not be very helpful. Therefore we encourage you to learn how to prepare lattice initialization files. Again, file definition is not complicated and we will explain every step. The lattice initialization file tells CompuCell3D how to lay out assign the simulation lattice pixels to cells.

The **Potts Initial File (PIF)** is a simple file format that we created for easy specification of initial cell positions. The PIF consists of multiple lines of the following format:

```
cell# celltype x1 x2 y1 y2 z1 z2
```

Where `cell#` is the unique integer index of a cell, `celltype` is a string representing the cell's initial type, and `x1` and `x2` specify a *range* of x-coordinates contained in the cell (similarly `y1` and `y2` specify a range of y-coordinates and `z1` and `z2` specify a range of z-coordinates). Thus each line assigns a rectangular volume to a cell. If a cell is not perfectly rectangular, multiple lines can be used to build up the cell out of rectangular sub-volumes (just by reusing the `cell#` and `celltype`).

A PIF can be provided to CompuCell3D by including the steppable object **PIFInitializer**.

Let's look at a PIF example for foams:

```
0 Medium 0 101 0 101 0 0
1 Foam 13 25 0 5 0 0
2 Foam 25 39 0 5 0 0
3 Foam 39 46 0 5 0 0
4 Foam 46 57 0 5 0 0
5 Foam 57 65 0 5 0 0
6 Foam 65 76 0 5 0 0
7 Foam 76 89 0 5 0 0
```

These lines define a background of Medium which fills the whole lattice and is then overwritten by seven rectangular cells of type Foam numbered 1 through 7. Notice that these cells lie in the *xy* plane (`z1=0 z2=0` implies that cells have thickness =1) so this example is a two-dimensional initialization.

You can write the PIF file manually, but using a script or program that will write PIF file for you in the language of your choice (Perl, Python, Matlab, Mathematica, C, C++, Java or any other programming language) will save a great deal of typing. We will provide some sample scripts that you will be able to modify later to create your own PIF files.

We will use simple script which creates initialization file for foam simulation. The syntax to use the script is the following:

```
./FoamInit.pl -r<row_size> -i<number of rows> -o<PIF file name> -z<random ratio> -m<min_width>.
```

What the script does it divides lattice into rows of width `<row_size>` then in each row it creates rectangular cells of height defined by `<row_size>` and width chosen randomly from interval [`<min_width>`, `<min_width>*<random ratio>`].

Let's do few exercises.

**Note:** Generate a new subdirectory for each run for each exercise that you do and do run in that subdirectory so that you can identify which output files belong to which simulation parameters. Type `mkdir <new_directory>`, then `cd <new_directory>`, finally start the simulation in the new directory by typing `<path_to_CompuCell_run_script>/run_CompuCellNew.sh`

### Exercise 1

Let's generate different initial condition by modifying `<min_width>` and `<random ratio>`. For example let's run `FoamInit.pl`: with the following arguments

```
./FoamInit.pl -r5 -i60 -ofoaminit2D_1.pif -z2 -m10.
```

The last command will create a lattice initialization file called "foaminit2D\_1.xml". The lattice will be 301x301 (5\*60+1) and will consist of 60 rows each of which is 5 pixel tall. Each row is divided into rectangular cells of randomly chosen width. The width is chosen from the interval [10,20] pixels.

Now in the .xml configuration file for foam simulation you need to change the line `<PIFName>foaminit2D.pif</PIFName>` to `<PIFName>foaminit2D_1.pif</PIFName>`

This ensures that CompuCell3D will use `foaminit2D_1.pif` initialization file that you have just created.



Observe what happens to the simulation. You may want to play with other values as well.

### Exercise 2

Change interaction range in the contact energy. In line `<Depth>2.0</Depth>` change 2.0 to 1.0, 1.43, 1.75 and see what happens.

Note: There is a pattern here, namely if in contact energy calculations you want to include 1 nearest neighbor you type:

`<Depth>1.0</Depth>`

If it is second nearest neighbor you use number that is slightly greater than  $\sqrt{2}$  in our case it is 1.43.

`<Depth>1.43</Depth>`

For third nearest neighbor it will be number slightly greater than  $\sqrt{3}$  - in our case it is 1.75 and so on.

Note that the Number of Neighbors ( $NN$ ) in two Dimensions goes as follows: For Nearest Neighbors (Depth=1),  $NN=4$ , For Next Nearest neighbors (Depth=1.43),  $NN=8$ , For Third neighbors (Depth=1.75),  $NN=12$ , For fourth neighbors (Depth=2)  $NN=20$  and for fifth neighbors (Depth=2.86)  $NN=24$ . See pictures below. Numbers on the grid denote order of neighbor (1<sup>st</sup>, 2<sup>nd</sup>, etc...).

2	1	2
1	X	1
2	1	2

5	4	3	4	5
4	2	1	2	4
3	1	X	1	3
4	2	1	2	4
5	4	3	4	5

You should see that for longer interaction ranges and the same temperature and interaction energy, the evolution is slower and the boundary walls are smoother. This is because longer interaction ranges have smaller LATTICE ANISOTROPY. That is, the variation in the energy of a wall as a function of the angle with respect to the underlying lattice is less for longer interaction ranges. E.g. for a DEPTH of 1 (Nearest neighbors), the energy of a wall at angle 0 or 90 degrees with respect to the lattice is 1 and for a wall at 45 or 135 degrees, the energy per unit length is  $\sqrt{2}$ .

Observe what happens to the simulation when you change interaction range.

### Exercise 3

Run simulation with different temperatures: 0, 0.5J, J, 3J, 10J where J is contact energy coefficient between foam cells. (`<Energy Type1="Foam" Type2="Foam">3</Energy>` in xml configuration file).

In this exercise you will change the TEMPERATURE. In the Potts Section of the code This line looks like `<temperature>5</temperature>`. So you will use, e.g.

<temperature>0</temperature>

or

<temperature>10</temperature>

Etc...

Letter J refers to the line which specifies J (contact energy coefficient) because what matters is the ratio  $J \cdot \text{Number of neighbors} / T$ . So for larger J, T needs to be bigger to have the same effect and for a longer neighbor interaction range, T must be bigger to have the same effect, because you have more neighbors. Note that the Number of Neighbors (NN) in two Dimensions goes as follows: For Nearest Neighbors, NN=4, For Next Nearest neighbors, NN=8, For Third neighbors, NN=12, for fourth neighbors NN=20 and for fifth neighbors NN=24. The point of this exercise is to study two things: the effect of lattice anisotropy and the transition from 'ferromagnetic' to disordered behavior. For T=0 for an interaction range of 1 (Depth), the pattern will FREEZE--it will not evolve at all, because all the boundaries will line up along low energy directions and T=0 means that only steps that DECREASE the pattern energy are allowed.

At T=0 for an interaction range (Depth) of 2 the pattern will evolve normally.

For higher T and the same interaction range and J, the boundaries will become rougher. You SHOULD find that as T increases, the coarsening rate increases up to some threshold value of J and then decreases again as T becomes very large. At very high T, the pattern SHOULD MELT. This is a ferromagnetic to isotropic phase transition. The bubbles will fall apart and the spin values will be random. This occurs because the energy of a pixel is always very small compared to T, so EVERY spin flip is accepted, even if it makes the boundaries longer.

#### Exercise 4

Now let's do quantitative exercise. In xml file add a steppable which for every cell will output cell id, volume, surface and number of neighbors. Simply place the following lines in your xml file:

```
<Steppable Type="FoamDataOutput" Frequency="10">
  <Output CellID="" Volume="" Surface="" NumberOfNeighbors="" FileName="data"/>
</Steppable>
```

Did you paste it in the right place?

The syntax is simple, as you can see, namely you specify frequency with which this steppable is called, and for the element Output you list properties of the cell that you want to output and give optional name of the output file (the default file name is "Output"). For the above example CompuCell will produce the following files: data.10, data.20, data.30 and so on. Each of these files will have the following contents:

73	131	64	8
74	46	32	6
75	57	50	5
76	16	18	5
77	106	58	8
79	6	10	3
80	84	48	7
13	70	46	5
2	128	66	7

Columns from left to right denote: cell id, volume, surface, number of neighbors.

Given that, data plot average bubble area as a function of time. For several chosen output files calculate average bubble area. Then plot this average area  $\langle A \rangle$  as a function of time.

Hints: for exercises requiring quantitative estimations and extraction of data from output files you may use Excel and use its sorting capabilities (Data->Sort...) . First open file in Excel (File->Open...) - this will take through several dialogs for importing text file into spreadsheets. Most of these dialogs are fairly well explained. Then once you have your spreadsheet ready you may select all the columns and go to the Data->Sort... to sort data with respect to a given column. Now, you should be able to do most of the exercises.

#### Exercise 5

For a chosen time (output data file) find average area of n-sided bubble. Plot a histogram of  $\langle A_n \rangle$  as a function of n.

### Exercise 6

For few chosen output files plot histograms of  $p(n)$ - probability of finding  $n$  sided bubble in at time  $t$  (*i.e.* in a given output file) and  $p(A/\langle A \rangle)$ - probability of finding bubble of area  $A/\langle A \rangle$  at time  $t$  (*i.e.* in a given output file)

Notice that  $p(n)$  is a ratio of number of  $n$ -sided bubbles and the number all the bubbles. Analogously for the  $p(A/\langle A \rangle)$ .

### Exercise 7

Generate different initial condition and check if  $p(A/\langle A \rangle)$  as a function of  $t$  and  $A/\langle A \rangle$  as a function of  $t$  depend on initial condition.

### Exercise 8

Change the boundary conditions from periodic to no flux. Does the pattern look different? Why?

Note: Do all of your other simulations with periodic boundary conditions.

### Exercise 9

Evaluate  $\langle n_A \rangle$  - the average number of sides of a bubble of area  $A/\langle A \rangle$ . You will have to use intervals of a convenient width in  $A/\langle A \rangle$  to get better statistics.

### Exercise 10

By comparing sequential files and looking for bubbles whose numbers of sides do not change between files,

calculate  $\frac{dA_n}{dt}$ . Does  $\frac{dA}{dt} = \kappa(n - 6)$  (*i.e.* does the foam obey von Neumann's law)? Why or why not?

Note that if you don't check that the number of sides doesn't change before you calculate your derivative, you will get the wrong answer. Why?

### Exercise 11

Do foams reach a scaling state (a state where the average area grows but the statistical properties remain constant)? When? When do they not do so?

### Exercise 12 (Extra Credit)

Repeat the above exercises in three dimensions, substituting number of faces for number of sides and volume for area. How does a three-dimensional foam differ from a two-dimensional foam?

## 4. A Slightly More Complex Simulation—Cell Sorting

Another relatively simple CompuCell simulation models biological cell sorting. In this simulation you start with a mixture of different cell types with different adhesivities to each other.

Embryonic cells of two different types, when dissociated, randomly mixed, and reaggreated can spontaneously sort to reestablish coherent homogenous tissues. Both complete and partial cell sorting (in which large clusters of one cell type are trapped inside a continuous structure of another type) have been observed experimentally in vitro in embryonic cells. Sorting is a key step in regeneration of a normal animal from aggregates of dissociated cells of adult hydra and involves neither cell division nor differentiation but only spatial rearrangement of cell positions. Physically cell sorting is caused by differences in adhesivities. In this simulation you will be able to verify how different hierarchies of adhesion coefficients will lead to different types of sorting.

In this set of exercises we will explore the role of the area (or volume constraint) and the effects of differential surface adhesivity. As you recall from lectures, these are described by the following equations:

$$E_{\text{volume}} = \lambda_{\text{volume}} (V_{\text{cell}} - V_{\text{target}})^2 - \text{volume constraint}$$

$$E_{\text{adhesion}} = \sum_{i,j,\text{neighbors}} \mathbf{J}(\boldsymbol{\tau}_{\sigma(i)}, \boldsymbol{\tau}_{\sigma(j)}) (1 - \delta_{\sigma(i),\sigma(j)}) - \text{contact energy}$$

We may add surface constraint but it is not required for cell sorting to work and for the sake of simplicity will be omitted.

You will be given a template xml file (*cellsort\_2D.xml*) with simulation description. For most of the exercises your task will be to slightly modify this file, run the simulation and describe the results. To add another twist to the exercises we do not include step-by-step instruction what needs to be modified. We leave it to you as an exercise.

However, if you feel you are stuck **ask for help**.

One thing to remember here is that this is a 2D simulation. That is, the lattice is 1 pixel thick and thus volume of the cell on such lattice is numerically equal to volume of the cell. Therefore, in exercises 1-4, we may interchange words “surface”, “volume”, “area”. They all really refer to cell volume.

However, in exercises 5-6 “surface” and “volume” mean different things and you will be using separate constraints for surface and volume. It is important to realize that if your cells are only one pixel thick, then the surface of such cells (and here I mean true surface) is numerically equal to perimeter. **This means that area of the cell on plane  $z=0$  and  $z=1$  does not count towards the surface.** In this way we simulate 2D behavior in CompuCell3D.

Throughout the exercises we use the following convention:

*L* – denotes “light” cells (NonCondensing in CompuCell terminology)

*D* – denotes “dark” (Condensing in CompuCell terminology)

*M* – denotes “medium” (Medium in CompuCell terminology)

*Example:*  $J_{LD}$  denotes adhesion coefficient between light (NonCondensing) and dark (Condensing) cells.

Before you begin, please copy *cellsort\_2D.xml* to your private directory. Issue the following command:

```
cp /Users/mswat/CompuCellFull5_install/cellsort_2D.xml <your private directory>
```

Remember also to create a separate directory for every simulation that you run and of course copy *cellsort\_2D.xml* to this new directory.

To create directory use the following command:

```
mkdir <directory> . Example: mkdir Exercise_1_a
```

After you copy *cellsort\_2D.xml* to this directory you may rename it to more a meaningful name:

```
mv cellsort_2D.xml cellsort_exercise_1_a.xml
```

Above I renamed file called *cellsort\_2D.xml* to *cellsort\_exercise\_1\_a.xml*.

In Unix renaming a file is done using *mv* command

**Hint:** In all of the exercises below it can happen that cells may stick to the wall. The simplest way to avoid this type of is to set boundary conditions in “x” and “y” direction (since we are in 2-D).

First let's look at the entire *.xml* description file for cell sorting before going into more detail:

```
<CompuCell3D>
<Potts>
```

```

<Dimensions x="100" y="100" z="1"/>
<Anneal>10</Anneal>
<Steps>10000</Steps>
<Temperature>10</Temperature>
<Flip2DimRatio>1</Flip2DimRatio>
<FlipNeighborMaxDistance>1.75</FlipNeighborMaxDistance>
</Potts>

<Plugin Name="PlayerSettings">
  <!--Project2D XYProj="2"/>
  <InitialProjection Projection="xy"/-->
  <Rotate3D XRot="27" YRot="-11"/>
</Plugin>

<Plugin Name="Volume">
  <TargetVolume>25</TargetVolume>
  <LambdaVolume>2.0</LambdaVolume>
</Plugin>

<!--
<Plugin Name="Surface">
  <TargetSurface>20</TargetSurface>
  <LambdaSurface>1.5</LambdaSurface>
</Plugin>
-->

<!--Plugin Name="CellSortType" /-->

<Plugin Name="CellType">
  <CellType TypeName="Medium" TypeId="0"/>
  <CellType TypeName="Condensing" TypeId="1"/>
  <CellType TypeName="NonCondensing" TypeId="2"/>
</Plugin>

<Plugin Name="Contact">
  <Energy Type1="Medium" Type2="Medium">0</Energy>
  <Energy Type1="NonCondensing" Type2="NonCondensing">16</Energy>
  <Energy Type1="Condensing" Type2="Condensing">2</Energy>
  <Energy Type1="NonCondensing" Type2="Condensing">11</Energy>
  <Energy Type1="NonCondensing" Type2="Medium">16</Energy>
  <Energy Type1="Condensing" Type2="Medium">16</Energy>
  <Depth>1.75</Depth>
</Plugin>

<Plugin Name="CenterOfMass"/>

<Steppable Type="OutputData"/>

<Steppable Type="BlobInitializer">
  <Gap>0</Gap>
  <Width>5</Width>
  <CellSortInit>yes</CellSortInit>
  <Radius>40</Radius>
  <!--Engulfment BottomType="Condensing" TopType="NonCondensing"
EngulfmentCoordinate="y" EngulfmentCutoff="50"/-->

```

```

</Steppable>

<!--Steppable Type="CellDiffusionDataOutput" Frequency="100">
  <OutputFormat FileName="Diffusion" DeltaPosition="" />
  <OutputID CellID="1" />
  <OutputID CellID="2" />
  <OutputID CellID="3" />
</Steppable-->

```

#### 4.1 Volume and Surface Constraints

```

<Plugin Name="Volume">
<TargetVolume>25</TargetVolume>
<LambdaVolume>2.0</LambdaVolume>
</Plugin>

<Plugin Name="Surface">
<TargetSurface>20</TargetSurface>
<LambdaSurface>1.5</LambdaSurface>
</Plugin>

```

These two plugins inform CompuCell that the Hamiltonian will have two additional terms associated with volume and surface conservation. That is when spin flip is attempted one cell will increase its volume and another cell will decrease. Thus overall energy of the system may or will change. Volume constraint essentially ensures that cells maintain the volume which close (this depends on thermal fluctuations) to target volume . The role of surface plugin is analogous to volume, that is to “preserve” surface. Note that surface plugin is commented out in the example above.

Energy terms for volume and surface constraints have the form:

$$E_{\text{volume}} = \lambda_{\text{volume}} (V_{\text{cell}} - V_{\text{target}})^2$$

$$E_{\text{surface}} = \lambda_{\text{surface}} (S_{\text{cell}} - S_{\text{target}})^2$$

Notice that flipping a single spin may cause surface change in more that two cells – this is especially true in 3D.

#### 4.2 Calculating Surface and Volume Constraints

Now that we have introduced volume and surface constraint we recommend that you go over few exercises that will teach you how to properly choose volume and surface constraint parameters so that cells do not freeze and do not disappear. One can of course use trial and error method but it is instructive to learn how these parameters can be estimated rather than guessed. If you are new to CompuCell3D you may skip this section at this time, however we strongly encourage you do come back to these exercises , as they show how to systematically pick parameters of most important energy terms.

This set of exercises was written by Dr. Nikodem Poplawski.

##### Exercise v1.

- a) Consider a 2D cell which consists of 1 (one) pixel. The energy of such a system is  $H_1 = 8J_{cm} + \lambda_V (V_T - 1)^2$ , where  $J_{cm}$  is the adhesion coefficient between the cell and the medium. Here, 8 is the number of the second nearest neighbors in 2D (of course, you can reduce or extend the neighbors). First, set  $T = 0$ . Your cell should disappear, if the energy of the system containing only the medium,

$H_0 = \lambda_V V_T^2$ , is lower,  $H_0 < H_1$ . This occurs if  $\lambda_V < \lambda_{V_0}$ , where  $\lambda_{V_0} = \frac{8J_{cm}}{2V_T - 1}$  (check it, you can set  $J_{cm} = 10$  and  $V_T = 25$ , for which  $\lambda_{V_0} = 1.63$ ). Otherwise, the cell should survive. For values of  $\lambda_V$  much higher than  $\lambda_{V_0}$ , your cell will grow until its volume reaches  $V_T$ .

b) Try  $T > 0$ . In this case, the processes increasing the energy of the system have a nonzero probability, but  $\lambda_{V_0}$  should still be the approximate threshold for stability of a 1-pixel cell. Check it, repeating point a).

### Exercise v2.

Now, consider a cell which is a square  $N \times N$ . The energy of this cell is

$H_N = (12N - 4)J_{cm} + \lambda_V (N^2 - V_T)^2$ . Assuming that the cells can take the form of squares only (otherwise the calculations would be too lengthy), we can ask the question what the equilibrium value  $N_0$  is. This value is

given by  $\frac{\partial H_N}{\partial N} = 0$ , and since  $\frac{\partial H_N}{\partial N} = 12J_{cm} + 4\lambda_V N(N^2 - V_T)$ , we arrive at  $N_0^3 - V_T N_0 + \alpha = 0$ , where

$\alpha = \frac{3J_{cm}}{\lambda_V} \geq 0$ . The above equation is cubic in  $N_0$ , and has three real solutions (one stable corresponding to

equilibrium) if  $\alpha < \alpha_{\max}$ , where  $\alpha_{\max} = 2\left(\frac{V_T}{3}\right)^{3/2}$ . Otherwise, there is only one real root which is negative, thus

there is no equilibrium value of  $N$ . Therefore, if  $\lambda_V < \lambda'_{V_0}$ , where

$\lambda'_{V_0} = \frac{3J_{cm}}{2\left(\frac{V_T}{3}\right)^{3/2}}$ , our cell will shrink and disappear (note that  $\lambda'_{V_0} < \lambda_{V_0}$  so we are in the region where a single pixel is unstable).

- For  $N = 5$ ,  $J_{cm} = 10$ , and  $V_T = 25$ , start from  $\lambda_V$  much larger than  $\lambda'_{V_0}$  (here equal to 0.62) and set  $T = 5$ . Your cell should approach the target volume  $V_T$ .
- Decrease  $\lambda_V$ . The cell volume should tend to a value smaller than  $V_T$ .
- Try  $\lambda_V$  just a little larger than  $\lambda'_{V_0}$ . The cell volume should approach the equilibrium value  $\frac{V_T}{3}$ .
- Try  $\lambda_V$  smaller than  $\lambda'_{V_0}$ . Your cell should shrink and disappear. There is no equilibrium volumes smaller than  $\frac{V_T}{3}$ .
- You may want to play with different values of  $V_T$ .

### Exercise v3.

In the Cellular Potts Model, there are two critical temperatures. Around the temperature of dissociation,  $T_{cl} = 8\delta E$ ,

where  $\delta E = J_{cm} - \frac{J_{cc}}{2}$ , a single pixel has a long life-time, and a cell may dissociate (single spins disconnect from the rest of the cell). When the temperature is of the order of  $T_{c2} = N^2 \delta E$ , for which the energy and entropy of the system are comparable (the temperature of spinoidal decomposition), the identities of the cell pixels and the medium pixels merge. The cell should “evaporate”.

- a) Begin with a  $N \times N$  square cell ( $N = 5$ ) with  $V_T = N^2 = 25$ ,  $J_{cm} = 10$ ,  $J_{cc} = 2$ , and  $\lambda_V$  much (not too much) larger than  $\lambda_{v0}$  (here equal to 0.62). Start from  $T = 0$  (nothing unusual should happen). Then, increase  $T$  until it reaches  $T_{c1} = 72$ . Check if the cell dissociates.
- b) Increase  $T$  further until it reaches  $T_{c2} = 225$ . See what happens. There should be a competition between  $T$  and  $\lambda_V$  in trying to save the cell from disappearance.

Let us come back to the description of the cell sort simulation.

### 4.3 Contact energy

Now the most important plugin in the cell sort simulation. Again, let's look at the contact plugin for the cell sort. It is a somewhat more complicated that corresponding plugin for foam simulation but the idea is the same

```
<Plugin Name="Contact">
  <Energy Type1="Medium" Type2="Medium">0</Energy>
  <Energy Type1="NonCondensing" Type2="NonCondensing">16</Energy>
  <Energy Type1="Condensing" Type2="Condensing">2</Energy>
  <Energy Type1="NonCondensing" Type2="Condensing">11</Energy>
  <Energy Type1="NonCondensing" Type2="Medium">16</Energy>
  <Energy Type1="Condensing" Type2="Medium">16</Energy>
  <Depth>1.75</Depth>
</Plugin>
```

The last object that shows up in the configuration file is a steppable called BlobField initializer. Remember, I told you that steppables are called every MCS. That is true, except when steppable is some kind of initializer. In this case steppable is called once at the beginning of the simulation. What this initializer does, it creates a blob of cells. Each cell is a cube  $5 \times 5 \times 1$  (`<Width>5</Width>`) and they are tightly packed (`<Gap>0</Gap>`). The additional line `<CellSortInit>yes</CellSortInit>` is used exclusively for cellsort simulation and tells the initializer that cell types will be only 0,1,2 or Medium, Condensing, NonCondensing if you prefer. You can also specify radius of the blob although this is not a requirement. If you do not specify the radius it will be equal to  $2/5 \times \text{lattice dimension}$ . If you want to increase or decrease radius from its default value, use `<Radius>40</Radius>` option. Any space in the lattice unfilled with cells becomes Medium *i.e.* effectively all the cells are immersed in Medium (unless the radius is so big that cells fill entire lattice).

This initializer is one of CompuCell3D stock initializers, so that you do not need to prepare your own PIF initialization file.

```
<Steppable Type="BlobInitializer">
  <Gap>0</Gap>
  <Width>5</Width>
  <CellSortInit>yes</CellSortInit>
  <Radius>40</Radius>
  <Engulfment BottomType="Condensing" TopType="NonCondensing" EngulfmentCoordinate="y"
  EngulfmentCutoff="50"/>
</Steppable>
```



In your initial simulation you will omit Engulfment entry for BlobInitializer. That's why it is commented out now. This and [CellDiffusionDataOutput](#) will be explained later in the text. Now let's do some exercises.

### Exercise 1

As we did for foams, explore the effects of changing the temperature on the pattern. We now have additional parameters (the strength of the volume constraint and the target volume as well as the surface energy), although, as before, only the ratio of energy to temperature matters.

- If the temperature  $T=0$  what happens (should freeze)
- If  $t$  is intermediate what happens?
- If  $t$  is very large what happens? (edges should become rough)
- What is the range of  $T$  over which these three regimes occur (is there a large intermediate regime)?

### Exercise 2

Now look at what happens as the strength of the area constraint changes?

- If  $\lambda_v=0$  what happens? (should disappear)
- If  $\lambda_v$  is intermediate, what happens?
- If  $\lambda_v$  is very large what happens? (Should freeze)
- What is the range of  $\lambda_v$  over which these three regimes occur (is there a large intermediate regime)?

### Exercise 3

Now we will look at the effects of the contact energies. You have five energies to play with.

Let  $0 < J_{LD} < J_{DD} < J_{LL} < J_{LM} = J_{DM}$ . What happens? Make sure that you adjust  $\lambda$  and  $T$  to be in the "middle" regime

Let  $0 < J_{DD} < J_{LD} < J_{LL} < J_{LM} = J_{DM}$ . What happens? Now slowly vary  $J_{LD}$  from being just above  $J_{DD}$  to being just below  $J_{LL}$ . Does the final pattern change? Does the kinetics of the pattern evolution change?

Let  $0 < J_{DD} < J_{LL} < J_{LD} < J_{LM} = J_{DM}$ . What happens? Is the result the same or different from b)? If  $J_{LD}$  is just above  $J_{LL}$  is the result different from if  $J_{LD}$  is much bigger than  $J_{LL}$ ?

Let  $0 < J_{DD} < J_{LD} < J_{LL} < J_{DM} < J_{LM}$ . What happens? Is the result different if the difference between  $J_{DM}$  and  $J_{LM}$  is small from what happens if it is very large?

Let  $0 < J_{DD} < J_{DM} < J_{LL} < J_{LM} < J_{LD}$ . What happens? Why?

Let  $0 < J_{LM} = J_{DM} < J_{DD} < J_{LD} < J_{LL}$ . What happens? Why?

### Remark:

For exercises 4,5,6 please use file

*/Users/mswat/CompuCellFull5\_install/cellsort\_engulfment\_diffusion\_2D.xml*

as a template (copy it into your private directory). Depending on the exercise you will need to modify it. However you will not need to type everything from scratch.

### Exercise 4

Now repeat Exercise 3 starting with the case of engulfment (light cells on the bottom and dark cells on the top)?

Hint . You can quickly initialize blob of cells with cells of one type at the bottom and of another at the top. Simply use, the following syntax in your [BlobInitializer](#) Steppable:

```

<Steppable Type="BlobInitializer">
  <Gap>0</Gap>
  <Width>5</Width>
  <CellSortInit>yes</CellSortInit>
  <Radius>40</Radius>
  <Engulfment BottomType="Condensing" TopType="NonCondensing"
EngulfmentCoordinate="y" EngulfmentCutoff="50"/>
</Steppable>

```

As you can see it looks almost the same as in the previous exercises except there is new line which describes engulfment. As you can infer from this syntax, you may specify which cells are at the top, which are at the bottom, and by changing the value of EngulfmentCutoff you may fix how many cells of a given type there should be. Try to play with this value (remember it must be positive integer) and figure out by yourself how it works. Try to find also what EngulfmentCoordinate means. The allowed values are 'x', 'y', 'z'.

### Exercise 5

Actually, when cells stick together, the energy between them should be negative.

- Repeat Exercise 3 for all energies negative (same hierarchy). What happens?
- Now include the surface area constraint as well. Repeat the exercises in Exercise 3. Are the results different? Vary the target surface area as a function of the target volume? What happens? For a fixed target surface area, vary the strength of the corresponding constraint. Can you identify the same three regimes as in exercises 1 and 2? What are these three regimes in this case?
- Finally, let  $J_{LL}$ ,  $J_{DD}$  and  $J_{LD}$  be negative and  $J_{LM}$  and  $J_{DM}$  be positive. How do your results compare to part a) and b)?

### Exercise 6

The diffusion of cells is a critical property. Use only one cell type and a fixed value of the target volume and surface area and their constraints. Measure the diffusion constant of a single cell (the slope of a plot of mean displacement vs  $t^2$ ).

- How does the diffusion constant vary with  $T$ ?
- How does the diffusion constant vary with  $J$ ?
- How does the diffusion constant vary with  $\lambda$ ?
- We usually think that only energy differences matter, not absolute energy values. Do two diffusion experiments, one where  $J > 0$  and the other when  $J < 0$ , but all  $\square J$  and other parameters are the same. Are the diffusion constants the same? Why?

For this Exercise you will use CompuCell3D module called [CellDiffusionDataOutput](#). This module allows you to pick one or several cell and every predefined number of Monte Carlo Steps it will output Center of Mass position of these cells as well as total displacement from initial position.

To load this module into your simulation put the following lines into your xml simulation description file:

```
<Steppable Type="CellDiffusionDataOutput" Frequency="100">
  <OutputFormat FileName="Diffusion" DeltaPosition="" />
  <OutputID CellID="1" />
  <OutputID CellID="2" />
  <OutputID CellID="3" />
</Steppable>
```

Every line of the xml description that begins with **OutputID** tells CompuCell3D to track one cell with id given by value of CellID attribute. Here we are tracking three cells with Id's 1,2,3 respectively.

The **OutputFormat** line describes what core name should be given to output files and whether to include total displacement in the output file (by default CompuCell will output only consecutive positions of center of mass of cells that it is tracking).

The output file will look something like this:

```
1 100 41.8 16.48 0 0.557136
1 200 44.7391 17.3043 0 2.75599
```

Columns from left to right denote: cell ID, MCS, xCM, yCM, zCM, displacement from initial position. CM above stands for Center of Mass.