1 Spatial Stochastic Modeling with MCell and CellBlender

Sanjana Gupta[1], Jacob Czech[2], Robert Kuczewski[3], Thomas M. Bartol[3], Terrence J. Sejnowski[3], Robin E. C. Lee[1], and James R. Faeder[1]

[1]*Department of Computational and Systems Biology, School of Medicine, University of Pittsburgh, Pittsburgh PA 15260 USA.*

[2]*Pittsburgh Supercomputing Center, Carnegie Mellon University, Pittsburgh, PA 15213 USA.*

[3]*Howard Hughes Medical Institute, The Salk Institute for Biological Studies, La Jolla, California 92037, USA.*

This chapter provides a brief introduction to the theory and practice of spatial stochastic simulations. It begins with an overview of different methods available for biochemical simulations highlighting their strengths and limitations. Spatial stochastic modeling approaches are indicated when diffusion is relatively slow and spatial inhomogeneities involve relatively small numbers of particles. The popular software package MCell allows particle-based stochastic simulations of biochemical systems in complex three dimensional (3D) geometries, which are important for many cell biology applications. Here, we provide an overview of the simulation algorithms used by MCell and the underlying theory. We then give a tutorial on building and simulating MCell models using the CellBlender graphical user interface, that is built as a plug-in to Blender, a widely-used and freely available software platform for 3D modeling. The tutorial starts with simple models that demonstrate basic MCell functionality and then advances to a number of more complex examples that demonstrate a range of features and provide examples of important biophysical effects that require spatially-resolved stochastic dynamics to capture.

1.1 Introduction: Why stochastic spatial modeling?

Mathematical and computational models are useful tools that can be used to make inferences about experimental data as well as predictions about a system of interest [1, 2]. As outlined in Table 1.1, there are many different modeling formalisms available with varying degrees of spatial and molecular resolution.

The degree of spatial resolution roughly divides these methods into two categories:

1. Non-spatial — ordinary differential equations (ODE), stochastic simulation algorithms (SSA), network-free simulations (NF);
2. Spatial — partial differential equations (PDE), reaction diffusion master equations (RDME), and particle-based stochastic simulations.

The well-mixed assumption posits that each reacting species is homogeneously distributed throughout each cellular compartment being modeled. The typical distance that a molecule travels over its lifetime, called the Kuramoto length [3], can be used to assess the applicability of the well-mixed assumption. The Kuramoto length, $l_k = \sqrt{D\tau}$, where D and τ are the diffusion coefficient and average lifetime of a given species respectively, is compared with the length scale of the reaction volume, L:

1. If $l_k \gg L$ then local changes in concentration diffuse throughout the reaction volume and the system remains homogeneous/well-mixed.
2. If $l_k \ll L$ then concentration changes remain localized and the well-mixed assumption breaks down, requiring application of spatial modeling methods.

The degree of molecular resolution, i.e., whether the molecular species are modeled as continuous populations, discrete populations, or discrete individuals, further differentiates the modeling methods, as shown in the second row of Table 1. We now describe these methods in more detail, beginning with the non-spatial methods and then proceeding to spatially-resolved methods, which are required when the well-mixed assumption no longer holds.

1.1.1 Non-spatial modeling formalisms

Ordinary differential equations (ODEs) based on reaction rate equations model continuous populations in a well-mixed system [4, 5]. These deterministic equations track the mean values of the molecular species populations, neglecting the stochastic fluctuations that occur when the number of molecules is small or the system has multiple steady states [3]. The ***chemical master equation*** (CME) is a set of coupled linear ODEs that describe the probability distributions of a set of molecular species populations. The CME is used

Table 1.1
Classification of different methods for biochemical simulation, with examples of available simulation tools.

Method *Attribute*	**ODE**[a]	**SSA**[b]	**NF**[c]	**PDE**[d]	**Spatial SSA**[e]	**Particle-based**[f]
Spatial resolution	Well-mixed	Well-mixed	Well-mixed	Lattice	Lattice	Continuous
Molecular resolution	Continuous populations	Discrete populations	Discrete individuals	Continuous populations	Discrete populations	Discrete individuals

[a]Ordinary differential equations [4, 5]. [b]Stochastic simulation algorithm [6]. [c]Network-free simulation algorithm, e.g., NFsim [7] [d]Partial differential equations, e.g., VCell [8]. [e]Spatial stochastic simulation algorithm, e.g., MesoRD [9] and URDME [10]. [f]Particle-based spatial stochastic simulation algorithm, e.g., MCell [11] and Smoldyn [12].

when stochastic effects are expected to be important, such as when either of the following is true:

1. The average number of particles n is small enough that the size of the fluctuations about the mean, which are order $\sqrt{n}$, become significant [3].
2. The system has multiple steady state attractors that can give rise to noise-induced transitions between states, which may be either fixed or oscillating [13].

The CME can only be solved directly for systems with a small number of degrees of freedom, but *system trajectories* can be computed using various methods including Gillespie's ***stochastic simulation algorithm*** (SSA) [6], and then binned or averaged to determine probability distributions and moments. Because the CME is a population-based method, it only permits tracking the collective behavior of species in the system. If the modeler is interested in tracking individual species, ***network-free*** (NF) simulation approaches can be used [7, 14]. These individual-based Monte-Carlo based algorithms have an additional advantage in that they avoid enumeration of the complete reaction network in advance, which can be prohibitively expensive for systems that exhibit combinatorial complexity [7].

1.1.2 Spatial modeling formalisms

There are many examples in biology of systems where the well-mixed assumption does not hold throughout the entire reaction volume, requiring a different set of modeling formalisms that take into account spatial effects.

Partial differential equations (PDEs) are used to model continuous molecular populations as they evolve in time and space. For simple systems, these can be solved analytically with a continuous representation of space [15–17]. However, most complex systems rely on numerical integration methods that discretize space [8]. As in the case of ODEs, PDEs are deterministic and break down when stochastic effects are important. An example of

this is the Min system in Escherichia coli. Low copy numbers of proteins in this system can produce spatial patterning that does not occur in the corresponding deterministic model [18]. Under some conditions, this system exhibits both a stable fixed point and a limit cycle attractor. Stochastic fluctuations drive the system into a limit cycle and render the stable fixed point unimportant for the observed dynamics [19].

The ***reaction diffusion master equation*** (RDME), which is the spatial extension of CME, divides the space into sub-volumes called voxels, within which well-mixed conditions are assumed to prevail [20, 21]. Reactions occurring within individual voxels are simulated using Gillespie's algorithm, and molecules move from one voxel to a neighboring one via discrete jumps to simulate diffusion. The set of voxels used to model a geometry is called a mesh: determining the geometry and resolution of the mesh is generally the most difficult part of performing RDME-based simulations. Rectangular mesh elements are appropriate for modeling relatively simple geometries, but the more complex geometries that are often encountered in realistic simulations of cells require so-called unstructured meshes, based on triangles or tetrahedrons. Software packages are available for both rectangular [22] and unstructured mesh elements [10]. Optimal determination of mesh resolution is important because the computational cost of these methods grows rapidly as the mesh size increases due to the increase in the number of diffusive jump events relative to the number of reaction events, but is unfortunately largely a matter of trial and error [21].

Particle-based spatial stochastic simulation methods simulate the trajectories of individual particles that represent the molecular species in the system. In addition to allowing the modeler to track individual particles, so-called off-lattice methods that form the basis for the popular MCell [11, 23] and Smoldyn [24] simulators, do not rely on discretization of three-dimensional (3D) space and can therefore circumvent the complexities involved in mesh generation mentioned above. At the same time, these tools enable modeling of complex 3D geometries that often arise in cell biology applications, such as studying the effects of macromolecular crowding, oligomerization, and self-assembly [25].

1.2 A brief overview of MCell

MCell, which is short for Monte Carlo Cell, is a particle-based reaction diffusion simulator [11, 23, 26–28]. A detailed description of the simulation algorithms used in MCell3, the current version, is presented in Kerr et al. [11]. Here, we describe the basic components of an MCell model and the basic elements of an MCell simulation step. An MCell simulation consists of a trajectory generated by a specified number of steps of a fixed time length.

1.2.1 Basic components

The basic components of an MCell model are:

Molecules Molecules are the fundamental components of the system and are simulated as diffusing point particles. Surface Molecules diffuse on 2D surfaces and Volume Molecules diffuse in 3D volume (Figure 1.1(A)).

Reactions Reactions define unimolecular or bimolecular reactions with mass action kinetics that determine how the molecules in the model interact with each other. These are discussed in more detail below.

Mesh Objects Mesh Objects, or "Objects" for short, define surfaces that limit the diffusion of associated Surface Molecules and may reflect, absorb, or transmit colliding Volume Molecules. Objects may be open (Figure 1.1(B)), but are more commonly closed in order to form compartments that restrict the diffusion of Volume Molecules (Figure 1.1(C)). Objects are composed of triangular faces, which allows the creation of complex curved geometries. Objects may be divided into Surface Regions, comprising a set of faces, that may affect molecule behavior and placement (Figure 1.1(C)). Properties of Surface Regions are assigned using Surface Classes. It is worth noting that within the MCell simulator, faces defining an Object are further divided into triangular tiles, each of which may only be occupied by a single Surface Molecule.

Release sites Release Sites define where molecules are placed at the start of a simulation or during a simulation. Examples of Release Sites are the surface of an Object, the interior or exterior of an Object, points in space, and predefined geometric shapes.

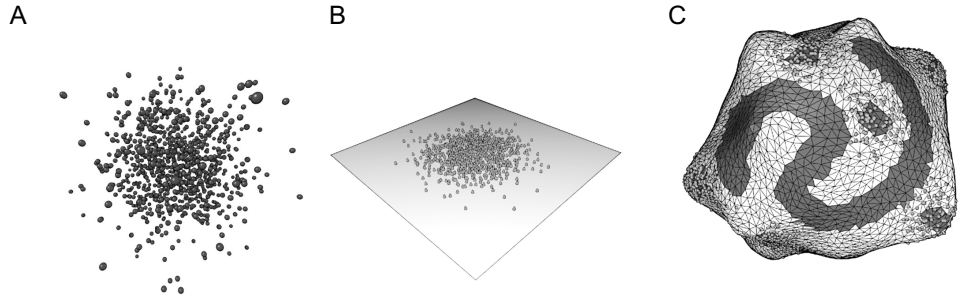

Figure 1.1

MCell Components. (A) Volume Molecules diffusing in free space. (B) Mesh Object defined by a Plane with Surface Molecules diffusing on it. (C) Mesh Object defined by a complex closed mesh with multiple defined Surface Regions, in which Surface Molecules have different diffusion constants, as defined by corresponding Surface Classes.

1.2.2 Simulation algorithm

At every time step in an MCell simulation, each particle can move, collide with other particles or surfaces, and undergo bimolecular and unimolecular reactions. The basic elements of a simulation step (Figure 1.2) are:

1. **Movement**. Particles move in random directions over straight-line trajectories of length sampled from probability distributions for diffusive motion, as described in more detail in Section 1.2.3.
2. **Detection of collisions**. Ray-tracing algorithms detect collisions between the diffusing particles and other particles or surfaces as it diffuses. As a particle moves along a straight-line trajectory, any particle within a specified collision radius is checked to determine if the pair undergoes a reaction (Figure 1.2). Ray marching algorithms propagate rays after collisions with surfaces.
3. **Bimolecular reactions**. When a collision occurs, the bimolecular reaction probability is a function of the user-specified bimolecular rate constant, the time step, and the diffusion constants of the particles. A Monte Carlo scheme is used to determine reaction firings, as described further in Section 1.2.4.
4. **Unimolecular reactions**. At any point along its trajectory a particle can undergo unimolecular transitions based on user-specified reactions. These transition events are scheduled whenever a particle is created or changes state and are determined probabilistically as a function of the rate constants for the allowed reactions, as described in more detail in Section 1.2.5.

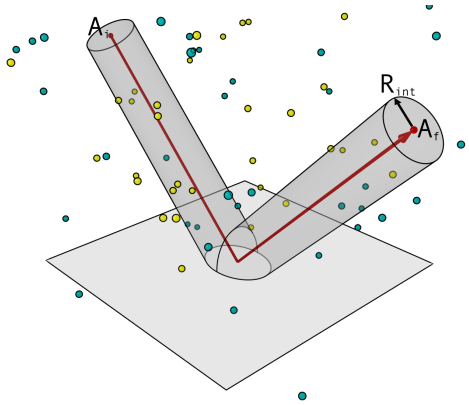

Figure 1.2

Simulation of a Volume Molecule time step in MCell. The particle A diffuses along a straight line path of a pre-determined length and is tested for reaction partners within a specified interaction radius, R_{int}, along its path. When a particle encounters a mesh tile, it is reflected according to specular reflection and continues in a straight line.

1.2.3 Particle diffusion

Following [11] we briefly describe the steps to simulate particle diffusion and the equations needed to determine both the magnitude and direction of particle displacement at each time step. Given a Volume Molecule at a point in 3D space at time $t = 0$, the probability density function for it having moved a distance r by time t is given by

$$\rho(r,t) = \frac{1}{(4\pi Dt)^{3/2}} e^{-r^2/4Dt}. \tag{1.1}$$

Inverse transform sampling is used to sample a displacement magnitude R from this distribution by computing $R = \mathrm{cdf}^{-1} X$, where X is a uniform random number between 0 and 1 and

$$\mathrm{cdf}(R,t) = \int_0^R \rho(r,t)\, 4\pi r^2 \, \mathrm{d}r. \tag{1.2}$$

This function can be expressed in terms of the Gaussian error function and its inverse computed from a lookup table [11].

In addition to the displacement magnitude, two angles are needed to determine the direction of displacement: the azimuthal angle ϕ and the polar angle θ. ϕ is chosen randomly from the uniform distribution on the interval $[0, 2\pi)$. θ is more complicated because the probability distribution is proportional to $\sin\theta$. Inverse transform sampling can be used to obtain θ from the equation $Y = (1 - \cos\theta)/2$, where Y is a uniform random number on $[0, 1]$. Since computing the inverse of a trigonometric function is computationally expensive, a modified form of table lookup is used to boost efficiency [11].

Overall, the steps for simulating Volume Molecule diffusion can be summarized as follows:

1. Determine the radial displacement R using inverse transform sampling of the cdf given in Equation 1.2.
2. Determine the direction of travel by sampling the polar angles θ and ϕ.

MCell also considers Surface Molecules that diffuse in 2D. As described in Section 1.2.1, Mesh Objects are composed of triangular Faces, which in turn are divided into smaller triangles called Tiles. Surface Molecules diffuse by hopping between tiles. Unlike Volume Molecules, which are treated as point particles and can exist at arbitrarily close points in space, Surface Molecules are restricted from occupying a Tile already occupied by another Surface Molecule. The algorithm for simulating 2D diffusion in MCell is similar to algorithms for simulating the Reaction Diffusion Master Equation [21] and are described in more detail in [11]. The basic steps can be summarized as follows:

1. Pick a direction of motion within the plane of the Tile on which the Molecule currently resides.

2. Ray-march along this vector.
3. As the ray crosses triangle boundaries, convert the vector from the local planar coordinate system of the first triangle to that of the second.

1.2.4 Bimolecular reactions

MCell checks for the occurrence of a bimolecular reaction whenever two particles collide. To obtain the correct bulk reaction rate, the probability of reaction per collision, p, must be chosen such the expected rate of collisions times the probability per collision equals the bulk reaction rate. We will briefly derive the reaction probability for the case of a Surface Molecule colliding with a Volume Molecule, and provide results for the surface-surface and volume-volume cases, which are derived in detail elsewhere [11].

Surface–volume reactions. Consider a Surface Molecule located on a tile of area A with an infinite column extending above (Figure 1.3). At equilibrium, the flux of molecules moving into this column equals the flux of molecules moving out of this column, and therefore we can restrict ourselves to the Volume Molecules within the column. The probability that a molecule starting at distance R away will reach the surface is

$$p_s(R) = \int_R^\infty \rho(x, \Delta t)\, \mathrm{d}x = \int_R^\infty \frac{1}{\sqrt{4\pi D \Delta t}}\, e^{-x^2/4D\Delta t}\, \mathrm{d}x = \frac{1}{2} \operatorname{erfc} \frac{R}{\sqrt{4D\Delta t}}, \tag{1.3}$$

where $\rho(x, \Delta t)$ is the probability density for a molecule to diffuse by a displacement x in time t. The number of molecules in a slice of the column of width $\mathrm{d}R$ with a concentration of Volume Molecules ρ_1 is $\rho_1 A\, \mathrm{d}R$, which allows us to determine the total number of Volume Molecules hitting the surface per time step as

$$n_s = \int_0^\infty p_s(R) \rho_1 A\, \mathrm{d}R = \int_0^\infty \frac{1}{2} \operatorname{erfc} \frac{R}{\sqrt{4D\Delta t}} \rho_1 A\, \mathrm{d}R = \frac{\rho_1 A \sqrt{D\Delta t}}{\sqrt{\pi}\, \Delta t}. \tag{1.4}$$

Now, setting the observed reaction rate, which is $n_s \times p$, equal to the bulk reaction rate, $k\rho_1$, allows us to solve for p to obtain

$$p = \frac{k}{A} \left(\frac{\pi\, \Delta t}{D} \right)^{1/2}. \tag{1.5}$$

We see that the probability of reaction per collision is proportional to the bimolecular rate constant k but the square root of Δt. If the reaction probability is greater than one, the reaction rate is considered *diffusion-limited*, because a reaction will occur any time the reacting particles collide and will be less than that specified by the bulk reaction rate constant. Unless one is explicitly trying to model diffusion-limited reactions, a reaction probability above one usually means that the step size needs to be reduced in order to prevent a loss of reaction flux.

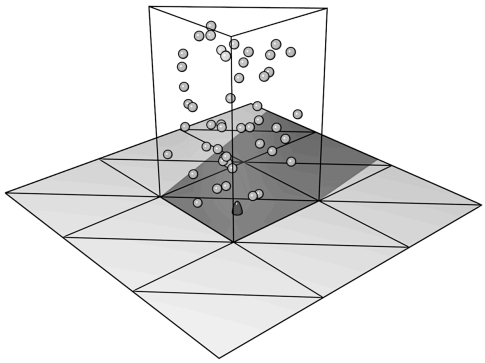

Figure 1.3

Surface-volume reactions. A Surface Molecule (dark protrusion) on a Tile of area A with a column directly above containing Volume Molecules (shaded spheres).

Surface-surface reactions. A similar derivation gives the reaction probability for bimolecular reactions between two Surface Molecules as

$$p = k\frac{\Delta t}{A}. \tag{1.6}$$

Volume–volume reactions. The reaction probability for bimolecular reactions between two Volume Molecules with diffusion length constants D_1 and D_2 respectively, is

$$p = k\frac{\sqrt{\pi\,\Delta t}}{4A_{\text{int}}\left(\sqrt{D_1}+\sqrt{D_2}\right)}, \tag{1.7}$$

where $A_{\text{int}} = \pi\, r_{\text{int}}^2$ and r_{int} is the collision radius of the two Volume Molecules.

1.2.5 Unimolecular reactions

When a new molecule is created in an MCell simulation, either through the occurrence of a reaction or at the start of a simulation, the time of the next unimolecular reaction that it will undergo is computed and added to a scheduler. The algorithm for determining the time and identity of the next unimolecular reaction is a special case of Gillespie's SSA [6].

1.3 Getting started with CellBlender and MCell

We now present a tutorial that describes how to use MCell via CellBlender, which is a plug-in for the popular 3D modeling software called Blender. CellBlender provides a graphical user interface (GUI) for building and simulating MCell models. Blender, CellBlender, and MCell are each freely available and open source software packages.

MCell models can also be built directly without the CellBlender GUI, using the Model Description Language (MDL). CellBlender generates MDL files automatically, making it possible for a user to create and run a model in MCell without learning MDL, but for advanced applications a knowledge of MDL may be useful. For more details about MDL see the MCell Quick Reference Guide and MCell Reaction Syntax documents available at `http://mcell.org/documentation`.

The tutorial presents a series of examples of increasing complexity that cover most major aspects of spatial modeling with CellBlender and MCell. Each example is divided into two parts: (1) CellBlender preliminaries, which describe how the modeled components are created using the interface, and (2) a modeling exercise, in which the user is instructed to build a specific model and results of the model are presented and discussed. Complete versions of all of the example models can be obtained from the MCell web site.

1.3.1 Obtaining the required software

The first step is to obtain CellBlender and MCell by following the directions at `http://www.mcell.org/tutorials/software.html`. This tutorial is based on CellBlender 1.1, and should be compatible with future version 1 releases. Figure 1.4 shows an overview of the CellBlender 1.1 interface. As shown in Figure 1.4, this tutorial was constructed using CellBlender with an ID of `ce75cdd9c7eb5374b55dfa499bf2155f3fc45595`.

Depending on the download configuration used for installation, there may be a number of objects in Blender's central 3D window—typically a cube, a camera, and a lamp. If so, they should be removed prior to starting an exercise. To do this use the mouse to move the pointer into the central 3D window and hit the "a" key to select all objects in the window. The selected objects will be outlined in orange. (Pressing the "a" key again would toggle the selection, and all objects would have a black outline). Next, hit the "x" key, which will bring up a dialog box containing a *Delete* button that will enable you to delete the selected objects. Selec *Save Startup File* from the *File* menu and click the highlighted text to confirm. This will ensure that Blender initiates with an empty window that is best suited for doing the exercises below.

1.3.2 A brief note about units

The following story illustrates the pitfalls of not handling units correctly in a model:

Designed to orbit Mars as the first interplanetary weather satellite, the Mars Orbiter was lost in 1999 because the NASA team used metric units while a contractor used imperial. The $125 million probe came too close to Mars as it tried to maneuver into orbit, and is thought to have been destroyed by

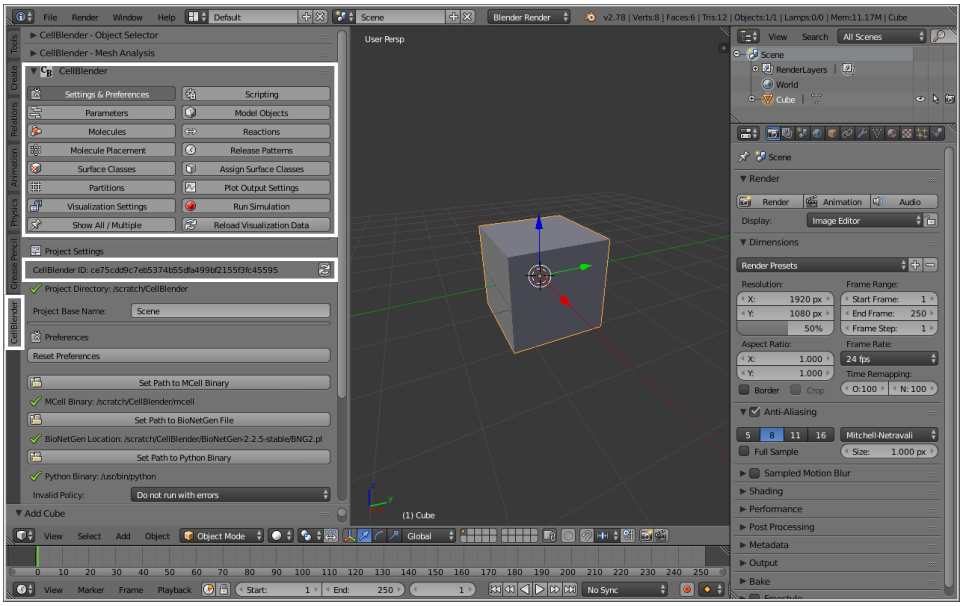

Figure 1.4

The CellBlender interface. Selection of the CellBlender tab in the Blender window (far left), brings up the CellBlender panel which consists of a set of buttons used to define the model. Selecting the *Settings & Preferences* button displays the *CellBlender ID*, which is useful for to report in publications for replicability and for reporting issues.

the planet's atmosphere. An investigation said the root cause of the loss was the failed translation of English units into metric units in a piece of ground software [29].

So be warned! Spatial dimensions in MCell are in microns ($1\mu\text{m} = 10^{-6}\text{m}$). Time is in seconds (s). Diffusion coefficients are specified in units of cm^2/s. Molecule amounts can be specified as numbers of molecules or as concentrations in molar (M) units for Volume Molecules and μm^{-2} for Surface Molecules. Unimolecular rate constants are given in $1/\text{s}$. Bimolecular reaction rate constants between two Volume Molecules or a Volume and Surface Molecule are in $\text{M}^{-1}\text{s}^{-1}$, while bimolecular reaction rate constants between two Surface Molecules are in $\mu\text{m}^2\text{s}^{-1}$. Each simulation runs for a specified number of iterations, and the duration of an iteration is one time step.

1.4 Simulating free molecular diffusion

Diffusing particles are a fundamental component of any MCell model. In this section, we consider the case of a single type of molecule freely diffusing in space.

1.4.1 CellBlender preliminaries

To set up this model we will need to make use of the following elements of the CellBlender interface (Figure 1.4).:

Molecules. Figure 1.5(A) shows the *Define Molecules* panel. As in other CellBlender tabs, the plus sign to the right of the *Define Molecules* panel is used to add a new element, in this case a Molecule. Properties of the molecule are specified by entering its Name, Type (Volume or Surface), and Diffusion Constant. The minus sign can be used to delete a previously defined molecule. The *Display Options* control the molecule's shape, size, color, and brightness in the display—note that these visualization options have no effect on the simulation. The quantity "lr_bar" that appears underneath the diffusion constant entry (Figure 1.5(A)) reports the mean diffusion distance given the value of the diffusion constant and the current time step.

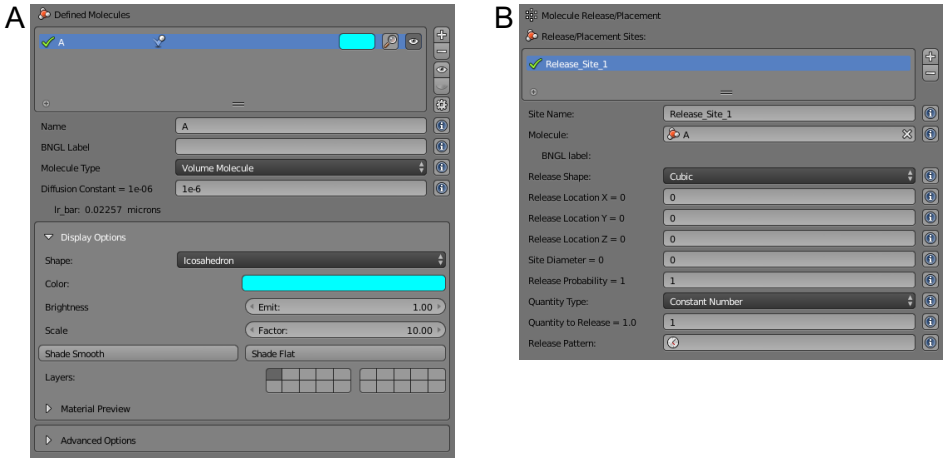

Figure 1.5

CellBlender interface tabs for defining molecules and their Release Sites. (A) The *Define Molecules* panel. (B) The *Molecule Release/Placement* panel. See text for details.

Molecule Placement. Molecules are placed into the simulation geometry by defining Release Sites using the *Molecule Release/Placement* panel (Figure 1.5(B)). Fully specifying a Release Site involves defining Site Name, Molecule (selected from the list of defined Molecules), Release Shape (selected from a list), and additional options that depend whether Object/Region or one of the three defined Release Shapes (Spherical Shell, Spherical, or Cubic) is selected. Here, the options required when the Cubic Release Shape is selected are shown. These define the position and size of the Release Shape. The other required items are Quantity Type, usually either Concentration/Density or Constant Number, and Quantity to Release, which will be specified as either a concentration (M) or

number of molecules. The Release Pattern, which controls the timing of the release either through explicit timing parameters or triggers defined by reactions, will not be discussed further in the tutorial.

Run Simulation. The *Run Simulation* panel (Figure 1.6(A)) is relatively simple. The number of iterations and the time step of each iteration are defined before launching the simulation with the *Run* button.

Reload Visualization data. The user clicks the *Reload Visualization Data* button to load and view the results of the simulation. The time line at the bottom of the Blender window (Figure 1.6(B)) provides controls to play, pause, and scroll through the simulation.

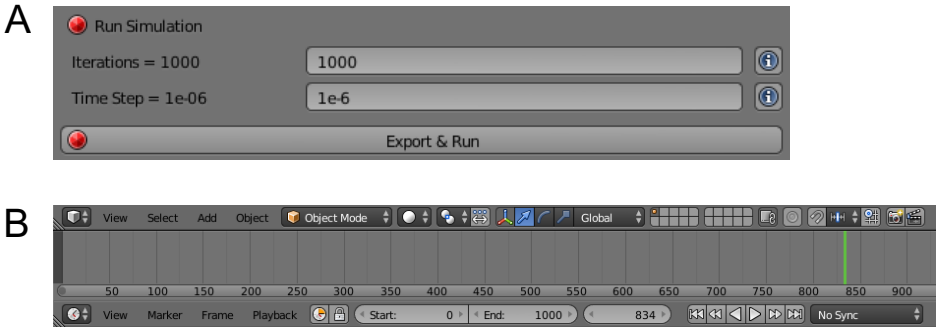

Figure 1.6

Running and visualizing simulations. (A) The *Run Simulation* panel. (B) The Blender time line, which appears at the bottom of the Blender window (see Figure 1.4).

1.4.2 Exercise: simple diffusion

Create a Volume Molecule named A with a diffusion constant $10^{-6}\,\mathrm{cm}^2/\mathrm{s}$, and release a single copy of it at the origin of a Cubic Release Site of diameter zero. Run a simulation for 1 ms (1000 iterations with a time step of 10^{-6} s), load the visualization data, and play the movie. Note that the molecule may appear very small, so you will probably need to zoom in with the mouse scroll wheel to see it. When you do, you will see two copies of your molecule. The one at the origin is a template that doesn't move.

1.5 Restricting diffusion by defining meshes

Model Objects (also called Mesh Objects) may be created using a number of Blender's built-in tools (for example, see `http://BlenderArtists.org`). CellBlender provides easy access to a subset of these tools in the *Model Objects* panel (Figure 1.7), which contains controls for centering the cursor, creating primitives (like planes, cubes,

and cylinders), renaming objects, and adding objects to the current CellBlender model. In the remaining examples we will use the *Model Objects* panel to create Model Objects that define surfaces on which Surface Molecules diffuse and that can restrict the diffusion of Volume Molecules.

1.5.1 CellBlender preliminaries

Creating Model Objects. Objects are created at the location of the 3D Cursor (Figure 1.4), which is typically a small circle of alternating red and white segments. Because the 3D Cursor can be tricky to place on the 2D screen, the *Center Cursor* button is provided to make it easier to create CellBlender model objects at the origin. To create a new Model Object, simply place the 3D Cursor at the desired location (the origin for all of the examples in this tutorial), then click one of the object buttons to the right of the *Center Cursor* button—cube, icosphere, cylinder, cone, torus, or plane—to generate a new object at the location of the 3D Cursor. After the object is created, it can be renamed using the *Active Object* field (Figure 1.8(A)). The Object's properties, such as size and number of vertices, may also be modified by changing the numbers entered in the *Add Object* panel that appears (Figure 1.8(A), bottom).

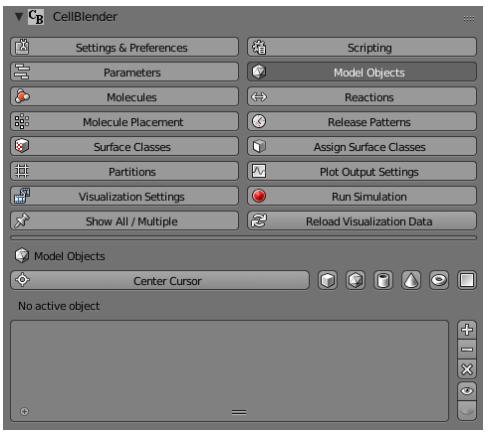

Figure 1.7
The *Model Objects* panel allows the user to create meshes with a number of different predefined geometries.

Adding Model Objects to a model. The newly created and named Object must be added to the CellBlender model using the "+" button on the right side of *Model Objects*, after which the Object appears in the Object list with a green check mark (Figure 1.8(B)). The Model Objects list also contains controls for changing an object's color and visibility.

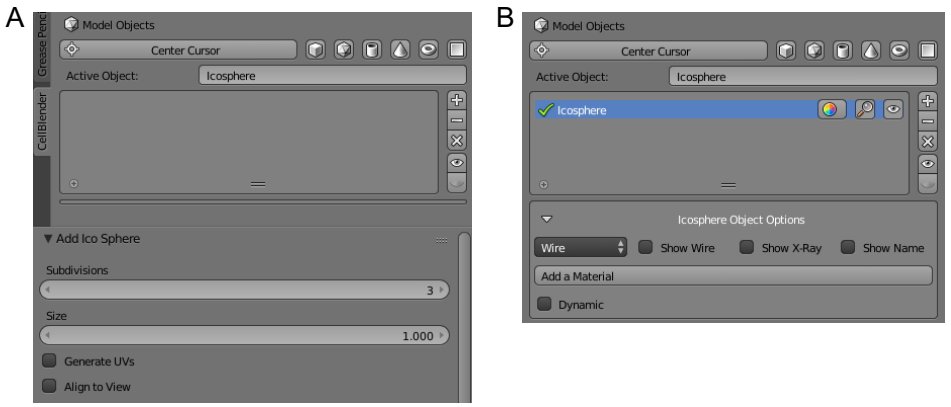

Figure 1.8

Adding an object to a model. (A) After clicking one of the object buttons in the *Model Objects* panel, the user can modify various properties of the new Object. In the example shown, the Icosphere has properties Subdivisions and Size. (B) Once the Object is added to the CellBlender model by clicking the "+" button, the user may adjust various display properties of the Object.

1.5.2 Exercise: restricted diffusion

In this exercise you will modify the model from Exercise 1.4.2 by adding an icosphere object and releasing 1000 A molecules inside that new object. Perform the following steps:

1. In the *Model Objects* panel center the cursor and then add an icosphere. The default name ("Icosphere") is fine for this exercise (Figure 1.8(A)). Change the number of subdivisions from 2 to 3, which gives a smoother icosphere.
2. With the icosphere is still selected, click the "+" button in the Model Objects panel to add the new icosphere to the list of CellBlender model objects. It should show up with the green check mark mentioned earlier.
3. Open the *Icosphere Object Options* panel directly below the list of Model Objects (Figure 1.8(B)) and change the display settings from Solid to Wire, which renders the Object transparent. As an alternative you can also add materials and give them varying degrees of transparency.
4. In the *Molecule Placement* panel change the number of molecules being released from 1 to 1000, run the simulation with the same time step and duration as before. When the simulation is finished, reload the visualization data again and play the simulation. You should see that all 1000 molecules start out at the origin and are now constrained to stay within the icosphere.

5. In order to release the 1000 molecules uniformly within the icosphere instead of at a single point, go back to the *Molecule Placement* panel and change the *Release Shape* from *Cubic* to *Object/Region*. Then type the name of the object ("Icosphere") in the *Object/Region* field below the *Release Shape* selector. If the entry corresponds to a valid Object name, a green check mark will appear in the *Release/Placement Sites* list. Run the simulation again and watch the resulting animation. You should see that the molecules are released uniformly throughout the Icosphere at the start of the simulation.

1.6 Simulating bimolecular reactions in a volume

In this section we show how to include reactions between molecules. We will first consider a simple bimolecular reaction confined within a spherical volume and then describe a more complex system—the Lotka-Volterra model of predator-prey interactions.

1.6.1 CellBlender preliminaries

This example will make use of the following CellBlender panels:

Reactions. Figure 1.9(A) shows the *Reactions* panel. A Reaction is defined by specifying the reactants separated by plus signs, the type of reaction (irreversible or reversible), the products separated by plus signs, and the rate constants governing the reaction, which may be entered directly as numbers or as expressions that refer to variables defined in the *Parameters* panel. Valid names for reactants are the names of the defined Molecules. CellBlender will report an error if an undefined name is used. In addition, the NULL keyword can be used to indicate the absence of products in a degradation reaction.

Parameters. Figure 1.9(B) shows the *Parameters* panel. Parameters must have a name and a value, and may further be given Units and a Description. Note that use of Parameters is optional—numerical values may be used instead of named parameters wherever a number is required.

Plot Output Settings. Figure 1.9(C) shows the *Plot Output Settings* panel, which allows the user to define quantities that will be tracked and output to file during a simulation. Both reaction firings and molecule counts can be tracked, and the items included can span the entire model ("World") or be restricted to a specific Object or Region of an Object. Clicking on one of the Plotter buttons following a simulation will bring up an interactive plot of the specified observables.

1.6.2 Exercise: simulating bimolecular degradation

Define two kinds of Volume Molecules A and B, each with a diffusion constant $10^{-6}\,\mathrm{cm}^2/\mathrm{s}$. Release 100 copies of each into an icosphere, and simulate the bimolecular degradation reaction A+B$\rightarrow$NULL with a rate constant of $10^8\,\mathrm{M}^{-1}\mathrm{s}^{-1}$. Define output

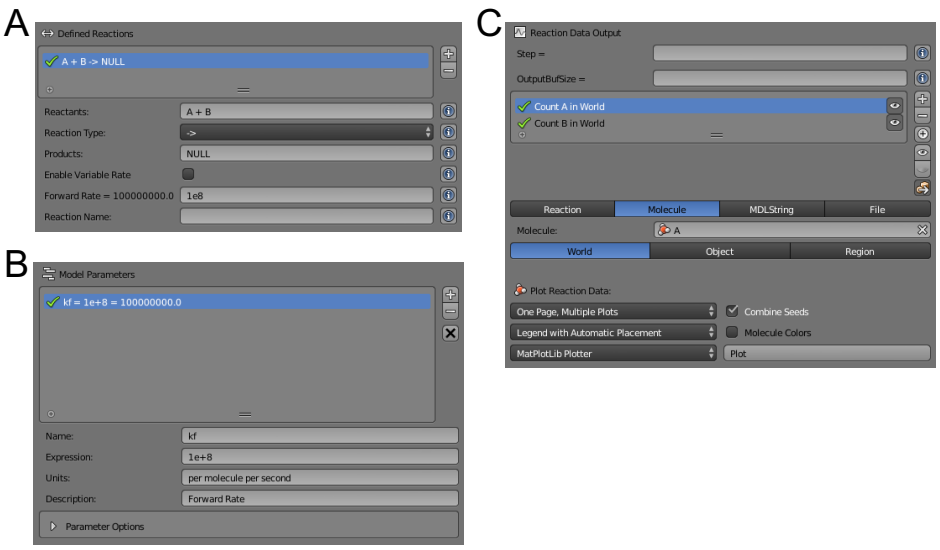

Figure 1.9

Defining Reactions, Parameters, and Plot Output data. (A) *Reactions* panel. (B) *Parameters* panel. (C) *Output Settings* panel. If the variable Step near the top is defined, output will occur at this interval in number of steps; otherwise, output occurs at every time step.

observables to count the number of A and B molecules. Run your simulation for 10^4 iterations with a time step of 10^{-5} s and plot the results, which should resemble those shown in Figure 1.10.

1.6.3 Exercise: The Lotka-Volterra predator-prey model

The Lotka-Volterra equations describe the interactions of two species, one of which preys upon the other. Although usually analyzed in the well-mixed context, this model can exhibit strong spatial effects under some conditions, as we will see by developing an MCell model of this system.

The basic interactions can be described with the following set of reactions, where the prey are sheep (S) and predators are wolves (W):

$$S \rightarrow S + S$$
$$S + W \rightarrow W + W$$
$$W \rightarrow \text{NULL},$$

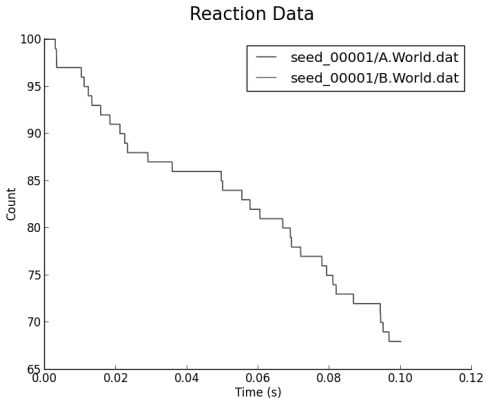

Figure 1.10

Simulation results for a bimolecular reaction plotted using *Simple Plotter*. As A and B collide and annihilate, the counts decrease. Since one molecule of A collides with one molecule of B to annihilate, the graphs for the two species are identical.

which can be summarized as: prey can spontaneously multiply, predators can eat prey to multiply, and predators spontaneously die. In our treatment, the species in the system will be confined to a thin slab, which could represent a two-dimensional forest.

1.6.3.1 Reaction-limited behavior

From the *Model Objects* panel, create a cube centered at the origin. Use Blender to deform this cube into a thin slab by adjusting the scales for the X, Y and Z axes as shown in Figure 1.11, and add the cube to your model. Define S and W as Volume Molecules with diffusion constant $6 \times 10^{-6}\,\mathrm{cm}^2/\mathrm{s}$. Release 1000 copies of each into the thin slab by defining an appropriate Release Site of type Object/Region. Enter the reactions defined above, with the rate constants $1.29 \times 10^5\,\mathrm{s}^{-1}$, $10^8\,\mathrm{M}^{-1}\,\mathrm{s}^{-1}$, and $1.30 \times 10^5\,\mathrm{s}^{-1}$ respectively. Define the *Plot Output Settings* to track the total counts of S and W and run the simulation for 500 iterations with a time step of $10^{-6}\,\mathrm{s}$. After the run completes, reload the visualization data and plot the results.

For these parameters the system is *reaction-limited*, because diffusion is fast compared with the rates of reaction, meaning that the concentrations are relatively uniform in space but not in time, which is the behavior observed in Figure 1.12(A,C). The S and W concentrations vary in time but remain spatially well-mixed. The time courses shown in Figure 1.12(C) would be very similar to trajectories obtained by solving the corresponding ODE's or using Gillespie's algorithm.

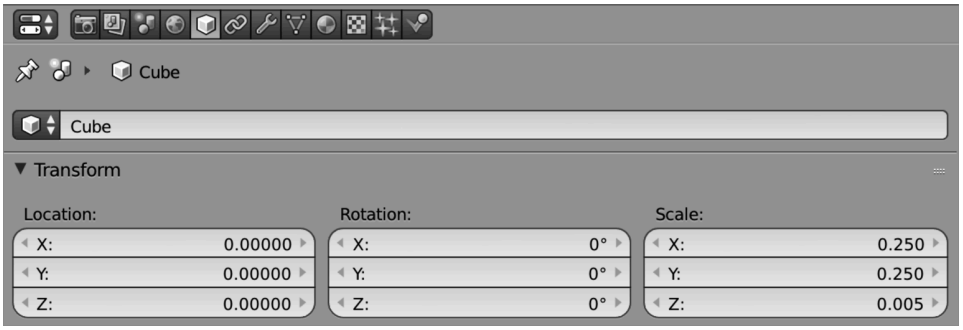

Figure 1.11

Changing Object properties in Blender. Selecting the cube icon in the bar at the top of the *Properties* panel (right hand column of main Blender window shown in Figure 1.4) brings up a list of panels with options for changing properties of the selected Object. The *Transform* panel enables translation, rotation, and scaling. Changing the X and Y scales to 0.25, and the Z scale of a cube to 0.005 creates a thin slab for the Lotka-Volterra model simulations. Set the display type to *Bounds* in *Object Options* as you did in the previous example to make the box transparent. Then use the middle mouse button (two-finger swipe on Mac) to rotate the view to view from the top of the slab, and then zoom with the right mouse button (two-finger pinch on Mac) to fill the window.

1.6.3.2 Diffusion-limited behavior

Now increase the reaction rates to $8.6 \times 10^6\,\mathrm{s}^{-1}$, $10^{12}\,\mathrm{M}^{-1}\,\mathrm{s}^{-1}$, and $5.0 \times 10^6\,\mathrm{s}^{-1}$ respectively. Reduce the time step to 10^{-8} s and run a simulation again for 500 iterations. The reduced time step ensures that the probabilities of unimolecular reactions stay below one, which is necessary to maintain accuracy. (You can check this by running the simulation again with the *Save Text Logs* option enabled in the *Output/Control Options* subpanel of the *Run Simulation* panel. After running the simulation, click on the *Screen Layout* button, which is next to *Help* on the menu bar at the top of the CellBlender interface (Figure 1.4), and select the *Scripting* layout. This layout allows you to see the MCell text log, which reports the reaction probabilities for each reaction with the specified time step. The output log for each simulation run is stored in a separate file with the name task_*PID*_output, where *PID* is the process ID reported in the *Run Simulation* panel. To view the log, click the notebook icon to the left of the *New* button and select the log name from the drop-down menu.) The reaction probability for the one bimolecular reaction remains above one, which is acceptable in this case because it simply means that every collision results in a reaction. This is the expected behavior in the diffusion limit.

For these reaction rate constants, the system becomes *diffusion-limited*, because diffusion becomes slow relative to the rates of reaction. Because of the strong predator-prey interaction, S molecules in the vicinity of a W molecule are rapidly converted to W, and the S and W populations tend to segregate. The overall system exhibits dramatic spatial heterogeneity with irregular oscillations (Figure 1.12(B,D)).

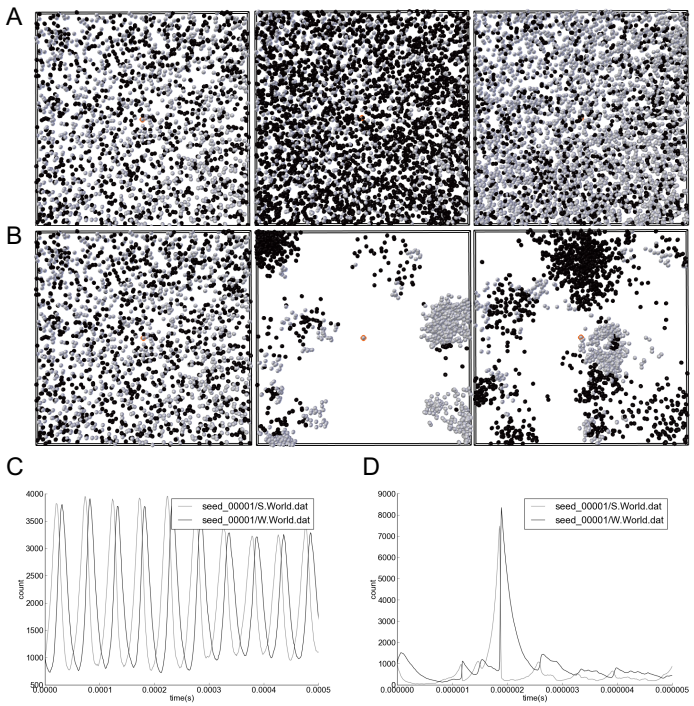

Figure 1.12

Sheep and wolf populations in the Lotka-Volterra model in the reaction-limited (A,C) and diffusion-limited (B,D) regimes. (A) Simulation snapshots along a trajectory in the reaction limited regime at 0, 140, and 320 ms going from left to right. (B) Simulation snapshots along a trajectory in the diffusion limited regime at 0, 140, and 280 ms. (C,D) Populations of S (light lines) and W (dark lines) versus time for the reaction-limited (C) and diffusion-limited (D) regimes.

1.7 Simulating molecules and reactions on surfaces

So far we have considered models in which molecules move and react in a 3D volume. Many molecules involved in cell signaling, however, are confined to membrane surfaces on and within cells. Extracellular ligands binding to cell surface receptors are a principal means by which cells sense their environment. Molecules confined to membranes have higher effective concentrations and thus membrane localization enhances catalysis and binding among surface-associated molecules, which is a critical mechanism for cell signaling [30–32]. This section provides an introduction to modeling molecules and reactions on surfaces in MCell.

1.7.1 CellBlender preliminaries

Surfaces in MCell have a distinct orientation (Front/Back) that is determined by the direction of a surface normal determined from the right-hand rule by the order of vertices in each triangle. To be a valid surface in MCell, the ordering of vertices in each triangle must give a consistent direction for the surface normal; fortunately, CellBlender takes care of this automatically. Molecules on surfaces also have an orientation (Top/Bottom) that can be aligned with or against the surface normal. The orientation of surfaces and molecules becomes important in specifying placement of Surface Molecules and in specifying reactions that involve either Surfaces or Surface Molecules. Syntactic features of reactions that involve surfaces are presented below. Here we briefly describe the elements required to specify surfaces and surface reactions:

Surface Regions. Surface Regions are composed of a subset of the Faces making up an Object. They are used to control the placement of Surface Molecules and to allow variation in surface properties in different regions—in particular how Surface Molecules move and how Volume Molecules interact with surfaces. Figure 1.13 shows the steps involved in creating a Surface Region for an existing CellBlender Model Object. First, with the Object selected the user enters Blender's *Edit Mode* by hitting the Tab key with the cursor in the main window or by selecting *Edit Mode* in the menu bar near the bottom of the Blender window (Figure 1.13(A)). Clicking the *FaceSelect* button to the right on the same bar will enable *Face Select Mode* (Figure 1.13(B)). Now, the user can select a set of Faces by right-clicking on the first Face and then holding Shift while right clicking to select additional Faces (Figure 1.13(C)). Figure 1.14 presents an alternate way to select Faces using the *Circle Select Tool*, which is more efficient for large Objects. The *Defined Surface Regions* subpanel, which is part of the *Model Objects* panel, is then used to create a named Surface Region containing the selected Faces (Figure 1.13(D)). The *Select* and *Deselect* buttons can be used to toggle the selected Faces and to determine which Faces are assigned to a Surface Region. Note that it is allowable for Surface Regions to overlap.

Surface Classes. Surface Classes define how a Molecule behaves when it encounters a surface, e.g., whether it reflects, passes through, or is absorbed. They allow for development of realistic models because biological membranes may have different properties with respect to different molecules. For example, some molecules may freely diffuse across a membrane, whereas others may not. It is also useful to define "transparent" membranes for the purpose of counting particular types of molecules in particular regions of space. Figure 1.15(A) shows the CellBlender interface for defining a Surface Class, which requires a name and one or more Properties and which can be of various types including *Transparent*, *Reflective*, and *Absorptive*. Each Property applies to only one type of Molecule (and may further depend on Orientation), and many Properties can be defined for a single Surface Class. For any Molecule–Surface interaction that is not governed by a defined Surface

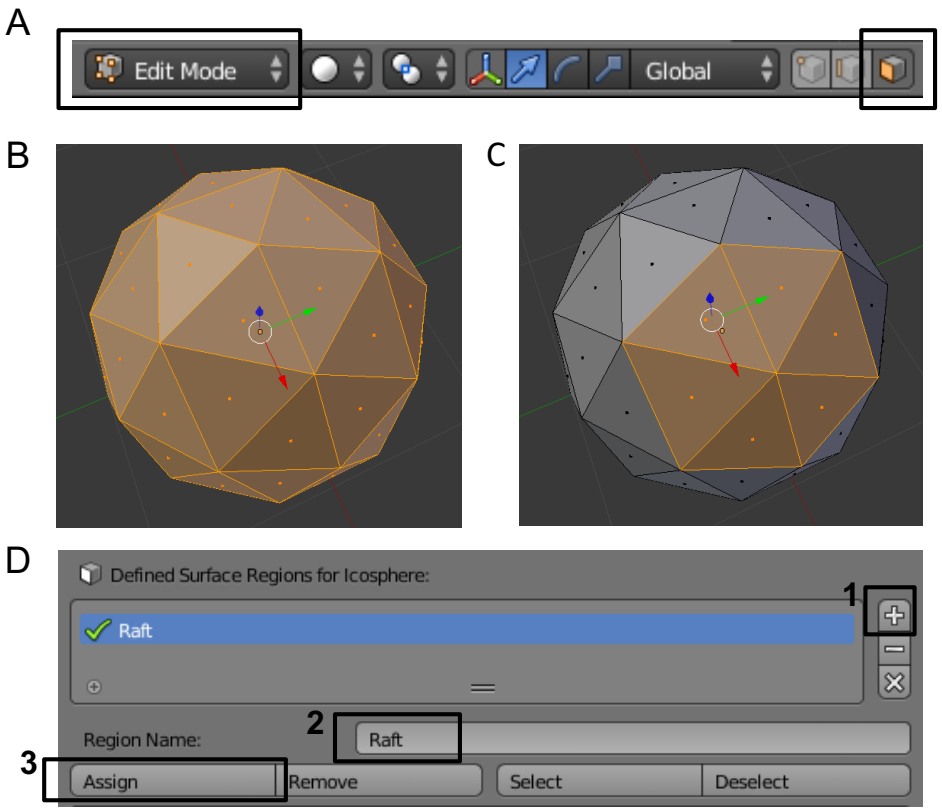

Figure 1.13

Defining Surface Regions. (A) The menu bar near the bottom of the Blender interface (Figure 1.4) with options to enter *Edit Mode* (left) and use *Face Select* (right). (B) Triangulated Object with all Faces selected. (C) Triangulated Object with six Faces selected. (D) *Defined Surface Regions* subpanel at the bottom of the *Model Objects* panel showing the steps required to define a Surface Region from a set of selected Faces.

Class, a default behavior is assumed: *Reflective* for Volume molecules and *Transparent* for Surface Molecules.

Assigning Surface Classes. Surface Classes are assigned to Surface Regions in the *Assign Surface Classes* panel (Figure 1.15(B)). Note that to apply a Surface Class to a whole Object, a Surface Region including all Faces in the Object must be defined.

Defining surface reactions. Surface reactions in MCell may be defined with reference to either the relative or absolute orientation of the reacting molecules, as introduced in [11]. For example, to specify a reaction where a Surface Molecule B flips its orientation,

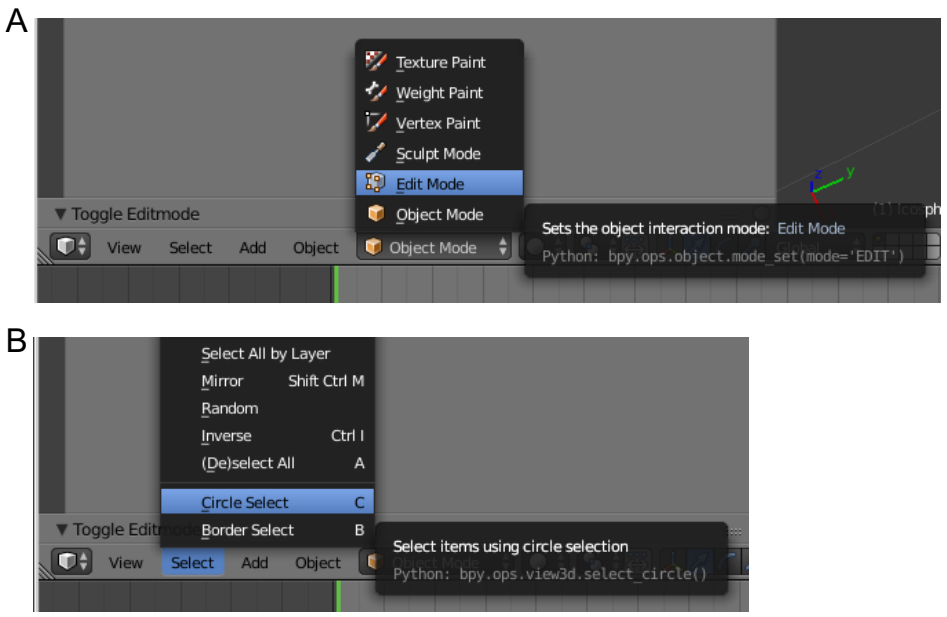

Figure 1.14

Selecting Faces using *Circle Select*. (A) Object interaction model menu. *Edit Mode* is used to select Faces of an object to define Surface Regions. (B) The *Circle Select* option can be used to select Faces on a triangulated Object. Faces are selected by holding the left mouse button while dragging the cursor over the desired Faces.

we write

$$B' \rightarrow B,$$

where the apostrophe and comma represent the relative orientation of B on the reactant and product side respectively. Note that whether the tick mark is up or down on the reactant side does not matter; what matters is that the position changes. MCell uses a shorthand notation where the number of ticks indicates the orientation class of a surface object, while the position (up or down) indicates the relative orientation. Thus, the reaction

$$B, \rightarrow B'$$

is equivalent to the previous one: it will flip the relative orientation of any B molecule on a surface regardless of its absolute orientation on that surface. Any reaction that involves a Surface Molecule must specify the relative orientations of all reactants and products. In this example

$$B' \rightarrow B' + A' + C,$$

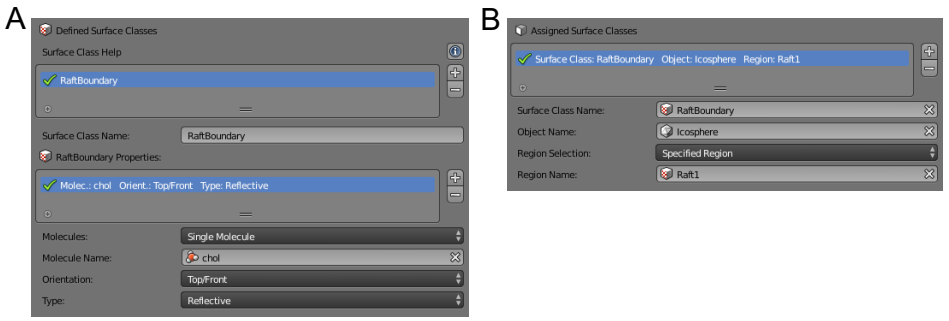

Figure 1.15

Defining and Assigning Surface Classes. (A) The *Surface Classes* panel. Each class (top list) may have multiple properties (bottom list). (B) The *Assign Surface Classes* panel. Both Object Name and Region Name are required.

B retains its orientation while A and C, if they are Surface Molecules, are generated with the same and opposite orientations respectively because they are in the same orientation class as the reactant B. If instead A and C are Volume Molecules, then the orientation class indicates the side of the surface the products will be generated on. Here, A would be generated on the side where B is pointing and C would be generated on the opposite side. If all of the B molecules have been given the same orientation, then A and C will always be produced on opposite sides of the surface. The absolute polarity of this model would be maintained as long as there are no reactions that flip the orientation of B.

It is sometimes useful to specify absolute orientation in reactions, which requires adding a Surface Class to the specification of reactants. For example, consider the following reaction where L is a Volume Molecule and R and LR are Surface Molecules:

$$L' + R'@surf' \rightarrow LR'$$

Here the fact that the Surface Molecule R and the Surface Class **surf** are in the same orientation class requires that R have a fixed absolute orientation (Top-Front) with respect to **surf**. In addition, L is required to be in a volume adjacent to the Front of **surf**, and LR will be produced with the same orientation as R. For a more details about surface reactions, see `http://mcell.org/documentation`.

1.7.2 Exercise: a simple model of receptors aggregation in lipid rafts

We will create a simple model of receptor clustering in lipid rafts. Fast-diffusing receptor molecules (Rf) react with cholesterol molecules (chol) in rafts to become slow-diffusing receptors (Rs). The model can be built in the following steps:

1. Create an icosphere to represent a cell and add it to your model. Increase the number of subdivisions to 3 to make the sphere smoother.

2. Create two non-touching Surface Regions named Raft1 and Raft2 comprised of 6 triangles forming a hexagon for Raft1 and one triangle for Raft2 following the procedures shown in Figures 1.13 and 1.14.
3. Define Surface Molecules Rf, Rs, and chol with diffusion constants $10^{-6}\,\mathrm{cm}^2/\mathrm{s}$, $10^{-9}\,\mathrm{cm}^2/\mathrm{s}$, and $10^{-6}\,\mathrm{cm}^2/\mathrm{s}$ respectively.
4. Create a Surface Class that is reflective with respect to chol (Figure 1.15) and assign it to both Surface Regions (Figure 1.16).
5. Release 250 chol molecules in Raft1 and 100 chol molecules in Raft2 as shown in Figure 1.16(A) and 1000 Rf molecules distributed over the whole surface as shown in Figure 1.16(B).
6. Define the surface reaction

   $$\mathrm{Rf}' + \mathrm{chol}' \rightarrow \mathrm{Rs}' + \mathrm{chol}'$$

   and assign it a rate constant of $10^8\,\mu\mathrm{m}^2/\mathrm{s}$.
7. Run your simulation for 1000 iterations at a time step of $10^{-5}\mathrm{s}$. If you change the colors in *Display Options* of the Molecules panel so that Rs and Rf are the same color, your results should resemble those shown in Figure 1.17(A,B). Because there is no reaction to convert Rs back to Rf, Rs molecules accumulate at the raft boundaries, particularly Raft1 (Figure 1.17(B)). Eventually all receptors will end up as Rs.
8. To make the model more realistic, modify the previous reaction scheme to include a reverse reaction:

   $$\mathrm{Rs}' \rightarrow \mathrm{Rf}'$$

   and give it a rate constant $10^4\,\mathrm{s}^{-1}$. Simulating the extended model should give results similar to those shown in Figure 1.17(C). There is still pronounced clustering in the rafts, but the receptors are more uniformly distributed throughout each raft.

1.8 Extended exercise: a density-dependent switch

We now consider a second mechanism for receptor clustering based on positive feedback [33]. In this model receptors can spontaneously become activated at a slow rate, but upon activation can activate additional receptors at a faster rate, which represents a form of positive feedback. The strength of the feedback depends on the receptor density, which in turn affects the spatial heterogeneity, as we will observe. Since all of the CellBlender operations needed to specify this model were presented in the previous sections, we can jump into a description of the model.

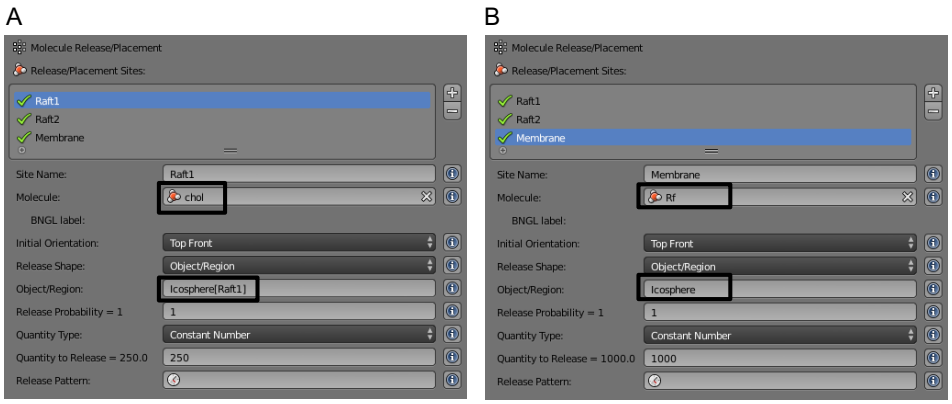

Figure 1.16

Releasing Molecules on Surface Regions. (A) The Surface Region is specified with the syntax ObjectName[RegionName] in the *Object/Region field*. (B) Specifying only the Object name causes Molecule release over the full surface.

Create Surface Molecules R and Ra, which represent the inactive and active forms of the receptor, and set the diffusion constants to be $10^{-7}\,\mathrm{cm}^2\,\mathrm{s}^{-1}$ and $10^{-10}\,\mathrm{cm}^2\,\mathrm{s}^{-1}$ respectively. The reaction scheme is

$$\mathrm{R}' \rightarrow \mathrm{Ra}'$$
$$\mathrm{Ra}' + \mathrm{R}' \rightarrow \mathrm{Ra}' + \mathrm{Ra}'$$
$$\mathrm{Ra}' \rightarrow \mathrm{R}'$$

The rate constants for the three reactions are $10^{-5}\,\mathrm{s}^{-1}$, $40\,\mu\mathrm{m}^2\,\mathrm{s}^{-1}$, and $40\,\mathrm{s}^{-1}$ respectively.

High density. Use the *Model Objects* panel to create a Plane Object (shape furthest to the right) with surface area $1\,\mu\mathrm{m}^2$ by setting Radius to 0.5 in the *Add Plane* subpanel. Add this Object to your model. Release 5 molecules of Ra and 395 molecules of R on to the plane. Define the reaction scheme and parameters as mentioned above, and run your simulation for $5,000$ iterations with a time step of 10^{-5} s. Play your simulation and compare with the results shown in Figure 1.18(A). At this density the active state is stabilized with respect to the inactive state and nearly all of the receptors end up as active. Because the active receptors diffuse slowly on the time scale of this simulation, clusters arise around the initially active receptors, and then these clusters merge as the proportion of active receptors grows further.

Medium density. Increase the area of the plane from $1\,\mu\mathrm{m}^2$ to $16\,\mu\mathrm{m}^2$ using the *Transform* panel as described above (Figure 1.11). For how many iterations do you need to

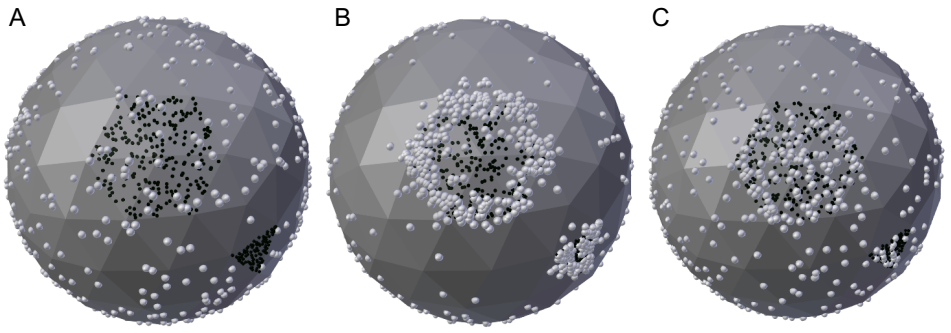

Figure 1.17

Simulation results for the model of receptor aggregation in lipid rafts. (A) Initial configuration with chol (black circles) confined to rafts and uniform distribution of receptors (white spheres). (B) Snapshot at 1 μs showing receptor clustering in the two raft regions. (C) Snapshot at 1 μs shown reduced clustering due to the addition of a reaction that converts Rs back to Rf.

simulate now in order to approach the steady state for number of active receptors? Compare your results to those shown in Figure 1.18(B). At this lower density, clusters persist and remain distinct unless the seeding Ra molecules were initially close.

Low density. Increase the surface area again to $1600\,\mu\text{m}^2$ and rerun your simulation ($10,000$ iterations should suffice). At this density the rate of receptor-catalyzed activation is too low to maintain receptor activity, and all of the Ra molecules eventually convert to R molecules (Figure 1.18(C)).

Changing the area to vary the system density shows that the system can exhibit at least three distinct behaviors going from high to low density (Figure 1.18): global clustering that links distinct sites of receptor activation, local clustering, in which each site of spontaneous activation nucleates a distinct cluster, and extinction, in which activation of a receptor results in little or no activation of nearby receptors. Such effects may be important for signal propagation and would be difficult to capture using a modeling framework that does not capture both spatial and stochastic effects.

1.9 Concluding remarks

Models play a valuable role in biology by both aiding our understanding of old observations and predicting new ones. As discussed in the Introduction, modeling formalisms vary in their molecular and spatial resolution, and understanding the limitations of each will help a modeler choose an appropriate method for a given problem. This chapter has introduced the theory and algorithms that can be used to carry out one form of spatial stochastic modeling as implemented in the MCell simulation package. It has also provided a tutorial

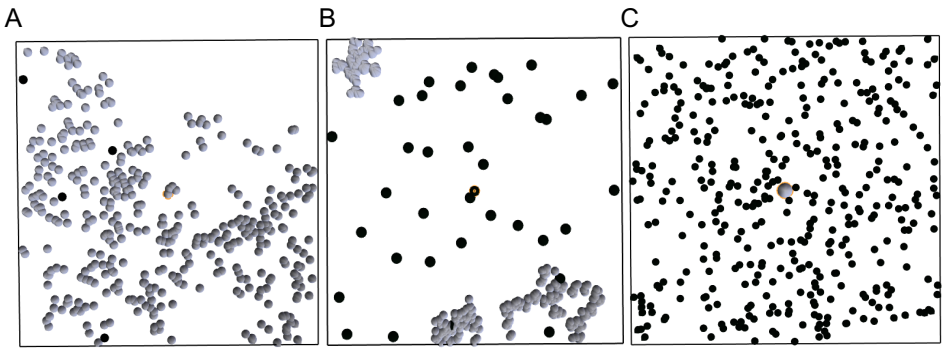

Figure 1.18

Simulation snapshots after 1 s simulation for the density dependent switch at (A) high density, (B) medium density, and (C) low density showing active (gray spheres) and inactive receptors (black circles). Model parameters are given in the text.

for building, simulating, and analyzing MCell models using the CellBlender interface starting from basic diffusion and reaction-diffusion systems, and building up to more advanced examples that demonstrate how spatial heterogeneity can arise from a variety of mechanisms including, diffusion-limited reactions (Lotka-Volterra), membrane organization and hindered diffusion (lipid rafts), and positive feedback.

For more information about MCell and CellBlender visit `http://www.mcell.org/`.

Acknowledgment

We would like to Markus Dittrich for his many contributions to MCell and CellBlender and the participants of the 2016 Cell Modeling Workshop and the 2016 q-bio Summer School (University of New Mexico campus) for their helpful input on the examples. This work was supported in part by the NIGMS-funded (P41-GM103712) National Center for Multiscale Modeling of Biological Systems (MMBioS) and NIH grant R35-GM119462.

Bibliography

[1] B. Goldstein, J. R. Faeder, W. S. Hlavacek. 'Mathematical and computational models of immune-receptor signalling.' *Nat. Rev. Immunol.*, **4**(6):445–56, 2004.

[2] B. N. Kholodenko. 'Cell-signalling dynamics in time and space.' *Nat. Rev. Mol. Cell Biol.*, **7**(3):165–76, 2006.

[3] R. Grima, S. Schnell. 'Modelling reaction kinetics inside cells.' *Essays Biochem.*, **45**:41–56, 2008.

[4] J. J. Tyson, K. C. Chen, B. Novak. 'Sniffers , buzzers , toggles and blinkers : dynamics of regulatory and signaling pathways in the cell'. *Curr. Opin. Cell Biol.*, **15**(2):221–231, 2003.

[5] B. B. Aldridge, J. M. Burke, D. A. Lauffenburger, P. K. Sorger. 'Physicochemical modelling of cell signalling pathways.' *Nat. Cell Biol.*, **8**(11):1195–1203, 2006.

[6] D. T. Gillespie. 'Exact stochastic simulation of coupled chemical reactions'. *J. Phys. Chem.*, **81**(25):2340–2361, 1977.

[7] M. W. Sneddon, J. R. Faeder, T. Emonet. 'Efficient modeling, simulation and coarse-graining of biological complexity with NFsim.' *Nat. Methods*, **8**(2):177–183, 2011.

[8] L. M. Loew, J. C. Schaff, L. M. Loew, J. C. Schaff. 'The Virtual Cell : a software environment for computational cell biology'. *TRENDS Biotechnol.*, **19**(10):401–406, 2001.

[9] J. Elf, M. Ehrenberg. 'Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases.' *Syst. Biol. (Stevenage).*, **1**(2):230–236, 2004.

[10] B. Drawert, S. Engblom, A. Hellander. 'URDME: a modular framework for stochastic simulation of reaction-transport processes in complex geometries.' *BMC Syst. Biol.*, **6**(1):76, 2012.

[11] R. A. Kerr, T. M. Bartol, B. Kaminsky, M. Dittrich, J.-C. J. Chang, S. B. Baden, T. J. Sejnowski, J. R. Stiles. 'Fast Monte Carlo Simulation Methods for Biological Reaction-Diffusion Systems in Solution and on Surfaces'. *SIAM J. Sci. Comput.*, **30**(6):3126–3149, 2008.

[12] S. S. Andrews, D. Bray. 'Stochastic simulation of chemical reactions with spatial resolution and single molecule detail.' *Phys. Biol.*, **1**(3-4):137–51, 2004.

[13] M. Ullah, O. Wolkenhauer. 'Stochastic approaches in systems biology'. *Wiley Interdiscip. Rev. Syst. Biol. Med.*, **2**(4):385–397, 2010.

[14] L. A. Chylek, L. A. Harris, C.-S. Tung, J. R. Faeder, C. F. Lopez, W. S. Hlavacek. 'Rule-based modeling: a computational approach for studying biomolecular site dynamics in cell signaling systems.' *Wiley Interdiscip. Rev. Syst. Biol. Med.*, **6**(1):13–36, 2013.

[15] K. Lipkow, D. J. Odde. 'Model for Protein Concentration Gradients in the Cytoplasm'. *Cell Mol Bioeng.*, **1**(1):84–92, 2008.

[16] M. J. Tindall, P. K. Maini, S. L. Porter, J. P. Armitage. 'Overview of Mathematical Approaches Used to Model Bacterial Chemotaxis II : Bacterial Populations'. *Bull. Math. Biol.*, **70**:1570–1607, 2008.

[17] S. Luckhaus, W. Jager. 'On explosions of solutions to a system of partial differential equations modelling chemotaxis'. *Trans. Am. Math. Soc.*, **329**(2):819–824, 1992.

[18] M. Howard, A. D. Rutenberg. 'Pattern formation inside bacteria: fluctuations due to the low copy number of proteins.' *Phys. Rev. Lett.*, **90**(12):128102, 2003.

[19] D. Fange, J. Elf. 'Noise-Induced Min Phenotypes in E . coli'. *PLOS Comput. Biol.*, **2**(6):637–648, 2006.

[20] D. T. Gillespie, L. R. Petzold, E. Seitaridou. 'Validity conditions for stochastic chemical kinetics in diffusion-limited systems'. *J. Chem. Phys.*, **140**(5), 2014.

[21] D. T. Gillespie, A. Hellander, L. R. Petzold. 'Perspective: Stochastic algorithms for chemical kinetics'. *J. Chem. Phys.*, **138**(17):170901–144908, 2013.

[22] J. Hattne, D. Fange, J. Elf. 'Stochastic reaction-diffusion simulation with MesoRD.' *Bioinformatics*, **21**(12):2923–2924, 2005.

[23] T. Bartol, M. Dittrich, J. Faeder. 'MCell'. In D. Jaeger, R. Jung, editors, *Encycl. Comput. Neurosci.*, pages 1–5. Springer New York, 2014.

[24] S. S. Andrews, N. J. Addy, R. Brent, A. P. Arkin. 'Detailed simulations of cell biology with Smoldyn 2.1.' *PLoS Comput. Biol.*, **6**(3):e1000705, 2010.

[25] J. Schöneberg, A. Ullrich, F. Noé. 'Simulation tools for particle-based reaction-diffusion dynamics in continuous space.' *BMC Biophys.*, **7**(1):11, 2014.

[26] T. M. Bartol, B. R. Land, E. E. Salpeter, M. M. Salpeter. 'Monte Carlo simulation of miniature endplate current generation in the vertebrate neuromuscular junction.' *Biophys. J.*, **59**(6):1290–307, 1991. ISSN 0006-3495.

[27] J. R. Stiles, D. Van Helden, T. M. Bartol, E. E. Salpeter, M. M. Salpeter. 'Miniature endplate current rise times less than 100 microseconds from improved dual recordings can be modeled with passive acetylcholine diffusion from a synaptic vesicle.' *Proc. Natl. Acad. Sci. U. S. A.*, **93**(12):5747–52, 1996. ISSN 0027-8424.

[28] J. R. Stiles, T. M. Bartol. 'Monte Carlo methods for simulating realistic synaptic microphysiology using MCell. In: Computational Neuroscience: Realistic Modeling for Experimentalists, ed. De Schutter, E'. *CRC Press. Boca Rat.*, pages 87–127, 2001.

[29] H. Cheung, S. Mulvey. 'Great miscalculations: The French railway error and 10 others.' *BBC News Magazine*, 22 May 2014.

[30] J. M. Haugh, D. A. Lauffenburger. 'Physical modulation of intracellular signaling processes by locational regulation.' *Biophys. J.*, **72**(5):2014–31, 1997. ISSN 0006-3495.

[31] B. N. Kholodenko, J. B. Hoek, H. V. Westerhoff. 'Why cytoplasmic signalling proteins should be recruited to cell membranes.' *Trends Cell Biol.*, **10**(5):173–8, 2000. ISSN 0962-8924.

[32] J. M. Haugh. 'A unified model for signal transduction reactions in cellular membranes.' *Biophys. J.*, **82**(2):591–604, 2002. ISSN 0006-3495.

[33] A. Jilkine, S. B. Angenent, L. F. Wu, S. J. Altschuler. 'A density-dependent switch drives stochastic clustering and polarization of signaling molecules.' *PLoS Comput. Biol.*, **7**(11):e1002271, 2011.