# SpringSaLaD Version 2 User's Guide and Tutorial

Paul Michalski
Email: pjmichal@gmail.com
Joseph Masison
Email: masison@uchc.edu

## Table of Contents

# INTRODUCTION

SpringSaLaD is a particle-based, stochastic, biochemical simulation platform. It is most suitable for modeling mesoscopic systems, which are far too large to be modeled with molecular dynamics but which require more detail than obtainable with macroscopic continuum models. A good example of such a system is ligand-mediated receptor clustering, where coarse-grained particle-level interactions are required but atom-level information is not of interest. In general, SpringSaLaD is best used to model mesoscopic particles (diameters of 1 – 10 nm, which could represent a functional protein domain), moderately sized systems (10 – 10,000 particles), and times scales on the order of 1 second, but no actual restriction is set on size or scale of the simulated system.

Here we will give a brief introduction to the modeling concepts underlying SpringSaLaD. The tutorial which follows explains all of the features of SpringSaLaD by showing the steps required to build a toy model of receptor kinase activation.

# MODELING FRAMEWORK

## Molecules

SpringSaLaD models molecules as a collection of spherical sites linked by stiff, unbreakable springs. In a loose sense the sites are intended to represent mesoscopic protein domains, such as SH2 domains or catalytic domains. Each molecule is assumed to contain various site types, and each site in the molecule is assigned one of these types. Parameters which define the site's physical and biochemical properties are assigned to the type, and then each site is assigned to a defined type. Each site type is associated with a physical radius which enforces excluded volume, a diffusion constant, a color (for display purposes only), and one or more internal states. These internal states are used to represent different biochemical states of the same site, for example, an enzymatic site which may be either active or inactive.

As an example to illustrate these concepts, consider protein kinase A, which is composed of two regulatory domains and two catalytic domains. Each regulatory domain can bind up to two cAMP molecules, and each catalytic domain can be either active or inactive. To create PKA in SpringSaLaD, we would first define the two site types, regulatory and catalytic. We would assign three states to the regulatory type to describe the number of bound cAMP molecules (0, 1 or 2), and would assign two states to the catalytic type to indicate an active or inactive site. We would then create four sites, assigning two sites to be regulatory sites and two sites to be catalytic sites. Lastly, we would create links (springs) between each regulatory/catalytic pair, and between the regulatory sites. Figure 1 shows a depiction of this model representation of PKA.

| Molecule: PKA | | |
|---|---|---|
| Site Types | Sites | Links |
| 1) Regulatory<br>a) State 1: cAMP_0<br>b) State 2: cAMP_1<br>c) State 3: cAMP_2<br>2) Catalytic<br>a) State 1: Inactive<br>b) State 2: Active | 1) Regulatory<br>2) Regulatory<br>3) Catalytic<br>4) Catalytic | 1) Site 1 to site 2<br>2) Site 1 to site 3<br>3) Site 2 to site 4 |

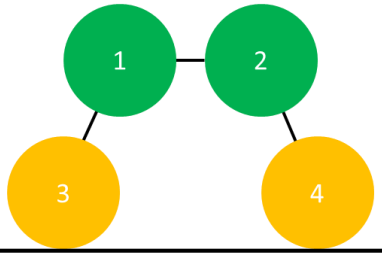



Figure 1: Model representation of protein kinase A in SpringSaLaD. The table at top shows the various site types, sites, and links used to define the molecule, as described in the text. The figure at bottom shows the sites and their connecting links.

## Reactions

SpringSaLaD supports a variety of reactions commonly used in biochemical simulations.

### Zeroeth Order Reactions

SpringSaLaD supports zeroeth order "creation" reactions (also known as "birth" reactions) which spontaneously generate new particles. These reactions are defined by a rate constant with units of $\mu M/s$.

### First Order Reactions

SpringSaLaD supports three general categories of first order reactions, each of which is characterized by a rate constant with units of $s^{-1}$.

a) Decay Reactions
These reactions spontaneously destroy a particle. Decay reactions are often used with creation reactions in order to buffer the concentration of a particular species.

b) Dissociation Reactions
These reactions destroy the spring connecting two sites, which allows to previously bonded sites to dissociate.

c) State Transition Reactions
These reactions change the internal state at a particular site. For example, a state transition reaction would be used to convert a catalytic site from inactive to active. Transition reactions are further divided into four groups depending on the restrictions placed upon the reaction.

- Free. The reaction can only occur if the site is unbound.
- Bound. The reaction can only occur if the site is bound.

- Allosteric. The reaction can only occur if another specific site in the same molecule is in a particular state.
- None. The reaction can proceed without conditions.

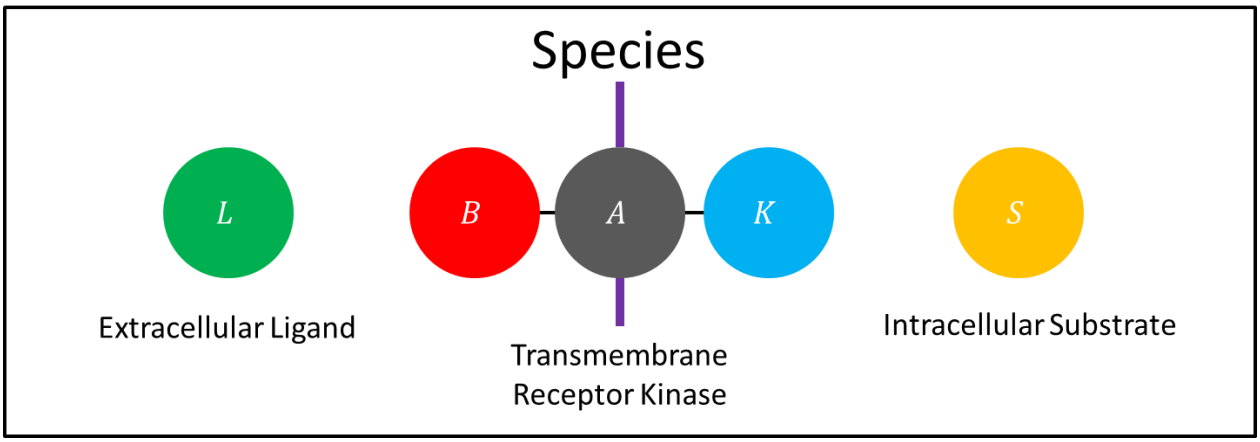
### Second Order Reactions

SpringSaLaD supports bimolecular binding reactions characterized by a rate constant with units $\mu M^{-1}s^{-1}$.

## TUTORIAL

In this tutorial we will show how to use SpringSaLaD to create a simple toy model resembling MAPK activation. Specifically, we will assume the system consists of an extra-cellular ligand, $L$, a trans-membrane receptor kinase, $T$, and an intra-cellular substrate, $S$, which is activated when phosphorylated by $T$. $L$ will consist of a single site with a single biochemical state. $T$ consists of three sites: 1) an extra-cellular ligand binding domain, $B$, 2) a trans-membrane anchor, $A$, and 3) an intra-cellular kinase domain, $K$. The binding domain has two states which represent different structural configurations in the absence or presence of bound ligand, which we simply label as states 0 (no ligand) and 1 (with ligand). The kinase domain will have two states, "off" and "on," and is activated by an allosteric reaction once ligand binds to the ligand-binding domain. $S$ consists of a single site with two possible states, phosphorylated (p) or unphosphorylated (u). The initial condition will have all $B$ domains in the unbound configuration (state 0), all $K$ domains turned off, and all $S$ unphosphorylated. The observable of interest is the number of phosphorylated $S$ molecules in the system. Figure 2 shows a schematic of the molecules and the reactions in the system, while Table 1 indicates the values of the parameters used in this tutorial.

While simple, this model allows us to show how to use most of the features in SpringSaLaD. Features which are not used in constructing this model will also be discussed in the appropriate sections.




## Reactions

Ligand binding and unbinding: $L + B_0 \overset{k_{L0,on},\ k_{L0,off}}{\longleftrightarrow} L \cdot B_0$

$L + B_1 \overset{k_{L1,on},\ k_{L1,off}}{\longleftrightarrow} L \cdot B_1$

Binding domain configuration change: $L \cdot B_0 \overset{r_0}{\rightarrow} L \cdot B_1$

$B_1 \overset{r_1}{\rightarrow} B_0$

Receptor kinase activation and inactivation: $K_{off} \overset{r_{act}}{\longrightarrow} K_{on}$ when ligand is bound

$K_{on} \overset{r_{inact}}{\longrightarrow} K_{off}$ when $K$ is free

Substrate phosphorylation: $K_{on} + S_u \overset{k_1,\ k_2}{\longleftrightarrow} K_{on} \cdot S_u$

$K_{on} \cdot S_u \overset{r_p}{\rightarrow} K_{on} \cdot S_p$

$K_{on} \cdot S_p \overset{k_3}{\rightarrow} K_{on} + S_p$

Substrate dephosphorylation: $S_p \overset{r_u}{\rightarrow} S_u$ when substrate is free

Figure 2: Schematic representation of the species and reactions for the system used in the tutorial.

| Parameter | Value |
|---|---|
| $k_{L0,on}$ | $20\ \mu M\ s^{-1}$ |
| $k_{L0,off}$ | $100\ s^{-1}$ |
| $k_{L1,on}$ | $40\ \mu M\ s^{-1}$ |
| $k_{L1,off}$ | $10\ s^{-1}$ |
| $r_0$ | $1000\ s^{-1}$ |
| $r_1$ | $1000\ s^{-1}$ |
| $r_{act}$ | $1000\ s^{-1}$ |
| $r_{inact}$ | $10\ s^{-1}$ |
| $k_1$ | $10\ \mu M\ s^{-1}$ |
| $k_2$ | $20\ s^{-1}$ |
| $k_3$ | $1000\ s^{-1}$ |
| $r_p$ | $100\ s^{-1}$ |
| $r_u$ | $10\ s^{-1}$ |

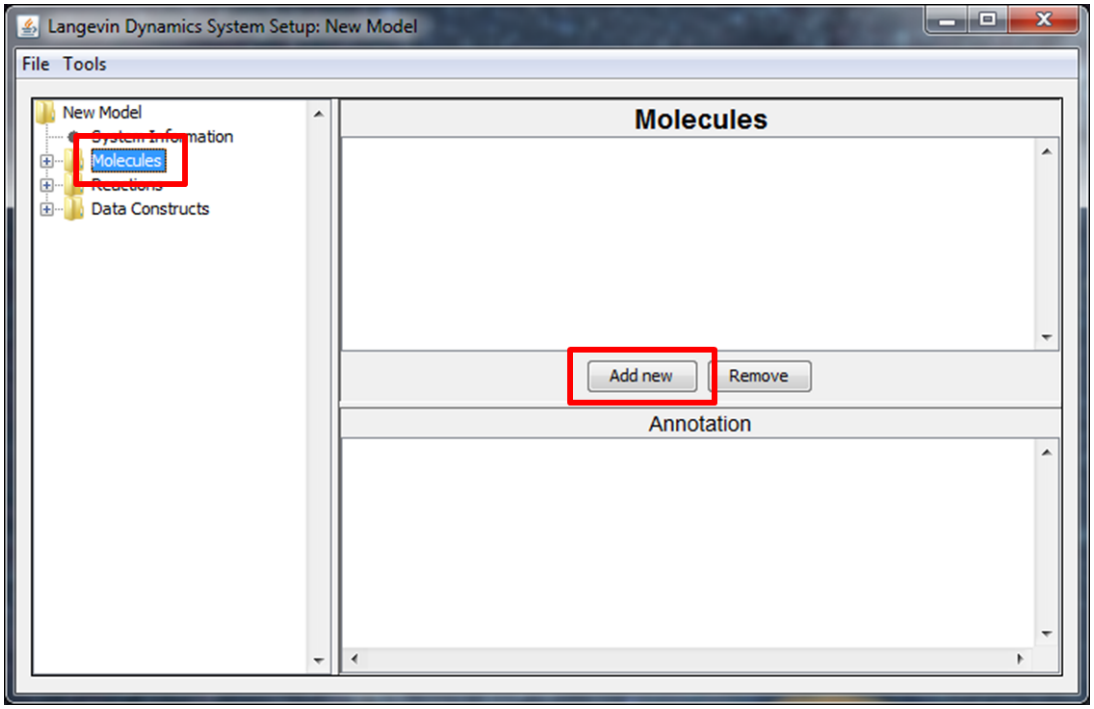
Table 1: Values of the rate constants used in the tutorial example. These values are all reasonable from a biological perspective, but are used here for illustrative purposes and are not intended as an accurate representation of any particular system.

## Running SpringSaLaD

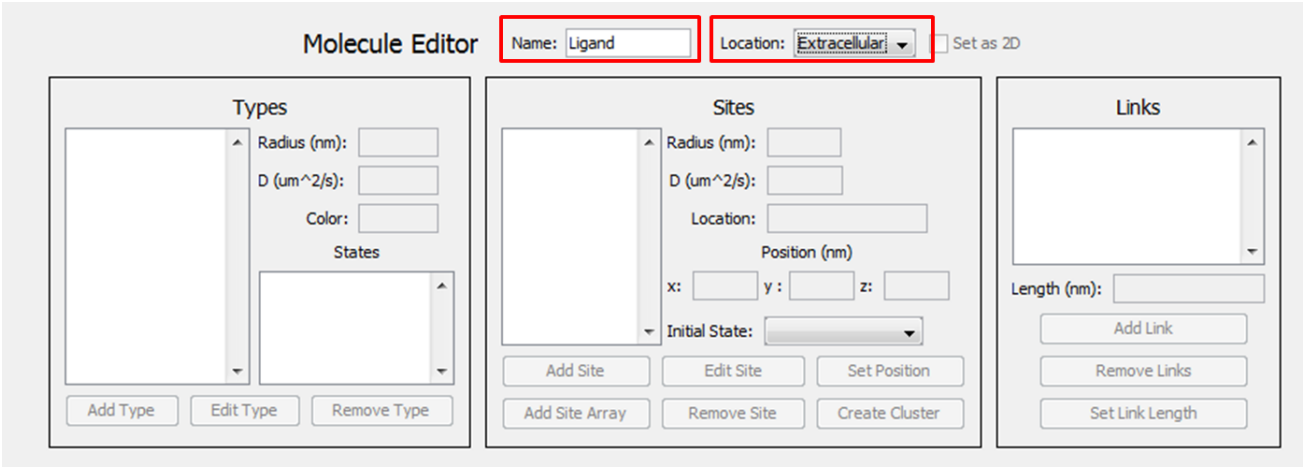
SpringSaLaD requires Java 8.0, which can be freely downloaded from https://www.java.com/en/download/ . SpringSaLaD is provided as a pair of JAR files, "SpringSalad.jar" and "LangevinNoVis01.jar". These JAR files contain all program dependencies; for example, Java3D is pre-packaged and does not require the user to perform a separate download and install. Put the two JAR files in the same folder, then double-click on SpringSalad.jar.

## Defining Molecules

Upon opening SpringSaLaD there will be a menu of options in the left-hand panel. Click on "Molecules" and then click "Add New" in the center panel.

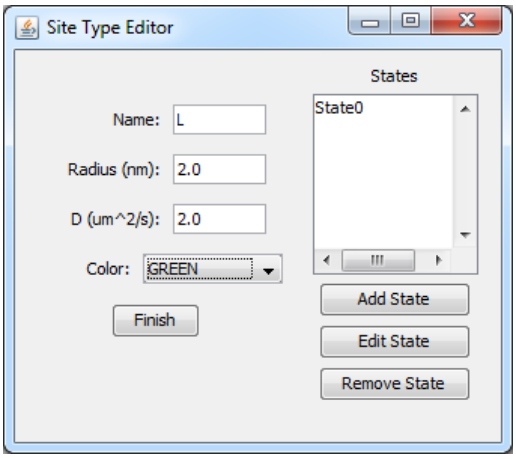



The center panel will now show the molecule editor pane. At the top are panels to define the site types, the sites, and the links. To begin we will name the molecule "Ligand" and set it to be "Extracellular."

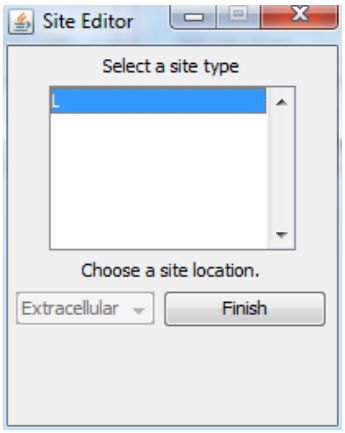



To define the ligand, we must first add a new site type, which we will call L. Click on "Add Type" and an editor will pop up allowing you to name the type and define its characteristics. We'll call the type "L", and give it a radius of 2 nm and a diffusion constant of 2 $\mu m^2/s$. We'll make the site green so that the coloring agrees with Figure 2. The ligand in this example only has a single biochemical state, so we will not add any states to this type. Click "Finish" and the newly defined type will be listed in the "Types" panel.

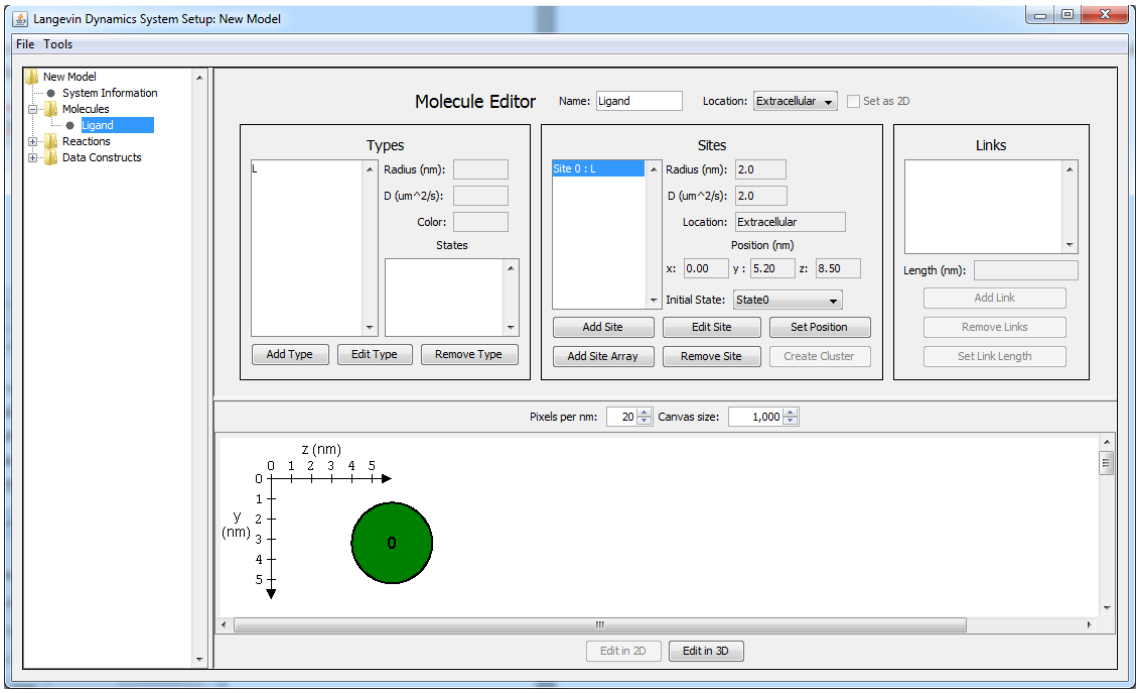
(**Note:** For the "Site Type Editor" and most other pop-up windows, it is not possible to click "x" to close the window. This feature has been disabled to force the user to use buttons, such as "Finish" or "Cancel", to close the window. This was done so that the program could provide some level of error checking to user input, and so the user's intention was made clear.)



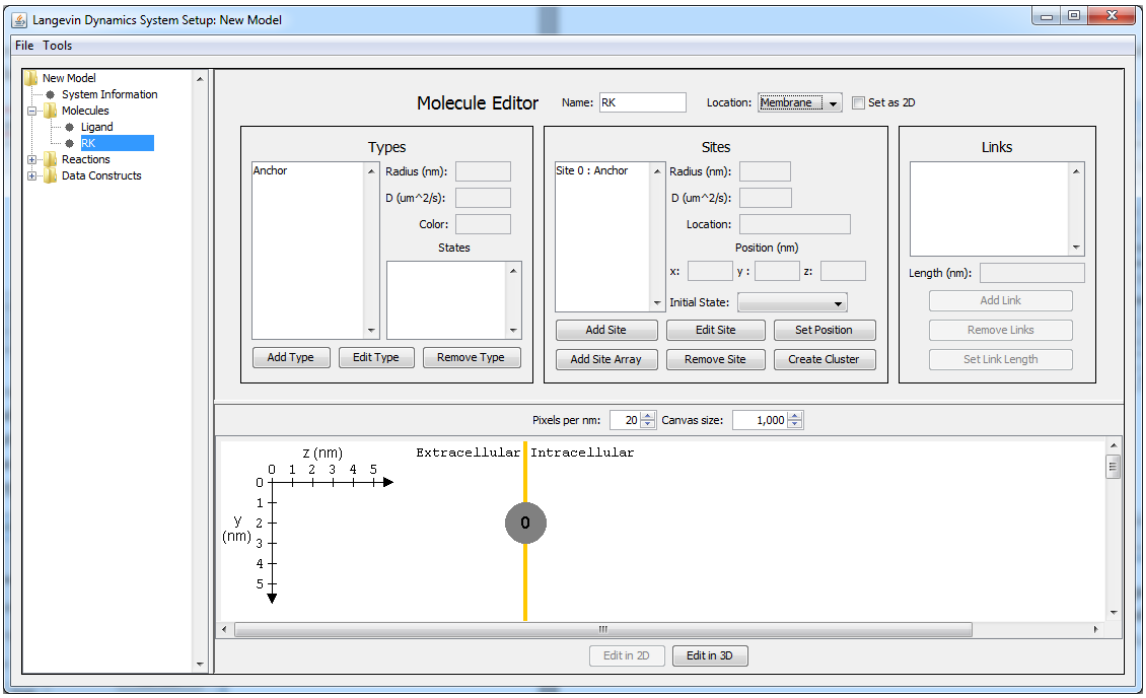

We can add a site to the molecule now that we've defined a site type. Click on "Add Site" and a site picker window pops up.




As we only have one site type to there is nothing to choose here. Click finish and you will see that the site has been added in in the molecule display panel.



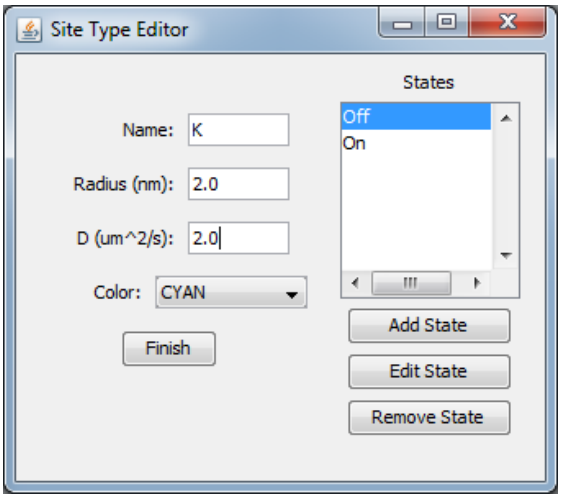
Now we define the receptor kinase. Click on the "Molecules" tab and again click on the "Add New" button. You will be taken to a blank molecule editor. Name the molecule RK and set it to be on the membrane. You will notice that (1) a site type called "Anchor" was created automatically, and (2) it was added to a membrane structure in the molecule display panel. The membrane structure may be dragged to a different location in the display panel. Finally, click the "Set at 2D" checkbox. This will force all sites to diffusion in a 2D plane parallel to the membrane.



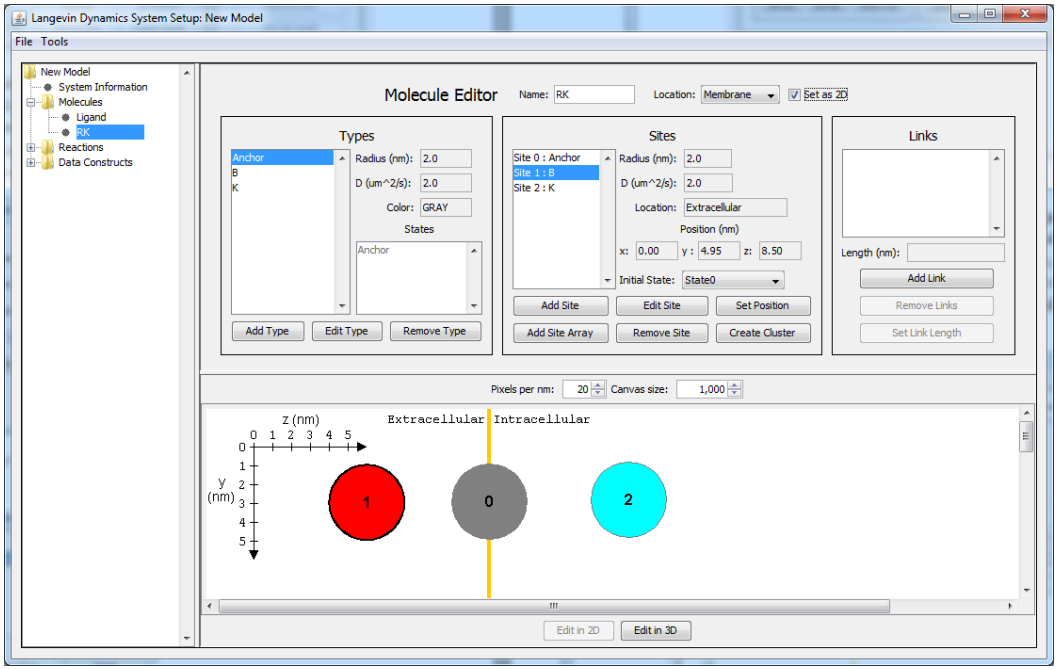

All membrane-bound molecules are required to have an inert membrane anchor whose sole purpose is to tether the molecule to the membrane. Additional anchors may be added if desired. The molecule display panel also indicates the extracellular and intracellular regions of the molecule. The anchor properties and color may be edited as with any other site type, but additional states may not be added. In this example we will change the Anchor type to have a radius of 2 nm.

We create a second site type called "B" for the ligand binding domain. This type will have two states, State0 and State1, representing a configuration without and with ligand, respectively. Open the site type editor and click on the "Add State" button. As these are the default names, just click finish.

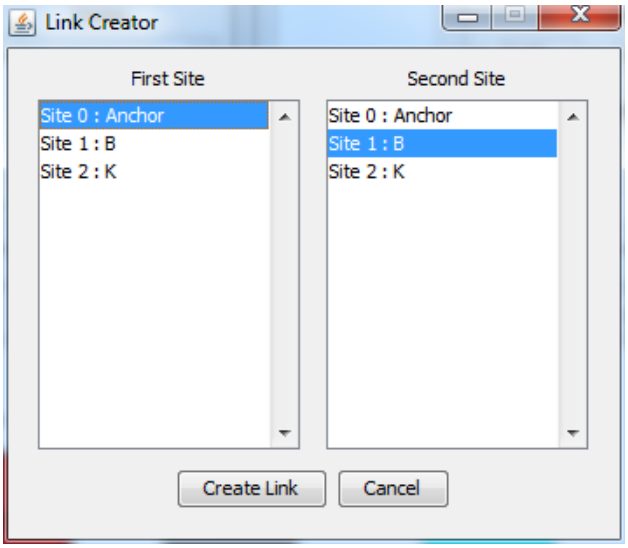
Lastly, we create a site type called "K" for the kinase domain. This type will have two states, "off" and "on." In the site type editor we click on the default state, "State0", and then click "Edit State" to change its name to "Off." We then click "Add State" and create the state "On." The site type editor should show the following before clicking "Finish."



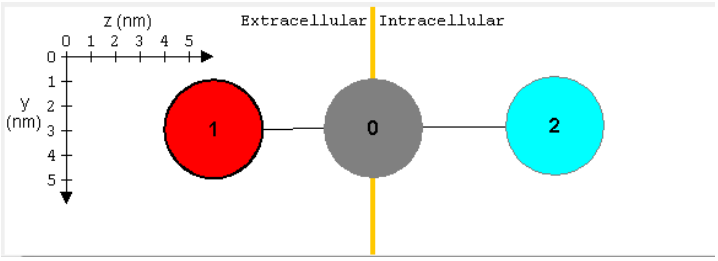

Now click the "Add Site" button to add a site. The site picking window will now let you choose between the three types defined for this molecule. Additionally, when adding the "B" and "K" types you can choose to add them to either the extra- or intra-cellular side of the membrane. Add a single "B" site to the extracellular side and a single "K" type to the intracellular side. The screen should show the following.



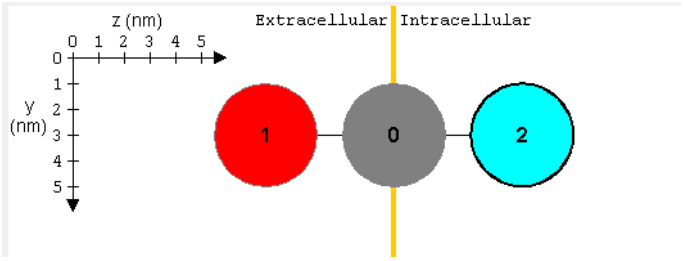

Now create links between these sites. We want to connect the anchor to each of the domains. Click on the "Add Link" button and the link creation window will pop up. Choose the Anchor site and the B site and click "Create Link."




Now add a second link between the Anchor and the K site. The molecule will look something like this.




To make the molecule linear we will explicitly set the positions of each site. First drag the Anchor site until its y-coordinate (which is shown in the "Sites" panel) is at 5 nm, and make note of its z-position. Then click on the B site, and in the Sites panel, click the "Set Position" button. A window will pop up allowing you to specify the position of the B site. Keep the x position at 0, set the y position to 5, and set the z-position to be 5 nm less than the z-coordinate of the anchor site. (Assuming the sites each have a radius of 2 nm, this will leave a bond of length 1 nm between the sites.) Repeat these steps for the K site, but now set the z-coordinate to be 5 nm larger than the Anchor's z-position. The molecule will look like this.

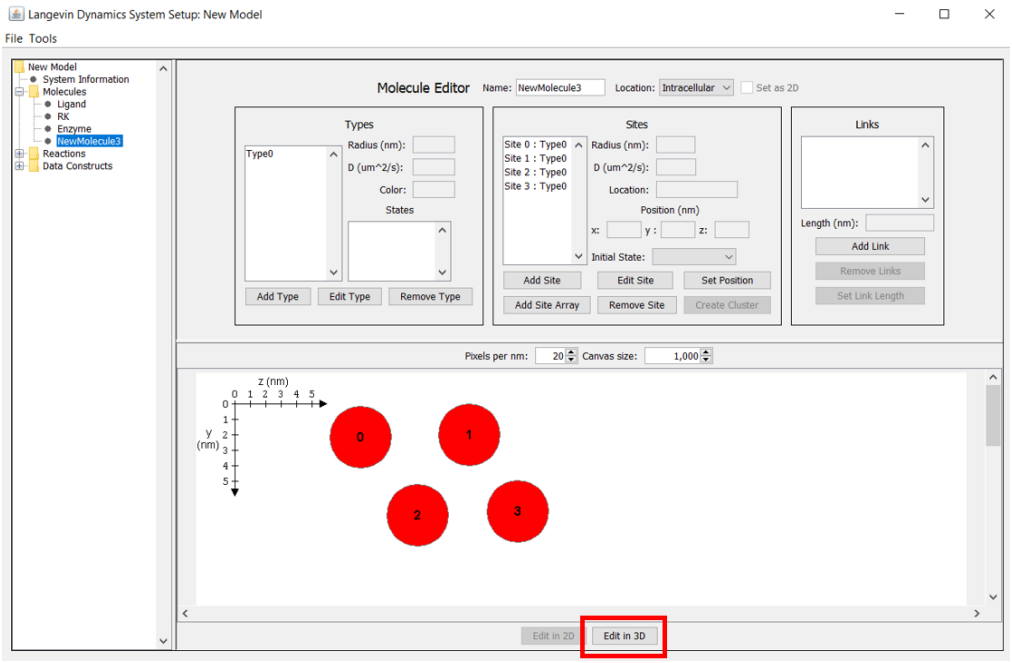



Before finishing with this molecule, click on the K site. The site panel will show its initial state. If it is not "off", click the drop down box and select the "off" state. Then click on the B site and make sure its initial state is "State0."

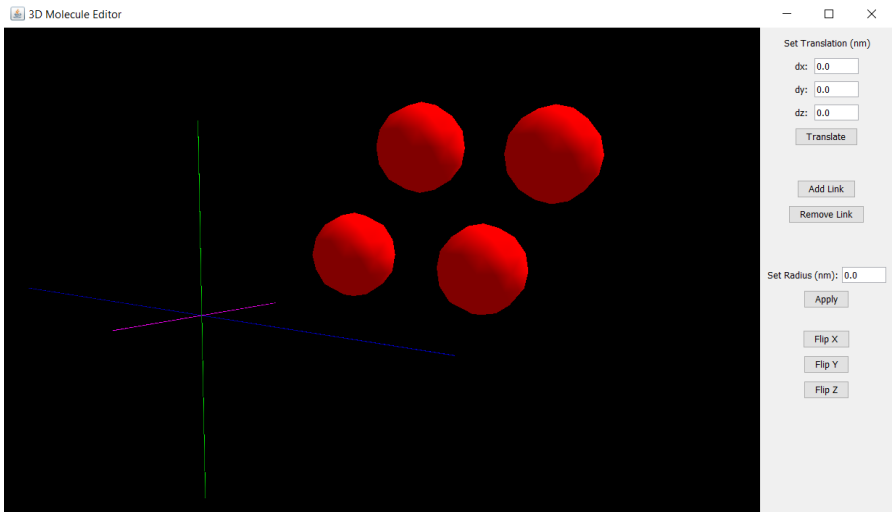
Lastly we create the substrate which is phosphorylated by the receptor kinase. Repeat the steps to add a new molecule, call it "Substrate", and set it to be intracellular. Create a single site type called S with two states, "u" and "p", and add a single site to the system. Check that the initial state is set to "u."

## 3D Editor

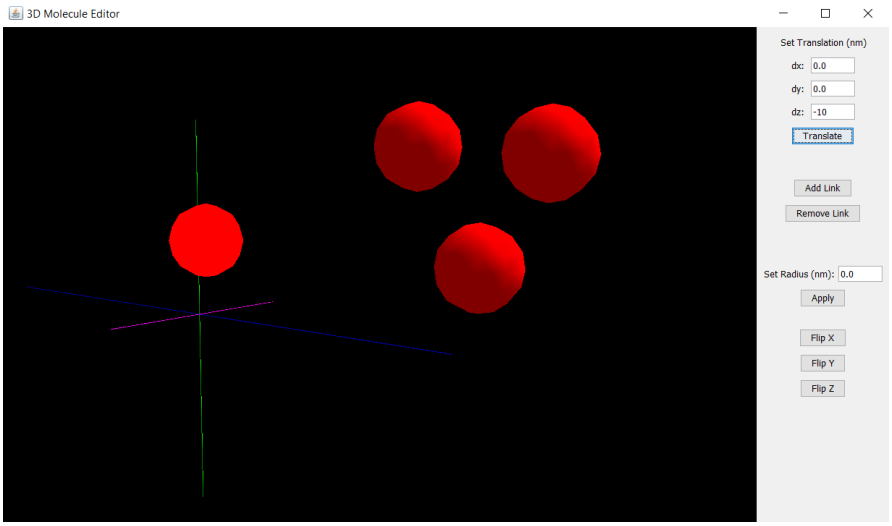
The molecules created above were either single sites or linear chains of sites. SpringSaLaD supports construction of an arbitrarily shaped molecule with an arbitrary number of sites. For example, suppose we want to construct a tetrahedral molecule composed of four identical sites. We create the molecule, make a single site type, and add four sites to the system. Next, click the "Edit in 3D" button at the bottom of the molecule display.




A new window will appear which shows the 3D configuration of the molecule.

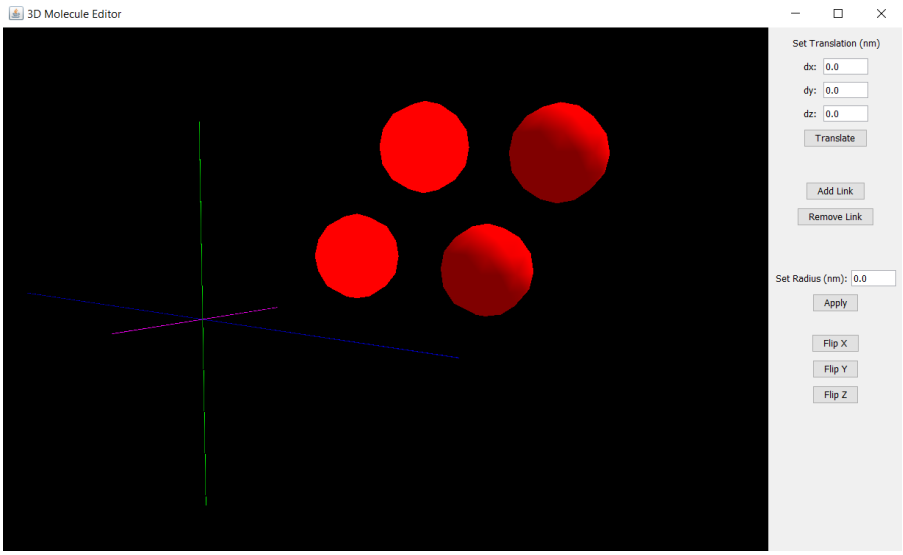



A particular site may be selected by clicking on it with the mouse. Its current position will be shown in the molecule editor panel. It may be translated to a new position by entering the desired translation distances on the right in the 3D editor panel, then clicking the "Translate" button.

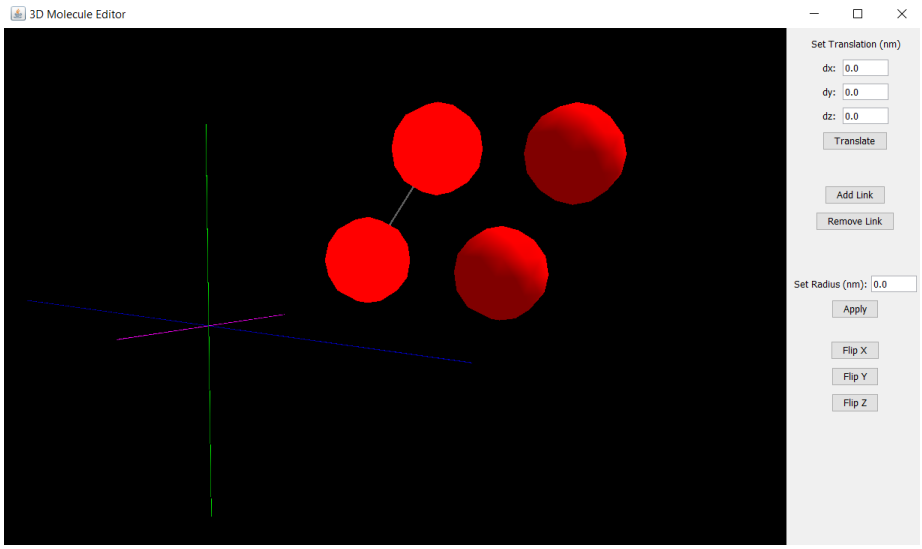



To translate multiple sites at once, hold the “Ctrl” key and use the mouse to click multiple sites. When all the desired sites are highlighted, enter the desired translation distances, and click the “Translate” button.

Links may be directly added between two sites in a molecule. To select two sites hold the “Ctrl” key and use the mouse to click on the two sites to link.




Pressing “Add Link” will create a link between the two sites. Similarly, selecting a link and clicking the “Remove Link” button will remove that link. Only one link between two sites may be added at a time.

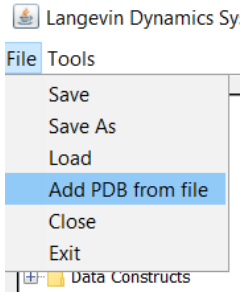



Additional sites may be added through the molecule editor panel, as in the 2D case. We will not bother with the remainder of the steps required to create the tetrahedron. The rest of the options and buttons in the 3D editing panel are especially useful in the creation of a model from a structure file and will be explained as part of that process.

### *Create 3D Molecule From Structure File*

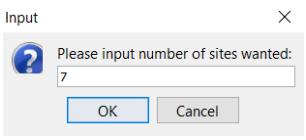
SpringSaLaD supports the semi-automated construction of a 3D molecule from a protein structure file (".pdb" or ".cif" format) that contains detailed atomic data about a molecule. The steps to add a molecule from such a file to the current model are outlined below:

1. Navigate to the "File" tab. Then select the "Add PDB from file" option:




   The user should then select the file using the file browser that is brought up. Following selection of the file, a pop up will appear, prompting further input.

2. Enter the number of sites to generate from the structure file and hit the "OK" button:

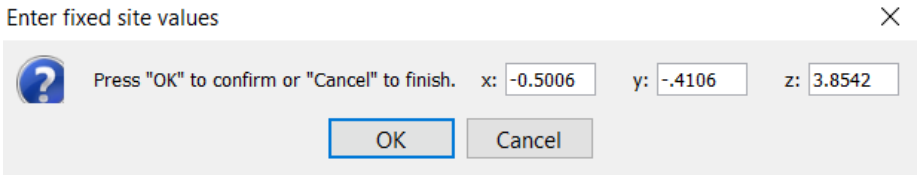



3. Enter coordinates to control automated molecule generation (optional):

With no additional input, SpringSaLaD will automatically generate a molecule with the number of sites entered. However, the user can control this behavior by "fixing" sites if the coordinates for the locations of important residues in the molecule are known. Fixing a site will ensure that when SpringSaLaD automatically generates the molecule it creates a site at the desired user location in that molecule. The decision to fix sites is ultimately a user decision based in part on how well the user decides SpringSaLaD generated the molecule.

The location of a residue can be found from opening the structure file either in a text editor and searching for the location, or viewing the file in a visualization program that can provide the coordinates when selecting an atom.


Here using Jmol, an atom at location (-5, -4, 38) is selected. It is important to translate the coordinate system used to nm. Once the coordinates are entered, click the "OK" button.

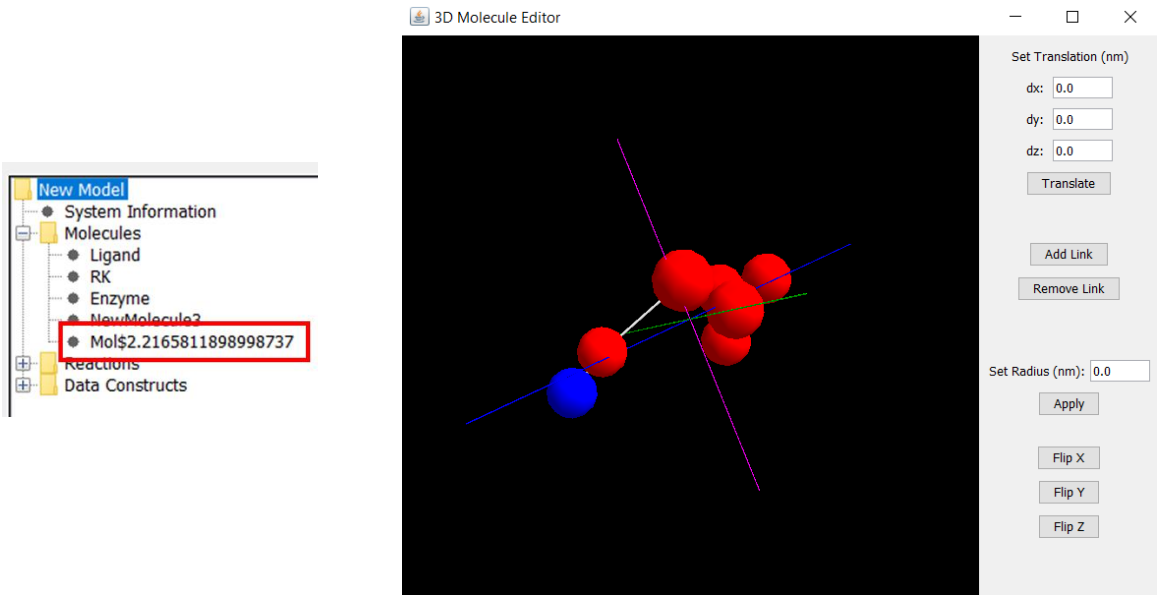



The user may input as many fixed sites as total sites entered in the previous step. To stop entering fixed sites after any number have been entered (or if 0 fixed sites are desired), click the "Cancel" button.

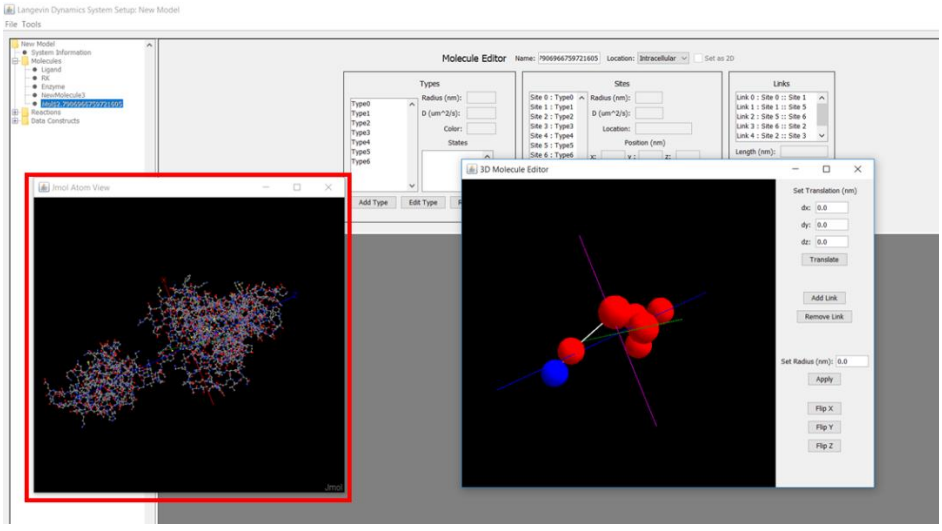
4. View the molecule and edit the molecule:

To view the automatically generated molecule, find the new molecule under the "Molecules" options in the left-hand panel. It will appear with a unique identifier that can be changed by changing the molecule name. After clicking on the molecule name, the 3D editing window will open. By default, each site will

have its own type. Fixed sites will appear blue and the automatically generated sites will appear red. These and other defaults can be changed as described previously.

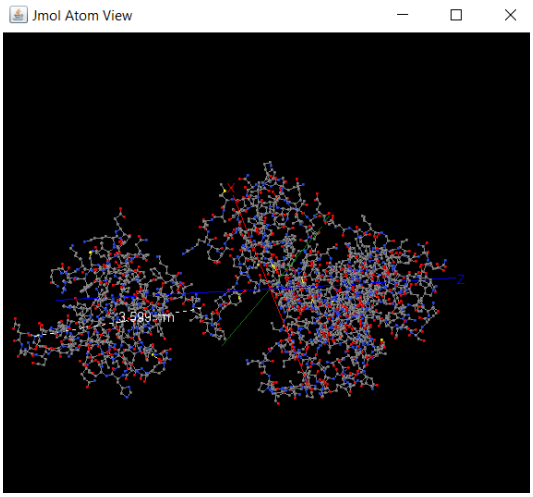



In addition, a "Jmol Atom View" window will appear. This Jmol window displays the structure file through Jmol integration with SpringSaLaD. It maintains many of the functions of the Jmol stand-alone application (http://jmol.sourceforge.net). An easy way to access these functions is by left clicking the mouse in the Jmol window. The Jmol window will also appear for molecules edited in 3D without a corresponding structure file, but will appear empty.

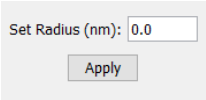



The user should use the two windows side by side to edit the automatically generated molecule in the 3D editing window to more accurately reflect the actual 3D structure. A few options SpringSaLaD supports to do this have been noted previously (add/remove sites/links, translate site location, etc.). Three additional options available for model editing are found below.

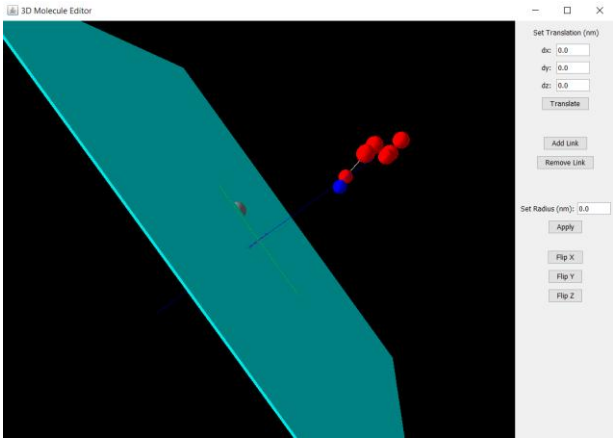
The user can use the Jmol window to measure the distance between two atoms. This can be used to check that the radius of a site and length of a chain of sites. To measure distance, double click on the start atom and then double click on the end atom. The distance between them will appear.

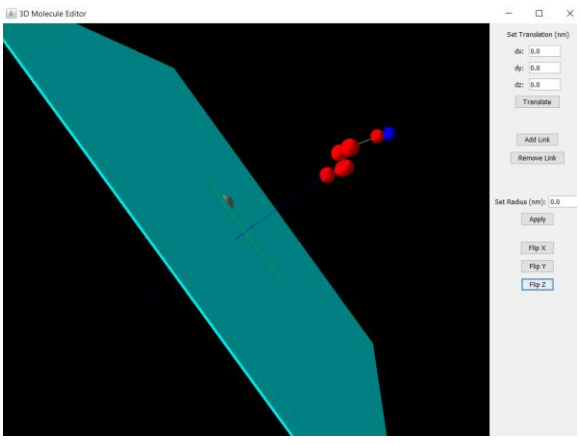



The radius of a site can be directly edited from the 3D Molecule Editor window, by entering the desired value next to “Set Radius (nm)” and then clicking the “Apply” button. In addition to the selected molecule, the radii of all sites with the same type will be changed as well.




The orientation of the entire molecule can be flipped with respect to the x, y or z axis. To do so the user can click either the “Flip X” button, “Flip Y” button, or the “Flip Z” button. The user may find the flipping function useful to orient the molecule with respect to the membrane. For example, if the fixed sited (blue) should be farthest from the membrane, the user can click the “Flip Z” button to achieve the desired orientation. Adding the proper links, and translating the molecule closer to the membrane are next steps that are not shown.



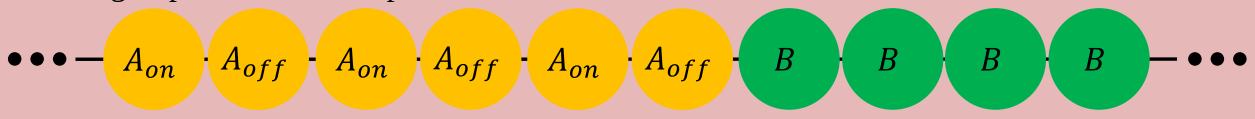



## Additional Options

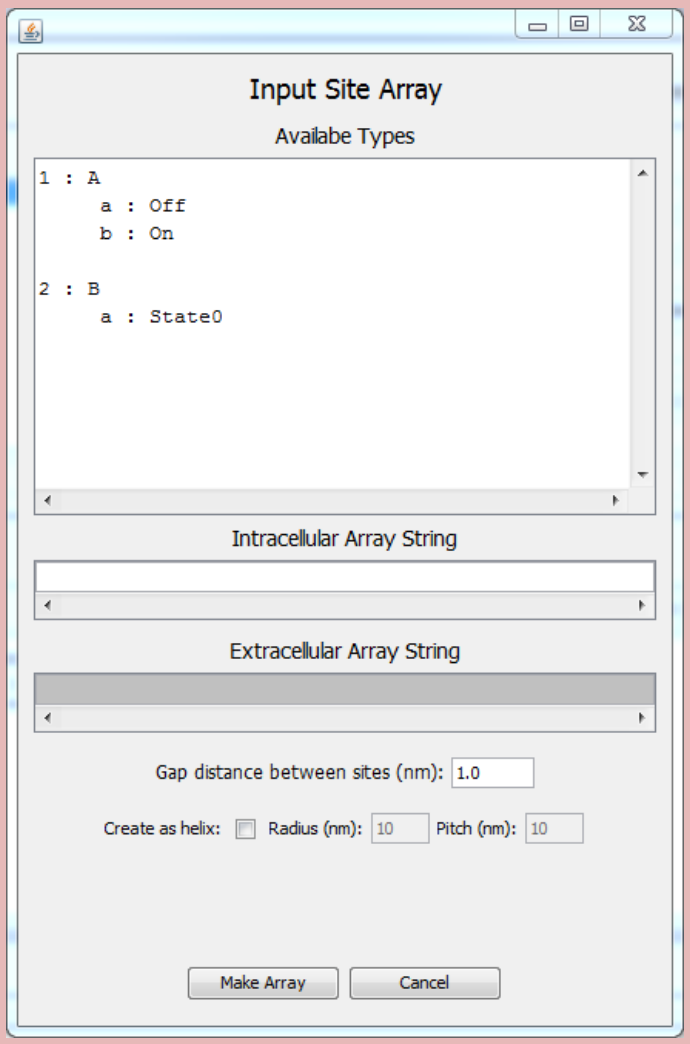
There are several buttons and options in the Molecule Editor panel which we did not discuss, but most of these are self-explanatory, such as the "Remove Site" button, and most will give a warning or prevent an action that could invalidate the model. For example, the "Remove Site" button will not remove a site which is currently connected to a link, and the program will ask you to remove the link first. However, there are two options which may require some instruction, and we discuss these here. The reader who is not interested in these options can safely skip this section.

### *Create Site Array*

SpringSaLaD contains a convenience method for creating long polymers (site arrays). For example, imagine a polymer which consists of two different site types, A and B, and assume type A can be in one of two states, "on" or "off", and that the polymer is arranged with 6 A sites alternating on and off, followed by 4 B sites, and this group of 10 sites is repeated 5 times for a total of 50 sites.




We follow the usual steps to create a new molecule and create the two site types. Click on the "Add Site Array" button in the Sites panel to open the site array editor.



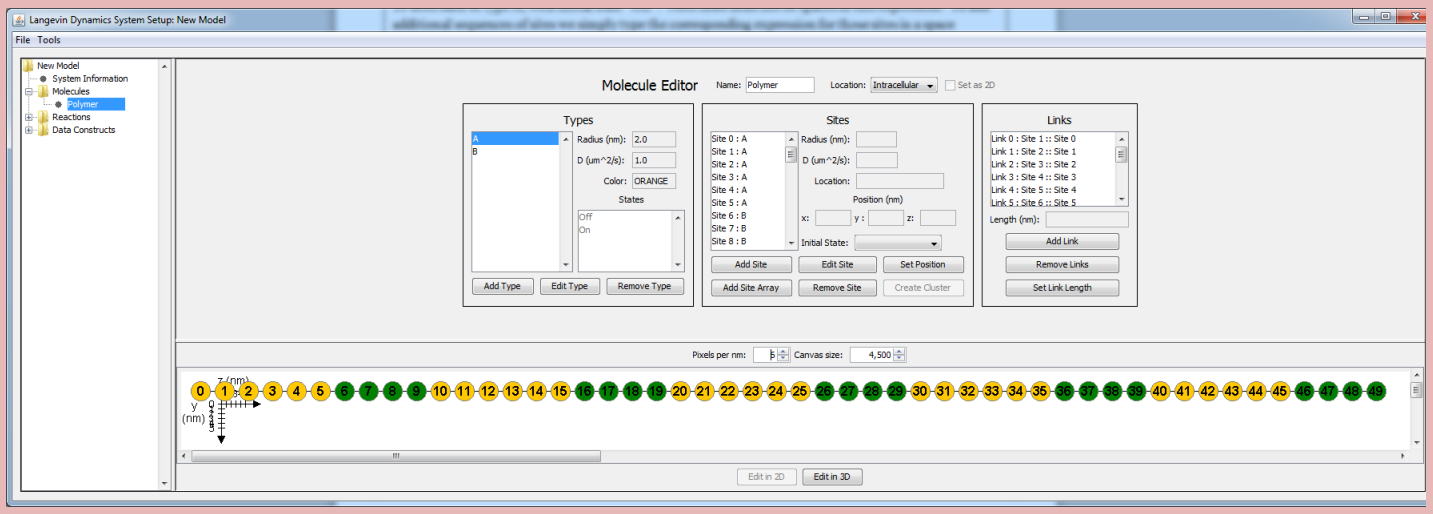
Each type has been assigned a number, in this case A is assigned to 1 and B is assigned to "2". Each state has been assigned a letter, as indicated in the screenshot. The syntax for creating an array is as follows:

To create N identical sites in a row we write, for example, 1a(N). Thus, 1a(10) tells the program to create 10 sites each of type A, with initial state "Off". Note there must not be spaces in this expression. To add additional sequences of sites we simply type the corresponding expression for those sites in a space separated list. For example, 1a(4) 2a(3) 1b(5) would create a polymer with 4 A off sites, 3 B state0 sites, and 5 A on sites. Finally, if a section of the polymer repeats we can indicate that by enclosing that section in square brackets, followed by the number of repeats in parentheses. For example, rather than typing 1a(3) 2a(4) 1a(3) 2a(4), we can simply type [1a(3) 2a(4)](2). This bracket system allows the user to define arbitrary patterns in a concise notation.

For the polymer described above we would enter the following expression in the text box under "Intracellular Array String", [ [1a(1) 1b(1)](3) 2a(4)](5) , which first creates a pair, Aoff and Aon, then repeats this pair three times, and follows those six sites with four B sites. Finally those 10 sites are repeated 5 times, for a total of 50 sites.

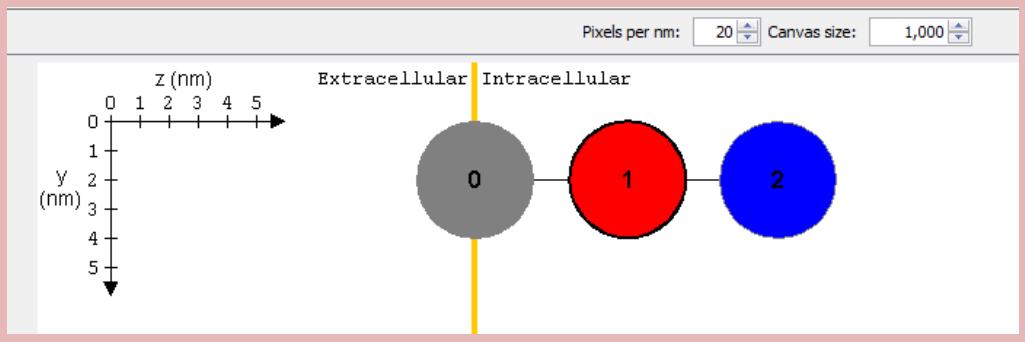
Next we define the gap distance between the sites. The default is 1 nm, which we'll use here.

Finally, we can create the polymer as either a linear chain or as a helix. The default is a linear chain. To make a helix click the "Create as helix" checkbox, and enter the desired radius and pitch of the helix. Here we'll use the default linear option. Click finish and we can view the polymer we just created (the display window can be resized to view the entire polymer).

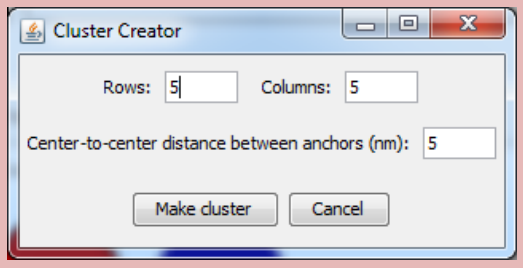


### *Create Membrane Cluster*

One of the reasons for making SpringSaLaD was to develop a platform to study the consequences of membrane clustering, and we have included a convenience method to create a large membrane cluster from a single membrane molecule. To use this feature, first create a membrane-bound molecule, then click on the "Create Cluster" button. For this example we created the following simple membrane molecule.

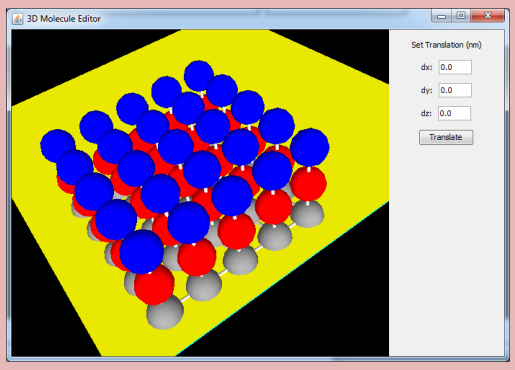



Clicking the "Create Cluster" button brings up the following window,




which is used to define the number of rows and columns of a rectangular grid and the spacing between the anchor sites.

We use the default options (5x5 cluster, 5 nm spacing) to create the following cluster,

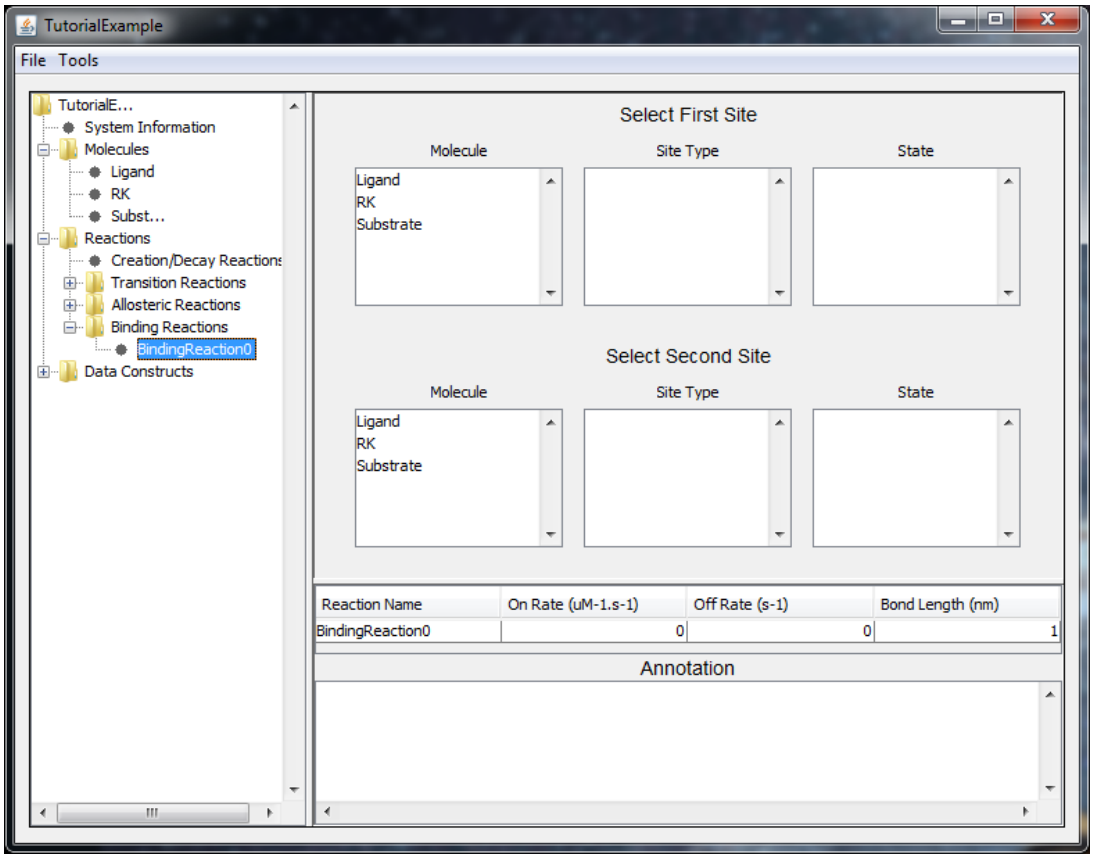



which is now treated as a single large molecule. Note that creating a cluster replaces the original molecule with the cluster.

## Defining Reactions

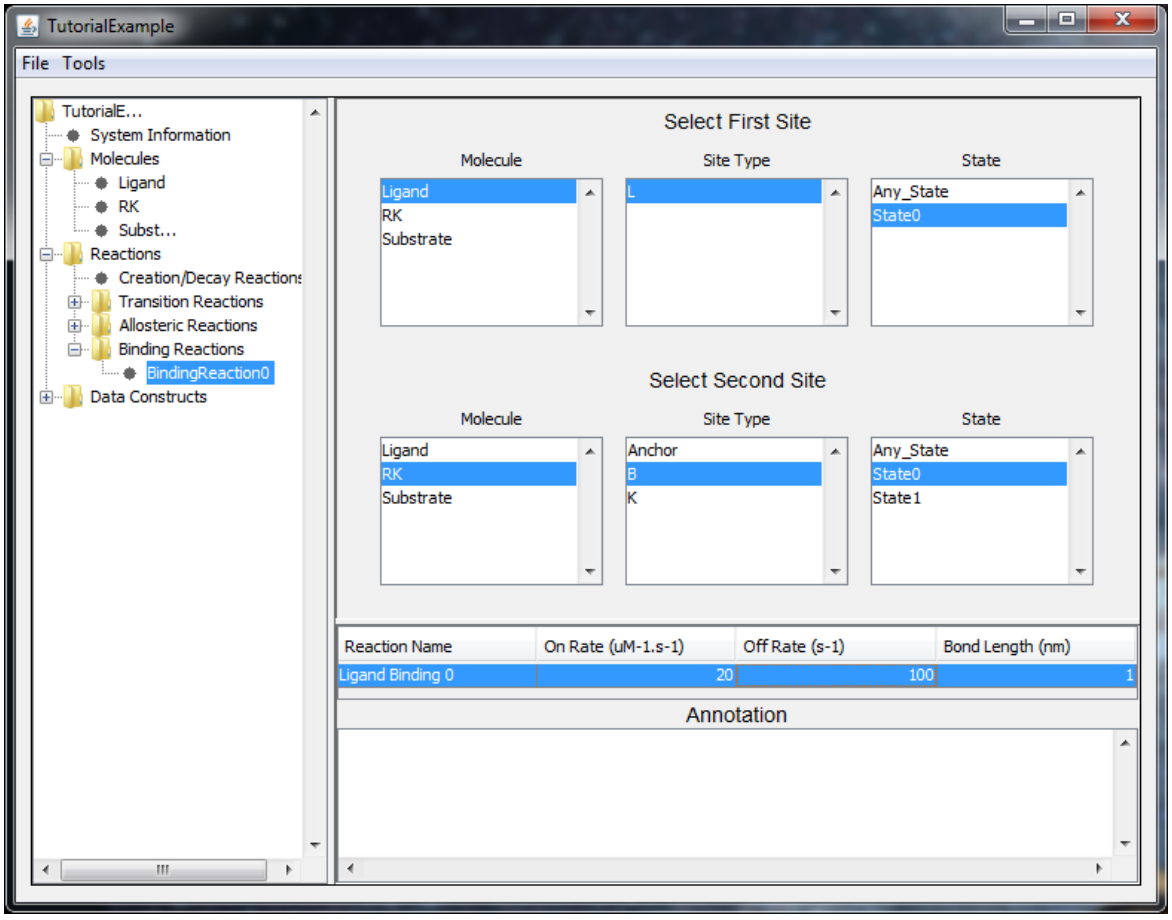
### Binding Reactions

We will first define the binding reaction between the ligand and receptor. Click on the "Binding Reactions" tab in the left panel, and then click the "Add new" button in the center panel. The binding reaction panel will appear.




Binding reactions are defined by specifying the following, in order: the molecule, the site type, and the state the type must be in to react. An "Any State" option exists to indicate a reaction is always allowed.

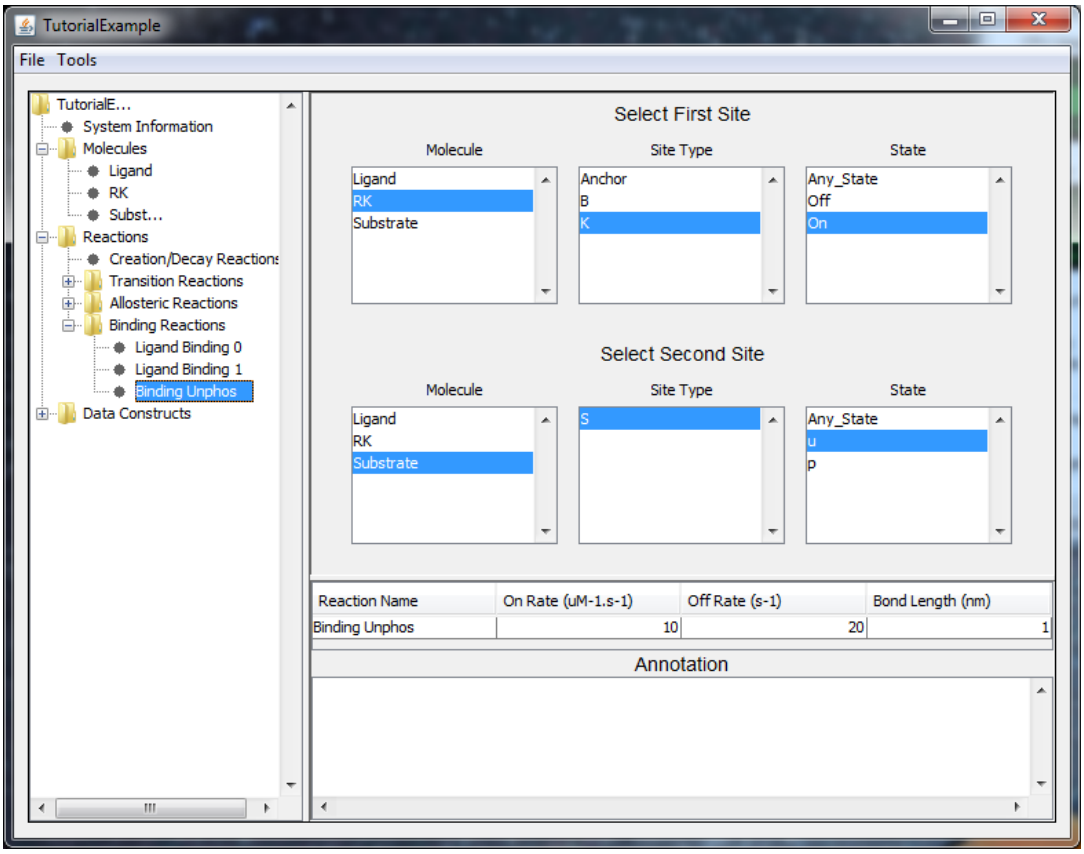
First we will define the binding reaction between ligand and the B site in state 0. The ligand only has a single site and a single state, so there's nothing to choose there. For the receptor kinase we choose the B site, and then choose state 0. We name this reaction "Ligand Binding 0", provide an on-rate and an off-rate, and define the length of the newly created bond. Note that this length refers to the distance between the surfaces of the sites, not the center-to-center distance. After defining the reaction the screen should look like the following snapshot.

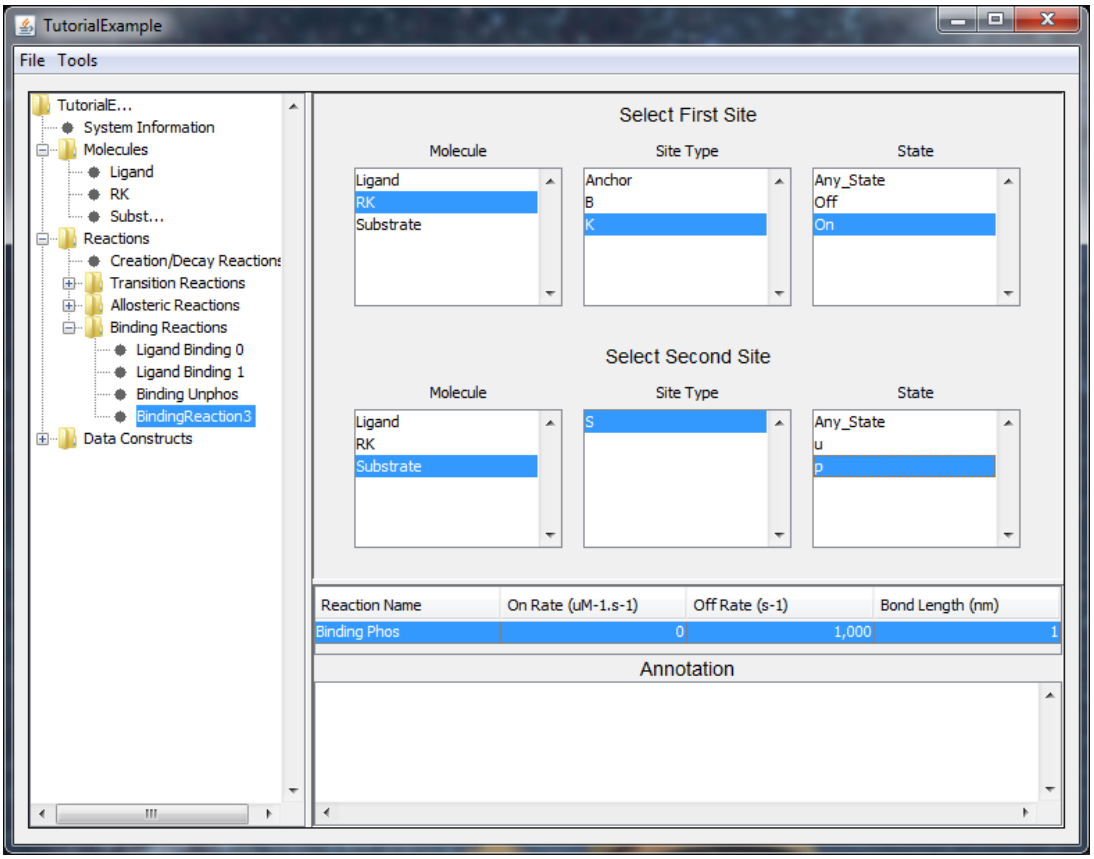



Note: All reactions are automatically saved (there is no need to click a "Finish" button). Partially completed reactions are allowed while building a model, but a model cannot be saved unless all reactions are completely defined. For example, if you start a reaction but then realize you forgot to add a state to a site, you can go back and edit the molecule without losing the partially completed reaction.

Next we define the binding reaction between ligand and the binding domain in state 1. We assume that this state has a very high affinity for the ligand, so we'll double the on rate and reduce the off rate 10-fold. The steps are similar to those above. Call this reaction "Ligand Binding 1."

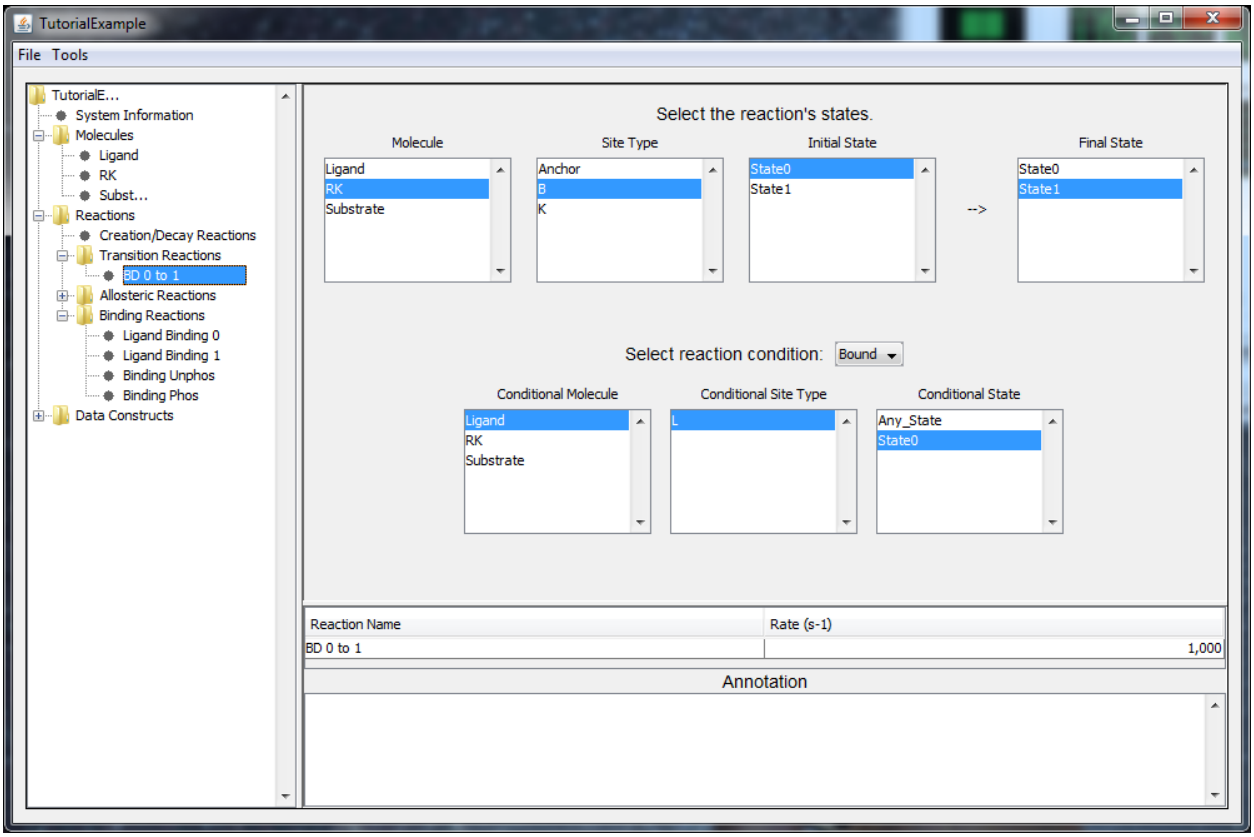
Next we define the binding reaction between the receptor-kinase and the substrate which it phosphorylates. We must define two different binding reactions, corresponding to the substrate in the phosphorylated and unphosphorylated state. The phosphorylated substrate can dissociate but cannot bind to the receptor kinase. We call these two reactions "Binding Unphos" and "Binding Phos". (Note that although we call the reaction "Binding Phos," we will only allow the phosphorylated form to dissociate.) The screen shots for each reaction follow. Recall that binding can only occur when the kinase site is on. The on rate for "Dissociation Phos" is set to 0 to prevent the phosphorylated substrate from binding to the kinase.


TutorialExample
File Tools
TutorialE...
System Information
Molecules
Ligand
RK
Subst...
Reactions
Creation/Decay Reactions
Transition Reactions
Allosteric Reactions
Binding Reactions
Ligand Binding 0
Ligand Binding 1
Binding Unphos
Data Constructs
Select First Site
Molecule
Site Type
State
Ligand
RK
Substrate
Anchor
B
K
Any_State
Off
On
Select Second Site
Molecule
Site Type
State
Ligand
RK
Substrate
S
Any_State
u
p
Reaction Name
On Rate (uM-1.s-1)
Off Rate (s-1)
Bond Length (nm)
Binding Unphos
10
20
1
Annotation

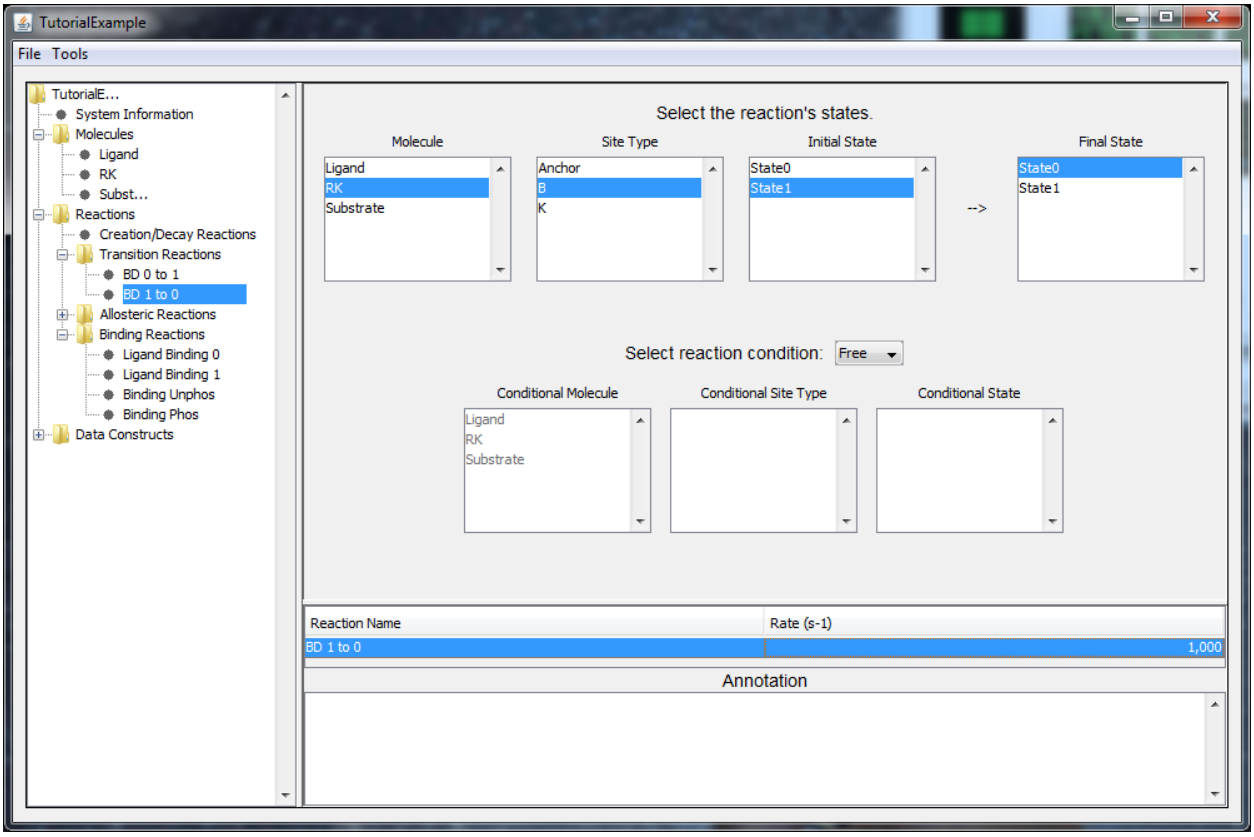

TutorialExample
File Tools
TutorialE...
System Information
Molecules
Ligand
RK
Subst...
Reactions
Creation/Decay Reactions
Transition Reactions
Allosteric Reactions
Binding Reactions
Ligand Binding 0
Ligand Binding 1
Binding Unphos
BindingReaction3
Data Constructs
Select First Site
Molecule
Site Type
State
Ligand
RK
Substrate
Anchor
B
K
Any_State
Off
On
Select Second Site
Molecule
Site Type
State
Ligand
RK
Substrate
S
Any_State
u
p
Reaction Name
On Rate (uM-1.s-1)
Off Rate (s-1)
Bond Length (nm)
Binding Phos
0
1,000
1
Annotation

## Transition Reactions

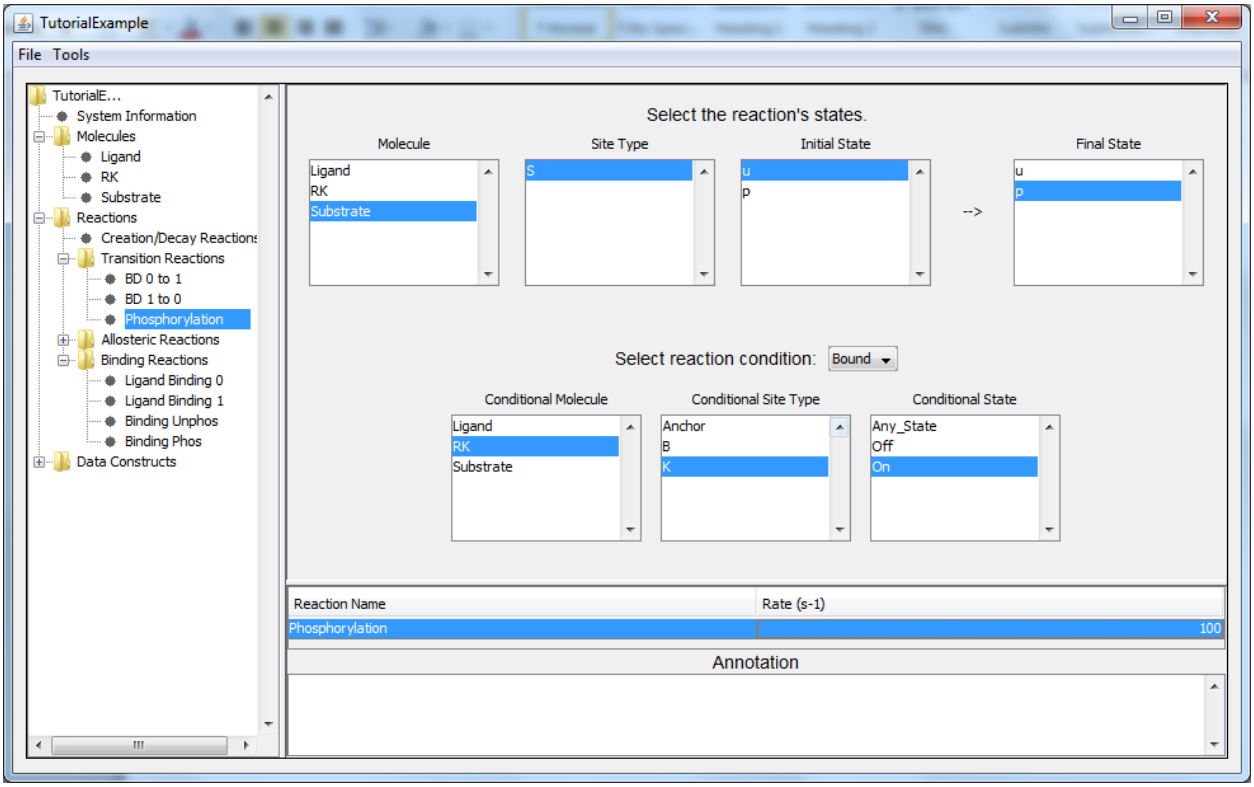
First we define the reaction which changes the configuration state of the binding domain from state 0 to state 1. Click on the "Transition Reactions" tab, then click "Add new." To define a transition reaction you must provide the molecule, site type, initial state, and final state of the reaction. In addition, a reaction condition can be defined. In this case the reaction can only proceed when the binding domain is actually bound by ligand. Thus, we choose the "bound" condition, and choose the ligand as the conditional molecule. Call this reaction "BD 0 to 1." The screen should look as follows.

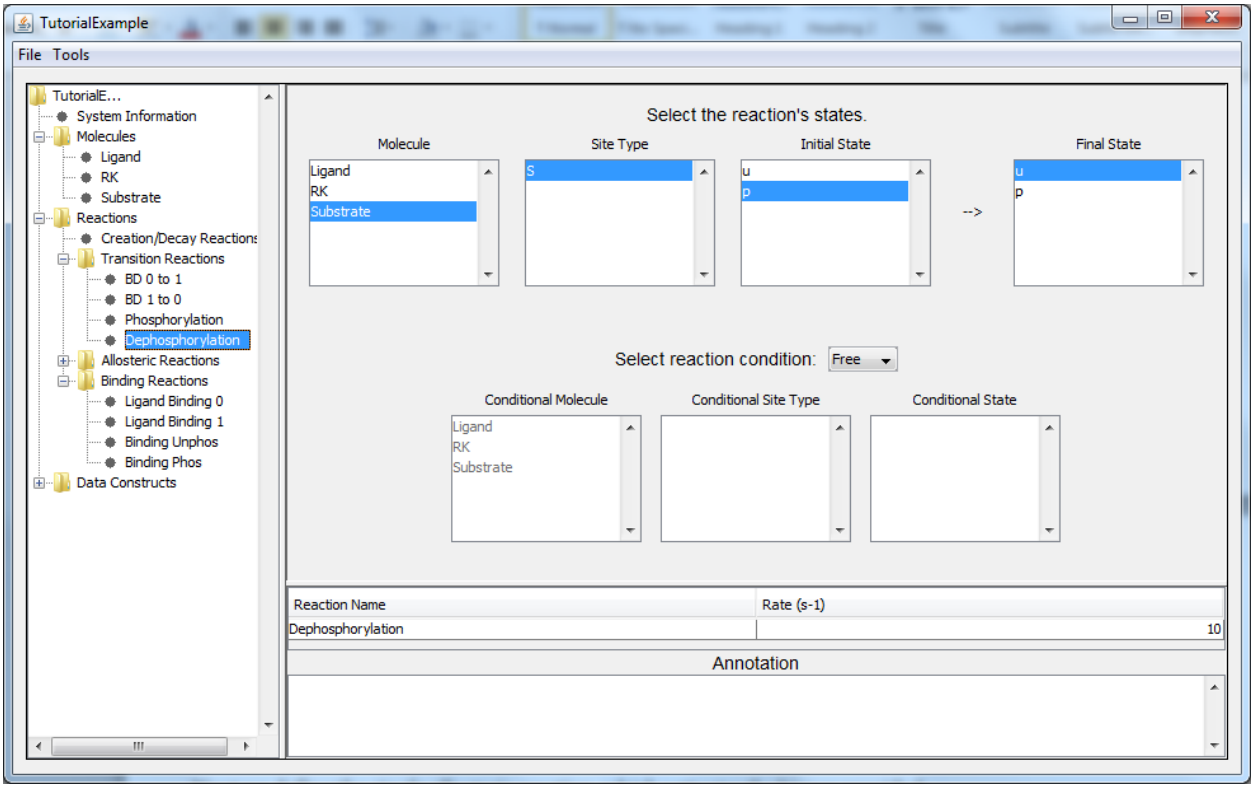



Next define the reaction which reverses the above reaction, converting the binding domain from state1 to state 0. In this case we only want the reaction to proceed when ligand is not bound to the domain, so we choose the "free" condition. Call this reaction "BD 1 to 0." The screen should look as follows.

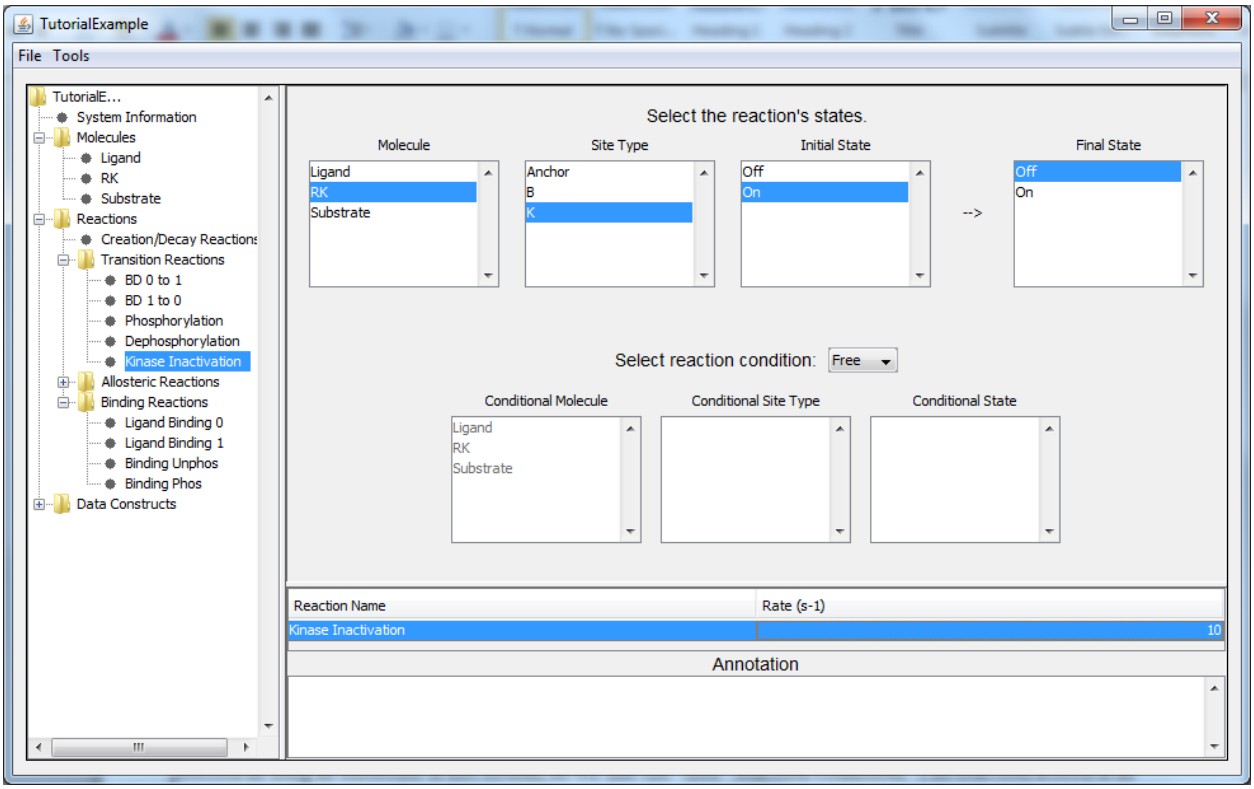



Now define the phosphorylation and dephosphorylation reactions for the substrate. The former requires the substrate to be bound to the "on" kinase domain, while the latter requires the substrate to be free (we could explicitly add a phosphatase, but we assume spontaneous dephosphorylation to keep the model simple). The reactions screens are as follows.



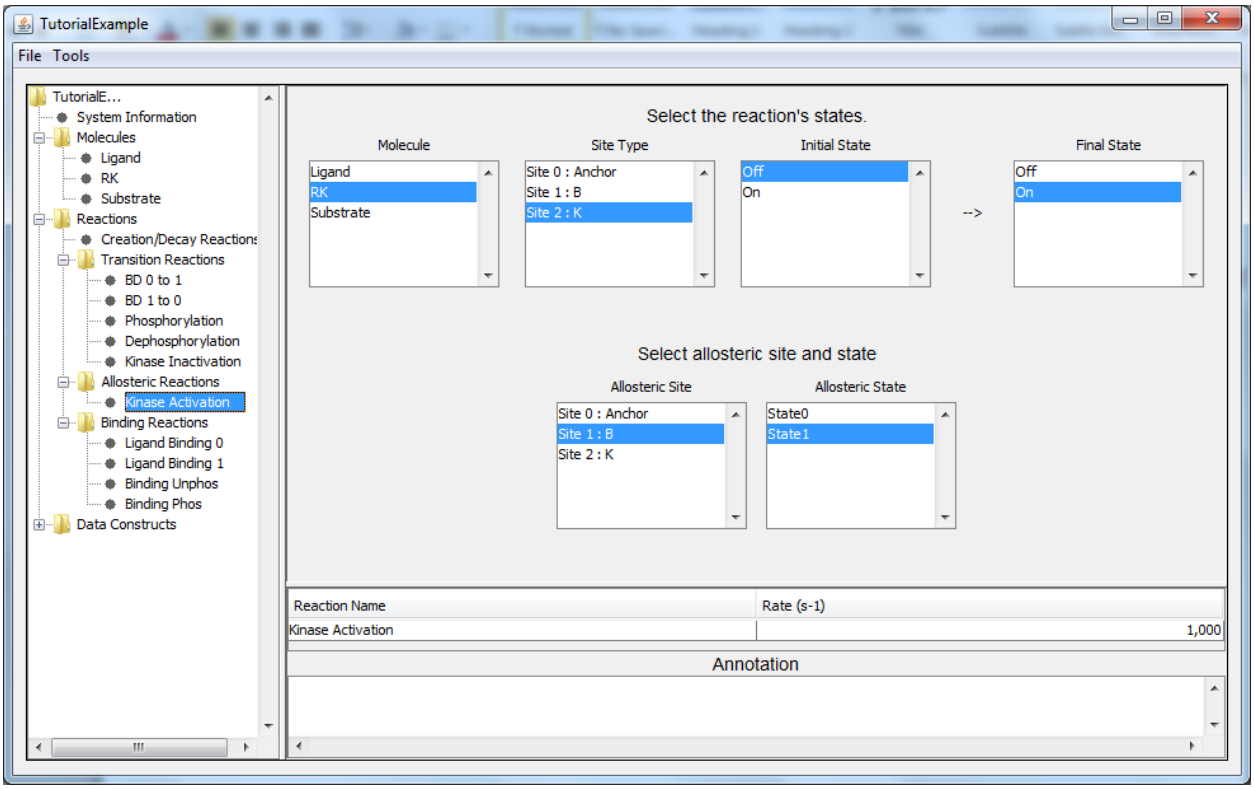



Lastly, we define the transition reaction which deactivates the kinase domain. We assume this reaction can proceed as long as substrate is not bound, so we use the "free" reaction condition. The reaction screen is as follows.



### Allosteric Reactions

Next we define the single allosteric reaction, which activates the kinase provided the binding domain is in state 1. Click on allosteric reactions, then click the "Add new" button. The upper part of the panel is similar to the transition reaction panel, but requires the user to select a specific site in the molecule, not just a site type. In this case there is no distinction because each site is assigned to a different site type, but it is possible for a molecule to have two sites with identical site types but restrict the allosteric reaction to one of them. The lower part requires the user to specify the allosteric site in the molecule and the state that site must be in for the reaction to proceed. In this case we choose the binding domain as the allosteric site and "State 1" as the allosteric state. The screen should look as follows.




## Data Constructs

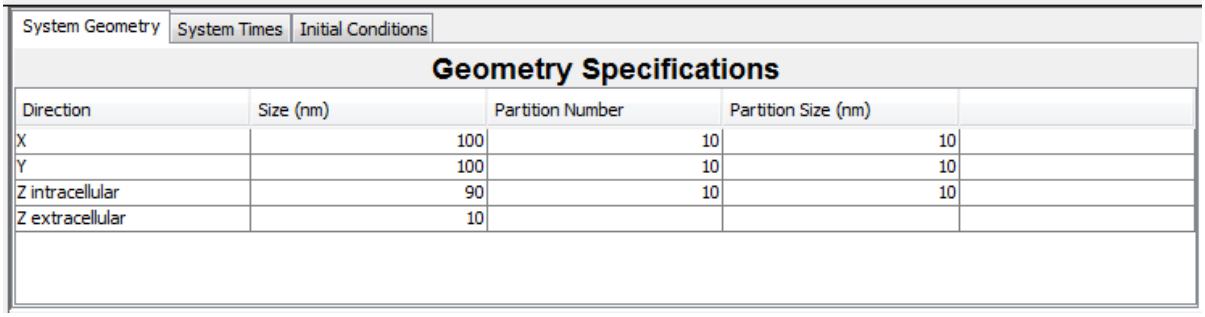
SpringSaLaD keeps track of various quantities at all levels of the simulation, from molecule statistics to information on the states of specific sites. There are five general classes of data counters which are listed under the "Data Constructs" tab: molecule counters, state counters, bond counters, site property counters, and cluster counters. By default the first four classes are activated, while cluster data is deactivated because tracking cluster sizes generates large data files. In general there is no reason to change the defaults unless the simulation is being used to track clustering, in which case the cluster counter should be turned on by setting the checkbox in "Cluster Counter" panel.

## System Information

The system information panel allows the user to specify the system geometry, the system times, and the initial conditions. Clicking on the "System Information" tab will open a panel with a tab corresponding to each of these categories.

### System Geometry

At this point SpringSaLaD only support a single system geometry, namely, a cubic geometry with a single planar membrane at z=0. The system geometry tab looks as follows.



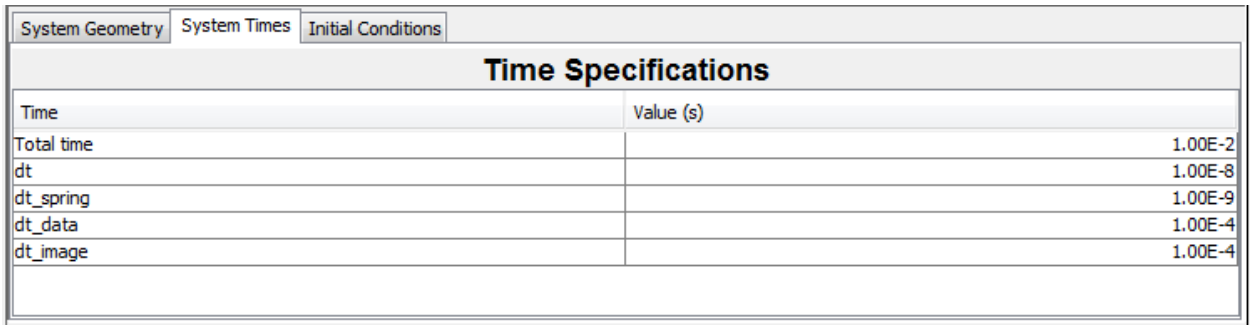

| Direction | Size (nm) | Partition Number | Partition Size (nm) |
|---|---|---|---|
| X | 100 | 10 | 10 |
| Y | 100 | 10 | 10 |
| Z intracellular | 90 | 10 | 10 |
| Z extracellular | 10 | | |

The first column specifies the size of the cube in the x and y directions (parallel to the membrane), as well as the height above (intracellular) and below (extracellular) the membrane. The default values specify a rather small system. We will increase the system size as follows: $x = 200$ nm, $y = 200$ nm, $z_{intra} = 200$ nm, $z_{extra} = 200$ nm.

SpringSaLaD reduces the computational cost of collision detection by partitioning space into cubic domains, and the second column specifies the number of partitions to use in each direction. By default there are 10 partitions in each direction, which equates to $10^3 = 1000$ partitions used in the simulation. This number usually offers a good tradeoff between the increased number of overlap checks needed with too few partitions and the computational overhead incurred when using many more partitions. Slight performance improvements might be obtained by finding an optimal number of partitions. The user must be aware of the following restriction: **the minimum partition size must be larger than the diameter of the largest site in the system**. Thus, the user must pay attention to the partition size, listed in column three, especially when using very small systems or a very large number of partitions. With the default values and the system sizes listed above, the partitions used here are 20 nm x 20 nm x 40 nm, safely larger than the maximum site diameter of 4 nm.

### System Times

The system times tab looks as follows.




| Time | Value (s) |
|---|---|
| Total time | 1.00E-2 |
| dt | 1.00E-8 |
| dt_spring | 1.00E-9 |
| dt_data | 1.00E-4 |
| dt_image | 1.00E-4 |

There are five times which must be specified.

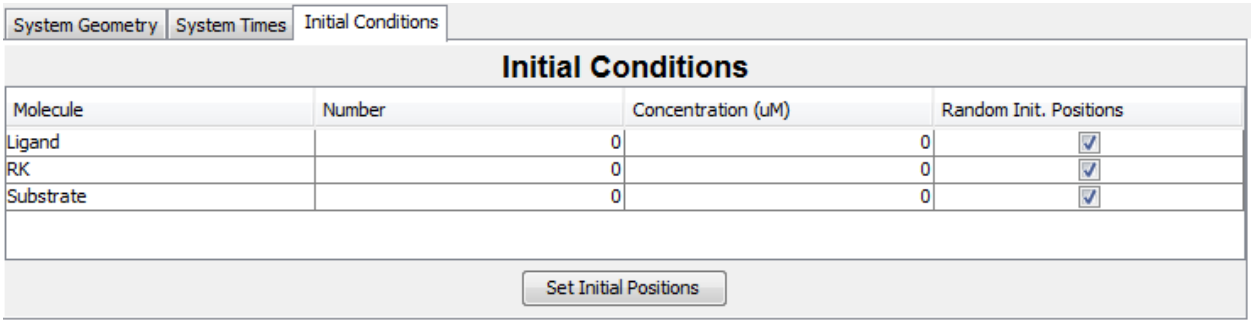
1) Total time: This is the total simulation time.
2) dt: The time step. Suitable values are in the $1 - 10$ ns range.
3) dt_spring: This specifies the time step to use for spring relaxation. There is no reason to change the default value.

4) dt_data: This specifies the time interval between taking data on the system configuration (molecule counts, state information, etc.). The total number of data points will be $N = dt/dt\_data$.
5) dt_image: This specifies the time interval between storing system snapshots for the 3D viewer.

We will use the default values for now.

### Initial Conditions

The initial conditions tab looks as follows.




The initial conditions can be defined using either particle number or concentration. By default the molecules are placed randomly in their allowed space (intra- or extra-cellular, or on the membrane). The initial positions of molecules can be set by the user by unchecking the "Random Init. Positions" checkbox and then clicking the "Set Initial Positions" button. The user is cautioned that this feature is still in beta and SpringSaLaD performs minimal error checking, and thus care must be taken to ensure the entered positions are valid.

SpringSaLaD uses the intracellular (extracellular) volume when calculating the concentration of intracellular (extracellular) species, as is appropriate. However, for membrane-bound species SpringSaLaD uses the total system volume when displaying the concentration in this pane. It is expected that future versions of SpringSaLaD will display surface concentrations for membrane-bound molecules.

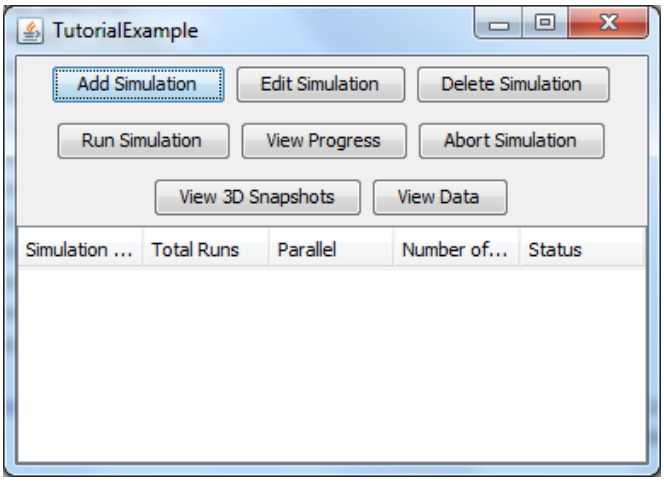
For the tutorial system we specify 20 ligand, 10 RK, and 20 substrate molecules.

## Saving the System

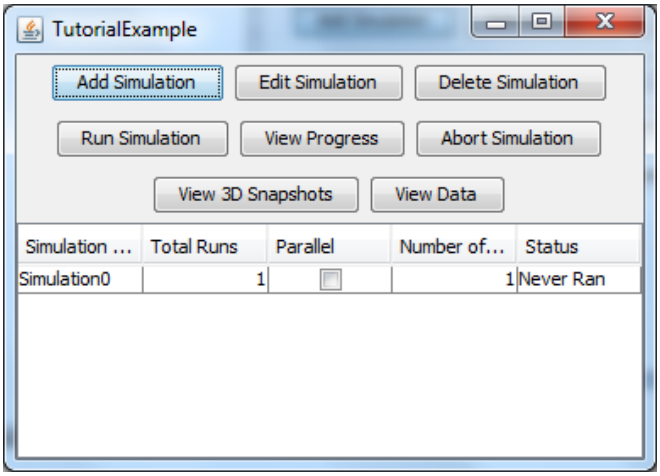
SpringSaLaD requires the system to be saved before running simulations. To save the system, click on the "File" menu, then click "Save." A save dialog will pop-up. Choose a folder to save the simulation and choose an appropriate name, such as TutorialExample. The simulation is saved as a plain text file. Details about the file structure of this and other output files can be found later in the tutorial.

## Running Simulations

Click on the "Tools" menu in the menu bar and select "Simulation Manager" to open the simulation manager frame.




Click “Add Simulation” to create a simulation.

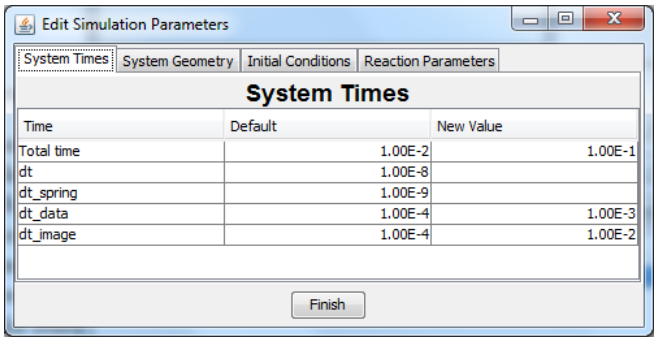


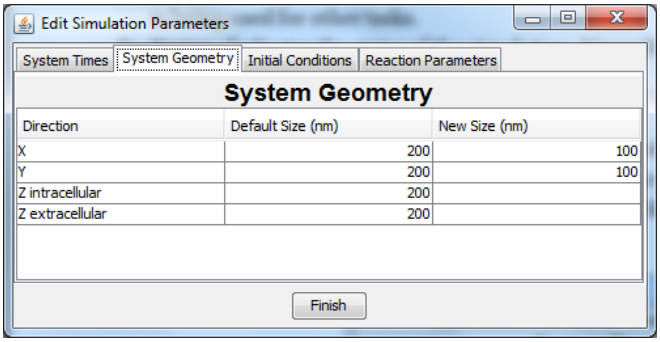

There are five columns specifying various features of the simulation.

1) Simulation Name: The name can be edited by clicking on it in the table.
2) Total runs: Many independent runs of the same system are usually required to calculate average system properties with statistical confidence.
3) Parallel: Indicate if the simulations should be run in parallel or sequentially.
4) Number of Simultaneous Runs: This option specifies the number of cores to use on a multiprocessor system. It should be kept below the total number of cores, and can be further restricted if the computer is being used for other tasks.
5) Status: Indicates the status of the simulation: Never Ran, Running, Aborted, or Finished.

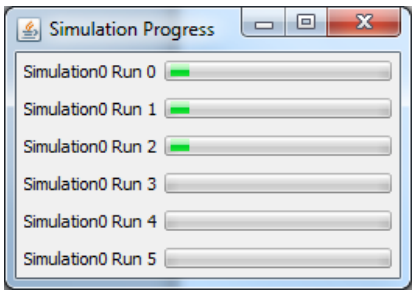
For this example we will run 6 simulations in parallel, using 3 cores.

The “Edit Simulation” button will bring up a window which allows the user to change the system geometry, times, initial conditions, and reaction parameters. In this case we will increase the total system time to 0.1 seconds, and set dt_data=0.001 s and dt_image=0.01 s. Furthermore, we decrease the system size in the x and y directions to 100 nm in each direction.




Finally, click "Run Simulation." A window will pop up indicating that the simulation engine is warming up. A progress bar is then displayed for each simulation to indicate its progress. (The tutorial system will probably take 10-20 minutes to run, depending on the processor speed and number of cores. This is a good opportunity to get up and go for a walk.)

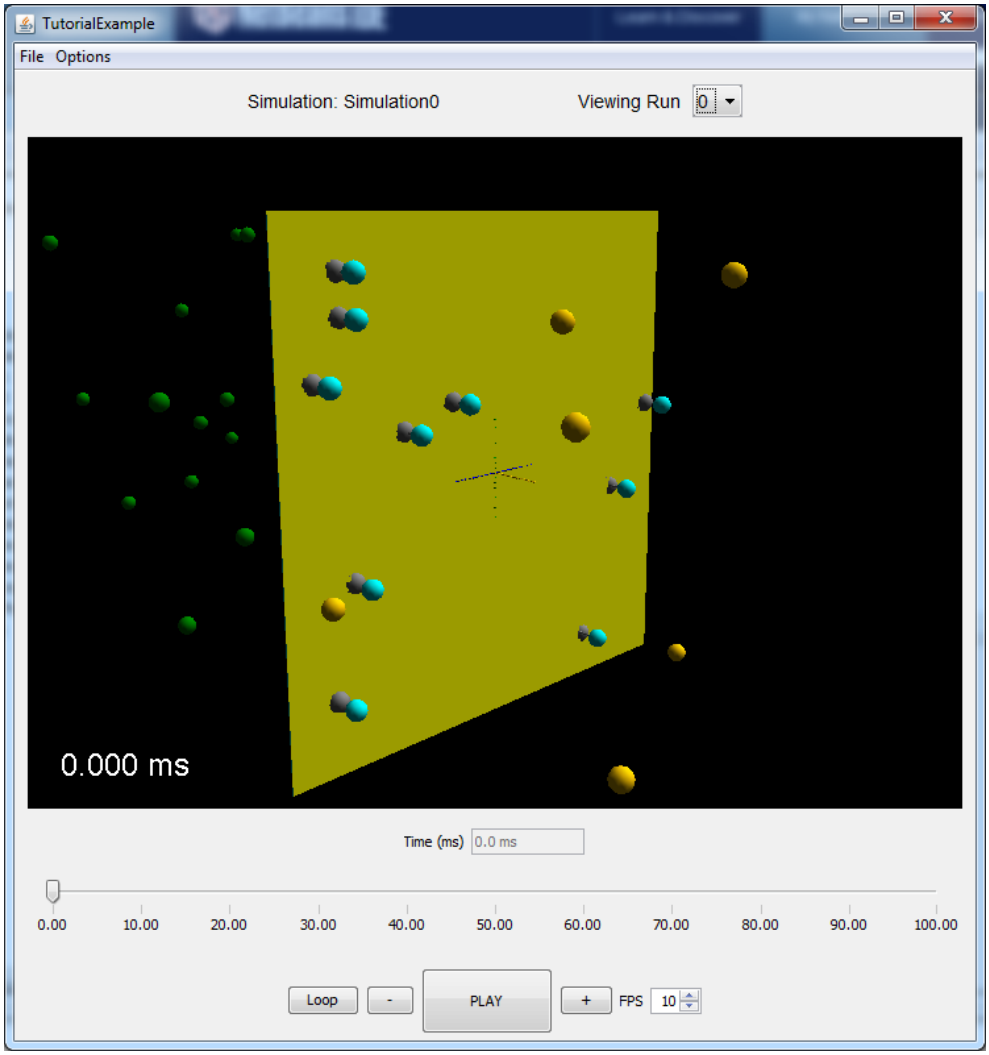



Note that the simulation status may not automatically change after the simulation finishes. This is a known bug that affects about half of all simulations. Minimizing the simulation management frame and then re-expanding it will refresh the table and display the correct status.

Note: There is currently no option to pause a simulation and restart it at a later time. This is a desirable feature and we plan to include this at a later time.

## The 3D Viewer

Click "View 3D Snapshots" after all simulations are finished to open the interactive viewer. The images for Run 0 are automatically loaded.



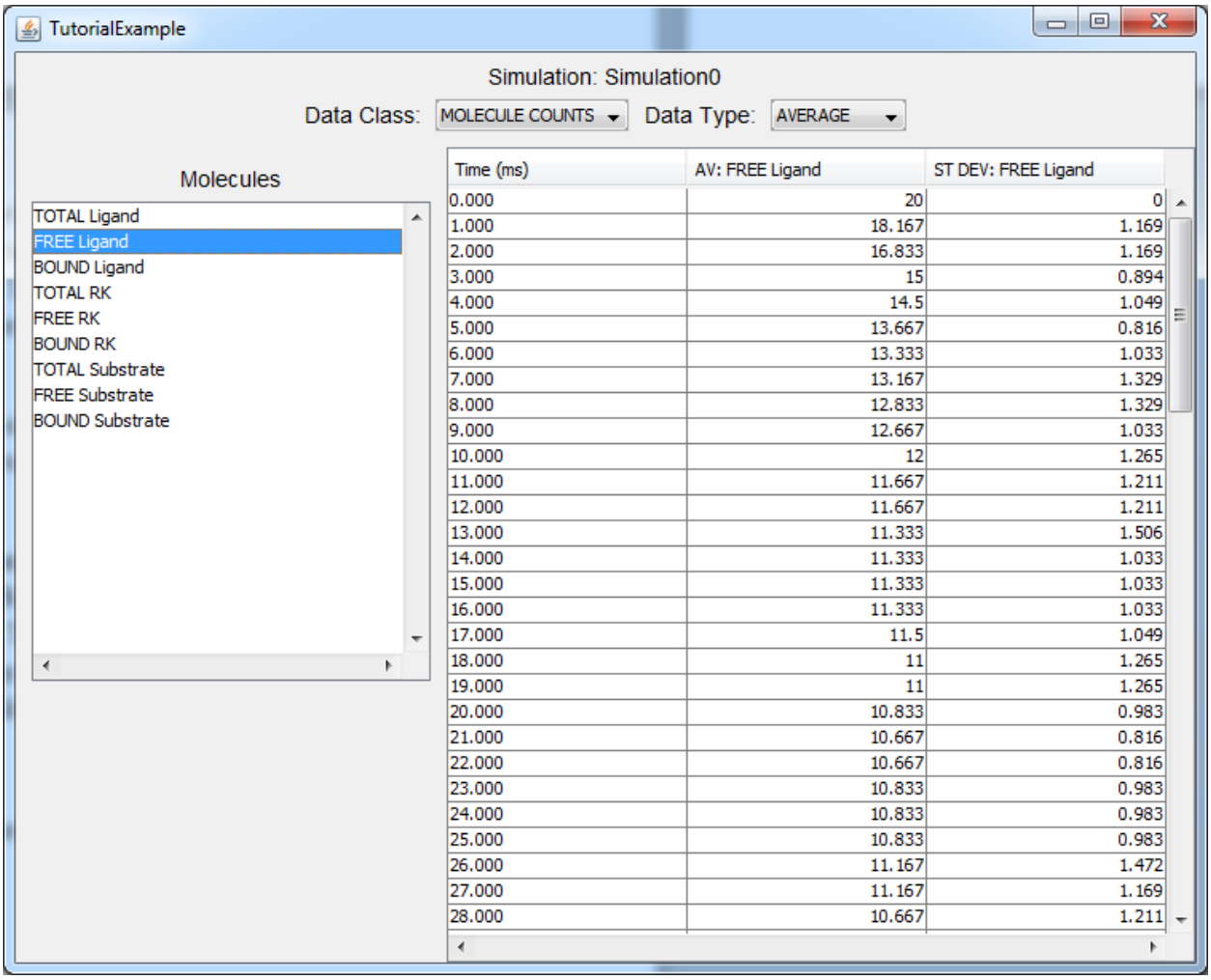

The viewer supports the full range of 3D interaction: click the left mouse button to rotate the scene, the right mouse button to translate the scene, and use the mouse wheel to zoom. Use the slider at the bottom to select a time point to view, or press play to view scenes sequentially. The animation speed can be controlled by changing the frames per second (FPS). A dropdown menu above the view panel allows the user from the different runs of the current simulation.

The file menu provides options for saving the current view in one of the popular image formats (PNG, JPEG, and GIF), as well as options of creating a video file from the current view point in either Quicktime or AVI formats.

The options menu allows the user to show or hide the time stamps, axes, and membrane, and provides options to configure the display of the latter two. The "Load Scenes to Buffer" option will construct all the 3D scenes and store them in memory instead of rendering them in real time. This will reduce the blinking which sometimes appears when rendering larger scenes, but it can require a large amount of memory which may not be available on all machines. Note that this option does not affect the quality of exported videos.

## Data Tables

Return to the simulation management frame and click the "View Data" button, which loads the data table frame.

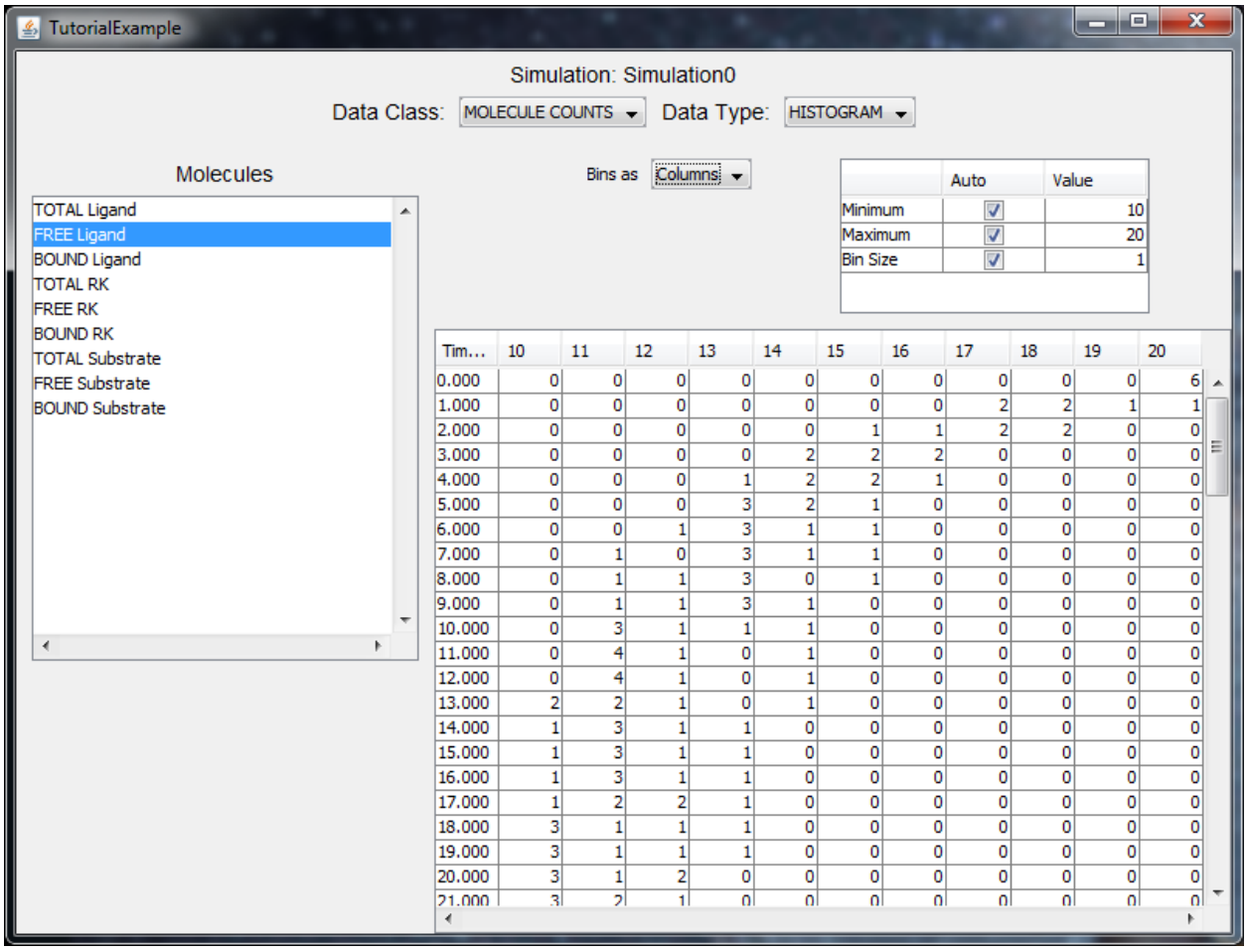



| Time (ms) | AV: FREE Ligand | ST DEV: FREE Ligand |
|---|---|---|
| 0.000 | 20 | 0 |
| 1.000 | 18.167 | 1.169 |
| 2.000 | 16.833 | 1.169 |
| 3.000 | 15 | 0.894 |
| 4.000 | 14.5 | 1.049 |
| 5.000 | 13.667 | 0.816 |
| 6.000 | 13.333 | 1.033 |
| 7.000 | 13.167 | 1.329 |
| 8.000 | 12.833 | 1.329 |
| 9.000 | 12.667 | 1.033 |
| 10.000 | 12 | 1.265 |
| 11.000 | 11.667 | 1.211 |
| 12.000 | 11.667 | 1.211 |
| 13.000 | 11.333 | 1.506 |
| 14.000 | 11.333 | 1.033 |
| 15.000 | 11.333 | 1.033 |
| 16.000 | 11.333 | 1.033 |
| 17.000 | 11.5 | 1.049 |
| 18.000 | 11 | 1.265 |
| 19.000 | 11 | 1.265 |
| 20.000 | 10.833 | 0.983 |
| 21.000 | 10.667 | 0.816 |
| 22.000 | 10.667 | 0.816 |
| 23.000 | 10.833 | 0.983 |
| 24.000 | 10.833 | 0.983 |
| 25.000 | 10.833 | 0.983 |
| 26.000 | 11.167 | 1.472 |
| 27.000 | 11.167 | 1.169 |
| 28.000 | 10.667 | 1.211 |

At the top are two drop down menus. The left drop down menu selects between different data types, most of which correspond to the data constructs discussed above: molecule counts, bond data, state data, site data, cluster data, and running times. The right drop down box selects between different data displays:

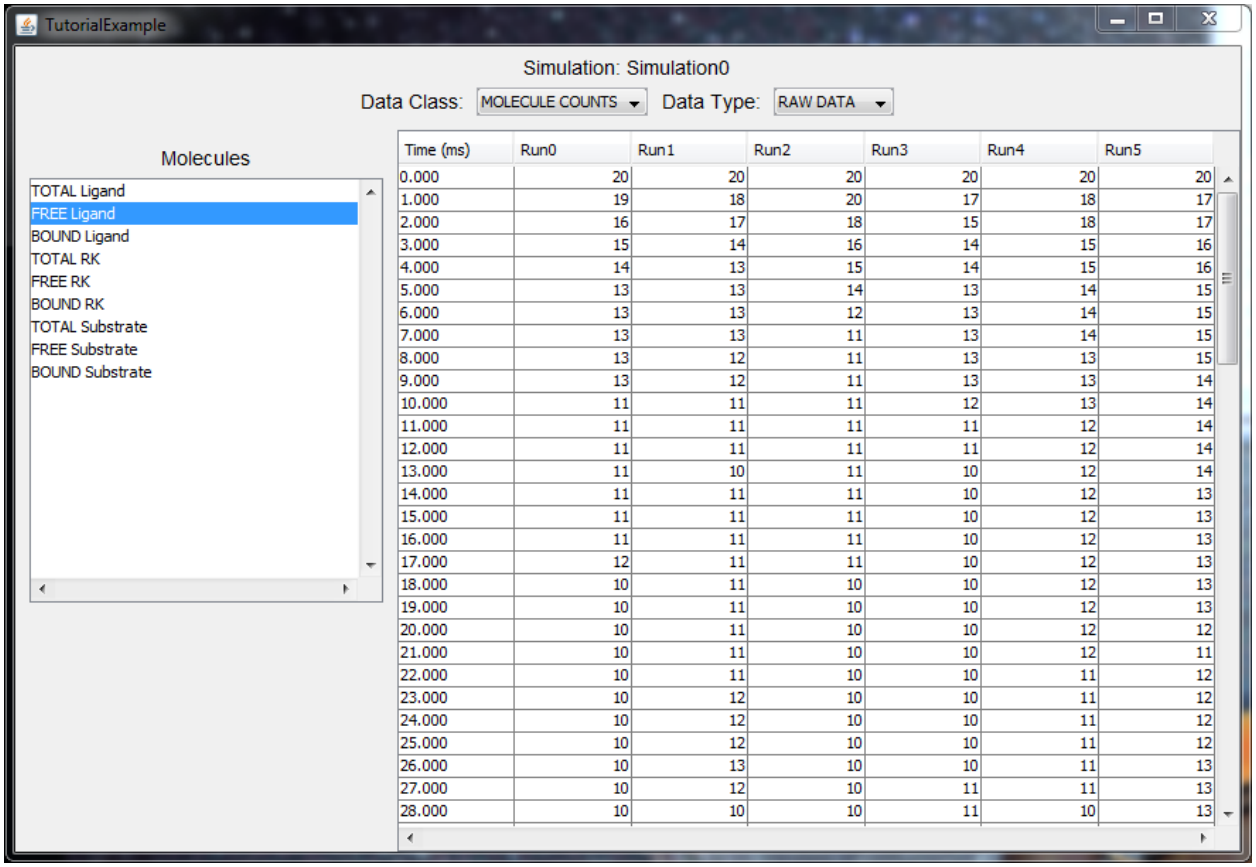
1) Average: Shows the average value and standard deviation as a function of time for the quantity selected in the list on the left-hand side. The average and standard deviation are calculated over the independent simulation runs. The image above shows the time course of free (unbound) ligand over the six runs.

2) Histogram: Provides values which can be used to construct a histogram with bins representing particle number and the counts representing number of run with that particle number, at each time point. For example, the image below gives values for a histogram of the free ligand count at each time point. At time 0 all 6 simulations have 20 free ligands, as expected because that was our initial condition. At 1 ms, 2 simulations have 17 free ligands, 2 have 18 free ligands, 1 has 19, and 1 still has 20 free ligands.



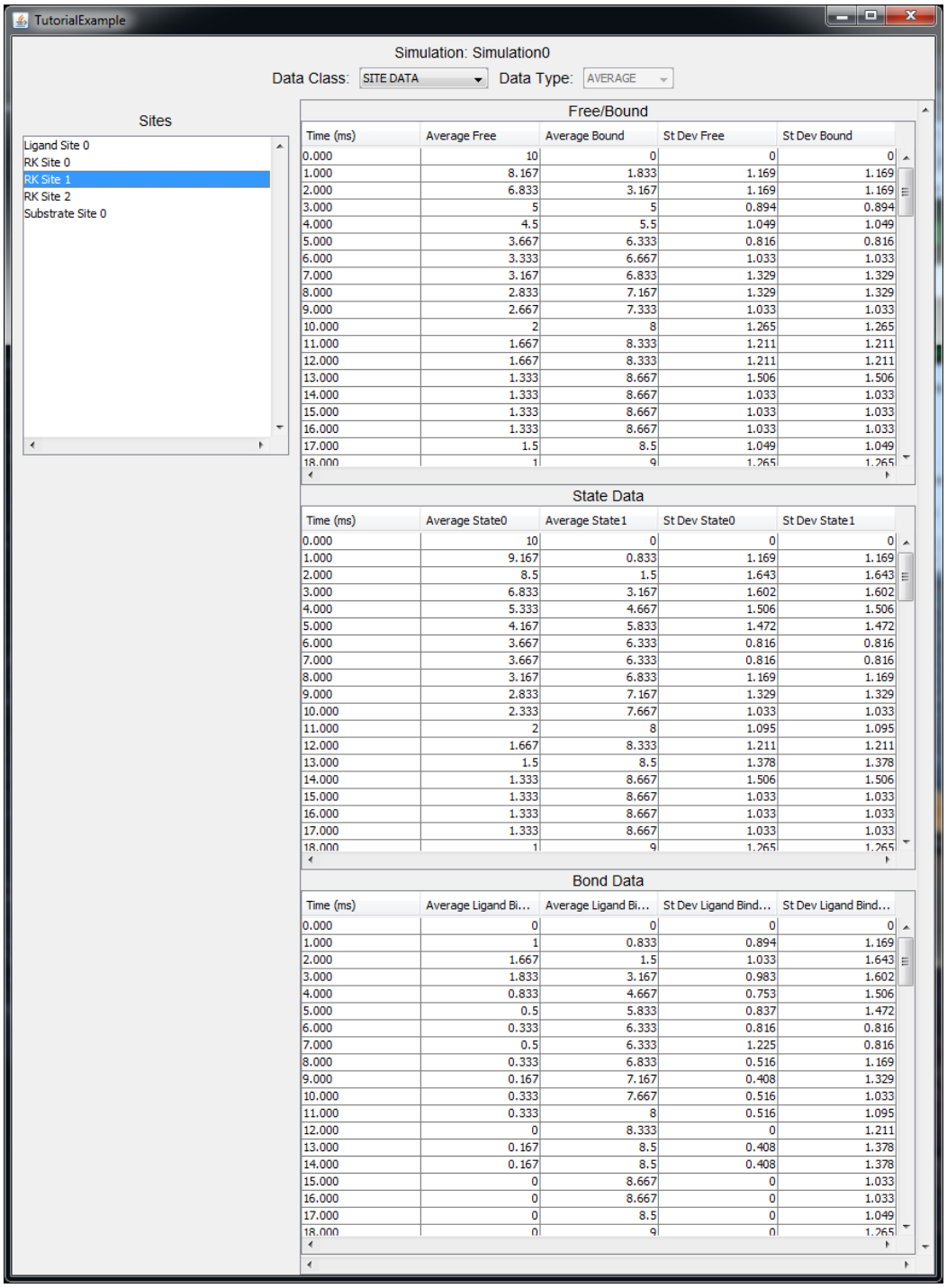

The dropdown box at the top selects whether the bins are displayed as the columns or the rows. The table to the right allows the user to customize the bin size and range used in the histogram.

3) Raw data: Provides the values for each run individually, as shown below.

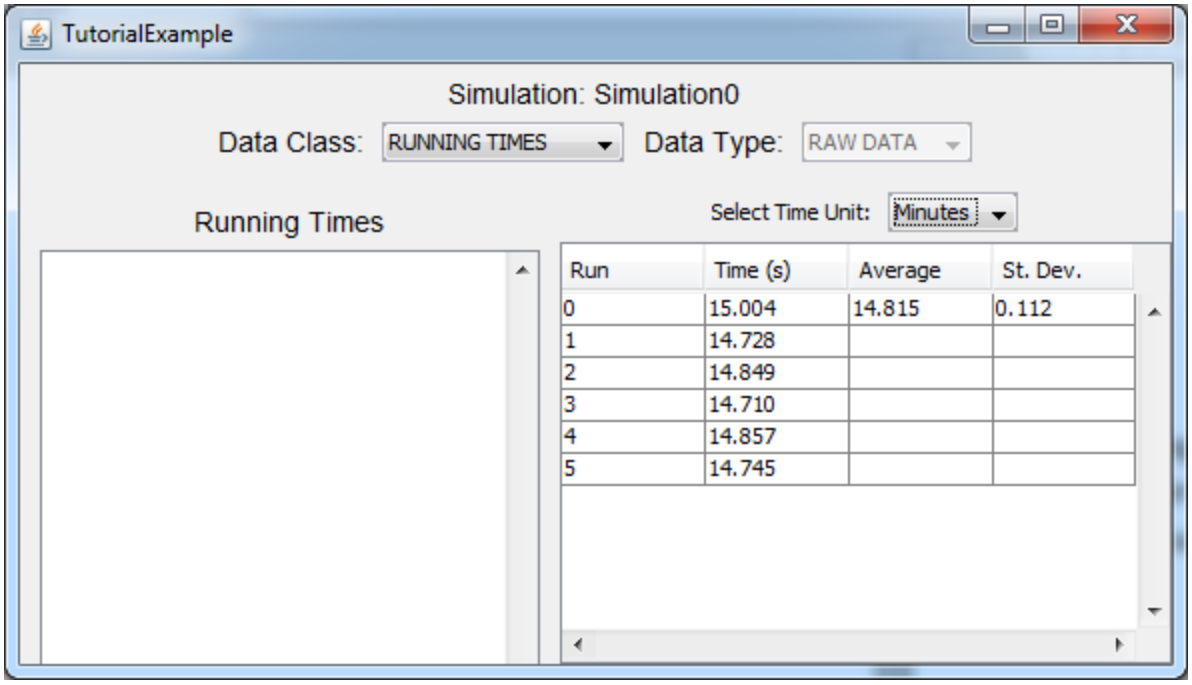



| Time (ms) | Run0 | Run1 | Run2 | Run3 | Run4 | Run5 |
|---|---|---|---|---|---|---|
| 0.000 | 20 | 20 | 20 | 20 | 20 | 20 |
| 1.000 | 19 | 18 | 20 | 17 | 18 | 17 |
| 2.000 | 16 | 17 | 18 | 15 | 18 | 17 |
| 3.000 | 15 | 14 | 16 | 14 | 15 | 16 |
| 4.000 | 14 | 13 | 15 | 14 | 15 | 16 |
| 5.000 | 13 | 13 | 14 | 13 | 14 | 15 |
| 6.000 | 13 | 13 | 12 | 13 | 14 | 15 |
| 7.000 | 13 | 13 | 11 | 13 | 14 | 15 |
| 8.000 | 13 | 12 | 11 | 13 | 13 | 15 |
| 9.000 | 13 | 12 | 11 | 13 | 13 | 14 |
| 10.000 | 11 | 11 | 11 | 12 | 13 | 14 |
| 11.000 | 11 | 11 | 11 | 11 | 12 | 14 |
| 12.000 | 11 | 11 | 11 | 11 | 12 | 14 |
| 13.000 | 11 | 10 | 11 | 10 | 12 | 14 |
| 14.000 | 11 | 11 | 11 | 10 | 12 | 13 |
| 15.000 | 11 | 11 | 11 | 10 | 12 | 13 |
| 16.000 | 11 | 11 | 11 | 10 | 12 | 13 |
| 17.000 | 12 | 11 | 11 | 10 | 12 | 13 |
| 18.000 | 10 | 11 | 10 | 10 | 12 | 13 |
| 19.000 | 10 | 11 | 10 | 10 | 12 | 13 |
| 20.000 | 10 | 11 | 10 | 10 | 12 | 12 |
| 21.000 | 10 | 11 | 10 | 10 | 12 | 11 |
| 22.000 | 10 | 11 | 10 | 10 | 11 | 12 |
| 23.000 | 10 | 12 | 10 | 10 | 11 | 12 |
| 24.000 | 10 | 12 | 10 | 10 | 11 | 12 |
| 25.000 | 10 | 12 | 10 | 10 | 11 | 12 |
| 26.000 | 10 | 13 | 10 | 10 | 11 | 13 |
| 27.000 | 10 | 12 | 10 | 11 | 11 | 13 |
| 28.000 | 10 | 10 | 10 | 11 | 10 | 13 |

The three options just described apply for molecule, bond, and state data. Site data is restricted to averages and standard deviations, but data is provided on the state and bond condition for each site in each molecule.




The running times panel shows the running time for each independent simulation, as well as the average and standard deviation. Options are provided to display the times in seconds, minutes, hours, or days. This data is especially useful for short, preliminary runs which are used to estimate the time required for longer simulations.




## Notes about Directory Structure and Output Files

Here we describe the directory structure and output files created while running simulations. Imagine we saved the tutorial system in a folder called "springsalad", and recall the system is saved as a plain text file, springsalad/TutorialExample.txt. The first time the simulation manager is opened a folder is created in the same directory, springsalad/TutorialExample_SIMULATIONS , which will hold all simulation data and results. (For brevity we will henceforth refer to this as simFolder.) When we make a simulation, say, Sim0, the program generates a plain text file in the simulations folder with the same name but appended with "_SIM". Thus, in this example we would find the file simFolder/Sim0_SIM.txt. This file contains all of the information required to run the simulation and stores the location of the simulation results when they are generated. When we run a simulation a new directory is created in the simulation folder, which will be named after the simulation with the word "_FOLDER" appended. In this case that folder is called Sim0_SIM_FOLDER. Inside this folder are four additional folders named data, images, videos, and viewer_files. The images and videos folder are created for convenience but are not used by the program. Inside the data folder are additional folders called "Run0", "Run1", …, each of which holds various data for an individual run of the simulation. In the data folder itself are a variety of files which store the various averages and histograms discussed in the previous section. The number of these files varies depending on which statistics are captured during each run. All are stored as csv files and all have descriptive names which should indicate the nature of the data held in the file. Inside the viewer_files folder will be a group of plain text files named "Sim0_SIM_VIEW_Run0.txt", "Sim0_SIM_VIEW_Run1.txt",…, each of which holds the information required by the 3D viewer for each run.

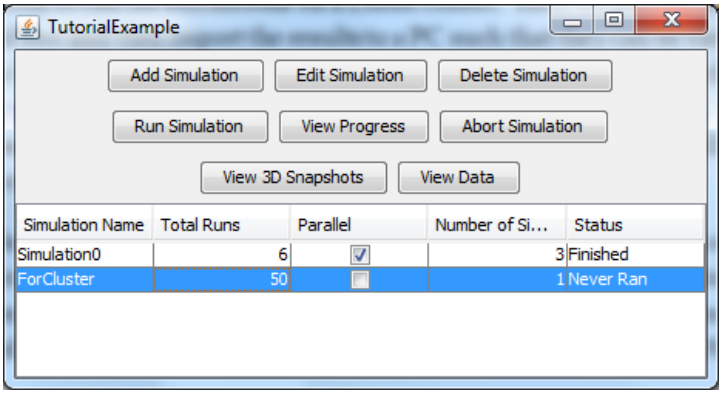
To summarize, the after running a simulation the following directory structure will be created:

/model_name.txt
/model_name_SIMULATIONS
sim_name_SIM.txt
/sim_name_SIM_FOLDER
/data
Various data files for averages and histograms.
/Run0
Various data files for run 0
/Run1
Various data files for run 1
…
/viewer_files
sim_name_SIM_VIEW_Run0.txt
sim_name_SIM_VIEW_Run1.txt
…
/images
/videos

## Running SpringSaLaD on the Command Line

The user may find the resources of a personal computer too limiting if they want to run tens or hundreds of independent simulations (for example, to generate highly accurate average values of the observables), and in this case it may be more convenient to run the simulations on a Linux cluster. Here we provide instructions to run simulations on the command line and then import the results to a PC such that they can be viewed with the SpringSaLaD GUI. We will use FileZilla (https://filezilla-project.org/) to transfer files between the PC and the cluster, and Cygwin (https://www.cygwin.com/) as a Windows-compatible Linux terminal shell. Both are freely available and relatively user friendly, although the user can use any terminal shell. We will use the model developed above for this example.

**Step 1**: Use the GUI to create the simulation you want to run on the cluster. Edit the simulation so that it has the correct parameters, and indicate the number of runs desired. All changes are saved automatically. **Do not run the simulation.** Here we create a simulation called "ForCluster," and change the parameters to match those of the simulation described above (total time 0.1 s, dt_data = 0.001 s, etc). We will make 50 runs.

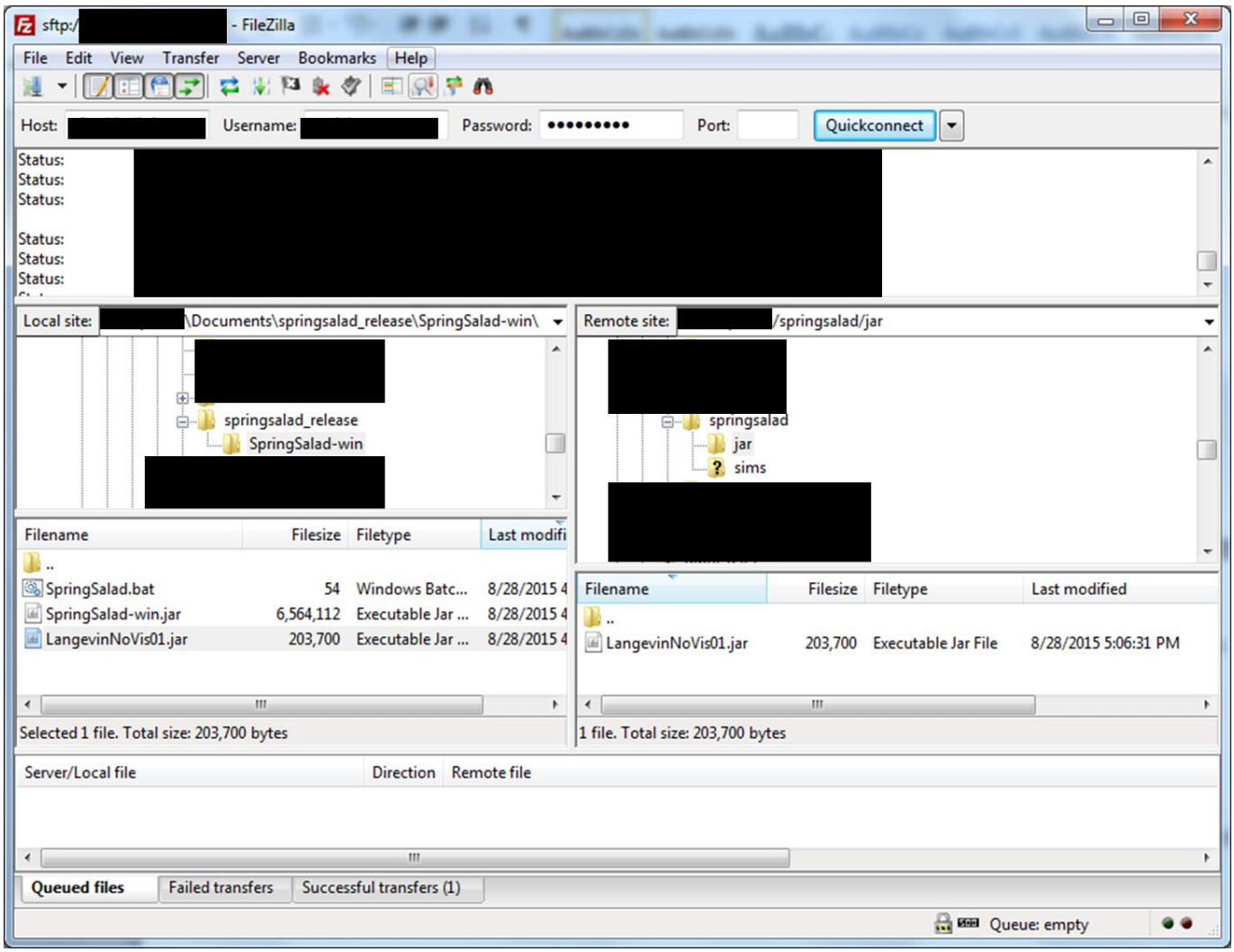



Now close SpringSaLaD.

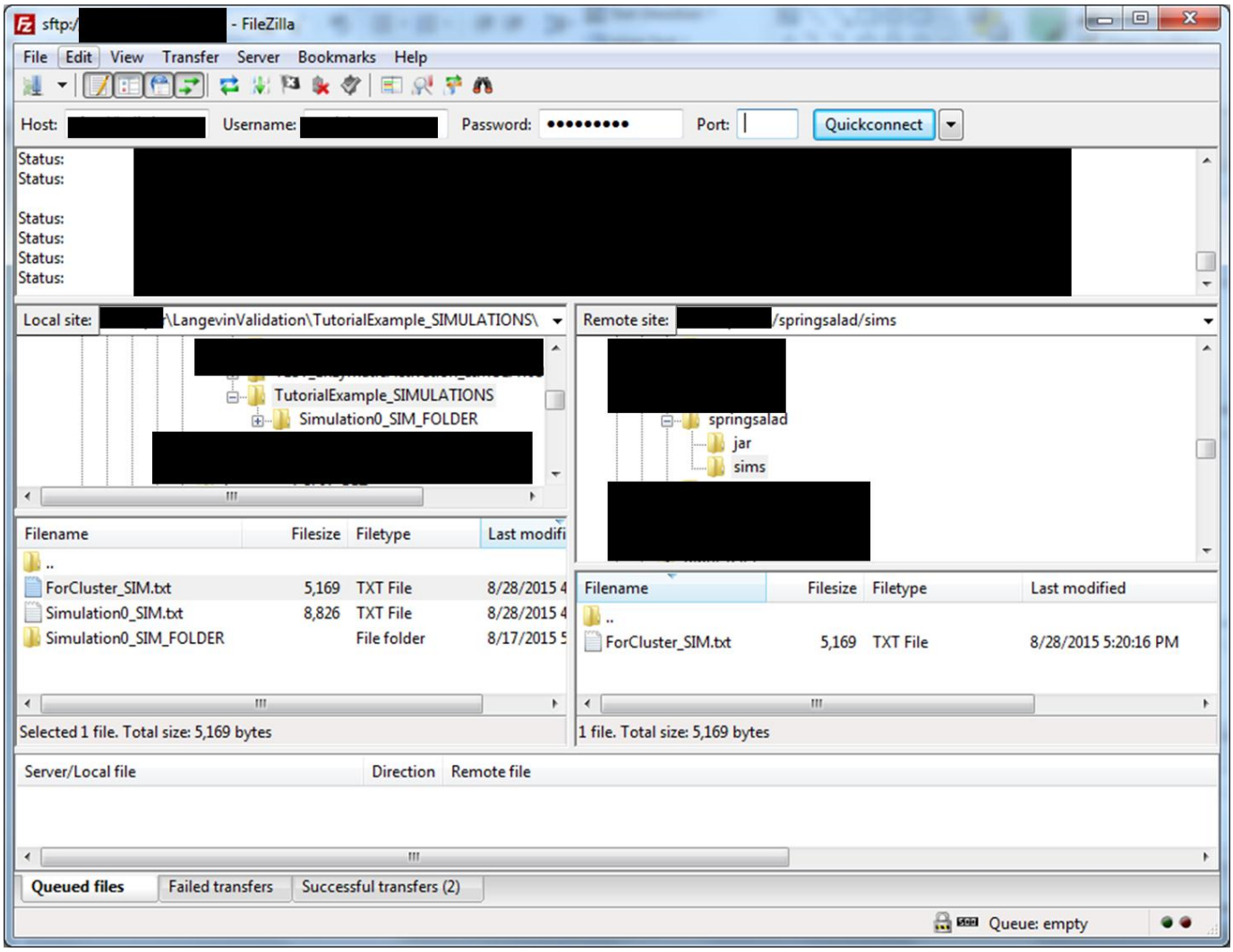
**Step 2**: We must now upload to the cluster both the input file and the jar file containing the simulation code. The jar file only needs to be uploaded once.

To begin, I suggest creating a folder in your home directory on the cluster called "springsalad", and three subfolders, "jar", "sims", and "stdout." The first holds the jar file, the second holds the simulation files, and the third will hold stdout files created and used by the simulations.

Return to the PC and navigate to the folder where SpringSaLaD was unzipped. In addition to SpringSalad-xxx.jar, you will see a second jar file, LangevinNoVis01.jar. (If you are curious, the name refers to the fact that these files contain no visualization components.) Upload this file to the jar subdirectory.




Now navigate to the folder where you saved the model. In addition to the model file, "TutorialExample.txt", you will see a folder named "TutorialExample_SIMULATIONS." (In general the folder will be named ModelName_SIMULATIONS.) Open this folder and you will see a file called "ForCluster_SIM.txt." This is the model input file. Upload it to the "sims" folder on the cluster.




**Step 3**: We create two files on the cluster, a generic script to submit the jar file, and a second script with a for loop to launch many runs at once. (A user familiar with the Linux environment and PBS job submission can use their own preferred scheme. The method here seems easiest to me, but I am by no means a Linux expert.)

Open Cygwin, ssh into the cluster, and navigate to the sims subfolder.


Use your favorite text editor, such as vim, to create a file called “runSpringSalad.pbs” in the sims folder, with the following content.

```
#!/bin/bash

#PBS -l walltime=120:00:00

#PBS -j oe

module unload java

module load java/1.8.0_45

java -Xms64m -Xmx1024m -jar /YOUR_PATH/springsalad/jar/LangevinNoVis01.jar "${fileName}" "${number}"
```

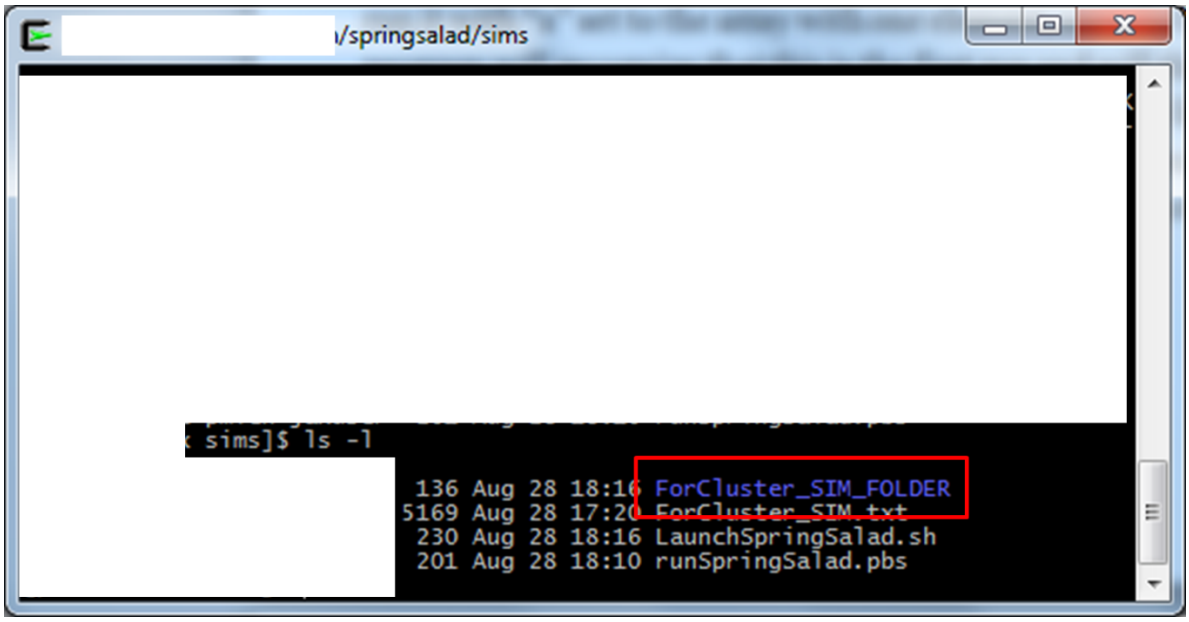
The version number of Java 1.8 is probably different on your machine. If your cluster does not use modules then you'll need another way to ensure Java 1.8 is loaded instead of an older version of Java.

Next, create a second file in the sims folder called "LaunchSpringSalad.sh", with the following content.

```
#!/bin/bash

# a=$(seq 1 1 49)

a=(0)

for i in ${a[@]}

do

qsub -N "SOME_NAME_${i}" -o "/YOUR_PATH/springsalad/stdout/SOME_NAME_${i}.stdout" \

-v "fileName=/YOUR_PATH/springsalad/sims/ForCluster_SIM.txt,number=${i}" runSpringSalad.pbs

done
```

Notice that the array "a" appears twice, once commented out. We will run this script twice. The first time we run it with "a" set to the array with one element, namely 0. This will launch the first run with index 0, and the program will recognize that this is the first run and will then create all relevant subdirectories. We will then comment out the "a=(0)" line, and uncomment the other line defining "a". Notice that this line defines an array which goes from 1 to 49. In general, this line should define an array which runs from 1 to (N-1) in increments of 1, where N is the total number of runs. The script is now run a second time to launch the remainder of the runs.

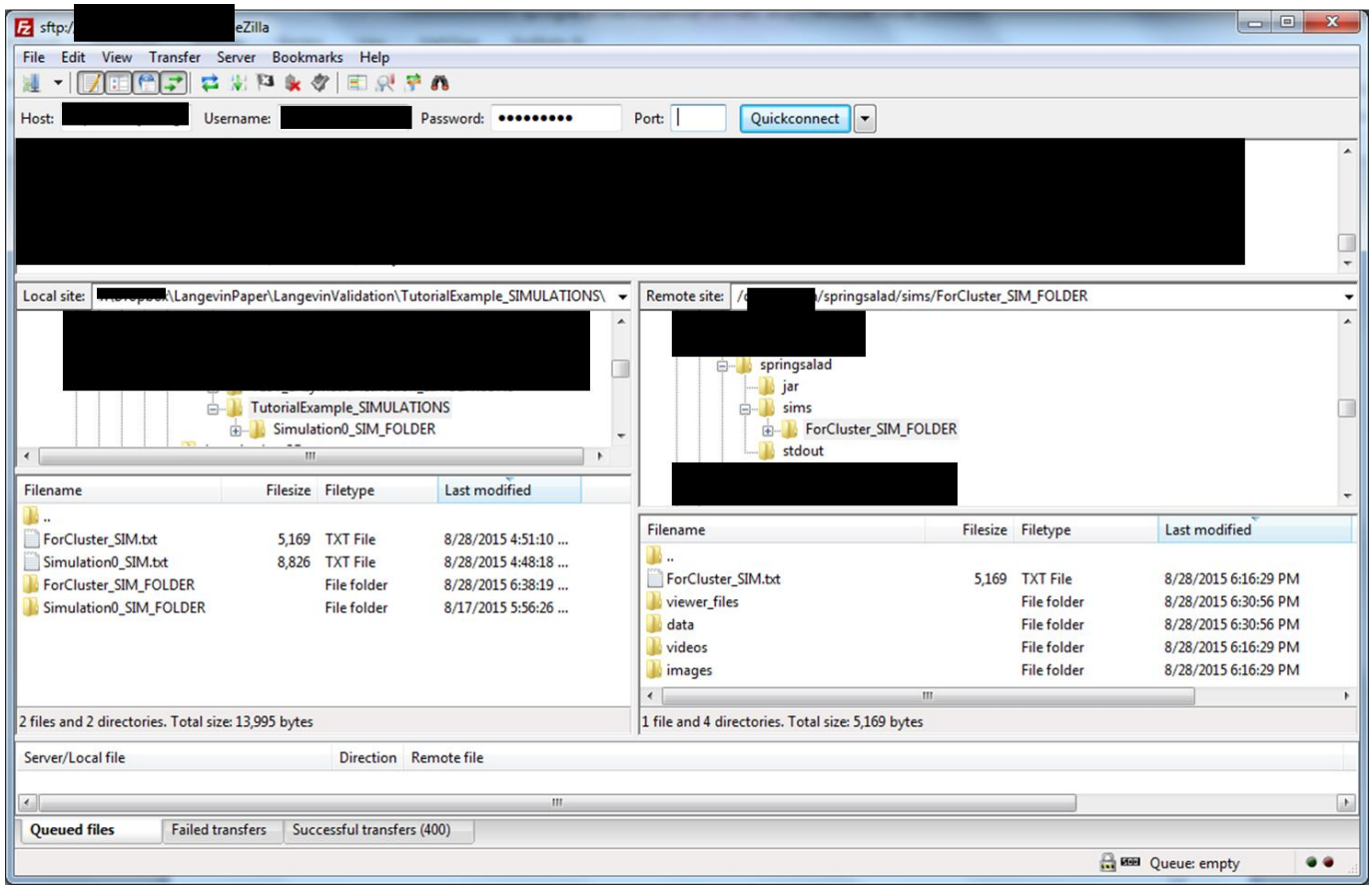
**Step 4**: Return to the command line and launch the first run by typing "bash LaunchSpringSalad.sh". If all goes well you should see a line indicating that your job has been submitted to the PBS queue. Wait about 30 seconds, then check to see that the folder "ForCluster_SIM_FOLDER" has been created in the "sims" directory.




Now, as described above, change the commenting in LaunchSpringSalad.sh to run the remainder of the clusters. Save the script and relaunch it by typing "bash LaunchSpringSalad.sh". You should see the remaining 49 runs submitted to the cluster.

**Step 5**: Wait for the simulations to finish. You can always use qstat to check the job status. Each run also writes its percentage complete to the corresponding stdout file in the stdout folder. As always, this is a great time to get away from the computer and go for a walk.

**Step 6**: Now download all of the data from the cluster to your local PC. To do so, simply download the entire "ForCluster_SIM_FOLDER" to the "TutorialExample_SIMULATIONS" folder (which contains the "ForCluster_SIM.txt" file). It may take a few minutes to download all of the files, depending on your connection speed.




**Step 7:** We now need to edit the simulation input file to notify SpringSaLaD that this job was run on the cluster. Open "ForCluster_SIM.txt" and scroll to the bottom, which should read as follows.

```
*** SIMULATION STATE ***

Runs: 50
Parallel: false
SimultaneousRuns: 1
Aborted: false
IsRunning: false
HasResults: false
RunOnCluster: false

*** PROCESSOR FILES ***

MoleculeAverages: 'null'
BondAverages: 'null'
StateAverages: 'null'
RunningTimes: 'null'

*** RAW DATA FILES ***

'null'

*** SITE DATA FILES ***

null
```

Change the two flags “HasResults” and “RunOnCluster” to true, so that the file looks as follows.

```
*** SIMULATION STATE ***

Runs: 50
Parallel: false
SimultaneousRuns: 1
Aborted: false
IsRunning: false
HasResults: true
RunOnCluster: true

*** PROCESSOR FILES ***

MoleculeAverages: 'null'
BondAverages: 'null'
StateAverages: 'null'
RunningTimes: 'null'

*** RAW DATA FILES ***

'null'

*** SITE DATA FILES ***

null
```

**Step 8**: Open SpringSaLaD, load the TutorialExample model, and navigate to the simulation window. The window may take a few seconds to open as SpringSaLaD computes the various observables and organizes the tables. You should see the “ForCluster” simulation as having results. At this point the visualization and data tables can be used as with any other simulation.