A Guide on Modelling Synapses with CellBlender and MCell

Jaron Lee

Abstract

This guide aims to provide an introduction to modelling with CellBlender and MCell. Upon completion the reader will have constructed a working model of transmitter release at a synaptic terminal.

1 Introduction

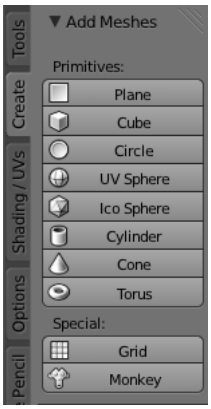
This guide details the steps taken to produce a model of a nerve terminal, as found at `https://github.com/kanilor/cellblender_project`.

It is expected that the user be familiar with introductory Blender and Python, and has completed the tutorials as per the README document on the website above.

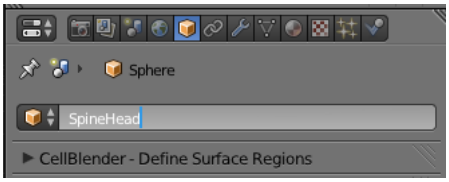
2 Pre- and Postsynaptic Geometry with Blender

2.1 Creating a Spine Head

1. Open Blender. In the '3D View' pane, delete the default object (shortcut: x)
2. Create a sphere. To do this at the centre, use snap (shortcut: Shift-S) and select 'Cursor to Centre'. Select the 'UV Sphere' option from the sidebar. Below a pane called 'Add Circle' appears; set 'Segments' and 'Rings' to 16, 'Radius to '0.7'.



3. Rename the sphere. Double-click the entry box below to change the default name to 'SpineHead'.



4. Change view to see the sphere from the 'Front' view. (shortcut: 1 on numpad). Note, it is **extremely** important that the user be in orthographic mode (shortcut: 5 on numpad) during Blender work.
5. Deselect the sphere (shortcut: a) and make it transparent (shortcut: z)
6. Select the vertices to be removed. First, switch from 'Object Mode' to 'Edit Mode'. Ensure that 'Edge select' is enabled. Then use box select (shortcut: b) to capture only the faces that make up the top half of the sphere. Delete these faces (shortcut: x) and select the 'Faces' option in the delete menu.

7. Close the opening. Select the topmost vertices (remaining in 'Edge select' mode) using box select (shortcut: b). Then, extrude (shortcut: e) and click 'Edges Only' under 'Extrude' in 'Mesh Tools'.

Set the extrude distance by pressing 0 and Enter to confirm. Scale the extrusion by pressing s, 0 and Enter to confirm. Select 'Remove Doubles' under 'Mesh Tools' to remove the duplicated vertices and reconnect the triangles. Blender should note that you remove 15 vertices as a result.

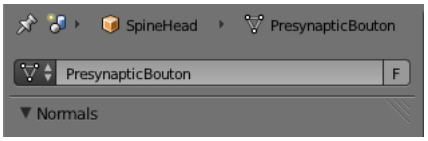
The object should now be closed by a flat top.

2.2 Creating a presynaptic bouton

1. Duplicate the *spine.blend* file as *bouton.blend*. in *bouton.blend* deselect all (shortcut: a).

2. Duplicate the spine head and rotate. First, switch to 'Front' view (shortcut: 1 on numpad). Duplicate (shortcut: Shift-d, Enter) and rotate 180 degrees (shortcut: r, 180, Enter). To separate the selection of the spine heads, press p and click 'Selection'.

3. Rename the item. Switch to 'Object Mode' (shortcut: Tab) and edit the name field to 'PresynapticBouton'.



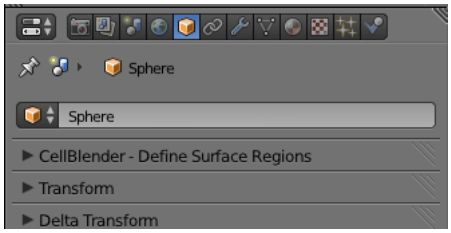
4. Shift and scale the bouton. Grab the object (shortcut: g), constrain movement to the z-axis (shortcut: z) and type 0.023, *Enter* to move the object.

2.3 Adding Axonal and Dendritic Extensions

1. Enter 'Edit Mode' (shortcut: Tab), and switch view to front (shortcut: 1 on numpad)
2. Zoom in on the top of the vertex. Select the vertex by right-clicking on it, ensuring that 'Vertex mode' selection is enabled. Perform two 'Select More' operations (shortcut: Control-Plus on numpad) until two rings are highlighted. Press x, select 'Faces' on the menu, *Enter* to confirm. Press b and select the vertices on the top edge using 'Box select'. Press e, click on 'Edges Only' in 'Tools' (left pane), then type z, 3.0, *Enter*. This should produce an axon on the presynaptic bouton.
3. Select the spine head. Hit *Tab* to enter 'Object Mode' and right click the spine head (the bottom half-sphere) to select it. *Tab* back into 'Edit Mode'
4. Create a cylindrical spine on the spine head. As in the earlier step, select the bottom vertex, perform two 'Select More' operations, press x and select 'Faces' on the 'Erase' menu. Hit b and select the vertices that line the hole in the bottom. Press e, select 'Edges Only' in 'Tools', press z, type -2.0, *Enter*.

2.4 Add Synaptic Vesicles with Regions for Calcium Binding

1. Create the first vesicle. First, ensure that the object is at the origin by moving it if necessary (shortcut: g). Move the cursor to the origin (shortcut: *Shift*-c). Ensure that 'Object Mode' is enabled. Under the 'Create' tab at left, select Ico-Sphere. A pane appears at lower left - change the default settings to 1 subdivision and a size of 0.023. A small sphere should appear at the origin.
2. It is necessary to rotate the vesicle. With the sphere selected, press r, x, 90, *Enter* to align the sphere with the x-axis. Enter the 'Outliner' pane at the top right, and edit the name of the sphere to 'Vesicle_1'.
3. Define the calcium binding region. With the vesicle still selected, change to 'Edit Mode' (shortcut: *Tab*) and triangulate the faces (shortcut: *Control-t*). First, select the 'Define Surface Region' tab which is under the 'Object' tab. Add a new region and name it 'vesicle_1_surf' (for the surface region of the vesicle). With the vesicle selected, click 'Assign'. We must now add the vesicle to the list of mesh objects to be included in the simulation. Under the 'Scene' tab, select 'Model Objects' and add the vesicle to this list. A green tick should appear.



4. Move the vesicle. Tab into 'Object Mode', press g to grab the vesicle, type x, -0.108 *Enter* to move the vesicle along the x-axis.

Hit g, 0.105, *Enter* to move it 0.105 units along the z-axis.

5. Duplicate the vesicle. Hit *Shift*-d, g, 0.216, *Enter* to move the vesicle the appropriate distance. This vesicle should appear as a new object under the 'Scene' pane - rename this vesicle to 'Vesicle_2' and repeat the steps for defining the surface region, this time naming it 'vesicle_2_surf'.

2.5 Defining region for voltage-gated calcium channels (VGCC)

1. Clip the existing mesh objects. Use *Alt-b* to draw a box around the presynaptic terminal and the vesicles. This will only show those items and hide the rest of the model. It is helpful to switch into 'Edit Mode', turn on wireframe (shortcut: z) and switch camera view to side (shortcut: 1,3 on numpad).

2. Switch to overhead view. Hit 7 on the numpad to look down from above on the vesicles.

3. Create the regions. We will create two small planes under each vesicle which will represent the VGCCs. Use the 'Plane' tool under 'Create' to define a plane. Adjust the settings using the 'Add Plane' pane at bottom left to position it underneath the vesicle. Duplicate it and copy it over to the other vesicle in the same fashion used to duplicate the vesicle.

4. Group the regions. With both panes selected, use *Ctrl-p* to merge the regions as one.

5. Assign the selected regions. In the bottom right pane under 'Objects', go to 'Define Surface Region', and label it 'vgcc_region'. We want this region to be an object in its own right for later selection and editing. With the region selected, press p and hit 'Selection'. Rename the object created to 'VGCC' in the top right pane. Remove the clipping box by pressing *Alt-b*.

2.6 Defining region for postsynaptic receptors

1. Hide the presynaptic bouton and the VGCC's. Click the eye button on the top right pane for the appropriate objects.

2. Define the postsynaptic receptor region. Right click on the postsynaptic dendrite and select the top face using the bounding box (shortcut: b). Use two 'Select Less' operations (shortcut: Control - minus on numpad) to deselect two outer rings.

3. Assign region. Under 'Objects' go to 'Define Surface Region' and label the receptor as 'postsynaptic_receptor'. In CellBlender, this region will now accessed as 'SpineHead[postsynaptic_receptor]'. Press *Control - t* to triangulate the faces, and then add it under 'Scene' in 'Model Objects'.

2.7 Adding Objects to CellBlender

The following objects should be under 'Model Objects':

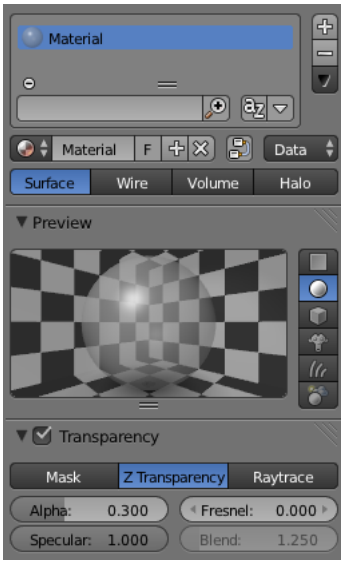
1. PresynapticBouton
2. SpineHead
3. VGCC
4. Vesicle_1
5. Vesicle_2

If not, simply select the model item, triangulate (shortcut: *Control*-t) and then with the item still selected add it under 'Model Objects'. This allows MCell to recognise the Blender object as part of the simulation.

2.8 Editing Model Object Appearance

Here are some steps which will improve the visualisation of our setup.

First, we want to make the presynaptic bouton semi-transparent so that we can see what is occuring inside the synapse. Under the 'Material' tab (look for the little red ball) create a new material by hitting the plus sign. Click the drag-down box and change the mode from 'Object' to 'Data'. Select the presynaptic bouton, enter 'Edit Mode' and assign the new material to the selected surface. Under 'Transparency', change the alpha level to 0.3. Exit 'Edit Mode'; the surface should be semi-transparent. If it is not, then under 'Objects', go to 'Display' and turn on 'Transparency'.



CellBlender will assign each molecule a certain colour automatically. If you want to change this, then go to the 'Outliner' (top right pane listing all the objects in the scene) and under 'molecules' go to the molecule of your choice. Click the small '+' to expand. For a molecule named 'a', you will see 'mol_a_pos' and 'mol_a_shaape'. Click on shape, and then navigate to 'Materials' (the little red ball) to change its colour.

2.9 Making Copies of the Model

At this stage, we have completed the model geometry - all the Blender objects have been set up for simulation. Make a copy of the file (File -¿ Save As) by saving it under another name. One copy will be used in

the next section (Phase I Simulation) while the other will be used in the section after (Phase II Simulation).

Phase I deals with the diffusion of calcium, the triggering of the SNAREpin mechanisms, and the fusion times of the vesicles, while Phase II deals with the release of neurotransmitter and the binding to receptors.

3 Simulation using the CellBlender Interface - Phase I

In this section, we simulate the fusion of the vesicles via a SNAREpin activation mechanism. We record the release times by creating tags for the 'fused' vesicles. Through an external Python script we record these times in preparation for the next phase.

3.1 Specification of Molecules

The following molecules will need to be added to our simulation under 'Scene' in 'Define Molecules'. The format is name of molecule, type of molecule, and diffusion constant.

1. Ca, Volume, 1e-6
2. VGCC_C, Surface, 0
3. VGCC_O, Surface, 0
4. CaBS, Surface, 0
5. CaBS_Ca, Surface, 0
6. TAG, Surface, 0

3.2 Define Reactions between Molecules

Given a set of molecules, we must define the interactions that occur when molecules meet. Under 'Define Reactions', enter the following (consists of reactants, products, and forward rate constant):

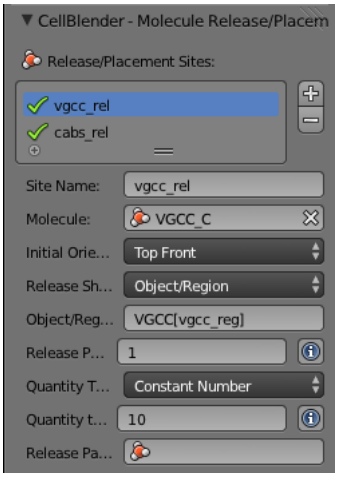
1. VGCC_C' $\rightarrow$ VGCC_O', 5e5
2. VGCC_O' $\rightarrow$ VGCC_C', 500
3. VGCC_O' $\rightarrow$ VGCC_O' + Ca', 1e3
4. Ca' + CaBS' $\rightarrow$ CaBS_Ca', 1e7
5. CaBS_Ca' + Ca' $\rightarrow$ TAG', 1e9

The apostrophes are used to indicate to CellBlender the geometry at which the reaction occurs.

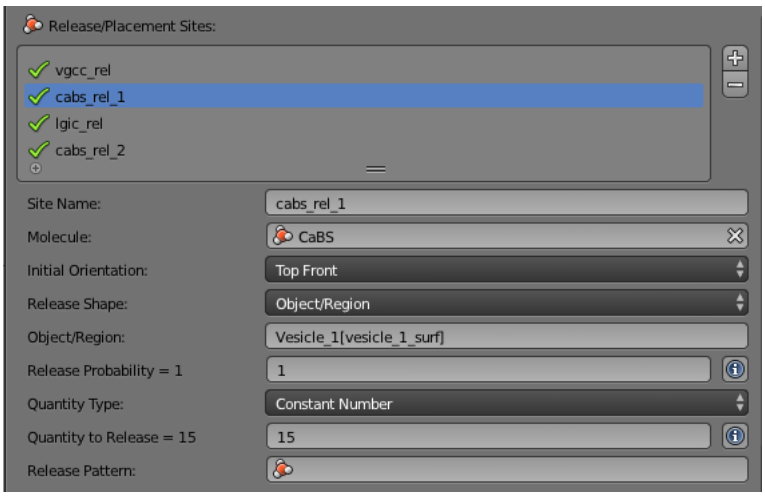
3.3 Adding Molecules to Regions

We only want the closed VGCC's and the calcium binding sites 'CaBS' to be present when the model is initialised. All other molecules will be generated by the reactions during simulation. The following items will be added under 'Molecule Release/Placement'.

Create a site called 'vgcc_rel'. This will be situated on the surface named 'VGCC[vgcc_reg]', and will represent the calcium channels. Enter the following settings.



Similarly, create a site called 'cabs_rel_1'. This will be situated on the 'Vesicle_1[vesicle_1_surf]' and will represent the calcium release sites. Enter the following settings.



Repeat again for 'cabs_rel_2', with identical settings.

3.4 Modifying and Defining Surface Classes

It may be desirable to define surface classes so that the surfaces in the model mimic real neuron surface structures better.

We can create surface classes under the 'Define Surface Classes' option. Create a surface class and add a property. The properties have three attributes - molecule, orientation, and type.

For this section, we will create a surface class called 'ca_absorb'. It will affect the Ca molecule, have Top/Front orientation, and have the type 'Absorb'. We then assign under 'Modify Surface Regions' this surface class to the presynaptic bouton - 'PresynapticBouton[presynaptic_shell]'.

Once the above instructions are complete, skip ahead and complete all the instructions under 'Preparing for Simulation' . Then, save the file and quit.

4 Simulating using the CellBlender Interface - Phase II

In this phase, we simulate the timed release of neurotransmitter molecules and their subsequent interaction with the receptor region on the spine head. The release times are obtained from the first phase of our simulation.

Complete the following instructions in the second copy of the Blender geometry.

4.1 Glial Cell Proxy

In the postsynaptic phase, neurotransmitter molecules will be dumped into the synaptic cleft and then will either contact a receptor or diffuse out of the cleft. In reality, these neurotransmitter molecules will be collected by glial cells and returned to neuron.

We can simulate this action by defining a cylinder around the whole synapse. Centre the cursor (shortcut: *Crtl*-b) and then create the cylinder. At the 'Add Cylinder' pane at bottom left, leave the radius at the default of 1.0, but ensure that for 'Cap Fill Type' the option 'None' is selected. Enter 'Edit Mode', select the whole cylinder, and then triangulate the object (shortcut: *Crtl*-t). Add the cylinder under 'Model Objects'. Define a surface region called 'surface' under 'Add Surface Regions'. Finally, name the cylinder 'GlialCells'.

4.2 Specification of Molecules

We want to add the following molecules to our simulation. The instruction format is once again molecule, molecule type, and rate constant.

1. NT, Volume, 5.3e-6
2. LGIC_C, Surface, 0
3. LGIC_O, Surface, 0

4.3 Define Reactions between Molecules

We want to specify interactions between our molecules. There is only one relevant reaction in this case.

1. NT' + LGIC_C' $\rightarrow$ LGIC_O', 4.6e6

4.4 Modifying and Defining Surface Classes

We want to create surface classes called 'nt_absorb' and 'nt_reflect'. They both affect the NT molecule, and have the types 'Absorb' and 'Reflective' respectively. We assign 'nt_absorb' to a single region - GlialCells[surface]. We assign 'nt_reflect' to two regions - PresynapticBouton[presynaptic_membrane] and SpineHead[receptor_region].

5 Preparing for Simulation

The instructions for this section are to be completed for both parts of the simulation.

Under 'Model Initialization', set the iterations to 10000 and the time step to 1e-6 (the amount of simulation time that passes between frames).

Under 'Reaction Output Settings', add an item for each molecule (should end up with 5 counts altogether). This setting will keep track of the number of molecules of each type currently existing in the model environment.

Ensure that the data is saved before continuing.

Refer to the IPython Notebook which contains the remainder of the instructions. These instructions are also integrated with scripts which perform backend tasks.

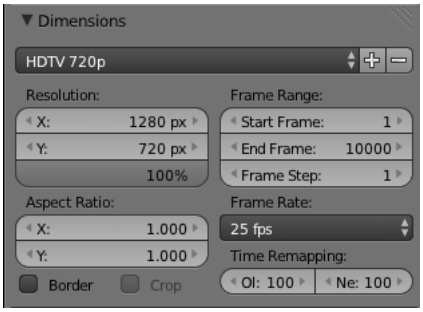
6 Rendering

Once the simulation is complete, the reader may wish to render their results to produce an animation of their model.

First, orientate the viewport to the view which is to be rendered.

Then, add a camera (Add → Camera). Right-click it to select, and then select the 'Align Active Camera to View' option (shortcut: Crtl Atl Numpad 0).

Next, click on the 'Render' pane. The following settings are the ones used to generate the animation files; however, the appropriate settings will depend on the computer used to generate the render. For example, it is not necessary to render all the frames - a subset can be selected if so desired.



Scroll down to 'Output' and select an appropriate folder. Scroll up to the top and hit 'Animation; then wait. This will take a while.

7 Troubleshooting

Simulation doesn't run

First, under 'Run Simulation' ensure that both output and error logs are being sent to file. Inspection of the error log in particular should reveal the cause of the issue. The most common error is a failure to properly assign a surface region to a Blender object - ensure that all regions are properly assigned before commencing a simulation run.