VND User's Guide

Version 1.14

NIH Biomedical Research Center for Macromolecular Modeling and Bioinformatics

Theoretical and Computational Biophysics Group
Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
405 N. Mathews
Urbana, IL 61801

`http://www.ks.uiuc.edu/Research/vnd/`

Description

The VND User's Guide describes how to run and use the neuronal visualization and analysis program VND. This guide documents the user interfaces displaying and graphically manipulating neurons and describes the methods of interacting with neuronal system animations.

VND development is supported by the National Institutes of Health grant numbers NIH P41-GM104601 & NIH 1U24NS124001 & NIH R24GM145965-03.

Contents

Chapter 1

Introduction

VND [1] is a neuroscience-specialized adaptation and repackaging of the popular molecular visualization program VMD. VND supports the SONATA neuronal network modeling file format and can create visualizations of the structure and dynamics of neuronal simulations. Key features of VND include:

- Support for Linux and macOS (Intel and ARM/Apple Silicon)
- Support for multicore processors.
- Support for GPU-accelerated computation.
- WIP support for a Python ecosystem to share data between BMTK and VND

`https://tcbg.illinois.edu/Research/vnd/`

1.0.1 How to cite us

The authors request that any published work or images created using VMD include the following reference:

Humphrey, W., Dalke, A. and Schulten, K., "VMD - Visual Molecular Dynamics" *J. Molec. Graphics* ***1996****, 14.1*, 33-38.

VMD has been developed by the Theoretical and Computational Biophysics Group at the Beckman Institute for Advanced Science and Technology of the University of Illinois Urbana-Champaign. This work is supported by the National Institutes of Health under grant numbers NIH 9P41GM104601, 5R01GM098243-02 and SSG 1U24NS124001.

1.1 Downloading VND and the Hardware and Software Requirements

Before staring the tutorial you need to download the current version of VND and the HDF5 (Hierarchical Data Format 5) binaries on your machine. This tutorial requires VND release 1.14 or later and requires HDF5 1.14. VND supports Linux and macOS, and can be obtained from the VND homepage shown below.

Link for VND:
`http://www.ks.uiuc.edu/Research/vnd/vnd-1.9.4/files/alpha/`
Download VND version 1.14 for your operating system.

Link for HDF5:
`https://support.hdfgroup.org/releases/hdf5/v1_14/v1_14_5/downloads/index.html/`
Download version 1.14 or greater. See instructions below.

Installing HDF5 on MacOS

1. Pick the correct binary, it is bundled as a dmg labeled: **hdf5-1.14.5-macos14_clang.dmg**
2. Download the dmg and double click on it
3. Move the 'HDF_Group' folder to 'Applications'
4. Set local environment variables so that VND can locate the folder package containing the HDF5 tools

   If tcsh:

   ```
   setenv PATH /Applications/HDF_Group/HDF5/1.14.5/bin/h5dump:$PATH
   ```

   If bash:

   ```
   export PATH="/Applications/HDF_Group/HDF5/1.14.5/bin/h5dump:$PATH"
   ```

Installing HDF5 on Linux

1. Picking the right binary download
   It is suggested to download the latest binary version compatible with Linux. **'hdf5-1.14.5-ubuntu-2204_gcc.tar.gz'**. There are other options such as downloading the RPM and/or deb package. For simplicity we are using the binary built by GCC.
2. Make a permanent directory for HDF5 binaries to reside in

   ```
   mkdir /home/your_username/hdf5_binaries
   ```

3. Move the zipped install to the hdf5_binaries directory

   ```
   mv ~/Downloads/hdf5-1.14.5-ubuntu-2204_gcc.tar.gz/
   home/your_username/hdf5_binaries
   ```

4. Uncompress by using built in Linux tools

```
cd /home/your_username/hdf5_binaries
gunzip hdf5-1.14.5-ubuntu-2204_gcc.tar.gz
tar xvf hdf5-1.14.5-ubuntu-2204_gcc.tar
cd hdf5
```

5. Uncompress HDF5-1.14.5-Linux.tar.gz

```
gunzip HDF5-1.14.5-Linux.tar.gz
tar xvf HDF5-1.14.5-Linux.tar
```

6. Verify your present working directory and copy this location to your clipboard. It should be **/home/your_username/hdf5_binaries/**

```
pwd
```

7. Set local environment variables so that VND can locate the folder package containing the HDF5 tools

   If tcsh:

```
setenv PATH /home/your_username/hdf5_binaries/hdf5/
HDF5-1.14.5-Linux/HDF_Group/HDF5/1.14.5/bin/h5dump:$PATH
```

   If bash:

```
export PATH="/home/your_username/hdf5_binaries/hdf5/
HDF5-1.14.5-Linux/HDF_Group/HDF5/1.14.5/bin/h5dump:$PATH"
```

Installing VND on Linux

Requires several terminal commands This section will show you how to install VND on Linux without sudo/root.

1. Pick a directory to install and replace the quotes below

```
cd "/home/your_username"
```

2. Make the test directory

```
mkdir /home/your_username/vnd-test-install
cd \$!
```

3. Make the lib and bin directory

```
mkdir lib
mkdir bin
```

4. Create a new temporary directory

```
mkdir ~/my-temp/
cd ~/my-temp/
```

5. Make the install directory

```
mkdir vnd-install
cd vnd-install
```

6. Move the zipped install to the vnd-install directory. Replace file name with the version downloaded on your system

```
mv ~/Downloads/vnd-1.9.4a53p7b.LINUXAMD64.opengl.tar.gz
~/my-temp/vnd-install
```

7. Uncompress by using built in Linux gunzip/tar

```
cd ~/my-temp/vnd-installer
gunzip vnd-1.9.4a53p7b.LINUXAMD64.opengl.tar.gz
tar xvf vnd-1.9.4a53p7b.LINUXAMD64.opengl.tar
cd vnd-1.9.4a53
```

8. Change paths in the configure script Change $install_bin_dir and $install_library_dir Directory where VMD startup script is installed, should be in users' paths.

```
$install_bin_dir="/home/your_username/vnd-test-install/bin";
```

   Directory where VMD files and executables are installed:

```
$install_library_dir="/home/your_username
/vnd-test-install/lib/$install_name";
```

9. Verify correct present working directory that it is:

```
~/my-temp/vnd-installer/vnd-1.9.4a53
```

10. Run the configure script and install

```
./configure
cd src
make install
```

11. Set local environment variables
    If tcsh:

```
setenv PATH /home/your_username/vnd-test-install/bin:$PATH
```

    if bash:

```
export PATH="/home/your_username/vnd-test-install/bin:$PATH"
```

12. Verify & Launch VND

```
which vnd
vnd
```

Installing VND on ARM MacOS

1. Drag and drop the .dmg file onto the desktop and/or Applications folder
2. Double click to run the executable

Installing VND on Intel MacOS

1. Drag and drop the .dmg file onto the desktop and/or Applications folder
2. Double click to run the executable
3. If prompted with a 'startup command' permission request, accept the request and give VND software to read & write to that directory

1.2 Quick Tutorial Topics and Files

The tutorial contains several sections, each section acts as an independent tutorial for a specific topic, with the section layout as shown in Contents. We suggest that readers with no prior VND experience work through the sections in the order they are presented. Readers already familiar with the basics of VND may selectively pursue sections of their interest. Several files have been prepared to accompany this tutorial. You will need to download these files at `https://www.tcbg.illinois.edu/Training/Tutorials/vnd/`.

1.3 Loading neuronal files for the first time

1.3.1 Rapid Introduction of VND

For those who would like to jump into quick visualization and analysis. The files can be downloaded from the Allen Institute's Tutorial Page.

1. Download example systems from TCBG website Go to: `https://www.ks.uiuc.edu/Training/Tutorials/vnd/vnd-tutorial-files.zip` to download 300_cells system
2. Launch VND
3. Load the file Go to the top right of VND and go to File -¿ Open file and open the **circuit_config.json** file `../vnd-tutorial/300_cells/circuit_config.json`
4. View & Interact with the 300_cells system in the OpenGL window by using your mouse/-trackpad

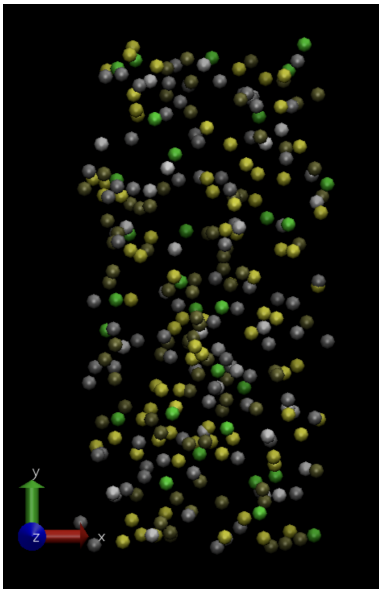

Figure 1.1: VND OpenGL window showing 300_cell example model. Default representation: selection 'all', style 'soma', coloring method 'Type' and material 'Opaque'.

1.3.2 BMTK and SONATA

The Brain Modeling Toolkit (BMTK) is a Python-based software package for building, simulating and analyzing large-scale neural network models. It supports the building and simulation of models of varying levels-of-resolution; from multi-compartment biophysically detailed networks, to point-neuron models, to filter-based models, and even population-level firing rate models. For more details visit `https://alleninstitute.github.io/bmtk/`.

The SONATA Data Format is a Scalable Open Data Format for multiscale neuronal network models and simulation output, jointly developed by the Allen Institute and the Blue Brain Project, with contributions from other institutes. The SONATA Data Format provides:

- A network representation format used to capture aspects of a network model including location and properties of individual cells and connectivity between cells.
- Standards for storing simulation recordings like spike-trains, ion and channel traces, and extracellular recordings.
- A JSON-based configuration that capture all properties and files required to run a full network simulation.

SONATA Network Model Overview

Modeling and simulation of a large-scale brain circuits is a complex process requiring a large number of disparate data structures and parameters. A single simulation may require hundreds of files to represent and capture elements like the network connectivity, the morphology and properties of individual cells, and the recorded results from each actual simulation. To alleviate this issue, the SONATA format defines a JSON-based configuration file for each simulation that points to all files used in the simulation along with any other relevant parameters.

The SONATA configuration file is utilized by VND to automatically locate the necessary files for displaying a network and any simulation results. The configuration files can also be easily opened in most text editors and easily read by users, which different aspects of each simulation segmented into different subsections of the JSON file:

- **network:** contains link the different SONATA circuit files used to specify the location, connectivity, types, and properties of cells within the simulation.
- **components:** contains location of any auxilary files required for instantiating and running the simulation, like any SWC or NeuroML files used in representing morphology of individual cells.
- **reports and outputs**: used to specify any recordings, statistics, or logs gathered during the simulation and where to locate them. For example the location of any SONATA spike-trains and/or membrane potential recording files from the simulation.
- **inputs:** The nature of any stimuli and external files used during simulation, such as spike-train files used for synaptic inputs, or the wave-forms used by current-clamps.

Chapter 2

VND Hardware and Software Requirements

2.1 Basic Hardware and Software Requirements

VND can be ran on the latest Linux distros and on Intel, M1, and M2 MacOS machines. VND's core utilizes demanding graphics-accelerators to power it's OpenGL display [1, 2, 3, 4, 5, 6].

2.1.1 Software

VND has been tested to run smoothly on the most updated versions of Linux and MacOS (14.5) VND is available as a pre-built binary that ships the Tcl/Tk 8.6.14, HDF5 library, Tachyon raytracing, FLTK, PThreads, and OpenGL [7]. So you don't need to worry about having these installed.

2.1.2 Copyright

2.1.3 Downloading

The current stable build can be accessed via the tcbg.illinois.edu website at the VND Test Release Page.

Currently the latest version is: VND Release 1.14: rotation selections and attribute display, fixes for rotation input including per-node override, improved prototype alignment tool. Historical developmental test versions can also be seen and downloaded.

Chapter 3

Loading systems in VND

The File menu is the only method of loading neuronal systems, spikes, compartment data, and/or polygon files into VND.

3.1 General overview of neuronal file formats

VND can load and read popular file formats used in representing neuronal such as:

SONATA configuration file, a human-readable **JSON** file containing properties and file paths required to instantiate a network model and simulation.

HDF5 and CSV files containing in-vivo or in-silico recordings of network, like captured-spike trains or membrane voltage traces.

SWC files, a popular standard for representing the morphology of individual cells.

Neurodata without Borders (NWB) files, a popular format adopted by many institutions for sharing experimental data.

3.1.1 Quickly view a system rapidly

When loading a certain type of file, one generally loads a circuit or JSON file. This file consists of the main neuronal system. The circuit file points to several auxiliary files within the local network directory. Additional files may introduce connections between the individual neurons and/or timeseries data representing voltage spiking often in the H5 format.

3.2 What occurs during file loading?

When loading a circuit file (JSON) the coordinates of the neurons are read.

```
about to: exec h5dump -n ../300_cells/network/virt_nodes.ht5
Results for./300_cells/network/virt_nodes.h5: {
File Contents {
group /
```

```
group /nodes/
group /nodes/virt/
group /nodes/virt/node_group_id
group /nodes/virt/node_group_index
group /nodes/virt/node_id
group /nodes/virt/node_type_id
}
}
Done with rpn_input_queue. length of rpn_stack is 1. length of outlist is 300
length of myNodeIdList is 300
starting    proto_show_nodes_from_list_soma_only.  colorMethod is Type
about to display 300 nodes

updateRepMenu: repdetails = 0 true 0 soma Type Opaque all 1 300 False 1.0 6
updateRepMenu: repselected = 0, curselection = 0
```

Chapter 4

Basic User interface and controls

4.1 Using the Mouse in the Graphics Window

The graphics window is labeled VND 1.9.4a57 OpenGL Display and contains a view of the neurons and other elements in the scene. When the mouse is in the graphics display window, it may be used to perform the following actions such as:

- Rotate, translate, or scale the displayed neurons
- Translate and rotate a set of atoms
- Move the lights

4.1.1 Mouse Modes

The mouse is in one of several *modes* at any time; the current mouse mode determines the effect of pressing and releasing mouse buttons or the mouse wheel while the mouse is in the graphics window. Each mouse mode, except the lights mode (see below), sets the mouse cursor to a characteristic shape. The mouse mode can be use by the hotkey interface options shown below.

The available mouse modes are as follows:

- **Rotate Mode** (hot key 'r')
  When the mouse is in rotate mode, holding the left mouse button down and moving the mouse rotates the neurons about axes parallel to the screen, in a 'virtual trackball' behavior. To get a rotation around the axes coming out of the screen (the 'z' axis), hold the middle button down and move the mouse left or right.

  You can leave neurons rotating without continuously moving the mouse. Start the neuron moving with the mouse, as above, then release the mouse button before you stop moving the mouse. With some practice it becomes easy to impart a slight spin on the neuron, or whirl it about madly. To stop the rotation, either press and hold the left mouse button down until the neuron stops moving, or select 'Stop Rotation' in the Mouse menu. Also, pressing the rotation hot key `r` or any of the other mouse mode hot keys causes rotation to stop.

- **Translate Mode** (hot key 't')
  When the mouse is in translate mode, holding the left button down allows you to move the neurons parallel to the screen plane (left, right, up, and down). To move the neuron

towards or away from you, hold the middle button down and move the mouse right or left, respectively.

- **Scale Mode** (hot key 's')
  Pressing either the left or middle button down and moving to the right enlarges the neurons, and moving the mouse left shrinks them. The difference is that the middle button scales faster than the left button. Scaling can also be accomplished with the mouse wheel (irrespective of the current mode setting) on computers equipped with an appropriate mouse.

- **Move Light**
  VND provides four directional lights to illuminate the neuronal scene. The lights provide diffuse lighting and specular highlights and help the user perceive surface shape in rendered objects. You can use the mouse to rotate each of the light source directions to a new position. If the light isn't on, moving it will not affect the displayed image. To turn a light on or off, use the Lights item within the Mouse menu.

4.1.2 Hot Keys

When the mouse is in the graphics window, many commands are accessible via programmable hot keys. Hot keys allow you to do things like change mouse modes or advance the animation by a frame by simply pressing a key. There are a number of predefined hot keys, as listed in tables 4.1, 4.2, 4.3, and 4.4. We are working on enabling Hot Key customization available to the end user.

Hot Key	Command	Purpose
r, R	mouse mode 0 0	enter rotate mode; stop rotation
t, T	mouse mode 1 0	enter translate mode
s, S	mouse mode 2 0	enter scaling mode
0	mouse mode 4 0	query item
c	mouse mode 4 1	assign rotation center
1	mouse mode 4 2	pick atom
2	mouse mode 4 3	pick bond (2 atoms)
3	mouse mode 4 4	pick angle (3 atoms)
4	mouse mode 4 5	pick dihedral (4 atoms)
5	mouse mode 4 6	move atom
6	mouse mode 4 7	move residue
7	mouse mode 4 8	move fragment
8	mouse mode 4 9	move neuron
9	mouse mode 4 13	move highlighted rep
%	mouse mode 4 10	apply force on atom
^	mouse mode 4 11	apply force on residue
&	mouse mode 4 12	apply force on fragment

Table 4.1: Mouse control hot keys.

Hot Key	Command	Purpose
x	`rock x by 1 -1`	spin about x axis
X	`rock x by 1 70`	rock about x axis
y	`rock y by 1 -1`	spin about y axis
Y	`rock y by 1 70`	rock about y axis
z	`rock z by 1 -1`	spin about z axis
Z	`rock z by 1 70`	rock about z axis
j, Cntl-n	`rotate x by 2`	rotate 2° about x
k, Cntl-p	`rotate x by -2`	rotate −2° about x
l, Cntl-f	`rotate y by 2`	rotate 2° about y
h, Cntl-b	`rotate y by -2`	rotate −2° about y
g	`rotate z by 2`	rotate 2° about z
G	`rotate z by -2`	rotate −2° about z
Cntl-a	`scale by 1.1`	enlarge 10 percent
Cntl-z	`scale by 0.9`	shrink 10 percent

Table 4.2: Rotation & scaling hot keys.

Hot Key	Command	Purpose
Alt-M	`menu main off;menu main on`	Show main menu
Alt-f	`menu files off;menu files on`	Show files menu
Alt-g	`menu graphics off;menu graphics on`	Show graphics menu
Alt-l	`menu labels off;menu labels on`	Show labels menu
Alt-r	`menu render off;menu render on`	Show render menu
Alt-d	`menu display off;menu display on`	Show display menu
Alt-c	`menu color off;menu color on`	Show color menu
Cntl-r	`display resetview`	Reset display
Alt-q	`quit confirm`	Quit VND with confirmation
Alt-Q	`quit`	Quit VND
Alt-h	`hyperref invert`	Invert hyper text mode (NOT help)

Table 4.3: Menu control hot keys.

Hot Key	Command	Purpose
+,f,F	`animate next`	move to next frame
-,b,B	`animate prev`	move to previous frame
.,>	`animate forward`	play animation forward
,	`animate reverse`	play animation reverse
<	`animate reverse`	play animation reverse
/, ?	`animate pause`	stop animation

Table 4.4: Animation hot keys.

4.2 Menu Tabs

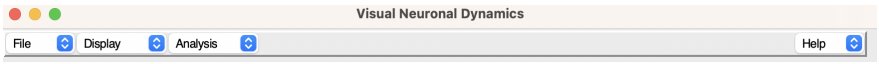

Figure 4.1: The main menu options

4.2.1 File Opening Menu

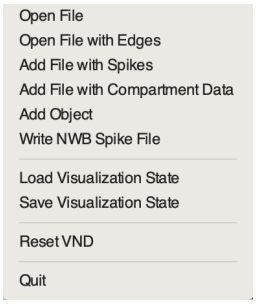

Figure 4.2: The file opening menu contains the buttons used throughout the user guide that load files with certain properties such as edges, spikes, and/or compartment data.

Open File
Use this option to load a general JSON circuit file.
Open File with Edges
To display connections between neurons, load this data file. See references
Add File with Spikes
For spiking activity, add the file with the spikes. See HDF sections.
Add file with Compartment Data
To visualize voltage animation use this option.
Add Object
This options adds an object such as a .PLY to the scene
Write NWB Spike File
This option is going to write a Neuronal Data Without Borders (.nwb) file to the directory. See:
Load Visualization State
This option loads a scene by using a .tcl file.
Save Visualization State
Save a scene by writing a .tcl file to the directory.
Reset VND
Pressing this button will reset VND back to the original state with no systems loaded. **Quit** This buttons quits and terminates the program.

4.2.2 Display Menu and Display Settings Window

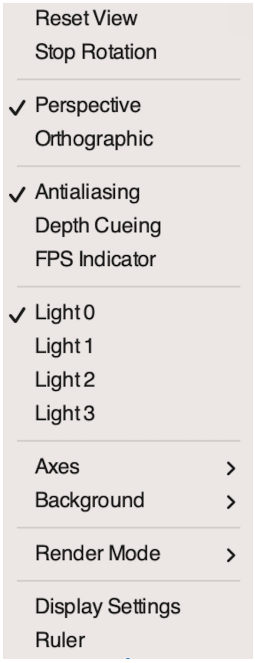

Figure 4.3: The display menu controls many of the characteristics of the graphics display window. The characteristics which may be modified include:

- **Reset View** – This menu item can be used to force VND to reset the scene back to the default viewing orientation and scale as is done when a neuron is first loaded.
- **Stop Rotation** – This menu item stops autorotation of the scene. The scene can be autorotated by quickly dragging the mouse while briefly depressing and releaseing the mouse button, leaving the scene spinning until it is stopped either by this menu item or by further mouse interactions.
- **Perspective** – The view of the scene can be Perspective or Orthographic. In the perspective view (the default), objects which are far away are smaller than those near by. In the orthographic view, both objects appear at the same scale. Note that several of the supported external rendering programs do not support orthographic rendering. As such, it may be necessary to "fake it" by translating the scene far away from the camera, and apply a zoom factor. This has the effect of significantly reducing the perspective, while not truly an orthographic view.
- **Antialiasing** – Turns antialiasing on or off. Antialiasing helps smooth out the jagged appearance of displayed geometry resulting from the inherently discrete pixels on the display device. The antialiasing feature is only available on platforms which support full-screen antialiasing, sometimes known as "multisample antialiasing". On platforms lacking the multisample capability, there may be alternate ways to perform full-screen antialiasing by selecting an option in the display driver setup. Windows machines most commonly place these controls in the display driver configuration panel.

- **Depth Cueing** – Turns depth cueing on or off. Depth cueing causes distant objects to blend into the background color, in order to aid in 3-D depth perception. The depth cueing settings are controlled in the Display Settings window. The **Cue Mode** parameter controls which type of fog equation is used. The **Linear** depth cueing mode provides a simple depth gradient with a defined starting point and endpoint. The **Exp** and **Exp2** depth cueing modes take a density parameter, and generally blend into the background color much more sharply than the linear depth cueing mode. Scaling up the neuron will increase the amount of depth cueing effect that is visible, since it will occupy a larger depth range. Scaling the neuron size down decreases the depth cueing effect. Translating the neuron into and out of the screen will cause it to blend into and out of the background color.

- **Culling** – Turns backface culling on or off. This feature is primarily used to accelerate rendering performance on software based implementations of OpenGL, such as Mesa. Backface culling actually reduces performance on some hardware renderers, so you'll have to use your own best judgement on whether or not it is helpful to use on your specific computer system.

- **FPS** – This option enables or disables on-the-fly display of the achieved VND rendering frame rate. The frame rate is displayed in the upper right hand corner of the graphics window when it is enabled.

- **Lights** – The graphics display window can use up to four separate light sources to add a realistic effect to displayed graphical objects. The Lights On browser turns these light sources on or off. If the number is highlighted, the light is on, and clicking on it turns the light off.

- **Axes** – A set of XYZ axes may be displayed at any one of five places on the screen (each of the corners or the center) or turned off. This is controlled by the Axes chooser.

- **Background** – The display background can either be set to a uniform color over the entire scene, or a vertical gradient can be set with a linearly changing color from the top of the viewport to the bottom.

- **Rendermode** – The Rendermode chooser controls which low-level rendering method VND uses. The **Normal** rendering mode is the default VND rendering algorithm based on standard fixed-function OpenGL. The **GLSL** rendering mode uses OpenGL Programmable Shading Language to implement real-time ray tracing of spheres, alpha-blended transparency, and high-quality per-pixel lighting for all geometry. On machines with high performance graphics boards supporting programmable shading, the **GLSL** rendering mode provides quality on par with many of the external software renderers supported by VND, but at interactive display rates. The Tachyon RTX RTRT is a custom made built in real time ray tracing engine that is especially powerful on NVIDIA GPUs. The Acrobat3D render mode utilize Adobes rendering packages.

- **Ruler** – The Ruler allows a user to see the micron unit scale of the current elements in the OpenGL window. The ruler has different settings such as the tape, grid, and/or scale ruler. The option to switch into orthographic mode exists in this menu as well.

- **Clipping Planes (Near Clip and Far Clip)** – These controls are found in the Display Settings window. Only those parts of the scene between the near and far clipping planes are drawn. The display clipping planes also set the depth cueing start and endpoints. Objects at the near clipping plane are distinct and crisp, objects at the far clipping plane will be blended

into the background. Clipping planes positions are changed with the Near Clip and Far Clip controls. It is not possible for the near clip to be farther away than the far clip. When using stereo, it may be useful to set the near clip plane much lower than the default value. This makes the geometry "pop out of the screen" a bit more, and can be used for greater dramatic effect.

- **Screen Height (Hgt) and Distance (Dist)** – These controls are found in the Display Settings window. The screen height, along with the screen distance, defines the geometry and position of the display screen relative to the viewer. The screen height is the vertical size of the display screen, in 'world' coordinates. Each neuron is initially scaled and translated to fit within a 2 x 2 x 2 box centered at the origin; so the screen height helps determine how large the neuron appears initially to the viewer.

  The screen distance parameter determines the distance, in 'world' coordinates, from the origin to the display screen. If this is zero, the origin of the coordinate system in which molecules (and all other graphical objects) are drawn coincides with the center of the display. If distance is negative the origin is located between the viewer and the screen, if it is positive, the screen is closer to the viewer than the origin. A negative value puts any stereo image in front of the screen, aiding the three-dimensional effect; a positive value results in a stereo image that is behind the screen, a less dramatic effect (but easier to see, for some people) stereo effect.

- **Shadows** – The shadows control enables and disables direct lighting shadowing when using the built-in Tachyon CPU or GPU renderers or when exporting the VND molecular scene to external renderers that implement shadowing algorithms. The simple direct lighting model implemented in most renderers yields shadows that are completely dark, producing a somewhat harsh lighting quality akin to what would be expected in a desert under full sunlight with no clouds.

- **Ambient Occlusion** – The ambient occlusion (AO) lighting control enables the use of so-called ambient occlusion indirect lighting when using the built-in Tachyon CPU or GPU renderers, or external renderers that implement AO. Images with much higher quality shading can be produced by augmenting direct lighting with ambient occlusion or broad angle or indirect lighting techniques. Ambient occlusion lighting emulates the broad lighting effects similar to what would be experienced on a cloudy or overcast day with omnidirectional light arriving on all surfaces. The AO ambient coefficient controls the strength of the omnidirectional lighting components. The AO direct coefficient scales the direct lighting contribution associated with the directional and positional lights.

- **Depth of Field (DoF)** – The depth of field (DoF) control enables or disables emulation of depth of field focal blur effects associated with fast focal ratio camera optics and close focus distances. The depth of field implementation provided by the built-in Tachyon ray tracer and most other renderers yield a plane of perfect focus at a specified distance from the camera. The degree of focal blurring with increasing distance from the plane of perfect focus depends on both the simulated f/stop and the distance between the plane of perfect focus and the camera.

4.2.3 Analysis Tools Menu

The analysis menu offers two different tools for analysis and can be launched from this menu. These analysis methods will be further discussed in: Chapter 6.

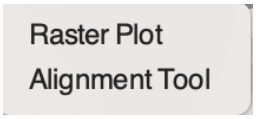

Figure 4.4: The analysis menu offers two different tools for analyis and can be launched from this menu.

For in depth discussion into analysis tools, See Chapter 6

Chapter 5

Advanced User interface and explanations

5.1 Main Window

Here we'll delve into the main graphical user interface with hopes of making things simple and easy to interact with neuronal systems.

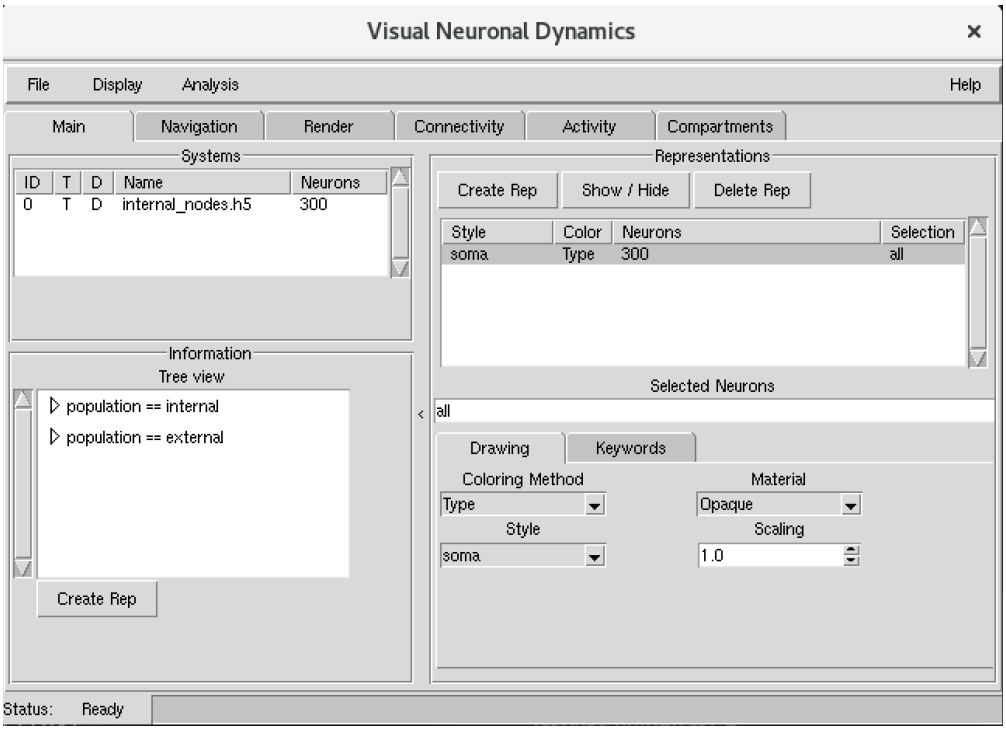

Figure 5.1: Visual Neuronal Dynamics window, showing the 'Main' tab with 'Systems', 'Information' and 'Representations' interfaces.

5.2 Main Tab

The main window is a graphical user interface developed by the TCL/Tk Library. The main window is divided into page tabs under a general menu bar. Each tab corresponds with interacting with neurobiological systems. For example there is 'Navigation,' 'Render,' 'Connectivity,' 'Activity,' and 'Compartments.' A page is subdivided into different 'Panels' that host different serialized GUI elements.

Systems Panel

The 'Systems panel' shows the main neuronal systems present in the file browser window. When a system is loaded, a 5 columned table is populated with 'ID,' 'T,' 'D,' 'Name,' and 'Neurons.'

The 'ID' is associated with the current number of file systems open, however the current version only supports a maximum of 1 file set.

The 'T' and 'D' columns are held constant and reflect the neuronal system file query.

The 'Name' column contains the name of the H5 file currently loaded.

The 'Neurons' column is an aggregate number of the virtual and non-virtual neurons in the file system. This number is helpful in making selections and is further discussed in the selection making chapter.

Information Panel

The 'Information' panel contains an interactive tree view structure peering into the loaded system. The neuronal system loaded hosts 'populations' of cells further containing branches that are 'group' and have auxiliary shoots that are 'type.' The tree can be expanded or collapsed by clicking on the horizontal left arrow.

The interactive features the ability to select a specific population, group, and/or type and create a graphical representations from it. After a single click selection a user can press the 'Create Rep' button to make a graphical representation of that selection.

Representations Panel

The 'Representations' panel contains a table with buttons to 'Create Rep,' 'Show/Hide,' and 'Delete Rep.' The buttons manage the representations currently present and let users create, toggle, and delete representations respectively.

The table contains columns and gets populated when a representation is made. Each table contains a 'Style,' 'Color,' 'Neurons,' and 'Selection' column.

The 'Style' column pertains to 2 different VND neuron drawing styles on the OpenGL. This is either 'soma' and or 'morphology.'

The 'Color' column is reflective of the several different colors available in VND and an extraneous to the color scale, a coloring of by 'type'

The 'Neurons' column contains an integer number of neurons in that particular representation.

The 'Selection' column contains a string that is the name of the selection and what query is executed to display the representation in the fileset.

- **Selection Entry**
  The *selection entry box* is the interface between the user and the internal neuronal data files. VND has a very simple neuronal selection language that effectively partitions any large

neuronal system to probe specific regions of interest. Changes can be seen instantaneously on the OpenGL display. Specific strings will be discussed in further chapters.

- **Drawing**
  A drawing method (representation style), which determines what shape to draw the neurons somatic cell and/or morphology.

  In the next section we will cover the various parameters for drawing neurons in VND:

  - *Coloring* Method which determines how to color each of the neurons in the system included in the view.
  - *Material* There are several materials that VND offers and these materials can impact performance and artistic quality of rendered neuron viewings.
  - *Style* The style parameter enables users to either see the body of the neuron or the entire morphology structure including the dendrites and the axons of the neuron.
  - *Scaling* The setting lets neurons drawn on the screen have a larger appearance relative to the resolution of the screen. This scaling is helpful for having pictures and rendered material incorporated into posters and presentations.

- **Keywords**
  The Keywords pages consists of an interactive, scrollable table. This is the main attribute browser of neuronal files loaded into VND. Each system loaded consists of different reserved/non-reserved attributes that can be readily browsed. A double click of any element will appear in the 'Selected Neurons' box and is the first step in building a string query.

5.3 A Brief Introduction to Neuronal Selections

In this section it is assumed that you have the 300_cells system loaded. If this is not true, go back and repeat the process described there. VND has a powerful neuronal selection method which is very helpful when generating attractive, informative, and complex graphics. To change which neurons are used to display each representation of a selection shown in the display window, go to the 'Representations' panel select the representation you want to change. You can then either edit the different fields (selection, coloring method, or drawing method) or use the Delete button to delete the view entirely. Try changing or deleting some of the views. When finished, delete all representations for the 300_cells structure. To get the basic line drawing view back, clear the neuronal selection text entry area, enter all and press the Create Rep button. Neurons may be selected by the basis of their reserved attributes such as coordinates and cell type. They may also be selected by name or by many other identifiers.

Several more abstract selection criteria are available. For instance, the selection soma x >0 finds all atoms with an x coordinate greater than 0, while y <0 and x >0 selects all atoms with a y value less than 0 and an x value greater than 0. Another selection example can be looking for excitatory and/or inhibitory neurons. A selection of ei = e generates the visualization of only the excitatory neurons, See A.

5.3.1 Browsing the Reserved/Non-Reserved Attributes of Neurons

By now you may be curious to know what other attributes are in the neuronal file loaded. If so, make your way to the Main Tab → Representations Panel → 'Keywords' Notebook Page. Here you will find a tablelist with 3 columns 'Attributes,' 'Values,' and 'Min/Max.' Various reserved and non-reserved attributes and their respective values can be browsed and incorporated to make selections as discussed in the next section.

5.3.2 Building Queries from the Attributes Table

If you double click any elements in the table, these elements will be appended to the 'Selected Neurons' entry field. For example, you can single click 'pop_name' and double click on the value 'Scnn1a' instead of typing the full name of the population. See Appendix for full selection examples. A

5.3.3 Examples of Selection Strings

```
(stride 100)
(soma x > 2)
(soma x < 0) && (soma y > 0) && (soma z < -1)
(model_name == PV1) && (soma x > 0) && (ei = i)
```

5.4 Navigation Tab

The Navigation tab is a general interface for manipulating objects loaded into the scene. Examples are polygon (.ply) files and can be neuronal probes. This tab is split into two panels that help facilitate object positioning.

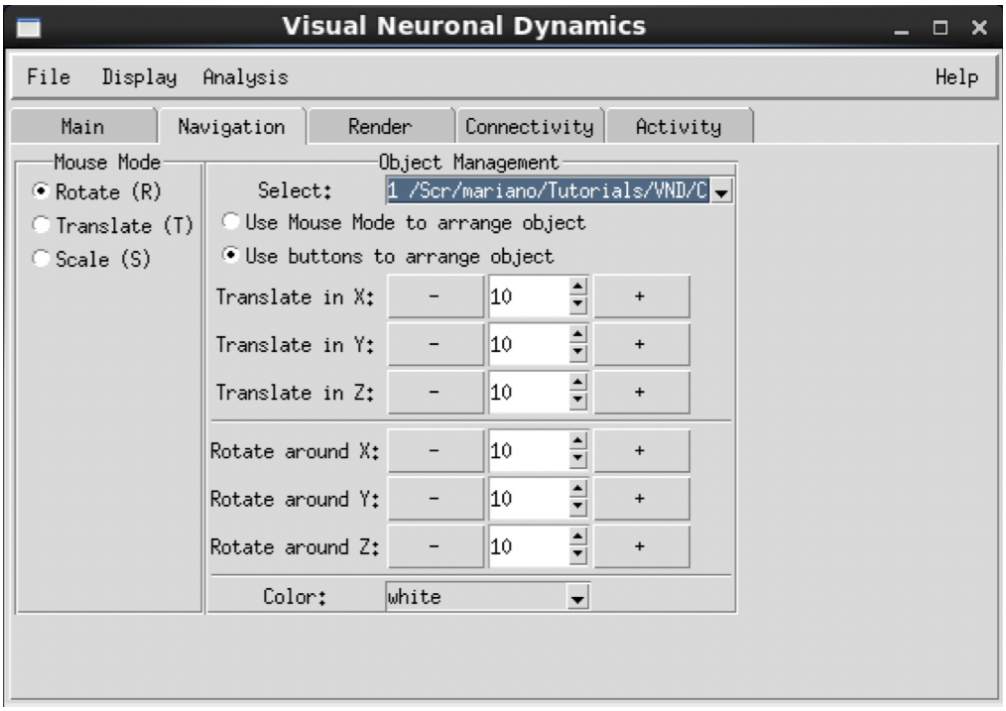

Figure 5.2: VND window showing the 'Navigation' tab, which contains the 'Mouse Mode' and 'Object Management' interfaces.

5.4.1 Mouse Mode

The radio buttons can be selected to edit the how the peripheral mouse will interact. According to chapter 5.1.1, there is a **Rotate Mode (R), Translate Mode (T), and Scale Mode (S)**

5.4.2 Object Management

The Navigation tab features the ability to select different objects within the VND scene. Primarily the object that can be modified with 6 degrees of freedom in translations, rotations, and scaling.

The first element in the object management is the "Selected" drop down menu. There will be a list of objects readily available to control. Underneath contain two radio buttons that legislate control given to either the mouse or buttons. If the latter option is selected, controls for navigation are below.

Translations and Rotations around X, Y, and Z

The units for translation are Angstroms (A) and the units for rotation are degrees (°). VND controls buttons include: 'Minus -,' an editable number field, small up/down arrows, and 'Positive +.' The larger buttons '-' and '+' indicates the positional direction of the control, it would be either the positive direction of X or the negative direction of X. The number field is the magnitude of the control, in this case entering a number of 15 would translate/rotate the unit by 1.5 A (Angstroms)/15 degrees respectively. The smaller up/down arrows next to the editable number field are quick incrementers by 10 and are useful in doing rapid navigation.

5.5 Render Tab

The Render window is used to export the currently displayed graphics scene to an image file or to a geometric scene description file suitable for use by one of several external renderers, which can produce a final image. The supported rendering packages are listed in the table below.

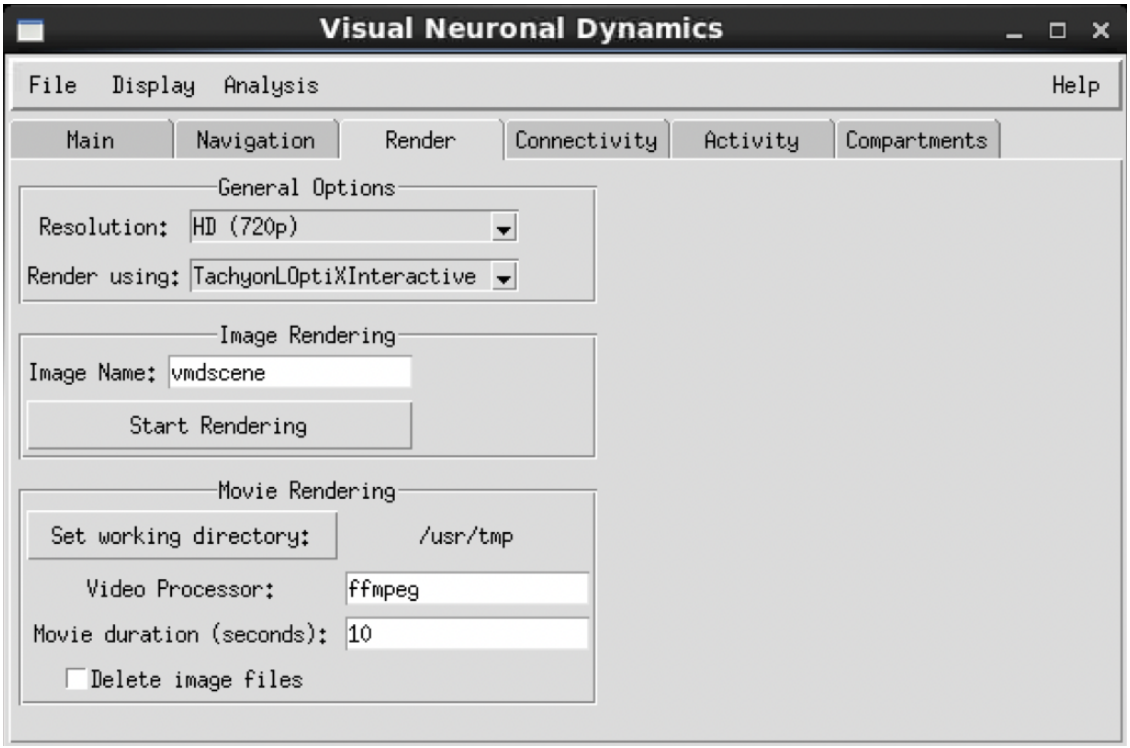

Figure 5.3: VND window with 'Render' tab. The options define OpenGL resolution and render selection to generate high quality images.

Name	Description
ART[1]	Simple VORT ray tracer
Gelato	NVIDIA Gelato PYG Format
PostScript	Simple Vector PostScript Output
POV3[2]	POV-Ray 3.x ray tracer
Radiance[3]	Radiosity ray tracer
Raster3D[4]	Fast raster file generator
Rayshade[5]	Rayshade ray tracer
RenderMan	PIXAR RenderMan RIB Format, render with Aqsis, Pixie, PRMan, RenderDotC
STL	Stereolithography format, triangles only
Tachyon[6]	High quality parallel ray tracer
TachyonInternal[6]	Fast, built-in Tachyon engine that generates images directly from VMD internal data structures with no intermediate step
TachyonLOptiXInternal	Fast, built-in GPU-accelerated Tachyon for NVIDIA GPUs (available on supported platforms)
TachyonLOptiXInteractive	Interactive, built-in, GPU-accelerated version of Tachyon for NVIDIA GPUs (available on supported platforms)
TachyonLOSPRayInternal	Fast, built-in OSPRay ray tracer for Intel CPUs (available on supported platforms)
TachyonLOSPRayInteractive	Interactive, built-in, OSPRay ray tracer for Intel CPUs (available on supported platforms)
VRML-1	Virtual Reality Markup Language V1.0
VRML-2	Virtual Reality Markup Language V2.0
Wavefront	Wavefront .OBJ/.MTL scene format, loads into 3DS Max, Blender, Maya, and others
X3D[7]	X3D declarative scene format
X3DOM[8]	Open source HTML5-based X3D viewing system

See: A.2

5.5.1 General Options

The first step in rendering is to select a target resolution for the image or video you want to preserve by selecting from the 'Resolution' drop-down menu. After selecting a resolution, select a render method from the next 'Render using' drop down menu. See Table 5.5

5.5.2 Image Rendering

The next few steps involve giving a name to the rendered image in the 'Image name' text field. But, before pressing 'Start Rendering' go to the next panel if you want to render a movie!

5.5.3 Movie Rendering

To generate a movie, first set a working directory. By default /usr/temp is listed. The next step is to select a video codec, by default it is libffmpeg or "ffmpeg." Lastly, enter the duration of the movie you want generated in units of seconds.

Once complete, go to the 'Image Rendering' panel and press 'Start Rendering.' Your new image and/or movie will be created in the 'Working directory.'

5.6 Connections Tab

This tab manages how neurons are connected to each other.

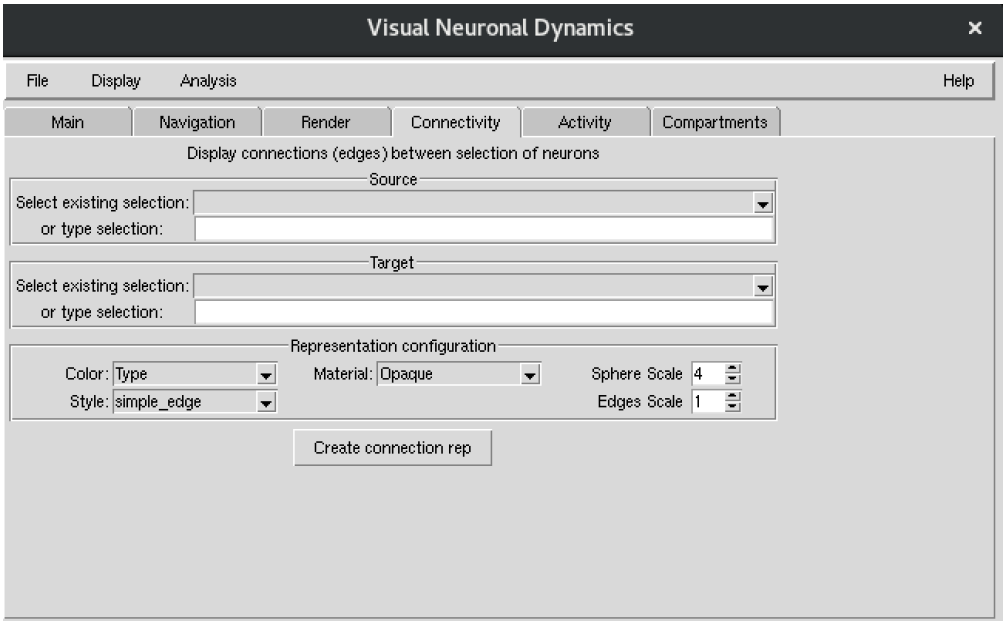

Figure 5.4: VND window showing the 'Connectivity' tab, which contains the selections and editable representations interface.

5.6.1 Draw a cylindrical connection between 2 neurons

The connection tab lets users add a visible connection between any neuron of choice within a given system. The graphics area of this tab lets users edit graphical features of the physically drawn connections between neurons.

5.6.2 Source

The source of the neuron is the initial spot where the neuron connection is going to start. A user can select from an existing population or enter a "selection string."

5.6.3 Target

The target of the neuron connection is the endpoint of the neuronal connection made. A user can select from an existing population or enter a "selection string."

5.6.4 Representation Configuration

This panel contains several config options:

- *Color* Can select from Type or a list of colors
- *Style* A table list of different drawing styles for edge connections

Table 5.1: Explanation of connection representations styles available in 'Connectivity' tab in VND.

Style	Explanation
simple_edge	Draws cylinder connecting neuron selections
source_soma	Draws spheres on source neurons' somas
target_soma	Draws spheres on target neurons' somas
source_target_soma	Draws spheres on both source and target neurons' somas
simple_edge_swc	Draws cylinder connecting morphology synapse locations
source_sphere_swc	Draws spheres at source morphology synapse locations
target_sphere_swc	Draws spheres at target morphology synapse locations
source_target_sphere_swc	Draws spheres at source and target morphology synapse locations

Table 5.2: A list of materials

Material
"Opaque"
"Transparent"
"BrushedMetal"
"Diffuse"
"Ghost"
"Glass1"
"Glass2"
"Glass3"
"Glossy"
"HardPlastic"
"MetallicPastel"
"Steel"
"Translucent"
"Edgy"
"EdgyShiny"
"EdgyGlass"
"Goodsell"
"AOShiny"
"AOChalky"
"AOEdgy"
"BlownGlass"
"GlassBubble"
"RTChrome"

- *Sphere Scale* A size for the soma spheres drawn from *1-10*

- *Edges Scale* A size for the cylinders connecting the neurons from *1-10*

5.7 Activity Tab

5.7.1 Displaying spiking animation

Visualizing spike data

While there are numerous metrics for assessing the dynamics of large interconnected networks of neurons, perhaps the most important is the spiking activity, i.e. action potentials. Most often spiking activity is stored in tables or rasters comprised of thousands-to-millions of individual data points. And though this format is good for calculating simple statistics, it also hampers scientists trying to discover more complex patterns of network activity. VND allows modelers to animate and replay the spiking activity with full spatial and temporal accuracy, providing opportunities for scientists to make even more insightful observations they would then with a flat tabular representation.

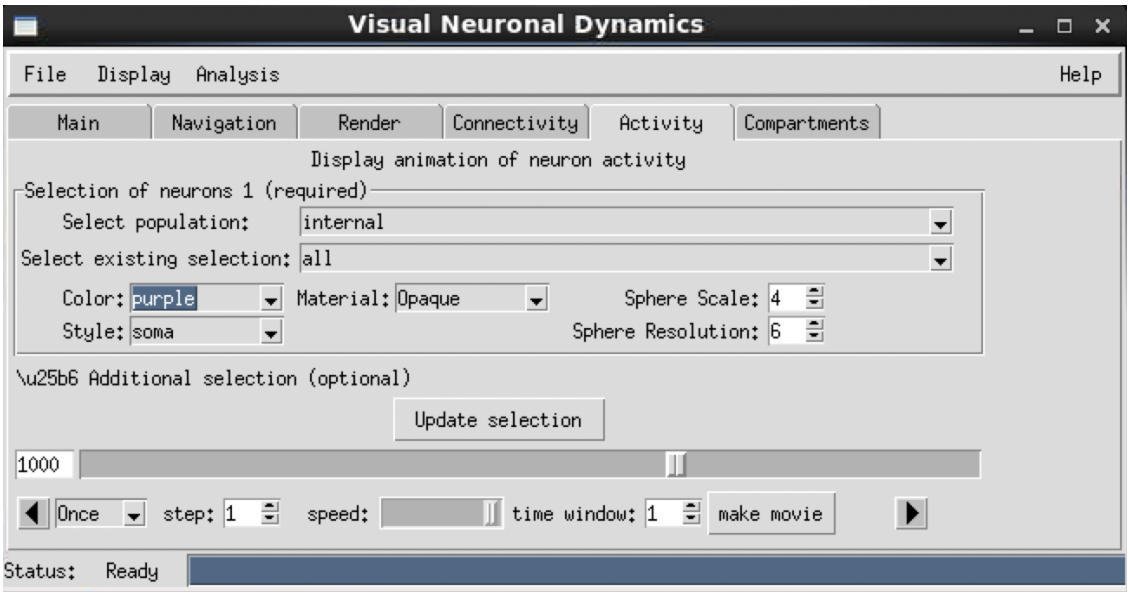

Figure 5.5: VND window with 'Activity' tab. Menus define selection of neurons for spike activity (e.g. 'all'), configuration (e.g. color 'purple', style 'soma') and animation playback (e.g. step '5', time window '5'). Now featuring a drop-down menu to have additional selections.

Spiking activity tools

VND offers the ability to load the timeseries data as an .H5 file and have controls over visualizing the firing neurons and the ability to video scrub an OpenGL display.

The 'Activity' tab allows the user to visualize spike activity data of the neurons in the model. The first panel contains the functions to define a population of neurons and selections using our selection language.

The latter panel contains a playback menu with buttons to 'play' / 'pause' / 'play backwards' and options to control the 'step', 'speed' and 'time window' for the animation.

5.7.2 Buttons for animations

A similar interface to the 'Connections Tab' but now including editable representations for each frame during several time steps.

- **Color**
  Can select from Type or a list of colors
- **Style**
  A table list of different drawing styles for edge connections
- **Material List**
  - Opaque
  - Transparent
  - BrushedMetal
  - Diffuse
    See 5.2
- **Sphere Scale** A size for the soma spheres drawn from *1-10*
- **Sphere resolution** A size for the pixel resolution of the spheres drawn at each frame from *1-10*

  You can also choose an additional selection by clicking on the right arrow to expand the 'Additional Selection (optional)'

Updating changes made in the panel

When a change is made to any frame field, press 'Update Selection'

Video scrubber

Below the selection panels in the 'Activity Tab' the video scrubber is an interface. The left most text field indicates the total number of frames in the neuronal system.

Underneath we see the 'rewind' button and to the right a dropdown menu containing 'Once,' 'Loop,' and 'Rock.' These options determine how the video is continuously played.

Next we have 'Step' and this is the general amount of frames that are cycled through during each play progression.

Then we have the 'Speed' bar that can change the brevity of the entire spiking activity time.

Next we have the 'time window' and this value is associated with how long each frame stays on the screen.

By pressing the 'make movie' button. A separate window is presented to help with the rendering options (See render tab).

The 'play' button is the right most button on the scrubber.

5.8 Compartment Tab

One of the most common ways for representing individual neurons is the "multi-compartment model". With it, each neuron is composed of many different computational units, or compartments, used to calculate and track things like membrane capacitance, the effects of any channels or receptors, and flow of ionic currents at each given time step of the simulation [8].

The compartment tab allows modelers to visualize and animate the dynamics of all the compartment for selected cell(s). For example, one might use it for visualizing the change of membrane potential over time. Coupling the cells potential dynamics along-side synapse placement and the spiking activity of nearby cells can be useful in studying things like functional vs. geometric connectivity of a network. And VND doesn't limit itself to just the membrane potential. The Compartment Tab can animate the activity of any continuous compartment variable recorded during a simulation, like changes in calcium or potassium concentrations, or even changes in channel strength.

Similar to activity and connectivity, the compartment tab offers the ability to make a selection and edit its representation characteristics.

5.8.1 Loading compartment data

The first step in viewing compartment data is to have a circuit json file loaded into VND. The next step is upload a compartment file by selecting the VND menu option: Go to 'File' → 'Add File with Compartment Data'. The file for compartment data is located in the 'Output' directory of the BMTK network structure and should have a name containing '*_report.h5'

5.8.2 Viewing compartment data

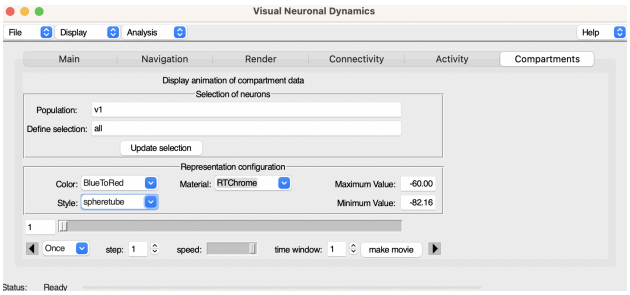

Figure 5.6: Comparment window

Similar to the activity tab, the compartment tab contains a Selection Panel and an 'Editable representations' panel along with a scrubber for playback.

Population

Set this value to the population containing the compartment output data. Usually it is 'v1'.

Selection

This field shows the amount of neurons present on the OpenGL display for viewing compartment animations.

Maximum Value

This value should be set to approximately -60 mV. It is the threshold amount of voltage to act as a stimulus for color change.

Minimum Value

This value should be 80 mv and is the maximum threshold amount of voltage to act as a stimulus for color change.

Visualizing the animation

To visualize the animation of voltage change over time be sure to press 'Update Selection' and then use the video scrubber to play the animation. The aformentioned scrubber can be edited to change timestep and speed of each frame. Notice the gradient of color change based on relative voltage between Minimum and Maximum.

Chapter 6

Analysis

This chapter is going to discuss how to use the tools for analyzing neuronal systems.

6.1 Raster Plot

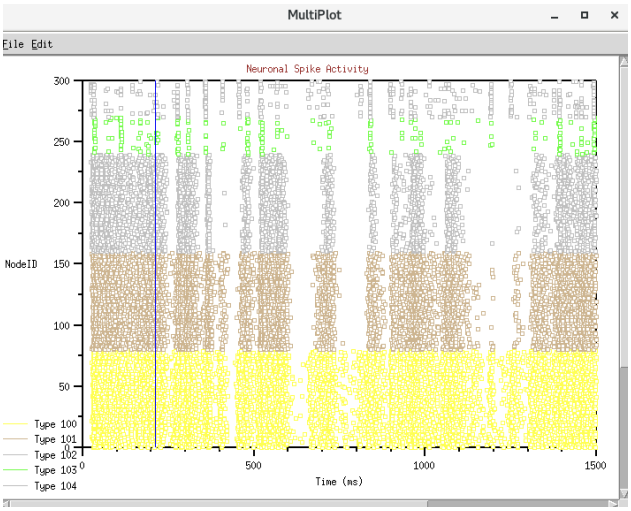

Figure 6.1: A raster plot shown of 'all' selected neurons. Each individual point is specified by nodeID plotted against time. The blue vertical line is an indicator of the current animation frame inside the VND engine.

Launch the tool by selecting the 'Analysis' button in the top main menu row.

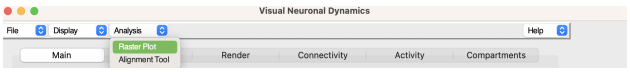

Figure 6.2: Location of the raster plot.

When one is visualizing spike data he/she may want to know what neuron is being fired. By loading Analysis - Raster Plot, a loaded graph plotting nodeID against time is shown. The blue vertical line is an indicator of the current animation frame inside the VND engine. This graph also offers an interactive feature to select a current node that is being fired in the animation frame.

6.2 Ruler

Located in the 'Display' in the top row menu bar, on the bottom of the list is 'Ruler'

A window is going to appear and you can select the different types of ruler options: — side tape measures, grid, and scale bar — change dynamically as you zoom out.

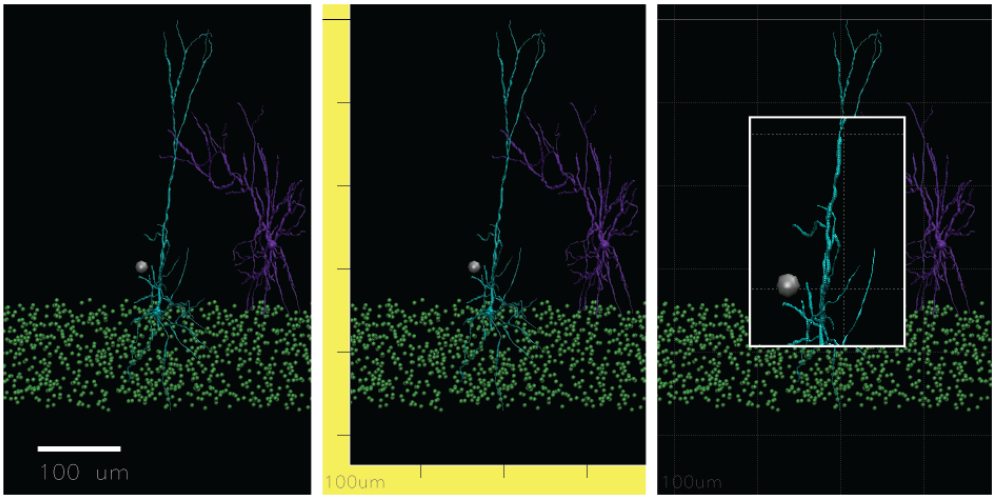

Figure 6.3: The three kinds of ruler overlays (from left to right): scale bar, tape, and grid.

6.3 Alignment Tool

At any point a user may want to manipulate a current selection so that a certain orientation is achieved. By utilizing the alignment tool, it calculates and displays the principal axes of the current selection and align these to a user-specified direction and additionally an input vector.

6.3.1 Using the Alignment Tool

1. Make a selection by choosing an existing selection in the first dropdown menu
2. In the Axes and Alignment Vector Panel, press 'Calculate Draw Axes' this is going to draw a main PCA for your selection. Press the "=" sign to reset the display view.
3. To make an alignment to an input vector: input numbers into 'input x,' 'input y,' and 'input z' then press 'Align to x,y,z'
4. Press 'Reset Position' to reset the neurons back to the original position or identity matrix
5. Us the navigation controls panel to translate and/or rotate by 6 degrees of freedom.

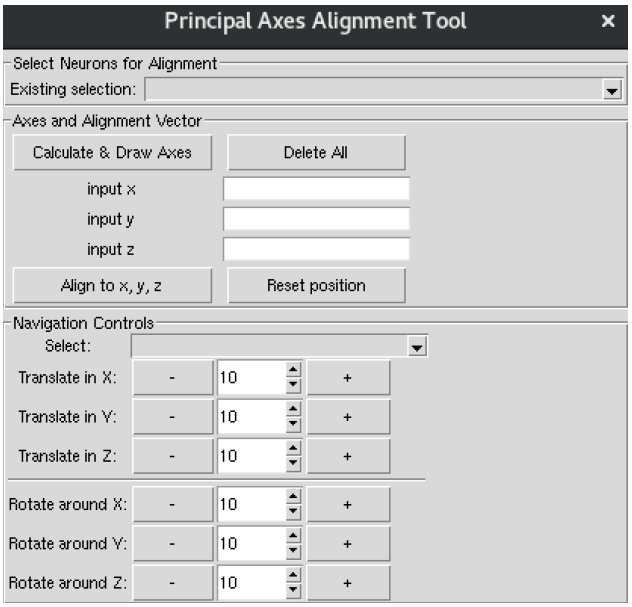

Figure 6.4: Alignment tool and it's GUI window

6. Use the arrow keys to increase the magnitude of translation/rotation and use the '+' and '-' sign to indicate direction

7. Before exiting, press 'Delete All' at the top of the GUI window.

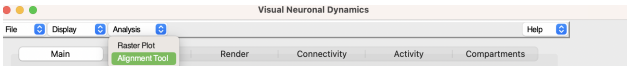

Figure 6.5: Location of the raster plot.

Functionality for incremental translations and rotations are also provided.

Appendix A

Neuron selection language

A.1 Neuron selection language

Logical operators:
And: `&&` Or: `||` Not: `!`
Group search terms by using parenthesis: `(` , `)`.
Nesting is permitted.
Examples:

```
(soma x < 0) && (population == LGN)
(node_id > 100) || ( (soma x > 0) && (population == LGN) )
```

search term	definition	comparators	example
`all`	all neurons in model	`<, >, ==`	`all`
`soma x | y | z`	pos. of neuron center	`<, >, ==`	`soma x > 0`
`soma xrot | yrot | zrot`	rot. of neuron (radians)	`<, >, ==`	`soma xrot > 0`
`morphology x | y | z`	pos. of any segment of morphology	`<, >, ==`	`morphology x > 0`
`type`	type ID	`==`	`type == 2003`
`node_id`	node ID	`<, >, ==`	`node_id < 100`
`node`	node ID (synonym)	`<, >, ==`	`neuron_id < 100`
`population`	population name	`==`	`population == LGN`
`group`	group number	`<, >, ==`	`group == 3`
`fileset`	file set number in circuit config.	`<, >, ==`	`fileset == 2`
`model_type`	name of model type	-	`model_type=biophysical`
`stride`	select every Nth neuron	-	`stride 50`
`within`	within N units of a neuron	-	`within 10 of node_id 8`
`has_morpho`	true if neuron has morphology	-	`has_morpho`
non-reserved attributes	per-model non-standard attributes	`==`	`ei == e`

Notes:
The comparators `<=`, `>=`, and `!=` are not yet permitted. Use `!(A == B)` instead of `(A != B)` .
The `within` term currently only works with `node_id`/`node`, e.g. `within 10 of node_id 1073`.
Combine with `population` and similar terms to choose specific neurons in multi-population systems.

A.2 Acknowledgements

[1]Available from `http://bund.com.au/~dgh/eric/` http://bund.com.au/ dgh/eric/ along with the rest of VORT package
[2]See `http://www.povray.org/` http://www.povray.org/
[3]See `http://radsite.lbl.gov/radiance/HOME.html` http://radsite.lbl.gov/radiance/HOME.html
[4]See `http://www.bmsc.washington.edu/raster3d/` http://www.bmsc.washington.edu/raster3d/
[5]See `http://graphics.stanford.edu/~cek/rayshade/rayshade.html`
http://graphics.stanford.edu/ cek/rayshade/rayshade.html
[6]See `http://www.photonlimited.com/~johns/tachyon/` http://www.photonlimited.com/ johns/tachyon/
[7]See `http://www.web3d.org/x3d/` http://www.web3d.org/x3d
[8]See `http://www.x3dom.org/` http://www.x3dom.org/

Bibliography

[1] William Humphrey, Andrew Dalke, and Klaus Schulten. VMD – Visual Molecular Dynamics. 14:33–38, 1996.

[2] R. Sharma, T. S. Huang, V. I. Pavlovic, K. Schulten, A. Dalke, J. Phillips, M. Zeller, W. Humphrey, Y. Zhao, Z. Lo, and S. Chu. Speech/gesture interface to a visual computing environment for molecular biologists. In *Proceedings of 13th ICPR 96*, volume 3, pages 964–968, 1996.

[3] John Stone and Mark Underwood. Rendering of numerical flow simulations using MPI. In *Second MPI Developer's Conference*, pages 138–141. IEEE Computer Society Technical Committee on Distributed Processing, IEEE Computer Society Press, 1996.

[4] Michael Zeller, James C. Phillips, Andrew Dalke, William Humphrey, Klaus Schulten, Rajeev Sharma, T. S. Huang, V. I. Pavlovic, Y. Zhao, Z. Lo, and S. Chu. A visual computing environment for very large scale biomolecular modeling. In *Proceedings of the 1997 IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 3–12. IEEE Computer Society Press, 1997.

[5] John E. Stone. *An Efficient Library for Parallel Ray Tracing and Animation*. Master's thesis, Computer Science Department, University of Missouri-Rolla, April 1998.

[6] John E. Stone, Justin Gullingsrud, Paul Grayson, and Klaus Schulten. A system for interactive molecular dynamics simulation. In John F. Hughes and Carlo H. Séquin, editors, *2001 ACM Symposium on Interactive 3D Graphics*, pages 191–194, New York, 2001. ACM SIGGRAPH.

[7] I. Wald, G. Johnson, J. Amstutz, C. Brownlee, A. Knoll, J. Jeffers, J. Gunther, and P. Navratil. OSPRay – a CPU ray tracing framework for scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):1–1, 2017.

[8] David Sterratt, Bruce Graham, Andrew Gillies, Gaute Einevoll, and David Willshaw. *Principles of computational modelling in neuroscience*, volume 2. 2023.