Viz: Brain Visualizer

Interactively Rendering BOSS Input and Output

Remy Oukaour
Dept. of Computer Science
Stony Brook University
Stony Brook, NY 11794

November 5, 2014

Contents

1. Introduction

The Brain Organization Simulation System (BOSS) developed at Stony Brook University models electrical activity in neuron networks. It can simulate up to 131 billion synapses and 16,000 dendritic synapses per neuron. It runs on 1,024 nodes of an IBM Blue Gene/P supercomputer, each of which has two gigabytes of memory. The most recent version of BOSS, V8, uses the Izhikevich integrate-and-fire neuron model, a more biologically accurate one than the threshold elements previously used.

BOSS runs in two stages: the initializer, INIT, generates a cerebellar model, and then the simulator, RUNSIM, processes activity in the model over time. Simulating one second of activity with 100 billion synapses takes two to four hours, depending on the neurons' firing frequencies. It is thus important to verify the model's accuracy before running BOSS. The Brain Visualizer, or Viz, renders models on a desktop computer. Users can explore the model and check that its neurons and neuritic fields are as expected. With the most recent version of Viz, they can also visualize the output of a BOSS simulation—data about neurons' firing patterns, voltages, and synaptic weights.

2. Version History

The first version of the Brain Visualizer was written in July 2010 by Pablo Montes Arango. It displayed somas and synaptic connections in 3D and allowed users to rotate, translate, and zoom the model. Individual somas could be selected to show their synapses and neuritic fields in a separate window.

The second version, by Jordan Ponzo and Vikas Ashok in October 2012, added more display options: somas could be hidden on a per-type basis, synapses could be drawn through intermediate locations instead of just direct connections between somas, and the colors used for soma types could be set with a configuration file. It also reduced memory usage with a data structure by Jack Zito: a table of all possible soma-to-soma pairs was replaced with an array of only the ones actually connected by a synapse.

The third version, by Remy Oukaour in December 2012, improved performance by caching data that did not need to be recalculated, and only redrawing the model when the view changed. The interface was made easier to use by allowing navigation states to be saved and restored, and by rearranging the controls and giving them more descriptive labels.

The fourth version was a complete rewrite using a different UI toolkit (FLTK instead of GLUT). It consolidated the visualization of somas, synapses, and neuritic fields into one window, with an interface that more closely followed OS conventions. Navigation was given a full undo/redo history. Performance was improved, supporting models with more than one million somas and 100 million synapses. It also added the option to view the dynamic firing data output by RUNSIM as animated color changes.

The current version of Viz has had its version number raised to 8, to match that of BOSS. It continues development from version 4, adding the ability to visualize gap junctions, somas' voltages, and synapses' weights.

3. Compilation

Viz is written in C++ and can be compiled on different operating systems, including Microsoft Windows, Apple Mac OS X, and GNU/Linux.

3.1. FLTK Library

Viz depends on a modified version of FLTK ("Fast Light Toolkit"), a cross-platform GUI toolkit written in C++ that supports OpenGL. The previous version used GLUT ("OpenGL Utility Toolkit"), which was less customizable and had extant bugs after eight years of no development. FLTK can be downloaded from www.fltk.org and built with the same compilers as Viz (Visual Studio, clang++, or g++).

Starting with a copy of FLTK v1.3.3, the file FL\Enumerations.H, line 30:
```
//#define FLTK_ABI_VERSION 10303
```
has been uncommented:
```
#define FLTK_ABI_VERSION 10303
```
This enables features which break the application binary interface (ABI) of v1.3.x. FLTK v1.4.0 is expected to enable the new ABI by default.

3.2. Compilation Flags

It is recommended to use the maximum optimization possible for your chosen compiler ("`/O2`" for Visual Studio, "`-O3`" for clang++ or g++).

By default Viz represents coordinates as single-precision (32-bit) floating-point values. To use 16-bit signed integers instead, define `SHORT_COORDS` during compilation. To use 32-bit integers, define `INT_COORDS`. The About dialog will indicate the coordinate type. Short coordinates use less memory, but floating-point values are usually faster on dedicated graphics processors. Short coordinates may not be able to store the full range of coordinates for some models.

The interface uses a 12-point font by default. To use a 16-point font, which is easier to read on high-DPI displays, define `LARGE_INTERFACE` during compilation.

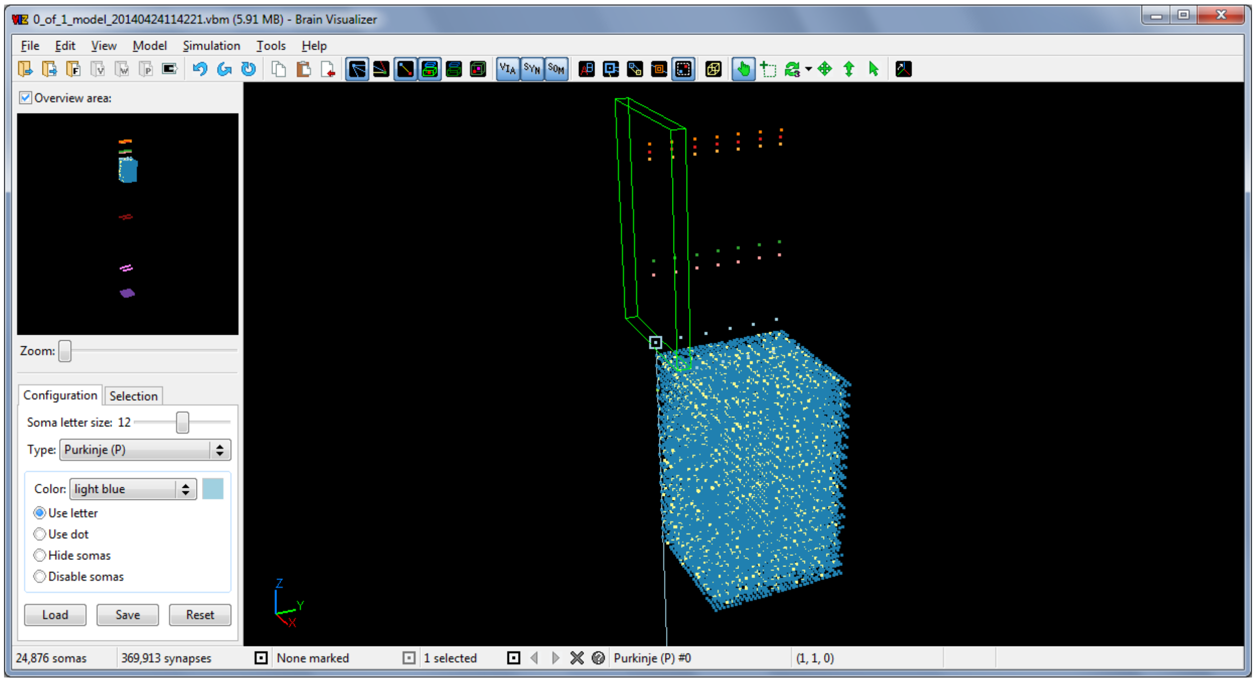
3.3. Visual Studio on Windows

Follow these instructions to build FLTK for Windows using Visual Studio 2010, 2012, or 2013:

1. Extract the contents of viz\lib\fltk-1.3.3-mod.zip to fltk-1.3.3-mod.
2. Open fltk-1.3.3-mod\ide\VisualC2012\fltk.sln in Visual Studio.
3. Set the solution configuration to Release.
4. Set the solution platform to x64 or x86, depending on which architecture you plan to use.
5. Build the fltk, fltkgl, and fltkimages projects.

After FLTK is built, follow these instructions to build Viz:

1. Copy the .lib files (fltk.lib, fltkgl.lib, fltkimages.lib, fltkpng.lib, and fltkz.lib) from fltk-1.3.3-mod\lib to viz\lib\win32\x64 or viz\lib\win32, depending on which architecture they were built for. You may need to rename fltkzlib.lib to fltkz.lib.
2. Navigate to viz\ide\vs2010, viz\ide\vs2012, or viz\ide\vs2013, depending on which version of Visual Studio you are using, and open Brain Visualizer.sln in Visual Studio.
3. Set the solution configuration to Release and the solution platform to x64 or x86.
4. Build the Brain Visualizer project.
5. Run viz\bin\win32\x64\Release\viz.exe or viz\bin\win32\Release\viz.exe, depending on which solution platform was used.



0_of_1_model_20140424114221.vbm (5.91 MB) - Brain Visualizer

Soma letter size: 12
Type: Purkinje (P)
Color: light blue
(●) Use letter
( ) Use dot
( ) Hide somas
( ) Disable somas

24,876 somas | 369,913 synapses | None marked | 1 selected | Purkinje (P) #0 | (1, 1, 0)

Figure 1: A screenshot of Viz running in Windows 7.

These are the custom properties needed for the Brain Visualizer project:

- General → Platform Toolset = v100 for 2010, v110 for 2012, or v120 for 2013
- General → Output Directory = ..\..\bin\win32\$(Configuration)\
- General → Intermediate Directory = ..\..\tmp\win32\$(Configuration)\
- Debugging → Working Directory = ..\..\config
- C/C++ → Additional Include Directories = ..\..\include;..\..\res
- C/C++ → Optimization → Optimization = Maximize Speed (`/O2`)
- C/C++ → Optimization → Enable Intrinsic Functions = Yes (`/Oi`)
- C/C++ → Optimization → Favor Size or Speed = Favor fast code (`/Ot`)
- C/C++ → Preprocessor → Preprocesor Definitions = `WIN32;NOMINMAX;_CRT_SECURE_NO_WARNINGS;NDEBUG;_WINDOWS;%(PreprocessorDefinitions)`
- C/C++ → Code Generation → Floating Point Model = Fast (`/fp:fast`)
- C/C++ → Advanced → Disable Specific Warnings = 4068;4351 (`/wd"4068;4351"`)
- Linker → Additional Library Directories = ..\..\lib\win32\x64 or ..\..\lib\win32
- Linker → Input → Additional Dependencies = fltkimages.lib;fltkpng.lib;fltkz.lib;fltk.lib;fltkgl.lib;opengl32.lib;glu32.lib;comctl32.lib;%(AdditionalDependencies)
- Linker → Input → Ignore Specific Default Libraries = libcmt.lib;%(IgnoreSpecificDefaultLibraries)

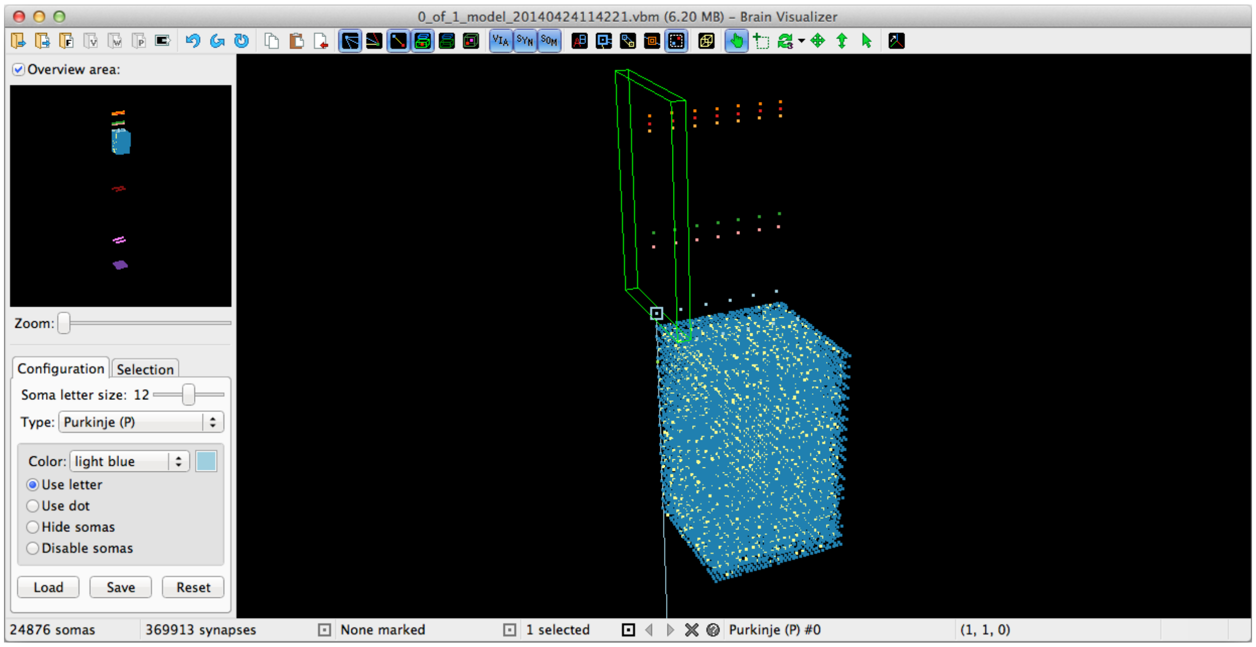
3.4. clang++ on Mac OS X

Follow these instructions to build FLTK for Mac OS X using clang++ (which is included in the Command Line Tools package for Xcode 5):

1. Extract the contents of viz/lib/fltk-1.3.3-mod.zip to fltk-1.3.3-mod.
2. Navigate to fltk-1.3.3-mod and run the command:
   ```
   make
   ```

After FLTK is built, follow these instructions to build Viz:

1. Copy the .a files (libfltk.a, libfltk_gl.a, libfltk_images.a, and libfltk_png.a) from fltk-1.3.3-mod/lib to viz/lib/osx.
2. Navigate to viz/ide/clang and run the command:
   ```
   make viz
   ```
3. Run the viz/bin/osx/Brain Visualizer application.



0_of_1_model_20140424114221.vbm (6.20 MB) – Brain Visualizer

Soma letter size: 12
Type: Purkinje (P)
Color: light blue

24876 somas | 369913 synapses | None marked | 1 selected | Purkinje (P) #0 | (1, 1, 0)

Figure 2: A screenshot of Viz running in Mac OS X 10.9.

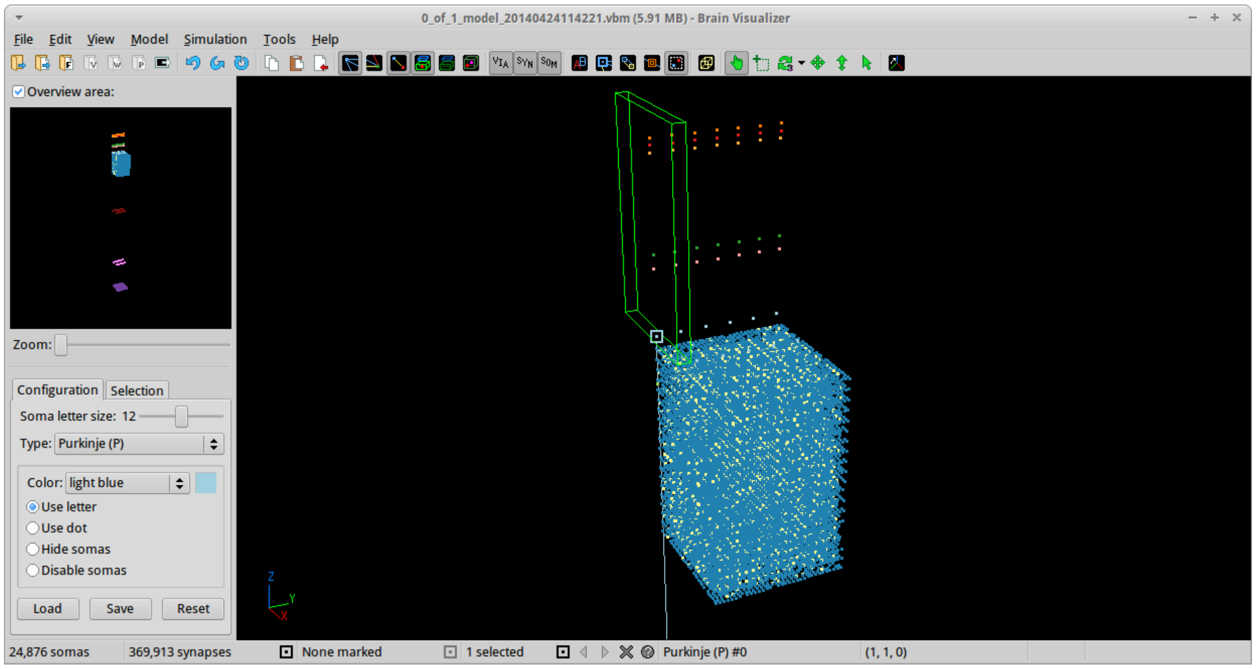
3.5. g++ on Linux

Follow these instructions to build FLTK for Linux using g++:

1. To ensure that all dependencies are installed, run the command:
```
sudo apt-get install g++ make freeglut3-dev zlib1g-dev libpng-dev
libxpm-dev libx11-dev libxft-dev libxinerama-dev libfontconfig1-dev
x11proto-xext-dev
```
2. Extract the contents of viz/lib/fltk-1.3.3-mod.zip to fltk-1.3.3-mod.
3. Navigate to fltk-1.3.3-mod and run the commands:
```
chmod +x configure
sudo make
```

After FLTK is built, follow these instructions to build Viz:

1. Copy the .a files (libfltk.a, libfltk_gl.a, and libfltk_images.a) from fltk-1.3.3-mod/lib to viz/lib/linux.
2. Navigate to viz/ide/gcc and run the command:
```
make viz
```
3. Run viz/bin/linux/viz.



0_of_1_model_20140424114221.vbm (5.91 MB) - Brain Visualizer

Soma letter size: 12
Type: Purkinje (P)
Color: light blue
[✓ Use letter] Use dot Hide somas Disable somas

24,876 somas | 369,913 synapses | None marked | 1 selected | Purkinje (P) #0 | (1, 1, 0)

Figure 3: A screenshot of Viz running in Xubuntu Linux 14.04.

4. File Formats

Most files that Viz reads are interpreted as lines of ASCII text with platform-dependent line breaks. Binary files have their own specifications.

4.1. Binary Model Files

Model files are generated by INIT to define the locations of somas and synapses in a particular model. Model files use the .vbm extension (for “Viz binary model”). This is a binary format, with all numbers represented as variable-length integers in network byte order (big-endian).

Variable-length integers (varints) allow small, common values to take up fewer bytes than large ones. An unsigned 64-bit integer can be encoded in 1 to 9 bytes; a signed 32-bit integer can be encoded in 1 to 5 bytes.

The leading bits in the first byte of a varint indicate how many additional bytes there are. The byte starts with zero to seven 1s, followed by a 0; any additional bits are used to encode data. The first byte may also be eight 1s. Decoding is as follows:

First byte	Unsigned	Signed
0xxxxxxx	Read 0 more bytes	Read 0 more bytes
10xxxxxx	Read 1 more byte	Read 0 more bytes and negate
110xxxxx	Read 2 more bytes	Read 1 more byte

1110xxxx	Read 3 more bytes	Read 1 more byte and negate
11110xxx	Read 4 more bytes	Read 2 more bytes
111110xx	Read 5 more bytes	Read 2 more bytes and negate
1111110x	Read 6 more bytes	Read 3 more bytes
11111110	Read 7 more bytes	Read 3 more bytes and negate
11111111	Read 8 more bytes	Read 4 more bytes and negate if the leading bit is 1

Table 1: Lookup table of initial bytes for decoding variable-length integers.

The first five bytes of a model file are `07 52 4A 56 F7` in hexadecimal. This is a signature identifying the file as a .vbm file. Next is a single-byte unsigned integer *v*, which is the file format version. Version 2 is described here. 0 is not a valid version. Next is a null-terminated string *s*, which is an optional comment string for additional information. The string *s* may be empty, meaning it consists only of the terminating byte `00`.

Next is an unsigned varint *n*, which is the number of cell types defined by the following *n* bytes. Each type definition takes one byte: an ASCII character c_i, which is the letter used to represent all cells of the type with index *i*.

Next is an unsigned varint *m*, which is the number of somas defined by the following bytes, and an unsigned varint *f*, which is the total number of neuritic fields belonging to the somas. (In version 1 files, *f* is not present.) Each soma definition takes seven varints: *t k x y z a d*. The unsigned soma type index is *t*. The unsigned soma ID is *k*. The signed coordinates of the cell are *x*, *y*, and *z*. The unsigned number of axonal fields for the cell is *a*, and the unsigned number of dendritic fields is *d*.

The bytes after a soma definition define axonal and dendritic fields for the cell. Each field definition takes six signed varints: x_1 x_2 y_1 y_2 z_1 z_2. These are the coordinates of the diagonal of the axis-aligned bounding box for the neuritic field.

Next after the *m* soma definitions is an unsigned varint *p*, which is the number of synapses defined by the following bytes. Each synapse definition starts with an unsigned varint *k* and a single-byte integer *v*. The synapse ID is *k*, and the *via* point flag is *v*. If *v* is 0, the synapse does not have a *via* point and its definition takes five more varints: *a d x y z*. The unsigned axonal soma ID is *a* and the unsigned dendritic soma ID is *d*. The signed coordinates of the synapse are *x*, *y*, and *z*. If *v* is 1, the synapse has a *via* point and its definition takes eight more varints: *a d* v_x v_y v_z *x y z*. The signed coordinates of the synapse's *via* point are v_x, v_y, and v_z. The other five integers are the same as for synapses without a *via* point.

Next after the *p* synapse definitions is an unsigned varint *g*, which is the number of gap junctions defined by the following bytes. Each gap junction definition takes five varints: s_1 s_2 *x y z*. The two unsigned somas' IDs are s_1 and s_2. The signed coordinates of the gap junction are *x*, *y*, and *z*.

4.2. Text Model Files

Model files used to be generated by INIT in a text-based format, but the current binary format takes up less space. Viz still supports either format.

Legacy model files use the .txt extension. Tokens are separated by one or more whitespace characters. The “#” character comments out all text until the end of the line. Blank lines and entirely commented lines are ignored.

The first line has a single integer n, which is the number of cell types. The following n lines each have a single integer t followed by a single letter c. The type index, t, increases from 0 to $n - 1$. The letter, c, is used to represent all cells of that type.

The next line has a single integer m, which is the number of somas defined by the following lines. Each soma is defined by a line that has seven integers: $t\ k\ x\ y\ z\ a\ d$. The soma type index is t. The soma ID is k. The coordinates of the cell are x, y, and z. The number of axonal fields for the cell is a, and the number of dendritic fields is d.

The next $a + d$ lines after a cell definition line define axonal and dendritic fields for the cell. Each line has six integers: $x_1\ x_2\ y_1\ y_2\ z_1\ z_2$. These are the coordinates of the diagonal of the axis-aligned bounding box for the neuritic field.

The line after the m cell definitions has a single integer p, which is the number of synapses defined by the following p lines. Synapses without a *via* point are defined by a line that has six integers: $k\ a\ d\ x\ y\ z$. The synapse ID is k. The axonal soma ID is a and the dendritic soma ID is d. The coordinates of the synapse are x, y, and z. Synapses with a *via* point are defined by a line that starts with a single integer k, followed by the letter “v”, followed by eight more integers: $a\ d\ v_x\ v_y\ v_z\ x\ y\ z$. The coordinates of the synapse’s *via* point are v_x, v_y, and v_z. The other six integers are the same as for synapses without a *via* point.

The line after the p synapse definitions has a single integer g, which is the number of gap junctions defined by the following g lines. Each gap junction is defined by a line that has five integers: $s_1\ s_2\ x\ y\ z$. The two somas’ IDs are s_1 and s_2. The coordinates of the gap junction are x, y, and z.

For backwards compatibility with files generated before the introduction of gap junctions, everything related to them can be left out, and it will be assumed that there are no gap junctions.

4.3. Compressed Model Files

Both binary and text model files can be compressed with the DEFLATE algorithm and stored using the gzip file format. Programs like 7-Zip and GNU Gzip can compress .vbm or .txt files into .gz files, which may be less than 20% of their original size. Viz supports opening compressed model files directly.

Compressed model files use the .vbm.gz or .txt.gz extensions.

4.4. Firing Spike Files

Firing spike files are generated by RUNSIM to record the firing spikes of the somas being simulated.

Firing spike files use the .txt extension. Tokens are separated by one or more whitespace characters. The "#" character comments out all text until the end of the line. Blank lines and entirely commented lines are ignored.

The first line optionally has the letter "v" followed by a single integer *v*, expected to be 2, which is the version of the firing spike file format. This is only for backwards compatibility with old firing spike files, and will be removed in a future version of Viz.

The second line has a single integer *t*, expected to be from 100 to 10,000, which is the number of microseconds (µs) per cycle. The third line has a single integer *m*, which is the number of somas defined by the corresponding model file. The fourth line has a single integer *c*, which is the number of cycles defined by the following *c* lines.

Each cycle definition line starts two integers: *s t*. The time instance being defined by the line is *t*. The number of somas that fired at cycle *t* is *s*. In version 2 files, these are followed by *s* pairs of integers, or 2*s* integers total. In each pair p_i, the first integer is the ID of the firing soma, and the last integer is the state of the firing soma: a bitmask of 1 for natural firing, 2 for forced firing, 4 for binary firing, and 8 for firing suppression. (Suppressions are firing events that aren't associated with an Izhikevich firing spike; rosette somas can exhibit them.) In files without a version line, implicitly treated as version 1, there are simply *s* firing soma IDs without corresponding firing states, and the states are assumed to be 1 for all firing events.

4.5. Voltage Files

Voltage files are generated by RUNSIM to record the voltages of particular somas being simulated. Not all the somas are recorded, since that would usually be too much data. The somas to record are specified by the selected soma report file, viz_selected_cells.txt.

Voltage files use the .txt extension. Tokens are separated by one or more whitespace characters. The "#" character comments out all text until the end of the line. Blank lines and entirely commented lines are ignored.

The first line has a single integer *m*, which is the number of somas defined by the corresponding model file. The second line has a single integer *n*, which is the number of somas which have voltages recorded in the file. The third line has *n* integers, which are the IDs of the somas with recorded voltages. The fourth line has a single integer *c*, which is the number of cycles defined by the following *c* lines.

Each cycle definition line starts with a single integer *t*, which is the time instance being defined by the line. It is followed by *n* pairs of integers, or 2*n* integers total. In each pair p_i, the first integer is the voltage of the *i*th soma at cycle *t* in millivolts (mV), and the last integer is the state of the firing soma: a bitmask of 1 for natural firing, 2 for forced firing, 4 for binary firing, and 8 for firing suppression.

4.6. Weight Files

Weight files are generated by RUNSIM to record the weight changes of particular synapses being simulated. Not all the synapses are recorded, since that would usually be too much data. The synapses to record are specified by the selected soma report file, viz_selected_cells.txt.

Weight files use the .txt extension. Tokens are separated by one or more whitespace characters. The "#" character comments out all text until the end of the line. Blank lines and entirely commented lines are ignored.

The first line has a single integer *m*, which is the number of somas defined by the corresponding model file. The second line has a single integer *y*, which is the number of synapses defined by the corresponding model file. The third line has a single integer *c*, which is the number of cycles defined by the corresponding firing spike file. The following lines define weight changes for individual synapses.

Each change definition line has seven integers: $t\ k\ a\ d\ r\ w_b\ w_a$. The cycle index is *t*. The synapse ID is *k*. The axonal soma ID is *a* and the dendritic soma ID is *d*. The RUNSIM synapse ID is *r*. (It is not used by Viz, but is recorded to reflect the reordering of synapses between INIT and RUNSIM.) The weight before the change is w_b and the weight after the change is w_a. Weights are measured in microvolts (μV).

4.7. Pruning Files

Pruning files are generated by RUNSIM to record the weight changes which occur as a result of pruning. They are formatted identically to weight files.

4.8. Selected Soma Report Files

Soma report files are generated by Viz to record the selected somas. BOSS uses the soma report file in its directory to identify somas for which to output voltage and weight files.

Soma report files use the .txt extension. Tokens are separated by one or more whitespace characters. The "#" character comments out all text until the end of the line. Blank lines and entirely commented lines are ignored.

The first line has a single integer *m*, which is the number of somas described by the following *m* lines. Each soma description line has five integers: *t k x y z*. The soma type index is *t*. The soma ID is *k*. The coordinates of the cell are *x*, *y*, and *z*.

After the *m* soma descriptions, the optional line "-1 0 0 0 0" prevents BOSS from finding the parents and children of the *m* somas.

4.9. Selected Synapse Report Files

Synapse report files are generated by Viz to record the synapses to or from the selected somas. This limited subset of synapses can be more easily read than an entire model file.

Synapse report files use the .txt extension. Tokens are separated by one or more whitespace characters. The "#" character comments out all text until the end of the line. Blank lines and entirely commented lines are ignored.

The first line has a single integer *m*, which is the number of somas described by the following *m* lines. Each soma description line has five integers: *t k x y z*. The soma type index is *t*. The soma ID is *k*. The coordinates of the cell are *x*, *y*, and *z*.

The line after the *m* cell descriptions has a single integer *y*, which is the number of synapses described by the following *y* lines. Each synapse description line has six integers: *k a d x y z*. The synapse ID is *k*. The axonal soma ID is *a* and the dendritic soma ID is *d*. The coordinates of the synapse are *x*, *y*, and *z*.

4.10. Configuration Files

Configuration files use the .cfg extension. Fields are separated by single ":" characters. The "#" character comments out all text until the end of the line. Blank lines and entirely commented lines are ignored.

Each line must have at least four fields. The first is a single letter indicating which soma type the line is configuring. The second is the name of the type. The third is the color used to represent the type. There are 24 valid colors: red, maroon, pink, magenta, orange, tan, brown, yellow, goldenrod, olive, yellow-green, green, lime, cyan, turquoise, blue, sky blue, light blue, indigo, violet, purple, lavender, white, and gray. Unknown colors are treated as gray. The fourth is the display status of the type: letter, dot, hidden, or disabled. Any extra fields are ignored.

Viz includes a hard-coded default configuration equivalent to the following configuration file:

```
P:Purkinje:light blue:letter
N:Granule:blue:dot
G:Golgi:yellow-green:letter
B:Basket +x:green:letter
A:Basket -x:pink:letter
S:Outer stellate +x:red:letter
T:Outer stellate -x:tan:letter
I:Outer stellate A:orange:letter
C:Climbing fiber:lavender:letter
M:Mossy fiber:purple:letter
R:Rosette tip:yellow:dot
D:Dentate nucleus:maroon:letter
```

5. Internal Data Structures

The data structures used by Viz are tuned for performance, to allow storing as much data as possible and to speed up the common operations on that data.

5.1. Coordinates

Depending on how Viz is built, coordinates may be represented in different ways and have different ranges. As single-precision floating-point values (the default format), they can unambiguously range from $-2^{24} - 1$ to $2^{24} - 1$. As 16-bit integers (specified by defining `SHORT_COORDS` during compilation), they range from -2^{15} to $2^{15} - 1$ (32,767). As 32-bit integers (specified by defining `INT_COORDS` during compilation), they range from -2^{32} to $2^{32} - 1$.

5.2. Brain Models

A brain model defined by a model file has numerous soma types, somas, neuritic fields, synapses, and gap junctions.

Soma types are stored in a single array with between 1 and $2^8 - 1$ (255) soma types.

Somas are stored in a single array with between 1 and $2^{24} - 2$ (more than 16 million) somas. Although a 32-bit integer is used to store the number of somas, selecting somas requires their indexes to be encoded as 24-bit RGB colors, with a single reserved background color. Each soma may have between 0 and $2^8 - 1$ axonal fields, and likewise for dendritic fields. Neuritic fields are stored in a single array with between 0 and $2^{32} - 1$ fields.

Synapses are stored in a single array with between 0 and $2^{32} - 1$ (more than 4 billion) synapses.

Gap junctions are stored in a single array with between 0 and $2^{32} - 1$ gap junctions.

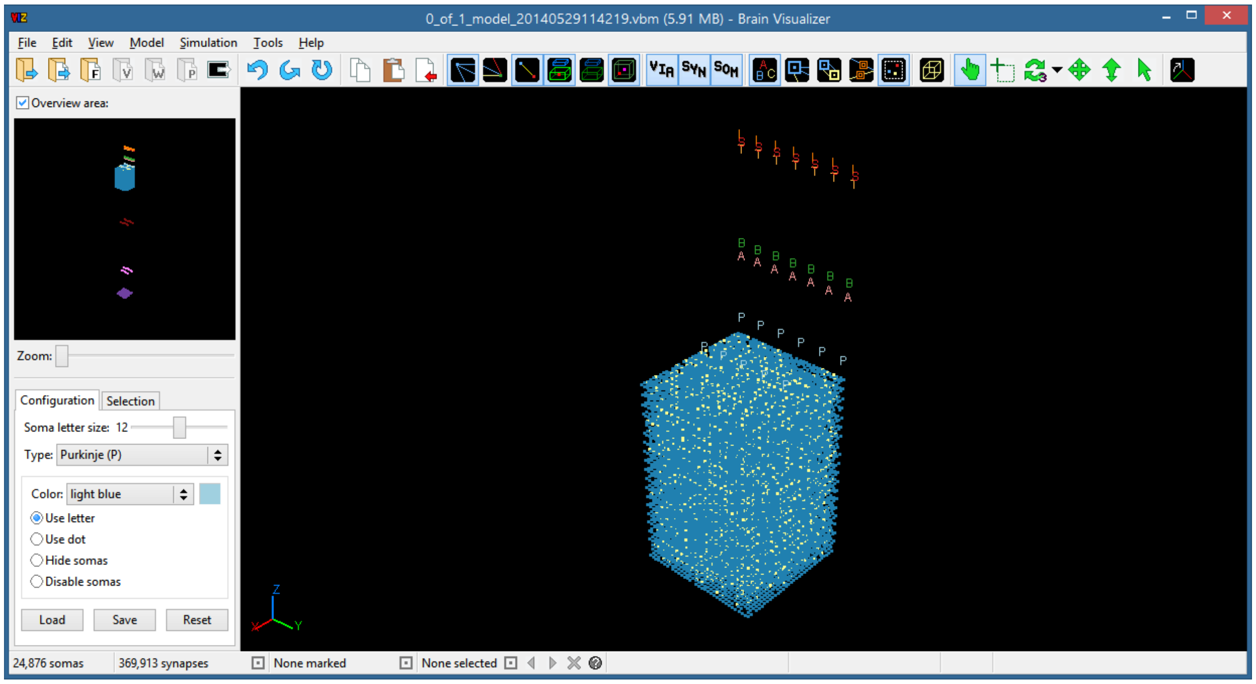
5.3. Simulation Data

Each set of simulation data (firing spikes, soma voltages, and synapse weights) has between 1 and $2^{32} - 1$ cycles. Cycle-specific data is stored in a single array. The data for each cycle is generally an associative map from soma indexes to their simulation data.

Voltages are recorded for a subset of active somas, which are stored in a single array with between 0 and $2^{32} - 1$ somas.

Synapse weights range from -2^{14} to $2^{14} - 1$ (16,383) microvolts (μV), which is −163.84 to 163.83 millivolts (mV).

6. User Interface



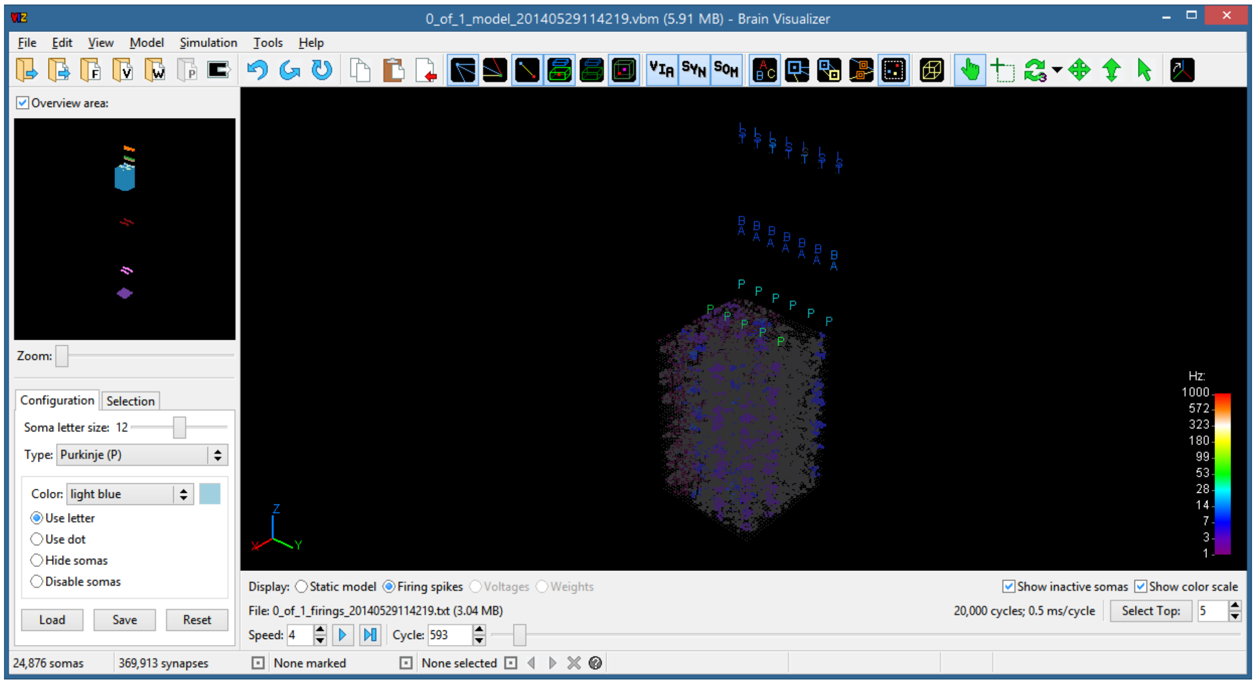
0_of_1_model_20140529114219.vbm (5.91 MB) - Brain Visualizer

Soma letter size: 12
Type: Purkinje (P)
Color: light blue
[✓ Use letter] Use dot Hide somas Disable somas

24,876 somas | 369,913 synapses | None marked | None selected

Figure 4: The Viz user interface in Windows 8 with a model file opened.

Viz uses a single window divided into sections: the menu bar at the very top, the toolbar below it, the sidebar on the left (containing the overview area at its top), the model area to the right, and the status bar at the bottom. The model area and overview area display the opened model; the status bar lists further information about the model; and the menu bar, toolbar, and sidebar contain controls that let the user interact with the model.



0_of_1_model_20140529114219.vbm (5.91 MB) - Brain Visualizer

☑ Overview area:

Zoom:

Soma letter size: 12

Type: Purkinje (P)

Color: light blue

[✓ Use letter] Use dot Hide somas Disable somas

Display: Static model [✓ Firing spikes] Voltages Weights

☑ Show inactive somas ☑ Show color scale

File: 0_of_1_firings_20140529114219.txt (3.04 MB)

20,000 cycles; 0.5 ms/cycle

Select Top: 5

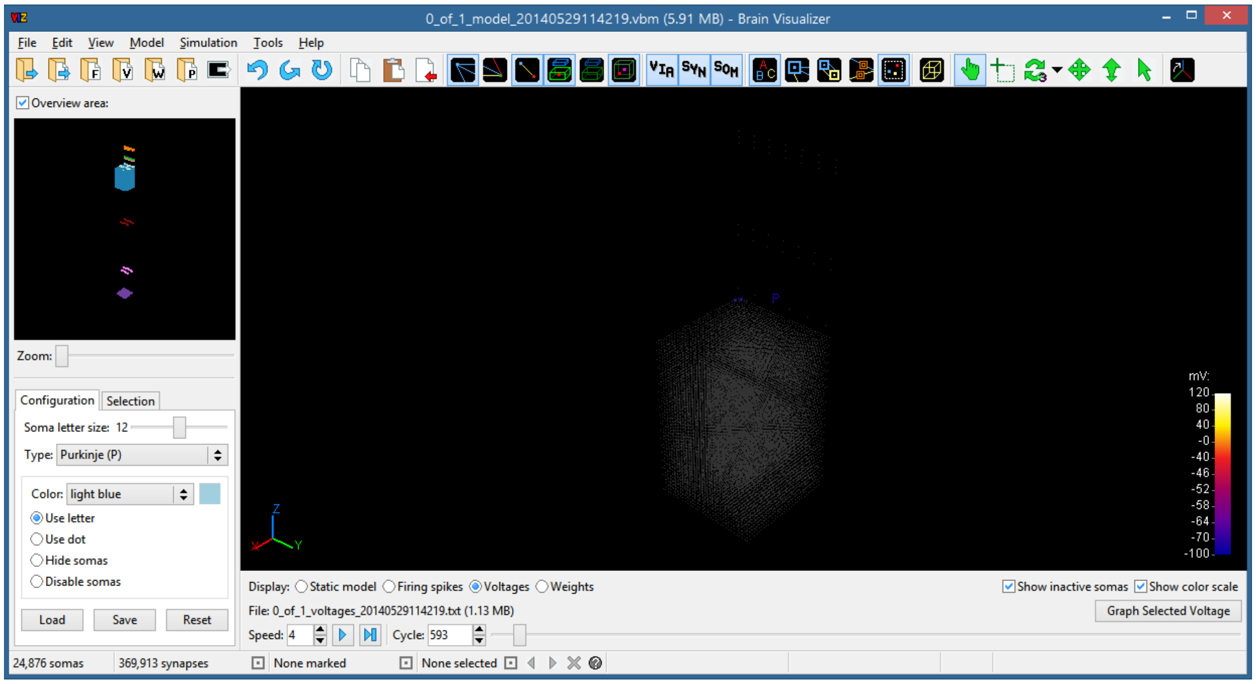
Speed: 4 Cycle: 593

24,876 somas 369,913 synapses None marked None selected

Hz: 1000 572 323 180 99 53 28 14 7 3 1

Figure 5: The Viz user interface with a firing spike file loaded.

When a firing spike file is loaded after a corresponding model file has been opened, the simulation bar appears below the model area, and the model area displays the model differently.



0_of_1_model_20140529114219.vbm (5.91 MB) - Brain Visualizer

☑ Overview area:

Zoom:

Soma letter size: 12

Type: Purkinje (P)

Color: light blue

[✓ Use letter] Use dot Hide somas Disable somas

Display: Static model Firing spikes [✓ Voltages] Weights

☑ Show inactive somas ☑ Show color scale

File: 0_of_1_voltages_20140529114219.txt (1.13 MB)

Speed: 4 Cycle: 593

24,876 somas 369,913 synapses None marked None selected

mV: 120 80 40 -0 -40 -46 -52 -58 -64 -70 -100

Figure 6: The Viz user interface with a voltage file and a weight file loaded.

After a firing spike file has been loaded, voltage and weight files can be loaded as well.

6.1. Menu Bar

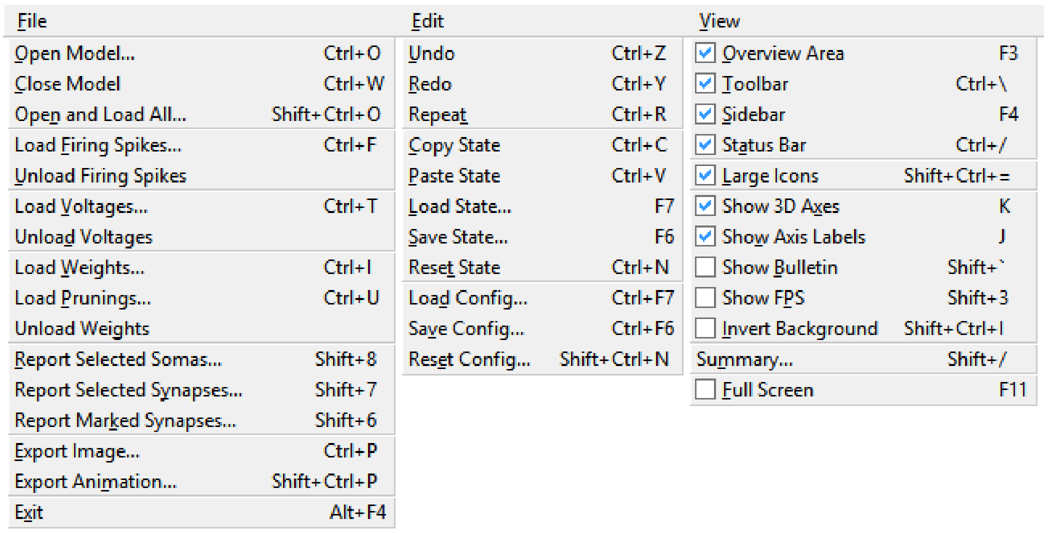
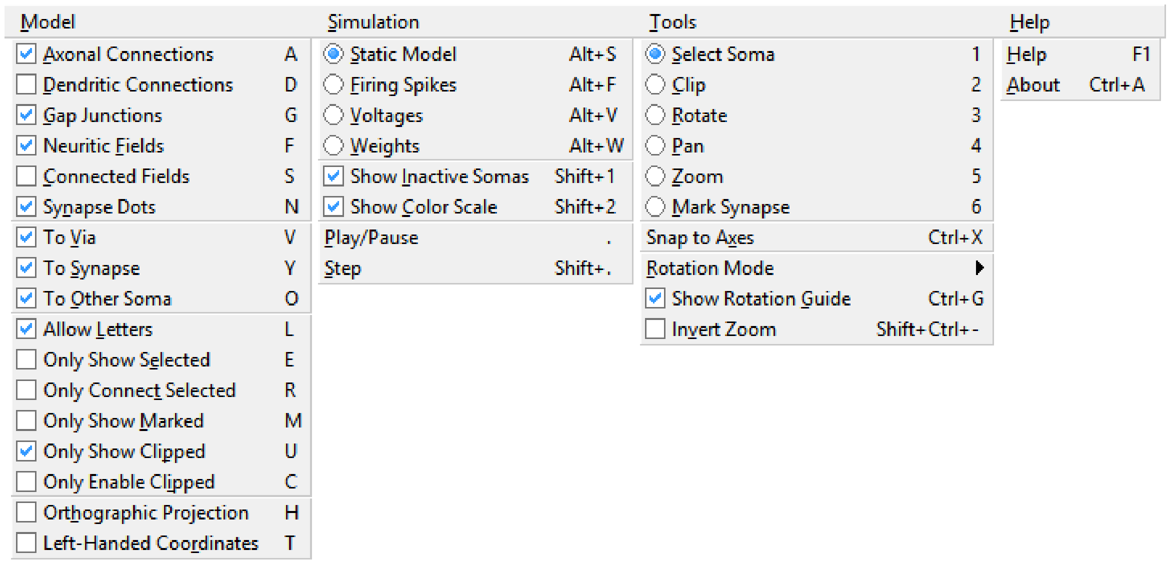


Figure 7: The first three menus in the menu bar.

The File menu contains commands for dealing with most types of files: models, firing spikes, voltages, weights, selected soma or synapse reports, and images. The Exit command is also located here on Windows by convention.

The Edit menu contains commands to manage the user's sequence of navigation actions. Actions can be undone or redone, and the present state can be kept on the clipboard for short-term storage or saved to a file for a longer term. Commands to manage configuration files are also located here.

The View menu contains commands to modify the interface's appearance. The overall interface can be set to imitate different operating systems. Certain parts of the user interface, and attributes of the model area, can be toggled on or off. The program can also enter full screen mode, which hides window chrome and takes up the whole screen.



Model	
✓ Axonal Connections	A
☐ Dendritic Connections	D
✓ Gap Junctions	G
✓ Neuritic Fields	F
☐ Connected Fields	S
✓ Synapse Dots	N
✓ To Via	V
✓ To Synapse	Y
✓ To Other Soma	O
✓ Allow Letters	L
☐ Only Show Selected	E
☐ Only Connect Selected	R
☐ Only Show Marked	M
✓ Only Show Clipped	U
☐ Only Enable Clipped	C
☐ Orthographic Projection	H
☐ Left-Handed Coordinates	T

Simulation	
● Static Model	Alt+S
○ Firing Spikes	Alt+F
○ Voltages	Alt+V
○ Weights	Alt+W
✓ Show Inactive Somas	Shift+1
✓ Show Color Scale	Shift+2
Play/Pause	.
Step	Shift+.

Tools	
● Select Soma	1
○ Clip	2
○ Rotate	3
○ Pan	4
○ Zoom	5
○ Mark Synapse	6
Snap to Axes	Ctrl+X
Rotation Mode	▶
✓ Show Rotation Guide	Ctrl+G
☐ Invert Zoom	Shift+Ctrl+-

Help	
Help	F1
About	Ctrl+A

Figure 8: The last four menus in the menu bar.

The Model menu contains options that affect how the model is displayed in the model area.

The Simulation menu contains options that affect how simulation data (firing spikes, voltages, or weights) is displayed in the model area, as well as commands to progress through the data automatically or step through it manually.

The Tools menu contains commands to interact with the model.

The Help menu contains commands to show information about the program. The Help dialog displays basic guidance from the file help.html, which should be in the same directory as the Viz executable file. The About dialog shows build information and development credits.

On Mac OS X, the Brain Visualizer application menu (left-most on the menu bar) contains commands dealing with the program itself, including the Exit item from the File menu and the About item from the Help menu. The Help menu has a search box to search the text of menu items.

6.2. Toolbar

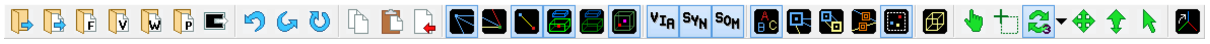


Figure 9: The toolbar, with large icons.

The toolbar contains buttons for the most useful commands. Each of them corresponds to a menu item. Some buttons execute a command when pressed; others toggle a setting on or off. Some buttons are disabled until a model file is open, or until a firing spike file is loaded.

- **Open Model:** Open a model file (using a platform-dependent file dialog).
- **Open and Load All:** Open a model file and load its corresponding firing spikes, voltages, and weights, if they exist.
- **Load Firing Spikes:** Load a firing spike file corresponding to the open model.
- **Load Voltages:** Load a voltages file corresponding to the open model.
- **Load Weights:** Load a weights file corresponding to the open model.
- **Export Image:** Export the current model area as a PNG, BMP, TGA, or PPM image.
- **Undo:** Undo the previous action (up to 50 times). Anything that modifies the navigation state (the position and orientation of the model in 3D space, and the set of selected somas) counts as an action.
- **Redo:** Redo the last undone action (up to 50 times).
- **Repeat:** Repeat the previous rotate, pan, or zoom action relative to the current state. This is useful for making incremental rotations in the same direction.
- **Copy:** Copy the current navigation state to the clipboard.
- **Paste:** Paste the saved navigation state from the clipboard. If no state has been saved, this acts like Reset.
- **Reset:** Reset the navigation state to the original when the model was first opened.
- **Axonal Connections:** Toggle showing axonal connections for selected somas (those from the selected somas to others).
- **Dendritic Connections:** Toggle showing dendritic connections for selected somas (those from other somas to the selected ones).
- **Gap Junctions:** Toggle showing gap junctions for selected somas.
- **Neuritic Fields:** Toggle showing the neuritic fields for selected somas.
- **Connected Fields:** Toggle showing the neuritic fields for somas connected to the selected somas. The connected fields are shown only if the Neuritic Fields option is on.
- **Synapse Dots:** Toggle showing the locations of synapses inside neuritic fields. The synapses are shown only if the Neuritic Fields option is on.
- AXN **To Axonal Soma:** Toggle whether synaptic connections pass through the axonal soma's location.
- VIA **To Via Point:** Toggle whether synaptic connections pass through the *via* point.

- **To Synapse:** Toggle whether synaptic connections pass through the synapse’s location.
- **To Dendritic Soma:** Toggle whether synaptic connections pass through the dendritic soma’s location.
- **Allow Letters:** Toggle allowing somas to be displayed as letters if their types are configured for it. This is off by default to improve performance.
- **Only Show Selected:** Toggle only showing selected somas (and the ones connected to them, if axonal or dendritic connections are shown).
- **Only Connect Selected:** Toggle only showing synaptic connections between pairs of selected somas.
- **Only Show Marked:** Toggle only showing marked synapses.
- **Only Show Clipped:** Toggle only showing somas inside the clipping region. The hidden outside somas will only be displayed if they are connected to a selected soma.
- **Orthographic Projection:** Toggle the use of orthographic projection. The default perspective projection shows further-away points as closer together, whereas orthographic projection does not distort distance.
- **Select:** Click a soma to select or deselect it. Hold down Shift or turn on Caps Lock while clicking to select multiple somas. Up to 1,000 somas can be selected at once. Click in empty space to deselect them. Hold down Ctrl while clicking a soma, or right-click it, to bring up a summary dialog with detailed information about the soma. Do the same in empty space for information about the whole model.
- **Clip:** Click and drag to draw a box around a group of somas. The box will then be projected into 3D space, and all somas outside of it will be clipped. Somas outside the clipping region can be hidden or disabled. Clipping also sets the pivot point for rotations to be in the center of mass of the clipping region.
- **Rotate:** Click and drag to rotate the model around the pivot point. By default, a wireframe sphere with three axes is superimposed on the model while rotating to make results clearer. There are five rotation modes, accessible via the dropdown arrow to the right of the Rotate button:
    - **2D Arcball:** Dragging left, right, up, and down rotates the model in the direction it was dragged. The initially clicked point is irrelevant; only the dragging direction affects the model.

  - **3D Arcball:** Clicking a point figuratively grabs a point on the sphere surrounding the clipping region (or the whole model, if it has not been clipped), and dragging moves that point to where it was dragged. This lets dragging in two dimensions affect the orientation around all three axes. This is the default mode.
  - **X-Axis:** Dragging perpendicular to the *x*-axis rotates the model around that axis. If the *x*-axis is nearly perpendicular to the screen, then clicking figuratively grabs a point on the cylinder surrounding the *x*-axis, and dragging moves that point to where it was dragged.
  - **Y-Axis:** Behaves like the *x*-axis mode, but for the *y*-axis.
  - **Z-Axis:** Behaves like the *x*-axis mode, but for the *z*-axis.
- **Pan:** Click and drag to pan (translate) the model in the desired direction. Panning too far away from the center can cause parallax distortion when using perspective projection.
- **Zoom:** Click and drag upwards to zoom in (making the model larger) and downwards to zoom out (making it smaller). Zooming is relative to the center of the clipping region. If Tools → Invert Zoom is on, the directions are reversed.
- **Mark Synapse:** Click a synapse dot to mark or unmark it. Hold down Shift or turn on Caps Lock while clicking to mark multiple synapses. Click in empty space to unmark them. Hold down Ctrl while clicking a synapse, or right-click it, to bring up a summary dialog with detailed information about the synapse.
- **Snap to Axes:** Rotate the model so that its axes are aligned straight horizontally and vertically.
- **Reset Settings:** Reset the menu and toolbar toggle settings to their default values.

6.3. Sidebar

The sidebar contains the overview area, as well as tabs containing controls for configuring the soma types, selecting sets of somas, and marking sets of synapses.

6.3.1. Overview Area

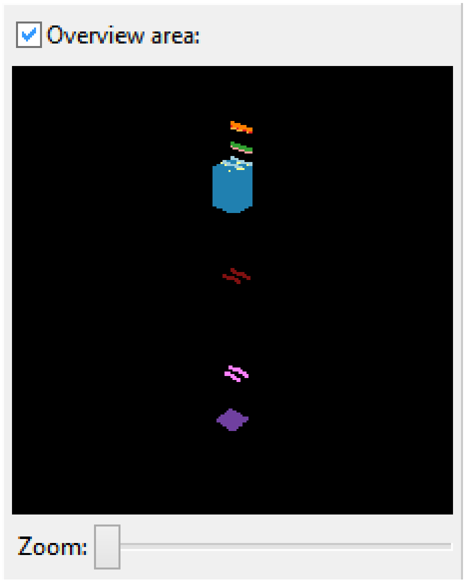
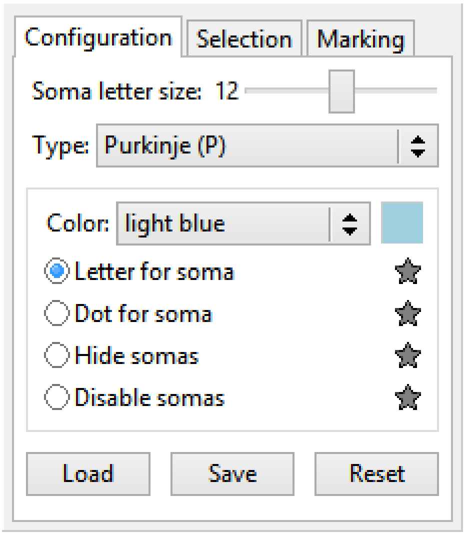


Figure 10: The overview area from the sidebar, fully zoomed out.

The overview area shows the entire model, including even the hidden or disabled somas. It zooms independently of the model area, and cannot be panned. Zooming completely out shows the entire model; zooming completely in focuses on the clipping region. The overview area can be minimized to improve performance and use less vertical space.

6.3.2. Configuration Tab


Configuration | Selection | Marking

Soma letter size: 12

Type: Purkinje (P)

Color: light blue

[✓ Letter for soma]
Dot for soma
Hide somas
Disable somas

Figure 11: The configuration tab from the sidebar.

From the configuration tab, each soma type can have its color and display state configured. There are four display states:

- **Letters:** Somas of this type will be displayed as a letter if Allow Letters is on.
- **Dots:** Somas of this type will be displayed as a dot, even if Allow Letters is on.
- **Hidden:** Somas of this type will only be displayed if they are connected to a selected soma or a marked synapse.
- **Disabled:** Somas of this type will not be displayed at all.

The star buttons next to each display state will set all soma types to use that state.

The size of the letters used to display somas can be set to 8, 10, 12, 14, or 16 points.

The configuration can be loaded from a file, saved to a file, or reset to its initially-loaded state. Overwriting an existing file by saving a new one with the same name is not allowed.

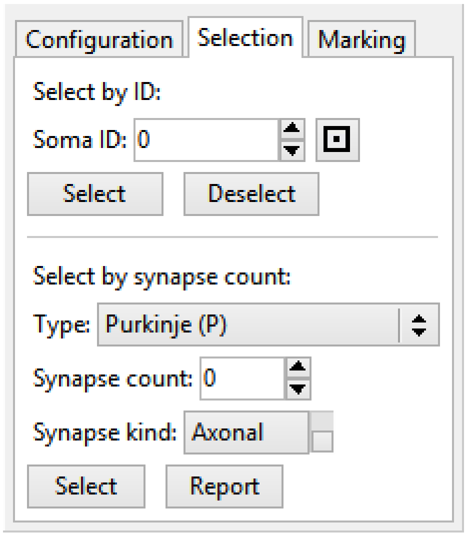
The initial configuration for an opened model is loaded automatically from the file viz.cfg. If no such file exists, one is generated with these default data:

Letter	**Name**	**Color**	**Display State**
P	Purkinje	light blue	letter
N	Granule	blue	dot
G	Golgi	yellow-green	letter
B	Basket +x	green	letter
A	Basket −x	pink	letter
S	Outer stellate +x	red	letter
T	Outer stellate –x	tan	letter
I	Outer stellate A	orange	letter
C	Climbing fiber	lavender	letter
M	Mossy fiber	purple	letter
R	Rosette tip	yellow	dot
D	Dentate nucleus	maroon	letter

Table 2: The soma type configuration data used if no configuration file is available.

(The granule and rosette tip somas are shown as dots, not as letters, because they are the most numerous and drawing them as letters slows performance.)

6.3.3. Selection Tab



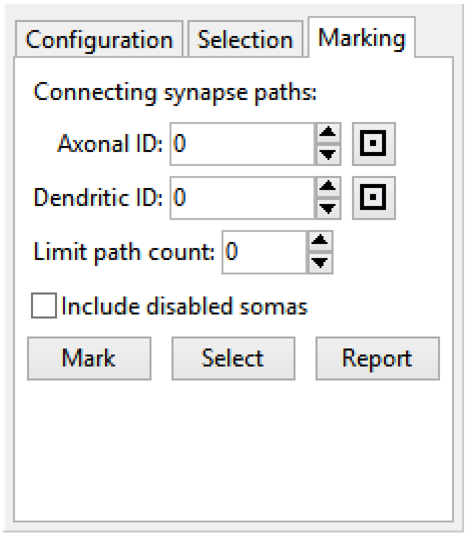
Select by ID:
Soma ID: 0

Select by synapse count:
Type: Purkinje (P)
Synapse count: 0
Synapse kind: Axonal

Figure 12: The selection tab from the sidebar.

From the selection tab, somas can be selected or deselected by ID. Somas can also be selected according to whether they have a certain count of axonal or dendritic synapses. Since this may find more somas than can be selected, the somas can also be reported in a text file.

6.3.4. Marking Tab



Connecting synapse paths:
Axonal ID: 0
Dendritic ID: 0
Limit path count: 0
☐ Include disabled somas

Figure 13: The marking tab from the sidebar.

From the marking tab, synapses can be marked along connecting paths. By choosing an axonal and dendritic soma to start and end the paths, the intermediate synapses can be marked or reported in a text file, or the intermediate somas can be selected.

6.4. Model Area

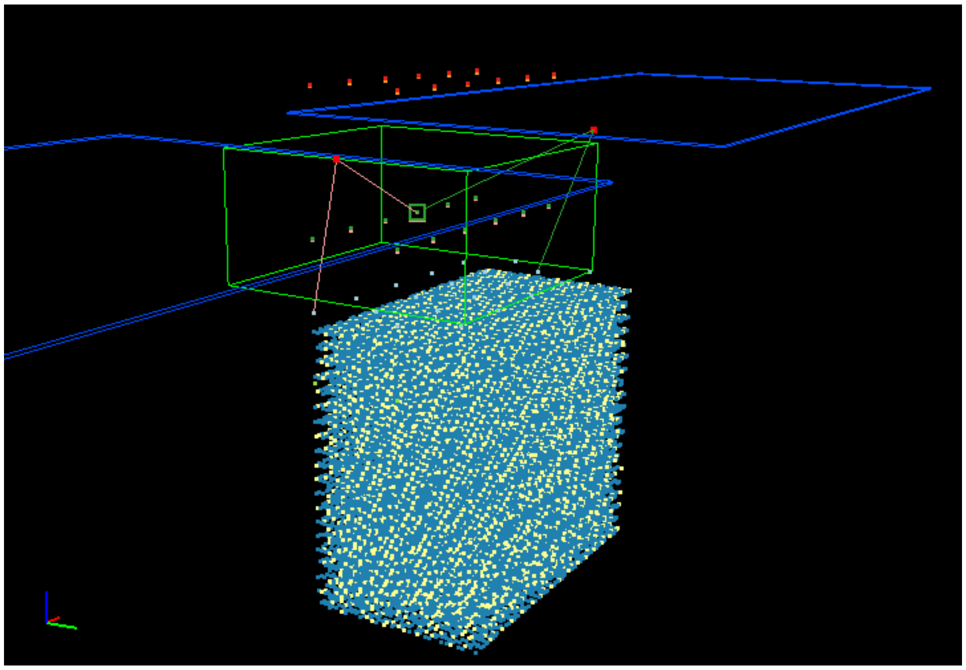


Figure 14: The model area, showing the upper portion of a model emulating a slice of cerebellar cortex. Most of the visible somas are granule cells and mossy fiber rosette tips. Two basket cells are selected; their synapses and neuritic fields are visible as lines and boxes, respectively.

The model area displays the cerebellar model from an opened file. It uses a perspective projection by default. Each soma is represented as a dot or letter in 3D space, colored according to its type. Selected somas are larger dots with a surrounding circle. They may also have their synaptic connections and fields displayed. Synapses are represented as straight lines connecting somas, colored according to the type of the axonal soma. Neuritic fields are represented as rectangular prisms: blue for axonal ones and green for dendritic ones, with the contained axonal synapses as red dots and dendritic synapses as purple dots. Marked synapses are represented as orange concentric circles.

In the bottom-left corner is an axis guide corresponding to the model's orientation. The *x*-axis is red, the *y*-axis green, and the *z*-axis blue. Viz uses a right-handed coordinate system by default.

In the top-left corner is a bulletin listing the selected somas (hidden by default). When the model area is displaying simulation data, the bulletin also shows the current cycle.

In the top-right corner is an estimate of the frames per second that can be rendered (hidden by default). Larger models take longer to render individual frames.

In the bottom-right corner, when the model area is displaying simulation data, is a color scale showing how soma colors correspond to the quantities being simulated over time.

6.5. Status Bar

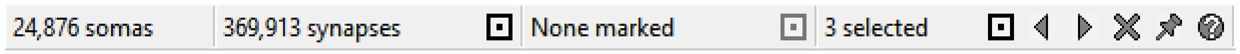

24,876 somas | 369,913 synapses | None marked | 3 selected

Figure 15: The left half of the status bar, with three selected somas.

The status bar displays counts of the somas and synapses in the entire open model, as well as counts of the selected somas and marked synapses. The buttons next to the synapse count, marked count, and selected count generate text file reports of their respective items. When multiple somas are selected, the one being described can be chosen with the arrow buttons next to the selected count, or deselected with the X button. The pin button sets the selected soma as the pivot point for rotations. (If no somas are selected, it will reset the pivot point to the center of mass of the clipping region.) The question mark button brings up a summary dialog with information about the selected soma. (If no somas are selected, it will bring up information about the entire model.)

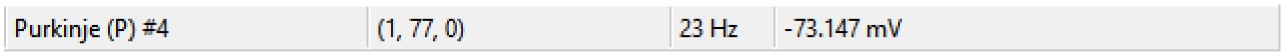

Purkinje (P) #4 | (1, 77, 0) | 23 Hz | -73.147 mV

Figure 16: The right half of the status bar, describing a selected soma.

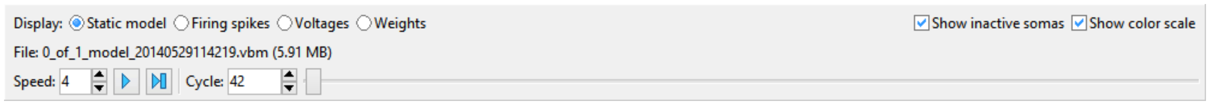
For a selected soma, the status bar displays its type, ID, and coordinates. If the soma has recorded firing spikes, its current firing frequency is displayed. If it has recorded voltages, its current voltage is displayed. Coordinates are measured in micrometers (μm) from the origin.

6.6. Simulation Bar

The simulation bar appears only when a firing spike file has been loaded. It allows the user to choose a simulation cycle time instance at which to view the model. If voltage or weight files have been loaded, they can also be chosen on the simulation bar.

After choosing an initial time instance, the user can press the Play button to animate the firing pattern. The animation speed can range from 1 (1 cycle per second) to 10 (100 cycles per second). The user can also press or hold down the Step button to manually advance the time instance.

6.6.1. Static Model



Display: [✓ Static model] Firing spikes Voltages Weights ☑ Show inactive somas ☑ Show color scale

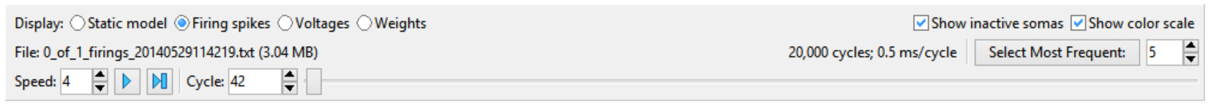
File: 0_of_1_model_20140529114219.vbm (5.91 MB)

Speed: 4 Cycle: 42

Figure 17: The simulation bar, while displaying the static model.

When displaying the static model, simulation bar shows the model's size along the three axes. The model area has the same display as if no simulation data were loaded.

6.6.2. Firing Spikes



Display: Static model [✓ Firing spikes] Voltages Weights ☑ Show inactive somas ☑ Show color scale

File: 0_of_1_firings_20140529114219.txt (3.04 MB) 20,000 cycles; 0.5 ms/cycle Select Most Frequent: 5

Speed: 4 Cycle: 42

Figure 18: The simulation bar, while displaying firing spikes.

When displaying firing spikes, the simulation bar shows the number of cycles and the milliseconds per cycle, and has controls to generate text file reports of the current or average firing frequencies of the enabled somas, and to select the most frequently firing somas. (The reports exclude disabled somas, and if any somas are selected, only the selected somas will be included in the reports.) If "Display value for somas" is enabled, somas which would otherwise be displayed as letters will be displayed as their frequency values in Hertz.

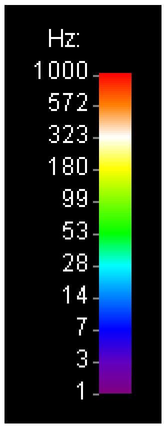
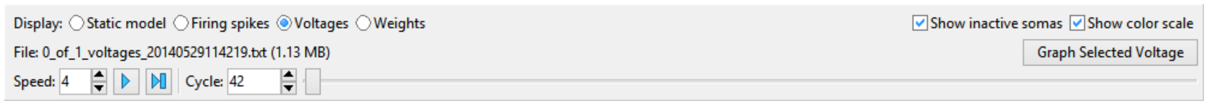


Figure 19: The color scale for firing spikes. The logarithm of the frequency determines the hue.

For the chosen cycle time, firing somas are displayed as large colored dots, and inactive ones are small gray dots. The inactive somas can be hidden so as not to obscure the firing ones. The firing somas' colors correspond to their frequency, estimated from the past 1,000 time instances. (If there are not enough past instances, at least 5 are assumed to exist so that the initial estimates are not too high.) As time passes without a soma firing again, its color fades to gray.

6.6.3. Voltages


Display: ◯ Static model ◯ Firing spikes [✓ Voltages] ◯ Weights ✓ Show inactive somas ✓ Show color scale

File: 0_of_1_voltages_20140529114219.txt (1.13 MB)

Figure 20: The simulation bar, while displaying voltages.

When displaying voltages, the simulation bar shows the number of somas with recorded voltages, and has a button to graph the selected soma's voltage. If "Display value for somas" is enabled, somas which would otherwise be displayed as letters will be displayed as their voltage values in millivolts.

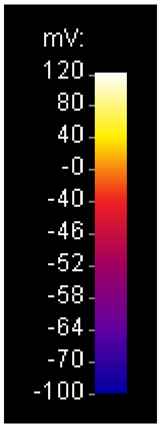


Figure 21: The color scale for voltages. More hues are used for voltages from −70 to −40 mV.

The voltage graph appears in a new window. Red dots indicate natural firing spikes, magenta dots indicate forced firings, orange dots indicate binary firings, and green dots indicate suppressions. Clicking anywhere on the graph shows a tooltip with the cycle time instance and voltage at that point. Scrolling up and down with the mouse wheel, or pressing or holding the zoom buttons, zooms in and out of the graph's *x*-axis. The range of the *y*-axis can also be adjusted. The "Export Image" button exports the entire graph as an image. The "Detect Peaks" button generates a text file report of peaks in the graph, and marks them with blue dots. Multiple graph windows can be opened at once.

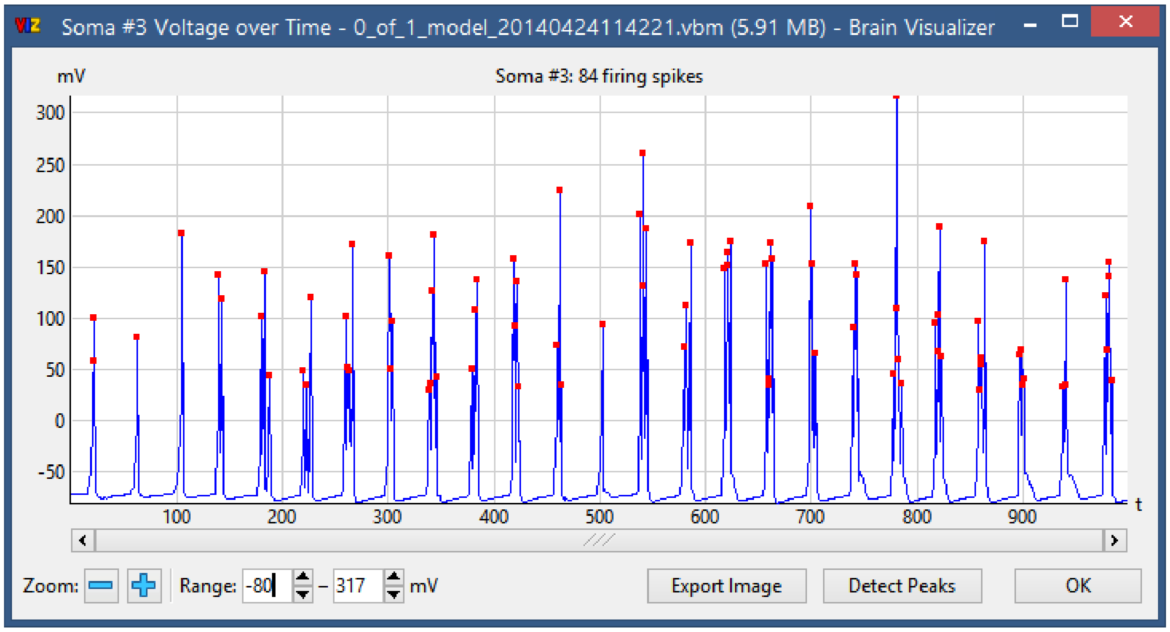


Figure 22: The voltage graph, showing the voltage of a Purkinje cell.

6.6.4. Weights

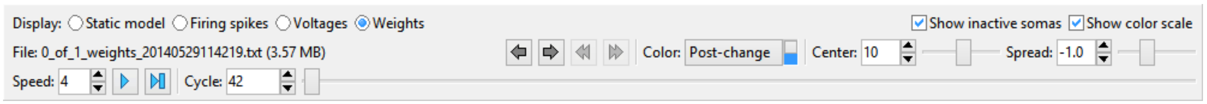

Display: ○ Static model ○ Firing spikes ○ Voltages ● Weights
☑ Show inactive somas ☑ Show color scale
File: 0_of_1_weights_20140529114219.txt (3.57 MB)
Color: Post-change
Center: 10
Spread: -1.0
Speed: 4
Cycle: 42

Figure 23: The simulation bar, while displaying weights.

When displaying weights, the simulation bar has buttons to skip to the previous or next time instance at which any weights changed, or at which a weight was changed involving the currently selected soma. It also has controls to toggle the color scale to correspond to pre-change or post-change weights, and to adjust the scale's center and spread. (Positive spreads are more linear, negative ones are more logarithmic.) The center and spread values are initially set to maximize the number of colors used, given the range of weights present in the file.

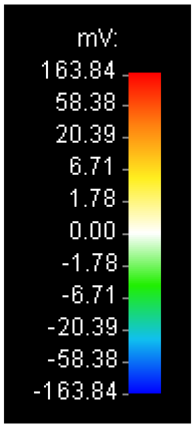


Figure 24: The color scale for weights. Cool colors represent negative weights, warm colors represent positive ones.

If prunings data is loaded after weights data, the weight changes due to prunings are added to the existing weight changes.

7. Performance

Viz has been tested on a variety of systems, from a laptop manufactured in 2008 with a dual-core 32-bit processor and 4 GB of RAM, to a desktop with a quad-core 64-bit Intel Xeon processor and 64 GB of RAM. The latter system was used to gather the data below, since testing very large models requires a correspondingly large amount of memory.

7.1. Loading Times

The 53.3 GB file viz_input_X1200xY6000_20131009202848.txt, with 4.37 million somas and nearly 1.1 billion synapses, takes 4 minutes and 33 seconds to load from a solid-state hard drive. A more reasonably sized file, the 2.0 GB viz_input_X720xY1000_20131009012923.txt, with 437 thousand somas and 46.6 million synapses, takes 11 seconds to load. Almost all the time is spent loading the synapses; overall Viz loads over 4 million synapses per second.

7.2. Memory Usage

Viz takes up approximately 10 to 12 MB of memory when it first starts, before opening any files. The 64-bit builds use slightly more than their 32-bit counterparts, but are capable of opening larger files since they can address more RAM. Builds with floating-point (32-bit) coordinates need around 80% as much memory compared to the input file size; those with short (16-bit) coordinates need only 55%.

File Name	File Size (MB)	Somas	Synapses	Memory Usage (MB)			
				x86 SC	x86 FP	x64 SC	x64 FP
None	0	0	0	10.55	10.56	12.43	11.80
viz_input_X120xY100 _20131009011601.txt	5.36	7,440	163,552	22.25	24.21	25.29	26.47
viz_input_X360xY100 _20131009011613.txt	15.30	22,167	414,113	30.05	35.38	33.35	37.47
viz_input_X360xY200 _20131009011627.txt	45.13	44,459	1,149,520	51.47	65.65	54.80	68.01
viz_input_X360xY300 _20131009011643.txt	91.43	65,624	2,274,627	82.70	110.54	87.38	113.30
viz_input_X360xY400 _20131009011705.txt	155.38	87,612	3,813,107	125.84	171.23	130.76	174.60
viz_input_X360xY500 _20131009011732.txt	234.29	109,829	5,705,086	178.05	246.22	183.22	249.39
viz_input_X720xY500 _20131009011813.txt	542.56	220,112	12,679,186	373.14	523.48	380.93	527.11
viz_input_X720xY600 _20131009011929.txt	760.69	262,164	17,561,094	507.04	714.45	516.03	718.35
viz_input_X720xY700 _20131009012113.txt	1,019.61	306,501	23,321,502	664.37	939.00	674.32	943.61
viz_input_X720xY800 _20131009012324.txt	1,333.09	350,792	30,314,184	855.09	1,210.49	865.78	1,215.71
viz_input_X720xY900 _20131009012607.txt	1,688.84	392,990	38,195,074	1,069.52	—	1,081.22	1,521.36
viz_input_X720xY1000 _20131009012923.txt	2,067.26	437,295	46,586,740	1,297.10	—	1,309.73	1,846.59
viz_input_X1200xY1000 _20131009013330.txt	3,339.37	729,176	74,210,394	—	—	2,077.88	2,933.60
viz_input_X1200xY2000 _20131009014104.txt	12,685.56	1,457,531	266,043,045	—	—	7,282.41	10,343.92
viz_input_X1200xY3000 _20131009021032.txt	22,946.90	2,182,778	472,403,983	—	—	12,876.12	18,309.40
viz_input_X1200xY4000 _20131009024959.txt	33,420.25	2,912,478	680,615,814	—	—	18,519.34	26,345.46
viz_input_X1200xY5000 _20131009190347.txt	44,017.51	3,641,476	890,376,475	—	—	24,204.03	34,441.41
viz_input_X1200xY6000 _20131009202848.txt	54,548.08	4,366,322	1,097,004,014	—	—	29,804.36	42,417.13

Table 3: Comparing memory usage for different input files and Viz builds, built with Visual Studio 2012 on Windows 7. All measurements are averages of three runs. Memory measurements marked FP use floating-point coordinates, while those marked SC use short coordinates. Empty cells for the 32-bit builds indicate that the files needed more memory than a 32-bit process can allocate.

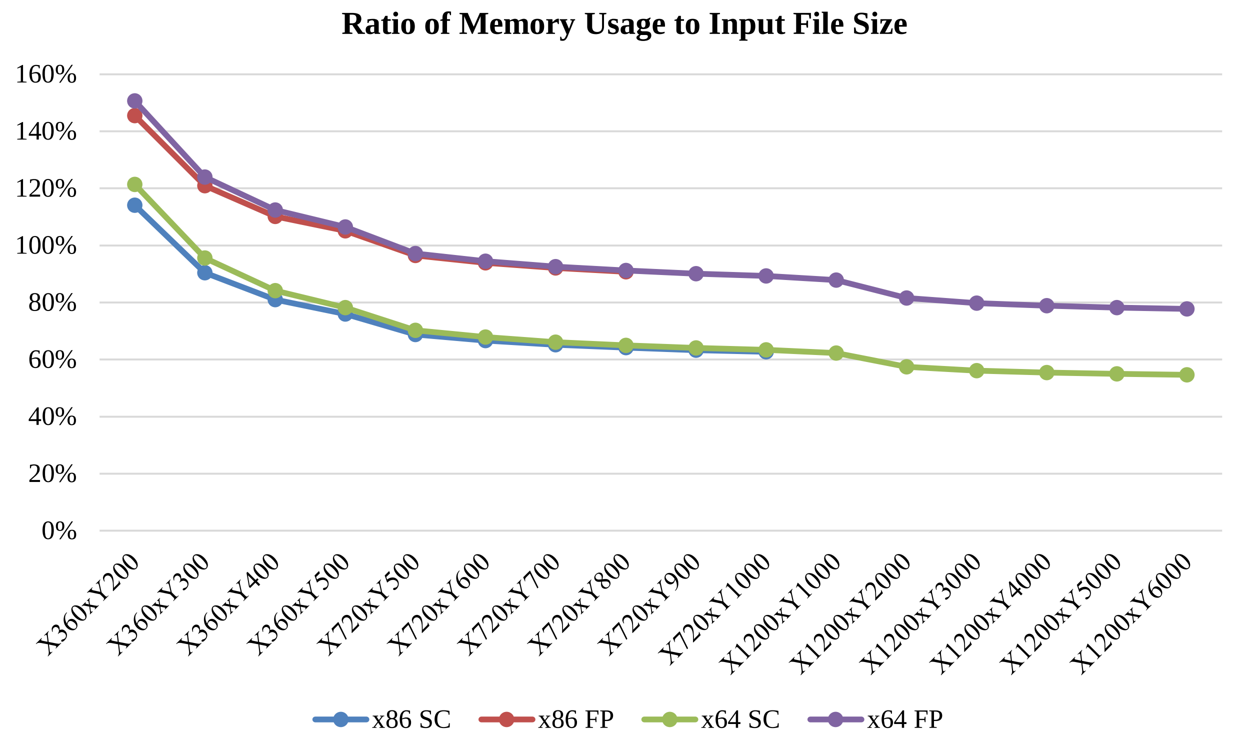


Figure 25: A chart of the ratio of memory usage to input file size for some of the input files in the above table. The very small files have exceptionally high ratios and are not present on the chart. For very large files, the floating-point coordinate (FP) builds level off at around 80%, and the short coordinate (SC) builds at 55%.

The most significant factor for memory usage is the number of synapses in a model, followed by the number of somas (which is one or two orders of magnitude smaller). The data structures for storing synapses and somas have been optimized to be as small as possible. To avoid 64-bit builds using extra memory, Viz often uses 32-bit array offsets instead of 64-bit pointers.

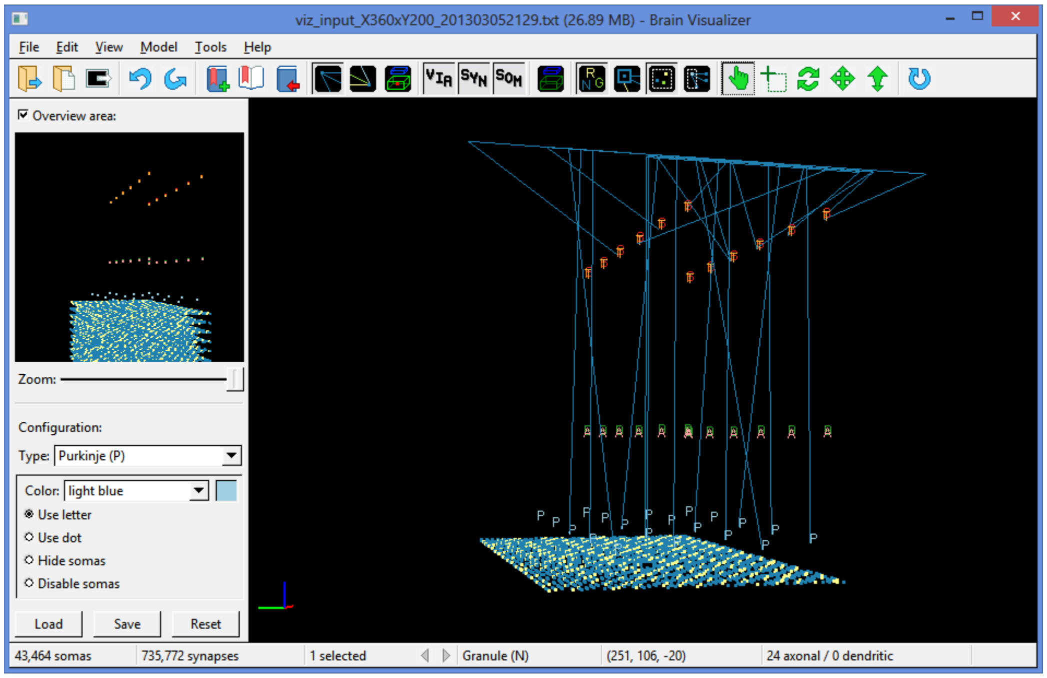
7.3. Frame Rate

For input files with up to around a million somas or half a billion synapses, the frame rate when interacting with models is smooth. For the files with closer to a billion synapses, actions like rotating and panning are choppier, but still responsive. Features that iterate over all the synapses, like when displaying an information dialog about the model, take a few seconds to complete. Interactive actions can be sped up by hiding some somas, either by disabling their types or by zooming in or clipping to a particular section.

A. Screenshots

These screenshots illustrate particular visualization options, as well as how to use the tools for interacting with the model area and configuring the model.

A.1. Synaptic Connections



viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer

☑ Overview area:

Zoom:

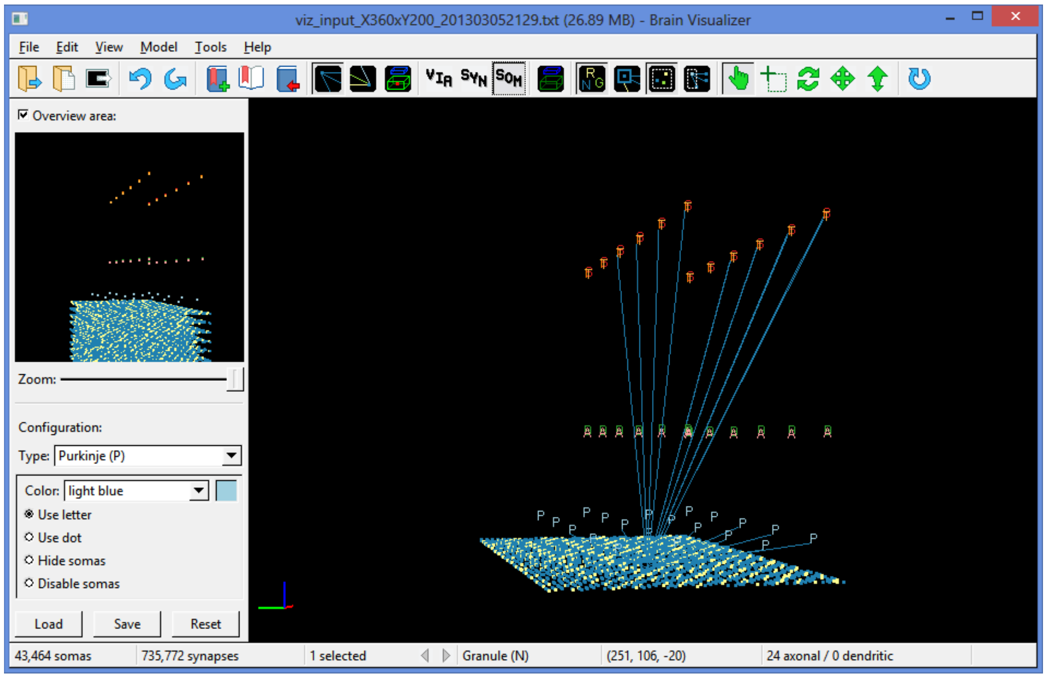
Configuration:

Type: Purkinje (P)

Color: light blue
(● Use letter) ○ Use dot ○ Hide somas ○ Disable somas

43,464 somas | 735,772 synapses | 1 selected | Granule (N) | (251, 106, -20) | 24 axonal / 0 dendritic

Figure 26: With the To Via, To Synapse, and To Other Soma options on, axonal connections are drawn from the selected granule cell, through its single *via* point and many synapses (located on its parallel fiber), to the dendritic somas (of types P, S, T, and I). The model has been clipped to show only the top part of a cerebellar cortex model.

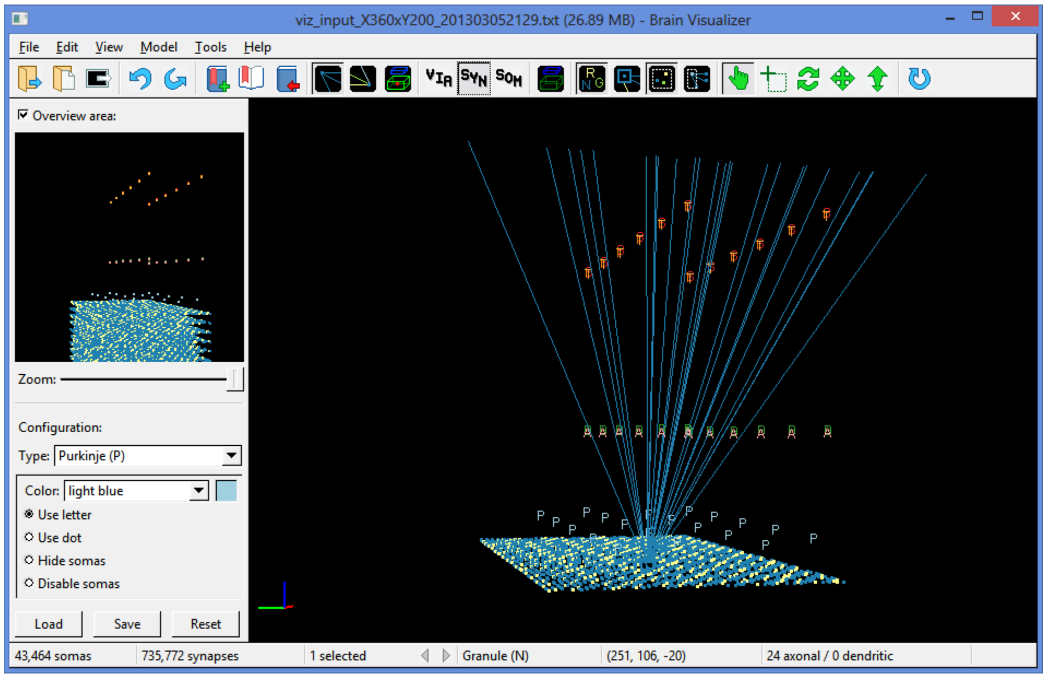

viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer

Overview area: [✓]

Configuration:
Type: Purkinje (P)
Color: light blue
[✓ Use letter]
Use dot
Hide somas
Disable somas

43,464 somas | 735,772 synapses | 1 selected | Granule (N) | (251, 106, -20) | 24 axonal / 0 dendritic

Figure 27: With only the To Other Soma option on, axonal connections are drawn directly from the selected granule cell to its dendritic somas (P, S, T, and I).

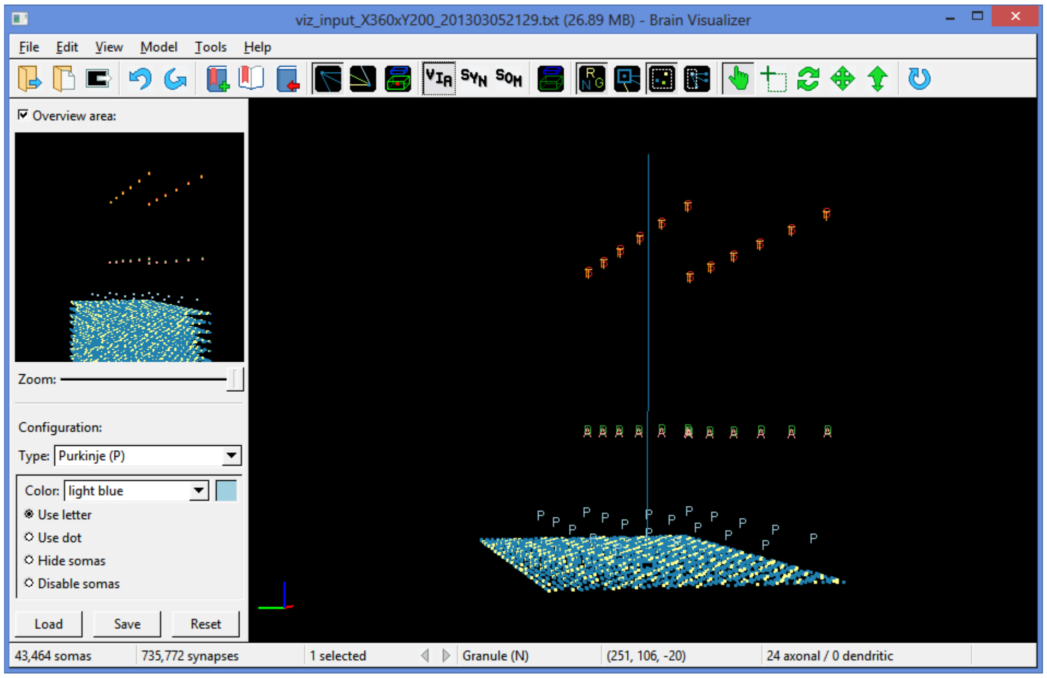

viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer

Overview area: [✓]

Configuration:
Type: Purkinje (P)
Color: light blue
[✓ Use letter]
Use dot
Hide somas
Disable somas

43,464 somas | 735,772 synapses | 1 selected | Granule (N) | (251, 106, -20) | 24 axonal / 0 dendritic

Figure 28: With only the To Synapse option on, axonal connections are drawn directly from the selected granule cell to its axonal synapses (all on the same parallel fiber).

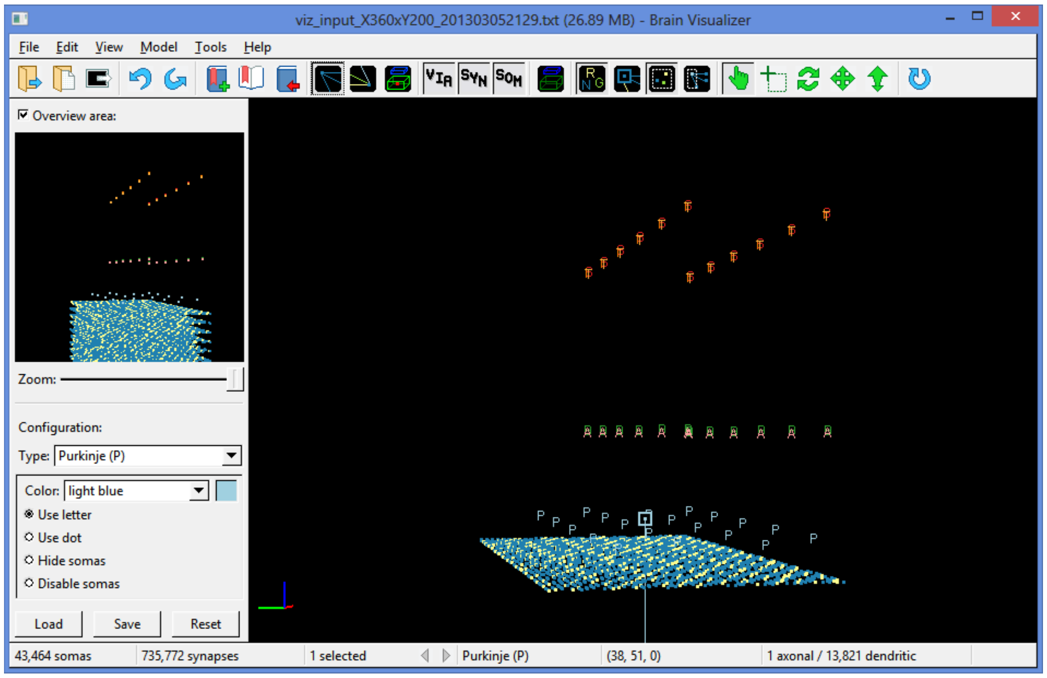

viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer
File
Edit
View
Model
Tools
Help
Zoom:
Load
Save
Reset

☑ Overview area:

Configuration:
Type: Purkinje (P)
Color: light blue
[✓ Use letter]
Use dot
Hide somas
Disable somas

43,464 somas | 735,772 synapses | 1 selected | Granule (N) | (251, 106, -20) | 24 axonal / 0 dendritic

Figure 29: With only the To Via option on, axonal connections are drawn directly from the selected granule cell to its *via* point, where its axon bifurcates to form a parallel fiber (not shown).

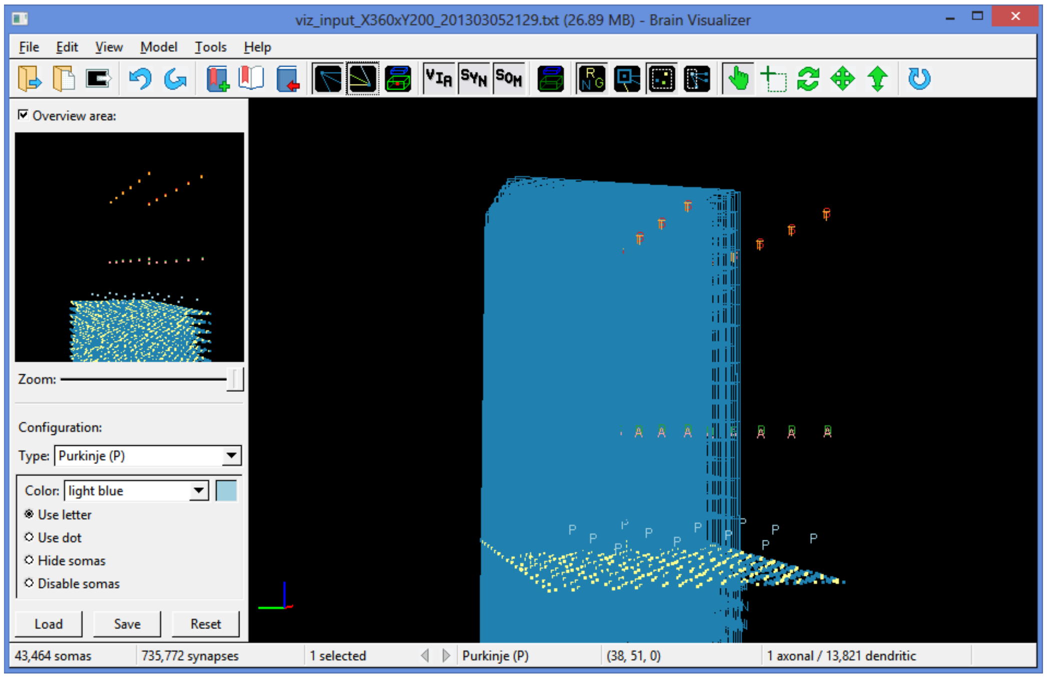

viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer
File
Edit
View
Model
Tools
Help
Zoom:
Load
Save
Reset

☑ Overview area:

Configuration:
Type: Purkinje (P)
Color: light blue
[✓ Use letter]
Use dot
Hide somas
Disable somas

43,464 somas | 735,772 synapses | 1 selected | Purkinje (P) | (38, 51, 0) | 1 axonal / 13,821 dendritic

Figure 30: With the Axonal Connections option on and the Dendritic Connections option off, only the axonal connection from the selected Purkinje cell is drawn.



viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer

[✓ Overview area:]

Zoom:

Configuration:

Type: Purkinje (P)

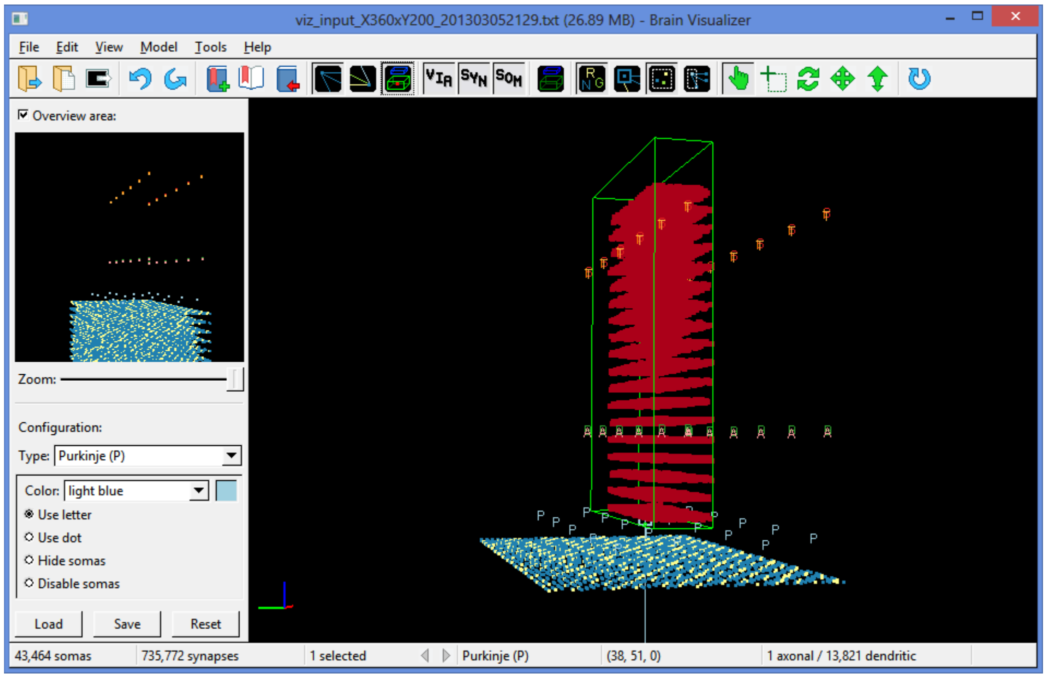
Color: light blue

[✓ Use letter] Use dot Hide somas Disable somas

43,464 somas | 735,772 synapses | 1 selected | Purkinje (P) | (38, 51, 0) | 1 axonal / 13,821 dendritic

Figure 31: With both the Axonal Connections and the Dendritic Connections options on, the myriad axonal and dendritic connections of the selected Purkinje cell are drawn.

A.2. Neuritic Fields



viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer

[✓ Overview area:]

Zoom:

Configuration:

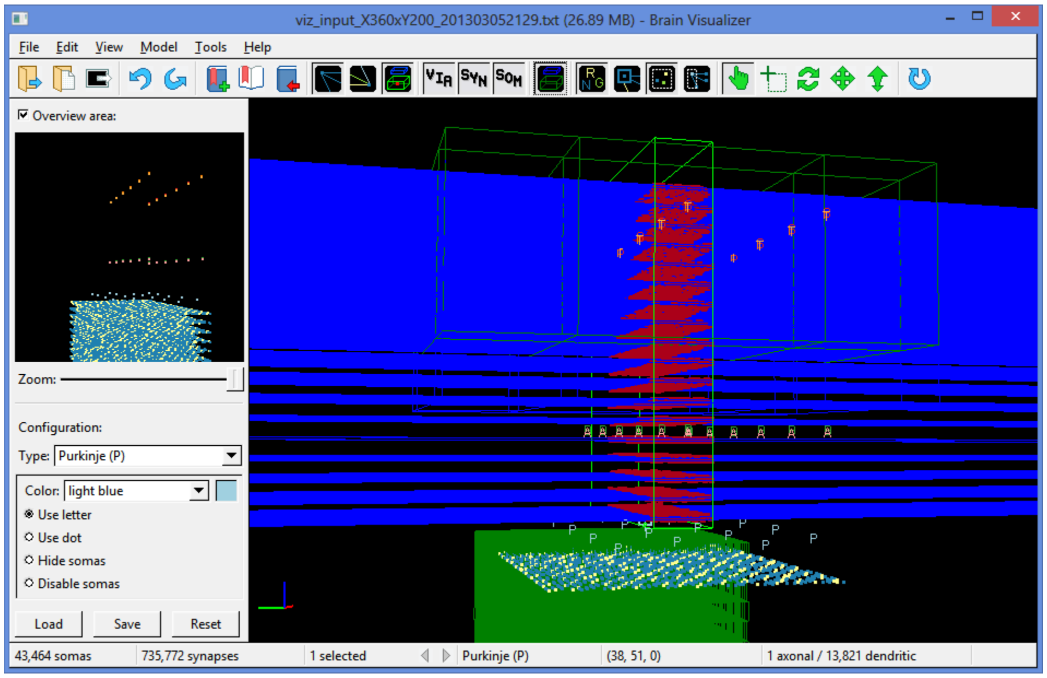
Type: Purkinje (P)

Color: light blue

[✓ Use letter] Use dot Hide somas Disable somas

43,464 somas | 735,772 synapses | 1 selected | Purkinje (P) | (38, 51, 0) | 1 axonal / 13,821 dendritic

Figure 32: With the Neuritic Fields option on, the neuritic fields and synapse locations of the selected Purkinje cell are drawn. The axonal field and synapses are outside the visible area, below the region being shown.

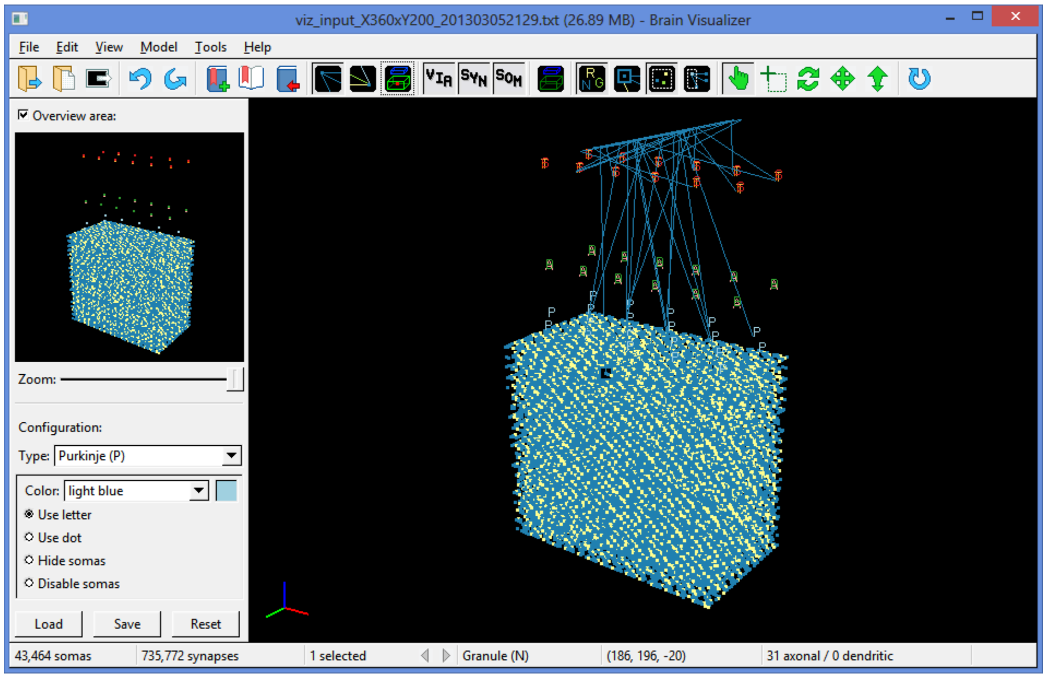

viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer
File
Edit
View
Model
Tools
Help
Overview area:
Zoom:
Load
Save
Reset

Configuration:
Type: Purkinje (P)
Color: light blue
[✓ Use letter] Use dot Hide somas Disable somas

43,464 somas | 735,772 synapses | 1 selected | Purkinje (P) | (38, 51, 0) | 1 axonal / 13,821 dendritic

Figure 33: With the Connected Fields option on, the neuritic fields of the somas connected to the selected Purkinje cell are drawn. The blue axonal fields are parallel fibers, and the green dendritic fields belong to granule cells making synapses (the red dots) within the Purkinje cell's large dendritic tree.

A.3. Selection


viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer
File
Edit
View
Model
Tools
Help
Overview area:
Zoom:
Load
Save
Reset

Configuration:
Type: Purkinje (P)
Color: light blue
[✓ Use letter] Use dot Hide somas Disable somas

43,464 somas | 735,772 synapses | 1 selected | Granule (N) | (186, 196, -20) | 31 axonal / 0 dendritic

Figure 34: One selected granule cell.

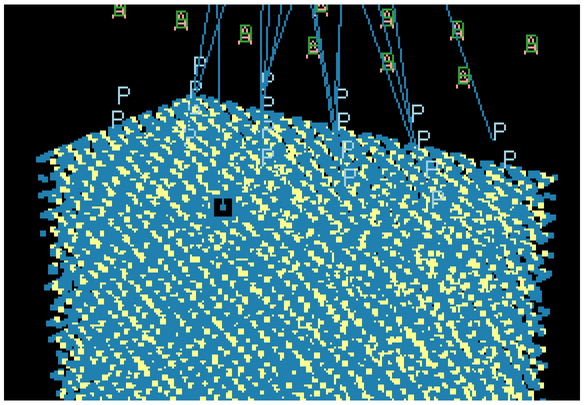
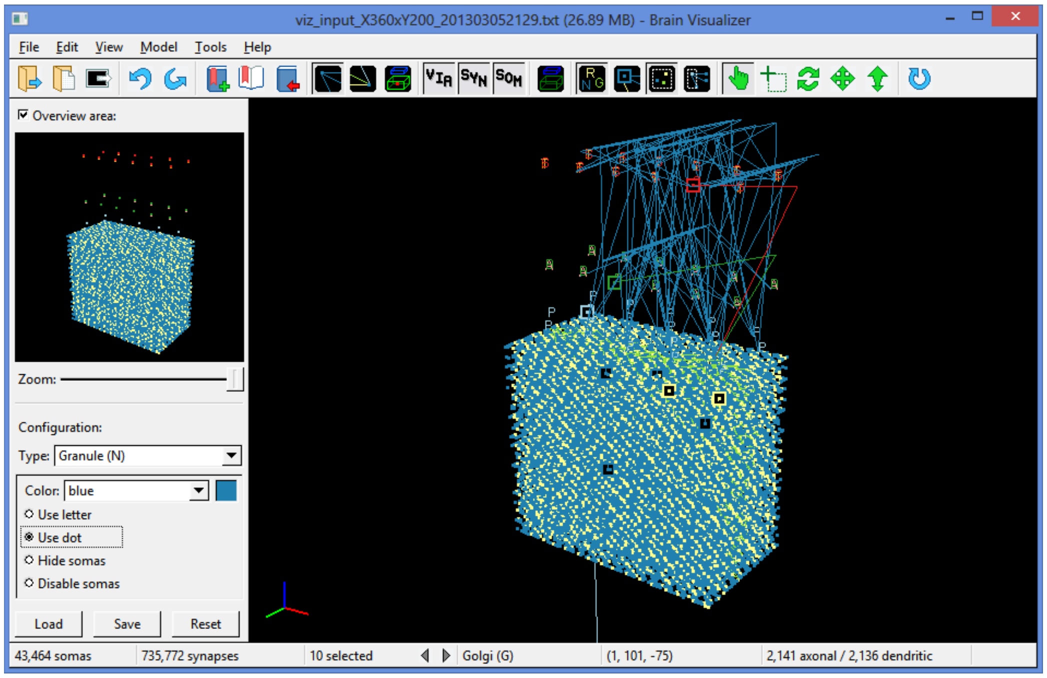


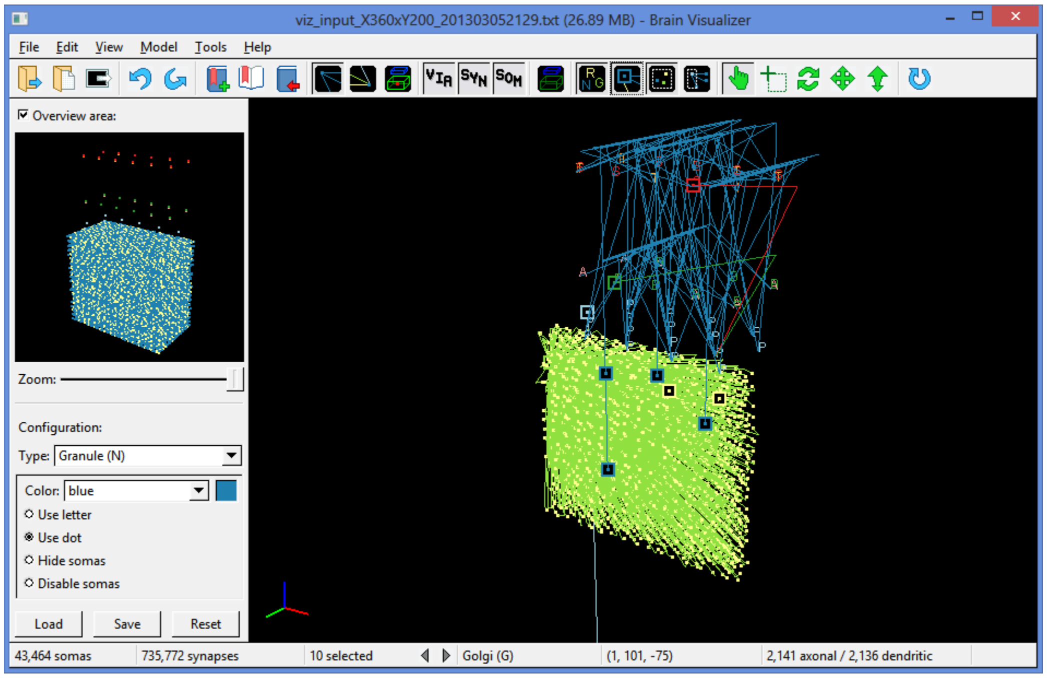
Figure 35: A close-up of the selected granule cell in the previous figure.


viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer
File
Edit
View
Model
Tools
Help
Overview area:
Zoom:
Load
Save
Reset

Configuration:
Type: Granule (N)
Color: blue
○ Use letter
◉ Use dot
○ Hide somas
○ Disable somas

43,464 somas | 735,772 synapses | 10 selected | Golgi (G) | (1, 101, -75) | 2,141 axonal / 2,136 dendritic

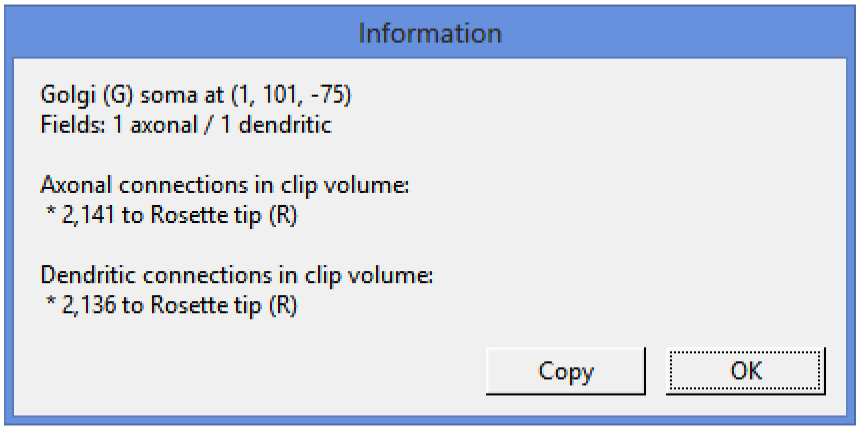
Figure 36: Ten selected somas: four granule cells, two mossy fibers' rosette tips, a Purkinje cell, a basket cell, an outer stellate cell, and a Golgi cell.


viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer
File Edit View Model Tools Help
Overview area:
Zoom:
Load
Save
Reset

Configuration:
Type: Granule (N)
Color: blue
Use letter
[✓ Use dot]
Hide somas
Disable somas

43,464 somas | 735,772 synapses | 10 selected | Golgi (G) | (1, 101, -75) | 2,141 axonal / 2,136 dendritic

Figure 37: The same ten selected somas as above, including their axonal connections, with the Only Show Selected option on.



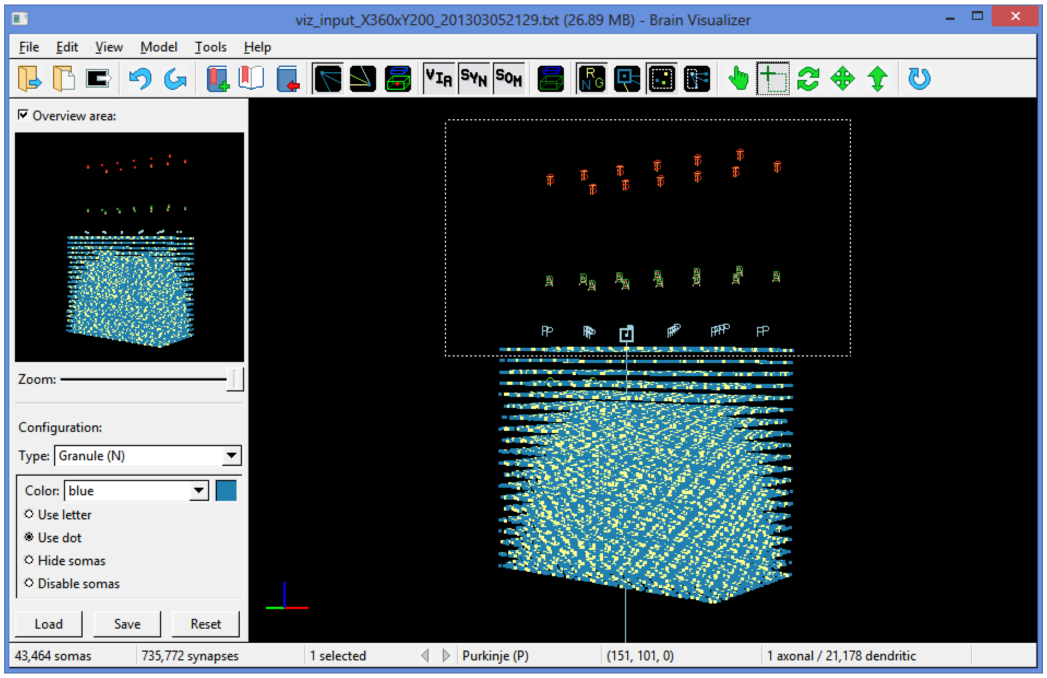
Golgi (G) soma at (1, 101, -75)
Fields: 1 axonal / 1 dendritic

Axonal connections in clip volume:
* 2,141 to Rosette tip (R)

Dendritic connections in clip volume:
* 2,136 to Rosette tip (R)

Figure 38: The information dialog brought up by Ctrl+clicking or right-clicking on a Golgi cell.

A.4. Clipping


viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer

[✓] Overview area:

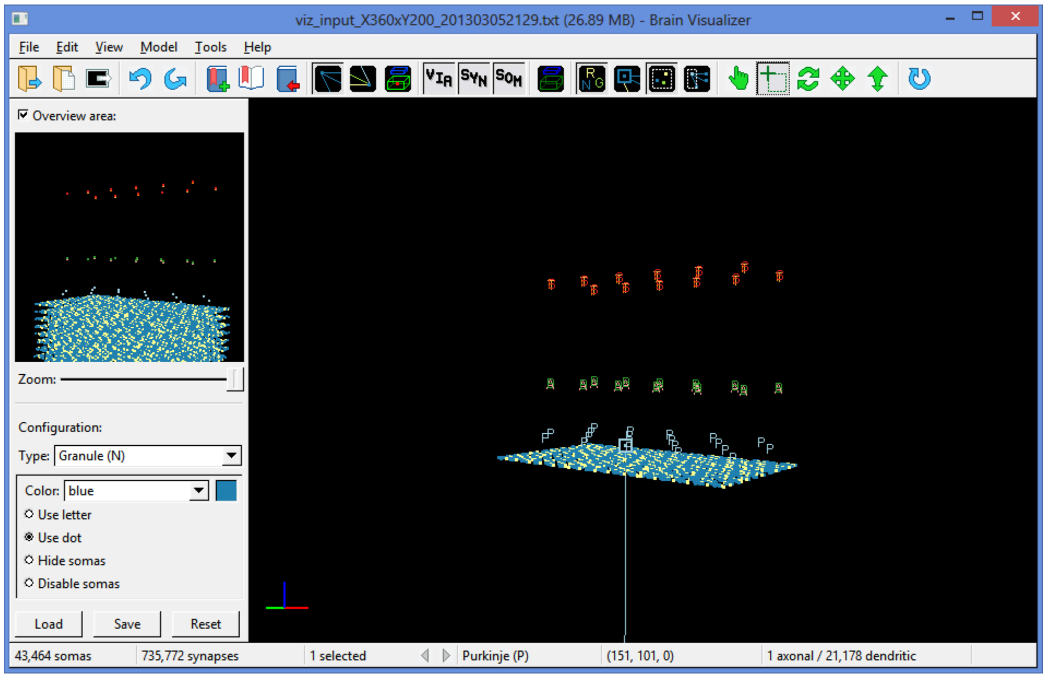
Type: Granule (N)

Color: blue

Use letter [✓ Use dot] Hide somas Disable somas

```
43,464 somas | 735,772 synapses | 1 selected | Purkinje (P) | (151, 101, 0) | 1 axonal / 21,178 dendritic
```

Figure 39: A clipping rectangle being drawn with the Clip tool. A single Purkinje cell is selected and the top of its long descending axon is shown.


viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer

[✓] Overview area:

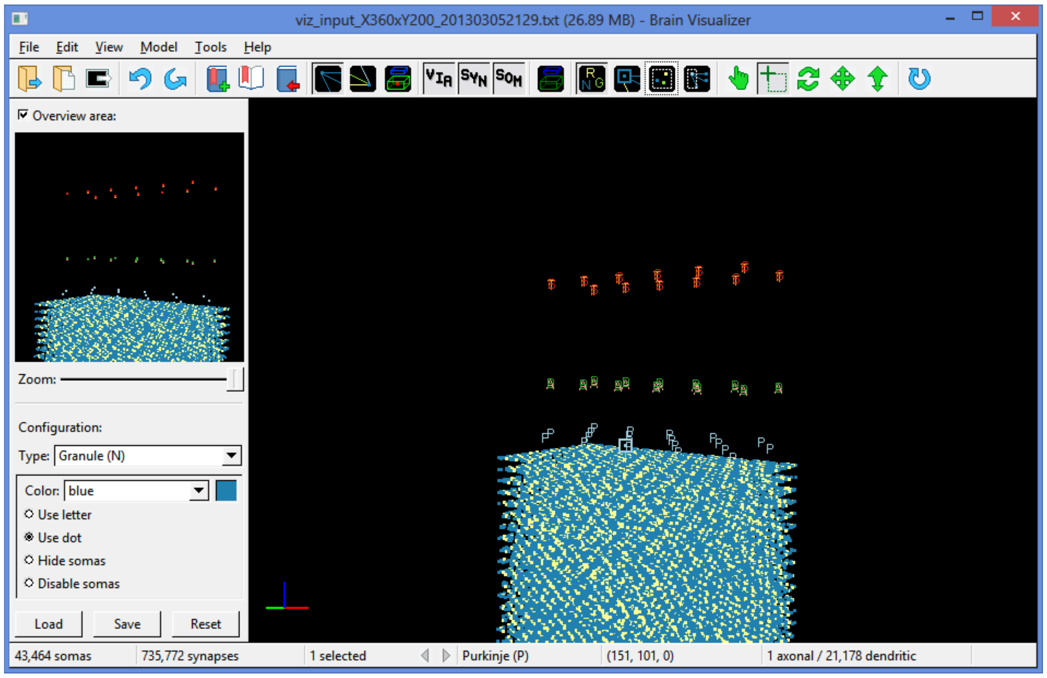
Type: Granule (N)

Color: blue

Use letter [✓ Use dot] Hide somas Disable somas

```
43,464 somas | 735,772 synapses | 1 selected | Purkinje (P) | (151, 101, 0) | 1 axonal / 21,178 dendritic
```

Figure 40: The result of the Clip action.


viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer

☑ Overview area:

Zoom:

Configuration:

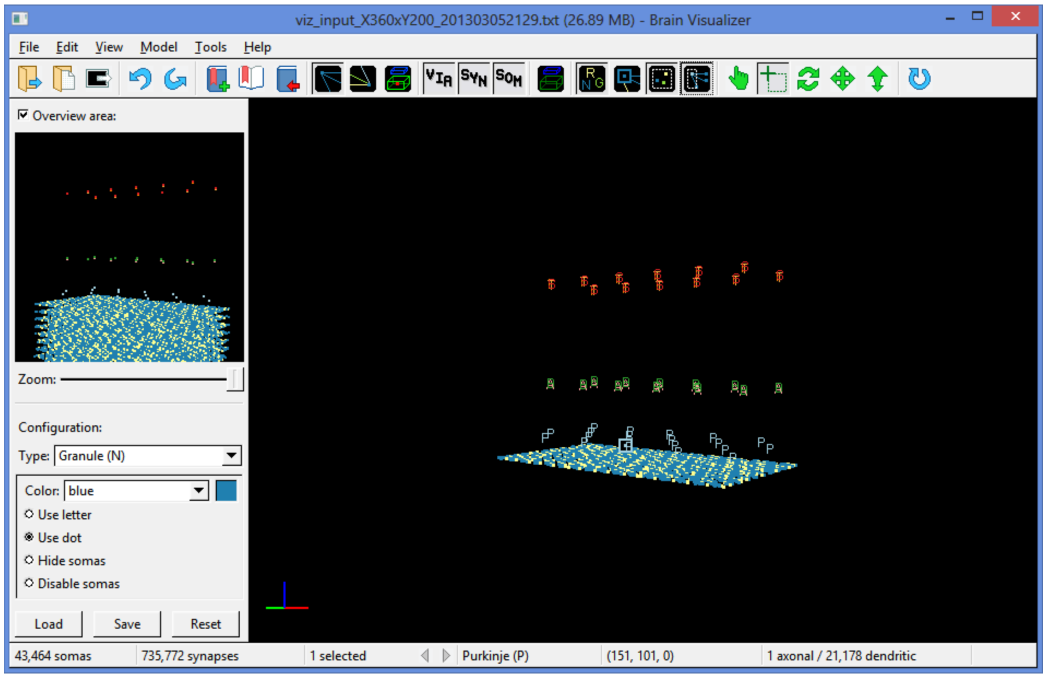
Type: Granule (N)

Color: blue 🟦

Use letter
[✓ Use dot]
Hide somas
Disable somas

43,464 somas | 735,772 synapses | 1 selected | Purkinje (P) | (151, 101, 0) | 1 axonal / 21,178 dendritic

Figure 41: A clipped region with the Hide Clipped option off shows somas outside the clipped region.


viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer

☑ Overview area:

Zoom:

Configuration:

Type: Granule (N)

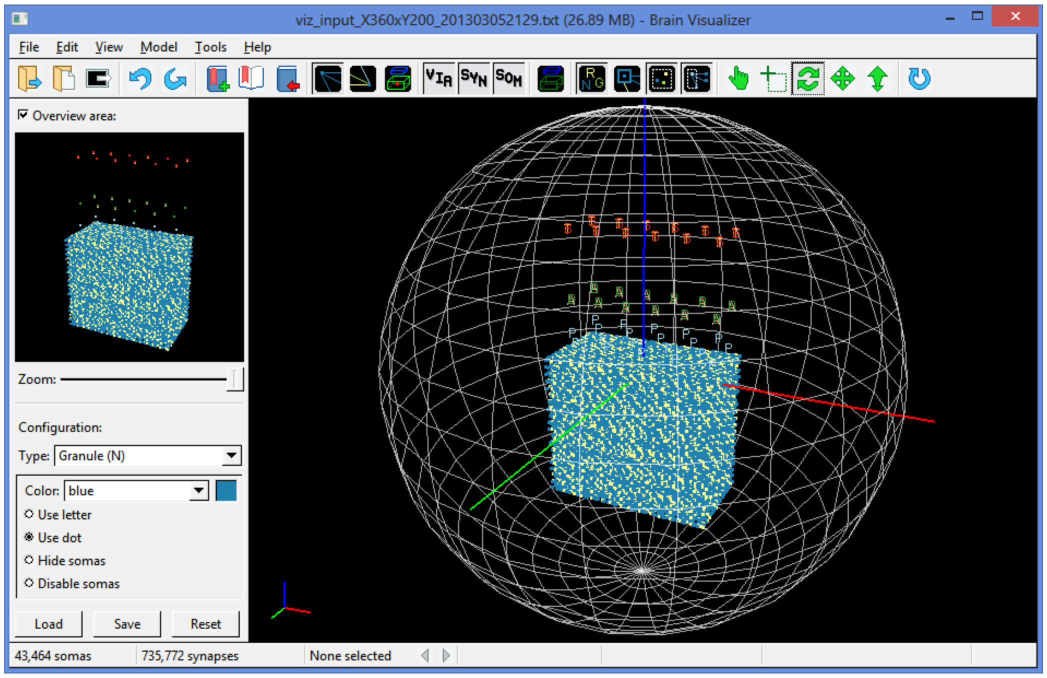
Color: blue 🟦

Use letter
[✓ Use dot]
Hide somas
Disable somas

43,464 somas | 735,772 synapses | 1 selected | Purkinje (P) | (151, 101, 0) | 1 axonal / 21,178 dendritic

Figure 42: A clipped region with the Only Enable Clipped option on; the selected Purkinje cell's connection to a dentate nucleus below the cortex and outside the clipped region is no longer shown.

A.5. Rotation


viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer
File
Edit
View
Model
Tools
Help
Load
Save
Reset

☑ Overview area:

Zoom:

Configuration:

Type: Granule (N)

Color: blue

○ Use letter

◉ Use dot

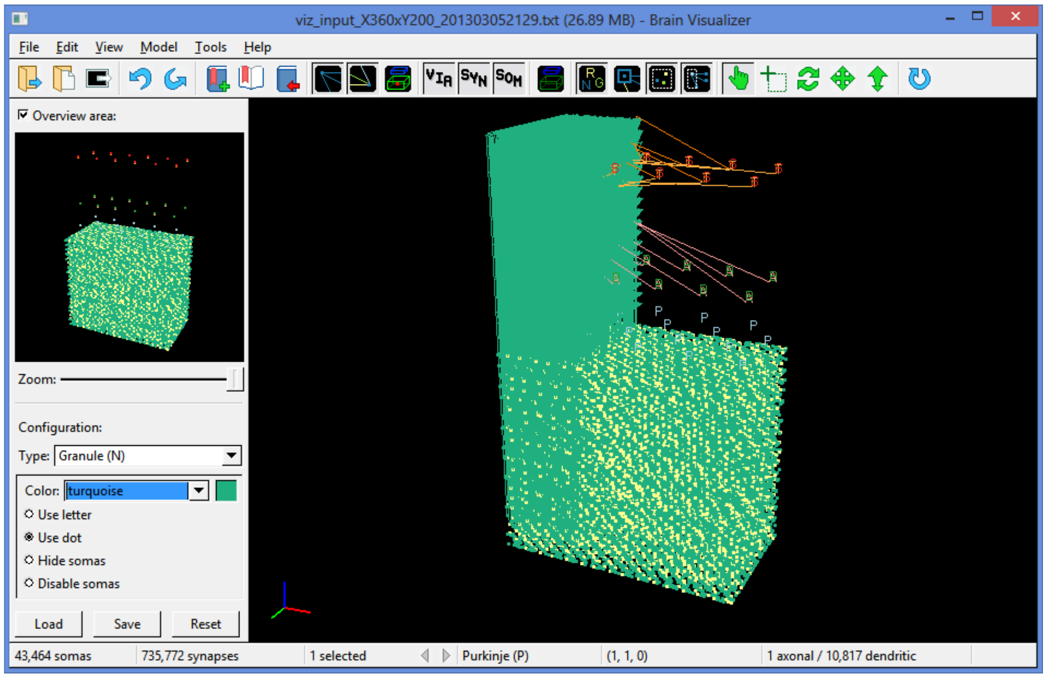
○ Hide somas

○ Disable somas

43,464 somas | 735,772 synapses | None selected

Figure 43: Using the rotation guide to rotate a model.

A.6. Configuration


viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer
File
Edit
View
Model
Tools
Help
Load
Save
Reset

☑ Overview area:

Zoom:

Configuration:

Type: Granule (N)

Color: turquoise

○ Use letter

◉ Use dot

○ Hide somas

○ Disable somas

43,464 somas | 735,772 synapses | 1 selected | Purkinje (P) | (1, 1, 0) | 1 axonal / 10,817 dendritic

Figure 44: A model with the granule type set to a different color. Granule cells and their axonal connections to a single selected Purkinje cell are shown in turquoise instead of blue.

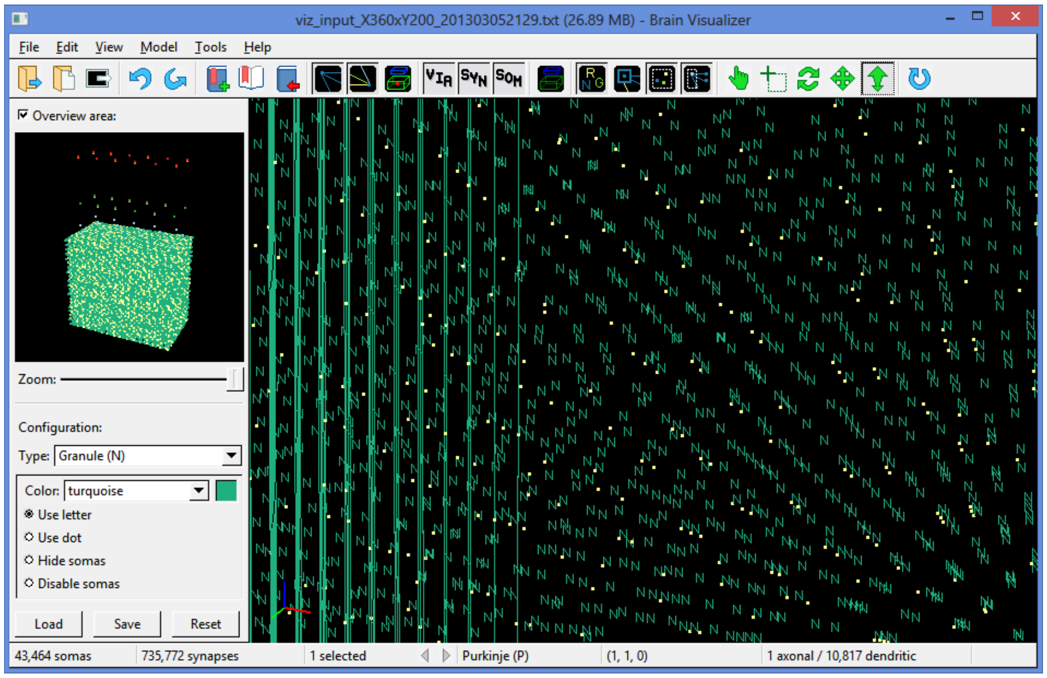


Figure 45: A zoomed-in close-up of a model with the granule cells displayed as letters. Some of the granule cells' axonal connections to a Purkinje cell are in the visible area.

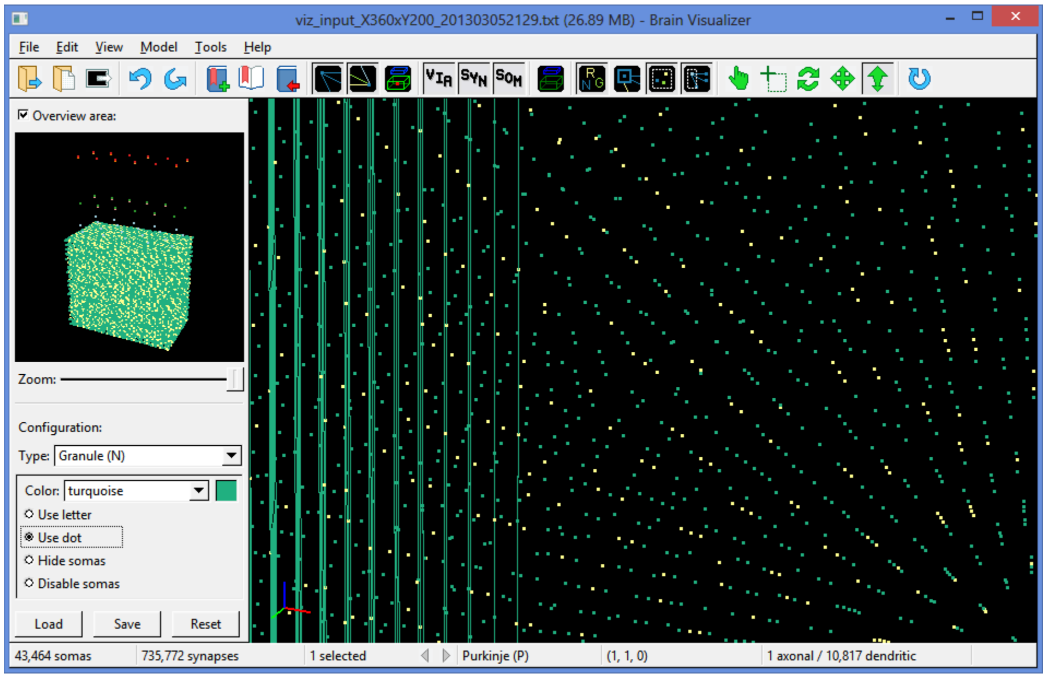
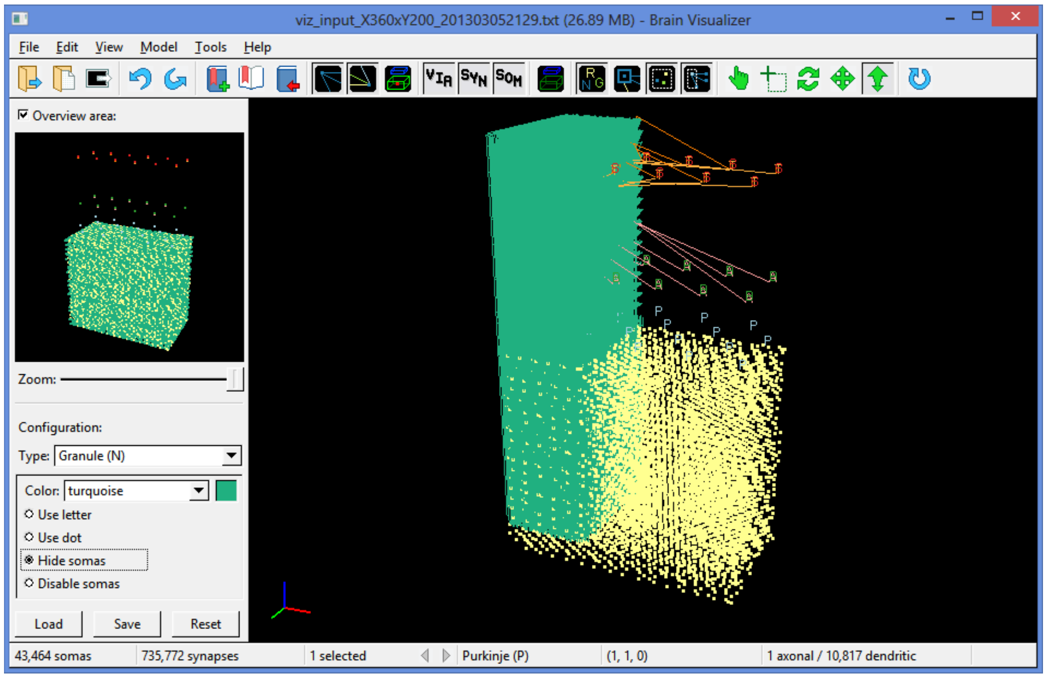


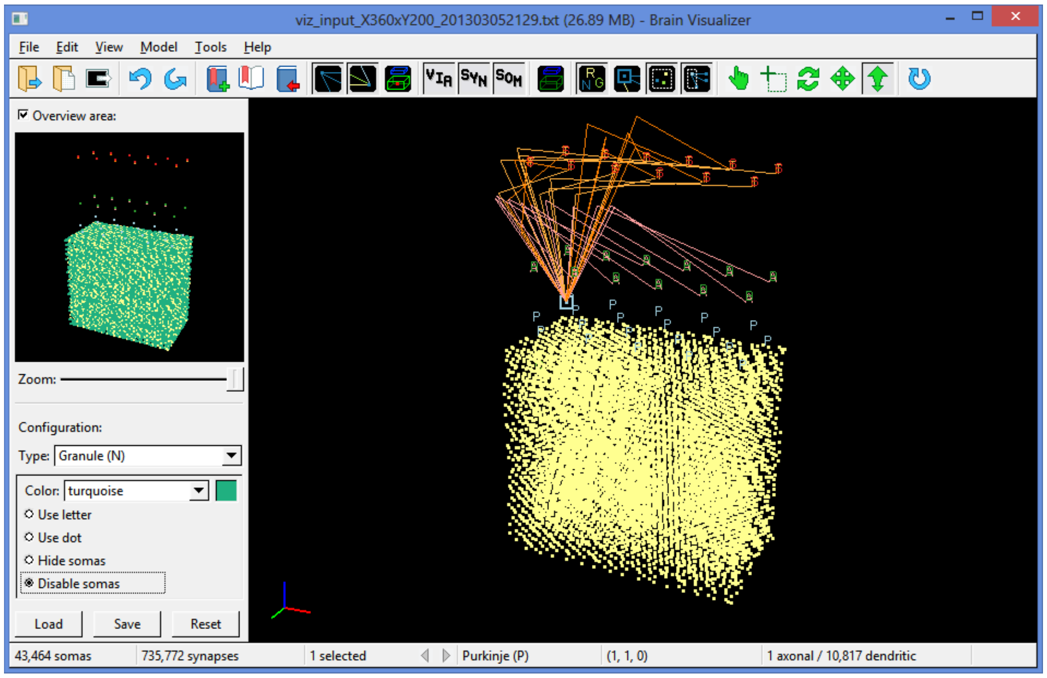
Figure 46: The same model as the previous figure, but with the granule cells displayed as dots.


viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer
File
Edit
View
Model
Tools
Help
Overview area:
Zoom:
Load
Save
Reset

Configuration:
Type: Granule (N)
Color: turquoise
Use letter
Use dot
[✓ Hide somas]
Disable somas

43,464 somas | 735,772 synapses | 1 selected | Purkinje (P) | (1, 1, 0) | 1 axonal / 10,817 dendritic

Figure 47: The same model with the granule type hidden. Only the granule cells connected to a selected Purkinje cell, and their axonal connections to that Purkinje cell, are still shown. (The yellow dots are mossy fiber rosette tips.)

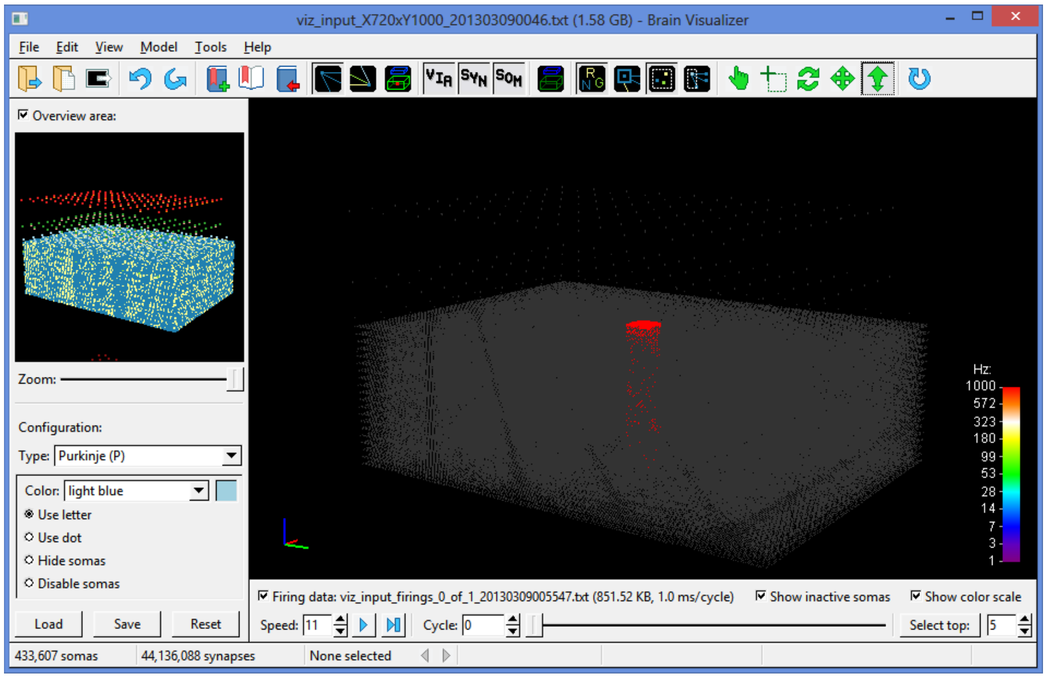

viz_input_X360xY200_201303052129.txt (26.89 MB) - Brain Visualizer
File
Edit
View
Model
Tools
Help
Overview area:
Zoom:
Load
Save
Reset

Configuration:
Type: Granule (N)
Color: turquoise
Use letter
Use dot
Hide somas
[✓ Disable somas]

43,464 somas | 735,772 synapses | 1 selected | Purkinje (P) | (1, 1, 0) | 1 axonal / 10,817 dendritic

Figure 48: The same model with the granule type disabled. Granule somas and axonal connections are no longer shown.

A.7. Firing Data


viz_input_X720xY1000_201303090046.txt (1.58 GB) - Brain Visualizer

☑ Overview area:

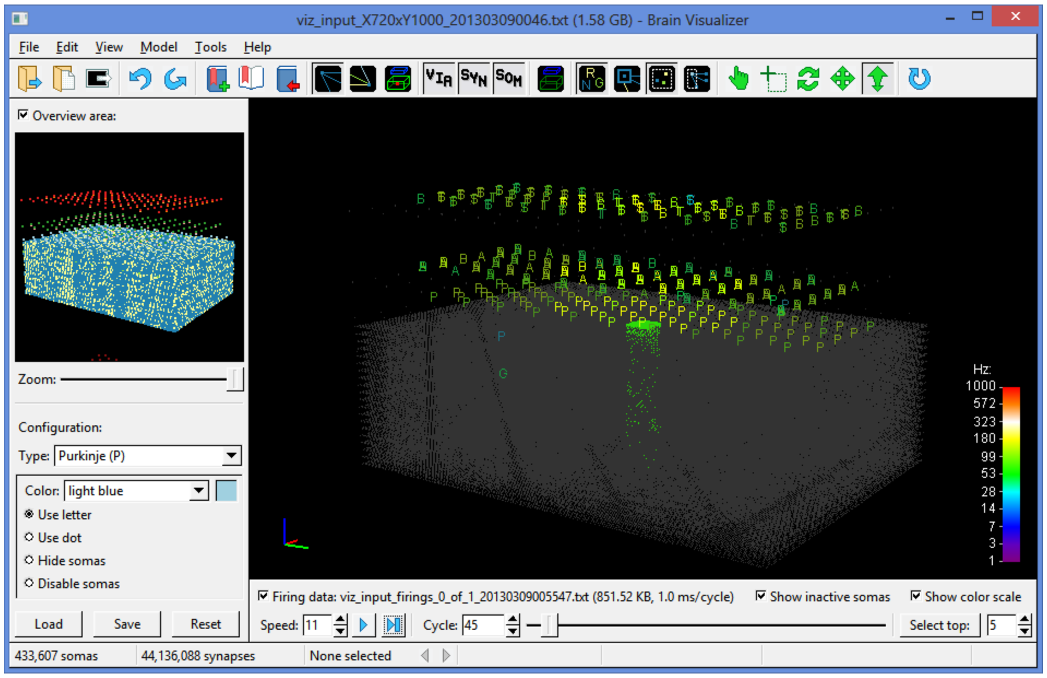
Configuration:
Type: Purkinje (P)
Color: light blue
(●) Use letter
○ Use dot
○ Hide somas
○ Disable somas

☑ Firing data: viz_input_firings_0_of_1_20130309005547.txt (851.52 KB, 1.0 ms/cycle) ☑ Show inactive somas ☑ Show color scale

Speed: 11 Cycle: 0 Select top: 5

433,607 somas 44,136,088 synapses None selected

Figure 49: A larger cerebellar cortex model (containing 433 thousand somas and 44 million synapses) with firing data loaded. All the granule cells within a small central column are stimulated to fire once every 20 cycles (which is every 10 milliseconds).


viz_input_X720xY1000_201303090046.txt (1.58 GB) - Brain Visualizer

☑ Overview area:

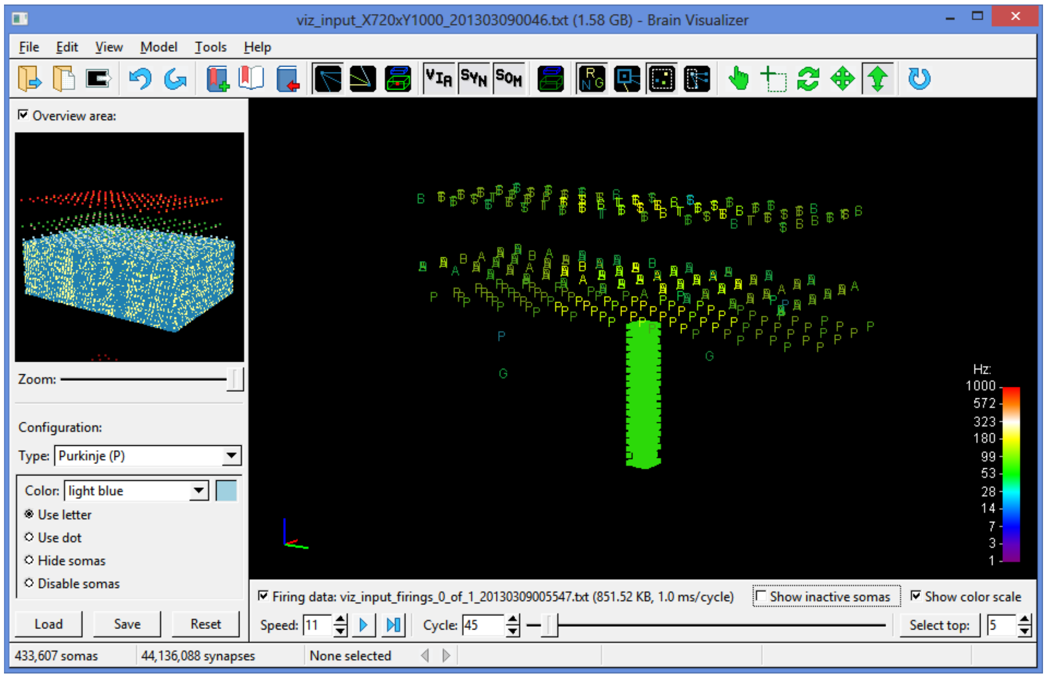
Configuration:
Type: Purkinje (P)
Color: light blue
(●) Use letter
○ Use dot
○ Hide somas
○ Disable somas

☑ Firing data: viz_input_firings_0_of_1_20130309005547.txt (851.52 KB, 1.0 ms/cycle) ☑ Show inactive somas ☑ Show color scale

Speed: 11 Cycle: 45 Select top: 5

433,607 somas 44,136,088 synapses None selected

Figure 50: The firing data of the model from the previous figure after simulating 45 cycles (22.5 ms) of electrical interaction.



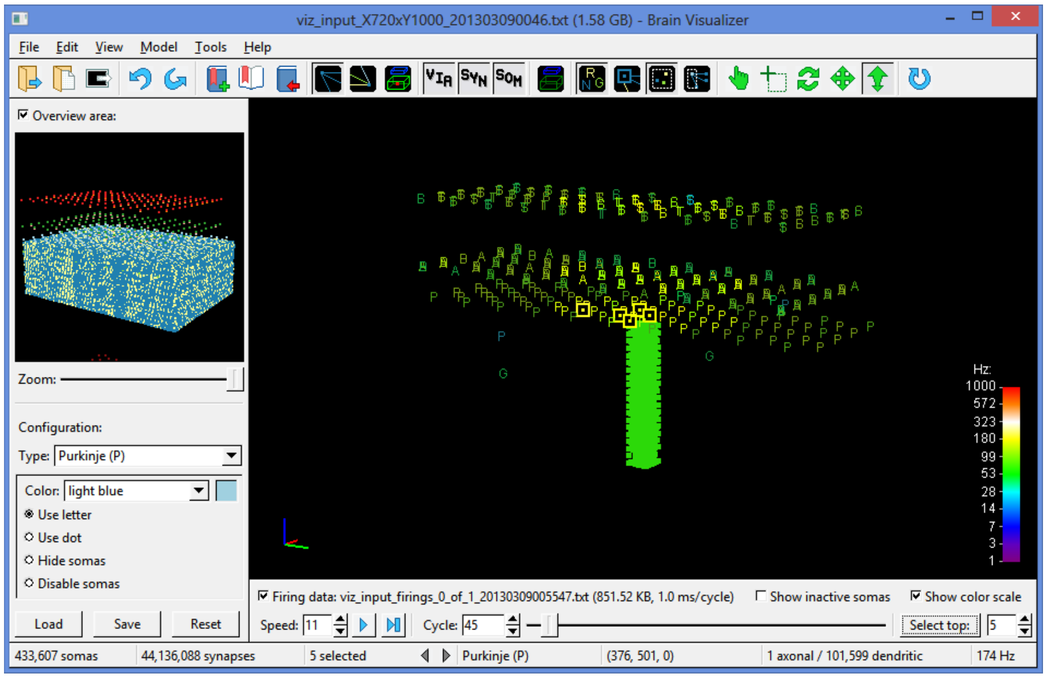
viz_input_X720xY1000_201303090046.txt (1.58 GB) - Brain Visualizer

☑ Overview area:
Zoom:
Configuration:
Type: Purkinje (P)
Color: light blue
● Use letter
○ Use dot
○ Hide somas
○ Disable somas

☑ Firing data: viz_input_firings_0_of_1_20130309005547.txt (851.52 KB, 1.0 ms/cycle)
☐ Show inactive somas
☑ Show color scale
Speed: 11
Cycle: 45
Select top: 5

433,607 somas | 44,136,088 synapses | None selected

Figure 51: The same firing data as the previous figure, with inactive somas hidden.



viz_input_X720xY1000_201303090046.txt (1.58 GB) - Brain Visualizer

☑ Overview area:
Zoom:
Configuration:
Type: Purkinje (P)
Color: light blue
● Use letter
○ Use dot
○ Hide somas
○ Disable somas

☑ Firing data: viz_input_firings_0_of_1_20130309005547.txt (851.52 KB, 1.0 ms/cycle)
☐ Show inactive somas
☑ Show color scale
Speed: 11
Cycle: 45
Select top: 5

433,607 somas | 44,136,088 synapses | 5 selected | Purkinje (P) | (376, 501, 0) | 1 axonal / 101,599 dendritic | 174 Hz

Figure 52: The firing data with the top 5 most frequently firing somas (all Purkinje cells) selected.

B. Source Code Files

These files are the contents of the src directory of the source code for Viz. Unlisted files include IDE project files and Makefiles, documentation, configuration files, FLTK library and header files, and resource files such as icons.

- main.cpp: Defines the `main()` function, the entry point into the Viz program that initializes FLTK, creates a new window and opens a model file if passed one via command-line arguments.
- version.h: Defines `VIZ_VERSION` and `VIZ_VERSION_STRING`, two constants that specify the current version of Viz. They are automatically updated by the build script for deploying a production copy of Viz.
- utils.h: Defines some useful low-level macros, types, and constants.
- algebra.cpp and algebra.h: Define mathematical constants like `PI`, and functions for vector and matrix algebra.
- from-file.cpp and from-file.h: Define the abstract `From_File` class for containing data read from a file. The class only stores the file's name and size; it can be extended to store more particular data.
- input-parser.cpp and input-parser.h: Define the `Input_Parser` class for parsing text files and the `Gzip_Input_Parser` class for parsing Gzipped text files. The text is expected to contain space-separated numeric tokens, with "#" characters commenting out all text until the end of the line.
- binary-parser.cpp and binary-parser.h: Define the `Binary_Parser` class for parsing binary files and the `Gzip_Binary_Parser` class for parsing Gzipped binary files. The binary file format is meant to store brain models.
- image.cpp and image.h: Define the `Image` class for writing RGB image data to a file in PNG, BMP, TGA, or PPM format.
- soma-type.cpp and soma-type.h: Define the `Soma_Type` class for storing a soma type's letter, name, color, and display state.
- soma.cpp and soma.h: Define the `Soma` class for storing a soma's coordinates and associating it with a set of neuritic fields and synapses.
- neuritic-field.cpp and neuritic-field.h: Define the `Neuritic_Field` class for storing a neuritic field's bounding box.
- synapse.cpp and synapse.h: Define the `Synapse` class for storing a synapse's coordinates and via coordinates, and associating it with an axonal and dendritic soma.
- gap-junction.cpp and gap-junction.h: Define the `Gap_Junction` class for storing a gap junction's coordinates and associating it with a pair of somas.

- brain-model.cpp and brain-model.h: Define the `Brain_Model` class for storing a brain model's arrays of soma types, somas, neuritic fields, synapses, and gap junctions, as well as associated simulation data.
- sim-data.cpp and sim-data.h: Define the abstract `Sim_Data` class for storing data associated with a simulation run. The class only keeps track of a current cycle and associated brain model and color map; it can be extended to store more particular data.
- firing-spikes.cpp and firing-spikes.h: Define the `Firing_Spikes` class for storing firing spike data from a simulation run. Somas' current firing frequency can then be estimated from a window of their past spikes.
- voltages.cpp and voltages.h: Define the `Voltages` class for storing soma voltage data from a simulation run.
- weights.cpp and weights.h: Define the `Weights` class for storing synapse weight data from a simulation run.
- fps.cpp and fps.h: Define the `FPS` class for estimating frames per second from counting clock ticks over time.
- bounds.cpp and bounds.h: Define the `Bounds` class for storing a 3D coordinate boundary around a set of points, and calculating their center and range.
- clip-volume.cpp and clip-volume.h: Define the `Clip_Volume` class for storing four clipping planes and enabling or disabling them in OpenGL.
- color.cpp and color.h: Define the `Color` class for storing a named RGB color.
- color-maps.cpp and color-maps.h: Define the abstract `Color_Map` class for mapping numeric values to colors, and subclasses that specify particular gradients to map onto, including `Rainbow_Map`, `Thermal_Map`, and `Opposed_Map`.
- draw-options.cpp and draw-options.h: Define the `Draw_Options` class for storing the drawing state of a model area.
- model-state.cpp and model-state.h: Define the `Model_State` class for storing the state of a model being visualized, including its rotation, panning, zoom, clip volume, selected somas, and marked synapses.
- model-area.cpp and model-area.h: Define the `Model_Area` class for displaying a brain model and maintaining an undo/redo history of its state changes.
- overview-area.cpp and overview-area.h: Define the `Overview_Area` class for displaying an overview of a brain model.
- widgets.cpp and widgets.h: Define various classes for common GUI widgets, including buttons, sliders, and spinners.
- os-themes.cpp and os-themes.h: Define appearances for GUI widgets which imitate the native widgets in Windows 7, Windows 8, Mac OS X Lion, and Linux (assuming a GTK desktop environment).

- modal-dialog.cpp and modal-dialog.h: Define the `Modal_Dialog` class for presenting a modal dialog with a message, OK button, and optional icon.
- progress-dialog.cpp and progress-dialog.h: Define the `Progress_Dialog` class for presenting a progress dialog with a message, progress bar, and Cancel button. On Windows the progress state is reflected in the taskbar button.
- waiting-dialog.cpp and waiting-dialog.h: Define the `Waiting_Dialog` class for presenting an indeterminate progress dialog with a message, progress bar, and Cancel button. On Windows the progress state is reflected in the taskbar button.
- report-dialog.cpp and report-dialog.h: Define the `Report_Dialog` class for presenting a dialog with options for generating a text report file.
- summary-dialog.cpp and summary-dialog.h: Define the `Summary_Dialog` class for presenting a dialog summarizing a brain model, a single soma, or a single synapse.
- viz-icons.h: Defines many icons based on XPM files.
- viz-window.cpp and viz-window.h: Define the `Viz_Window` class for presenting a complete windowed instance of Viz, and callback functions for many user actions.
- help-window.cpp and help-window.h: Define the `Help_Window` class for presenting a window with help information loaded from an HTML file.
- voltage-graph.cpp and voltage-graph.h: Define the `Voltage_Graph` class for presenting a graph of soma voltage over time, the `Voltage_Graph_Tooltip` class for presenting a particular (time, voltage) pair as a tooltip, and the `Voltage_Graph_Scroll` class for scrolling a voltage graph when zooming into a small region.
- voltage-graph-window.cpp and voltage-graph-window.h: Define the `Voltage_Graph_Window` class for presenting a voltage graph in a modal window.