Starting NEURON "by itself"

with the GUI

- double click on nrngui icon
- double click on a hoc file (MSWin)
- drag and drop hoc file onto nrngui icon (MSWin and MacOS)

from the command line

- `nrngui # loads GUI and standard run libs`
- `nrniv # loads neither`

```
nrn... bah.hoc # executes bah.hoc
```

Windows 10 because of PATH issues--

Terminal:	nrngui	nrniv
Command Prompt	N	N
NEURON's bash	Y	Y
Anaconda Prompt	N	Y

Using NEURON as a Python module

```
python foo.py
```

where foo.py starts with

```
from neuron import h
```

gets NEURON's computational engine

```
from neuron import h, gui
```

gets N's engine + GUI and standard run libraries, shows NEURON Main Menu toolbar

```
from neuron import h
h.load_file("stdgui.hoc")
```

gets GUI and std run libs, doesn't show toolbar

Hint: use -i to prevent autoexit

```
python -i foo.py
```

Exiting NEURON

Command:

hoc interpreter (oc> prompt) `quit()`

Python interpreter (>>> prompt) `exit()`

Keyboard shortcut: ^D (ctrl-D)

Exception: under MSWin if you started by

```
nrngui -python
```

or

```
nrniv -python
```

the shortcut is ^Z.

But you won't be using the -python option . . .

Why the GUI?

Improves productivity regardless of programming (in)experience by making it easier to

- develop, debug, and maintain models
- understand models developed by others
- visualize and understand simulation results
- use exploratory simulations to study model behavior
- optimize model parameters
- quickly create prototype models that can be mined for reusable code

Save time and avoid creating bugs--write less code!

Result: do more with less effort.

Using the GUI with Python

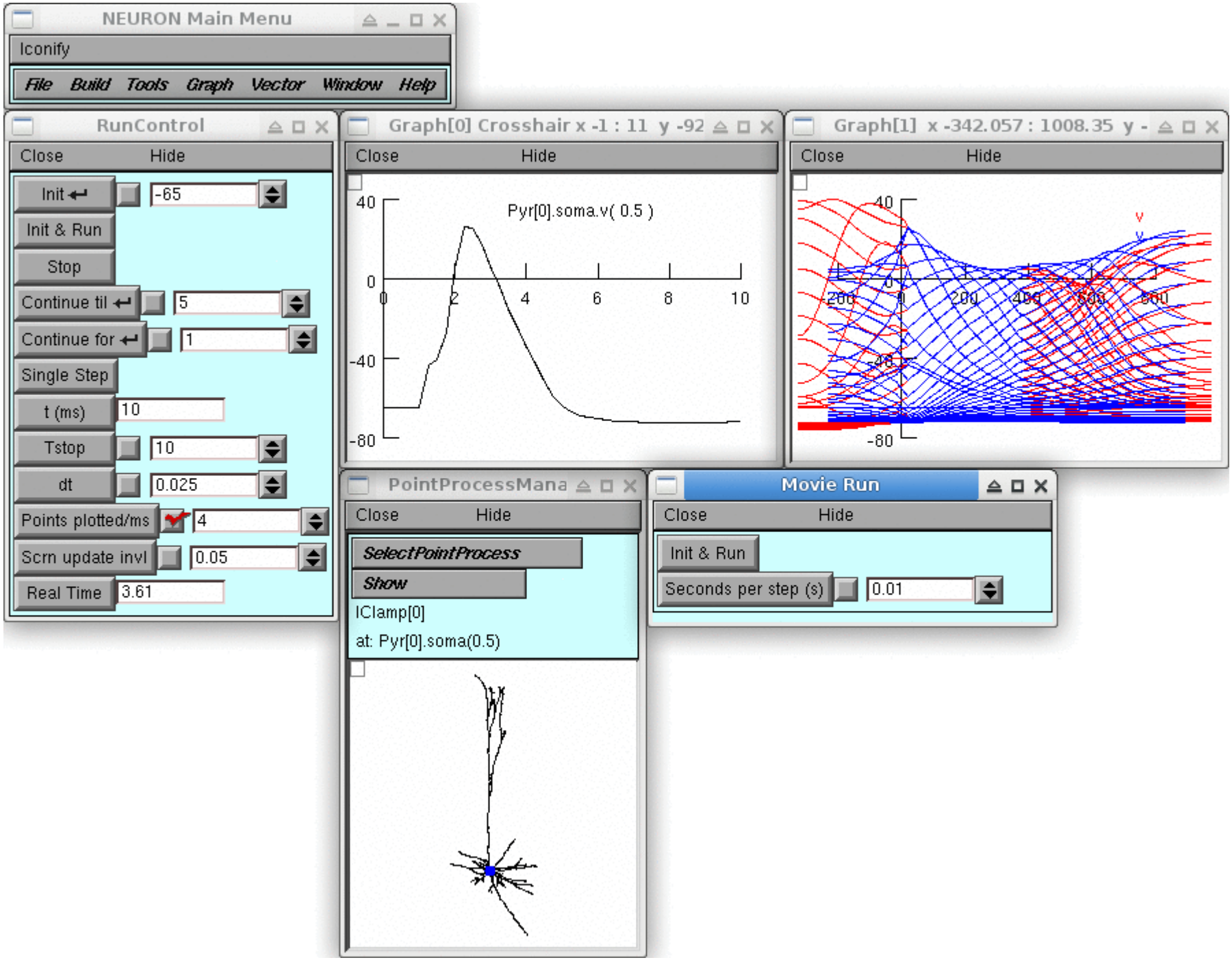
"I don't need to hear this. I use Python, and the GUI doesn't work with Python."--Anonymous dodo

Not so. The GUI works well with Python. Cells can be specified in hoc or Python (see "Scripting NEURON" by R McDougal)

Example: pyrtest.py

```
from neuron import h,gui
h.load_file('Pyr.hoc') # hoc template for Pyr class
        # built with and exported from CellBuilder
pyr = h.Pyr() # create instance of Pyr class
h.load_file('pyrtestrig.ses') # user interface
        # built with NEURON's GUI tools

python -i pyrtest.py
```


NEURON Main Menu
Iconify
File Build Tools Graph Vector Window Help
RunControl
Close Hide
Init
-65
Init & Run
Stop
Continue til
5
Continue for
1
Single Step
t (ms)
10
Tstop
10
dt
0.025
Points plotted/ms
4
Scrn update invl
0.05
Real Time
3.61
Graph[0] Crosshair x -1 : 11 y -92
Close Hide
Pyr[0].soma.v(0.5)
40
0
-40
-80
0
2
4
6
8
10
Graph[1] x -342.057 : 1008.35 y -
Close Hide
v
40
0
-40
-80
-200
0
200
400
600
800
PointProcessMana
Close Hide
SelectPointProcess
Show
IClamp[0]
at: Pyr[0].soma(0.5)
Movie Run
Close Hide
Init & Run
Seconds per step (s)
0.01

GUI tools

Many do things that would be very difficult, if not impossible, to accomplish with user-written code.

Import3d, Linear Circuit Builder, Multiple Run Fitter (optimizer), Impedance tools for analyzing electrical signaling in cells.

Some export code that can be reused with hoc and Python.

CellBuilder, Channel Builder, Linear Circuit Builder, Network Builder, Import3d, Model View (exports NeuroML)

Many can be saved directly to files for use by user-written hoc or Python script (example: pyrtest.py's custom interface)

Graphs, RunControl, any of the "Builders," Variable Step Control

See GUI tool tutorials on the Documentation page

https://neuron.yale.edu/neuron/docs

The most powerful approach: combine code and the GUI

The GUI

- always works
- can only do what it was designed to do

Coding is best for classical programming tasks, e.g.

- dealing with collections of things
- specifying custom initializations
- constructing complex simulation protocols
- filling gaps that aren't covered by the GUI

For maximum productivity, combine user-written code and the GUI to exploit the strengths of both.

Building and Using Models

Cell models

- Specify topology: create and connect sections
- Specify geometry: stylized (L & diam)
  or 3D (x,y,z,diam)
- Specify biophysics: insert density mechanisms,
  attach "biological" point processes (synapses)

Network models

- Define cell classes
- Create cells (instances of cell classes)
- Connect cells

Example: using the GUI to build and exercise a stylized model

1. Using the CellBuilder to create and manage a model cell.
2. Using the GUI to make an interface for running simulations.

Step 0: Conceptualize the task

Shape

stick figure *or* anatomically detailed?

Channel distribution

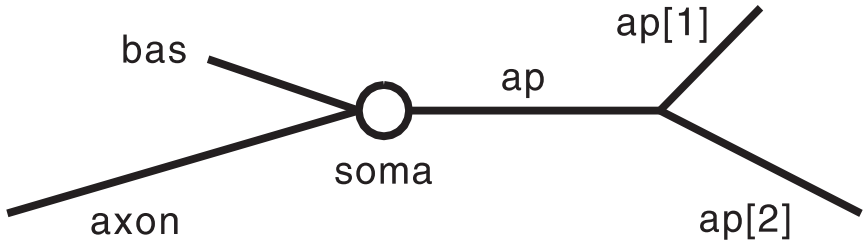
uniform *or* nonuniform?

individual neurite *or* whole cell *or* region?

Creation

single cell *or* use in a network?

Step 1: using the CellBuilder to make a stylized model



Section	L	diam	Biophysics
soma	20 µm	20 µm	hh
ap[0]	400	2	reduced hh *
ap[1]	300	1	reduced hh *
ap[2]	500	1	reduced hh *
bas	200	3	pas §
axon	800	1	hh

* gnabar_hh and gkbar_hh reduced to 10%, el_hh = - 64 mV

§ e_pas = - 65 mV

Throughout the cell Ra = 160 Ω cm, cm = 1 µf / cm^2

Launch NEURON with its GUI library

```
nrngui
```

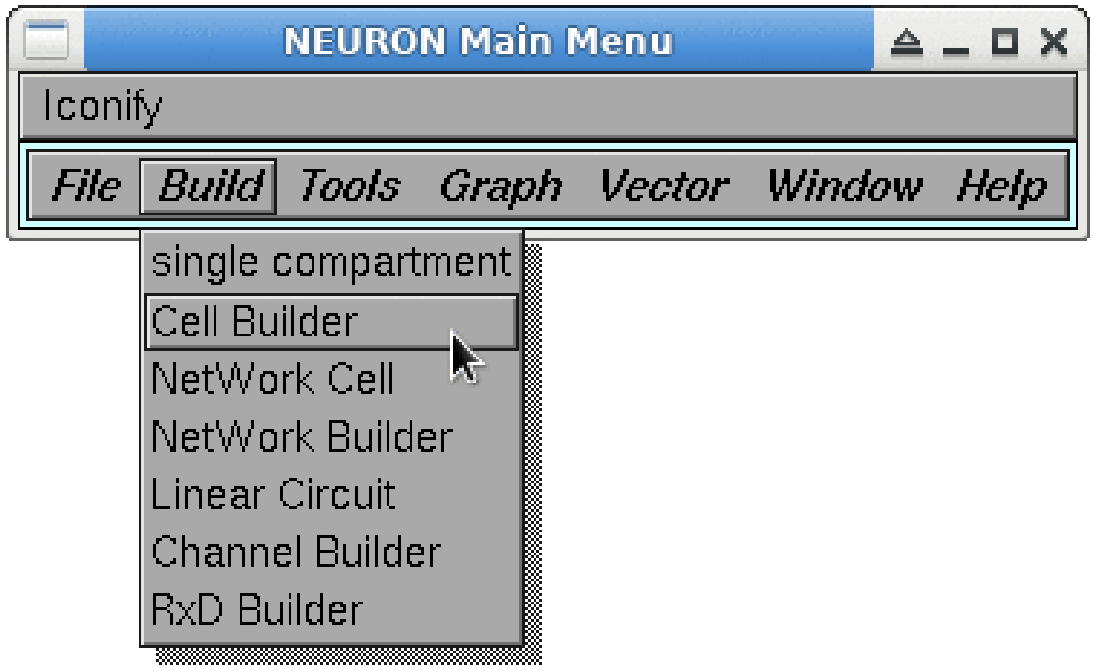
or

```
python
from neuron import h, gui
```

or

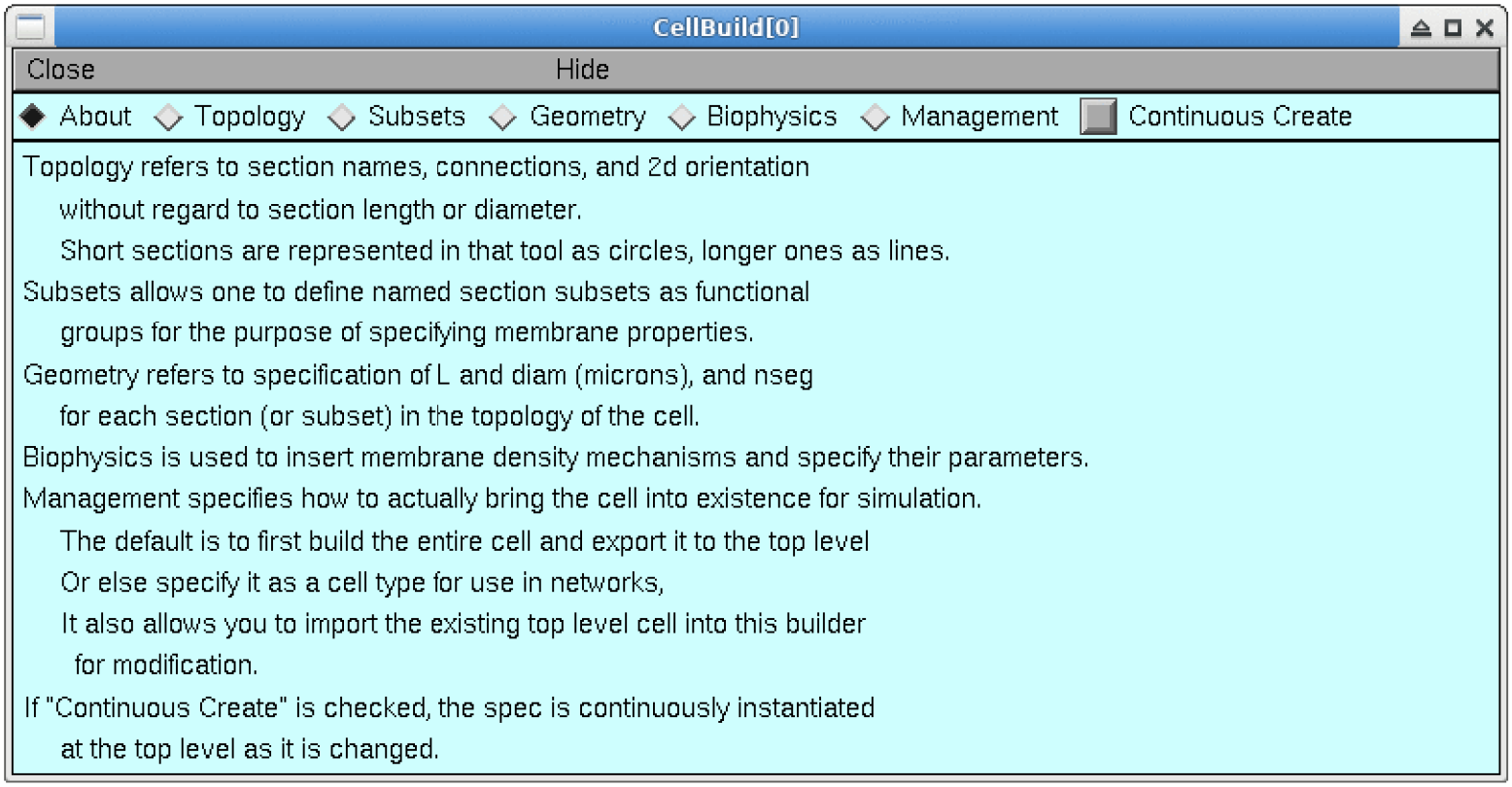
click on `nrngui` icon (MSWin, MacOS)

Bring up a CellBuilder



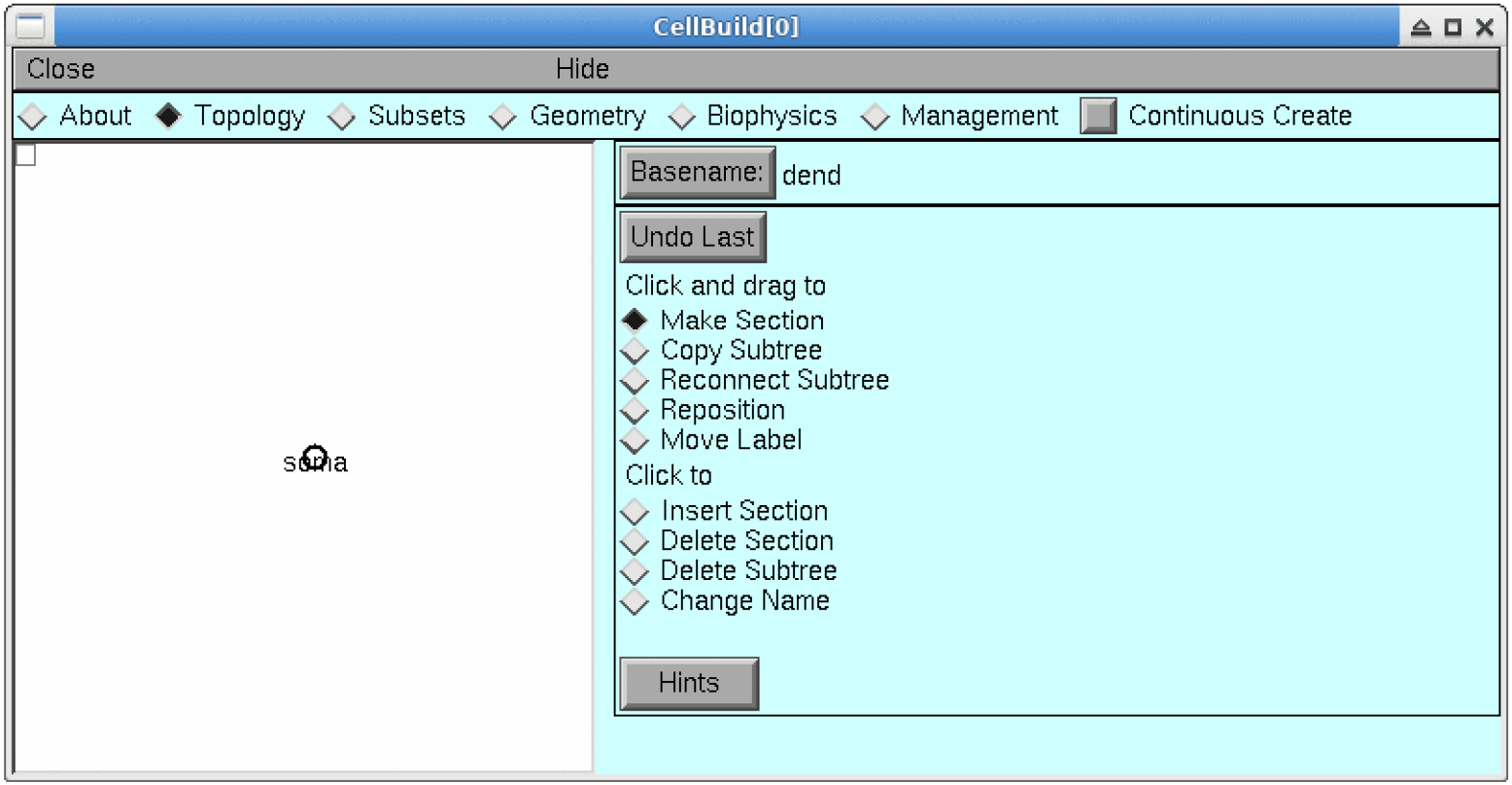
NEURON Main Menu / Build / Cell Builder

The CellBuilder



Use buttons from left to right.

Topology



CB starts with a "soma" section.

We want to create new sections.

Specifying the "Basename"

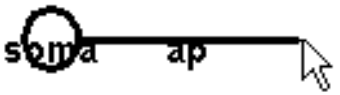
Making a new section

Place cursor near end of existing section

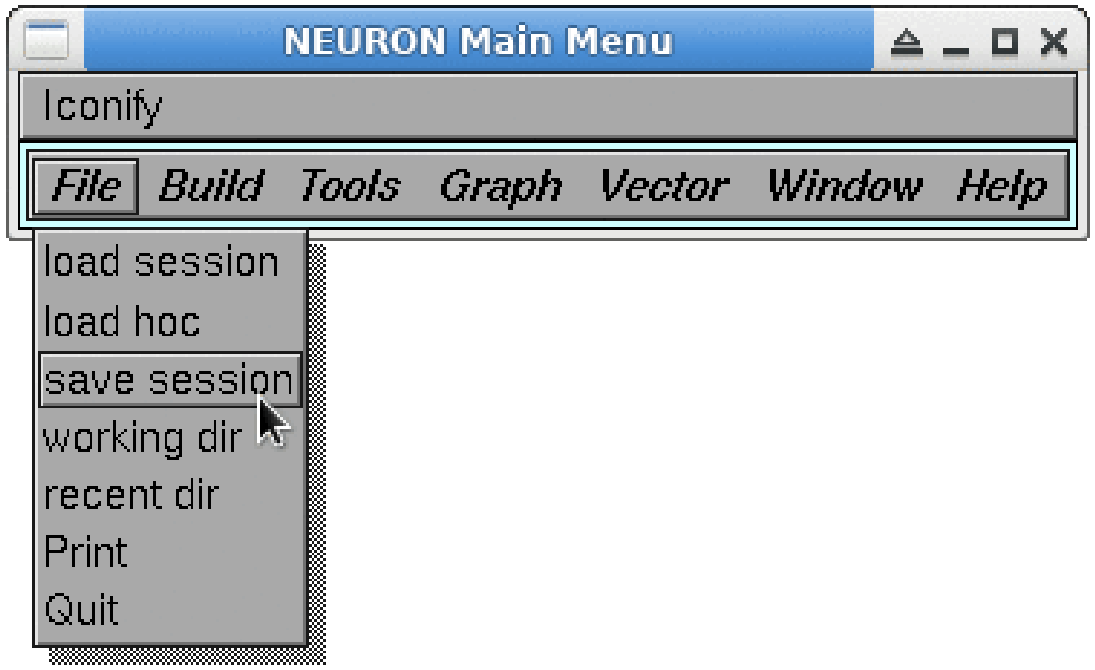
Click to start new section

Drag to desired length

Release mouse button

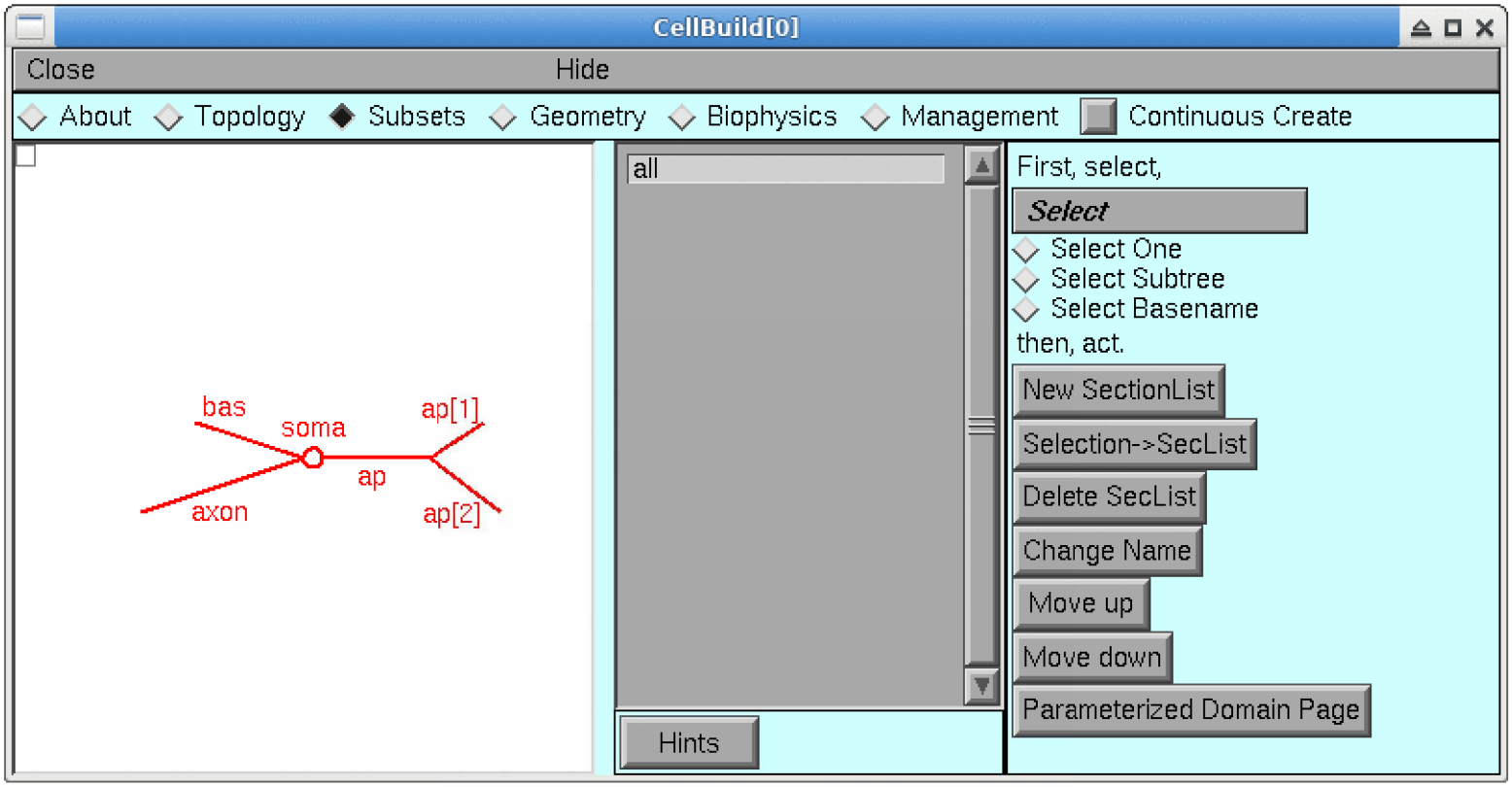


Save your work as you make progress!



NEURON Main Menu / File / save session

Subsets

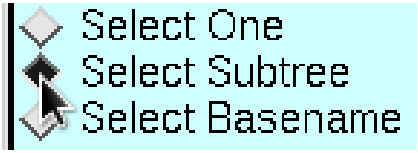


Group sections that have shared properties.

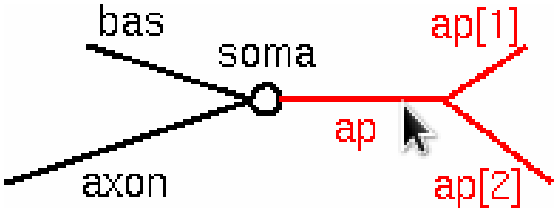
We want to make an "apicals" subset.

Making a new subset

Click "Select Subtree"



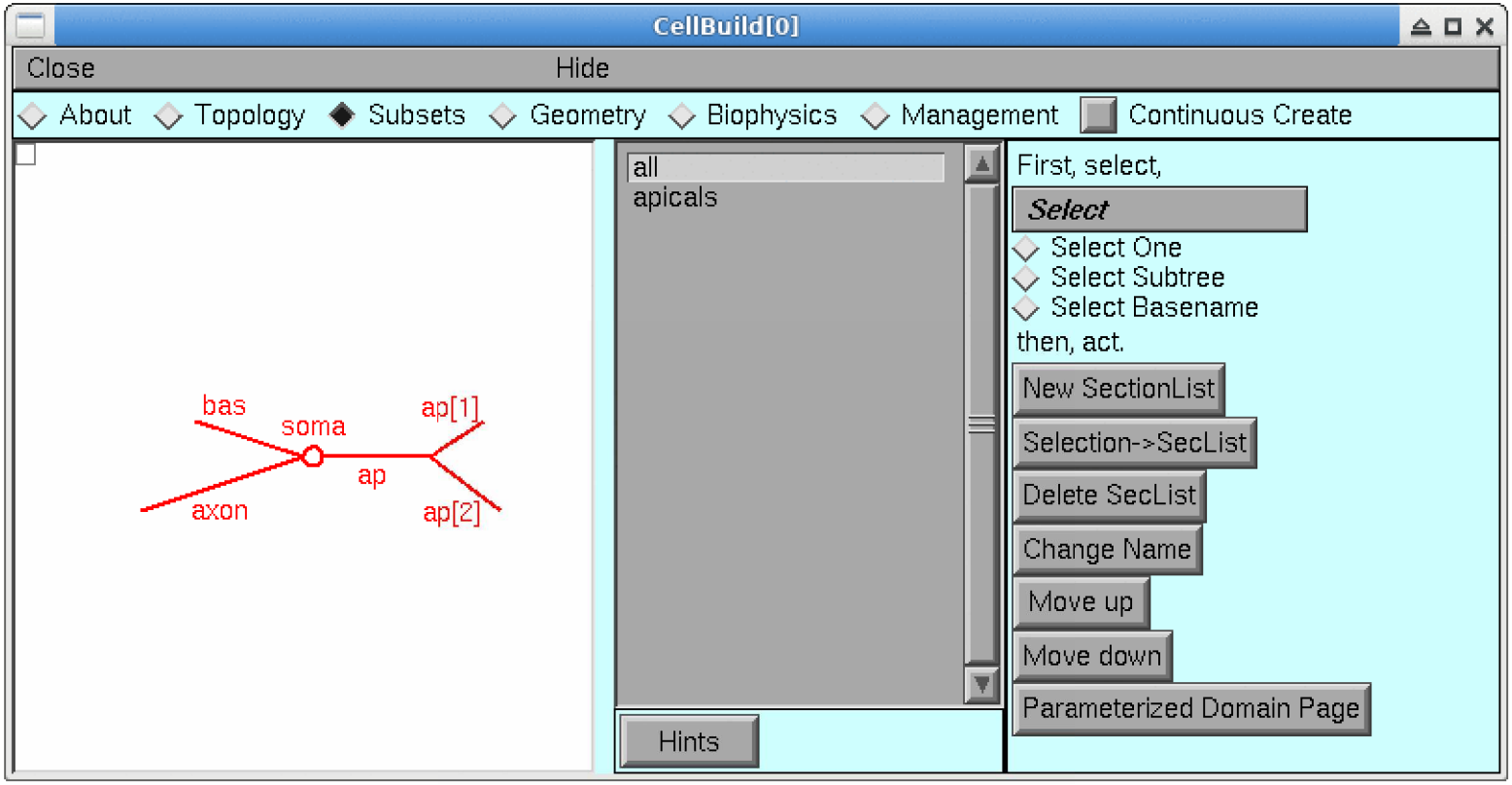
Click root of apical tree . . .



. . . then "New SectionList"

Making a new subset *continued*

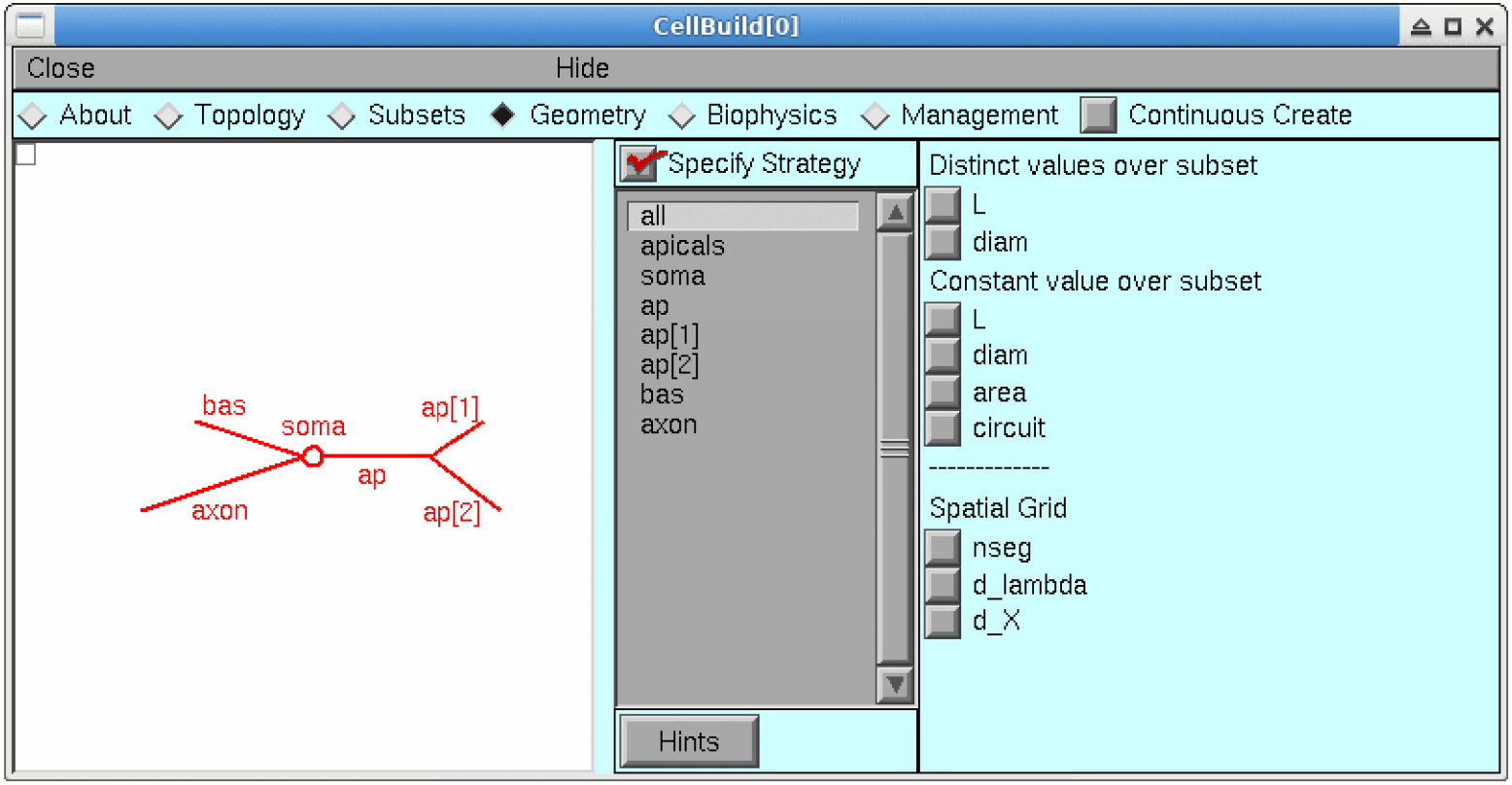
Subsets finished



Note "apicals" in middle panel.

Time to save a new session file.

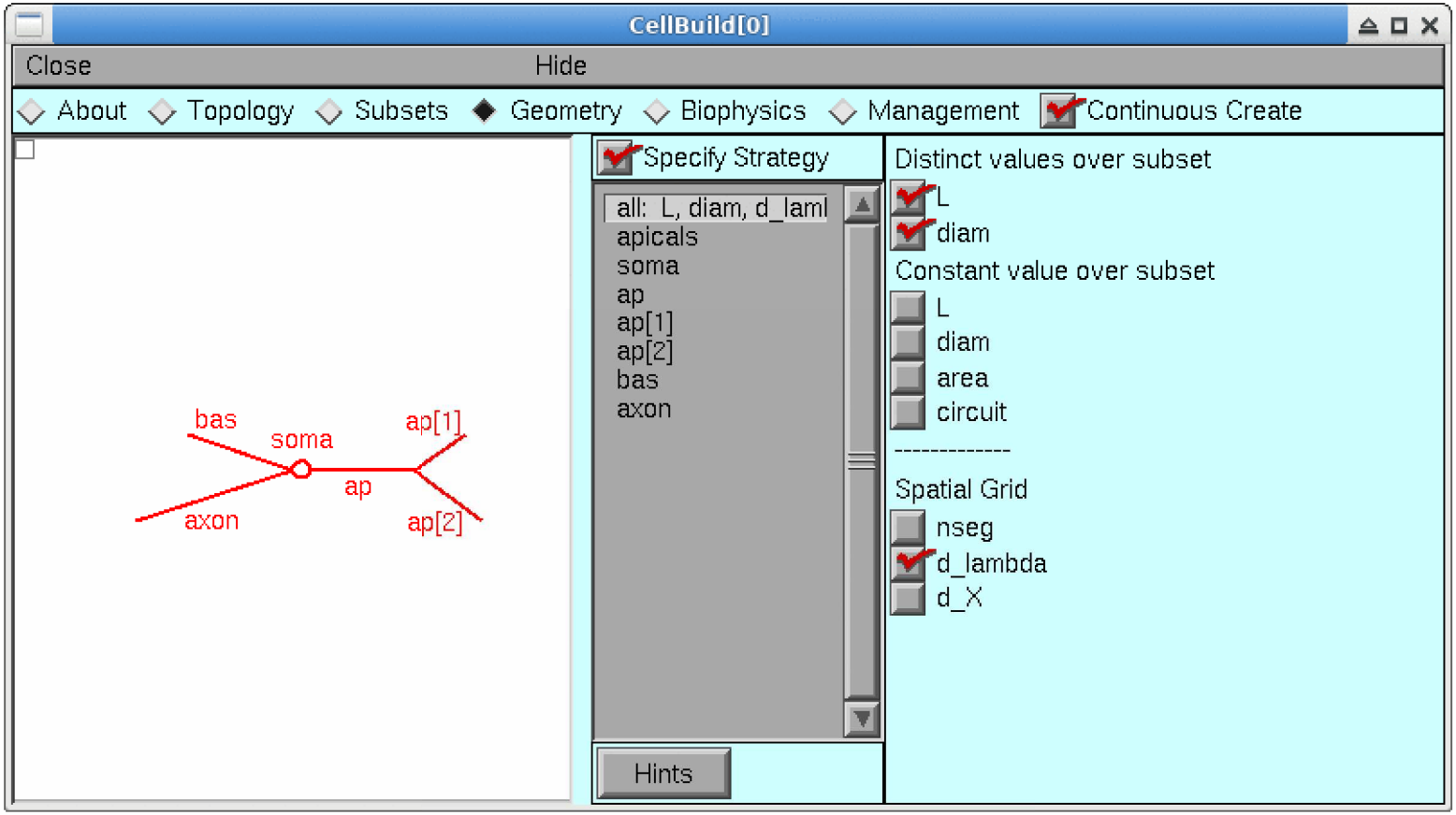
Geometry



"Specify Strategy" is ON.

A good strategy is a concise strategy.

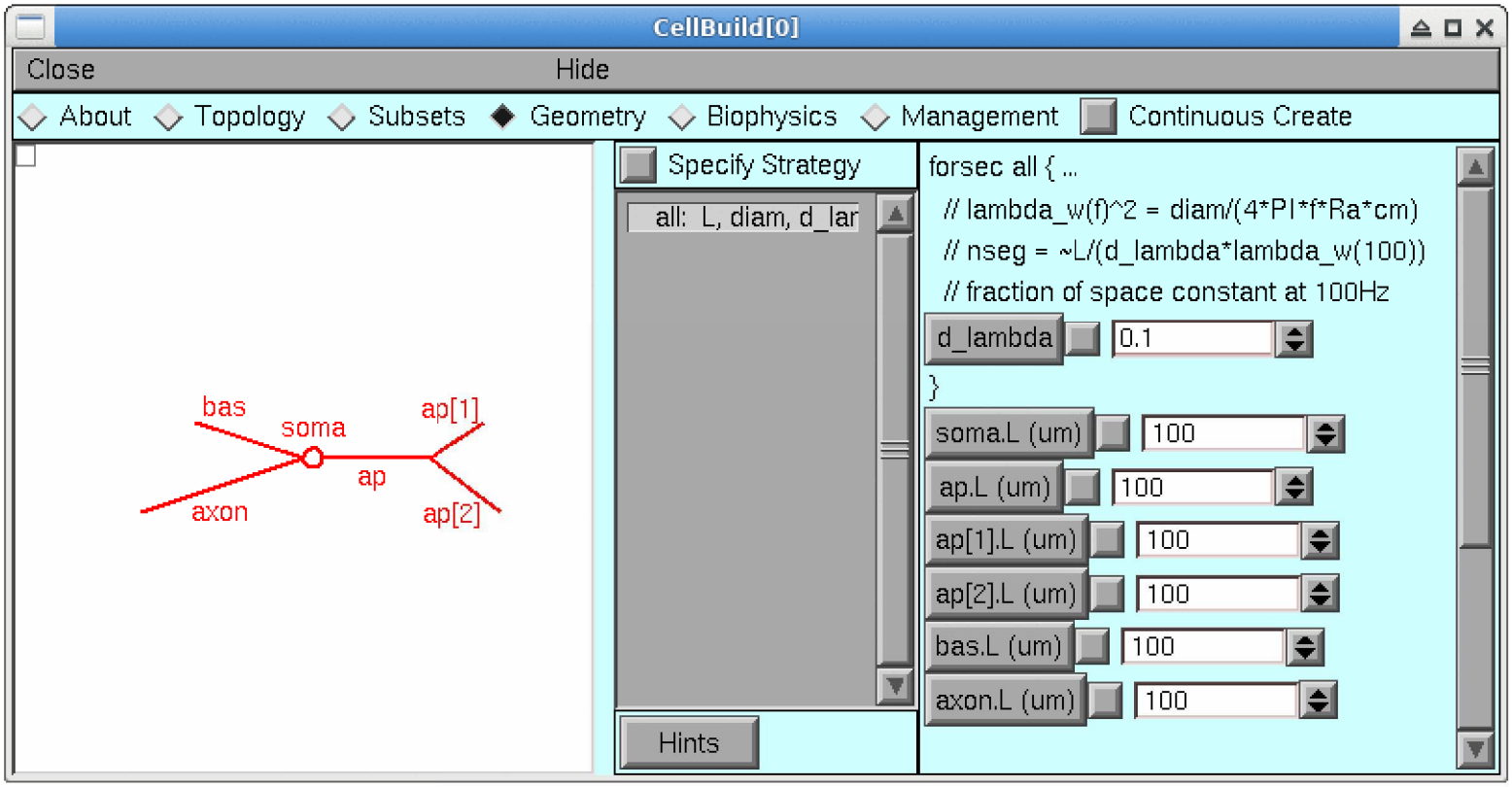
Geometry strategy



Each section has a different L and diam.

Compartmentalize according to $\lambda_{100\ Hz}$ (d_lambda rule).

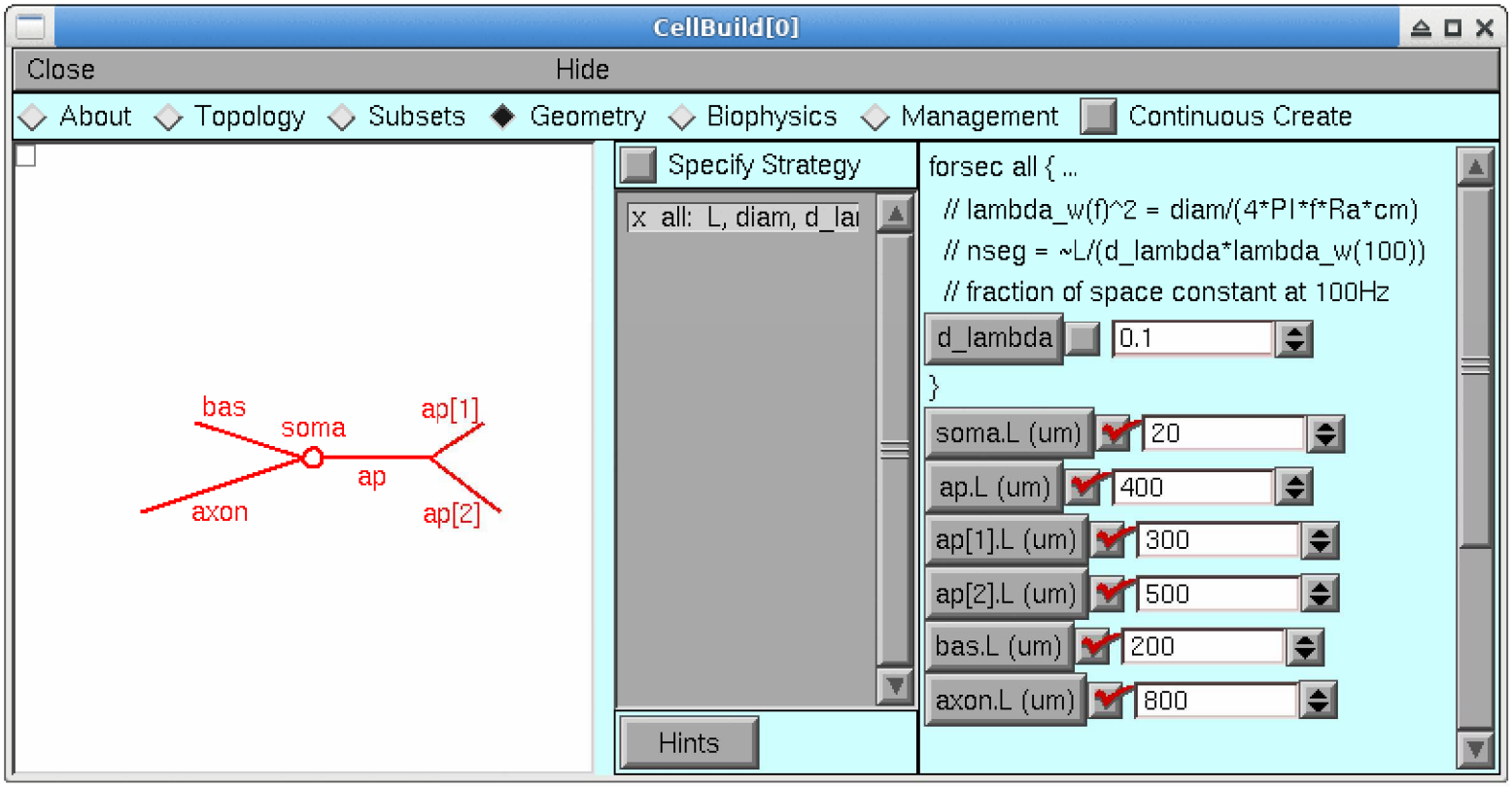
Implementing geometry strategy



When strategy is complete, turn "Specify Strategy" OFF and start assigning values to parameters.

d_lambda = 0.1 at 100 Hz usually gives good spatial accuracy.

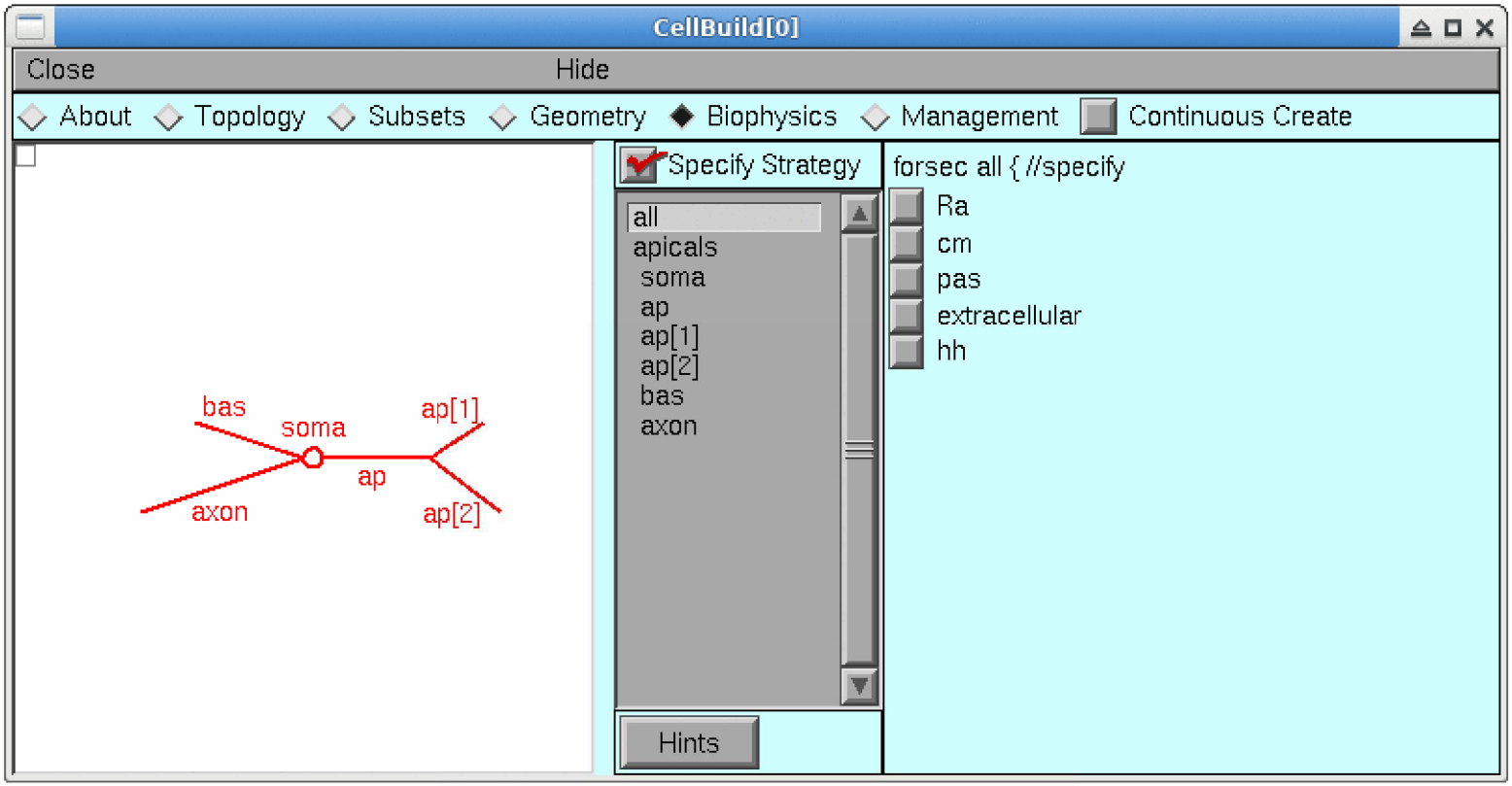
Implementing geometry *continued*



Set L and diam for all sections.

Time to save to a session file!

Biophysics

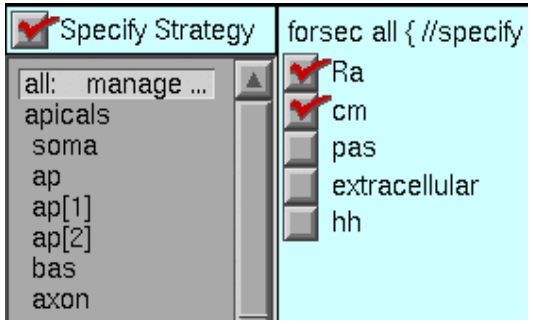


"Specify Strategy" is ON.

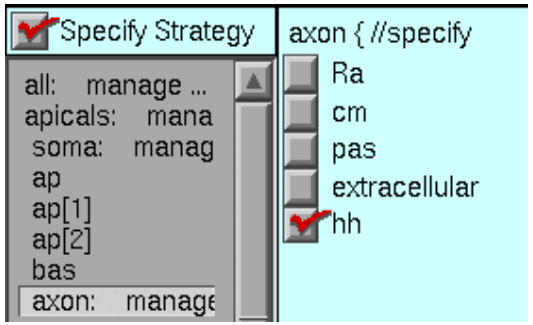
Base the plan on shared properties.

Biophysics strategy

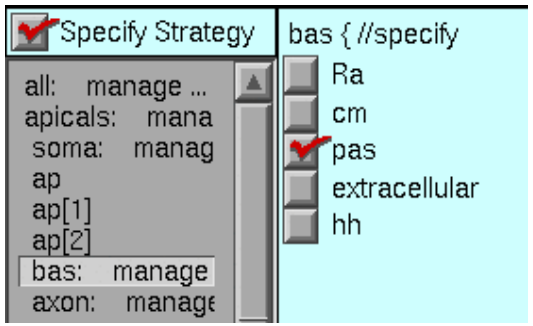
Ra and cm are homogeneous



apicals, soma and axon have hh

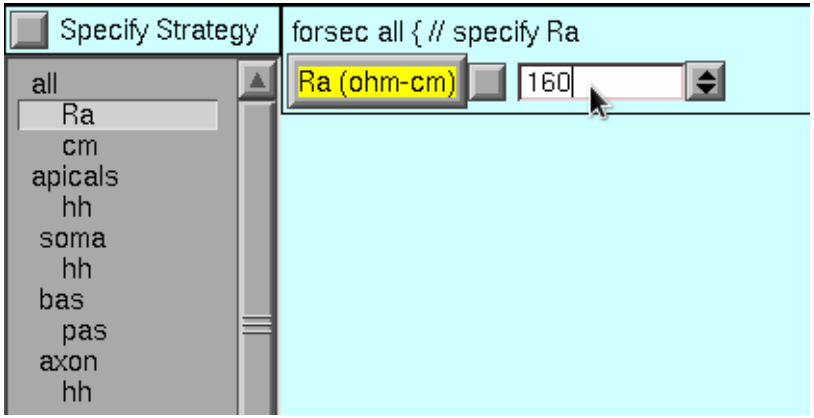


bas has pas

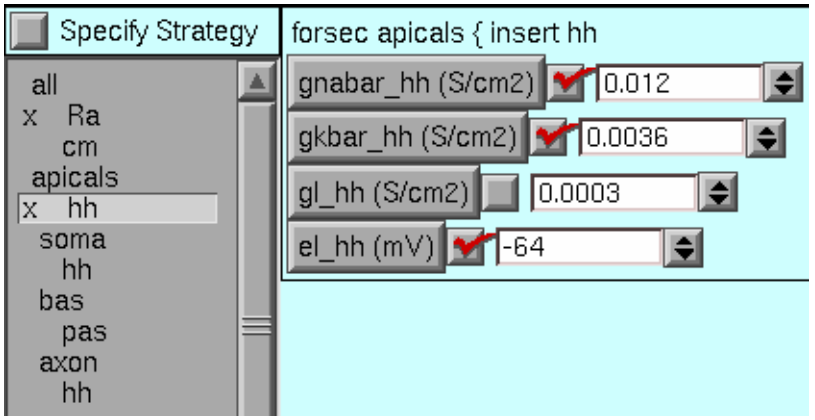


Implementing biophysics strategy

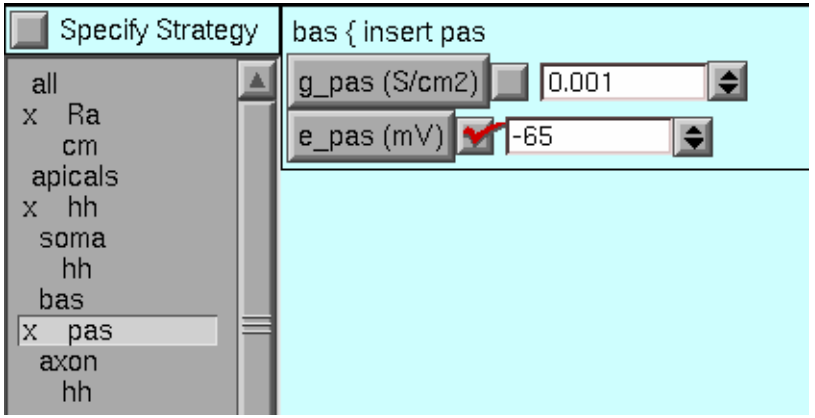
Set Ra



Change apicals hh params



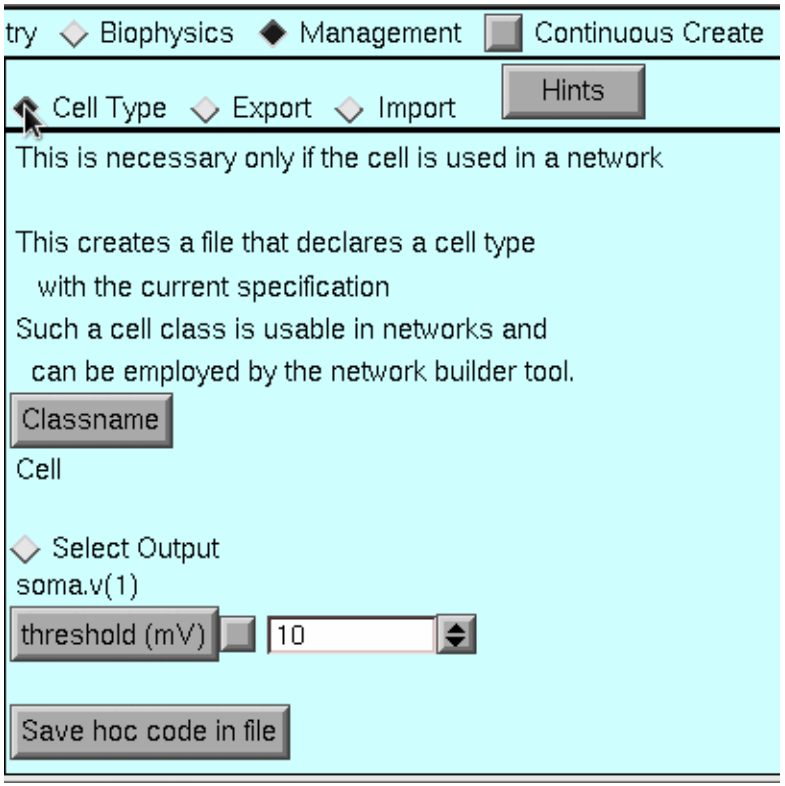
Shift e_pas in bas



Save another session file!!

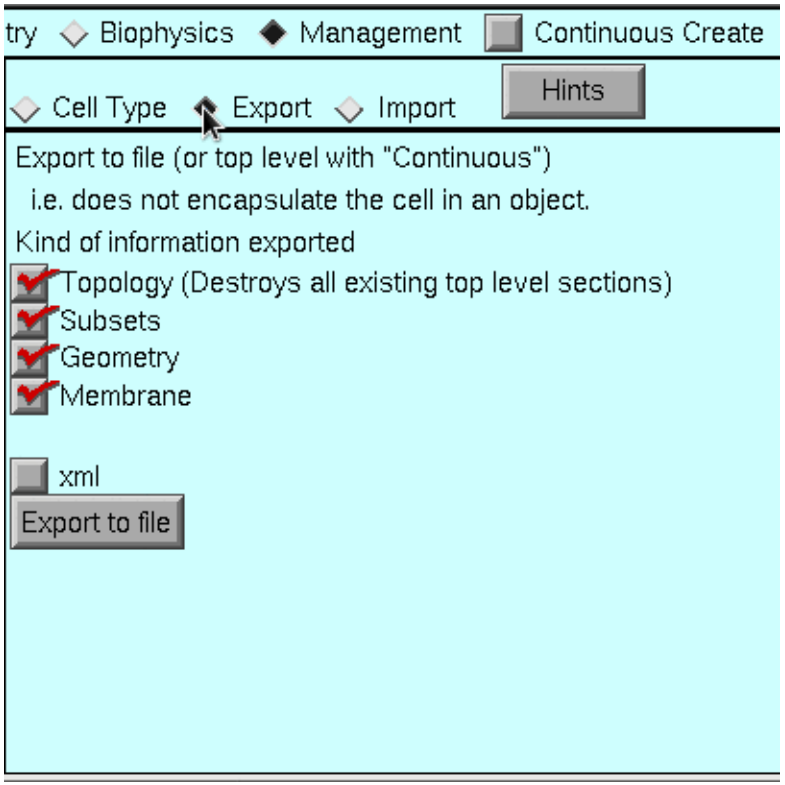
Management

Option 1: save as a Cell Type for use in a network



Management *continued*

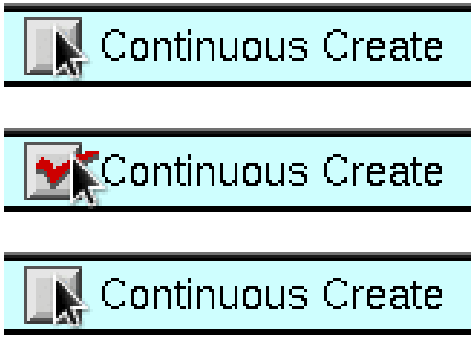
Option 2: save as hoc file



Management *continued*

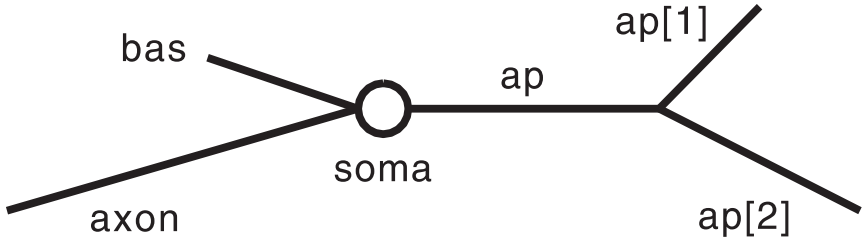
Option 3: export to interpreter

Toggle Continuous Create ON and OFF



or just leave it ON all the time.

Step 2: using the GUI to build an interface for running simulations



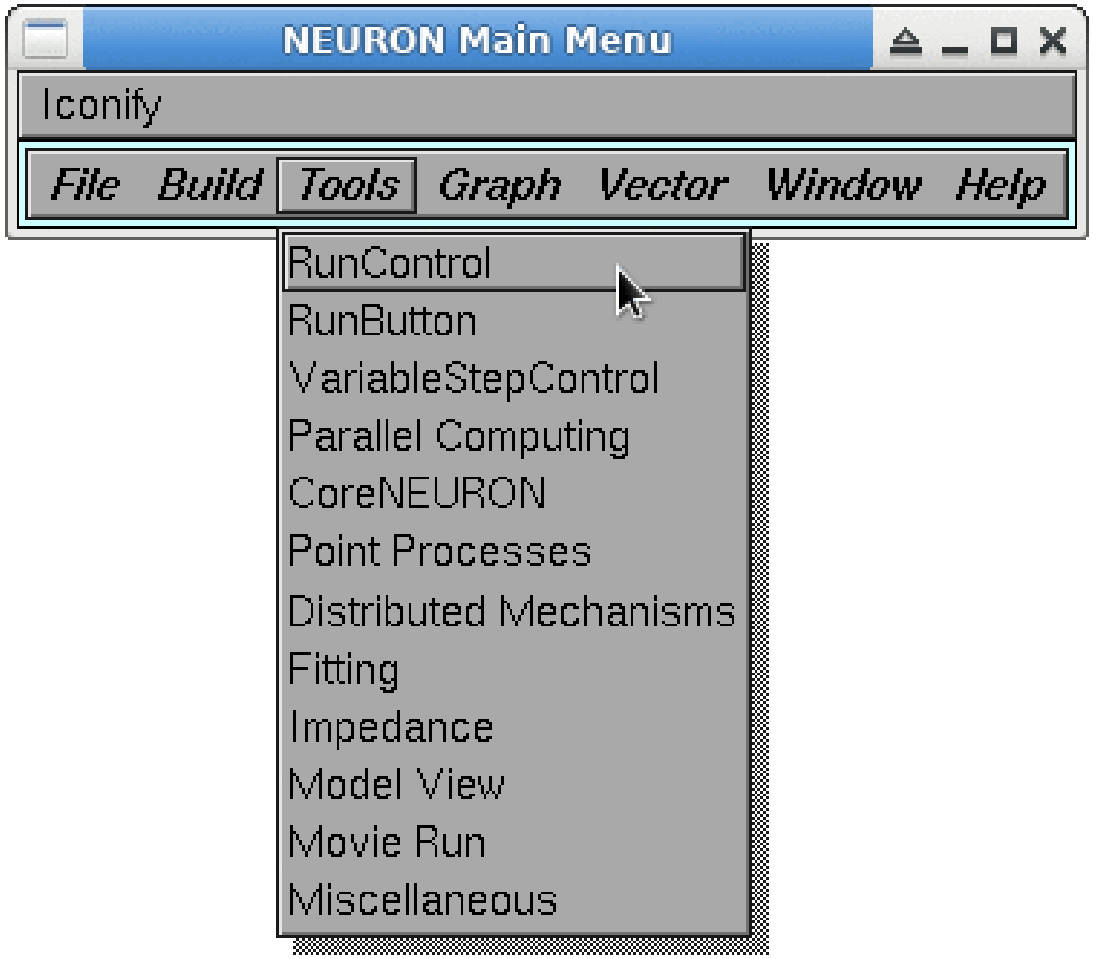
We want to

- attach a stimulating electrode
- evoke an action potential
- show time course of Vm at soma
- show Vm along a path from one end of the cell to the other

We need

- a "Run" button
- graphs to plot results
- a stimulator

Get a "Run" button



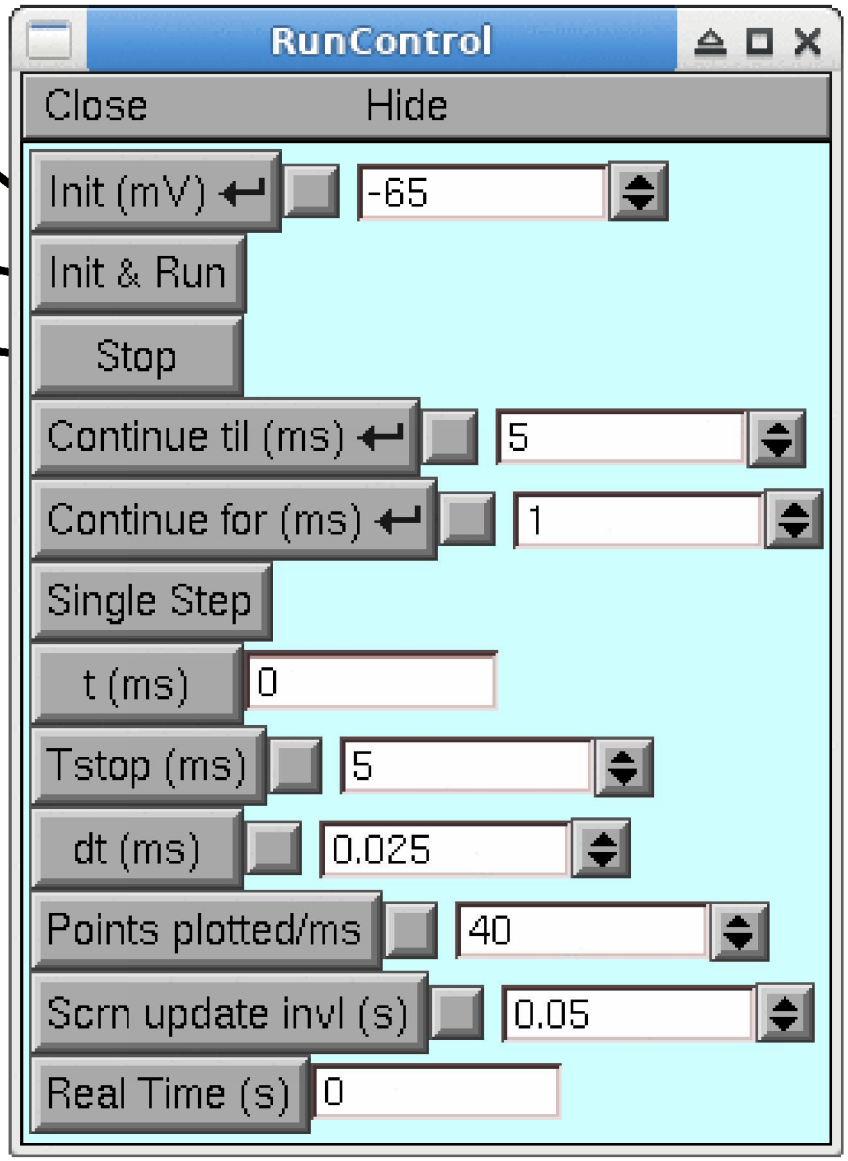
NEURON Main Menu / Tools / RunControl

RunControl panel

Init sets time to 0,
Vm to displayed value, and
conductances to steady-state

Init & Run does an Init,
then starts a simulation

Stop interrupts the simulation

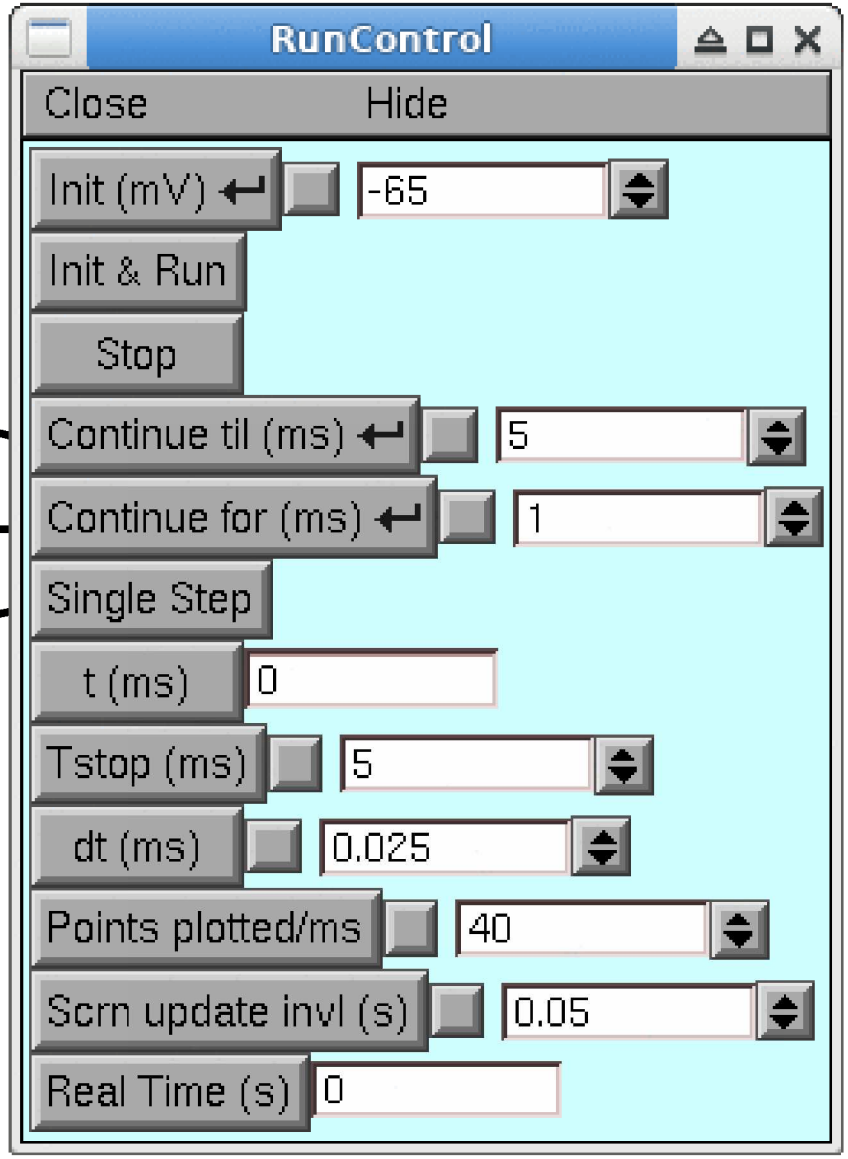


RunControl panel

Continue til runs until displayed time

Continue for runs for displayed interval

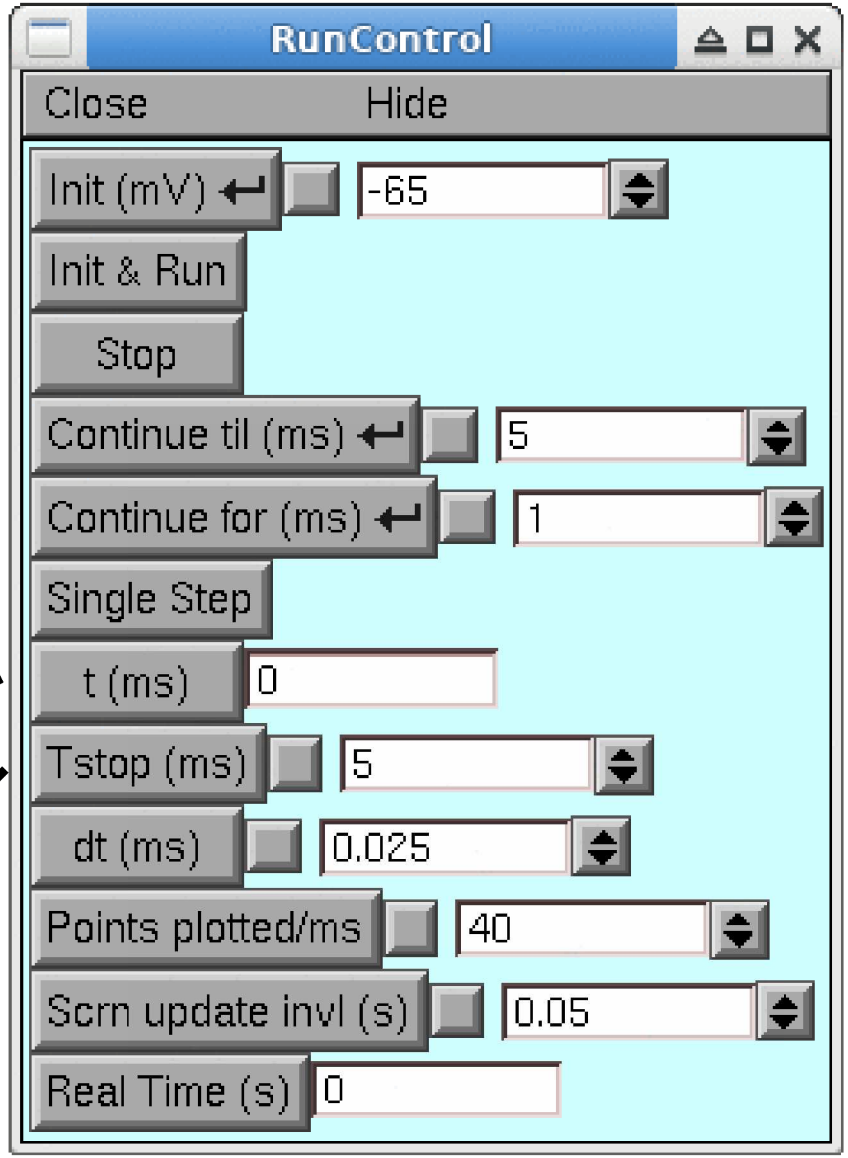
Single step advances by 1/(**Points plotted/ms**)



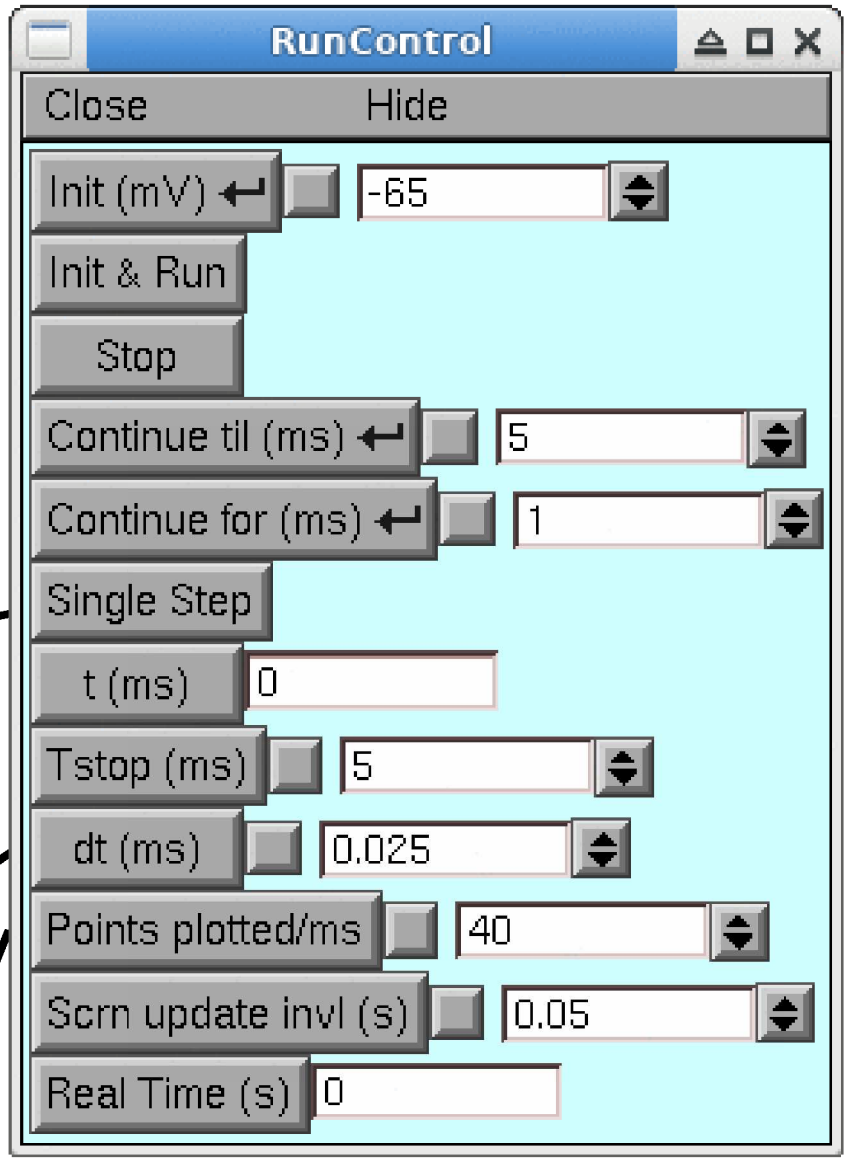
RunControl panel

t numeric field shows model time

Tstop specifies when simulation ends



RunControl panel



Single step advances by 1/(**Points plotted/ms**)

dt is integration time step; must be integer fraction of 1/(**Points plotted/ms**)

Points plotted/ms is 1/plotting interval

RunControl panel

Init sets time to 0,
Vm to displayed value, and
conductances to steady-state

Init & Run does an Init,
then starts a simulation

Stop interrupts the simulation

Continue til runs until displayed time

Continue for runs
for displayed interval

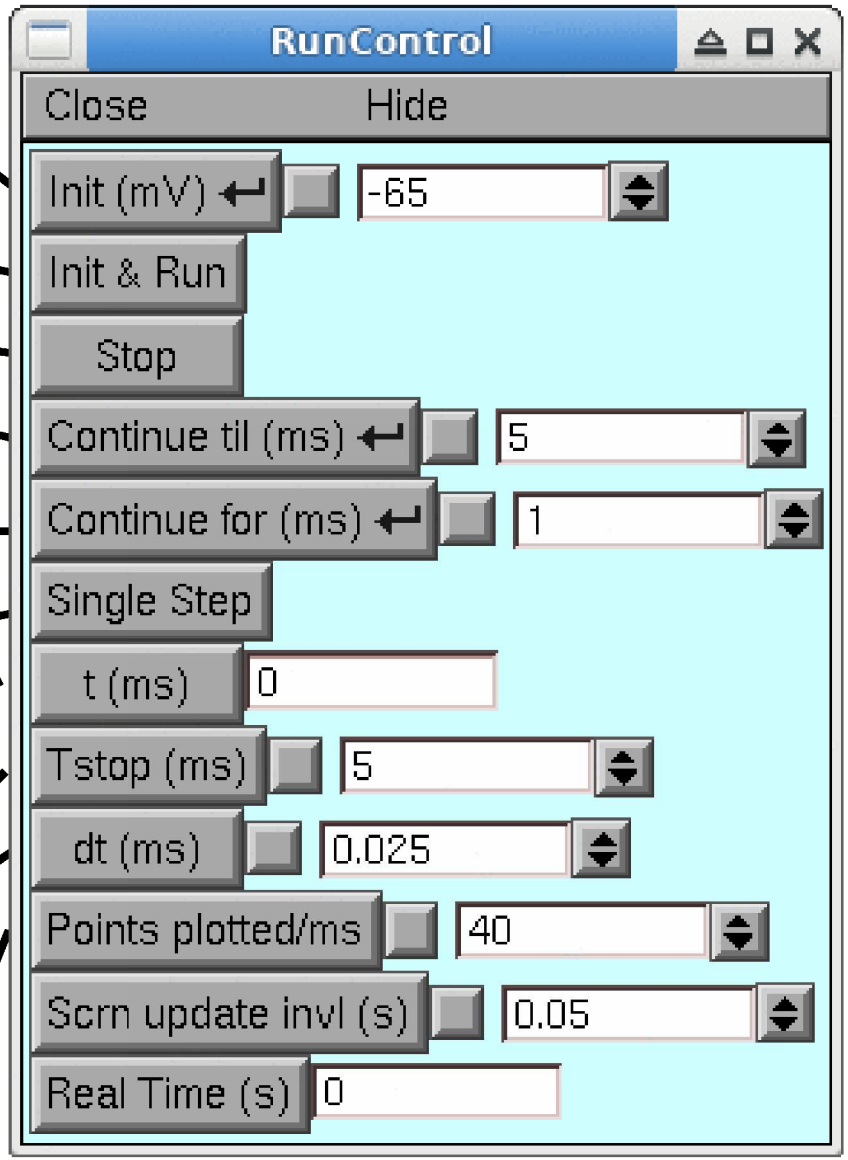
Single step advances by
1/(**Points plotted/ms**)

t numeric field shows model time

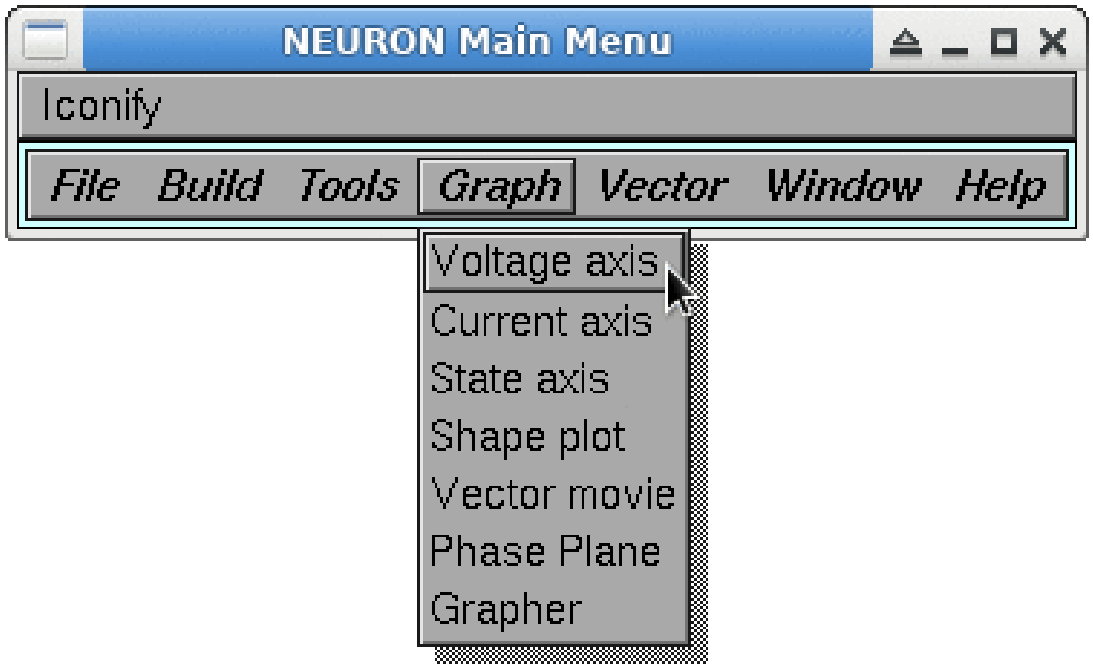
Tstop specifies when simulation ends

dt is integration time step;
must be integer fraction of
1/(**Points plotted/ms**)

Points plotted/ms is 1/plotting interval

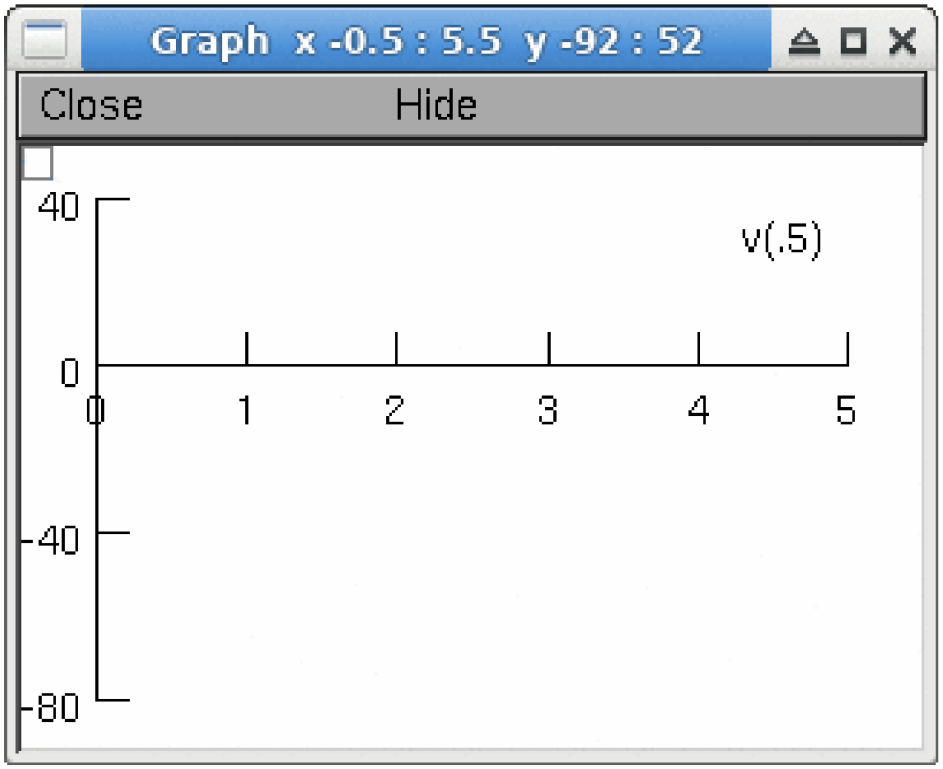


We need to plot Vm(t) at soma



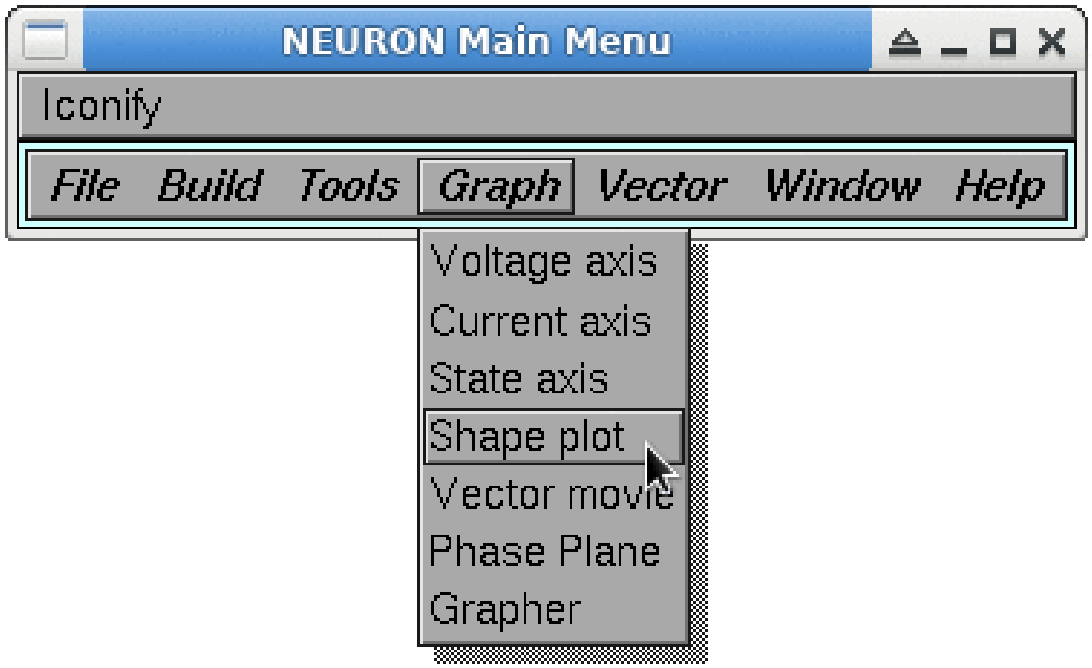
NEURON Main Menu / Graph / Voltage axis

Graph window



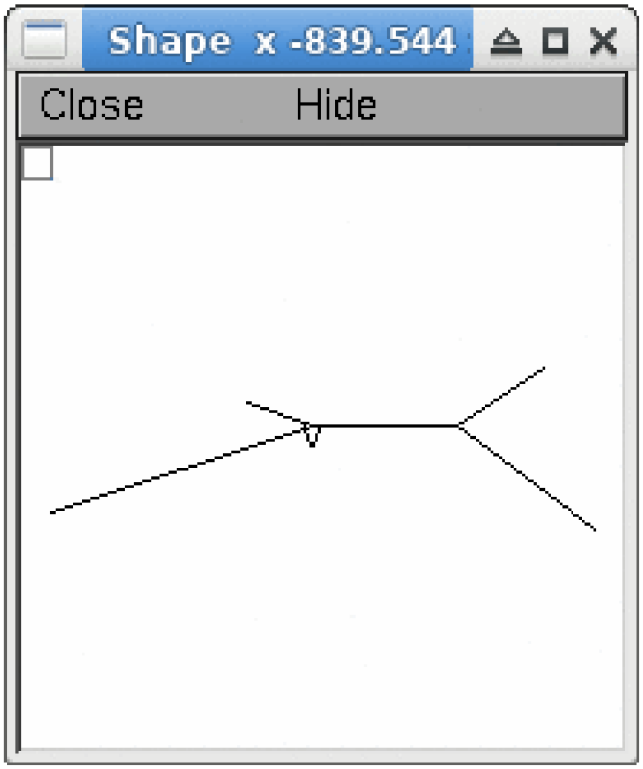
v(.5) is Vm at middle of default section
(first section in the CellBuilder)

We need to plot Vm along a path



NEURON Main Menu / Graph / Shape plot

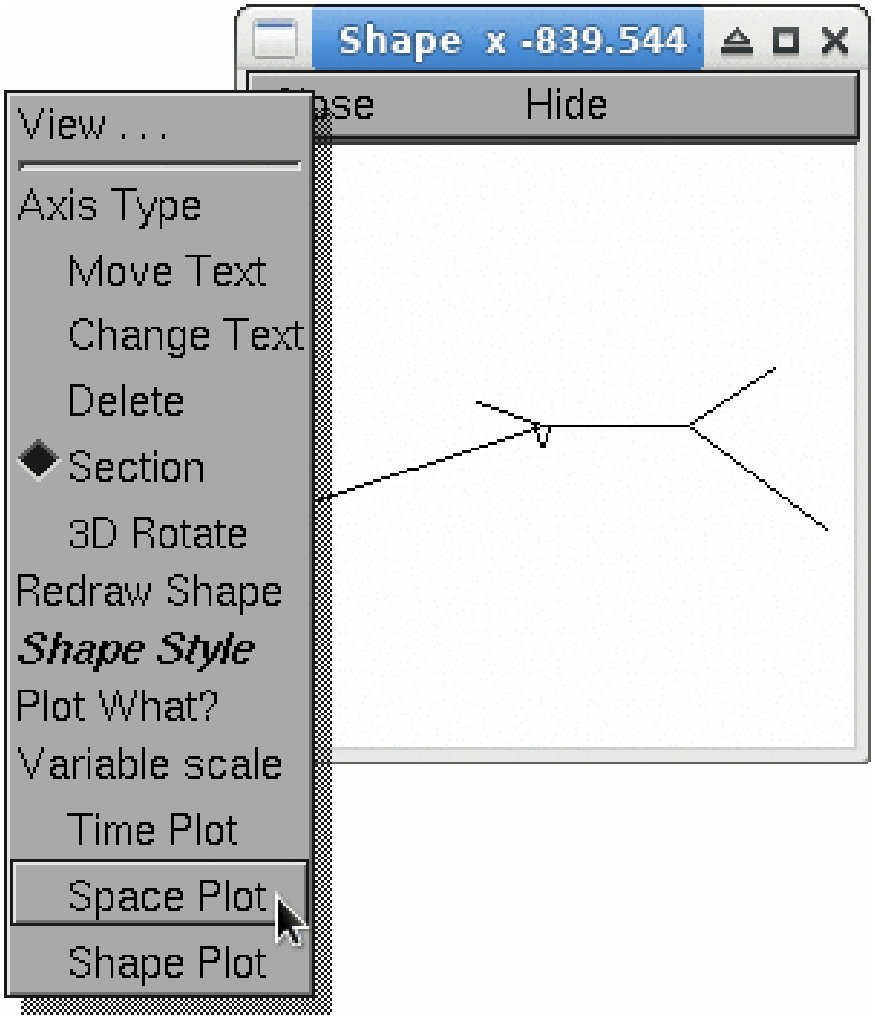
Bringing up a space plot



Use this "shape plot" to create a "space plot".

Click on its "menu box" . . .

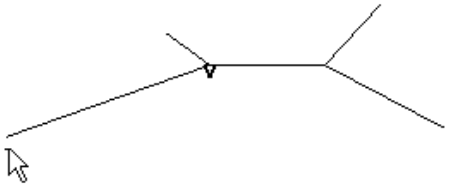
Bringing up a space plot *continued*



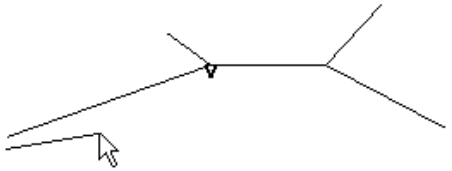
. . . and scroll down to "Space Plot".

Bringing up a space plot *continued*

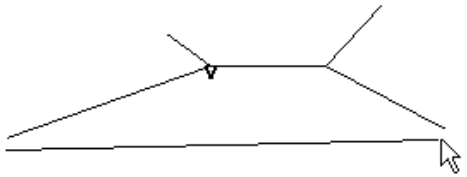
Click just left of the shape



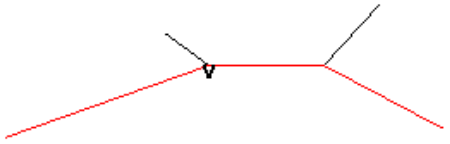
Hold button down while dragging from left . . .



. . . to right . . .

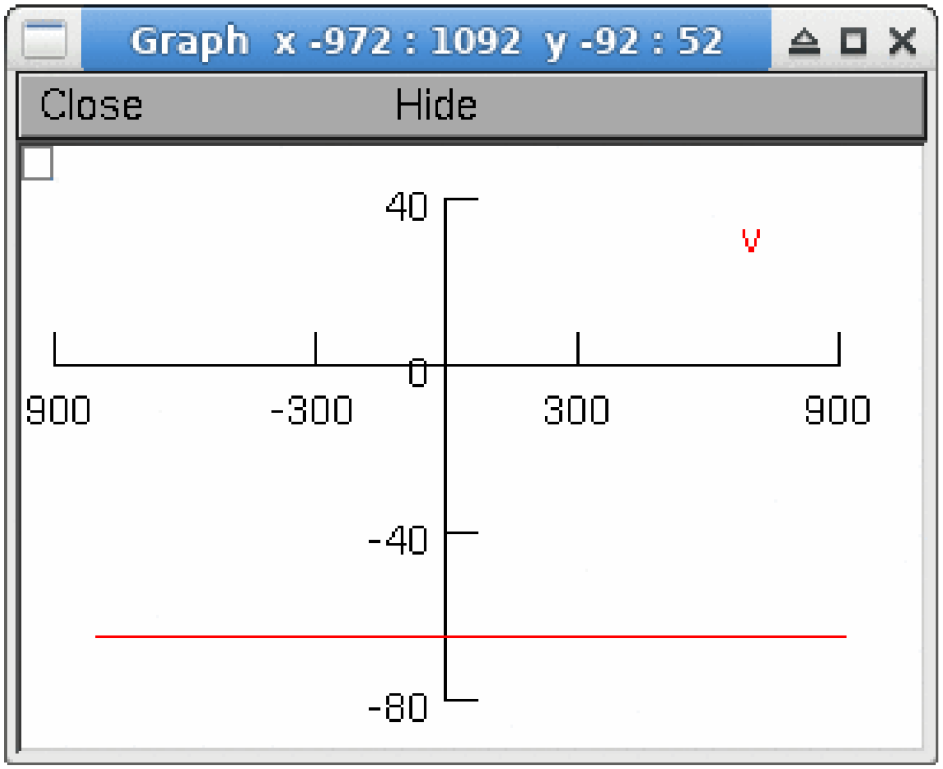


. . . then release button.



This pops up . . .

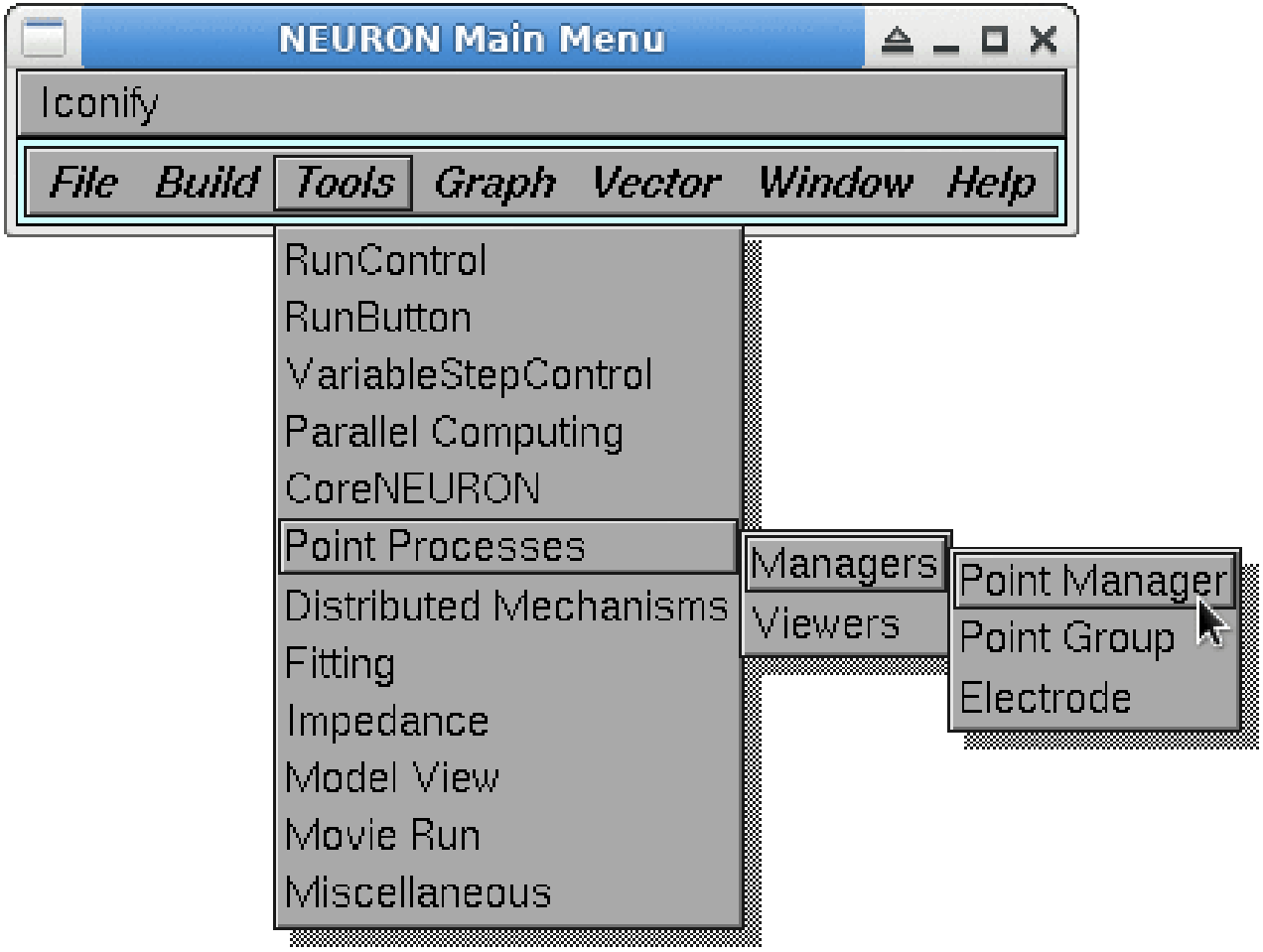
Space plot



. . . a plot of Vm vs. distance along a path.

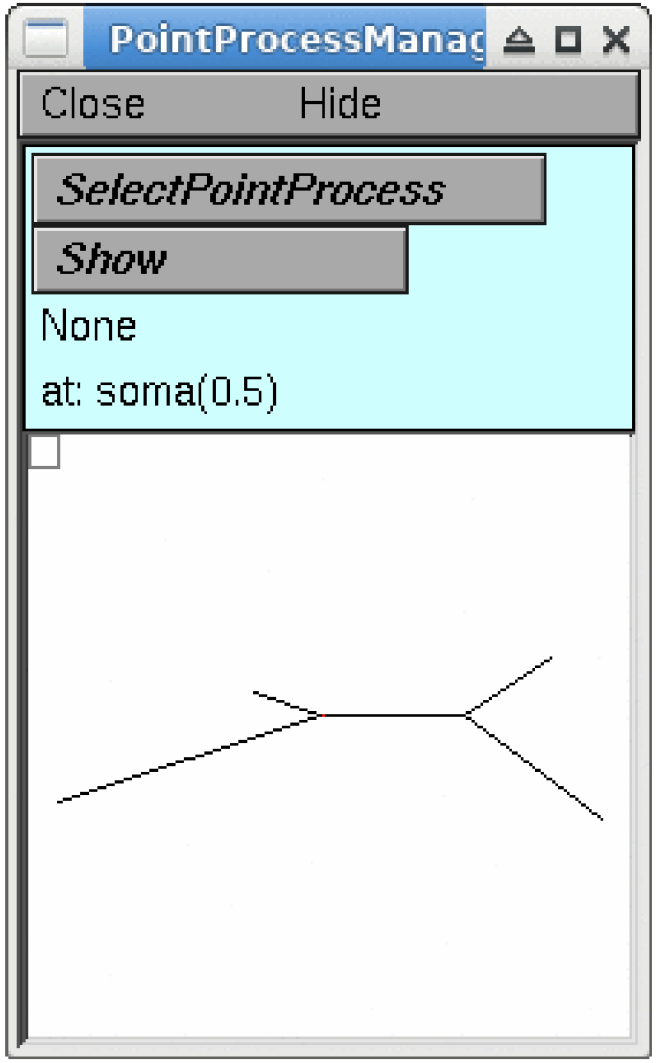
Better save a session file.

We need a stimulator



NEURON Main Menu / Tools / Point Processes / Managers / Point Manager

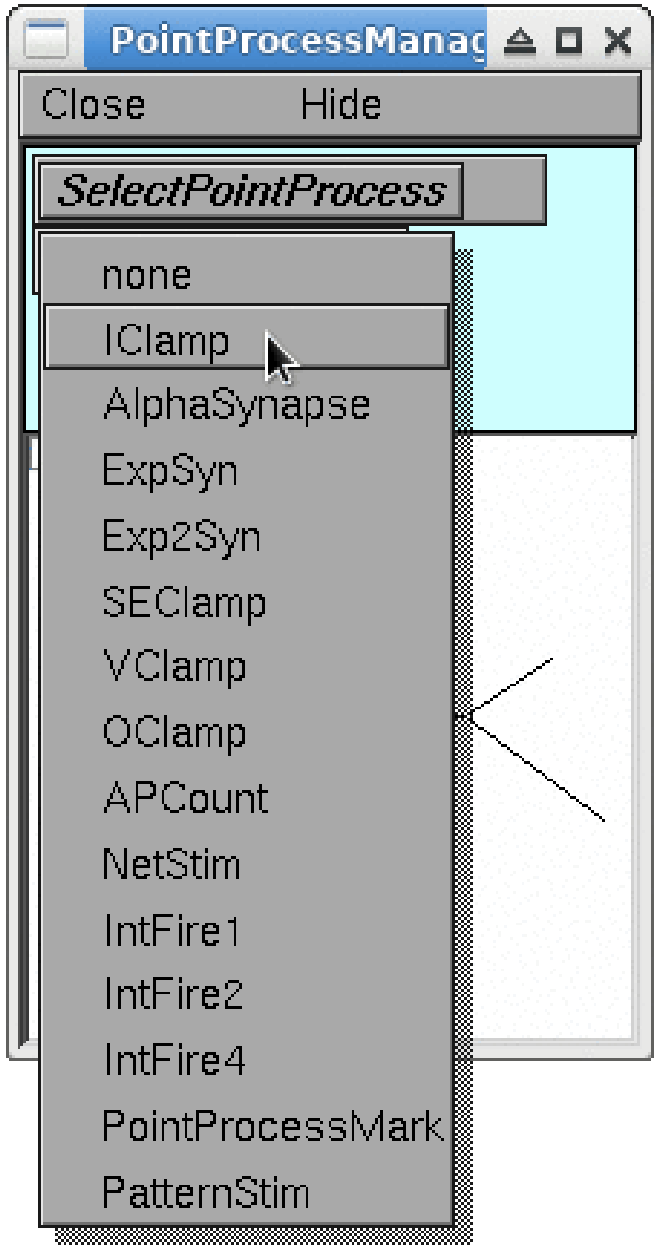
PointProcessManager window



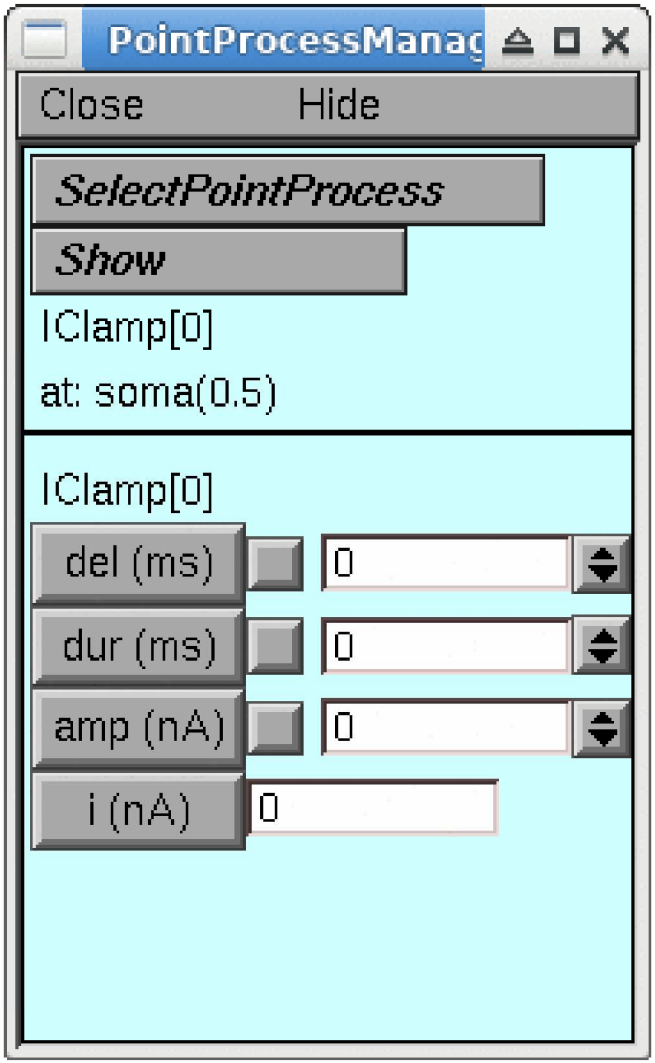
To make this an IClamp . . .

Creating an IClamp

. . . click on SelectPointProcess and scroll down to IClamp.

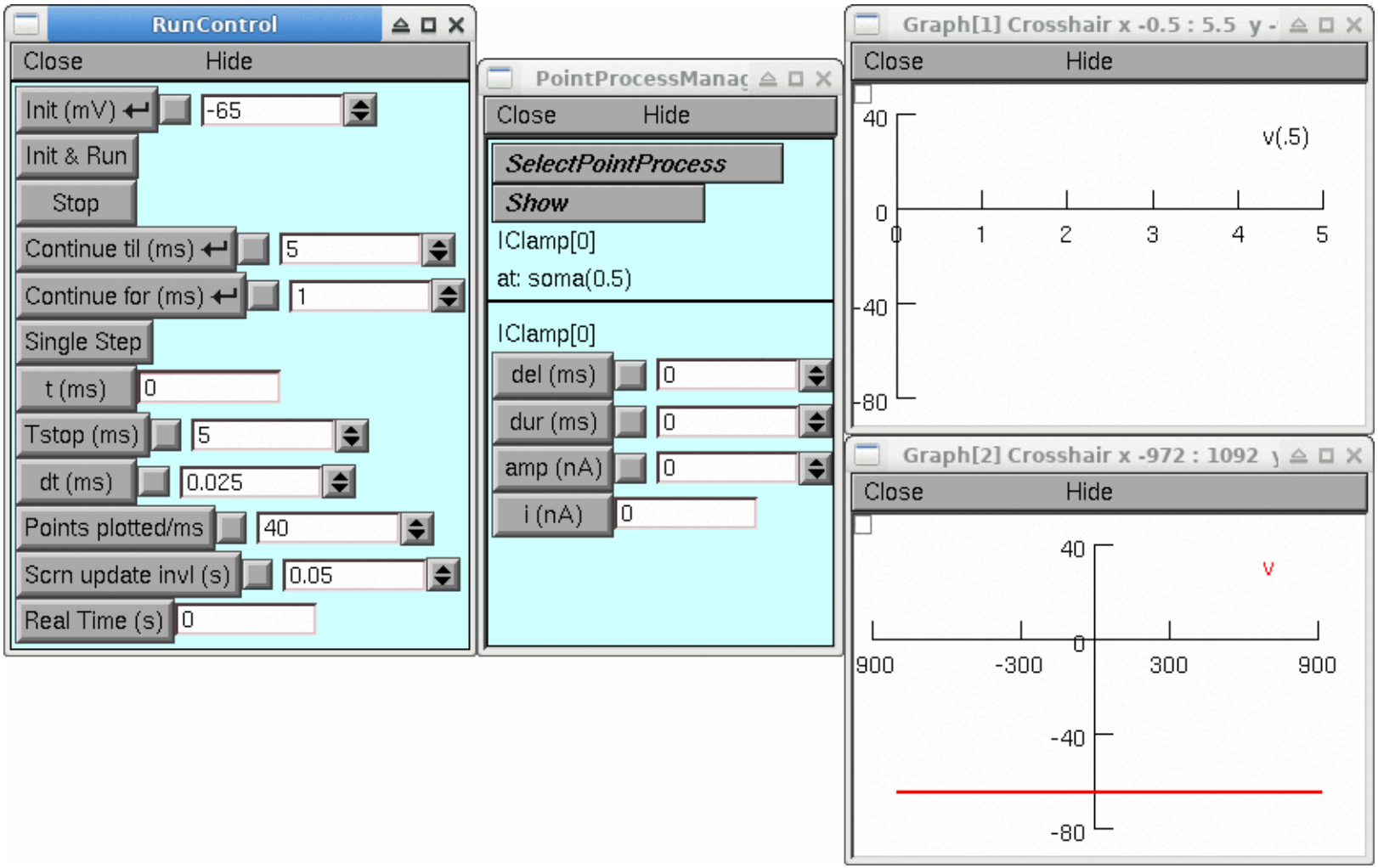


IClamp parameter panel



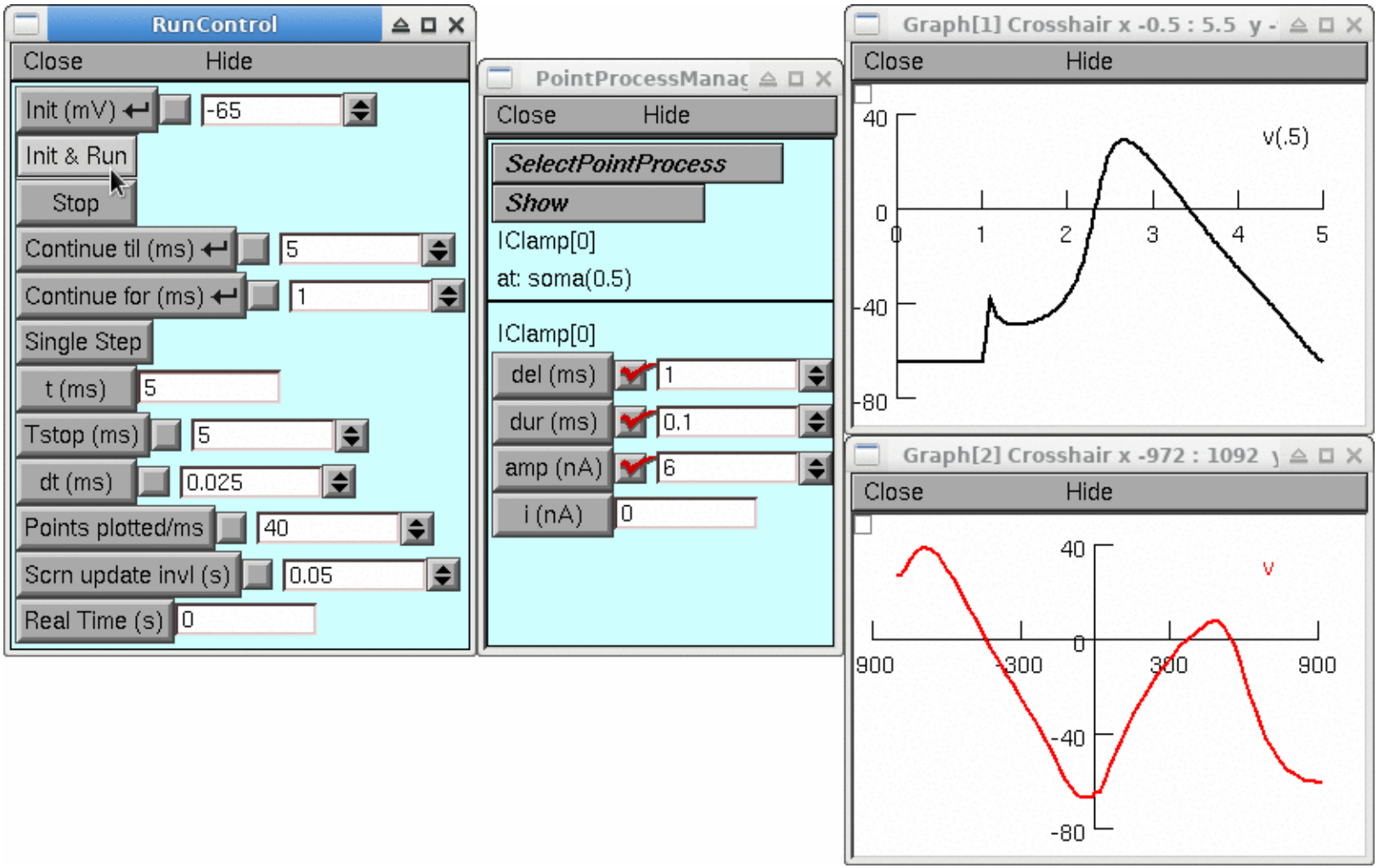
Next: set parameter values.

Our user interface

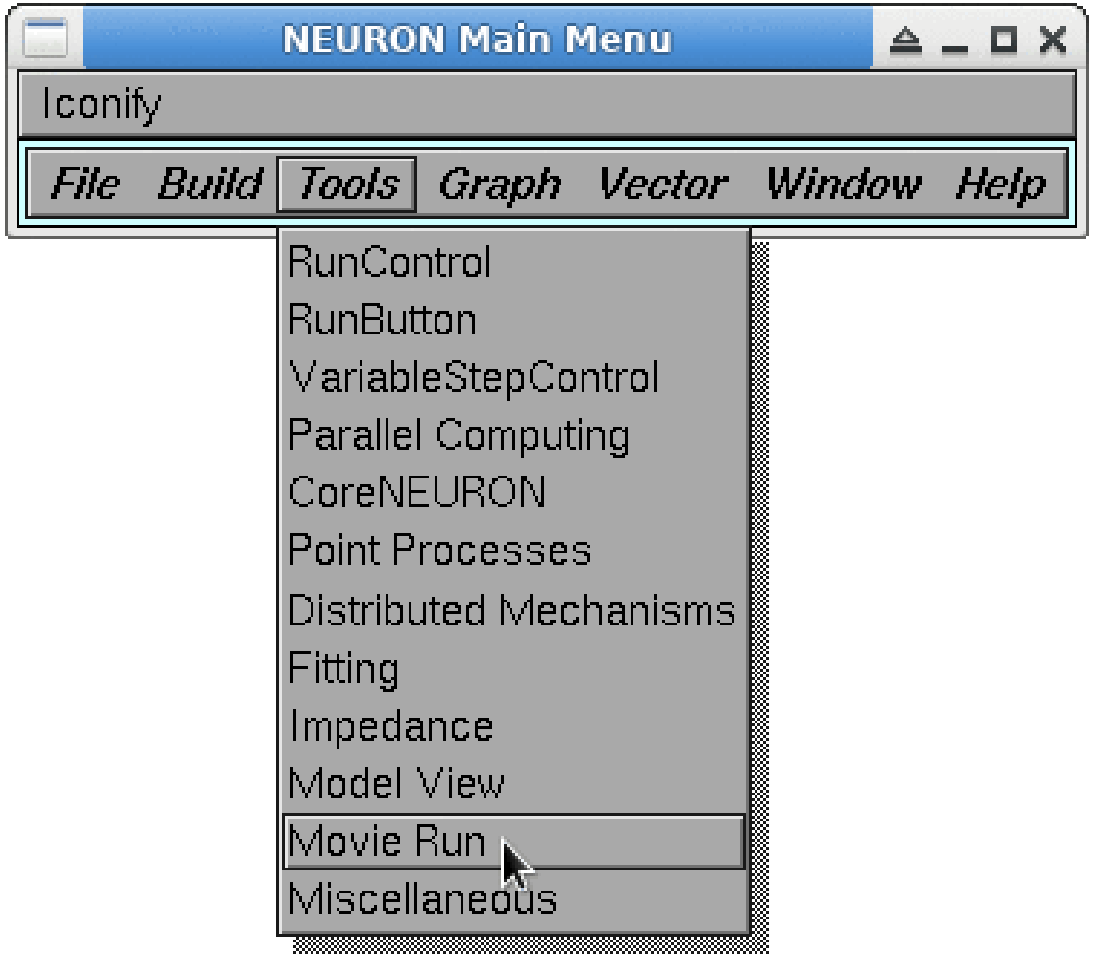


Time to save to a new session file!

It works!

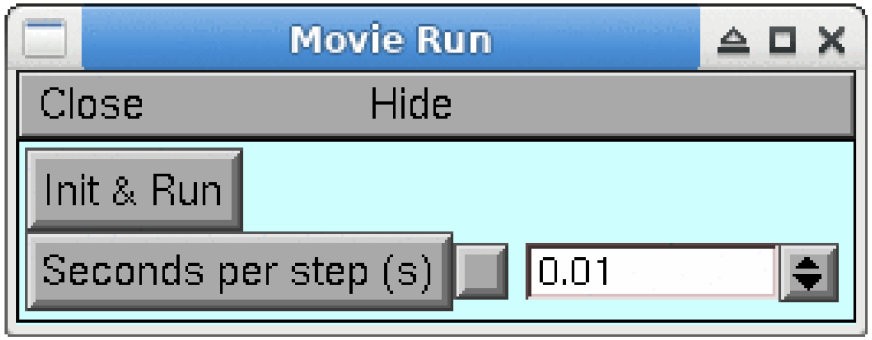

RunControl
Close
Hide
Init (mV)
-65
Init & Run
Stop
Continue til (ms)
5
Continue for (ms)
1
Single Step
t (ms)
5
Tstop (ms)
5
dt (ms)
0.025
Points plotted/ms
40
Scrn update invl (s)
0.05
Real Time (s)
0
PointProcessManag
Close
Hide
SelectPointProcess
Show
IClamp[0]
at: soma(0.5)
IClamp[0]
del (ms)
1
dur (ms)
0.1
amp (nA)
6
i (nA)
0
Graph[1] Crosshair x -0.5 : 5.5 y -
Close
Hide
v(.5)
40
0
-40
-80
0
1
2
3
4
5
Graph[2] Crosshair x -972 : 1092
Close
Hide
v
40
0
-40
-80
900
-300
300
900

How to get nice space plot "movies"



NEURON Main Menu / Tools / Movie Run

Space plot "movies" *continued*



Movie Run / Init & Run