

# 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

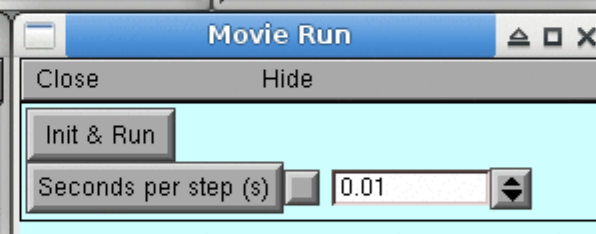
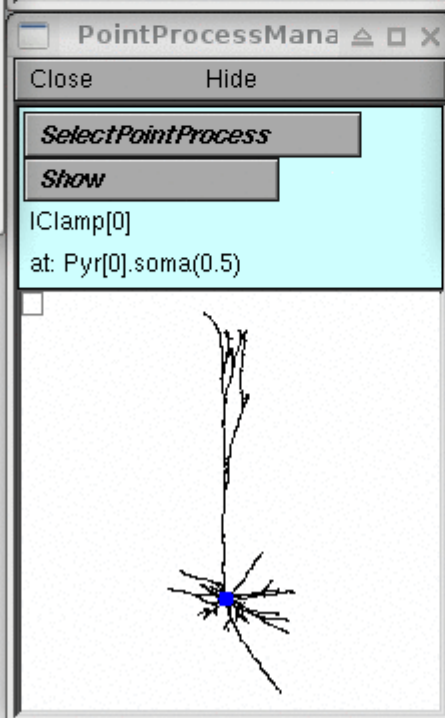
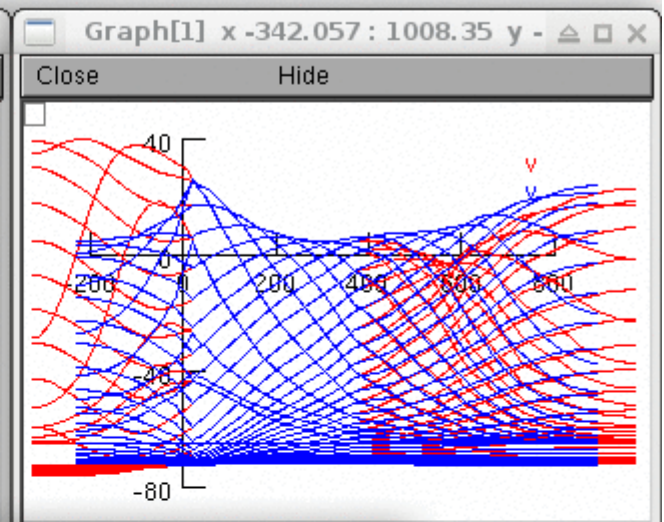
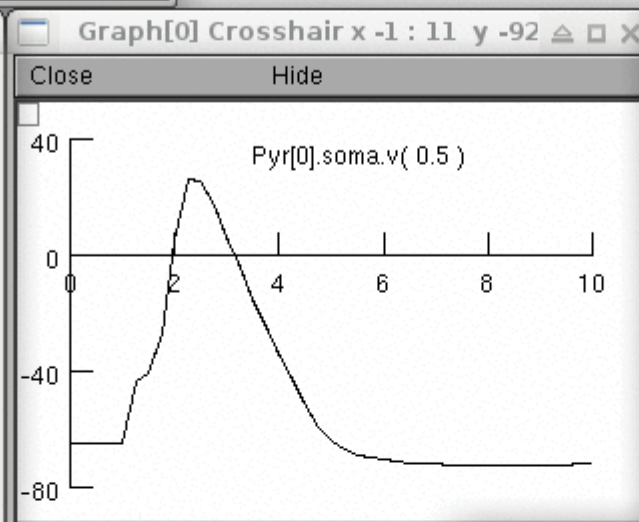
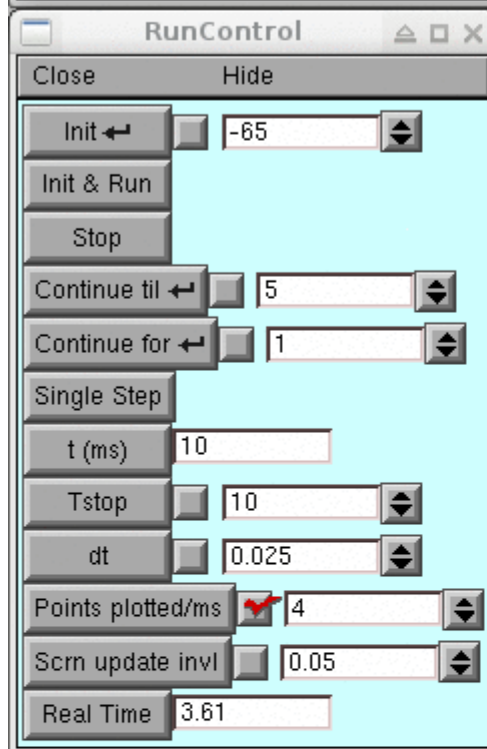
*"I don't need to hear this. I use Python, and the GUI doesn't work with Python."--Anonymous dodo*

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

Example: pyrttest.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('pyrttestrig.ses') # user interface
                        # built with NEURON's GUI tools

python -i pyrttest.py
```



# 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: pyrttest.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.