Tutorial: Adding Run Time Interactive Graphics to Physics Codes

Viktor K. Decyk
UCLA

Interactive graphics based on Python: using tkinter, numpy, and matplotlib

Tkinter is a graphical user interface (GUI) included with Python

Model-View-Controller (MVC) design pattern, in multi-threaded shared memory system

- MVC establishes a separation of concerns
- Controller and graphics runs in main thread, physics runs in other thread(s)
- Controller is event driven, and it controls View, and synchronizes with physics
- Only View imports matplotlib, does not generally process events
- Physics may use OpenMP in Fortran or C dynamic libraries, does not use tkinter or matplotlib

Separate Python scripts cannot safely share global memory

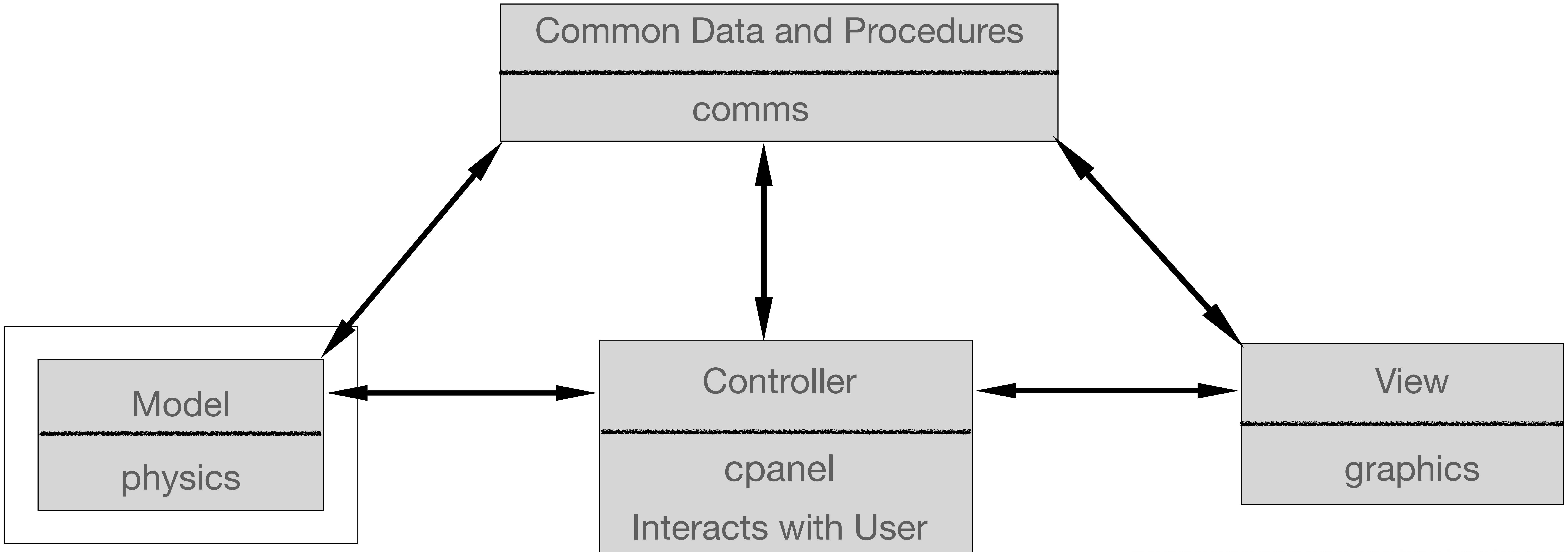
- But they can share a common global memory

Only two ways to safely communicate between threads

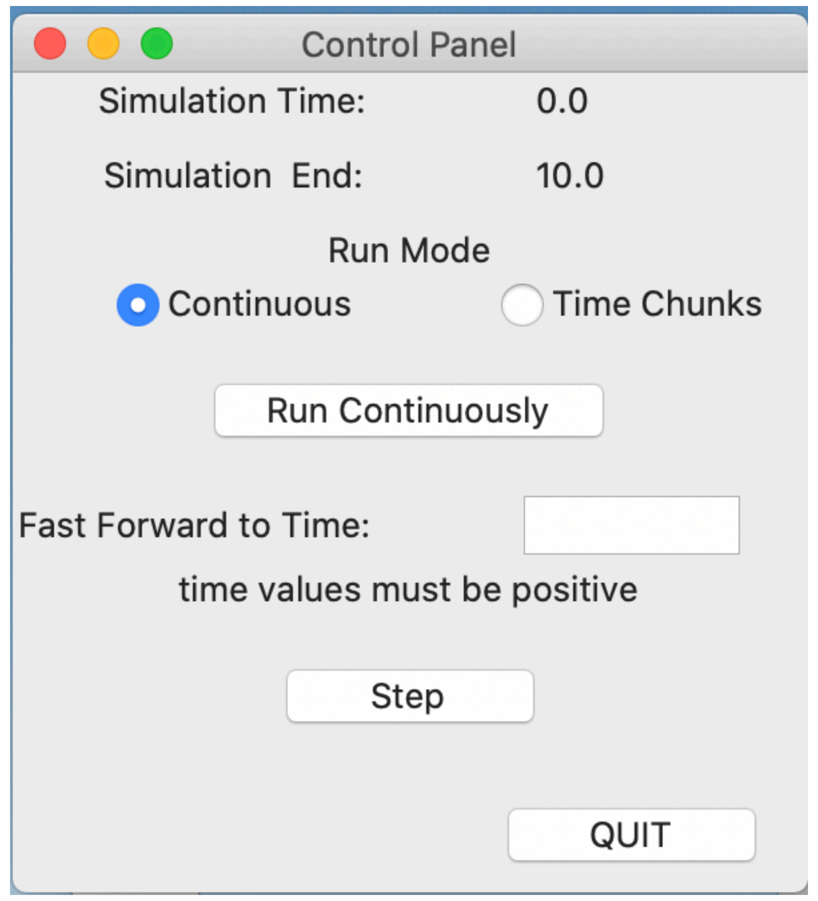
- By generating events
- By writing to thread-safe queue objects

Alan D. Moore, "Python GUI Programming with Tkinter", 2021

Model-View-Controller Design Pattern



Physics runs in separate thread(s)



Control Panel

Simulation Time: 0.0

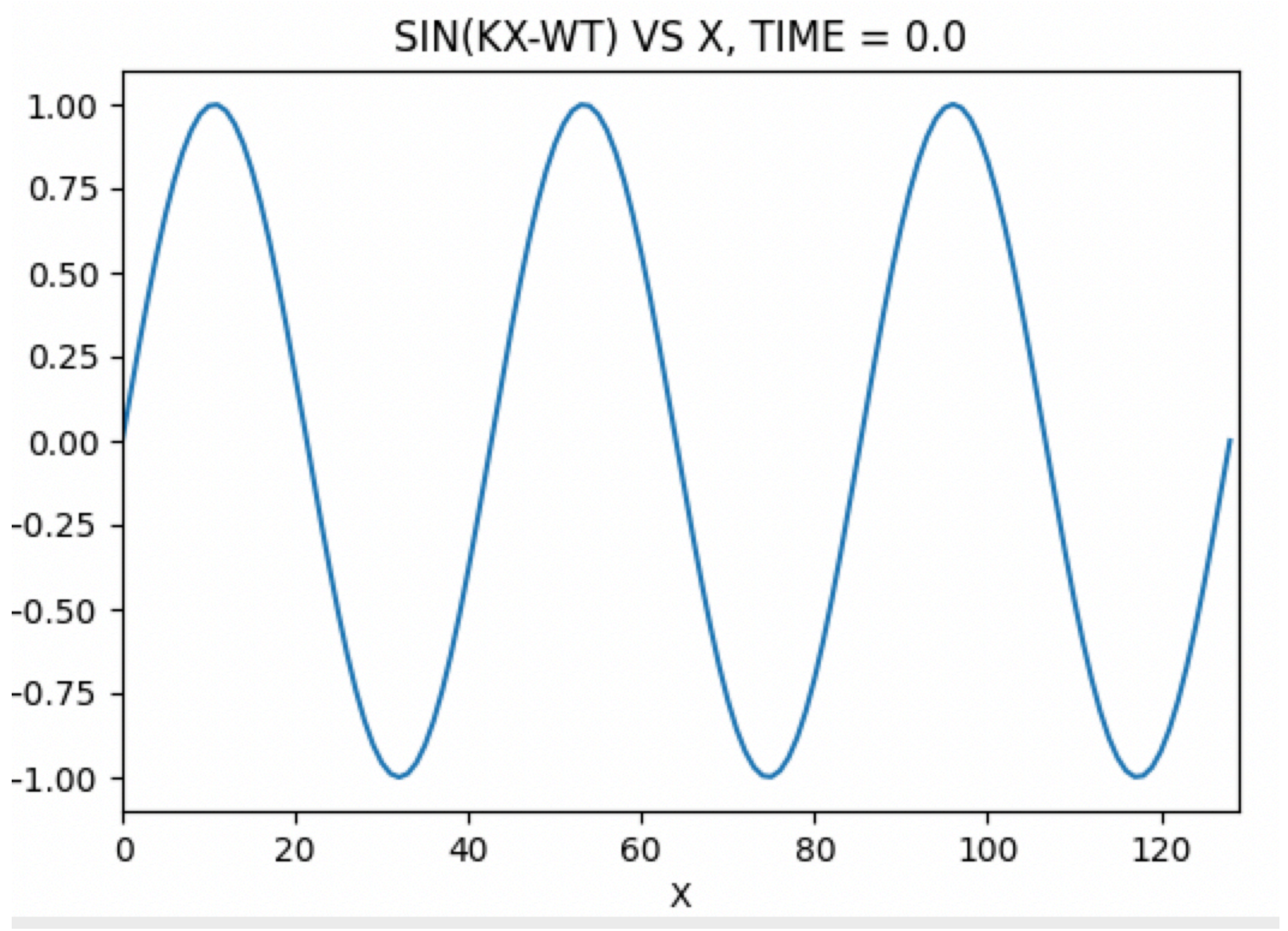
Simulation End: 10.0

Run Mode

● Continuous ○ Time Chunks

Fast Forward to Time:

time values must be positive



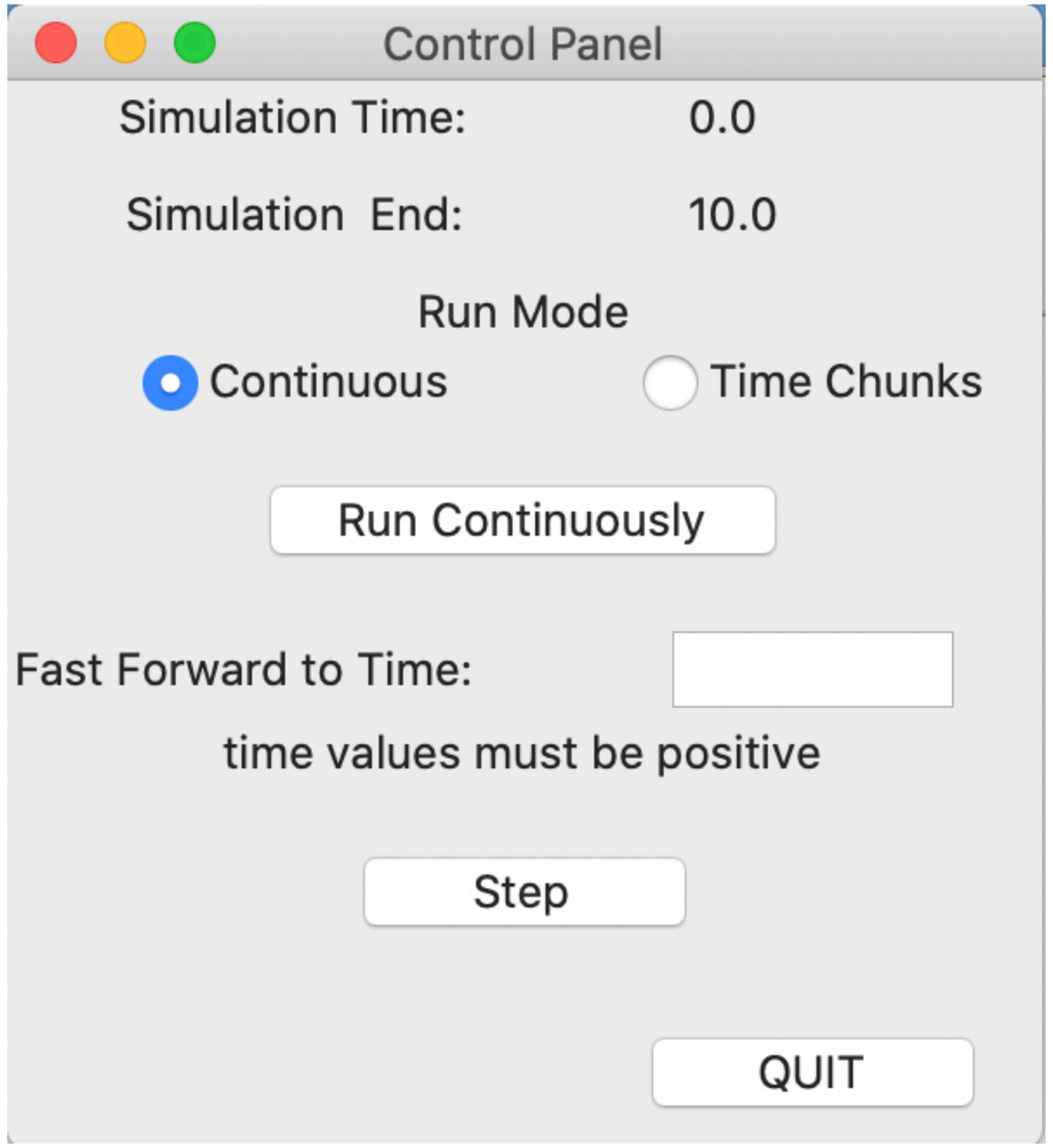
Demonstration

cd to the directory PICGUIdemo
execute: python3 cpanel.py
you should see the control panel and three plots in a window

The current simulation time and end time are shown
Step: Goes one time step and pauses
Run Continuously: Runs and plots without pausing

Since physics often runs faster than plots:
Fast Forward Time: Runs to designated time without plotting
Run Chunk: Runs without plotting to current + jump time

Quit: Terminate Program



Control Panel

Simulation Time: 0.0

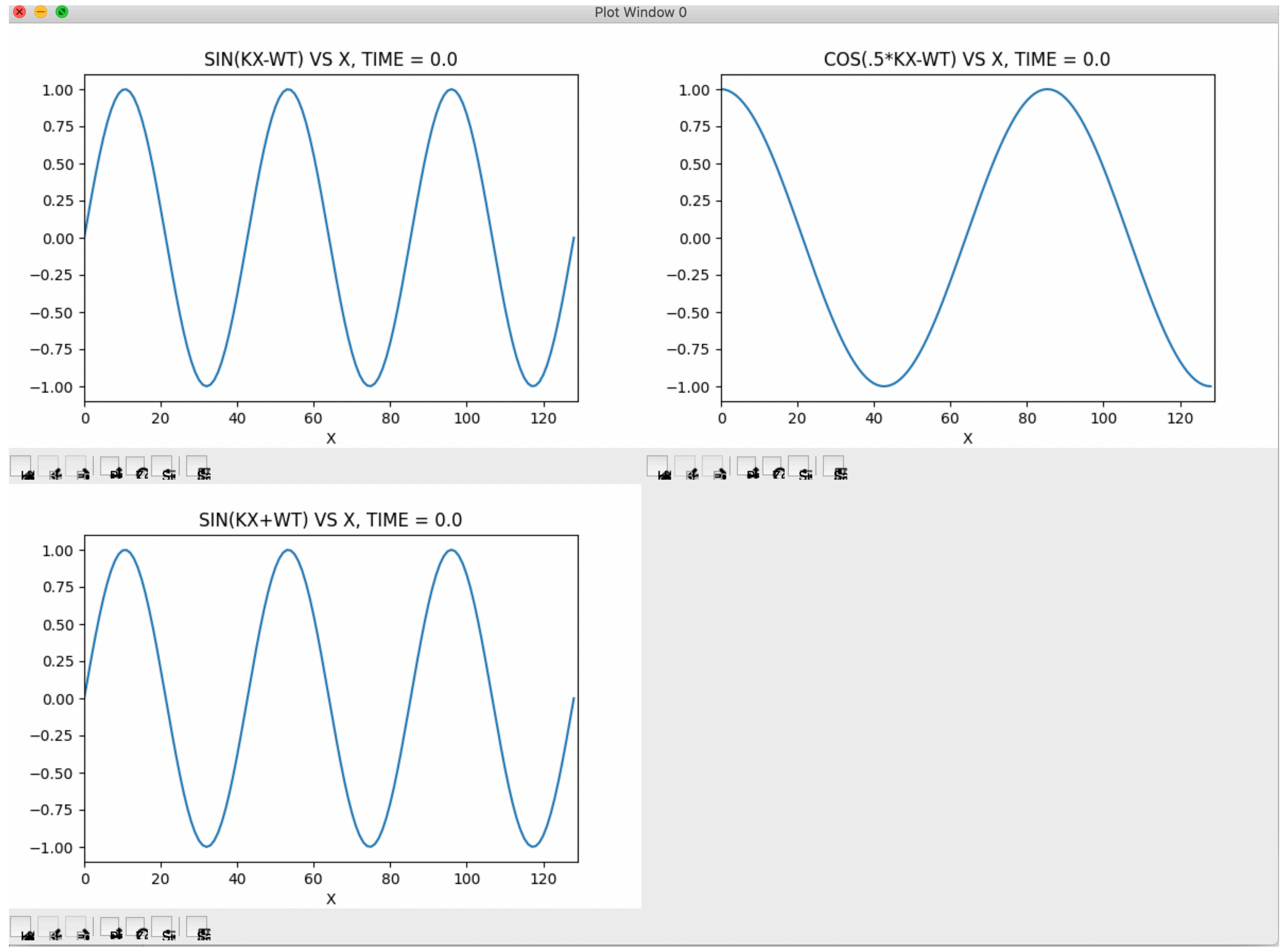
Simulation End: 10.0

Run Mode

● Continuous ○ Time Chunks

Fast Forward to Time:

time values must be positive



Plot Window 0
SIN(KX-WT) VS X, TIME = 0.0
COS(.5*KX-WT) VS X, TIME = 0.0
SIN(KX+WT) VS X, TIME = 0.0
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
20
40
60
80
100
120
X

Incorporating GUI features into your own physics program

What do you change in your code to use these features?

import comms

Before main loop, start gui, send 2 dictionaries to Controller and wait for message from controller

- plot_loc = {"Label for first plot":0,"Label for second plot":1,...}
- sim_data = {"DT": 0.1,"TEND": tend,...} plus other constants needed by your plots

2. At top of main loop, send current time to Controller
3. For each plot, send plot name and plot data and wait message from graphics
4. At end of main loop, wait to for message from controller

Demo physics code:

```
 global sfield
# send plot_loc, sim_data dictionaries and custom event to initialize plots
 comms.update_gui({"SIN(KX-WT) VS X": 0, "SIN(KX+WT) VS X": 1,
       "COS(.5*KX-WT) VS X": 2}, {"DT": 0.1,"TEND": tend,"NX": nx})
 gui_err = comms.check_run_status()
 if (gui_err=='QUIT'): exit()

#main loop
 for ntime in range(nstart,nloop):
  time = dt*float(ntime)
# send current time to GUI at each loop iteration
  comms.update_time(time)

# calculate sin(kx-wt)
  sfield[:nx+1] = numpy.sin(dnx*numpy.array(range(nx+1))-omega*time)
  comms.update_plot('SIN(KX-WT) VS X')
  gui_err = comms.check_plot_status()
  if gui_err != comms.plot_name:
   if (gui_err=='QUIT'): break

# wait for GUI at each loop iteration
  gui_err = comms.check_run_status()
  if (gui_err=='QUIT'): break
```

Incorporating GUI features into your own physics program

Start GUI as follows:

```
comms.update_gui(plot_loc,sim_data}) with your own dictionaries
gui_err = comms.check_run_status()
```

To send time:

```
comms.update_time(time)
```

To plot data:

```
make your_data global   (your_data is the data you are plotting)
comms.update_plot('Label for first plot')
gui_err = comms.check_plot_status()
```

repeat for other plots

To wait at end of each loop iteration

```
gui_err = comms.check_run_status()
```

Incorporating GUI features into your own physics program

In cpanel.py
Change all references to physics to your script name

- or change your script name to physics

In this simple demo code, the only plot function supported is graphs.dscaler1

- plots Y vs linear function

In cpanel.on_plotstart callback:
For plots using graphs.dscaler1:

- change plot name to one in your dictionary, and the name sfield to match your data

For other kinds of plots, add them to graphs.py