TRAINING
GRAPHICAL PROGRAMMING INTERFACE (GPI)

Nick Zwart
2014sep11

gpilab.com

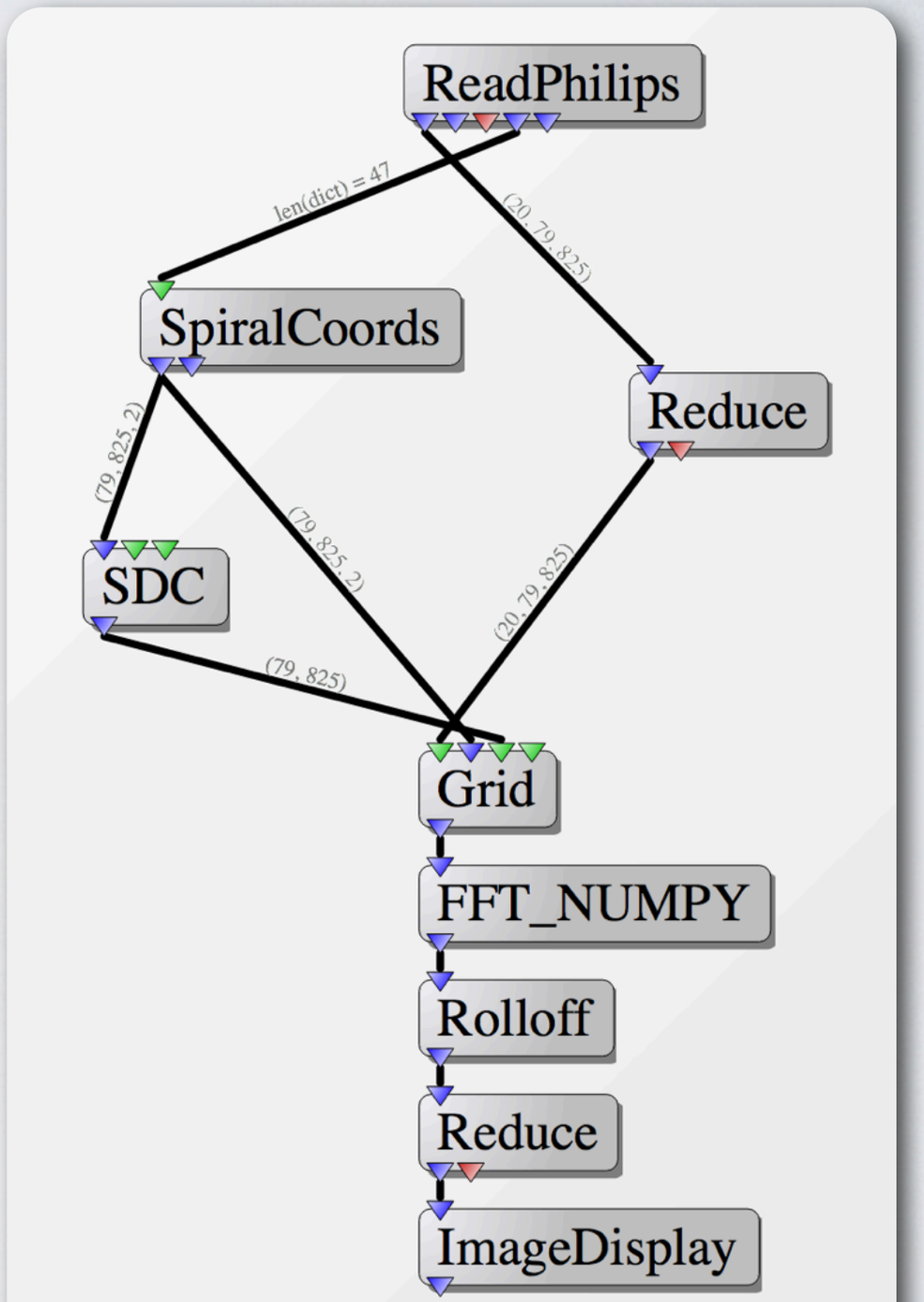
Keller Center
For Imaging Innovation

GPI

A Graphical Development Environment for Scientific Algorithms

INTRODUCTION

- Modular
  - Component & Algorithm
    - Comparison
      - Analysis
      - Investigation
  - Data Analysis & Visualization
- Reconstruction, Simulations, Pulse Sequence Development



Spiral Reconstruction

EXAMPLES

K-SPACE FILTER

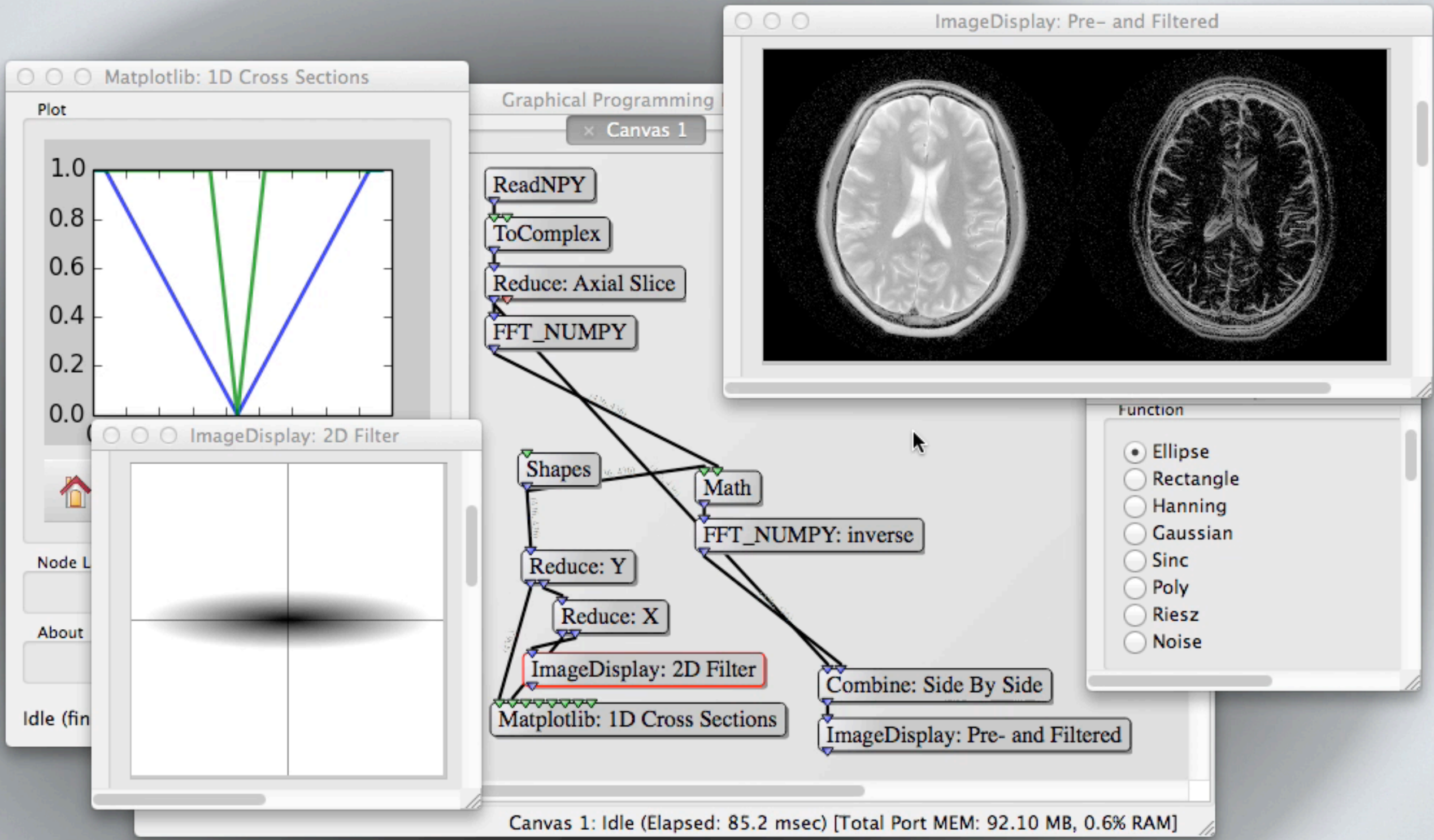
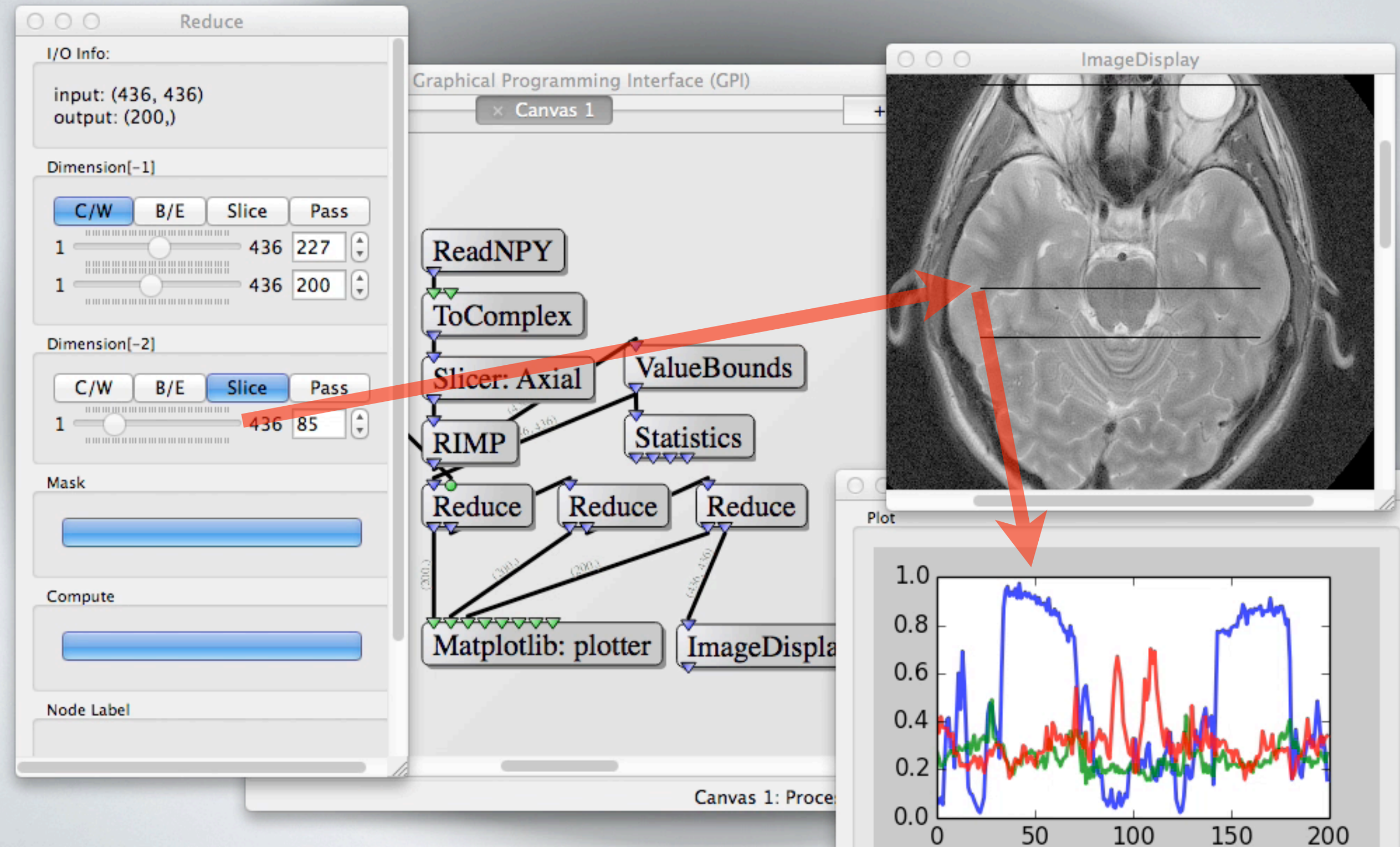
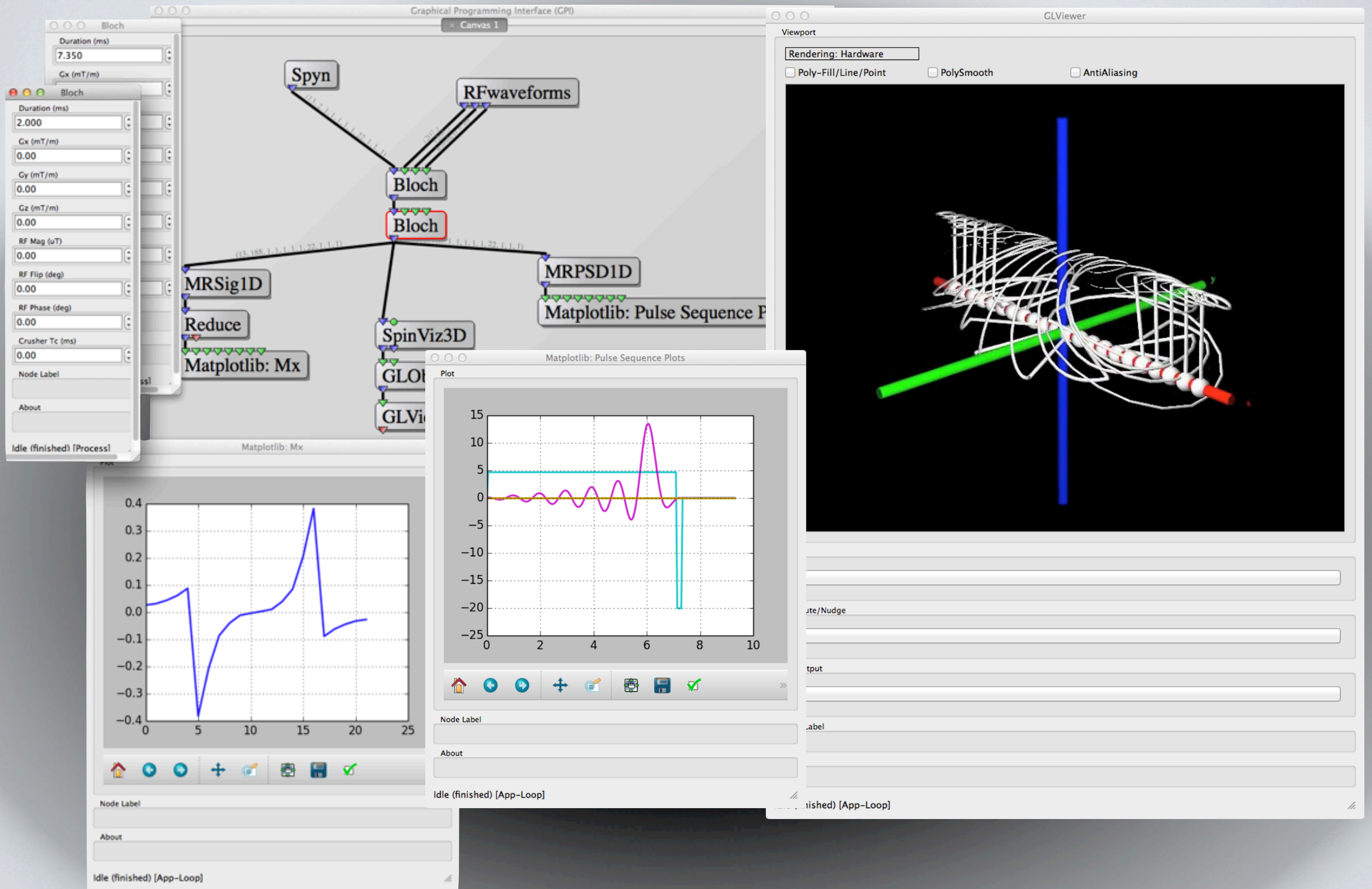


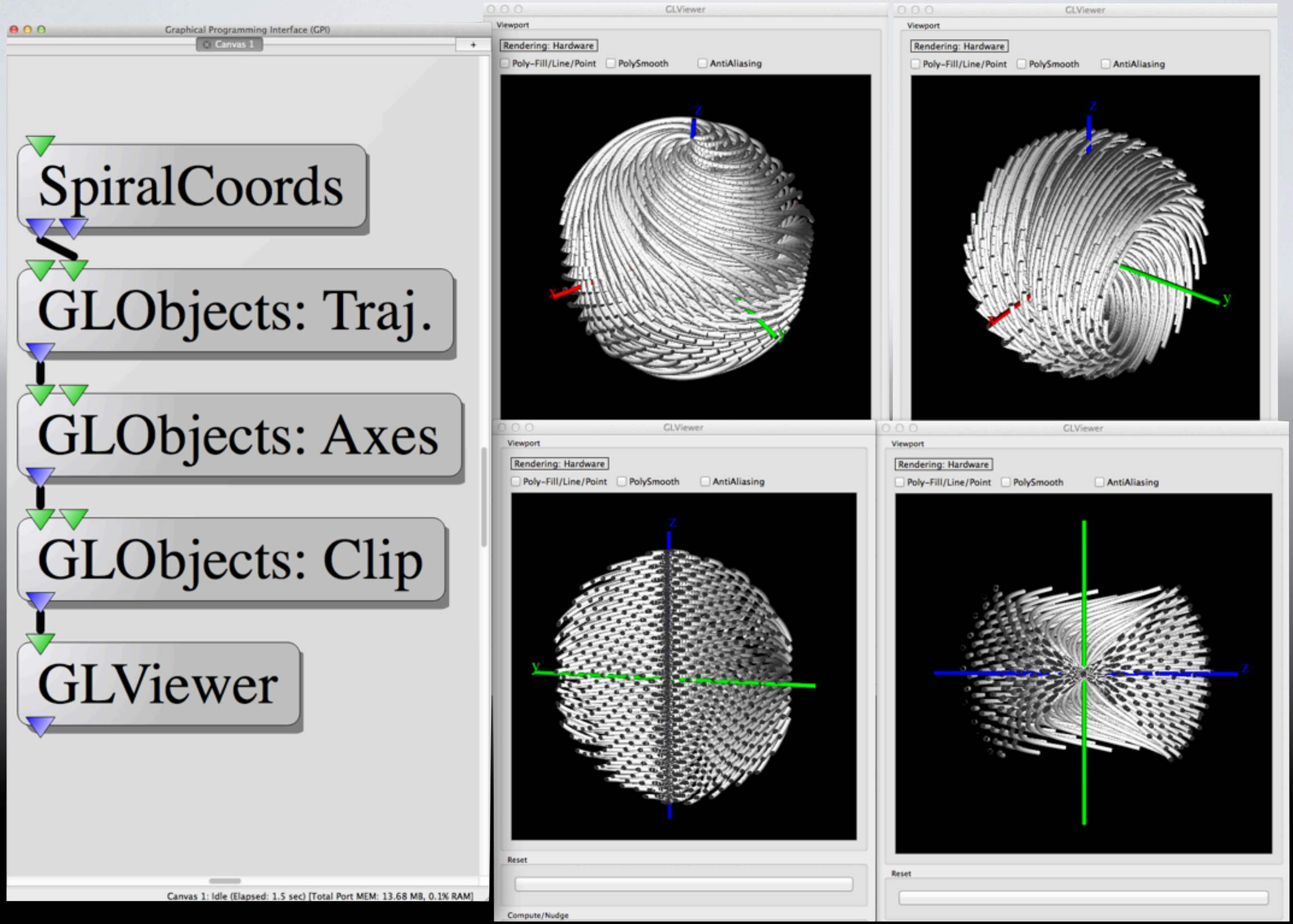
IMAGE CROSS SECTIONS



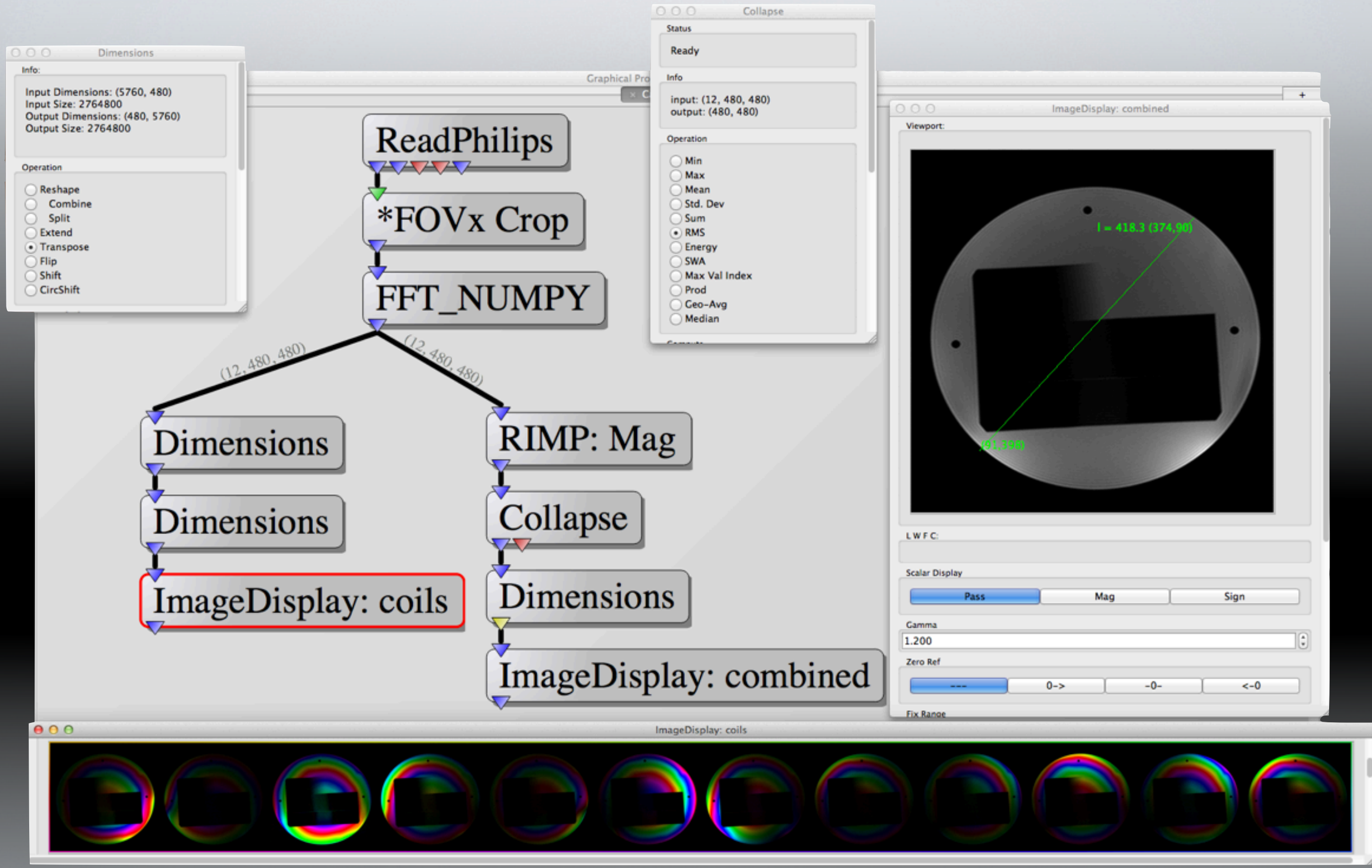
SPIN SIMULATION



3D GLVIEWER


Graphical Programming Interface (GPI)
Canvas 1
SpiralCoords
GLObjects: Traj.
GLObjects: Axes
GLObjects: Clip
GLViewer
Canvas 1: Idle (Elapsed: 1.5 sec) [Total Port MEM: 13.68 MB, 0.1% RAM]
GLViewer
Viewport
Rendering: Hardware
Poly-Fill/Line/Point
PolySmooth
AntiAliasing
Reset
Compute/Nudge

COIL COMBINATION



TRAINING MODULES

- **LESSON 1: Graphical Programming**
  - GUI Familiarity
  - Example Network Exercises
  - Algorithm Exercises
- **LESSON 2: Pure Python Nodes**
  - GPI Node Interface
  - Code Exercises
- **LESSON 3: C++ PyMods**
  - PyFI C++/Python Interface
  - Code Exercises

CHECKLIST

- Computer (50GB HDD, 8GB RAM, 4-Core Intel 64bit)
  - VMWare Fusion / VMPlayer ≥ 5.0
  - GPI Virtual Machine 0.2 (via Download or USB Drive)
- 3 Button Mouse (trackpad and mapped keys are not setup)
- NOTE: Upgrade Ubuntu at your own risk. GPI was only tested with the current revision of the installed libraries.

REFERENCE MATERIAL

- Node Guide
  - List of All Nodes with Summaries
- Node Developer's Guide (Lesson 2&3)
  - Code Guide for the Node and PyFI Interfaces
- Quick Start Guide
  - Keyboard and Mouse Key Mappings and Shortcuts
- Training Slides (this)

SOFTWARE

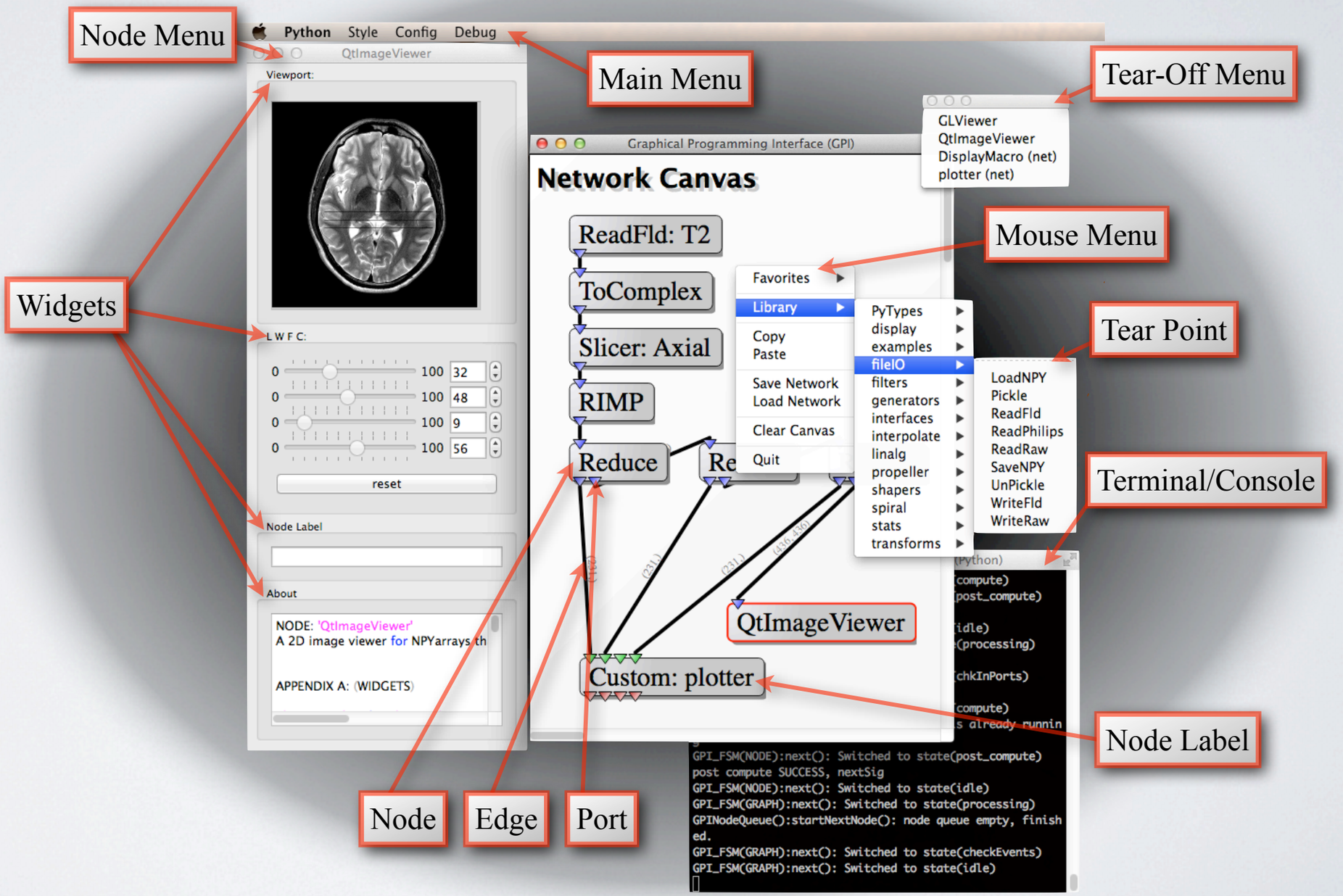
- Multi-Platform
  - Python Based
  - OSX & Linux & VM
- C++ PyMODs
  - Compiler (g++, XCode)
- Software Specs
  - README.pdf

LESSON 1 GOALS

- Familiarization with the GUI
  - Network Assembly
  - Widget Options
  - Canvas Behavior
- Graphical Programming / Network Debugging Skills
  - Inspecting Port Requirements
  - Visualizing Data at Points of Interest
  - Data Manipulation

GUI ELEMENTS

- QuickStart.pdf
- docs/
-

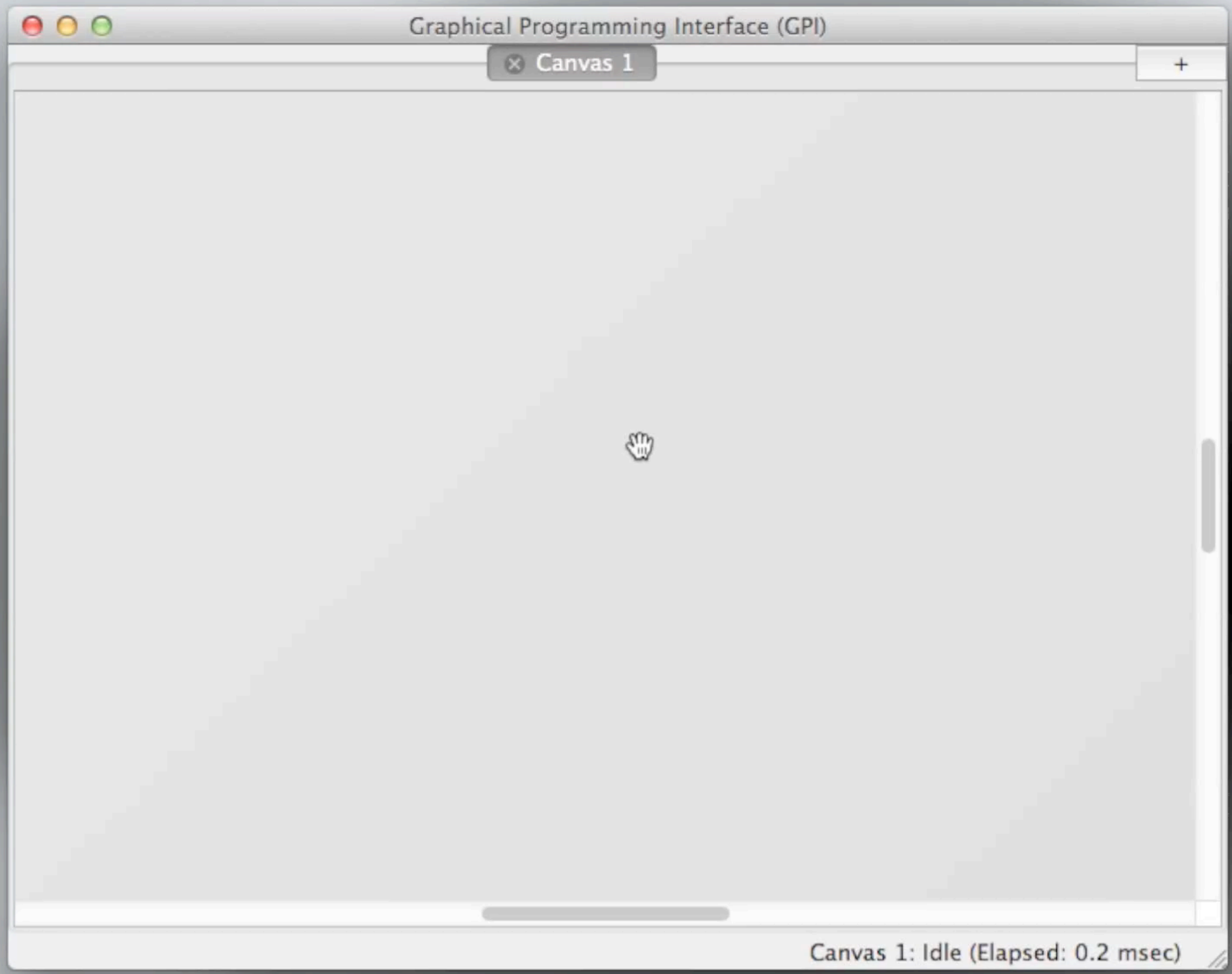


START EXERCISES

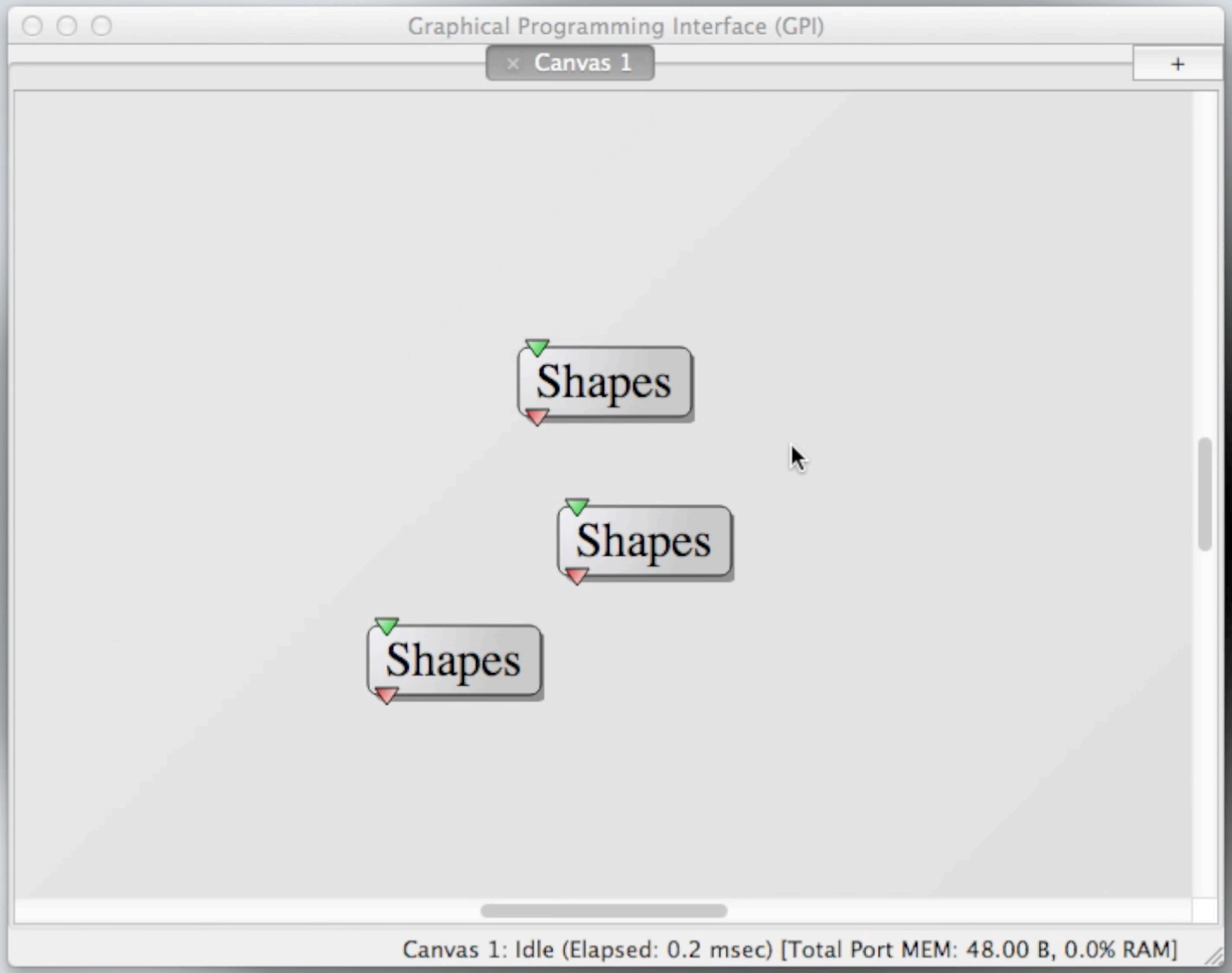
ELEMENTS TO COVER

- Network Canvas
  - Pan/Zoom
  - Multiples
- Nodes
  - Widgets
    - Label
    - About
  - Ports
- Edges
  - Hover
- Mouse Menu
  - Tear Points
  - Library
- Main Menu
  - Log Level
- Terminal/Console

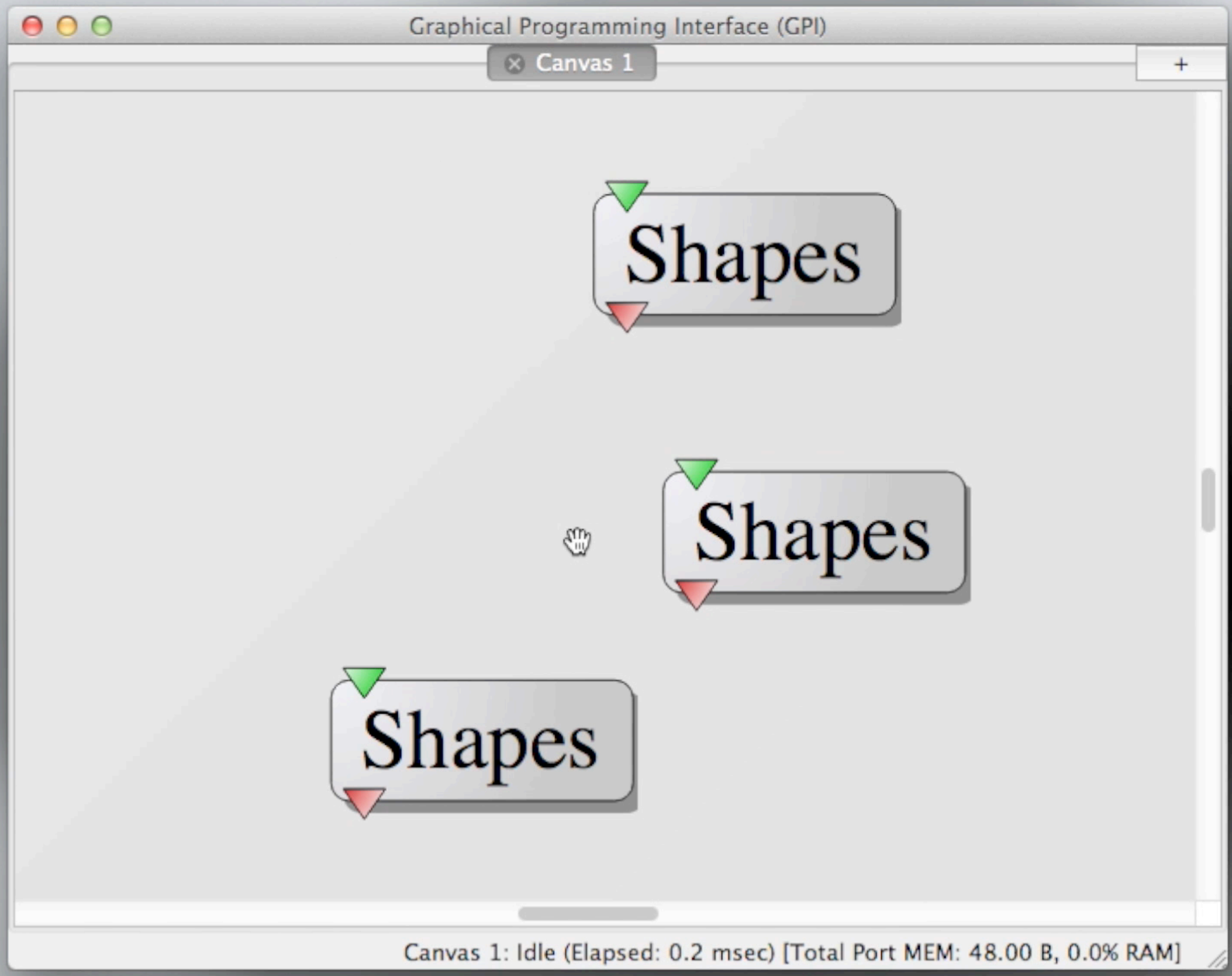
NODE INSTANTIATION



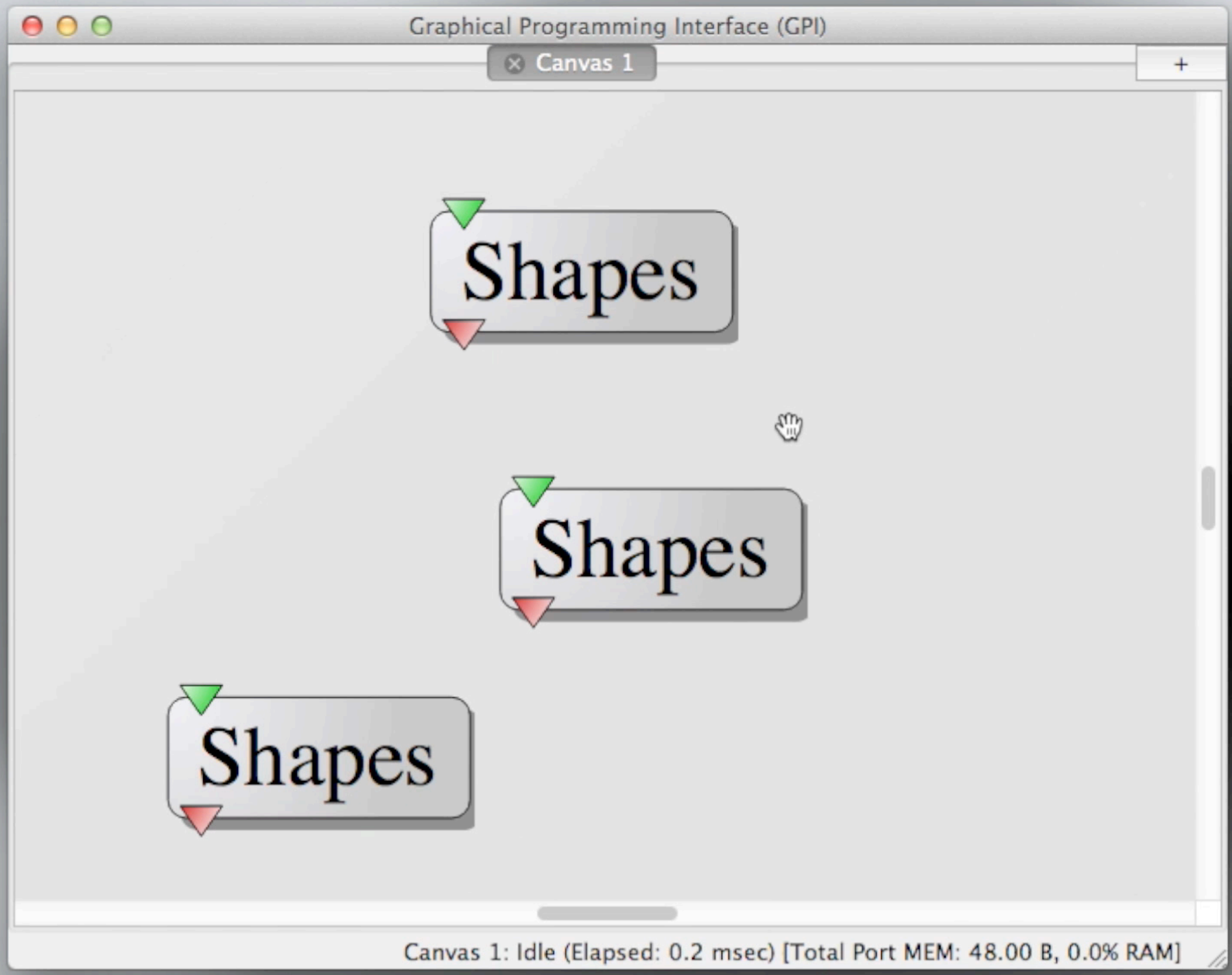
CANVAS ADJUSTMENT



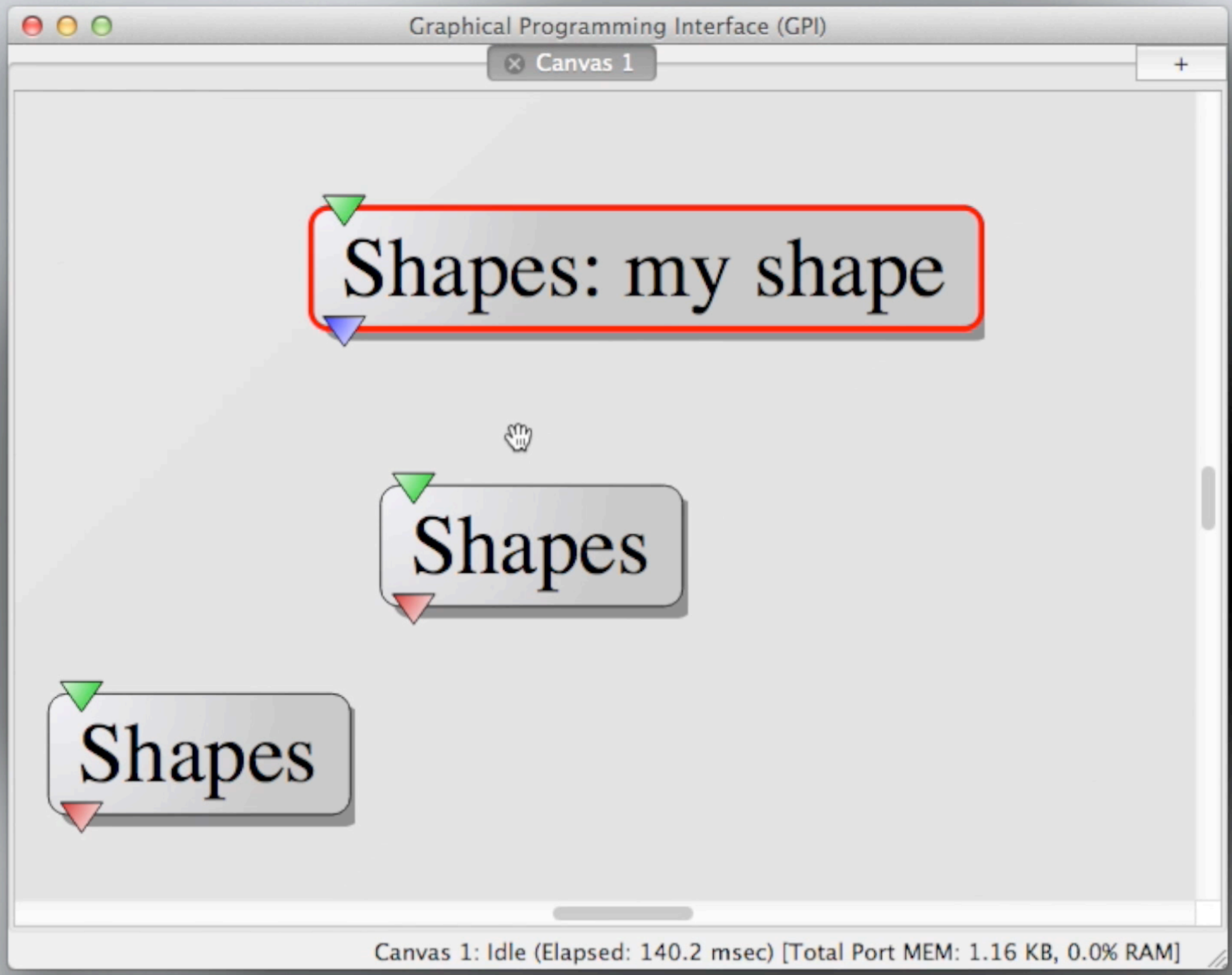
MULTIPLE CANVASES



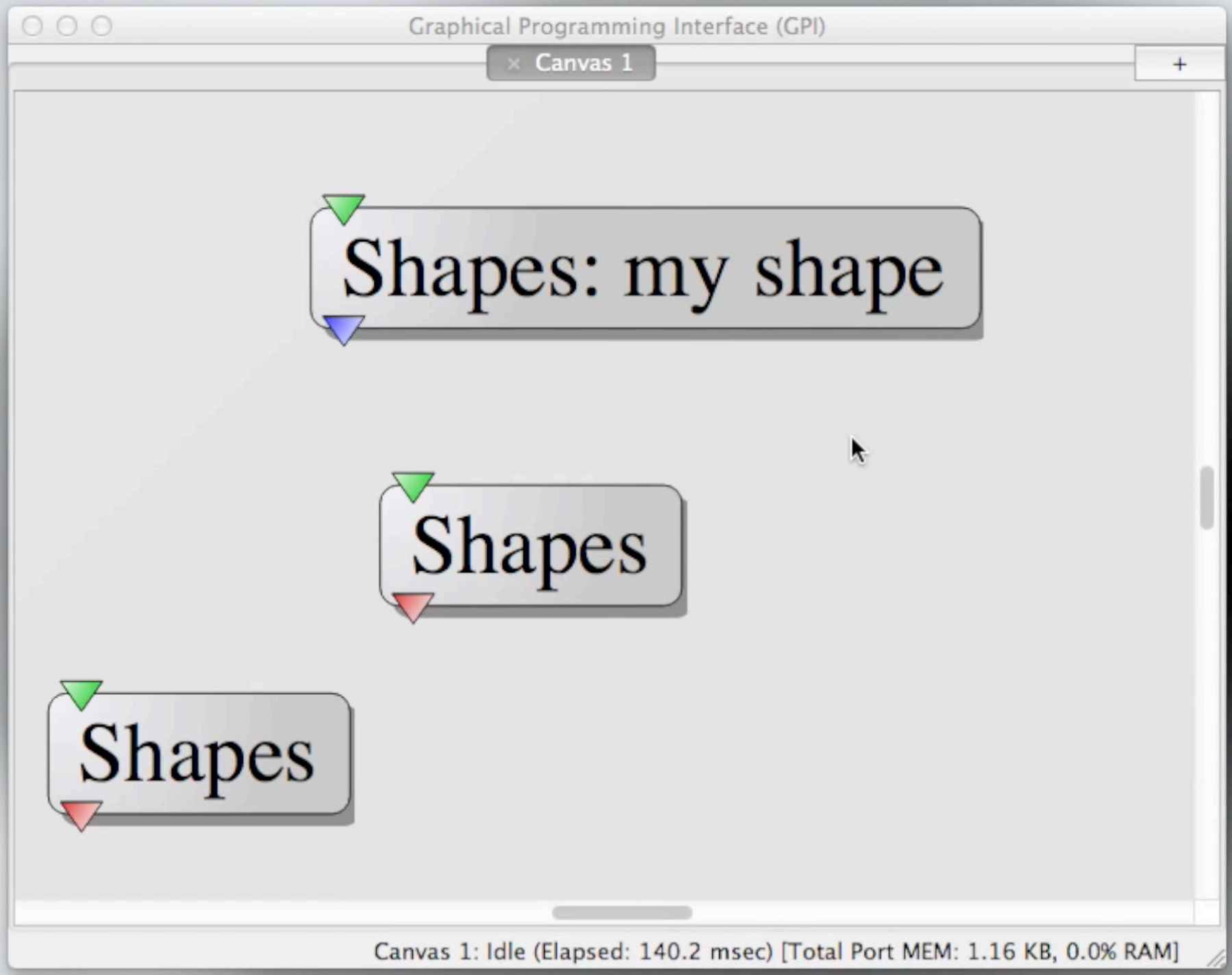
NODE MENU


Graphical Programming Interface (GPI)
Canvas 1
+
Shapes
Shapes
Shapes
Canvas 1: Idle (Elapsed: 0.2 msec) [Total Port MEM: 48.00 B, 0.0% RAM]

RUNTIME FEEDBACK

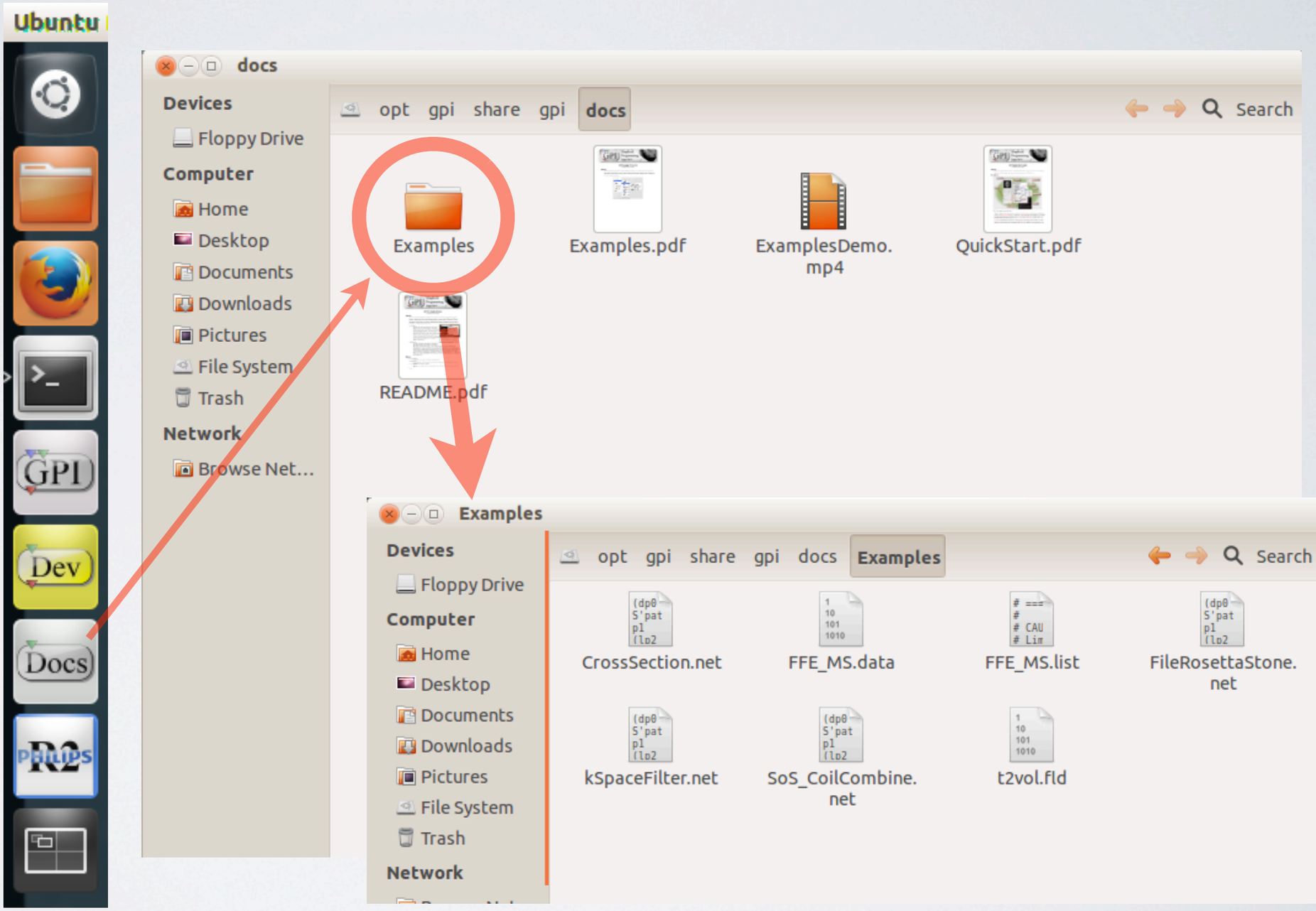


CONNECT & DELETE



EXAMPLE NETWORKS

- Examples.pdf
  - docs/



CROSS SECTION

- Reader - OS info, dimensions
- Vector Vs. Complex
- Reduce - Mask: Slice & Complement
- ImageDisplay - Zoom, Interpolate
- Plotter - Port Independence

Task:

Change to vertical cross sections and add a 4th cross section.

Goal:

Familiarization with important nodes such as Reduce, ImageDisplay and Matplotlib.

FILE ROSETTA STONE

- File Types
  - .data/.list, .raw/.lab/.sin, .par/.rec/.xml, .raw, .npy, .hdf5, .pickle, .csv, .cpx, .png, .jpg, .mat
  - Drag'n Drop
- Numpy is the Medium

Task:

Read in an array and save it to a different file format. Practice inspecting the requirements for a few file formats. Practice manipulating the data to fit desired format.

Goal:

Familiarization with implemented file formats.

K-SPACE FILTER

- Shapes
- Math
- FFT
- Combine
- ImageDisplay - RIMP Features

Task:

Make a low-pass filter.

Goal:

Recognizing points of interest in algorithm/network structure.

MACRO NODE PRIMER

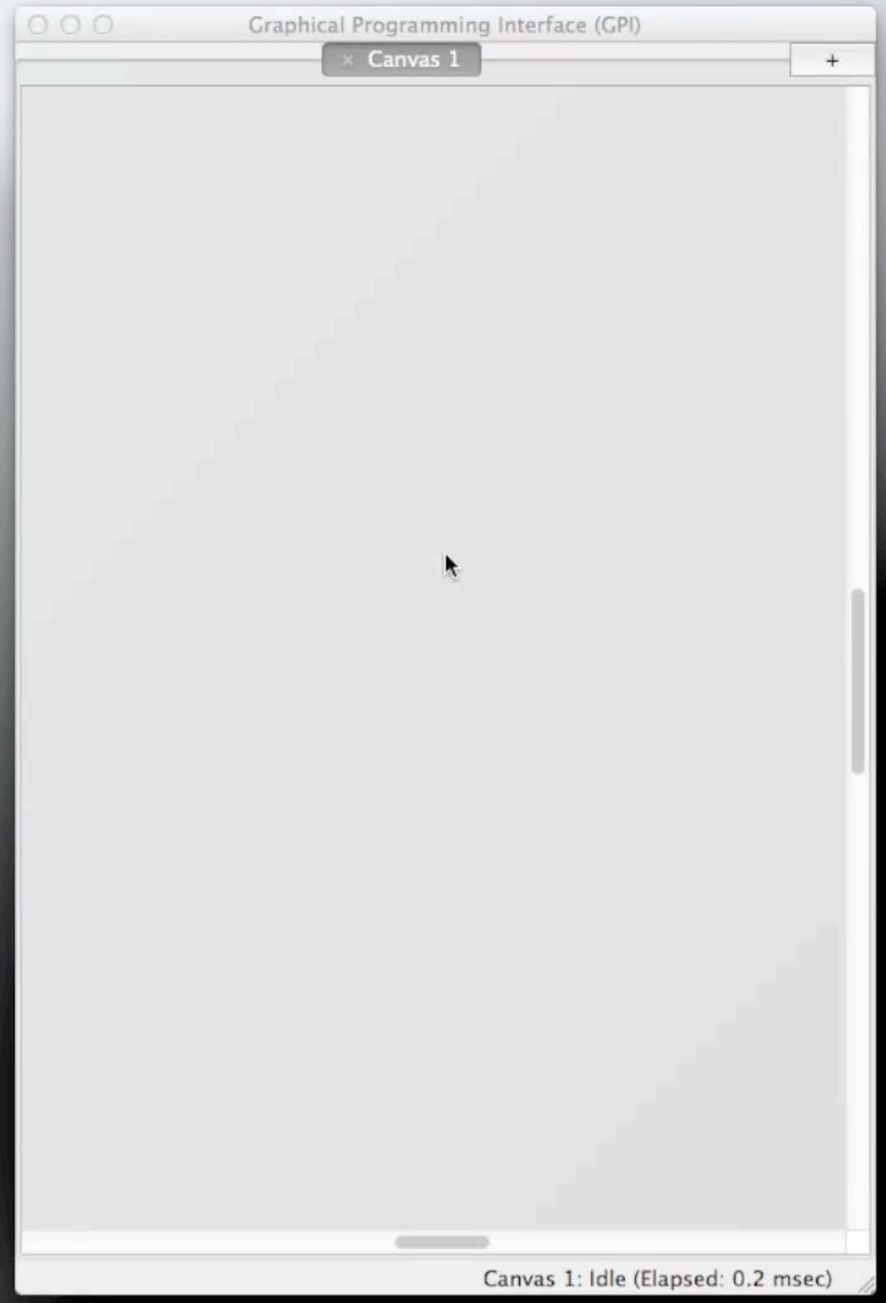
1. Mouse Menu
   > Macro Node

2. Connect Ports

3. Drag'n Drop Widgets
   OSX: alt+middle button
   Linux: middle button

4. Label

5. Double Click to
   Close and Open



GL MACRO NODE

- Macros
- Dimensions
- Custom - Combine Code
- GL Widget
- GL Objects - Lists

Task:

Rotate, translate, and zoom rendered objects, scene and lighting.

Add GL Objects to the Scene. Hint: the 'GLObjects' node appends new objects to an input list.

Goal:

Familiarization with GL Object construction and visualization.

SOS COIL COMBINE

- Macros
  - Widget Layout
- Collapse - RMS, Collapse All
- ImageDisplay - Options

Task:

Select different slices. Measure object diameter in pixels. Normalize the final image scale by the max value.

Goal:

Familiarization with important nodes such as RIMP, Collapse and ImageDisplay.

SPIRAL RECONSTRUCTION

- 2D or 3D Spiral (2D Example)
- .data/.list/.txt
- Non Cartesian Recon
  - Coordinates
  - Density Correct (SDC)
  - Gridding & Rolloff

Task:

Scroll through reconstructed stack of images. Change the network to reconstruct only one slice at a time. Plot a single spiral interleave.

Goal:

Familiarity with non-cartesian reconstruction nodes and Widget parameter 'Dims per Set'.

RECONSTRUCTION TROUBLESHOOTING

- Common Probing Nodes
  - Reduce
    - Matplotlib
    - ImageDisplay

Task:

Determine the cause of the image quality issue. Implement a simple fix for a couple of images.

Goal:

Familiarity with working backwards through an algorithm.

SPIN SIMULATOR

- Spyn - Spin Generator
- Bloch - Numeric Solver
- 1D & 3D Visualization
- 3D Movies
  - Save ARGB Frames as '.raw'
  - ImageJ - Stack > Movie
  - Examples/SpinSim.m4v

Task:

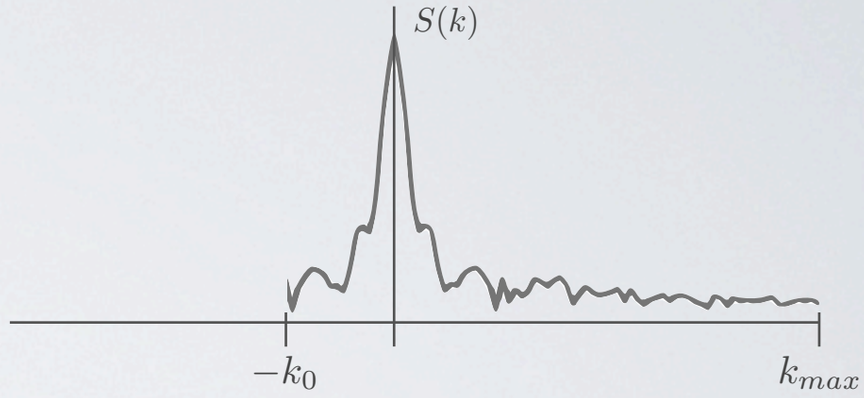
Apply a symmetric RF pulse and observe the result. Use AutoNum to loop through the time points.

Goal:

Familiarity with spin simulation nodes, GLViewer and AutoNum.

ALGORITHM EXERCISE
HOMODYNE RECONSTRUCTION

- Zeropad



$$k_0 = 0.2k_{max}$$

- Highpass Filter

$$H(k) = \begin{cases} 0 & k < -k_0 \\ ? & -k_0 \leq k < k_0 \\ 1 & k \geq k_0 \end{cases}$$

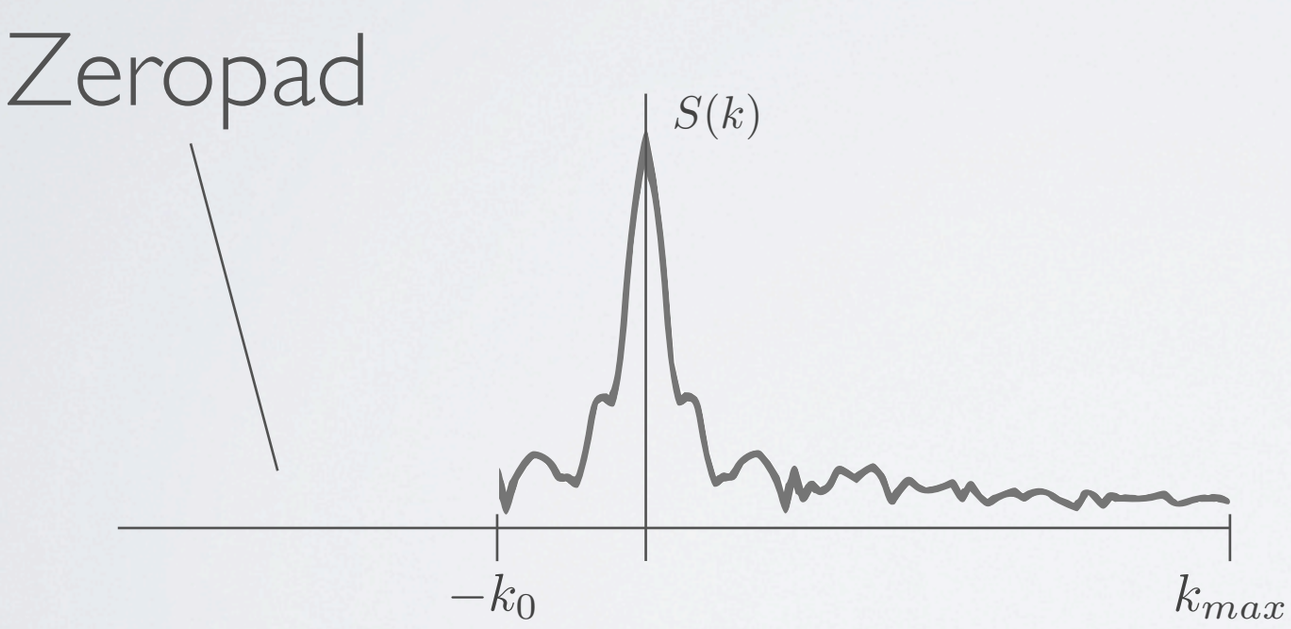
- Assume Real Valued

$$\hat{S}(k) = S(k)H(k)$$

$$\hat{s}(x) = Re\left[\mathcal{F}(\hat{S}(k))\right]$$

ALGORITHM EXERCISE
HOMODYNE RECONSTRUCTION

Step 1: Zeropad



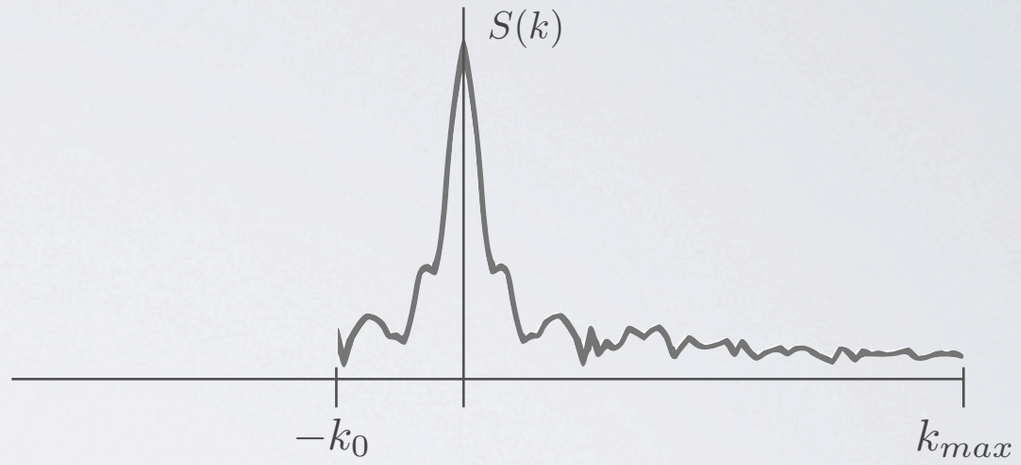
$$k_0 = 0.2k_{max}$$

Nodes:

- ReadNPY
  - (Examples/partialEcho.npy)
- Shapes
- ToComplex
- Combine

ALGORITHM EXERCISE
HOMODYNE RECONSTRUCTION

Step 2: Generate Filter



Stop Band
Transition
Pass Band

Filter

$$H(k) = \begin{cases} 0 & k < -k_0 \\ ? & -k_0 \leq k < k_0 \\ 1 & k \geq k_0 \end{cases}$$

Nodes:
- Shapes
- Dimensions

ALGORITHM EXERCISE
HOMODYNE RECONSTRUCTION

Step 3: Apply Filter

Sampled Data

Filter

$$\hat{S}(k) = S(k)H(k)$$

Modulated Data

Nodes:

Math

ALGORITHM EXERCISE
HOMODYNE RECONSTRUCTION

Step 4: Transform and Realize

Real

Modulated k-Space

$$\hat{s}(x) = Re\left[\mathcal{F}(\hat{S}(k))\right]$$

Final Image

Fourier Transform

Nodes:

- FFTW
- RIMP

ALGORITHM EXERCISE

COIL COMBINE

$$\hat{C}_n = C_n \otimes K$$

- Estimate B_1^- for each coil.

$$B_n = \frac{\hat{C}_n}{\sqrt{\sum^n \hat{C}_n^2}}$$

- Remove B_1^- sensitivity profile from coil images.

$$\hat{I}_n = C_n B_n^*$$

- Phase preserving coil combine.

$$I = \sum^n \hat{I}_n$$

ALGORITHM EXERCISE

COIL COMBINE

Step 1: Blur Coil Images

Blurred Images

Images with Coil Modulation

$$\hat{C}_n = C_n \otimes K$$

Blur Kernel

Nodes:

- ReadNPY (Examples/MultiCoil.npy)
- Math
- Shapes
- FFT_NUMPY

ALGORITHM EXERCISE

COIL COMBINE

Step 2: Remove Image Modulus.

Complex Coil Sensitivity

Blurred Images

$$B_n = \frac{\hat{C}_n}{\sqrt{\sum^n \hat{C}_n^2}}$$

Nodes:

- RIMP
- Collapse
- Dimensions
- ValueBounds
- Math

ALGORITHM EXERCISE

COIL COMBINE

Step 3: Remove B_1^- phase from coil images.

Images w/o Coil Phase

Images with Coil Modulation

$$\hat{I}_n = C_n B_n^*$$

Complex Coil Sensitivity

Conjugate

Nodes:

Math

ALGORITHM EXERCISE

COIL COMBINE

Step 4: Phase preserving coil combine.

Final Image

Images w/o Coil Phase

$$I = \sum^{n} \hat{I}_n$$

Nodes:

Collapse

ALGORITHM EXERCISE
EDDY CURRENT CORRECTION

- Estimate linear phase
  - By Estimating Δ_k
  - By Estimating $\frac{\phi}{x}$

$$\frac{\phi}{x} = \frac{2\pi\Delta_k}{N}$$

- Remove Phase
  - Image Space or k-Space

ALGORITHM EXERCISE

EDDY CURRENT CORRECTION

Method #1

Nodes:

- ReadMatlab
  (Examples/eddycurrents.mat)

(phase estimation)

- Collapse
- FFT
- Float_Math
- RIMP

(phase removal)

- Shapes
- ToComplex
- Math

Method #2

Nodes:

- ReadMatlab
  (Examples/eddycurrents.mat)

(phase estimation)

- Collapse
- Custom (numpy to float)
- RIMP
- Reduce
- Regression

(phase removal)

- Shapes
- ToComplex
- Math

phase per pixel

k-shift

$$\frac{\phi}{x} = \frac{2\pi\Delta_k}{N}$$

matrix size

TRY READING **YOUR** DATA

- Use ReadPhilips Node
- Visualize
- Reconstruct (or partially reconstruct)

TRAINING MODULES

- **~~LESSON 1: Graphical Programming~~**
  - ~~GUI Familiarity~~
  - ~~Example Network Exercises~~
  - ~~Algorithm Exercises~~
- **LESSON 2: Pure Python Nodes**
  - GPI Node Interface
  - Code Exercises
- **LESSON 3: C++ PyMods**
  - PyFI C++/Python Interface
  - Code Exercises

GPI

A Graphical Development Environment for Scientific Algorithms

NODE DEVELOPMENT

- **Core Nodes**
  - Basic Subset
  - Common Utility
- **Node Development**
  - Additional Functionality
  - Simple Code Interface
  - Application Specific Prototyping
- **Standard Interface** (Widgets & Ports)
  - Team Development / Collaboration

LESSON 2 GOALS

- Introduce the GPI Node Interface
- Python Workflow
  - Directory Structure
  - Node Update Procedure
- GPI_PROCESS

DIRECTORY STRUCTURE

Format:

`<library>/<sub-lib>/GPI/<mynode>_GPI.py`

Library Name in Mouse Menu

Sub-Library Name in Mouse Menu

Node Name on Canvas

Required Suffix

Examples:

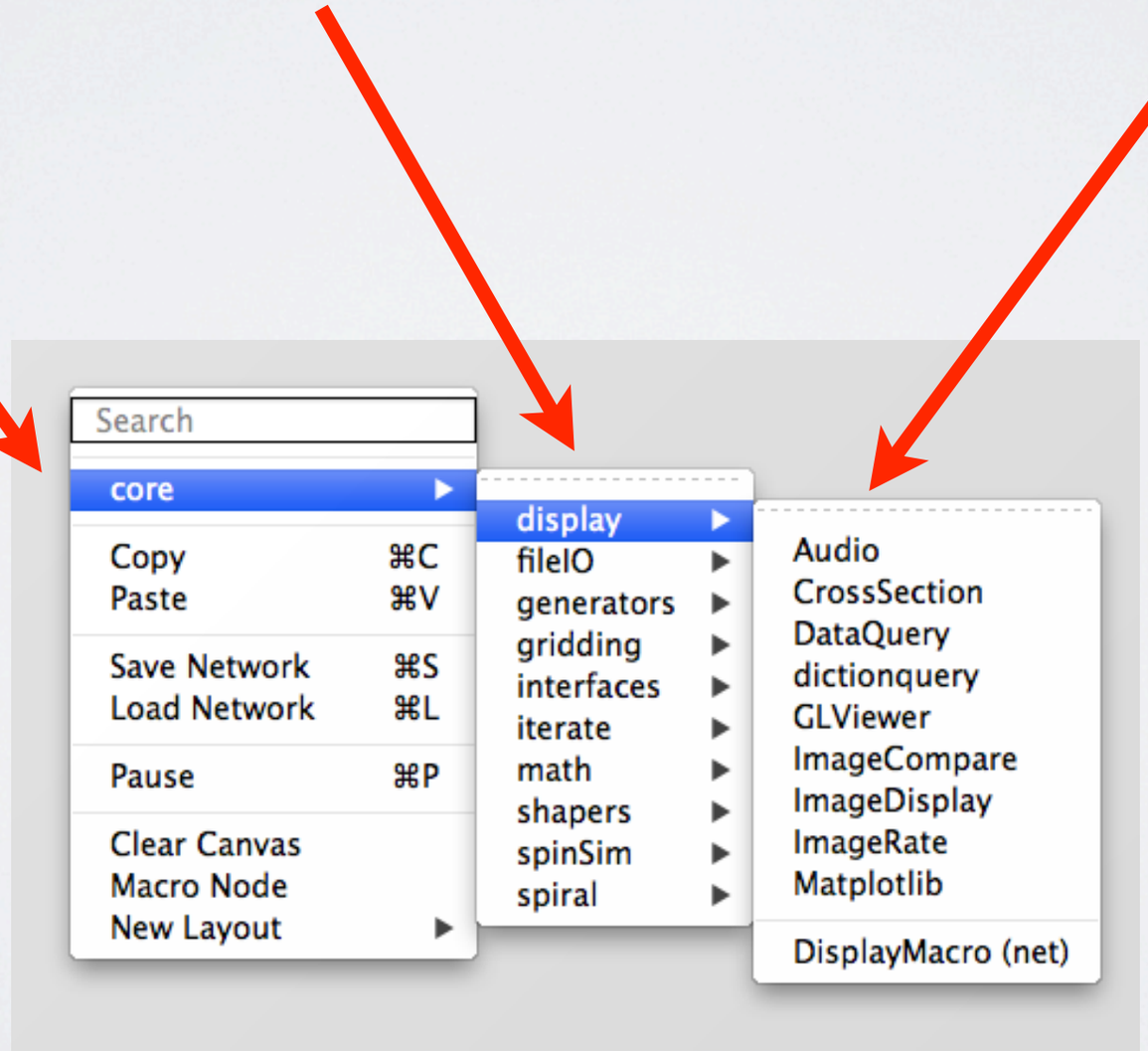
`core/math/GPI/RIMP_GPI.py`

`~/gpi/exercises/ex1/GPI/e1_widgets_GPI.py`

DIRECTORY STRUCTURE

Format:

```
<library>/<sub-lib>/GPI/<mynode>_GPI.py
```



ENVIRONMENT VARIABLES

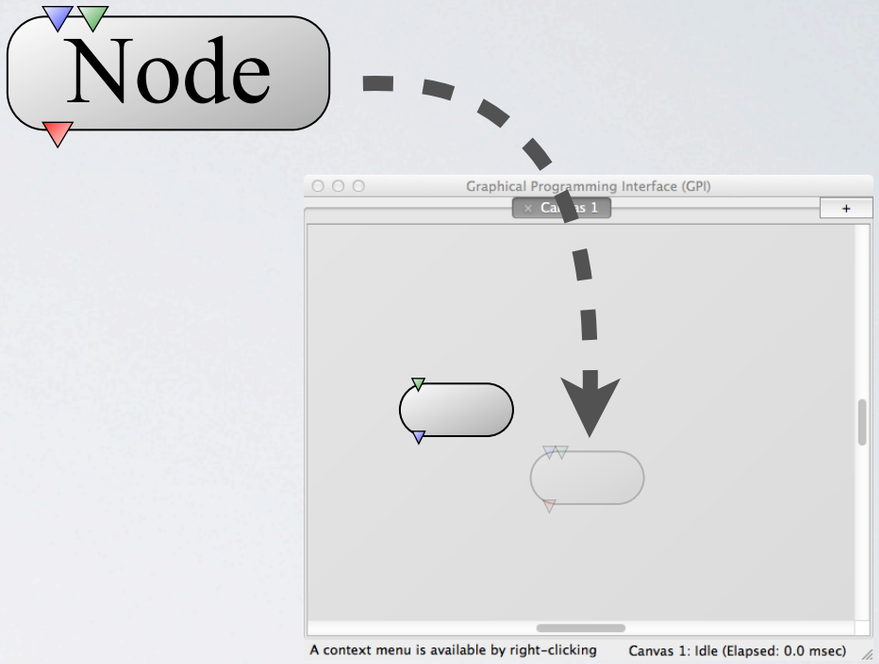
- ~/.bashrc
  - ‘gpi’ & ‘gpi_make’ commands
    - PATH = “/opt/gpi/bin:$PATH”
- Code Editor
  - Linux: “EDITOR” environment variable
    - full path to desired code editor (use a GUI editor)
  - OSX: use file associations

PYTHON NOTES

- Whitespace Sensitive
  - Indentation is 4 Spaces
    - Defines Scope for: **`if, for, while, def,`** etc...
    - gedit Tab-width
- Use the '**`import`**' Statement to Include Libraries
  - **`numpy`** - Python's Numeric Array Library

GPI NODE INTERFACE

Instantiation



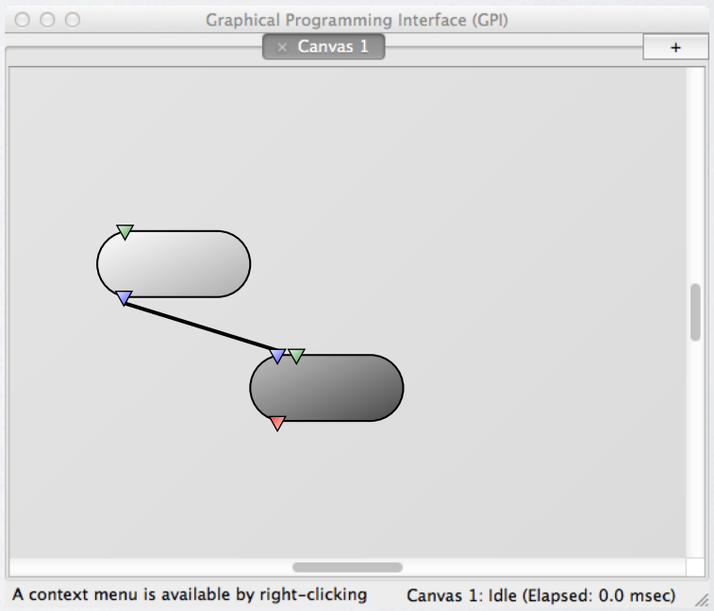
`initUI()`

- Widgets
- Ports

Execution

`compute()`

- Algorithm



START EXERCISES

→ exercises/ex1/GPI/e1_widgets_GPI.py

ex1/GPI/e1_widgets_GPI.py

- Drill Down the Library Menu **or** Search
- Ctrl - Right Mouse Click (Open the Code)
- NodeDevGuide.pdf (Widget Attributes)
- Terminal Window
  - `gpi_make e1_widgets_GPI.py`
- 'about' widget
- initUI() & compute()

ex1/GPI/e2_ports_GPI.py

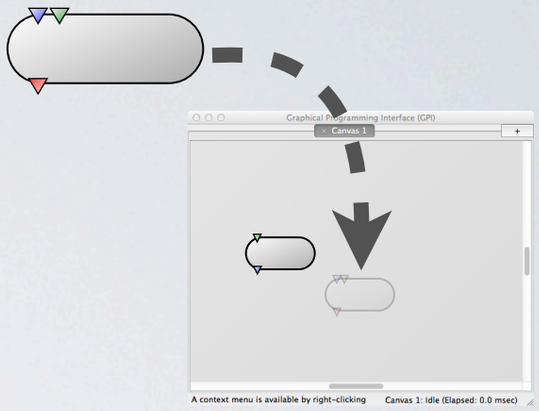
- Navigate to the Code (Use Drag & Drop)
- Port Specifications
  - NodeDevGuide.pdf (Type Attributes)
    - `import numpy`
- Shapes & Statistics Nodes
  - Check InPort Enforcement
  - Observe OutPort Errors (for Wrong Shape)
  - Check Shape with Statistics, Edge, Port-Hover

GPI NODE INTERFACE

Instantiation

`initUI()`

- Widgets
- Ports



Validation

`validate()`

- Widgets
- Ports

Execution

`compute()`

- Algorithm

ex1/GPI/e3_validate_GPI.py

- validate()
- Return Codes
  - Use Shapes as Input
  - Practice Resolving Warn & Error States
- Update Logger Message Based on Code Reqs

ex1/GPI/e4_logger_GPI.py

- Main Menu > Debug > Log Level
  - Test Against Chosen Level in Node
    - Terminal Window Output
      - Time, Line #, Code, etc...
    - Messages Demonstrate Appropriate Info

ex1/GPI/e5_events_GPI.py

- Observe Event Types
  - Widget, Port, Init

ex1/GPI/e6_import_GPI.py

- Pure-PyMODs
  - import Dev Modules in compute()
- Reinforce Library Directory Structure
- Identify Kernel Compute Code
  - Relocate to Module

YOUR NODE PROJECT

Take the rest of the time to work on a node from the challenge list or your own algorithm.

TRAINING MODULES

- ~~**LESSON 1: Graphical Programming**~~
  - ~~GUI Familiarity~~
  - ~~Example Network Exercises~~
  - ~~Algorithm Exercises~~
- ~~**LESSON 2: Pure Python Nodes**~~
  - ~~GPI Node Interface~~
  - ~~Code Exercises~~
- **LESSON 3: C++ PyMods**
  - PyFI C++/Python Interface
  - Code Exercises

DIRECTORY STRUCTURE

Node Format:

```
<library>/<sub-lib>/GPI/<mynode>_GPI.py
```

Library Name in Mouse Menu

Sub-Library Name in Mouse Menu

Node Name on Canvas

Required Suffix

PyMOD Format:

```
<library>/<sub-lib>/<mymod>_PyMOD.cpp
```

Library Name in Mouse Menu

Sub-Library Name in Mouse Menu

import Name

Required Suffix

```
import library.sublib.mymod
```

DIRECTORY STRUCTURE

Python Package:

```
<library>/__init__.py
<library>/<sub-lib>/__init__.py
```

```
import library.sublib.mymod
```

PYTHON FUNCTION INTERFACE (PYFI)

- Translates Numpy (.py) to PyFI Arrays (.cpp)
  - Wraps Original Data Segment (No Copying)
- Types
  - double, long (int64_t), std::string
  - PyFI Array Types
    - float, double, int32_t, int64_t, complex<float>, complex<double>

START EXERCISES

→ exercises/ex2/GPI/e1_hello_GPI.py

ex2/GPI/e1_hello_GPI.py

- Terminal Window
  - `gpi_make e2_module_PyMOD.cpp (or e2_module)`
  - `ls -l`
  - <module>.so Binary File
- Module Re-Uptake
- Look at PyFI Code

ex2/GPI/e2_PosArgs_GPI.py

- Positional Arguments
  - Pointers
  - Code: Top-Bottom
  - Py-Function: Left-Right
- Multiple Outputs are Returned in a Tuple

ex2/GPI/e3_arrays_GPI.py

- PyFI Arrays
  - Declaration & Initialization
  - Operations
  - Destructor
  - coutv()
  - Copy Mode Output

ex2/GPI/e4_ArrayBounds_GPI.py

- PyFI Arrays
- Declaration & Initialization
- Bounds
- Pre-Allocate (PYFI_SETOUTPUT_ALLOC())
- Observe Output Order

ex2/GPI/e5_KeywordArgs_GPI.py

- Default Arguments
  - Array Defaults
- Positional First, then Keyword Args

ex2/GPI/e6_errors_GPI.py

- Python Exceptions <> PyFI Errors
  - PYFI_ERROR()
- Use Python Docs for Exception Handling

ex2/GPI/e7_kernels_GPI.py

- Kernel Code
  - `<algorithm-name>_kernel.cpp`
- Portability Guidelines
- Task:
  - Pass dimensions from python (using a numpy array).
  - Generate a new PyFI Array inside the kernel function.
  - Pass the newly generated array back to python.

ex2/GPI/e8_threads_GPI.py

- Observe Threaded `stdout` in the Terminal

- Test w/o Threading Interface (e.g. non-pThread Portability)

YOUR NODE PROJECT

Take the rest of the time to work on a node from the challenge list or your own algorithm.

DISCUSSION

- Questions
- Future Features
- Communication & Feedback
  - Site Contact

APPENDIX

- Extra notes about GPI

DOWNLOADS

http://goo.gl/6MuJUz

WEBSITE

gp1lab.com

NODE DEVELOPER CHALLENGE

Pick a node description from the list,
claim your node by signing up at the
front, write it before the end of class
and it will be included in the GPI
core with
your name as author.

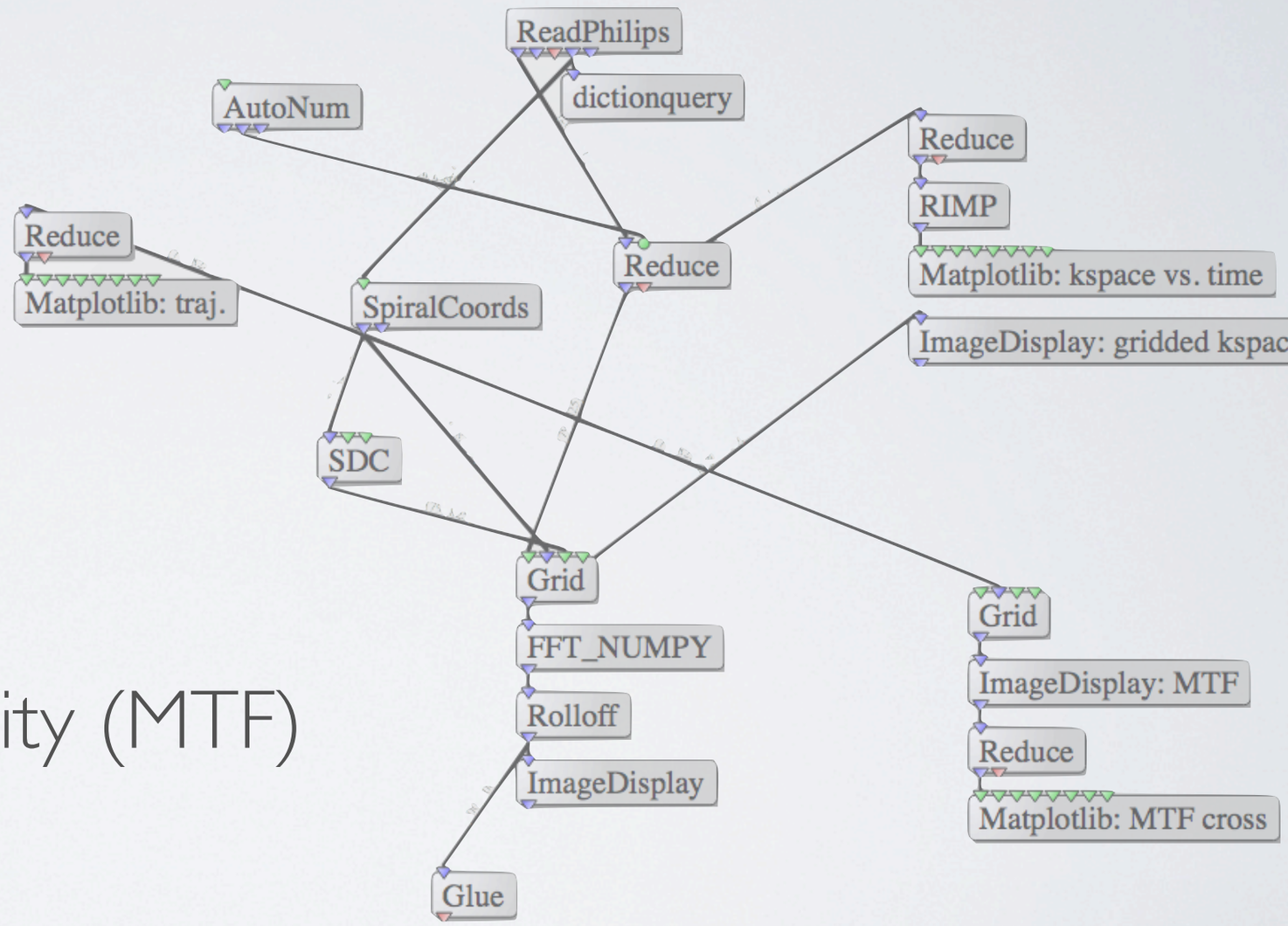
Or follow along with your own node idea.

KCII LAB EXAMPLES

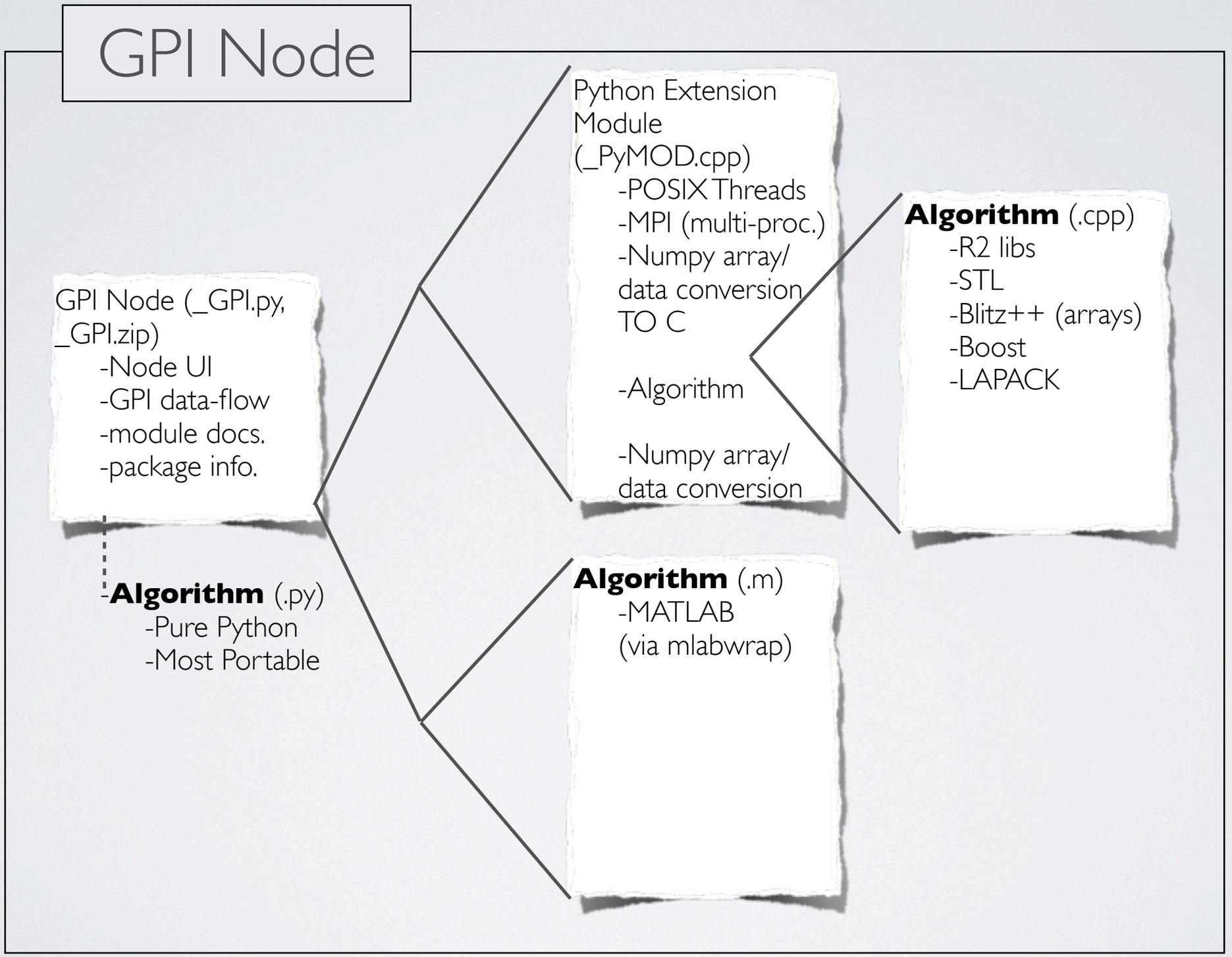
- Nick Zwart - Spiral CG Deblur (iterative)
- Dinghui Wang - Spiral Fat-Water (iterative)
- Zhiqiang Li - Spiral TSE, ASL
- Ryan Robison - System Characterization
- Yuchou Chang - PROPELLER
- Mike Schar - PROPELLER
- Jim Pipe - Spin Simulation

SPIRAL RECONSTRUCTION EXAMPLE

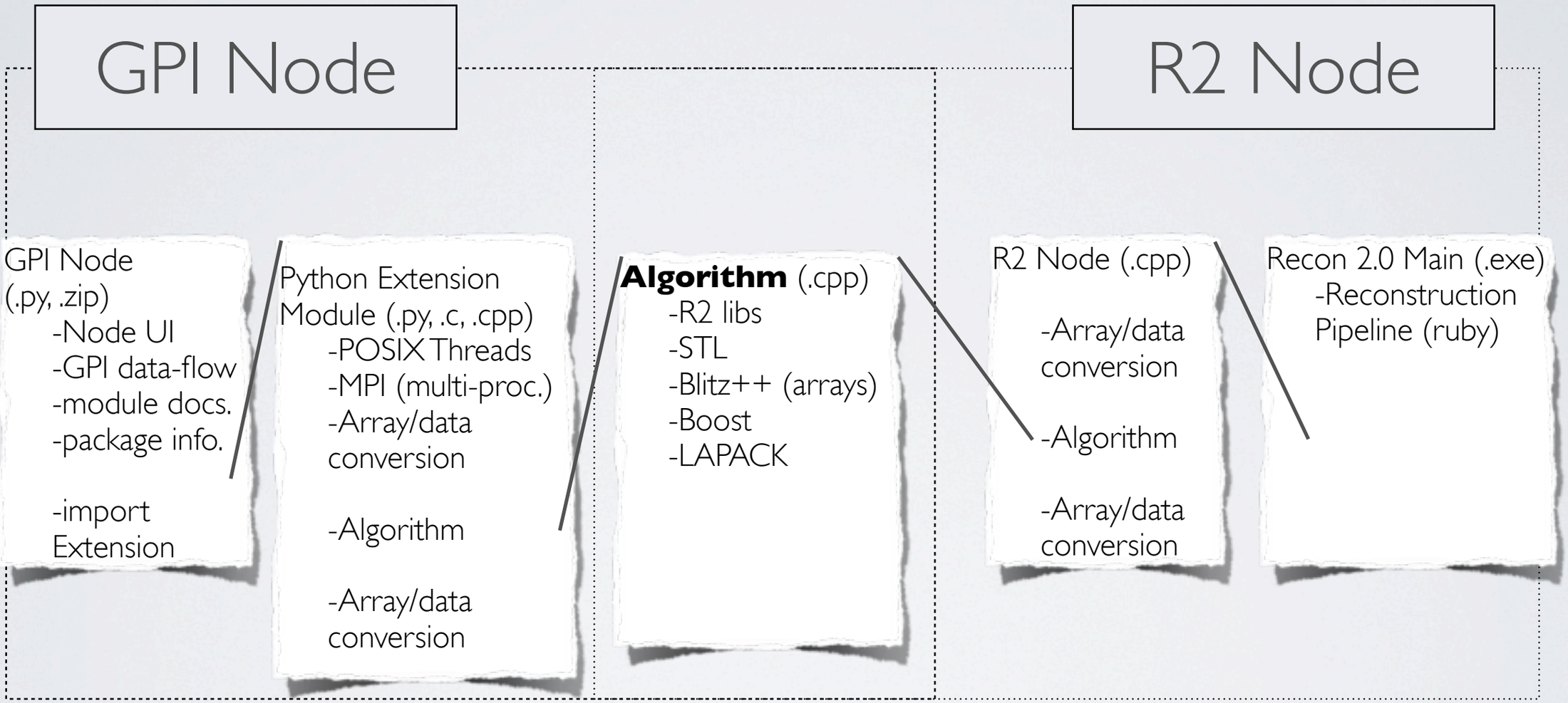
- k-Space Data
  - 2D & Plot Views
- Trajectory Coordinates
  - Plotted
  - Gridded Sample Density (MTF)
    - Cross-section
- Single Slice Recon (Accumulated)



CODE STRUCTURE

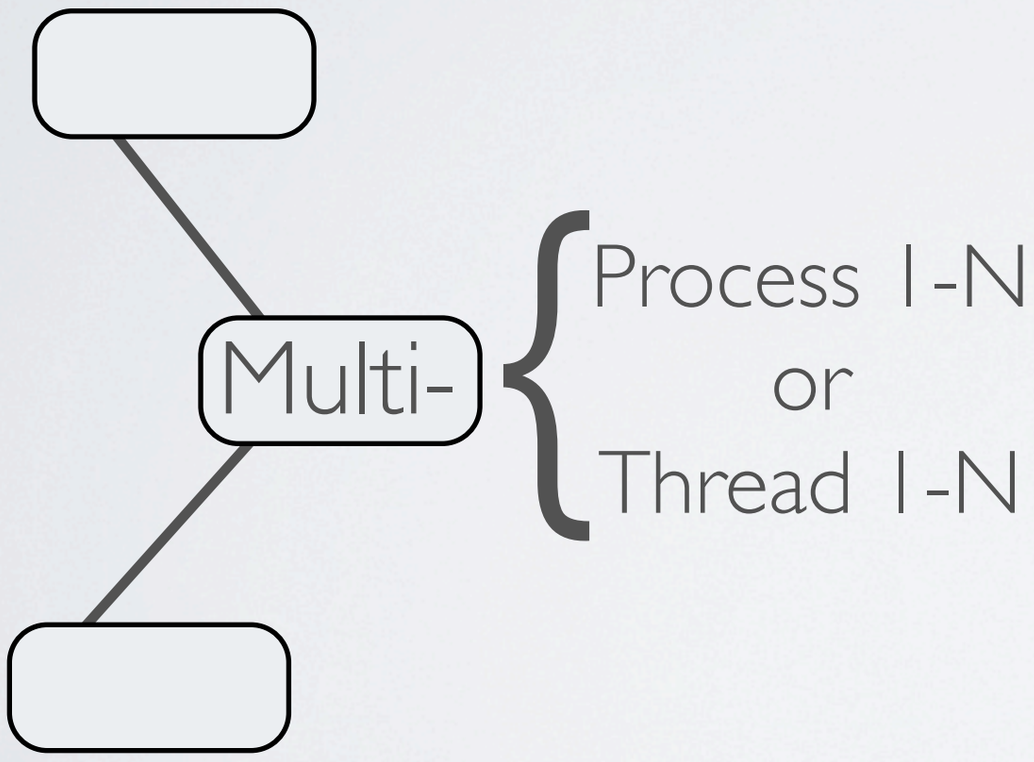


CODE STRUCTURE



GPI VS. R2 TOPOLOGY

GPI Multi-Processing
POSIX, TBB, MPI...



R2 Multi-Processing
TBB