Forge

A high-performance visualization library

Overview

- Background and motivation
- What does Forge do?
- Forge Workflow
- Examples
- Conclusion

Popular Plotting Libraries

C/C++

- Visualization Toolkit (VTK, Kitware)
- QCustomPlot
- QtPlot (QT 5.6-ish)

R

- Plotly (interactive)
- rgl (uses OpenGL)

Python

- Bokeh (web-based)
- Glumpy (uses OpenGL)
- Matplotlib
- PyQtGraph
- Galry (2D GPU-friendly)

Many one-off solutions

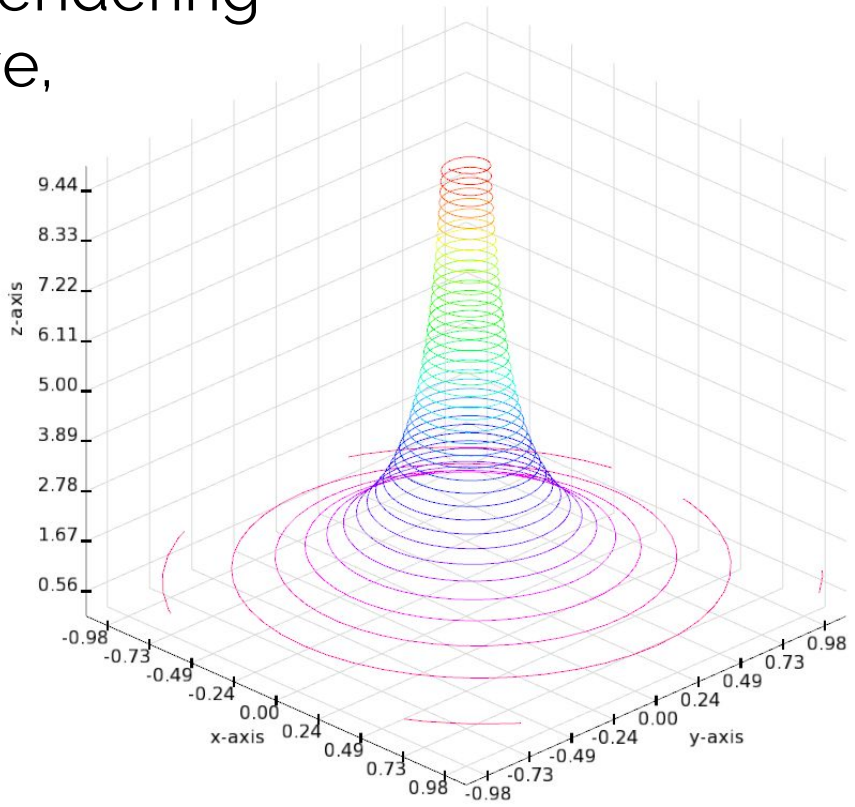
See https://github.com/fasouto/awesome-dataviz and http://web.cse.ohio-state.edu/~hwshen/hwshen/ParallelVis.html for more examples

Motivation

- Scientists and Engineers want to see results
  - Focus on science, not on code.
- Most popular plotting libraries are CPU-only
  - Require GPU -> CPU -> GPU data copy for rendering!
  - Tend to focus on publication-quality figures, not rapid rendering
- GPU programming is (still) considered difficult
  - Need to know CUDA
  - Need to think for parallel programming
  - Direct porting of CPU applications to GPU isn't trivial.
- Make high performance visualzation as easy as GPU programming ArrayFire

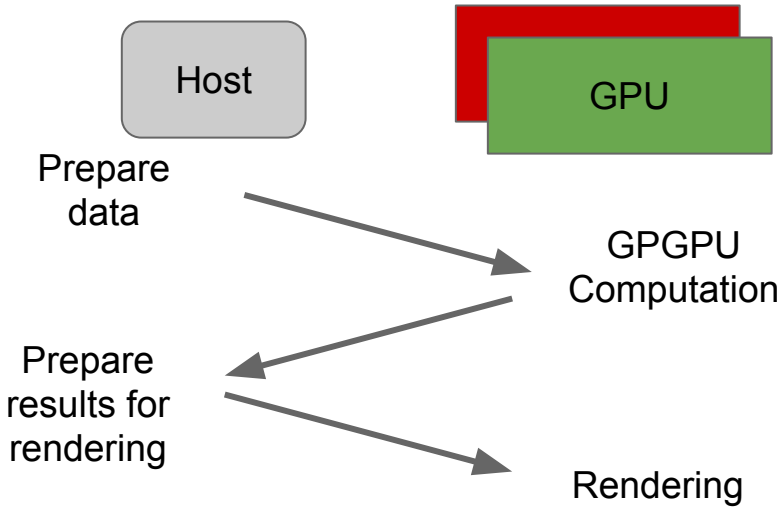
Our solution: ArrayFire Forge

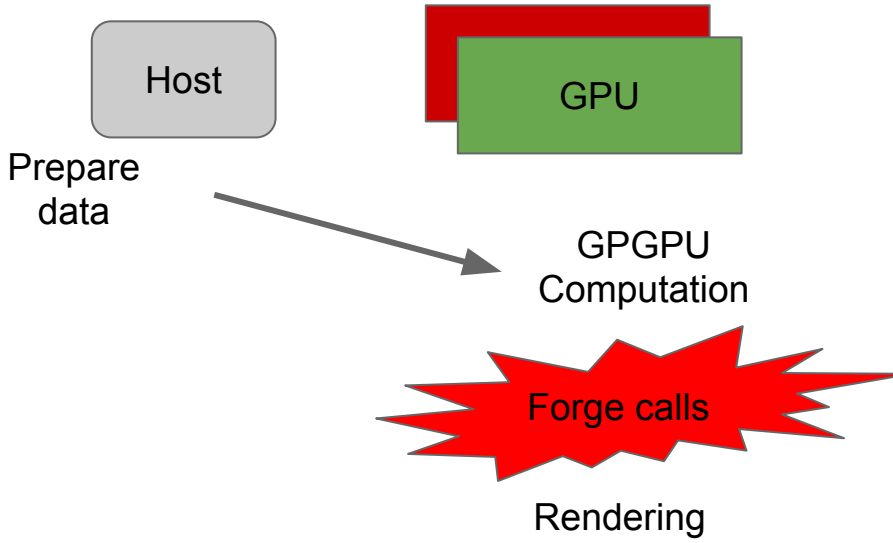
- Provide an easy-to-use API
- Design library for visualizing GPU computations
- Levege OpenGL for rapid rendering
- Enable real-time, interactive,
  2D or 3D visualizations



What does Forge do?

- Forge is for visualzing data only
  - Forge does not compute data for plots
- Use OpenGL interoperability to avoid data copies
  - Faster rendering than on the CPU





What does Forge do?

- Implement the most popular visualizations
  - 2D: Line, scatter, bar, images, vector fields, etc.
  - 3D: Line, scatter, surface.
- Use cross-platform dependencies for portability
  - GLEW
  - GLFW
  - Freetype
  - fontconfig
  - OpenGL 3.3
- Make plotting data on the GPU easy

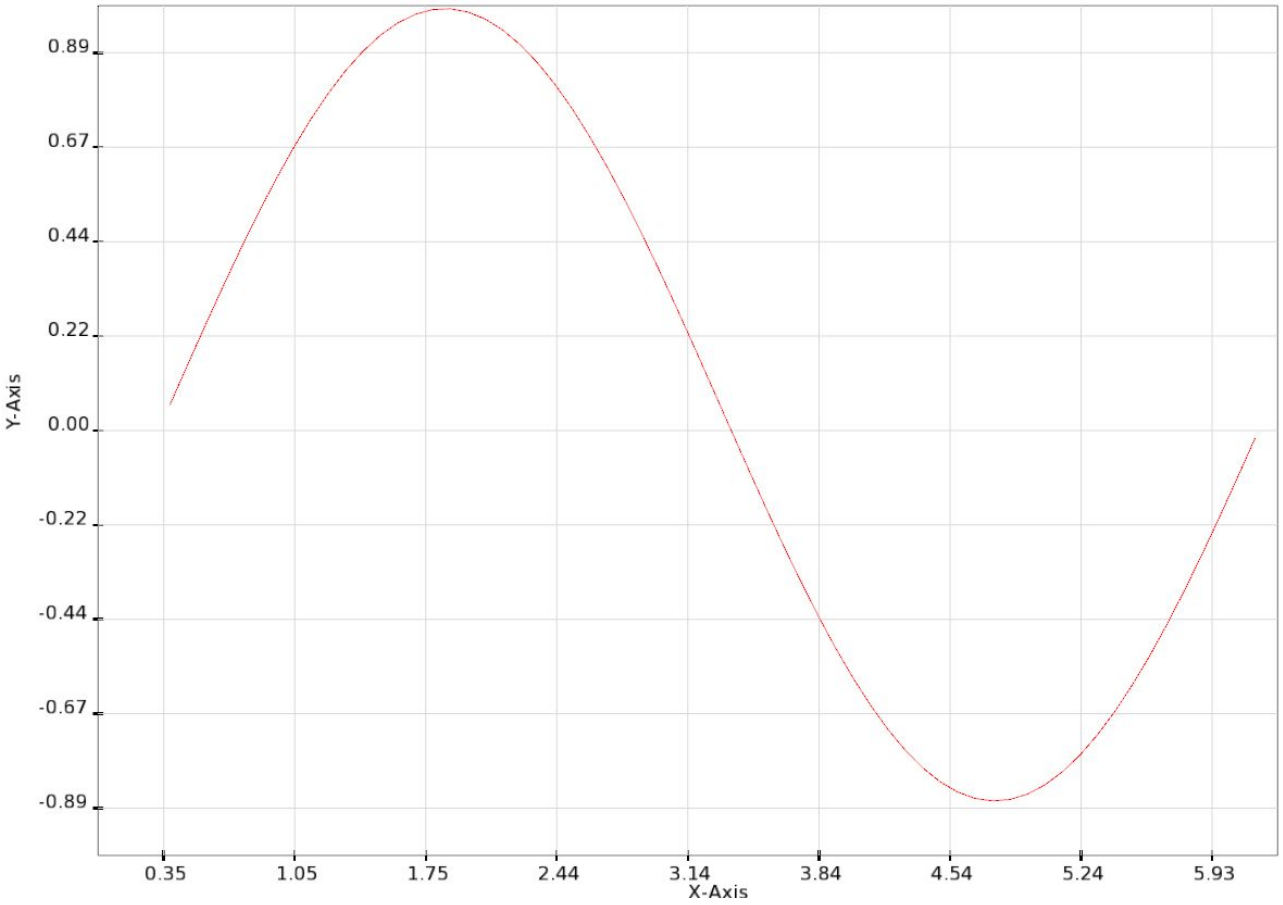
General workflow in Forge

- Create OpenGL context (must be first call)
- Prepare data using CUDA (or similar)
- Create an image or 2D/3D chart
- Create plots to be shown within the chart
- Alter chart/plot properties
- Move data to plot's VBO
- Display plots

Example: Plotting sin(x) using Forge

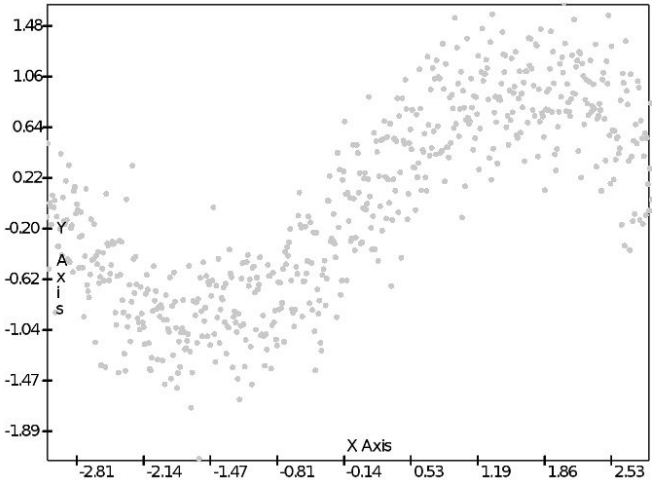
```
// Create data
std::vector<float> sinData;
map_range_to_vec_vbo(RANGE_START, RANGE_END, DX, sinData, &sinf);

// Make a Forge Window / OpenGL context
fg::Window wnd(DIMX, DIMY, "Plotting Demo");
wnd.makeCurrent();

// Create a Forge Chart
fg::Chart chart (FG_2D);
chart.setAxesLimits (RANGE_START, RANGE_END, MINVAL, MAXVAL);

// Add a line plot to the chart
fg::Plot plot = chart.plot(NUM_POINTS, f32);
plot.setColor(FG_RED);

// Copy data to the plot's VBO and render
fg::copy(plot.vertices(), plot.verticesSize(), (const void*)sinData.data());
wnd.draw(chart);
```

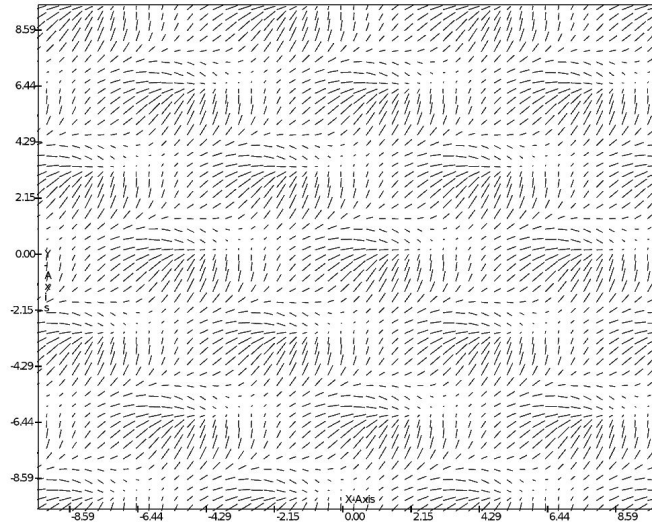
Example: Plotting sin(x) using Forge



Forge 2D plot examples

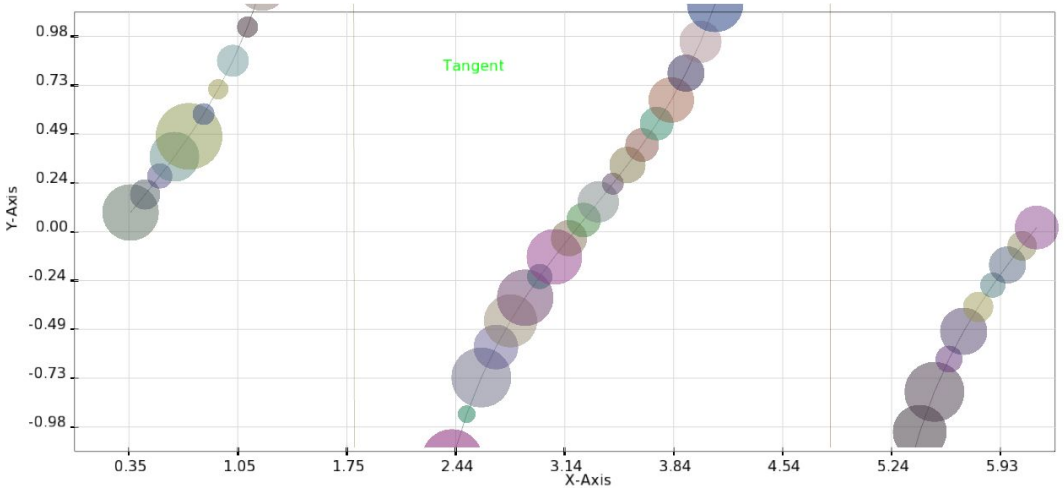
- Scatter
- Vector Field

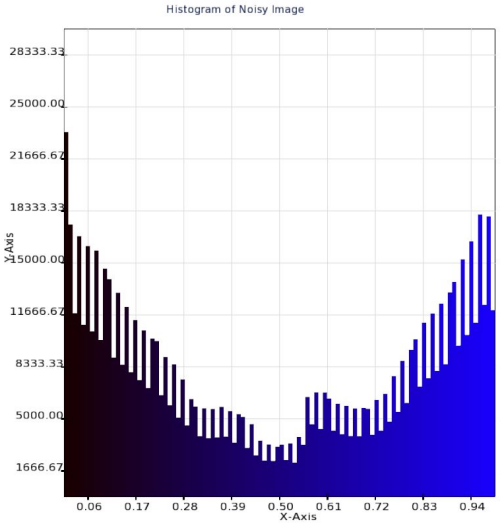




Forge 2D plot examples

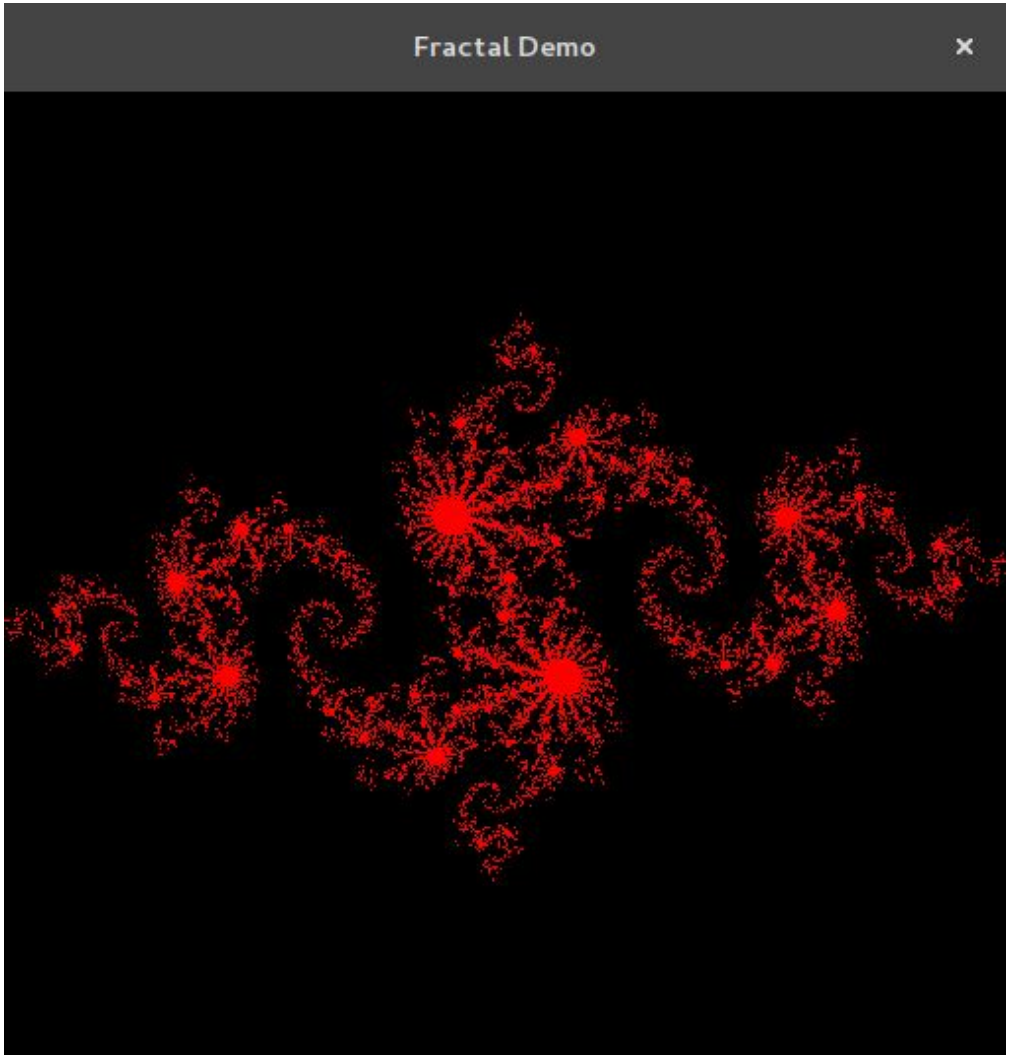
- Scatter
- Vector Field
- Bubble
- Bar
- Histogram



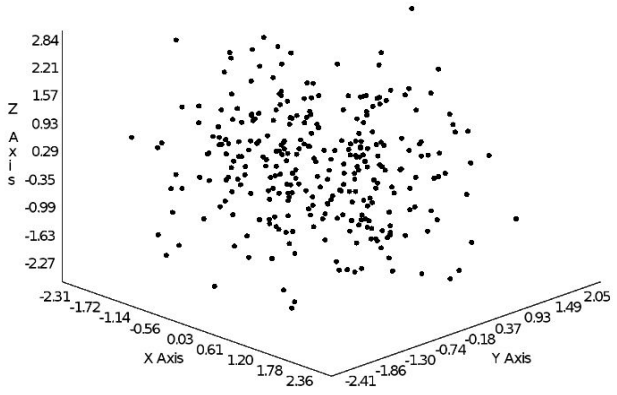


Forge 2D plot examples

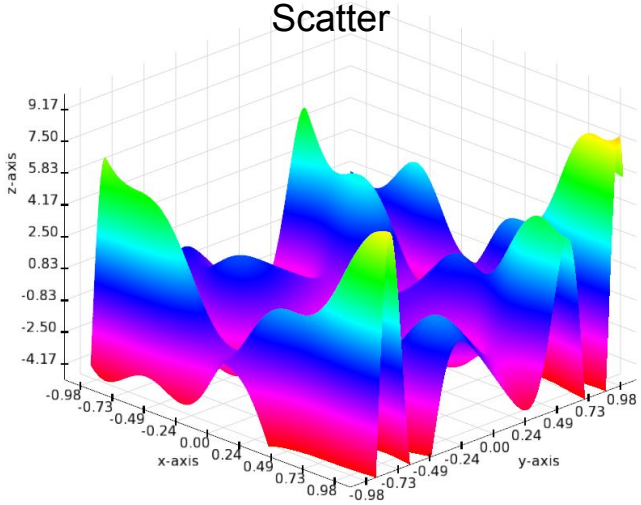
- Scatter
- Vector Field
- Bubble
- Bar
- Histogram
- Images
- Evolving simulations



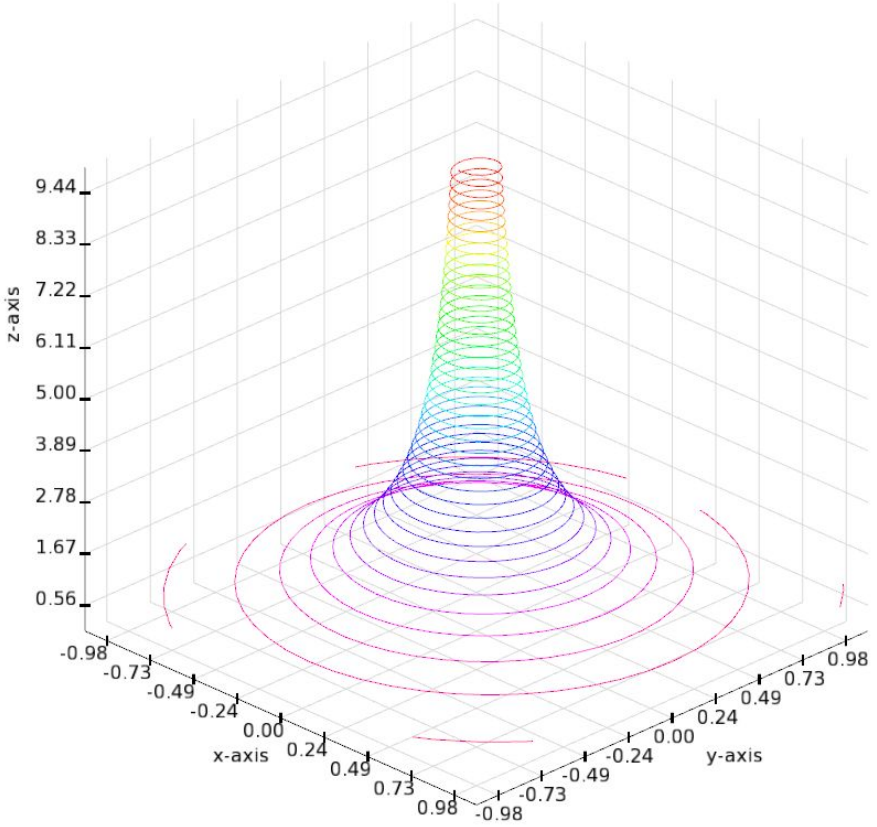
Forge 3D plot examples



Scatter



Surface



Line

Modifying Forge Plots

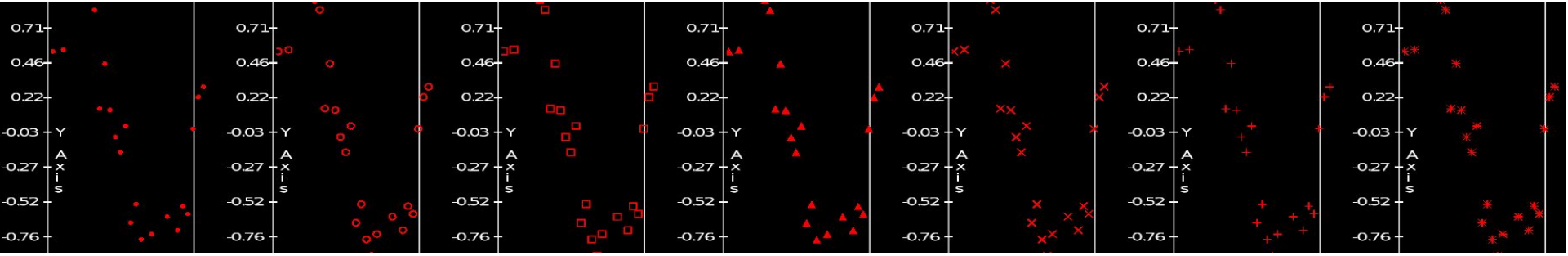
Forge plots allow editing several common properties:

- Titles
- Axis limits
- Colors
- Marker Types
- Legend
- Alpha

Consult documentation for full list!

Example: Changing glyph color

```
/*
 * Plot properties can be set during plot initialization
 */
fg::Plot plt = chart.plot(logData.size(), f32, FG_SCATTER, FG_CROSS);

/*
 * Or plot properties can be modified at a later time
 */
plt.setColor( FG_RED );
```



Conclusion

- Forge is for visualizing data
- Leverages CUDA/OpenCL OpenGL interoperability
- Most common plots implemented
- Cross platform
- Open source: BSD 3-Clause

Get a copy, contribute, and comment:

https://github.com/arrayfire/forge

Almost ready for 1.0 release, send us your comments!