TOC?

Der Universal Plotter

Was? Und vor allem.. warum?

Der Universal-Plotter ist eine Feature, mit welchem ihr euch möglichst einfach beliebige Testdaten visualisieren könnt. Auf Basis von Marcels Ideen und Erklärungen habe ich hier eine einfache Möglichkeit zum Exportieren von Daten geschaffen, welche auf dem frei zugänglichen Programm Gnuplot fußt. Die Klassen befinden sich im Package model.testdata. Der Aufbau des Plotters ermöglicht es euch, mehrere Schaubilder anzulegen, aus verschiedenen Klassen heraus verschiedene data-files zu verschiedenen Schaubildern anzulegen. Dabei habe ich versucht, die Nutzung flexibel, jedoch einfach zu halten.

Vorbereitung

Was braucht man also um Testdaten mit Gnuplot zu visualisieren? Testdaten.. und Gnuplot! Fangen wir doch einfach mit Gnuplot an: auf Sourceforge sollte es immer die aktuelle Version für euer OS geben (sogar für dieses Linuks, obwohl es gnu heißt.. was es nicht alles gibt). Die 32-Bit Version läuft übrigens auch auf 64-bit Systemen, anscheinend kann jedoch die Darstellung der Graphen wohl etwas dauern.

Von der Idee zum fertigen Plot

Wie bekomme ich nun meine Wertvollen Daten raus aus meiner Klasse und rein in eine Datei? Das 7-Schritte Erfolgsrezept für euch Helden (werktags von 9 bis 5).

1. Erstellen eines Schaubildes

Zuerst solltet ihr euch Gedanken darüber machen, was ihr eigentlich sehen wollt. Hilfe dazu gibt es hier. Dann einfach mal schauen, was euch new Plotter() so für Sachen erzählt: Zuerst müsst ihr wissen, wie eure Plotter-Datei heißen soll. Erstaunlicherweise hilft es hier, dem Kind einen sprechenden Namen zu geben ;). Dann muss noch eine Beschriftung der x- und der y-Achse her. Kein Problem, kennen wir ja aus der Schule. Tipp vom Profi: Einheiten machen das Leben leichter [kg]. Diese Plot-Datei wird jetzt im Root-Ordner eures Projektes angelegt. Wenn ihr zusätzlich noch einen Pfad angebt, könnt ihr sie sicher auch direkt in /dev0 anlegen. Es hilft, sich dieses Objekt an einer zentralen Stelle (beispielsweise eurer moduli-Hauptklasse) anzulegen. Für Dateiendungen wird übrigens überall automatisch gesorgt, Pfade sollten mit einem "/" enden.

2. Erstellen einer Datafile

Die wirklichen Daten eures Schaubildes werden nicht im Plotter selbst, sondern in einer zugeordneten data-File gespeichert. Dafür braucht man diese aber erst: also an beliebiger Stelle (vorzugsweise in der Nähe des Plotter-Objektes) mal new PlotFile?() ausprobieren. Für dieses Rezept werden benötigt: Mal wieder der Name der Datei (zB 'fileName = "X-Data".replace("X", this.toString())'), sowie die Angabe des Headers der Datentabelle. Grundsätzlich gilt, dass für jedes durch ein Leerzeichen getrennte Wort dieses Strings eine neue Spalte angelegt/erwartet wird. So resultiert "Frame Timestamp" also in zwei Spalten. Ihr könnt euch zum Beispiel für jede interessante Klasse einen (oder mehrere) Datensatz (-sätze) schreiben lassen. So erfolgt eine einfache Trennung von Daten aus verschiedenen Quellen. Weiterhin könnt ihr jeder Spalte Parameter mitgeben, welche die Formatierung erleichtern (sollen). Zur Zeit sind folgende Parameter implementiert:

- %TS : TimeStamp?. Die Spalte wird als TimeStamp? interpretiert. Es werden double-Werte in nanoSekunden erwartet, die Ausgabe erfolgt skaliert in milliSekunden. Soll der Beschriftung dienen.

- %0 : ZeroShift?. Der erste Wert dieser Spalte wird als Startpunkt interpretiert und auf 0 gesetzt, folgende Werte werden entsprechend verschoben. Dies ist zum Beispiel nützlich, um Echtzeitstempel mit 0 beginnen zu lassen.
- %G0 : GlobalZeroShift?. Dieser Parameter dient der Koordination der Spalten untereinander. Der erste eintreffende Wert irgendeiner mit %G0 indizierten Spalte setzt den Nullpunkt für das gesamte Plotsystems. Dieser wirkt (falls zum Beispiel verschiedene Teams zeitgleich Auswertungen betreiben) Plotter-übergreifend. Zum Beispiel können so Echtzeitstempel verschiedener dataFiles miteinander koordiniert werden.

Zur Zeit wird die header-Zeile in einem einzigen String übergeben, beispielsweise "Time%TS%G0 Framerate".

3. Einbinden der Plotter-Dateien

Jetzt muss jeder Plotter wissen, welche dataFiles zu ihm gehören. Einfach in die .add-Methode eures erstellten Plotter-Objektes das gewünschte PlotFile?-Objekt schieben, voilà, der Rest erledigt sich von selbst.

4. Anlegen des Graphen

Nun muss dem Plotter beigebracht werden, was er eigentlich zeichnen soll. Dazu muss er wissen, welche Spalten aus welcher Datei miteinander in Beziehung gesetzt werden sollen. Dazu die .addGraph-Methode des Plotters nutzen, einen Namen für den Graphen festlegen, PlotFile?-Objekt reinschieben und die beiden für einander bestimmten Spalten angeben. (Dies kann alternativ auch über Strings geschehen, so kann per Hand nachskaliert werden. Syntax: ($2*10), wobei die Zahl mit dem Dollarzeichen die Spalte angibt, mit der allerlei gerechnet werden darf.) Die erste Spalte hat übrigens die Nummer 1.

5. Anfügen der Daten

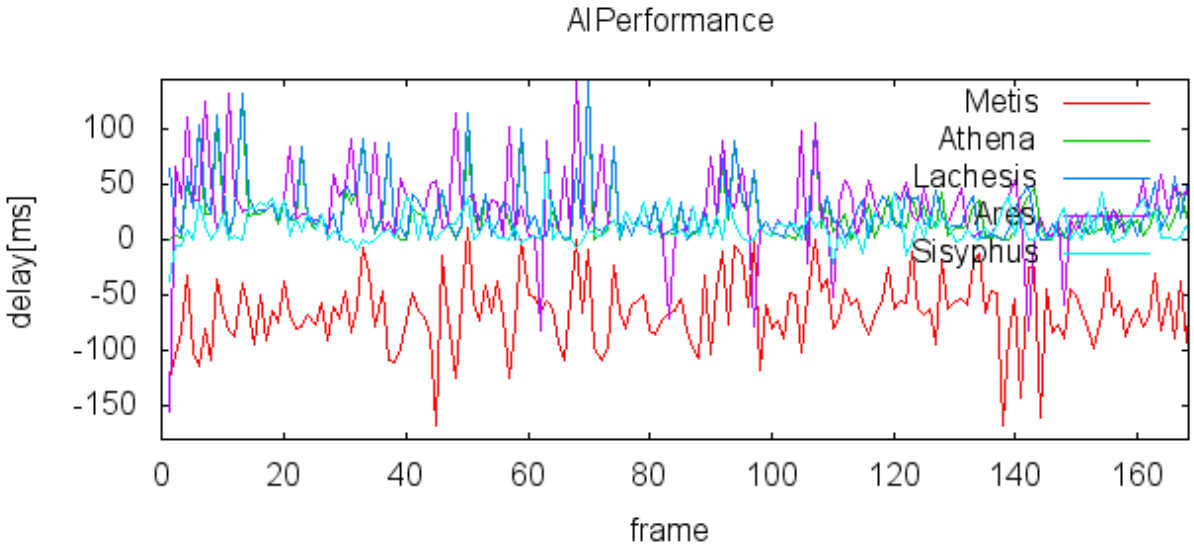
Um nun einen neuen Datensatz aus meinem Programm anzufügen, muss die .add-Methode der entsprechenden data-file genutzt werden. Hier kann - entsprechend der Anzahl der Spalten - einfach der gewünschte Datensatz (in der richtigen Reihenfolge) eingefügt werden.

6. Schreiben der Datensätze

Um das Programm nicht unnötig zu belasten, werden die Datensätze in Listen zwischengespeichert. Damit hier das System nicht ausgebremst wird, können sämtliche Listen über einen statischen flush-Befehl .. geflusht werden (Plotter.flush()). Das sollte vor allem passieren, wenn ihr fertig mit dem Aufzeichnen seid, zum Beispiel in der stop()-Methode des Moduls.

7. ... and profit!

Jetzt muss man nur noch den fertigen Graph genießen (und gegebenfalls analysieren o0)



Weiteres

- Es hat sich bisher bewährt, die dataFiles entweder in der zentralen Klasse anzulegen und auf Instanz-Variablen der schreibenden Klassen zu verteilen, oder aber in der schreibenden Klasse direkt auf einen public static PlotFile? der zentralen Klasse zuzugreifen.
- Bitte checkt (ohne Rücksprache) keinen Code in den Trunk ein, der plottet.
- Zum Öffnen der .gpl (gnu plot) müsst ihr dieses noch mit der gnuPlot.exe verlinken (auffindbar im binary-Ordner der geladenen gnu-plot Version). Datensätze und .gpl lassen sich (unter windows) am besten mit dem wordpad anschauen, der editor stellt die Zeilenumbrüche nicht dar.
- Da gnuPlot weiterhin ein Konsolenprogramm ist, wird im Hintergrund - eine Konsole erscheinen. Auf dieser steht "paused" - dankbarerweise, denn sonst müsstet ihr ziemlich schnell gucken, um eure Graphen zu erkennen :D Nach dem Schließen des Plots einfach ein Enter auf die Konsole geben. So wird diese sauber geschlossen und ein "last.png" mit eurer letzten (gezoomten) Ansicht des Plots erstellt (Danke an Marcel an dieser Stelle)
- Apropos Zoom, per Rechtsklick-Halten-Ziehen-Linksklick könnt ihr an eine beliebige Stelle des Plots zoomen, oben gibts einen zoom-zurück-Button

Neue Ideen und Bugs

Solltet ihr beim Testen Fehler bemerken, oder euch fällt etwas Neues ein, was ergänzt werden kann, tragt dies bitte hier ein:

ID	Description	Reporter	Status	Comment
0001	Alles läuft unglaublich gut	Geheimer-Verehrer		mit erstem Bug / erster Idee ersetzen

Last modified on Jun 6, 2010, 9:05:02 PM

Attachments (1)