Bachelor - Praktikum (Graphenlayout)

Designdokumentation Rev. 0

Gruppe: $G^{2^{2^2}}$

Stand: 14. Dezember 2006
`http://bp.macrolab.de`

Tutor: Thorsten Volland
Aufraggeber: Michael Eichberg
Gruppenmitglieder:

- Andreas Franke
- Peter Schauss
- Martin Konrad
- Marco Möller

Inhaltsverzeichnis

1 Design

1.1 Einleitung

Das Designdokument ist Teil unserer Dokumentation für das Bachelorpraktikum 'Graphenlayout'. Es soll einen Überblick über die Architektur unseres Produktes geben. Zudem sind unserer Designentscheidungen über die verwendeten Programme samt deren innerer Struktur und Zusammenspiel dokumentiert und deren Verwendung begründet.

Das Designdokument wird die Struktur des fertigen Produktes erklären. Dabei wird auf die Zusammenhänge der einzelnen Komponenten untereinander eingegangen, sowie die Interaktion der verschiedenen Klassen und Methoden betrachtet. Es wird dabei Bezug auf die Use-Cases aus unserem Pflichtenheft genommen.

1.2 Anwendungsplattform

Ziel unseres Projektes ist es ein Plugin für Eclipse 3.2 zu entwickeln. Zudem ist eine Lauffähigkeit unter Java 1.5 zu garantieren. Mit diesen Vorgaben ist die geforderte Hardware und Betriebssystem Unabhängigkeit leicht zu garantieren.

1.3 Architektur

Unser Produkt besteht aus drei Eclipse-Projekten. Das größte Projekt ist die Graph-API inkl. einer SWT-View Komponente. Darauf aufbauend werden wir eine Standalone-Version und eine Eclipse-Plugin-Version unseres Produktes erstellen. Dies hat den Vorteil, dass wir ohne großen Aufwand leichter Testen können und zudem einen echten Mehrwert schaffen. Diese letzten beiden Projekte dienen nur als GUI-Wrapper für das Graph-API Projekt.

1.4 Die Graph-API

Die Graph-API stellt Klassen und Methoden bereit, um Graphen zu erstellen und deren Layout zu berechnen. Graphen können aus Dateien geladen und wieder in Dateien gespeichert werden. Außerdem können Skripte ausgeführt werden, die auf mit den Graphen operieren.

Man kann den Graphen in einen GC (GraphicsContext) zeichnen lassen. Dazu muss vorher das Layout berechnet worden sein. Beim Zeichnen können Teile des Graphen mit Hilfe eines Filters ausgelassen werden.

Außerdem stellt die Graph-API eine View-Komponente zur Verfügung, mit Hilfe derer sich einfach die Darstellung und Interaktion von Graphen in SWT-GUIs einbinden läßt. Die View-Komponente ist vom SWT-Composite abgeleitet und läßt sich daher wie jedes andere SWT-Widget hinzufügen.

Die Graph-API besteht aus folgenden Klassen und Interfaces:

- GGraph
- GNode
- GEdge
- GFilterable
- GGraphListener
- GRubyScript
- GScriptable
- GView

1.4.1 Graphen erzeugen

Um einen Graphen zu erzeugen, erstellt man einfach ein neues GGraph-Objekt. Knoten und Kanten für den Graphen erstellt man zuerst unabhängig von diesem und fügt sie dann mit der add-Methode des Graphen hinzu.

1.4.2 Layout eines Graphen berechnen

Man kann das Layout eines Graphen berechnen lassen. Intern geschieht dies mit der Hilfe von Draw2D.

Die Berechnung des Layouts eines Graphen kann lange dauern. Deshalb geschieht diese Berechnung in einem eigenen Thread. Dadurch kann man die Berechnung abbrechen, falls sie zu lange dauert.

1.4.3 Die View-Komponente

Die View-Komponente ist ein SWT-Composite, das ein SWT-Canvas und eine SWT-Toolbar enthält. Wie bei SWT üblich, übergibt man ihr im Konstruktor eine SWT-Shell oder ein anderes SWT-Composite.

Der Graph wird im Canvas angezeigt. Der Benutzer kann ihn bewegen, indem er mit der linken Maustaste auf das Canvas drückt und dabei die Maus bewegt. Und er kann den Graphen zoomen, indem er mit der rechten Maustaste auf das Canvas drückt und dabei die Maus nach oben oder unten bewegt bzw. am Mausrad dreht.

Die Toolbar enthält die folgenden Buttons:

- Laden
- Speichern
- Exportieren
- ...

1.4.4 Klassenbeschreibungen

- GGraph ist ein Graph, bestehend aus Knoten (GNode) und Kanten (GEdge).
- GEdge ist eine Kante, die sich zu einem Graphen hinzufügen läßt.
- GNode ist ein Knoten, der sich zu einem Graphen hinzufügen läßt.
- GFilterable
- GScriptable
- GRubyScript
- GGraphListener
- GView ist ein SWT-Composite, das einen Graphen anzeigt und den Benutzer damit interagieren läßt.

1.4.5 Modulstruktur

Siehe Abbildung 1.

1.4.6 Klassendiagramme

Siehe Abbildung 2.

1.4.7 Dateiformat

Wir haben ein eigenes XML Dateiformat analog zu GraphML[5] spezifiziert.

Abbildung 1: Die Grobe Modulstruktur unseres Plugins.

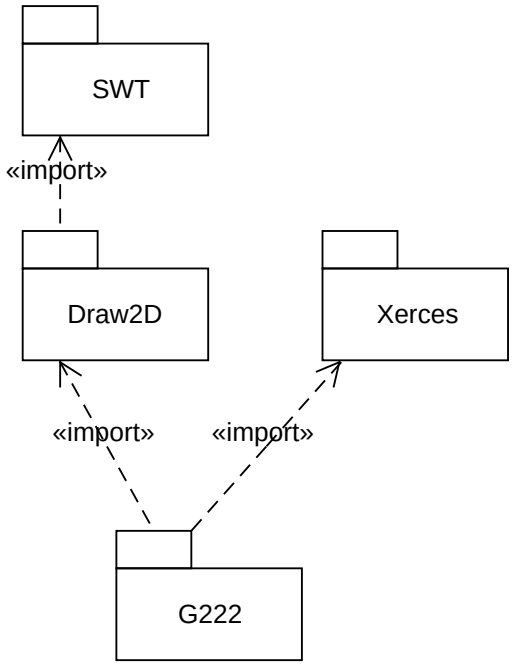
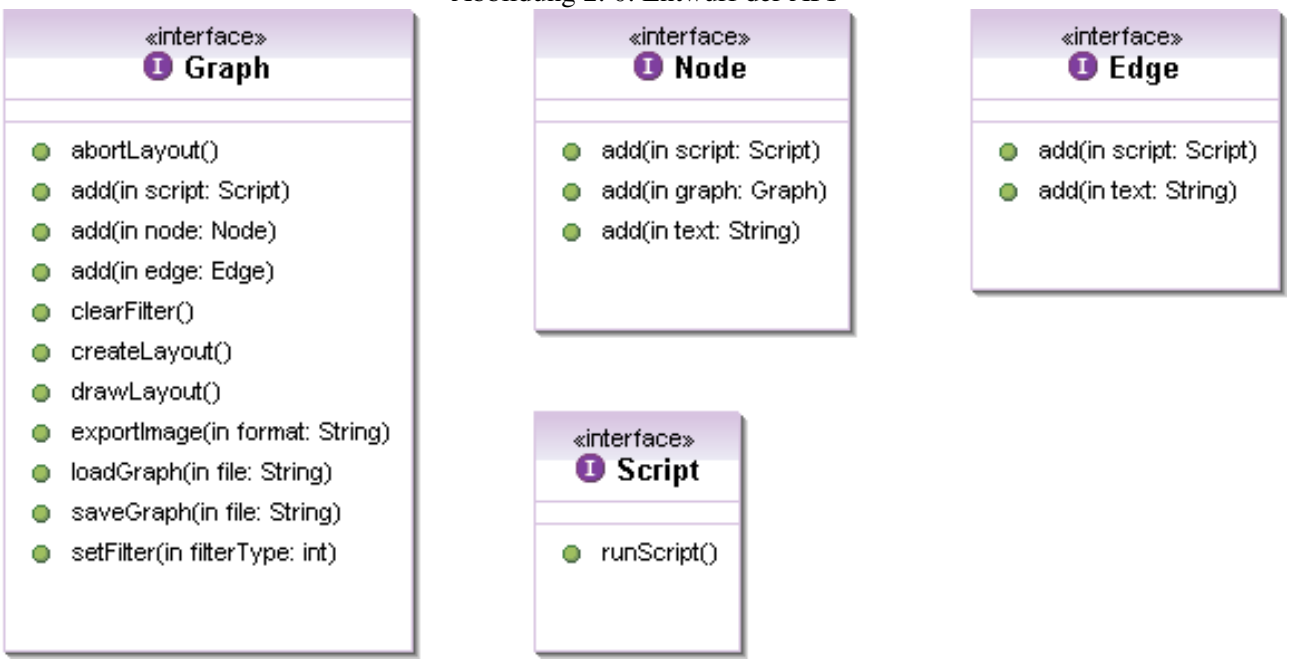


Abbildung 2: 0. Entwurf der API



1.5 Das Graph-Plugin

Das Graph-Plugin ist eine Eclipse-View, die mit Hilfe der View-Komponente aus der Graph-API einen Graphen darstellt. Die Funktionalität ist dieselbe, die schon die View-Komponente selbst hat.

1.6 Das Stand-Alone-Programm

Dieses Programm zeigt ähnlich wie das Graph-Plugin, mit Hilfe der View-Komponente aus der Graph-API, einen Graphen an. Die Funktionalität ist dieselbe, die schon die View-Komponente selbst hat.

1.7 Änderungshistorie

Datum	Thema	Inhalt	seite
14.12.2006	alles	Beginn der History mit Revision 0	*

Glossar

Eclipse

Eclipse ist eine offene Entwicklungsplatform, die auf Java-Technologie beruht. *3*

Literatur

[1] http://www-128.ibm.com/developerworks/opensource/library/os-ecplug/.

[2] Eclipse 3.2 Documentation. http://help.eclipse.org/help32/index.jsp.

[3] PDE Does Plug-ins. http://www.eclipse.org/articles/Article-PDE-does-plugins/PDE-intro.html.

[4] SVG Eclipse Plugin. http://sourceforge.net/projects/svgplugin/.

[5] The GraphML File Format. http://graphml.graphdrawing.org/.

[6] Eclipse.org home. http://www.eclipse.org/, 2006.

[7] StarUML. http://staruml.sourceforge.net/, 2006.

[8] subversion. http://subversion.tigris.org/, 2006.