Designdokumentation des Prototypen: dgsdemo Gruppe: geo10

Verantwortlicher: Alexander Ullrich

4. Mai 2003

1 Allgemeines

Die Aufgabe war es einen Prototypen zu schreiben, der die grundlegenden Funktionen eines *Dynamischen Geometrie Softwareproduktes* darstellt. Damit ist die Eingabe per Maus und Tastatatur, sowie die Darstellung eines beweglichen und zoombaren Gitters gemeint.. Da es sich bei dem Prototypen um eine Anwendung mit grafischer Oberfläche handelt, kommt hier das MVC-Prinzip zum Einsatz, welches standardmäßig bei Anwendungen mit GUI (Graphical User Interfaces) genutzt wird. Wer mehr über MVC lesen möchte kann dies zum Beispiel in dem MVC-Bericht dieser Gruppe auf der Homepage pcai003.informatik.uni-leipzig.de/~ geo10 tun.

2 Produktübersicht

Was also soll denn der Prototyp *dgsdemo* genau tun?

Die Aufgabe war klar, implementieren sie ein Fenster, in der die Mausbewegung überprüft wird. In einer Statusleiste am unteren Rand des Fensters soll die aktuelle Mausposition ausgegeben werden und ob die Maus sich innerhalb des Fensters bewegt.

Desweiteren sollte ein weiteres Fenster implementiert werden, welches ein Gitter und ein Koordinatensystem ausgibt. Per Tastatur soll es möglich sein das Gitter zu verschieben (Cursortasten) und zu zoomen (Plus- und Minustaste). Es wurde nicht gesagt welche Plus-/Minustaste benutzt werden soll, deswegen haben wir uns entschieden , beide nutzbar zu machen. Eine Statusleiste gibt wieder den aktuellen Zustand des Fensters an, diesmal jedoch nicht die Daten der Maus, sondern die Position des Koordinatenursprungs in Weltkoordinaten und denn aktuellen Zoomfaktor. Der Zoomfaktor für die Verkleinerung wurde auf ein Minimum von 0.2 gesetzt, da bei kleineren Faktoren das Gitter zu einer schwarzen Fläche wird und zusätzlich der Rechner arge Probleme bekommt (in Bezug auf die Berechnung der unzähligen Linien).

Beide Fenster haben am oberen Rand des Fensters jeweils einen ‘Schließen’- und einen ‘Info’-Button. Ebenso wie das Hauptfenster, welches der Einstieg in das Programm ist. Der Nutzer kann alles Schließen (‘Schließen’-Button), einen Informationsdialog aufrufen (‘Info’-Button) oder die beiden Fenster (MPanelView-das Maustestprogramm; CPanelView-Gittertestprogramm) öffnen. Wird das Hauptfenster per ‘Schließen’-Button geschlossen, so werden noch geöffnete Fenster mit geschlossen.

3 Grundlegendes Design

Der Prototyp wurde in drei Hauptkomponenten konzipiert. Das Hauptfenster (*Main*), das Demofenster für die Mausbewegungen (*MPanelView*) und das Demofenster für die Darstellung des beweglichen Koordinatensystems (*CPanelView*), die allesamt von der Swing-Klasse *JFrame* abgeleitet werden. Zur Realisierung der Buttons in den jeweiligen Fenstern werden Instanzen der Klasse *JButton* verwendet. Die Behandlung der, von den Buttons ausgelösten, Action-Events erfolgt über eine Spezialisierung der Klasse *AbstractAction*, die Klasse *AbstractActionAdapter*. Diese überschreibt die Methode *actionPerformed()* und nutzt die integrierte Hash-Tabelle um per *putValue()* und *getValue()* Referenzen auf externe Objekte zu halten. Über die genannte Referenzen ist es möglich Operationen auf externe View-Objekte (*CPanelView*, *MPanelView*, *CStatusBar*, etc.) auszuführen und die auslösenden Buttons eindeutig per Hash-Code zu identifizieren.
Für das Mausbewegungsgebiet in *MPanelView* und das Koordinatengitter in *CPanelView* werden die entsprechend angepaßten Ableitungen (*MPanel* und *CPanel*) der Swing-Klasse *JPanel* verwendet. Da bei der Mausüberwachung in *MPanel* die Events *mouseEntered*, *mouseExited* (*MouseListener*) sowie *mouseMoved* (*MouseMotionListener*) Verwendung finden, wird als Adapter eine

modifizierte Version des *MouseInputListener* benötigt. Zu beachten ist dabei, dass diese Klasse als *MouseListener* UND als *MouseMotionListener* in *MPanel* registriert werden muß.
Zur Vereinfachung der Bewegung und Skalierung des Koordinatengitters in *CPanel*, wird auf die Methoden *transform()*, *translate()* und *scale()* der Klasse *AffineTransform* zurückgegriffen. Die benötigten Tastatureingaben werden über eine Spezialisierung (*CKeyAdapter*) der Klasse *KeyAdapter* behandelt. Zu diesem Zweck wird die Methode *keyPressed()* überlagert. Auch hier gibt es wieder eine Besonderheit zu beachten: per Default verliert, beim Start der Fensters, das *CPanel* seinen Fokus an die beiden Buttons „Info“ und „Schliessen“. Da dieser Fokus aber zur korrekten Weiterleitung der Key-Events not- wendig ist, muß er per Hand (*Component.requestFocus()*) wieder an *CPanel* zurückgegeben werden.
Da in der Swing-Bbliothek keine vordefinierte Klasse für eine Statusleiste existiert wurde die Statusleiste (*CStatusBar*) als *JPanel* mit 2 integrierten *JLabel*-Objekten konstrukiert. Damit kann sie bequem in jeden LayoutManager (empfohlen wird *BorderLayout*) aufgenommen und über die entsprechenden Schnittstellenmethoden (*setText1()*, *setText2()*, *clear()*, etc.) angesprochen werden. Zur detailierten Darstellung der Klassenbeziehungen sei auf die entsprechenden Klassendiagramme im nächsten Abschnitt verwiesen.

4 Klassendiagramme

Abbildung 1: Übersicht übder die Klassenbeziehung ohne Betrachtung der Methoden und Attribute

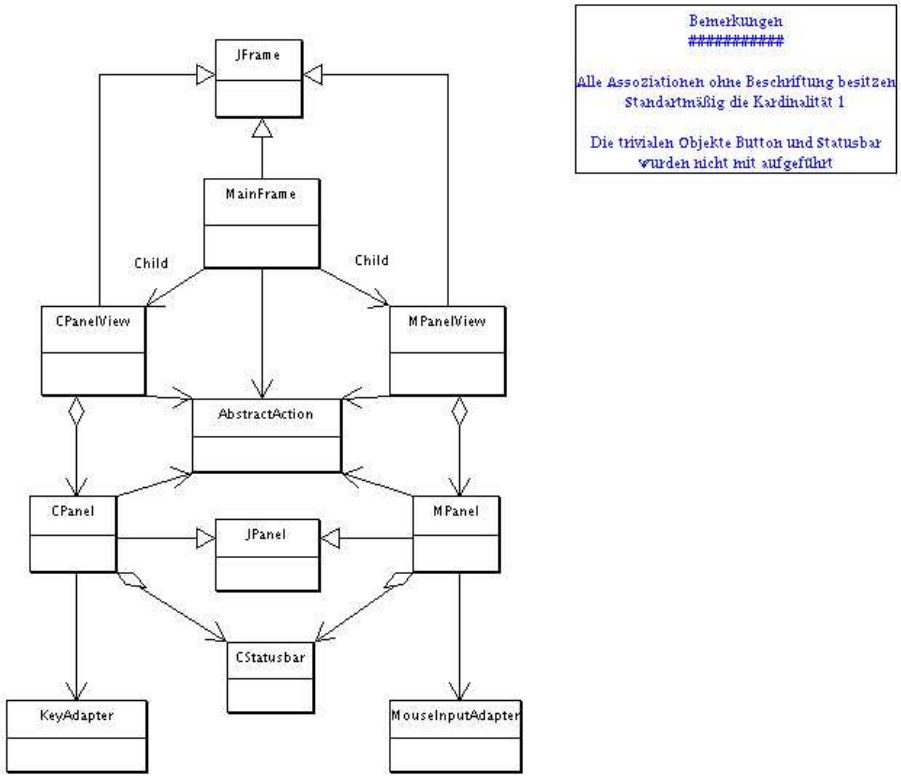

Abbildung 2: Die Klasse - CStatusbar

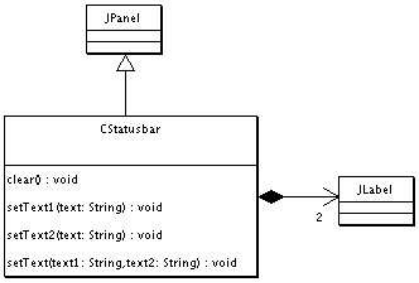

Abbildung 3: Die Klasse CPanelView

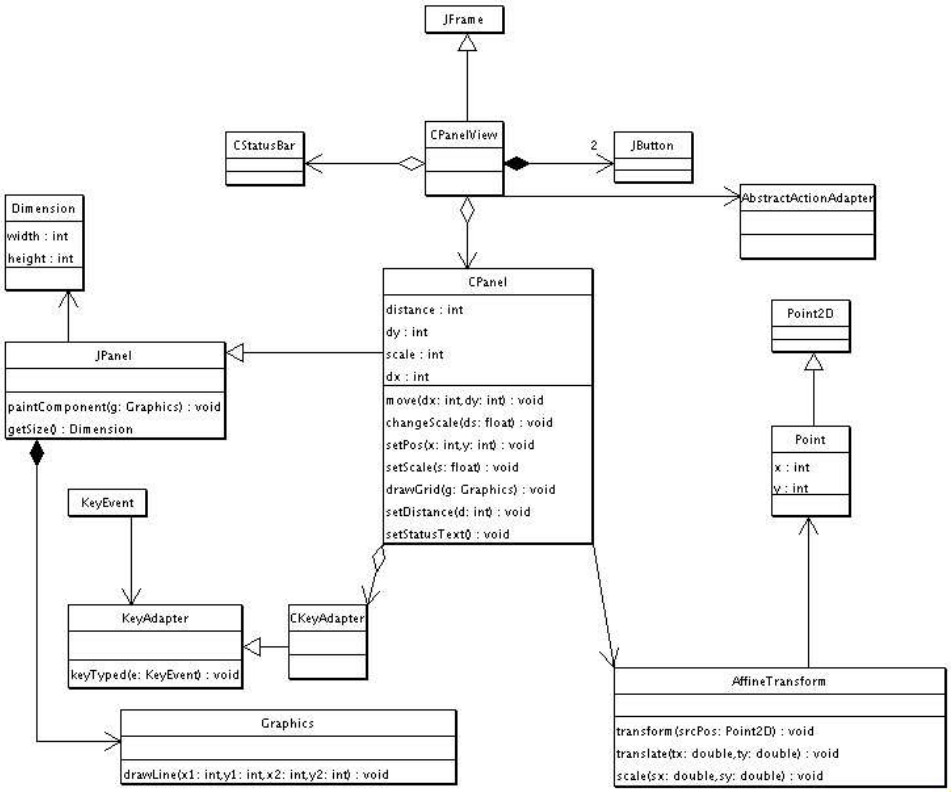

Abbildung 4: Die Klasse MPanelView