Erstellt von Andrej Aderhold 8961518, Yves Metzner 8960810 [geo08]
Erstelldatum 04.05.2003 7:23

Designbeschreibung

[Prototyp]

Inhaltsüberblick

Erstellt von Andrej Aderhold 8961518, Yves Metzner 8960810 [geo08]
Erstelldatum 04.05.2003 7:23

1 Allgemeines

1.1 Kurzcharakteristik

Der Prototyp soll als Studie eines ersten Frameworks des GUI (graphic user interface - *View*) sowie der Benutzer - Applikations Interaktion (*Control*) dienen. Dabei handelt es sich um den ersten Ansatz einer stark grafisch orientierten Einzelplatzanwendung die als Applikation oder innerhalb eines Webbrowsers als Applet zum Einsatz kommen kann.

Die dabei implementierte Funktionalität beruht hierbei auf den Vorgaben aus den in der Aufgabenstellung geforderten Funktionen [*Geo-2 Ergänzungsblatt*]. Dabei wurde im Vorfeld darauf geachtet die Prinzipien des MVC-Ansatzes zu befolgen und das Paradigma des Java-Ereigniskonzepts umzusetzen.

1.2 Systemvoraussetzungen

Der Prototyp läuft auf allen Systemen mit einem aktuellen Stand-Alone-Java-Interpreter (java virtual machine), während die Ausführung als Java-Applet aus einer HTML-Seite heraus einen Java-fähigen Webbrowser benötigt. Eine Mindestanforderung an die Performance des Systems ist nicht bekannt.

1.3 Entwicklungsumgebung

Das Klassendesign wurde in Rational Rose ausgearbeitet. Die Implementierung wurde mittels eines Texteditors durchgeführt (UltraEdit). Das Rational Rose Projekt dient als Basis für weitere Versionen.

2 Produktübersicht

2.1 Modulbeschreibungen

Der Prototyp setzt sich aus 7 Klassen zusammen. Im folgendem werden diese näher beschrieben:

Main – implementiert das Kontrollfenster; ausführbar als Applet oder Applikation
CPanelView – implementiert das Koordinatenfenster mit Tastaturinteraktion
MPanelView – implementiert das Mausfenster mit Mausinteraktion
InfoBox – zeichnet ein Fenster mit Benutzer-Interaktionsinformationen
InteractionArea – implementiert Maus- und Tastatur Events
CoordinateArea* – implementiert Koordinatensystem; Schnittstelle zwischen Control und View Komponente (M/CPanelView)

2.2 Funktionsbeschreibung

Die Funktionalität schließt zunächst ein Kontroll Fenster ein in dem via Mausinteraktion weitere Fenster, die als separate Viewklassen realisiert wurden, geöffnet werden können. Die Fenster demonstrieren Mausinteraktion mittels Ereignisbehandlung und die Einbettung einer Zeichenfläche auf der ein Koordinatensystem dargestellt wird, welches mittels Benutzerinteraktion transformiert werden kann.
Weiterhin bieten alle Fenster die Möglichkeit eine separate Infobox mit Eingabeoptionen für den Benutzer einzublenden.

2.3 Aussicht

Der nächste Schritt wird daraus bestehen, dass Klassendesign mit Hilfe von Rational Rose weiter zu verfeinern und vor allem im *Modell*-Bereich erste Ergebnisse zu erzielen.

Es wird eine einheitliche Klasse für das Koordinatensystem aus den schon Bestehenden konzipiert, sowie eine zentrale Klasse *PanelView*. Davon ausgehend planen wir das Hinzufügen von Graphikelementen, darunter ein ‚Iconbar' mit ‚Speedbuttons'über die man die verschiedenen Geometrieklassen auswählen kann. Ein ‚Iconbar' mit verschiedenen Transformationsoperationen sowie Menüleiste, Statusleiste und ein Fenster über das man auf die schon erzeugten Geometrie Objekte zugreifen kann. Die Geometrieklassen werden in Zukunft im Packet *models* vereint.

3 Grundsätzliche Designentscheidung

3.1 MVC-Konzept

Die Designentscheidung für den Prototyp orientiert sich direkt an den Vorgaben des Ergänzungsblattes Geo-2. Dabei steht das MVC Konzept im Mittelpunkt des Klassendesigns.

Zur Konstruktion des User-Interface wird das Model-View-Controller-Paradigma angewandt, das aus den drei Klassen Model, View und Controller besteht. Das Model-Objekt stellt die Kernfunktionalität und das Anwendungsobjekt dar, das View-Objekt seine Bildschirmrepräsentation, und das Controller-Objekt bestimmt die Möglichkeiten, mit denen die Benutzungsschnittstelle auf Benutzereingaben reagieren kann. Das MVC-Konzept erlaubt somit eine klare Schichtentrennung zwischen den Anwendungsdaten, den Sichten auf die Anwendungsdaten und der Benutzungsschnittstelle.

Die konkrete Realisierung dieses Paradigmas in unserem Projekt wird aus den Darstellungen der Packet- und Klassenstruktur deutlich.

3.2 Event-Handling

Das JDK Version 1.1 führte das Delegation Event Model ein, welches eine klare Trennung zwischen Programmcode zur Oberflächengestaltung und solchem zur Implementierung der Anwendungslogik erlaubt. Der Grundgedanke dabei ist es, auch Nicht-Komponenten die Reaktion auf GUI-Events zu ermöglichen. Jedes Objekt kann sich als Ereignisempfänger registrieren, solange es die erforderlichen Listener-Interfaces implementiert.

Bei dem Prototyp passiert die Event Behandlung in der Klasse *InteractionArea,* die die Interfaces *MouseMotionListener, KeyListener* und *MouseListener* implementiert. Klassen die sich von dieser ableiten können die bereitgestellten Funktionen beliebig überladen bzw. direkt benutzen. Die Klasse *InteractionArea* wurde von uns dem Packet control zugeordnet.

4 Packet- und Klassenstruktur

4.1 Packetstruktur

Bei der Packetstruktur haben wir uns am MVC-Konzept orientiert. Die Klasse Main.class bleibt zunächst im Basisverzeichnis und wird auch keinem Packet zugeordnet. Die für die Darstellung verantwortlichen Klassen ordnen wir dem Packet *views* zu. Die für die Benutzerinteraktion verantwortlichen Klassen (*InteraktionArea)* befinden sich im *control* Packet.

4.2 Klassenstruktur