Swing Framework

Teil III:

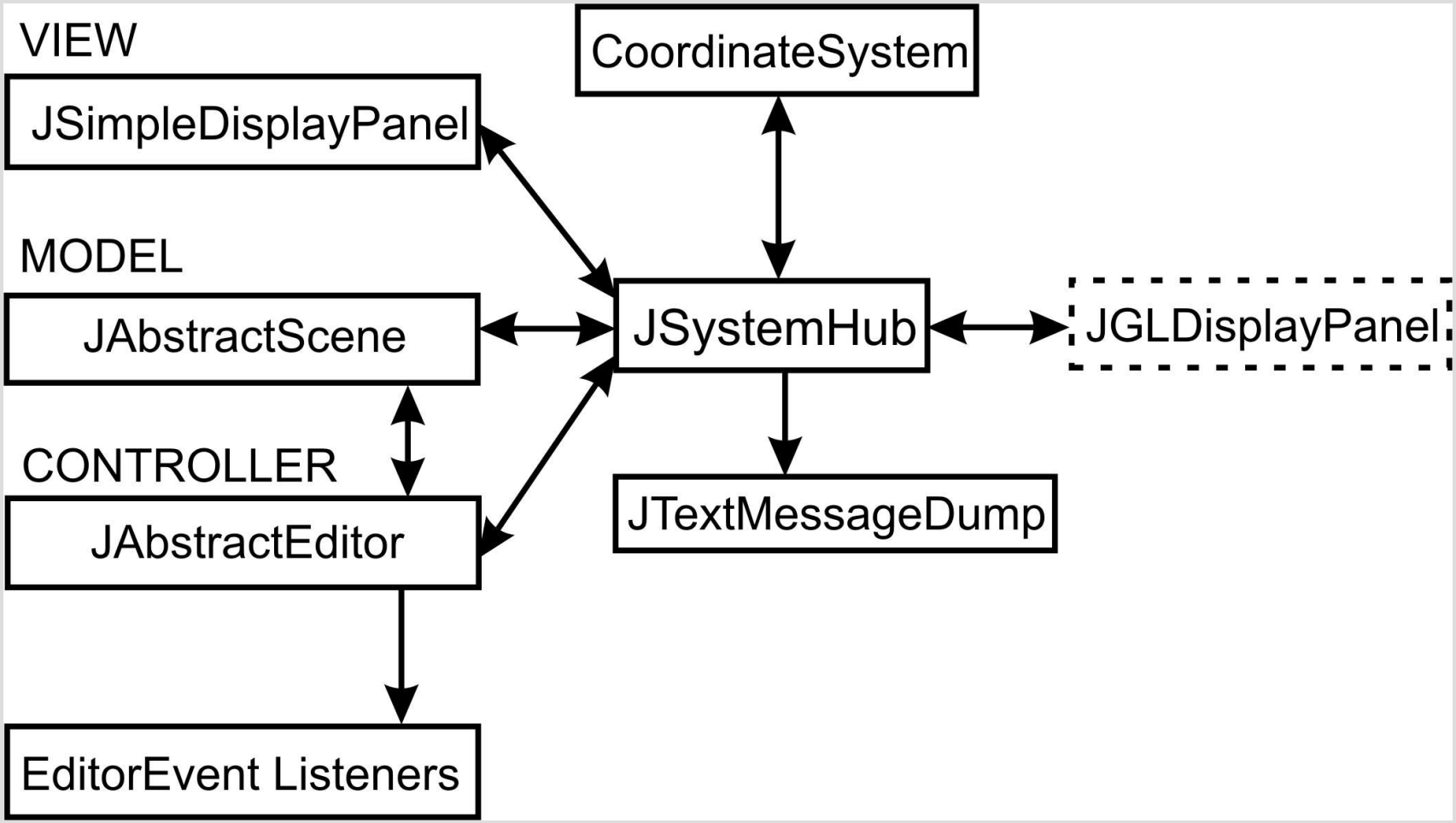
Kurze Einführung in Swing Framework
und Einblick in die Funktionalität

Swing Framework Überblick

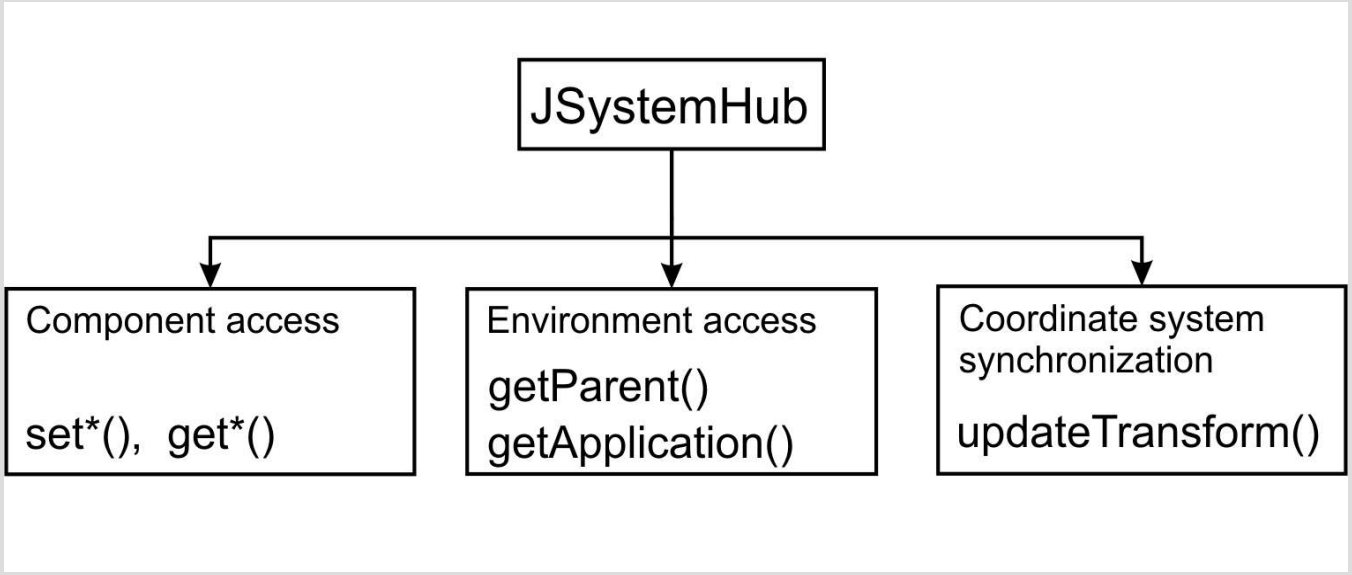
- Neues Softwaregerüst für verschiedene Swing-basierte Programme.
- Ein (hoffentlich) einfaches und benutzerfreundliches Aufbau.
- Leicht erweiterbar.
- Einige vorgefertigte Komponenten sind schon verfügbar.

Aufbau

Basiert weitgehend auf dem Model-View-Controller Paradigm:

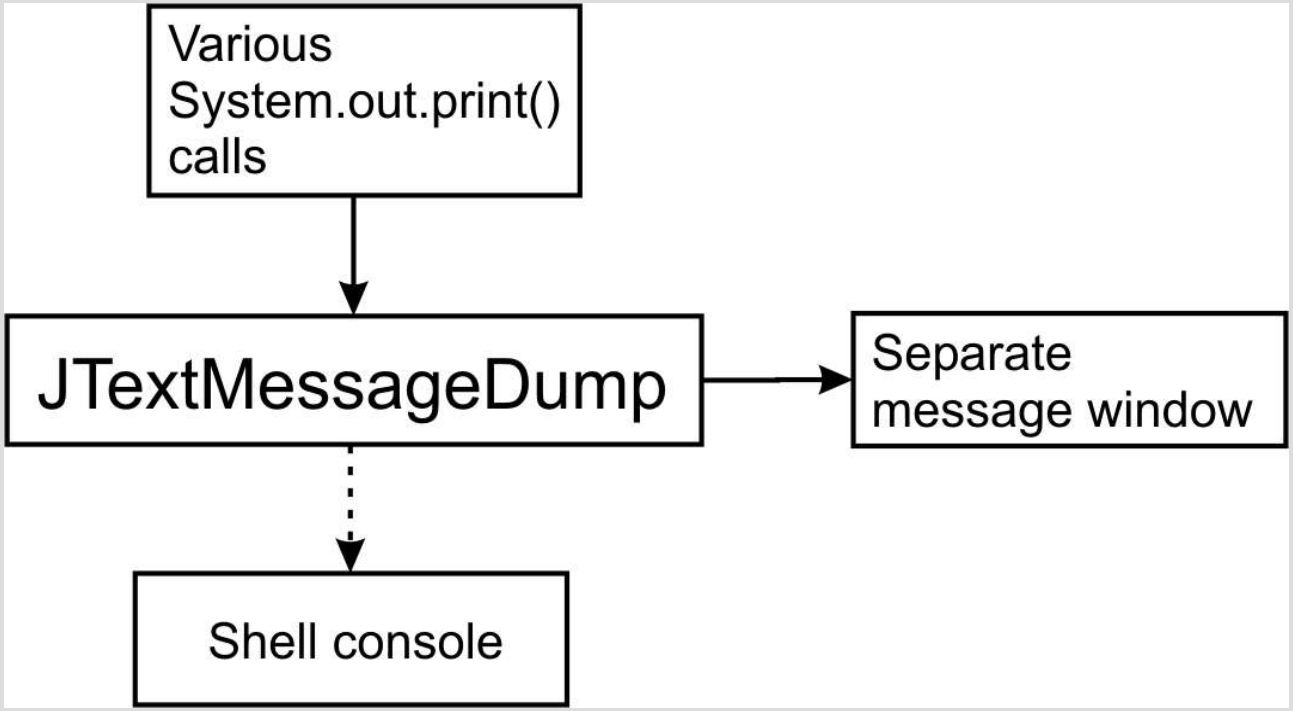


JSystemHub



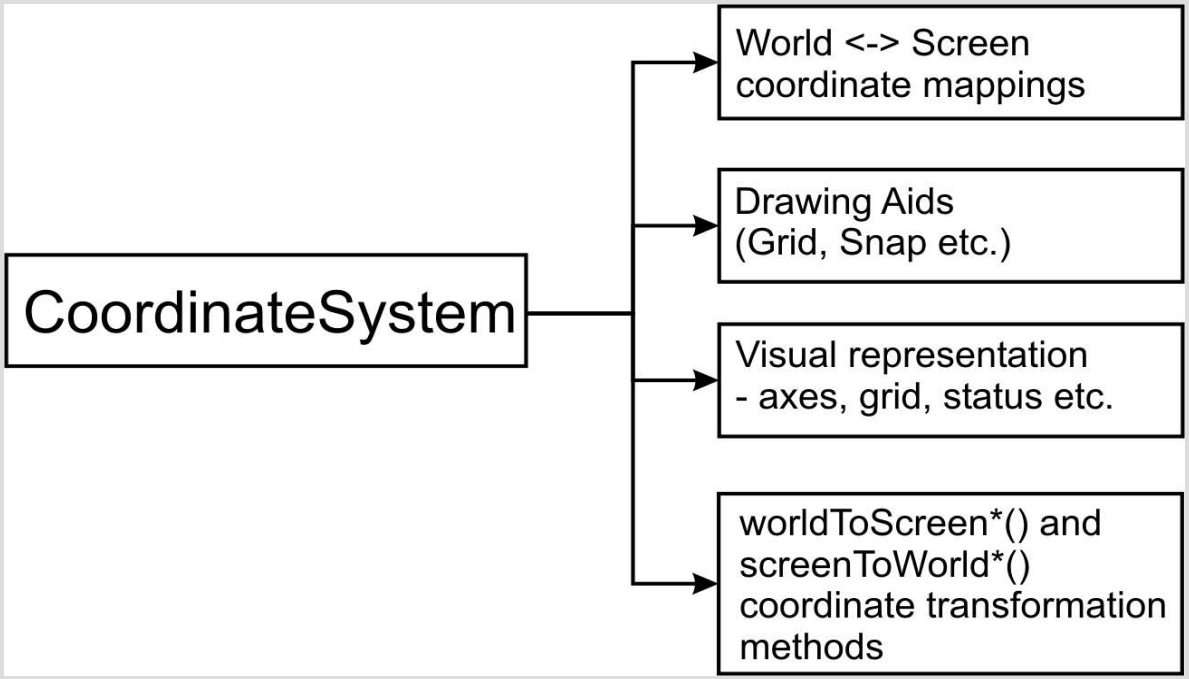
- Durch JSystemHub greifen die Komponente aufeinander zu.
- Mann kann auch auf das Applikation-Objekt zugreifen (JFrame).
- updateTransform() synchronisiert Koordinatensystem – Veränderungen mit allen Komponenten.

JTextMessageDump



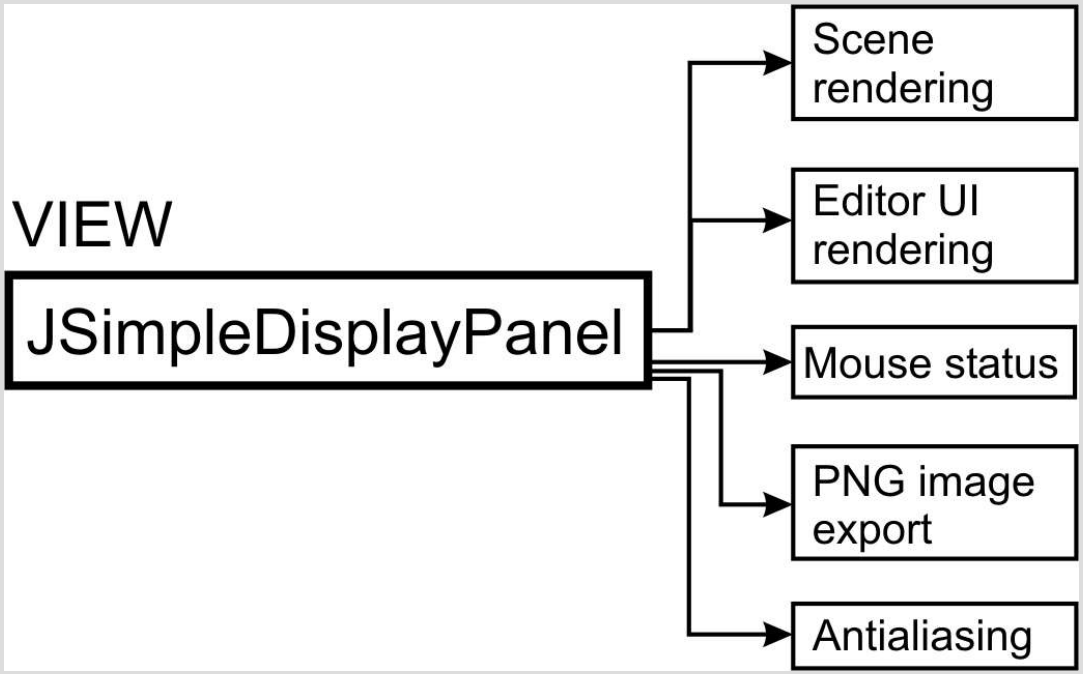
- Fängt alle System.out.print*() und System.err.print*() Aufrufe ab.
- Kann für Ausgabe von zusätzlichen Infos bzw. Debugging verwendet werden.
- Kann jederzeit abgeschaltet werden.

CoordinateSystem



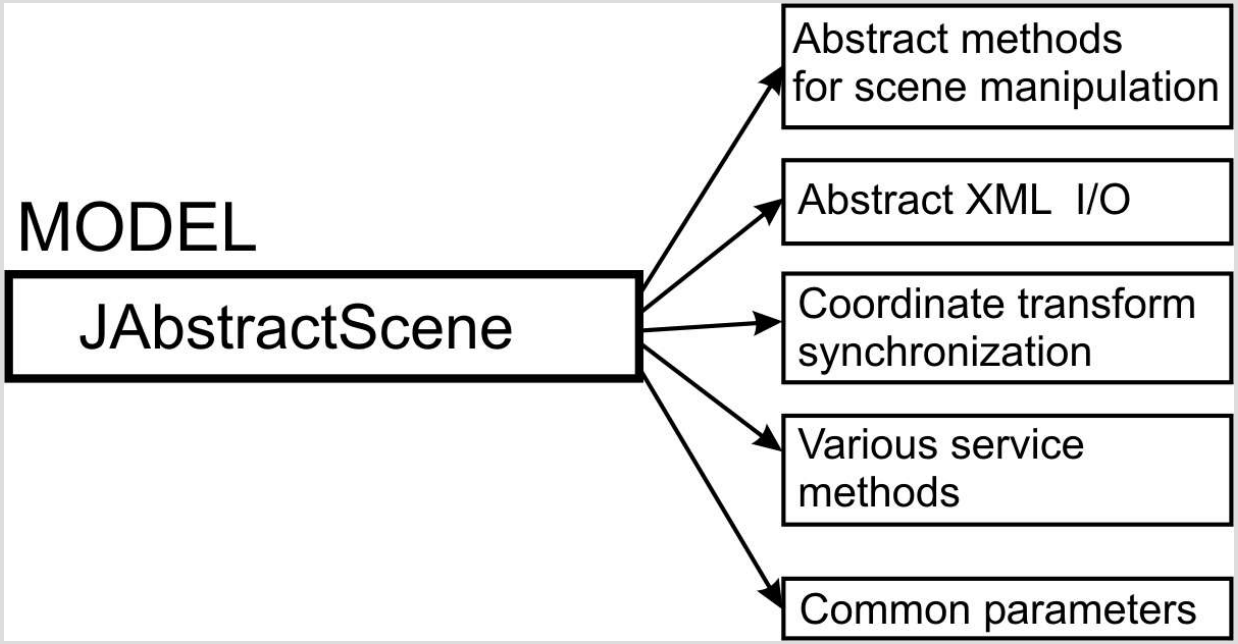
- Bildet Welt- und Bildschirm Koordinaten aufeinander ab.
- Stellt das Koordinatensystem visuell dar.
- Bietet einige einfache Hilfsmittel fürs präzise Zeichnen an.
- Stellt verschiedene Transformationsmethoden zur Verfügung.

JSimpleDisplayPanel



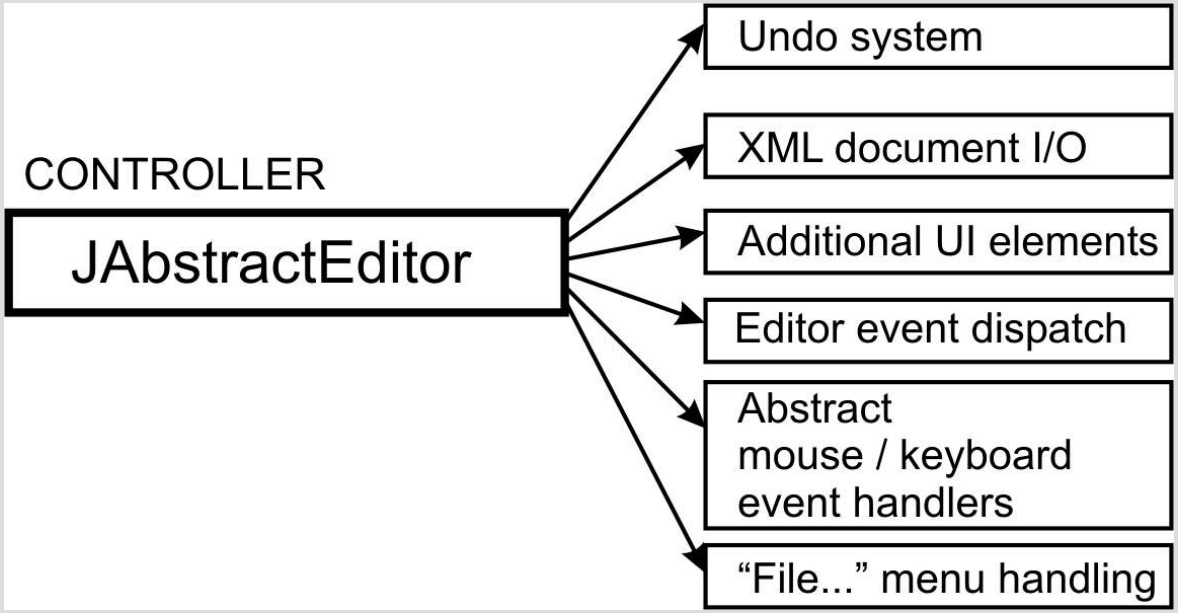
- Visualisiert die Szene und evtl. Editor-Elemente.
- Leitet Maus-events zum Koordinatensystem / Editor.
- Zeigt aktuelle Zeigerposition in kartesischen / polaren Koordinaten.
- Führt evtl. Kantenglättung durch.
- Erlaubt Speicherung des aktuellen Bildes als PNG Datei.

JAbstractScene



- Spezifiziert die grundlegenden Eigenschaften einer Szene.
- Abstrakte Spezifikation von Methoden. (display, modifikation, etc.)
- Keine Datenstrukturen! (Da abhängig von der Anwendung.)
- Einige gemeinsame Methoden (scene change flag etc.)
- Abstraktes XML I/O.

JAbstractEditor



- Eine abstrakte Spezifikation eines Editors.
- Gemeinsame Methoden für das Undo System (mit JAbstractUndo).
- XML-Dokument I/O.
- Zusätzliche Bedienelemente und Menüs.
- Gemeinsame Methoden für EditorEvent – Verteilung.
- Abstrakte Spezifikation von Maus / Tastatur Eventhandlern.

JGLDisplayPanel

- Optionales 3-D Display (siehe z.B. Voronoi Dilation Viewer).
- Baut auf JOGL (OpenGL Java Bindings) auf.
- Kann unabhängig von dem 2-D Ansicht bedient werden.
- Kann den EditorListener Interface implementieren und damit auf die Veränderungen im 2-D Ansicht reagieren.

Achtung: Momentan erfordert das Swing Framework das Vorhandensein vom JOGL auf dem System, um richtig zu kompilieren bzw. zu laufen! In kürzer Zeit werde ich eine zusätzliche Version von JsystemHub bereitstellen, für Anwendungen die kein GL Display brauchen.

Hilfsklassen

JAbstractUndo - Basisklasse für alle Undo-Funktionen.

JTinyColorChooser - eine platzsparende Alternative zum Swing-internen farbauswahl - Tool.

JAbstractFileFilter - vereinfacht Aufbau von Dateidialogen.

JVirtualTrackball - ein Navigationswerkzeug für 3-D Ansichten.

Und noch ein paar andere....

Subpackages

Wichtige Pakete:

- event.* - Enthält Interfaces und Basisklassen für verschiedene Editor eventobjekte.
- icons.* - Alle momentan vom System benutzte Icons / Cursors im PNG Format.
- graph.* - Enthält alle Komponente für ein generelles Graph Editor, fungiert auch als Basis für andere Graphen-bezogene Editoren.
- point.* - Enthält im Moment zwei vom graph.* abgeleitete Punktmengen-Editoren (generisches und für Dilationsaufgabe).

Kleines Beispiel

```
public VoronoiDilationEditor(JFrame parent)
{
   super();

   // initialize and glue together editor parts
   m_Hub = new JSystemHub();
   m_Hub.setParent(this);

   m_textDump = new JTextMessageDump(m_Hub);
   m_textDump.setMaxNumberOfLines(500);

   m_Coordsystem = new CoordinateSystem(m_Hub);
   m_Coordsystem.setVisible(true);
   m_Coordsystem.setAxisLabels("X", "Y");
```

Initialisierung

```
    m_Scene     = new JPointDilationScene(m_Hub);
    m_Editor    = new JPointDilationEditor(m_Hub);
    m_Display   = new JSimpleDisplayPanel(m_Hub);
    m_glDisplay = new DilationConeDisplay(m_Hub);

    /*
     * This makes the dilation cone display to
     * listen to graph editor events.
     */
    m_Editor.addEditorListener(m_glDisplay);

    m_Coordsystem.addActionListener(m_glDisplay);

    createUI(parent);
}
```

Initialisierung

```
/**
 * This sets various UI properties and creates UI components that
 * are not part of the SwingFramework, i.e. menu bars, borders
 * around editor panels etc.
 *
 */
protected void createUI(JFrame parent)
{
    //set up the UI
    JMenuBar menubar = new JMenuBar();

    // add menus from the editor etc.
    menubar.add(m_Editor.getFileMenu());
    menubar.add(m_Editor.getEditMenu());

    menubar.add(m_Display.getMenus());
    menubar.add(m_Coordsystem.getMenus());

    parent.setJMenuBar(menubar);
```

```
.....

    // put everything together
    this.setLayout(new GridLayout(1,2));

    // on the left side is the container for the editor
    JPanel editor_container = new JPanel();

    editor_container.setLayout(new BorderLayout());
    editor_container.add(m_Display, BorderLayout.CENTER);
    editor_container.add(m_Editor.getEditorUI(),
BorderLayout.SOUTH);

    .....

    this.add(editor_container);
    this.add(m_glDisplay);

    // UI is ready - outta' here!
}
```

Abschluss

- Für weitere Informationen könnt ihr die Implementierung vom Dilation Diagramm Editor anschauen – es ist im Grunde ein komplettes Beispiel für Verwendung von SwingFramework.
- Die Klassendokumentation ist weitgehend fertig, fehlende Teile werden nachträglich ergänzt.
- Für eventuelle Fragen stehe ich gerne zur Verfügung.

Danke für eure Aufmerksamkeit !