The Graphical Editing Framework
Agenda

- About the GEF project
- Draw2d Introduction
- GEF Tutorial
- Tips and Techniques
- On the horizon
- Q & A
What is GEF?

- Graphical Editing Framework
- Part of the Eclipse Tools Project
  - http://www.eclipse.org/gef
- A feature with 2 plug-ins
  - Draw2d
  - GEF
- Stable
- Active
  - news://news.eclipse.org/eclipse.tools.gef
GEF History

- Visual Age
- Create a foundation for GUI builders, and more
  - (Now the Eclipse Visual Editor Project)
- 4 years active development
- Used for
  - Class diagrams
  - Organization charts
  - Flow/Activity diagrams
  - State machines
  - E-R diagrams
  - GUI builders
GEF Overview

- Interaction Layer
- Model-to-View mapping
- Workbench Integration
- Rendering
- Layout
- Printing
- Native (SWT) Layer
GEF Overview
Draw2d

- Lightweight toolkit built with SWT
- Optimized layout and painting

Features:

- Zoom
- Print
- Overview
- Layering

- Hierarchical Tree Layout 3.0
- Directed Graph Layout 3.0
- Non-rectangular figures
- Decorated connections
GEF (the plug-in)

- An editing framework based on *Viewers*
- The “interaction” layer
- **Draw2d** for graphics
- MVC architecture
  - Flexible mappings between model and view
  - B.Y.O.M.
GEF Features

- Palette
  - Standard set of tools
  - User customization allowed
- Undo/Redo support
- Direct-edit (in place editing)
- Rulers and Guides 3.0
- Snap-to-{Guide, Grid, Geometry} 3.0
- Accessible: keyboard, voice, magnifier
Draw2d – Introduction

- Hello World
- Constructing a UML diagram
Draw2d – Hello World

```
1. Display d = new Display();
2. Shell shell = new Shell(d);
3. shell.setLayout(new FillLayout());
4.
5. FigureCanvas canvas = new FigureCanvas(shell);
6. canvas.setContents(new Label("Hello World"));
7.
8. shell.open();
9. while (!shell.isDisposed())
10.     while (!d.readAndDispatch())
11.         d.sleep();
```
Draw2d – Behind the Scenes

- `swt.widgets.Canvas`
- `FigureCanvas`
- `IFigure`
- `LightweightSystem`
  - `EventDispatcher`
    - Mouse
    - Keyboard
    - Focus
  - `UpdateManager`
    - Schedule layouts
    - Calculate damage
Draw2d – UML Diagram Example

- Class (or type) figure
  - Name and icon
  - Compartments for attributes/methods
- Associations/Inheritance
- “Sticky” notes for documentation
Draw2d – UML Class Figure design

- Extend Figure
- Header
  - Just a Label figure
- Compartments
  - Vertical ToolbarLayout
  - Custom border for separator line
- LineBorder around class
- Another ToolbarLayout for the whole figure
Draw2d – UML Class

- Hmm, Not quite perfect..
- Toolbar layout tweaks
  - #setMinorAlignment(TOP_LEFT)
  - #setStretchMinorAxis(false)
Draw2d – Relationships and Inheritance

- PolylineConnection
  - Delegating layout manager
- PolygonDecoration
  - #setTemplate(PointList)
- ChopboxAnchor
Draw2d – Labeling Connections

- ConnectionEndpointLocator
- ConnectionLocator

Fig 1

Midpoint

Endpoint

Fig 2
Draw2d – Sticky Notes

- org.eclipse.draw2d.text
- FlowPage – root for “flowing” figures
- TextFlow – wraps text in a paragraph
- Custom “folded-corner” border
Draw2d – UML Diagram

This is a sticky note. It wraps text based on the width of the page.
GEF – Tutorial

- Based on article @ developerWorks
- For more reference articles, go to:
  http://www.eclipse.org/gef
GEF – Tutorial

Step 1: Model

Step 2: View

Step 3: Controller

Step 4: “Editor”

Editing Behavior

Edit Policies

Property Sheet

Palette and Tools
Step 1 – The model

- Bring your own model
- Model requirements
  - Notification mechanism
  - Persistence is your responsibility
- Business model vs. Diagram model
- Commands
Step 2 – The View

- Draw2d Figures
- Don’t reinvent the wheel
- Information hiding
- Encapsulate to reduce coupling
Step 3 – Controllers: EditParts

- Unit of “interaction”
- Selection is comprised of EditParts
- Figure or TreeItem-based
- EditPartViewer
- Special EditParts:
  - Root
  - Contents
  - Connections
Step 4 – Bring it all together

- Create your workbench part (EditorPart)
- An EditDomain
- Instantiate some viewer
- Set the factory and contents for the viewers
Next – Adding Edit Support

- Add editing capability
  - Commands to change model
  - Install *Edit Policies* for commands and feedback
  - Add listeners to the model to refresh UI

- Edit policies
  - Pluggable helpers on an editpart
  - Handle a specific part feature
  - May contribute to feedback, commands, and targeting
  - Examples include
GEF – Conventions and Patterns

- Tools to interpret mouse and keyboard
- Requests to encapsulate interactions
- Absolute coordinates
- Edit Policies for separation of concerns
- Command pattern for undo/redo
- Use of IAdaptable
GEF: Model - Controller – View
GEF – Tips and Techniques
T&T: Accessibility

- Eclipse is accessible
- GEF is accessible
- IAdaptable#getAdapter(Class)
  - AccessibleEditPart
    - Magnifier and Screen reader API
- Focus indication (Selection Edit Policies)
- Default keyboard handlers
- Accessible Tools
  - AccessibleAnchorProvider
  - AccessibleHandleProvider
T&T: Auto Scrolling

- During drag operations, including native DND
- Search from target part upwards
- **AutoExposeHelper**
  - `#detect(Point)`
  - `#step(Point)`
- Not just for scrolling
  - Expanding
  - Page-flipping
- **Related**: **ExposeHelper**
  - Programmatically “reveal” an EditPart
T&T: Rulers and Guides

- Work in progress
- Viewers now have properties
- Ruler-specific properties:
  - Horizontal ruler
  - Vertical ruler
  - Ruler visibility
- RulerComposite
- RulerProvider
  - Provide guide locations
T&T: Snap-To-“G”

- **Grid**
  - Dragging and resizing confined to grid coordinates

- **Geometry**
  - Dragging and resizing snap to the rows or columns implied by existing objects in the diagram

- **Guides**
  - Snap to user defined horizontal or vertical guides
T&T: Snap-To-“G” continued

- IAdaptable#getAdapter(Class)
- SnapToStrategy
- Extended data on Request
- Client’s responsibilities
  - Representing guides in the model
  - Model/Command support to attach parts to guides
T&T: Direct Editing

- `DirectEditRequest` sent by select tracker
- May contain mouse location or modifier keys
- `DirectEditManager`
  - Manages CellEditor lifecycle
  - Tracks modification and committing
  - Live feedback on diagram
  - Value validation
  - Obtains the command for applying value
- Improved CellEditor API in Eclipse 3.0
T&T: Animation

- Demonstrated in GEF palette and Flow Example
- Create an Animation coordinator class
- Capture the effects of layouts as they occur
- Playback the layout incrementally
  - $0 < \text{progress} < 1$
  - Each step calls revalidate on participants of animation
- Force layouts without repainting
- It’s not always easy
T&T: Property Sheet Support

- **Implement** `IPropertySource` on
  - the model object
  - the EditPart
  - Custom adapter for combining multiple sources
- GEF provides undo/redo support via commands
T&T: Zoom and Scaling

- Use a root EditPart supporting scaling
- Actions and widgets for toolbars/menus

- Separation and compression
- Use back-to-front painting
- Same issues apply for overview and printing
Future Work

- More and better graph layout algorithms
- Connection Routers
- Investigate new mediums
  - OpenGL
  - Java2D
- Provide common shapes/symbols
- WYSIWYG Documents and Text
- New Palette objects and presentations
Questions?