G.E.F. 4

Alexander Nyßen
itemis AG
Graphical Editing Framework Project Lead

Image courtesy of Julie Lafrance / flickr

GEF 3.x / Zest 1.x

- **Standard** for **graphical editors/views** in Eclipse
- **Mature** project with quite **long history**
- Base technology with **lot's of users** (direct & indirect through GMF/Graphiti)
- **Stable API**, no breaking API changes since 2004 (GEF 3.0)

Image courtesy of peptic_ulcer/ flickr

GEF celebrated 10th Birthday in 2012!

Initial contribution by IBM in 2002

Image courtesy of Will Clayton / flickr

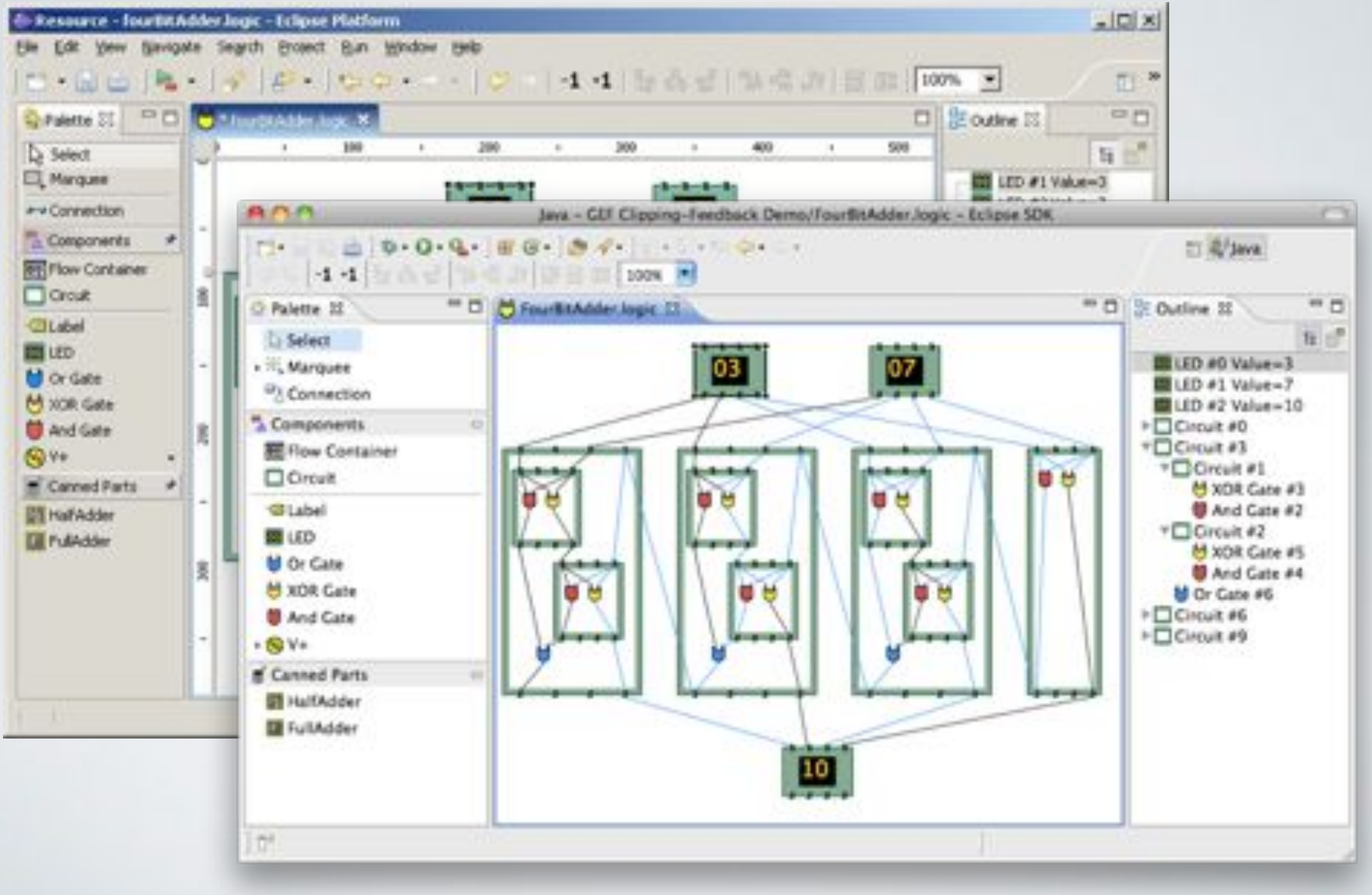
Draw2d & GEF (MVC)

- Initial contribution of Draw2d & GEF (MVC) by IBM in 2002.

Draw2d - 2D rendering framework; lightweight extension to SWT. May be used stand-alone or as visualization technology for GEF (MVC).

GEF (MVC) - an interactive model-view-controller framework, which fosters the implementation of SWT-based tree editors and Draw2d-based graphical editors (and views) for the Eclipse UI Workbench.

Draw2d & GEF (MVC)

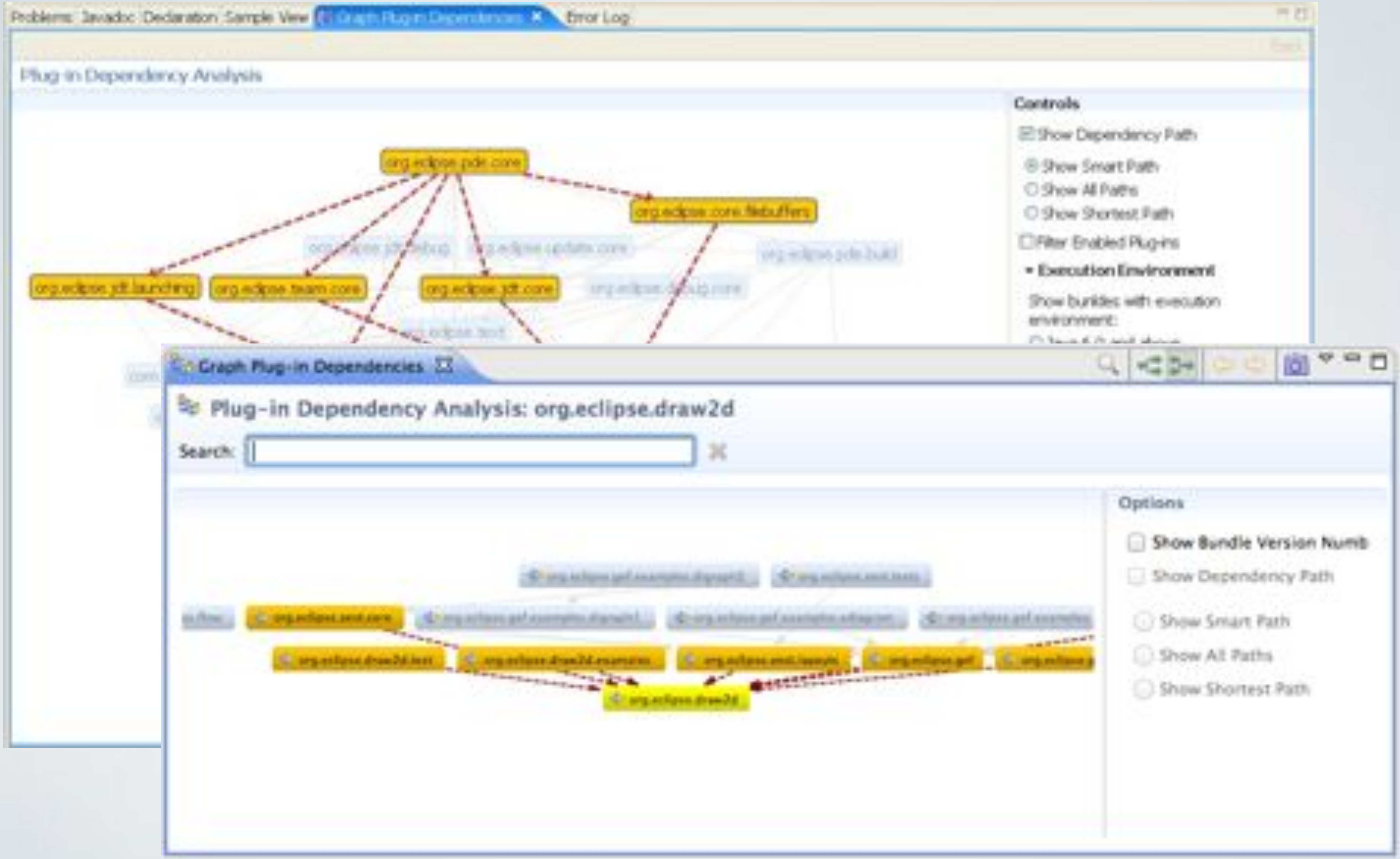


Zest

- Initial contribution of Zest by **University of Victoria** and **IBM Centre for Advanced Studies** as part of **Mylyn** in 2005.
- Joined in on GEF as **third component** with the **3.4 release** in **2007**.

> **Zest** - a visualization toolkit based on SWT and Draw2d to support the implementation of views with automatic or semi-automatic layout for the Eclipse Workbench UI.

Zest



Controls

☑ Show Dependency Path

● Show Smart Path
○ Show All Paths
○ Show Shortest Path

□ Filter Enabled Plug-ins

▾ Execution Environment

Show bundles with execution environment:

Options

□ Show Bundle Version Numb

□ Show Dependency Path

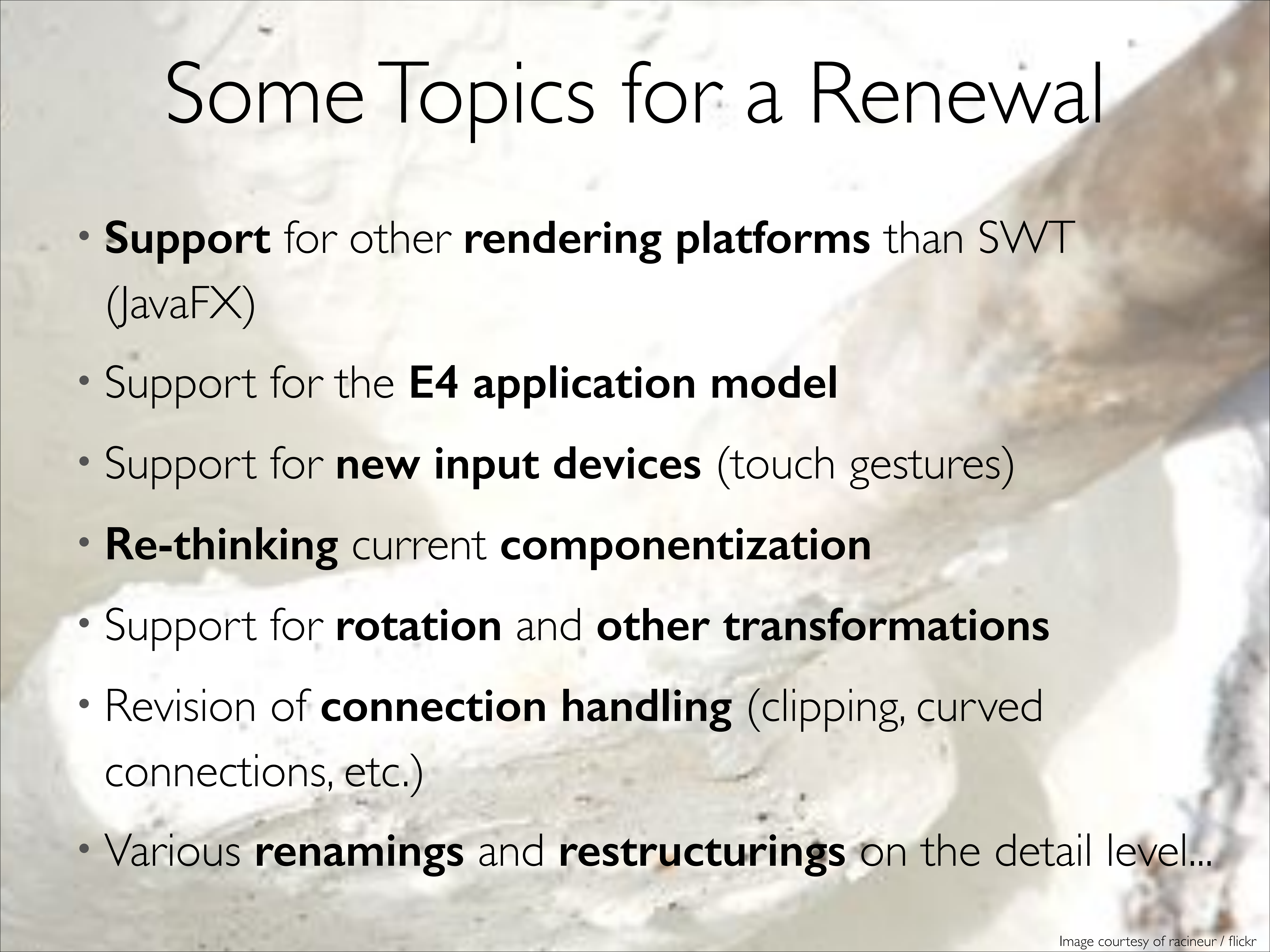
○ Show Smart Path
○ Show All Paths
○ Show Shortest Path

There is quite some decay…

- **API** is **organically evolved** and there are **~400 bugzillas**, out of which several would **require to break** it

Image courtesy of Pietro Izzo / flickr

Some Topics for a Renewal



- **Support** for other **rendering platforms** than SWT (JavaFX)
- Support for the **E4 application model**
- Support for **new input devices** (touch gestures)
- **Re-thinking** current **componentization**
- Support for **rotation** and **other transformations**
- Revision of **connection handling** (clipping, curved connections, etc.)
- Various **renamings** and **restructurings** on the detail level...

Image courtesy of racineur / flickr

Zest 2 (since 2010)

- A provisional **Zest 2** component was initiated in 2010, to develop the **next generation Zest API**.
- Goal was to develop a new version of Zest **in parallel to the maintenance of Zest 1.x.**, with a **provisional API**
- Sources were intended to be placed in its own **Zest2 Git repository**, results were published separately via **Eclipse Marketplace**.

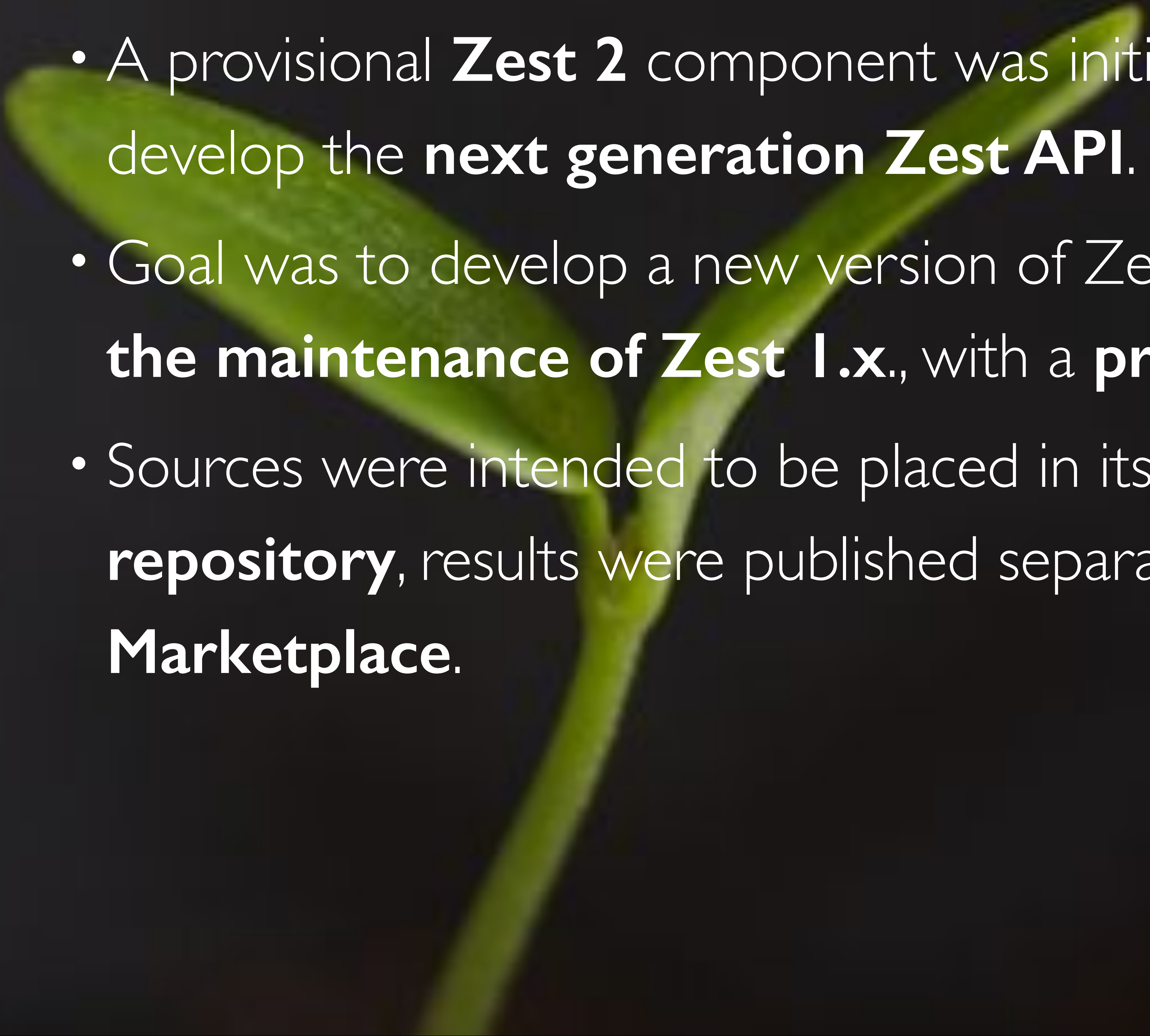
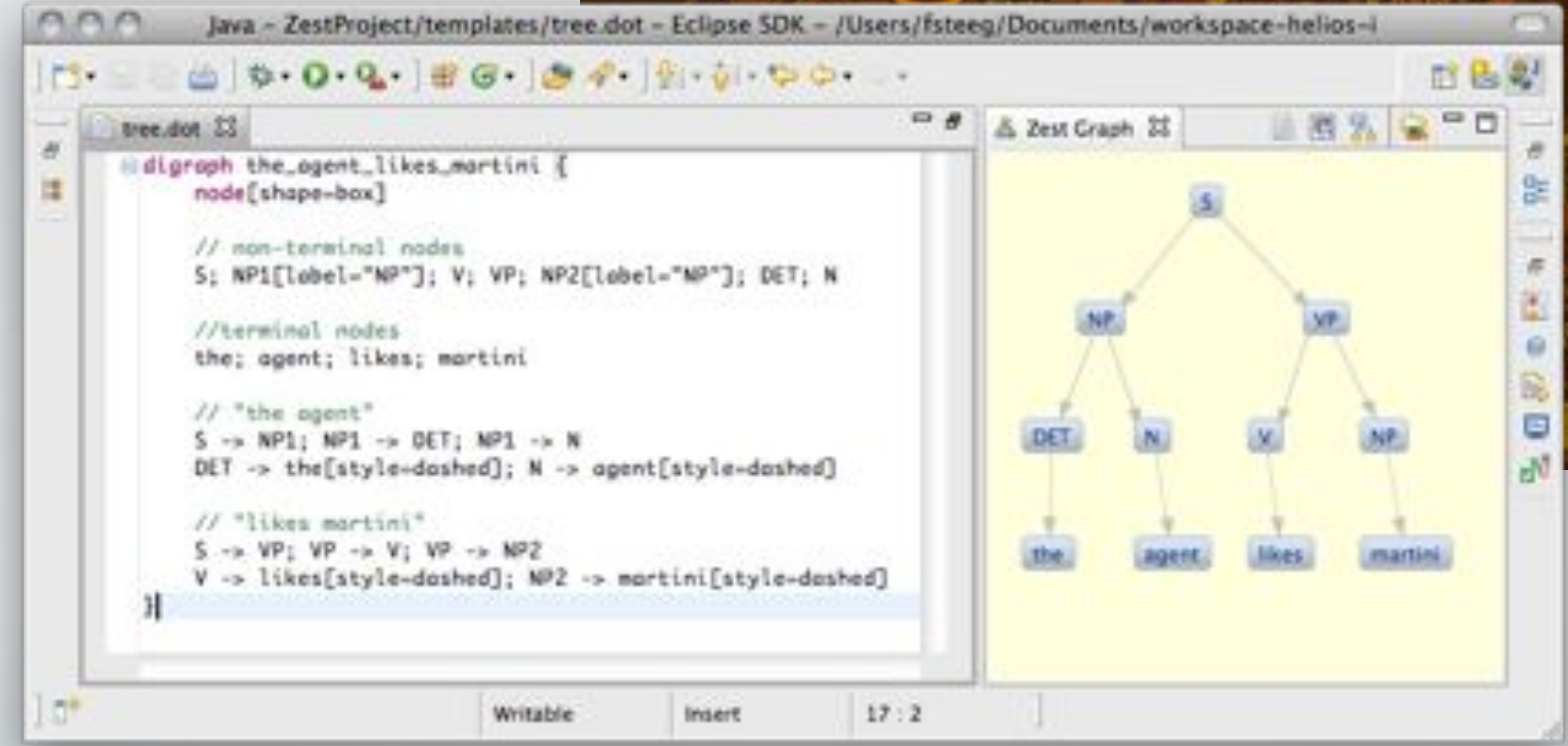


Image courtesy of __sjg__ / flickr

Zest 2 - New Features

- **Dot 4 Zest**
- **Cloudio**


Java – ZestProject/templates/tree.dot – Eclipse SDK – /Users/fsteeg/Documents/workspace-helios-i
tree.dot
Zest Graph
S
NP
VP
DET
N
V
NP
the
agent
likes
martini
Writable
Insert
17 : 2

```
digraph the_agent_likes_martini {
    node[shape=box]

    // non-terminal nodes
    S; NP1[label="NP"]; V; VP; NP2[label="NP"]; DET; N

    //terminal nodes
    the; agent; likes; martini

    // "the agent"
    S -> NP1; NP1 -> DET; NP1 -> N
    DET -> the[style=dashed]; N -> agent[style=dashed]

    // "likes martini"
    S -> VP; VP -> V; VP -> NP2
    V -> likes[style=dashed]; NP2 -> martini[style=dashed]
}
```

GEF4 (since 2011)

- **GEF4** was initiated - in analogy to Zest 2 - to **develop the next generation Draw2d and GEF (MVC) API**.
- Similar to Zest2, development was intended to take place **in parallel to maintenance** of **Draw2d / GEF (MVC) 3.x**

- Initial plans (prior to 3.8):
  - Create **new double-precision Geometry API** before Juno release.
  - Start to **migrate** the Draw2d and GEF (MVC) **code base** on a step-by-step basis afterwards.

Image courtesy of —Tico— / flickr

GEF4 + Zest 2 = GEF4

Image courtesy of spadger / flickr

GEF4

- A **unified** provisional **approach** to **develop** the **next generation API**.
- Development takes place **in parallel to maintenance** of **GEF proper** (Draw2D/GEF 3.x / Zest 1.x)

- **Advantages** of this procedure:
  - Clear **distinction** between **GEF proper** as the production and **GEF4** as the provisional component
  - Chance to not only **refactor** GEF components but the **componentization** itself, which is only "historically" justified.

Image courtesy of edtechie99 / flickr

Status Quo (a year ago)

- **GEF4 Geometry** was finalized before Juno
- **GEF4 Graphics** was initiated before Kepler
  - Idea was to provide a common graphics abstraction over SWT/AWT, and also JavaFX
- **GEF4 ‚Glyphs' (SwtFX)** was planned:
  - Figures/Shapes abstractions inspired by Draw2d, SVG, and JavaFX (SceneGraph)
  - Intended as replacement of Draw2d ‚Core'
- **Zest2** had been transferred to namespace and repository.

Image courtesy of pareeerica / flickr

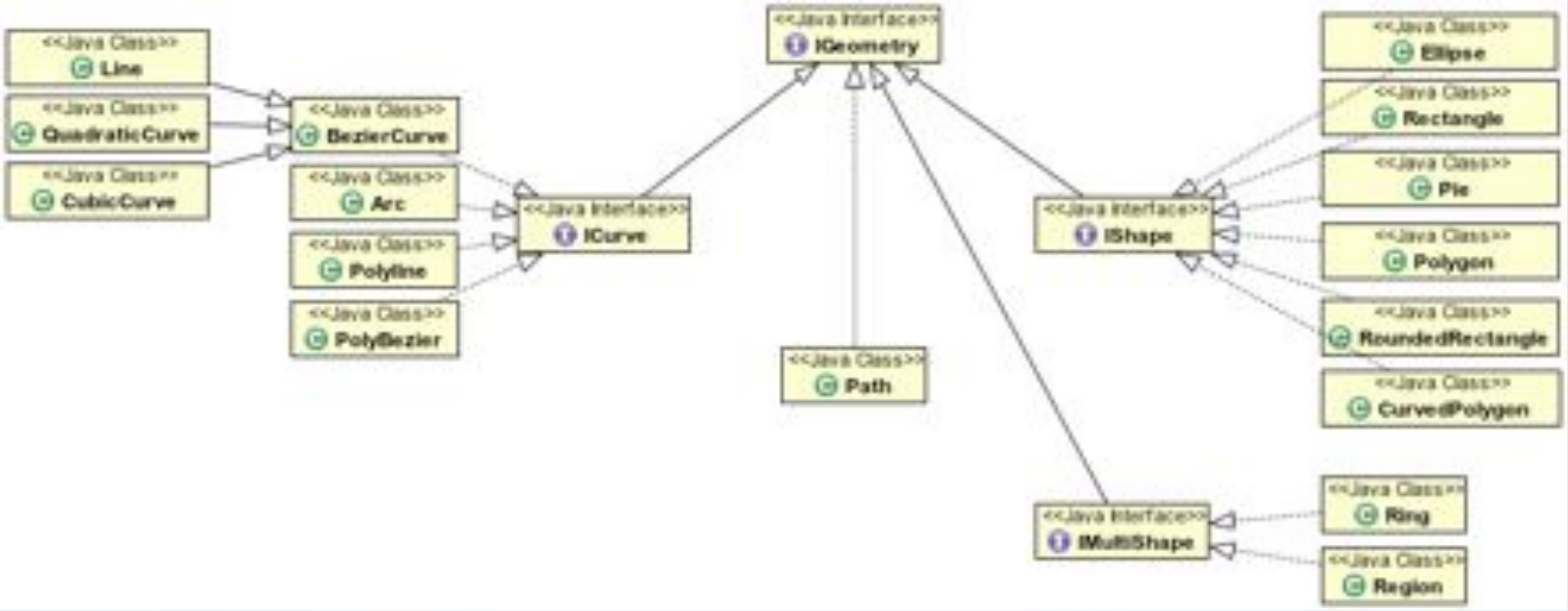
GEF4 Geometry

- No distinction in low and high precision, but just a single **double-precision API** (with **built-in imprecision** for comparisons).

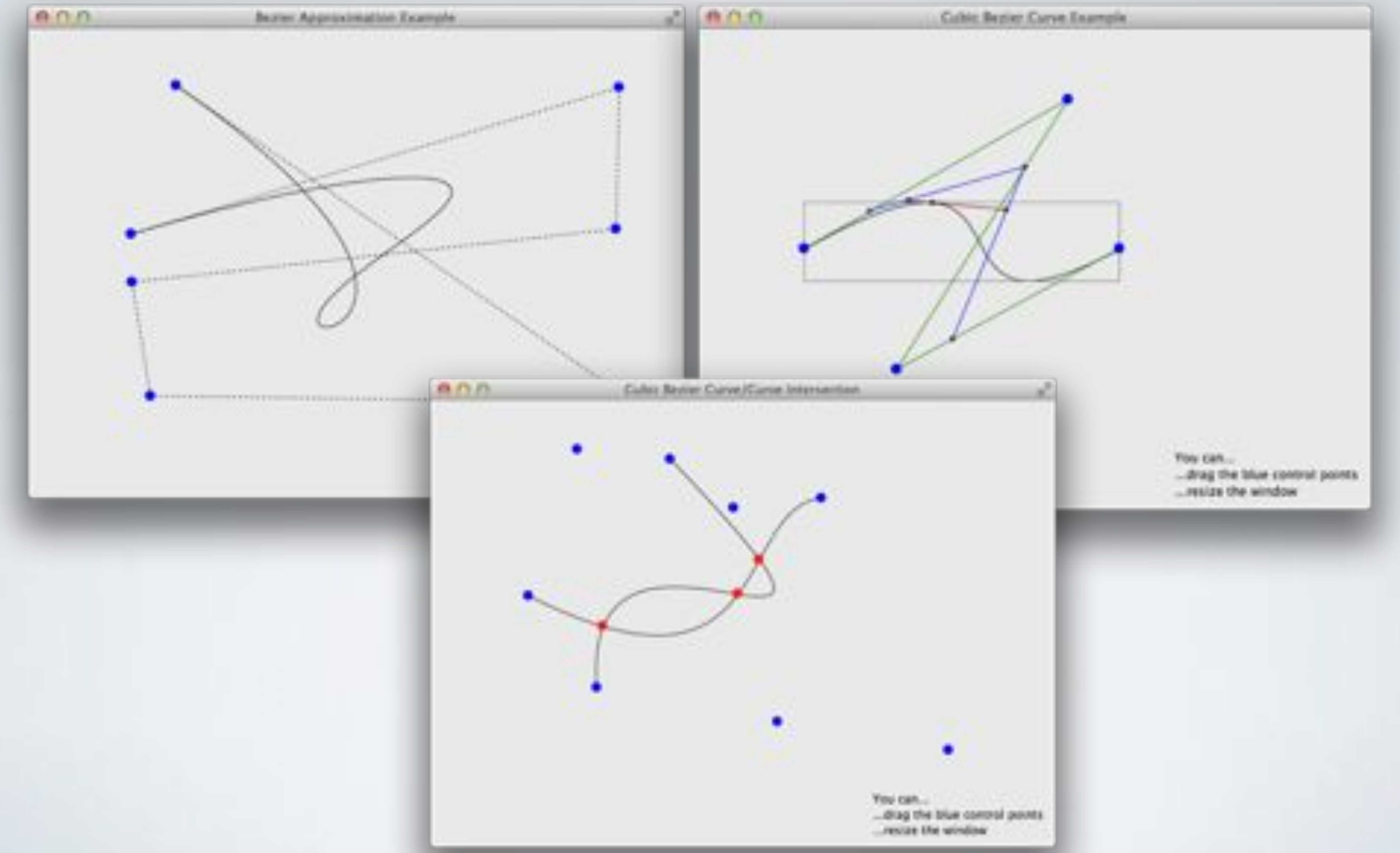
- **Different geometric abstractions** for different purposes:
  - **Euclidean** (Vector, Straight, Angle)
  - **Projective** (Vector3D, Straight3D)
  - **Planar** (Point, Dimension, Line, QuadraticCurve, CubicCurve, BezierCurve, Polyline, PolyBezier, Ellipse, Rectangle, Pie, Arc, Polygon, CurvedPolygon, RoundedRectangle, Ring, Region, Path)

- **Conversions** to/from **AWT** and **SWT** (and between them)

GEF4 Planar Geometry - Overview



GEF4 Geometry - Examples


Bezier Approximation Example
Cubic Bezier Curve Example
Cubic Bezier Curve/Curve Intersection

You can...
...drag the blue control points
...resize the window

You can...
...drag the blue control points
...resize the window

Status Quo (now)

- **GEF4 Geometry** further matured
- **GEF4 Cloudio** extracted from **GEF4 Zest** code base, still based on SWT/JFace (and GEF 3.x Draw2d)
- **GEF4 SwtFX** under development; initial prototype being revised
- **GEF4 MVC** initiated, based on JavaFX, later also on SwtFX

Image courtesy of pareeerica / flickr

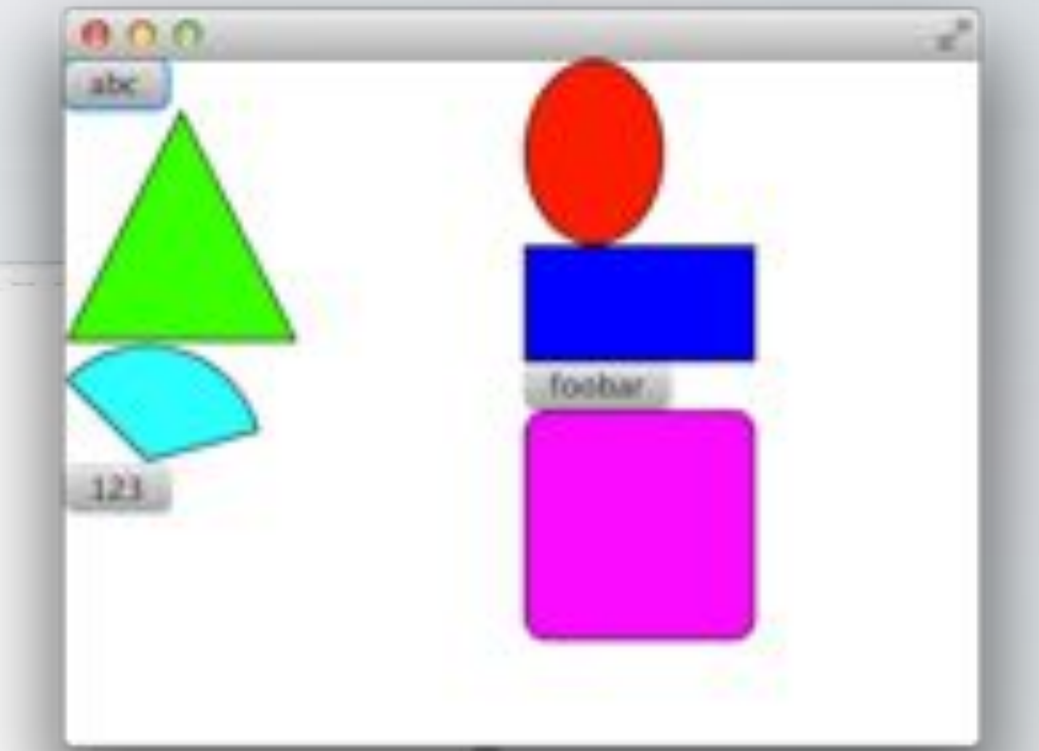
GEF4 SwtFX - Initial Scope

- **Replacement** of **Draw2d** 'Core', making use of **JavaFX key abstractions** (Scene, Parent)
- **Combination of heavyweight** (SWT Controls) and **lightweight nodes** (Figures) **in** a single **scene graph**, rendered on a SWT Canvas
  - **SWT Controls** wrapped into adapters
  - **ShapeFigure** based on GEF4 Geometry planar shapes
  - **CanvasFigure** provides a lightweight node that allows painting on a Graphics*

*) based on former GEF4 Graphics code

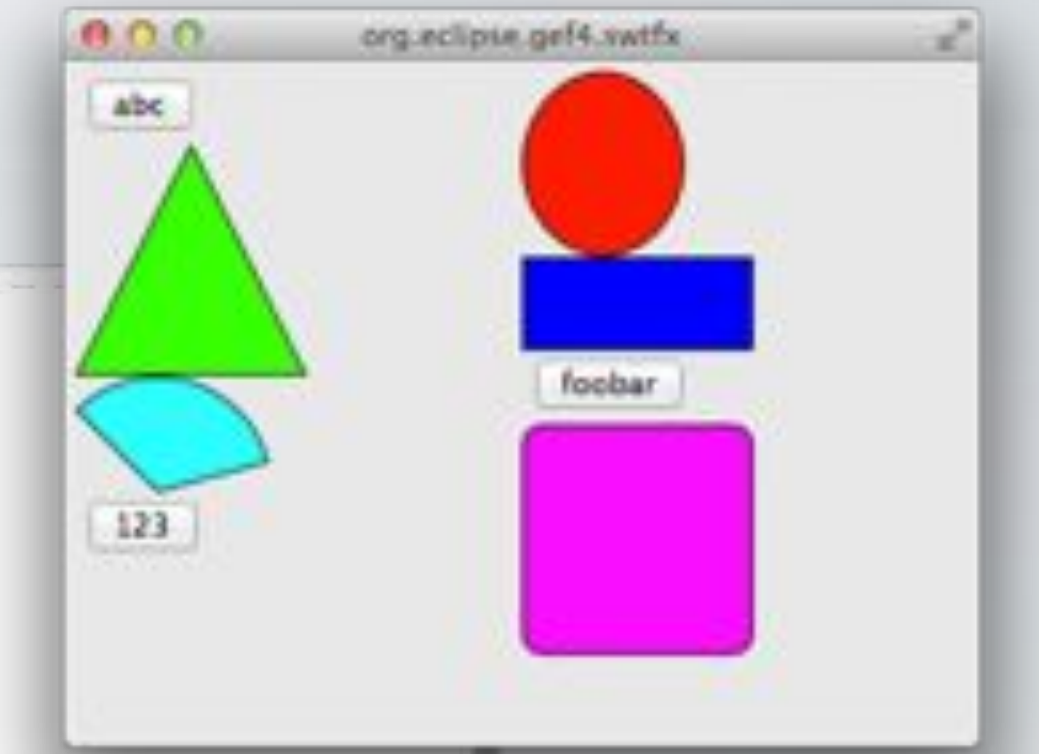
GEF4 SwtFX - Sample Code

```
HBox hbox = new HBox();
VBox col1 = new VBox();
VBox col2 = new VBox();
hbox.getChildren().addAll(col1, col2);
HBox.setHgrow(col1, Priority.ALWAYS);
HBox.setHgrow(col2, Priority.ALWAYS);

col1.getChildren().addAll(
  new Button("abc"),
  shape(new Polygon(50, 0, 100, 100, 0, 100), 0, 1, 0),
  shape(new Arc(0, 0, 50, 50, 15, 120) {{ setType(ArcType.ROUND); }}, 0, 1, 1),
  new Button("123"));

col2.getChildren().addAll(
  shape(new Ellipse(30, 40, 30, 40), 1, 0, 0),
  shape(new Rectangle(0, 0, 100, 50), 0, 0, 1),
  new Button("foobar"),
  shape(new Rectangle(0, 0, 100, 100) {{ setArcHeight(20); setArcWidth(20); }}, 1, 0, 1));

// create scene (and set scene size)
Scene scene = new Scene(hbox, 400, 300);
stage.setScene(scene);
stage.show();
```

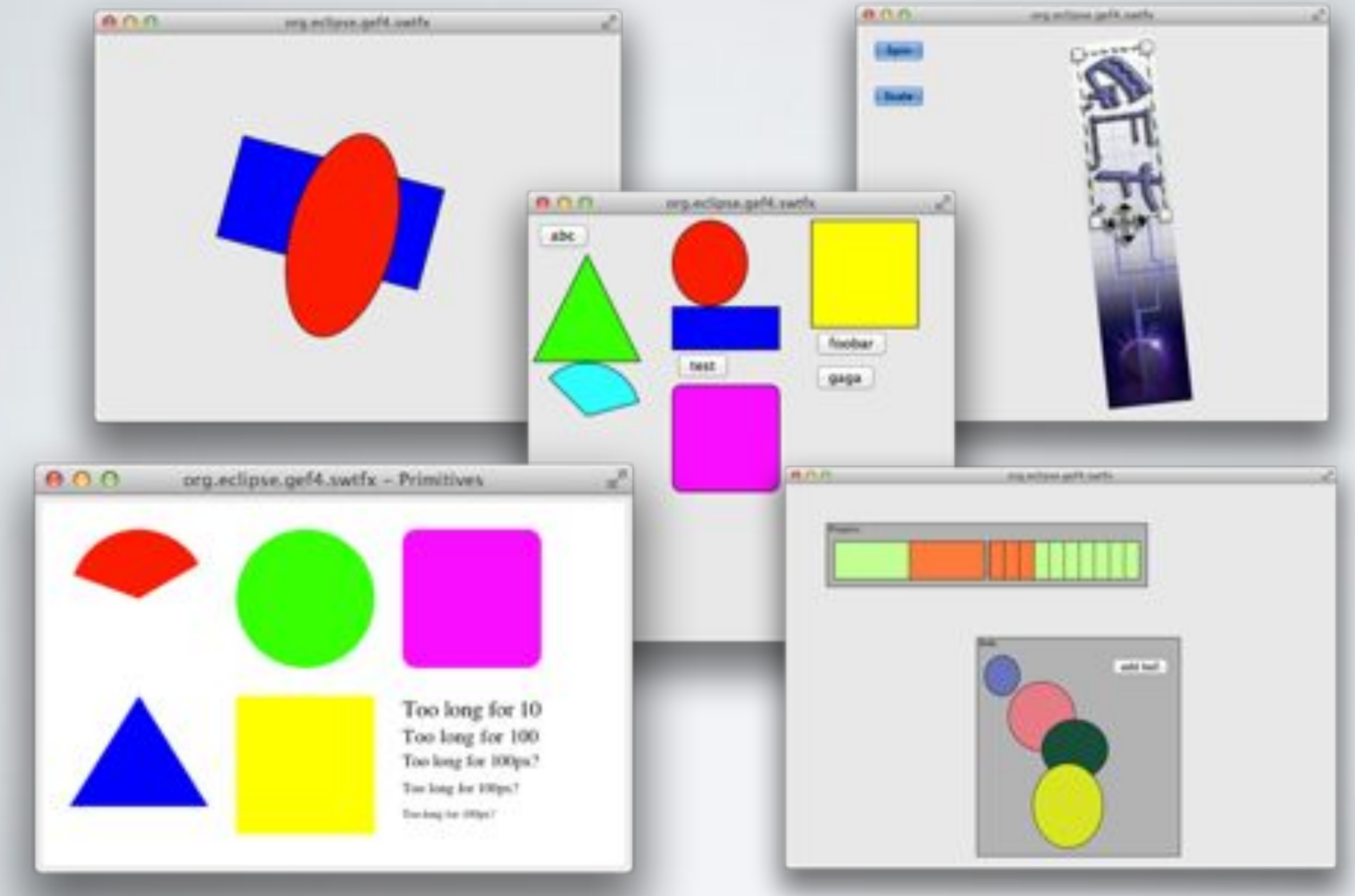


GEF4 SwtFX - Sample Code

```
HBox root = new HBox();
VBox col1 = new VBox();
VBox col2 = new VBox();
root.addChildNodes(col1, col2);



col1.addChildNodes(
  new SwtButton("abc"),
  shape(new Polygon(50, 0, 100, 100, 0, 100), 0, 1, 0),
  shape(new Pie(0, 0, 100, 100, Angle.fromDeg(15), Angle.fromDeg(120)), 0, 1, 1),
  new SwtButton("123"));

col2.addChildNodes(
  shape(new Ellipse(0, 0, 70, 80), 1, 0, 0)
  shape(new Rectangle(0, 0, 100, 40), 0, 0, 1),
  new SwtButton("foobar"),
  shape(new RoundedRectangle(0, 0, 100, 100, 10, 10), 1, 0, 1));

// set root size and create scene
root.setPrefWidth(400);
root.setPrefHeight(300);
return new Scene(shell, root);
```

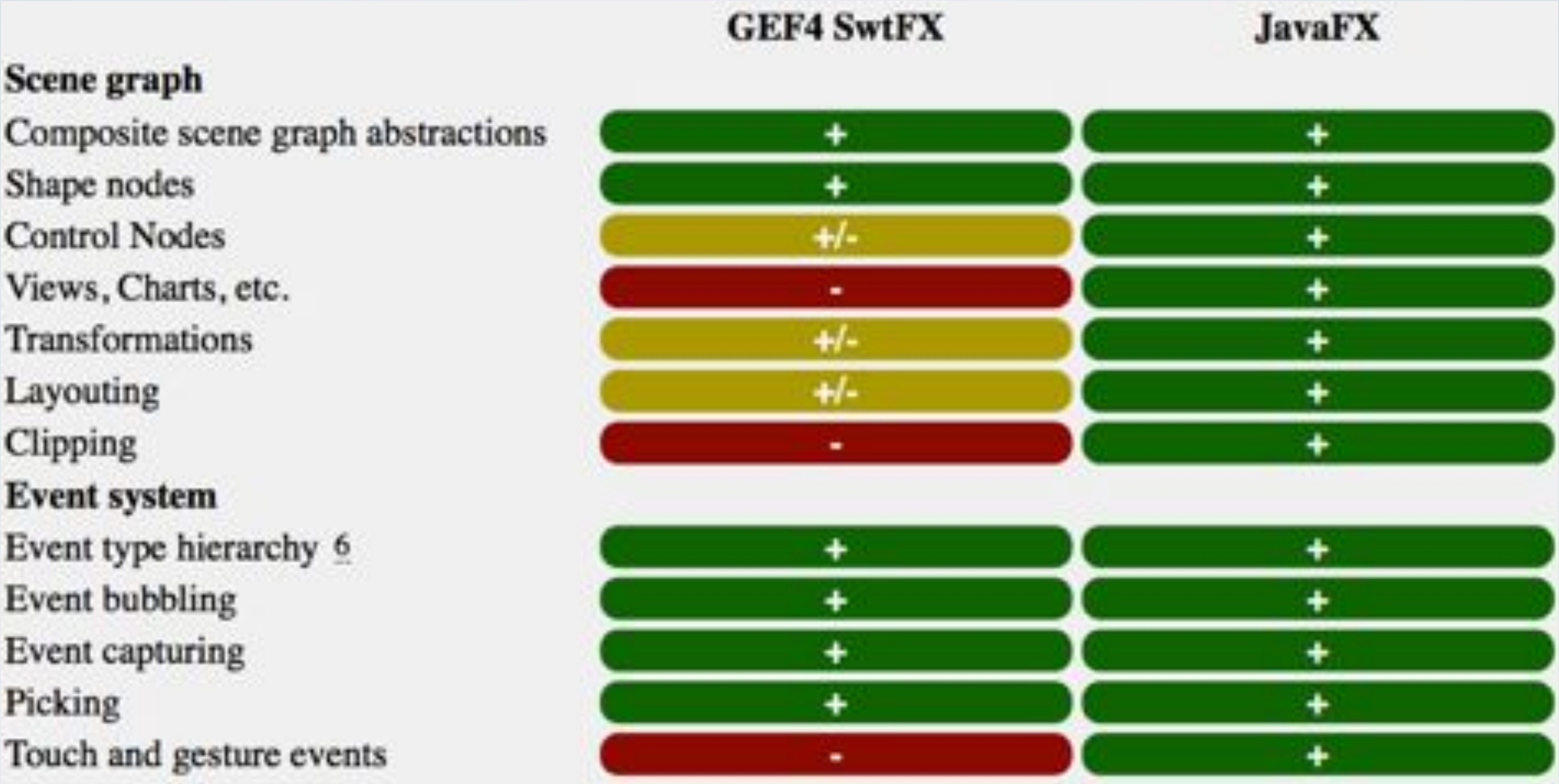


(SELF-) DEMO - GEF4 SwtFX Examples*



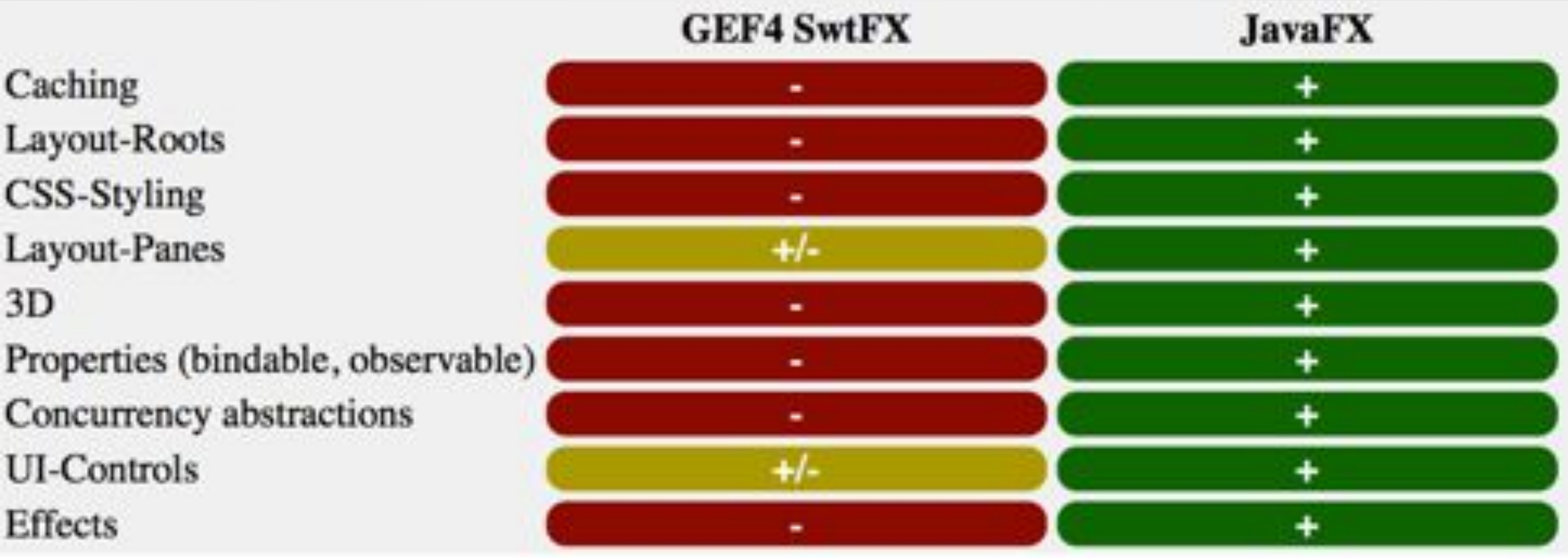
*) Code base is available via Git tag "replica"

GEF4 SwtFX - Initial Prototype Features



	GEF4 SwtFX	JavaFX
Scene graph		
Composite scene graph abstractions	+	+
Shape nodes	+	+
Control Nodes	+/-	+
Views, Charts, etc.	-	+
Transformations	+/-	+
Layouting	+/-	+
Clipping	-	+
Event system		
Event type hierarchy [6]	+	+
Event bubbling	+	+
Event capturing	+	+
Picking	+	+
Touch and gesture events	-	+

GEF4 SwtFX - Initial Prototype Limitations



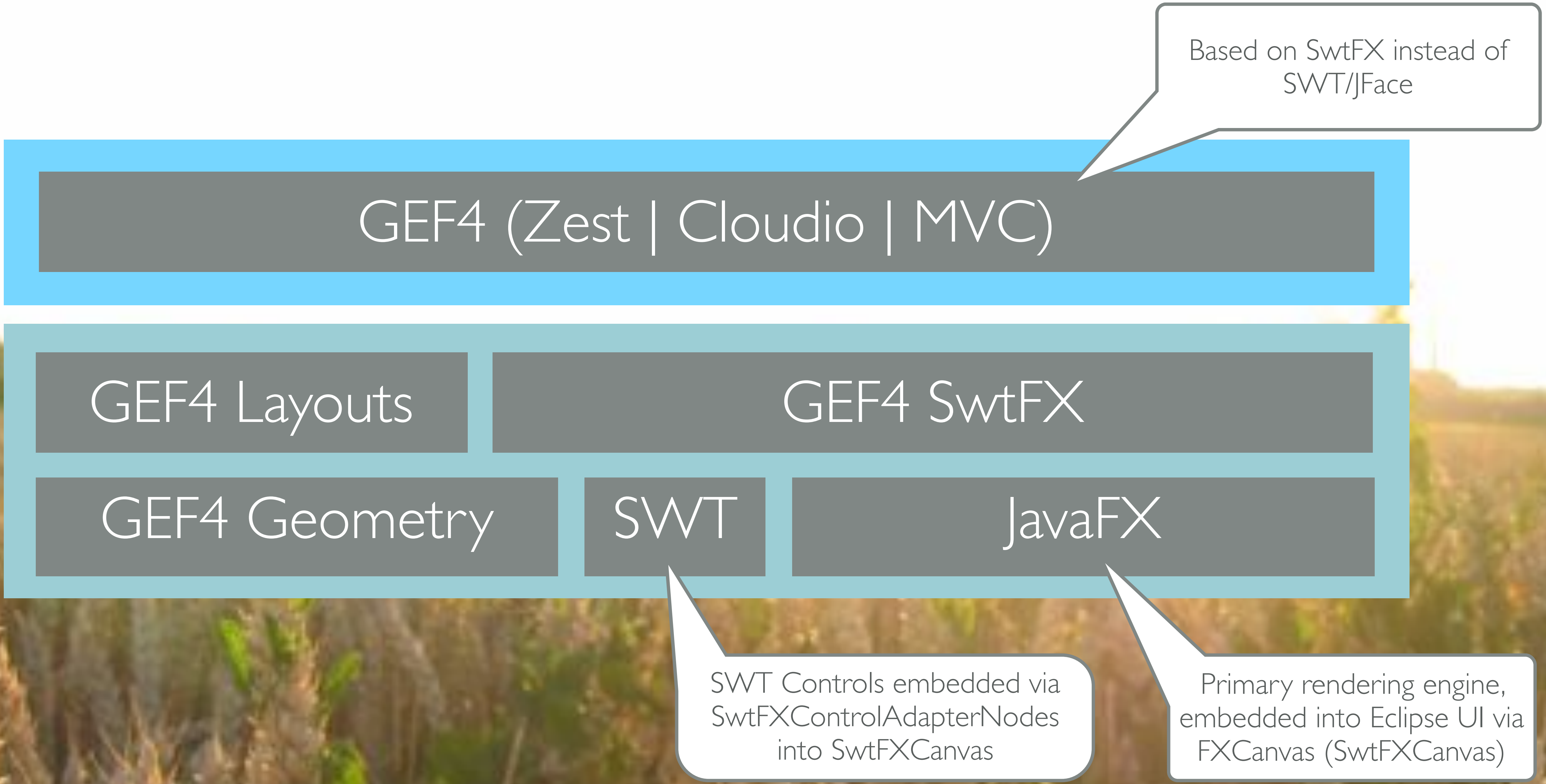
	GEF4 SwtFX	JavaFX
Caching	-	+
Layout-Roots	-	+
CSS-Styling	-	+
Layout-Panes	+/-	+
3D	-	+
Properties (bindable, observable)	-	+
Concurrency abstractions	-	+
UI-Controls	+/-	+
Effects	-	+

GEF4 SwtFX - Revised Scope & Future Plans

- **Migrate SwtFX** from a functionally limited alternative of JavaFX **into an extension**:
  - **Use JavaFX to render everything except SWT controls**, i.e. specialize javafx.embed.swt.FXCanvas (SwtFXCanvas) and Scene (SwtFXScene) to transparently integrate SWT Controls via SwtFXAdapterNodes.
- **Add JFace-like abstractions** (viewer), so it can used as a base layer for GEF4 Zest/Cloudio, etc.

Image courtesy of Ruben Holthuijsen / flickr

GEF4 - My (Current) Vision



Based on SwtFX instead of SWT/JFace

GEF4 (Zest | Cloudio | MVC)

GEF4 Layouts

GEF4 SwtFX

GEF4 Geometry

SWT

JavaFX

SWT Controls embedded via SwtFXControlAdapterNodes into SwtFXCanvas

Primary rendering engine, embedded into Eclipse UI via FXCanvas (SwtFXCanvas)

Image courtesy of AndiH / flickr

Please get involved!

- **Evaluate and Provide Feedback!**
  - Try out early snapshots!
  - Report bugs, report enhancement requests!
- **Contribute!**
  - Participate in discussions (bugzilla, mailing list)
  - Supply patches

Image courtesy of EricMagnuson / flickr

Thank You! Questions?

http://wiki.eclipse.org/GEF/GEF4

Image courtesy of Bilal Kamoon / flickr