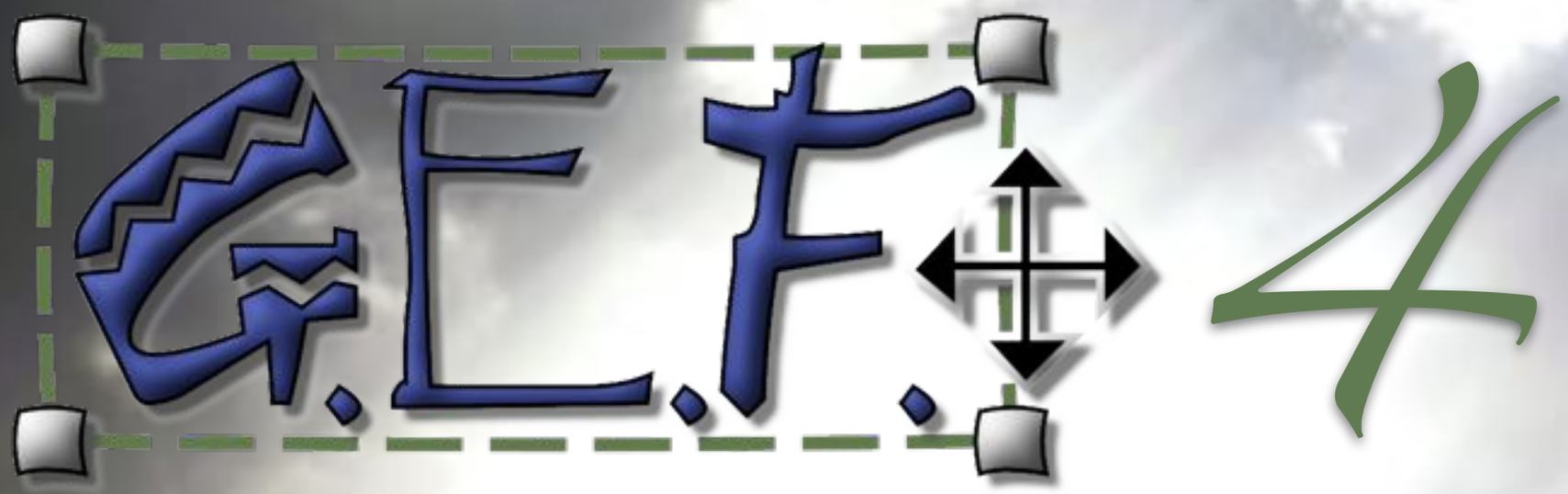


4

There's really something going on!

Alexander Nyßen
itemis AG
Graphical Editing Framework Project Lead

Image courtesy of Julie Lafrance / flickr

GEF celebrated 10th Birthday in 2012!

Initial contribution by IBM in 2002

Image courtesy of Will Clayton / 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

But, there is quite some decay…

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

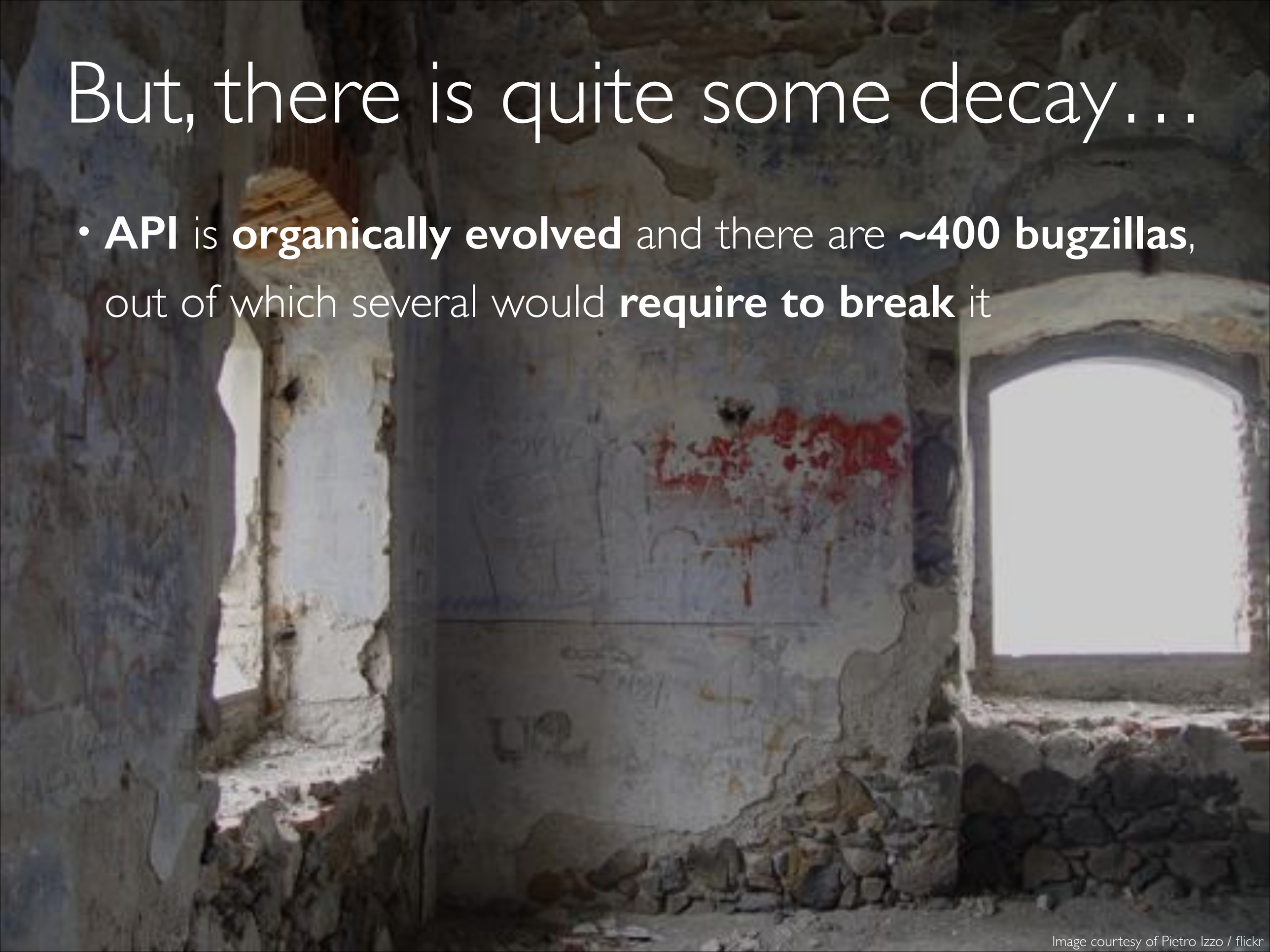


Image courtesy of Pietro Izzo / flickr

Some Topics for a Renewal

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

GEF4

- Our **approach** to **develop** the **next generation API**
- Development takes place **in parallel to maintenance** of **GEF proper** (Draw2D/GEF 3.x / Zest 1.x), **API is provisional** until a final replacement

- **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 **modularization** itself, which is only "historically" justified.

Concrete Goals for GEF4

- A **technically** (and not historically) **justified modularity**
- Use of **modern rendering technology**
- Being **lightweight,** i.e. **no Eclipse-UI dependencies** where avoidable
- Integration of **automatic layouts** also in editors
- Support of **touch-gestures**
- Support for **„classical“** and **„handheld-like“ look & feel**

GEF4 - Vision

Based on JavaFX, Integration into Eclipse UI via SwtFX

UI-toolkit independent

Based on SWT, Direct Integration into Eclipse UI

GEF4 Zest.FX.UI

GEF4 MVC.FX.UI ★

GEF4 MVC.UI ★

GEF4 Cloudio

GEF4 Zest.FX

GEF4 MVC.FX ★

Eclipse UI

GEF4 Layout

GEF4 Graph/ DOT

GEF4 MVC ★

GEF4 FX ★

GEF4 SwtFX ★

JFace

JavaFX

SWT

GEF4 Geometry ★

UI-toolkit independent

UI ≃ Eclipse UI

FX ≃ JavaFX

Image courtesy of AndiH / flickr

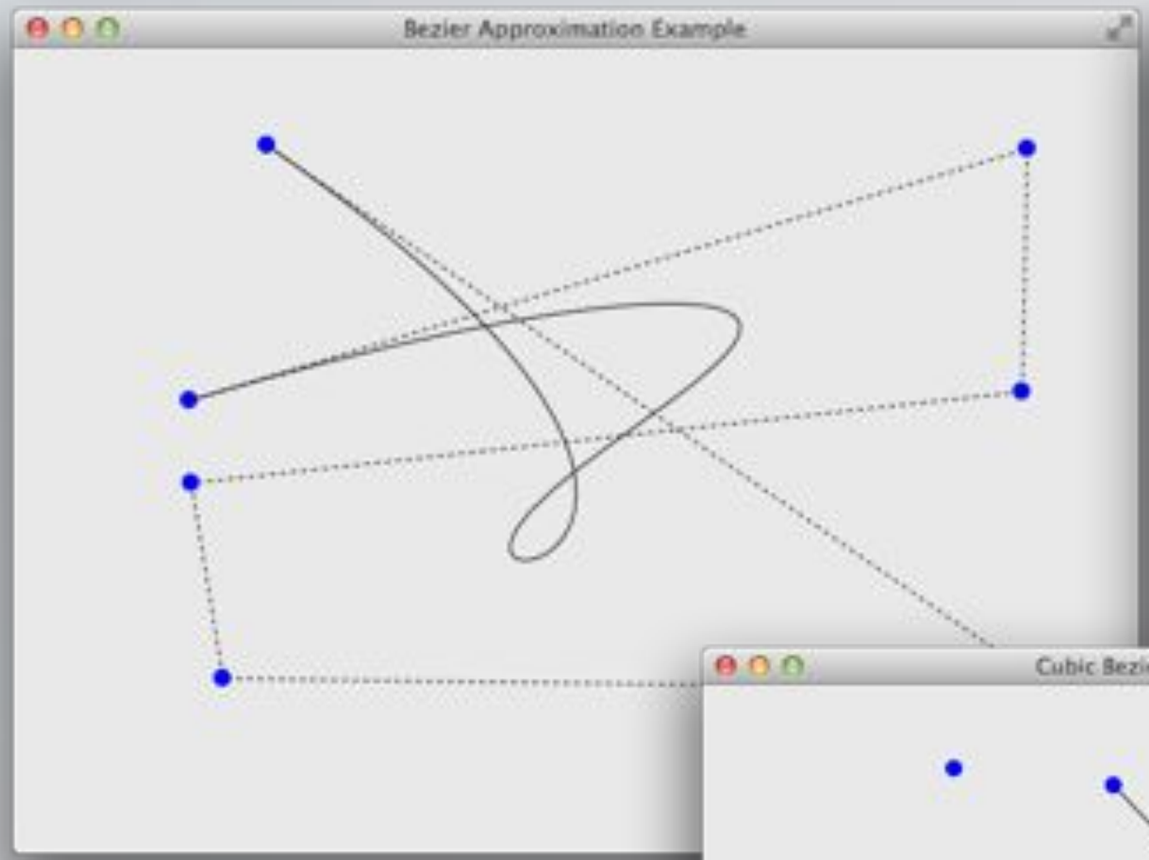
GEF4 Geometry

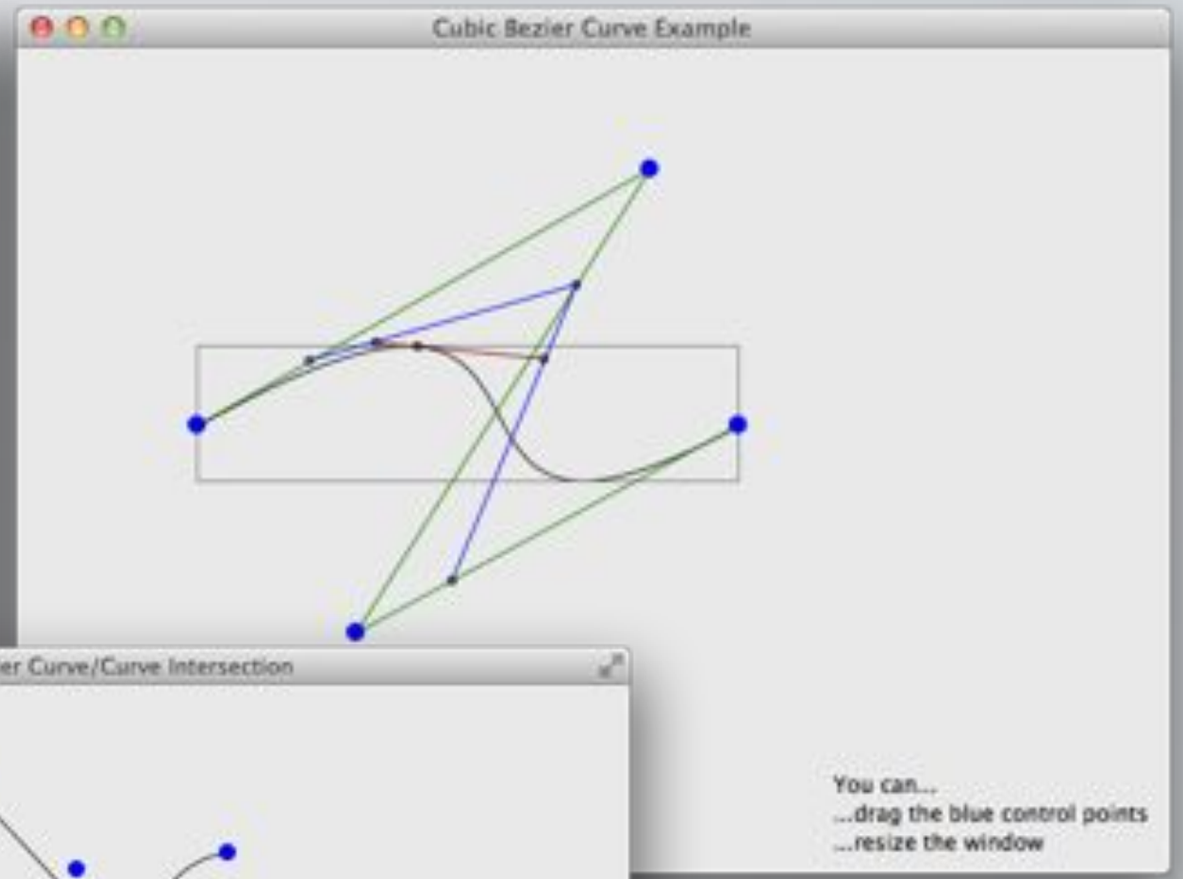
- No distinction in low and high precision, but just a single **double-precision API** (with **built-in imprecision** for comparisons).

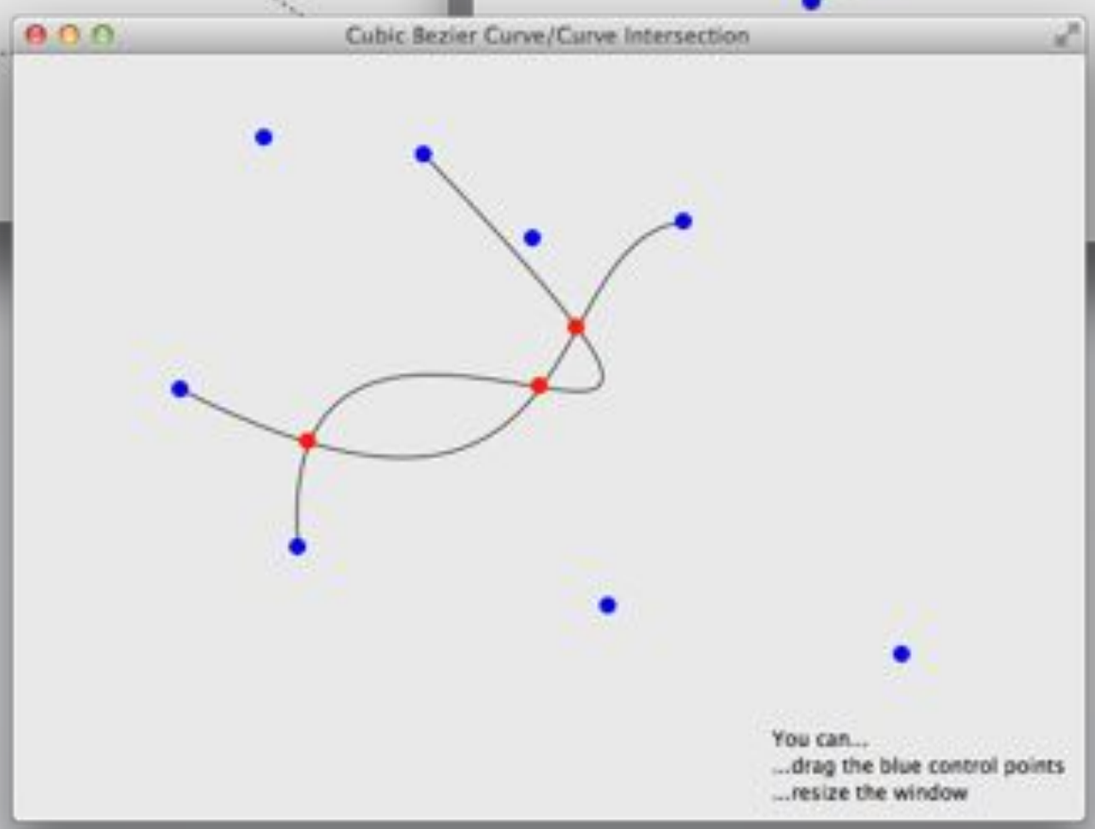
- **Different geometric abstractions** for different purposes:
  - **Euclidean** (Vector, Straight, Angle)
  - **Projective** (Vector3D, Straight3D)
  - **Planar** (Point, Dimension, Line, QuadraticCurve, CubicCurve, BezierCurve, Polyline, PolyBezier, Ellipse, Rectangle, Pie, Arc, Polygon, CurvedPolygon,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

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

GEF4 Geometry - Examples







GEF4 FX

- Provides **JavaFX-related additions** to be used in upstream components like MVC.FX or Zest.FX:
  - **FXGeometryNode** to create IGeometry-based shapes
  - **IFXAnchor** abstraction and implementations (static, chopbox)
  - **IFXConnection** abstraction with default **FXCurveConnection** implementation, based on FXGeometricNode<ICurve>.
  - **Gesture-/Compound-Listener** support

- Future additions: **Decorators**, **Widgets**, …

GEF4 FX - Example

```
public Scene createScene() {
  FXGeometryNode<CurvedPolygon> eLetterShape = new FXGeometryNode<CurvedPolygon>(createEShapeGeometry());
  eLetterShape.setTranslateX(25);
  eLetterShape.setTranslateY(25);
  eLetterShape.resize(200, 250);
  eLetterShape.setEffect(GEF_SHADOW_EFFECT);
  eLetterShape.setFill(GEF_COLOR_BLUE);

  HBox hbox = new HBox();
  hbox.getChildren().add(eLetterShape);
  return new Scene(hbox, 250, 300);
}

private CurvedPolygon createEShapeGeometry() {
  List<BezierCurve> segments = new ArrayList<BezierCurve>();
  segments.add(new Line(1, 10, 6, 10));
  segments.addAll(Arrays.asList(PolyBezier.interpolateCubic(6, 10, 5, 25, 7, 52, 6, 70, 6, 81).toBezier()));
  segments.addAll(Arrays.asList(PolyBezier.interpolateCubic(6, 81, 5, 81, 3, 84).toBezier()));
  segments.add(new Line(3, 84, 3, 87));
  segments.add(new Line(3, 87, 64, 86));
  segments.add(new Line(64, 86, 65, 79));
  segments.addAll(Arrays.asList(PolyBezier.interpolateCubic(65, 79, 59, 81, 51, 82).toBezier()));
  segments.add(new Line(51, 82, 12, 82));
  segments.addAll(Arrays.asList(PolyBezier.interpolateCubic(12, 82, 11, 56, 11, 30).toBezier()));
  segments.addAll(Arrays.asList(PolyBezier.interpolateCubic(11, 30, 27, 30, 45, 31).toBezier()));
  segments.add(new Line(45, 31, 48, 25));
  segments.addAll(Arrays.asList(PolyBezier.interpolateCubic(48, 25, 35, 27, 19, 27, 10, 26).toBezier()));
  segments.addAll(Arrays.asList(PolyBezier.interpolateCubic(10, 26, 10, 20, 11, 10).toBezier()));
  segments.addAll(Arrays.asList(PolyBezier.interpolateCubic(11, 10, 24, 11, 31, 11, 51, 12).toBezier()));
  segments.add(new Line(51, 12, 55, 6));
  segments.addAll(Arrays.asList(PolyBezier.interpolateCubic(55, 6, 45, 7, 33, 8, 15, 7, 7, 6).toBezier()));
  segments.add(new Line(7, 6, 1, 10));
  return new CurvedPolygon(segments);
}
```

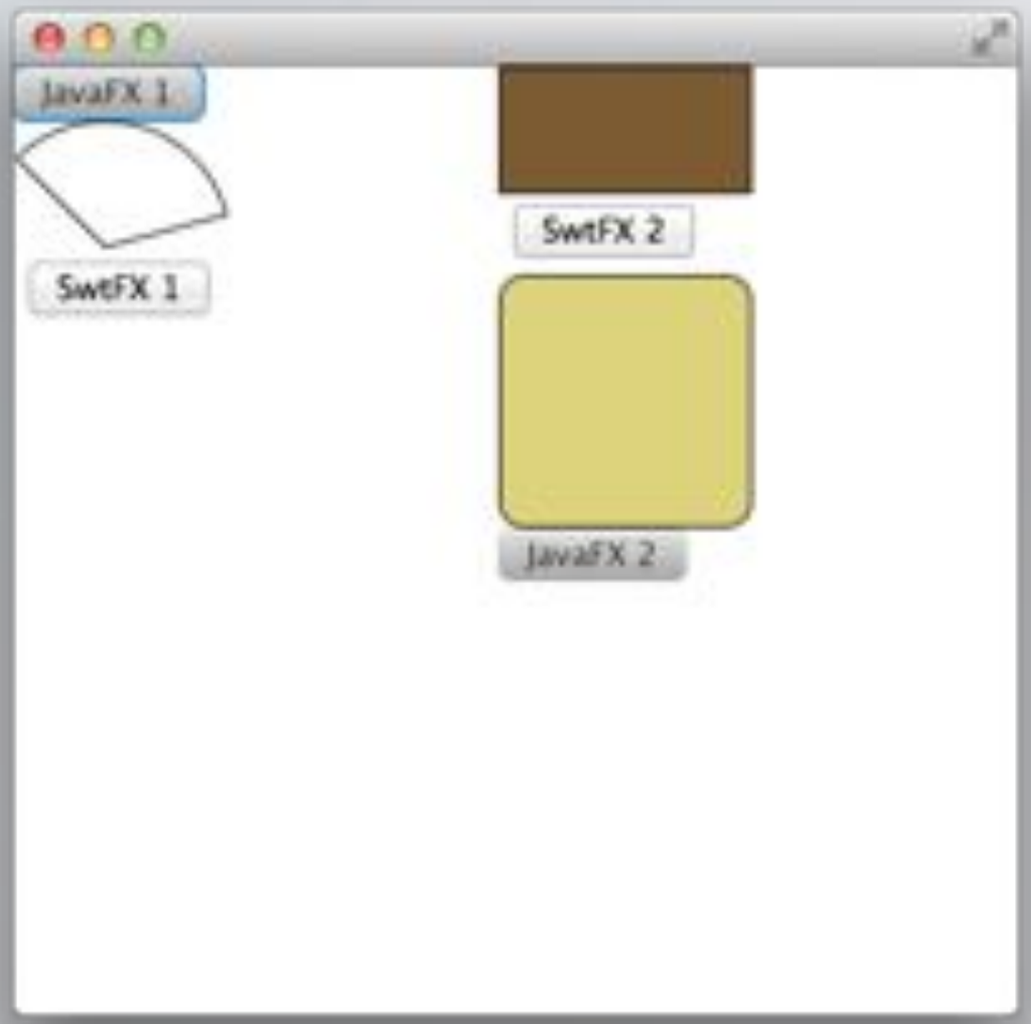
GEF4 SwtFX

- Provides **SwtFXCanvas** as a specialization of FXCanvas (javafx.embed.swt), which provides:
  - **forwarding of SWT touch gesture events** to JavaFX (SwtToFXGestureConverter)*
  - transparent **integration of SWT Controls** into a JavaFX scene (SwtFXScene)

- Future: **Support additional SWT controls** and **Z-ordering**

*) Contribution of J. Köhnlein

GEF4 SwtFX - Example




```
public SwtFXScene createScene() {
  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("JavaFX 1"),
    shape(new Arc(0, 0, 50, 50, 15, 120) {
      {
        setType(ArcType.ROUND);
      }
    }, 0.52, 0.49, 0.15), new SwtFXButton("SwtFX 1"));

  col2.getChildren().addAll(
    shape(new Rectangle(0, 0, 100, 50), 0.49, 0.36, 0.20),
    new SwtFXButton("SwtFX 2"),
    shape(new Rectangle(0, 0, 100, 100) {
      {
        setArcHeight(20);
        setArcWidth(20);
      }
    }, 0.87, 0.83, 0.49),
    new Button("JavaFX 2"));
  return new SwtFXScene(hbox, 400, 400);
}

private static Shape shape(Shape shape, double r, double g, double b) {
  shape.setFill(new Color(r, g, b, 1));
  shape.setStroke(new Color(0, 0, 0, 1));
  return shape;
}
```

GEF4 MVC

- **Dedicated** to **graphical editors** and **views** (no tree support)
- **Intentionally light-weight:**
  - Split into **UI-toolkit-independent** abstractions (MVC), **JavaFX-based specializations** (MVC.FX), and related **Eclipse UI-integration** (MVC.UI/MVC.FX.UI)
- **Transfers** (but revises) **core concepts** of GEF (MVC) 3.x.:
  - **Controller** (IVisualPart) **hierarchy** with explicit parts for **content**, **feedback**, and **handles** (IContentPart, IHandlePart, IFeedbackPart).
  - **Modularized** interaction **behavior** (ITool, IBehavior, IPolicy) with **aspect-bound** interfaces (FXDragTool, AbstractFXDragPolicy)

DEMO - GEF4 MVC.FX.UI Example

GEF4 MVC - Details

- **No** dedicated **connection layer,** instead dedicated **layers** for **contents**, **feedback**, and **handles**
- **No** dedicated **connection parts,** but **parent↔child** and/or **anchorage↔anchored** relationships
- **Accessible viewer/interaction state** via **explicit models** (ISelectionModel, IContentModel, IZoomModel, …)
- **No** own **command-framework** but direct integration with IUndoableOperationHistory
- **Continuous interaction feedback** and **nice look & feel** (via GEF4 FX and JavaFX)

Status Quo

- **GEF4 Geometry** (✓)
  - Is already quite stable, but still requires some performance optimizations and will probably be extended to provide change notification support.
- **GEF4 FX** (✓)
  - Will have to be extended with additional anchor implementations, decorations, and custom widgets.
- **GEF4 SwtFX** (✓)
  - Needs to be matured and dedicated support for additional SWT controls will have to be added.
- **GEF4 MVC / MVC.UI / MVC.FX / MVC.FX.UI** (✓)
  - Already provides the basics that are needed to build up graphical viewers, interaction support however is just being built-up.

Status Quo (continued)

- **GEF4 Cloudio** (✓)
  - A word-cloud-viewer based on SWT/JFace, integrated in the Eclipse UI, currently not intended to be ported to JavaFX
- **GEF4 Graph** (✓)
  - Provides a very simply data-model (Graph, Node, Edge). Will need to be extended to suit as underlying data model for GEF4 Layout (sub-graphs)
- **GEF4 Layout** (✓)
  - Provides data-model facade and layout implementations. Will have to be internally refactored and harmonized with GEF4 Graph
- **GEF4 DOT** (✓)
  - Provides GraphViz DOT-Editor (Xtext) and import/export to GEF4 Graph. Will have to be extended to e.g. support sub-graphs.
- **GEF4 Zest.FX / Zest.FX.UI** (✗)
  - Encapsulates the original Zest2 code base without those parts already extracted into GEF4 Layout, GEF4 Graph, GEF4 DOT, and GEF4 Cloudio, only adopted to GEF4 namespace

Future Plans - Roadmap

- **Extend functionality** of **FX**, **SwtFX**, and **MVC** components to close the remaining gap to Draw2d/GEF (MVC) 3.x.
- **Complete refactoring** of **Layout**, **Graph**, and **Zest** components and **built-up Zest.FX** and **Zest.FX.UI** components so GEF4 is fully self-contained.
- **Join Mars release** train **with** a **first** initial **release** of GEF4 components, based on yet **provisional API**.

Please get involved!

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

Image courtesy of EricMagnuson / flickr

Evaluate This Session

1. Sign-in: www.eclipsecon.org

2. Select session from schedule

3. Evaluate: