CIB GROUP

DRAWINGLAYER PRIMITIVES WORKSHOP

WHAT THEY ARE, HOW TO USE OR CREATE NEW ONES

Contents

Who is talking about it?

- Working on the Office for 15 Years
- Experience in Graphic Processing
- Always interested in graphic aspects
- Involved in Draw/Impress/DrawingLayer
- Involved in DrawingLayer usages (all other Apps)
- Often got Involved when graphics 'did not work'
- Started to change how the Office is processing Graphis because it needs to be done

Why are DrawingLayer Primitives needed?

Past:

> Procedural Paint model (no 'real' MVC)

> No separation of repaint/direct paint (invalidate)

> Vcl and OutputDevice for everything – pros and cons

> Paint, Print, Import/Export

> Metafile (Contained 'Workarounds', unflexible)

> Flickering when scrolling, Interactions

> Missing AnitiAliasing, 3D

> Missing quality (Integer coordinates, what...?)

> No Reusability (Tooling, Processing, Im/Export)

Why are DrawingLayer Primitives needed?

Future (Ideal State):

> Generic, Minimal, Reusability, Object-Oriented, Modular, Self-Contained (Read-Only)

> On-Demand create/destroy/buffer (object lifetime)

> AntiAliasing, Buffering

> Not only 'Paint', but Graphic Processing

> Easily adaptable to new Graphic Sub-Systems

> Easy to understand/flat learning curve

> Based on UNO API

> Render all EditViews in the future using Primitives

Why are DrawingLayer Primitives needed?

Present:

- Good working Implementation (prove by doing)
- Draw/Impress completely using Primitives (including UI)
- Writer/Calc partially, coexistence of old/new required
- Renderers used for Print/Paint/HitTest/Conversions
- Conversion from/to Metafile (with all hacks...)
- Exporters/Importers (SVG)
- NO Load/Save, more to be seen as a runtime Tooling
- Limited UNO API
- Using basegfx graphic tooling (quality/precision)

DrawingLayer Primitive

Definition:

- Based on UNO API (XPrimitive2D)
  - Two methods
    - getDecomposition(ViewPrarameters)
    - getRange(ViewPrarameters)
  - Ref-Counted, Read-Only, sequence< XPrimitive2D >
- getDecomposition adds the needed flexibility
  - This makes the difference compared to metafile
- Two basic types
  - Generic Primitives: can not be decomposed further
  - Group Primitives: allow structuring and embedding

DrawingLayer Primitive

Definition:

- The four Generic Primitives are:
  - BitmapPrimitive2D
  - PointArrayPrimitive2D
  - PolygonHairlinePrimitive2D
  - PolyPolygonColorPrimitive2D
- A Primitive that can not be decomposed further is a Generic Primitive
- Everything else can be decomposed to these
- Being extreme, this could even be reduced (all Bitmaps…?)

DrawingLayer Primitive

Definition:

- GroupPrimitive2D for structuring (returns Childs)
- The four basic Group Primitives are:
  - TransformPrimitive2D
  - TransparencePrimitive2D
  - MaskPrimitive2D
  - ModifiedColorPrimitive2D
- For processing, this is the set to be supported
  - Some processors even use less (geometry extractors)
  - For better performance, support more (e.g. fat line, simple transparence)

DrawingLayer Primitive

Requirements:

- Self-contained (Data, References)
- Read-Only (just get...() methods)
- Separate Graphics into Definition and Processing
- getDecomposition for non-Generic Primitives required
- Support for buffering decompositions
- Support for View-Dependent decompositions
- Support for getRange uses decomposition
- GroupPrimitive2D derivates can anytime be used to encapsulate specific information, does no harm

DrawingLayer Primitive

Implementation:

- Basic Primitives are in drawinglayer project
- More are implemented wherever needed (60-80?)
- As long as decomposition creation is supported, the Primitive will be processed/rendered
- GetRange() may even be moved to an own Processor
- Operator== in implementation may be removed (aw080)
- Primitives need to be self-contained
- Primitives may be very 'complex' (whole SdrObject...?)
- Implementation supports unique ID for switch Statements

DrawingLayer Primitive

3D:

- In parallell, primitive definitions exist for 3D
- XPrimitive3D UNO API and implementation
  - Generic 3D Primitives are:
    - PolygonHairlinePrimitive3D
    - PolyPolygonMaterialPrimitive3D
  - Group 3D Primitives are:
    - TransformPrimitive3D
    - TexturePrimitive3D (Gradient, Transparence, Bitmap, Hatch)
- Implementation of 3D Scene is itself a 2D Primitive, decomposing to a Bitmap (except shadow, uses a processor)

DrawingLayer Primitive

Examples:

- Generic Primitives:
  - AnimatedPrimitive
  - CropPrimitive
  - GridPrimitive
  - UnifiedTransparencePrimitive
  - TextDecoratedPrimitive
  - Metafileprimitive
- Group Primitives:
  - HiddenGeometryPrimitive
  - InvertPrimitive
  - ShadowPrimitive

DrawingLayer Processor

Definition:

- Not (yet) based on UNO API, BaseProcessor2D/3D
  - Unified processing of a sequence of Primitives
    - Contains needed ViewInformation
    - Single call 'process(Primitives)'
  - Detects if Primitive instances are own implementation. If not, get decomposition using UNO API and call recursively
  - Implementations fetch unique ID from Prinitive and use one switch..case Block
  - Generic Primitives need to be implemented and rendered
  - Group primitives are partially generic supported
    - TransformPrimitive2D creates updated ViewTransformation, calls recursively with children

DrawingLayer Processor

Implementation:

- Basic Processors for 2D
  - VclProcessor for Pixel-Target (VCL OutputDevice)
  - MetafileProcessor (VCL Metafile, Print, PDF export, ...)
- Basic Processor for 3D (soft-renderer, AAed, ...)
- Todo: Generic, system-specific Renderers
  - For 2D Pixel-Target, could greatly increase Speed
  - For 3D Target
  - For Exports: PDF, SVG, ...

DrawingLayer Processor

Examples:

- Not only rendering – lot of other processing:
  - ContourExtractor
  - HitTestProcessor
  - LineGeometryExtractor
  - TextAsPolygonExtractor
- UNO API:
  - The decomposition implementation of UNO API incarnations of Primitives is used
  - There is the interface XPrimitive2DRenderer to convert any sequence of Primitives to Bitmap format

How to create a new Primitive

From Generic Primitive:

- Minimal Steps:
  - Derive from
    - BasePrimitive2D
    - Implement decomposition
  - Use it (Incarnate, add to sequence, …)
- From that moment on, Your Geometry will be handled correctly throughout the Office. Screen visualization, Print, PDF Export, SVG export, …

How to create a new Primitive

From Generic Primitive:

- Optional:
  - Derive from BufferedDecompositionPrimitive2D (optional use DiscreteMetricDependentPrimitive2D ViewportDependentPrimitive2D ViewTransformationDependentPrimitive2D)
  - React eventually View-Dependent inside decomposition
  - Implement operator==
  - Implement getB2DRange
  - Add to Processors of your choice when you want/need special handling in that Rendererer

How to create a new Primitive

From Group Primitive:

- Data embedding:
  - Derive from Group Primitive, add your Data, embed all other Primitives as children
  - Add to the Processor where you need (and know) it, use it there
  - Every other Processor will ignore it, using children as decomposition
- Using existing sequence of Primitives:
  - Get it's current Range, create needed Transformation, embed to new TransformPrimitive2D
  - Embed to MaskPrimitive2D with new clipping PolyPolygon
  - Embed to ModifiedColorPrimitive2D to force e.g. to all-Black

Future

- Make even more Generic:
  - Aw080: operator== removed
  - UNO API: provide/implement for the basic Primitives to allow using them from other languages
  - Reduce number of basic Primitives further (All Bitmaps…?)
  - Remove getRange(), replace with dedicated Processor
- Use more:
  - Migrate more parts of the Office to use Primitives
  - EditViews, EditEngine, …
- Get faster:
  - Create system-specific Renderers for 2D/3D/all Systems

Future

- Add Support for nicer Gradients
  - What about a SVGGradientPrimitive...?
- Add Support for Graphic Im/Exporters
  - SVG import already uses it
  - SVG export, PDF export could profit (quality, precision)
  - Most known Metafiles, would increase quality and Clipboard
- No-Go's:
  - Do not add a GraphicFormat to save/load Primitives, this would freeze current definitions. That was the beginning of all Problems with Metafiles...

Examples

- Enough Text, Let the Office talk…

Thank You for watching!

> To not Forget:

Your Help is Needed to drive this forward!