Pharo Graphics

The Pillar team

January 6, 2023

Layout and typography based on the `sbabook` LaTeX class by Damien Pollet.

Contents

Illustrations

CHAPTER 1

Vector graphics in Athens

There are two different computer graphics: vector and raster graphics. Raster graphics represents images as a collection of pixels. Vector graphics is the use of geometric primitives such as points, lines, curves, or polygons to represent images. These primitives are created using mathematical equations.

Both types of computer graphics have advantages and disadvantages. The advantages of vector graphics over raster are:

- smaller size,
- ability to zoom indefinitely,
- moving, scaling, filling, and rotating do not degrade the quality of an image.

Ultimately, picture on a computer are displayed on a screen with a specific display dimension. However, while raster graphic doesn't scale very well when the resolution differ too much from the picture resolution, vector graphics are rasterized to fit the display they will appear on. Rasterization is the technique of taking an image described in a vector graphics format and transform it into a set of pixels for output on a screen.

Morphic is the way to do graphics with Pharo. However, most existing canvas are pixel based, and not vector based. This can be an issue with current IT ecosystems, where the resolution can differ from machine to machine (desktop, tablet, phones, etc...)

Enter Athens, a vector based graphic API. Under the scene, it can either use balloon Canvas, or the cairo graphic library for the rasterization phase.

1.1 Example

Here it would be nice to have an example

1.2 Athens details

`AthensSurface` and its subclass `AthensCairoSurface` instances represent a surface. A surface is the area in pixel where your drawing will be rendered. You never draw directly on the surface. Instead, you specify what you want to display on the canvas, and Athens will render it on the area specified by the surface.

The class `AthensCanvas` is the central object used to perform drawing on an `AthensSurface` A canvas is not directly instanciated but used through a message such as:`surface drawDuring: [:canvas | .... ]`

The Athens drawing model relies on a three layer model. Any drawing process takes place in three steps:

- First a `path` is created, which includes one or more vector primitives , i.e., circles, lines, TrueType fonts, Bézier curves, etc...
- Then painting must be defined, which may be a color, a color gradient, a bitmap or some vector graphics
- Finally the result is drawn to the Athens surface, which is provided by the back-end for the output.

1.3 Path

Athens has always an active path. Use `AthensPathBuilder` or `AthensSimplePathBuilder` to build a path. They will assemble path segments for you.

The method `createPath:` exists in all important Athens class: `AthensCanvas`,`AthensSurface`, and `AthensPathBuilder`. The message `createPath: aPath`

Using it returns a new path:

```
surface createPath: [ :builder |
 builder
  absolute;
  moveTo: 100@100;
  lineTo: 100@300;
  lineTo: 300@300;
  lineTo: 300@100;
  close ].
```

Here are some helper messages in `AthensSimplePathBuilder`:

- `pathStart`
- `pathBounds` gives the limit of the bounds associated to the path

If you want to build path using only straight line, you can use the class `AthensPolygon`.

path builder Messages	Object Segment	comment
-		
ccwArcTo: angle:	AthensCCWArcSegment	counter clock wise segment
cwArcTo:angle:	AthensCWArcSegment	clock wise segment
lineTo:	AthensLineSegment	straight line
moveTo:	AthensMoveSegment	start a new contour
curveVia: to:	AthensQuadSegment	quadric bezier curve
curveVia: and: to:	AthensCubicSegment	Cubic bezier curve
reflectedCurveVia: to:	AthensCubicSegment	Reflected cubic bezier curve
string: font:		specific to cairo
close	AthensCloseSegment	close the current contour

1.4 Coordinate class: Absolute or Relative

absolute: absolute coordinate from surface coordinate. This will draw a square in a surface which extent is 400@400 using absolute move.

```
builder absolute;
 moveTo: 100@100;
 lineTo: 100@300;
 lineTo: 300@300;
 lineTo: 300@100;
 close
```

relative: each new move is relative to the previous one. This will draw a square in a surface which extent is 400@400 using relative move.

```
 builder relative ;
  moveTo: 100@100;
  lineTo: 200@0;
  lineTo: 0@200;
  lineTo: -200@0;
  close
```

cwArcTo:angle: and ccwArcTo: angle: will draw circular arc, connecting previous segment endpoint and current endpoint of given angle, passing in clockwise or counter clockwise direction. The angle must be specified in Radian.

curveVia: to: and |curveVia: and: to:

This call is related to bezier curve. A Bézier curve consists of two or more control points, which define the size and shape of the line. The first and last points mark the beginning and end of the path, while the intermediate points define the path's curvature.

More detail on Bezier curve on available at: https://pomax.github.io/bezier-info/

path transformation.

A path can be rotated, translated and scaled so you can adapt it to your need. For example, you can define a path in your own coordinate system, and then scale it to match the size of your surface extent.

1.5 The different type of painting.

Paints can be created either from the surface or directly from a class that will do the call to the surface for you.

any object can be treated as paint: - `athensFillPath: aPath on: aCanvas` - `athensFillRectangle: aRectangle on: aCanvas` - `asStrokePaint`

---

surface message

createFormPaint:
createLinearGradient: start: stop:
createRadialGradient: center: radius:
createRadialGradient: center: radius: focalPoint:
createShadowPaint:
createSolidColorPaint:
createStrokePaintFor:

---

a Canvas define its paint method we will see in detail below:

- setPaint:
- setStrokePaint:

1.6 Stroke paint (a pen that goes around the path)

The `createStrokePaintFor:` operation takes a virtual pen along the path. It allows the source to transfer through the mask in a thin (or thick) line around the path

`AthensStrokePaint` represents a stroke paint.

1.7 Solid paint (a pen that fill the path)

The `createSolidColorPaint` operation instead uses the path like the lines of a coloring book, and allows the source through the mask within the hole whose boundaries are the path. For complex paths (paths with multiple closed sub-paths—like a donut—or paths that self-intersect) this is influenced by the fill rule

SD: We need examples

1.8 Gradient

Gradient will let you create gradient of color, either linear, or radial.

The color ramp is a collection of associations with keys - floating point values between 0 and 1 and values with Colors, for example:

Bibliography