Stylo: the diagram toolkit for SATySFi

Yu Shimura

Stylo is a satysfactory graphics library which focuses on drawing geometrical diagrams in its entirety. Rendered diagrams are meant to be comparable to Xy-pic package of LaTeX, but the code looks readable and descriptive.

1. Concepts and datatypes

TBW (some amazing art)

TBW

1.1. Geometricals

Concepts described under this section are purely geometrical and are essentially not tied to actual style when rendered.

1.1.1. Points

TBW (illustration of the Big Dipper?)

A point is one of the most primitive object in the world of graphics.

1.1.2. Lines

TBW

A line consists of two points.

1.1.3. Paths

TBW

A path—that's all about diagrams. Stylo uses its own implementation of path instead of SATySFi's built-in one, in order to achieve a series of manipulation around paths in depth. Constructed paths are translated into built-in representations at the time of rendering.

Unlike SATySFi's path implementation, Stylo has no concept of "pre-path". Each construct always returns a complete path, closed or not. Basically returned paths are open, so if you want to close them, connect the starting point and the ending point and just pass them into `finish`.

A path consists of two or more path segments. Each segment has just one endpoint.

Path ending can be closed or not. The latter case is called "open" or "terminated".

1.1.4. Positions

TBW

A position is a waypoint on a path. Some operations require some positions on a path to be specified by scalar values. Usually, in the term of specifying these positions, paths are always treated as if the starting point and ending point are connected to each other, regardless of being closed or not. That is:

- When specified values are negative, interpreted backwardedly from the ending point of the path.
- When the absolutes of specified values exceed the total length of the path, interpreted cyclically through the path.

Positions can be specified in scale or length. That is:

- When in scale, interpreted assuming `0.0` and `1.0` are the starting point and the ending point of the path respectively.
- When in length, interpreted assuming `0cm` is the starting point of the path.

1.1.5. Angles

TBW (some angels?)

Angles can be specified in radians or degrees.

1.2. Decoratives

In this section, style-related concepts and predefined decoratives are shown in series.

- TBW

1.2.1. Pins

TBW

Pins are used to indicate specific points in the diagram.

A pin is actually a small footage with its origin point placed at a given point. Thereby any types of complex paths can be used as pins, including labels.

Pins are oftenly seen to be arrowheads but also can be used as a pseudo-arrowshaft by splitting a path into multiple subpaths.

1.2.2. Wires

TBW

Wires indicate lines and curves in the diagram. Sometimes wires are used as arrowshafts.

1.2.3. Labels

TBW

TBW

1.3. Footages

A footage is a ready-to-render object containing some decorated paths and/or labels, but still accepts some manipulations such as affine transformations and configurations of contextural options.

1.3.1. Contexts

2. Constructs and functions

In the world of Stylo, the noun “construct” means “instruction”, “command” or “operation”. There's a set of constructs for drawing, splicing, measuring and decorating paths.

2.1. Naming convention

Constructs are named so that what each parameter means can be guessed, in the form of `<x>-<t0>-<t1>-...-<tn>`.

- `<x>` means what is drawn or performed by the construct, such as `start`, `arc` and `split`.
- `<t0>` means what type the paenultima parameter takes.
- `<t1>` means what type the antepaenultima parameter takes.
- And so on. The ultima parameter is always a path, so it's not appeared in the name.

Note that these specifiers are appeared in reverse order. The full list of type specifiers is below.

- `to`: A point, to which a translation is performed or a path is drawn.
- `upto`: A length, upto which the path is drawn.
- `by`: An angle, by which a rotation is performed or an arc is drawn.
- `around`: A point, around which a linear transformation is performed or an arc is drawn.
- `at`: A position on the path, at which a cut or join is performed.

For example, `arc-by-around` firstly takes a centerpoint of the arc, secondly takes an angle to draw, and lastly takes a path.

2.2. Constructing paths

Paths can be constructed from scratch using these constructs.

2.2.1. Basic constructs

These constructs are just straightforward to SATySFi's built-in functions with a few exceptions.

`start` : point → pre-path
: Starts a path from a given point.

`start-with-tangent` : point → point → pre-path
: Starts a path from a second given point, with the pre-start tangent from a first given point. The tangent may be referred by the subsequent construct such as `arc-to-with-tangents`.

`line-to` : pre-path → point → path
: Draws a line to a given point.

`bezier-to` : pre-path → (point × point × point) → path
: Draws a Bézier curve to the last given point with the first two given control points.

`finish` : pre-path → path
: Close a path if the current point is meeting to the starting point, otherwise leave it open.

`close-with-line` : pre-path → path
: Closes a path with a line.

`close-with-bezier` : (point × point) → pre-path → path
: Closes a path with a Bézier curve. Two points stand for control points.

`restart` : path → pre-path
: Restarts a path, opening if closed. Yields the same result as `restart-at (Scale 1.0)`.

`restart-at` : position → path → pre-path
: Restarts a path from a given position. The path portion after the position is discarded. Passing a zero returns the same result as `start (first-point-of path)`.

2.2.2. Drawing arcs

Stylo has a powerful set of constructs for drawing arcs in different ways.

`arc-by-around` : point → angle → pre-path → pre-path

Draws an arc around a given point, with a given angle.

`arc-by-aside` : length → angle → pre-path → pre-path

Draws an arc around a given point which is a given length away from the current point orthogonally to the current tangent, with a given angle. Positive length means the left direction.

`arc-to-with-tangents` : point → point → pre-path → pre-path

Draws an arc, followed by a line if needed, to the first given point, referring the current tangent and the second given point as a forwarding tangent. As a result, the arc drawn is smoothly tangented to the current path and also to the subsequent construct as long as it uses the identical point to the forwarding tangent as its starting tangent. See the illustration below:

`arc-to-with-tangents-counter` : point → point → pre-path → pre-path

Same as `arc-to-with-tangents`, except that it draws the counter arc against what `arc-to-for` draws thereby tangented like a pair of needles. Not implemented yet.

2.2.3. Tracing another path

Not implemented yet.

2.3. Predefined shapes

Not implemented yet.

2.3.1. Arrows

TBW

An arrow is constructed from a wire and one or two pin(s). TBW

2.4. Splicing paths

Splicing constructed paths is one of the main ways to produce complex paths.

2.4.1. Splitting

Paths can be splitted into subpaths at an arbitrary position.

`split-at` : position → path → (path × path)
: Splits a path in two at a given position.

`split-into-at` : position → float → path → path list
: Splits a path into a given number of subpaths, starting at a given position. The division number can be fractional and/or negative. When positive, the fractioned subpath comes in last. When negative, comes in first. Passing a zero returns an empty list.

`split-incr-at` : position → position → path → path list
: Splits a path in increments of a second given position, starting at a first given position. The incremental position can be negative. When positive, the fractioned subpath comes in last. When negative, comes in first. Passing a zero always results in an infinite loop. Unusually, when the incremental position longer than the length of the path, no split is performed and it returns just the original path wrapped in a list.

2.4.2. Trimming

Paths can be trimmed at an arbitrary endpositions. 切り落としの始端位置から終端位置への向きがパスを逆行する場合、切り出されるパスの向きもそれに従う。Not implemented yet.

`trim-within` : (position × position) → path → path
: Trims a path at given endpositions.

2.5. Measuring

Stylo provides a set of features that supports measuring distances, lengths, areas and angles amongst geometrical objects.

2.5.1. Distances

TBW

2.5.2. Lengths

Stylo's internal representations of lengths are all in centimeters, thereby returned lengths

of these functions are all in centimeters.

2.5.3. Areas

TBW

2.5.4. Angles

Stylo's internal representations of angles are all in radians, thereby returned angles of these functions are all in radians.

2.6. Compositing paths

Not implemented yet.

2.6.1. Boolean operations

Not implemented yet.

2.7. Transforming paths

Stylo supports affine transformations of points, paths and graphics. Not implemented yet.

3. Syntax sugars

Stylo provides some syntax sugars for specific use cases.

3.1. Grid layout

An equivalent feature to XY-matrices of XY-pic. Not implemented yet.