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.

closed: true

1. Concepts

TBW (some amazing art)

TBW

1.1. Geometricals

Concepts described under this section are purely geometrical and are essentially not tied to actual appearance 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. 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 transformed into built-in representations at the time of conversion into graphics.

A path consists of two or more path segments.

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

1.1.3. 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.

1.1.4. Angles

TBW (some angels?)

Angles can be specified in radians or degrees.

1.2. Decoratives

In this section, style-related datatypes are shown in series.

- TBW

1.2.1. Pins

TBW

TBW

1.2.2. Wires

TBW

TBW

1.2.3. Arrows

TBW

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

1.3. Textuals

1.3.1. Labels

TBW

TBW

2. Constructs

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.

TBW

2.1. Constructing paths

Paths can be constructed from scratch using these constructs.

2.1.1. Basic constructs

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

`start` : point → pre-path

Starts a pre-path from a given point.

`start-with-tangent` : point → point → pre-path

Starts a pre-path from a given point, with a phantom tangent which may be referred by the subsequent construct such as `arc-to-for`. The first argument is a point representing the tangent from the second argument which is the starting point.

`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.

`terminate` : pre-path → path

Terminates a pre-path leaving it open.

`close` : pre-path → path

Close a pre-path if the current point is meeting to the starting point, otherwise terminates leaving it open. Normally you want to use this construct instead of `terminate`.

`close-with-line` : pre-path → path

Closes a pre-path with a line.

`close-with-bezier` : (point × point) → pre-path → path

Closes a pre-path with a Bézier curve. Two points stand for control points.

`restart` : path → pre-path

Restarts a path. Yields the same result as `restart-at (Scale 1.0)`.

`restart-at` : position → path → pre-path

Restarts a path at a given position. Passing a zero position returns the same result as `start`.

2.1.2. Arcs

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

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

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

`arc-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-for` : point → point → pre-path → pre-path

Draws an arc 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 pre-path and also to the subsequent construct as long as it uses the identical point to the forwarding tangent as its starting tangent.

`arc-to-for-opposite` : point → point → pre-path → pre-path

Same as `arc-to-for` with the exception of its opposed arc so tangented like a pair of needles. Not implemented yet.

2.2. Predefined shapes

Not implemented yet.

2.3. Splicing paths

Manipulating constructed paths is the main way to produce complex paths.

2.3.1. 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.

2.3.2. 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-at` : $(\text{position} \times \text{position}) \to \text{path} \to \text{path}$

Trims a path at given endpositions.

2.3.3. Tracing

Not implemented yet.

2.4. Measuring

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

2.4.1. Distances

TBW

2.4.2. Lengths

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

2.4.3. Areas

TBW

2.4.4. Angles

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

2.5. Compositing paths

Not implemented yet.

2.5.1. Boolean operations

Not implemented yet.

2.6. Transforming paths

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

3. Special syntaxes

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.