Syntax of the intermediate representation

Origram

October 29, 2013

Abstract

Here is the description of the chosen syntax of the intermediate representation. First we give the data structures needed to represent an origami. Then we precise the syntax of the different functions one may call while building an origami.

1 Data structures

The two main structures that are needed are the points and the lines. A point is given by its name, and a line by two points which are its extremities.

At the beginning, we work on a regular n-gon. The points are called $V_1 ... V_n$. The lines are $V_1V_2 ... V_{n-1}V_n$.

For more informations about the internal definition and manipulation of these structures, and of other important structures, see the document *Data structures of the intermediate representation*.

2 Folding functions

There are seven basic folding functions, corresponding to the axioms of the origami. Each fold has a parity argument, which is "mountain" or "valley", and is realized w.r.t. this parity.

We write $fold\_name\,(args) \to output$ to define a function. Then we describe the behaviour of this function.

- $fold\_along\,(parity, line) \to unit$: fold along $line$, return nothing
- $fold\_VtoV\,(parity, point_1, point_2) \to line$: fold $point_1$ over $point_2$, return the line defined by the fold (the crease)
- $fold\_EtoE\,(parity, line_1, line_2) \to line$: fold $line_1$ over $line_2$, return the crease.
- $fold\_ortho\,(parity, line, point) \to line'$: fold along the line which is orthogonal to $line$ and to which $point$ belongs, return the crease
- $fold\_VtoE\,(parity, point_1, line, point_2) \to line'$: fold $point_1$ over $line$, along $line'$ to which $point_2$ belongs, return the crease
- $fold\_VVtoEE\,(parity, point_1, line_1, point_2, line_2) \to line$: fold $point_i$ over $line_i$ for $i \in \{1, 2\}$, return the crease
- $fold\_VtoEortho\,(parity, point, line_1, line_2) \to line$: fold $point$ over $line_1$, along $line$ which is orthogonal to $line_2$, return the crease

Moreover, every folding function has an optional boolean argument, which says whether or not the corresponding folding arrows must be generated automatically. By default, it is set to `true`. It will be set to `false` when we want to define macros for more advanced folds which require a different kind of arrow.

3 Fold updates

One may want to change some caracteristics of a fold. Thus, we need some "update" functions to perform these modifications.

- $reverse\,(line) \rightarrow unit$: reverse the fold defined by *line*, return nothing
- $unfold\,(line) \rightarrow unit$: unfold along *line*, return nothing

4 Diagramming functions

The intermediate representation is used both to generate the 3D model of the origami, and the final diagram. Thus, we need to add some diagram-specific functions.

- $step(annotation) \rightarrow unit$: end of a diagram step. It genetates a new step on the diagram, with an annotation. This corresponds to the "snapshot" button in the user interface.
- $show() \rightarrow unit$: display the current state of the 3D model in the GUI. This can be done several times within one step. This function is used to specify that several folds must be done simultaneously : some macros (e.g., *squash fold*) are composed of several basic folds which cannot be done in a sequential manner without intersert-ing surfaces. We want to avoid displaying such steps. All folding functions called between two consecutive calls to $show()$ are applied simultaneously.
- $move\_cam(\theta, \phi, \psi) \rightarrow unit$: move the camera relatively to the current position. The arguments are Euler's angles.
- $reset\_view() \rightarrow unit$: reset the camera to its initial position.
- $arrow\_fold(point_1, point_2, type) \rightarrow unit$: draw an arrow from $point_1$ to $point_2$. The *type* argument specifies the kind of arrow to be drawn.
- $arrow\_push(line) \rightarrow unit$: Draw the arrow which corresponds to pushing the *line*.
- $arrow\_pull(line) \rightarrow unit$: Draw the arrow which corresponds to pulling the *line*.

5 Defining new objects

During the creation of an origami, several new lines and points are created. In order to be able to use them, we need some definition functions.

- $assert\_point(line, coef_1, coef_2) \rightarrow point$: Define a new point given by its arbitrary position on an existing line. This position is given as the barycenter of the extremities of the line, with coefficients $coef_1$ and $coef_2$.
- $intersect(line_1, line_2) \rightarrow point$: Defines a new point as the intersection of two existing lines.
- $def\_line(point_1, point_2) \rightarrow line$: Defines a new line which has for extremities $point_1$ and $point_2$.