Documentation CGAShape2D

Overview

The CGAShape grammar consists of a set of Rules. Each Rule has the following format:

```
<Symbol-Id> --> <Operation>+
```

In reach rule, at least one operation is required. If multiple operations are used in a rule, the operations are subsequently applied to the same symbol. Multiple Operations are separated by an empty space.

Example of a rule with more than one operations:

```
A --> component_split(){B,C,D,E} identity(){F}
```

Each Operation has the form

```
<Name>(Parameter*){Successor-ID+}
```

An Operation can have an arbitrary number of parameters. Parameters are separated by commas (","). The number of successors-ids depends on the operation. Multiple successor-ids are separated by commas.

Remark: multiple rules for the same symbol are currently ignored because probabilities for rules are currently not implemented yet.

Shapes and Shape-Tree

The basic building block in this grammar is a Shape2D. Each shape consists of a symbol (id, String) and a homogenious transformation (scope) which is represented as a homogenious 3x3 matrix. A shape can have an arbitrary number of child shapes.

Axiom: The axiom shape is a Point2D with id *Origin*

The following shapes are available:

- Point2D: A point of dimension 0. Its coordinates are (0,0) in its local coordinate system.
- Line2D: A line of dimension 1. Specified by its length. Its end points in its local coordinate system are (0,0) and (length,0).
- Polygon2D: A polygon of dimension 2 with at least three points. The points are organized in such a way that they all have positive coordinates in its local coordinate system. At least one point lies on the x-axis and and at least one point lies on the y-axis.
- Special: A special shape of dimension 2 to represent details in a generated complex shape. Currently supported are

- Roof: Represents a roof, replaces a Line2D
- Window: Represents a window, replaces a Polygon2D (should be rectangular)
- Door: Represents a door, replaces a Polygon2D (should be rectangular)

When applying a rule to a shape, the id of the shape must be on the left side of the rule. The operation then creates new shapes with the given successor symbol-ids. These new shapes are set as the children of the original shape. Afterwards other rules can be applied to the child shapes. No more rules can be applied to the original shape after a rule was applied to it. A shape tree is generated subsequently. Its leaf node shapes are the ones that make up the final result.

Operations

Identity (*identity*)

The Identity-Operation just copies the shape as a child of the input shape. This Operation can be used, if an operation should be applied to a shape and the same shape (with a different symbol) shall be kept to be used with later rules. The Operation has no parameters.

Example:

```
A --> identity(){B}
```

Extrude (*extrude*)

The Extrude operation generates a new shape from the input shape with an increased dimension. It has one parameter: the length of the extrusion.

Can be applied to a

- Point2D: Results in a Line2D, the extrusion length is used as line length.
- Line2D: Results in a Polygon2D, the polygon has four points and is rectangular with width = line-width and height extrusion-length.

The operation must have one successor-id.

Example:

```
A --> extrude(0.7){B}
```

Split (*split*)

The Split operation partitions the shape into subshapes of the same type. It has one parameter for the splitting direction and additionally an arbitrary number of split-length-parameters (at least two). Each split-length-parameter can either be an absolute floating point value or a relative value. Relative values are marked with an r.

Can be applied to

- Line2D: generates new lines
- Polygon2D: generates new polygons (**Attention:** The current implementation only works correct for rectangular polygons.)

The number of successor-ids must be the same as the number of split-length-parameters.

Example:

```
A --> split(x,1r,0.5,2r){B,C,B}
```

This Extrude operation splits along the x-axis. The length of the shape A in this direction is subdivided into:

- 1/3*(length-0.5)
- 0.5
- 2/3*(length-0.5)

Therefore, the length of the input must be at least 0.5.

Component-Split (*component_split*)

The Component-Split operation generates shapes of dimension - 1.

Can be applied to

- Line2D: Generates two Point2D shapes. One for the start point (0,0) and one for the end points. Note, that the local coordinates of the end point shape are also (0,0), but is local scope is a Translation to (line-length,0).
- Polygon2D: Generates Line2D shape for each segment in the polygon. The first line is generated for the points pair (p0,p1). The x-direction of the new scope (local coordinate system) points along p1->p0. The y-axis is perpendicular and points outside. The origin of the local coordinate system is p1.

Example:

```
A --> component_split(){B,C,B,C}
```

This Component-Split should be applied to a Polygon2D with four points. Two of the resulting lines have the id B, the other two have the id C.

Special (*special*)

The Special operation allows to replace shapes by specialized details. Details are represented as SpecialShape2D shapes. These can usually not be derived further. The Operation has one parameter which must be one of:

- roof: can be applied to a Line2D, represents a roof
- window: can be applied to a (4-point, rectangular) Polygon2D, represents a window
- door: can be applied to a (4-point, rectangular) Polygon2D, represents a door

Example:

```
A --> special(roof){B}
```