PGA Shape Grammar

Shape Grammar

The shape grammar describes the production system to modify geometric shapes. A rule set is used to transform input shapes (called axioms) to the desired output. The process of computing the output is called rule derivation.

Shapes

For the moment, shapes can be boxes or quads in our shape grammar. The starting shape is called an axiom.

Rule Set

A rule set consists of an arbitrary number of rules that are needed to produce the desired output geometry.

Rules

Rules are used to describe how you want to modify the input shape. They consist of symbols and operators. Some operators in a rule can produce more output shapes than were originally put into the rule.

```
input-symbol --> operators(parameters) output-symbol;
```

Symbols

Practically, symbols are used to name rules. The processing starts with the execution of the rule that matches the special symbol called axiom. The axiom has a name and a type. For example:

axiom Box A;

If a rule exists in the rule set, which is named after the output-symbol, the output-symbol is called an intermediate-symbol (B). If no rule is found for the output-symbol, then it is called a terminal-symbol (C) and the processing of the shape is finished. The output-symbol is also called successor. Example (without operators):

```
A --> B;
B --> C;
```

Stochasticity

Rules can have output with a certain probability. For example, if a rule with three different outputs is desired, where the first output should be of 20% chance, the second of 30% and 50% for the rest, an example rule would look like this:

```
A --> 20%:   B
      30%:   C
      else:  D;
```

For this kind of construct there always has to be the "else:" in the last output which in this case will have a probability of 50% because the sum of all probabilities has to add up to 100%.

Notice the semicolon (;) at the end of the third line. A rule can be written over several lines and always has to end with a semicolon.

Operators

Translate

Syntax: `Translate ( x, y, z)`
Description: Moves a shape to coordinates described by x,y,z

Rotate

Syntax: `Rotate ( x, y, z)`
Description: Rotates a shape by specified degrees around the three axes.

Scale

Syntax: `Scale ( x, y, z)`
Description: Changes the size of a shape in x,y,z dimension

Repeat

Syntax: `Repeat ( <axis>, <extent> )`
Description: Produces as many output shapes (of the same type) that <extent> fits into the original shape on the specified axis. For example Repeat(x,2) applied on a box which is 10 wide on the x-extent, would produce 5 boxes which are 2 wide on the x-axis and have the same extents on the y and z axis. All output shapes have the same successor.

SubDiv

Syntax: `SubDiv ( <axis> ) { <extent> : successor | <extent> : successor | … }`
Description: The subdivide operator divides a shape along the specified axis according to the specified extents. The produced new shapes each may have their own successor. There can be an arbitrary number of extent/successor pairs specified, as long as the sum of extents fits into the extent of the original shape.

CompSplit

Syntax: `CompSplit () { Top | Bottom | Sides }`
Description: Splits a box into its six faces (quads). The output-symbols can be specified for the top, the bottom and the four sides. So all four sides of the box will be processed by the same successive rule.

Extrude

Syntax: `Extrude ( <extent> )`
Description: Generates a box with the height specified by <extent> out of a quad.

Discard

Syntax: `Discard ()`
Description: Stops processing and does not produce an output shape.

Parameters

A parameter can be:

A plain floating point number (i.e: 0, -1, 3.1415).
An arithmetic expression (+,-,*,/) (i.e.: Exp(0.75 * ShapeSize(X)).
An axis (i.e.: X|Y|Z).
A repeat mode (i.e.: ANCHOR_TO_START, ANCHOR_TO_END, ADJUST_TO_FILL).

A shape attribute (i.e.: ShapePosition, ShapeSize, ShapeRotation, ShapeNormal, ShapeSeed and ShapeCustomAttribute).