

Context Free in a Nutshell

Basic Directives

startshape
startshape core

Indicates which rule or path is used as the starting point for the generated image. If there are multiple startshape directives then the first one is used and the rest are ignored.

include
include alfreebet.cfg
include "lib/alien eyes.cfg"

Causes the contents of a specified file to be parsed after the current file is finished parsing. The file name can be in quotes or not. If the file name is in a different directory, contains white space, or does not end in '.cfg' then it must be enclosed in quotes.

background

background {a -1} // transparent

Causes the color of the background to be other than opaque white. If there is more than one background directive then the first one is used and the rest are ignored.

tile
tile {s 30}
tile {s 20 30}
tile {skew 15 15 r 15 s 3} # hex tile

Enables tiled rendering and sets the size of the tiling lattice. If there is more than one tile directive then the first one is used and the rest are ignored. The tiling lattice can be square, rectangular, skewed, or rotated as long as one lattice axis is either perfectly vertical or horizontal.

In the third example the **skew 15 15** creates the hexagonal lattice, but neither lattice axis is vertical or horizontal, so the **r 15** is required to make a legal hexagonal tiling.

tile {s 30 x 15 y 10}
x or y shifts move the content with respect to the tiling lattice. This allows the user to center part of the design in the tile.

Rules

```
rule BoxedCircle {  
    SQUARE {}  
    CIRCLE {b 1}  
}
```

A rule is a list of shapes that describes how a particular shape can be made from other shapes.

x num
y num
z num

move on the x, y, or z-axis by num
size scale
scale in x and y by the same amount

size xscaleyscale
scale in x and y independently
size xscaleyscale zscale
scale in x, y and z independently

rotate num
rotate num degrees
flip num
reflect across a line through the origin at num degrees

skew ynum xnum
shear ynum degrees from the y axis and xnum degrees from the x axis
Basic & Ordered Adjustments

```
// move (2,5), rotate 30°, scale (2,1)  
SQUARE {s 2 1 r 30 x 2 y 5}  
// move (0.5,0), scale 0.95, move (0.5,0)  
SQUARE [x 0.5 s 0.95 x 0.5]
```

If the geometric adjustments are in square brackets, [], then they are applied in order. If the adjustments are in curly brackets, {}, then they are re-ordered to x, y, rotate, size, skew, and flip. Duplicates are dropped.

Shape Adjustments

A shape passes on its state (geometry, color, and z position) to the shapes that replace it when a rule for the shape is executed. Each replacement can have shape adjustments that modify the state in the replacement shape:

x num
y num
z num

move on the x, y, or z-axis by num
size scale
scale in x and y by the same amount

size xscaleyscale
scale in x and y independently
size xscaleyscale zscale
scale in x, y and z independently

rotate num
rotate num degrees
flip num
reflect across a line through the origin at num degrees

skew ynum xnum
shear ynum degrees from the y axis and xnum degrees from the x axis
Basic & Ordered Adjustments

```
rule foo { SQUARE {} } # 1/11.01=9.1%  
rule foo 10 { CIRCLE {} } # 10/11.01=90.8%  
rule foo 0.01 {} # 0.01/11.01=0.09%
```

A rule can have a rule weight following the shape name that indicates the relative probability for that rule to draw the shape.

Loops

Simple form
integer * {adjustments} name{adjustments}

Where both *adjustments* can either be basic or ordered {} or [] and *name* is the name of a rule, path, or primitive shape.

Complex form

```
rule grid {  
10* {y 1} {  
10* {x 1} shape {b 0.5} # complex form  
10* {x 1} shape {b 0.5} # simple form  
}
```

Color Adjustments

Paths

Path operations

```
MOVETO {x xnum y ynum}  
LINETO {x xnum y ynum}  
ARCTO {x xnum y ynum rx xradius ry yradius r angle  
p params}  
ARCTO {x xnum y ynum r radius p params}  
CURVETO {x xnum y ynum x1 xcstr1 y1 ycstr1}  
CURVETO {x xnum y ynum x2 xcstr2 y2 ycstr2}  
CURVETO {x xnum y ynum x1 xcstr1 y1 ycstr1  
x2 xcstr2 y2 ycstr2}
```

Target Color

```
lhue num  
lsat num  
lb num  
lalpha num
```

These change the target color in the same manner that the first four change the drawing color.

hue num

Range [-1, 1]. If num < 0 then change the drawing hue num% toward the target hue moving clockwise around the color wheel. If num > 0 then change the drawing hue in the counterclockwise direction.

sat num

Range [-1, 1]. If num < 0 then change the drawing saturation, brightness, or alpha num% toward 0 or the target value, whichever is closer. If num > 0 then change it num% toward 1 or the target value, whichever is closer.

b num

Range [-1, 1]. If num < 0 then change the drawing saturation, brightness, or alpha num% toward 0 or the target value, whichever is closer. If num > 0 then change it num% toward 1 or the target value, whichever is closer.

a num

Range [-1, 1]. If num < 0 then change the drawing saturation, brightness, or alpha num% toward 0 or the target value, whichever is closer. If num > 0 then change it num% toward 1 or the target value, whichever is closer.

Expressions

Operators

()	parentheses
[^]		exponentiation
-		negation
*, /		multiplication and division
+, -		addition and subtraction

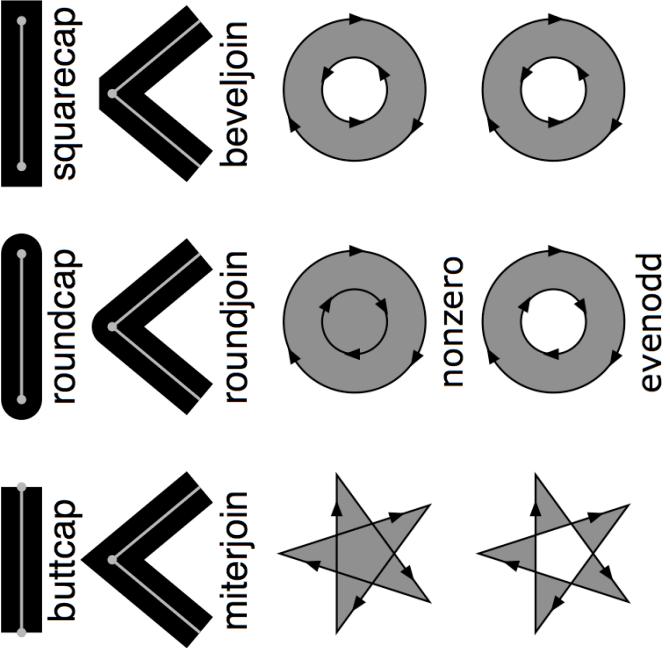
Functions

sin, cos, tan, cot,	all in degrees
asin, acos, atan,	acot, returning degrees
sinh, cosh, tanh,	asinh, acosh, atanh
log, log10, sqrt,	exp, abs
atan2(y, x),	mod(x, y)

An expression must be enclosed in parentheses unless it is a positive or negative constant or a simple function invocation.

Path commands

```
STROKE {adjustments width p params}  
FILL {adjustments p params}
```



```
path trill {  
    // no parentheses required  
    MOVETO {x cos(234) y sin(234)}  
    5 * {r -144} // ditto  
    // parentheses required because of  
    // addition  
    CURVETO {x 0 y 1  
             x1 (cos(234) + cos(324))  
             y1 (sin(234) + sin(324))  
             x2 1 y2 1}  
    CLOSEPOLY {p align}  
    FILL {p evenodd}  
    STROKE {p roundjoin a -0.5}  
}
```

