```
choose
	:
	(
	cfdg2
	|
	cfdg3
	)
	eof
	;

cfdg2
	:
	cfdg2 statement_v2
	|
	;

cfdg3
	:
	cfdg3 statement_v3
	|
	;

statement_v2
	:
	initialization_v2
	| directive_v2
	| inclusion_v2
	| rule_v2
	| path_v2
	;

statement_v3
	:
	initialization_v3
	| import_v3
	| rule_v3
	| path
	| shape
	| shape_singleton
	| global_definition
	| v2stuff .*?
	;

v3clues
	:
	USER_STRING BECOMES
	| modtype BECOMES
	| PARAM BECOMES
	| USER_STRING '('
	| USER_STRING USER_STRING '('
	| IMPORT
	| SHAPE
	| PATH USER_STRING '('
	| STARTSHAPE USER_STRING '('
	| STARTSHAPE USER_STRING '['
	| STARTSHAPE USER_ARRAYNAME '['
```

```
        ;

v2stuff
        :
        BACKGROUND modification_v2
        | TILE modification_v2
        | modtype modification_v2
        | INCLUDE fileString
        | rule_header_v2
        ;

inclusion_v2
        :
        INCLUDE USER_QSTRING
        |
        INCLUDE USER_FILENAME
        ;

import_v3
        :
        IMPORT fileNameSpace fileString
        ;

eof
        :
        EOF
        ;

fileString
        :
        USER_FILENAME
        |
        USER_QSTRING
        ;

fileNameSpace
        :
        '@' USER_STRING
        |
        ;

initialization_v3
        :
        STARTSHAPE USER_STRING parameter_spec modification
        |
        STARTSHAPE USER_ARRAYNAME modification
        |
        STARTSHAPE USER_STRING parameter_spec
        ;

initialization_v2
        :
        STARTSHAPE USER_STRING
        ;

directive_v2
```

```
        :
        directive_string modification_v2
        ;

directive_string
        :
        BACKGROUND
        |
        TILE
        |
        modtype
        ;

shape
        :
        SHAPE USER_STRING parameter_list
        ;

shape_singleton_header
        :
        shape '{'
        ;

shape_singleton
        :
        shape_singleton_header buncha_elements '}'
        ;

rule_header_v2
        :
        RULE USER_STRING
        |
        RULE USER_STRING user_rational
        ;

rule_v2
        :
        rule_header_v2 '{' buncha_replacements_v2 '}'
        ;

rule_header
        :
        RULE
        |
        RULE user_rational
        ;

path_header
        :
        PATH USER_STRING parameter_list
        ;

rule_v3
        :
        rule_header '{' buncha_elements '}'
        ;
```

```
path
        :
        path_header '{' buncha_elements '}'
        ;

path_header_v2
        :
        PATH USER_STRING
        ;

path_v2
        :
        path_header_v2 '{' buncha_pathOps_v2 '}'
        ;

parameter
       :
       USER_STRING USER_STRING
        |
        SHAPE USER_STRING
        |
        USER_STRING modtype
        |
        SHAPE modtype
        |
        USER_STRING
        |
        modtype
        ;

buncha_parameters
        :
        buncha_parameters ',' parameter
        |
        parameter
        ;

parameter_list
        :
        '(' buncha_parameters ')'
        |
        ;

function_parameter_list
        :
        '(' buncha_parameters ')'
        |
        '(' ')'
        ;

parameter_spec
        :
        '(' arglist ')'
        |
        '(' BECOMES ')'
```

```
        | '(' ')'
        |
        ;

buncha_elements
        :
        buncha_elements element
        |
        ;

buncha_pathOps_v2
        :
        buncha_pathOps_v2 pathOp_v2
        |
        ;

pathOp_simple_v2
        :
        USER_PATHOP '{' buncha_adjustments '}'
        |
        shapeName modification_v2
        ;

element_simple
        :
        USER_PATHOP '(' exp2 ')'
        |
        USER_PATHOP '(' ')'
        |
        shapeName parameter_spec modification
        |
        IF '(' exp2 ')' modification
        |
        letHeader letBody modification
        |
        PATH USER_STRING parameter_spec modification
        ;

one_or_more_elements
        :
        '{' buncha_elements '}'
        |
        element_simple
        ;

one_or_more_pathOp_v2
        :
        '{' buncha_pathOps_v2 '}'
        |
        pathOp_simple_v2
        ;

caseBody
        :
        caseBody_element caseBody
        |
```

```
        ;

caseBody_element
        :
        caseHeader one_or_more_elements
        ;

element
        :
        element_simple
        |
        definition
        |
        element_loop
        |
        element_loop FINALLY  one_or_more_elements
        |
        ifHeader one_or_more_elements
        |
        ifElseHeader one_or_more_elements
        |
        transHeader one_or_more_elements
        |
        switchHeader '{' caseBody '}'
        |
        element_v2clue .*?
        ;

element_v2clue
        :
        user_rational '*'
        | USER_STRING '{'
        | USER_PATHOP '{'
        ;

pathOp_v2
        :
        pathOp_simple_v2
        |
        loopHeader_v2 one_or_more_pathOp_v2
        ;

pathOp_v3clues
        :
        USER_PATHOP '('
        | USER_STRING '('
        | PATH
        | LOOP
        | USER_STRING BECOMES
        | modtype BECOMES
        | IF
        | modtype
        | SWITCH
        ;

element_loop
```

```
        :
        loopHeader modification one_or_more_elements
        ;

buncha_replacements_v2
        :
        replacement_v2 buncha_replacements_v2
        |
        ;

one_or_more_replacements_v2
        :
        '{' buncha_replacements_v2 '}'
        |
        replacement_simple_v2
        ;

replacement_simple_v2
        :
        shapeName modification_v2
        ;

replacement_v2
        :
        replacement_simple_v2
        |
        loopHeader_v2 one_or_more_replacements_v2
        ;

loopHeader_v2
        :
        user_rational '*'  modification_v2
        ;

loopHeader
        :
        LOOP USER_STRING BECOMES exp2
        |
        LOOP modtype BECOMES exp2
        |
        LOOP exp2
        ;

ifHeader
        :
        IF '(' exp2 ')'
        ;

ifElseHeader
        :
        ifHeader one_or_more_elements ELSE
        ;

transHeader
        :
        modtype exp2
```

```
        |
        clone exp2
        ;

switchHeader
        :
        SWITCH '(' exp2 ')'
        ;

caseHeader
        :
        CASE exp2 ':'
        |
        ELSE ':'
        ;

modification_v2
        :
        '{' buncha_adjustments '}'
        |
        '[' buncha_adjustments ']'
        ;

modification
        :
        '[' buncha_adjustments ']'
        |
        '[' '[' buncha_adjustments ']' ']'
        ;

buncha_adjustments
        :
        buncha_adjustments adjustment
        |
        ;

adjustment
        :
        modtype explist
        |
        modtype exp 'l'
        |
        PARAM USER_STRING
        |
        PARAM USER_QSTRING
        ;

letHeader
        :
        LET
        ;

letBody
        :
        '(' letVariables ';' exp2 ')'
        ;
```

```
letVariables
	:
	letVariables ';' letVariable
	|
	letVariable
	;

letVariable
	:
	definition
	;

explist
	:
	explist exp
	|
	exp
	;

arglist
	:
	arglist ',' exp3
	|
	exp3
	;

exp
	:
	(
	user_rational
	|
	CF_INFINITY
	|
	USER_STRING '(' arglist ')'
	|
	expfunc
	|
	'(' exp2 ')'
	|
	'-' exp
	|
	'+' exp
	)
	(
	RANGE exp
	|
	PLUSMINUS exp
	)?
	;

exp2
	:
	exp2 ',' exp3
	|
	exp3
```

```
        ;

exp3
        :
        (
        user_rational
        |
        CF_INFINITY
        |
        USER_STRING '(' arglist ')'
        |
        expfunc
        |
        '(' exp2 ')'
        |
        '-' exp3
        |
        '+' exp3
        |
        NOT exp3
        |
        modification
        )
        (
        '+' exp3
        |
        '-' exp3
        |
        '_' exp3
        |
        '*' exp3
        |
        '/' exp3
        |
        '^' exp3
        |
        LT exp3
        |
        GT exp3
        |
        LE exp3
        |
        GE exp3
        |
        EQ exp3
        |
        NEQ exp3
        |
        AND exp3
        |
        OR exp3
        |
        XOR exp3
        |
        RANGE exp3
        |
```

```
        PLUSMINUS exp3
        )?
        ;

expfunc
        :
        USER_STRING '(' ')'
        |
        USER_ARRAYNAME '(' exp2 ')'
        |
        IF '(' exp2 ')'
        |
        USER_STRING '(' BECOMES ')'
        |
        letHeader letBody
        |
        USER_STRING
        ;

shapeName
        :
        USER_STRING
        |
        USER_ARRAYNAME
        ;

global_definition
        :
        global_definition_header exp2
        ;

function_definition_header
        :
        SHAPE USER_STRING function_parameter_list BECOMES
        |
        USER_STRING function_parameter_list BECOMES
        |
        USER_STRING USER_STRING function_parameter_list BECOMES
        |
        SHAPE modtype function_parameter_list BECOMES
        |
        modtype function_parameter_list BECOMES
        |
        USER_STRING modtype function_parameter_list BECOMES
        ;

global_definition_header
        :
        function_definition_header
        |
        definition_header
        ;

definition_header
        :
        USER_STRING BECOMES
```

```
        | modtype BECOMES
        ;

definition
        :
        definition_header exp2
        ;

modtype
    :
    (TIME | TIMESCALE | X | Y | Z | ROTATE | SIZE | SKEW | FLIP | HUE | SATURATION |
BRIGHTNESS | ALPHA | TARGETHUE | TARGETSATURATION | TARGETBRIGHTNESS |
TARGETALPHA | X1 | X2 | Y1 | Y2 | RX | RY | WIDTH | TRANSFORM)
    ;

clone
    :
    CLONE
    ;

user_rational
    :
    (INTEGER | RATIONAL | FLOAT)
    ;

STARTSHAPE
    :
    'startshape'
    ;

BACKGROUND
    :
    'background'
    ;

INCLUDE
    :
    'include'
    ;

IMPORT
    :
    'import'
    ;

TILE
    :
    'tile'
    ;

RULE
    :
    'rule'
    ;

PATH
```

```
    :
    'path'
    ;

SHAPE
    :
    'shape'
    ;

LOOP
    :
    'loop'
    ;

FINALLY
    :
    'finally'
    ;

IF
    :
    'if'
    ;

ELSE
    :
    'else'
    ;

SWITCH
    :
    'switch'
    ;

CASE
    :
    'case'
    ;

RANGE
    :
    '..' | '\u2026'
    ;

PLUSMINUS
    :
    '+/-' | '\u00b1'
    ;

TIME
    :
    'time'
    ;

TIMESCALE
    :
```

```
    'timescale'
    ;

X
    :
    'x'
    ;

Y
    :
    'y'
    ;

Z
    :
    'z'
    ;

ROTATE
    :
    'rotate' | 'r'
    ;

SIZE
    :
    'size' | 's'
    ;

SKEW
    :
    'skew'
    ;

FLIP
    :
    'flip' | 'f'
    ;

HUE
    :
    'hue' | 'h'
    ;

SATURATION
    :
    'saturation' | 'sat'
    ;

BRIGHTNESS
    :
    'brightness' | 'b'
    ;

ALPHA
    :
    'alpha' | 'a'
```

```
    ;

TARGETHUE
    :
    'lhue' | 'lh'
    ;

TARGETSATURATION
    :
    'lsaturation' | 'lsat'
    ;

TARGETBRIGHTNESS
    :
    'lbrightness' | 'lb'
    ;

TARGETALPHA
    :
    'lalpha' | 'la'
    ;

X1
    :
    'x1'
    ;

X2
    :
    'x2'
    ;

Y1
    :
    'y1'
    ;

Y2
    :
    'y2'
    ;

RX
    :
    'rx'
    ;

RY
    :
    'ry'
    ;

WIDTH
    :
    'width'
    ;
```

```
TRANSFORM
    :
    'transform' | 'trans'
    ;

PARAM
    :
    'param' | 'p'
    ;

BECOMES
    :
    '='
    ;

LT
    :
    '<'
    ;

GT
    :
    '>'
    ;

LE
    :
    '<=' | '\u2264'
    ;

GE
    :
    '>=' | '\u2265'
    ;

EQ
    :
    '=='
    ;

NEQ
    :
    '<>' | '\u2276'
    ;

NOT
    :
    '!'
    ;

AND
    :
    '&&'
    ;
```

```
OR
    :
    'll'
    ;

XOR
    :
    '^^'
    ;

CF_INFINITY
    :
    'CF_INFINITY' | '\u221E'
    ;

USER_PATHOP
    :
    'MOVETO'
    |
    'LINETO'
    |
    'ARCTO'
    |
    'CURVETO'
    |
    'MOVEREL'
    |
    'LINEREL'
    |
    'ARCREL'
    |
    'CURVEREL'
    |
    'CLOSEPOLY'
    ;

CLONE
    :
    'clone'
    ;

LET
    :
    'LET'
    ;

INTEGER
    :
    ('0'..'9')+ '%'?
    ;

RATIONAL
    :
    ('0'..'9')+ '.' ('0'..'9')+ '%'? | '.' ('0'..'9')+ '%'?
    ;
```

```
FLOAT
    :
    ('0'..'9')+ '.' ('0'..'9')+ ('e'|'E') ('+|-')? ('0'..'9')+ '%'? | '.' ('0'..'9')+ ('e'|'E') ('+|-')? ('0'..'9')+ '%'? | ('0'..'9')
+ ('e'|'E') ('+|-')? ('0'..'9')+ '%'?
    ;

USER_STRING
    :
    ('a'..'z'|'A'..'Z'|'_'|'\u0200'..'\u0301'|'\u0303'..'\u0377')
(('a'..'z'|'A'..'Z'|'0'..'9'|'_'|'::'|'\u0200'..'\u0301'|'\u0303'..'\u0377') |
('\u0302'('\u0200'..'\u0260'|'\u0262'..'\u0377')))*
    ;

USER_QSTRING
    :
    '"' USER_STRING '"'
    ;

USER_FILENAME
    :
    ('a'..'z'|'A'..'Z'|'\u0200'..'\u0377') ('a'..'z'|'A'..'Z'|'0'..'9'|'_'|'-'|'\u0200'..'\u0377'|'.')* '.cfdg'
    ;

USER_ARRAYNAME
    :
    ('a'..'z'|'A'..'Z'|'_'|'\u0200'..'\u0301'|'\u0303'..'\u0377')
(('a'..'z'|'A'..'Z'|'0'..'9'|'_'|'\u0200'..'\u0301'|'\u0303'..'\u0377') |
('\u0302'('\u0200'..'\u0260'|'\u0262'..'\u0377')))*
    ;

COMMENT
    :
    ('//' ~('\n'|'\r')* '\r'? '\n' {} | '/*' (.)*? '*/' {}) -> skip
    ;

WHITESPACE
    :
    ( ' ' | '\t' | '\r' | '\n' ) -> skip
    ;
```