Doodle Language

Language reference Manual

COMS W4115: Programming Language and Translator
Professor Stephen A. Edward

Yusr Yamani

Language Manual

1. Syntax Notation

Regular expressions are used for the syntax notation in this manual. Nonterminals categories are indicated by *italic* style. Quoted or bold style symbols are all terminals. Alternative categories are separated by '|'. An optional category ends with '?' . 'a*' indicates that 'a' may occur zero or more times. 'a+' indicates that 'a' may occur one or more times. '(a|b)' denotes a choice between the categories 'a' and 'b'. Parentheses are used to group symbols with respect to '?', '*', and '+'.

2. Lexical Conversions

2.1 Comments

Comments enhance programs readability. In Doodle, comments begin with '<*' and ends with '*>'. Any statement in between those symbols is ignored by the compiler.

2.2 Whitespace

Whitespaces, including ASCII space, carriage return and horizontal tab separate tokens. Any sequence of whitespaces is ignored by the compiler.

2.3 Tokens

There are six classes of tokens: identifiers, keywords, integer constants, string literals, operators, and separators.

Tokens are case insensitive. Upper and lower case of a letter are equally treated by the compiler.

2.3.1 Identifiers

An identifier is any sequence of letters and digits that begins with a letter. An identifier is longer than 8 characters. Doodle is case sensitive; two identifiers are the same if they have the same Unicode character for every letter and digit.

$$Identifier \rightarrow Letter(Letter \mid Digit)*$$

$$Letter \rightarrow ['a'-'z' \quad 'A'-'Z']$$

$$Digit \rightarrow ['0'-'9']$$

2.3.2 Keywords

The following terms are keywords of the language that may not be used otherwise

Declare	**String**	**Rectangle**	**Red**	**Loop**	**If**
Window	**Int**	**Ellipse**	**Blue**	**Endloop**	**Else**
Object	**Lcolor**	**Line**	**White**	**Func**	**Endif**
	Bcolor	**Text**	**Black**	**Endfunc**	

2.3.3 Integer Constants

An Integer constant is a sequence of ASCII digits that represents a decimal number. Integers are positive.

$$IntegerConstant \rightarrow digit+$$

2.3.4 String Literal

A String literal is a sequence of one character or more enclosed in double quotes.

$$StringLiteral \rightarrow \ ' \ " \ ' \ Letter+ \ ' \ " \ '$$

2.3.5 Operators

Operators are: plus '+' minus '-' times '*', divide '/', Assignment '='

2.3.6 Separators

The following symbols are separators in Doodle:

[ ] ; ( )

3. The structure of Doodle:

A Doodle program consists of three main parts: Declaration, WindowSpecification, and ObjectSection

$$DoodleProgram \rightarrow Declaration?\ \ WindowSpecification\ \ ObjectSection$$

3.1 Declaration:

This section is optional. It contains identifiers and functions being declared. Variables are only declared in this part of the program, and should be initialized to a value when they are declared. A variable could be either an integer or a string.

$$Declaration \rightarrow \quad '['DeclSpecification + ']'$$

3.1.1 DeclSpecification:

$$DeclSpecification \rightarrow IdentifierDec \mid FunctionDec$$

3.1.2 IdentifierDec:

$$IdentifierDec \rightarrow Datatype\ \ Identifier \quad '=' \ value;$$

3.1.3 Dataype:

There are two datatypes in this language: integer and string. Integers are positive and strings are either one or more character.

$$datatype \rightarrow \boldsymbol{Int} \mid \boldsymbol{String}$$

3.1.4 identifier

Check section 2.3.1

3.1.5 Value:

If the type of an identifier is an **Int**, then its value is an integer constant, otherwise is a string literal.

$$value \rightarrow \ IntegerConstant \mid StringLiteral$$

3.1.6 functionDec:

Users are able to define there own functions. Functions are defined in declaration part

$$functionDec \rightarrow \boldsymbol{Func}\ \ FuncName\ \ FuncBody\ \boldsymbol{EndFunc}$$

3.1.5 FuncName:

$$FuncName \rightarrow identifier$$

3.1.6 FuncBody:
Here all statements of a function is specified

$$FuncBody \rightarrow statement*$$

For *statement*, check section 3.3.1

3.2 WindowSpecification

This section is mandatory. It starts with " Window [" and ends with "]". It sets the general parameters of the output graphics window

$$WindowSpecification \rightarrow' [' WindowSize \quad ';' BackgroundColor \quad ';' ObjectColor';' ']'$$

3.2.1 WindowSize:
WindowSize sets the size of the output windows in pixels. ***Size=(****width in pixels, height in pixels****)***

$$WindowSize \rightarrow \boldsymbol{Size} \quad ' =' '(' integer \quad ',' integer ')'$$

3.2.2 BackgroundColor:
BackgroundColor sets the background color of the output window

$$BackgroundColor \rightarrow \boldsymbol{Bcolor} \quad ' =' Color$$

3.2.3 ObjectColor*:*
sets the color of shapes and texts

$$ObjectColor \rightarrow \boldsymbol{Ocolor} \quad ' =' Color$$

3.2.4 Color:
A window background or a shape color could be one of the following:

Black, White, Blue, Red

$$Color \rightarrow \boldsymbol{Black} \mid \boldsymbol{White} \mid \boldsymbol{Blue} \mid \boldsymbol{Red}$$

3.3 ObjectSection

This section is mandatory. It starts with " Object [" and ends with "]". Statements are added to this section.

$$ObjectSection \rightarrow' [' Statement* ']'$$

3.3.1 Statement:
There are 3 kids of statements. They are executed in sequence in the object part.

$$Statement \rightarrow ExpressionStatment \mid ConditionalStatement \mid IterationStatement$$

3.3.1.1 ExpressionStatement

$$ExpressionStatment \rightarrow Expression$$

For expression check section 3.3.3

3.3.1.2 ConditionalStatement
If the expression between parenthesis is true, the first statement is executed, otherwise the second statement is executed.

$$ConditionalStatement \rightarrow \textbf{\textit{if}} \quad '(' \quad 'EqualityTest \quad ')' \; statement \; \textbf{\textit{else}} \; statement \; \textbf{\textit{Endif}}$$

3.3.1.3 IterationStatement

$$IterationStatement \rightarrow \textbf{\textit{Loop}} \quad '(' \; integer \quad ')' \; statement \; \textbf{\textit{Endloop}}$$

3.3.2EqualityTest

Equality test returns 1 if both expressions are equal, and 0 otherwise

$$EqualityTest \rightarrow ArithExp \quad '=' \; ArithExp$$

3.3.3Expressions

There are three kinds of expressions: Assignment Expressions, ObjectCalls, and FunctionCalls

3.3.3.1 Assignment Expression:

$$AssingExp \rightarrow identifier \quad '=' \; (ArithExp|StringLiteral)$$

3.3.3.2 ArithExp:

Operations are left-associative. '/' and '*' have higher precedence than '+' and '-'

$$ArithExp \rightarrow ArithExp \quad '+' Term$$

$$|ArithExp \quad '-' \; Term$$

$$|Term$$

3.3.3.3 Term:

$$Term \rightarrow Term \quad '*' \; Term2$$

$$| \; Term \quad '/' \; Term2$$

$$|Term2$$

3.3.3.4 Term2:

$$Term2 \rightarrow Identifier$$

$$|IntegerConstant$$

3.3.3.2 Function calls:

$$FunctionCall \rightarrow \; FuncName$$

3.3.3.3 ObjectCall:

$$ObjectCall \rightarrow \textbf{\textit{Ellipse}} \quad '(' \quad 'ArithExp \quad ',' \; ArithExp \quad ',' \quad ArithExp \quad ',' \quad ArithExp)$$

$$| \; \textbf{\textit{Rectangle}} \quad '(' \; ArithExp \quad ',' \; ArithExp \quad ',' \quad ArithExp \quad ',' \quad ArithExp)$$

$$|\textbf{\textit{Line}} \quad '(' \; ArithExp \quad ',' \; ArithExp \quad ',' \quad ArithExp \quad ',' \quad ArithExp)$$

$$| \; \textbf{\textit{Text}} \; '(' \quad 'StringLiteral|identifier \quad ',' \quad ArithExp \quad ',' \quad ArithExp)$$

Ellipse (rx, ry, x, y)
draws an ellipse with horizontal radius rx, vertical radius ry and center at point(x, y)

Rectangle (w, h, x, y)
draws a rectangle with width w, height h, and the lower corner at point (x, y)

Line (x1, y1, x2, y2)

Draws a line from point (x1, y1) to point (x2, y2)

Text ("Hello World", x, y, s);
Print “Hello World” starting from point (x, y), with font size s

3.8 Lexical Scope:

Identifiers, objects and keywords all fall into the same name space. Therefore, names may not be repeated in the same program.

Doodle uses static, open scoping. A name begins life where it is declared in Declare section, and ends at the end of Object section. There is no nested scoping in Doodle.

Grammar

The following is a list of Doodle grammar . The start symbol is DoodleProgram

$$
\begin{aligned}
Identifier &\rightarrow Letter(Letter \mid Digit)^* \\
IntegerConstant &\rightarrow digit+ \\
StringLiteral &\rightarrow \ '"' \ \ Letter+ \ \ '"' \\
Letter &\rightarrow ['a'-'z' \ \ 'A'-'Z'] \\
Digit &\rightarrow ['0'-'9'] \\
DoodleProgram &\rightarrow Declaration? \ \ WindowSpecification \ \ ObjectSection \\
Declaration &\rightarrow \ \ '['DeclSpecification+']' \\
DeclSpecification &\rightarrow IdentifierDec \\
&\quad \mid functionDec \\
IdentifierDec &\rightarrow Datatype \ \ Identifier \ \ '='value';' \\
value &\rightarrow IntegerConstant \mid StringLiteral \\
datatype &\rightarrow \textbf{\textit{Int}} \mid \textbf{\textit{String}} \\
functionDec &\rightarrow \textbf{\textit{Func}} \ \ FuncName \ \ FuncBody \ \textbf{\textit{EndFunc}} \\
FuncName &\rightarrow identifier \\
FuncBody &\rightarrow statement* \\
WindowSpecification &\rightarrow ['WindowSize \ \ ';' \ BackgroundColor \ \ ';' \ ObjectColor';''']' \\
WindowSize &\rightarrow Size \ \ '=' \ '('integer \ \ ','integer \ ')' \\
BackgroundColor &\rightarrow \textbf{\textit{Bcolor}} \ \ '='Color \\
ObjectColor &\rightarrow \textbf{\textit{Ocolor}} \ \ '='Color \\
Color &\rightarrow \textbf{\textit{Black}} \mid \textbf{\textit{White}} \mid \textbf{\textit{Blue}} \mid \textbf{\textit{Red}} \\
ObjectSection &\rightarrow ['Statement* ']' \\
Statement &\rightarrow ExpressionStatment \mid ConditionalStatement \mid IterationStatement \\
ExpressionStatment &\rightarrow Expression* \\
ConditionalStatement &\rightarrow \textbf{\textit{if}} \ \ '(' \ \ 'EqualityTest \ \ ')' statement \ \textbf{\textit{else}} \ statement \ \textbf{\textit{Endif}} \\
IterationStatement &\rightarrow \textbf{\textit{Loop}} \ \ '('integer \ \ ')' statement \ \textbf{\textit{Endloop}} \\
EqualityTest &\rightarrow ArithExp \ \ '='ArithExp \\
AssingExp &\rightarrow identifier \ \ '=' \ (ArithExp \mid StringLiteral) \\
ArithExp &\rightarrow ArithExp \ \ '+'Term \\
&\quad \mid ArithExp \ \ '-'Term \\
&\quad \mid Term \\
Term &\rightarrow Term \ \ '*'Term2 \\
&\quad \mid Term \ \ '/'Term2 \mid \\
&\quad \mid Term2 \\
Term2 &\rightarrow Identifier \\
&\quad \mid IntegerConstant \\
ObjectCall &\rightarrow \textbf{\textit{Ellipse}} \ \ '(' \ \ 'ArithExp \ \ ','ArithExp \ \ ',' \ ArithExp \ \ ',' \ ArithExp) \\
&\quad \mid \textbf{\textit{Rectangle}} \ \ '('ArithExp \ \ ','ArithExp \ \ ',' \ ArithExp \ \ ',' \ ArithExp) \\
&\quad \mid \textbf{\textit{Line}} \ \ '('ArithExp \ \ ','ArithExp \ \ ',' \ ArithExp \ \ ',' \ ArithExp) \\
&\quad \mid \textbf{\textit{Text}}'(' \ \ 'StringLiteral \mid identifier \ \ ',' \ ArithExp \ \ ',' \ ArithExp)
\end{aligned}
$$