Language Reference Manual

EasyDraw

Stanislav Ivaciov

COMS W4115 – Spring 2009 (CVN)
Dr. Edwards

Contents

Introduction

EasyDraw programming language is designed to facilitate development of programming skills by novices. The language provides one-to-one mapping between general programming constructs and concepts and their graphical representation.

It has been shown that for many students it is easier to familiarize themselves with abstract concepts of Computer Science using graphical mnemonics. There is not much of such that is currently offered for undergraduate students just entering the field.

On the other side humanitarian and most other technical studies offer a wide range of visualization tools for their entry-level students.

Therefore EasyDraw strives to bridge the gap between computer and other sciences buy providing students with visualization techniques that can be used for introducing sub-routines, for and while loops, statements, and expressions.

Development and Execution

Programs in EasyDraw can be written with just a simple text editor that is capable saving files in ASCII format. Other that a text editor, such as Notepad, or VI, students will need EasyDraw interpreter that is available free of charge.

The EasyDraw interpreter reads source file and executes it on the fly without translating the source in to any student visible form.

Therefore development model for EasyDraw is as follows: a source file in EasyDraw is edited and saved, and then EasyDraw interpreter is invoke with this file as the parameter. Output is generated on the screen.

Language Manual

EasyDraw language is very simple to use and understand. It is designed specifically for novices and therefore lacks the complexity associated with general programming languages like C or Java.

Higher-Level Primitives

To facilitate understanding and learning EasyDra provides several higher-level primitive constructs such as circle, rectangle, square, and other similarly named primitives execution of which results in display of appropriate geometrical figure on the screen. The programmer specifies the rectangular coordinates of the figure and appropriate parameter as in the following example.

circle 50, 70, 45, 0

Execution of this statement results in display of a circle centered in (50, 70) x,y-coordinates with radius equals to 45, rendered in black color.
Types

EasyDraw strives to provide simple development experience with all the tools necessary to construct useful and comprehensible programs. For this purpose two types are provided: Number and Color. The language is statically typed.

Number

All variables of type Number are positive integers in the range from 0 to 2^{32} . Every variable of type Number must be declared. Declaration order is not important but it is recommended that students declare all variables in the beginning of a block as in C syntax following by their use. EasyDraw does not count on this, in fact it is similar to ECMAScript in that declarations can follow the use, and students are encourage to explore this feature after they master the C-like syntax.

For example, following code snipplet declares two variables of type Number with names x and y.

```
Number x;
Number y;
```

Variables can be given values in the declarations as so: Number x := 100;. Notice that EasyDraw uses := symbol to indicate assignment.

Color

Color type is a subset of Number type. EasyDraw distinbuishes between 256 colors beginning at 0 that is black and ending at 255 that is white. It statically checs that every variable of type Color does not get below or above this threeshold. There is no coercion or assignment from one type to another as in most other modern languages. This feature is designed to teach students to distinguisnhe between different types and to exercise caution in languages such as C or Java where coercion can lead to subtle bugs.

Following code snipplet declares a one variable of type Color and another variable of type Number followed by an attempt to assign one variable to another that will result in a syntax error.

```
Color c;
Number x;
c := x;  /* syntax error, declared by interpreter */
```

Program Structure

All programs in EasyDraw begin execution in the main method. The interpreted assumes the existence of such method in the source file and not providing it results in a syntax error.

For example following main method if does not appear to do much, begins execution of a EasyDraw program:

```
# file example1.ezd

main ()
{
}
```

If you type the above text in a file named example.ezd, save it, and execute it by passing it as a parameter to EasyDraw interpreter you will see a white screen that indicates successful interpretation of the file and readiness of the interpreter.

Other methods can be defined in a single .esd file. However, single file can be passed to the interpreter, this means that you program has to be written in a single file.

Other methods defined in a .esd file can be called from main method or other methods including recursive calls.

For example following code begins execution in main method, from which method named A is called from which method named B is called.

```
#file example2.esd

main()
{
        A();
}

A()
{
        B();
}

B()
{
```

```
}
```

Methods

Methods also known as sub-routines in EasyDraw group code elements in logically connected units. All files i.e., programs, must have at least one method named main to execute otherwise a syntax error will be declared by the interpreted.
Parameters can be passed between methods, but no parameters can be returned from a method, in other words all parameters are passed by value and no return values are allowed.
Following example illustrates previously stated concepts. Specifically, method named main is defined. It calls method A which call method B and passes it parameter x.

```
main()
{
A();
}

A()
{
        B(45);
}

B(Number x)
{
        Color c : = 0;
        Point x, c;
}
```

Scope

EasyDraw is statically scoped. All variables declared in a block of code will “hide” variable with the same name in the outer blocks. Syntax rules of the language are such that there only a single declaration with a particular name is allowed per block of code.

Following example declares two variables of type Number named x and y in the outer code block, following by redeclaration of x in the inner block.

```
{ /* outer block begins */
Number x := 10;
Number y := 20;
{ /* inner block begins */
        Number x := 15;
        /* Now x is 15 and y is 20 */
} /* inner block ends */
/* not x is 10 and y is 20 */
} /* outer block ends */
```

Comments

EasyDraw has single comment syntax. Starting with /* and ending with */ , anything in between the */ and the /* is considered to be a comment and is discarded during interpretation.
As an example consider the following code that declared a vatiable of type Color inside a comment. As is expected this declaration does not result in any execution.

```
/*
        This declaraiont will be ignored by the interpreter because it is inside a comment:
        Color x := 10;
*/
```

However following code will declare and assign number 10 to a variable of type Color during interpretation.

```
/* Following code will be executed because it is outside of the comment block */

Color x := 10;
```

Whitespace

ASCII character new line, tab, space, and new lines are all not considered by the EasyDraw interpreter. In fact they are removed before the interpreter sees the source program.

Following two code fragments are equivalent:

```
/* Fragment 1 */
        Number x := 1;
        Number y := 2;

/* Fragment 2 */
Number x:=1;Number y:=2;
```

However, whitespace is essential for the interpreter to distinguish between different tokens. Following code fragment is invalid because there is no whitespace between type and variable.

```
Numberx:=10; /* Syntax error */
```

Identifiers

Identifiers can be of arbitrary length. They must begin with a letter followed by zero ore more letters and digits. Following are some valid and invalid identifiers.
X /* valid */
xx01 /* valid */
0x /* invalid */
Keywords

Keywords are reserved identifiers. Programmers can not use names that are reserved by the language to name their identifiers.

if
then
else
for
while
number
color

Operators And Punctuation

Following is a precedence table for operators.

*, /	Multiply, Divide
+, -	Add, Subtract
:=	Assignment operation
&	Logical AND operator
|	Logical OR operator
=	Equivalence operator
<,>, <=, >=	Less than operator, Greater than operator, Less than or equal to, Greater that or equal to
!	Logical NOT operator
(,)	Left Parenthesis, Right Parenthesis
;	Statement delimeter.

Logical NOT operator is applied to and expression that has a type Boolean, even though EasyDra does not supply this type explicitly it can be obtained by performing comparisons between variables or literals of the same type.

Such as following comparison of two literal Numbers.

10 = 15; /* is false */

```
Number x := 10;
10 = x  /* is true */
```

Note that two different types can not be compared.

```
Number x := 10;
Color c := 10;

x = c;    /* produces syntax error */
```

References

Language Proposal For EasyDraw Programming Language