YAGL: Yet Another Graphing Language

Edgar Aroutiounian, Jeff Barg, Robert Cohen

July 14, 2014

1 Description of YAGL

YAGL is a new programming language for constructing graphics, with the intent of allowing programmers to construct graphics with various different output formats.

The language features a static type system with a void type that can conditionally downcast to the other primitives (Int, Array, etc..). Types are indicated by the type name declarator followed by the variable name. All types are objects. Each object holds an enumeration for what its type is, along with the underlying data. There is no string type. There are integer and array literals.

There are limited control structures: an iterated for loop, an if statement and break. A for loop can iterate on Stream and Array types and will yield a void type as the iterated item which needs to be downcast.

2 What Does YAGL Solve?

YAGL allows users to take JSON formatted data and produce clear and concise graphs. The properties of these graphs (such as color) can be manipulated utilizing algorithms. The output of the YAGL program can be determined by the user (SVG, ASCII, or PDF). YAGL is essentially and clean and minimal language for producing graphs quickly and easily.

3 Examples of YAGL Syntax

YAGL will have a void type, which cannot be operated upon. The void type can be conditionally downcast using the syntax:

```
if let b = a as Int {
        # scope of b is only in this function
}
```

This allows us to safely operate on JSON streams.

YAGL does not have a string type. In order to pipe in filenames to open JSON files, the open open keyword takes in $[n] which specifies the *nth* command line parameter. *open* constructs a Stream.

```
Stream s = open $0
```

YAGL has an iterated for loop and array literals:


```
Stream s = open $0
```

YAGL has an iterated for loop and array literals:


```
for a in [[4,1,5], 3, 5, [2, 3]] {
        if (let int_a = a as Int) {
        }
        elif (let int_a = a as Array) { }
}
```

KEYWORDS	OPERATORS	TYPES
if	++	Void
elif	+	Array
for	-	Canvas
in	*	Int
break	/	Stream
let		
in		
func		

Table 1: YAGL PRIMITIVES

4 Example YAGL Program

```
func makeGraph(Array: param1)
{
        Canvas myGraph = new Canvas()
        # 0 is ascii art graph, 1 is SVG meant for a browser, 2 is PDF?
        myGraph.graphType = 0
        # Opens up command line argument, presumably JSON.
        Stream s = open $0
        for (item in s)
        {       # Say there are only 3 enums of types
                # 0 is integer, 1 is array, 2 is canvas Object
                if (let integer_item = item as Int)
                {
                        myGraph.addRectAllParams(item)
                }
                elif (let array_item = item as Array)
                {
                        myGraph.addRect(item[0], item[1], item[2], item[3])
```

```
                }
        }
}

# Notice Last item is a nested Array
myData = [3, 4, 5, 1, [3, 1, 5, 6]]
build makeGraph(myData)

## Standard Library - preview graph with predefined parameters,
spits out ascii art graph to standard output preview(data)
```