rocky2

“I think a something-to-Pd compiler might be really interesting and useful - I've never seen anyone try that.”

- Miller Puckette, 2016

“I feel like a Kentucky Fried Idiot”

- Rocky Balboa, 1979

Un Sung Yoo	(uy2106)	Systems Architect, Tester
Anish King	(awk2123)	Project manager Language Guru, Tester

Contents

0 Outro

Discussion with Prof. Miller Puckette, creator of Pure Data (Pd), reveals the following:

“most probably people will want to mix rocky2 (for stuff like iterative algorithms) with straight-up Pd for interactivity design. So even though you're not likely to need the GUI, it would be most useful as a source of 'abstractions' in Pd.”

Pd is a visual programming language built on and around data flow for music simulation and signal processing. A basic Pd file is called a patch. rocky2 is a way to program in Pd minus the visual GUI. every rocky2 file compiles to a Pd patch.

rocky2 is wabisabi & minimalistic. If anything, rocky2 will help Pd users make patches more quickly. We are hoping the Pd community will devise new ways to use rocky2 and its aesthetic in ways the authors never imagined.

Cheers and yours in all things rocky2,

Anish & Un Sung

1 Lexical Conventions and Parsing

`/*  */`	multi-line comment
`=`	assignment
`+, -, *, /, %`	arithmetic operators
`->`	connection operator
`;`	statement ending
`{}, ()`	function block, scope
`<,>, ==, !=`	comparators
`int, string`	primitive data types
	function declaration
`include, as`	importing external subpatches from the same dir
`,`	separation marker for “for” loop and function parameters
`main`	main function implements universal canvas
`restore`	close a subpatch canvas
`obj`	objectize string values or functions
`for`	loop

Just a note: the ~ is a commonly used character in .pd files. It is used to define specialized audio-rate DSP functional objects, like `pink~`, `osc~`, `fft~`, etc. rocky2 will take this into account.

2 Types and Literals

2.1 Primitive Types

integer:

A literal such as 50 is a 64-bit signed integer.

We declare integers like:

```
int [name] = [integer value];
```

For example:

```
int a = 5;
```

string:

ASCII character sequences enclosed by double quotations and where special characters are handled via a backslash. A literal such as “gug” is a string.

The following escape sequences are supported:

```
\n newline                \r carriage return
\t horizontal tab   \v vertical tab
\\ backslash          \” double quotation
```

We declare strings like:

```
string [name] = “[string value]”;
```

For example:

```
string puredata = “is\nbodacious”;
```

2.2 Arrays

We *may* implement array capabilities. This is not important at the moment.

3 Operators and Expressions

3.0 Assignment and Variables

= is used to assign values of an expression to an identifier, returning a single assignment. Keep in mind this operation is not associative, so int `a = int b = 3`; is incorrect.

3.1 Arithmetic Operators

These operators are used mostly for “for” loops. Precedence (highest to lowest, separated by comma):

```
- (unary)
*, /, % (binary)
+, - (binary)
```

All operators are associative left wise. Recall rocky2 includes only integers, so modulus returns only integer values.

3.2 Logical and Relational Operators

These operators help in making conditional statements. Precedence (highest to lowest, separated by comma):

>,<
==, !=

3.3 Wiring Operator

-> is an operator used as follows. Pd is about data flow. We want to make it easy for users of rocky2 to be able to connect various objects in a patch and define the flow they want more precisely:

[source] [source outlet] -> [target] [target inlet]

Recall in Pd the object is given an index / count so it has a certain ordering when appearing in the text version of the .pd file. Similarly, outlets and inlets of an object are counted from 0 to however many there are on a particular object.

Wiring information is taken care of at the end of a universal canvas / .pd patch. rocky2 first lays out the objects in a user-specified order, then handles connections, or wiring. Therefore, our semantics needs to take into account a way to first handle the

Therefore, for example, say

- Ex. string a = “pink~”; /* all declaration and obj calls must be at the end */
- int b = 0;
- int c = 25;
- obj(a, b, c);
- obj (“output~”, 0, 25);

/* wiring must be at the end of statements */

- a -> “output~” 0 0;
- a -> “output~” 0 1;

3.4 Conditional Statements

3.5 For loop

- control statements
  loop statement
  for loop
    - for ([declaration], [condition], [expression]) {}
    - for (int i = 0, i < 5 i = i + 1) {
    - // statements
    - }

  when we use: when we call the same function in multiple times

4 Function Declaration

4.0 Default Patching

floatatm, obj, message, etc.

defining objects:

- obj ([name for obj], [x position], [y position]);
- Ex. obj (“pink~”, 0, 25);
- Ex. string a = “pink~”;
  int b = 0;
  int c = 25;
  obj(a, b, c);

Objects are virtually numbered in order of appearance in the file, starting from zero. Inlets and outlets of the objects are numbered likewise

for the objects, keep a count. use the current count for the new object put in, increment the count for the next object.

_____

all basic button features in the put menu. these are built in functions.
connects are on the bottom of the

we define #N in a wrapper

obj a = “pink~”
a.name
a.x
a.y

obj

4.1 Internal Subpatch

To make use of Pd’s internal subpatch capability, we define the following format:

```
pd [function name] (int xPos, int yPos, int xSize, int ySize, string
name_for_internal_subpatch, int font_size) {
        //statements
}
```

4.2 External Subpatch

External Function Declaration

- include karpluck~ as kp; importing subpatch .pd file from external source (same directory tho)
- when we declare function, it will create another .pd file which is the subpatch for main canvas. (if we don't have canvas for subpatch, we felt like there is no need for user-built in function)
- parameters for all functions are predefined as shown in bottom
-
- pd [function name]
- Ex. subpatch a (10, 10, 100, 100,foo~, 20) {

```
        string a = "pink~";
         int b = 0;
        int c = 25;

        obj(a, b, c);

        obj ("output~", 0, 25);

        a -> "output~" 0 0;

        a -> "output~" 0 1;
```

  }

- output: #N canvas 10 10 100 100 foo~ 12;
- How to use function:
- all functions are considered as obj, so you can wire up with other objects or other subpatch.
- To use a function, you have to call obj function to objectize the subpatch(function)
- obj ([subpatch name], [x axis], [y axis]);
- obj (a, 10, 5);
- obj ("temp", 3, 2);
- "temp" -> a 1 0;

5 Program Structure and Scope

5.1 File Extension

Pd file extension is .pd. rocky2 file extension is .dp.

5.2 File Format

all declaration for variables and objects must be before other statements to prevent error when compiling it to .pd because connect should be in top bottom for .pd
So we have to error handling for this

5.3 Scoping

- {} block for scope of statements
- () block for scope of functions and for loop setup
- everything inside {} or sometimes () for special cases is considered in the same scope

6 Built-in Functions / Standard Library

rocky2 will incorporate a main function like so:

```
main(int xPos, int yPos, int xSize, int ySize, int font size ) {
}
```

The purpose of this main function is to make it easier to bunch together the objects defined in this patch, within a built-in subpatch, and also the external subpatches that exist within the same directory as the rocky2 .dp file you wish to execute.

The execution procedure is as follows:

```
rocky2 helloworld.dp
```

rocky2 need not take any command line inputs, since ultimately the Pd standalone environment will handle the .pd file that rocky2 compiles to, compiling it and allowing for user interaction. It is important to stress here that rocky2 at the moment does not allow users to directly create sounds / analyze signals. For this, one still needs Pd standalone environment.

The Pure Data website http://puredata.info/ has a lot of information regarding various objects. Also, Google searches for specific results also yield a For the best use of rocky2, the authors suggest downloading the extended Pd package instead of the vanilla version, which lacks the open-source community's contributions to Pd over the years. That way, the possibilities and the application of rocky2 for purposes of enjoyment will be greater enjoyed.

7 Sample Code

```
/*pink.dp*/
/*
pink.dp compiles to a Pd patch that plays pink noise as varying
amplitudes based on a user-controlled slider. The noise may also
be turned off/on by the user.
*/

/*this nested for loop handles the creation of all necessary
objects as well as the wiring between them*/

int i = 0;
int j = 0;
int k = 0;

for(i = 0, i < 2, i++){

}

/* the main function */
main(int xPos, int yPos, int xSize, int ySize, int font size){

}
```