Photon - Programming Language Proposal

Akira Higaki (abh2171) - Manager
Calum McCartan (cm4114) - System Architecture
Franky Campuzano (fc2608) - Language Guru
Phu Pham (pdp2121) - Tester

February 1, 2021

1 Introduction

Photon is a language that is centered around modifying and editing images, similar to the functionality of Adobe Photoshop. The language is inspired by workflows in the visual effects industry, especially the node-based software Nuke. We aim to be able to provide functionality similar to that of Photoshop through a C-like syntax. The end goal is to have users upload JPG images (or multiple JPGs for videos) and put it through an automated and efficient editing pipeline through our language to output simple editing procedures quickly.

2 Language Structure

2.1 Primitives

Type	Description
Int	An integer
Float	A floating-point number
String	A sequence of characters
Boolean	True or False values
Pint	A special type of Int whose value can only range from 0 to 255
Null	Primitive that holds nothing

2.2 Structures

Type	Description
Array	A single unit of multiple grouped values
Pixel	A group of four Pint values
Image	A matrix of pixels

2.3 Keywords

Name	Description
this	Refers to the current instance of the class
func	Starts a function declaration
return	Sends the function result back to the caller

2.4 Operators

Type	Description
+	Appends primitives or images (point-wise)
-	Subtracts primitives or images (point-wise)
*	Multiplies primitives; images (point-wise with constants, images)
/	Divides non-zero primitives or images (point-wise)
==, <=, >=, <, >	Compares primitives
[ ]	Array creation and element calling

2.5 Functions

Name	Description
load(String filepath)	Loads an image into the program
output(String filename)	Saves an image
flip(Image sample)	Reverses the image along the horizontal axis
rotate(Image sample)	Rotates the image 90 degrees to the right

2.6 Flow Control

Type	Description
if, elif, else	Conditional statements
for, while	Iterative Statements

3 Language Features

3.1 Pint

The 'pint' primitive type (pixel-integer) is an unsigned 8-bit integer with a range of 0-255. This is ideal for efficiently storing each of the four RGBA components of a pixel. In addition, we would like to automatically prevent integer overflow on the pint type (eg. 200 + 60 = 255). This will be useful for the operations that are performed on pixels.

3.2 Pixel

The 'pixel' type is made up of 4 components: red, green, blue, and alpha (RGBA). The alpha layer will allow us to combine images in more powerful ways. It will be possible to combine pixels using the '+' and '-' operators, but

this will not add the components of the image in the same was as normal matrix addition. Instead, the colour components will be added using:

```
red1 * (alpha1/255) + red2 * (alpha2/255)
```

The alpha values will be combined using:

```
alpha1 + alpha2
```

Our 'pint' type will ensure that the resulting values are clamped between 0 and 255.

3.3 Image

The 'image' type is a 2D array (matrix) of pixels. It will be possible to combine images using the '+' and '-' operators, and this will combine each pixel in the images as described above.

3.4 Types

All variables must be declared with their type (static typing). When using operators on two variables they must both be of the same type, with the following exceptions:

- Any numeric types can be used together, and will return the more precise type (eg. int + float = float)
- A string and numeric type will be concatenated when used with the '+' operator

3.5 Syntax

- The end of a statement is denoted by a semicolon
- The language is not sensitive to indentation
- A line comment is denoted using the # symbol

4 Example Code

4.1 Maximum of an array

This is a simple subroutine that finds the largest element in an array.

```
func int maxElement(inArray) {
    int max = 0;
    for (int i = 0; i < inArray.length; i = i + 1) {
        if (inArray[i] > max) {
            max = inArray[i];
        }
    }
    return max;
}
```

4.2 Flipping an image

Below is a subroutines that uses one of the built-in functions to flip an image.
This demonstrates loading/saving in an image.

```
func string flipImage(filePath) {
    image testImage = load(filePath);
    testImage = flip(testImage);
    returnPath = output(testImage);
    return returnPath;
}
```

4.3 Image addition

Below is a subroutine that uses the alpha values to combine two images together.

```
func image halfHalf(image1, image2) { #600x600 pixel image, alphas = 255 by default
    for (int i = 0; i <= 300; i = i + 1) {
        for (int j = 0; j <= 600; j = j + 1) {
            image1[i,j].alpha = 0;
            image2[i+300,j].alpha = 0;
        }
    }
    image3 = image1 + image2;
    return image3; #left side of image1, right side of image 2
}
```

5 References

VSCOde - Fall 2018 Project
Coral - Fall 2018 Project
Nuke - Video Editing Software