VSCOde

Jessica Cheng (jc4687)
Anna Lu (ajl2256)
Hana Mizuta (hm2694)
Kenneth Yuan (kky2114)
Spencer Yen (ssy2121)

September 19, 2018

1 Introduction and Motivation

VSCOde is a language that is designed to analyze and manipulate images. The inspiration behind our language is the high definition mapping technology used by autonomous vehicles, only our scope is on a much more rudimentary level.

Users will be able to upload JPG image files and modify the image in a number of ways, such as by rotating the image, adjusting colors (saturation, brightness, contrast, etc.), basic edge detection, and creating custom filters. The images will be represented as matrices in our language to grant users greater control over the individual pixels in an image and allow for easy application of filters through matrix transformations.

The syntax of our language draws from C and Python because of its matrix-focused nature (and MatLab and R are too difficult). We want to take the simplicity of Python standard library functions, but keep C's use of semicolons and brackets to detect statement endings and clauses.

2 Language Details

2.1 Primitives

Type	Description
int	32-bit signed integer
double	64-bit float point number
bool	8-bit boolean variable
string	Array of ASCII characters

2.2 Structures

Type	Description	Syntax
tuple	Immutable sequence of any mixture of object types	`(object1, object2, ...)`

matrix	Mutable data structure storing multi-dimensions of objects	`[`
		`  -1, -1, -1;`
		`  -1, 8, -1;`
		`  -1, -1, -1;`
		`];`

2.3 Keywords

Keyword	Description
this	Object self-reference
func	Indicates function declaration
return	Return statement
void	Indicates that function has no return value
true	Boolean keyword for true
false	Boolean keyword for false
->	Denotes return type of function

2.4 Operators

Operator	Description
+	Addition (scalar and matrix)
-	Subtraction (scalar and matrix)
*	Multiplication (scalar and matrix)
/	Division (scalar and matrix)
%	Modulo (scalar)
++, –	Increment (scalar), Decrement (scalar)
+=, -=, *-, /=	Add, subtract, multiply, divide left by/to right
<, >, <=, >=	Greater than, less than, greater than or equal to, less than or equal to

2.5 Functions

Name	Description	Return Type
print(string s)	Prints argument to standard output	void
dim(matrix m)	Gets the dimensions of an object	(int, int)
load(string name)	Loads an image into a 3-tuple of red, green, blue matrices	(matrix, matrix, matrix)
save(matrix r, matrix g, matrix b, string s)	Saves 3 matrices corresponding to RGB values as a jpg image with name s	bool

2.6 Built-in Matrix Transformations

Method	Description
transpose(matrix m)	Transposes matrix m
replace(matrix m, int a, int b)	Replaces every instance of a in matrix m with b
multiply(matrix a, matrix b)	Multiplies matrix a by matrix b
rotate(matrix a, double deg)	Rotates matrix a by deg degrees by multiplying the matrix by [[cos(deg), sin(deg)], [-sin(deg), cos(deg)]]
convolute(matrix a, matrix b)	Convolutes matrix a with matrix b

2.7 Flow Control

Statement	Description	Syntax

if / elif / else	Conditional statements	`if (condition) {`
		`    ...`
		`} elif (condition) {`
		`    ...`
		`} else {`
		`    ...`
		`}`

while / for	Iterative statements	`while (condition) {...}`
		`for element in matrix {...}`

continue / break	Branching statements	`continue;`
		`break;`

3 Language Features

Indentation: Our language will not be whitespace-sensitive

Index: Using zero index

Semicolons: Semicolons can indicate the end of a matrix row or the end of a statement

Single-line Comment: Denoted by //

Multi-line Comment: Denoted by /* */. Can nest comments as follows: /* /* */ */

Accessing Elements in a Matrix: If we define a matrix with name M, we can access its elements with the following syntax:

```
matrix M =
[
  -1, -1, -1;
  -1,  8, -1;
  -1, -1, -1;
];

print(M[1, 2]) // should print element in 2nd row, 3rd column: -1
```

4 Code Samples

4.1 GCD Algorithm

```
// greatest common denominator function in VSCOde
func gcd (int m, int n) -> int {
    while (m > 0) {
        int c = n % m;
        n = m;
        m = c;
    }
    return n;
}
```

4.2 Grayscale

```
/* multi-line comment
function to grayscale images */

func applyGrayscale (String imageName) -> void {
    // read image into a matrix
    matrix r, g, b;
    r, g, b = load(imageName); // load returns a tuple

    // weighted method of grayscale
    multiply(r, 0.3);
    multiply(g, 0.59);
    multiply(b, 0.11);

    save(r, g, b, "new.jpg");
}
```

4.3 Edge Detection

```
func applyAllDirectionEdgeDetection (string imageName) -> void {
    // read image into a matrix
    matrix r, g, b;
    r, g, b = load(imageName); // load returns a tuple

    // all direction edge direction matrix
    matrix edgeDetection =
    [
      -1, -1, -1;
      -1,  8, -1;
      -1, -1, -1;
    ];

    convolute(r, edgeDetection);
    convolute(g, edgeDetection);
    convolute(b, edgeDetection);

    save(r, g, b, "new.jpg");
}
```

5 References

FaceLab Report.
Lane Detection for Self-Driving Cars with OpenCV.
Lode's Computer Graphics Tutorial.