Reptile (.rt)

Project Proposal

Aileen Cano (ac4440), Aviva Weinbaum (aw3156),
Lindsey Weiskopf (ltw2115), and Hariti Patel (hvp2105)

February 2021

1 Introduction

Reptile is a programming language that is intended to support libraries that streamline the process of creating simply-coded graphics. As more children are learning computer science at a younger age, there is a demand for simple programming languages that teach computer science principles in a digestible and visual manner. Languages typically labeled for beginners like Scratch and Swift Playgrounds teach kids to code by showing immediate visual results from code – whether that is a simple square or a complex environment built upon existing code blocks. Further, libraries like Turtle graphics add novelty to simple image-building operations by showing a turtle drawing the desired shape. The goal of Reptile is to build upon the success of these "beginner" programming languages to provide immediate gratification to the coders through graphics.

2 Description

Reptile is a geometric figure drawing language built on a standard canvas and pointer data type. The language is object-oriented, allowing for the creation of libraries to build on the existing language architecture to create different kinds of graphics. It is procedural and imperative. The language will use java-like syntax and is strictly typed.

To showcase the object-oriented nature and intended purpose of Reptile, we will create the library Tortoise. This library will use the canvas and the pointer provided by the language to create a tortoise object with simple functions. Ideally, this would be easy enough for a beginner programming to use while yielding a tangible image result.

Features of Reptile

- Simple arithmetic and functional operations using objects
- Creation of canvas with a movable pointer
- Ability to mark pixels with different colors
- Produce an SVG

Features of the Tortoise library

- Pixel manipulation within the canvas and all features from main language
- Functions to move the tortoise specifying pixels, angles, and colors
- Encapsulation of vector calculations to provide complex functions with simple calls

Other possible user-constructed libraries

- Lizard - Library for drawing simple shapes
- Snake - Library for creating curved shapes and lines
- Chameleon - Library for creating color gradients

3 Simple Data Types

Simple Data Type	Details	Example
int	any signed integer	3
float	any floating point decimal	0.3
boolean	0 or 1	0
string	standard string	”hello”

4 Complex Data Types

Complex Data Type	Description	Constructor Parameters	Functions
RGB	List of 3 color values ranging from 0 to 255	int r: red value int g: green value int b: blue value	
List	Standard array		**list.length()**
Canvas	Two-dimensional array of pixels which Pointers act on	int x: number of pixels long int y: number of pixels high	**canvas.x**: length of canvas **canvas.y**: height of canvas **canvas.close()**: stop editing canvas and generate SVG. There will be as many SVGs produced as there are canvas.close() calls.
Pointer	Pen used to draw pixels and move around on canvas	Canvas c: canvas on which pointer will draw int x: starting x coordinate int y: starting y coordinate	**pointer.x**: x coordinate **pointer.y**: y coordinate **pointer.color(RGB rgb)**: set color for future markings **pointer.pixel**: mark pixel with color

5 Keywords

Keyword	Example
if/else	if(1) { }
for	for (int i = 0; i < 5; i++) { }
while	while(1) { }

6 Operators

Operator	Usage	Example
=	Assignment	int a = 3
==, ! =	Equality	a == b
<, >, <=, >=	Comparison	2 < 3
+, -, *, /, %	Addition, subtraction, multiplication, division, modulo	a + b
++, –	Increment/decrement int by 1	a++
&&, ||, !	Logical AND, OR, NOT	a && b
/\	Comment	/\This is a comment

7 Tortoise Library Functions

Function	Description
draw(int pixels, int degrees, RGB rgb)	colors *pixels* number of pixels in the direction of *degrees* the RGB value. This function is useful to minimize loops written in Reptile.
set(int x, int y)	changes Tortoise object's coordinates to (x,y)

8 Sample Code

gcd.rt

```
/\ Reptile retains basic Java functionality

int gcd(int a, int b) {
    if (b==a) {
        return a;
    }
    else {
        return gcd(b, a%b);
    }
}
```

main.rt

```
/\ Building basic shapes with Reptile

/\ make a canvas
Canvas canvas = new Canvas(100,200);
Pointer ptr = new Pointer(canvas,0,0);
RGB blue = new RGB(0,0,255);

/\ set color for drawing
ptr.color(blue);

/\ point to the right and start drawing
ptr.point(90);
for (int i = 0; i < 50; i++) {
    ptr.pixel(ptr.x, ptr.y);
    ptr.x ++;
}
ptr.point(180);
for (int i = 0; i < 70; i++) {
    ptr.pixel();
    ptr.y ++;
}
ptr.point(270);
for (int i = 0; i < 50; i++) {
    ptr.pixel();
    ptr.x --;
}
ptr.point(0);
for (int i = 0; i < 70; i++) {
    ptr.pixel();
    ptr.y --;
```

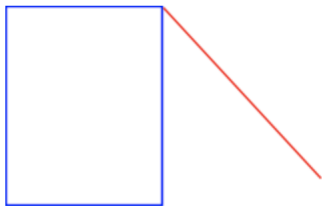
```
}

/\ diagonal line:
ptr.x = 50;
ptr.y = 0;
ptr.color(new RGB(255,0,0));
ptr.point(135);

for (int i = 0; i < canvas.x; i++) {
    ptr.pixel();
    ptr.x ++;
    ptr.y ++;
}

canvas.close();
```

Output:



mainWithTortoise.rt

```
/\ This program accomplishes the same thing

import Tortoise;

Canvas canvas = new Canvas(100,200);
Tortoise tortoise = new Tortoise(canvas,0,0);

tortoise.draw(50,90,new RGB(0,0,255));
tortoise.draw(70,180,new RGB(0,0,255));
tortoise.draw(50,270,new RGB(0,0,255));
tortoise.draw(70,0,new RGB(0,0,255));

tortoise.set(50,0);
tortoise.draw(50,135,new RGB(255,0,0));

canvas.close();
```

Output: