Programming Languages and Translators
Project Proposal
23 January 2021

1. Team
    a. Manager: Sofía Sánchez-Zárate (ss5664)
    b. Language Guru: Jason Cardinale (jrc2241)
    c. System Architects: Ben Synder (bs3148) & Evan Tilley (elt2141)
    d. Tester: Michael Winitch (mlw2173)

2. Overview
    a. A simple OOP language for creating 2D animated graphics similar to p5.js using a blend of Python and Java syntax. It will streamline the process of creating and animating primitive objects (i.e. rectangles, circles, triangles, etc.) as well as custom objects (spline curves, polygons, etc.). The output will be a GIF (graphics interchange format) since it uses lossless compression to shrink files, making them small enough to put on the web or send as a message.

3. Language Details
    a. Coordinate System
        i. The origin point of the canvas is going to be the lower left, but it can be customized by the user
        ii. A default center will be (width/2, height/2) which is where shapes will be created
    b. Types and Operations
        i. Types
            1. `num` → any number
            2. `bool` → boolean
            3. `char` → ASCII characters
            4. `string` → string
            5. `array` → array using [] not {}
        ii. Inbuilt objects/types:

Type	Description	Example
pt()	Object that represents a point	pt x = pt(4,3)
shape()	Default object parent of all builtin shapes that defines thickness which extends to all other objects	shape(thickness) shape(3)
square()	Defined from the center and given a size square(pt(x,y), size)	square(pt(0,0), 3)
rect()	Like a square	rect(center, width, height); rect(pt(0,0), 2, 3)
triangle()	3-pointed polygon	triangle(center, width, height); triangle(pt(4,5), 5, 8);
circle()	Defined by its center point and radius	circle(center, radius) circle(pt(2,3), 5)
ellipse()	An oval, like a circle, but fatter. Defined by its center point and an “a” and “b” which are the major and minor radii	ellipse e = ellipse(pt(0,0), 4, 5);
line()	A straight line defined by two points	line(pt(x,y), pt(x,y)) line L = new line(pt(1,2), pt(4,5))
canvas()	Define a 2D plane where shapes are required to be located within	canvas c = canvas(alignment: [.topLeft, .bottomLeft, .topRight, .bottomRight, .center], width: num, height: num) canvas([pt(2,2), pt(4,5), pt(1,1), pt(6,7)], 80, 100)
polygon()	A plane figure with at least three lines. Pass in a number of points and an array of their values as the parameters. Points are connected in	polygon(number, points) polygon([pt(1,1), pt(2,2,), pt(3,3), pt(5,4)])
regagon()	Creates a regular x sided polygon (such that all of the sides are the same length)	regagon(9)
spline()	A curve with several anchor points	spline s = spline([pt(x,y)])

iii. Manipulating objects:
   1. `x.animateTo(pt(x,y))`
   2. `x.shrink(factor: 5)`
   3. `x.rotate(degree: 6)`
   4. `x.animateTo(pt(x,y), speed: 24, scale: 0.6)`

c. Keywords
   i. Defining functions: `return_type function_name {...}`
   ii. Conditional
      1. `if`
      2. `elif`
      3. `else`
   iii. Control Flow
      1. `while (condition) { … }`
      2. `for (num i = 0; i < 10; i++) { … }`
      3. `if {...} elif {...} else {...}`
      4. `continue`
      5. `break`
      6. `try {...} catch {...}`
      7. `raise`
      8. `pass`
   iv. Logic
      1. `and` → compare two values and return True if they are equal to one another
      2. `or`
      3. `True` or `true`
      4. `False` or `false`
      5. `!` → not
      6. Operators:

Operator	Symbol	Example
Assignment	=	num x = 5
Equality	=?	a =? b
addition	+	a + b
subtraction	-	a - b
ternary	val ? bool : val	a =? b ? res1 : res2
multiplication	*	a * b
exponent	^	a^2

modulo	%	a % b
Greater than	>	a > b
Less than	<	a < b
Greater equal to	>=	a >= b
Less equal to	<=	a <= b

d. Comments

   i. `// This is an example of a single line comment, starts with '//'`

   ii. `/* Multi line comments are bounded by slash asterisk and asterisk slash */`

4. Sample Code

```
/* This is a GCD algorithm */

num gcd(int a, int b) {
	while (a != b) {
		if (a > b) {
			a = a - b;
		} else {
			b = b - a;
		}
	}
	return a;
}

/* This is a function to animate a square */

void animateSquare() {
	square x = square(pt(0,0), 2);
	// moves to 5,6 with speed of 24 and scales down while doing that
	x.animateTo(pt(5,6), speed: 24, scale: 0.6)
}
```