Instituto Tecnológico y de Estudios Superiores de Monterrey

Propuesta Inicial de Compilador: Light Language

Diseño de Compiladores
Maestra Elda Quiroga
Equipo #15

Luis Alberto Lamadrid Tafich	A01191158
Enrique Octavio Hernández Chávez	A01185423

28 de Febrero del 2016

1. Visión y Propósito del Proyecto

La visión a largo plazo de este proyecto es ayudar a las escuelas en México a tener una herramienta de uso fácil para poder poner a la programación como un curso esencial dentro de las primarias y secundarias. Asimismo, al tener el compilador buena documentación, proveer a la comunidad desarrolladora un buen ejemplo de como realizar su propio lenguaje gráfico y como contribuir al nuestro para su mejoramiento.

2. Objetivo del Lenguaje

Este lenguaje tiene como uno de sus objetivos principales el proveer a audiencias de todas las edades que sean ajenas a la programación y tecnología una manera fácil de aprender estos conceptos básicos a través de un lenguaje de programación con sintaxis descriptiva y utilizando retroalimentación de las salidas de la ejecución de este mediante una interfaz gráfica.

3. Requerimientos del Proyecto

3.1 Componentes Léxicos del Lenguaje

Palabras Reservadas						
`for`	`in`	`for_each`	`var`	`and`	`or`	`watch`
`when`	`default`	`while`	`is`	`true`	`false`	`function`
`boolean`	`integer`	`int`	`decimal`	`string`	`end`	`returns`
`return`		`program`	`do`	`loop`	`mod`	`if`
`unless`	`line`	`point`	`circle`	`square`	`rectangle`	`triangle`
	`polygon`	`main`	`star`	`light`	`draw`	

Tipo de Constante	**Expresión Regular**
`id`	`[a-zA-Z](_?([a-zA-Z]|[0-9]))*`
`constante entera`	`" ^( \r | \n | " )* "`
`constante decimal`	`[0-9]+`

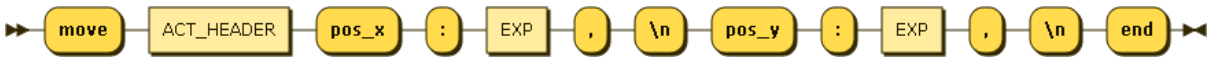
constante string	[0-9]+.[0-9]+

3.2 Diagramas de Sintaxis

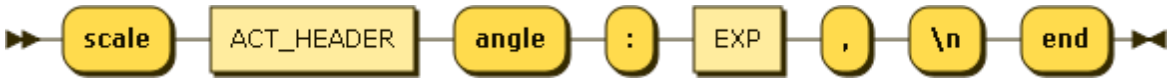
NOTA
Entrar a este sitio para poder ver los diagramas de forma dinámica http://bit.do/light_compiler

ACT_HEADER

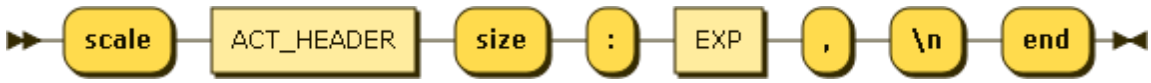
ACT_MOVE



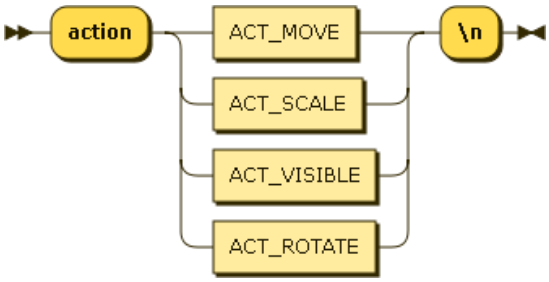
ACT_ROTATE



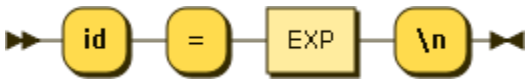
ACT_SCALE



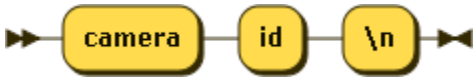
ACTION



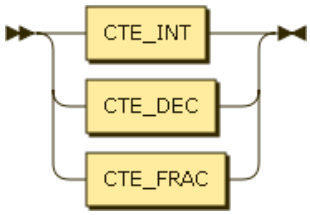
ASSIGNMENT



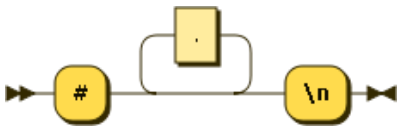
CAMERA



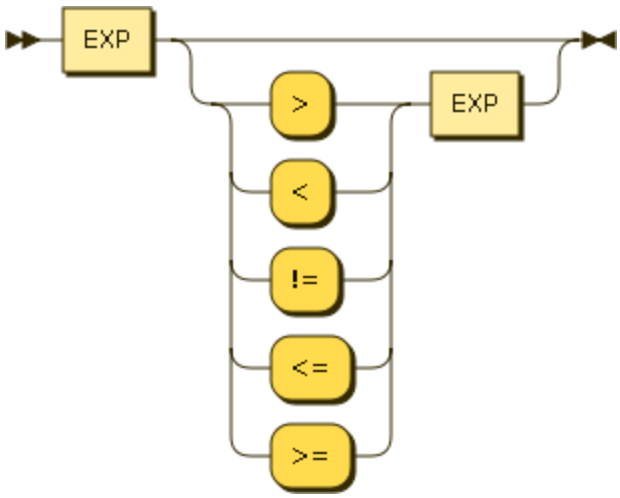
CNT_PRIM



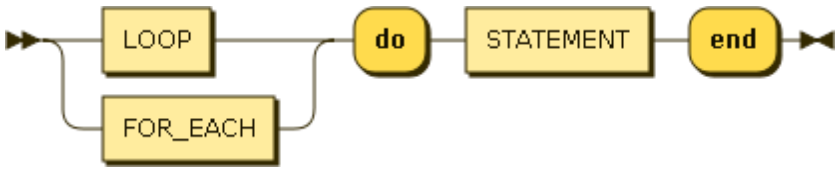
COMMENTS



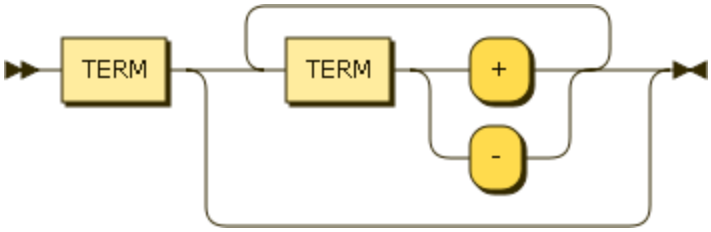
CONDITION



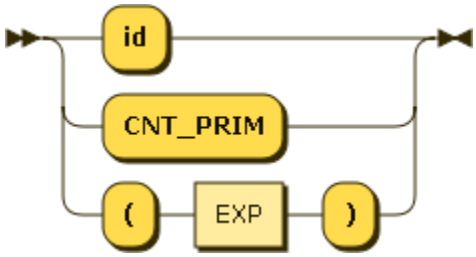
CYCLE



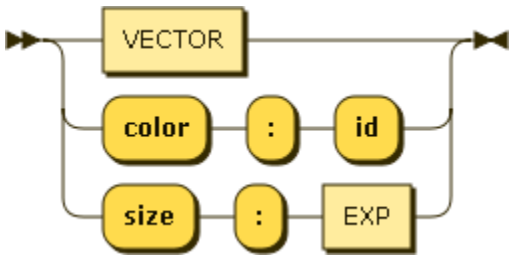
EXP



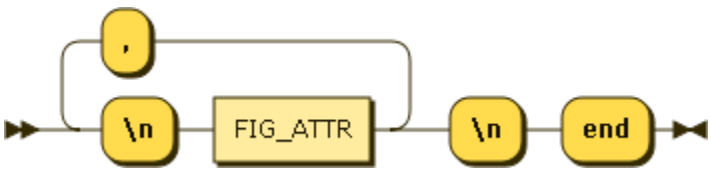
FACTOR



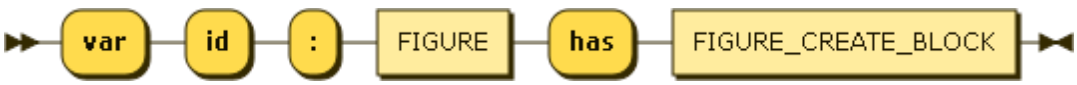
FIG_ATTR



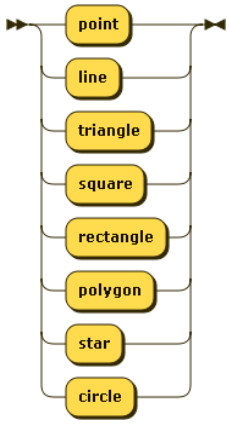
FIG_CREATE_BLOCK



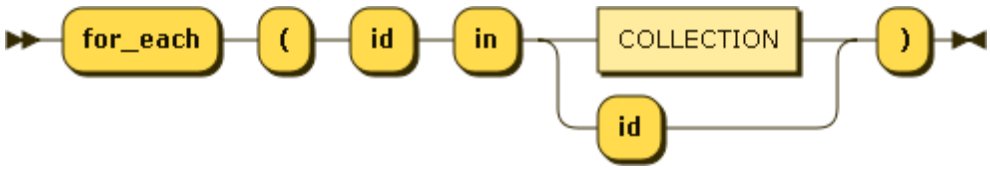
FIGURE_CREATIONS



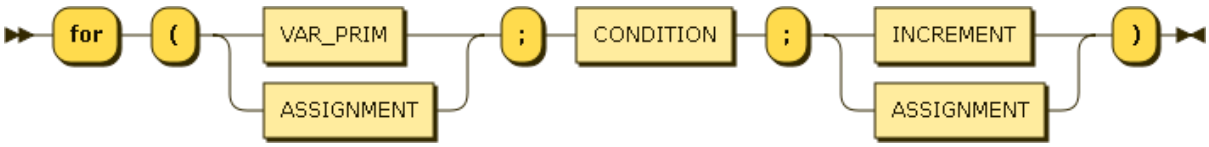
FIGURE



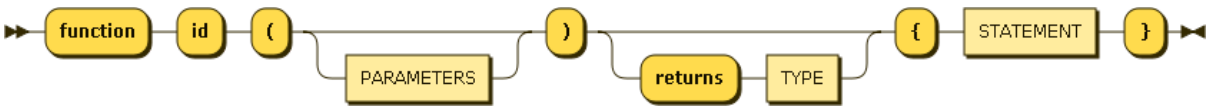
FOR_EACH



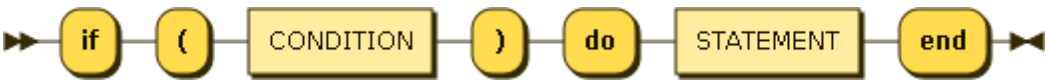
FOR



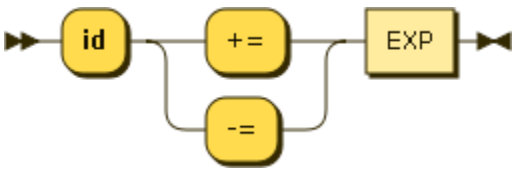
FUNCTION



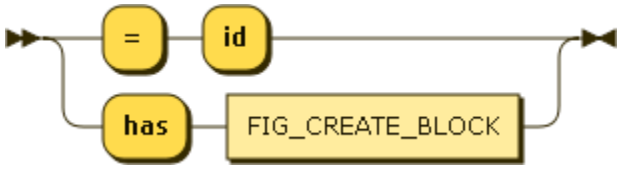
IF



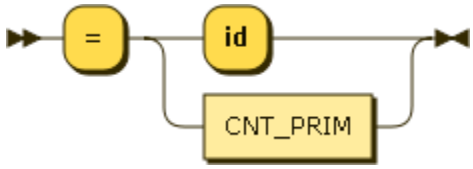
INCREMENT



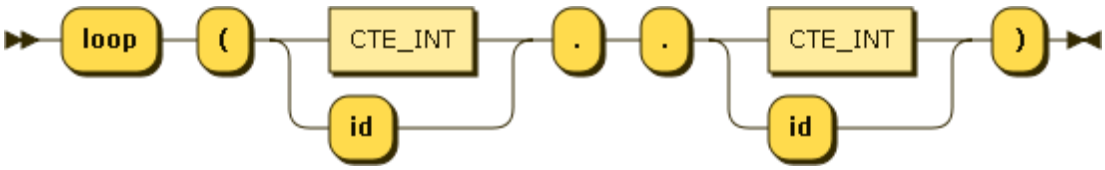
INIT_FIG



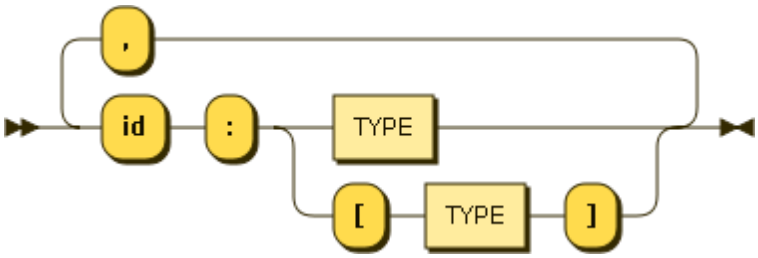
INIT_PRIM



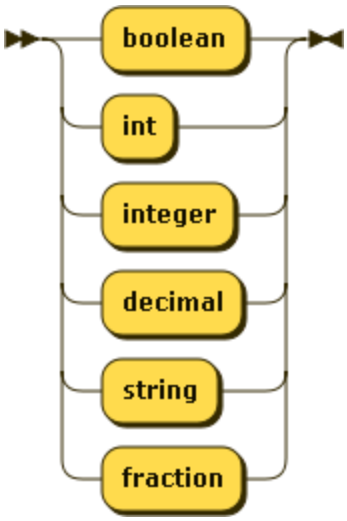
LOOP



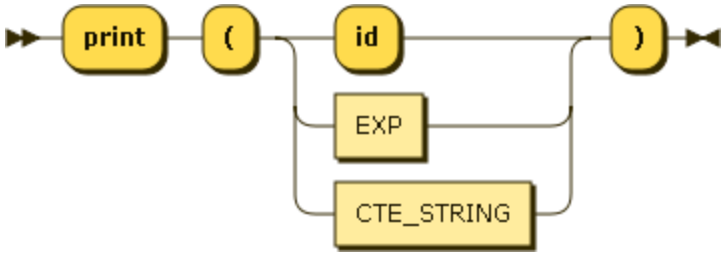
PARAMETERS



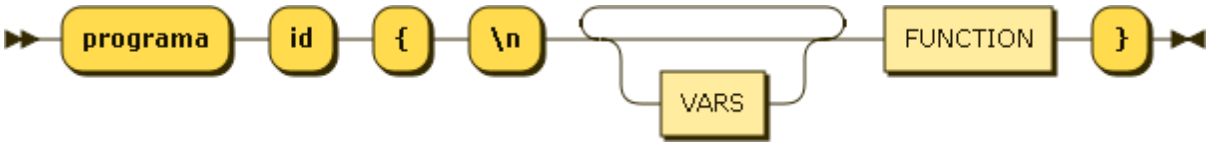
PRIMITIVE



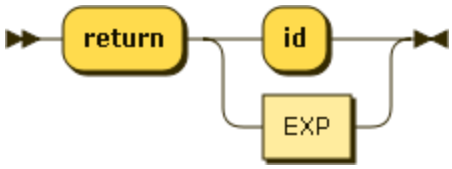
PRINT



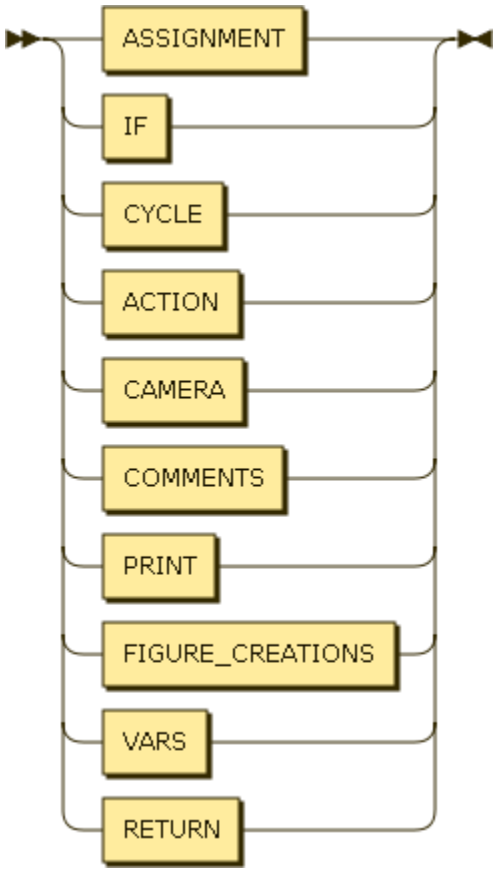
PROGRAM



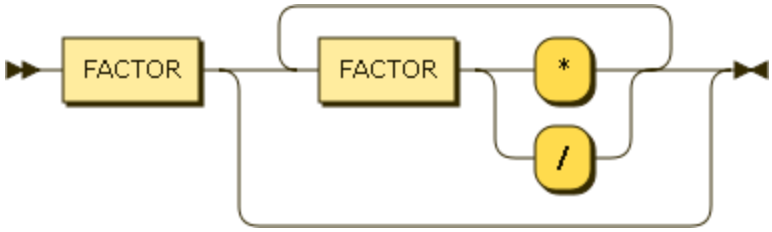
RETURN



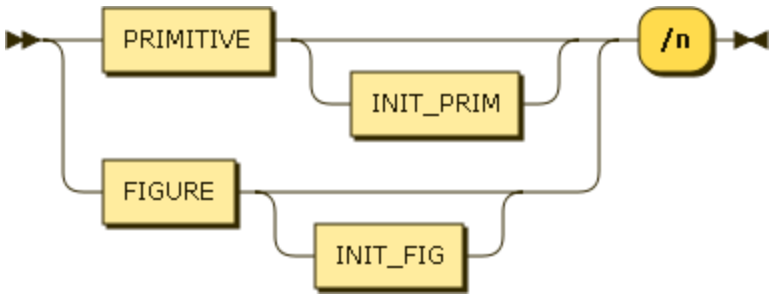
STATEMENT



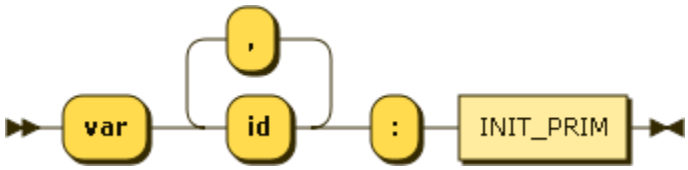
TERM



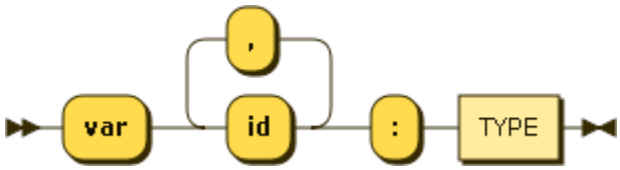
TYPE



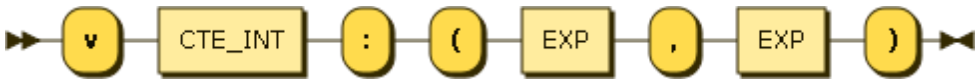
VARS_PRIM



VARS



VECTOR



Expresiones Regulares

```
PROGRAM ::= 'programa' 'id' '{' '\n' (VARS?)+  FUNCTION '}'

TYPE ::= (PRIMITIVE  (INIT_PRIM)? | FIGURE  (INIT_FIG)?) '/n'

PRIMITIVE ::= ('boolean' | 'int' | 'integer' | 'decimal' | 'string' | 'fraction' )

FIGURE ::= ( 'point' | 'line' | 'triangle' | 'square' | 'rectangle' | 'polygon' | 'star' |
'circle')

FUNCTION ::= 'function' 'id' '('    (PARAMETERS)?    ')' ('returns' TYPE )?    '{' STATEMENT '}'

PARAMETERS::=  ( ('id' ':' ((TYPE)|('[' TYPE ']'))  ) ( ',' 'id' ':' ((TYPE)|('[' TYPE ']')))* )

ASSIGNMENT ::= 'id' '=' EXP '\n'

CYCLE::= (LOOP | FOR_EACH)  'do' STATEMENT 'end'

LOOP ::= 'loop' '(' (  CTE_INT |  'id' ) '.' '.' (  CTE_INT |  'id' )  ')'

FOR_EACH ::=  'for_each'  '(' 'id' 'in' ( COLLECTION | 'id' ) ')'

FOR ::= 'for' '(' ( VAR_PRIM | ASSIGNMENT) ';' CONDITION ';' (INCREMENT | ASSIGNMENT) ')'

ACTION::= 'action' (ACT_MOVE | ACT_SCALE | ACT_VISIBLE | ACT_ROTATE) '\n'

ACT_HEADER::= 'id' 'do' '\n' ('begins' ':' EXP ',' '\n') ('ends' ':' EXP ',' '\n')

ACT_MOVE ::= 'move' ACT_HEADER ('pos_x' ':' EXP ',' '\n') ('pos_y' ':' EXP ',' '\n') 'end'

ACT_SCALE ::= 'scale' ACT_HEADER ('size' ':' EXP ',' '\n') 'end'

ACT_ROTATE ::= 'scale' ACT_HEADER ('angle' ':' EXP ',' '\n') 'end'

ACT_VISIBLE ::=  ('hide' | 'show') ACT_HEADER 'end'

CAMERA::= 'camera' 'id' '\n'

CONDITION::= EXP ('>' | '<' | '!=' | '<=' | '>=') EXP )?

EXP::= (TERM) (TERM ('+'| '-'))*

TERM::= (FACTOR) (FACTOR ('*'| '/'))*

FACTOR::= ('id' | 'CNT_PRIM' | ('(' EXP ')'))

INCREMENT::= 'id' ( '+=' | '-=' ) EXP

IF::= 'if' '(' CONDITION ')' 'do' STATEMENT 'end'

STATEMENT ::= ( ASSIGNMENT | IF | CYCLE | ACTION | CAMERA | COMMENTS | PRINT| FIGURE_CREATIONS |
                VARS | RETURN)

VARS ::= 'var' (('id') (',' 'id')*)  ':' TYPE

VARS_PRIM ::= 'var' (('id') (',' 'id')*)  ':' INIT_PRIM

INIT_PRIM ::= '=' ('id' | CNT_PRIM)

INIT_FIG::= ( ('=' 'id') | ('has' FIG_CREATE_BLOCK ))

FIG_CREATE_BLOCK ::= (  ('\n' FIG_ATTR) ( ',' '\n' FIG_ATTR)* ) '\n' 'end'

FIG_ATTR::= (VECTOR | ('color' ':' 'id') | ('size' ':' EXP))

VECTOR::= 'v' CTE_INT ':' '(' EXP ',' EXP ')'

CNT_PRIM::= ( CTE_INT | CTE_DEC | CTE_FRAC )
```

```
RETURN ::= 'return' ( 'id' | EXP )

COMMENTS::= '#' (.)* '\n'

PRINT::= 'print' '(' ('id'| EXP | CTE_STRING )  ')'

FIGURE_CREATIONS::= 'var' 'id' ':' FIGURE 'has' FIGURE_CREATE_BLOCK
```

3.3 Principales Características Semánticas

Conversión en Suma (+)					
	boolean	int	decimal	fraction	string
boolean	ERROR	ERROR	ERROR	ERROR	ERROR
int		int	decimal	fraction	string
decimal			decimal	decimal	string
fraction				fraction	string
string					string

Conversión en Resta, Multiplicación, y División (-,*,/)					
	boolean	int	decimal	fraction	string
boolean	ERROR	ERROR	ERROR	ERROR	ERROR
int		int	decimal	fraction	ERROR
decimal			decimal	decimal	ERROR
fraction				fraction	ERROR
string					ERROR

3.4 Funciones Especiales

Nombre	Explicación
camera	Dibuja los objetos en pantalla.
light	Es nuestra ‘main’ function.
action <act_name>	Las declaraciones de acciones de figuras podrían ser consideradas funciones especiales para el lenguaje. Estas hacen que las figuras se muevan de acuerdo a los atributos de la acción.

3.5 Tipos de Datos y Limitantes

Primitivos:	boolean	int	decimal	fraction	string
Figuras:	point	line	triangle	square	rectangle
	polygon	star	circle		

3.6 Código Ejemplo

```
program miDibujo {

	var global:int = 0

	function light() {

		sayText(text: “hello”)
		var numberOne:int = addOne(number: 25)

		#*
			Comentario de bloque
		*#
		# printArray
		var numArray:[int] = {1,2,3,4}
		printArray(numbers: numArray)

		# sumTimes
		print(sumTimes(first:10, last:20))

		# sumNumbers
		print(sumNumbers(first:10, last:20))


		# camera
		window_size(width: 1000, height: 1000)
		coordinates(x: 100, y: 100)

		var xpos:int = 0
		var ypos:int = 0

		var points:[point] = [5]

		for_each (var i in [1..5]) do
			points[i] has
				v1: (xpos, ypos),
				color: green,
				size: 10
			end

			if(xpos > 3) do
				xpos = 0
			end
```

```
        xpos++
        ypos++
end

camera points

var t0:triangle

t0 has
        v1: (0,0),
        v2: (2,0),
        v3: (4,2),
        color: green
end


camera t0

var t1:triangle has
        v1: (1,2),
        v2: (2,4),
        v3: (4,5),
        color: green
end

camera(t1)
var c1:circle has
        size: 5,
        color: red
end

camera c1

#action
action move t1 do
        begins: 2
        ends: 5
        pos_x: 5
        pos_y: 6
end
action move c1 do
        begins: 5
        ends: 9
        pos_x: 9
        pos_y:10
end
action scale do
        begins: 1
        ends: 2
        size: 5
end
action hide t1 do
        begins: 10
        ends: 15
end
action show t1 do
        begins: 16
```

```
                    ends: 17
            end

        }
        function sayText(text: string){
            print(text)
        }
        function addOne(number: int) returns int {
            return int + 1
        {
        function printArray(numbers: [int]){
            for_each (element in numbers) do
                print(element)
            end
        }
        function sumTimes(first: int, last: int) returns int{

            var sum:int = 0
            loop (first..last) do
                sum++
            end

            return sum
        }
        function sumNumbers(first: int, last: int) returns int{

            var sum:int = 0
            for (var index:int = first; index<last; index++) do
                sum = sum + index
            end

            return sum
        }

}
```

4. Plataforma de Desarrollo

- Utilizaremos la el analizador de PLY en el lenguaje PYTHON.
- Para procesar y desplegar los gráficos de “Light” © utilizaremos la librería de OpenGL para Python.
- Utilizaremos la herramienta de control de versiones Git

5. Firmas

Luis Alberto Lamadrid Tafich

Enrique Octavio Hernández Chávez

Ma. Elda Quiroga

6. Bibliografía

http://www.bottlecaps.de/
http://www.dabeaz.com/ply/ply.html
http://pyopengl.sourceforge.net/