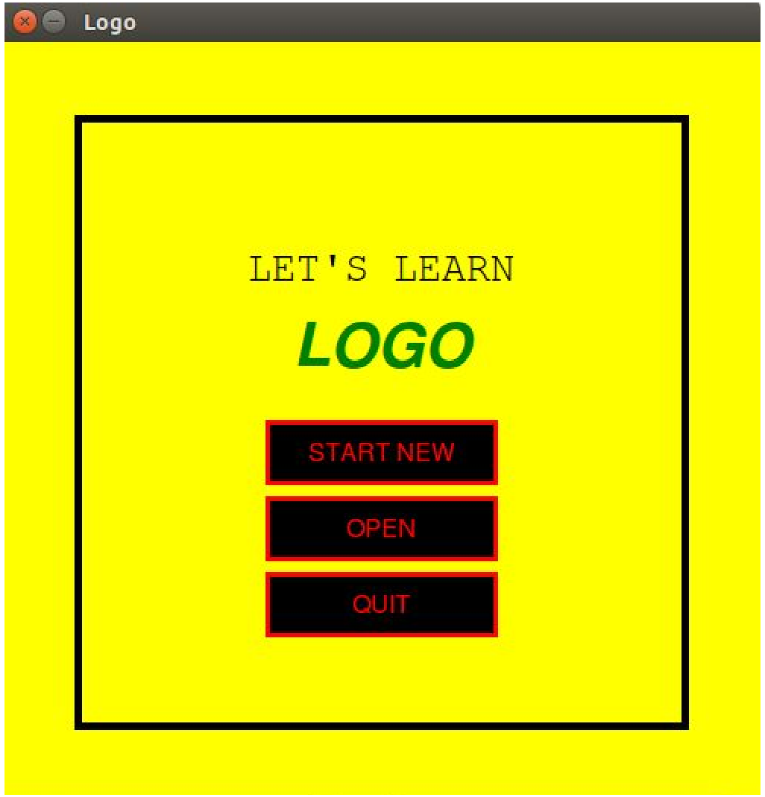
Let's Learn Logo

Animator and compiler

Aim

Make an animator and compiler for LOGO programming language.



First screen

What is LOGO ?

Logo is an educational **programming language**, designed in 1967 by Wally Feurzeig, Seymour Papert and Cynthia Solomon."Logo" is not an acronym. It was derived from the Greek logos meaning word or "thought" by Feurzeig, to distinguish itself from other programming languages that were primarily numbers, not **graphics or logic, oriented**.

Commands

1. **fd {number} or forward {number}** : to move {number} of steps forward
2. **bk {number} or backward {number}** : to move {number} of steps backward
3. **rt {number} or right {number}** : to turn the pointer by {number} degrees towards right
4. **lt {number} or left {number}** : to turn the pointer by {number} degrees towards left
5. **repeat {number} [{set of commands}]** : loop the {set of commands} {number} of times
6. **to {-function name} {set of commands} end** : create a function with function name {-function name} which can be called to execute the {set of commands}
7. **{command1}** + **{command2}** : to make a set of commands where {command2} is executed after {command1}
8. **clear** : clear and reset the window
9. **penup or pu** : to stop making line and just move pointer
10. **pendown or pd** : to start making the lines (default)

Grammar

statement ↣ expr

expr ↣ command | command + expr

command ↣ optionm **number** | **repeat number [** expr **]** | optionp

optionm ↣ **right** | **rt** | **left** | **lt** | **forward** | **fd** | **backward** | **bk**

optionp ↣ **penup** | **pu** | **pendown** | **pd** | **clr** | **clear** | func

func ↣ **to func_name** expr **end** | **func_name**

Grammar

statement ↣ expr

Added extra

expr ↣ command | command + expr

command ↣ optionm **number** | **repeat number [** expr **]** | optionp

optionm ↣ **right** | **rt** | **left** | **lt** | **forward** | **fd** | **backward** | **bk**

optionp ↣ **penup** | **pu** | **pendown** | **pd** | **clr** | **clear** | func

func ↣ **to func_name** expr **end** | **func_name**

Must start with '_'

TOKENS

RIGHT ↣ ‘right’ or ‘rt’

LEFT ↣ ‘left’ or ‘lt’

FORW ↣ ‘forward’ or ‘fd’

BACK ↣ ‘backward’ or ‘bk’

LPAREN ↣ ‘[’

RPAREN ↣ ‘]’

PLUS ↣ ‘+’

REPEAT ↣ ‘repeat’

PENU ↣ ‘penup’ or ‘pu’

PEND ↣ ‘pendown’ or ‘pd’

CLR↣ ‘clear’ or ‘clr’

NAME↣ ‘-{followed by any string of alphabets}’

FUNS ↣ ‘to’

FUNE ↣ ‘end’

Grammar with tokens

statement ↣ expr

expr ↣ command | command PLUS expr

command ↣ optionm **NUMBER** | **REPEAT NUMBER LPAREN** expr **RPAREN** | optionp

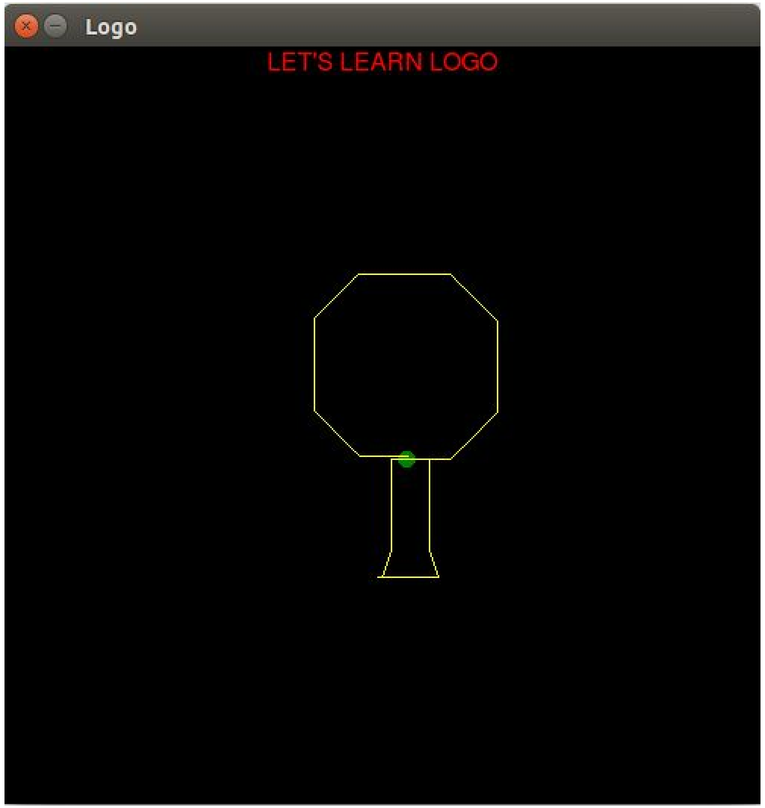
optionm ↣ **RIGHT** | **LEFT** | **FORW** | **BACK**

optionp ↣ **PENU** | **PEND** | **CLR** | func

func ↣ **FUNS NAME** expr **FUNE** | **NAME**

Example explained

```
to -ls rt 20 + fd 20 + lt 20 + fd 60 end

to -ts rt 90 + fd 25 + rt 90 end

to -rs fd 60 + lt 20 + fd 20 + rt 20 end

to -res rt 90 + fd 40 + rt 90 end

to -tr -ls + -ts + -rs + -res end

to -cp pu + fd 80 + rt 90 + fd 20 + lt 90 + pd end

to -cc repeat 360 [fd 2 + rt 1] end

to -te pu + bk 100 + pd + -tr + -cp + left 90 + -cc end

-te
```



Output

Thank you

Members :

Chaitanya Nagpal
IS201401011
Ayushi Anand
IS201401006