GeneticL – A L-System Generator using Genetic Algorithms

1.0 About

this project is to create a program to generate L-systems using genetic algorithms. The algorithm starts with random set of systems and evolute them using the user's vote as the fitness function.

2.0 Implementation

In the program a L-System has one common rule (F=FF), one unique rule. And they also has same axiom (XF) and same turning angle (25^0). The unique rule is defined by using a genetic algorithm. First a random set of rules are generated (with balanced brackets). Then they are evaluated and drawn on the screen. Then the user can give a score according to each L-System. Then the genetic algorithm will create new generation of L-Systems by crossing over and mutating them. This process will be repeated again and again by the program.

The task of the cross over function is to swap parts of two L-System rules and create two new rules. And the mutation function swap two characters in a single rule. After both of this tasks the brackets must be balances to make it a valid rule.

```
def cross_over(self, i1, i2, i):
    if random.uniform(0, 100) < self.crossover_rate:
        l1 = random.randrange(0, len(self.chromosoms[i1]))
        l2 = random.randrange(0, len(self.chromosoms[i2]))
        s11 = self.chromosoms[i1][:l1]
        s12 = self.chromosoms[i1][l1:]
        s21 = self.chromosoms[i2][:l2]
        s22 = self.chromosoms[i2][l2:]
        self.chromosoms_new[i] = s11 + s22
        self.chromosoms_new[i + 1] = s21 + s12
    else:
        self.chromosoms_new[i] = self.chromosoms[i1]
        self.chromosoms_new[i + 1] = self.chromosoms[i2]
```

```
def mutate(self, i1):
    if random.uniform(0, 100) < self.mutation_rate:
        l1 = random.randrange(0, len(self.chromosoms_new[i1]))
        l2 = l1
        while l1 != l2:
            l2 = random.randrange(0, len(self.chromosoms_new[i1]))
            ls = list(self.chromosoms_new[i1])
            c = ls[l1]
            ls[l1] = ls[l2]
            ls[l2] = c
```

3.0 Pre-Requirements

This script is written in python. Install 'python' and 'Tkinter' to run the script. (use synaptic package manager in Ubuntu to install them)

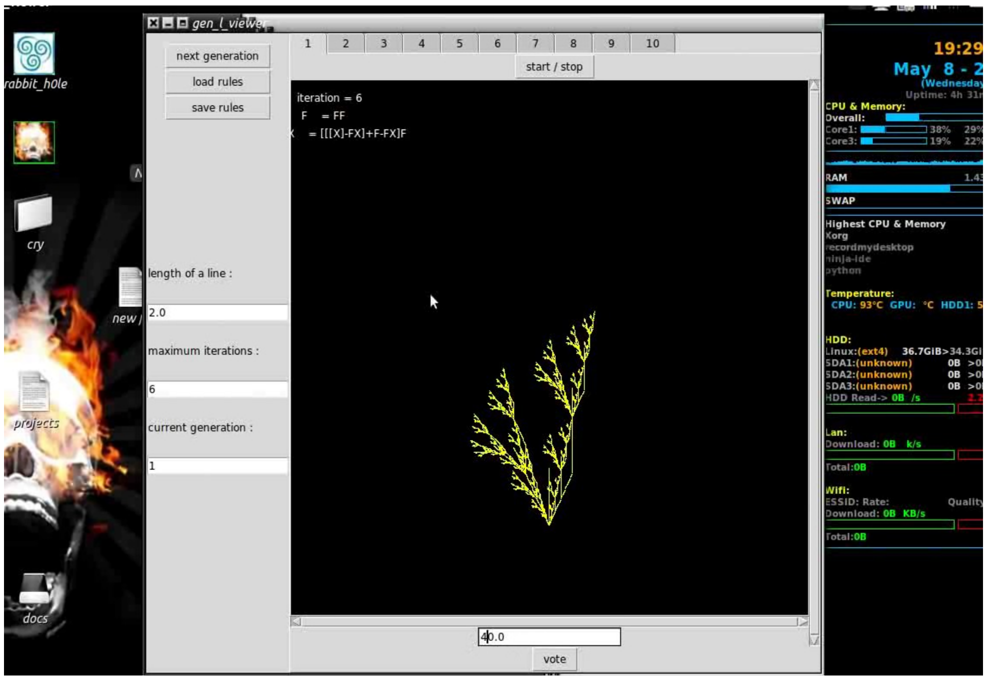
4.0 Running the Application

Use fallowing command to run the application.

```
python GeneticL.py
```

when the application stearts runnings the initial random rule set will be created and the current generation will be marked as '0'. Then got to each tab and start each L-system and score the using vote button. The score must be a value 0 to 100. Then use next generation button to make next generation. And you can repeat the process. You can also save current set of rules by using save button. It will be saved in a text file called 'rules'. Then when you are running the application again use load button to load the rules in the 'rules' file in to the application. All the rules generated and there scores will be loges in the file call 'loges'.

5.0 Screen-shots



```
iteration = 6
 F   = FF
X   = [[[X]-FX]+F-FX]F
```

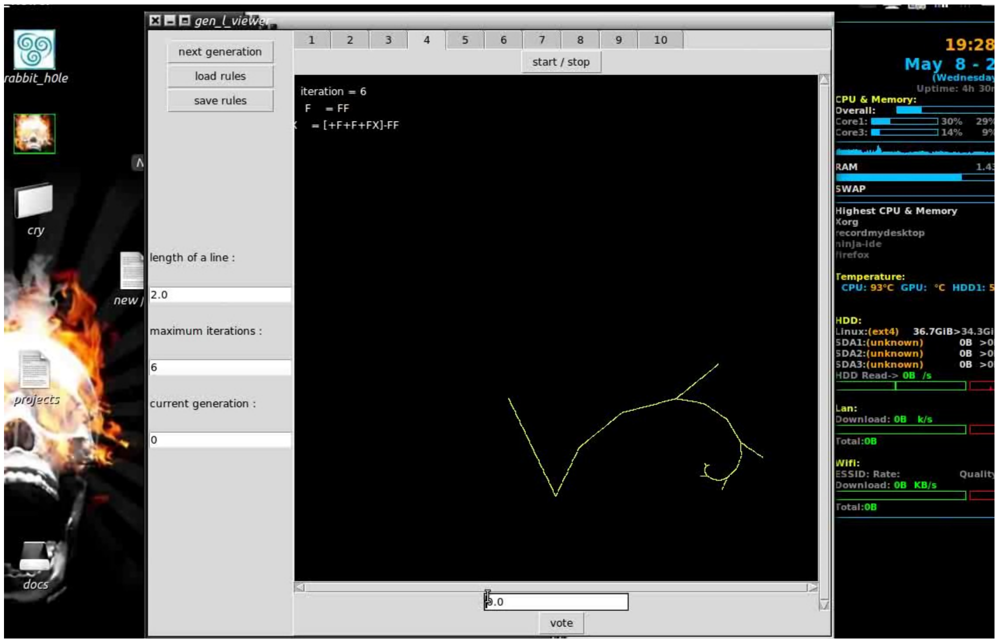
length of a line : 2.0

maximum iterations : 6

current generation : 1

40.0

figure 5.1



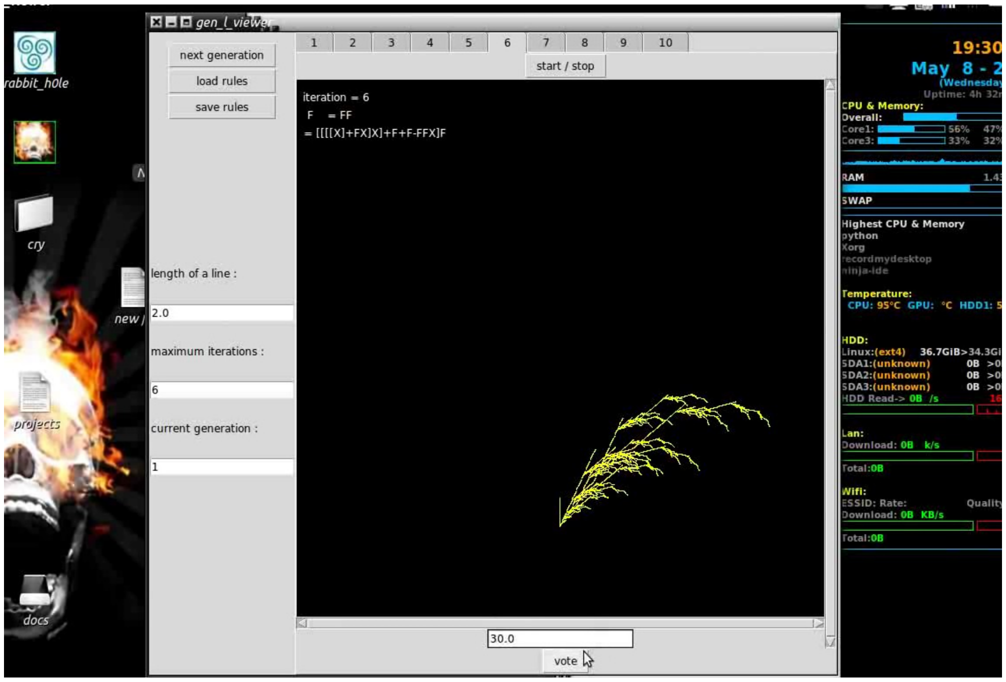
```
iteration = 6
 F   = FF
X   = [+F+F+FX]-FF
```

length of a line : 2.0

maximum iterations : 6

current generation : 0

figure 5.2



```
iteration = 6
 F   = FF
= [[[[X]+FX]X]+F+F-FFX]F
```

length of a line : 2.0

maximum iterations : 6

current generation : 1

30.0

figure 5.3

5.0 More inf0

- tcg.galahena@gmail.com
- http://www.inf0warri0r.blogspot.com

6.0 Licenses