L-Systems

FOCS - Fall 23

Sparsh Gupta and Daniel Sudzilouski
https://github.com/daniel-sudz/focs-lsystems

What are L-systems?

- Basically grammar*
  - One rule for every symbol to convert
  - Every symbol is converted if possible per iteration
  - Can be context-sensitive or not

- Usually run for some number of steps

- Symbols can be used as drawing instruction to make cool visuals

Traditional Grammar

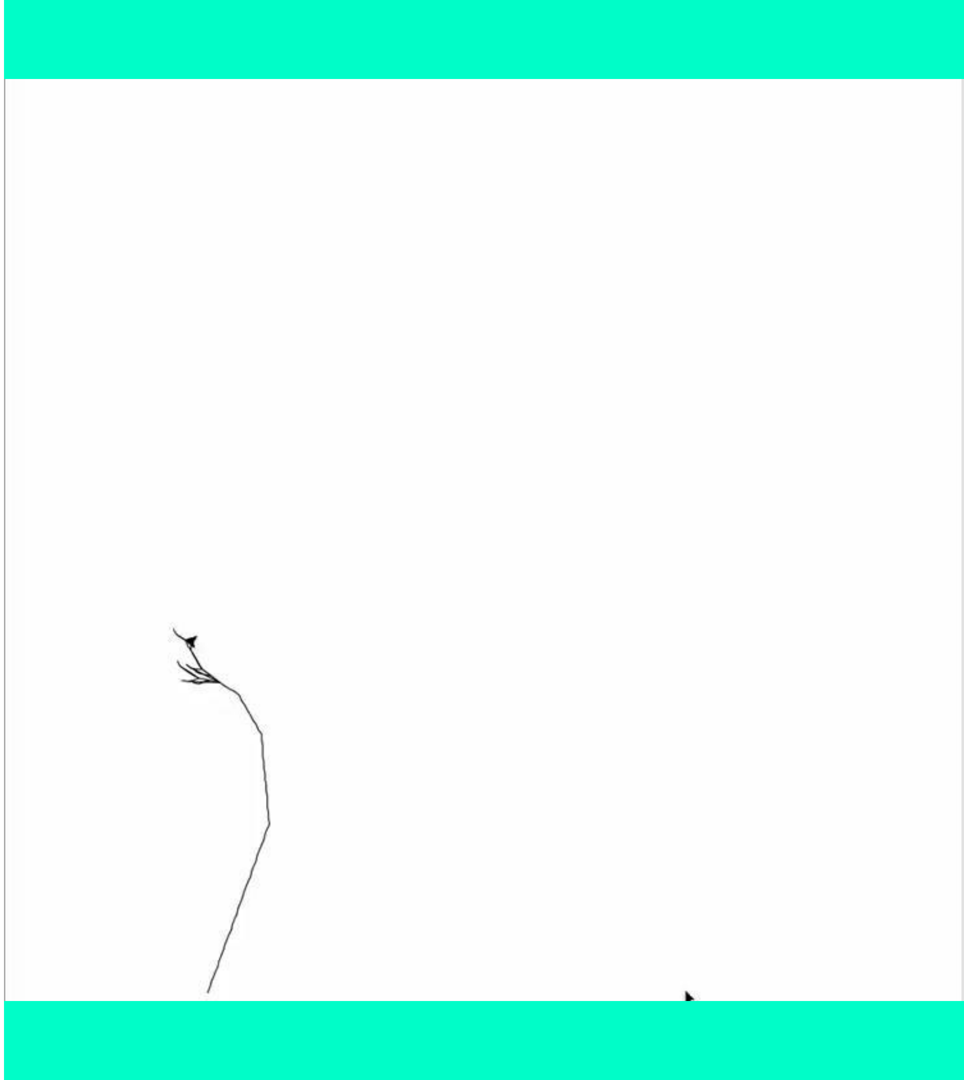
Start	S
Terminals	{S}
Non-Terminals	{}
Rules	{S -> SS}
Language	{S, SS, SSS…}

L-system Grammar

Start	S
Terminals	{S}
Non-Terminals	{}
Rules	{S -> SS}
Language	{S, SS, SSSS, …}

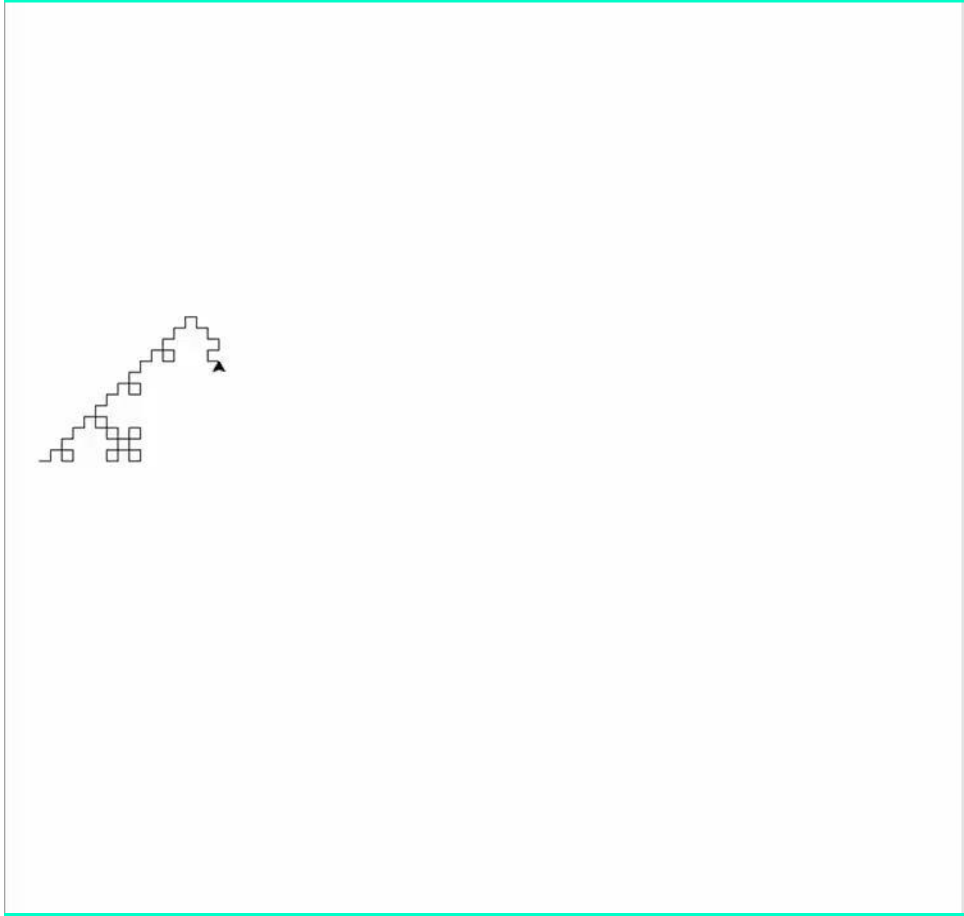
Fractal Plant

```
# list to store stack
stack = []

# Functions to push/pop stack for fractal plant
def push_stack(t):
    stack.append((t.pos(), t.heading()))

def pop_stack(t):
    pos, heading = stack.pop()
    t.setpos(pos)
    t.setheading(heading)

# Fractal Plant LSystem
fractal_plant = LSystem(
    start="X",
    rules=
    {
        "X": ProductionRule("F+[[X]-X]-F[-FX]+X", None, None),
        "F": ProductionRule("FF", None, None)
    },
    iterations=6,
    visualizations=
    {
        "F": lambda t: t.forward(5),
        "+": lambda t: t.left(25),
        "-": lambda t: t.right(25),
        "[": push_stack,
        "]": pop_stack
    },
    render_start_pos=(-250, -400),
    render_heading=70,
    debug=True
)

# Visualize the Fractal Plant
fractal_plant.visualize()
```



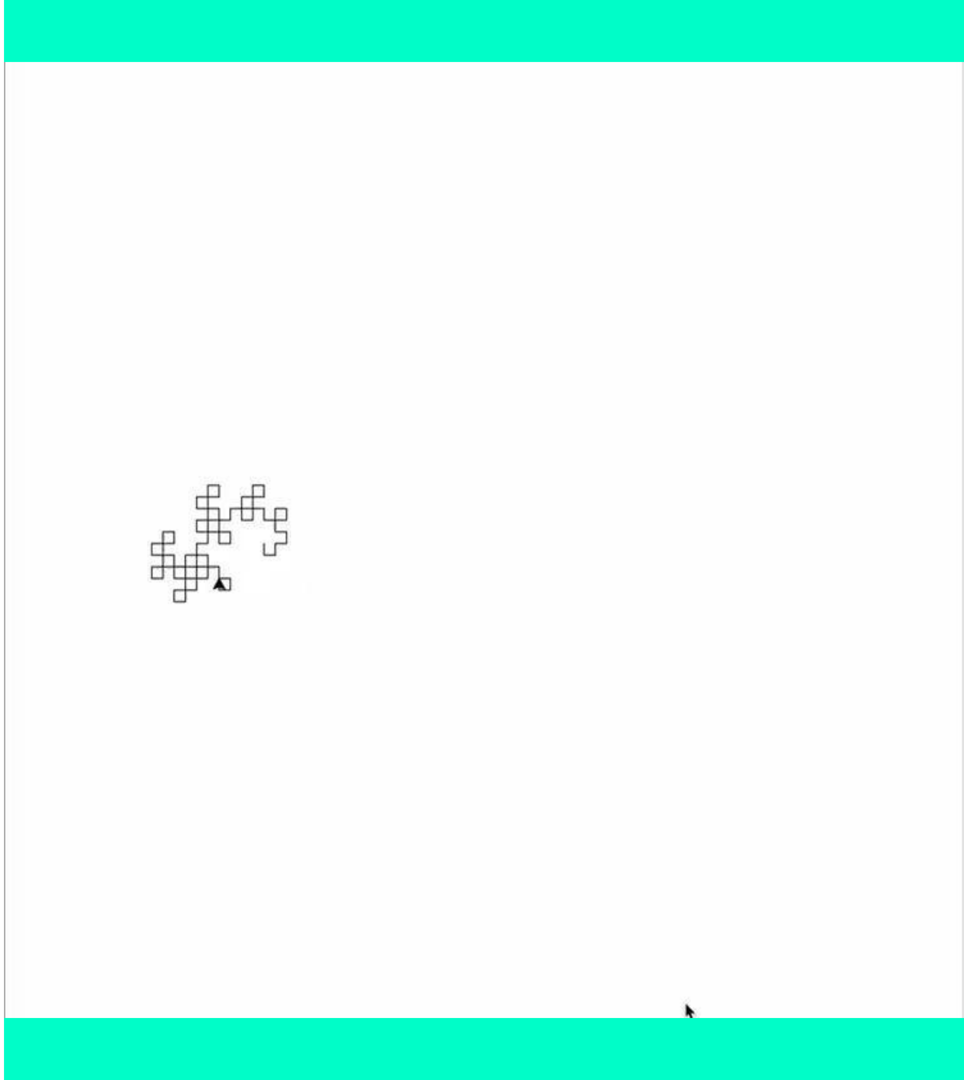
Koch Curve

```python
# Koch curve LSystem
koch_curve = LSystem(
    start="F",
    rules=
    {
        "F": ProductionRule("F+F-F-F+F", None, None),
    },
    iterations=4,
    visualizations=
    {
        "F": lambda t: t.forward(10),
        "+": lambda t: t.left(90),
        "-": lambda t: t.right(90)
    },
    render_start_pos=(-400, 0),
    render_heading=0,
    debug=True
)

koch_curve.visualize()
```



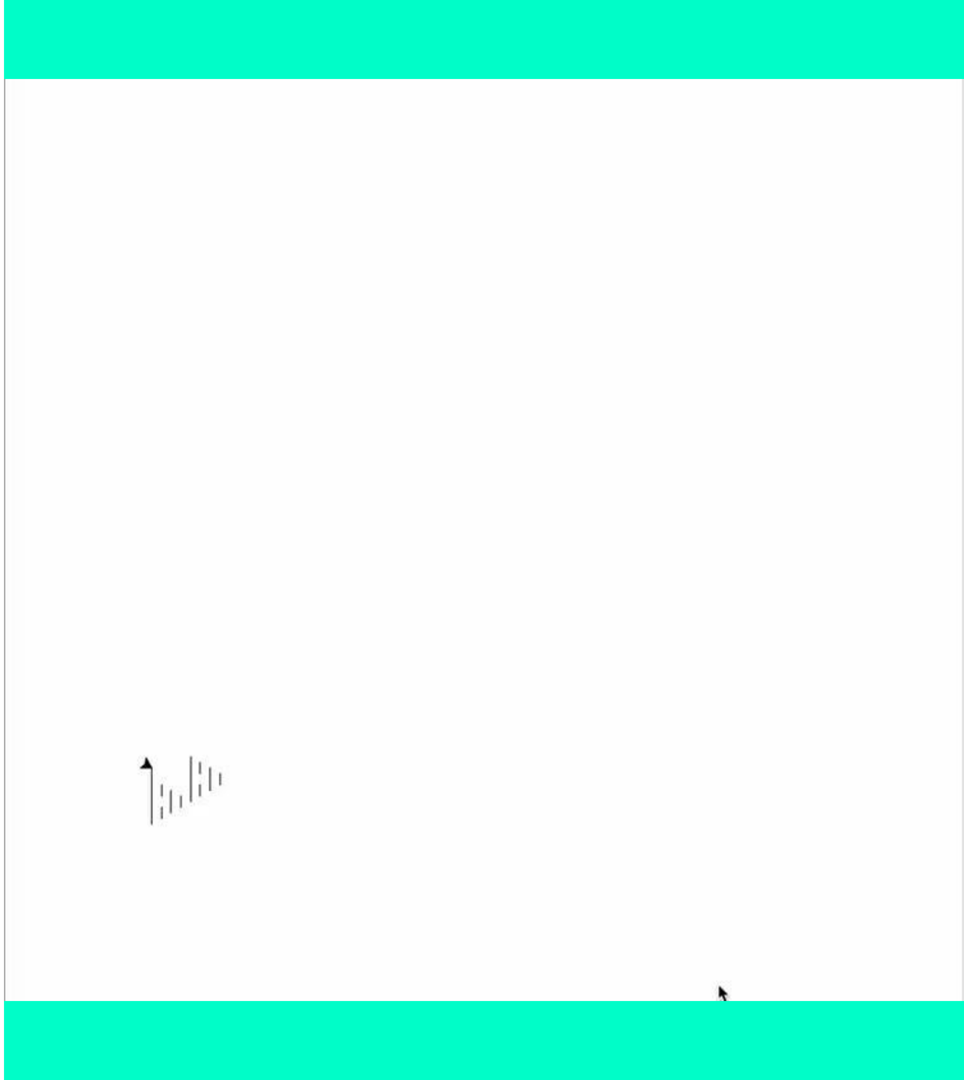
Dragon Curve

```python
# Dragon curve LSystem
dragon_curve = LSystem(
    start="F",
    rules=
    {
        "F": ProductionRule("F+G", None, None),
        "G": ProductionRule("F-G", None, None)
    },
    iterations=10,
    visualizations=
    {
        "F": lambda t: t.forward(10),
        "G": lambda t: t.forward(10),
        "+": lambda t: t.left(90),
        "-": lambda t: t.right(90)
    },
    render_start_pos=(-200, 0),
    render_heading=270,
    debug=True
)

dragon_curve.visualize()
```



Sierpinski triangle

```python
# Distance to move/write forward
distance = 10

# Function to move forward for Sierpinski Triangle
def move_forward(t):
    t.penup()
    t.forward(distance)
    t.pendown()

# Sierpinski Triangle LSystem
sierpinski_triangle = LSystem(
    start="F-F-F",
    rules=
    {
        "F": ProductionRule("F-G+F+G-F", None, None),
        "G": ProductionRule("GG", None, None)
    },
    iterations=5,
    visualizations=
    {
        "F": lambda t: t.forward(distance),
        "G": move_forward,
        "+": lambda t: t.left(120),
        "-": lambda t: t.right(120)
    },
    render_start_pos=(-300, -250),
    render_heading=90,
    debug=True
)

sierpinski_triangle.visualize()
```



Stochastic L-Systems

- Introducing randomness to L-system rules
- Each rule has associated probabilities
- More unpredictable, natural-looking visuals
- Unique visuals in every run

Stochastic L-system Grammar

Start	S
Terminals	{S}
Non-Terminals	{}
Rules	{S - P(0.5) -> SS}, {S - P(0.5) -> SSSS}
Language	{S, SS, SSSS, …}

Stochastic Fractal Plant

```python
# List to store stack
stack = []


# Functions to push/pop stack for fractal plant
def push_stack(t):
    stack.append((t.pos(), t.heading()))


def pop_stack(t):
    pos, heading = stack.pop()
    t.setpos(pos)
    t.setheading(heading)


# Fractal Plant Stochastic LSystem with stochastic rules
fractal_plant = StochasticLSystem(
    start="F",
    rules=
    {
        "F":
        [
            ("F[+F]F[-F]F", 0.33),  # Probability 1/3
            ("F[+F]F", 0.33),       # Probability 1/3
            ("F[-F]F", 0.34)        # Probability 1/3
        ]
    },
    iterations=5,
    visualizations={
        "F": lambda t: t.forward(5),
        "+": lambda t: t.left(25),
        "-": lambda t: t.right(25),
        "[": push_stack,
        "]": pop_stack
    },
    render_start_pos=(0, -400),
    render_heading=90,
    debug=True
)

# Visualize the Stochastic Fractal Plant
fractal_plant.visualize()
```