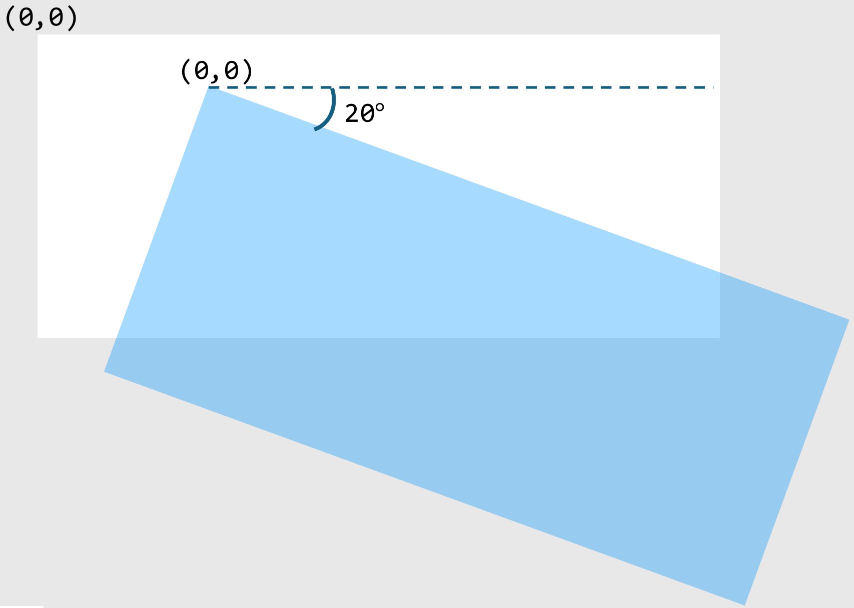
L-Systems

CMSC 326 Simulations

Today’s Lecture

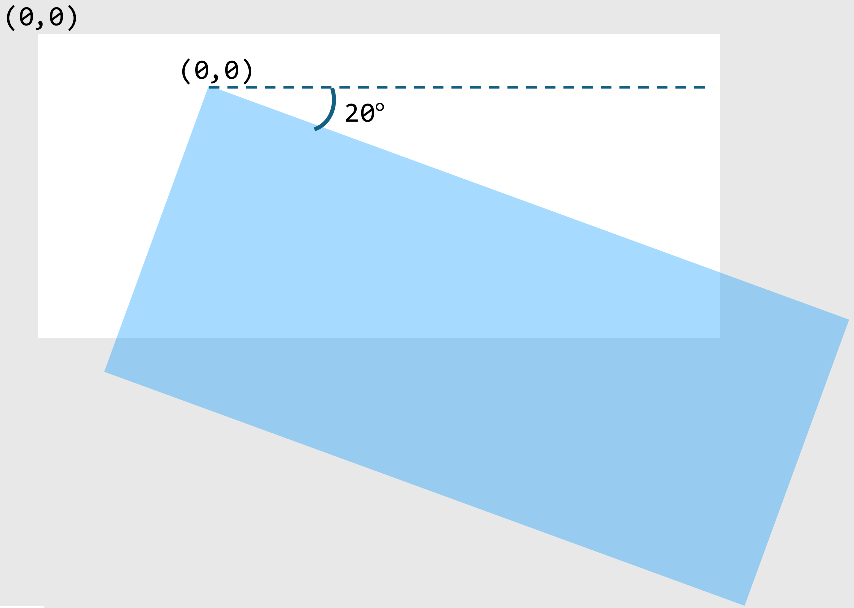
Translation and Rotation



L-systems

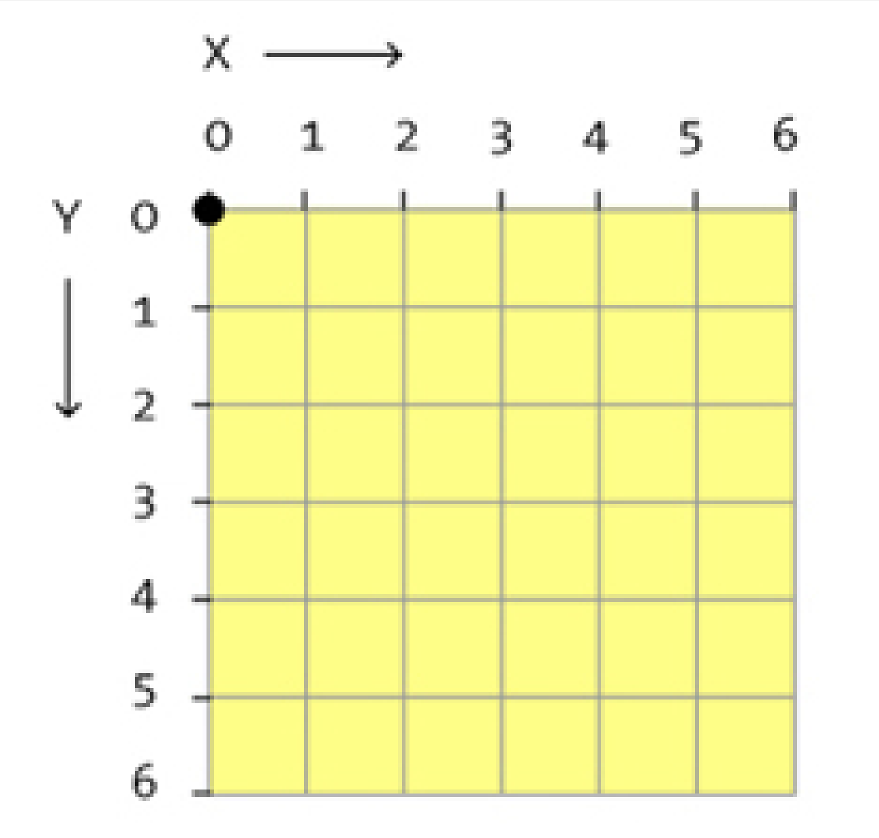
Today's Lecture

Translation and Rotation



L-systems

Coordinate System in Py5

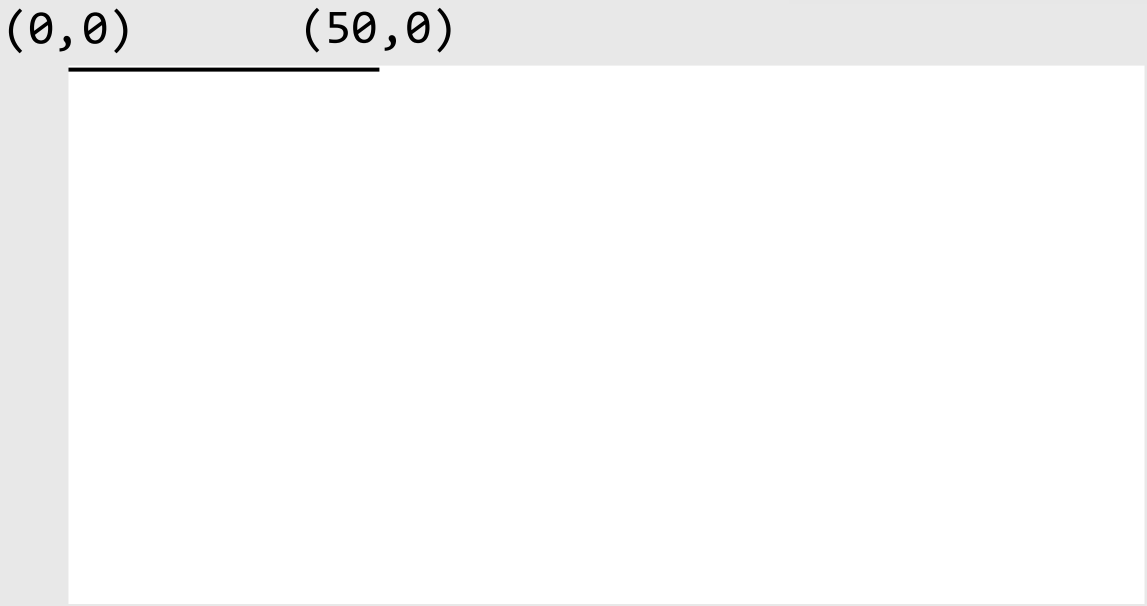


Canvas in Py5

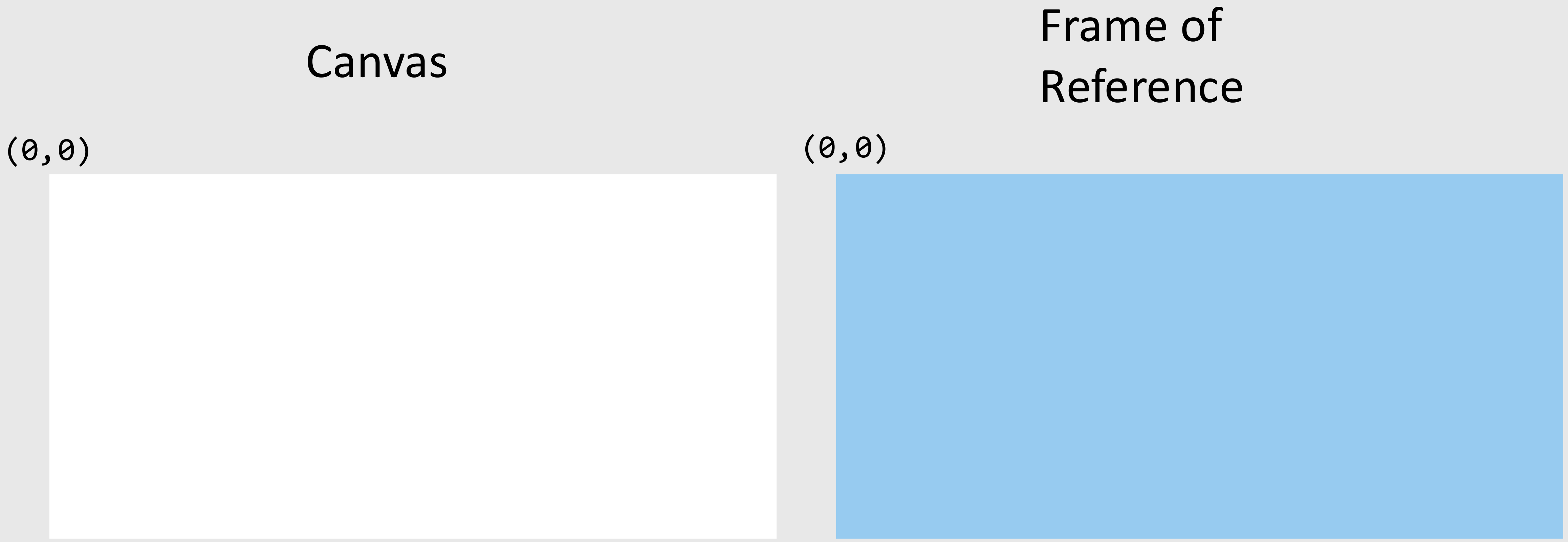
Canvas in Py5

Draw a line from **(0, 0)** to **(50, 0)**

```
py5.line(0, 0, 50, 0)
```

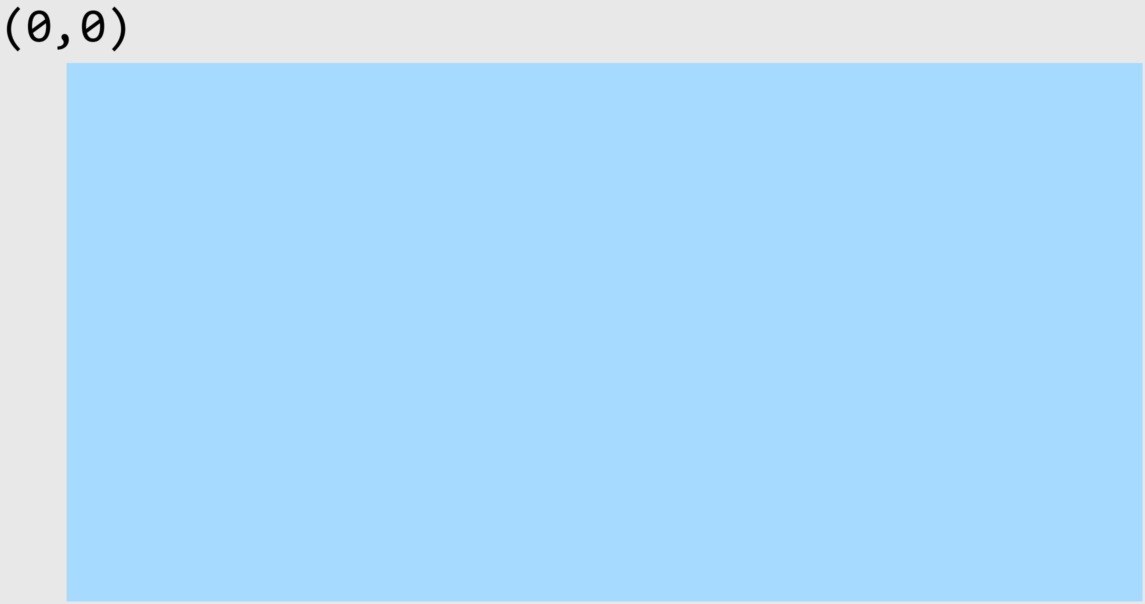


Frame of Reference in Py5



Frame of Reference in Py5

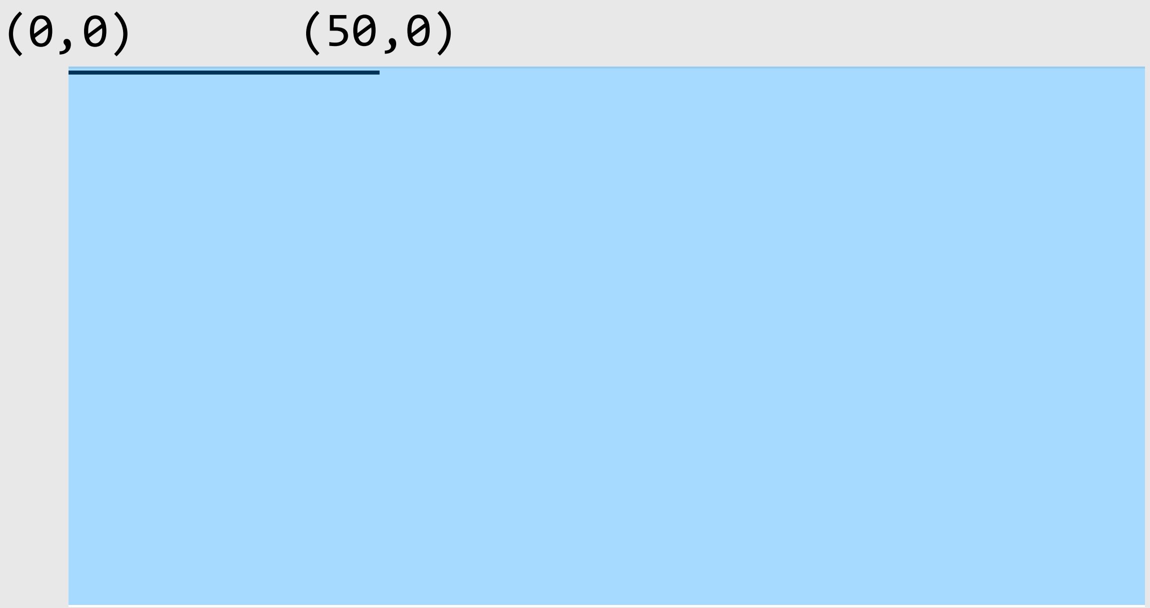
Frame of Reference
is directly on the canvas by
default



Frame of Reference in Py5

Draw a line from **(0, 0)** to **(50, 0)**

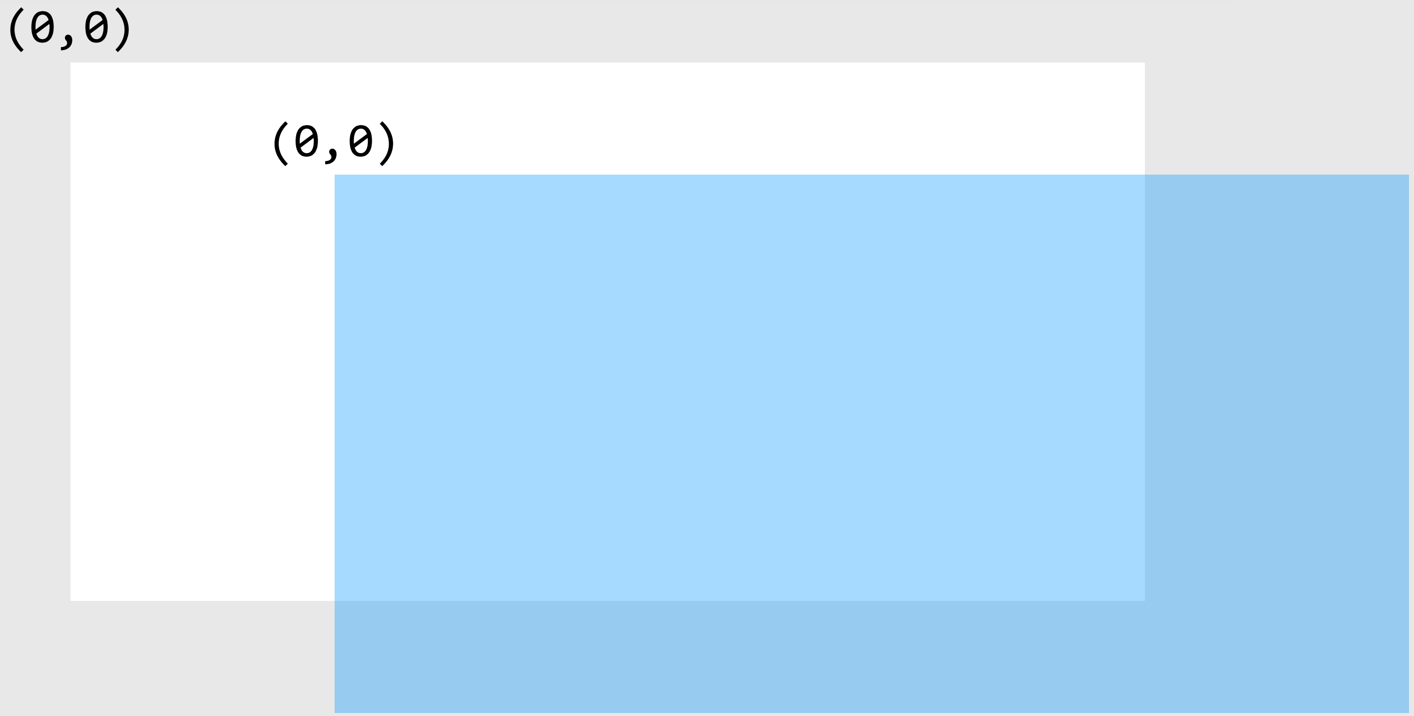
```
py5.line(0, 0, 50, 0)
```



Translation in Py5

Translate moves the **frame of reference** origin to the location specified

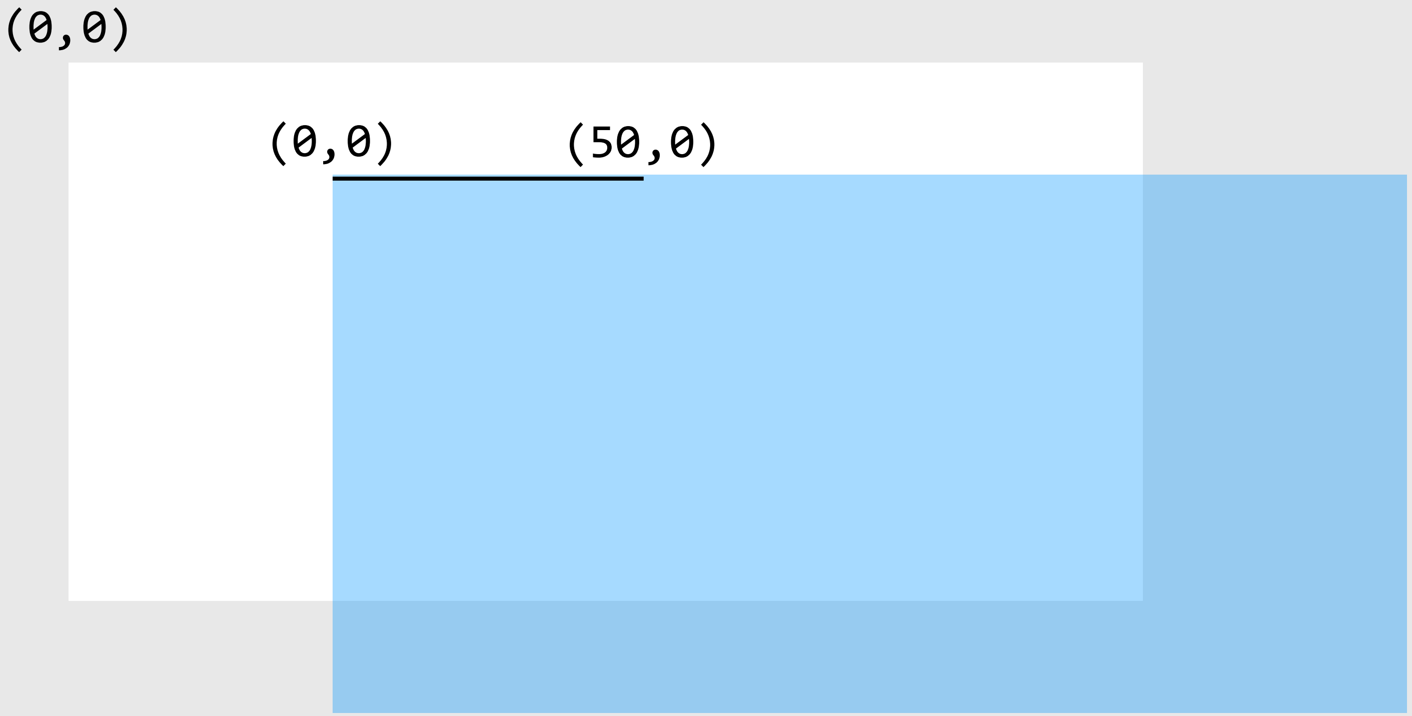
```
py5.translate(200, 100)
```



Translation in Py5

Draw a line **after translation**:
you are using the current frame of reference

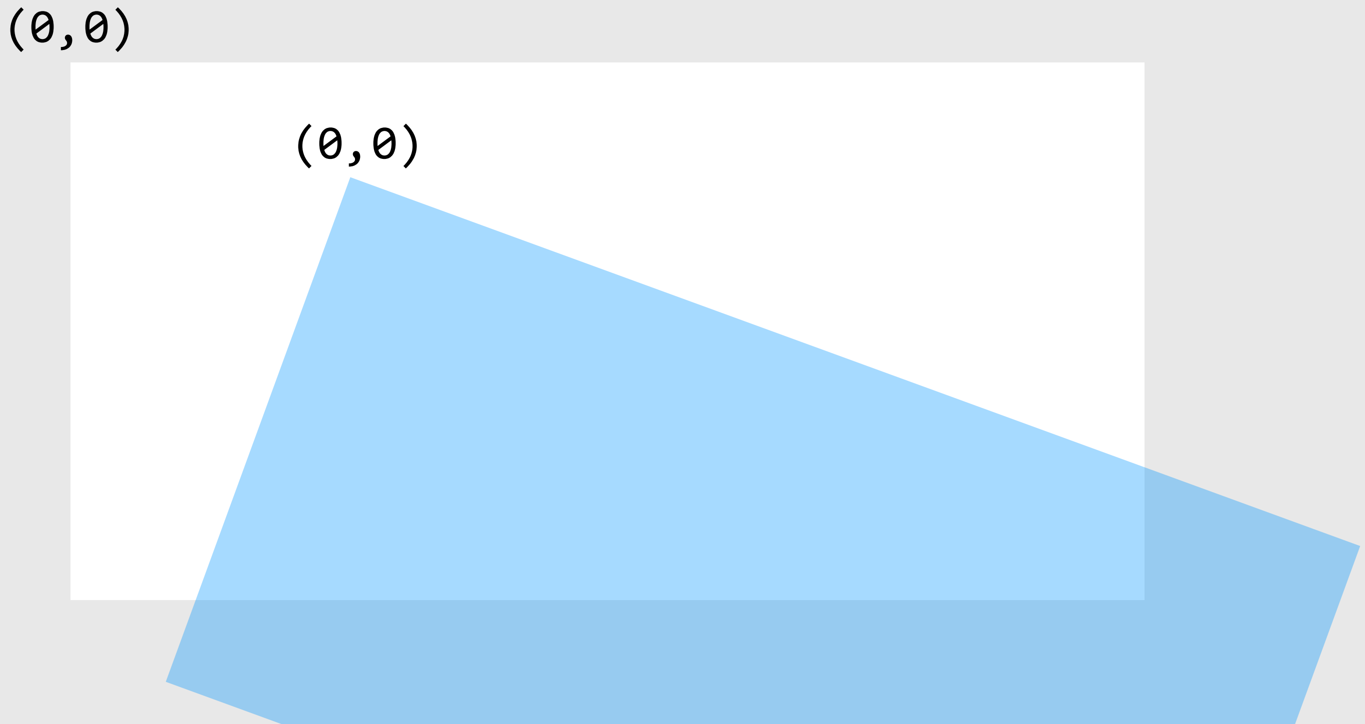
```
py5.translate(200, 100)
py5.line(0, 0, 50, 0)
```



Rotation in Py5

Rotation rotates the **frame of reference** from the origin by the specified angle (radians)

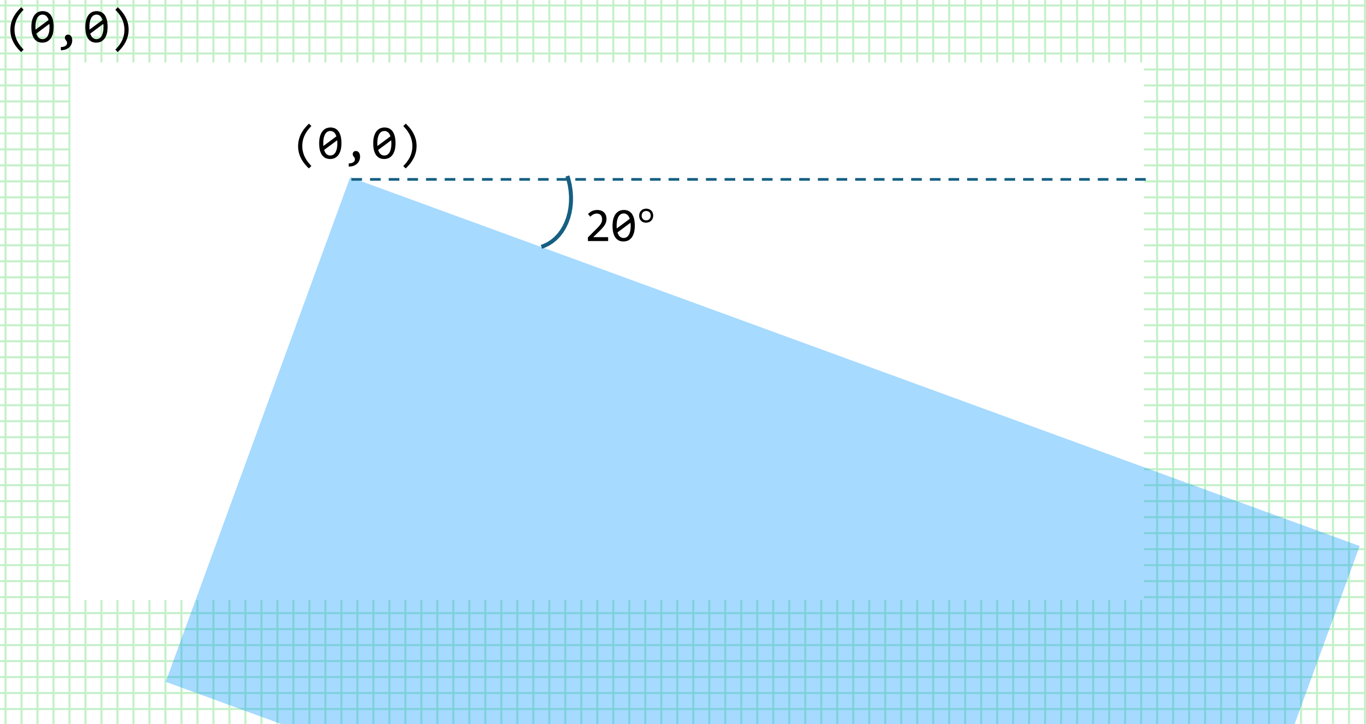
```
py5.translate(200, 100)
py5.rotate(py5.radians(20))
```



Rotation in Py5

Rotation rotates the **frame of reference** from the origin by the specified angle (radians)

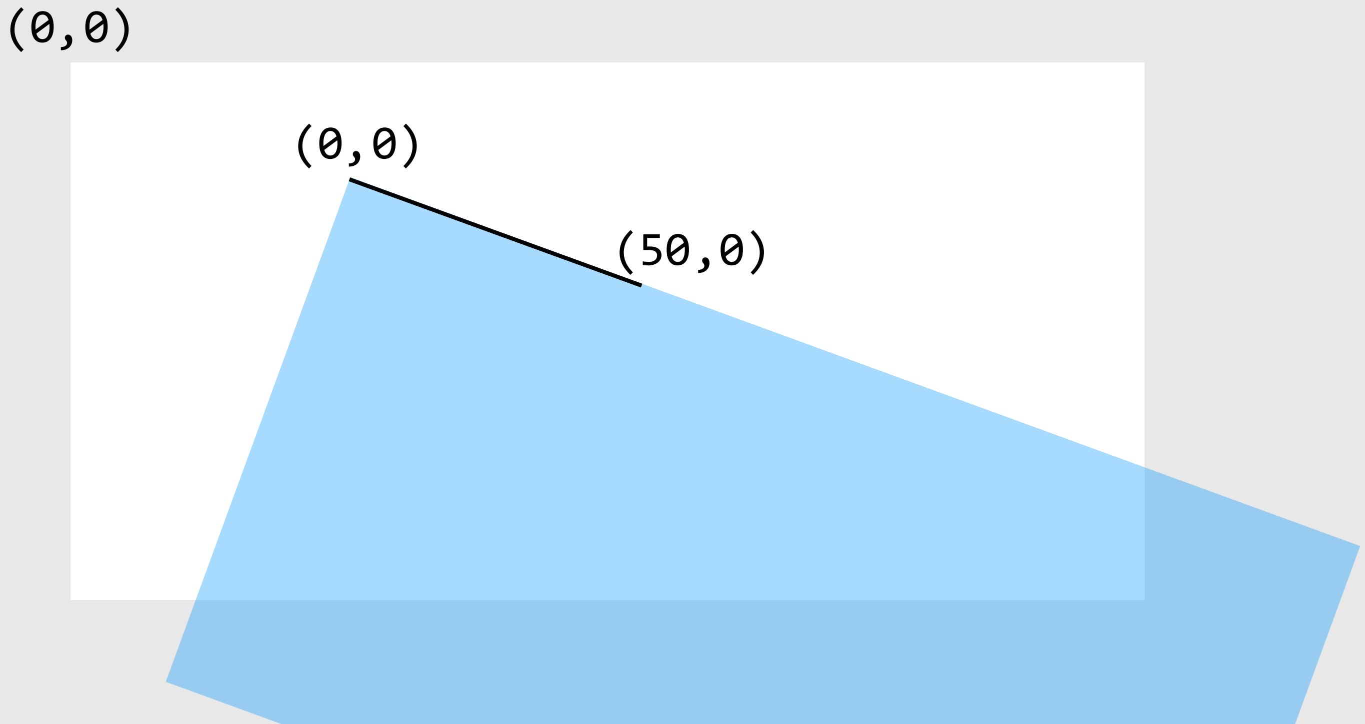
```
py5.translate(200, 100)
py5.rotate(py5.radians(20))
```



Rotation in Py5

Draw a line **after rotation**:
you are using the current frame of reference

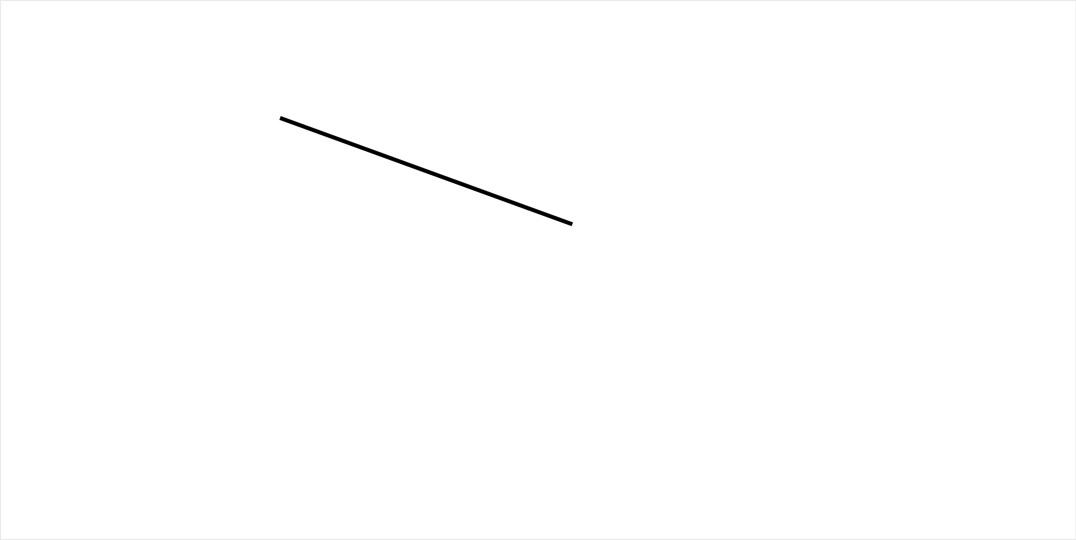
```
py5.translate(200, 100)
py5.rotate(py5.radians(20))
py5.line(0, 0, 50, 0)
```



Rotation in Py5

Draw a line **after rotation**:
you are using the current frame of reference

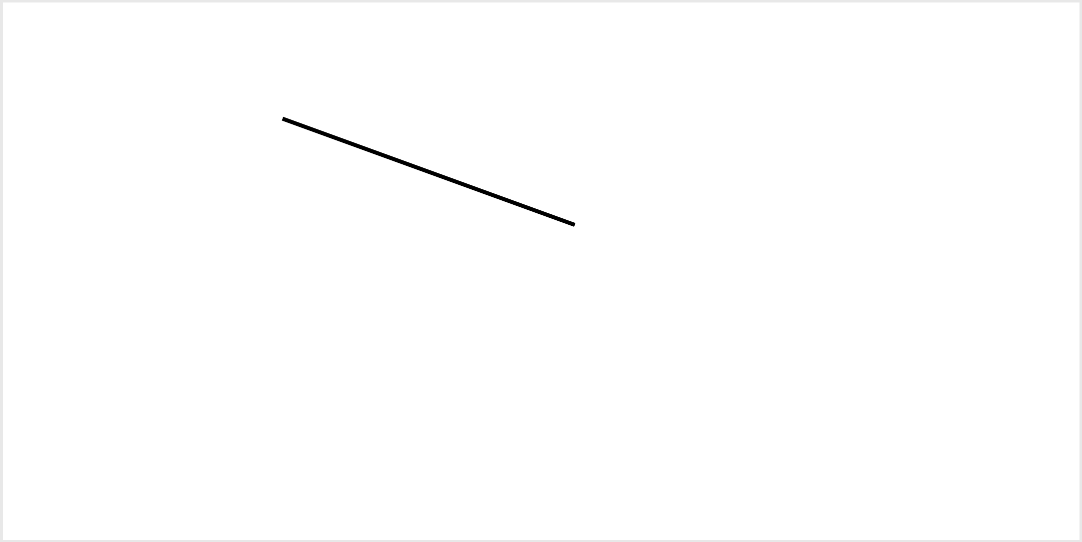
```
py5.translate(200, 100)
py5.rotate(py5.radians(20))
py5.line(0, 0, 50, 0)
```



Translation and Rotation in Py5

```
py5.line(0, 0, 50, 0)
```

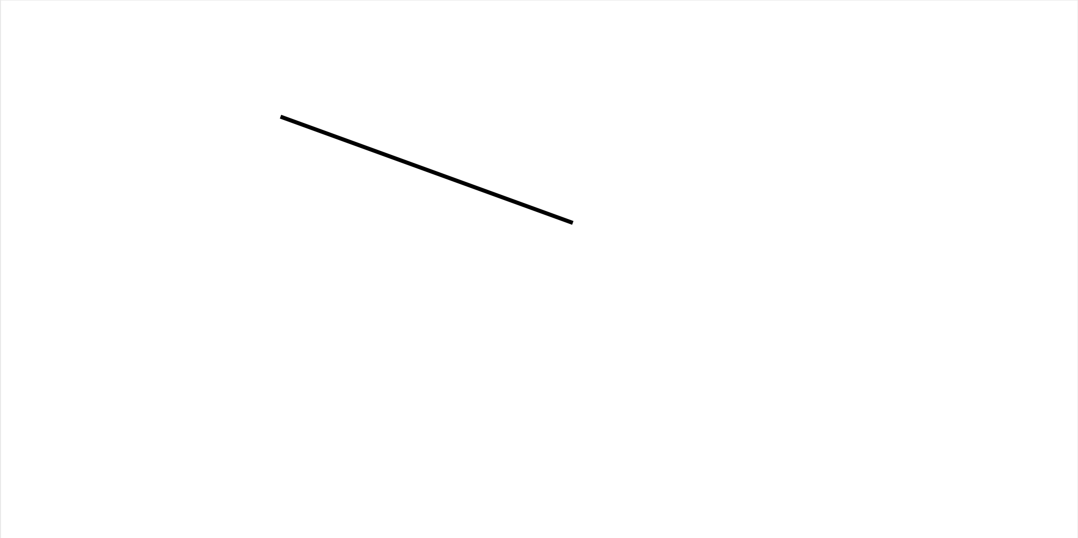
```
py5.translate(200, 100)
py5.rotate(py5.radians(20))
py5.line(0, 0, 50, 0)
```

Translation and Rotation in Py5

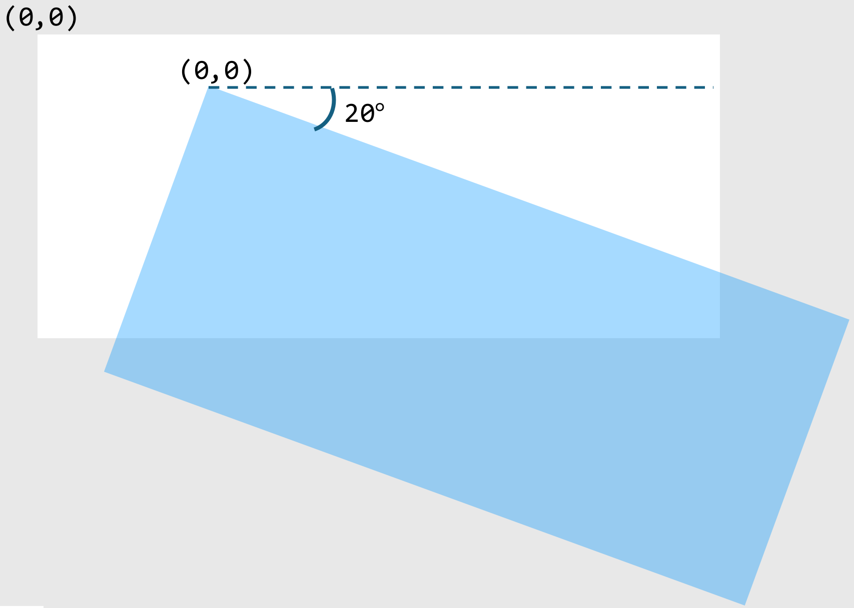
```
py5.line(0, 0, 50, 0)
```

```
py5.translate(200, 100)
py5.rotate(py5.radians(20))
py5.line(0, 0, 50, 0)
```

Today's Lecture

Translation and Rotation



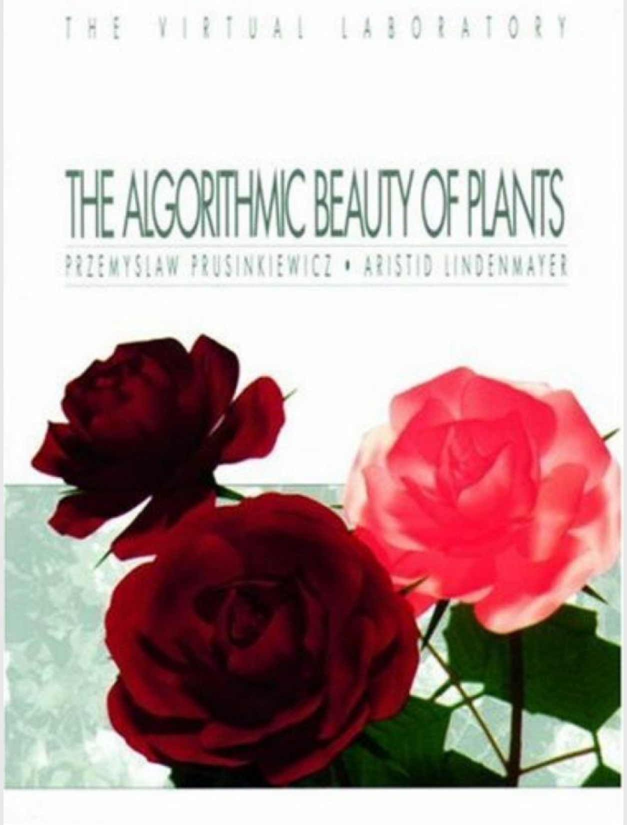
L-systems

Lindenmayer Systems

L-system: Technique to generate recursive fractal patterns

Conceived as a mathematical theory of plant development

The central concept of L-systems is that of **rewriting**



https://archive.org/details/the-algorithmic-beauty-of-plants

L-systems

An L-system has three main components:

1. **Alphabet:** the valid characters that can be included

Example: $A \; B \; C$

L-systems

An L-system has three main components:

1. **Alphabet:** the valid characters that can be included
2. **Axiom:** a sentence that describes the initial state of the system

Example: AAA, or B, or $ACBAB$

L-systems

An L-system has three main components:

1. **Alphabet:** the valid characters that can be included
2. **Axiom:** a sentence that describes the initial state of the system
3. **Rules:** ways of transforming the sentence that are applied recursively, starting with the axiom, generating new sentences repeatedly

Example: $A \rightarrow AB$

Example: A Simple L-system

Alphabet	A, B
Axiom	A
Rules	$A \rightarrow AB$ $B \rightarrow A$

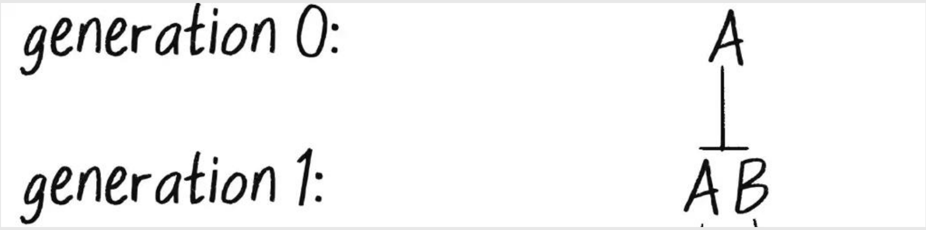
Example: A Simple L-system

Alphabet	A, B
Axiom	A
Rules	$A \rightarrow AB$ $B \rightarrow A$

generation 0: A

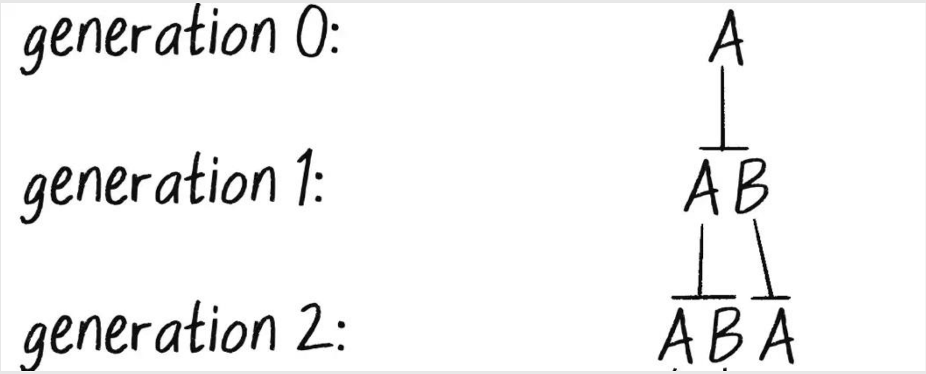
Example: A Simple L-system

Alphabet	A, B
Axiom	A
Rules	$A \rightarrow AB$ $B \rightarrow A$



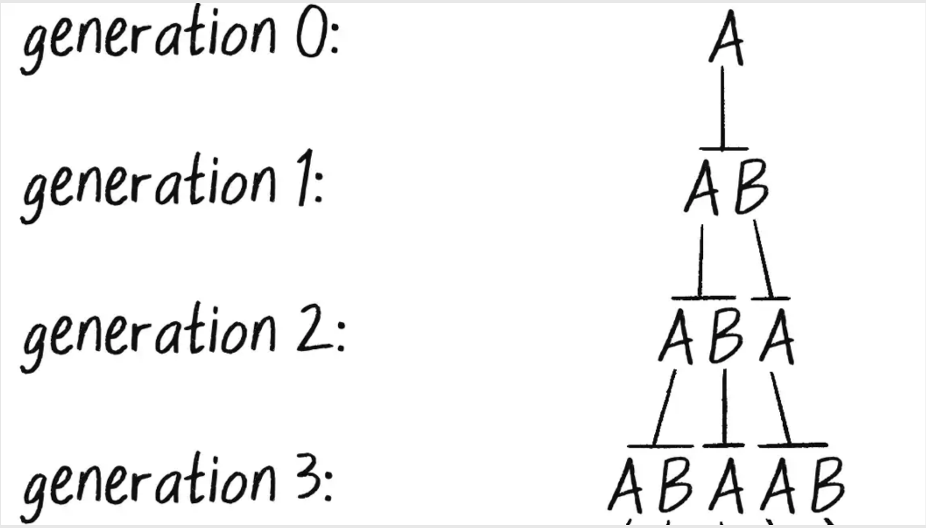
Example: A Simple L-system

Alphabet	A, B
Axiom	A
Rules	$A \rightarrow AB$ $B \rightarrow A$



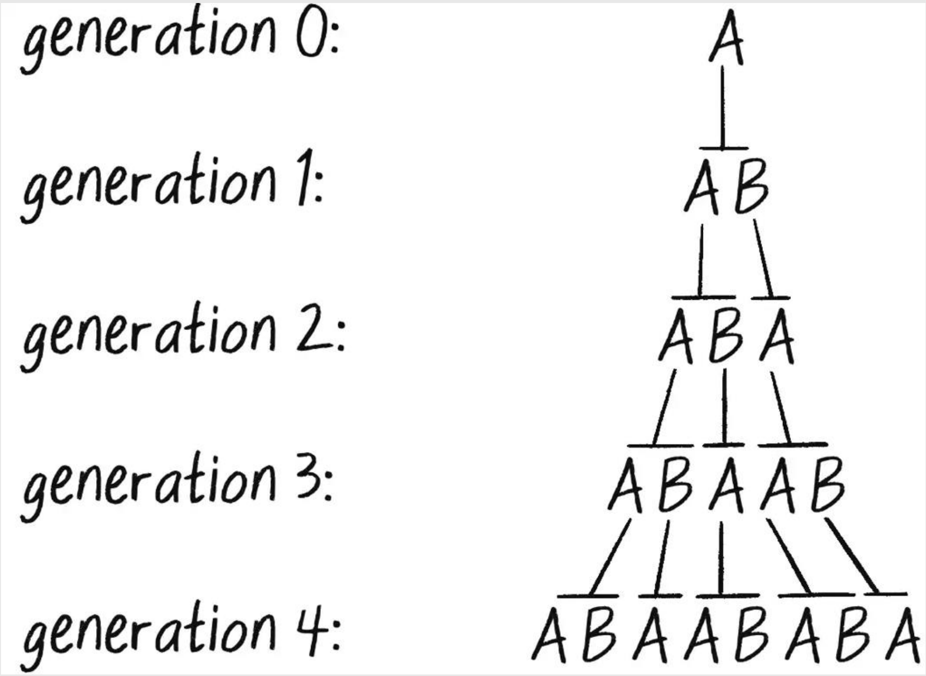
Example: A Simple L-system

Alphabet	A, B
Axiom	A
Rules	$A \rightarrow AB$ $B \rightarrow A$



Example: A Simple L-system

Alphabet	A, B
Axiom	A
Rules	$A \rightarrow AB$ $B \rightarrow A$



Example: A Simple L-system

Alphabet	A, B
Axiom	A
Rules	$A \to AB$ $B \to A$

```
# Start with the axiom
current_sentence = "A"

next_sentence = ""

# For every character of the current sentence
for character in current_sentence:
    # Apply the production rules A->AB, B->A
    if character == "A":
        next_sentence = next_sentence + "AB"
    elif character == "B":
        next_sentence = next_sentence + "A"

current_sentence = next_sentence
```

Example: A Simple L-system

Alphabet	A, B
Axiom	A
Rules	$A \rightarrow AB$ $B \rightarrow A$

```
# Start with the axiom
current_sentence = "A"

next_sentence = ""

# For every character of the current sentence
for character in current_sentence:
    # Apply the production rules A->AB, B->A
    if character == "A":
        next_sentence = next_sentence + "AB"
    elif character == "B":
        next_sentence = next_sentence + "A"

current_sentence = next_sentence
```

Example: A Simple L-system

Alphabet	A, B
Axiom	A
Rules	$A \rightarrow AB$ $B \rightarrow A$

```
# Start with the axiom
current_sentence = "A"

next_sentence = ""

# For every character of the current sentence
for character in current_sentence:
    # Apply the production rules A->AB, B->A
    if character == "A":
        next_sentence = next_sentence + "AB"
    elif character == "B":
        next_sentence = next_sentence + "A"

current_sentence = next_sentence
```

Example: A Simple L-system

Alphabet	A, B
Axiom	A
Rules	$A \rightarrow AB$ $B \rightarrow A$

```
# Start with the axiom
current_sentence = "A"

next_sentence = ""

# For every character of the current sentence
for character in current_sentence:
    # Apply the production rules A->AB, B->A
    if character == "A":
        next_sentence = next_sentence + "AB"
    elif character == "B":
        next_sentence = next_sentence + "A"

current_sentence = next_sentence
```

Example: A Simple L-system

Alphabet	A, B
Axiom	A
Rules	$A \rightarrow AB$ $B \rightarrow A$

```
# Start with the axiom
current_sentence = "A"

next_sentence = ""

# For every character of the current sentence
for character in current_sentence:
    # Apply the production rules A->AB, B->A
    if character == "A":
        next_sentence = next_sentence + "AB"
    elif character == "B":
        next_sentence = next_sentence + "A"

current_sentence = next_sentence
```

Example: A Simple L-system

Alphabet	A, B
Axiom	A
Rules	$A \rightarrow AB$ $B \rightarrow A$

```
0: AB
1: ABA
2: ABAAB
3: ABAABABA
4: ABAABABAABAAB
5: ABAABABAABAABABAABABA
6: ABAABABAABAABABAABABAABAABABAABAAB
7: ABAABABAABAABABAABABAABAABABAABAABABAABABAABAABABAABABA
```

Another Example: L-system Fractal

Alphabet	A, B
Axiom	A
Rules	$A \rightarrow ABA$ $B \rightarrow BBB$

Another Example: L-system Fractal

Alphabet	A, B
Axiom	A
Rules	$A \rightarrow ABA$ $B \rightarrow BBB$

Generation 0	A

Another Example: L-system Fractal

Alphabet	A, B
Axiom	A
Rules	$A \rightarrow ABA$ $B \rightarrow BBB$

Generation 0	A
Generation 1	ABA

Another Example: L-system Fractal

Alphabet	A, B
Axiom	A
Rules	$A \rightarrow ABA$ $B \rightarrow BBB$

Generation 0	A
Generation 1	ABA
Generation 2	$ABABBBABA$

Another Example: L-system Fractal

Alphabet	A, B
Axiom	A
Rules	$A \rightarrow ABA$ $B \rightarrow BBB$

Generation 0	A
Generation 1	ABA
Generation 2	$ABABBBABA$
Generation 3	$ABABBBABABBBBBBBBBABABBBABA$

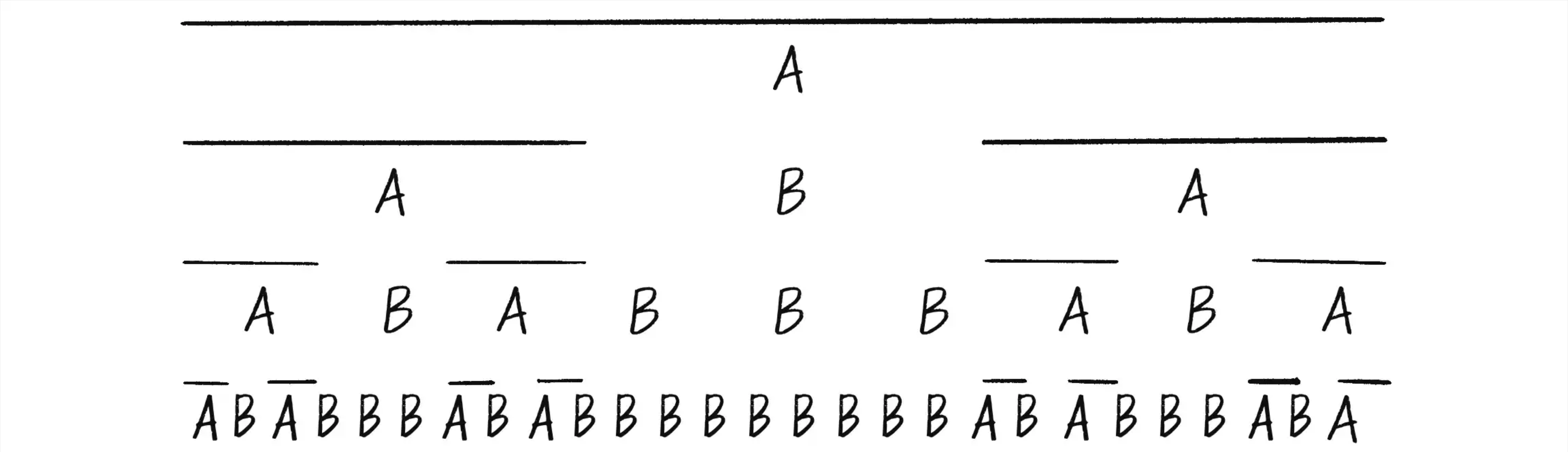
Another Example: L-system Fractal

To turn this into a drawing,
translate the system's alphabet:

A	Draw a line forward
B	Move forward (without drawing a line)

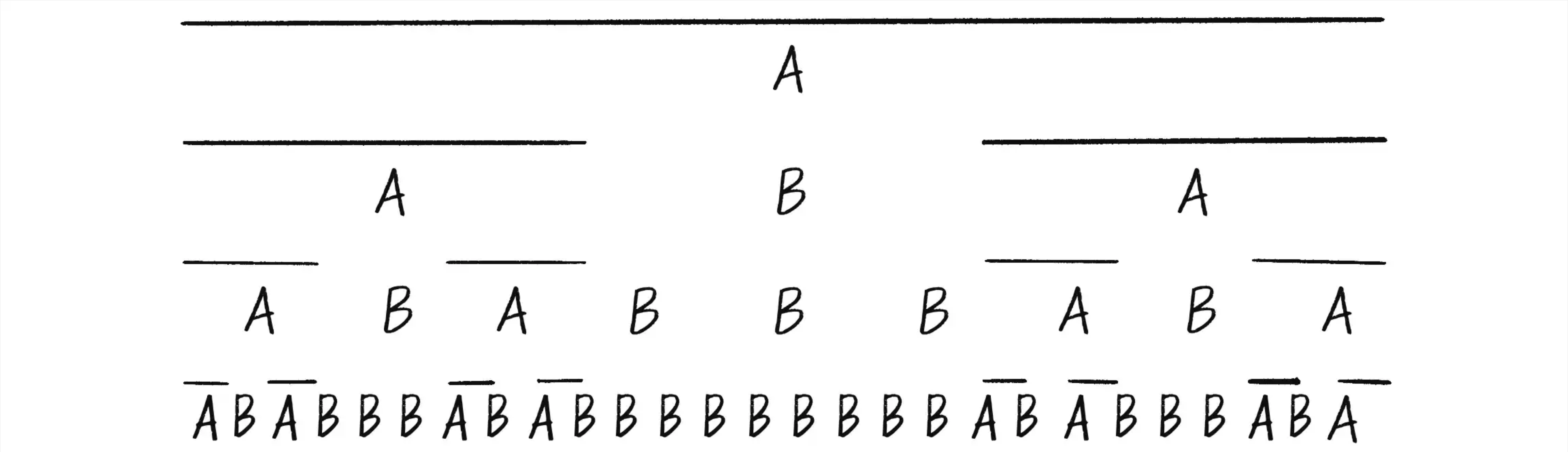
Another Example: L-system Fractal

A	Draw a line forward
B	Move forward (without drawing a line)



Another Example: L-system Fractal

Cantor Set



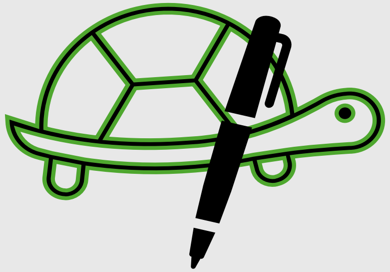
Drawing with L-systems

Drawing Alphabet:

F	Draw a line forward
G	Move forward (without drawing a line)
$+$	Turn right
$-$	Turn left
$[$	Save current state
$]$	Restore current state

Drawing with L-systems

This type of drawing framework is often referred to as **turtle graphics**



Drawing with L-systems in Py5

Drawing Alphabet with Py5 code:

F	`line(0, 0, 0, length)` `translate(0, length)`

Drawing with L-systems in Py5

Drawing Alphabet with Py5 code:

F	`line(0, 0, 0, length)` `translate(0, length)`
G	`translate(0, length)`

Drawing with L-systems in Py5

Drawing Alphabet with Py5 code:

F	`line(0, 0, 0, length)` `translate(0, length)`
G	`translate(0, length)`
$+$	`rotate(angle)`

Drawing with L-systems in Py5

Drawing Alphabet with Py5 code:

F	`line(0, 0, 0, length)` `translate(0, length)`
G	`translate(0, length)`
$+$	`rotate(angle)`
$-$	`rotate(-angle)`

Drawing with L-systems in Py5

Drawing Alphabet with Py5 code:

F	`line(0, 0, 0, length)` `translate(0, length)`
G	`translate(0, length)`
$+$	`rotate(angle)`
$-$	`rotate(-angle)`
$[$	`push()`

Drawing with L-systems in Py5

Drawing Alphabet with Py5 code:

F	`line(0, 0, 0, length)` `translate(0, length)`
G	`translate(0, length)`
$+$	`rotate(angle)`
$-$	`rotate(-angle)`
$[$	`push()`
$]$	`pop()`

Example: L-system with Turtle Graphics

Alphabet	$F, G, +, -, [,]$
Axiom	F
Rules	$F \rightarrow FF + [+F - F - F] - [-F + F + F]$

Example: L-system with Turtle Graphics

Alphabet	$F, G, +, -, [,]$
Axiom	F
Rules	$F \rightarrow FF + [+F - F - F] - [-F + F + F]$

Generation 0	F

Example: L-system with Turtle Graphics

Alphabet	$F, G, +, -, [,]$
Axiom	F
Rules	$F \rightarrow FF + [+F - F - F] - [-F + F + F]$

Generation 0	F
Generation 1	$FF + [+F - F - F] - [-F + F + F]$

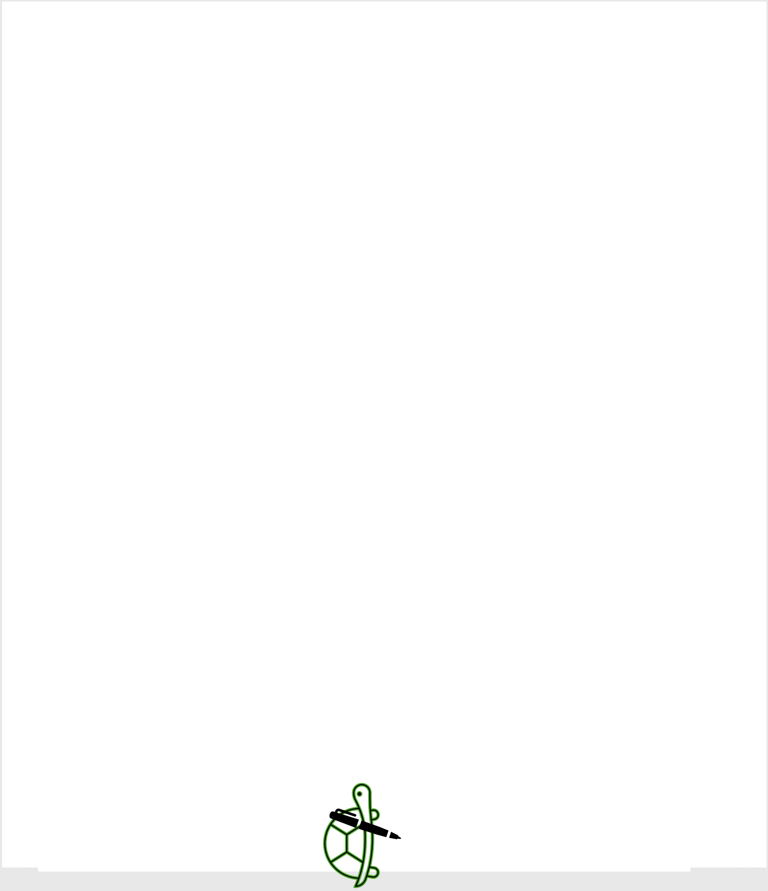
Example: L-system with Turtle Graphics

Alphabet	$F, G, +, -, [,]$
Axiom	F
Rules	$F \rightarrow FF + [+F - F - F] - [-F + F + F]$

Generation 0	F
Generation 1	$FF + [+F - F - F] - [-F + F + F]$
Generation 2	$FF + [+F - F - F] - [-F + F + F]FF + [+F - F - F] - [-F + F + F] + [+FF + [+F - F - F] - [-F + F + F] - FF + [+F - F - F] - [-F + F + F] - FF + [+F - F - F] - [-F + F + F]] - [-FF + [+F - F - F] - [-F + F + F] + FF + [+F - F - F] - [-F + F + F] + FF + [+F - F - F] - [-F + F + F]]$

Example: L-system with Turtle Graphics

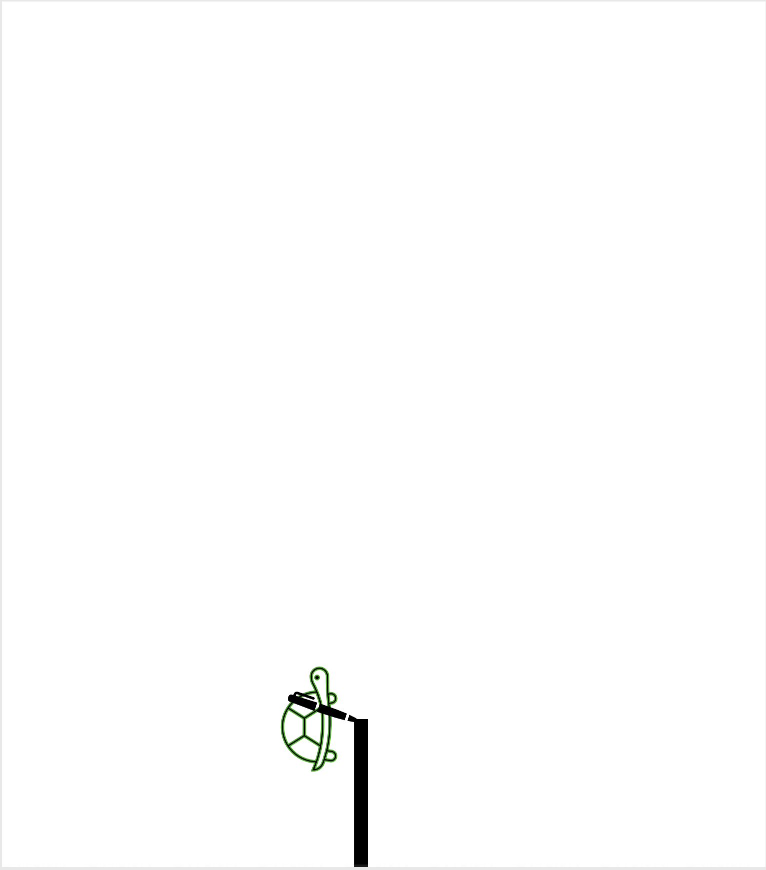
Generation 0 | F



Example: L-system with Turtle Graphics

Generation 0

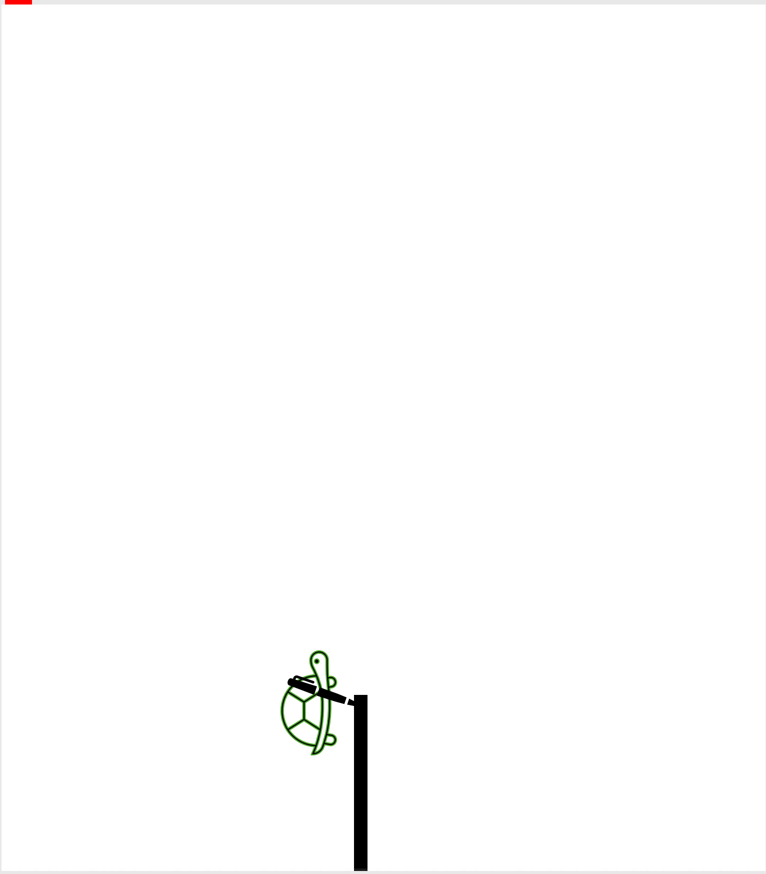
F



Example: L-system with Turtle Graphics

Generation 0	F
Generation 1	$FF + [+F - F - F] - [-F + F + F]$

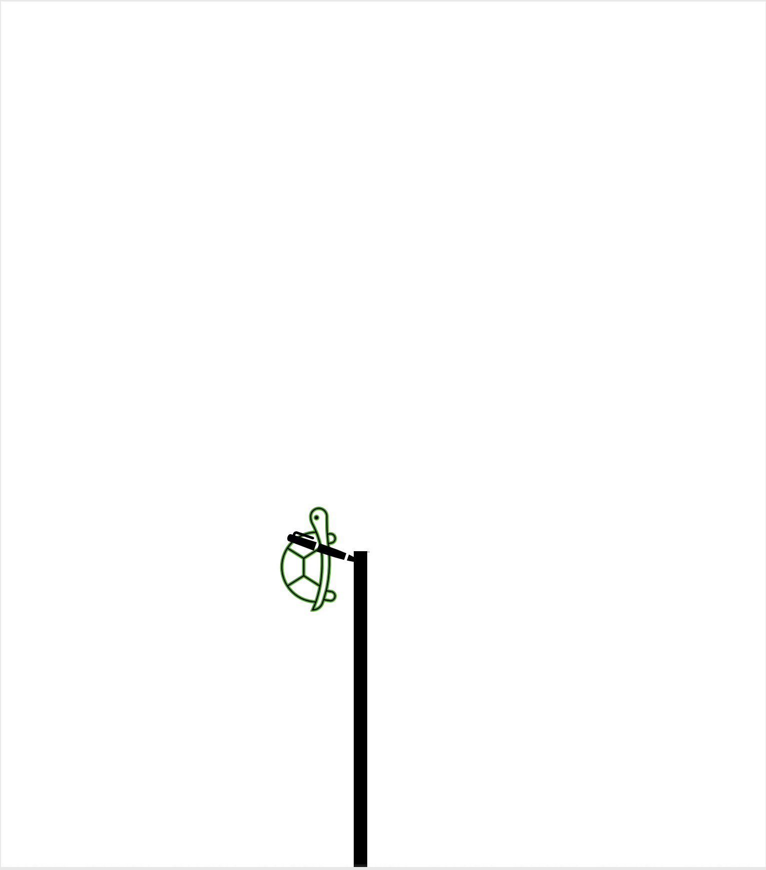
Angle = 25°



Example: L-system with Turtle Graphics

Generation 0	F
Generation 1	$FF + [+F - F - F] - [-F + F + F]$

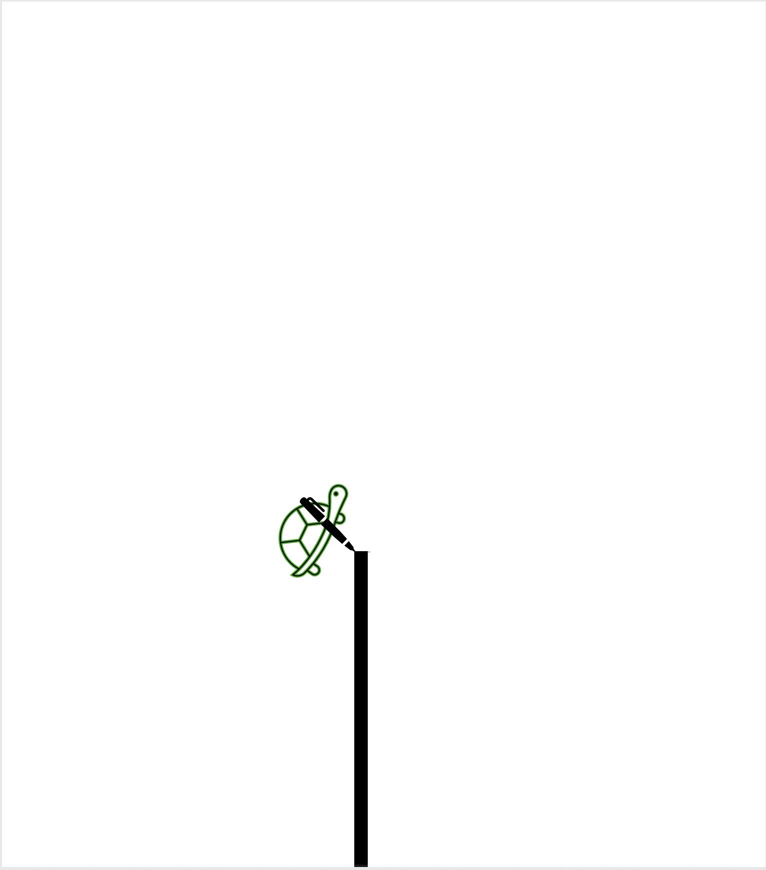
Angle = 25°



Example: L-system with Turtle Graphics

Generation 0	F
Generation 1	$FF + [+F - F - F] - [-F + F + F]$

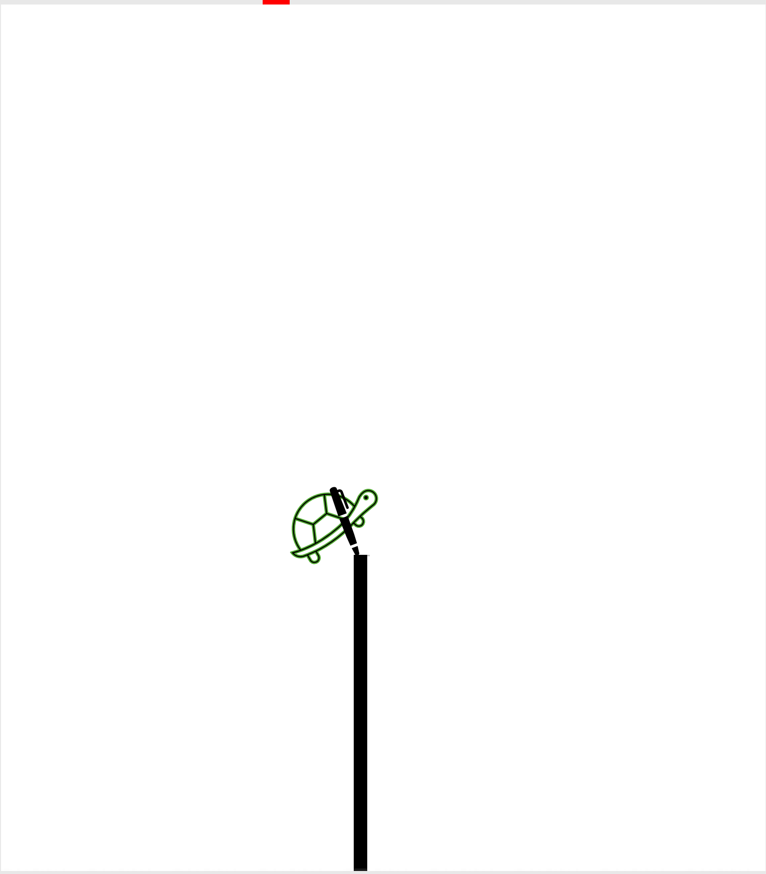
Angle = 25°



Example: L-system with Turtle Graphics

Generation 0	F
Generation 1	$FF + [+F - F - F] - [-F + F + F]$

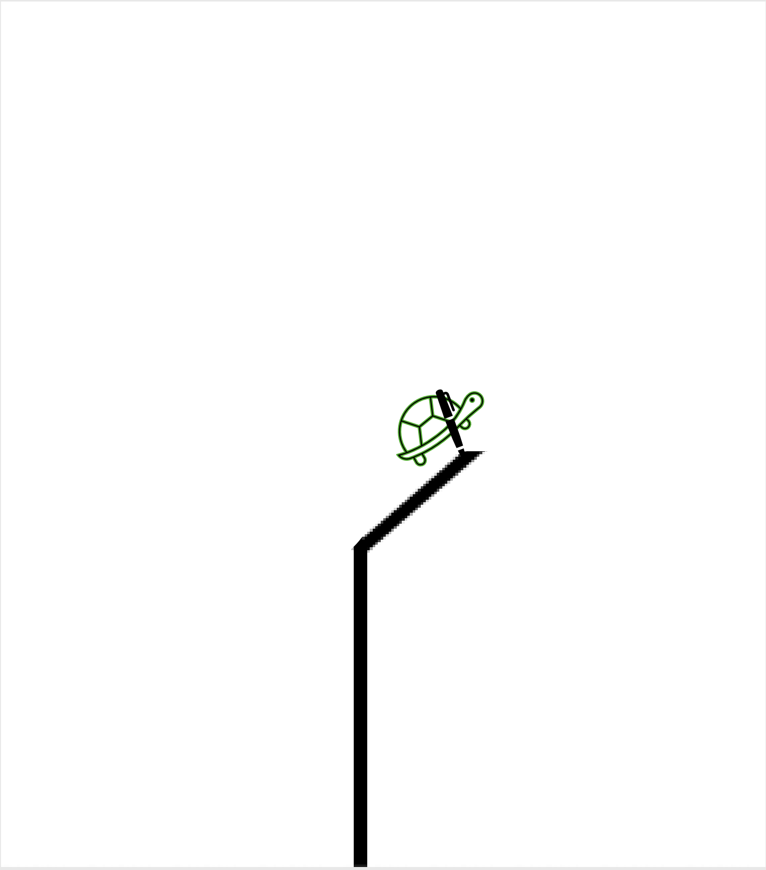
Angle = 25°



Example: L-system with Turtle Graphics

Generation 0	F
Generation 1	$FF + [+F - F - F] - [- F + F + F]$

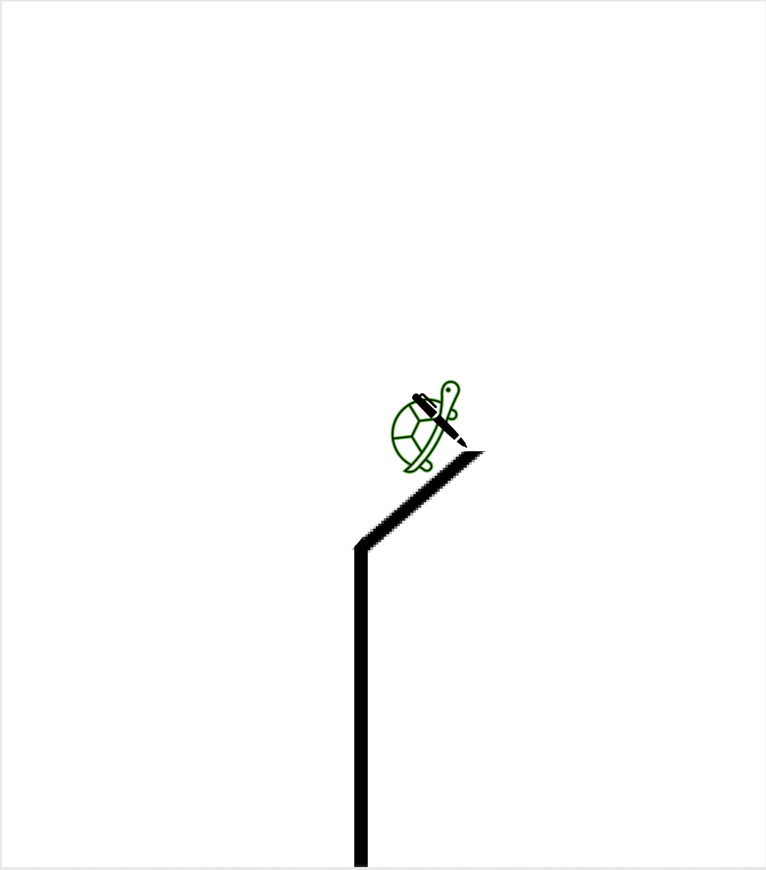
Angle = 25°



Example: L-system with Turtle Graphics

Generation 0	F
Generation 1	$FF+[+F-F-F]-[-F+F+F]$

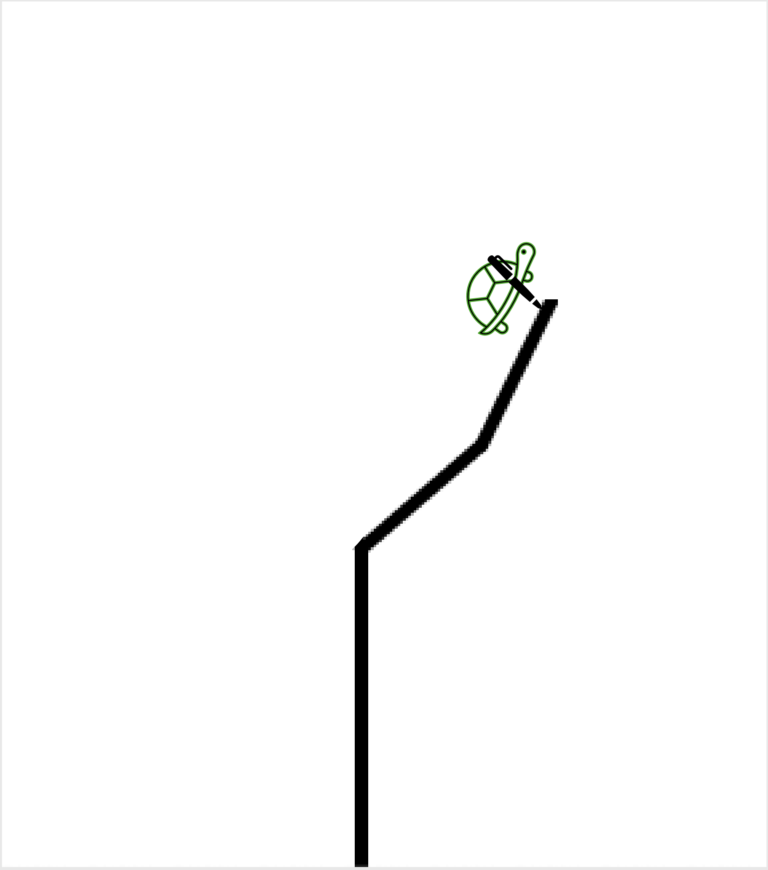
Angle = 25°



Example: L-system with Turtle Graphics

Generation 0	F
Generation 1	$FF + [+F - F - F] - [-F + F + F]$

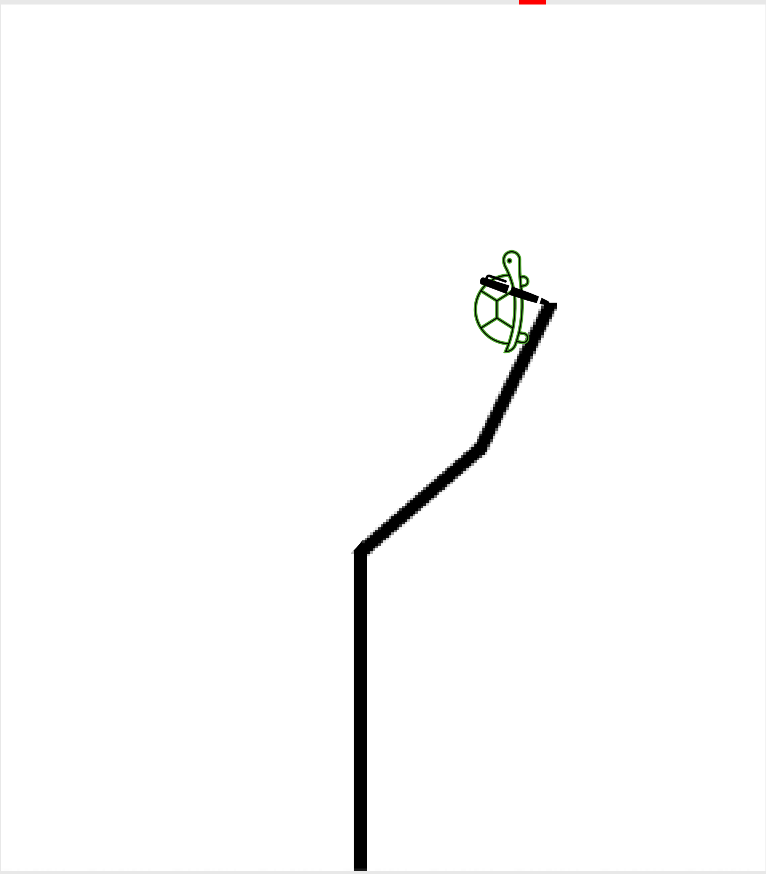
Angle = 25°



Example: L-system with Turtle Graphics

Generation 0	F
Generation 1	$FF + [+F - F - F] - [-F + F + F]$

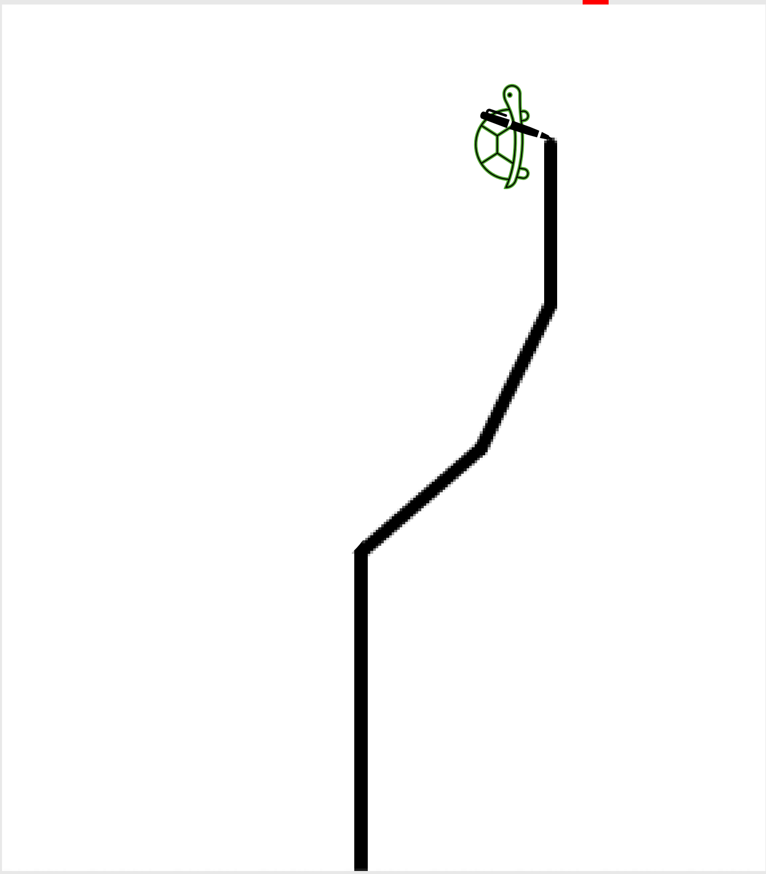
Angle = 25°



Example: L-system with Turtle Graphics

Generation 0	F
Generation 1	$FF+[+F-F-F]-[-F+F+F]$

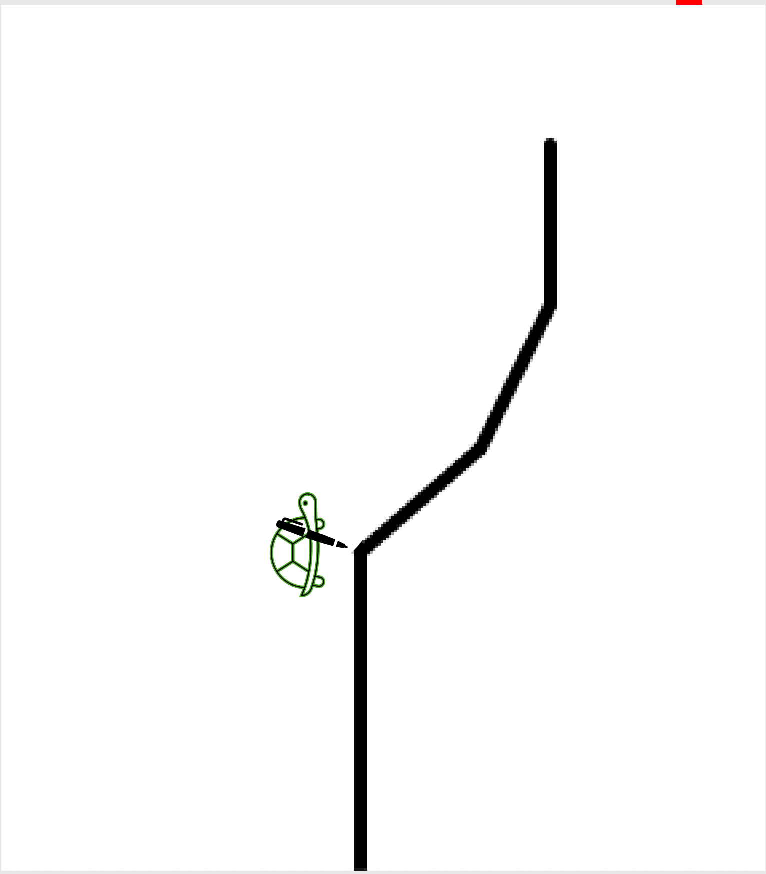
Angle = 25°



Example: L-system with Turtle Graphics

Generation 0	F
Generation 1	$FF + [+F - F - F] - [-F + F + F]$

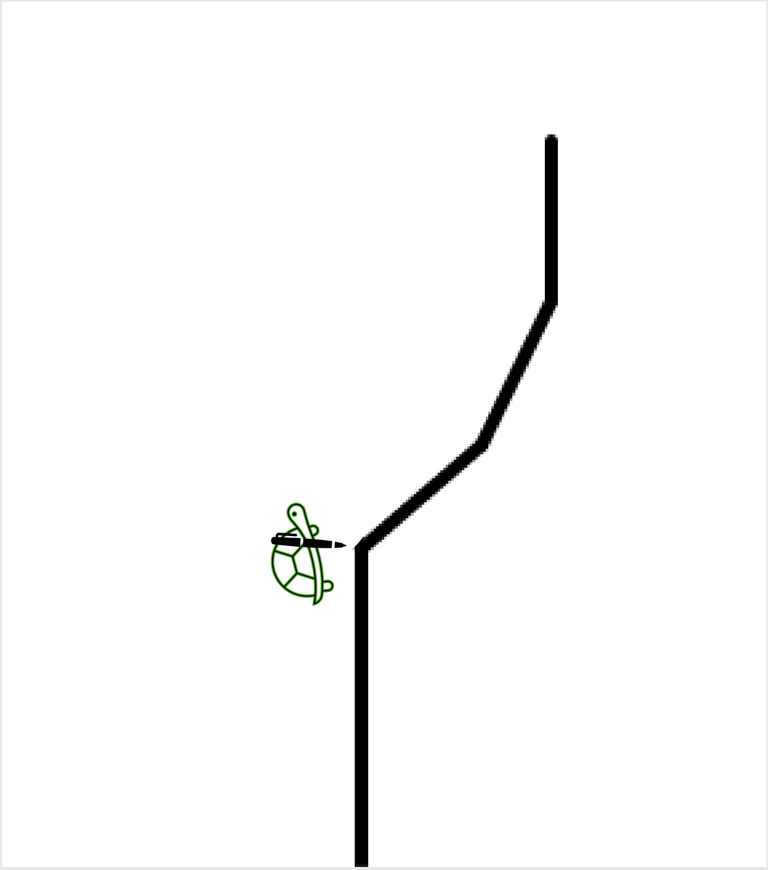
Angle = 25°



Example: L-system with Turtle Graphics

Generation 0	F
Generation 1	$FF + [+F - F - F] - [-F + F + F]$

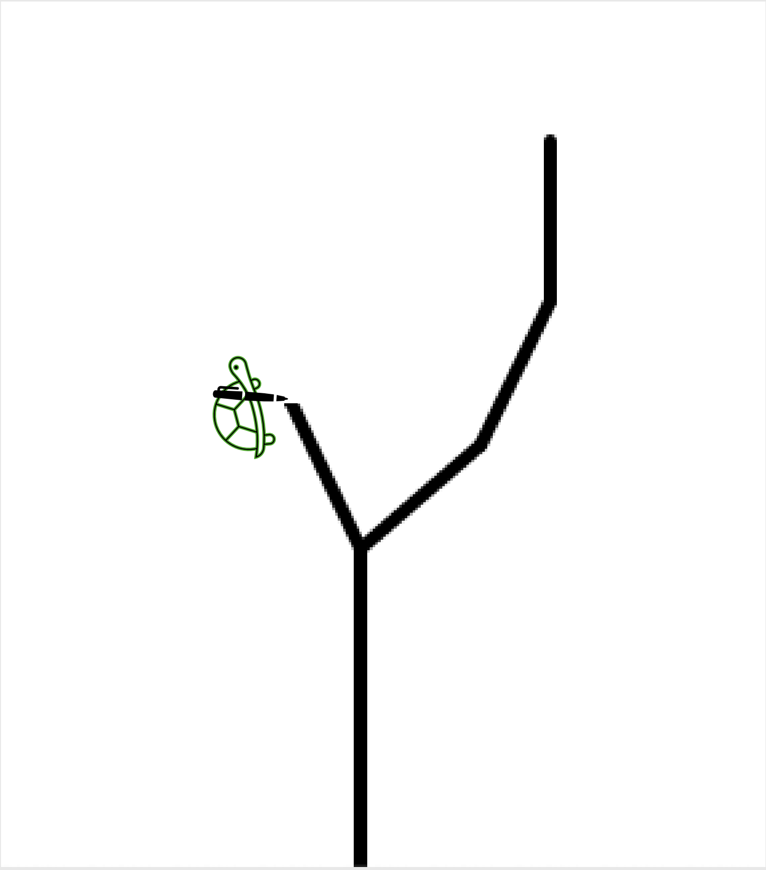
Angle = 25°



Example: L-system with Turtle Graphics

Generation 0	F
Generation 1	$FF + [+F - F - F] - [-F + F + F]$

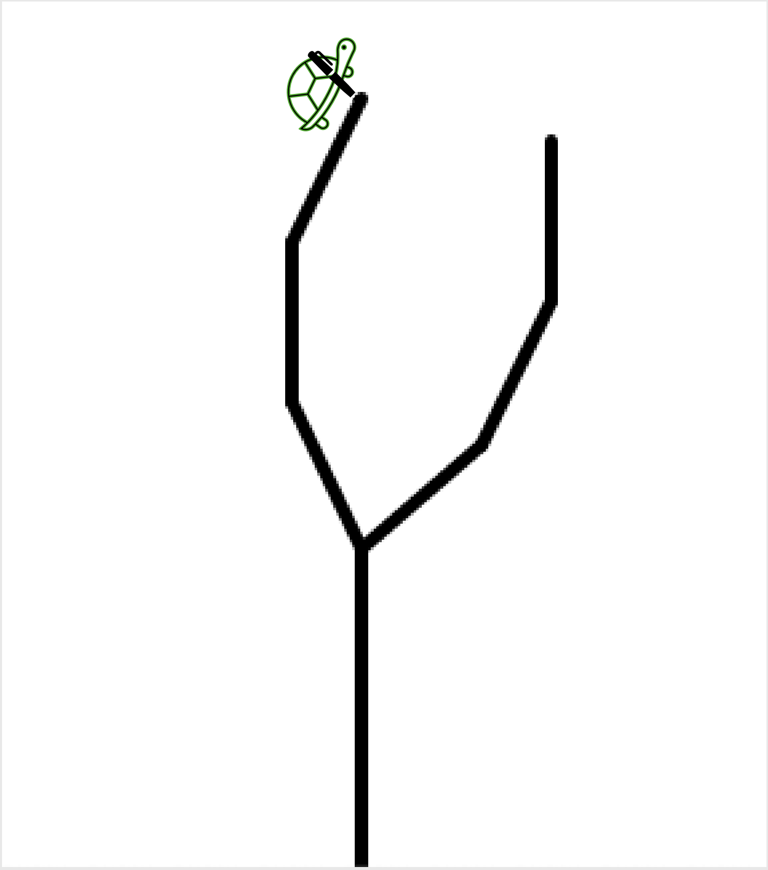
Angle = 25°



Example: L-system with Turtle Graphics

Generation 0	F
Generation 1	$FF + [+F - F - F] - [-F + F + F]$

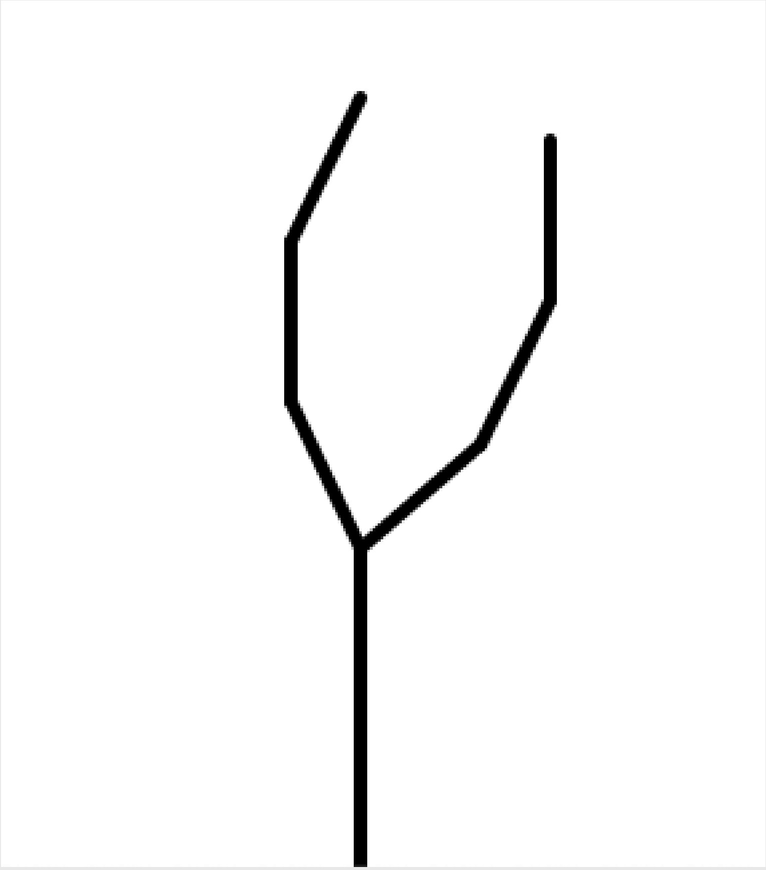
Angle = 25°



Example: L-system with Turtle Graphics

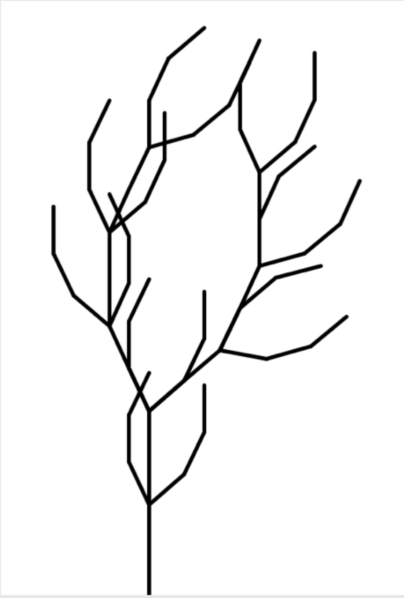
Generation 0	F
Generation 1	$FF + [+F - F - F] - [-F + F + F]$

Angle = 25°

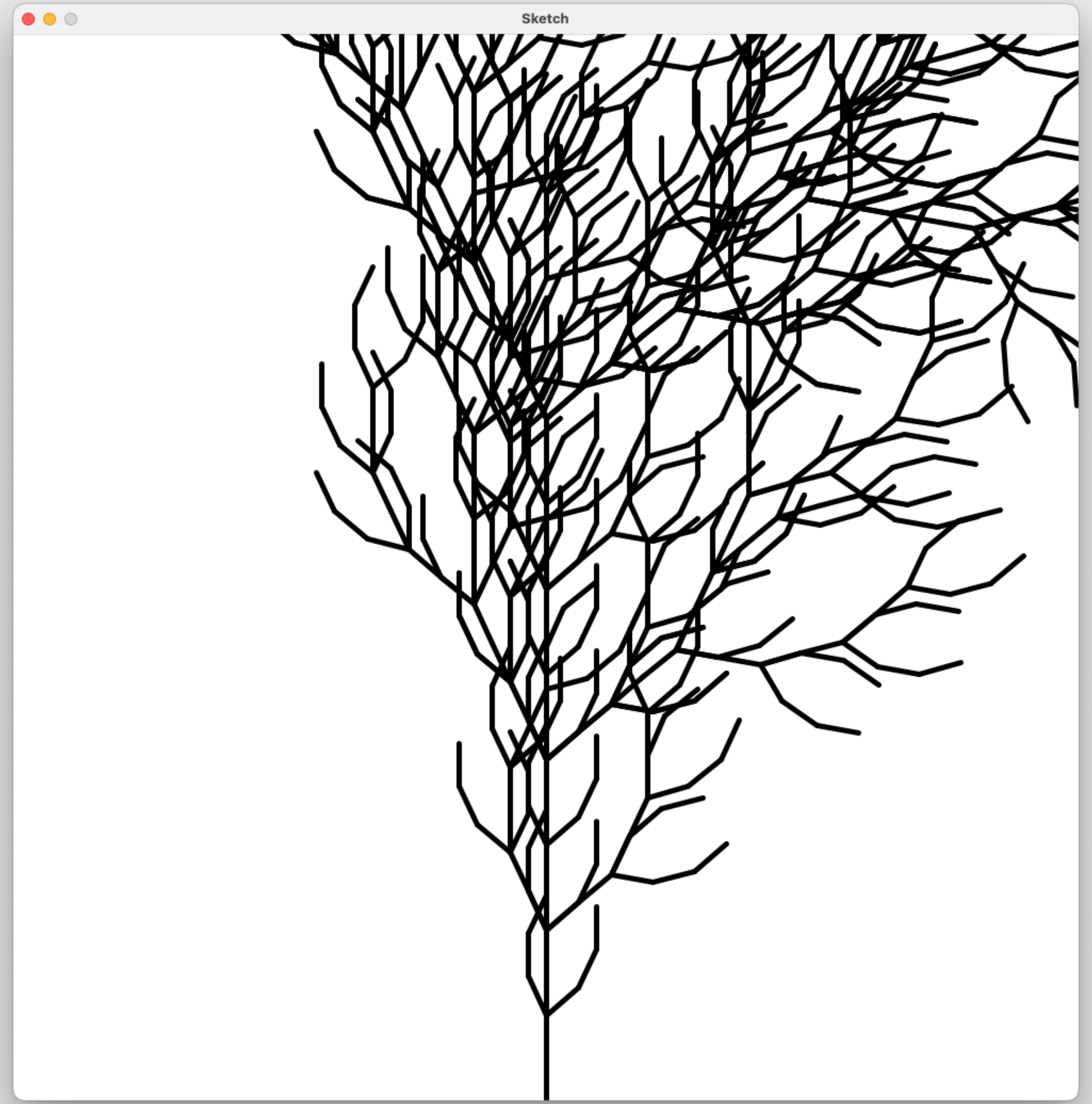


Example: L-system with Turtle Graphics

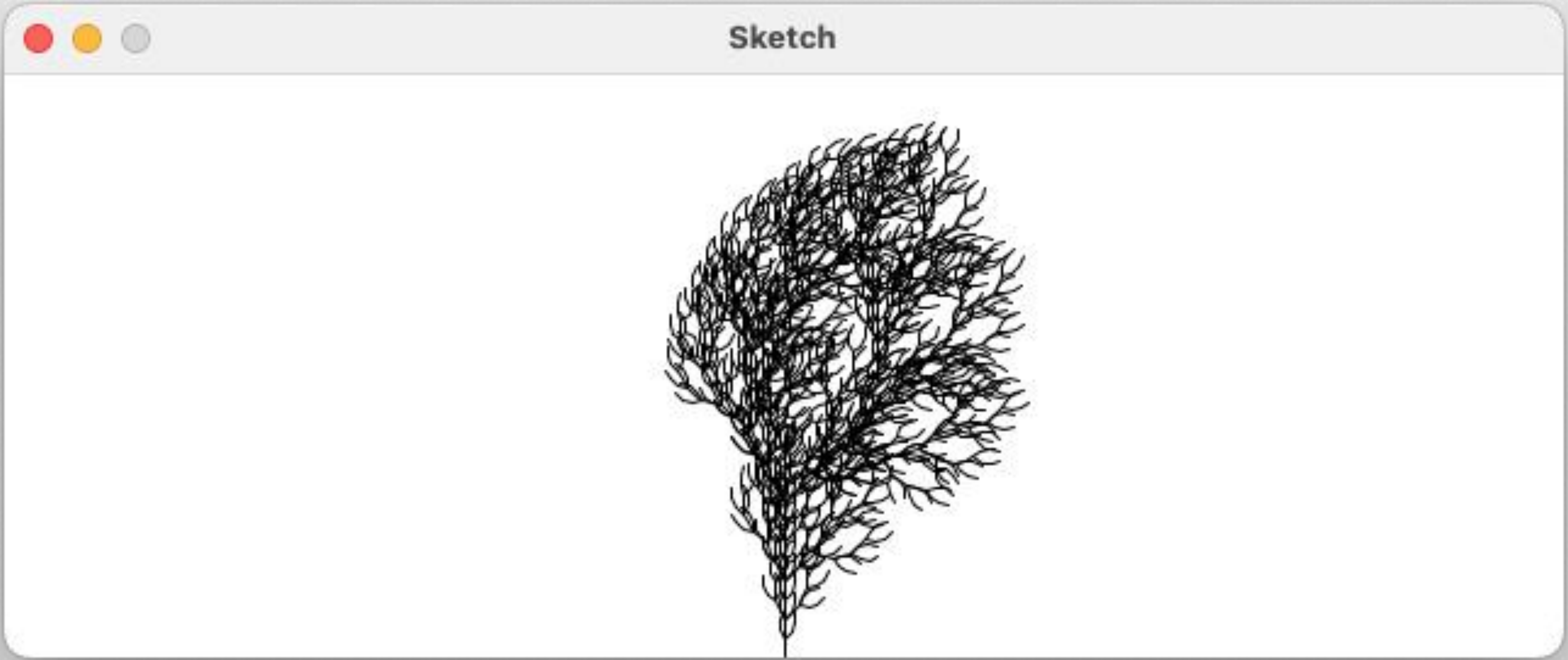
Generation 0	F
Generation 1	$FF + [+F - F - F] - [-F + F + F]$
Generation 2	$FF + [+F - F - F] - [-F + F + F]FF + [+F - F - F] - [-F + F + F] + [+FF + [+F - F - F] - [-F + F + F] - FF + [+F - F - F] - [-F + F + F] - FF + [+F - F - F] - [-F + F + F]] - [-FF + [+F - F - F] - [-F + F + F] + FF + [+F - F - F] - [-F + F + F] + FF + [+F - F - F] - [-F + F + F]]$



Example: L-system with Turtle Graphics

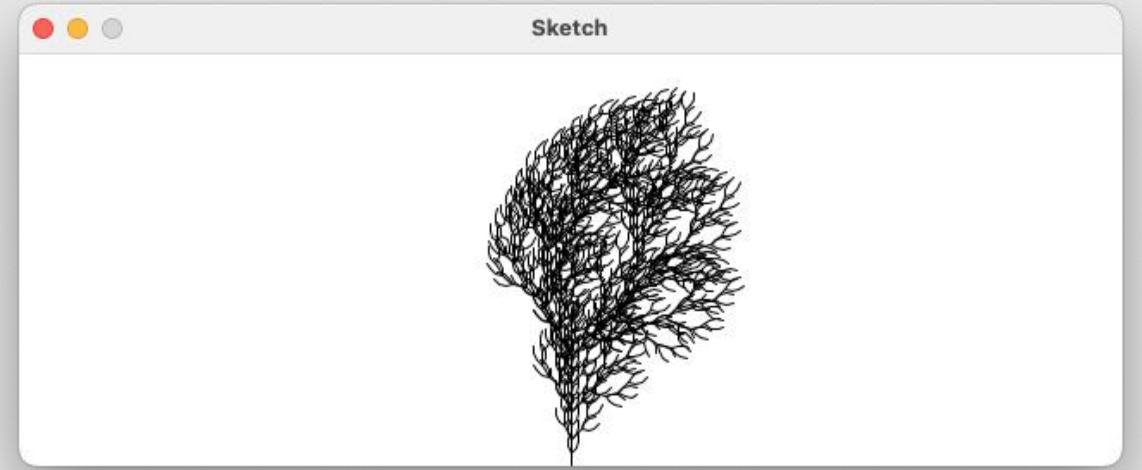


Example: L-system with Turtle Graphics

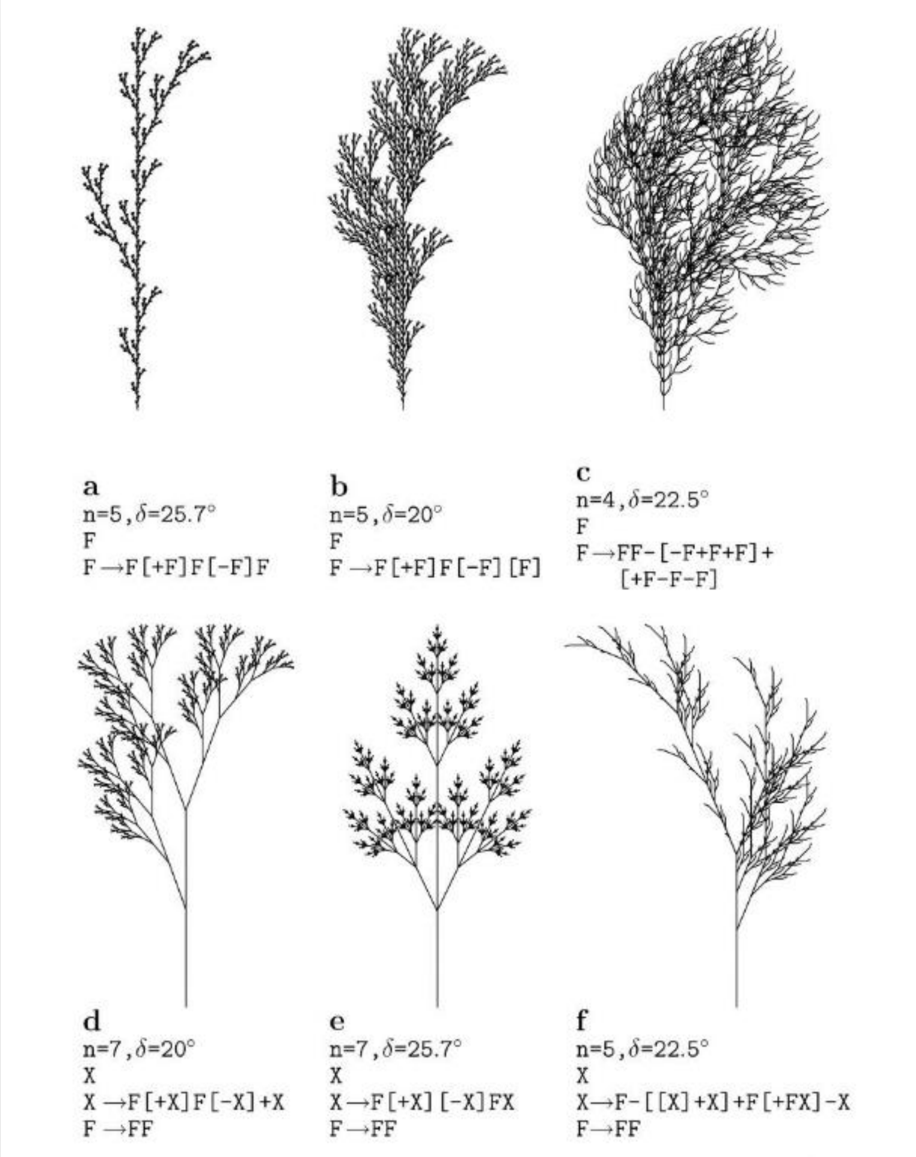


Example: L-system with Turtle Graphics

Alphabet	$F, G, +, -, [,]$
Axiom	F
Rules	$F \rightarrow FF + [+F - F - F] - [-F + F + F]$



The Algorithmic Beauty Of Plants

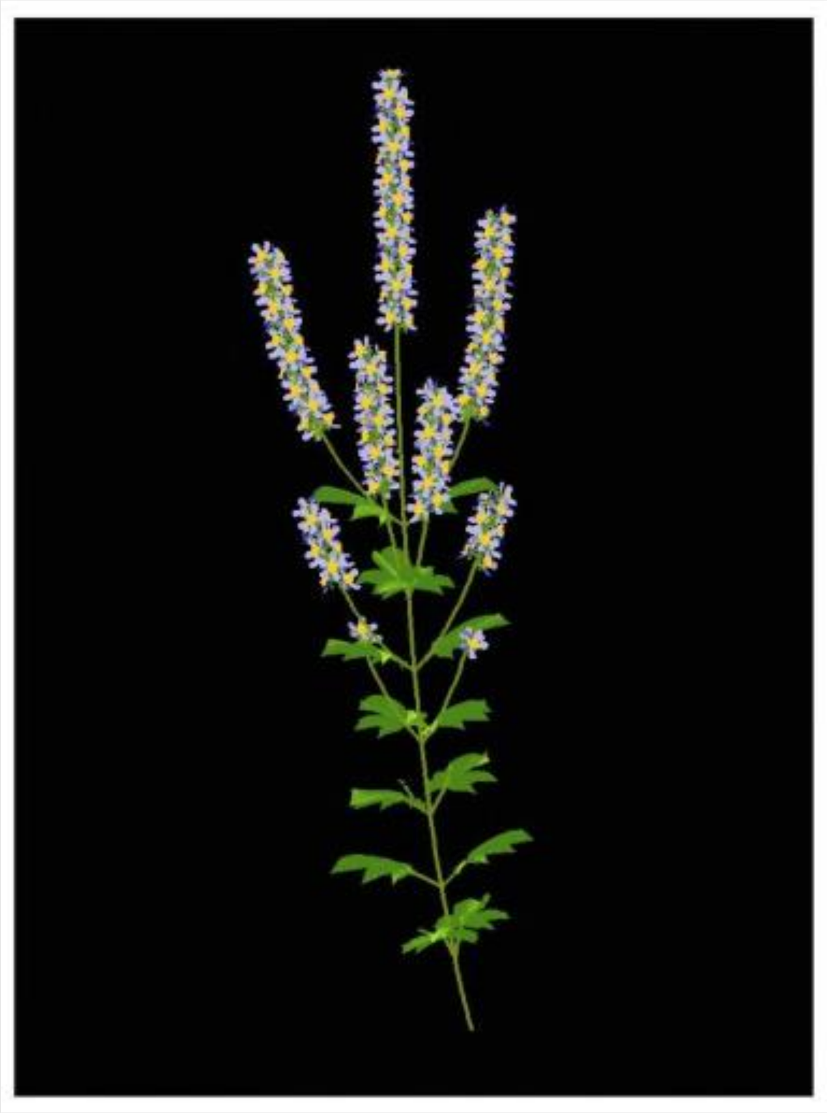


The Algorithmic Beauty Of Plants

n=7, δ=22.5°

ω : A
p_1 : A → [&FL!A]/////'[&FL!A]///////'[&FL!A]
p_2 : F → S ///// F
p_3 : S → F L
p_4 : L → ['''∧∧{-f+f+f-|-f+f+f}]

The Algorithmic Beauty Of Plants



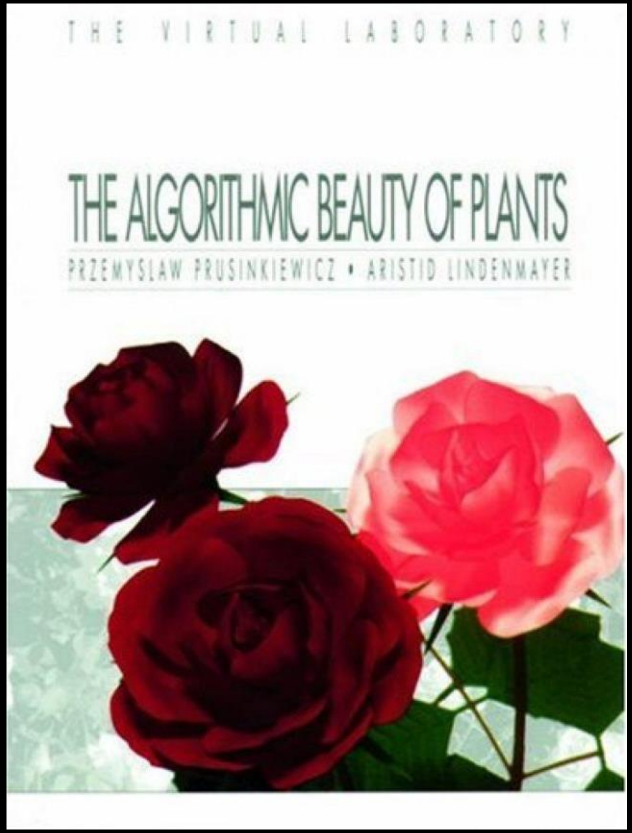
L-system Research

https://www.researchgate.net/publication/220184753_Modeling_and_Rendering_of_Realistic_Feathers

Lindenmayer Systems

“The central concept of L-systems is that of rewriting. In general, rewriting is a technique for defining complex objects by successively replacing parts of a simple initial object using a set of *rewriting rules* or production.”

- Aristid Lindenmayer



https://archive.org/details/the-algorithmic-beauty-of-plants