Turtle Plots

Moving the Turtle

- Idea: trace walk-path

Moving the Turtle

- Idea: trace walk-path

C++ Easy Commands

- Step (drawn): `turtle::forward();`
- Rotation left: `turtle::left(my_angle);`
- Rotation right: `turtle::right(my_angle);`

Requires:

a) `#include "turtle.cpp"`

b) `turtle.cpp` and `bitmap.cpp` have to be in the same folder as your program.

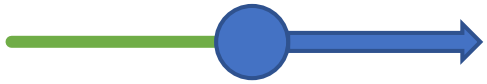
turtle.cpp and bitmap.cpp can be downloaded from the lecture website. Furthermore, there are additional turtle commands available, see Turtle_Extended.pdf

Moving the Turtle

```
turtle::forward();
turtle::left(45);
turtle::forward();
turtle::right(90);
turtle::forward();
```

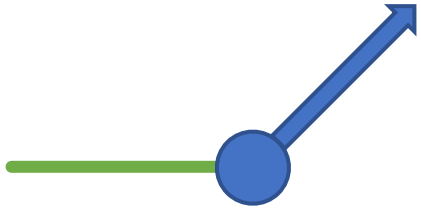
Moving the Turtle

```
turtle::forward();
turtle::left(45);
turtle::forward();
turtle::right(90);
turtle::forward();
```



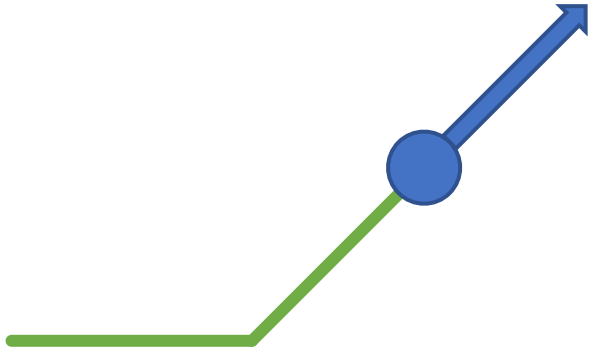
Moving the Turtle

```
turtle::forward();
turtle::left(45);
turtle::forward();
turtle::right(90);
turtle::forward();
```



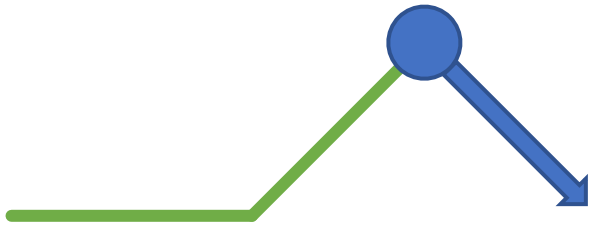
Moving the Turtle

```
turtle::forward();
turtle::left(45);
turtle::forward();
turtle::right(90);
turtle::forward();
```



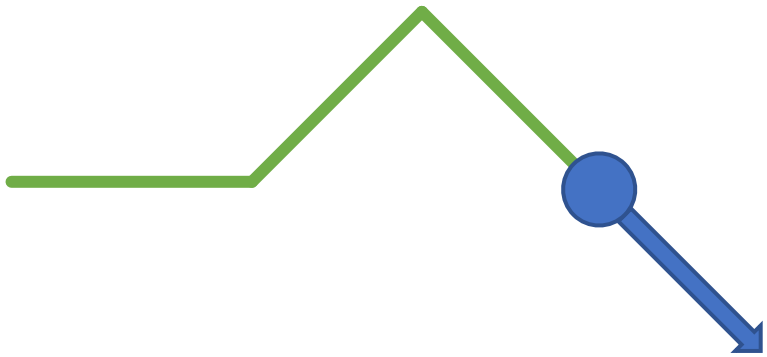
Moving the Turtle

```
turtle::forward();
turtle::left(45);
turtle::forward();
turtle::right(90);
turtle::forward();
```



Moving the Turtle

```
turtle::forward();
turtle::left(45);
turtle::forward();
turtle::right(90);
turtle::forward();
```



Lindenmayer Systems

Lindenmayer Systems

- Characterized by three things:
  1. Alphabet Σ - the allowed symbols
  2. Production P - how to replace each symbol
  3. Initial word s - the word to start with

Lindenmayer Systems

- Characterized by three things:
  1. Alphabet Σ - the allowed symbols
  2. Production P - how to replace each symbol
  3. Initial word s - the word to start with

- Example:
  1. $\Sigma := \{F, +, -\}$
  2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
  3. $s := F$

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1:

w_2:

w_3:

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2:

w_3:

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2:

w_3:

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2: $\mathbb{F}+\mathbb{F}+$

w_3:

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2: $\mathbb{F}+\mathbb{F}++$

w_3:

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}+$

w_3:

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}++$

w_3:

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}++$

w_3:

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}++$

w_3: $\mathbb{F}+\mathbb{F}+$

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}++$

w_3: $\mathbb{F}+\mathbb{F}++$

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}++$

w_3: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}+$

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}++$

w_3: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}++$

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}++$

w_3: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}+++$

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}++$

w_3: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}+++\mathbb{F}+\mathbb{F}+$

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}++$

w_3: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}+++\mathbb{F}+\mathbb{F}++$

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}++$

w_3: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}+++\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}+$

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}++$

w_3: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}+++\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}+$

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}++$

w_3: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}+++\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}++$

Lindenmayer Systems

- How does it look after 3 rounds?

1. $\Sigma := \{F, +, -\}$
2. $P := \begin{cases} F \mapsto F + F + \\ + \mapsto + \\ - \mapsto - \end{cases}$
3. $s := F$

s: $\mathbb{F}$

w_1: $\mathbb{F}+\mathbb{F}+$

w_2: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}++$

w_3: $\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}+++\mathbb{F}+\mathbb{F}++\mathbb{F}+\mathbb{F}+++$

Draw Lindenmayer Systems

Two Step Procedure

- Goal: Draw n-th step of Lindenmayer system

- Done in 2 steps
  1. Obtain n-th step
  2. Draw it

Step 1 – Obtain n-th Word

- Write and use the following two functions
  - **`std::string production (const char c)`**
    - In: symbol e.g. `F`
    - Out: its production e.g. `F+F+`

Step 1 – Obtain n-th Word

- Write and use the following two functions
  - **`std::string production (const char c)`**
    - In: symbol e.g. `F`
    - Out: its production e.g. `F+F+`
  - **`std::string next_word (const std::string word)`**
    - In: w_n (Word of step `n`) e.g. `FF`
    - Out: w_{n+1} (Word of step `n+1`) e.g. `F+F+F+F+`
    - Applies `production` to each character in w_n and concatenates the results.

Step 2 – Draw It

- Idea: view alphabet as turtle commands

- Example:

Alphabet: $\Sigma := \{F, +, -\}$

F	`turtle::forward()`
$+$	`turtle::left(90)`
$-$	`turtle::right(90)`