How to model plant architecture in GroIMP

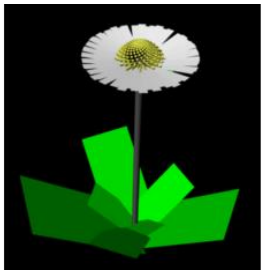
Katarína Smoleňová

FSPM & GroIMP Workshop

PMBDA 2025, Nanchang, China

November 1, 2025

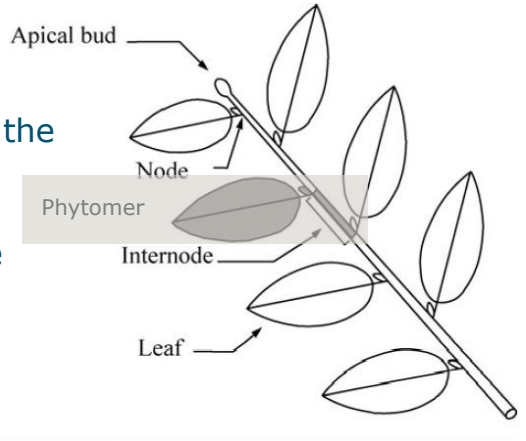
```
Axiom ==>
  // create rosette of 7 leaves
  for (int i:(1:7)) (
    [
      RH(i * 137.5)
      M((i - 1) / 2)
      RU(LEAF_ANGLE)
      RH(90)
      { double r = 50 - i * 5; }
      Leaf(r, r * 0.5)
    ]
  ) ...
;
```



WAGENINGEN
UNIVERSITY & RESEARCH

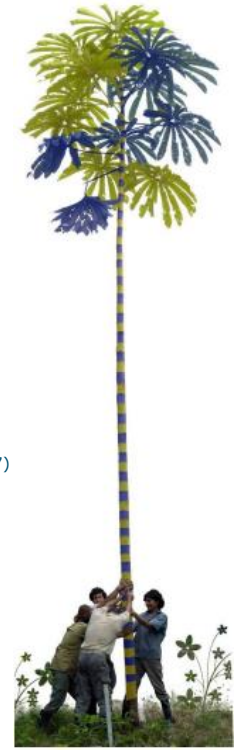
Plants as modular organisms

- **Plant architecture** defined by
  - The type of individual components making up the plant (*decomposition*),
  - Their shape, location, and orientation in space (*geometry*), and
  - How they are related to each other (*topology*)



(acc. Barthélémy & Caraglio, 2007)

- **Plants develop** by the *repetition* of these individual components (and/or botanical units) in time and space
  - Described by *a set of rules*



(from Heuret)

Plants as modular organisms

- **Plant architecture** defined by
  - The type of individual components making up the plant (*decomposition*),
  - Their shape, location, and orientation in space (*geometry*), and
  - How they are related to each other (*topology*)

- **Plants develop** by the *repetition* of these individual components (and/or botanical units) in time and space
  - Described by *a set of rules*

In XL:

- *Decomposition*:
  Definition of modules
- *Geometry:*
  3D objects & turtle commands
- *Topology:*
  Internal property of the graph (nodes & edges)

Rules:

- *L-system:* **==>**
- *Execution:* **::>**
- *General graph rewriting:* **==>>**

WAGENINGEN
UNIVERSITY & RESEARCH

Specifying development with rewriting rules

Modules: Bud, Internode, Leaf, Flower

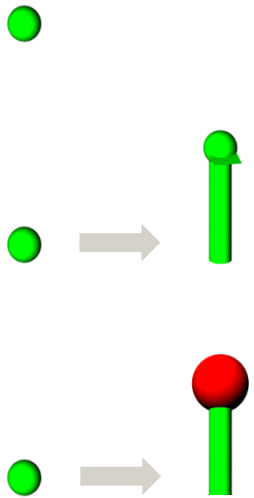
Axiom: Bud
(initial structure)

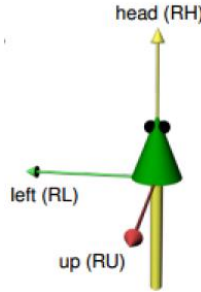
Rules:

```
Bud ==>
    Internode
    [RL(110) Leaf]
    RH(137.51) Bud
;
```

time == 6

```
Bud ==>
    Internode Flower
;
```





3D turtle graphics commands

Specifying development with rewriting rules

Modules: Bud, Internode, Leaf, Flower

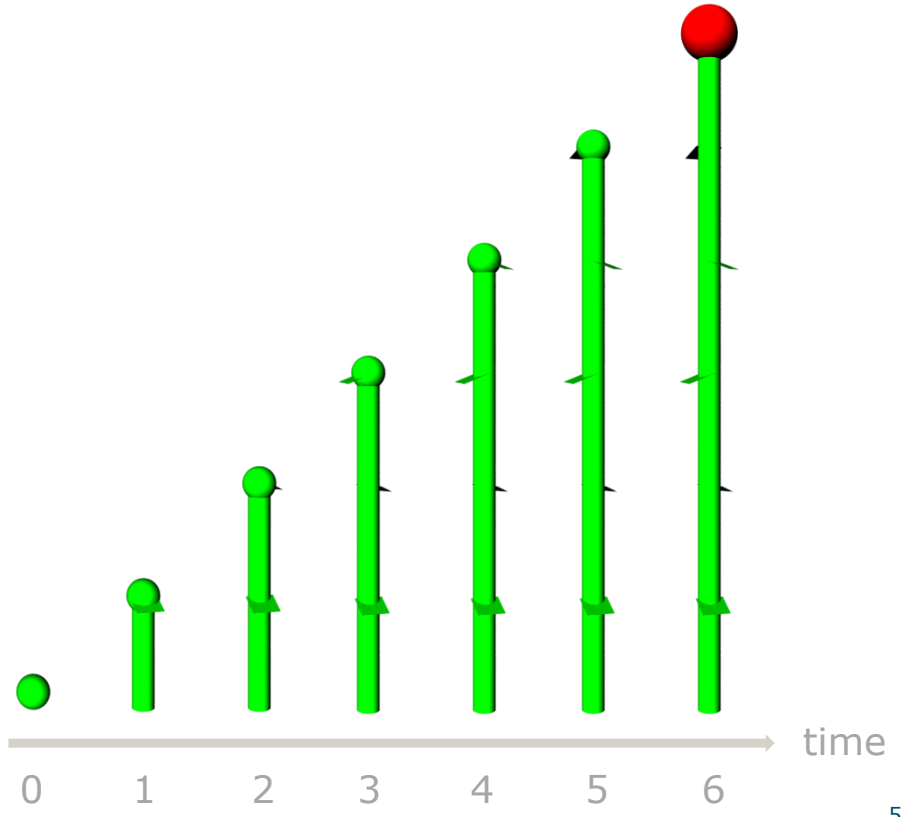
Axiom: Bud
(initial structure)

Rules:

```
Bud ==>
    Internode
    [RL(110) Leaf]
    RH(137.51) Bud
;
```

time == 6

```
Bud ==>
    Internode Flower
;
```



Execution rules

```
l:Leaf ::> { l.calcPhotosynthesis(); }
```

- Used to update attribute values of searched organs or execute imperative statements without changes in the (graph) structure

Rewriting (L-system) rule:

```
Leaf(length, width) ==>
    Leaf(length+0.015, width+0.005)
;
```

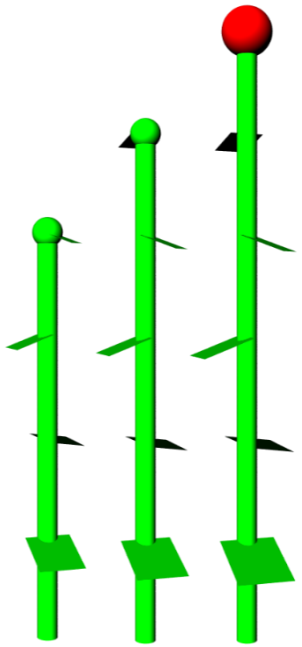
Execution rule:

```
l:Leaf ::> {
    l[length] += 0.015;
    l[width] += 0.005;
}
```

```
l:Leaf ::> {
    l[length] :+= 0.015;
    l[width] :+= 0.005;
}
```

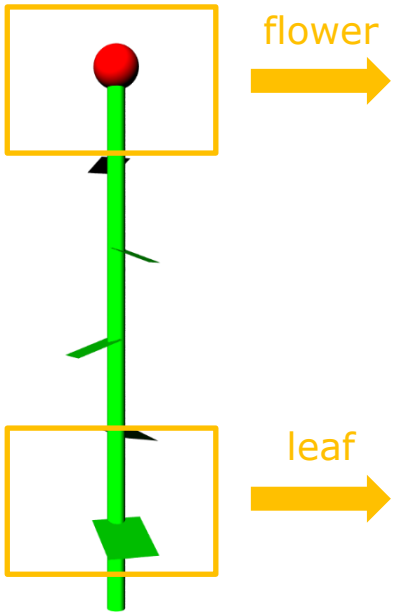
deferred assignment

(parallel execution of assignments)



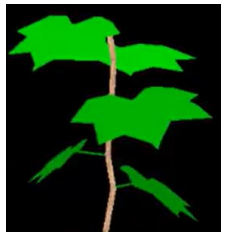
WAGENINGEN UNIVERSITY & RESEARCH

Improving geometrical representation

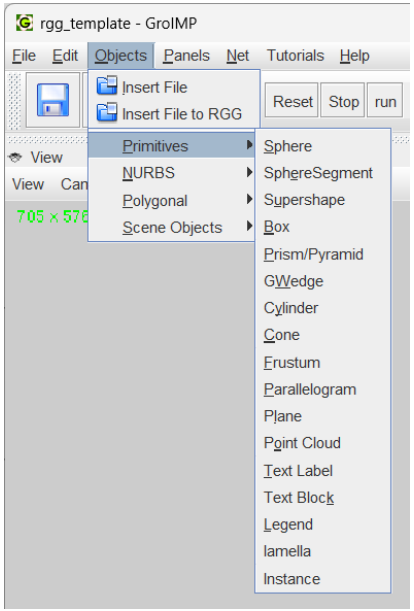


flower

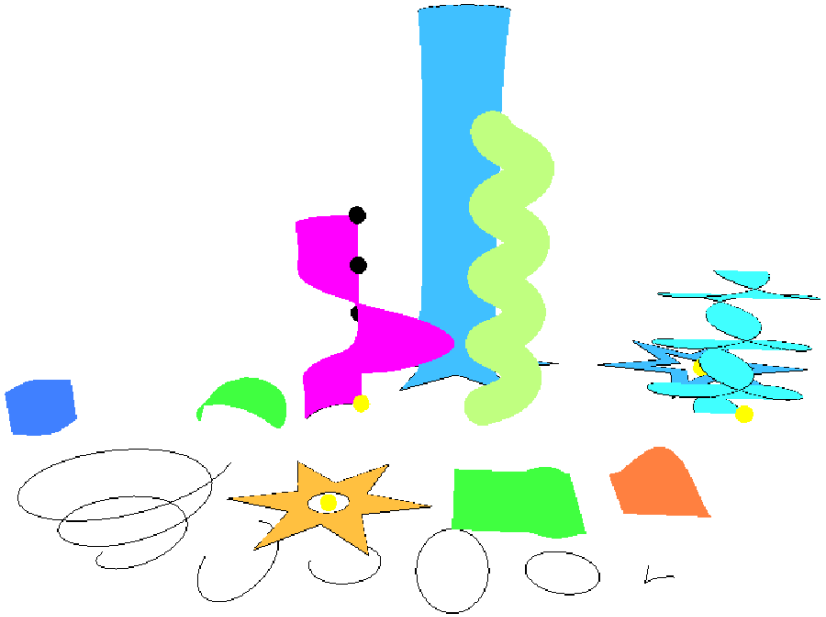
leaf



Supported objects in GroIMP



Primitives



Parametric curves & surfaces

Mesh

How to assign geometry to modules

- Derivation from an existing geometric object (Menu: Objects):

```
module Internode extends Cylinder(1, 0.1);
```

```
module Internode(super.length, super.radius) extends Cylinder(length, radius);
```

```
module Internode(super.length, float diameter) extends Cylinder(length, diameter/2);
```

- Using *instantiation* rules:

```
module Internode(float length, float diameter)
==> Cylinder(length, diameter/2);
```

Inside a rule:

```
Internode
```

```
Internode(1, 0.05)
```

```
Internode(1, 0.1)
```

```
Internode(1, 0.1)
```

Putting it all together in XL

```
// modules
module Bud extends Sphere(0.03).(setShader(GREEN));

module Internode extends Cylinder(0.2, 0.2).(setShader(GREEN));

module Leaf(double length, double width)
==> leaf(length, width); // leaf is a green parallelogram

module Flower extends Sphere(0.05).(setShader(RED));
```

```
// model variables & parameters
int time;
const double LEAF_ANGLE = 110;
const double PHYLLOTAXIS_ANGLE = 137.51;
```

```
// axiom (initial structure)
protected void init() [

    { time = 0; }

    Axiom ==>
        Bud
    ;
]
```

Putting it all together

```
// modules
module Bud extends Sphere(0.03).(setShader(GREEN));

module Internode extends Cylinder(0.2, 0.2).(setShader(GREEN));

module Leaf(double length, double width)
==> leaf(length, width); // leaf is a green parallelogram

module Flower extends Sphere(0.05).(setShader(RED));


// model variables & parameters
int time;
const double LEAF_ANGLE = 110;
const double PHYLLOTAXIS_ANGLE = 137.51;


// axiom (initial structure)
protected void init() [

    { time = 0; }

    Axiom ==>
        Bud
    ;
]
```

```
public void run() [

    // rewriting rule
    b:Bud ==>
        if (time < 5) (
            Internode
            [RL(LEAF_ANGLE) Leaf(0.1, 0.1)]
            RH(PHYLLOTAXIS_ANGLE) b
        ) else (
            Internode Flower
        )
    ;

    // execution rule
    l:Leaf ::> {
        l[length] :+= 0.01;
        l[width] :+= 0.01;
    }

    { time :+= 1; }
]
```

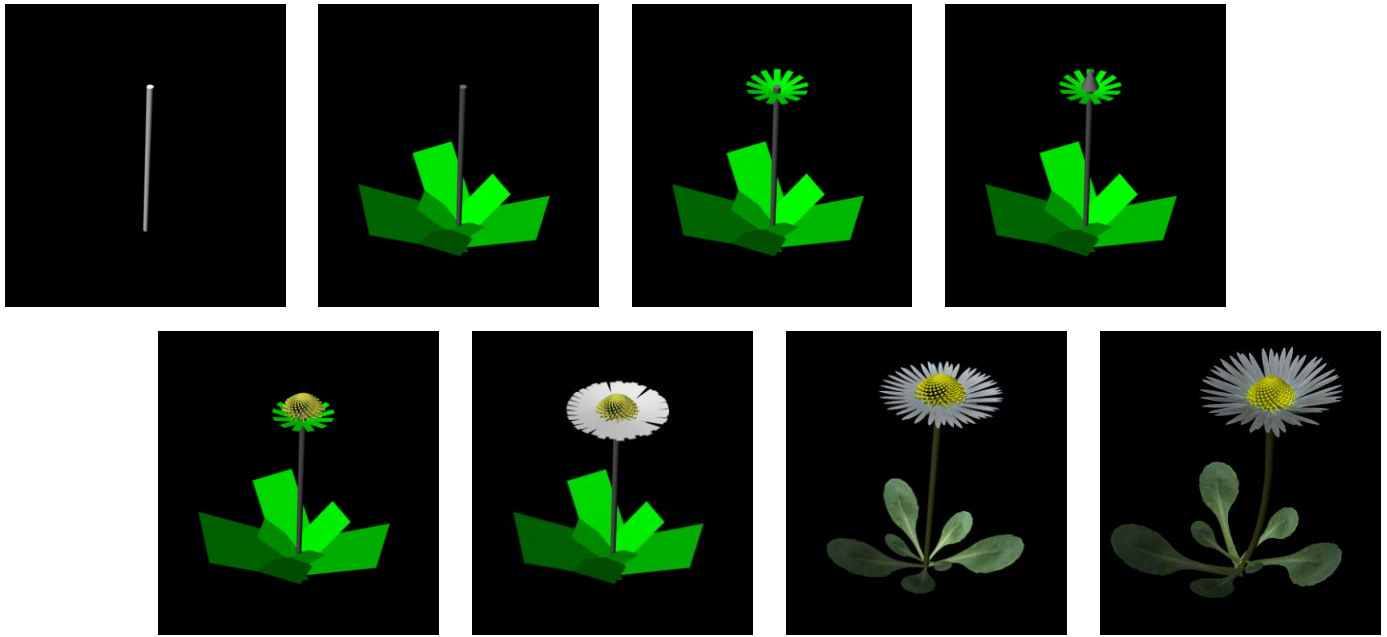
WAGENINGEN
UNIVERSITY & RESEARCH

Model example: Tomato

- Plant architecture with simplified development
- Leaf reconstruction options

Model example: Daisy

- Plant architecture reconstruction



Texture mapping

Menu: Panels > Explorers > 3D > Shaders

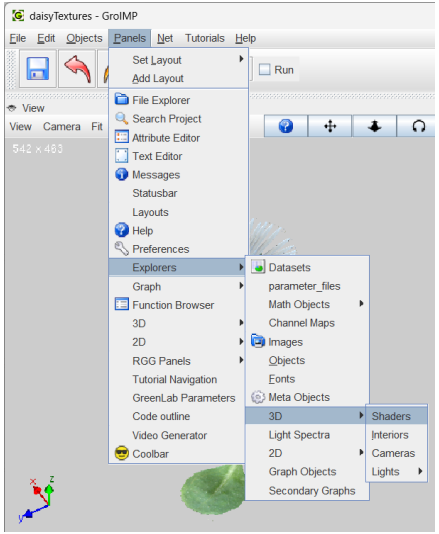
Shaders: Object > New > Lambert (2 clicks or F2 to rename)

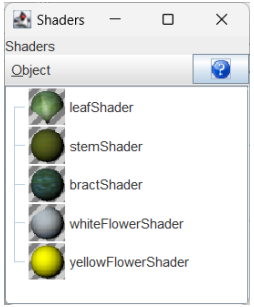
Attribute Editor: Diffuse colour > Surface Maps > Image; Image > From File

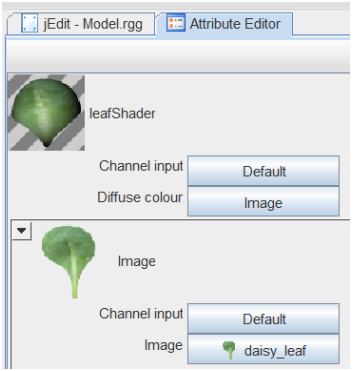
Photographs with removed background

(with alpha channel, png file format)

gimp.org







Create a reference in XL:

```
const ShaderRef leafSh =
  shader("leafShader");
```

Try yourself

- Materials:

  wiki.grogra.de > Workshops > FSPM and GroIMP tutorials at PMA conference 2025

  https://wiki.grogra.de/doku.php?id=workshops:tutorial_pma_25:01_saturday

- Simple plant architecture (from this presentation)

  -> add branching

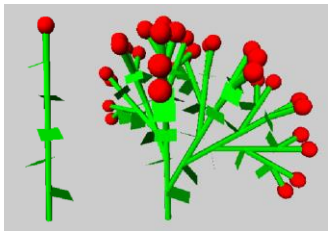
- Simple tomato plant

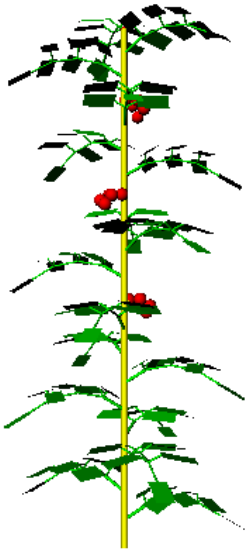
  -> added branching & different leaf shapes

  -> modify to another plant (change internode length, #leaflets, etc.)

- Daisy

  -> modify parameters/textures to create a different flower/plant





wiki.grogra.de

fspm.discourse.group