Introduction to GroIMP Modelling Platform

K. Smoleňová[1], G. Buck-Sorlin[2], R. Hemmerling[1], W. Kurth[1]

[1]Georg-August University of Göttingen, Germany

[2]Wageningen UR, The Netherlands

August 25, 2010

IHC Lisboa 2010

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Virtual Plant Network WAGENINGEN UR

Outline

Outline

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
Virtual Plant Network WAGENINGEN UR

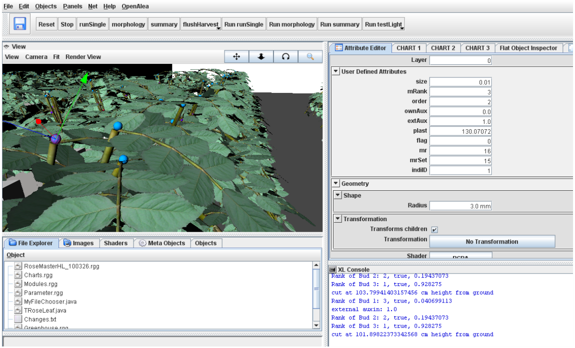
GroIMP (Open-source)

- **Gro**wth-grammar related **I**nteractive **M**odelling **P**latform
- Editable GUI, possible configuration:

Menu
Methods

3D View

File Explorer
Shaders
. . .



Attribute Editor
Graph
Charts
. . .

XL Console

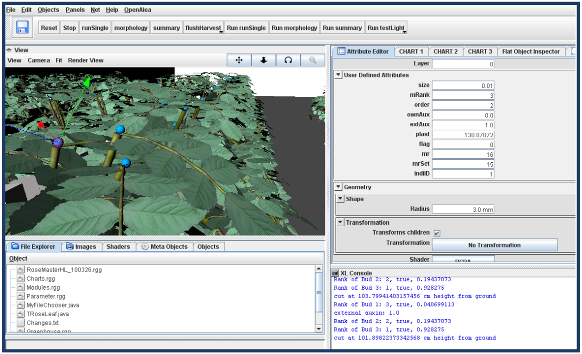
GroIMP (Open-source)

- **Gro**wth-grammar related **I**nteractive **M**odelling **P**latform
- Editable GUI, possible configuration:

Menu
Methods

3D View

File Explorer
Shaders
. . .



Attribute Editor
Graph
Charts
. . .

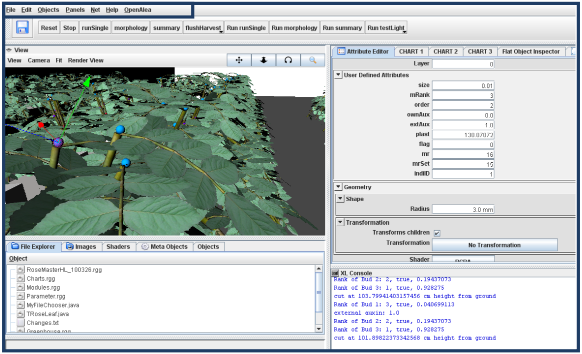
XL Console

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
Virtual Plant Network WAGENINGEN UR

GroIMP (Open-source)

- **Gro**wth-grammar related **I**nteractive **M**odelling **P**latform
- Editable GUI, possible configuration:

Menu
Methods
3D View
File Explorer
Shaders
. . .



Attribute Editor
Graph
Charts
. . .

XL Console

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
Virtual Plant Network WAGENINGEN UR

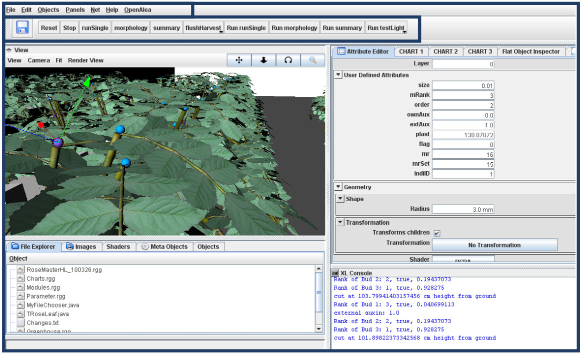
GroIMP (Open-source)

- **Gro**wth-grammar related **I**nteractive **M**odelling **P**latform
- Editable GUI, possible configuration:

Menu
Methods

3D View

File Explorer
Shaders
. . .



Attribute Editor
Graph
Charts
. . .

XL Console

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
Virtual Plant Network WAGENINGEN UR

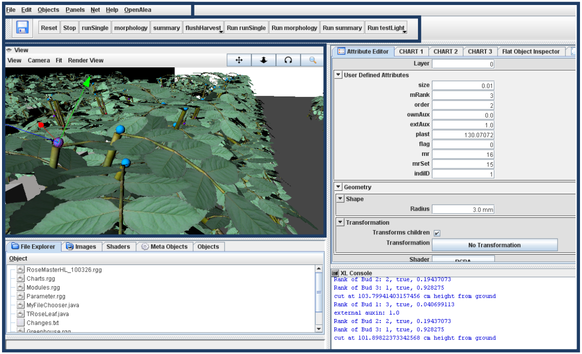
GroIMP (Open-source)

- **Gro**wth-grammar related **I**nteractive **M**odelling **P**latform
- Editable GUI, possible configuration:

Menu
Methods

3D View

File Explorer
Shaders
...



Attribute Editor
Graph
Charts
...

XL Console

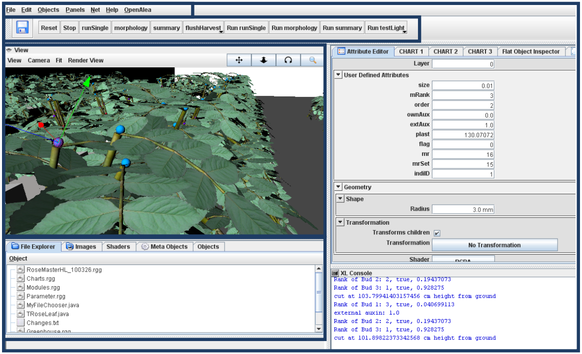
GroIMP (Open-source)

- **Gro**wth-grammar related **I**nteractive **M**odelling **P**latform
- Editable GUI, possible configuration:

Menu
Methods

3D View

File Explorer
Shaders
. . .



Attribute Editor
Graph
Charts
. . .

XL Console

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
Virtual Plant Network WAGENINGEN UR

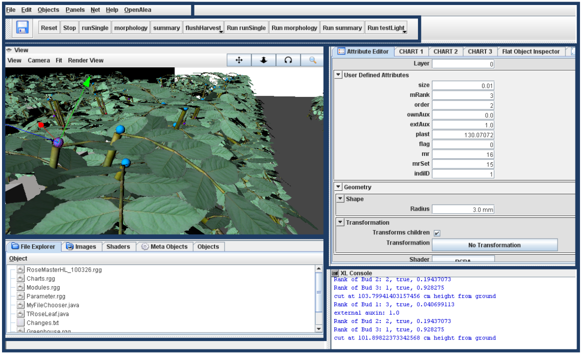
GroIMP (Open-source)

- **Gro**wth-grammar related **I**nteractive **M**odelling **P**latform
- Editable GUI, possible configuration:

Menu
Methods

3D View

File Explorer
Shaders
. . .



Attribute Editor
Graph
Charts
. . .

XL Console

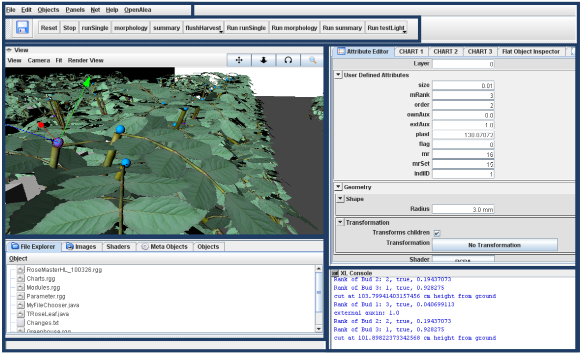
GroIMP (Open-source)

- **Gro**wth-grammar related **I**nteractive **M**odelling **P**latform
- Editable GUI, possible configuration:

Menu
Methods

3D View

File Explorer
Shaders
. . .



Attribute Editor
Graph
Charts
. . .

XL Console

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
Virtual Plant Network WAGENINGEN UR

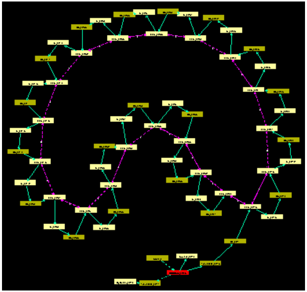
Outline

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Virtual Plant Network WAGENINGEN UR

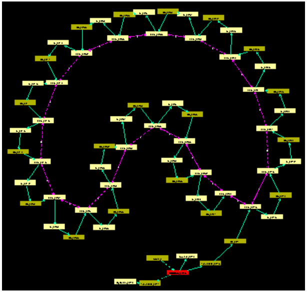
RGG

- **R**elational **G**rowth **G**rammars
- Graph structure rewriting formalism
- L-systems included as subset (parallel rewriting of strings)
- Plant structure and development described by RGG
  - Plant as an assemblage of organs or modules (nodes) which are connected (by edges)
  - Rules describe how the structure develops



GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
Virtual Plant Network WAGENINGEN UR

RGG

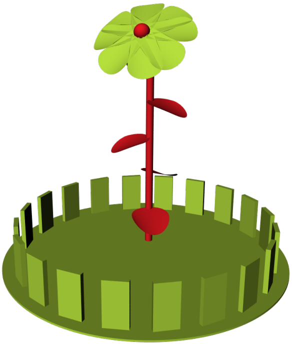


- **R**elational **G**rowth **G**rammars
- Graph structure rewriting formalism
- L-systems included as subset (parallel rewriting of strings)
- Plant structure and development described by RGG
  - Plant as an assemblage of organs or modules (**nodes**) which are connected (by **edges**)
  - Rules describe how the structure develops

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
Virtual Plant Network WAGENINGEN UR

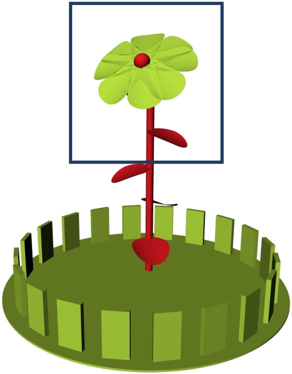
Graph Structure - Example

- Node
- Edge
  - Successor
  - Branch
  - User-defined



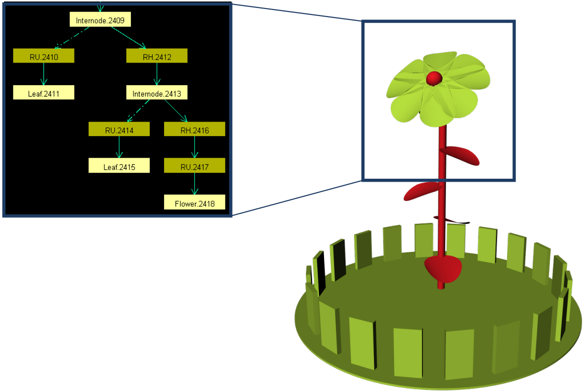
Graph Structure - Example

- Node
- Edge
  - Successor
  - Branch
  - User-defined



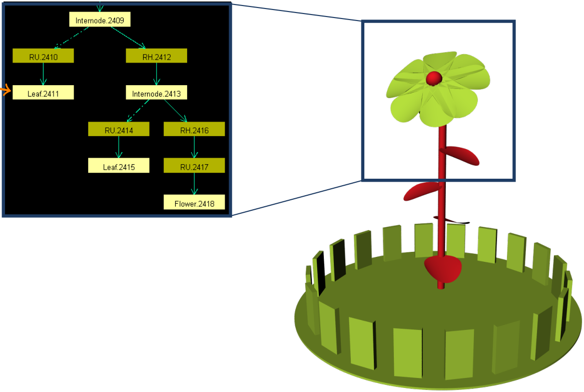
Graph Structure - Example

- Node
- Edge
  - Successor
  - Branch
  - User-defined



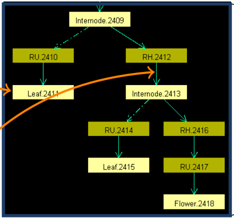
Graph Structure - Example

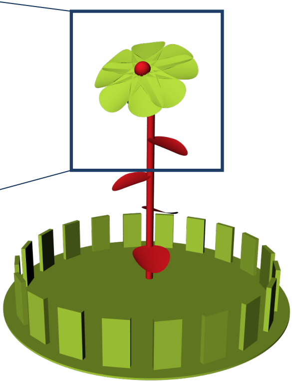
- Node
- Edge
  - Successor
  - Branch
  - User-defined



Graph Structure - Example

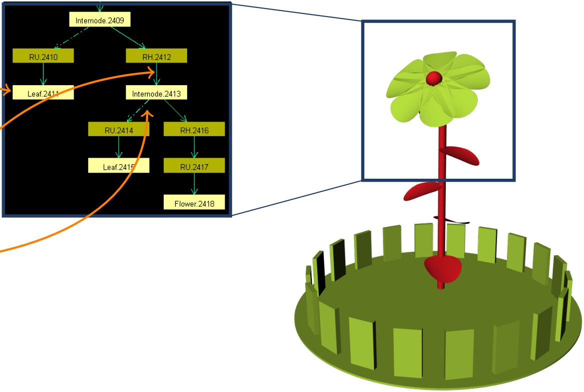
- Node
- Edge
  - Successor
  - Branch
  - User-defined





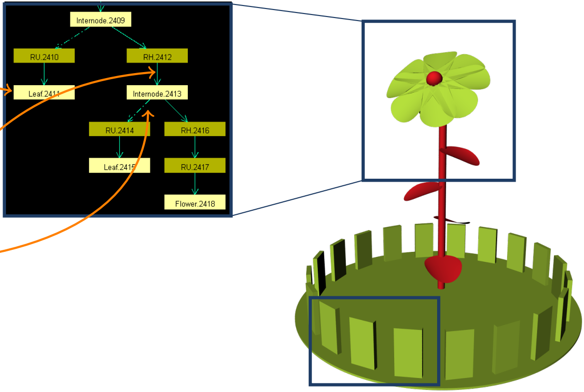
Graph Structure - Example

- Node
- Edge
  - Successor
  - Branch
  - User-defined



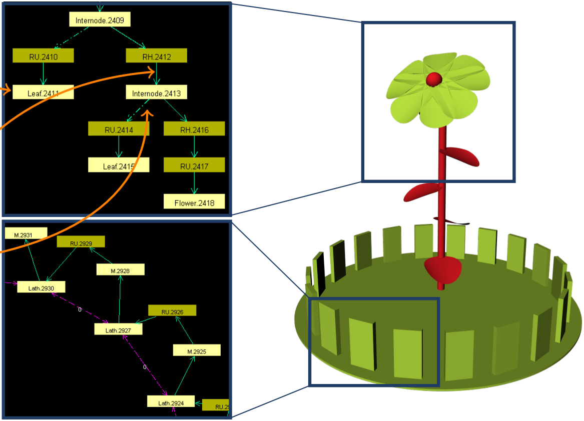
Graph Structure - Example

- Node
- Edge
  - Successor
  - Branch
  - User-defined



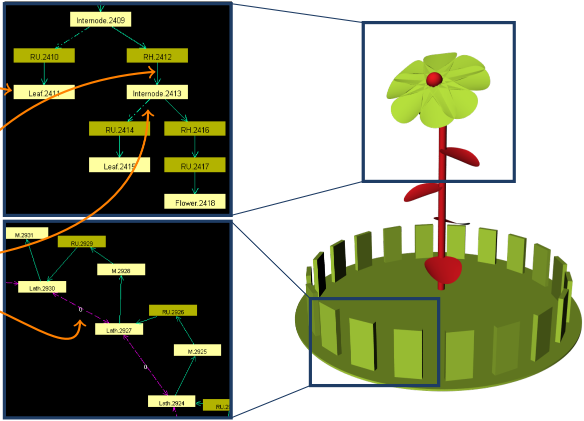
Graph Structure - Example

- Node
- Edge
  - Successor
  - Branch
  - User-defined



Graph Structure - Example

- Node
- Edge
  - Successor
  - Branch
  - User-defined



Outline

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
Virtual Plant Network WAGENINGEN UR

XL

- e**X**tended **L**-systems language
- Implementation of RGG formalism
- Based on Java (object-oriented)
- XL rules and Java code can be freely mixed and nested
  - [ ] rule block in XL
  - { } code block in Java
- Different types of rules
  - ==> L-system rule
  - ==>> general graph rewriting rule
  - ::> application rule (only parameters are changed)

XL

- e**X**tended **L**-systems language
- Implementation of RGG formalism
- Based on Java (object-oriented)
- XL rules and Java code can be freely mixed and nested
  - [ ] rule block in XL
  - { } code block in Java
- Different types of rules
  - ==> L-system rule
  - ==>> general graph rewriting rule
  - ::> application rule (only parameters are changed)

XL

- e**X**tended **L**-systems language
- Implementation of RGG formalism
- Based on Java (object-oriented)
- XL rules and Java code can be freely mixed and nested
  - `[ ]` rule block in XL
  - `{ }` code block in Java
- Different types of rules
  - `==>` L-system rule
  - `==>>` general graph rewriting rule
  - `::>` application rule (only parameters are changed)

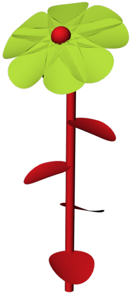
Outline

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
Virtual Plant Network WAGENINGEN UR

How to Model a Structure with XL Rules?

- Declaration of modules (plants as modular structures)

```
module Meristem;
module Internode;
module Leaf;
module Flower;
```



- Initial structure, initial conditions

- Methods, rules

How to Model a Structure with XL Rules?

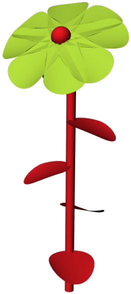
- Declaration of modules (plants as modular structures)

```
module Meristem;
module Internode;
module Leaf;
module Flower;
```

- Initial structure, initial conditions

```
protected void init()
[
   Axiom ==> Meristem;
]
```

- Methods, rules



How to Model a Structure with XL Rules?

- Declaration of modules (plants as modular structures)

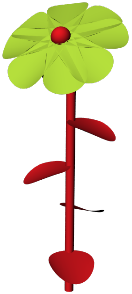
```
module Meristem;
module Internode;
module Leaf;
module Flower;
```

- Initial structure, initial conditions

```
protected void init()
[
   Axiom ==> Meristem;
]
```

- Methods, rules

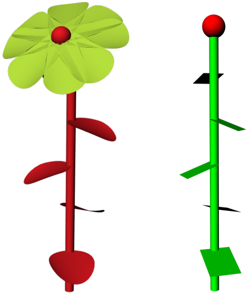
```
public void grow()
[
   m:Meristem ==> ...;
]
```



How to Model a Structure with XL Rules?

- Declaration of modules (plants as modular structures)

```
module Meristem;
module Internode;
module Leaf;
module Flower;
```



- Initial structure, initial conditions

```
protected void init()
[
   Axiom ==> Meristem;
]
```

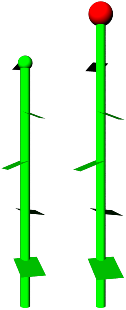
- Methods, rules

```
public void grow()
[
   m:Meristem ==> ...;
]
```

Declaration of Modules

- Adding graphical interpretation, setting the color

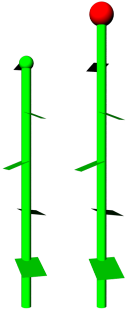
```
module Meristem;

module Internode;

module Leaf;

module Flower;
```



Declaration of Modules

- Adding graphical interpretation, setting the color

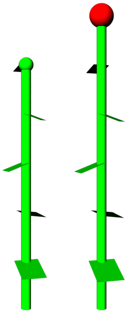
```
module Meristem extends Sphere(0.02);

module Internode extends Cylinder(0.2, 0.02);

module Leaf extends Parallelogram(0.05, 0.05);

module Flower extends Sphere(0.05);
```



Declaration of Modules

- Adding graphical interpretation, setting the color

```
module Meristem extends Sphere(0.02)
{
    {setShader(GREEN);}
}
module Internode extends Cylinder(0.2, 0.02)
{
    {setShader(GREEN);}
}
module Leaf extends Parallelogram(0.05, 0.05)
{
    {setShader(GREEN);}
}
module Flower extends Sphere(0.05)
{
    {setShader(RED);}
}
```



Setting the Rules

- Axiom, production rules (==>), global variables (time)

RH(137.51)

RL(110) RH(137.51) RL(110)

```
protected void init()
[

   Axiom ==> Meristem;
]
public void grow()
[
   m:Meristem ==>







   ;
]
```

Setting the Rules

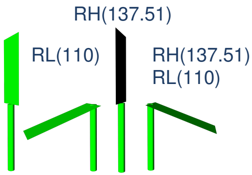
- Axiom, production rules (==>), global variables (time)


```
protected void init()
[

   Axiom ==> Meristem;
]
public void grow()
[
   m:Meristem ==>

         Internode [ RL(110) Leaf ]
         RH(137.51) m



   ;
]
```

Setting the Rules

- Axiom, production rules (==>), global variables (time)

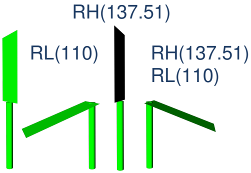
```
protected void init()
[

   Axiom ==> Meristem;
]
public void grow()
[
   m:Meristem ==>

         Internode [ RL(110) Leaf ]
         RH(137.51) m



   ;
]
```



Setting the Rules

- Axiom, production rules (==>), global variables (time)

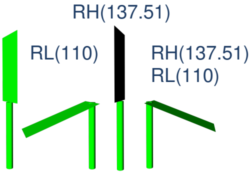
```
protected void init()
[

   Axiom ==> Meristem;
]
public void grow()
[
   m:Meristem ==>

         Internode [ RL(110) Leaf ]
         RH(137.51) m


   ;
]
```

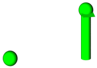


Setting the Rules

- Axiom, production rules (==>), global variables (time)

```
protected void init()
[
    Axiom ==> Meristem;
]
public void grow()
[
    m:Meristem ==>

        Internode [ RL(110) Leaf ]
        RH(137.51) m



    ;
]
```

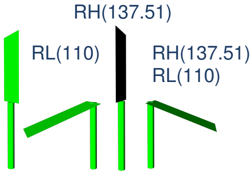


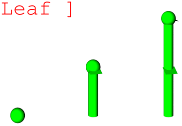


Setting the Rules

- Axiom, production rules (==>), global variables (time)

```
protected void init()
[
   Axiom ==> Meristem;
]
public void grow()
[
   m:Meristem ==>
         Internode [ RL(110) Leaf ]
         RH(137.51) m
   ;
]
```

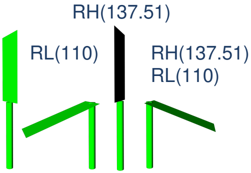


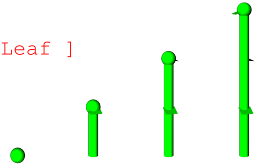


Setting the Rules

- Axiom, production rules (==>), global variables (time)

```
protected void init()
[

   Axiom ==> Meristem;
]
public void grow()
[
   m:Meristem ==>

        Internode [ RL(110) Leaf ]
        RH(137.51) m



   ;
]
```

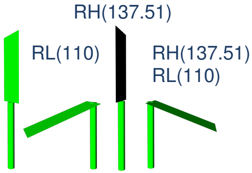


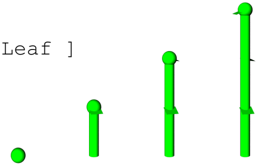


Setting the Rules

- Axiom, production rules (==>), global variables (time)

```
int time;
protected void init()
[
   { time = 0; }
   Axiom ==> Meristem;
]
public void grow()
[
   m:Meristem ==>
      if (time < 5) (
         Internode [ RL(110) Leaf ]
         RH(137.51) m
      ) else (

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



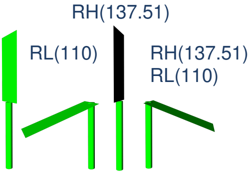


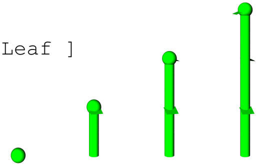
GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
Virtual Plant Network WAGENINGEN UR

Setting the Rules

- Axiom, production rules (==>), global variables (time)

```
int time;
protected void init()
[
   { time = 0; }
   Axiom ==> Meristem;
]
public void grow()
[
   m:Meristem ==>
      if (time < 5) (
         Internode [ RL(110) Leaf ]
         RH(137.51) m
      ) else (
         Internode Flower
      )
   ; { time :+= 1; }
]
```

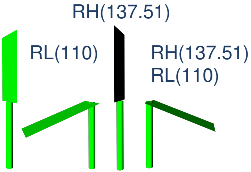


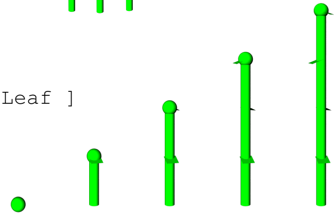


Setting the Rules

- Axiom, production rules (==>), global variables (time)

```
int time;
protected void init()
[
   { time = 0; }
   Axiom ==> Meristem;
]
public void grow()
[
   m:Meristem ==>
      if (time < 5) (
         Internode [ RL(110) Leaf ]
         RH(137.51) m
      ) else (
         Internode Flower
      )
   ; { time :+= 1; }
]
```

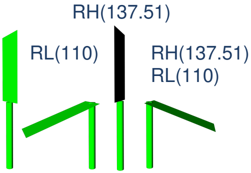


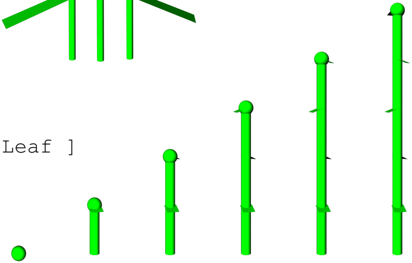


Setting the Rules

- Axiom, production rules (==>), global variables (time)

```
int time;
protected void init()
[
   { time = 0; }
   Axiom ==> Meristem;
]
public void grow()
[
   m:Meristem ==>
      if (time < 5) (
         Internode [ RL(110) Leaf ]
         RH(137.51) m
      ) else (
         Internode Flower
      )
   ; { time :+= 1; }
]
```

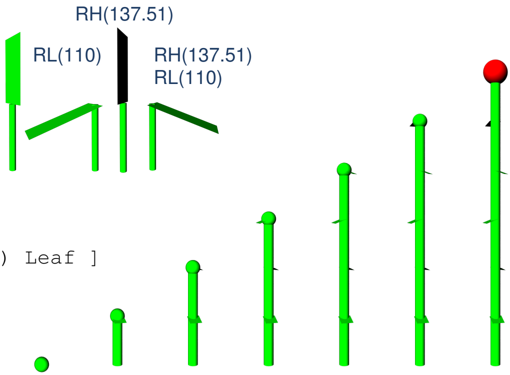




Setting the Rules

- Axiom, production rules (==>), global variables (time)

```
int time;
protected void init()
[
  { time = 0; }
  Axiom ==> Meristem;
]
public void grow()
[
  m:Meristem ==>
    if (time < 5) (
      Internode [ RL(110) Leaf ]
      RH(137.51) m
    ) else (
      Internode Flower
    )
  ; { time :+= 1; }
]
```



Setting the Rules (2)

- Application rules (::>), general rewriting rules (==>>)

```
public void grow()
[
    ...



]
```

→

Setting the Rules (2)

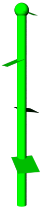
- Application rules (::>), general rewriting rules (==>>)

```
public void grow()
[
    ...
    leaf:Leaf ::> {
        leaf[length] :+= 0.015;
        leaf[axis][x] :+= 0.005;
    }
]
```

Setting the Rules (2)

- Application rules (::>), general rewriting rules (==>>)

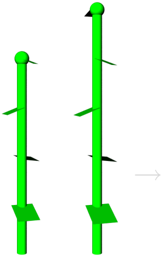
```
public void grow()
[
   ...
   leaf:Leaf ::> {
      leaf[length] :+= 0.015;
      leaf[axis][x] :+= 0.005;
   }
]
```



Setting the Rules (2)

- Application rules (::>), general rewriting rules (==>>)

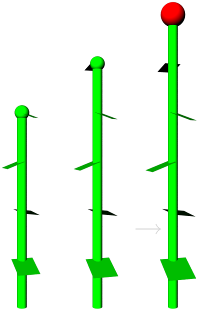
```
public void grow()
[
    ...
    leaf:Leaf ::> {
        leaf[length] :+= 0.015;
        leaf[axis][x] :+= 0.005;
    }
]
```



Setting the Rules (2)

- Application rules (::>), general rewriting rules (==>>)

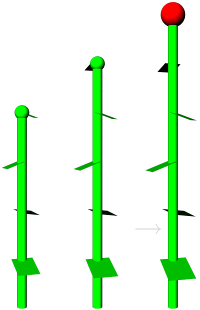
```
public void grow()
[
    ...
    leaf:Leaf ::> {
        leaf[length] :+= 0.015;
        leaf[axis][x] :+= 0.005;
    }
]
```



Setting the Rules (2)

- Application rules (::>), general rewriting rules (==>>)

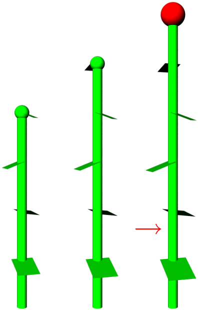
```
public void grow()
[
   ...
   leaf:Leaf ::> {
      leaf[length] :+= 0.015;
      leaf[axis][x] :+= 0.005;
   }
]

public void harvest()
[
   i:Internode, (isSelected(i)) ==>> ;
]
```



Setting the Rules (2)

- Application rules (::>), general rewriting rules (==>>)

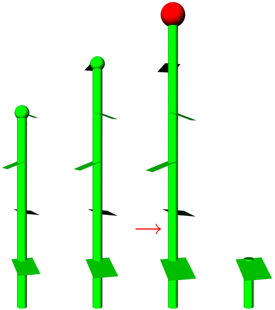
```
public void grow()
[
    ...
    leaf:Leaf ::> {
        leaf[length] :+= 0.015;
        leaf[axis][x] :+= 0.005;
    }
]

public void harvest()
[
    i:Internode, (isSelected(i)) ==>> ;
]
```



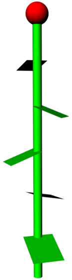
Setting the Rules (2)

- Application rules (::>), general rewriting rules (==>>)

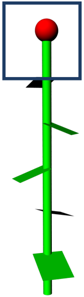
```
public void grow()
[
   ...
   leaf:Leaf ::> {
      leaf[length] :+= 0.015;
      leaf[axis][x] :+= 0.005;
   }
]

public void harvest()
[
   i:Internode, (isSelected(i)) ==>> ;
]
```

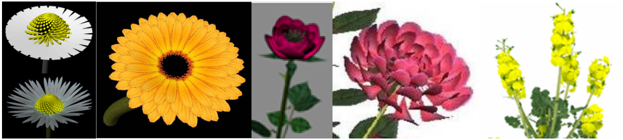


Improving the Graphical Interpretation

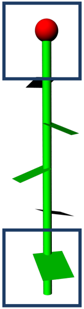


Improving the Graphical Interpretation

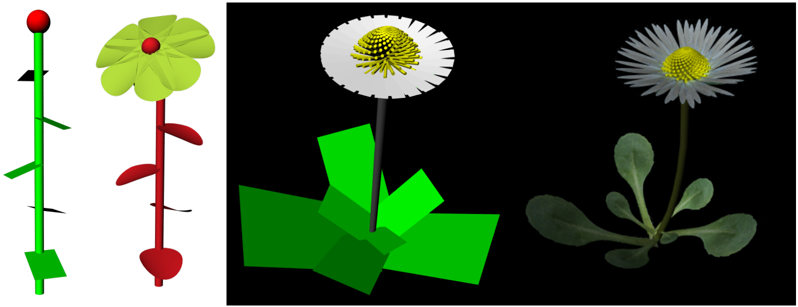




Improving the Graphical Interpretation



Improving the Graphical Interpretation (2)



Outline

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN Virtual Plant Network WAGENINGEN UR

How to Model Light Distribution and Interception?

- Using GroIMP's radiation model
  - Based on Kajiya's path tracing technique
- Workflow
  - Emit light from light sources
  - Absorb light (partially) when it hits an object
  - Reflect / transmit unabsorbed light
- Input data
  - Objects with assigned materials
  - Lights (point, spot, directional, area light, etc.) having specific properties
- Output data
  - Amount of accepted light by an object in *W*

How to Model Light Distribution and Interception?

- Using GroIMP's radiation model
  - Based on Kajiya's path tracing technique
- Workflow
  - Emit light from light sources
  - Absorb light (partially) when it hits an object
  - Reflect / transmit unabsorbed light
- Input data
  - Objects with assigned materials
  - Lights (point, spot, directional, area light, etc.) having specific properties
- Output data
  - Amount of accepted light by an object in *W*

How to Model Light Distribution and Interception?

- Using GroIMP's radiation model
  - Based on Kajiya's path tracing technique
- Workflow
  - Emit light from light sources
  - Absorb light (partially) when it hits an object
  - Reflect / transmit unabsorbed light
- Input data
  - Objects with assigned materials
  - Lights (point, spot, directional, area light, etc.) having specific properties
- Output data
  - Amount of accepted light by an object in W

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
Virtual Plant Network WAGENINGEN UR

How to Model Light Distribution and Interception?

- Using GroIMP's radiation model
  - Based on Kajiya's path tracing technique
- Workflow
  - Emit light from light sources
  - Absorb light (partially) when it hits an object
  - Reflect / transmit unabsorbed light
- Input data
  - Objects with assigned materials
  - Lights (point, spot, directional, area light, etc.) having specific properties
- Output data
  - Amount of accepted light by an object in *W*

How to Model Light Distribution and Interception? (2)

```
public void grow()
[



    ...
]
```

How to Model Light Distribution and Interception? (2)

```
const int LM_RAYS = 1000000; const int LM_DEPTH = 10;

LightModel lm = new LightModel(LM_RAYS, LM_DEPTH);

public void grow()
[
    { lm.compute(); }

    leaf:Leaf ::> {
        float radiation =
            lm.getAbsorbedPower(leaf).integrate();
    }
    ...
]
```

How to Compute Photosynthesis per Leaf?

- Integration of a photosynthetic model, e.g. Lieth and Pasian (1990) → amount of assimilates per leaf per step

```
module Leaf          extends Parallelogram(0.05, 0.05)
{
   {setShader(GREEN);}
}

public void grow()
[  ...
   leaf:Leaf ::> {
      float radiation =
         lm.getAbsorbedPower(leaf).integrate();

   } ...
]
```

- Next step: transport of assimilates …

How to Compute Photosynthesis per Leaf?

- Integration of a photosynthetic model, e.g. Lieth and Pasian (1990) → amount of assimilates per leaf per step

```
module Leaf            extends Parallelogram(0.05, 0.05)
{
   {setShader(GREEN);}
}

public void grow()
[  ...
   leaf:Leaf ::> {
      float radiation =
         lm.getAbsorbedPower(leaf).integrate();

   } ...
]
```

- Next step: transport of assimilates ...

How to Compute Photosynthesis per Leaf?

- Integration of a photosynthetic model, e.g. Lieth and Pasian (1990) → amount of assimilates per leaf per step

```
module Leaf(float as) extends Parallelogram(0.05, 0.05)
{
   {setShader(GREEN);}
}
const float LEAF_TEMP = 25;
public void grow()
[  ...
   leaf:Leaf ::> {
      float radiation =
         lm.getAbsorbedPower(leaf).integrate();
      leaf[as] :+= PS(LEAF_TEMP, radiation);
   } ...
]
double PS(float Temp, float Rad) {...}
```

- Next step: transport of assimilates ...

How to Compute Photosynthesis per Leaf?

- Integration of a photosynthetic model, e.g. Lieth and Pasian (1990) → amount of assimilates per leaf per step

```
module Leaf(float as) extends Parallelogram(0.05, 0.05)
{
   {setShader(GREEN);}
}
const float LEAF_TEMP = 25;
public void grow()
[  ...
   leaf:Leaf ::> {
      float radiation =
         lm.getAbsorbedPower(leaf).integrate();
      leaf[as] :+= PS(LEAF_TEMP, radiation);
   } ...
]
double PS(float Temp, float Rad) {...}
```

- Next step: transport of assimilates ...

Outline

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

Virtual Plant Network WAGENINGEN UR

Complex Structure Development

- Modules defined for different scales
  - Canopy
  - Individual
  - BentCanopy
  - Shoot (HarvestableShoot, InstantUShoot)
  - Organ (Root, Bud, Leaf, Internode, Flower)

Complex Structure Development (2)

- For graphical interpretation, modules extend
  - Cylinder (Internode)
  - Parallelogram (Leaflet)
  - NURBSSurface (Flower), etc.
- Rules based on observations of rose development
- Parameters obtained from experiments

Complex Structure Development (2)

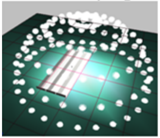
- For graphical interpretation, modules extend
  - Cylinder (Internode)
  - Parallelogram (Leaflet)
  - NURBSSurface (Flower), etc.
- Rules based on observations of rose development
- Parameters obtained from experiments

Simulation of Light Distribution

- Light distribution depends on the scene, position of objects in it and their materials
  $\longrightarrow$ virtual greenhouse
- Simulation of
  - Diffuse light (sky)
  - Direct light (sun)
  - Assimilation lamps (SON-T, LED)

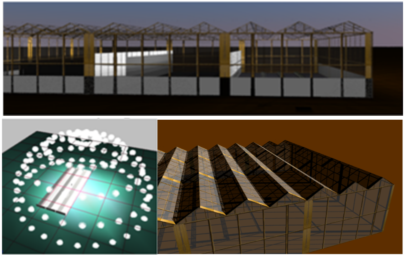
Simulation of Light Distribution

- Light distribution depends on the scene, position of objects in it and their materials
  $\longrightarrow$ virtual greenhouse
- Simulation of
  - Diffuse light (sky)
  - Direct light (sun)
  - Assimilation lamps (SON-T, LED)



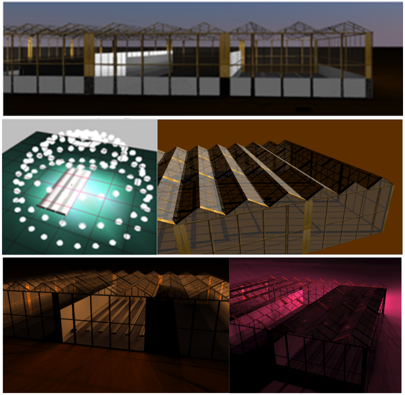
Simulation of Light Distribution

- Light distribution depends on the scene, position of objects in it and their materials $\longrightarrow$ virtual greenhouse
- Simulation of
  - Diffuse light (sky)
  - Direct light (sun)
  - Assimilation lamps (SON-T, LED)

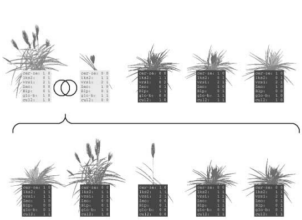


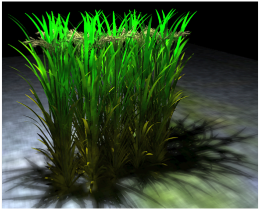
Simulation of Light Distribution

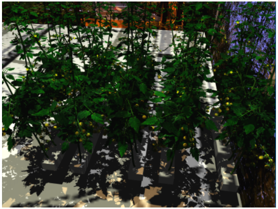
- Light distribution depends on the scene, position of objects in it and their materials
  → virtual greenhouse
- Simulation of
  - Diffuse light (sky)
  - Direct light (sun)
  - Assimilation lamps (SON-T, LED)



More examples of FSPMs







- Barley (Buck-Sorlin *et al.*)
- Rice (Xu *et al.*)
- Rapeseed (Groer *et al.*, Henke *et al.*)
- Arabidopsis (Evers *et al.*)
- Tomato (Buck-Sorlin *et al.*)
- Beech, Spruce (Hemmerling *et al.*, Kurth *et al.*)
- ...

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
Virtual Plant Network WAGENINGEN UR

Conclusions and Future Work

- GroIMP as a general platform for FSPM
  - Decision-support tool
  - Educational tool
  - Experimental research
- Advanced features
  - Searching in a graph structure
  - Queries
  - ODE solver
- Future Work
  - Extension of radiation model
  - Temperature distribution in greenhouse

Conclusions and Future Work

- GroIMP as a general platform for FSPM
  - Decision-support tool
  - Educational tool
  - Experimental research
- Advanced features
  - Searching in a graph structure
  - Queries
  - ODE solver
- Future Work
  - Extension of radiation model
  - Temperature distribution in greenhouse

GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN
Virtual Plant Network WAGENINGEN UR

Conclusions and Future Work

- GroIMP as a general platform for FSPM
  - Decision-support tool
  - Educational tool
  - Experimental research
- Advanced features
  - Searching in a graph structure
  - Queries
  - ODE solver
- Future Work
  - Extension of radiation model
  - Temperature distribution in greenhouse

http://www.grogra.de

http://www.grogra.de

{ksmolen,rhemmer}@gwdg.de
gerhard.buck-sorlin@wur.nl
wk@informatik.uni-goettingen.de

Thank you for your attention.

http://www.grogra.de

{ksmolen,rhemmer}@gwdg.de
gerhard.buck-sorlin@wur.nl
wk@informatik.uni-goettingen.de