

GroIMP v1.4.2

Introduction and Overview

Michael Henke

Department Ecoinformatics, Biometrics and Forest Growth, University of Göttingen,
Germany

International Summer School
"Modelling of Ecosystems by Tools from Computer Science"

Prague, 16.09.2013



GroIMP

Overview I

Paradigm(s)	multi-paradigm: imperative, object-oriented, rule-based, chemical
Appeared in	2003
Designed by	Ole Kniemeyer
Stable release	1.4.2 (June 06, 2013)
Typing discipline	static, strong, safe
Influenced by	GROGRA
Operating System	Cross-platform, (multi-platform - JVM)
License	GNU General Public License V.3
Usual filename extensions	gsz, rgg, xl
Available at	sourceforge.net



GroIMP

Overview II

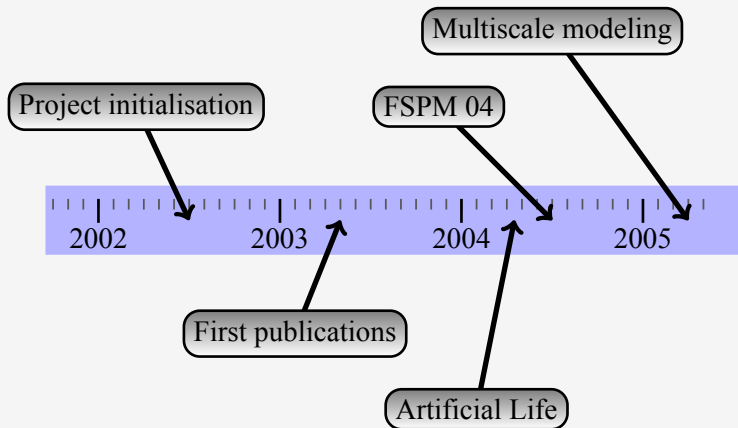
Growth-grammar related **I**nteractive **M**odelling **P**latform

- ▶ Objectives
 - ▶ 3D modelling software specialized for FSPMs
- ▶ Target groups
 - ▶ Students (e.g. biology, agronomy, horticulture)
 - ▶ Scientists
 - ▶ Modeller
- ▶ Application fields
 - ▶ Educational tool
 - ▶ (Experimental) Research & Science
 - ▶ Decision-support tool
 - ▶ Industry



GroIMP

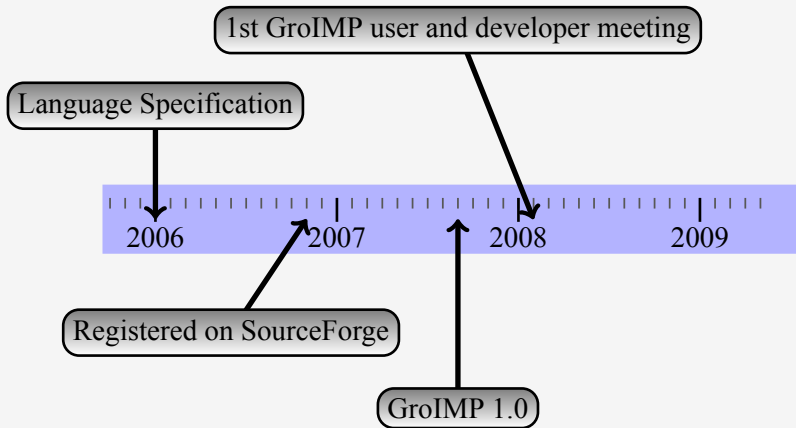
Timeline





GroIMP

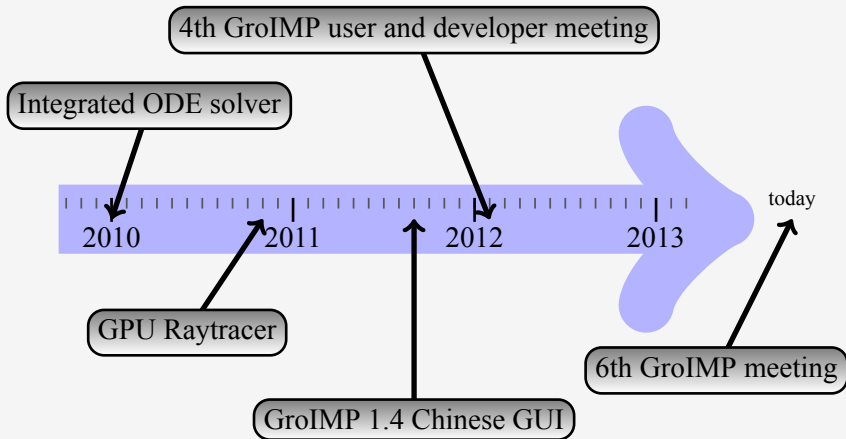
Timeline





GroIMP

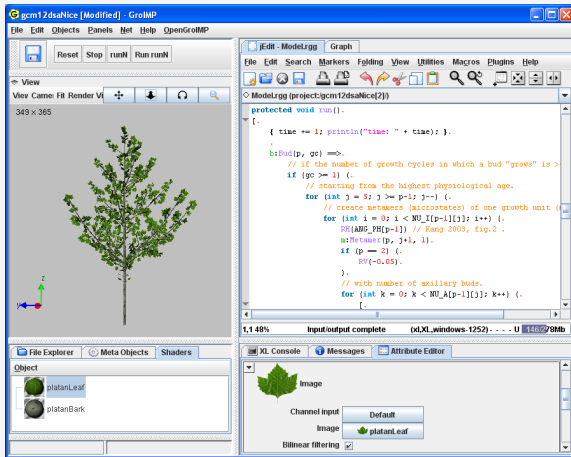
Timeline





GroIMP

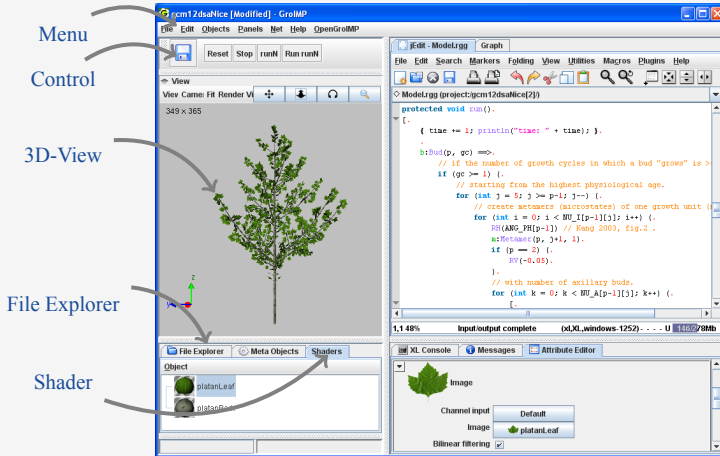
User interface





GroIMP

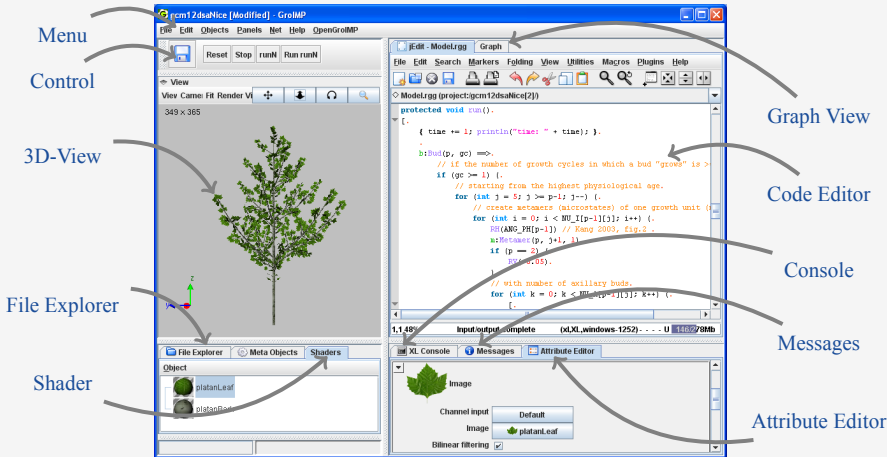
User interface





GroIMP

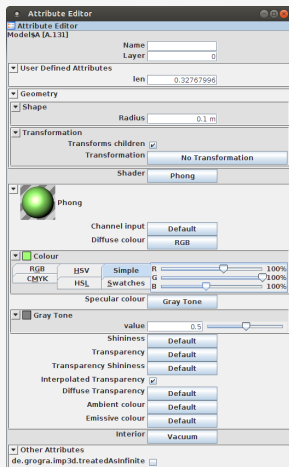
User interface



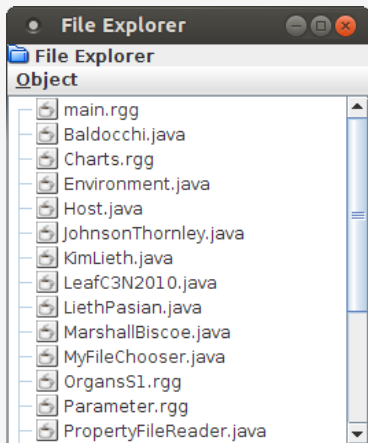


Attribute Editor

- ▶ The easy way to change attributes of an object.
- ▶ Attribute of a selected object are shown



File Explorer

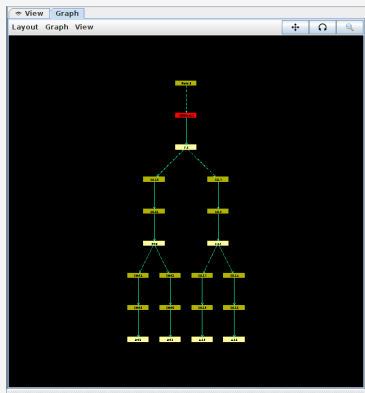


- ▶ Projects may contain several RGG, XL and Java files. In addition, JAR files can be included.
- ▶ Plain text files (e.g. parameter files)



2D Graph Panel

- ▶ Graph structure produced by your model
- ▶ Several layout algorithm



... Example, drawn in tree layout



XL Console

Command line with History function

```
de.grogra.turtle.F[id=157]|@680a6168
de.grogra.turtle.F[id=122]|@179ea7d1
de.grogra.turtle.F[id=129]|@1d24fcc4
de.grogra.turtle.F[id=115]|@606d5353
> (* F *).length
0.64
0.64
0.64
0.64
0.64
0.8
0.8
1.0
> count((* x:F, (x.length<=0.64) *))
4
> mean((* F *).length)
0.73714286
> sum((* F *).length)
5.16
> count((* Model.A *))
8
> 1+1
2
>
```

```
> (* F *)
> (* F *).length
> count((* x:F, (x.length<=0.64) *))
> mean((* F *).length)
> sum((* F *).length)
> count((* Model.A *))
> 1+1
```



XL Console

Command line with History function

```

Messages  XL Console
de.grogra.turtle.F[id=157]@680a6168
de.grogra.turtle.F[id=122]@179ea7d1
de.grogra.turtle.F[id=129]@1d24fcc4
de.grogra.turtle.F[id=115]@606d5353
> (* F *).length
0.64
0.64
0.64
0.64
0.8
0.8
1.0
> count((* x:F, (x.length<=0.64) *))
4
> mean((* F *).length)
0.73714286
> sum((* F *).length)
5.16
> count((* Model.A *))
8
> 1+1
2
>

```

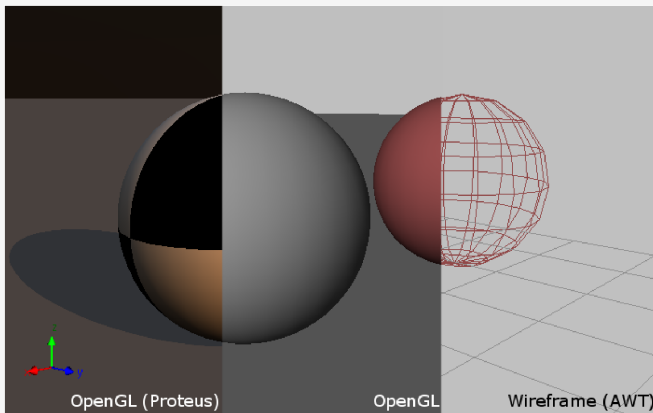
```

> (* F *)
> (* F *).length
> count((* x:F, (x.length<=0.64) *))
> mean((* F *).length)
> sum((* F *).length)
> count((* Model.A *))
> 1+1

```



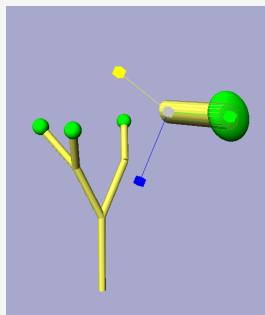
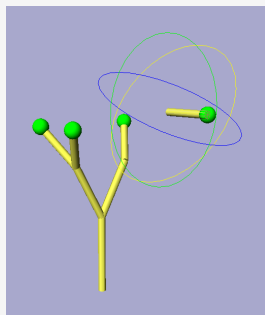
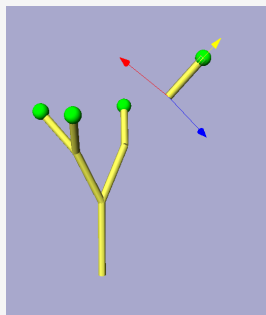

Live 3D-View



Comparison of the different 3D views in GroIMP. From left to right: advanced OpenGL view, normal OpenGL view, software renderer.



Interactive Live 3D-View

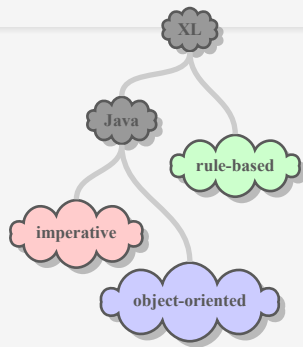


Objects can be interactively edited: moved, rotated, scaled
inserted and deleted



XL programming language

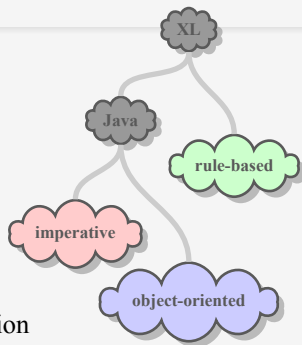
- ▶ eXtended L-system language
- ▶ Superset of Java, multiparadigm language





XL programming language

- ▶ e**X**tended **L**-system language
- ▶ Superset of Java, multiparadigm language
- ▶ Different types of rules
 - ▶ **L-system rule** (\Rightarrow)
A \Rightarrow A [B C];
 - ▶ **General graph rewriting rule** ($\Rightarrow\Rightarrow$)
more general rules for graph transformation
 - ▶ **Application rule** ($::\Rightarrow$)
only parameters are updated
(age, demand, biomass, length, etc.)





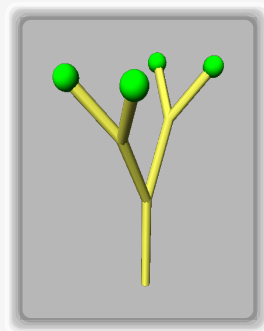
XL - Code example

```
1 import ...;
2
3 module A(float len) extends Sphere(0.1) {
4     {setShader(GREEN);}
5 }
6
7 protected void init () [
8     Axiom ==> A(1);
9 ]
10
11 public void run () [
12     A(x) ==> F(x)
13     [RU(30) RH(90) A(x*0.8)]
14     [RU(-30) RH(90) A(x*0.8)];
15 ]
```



XL - Code example

```
1 import ...;
2
3 module A(float len) extends Sphere(0.1) {
4     {setShader(GREEN);}
5 }
6
7 protected void init () [
8     Axiom ==> A(1);
9 ]
10
11 public void run () [
12     A(x) ==> F(x)
13     [RU(30) RH(90) A(x*0.8)]
14     [RU(-30) RH(90) A(x*0.8)];
15 ]
```





Integrated Query Language

- ▶ **Aggregate operations** to collect / return specific values
 - ▶ count, sum, min, max, mean, first, last, selectRandomly, exist, etc.
- ▶ **Queries** to analyse the plant (graph) structure
 - ▶ Search for a specific pattern
 - ▶ Can be combined with aggregators, e.g., search for all leaves and sum up their area

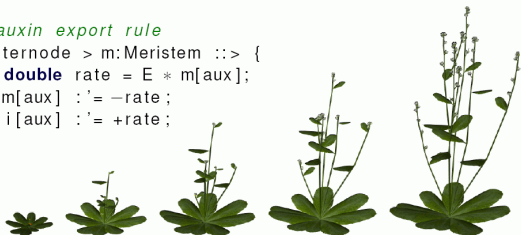
```
1 > (* Leaf *)  
2 > (* Leaf *).getArea()  
3 > sum(* Leaf *).getArea()
```

ODE System

ODE system for easy specification of ordinary differential equations (ODEs) within rules, and their (separated) solution

```
// cytokinin biosynthesis in root system
r:Roots ::> { r[cyt] := P - Q * r[aux]; }

// auxin export rule
i:Internode > m:Meristem ::> {
  double rate = E * m[aux];
  m[aux] := -rate;
  i[aux] := +rate;
}
```

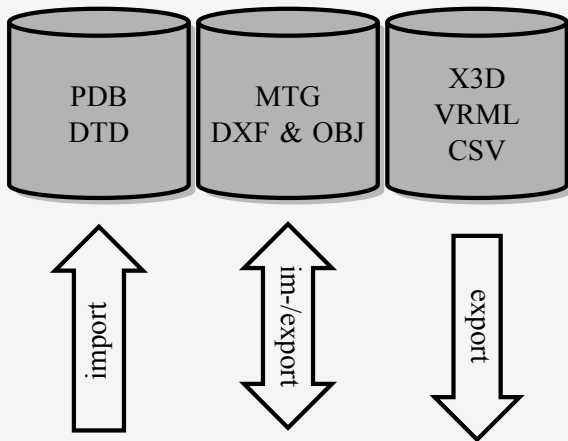


(Evers 2012)


⇒ Hemmerling 2012: [Extending the Programming Language XL to Combine Graph Structures with Ordinary Differential Equations.](#)

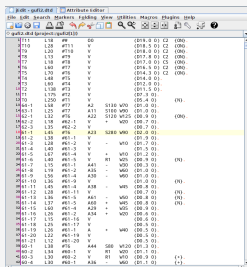


Import- and Export formats



Descriptive tree data - DTD Editor

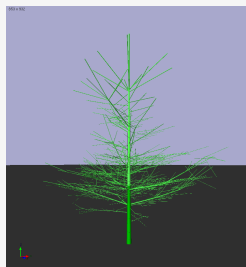
- ▶ Plain text file (Example file: )
- ▶ Every line represents a shoot
- ▶ Whole file represents a plant or a part of a plant
 - ▶ Used primarily for trees
 - ▶ General: All branching structures



```

#111 1.14 #1 50 (014.8 0) C2 (0R)
#119 1.28 #111 V (014.5 0) C2 (0R)
#120 1.20 #119 V (014.8 0) C2 (0R)
#128 1.13 #119 V (017.8 0) C2 (0R)
#127 1.18 #118 V (017.8 0) C2 (0R)
#126 1.80 #117 V (014.5 0) C2 (0R)
#125 1.20 #116 V (014.5 0) C2 (0R)
#124 1.80 #115 V (014.5 0)
#123 1.80 #114 V (014.5 0)
#122 1.138 #113 V (011.5 0)
#121 1.176 #112 V (011.5 0)
#120 1.250 #111 V (05.4 0)
#104-1 1.25 #111 A3 S110 W0 (01.0 0) (R)
#102-1 1.32 #116 A22 S110 W25 (08.0 0) (R)
#102-2 1.18 #111 A3 S110 W0 (01.0 0) (R)
#102-3 1.35 #12-2 V (08.7 0)
#101-1 1.80 #111 A33 S110 W0 (02.0 0)
#101-2 1.38 #11-1 V (01.0 0)
#101-3 1.28 #11-2 V (01.0 0)
#101-4 1.80 #11-5 V (01.5 0)
#101-5 1.87 #11-4 V (01.0 0)
#101-6 1.80 #11-5 V R1 W0 (08.0 0) (R)
#101-7 1.12 #11-1 A41 + W0 (08.0 0) (R)
#101-8 1.19 #11-2 A35 - W0 (01.0 0)
#101-9 1.26 #11-4 A30 + W0 (01.0 0) (R)
#101-10 1.36 #11-9 V - W0 (01.0 0) (R)
#101-11 1.80 #11-8 A38 + W0 (08.0 0) (R)
#101-12 1.28 #11-11 V - W0 (08.7 0) (R)
#101-13 1.36 #11-5 A41 + W0 (08.4 0) (R)
#101-14 1.37 #11-5 A48 + W0 (08.0 0) (R)
#101-15 1.80 #11-8 A38 + W0 (08.0 0) (R)
#101-16 1.26 #11-2 A34 + W0 (08.0 0) (R)
#101-17 1.15 #11-11 V (08.0 0) (R)
#101-18 1.26 #11-11 V (08.0 0) (R)
#101-19 1.20 #11-1 V (08.0 0) (R)
#101-20 1.22 #11-11 V (08.0 0) (R)
#101-21 1.12 #11-20 V (08.0 0) (R)
#101-22 1.38 #11-1 A44 S0 W20 (01.0 0) (R)
#101-23 1.84 #10-1 V R1 W0 (01.5 0) (R)
#101-24 1.30 #11-2 V R1 W0 (08.0 0) (R)
#101-25 1.80 #11-1 A36 + W0 (01.0 0) (R)

```



http://www.uni-forst.gwdg.de/~wkuurth/cb/html/workshop07/ws07_derer.pdf

http://www.uni-forst.gwdg.de/~wkuurth/dtd_code.pdf



Simple File export

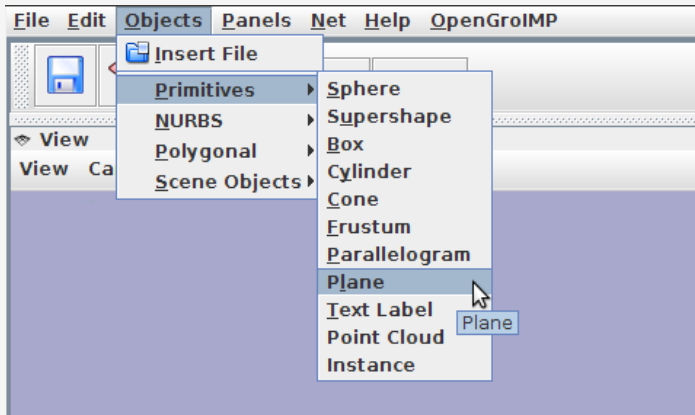
⇒ Example: *Gallery/Technics/print_in_file2.gsz*

```
1 import java.io.*;
2
3 PrintWriter tmpfile;
4
5 protected void init () {
6     // create the file
7     tmpfile = new PrintWriter(new FileWriter("/home/./out1.csv"));
8     [ Axiom ==> A(1); ]
9 }
10
11 public void run () [
12     a1:A(x) ==> F(x) [RU(30) RH(90) a2:A(x*0.8)] [RU(-30) RH(90) a3:A(x*0.8)]
13     {
14         tmpfile.println(" " + a1.len + " " + a2.len + " " + a3.len);
15         //write data to the file
16         tmpfile.flush();
17     };
18 ]
19
20 // close the file
21 public void end(){
22     tmpfile.close();
23 }
```



Objects

Available primitives objects, NURBS, Polygons and LightNodes





Tropism Nodes

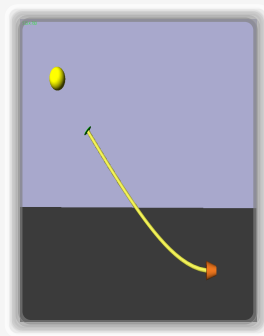
- ▶ growth in response to an environmental stimulus
- ▶ Commands: RD, RP, RN
- ▶ e.g. Phototropism: Shoots bends towards a light source

```

1 Sphere sun = new Sphere(0.25).setShader(YELLOW),
   setTransform(5,0,5);
2
3 ...
4
5 public void run () [
6     top_bud ==> RN(sun, 0.2) F(0.2) top_bud;
7 ]

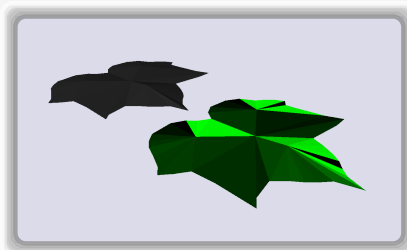
```

Point-Tropism(*point/direction, power*)



Heliotropism

Algorithm Switch Shader

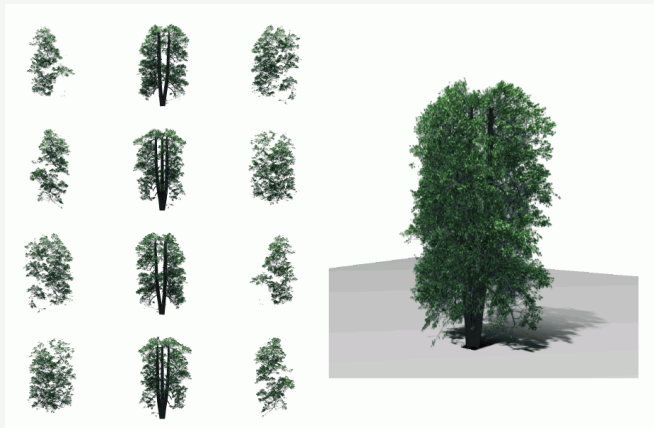


Allows us to define two different textures for an object. One is used by the light model (raytracer) and defines the optical properties of the object for light absorption, reflection etc., the second one is used for visualisation in the 3D view.

```
1 Shader leafTexture = shader("leaf");
2 Phong leafShader = new Phong().(
3     setDiffuse(new RGBColor(reflectance, reflectance, reflectance)),
4     setDiffuseTransparency(new RGBColor(transmittance, transmittance,
5     transmittance)));
6 ...
7 leaf.setShader(new AlgorithmSwitchShader(leafTexture , leafShader));
```

Billboard generator

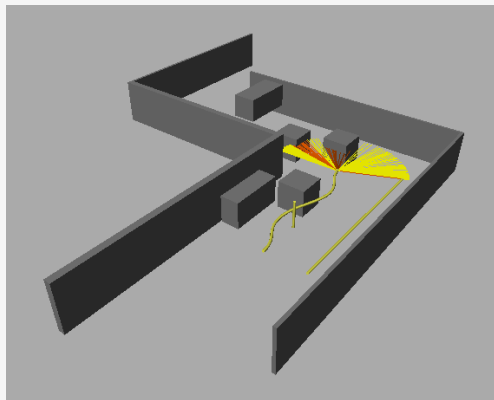
Integrated tool to generate billboards.



(Hemmerling 2010)

Intersection avoidance mechanism

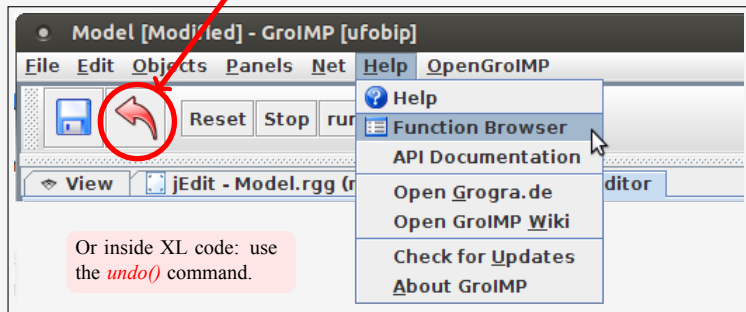
The class `AvoidIntersection` uses rays and intersection tests to obtain a new direction for growth.



⇒ Example: [Gallery/Technics/smart_line.gsz](#)

Undo

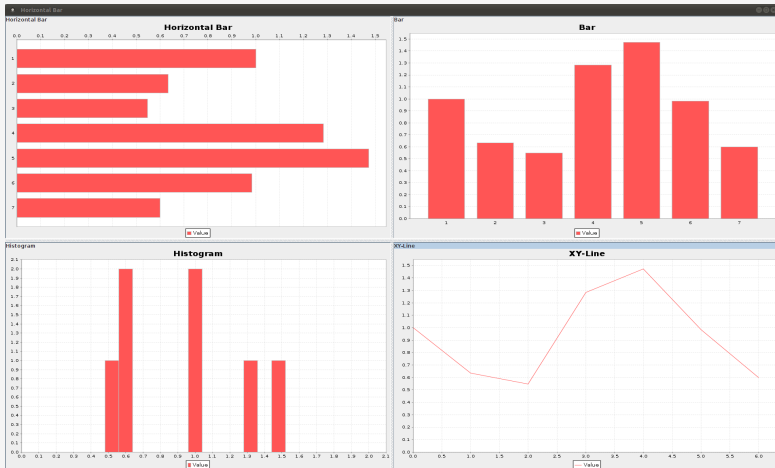
Undo derivation steps.



A helpful tool, e.g., when you search for certain criteria of your model. After each step you can check you structure against you conditions. If they are not fulfilled, undo the last step, change you parameters and try again.



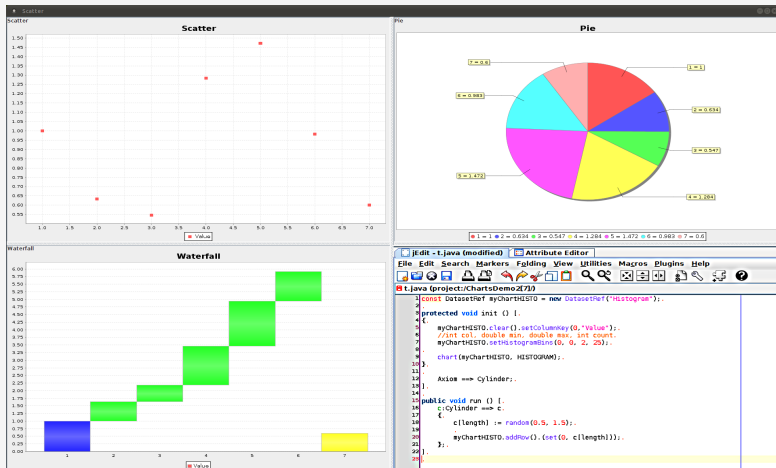
Chart demo



⇒ Example: *Gallery/Technics/ChartsDemo2.gsz*



Chart demo



⇒ Example: *Gallery/Technics/ChartsDemo2.gsz*



Analysis Functions

- ▶ Usable for all on XL-based structures, e.g. RGG-models, models from DTD/LSY/SSY-files
- ▶ Export interfaces:
*.txt, *.html, *.xml, *.csv, *.xls, *.pdf
- ▶ Functions based on the Grogra analysis functions
- ▶ Some analysis functions give a list for further processing e.g. number of daughter shoots, shoots branching positions, length and angles, ...
- ▶ It's a small overview of the analysis functions:
 - ▶ Current 16 analysis functions
 - ▶ Topological & distribution analysis, analysis ...



Analysis Functions

- ▶ Usable for all on XL-based structures, e.g. RGG-models, models from DTD/LSY/SSY-files

View GROGRA Functions									
data Analysis									
#	shoot-ID	diameter	av.length	max.length	no. of longe...	length sum	volume sum	n sum	order
1	6	0.10000000...	3.36159920...	3.36159992...	216	15.8096027...	0.12416827...	0.0	0
2	20	0.10000000...	2.36159992...	2.36159992...	216	7.40479898...	0.05815715...	0.0	1
3	48	0.10000000...	1.56159996...	1.56159996...	216	3.30240011...	0.02593698...	0.0	2
4	104	0.10000000...	0.92159992...	0.92159992...	216	1.33119988...	0.01045522...	0.0	3
5	216	0.10000000...	0.40959995...	0.40959995...	216	0.40959995...	0.00321699...	0.0	4
6	209	0.10000000...	0.40959995...	0.40959995...	209	0.40959995...	0.00321699...	0.0	4
7	97	0.10000000...	0.92159992...	0.92159992...	202	1.33119988...	0.01045522...	0.0	3
8	202	0.10000000...	0.40959995...	0.40959995...	202	0.40959995...	0.00321699...	0.0	4
9	195	0.10000000...	0.40959995...	0.40959995...	195	0.40959995...	0.00321699...	0.0	4
10	41	0.10000000...	1.56159996...	1.56159996...	188	3.30240011...	0.02593698...	0.0	2
11	90	0.10000000...	0.92159992...	0.92159992...	188	1.33119988...	0.01045522...	0.0	3
12	188	0.10000000...	0.40959995...	0.40959995...	188	0.40959995...	0.00321699...	0.0	4
13	181	0.10000000...	0.40959995...	0.40959995...	181	0.40959995...	0.00321699...	0.0	4
14	83	0.10000000...	0.92159992...	0.92159992...	174	1.33119988...	0.01045522...	0.0	3
15	174	0.10000000...	0.40959995...	0.40959995...	174	0.40959995...	0.00321699...	0.0	4
16	167	0.10000000...	0.40959995...	0.40959995...	167	0.40959995...	0.00321699...	0.0	4
17	13	0.10000000...	2.36159992...	2.36159992...	160	7.40479898...	0.05815715...	0.0	1
18	34	0.10000000...	1.56159996...	1.56159996...	160	3.30240011...	0.02593698...	0.0	2
19	76	0.10000000...	0.92159992...	0.92159992...	160	1.33119988...	0.01045522...	0.0	3



Multilingual user interface



English, Chinese



Slovak, French, German



Batch mode

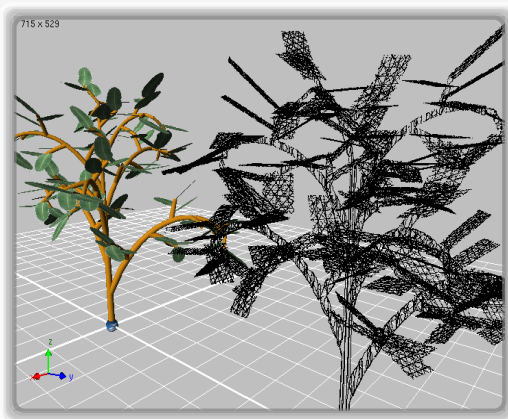
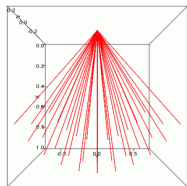
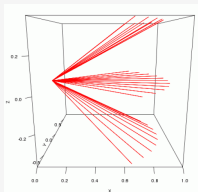
Run GroIMP headless to realise repetitive simulations like automated scenario test.

```
1 java -cp ... de.grogra.pf.boot.Main --project-tree --headless Model.gsz
```

⇒ Examples: *Gallery/Technics/renderHeadlessDemo2.gsz*

Virtual Laser Scanner

Produced point clouds

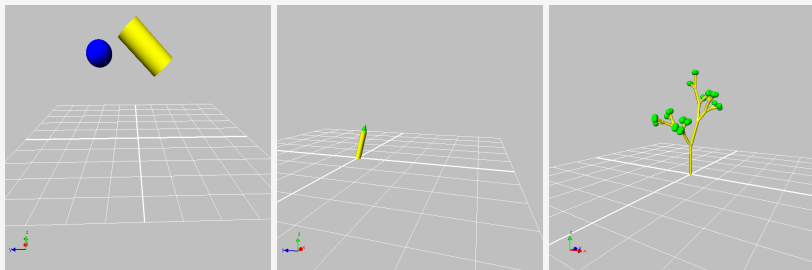


⇒ Etard 2011: [Virtual Laser Scanner for GroIMP](#)

Physics Engine

Robotropism: Distributed Control of Plant

Control of the spatial shape of plants through small artificial controllers attached to the plan.



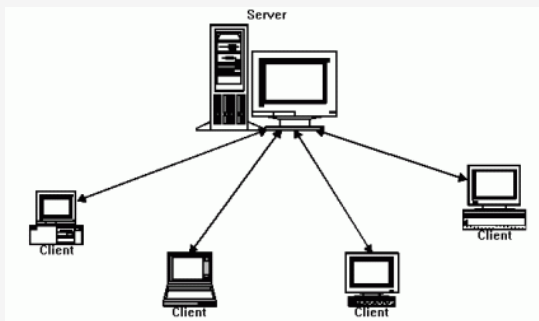
⇒ Masters 2010:

http://www.uni-forst.gwdg.de/~rhemmer/workshop/XL10/Robotropism_v1.pdf



Open-GroIMP Specification

HTTP server: The Open-GroIMP plugin allows to call GroIMP from other applications via the network.

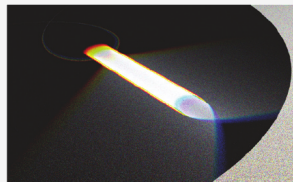
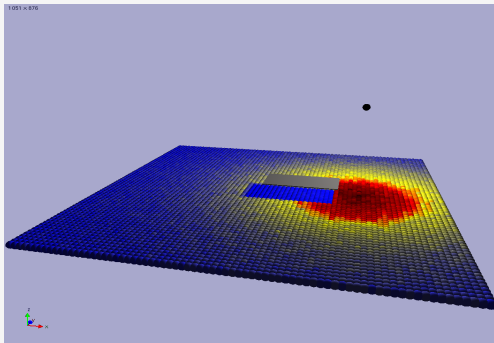


<http://wwwuser.gwdg.de/~groimp/grogra.de/documentation/Specification.txt>

<http://wwwuser.gwdg.de/~groimp/grogra.de/documentation/Description.txt>

Integrated Light Model

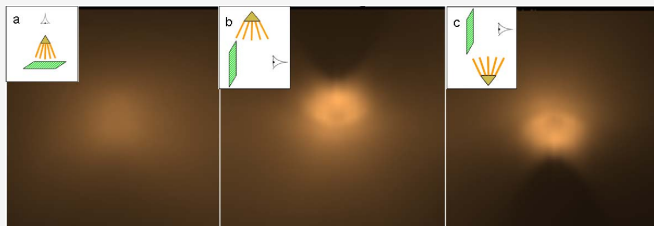
- ▶ External raytracer (POV-Ray)
- ▶ Built-in raytracer (Twilight) & stochastic pathtracer; HDR
- ▶ High performance light model (GPU Flux)
for pathtracer, bidirectional pathtracer and spectral rendering



Supports full spectral rendering (van Antwerpen 2011)

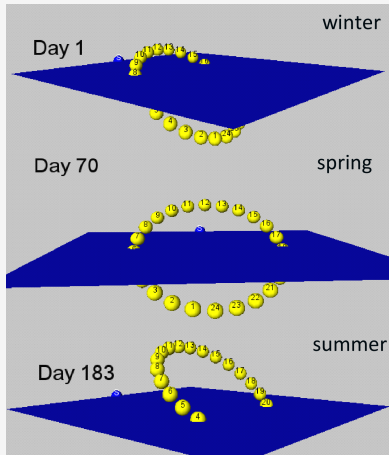
Different Light Sources

- ▶ Direct and indirect light sources
- ▶ Light nodes: point-, spot light
- ▶ Sky object (SunSkyLight shader based on Sunflow)
- ▶ Individually defined light sources: SON-T, LED
Specific light spectrum



⇒ Buck-Sorlin et al. 2010: [Modelling of spatial light distribution in the greenhouse: Description of the model.](#)

Sun model



- ▶ input: latitude, day of year, time of day [h]
- ▶ simulation of direct sunlight as a directional light source (parallel rays), which changes position, thereby adjusting direction



- ▶ simulation of diffuse skylight with sky object (Preetham et al. 1999)

⇒ Buck-Sorlin et al. PMA09: Modelling of spatial light distribution in the greenhouse:
[Description of the model.](#)



GreenLab/XL model

GreenLabModel - GroIMP

File Edit Objects Panels Net Help OpenGroIMP

Reset Stop run Run run

View
View Camera Fit Render View

435 x 344

XL Console Images Shaders Objects

```

Accumulated biomass (g):
Blade: 309.8669178850119
Petiole: 207.23200546203122
Internode: 417.3483291952792
Female: 908.0097569310935
Male: 9.098101437736464
Layer: 0.0
Root: 0.0
Total: 1851.5551109111523

```

jEdit - GreenLabModel.rgg Attribute Editor GreenLab Parameters

Apply Reset to Default Import Parameters Export Parameters

Global Topology Growth Geometry Environment

Chronological Age 33

Physiological Age (PA) 1

Compute Physiology

Parameter Description

Chronological Age (N) Type : Integer
Range : [1, 100]
Unit : growth cycle

D Q QID QR LA LAI E Messages

Growth rate

Graph showing Growth rate (Y-axis, 0 to 100) versus Time (X-axis, 0.0 to 32.5). The graph displays curves for blade, petiole, internode, female, male, layer, and root. The total biomass curve peaks around 17.5.

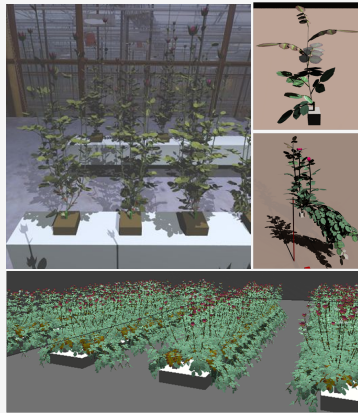
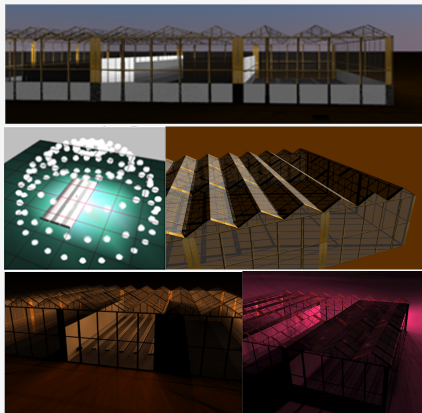


Main attention / Features of the GreenLab/XL model

- ▶ GUI aided input method for parameter input
- ▶ Make it “user save”
 - ▶ Control of inputs (general)
 - ▶ Number fields
 - ▶ Choice boxes
 - ▶ Check boxes
 - ▶ Check input ranges
 - ▶ Deactivate not used fields
 - ▶ Measurement units
- ▶ Description window (parameter description, type, unit, range)
- ▶ Provide im- and export of GreenLab parameter files (“*sci*”)
- ▶ Support AMAPSymbol files (“*smb*”) for shape objects
- ▶ Support of textures



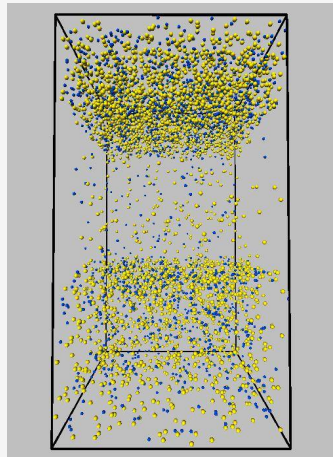
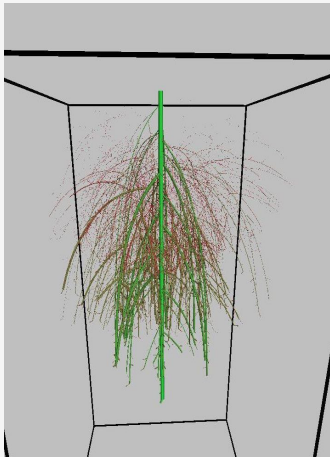
Cut-rose FSPM



(Buck-Sorlin *et al.* 2009, 2011)

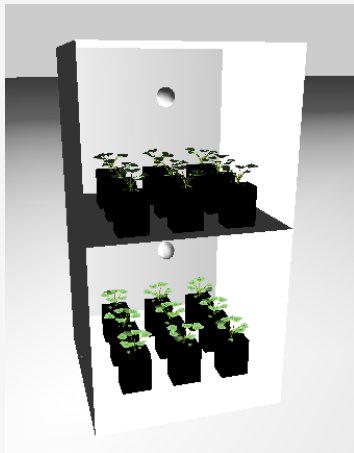


Soil / Root FSPM



(Henke, Sarlikoti, Pagès, Buck-Sorlin, Kurth; unpubl.)

Temperature sensitive rapeseed FSPM

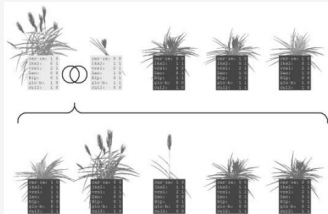
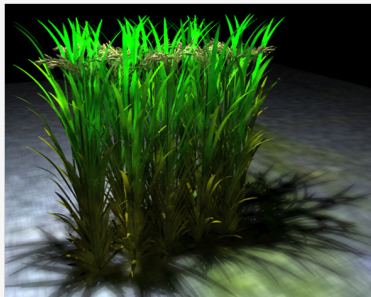


(Tian, Wu, Henke, Zhou, Buck-Sorlin; unpubl.)

Three different temperature regimes are applied to young rapeseed plants in a simulated climate chamber.

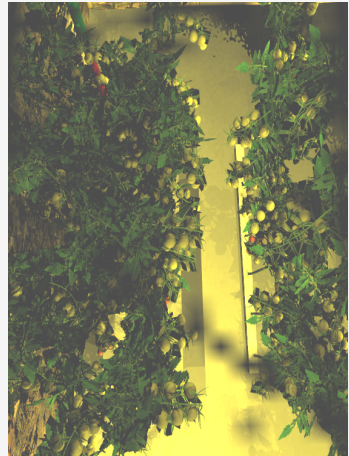


Rice, barley breeder, wheat FSPMs



(rice: Xu *et al.* 2011, barley: Buck-Sorlin *et al.* 2005, wheat: Evers *et al.* 2010)

Arabidopsis, rapeseed, tomato FSPMs



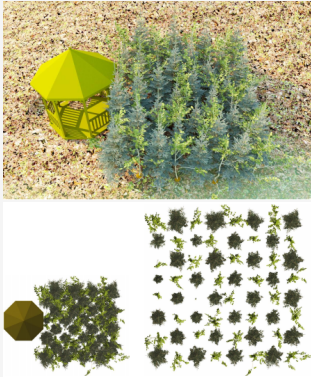
(*Arabidopsis*: Evers *et al.* 2011, rapeseed: Groer *et al.* 2007, tomato: Buck-Sorlin *et al.* unpubl.)

Modelling Park Landscapes



(Rogge & Moschner 2007, für Stiftung Branitzer Park, Cottbus.)

Tree competition (spruce and beech FSPMs)



(Hemmerling *et al.* 2008)



Where to find Help?

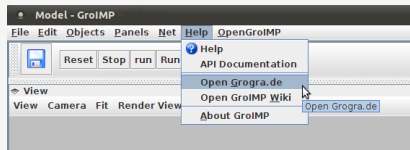
- ▶ GroIMP Project Page:

grogra.de



- ▶ GroIMP Wiki page:

http://sourceforge.net/apps/mediawiki/groimp/index.php?title=Main_Page



- ▶ E-Mail: [info\(at\)grogra.de](mailto:info@grogra.de)



Where to find Help?

- ▶ Lectures:
 Struktur-Funktions-Modelle auf ökophysiologischer Basis
 (german)
- ▶ Video tutorial:
 coming soon (english & german)
- ▶ Courses and tutorials:
 Step by step models
- ▶ Example Gallery:
 At grogra.de
- ▶ Literature:
 Publications W.Kurth
 Publications grogra.de



Where to find Help?

- ▶ General XL:
 - List of turtle commands (german)
 - Overview of XL syntax (german)
- ▶ Dissertations:
 - Kniemeyer, 2008
 - Hemmerling, 2012
- ▶ Master and bachelor thesis:
 - Selected students' theses



GroIMP API

Overview over the whole code of GroIMP

All Classes

Packages

- [de.grogra.annotation](#)
- [de.grogra.bifboard](#)
- [de.grogra.bifboard.vrml](#)
- [de.grogra.blocks](#)
- [de.grogra.blocks.arrangeBlock](#)
- [de.grogra.blocks.functionParser](#)
- [de.grogra.blocks.xFrogFileParser](#)
- [de.grogra.cham](#)
- [de.grogra.ext.cpfq](#)
- [de.grogra.ext.exchangegraph](#)

ProjectionType

- [ProjectLoader](#)
- [Property](#)
- [PropertyAssignment](#)
- [PropertyCompileTime](#)
- [PropertyCompileTimeModel](#)
- [PropertyEditor](#)
- [PropertyEditorManager](#)
- [PropertyEditorTree](#)
- [PropertyEditorTreeNode](#)
- [PropertyEditorTree.SelectionNode](#)
- [PropertyFileConfiguration](#)
- [PropertyFileReader](#)
- [PropertyQueue](#)
- [PropertyQueue.MakePersistentDesc](#)
- [PropertyQueue.PropertyQueueDesc](#)
- [PropertyRuntime](#)
- [PropertyRuntime.GraphProperty](#)
- [PropertyRuntimeModel](#)
- [PropertyRuntimeModel.ElementProp](#)
- [ProtoDeclareImport](#)
- [ProtoInstanceImport](#)
- [PublicCloneable](#)
- [PushInts](#)
- [QuadraticLine](#)
- [Quality](#)
- [Quantity](#)
- [Quarter](#)
- [Quat4d](#)
- [Quat4f](#)
- [Query](#)
- [QueryState](#)
- [QueryState.Break](#)

Overview Package [Class](#) [Tree index](#) [Help](#)

[PRIMARY CLASS](#) [NEXT CLASS](#) [SUMMARY](#) [NESTED](#) | [FIELD](#) | [CONSTRUCTOR](#) | [METHOD](#) [FRAMES](#) [NO FRAMES](#) [DETAIL](#) | [FIELD](#) | [CONSTRUCTOR](#) | [METHOD](#)

de.grogra.pf.io

Class PropertyFileReader

java.lang.Object
↳ de.grogra.pf.io.PropertyFileReader

```
public class PropertyFileReader
extends java.lang.Object
```

Constructor Summary

[PropertyFileReader](#)(FileSource fs)

[PropertyFileReader](#)(java.lang.String fileName)

Method Summary

boolean	getBoolean (java.lang.String propertyString)	Returns the specified property as boolean value.
double	getDouble (java.lang.String propertyString)	Returns the specified property as double value.
double[]	getDoubleArray (java.lang.String propertyString)	Returns the specified property as array of double values.
float	getFloat (java.lang.String propertyString)	Returns the specified property as float value.
float[]	getFloatArray (java.lang.String propertyString)	Returns the specified property as array of float values.
int[]	getIntArray (java.lang.String propertyString)	Returns the specified property as array of integer values.
int	getInteger (java.lang.String propertyString)	Returns the specified property as integer value.
int	getNumberOfReadProperties()	Returns the number of read properties.
java.lang.String	getString (java.lang.String propertyString)	Returns the specified property as string value.
boolean	load()	Loading of a property file.



GroIMP Example Gallery

Included as "Show Examples" or on the net with nearly 400 examples.

The image shows two overlapping windows. The background window is a Mozilla Firefox browser displaying the GroIMP website (grogra.de). The website has a navigation menu with 'Home', 'News', 'Documentation', 'Growth Grammars', 'Software', 'Publications', 'Download', and 'About'. Under '14 models', there is a 'by image:' section with a grid of 14 small images of various plant models, including 'barley_breeder', 'beech', 'bush', 'cypress', 'fir', 'grass', 'linden', 'maple', 'oak', 'poplar', 'rice', 'spring wheat', and 'tamarix model'. The foreground window is the GroIMP software interface, showing a menu bar (File, Edit, Objects, Panels, Net, Help) and a toolbar. The main window displays 'Example Projects for GroIMP' with a list of projects under the heading 'Fractals':

- Fractals
 - [Koch curve](#)
A classical example of L-systems.
 - [Tree with symodial branching](#)
This simple L-system models a tree with symodial branching. The example is based on an example in the book "The Algorithmic Beauty of Plants" of Przemyslaw Prusinkiewicz, Lindenmayer.
- FSPM
 - [FSPM model](#)
This model simulates a mixed-species beech and spruce forest and competition for presented at the FSPM07 (8th International Workshop on Functional Structural Plant Modelling, NZ).
 - [Simple Rapeseed](#)
A structural model of a simple rapeseed plant.
 - [Tree based on pipe model of branch width](#)
This L-system models the branching width of a tree based on the pipe model. The translation of the example "tree-shedding" of the L-Studio software.
 - [Tree branch lengths defined by a function of height](#)
A user-defined function is used to specify the branch length at a given tree height.
 - [Bush](#)
A bush is modelled based on an example in the book "The Algorithmic Beauty of Plants" of



GroIMP Command Observer

Overview and description of GroIMP commands via XL Console

list()

lists all available commands

list("key")

lists all available commands starting with the specific key

help("key")

prints out the help of the command starting with the specific key

```

> list()
A      allowNoninjectiveMatchesByDefault
allowNoninjectiveMatchesForNextQuery ancestor ancestor angle angle
apply apply applyUntilFinished assignparent
B      booleanValue byteValue
C      chart charValue cloneNode cloneSubgraph
cone cone console curve
D      dataset defer derive descendants
directionalTropism distance distanceSquared d
distribution doubleValue
E      enddirection endlocation
export3DScene exportGraphToFile extent
F      file filter floatValue function
G      getComponentGraph getGraphSize
getProjection getSceneGraphSize getTimeForPro
getTimeForTraversingGraph graph graphState
H      height hide
I      image inclination irandom interpretive
intersectionLength intValue irandom isAncestorIn
L      leaf location location lognormal l
M      makeRenderedImage makeRenderedImage m
makeRenderedImage makeSnapshot material m
mergeNonTropismTransformations mergeNonTropismTransfo
mergeTransformations minDescendants moveToExtent
N      namedNode newGVVertices normal n
O      objectValue orthogonalTropism
P      passBoundary plot plot plot plotPoints
positionalTropism print print print print print
print print println println println println

```

```

> list("d")
dataset defer derive descendants direction direction
directionalTropism distance distanceSquared distanceToLine
distribution doubleValue
> help("derive")
Name:
void derive()
Description:
This method induces a , em, transformation boundary, /em, on the current RGG
extent (see the XL Language Specification). This means that all pending graph
modifications are, applied to the graph.
> |

```



GroIMP Function Browser

...or use the Function Browser

The screenshot displays the GroIMP interface. On the left, the 'Function Browser' window is open, showing a list of functions. The 'derive' function is selected and highlighted. The 'Description' pane on the right shows the details for the 'derive' function:

Name: void derive()
Description: This method induces a , em, transformation boundary ,/em, on the current RGG extent (see the XL Language Specification). This means that all pending graph modifications are, applied to the graph.

In the foreground, a menu is open over the 'Help' menu item of the main application window. The menu items are: Help, Function Browser (highlighted), API Documentation, Open Grogra.de, Open GroIMP Wiki, Check for Updates, and About GroIMP.



The Developer Team

Contact

Katarína Smoleňová : ksmolen@uni-goettingen.de
Michael Henke : mhenke@uni-goettingen.de
Yongzhi Ong : yong@gwdg.de

Winfried Kurth : wk@informatik.uni-goettingen.de



Invitation and Call for Papers

International Summer School

“Modelling of Ecosystems by Tools from Computer Science”

combined with the 6th GroIMP user and developer meeting and with a workshop

“Functional-structural plant models in XL”

Date: 16 – 20 September 2013

Start:	16 Sept., 9:00 h
End:	20 Sept., ca. 16:00 h
Workshop:	17 Sept., 14 – 17 h
GroIMP user and developer meeting:	17 Sept., 17 – 18 h

Location:

Czech University of Life Sciences (CULS / ČZU), Faculty of Forestry and Wood Sciences, Praha-Suchbát, Prague, Czech Republic

WWW:

http://www.uni-forst.gwdg.de/~wkurth/ssc_prague.pdf



Thank you for your attention!

www.grogra.de

