.mtg file is textual, contains topological structure, .bgeom file contains objects defined in PlantGL graphics library.

.xeg files are textual, contains graph structure organized in XEG schema.

.gsz file is the GroIMP model file, contains model in XL modeling language.

----------------------------------------------------------------------------------------------------------------------------------

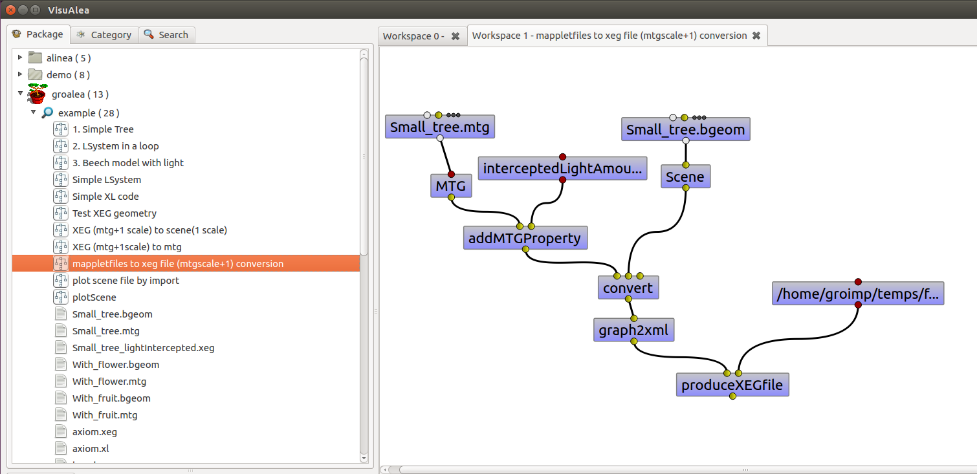
Small\_tree.mtg, Small\_tree.bgeom : files from Vincent. Small tree structure generated by MAppleT.

----------------------------------------------------------------------------------------------------------------------------------

Small\_tree.xeg :

How to get it from OpenAlea platform:

1. file converted from Small\_tree.mtg and Small\_tree.bgeom, through python-OpenAlea “mappletfiles to xeg file (mtgsclae+1) conversion” process.



Remark: it can be imported into GroIMP platform.

----------------------------------------------------------------------------------------------------------------------------------

Small\_tree\_intercepted.xeg :

How to get it from GroIMP platform:

1. Import small tree structure,
2. Run preProcess, lightIntercept (only upscale the property to metamers ), postProcess
3. Export as .xeg file.

Remark: The interceptedLightAmount properties of all the BezierSurface objects have been assigned with updated values.  
----------------------------------------------------------------------------------------------------------------------------------

Small\_tree\_intercepted\_fullscale.xeg :

How to get it

1. Import small tree structure,
2. Run preProcess, lightIntercept (upscale the property to all nodes in graph ), postProcess
3. Export as .xeg file.

Remark: The interceptedLightAmount properties of all the BezierSurface objects have been assigned with updated values, and the interceptedLightAmount properties of all the nodes in the other scales have also been assigned with aggregated values.

----------------------------------------------------------------------------------------------------------------------------------

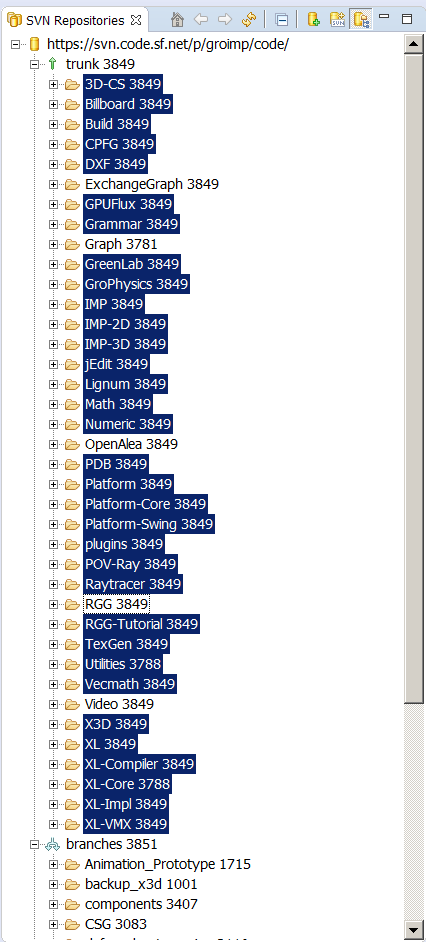
LightInterception\_pre\_intecept\_post\_fullscale.gsz :

How to use it:

1. Checkout the source code of GroIMP platform from svn repository on sourceforge (<https://sourceforge.net/p/groimp/code/HEAD/tree/> ):

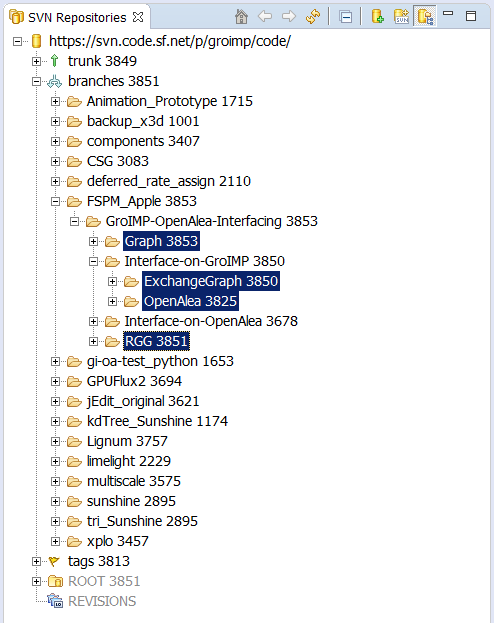
* Checkout platform packages from trunk:

All packages under trunk, except: ExchangeGraph, Graph, OpenAlea, RGG, Video



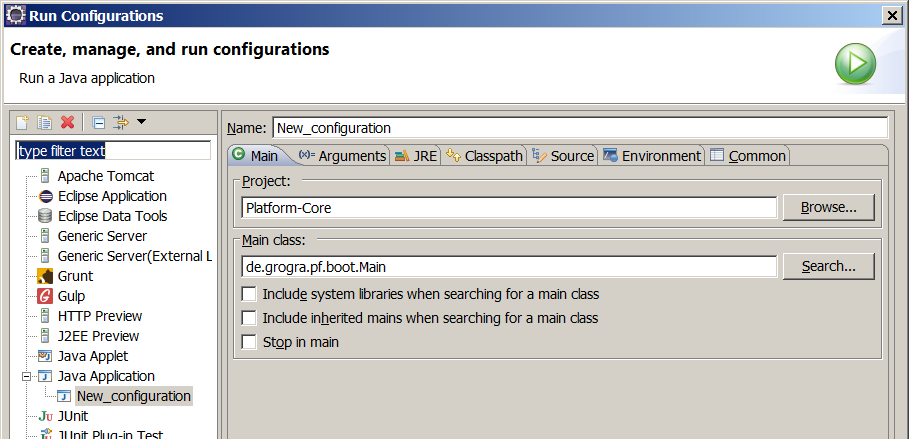
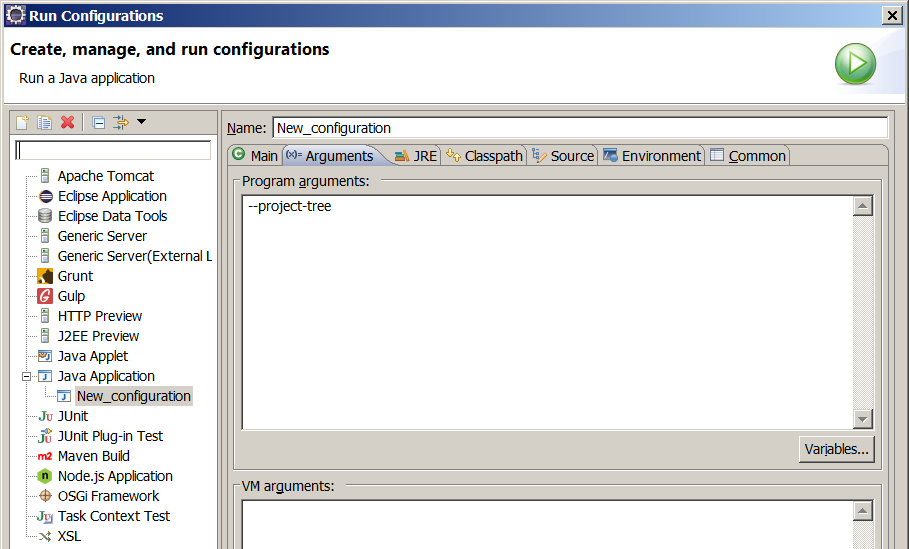
* Checkout interface relevant packages from FSPM\_Apple branch:

4 packages : ExchangeGraph, Graph, OpenAlea, RGG

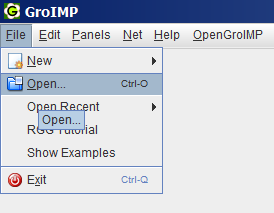
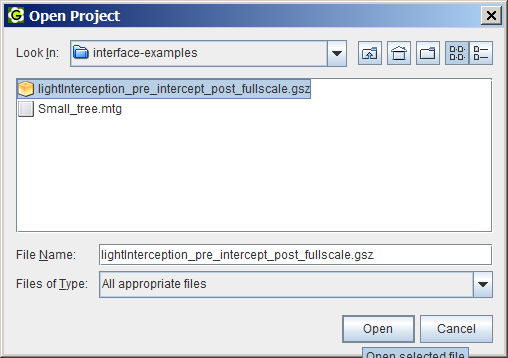


1. Run GroIMP platform:

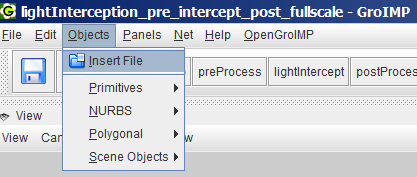
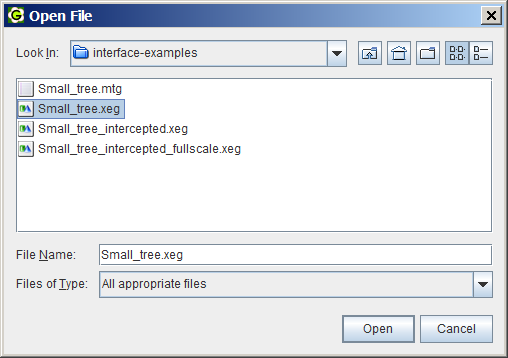
* In eclipse, right click [Run Configurations/Java Application] to add a new configuration with following setting:
  + Main/Project: Platform-Core
  + Main/Main class: de.grogra.pf.boot.Main
  + Arguments/Program arguments: --project-tree
* Then click [apply], the GroIMP platform will be ready to import the XEG file and to run model simulation.

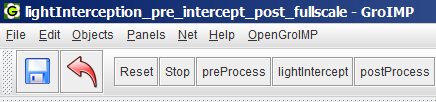
1. Click [File/Open], and choose this file in the [Open Project] dialogue, and click [Open] to open integrative GroIMP model (GroIMP model with additional processes to enable the integration of different FSPMs)

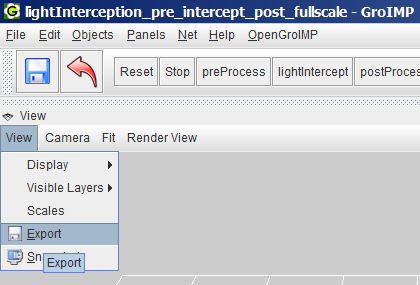
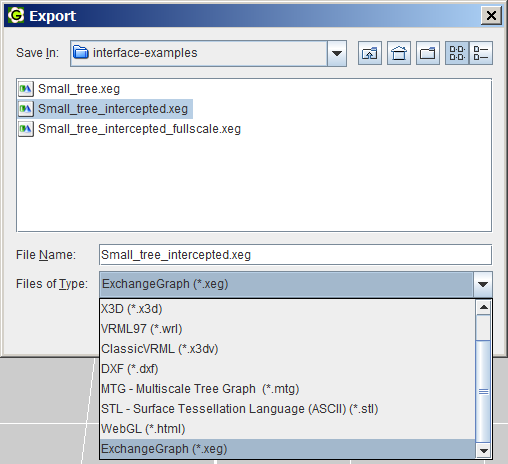
1. Click [Objects/Insert File], and choose .xeg file in the [Open File] dialogue for importing it, and click [Open]

1. Run preProcess, lightIntercept, postProcess by click the named buttons beside of Reset and Stop button



1. Click [View/Export], and choose the file type “ExchangeGraph(\*.xeg)” for exporting the structure into an .xeg file with given name to a directory, and click [Save]

Remark:

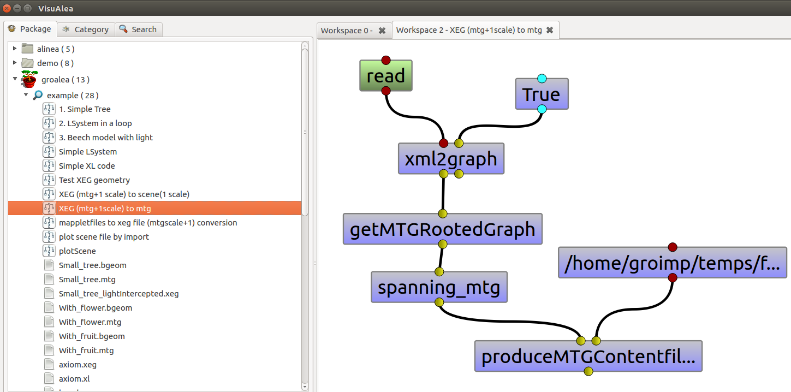
1. In GroIMP, “module” types are extends from graphic types to carry functional properties. Here the Leaf type extends NURBSSurface type to carry interceptedLightAmount property.
2. In GroIMP, there are both visible graphic types (e.g. NURBSSurface) and invisible mathematic types (e.g. BezierSurface). To make the objects of invisible types visible, they need to be wrapped as objects of visible types. (e.g. instantiate an object of NURBSSurface type using an object of BezierSurface as one of the arugments).
3. The preProcess method is to replace all the graphic objects that will involve the GroIMP model simulation with the “module” type defined in GroIMP model properties). Here is the BezierSurface objects, which will be wrapped as NURBSSurface objects and be replaced with leaf objects.
4. The lightIntercept method is to do the light interception by call a dummy model, and to upscale the light values to upper scales. (Using codes without the lines with “\*”, light values will be upscaled to the metamer scale only, otherwise light values will be upscaled to nodes at all scales).
5. The postProcess method is to replace all leaf objects back to objects of NURBSSurface type.

And when the plant structure is exported, all the objects of NURBSSurface type will be changed to objects of BezierSurface type.

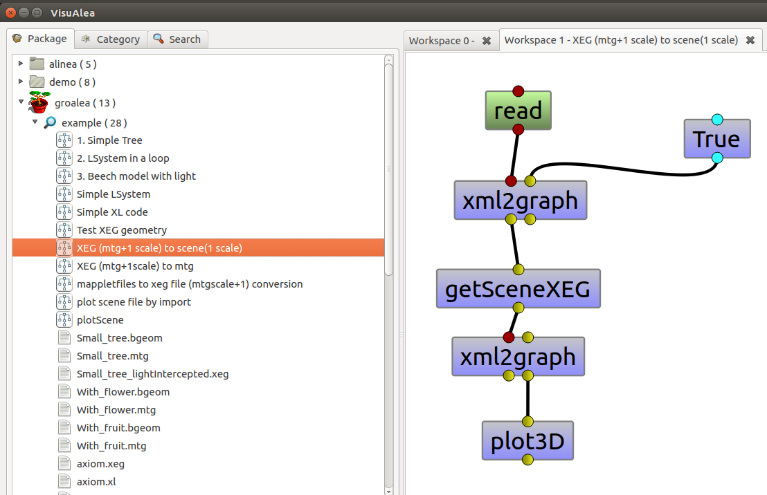
----------------------------------------------------------------------------------------------------------------------------------

To use the Small\_tree\_intercepted.xeg and Small\_tree\_intercepted\_fullscale.xeg :

* Use the python-OpenAlea process “XEG (mtg+1 scale) to mtg” to generate MTG file content



* Use the python-OpenAlea process “XEG (mtg+1 scale) to scene(1 scale) to generate and plot a scene”



* Or new processes to use the .xeg files can be created using defined python nodes:

