HAL
archives-ouvertes.fr

VisuAlea, Towards a Scientific Modelling Environment using Visual Programming

Christophe Pradal, Daniel Barbeau, Thomas Cokelaer, Eric Moscardi

▶ **To cite this version:**

Christophe Pradal, Daniel Barbeau, Thomas Cokelaer, Eric Moscardi. VisuAlea, Towards a Scientific Modelling Environment using Visual Programming. EuroSciPy 2010, 2010, Paris, France. 2010. hal-00831794

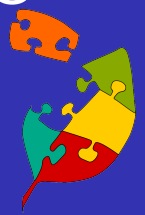
HAL Id: hal-00831794
https://hal.inria.fr/hal-00831794

Submitted on 7 Jun 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

VisuAlea, Towards a Scientific Modelling Environment using Visual Programming



Christophe Pradal[1,2] Daniel Barbeau [1], Thomas Cokelaer[1]
Eric Moscardi[1]

[1]INRIA, [2]CIRAD

EuroSciPy 20010, Paris, 8-11 July 2010

OpenAlea Goals

OpenAlea is an open source platform for modelling plant development and functionning at different scales.

Sharing knowledge

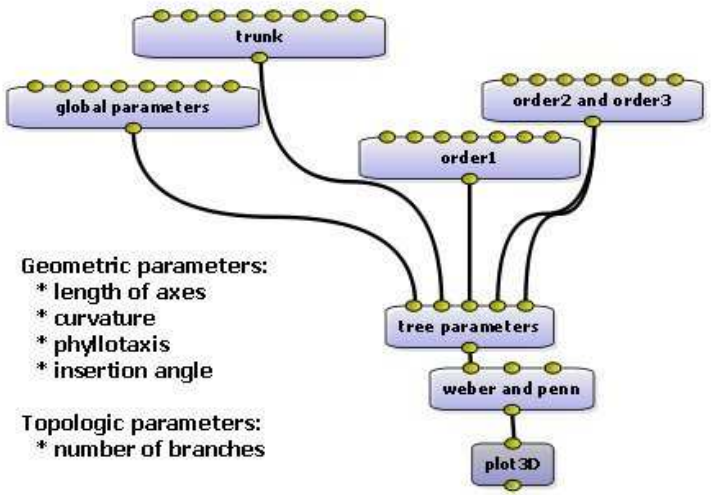
- Reuse software and tools
- Share development between various labs
- Share databases and training efforts

Common software platform

- Integration of existing software & tools
- Rapid development of new models
- Enhance accessibility
- Quality rules

Design choices

- Open Source scientific community
  - Distributed development (sprints)
- Language centric (Python)
  - Common modelling language
  - Glue language
- Component architecture
  - Dynamic composition
  - High-level dataflow approach
- Visual programming (**VisuAlea**)
  - Graphical model representation
  - Automatic GUI generation
- Shared deployment tools
  - Build, packaging, installation, distribution, update



Visual Programming

Visual Programming Environments

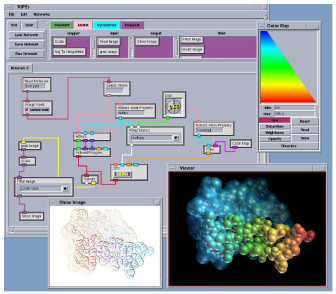
LabView, VTK, Vision, Orange, VisTrails, ...

Advantages

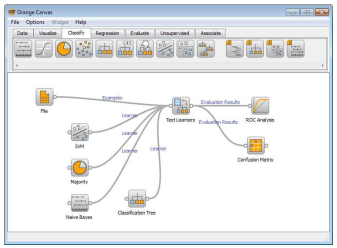
- Interactive creation and modification of flexible workflows
- Visual representation of the structure of a model
- Dynamic composition of software components

Drawbacks

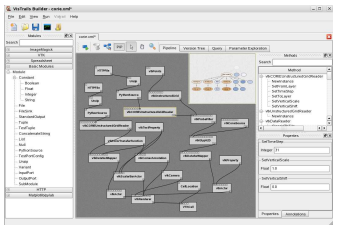
- Less expressive than textual languages (for, while)



Vision (Sanner *et al.*, 2002)

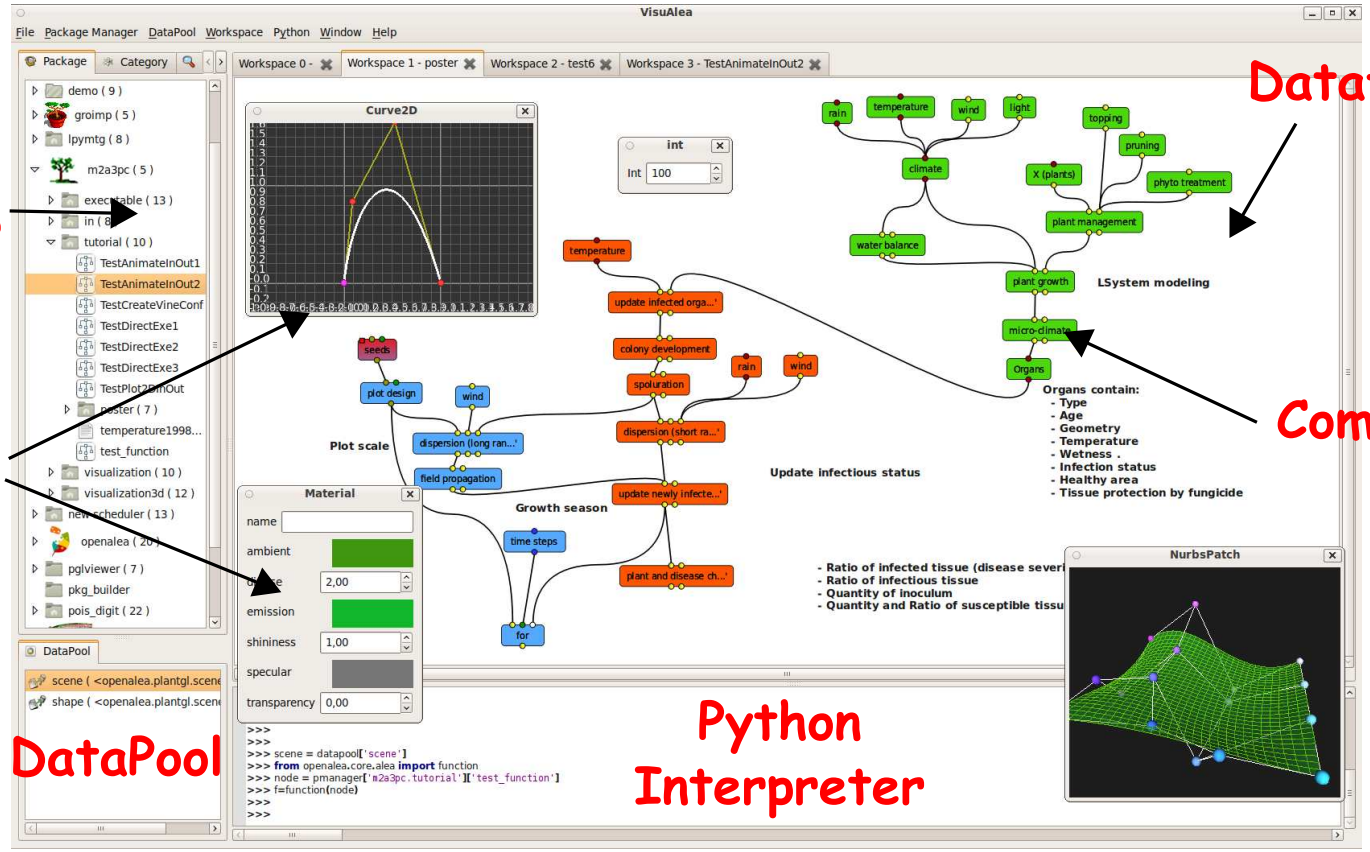


Orange (Demsar *et al.*, 2004)



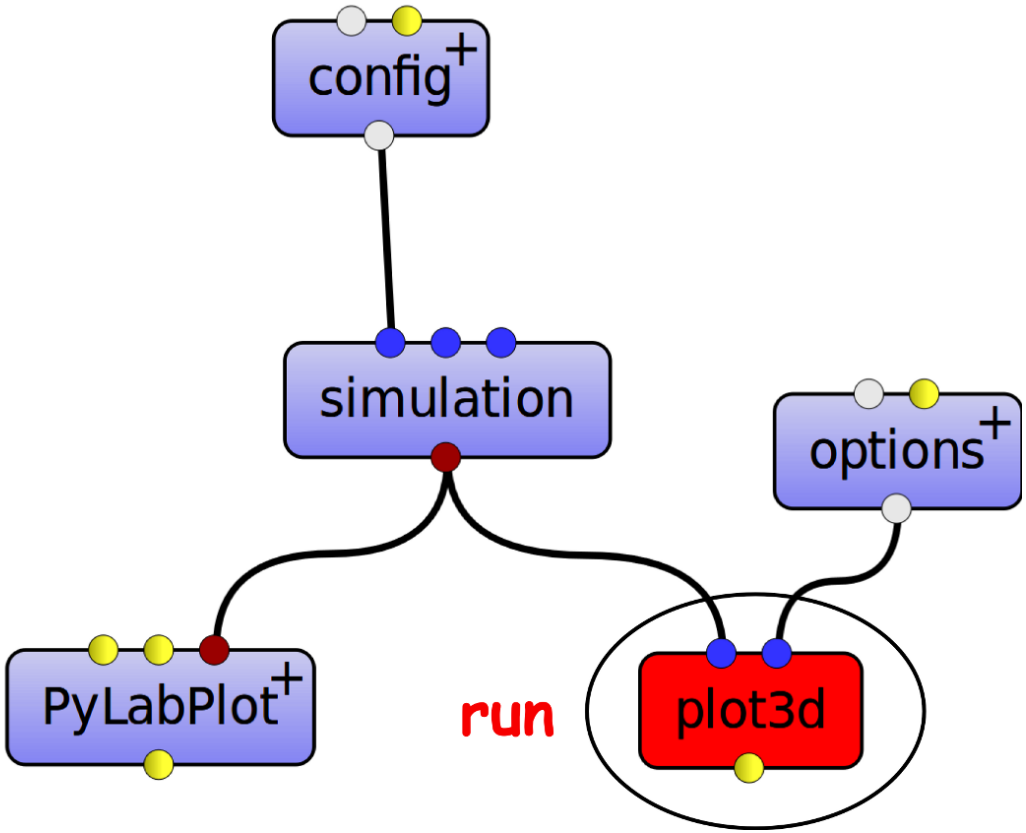
VisTrail (Freire *et al.*, 2005)

VisuAlea


Package Manager
Widgets
DataPool
Dataflow
Component
Python Interpreter
VisuAlea
File Package Manager DataPool Workspace Python Window Help
Workspace 0 - Workspace 1 - poster Workspace 2 - test6 Workspace 3 - TestAnimateInOut2
Curve2D
int
Material
NurbsPatch
LSystem modeling
Organs contain:
- Type
- Age
- Geometry
- Temperature
- Wetness .
- Infection status
- Healthy area
- Tissue protection by fungicide
Update infectious status
Growth season
Plot scale

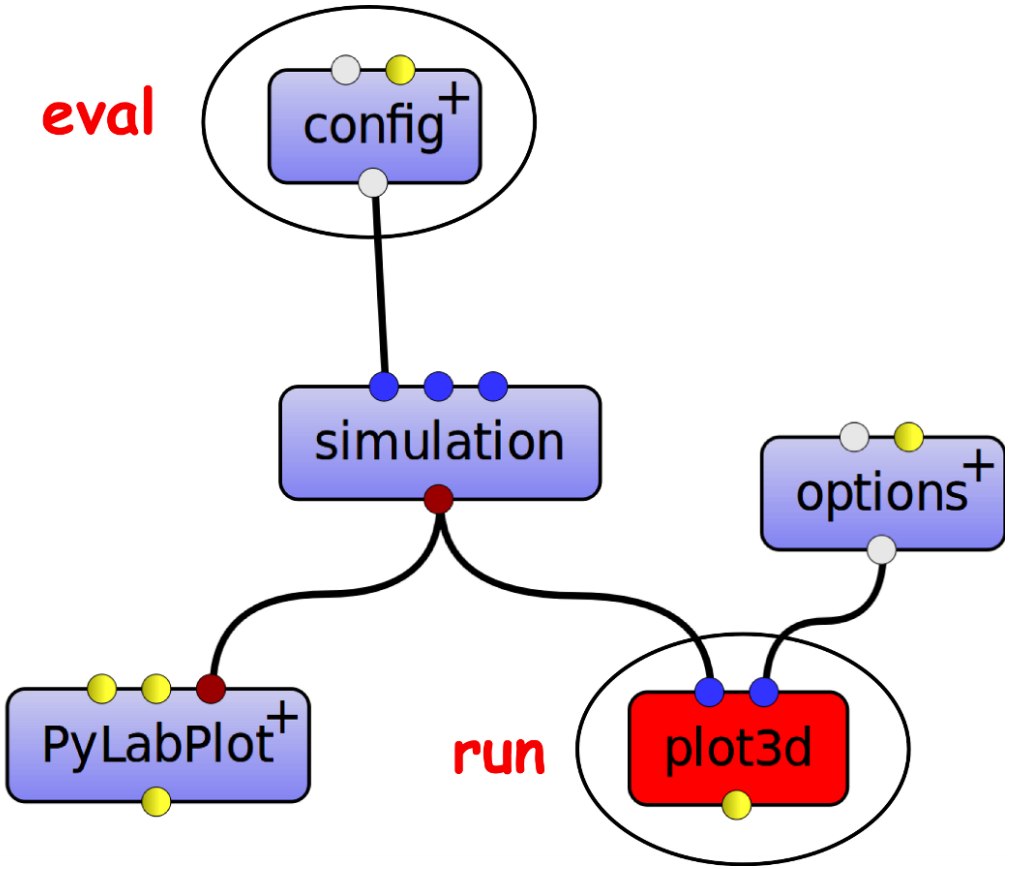
Dataflow Evaluation

Demand driven evaluation



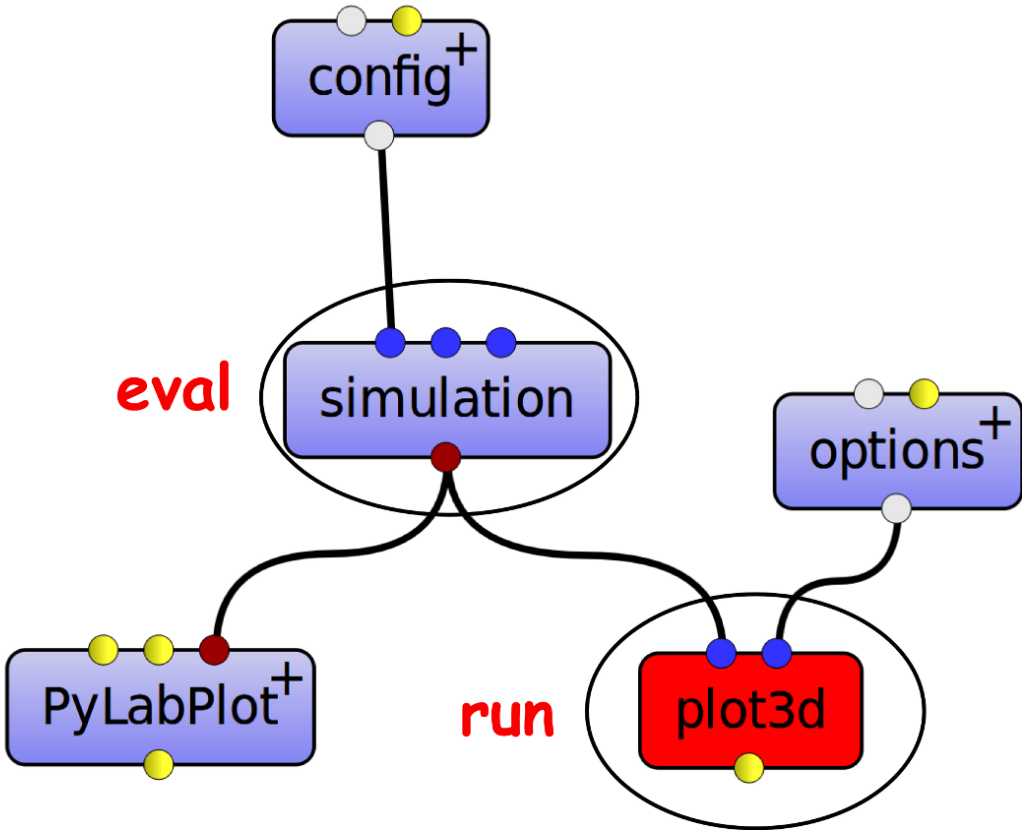
Dataflow Evaluation

Demand driven evaluation



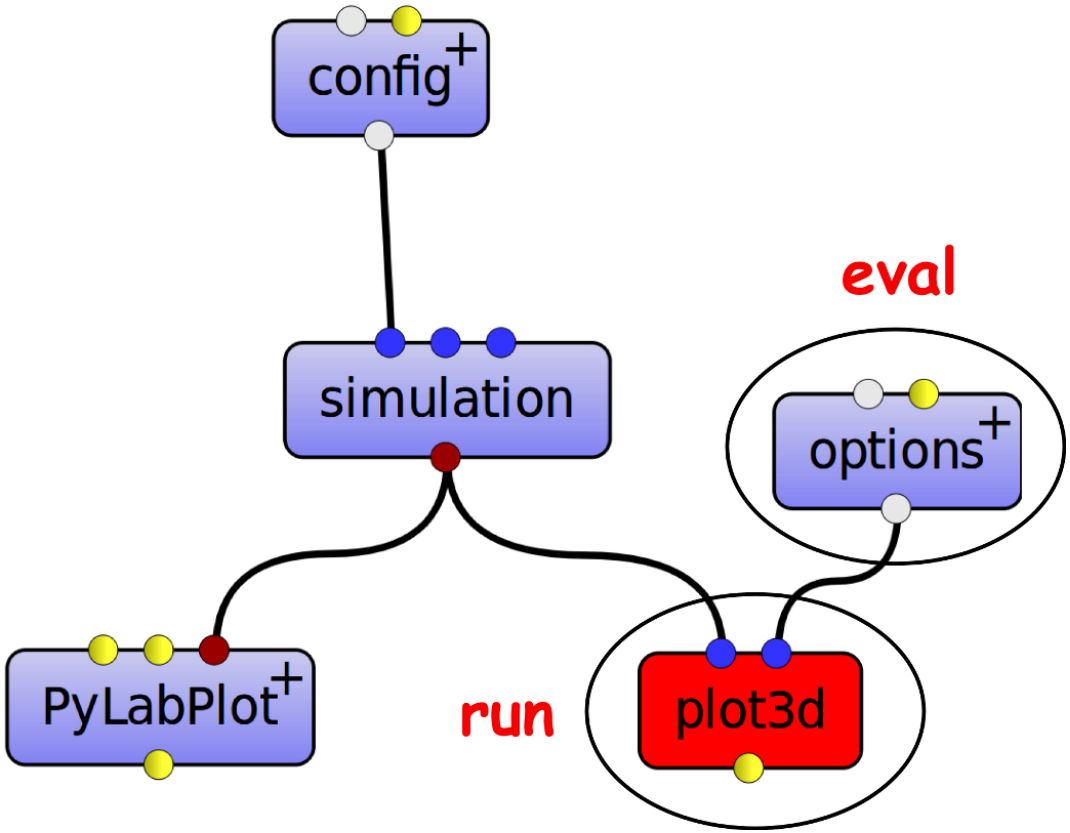
Dataflow Evaluation

Demand driven evaluation



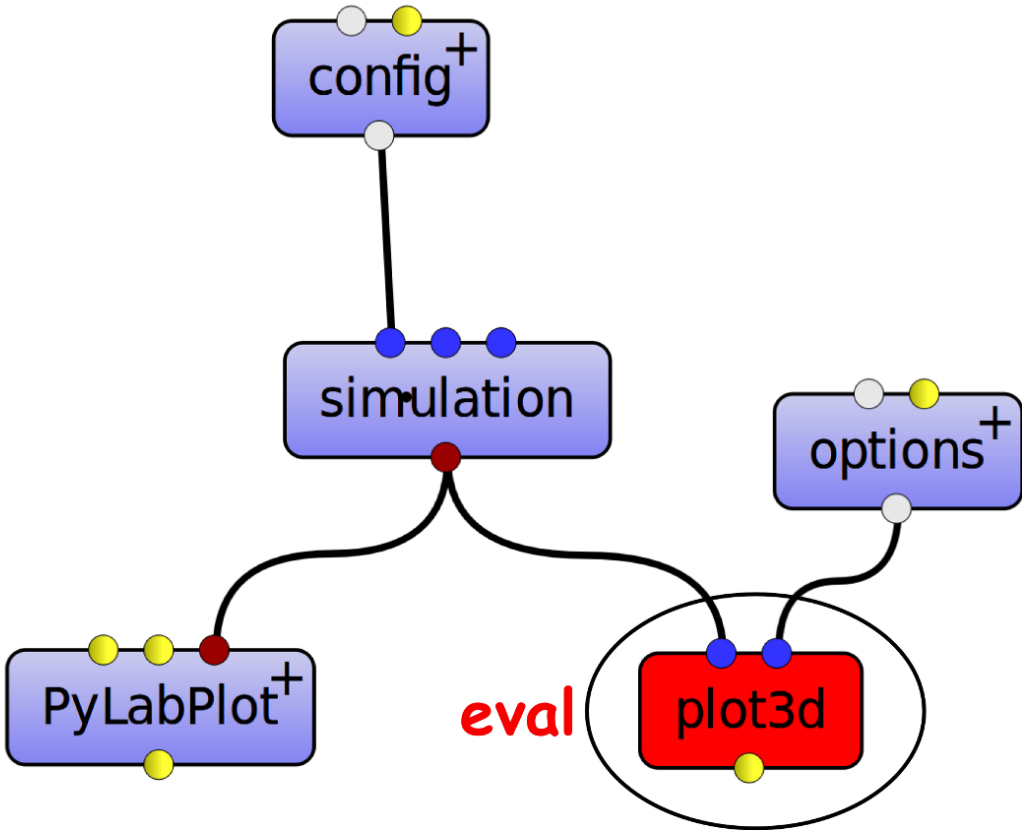
Dataflow Evaluation

Demand driven evaluation



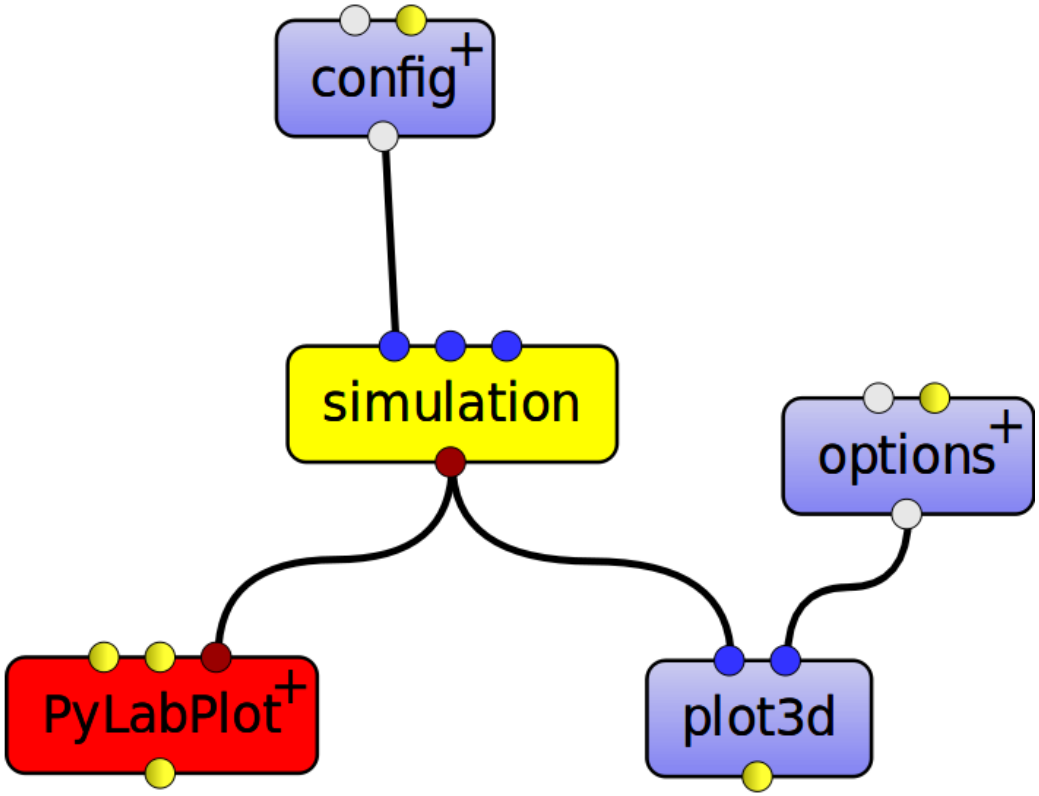
Dataflow Evaluation

Demand driven evaluation



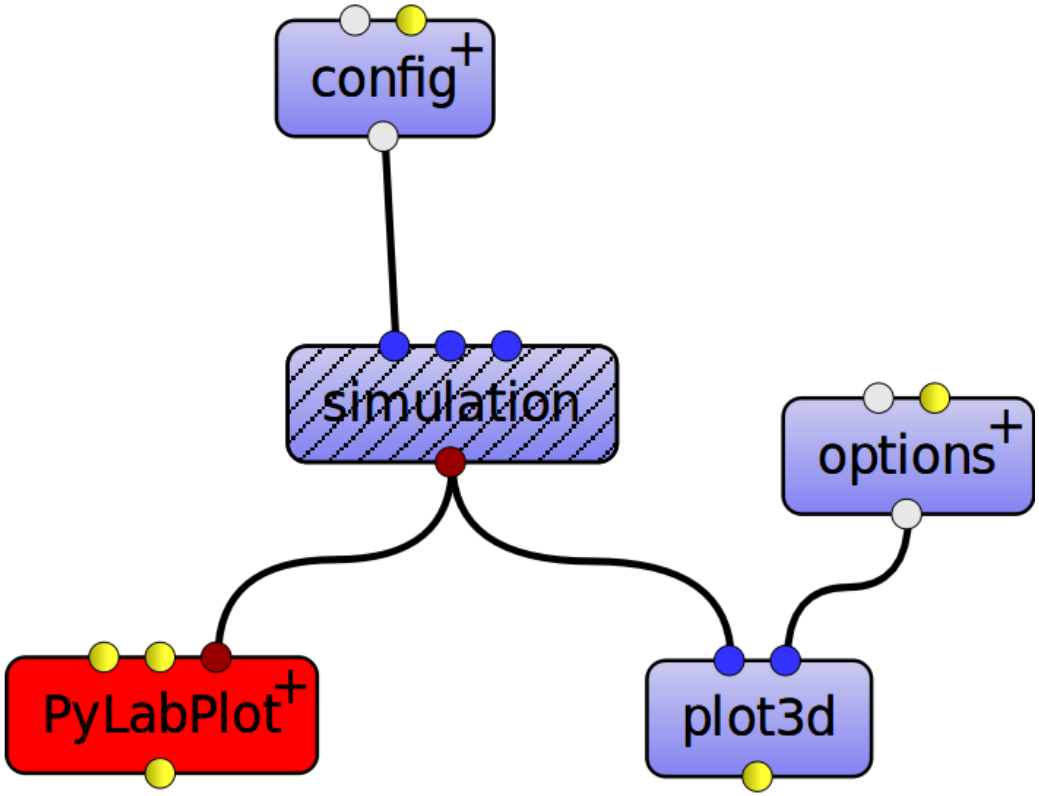
Dataflow Evaluation

Lazy node: re-evaluated only when one of its inputs has changed



Dataflow Evaluation

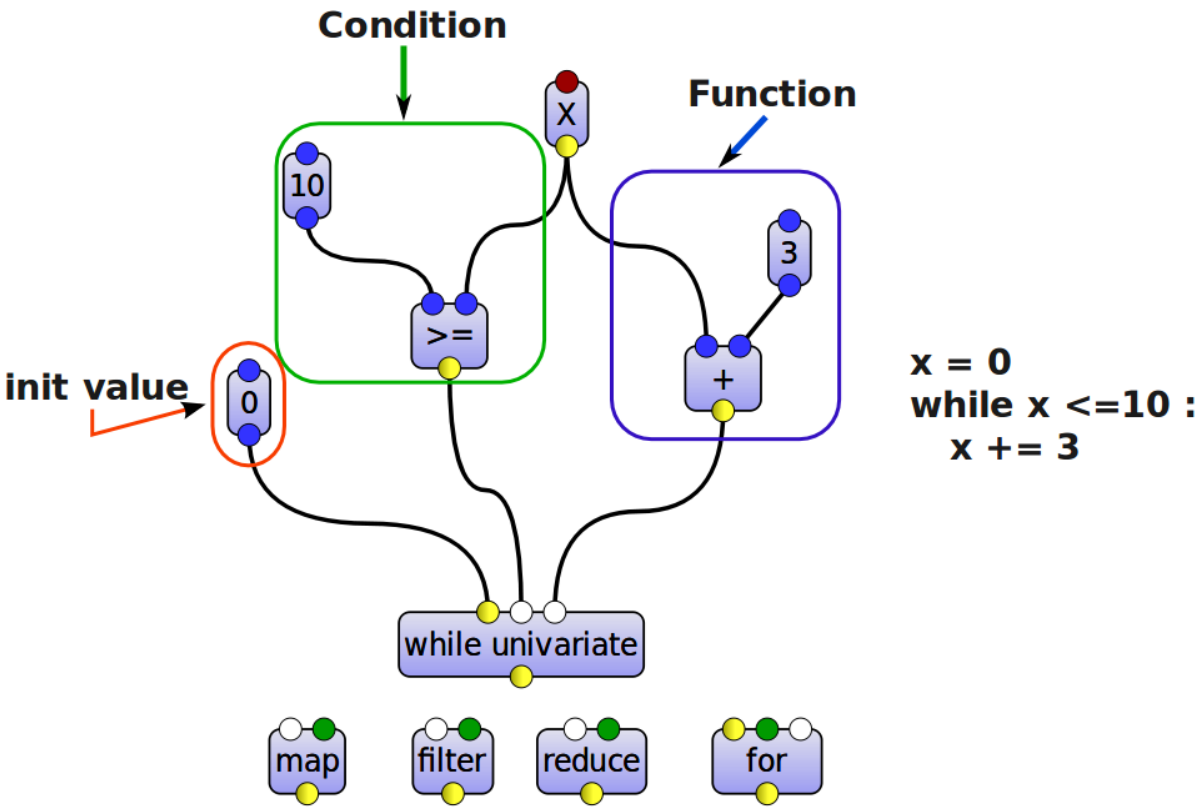
Block node: do not propagate the evaluation



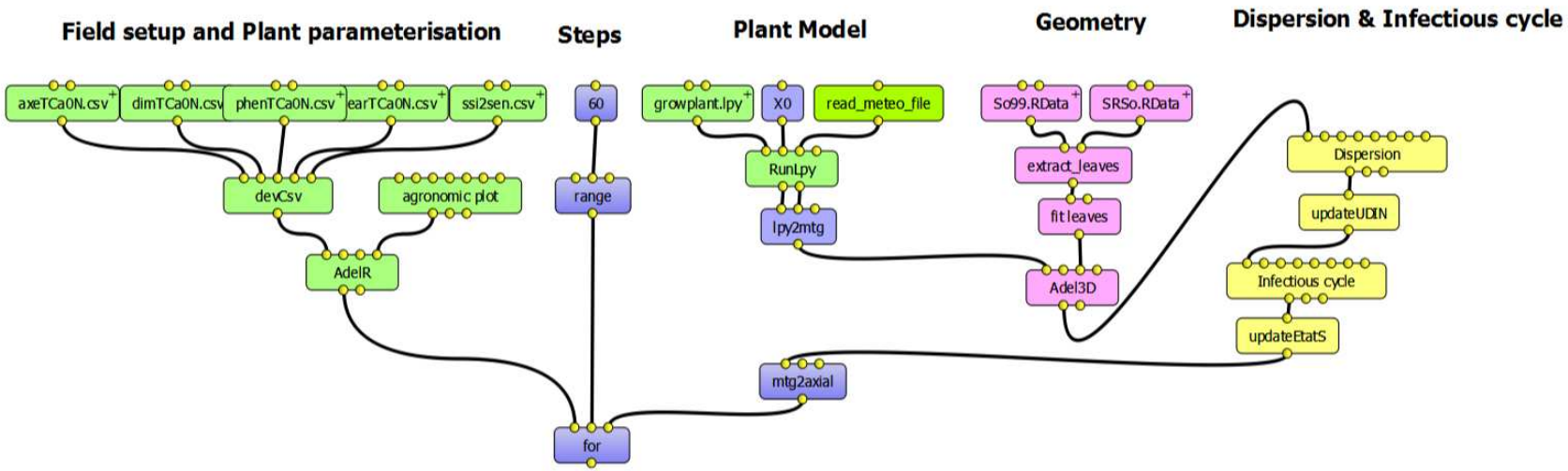
Dataflow Evaluation

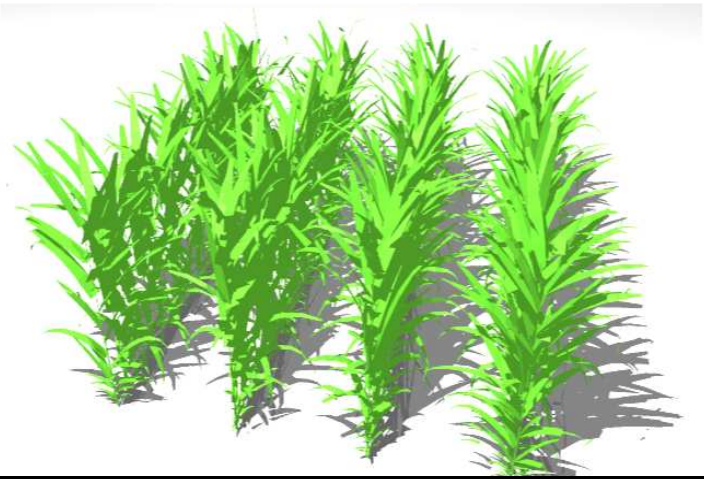
Dataflow = no side effects + no cycle.

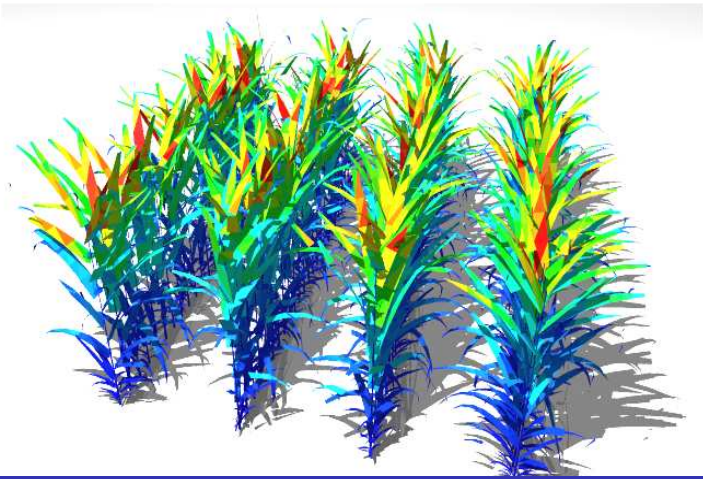
X node: transform a sub-dataflow into a **lambda** function



Example: simulation of plant/disease interaction



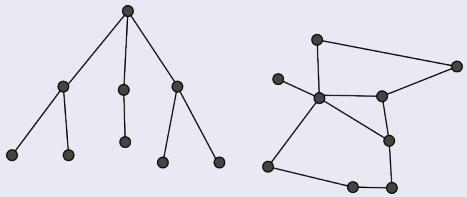




GraphEditor

Need for a **reusable python library** to **view and edit** (m)any **different graph types**, with support for **PyQt4**.

Concepts

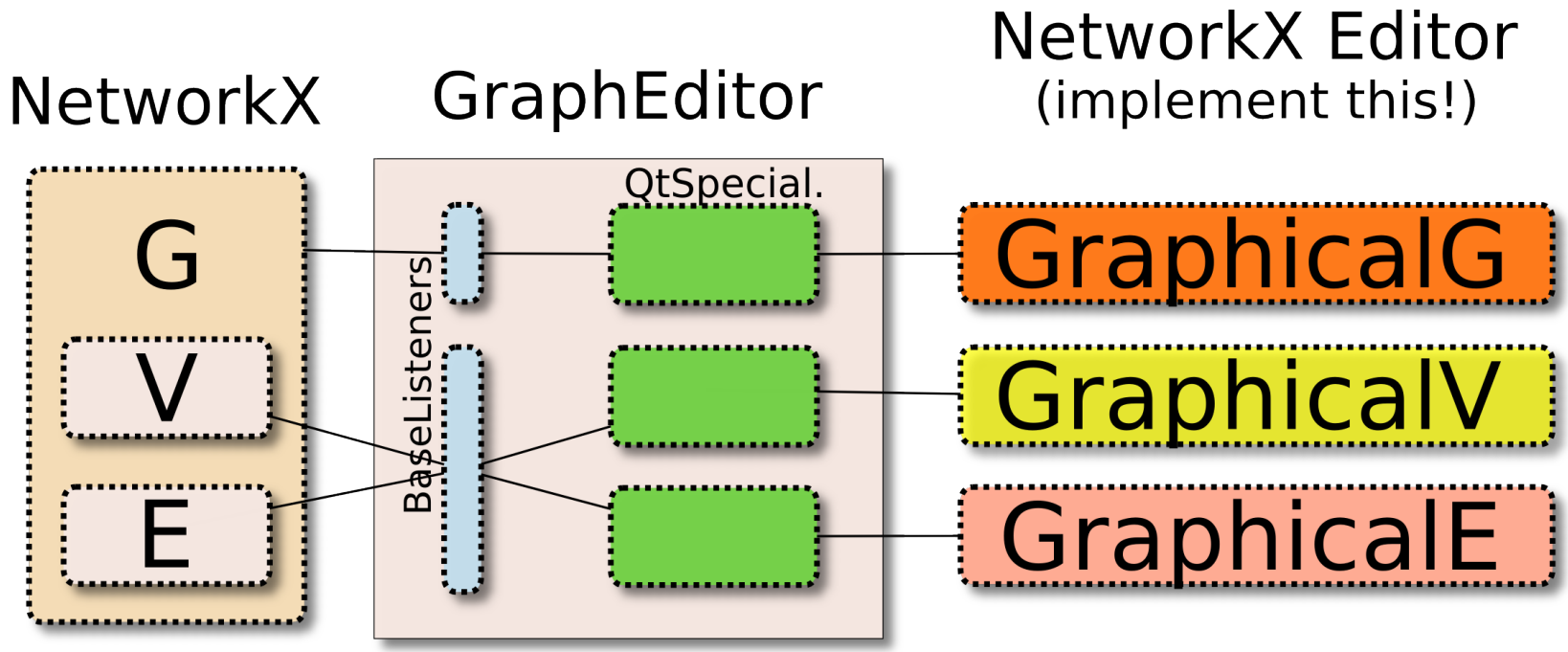


Trees, networks, dataflows (etc ...) boil down to G = { V, E } so
GraphicalG = { *GraphicalV*, *GraphicalE* }

GraphEditor

- Simplifies the implementation of custom graph editors
- Both aspect and interaction are customizable
- Has a PyQt4 implementation of the basic API

Example: Building an editor for NetworkX



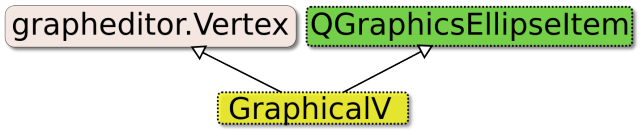
The user implements a strategy to view the data

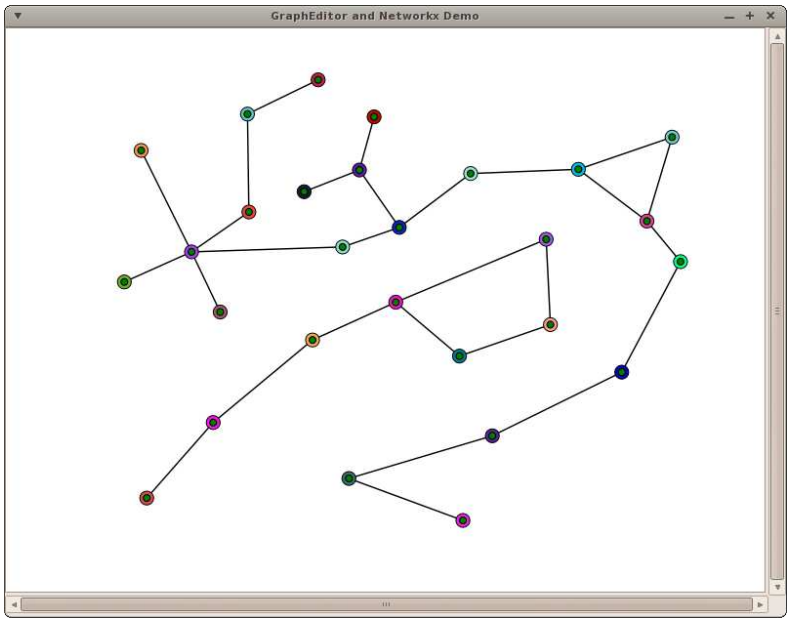
Example: Building an editor for NetworkX

Implement a simple vertex representation

```
class GraphicalVertex(Vertex, QGraphicsEllipseItem):
  def __init__(self, vertex, graph):
    QGraphicsEllipseItem.__init__(self, 0, 0,
                                  20, 20, None)
    Vertex.__init__(self, vertex, graph,
                    defaultCenterConnector=True)
    self.initialise_from_model()
```

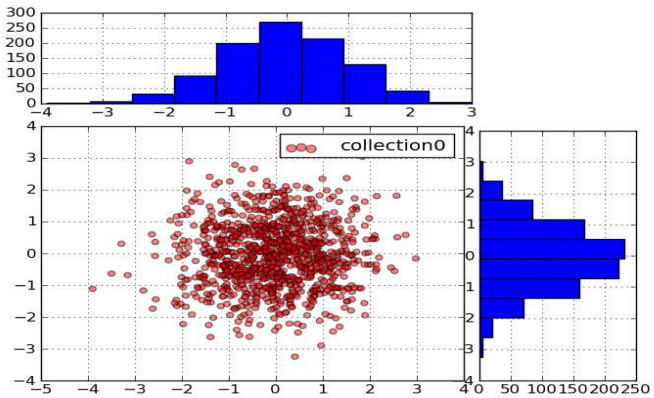
```
  def initialise_from_model(self):
    ''' Read the properties stored in the NetworkX
    graph that can be useful for the view. '''

    # Define the position of the vertex in the view
    self.setPos(self.node()['position'])
    # Define the color of the vertex in the view
    color = self.node()['color']
    self.setBrush(QBrush(color))
```



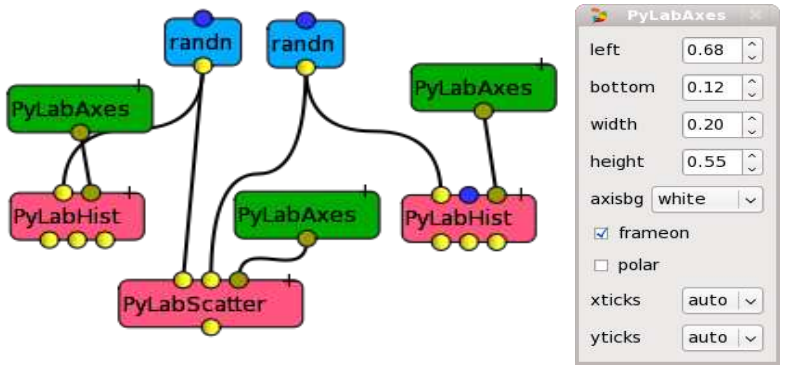


Libraries integration

- In VisuAlea, wrapping/integrating existing librairies into a GUI is made simple.
- Pylab/Matplotlib example: most of Pylab functionalties are available showing the feasability of integrating complex standard librairies into VisuAlea.



PyLab output figure from the dataflow above.



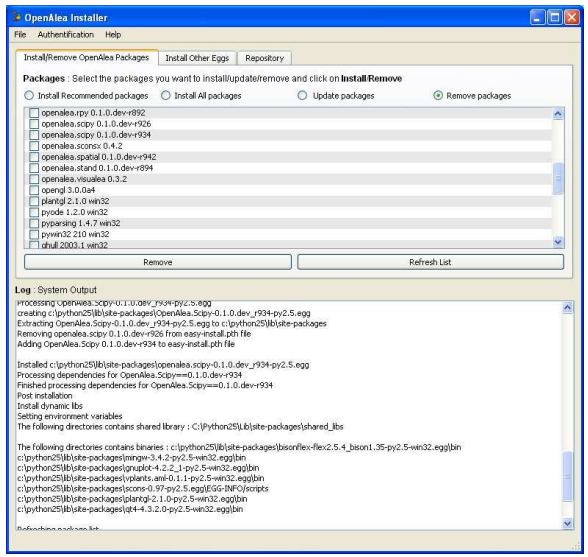
Dataflow that combines scatter and histogram nodes applied on binormal random distribution using Pylab and Numpy functionalities.

- Main advantage: existing options are now accessible as widgets.
- Numpy and Scipy components are integrated on demand.

Deployment and QA

How to distribute large number of binary packages on Mac, Linux, Windows?

- Building & Packaging
  - SCons (C/C++ building) and setuptools: creation of eggs
  - Retrieve the eggs from the web
- Graphical Installer
- Continuous integration (buildbot)
- Automated package creation:
  - SCons files, setup.py, Sphinx conf, ...



Drawbacks

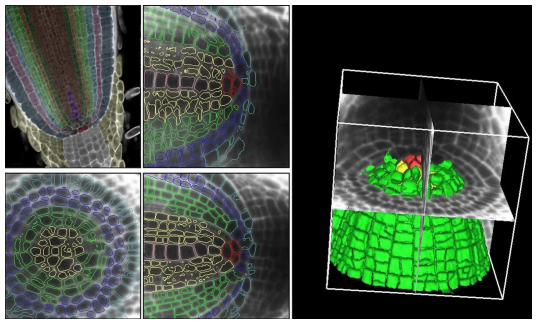
Time consuming and fragile.

Conclusions

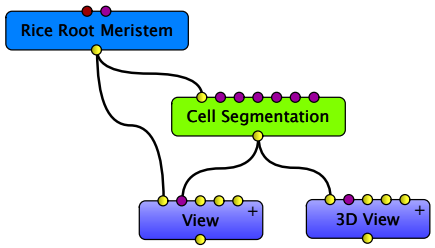
- OpenAlea provides a visual programming environment called VisuAlea
- VisuAlea allows to manage various scientific models in a GUI
  - Foster components/widgets reuse between labs
  - Ease communication
- Recent improvements:
  - Feedback loops using functional programming
  - Graph Editor
  - Many new packages from co-developers: (Biophysics models, image processing, ...)

Perspectives

- Integration of image processing algorithms and visualization tools
  - Registration
  - Fusion
  - Automated cell segmentation
  - Lineage computation
- Parallelization
- Reproductible dataflow simulation



Cells Segmentation and visualization in a rice root meristem (Fernandez *et al.*, Nature Methods, 2010)



Dataflow using a segmentation algorithm and visualization tools

Thank you!

http://openalea.gforge.inria.fr