VisuAlea, Towards a Scientific Modelling Environment using Visual Programming

Christophe Pradal\textsuperscript{1,2}, Daniel Barbeau \textsuperscript{1}, Thomas Cokelaer\textsuperscript{1}, Eric Moscardi\textsuperscript{1}

\textsuperscript{1}INRIA, \textsuperscript{2}CIRAD

EuroSciPy 2010, Paris, 8-11 July 2010
OpenAlea is an open source platform for modelling plant development and functioning at different scales.

**Sharing knowledge**
- Reuse software and tools
- Share development between various labs
- Share database and training effort

**Common software platform**
- Integration of existing software & tools
- Rapid development of new models
- Enhance accessibility
- Quality rules
Design choice

- 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 Environment
LabView, VTK, Vision, Orange, VisTrails, ...

Advantages
- Interactive creation and modification of flexible workflow
- 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

Dataflow

Component

Python Interpreter

OpenAlea
Dataflow

GraphEditor and Libraries integration

Project management

Conclusions and future directions

OpenAlea Goals

The OpenAlea project

Visual Programming
Dataflow Evaluation

Demand driven evaluation

Dataflow Evaluation

Demand driven evaluation

eval \rightarrow config^+ \rightarrow simulation \rightarrow PyLabPlot^+ \rightarrow run

plot3d \rightarrow options^+
Dataflow Evaluation

Demand driven evaluation

Diagram:

- config
- simulation
- PyLabPlot
- plot3d
- options

Dataflow Evaluation

Demand driven evaluation

Dataflow Evaluation

Demand driven evaluation
Lazy node: re-evaluated only when one of its inputs has changed
Block node: do not propagate the evaluation

Dataflow Evaluation

Dataflow = no side effect + 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 libraries into a GUI is made simple.
- Pylab/Matplotlib example: most of Pylab functionalities are available showing the feasibility of integrating complex standard libraries into VisuAlea.

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 demands.
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, ...)
Integration of image processing algorithms and visualization tools
- Registration
- Fusion
- Automated cell segmentation
- Lineage computation

Parallelization

Reproducible dataflow simulation

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

http://openalea.gforge.inria.fr