VisTrails

David Koop
NYU-Poly
VisTrails
VisTrails
VisTrails

• Comprehensive **provenance infrastructure** for computational tasks
• Focus on **exploratory** tasks such as simulation, visualization, and data analysis
• **Transparency**ly tracks provenance of the discovery process—from data acquisition to visualization
  - The **trail** followed as users generate and test hypotheses
  - Users can refer back to any point along this trail at any time
• Leverage provenance to **streamline exploration**
• Focus on **usability**—build tools for scientists
VisTrails

- Open-source, freely downloadable system (www.vistrails.org)
  - Mac & Windows binaries, also on GitHub (github.com/vistrails)
- Multi-platform: users on Mac, Linux, and Windows
- Python code and uses PyQT and Qt for the interface
- User’s guide, wiki, and mailing list
- Many users in different disciplines and countries:
  - Visualizing environmental simulations (CMOP STC)
  - Simulation for solid, fluid and structural mechanics (Galileo Network, UFRJ Brazil)
  - Quantum physics simulations (ALPS, ETH Zurich)
  - Climate analysis (CDAT)
  - Habitat modeling (USGS)
  - Open Wildland Fire Modeling (U. Colorado, NCAR)
  - High-energy physics (LEPP, Cornell)
  - Cosmology simulations (LANL)
  - Using tms for improving memory (Psychiatry, U. Utah)
  - eBird (Cornell, NSF DataONE)
  - Astrophysical Systems (Tohline, LSU)
  - NIH NBCR (UCSD)
  - Pervasive Technology Labs (Heiland, Indiana University)
  - Linköping University
  - University of North Carolina, Chapel Hill
  - UTEP
UV-CDAT: Climate Analysis

- Climate-specific app built on VisTrails workflows and provenance
data = vtk.vtkStructuredPointsReader()
data.SetFileName('..//examples/data/head.120.vtk')

contour = vtk.vtkContourFilter()
contour.SetInput(data.GetOutput())
contour.SetValue(0, 67)

mapper = vtk.vtkPolyDataMapper()
mapper.SetInput(contour.GetOutput())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)

cam = vtk.vtkCamera()
cam.SetViewUp(0, 0, -1)
cam.SetPosition(745, -453, 369)
cam.SetFocalPoint(135, 135, 150)
cam.ComputeViewPlaneNormal()

ren = vtk.vtkRenderer()
ren.AddActor(actor)
ren.SetActiveCamera(cam)
ren.ResetCamera()
renwin = vtk.vtkRenderWindow()
renwin.AddRenderer(ren)
style = vtk.vtkInteractorStyleTrackballCamera()
iren = vtk.vtkRenderWindowInteractor()
iren.SetRenderWindow(renwin)
iren.SetInteractorStyle(style)
iren.Initialize()
iren.Start()
data = vtk.vtkStructuredPointsReader()
data.SetFileName(../examples/data/head.120.vtk)

contour = vtk.vtkContourFilter()
contour.SetInput(data.GetOutput())
contour.SetValue(0, 67)

mapper = vtk.vtkPolyDataMapper()
mapper.SetInput(contour.GetOutput())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)

cam = vtk.vtkCamera()
cam.SetViewUp(0, 0, -1)
cam.SetPosition(745, -453, 369)
cam.SetFocalPoint(135, 135, 150)
cam.ComputeViewPlaneNormal()

ren = vtk.vtkRenderer()
ren.AddActor(actor)
ren.SetActiveCamera(cam)
ren.ResetCamera()
renwin = vtk.vtkRenderWindow()
renwin.AddRenderer(ren)

style = vtk.vtkInteractorStyleTrackballCamera()
iren = vtk.vtkRenderWindowInteractor()
iren.SetRenderWindow(renwin)
iren.SetInteractorStyle(style)
iren.Initialize()
iren.Start()
Workflows

data = vtk.vtkStructuredPointsReader()
data.SetFileName(../examples/data/head.120.vtk)

contour = vtk.vtkContourFilter()
contour.SetInput(data.GetOutput())
contour.SetValue(0, 67)

mapper = vtk.vtkPolyDataMapper()
mapper.SetInput(contour.GetOutput())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)

cam = vtk.vtkCamera()
cam.SetViewUp(0, 0, -1)
cam.SetPosition(745, -453, 369)
cam.SetFocalPoint(135, 135, 150)
cam.ComputeViewPlaneNormal()

ren = vtk.vtkRenderer()
ren.AddActor(actor)
ren.SetActiveCamera(cam)
ren.ResetCamera()
renwin = vtk.vtkRenderWindow()
renwin.AddRenderer(ren)

style = vtk.vtkInteractorStyleTrackballCamera()
iren = vtk.vtkRenderWindowInteractor()
iren.SetRenderWindow(renwin)
iren.SetInteractorStyle(style)
iren.Initialize()
iren.Start()
Workflows

```python
import vtk

data = vtk.vtkStructuredPointsReader()
data.SetFileName(../examples/data/head.120.vtk)

contour = vtk.vtkContourFilter()
contour.SetInput(data.GetOutput())
contour.SetValue(0, 67)

mapper = vtk.vtkPolyDataMapper()
mapper.SetInput(contour.GetOutput())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)

cam = vtk.vtkCamera()
cam.SetViewUp(0,0,-1)
cam.SetPosition(745,-453,369)
cam.SetFocalPoint(135,135,150)
cam.ComputeViewPlaneNormal()

ren = vtk.vtkRenderer()
ren.AddActor(actor)
ren.SetActiveCamera(cam)
ren.ResetCamera()
renwin = vtk.vtkRenderWindow()
renwin.AddRenderer(ren)

style = vtk.vtkInteractorStyleTrackballCamera()
iren = vtk.vtkRenderWindowInteractor()
iren.SetRenderWindow(renwin)
iren.SetInteractorStyle(style)
iren.Initialize()
iren.Start()
```

- Orchestrate multiple tools
- Structured: easier to understand
- Natural granularity for tracking modifications
- Simpler maintenance
VisTrails Change-based Provenance

• **Linear** undo/redo stacks mean that the **history** of **exploration** is often lost

• Old Solution: User saves files/state

• VisTrails Solution:
  - **Automatically** & **transparently** capture entire history as a **tree**
  - Users can tag or annotate each version
  - Users can go back to **any** version by selecting it in the tree
Change-based Provenance

- Isosurface
- Isosurface Script
- Volume Rendering HW
- Volume Rendering SW
- Histogram
- Clipping Plane HW
- Clipping Plane SW
- Combined Rendering HW
- Combined Rendering SW
- Image Slices HW
- Image Slices SW
Structure of Changes

Change 1 (add module):
add module MplPlot

Change 2 (change configuration):
add function source("vspr = self.getInputFromPort(...)"

Change 3 (add connection):
add connection vtkStructuredPointsReader → MplPlot

Change 4 (paste):
add module MplFigure
add module MplFigureCell
add connection MplFigure → MplFigureCell

Change 5 (add connection):
add connection MplPlot → MplFigure
Execution Provenance

<module id="12" name="vtkDataSetReader"
    start_time="2010-02-19 11:01:05"
    end_time="2010-02-19 11:01:07">
    <annotation key="hash"
        value="c54bea63cb7d912a43ce"/>
</module>

<module id="13" name="vtkContourFilter"
    start_time="2010-02-19 11:01:07"
    end_time="2010-02-19 11:01:08"/>

<module id="15" name="vtkDataSetMapper"
    start_time="2010-02-19 11:01:09"
    end_time="2010-02-19 11:01:12"/>

<module id="16" name="vtkActor"
    start_time="2010-02-19 11:01:12"
    end_time="2010-02-19 11:01:13"/>

<module id="17" name="vtkCamera"
    start_time="2010-02-19 11:01:13"
    end_time="2010-02-19 11:01:14"/>

<module id="18" name="vtkRenderer"
    start_time="2010-02-19 11:01:14"
    end_time="2010-02-19 11:01:14"/>

...
Filenames are often the mode of identification in data exploration

We might also use URIs or access curated data stores

- Can this always be expected for exploratory tasks?
- What happens if offline?

Solution:

- Managed store for data associated with computations
- Improved data identification
- Automatic versioning
Linking Provenance and Data

- Filenames are often the mode of identification in data exploration
- We might also use URIs or access curated data stores
  - Can this always be expected for exploratory tasks?
  - What happens if offline?
- Solution:
  - Managed store for data associated with computations
  - Improved data identification
  - Automatic versioning
Full Data Provenance

12ab3-45ef2...
12ab3-45ef2...
12ab3-45ef2...

newfilename.dat

HASH CONTENTS

0ab678cd...

QUERY FILE STORE

FILE STORE

OBTAIN FILE REFERENCE

12ab3-45ef2...

QUERY PROVENANCE

OBTAIN INPUT REFS

input files

0ab678cd...

QUERY FILE STORE

OBTAIN INPUT FILES

12ab3-45ef2...

Thursday, May 30, 13
Making code available in VisTrails

- Package infrastructure
- Wrap python libraries, command-line calls (CLTools), or use other interfaces (jpype, rpy, Web services, etc.)

Need to specify:
1. Package identification information
2. Module structures: input & output ports
3. Compute method for each module
Example: Wrapping an existing python library

- seawater python package:
  - http://pypi.python.org/pypi/seawater/1.0.3

```python
identifier = 'org.ocefpaf.seawater'
version = '1.0.3'
name = 'Seawater Routines'

import seawater

class SaturationN2(Module):
    _input_ports = [('S', Float),
                    ('T', Float)]
    _output_ports = [('res', Float)]

    def compute(self):
        s = self.getInputFromPort("S")
        t = self.getInputFromPort("T")
        res = seawater.satN2(s, t)
        self.setResult('res', res)

    _modules = [SaturationN2,
```
Publishing Results in LaTeX

The ALPS project release 2.0:
Open source software for strongly correlated systems


¹Theoretische Physik, ETH Zurich, Switzerland
²Department of Physics, Colorado School of Mines, Golden, CO 80401, USA
³Department of Physics and Astronomy, University of Wyoming, Laramie, Wyoming 82071, USA
⁴Institute for Theoretical Physics, Georg-August-Universität Göttingen, Göttingen, Germany
⁵Columbia University, New York, NY 10027, USA
⁶Bartol Research Institute, Swarthmore, PA, USA
⁷Center for Computational Science & Engineering, Lawrence Berkeley National Laboratory, Berkeley, CA, USA
⁸CERN, European Organization for Nuclear Research, Geneva, Switzerland
⁹Max-Planck-Institut für Festkörperforschung, Stuttgart, Germany
¹⁰Institut für Theoretische Physik, Universität Regensburg, Regensburg, Germany
¹¹Helmholtz Institute for Solutions Chemistry, University of Konstanz, Konstanz, Germany
¹²Institut für Theoretische Physik, Universität zu Köln, Köln, Germany
¹³Technische Universität München, München, Germany
¹⁴University of California, Los Angeles, USA
¹⁵University of California, Davis, USA
¹⁶University of California, Santa Barbara, USA
¹⁷University of California, San Diego, USA
¹⁸University of California, Los Angeles, USA
¹⁹University of California, Berkeley, USA
²⁰University of California, Los Angeles, USA
²¹University of California, San Diego, USA

1. Corresponding author: tropez@empa.ch


Transcript written on alps20.log.
Longevity and Maintenance

• Virtual machines enable repeatability

• Our focus is on making computations repeatable and reusable
  - Adapt existing work for user’s environment
  - Adapt existing work to take advantage of new algorithms or resources (architectures, memory bandwidth, etc.)

• Workflow Upgrades
  - Developers can define changes from one version to the next
  - Users can also make modifications: changes are stored as provenance
Other Benefits of Reproducibility

- Support reflective reasoning
- Compare different data products
- Explore parameter spaces and compare results
- Suggest new directions
- Sharing and collaboration
Exploring and Comparing Data & Results

- Parameter Exploration
VisComplete

• Similar to textual completions on the web and in user interfaces
• **Mine** provenance collection: Identify fragments that co-occur in a collection of workflows
• Predict sets of likely workflow additions to a given partial workflow
VisComplete
Sharing and Collaboration

- crowdLabs: a social web site for sharing workflows and provenance
  - www.crowdlabs.org
  - Upload workflows from VisTrails
  - Run and explore workflows from a web browser
Adding Provenance to 3rd-Party Tools

Autodesk Maya
Adding Provenance to 3rd-Party Tools

Autodesk Maya

ParaView
Adding Provenance to 3rd-Party Tools

Autodesk Maya

ParaView

VisIt
Provenance SDK

• Enable existing and new applications to incorporate provenance
Reproducibility Summary

- Installation: Mac/Windows binaries, Linux package manager support
- Representation: Workflows and annotations
- Capture: Automatic capture of provenance during workflow creation and execution
- Portability: Python modules are portable, portable as the dependencies, associated data needs to be accessible or bundled
- Document Support: LaTeX extensions allow embedded workflows and provenance in published papers
- Extensions: Workflow updates, parameter exploration, adding new packages via VisTrails wrappers
- Longevity: Versioning and workflow upgrades