Embrio and Wuiw
two web interfaces for WPS

Lorenzo Becchi (ominiverdi.org)

1:00 pm, September 26, 2007

Adapted from Jachym Cepický (GRASS goes web: PyWPS 0.1)
http://Les-ejk.cz

ominiverdi

Embrio & Wuiw

- Embrio
  - Technologies: DHTML + AJAX + PHP/Mapscript
  - Scope: an interface based on UMN Mapserver Mapscript to access GRASS functions (Mapserver and WPS on the same server).
  - Dev state: beta
- Wuiw
  - Tecnologie: DHTML + AJAX
  - Scope: an interface as man in the middle between a WPS server and other OWS servers (WCS, WFS) joining remote resources for data and processing.
  - Dev state: alpha

PyWPS in action - WPS Demo

- http://pywps.ominiverdi.org/
  - Repository of demo apps created by ominiverdi.org
  - Developers:
    - Lorenzo Becchi (ominoverde)
    - Luca Casagrande (doktoreas)
  - All applications shown have to be considered under development. If you find a bug, please, post it to PyWPS's mailing list : pywps-devel@wald.intevation.org

Embrio on pywps.ominiverdi.org

Embrio: A new way to see through the GRASS (by doktoreas and ominiverdi)

http://pywps.ominiverdi.org/subversion/trunk/web/

test_los3.png (PNG Image, 5842... | Embrio: A new way to see throug...

EMBRIO - a simple PyWPS AJAX Web Interface

This page is intended to show the development status of a **Web User Interface** for **PyWPS**

Demo applications

- **V.Buffer** - Create a buffer around features of given type (areas must contain centroid).
- **R.Los** - Generates a raster map output in which the cells that are visible from a user-specified observer location are marked with integer values that represent the vertical angle (in degrees) required to see those cells (viewshed)..
- **V.Net.Path** - Find shortest path on vector network.
- **R.walk and r.drain** Find shortest path between 2 points using slope factor as cost value.

Demo applications of ka-Map tool

This applications use ka-Map API to create tiled cache and map navigation. SLD style definition is supported for the WPS output layer.

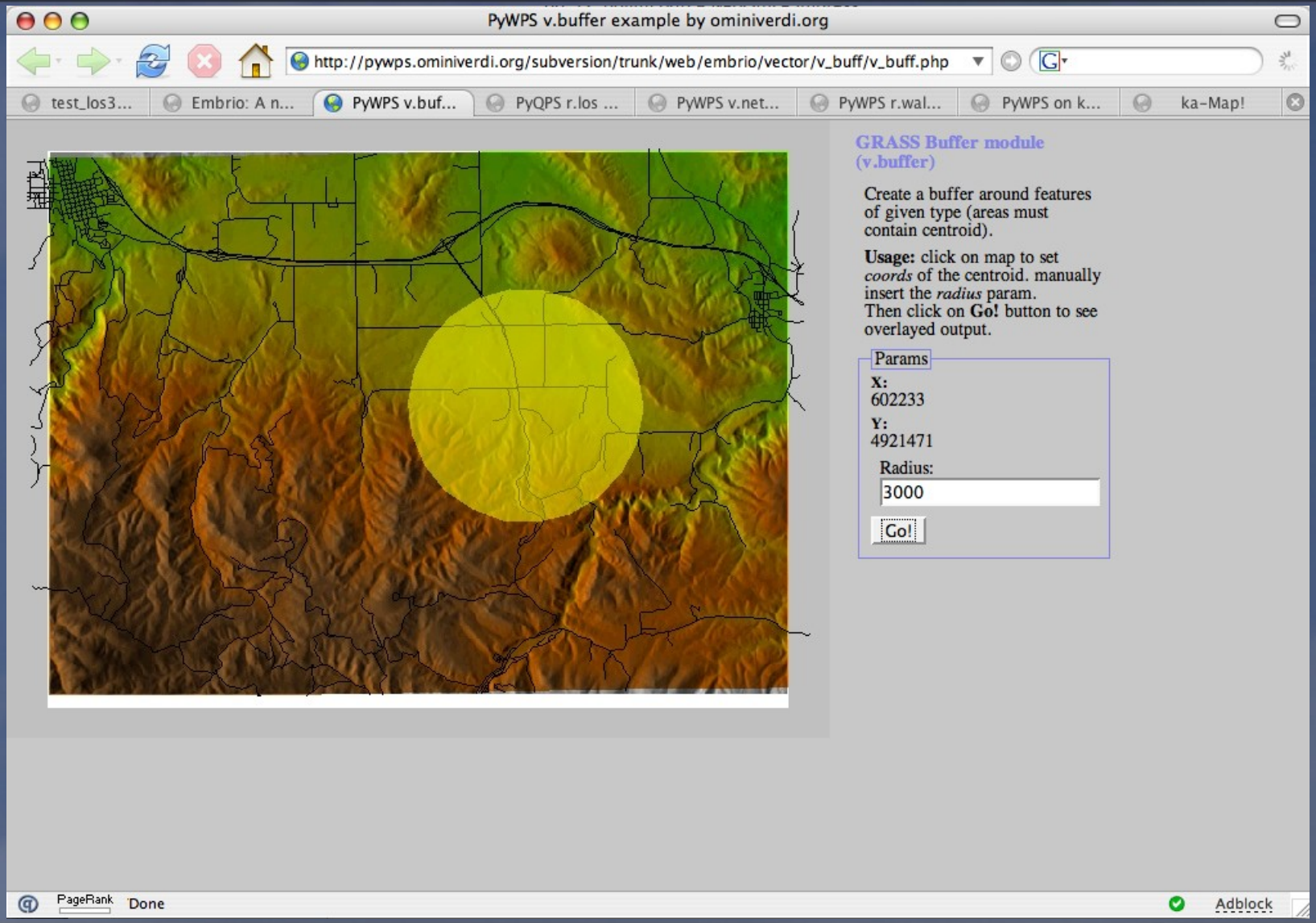
- **R.Los with Embrio interface**
- **R.Los with Winman interface**

Developers

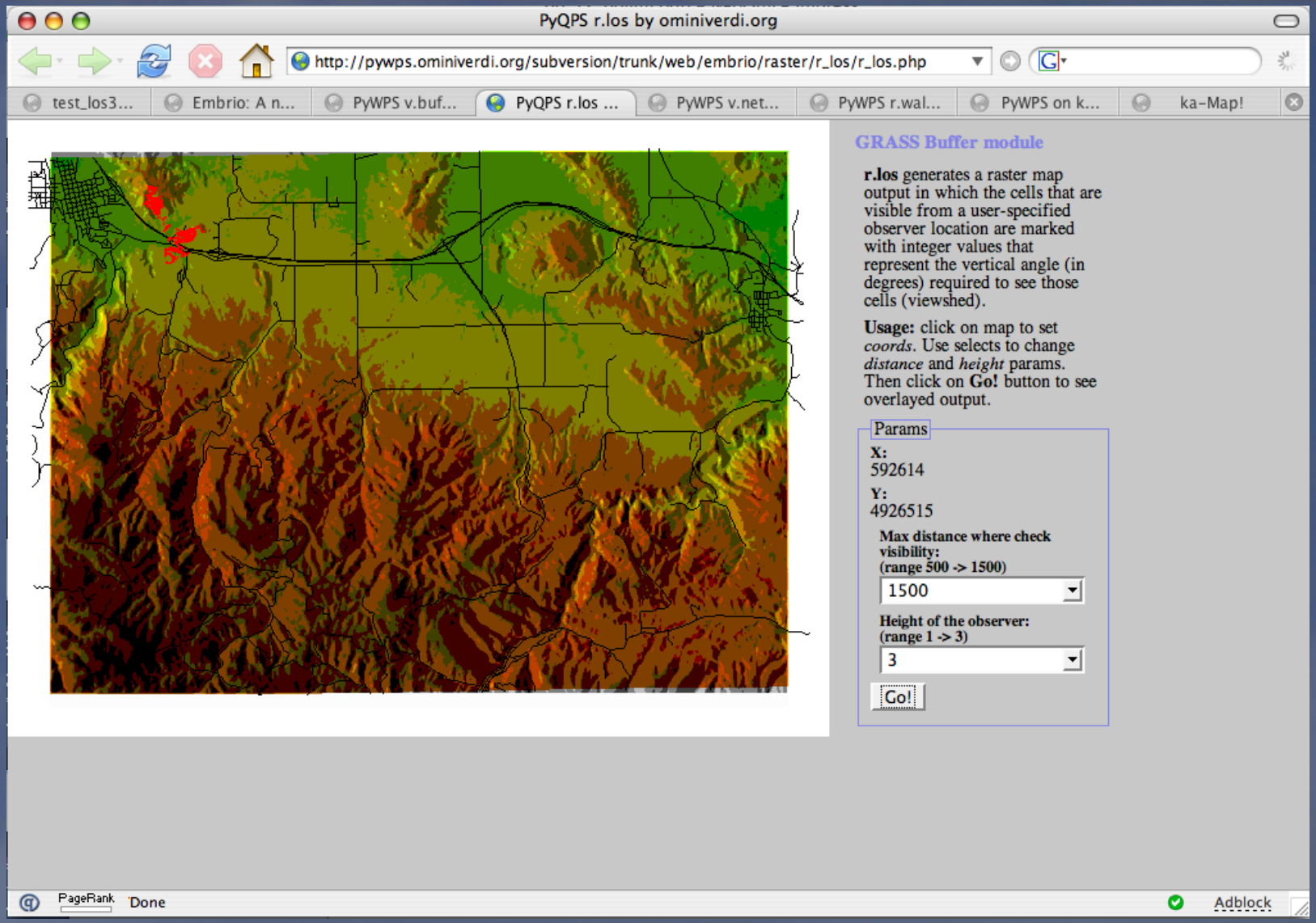
- **doktoreas**
- **ominiverdi**

PageRank Done Adblock

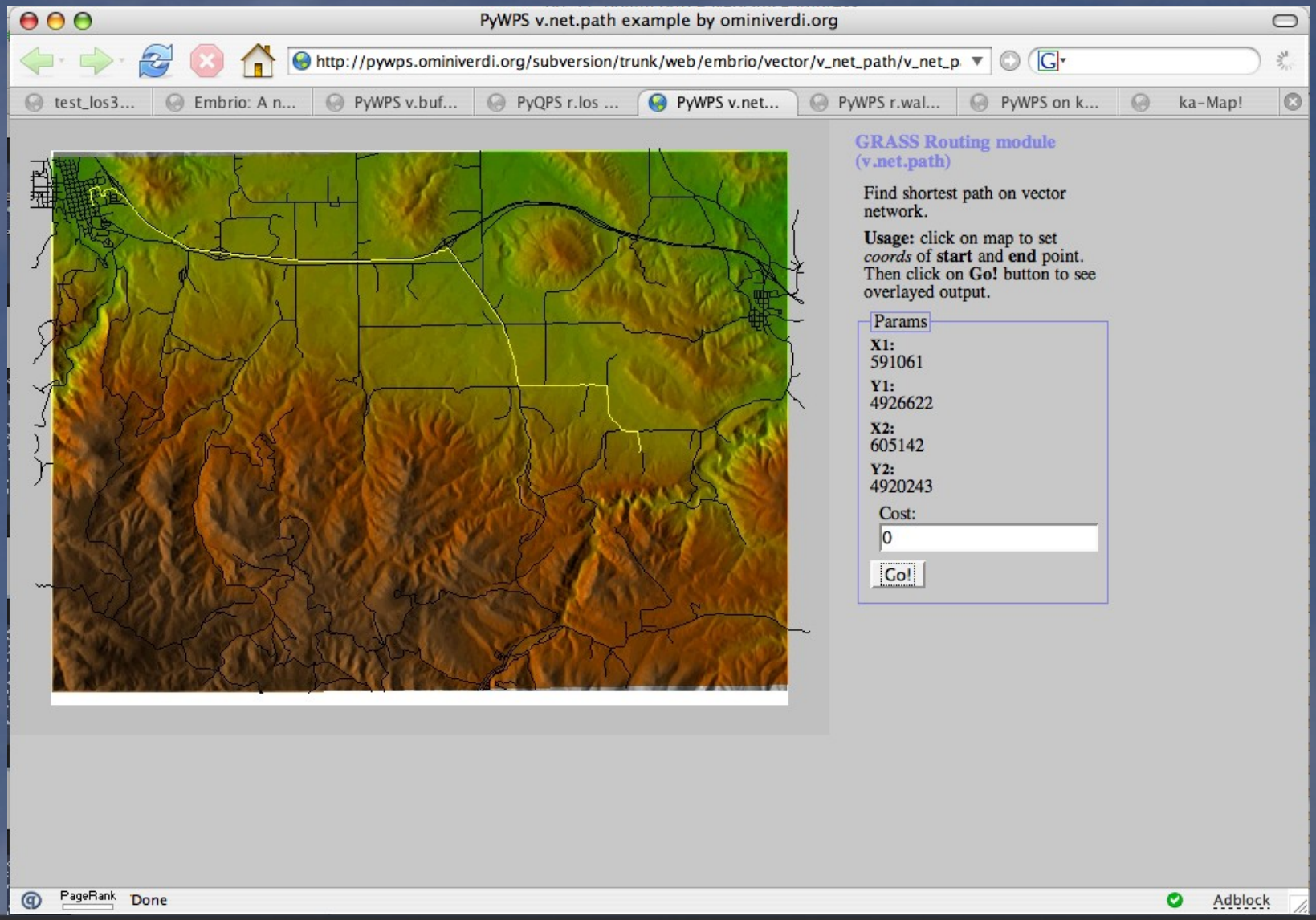
Embrio - V.Buffer



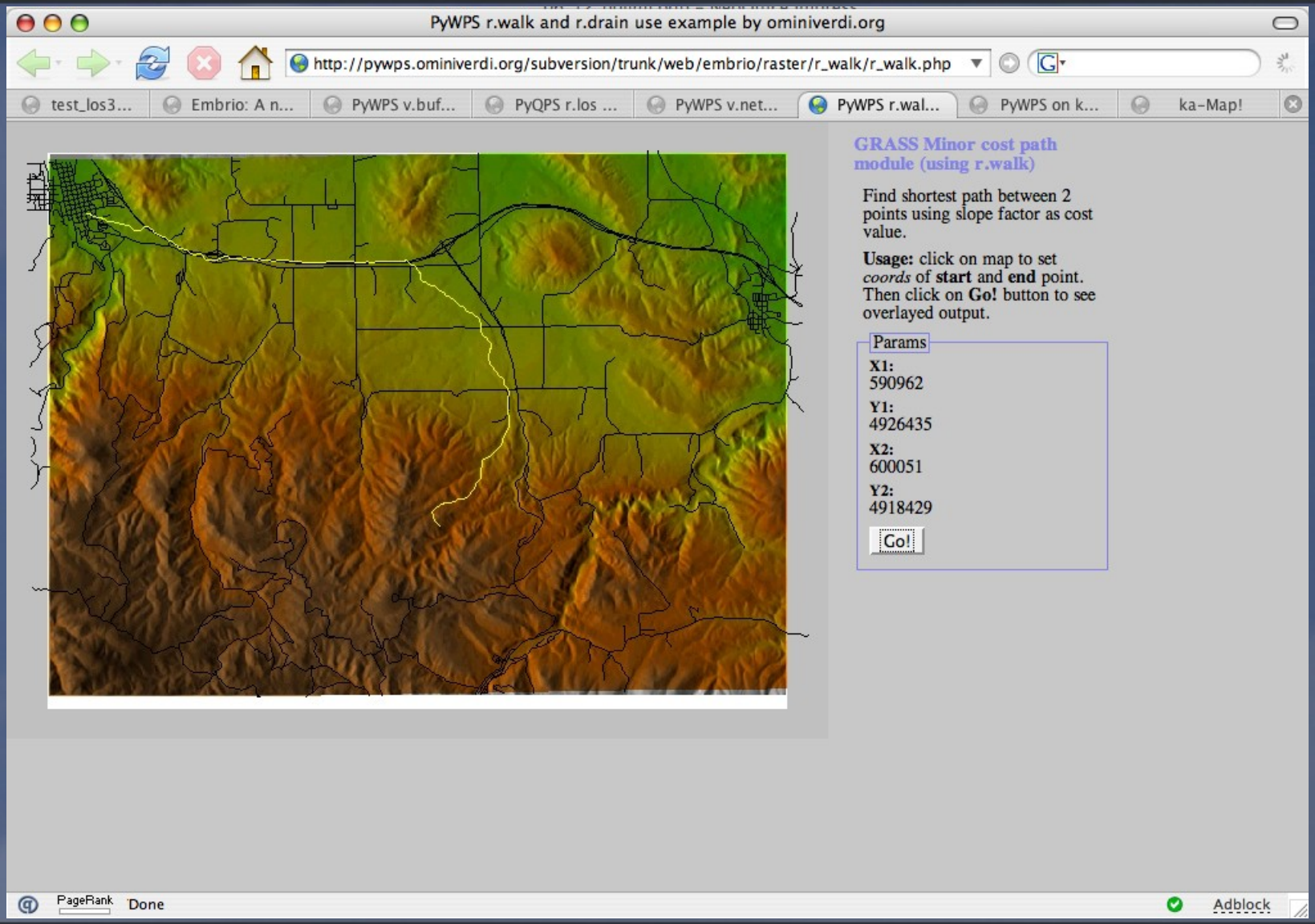
Embrio - R.Los



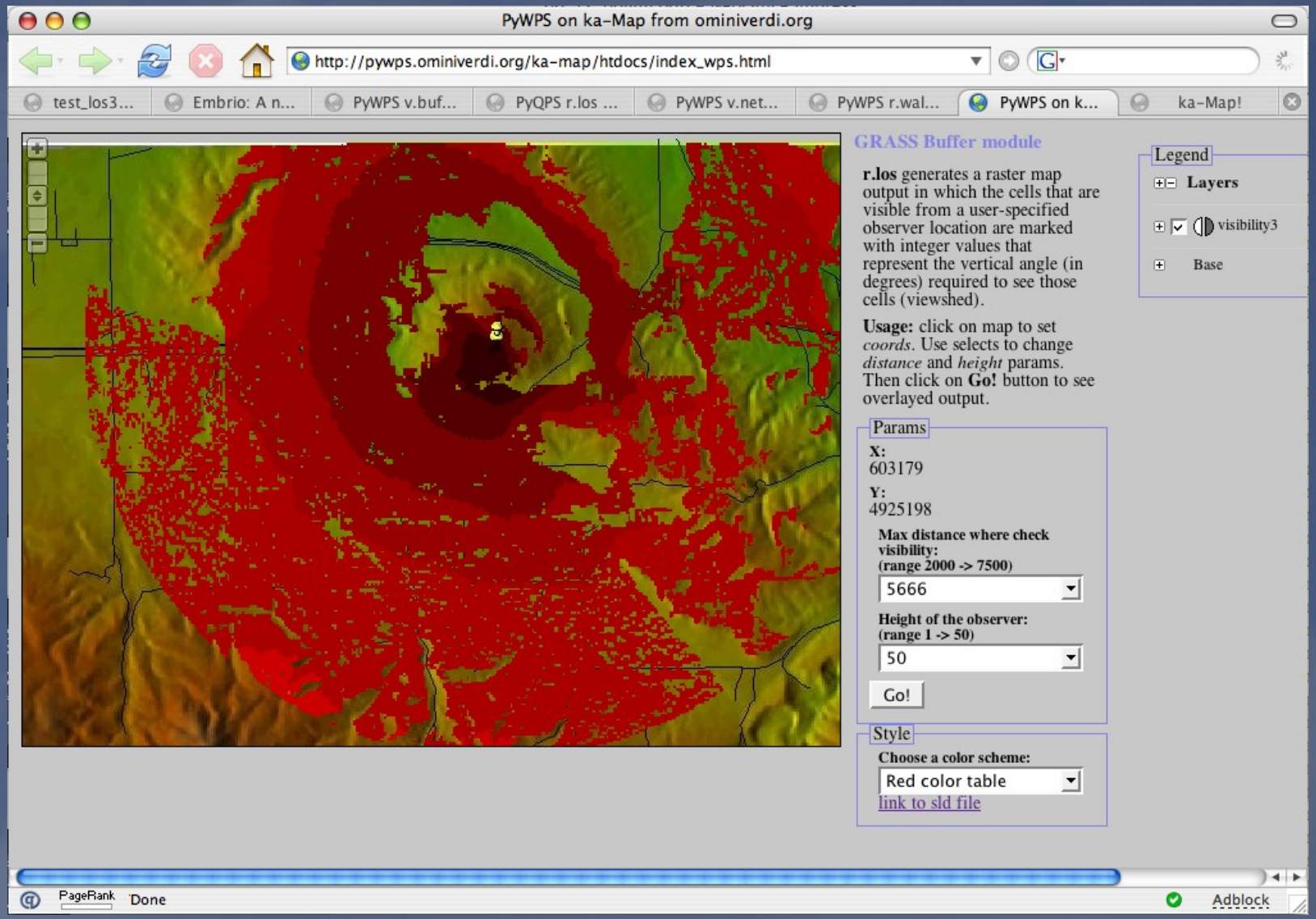
Embrio - V.Net.Path



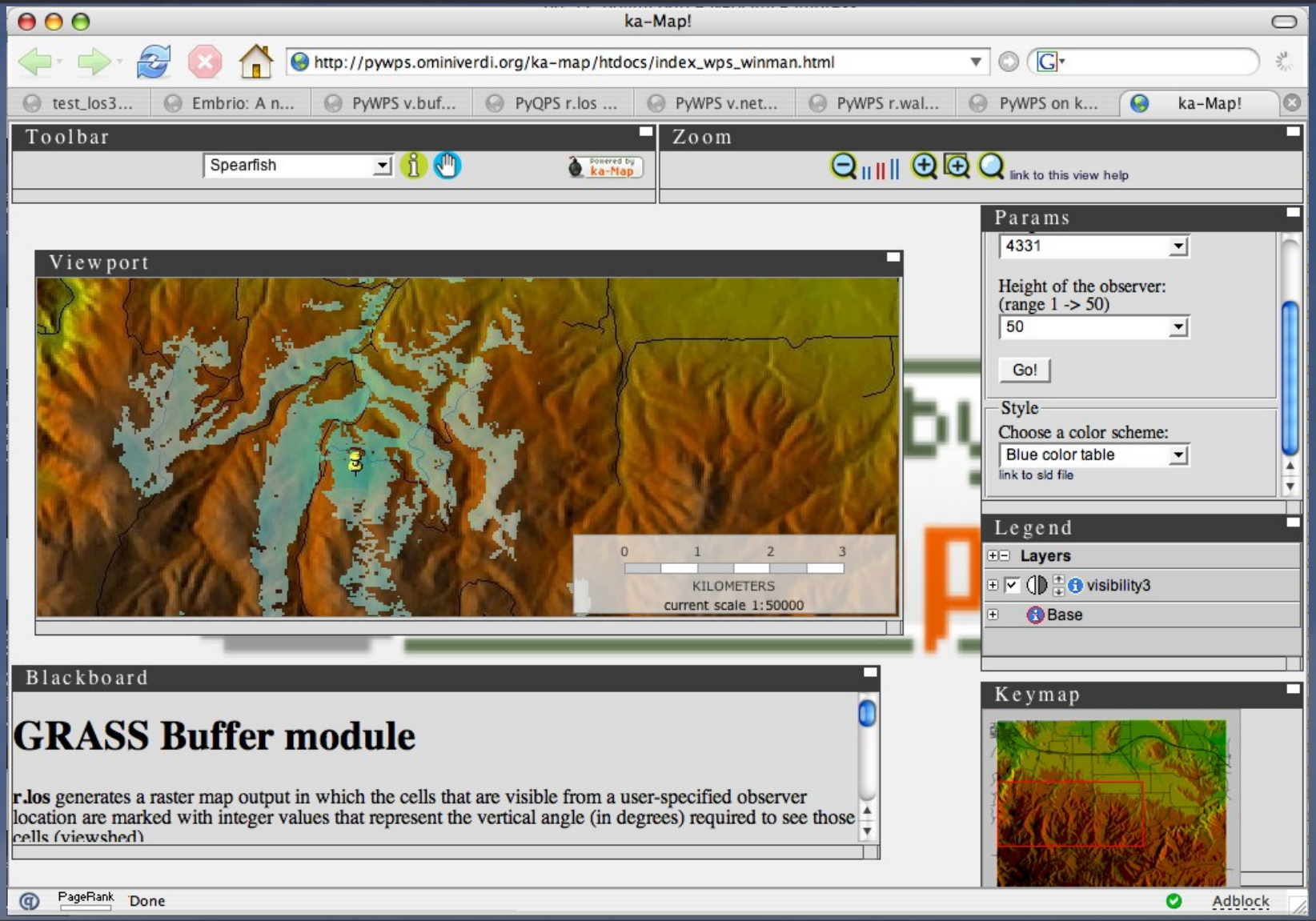
Embrio – R.Walk



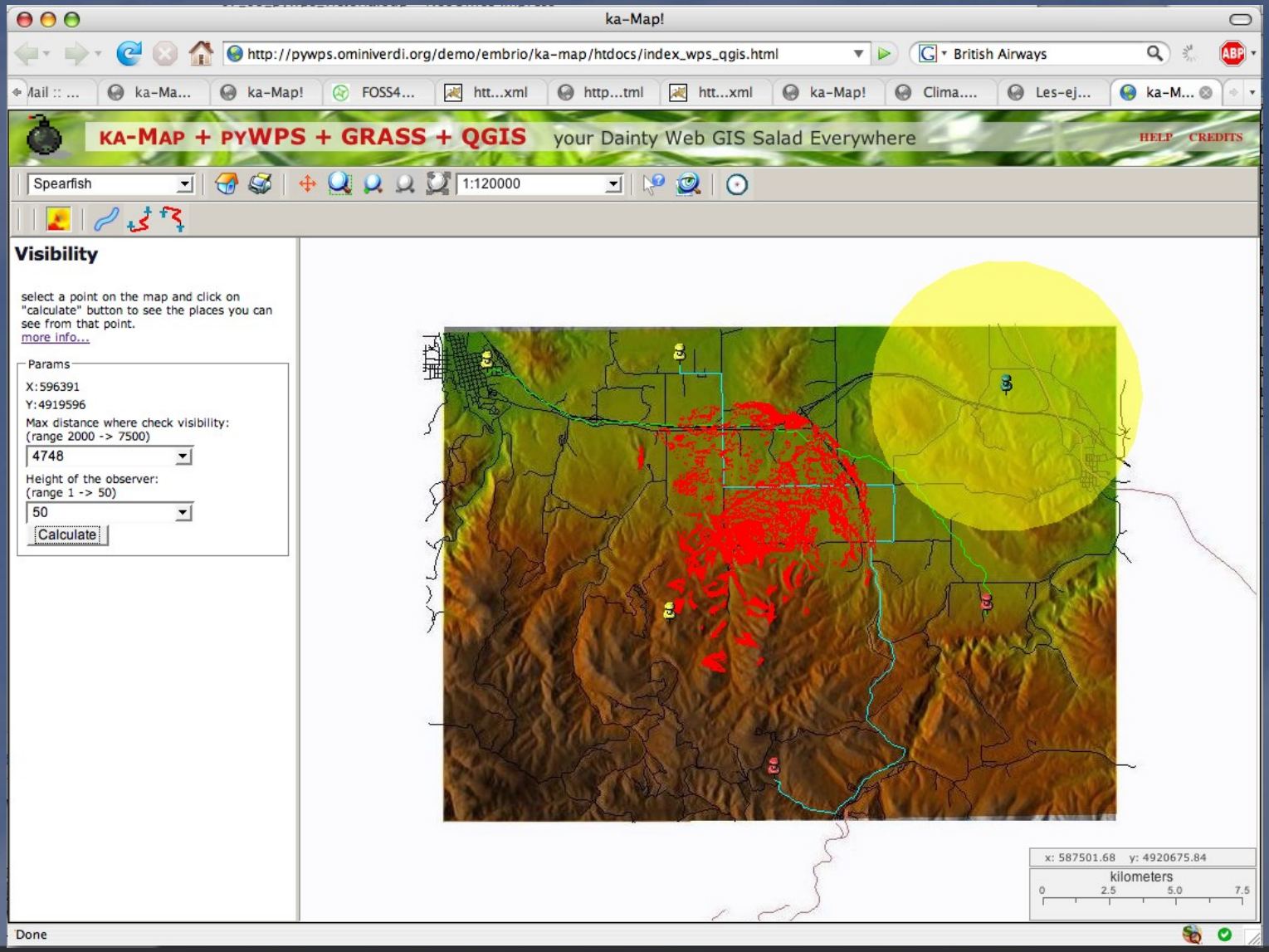
Embrio + ka-Map - R.Los



Embrio + ka-Map + Winman - R.Los



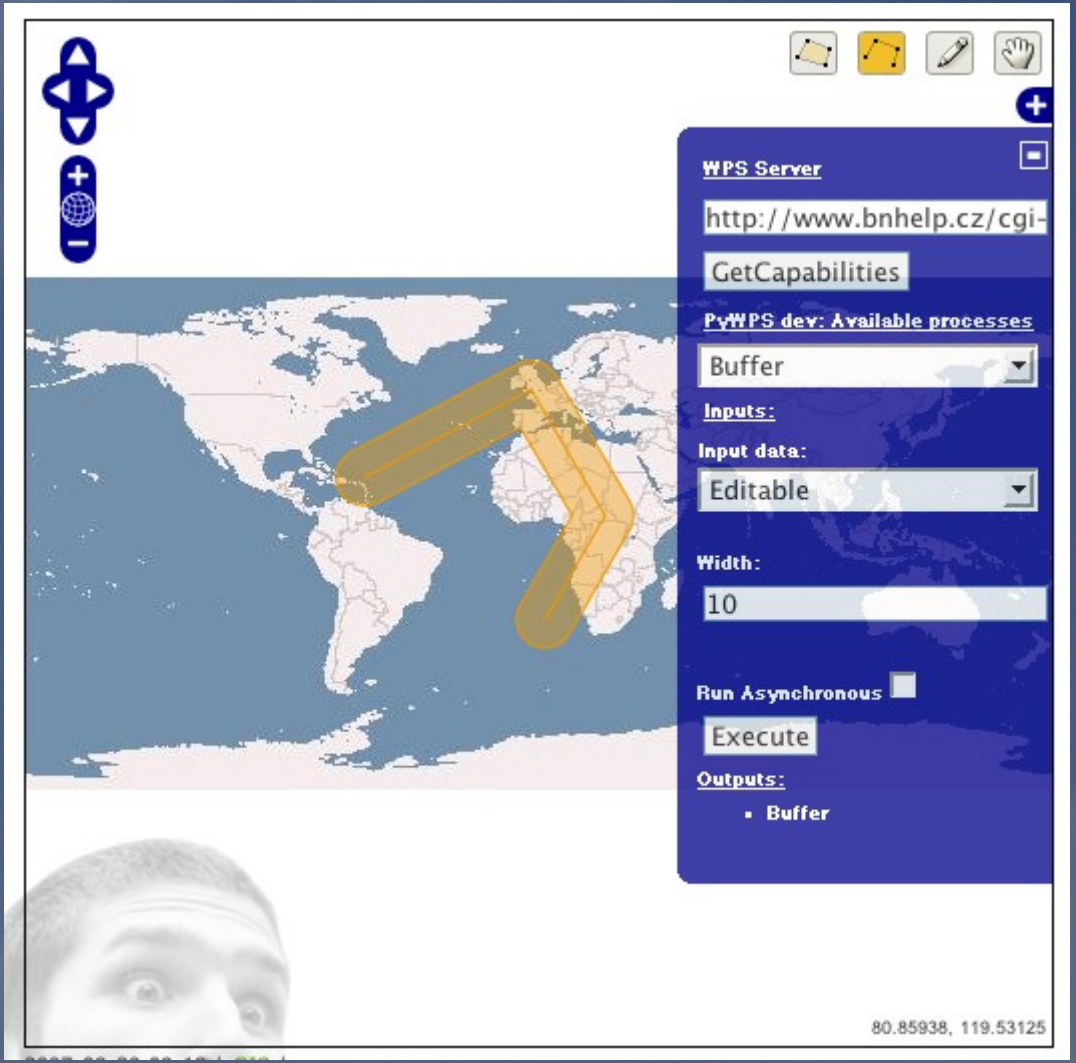
Embrio – last prototype



PyWPS in action - WUIW Demo

- http://www.les-ejk.cz/english/pywps-and-openlayers
  - Code status: Alpha
  - Developers:
    - Jachym Cepicky (ominoverde)
  - Based on OpenLayers API

PyWPS in action - WUIW Demo



Open Geospatial Consortium, Inc (OGC)

- International voluntar y consensus standards organization
- Development and implementation of standards for geospatial content and services
- Active Members: 350 (Sep 2007) – University of Minnesota, US National Oceanic and Atmospheric Administration (NOAA) Coastal Service Center, ESRI, Autodesk Inc., MIT, . . .
- http://www.opengeospatial.org/

Web service

- Software system designed to support interoperable machine-to-machine interaction over a network (Wikipedia)
- In OGC terminology, "Service" refers to a processing task that is invoked by a client and executed by a server, usually across a network.
- The OpenGIS Specifications that make this possible are referred to as "OGC Web Services".
- OpenGIS Web Service (OWS):
  - OpenGIS Catalog Service (CAT)
  - OpenGIS Web Coverage Service (WCS)
  - OpenGIS Web Feature Service (WFS)
  - OpenGIS Web Map Service (WMS)
  - . . .
  - Web Processing Service (WPS) (draft)

OpenGIS Web Processing Service

- Document OGC 05-007r4, version 0.4.0
- Not yet OGC standard, "Discussion Paper", Draft
- To offer any sort of GIS functionality to clients across a network
- XML-based communication protocol
-
- Es: http://pywps.ominiverdi.org/cgi-bin/wps?service=WPS&version=0.4.0&request=...

WPS request=GetCapabilities

http://www.bnhelp.cz/cgi-bin/wps?service=WPS&version=0.4.0&request=GetCapabilities

---

```
<?xml version="1.0" ?>
    <Capabilities version="0.4.0" ... >
        <ows:ServiceIdentification>
        <ows:Title>Sample WPS server</ows:Title>
        <ows:Abstract>WPS for Lausanne</ows:Abstract>
        <ows:ServiceType>WPS</ows:ServiceType>
        <ows:Fees>free</ows:Fees>
        </ows:ServiceIdentification>
        <ows:ServiceProvider>
            <ows:ProviderName>GDF</ows:ProviderName>
            <ows:ServiceContact>
                <ows:IndividualName>Jachym Cepicky</ows:IndividualName>
                <ows:PositionName>Student</ows:PositionName>
                ...
            </ows:ServiceContact>
        </ows:ServiceProvider>
...
```

WPS request=GetCapabilities


```
...
<ProcessOfferings>
    <Process processVersion="0.1">
        <ows:Identifier>addvalue</ows:Identifier>
        <ows:Title>Add some value to raster map</ows:Title>
    </Process>
    <Process processVersion="0.1">
        <ows:Identifier>classify</ows:Identifier>
        <ows:Title>Image classification</ows:Title>
        <ows:Abstract>
            GRASS processed imagery
            classification. Only unsupervised is supported at the moment.
        </ows:Abstract>
    </Process>
    <Process processVersion="0.1">
        <ows:Identifier>shortestpath</ows:Identifier>
        <ows:Title>Shortest path</ows:Title>
    </Process>
</ProcessOfferings>
```

WPS request=DescribeProcess

http://www.bnhelp.cz/cgi-bin/wps.py?service=WPS&version=0.4.0&
request=DescribeProcess&identifier=addvalue

```
<?xml version="1.0" ?>
    <ProcessDescriptions ...>
        <ProcessDescription ...>
            <ows:Identifier>addvalue</ows:Identifier>
            <ows:Title>Add value</ows:Title>
            <ows:Abstract>Adds some value to each cell of input raster map</ows:Abstract>
            <DataInputs>
                <Input>
                    <ows:Identifier>value</ows:Identifier>
                    <ows:Title>Value to be added</ows:Title>
                    <LiteralData>
                        <AllowedValues>
                            <Value>1</Value>
                            ...
                        </AllowedValues>
                        <ows:DefaultValue>10</ows:DefaultValue>
                    </LiteralData>
                </Input>
...
```

WPS request=DescribeProcess

```
            <Input>
                <ows:Identifier>map</ows:Identifier>
                <ows:Title>Input raster map</ows:Title>
                <ComplexData defaultFormat="image/tiff">
                </ComplexData>
            </input>
        </DataInputs>
        <ProcessOutputs>
            <Output>
                <ows:Identifier>value</ows:Identifier>
                <ows:Title>literal value + 1</ows:Title>
                <LiteralOutput> ... </LiteralOutput>
            </Output>
            <Output>
                <ows:Identifier>map</ows:Identifier>
                <ows:Title>Resulting output map</ows:Title>
                <ComplexOutput defaultFormat="image/tiff">
                    ...
                </ComplexOutput>
            </Output>
        </ProcessOutputs>
    </ProcessDescription>
</ProcessDescriptions>
```

WPS request=Execute

http://www.bnhelp.cz/cgi-bin/wps.py?service=WPS&version=0.4.0&
request=Execute&identifier=addvalue&DataInputs=value,5,map,http://localhost/data/soils.tif

---

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <Execute service="wps" version="0.4.0" store="true" status="false"
            xmlns="http://www.opengeospatial.net/wps"
            xmlns:ows="http://www.opengeospatial.net/ows">
        <ows:Identifier>addvalue</ows:Identifier>
        <DataInputs>
            <Input>
                <ows:Identifier>value</ows:Identifier>
                <LiteralValue>5</LiteralValue>
            </Input>
            <Input>
                <ows:Identifier>map</ows:Identifier>
                <ComplexValueReference  reference="http://localhost/data/soils.tif" />
            </Input>
            ...
        </DataInputs>
    </Execute>
```

WPS Response XML


```
<?xml version="1.0" ?>
    <ExecuteResponse ...>
        <ows:Identifier>addvalue</ows:Identifier>
        <Status>
            <ProcessSucceeded/>
        </Status>
        <ProcessOutputs>
            <Output>
                <ows:Identifier>value</ows:Identifier>
                <ows:Title>literal value + 1</ows:Title>
                <LiteralValue>6</LiteralValue>
            </Output>
            <Output>
                <ows:Identifier>value</ows:Identifier>
                <ows:Title>Resulting output map</ows:Title>
                <ComplexValueReference format="image/tiff"
        ows:reference="http://www.bnhelp.cz/wpsoutputs/output2-2006-8-21-14-54-42.tif"
                />
            </Output>
        </ProcessOutputs>
    </ExecuteResponse>
```

PyWPS 2.0

- Implementation OGC's WPS standard (90-95 %)
- CGI Application
- Python programming language

PyWPS 2.0

- PyWPS starts as dedicated connector to GRASS
  - CLI
  - More than 300 modules for raster and vector analysis
  - GNU/GPL
  - GRASS Functionality can be via PyWPS offered in Internet
- User does not need Desktop-GIS (GRASS, ESRI, Idrisi, ...) – Web browser becomes GIS
- One can use other CLI-oriented programs (PROJ.4, GDAL, R, ...)

PyWPS 2.0 – Execute – how it works

- Controlling input data, if all necessar y parameters have arrived (Identifier, DataInputs,. . .)
- Loading process, for each input:
  - LiteralValue: Controlling, if input fits AllowedValues array
  - ComplexValue: Embed input files will be extruded from input XML request into separate files
  - ComplexValueReference: Tries to download the data from external source and stores it to new file
  - BoundingBoxValue
- If some DataInput is missing, it looks for the default value value

PyWPS 2.0 – Execute – how it works

- Creates temporar y GRASS Location or just temporar y Mapset within existing location, which will be deleted, after the work is done
- Calls function execute() of the process
- Formulates output XML file
- Deletes temporar y files (location, mapset, pid file)
- Returns output XML or resulting map file (TIFF, GML) to the client
- Process can be run asynchronously: After the request is accepted, XML responce is imidietly returned with <ProcessAccepted /> element and the calculation is forked to background.

PyWPS 2.0 - Addvalue – Sample process

- Inputs
  - Literal input: il valore da aggiungere
  - ComplexValueReference input: una mappa raster
- Outputs
  - Literal output: Input+1
  - ComplexValueReference: la mappa risultante (GeoTIFF)

PyWPS 2.0 – Sample process

```
class Process:
  def __init__(self):
        self.Identifier = "addvalue"
        self.Title="Sample process for demonstration purposes"
        self.Inputs = [ { # 0
                              'Identifier':'value',
                              'Title': 'Value to added',
                              'LiteralValue': {'values':[0,1,2,3,4,5]},
                              'dataType': type(0),
                              'value':0, # default},
                        { #1
                              'Identifier': 'map',
                              'Title': 'The raster map',
                              'ComplexValueReference': {'Formats':["image/tiff
                        }... ]
        self.Outputs = [ { # 0
                              'Identifier': 'value',
                               'Title': 'Input value + 1',
                              'LiteralValue': {},
                              'value':1 },... ]
```

PyWPS 2.0 – Sample process

```
29          def execute(self):
30              self.status = ["The start", 5]
31              self.Outputs[0]['value'] = self.Inputs[0]['value']+1
32              self.status = ["LiteralValue set", 20]
33
34          self.status = ["Data import", 25]
35          os.system("r.in.gdal in=%s out=map" % (self.Inputs[1]['value'
36
37          self.status = ["Creating output map", 50]
38          os.system("r.mapcalc map=map+%d" % (self.Inputs[0]['value']))
39
40          self.status = ["Exporting map", 75]
41          if os.system("r.out.gdal in=map out=output.tif type=UInt16 >
42          return "Could not export map"
43          else: # ok
44          return
```

Conclusions

- WPS Standard implemented to usable degree
- Making GRASS scripts run via web-interface was never easier
- It is relatively simple to connect UMN MapServer (or ARC IMS) with GRASS via PyWPS. Fur ther GRASS development will make this even easier

Sviluppi futuri

- PyWPS
- Process definition (data inputs and outputs) is primitive – build set of classes for process definition
- GRASS
  - Implementation of new GRASS-python interface (Alessandro Frigeri aka ’geoalf ’)
  - 3D views via VTK (Sören Gebbert aka ’huhabla’)
- Embrio & Wuiw
  - Road to stable versions

Attribution-NonCommercial-ShareAlike 2.5

creative commons

Attribution-NonCommercial-ShareAlike 2.5

You are free:

- **to Share** -- to copy, distribute, display, and perform the work
- **to Remix** -- to make derivative works

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor.

Noncommercial. You may not use this work for commercial purposes.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Disclaimer

Your fair use and other rights are in no way affected by the above.
This is a human-readable summary of the Legal Code (the full license).