

Scripting in gvSIG 2.3: New features, improvements and enhancements

Óscar Martínez
Twitter: @masquesig
Email: omartinez@gvsig.com



Scripting Module

The image shows the gvSIG Scripting Module interface. On the left, a sidebar titled "Scripting La..." lists script categories under "Usuario" and "Sistema". The "Usuario" section includes "Scripts del usuario" with sub-folders like "Formulariosnull", "Guía", "Jornadas", "MDTArtinull", and "outreachnull", each containing various Python scripts. The "Sistema" section lists system scripts such as "003-creargeometria", "1current", "1get", "9Jornadas", "addinfo", "areafin", "baseCrearCapa", "baseCrearCapaBeta", "buffer", "bufferLinea", "capaCentroides", "capaSelect", "centroides", "codigoDistancia", "commonsdialog", "containsEntidad", and "convexHull".

The main window is titled "Scripting Composer" and contains a "File Edit Tools Help" menu and a toolbar with icons for file operations. It features two tabs: "1get" and "addinfo", with "baseCrearCapa" selected. The code editor displays the following Python script:

```
1 from gvsig import *
2 from geom import *
3
4
5 def crearCapa():
6     ruta = "C:/gvsig/prjavageom6.shp"
7     CRS = currentProject().getProjectionCode()
8     schema = createSchema()
```

The "Properties" tab is visible below the code editor. The "Console" tab shows the output of running the script:

```
Welcome to ScriptingFramework v1.0
Running script baseCrearCapa.
Script baseCrearCapa terminated.
```

Line 12:5

At the bottom right, there is a separate window titled "Jython" showing the "Jython Completion Shell" with the following text:

```
Jython
Jython 2.5.2 (Release_2_5_2:7206, Mar 2 2011, 23:12:06)
[Java HotSpot(TM) Client VM (Sun Microsystems Inc.)] on
>>> |
```



Scripting Module

- Integrated programming module (IDE)
- Create & Execute & Organize scripts
- Different allowed languages:
 - Jython (Java + Python)
 - Groovy
 - R
 - .. and more



Scripts

- Scripts for:
 - ... create new geoprocess
 - ... automate tasks
 - ... develop new tools
 - ... add visual interface (Abeille)
- Access to all gvSIG functions



Scripting in gvSIG

Useful for:

Users

&

Developers



New features



Jython

- Updated version Jython 2.7.1 beta3
more info: <http://fwierzbicki.blogspot.com.es/>
- Full compatibility with Java libraries
... and some of Python (without C)
 - add Python libs to:
gvsig\gvSIG\extensiones\org.gvsig.scripting.app.mainplugin\scripting\lib
 - or jars to:
gvsig\gvSIG\extensiones\org.gvsig.scripting.app.mainplugin\lib
- Future: JyNI (compatibility with Numpy and with almost all Python libraries)
more info: <http://jyni.org/>



R

- R through Renjin

- more info: <http://www.renjin.org/>

Renjin is a JVM-based interpreter for the R language

- Full integrated with Java/gvSIG
- Not full integrated with R packages

- R native

- Full integrated with R
- Not full integrated with Java/gvSIG
- One installation of R will come with gvSIG

```
1 import (org.gvsig.app.project.documents.view.ViewDocument)
2 import (org.gvsig.app.ApplicationLocator)
3 import (org.gvsig.fmap.mapcontext.MapContext)
4 main <- function() {
5
6   ....cat(`View.access\n`.)
7   ....application <- ApplicationLocator$getManager()
8
9   ....view <- application$getActiveDocument(ViewDocument);
10  ....print(view$name);
11  ....prj <- view$projection;
12  ....print(prj);
13  ....crs <- prj$fullCode;
14  ....print(crs);
15  ....mapcontext <- view$mapContext;
16  ....layers <- mapcontext$layers;
17  ....layer <- layers[[1]];
....}
```

GDAL

- GDAL integrated in gvSIG

..so we can use gdal in our scripts

<http://gdal.org/java/>

- Also, we could use:

- Ogr2ogr
- Gdaltranslate

```
def main(*args):  
  
    """ogr2ogr from gdal"""  
    |  
    ogr2ogr.main(["-t_srs", "CRS:84",  
    ....."-f",  
    ....."ESRI Shapefile",  
    ....."C:/temp/j1",  
    ....."C:/temp/countries.geojson",  
    ....."-overwrite",  
    ....."-skipfailures"  
    ..])  
  
    ...  
  
    gdalapp("ogr2ogr -t_srs 'CRS:84' -f 'ESRI S
```

```
1  from gvsig import *  
2  import uselib  
3  uselib.use_plugin("org.gvsig.gdal.app.mainplugin")  
4  
5  from org.gdal.gdal import gdal  
6  from org.gdal.ogr import ogr  
7  from org.gdal import osr  
8  
9  def main(*args):  
...  #Book: Python Geospatial Analysis Cookbook  
11  gdal.AllRegister()  
12  
13  #.get.raster.data.source  
14  rasterfile = r"C:\gvdata\data_example\mdt_jaen.tif"  
15  open_image = gdal.Open(rasterfile)  
16  input_band = open_image.GetRasterBand(1)  
17  
18  #.create.output.data.source  
19  output_shp = getTempFile("newshp", "")  
20  shp_driver = ogr.GetDriverByName("ESRI Shapefile")  
21  
22  #.create.output.file.name  
23  output_shapefile = shp_driver.CreateDataSource(output_shp + '')  
24  
25  sp = osr.SpatialReference('PROJCS["ETRS89 / UTM zone 30N",GEO  
26  
27  new_shapefile = output_shapefile.CreateLayer(output_shp, sp)  
28  #mask_band = input_band.GetMaskBand()
```

Geoprocess

Call geoprocess that already exist inside gvSIG

Área de influencia

Parámetros

Entrada

Opciones
 Área definida por un campo en metros

 Área definida por una distancia en metros

 Geometrías seleccionadas
 Disolver entidades (solo un anillo)
 Borde redondeado
 Fuera del polígono

```
1  from gvsig import *
2  import gvpypy
3
4
5
6  def main(*args):
7
8      ... a ... = gvpypy.runalg("randomvector", -COUNT=10, -TYPE=2)
9
10     ... b ... = gvpypy.runalg("voronoi", -a)
11     ... pass
12
13
```



< [63] runalg("gvSIG-buffer", "Arment_Halt") >

Aceptar

Cancelar

i



Addons

Create Package (Addons) from Scripts

The screenshot illustrates the gvSIG interface for creating an addon package from scripts. It shows three main windows:

- Top Window:** A file browser titled "Editor de scripts" showing a directory structure under "Usuario \ Sistema \ Usuario". It lists several folders and files, including "coordenadas", "coordenadas", "libs", "sqlconsole", "gr1", "JS js1", and various "test" files.
- Middle Window:** A dialog titled "install_package" with the sub-titile "Selección paquetes". It displays a list of packages:

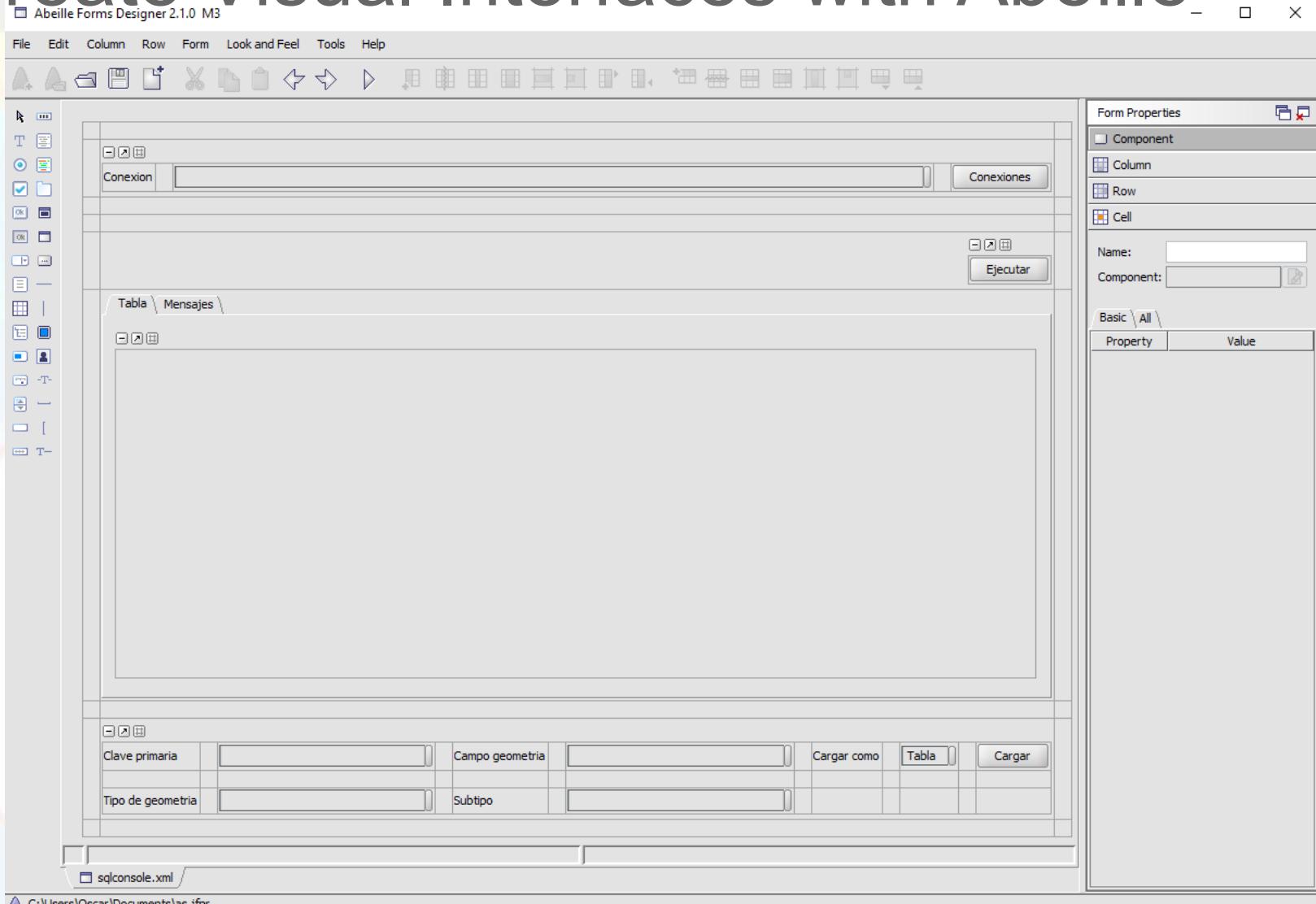
Todos	Nombre	Versión	Tipo
Script	coordenadas	1.0.0-0	Script
Plugin	SQL Console	1.0.0-0	Script
- Bottom Window:** A dialog titled "Descripción del paquete" for the "coordenadas" package. The fields are as follows:

Type	Script
Code	coordenadas
Name	coordenadas
Description	(empty)
Categories	(empty)
Version	1.0.0-0
Build	0
State	testing
Official	<input type="checkbox"/>
Source code URL	(empty)
Package owner	(empty)
Package owner URL	(empty)
Operating System	all



Visual Interfaces

Create Visual Interfaces with Abeille



The screenshot shows the Abeille Forms Designer 2.1.0 M3 application window. The menu bar includes File, Edit, Column, Row, Form, Look and Feel, Tools, and Help. The toolbar contains various icons for file operations and form design. The main workspace displays a form layout with several components:

- A top section labeled "Conexion" with a text input field and a "Conexiones" button.
- A middle section labeled "Tabla \ Mensajes" containing a large empty area.
- A bottom section with two rows of input fields:
 - Row 1: "Clave primaria" (Primary Key) and "Campo geometria" (Geometry Field).
 - Row 2: "Tipo de geometria" (Geometry Type) and "Subtipo" (Subtype).
- Buttons: "Cargar como" (Load as), "Tabla" (Table), and "Cargar" (Load).

The right side of the interface features the "Form Properties" panel, which lists Component, Column, Row, and Cell options. It also includes a "Basic \ All \ Property" table with columns for Name, Component, and Value. The status bar at the bottom shows the file path "C:\Users\Oscar\Documents\las.ifor" and the file name "sqlconsole.xml".

Visual Interfaces

New lib development “FormPanel”

- Create easily visual interfaces

- Assign actions to buttons
(action click)

- Manage listeners
(action change)

```
1  from gvsig import *
2  from libs.formpanel import FormPanel
3
4
5  class Panel(FormPanel):
6      def __init__(self):
7          FormPanel.__init__(self, script.getResource("ui_basic.xml"))
8
9      def btnCalcular_click(self, *args):
10         self.textField.setText("Clicked!")
11         print "Clicked!"
12
13     def btnClose_click(self, *args):
14         self.hide()
15
16     def spinnerValue(self, *args):
17         print "Value has change!"
18
19     def main(*args):
20         l = Panel()
21         l.showTool("Visual")
22         pass
```

Composer

IDE Improvements

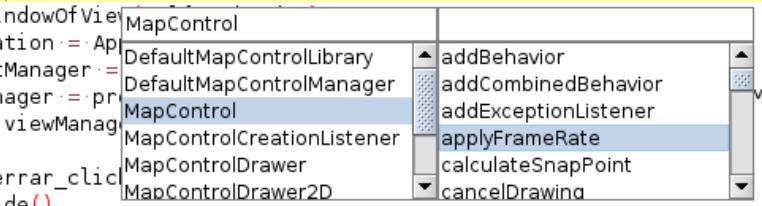
Autocomplete
 (New: also with variables
 names)

Tree scripts

```

13 class Coordenadas(FormPanel, PointListener):
14     def __init__(self, viewdoc):
15         FormPanel.__init__(self, script.getResource("coordenadas.xml"))
16         self.viewdoc = viewdoc
17         self.mapControl = self.getWindowOfView(self.viewdoc).getMapControl()
18         self.mapControl.addBehavior("TestGetXYPointTool", PointBehavior(self))
19         self.mapControl.setTool("TestGetXYPointTool")
20         self.mapControl.
21         def getWindowOfView(self):
22             application = Application.getInstance()
23             projectManager = application.getProjectManager()
24             viewManager = projectManager.getViewManager()
25             return viewManager
26
27     def btnCerrar_click(self, event):
28         self.hide()
29
30     def point(self, event):

```



A screenshot of the gvSIG IDE interface. On the left, there is a code editor window displaying Python-like code. In the middle, a tooltip-style completion dropdown is open over the word 'mapControl'. The dropdown lists several methods and properties of the 'MapControl' class, such as 'addBehavior', 'addCombinedBehavior', 'addExceptionListener', 'applyFrameRate', 'calculateSnapPoint', and 'cancelDrawing'. The method 'applyFrameRate' is currently highlighted.

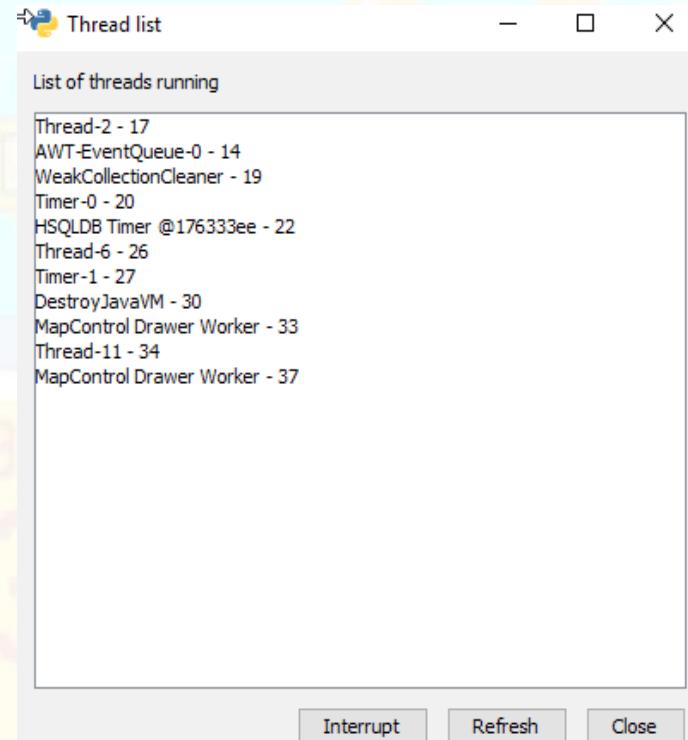




Composer

IDE Improvements

- Threads Killer
- Code navigator





Composer

Replace

The image shows the gvSIG Composer interface. On the left is a file browser window titled "Archivo" showing a tree view of "Usuario \ Sistema \". In the center is the "Editor de scripts" window titled "coordenadas". The code editor contains a Python script:

```
1 execfile(script.getResource("../libs/import_utils.py").ge
2
3
4 from gvsig import *
5
6 from org.gvsig fmap import IconThemeHelper
7 from org.gvsig fmap.mapcontrol.tools.Listeners import Point
8 from org.gvsig fmap.mapcontrol.tools.Behavior import Point
9 from org.gvsig app.project.documents.view import ViewManager
10
11 import_from_module("../libs.formpanel", "FormPanel")
```

Below the code editor is a search and replace dialog titled "Reemplazar" with fields for "Buscar" (mapcontrol), "Reemplazar con" (empty), and checkboxes for "Coincidir mayúsculas/minúsculas", "Expresión", "Palabra completa", "Marcar Todo", "Arriba", and "Abajo". At the bottom of the interface, two status bars show "Text found; occurrences marked: 6" and "Línea 7:30".



Composer

Search

The screenshot shows the gvSIG Composer interface. On the left is a file browser window titled "Archivo" with a tree view of files and folders under "Usuario \ Sistema \ Usuario". The "coordenadas" folder is selected. On the right is the "Editor de scripts" window, which has two panes. The left pane shows a script named "coordenadas" with the following code:

```
1 execfile(script.getResource("../libs/import_utils.py").getAbsolutePath())
2
3 from gvsig import *
4
5 from org.gvsig.fmap import IconThemeHelper
6 from org.gvsig.fmap.mapcontrol.tools.Listeners import PointListener
7 from org.gvsig.fmap.mapcontrol.tools.Behavior import PointBehavior
8 from org.gvsig.app.project.documents.view import ViewManager
9
10 import_from_module("../libs.formpanel","FormPanel")
11
12
13 class Coordenadas(FormPanel.PointListener):
14     pass
```

The right pane shows the output of running the script: "Running script coordenadas. Script coordenadas terminated." Below the editor is a search dialog with the word "fmap" entered in the search field.



Toolbox and Buttons

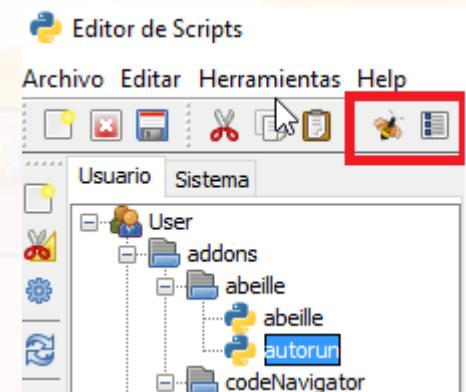
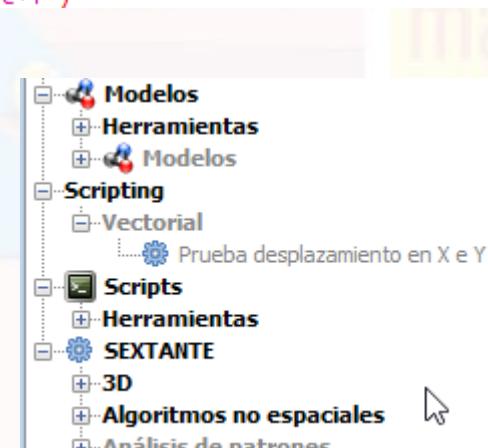
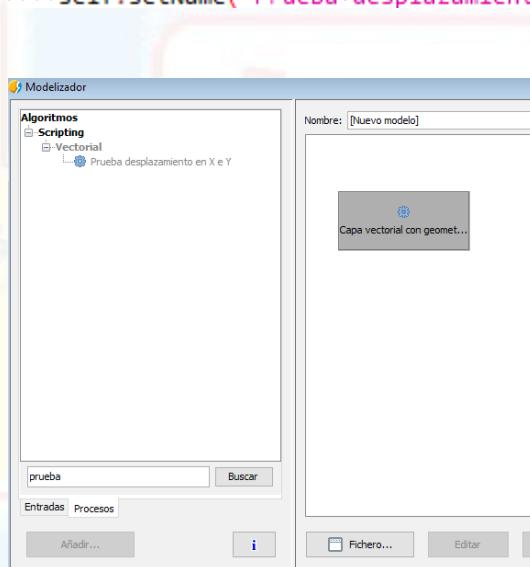
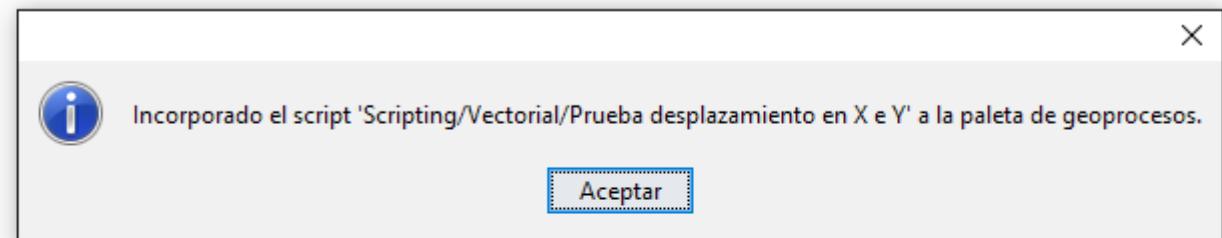
Add geoprocess to the Toolbox or as Buttons

```
import_from_module("../libs.toolbox","*")

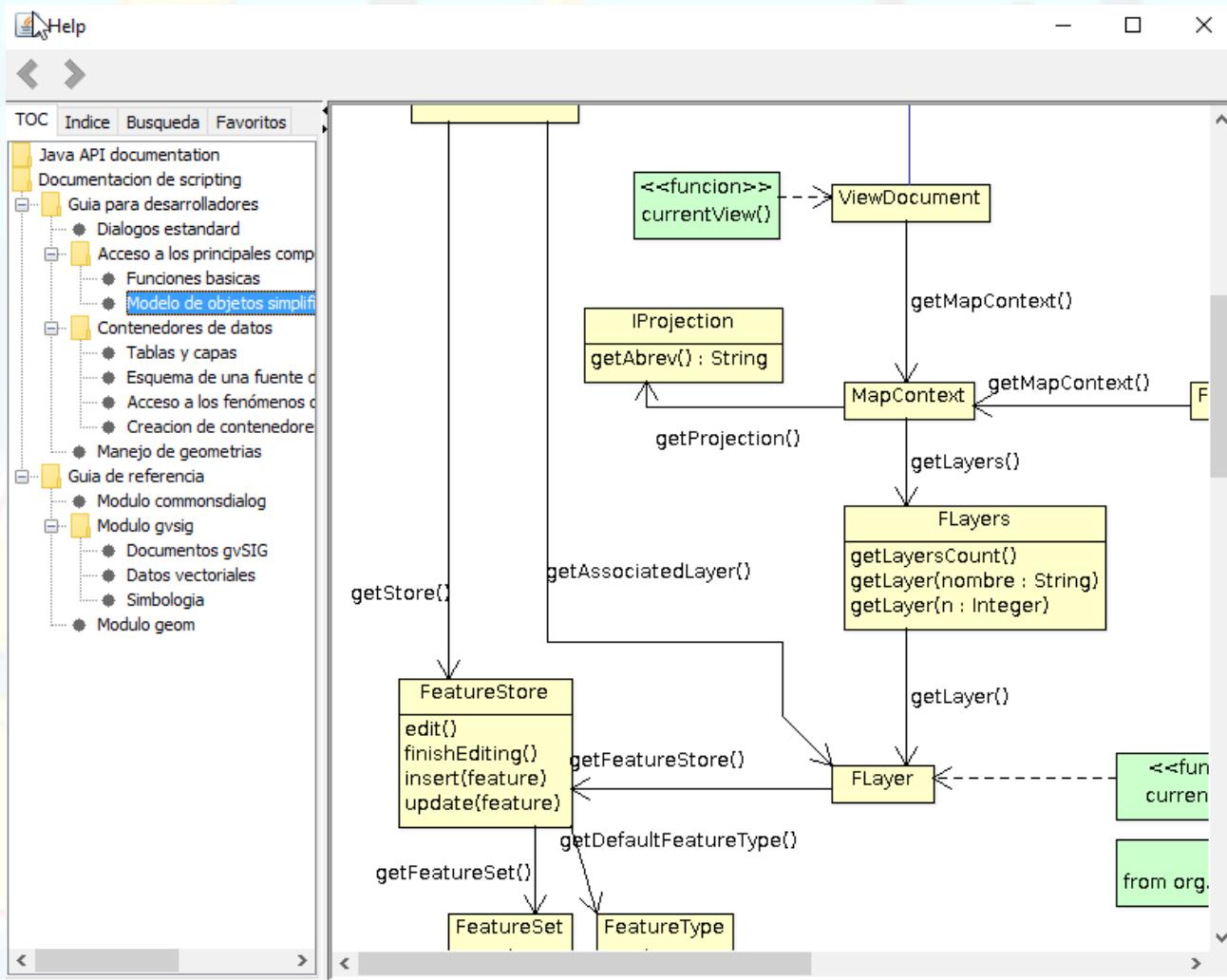
From gvsig import *
From commonsdialog import *
From es.unex.sexante.core import Sextante

class XYShift(ToolboxProcess):
    ...

    def defineCharacteristics(self):
        """
        En esta operacion debemos definir los parametros de entrada y salida que va a precisar nuestro proceso.
        """
        ...# Fijamos el nombre con el que se va a mostrar nuestro proceso
        ...self.setName("Prueba.desplazamiento.en.X.e.Y")
```



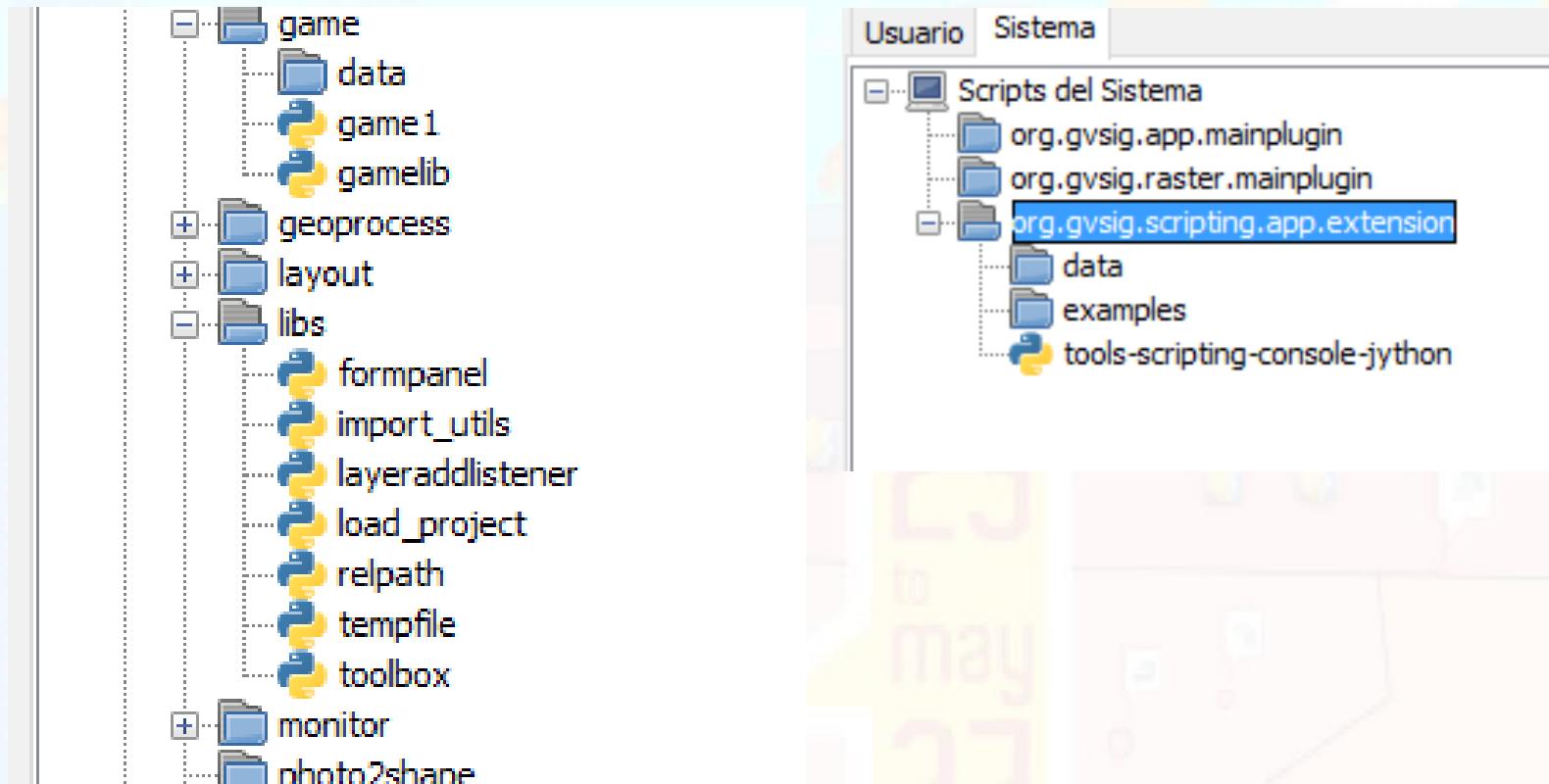
Help documentation



Scripting and
Javadocs
documentation

Imports

Import functions from others scripts or
libs directly with *import libname*





Use plugin & jar

Use plugins already installed inside
gvSIG
(use_plugin)

Use new jars from outside gvSIG
Java (jars)
(use_jar)



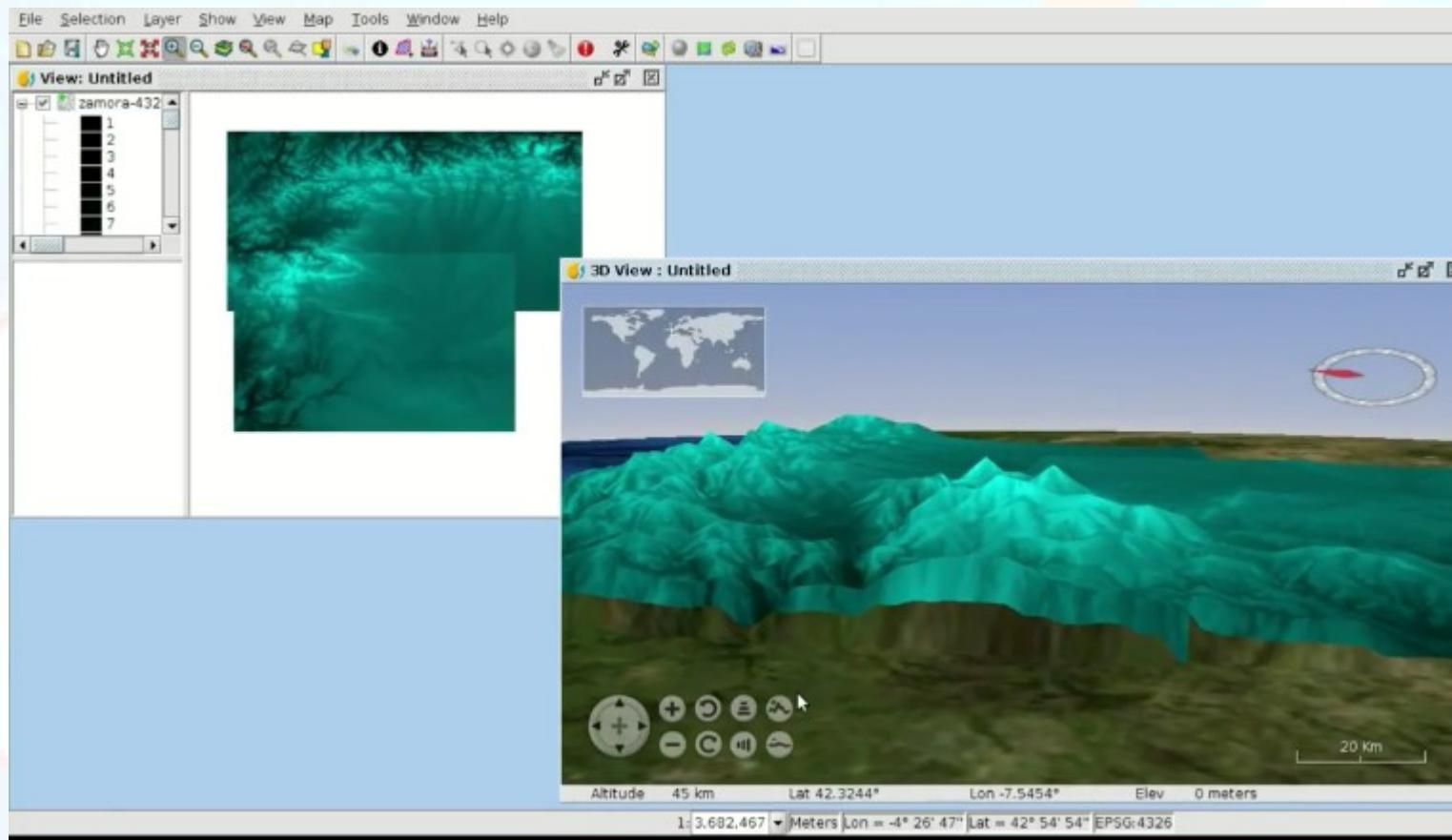
Libraries

Included some libraries:

- **geopy**: address geocodification
- **jOpenDocument**: reports
- **cartodb**: upload files and more

WorldWind View Plugin

Still not fully developed: Accessing to the 3D View
from scripts
It will be a big feature



F

E

S

T

I

V

gvSIG

AL

Big Improvements



Jython

- Updated libraries (gvsig and geom)
- All the module has been updated
- We tried don't lose compatibility (same syntax)
- **New approached to gvsig lib:**
Improve Java API with Python functions

Jython

Before (gvSIG 2.2)

- Less powerful
- Requires a lot of development
- Rewrite all classes (mix of Python and Java Classes)
- Hard if one method doesn't exist

```
class View(WrapperToJava):
    """
    Represents gvSIG view document
    (org.gvsig.app.project.documents.view.DefaultViewDocument).
    """

    def __init__(self, view):
        WrapperToJava.__init__(self, view)

    def getLayer(self, name=None):
        """
        Returns one view layer. If name is None returns active
        layer if the view has one, if name is not None returns layer
        else returns None.
        :param name: view name in Table Of Contents
        :type name: string
        :return: View
        :return: None
        """
        map = self.getMapContext();
        if name == None:
            activeLayers = map.getLayers().getActives()
            if len(activeLayers) != 1 :
```

After (gvSIG 2.3 and more)

- All of the gvSIG and Java power
- Doesn't require big developments
- All gvSIG Objects are Java
- Added Python methods to Java classes

```
def __getWindowOfView(self):
    application = ApplicationLocator.getManager()
    projectManager = application.getProjectManager()
    viewManager = projectManager.getDocumentManager(V
    return viewManager.getMainWindow(self,None)

def currentLayer():
    try:
        return currentView().getLayer()
    except:
        return None

#
# Inject new methods in the class JViewDocument
#
JViewDocument.centerView = __centerView
JViewDocument.showWindow = __showWindow
JViewDocument.getLayer = __getLayer
JViewDocument.getLayers = __getLayers
```

Jython

So.. you could use directly all Java classes and methods from the Javadocs

The screenshot shows a JavaDoc interface for the class **DefaultFLyrRaster**. On the left, there is a navigation tree with various Java packages and classes listed under **org.gvsig.raster.fmap.layers**. The main content area displays the following information:

- Class DefaultFLyrRaster**
- java.lang.Object**
 - org.gvsig.tools.dispose.impl.AbstractDisposable
 - org.gvsig.fmap.mapcontext.layers.FLyrDefault
 - org.gvsig.raster.fmap.layers.DefaultFLyrRaster
- All Implemented Interfaces:**
 - java.util.EventListener, Projected, VisualPropertyListener, ExtendedPropertiesSupport, FLayer, FLayerHidesArea, LayerListener, Classifiable, InfoByPoint, SingleLayer, Metadata, FLyrRaster, ILayerState, IRasterLayerActions, Multiresolution, Disposable, DynObject, Persistent
- public class DefaultFLyrRaster**
extends FLyrDefault
implements FLyrRaster, Multiresolution, InfoByPoint, Classifiable, IRasterLayerActions, IL...
- Raster layer**
- Nested Class Summary**
- Nested Classes**

Modifier and Type	Class and Description
class	DefaultFLyrRaster.RasterTaskStatus

- Nested classes/interfaces inherited from class org.gvsig.fmap.mapcontext.layers.FLyrDefault**
- FLyrDefault.RegisterMetadata, FLyrDefault.RegisterPersistence**

+ Pythons functions and methods that will make your scripts so easy to develop and maintain



Development examples

Intersection

“Es decir elijo la capa [...] pedir que se interseque con otras capas y que me indique cuáles de esos puntos se encuentran dentro de una cuenca hidrográfica, un área protegida, una provincia, entre otros

¿existe alguna forma de hacer esto automatizado?”

```

Point info: {u'campo2': u'Venecia', u'campo1': 101L}
- Layer: valencia_index
-- {u'fecha': u'1980-01-01', u'location': u'/cdrom/dat
-- {u'fecha': u'1992-01-01', u'location': u'/cdrom/dat
-- {u'fecha': u'2002-01-01', u'location': u'/cdrom/dat
- Layer: parcelas_Valencia
-- {u'PARCELA': u'03', u'AREA': 327L, u'FECHAALTA': 2002-01-01
- Layer: manzanas_valencia
-- {u'AREA': 1061L, u'FECHAALTA': 20011119, u'MUNICIPIO': u'Valencia', u'NUMERO': 101L}
```

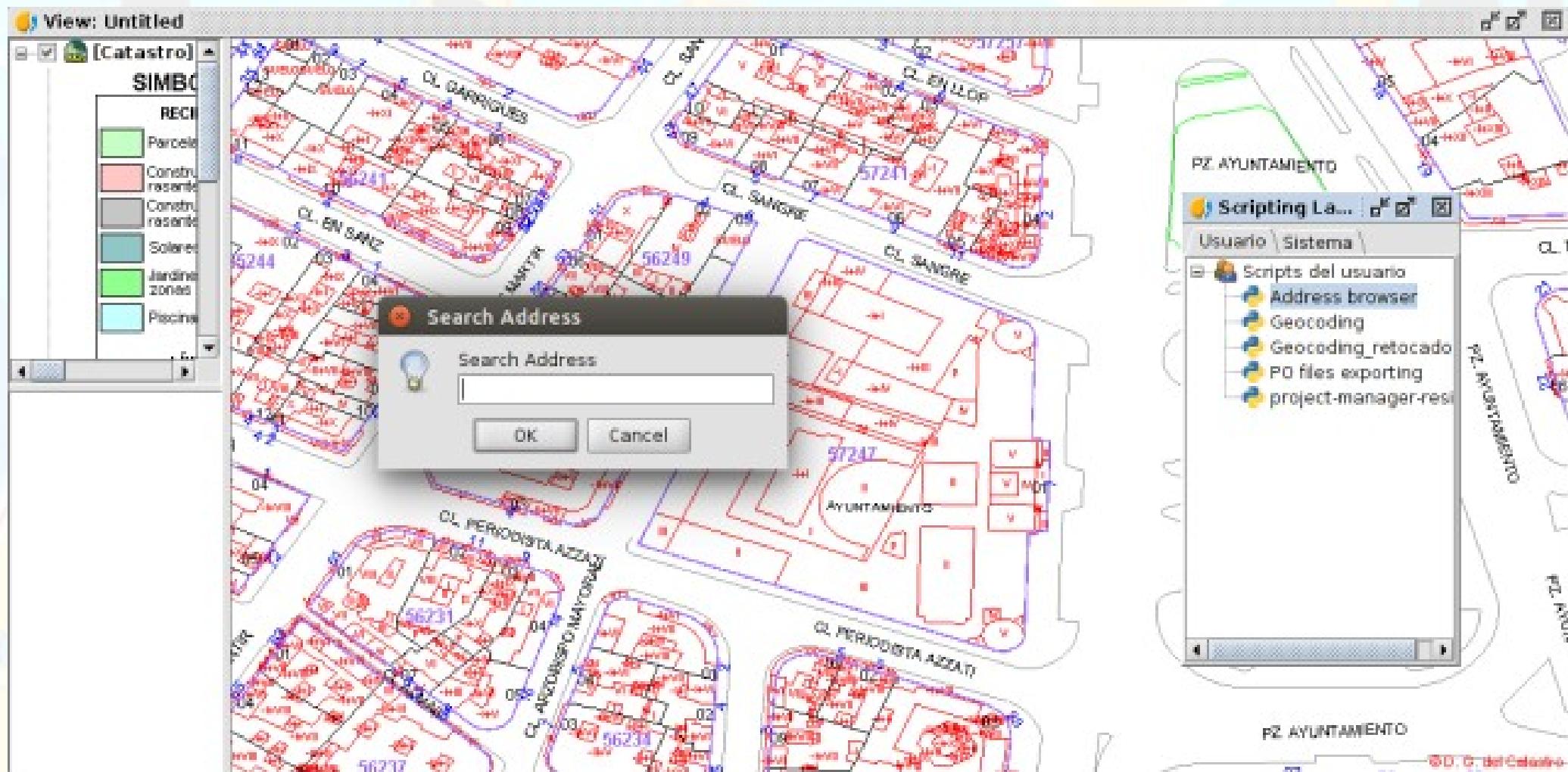
```

1
2 from gvsig import *
3
4 def infoInter(featurePoint, lstLayers):
5     ... pointValues = featurePoint.getValues()
6     ... pointValues.pop('GEOMETRY')
7     ... gfeat = featurePoint.geometry()
8     ... print "\nPoint.info: ", pointValues
9     ... for layer in lstLayers:
10        ...     print "--Layer: ", layer.name
11        ...     for pol in layer.features():
12            ...         polValues = pol.getValues()
13            ...         gpol = pol.geometry()
14            ...         polValues.pop('GEOMETRY')
15            ...         if gpol.intersects(gfeat):
16                ...             print "--", polValues
17
18 def main(*args):
19     #Intersection.info
20     lyrPoints = currentLayer()
21
22     #Capas diferentes a la de puntos
23     lstLayers = []
24     for capa in currentView().getLayers():
25         ... if not capa.name == lyrPoints.name:
26             ...     lstLayers.append(capa)
27
28     #intersecar puntos
29     for point in lyrPoints.features():
30         ...     infoInter(point, lstLayers)
31
```



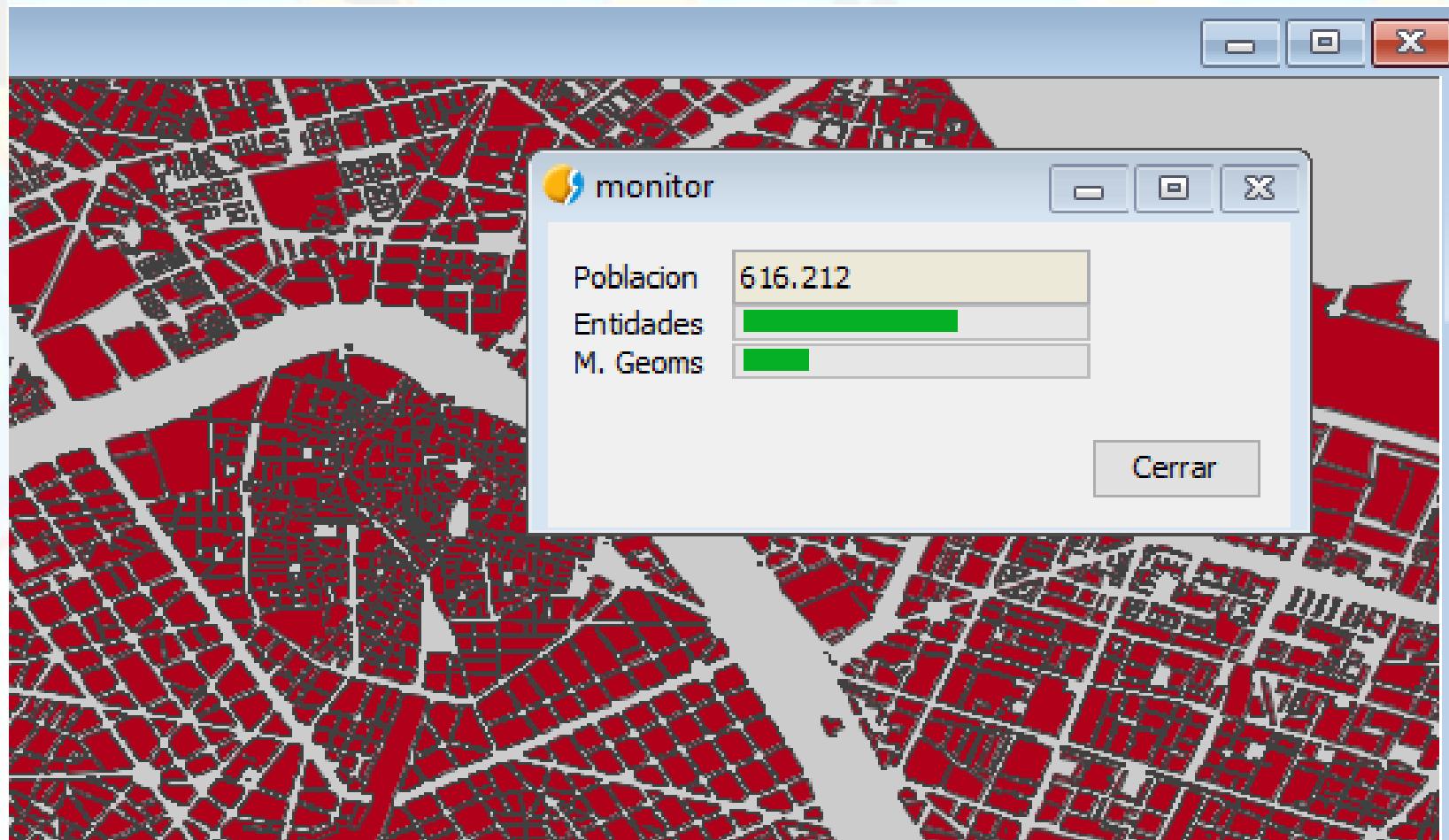
gvsig

geopy



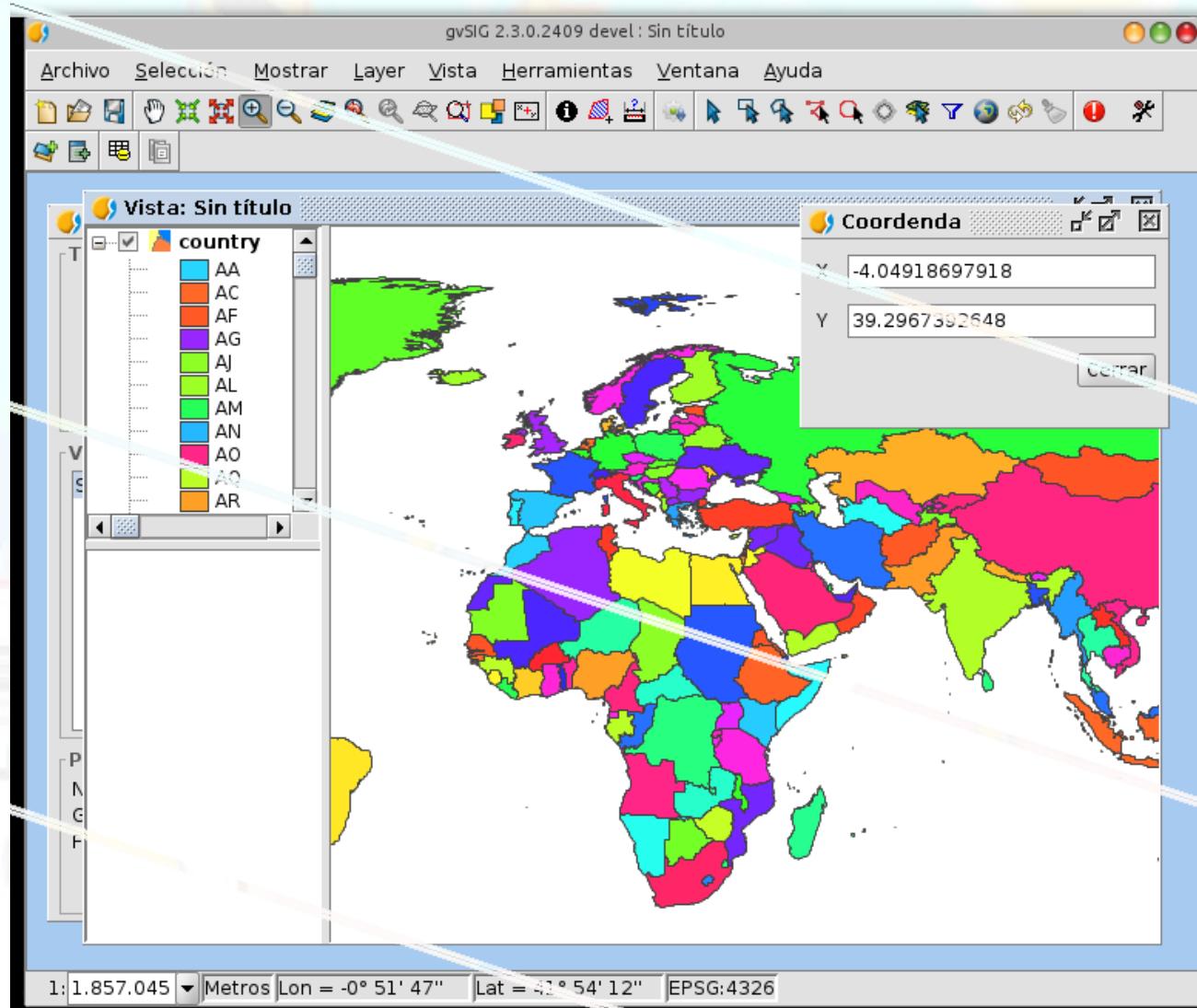


Monitor





Coordinates

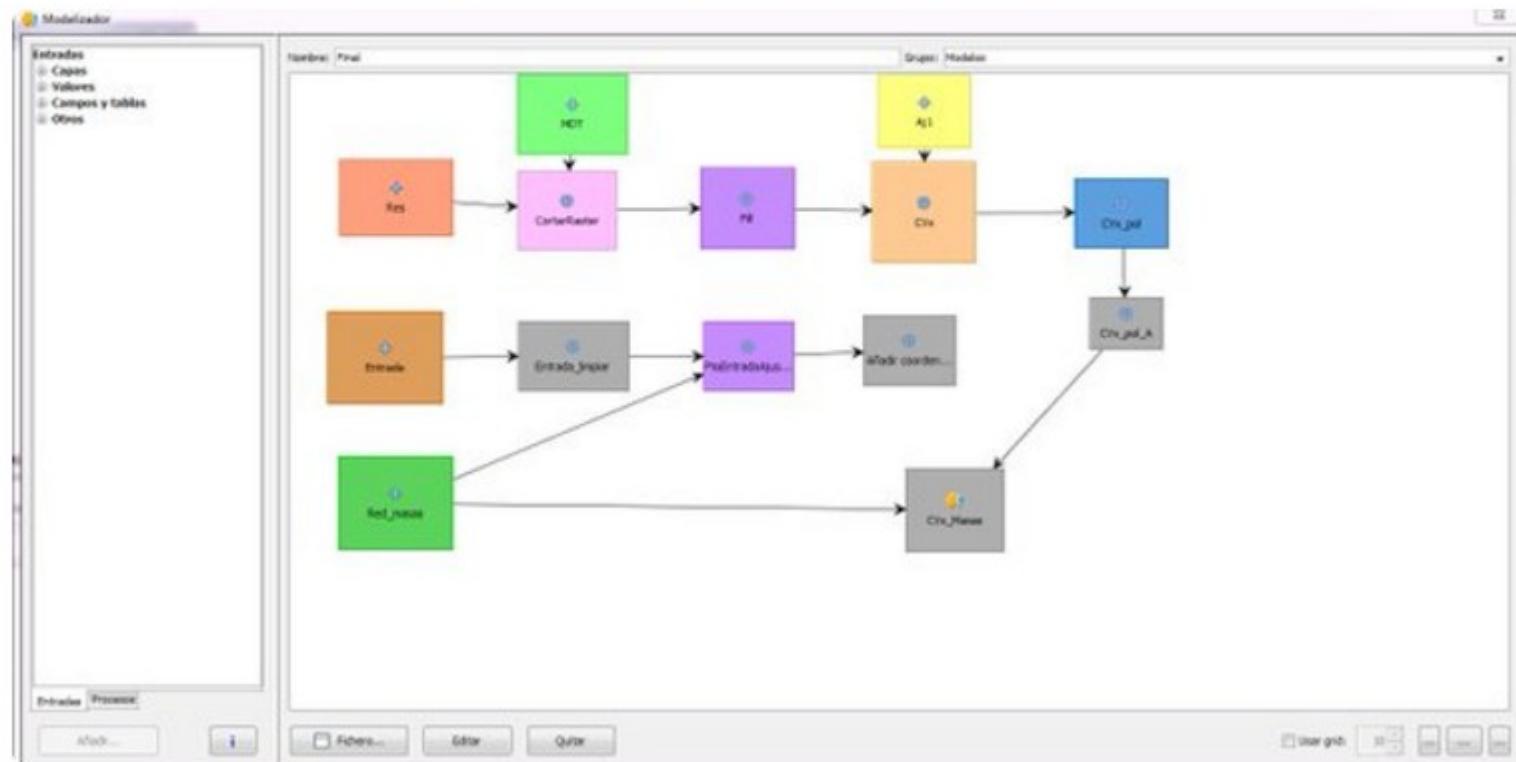


Ideya



ideyared @ideyared · 7 oct.

Probando la potencia del Model Builder de @gvSIG en la aplicación de proyectos hídricos. @AlvaroAnguix @masquesig



CartoDB

```
1  from gvsig import *
2  import cartodb1
3
4
5  def main(*args):
6      ... API_KEY = '*****'
7      ... cartodb_domain = 'user'
8      ... cl = CartoDBAPIKey(API_KEY, cartodb_domain)
9      ... print(cl.sql('select * from mytable'))
10
11
```



<https://github.com/CartoDB/cartodb-python>



jOpenDocument

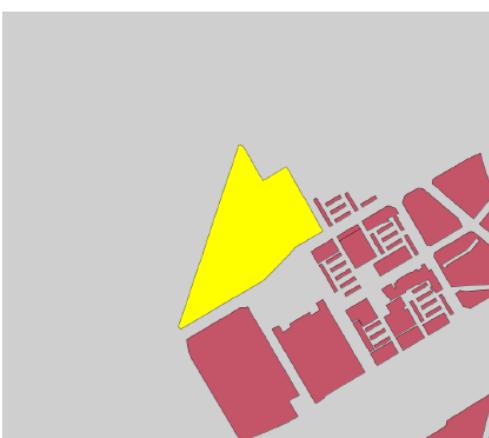
test_parcela00.fodt - LibreOffice Writer

Archivo Editar Ver Insertar Formato Tabla Herramientas jOpenDocument Ventana Ayuda

Encabezado 1 Arial 18,2

Información parcelaria

Hoja	YJ2743H
Área	763681
Coord. X	724568
Coord. Y	437396



Imagen



The screenshot shows a LibreOffice Writer document titled "test_parcela01.odt". The main content is a map of land parcels, with one specific parcel highlighted in yellow. The document includes a header with "Encabezado 1" and "Arial" font, and a toolbar with various icons. On the right side, there is a sidebar with "Recibo" and "Envio" sections, and a table with columns "DESCRIPTION" and "QUANTITY". The table contains two entries: "Área de parcela Superficie" with a quantity of 123, and "Personal" with a quantity of 10. The bottom left corner shows the word "Imagen".

DESCRIPTION	QUANTITY
Área de parcela Superficie	123
Personal	10

The screenshot shows a LibreOffice Calc spreadsheet with the following content:

Presupuesto parcela n. 1

Topografía. S.A
Av. Blasco Ibáñez
Valencia
España

\$3,095.00

Total due by October 5, 2015

Recibo
Tech. N.A
Av. Madrid
Barcelona, España

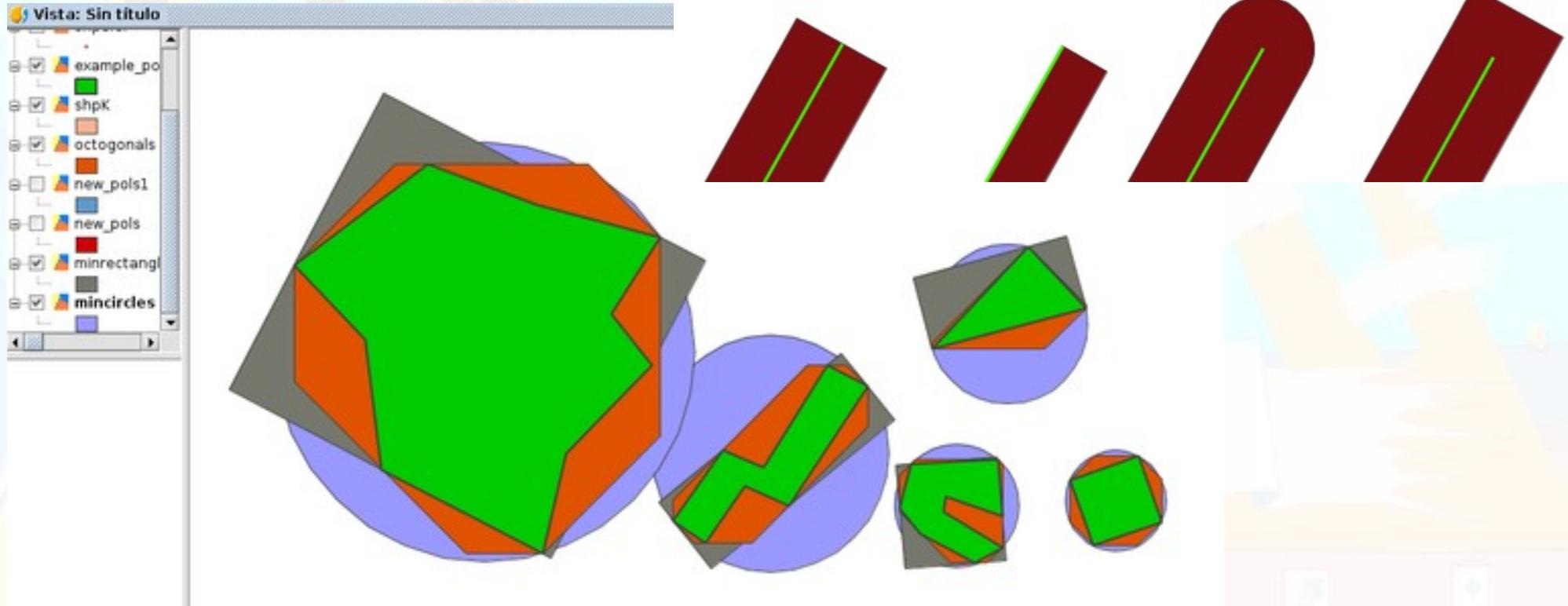
Envío
Tech. N.A
Av. Madrid
Barcelona, España

Pedido n. 01

DESCRIPTION	QUANTITY	PRICE	TOTAL
Área de parcela Superficie	123	\$25.00	\$3,075.00
Personal	10	\$2.00	\$20.00

Sub Total	\$3,095.00
Tax	\$0.00
Free	\$0.00
Discount	\$0.00
Total	\$3,095.00

Java Topology Suite (JTS)



```
... import com.vividsolutions.jts.operation.distance.DistanceOp as DistanceOp
... d = DistanceOp(l1, l2)
... nearestPoints = d.nearestPoints()
... print nearestPoints
... print "Punto de geom.1: ", nearestPoints[0]
... print "Punto de geom.2: ", nearestPoints[1]
... 
```

jgrasstools

```
1  from gvsig import *
2  from com.vividsolutions.jts.io import WKTReader
3
4  from java.util import List
5
6  from org.geotools.data.simple import SimpleFeatureCollection
7  from org.geotools.feature import DefaultFeatureCollection
8  from org.geotools.feature.simple import SimpleFeatureBuilder
9  from org.geotools.feature.simple import SimpleFeatureTypeBuilder
10 from org.geotools.referencing.crs import DefaultGeographicCRS
11 from org.jgrasstools.gears.modules.v.smoothing import OmsLineSmootheningJaitools
12 from org.jgrasstools.gears.modules.v.smoothing import OmsLineSmootheningMcMaster
13 from org.jgrasstools.gears.utils.features import FeatureUtilities
14 from org.opengis.feature.simple import SimpleFeature
15 from org.opengis.feature.simple import SimpleFeatureType
16
17
18 from com.vividsolutions.jts.geom import Geometry
19 from com.vividsolutions.jts.geom import LineString
20
21 from java.lang import Integer
22
23 def main(*args):
24     ... #https://code.google.com/p/jgrasstools/source/browse/jgrassgears/src/test/java/org
25
26     ... print "Test JGrassTools: TestLineSmootheningJaitools\n"
27     ... b = SimpleFeatureTypeBuilder()
28         ...
```

Code Properties

Problems Console

```
Running script test_jgrasstools_2.
Test JGrassTools: TestLineSmootheningJaitools

Collecting geometries...
Smoothing features...
100%...
Finished.
```

```
New geom: LINESTRING (0 0, 0.0342935528120713 0.0342935528120713, 0.12620027434
10
```



SQL Console

SQL Console

Conexion

```
1 select.* from municipios_navarra
```

Ejecutar

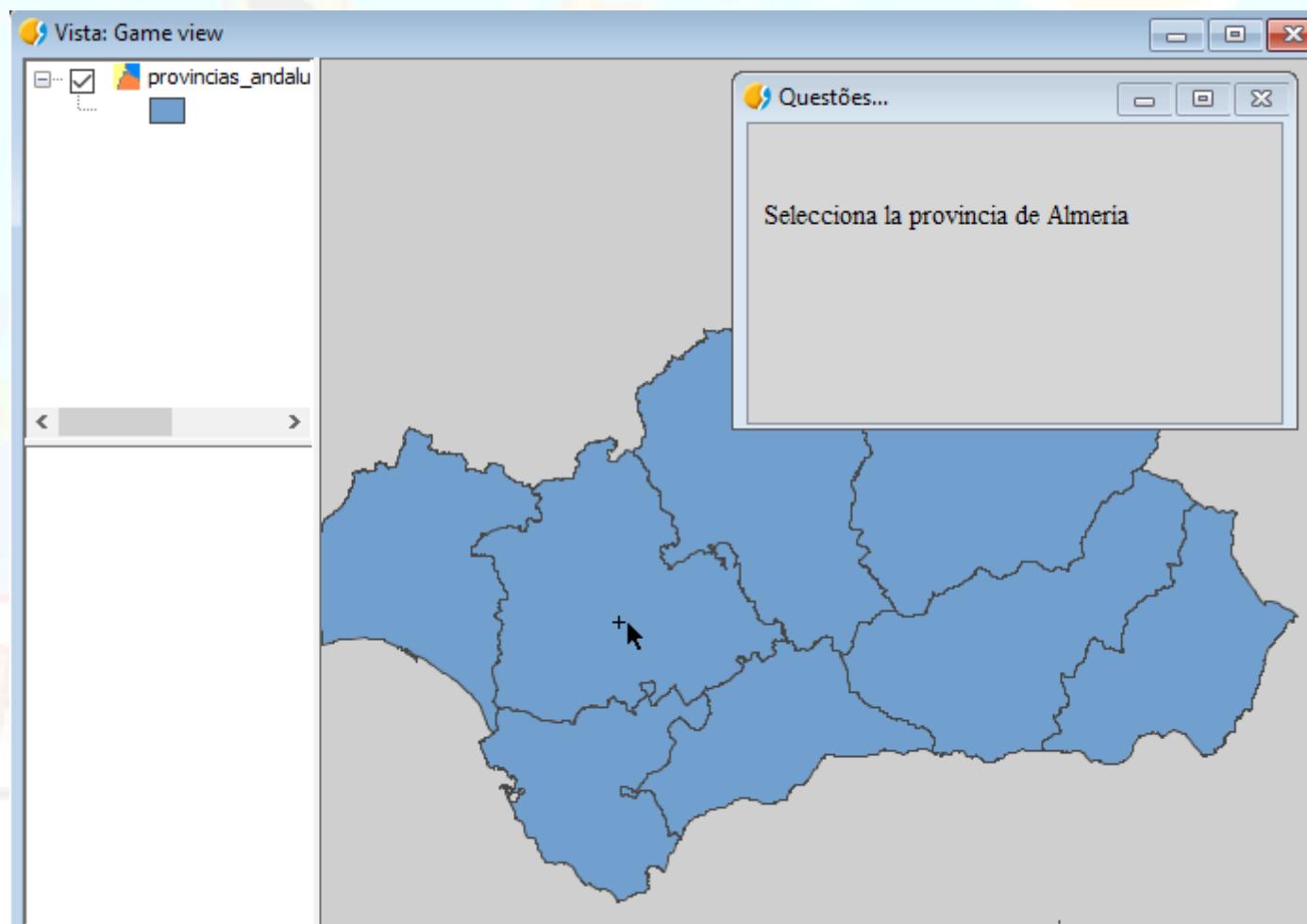
Tabla Mensajes

Clave primaria Campo geometria Cargar como Ta... Cargar

Tipo de geometria Subtipo

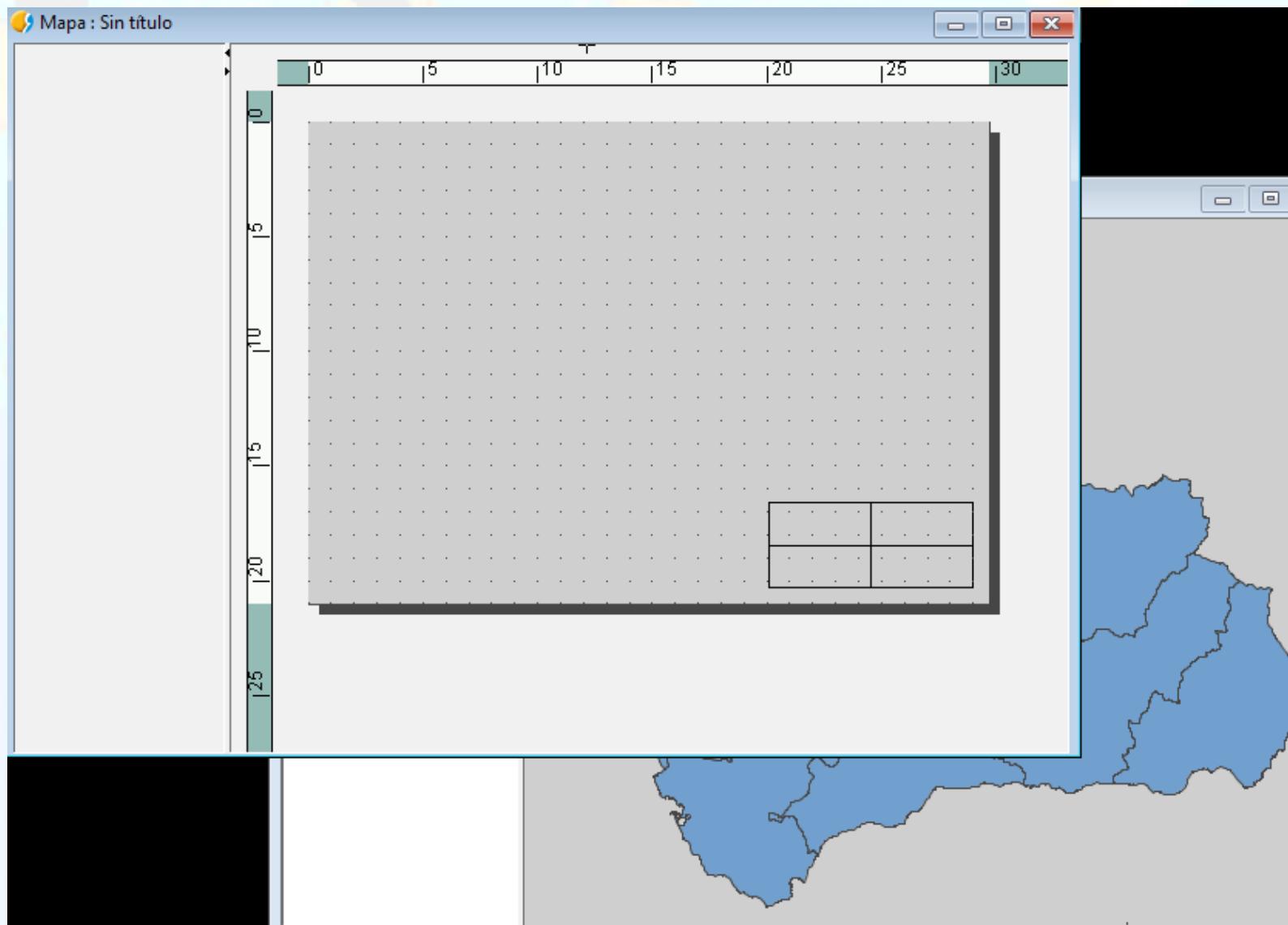


Game





Automate maps creation





Geotagging Digital Photos

```
def getLatitude(self, defaultValue=None):
    return self.__values.get("GPS.Latitude", defaultValue)

def getLatitudeRef(self, defaultValue=None):
    return self.__values.get("GPS.Latitude.Ref", defaultValue)

def getLongitude(self, defaultValue=None):
    return self.__values.get("GPS.Longitude", defaultValue)

def getLongitudeRef(self, defaultValue=None):
    return self.__values.get("GPS.Longitude.Ref", defaultValue)

def getAltitude(self, defaultValue=None):
    return self.__values.get("GPS.Altitude", defaultValue)

def getAltitudeRef(self, defaultValue=None):
    return self.__values.get("GPS.Altitude.Ref", defaultValue)

def getDatum(self, defaultValue=None):
    return self.__values.get("GPS.Map.Datum", defaultValue)

def getDate(self, defaultValue=None):
    return self.__values.get("GPS.Date.Stamp", defaultValue)

def getTime(self, defaultValue=None):
    return self.__values.get("GPS.Time.Stamp", defaultValue)
```



Scripts from MOOC students

Enseñanza de la geografía política mediante ejercicios y juegos
(Manuel Maria Alvarez)

Column Statistics
(Piotr Pachól)

Importador / Exportador de geojson a shapes
(Jose Alberto Gonzalez y Francisco Puga)

Crear copia de capa con los campos deseados y en orden
(Jose Luis Lopez Janas)

**Reporte de viaje del servicio Sub-urbano Asunción-Ypacarai
del FCPPCAL y ubicación en el tiempo**
(Sergio Spiridonoff Reyes)

Sinuosity Index Schumm classification
(Carles Clua Millan)



More info:

- Mooc: Scripting in gvSIG
- Blog gvSIG
<http://blog.gvsig.org/>
- Outreach: Scripts examples
<http://outreach.gvsig.org/scripts/>
- Emails lists: Users and developers
<http://www.gvsig.com/es/comunidad/listas-de-correo>



Thanks!

Óscar Martínez
@masquesig
omartinez@gvsig.com

info@gvsig.com
www.gvsig.com