GENERALITAT VALENCIANA
CONSELLERIA D'INFRAESTRUCTURES I TRANSPORT

Scripting guide Version 2
gvSIG 1.0

IVER - OFICINAS CENTRALES EN VALENCIA
C/ Salamanca nº 50-52 , 46005-VALENCIA
Telf: 902 25 25 40 - Fax: 96 316 27 16
E-Mail dac@iver.es www.iver.es

Conselleria de Infraestructuras y Transporte
C/ Blasco Ibáñez Nº 50 , 46010
VALENCIA

E-Mail gvsig@gva.es

Project web: http://www.gvsig.gva.es

Índice de contenido

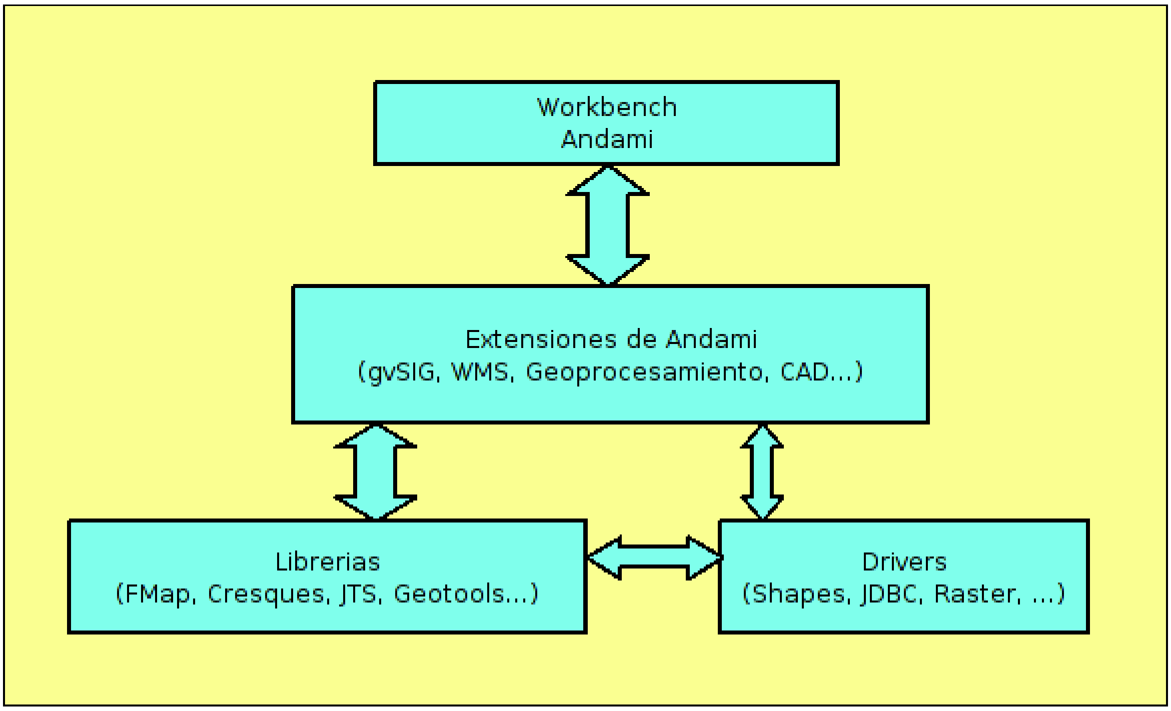
1 Introduction

The gvSIG project is a framework which, together with the scripting engine, makes it more functional without advanced knowledge of the application core.

The gvSIG application is built on a system of layers which are integrated using pre-defined mechanisms called **extensions**. In turn, each extension defines its own extension points. This model allows developers to add a wide range of variables to the basic gvSIG framework so that each tool's artefacts, such as the different types of layers or buttons, are presented to the user from a common platform.

The extension developers also benefit from this architecture. gvSIG's basic framework provides them with a series of services they do not have to worry about and this means they can concentrate on the specific tasks of their extension.

The following graph shows a simplified version of this system.


Workbench
Andami
Extensiones de Andami
(gvSIG, WMS, Geoprocesamiento, CAD...)
Librerias
(FMap, Cresques, JTS, Geotools...)
Drivers
(Shapes, JDBC, Raster, ...)

The gvSIG platform core is made up of three subsystems:

- **Andami**. This represents the framework gvSIG is based on. It is like a frame the different extensions that make up the application fit into. In addition, it shows a MDI user environment and defines the appearance of the application windows.

- **Fmap**. This is the GIS heart of the platform. It includes all the necessary classes to handle GIS objects, such as drivers and adapters to be able to use the most common formats used to store cartographic data. This library includes classes to read and write the supported formats, draw the maps using the best scales, assign legends, define symbols, carry out searches, requests, analyses, etc.

- **gvSIG extension**. This extension contains the part of the user interface that shows the geographic data handled by **Fmap**. This subsystem contains the classes that implement the majority of the dialogue boxes used by the final application and the classes which support these dialogue boxes. For example, the forms for assigning legends, creating maps and views, defining scales, etc. are found in this subsystem.

1.1 Andami extensions

Andami extensions are defined using an .xml file which must be located in a subfolder of the gvSIG bin/gvSIG/extensiones directory. This configuration file indicates the classes Andami needs to load when starting the application, specifies the menu options that need to appear and the buttons that need to appear in the tool bar.

1.2 The GUI library for scripting

Thinlet is a GUI toolkit graphic library. It separates the graphic presentation and application methods. An XML file with XUL format is used to define the graphic interface.

A tool called ThinG allows the graphic interface to be designed.

For further information about Thinlet:

http://thinlet.sourceforge.net/

http://sjobic.club.fr/thinlet/scriptablethinlet/index.html

http://thing.sourceforge.net/

2 Examples

gvSIG supports several programming languages for scripting.

Examples are put together using the Python version 2.1. programming language in its implementation for the Java virtual machine (Jython).

For information about Jython, see www.jython.org

For information about Python, see www.python.org

2.1 Centring a view on a point

An extension is created which will allow us to specify coordinates so that the view can be centred on the input coordinates and a point can be drawn on this spot.

To do so:

- We will create a “bin/gvSIG/extensiones/centrarVistaSobreUnPunto” folder in the gvSIG directory.
- We will create a file in the folder called config.xml with the contents:

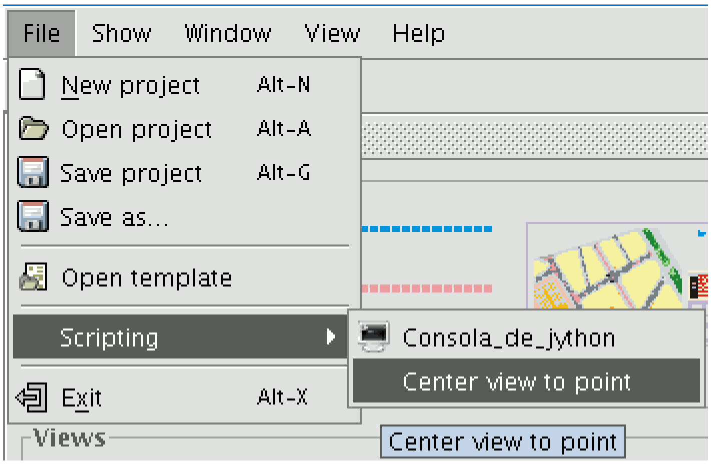
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<plugin-config>
        <libraries library-dir="../org.gvsig.scripting"/>
        <depends plugin-name="org.gvsig.scripting"/>
        <resourceBundle name="text"/>
        <extensions>
                <extension class-name="org.gvsig.scripting.ScriptingExtension"
                        description="Extension de soporte para Scripts de usuario."
                        active="true">

                        <menu text="Archivo/Scripting/Centrar vista en un punto"
                          tooltip="Centrar la vista en un punto"
                                action-command=""
                                position="55"
                                />
                </extension>
        </extensions>
        </plugin-config>
```

The following tags are located in the XML:

  - *libraries library-dir*: This indicates the directory the scripting libraries are located in.

- *depends plugin-name*: This indicates that the extension needs another plug-in to function.
- *resourceBundle name*: This indicates where the translation file is located.
- *extension class-name:* This class implements the scripting in gvSIG. The value for this tag should be ”org.gvsig.scripting.ScriptingExtension”.
- *description*: Description of the extension.
- *active*: This indicates whether the extension is active or not, the value must always be “true”.
- *menu text:* This tag defines where and how the entry is added to the menu bar. In this case, a new entry will be created in the “File” menu then “Scripting” with the name “Centre view on a point”. The properties which must be defined in this tag are:
  - *tooltip:* This mark defines a text which will be shown when the mouse pointer is positioned in the menu entry.
  - *action-command*: This admits two types of actions; **show** displays a window, and **run** executes a script. Both admit several parameters which are explained later.
  - *icon*: This establishes the path of the icon associated with the menu entry we are creating. The path is relative to the extension directory. If the path is not correct, the gvSIG application cannot be started.
  - *position*: This establishes the position in which the menu entry must be loaded in the menu bar. Positions between 1 and 99 can be defined, 1 is the first in the menu and 99 is the last. If two entries in the menu are in the same position only one of them will be loaded.

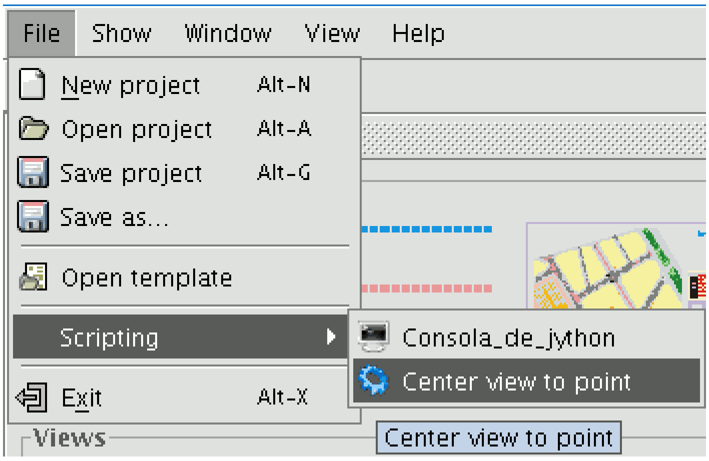
When the file has been created, we can start gvSIG and check that the entry has been added to the menu.



- An "images" directory will be created in the folder we created previously. A file will be copied to it and this will be used as an icon. This file is located in "bin/gvSIG/extensiones/org.gvsig.scripting/images/default.png".
- We must add a new tag to the menu entry in the config.xml.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<plugin-config>
        <libraries library-dir="../org.gvsig.scripting"/>
        <depends plugin-name="org.gvsig.scripting"/>
        <resourceBundle name="text"/>
        <extensions>
                <extension class-name="org.gvsig.scripting.ScriptingExtension"
                        description="Support extension for user scripts."
                        active="true">
                        <menu text="File/Scripting/Center view to point"
                          tooltip="Center view to point"
                              action-command=""
                              icon="images/default.png"
                              position="55"
                              />
                </extension>
        </extensions>
</plugin-config>
```

- Start the application. We will see that the option and the icon appear in the menu.



- Create the centrarVistaSobreUnPunto.xml file together with the recently created config.xml file, with the following content:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<panel columns="3" gap="3">
    <label colspan="3" text="Coordinates to center view"/>
    <label colspan="2" halign="right" text="x:"/>
    <textfield name="txtX"/>
    <label colspan="2" halign="right" text="y:"/>
    <textfield name="txtY"/>
    <panel colspan="3" gap="2" halign="right">
        <button halign="right" name="botAplicar" text="Apply"/>
        <button halign="right" name="botCerrar" text="Close"/>
    </panel>
</panel>
```

  The centrarVistaSobreUnPunto.xml file defines a window which will be shown when you click on the menu entry you have added.

- Modify the config.xml file and add the “**show**” instruction to the tag action-command.

  The **show** instruction receives the parameters:

  filename: This indicates where the window to be shown is defined.

  language: This indicates the language used in the script. gvSIG currently supports several script languages although only jython has been tested.

  title: This defines the title with which the window will be shown.

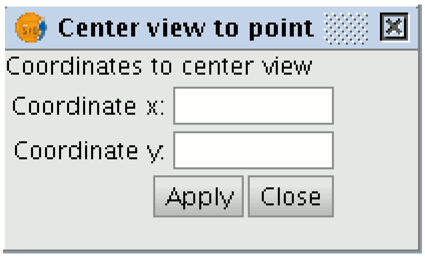
  width: The initial width of the window.

  height: The initial height of the window.

The config.xml will have the following contents when the tag has been added.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<plugin-config>
        <libraries library-dir="../org.gvsig.scripting"/>
        <depends plugin-name="org.gvsig.scripting"/>
                <resourceBundle name="text"/>
        <extensions>
                <extension class-name="org.gvsig.scripting.ScriptingExtension"
                        description="Support extension for user scripts."
                        active="true">
                        <menu text="Archivo/Scripting/Center view to point"
                          tooltip="Center view to point"
                              action-command="show(
fileName='gvSIG/extensiones/centrarVistaSobreUnPunto/centrarVistaSobreUnPunto.xml',
language='jython', title='Center view to point', width=210, height=86)"
                                icon="images/default.png"
                                position="55"
                                />
                </extension>
        </extensions>
</plugin-config>
```

- Start the application and you will see that when you click on the added menu option the window appears.



- We have to add an action on the “Close” button. To do so, we must include the “**thinlet.closeWindow()**” action on the tag corresponding to the button in the file which defines the window (in our case “centrarVistaSobreUnPunto.xml”).

  The **thinlet** object is the extension window and allows access to the controls.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<panel columns="3" gap="3">
    <label colspan="3" text="Coordinates to center view"/>
    <label colspan="2" halign="right" text="x:"/>
    <textfield name="txtX"/>
    <label colspan="2" halign="right" text="y:"/>
    <textfield name="txtY"/>
    <panel colspan="3" gap="2" halign="right">
       <button halign="right" name="botAplicar" text="Apply"/>
       <button halign="right" name="botCerrar" text="Close" action="thinlet.closeWindow()"/>
    </panel>
</panel>
```

Even though modifications are made in the xml file, the application does not have to be restarted. Run the menu option again to check that the "Close" button closes the window.

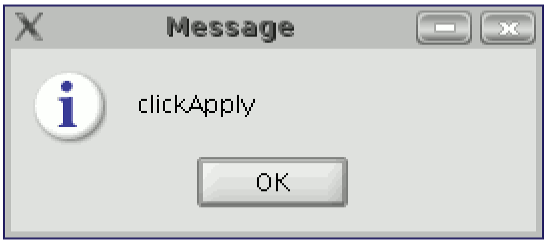
- We will now add an action to the “Apply” button. To do so, we have to create a file called “centrarVistaSobreUnPunto.py” together with the file “centrarVistaSobreUnPunto.xml” with the following contents:

```
from gvsiglib import *

def clickAplicar(thinlet):
    showMessageDialog("clickAplicar")
    return
```

The “centrarVistaSobreUnPunto.xml” file will be modified as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<panel columns="3" gap="3">
    <script language="jython" method="init" src="centrarVistaSobreUnPunto.py"/>
    <label colspan="3" text="Coordinates to center view"/>
    <label colspan="2" halign="right" text="x:"/>
    <textfield name="txtX"/>
    <label colspan="2" halign="right" text="y:"/>
    <textfield name="txtY"/>
    <panel colspan="3" gap="2" halign="right">
       <button halign="right" name="botAplicar" text="Apply"
action="clickAplicar(thinlet)"/>
       <button halign="right" name="botCerrar" text="Close"
action="thinlet.closeWindow()"/>
    </panel>
</panel>
```

- If we run the menu option again and click on the “Apply” button, a message appears.



clickApply

- We will modify the code of the “centrarVistaSobreUnPunto.py” file to define the window values and centre the view. The code should be as follows:

```
import java.awt.geom.Point2D as Point2D
import java.awt.geom.Rectangle2D as Rectangle2D

from gvsiglib import *

mapContext = None

def getMapContext():
    “””
    Devuelve el objeto mapContext asociado a la vista de gvSIG
    que tiene el foco en este momento.
    Si la ventana activa no es una vista retorna None
    “””
    vista = gvSIG.getActiveDocument()
    if vista == None:
      print "No se puede acceder al documento activo."
      return None
    try:
      mapContext = vista.getModel().getMapContext()
    except Exception, e:
      print "El documento activo no parece ser una vista. Error %s %s" % (
str(e.__class__),
str(e)
)
      return None

    return mapContext

# Obtenemos el mapContext antes de que se muestre nuestra ventana
mapContext = getMapContext()

def clickAplicar(thinlet):
    global mapContext

    if mapContext == None:
      print "No se puede acceder al documento activo."
      return

    if mapContext.getLayers().getLayersCount() < 1:
      print "El documento activo no tiene capas disponibles."
      return

    # Accedemos a los controles de la ventana que
    # hemos definido para recoger las coordenadas x e y
    # a través del objeto thinlet.
    x = float(thinlet.getString(txtX, "text"))
    y = float(thinlet.getString(txtY, "text"))
    return centrarEnLasCoordenadas(mapContext, x,y)

def centrarEnLasCoordenadas(mapContext, x,y):
  try:
    oldExtent = mapContext.getViewPort().getAdjustedExtent()
    oldCenterX = oldExtent.getCenterX()
    oldCenterY = oldExtent.getCenterY()
    center=Point2D.Double(x,y)
    movX = x - oldCenterX
    movY = y - oldCenterY
    upperLeftCornerX = oldExtent.getMinX()+movX
    upperLeftCornerY = oldExtent.getMinY()+movY
    width = oldExtent.getWidth()
    height = oldExtent.getHeight()
    extent = Rectangle2D.Double(upperLeftCornerX, upperLeftCornerY, width, height)
    mapContext.getViewPort().setExtent(extent)
    return center
  except ValueError, e:
    print "Se ha producido un error realizando zoom a las coordenadas (%s,%s). Error  %s, 
%s" % (
      repr(x),
      repr(y),
      str(e.__class__),
      str(e)
    )
    return None
```

- To test it, we will need a layer view, to which we could add control so that if the active document is not a layer view the “Apply” button will not be enabled. To do so, we need to include the following code at the end of the file:

```
def elDocumentoActivoEsUnaVistaValida():
    global mapContext

    if mapContext == None:
      print “El documento activo no parece ser una vista”
      return False

    if mapContext.getLayers().getLayersCount() < 1:
      print "El documento activo no tiene capas disponibles."
      return False
    return True

if elDocumentoActivoEsUnaVistaValida():
  thinlet.setBoolean(botAplicar,"enabled",True)
else:
  thinlet.setBoolean(botAplicar,"enabled",False)
```

- The next thing to be done is to draw a point on the input coordinates. This is achieved with the following function:

```
def drawPoint(mapContext, center, color=None):
    """
    Esta función pintará un punto sobre la capa de gráficos
    asociada al mapContext.
    Todo mapContext además de las capas que tenga cargadas dispone
    una capa graphics sobre la que dibujar elementos gráficos.
    """

    if color == None:
        import java.awt.Color as Color
        color = Color.blue

    layer=mapContext.getGraphicsLayer()
    layer.clearAllGraphics()
    theSymbol = FSymbol(FConstant.SYMBOL_TYPE_POINT,color)
    idSymbol = layer.addSymbol(theSymbol)
    geom = ShapeFactory.createPoint2D(center.getX(),center.getY())
    theGraphic = FGraphic(geom, idSymbol)
    layer.addGraphic(theGraphic)
```

And we could modify the clickAplicar function as follows:

```
def clickAplicar(thinlet):
    vista = gvSIG.getActiveDocument()
    if vista == None:
      print "No se puede acceder al documento activo."
      return
    try:
      mapContext = vista.getModel().getMapContext()
    except:
      print "El documento activo no parece ser una vista."
      return

    if mapContext.getLayers().getLayersCount() < 1:
      print "El documento activo no tiene capas disponibles."
      return
    x = float(thinlet.getString(txtX, "text"))
    y = float(thinlet.getString(txtY, "text"))
    centro = centrarEnLasCoordenadas(mapContext, x,y)
    drawPoint(mapContext,centro)
```

- When the point has been drawn… when is the point deleted from the view? Let us add a menu option which cleans the graphics layer. This time we will use the **run** command.
    - The parameters for the **run** command are:

      fileName: The path related to the gvSIG bin directory in which the file with the code to be run is located.

language: The language the code to be run is written in.

We can add another menu entry to the config.xml file.

```
<menu text="File/Scripting/Delete points"
      tooltip="Delete points"
      action-command=
"run(fileName='gvSIG/extensiones/centrarVistaSobreUnPunto/limpiarElGraphics.py',language='jy
thon')"
      icon="images/default.png"
      position="56"
        />
```

We then create a file called limpiarElGraphics.py with the following contents:

```
from gvsiglib import *

def main():
    vista = gvSIG.getActiveDocument()
    if vista == None:
      print "No se puede acceder al documento activo."
      return None
    try:
      mapContext = vista.getModel().getMapContext()
    except Exception, e:
      print "El documento activo no parece ser una vista."
      print "Error %s %s" % (str(e.__class__),str(e))
      return None
    if mapContext == None:
      return
    layer=mapContext.getGraphicsLayer()
    layer.clearAllGraphics()
    mapContext.invalidate()

main()
```

2.2 Requesting information about a point

In this example, we are going to create tools which will allow us to obtain information associated with a point in a specific layer loaded in a view. To do so, we will create a layer from a csv file (and we will use it as our example).

We will create two new tools, one to add the layer we are going to work with and another one to implement the layer information request operation.

For the first tool, we will create a csv file which will be used to create a gvSIG data source. From this, we can generate an event layer which can be loaded into the view. A property will be added to this layer so we can identify it and present the customised information window created for this layer. The tool must also ensure that the working layer we have created is not added to the view more than once if the tool is used repeatedly.

We will have to register a listener on the view's mapControl for the customised information tool. This listener receives the click events when the tool is activated. The information received from a point in a layer is an xml structure, which therefore requires a SAX parser to analyse it. This means that we need to create a SaxContentHandler to process the xml.

In addition to carrying out these options, we will create a user interface which is suitable for the data in our work layer.

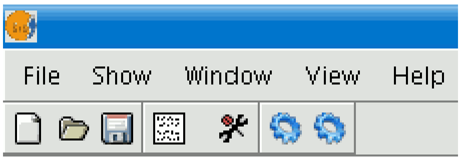
To create this tool we need to:

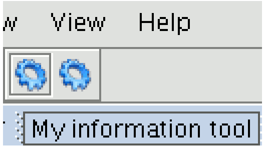
- Create a folder "bin/gvSIG/extensiones/miHerramientadeInformacion" in the gvSIG directory
- When "miHerramientadeInformacion" has been created, another folder has to be created inside it which can be called "images". Then we have to copy the "bin/gvSIG/extensiones/org.gvsig.scripting/images/default.png" file to this folder.
- The config.xml file is then created in the "miHerramientadeInformacion" folder. This file is responsible for defining the elements that need to be included in the menu bar and in the tool bar. The file contains the following:

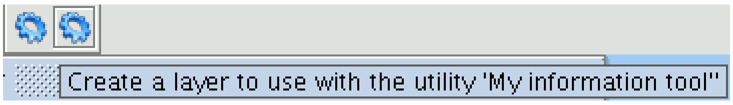
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<plugin-config>
        <libraries library-dir="../org.gvsig.scripting"/>
        <depends plugin-name="org.gvsig.scripting"/>
                <resourceBundle name="text"/>
        <extensions>
                <extension class-name="org.gvsig.scripting.ScriptingExtension"
                        description="Extension de soporte para Scripts de usuario."
                        active="true">
```

```
                      <menu text="Archivo/Scripting/Mi herramienta de información"
                            tooltip="Mi herramienta de información"
                            action-
command="run(fileName='gvSIG/extensiones/miHerramientaDeInformacion/miHerramientaDeInformaci
on.py',language='jython')"
            icon="images/default.png"
                              position="55"
                              />
      <menu text="Archivo/Scripting/Crear capa de trabajo"
            tooltip="Crea una capa para utilizarla con la utilidad 'Mi herramienta de
informacion'"
            action-
command="run(fileName='gvSIG/extensiones/miHerramientaDeInformacion/anyadirMiCapaDeTrabajo.p
y',language='jython')"
            icon="images/default.png"
            position="55"
        />
      <tool-bar name="Scripting">
        <action-tool icon="images/default.png"
                     action-
command="run(fileName='gvSIG/extensiones/miHerramientaDeInformacion/anyadirMiCapaDeTrabajo.p
y',language='jython')" tooltip="Crea una capa para utilizarla con la utilidad 'Mi
herramienta de informacion'"
                     />
        <action-tool icon="images/default.png"
                     action-
command="run(fileName='gvSIG/extensiones/miHerramientaDeInformacion/miHerramientaDeInformaci
on.py',language='jython')" tooltip="Mi herramienta de información"
                     />
      </tool-bar>
               </extension>
       </extensions>
</plugin-config>
```

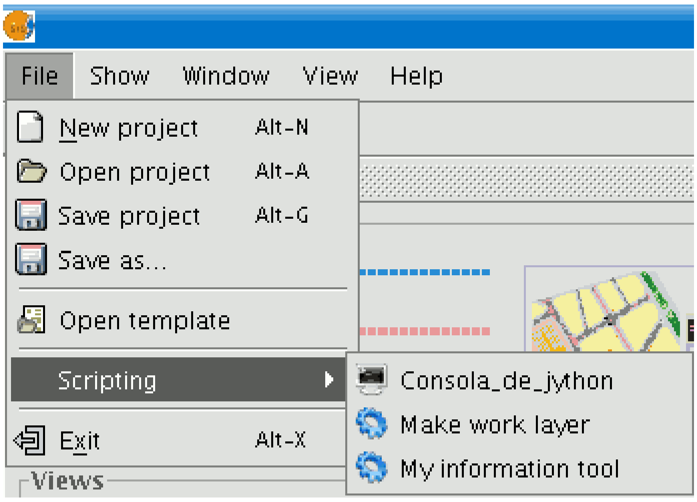
- At this point, the application can be started and we will see that the buttons have been added in the tool bar and the entries appear in the menu bars.







Create a layer to use with the utility 'My information tool"



- A “.csv” file is generated in the “miHerramientadeInformacion” folder file with the data that will be presented with the information tool. The file will be called “municipiosAdemuz.csv” and will contain the following:

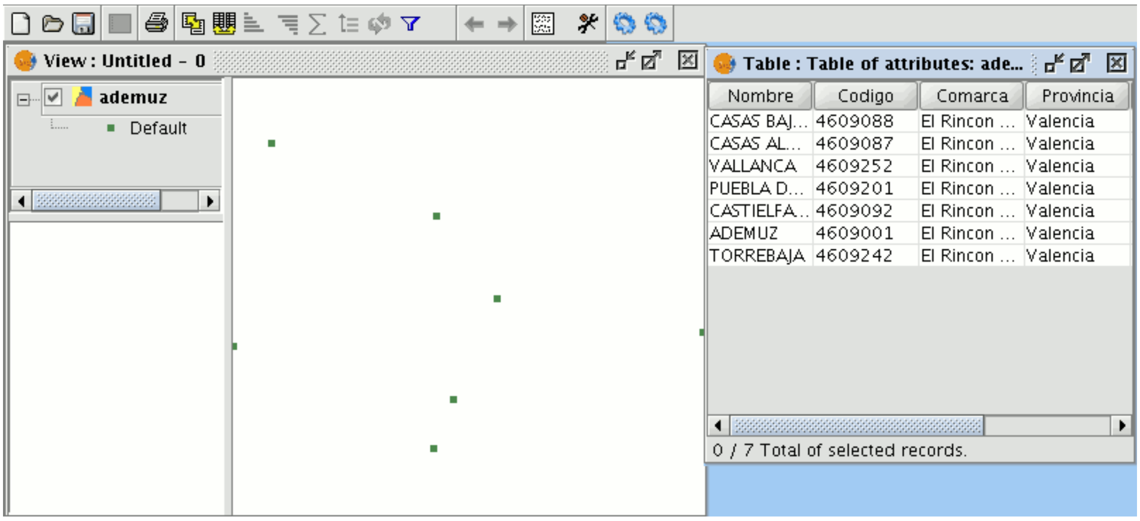
```
Nombre;Codigo;Comarca;Provincia;X;Y
CASAS BAJAS;4609088;El Rincon de Ademuz;Valencia;648522.72;4431068.44
CASAS ALTAS;4609087;El Rincon de Ademuz;Valencia;649319.84;4433082.21
VALLANCA;4609252;El Rincon de Ademuz;Valencia;640425.70;4435263.80
PUEBLA DE SAN MIGUEL;4609201;El Rincon de Ademuz;Valencia;659430.60;4435809.18
CASTIELFABIB;4609092;El Rincon de Ademuz;Valencia;641977.97;4443528.63
ADEMUZ;4609001;El Rincon de Ademuz;Valencia;651081.88;4437193.65
TORREBAJA;4609242;El Rincon de Ademuz;Valencia;648648.53;4440549.03
```

- The script responsible for creating the working layer to be used in our example can be created at this point. The file that contains this script will be called “anyadirmiCapadeTrabajo.py”. The content of this script will be as follows:

```
"""
Script que genera una capa de puntos a partir de un fichero csv
(municipiosAdemuz.csv) que se utiliza para trabajar con la herramienta
miHerramientaDeInformacion.
"""
```

```
import os.path

from gvsiglib import *


def getMapContext():
  """
  Comprueba que el documanto activo es una vista y devuelve
  el mapContext asociado a ella
  """
  vista = gvSIG.getActiveDocument()
  if vista == None:
    print "No se puede acceder al documento activo."
    return None
  try:
    mapContext = vista.getModel().getMapContext()

  except Exception, e:
    print "El documento activo no parece ser una vista."
    print "Error %s %s" % (str(e.__class__),str(e))
    return None

  return mapContext

def estaMiCapa(layers):
  """
  Funcion encargada de comprobar si la capa de trabajo
  se encuentra en la coloeccion de capas indicada.
  """
  for n in range(layers.getLayersCount()):
    layer = layers.getLayer(n)
    if isinstance(layer,LayerCollection):
      if estaMiCapa(layer):
        return True
    if layer.getProperty("capaConMiInformacionEspecialDeAdemuz")==1:
      return True
  return False

def crearMiCapaDeTrabajo():
  """
  Funcion encargada de crear y cargar en la vista la capa
  de trabajo a partir del fichero csv
  """
  mapContext = getMapContext()
  if mapContext==None:
    return

  #comprobamos si ya esta cargada la capa de trabajo en la vista
  layers=mapContext.getLayers()
  if estaMiCapa(layers):
    return

  # Lo primero a hacer sera crear un dataSource basado en el fichero
  # csv
  dataSourceFactory=LayerFactory.getDataSourceFactory()

  fileName = os.path.join(
      gvSIG.getScriptsDirectory(),
      "..",
      "..",
      "miHerramientaDeInformacion",
      "municipiosAdemuz.csv"
      )
  dataSourceFactory.addFileDataSource("csv string", "ademuz", fileName)
  ds = dataSourceFactory.createRandomDataSource("ademuz")
  ds.start()

  # Crearemos el driver que gestiona la capa de eventos y lo enlazaremos con
  # la fuente de datos que acabamos de crear indicandole que columnas de esta
  # representan los puntos de la geomretria
  xFieldIndex = ds.getFieldIndexByName("X")
  yFieldIndex = ds.getFieldIndexByName("Y")
  AddEventThemeDriver=gvSIG.classForName("com.iver.gvsig.addeventtheme.AddEventThemeDriver")
  addEventThemeDriver = AddEventThemeDriver()
  addEventThemeDriver.setData(ds, xFieldIndex, yFieldIndex)

  # Crearemos ahora la nueva capa basada en este driver
  capa = None
  try:
    capa=gvSIG.getExtensionPoints().get("Layers").create("com.iver.cit.gvsig.fmap.layers.FLa
yerGenericVectorial")
    capa.setName("ademuz")
    capa.setDriver(addEventThemeDriver)
    capa.setProjection(mapContext.getProjection())
  except Exception, e:
    print "Se ha producido un error creando la capa. Error %s %s" %
(str(e.__class__),str(e))
    return

  # Una vez creada la capa se le añade una propiedad para reconocerla
  # como nuestra capa de trabajo
  capa.setProperty("capaConMiInformacionEspecialDeAdemuz",1)

  # La añadiremos a la lista de capas del mapContext de la vista
  mapContext.getLayers().addLayer(capa)

  # Indicamos al mapCOntext que se debe repintar
  mapContext.invalidate()

crearMiCapaDeTrabajo()
```

- When the script has been created, we can check whether when the tool is run, a points layer is loaded and the table data associated with the layer coincide with the previously created csv file data. **Check whether when the tool is run the correct layer is added.**



Nombre	Codigo	Comarca	Provincia
CASAS BAJ...	4609088	El Rincon ...	Valencia
CASAS AL...	4609087	El Rincon ...	Valencia
VALLANCA	4609252	El Rincon ...	Valencia
PUEBLA D...	4609201	El Rincon ...	Valencia
CASTIELFA...	4609092	El Rincon ...	Valencia
ADEMUZ	4609001	El Rincon ...	Valencia
TORREBAJA	4609242	El Rincon ...	Valencia

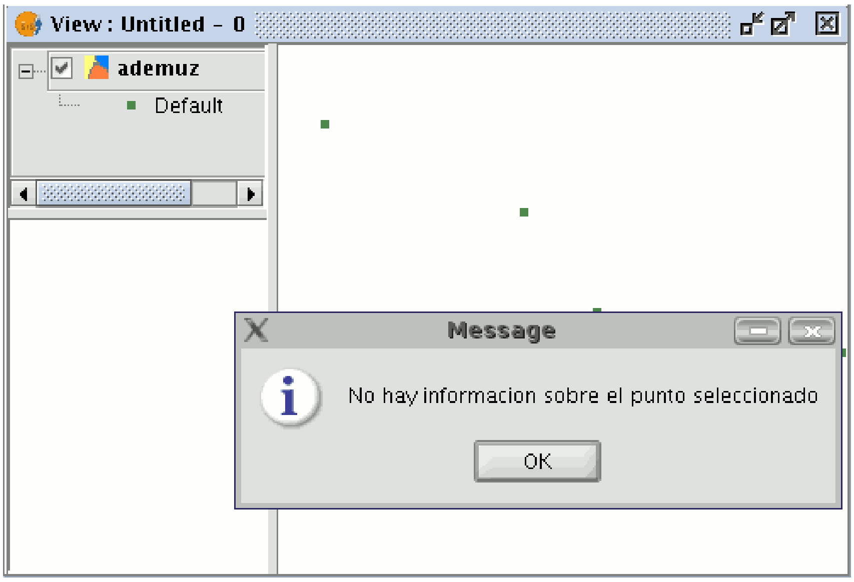
0 / 7 Total of selected records.

- Next, the script which will manage the information tool will be created. We will call this script “miHerramientadeInformacion.py” and it will contain the following:

```
from java.util import HashMap
from java.awt import Cursor, Point
from java.awt.event import MouseEvent

from gvsiglib import *

panel = None # Almacena una instancia de la ventana de informacion mostrada

class MyContentHandler(SaxContentHandler):
  """
  Parsea el xml asociado a la informacion de un punto
  transformandolo en un diccionario clave-valor

  FIXME: Solo funciona con capas vectoriales. Habria que arreglarlo
  para otro tipo de capas
  """
  def __init__(self, valores):
    self.valores = valores

  def startElement(self, nameSpace, localName, qName, attrs):
    valor = {} # Crea un diccionario vacio
    for i in range(attrs.getLength()):
      name=attrs.getQName(i)
      if name in ("",None):
        name=attrs.getLocalName(i)
      valor[name] = attrs.getValue(i)
    if len(valor) >0:
      self.valores.append(valor)

  def endElement(self, nameSpace, localName, qName):
    pass

  def characters(self, value, start, length):
    pass

class MiHerramientaDeInformacionListener(PointListener):
    """
    Esta clase recibe los eventos de clicado sobre el mapControl
    """
    def __init__(self,vista,mapControl):
        self._cursor = Cursor.getPredefinedCursor(Cursor.HAND_CURSOR)
        self._mapControl=mapControl
        self._vista=vista

    def getCursor(self):
      "@sig public java.awt.Cursor getCursor()"
      return self._cursor

    def cancelDrawing(self):
      "@sig public boolean cancelDrawing()"
      return False;

    def pointDoubleClick(self, event):
      "@sig public void pointDoubleClick(PointEvent event) throws BehaviorException"
      pass

    def point(self,event):
      "@sig public void point(PointEvent event) throws BehaviorException"
      global panel
      # Este evento es invocado cada vez que se produce un clic sobre el mapControl
      # estando nuestra herramienta activa

      # Lo primero que haremos sera obtener la lista de capas activas en el TOC
      capasSeleccionadas = self._mapControl.getMapContext().getLayers().getActives()

      # Si no hay ninguna capa seleccionada en el TOC no hacemos nada
      if len (capasSeleccionadas)<1:
        return
      # Si hay mas de una capa activa presentaremos la ventana de informacion predeterminada
      if len(capasSeleccionadas) >1:
        showInfo(self._vista, event.getPoint())
        return
      # Si no esta activa la capa de trabajo presentaremos la ventana de informacion 
predeterminada
      if capasSeleccionadas[0].getProperty("capaConMiInformacionEspecialDeAdemuz")!=1:
        showInfo(self._vista, event.getPoint())
        return
      # Si hemos llegado hasta aqui es que solo estaba activa en el TOC nuestra capa de
trabajo
      # procederemos a recuperar la informacion asociada al pùnto que se ha clicado
      tolerancia = self._mapControl.getViewPort().toMapDistance(10) # Transforma pixels a
unidades de mapa
      punto = Point(int(event.getPoint().getX()),int(event.getPoint().getY()))
      valores = [] # Crea una lista vacia

      capa = capasSeleccionadas[0]
      info = capa.getInfo(punto, tolerancia, None)
      for atributo in info:
        atributo.parse(MyContentHandler(valores))

      if len(valores)<1:
        showMessageDialog("No hay informacion sobre el punto seleccionado")
        return

      # Una vez hemos recogido los atributos los presentamos usando el panel de informacion
definido para ello
      parametros= HashMap()
      parametros.put("valores", valores)

      if panel != None:
        panel.close()

      panel=gvSIG.show("gvSIG/extensiones/miHerramientaDeInformacion/miPanelDeInformacion.xm
l","jython",325,150,parametros)

def showInfo(vista,point):
  """
  Muestra la ventana de informacion predeterminada para el punto indicado de la vista
  """
  mapControl = vista.getMapControl()
  infoListener = InfoListener(mapControl)
  event =
MouseEvent(vista,MouseEvent.BUTTON1,MouseEvent.ACTION_EVENT_MASK,MouseEvent.MOUSE_CLICKED,50
0,400,1,True)
  pointEvent = PointEvent(point,event)
  infoListener.point(pointEvent)


def main():
    vista = gvSIG.getActiveDocument()
    if vista == None:
      print "No se puede acceder al documento activo."
      return None
    try:
      mapContext = vista.getModel().getMapContext()
      mapControl = vista.getMapControl()

    except Exception, e:
      print "El documento activo no parece ser una vista."
```

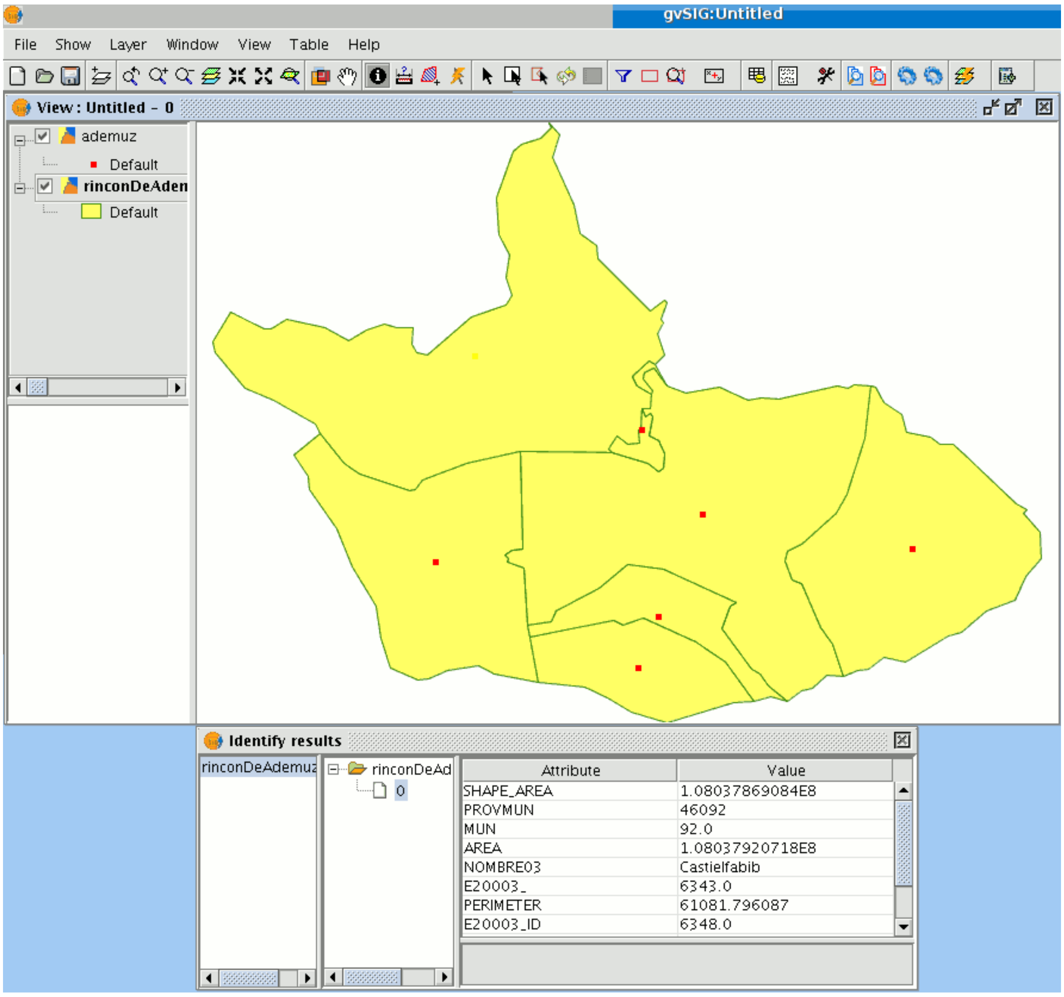
```
        print "Error %s %s" % (str(e.__class__),str(e))
        return None
      if mapContext == None:
        return

      # Si no hemos registrado en el mapControl nuestra herramienta de informacion
      # Creamos nuestro Listener y lo registramos en el mapControl
      if not mapControl.hasTool("MiHerramientaDeInformacion"):
        il=MiHerramientaDeInformacionListener(vista,mapControl)
        mapControl.addMapTool("MiHerramientaDeInformacion", PointBehavior(il))

      # Indicamos al mapControl que esta activa nuestra herramienta de informacion
      mapControl.setTool("MiHerramientaDeInformacion")

  main()
```

- If we start the application and run the tool to create the working layer and then we execute the point information request tool when the layer has been loaded, we will be able to check that if we do not click on the points or the tolerance area of this point, a window will appear to indicate that there is no information associated with the selected point.



No hay informacion sobre el punto seleccionado

- We have used a polygon layer for this example with places that make up the municipal area of the Rincón de Ademuz. If the “miHerramientadeInformacion” utility is activated with another active layer which is not the layer we have created previously, and we request information about it, the standard gvSIG information window will appear.



Attribute	Value
SHAPE_AREA	1.08037869084E8
PROVMUN	46092
MUN	92.0
AREA	1.08037920718E8
NOMBRE03	Castielfabib
E20003_	6343.0
PERIMETER	61081.796087
E20003_ID	6348.0

- We then need to create another new file in the same folder we created the previous files in. This file will be called “miPanelDeInformacion.xml” and it will define the window we will use to show our layer information. The file will contain the following:

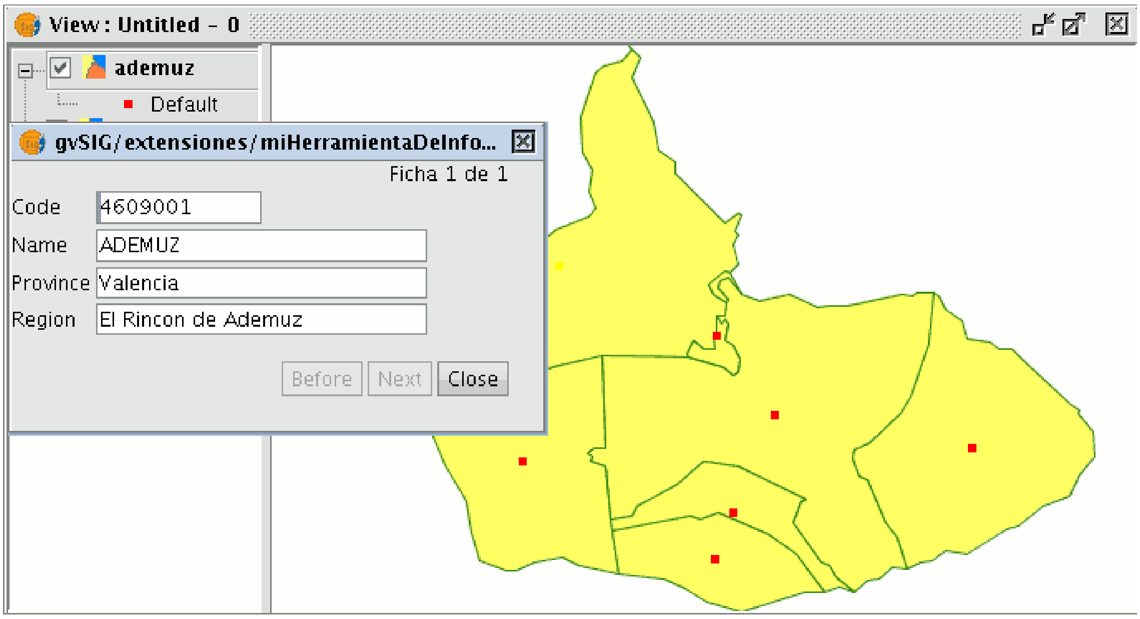
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel columns="1" gap="3" >
  <script language="jython" method="init" src="miPanelDeInformacion.py"/>
  <panel halign="right" >
        <label name="lblContadorDeFichas" text="Ficha 1 de X"/>
    </panel>
    <panel columns="2" gap="3" height="100" valign="top" width="300">
```

```
            <label name="lblCodigo" text="Codigo"/>
            <textfield halign="left" height="20" name="txtCodigo" width="100"/>
            <label name="lblNombre" text="Nombre"/>
            <textfield height="20" name="txtNombre" width="200"/>
            <label name="lblProvincia" text="Provincia"/>
            <textfield name="txtProvincia"/>
            <label name="lblComarca" text="Comarca"/>
            <textfield name="txtComarca"/>
        </panel>
        <panel gap="3" halign="right" valign="center">
            <button action="clickAnterior(thinlet)" name="btnAnterior" text="Anterior"/>
            <button action="clickSiguiente(thinlet)" name="btnSiguiente" text="Siguiente"/>
            <button action="thinlet.closeWindow()" name="btnCerrar" text="Cerrar"/>
        </panel>
    </panel>
```

- The next step is to create a file called “miPanelDeInformacion.py”, which will be responsible for managing how the data in the window we have previously created will be displayed. This file will contain the following:

```
"""
Modulo que se encarga de gestionar la visualizacion de los datos en la
ventana miPaneldeInformacion.xml
"""
from gvsiglib import *

current=0
valores=None

def clickSiguiente(thinlet):
  rellenarFicha(current+1)

def clickAnterior(thinlet):
  rellenarFicha(current-1)

def setValores(thinlet, misValores):
  global valores

  valores = misValores
  rellenarFicha(0)

def rellenarFicha(indice):
  """
  Carga los datos en los controles asignados y habilita los botones de
  siguiente y anterior segun proceda
  """
  global current

  if indice <0:
    return
  if indice >= len(valores):
    return

  current = indice

  valor=valores[current]

  thinlet.setString(lblContadorDeFichas,"text","Ficha %s de %s" %(current +1 ,len(valores)))
  thinlet.setString(txtCodigo,"text",valor.get("Codigo",""))
  thinlet.setString(txtNombre,"text",valor.get("Nombre",""))
  thinlet.setString(txtProvincia,"text",valor.get("Provincia",""))
  thinlet.setString(txtComarca,"text",valor.get("Comarca",""))
  if current <1:
    thinlet.setBoolean(btnAnterior,"enabled",False)
  else:
    thinlet.setBoolean(btnAnterior,"enabled",True)

  if current >=len(valores)-1:
    thinlet.setBoolean(btnSiguiente,"enabled",False)
  else:
    thinlet.setBoolean(btnSiguiente,"enabled",True)

# Recoge los parametros pasados al thinlet en la llamada a la funcion
# gvSIG.show que recibimos en la variable global params para inicializar
# los datos del gui
setValores(thinlet,params.get("valores"))
```

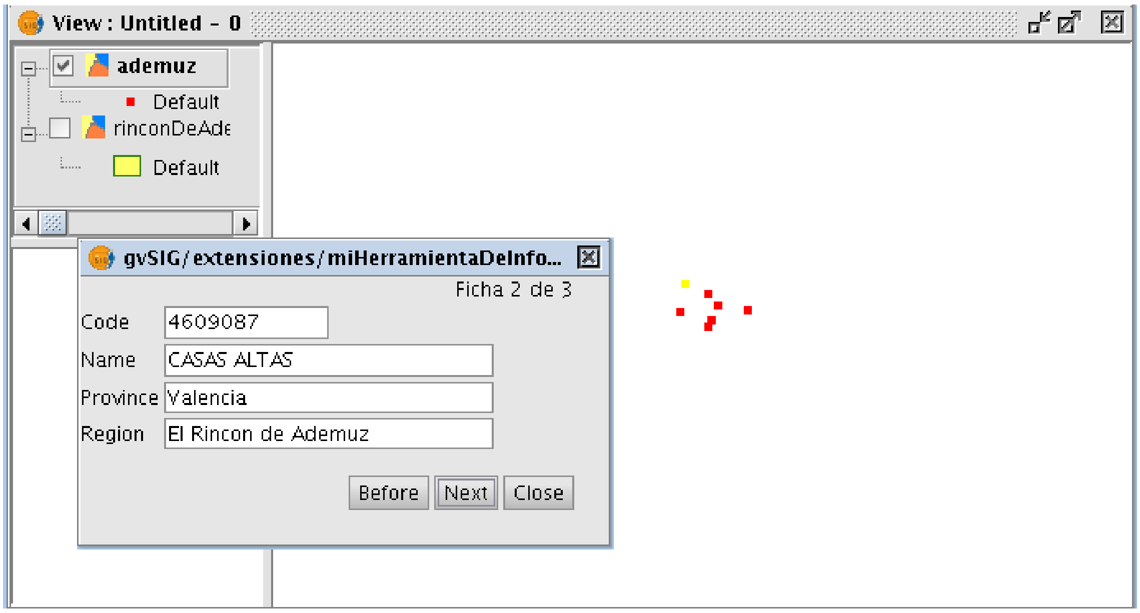
- At this point, we can start gvSIG and use our tools. To check whether it works, activate miHerramientadeInformacion with the selected points layer we have created and then click on one of the points. The information window we have created will open.



Ficha 1 de 1

Code 4609001
Name ADEMUZ
Province Valencia
Region El Rincon de Ademuz

- If we zoom out of the layer, so that the points are superposed over each other, and we request information using our tool, the information about the points that are located inside the defined tolerance area will be loaded in the window. The “Previous” and “Next” buttons are enabled to view the information when there are multiple elements.



Ficha 2 de 3

Code 4609087
Name CASAS ALTAS
Province Valencia
Region El Rincon de Ademuz

3 Annexes. Source code

3.1 Centring a view on a point

3.1.1 config.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<plugin-config>
        <libraries library-dir="../org.gvsig.scripting"/>
        <depends plugin-name="org.gvsig.scripting"/>
                <resourceBundle name="text"/>
        <extensions>
                <extension class-name="org.gvsig.scripting.ScriptingExtension"
                        description="Extension de soporte para Scripts de usuario."
                        active="true">
                        <menu text="Archivo/Scripting/Centrar vista en un punto"
                          tooltip="Centrar la vista en un punto"
                              action-command =
"show(fileName='gvSIG/extensiones/centrarVistaSobreUnPunto/centrarVistaSobreUnPunto.xml',language='j
ython',title='Centrar la vista a un punto',width=210,height=86)"
                                icon="images/default.png"
                                position="55"
                                />
                        <menu text="Archivo/Scripting/Borrar puntos"
                              tooltip="Borrar puntos"
                              action-command =
"run(fileName='gvSIG/extensiones/centrarVistaSobreUnPunto/limpiarElGraphics.py',language='jython')"
                              icon="images/default.png"
                              position="56"
                                />
                </extension>
        </extensions>
</plugin-config>
```

3.1.2 centrarVistaSobreUnPunto.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel columns="3" gap="3">
    <script language="jython" method="init" src="centrarVistaSobreUnPunto.py"/>
    <label colspan="3" text="Coordenadas para centrar la vista"/>
    <label colspan="2" halign="right" text="Coordenada x:"/>
    <textfield name="txtX"/>
    <label colspan="2" halign="right" text="Coordenada y:"/>
    <textfield name="txtY"/>
    <panel colspan="3" gap="2" halign="right">
        <button halign="right" name="botAplicar" text="Aplicar" action="clickAplicar(thinlet)"/>
        <button halign="right" name="botCerrar" text="Cerrar"      action="thinlet.closeWindow()"/>

    </panel>
</panel>
```

3.1.3 centrarVistaSobreUnPunto.py

```
import java.awt.geom.Point2D as Point2D
import java.awt.geom.Rectangle2D as Rectangle2D

import sys

from gvsiglib import *

mapContext = None

def getMapContext():
    view = gvSIG.getActiveDocument()
    if view == None:
      print "No se puede acceder al documento activo."
      return None
    try:
      mapContext = view.getModel().getMapContext()

    except Exception, e:
      print "El documento activo no parece ser una vista."
      print "Error %s %s" % (str(e.__class__),str(e))
      return None

    return mapContext

mapContext = getMapContext()

def clickAplicar(thinlet):

    global mapContext

    if mapContext == None:
      print "No se puede acceder al documento activo."
      return

    if mapContext.getLayers().getLayersCount() < 1:
      print "El documento activo no tiene capas disponibles."
      return
    x = float(thinlet.getString(txtX, "text"))
    y = float(thinlet.getString(txtY, "text"))
    center = zoomToCoordinates(mapContext, x,y)
    drawPoint(mapContext,center)

def zoomToCoordinates(mapContext, x,y):
  try:
    oldExtent = mapContext.getViewPort().getAdjustedExtent()
    oldCenterX = oldExtent.getCenterX()
    oldCenterY = oldExtent.getCenterY()
    center=Point2D.Double(x,y)
    movX = x-oldCenterX
    movY = y-oldCenterY
    upperLeftCornerX = oldExtent.getMinX()+movX
    upperLeftCornerY = oldExtent.getMinY()+movY
    width = oldExtent.getWidth()
    height = oldExtent.getHeight()
    extent = Rectangle2D.Double(upperLeftCornerX, upperLeftCornerY, width, height)
    mapContext.getViewPort().setExtent(extent)
    return center
  except ValueError, e:
    print "Se ha producido un error realizando zoom a las coordenadas (%s,%s). Error  %s, %s" % (
      repr(x),
      repr(y),
      str(e.__class__),
      str(e)
    )
    return None

def drawPoint(mapContext, center, color=None):
    """
    Esta función pintará un punto sobre la capa de gráficos
    asociada al mapContext.
    Todo mapContext además de las capas que tenga cargadas dispone
    una capa graphics sobre la que dibujar elementos gráficos.
    """

    if color == None:
        import java.awt.Color as Color
        color = Color.blue

    layer=mapContext.getGraphicsLayer()
    layer.clearAllGraphics()
    theSymbol = FSymbol(FConstant.SYMBOL_TYPE_POINT,color)
    idSymbol = layer.addSymbol(theSymbol)
    geom = ShapeFactory.createPoint2D(center.getX(),center.getY())
    theGraphic = FGraphic(geom, idSymbol)
    layer.addGraphic(theGraphic)

def elDocumentoActivoEsUnaVistaValida():
    global mapContext

    if mapContext == None:
      print "El documento activo nop parece ser una vista"
      return False

    if mapContext.getLayers().getLayersCount() < 1:
      print "El documento activo no tiene capas disponibles."
      return False
    return True

if activeDocumentIsAValidView():
  thinlet.setBoolean(botAplicar,"enabled",True)
else:
```

```
thinlet.setBoolean(botAplicar,"enabled",False)
```

3.1.4 limpiarElGraphics.py

```
from gvsiglib import *

def main():
    view = gvSIG.getActiveDocument()
    if view == None:
      print "No se puede acceder al documento activo."
      return None
    try:
      mapContext = view.getModel().getMapContext()
      mapControl = view.getMapControl()

    except Exception, e:
      print "El documento activo no parece ser una vista."
      print "Error %s %s" % (str(e.__class__),str(e))
      return None
    if mapContext == None:
      return
    layer=mapContext.getGraphicsLayer()
    layer.clearAllGraphics()
    mapContext.invalidate()

main()
```

3.2 *My information tool*

3.2.1 config.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<plugin-config>
       <libraries library-dir="../org.gvsig.scripting"/>
       <depends plugin-name="org.gvsig.scripting"/>
               <resourceBundle name="text"/>
       <extensions>
               <extension class-name="org.gvsig.scripting.ScriptingExtension"
                      description="Extension de soporte para Scripts de usuario."
                      active="true">
                      <menu text="Archivo/Scripting/Mi herramienta de información"
                            tooltip="Mi herramienta de información"
                            action-
command="run(fileName='gvSIG/extensiones/miHerramientaDeInformacion/miHerramientaDeInformacion.py',l
anguage='jython')"
            icon="images/default.png"
                              position="55"
                              />
      <menu text="Archivo/Scripting/Crear capa de trabajo"
            tooltip="Crea una capa para utilizarla con la utilidad 'Mi herramienta de informacion'"
            action-
command="run(fileName='gvSIG/extensiones/miHerramientaDeInformacion/anyadirMiCapaDeTrabajo.py',langu
age='jython')"
            icon="images/default.png"
            position="55"
        />
      <tool-bar name="Scripting">
        <action-tool icon="images/default.png"
                     action-
command="run(fileName='gvSIG/extensiones/miHerramientaDeInformacion/anyadirMiCapaDeTrabajo.py',langu
age='jython')" tooltip="Crea una capa para utilizarla con la utilidad 'Mi herramienta de
informacion'"
                     />
        <action-tool icon="images/default.png"
                     action-
command="run(fileName='gvSIG/extensiones/miHerramientaDeInformacion/miHerramientaDeInformacion.py',l
anguage='jython')" tooltip="Mi herramienta de información"
                     />
      </tool-bar>
               </extension>
       </extensions>
</plugin-config>
```

3.2.2 municipiosAdemuz.csv

```
Nombre;Codigo;Comarca;Provincia;X;Y
CASAS BAJAS;4609088;El Rincon de Ademuz;Valencia;648522.72;4431068.44
CASAS ALTAS;4609087;El Rincon de Ademuz;Valencia;649319.84;4433082.21
VALLANCA;4609252;El Rincon de Ademuz;Valencia;640425.70;4435263.80
PUEBLA DE SAN MIGUEL;4609201;El Rincon de Ademuz;Valencia;659430.60;4435809.18
CASTIELFABIB;4609092;El Rincon de Ademuz;Valencia;641977.97;4443528.63
ADEMUZ;4609001;El Rincon de Ademuz;Valencia;651081.88;4437193.65
TORREBAJA;4609242;El Rincon de Ademuz;Valencia;648648.53;4440549.03
```

3.2.3 miHerramientaDeInformacion.py

```
from java.util import HashMap
from java.awt import Cursor, Point
from java.awt.event import MouseEvent

from gvsiglib import *

panel = None # Almacena una instancia de la ventana de informacion mostrada

class MyContentHandler(SaxContentHandler):
  """
  Parsea el xml asociado a la informacion de un punto
  transformandolo en un diccionario clave-valor

  FIXME: Solo funciona con capas vectoriales. Habria que arreglarlo
  para otro tipo de capas
  """
  def __init__(self, values):
    self.values = values

  def startElement(self, nameSpace, localName, qName, attrs):
    value = {} # Crea un diccionario vacio
    for i in range(attrs.getLength()):
      name=attrs.getQName(i)
      if name in ("",None):
        name=attrs.getLocalName(i)
      value[name] = attrs.getValue(i)
    if len(value) >0:
      self.values.append(value)

  def endElement(self, nameSpace, localName, qName):
    pass

  def characters(self, value, start, length):
    pass

class MyInformationToolListener(PointListener):
    """
    Esta clase recibe los eventos de click sobre el mapControl
    """
    def __init__(self,view,mapControl):
        self._cursor = Cursor.getPredefinedCursor(Cursor.HAND_CURSOR)
        self._mapControl=mapControl
        self._view=view

    def getCursor(self):
      "@sig public java.awt.Cursor getCursor()"
      return self._cursor

    def cancelDrawing(self):
      "@sig public boolean cancelDrawing()"
      return False;

    def pointDoubleClick(self, event):
      "@sig public void pointDoubleClick(PointEvent event) throws BehaviorException"
      pass

    def point(self,event):
      "@sig public void point(PointEvent event) throws BehaviorException"
      global panel
      # Este evento es invocado cada vez que se produce un clic sobre el mapControl
      # estando nuestra herramienta activa

      # Lo primero que haremos sera obtener la lista de capas activas en el TOC
      selectedLayers = self._mapControl.getMapContext().getLayers().getActives()

      # Si no hay ninguna capa seleccionada en el TOC no hacemos nada
      if len (selectedLayers)<1:
        return
      # Si hay mas de una capa activa presentaremos la ventana de informacion predeterminada
      if len(selectedLayers) >1:
        showInfo(self._view, event.getPoint())
        return
      # Si no esta activa la capa de trabajo presentaremos la ventana de informacion predeterminada
      if selectedLayers[0].getProperty("capaConMiInformacionEspecialDeAdemuz")!=1:
        showInfo(self._view, event.getPoint())
        return

      # Si hemos llegado hasta aqui es que solo estaba activa en el TOC nuestra capa de trabajo
      # procederemos a recuperar la informacion asociada al pùnto que se ha clicado
      tolerance = self._mapControl.getViewPort().toMapDistance(10) # Transforma pixels a unidades de
mapa
      thePoint = Point(int(event.getPoint().getX()),int(event.getPoint().getY()))
      values = [] # Crea una lista vacia

      layer = selectedLayers[0]
      info = layer.getInfo(thePoint, tolerance, None)
      for atribute in info:
        atribute.parse(MyContentHandler(values))

      if len(values)<1:
        showMessageDialog("No hay informacion sobre el punto seleccionado")
        return

      # Una vez hemos recogido los atributos los presentamos usando el panel de informacion definido
para ello
      params= HashMap()
      params.put("values", values)

      if panel != None:
        panel.close()

      panel=gvSIG.show("gvSIG/extensiones/miHerramientaDeInformacion/miPanelDeInformacion.xml","jyth
on",325,150,params)

def showInfo(view,point):
  """
  Muestra la ventana de informacion predeterminada para el punto indicado de la vista
  """
  mapControl = view.getMapControl()
  infoListener = InfoListener(mapControl)
  event =
MouseEvent(view,MouseEvent.BUTTON1,MouseEvent.ACTION_EVENT_MASK,MouseEvent.MOUSE_CLICKED,500,400,1,T
rue)
  pointEvent = PointEvent(point,event)
  infoListener.point(pointEvent)


def main():
    view = gvSIG.getActiveDocument()
    if view == None:
      print "No se puede acceder al documento activo."
      return None
    try:
      mapContext = view.getModel().getMapContext()
      mapControl = view.getMapControl()

    except Exception, e:
      print "El documento activo no parece ser una vista."
      print "Error %s %s" % (str(e.__class__),str(e))
      return None
    if mapContext == None:
      return

    # Si no hemos registrado en el mapControl nuestra herramienta de informacion
    # Creamos nuestro Listener y lo registramos en el mapControl
    if not mapControl.hasTool("MyInformationToolListener"):
      il=MyInformationToolListener(view,mapControl)
      mapControl.addMapTool("MyInformationToolListener", PointBehavior(il))

    # Indicamos al mapControl que esta activa nuestra herramienta de informacion
    mapControl.setTool("MyInformationToolListener")

main()
```

3.2.4 miPanelDeInformacion.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generated by ThinG, the Thinlet GUI editor -->
<panel columns="1" gap="3" >
  <script language="jython" method="init" src="miPanelDeInformacion.py"/>
  <panel halign="right" >
        <label name="lblContadorDeFichas" text="Ficha 1 de X"/>
    </panel>
    <panel columns="2" gap="3" height="100" valign="top" width="300">
        <label name="lblCodigo" text="Codigo"/>
        <textfield halign="left" height="20" name="txtCodigo" width="100"/>
        <label name="lblNombre" text="Nombre"/>
        <textfield height="20" name="txtNombre" width="200"/>
        <label name="lblProvincia" text="Provincia"/>
        <textfield name="txtProvincia"/>
        <label name="lblComarca" text="Comarca"/>
        <textfield name="txtComarca"/>
    </panel>
    <panel gap="3" halign="right" valign="center">
        <button action="clickAnterior(thinlet)" name="btnAnterior" text="Anterior"/>
        <button action="clickSiguiente(thinlet)" name="btnSiguiente" text="Siguiente"/>
        <button action="thinlet.closeWindow()" name="btnCerrar" text="Cerrar"/>
    </panel>
</panel>
```

3.2.5 miPanelDeInformacion.py

```
"""
Modulo que se encarga de gestionar la visualizacion de los datos en la
ventana miPaneldeInformacion.xml
"""
from gvsiglib import *

current=0
values=None


def clickSiguiente(thinlet):
  fillCard(current+1)

def clickAnterior(thinlet):
  fillCard(current-1)

def setValues(thinlet, myValues):
  global values

  values = myValues
  fillCard(0)

def fillCard(index):
  """
  Carga los datos en los controles asignados y habilita los botones de
  siguiente y anterior segun proceda
  """
  global current

  if index <0:
    return
  if index >= len(values):
    return

  current = index

  value=values[current]

  thinlet.setString(lblContadorDeFichas,"text","Ficha %s de %s" %(current +1 ,len(values)))
  thinlet.setString(txtCodigo,"text",value.get("Codigo",""))
  thinlet.setString(txtNombre,"text",value .get("Nombre",""))
  thinlet.setString(txtProvincia,"text",value .get("Provincia",""))
  thinlet.setString(txtComarca,"text",value .get("Comarca",""))
  if current <1:
    thinlet.setBoolean(btnAnterior,"enabled",False)
  else:
    thinlet.setBoolean(btnAnterior,"enabled",True)

  if current >=len(values)-1:
    thinlet.setBoolean(btnSiguiente,"enabled",False)
  else:
    thinlet.setBoolean(btnSiguiente,"enabled",True)

# Recoge los parametros pasados al thinlet en la llamada a la funcion
# gvSIG.show que recibimos en la variable global params para inicializar
# los datos del gui
setValues(thinlet,params.get("values"))
```

3.2.6 anyadirMiCapaDeTrabajo.py

```
"""
Script que genera una capa de puntos a partir de un fichero csv
(municipiosAdemuz.csv) que se utiliza para trabajar con la herramienta
miHerramientaDeInformacion.
"""

import os.path

from gvsiglib import *

def getMapContext():
  """
  Comprueba que el documanto activo es una vista y devuelve
  el mapContext asociado a ella
  """
  view = gvSIG.getActiveDocument()
  if view == None:
    print "No se puede acceder al documento activo."
    return None
  try:
    mapContext = view.getModel().getMapContext()
  except Exception, e:
    print "El documento activo no parece ser una vista."
    print "Error %s %s" % (str(e.__class__),str(e))
    return None

  return mapContext

def isThereMyLayer(layers):
  """
  Funcion encargada de comprobar si la capa de trabajo
  se encuentra en la coloeccion de capas indicada.
  """
  for n in range(layers.getLayersCount()):
    layer = layers.getLayer(n)
    if isinstance(layer,LayerCollection):
      if isThereMyLayer(layer):
        return True
    if layer.getProperty("capaConMiInformacionEspecialDeAdemuz")==1:
      return True
  return False

def createMyWorkLayer():
  """
  Funcion encargada de crear y cargar en la vista la capa
  de trabajo a partir del fichero csv
  """
  mapContext = getMapContext()
  if mapContext==None:
    return
  #comprobamos si ya esta cargada la capa de trabajo en la vista
  layers=mapContext.getLayers()
  if isThereMyLayer(layers):
    return

  # Lo primero a hacer sera crear un dataSource basado en el fichero
  # csv
  dataSourceFactory=LayerFactory.getDataSourceFactory()

  fileName = os.path.join(
      gvSIG.getScriptsDirectory(),
      "..",
      "..",
      "miHerramientaDeInformacion",
      "municipiosAdemuz.csv"
      )
  dataSourceFactory.addFileDataSource("csv string", "ademuz",fileName)
  ds = dataSourceFactory.createRandomDataSource("ademuz")
  ds.start()

  # Crearemos el driver que gestiona la capa de eventos y lo enlazaremos con
  # la fuente de datos que acabamos de crear indicandole que columnas de esta
  # representan los puntos de la geomretria
  xFieldIndex = ds.getFieldIndexByName("X")
  yFieldIndex = ds.getFieldIndexByName("Y")
  AddEventThemeDriver=gvSIG.classForName("com.iver.gvsig.addeventtheme.AddEventThemeDriver")
  addEventThemeDriver = AddEventThemeDriver()
  addEventThemeDriver.setData(ds, xFieldIndex, yFieldIndex)

  # Crearemos ahora la nueva capa basada en este driver
  capa = None
  try:
    myLayer=gvSIG.getExtensionPoints().get("Layers").create("GenericVectorial")
    myLayer.setName("ademuz")
    myLayer.setDriver(addEventThemeDriver)
    myLayer.setProjection(mapContext.getProjection())
  except Exception, e:
    print "Se ha producido un error creando la capa. Error %s %s" % (str(e.__class__),str(e))
    return

  # Una vez creada la capa se le añade una propiedad para reconocerla
  # como nuestra capa de trabajo
  myLayer.setProperty("capaConMiInformacionEspecialDeAdemuz",1)

  # La añadiremos a la lista de capas del mapContext de la vista
  mapContext.getLayers().addLayer(myLayer)

  # Indicamos al mapContext que se debe repintar
  mapContext.invalidate()

createMyWorkLayer()
```

GNU GENERAL PUBLIC LICENSE Version 2

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.,
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS

TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
    Gnomovision version 69, Copyright (C) year name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> <signature of Ty Coon>, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.