SUMO devoloper manual

Francois-Xavier Thoorens

March 5, 2010

Contents

Chapter 1

Introduction

This manual is dedicated for the people that will develop the core functionnality of SUMO as well as plugins to the platform. It is not meant to normal end users. SUMO as a package is divided in 3 parts:

1. The Image Reader Library project called **GeoImage** (package org.geoimage)
2. The Core Platform project called **GeoImageViewer** (package org.geoimage.viewer)
3. A set of plugins, including the VDS analysis project called **GeoImageAnalysis**

The manual will extensively describes those three parts. It will start by describing how to set up the environment to develop SUMO, including the needed libraries

Chapter 2

Setting Up Your System

SUMO is entirely developed in JAVA. You need an IDE to be able to compile properly the project. Build system is based on ANT, as generated by the netbeans IDE (as of today version 6.8). The source code is back up by a Subversion (SVN) repository hosted in JRC. The source is not open yet to people outside JRC.

2.1 Environement Requirements

2.1.1 Minimum

- RAM: 1G
- CPU: 1.8Ghz
- Graphic Card: supporting OpenGL 2.0. Don't forget to update your drivers. More information at `http://worldwind.arc.nasa.gov/java/video.html`

2.1.2 Recommended

- RAM: 2G
- CPU: dual core 1.8 Ghz
- Graphic Card: NVidia GForce

2.2 The IDE

- Download and install the latest Java JDK, **not the JRE**, you can find at `http://java.sun.com`.
- Download and install the last version of Netbeans, at the time of writing versin 6.8, check it out here `http://www.netbeans.org`. You need to download the Java developer version (recommended).
- Download and install also the **JOGL netbeans pack** you can find at `http://plugins.netbeans.org/PluginPortal/faces/PluginDetailPage.jsp?pluginid=3260`.
- if you are on linux system, be sure you have subversion installed (most of the time it is present by default). On Ubuntu the command is *sudo apt-get install subversion.* For windows, Netbeans has a plugin that is automatically installed the first time you decide to check out a subversion repository. You may also use your own one like **Tortoise SVN**.

You may use Eclipse IDE as well, since the build system is based on ANT, but you will have to find out manually how to bing JOGL in Eclipse.

2.3 Checking out the Subversion Repository

On netbeans 6.8 in the go to **team** > **subversion** > **checkout**. The "check out" URL is `https://ipsc-trac.jrc.it/subversion/fish`. Use the login/password you have to log to the JRC email system since the Suversion repository is using LDAP for authentication. If you have problem with failed authentication, please contact the ipsc-trac.jrc.it administrator (at the time of writing **francesco.storti@ext.jrc.ec.europa.eu**). Leave other parameters blank. In the next step, checkout out the the four following projects:

1. trunk/sumo/trunk/GeoImage
2. trunk/sumo/trunk/GeoImageViewer
3. trunk/sumo/trunk/GeoImageAnalysis
4. trunk/sumo/trunk/FileFinder (because it contains one needed jar file)

Leave other parameters as default.

> There are some extra projects in trunk/sumo/trunk that are plugins you may checkout later.

Netbeans proceed to the check out. It will ask to open automatically the projects. Answer yes.

2.4 Configuring your project

The libary you needs are downloaded from the SVN. However you have to properly link the libraries in your project:

2.4.1 Setting your OS

Also it is JAVA, the JOGL library is using native libraries and need to know your default OS. In the project.properties file you can find in GeoImageViewer/nbproject/project.properties, navigate to **natives.platform** and set the values properly: **linux-i586** or **windows-i586**. Other OS are also suported, please refer to JOGL Netbeans Pack manual.

2.4.2 Geotools

In the project properties (**right click on the project** > **properties**) switch to library tabs and add a new Library you name Geotools. Add the jars contained in GeoImageViewer/lib/geotools-2.5.1/.

2.4.3 Other Libraries

Open your GeoImageViewer/nbproject/project.properties file and be sure that the following variables are set correctly:

file.reference.commons-beanutils-bean-collections.jar=lib/commons-beanutils-bean-collections.jar

file.reference.commons-beanutils-core.jar=lib/commons-beanutils-core.jar

file.reference.commons-beanutils.jar=lib/commons-beanutils.jar

file.reference.commons-collections-3.1.jar=lib/commons-collections-3.1.jar

```
file.reference.commons-digester-1.8.jar=lib/commons-digester-1.8.jar

file.reference.commons-fileupload-1.2.1.jar=lib/apache/commons-fileupload-1.2.1/lib/commons-fileupload-1.2.1.jar

file.reference.commons-logging.jar=lib/commons-logging.jar

file.reference.commons-net-2.0.jar=lib/apache/commons-net-2.0/commons-net-2.0.jar

file.reference.commons-net-ftp-2.0.jar=lib/apache/commons-net-2.0/commons-net-ftp-2.0.jar

file.reference.dsn.jar=lib/javamail-1.4.1/lib/dsn.jar

file.reference.FengGUI-src.zip=lib/FengGUI-src.zip

file.reference.FengGUI.jar=lib/FengGUI.jar

file.reference.gekmllib.jar=lib/gekmllib.jar

file.reference.georss-rome-0.9.8.jar=../GeoImage/lib/georss/georss-rome-0.9.8.jar

file.reference.georss-rome-src-0.9.8.jar=../GeoImage/lib/georss/georss-rome-src-0.9.8.jar

file.reference.h2-1.2.121.jar=../FileFinder/lib/h2-1.2.121.jar

file.reference.imap.jar=lib/javamail-1.4.1/lib/imap.jar

file.reference.mail.jar=lib/javamail-1.4.1/mail.jar

file.reference.mailapi.jar=lib/javamail-1.4.1/lib/mailapi.jar

file.reference.pop3.jar=lib/javamail-1.4.1/lib/pop3.jar

file.reference.RepositoryManager.jar=../RepositoryManager/dist/RepositoryManager.jar

file.reference.rome-0.9.jar=../GeoImage/lib/georss/rome-0.9.jar

file.reference.smtp.jar=lib/javamail-1.4.1/lib/smtp.jar

file.reference.worldwind.jar=lib/worldwind.jar

...

javac.classpath= $libs.JOGL.classpath:\

$libs.GLUEGEN-RT.classpath:\

$reference.GeoImage.jar:\

$reference.GeoImageAnalysis.jar:\

$libs.Geotools.classpath:\

$libs.swing-app-framework.classpath:\

$libs.absolutelayout.classpath:\

$libs.swing-layout.classpath:\

$file.reference.commons-beanutils-bean-collections.jar:\

$file.reference.commons-beanutils-core.jar:\

$file.reference.commons-beanutils.jar:\

$file.reference.commons-collections-3.1.jar:\

$file.reference.commons-digester-1.8.jar:\

$file.reference.commons-logging.jar:\

$file.reference.FengGUI-src.zip:\

$file.reference.FengGUI.jar:\

$file.reference.gekmllib.jar:\

$file.reference.worldwind.jar:\

$file.reference.mail.jar:\

$file.reference.dsn.jar:\

$file.reference.imap.jar:\

$file.reference.mailapi.jar:\

$file.reference.pop3.jar:\

$file.reference.smtp.jar:\

$file.reference.commons-net-2.0.jar:\

$file.reference.commons-net-ftp-2.0.jar:\

$file.reference.commons-fileupload-1.2.1.jar:\

$file.reference.georss-rome-0.9.8.jar:\

$file.reference.georss-rome-src-0.9.8.jar:\
$file.reference.rome-0.9.jar:\
$file.reference.h2-1.2.121.jar
```

If your computer does not have enough RAM you may change as well:
run.jvmargs=-Xmx1024m
to a value like -Xmx512m

Chapter 3

GeoImage Library

This is the basic underlying library able to access different satellite formats. The supported formats are:

- Envisat Images and ERS2 (same format)
- Radarsat1 (CEOS format)
- Radarsat2
- Terrasar-X
- Cosmo-Skymed
- A subset of Geotiff images

3.1 The interface GeoImageReader

The main class handling the image is the interface **GeoImageReader** presenting a common API for all formats:

```
package org.geoimage;

import java.io.File;
import java.util.List;
import org.geoimage.utils.IProgress;

/**
 * Interface to be implemented by all the readers for "geographic" images
 */
public interface GeoImageReader extends GeoMetadata {

    /**
     * version of the reader
     */
    public static final String version = "1.0beta";

    /**
     *
     * @return the current displayed image band
     */
    public int getBand();

    /**
     *
     * @return the image bounding box in lattitude and longitude coordinates
     */
    public List<double[]> getFrameLatLon();
```

```
/**
 *
 * @return the name of the image
 */
public String getName();

/**
 *
 * @return the width of the image raster in pixels
 */
public int getWidth();

/**
 *
 * @return the length of the image in pixels
 */
public int getHeight();

/**
 *
 * @return the number of bands of the image
 */
public int getNBand();

/**
 *
 * @return the format of the image, should be the Class name
 */
public String getFormat();

/**
 * @return a String with a human readable description of the image
 */
public String getDescription();

/**
 *
 * @return the number of bytes for each band. So far it is supposed
 * to be the same for each band
 */
public int getNumberOfBytes();

/**
 * @param oneBand whether it is for 1 band or all together
 * @return One of the BufferedImage.TYPE_....
 * @parameter: oneBand means one single access type (in case of 16 bits per band like Envisat)
 * if false for multi bands access, the user will now that he should downsize each band to 8 ↩
     bits
 */
public int getType(boolean oneBand);

/**
 *  Gets the WKT form of the projection system as defined by OpenGIS
 * @see org.geoimage.AffineGeoTransform
 * @see org.geoimage.GcpsGeoTransform
 * @return an implementation of Geotransform.
 */
public GeoTransform getGeoTransform();

/**
 *  @return the tie points (or gcps) of the image.
 */
public List<Gcp> getGcps();

/**
 *  Gets the access rights:<br>&quot;r&quot; = read only<br>&quot;rw&quot; = read/write
 * @return
 */
public String getAccessRights();

/**
 *
 * @return the Files absolute path that belongs to the image
 */
public String[] getFilesList();
```

```
/**
 * Initialises the image
 * @return whether the image has been properly initialised , thus readable
 */
public boolean initialise();

/**
 * Sets the file. Should used BEFORE the call to initialise()
 * @param imageFile
 */
public void setFile(File imageFile);

/**
 * Reads the data in int[]. Access the pixels value (x,y): data[x+y*width].
 * A call to preloadLineTile can be considered to improve memory management.
 * @param x
 * @param y
 * @param width
 * @param height
 * @return
 */
public int[] readTile(int x, int y, int width, int height);

/**
 *
 * @param x
 * @param y
 * @param width
 * @param height
 * @param outWidth
 * @param outLength
 * @param filter
 * @return
 */
public int[] readAndDecimateTile(int x, int y, int width, int height, int outWidth, int ↩
    outLength, boolean filter);

/**
 *
 * @param x
 * @param y
 * @param width
 * @param height
 * @param scalingFactor
 * @param filter
 * @param progressbar
 * @return
 */
public int[] readAndDecimateTile(int x, int y, int width, int height, double scalingFactor, ↩
    boolean filter, IProgress progressbar);

/**
 * Reads single pixel value of the curent band.
 * @param x
 * @param y
 * @return pixel value
 */
public int read(int x, int y);

/**
 * Return the name of the band number. For instance in RGBA image this would be:
 * 0 = Red
 * 1 = Green
 * 2 = Blue
 * 3 = Alpha
 * @param band number
 * @return
 */
public String getBandName(int band);

/**
 * Sets the band to be read (before the call to Read, ReadTile, etc...)
 * @param band
 */
public void setBand(int band);
```

```
    /**
     * Method to manage the memory for big images. The data within the interval
     * (y,y+height) is preloaded in memory for fast access for readTile.
     * @param y first line of the part of the image to preload.
     * @param height of the part of the image to preload.
     */
    public void preloadLineTile(int y, int height);

    /**
     * clear all the resources opened to read the image
     */
    public void dispose();

}
```

The basic instructions to read images is:

```
GeoImageReader gir=GeoImageReaderFactory.create("path/to/file");
```

Notice the method called **dispose()**. It is very uncommon to use disposable objects in JAVA but this method is useful to free the System, dereferencing all instances of objects contianed in the file. This way the garbage collector is more efficient.

You may also implement your custom image reader (for new satellite sensors for instance) implementing the **GeoImageReader** interface and adding the class name to the static array GeoImageReaderFactory.FORMATS. You will need to recompile the library. An externalisation of the implementation classes is foreseen. The abstract class **SarImageReader** implements common methods of the **GeoImageReader** interface. you may extends this class for faster implementation:

```
import org.geoimage.SarImageReader

public abstract class MyImage implements SarImageReader {
  public int getNBand() {
    // code here
  }
  public String getFormat() {
    // code here
  }
  public int getNumberOfBytes() {
    // code here
  }
  public int getType(boolean oneBand) {
    // code here
  }
  public String getAccessRights() {
    // code here
  }
  public String[] getFilesList() {
    // code here
  }
  public boolean initialise() {
    // code here
  }
  public void setFile(File imageFile) {
    // code here
  }
  public int[] readTile(int x, int y, int width, int height) {
    // code here
  }
  public int read(int x, int y) {
    // code here
  }
  public String getBandName(int band) {
    // code here
  }
  public void setBand(int band) {
    // code here
  }
  public void preloadLineTile(int y, int height) {
    // code here
  }
}
```

3.2

Chapter 4

GeoImageViewer

4.1 Introduction

The User Interface to deal with the satellite images is managed by the GeoImageViewer project. It also contains a plugin architecture in order to allow third part jars to contribute to the UI.

4.2 Achitecture

The architecure is a pure SWING (the native Java User Interface toolkit) application binded with JOGL. JOGL is a native port of OpenGL using native library for different OS. Netbeans JOGL plugin as installed according to the manual is dealing automatically with the OS you use, if you have setup your **natives.platform** properly. The platform has been designed in order to let developer to contribute to the UI adding menu and panel.

4.3 API

The API is contained in package **org.geoimage.viewer.core.api**. The following classes are part of the API:

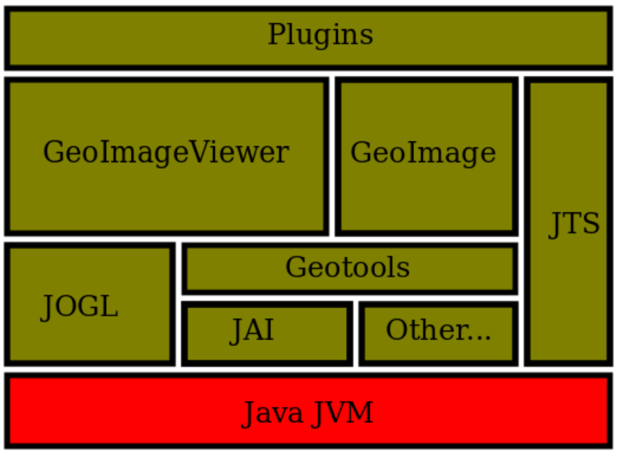

Figure 4.1: The overall architecture of GeoImageViewer and its depedencies

Argument	Class to deal with arguments passed to a plugin
Attributes	Class to deal with attributes associated to a geometry
GeoContext	Class containing the context of the current frame
GeometricLayer	The main class backing the vector model displayed over the raster
IClickable	Interface that will enable a ILayer to pass click events
IComplexVDSVectorLayer	Interface that will enable an **ILayer** to be dealt as a VDS layer
IComplexVectorLayer	Interface that will enable a ILayer to be dealt as a Complex Vector Layer, ie mixing several **GeometricLayer**s
IConsoleAction	Interface implemented by any class to register a plugin
IEditable	Interface that will enable an **ILayer** to be notified of editing events (dragging, delete etc...)
IImageLayer	Interface to enable an **ILayer** to be treated as contributing to the raster and containing several sublayers
IKeyPressed	Interface that will enable an **ILayer** to be notified of keyboard events
ILayer	Generic Interface of a Layer that can be displayed as such in the UI
ILayerListener	Interface to listen to an **ILayer** activity
IListenerUser	Interface to be notified by a **ILayer** activity
IMouseDrag	Interface that will enable an **ILayer** to be notified of dragging events
IMouseMove	Interface that will enable an **ILayer** to be notified of mouse movements events
ISave	Interface that will enable an **ILayer** to be notified of save events
ISelect	Interface that will enable an **ILayer** to be notified of select events
IThreshable	Interface that will enable an **ILayer** to be notified of Thresholding events (passing a the threshold value)
ITime	Interface that will enable an **ILayer** to be notified of ITime events (passing the current timespan)
IVectorLayer	Interface that will enable an **ILayer** to be trated as contributing to the Vector display

4.4 plugins