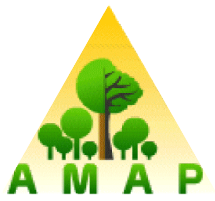
AMAP

AMAPVox
LIDAR data voxelization software

Developer's guide

Creation date : January 20, 2016
Last edited : 10/03/2016
Author : Julien Heurtebize



Summary

I - Development environment

I.1 - Maven

I.1.1 – General

Here i'm gonna talk about Maven version 2, but next it will be implicit.
Maven purpose is the simplification of the project management.
This is mainly achieved by :

- -the simplification of the build process
- -guidelines for best practices development
- -transparent migration to new features

I.1.1.1 - Working tree

Maven help you to organize your working tree, the test, test resources, application resources and source code in separated partitions.

```
►src
        ►main
                ►java----------java files, application source code
                ►resources---------resources required by the application
        ►test
                ►java----------java files, test source code
                ►resources---------resources required by the tests, not deployed
►target--------------------------------Output build directory
pom.xml--------------------------------Project Object Model
```

I.1.1.2 – POM

The Project Object Model describes the project as an xml file, it is the core of a maven project.
It mainly contains the versioning, dependencies, plugins and various configuration detail used by Maven to build the project(s).

A basic POM structure will looks like that :

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>com.companyname.project-group</groupId>
        <artifactId>project</artifactId>
        <version>1.0</version>
</project
```

The root element of the POM is <project>

All POM files require the project element and three mandatory fields : groupId, artifactId, version.

In a repository, the project is identified by those nodes, in this scenario, the project is identified by : com.companyname.project-group:project:1.0

Node	Description
groupId	This is an Id of project's group. This is generally unique amongst an organization or a project. For example, a banking group com.company.bank has all bank related projects.
artifactId	This is an Id of the project.This is generally name of the project. For example, consumer-banking. Along with the groupId, the artifactId defines the artifact's location within the repository.
version	This is the version of the project.Along with the groupId, It is used within an artifact's repository to separate versions from each other. For example: com.company.bank:consumer-banking:1.0 com.company.bank:consumer-banking:1.1.

This POM automatically inherit from the Super POM, also called the effective pom. The effective POM contains default configurations, like default plugin configurations. This is why we don't have to specify plugins from the « org.apache.maven.plugins »and their configurations.

Source : http://www.tutorialspoint.com/maven/maven_pom.htm

I.1.1.3 – Dependencies

For me this is the most interesting part of maven.
You can add a dependency to your project just by adding few lines to the pom.xml file.

The most used dependencies can be found on the maven remote repository.
You can search a dependency on the website http://mvnrepository.com/.
Once you have found the dependency you were looking for, you can add the specified tag to your pom.xml file into the <dependencies> tag.

Example :
You want the apache commons math library, the url of the maven description page is http://mvnrepository.com/artifact/org.apache.commons/commons-math3/3.6

You just have to copy the tag :

```
<dependency>
	<groupId>org.apache.commons</groupId>
	<artifactId>commons-math3</artifactId>
	<version>3.6</version>
</dependency>
```

and paste it to your pom.xml file into the dependencies tag:

```
<dependencies>
	<dependency>
		<groupId>org.apache.commons</groupId>
		<artifactId>commons-math3</artifactId>
		<version>3.6</version>
	</dependency>
</dependencies>
```

Other repository

If you want to get a dependency from somewhere else, you have to add the repository server in pom file.

```
<repositories>
   <repository>
      <id>id_server</id>
      <url>https://repositoryserver/maven-repo</url>
   </repository>
</repositories>
```

I.1.1.4 – Variables

Source : https://books.sonatype.com/mvnref-book/reference/resource-filtering-sect-properties.html

Maven allow you to use your own or pre-defined variables in the POM.
To access to a maven property variable, you would reference it inside the POM like so: ${project.version}

I.1.1.4.1 – Maven project properties

Some most commons pre-defined variables:
${project.groupId}
${project.artifactId}
${project.build.sourceDirectory}

The complete list can be found to the following url :
http://maven.apache.org/ref/3.0.3/maven-model/maven.html

I.1.1.4.2 – Maven settings properties

You can also reference any properties in the Maven Local Settings file which is usually stored in ~/.m2/settings.xml. This file contains user-specific configuration such as the location of the local repository and any servers, profiles, and mirrors configured by a specific user.

A full reference for the Local Settings file and corresponding properties is available here :
http://maven.apache.org/ref/3.0.3/maven-settings/settings.html

I.1.1.4.3 – User defined properties

You can define your own variables inside a POM file like this :

```
<properties>
        <property1>value1</property1>
</properties>
```

and use it with ${property1}

I.1.1.4.4 – Java System properties

Maven exposes all properties from java.lang.System. Anything you can retrieve from System.getProperty() you can get it as a Maven property.

Example : ${os.version}
Full list here.

I.1.1.4.5 – Environment variable properties

The system environment variables can be retrieved with the env.* prefix.
Example : ${env.M2_HOME}, contains the Maven 2 installation directory.

I.1.1.6 – Plugins

A mojo is a **M**aven plain **O**ld **J**ava **O**bject. Each mojo is an executable *goal* (a task which Maven can execute) in Maven, and a **plugin** is a distribution of one or more related mojos.
The plugin should be described into the POM file.

A <plugin> tag is specified inside the <plugins> tag which is inside the <build> tag.
The template is like this :

```
<build>
	<plugins>
		<plugin>
			<groupId> </groupId>
			<artifactId> </artifactId>
			<version> </version>
		</plugin>
	</plugins>
</build>
```

I.1.1.6 - Multiple modules

TODO : parler du rôle et du fonctionnement du reactor

I.1.2 – Usage

I.1.2.1 – Plugins

Group id	Artifact id	Description
org.codehaus.mojo	buildnumber-maven-plugin	Get a unique build number for each time you build your project. This plugin is configured to get the 8 first number of the revision if from the git repository. This plugin create a maven variable called ${buildNumber}, which can be used into the pom.xml
org.apache.maven.plugins	maven-javadoc-plugin	Generate javadoc for the project.

		It is configured to output the javadoc into a jar file during the deploy phase.
org.apache.maven.plugins	maven-deploy-plugin	Add artifacts to a remote repository. It is configured to add the generated artifacts to the local repository.
com.github.github	site-maven-plugin	Commit generated files and update a specific branch reference in a github repository. This plugin is used in the project to share generated artifacts. So any AMAPVox module can be used from any other project across the world.
org.apache.maven.plugins	maven-dependency-plugin	Provides the capability to manipulate artifacts. It can copy and/or unpack artifacts from local or remote repositories to a specified location. This plugin is used in order to copy the dependencies to the target directory. Like this the dependencies are not encapsulated into the final jar but the final jar refered to them.
org.apache.maven.plugins	maven-jar-plugin	Provides the capability to build jars. It is also configured to add entries to the manifest, like the build number and it specify the classpath because dependencies are in a separated folder.
org.codehaus.mojo	exec-maven-plugin	Provides 2 goals to help execute system and Java programs. It allows to configure the java executable location and to configure command line arguments (jar to execute, xmx parameter, ...). This is how AMAPVoxGUI is launched from the IDE.
org.apache.maven.plugins	maven-compiler-plugin	Compile the sources of your project.
org.apache.maven.plugins	maven-resources-plugin	The Resources Plugin handles the copying of project resources to the specified output directory. I'm currently using it because i need to copy launcher script files next to the target jar.
com.google.code.maven-replacer-plugin	replacer	It performs some advanced text replacement functions. I'm using it to replace the token $BUILD_NUMBER$ of the launcher script files by the actual generated build number.

Upload AMAPVox dependencies to a remote repository via the plugins

Currently, AMAPVox project is configured with the ability of uploading the dependencies into a remote git repository.

The main steps of

I.1.2.2 – POM

I.2 – Netbeans /Eclipse

I.3 – Git

I.3.1 – Install

Linux :

Open a terminal in root mode and enter the following command :

```
apt-get install git
```

Windows :

Download the setup, link : https://git-scm.com/download/win
Install the downloaded setup file.

Test the git command. Open a console and type in « git ».
If the command is not recognized in Windows make sure the environment variable was configured by the installer, one of the available options disable it. Or you can also make a right click inside the browser file and choose the entry « *Open git bash here* ».

I.3.2 – Configure

You can configure your name and e-mail.
Open a console and enter the following commands :

```
git config –global user.email "your_email@example.com"
```

```
git config --global user.name "Your_user_name"
```

I.3.3 – Usage

I.3.3.1 – Initialize local repository

Step 1 : Create folder for the local repository.
On Linux OS: `mkdir /home/myfolder && cd $_`
Step 2 : Initialize new git repository
Open console and, from the folder you created before, enter command:

```
git init .
```

I.3.3.2 – Import project

Get the content of the remote repository with the following command :

```
git pull http://amap-dev.cirad.fr/git/voxelidar.git
```

Enter your amap-dev credentials when required.

Then a copy of the remote repository is performed and the content will be into your local repository.

I.3.3.3 – Update project

To update your local repository with the remote, you can enter the same command as above :

```
git pull http://amap-dev.cirad.fr/git/voxelidar.git
```

To update the repository to a specific branch, you can use the command above followed by the branch name :

```
git pull http://amap-dev.cirad.fr/git/voxelidar.git mybranch
```

I.3.3.4 – Configure remote repository

If the configuration is broken, the following commands will configure the « ./git/config » file.

Add the remote :

```
git remote add origin http://amap-dev.cirad.fr/git/voxelidar.git
```

(« origin » is a handcut name)

Get the remote master branch :

```
git pull origin master
```

Link the local master branch to the remote master branch :

```
git branch master --set-upstream-to  origin/master
```

I.3.3.5 – Use branches

I.3.3.5.1 – Create a branch

```
git branch branch_name
```

I.3.3.5.2 – Switch between branches

```
git checkout branch_name
```

I.3.3.5.3 – Merge a branch into another

Merge the specified branch into the current branch :

```
git merge branch_name
```

I.3.3.5.4 – Create or update a remote branch

Switch to the desired branch :

```
git checkout local_branch_name
```

Push the branch to the remote :

```
git push remote_branch_name
```

I.3.3.5.5 – Remove a branch

To delete a local branch :

```
git branch -d local_branch_name
```

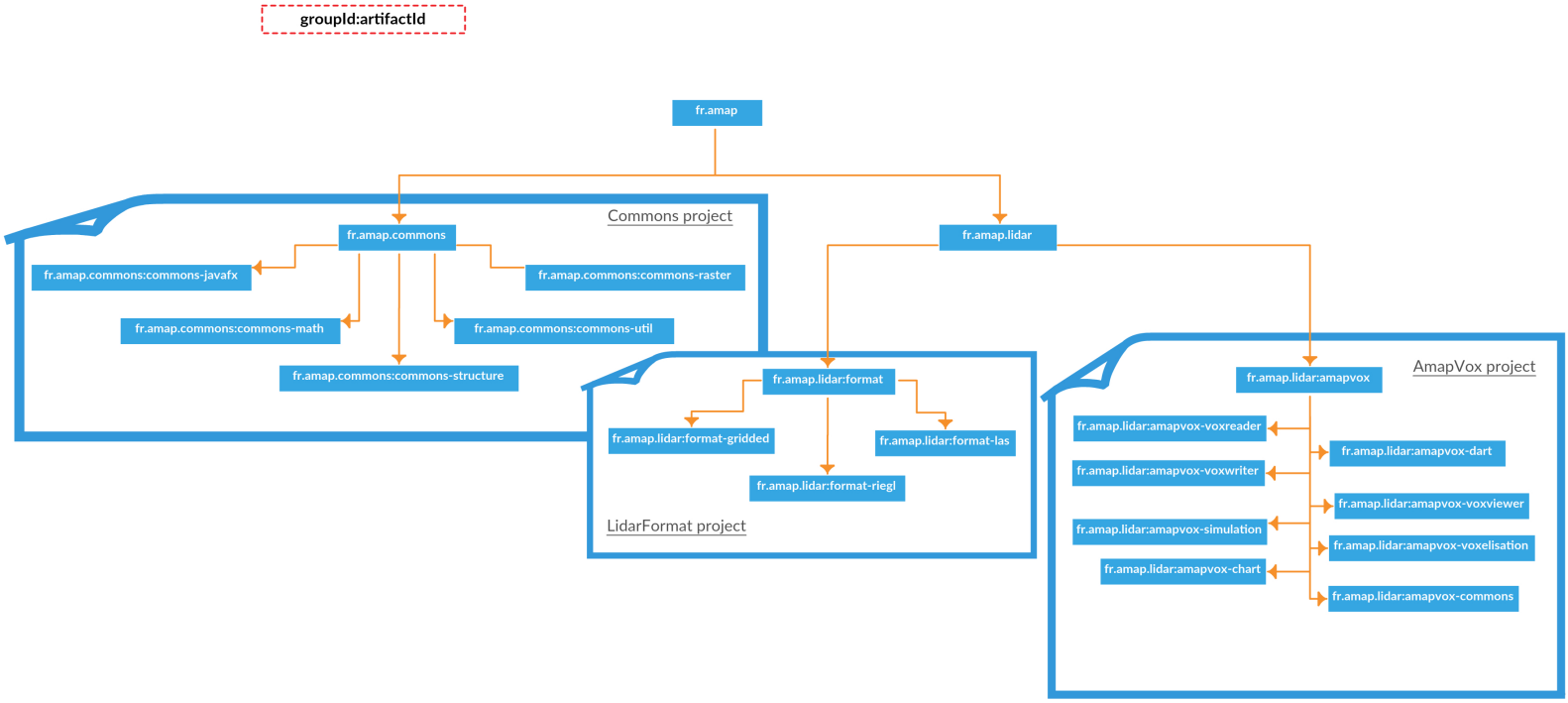
To remove a remote branch (be careful) :

```
git push origin --delete origin/branch_name
```

II - Project architecture

The AMAPVox project has been initially developped in a single project, but in the aim of reusing modules for other projects, AMAPVox have been splitted in multiple projects, each one producing one or more dependencies.
We need to distinguish projects, modules and packages.
Each maven project contains a POM file (pom.xml), this file contains the groupId and artifactId tags, it defines the project among all projects.
The *groupId* generally contains the company name and the project category.
The *artifactId* generally contains the name of the project or the name of a project's module.

The diagram below shows the three current projects and the related artifact and group ids of the different modules.



II.1 – Modules

II.1.1 – Handle lidar files

II.1.1.1 – LAS files(*.las/*.laz)

Group id	**Artifact id**
fr.amap.lidar	format-las

Usage :

Read a *.laz file :

```
LazExtraction lazReader = new LazExtraction();

try {

    lazReader.openLazFile(new File("/home/Documents/sample.laz"));

    //read header
    LasHeader header = lazReader.getHeader();

    System.out.println("Number of point records :
"+header.getNumberOfPointrecords());

    //get the points from the file
    Iterator<LasPoint> iterator = lazReader.iterator();

    while(iterator.hasNext()){

        LasPoint point = iterator.next();

        double ptX = header.getxOffset() + point.x * header.getxScaleFactor();
        double ptY = header.getyOffset() + point.y * header.getyScaleFactor();
        double ptZ = header.getzOffset() + point.z * header.getzScaleFactor();

        System.out.println(ptX+"\t"+ptY+"\t"+ptZ);
    }

} catch (Exception ex) {
    Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
}finally{
    lazReader.close();
}
```

Read a *.las file :

```
LasReader lasReader = new LasReader();

try {
    lasReader.open(new File("/home/Documents/sample.las"));

    LasHeader header = lasReader.getHeader();

    Iterator<PointDataRecordFormat> iterator = lasReader.iterator();

    while(iterator.hasNext()){

        PointDataRecordFormat point = iterator.next();

        double ptX = header.getxOffset() + point.getX() * header.getxScaleFactor();
        double ptY = header.getyOffset() + point.getY() * header.getyScaleFactor();
        double ptZ = header.getzOffset() + point.getZ() * header.getzScaleFactor();

        System.out.println(ptX+"\t"+ptY+"\t"+ptZ);
    }

} catch (Exception ex) {
    Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
}
```

II.1.1.2 – Riegl files(*.rsp, *.rxp)

Group id	Artifact id
fr.amap.lidar	format-riegl

Usage :

Read a Riscan Project File (*.rsp) :

```
Rsp rsp = new Rsp();

try {
    rsp.read(new
File("/media/forestview01/BDLidar/TLS/Paracou2014/FTH2014.RiSCAN/project.rsp"));

    System.out.println("Riscan project POP matrix :
\n"+rsp.getPopMatrix().toString()+"\n");

    ArrayList<Scans> rxpList = rsp.getRxpList();

    System.out.println("Scans list :\n");

    for(Scans scans : rxpList){

        System.out.println("Scan name :"+scans.getName());

        System.out.println("Scan SOP matrix :\n"+scans.getSopMatrix());

        System.out.println("Rxp file path :"+scans.getScanFull().getAbsolutePath()
+"\n");
    }

} catch (JDOMException ex) {
    Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
} catch (IOException ex) {
    Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
}
```

Read a *.rxp file :

```
RxpExtraction rxpReader = new RxpExtraction();

try {
    rxpReader.openRxpFile(new File("/home/Documents/sample.rxp"),
RxpExtraction.REFLECTANCE, RxpExtraction.AMPLITUDE, RxpExtraction.DEVIATION,
RxpExtraction.TIME); //select echoes attrbutes to import

    Iterator<Shot> iterator = rxpReader.iterator();

    while(iterator.hasNext()){

        Shot shot = iterator.next();

        int ptIndex = 0;

        for(double range : shot.ranges){

            double ptX = shot.origin.x + shot.direction.x * range;
            double ptY = shot.origin.y + shot.direction.y * range;
            double ptZ = shot.origin.z + shot.direction.z * range;

            System.out.println(ptX+"\t"+ptY+"\t"+ptZ+"\t"+shot.reflectances[ptIndex]
+"\t"+
                    shot.deviations[ptIndex]+"\t"+shot.amplitudes[ptIndex]
+"\t"+shot.times[ptIndex]);

            ptIndex++;
        }
    }

} catch (Exception ex) {
    Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
}finally{
    rxpReader.close();
}
```

II.1.1.3 – Leica gridded point format files (*.ptx, *.ptg)

Group id	Artifact id
fr.amap.lidar	format-gridded

Usage :

Read a *.ptg file :

```
PTGReader ptgReader = new PTGReader();

try {
    File ptgFile = new File("/home/Documents/scan-1.ptg");

    ptgReader.openPTGFile(ptgFile);

    if(ptgReader.isAsciiFile()){ //the opened file contains the scan list

        List<File> scanList = ptgReader.getScanList();

        System.out.println("Scan list :\n");

        for(File file : scanList){
            System.out.println(file.getAbsolutePath());
        }

    }else{ //the opened file is a binary scan

        PTGScan scan = new PTGScan();

        try {
            scan.openScanFile(ptgFile);

            PTGHeader header = scan.getHeader();

            //get the transformation matrix of the scan
            Mat4D transfMatrix = header.getTransfMatrix();

            Iterator<LPoint> iterator = scan.iterator();

            while(iterator.hasNext()){

                LPoint point = iterator.next();

                double ptX, ptY, ptZ;

                if(header.isPointInFloatFormat()){
                    ptX = ((LFloatPoint)point).x;
                    ptY = ((LFloatPoint)point).y;
                    ptZ = ((LFloatPoint)point).z;
                }else{
                    ptX = ((LDoublePoint)point).x;
                    ptY = ((LDoublePoint)point).y;
                    ptZ = ((LDoublePoint)point).z;
                }

                Vec4D ptTransformed = Mat4D.multiply(transfMatrix, new Vec4D(ptX,
ptY, ptZ, 1));


System.out.println(ptTransformed.x+"\t"+ptTransformed.y+"\t"+ptTransformed.z);
            }


        } catch (Exception ex) {
            Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

} catch (IOException ex) {
    Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
}
```

Read a *.ptx file :

```
PTXReader ptxReader = new PTXReader();

try {
    ptxReader.openPTXFile(new File("/home/Documents/sample.ptx"));

    List<PTXScan> singlesScans = ptxReader.getSinglesScans();

    if(ptxReader.getNbScans() > 0){

        PTXScan scan = singlesScans.get(0);
        PTXHeader header = scan.getHeader();

        Mat4D transfMatrix = header.getTransfMatrix();

        Iterator<LPoint> iterator = scan.iterator();

        while(iterator.hasNext()){

            LPoint point = iterator.next();

            double ptX, ptY, ptZ;

            if(header.isPointInFloatFormat()){
                ptX = ((LFloatPoint)point).x;
                ptY = ((LFloatPoint)point).y;
                ptZ = ((LFloatPoint)point).z;
            }else{
                ptX = ((LDoublePoint)point).x;
                ptY = ((LDoublePoint)point).y;
                ptZ = ((LDoublePoint)point).z;
            }

            Vec4D ptTransformed = Mat4D.multiply(transfMatrix, new Vec4D(ptX, ptY,
ptZ, 1));


System.out.println(ptTransformed.x+"\t"+ptTransformed.y+"\t"+ptTransformed.z);
        }
    }

} catch (IOException ex) {
    Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
}
```

II.1.2 – Data structures

Group id	Artifact id
fr.amap.commons	commons-structure

II.1.2.1 – Octree

Usage :

```
//Initialize a new octree structure configured to have a maximum of 50 points by
leaf
Octree octree = new Octree(50);

//generate random values
int nbPoints = 1000000;

Point3D[] points = new Point3D[nbPoints];

Random r = new Random();

for(int i=0;i<nbPoints;i++){

    points[i] = new Point3D(r.nextDouble(), r.nextDouble(), r.nextDouble());
}

//set points
octree.setPoints(points);

try {
    octree.build(); //build octree

    //usage
    float errorMargin = 0.0001f;

    //search the nearest point in a spherical zone (diameter 0.0001)
    Point3D nearestPoint = octree.searchNearestPoint(new Point3D(0.5, 0.1, 0.3),
Octree.INCREMENTAL_SEARCH, errorMargin);

    if(nearestPoint == null){
        System.out.println("No point found");
    }else{
        System.out.println("A point with coordinates
("+nearestPoint.x+";"+nearestPoint.y+";"+nearestPoint.z+") was found.");
    }

} catch (Exception ex) {
    Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
}
```

II.1.3 – Raster

Group id	Artifact id
fr.amap.commons	commons-raster

The main class of this module is the « Raster » class.
It provides the following tools :

-getting the height (z value) at a x,y position
-build to a mesh (triangles, vertices and indices)
-export to obj
-subset a smallest raster with a given 2d bounding box

II.1.3.1 – Ascii grid format (*.asc)

The *commons-raster* module can be used to read ascii grid format file (https://en.wikipedia.org/wiki/Esri_grid).
The class « *AsciiGridHelper* » read an ascii grid file and convert it to a raster as a « *Raster* » object.

This format is commonly used by LAStools to export DTM or DEM.

Usage :

```
try {
    Raster raster = AsciiGridHelper.readFromAscFile(new
File("/home/Documents/sample.asc"));

    //get the height of the x,y location
    float height = raster.getSimpleHeight(50, 60);

    //constructs a mesh (triangles, points, indices) from the raster
    raster.buildMesh();

    //export the raster to obj format
    raster.exportObj(new File("/home/Documents/mesh.obj"));

} catch (Exception ex) {
    Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
}
```

II.1.3.1 – Multi-band rasters (*.bil, *.bip, *.bsq)

The specification of the multi-band raster formats bil, bip and bsq can be found here : http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=BIL,_BIP,_and_BSQ_raster_files

The only difference between those formats is the organization of the data (bands) into the file.

The reading of those formats is currently unavailable, only the writing can be processed.

The methods available allows you to :

-instantiate a multi-band raster in the following formats : bil, bip, bsq
-set a pixel to the raster to the x, y indices and band index
-write the raster with the apporpriate extension
-writer the header (*.hdr)

Usage :

```
int nbColumns = 100;
int nbRows = 100;
int nbBands = 10;

//defines a multi-band raster with a 1 byte value storage
BHeader header = new BHeader(nbColumns, nbRows, nbBands,
BCommon.NumberOfBits.N_BITS_8);

header.setXdim(1); //pixel x dimension in map unit
header.setYdim(1);//pixel y dimension in map unit

BSQ raster = new BSQ(new File("/home/calcul/Documents/Julien/sample.bsq"), header);

try{

    for(int i=0;i<nbColumns;i++){
        for(int j=0;j<nbRows;j++){
            for(int k=0;k<nbBands;k++){

                raster.setPixel(i, j, k, (byte)(i/(float)nbColumns));
            }
        }
    }

    try {
        raster.writeImage();
        raster.writeHeader(); //header is in a separate file

    } catch (IOException ex) {
        Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
    }

}catch(Exception e){
    Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, e);
}
```

II.1.4 – Mathematiques

Group id	Artifact id
fr.amap.commons	commons-math

Content : This module can be used to handle geometric objects like vectors, matrices, points.
The packages available are :

-matrix (3d, 4d)
-point (2d, 3d)
-vector (2d, 3d, 4d)

All those classes are declined in multiple storage format : integer, float and double.
For performance issue the generic types are not used here.

II.1.5 – Voxel files

II.1.5.1 – Read a voxel file

Group id	Artifact id
fr.amap.lidar.amapvox	amapvox-voxreader

Usage :

```
try {
    VoxelFileReader reader = new VoxelFileReader(new
File("/home/Documents/sample.vox"),
            true); //if true, the voxels are kept in memory

    VoxelSpaceInfos infos = reader.getVoxelSpaceInfos();

    System.out.println("Splitting : "+infos.getSplit().toString());
    System.out.println("Min corner : "+infos.getMinCorner().toString());
    System.out.println("Max corner : "+infos.getMaxCorner().toString());

    Iterator<Voxel> iterator = reader.iterator();

    while(iterator.hasNext()){

        Voxel voxel = iterator.next();

        System.out.println(voxel.$i+"\t"+voxel.$j+"\t"+voxel.
$k+"\t"+voxel.transmittance);
    }

} catch (Exception ex) {
    Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
}
```

II.1.6 – Gui reusable components (JavaFX)

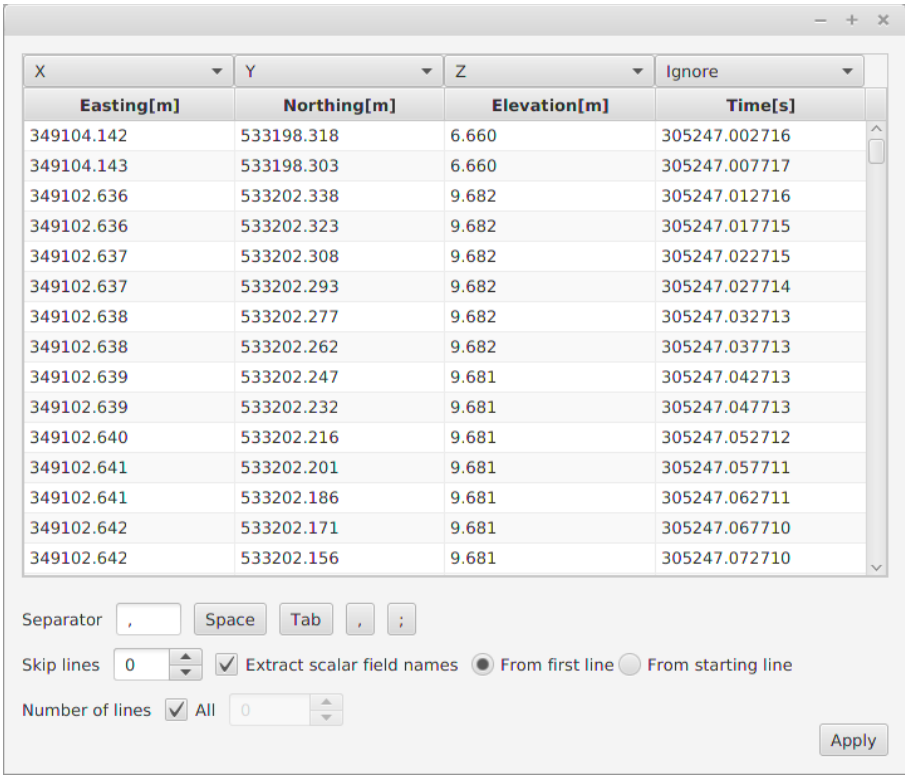
Group id	Artifact id
fr.amap.commons	commons-javafx

II.1.7 – CSV file parser (Interactive frame)

Purpose : Let the user choose how a csv file should be read.

The user has to configure several parameters :

-the column separator (comma, space, etc.)
-the existence or non-existence of a header
-the number of lines to read
-the number of lines to skip
-the column assignment (determines what the column values are).

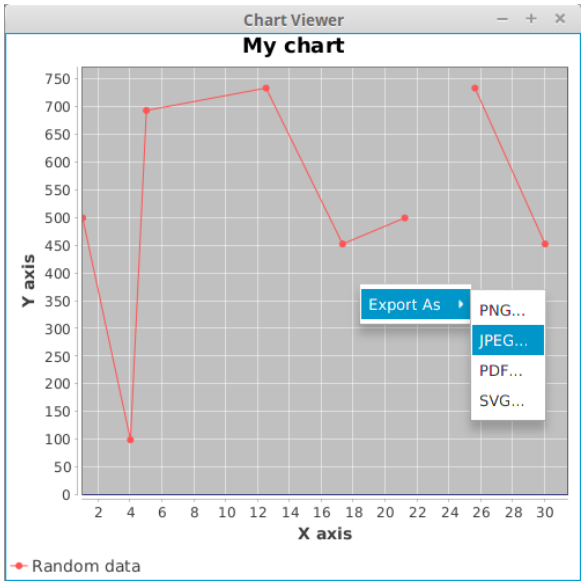


Usage :

```
final TextFileParserFrameController controller =
TextFileParserFrameController.getInstance();

controller.setColumnAssignment(true);

//set the combobox choices
controller.setColumnAssignmentValues("Ignore", "X", "Y", "Z", "Red", "Green",
"Blue", "Scalar field");

//default column assignment
controller.setColumnAssignmentDefaultSelectedIndex(0, 1);
controller.setColumnAssignmentDefaultSelectedIndex(1, 2);
controller.setColumnAssignmentDefaultSelectedIndex(2, 3);
controller.setColumnAssignmentDefaultSelectedIndex(3, 4);

//set the viewed text file
controller.setTextFile(new File("/home/Documents/sample.txt"));

Stage frame = controller.getStage();
frame.show();

frame.setOnHidden(new EventHandler<WindowEvent>() {
    @Override
    public void handle(WindowEvent event) {

        System.out.println("Separator : "+controller.getSeparator());
        System.out.println("Maximum number of lines to read:
"+controller.getNumberOfLines());

        List<String> assignedColumnsItems = controller.getAssignedColumnsItems();

        for (int i = 0; i < assignedColumnsItems.size(); i++) {

            String item = assignedColumnsItems.get(i);

            if (item != null) {
                System.out.println("Item "+item+" was assigned to the column number
"+i);
            }
        }
    }
});
```

II.1.8 – Chart viewer (JFreeChart)

Purpose : Provides a frame to display JfreeChart charts and tools to export chart to png, jpeg, svg and pdf.



Usage :

The following code displays the frame above.

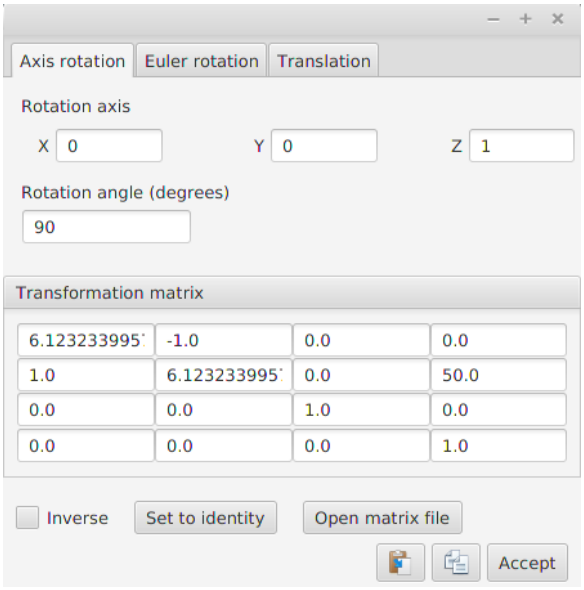
```
/*instantiate a new chart viewer with a size of 500x500 and a maximum
of one chart in a row*/
ChartViewer viewer = new ChartViewer("Chart Viewer", 500, 500, 1);

XYSeriesCollection dataset = new XYSeriesCollection();

XYSeries series = new XYSeries("Random data");
series.add(1.0, 500.2);
series.add(5.0, 694.1);
series.add(4.0, 100.0);
series.add(12.5, 734.4);
series.add(17.3, 453.2);
series.add(21.2, 500.2);
series.add(21.9, null);
series.add(25.6, 734.4);
series.add(30.0, 453.2);

dataset.addSeries(series);

JFreeChart chart = ChartViewer.createBasicChart("My chart", dataset, "X axis", "Y
axis");

//add the chart to the viewer, viewer can contains multiple charts
viewer.insertChart(chart);

viewer.show();
```

II.1.9 – Transformation matrix frame

Purpose : Edit interactively a transformation 4x4 matrix.
The rotation and translation of the transformation matrix are being displayed at the same time the matrix is edited, the inverse, editing of rotation or translation will also update the matrix.

Tools available :

-Open a matrix from a file
-Copy/paste from the clipboard
-Inverse the current matrix
-Reset matrix to identity



Usage :

```
final TransformationFrameController controller =
TransformationFrameController.getInstance();

Stage stage = controller.getStage();

controller.reset();

stage.setOnHidden(new EventHandler<WindowEvent>() {

    @Override
    public void handle(WindowEvent event) {

        if(controller.isConfirmed()){ //user clicked on confirm button

            Matrix4d matrix = controller.getMatrix();

            System.out.println(matrix.toString());
        }
    }
});

stage.show();
```

II.1.7 – 3D viewer

II.1.7.1 – Create a window with an OpenGL context

```
final Viewer3D viewer3D = new Viewer3D(0, 0, 500, 400, "OpenGL window");
viewer3D.show();
```

The code below create a window at location 0, 0 with a size of 500x400 and creates an OpenGL context.

II.1.7.2 - Add a 3d scene object

The following code will create a new SceneObject with a mesh representing a cube. GLMeshFactory provides some methods to generate current primitives.

```
final Viewer3D viewer3D = new Viewer3D(0, 0, 500, 400, "OpenGL window");

SceneObject cube = new SimpleSceneObject(GLMeshFactory.createCube(2));
viewer3D.getScene().addSceneObject(cube);


viewer3D.show();
```

II.1.7.3 – Bind a shader to an object

```
SceneObject cube = new SimpleSceneObject(GLMeshFactory.createCube(2));

//use a global shader, shared by any objects using this shader
cube.setShader(Scene.colorShader);

//or use an unique shader
cube.setShader(new ColorShader());

viewer3D.getScene().addSceneObject(cube);
```

Available shaders :

Shader	Variables	Usage
SimpleShader	uniform mat4 viewMatrix	Camera transformation
	uniform mat4 projMatrix	Camera projection
	uniform mat4 transformation	Object transformation
	uniform vec3 color	Unique color
	attribute vec4 position	Vertex positions
ColorShader	uniform mat4 viewMatrix	Camera transformation
	uniform mat4 projMatrix	Camera projection
	uniform mat4 transformation	Object transformation
	attribute vec3 color	Vertex colors
	attribute vec4 position	Vertex positions
PhongShader	uniform mat4 viewMatrix	Camera transformation
	uniform mat4 projMatrix	Camera projection
	uniform mat4 transformation	Object transformation
	attribute vec3 color	Vertex colors
	attribute vec3 position	Vertex positions
	attribute vec3 normal	Vertex normales
InstanceShader/ InstanceLightedShader	uniform mat4 viewMatrix	Camera transformation
	uniform mat4 projMatrix	Camera projection
	attribute vec3 position	Vertex positions
	attribute vec3 instance_position	Position of the object's instance

	attribute vec4 instance_color	Color of the object's instance
TextureShader	uniform mat4 viewMatrixOrtho	Camera transformation
	uniform mat4 projMatrixOrtho	Camera projection
	attribute vec4 position	Vertex positions
	attribute vec2 textureCoordinates	Vertex linked texture coordinates

The SimpleShader can be used when you want to draw an object with a single color and be able to move the object.

The ColorShader can be used to draw an object with a different color for each vertex. This shader allows the object transformation too.

The PhongShader simulates a light with a phong model, built with specular, diffuse and ambiant colors. Those color values are fixed in the shader.

The InstanceShader/InstanceLighted shaders both allows to render multiple instances of an object, those instances can only be parametrized with their locations.
The InstanceLightedShader is the same as InstanceShader but simulate a prebuilt light. This one is specific and should not be used.

TextureShader is a shader handling texture coordinates. It can be used to draw head up display (HUD) on the screen.

Global uniforms :

The uniforms « viewMatrix » and « projMatrix » are updated from the Scene when the camera is updated.
This means you can create your own shader and those two variables will be automatically udpated.
In another hand, if you want to update your own view matrix and projection matrix, don't name your variables like the two above.

II.1.7.4 - The scene object types

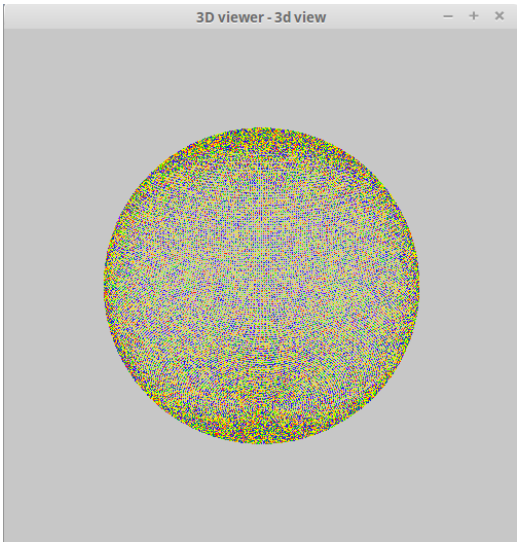
PointCloudSceneObject :
Represents a point cloud.

-Provides methods to add points with attributes easily.
-Can be mouse pickable (explained after).

Usage example, spherical point cloud creation with mouse picking:

```
final Viewer3D viewer3D = new Viewer3D(0, 0, 500, 500, "3d view");
viewer3D.setDynamicDraw(true);
viewer3D.getAnimator().setUpdateFPSFrames(1, null);

PointCloudSceneObject pointCloud = new PointCloudSceneObject();
pointCloud.setMousePickable(true);

for (int i = 1; i < 180; i++) {

    for (int j = -179; j < 179; j++) {

        double theta = Math.toRadians(i);
        double phi = Math.toRadians(j);

        float x = (float)(10*Math.sin(theta)*Math.cos(phi));
        float y = (float)(10*Math.sin(theta)*Math.sin(phi));
        float z = (float)(10*Math.cos(theta));

        pointCloud.addPoint(x, y ,z);
        pointCloud.addValue("attribut", (float) (Math.random()*255));

    }
}

pointCloud.initMesh();
pointCloud.setShader(Scene.colorShader);

viewer3D.getScene().addSceneObject(pointCloud);

SceneObjectListener pointcloudListener = new SceneObjectListener() {
    @Override
    public void clicked(SceneObject sceneObject, MousePicker mousePicker) {

        Point3F point = (Point3F) sceneObject.doPicking(
                viewer3D.getScene().getMousePicker());

        if(point != null){
            viewer3D.getScene().getCamera().setLocation(
                    new Vec3F(point.x, point.y, point.z));
        }
    }
};

pointCloud.addSceneObjectListener(pointcloudListener);
viewer3D.show();
```

Result :



VoxelSpaceSceneObject :
Scene object dedicated to display voxels.

-All voxels are drawn with a single draw call (instance rendering)
-Can be mouse pickable
-Handle filtering

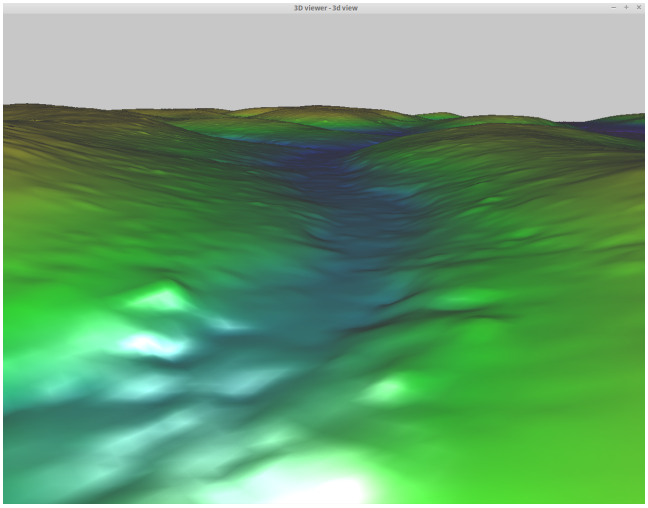
RasterSceneObject :
Represents a DTM.

-Is mouse pickable

Usage example, load a dtm from a file and handle selection events

```
final Viewer3D viewer3D = new Viewer3D(0, 0, 500, 500, "3d view");
viewer3D.setDynamicDraw(false);

RasterSceneObject dtm = new RasterSceneObject(
        AsciiGridHelper.readFromAscFile(
                new File("/media/example.asc")));

dtm.setMousePickable(true);

viewer3D.getScene().addSceneObject(dtm);

//add a camera listener
viewer3D.getScene().getCamera().addCameraListener(new CameraAdapter() {
    @Override
    public void locationChanged(Vec3F location) {

viewer3D.getScene().setLightPosition(viewer3D.getScene().getCamera().getLocation());
    }
});

//add a scene object listener
SceneObjectListener objectListener = new SceneObjectListener() {
    @Override
    public void clicked(SceneObject sceneObject, MousePicker mousePicker) {

        Integer index = ((RasterSceneObject)sceneObject).doPicking(mousePicker);

        if(index != null){
            Point3F position = ((RasterSceneObject)sceneObject).getVertex(index);
            System.out.println(position.x + " "+position.y+ " "+position.z);
        }
    }
};

dtm.addSceneObjectListener(objectListener);

viewer3D.show();
```

Result :



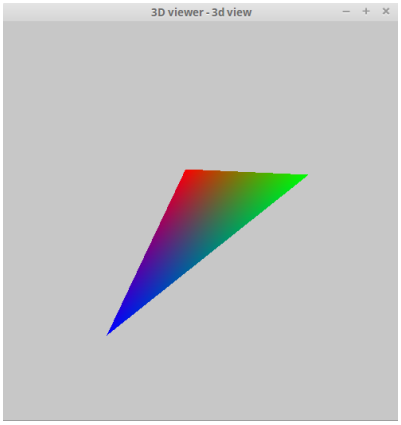
SimpleSceneObject :
Represents a simple scene object.

-Draw the provided mesh

Usage example, create a colored triangle

```
final Viewer3D viewer3D = new Viewer3D(0, 0, 500, 500, "3d view");
viewer3D.setDynamicDraw(false);

GLMesh mesh = new SimpleGLMesh();

float vertexData[] = new float[]
{-2.0f, 2.0f, 2.0f,
2.0f, 2.0f, 2.0f,
-2.0f, -2.0f, -2.0f};

float colorData[] = new float[]
{1.0f, 0.0f, 0.0f,
0.0f, 1.0f, 0.0f,
0.0f, 0.0f, 1.0f};

int indexData[] = new int[]
{0, 1, 2};

mesh.setVertexBuffer(Buffers.newDirectFloatBuffer(vertexData));
mesh.indexBuffer = Buffers.newDirectIntBuffer(indexData);
mesh.colorBuffer = Buffers.newDirectFloatBuffer(colorData);
mesh.vertexCount = indexData.length;

SimpleSceneObject sceneObject = new SimpleSceneObject(mesh);
sceneObject.setShader(new ColorShader());

viewer3D.getScene().addSceneObject(sceneObject);

viewer3D.show();
```

Result :



II.1.7.5 - Set up camera

The default implemented camera is a trackball camera.
You can configure the target of the camera (where it looks at) and its position.

```
final Viewer3D viewer3D = new Viewer3D(0, 0, 500, 500, "3d view");
viewer3D.setDynamicDraw(false);

SimpleSceneObject sceneObject = new SimpleSceneObject(GLMeshFactory.createCube(2));

//set the scene object as the pivot
viewer3D.getScene().getCamera().setPivot(sceneObject);

//set the camera location
viewer3D.getScene().getCamera().setLocation(new Vec3F(10, 50, 40));

viewer3D.getScene().addSceneObject(sceneObject);

viewer3D.show();
```

By default the pivot of the camera is the first scene object added to the scene.
Its default location is configured as the first scene object gravity center, plus x, y, z offsets.

II.1.7.6 - Set up light

This part is useful if you use the phong shader, because this shader use the light position information to compute the mesh color.

```
final Viewer3D viewer3D = new Viewer3D(0, 0, 500, 500, "3d view");
viewer3D.setDynamicDraw(false);

SimpleSceneObject sceneObject = new SimpleSceneObject(GLMeshFactory.createCube(2));
sceneObject.setShader(new PhongShader());

sceneObject.setMousePickable(true);

viewer3D.getScene().addSceneObject(sceneObject);

//add a camera listener
viewer3D.getScene().getCamera().addCameraListener(new CameraAdapter() {
    @Override
    public void locationChanged(Vec3F location) {
        viewer3D.getScene().setLightPosition(
                viewer3D.getScene().getCamera().getLocation());
    }
});

viewer3D.show();
```

This code display a cube with a shader that handles light, and move the light according to the camera position.

II.1.7.7 - Add keyboard, mouse, window listeners

To get access to the windows events (window resizing, keyboard, mouse), you should get the GLRenderFrame. It provides access to all of those.

Add a mouse listener :

```
viewer3D.getRenderFrame().addMouseListener(new MouseListener() {
    @Override
    public void mouseClicked(MouseEvent e) {}

    @Override
    public void mouseEntered(MouseEvent e) {}

    @Override
    public void mouseExited(MouseEvent e) {}

    @Override
    public void mousePressed(MouseEvent e) {}

    @Override
    public void mouseReleased(MouseEvent e) {}

    @Override
    public void mouseMoved(MouseEvent e) {}

    @Override
    public void mouseDragged(MouseEvent e) {}

    @Override
    public void mouseWheelMoved(MouseEvent e) {}
});
```

Add a keyboard listener :

```
viewer3D.getRenderFrame().addKeyListener(new KeyListener() {
    @Override
    public void keyPressed(KeyEvent e) {}

    @Override
    public void keyReleased(KeyEvent e) {}
});
```

Add a window listener :

```
viewer3D.getRenderFrame().addWindowListener(new WindowListener() {
    @Override
    public void windowResized(WindowEvent e) {}

    @Override
    public void windowMoved(WindowEvent e) {}

    @Override
    public void windowDestroyNotify(WindowEvent e) {}

    @Override
    public void windowDestroyed(WindowEvent e) {}

    @Override
    public void windowGainedFocus(WindowEvent e) {}

    @Override
    public void windowLostFocus(WindowEvent e) {}

    @Override
    public void windowRepaint(WindowUpdateEvent e) {}
});
```

II.1.7.8 – Handling events

The Viewer3D add automatically a default event manager at instantiation.
This default event manager provides the following functionnalities :

Command	Action
Middle Mouse button clicked	update mouse picker
Left mouse button down + mouse dragging	rotate camera
Right mouse button down + mouse dragging	camera side translation
Mouse wheel down/up	camera forward translation
D key down	move light to the right
Q key down	move light to the left
Z key down	move light to top
S key down	move light to bottom
Numpad 1 pressed	set camera to look at the front side
Numpad 1 pressed + ctrl left down	set camera to look at the back side
Numpad 3 pressed	set camera to look at the right side
Numpad 3 pressed + ctrl left down	Set camera to look at the left side
Numpad 7 pressed	set camera to look at the top side
Numpad 7 pressed + ctrl left down	set camera to look at the bottom side

If you want you can override the default comportment with the call to « viewer3D.removeDefaultEventManager() ».
Then you can add your event manager.

```
//remove the default event manager
viewer3D.removeDefaultEventManager();

viewer3D.addEventListener(new EventManager(new InputMouseAdapter(), new
InputKeyListener()) {
    @Override
    public void updateEvents() {
        //handle the events here

        //get access to the mouse
        mouse.isButtonDown(InputMouseAdapter.Button.LEFT);

        //get access to the keyboard
        keyboard.isKeyDown(KeyEvent.VK_E);
    }
});
```

III – Use module from other projects

The current architecture allows to use AMAPVox modules as dependencies.
To accomplish this in a Maven project, you have to :

Step 1 : In POM file add repository server :

```
<repositories>
    <repository>
        <id>github</id>
        <url>https://rawgit.com/MilWolf/AMAPVox/mvn-repo</url>
    </repository>
</repositories>
```

The curent address is « https://rawgit.com/MilWolf/AMAPVox/mvn-repo » but it could change.

Step 2 : Add module dependency (example for las reader module):

```
<dependencies>
    <dependency>
        <groupId>fr.amap.lidar</groupId>
        <artifactId>format-las</artifactId>
        <version>1.0</version>
    </dependency>
</dependencies>
```