A closer look at some examples from the grogra.de gallery

Michael Henke

Department Ecoinformatics, Biometrics and Forest Growth,
University of Göttingen, Germany

Tutorial and Workshop
"Modelling with GroIMP and XL"
combined with the 5th GroIMP user and developer meeting

Göttingen, 2012-02-27

Model configuration

- Why:
  - Reconstruction of configurations
  - Systematic scenarios test
- How:
  - External property file

Model configuration

- Why:
  - Reconstruction of configurations
  - Systematic scenarios test
- How:
  - External property file

Common solutions:

Rewriting:

```
final int A = 5; // scenario B: 3
final float B = 5.1; // 5.3
```

Model configuration

- Why:
  - Reconstruction of configurations
  - Systematic scenarios test
- How:
  - External property file

Common solutions:

Rewriting:

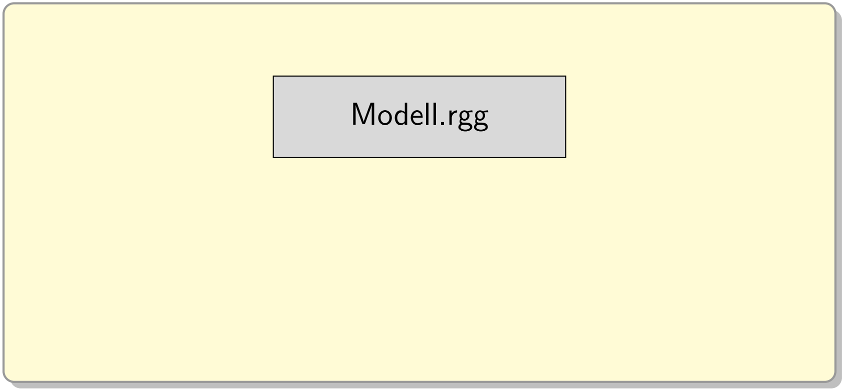
```
final int A = 5; // scenario B: 3
final float B = 5.1; // 5.3
```

Array:

```
final int SCENARIO = 1;

final int[] A = {5, 3};
final float[] B = {5.1, 5.3};

int c = A[SCENARIO] + ...;
```

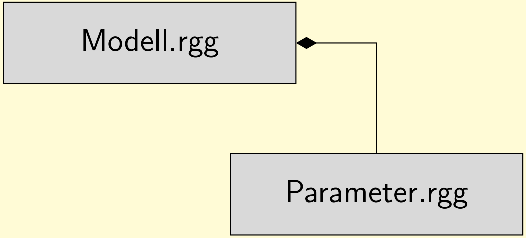
Model configuration

- Why:
    - Reconstruction of configurations
    - Systematic scenarios test
- How:
    - External property file



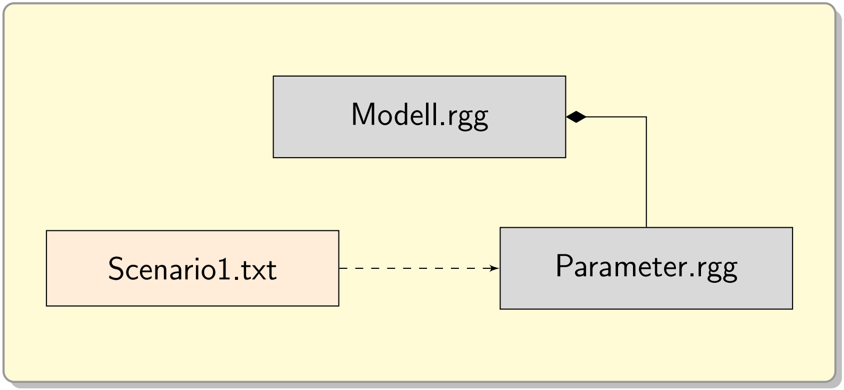
Model configuration

- Why:
  - Reconstruction of configurations
  - Systematic scenarios test
- How:
  - External property file



Model configuration

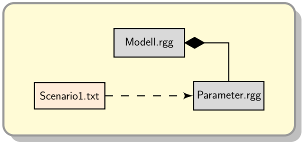
- Why:
  - Reconstruction of configurations
  - Systematic scenarios test
- How:
  - External property file



Model configuration - Parameter.rgg

Parameter.rgg

```
//import
import de.grogra.pf.io.PropertyFileReader;

/* PUBLIC VARIABLES AND CONSTANTS */
// linux
private final static String PATH = "/home/../../";
// windows
//private final static String PATH = "c:\\...\\..\\";

// property file
private final static String SCENARIO_FILE_NAME = "...";

/* Variable declaration, loaded by initParameters() */
// test boolean
protected static boolean BOX;

/* Help functions to load global parameters */
protected static boolean initParameters() {
... loadPropertyFile(); ...
}
```

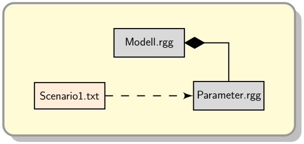


- Imports

Model configuration - Parameter.rgg

Parameter.rgg

```
//import
import de.grogra.pf.io.PropertyFileReader;

/* PUBLIC VARIABLES AND CONSTANTS */
// linux
private final static String PATH = "/home/../../";
// windows
//private final static String PATH = "c:\\...\\..\\";

// property file
private final static String SCENARIO_FILE_NAME = "...";

/* Variable declaration, loaded by initParameters() */
// test boolean
protected static boolean BOX;

/* Help functions to load global parameters */
protected static boolean initParameters() {
... loadPropertyFile(); ...
}
```

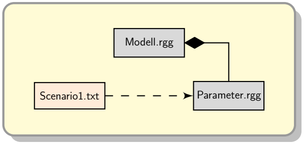


- Imports
- Constants

Model configuration - Parameter.rgg

Parameter.rgg

```
//import
import de.grogra.pf.io.PropertyFileReader;

/* PUBLIC VARIABLES AND CONSTANTS */
// linux
private final static String PATH = "/home/../../";
// windows
//private final static String PATH = "c:\\...\\..\\";

// property file
private final static String SCENARIO_FILE_NAME = "...";

/* Variable declaration, loaded by initParameters() */
// test boolean
protected static boolean BOX;

/* Help functions to load global parameters */
protected static boolean initParameters() {
... loadPropertyFile(); ...
}
```

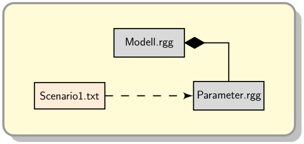


- Imports
- Constants
- Model Parameters

Model configuration - Parameter.rgg

Parameter.rgg

```
//import
import de.grogra.pf.io.PropertyFileReader;

/* PUBLIC VARIABLES AND CONSTANTS */
// linux
private final static String PATH = "/home/../../";
// windows
//private final static String PATH = "c:\\...\\..\\";

// property file
private final static String SCENARIO_FILE_NAME = "...";

/* Variable declaration, loaded by initParameters() */
// test boolean
protected static boolean BOX;

/* Help functions to load global parameters */
protected static boolean initParameters() {
... loadPropertyFile(); ...
}
```



- Imports
- Constants
- Model Parameters
- Read Parameters

Model configuration - Parameter.rgg

Parameter.rgg

```
/* Help functions to load global parameters */
protected static boolean initParameters() {
  boolean error = false;
  error = error || loadPropertyFile();
  //error = error || loadClimateData();
  if (error) println("Error during reading of parameter file(s)!");
  return error;
}

private static boolean loadPropertyFile() {
  PropertyFileReader propertyFile = new PropertyFileReader(PATH + SCENARIO_FILE_NAME);

  //iff there was an error during reading the property file
  if(propertyFile.load()) return true;

  loadProperties(propertyFile);
  println("Parameter file successfully read. "+propertyFile.
       getNumberOfReadedProperties()+" parameter read.");
  return false;
}

private static void loadProperties(PropertyFileReader propertyFile) {
  BOX = propertyFile.getBoolean("BOX");
  ...
}
```

Model configuration - Parameter.rgg

Parameter.rgg

```
/* Help functions to load global parameters */
protected static boolean initParameters() {
  boolean error = false;
  error = error || loadPropertyFile();
  //error = error || loadClimateData();
  if (error) println("Error during reading of parameter file(s)!");
  return error;
}

private static boolean loadPropertyFile() {
  PropertyFileReader propertyFile = new PropertyFileReader(PATH + SCENARIO_FILE_NAME);

  //iff there was an error during reading the property file
  if(propertyFile.load()) return true;

  loadProperties(propertyFile);
  println("Parameter file successfully read. "+propertyFile.
       getNumberOfReadedProperties()+" parameter read.");
  return false;
}

private static void loadProperties(PropertyFileReader propertyFile) {
  BOX = propertyFile.getBoolean("BOX");
  ...
}
```

Model configuration - Parameter.rgg

Parameter.rgg

```
/* Help functions to load global parameters */
protected static boolean initParameters() {
  boolean error = false;
  error = error || loadPropertyFile();
  //error = error || loadClimateData();
  if (error) println("Error during reading of parameter file(s)!");
  return error;
}

private static boolean loadPropertyFile() {
  PropertyFileReader propertyFile = new PropertyFileReader(PATH + SCENARIO_FILE_NAME);

  //iff there was an error during reading the property file
  if(propertyFile.load()) return true;

  loadProperties(propertyFile);
  println("Parameter file successfully read. "+propertyFile.
       getNumberOfReadedProperties()+" parameter read.");
  return false;
}

private static void loadProperties(PropertyFileReader propertyFile) {
  BOX = propertyFile.getBoolean("BOX");
  ...
}
```

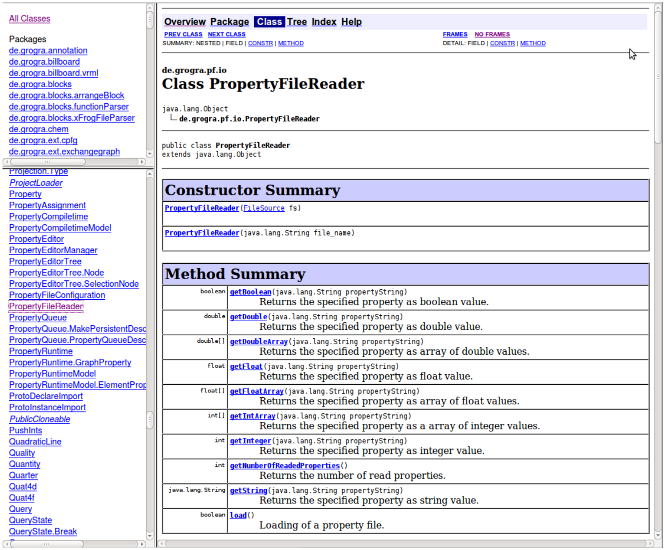
Model configuration - Parameter.rgg

Parameter.rgg

```
/* Help functions to load global parameters */
protected static boolean initParameters() {
  boolean error = false;
  error = error || loadPropertyFile();
  //error = error || loadClimateData();
  if (error) println("Error during reading of parameter file(s)!");
  return error;
}

private static boolean loadPropertyFile() {
  PropertyFileReader propertyFile = new PropertyFileReader(PATH + SCENARIO_FILE_NAME);

  //iff there was an error during reading the property file
  if(propertyFile.load()) return true;

  loadProperties(propertyFile);
  println("Parameter file successfully read. "+propertyFile.
       getNumberOfReadedProperties()+" parameter read.");
  return false;
}

private static void loadProperties(PropertyFileReader propertyFile) {
  BOX = propertyFile.getBoolean("BOX");
  ...
}
```

Model configuration - Parameter.rgg

GroIMP API


All Classes

Packages
de.grogra.annotation
de.grogra.billboard
de.grogra.billboard.vrml
de.grogra.blocks
de.grogra.blocks.arrangeBlock
de.grogra.blocks.functionParser
de.grogra.blocks.xFrogFileParser
de.grogra.chem
de.grogra.ext.cpfg
de.grogra.ext.exchangegraph

Projection.Type
ProjectLoader
Property
PropertyAssignment
PropertyCompiletime
PropertyCompiletimeModel
PropertyEditor
PropertyEditorManager
PropertyEditorTree
PropertyEditorTree.Node
PropertyEditorTree.SelectionNode
PropertyFileConfiguration
PropertyFileReader
PropertyQueue
PropertyQueue.MakePersistentDesc
PropertyQueue.PropertyQueueDesc
PropertyRuntime
PropertyRuntime.GraphProperty
PropertyRuntimeModel
PropertyRuntimeModel.ElementProp
ProtoDeclareImport
ProtoInstanceImport
PublicCloneable
PushInts
QuadraticLine
Quality
Quantity
Quarter
Quat4d
Quat4f
Query
QueryState
QueryState.Break

Overview Package Class Tree Index Help

PREV CLASS NEXT CLASS — FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD — DETAIL: FIELD | CONSTR | METHOD

de.grogra.pf.io
Class PropertyFileReader

```
java.lang.Object
  └ de.grogra.pf.io.PropertyFileReader
```

```
public class PropertyFileReader
extends java.lang.Object
```

Constructor Summary

PropertyFileReader(FileSource fs)
PropertyFileReader(java.lang.String file_name)

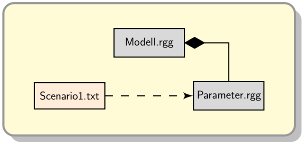
Method Summary

boolean	getBoolean(java.lang.String propertyString) Returns the specified property as boolean value.
double	getDouble(java.lang.String propertyString) Returns the specified property as double value.
double[]	getDoubleArray(java.lang.String propertyString) Returns the specified property as array of double values.
float	getFloat(java.lang.String propertyString) Returns the specified property as float value.
float[]	getFloatArray(java.lang.String propertyString) Returns the specified property as array of float values.
int[]	getIntArray(java.lang.String propertyString) Returns the specified property as a array of integer values.
int	getInteger(java.lang.String propertyString) Returns the specified property as integer value.
int	getNumberOfReadedProperties() Returns the number of read properties.
java.lang.String	getString(java.lang.String propertyString) Returns the specified property as string value.
boolean	load() Loading of a property file.

Model configuration - Model.rgg

Parameter.rgg

```
//import
import de.grogra.pf.io.PropertyFileReader;

/* PUBLIC VARIABLES AND CONSTANTS */
// linux
private final static String PATH = "/home/../../";
// windows
//private final static String PATH = "c:\\...\\..\\";

// property file
private final static String SCENARIO_FILE_NAME = "...";

/* Variabel declaration, loaded by initParameters() */
// test boolean
protected static boolean BOX;

/* Help functions to load global parameters */
protected static boolean initParameters() {
... loadPropertyFile(); ...
}
```



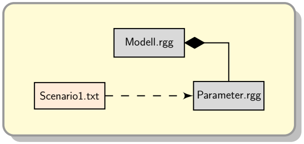
Model.rgg

```
import static Parameter.*;

protected void init () [
  {
    initParameters();
  }
  Axiom ==>
    if(!BOX) (
      Sphere(RADIUS).(
        setName(NAME),...
      ));
]
```

Model configuration - Model.rgg

Parameter.rgg

```
//import
import de.grogra.pf.io.PropertyFileReader;

/* PUBLIC VARIABLES AND CONSTANTS */
// linux
private final static String PATH = "/home/../../";
// windows
//private final static String PATH = "c:\\...\\..\\";

// property file
private final static String SCENARIO_FILE_NAME = "...";

/* Variabel declaration, loaded by initParameters() */
// test boolean
protected static boolean BOX;

/* Help functions to load global parameters */
protected static boolean initParameters() {
... loadPropertyFile(); ...
}
```



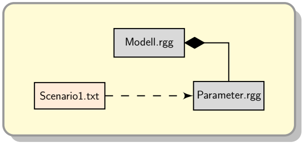
Model.rgg

```
import static Parameter.*;

protected void init () [
  {
   initParameters();
  }
  Axiom ==>
    if(!BOX) (
      Sphere(RADIUS).(
        setName(NAME),...
      ));
]
```

Model configuration - Model.rgg

Parameter.rgg

```
//import
import de.grogra.pf.io.PropertyFileReader;

/* PUBLIC VARIABLES AND CONSTANTS */
// linux
private final static String PATH = "/home/../../";
// windows
//private final static String PATH = "c:\\...\\..\\";

// property file
private final static String SCENARIO_FILE_NAME = "...";

/* Variabel declaration, loaded by initParameters() */
// test boolean
protected static boolean BOX;

/* Help functions to load global parameters */
protected static boolean initParameters() {
... loadPropertyFile(); ...
}
```



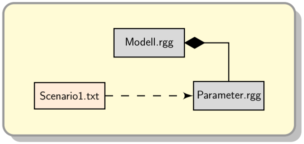
Model.rgg

```
import static Parameter.*;

protected void init () [
  {
    initParameters();
  }
  Axiom ==>
    if(!BOX) (
      Sphere(RADIUS).(
        setName(NAME),...
      ));
]
```

Model configuration - Scenario1.txt

Scenario1.txt

```
// Example of a property file

// test boolean
BOX = false

// colour
RGB = 0.2,0.4, 0.3

// name
NAME = myObject

// object radius
RADIUS = 0.55

// and one integer
NUMBER = 42
```



Syntax:

$<$key$>$ = $<$value$>$

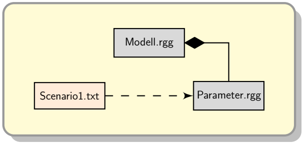
$<$key$>$ = $<$value$_1$ $>$,

…, $<$value$_n$ $>$

Model configuration - Scenario1.txt

Scenario1.txt

```
// Example of a property file

// test boolean
BOX = false

// colour
RGB = 0.2,0.4, 0.3

// name
NAME = myObject

// object radius
RADIUS = 0.55

// and one integer
NUMBER = 42
```



Syntax:

$<$key$>$ = $<$value$>$

$<$key$>$ = $<$value$_1$ $>$,

$\ldots$, $<$value$_n$ $>$

$\Longrightarrow$ Example: *Gallery/Technics/PropertyFileDemo.zip*

Automated model runs

Additional test function:

```
public void testScenarios() {
  SCENARIO_FILE_NAME = "Scenario1.txt";
  initModel();
  runModel();

  ...

  SCENARIO_FILE_NAME = "ScenarioN.txt";
  initModel();
  runModel();
}
```

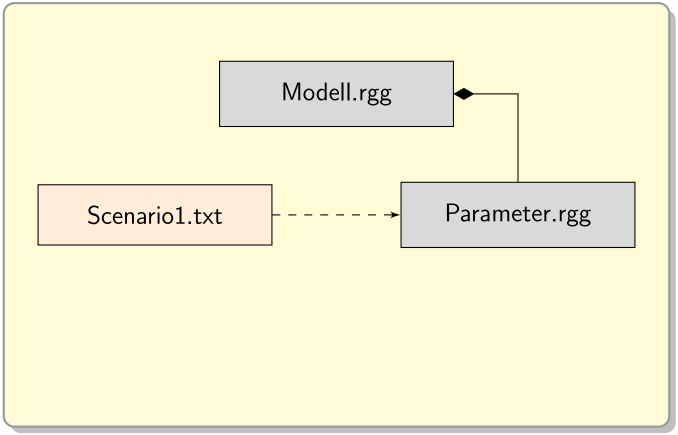
- Sequence of model runs with different configuration

Import Excel Files

- Why:
  - Import large sets of data
- How:
  - Using Apache POI library

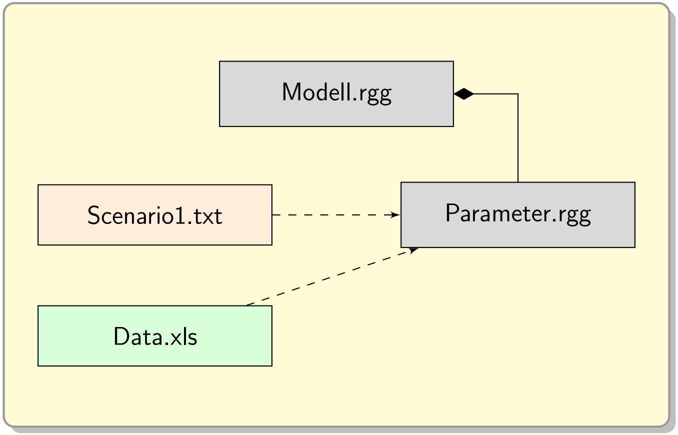
Import Excel Files

- Why:
  - Import large sets of data
- How:
  - Using Apache POI library



Import Excel Files

- Why:
  - Import large sets of data
- How:
  - Using Apache POI library



Import Excel Files

```java
// imports
import org.apache.poi.ss.usermodel.*;
...

// data set
protected static float[] DATA;
...

private static void loadDataFile(String inFile) {
  InputStream inp = new FileInputStream(inFile);
  Workbook wb = WorkbookFactory.create(inp);
  Sheet sheet = wb.getSheetAt(0);

  // data to arrays
  DATA = new float[sheet.getLastRowNum()+1];
  int i = 0;
  for (Iterator rit = sheet.rowIterator(); rit.hasNext();) {
    Row row = (Row)rit.next();
    Iterator cit = row.cellIterator(); cit.hasNext();

    DATA[i] = getNumeric((Cell)cit.next());
    i++;
  }
}
```

Import Excel Files

```
// imports
import org.apache.poi.ss.usermodel.*;
...

// data set
protected static float[] DATA;
...

private static void loadDataFile(String inFile) {
  InputStream inp = new FileInputStream(inFile);
  Workbook wb = WorkbookFactory.create(inp);
  Sheet sheet = wb.getSheetAt(0);

  // data to arrays
  DATA = new float[sheet.getLastRowNum()+1];
  int i = 0;
  for (Iterator rit = sheet.rowIterator(); rit.hasNext();) {
    Row row = (Row)rit.next();
    Iterator cit = row.cellIterator(); cit.hasNext();

    DATA[i] = getNumeric((Cell)cit.next());
    i++;
  }
}
```

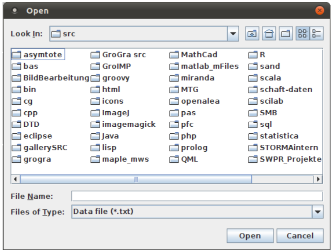
⟹ Example: *Gallery/Technics/ExcelFileDemo.zip*

File Chooser Example

```java
import FileChooserDemo.*;

protected void init () {
  File file;
  // file chooser or hard coded file
  if(USE_FILE_CHOOSER) {
    FileChooserDemo fcd = new FileChooserDemo();
    file = fcd.getFile();
  } else {
    file = new File("/../M2-new.ply");
  }
  ...

  if(file != null) {
    double[] points = readFile(file);
  }
}
```

File Chooser Example

```
import FileChooserDemo.*;

protected void init () {
  File file;
  // file chooser or hard coded file
  if(USE_FILE_CHOOSER) {
    FileChooserDemo fcd = new FileChooserDemo();
    file = fcd.getFile();
  } else {
    file = new File("/../M2-new.ply");
  }
  ...

  if(file != null) {
    double[] points = readFile(file);
  }
}
```


Open

Look In: src

asymtote
bas
BildBearbeitung
bin
cg
cpp
DTD
eclipse
gallerySRC
grogra
GroGra src
GroIMP
groovy
html
icons
ImageJ
imagemagick
Java
lisp
maple_mws
MathCad
matlab_mFiles
miranda
MTG
openalea
pas
pfc
php
prolog
QML
R
sand
scala
schaft-daten
scilab
SMB
sql
statistica
STORMAintern
SWPR_Projekte

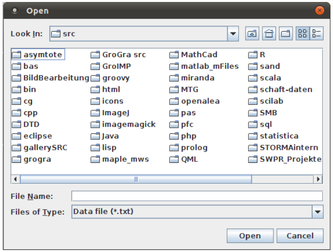
File Name:

Files of Type: Data file (*.txt)

Open Cancel

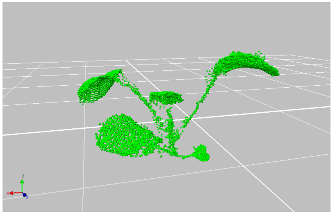
File Chooser Example

```
import FileChooserDemo.*;

protected void init () {
  File file;
  // file chooser or hard coded file
  if(USE_FILE_CHOOSER) {
    FileChooserDemo fcd = new FileChooserDemo();
    file = fcd.getFile();
  } else {
    file = new File("/../M2-new.ply");
  }
  ...

  if(file != null) {
    double[] points = readFile(file);
  }
}
```


Look In: src

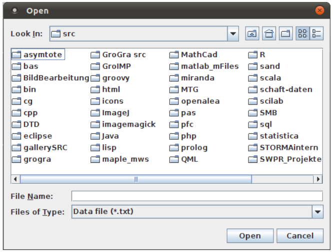
File Name:

Files of Type: Data file (*.txt)



File Chooser Example

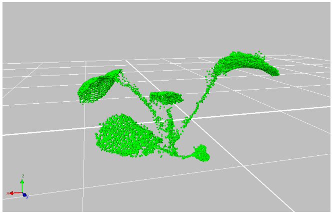
```java
import FileChooserDemo.*;

protected void init () {
  File file;
  // file chooser or hard coded file
  if(USE_FILE_CHOOSER) {
    FileChooserDemo fcd = new FileChooserDemo();
    file = fcd.getFile();
  } else {
    file = new File("/../M2-new.ply");
  }
  ...

  if(file != null) {
    double[] points = readFile(file);
  }
}
```


Open

Look In: src

File Name:

Files of Type: Data file (*.txt)



⟹ Example: *Gallery/Technics/OpenFileDemo.zip*

Print in File

```
import java.io.*;

// global print writer
const PrintWriter tmpfile;

protected void init () [
  {
    tmpfile = new PrintWriter(
        new FileWriter("/home/.../data.txt"));
  }
  Axiom ==> A(1);
]

public void run () [
  A(x) ==> A(x*0.8)
    { tmpfile.println("a = " + x); };
]

public void end() {
  tmpfile.flush();
  tmpfile.close();
}
```

- Imports

Print in File

```
import java.io.*;

// global print writer
const PrintWriter tmpfile;

protected void init () [
  {
    tmpfile = new PrintWriter(
        new FileWriter("/home/.../data.txt"));
  }
  Axiom ==> A(1);
]

public void run () [
  A(x) ==> A(x*0.8)
    { tmpfile.println("a = " + x); };
]

public void end() {
  tmpfile.flush();
  tmpfile.close();
}
```

- Imports
- Global print writer and initialisation

Print in File

```
import java.io.*;

// global print writer
const PrintWriter tmpfile;

protected void init () [
  {
    tmpfile = new PrintWriter(
        new FileWriter("/home/.../data.txt"));
  }
  Axiom ==> A(1);
]

public void run () [
  A(x) ==> A(x*0.8)
    { tmpfile.println("a = " + x); };
]

public void end() {
  tmpfile.flush();
  tmpfile.close();
}
```

- Imports
- Global print writer and initialisation
- Run-method $\rightarrow$ write data

Print in File

```
import java.io.*;

// global print writer
const PrintWriter tmpfile;

protected void init () [
  {
    tmpfile = new PrintWriter(
        new FileWriter("/home/.../data.txt"));
  }
  Axiom ==> A(1);
]

public void run () [
  A(x) ==> A(x*0.8)
    { tmpfile.println("a = " + x); };
]

public void end() {
  tmpfile.flush();
  tmpfile.close();
}
```

- Imports
- Global print writer and initialisation
- Run-method → write data
- End-method → closing the file

Print in File

```
import java.io.*;

// global print writer
const PrintWriter tmpfile;

protected void init () [
  {
    tmpfile = new PrintWriter(
        new FileWriter("/home/.../data.txt"));
  }
  Axiom ==> A(1);
]

public void run () [
  A(x) ==> A(x*0.8)
    { tmpfile.println("a = " + x); };
]

public void end() {
  tmpfile.flush();
  tmpfile.close();
}
```

- Imports
- Global print writer and initialisation
- Run-method $\rightarrow$ write data
- End-method $\rightarrow$ closing the file

$\Longrightarrow$ Example: *Gallery/Technics/print_in_file2.gsz*

Chart demo

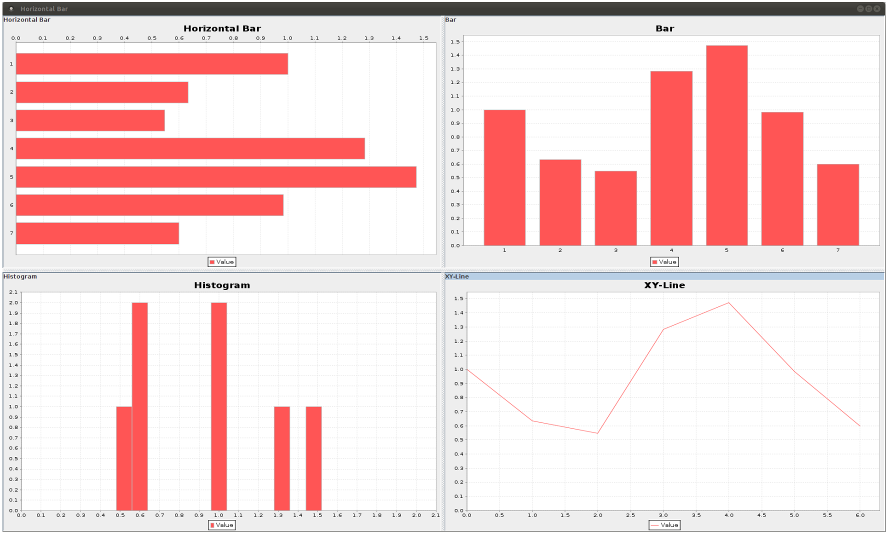
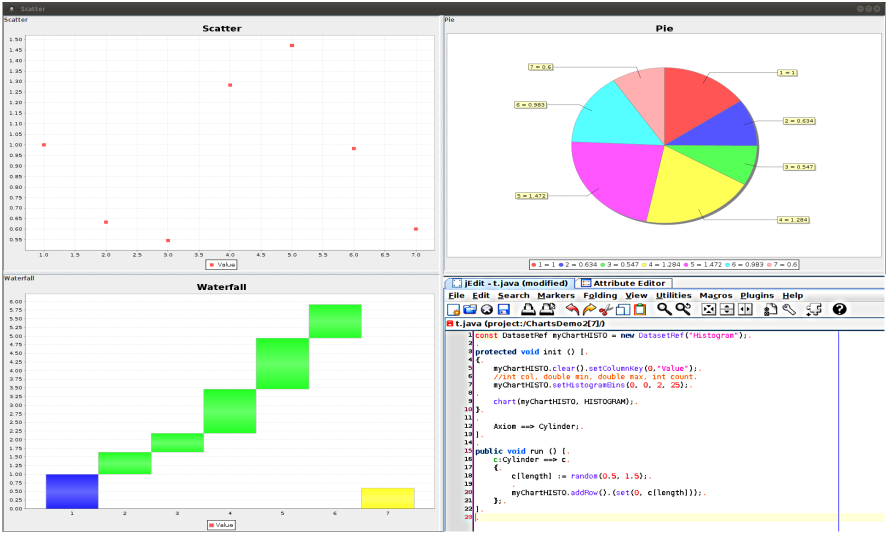
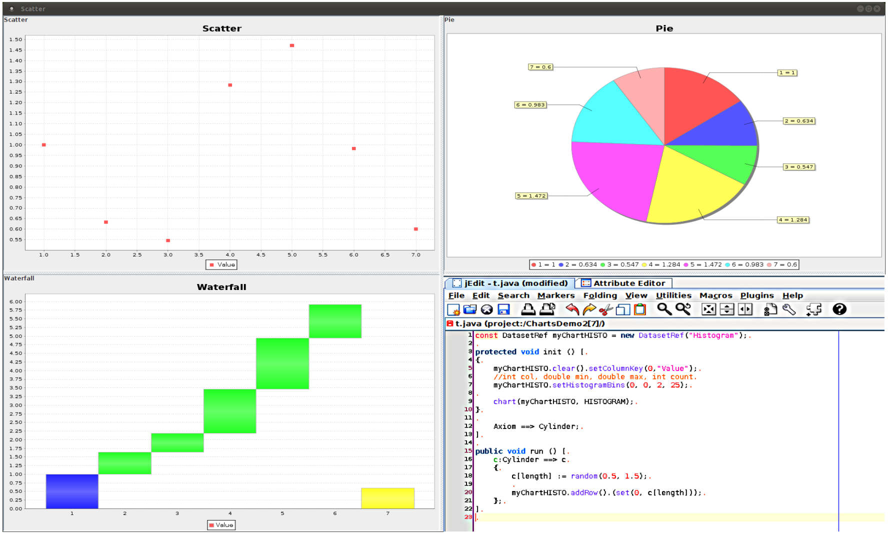

Horizontal Bar
Horizontal Bar
Horizontal Bar
Value
Bar
Bar
Value
Histogram
Histogram
Value
XY-Line
XY-Line
Value

Chart demo



```
const DatasetRef myChartHISTO = new DatasetRef("Histogram");

protected void init () [
{
    myChartHISTO.clear().setColumnKey(0,"Value");
    //int col, double min, double max, int count
    myChartHISTO.setHistogramBins(0, 0, 2, 25);

    chart(myChartHISTO, HISTOGRAM);
}

    Axiom ==> Cylinder;
]

public void run () [
    c:Cylinder ==> c
    {
        c[length] := random(0.5, 1.5);

        myChartHISTO.addRow().(set(0, c[length]));
    };
]

```

Chart demo

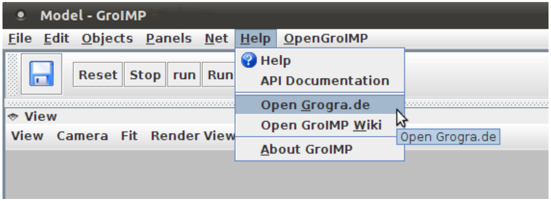


```
const DatasetRef myChartHISTO = new DatasetRef("Histogram");

protected void init () [
{
    myChartHISTO.clear().setColumnKey(0,"Value");
    //int col, double min, double max, int count
    myChartHISTO.setHistogramBins(0, 0, 2, 25);

    chart(myChartHISTO, HISTOGRAM);
}

    Axiom ==> Cylinder;
]

public void run () [
    c:Cylinder ==> c
    {
        c[length] := random(0.5, 1.5);

        myChartHISTO.addRow().(set(0, c[length]));
    };
]
```

⟹ Example: *Gallery/Technics/ChartsDemo2.gsz*

Where to get Help?

- GroIMP Gallery:
  http://www.grogra.de
- GroIMP Wiki page:
  http://sourceforge.net/apps/mediawiki/groimp/index.php?title=Main_Page



- E-Mail: info(at)grogra.de