

# Bio 7



# **Bio7 User Guide**

**Version 1.1**

©Department of Experimental and Systems Ecology

M. Austenfeld

<b>1</b>	<a href="#"><u>Introduction</u></a>	<a href="#"><u>S.1</u></a>
<b>2</b>	<a href="#"><u>Why Bio7 ?</u></a>	<a href="#"><u>S. 3</u></a>
	<b>2.1</b> Ecological Background	
<b>3</b>	<a href="#"><u>The components of Bio7</u></a>	<a href="#"><u>S.4</u></a>
	<b>3.1</b> Image Analysis	
	<b>3.2</b> Statistical Analysis	
	<b>3.3</b> Visualisation	
	<b>3.4</b> Database	
	<b>3.5</b> Plants	
	<b>3.6</b> Nutrients	
	<b>3.7</b> Sensitivity Analysis	
<b>4</b>	<a href="#"><u>Getting Started</u></a>	<a href="#"><u>S.11</u></a>
<b>5</b>	<a href="#"><u>The Platform Architecture</u></a>	<a href="#"><u>S.14</u></a>
	<b>5.1</b> The Toolbar	
	<b>5.2</b> The Menu	
<b>6</b>	<a href="#"><u>The Perspectives of Bio7</u></a>	<a href="#"><u>S.18</u></a>
	<b>6.1</b> The Resource Perspective	
	<b>6.1.1</b> The Navigator	
	<b>6.1.2</b> The Editor Area	
	<b>6.1.3</b> The Consoles of Bio7	
	<b>6.1.4</b> The Snippets view	
	<b>6.2</b> The 2d Perspective	
	<b>6.2.1</b> The Quadgrid	
	<b>6.2.2</b> The Hexgrid	
	<b>6.2.2.1</b> Storage of patterns and IBM properties	
	<b>6.2.3</b> The Linechart and the Piechart	
	<b>6.2.4</b> The Control	
	<b>6.2.4.1</b> Time and Space	
	<b>6.2.4.2</b> Plant Individual	
	<b>6.2.4.3</b> Disturbance	
	<b>6.2.4.4</b> Abiotic Factors	

- 6.2.5** Plants (States)
- 6.2.6** Spreadsheet
- 6.3** Image Perspective
  - 6.3.1** ImageJ-Canvas
  - 6.3.2** Points panel
    - 6.3.2.1** Using the Points panel for spatial continuous simulations
  - 6.3.3** Thumbnails
- 6.4** The Soil Perspective
  - 6.4.1** The Soil
  - 6.4.2** Nitrate
  - 6.4.3** Phosphate
  - 6.4.4** Carbon
  - 6.4.5** Water
  - 6.4.6** Root-Density
  - 6.4.7** Additionally
- 6.5** The Database Perspective
- 6.6** The 3d Perspective
- 6.7** Custom Perspectives
  - 6.7.1** Custom Views
  - 6.7.2** Custom Menus
  - 6.7.3** Custom Startup

## **7** [The Editors of Bio7](#)

[S.52](#)

- 7.1** General Features
  - 7.1.1** Code Templates
  - 7.1.2** Preferences of the Editors
- 7.2** The R-Editor
  - 7.2.1** Evaluation
  - 7.2.2** Plotting
- 7.3** The Java- and BeanShell Editor
  - 7.3.1** Java
  - 7.3.2** BeanShell
  - 7.3.3** Differences

<b>7.3.4</b>	The context menu of the Java (BeanShell) editor	
<b>7.4</b>	The Flow editor	
<b>7.4.1</b>	The Palette of the Flow editor	
<b>7.4.2</b>	Options for the Layout	
<b>7.4.3</b>	The Context menu	
<b>7.4.4</b>	The Properties for the Components	
<b>7.4.5</b>	What to consider when creating a flow	
<b>7.4.6</b>	How to cancel a flow	
<b>7.4.7</b>	Executed Files	
<b>7.4.7.1</b>	Dropped from the Navigator	
<b>7.4.7.2</b>	Dropped from outside the Navigator	
<b>7.4.8</b>	Important to know	
<b>8</b>	<a href="#">The Preferences</a>	<a href="#">S.76</a>
<b>8.1</b>	The Bio7 Preferences	
<b>8.1.1</b>	Application Paths	
<b>8.1.2</b>	Script Paths	
<b>8.1.3</b>	Bio7 Editor Preferences	
<b>8.1.4</b>	Preferences Database	
<b>8.1.5</b>	Preferences Rserve	
<b>8.2</b>	Default Platform Preferences	
<b>8.2.1</b>	General	
<b>8.2.2</b>	Colours and Fonts	
<b>8.2.3</b>	Text Editors Preference Page	
<b>8.2.4</b>	Help Preferences	
<b>8.2.5</b>	Perspectives	
<b>8.2.6</b>	File Associations	
<b>9</b>	<a href="#">Help</a>	<a href="#">S. 87</a>
<b>10</b>	<a href="#">Installing new features with the update manager</a>	<a href="#">S. 87</a>
<b>11</b>	<a href="#">Additional views in Bio7</a>	<a href="#">S. 89</a>
<b>12</b>	<a href="#">External tools</a>	<a href="#">S. 90</a>
<b>12.1</b>	R features of Bio7	
<b>12.2</b>	OpenOffice features of Bio7	

<b>12.3</b>	ImageJ features of Bio7	
<b>13</b>	<a href="#">What else</a>	<a href="#">S. 96</a>
<b>13.1</b>	Plugins for Bio7	
<b>13.2</b>	Programming examples for Bio7	
<b>14</b>	<a href="#">How to's</a>	<a href="#">S. 97</a>
<b>14.1</b>	<a href="#">First steps</a>	<a href="#">S. 97</a>
<b>14.1.1</b>	Game of Life	
<b>14.1.2</b>	Random Walk	
<b>14.1.3</b>	Gears (3d)	
<b>14.1.4</b>	A Matrix Model in R	
<b>14.2</b>	<a href="#">The first selfmade grid based simulation</a>	<a href="#">S. 99</a>
<b>14.2.1</b>	Explanation	
<b>14.2.2</b>	Again in BeanShell	
<b>14.2.3</b>	Explanation	
<b>14.2.4</b>	Conclusion	
<b>14.3</b>	<a href="#">The first gridded IBM Model</a>	<a href="#">S. 101</a>
<b>14.3.1</b>	How to get and set Plant values in the Quadgrid(Hexgrid)	
<b>14.3.2</b>	Annotation	
<b>14.3.3</b>	Copying instead of creating new objects	
<b>14.4</b>	<a href="#">Point Pattern Analysis</a>	<a href="#">S. 111</a>
<b>14.4.1</b>	Additionally	
<b>14.4.2</b>	Getting all attributes of a Particle Analysis	
<b>14.4.3</b>	Conclusion	
<b>14.5</b>	<a href="#">Transfer a spatial point pattern to a grid</a>	<a href="#">S. 120</a>
<b>14.6</b>	<a href="#">Import and Export Projects and files</a>	<a href="#">S. 122</a>
<b>14.6.1</b>	Import existing projects	
<b>14.6.2</b>	Import resources from the file system	
<b>14.6.3</b>	Import resources from an Archive file	
<b>14.6.4</b>	Export resources to the file system	
<b>14.6.5</b>	Export resources to an Archive file	
<b>14.7</b>	<a href="#">Execute, evaluate and plot a R script</a>	<a href="#">S. 127</a>
<b>14.8</b>	<a href="#">Create a custom GUI for a Rscript</a>	<a href="#">S. 129</a>

<b>14.9</b>	<a href="#">Create a Flash file presentation from a simulation</a>	S. 129
<b>14.9.1</b>	Note	
<b>14.10</b>	<a href="#">Open perspectives and views programmatically</a>	S. 134
<b>14.11</b>	<a href="#">Add states without the database</a>	S. 134
<b>14.12</b>	<a href="#">Add self defined menus to Bio7 with scripts</a>	S. 136
<b>14.13</b>	<a href="#">Compile Java files to BeanShell</a>	S. 139
<b>14.14</b>	<a href="#">Call compiled methods in BeanShell in the calculation thread of Bio7</a>	S. 140
<b>14.15</b>	<a href="#">Use R calculations interactively in a Java method</a>	S. 140
<b>14.16</b>	<a href="#">Import external libraries to BeanShell</a>	S. 142
<b>14.17</b>	<a href="#">Compile and run Java files</a>	S. 143
<b>15</b>	<a href="#">Notes</a>	S. 144
<b>15.1</b>	Notes about R	
<b>15.2</b>	Notes about BeanShell	
<b>15.3</b>	Notes about the Java-Compiler	
<b>15.4</b>	Notes about ImageJ	
<b>16</b>	<a href="#">Literature and Links</a>	S. 147
<b>16.1</b>	Useful Libraries	
<b>16.2</b>	Documentation and programming links	
<b>16.3</b>	Useful Literature	
<b>17</b>	<a href="#">Addendum</a>	S. 149
<b>18</b>	<a href="#">Definition of expressions</a>	S. 151



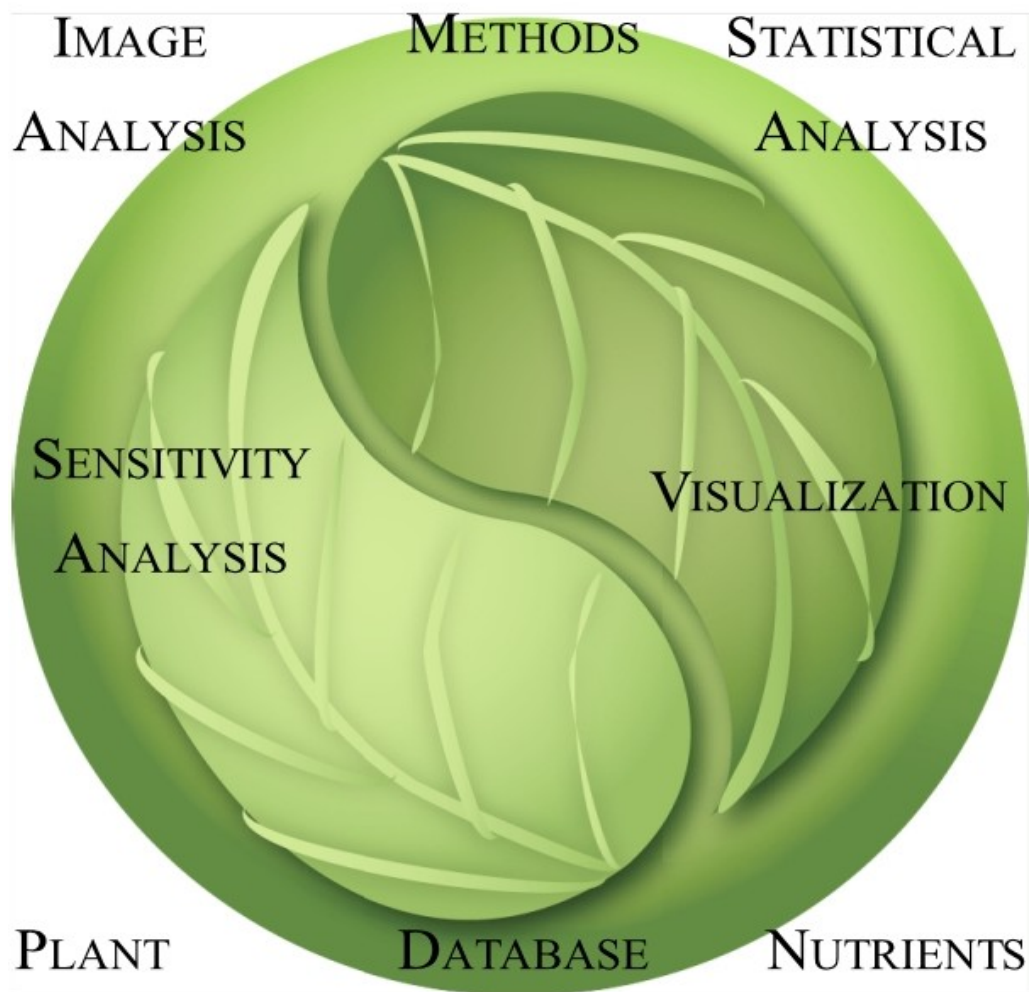


# 1 Introduction

The application Bio7 is an integrated development environment for ecological modelling with a main focus on individual based modelling and spatially explicit models. The application itself is based on a RCP-Eclipse-Environment (**R**ich-**C**lient-**P**latform) which offers a huge flexibility in configuration and extensibility because of its plug-in structure and the possibility of customization. In this platform powerful known tools were integrated for the purpose of simulations and analysis of ecological systems. A lot of complexity of a modelling process is hidden behind a user-friendly flexible **G**raphical **U**ser **I**nterface which should assist the development and analysis of simulation models for different ecosystems.

Therefore Bio7 offers and embeds also powerful and well proofed third party tools which are capable to do Image Analysis (ImageJ), Statistical Analysis (R), an easy to learn scripting language (BeanShell) to ease the development of simulation models and an embedded Compiler for the need of speed in complex calculations. Additionally editors (Java, R, BeanShell) and different panels for visualization are integrated for an easier development. Communication in between the different components is possible by means of scripts and compiled code and can be collected in a flowchart for Sensitivity-Analysis, etc.

Furthermore an advantage of this platform is the plugin structure which can be extended with all kind of third-party plugins to support theoretical hypothesis on ecosystems functioning and the understanding of complex systems. The following picture illustrates the different components of Bio7.



## 2 Why Bio7 ?

### 2.1 Ecological Background

One research area of the chair of experimental and systems biology (Bio7) are dry acidic grasslands on inland sand dunes. Only specialists like *Corynephorus canescens*, *Polytrichum piliferum*, etc. can settle on this nutrient poor water limited sandy soils. Additionally high temperatures on the surface of the soil and erosion make it very hard for other species to settle there. In comparison to other ecosystems this habitat is less complex in its species structure and nutrient dynamics which facilitates monitoring of vegetation dynamics to identify the driving forces or key factors of the system. With the survey of this ecosystem several questions and hypothesis should be answered and verified. Because many specialized plants and animals occur in this habitat the main question is, how we can protect and preserve this valuable ecosystem at all. Also the key Factors are from high interest and can maybe serve to identify similar processes in more complex systems like the tropical rainforest. From various experiments in these systems we have knowledge about seed-dispersion, competition, and nutrients availability or dependencies from different species which are typical for this habitat. We use the capabilities of this application to express the features of this system in syntactical rules for simulation.



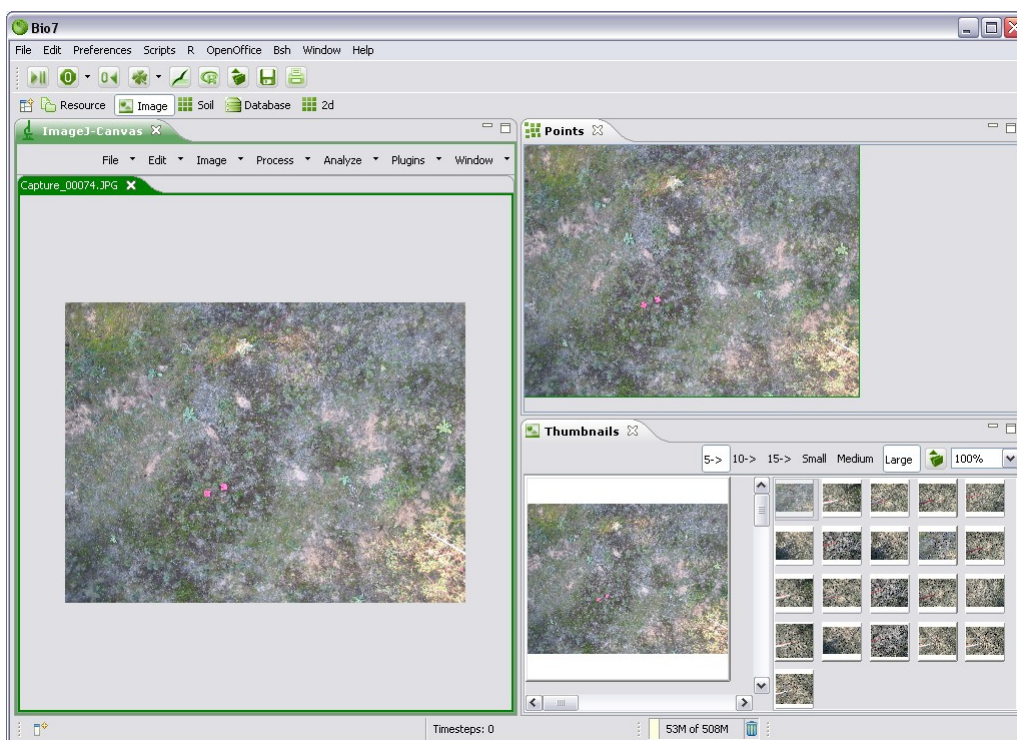
## 3 The Components of Bio7

### 3.1 Image Analysis

Image Analysis is a fundamental method to observe patterns and processes behind them.

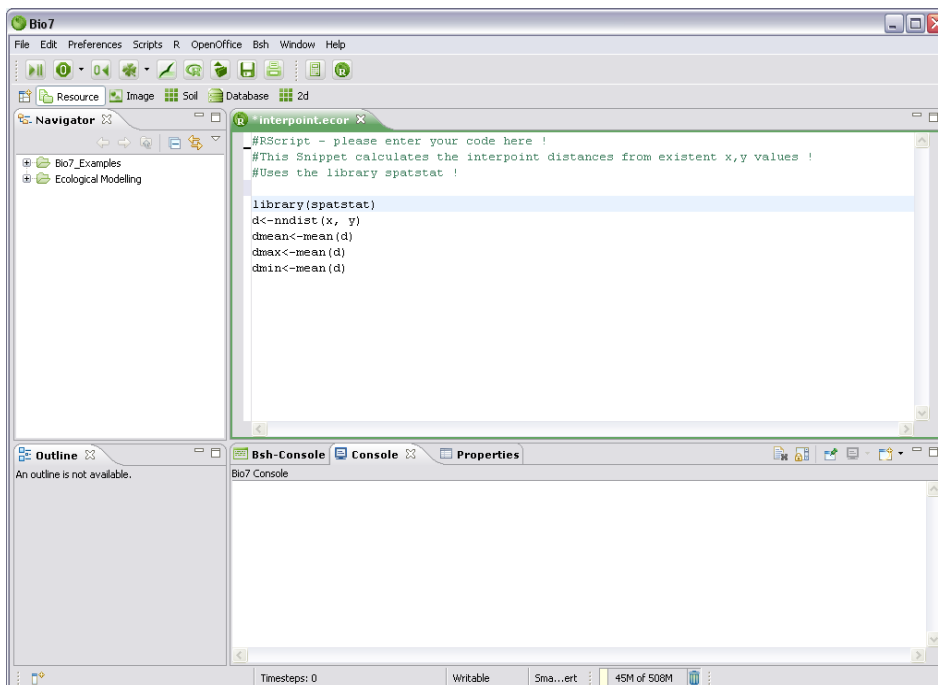
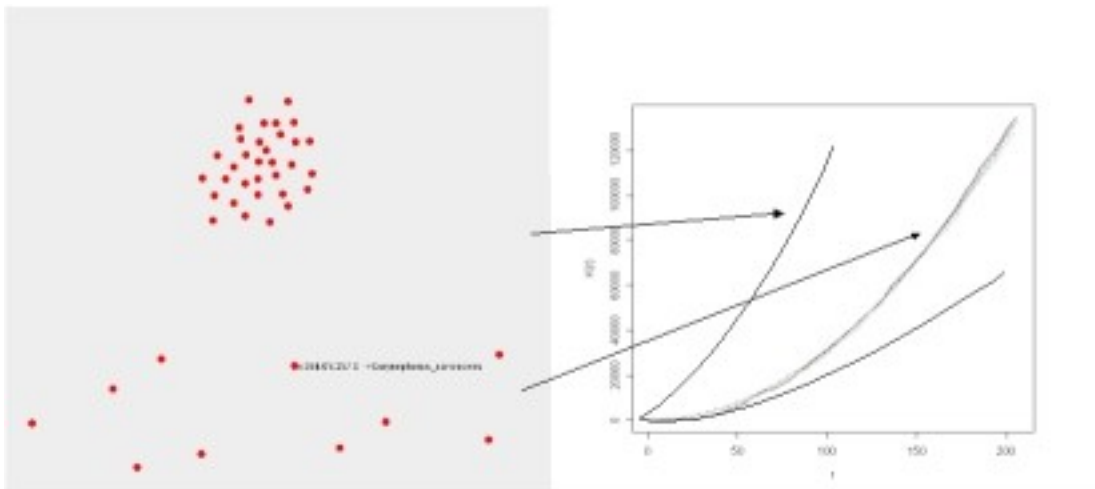
Bio7 addresses this with direct integration of the well proofed analysis tool ImageJ.

With this interface it is possible to detect pattern and integrate real pattern in a spatial simulation. It is also easy to combine the image analysis with the statistical tool "R" to make well known point-pattern analysis, voronoi-diagrams, kriging, etc. This is realized by a special second drawing panel which offers the possibility to set points on top an optional image. These placed coordinates can be sent directly to "R". Another feature of Bio7 uses the strength of ImageJ to detect patterns by combining spatially explicit simulation models with the detected pattern from image analysis. The measured patterns can be automatically transferred and assigned to a discrete pattern.



## 3.2 Statistical Analysis

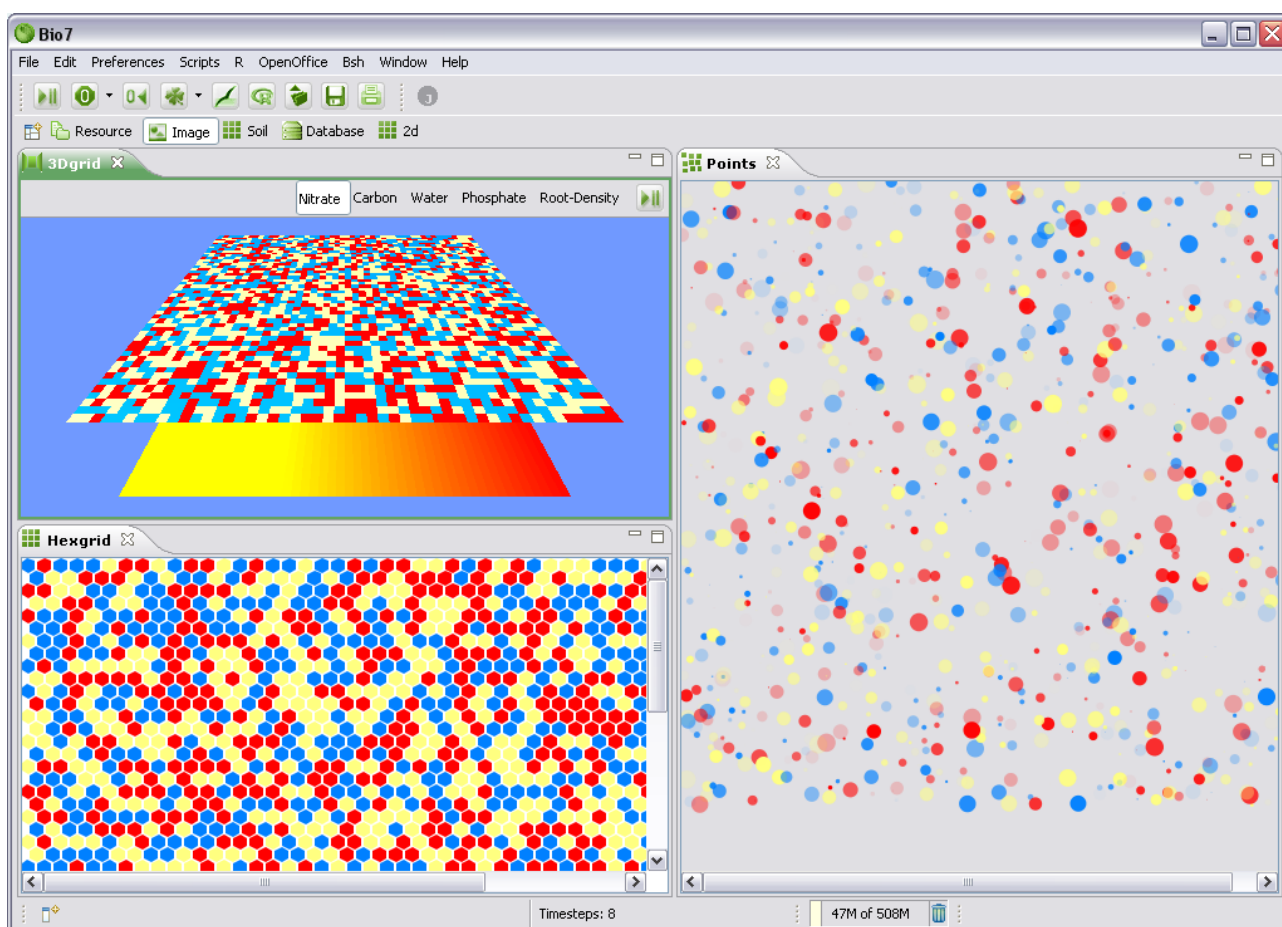
As a fundamental part of complexity analysis we have connected our application with the well known statistical program "R" to get well proofed statistical methods for our simulation. Simple scripts can be written to send data from a simulation to the statistical package and evaluate different scenarios. An integrated RScript-Editor allows the writing of required scripts and can send them to "R". It is also possible to get values from "R". Bio7 offers also a panel to make spatial analysis. The values can be sent directly to "R" to make a point-pattern-analysis, etc.





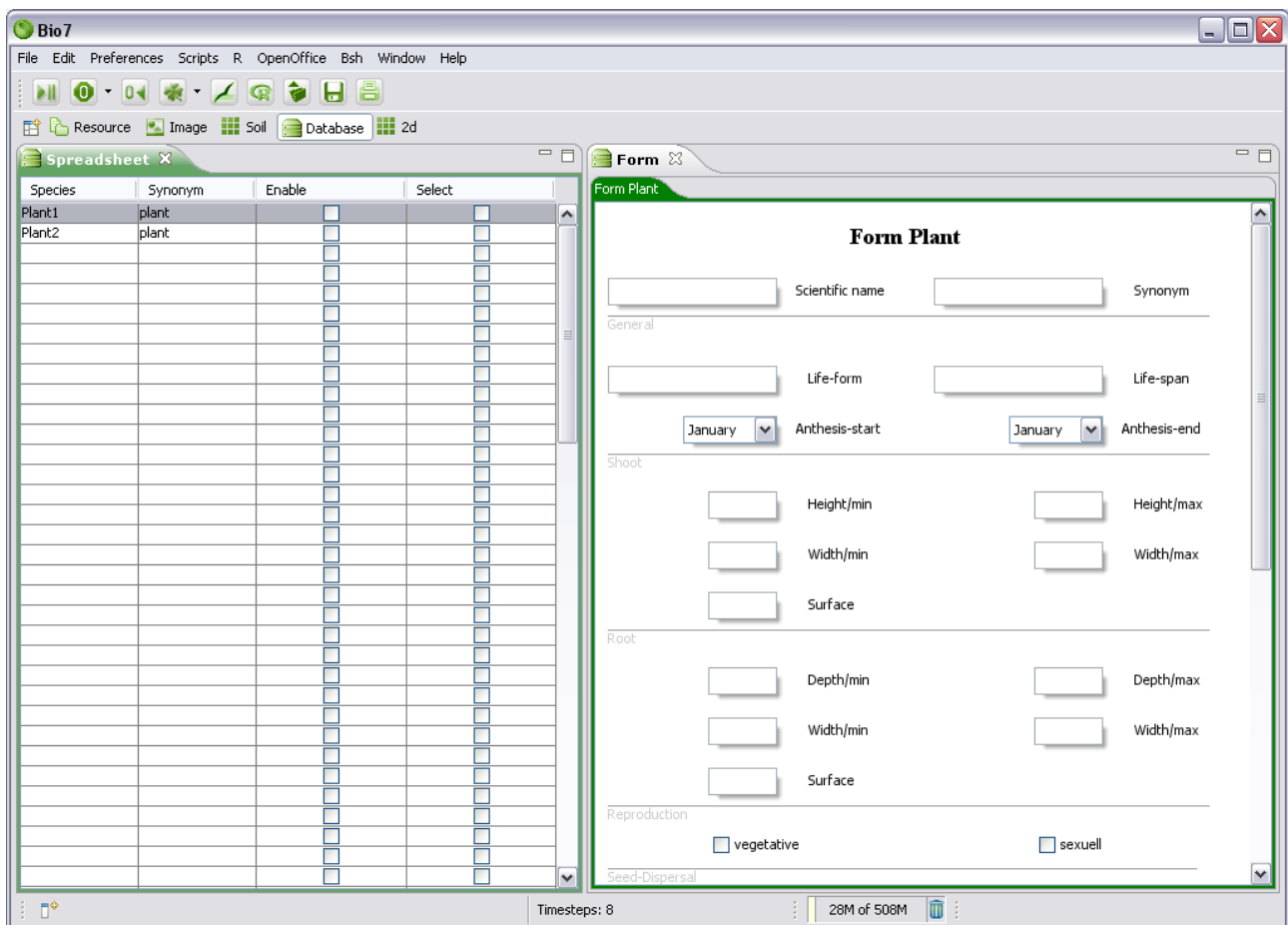
### 3.3 Visualization

Bio7 offers several visualization panels for spatial models. The application can visualize discrete hexagons and quads in 2-dimensional and 3-dimensional space. For spatial analysis and modelling Bio7 offers a 2-dimensional drawing panel. This panel can be used to discretize values via the Image Analysis support and set values which then can be send to the statistical tool to make all kind of spatial analysis. All panels can be accessed and modelled within an editor. The 2-dimensional panels are highly interactive to support the settings of values for nutrients and plants.



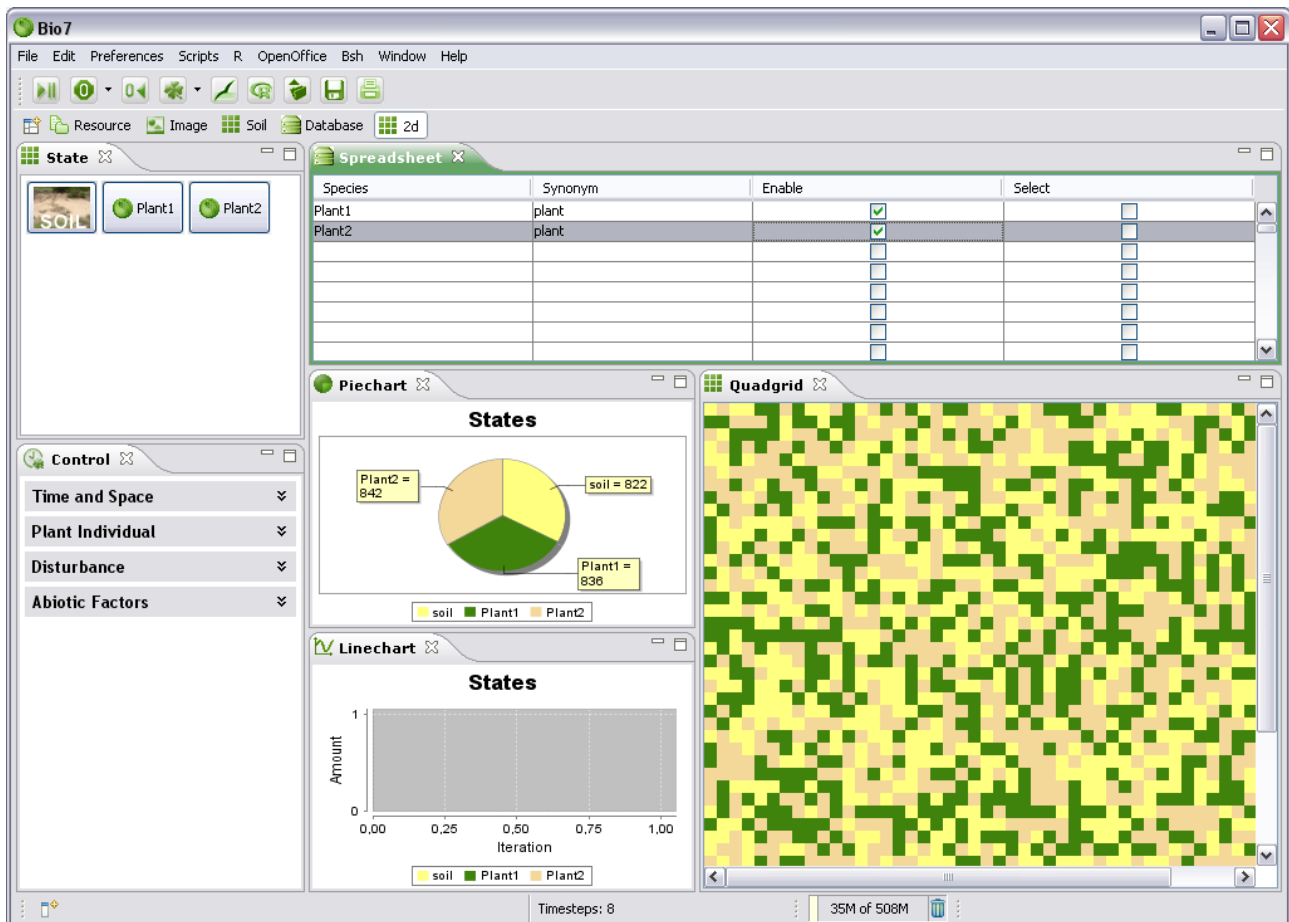
### 3.4 Database

All plants for simulation or analysis can be stored in an object-orientated database. The database-form contains general attributes which are useful in simulation and also descriptive for the plant species. For a simulation the database plants can be activated by the spreadsheet-form. After activation the plant-object and all of the plant attributes from the database are accessible. Also deleting or updating of values of a plant in the database is easily done by activating it in the spreadsheet.



### 3.5 Plants

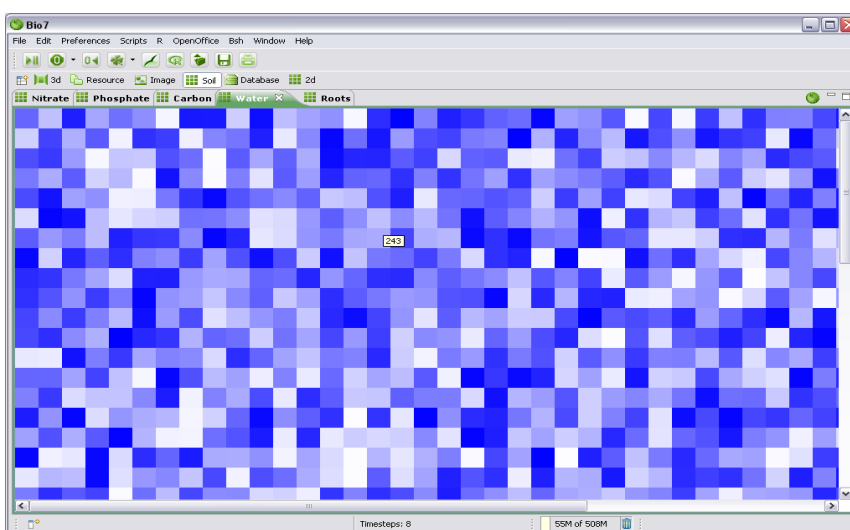
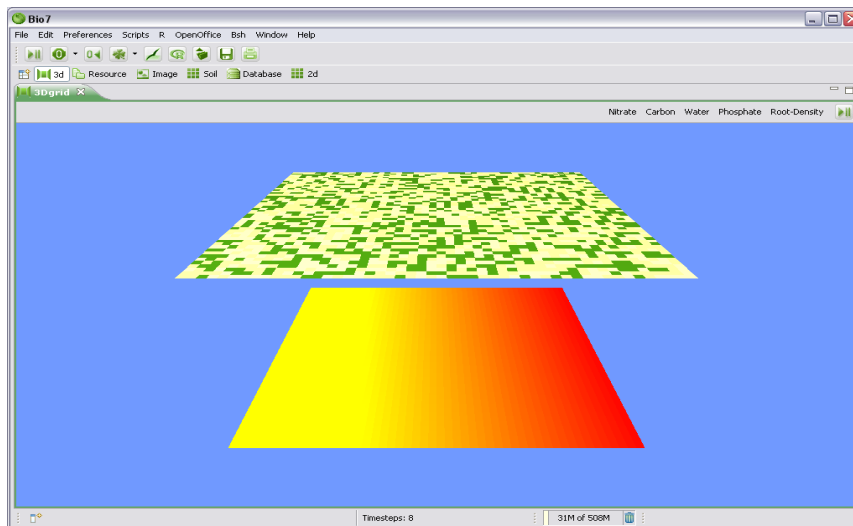
Bio7 is an integrated development environment with a strong support for individual based models. After activation in the database every cell in the visualization represents an individual plant. Each plant individual can then be adjusted by certain parameters





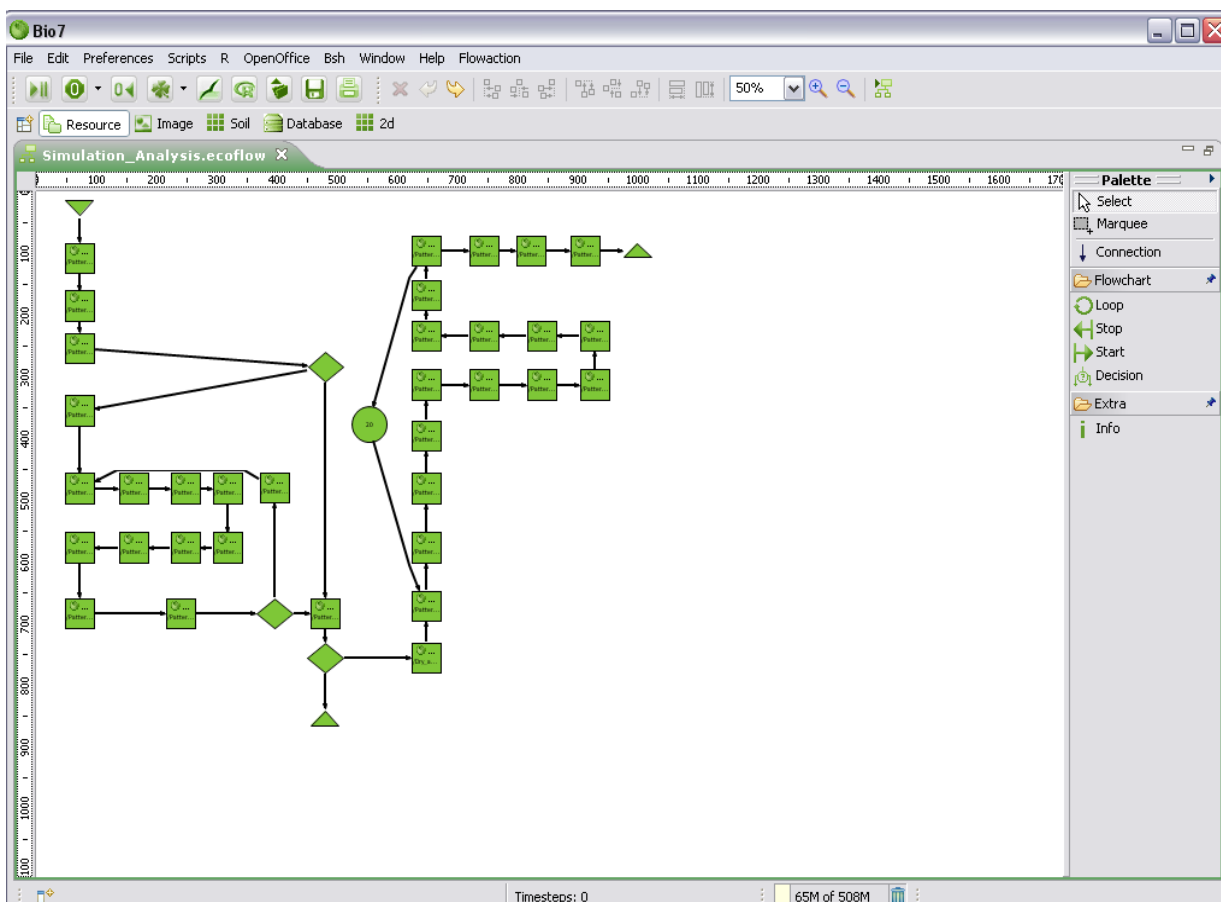
### 3.6 Nutrients

To analyse the effect of nutrient patterns in time and space the application offers possibilities to model the nutrient dynamics of different ecosystems. The dynamics of nutrients in a year can be set in a form. The model allows to set different nutrient layers parallel to the plant layer. The rules for the nutrient dynamics can be set in the editor. The layered view allows to analyse the dynamics with the plant layer above. This is helpful if you want to look at direct effects of nutrient dynamics.



### 3.7 Sensitivity Analysis

Ecosystems are highly dynamic in their aboveground and belowground processes. To test hypotheses on system functioning it is often required to run simulations under different conditions (nutrients, competition, etc.) and compare the results with the real system. The Flow editor in Bio7 is capable to handle the required flexibility to analyse a huge space of different situations and possibilities. All programmed methods and statistical analysis which are required can be dragged into the Flow editor. Because of the possibility of creating loops and decisions in between a flow of files hundreds or thousands situations with different parameters can be analysed with statistical well proofed methods ("R") and summarized.



## 4 Getting started

### System requirements

- It is recommended that your computer should have at least 512 mb ram and a 1ghz processor. The installation requires about 150 mb free space on the hard-disk.
- A 3d Graphics Card which is OpenGL enabled (only necessary for 3D view and for 3D programming!).
- To use the OpenOffice-feature of Bio7 an installation of OpenOffice  $\geq 2.0$  is required. The path to OpenOffice will be automatically fetched by Bio7 from the registry or is adjusted (Linux) to the default path (See section 8: The Bio7 Preferences).
- For the Flow editor: Older Windows versions require the gdi+ library which has to be downloaded from Microsoft and installed (e.g copied in the ..Windows/System32 folder)  
Candidates are: Windows NT, Windows 2000, Windows 98, Windows ME.

This is not necessary if the Flow editor is working by default!

### Windows

- For the convenience of the user the following applications are embedded:  
For statistical computing: Installation of **R** (and packages **spatstat**, **tripack**, **odesolve**) with the connecting application **Rserve**.
- A **Java Runtime Environment** **>1.5.0**

### Linux

Because of the different Linux distributions an installation of R is required. On the R website <http://www.r-project.org/> several R versions for different distributions are available.

Also the installation of the Rserve library is required. This can be easily done inside R by typing `install.packages("Rserve", dependencies=TRUE)` in the console.

After the installation of R the path to the R application has to be adjusted inside Bio7 (**Preferences->Preferences Bio7**).

Please install also the following libraries (**spatstat**, **tripack**) for the Bio7 examples.

The Bio7 application was tested on Ubuntu and Suse linux.

## Installation

Windows and Linux:

The installation of Bio7 is similar to the installation of the Eclipse environment.

Simply **decompress** the downloaded **\*.zip** file in a preferred location on your file system.

After decompressing with a standard zip-tool (like WinZip, Win Rar) the typical file-structure of an Eclipse based application will be created.

To start the application simply double click or click (linux) on the *Bio7.exe (Windows)* or *Bio7* (linux) file.

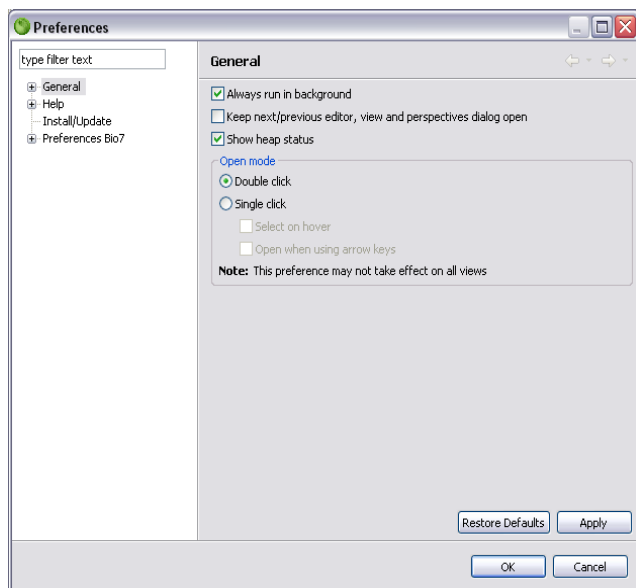
### Note (Windows):

It is also possible to install Bio7 on an Usb-Stick with the described procedure.

Moreover OpenOffice can also be installed portable with Bio7. Download the portable version ([http://portableapps.com/apps/office/openoffice\\_portable](http://portableapps.com/apps/office/openoffice_portable)) and adjust the OpenOffice path to the OpenOfficePortable\App\openoffice\program subfolder in the Preferences.

### After the first start

Please select the option **Preferences->General->Always run in background** to disable the progress monitor dialog of Bio7. This avoids a blocking dialog in many operations of Bio7 (important for the integrated Flow editor!)



### Troubleshooting for Windows and important to know !

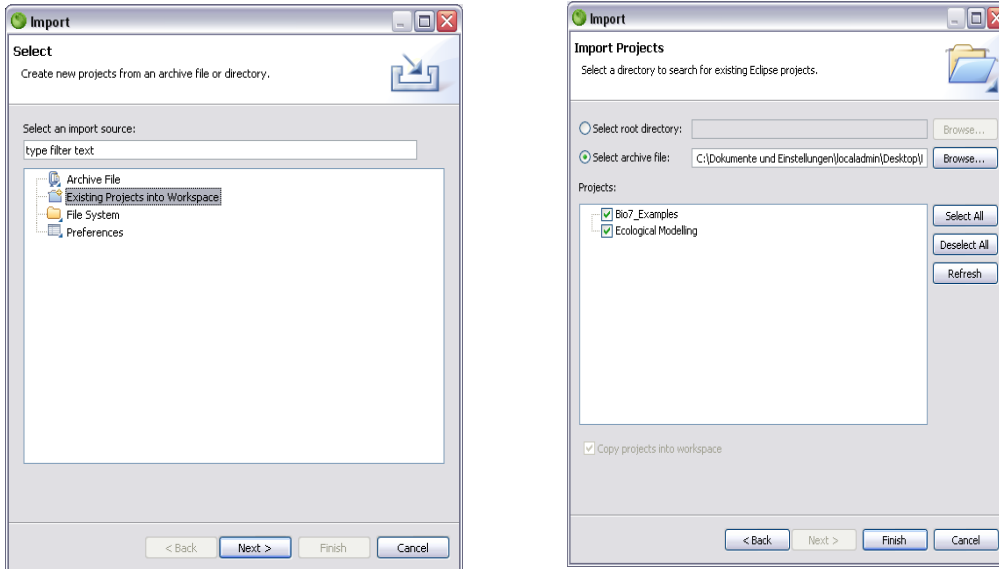
The Rserve application on Windows is sensible to wrong input.

If a failure in an R-expression occurs the Server has to be restarted again!

## Examples

On the Bio7 SourceForge website you can download examples for the Bio7 application. To install the Bio7 examples please import the examples from the *Examples.zip* file.

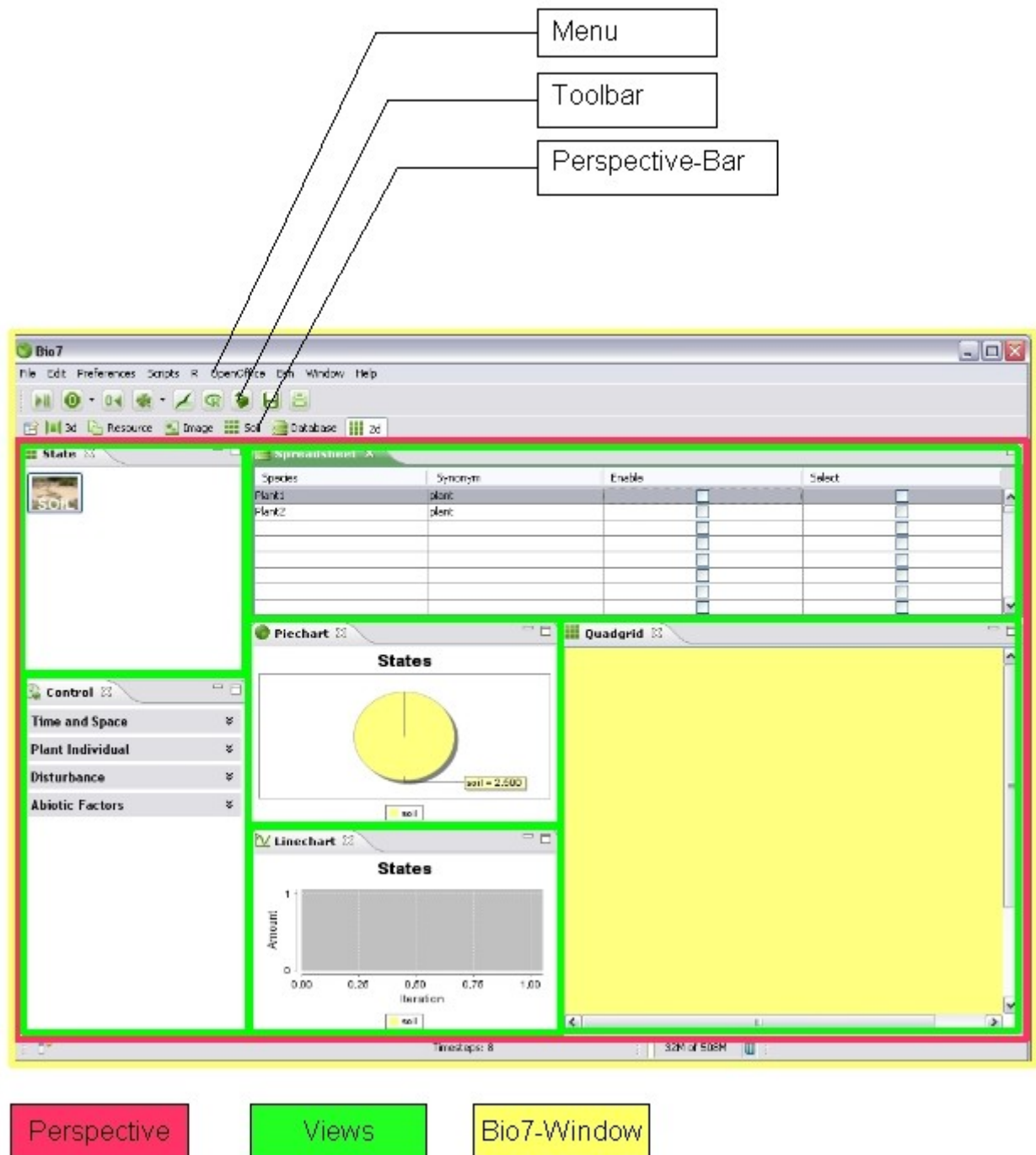
### File->Import->Existing Projects into Workspace



Select the archive file *Examples.zip* and the two projects *Bio7 Examples* and *Ecological Modelling*. Press Finish to import them into Bio7 (they will be imported to the workspace location).

For the first steps it is recommended to read through the section 14 (How to's) of this document to start with the examples of Bio7.

## 5 The Platform-Architecture

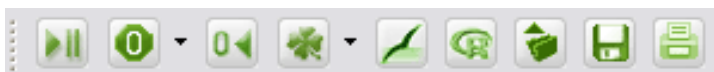


After the Bio7 application has been launched the typical Workbench window is displayed. A Bio7 window offers one or more perspectives. A perspective contains editors and views, such as the Navigator. In the first Workbench window that is opened, the Resource perspective is displayed as default.

In Eclipse terms: (A window contains a page, which contains an arrangement of views and editors whose layout is defined by a perspective).

## 5.1 The Toolbar

The toolbar offers several actions:



**1**   **2**   **3**   **4**   **5**   **6**   **7**   **8**   **9**

**1: Play/Pause->** Starts or stops a simulation run. This action will start the calculation thread of the application. The calculation thread will call the *ecomain* method in the embedded Ecoclass. When using the embedded compiler a new object of that class will be created. The visible part in a new created Java file is the classbody which will be compiled to use in the calculation thread. See the different snippets for examples.

**2: Reset->** Resets the discrete grid in the application. This action will reset all states in the Quadgrid(Hexgrid) arrays to the value 0 and to soil object (Plant array). The submenu item resets the plant array to a null reference

**3: Counter-Reset->** Resets the counter to null. This action will reset the time to the value 0 and will also update the Linechart.

**4: Random->** Distributes selected plants from the database in the discrete field. There are two random functions available in the menu of this item to randomize the field. If you press directly on the button only activated plants from the database and the soil state will be dispersed (objects and numeric values for use in a IBM model). If you select the submenu item all states will be dispersed in the numeric array except the objects in the Plant array (see section 6.2.1: The Quadgrid). If you want to disperse random cells (for example, for the Game of Life) the last option is sufficient.

**5: OpenOffice->** Will open a connection to OpenOffice. If you select this action Bio7 tries to connect to OpenOffice with the OpenOffice API (**A**pplication **P**rogramming **I**nterface).

A progress bar in the bottom right will show the progress. The path to OpenOffice will be fetched from the Preferences and can be adjusted there (Windows: If OpenOffice is installed, it will be automatically fetched from the Registry).

**6: R->** Will open a connection to R by means of a Server (Rserve). This action will connect Bio7 to the Rserve application which builds a connection to the statistical tool R. A progress bar in the bottom right will show the progress.

**7: Open->** Opens a simulation pattern with the **I**ndividual **B**ased **M**odels properties (IBM)file. If you open a pattern file for Bio7 all stored states and some adjustments will be loaded to the Quadgrid (Hexgrid). If objects have been stored all objects will be assigned to the array for use in a IBM model in the Quadgrid (Hexgrid). The stored field size, the states and the sequence of activation (and therefore their numeric values) will be recreated too.

States which are active at the moment of loading, but were not active as the pattern was stored will be inactivated in the spreadsheet. The condition for a successful recreation of a pattern is that the required state exists in the database (only for IBM models, not necessary when states out of the database context were stored).

**8: Save->** Saves simulation pattern with the IBM properties (IBM)file. As already described, field size, objects and states will be stored.

**9: Print->** Prints the content of an opened editor. If an editor in Bio7 has been opened this method will print the content.

#### **Annotation:**

When a file has been opened additional entries occur according to the opened editor (see Section 7).

## **5.2 The Menu**

The menubar of the Bio7 application offers several menus for the work with the different components of the application.

The **File** menu offers methods for import and export of projects and files which can be extended by BeanShell scripts.



The **Edit** menu offers several methods for the work with the different editors in Bio7.

The **Preferences** menu opens the preferences of the Bio7 application .

The **Scripts** menu (with several submenus) is extendable for self-defined scripts (BeanShell and ImageJ-Macros) in different categories (see Section 14.12).

The **R** menu offers methods for the interaction with R and to transfer data to the embedded R application.

The **OpenOffice** menu offers actions for acquiring and setting values from and to OpenOffice. Three further methods can transfer data to R or BeanShell which facilitates data acquisition through the options of OpenOffice application.

The menu **BeanShell** offers two methods to clear the "namespace" of the embedded interpreter and to reassign the default Bio7 imports.

In the **Window** menu perspectives and views can be opened. Additionally the editor area can be opened or closed.

The menu **Help** offers help for the application and the possibility to make updates and installations of new components for Bio7.

## 6 The Perspectives of Bio7

In the Perspective-Bar five different perspectives can be activated via the Perspective-Buttons. All the views or editors in the perspective can be closed, opened, detached or moved.

The perspectives in Bio7 can be extended by several other views which can be opened in the **Windows->Views** menu or with the action in the status bar at the bottom-left of the Program. Additionally self-defined perspectives can be created or deleted (See Section 6.7)

*Table 1: From the Eclipse Workbench User Guide (<http://help.eclipse.org/>)*

Perspectives provide combinations of views and editors that are suited to performing a particular set of tasks. For example, you would normally open the Debug perspective to debug a Java program.

To open a new perspective:

- 1.** Click the **Open Perspective** button on the shortcut bar on the left side of the Workbench window. (This provides the same function as the **Window > Open Perspective** menu on the menu bar.)
- 2.** To see a complete list of perspectives, select **Other...** from the drop-down menu.
- 3.** Select the perspective that you want to open.

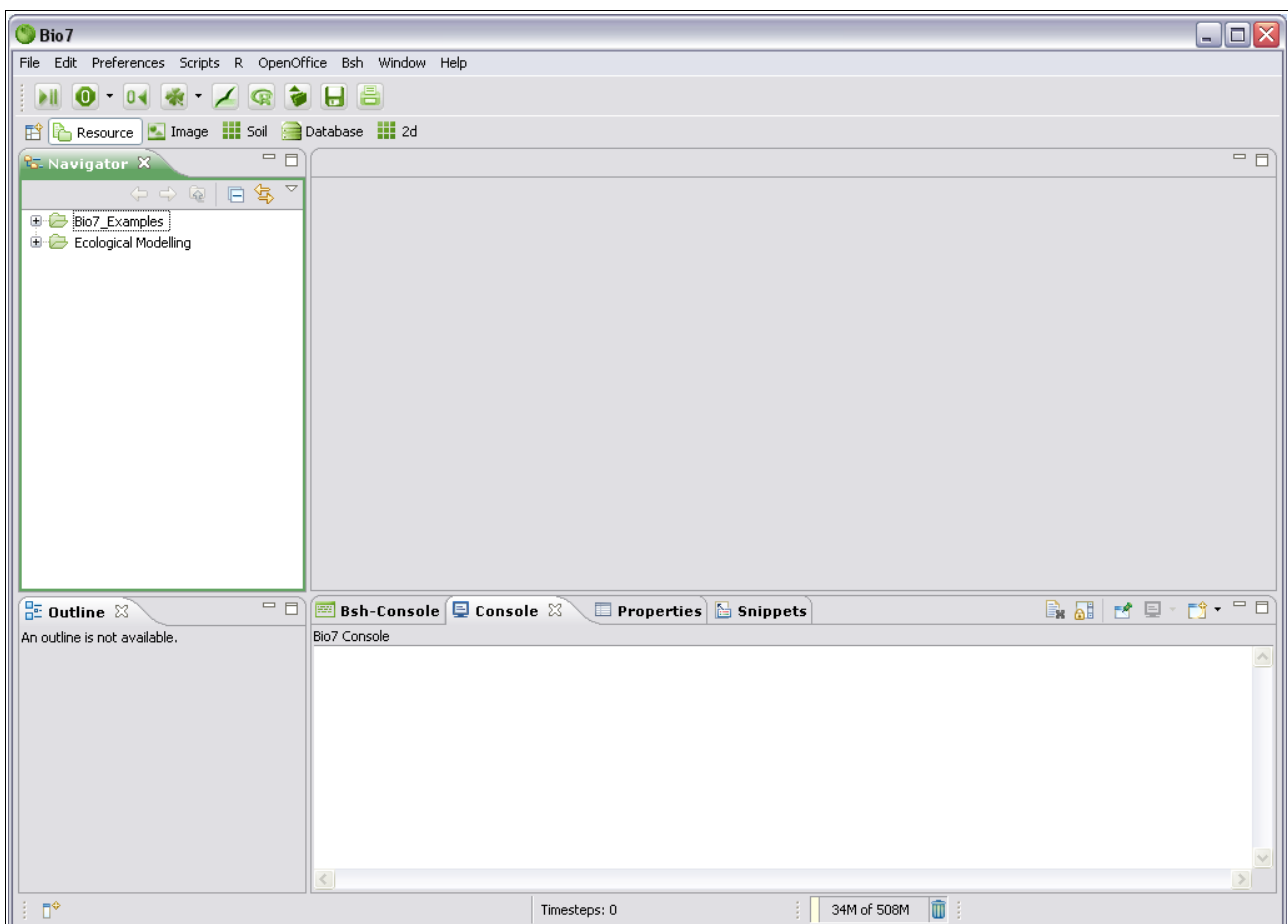
When the perspective opens, the title bar of the window it is in changes to display the name of the perspective. In addition, an icon is added to the shortcut bar, allowing you to quickly switch back to that perspective from other perspectives in the same window.

## 6.1 Resource perspective

After Bio7 has been started the perspective in front is the **Resource** perspective.

The parts of the Resource perspective are:

1. The **Navigator** to import, export, open and manage files.
2. The **Editor** area where files are opened.
3. The **Consoles** for Bio7 and BeanShell.
4. An **Outline** (active when the Flow editor has been opened)
5. The **Properties** view which shows and enables attributes of selected elements  
(Also important for the adjustment of the Flow editor properties).
6. The **Snippets** view in which general snippets for the different text based editors of Bio7 can be created.



*Illustration 1: The Resource perspective of Bio7*

### 6.1.1 The Navigator

In the Navigator view files can be managed and stored. The context menu offers several possibilities to import and export files for: BeanShell, R, Java and the Flow editor. All the files are managed by projects which can have folders and subfolders, etc. for managing a large amount of different files. These projects and their files can then be exported or imported as \*.zip files for data exchange. The Navigator of Bio7 allows you to create Java files, BeanShell files, RScript files, Flow files, textfiles and arbitrary files in their own editor environment with a help of an installed wizard. Some specialized integrated functions allow the compilation of Java files to BeanShell (after \*.java-files have been selected, multiple files allowed -> see Section 14.13) and the integration of Java libraries in BeanShell by right-click on a \*.jar file. Double-click on the file in the Navigator will open the file in the editor-area. According to the selected file the appropriate editor for the file will be opened.

See Section 7 (The Editors of Bio7) for more information.

### 6.1.2 The Editor Area

The editor area is the area where all files are opened and edited. The context menu varies depending on which file is opened. The R, BeanShell and Java editors when opened offer typical text functionalities like: Copy, Paste, Undo, Redo, Cut, Find/Replace, Print, etc. in the context menu and in the edit menu (menu-bar) of Bio7. When a file has been opened editor specific buttons will be added to the toolbar which execute an editor action according to the

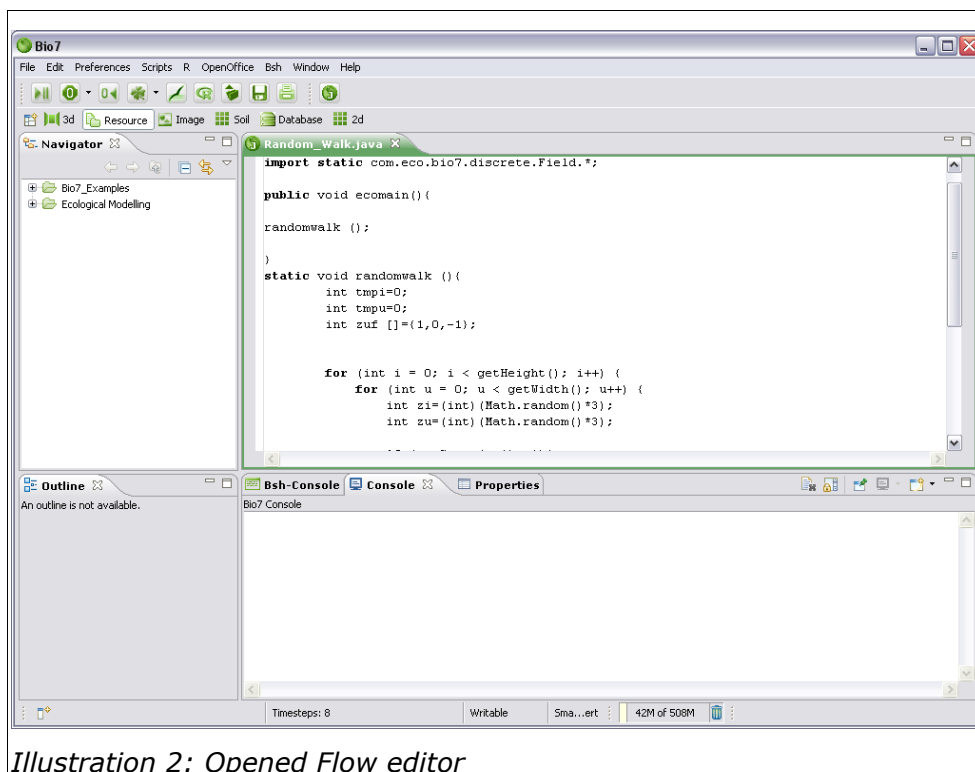


Illustration 2: Opened Flow editor

opened file (Compilation for Java, Interpreting for BeanShell or R and execution of a Flow!). See Section 7 (The Editors of Bio7) for more information.

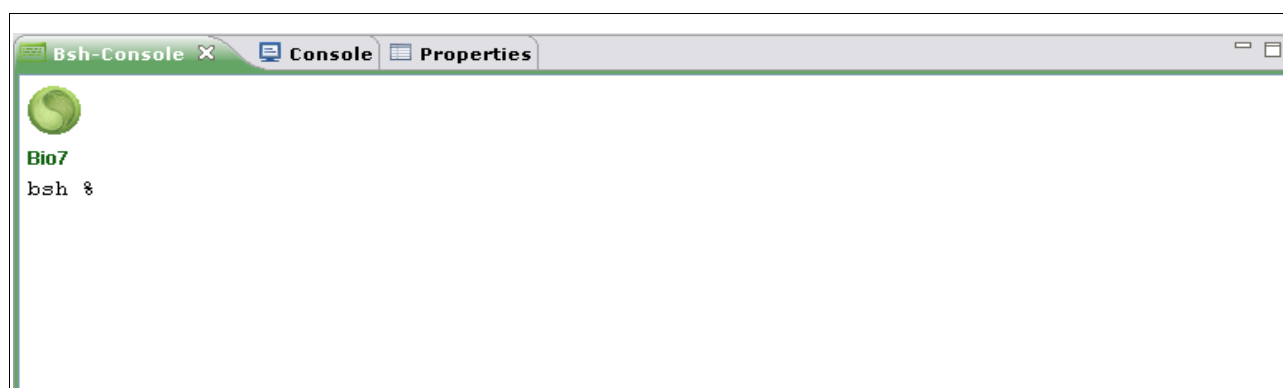
### 6.1.3 The Consoles of Bio7

The consoles of Bio7 are important for the output of information gained from the embedded Java compiler or from the different script languages. The **Bsh-Console** (Beanshell) can interactively communicate with the Bio7 application or simply interpret different instructions in Java or the optional scripting language.

The **Console** is responsible for the:

1. Output messages for the interpreter
2. Output messages of the embedded Java compiler (compiler warnings, infos, etc.)
3. Output of the R interpreter (Rserve warnings, etc.) and evaluated expressions in R.

For example a call like `System.out.println("Hello World");` will be printed in the **Console** if executed in the BeanShell-Console (or Java) . Otherwise a call like `print("Hello World")` will be written to the BeanShell console if executed in the same BeanShell console. Evaluations of R expression will be sent to the Console.



*Illustration 3: The BeanShell console*

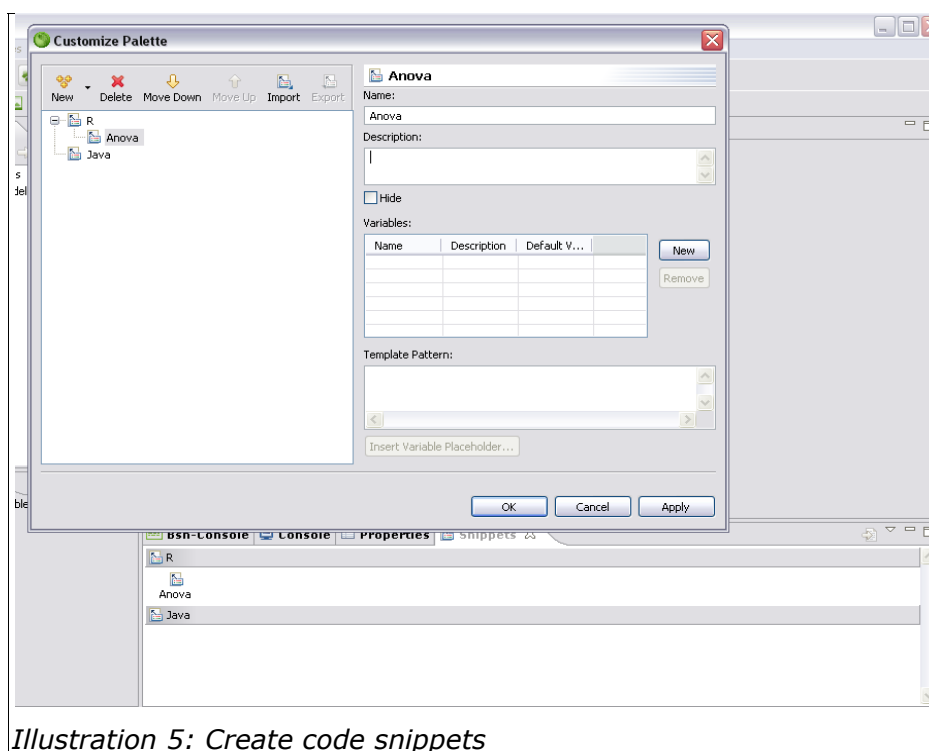


*Illustration 4: The Platform console*

The default platform console (Illustration 4) will display results from different editors and also display errors which occur when compiling or interpreting a file. There are also several actions in this console available to clear displayed text or copy the text from the console to the clipboard (available in the context menu).

### 6.1.4 The Snippets view

In the Snippets view code templates can be created and stored for the different editors in Bio7. By default every text based editor of Bio7 (Java, BeanShell ,R) has an integrated template editor (see section 7).



*Illustration 5: Create code snippets*

To create snippets inside this specialized view, category's have to be created inside a template editor. This editor can be opened with a right-click in the Snippets view. Copied text from any text based editor of Bio7 can be pasted directly as a snippet after a category has been created inside the view. Also more complex templates with variables and a detailed description can be created. This is extremely useful if you want to create custom formulas or statistics. A created template will be transferred to an active editor with a double-click of the mouse device. Optional the different categories can also be folded and customized for a better overview.

## 6.2 The 2d perspective

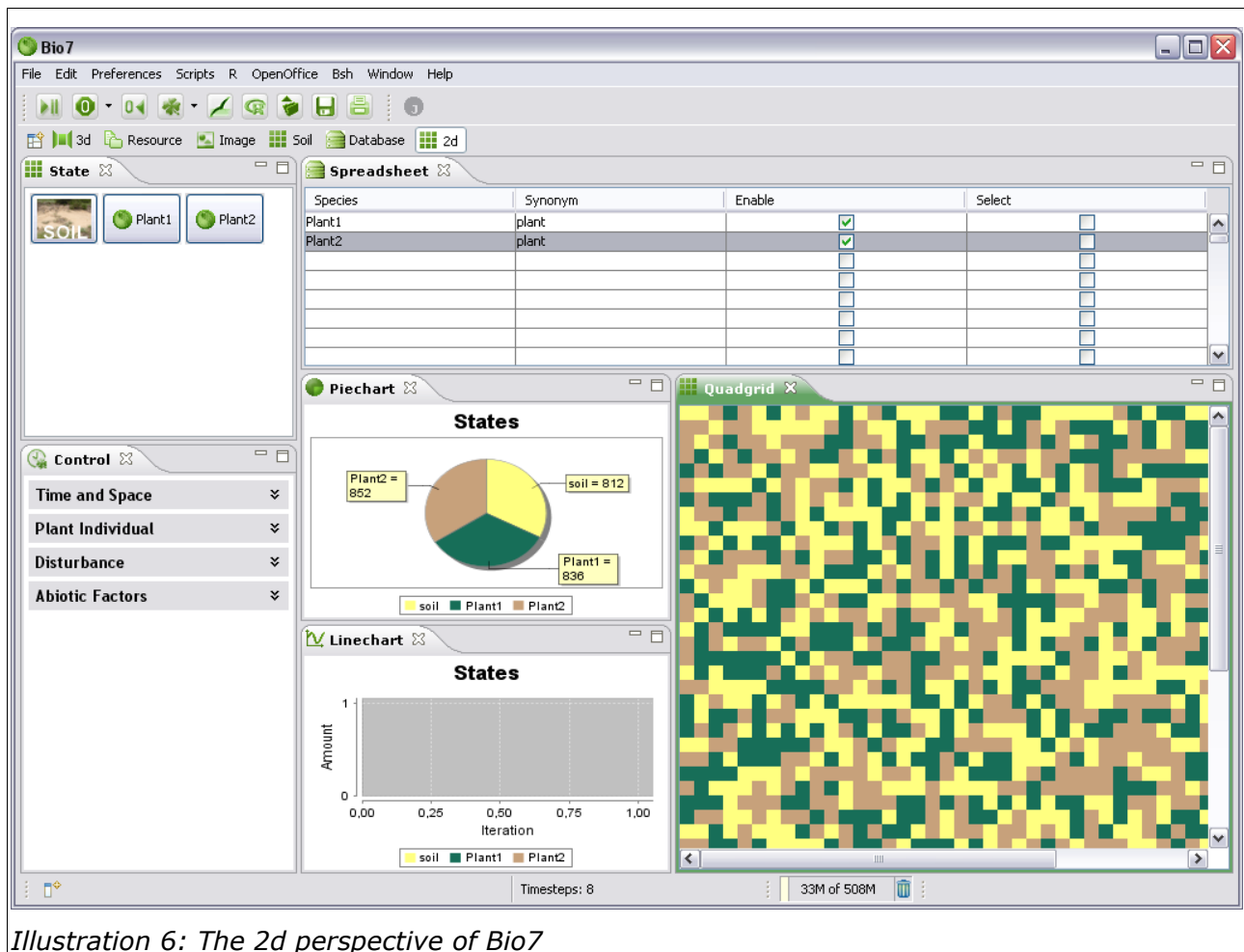
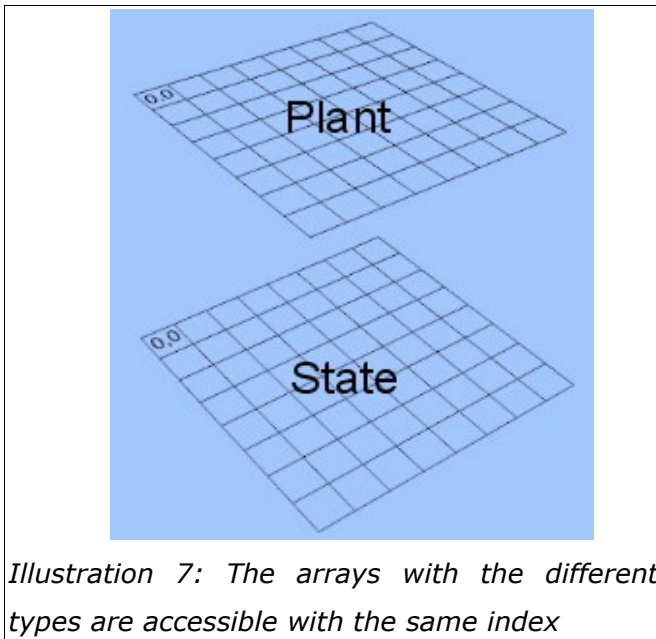


Illustration 6: The 2d perspective of Bio7

The 2d-perspective offers several views for displaying and adjusting simulations of spatial discrete models and spatial discrete individual based models of plants.

### 6.2.1 The Quadgrid

The Quadgrid panel (like the by default hidden Hexgrid panel) visualizes a simulation field for discrete simulation models. By default only "soil" as state with value 0 is dispersed in the Quadgrid (respectively Hexgrid) field. The panel itself is a visualization of states stored in an two dimensional grid array. In the grid array (called **xystate**) numeric states of type integer are stored for simulation purposes. Furthermore in a second two dimensional grid array, (**plant**) of **type Plant**, plant individuals can be stored as an object. The two array approach has been chosen because of the advantage to easily choose between the creation of classical cellular automata or grid based models and grid based individual models. By default there are also two further temporary arrays available (**xytempstate**, **tempplant**) with the appropriate type for an intermediate store of states when calculating patterns for the next timestep ( $t+1$ ) as it is usual in grid based modelling.



*Illustration 7: The arrays with the different types are accessible with the same index*

*Table 2: Code snippet*

```

/*A simple example to show how to produce and set random cells without setting any individual plant information
from the database. States: soil=0, plant=1*/
int rand;
public void ecomain() {
    for (int y = 0; y < Field.getHeight(); y++) {
        for (int x = 0; x < Field.getWidth(); x++) {
            rand = (int) (Math.random() * 2);
            Field.setState(x,y,rand);
            // We also fill the temp array !
            Field.setTempState(x,y,rand);
        }
    }
}

```

The arrays can be accessed by means of a small API to facilitate the development of such models. In the panel it is possible to zoom in or zoom out with the scroll wheel of the mouse. The state of cells can be adjusted by clicking and dragging on them. The value to which a cell will be changed is dependant on the previously changed button in the Plants view (see Section 6.2.5). By default this value is the value of the soil. By right click on the Quadgrid panel a context menu appears with the possibility to select plant individuals in the Quadgrid panel (Keys: Alt+S). When a plant has been selected the individual properties will be displayed in the Control view under the tab "Plant Individual" and will be displayed in the status bar too. In the Control view the individual attributes of the plant can be adjusted, etc. (see Section 6.2.4.2).



Table 3: Quadgrid (Hexgrid)

Device	Event	Function
Mouse	Dragged	Set numeric values or plants (last selected Plant or State !)
Mouse	Scrolled	Zoom in, out (Panel)
Menu->Select Plant->Mouse	Pressed	Select a Plant
Mouse	Pressed	Set numeric values or plants

## 6.2.2 The Hexgrid

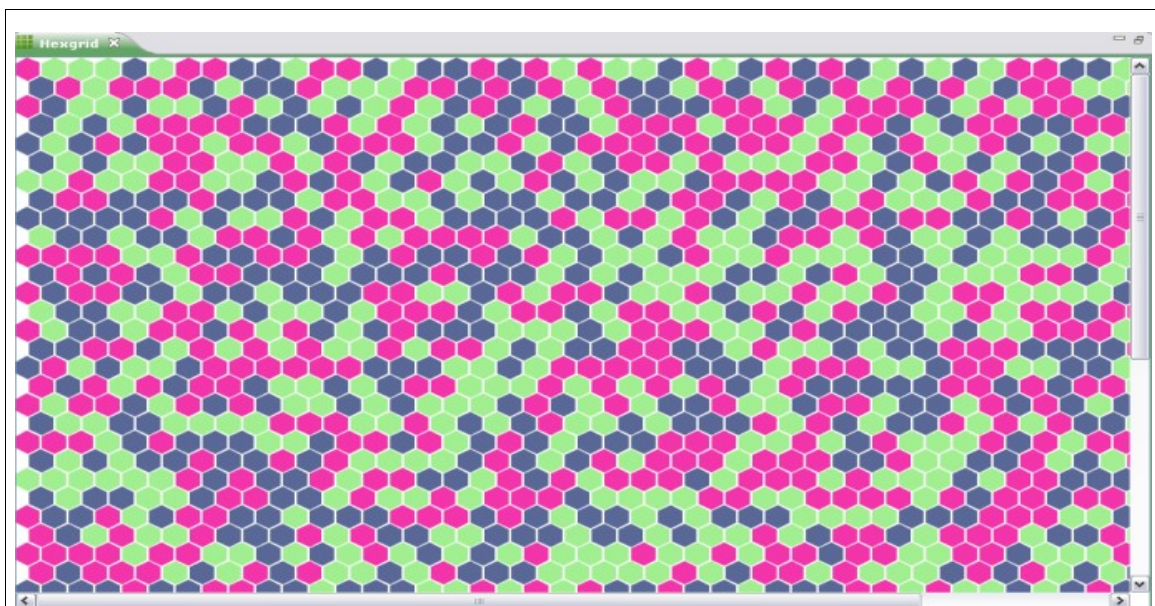


Illustration 8: Hexgrid in Bio7

It has sometimes advantages to create hexagonal designs in experimental setups which will reproduce a binned pattern more naturally instead of the Quadgrid. A central cell in a Hexgrid is surrounded by six neighbours. All have the same euclidean distance from the centre of the central cell. Bio7 offers a hexagonal grid which is by default hidden, but can easily be activated with **Window->Show View->Other->Bio7-Panels->Hexgrid**

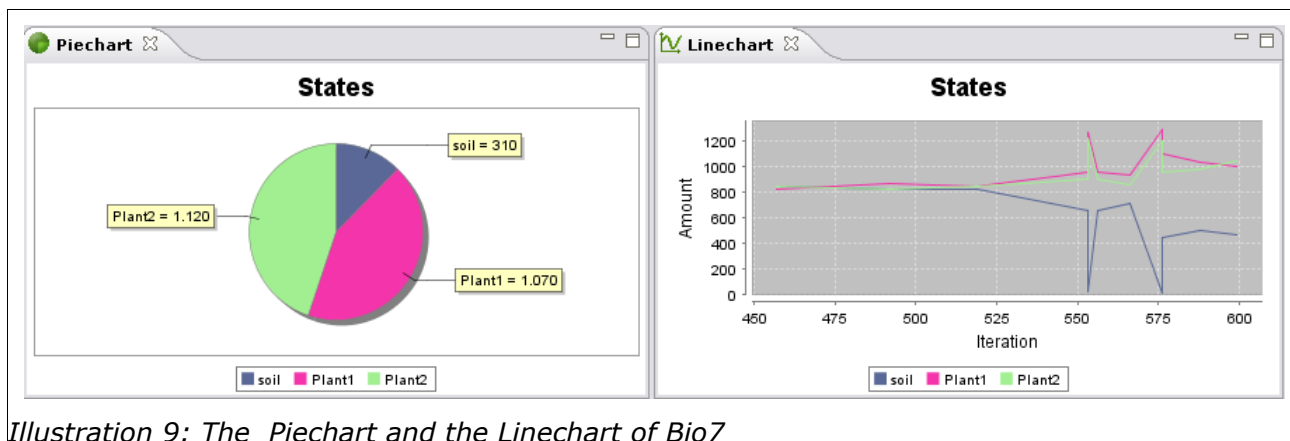
### 6.2.2.1 Storage of patterns and IBM properties

To store the pattern of the cell grid and the IBM properties of each cell the **Save pattern** action stores all properties from every plant in the grid. Also the grid size and the selected species in the spreadsheet will be stored when selecting this action (sequence will be preserved !). The file will be stored as a database file (\*.yap) which can be opened again with the **Load pattern** action. When this action is selected the state of all stored cell with there IBM properties are restored and the stored plant species will automatically be selected in the

spreadsheet view for simulation. Furthermore the field size will be adjusted according to the stored value.

### 6.2.3 The Linechart and the Piechart

The Linechart and the Piechart views serve as a **monitor** to watch population size and count the different states in a simulation run. Regardless how fast a simulation runs the charts get **updated every second**. By right-click on the charts several options are available. For example to store an image of the chart or to adjust some chart parameters. The Piechart visualizes the count of the activated states inside the Quadgrid (Hexgrid) every second. (Changes can easily be viewed by setting states with the mouse inside the Quadgrid). The Linechart displays the amount of states in dependence of the simulation steps. When a simulation is started different coloured lines will visualize the amount of states over the time. The colours of the charts will be automatically adjusted when changing colours for a state in the Plants view.



## 6.2.4 The Control

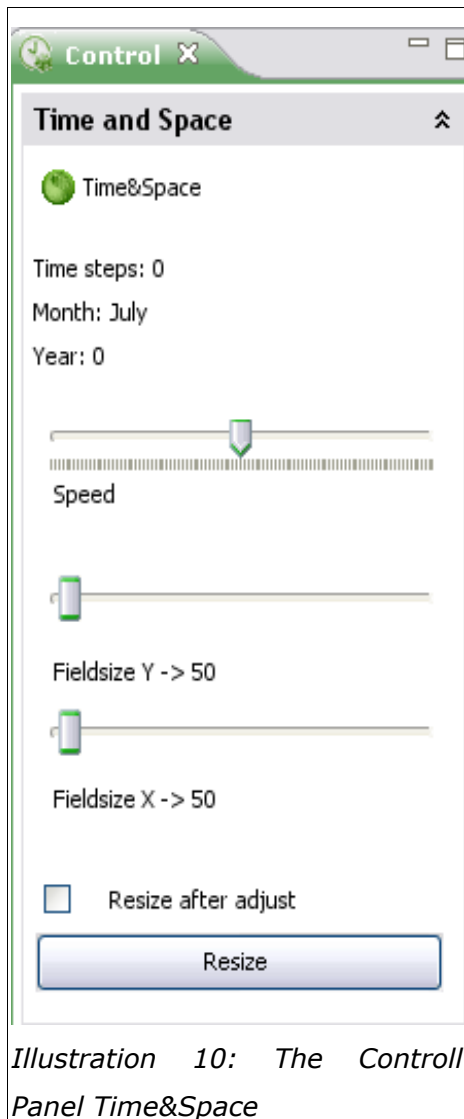


Illustration 10: The Control Panel Time&Space

In the ExpandBar of the Control view several panels are accessible by selecting the ExpandBar items (Time&Space, Plant Individual, Disturbance, Abiotic Factors).

### 6.2.4.1 Time and Space

The topmost item is called **Time&Space**. When this item is enrolled a panel appears showing information about the time steps a simulation has proceeded. Additionally for "plant interactions" every time step is calculated as a month and 12 time steps are summarized as a year which will be shown too. Furthermore several sliders for the adjustment of the parameters calculation speed (time resolution) and field size (capacity) are available. Additionally a checkbox and a button are displayed which enable the resize event after an adjustment. The first slider (**Speed**) is responsible for the calculation speed of the calculation thread. It can be set to values from 10-1000 units. The visible sliders below (**Fieldsize**) can be moved to values from 1 to 1000. They are responsible to adjust the field size of the different grids in Bio7 (Quadgrid, Hexgrid, Nitrate...) So the maximum field size which can be adjusted in Bio7 is a field size of

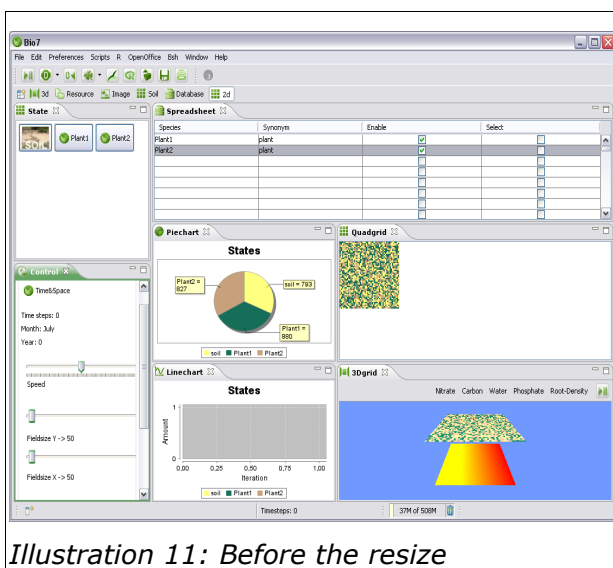


Illustration 11: Before the resize

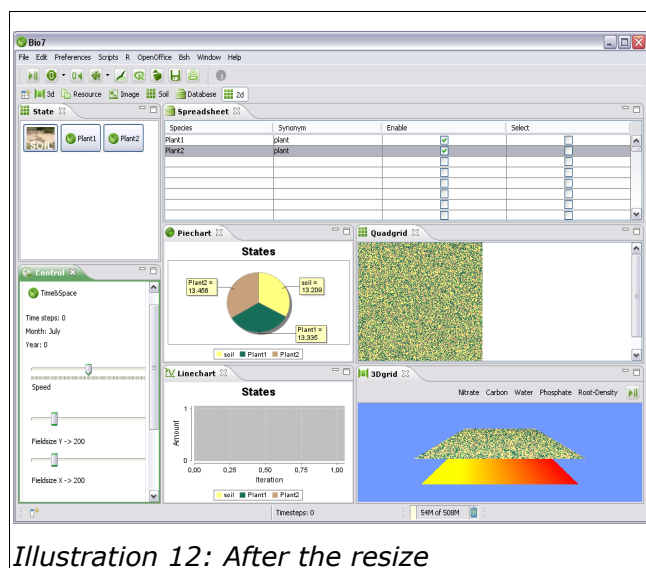
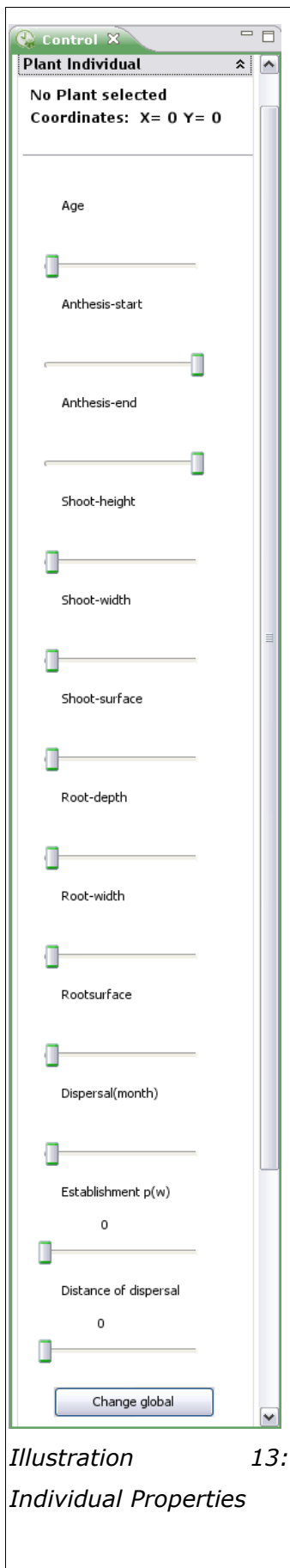


Illustration 12: After the resize

1000\*1000. When moving the sliders for the field size several views and arrays will be updated according to the new values. The following panel with their underlying arrays will be



updated when a resize event takes place: The Quadgrid panel, the Hexgrid panel, the soil panels and the visualization of these in 3d the 3d panel. After resizing the new indexes are instantly available for plants and numeric values (e.g programmatically). When the checkbox **Resize after adjust** is activated the resizing takes place after the button **Resize** has been pushed. This is recommended when the field size is >400 because the resizing then is done in one call.

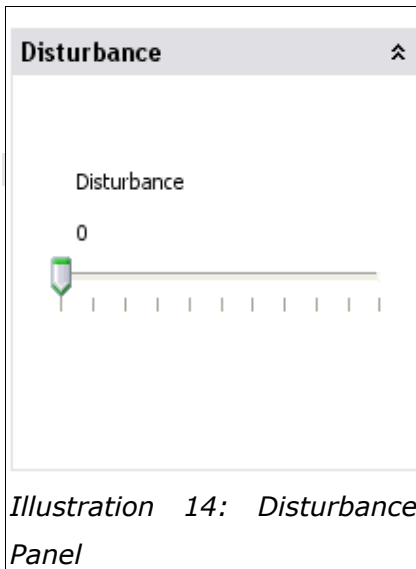
Info: The speed of a resize event is also dependant on the amount of grid panels which are currently visible in the Bio7 window.

#### 6.2.4.2 Plant Individual

If the item **Plant Individual** is activated a panel will be opened which embeds several sliders for adjusting of individual parameters for the selected Quadgrid cells (Hexgrid). Additionally information will be displayed about the position of a cell in the grid and the plant species (if available) which has been selected. This information will also be shown in the status bar of the Bio7 application. When a cell has been selected with an existing plant object in the underlying (plant) object array (no null reference !) the individual properties will be displayed by means of the sliders. All sliders will be updated with the values received from the object. By changing the values of the sliders the values of the object at the selected index (y,x) will be updated automatically. So every individual plant object in the grid array can be updated separately from the others. It is also possible to update all other plants which represent the same species in the grid with the action **Change global**. This is analogue to nature where we have individuals of one species which often differ in some characteristics. It is a generalization when these properties are responsible for every individual of the same kind (by pressing the button). If a cell is selected which has no plant object in the array (plant, planttemp) of type "Plant" the message "No plant selected" occurs. For example when only a cellular automata is needed with custom states there is no need to use the plant array with the templates for the database. In that case only the array of the type int (state, tempstate) for the states is needed.

### 6.2.4.3 Disturbance

Often ecosystems are marked by random disturbances by means of biotic or abiotic factors.



*Illustration 14: Disturbance Panel*

The item **Disturbance** contains a slider for the adjustment of an embedded disturbance function in Bio7. This function can be used in a custom simulation by calling :

**Environment.randomDisturbance(y, x);**

When this function is called, cells which are affected by the function (A cell has a change to be affected with a probability of the slider value [0-1000]) will receive the numeric value 0 (state, tempstate) which is by default the value of the soil. Additionally the object array for the plants will receive a null reference. An example for this special function is the following piece of code (Table 4).

*Table 4: Code snippet*

```
public void ecomain() {  
    for (int y = 0; y < Field.getHeight(); y++) {  
  
        for (int x = 0; x < Field.getWidth(); x++) {  
  
            Environment.randomDisturbance(x, y,0);  
            //Plant s are deleted by chance according to the disturbance value !  
        }  
    }  
}
```

## 6.2.4.4 Abiotic Factors

Behind the last item in the Expandbar a panel is hidden, which offers several textfields for custom formulas of abiotic factors. Self defined formulas can be compiled and directly be used in a custom simulation. These formulas can be made dynamic by adding two special variables for simulation purposes. The first variable is the  $\$t$  variable. This variable can be used to

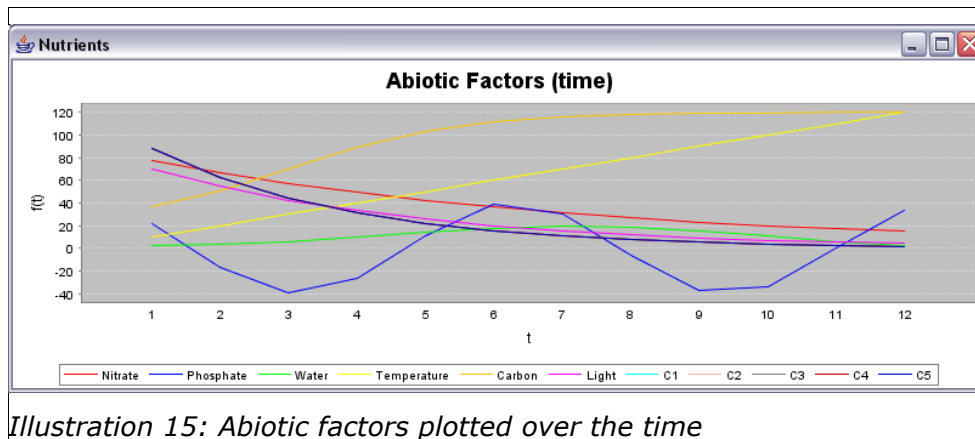


Illustration 15: Abiotic factors plotted over the time

simulate a time series function over the year. For example if you have rain precipitation data from different years and want to use an average function for your simulation. Then you simply have to add the  $\$t$  variable in your formula which represent a modulo function to repeat the values when a special self-defined value has been reached. This self defined value can be adjusted in the **Repeat after** textfield. By default this value is adjusted to 12 for a year. So by default after 12 time steps (calculation begins with 0!) the  $\$t$  variable will be set to 0 again

Illustration 16: Abiotic factors

and the formula will be reset for the year to start again. Optional a second variable is available. This is the counter variable  $\$t$  when a progressing variable is needed. So in that case the simulation steps can be used as an argument in the self defined formula. The formulas must be expressed in Java which offers a lot of mathematical expressions by default (it's not necessary to set a ';' char at the end !). External packages can also be used for custom expression. Some library's offer a huge amount of well proofed functions. Some of these library's are embedded in Bio7 (Colt, JScience, Apache commons-math) by default. When a new formula has been expressed in the textfields **f(t)**... the changes occur when equations have been compiled. This can be done by selecting the **Compile** button. To plot the compiled functions as a preview the function **Preview** plots the different equations over the time according to the adjusted **Repeat after** value. To use the formulas and embed it in a simulation see the following Code Snippet which calls the compiled ecomain method to recalculate the equation with the dynamic variable  $\$t$  or

*Time.getCounter():*

*Table 5: Code snippet*

```
public void ecomain() {
    AbioticFactors f = Compiled.getAbioticFactors();
    if (f != null) {
        // Call the main method!
        f.ecomain();
        // Get the values!
        double light = f.light;
        double nitrate = f.nitrate;
        double phosphate = f.phosphate;
        double water = f.water;
        double temp = f.temp;
        double carbon = f.carbon;
        System.out.println(light+" "+nitrate+" "+phosphate+
            " "+water+" "+temp+" "+carbon);

    } else {
        System.out.println("Formulas are not compiled !!!");
    }
}
```

## 6.2.5 Plants(States)

The **Plants** view is responsible to display all activated states in a simulation or a spatial analysis in Bio7 and the setup of the colours of the different activated states. Furthermore the sequence of activation determines the numeric value of the state inside the Quadgrid and the Points panel (starts with 0 !). By default one state is enabled in the Plant view. This is the state **soil** (0) which is useful for simulation of plants this application initially was intended to deal with. If plants are enabled for simulation in Bio7, additional buttons will be labelled with the name of the plants and set into the panel in the sequence they have been enabled. If you want an image to appear instead of the name on a button, an appropriate image with the name of the species in the database (e.g.: Hieracium\_pilosella.bmp) can be dropped to the pics folder of Bio7 (...\\plugins\\com.eco.bio7\_1.0.0\\bin\\pics). If a Button gets pressed a colour dialog pops up for changing the colour of the enabled state in the simulation (automatically colours for the states in the Quadgrid and in the charts will be updated !). The default states can be changed programmatically and it is also possible to set states independent of the database (See section 14.11). The colour changes made in this view will effect the Quadgrid, Hexgrid, Points panel, Linechart, Piechart and the 3d panel.

### 6.2.6 Spreadsheet

In the top-right of the perspective the **Spreadsheet** offers a visualization of stored plants in a object-oriented database. The species name and the synonym are displayed by default. Also two columns with check boxes are displayed. By enabling a check box in the column **Enable** Plant species from the database will be enabled for simulation.

The following changes occur after activation:

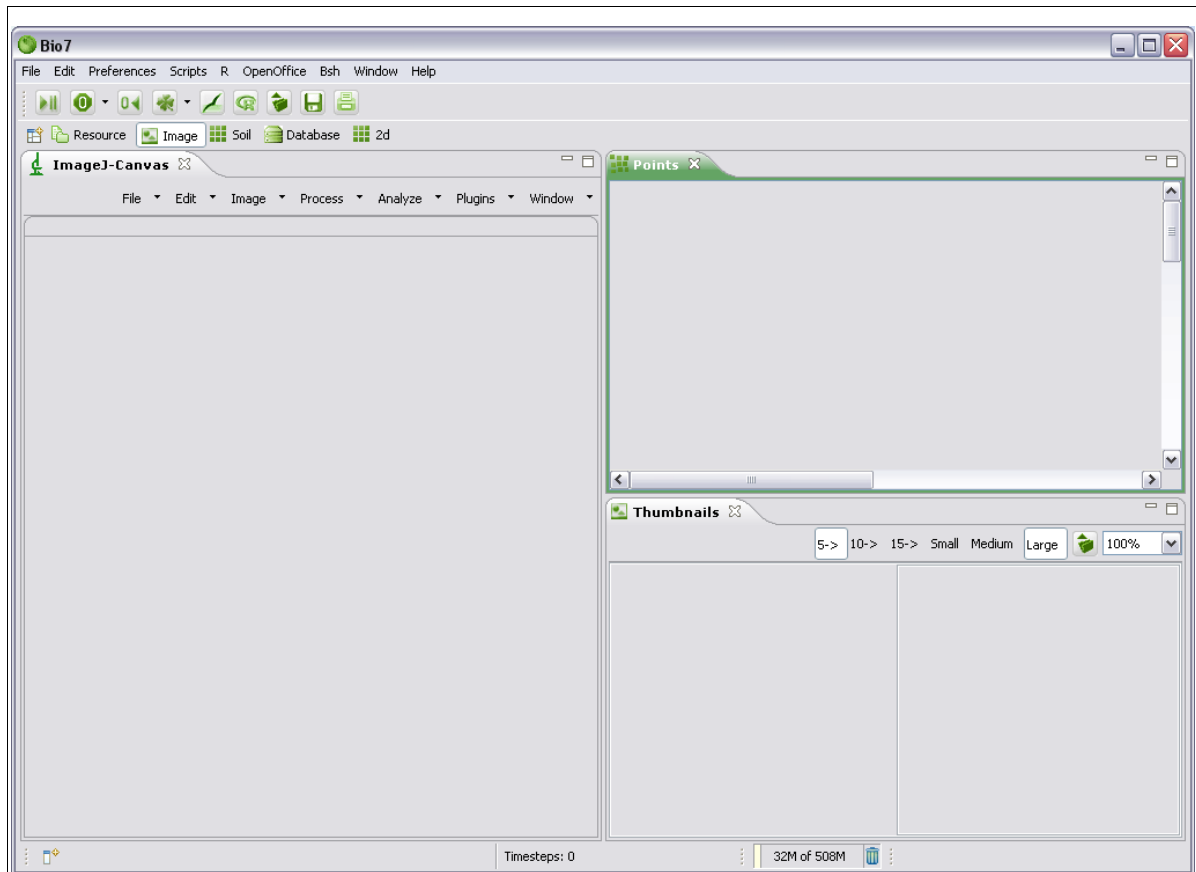
1. Automatically a random function disperses the activated species from the database (template) in the grid.
2. The **Linechart** and the **Piechart** will be updated with the activated plants. The Piechart will display the numbers of the different plants in the Quadgrid matrix.
3. In the top-left panel (**Plants**) dynamically buttons occur to
  - (a) show the activated plants
  - (b) change the colour of the plants after activation.

See Section 6.5 for the database options of the spreadsheet

### 6.3 Image Perspective

At startup the image perspective shows three different views for several tasks in the area of image analysis, point-pattern analysis and simulation of 2-dimensional spatially continuous models. In the **ImageJ-Canvas** view a complete version of the well known image analysis tool **ImageJ** has been integrated in the Bio7 application for profound image-analysis and measurements of ecological parameters. Because ImageJ offers also complete API (**A**pplication **P**rogramming **I**nterface) the development of customized image methods are possible. Directly connected to this view a scalable **Points panel** for adjustments and visualization of point pattern has been integrated. This panel is also programmatically available by means of a small API which facilitates the construction of 2d continuous (also IBM-Models) simulation models. An easy to use thumbnail browser (**Thumbnails**) offers the possibility to open directories and the included images as thumbs which than can be opened directly in ImageJ.



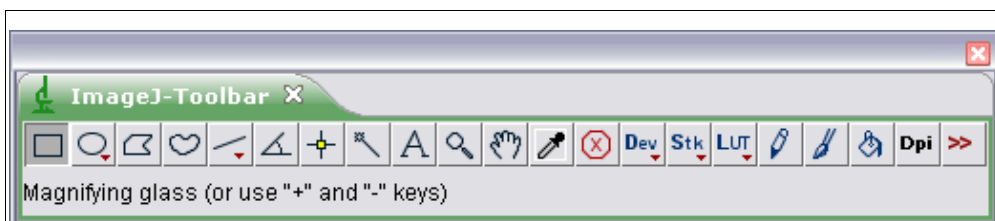


*Illustration 17: The image perspective*

### 6.3.1 ImageJ-Canvas

For a complete description of the program ImageJ follow the link <http://rsb.info.nih.gov/ij/>. The differences between the regular version of ImageJ and the integrated version in Bio7 are described here:

1. The ImageJ Canvas is embedded in Bio7.
2. The **Toolbar** of ImageJ (Illustration 18) is integrated in a view (when called from the menu **Window-> ImageJ-Toolbar** as a detached view).



*Illustration 18: ImageJ tools as a detached view(Simply drag and drop images on the view to open them!).*

- 3.** The menu of the integrated application has been reprogrammed as a view menu which is similar to the original ImageJ menu.
- 4.** All images are opened in tabs and can be edited separately.
- 5.** Since Bio7 1.1 external plugins are supported. It depends on the plugins if they work correctly with the embedded ImageJ (uses Swing!).
- 6.** Macros can extend the main menu of Bio7 and can be created like in the regular version.
- 7.** Two OpenSource plugins for ImageJ have been added to the view menu of Bio7.
  - (a)** The first is a plugin for multiple image analysis, where multiple images can be opened and edited by macros at once (<http://ciar.rcm.upr.edu/projects/imageJ>).
  - (b)** The second plugin FracLac 2.5 (<http://rsb.info.nih.gov/ij/plugins/frac-lac.html>) is a specialized plugin for the analysis of fractals in digital images.

For more information about these plugins visit the different websites (for manuals which are available).
- 8.** For Windows a small application has been added for convenience which embeds an Avi to Swf Converter (<http://www.swftools.org/>). It enables the conversion of created avis (by means of a stack) to swf files which then can be used to visualize presentations from Bio7 (see Section 14.9).
- 9.** The ImageJ API is accessible by means of the embedded compiler or with BeanShell scripts.
- 10.** Bio7 offers some specialized methods for transferring particle values to the Points panel or to R (all particle measurements can be transferred from ImageJ ->see Section 14.4.2).

As mentioned before functions of ImageJ can be directly called from the Java compiler or from BeanShell. A lot of available examples from the ImageJ website can be executed directly within the different editors in Bio7 by compiling or interpreting them. All standard functions of ImageJ

are integrated.

**Macros** are fully supported and can also be added dynamically to the main menu of Bio7 for efficient image analysis. In the menu of the ImageJ view some additional menu points (->**Windows** )have been added for:

1. Opening the toolbar of ImageJ (**ImageJ-Toolbar** as a detached view)
2. Opening a control panel for the Points panel with functions for the exchange and adjustment of images between ImageJ and the Points panel.

### 6.3.2 Points panel

In the Points panel points can be set or deleted with the mouse by left- or right-click in the selected coordinates. Additionally points can be dragged to any locations inside the boundary of the adjusted size inside the "**Area of Analysis**" of the Points panel marked by a green rectangle. The coordinates (pixel location) are displayed in the status bar of the Bio7 application.

The panel itself can be resized to a virtual size of  $10^6 \times 10^6$  units (pixel) inside a scrolling panel. (e.g. you can set points in a virtual area of  $10^6 \times 10^6$  cm (as unit !)) which means that you can adjust points in an area of 10km with cm precision). Also zooming is supported when a big area is required. If the panel is zoomed or scaled down the points automatically will get translated, resized and stay selectable. It is also possible to add the quads from the Quadgrid panel to the Points panel and to select this quads by setting a point inside its area (see Section 14.5).

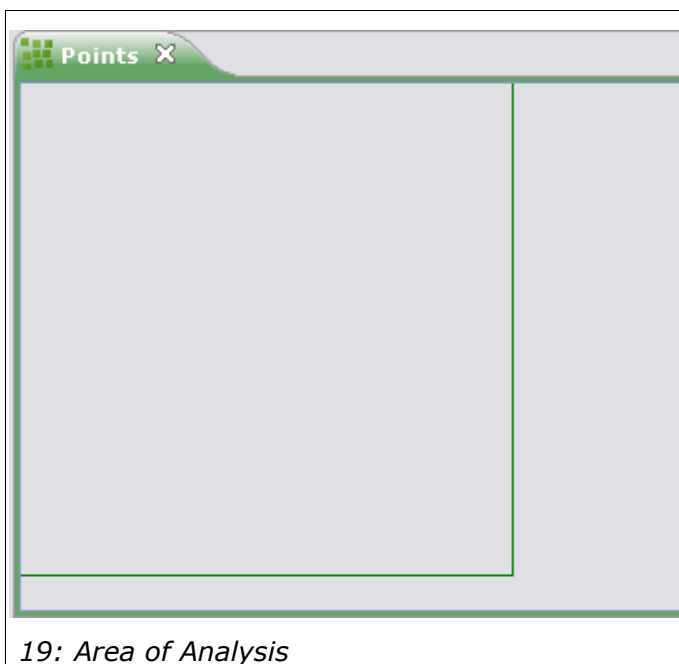
*Table 6: Points panel actions*

Device	Event	Function
Mouse	Left-click (Double-click)	Set point
Mouse	Right-click	Delete point
Mouse	Left-click+Drag	Move point
Mouse	Right-click (Double-click)	Move ImageJ viewport with resized image in Points panel!

To archive this a special **toolbar** is available in the Window-menu (**Window->Bio7-Toolbar**)of the ImageJ view. If selected, a view gets detached which offers several possibilities to use the Points panel in interaction with ImageJ.

The following methods (Illustration 20) are available:

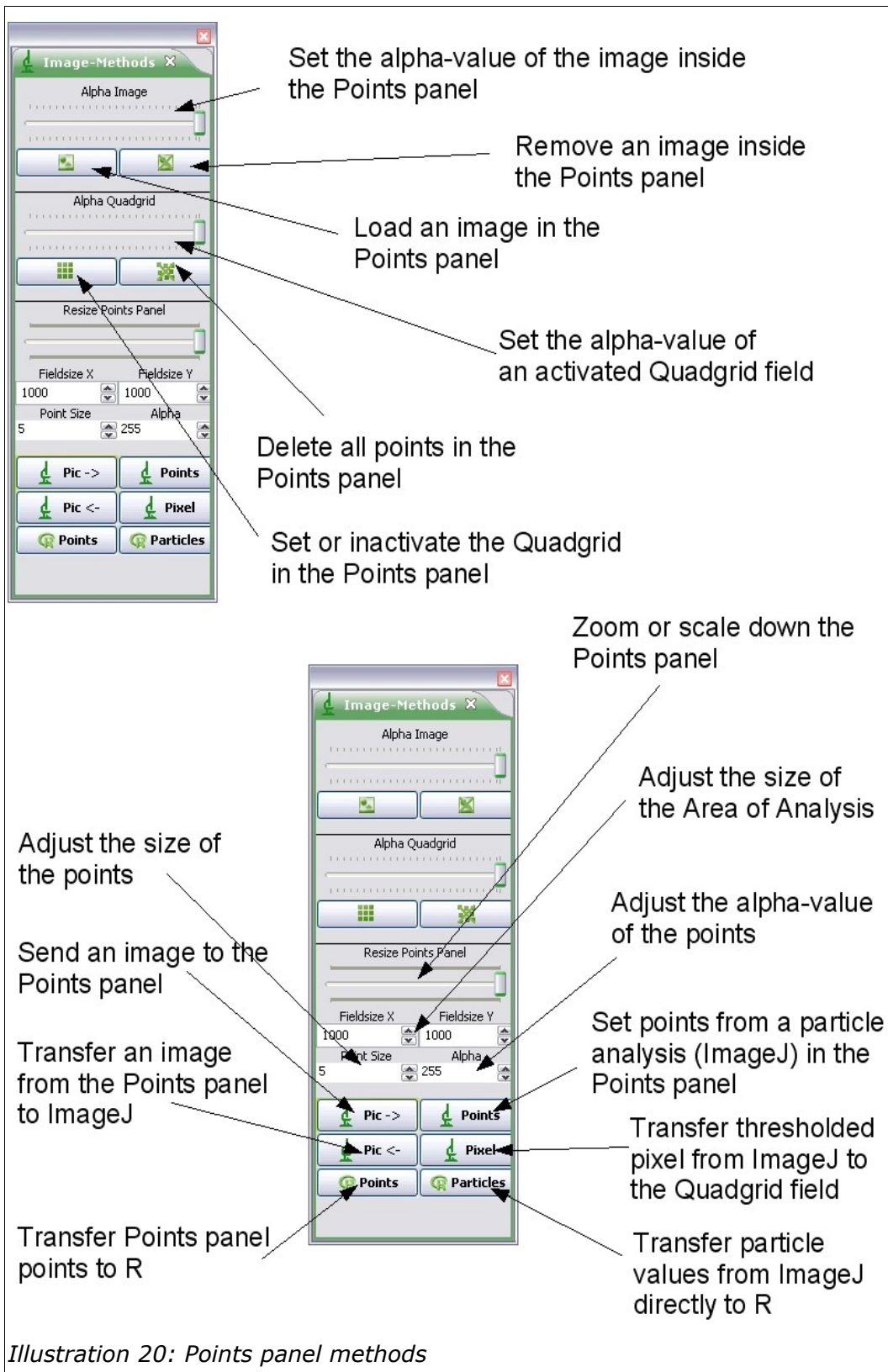
1. With the **Alpha Pic** slider the alpha value of a loaded image in the Points panel can be adjusted.
2. The two Buttons below open an image in the Points panel or remove it. When an image is loaded (From ImageJ or from Bio7) the size of the "Area of Analysis" will be adjusted automatically to the image size and the new size will be displayed in the spinners (Fieldsize). The "**Area of Analysis**" is displayed by the green rectangle in the Points panel.
3. The following slider can adjust the alpha value of the Quadgrid field which can be added or removed with the below left button.
4. With the right button all points that have been activated in the Points panel will be deleted.
5. With the left button the Quadgrid can be activated for an overlay in the Points panel.  
When the Quadgrid is activated the "Area of Analysis" will be adjusted to the size of the Quadgrid.

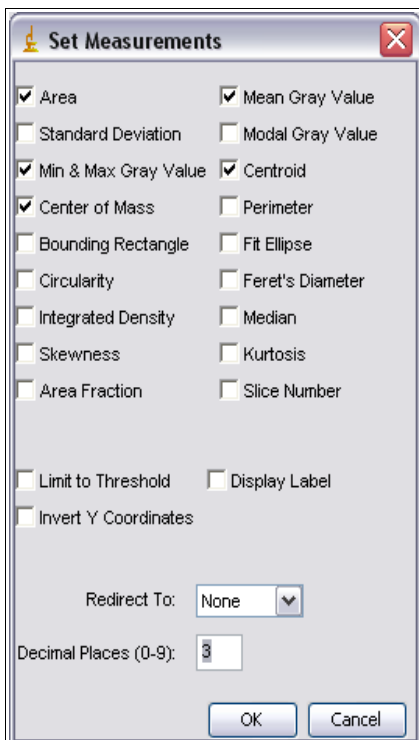


6. The next slider has the function to zoom out of the panel for an overview. When zooming has been activated the points still stay selectable or draggable because the coordinates are automatically translated. The scrollbar will be adjusted to the size of the "Area of Analysis".

- 7.** The size of the "Area of Analysis" can be adjusted with the two spinners (**Fieldsize X**, **Fieldsize Y**) which can be set from  $1-10^6$  units. When the size of the area will be changed automatically the scrollbar will be fitted to the new size of the green rectangle.
- 8.** The size (**Particle Size**) and the alpha value (**Alpha**) of the points (or particles) can be changed with the spinners below the size spinners.
- 9.** With the **Pic->** button it is possible to transfer images that have been loaded with ImageJ directly to the Points panel to use as a matrix to select point patterns, etc. The size of the "Area of Analysis" will be changed according to the size of the image.
- 10.** The below **Pic<-** button sends an rgb image back to the ImageJ canvas. It is important to know that the transferred image to ImageJ depends on the "Area of Analysis" and the zoom factor of the Points panel. If the Points panel is not zoomed and the "Area of Analysis" is  $1000 \times 1000$  this will result in a  $1000 \times 1000$  image in ImageJ. If the Points panel was scaled with a  $1000 \times 1000$  "Area of Analysis" the resulted transferred picture to ImageJ would be a scaled image ( $<1000 \times 1000$ ) which shows the same area.
- 11.** The two **Points** button are responsible for transferring Particle Analysis data from ImageJ to the Points panel or directly to R.
- a)** The first button selects a method which will transfer particle measurements to points in the Points panel. If activated in ImageJ (as by default!) the rounded centroid coordinates (values  $\geq 1.5$  result in 2; values  $< 1.5$  result in 1) are used to set points in the Points panel (see Section 14.4). These values then (or the values which have been set by hand) will be transferred to R (if active!) when the **R** button is pressed. A message dialog confirms the transfer. With this procedure the following values will be transferred to R:
- **alpha** -> the alpha value of the points
  - **diameter\_panel** -> the diameter of the points in the panel
  - **fieldx ,fieldy** -> the size of the "Area of Analysis"
  - **species** -> the species which have been set
  - **x ,y** -> the coordinates of the points
- b)** To support a kind of pattern or shape recognition the other **Points** button will execute a method which directly transfers particle measurements from ImageJ to R. If a particle analysis with selected measured attributes (Set measurements, see Illustration 21) was made with ImageJ, the selected attributes would be transferred

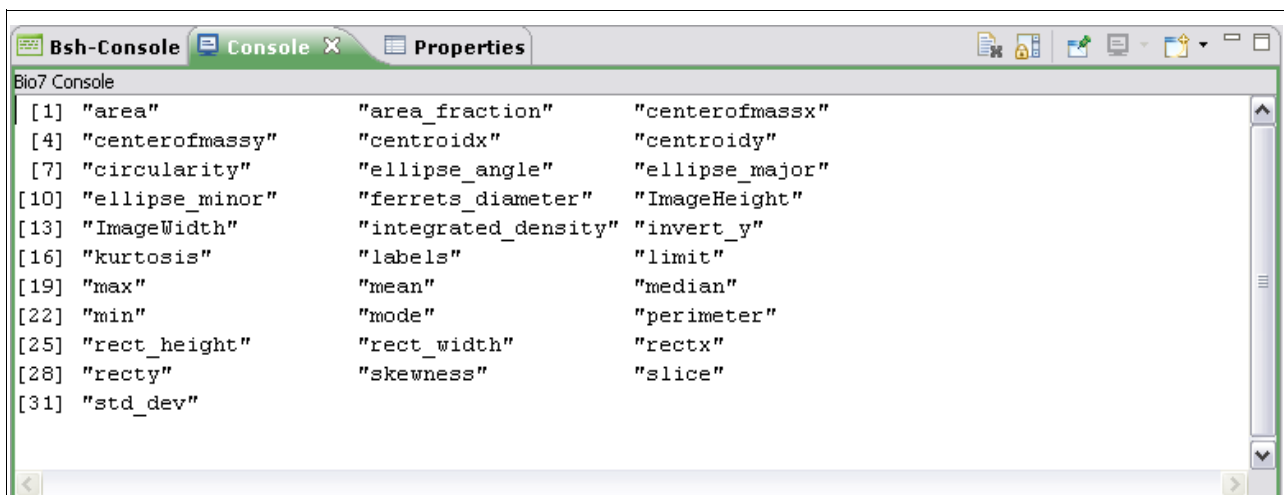
to R for further statistical analysis. Here the exact (not rounded ->doubles) values will be transferred to R (Illustration 22).





*Illustration 21: Particle attributes*

**12.** With the **Pixel** button thresholded pixel from the ImageJ view will be mapped to the active state of the Quadgrid. If the Quadgrid size is  $>1$  pixel than the thresholded pixel will fall into one quad coordinate of the (scaled) Quadgrid (looks like a coarsening of the structure !). If the pixel size of the Quadgrid is 1, all thresholded pixels will fall into one Quadgrid quad of the available Quadgrid area.



*Illustration 22: Particle values from ImageJ tranferred to R*

### 6.3.2.1 Using the Points panel for spatial continuous simulations

The Points panel can also be used to create spatially continuous models. The following snippet demonstrates the use of the Points panel for simulations in the calculation thread of the Bio7 application. Therefore an one dimensional point array (type Individual) has to be created and transferred to the Points panel (*setIndividual()*). The Individual objects must be parametrized with the coordinates (y,x), the state, the diameter and the alpha value.

Tip: The parameters diameter and alpha value, for example, can be used to display fitness or age in a simulation!

*Table 7: 2d continuous model*

```
/*This snippet creates individuals dynamically in the Points panel! */

Individual[] individual = new Individual[1000];
public void ecomain() {

    PointPanel.delete();

    for (int i = 0; i < individual.length; i++) {
        int x = (int) (Math.random() * 1000);
        int y = (int) (Math.random() * 1000);
        int species = (int) (Math.random() * Bio7State.getStateSize());
        float alpha = (float) (Math.random() * 255);
        int diameter = (int) (Math.random() * 50);
        individual[i] = new Individual(x, y, species, diameter, alpha);

    }
    //Set the individuals in the Points panel !!
    PointPanel.setIndividual(individual);
}
```





*Illustration 23: The thumbnail browser*

### 6.3.3 Thumbnails

With the thumbnail browser it is possible to open a directory of images as thumbnails. Different options are available for the display of thumbs.

1. With the "**x->...**" buttons you can select how many thumbs should be created in a row
2. The **Small**, **Medium**, **Large** labelled buttons will create preview images with a size of 100\*100, 150\*150 or 200\*200 pixel.
3. The button with the folder image opens the directory which should be displayed.
4. The combobox will adjust the level of magnification. The thumbs will be resized by means of the lens (blue rectangle in the right panel for the overview!).

Images or file extensions in the following formats will be opened:

- \*.gif, \*.jpeg, \*.png, \*. bmp, \*.tiff (with preview!)
- \*.pnm, \*.pcm (without preview!)
- \*.txt (e.g. macros, results!)
- \*.roi (single ROI!)
- \*.zip (Stacks, ROI Manager files!)

The images can be **opened** in the ImageJ view by selecting the images in the left panel (press to open!). If you set the mouse over a thumbnail some **information** about the image will be displayed.

## Annotation:

If some images in the correct file format do not open in the ImageJ view, it will be necessary to rename the extension of the images to the previously mentioned endings. Some image extensions (\*.JPG instead \*.jpeg) opened by the thumbnail browser cannot be recognized by ImageJ (the thumbnail browser uses swt to draw the thumbs!).

## 6.4 The Soil Perspective

The **Soil** perspective supports the visualization of several abiotic factors in a discrete grid which are:

- 1. Nitrate**
- 2. Phosphate**
- 3. Carbon**
- 4. Water**
- 5. Root-Density**

The different soil panels can be magnified like the Quadgrid panel. Furthermore values can be set, using the mouse according to adjustment in a specialized option panel (drag values like in the Quadgrid panel). The field size of the different soil panels is equivalent to the field size of the Quadgrid. If the Quadgrid is resized the soil panels will be resized too.

All values in the underlying grid array can be accessed and altered through a small API programmatically. It is also possible to make the grids dynamic in the calculation thread of Bio7.

```
/*A simple example how to transfer data dynamically to the soilpanels! */
public void ecomain() {
    for (int y = 0; y < Field.getHeight(); y++) {

        for (int x = 0; x < Field.getWidth(); x++) {
            // Fill array from the different panels with random values
            Nitrate.setNitrate(x, y, (int) (Math.random() * 999));
            Carbon.setCarbon(x, y, (int) (Math.random() * 999));
            Phosphate.setPhosphate(x, y, (int) (Math.random() * 999));
            Water.setWater(x, y, (int) (Math.random() * 999));
            RootDensity.setRootDensity(x, y, (int) (Math.random() * 999));
        }
    }
}
```

```

//Only for painting required !

Nitrate.doPaint();

Carbon.doPaint();

Phosphate.doPaint();

Water.doPaint();

RootDensity.doPaint();

}

```

The different panels are also visualized in the 3d-view of the program so that the influence of certain abiotic factors on the plant composition, etc. is directly visible.

### 6.4.1 The Soil

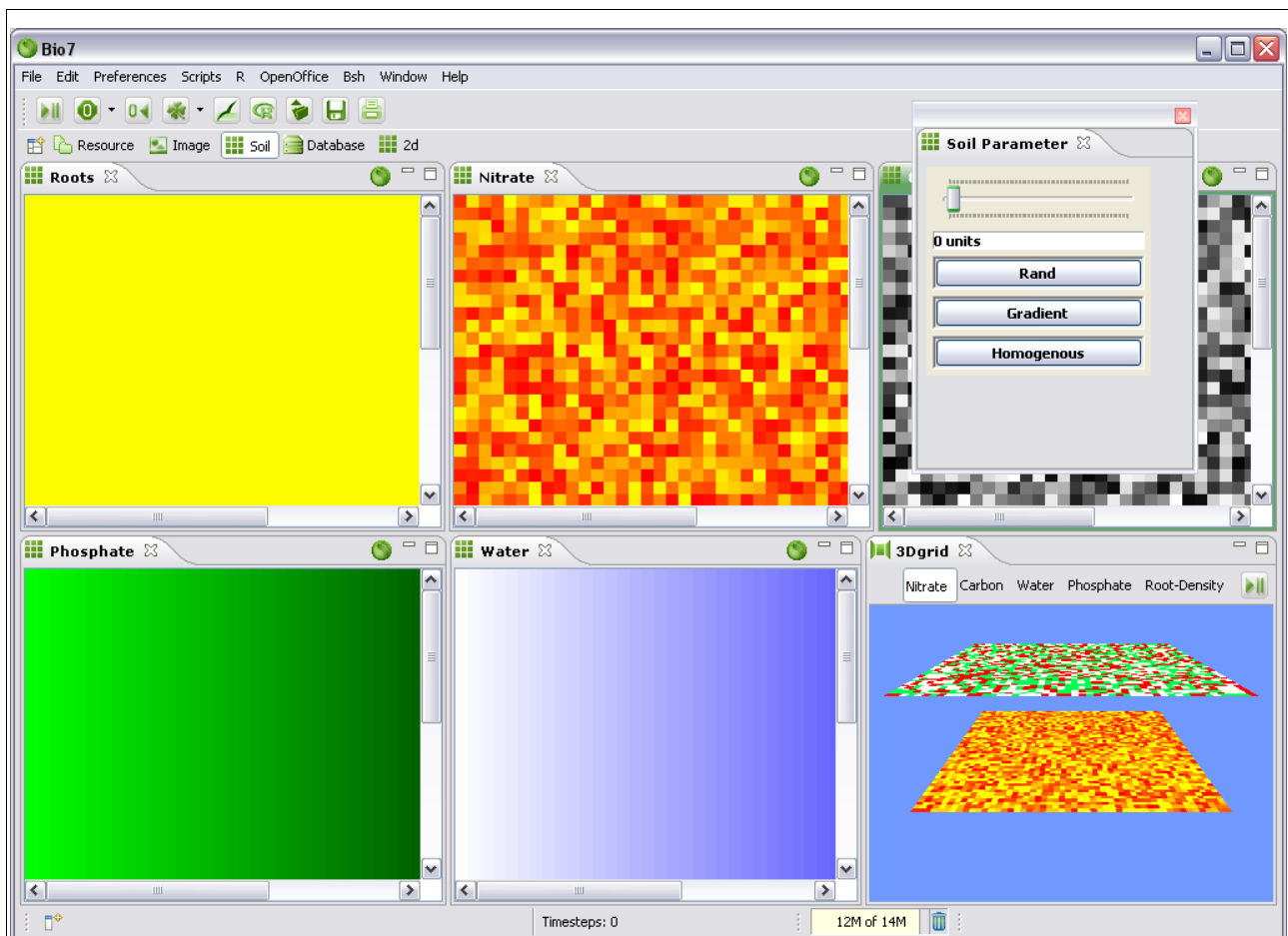
In the top right of the Soil perspective a detached dialog can be activated to adjust parameters for the soil panels. Values (0-1000 units) can be adjusted for the different panels with a slider. After adjusting they can be set directly with the mouse in the activated panels. The values are displayed in units which then can be assigned or mapped to any scale of concentration, etc. The three available buttons in the panel offer methods to bring in random, homogeneous or gradient values in the soil fields.

1. If the **Rand** button is pressed, values will be dispersed from 0-1000 units randomly across the whole field.
2. If the **Homogeneous** button is pressed, all values inside the field will be set to zero.
3. Additionally, if the **Gradient** button is pressed, values in the field from 0-1000 units will be dispersed in a gradient from left to right.

The values are displayed with 256 colours by default.

### 6.4.2 Nitrate

The Nitrate Panel visualizes a discrete field for nitrate values. Values can be set from 0-1000 units (with the **Soil Parameter** panel). The lowest values are coloured yellow. The highest



*Illustration 24: The different soil panels and the visualisation in 3d*

values are in red. The 1000 possible units will be displayed with 256 colours.

### 6.4.3 Phosphate

The Phosphate Panel visualizes phosphate values from the lowest values (green) to the highest values (black) in a discrete grid.

### 6.4.4 Carbon

Like the above described panels the Carbon panel visualizes another important nutrient in ecological systems. The values will be displayed in the colours white (lowest) and black (highest).

### 6.4.5 Water

A Panel for water availability is also available for displaying water content in a discrete grid. Values from white to blue will display values from 0-1000.

## 6.4.6 Root-Density

A further panel visualizes the root density in a grid. This Panel can be used to model belowground activities of plants. Low values are yellow, high values are marked black.

## 6.4.7 Additionally

The panels can also be used to visualize other important driving factors in an ecosystem.

For example the Root-Density Panel can be used to visualize mycorrhizal activities or the nitrate panel can be used to display the amount of nitrogen fixation of bacteria, etc.

All soil panels serve to visualize simplified and generalized information of complex processes in the belowground sphere of ecological systems.

## 6.5 The Database Perspective

The **Database** perspective offers two views. In the **Formular** view a form is available for creating Plants in an object oriented database (dbo4). These stored plants are used as templates for the Quadgrid (Hexgrid) Plant array. The database will be updated with a new plant when the **Send** action has been invoked.

Update and deletion of plants is similarly simple. If plants in the database are to be updated or

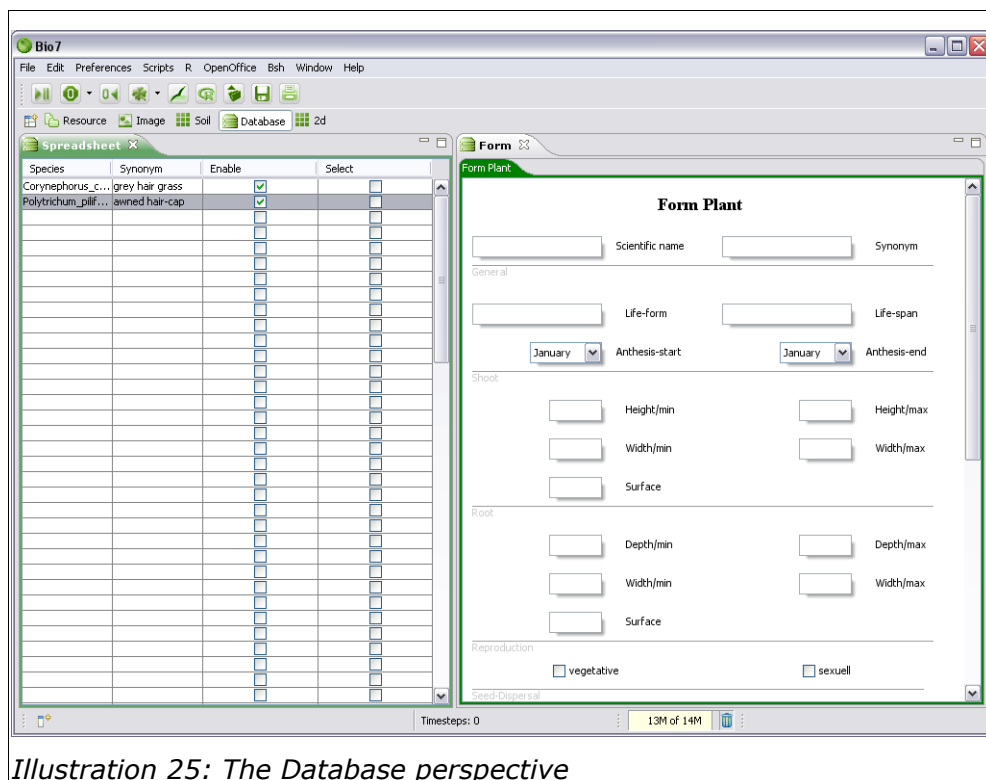
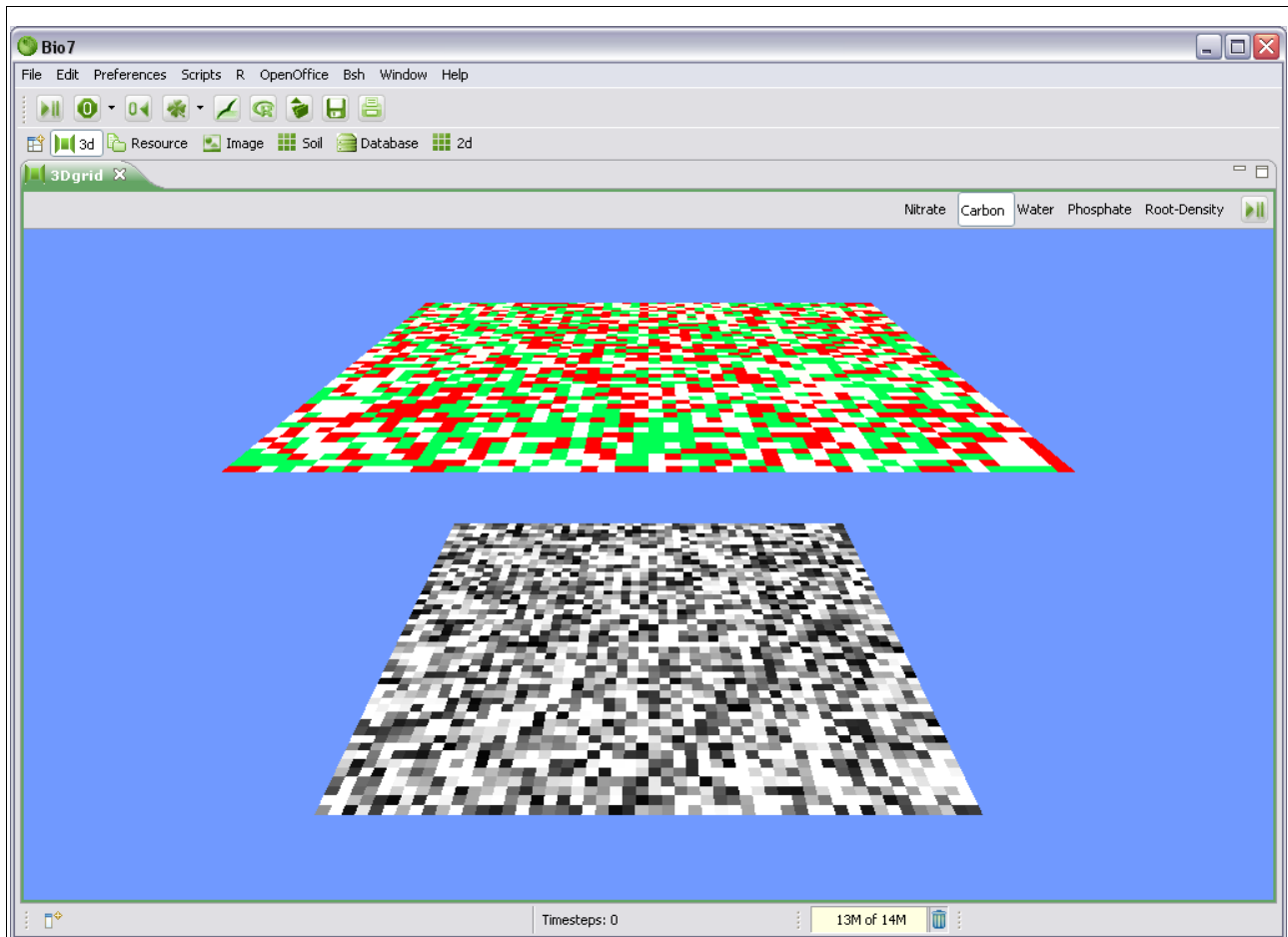


Illustration 25: The Database perspective

deleted they have to be selected in the **Spreadsheet** view. After the selection of the right plant in the **Select** column the current attributes of the plants appear in the form of the Formular view. For deletion (also for updating and sending !) it is required to disable all active plants in the spreadsheet (column Enable) for a simulation. If plants have been stored in the database they can be enabled for simulation in the **Spreadsheet** view. By selecting them in the appropriate row with the checkbox in the column **Enable** plant, objects will be created according to the selected Plant template from the database and dispersed in the Plant array (plant, tempplant) as well as in the numeric array (xystate, xytempstate) of the Quadgrid (Hexgrid). The object properties of the dispersed plants can easily be altered and accessed programmatically. Additionally in a specialized view these properties can individually be changed without any effort of programming (see Section 6.2.4.2).

## 6.6 The 3d Perspective



*Illustration 26: Layered grids in 3d*

In this perspective the Quadgrid panel and one (optional) soil panel are visualized. With the Buttons in the toolbar the different soils can be activated in this view. Dragging the mouse in the view to the left or to the right shifts the layers between front and back. Dragging the mouse up or down will change the rotation axis of the displayed grids. With the scroll wheel of the mouse the distance of the layers can be altered. Finally, by pressing the arrow-keys of the keyboard, you can move the layers up and down, left or right for calibration.

*Table 8: Functions*

Device	Event	Function
Mouse	Drag left, right	Bring layers to front, back
Mouse	Scroll	Distances between layers
Mouse	Drag up, down	Rotation of layers
Key	Arrow left, right	Move layers left, right
Key	Arrow up, down	Move layers up, down

## 6.7 Custom Perspectives

It is also possible to define custom perspectives with Bio7. All views from the application can be used to define a specialized perspective for certain tasks.

*Table 9: From the Eclipse 3.2 Platform*

If you have modified a perspective by adding, deleting, or moving (docking) views, you can save your changes for future use.

- 1.** Switch to the perspective that you want to save.
- 2.** Click **Window > Save Perspective As....**
- 3.** Type a new name for the perspective into the **Name** field.
- 4.** Click **OK**.

The name of the new perspective is added to the **Window > Open Perspective** menu.

The Custom perspective can be deleted in the Preferences dialog (**Preferences->General->Perspectives**). There you can simply remove the perspective.

### 6.7.1 Custom Views

Besides the possibility to extend Bio7 with custom perspectives also three **custom-views** are available (**Custom3d**, **Custom Controls**, **Custom-Charts**) for embedding self defined GUI's in Bio7. Self defined JPanels (Swing), swt-canvas and Draw2d-Panels can be easily embedded in the CustomControlls view. Furthermore for embedding OpenGL a GLCanvas can be sent to the Custom3d view in Bio7. Finally it is possible to integrate JFreecharts into the Custom-Charts view.



Table 10: Custom views API

View	Component	How to call
CustomControlls	JPanel (Swing)	CustomView.setPanel(JPanel,"name");
CustomControlls	Canvas (Swt)	Composite parent=CustomView.getComposite("name"); (see Table 11 for details)
CustomControlls	GLCanvas (OpenGL)	Custom3D.setPanel(canvas,"name",animator); (see Table 12 for details)
Draw2d		LightweightSystem lws=CustomView.getDraw2d("name"); (similar to Table 11..see Snippets)
Custom-Charts	JFreechart (0.9.20)	ChartView.setChart(chartPanel,"name");

Table 11: How to wrap swt

<pre> org.eclipse.swt.widgets.Composite parent=CustomView.getComposite("name"); Display dis=CustomView.getDisplay();          dis.syncExec(new Runnable() {             public void run() {                  // The Code goes here !              }         }); </pre>
--

### To OpenGL:

It is likely that an Animator instance starts in an OpenGL script. For this case it is required to pass an animator variable to the Bio7 function. If the view or the panel is closed, the animator will be stopped too (with the command `animator.stop()`) which would not happen if no animator reference was passed to Bio7.

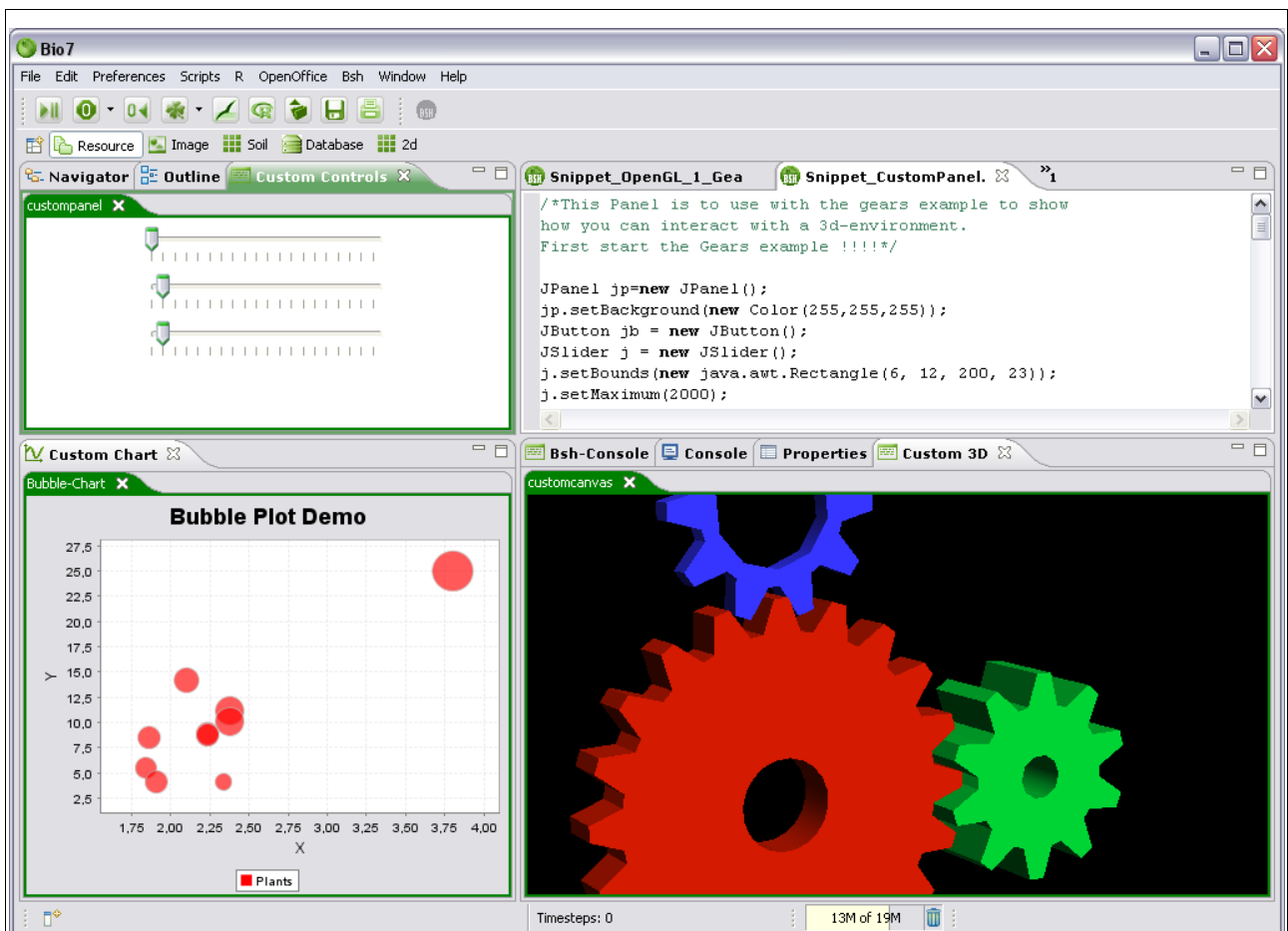


Illustration 27: Custom views in Bio7

Table 12: How to transfer the animator

```
.....
// The Animator !
Animator animator = new Animator(canvas);
animator.start();
// Call the view with the custom GLPanel , pass a name, and pass the animator !
Custom3D.setPanel(glpanel,"name",animator);
```

Each animator can also be stopped by calling the stop function inside the BeanShell console.

## 6.7.2 Custom Menus

Another nice feature of Bio7 is the simple way to extend the Script menus in Bio7 with BeanShell scripts and ImageJ macros. Bio7 recognizes files dropped to special folders (For the path open the Preferences of Bio7) automatically and extends the menus with the filename of the dropped files. Menus which can be extended are:

Table 13: Custom menus

Location	Name	Category	Script type
File Menu	Import-Scripts	File import	BeanShell
File Menu	Export-Scripts	File export	BeanShell
Scripts	General-Scripts	General	BeanShell
Scripts	Statistic-Scripts	Statistic	BeanShell
Scripts	ImageJ-Macros	Image Analysis	Macros
Scripts	Chart-Scripts	Charts	BeanShell

The menus can be extended in different categories for the execution of custom methods in Bio7. The categories all be a help in organizing custom methods in Bio7. In addition it is possible to execute R scripts by means of a BeanShell Script and the Rserve connection. See Section 14.12 for a detailed description how to add a menu.

## 6.7.3 Custom Startup

BeanShell scripts can be used to customize the platform directly behind the start of the Bio7 application. These scripts will be executed with the BeanShell interpreter. The scripts have to be placed in a specialized folder **Preferences Bio7->Custom Startup** for custom activities like :

- BeanShell imports at startup.
- Loading a Custom Perspective or a view.
- Execute a startup setup in the different panels.
- Invoke an action from the Bio7 platform
- Import libraries
- Start Server application at startup

See Section 14.12 for placing a script for the Bio7 application.

## 7 The Editors of Bio7

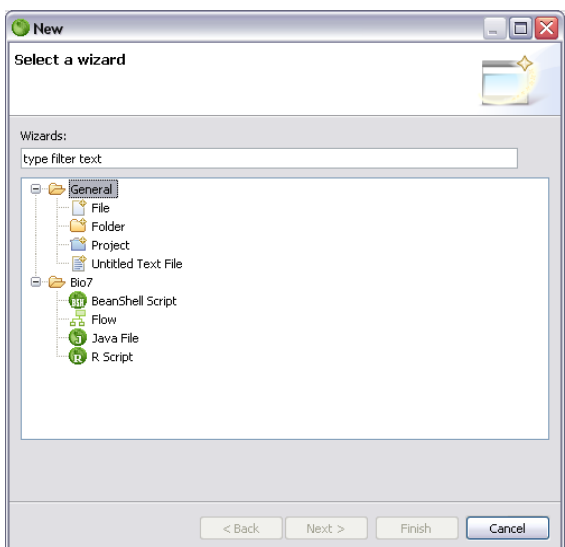
### 7.1 General Features

There are several editors integrated in Bio7 which support the syntax of different programming languages and a specialized Flow-Editor to drop files from the other editors for execution. The "language" editors support the syntax of the languages Java (BeanShell uses also a Java-editor) and R. For an instant help select an active Editor (Java, R, BeanShell, Flow editor) and press **F1**. A text editor is also offered by default integrated in the platform (an editor for arbitrary files as well!). All editors (if opened) offer more or less customized menus and toolbars and also offer custom context menus for the work with the different languages.

By activating a language editor, typical text tools are activated in the edit menu of the main menubar of the Bio7 application.

- Before creating a file, a Project has to be created where the files can be placed (**File->New->General->Project**).
- A file can be created by using a wizard, which can be activated in the File menu. (**File->New->Other->Java File**) or in the context menu of the Navigator. (e.g.: **New->Other->Java File**).
- The file itself will be opened in the editor area, which can be hidden or shown in a perspective by selecting the action **Window->Show/Hide Editor**.

In the Flow editor Java, R or BeanShell files can be dragged directly to the flow.



*Illustration 28: The Select a wizard dialog of Bio7*

Table 14: From the Eclipse Workbench User Guide (<http://help.eclipse.org/>)

Most perspectives in the Workbench are comprised of an editor area and one or more views.

You can associate different editors with different types of files. For example, when you open a file for editing by double-clicking it in one of the navigation views, the associated editor opens in the Workbench. If there is no associated editor for a resource, the Workbench attempts to launch an *external editor* outside the Workbench. (On Windows, the Workbench will first attempt to launch the editor in place as an OLE document. This type of editor is referred to as an *embedded editor*. For example, if you have a .doc file in the Workbench and Microsoft Word is registered as the editor for .doc files in your operating system, then opening the file will launch Word as an OLE document within the Workbench editor area. The Workbench menu bar and toolbar will be updated with options for Microsoft Word.)

Any number of editors can be open at once, but only one can be active at a time. The main menu bar and toolbar for the Workbench window contain operations that are applicable to the active editor.

Tabs in the editor area indicate the names of resources that are currently open for editing. An asterisk (\*) indicates that an editor has unsaved changes.

By default, editors are stacked in the editor area, but you can choose to tile them in order to view source files simultaneously.

### 7.1.1 Code Templates

Alternatively to the simple Snippets view (see 6.1.4) all language editors offer Code-Completion by pressing **Ctrl+Space** or by right-clicking the Context menu -> Code Completion.

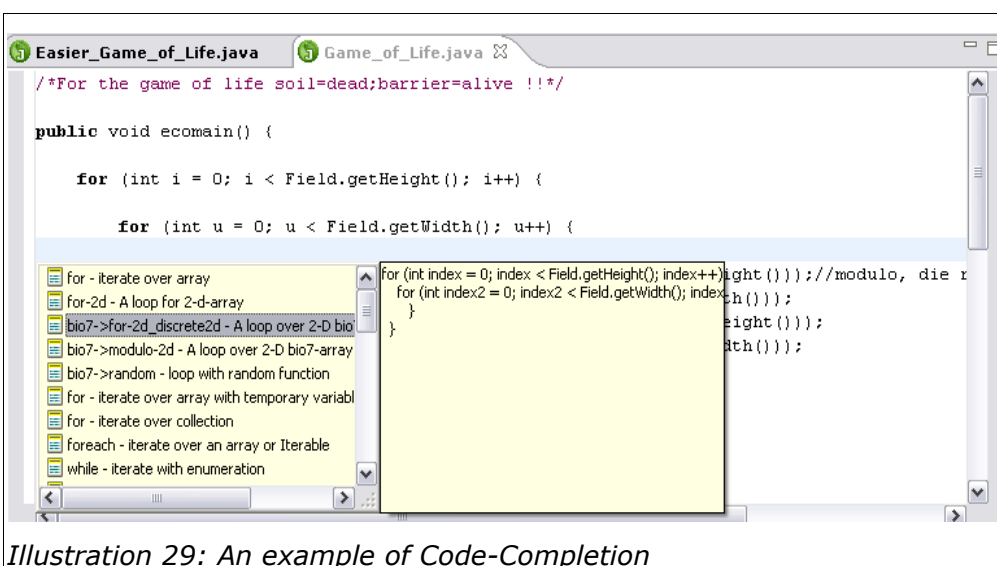


Illustration 29: An example of Code-Completion

Table 15: From the Eclipse Workbench User Guide (<http://help.eclipse.org/>)

Templates are a structured description of coding patterns that reoccur in source code. The (Bio7) editors support the use of templates to fill in commonly used source patterns. Templates are inserted using content assist (**Ctrl+Space**).

For example, a common coding pattern is to iterate over the elements of an array using a for loop that indexes into the array. By using a template for this pattern, you can avoid typing in the complete code for the loop. Invoking content assist after typing the word `for` will present you with a list of possible templates for a for loop. You can choose the appropriate template by name (`iterate over array`). Selecting this template will insert the code into the editor and position your cursor so that you can edit the details.

Templates for the languages can be created and stored in the preferences dialog when opening **Preferences->Preferences Bio7->Preferences BeanShell-Editor ->Template Editor BeanShell**.

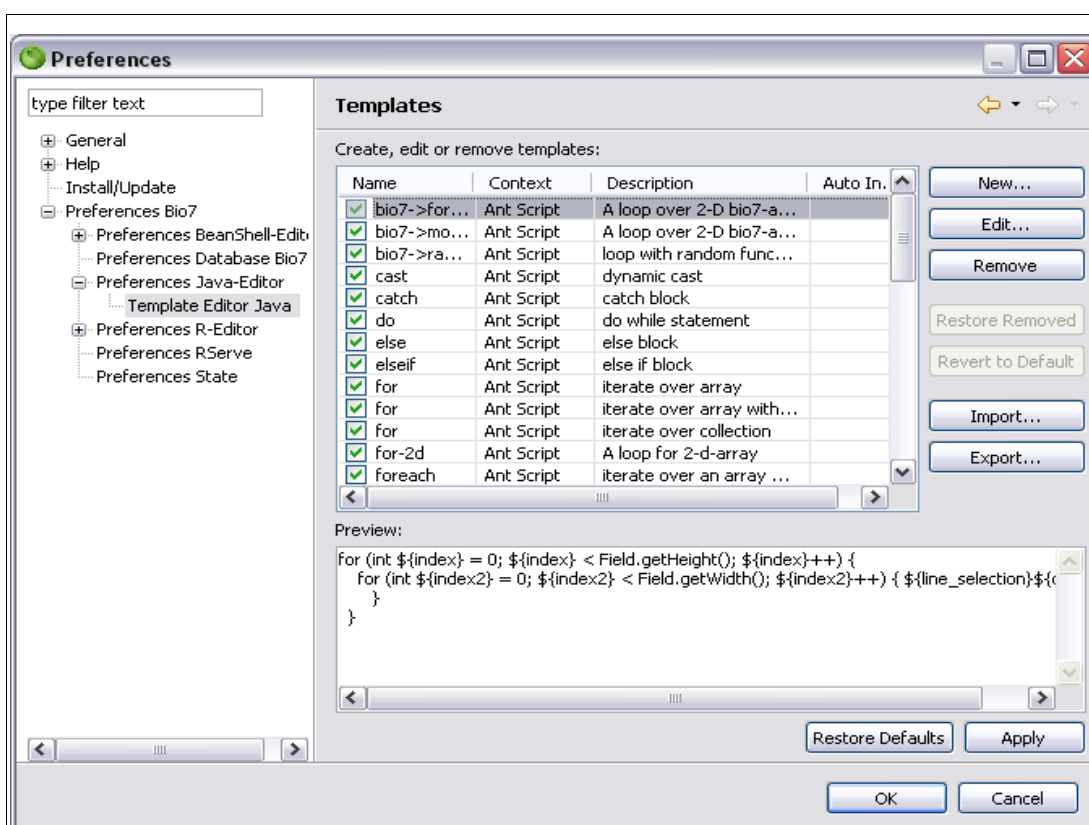


Illustration 30: Creation of templates

#### Annotation:

The templates can also be used to create custom formulas to facilitate the development of ecological models. By default it is possible to share this self-defined code-templates with

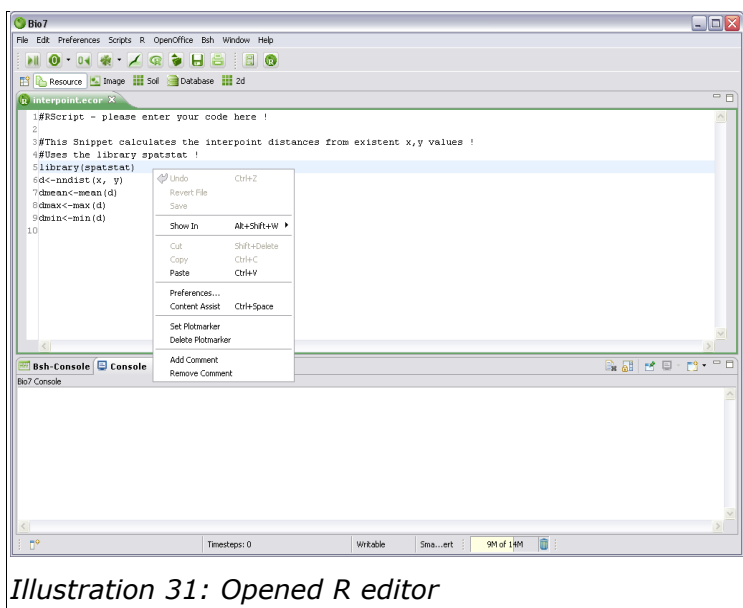
others by using the integrated import-export functions of the different template editors available for each language in Bio7.

### 7.1.2 Preferences of the Editors

In the preferences dialog of Bio7 there are a lot of functions available to customize the different language editors. For example, the fonts for all editors can be customized under **General->Appearance->Colours and Fonts**. If the colours of the keywords, etc. should be changed, it could be done under **Preferences Bio7->Preferences BeanShell-Editor** for the BeanShell editor.

## 7.2 The R-Editor

The R-Editor of Bio7 supports the syntax of the R-language. If an R file is created with the default wizard the extension of the created RScript file is **\*.ecor** instead of **\*.R** (to distinguish between other editors which use \*.R as extension). If you have an \*.R file outside the workbench, this file can simply be dropped to a Project of Bio7. The only change which has to

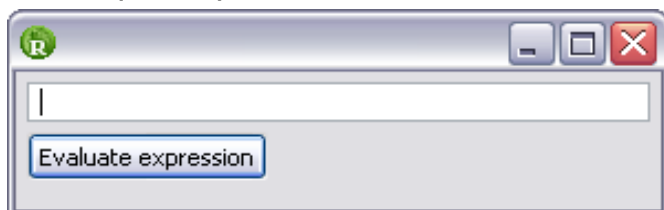


*Illustration 31: Opened R editor*

be made, is to rename the file extension to **\*.ecor** so that the default editor can recognize the file. If a R file has been opened, the toolbar of the Bio7 application displays two new items which have some special functions (if a connection to the Rserver and R has been established). The **Interpret RScript** button executes the interpretation of the active RScript file (you can also choose a splitted view with two files) in the editor area. By selecting this action, the whole file will be interpreted. Created variables will be accessible by means of the second item (or in the main menu of Bio7: **R->Evaluate Expression**).

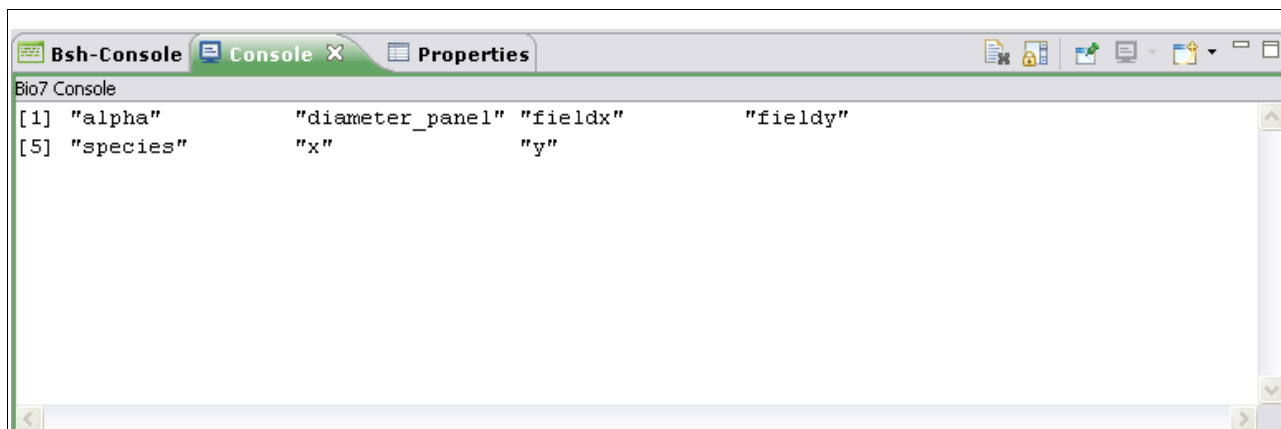
## 7.2.1 Evaluation

The **Evaluate Expression** action simulates a kind of console like in R. When this action is selected a dialog with a textfield inside pops up where R expressions can be evaluated. All evaluated expressions are shown in the console of the Bio7 application. Because Bio7 uses a server (Rserve) as a fast connection to R, there are some differences between a regular R



*Illustration 32: Expression dialog*

version and the R version of Bio7. Bio7 tries to mimic some of the functions which are normally not available with the Rserve application. The functionality of a shell in R is imitated in Bio7 with the action Evaluate Expressions which mimics a kind of R console in Bio7.



*Illustration 33: Output in the console*

**Note:** If the key **F3** inside the text field of the expression dialog is pressed, a command list will open with some common expressions for R. If you press **F2**, a history dialog will be displayed.



### 7.2.2 Plotting

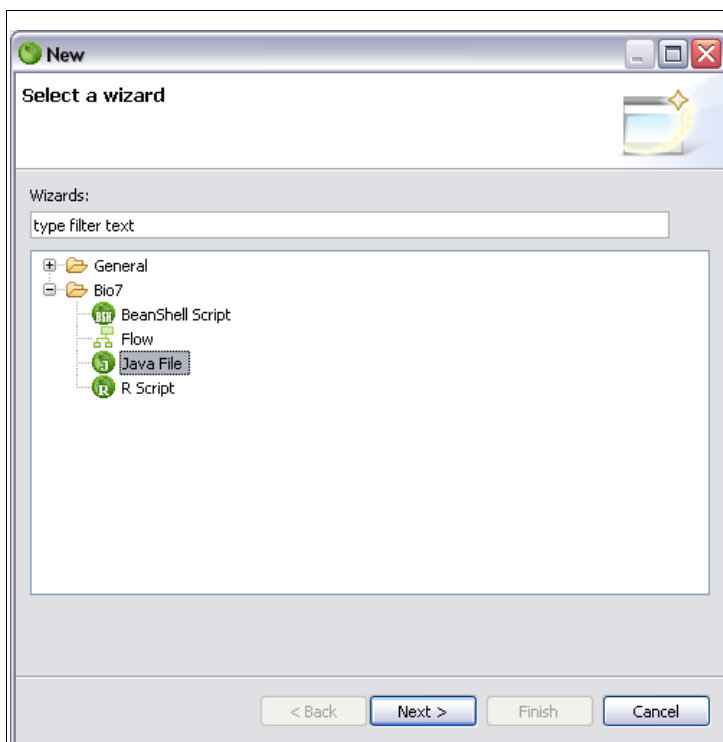
Another feature and one advantage of R are the plotting functions which are not available directly with the server connection in Bio7 (you can create plots in pdf , jpeg or png files with the Server and open them!). There are two actions available, to imitate this plotting function in the context menu of the Bio7 editor where you can set or unset **plot markers** for plotting operations. If a regular R-command for a plot is marked inside a script (This can be done by line selection. By default the blue marked line is the selection) and the script is executed, a temporary file (\*.pdf) will be created and automatically opened by the default pdf application (multiple line selection is possible !).

The plot selections are created as tasks in the Bio7 platform (the taskview of the Bio7 platform can be opened as a management tool for the marked plots in different \*.ecor files !). It is important to know that the marked plot instructions are evaluated after the script has been evaluated. So it is an advantage to write the plotting instruction at the end of the script. If more than one plot is marked in one script the resulting \*.pdf file contains all the plotting instructions in one file. In addition, there is a method available, in the context menu of the R editor which places a **comment** at the selected lines. All other functions which are available in the context menu of the editor and the edit menu inside the main menu bar of Bio7 are well known functions like: undo, redo, cut, copy, paste, delete, select all, print and find/replace (open a search dialog for the document).

## 7.3 The Java- and the BeanShell Editor

### 7.3.1 Java

The Java editor of Bio7 is a regular Java editor which supports Code templates and the colouring of the syntax of Java. Text functionalities (Copy,Paste, etc.) are the same as in the RScript editor. For the compilation of the Java file the Janino compiler is used embedded in Bio7 (<http://www.janino.net/>). A new Java file can be created with the Java wizard in the Bio7 File menu (**New->Bio7->JavaFile**). The instructions of the wizard will guide you through the procedure of file creation.



*Illustration 34: Creating a Java file*

When a Java file has been created, with the wizard one item will be added to the Bio7 toolbar for an easy compilation of the Java file. **The syntax of the Java file must be the syntax of the Java <1.5.0 (1.4.2)** specification. All new Java  $\geq 1.5.0$  features like annotations or generics, etc. are not supported (exceptions are static imports and autoboxing and unboxing!) by the embedded compiler which compiles the file from the editor directly to the calculation thread of Bio7. The Java wizard creates a file which contains a main method (ecomain) which is necessary for the compilation of the file. If a Java file was compiled and the calculation thread was started (by pressing the start/pause button), in every time step this main method would be invoked. To go into more detail the embedded compiler compiles a class, which visible part is the class body that can be defined by the user.

```
Ecoclass X { // Not visible!
```

```
import com.eco.example; // The imports can be made here !
```

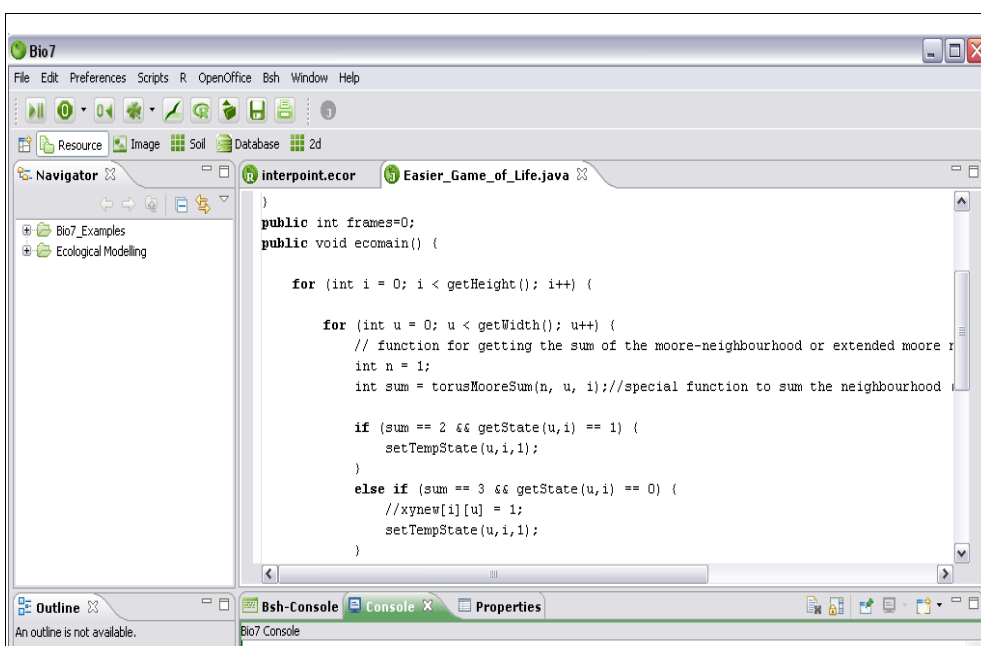
```
public int a=0;
```

```
public void ecomain(){
```

```
    // This is the main method !
```

```
}
```

```
} // not visible
```



*Illustration 35: Opened Java editor*

## Compilation

Compilation of the active Java file will be easily executed when clicking on the **Compile Java** toolbar item which is visible when a java file has been opened. The compiled class is programmatically available by a static variable "method" in the class **Compiled** (Compiled.getEcoclass().ecomain(); ->will call the ecomain method), which has been compiled by the embedded compiler. A lot of imports are transferred to the compiler by default, to access all Bio7 methods and the API. Extra imports can be made like in regular Java files (see Snippets).

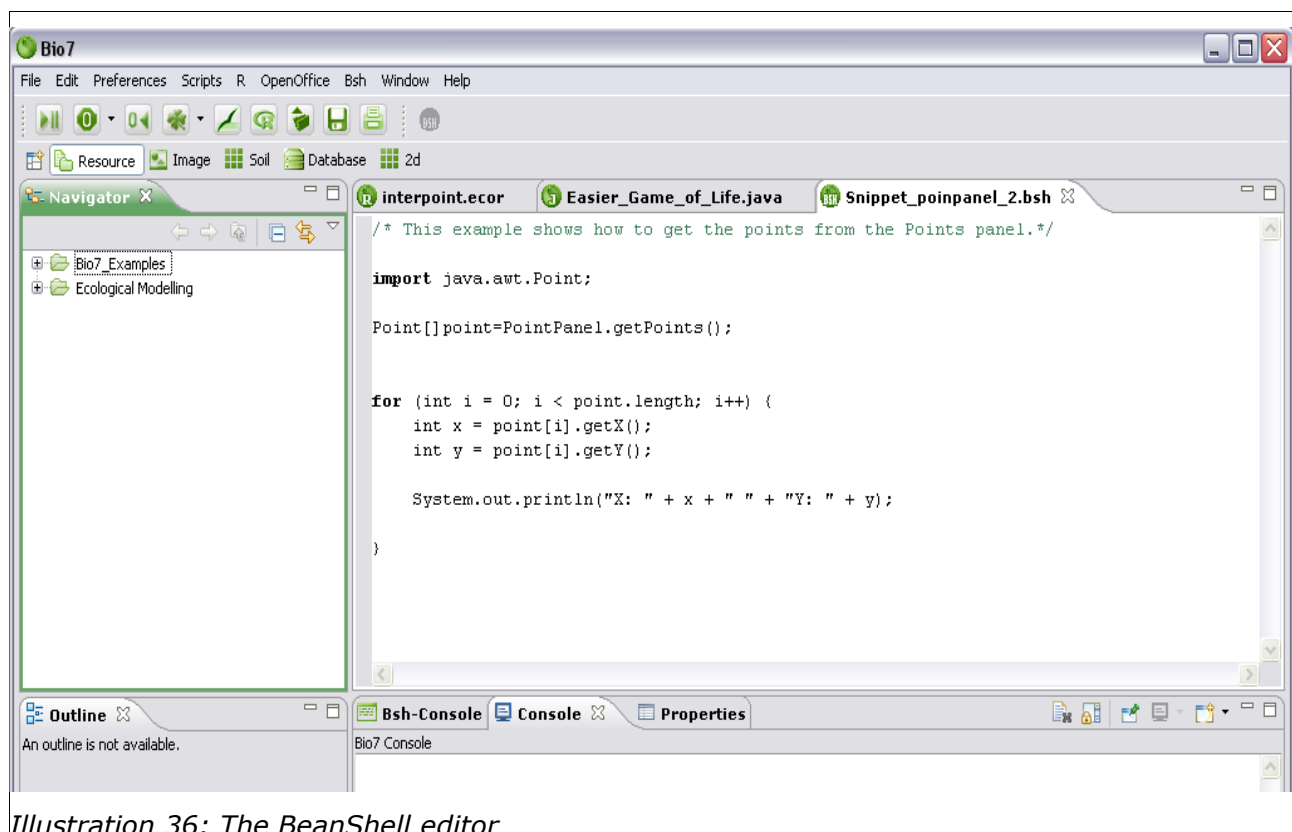
### Note:

The default imports for the java compiler can be changed in the file ...\\plugins\\com.eco.bio7\_1.0.0\\bin\\imports\\compiler.properties (Windows). It is also possible to create a **setup** method. If you embed a method with name setup **public void**

**setup(){.....}** after the compilation, you will be able to call this method with an embedded script to set states or call another setup method **Scripts->General-Scripts->Setup** (You can also change the script for calling custom methods, etc. from the compiled class!).

### 7.3.2 BeanShell

The BeanShell editor as well as the Java editor supports the Java syntax. Code templates, (identically to that in the Java editor) are available too (some script code templates can differ!). In General BeanShell is capable to interpret ordinary Java code, but also has complete scripting capabilities to facilitate the development process of programming. Scripting code as well as Java code can be executed with the item, which appears when a BeanShell file \*.bsh has been opened. For more scripting examples visit the website of BeanShell (<http://www.beanshell.org>)



*Illustration 36: The BeanShell editor*

The reasons that BeanShell and a separate Java-compiler have been added to Bio7 are the following:

1. Need for speed in complex applications-> The embedded Java compiler compiles the Java code to bytecode while BeanShell interprets the code which is much slower and

sometimes not sufficient for resource intensive calculations (though compiled code will be executed also very efficiently by BeanShell!).

2. Scripting capabilities-> With BeanShell it is possible to create highly dynamic applications with customized scripts. Script templates like the Snippets can easily be recycled for custom modification. Scripts can be used to extend the Bio7 script menus.
3. Direct interaction ->With the BeanShell console you can interact directly with methods and integrated API's of Bio7.

Code, which was written for the Java compiler can easily be switched to BeanShell (supports full Java syntax!). A complete manual with the features of BeanShell and the use of the scripting capabilities is available at the mentioned website of BeanShell. In Bio7 the console of BeanShell is also embedded for direct interaction with the interpreter of BeanShell and the Bio7 environment.

*Table 16: BeanShell Uses (source: BeanShell website)*

BeanShell Uses :

- Interactive Java - try out object features, APIs and GUI widgets - "hands on".
- Scripting extension for applications - Allow your applications to be extended via scripts in an intuitive and simple way.
- Macro Languages - Generate scripts as macros and execute them live in your VM easily.
- Education - Teach Java in a hands-on, live environment
- Expression evaluator for scientific, financial apps and rules engines - evaluate complex expressions with conditions and loops.
- Remote debugging - Embed a live, remotely accessible shell / command line in your application with just a few lines of code.
- Use BeanShell declaratively to replace properties files and replace startup config files with real scripts that perform complex initialization and setup with the full Java syntax at their disposal.

Bio7 uses one connection (like in R) or "namespace" in BeanShell. That means that a defined variable or class is active as long as it is not removed from the "namespace". By call the function "clear()" all variables, methods, and imports will be cleared from the "namespace" of the connection (this command is available in the BeanShell menu !)

The interpreter itself can also be accessed programmatically by calling

```
Interpreter.doInterpret("a=4;");
```

It is also possible to combine a compiled class with an Interpreter call. Before you can call the methods, the Java files have to be compiled to BeanShell by a right-click on the Java files in the Navigator. Multiple selections are possible (See Section 14.13 for more details)!

*Table 17: Code snippet*

```
public void ecomain(){  
  
Interpreter.doInterpret("zufall.ecomain()",null);  
Interpreter.doInterpret("zufall2.ecomain()",null);  
  
}
```

**Note:** The default imports for the BeanShell interpreter can be changed in the file  
**...\plugins\com.eco.bio7\_1.0.0\bin\imports\beanshell.properties** (Windows)

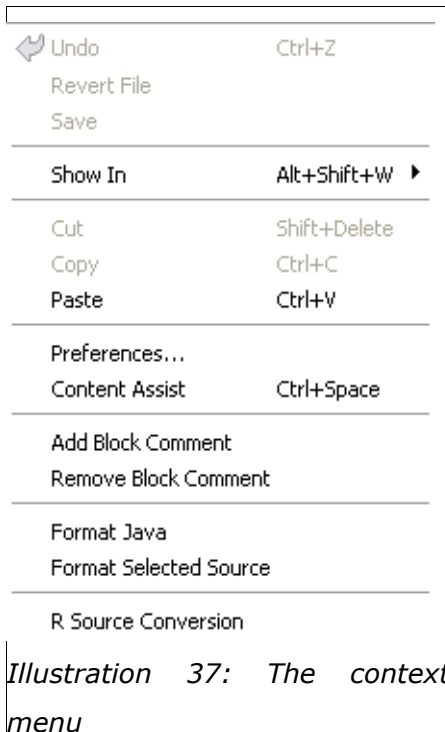
### 7.3.3 Differences

In principle the embedded compiler and BeanShell are compatible to each other in their language features, because BeanShell is also capable to interpret Java source code. The combination of both forms a strong tool as described previously.

Some differences have to mentioned:

- 1.** The BeanShell script will not be executed in the calculation thread.
- 2.** Both, compiled classes and BeanShell methods can be used independently from each other
- 3.** In a BeanShell script it is not necessary to have an ecomain method, but will be created by default. Afterwards this method or other methods can be accessed directly from BeanShell.
- 4.** BeanShell can access the compiled class of the compiler (access to methods, variables, etc.)
- 5.** The compiled class in Java can select the interpreter of BeanShell to execute methods, etc.(so you can use BeanShell methods also in the calculation thread).

6. Both, the compiler and BeanShell can interact with Rserve (the interpreter of Rserve) for calling statistical methods and transferring results in both directions. (For more information about the differences please consult the BeanShell manual! )



### 7.3.4 The context menu of the Java (BeanShell) editor

The context menu of the BeanShell editor and the Java editor offers some methods for editing the source and for source conversions.

These are (beside the regular actions of the platform):

1. **Add Block Comment** -> Surrounds the selected code with a block comment
2. **Remove Block Comment** -> Removes a selected block comment. (A precise selection is necessary!).
3. **Format Java** -> Formats regular java code (To work with BeanShell please create java code inside the BeanShell editor!)
4. **Format selected Java** -> Formats only the selected code. This is sometimes necessary if the formatter can not format the entire code.
5. **R Source Conversion** -> Converts Rscript code. Rscript expressions can be executed inside java and BeanShell. This converter helps to convert R expressions for use, e.g. in an Rscript. Please select the R expression and then execute this action.

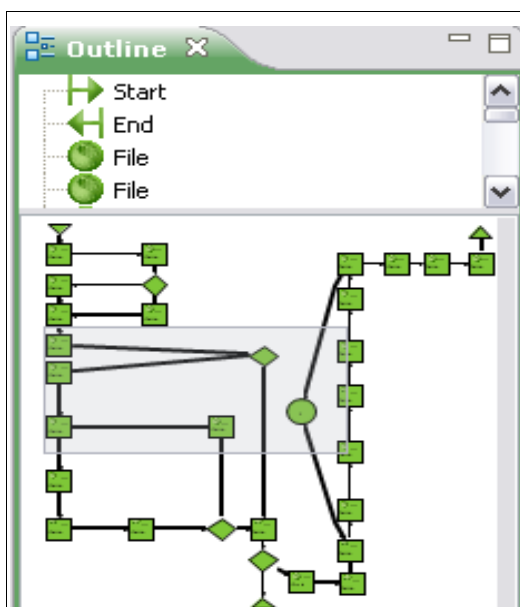
## 7.4 The Flow editor

The Flow editor of the Bio7 application is capable to handle and execute different files by their extension and display and arrange them in a "flowchart-like" structure.

- It can be used to create complex sensitivity analysis for created models
- It can serve as a guide for complex analysis of all tools offered by Bio7.
- Or it simply can be used to visualize complex workflows.

The main strength of this tool is that you can assemble all methods and information, created by files for modelling, statistical analysis or image analysis in a profound analysis. The Flow editor of Bio7 helps to organize and visualize the creation and analysis of ecological models. A flowchart by definition is a "schematic representation of a sequence of operations". In the flowchart of Bio7 files can simply be dragged from the navigator of Bio7 and dropped into the Flow editor of the application.

According to their file extension these files are executed in the sequence of the flow which has been created. A created Flow editor file has the extension **\*.ecoflow**. Some other files will be handled by default from the Flow editor when they are dragged from outside the workspace. Executable files (\*.exe) will be executed too and it is also possible to execute these files with an argument (by creating a file with the extension \*.argument). Files from outside, that have



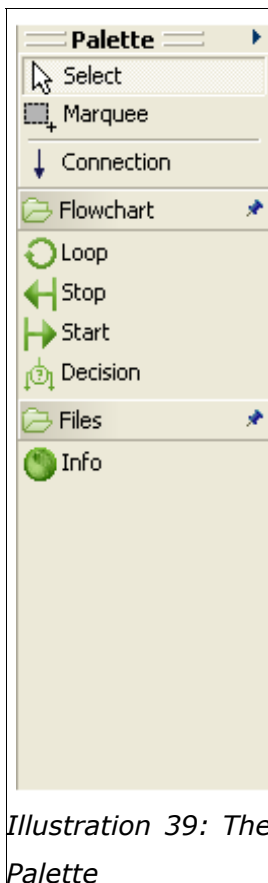
*Illustration 38: Outline of the Flow editor*

different extensions will be executed according to their registration in the system environment (e.g if you have an \*.pdf file and the registered program for that filetype is the Acrobat Reader, the file will be opened by default with this program). If the editor is opened, a lot of items will be added to the Bio7 toolbar and a new menu will appear in the menubar of the application.

Additionally, a specialized context menu will be available for calling different methods inside the Flow editor. A new **palette** bar will also be displayed for inserting flowchart elements in a new created file. Furthermore an **Outline** on the left will show an overview of the flow (if the flow cannot easily be displayed in the main window because of its size, a blue rectangle will be displayed in the Outline for a

flow selection). The selection is the part of the flow which will be displayed in the main Flow editor window.





### 7.4.1 The Palette of the Flow editor

For the creation of a running flowchart it is necessary to create a start and a stop element in the flowchart. By activating an element in the **Palette** these elements can be placed anywhere in the flowchart with the mouse at the appropriate coordinates (pressing mouse button will drop the component). Additionally when an element has been placed in the flow, it can be dragged to any coordinates in the flow panel. All other elements in the Palette can be placed by this procedure too.

The following elements are available and their values can be adjusted in the properties dialog. Furthermore some properties can be changed directly in the shape (mouseclick on the element) The following elements are available:

**1.Start** : The Start of the flow ->necessary for execution !

**2.Stop** : The end of the flow ->necessary for execution !

**3.Loop** : A loop will repeat a sequence in the flowchart. The number of iterations can be adjusted with an integer value directly in the shape.

**4.Decisions** : A decision in the flow will execute an if-statement. According to a self defined condition a static variable (`Bio7Flow.setDecision("true")`) can be set to true or false. The direction of the flow then depends on the linked connection which can be set to true or false in the properties dialog (by selecting the connecting lines !).

**5.Info** : A single line label for description which can be changed directly in the shape.

**6.Connection:** To determine the sequence of the flow **Connections** are available to connect components. If selected in the Palette directed connections can be drawn from component to component.

**7.Selections** can be made with the **Marquee** tool and the **Select** tool (multiple selections are possible).

The execution of a flow will be triggered with the item **FlowAction** in the top-right of the toolbar.

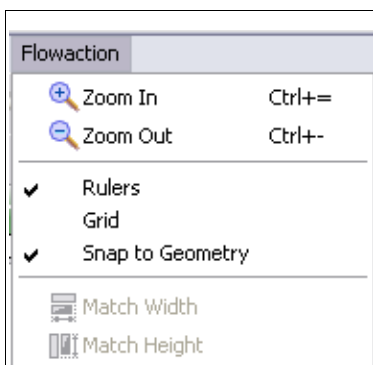
## 7.4.2 Options for the Layout

There are several tools available for adjusting the components and the layout of the flow. In the toolbar of Bio7 several items will be enabled when components are selected alone or in a group.



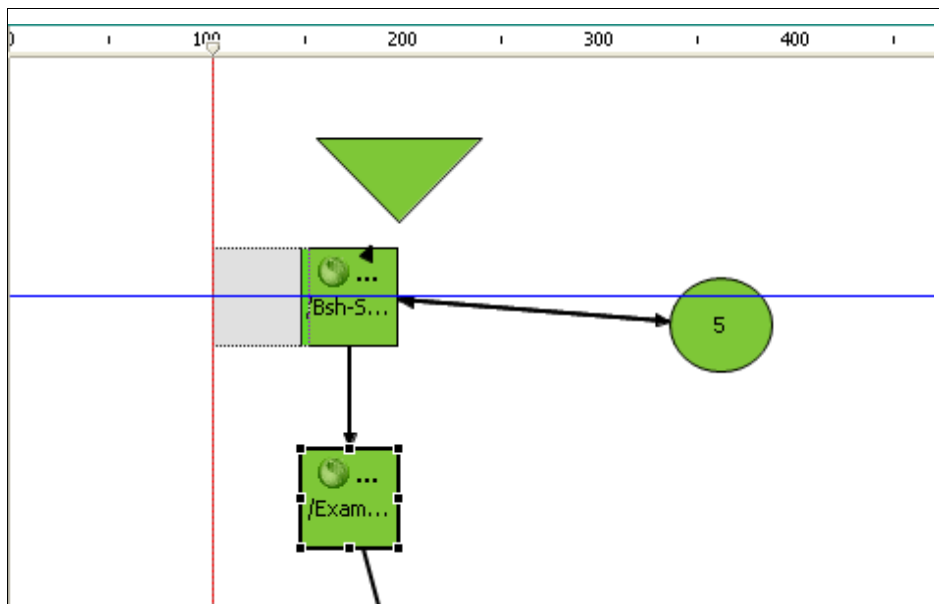
*Illustration 40: Added items to the toolbar of Bio7*

By default the zoom action and the flow action are enabled. With the **Zoom In, Out** actions it is possible to zoom in or to zoom out in the flow (by using the combo box or the lens items). If an opened flow was changed, the toolbar items for **undo** and **redo** could be used to undo or redo changes in the flow. Furthermore components can be deleted with the delete item and so removed from the flow (or by pressing the key *Delete*). If more than one component is selected, the alignment actions (**Align Width**, etc.) and the **Match Width/Height** actions will be available to align shapes to the first selected shape or adjust the width or the height of selected shapes to the properties of the first selected component. Additionally the added menu **Flowaction** in the menubar of Bio7 offers three further options for the alignment and layout of the components. When the option **Rulers** is selected and marked, rulers in the flow will be made visible for the exact layout of the flow components. By dragging horizontal and vertical lines from the ruler (like it is known from text editors), components can easily be aligned to the visible ruler lines. The alignment will be supported by artificial lines (red lines) which occur when alignment criteria are met. The option **Grid** will enable a grid in the flow. When enabled, the components can be arranged in discrete coordinate steps according to the default grid size in the flow. Furthermore artificial lines will occur (green lines), with the enabled option **Snap to Geometry** when a translated component matches the alignment of another component in the flow.



*Illustration 41: The Flowaction menu added to Bio7*

the ruler (like it is known from text editors), components can easily be aligned to the visible ruler lines. The alignment will be supported by artificial lines (red lines) which occur when alignment criteria are met. The option **Grid** will enable a grid in the flow. When enabled, the components can be arranged in discrete coordinate steps according to the default grid size in the flow. Furthermore artificial lines will occur (green lines), with the enabled option **Snap to Geometry** when a translated component matches the alignment of another component in the flow.



*Illustration 42: Artificial lines of the Flow editor as a help for alignment*

*Table 18: From the GEF website (<http://www.eclipse.org/gef/>)*

### **Rulers and Guides**

GEF viewers can now display accessible rulers along their top and left edges.... Guides can be created on the ruler by clicking on an empty spot on the ruler... Parts in the graphical editor can be dragged and attached to the guides. They can also be attached when resizing. Alternatively, snapping can be disabled by holding down the Alt key while dragging. Feedback is shown in the form of a red line when a part is being attached to a guide.

### **Grid**

GEF now provides a grid. Parts can be snapped to a grid during creation, moving or resizing.... Alternatively, snapping can be disabled by holding down the Alt key while dragging. Grid snapping and visibility are two distinct properties, and it is possible to enable one without the other. Like the rulers, the grid can be turned off, if so desired.

### **Snap to Geometry**

The feature allows you to quickly align parts being dragged to other parts in the diagram. Edges of parts being resized can be snapped as well.....

### **Constrained Move and Resize**

Holding down the Shift key while moving a part will restrict that part's movement to one of the primary eight directions (N, S, E, W and the intermediaries). Doing the same while resizing will cause a proportionate resize: the height to width ratio will be maintained.

### **Centered Resize**

Holding down the Ctrl key while resizing will cause a resize to happen in the opposite direction of the drag as well, such that the center of the part being resized will not change after the resize is done.

### **Fly-out Palette**

For clients that require the palette to be always visible, and to provide further customizability to the end-user, the fly-out palette is also being introduced. The palette flies out over the top of the editor when the user hovers over it (or clicks on it), and auto-hides when not in use.

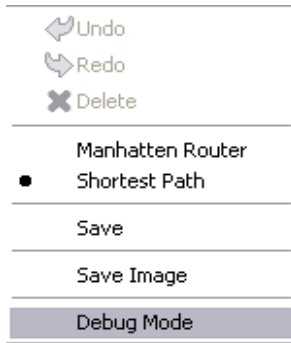
The user can also pin it open, if so desired.

It can be resized, docked on either side of the editor, and is completely accessible. The fly-out is only visible when the palette view is not. It automatically comes up when the palette view is closed (or when the user switches to a perspective in which the view is not open), and disappears when the palette view is opened. The palette state (selection, drawer expansion, etc.) is maintained when switching from the view to the fly-out and vice-versa.

### **Panning**

With the PanningSelectionTool, it is possible to pan by moving the mouse while holding down spacebar and mouse button 1 (left mouse button)....

### 7.4.3 The Context menu



*Illustration 43: The context menu of the Flow editor*

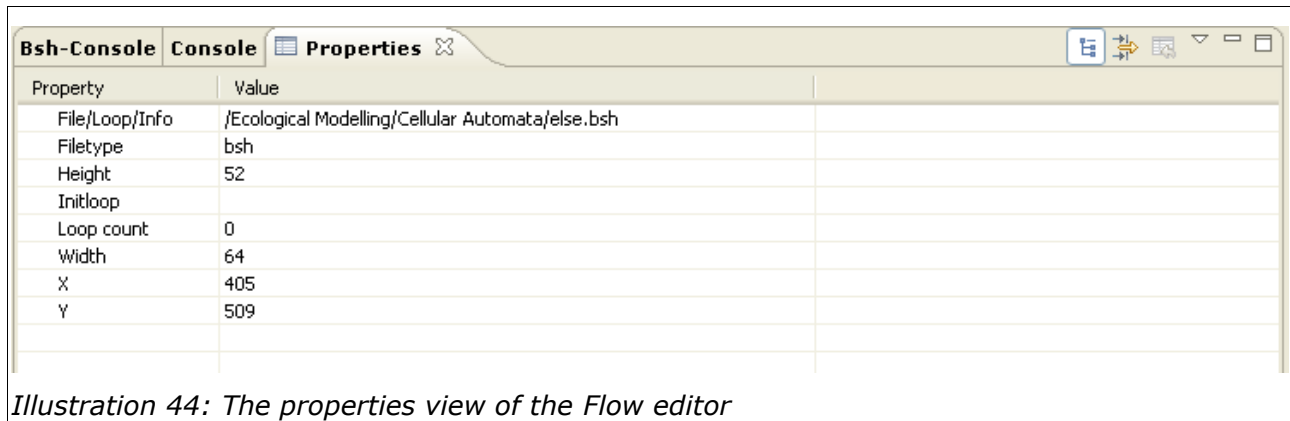
The context menu offers some of the mentioned methods and several new methods.

The new methods are:

1. The **Save** method will save the current flow to the before defined file (**Save as** in the Bio7 main menu available!).
2. The **Save Image** method will save an image in the formats \*.gif\*, \*.jpeg, \*.ico to a file. The current size of the flow will determine the pic size (the rulers display the size of the flow in pixel. So be cautious not to store large images with a resized flow !).
3. The following methods **Manhattan Router** and **Shortest Path** are responsible for the layout of the connections.
  - (a) The **Shortest Path** method applies the shortest path algorithm to find paths around obstacles in the flow.
  - (b) The **Manhattan Router** algorithm will use rectangular lines for connections in the flow.
4. The **Debug Mode** option enables a visualization of the current flow by bleaching the active shape to mark files that are currently interpreted or compiled, etc.

## 7.4.4 The Properties for the Components

If a shape or a connection is activated, the properties of the component or the connection can be adjusted in the properties view. This view can be activated in the **Windows ->Show View** dialog.

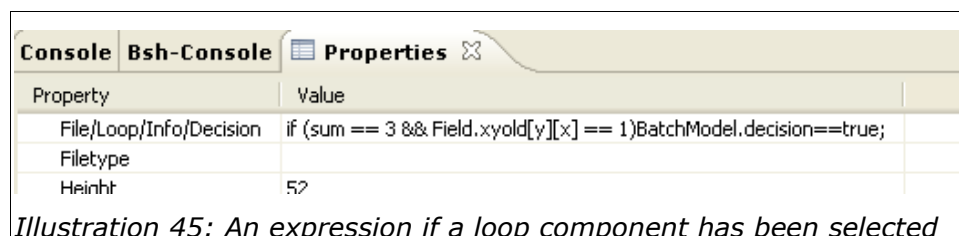


*Illustration 44: The properties view of the Flow editor*

The properties dialog will only be necessary, if you want create a decision inside a flow. All other component attributes are adjustable by clicking the mouse.

The **File/Loop/Info/Condition** property displays the following attributes in dependence of the selected component:

1. The filepath of a dropped component (relative path in the workspace. This path will be used for execution!)
2. The amount of loops which should be made when the loop component has been selected.
3. An info text if the info component has been selected.
4. When a decision component has been selected this property will be interpreted as a scripting expression in BeanShell.



*Illustration 45: An expression if a loop component has been selected*

5. The **Filetype** property shows the extension of the selected file, which has been dropped into the flow (has no effect on the other components).

- 6.** The properties **Height, Width, X, Y** display the size of the component and the location inside the flow panel, which can also be adjusted in the properties view.
- 7.** There are two properties for the connections.
  - (a)** The property **LineStyle** can be set to dashed or solid (by default the style is solid).
  - (b)** The **Boolean** property is important when a decision is to be created in the flow. The property can be set to true or false

Property	Value
Boolean	true
LineStyle	Solid

*Illustration 46: The properties of a connection*

**Note:** The properties File and Filetype are important for the execution of a file inside the Flow editor! The file type information is used to select the appropriate interpreter or compiler.

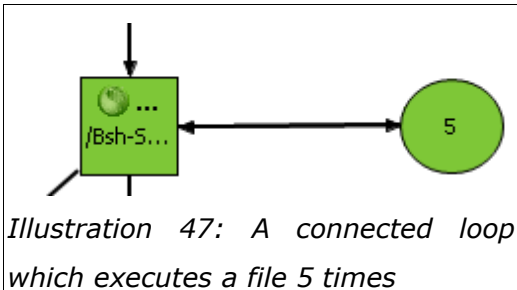
### 7.4.5 What to consider when creating a flow

It should be mentioned that some rules are important for the creation of a regular flow in the Flow editor.

- 1.** A start and a stop have to be placed in the flow and they have to be connected with the components for a regular Bio7 flow.
- 2.** A flow follows the outgoing direction (directed graph) marked with an outgoing arrowhead.
- 3.** Files:
  - (a)** A dropped file in the Flow editor (rectangular shape) can only be connected with another file or by means of a loop, which connects two files. Two ancestors (shapes) are allowed and necessary if a loop is connected to a file component.
  - (b)** A file component has to have one outgoing connection.
  - (c)** Files in the Flow editor will be executed sequentially by means of the information in the file property and the filetype property (the appropriate interpreter or program will be selected automatically).

#### 4. Loops:

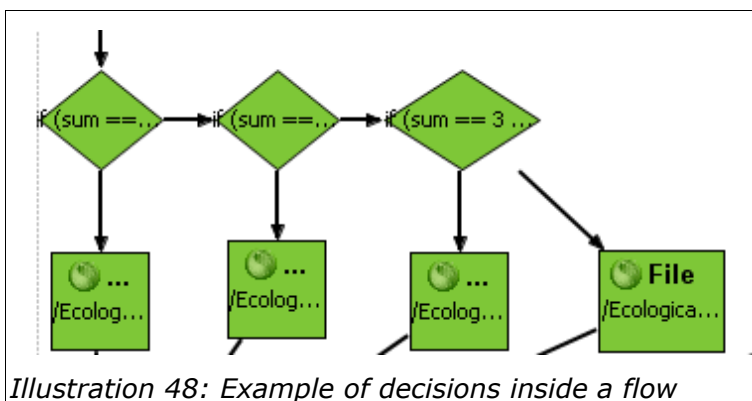
- (a)** A loop can have one incoming and one outgoing connection.
- (b)** A loop cannot be the first component (after the start) in a Bio7 flow.
- (c)** It is possible to execute a loop with one file component.



- (d)** Loops can be created nested. The loops are executed "iteratively" (not "recursive"!).
- (e)** A loop can be connected (point) to a decision.

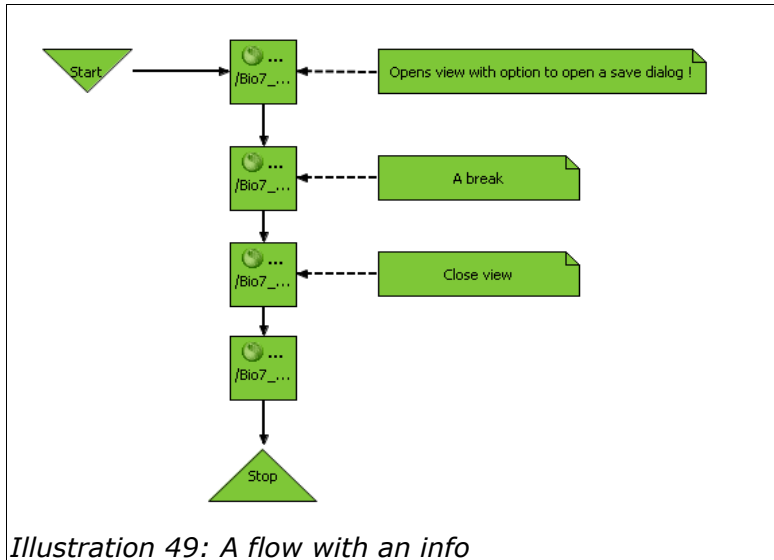
#### 5. Decisions:

- (a)** A decision gets an input from a component and has to have two outgoing connections. One connection has to be set to false and the other has to be set to true.
- (b)** A decision can be connected to another decision (else if).
- (c)** A decision cannot be connected (point) directly to a loop.
- (d)** It is allowed in Bio7 to place a decision right after the start.



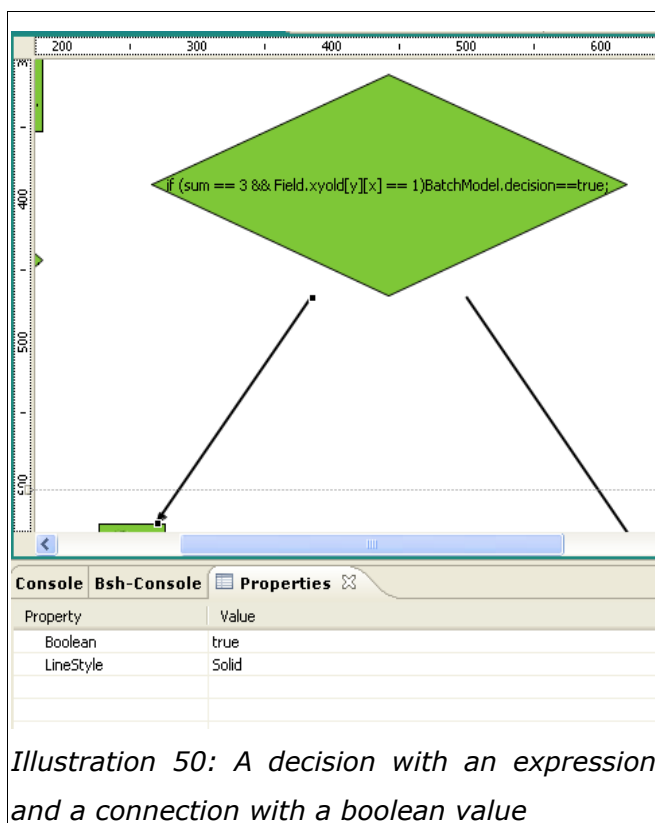


(e) The condition of a decision component can be set directly in the properties view or in the component itself. The static variable **decision** has to be set to "true" or "false" (type: String) to follow a connection with the appropriate value -> (**BatchModel.setDecision("true");**).



## 6. Info:

An info component can be placed outside the connected flow for an information. It is also allowed to draw connections to a file component as a pointer (you can mark this connection with a changed line style).



## 7.4.6 How to Cancel a flow

A loop can be cancelled with the **Cancel** button in the progress dialog (the dialog is only displayed for long operations) or by selecting the key combination **CTRL+X** (short and long operations). In some cases where the application Rserve is involved it can become necessary to stop and restart the application (this can be done with the Rserve connection action in the toolbar of Bio7). If possible it is highly recommend to test all files before they are used in the Flow editor.

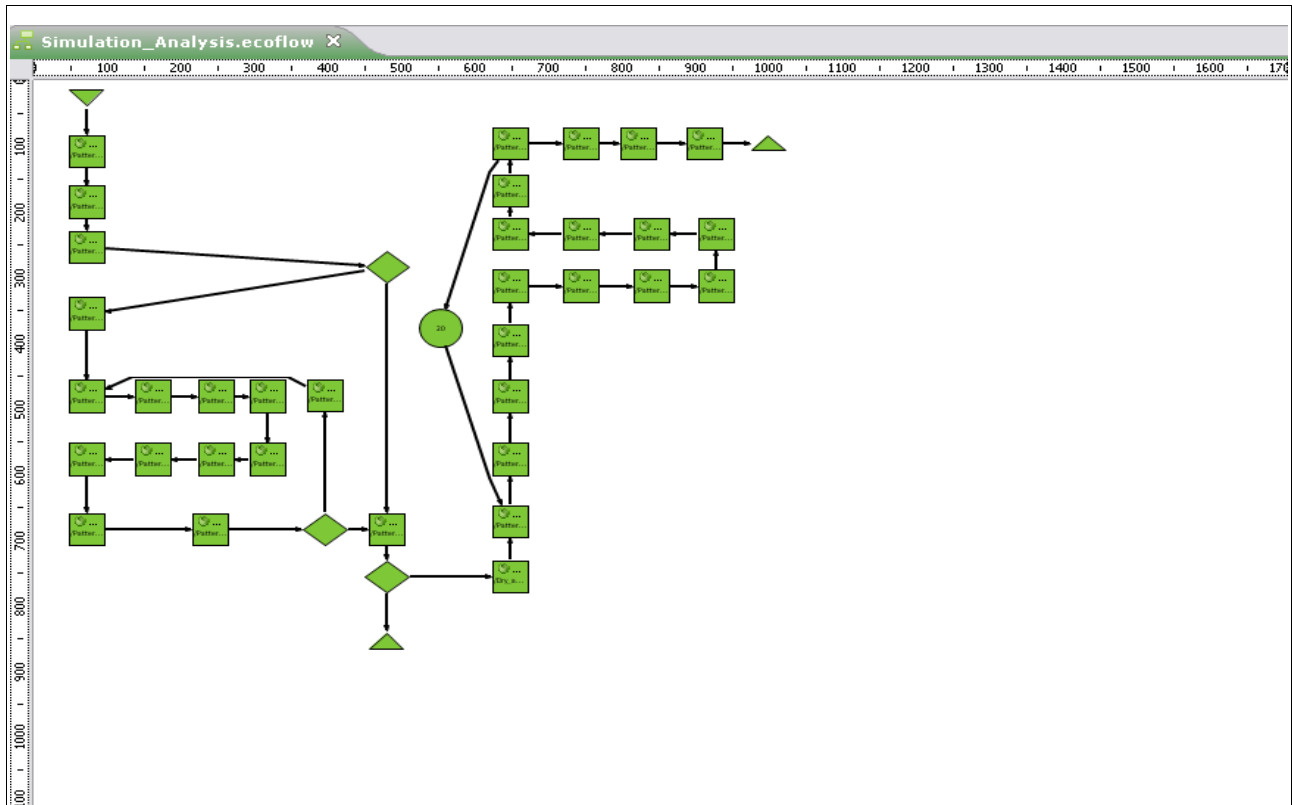


Illustration 51: A complete defined flow in the Flow editor

## 7.4.7 Executed Files

### 7.4.7.1 Dropped from the Navigator

The Flow editor recognizes several different file types. Files, created with the Bio7 editors, will be executed in the flow with the appropriate interpreter or the embedded compiler (\*.bsh, \*.java, \*.ecor). Additionally the Flow editor will also execute \*.exe files. Files with the extension \*.argument can be created and dropped before the \*.exe file in the flow sequence. This will execute the dropped program with an argument (xxx -argument).

### 7.4.7.2 Dropped from outside the Navigator

Files, dropped from outside the navigator, will be opened or executed with the registered program of the operating system !

Table 19: Action according to location and extension

File location	Extension	Action
Navigator	*.bsh	Interpreting (BeanShell)
Navigator	*.java	Compiling (Java Compiler)
Navigator	*.ecor	Interpreting (R)
Navigator	*.txt	Executes an ImageJ Macro
Navigator	*.pdf	Opens a pdf-file
Navigator	*.bat	Executes a batch file (Windows)
Navigator	*.sh	Executes a shell file (Linux)
Navigator	*.argument	Assignment to BatchModel.argument (String)
External	*.exe	Execution (with argument when available !)
External	*.xxx	Execution of the registered program

### 7.4.8 Important to know

The dropped file components in the flow can be opened and edited with a double-click on that component. If a flow is executed, a modal dialog will appear each time a file is executed. Please select the **General->Always run in background** option in the preferences dialog to avoid this. A static variable is available to trigger the cancellation (This can be used, for example, in swt shells). See the Bio7 API for more methods (Class: Bio7Flow).

Table 20: Code snippet

```
Bio7Flow.setCancel(true);
```

## 8 The Preferences

The preferences dialog in Bio7 is divided in the preferences for the platform (Generals, Help, Install/Update) and the Bio7 Preferences. Each of the preferences are organized hierarchically in a tree viewer structure for a better overview.

Table 21: From Eclipse 3.2

The **Preferences** dialog is the dialog used to set user preferences. The Preferences dialog pages can be searched using the filter function. To filter by matching the page title, simply type the name of the page you are seeking and the available pages will be presented below. The filter also searches on keywords such as appearance and Java. The history controls allow you to navigate through previously viewed pages. To step back or forward several pages at a time, click the drop down arrow and a list of the most recently viewed preference pages will appear.

The Preferences dialog can be found from the main workbench **Window** menu under **Window > Preferences**. Preference pages contributed by plug-ins will be found in this dialog.

### 8.1 The Bio7 Preferences

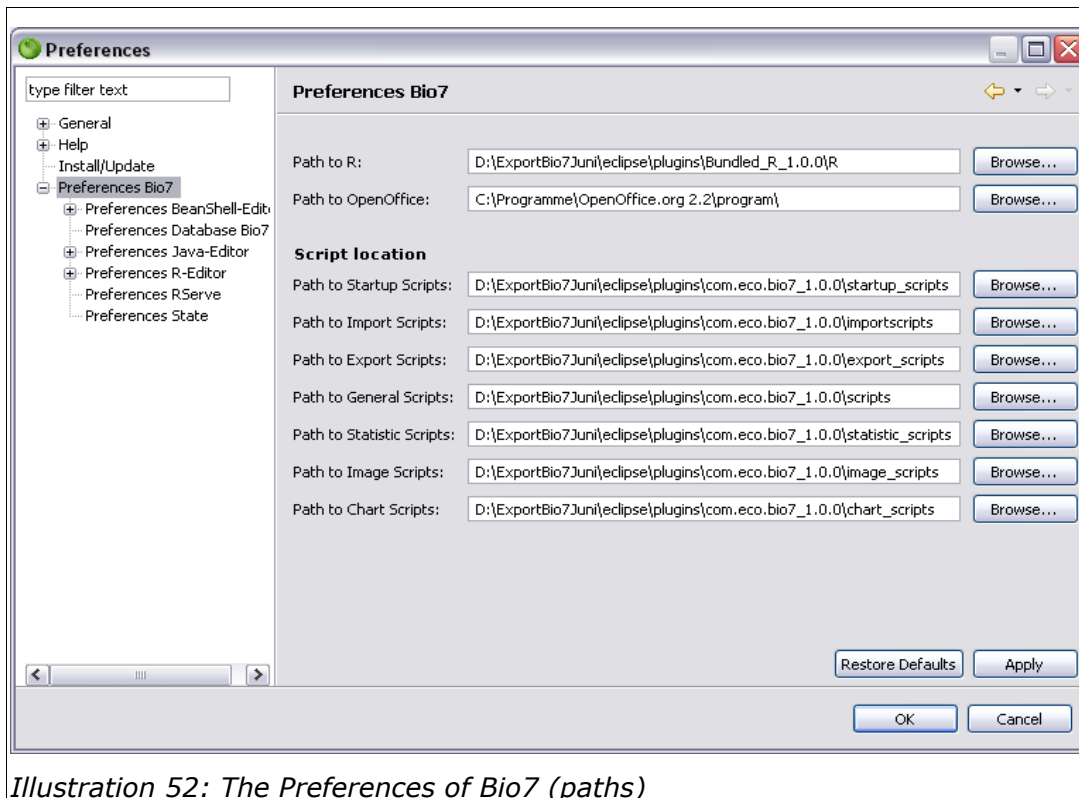


Illustration 52: The Preferences of Bio7 (paths)

### 8.1.1 Application Paths

If the topmost item in the Bio7 Preferences dialog is selected, several text fields are displayed which store different paths for the Bio7 application (**Preferences Bio7**). The two topmost paths (**Path to R, Path to OpenOffice**) are paths for the R application (embedded) and Open Office (if installed !), which will be automatically fetched from the registry (Windows) when Bio7 starts. They can also be adjusted manually to any location in the file system, where R or OpenOffice are installed (**Browse...** button). It should be mentioned that the path to R can be changed to any R installation. The communication between Bio7 and R will be successful, if the Rserve application is installed correctly in that location.

### 8.1.2 Script Paths

The following paths in the Preferences Bio7 are script path locations where scripts and macros (ImageJ macros) can be dropped which update the menu (except Startup Scripts!) of the Bio7 application at runtime (see Section 14.12). The import- and export scripts will update the file menu. The other scripts and macros (General, Statistic, Image, Chart) will update the scripts menu in Bio7. The locations of the scripts can be browsed easily with the browse action and files can be transferred simply by copy and paste from the Navigator. The files will be detected automatically and displayed in the appropriate menu for execution.

### 8.1.3 Bio7 Editor Preferences

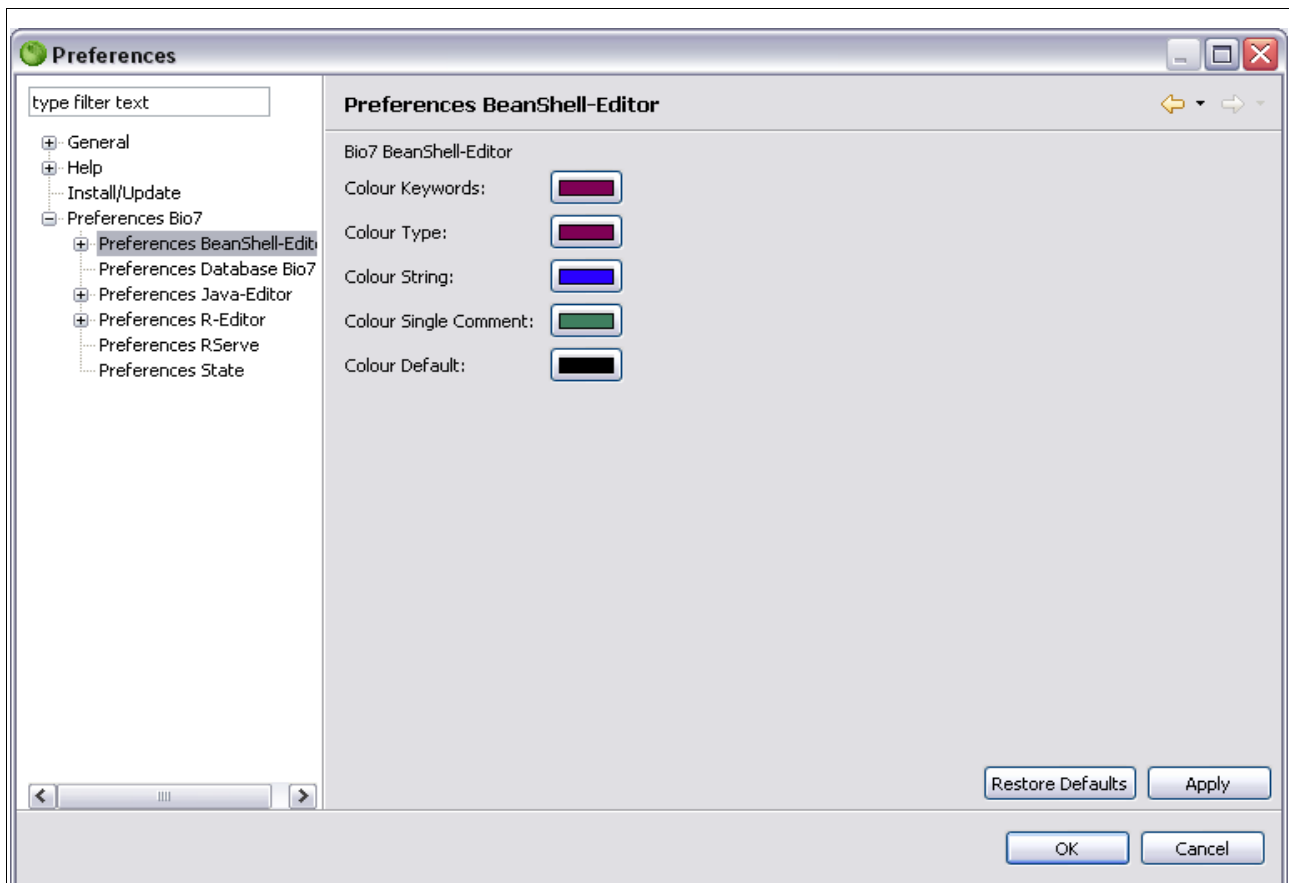
The Preferences of the different language editors in Bio7 (R, BeanShell, Java) offering methods to adjust the default language and colours of the editors as well as to create or change templates for them.

**Preferences Bio7->\*Preferences *BeanShell-Editor*** (\*or Java-Editor, R-Editor)  
for changing the default colours.

**Preferences Bio7->Preferences *BeanShell-Editor*->\*Template Editor *BeanShell*** (\*or Template Editor Java, Template Editor R) for modification or creation of the templates.

#### **Note:**

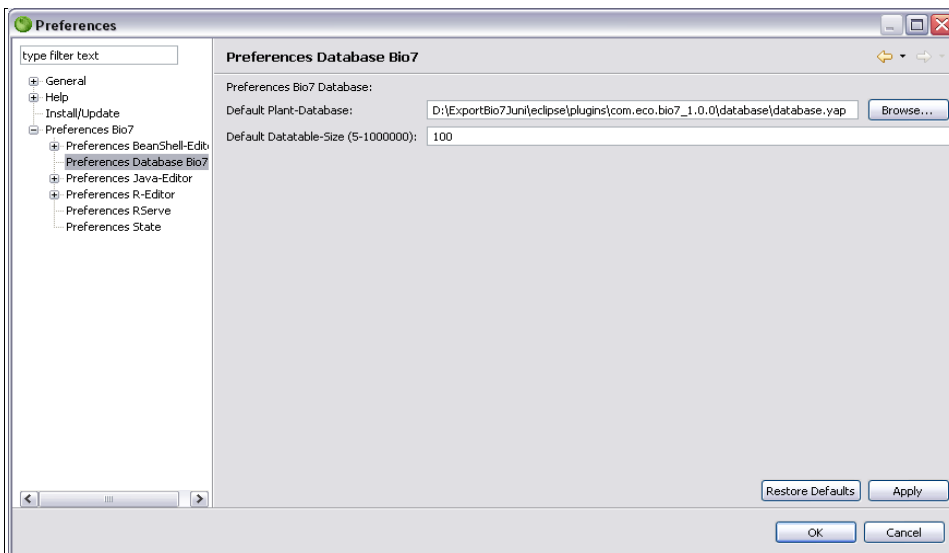
It should be mentioned that some aspects of the language editors can be changed in the default text preferences of the platform (e.g. font attributes).



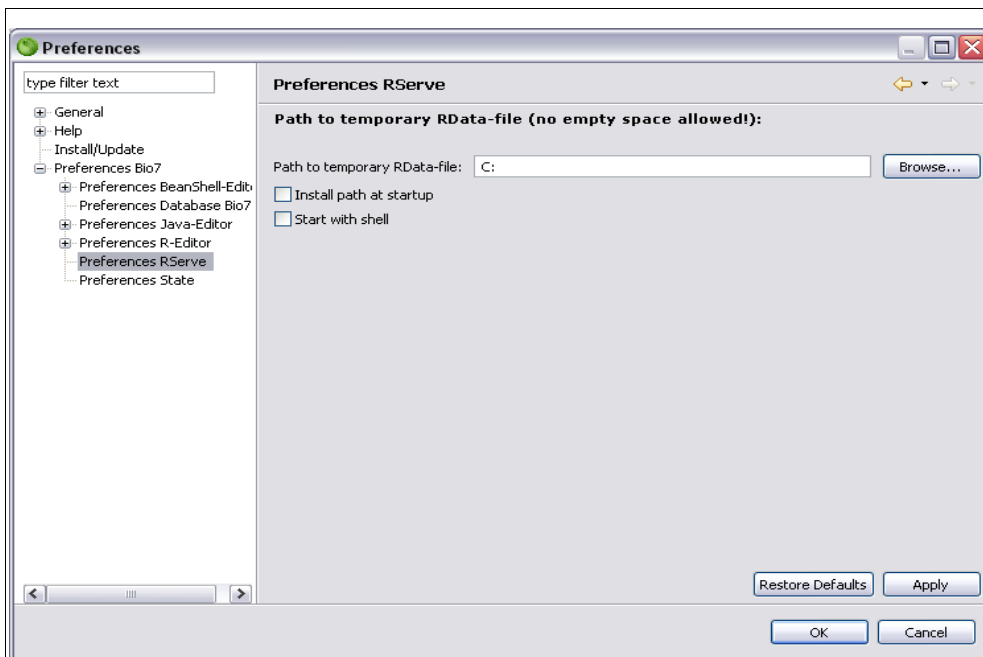
*Illustration 53: Colour preferences of the BeanShell editor*

#### 8.1.4 Preferences Database

In the preferences of the database (**Preferences Bio7->Preferences Database**) the default location of the database is displayed in a textfield. With the textfield for the **Default Datable-Size** the size of the table, which displays the plants (e.g in the Database perspective), can be changed according to the size of the self-defined database. **The changes occur after the Bio7 application has been restarted!** Normally it is not necessary to do so, because it is sometimes not plausible to do simulations >100 plants. But if you want to build up a database for different simulations the backend for this application is a full featured object oriented database (dbo4). If you want to backup the database simply browse to the database file (database.yap) and copy it elsewhere for storage (Also databases can be exchanged for different analysis!).



*Illustration 54: Preferences Database of Bio7*



*Illustration 55: Preferences Rserve*

### 8.1.5 Preferences Rserve

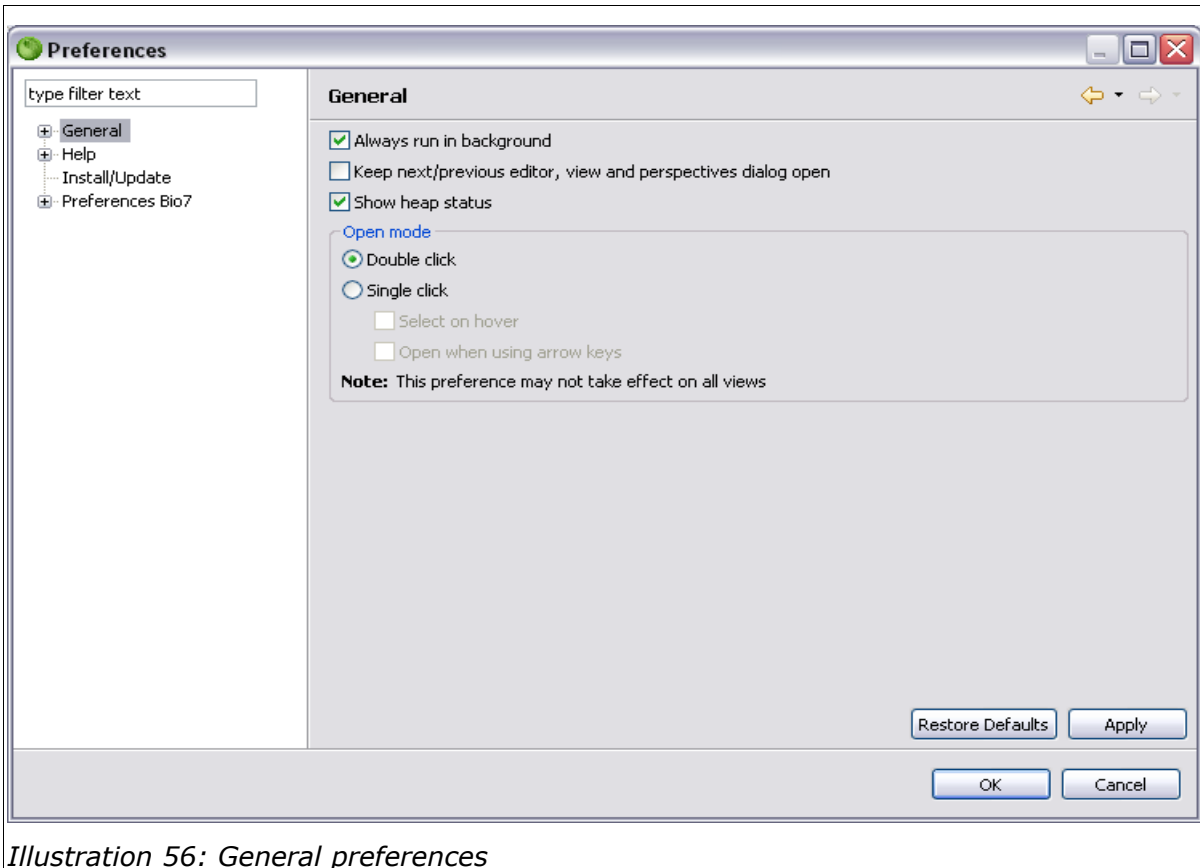
The preferences for the Rserve application (**Preferences Bio7->Preferences Rserve**) offers three options. The first option is an adjustable path to a temporary R file, which will be stored and opened when the RGui application is called (in the R menu) . It is important that the path to the temporary file has no empty space (due to some Rserve restrictions of the path). By default the location is set to **C:\** (Windows). The second option installs a registry path for the embedded R(only windows!) at startup. If selected, the third option starts the Rserve application with an opened shell.

## 8.2 Default Platform Preferences

A detailed description of the preferences of the platform can be found on <http://help.eclipse.org/>.

Only some important preferences for the Bio7 platform are described here. Notes and additions regarding Bio7 are marked red.

### 8.2.1 General



*Illustration 56: General preferences*

General settings for the Workbench. The term *Workbench* refers to the desktop development environment. Each Workbench window contains one or more perspectives. Perspectives contain views and editors and control what appears in certain menus and tool bars. More than one Workbench window can exist on the desktop at any given time.

The following preferences can be changed on the **General** preference page.



Table 22: *From the Eclipse Workbench User Guide (<http://help.eclipse.org/>)*

Option	Description	Default
Always run in background	Turn this option on to perform long running operations in the background without blocking you from doing other work. <b>Please turn on for Bio7 !</b>	Off
Keep next/previous part dialog open	If this option is turned on then the editor and view cycle dialogs will remain open when their activation key is let go. Normally the dialog closes as soon as the key combination is release.	Off
Show Heap Status	Turn this option on to display an indicator showing information about current Java heap usage.	Off
Open mode...	You can select one of the following methods for opening resources: <ul style="list-style-type: none"> <li>• Double click - Single clicking on a resource will select it and double clicking on it will open it in an editor.</li> <li>• Single click (Select on hover) - Hovering the mouse cursor over the resource will select it and clicking on it once will open it in an editor.</li> <li>• Single click (Open when using arrow keys) - Selecting a resource with the arrow keys will open it in an editor.</li> </ul> <p><b>Note:</b> Depending on which view has focus, selecting and opening a resource may have different behavior.</p>	Double click

## 8.2.2 Colors and Fonts

Many of the fonts and colors as used by eclipse components can be set using the **General > Appearance > Colors and Fonts** preference page.

A tree is used to navigate among and show a short preview of the various colors and fonts. The current face (but not size) of any font is previewed in its label. Colors are previewed in the icon associated with its label. Additionally, some categories (Workbench in particular) provide a more detailed preview of their contributions. This preview is shown below the description area if available. Font settings can be changed either by selecting the font from the list and clicking **Use System Font** to choose the Operating System font setting or by clicking **Change** to open up a font selection dialog. **Reset** can be used to return to the default value.

Color settings can be changed by clicking **color** to the right of the tree area when a color is selected. **Reset** can be used to return to the default value.

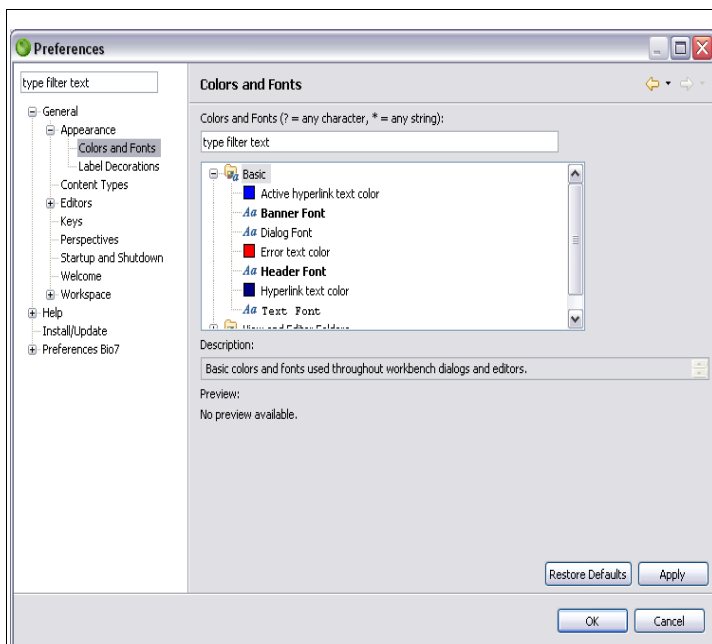


Illustration 57: Preferences for the Text in Bio7

### 8.2.3 Text Editors Preference Page

The following preferences can be changed on the **General > Editors > Text Editors** page.

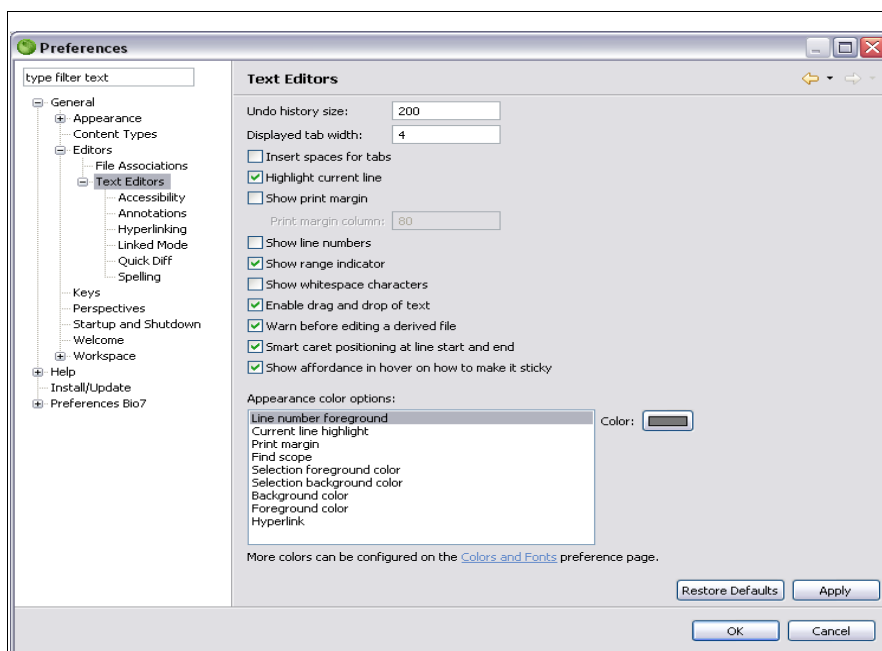


Illustration 58: Appearance options

Table 23: *Appearance: From the Eclipse Workbench User Guide (<http://help.eclipse.org/>)*

Option	Description	Default
Displayed tab width	This option allows you to set the displayed tab width for the text editor.	4
Undo history size	This option allows you to undo the history size for the text editor.	25
Highlight current line	This option controls whether or the current line is highlighted or not.	On
Show print margin	This option controls whether the print margin is visible or not.	Off
Show line numbers	This option controls whether or not line numbers are shown on the left side of the text editor.	Off
Show range indicators	This option controls whether or not range indicators are shown in the text editor.	On
Support hyperlink style navigation	This option controls whether or not hyperlink style navigation is supported.	On
Hyperlink style navigation key modifier	This option sets the hyperlink style navigation key modifier.	Ctrl
Disable overwrite typing mode	This option controls whether the overwrite typing mode is enabled or disabled.	Off
Appearance color options	This option controls various appearance colors.	

## 8.2.4 Help preferences

On the **Help** preferences page, you can indicate how to display help information.

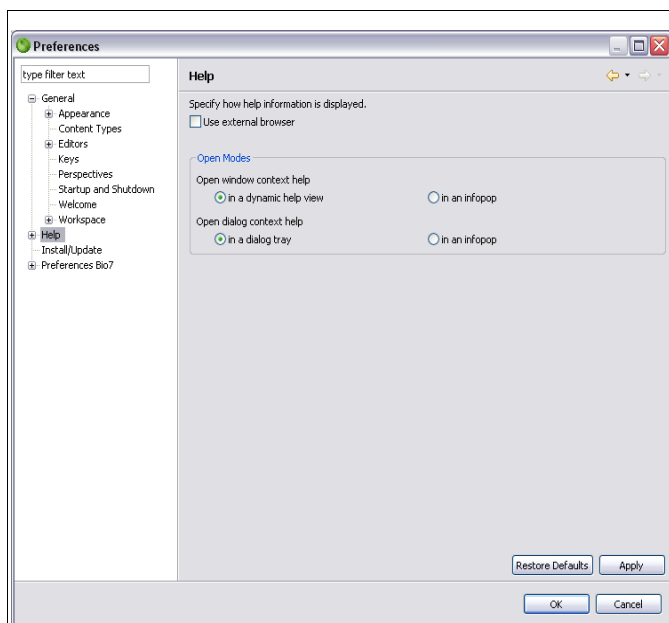


Illustration 59: Help preferences

Table 24: From the Eclipse Workbench User Guide (<http://help.eclipse.org/>)

Option	Description	Default
Use external browsers	If embedded web browser is supported on your system, help window uses an embedded help browser to display help contents, whenever possible, and this option is available. Select it, to force help to use external browsers. Use "Web Browser" preference page to select browser to use.	Off
Open window context help	This option allows you to determine whether the window context help will be opened in a dynamic help view or in an infopop. <b>In Bio7 press F1 to activate the context help !</b>	in a dynamic help view
Open dialog context help	This option allows you to determine whether the dialog context help will be opened in a dynamic help section of help view or in an infopop.	in dialog tray
Help view document open mode	This option allows you to determine whether the documents selected in the help view will be opened in place or in the editor area.	Open in place
Search	Determines whether potential hits should be shown while searching. Showing potential hits will increase search performance, at the cost of potential loss of accuracy.	Show all potential hits (faster)

## 8.2.5 Perspectives

On the **General > Perspectives** preference page, you can manage the various perspectives defined in the Workbench.

Table 25: From the Eclipse Workbench User Guide (<http://help.eclipse.org/>)

Option	Description	Default
Open a new perspective	Use this option to set what happens when you open a new perspective. Do you want the perspective opened within the current Workbench window or opened in a new window?	In the same window (leave this option as it is in Bio7 !)
Open a new view	Use this option to specify what happens when a new view is opened. It is either opened to its default position within the current perspective or it is opened as a fast view and docked to the side of the current perspective.	Within the perspective

Option	Description	Default
New project options	Use this option to specify the perspective behaviour when a new project is created. You can set it to switch the current perspective to be the one associated with the project type and open the perspective in the same Workbench window as the current one, switch the perspective and open it in a new Workbench window, or not to switch perspectives at all.	Open perspective in the same window

Table 26: Available Persp.:From the Eclipse Workbench User Guide (<http://help.eclipse.org/>)

Option	Description	Default
Make Default	Sets the selected perspective as the default perspective.	Resource
Reset	Resets the definition of the selected perspective to the default configuration. This option is only applicable to built-in perspectives that have been overwritten using Window > Save Perspective As...	n/a
Delete	Deletes the selected perspective. This option is only applicable to user-defined perspectives (built-in perspectives can not be deleted).	n/a

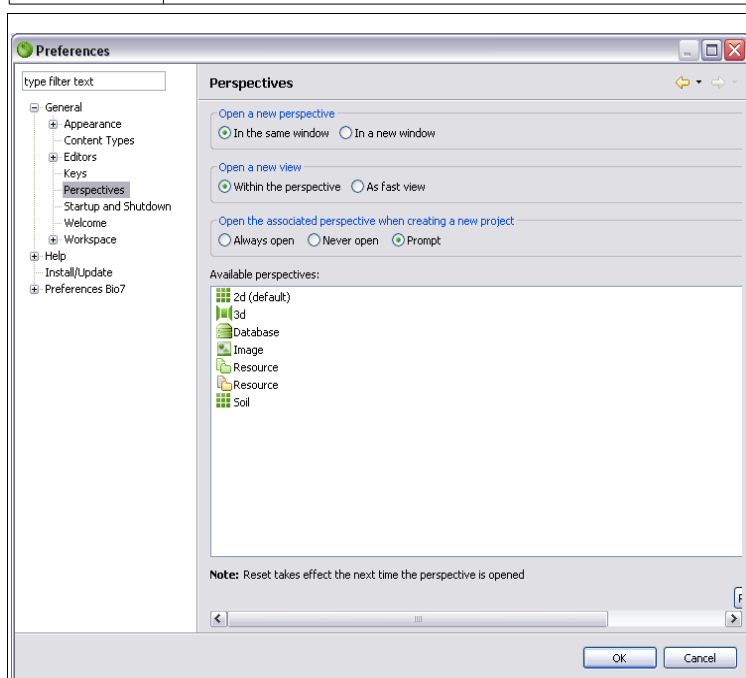


Illustration 60: Perspective options of Bio7

## 8.2.6 File Associations

Table 27: From the Eclipse Workbench User Guide (<http://help.eclipse.org/>)

On the **[General > Editors > File Associations](#)** preference page, you can add or remove file types recognized by the Workbench. You can also associate editors with file types in the file types list.

### File types list

- **Add...:** Adds a new file or file type (extension) to the predefined list. In the resulting New File Type dialog, type the name of a file or a file extension. If you are adding a file extension, you must type either a dot or a "\*" before the file type (e.g. ".xml" or "\*.xml" as opposed to simply "xml").
- **Remove:** Removes the selected file type from the list

### Associated editors list

- **Add...:** Adds a new editor to the list of editors associated with the file type selected above. In the resulting Editor Selection dialog, you can choose an editor to launch either inside the Workbench (internal) or outside the Workbench (external); click **Browse** to locate an editor yourself if the editor you want is not displayed in the list.
- **Remove:** Removes the association between an editor and the file type selected above.
- **Note:** Any editor that is bound by content type may not be removed from this list. Currently, there is no mechanism available to remove these editors.
- **Default:** Sets the selected editor as the default editor for the file type selected above. The editor moves to the top of the Associated Editors list to indicate that it is the default editor for that file type.

For more information about the preferences of the platform please read the Workbench User Guide (Eclipse platform)!

## 9 Help

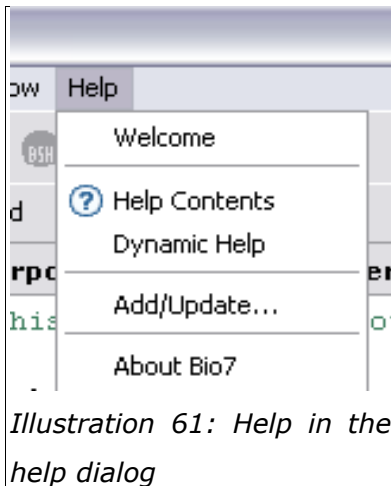


Illustration 61: Help in the help dialog

For a first help simply activate a **Dynamic Help** (selecting a view and activating the F1 key). After activation a context help will be shown in the help view (can be changed to an infopop in the preferences) if activated. By selecting the submenu **Help Contents** the regular help will be opened. The **Welcome** action opens a welcome page for the Bio7 application. The **About Bio7** dialog opens an information about the current release of Bio7.

## 10 Installing new features with the update manager

Table 28: From the Eclipse Workbench User Guide (<http://help.eclipse.org/>)

To locate and install a new feature into a product (ordinarily requires web access):

1. Open the Install Wizard by clicking **Help -> Add/Update...**. This opens the wizard.
2. Select the second button, "**Search for new features to install**" and click **Next**.
3. Create a bookmark for an update site where Eclipse features and plug-ins are published. In the sites to search list, select **Add Update Site** to add a remote site, or **Add Local Site** if the site is available on a local drive (including a CD), or **Add Archived Site**, when the site is available locally but is packaged as a jar or zip file.
4. In the add site dialog, give the site a name such as "Company A" and enter the URL such as "http://companyA.example.com/eclipseupdates".
5. After adding the site, expand it to the categories of feature versions available at that update site. This will contact the web site to discover what features are available.
6. Select the categories you want to search and click **Next**.
7. Wait for the search to finish and selecting the features to be added. You can view the description or more detailed properties for any feature by selecting the feature and

pressing the "Properties" button.

- 8.** Once you're decided which features to install, click **Next**.
- 9.** Carefully review the license agreements for the features. If the terms of all these licenses are acceptable, check "I accept the terms in the license agreements". Do not proceed to download the features if the license terms are not acceptable.
- 10.** If a feature selected for install include optional features, a page will show up allowing you to select whether you want them installed or not. Optional features typically carry functionality that is not essential for proper functioning of the main feature.
- 11.** The Install Location page controls where the new feature's files are to be installed on the local computer. Select the directory into which the product is installed and hit **Next**. (If the product is installed in a directory to which you do not have write access, you should contact your system administrator and get them to install this feature so that it will be generally available. The other option is to click **Add** and point to a directory to which you do have write access.
- 12.** Feature versions can be digitally signed by the company that provides them. This allows you to verify more easily that the features and plug-ins that are about to be downloaded and installed are coming from a trusted supplier.

**Warning:** Because of the possibility of harmful or even malicious plug-ins, you should only download features from parties that you trust.

Click **Install** to allow the downloading and installing to proceed.

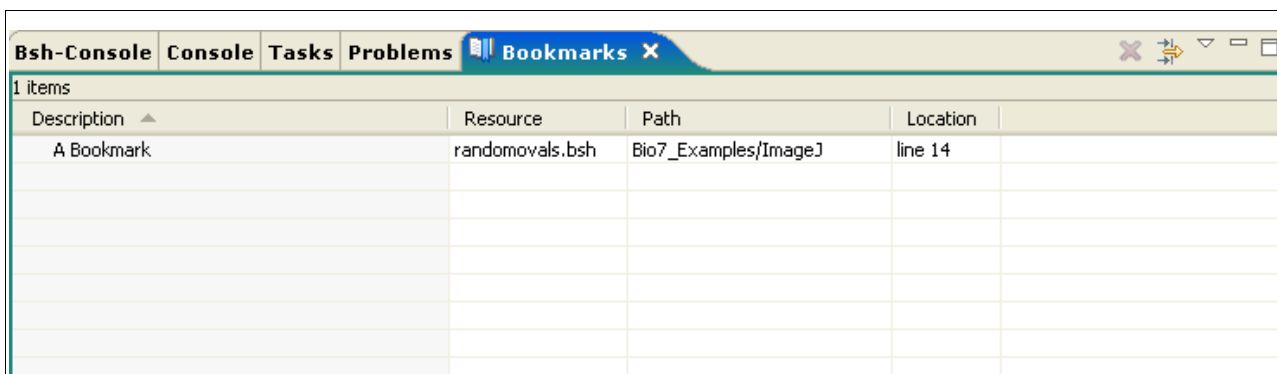
Once the new feature and plug-ins have been downloaded successfully and their files installed into the product on the local computer, a new configuration that incorporates these features and plug-ins will be formulated. Click **Yes** when asked to exit and restart the workbench for the changes to take effect. To add other new features at the same time before restarting, click **No** and repeat. If this was a new feature and can be dynamically installed into the current configuration, you will also be presented with a **"Apply Now"** button. This will not restart the platform, but configures the feature and the plug-ins into the current configuration.



## 11 Additional views in Bio7

By default some more views are available in Bio7:

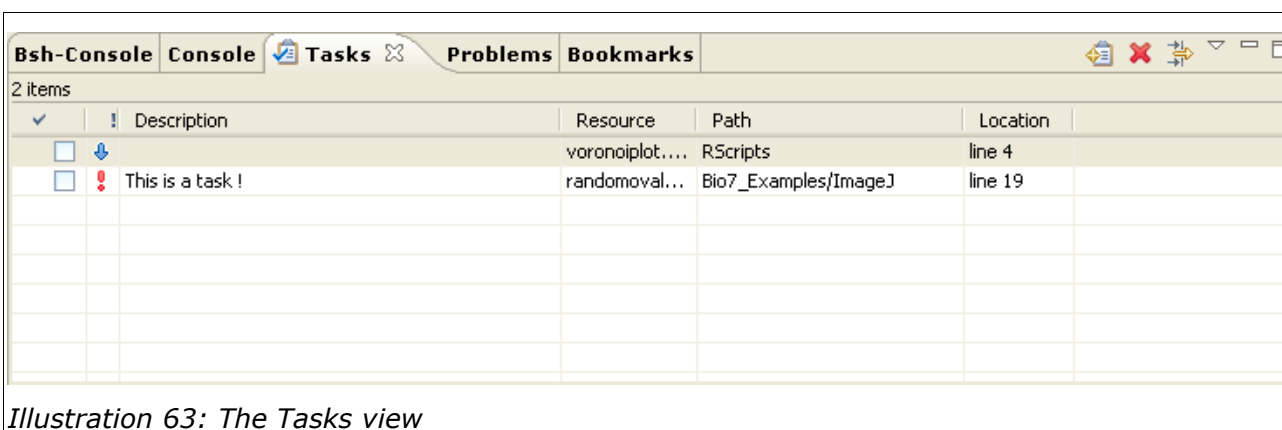
The **Bookmarks** view is a view which shows marked bookmarks in all language editors of Bio7 for easily find points in code which have to be edited.



The screenshot shows the Bio7 interface with the 'Bookmarks' tab selected. It displays a table with the following data:

Description	Resource	Path	Location
A Bookmark	randomovals.bsh	Bio7_Examples/ImageJ	line 14

Illustration 62: The Bookmarks view



The screenshot shows the Bio7 interface with the 'Tasks' tab selected. It displays a table with the following data:

Description	Resource	Path	Location
voronoiplo... ↓	voronoiplo...	RScripts	line 4
! This is a task !	randomoval...	Bio7_Examples/ImageJ	line 19

Illustration 63: The Tasks view

The **Tasks** view in the R-Editor serves as a marker for plots in the source code. For all other editors it can be used to leave some information about tasks, that have to be done or as information in the Bio7 language editors (They can be marked with different priorities!).

The **Problems** view has no function in Bio7 at the moment, but it can be a useful resource for later developments.

The **Cheat Sheets** view can be used to load created Cheat Sheets from Eclipse.

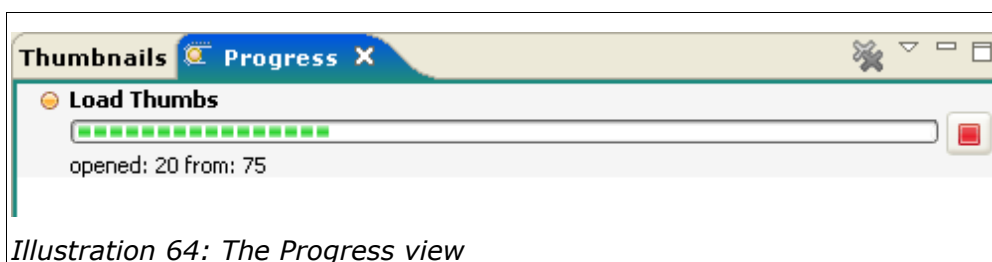
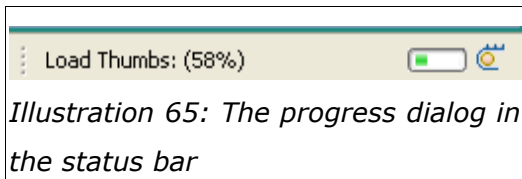


Illustration 64: The Progress view

The **Progress** view shows information about running jobs in the application. Each time a job has started, this progress view can be opened in the progress dialog in the status bar.

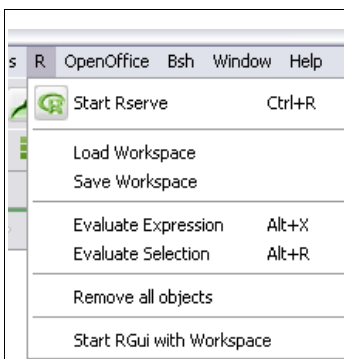


*Illustration 65: The progress dialog in the status bar*

## 12 External tools

### 12.1 R features of Bio7

Bio7 embeds R as a statistical application to do profound statistical analysis of ecological models. To interact with the Bio7 application and all of its components the Rserve connection has to be started. In a start process the Bio7 app. tries to connect to the Server (displayed in the progress dialog). If the connection trials exceed 10 trials (1/second) the operation will be aborted. After the connection has been established several methods for the interaction with R are available. By using the R editor (see section 7.2) RScripts can easily be interpreted. Additionally some specialized methods are available in the R menu of Bio7.



*Illustration 66: R menu of Bio7*

- 1.** With the **Load Workspace** action R-Workspaces can be loaded and used for simulation purposes in Bio7.
- 2.** The **Save Workspace** action saves the current workspace, which has been established with the Rserver.
- 3.** **Evaluate Expression** opens a dialog where R expression can be evaluated. The evaluated expression will be displayed in the Console of the Bio7 platform.

4. **Evaluate Selection** evaluates the selected line in the R editor and then jumps to the next line.
5. Since Bio7 works with one connection to R, sometimes it is necessary to delete all variables from the current workspace. The action **Remove all objects** removes all objects from the current workspace.
6. Because sometimes capabilities of Bio7 in respect to R do not fulfill the needs for a direct evaluation (for example, plotting, error analysis) it is possible to start a complete standard RGui (Windows) with the current workspace (a temporary workspace will be created and then opened with the RGui.  
This will be invoked with the **Start RGui with Workspace** action. (The path to the temporary file can be adjusted in the Preferences of Rserve).
7. Another feature is a direct interaction with OpenOffice (if installed!), where values easily can be transferred to the current workspace of R (see OpenOffice features).

Additionally a nice feature in Bio7 is the direct interaction with the Rserve interpreter. Not only for evaluation, but also to obtain values from R to Java or BeanShell is valuable for the creation of ecological models. Not only a call to a R method is possible by means of compiled code, but also calls from the BeanShell interpreter can be realized to transfer values to a BeanShell workspace.

*Table 29: Code Snippet: Transfer values from R to Beanshell*

```
/* This Snippet is an easy example how to get variables
from R after a calculation !*/

Rserve.getConnection().eval("A<-c(3,5,3,6)");
Rserve.getConnection().eval("B<-c(3,6,4,5)");
Rserve.getConnection().eval("C<-A+B");

//Get the result from R!
double [] result =(double[])Rserve.getConnection().eval("summary(A)").getContent();

RServe.print("C"); //Print to console !

// Print with Java code!
for (int i = 0; i < result.length; i++) {
    System.out.print(result[i]+" ");
}
```

Because the bidirectional transfer of data in between, for example, R and BeanShell or Java is possible, the following interaction methods can be done.

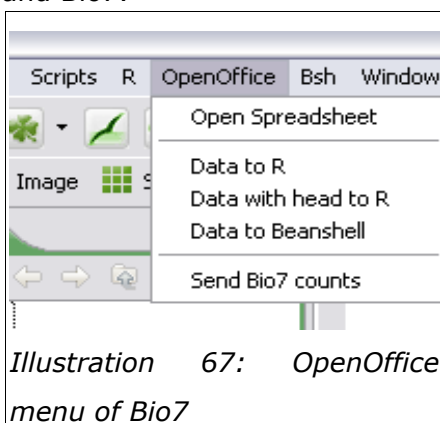
1. Custom GUI's can be created in BeanShell and the values can be evaluated in R. After evaluation the results can be sent back to the BeanShell "namespace" or to a method in Java.
2. Build frontends for R methods with created GUI's in BeanShell.
3. Interaction between R and the API of ImageJ to do profound Image Analysis and statistical evaluation with BeanShell or by compiled code.
4. Extend the script menus of Bio7 with R methods called by BeanShell.
5. Creation of a complete flow from easy data transfer (OpenOffice) over evaluation (RScript) up to data visualization (Also possible in one file with BeanShell).
6. The Flow editor can be used to arrange a flow of interacting files which guides the user through a complex process of statistical analysis.

See the Code Snippets for more examples.

## 12.2 OpenOffice features of Bio7

If the free Office Software OpenOffice is installed, some methods for data acquisition and a small API for interaction with the OpenOffice spreadsheet are available in Bio7.

To use the OpenOffice features it is necessary to establish a connection between OpenOffice and Bio7.



This will be done automatically when selecting the action "OpenOffice connection" in the toolbar of Bio7. If a connection is established, several methods for the interaction with OpenOffice will be available in the OpenOffice menu.

1. To receive or transfer values from or to the spreadsheet of OpenOffice, it is necessary to open a spreadsheet. This can easily be done with the action **Open Spreadsheet**. After a spreadsheet has been opened (also possible by means of a small API) the spread can be filled with values by hand or with values from a file (in the opened spreadsheet menu of OpenOffice: **Insert->Sheet from file**).
2. With the action **Data to R** OpenOffice spreadsheet values can be transferred to the active R workspace by means of the Rserve connection (the columns will be stored as lists and the selected values as a dataframe).
3. With the action **Data with head to R** numeric values (type double) with a head can be transferred directly to R.
4. With the menu **Data to BeanShell**, values will be transferred to the workspace of BeanShell (will be stored in a static double array ->**OpenOffice.sheetdata**).
5. With the action **Send Bio7 counts** counted values of the activated states in a simulation for all time steps are transferred to an opened spreadsheet which then can be stored to all kinds of data files.

There is also a small API available for the transfer of data (get and set), inserting new sheets, getting values from sheets and the marking of the data in a spreadsheet (see Snippets and API).

*Table 30: Example to set values programmatically*

```
OpenOffice.openSpread();// Open a new spreadsheet!

String name="Customsheet";
OpenOffice.newSheet(name);// Insert a new sheet!

String []head={"A","B","C","D"};
double [][]values=new double[head.length][40];// Defined 4 columns!

for (int i = 0; i < values.length; i++) {
    for (int u = 0; u < values[0].length; u++) {
```

```

        values[i][u]=Math.random()*150;

    }
}

OpenOffice.sheetInsertValues(values,head); // Insert all!
OpenOffice.activateSheet(name);// Activate and bring to front!

```

## 12.3 ImageJ features of Bio7

Also ImageJ offers a vast amount of methods in a clean accessible API. With the methods of ImageJ it is possible to make statistical image-analysis, shape analysis, pattern analysis, particle count, etc. All kind of methods are available. Furthermore in the combination with R and Bio7 the strong image methods can help detecting patterns and offer a more flexible support for the analyses of spatial patterns in time and space on different scales. Additionally, with the powerful libraries from Java everything can be combined for a thorough analysis of ecological models.

*Table 31: Example from the ImageJ Website as a BeanShell-Script*

```

import ij.*;
import ij.process.*;
import ij.gui.*;
import java.awt.*;
import java.util.Random;
import ij.plugin.*;

public class Random_Ovals implements PlugIn {

    public void run(String arg) {
        IJ.run("New...", "name='Random Ovals' type='32-bit RGB' width=400
            height=400");
        if (IJ.altKeyDown())
            drawOvalsFaster();
        else
            drawOvals();
    }

    void drawOvals() {
        ImagePlus imp = WindowManager.getCurrentImage();
        Random ran = new Random();
        int width = imp.getWidth();
        int height = imp.getHeight();
        for (int i=0; i<1000; i++) {

```

```

        int w = (int)(ran.nextDouble()*width/2+1);
        int h = (int)(ran.nextDouble()*width/2+1);
        int x = (int)(ran.nextDouble()*width-w/2);
        int y = (int)(ran.nextDouble()*height-h/2);
        IJ.setForegroundColor((int)(ran.nextDouble()*255),
                               (int)(ran.nextDouble()*255), (int)(ran.nextDouble()*255));
        IJ.makeOval(x, y, w, h);
        IJ.run("Fill");
    }
}

void drawOvalsFaster() {
    ImagePlus imp = WindowManager.getCurrentImage();
    Random ran = new Random();
    int width = imp.getWidth();
    int height = imp.getHeight();
    ImageProcessor ip = imp.getProcessor();
    for (int i=0; i<1000; i++) {
        int w = (int)(ran.nextDouble()*width/2+1);
        int h = (int)(ran.nextDouble()*width/2+1);
        int x = (int)(ran.nextDouble()*width-w/2);
        int y = (int)(ran.nextDouble()*height-h/2);
        ip.setColor(new Color((int)(ran.nextDouble()*255),
                               (int)(ran.nextDouble()*255), (int)(ran.nextDouble()*255)));
        Roi roi=new OvalRoi(x, y, w, h, null);
        ip.setMask(roi.getMask());
        ip.setRoi(roi.getBoundingRect());
        ip.fill(ip.getMask());
        if (i%10==0)
            imp.updateAndDraw();
    }
}

}

// Here the slightly different call in BeanShell !
Random_Ovals ov=new Random_Ovals();
ov.run(null);

```

A lot of documents are linked to the ImageJ website (<http://rsb.info.nih.gov/ij/>) and also API documents or well written books are available for ImageJ, see literature, etc..

**Note:** Bio7 integrates the version number 1.39 of ImageJ.

## 13 What else ?

### 13.1 Plugins for Bio7

Because Bio7 is a RCP-application it is possible to create perspectives with views as an Eclipse plugin to extend the abilities of Bio7. For instance it is easy to create a plugin which embeds other views and perspectives. This is a higher level of customization as the easy to use customization methods, which were previously mentioned for Bio7 (see Section 6.7). A useful resource for all kind of tutorials, articles and links is the eclipse website at (<http://www.eclipse.org>).

### 13.2 Programming examples for Bio7

The application Bio7 is distributed with a rich set of programming examples, which should facilitate the development of ecological models. The examples are so called **Snippets** (short programming examples) which are numbered and divided into two projects. One project (Bio7\_Examples) contains snippets for the different components of Bio7 which demonstrates how to use them and the flexibility of the platform in general. The other project (Ecological Modelling) contains snippets as examples for ecological modelling. This Snippets also show that you can use the different embedded application and modules to understand and analyse complex systems or bundle them in ecological modelling for more complex purposes.

The examples are available as projects. Please import the file **File->Import->Existing Projects into Workspace** into the workbench (see Section 14.6.1 for details about import).



## 14 How to's

In this section several tutorials describe the abilities of the Bio7 platform.

### 14.1 First steps

#### 14.1.1 Game of Life

1. Open the Game of Life in the Project **Ecological Modelling->Cellular Automata** (*Game\_of\_Life.java* or *Easier\_Game\_of\_Life.java*) with a double-click of the mouse.
2. Compile the source directly with the compile action which is active when the file has been opened.
3. Create an additional state.
4. Now select the random function which disperses active states in the cell (select the first random function which disperses all states ; Soil =dead, alive=1).
5. Press the start action of Bio7 to see the applied rules in the Quadgrid (Hexgrid).

#### 14.1.2 Random Walk

1. Open the *RandomWalk.java* file (**Ecological Modelling->Continuous**).
2. Compile it and then switch to the Points panel.
3. Press Start/Stop to see a random walk in a continuous space (where the precision can be adjusted according to the virtual size of the Points panel, etc.).
4. In the Points panel points will be set according to their coordinates.

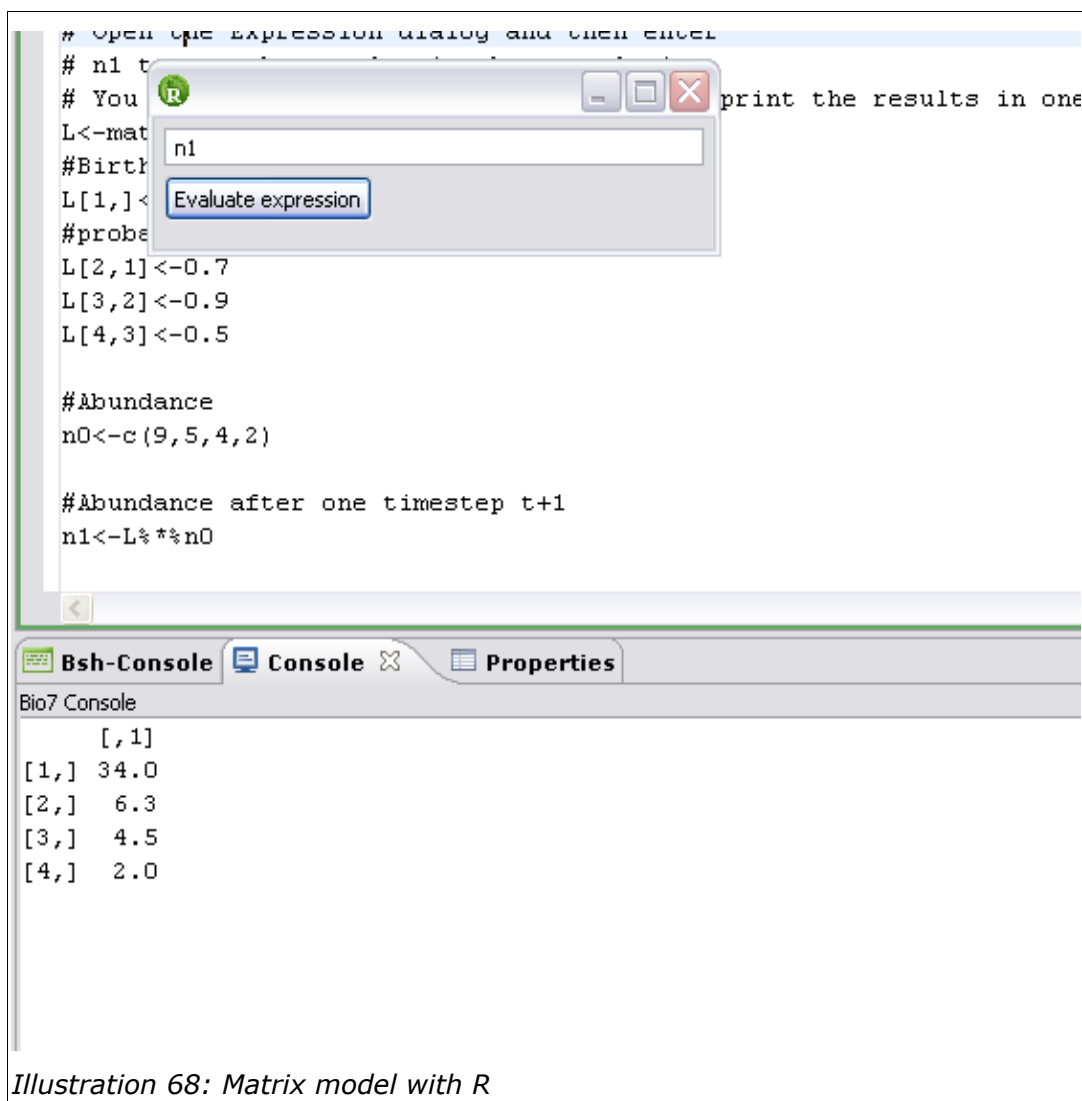
#### 14.1.3 Gears (3d)

Modelling in 3d is also possible. The "gears" example is an OpenGL script, which shows the popular Gears example in a Custom 3D view.

1. Open the gears example *Snippet\_OpenGL\_1\_Gears.bsh* (**Bio7\_Examples->OpenGL-Snippets**).
2. Interpret the script with the interpret action.

### 14.1.4 A Matrix Model in R

1. Open a connection to Rserve and then the file *LeslieMatrix.ecor* (**EcologicalModelling->MatrixModel**).
2. Execute the Script with the RScript action.
- 3.
4. Now open the R-expression dialog and type in *n1* to print an output of the abundance after one timestep (  $t+1$ ).

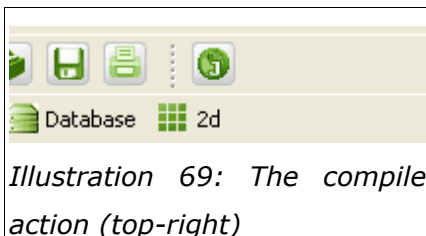


## 14.2 The first self made grid based simulation

1. Create a new project under: **File->New->General->Project** and choose a name for it.
2. Create a new Java file in this project **File->New->Bio7->JavaFile** and name it **random.java**
3. In the *ecomain* method of this file insert the code from the table below.
4. Now compile the Java code with the compile action of the embedded Compiler.
5. Start the simulation with the Start/Pause action.

Table 32: Source random method

```
public void ecomain(){
// Your code goes here!
for (int y = 0; y < Field.getHeight(); y++) {
    for (int x = 0; x < Field.getWidth(); x++) {
        Field.setState(x,y,(int)(Math.random()*Bio7State.getStateSize()));
    }
}
}
```



### 14.2.1 Explanation

In every timestep ( $t+1$ ) the *ecomain* method will be automatically invoked by the calculation thread of the Bio7 application.

This method automatically disperses all states (the function `Bio7State.getStateSize()` will return a number of all activated states) which are activated by change in the Quadgrid.

Furthermore, in each time step the amount of every state will be calculated and stored in a vector.

Tip: The counted data of each cell over the time can easily be transferred to OpenOffice with the OpenOffice menu (Open a connection->Open a spreadsheet->Send Bio7 Population Data).

### 14.2.2 Again in BeanShell

1. Create a BeanShell Script in the previously created project **File->New->BeanShell Script**
2. It is not necessary to have a ecomain method in BeanShell!
3. Insert the same code in the file, but at the end call (Table 33) or delete the ecomain method (Table 34).

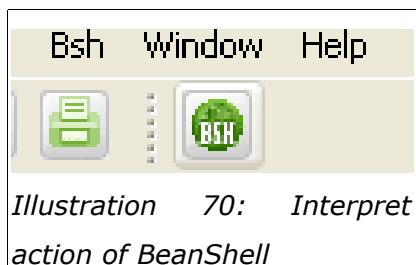
*Table 33: with ecomain method*

```
public void ecomain(){  
// With the ecomain method !  
for (int y = 0; y < Field.getHeight(); y++) {  
    for (int x = 0; x < Field.getWidth(); x++) {  
        Field.setState(x,y,(int)(Math.random()*Bio7State.getStateSize()));  
    }  
}  
}  
ecomain();
```

*Table 34: without ecomain method*

```
// Without the ecomain method....the grey marked word are also optional !  
  
for (int y = 0; y < Field.getHeight(); y++) {  
    for (int x = 0; x < Field.getWidth(); x++) {  
        Field.setState(x,y,(int)(Math.random()*Bio7State.getStateSize()));  
    }  
}
```

Interpret the code with the Interpret action of BeanShell.



### 14.2.3 Explanation

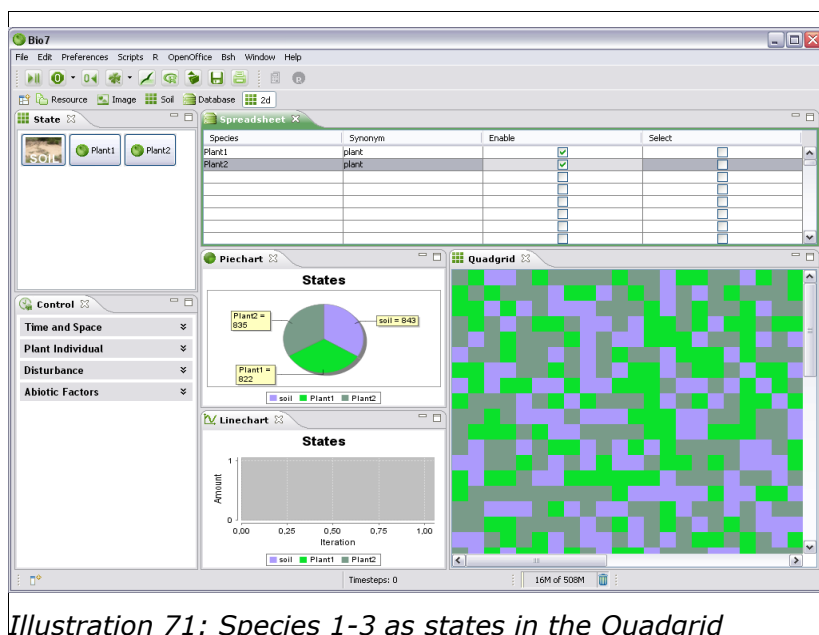
The method is the same as the method for the Java-Compiler. Different is, that the source is interpreted directly and executed in the Quadgrid, similar to one timestep in the compiled code (if started in the calculation thread). If the `ecomain` method was placed in the code, the method should be called for interpretation (at the end of the source in Table 33). If no `ecomain` method is present, the expressions will simply be interpreted (Table 34).

### 14.2.4 Conclusion

The Interpreter can be used to prototype different methods or to debug some piece of code (e.g. it can directly interact with the Quadgrid). For example if you type and execute (return) an expression like `Field.setState(0,0,1)`, the cell at the coordinates 0,0 (top left) in the BeanShell Console of Bio7 will directly be marked in the Quadgrid with the colour of the state (1).

## 14.3 The first gridded IBM Model

1. Create a database of plants with the help of the form in the Form view.
2. Now select the plants for simulation by activating them in the Spreadsheet view. The plants will be dispersed in the Quadgrid.



3. Now create a Java file with a random function which disperses Objects in the Quadgrid.

Table 35: Source First IBM-Model

```

/*This Snippet shows how to create discrete individual-based models.
In this this example activated plant objects are set by chance.
The default properties are the properties from the database.
After the creation of plants, individual properties can be adjusted
in the Control view!*/

1
public int zuff;

public void ecomain() {

2
    for (int y = 0; y < Field.getHeight(); y++) {

        for (int x = 0; x < Field.getWidth(); x++) {

            //State.getStateSize()-> how many states a present-> 0 = soil
3            zuff = (int) (Math.random() * (Bio7State.getStateSize()));

4            if (zuff > 0) { // Zero is the value of the Soil!

5                Field.setState(x,y,zuff);
                Field.setTempState(x,y,zuff);
                //Set individual in the array!

6                Planttemplate p = Bio7State.getPlanttemplate(zuff);
                //Get a Plant template by chance!
                //Create a new individual from the given template!
                Plant plant=new Plant(y, x, p);

7                Field.setPlant(x,y,plant);
                Field.setTempPlant(x,y,plant);

            }

8            else if (zuff < 1) {
                Field.setState(x,y,0);
                Field.setTempState(x,y,0);

9                Field.setPlant(x,y,Field.getSoil(x,y));
                Field.setTempPlant(x,y,Field.getSoil(x,y));

            }

        }

    }

}

```

1. Variable which stores the random values.
2. We create a nested loop over all cells in the grid.
3. By change a state will be selected with a random generator (the index is the number of the sequence of activation).

If Plant1 and Plant2 is enabled in the spreadsheet:

0 = soil // Soil by default has the numeric value 0!!!

1 = Plant1

2 = Plant2

4. If by chance a value for a plant has been selected (>0).
5. We set the state (the numeric value) at the specified index in the array.
6. A reference to the plant template with the selected number will be created.
7. A new object (Plant) will be created at the index inside the array (plant) of type Plant and the optional temporary array (planttemp->not necessary in this example!).
8. In the last step: if the value for the state soil (0) has been selected.
9. A soil object will be created at the specified index.

To create the method we compile everything to the calculation thread and then start the calculations with the **Start/Pause** action.

### 14.3.1 How to get and set Plant values in the Quadgrid(Hexgrid)

E.g., if a Quadgrid was filled with Plants, the first step to change a value would be to use the individual plant panel in the Control view. Every plant, which has been selected, is an individual object and none of the modifications to this does reflect on the other objects in the grid. If the action **Change global** is selected, all values of the selected individual will modify the values of all objects that belong to the same species (identified by the species name) in the grid. This can also be done programmatically by selecting an individual through its index in the Plant array. Then the attributes of this selected plant can be adjusted. The following list is a description of all available database attributes from a Plant object with methods to access it (get, set). Underlined (individual) attributes (with their Control view synonym in brackets) are adjustable in the **Control->Individual Plants** view. For some individual attributes the maximum values will be adjusted (for instance: sheightmax=Shoot-height) according to the sliders in the Individual Plants view.

**1.** String **species** -> the name of the species.

**1.** getSpecies()

**2.** setSpecies()

**2.** String **synonym** -> the synonym of the species.

**1.** getSynonym()

**2.** setSynonym()

**3.** String **lifeform** -> the lifeform of the species (hemikryptophyt, kryptophyt....).

**1.** getLifeform()

**2.** setLifeform()

**4.** String **lifespan** -> the life span; how long does the species exist (can be casted for a numeric value or used as categoric: annual...) the slider in the Control panel expects a numeric value !

**1.** getLifespan()

**2.** setLifespan()

**5.** String **anthesisstart** (Anthesis-start)-> the start of the anthesis.

**1.** getAnthesisstart()

**2.** setAnthesisstart()

**6.** String **anthesisend** (Anthesis-end)-> the end of the anthesis.

**1.** getAnthesisend()

**2.** setAnthesisend()

**7.** int **sheightmin**; -> minimum shoot height to use as an interval with the height.

**1.** getSheightmin()

**2.** setSheightmin()

**8.** int **sheightmax** (Shoot-height)-> maximum height of the shoot .

**1.** getSheightmax()

**2.** setSheightmax()



**9.** int **swidthmin** -> minimum width or diameter of the shoot.

**1.** getSwidthmin()

**2.** setSwidthmin()

**10.** int **swidthmax** (Shoot-width)-> maximum width or diameter of the shoot.

**1.** getSwidthmax()

**2.** setSwidthmax()

**11.** int **ssurface** (Shoot-surface)-> value for the shoot surface

**1.** getSsurface()

**2.** setSsurface()

**12.** int **rdepthmin**; -> r=root; minimum root depth .

**1.** getRdepthmin()

**2.** setRdepthmin()

**13.** int **rdepthmax** (Root-depth)-> maximum root depth.

**1.** getRdepthmax()

**2.** setRdepthmax()

**14.** int **rwidthmin** -> minimum root width.

**1.** getRwidthmin()

**2.** setRwidthmin()

**15.** int **rwidthmax** (Root-width)-> maximum root width.

**1.** getRwidthmax()

**2.** setRwidthmax()

**16.** int **rsurface** (Root-surface)-> surface of the root.

**1.** getRsurface()

**2.** setRsurface()

**17.** boolean **vegetative** -> ability to disperse vegetatively

**1.** isVegetative()

**2.** setVegetative()

**18.** boolean **sexuell** -> generally true (can be also apomictic!).

1. isSexuell()
2. setSexuell()

**19.** int **dispersalmax** (Distance of dispersal)-> maximum dispersal distance of seeds.

1. getDispersalmax()
2. setDispersalmax()

**20.** int **dispersalmin** -> minimum dispersal distance of the seeds.

1. getDispersalmin()
2. setDispersalmin()

**21.** int **spreadafter** (Dispersal (month))-> age of first reproduction of plant

1. getSpreadafter()
2. setSpreadafter()

**22.** int **established** (Establishment p(w))-> How many seedlings will survive and can establish.

1. getEstablished()
2. setEstablished()

**23.** boolean **near** -> disperses the plant in the near surroundings ?

1. isNear()
2. setNear()

**24.** boolean **far** -> does the plant species disperse over long distances ?

1. isFar()
2. setFar()

**25.** boolean **seedbank** -> importance of seedbank for plant species and its survival.

1. isSeedbank()
2. setSeedbank()

The following 7 attributes are chosen **Ellenberg numbers** which can be useful in a simulation, but can also be used for other purposes.

**26. int light**

1. getLight()
2. setLight()

**27. int temperature**

1. getTemperature()
2. setTemperature()

**28. int kontinental**

1. getKontinental()
2. setKontinental()

**29. int moist**

1. getMoist()
2. setMoist()

**30. int reaction**

1. getReaction()
2. setReaction()

**31. int n**

1. setN()
2. getN()

**32. int salt**

1. getSalt()
2. setSalt()

The following attributes do not appear in the database, but are attributes of all Plant objects.

**33.int age** (Age) not available in the database!

1. `getAge()`
2. `setAge()`

**34.int x** not available in the database!

1. `getX()`
2. `setX()`

**35.int y** not available in the database!

1. `getY()`
2. `setY()`

To set or receive a value simply call (in this example for the attribute age !):

1. **get:** `int age=Field.getPlant(x,y).getAge();`
2. **set:** `Field.getPlant(x,y).setAge(10);`

In the following simple example attributes from randomly dispersed plants are printed to the console

*Table 36: Simple example for a printed output of IBM properties*

```
public int zuff;
public void ecomain() {
    //Call method to set activated Plants randomly !
    random();
    //Access and print some properties. e.g.: species and age !
    if (Field.getPlant(10,10)!= null) {
        Plant pl=Field.getPlant(10,10);
        System.out.println(pl.getSpecies());
        System.out.println(pl.getAge());
    }
}

public void random() {
    for (int y = 0; y < Field.getHeight(); y++) {

        for (int x = 0; x < Field.getWidth(); x++) {
            //State.getStateSize()-> how many states a present-> 0 = soil
```

```

        zuff = (int) (Math.random() * (Bio7State.getStateSize()));

        if (zuff > 0) { // Zero is the value of the Soil!

            Field.setState(x, y, zuff);
            Field.setTempState(x, y, zuff);
            //Set individual in the array !
            Planttemplate p = Bio7State.getPlanttemplate(zuff);
            //Get a plant template by chance!
            //Create a new individual from the given template properties!
            Plant plant = new Plant(y, x, p);
            Field.setPlant(x, y, plant);
            Field.setTempPlant(x, y, plant);

        }

        else if (zuff < 1) {
            Field.setState(x, y, 0);
            Field.setTempState(x, y, 0);
            Field.setPlant(x, y, Field.getSoil(x, y));
            Field.setTempPlant(x, y, Field.getSoil(x, y));

        }

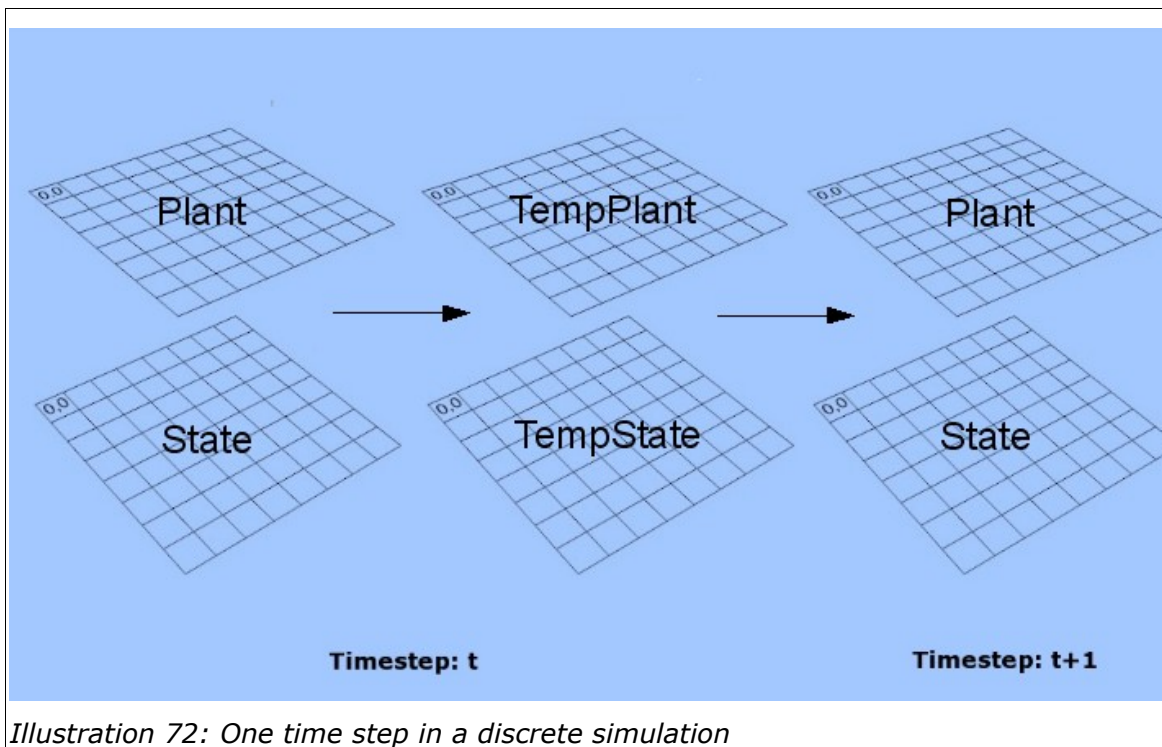
    }

}

```

### 14.3.2 Annotation

Use the tempstate, planttemp arrays (getTempArray(x,y), getTempPlant(x,y)) for storing intermediate patterns for the grid. This will be necessary, if all cells shall be calculated in one timestep under the same conditions. Once the calculations for the whole field are done, the temporary values can be copied to the **state**, **plant** arrays as a setup for the next calculation step.



*Illustration 72: One time step in a discrete simulation*

### 14.3.3 Copying instead of creating new objects

Creating and deleting instances in an object oriented approach is a cost and time intensive procedure. For a better performance and memory use in a fixed grid it is a better approach to copy **Plant** objects in the **Plant** array instead of creating and deleting **Plants** each time. In a simulation you can simply copy (**not referencing!**) all attributes from one grid place to another, e.g. when a plant disperses to a new grid place. Then values can be adjusted according to the needs (for example, set the age to the value 0, etc.) Attributes can also be copied to a "soil" place because the soil state is also a plant object.

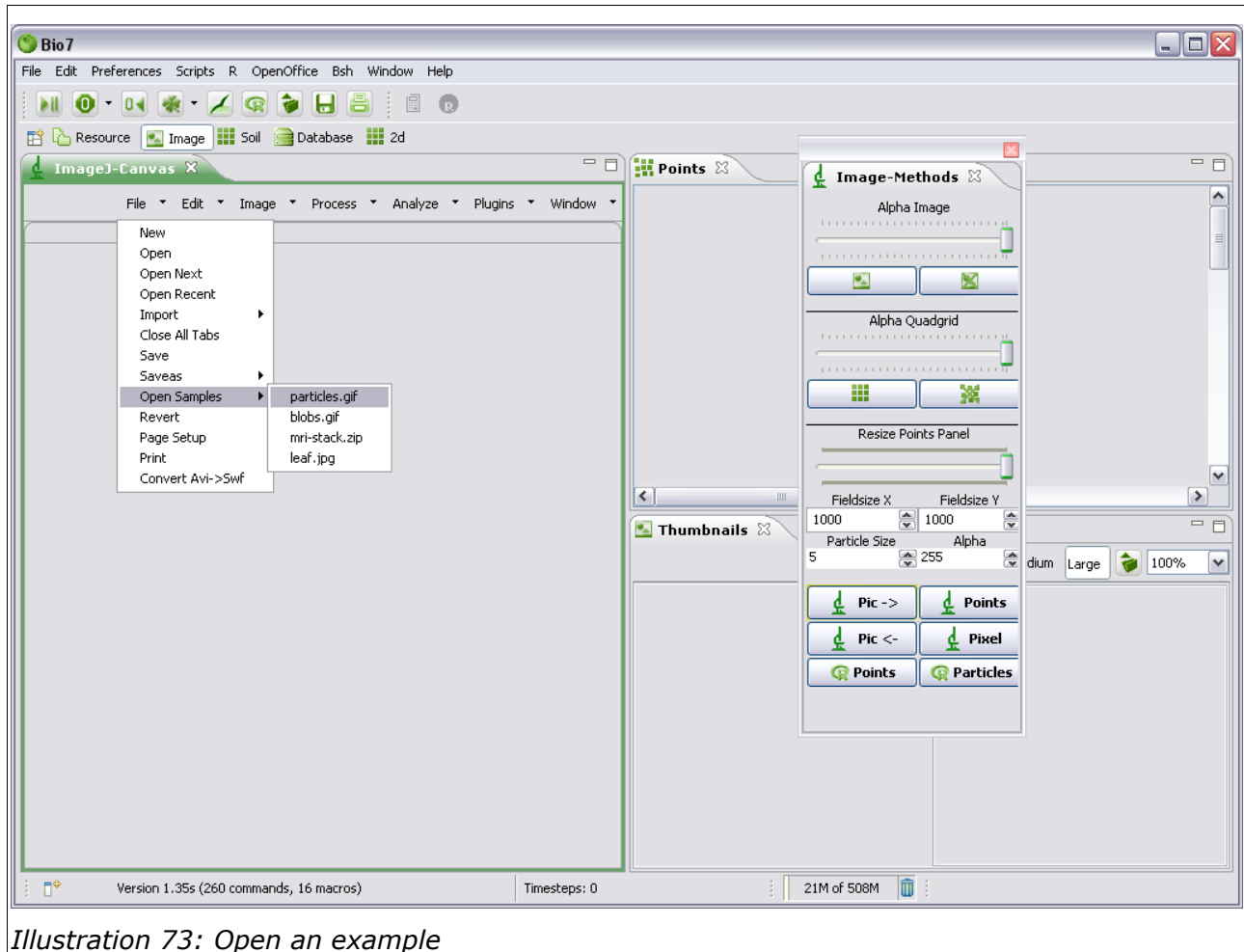
```
import static com.eco.bio7.discrete.Field.*; //A static reference to the Field !
[-----some code -----]

//Get the plant which will disperse to x and to y in the grid
Plant p=getPlant(x,y)
//Get the temporary plant (also soil is a plant object !)at the new place.
Plant temp=getTempPlant(to_x,to_y);
/* Copy the species name and values which are important for the simulation to
the temporary plant array. We do not reference the old object!*/
temp.setSpecies(p.getSpecies());
// Set the new age of the dispersed individual !
temp.setAge(0);
[-----some code -----]
// etc. if needed !
```

## 14.4 Point Pattern Analysis

Open an example (particles.gif) from ImageJ in the ImageJ view menubar

**File->Open Samples** then follow the steps described below.



*Illustration 73: Open an example*

1. Open the Bio7 toolbar (Image-Methods).
2. Convert the image to 8-bit image **Image->Type->8-bit** (for particles.gif example not necessary!).

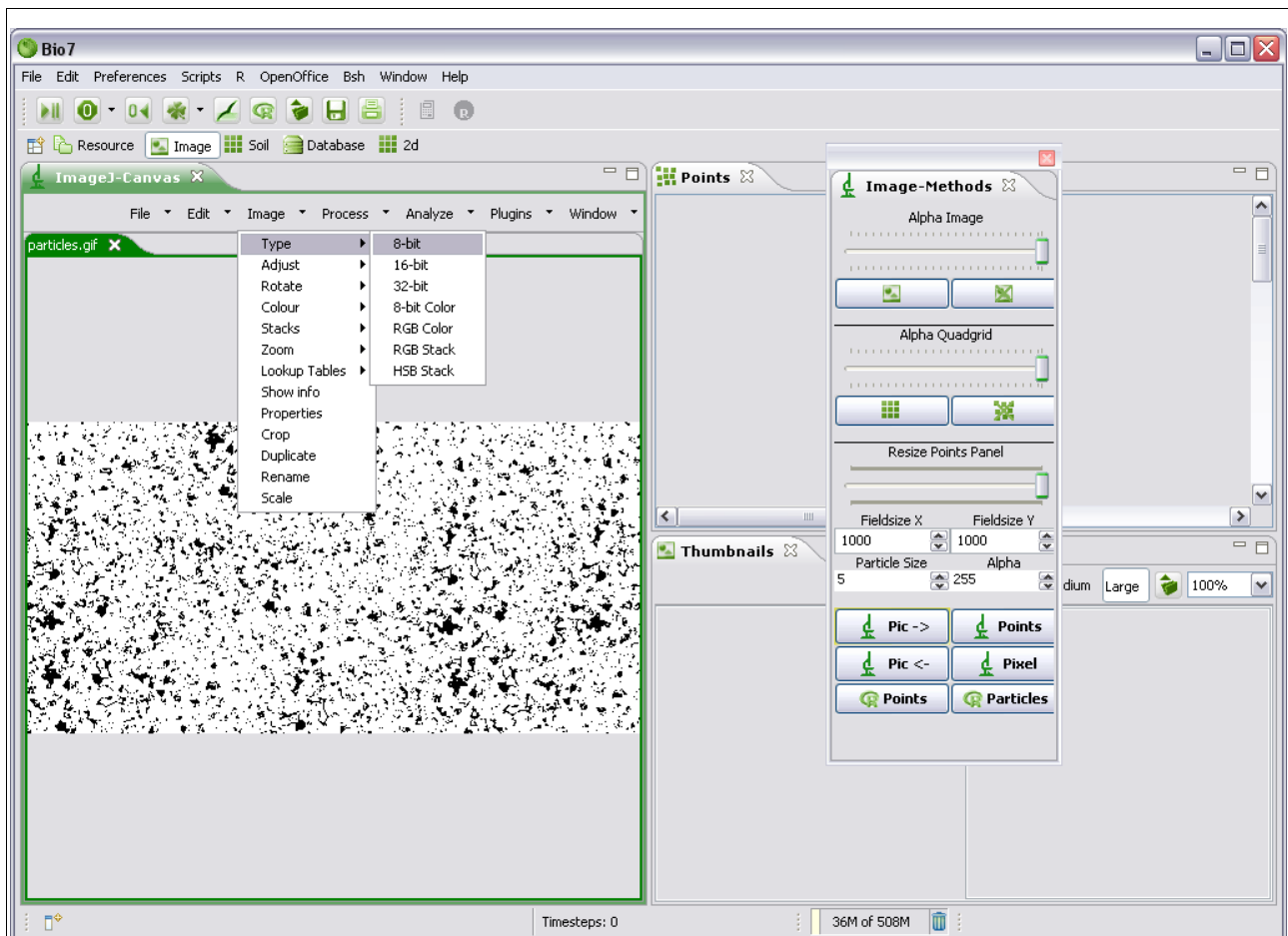


Illustration 74: Convert image

3. Adjust the threshold of the image in the way that all particles are marked with red  
**Image->Adjust->Threshold.**

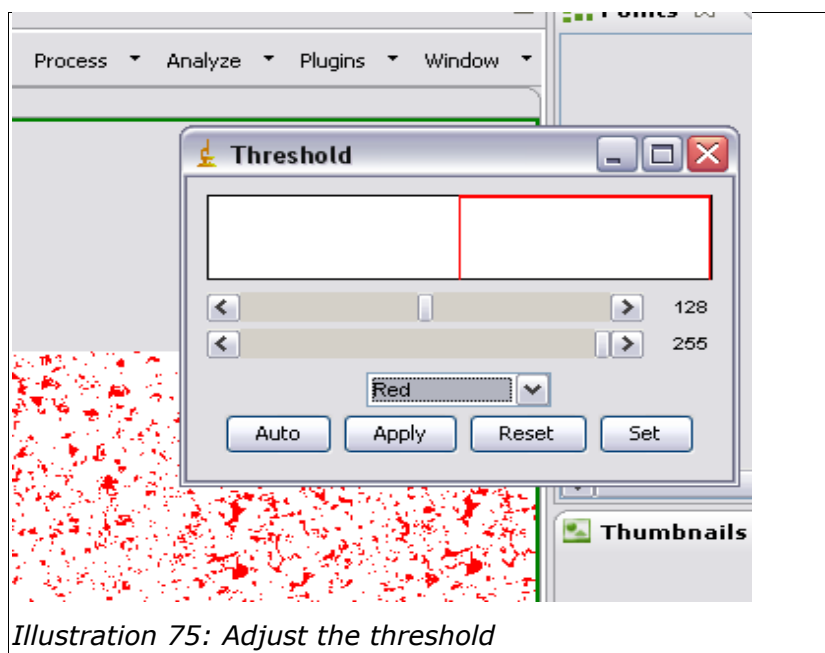
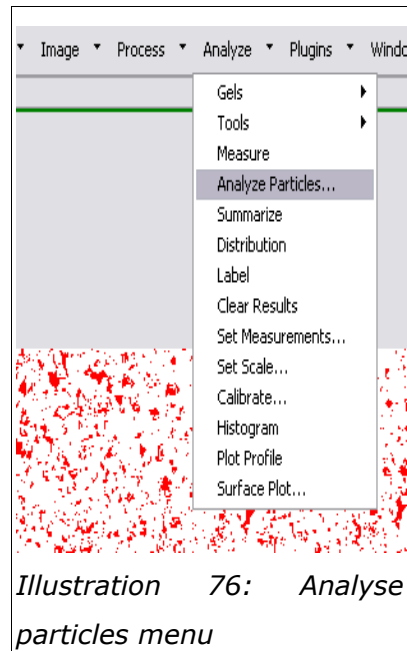


Illustration 75: Adjust the threshold

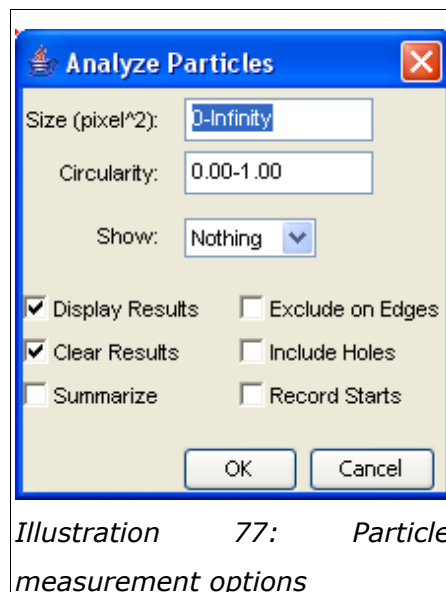


4. Now open the Particle Analysis dialog **Analyze->Analyze Particles**.



5. Select the appropriate options in the dialog.

- a) Select the particle size and the circularity interval of the particles which should be selected for measurements.
- b) The **Show** combobox item will let you select options to display an image with the measured particles.
- c) **Display results** will actively show you the current measurements in a spreadsheet (Only keep activated, when you want to see the measured values in a table ->slower !).



**6.** Confirm the dialog to analyse the particles.

**7.** Go to the Image methods dialog (**Window->Bio7-Toolbar**) and transfer the image to the Points panel which will calibrate the Points panel's "Area of Analysis" (2000\*1000 for the particles.gif example).

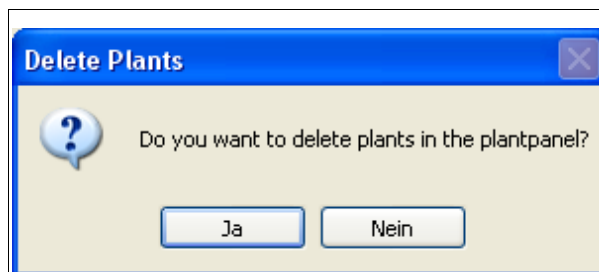


**8.** Before you proceed you have the possibility to select a plant in the Plants view (activated plants from the spreadsheet) or a state which was defined before (programmatically) to map the points to a special state or plant species from the database. For this example the points will simply be mapped to the default soil state.

**9.** The alpha value of the points and the size of the points can also be adjusted before the points will be transferred.

**10.** Now press the top Points button to create points in the Points panel according to the coordinates of the measurements (coordinates with values  $\geq 1.5$  will be mapped to 2.0 "int" values).

**11.** Select "Yes" in this dialog if you want delete all points that are present in the panel. Select "No" if you want to add points to existing points in the Points panel.



*Illustration 78: Add or delete existing points*

**12.** The points are drawn in the Points panel with the colour of the default state (You can change the state colour in the Plants view!).

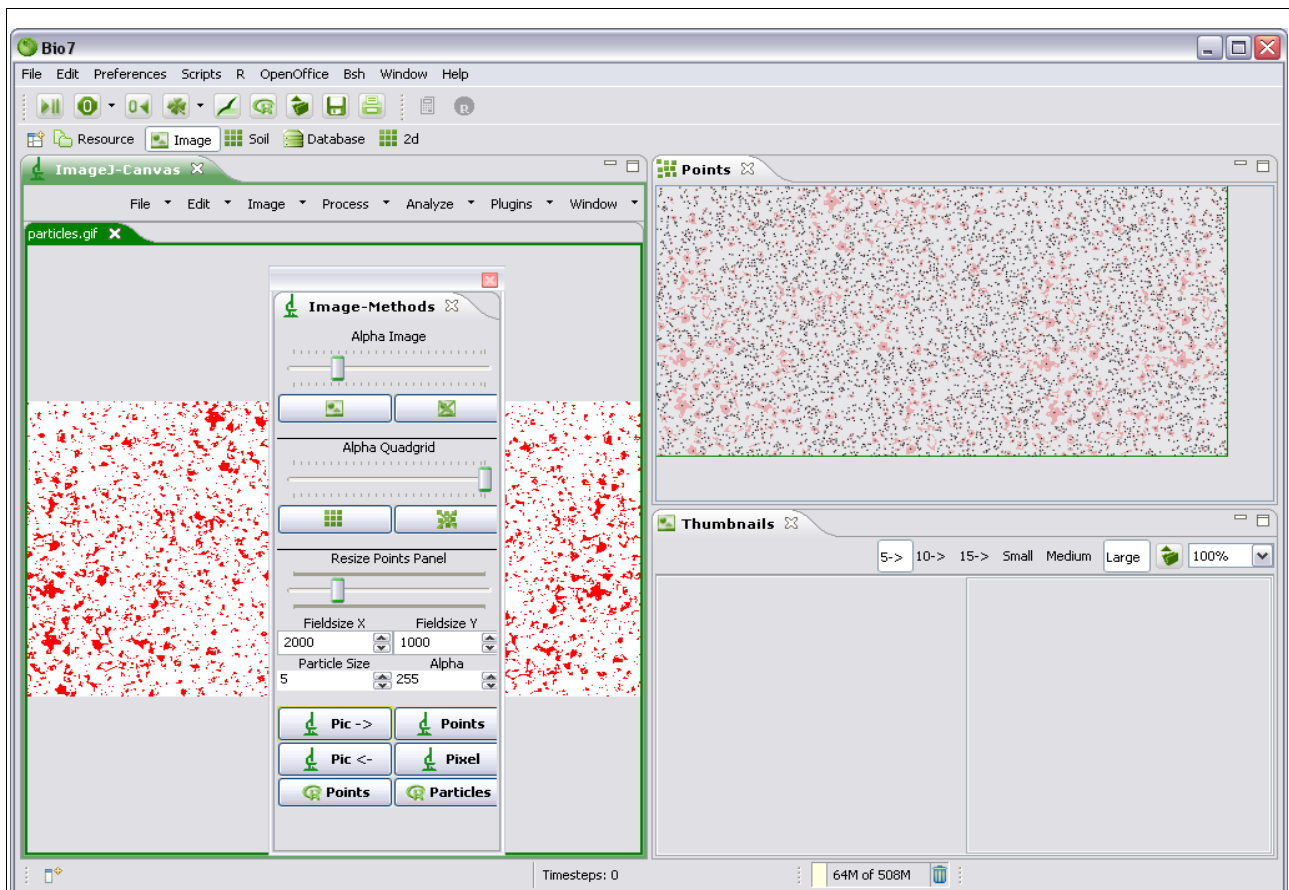


Illustration 79: Transferred values as points in the Points panel

- 13.** To transfer the data to the R workspace start the Rserver and transfer the measured points to the R workspace by pressing the R transfer button.

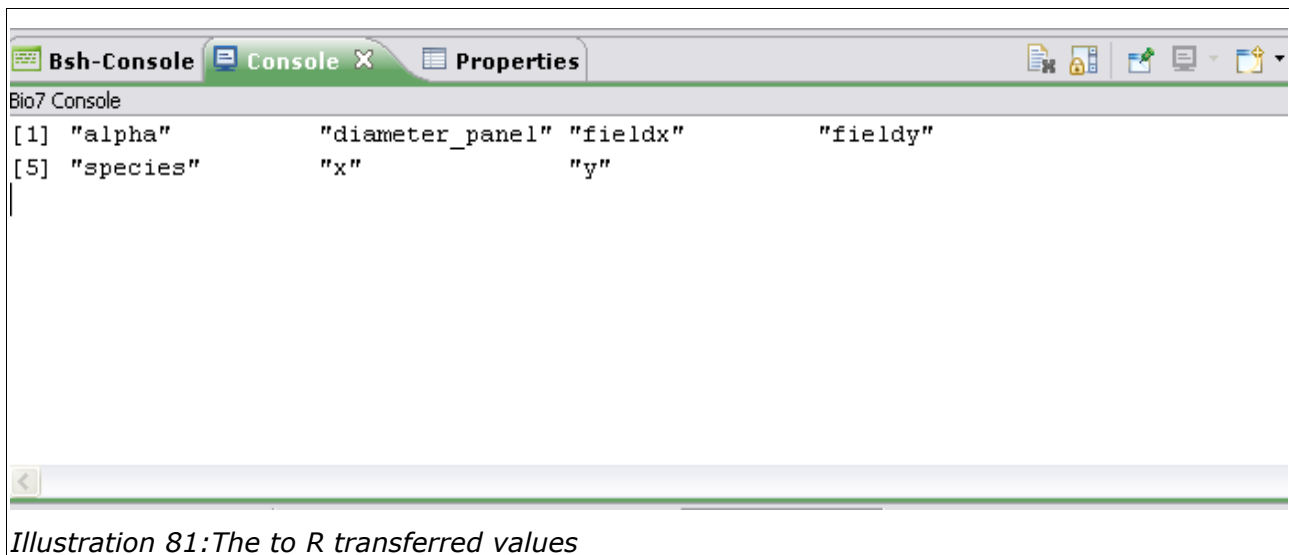


- 14.** Open the R expression dialog of Bio7 and list the transferred values.

**R->Evaluate Expression** type: `ls()` for a list.



Illustration 80: Expression dialog



**15.** Optional you may start the RGui (Windows) with the transferred values (temporary workspace file) and type the command for list `ls()` in the R-Console.

The transferred values for this method are:

- The **alpha** value for the points (can be used as an additional attribute in modelling displaying the magnitude of certain parameters (competitive strength, etc.)
- **diameter\_panel** will give you a list of all point diameters in the panel.
- **imagex**, **imagey** are the attributes for the "Area of Analysis".
- **species** is a numeric value (0,1,2,...), which indicates the species or state from Bio7
- **x,y** are the coordinates of the state

One index in the list ( `alpha[1]`, `x[1]`, `y[1]`...) belongs to one point of the analysis.

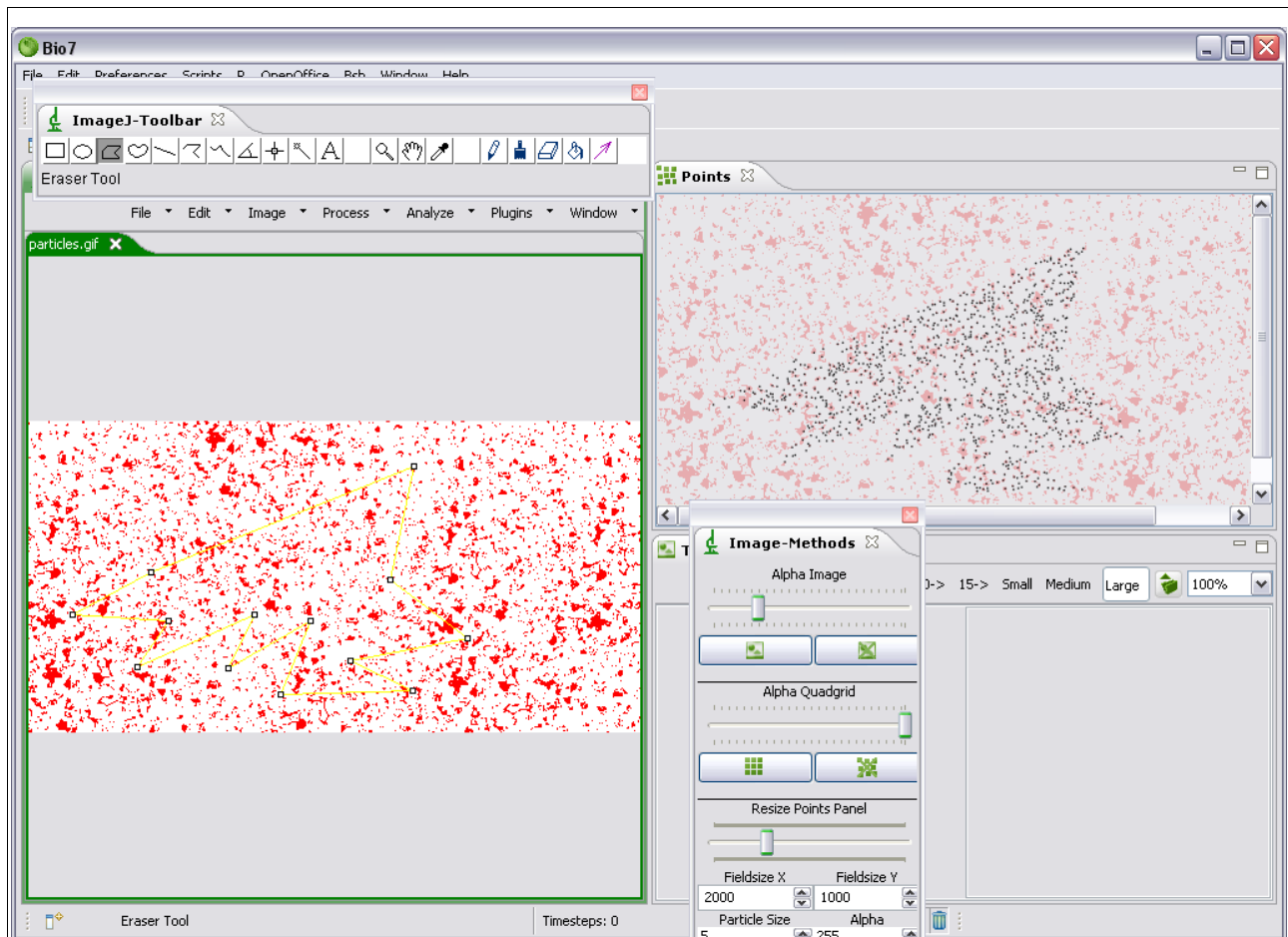
For a further analysis you now can proceed with a custom analysis (see `R_snippets`) for point-pattern, voronoi, etc.

**Note:** R plots differ from the Points panel paintings in the origin of the y coordinate.

The 0,0 coordinate can be found in the top left of the Points panel whereas the 0,0 coordinates in a plot of R can be found in the bottom left.

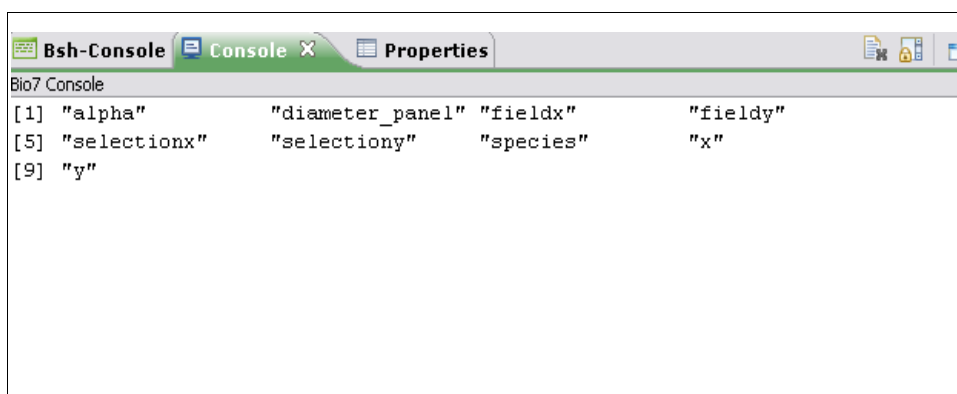
### 14.4.1 Additionally

If you select an ROI (Region of interest) in ImageJ, only this region will be analysed. So it is possible to create an irregular region for analysis.



*Illustration 82: Transferred points from selection in ImageJ*

The points of the selection will be transferred to R as an array (**selectionx**, **selectiony**) which supports the analysis of irregular regions (methods for this kind of analysis are available in the default installed package spatstat).

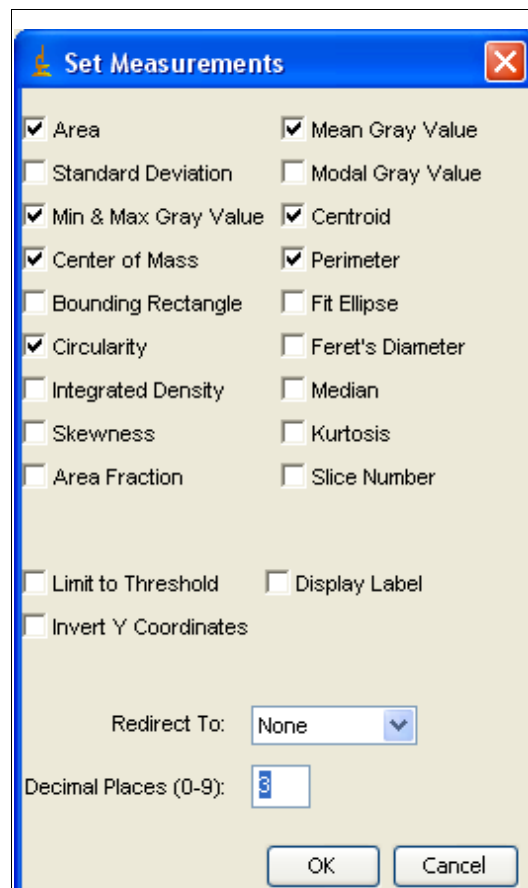


*Illustration 83: Transferred values with the selection values*

Tip: The whole procedure of particle analysis can be automated in ImageJ with self defined macros which can be recorded and executed for repetition. Then this recorded macro can be dropped to the Bio7 folder (for the ImageJ macros), which extends the **Scripts->ImageJ-Macros** menu.

#### 14.4.2 Getting all attributes of a Particle Analysis

In Bio7 it is also possible to get more attributes from a particle analysis. By selecting all options in the **Set Measurements** dialog before the Particle Analysis, many more attributes can be transferred to R. Furthermore, coordinates will be transferred without being rounded (centerofmass). To transfer this values it is necessary to select all particle measurements to be transferred, before the analysis is started. **Analyse-> Set Measurements.**

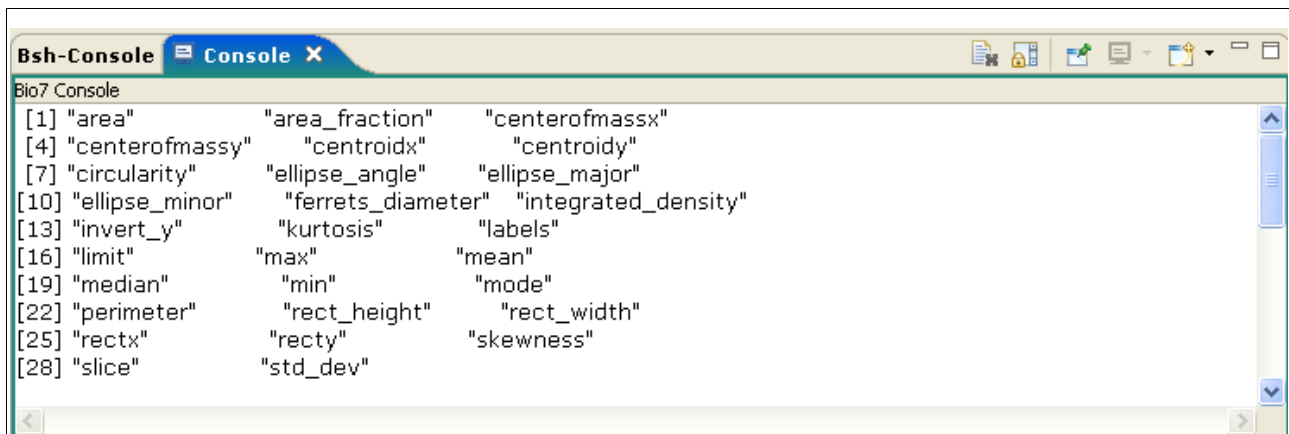


*Illustration 84: Set measurements dialog*

1. In the dialog all required measurements have to be selected before a Particle Analysis is started (The **Centroid** option always needs to be selected and is by default).
2. Then the particles can be analysed.

3. Now, instead of the first points transfer method the second method must be selected in the detached Image-Methods view.

When executing this action, all measured particle values will be transferred directly to R without detour over the Points panel.



*Illustration 85: All attributes from an ImageJ measurement transferred to R*

**Note:** After each transfer the Analysis has to be repeated.

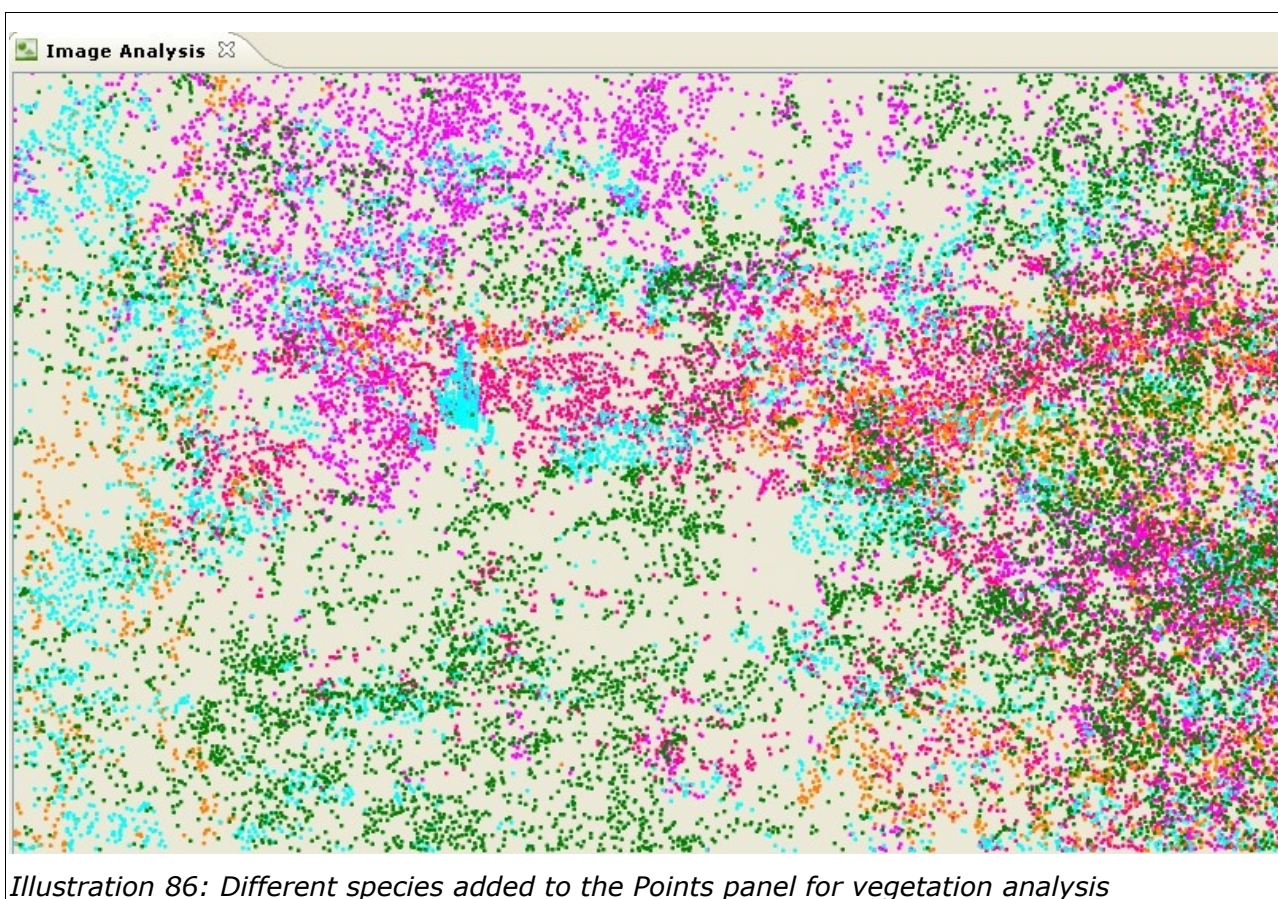
### 14.4.3 Conclusion

This description is a simple example for the analysis and the mapping of particles to the Points panel of the Bio7 application and R. If you see this example in the context of plant communities, the particles in this examples could also be examples for plants, which were measured in the field or selected from digital photographs. When selecting plants from digital photographs Bio7 is capable to automatically transfer these values to the Points panel or directly to R. If the vegetation is not easily to separate into different species, more advanced methods of Image Analysis can be used with the API of ImageJ or by using pattern recognition methods in R. Furthermore it is possible with this application to examine spatial setups of different plant communities using the advanced spatial statistic methods in R.

(In the plants dialog different species can be selected and then be added to the Points panel after the analysis) . You can also write custom scripts for the imports of x,y values to the Points panel (see the Points panel Snippets).

**Tip for digital images:** If a vegetation is to homogeneous to be identified automatically from an image, it could be a good approach to prepare the image before the analyses is taking place (mark the different species in a graphic program like Gimp to facilitate the automatic recognition of these in ImageJ).





*Illustration 86: Different species added to the Points panel for vegetation analysis*

A good start for similar topics are the tutorials on the Imagej website, which may be applicable to complex vegetation issues . Additionally R offers strong packages for spatial analysis. Visit the RSpatial project website at <http://sal.uiuc.edu/csiss/Rgeo/> for more information.

## 14.5 Transfer a spatial point pattern to a grid

In Bio7 it is possible to map a spatial pattern from an image to the discrete Quadgrid (and indirectly to the Hexgrid). To do this follow exact the procedure of the previously described method to transfer points from an analysed image to the Points panel. The difference is that the Quadgrid has to be activated in the Points panel. This can be achieved with the method **set** or **unset discrete grid field** in the Points panel. If the grid is activated, it will be shown as an additional layer in the Points panel. The alpha for the grid can also be adjusted to make all activated components visible in the panel (image or points).

1. Activate and scale the Quadgrid in the Points panel (Scaling can be done with the scroll wheel for the Quadgrid in the Points panel)
  - (a) The scaling can be used to calibrate the size of the grid to which particles should be mapped (grid resolution).



**2.**Open the blobs example in the ImageJ view **File->OpenSamples->blobs.gif**.

**3.**Adjust the threshold of the image the way that all blobs are selected and marked with red **Image->Adjust->Threshold**.

**4.**Send the image to the Points panel to adjust the "Area of Analysis" (You can remove the image afterwards).

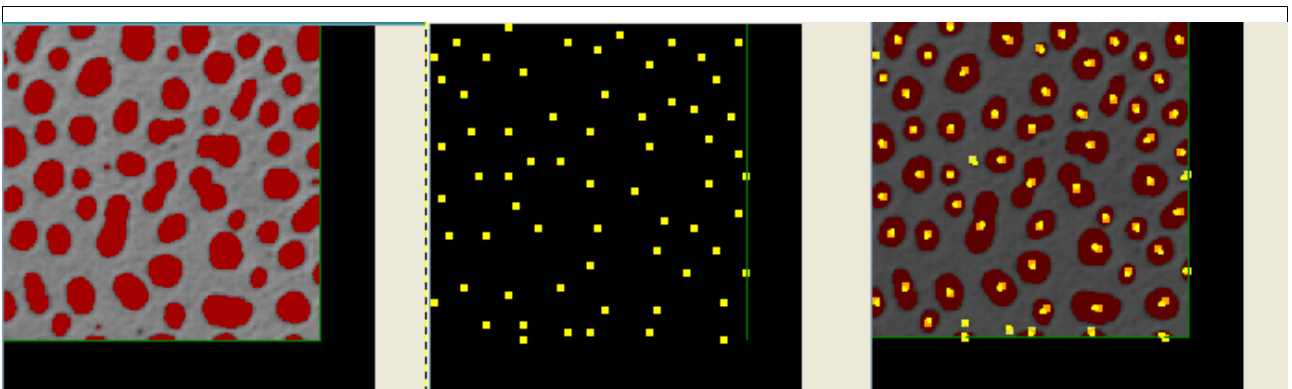


**5.**Select an activated state in the Plants view.

**6.**Now execute a particle analysis **Analyze->Analyze Particles...**

**7.**Transfer the particle values to the Points panel.

**8.** Which grid cell will be marked as adjusted state depends on the centroid values from the particles measurement (rounded values) .



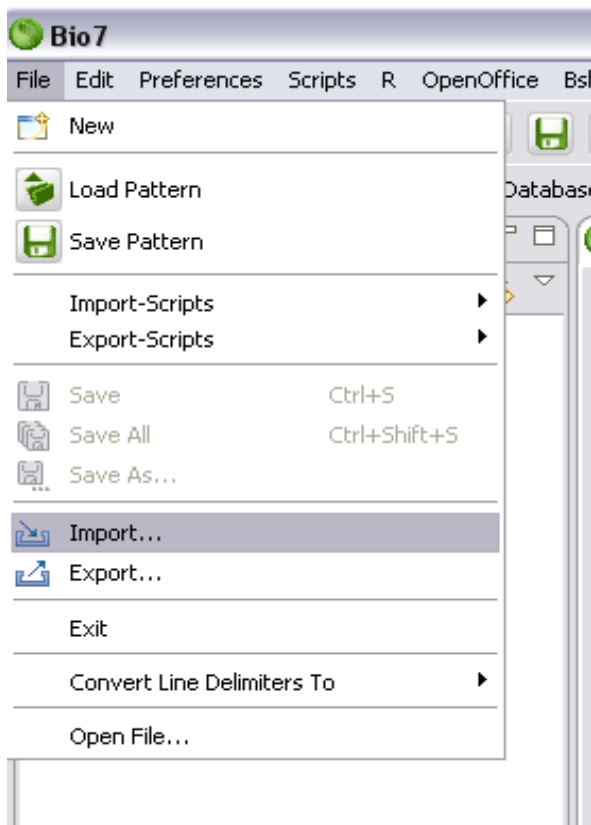
*Illustration 88: Transfer of Points to the Quadgrid*

## 14.6 Import and Export Projects and Files

### 14.6.1 Import existing projects

Table 37: *From the Eclipse Workbench User Guide (<http://help.eclipse.org/>)*

1. From the main menu bar, select **File > Import...**. The Import wizard opens.



2. Select **General > Existing Project into Workspace** and click **Next**.
3. Choose either **Select root directory** or **Select archive file** and click the associated **Browse** to locate the directory or file containing the projects.
4. Under **Projects** select the project or projects which you would like to import.
5. Click **Finish** to start the import.

## 14.6.2 Import resources from the file system

Table 38: *From the Eclipse Workbench User Guide (<http://help.eclipse.org/>)*

You can use the Import Wizard to [import resources from the local file system into an existing project](#).

- 1.** From the main menu bar, select **File > Import...**. The Import wizard opens.
- 2.** Select **General > File System** and click **Next**.
- 3.** Click the **Browse** button on the next page of the wizard to select the directories from which you would like to add the resources.
- 4.** In the import selection panes, use the following methods to select exactly the resources you want to add:
  - Expand the hierarchies in the left pane and select or clear the checkboxes that represent the folders in the selected directory. Then in the right pane, select or clear checkboxes for individual files.
- 5.** Click **Filter Types** to filter the current selection for files of a specific type.
- 6.** Click **Select All** to select all resources in the directory, then go through and deselect the ones that you do not want to add.
- 7.** Click **Deselect All** to deselect all resources in the directory, then go through and choose individual resources to add.
- 8.** Specify the Workbench project or folder that will be the import destination.
- 9.** When you have finished specifying your import options, click **Finish**.

*Tip:* You can also import folders and files by dragging them from the file system and dropping them into one of the navigation views, or by copying and pasting.

### 14.6.3 Import resources from an Archive file

Table 39: *From the Eclipse Workbench User Guide (<http://help.eclipse.org/>)*

You can use the Import wizard to [extract files from an archive file into the Workbench](#).

- 1.** From the main menu bar, select **File > Import...**. The Import wizard opens.
- 2.** Select **General > Archive File** and click **Next**.
- 3.** Click the **Browse** button on the next page of the wizard, to select the archive files that contain the files you want to extract and import into the Workbench.
- 4.** In the import selection panes, use the following methods to select exactly the resources you want to add:
  - Expand the hierarchies in the left pane and select or clear the checkboxes that represent the folders in the selected directory. Then in the right pane, select or clear checkboxes for individual files.
- 5.** Click **Filter Types** to filter the current selection for files of a specific type.
- 6.** Click **Select All** to select all resources in the directory, then go through and deselect the ones that you do not want to add.
- 7.** Click **Deselect All** to deselect all resources in the directory, then go through and choose individual resources to add.
- 8.** Specify the Workbench project or folder that will be the import destination.
- 9.** When you have finished specifying your import options. click **Finish**.

## 14.6.4 Export resources to the file system

Table 40: *From the Eclipse Workbench User Guide (<http://help.eclipse.org/>)*

- 1.** In one of the navigation views, select the resources that you want to export.
- 2.** From the main menu bar, select **File > Export...**. The Export wizard opens.
- 3.** Select **General > File System** and click **Next**.
- 4.** By default, the resources that you selected will be exported, along with all their children. Optionally, use the checkboxes in the left and right panes to select the set of resources to export, and use push buttons such as **Select Types** to filter the types of files that you want to export.
- 5.** Click the **Browse** button on the next page of the wizard, to select the directory you would like to export the resources to.
- 6.** Specify the directory in the file system that will be the export destination.
- 7.** Click **Finish**.

Tip: You can also export folders and files by dragging them from the Navigator view to the file system or by copy and paste.

## 14.6.5 Export resources to an Archive file

Table 41: *From the Eclipse Workbench User Guide (<http://help.eclipse.org/>)*

You can use the Export wizard to [export resources from the Workbench to an archive file in the file system](#).

- 1.** In one of the navigation views, select the resources that you want to export.
- 2.** From the main menu bar, select **File > Export...**. The Export wizard opens.
- 3.** Select **General > Archive File** and click **Next**.
- 4.** By default, the resources that you selected will be exported along with their children. Optionally, use the checkboxes in the left and right panes to select the set of resources to export, and use push buttons such as **Select Types** to filter the types of files that you want to export.
- 5.** Specify the path and name of the archive file into which you want to export the selected resources.
- 6.** Click **Finish**.

## 14.7 Execute, evaluate and plot a R script

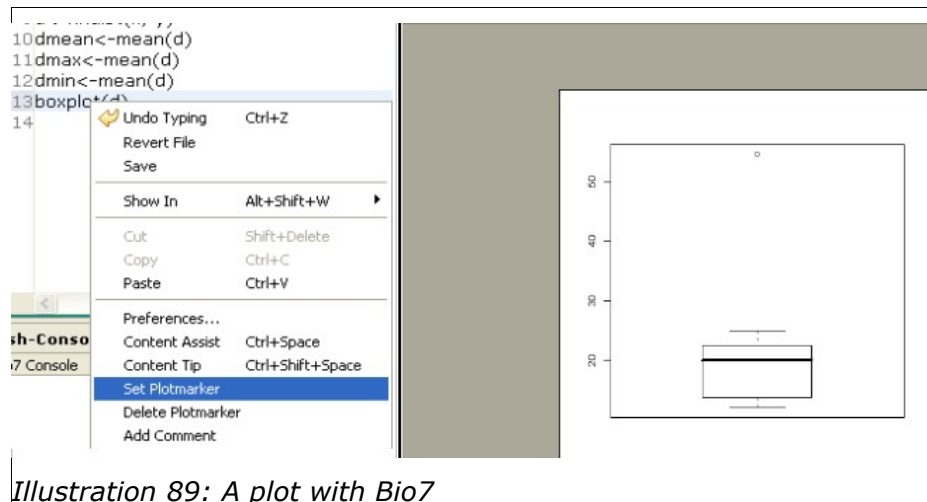
To execute, evaluate and plot a RScript the following procedure can be applied:

1. Create a new RScript with the Wizard of Bio7 **File->New->Bio7->RScript**.
2. Insert the code from Table 42 in the script.
3. Then interpret the RScript.
4. Open the expression dialog **R->Evaluate Expression** and evaluate the following expression:  
**(a)** `summary(d)` ->prints a summary of the interpoint distances to the console.
5. Now create a simple boxplot of the interpoint distances:  
Add the following line at the end of the script in Table 42:  
**(a)** `boxplot(d)`
6. Then mark this line by selection with the plotmarker in the context menu.
7. Interpret the script again.

Automatically a pdf will be opened from a temporary file with the registered pdf viewer. The temp file is named: "temp\_plot+last selected line number+.pdf".

*Table 42: Interpoint distances*

```
#This Snippet calculates the interpoint nearest neighbour distances from existent x,y values !  
#Uses the library spatstat !  
  
library(spatstat)  
x<-c(20,45,23,100,50,45,25)  
y<-c(45,22,30,34,56,2,56)  
d<-nndist(x, y)
```



*Illustration 89: A plot with Bio7*

**Annotation:**

A wrong input in Windows will cause the Rserve application to close and a loss of the current workspace will occur (save the workspace to avoid a loss of valuable data!). Simply restart the Rserve application when no server connection is available any more.



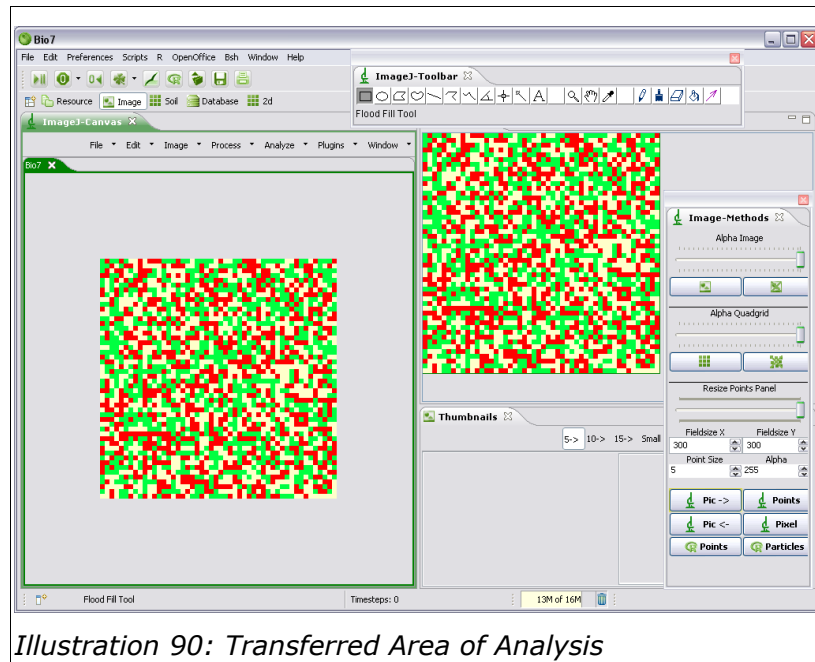
## 14.8 Create a custom GUI for a Rscript

With the Bio7 application it is possible to create a Graphical User Interface (GUI) for statistical methods in R. With this GUI the user can be guided through a complex sequence of necessary steps. These steps can be arranged in a flow or by using the resources of BeanShell to communicate with R and OpenOffice. Therefore BeanShell can be used to create a custom GUI for R methods. For a better input and output the OpenOffice API can be used to get values from an OpenOffice spreadsheet and send the R results back to the office application. In the **Bio7\_Examples->R\_BeanShell\_Snippets->Snippet8\_GuiForR.bsh** a spreadsheet will be opened and the user will be asked to insert values. After selection by the user a summary function is invoked and the results will be sent back to the spreadsheet as a new table.

## 14.9 Create a Flash file presentation from a simulation

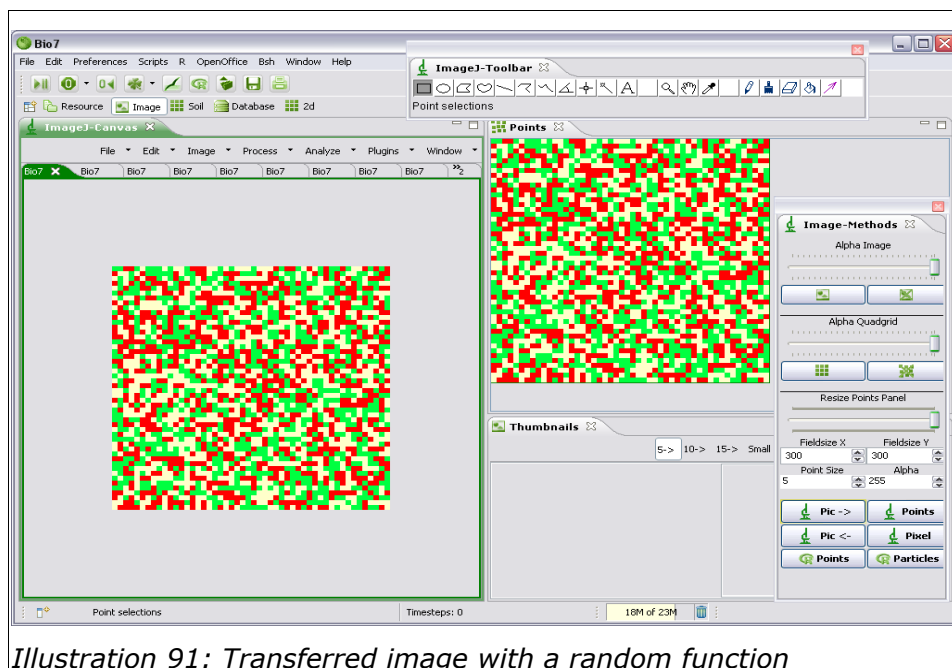
To create a presentation from a simulation in Bio7 (and to display the results, e.g. in a website) Bio7 embeds a tool which can convert \*.avi files to \*.swf files (avi2swf from swftools). The possibility to transfer images from the Points panel to ImageJ allows to capture all kind of activities in the Points panel and to transfer them to ImageJ. In ImageJ the tabbed images then can be converted to a stack. In an ImageJ stack all converted images are available as frames, which can be edited (It is possible to create context information or shapes in a stack by means of the different methods in ImageJ). After the frames have been edited, the stack can be stored as an \*.avi file. The \*.avi file can be converted to a \*.swf file with the embedded converter in Bio7.

- 1.**Open the Image perspective of Bio7 and then open the toolbar of ImageJ **Window-> ImageJ-Toolbar** and **Bio7-Toolbar** in the ImageJ view.
- 2.**In the Bio7-Toolbar activate the Quadgrid layer. By default this should fit into the "Area of Analysis" which will be transferred as an RGB-Image.
- 3.**Adjust the magnification of the Points panel which determines the size in ImageJ.
- 4.**Transfer the first image to ImageJ by triggering the transfer button.



*Illustration 90: Transferred Area of Analysis*

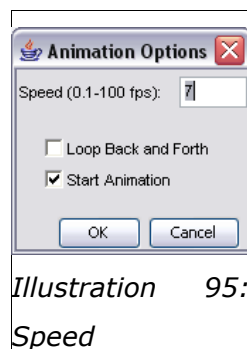
5. Now select the random function in the toolbar of Bio7 which disperses all default states in the Quadgrid.
6. Select the Points panel to trigger a repaint event for the Points panel and then transfer the changed image.
7. Repeat the steps several times (10 times).



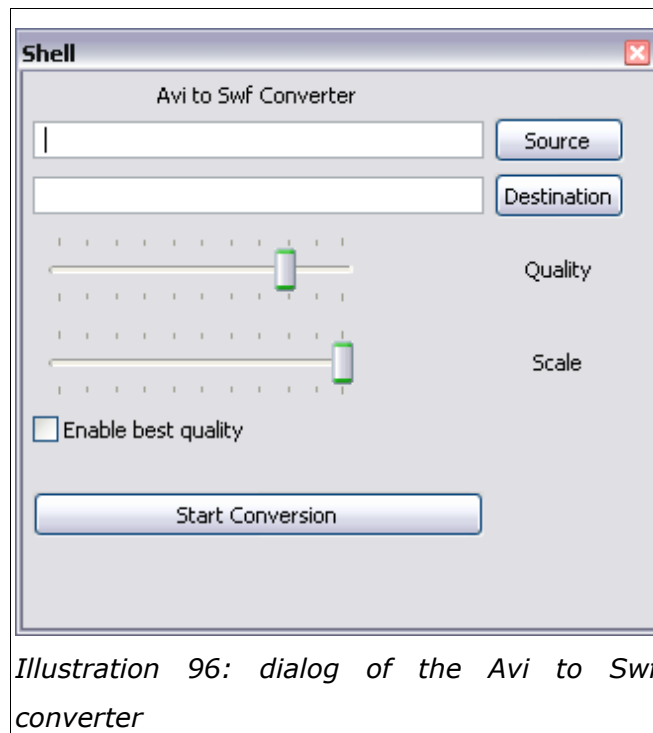
*Illustration 91: Transferred image with a random function*



- 10.** Use the the **Polygon selection** from the toolbar of ImageJ to create a message box selection. Select the colour dialog and then select an appropriate colour. Fill the selection with the **Fill** method in the context menu.
- 11.** Select the colour dialog (opened as a tab in Bio7) to adjust the colour again and then select the **Text tool** to create a text in the message box (open context menu and fill text or type Ctrl+d).
- 12.** Now customize the presentation with some animation options (framerate) **Image->Stacks->Animation Options** to a reasonable framerate .



- 13.** Export the Stack as an \*.avi file **File->Save as->Avi...** to a location on the local filesystem.
- 14.** Then open the dialog for the avi to swf conversion tool and select the file to convert to a swf file: **File->Convert Avi->Swf** .



*Illustration 96: dialog of the Avi to Swf converter*

The converted file can now be embedded in a website as a presentation of simulation results, etc.

### **14.9.1 Note**

This is just one example of the many possibilities to create a presentation from a simulation. You can also embed pictures, cut elements out of an image and paste them into the frame of a stack, add stack frames or delete them or draw different shapes to create a full featured presentation. Also you can take R Charts and animate them for presentation (plot a voronoi several times with different parameters and then combine them to an animation). The avi2swf converter is also able (out of the context of Bio7) to convert arbitrary \*.avi files.

## 14.10 Open perspectives and views programmatically

For presentation issues or for guiding users through self defined methods (for instance in a flow), perspectives, editors and views can be accessed programmatically.

*Table 43: Example which opens a view*

```
Work.openView("com.eco.bio7.quadgrid");  
//Use Work.openMaxView("com.eco.bio7.quadgrid"); for maximizing the view !
```

See the Bio7 Snippets for more examples.

## 14.11 Add states without the database

In Bio7 it is possible to define states programmatically without using the database (for instance if you want to make a simulation with animals, etc.). States can be defined, set or removed programmatically. When a state has been defined by name the Plant view will automatically produce a button with the name for that state (or will display an image if available). Furthermore the charts will be updated automatically and a counter will be added to each defined state. The states can be saved or loaded as usual with the Load, Save pattern method. Bio7 will automatically update all views and activate plants which have been stored to that file and inactivate plants, which have been not. All states get a numeric value according to the sequence of creation (starts with 0!). In the following Snippet 50 states will be added to Bio7 without deleting the soil state (but can also be removed for instance when creating a cellular automata like the Game of Life).

*Table 44:*

```
for (int i = 2; i < 50; i++) {  
    Bio7State.createState(Integer.toString(i));  
    //Use Bio7State.removeState(Integer.toString(i)); for deletion !  
}
```

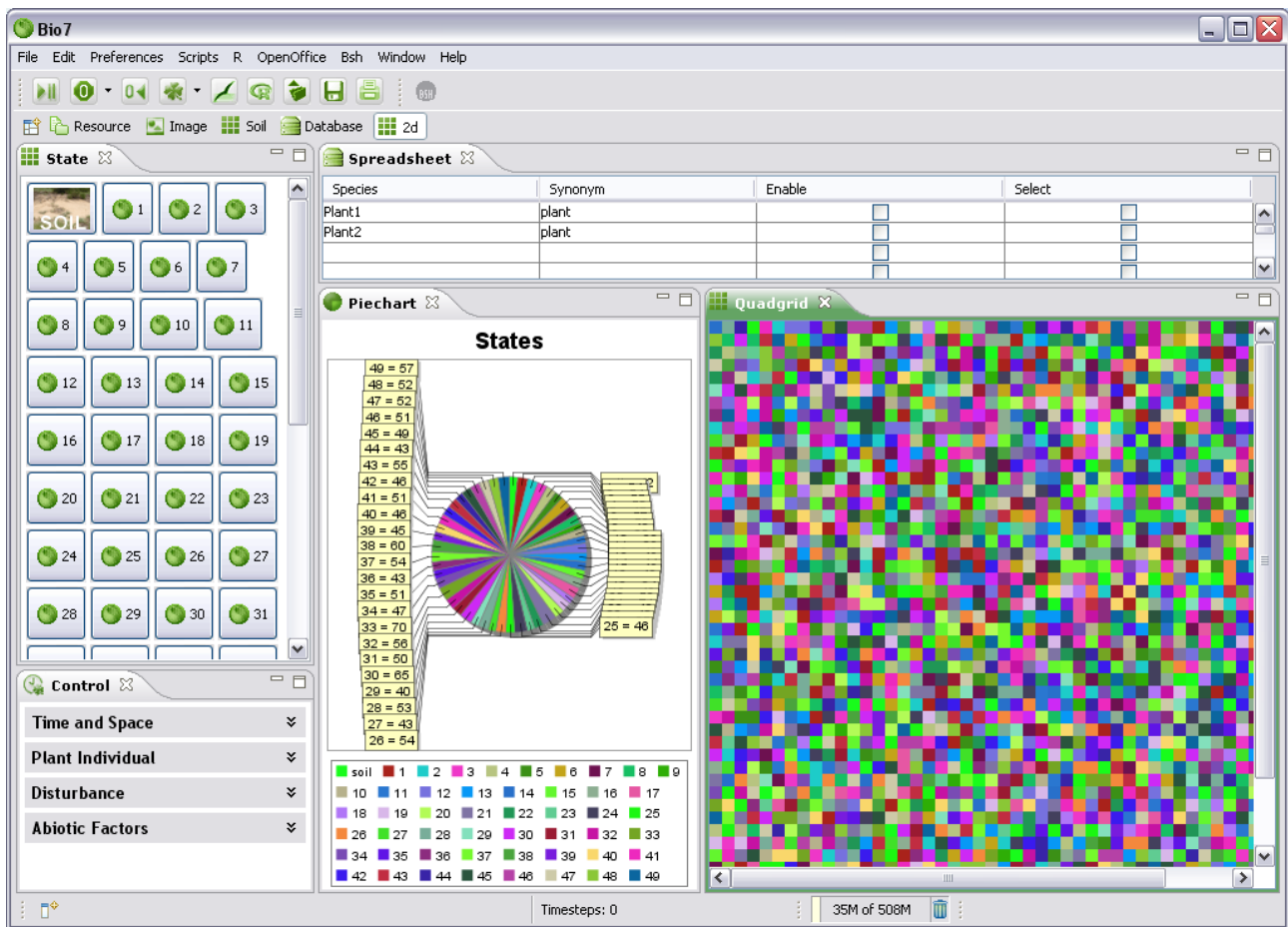
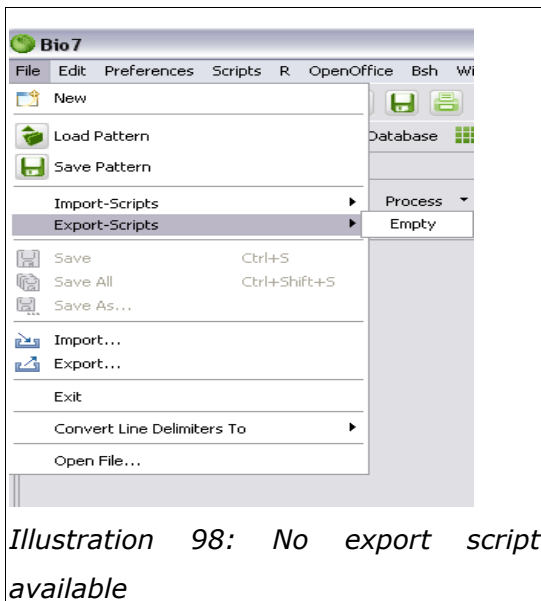


Illustration 97: Fifty states defined programmatically

**Note:** The states which will be created with this procedure are just numeric values and no individual (Plant) properties are available like for the states from the database. The states can be dispersed in the Quadgrid with the mouse (drag or click). It is of course possible to mix database states with the states, that have been defined programmatically. In case the state *soil* is not needed for simulation, it is legal to remove it to avoid unnecessary states in a custom simulation.

## 14.12 Add self defined menus to Bio7 with scripts

In this example we will add an export script to the Bio7 menu



We will create a script (BeanShell) for the export of the numeric values from the grid array. Therefore we define a script *Export\_Pattern.bsh* in Table 45 which creates an export dialog and saves the grid pattern to a file with a self-defined ending *\*.pattern*.

Table 45: Source of *Export\_Pattern.bsh*

```
JFrame frame = new JFrame("Pattern");
frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
frame.setAlwaysOnTop(true);
JFileChooser file = new JFileChooser();

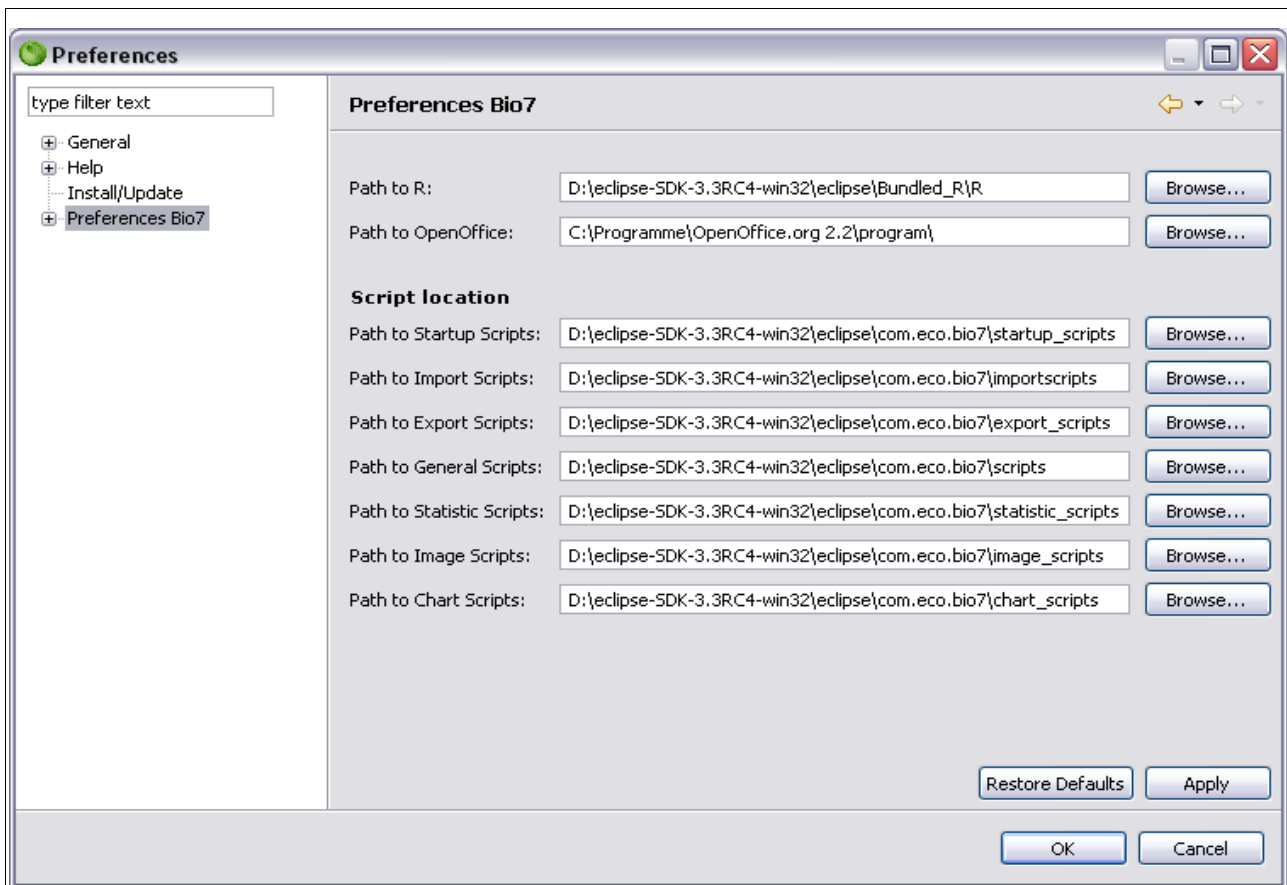
if (file.showSaveDialog(frame) != JFileChooser.CANCEL_OPTION) {
    File pict = file.getSelectedFile();

    BufferedWriter out = new BufferedWriter(new OutputStreamWriter(
        new FileOutputStream(pict)));
    for (int i = 0; i < Field.getHeight(); i++) {
        for (int u = 0; u < Field.getWidth(); u++) {
            out.write(Integer.toString(Field.getState(u, i)) + " ");
        }
        out.newLine();// Linebreak
    }

    out.close();
}
```

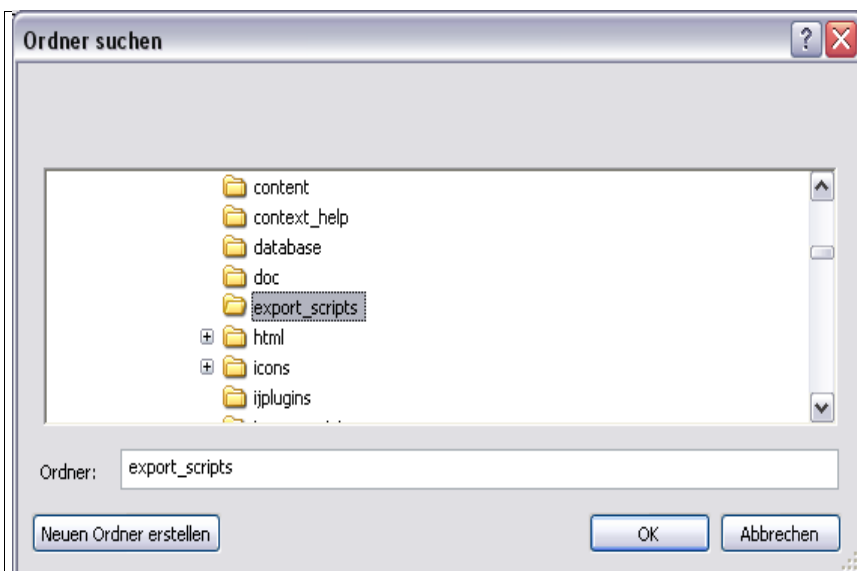


1. After storing this file in the Navigator we simply copy it to the clipboard, open the preferences in Bio7 and select the **Preferences Bio7** dialog.

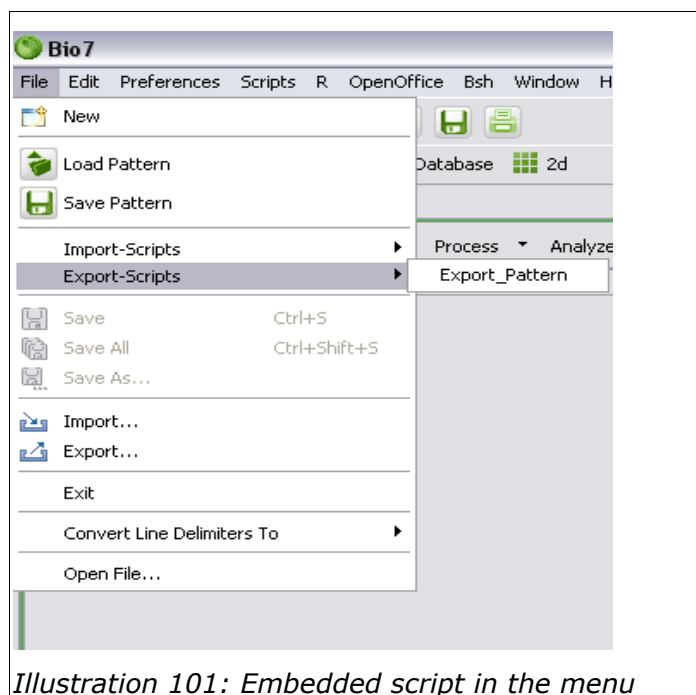


*Illustration 99: Preferences of Bio7*

2. Then we browse to the export\_scripts location and simply paste our file to that location.
3. Now we open **File->Export-Scripts**. The script is added automatically to the menu and will be executed when selected.



*Illustration 100: Script location*



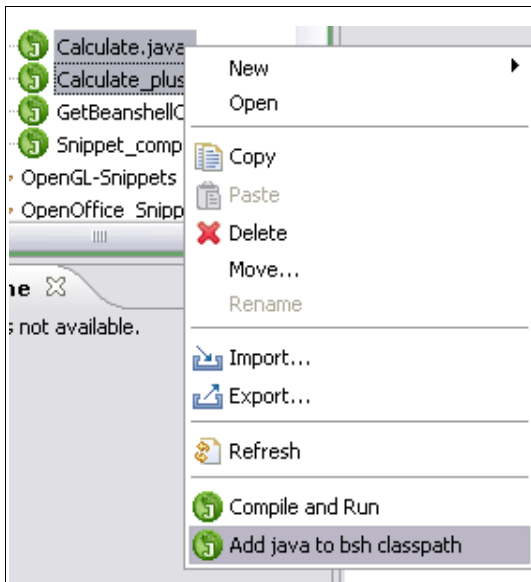
*Illustration 101: Embedded script in the menu*

The same procedure is applicable to the other script menus . Macros for ImageJ have to be dropped to the ImageJ-Scripts location with the same procedure.

### 14.13 Compile Java files to BeanShell

To add a compiled Java file to the BeanShell “namespace” you have to simply right-click on a \*.java file in the Navigator. Only if a \*.java file is selected, the menu will be extended with the action.

*Add Java to bsh classpath.*



*Illustration 102: Extended the context menu in the Navigator*

When the action is invoked, the file will be compiled from the embedded compiler. The compiled class is then available with the name of the selected file. To call, for example, the `ecomain` method of the compiled file `Calculate.java`, you have to type `Calculate.ecomain()`; Additional methods in the file are called in the same way. (for example, `Calculate.print()`; for the `print` method in the file ). Furthermore, it is possible to select more than one file in the Navigator which will be compiled and added to the “namespace” of BeanShell. They can also be called directly from the BeanShell console. The compiled methods are much faster than the default interpreted methods in BeanShell. Normally the embedded compiler is only used to compile one file to the main calculation thread. With BeanShell it is possible to store the compiled class by means of a reference to a static variable (the current compiled class) Bio7 offers. The compiler compiles a class of the type `Ecoclass` to the `Ecoclass` variable `Compiled.ecoclass`. You then can set a reference to that instance in BeanShell with:

```
Ecoclass eco=Compiled.getEcoclass();
```

Now you can call `eco.ecomain()` If you compile another file with a different `ecomain` method, you will be able to set another reference to the compiled class like:

```
Ecoclass eco2=Compiled.getEcoclass();
```

which gives you a reference to the currently compiled file.

Now two different references (and `ecomain` methods) are available in the BeanShell “namespace”, which both can be invoked with BeanShell.

(You can also automate this, e.g. in a flow for a simulation setup).

### 14.14 Call compiled methods in BeanShell in the calculation thread of Bio7

In Bio7 it is possible to invoke the interpreter with compiled code in the main calculation thread of the application (the calculation thread which calls the `ecomain` method in Bio7 at each timestep). In the following example (Table 46) two methods from two different files (`Calculate`, `Calculate_plus`) will be executed in the calculation thread (which has been compiled to BeanShell).

*Table 46: Call the interpreter of BeanShell*

```
public void ecomain(){
Interpreter.doInterpret("Calculate.ecomain()",null);
Interpreter.doInterpret("Calculate_plus.ecomain()",null);
}
```

With this option it is possible to execute more than one compiled class (available as an object) in the main execution thread, if a model is more complex and the code is dispersed to more single files.

### 14.15 Use R calculations interactively in a Java method

Another nice feature in Bio7 is the possibility to use calculation results or R methods directly in the main calculation thread of Bio7. This can be realized by invoking the R interpreter directly from the Compiler and receive the results from R calculations. The following source in Table 47 calls the `Rserve` interpreter directly and prints the received output to the console in each timestep.

*Table 47: Dynamic R in Bio7*

```
/* This Snippet is an easy example how to get variables
from R after a calculation and embed it in an simulation run (try..catch.. are necessary)!*/

import org.rosuda.JRclient.RSrvException;
```

```

public void ecomain() {
    double[] result=null;
    try {
        Rserve.getConnection().eval("A<-c(runif(100)*100)");

        //Get the results from R!
        result = (double[])
        RServe.getConnection().eval("summary(A)").getContent();

    } catch (RSrvException e) {

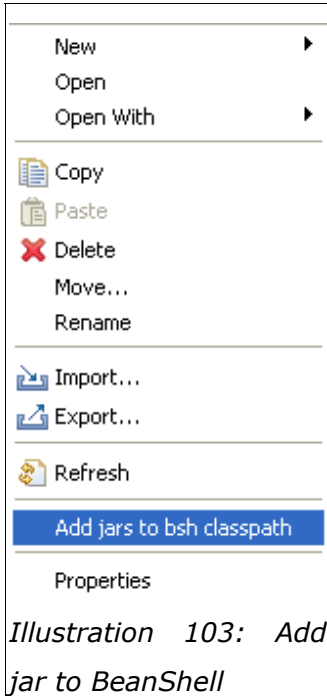
        System.out.println(e.getRequestErrorDescription());
    }

    for (int i = 0; i < result.length; i++) {
        System.out.println(result[i] + " ");
    }
}

```

## 14.16 Import external libraries to BeanShell

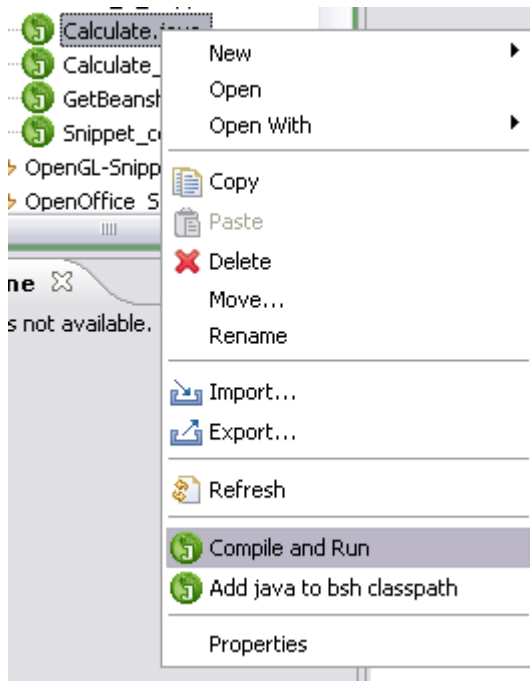
To import external libraries to BeanShell simply right-click in the Navigator on a \*.jar Java library file. The action will only be visible in the context menu of the Navigator if a \*.jar file is selected.



Multiple selections are possible and will be added to the "namespace" of the BeanShell interpreter. See the BeanShell manual for more information.

## 14.17 Compile and run Java files

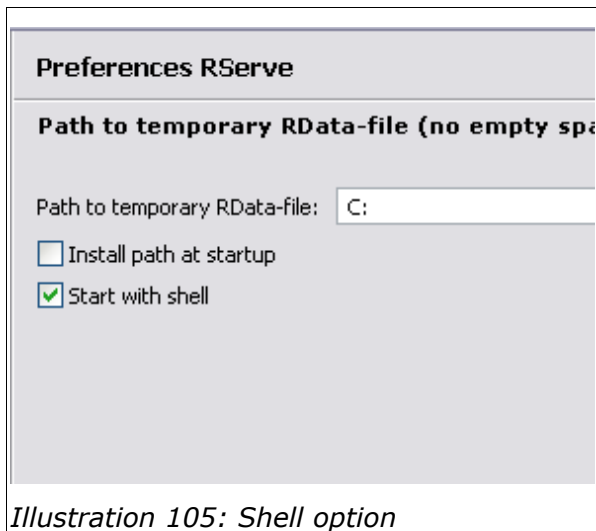
In the context menu of the navigator another option is available to compile and run an opened \*.java file. This menu point is enabled when a \*.java file was selected with a right-click of the mouse. By executing this action the selected file will be compiled and referenced in BeanShell. Then the **ecomain** method will automatically be invoked (like the main method in a regular \*.java file).



*Illustration 104: Compile and Run in Bio7*

## 15 Notes

### 15.1 Notes about R



*Illustration 105: Shell option*

You can start the Rserve applications with two options. If you enable the option "Start with shell" in the preferences dialog **Preferences->Preferences Bio7->Preferences Rserve** you can start the Rserve application with an opened shell. In this shell errors are displayed and you can also use the scan function which by default is not possible with the regular connection mode. It is recommended to start and work with two options to see the differences and the advantages of both. The Rserve application is not stable on Windows. Evaluation of a wrong

expression will lead to an interruption of the communication to the server. The server then has to be restarted. If the server is restarted in Bio7, an embedded application (Windows) will kill all running Rserve instances to guarantee that only one Rserve instance is active (else communication errors could occur). Bio7 offers also the possibility to set the registry path for the embedded R. By default this tool is disabled (That the Bio7 installation does not harm any settings of the user !).

You can enable this option in the preferences dialog:

**Preferences ->Preferences Bio7->Preferences Rserve ->Install path at startup**

After the path was recreated (after a restart of the Bio7 application), you can disable this option again. An installation of R also offers the option to create or delete a registry path in Windows. Please read the R-FAQ for this options !



*Illustration 106: Set the registry path*

For more information about the Rserve application and the Windows version and also for examples which can be used in Bio7 visit:

<http://www.rforge.net/Rserve/>



## 15.2 Notes about BeanShell

Sometimes it is necessary to clear the "namespace" of BeanShell to correctly interpret a script, because Bio7 works with one connection to the interpreter. **Bsh->Clear**

(In the console with the command: `clear()` )

The BeanShell interpreter has no method for an interruption of a running interpretation.

It is a good approach to test all scripts before they are, e.g. embedded in a flow.

The default imports for the interpreter are available in a text file and can be changed in the file: **...\plugins\com.eco.bio7\_1.0.0\bin\imports\beanshell.properties** (Windows).

Please consider to make a backup before you change the file !

For more information about BeanShell visit:

<http://www.beanshell.org/>

## 15.3 Notes about the Java-Compiler

Sometimes in a running simulation an exception can occur.

A typical error message of the compiler is:

```
java.lang.NullPointerException
.....
at SC.ecomain(SC.java:71)
.....
```

Generally this means that an error occurred at the line number 71 of the compiled java source.

If that is the case, please recompile everything to restart a simulation. For more information about the compiler visit:

<http://www.janino.net/>

The default imports for the compiler are available in a text file and can be changed in the file:

**...\plugins\com.eco.bio7\_1.0.0\bin\imports\compiler.properties** (Windows).

Please consider to make a backup before you change the file !

## 15.4 Notes about ImageJ

It is recommend to visit the website from ImageJ. There are a lot of programming examples and tutorials which help to create image based calculations and analysis. The ImageJ version, used in this application, is based on the version 1.39. The integration of ImageJ followed the approach to change the original as less as possible to get the highest compatibility between the original version and the version of Bio7. ImageJ macros for a startup can be placed in the macros folder (StartupMacros.txt):

**...\plugins\com.eco.bio7.image\_1.39.4\macros.** ImageJ plugins can be placed in the

plugins folder: ...\**plugins\com.eco.bio7.image\_1.39.4\plugins**.

It is important to know that some of the free available plugins (<http://rsb.info.nih.gov/ij/plugins/>) may not work correctly for the embedded ImageJ because of the different GUI (Swing instead AWT!) approach. New ImageJ plugins can be accessed in the plugins menu of the ImageJ view. Simple plugins extend the "plugins" submenu of the ImageJ menu Plugins->Plugins. Plugins located in a \*.jar are accessible and executable with the help of the Control panel (Plugins->Utilities->Control Panel...).

Additionally:

Since Bio7 1.1 it is possible to compile ImageJ plugins directly with Bio7 using the embedded IBM Compiler. To start the compiled plugins the source has to be compiled to the plugins folder or the Bio7 application has to be restarted. After a restart of Bio7 the compiled plugins are accessible in the plugins submenu.

## 16 Literature and Links

### 16.1 Useful Libraries

JScience : <http://jscience.org/>  
Colt : <http://dsd.lbl.gov/~hoschek/colt/>  
Apache commons math: <http://jakarta.apache.org/commons/math/>

### 16.2 Documentation and programming links

Programming:

BeanShell: <http://www.beanshell.org/>  
R: <http://www.r-project.org/>  
Rserve: <http://www.rforge.net/Rserve/>  
ImageJ: <http://rsb.info.nih.gov/ij/>  
Java Compiler: <http://www.janino.net/>

Database:

dbo4: <http://developer.db4o.com/>

Spatial Analysis:

R Spatial Projects <http://sal.uiuc.edu/csiss/Rgeo/>  
Spatstat: <http://www.spatstat.org/>

Forums and mailing lists for information:

<http://www.nabble.com> (search for the different applications BeanShell, R, ImageJ, Janino)

## 16.3 Useful Literature

### Ecological Modelling

---

JØRGENSEN, S.E. (2001) Fundamentals of Ecological Modelling. Elsevier, Amsterdam.

HAEFNER, J (2005) Modeling Biological Systems – Principles and Applications. Chapman&Hall, NY.

GOTELLI, N (2001) A primer of Ecology. Sinauer Associates, Inc., Massachusetts.

GERHARDT M., SCHUSTER, H. (1995) Das digitale Universum. Vieweg, Braunschweig.

DIECKMAN, U. LAW, R., METZ, J. (2000) The Geometry of Ecological Interactions. Cambridge P.

Grimm, V., S. F. Railsback (2005) Individual-based Modeling and Ecology. Princeton Univ. Press.

D. Brown, P. Rothery (1993) Models in Biology: Mathematics, Statistics and Computing. John Wiley & Sons.

Gotelli, N.J. (2001) A Primer of Ecology. 3rd edition. Sinauer Associates, Inc., Sunderland, MA.

Peitgen, Jürgens, Saupe (2004) Chaos and Fractals - New frontiers of science. Springer Verlag, NY.

Gergel, Turner (2003) Learning landscape ecology. Springer Verlag, NY.

### Statistics

---

Venables, W.N., Ripley, B.D. (2002) Modern Applied Statistics with S. Springer Verlag, NY.

Michael J. Crawley (2007) The R Book. Wiley & Sons, Ltd.

Kaluzny, S. P. Vega, S. C. Cardoso, T.P. et al. (1998) S+ Spatial Stats: User's Manual for Windows and UNIX. 1. Auflage. Berlin: Springer.

### Image Analysis

---

Burger, Wilhelm; Burge, Mark (2007) Digital Image Processing in Java. Springer Verlag, London.

## 17. Addendum

View	Id
3Dgrid	com.eco.bio7.3d
Piechart	com.eco.bio7.piechart
Linechart	com.eco.bio7.linechart
Quadgrid	com.eco.bio7.quadgrid
Spreadsheet	com.eco.bio7.spreadsheet
Form	com.eco.bio7.form
State	com.eco.bio7.states
Hexgrid	com.eco.bio7.hexgrid
Custom Chart	com.eco.bio7.custom_chart
Points	com.eco.bio7.points
Image-Methods	com.eco.bio7.image_methods
Nitrate	com.eco.bio7.nitrate
Bsh-Console	com.eco.bio7.bsh
Control	com.eco.bio7.control
Soil Parameter	com.eco.bio7.soil_parameter
Phosphate	com.eco.bio7.phosphate
Water	com.eco.bio7.water
Carbon	com.eco.bio7.carbon
Roots	com.eco.bio7.roots
Custom Controls	com.eco.bio7.custom_controls
Custom 3d	com.eco.bio7.custom_3d
Navigator	org.eclipse.ui.views.ResourceNavigator
Progress	org.eclipse.ui.views.ProgressView
Bookmark	org.eclipse.ui.views.BookmarkView
Tasks	org.eclipse.ui.views.TaskList
Problems	org.eclipse.ui.views.ProblemView
Console	org.eclipse.ui.console.ConsoleView
Properties	org.eclipse.ui.views.PropertySheet
Outline	org.eclipse.ui.views.ContentOutline

To use programmatically:

```
Work.openView("Id");
```

Example:

```
Work.openView("com.eco.bio7.quadgrid");
```

<b>Perspective</b>	<b>Id</b>
2d	com.eco.bio7.perspective_2d
3d	com.eco.bio7.perspective_3d
Database	com.eco.bio7.perspective_database
Image	com.eco.bio7.perspective_image
Soil	com.eco.bio7.perspective_soil
Resource	org.eclipse.ui.resourcePerspective

To use programmatically:

```
Work.openPerspective("Id");
```

Example:

```
Work.openPerspective("com.eco.bio7.perspective_2d");
```

A saved custom perspective is available by it's name !

For example:

Saved as "Custom" ->

```
Work.openPerspective("Custom");
```

### **Bio7 API :**

A small API is integrated in Bio7.

The API is available within the help of Bio7.

## 18. Definition of expressions

**Area of Analysis:** A name for the area in the Points panel which is marked by a green rectangle for spatial point patterns.

**Programmatically:** Using programming to accomplish a task.

**Namespace (BeanShell) :** A namespace in which methods, variables, and imports (class names) live.

**Workspace (R):** A saveable environment in which methods, variables and objects exists.

**Workspace (Bio7):** The location where the projects and files are stored.

**Calculation thread:** A time dependant execution of a compiled method (ecomain) in Bio7 started, e.g. with the Play/Pause button.

**Discretize:** In Bio7 the process of transferring continuous data to discrete data.

**IDE (Integrated Development Environment):** a type of software that assists in developing software.

**Snippets:** In Bio7 a small piece of programming source code to demonstrate some of the Bio7 features.