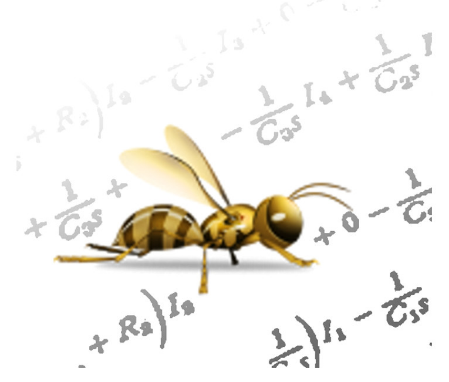
**Models and Model Editor**



**Jacques Régnière**

**Rémi Saint-Amant**

**Ariane Béchard**

**2017**

**Natural Resources Canada**

**Canadian Forest Service**

**Laurentian Forestry Centre**

**P.O. Box 10380, Stn. Sainte-Foy**

**Quebec, QC Canada, G1V 4C7**

Table of Contents

[1. Model Editor 1](#_Toc504052850)

[1.1. Overview 1](#_Toc504052851)

[1.2. How to link a model (dll or exe) to BioSIM 2](#_Toc504052852)

[1.3. Linked Data Editor’s Models page 2](#_Toc504052853)

[1.4. Model Editor dialog 3](#_Toc504052854)

[1.4.1. General tab 3](#_Toc504052856)

[1.4.2. WG input tab 4](#_Toc504052857)

[1.4.3. Specific Site Information tab 5](#_Toc504052858)

[1.4.4. Input tab 5](#_Toc504052859)

[1.4.5. Edit Model Interface dialog 7](#_Toc504052860)

[1.4.6. Output tab 8](#_Toc504052861)

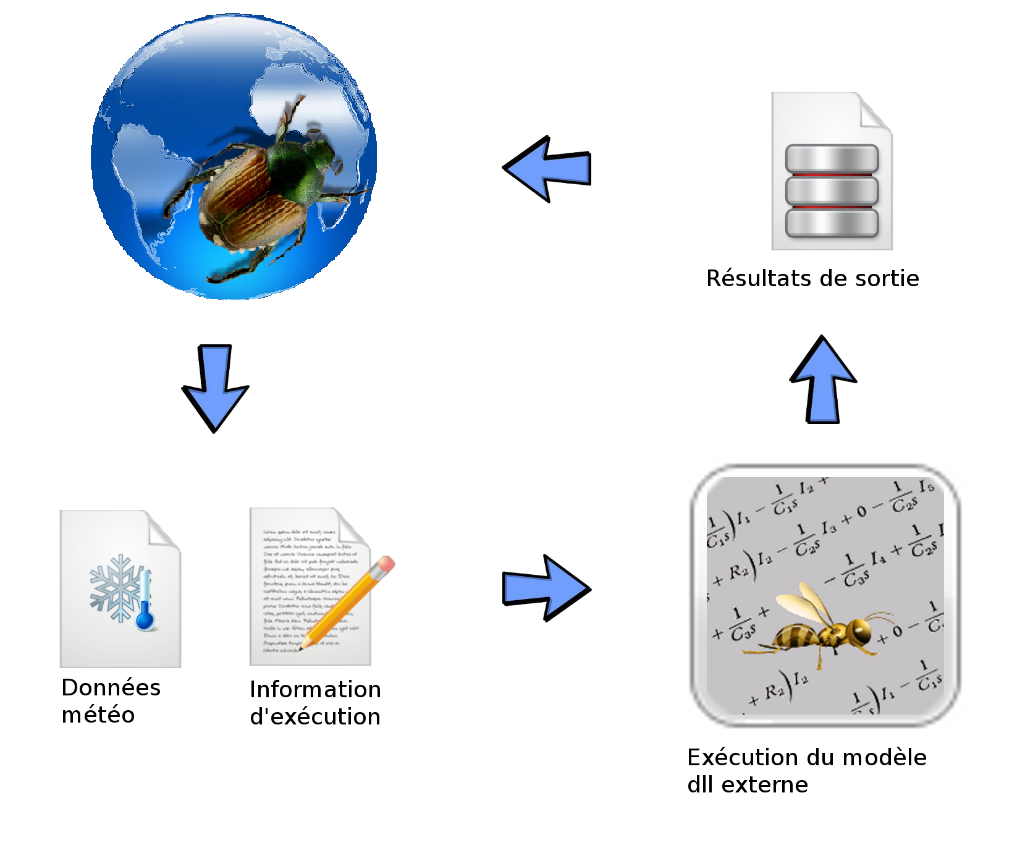
[1.4.7. Source tab 9](#_Toc504052862)

[2. Example of code 11](#_Toc504052863)

# Model Editor

## Overview

Two steps are necessary to create a model in BioSIM. To begin with, the user needs to create a model executable or a dll. Then, he must create a model interface (in BioSIM) to link this dll/exe into BioSIM. The first step requires the user to possess good working knowledge of computer programming in C or C++. Other languages may be used. However, if this is the case, the development team can only provide limited support. If the user is unfamiliar with programming, BioSIM’s programmer can provide assistance in creating the model’s framework (first step). The second step is relatively simple and is explained in this document.



Each time a simulation is executed in BioSIM (), two files are created: a weather file containing data on all the daily variables (for all years) required as input by the model, and a communication file. The communication file contains information concerning the execution of the model (e.g. the names of the input weather and model output files that are to be used). In BioSIM, communication with models can have four different levels of complexity; at the most basic, it takes the form of a file containing a few lines of text; at the most complex, it consists of memory streaming. Memory streaming allows BioSIM to work in parallel or grid computing modes. When using BioSIM’s model-base framework (class CBioSIMModelBase), stream communication between BioSIM and model is very straightforward to program.

The following instructions are intended for users who want to add a new model interface. However, the Model Editor dialog can also be used to view or edit an existing model interface.

## How to link a model (dll or exe) to BioSIM

When a BioSIM-compatible dll or exe has been created, it must be linked to BioSIM through the Model Editor dialog. First, the user must copy the executable file (.dll or .exe) into the …\BioSIM\Models\ sub-directory. Only then can the Model Editor dialog be used to create a new model interface.

## Linked Data Manager’s Models page

The Linked Data Manager’s *Models* page allows the user to create, delete and modify model interfaces.

Once a new model interface has been created it can be opened by double clicking on it in the list field for editing. Already existing model interfaces can be accessed in the same manner.

Buttons and fields available in the *Models* page of this dialog are as follows:

Nouveau New: Add a new model interface to BioSIM; an editable line will appear in the list field allowing the user to enter a name for this new model interface.

Supprimer Delete: Delete a selected model interface.

Éditer Edit: Edit a selected model interface.

Copier_2 Copy: Copy a selected model interface.

**F2** key: Rename a selected model interface.

After adding a new model interface by assigning it a name, the user must select it in the list field and double click on it, or click on the Edit button Modèles_Edit to access the Model Editor dialog and define the model interface.

## Model Editor dialog

The Model Editor dialog can only be accessed via the Linked Data Manager dialog’s *Models* page which in turn can be opened by clicking on the Linked Data Manager button  on the main window’s toolbar, or by selecting [Tools] [Linked Data Manager] [*Models* page].

The Model Editor dialog is composed of six tabs:

General_tab_Models_page [*General*](#_General_tab:) tab: Enter general information pertaining to a model (model and executable names, model description, etc.).

WGInput_tab_Models_page [*Weather Generator Input*](#_TG_input) tab: Set the type of weather data to be sent by BioSIM as input to the model.

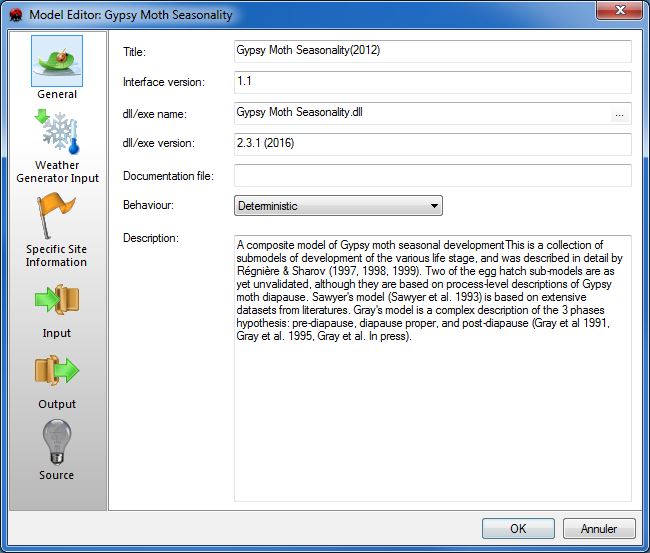
*[Specific Site Information](#_Specific_Site_Information)* tab:

Input_tab_Models_page [*Input*](#_Model_parameters)tab: Create an interface which can be used to set various model parameter values (these parameters are model-specific).

Outputs_tab_Models_page [*Output*](#_Output_variables) tab: Define the number and nature of model output variables.

Source_tab_Models_page [*Sources*](#_Model_source)tab: Enter information on intellectual property or credits concerning the model.

### General tab



The *General* tab of the Model Editor dialog is used to enter general information such as the name of the model and of its executable file (exe or dll).

Buttons and fields in the *General* tab are as follows:

**Title** field: Enter the name of the model

**Interface version** field: Enter the version of the interface.

**dll/exe name** field: Enter the name of the model’s executable file (.dll or .exe). The browse button (…) to the right of this field can be used to locate the model’s executable file. Model executables must be placed into the …\BioSIM\Models\ subdirectory.

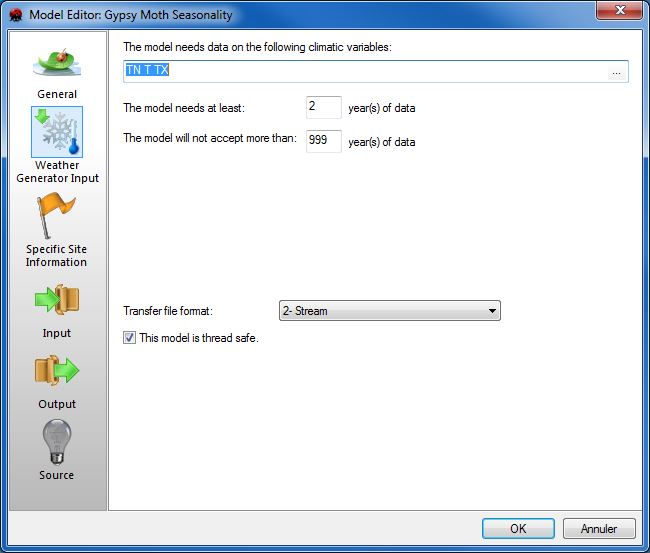
**dll/exe version** field: If the executable file was created using BioSIM’s framework, the version of the file will be extracted from the dll and displayed here.

**Documentation file** field: Provides a link to the model documentation (usually a pdf file). The documentation file of a model must be located in the same ...\BioSIM\Models\ subdirectory as the model executable and its interface.

**Behaviour** field (drop-down list): Specifies whether the model is deterministic or stochastic. Deterministic models always return the same output when provided with the same input, as they contain no random elements. Stochastic models, on the other hand, contain random elements, and therefore never return exactly the same output when provided with the same input. This is information about the model, and does not affect its behaviour.

**Description** field: Enter a general description of the model

### WG input tab



The *WG Input* tab of the Model Editor dialog is used to define the type of weather input required by the model.

**The model needs these climatic variables** field**:** For each model interface, the user needs to specify the climatic variables (browse climatic variables list by …) the model needs as input by checking Cocher the relevant variables (and only those). Please note that the snow and solar radiation variables are derived (not contained in weather databases).

**The model needs at least Modèles_Box_1 year(s)** field**:** The user must specify the minimum number of years of data that the model requires. Models in BioSIM require at least one year of climatic data. The default value of this parameter is thus 1.

**The model will not accept more than Modèles_Box_999 year(s)** field: If there is a limit to the number of years of weather input a model can accept, the user must enter it in this field. If not, the default value of this parameter is 999 years which corresponds to the maximum number of years BioSIM will accept.

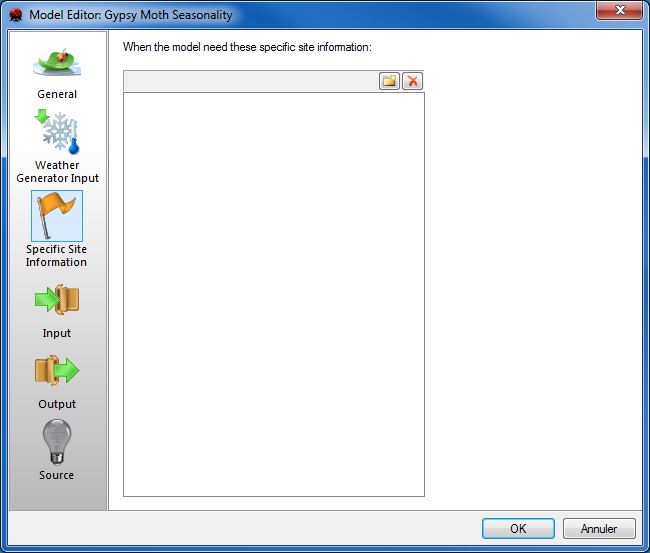
It is recommended to create models that can process an unlimited number of years. However, under certain circumstances it can be useful to limit the number of years the model can process. For example, the current version of the gypsy moth phenology model in BioSIM need at least two years. The number of years needed by a model is determined by the model’s programmer. If an invalid number of years is entered into the Weather Generator Input dialog, BioSIM will warn the user that the model needs to receive a valid number of years.

**Transfer file format** field (drop-down list): Select a communication file format; in BioSIM, there are four levels of complexity to communication files. The user must choose from the following options:

1. Text (by file)
2. XML (by file)
3. Stream (by memory streaming)

This model is “thread safe” checkbox Cocher: When a model is “thread safe”, it means that BioSIM can execute it in parallel (multiple runs can be sent to the dll at the same time). The BioSIM model framework is thread safe. If the user is uncertain whether or not a model is thread safe, he can contact the BioSIM programmer.

### Specific Site Information tab

The specific sites information tab allows you to add additional columns in the locations file. These columns have a specific name and become compulsory to execute the model.

For example, the model could need the type of ground whose being TypeSol headers could to be able to bye executed.

Modèles_New New: Add a new column.

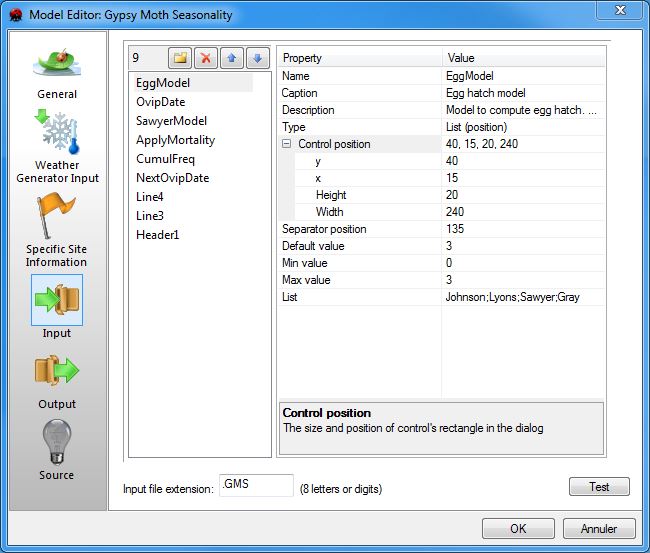
Modèles_Delete Delete: Delete a selected column.

### Input tab

The *Input* tab is used to define the input parameters specific to the model. This tab has a secondary dialog: the Edit Model interface, used to design the model’s graphic user interface.

The Model Editor’s *Input* tab is used to define model interface elements, while the Edit Model interfacedialog (see below) is used to design the model’s specific interface dialog as it will appear in BioSIM when the model is used in the definition of a simulation.

Please note that to create a new model interface; the user must have the administrative privileges to BioSIM’s Model directory.

The left hand side panel lists the model’s current parameters. It can be used to perform the following:

Modèles_New New: Add a new parameter to the interface.

Modèles_Delete Delete: Delete a selected parameter from the interface.

Modèles_up Move Item up: Move a selected parameter up in the list.

Modèles_down Move Item down: Move a selected parameter down in the list.

The order in which the parameters appear in this list is very important; because BioSIM will transmit parameter values to the model in the order they are listed. This order is determined by the model’s code.

The right hand side panel displays the properties of a selected parameter. Fields in this panel are as follows:

**Name** field: Internal parameter name, must not contain spaces or special characters (e.g. +,-,\*,/). Used when the parameter is to be varied in the Parameter Variation dialog (accessed through the Simulation Editor dialog using the buttonParameters_Variations). The name in the **Name** field corresponds to the name given to the parameter in the left hand side panel and must be edited there and not in the right hand side panel (either by double clicking on it, or using the F2 key).

**Caption** field: Text that will appear in the Model interface dialog to the left of the parameter’s entry field.

**Description** field: A useful description that will appear in the model’s description text file (accessible by clicking the Description button in the Simulation Editor dialog).

**Type** field (drop-down list): Select each parameter’s type from one of the following options:

* **Boolean** True/False
* **Integer** Integer (whole number)
* **Real** Real (decimal)
* **Text** Text
* **File** File path to be sent to the model. Creates an edit box with a browse button on the right hand side for file searching.
* **List (position)** the model will receive a 0-based index of the selection
* **List (text)** The model will receive a text of the selection
* **List (CSV)** The model will receive a csv file list of the selection

The three following parameters are used for graphical interface design or to provide information to the user.

* **Header** Read only text that will appear in the model interface dialog. Provides information to the model user.
* **Line** Horizontal line used as a separator in the design of the user interface.
* **Caption** A title or brief explanation appended to the model

**Control rectangle** field: Allows the user to modify the default size and position of a selected parameter’s rectangle in the interface.

**Separator position** field: The user can move the position of the separator (between the parameter’s name and the value’s entry field) by changing the value in this field.

**Default value** field: A selected parameter’s default value.

**Min Value** field: A selected parameter’s minimum valid value.

**Max value** field: A selected parameter’s maximum valid value.

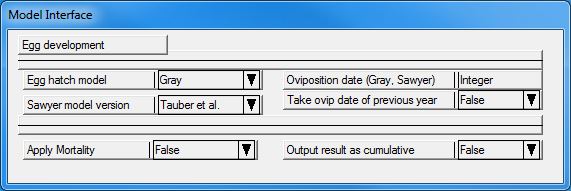
**List** field (drop-down list): List value separated by a semi-colon. Only used with “List” parameter type.

The greyed read-only field at the bottom of the right hand side panel displays information about the property selected in the upper part of the panel.

**Input file extension** field: The model’s input file extension. Models that have identical input variables can use the same extension.

Test Displays how the model interface will appear in BioSIM.

### Edit Model Interface dialog



The Edit Model Interface dialog is used to change the appearance of a model’s input dialog. The user can change the size and the position of all interface elements by clicking-and-dragging them using the mouse pointer.

To add a new parameter in a model’s interface, the user must click on the New button Modèles_New (in the left panel of the Model Editor’s *Inputs* tab) and select the type of parameter to be added from the drop-down list of the **Type** field (in the right hand side panel). A new rectangle will appear in the left hand side list field.

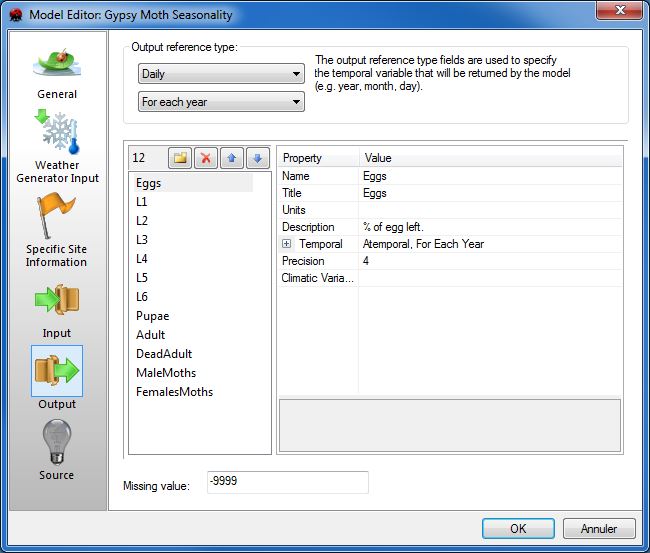
Once the parameter has been assigned a name, it will appear in the Edit Model Interface dialog (the name of the parameter will be visible in the left part of the parameter box while the right part will be used to enter the value of the parameter). The user can move the rectangle anywhere in the window and/or resize it using the double-headed arrow cursor.

It is possible to change the size of the entry fields in each part of the rectangle by dragging the dividing sash.

Note: There is no relationship between the position of model parameter fields in the model’s graphical user interface (the Model Interface dialog) and the order in which the parameters are transmitted to the model by BioSIM. That order is specified by the order in which parameters are listed in the *Input* tab of the Model Editor dialog.

### 

### Output tab

The *Output* tab is used to specify the output time reference (e.g. date) format and the list (names) of the model’s output variables. The number and order of the variables in this list must be identical to that of the output variables (other than time) generated by the model.

**Output reference type** fields (drop-down lists)**:** Models in BioSIM generate output in time steps. Therefore, each output line usually contains a time reference (such as a date, often Year, Month, Day). In this field the user must define the type of the output line reference the model uses. It is particularly important for BioSIM to know how many output columns will make up this output reference because reference columns are used in output analyses. The following is a list of the output reference types available which the user can choose from when creating a model:

|  |  |  |
| --- | --- | --- |
| Nb of reference columns | For each year | Overall years |
| Annual | 1 (year) | 0 |
| Monthly | 2 (year, month) | 1 (month) |
| Daily | 3 (year, month, day) | 2 (month, day) |
| Hourly | 4 (year, month, day, hour) | 3 (month, day, hour) |
| Atemporal | 1 (#ref) | 0 |

**Model output variable definitions:**

The left hand side panel lists all the model’s output variables. It can be used to perform the following:

Modèles_New New: Add a new output variable.

Modèles_Delete Delete: Delete a selected output variable.

Modèles_up Move Item up: Move a selected output variable up in the list.

Modèles_down Move Item down: Move a selected output variable down in the list.

The right hand side panel displays the various properties of a selected variable. Fields in this panel are as follows:

**Name** field: Internal variable name. Internal variable name, must not contain spaces or special characters (e.g. +,-,\*,/,etc.). Used in analysis definition. The name in the **Name** field corresponds to the name given to the parameter in the left hand side panel and must be edited there and not in the right hand side panel (either by double clicking on it, or using the F2 key).

**Title** field: Text that will appear in the result column header either in the *Data* tab of the main window where the results can be viewed, or in export files (text or spreadsheet) when the results are exported using the Export Now Exporter or To Spreadsheet Vers_chiffrier_ button in the Export window.

**Description** field: A useful description that will appear in the model’s description text file (accessible by clicking the Description button in the Simulation Editor dialog).

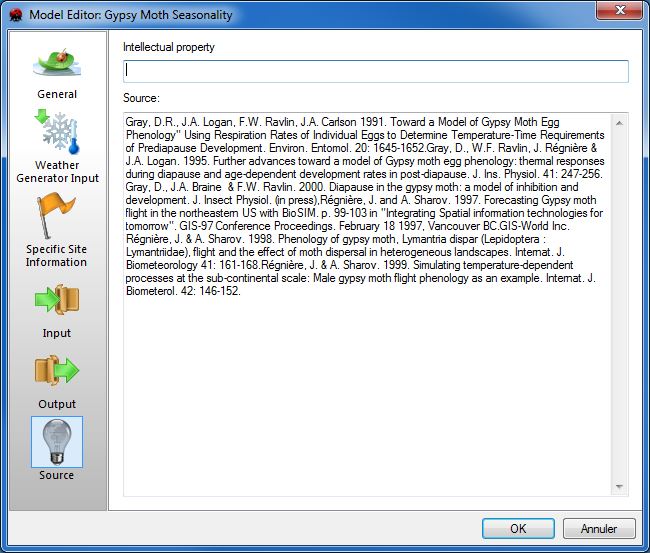
**Precision** field: The number of decimal places used to display or export results

The greyed read-only field at the bottom of the right hand side panel displays information about the property selected in the upper part of the panel.

**Missing value** field: Missing value the model will return.

### Source tab

The Model Editor’s *Source* tab allows the user to enter (or view) information pertaining to the intellectual property and the source of the simulation model.

**Intellectual property** field**:** The information in this field will appear as a line of text in the Execute Message Log window each time the model is run. It is here that the user can enter the identity of the model’s designer as well as any other intellectual property claims or constraints attached to the model: the length of this field is limited to 128 characters.

**Source** field**:** This field is used to list literature references, credits for development, acknowledgements and other useful information concerning the model. This information can be displayed in BioSIM by clicking the Description button in the Simulation Editor dialog.

# Example of code

The following example is a Degree-day program in C++:

When using BioSIM’s framework, it is relatively easy to create new models since all IO is managed by the framework.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// DegreeDay summation

//

// Jacques Régnière

// Canadian Forest Service

//

// Programmer: Rémi Saint-Amant

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// File: DegreDay.cpp

//

// Class: CDegreeDay

//

// Description: CDegreeDay is a BioSIM model that computes heating degree day.

// The model accepts any number of years and returns the

// daily degree day value or the cumulative sum.

//

// Input parameters:

// FirstDay: the first day of summation (1..365)

// LastDay: the last day of summation (1..366)

// Threshold: base threshold of the summation.

// SummationType: can be cumulative or not.

//

//

// Output variable:

// Degree days or their summation.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// 08/02/2008 Rémi Saint-Amant Creation from old code

// 06/06/2011 Rémi Saint-Amant Update

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "DegreeDay.h"

#include "EntryPoint.h"

//this line links this model with the entry point of the DLL

static const bool bRegistred =

CModelFactory::RegisterModel( CDegreeDay::CreateObject );

//The constructor of the class

CDegreeDay::CDegreeDay()

{

//NB\_INPUT\_PARAMETER and VERSION are 2 framework variables

NB\_INPUT\_PARAMETER = 4; //set the number of parameters for this model

VERSION = "2.1 (2011)"; //set the version of this model

//This model has 4 input parameters

m\_firstDay=-1;

m\_lastDay=-1;

m\_threshold=4;

m\_summationType=CUMULATIVE;

}

//The destructor of the class

CDegreeDay::~CDegreeDay()

{

}

//Create the output vector type (in this case, a vector with one output variable)

typedef CModelStatVectorTemplate<1> CDDStatVector;

//Called by the framework to implement daily computation

ERMsg CDegreeDay::OnExecuteDaily()

{

ERMsg msg;//define error message result

//Create an output vector that has the same size as the number of input days

//and the first time reference (first day)

CDDStatVector stat(m\_weather.GetNbDay(), m\_weather.GetFirstTRef());

//for all days

for(int y=0; y<m\_weather.GetNbYear(); y++)

{

//compute the last day of accumulation: Because the last day can

//be greater than the number of days in the year.

int lastDay = m\_lastDay;

if( lastDay >= m\_weather[y].GetNbDay())

lastDay = m\_weather[y].GetNbDay()-1;

//Degree-day summation

double DDSum = 0;

//for the first day of accumulation to the last day of accumulation

for(int jd=m\_firstDay; jd<=lastDay; jd++)

{

//in cumulative mode, do sum; otherwise take the value itself

if( m\_summationType == CUMULATIVE )

DDSum += m\_weather[y].GetDay(jd).GetDD(m\_threshold);

else DDSum = m\_weather[y].GetDay(jd).GetDD(m\_threshold);

//set the degree-day sum into the result vector

stat[ m\_weather[y].GetFirstTRef()+jd ][0] = DDSum;

}

}

//Set the results of the vector into the framework

SetOutput(stat);

//returns an error message (when applicable)

return msg;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//this method is called by the framework to load the parameters

ERMsg CDegreeDay::ProcessParameter(const CParameterVector& parameters)

{

ASSERT( m\_weather.GetNbYear() > 0);

ERMsg msg;

int cur = 0;

//read the 4 input parameters: must be in the same order as the

//model's interface. Julian days are in zero base.

m\_firstDay = parameters[cur++].GetInt()-1;

m\_lastDay = parameters[cur++].GetInt()-1;

m\_threshold = parameters[cur++].GetReal();

m\_summationType = parameters[cur++].GetInt();

//perform verifications

\_ASSERTE( m\_firstDay >= 0 && m\_firstDay < 365 );

\_ASSERTE( m\_lastDay >= 0 && m\_lastDay < 366 );

\_ASSERTE( m\_summationType >= 0 && m\_summationType < NB\_SUMMATION\_TYPE);

return msg;

}