

CORMAS

Common-pool Resources and Multi-Agents Systems

Building a Cormas model from scratch step by step

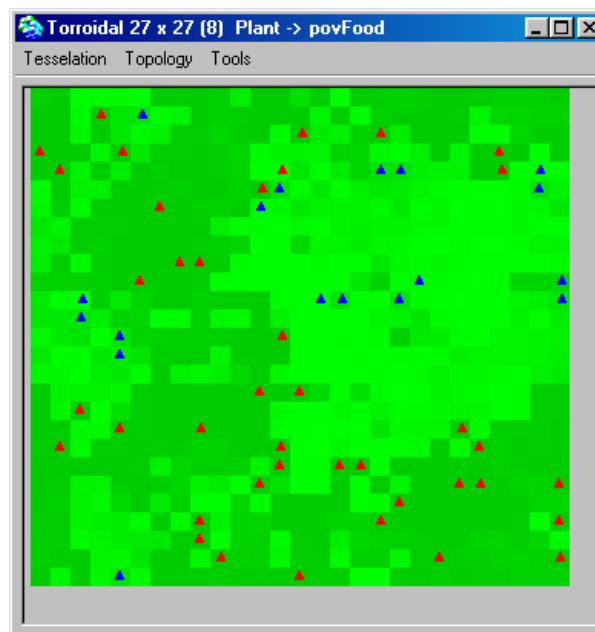


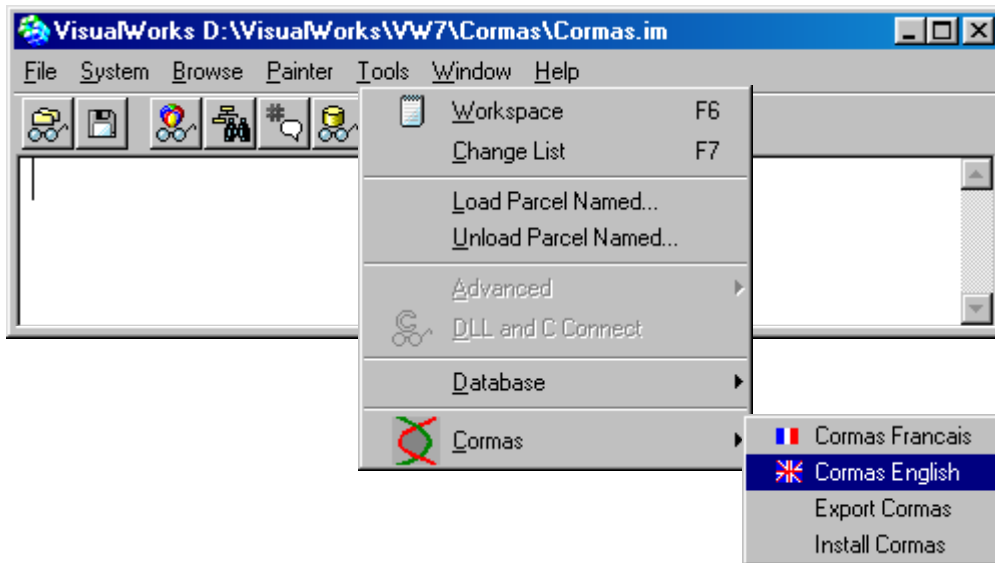
Table of contents

<i>Building a Cormas model from scratch step by step</i>	1
1. Opening Cormas	4
2. Creating a new Cormas model	4
3. Designing a spatial entity holding a renewable resource	5
3.1. Setting the parameters of a logistic growth	6
3.1.1. Create the new class variables with the contextual menu of the code editor	6
3.2. Adding a new attribute named “energy”	8
3.2.1. Create the new attribute with the contextual menu of the code editor	8
3.2.2. Create the accessing methods	8
3.3. Write a logistic growth method in a new protocol	9
3.4. Write the basic step method in the “control” protocol	10
4. Designing an agent foraging the resource	10
4.1. Setting the life-history parameters of any forager agent	11
4.1.1. Create the new class variables with the contextual menu of the code editor	11
4.1.2. Add two attributes named “energy” and “strategy” and create their accessors.....	11
4.2. Create a new protocol named “biology” and write the biological methods	11
4.3. Create a protocol named “control” and write the “step” method of the forager agent	12
5. Designing methods to observe the entities	13
5.1. Assigning a green intensity according to the energy level of the plant	13
5.1.1. Add a “point of view” method named “povFood”	13
5.1.2. Write the code of the “povFood” method	13
5.1.3. Adjust the write-access method of all the attributes involved in “pov” methods.....	14
5.2. Assigning a colour to distinguish foragers according to their strategy	15
5.2.1. Add a “pov” method named “povStrategy”	15
5.2.2. Add the symbols that may be returned by the “pov” method	15
5.2.3. To each added symbol, associate a colour and a shape.....	15
6. Designing “probes” to record the variations of markers	16
6.1. Open the “probes” definition window from the main Cormas interface	16
6.2. Add probes based on attributes of the model (global level)	16
6.3. Write the code for the first one: a cumulative value over a collection	16
6.4. Define 2 other probes : the number of restrained and unrestrained foragers	17
7. Designing control methods for the scheduling of simulation experiments	17
7.1. Create attribute agentsInitialNumber	17
7.2. Write, in a new protocol (“entities creation”), a method to create the forager agents	18
7.3. Write (in the “init” protocol) 3 different methods to be used as initial situations for simulation experiments	18
7.4. Write (in the “control” protocol) a step method to be executed at each time-step of simulation	19
8. Simulating the model	19
9. Saving, loading and versioning the model	22
9.1. Saving and versioning	22
9.2. Loading the model	23
10. Analysing the model	23
10.1. Choose the parameter's values	24
10.1.1. Choose one value.....	25
10.1.2. Choose a set of values.....	25
10.2. Output format	27
10.2.1. Ascii or Excel format.....	27
10.2.2. Data base format	28

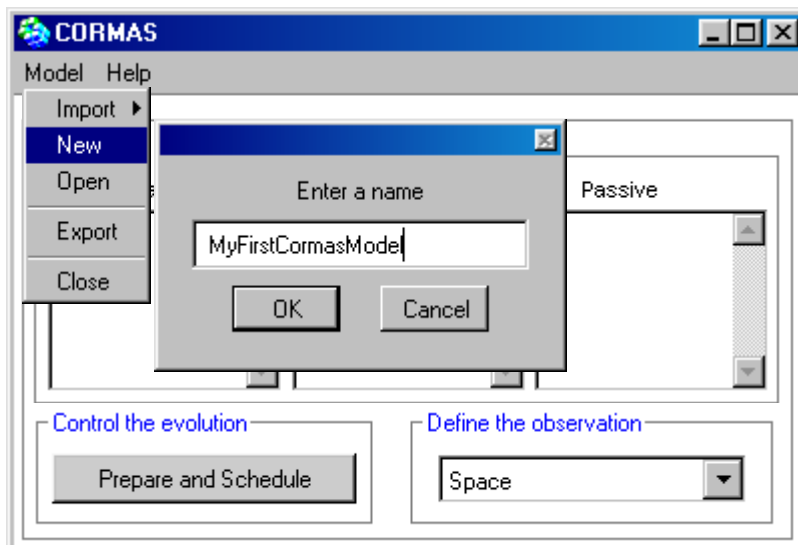
11. Appendices 33
11.1. How to connect VisualWorks to Access 33
 11.1.1. Set up the ODBC driver ODBC..... 33
11.2. UML : Class diagram of Cormas - Entities category..... 36

The model we present here and that you will build, is inspired from a paper by Pepper and Smuts, "Evolution of Cooperation in an Ecological Context". This ECEC model can be found on the Cormas web site: <http://cormas.cirad.fr>

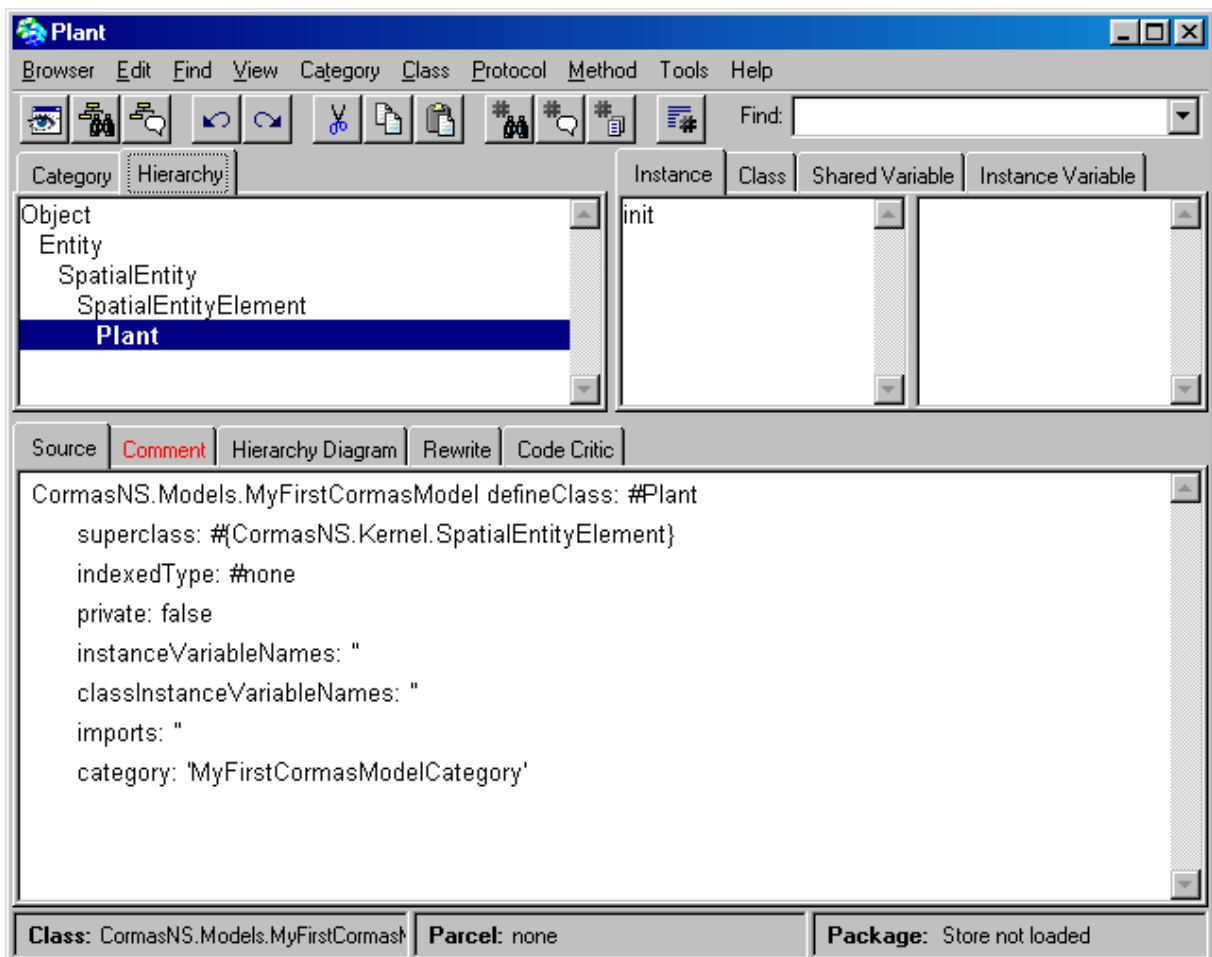
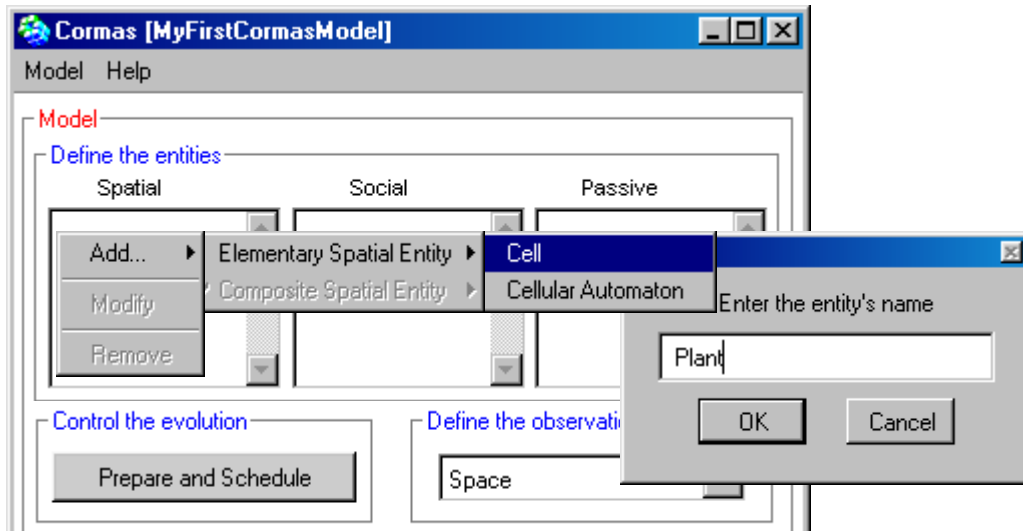
1. Opening Cormas



2. Creating a new Cormas model



3. Designing a spatial entity holding a renewable resource



3.1. Setting the parameters of a logistic growth

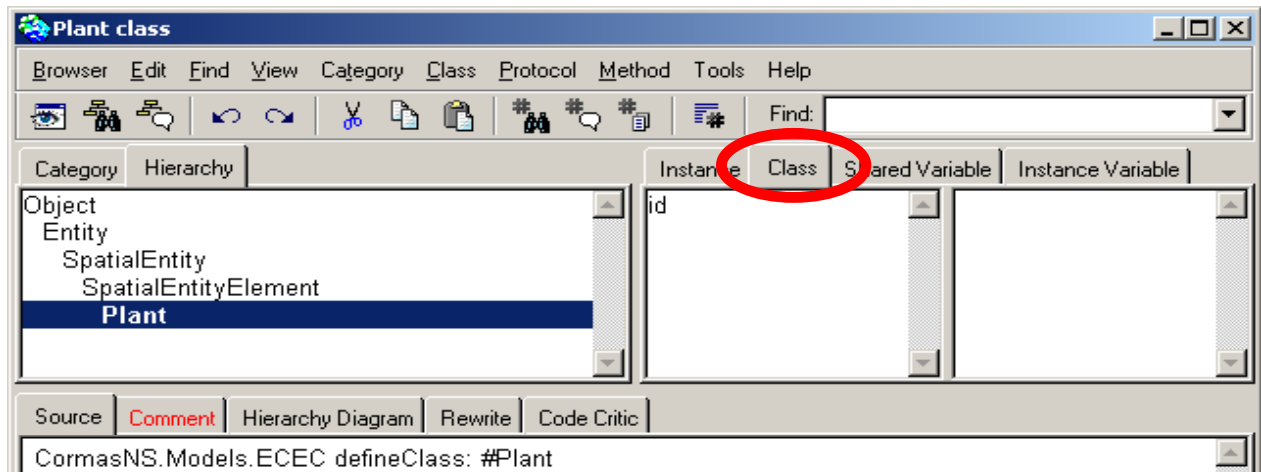
$$x_{t+1} = x_t + rx_t \left(1 - \frac{x_t}{K}\right)$$

The logistic equation needs 2 parameters: the carrying capacity “*k*” and the growth rate “*r*”.

The values of these parameters are equals for each Plant instances. We can defines them as class variables.

3.1.1. Create the new class variables with the contextual menu of the code editor

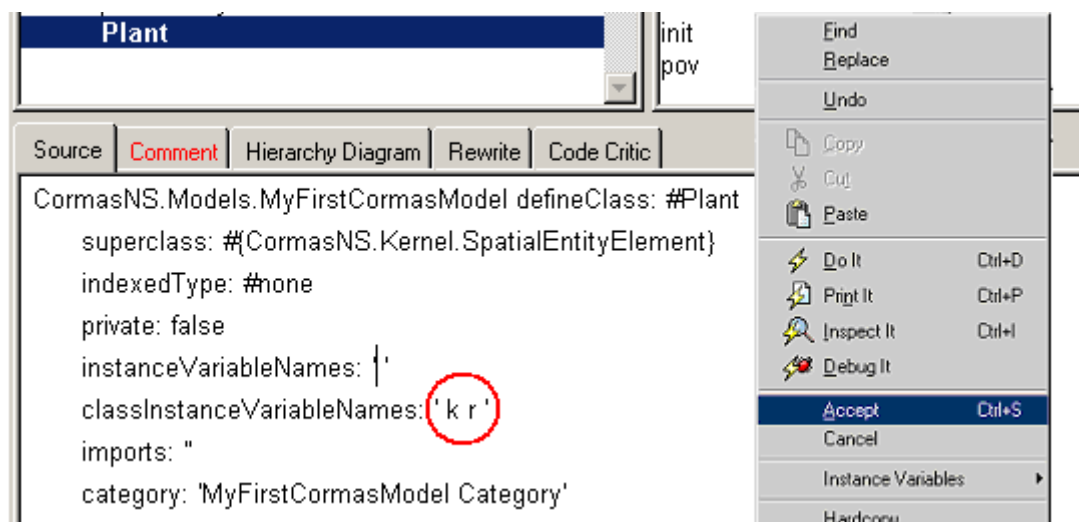
Switch from “Instance” to “Class”.



3.1.2. Create the attributes

Create the carrying capacity “*k*” and “*r*”:

- write them between the quotes on the line "classInstanceVariableNames:", then
- Accept : right click and select "Accept"



3.1.3. Create the accessing methods

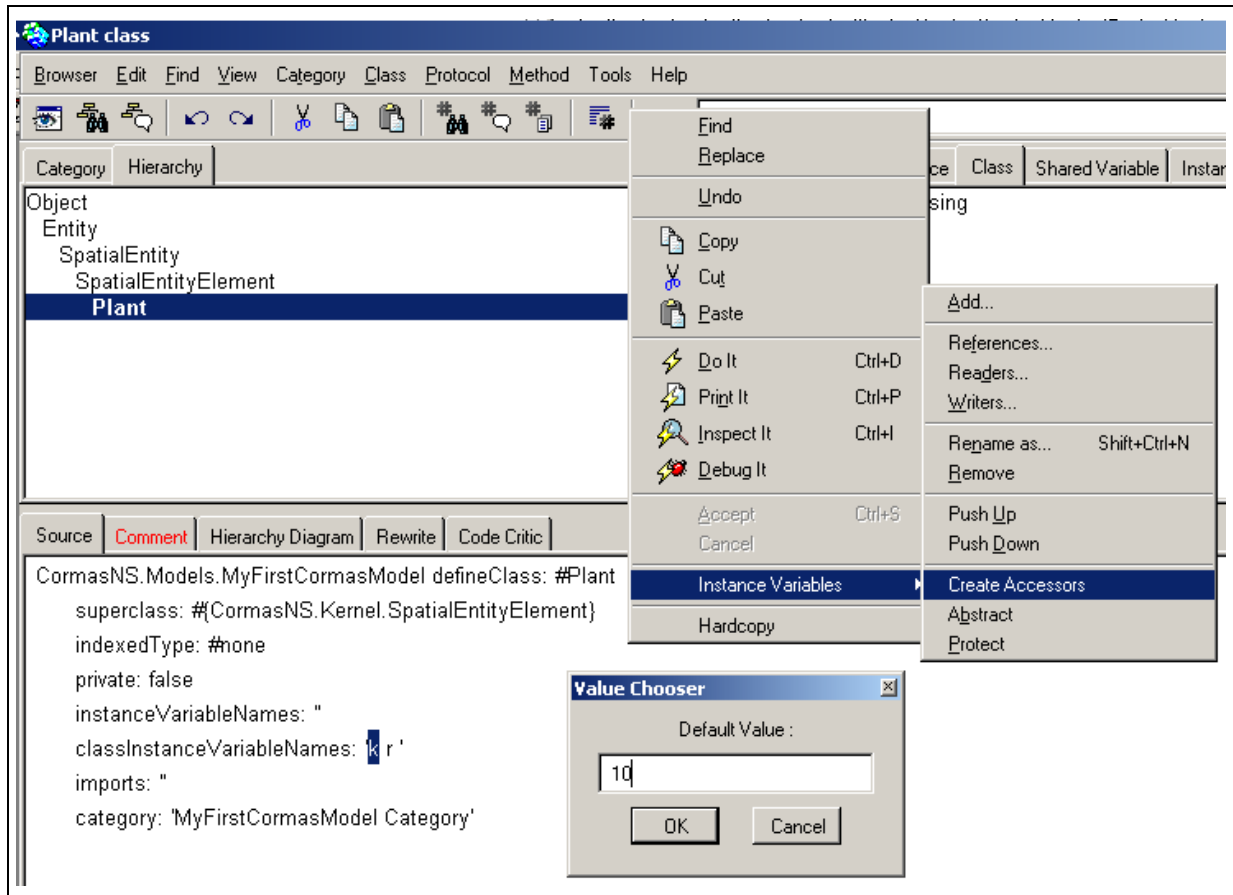
Select the *k* variable then, with right button,

select "Instance Variables" → "create Accessors".¹

A new window pops up asking you to define the default value of this attribute k .

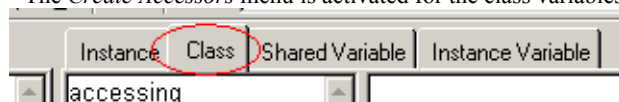
Set the value of k equal to 10.

Then click on "OK". If you click on "Cancel" button, the accessors are created with *nil* as default value.



In the same way, let the growth rate parameter " r " be equal to 0.2 (Select the r variable then, with right button, select "Instance Variables" → "create Accessors". In the new pop-up window set the default value of r equal to 0.2. Then click on "OK").

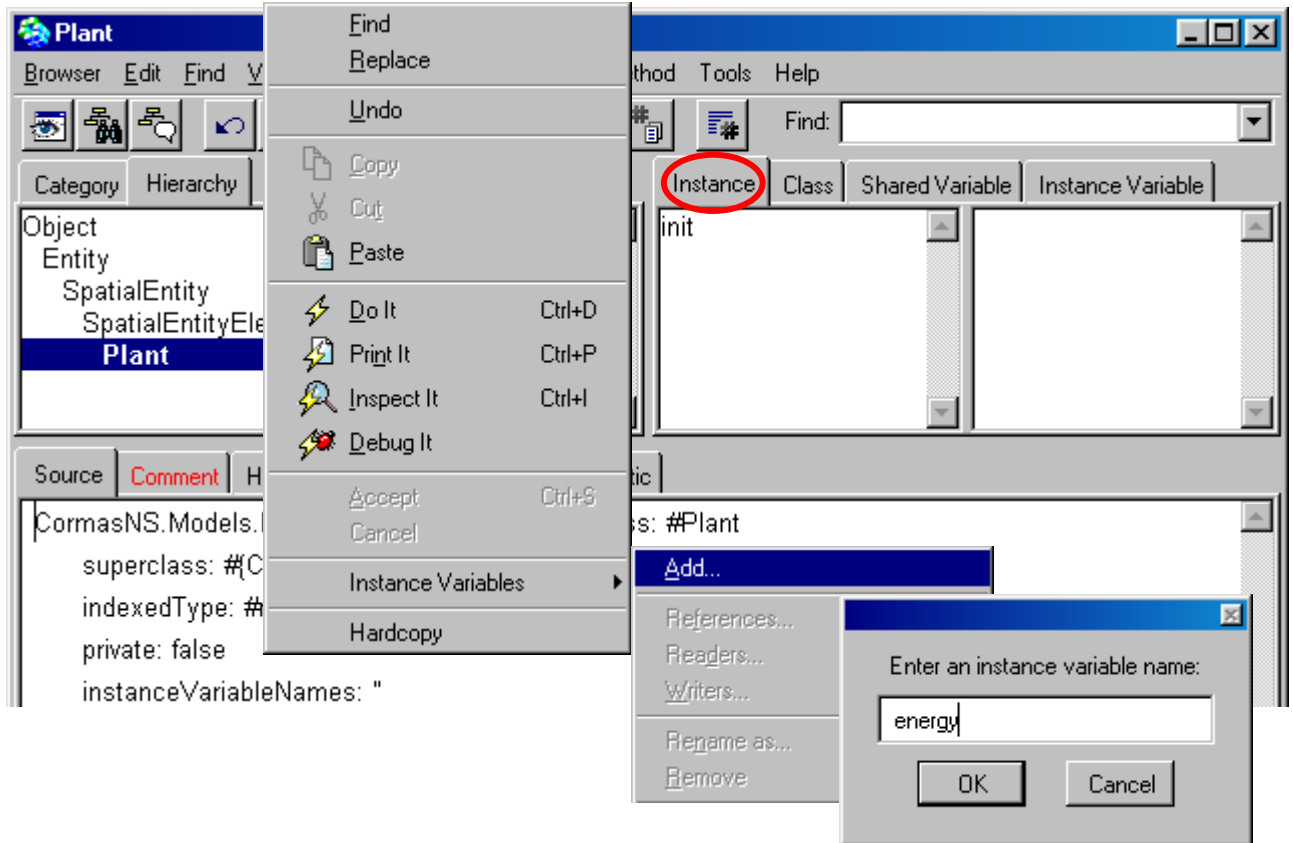
¹ The *Create Accessors* menu is activated for the class variables only when the Class tab is selected.



3.2. Adding a new attribute named “energy”

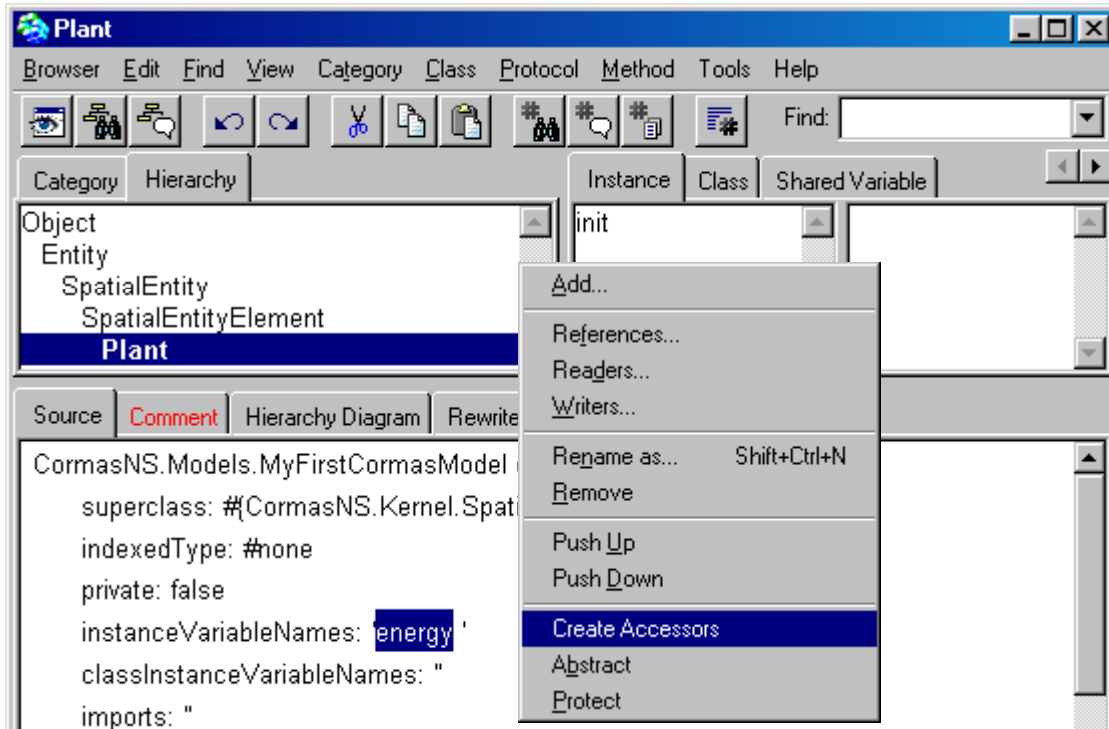
Switch back from the “Class” to the “Instance” level.

3.2.1. Create the new attribute with the contextual menu of the code editor



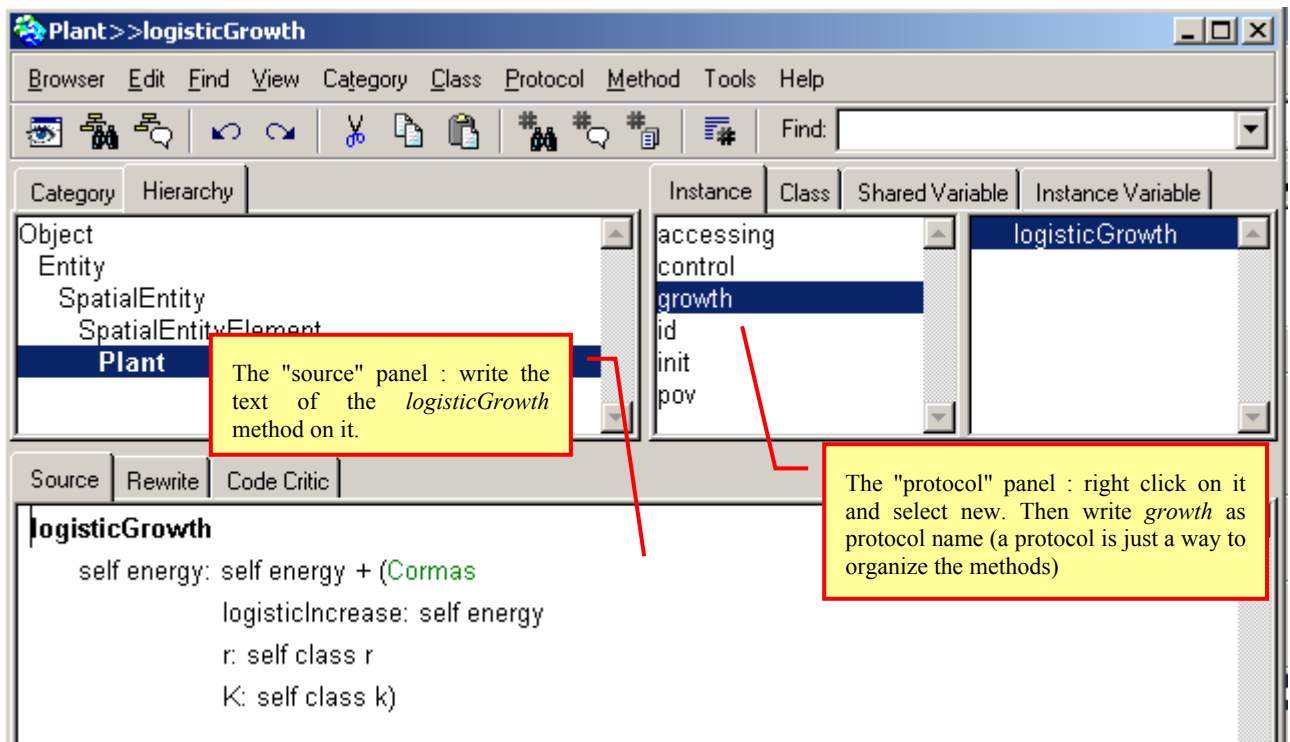
3.2.2. Create the accessing methods

Create the accessing methods of the energy attribute and set the default value equals to 0.



3.3. Write a logistic growth method in a new protocol

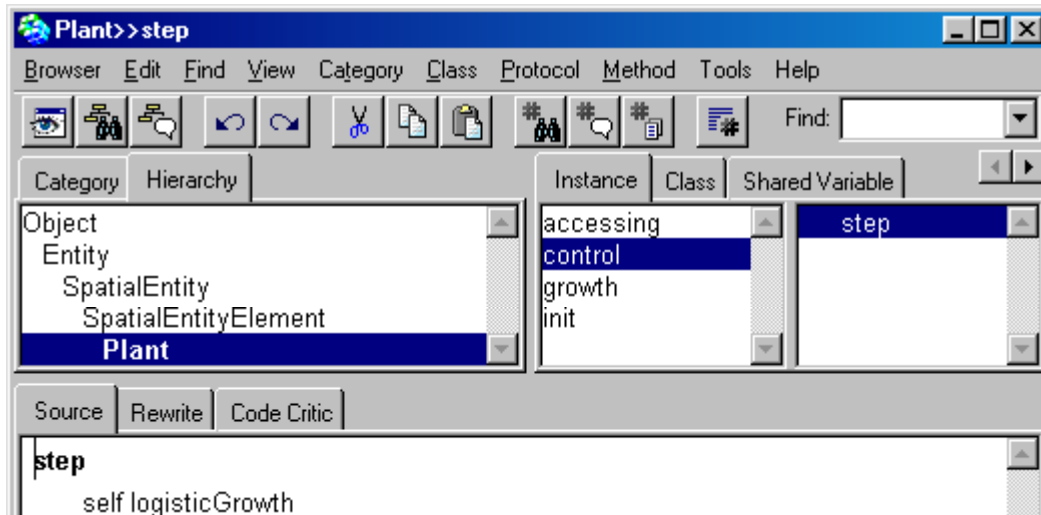
A protocol is just a way to organize the methods. To create a new protocol, right click on the "protocol" panel and select "new". Then write *growth* as protocol name.



To create a new method into the "growth" protocol, remove the text on the "source" panel and write your method on it, then accept (right button → Acept or Ctrl S). The *logisticGrowth* method uses a static method from Cormas class (logisticIncrease:r:K:).

3.4. Write the basic step method in the “control” protocol

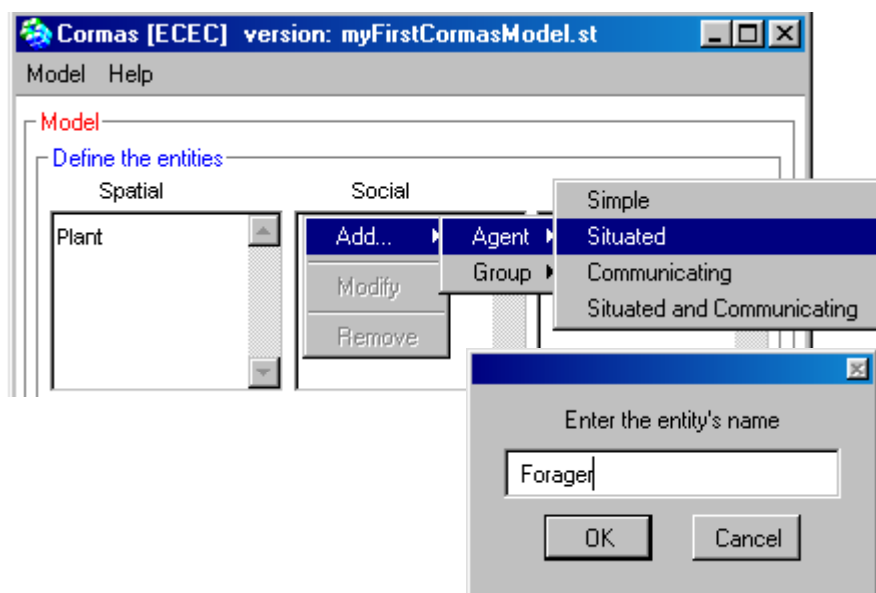
In the same way as previously, create a new protocol called *control* and the *step* method in it.



You can now close the browser "Plant".

4. Designing an agent foraging the resource

Let us define the Forager agent as a kind of situated agent :



As you can see in the new browser, Forager inherits from *AgentLocation*

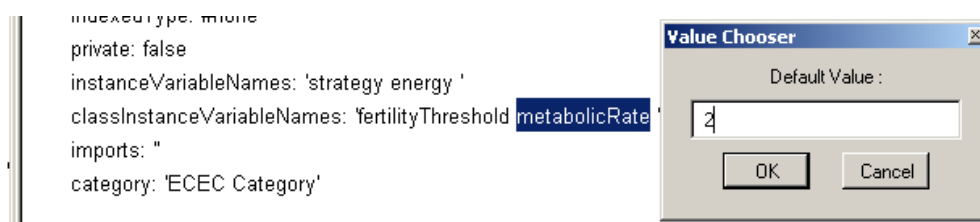
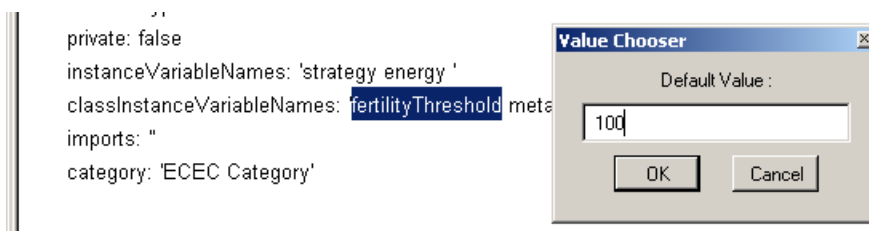
4.1. Setting the life-history parameters of any forager agent

The values of these parameters are equal for each agent. We can define them as class variables as describe in chapter 1 (Setting the parameters of a logistic growth).

4.1.1. Create the new class variables with the contextual menu of the code editor

Switch from “Instance” to “Class”. At this Class level, create the new class variables *fertilityThreshold* and *metabolicRate* (write them between the quotes on the line "classInstanceVariableNames:" and accept).

Then create the accessors with 100 as default value for *fertilityThreshold* and 2 for *metabolicRate* (Select the variable then, with right button, select "Instance Variables" → "create Accessors". In the new pop-up window set the default value of *fertilityThreshold* equal to 100. Then click on "OK")².



4.1.2. Add two attributes named “energy” and “strategy” and create their accessors

Proceed like in section 3.2.

Set the default value of *energy* equal to 50.

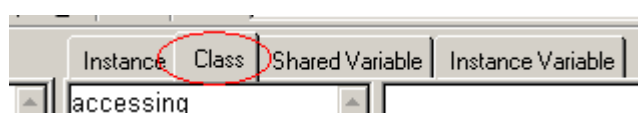
Click on Cancel for the default value of *strategy*.

4.2. Create a new protocol named “biology” and write the biological methods

harvestRate

```
self strategy = #unrestrained ifTrue: [^0.99].
self strategy = #restrained ifTrue: [^0.5]
```

² Remember that the *Create Accessors* menu is activated for the class variables only when the Class tab is selected.



eat

```
| qty |  
  qty := self patch energy * self harvestRate.  
  self energy: self energy + qty.  
  self patch energy: self patch energy - qty
```

move

```
(self patch neighboursMaxOf: #energy) < self class metabolicRate  
  ifTrue: [self randomWalkConstrainedBy: [:c | c noOccupant]]  
  ifFalse: [self walkToMaxOf: #energy constrainedBy: [:c | c noOccupant]]
```

reproduce

```
| newForager |  
  self energy > self class fertilityThreshold  
  ifTrue:  
    [newForager := self class new.  
     newForager strategy: self strategy.  
     self energy: self energy - newForager energy.  
     newForager moveTo: self nearestEmptyLocation].  
  ^newForager
```

die

```
self energy <= 0 ifTrue: [self dead: true].
```

updateEnergy

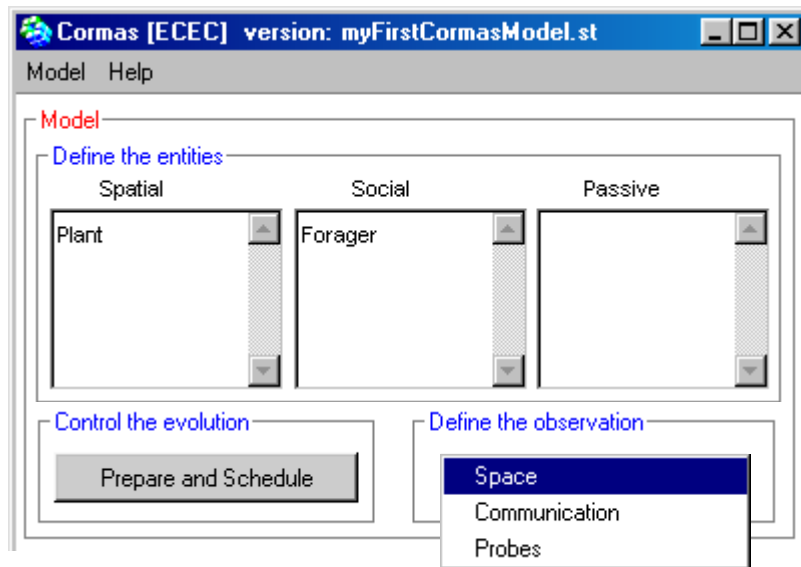
```
self energy: self energy - self class metabolicRate
```

4.3. Create a protocol named “control” and write the “step” method of the forager agent

step

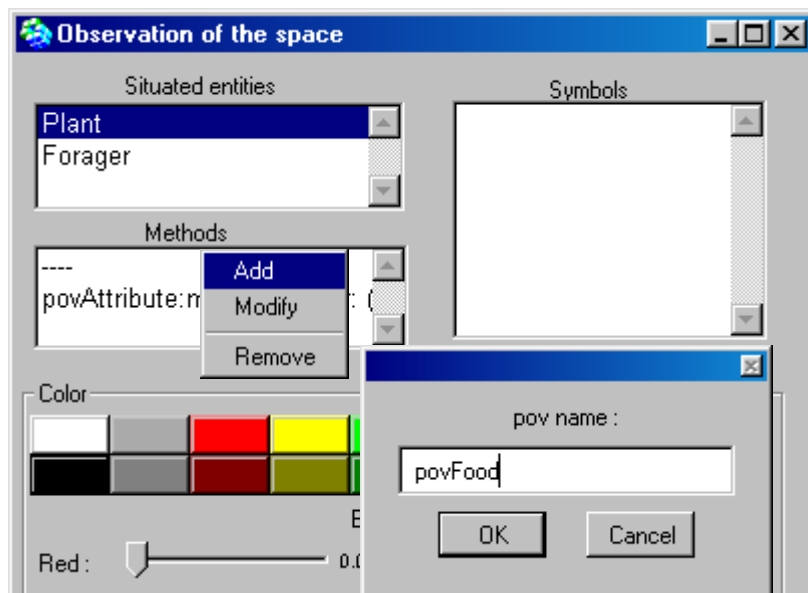
```
| newForager |  
  self updateEnergy.  
  self move.  
  self eat.  
  newForager := self reproduce.  
  self die.  
  ^newForager
```

5. Designing methods to observe the entities



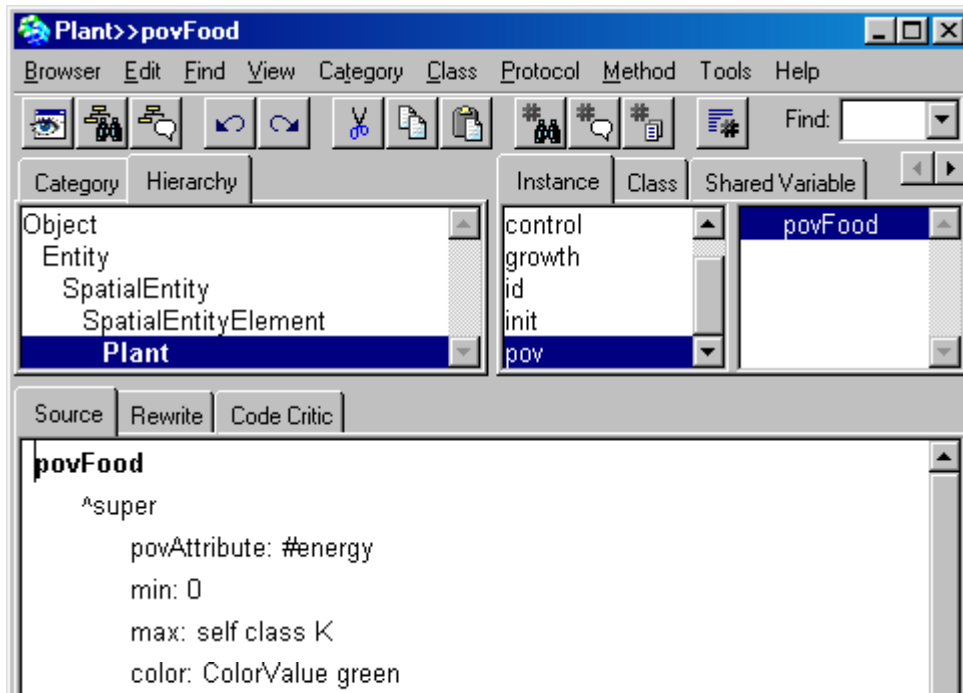
5.1. Assigning a green intensity according to the energy level of the plant

5.1.1. Add a “point of view” method named “povFood”



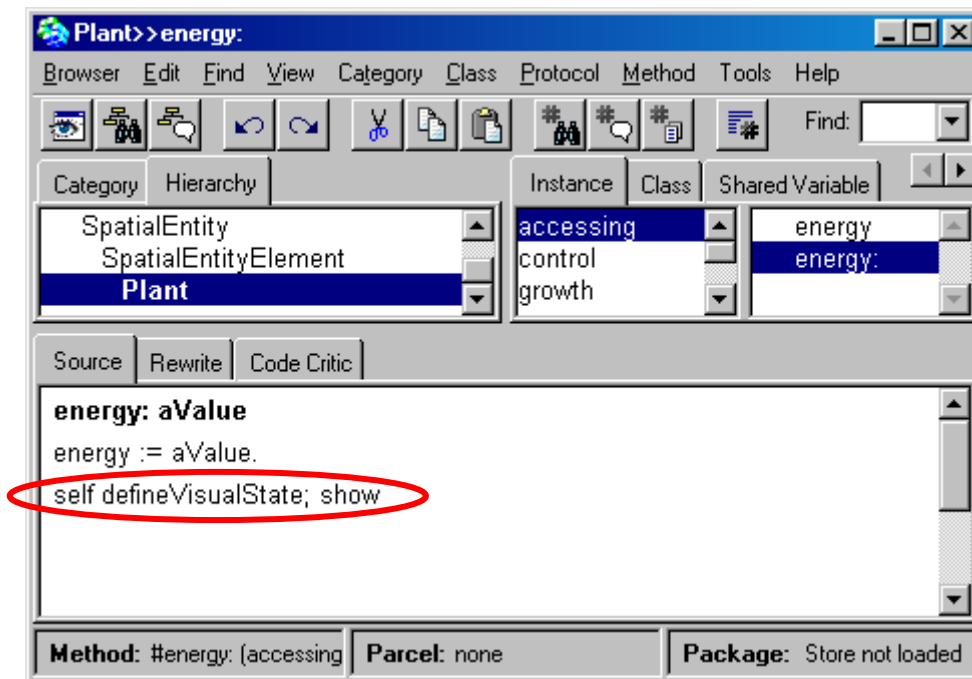
5.1.2. Write the code of the “povFood” method

A “pov” method has to return symbols, which needs to be associated to corresponding colours via the palette tool. This classical way of writing “pov” methods will be presented in section 5.2. Here, another way is used: “pov” methods may return directly colour values. Cormas provides a generic method that relates the brightness of a base colour to the value of a quantitative attribute.



5.1.3. Adjust the write-access method of all the attributes involved in “pov” methods

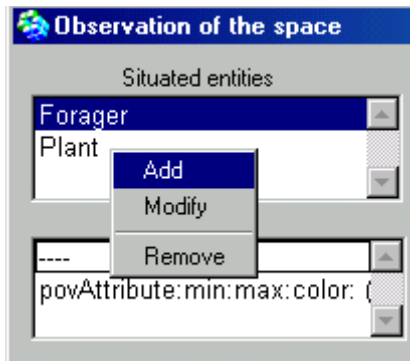
The only attribute involved in a “pov” method of the Plant entity is “energy”. To ensure that the visualization will be updated each time a new value of energy will be assigned to a plant, you need to add a specific instruction at the end of the corresponding write-access method :



5.2. Assigning a colour to distinguish foragers according to their strategy

5.2.1. Add a “pov” method named “povStrategy”

Click on the Forager object and ‘Add a new method’ in the list of methods, and then type the code below:



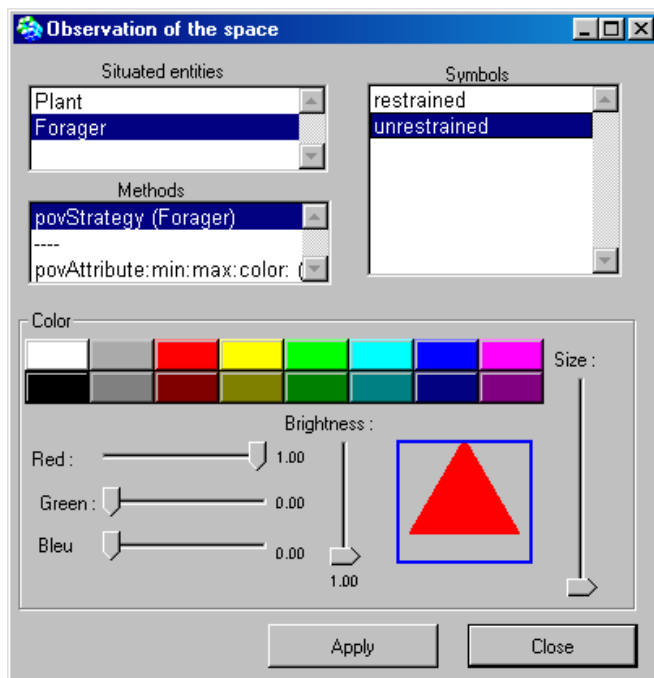
```
povStrategy
  ^self strategy
```

5.2.2. Add the symbols that may be returned by the “pov” method

You need to add, in the “Symbols” window, all the possible symbols that may be returned by any “pov” methods of the selected entity. Here, povStrategy will be the only “pov” method of the Forager entity. This method is returning the strategy of a forager agent. The possible values of a forager’s strategy are #restrained and #unrestrained. When you enter these values in the list of symbols, do not include the initial # character, and use exactly the same spelling (be careful, smalltalk is case-sensitive !).

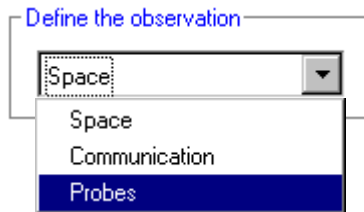
5.2.3. To each added symbol, associate a colour and a shape

Use the palette or the Red, Green and Blue sliders to choose a colour. Use the contextual menu within the blue squared to choose a polygonal shape. Do not forget to apply after each choice !

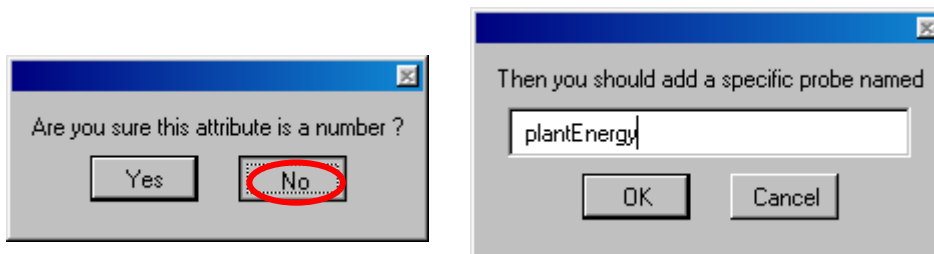
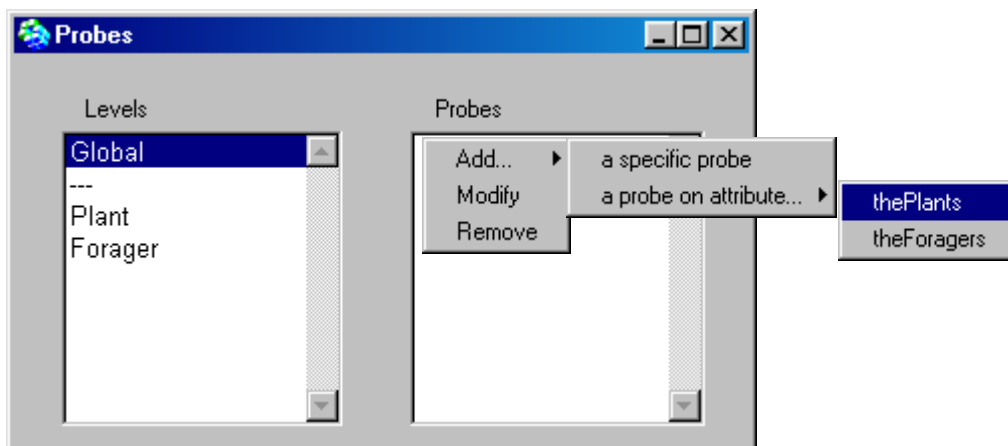


6. Designing “probes” to record the variations of markers

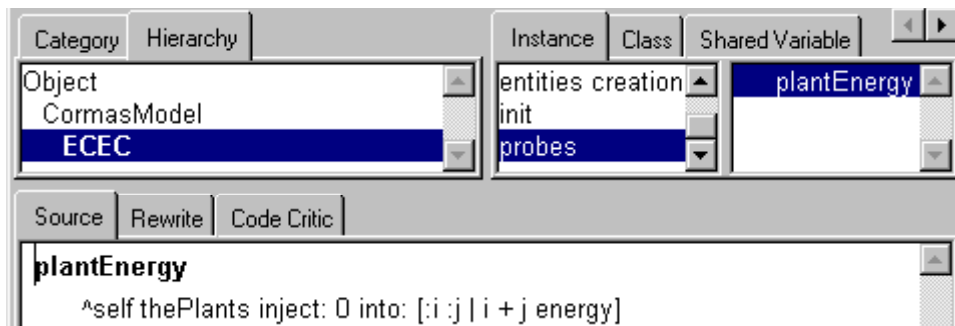
6.1. Open the “probes” definition window from the main Cormas interface



6.2. Add probes based on attributes of the model (global level)



6.3. Write the code for the first one: a cumulative value over a collection



6.4. Define 2 other probes : the number of restrained and unrestrained foragers

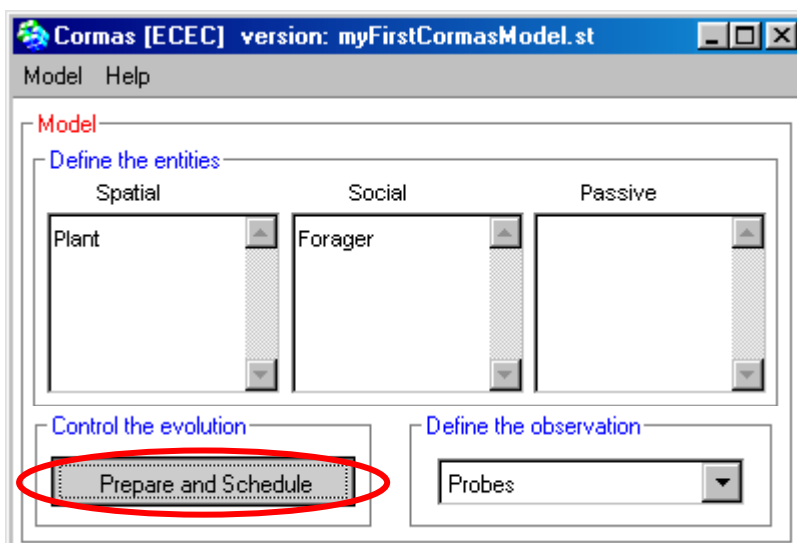
restrained

```
^(self theForagers select: [:f]
  f strategy = #restrained)) size
```

unrestrained

```
^(self theForagers select: [:f]
  f strategy = #unrestrained)) size
```

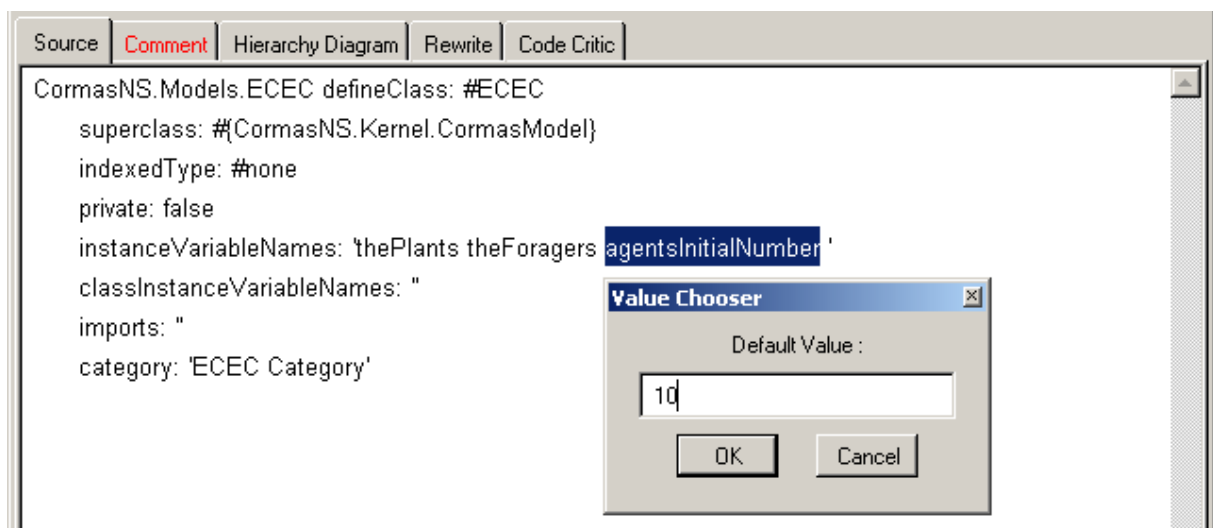
7. Designing control methods for the scheduling of simulation experiments



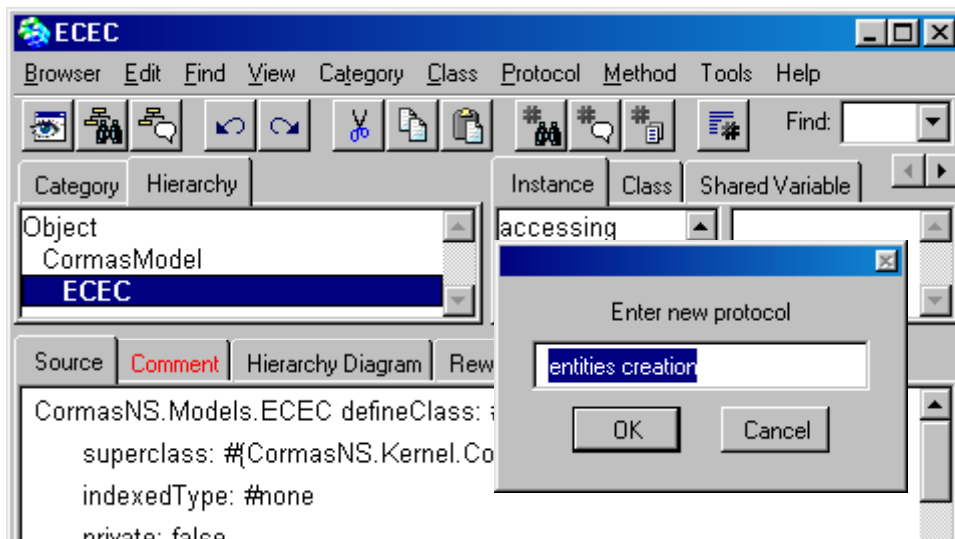
7.1. Create attribute agentsInitialNumber

Create the *agentsInitialNumber* attribute which allows user to choose the initial number of Foragers having each strategy.

Set the default number equals to 10.



7.2. Write, in a new protocol ("entities creation"), a method to create the forager agents



Write the `initForagers` methods :

initForagers

"initialize '`agentsInitialNumber`' foragers with restrained strategy and '`agentsInitialNumber`' foragers with unrestrained strategy"

```
super initAgents.  
self setRandomlyLocatedAloneAgents: Forager  
  n: self agentsInitialNumber  
  initMethod: #strategy:  
  arguments: ##restrained).  
self setRandomlyLocatedAloneAgents: Forager  
  n: self agentsInitialNumber  
  initMethod: #strategy:  
  arguments: ##unrestrained)
```

7.3. Write (in the "init" protocol) 3 different methods to be used as initial situations for simulation experiments

noForagers

```
self spaceModel loadEnvironmentFromFile: 'poor.env' withPov: #povFood.  
super initCells.  
super initAgents.
```

homogeneousEnv

```
self spaceModel loadEnvironmentFromFile: 'random.env' withPov: #povFood.  
super initCells.  
self initForagers.
```

fragmentedEnv

```
self spaceModel loadEnvironmentFromFile: 'fragmented.env' withPov: #povFood.  
super initCells.  
self initForagers.
```

7.4. Write (in the “control” protocol) a step method to be executed at each time-step of simulation

step: t

"Resource dynamics"

```
super stepEntities: self thePlants.
```

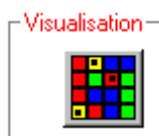
"Agents dynamics"

```
super stepDynPop: self theForagers.
```

8. Simulating the model

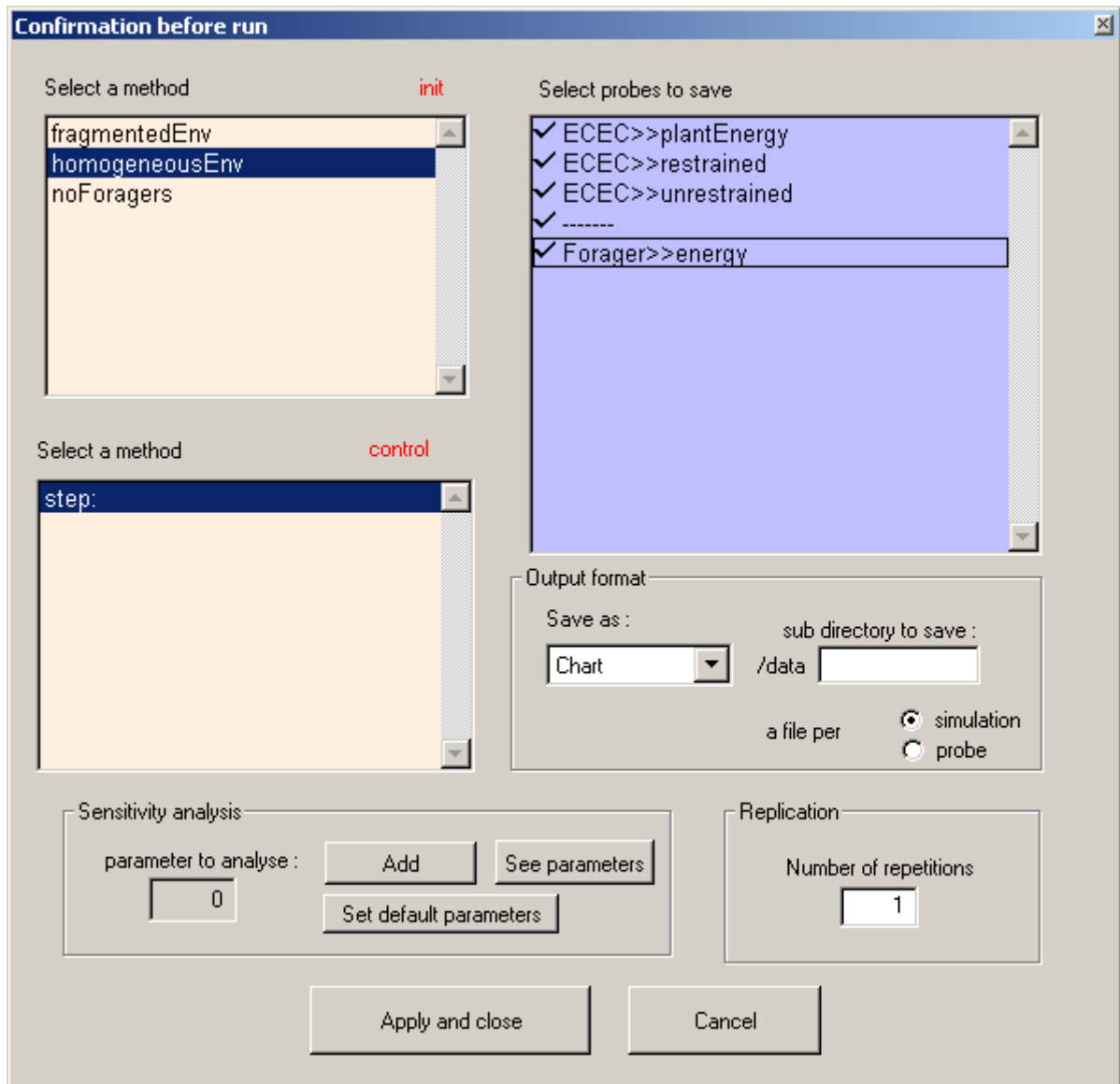
First of all you have to export your model. This creates a “MyFirstCormasModel” directory under “cormas/models”. Within this new models directory, create a sub-directory “maps” and put into that directory the files ‘poor.env’, ‘random.env’ and ‘fragmented.env’.

Then you have to open the spatial grid interface to be able to visualize the simulation. Click on :



Back to Cormas main interface, you can click on the “Initialize...” button. A window will open where you have to choose an initialisation method and a control method. You can also check the probes you would like to save³.

³ To select all the probes, click on the first one and shift click on the last one.

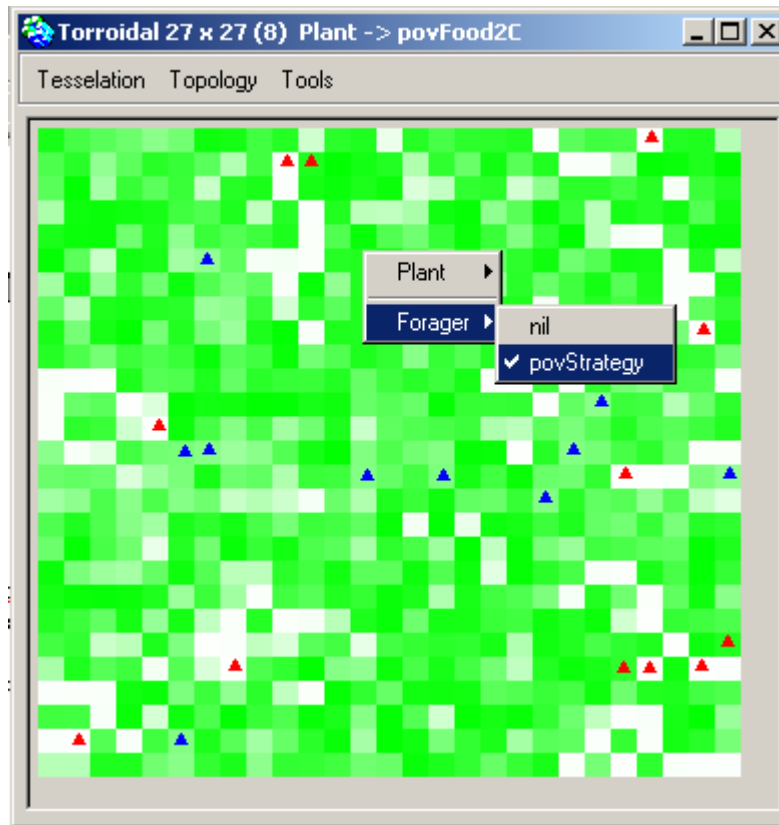


Then click on *Apply and close* button.

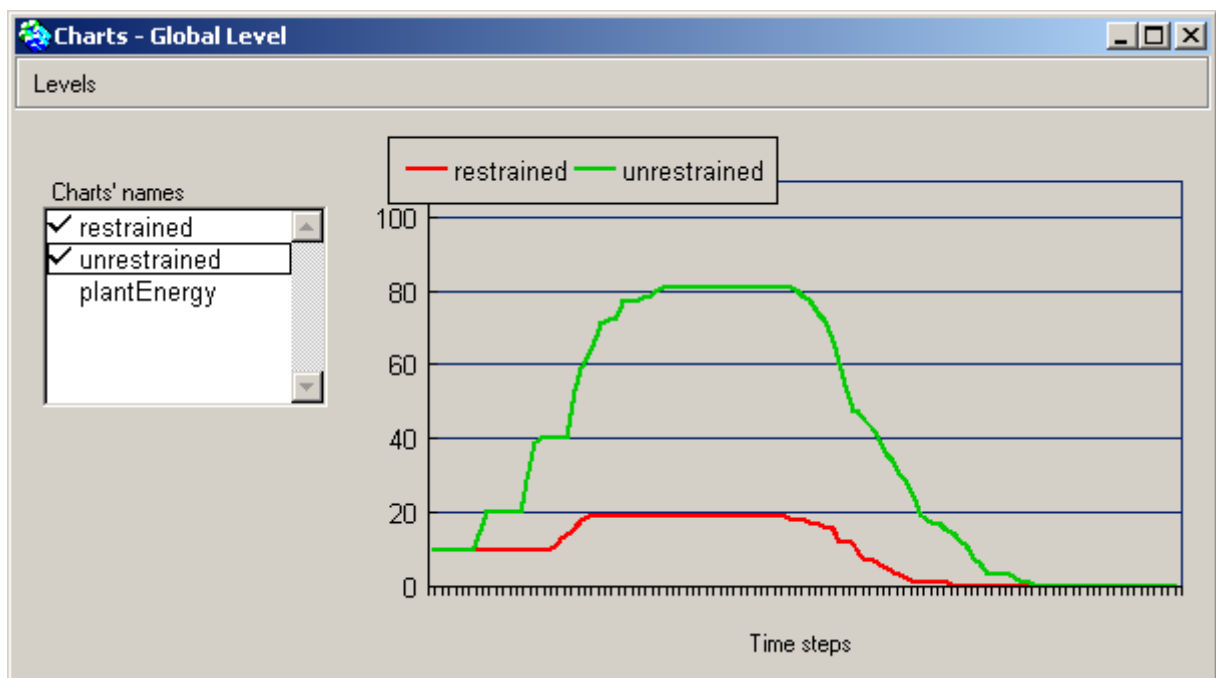
You can now simulate by clicking on the “Step” button or on the “Run” button once you have typed a duration.



In order to see also the foragers on the map, right click on the grid and select the Forager → povStrategy.



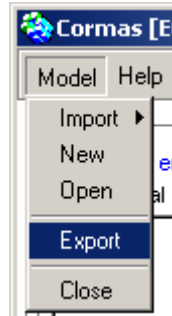
In order to see the resulting curves of the simulation, click on the *Charts* button. Then click on the probe you want to see. If you want to see several probes at once, select them with Ctrl key.



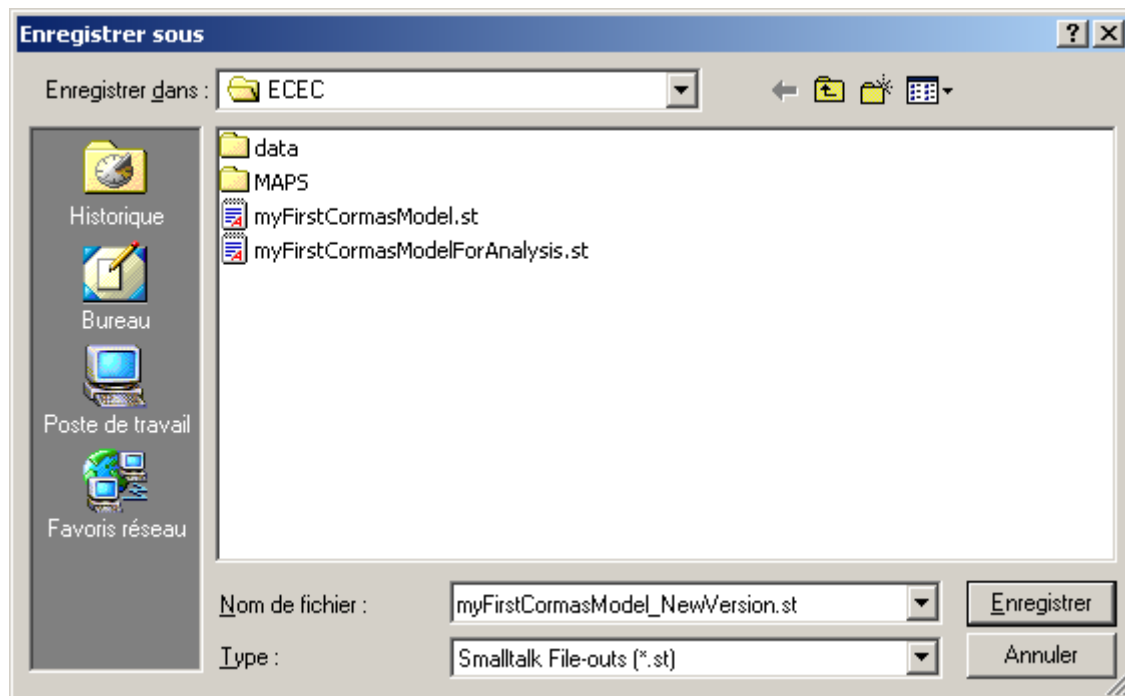
9. Saving, loading and versioning the model

9.1. Saving and versioning

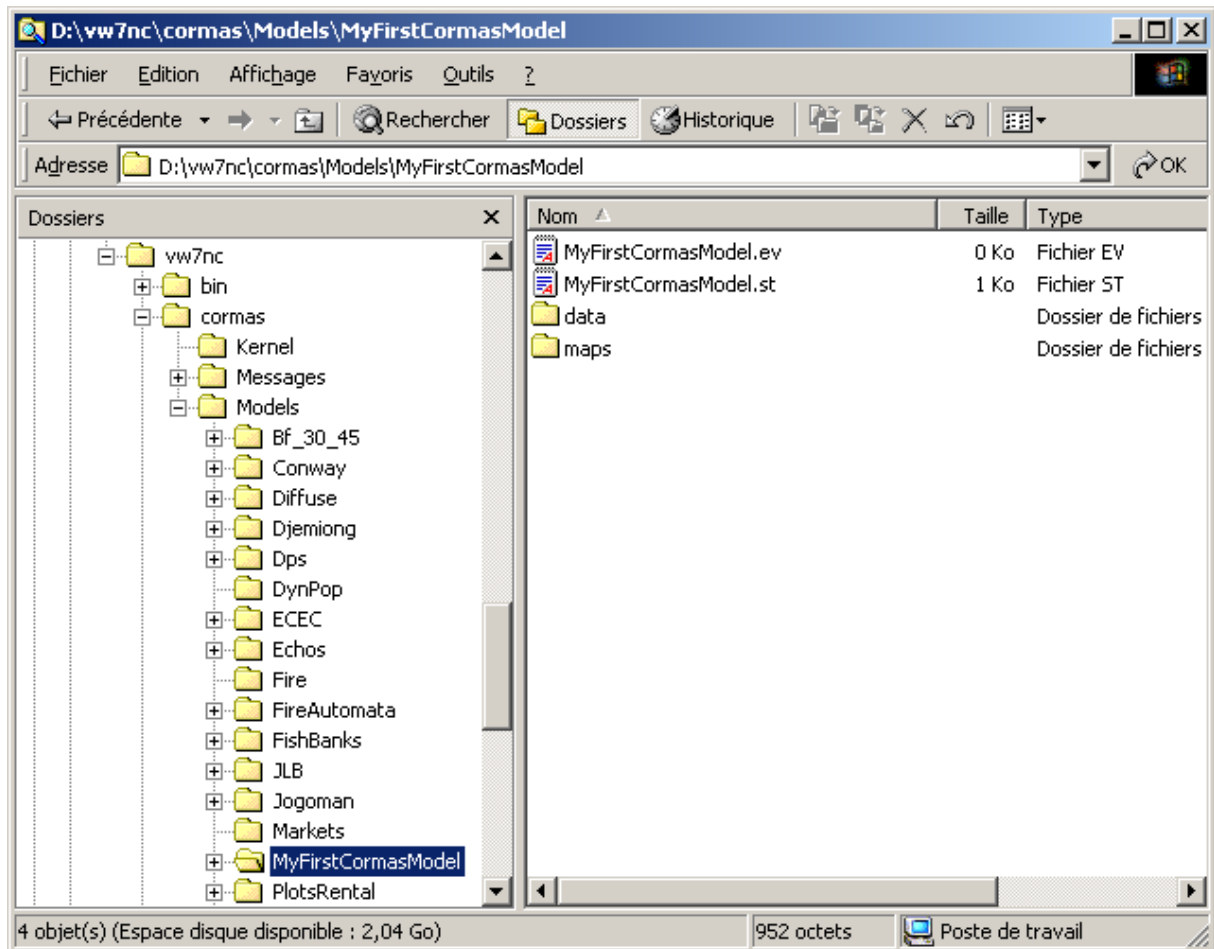
When you want to save your model, click on Model → Export



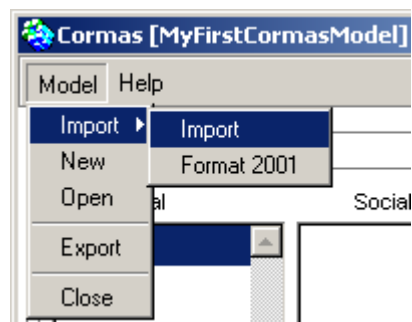
You can then choose the version of your model, for example "myFirstModel_newVersion.st" :



The model is then saved on the disk as a file with .ST extension (ST for SmallTalk). Cormas has created a directory for your model. You should get the following architecture :



9.2. Loading the model



Then choose the version you want to load.

10. Analysing the model

We present now the ways to conduct *sensitivity analysis* with Cormas. Analysing is an important part of the modelling activity and should not be forgotten.

Cormas doesn't propose analysis tools; it just allows to launch many simulations according to parameters values and to export the data of the simulations in different kinds of files :

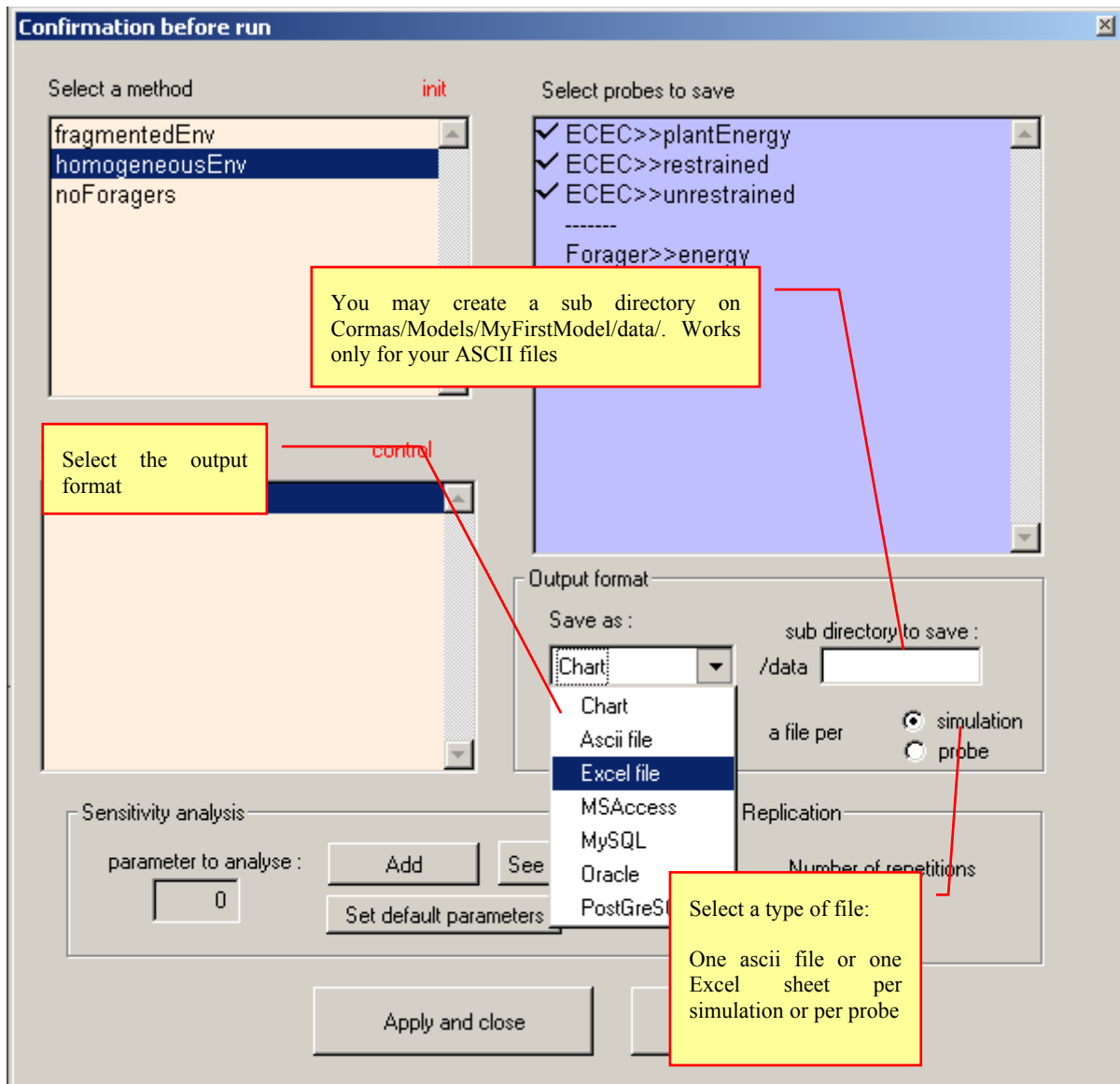
ASCII files,

Excel files or

into data bases.

You can then analyse your data with your favourite software...

Select the output format of the files :

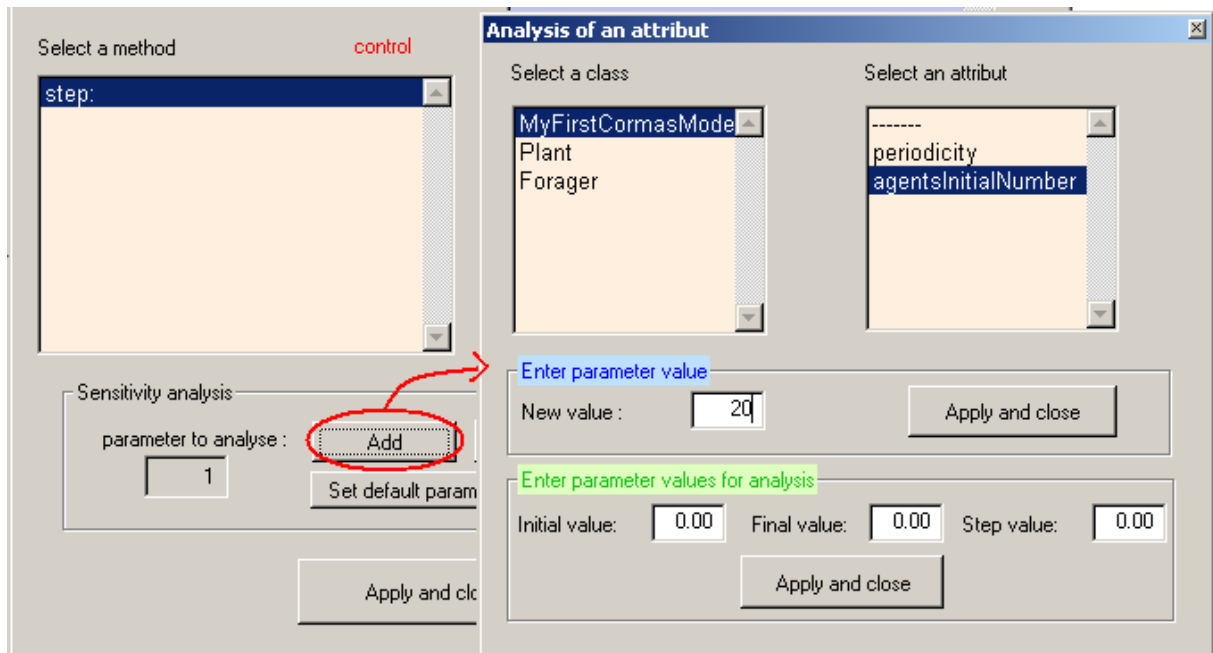


To get more information about the output format, see the Output format section below (page 27).

10.1. Choose the parameter's values

Click on the *Add* button on the *Sensitivity analysis* panel.

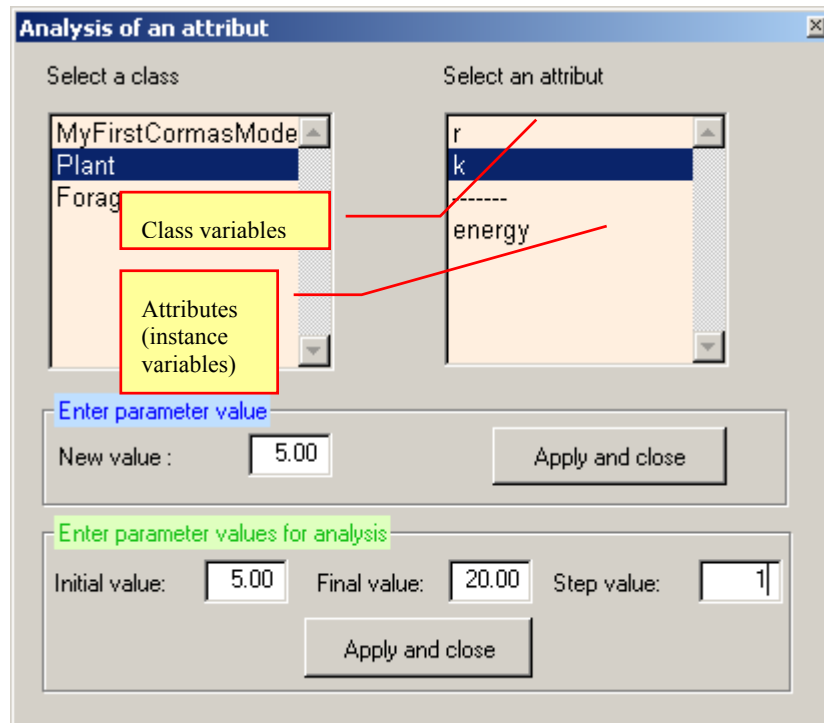
10.1.1. Choose one value



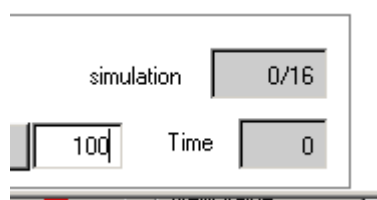
On the "Analysis of an attribute" window, select the class (ex. MyFirstCormasModel) of the parameter you want to choose and then, select the attribute (ex: agentsInitialNumber). You can now enter a new value (ex: 20), then *Apply and close*.

10.1.2. Choose a set of values

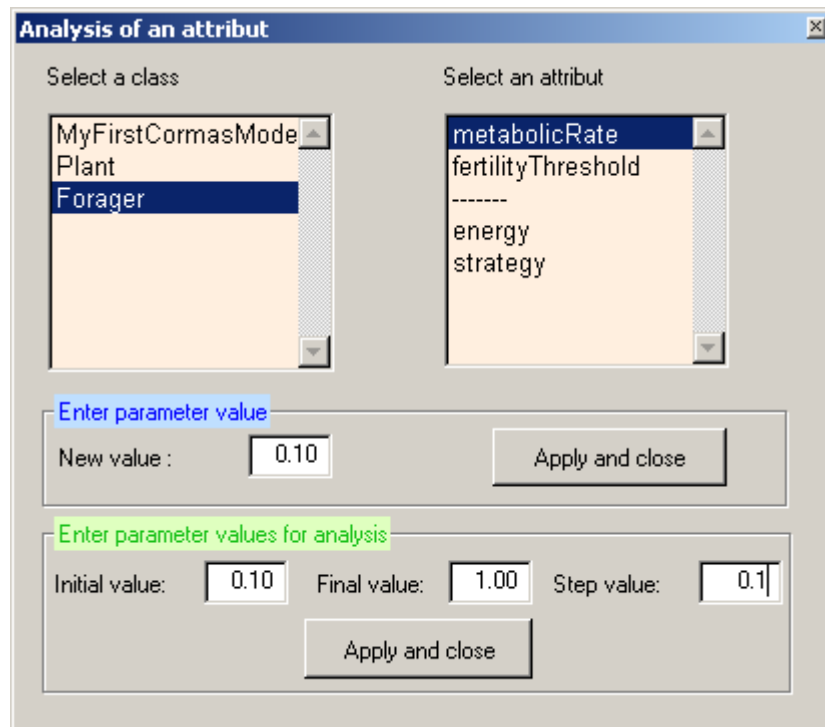
The figure below shows how to analyse the model according to different values of K, the class variable of Plant (from 5 to 20 by step 1) :



If you accept now, you can see that you can run 16 simulations (5 to 20 by step 1 = 16 simulations).



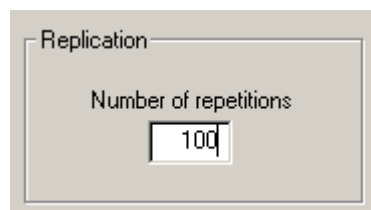
If you want to add a new parameter to analyse to this 16 simulations, click again on *Initialize* button, then *Add* button then choose another set of values. Example: metabolicRate, from 0,1 to 1 by step of 0,1.



The number of simulations is now equal to 160 : 16 (for the first set of parameter) x 10 (for the second set of parameter).

The previous parameters have been kept in memory.

If your model contains randomness, you can choose to repeat the same simulation for each set of parameters. Click again on *Initialize* button, then choose a value for the *Number of repetitions* :

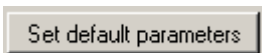


Now, the number of simulations is 16000 !

To see the parameter values you entered, click on **See parameters** button.

	Class	Attribut	from	to	step	current
1	Forager	metabolicRate	0.1	1	0.1	0.1
2	Plant	k	5	20	1	5
3	MyFirstCormasModel	agentsInitialNum	20	20	1	20

To forget the parameter's sets, click on the



button. This action reset the default values of the parameters.

10.2. Output format

By default, the probes are "saved" as charts. Be careful : this option shows only the last simulation data.

10.2.1. Ascii or Excel format

If you choose to save the probe's data as Ascii files or Excel sheets, Cormas will only save the global probes.

If you want to save also the instance's information, you should use data base format (see below).

With these format, you can choose to save a file (or an Excel sheet) per simulation or per probe.

If you select a file per probe, you will obtain 3 files as following :

Nom	Taille
MyFirstCormasMo...	25 Ko
plantEnergy.txt	2 Ko
restrained.txt	1 Ko
unrestrained.txt	1 Ko

or 3 Excel sheets :

	A	B	C	D	E	F	G
1	Time\MyFirst	plantEnergy:	plantEnergy:	plantEnergy:	plantEnergy:	plantEnergy:	plantEnergy:
2	0	3697,22583	3697,22583	3697,22583	3697,22583	3697,22583	3697,22583
3	1	3236,53174	3384,43213	3483,35718	3570,58618	3626,0249	3685,40454
4	2	2961,01514	3183,3374	3339,93848	3477,4248	3586,24878	3677,23755
5	3	2775,94946	3030,979	3232,74023	3406,03369	3554,22021	3670,80688
6	4	2630,74976	2907,99731	3141,69043	3340,30273	3521,11035	3670,87354

The first cell of each column contains the probe name, the simulation number, and the name of the parameters and their initial value for this simulation.

If you select a file per simulation, you will obtain an Ascii file per simulation (16000 files !) or an Excel sheet per simulation :

	A	B	C	D	E	F	G	H	I	J
1	MyFirstCorm	plantEnergy	restrained	unrestrained						
2	0	3697,22583	20	20						
3	1	3233,14697	20	20						
4	2	2958,88989	20	20						
5	3	2770,78564	20	20						

The first Excel cell contains information about the simulation :

The name of the model, the current repeat number and the parameter's current value.

10.2.2. Data base format

10.2.3. Launching the ODBC connection

If you select a data base as output format, you are invited to complete a form :

LOGIN to Database:

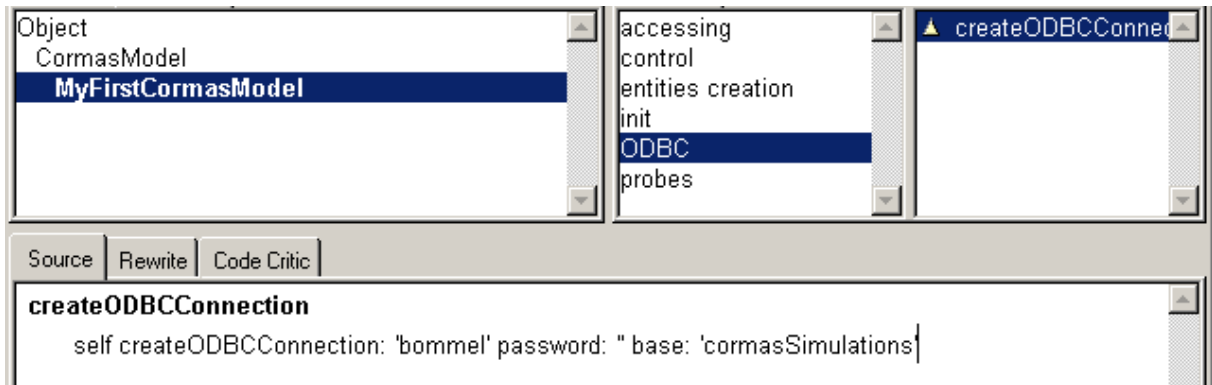
User Name:

Password:

OK Cancel

To set up the ODBC bridge under Windows, please see the end notes (Appendices p.33).

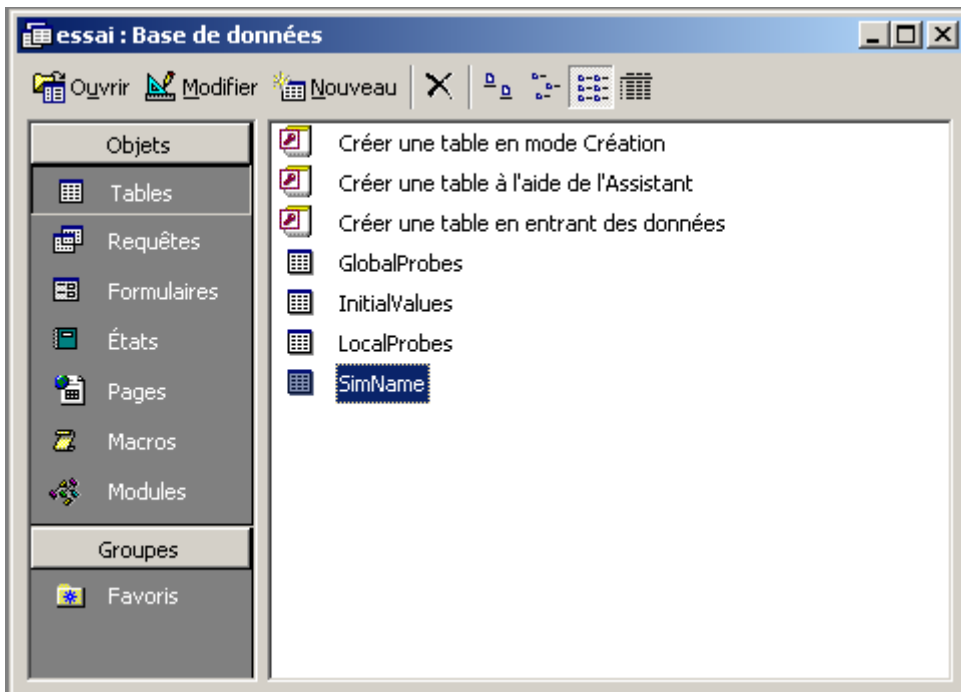
If you don't want to have this form for each analysis session, add a method in your model (you can create a new protocol "ODBC") :



Of course, you should define it with the right data base (<String>) and your username (<String>) and your password (<String>).

10.2.4. Data base description

The 4 tables of the data base (Access) :



The main table is called "SimName"; it contains information about the simulations :

SimName : Table	
Nom du champ	Type de données
num	NuméroAuto
modelName	Texte
version	Texte
analysis	Numérique
simulationName	Texte
simNumber	Numérique
day	Date/Heure
timeHMS	Date/Heure

- simulationName: a non ambiguous name with the simulation number, plus the date and the time (ex: 6_8/19/02_7:30:55 am),

- modelName: example 'MyFirstCormasModel'
- version: the version of the model, ie. the ST file (ex: myFirstCormasModelForAnalysis.st)
- analysis: the number of the current analysis session
- simNumber: the number of the current simulation of this analysis session
- day
- time

Example :

SimName : Table								
	num	modelName	version	analysis	simulationName	simNumber	day	timeHMS
+	4	Test2	TestWithSimManager2.st	1	11/18/02_5:35:45 am	1	18/11/2002	05:35:45
+	5	Test2	TestWithSimManager2.st	2	11/18/02_5:44:37 am	1	18/11/2002	05:44:37
+	6	FireAutomata	FireAutomataPourANALYSE.st	3	12/3/02_4:17:05 am	1	12/03/2002	04:17:05
+	7	MyFirstCormasModel	myFirstCormasModelForAnalysis.st	4	18/08/02_3:35:03 am	1	19/08/2002	03:35:03
+	8	MyFirstCormasModel	myFirstCormasModelForAnalysis.st	4	28/08/02_3:57:36 am	2	19/08/2002	03:57:36
+	9	MyFirstCormasModel	myFirstCormasModelForAnalysis.st	4	38/08/02_3:57:41 am	3	19/08/2002	03:57:41
+	10	MyFirstCormasModel	myFirstCormasModelForAnalysis.st	4	48/08/02_3:57:47 am	4	19/08/2002	03:57:47
+	11	MyFirstCormasModel	myFirstCormasModelForAnalysis.st	4	58/08/02_3:57:52 am	5	19/08/2002	03:57:52
+	12	MyFirstCormasModel	myFirstCormasModelForAnalysis.st	4	68/08/02_3:57:57 am	6	19/08/2002	03:57:57
+	13	MyFirstCormasModel	myFirstCormasModelForAnalysis.st	4	78/08/02_3:58:03 am	7	19/08/2002	03:58:03
+	14	MyFirstCormasModel	myFirstCormasModelForAnalysis.st	4	88/08/02_3:58:09 am	8	19/08/2002	03:58:09
+	15	MyFirstCormasModel	myFirstCormasModelForAnalysis.st	4	98/08/02_3:58:15 am	9	19/08/2002	03:58:15
+	16	MyFirstCormasModel	myFirstCormasModelForAnalysis.st	4	108/08/02_3:58:20 am	10	19/08/2002	03:58:20

The "InitialValues" table contains 5 fields :

InitialValues : Table		
	Nom du champ	Type de données
🔑	num	NuméroAuto
▶	simulationName	Texte
	className	Texte
	parameterName	Texte
	initialValue	Texte

- num: a unique value of the table
- simulationName: the foreign key associated to the primary key of SimName table
- className: example 'MyFirstCormasModel' or 'Plant class' for the class variable value
- parameterName: the name of the attribute
- initialValue: the value of the parameter for the current

321	18_8/19/02_1:20:05 pm	Plant class	r	0.2
322	18_8/19/02_1:20:05 pm	Plant class	k	6
323	18_8/19/02_1:20:05 pm	MyFirstCormasModel	agentsInitialNumber	20
324	18_8/19/02_1:20:05 pm	Forager class	metabolicRate	0.2
325	18_8/19/02_1:20:05 pm	Forager class	fertilityThreshold	100

The "GlobalProbes" table contains 5 fields :

GlobalProbes : Table		
	Nom du champ	Type de données
🔑	num	NuméroAuto
	simulationName	Texte
	step	Numérique
	probeName	Texte
	probeValue	Numérique

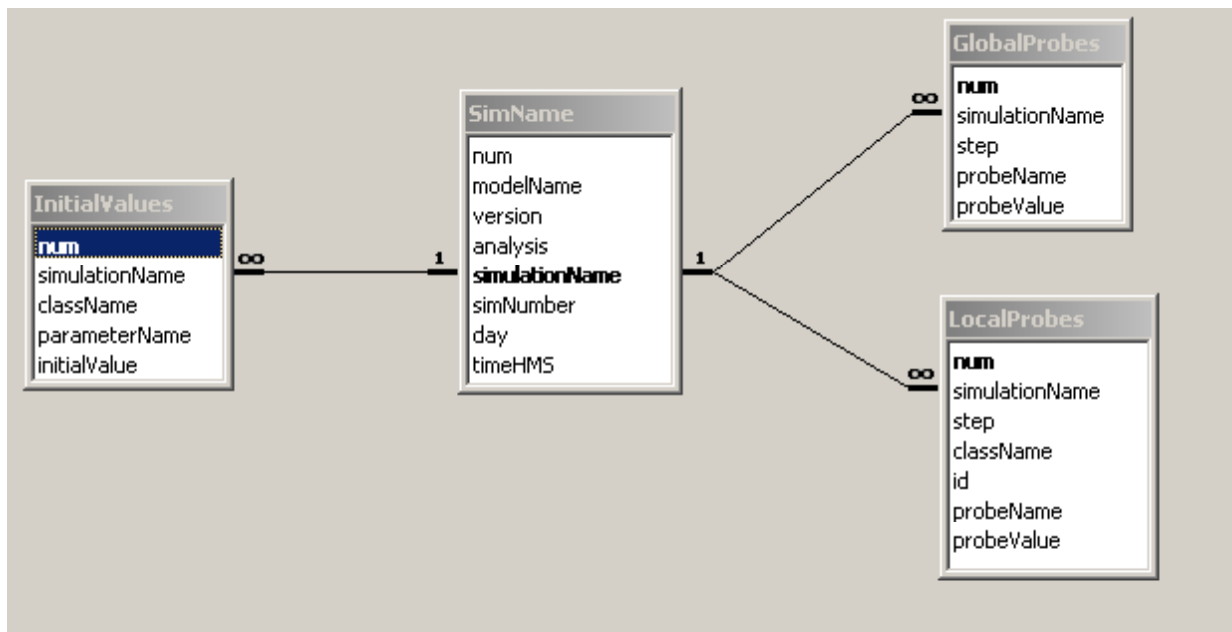
GlobalProbes : Table						
	num	simulationName	step	probeName	probeValue	
▶	830	1_8/19/02_1:17	0	plantEnergy	3697,23	
	831	1_8/19/02_1:17	0	restrained	20	
	832	1_8/19/02_1:17	0	unrestrained	20	
	833	1_8/19/02_1:17	1	plantEnergy	3235,62	
	834	1_8/19/02_1:17	1	restrained	20	
	835	1_8/19/02_1:17	1	unrestrained	20	

The "LocalProbes" table contains 7 fields :

LocalProbes : Table		
	Nom du champ	Type de données
🔑	num	NuméroAuto
	simulationName	Texte
	step	Numérique
	className	Texte
	id	Numérique
	probeName	Texte
	probeValue	Numérique

LocalProbes : Table							
	num	simulationName	step	className	id	probeName	probeValue
	15731	1_8/19/02_1:17	0	Forager	1	energy	50
	15732	1_8/19/02_1:17	0	Forager	2	energy	50
	15733	1_8/19/02_1:17	0	Forager	3	energy	50
	15734	1_8/19/02_1:17	0	Forager	4	energy	50
	15735	1_8/19/02_1:17	0	Forager	5	energy	50

The graph below shows the relations (foreign keys and primary keys) between the 4 tables and their field names.



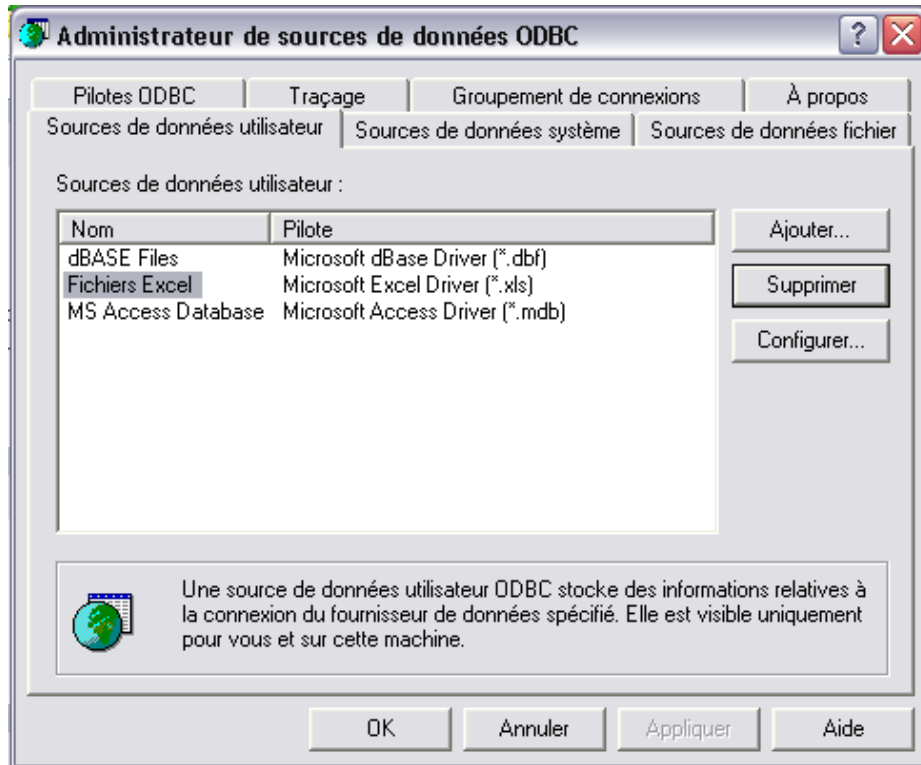
11. Appendices

11.1. How to connect VisualWorks to Access

11.1.1. Set up the ODBC driver ODBC

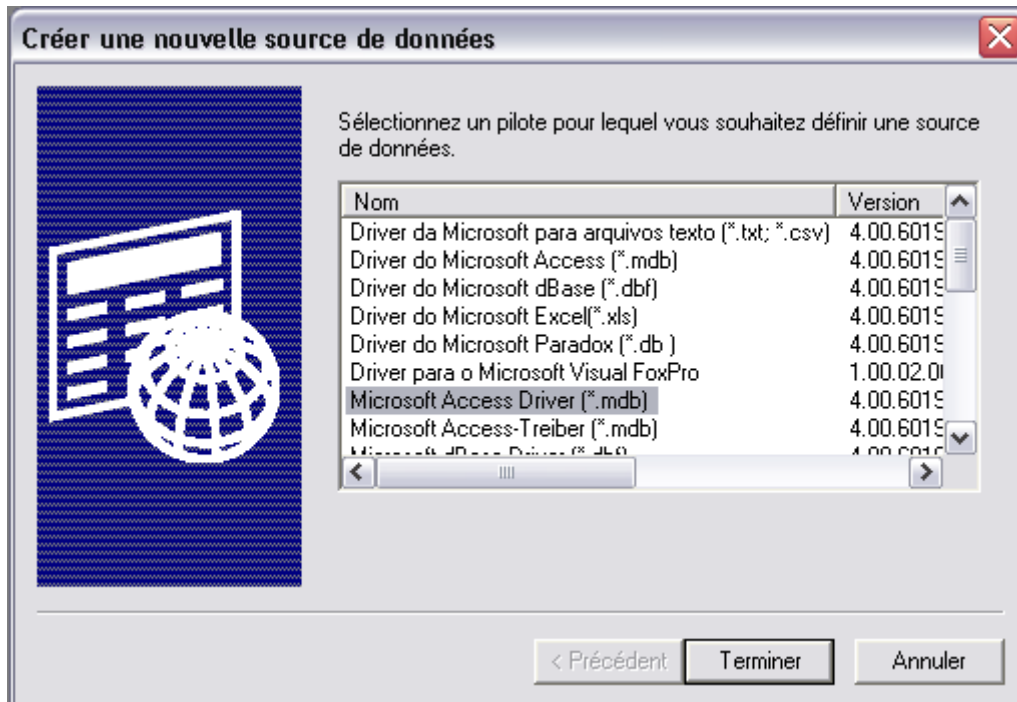
First of all, create an ACCESS file in the directory VW7\cormas\DataBase. Call it *cormasSimulations.mdb*.

Launch now the « Data Sources (ODBC) » tool, located in the configuration pannel → Administration tools (or directly launch file "%system32\odbcad32.exe").



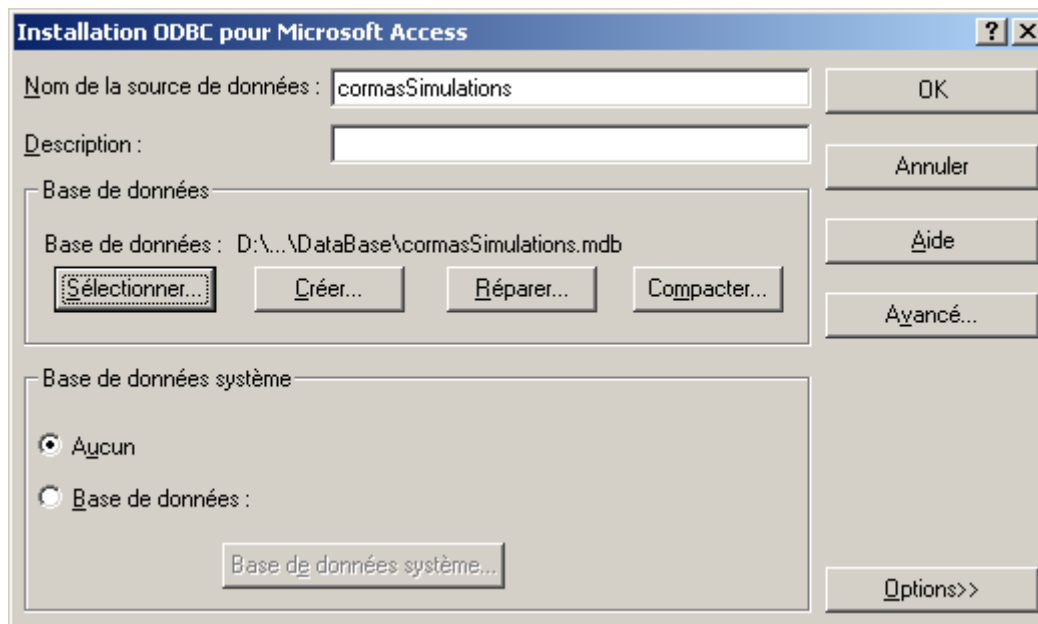
Create now a link for your database with a logical name : Click on « Add » button.

Select the « Microsoft Access Driver (*.mdb) driver, as shown below :



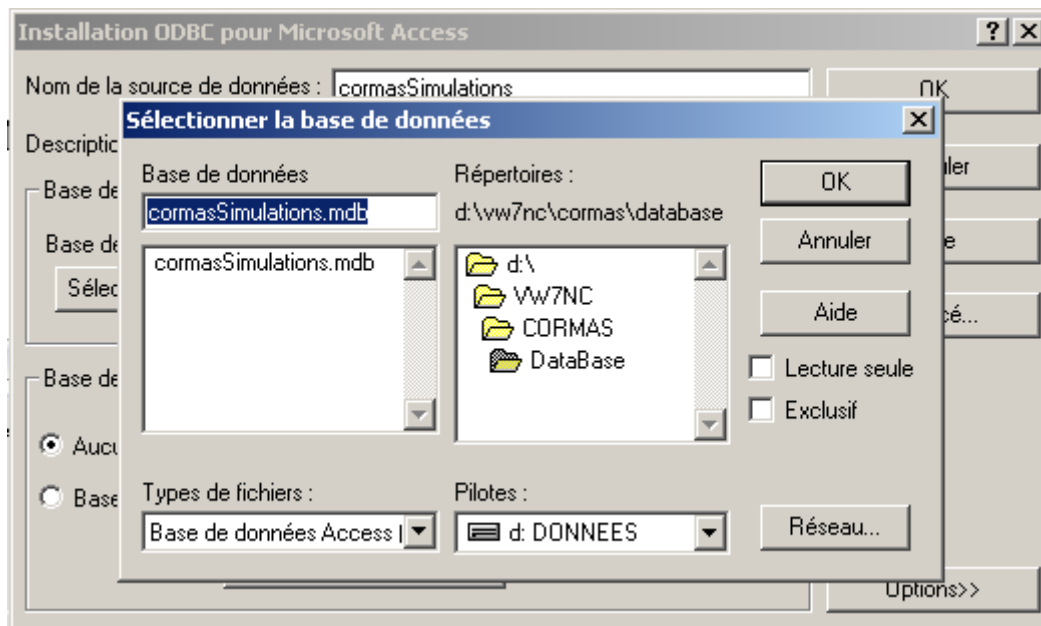
Click on « Finish » .

A new window pops up :

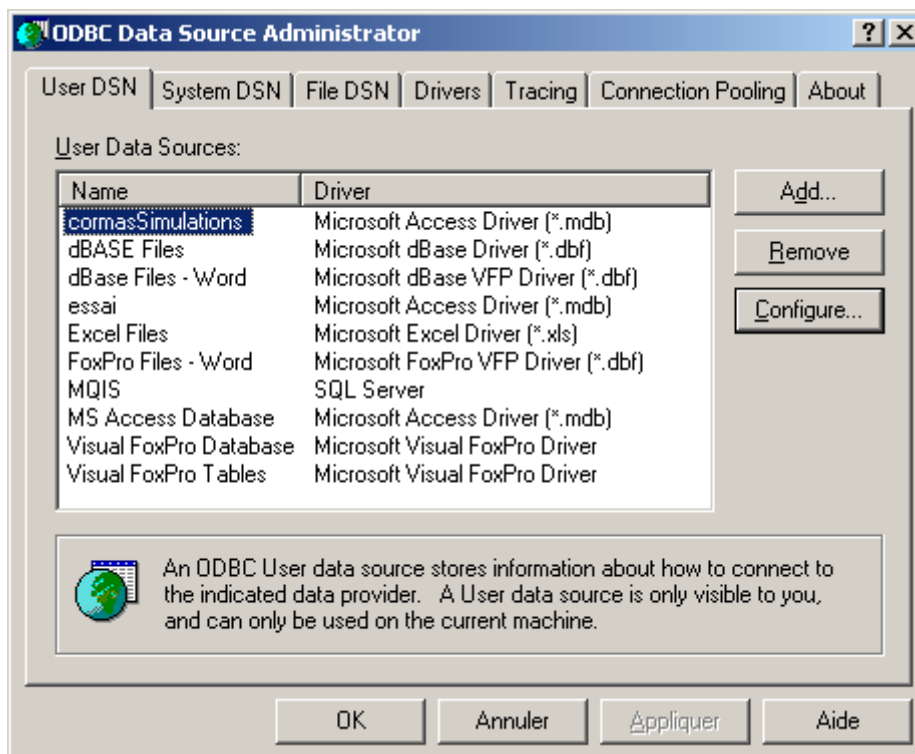


In the field « Data source name », write the logical name of the link. This name will be used in VisualWorks to identify the database.

Click on « Select... » button.

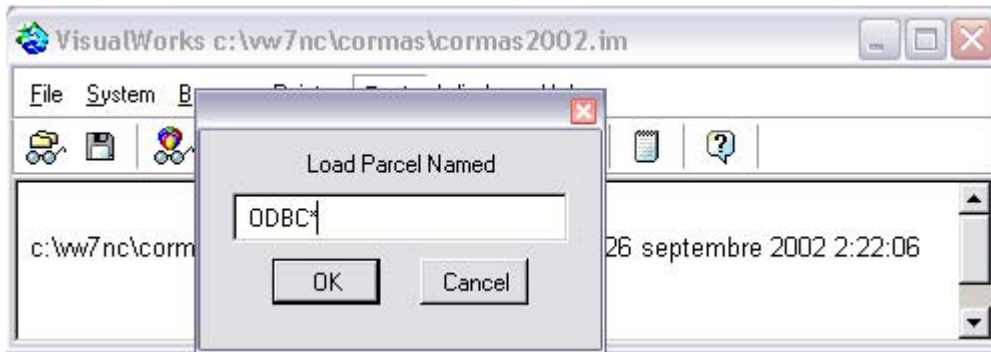


Select the Access file called « cormasSimulations.mdb ».

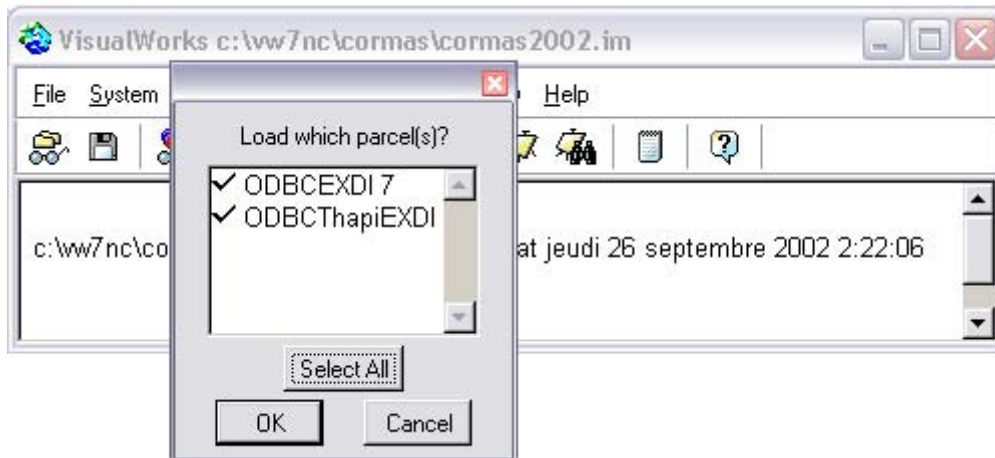


Click on « OK ». That's it !

To use this connection, you must verify that the right parcels are loaded in VisualWorks. Tools → Load Parcel Named... and write « ODBC* ».



If « ODBCEXDI 7 » and « ODBCThapiEXDI » parcels are not loaded, select them and click OK :



Don't forget to save VisualWorks.

11.2. UML : Class diagram of Cormas - Entities category

