

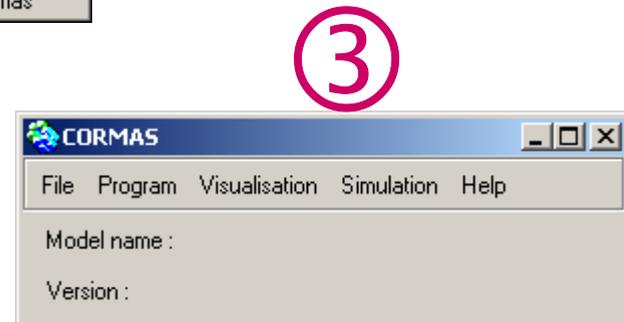
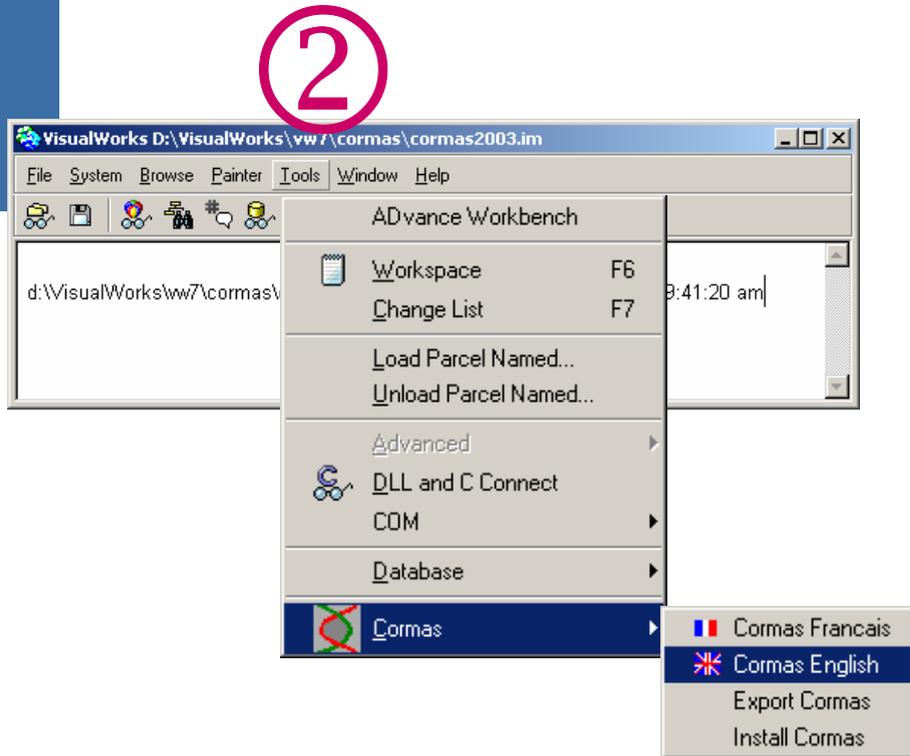
**Introduction à Cormas**

**Automate**  
**Référendum**

# Description

- L'objectif du modèle est de simuler un système de diffusion des opinions par effet de voisinage lors d'un référendum.
- Le territoire est découpé en portions régulières représentant des foyers. Chaque foyer exprime une intention de vote.
- Initialement, chaque foyer détermine au hasard son intention de vote parmi les 4 possibilités:  
abstention, blanc, oui, non
- A chaque tour, un foyer se rallie à l'opinion majoritaire dans son voisinage immédiat

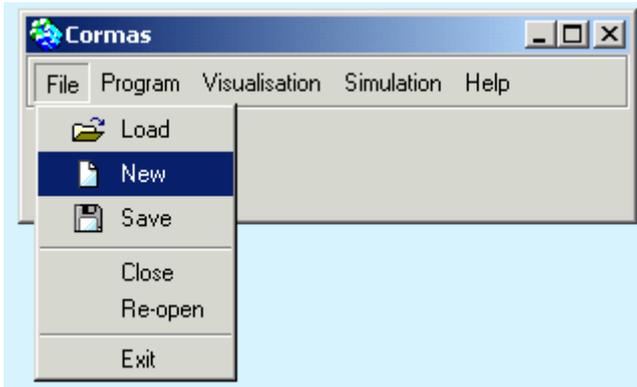
# Lancer Cormas



# Créer un nouveau modèle

Dans le menu “Fichier”, sélectionner “new” et taper le nom “Referendum” comme nom de nouveau modèle

①



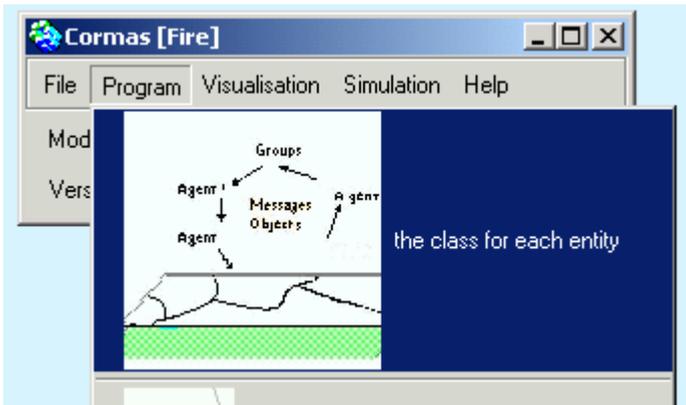
File → New

②



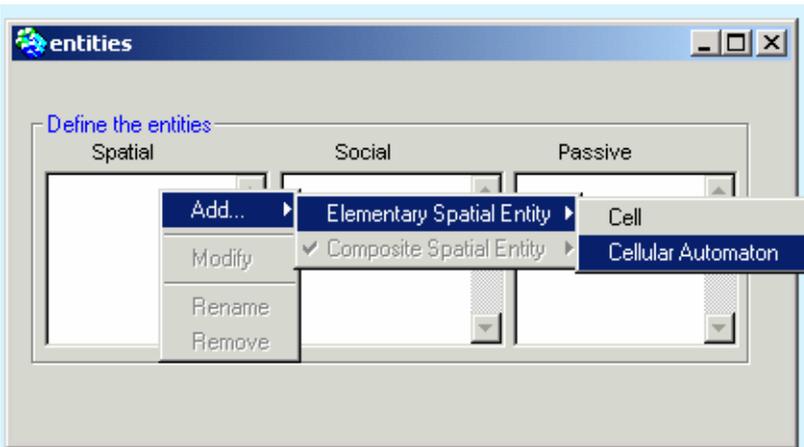
# Créer une entité spatiale

1



Programmer → les classes pour les entités

2



Clic-droit dans la zone “**Spatiales**”

Choisir **Entité Spatiale Elementaire**

Puis **Cellule Automate**

Taper le nom: **SpacePortion**

**Cellule** ≠ **Cellule Automate**

**Cellule** : pas d'attributs **state** ni **bufferState**

**Cellule Automate** : définit **state** et **bufferState**

# Fenêtre de définition des classes: le “Refactoring Browser”

The screenshot shows the Refactoring Browser window in SpacePortion. The window title is "SpacePortion". The menu bar includes "Browser", "Edit", "Find", "View", "Category", "Class", "Protocol", "Method", "Tools", and "Help". The toolbar contains various icons for navigation and editing. The "Category" tab is selected, and the "Hierarchy" sub-tab is active. The class hierarchy is displayed in a list on the left, with "SpacePortion" selected. The right pane shows the class definition for "SpacePortion", which is a subclass of "SpatialEntityCell". The definition includes the following attributes:

```
CormasNS.Models.ForestFire defineClass: #SpacePortion
  superclass: #{CormasNS.Kernel.SpatialEntityCell}
  indexedType: #none
  private: false
  instanceVariableNames: "
  classInstanceVariableNames: "
  imports: "
  category: 'ForestFireCategory'
```

The "instanceVariableNames: " line is circled in red, and a red arrow points to it with the label "Attributes".

At the bottom of the window, the following information is displayed:

- Class: CormasNS.Models.ForestFire.Spar
- Parcel: none
- Package: Store not loaded

# Vue hiérarchique dans le Refactoring Browser

Super classes

Category: **Hierarchy**

- Object
- Entity
  - SpatialEntity
  - SpatialEntityElement
  - SpatialEntityCell**
  - SpacePortion

Instance | Class | Shared Variable | Instance Variable

accessing  
control  
init

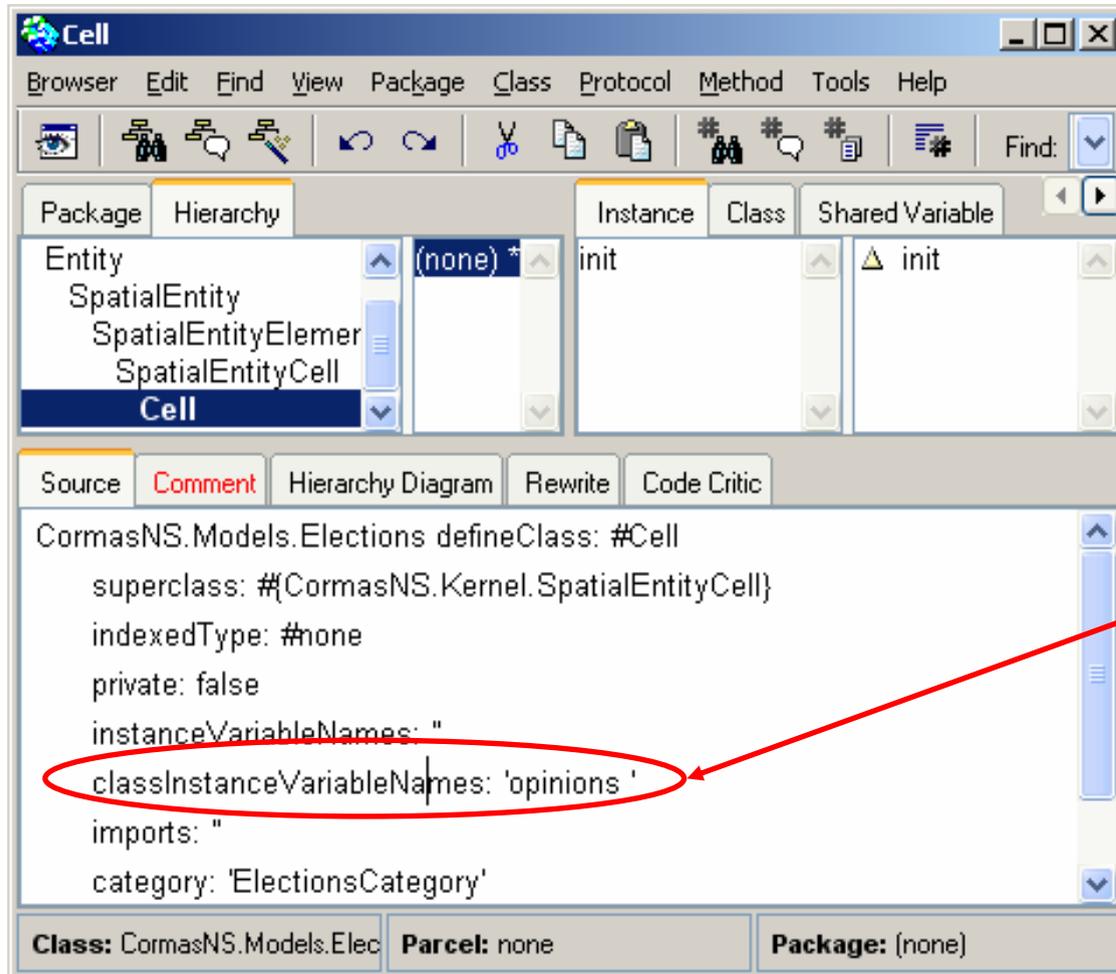
Source | Comment | Hierarchy Diagram | Rewrite | Code Critic

```
CormasNS.Kernel defineClass: #SpatialEntityCell
  superclass: #{CormasNS.Kernel.SpatialEntityElement}
  indexedType: #none
  private: false
  instanceVariableNames: 'state bufferState'
  classInstanceVariableNames:
  imports: "
  category: 'CormasKernel-Entities'
```

Class: CormasNS.Kernel.SpatialEntityCell | Parcel: none | Package: Store not loaded

Attributs hérités

# Définir une variable de classe spécifique à “SpacePortion”



The screenshot shows the Cell IDE interface. The top menu bar includes Browser, Edit, Find, View, Package, Class, Protocol, Method, Tools, and Help. Below the menu is a toolbar with various icons. The main window is divided into several panes. On the left, the Package pane shows a hierarchy of classes: Entity, SpatialEntity, SpatialEntityElement, SpatialEntityCell, and Cell (selected). The middle pane shows the Instance and Class tabs, with 'init' listed under both. The right pane shows the Shared Variable tab, which is currently empty. Below these panes are tabs for Source, Comment, Hierarchy Diagram, Rewrite, and Code Critic. The Source tab is active, displaying the following code:

```
CormasNS.Models.Elections defineClass: #Cell
  superclass: #(CormasNS.Kernel.SpatialEntityCell)
  indexedType: #none
  private: false
  instanceVariableNames: "
  classInstanceVariableNames: 'opinions '
  imports: "
  category: 'ElectionsCategory'
```

The text `classInstanceVariableNames: 'opinions '` is circled in red. A red arrow points from the text "Variable de classe" to this circled text.

**Class:** CormasNS.Models.Elec **Parcel:** none **Package:** (none)

Variable de classe

# Entrer une valeur par défaut pour la variable de classe "opinions"

The screenshot shows a software development environment with a class browser on the left and a code editor in the center. The class browser shows a hierarchy of classes: Entity, SpatialEntity, SpatialEntityElement, SpatialEntityCell, and Cell. The code editor displays the definition of the #Cell class, including its superclass, indexed type, and instance variable names. The instance variable names are listed as "opinions". A context menu is open over the code editor, with the "Create CORMAS Accessors" option selected. A "Default value chooser" dialog box is also visible, showing the default value for the "opinions" attribute as "#(#oui #non #blanc #abstention)".

**Cell class**

Browser Edit Find View Package Class Protocol Method Tool

Package Hierarchy Instance Class SH

Entity  
SpatialEntity  
SpatialEntityElement  
SpatialEntityCell  
**Cell**

Source Comment Hierarchy Diagram Rewrite Code Critic

```
ComasNS.Models.Elections defineClass: #Cell
  superclass: #{ComasNS.Kernel.SpatialEntityCell}
  indexedType: #none
  private: false
  instanceVariableNames: "
  classInstanceVariableNames: opinions "
  imports: "
  category: 'ElectionsCategory'
```

**Class:** ComasNS.Models.Elec **Parcel:** none **Package:** (none)

**Context Menu:**

- Find
- Replace
- Undo
- Copy
- Cut
- Paste
- Do It Ctrl+D
- Print It Ctrl+P
- Inspect It Ctrl+I
- Debug It
- Accept Ctrl+S
- Cancel
- Instance Variables
- Create CORMAS Accessors

**Default value chooser**

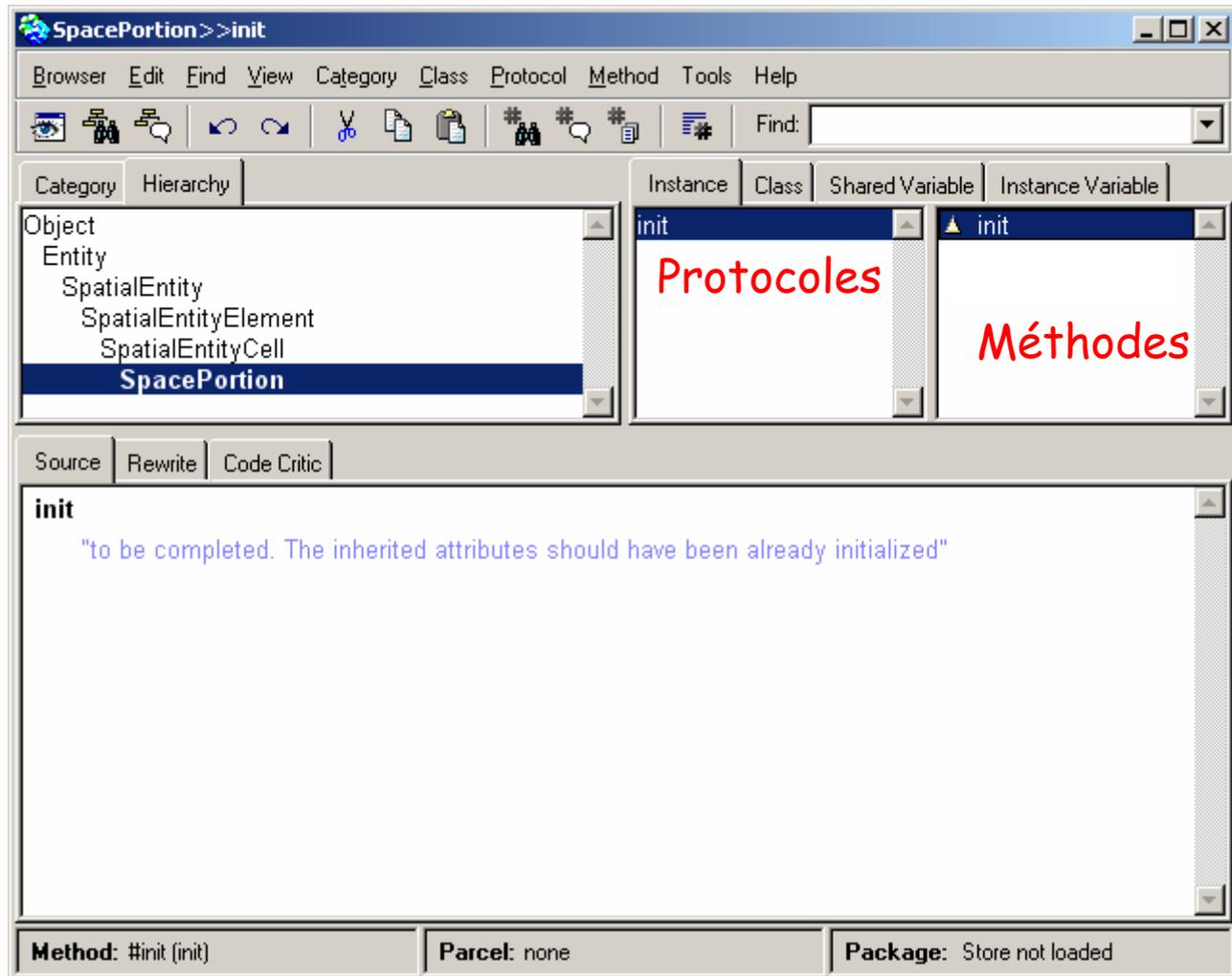
Setter accessor  Getter accessor

Set a default value for this attribut :

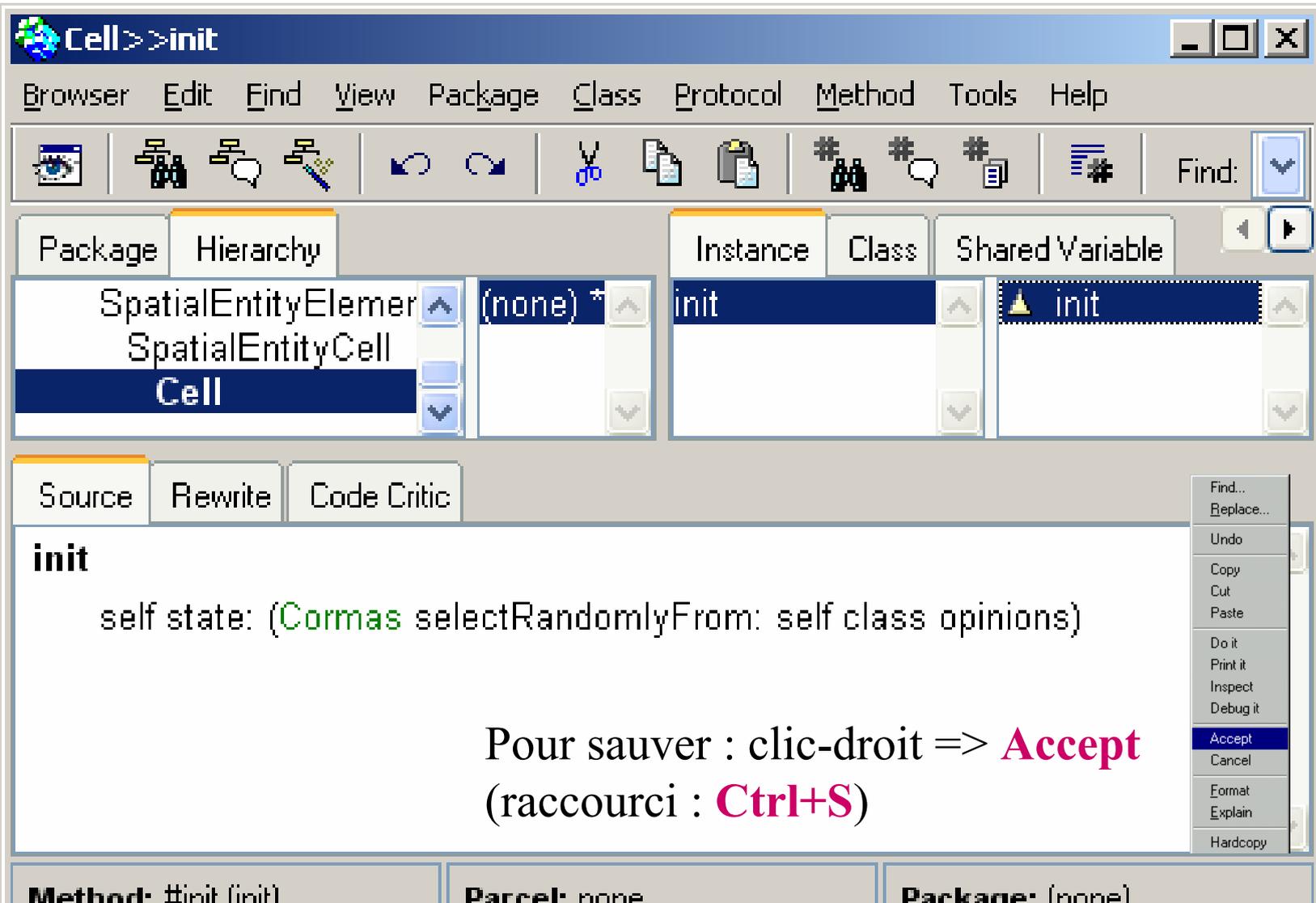
#(#oui #non #blanc #abstention)

OK Without value

# Editer la méthode d'initialisation de la classe "SpacePortion"



# Ecrire une méthode pour initialiser une portion d'espace



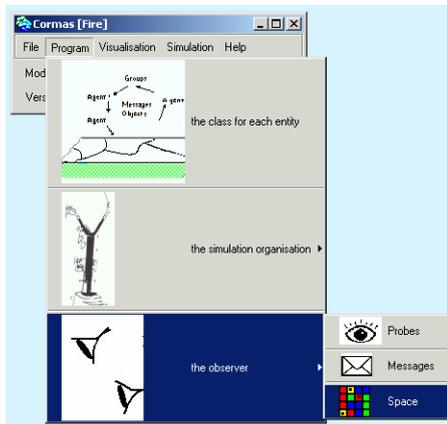
The screenshot shows the Cell environment interface. The title bar reads "Cell >> init". The menu bar includes "Browser", "Edit", "Find", "View", "Package", "Class", "Protocol", "Method", "Tools", and "Help". The toolbar contains various icons for navigation and editing. Below the toolbar, there are tabs for "Package", "Hierarchy", "Instance", "Class", and "Shared Variable". The "Hierarchy" tab is active, showing a tree view with "SpatialEntityElementer", "SpatialEntityCell", and "Cell" (selected). The "Instance" tab is also active, showing the "init" method. The "Class" tab is active, showing the "init" method being edited. The "Source" tab is active, showing the code for the "init" method:

```
init
self state: (Cormas selectRandomlyFrom: self class opinions)
```

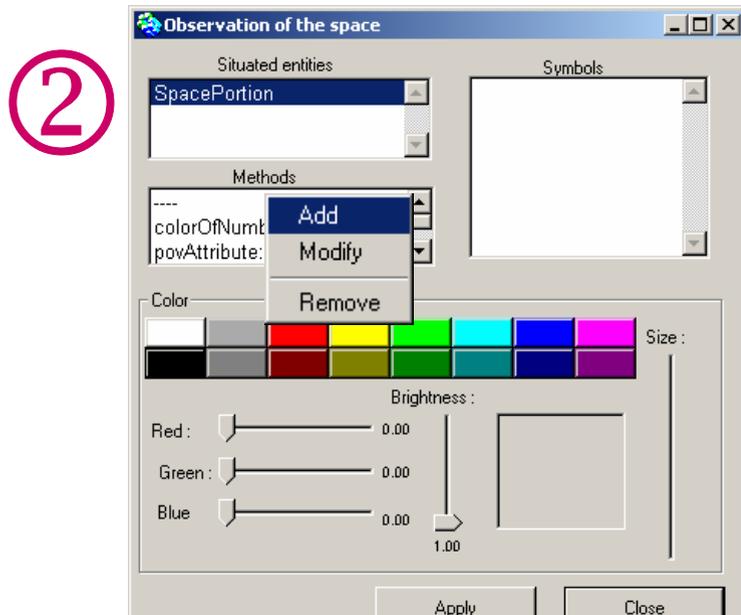
A context menu is open over the code, with the "Accept" option highlighted. The menu items are: Find..., Replace..., Undo, Copy, Cut, Paste, Do it, Print it, Inspect, Debug it, **Accept**, Cancel, Format, Explain, and Hardcopy.

Method: #init (init)      Parcel: none      Package: (none)

# Ecrire une méthode pour observer une portion d'espace



① Programmer → l'observateur → Espace



② Sélectionner **SpacePortion**  
Clic-droit dans la liste "**Methods**"  
Sélectionner **Ajouter**  
Entrer un nom de méthode : **pov**

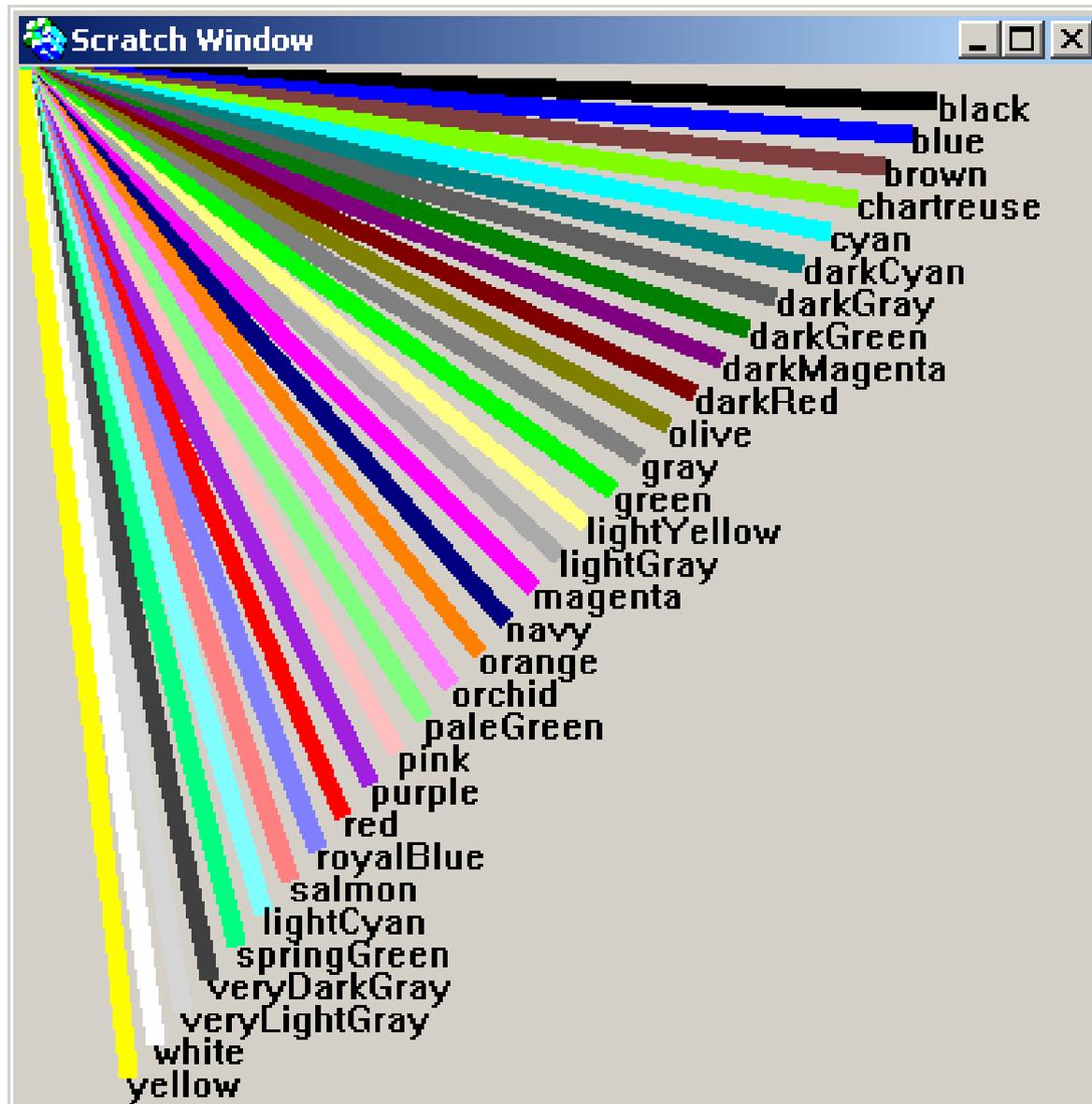
# Ecrire une méthode "pov"

The screenshot shows the Cell IDE interface. The title bar reads 'Cell >> pov'. The menu bar includes 'Browser', 'Edit', 'Find', 'View', 'Package', 'Class', 'Protocol', 'Method', 'Tools', and 'Help'. The toolbar contains various icons for navigation and editing. Below the toolbar, there are tabs for 'Package', 'Hierarchy', 'Instance', 'Class', 'Shared Variable', and 'Instance Variable'. The 'Hierarchy' tab shows a tree view with 'Cell' selected. The 'Instance' tab shows 'init' and 'pov' methods. The 'Instance Variable' tab shows 'pov'. Below these tabs are 'Source', 'Rewrite', and 'Code Critic' buttons. The main editor area displays the following code for the 'pov' method:

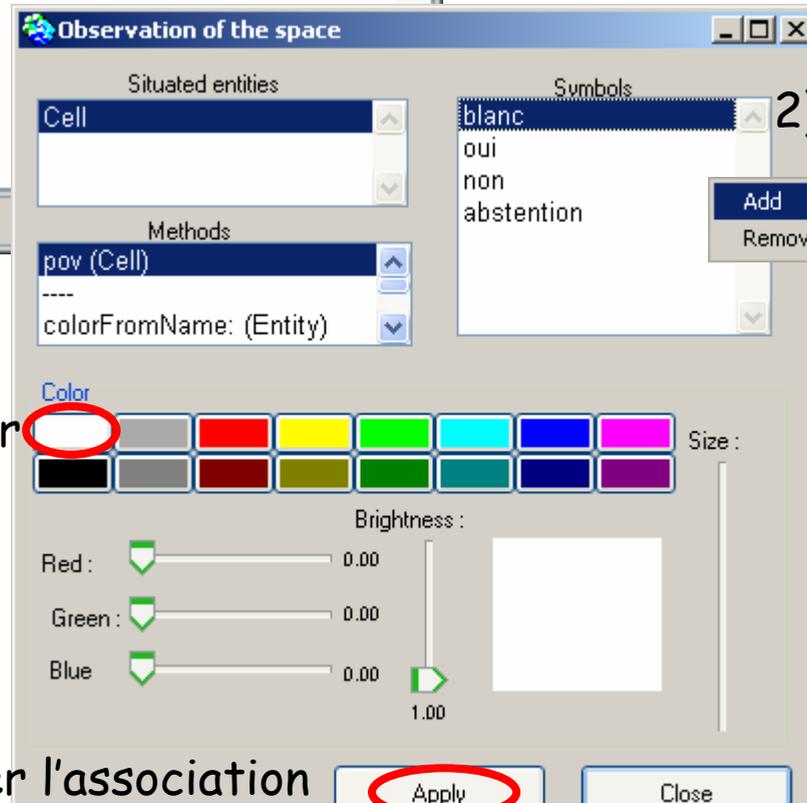
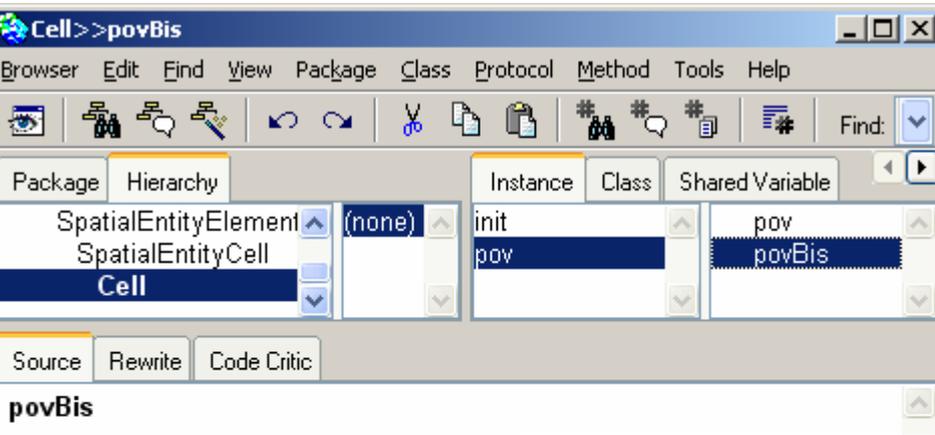
```
pov
self state = #blanc
  ifTrue: [^ColorValue white].
self state = #abstention
  ifTrue: [^ColorValue black].
self state = #non
  ifTrue: [^ColorValue red].
self state = #oui
  ifTrue: [^ColorValue green]
```

Overlaid on the right side of the editor is a text box with the instruction: 'Clic-droit et **Accept** pour sauver (ou raccourci **Ctrl+S**)'. To the right of the code is a context menu with the following items: 'Find...', 'Replace...', 'Undo', 'Copy', 'Cut', 'Paste', 'Do it', 'Print it', 'Inspect', 'Debug it', 'Accept' (highlighted), 'Cancel', 'Format', 'Explain', and 'Hardcopy'. At the bottom of the IDE, there are three status fields: 'Method: #pov (pov)', 'Parcel: none', and 'Package: (none)'.

# Couleurs prédéfinies



# Ecrire une méthode "povBis"



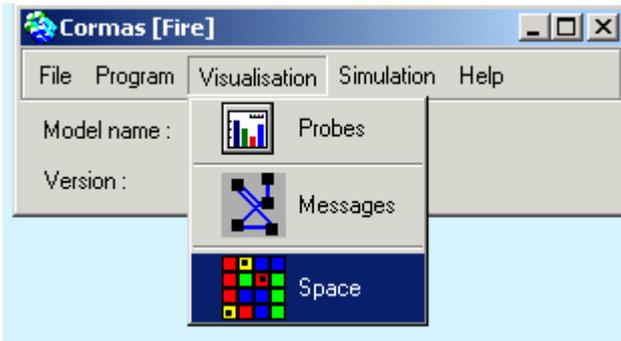
2) Sélectionner le symbole

1) Ajouter le symbole

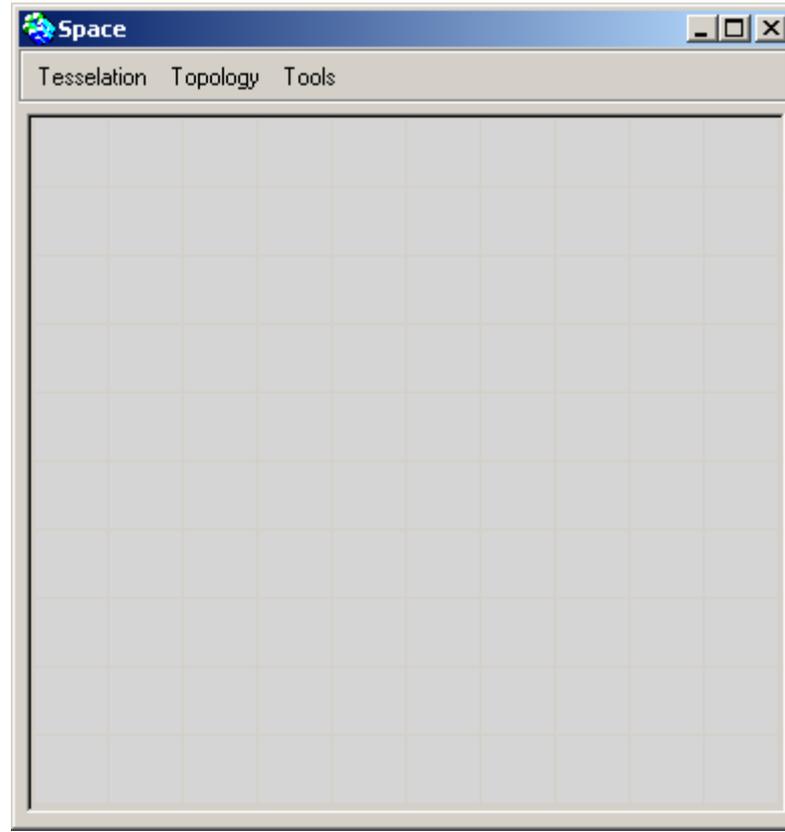
3) Choisir la couleur

4) Sauvegarder l'association

# Ouvrir la grille spatiale

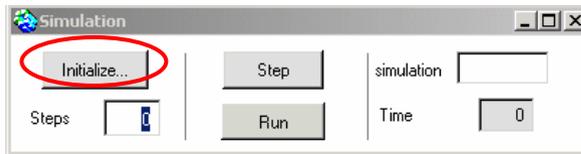


**Visualisation → Espace**

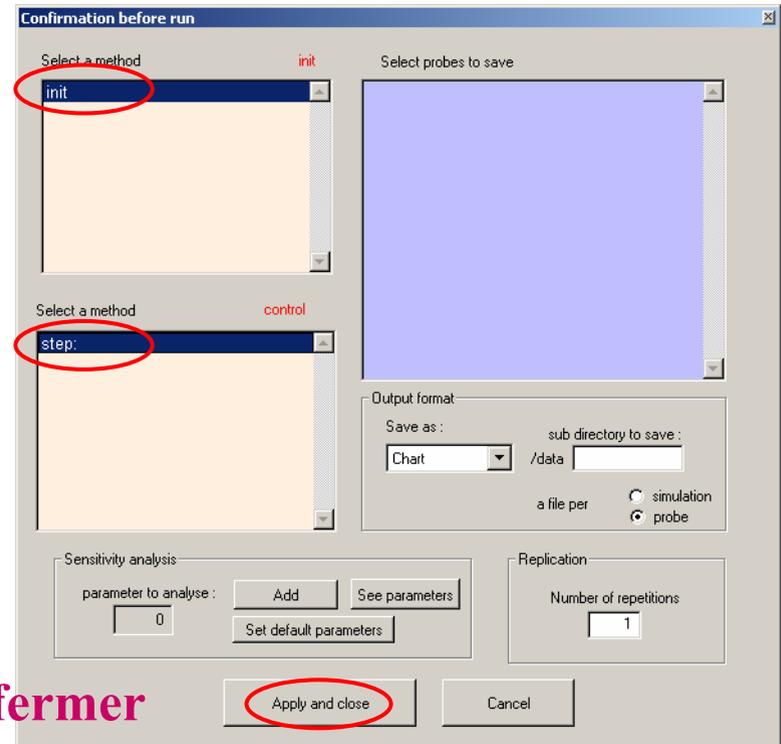


# Tester l'initialisation

Simulation → Interface de simulation



Sélectionner les méthodes: **init** et **step:**



Appliquer et fermer

# Visualiser l'état initial

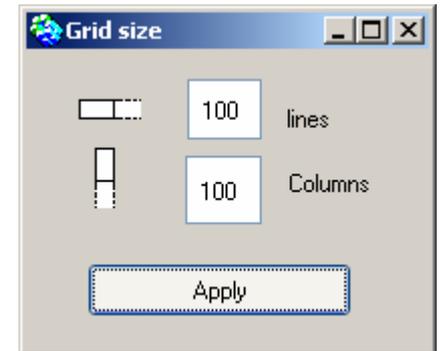
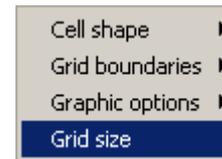
Clic-droit dans la grille

Sélectionner **SpacePortion** → **pov**



Redimensionner la grille

Menu **Topology** → **Grid size**



Fixer le voisinage à 8

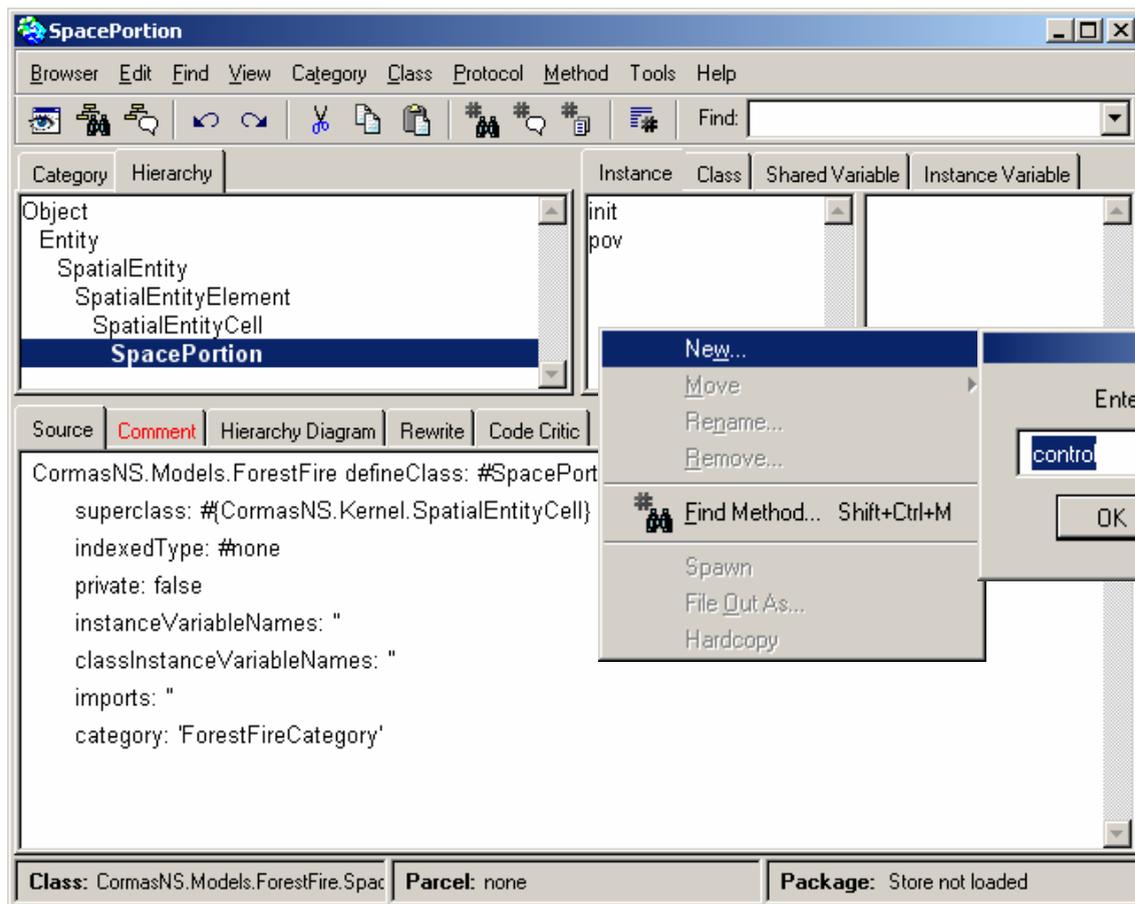
Menu **Topology** → **Cell shape** →



# Ecrire une fonction de transition

Programmer → les classes pour les entités

Double-clic sur **SpacePortion**



Clic-droit dans la liste de “Protocoles”, item **New** pour créer le protocole “**control**”

# Ecrire une fonction de transition

The screenshot shows the Cell IDE interface. The main window displays the source code for the `newState` method in the `Cell` class. The code is as follows:

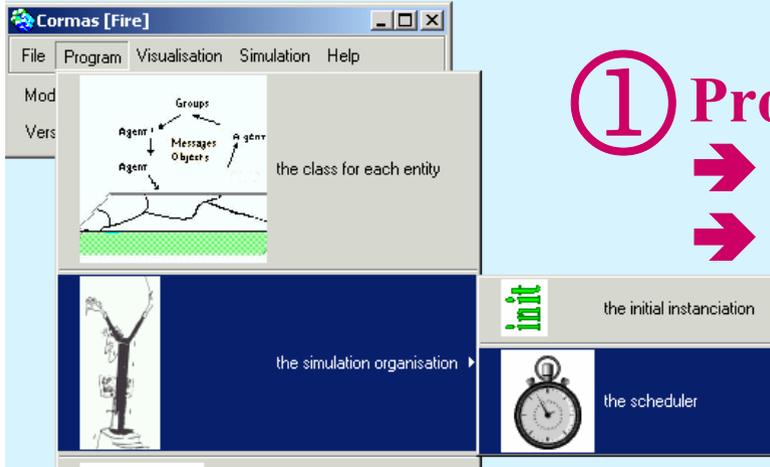
```
newState
| majorite nb |
  majorite := self neighbourhood size / 2.
  self class opinions do:
    [:uneOpinion |
      nb := (self neighbourhood select: [:c | c state = uneOpinion]) size.
      nb > majorite ifTrue: [^self bufferState: uneOpinion]].
  self bufferState: self state
```

Annotations on the right side of the code indicate the next steps: "Clic-droit et **Accept** pour sauver (ou raccourci **Ctrl+S**)".

At the bottom of the IDE, the status bar shows: **Method:** #newState (control), **Parcel:** none, **Package:** (none).

- Find...
- Replace...
- Undo
- Copy
- Cut
- Paste
- Do it
- Print it
- Inspect
- Debug it
- Accept**
- Cancel
- Format
- Explain
- Hardcopy

# Séquencer les activités des entités du modèle



① Programmer  
→ l'organisation de la simulation  
→ l'ordonnanceur

② Modifier la méthode **step**:

The screenshot shows the Elections software interface with the following components:

- Browser**: A menu bar with options: Browser, Edit, Find, View, Package, Class, Protocol, Method, Tools, Help.
- Instance**: A table showing the instance of the step method.
- Class**: A table showing the class of the step method.
- Shared Variable**: A table showing the shared variables of the step method.
- Source**: A menu bar with options: Source, Rewrite, Code Critic.
- Code**: The source code of the step method, showing a self stepSynchronously: t call.
- Method**: A table showing the method of the step method.
- Parcel**: A table showing the parcel of the step method.
- Package**: A table showing the package of the step method.

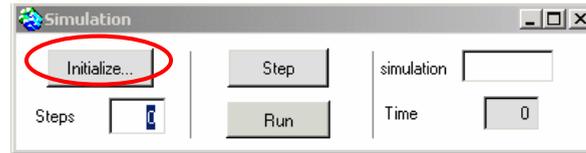
Instance	Class	Shared Variable
accessing	▲ step:	
control		
init		

```
step: t
  self stepSynchronously: t
```

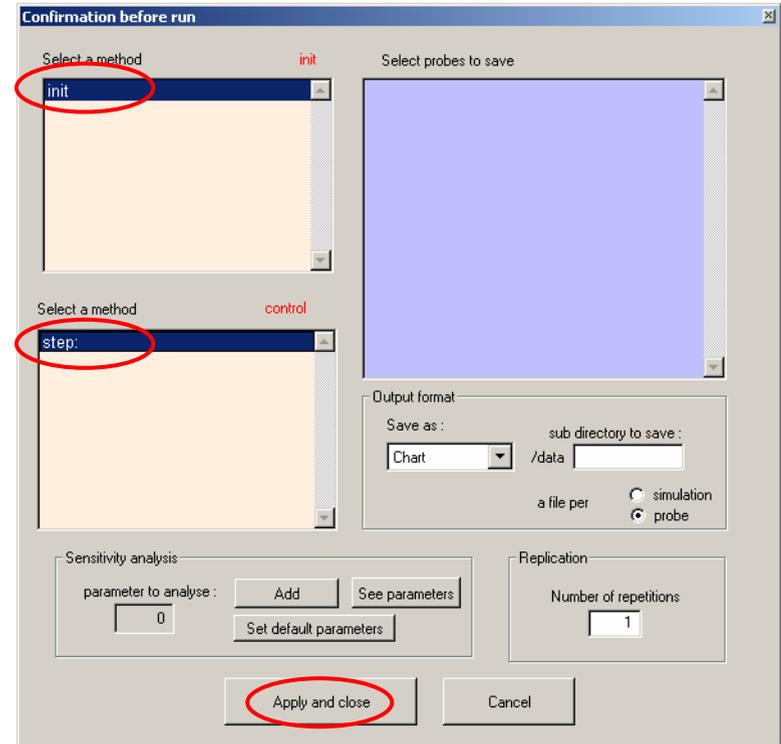
Method	Parcel	Package
#step: (control)	none	(none)

# Lancer une simulation

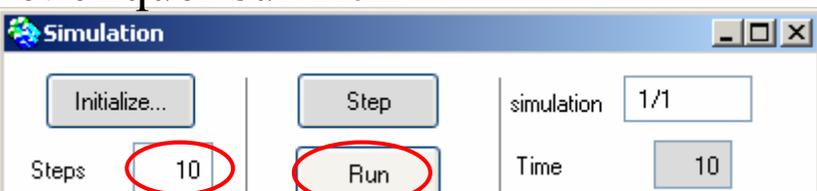
Simulation → Interface de simulation



Sélectionner les méthodes **init** et **step**



Entrer un nombre de pas de temps et cliquer sur **Run**



Appliquer et fermer