

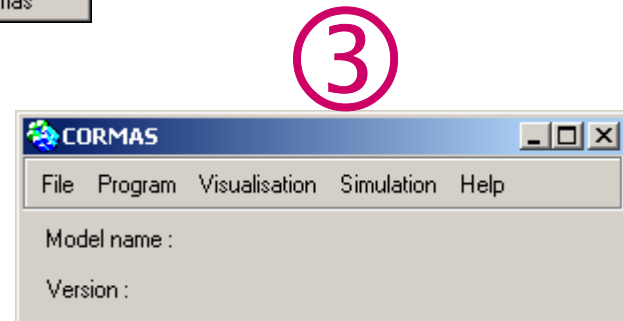
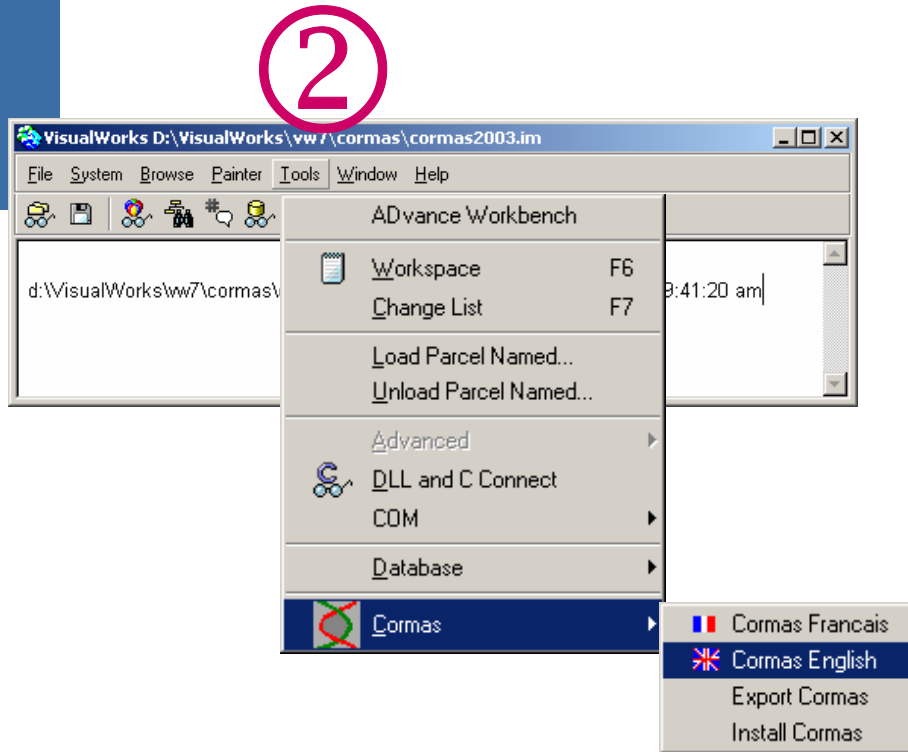
Introduction à Cormas

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

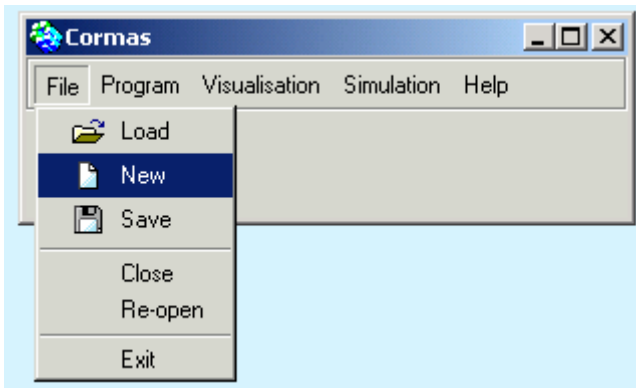
Lancer Cormas



Créer un nouveau modèle

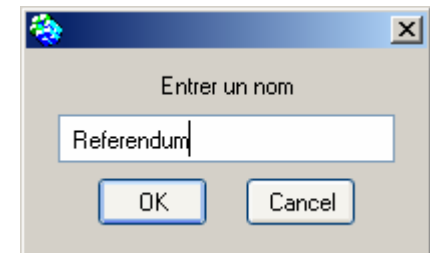
Dans le menu “Fichier”, sélectionner “new” et taper le nom “Referendum” comme nom de nouveau modèle

①



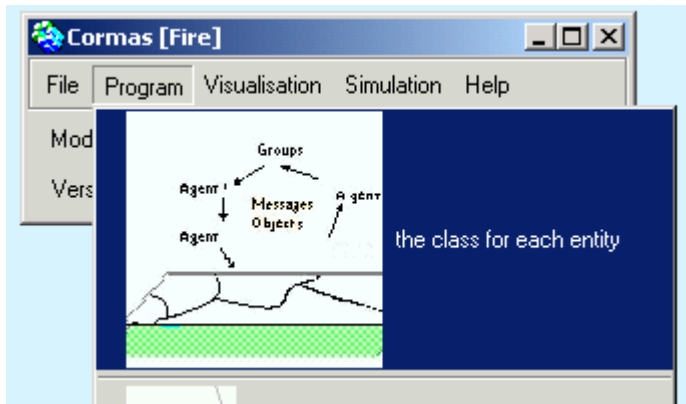
File → New

②



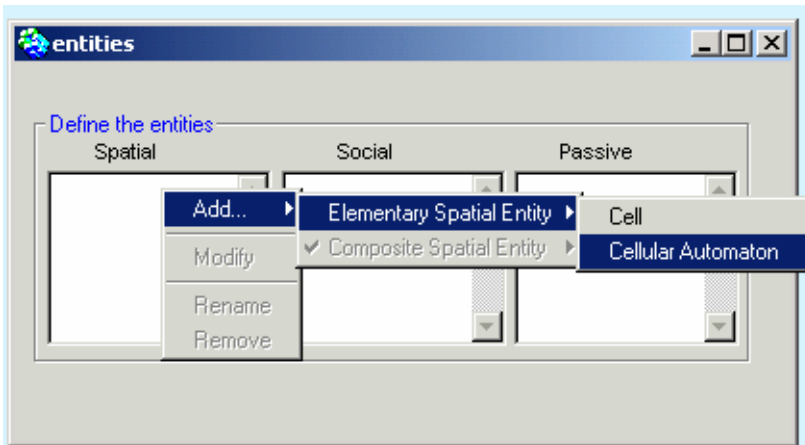
Créer une entité spatiale

1



Programmer → les classes pour les entités

2



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

Choisir **Entité Spatiale Elementaire**

Puis **Cellule Automate**

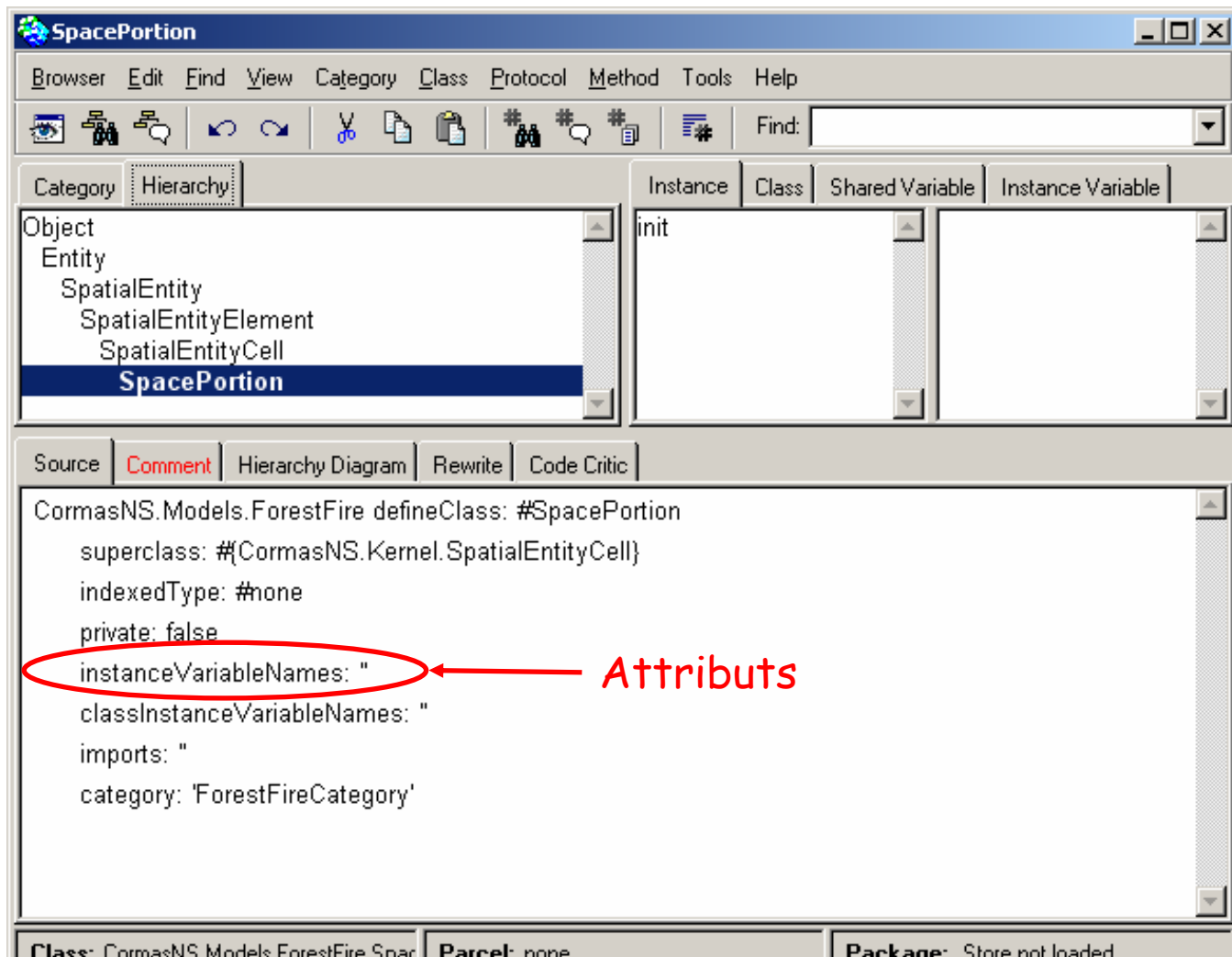
Taper le nom: **SpacePortion**

Cellule ≠ **Cellule Automate**

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

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

Fenêtre de définition des classes: le “Refactoring Browser”



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 workspace is divided into several panes. On the left, a 'Hierarchy' pane shows a tree of classes: Entity, SpatialEntity, SpatialEntityElement, SpatialEntityCell, and Cell (selected). The middle pane shows the 'Class' view for the selected class, displaying 'init' under both 'Instance' and 'Shared Variable' tabs. The bottom pane shows the source code for the class definition. The code is as follows:

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

The line `classInstanceVariableNames: 'opinions '` is circled in red, and a red arrow points from the text "Variable de classe" to it.

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 the Cell class editor interface. The 'Class' tab is selected, and the context menu is open. The 'Instance Variables' option is highlighted, and the 'Create CORMAS Accessors' option is selected. The 'Default value chooser' dialog is open, showing the default value for the 'opinions' attribute.

Cell class
Browser Edit Find View Package Class Protocol Method Tool

Package Hierarchy Instance **Class** SH

Entity
SpatialEntity
SpatialEntityElementer
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'
```

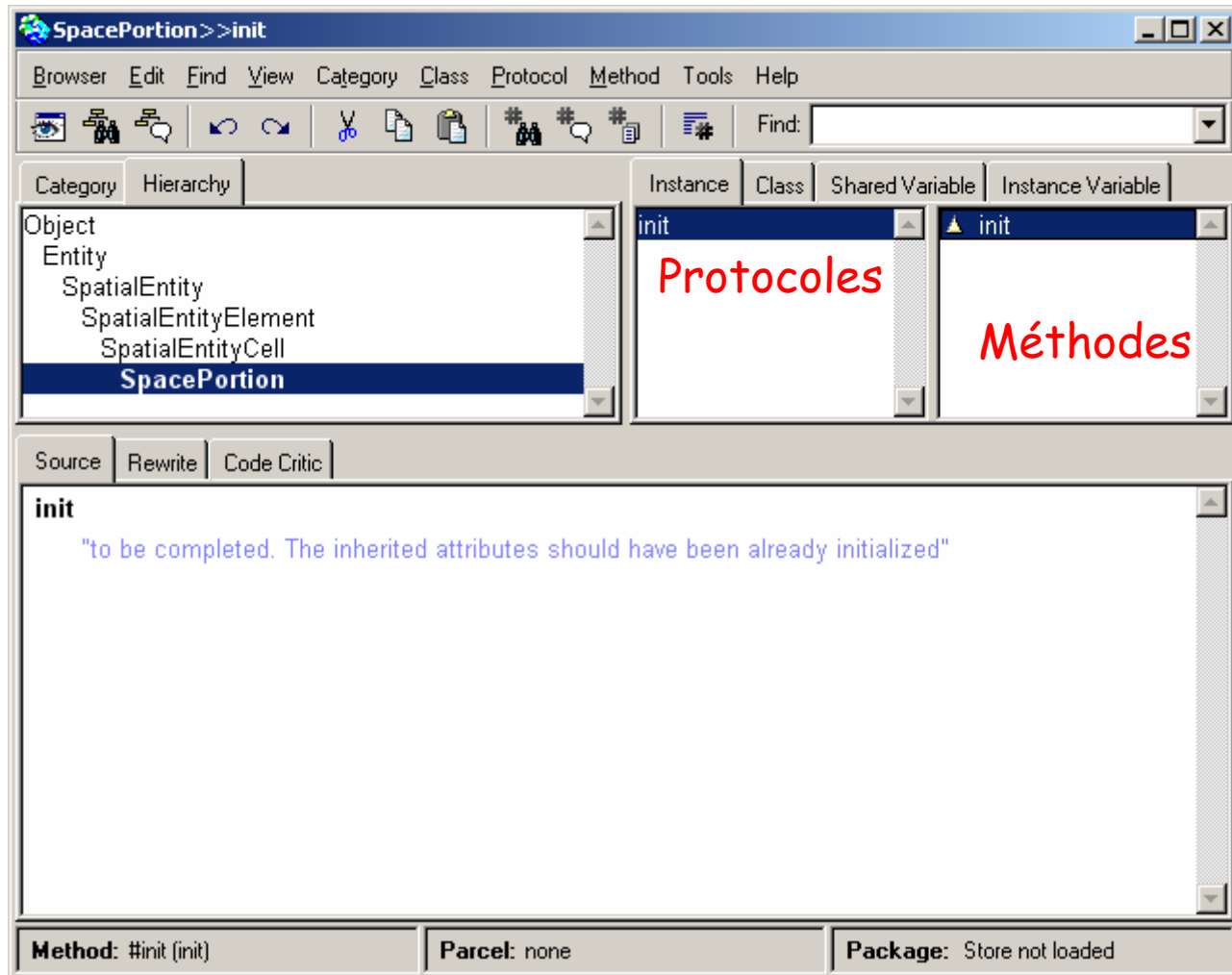
Class: ComasNS.Models.Elec **Parcel:** none **Package:** (none)

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

Clic-droit →

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 selected. The "Shared Variable" tab is active, showing the "init" method selected. Below the tabs, there are buttons for "Source", "Rewrite", and "Code Critic". The main editing area shows the following code:

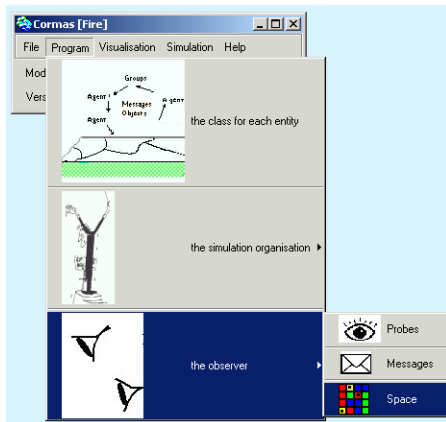
```
init
self state: (Cormas selectRandomlyFrom: self class opinions)
```

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

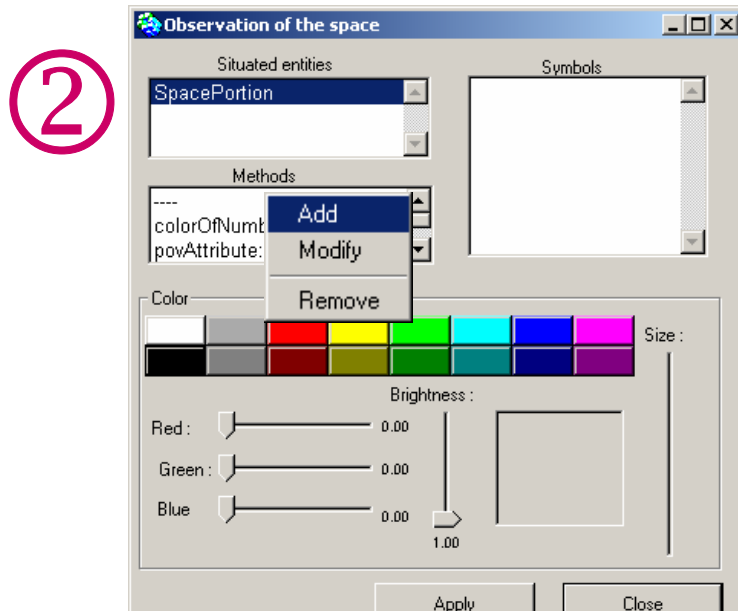
At the bottom of the interface, there are three status bars: "Method: #init (init)", "Parcel: none", and "Package: (none)".

Pour sauver : clic-droit => **Accept**
(raccourci : **Ctrl+S**)

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 top menu bar includes Browser, Edit, Find, View, Package, Class, Protocol, Method, Tools, and Help. Below the menu is a toolbar with various icons for navigation and editing. The main workspace is divided into several panes:

- Package Hierarchy:** Shows a tree view with 'Object' at the top, followed by 'Entity', 'SpatialEntity', 'SpatialEntityElement', 'SpatialEntityCell', and 'Cell' (selected).
- Instance:** Shows 'init' and 'pov' (selected).
- Class:** Shows 'pov' (selected).
- Shared Variable:** Empty.
- Instance Variable:** Empty.

Below the panes are buttons for 'Source', 'Rewrite', and 'Code Critic'. The main editor displays the following code for the 'pov' method:

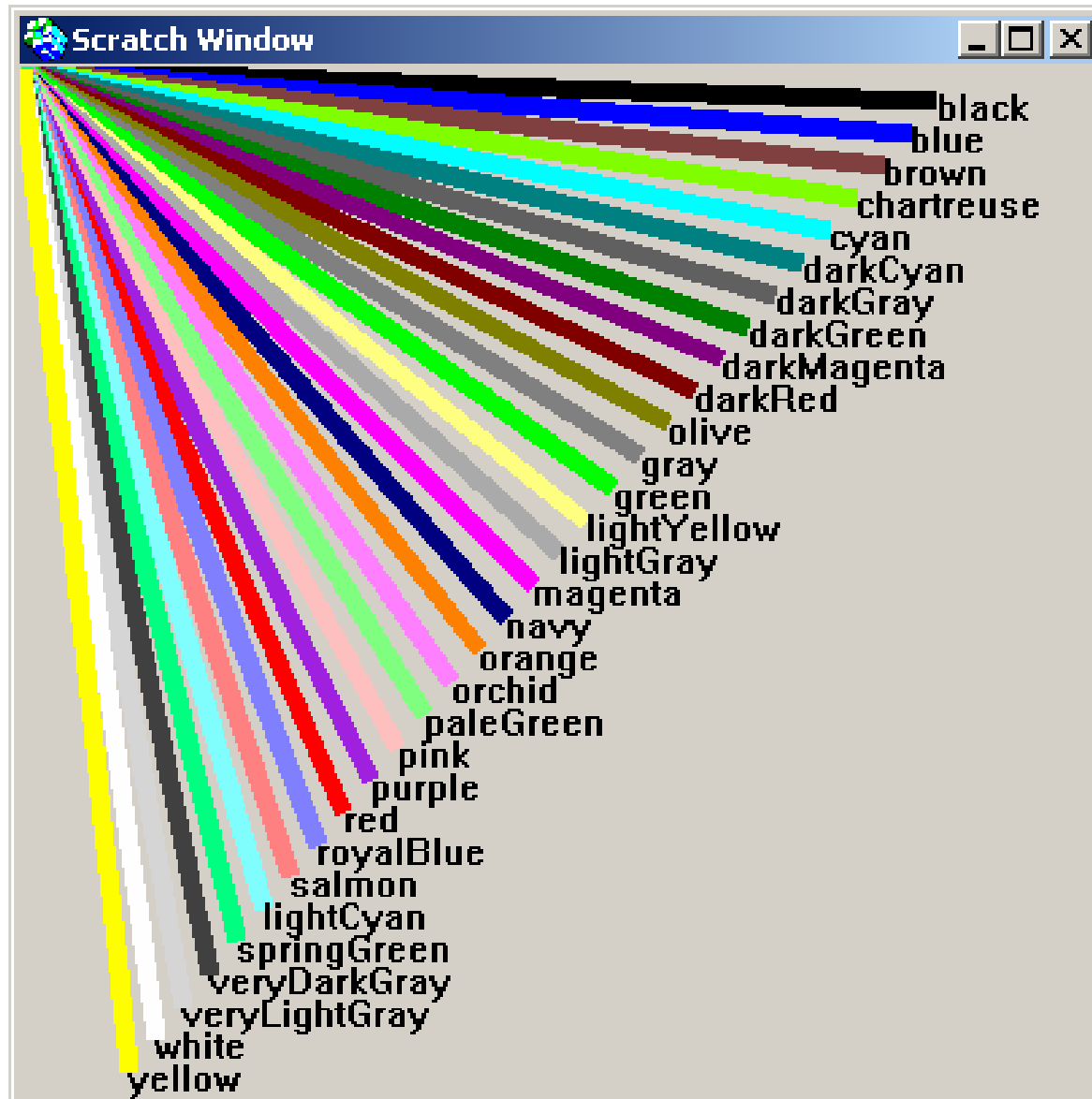
```
pov
self state = #blanc
  ifTrue: [^ColorValue white].
self state = #abstention
  ifTrue: [^ColorValue black].
self state = #non
  ifTrue: [^ColorValue red].
self state = #oui
  ifTrue: [^ColorValue green]
```

On the right side, a context menu is open, showing options like Find..., Replace..., Undo, Copy, Cut, Paste, Do it, Print it, Inspect, Debug it, **Accept** (highlighted), Cancel, Format, Explain, and Hardcopy.

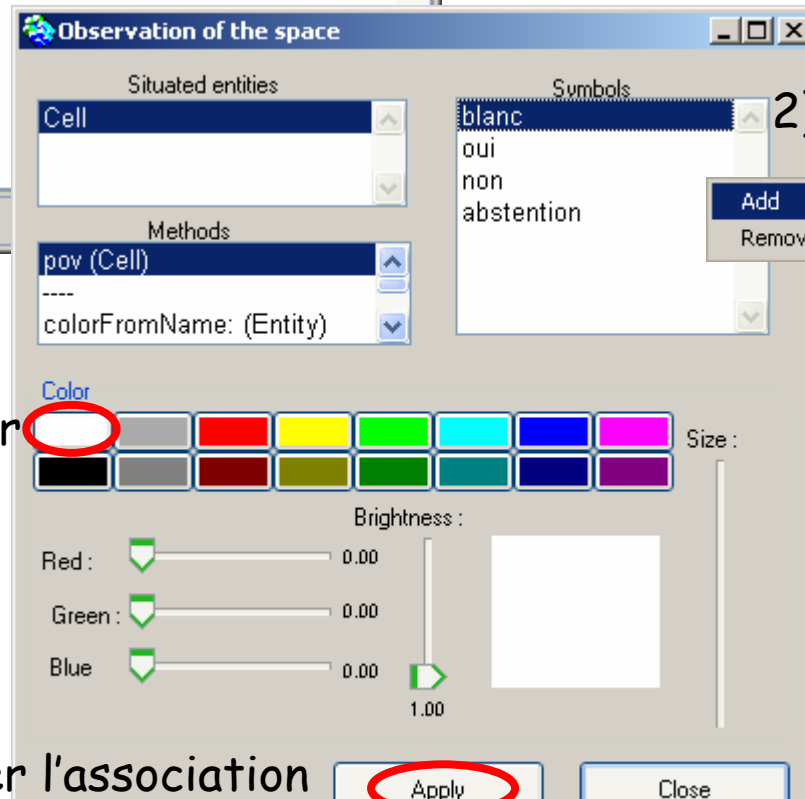
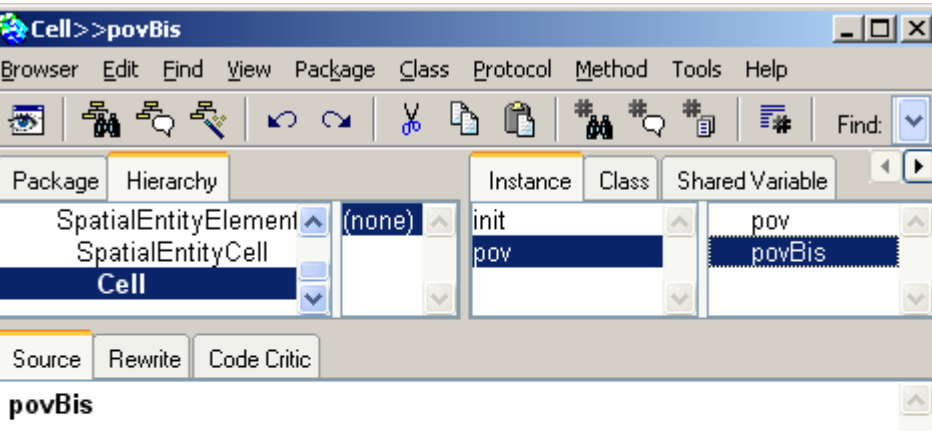
At the bottom, the status bar shows: **Method:** #pov (pov) | **Parcel:** none | **Package:** (none)

Clic-droit et **Accept** pour sauver
(ou raccourci **Ctrl+S**)

Couleurs prédéfinies



Ecrire une méthode "povBis"



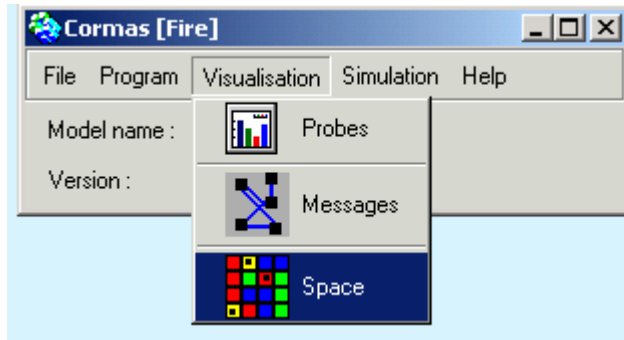
2) Sélectionner le symbole

1) Ajouter le symbole

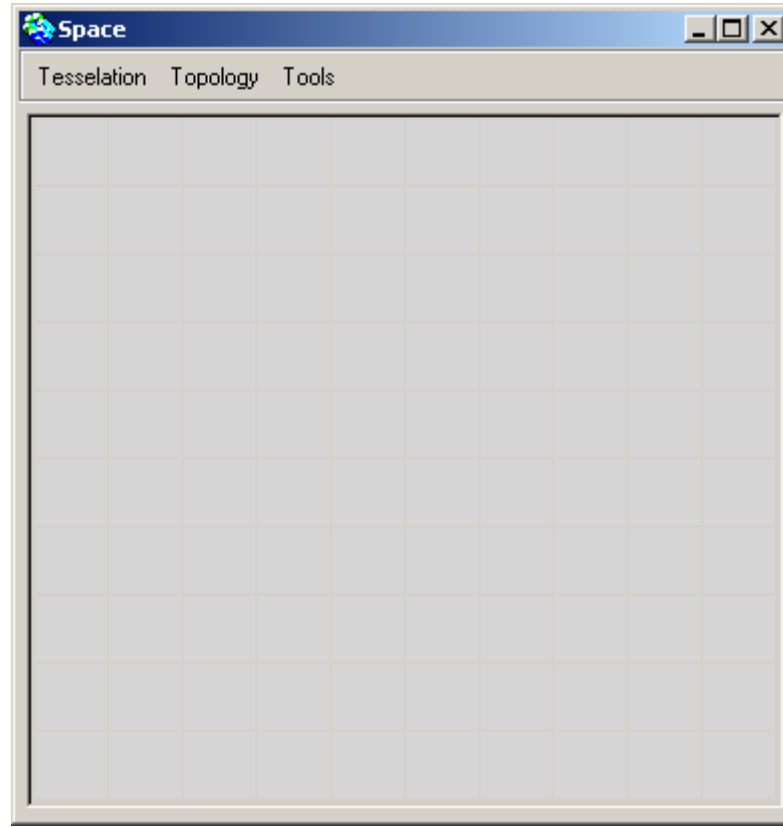
3) Choisir la couleur

4) Sauvegarder l'association

Ouvrir la grille spatiale

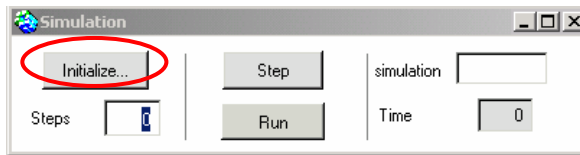
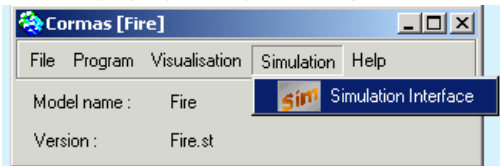


Visualisation → Espace

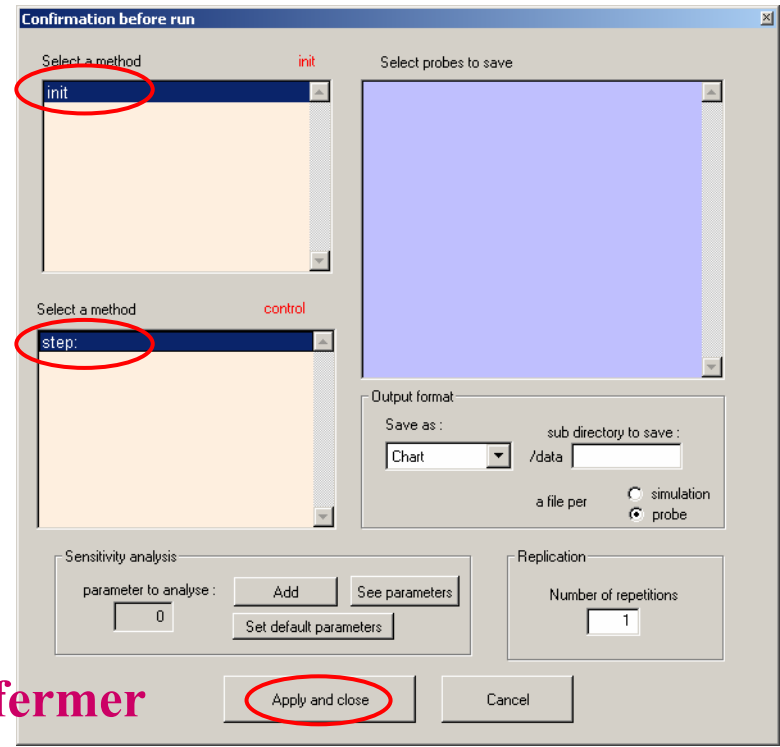


Tester l'initialisation

Simulation → Interface de simulation



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

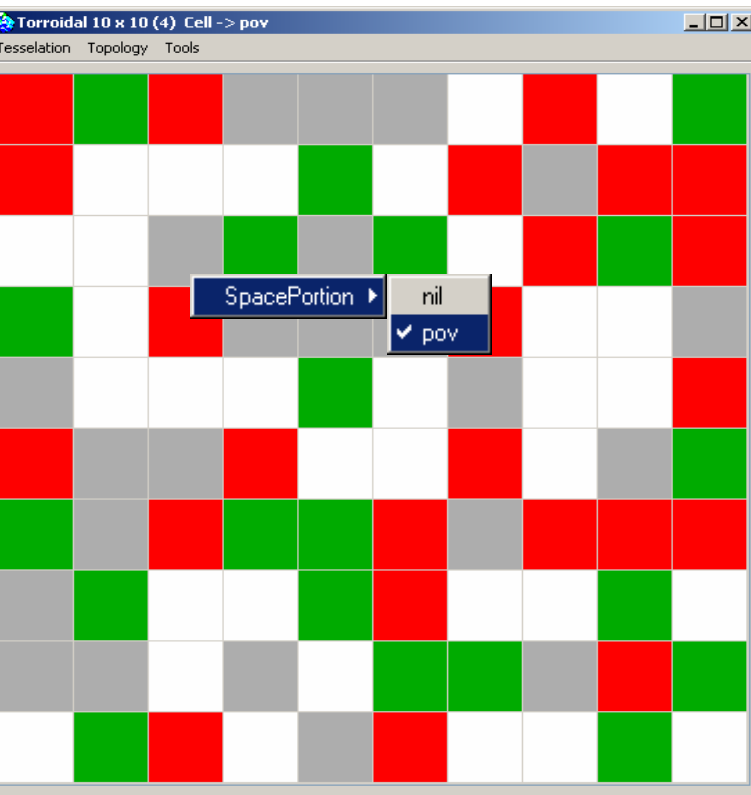


Appliquer et fermer

Visualiser l'état initial

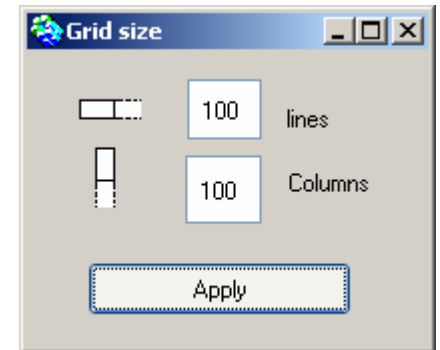
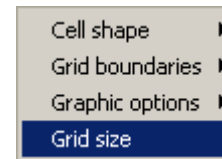
Clic-droit dans la grille

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



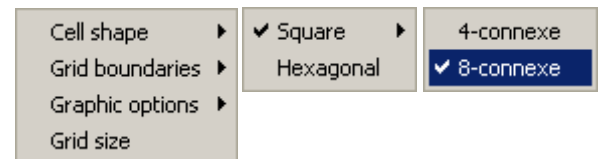
Redimensionner la grille

Menu **Topology** → **Grid size**



Fixer le voisinage à 8

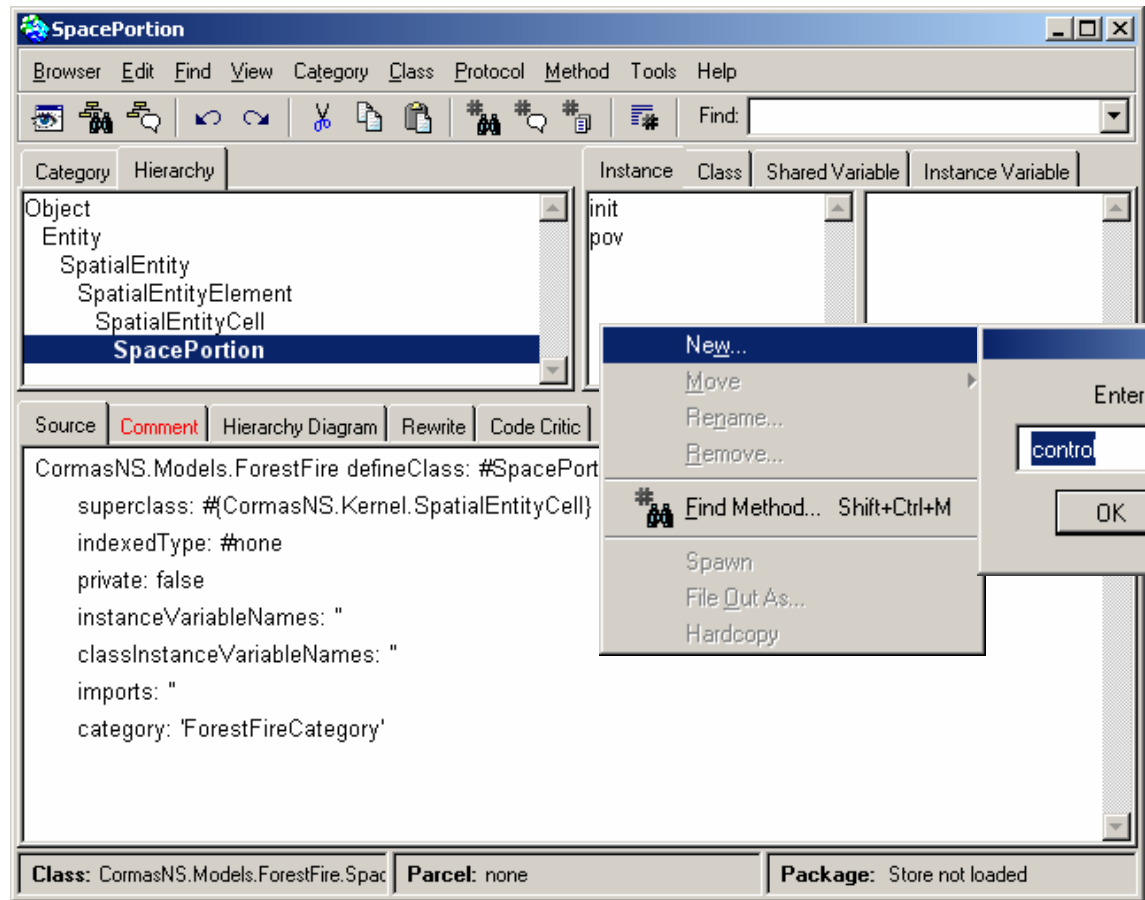
Menu **Topology** → **Cell shape** →



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:

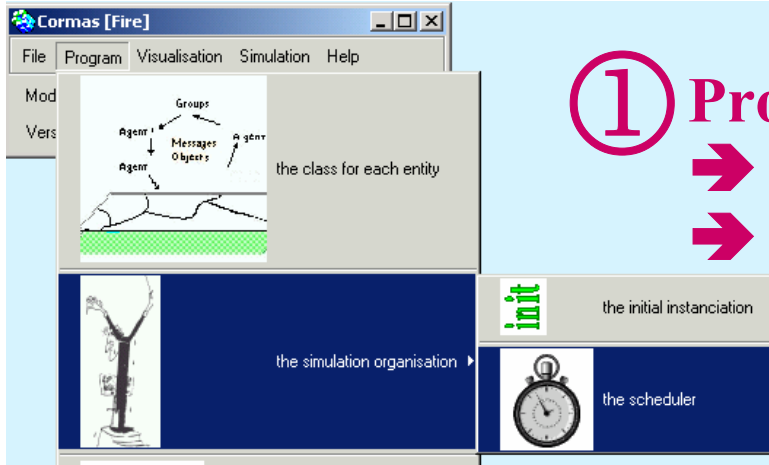
```
newState
| majorite nb |
  majorite := self neighbourhood size / 2.
  self class opinions do:
    [:uneOpinion |
      nb := (self neighbourhood select: [:c | c state = uneOpinion]) size.
      nb > majorite ifTrue: [^self bufferState: uneOpinion]].
  self bufferState: self state
```

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).

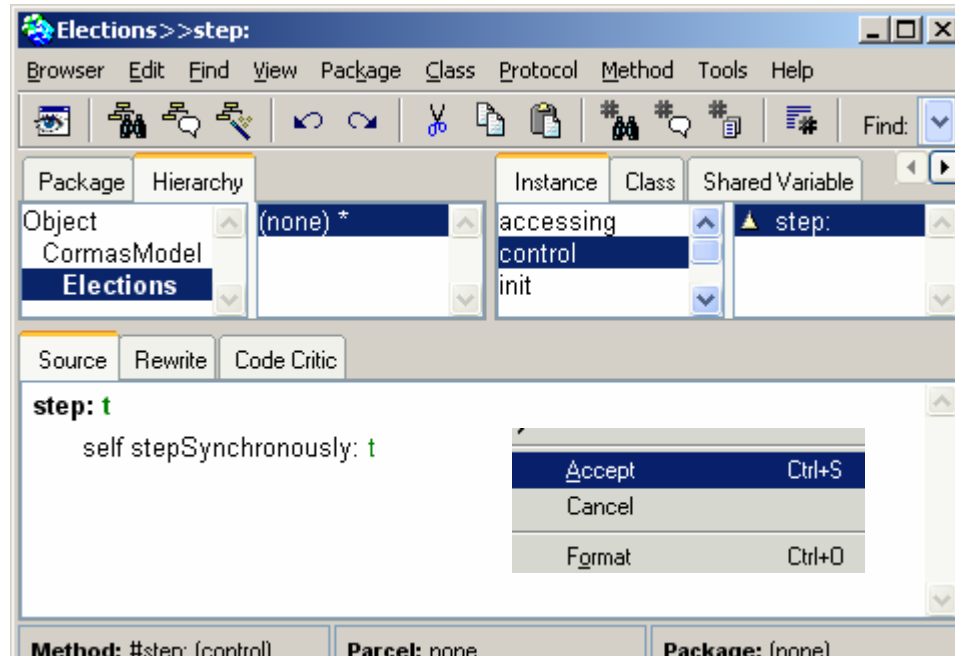
- Find...
- Replace...
- Undo
- Copy
- Cut
- Paste
- Do it
- Print it
- Inspect
- Debug it
- Accept**
- Cancel
- Format
- Explain
- Hardcopy

Séquencer les activités des entités du modèle



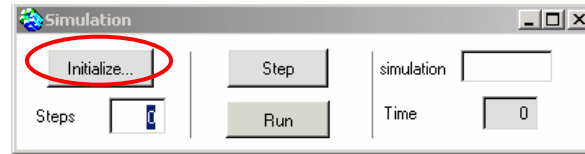
① Programmer
→ l'organisation de la simulation
→ l'ordonnanceur

② Modifier la méthode **step**:

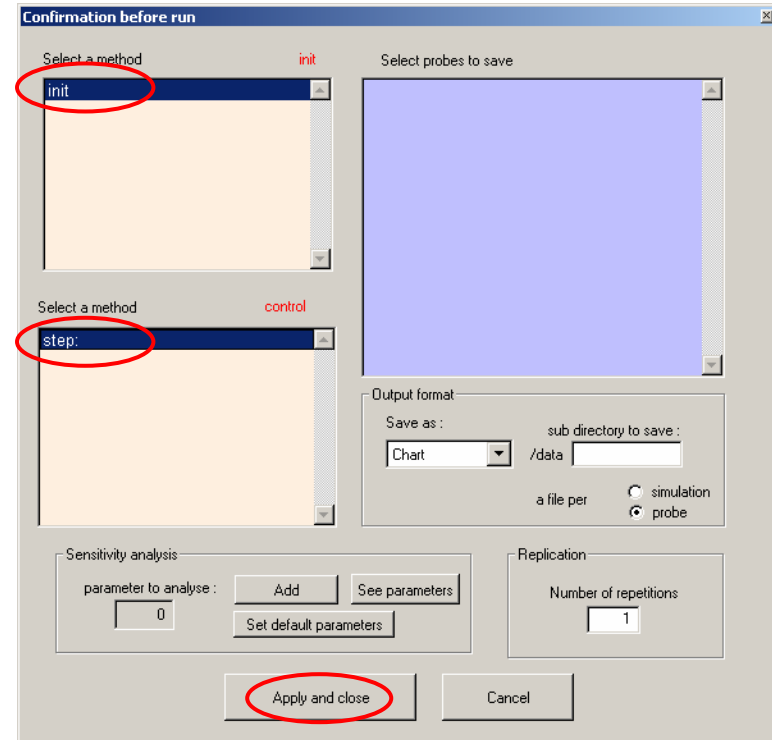


Lancer une simulation

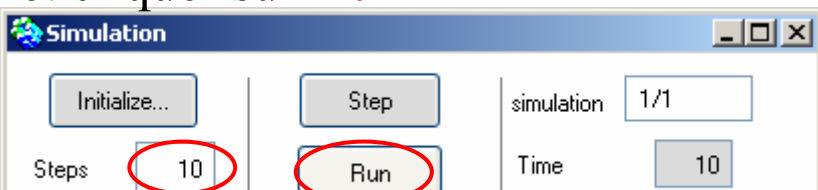
Simulation → Interface de simulation



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



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



Appliquer et fermer