CORMAS : A multiagent simulation toolkit to model natural and social dynamics at multiple scales

Christophe Le Page [a*], François Bousquet [a], Innocent Bakam [b], Alassane Bah [c], Christian Baron [a]

[a] *CIRAD, TA60/15, 73 av. J.-F. Breton, 34938 Montpellier Cedex 5, France*
[b] *Yaounde I University, P.O. Box 812, Yaounde, Cameroon*
[c] *Ecole Supérieure Polytechnique, Dakar, Senegal*

[*] Corresponding author. Tel: +33 467 593 828; fax: +33 467 593 827; e-mail: lepage@cirad.fr

Abstract

Dealing with multiple scales is often a key question in renewable resources management. In some cases, the decision to incorporate a spatial entity is influenced by the fact that information is available at this level. In other cases, the system dynamics is intrinsically linked to a specific spatial entity, which should obviously be taken into account in the model. Nevertheless, it is important to have the possibility to manipulate and to incorporate into the same model spatial entities defined at different hierarchical levels.
Originated from the field of Distributed Artificial Intelligence, Multi-Agent Systems (MAS) are potentially suitable for linking several hierarchical levels. In a MAS, an agent is a computerized autonomous entity that is able to act locally in response to stimuli from the environment or to communication with other agents. Cormas (Common-pool Resources and Multi-Agent Systems) is a multi-agent simulation platform specially designed for renewable resource management. It provides the framework for building models of interactions between individuals and groups sharing natural resources. With Cormas, the design of the spatial support rests on spatial entities, which are themselves a category of agents. When these entities yield resources, they are competent to arbitrate their allocation, according to pre-defined protocols, between concurrent demands formulated by other agents exploiting these resources. The way agents are exploiting resources may depend on their own partial representation of the environment, which are based on these same spatial entities.
Following a general overview of the Cormas simulation platform, examples of models built by using this toolkit are presented, by emphasizing the overlapping of their multiple hierarchical scales. Finally, the use of multi-agent systems to represent knowledge on processes at various levels of complexity and to simulate their interactions according to a bottom-up approach for understanding landscape dynamics are discussed.

Keywords: agent-based simulation, multi-agent system, natural resources management, object-oriented programming, spatially explicit individual-based model.

Introduction

There is a trend in ecological modelling to define spatially-explicit individual-based models (Grimm, 1999; Lomnicki, 1999). One of the main challenges is to connect landscape patterns to population processes (Kawata and Toquenaga, 1994; Kareiva and Wennergren, 1995). In this way towards a behavioural ecology of ecological landscapes, the major problem to face deals with scale, as different levels of aggregation are classically used by ecologists and behaviourists (Lima and Zollner, 1996). In the meanwhile the ecological modelling community has shown a growing interest about the relations among scales (Wu and Levin, 1997). Ecological systems viewed as hierarchical dynamic mosaics of patches are thus generated and maintained by processes of patch formation, patch development and disappearance. Mainly based on object-oriented conception, several software frameworks have implemented such hierarchically structured spatial environments. Liu and Ashton (1998) have developed a landscape model (FORMOSAIC) for simulating forest dynamics based on four spatial scales: the landscape, made of forests, a forest being made of 10x10 grid cells, a cell containing many individual trees located at the point level. ECOTALK (Baceco and Lingeman, 1992) and HOBO (Lhotka, 1994) are simulation systems based on Smalltalk with an overall space structure being a tree-like configuration that facilitates specification of various spatial organisations. ECOSIM (Lorek and Sonnenschein, 1999) is a C++ framework in which the basic unit of environment is a discrete cell, hierarchical environments being defined by structuring the cells using topologies. SHALOM (Ziv, 1998), a C++ landscape simulation model with several predefined physical classes (landscape, habitat, cell, patch) is used to investigate the effect of habitat heterogeneity on species diversity patterns.

The need to define realistic virtual landscapes is strengthened when human activities (agriculture, hunting, forestry, ...) are part of the system and when the goal of the model is to understand the interactions between natural and social dynamics to give some insights about renewable resources management. We present in this paper an agent-based simulation framework involving multiagent system (MAS) in order to understand the complexity of these interactions. Models of this type have been developed for irrigated land management in Senegal (Barreteau and Bousquet, in press), herd mobility in the Sahel (Bousquet et al., 1999a), hunting wild meat in Eastern Cameroon (Bousquet et al., in press). In association with these research projects, tools have been developed and an approach to modelling has been proposed (Bousquet et al., 1999b). We have developed a simulation environment called Cormas that combines these tools. Before describing this simulation framework, multiagent systems are shortly presented, and agent-based simulation softwares are reviewed.

Multiagent Systems

Originated from the field of Distributed Artificial Intelligence (DAI), multiagent systems are based on the principles of distribution and interaction (Ferber, 1999; Weiss, 1999). Creating a MAS involves reproducing for experimental purposes an artificial world. Multiagent systems are made of collections of agents, an agent being a computerised autonomous entity that is able to act locally in response to stimuli from the environment or to communication with other agents (figure 1).

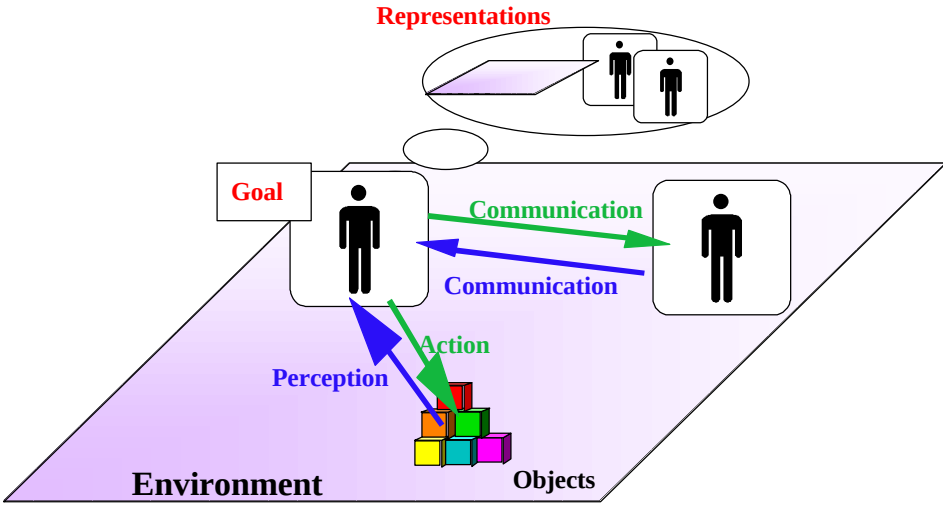


***Figure 1**. MAS general principles (from Ferber, 1995)*

Agents have:

- internal data representations (memory or state),
- means for modifying their internal data representations (perceptions),
- means for modifying their environment (behaviours)

The key-concept of MAS concerns the interactions between agents. These interactions may occur through the environment, either by being at the same place at the same time or less directly (for instance by ownership, resource depletion, pheromone depletion), or may occur explicitly, either via direct communication (exchanges of messages) or via transactions (e.g., financial).

Knowledge is represented at the microscopic level and phenomena are represented at the macroscopic level. For several years, MAS have been used to study the relationships among different hierarchical levels. The macroscopic level is represented in the form of a group or macro-agent. The question is how interactions between entities at the microscopic level can cause phenomena at the macroscopic level, and conversely, how the macroscopic level affects processes at the microscopic level. It is possible to depict interactions between agents at different levels in a single MAS. For instance, a farmer agent interacts with a tree agent while a village agent interacts with a forest agent. The challenge is to move from the concept of a simple hierarchy to that of dynamics interconnected hierarchies, which corresponds to the reality of the natural environment.

Multiagent simulation platforms

For several years now, multiagent simulation software has been available. User groups (including ecologists and sociologists) are organized around generic tools that facilitate the construction of models and offer facilities ("virtual laboratories") for monitoring and analysing simulation trials. The example of Swarm (http://www.swarm.org) —which uses the object-oriented language Objective-C for writing code and Tcl/Tk for developing interfaces (Minar et al., 1996)— clearly reflects the current trend. Since the launch of the project at the Santa Fe Institute in 1994, groups using Swarm have joined forces to try and resolve common problems. As a result, new software based on Swarm, with specific applications for

different disciplinary fields (ecology, for example), has been developed. In this category of "ecology-specialised" agent-based simulation platform, Echo (Hraber et al., 1997) is based on the fact that evolution is built in as a fundamental components of the system, Sugarscape (Epstein and Axtell, 1994) is focusing on artificial societies by putting emphasise into exchanges of goods between agents.

The Cormas simulation platform

Cormas (Common-pool Resources and Multiagent Systems; Bousquet et al., 1998) has been developed to provide a multiagent framework that can be used to simulate the interactions between a group of agents and a shared environment holding natural resources (figure 2). It aims to simplify the task of simulating resource management.

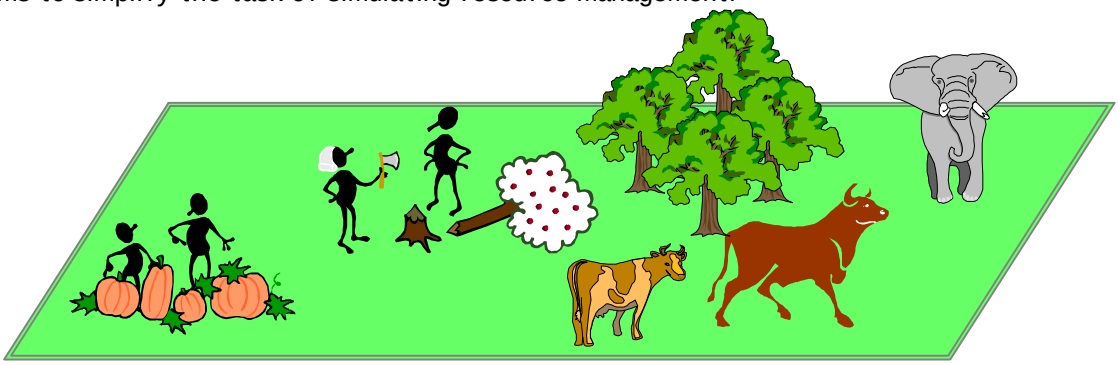


Figure 2. *MAS and Natural Resources Management. The MAS environment is made of a collection of elementary spatial entities (cells) defining a topological support for the agents. Each cell may hold some resources that are used or managed by the agents in a specific way.*

As claimed by Resnick (1996) who initiated the StarLogo project, new computational tools can be useful to develop heuristics and metaphors to help people think about decentralised systems in a new way. Following this idea, CORMAS provides a set a heuristics for thinking about common-pool resources management in a decentralised and distributed way. The main goal of the CORMAS tool is not to make accurate predictions about the behaviour of complex systems, but rather to provide a framework to help people develop new ways of thinking. As people make use of the simulation tool, they naturally engage in thinking about these ideas. We had to find a compromise to enable people to build their own models as easier as possible, while preserving flexibility and providing interesting functionalities especially to define realistic environments.

Cormas is based on the software VisualWorks which, in turn, is a programming environment based on Smalltalk. Cincom, the American company that markets VisualWorks, distributes the software freely (for educational and research purposes). Cormas is also available to the scientific community (http://www.cirad.fr/presentation/programmes/espace/cormas/eng/index.shtml) in the form of a set of Smalltalk classes that are representing generic social entities encoding behaviour classically exhibited by actors exploiting natural resources, but also generic spatial entities organised in a hierarchical way.

Basic steps to build an agent-based model with Cormas

The architecture of the main interface of the platform (figure 3) has been designed to guide the user of Cormas during the modelling process.

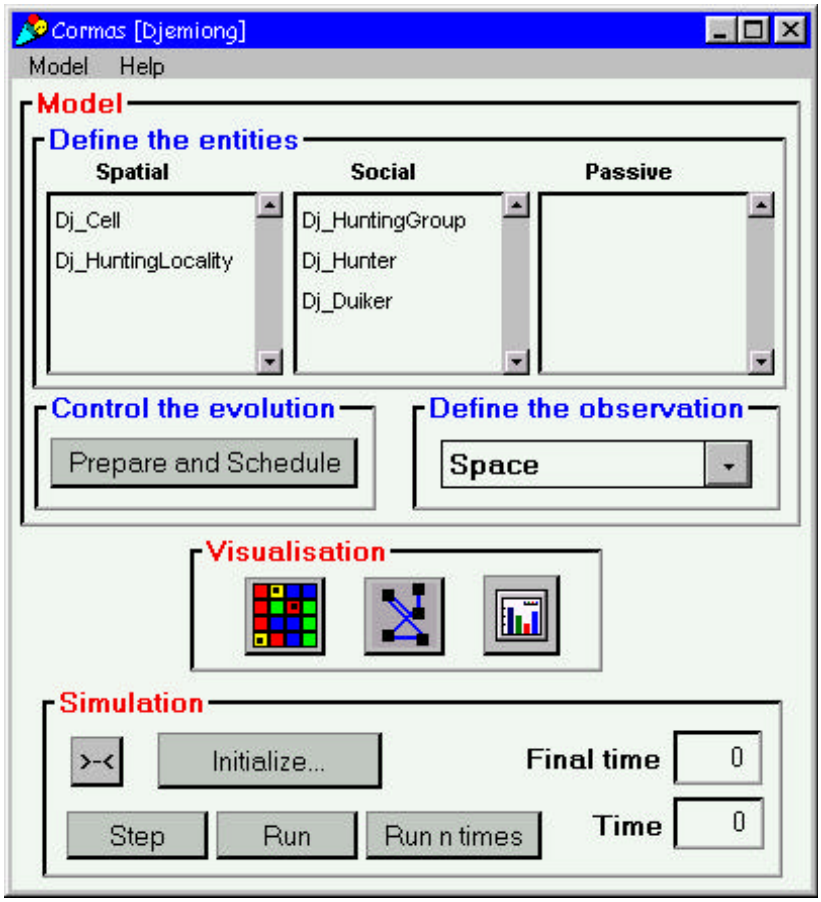


Figure 3. *The CORMAS main interface.*

The organisation of the modelling group box in the upper part suggests three successive steps. The first one consists in defining the entities of the model into three categories (spatial, social, passive).

Defining specific entities from the CORMAS generic entities

Objects are defined as computational entities that encapsulate some state, are able to perform actions, or methods on this state, and communicate by messages passing. While there are obvious similarities between objects and agents, there are also significant differences. The main one deals with the degree of autonomy. In the object-oriented case, the decision about whether to execute an action lies with the object that invokes the method. In the agent-oriented case, the decision lies with the agent that receives the request (Wooldridge, 1999).

Requesting actions to be performed has been set as the basic behaviour of the CORMAS root entity (see figure 4). It is particularly useful when dealing with the share of common-pool resources. Typically, an agent will send a request to a spatial entity for an amount of

natural resource. Two main rules have been implemented for sharing the resources, either to account for an asynchronous mode ("first asking, first served") or to account for a synchronous mode (each agent receives an amount proportional to its request). Any entity of Cormas has two attributes *request* and *qtyGiven*. When an entity wants to harvest a resource held by another entity, it uses the following message:

<receiver_entity>
receiveRequestAbout: <attribute>
qty: <q>
from: <sender_entity>

The sender (usually *self*) asks for a quantity of resource held by the receiver in an attribute. To use this programme the sender and the receiver must have the same attribute. The receiver will stock this kind of request in its attribute *request*. It is a collection of triplets (attribute, quantity, sender). Then, once all the requests have been done, the entity will share the resource. Different programs are available.

- *treatRequestOrder* gives the resource according to the order of reception. First request, first served.
- *treatRequestProrata* shares the resource between the senders depending on the quantity requested.

One of the models described later (fuelwood consumption) gives an example of this basic functionality of CORMAS.

The general hierarchical organisation of these categories is shown in figure 4, as the organisation of the CORMAS spatial entities is detailed in figure 5. For each of these CORMAS generic classes, a set of standard methods exists.

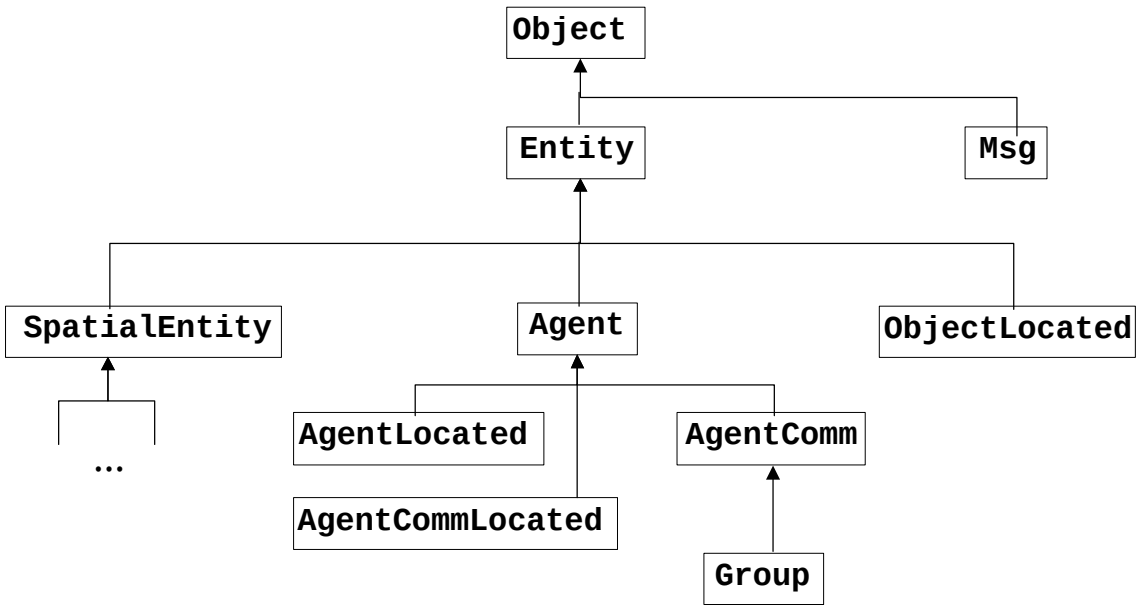


Figure 4. *Hierarchy of the CORMAS generic classes. The Object class is the Smalltalk root class*

Apart the spatial entities described in detail in the next section, the main CORMAS entities are *AgentComm* and *AgentLocated* (as Smalltalk does not allow multiple inheritance, the *AgentCommLocated* class is actually just a combination of *AgentComm* and *AgentLocated*).

AgentLocated

Methods have been implemented in this class to give the agents spatial references. The agent is situated on a spatial entity. Its *cell* attribute is a pointer to this spatial entity instance. Two main methods have been programmed to model movements.

1. The *leave* method breaks the link between an agent and its spatial reference. The attribute *cell* of the agent is set to nil, and the spatial entity removes the agent from its attribute *theOccupants*.
2. The *moveTo: aDestination* links the agent and the object *aDestination* which is a spatial entity. The *cell* attribute of the agent is set to *aDestination*, which adds the agent in its attribute *theOccupants*.

Usually the movement method of such an agent ends with these two instructions. The previous code aims at defining which spatial entity will be chosen. The randomWalk method gives a good example of a simple implementation:

randomWalk

```
| destination |
destination := Cormas selectRandomlyFrom: self cell neighbourhood.
self leave.
self moveTo: destination
```

AgentComm

An instance of *AgentComm* class or an inherited class possesses two main attributes *channel* and *mailBox*. These attributes are implemented to allow communication. A communicating agent is linked to a communication channel (the attribute *channel*). This channel is in charge of managing communications. The modeller does not have access to the channel. To implement communications the modeller just has to make agents read their *mailBoxes* and send *messages*. The basic steps are the following:

- Create an instance of the required message class defined from the generic Cormas *Msg* class, and assign values to the required attributes: *sender*, *receiver* and *symbol* (used to define the type of message).
- Send the message. Two methods are available depending on the scheduling of the model. To account for an asynchronous mode the message should be sent by invoking the *sendMessageAsynchronously:* method. It will be immediately delivered to the receiver's mailbox. In synchronous mode, the message should be sent by invoking the *sendMessageSynchronously:* method. It will then be delivered at the end of the time step, once all entities have sent their messages.
- Processing a message is simply a matter of reading the *mailBox*, which is defined as a collection of messages.

Even if the conception of the model should get an insight into which entities should be defined, there are always many different ways to create entities corresponding to a given question to address when building a model. For instance a single cell entity seems sufficient to build a cellular automata when one need to represent a diffusion process. But sometimes it may becomes useful -by making the implementation easier- to define attributes of the cell as entities themselves.

Controlling and scheduling a model with CORMAS

The second step consists in defining the control and the scheduling of the model. Here again, the way is has been implemented in CORMAS deals with another important distinction between objects and agents stressed by Wooldridge (1999) : agents are each considered to have their own thread of control, whereas in the standard object model, there is a single thread of control in the system. Yet this last solution has been adopted for CORMAS. The scheduling of the agents has to be specified in a "step" method which is automatically executed at each time-step of the model. This time-step being used to schedule all the entities of the model, it has to be defined as the shortest unit of time during which an entity of the model evolve. Any particular time feature (i.e. periodic events) has to be written as a part of a Smalltalk instruction (usually a test on the current time-step value). Lorek and Sonnenschein (1998) remark that this kind of implementation defines a discrete-time simulation approach, as on the other hand, a discrete-event simulation approach is characterised by a decentralised control threading. The main disadvantage of using a single thread of control in the system is to test at each time-step for all the entities of the model whether it has something to perform, which is not very efficient from an optimisation point of view. Nevertheless, such a way to handle the scheduling of the events has been chosen because of its simplicity. As the events are not queued, the user do not have to care about problems that may occur when an entity disappears from the system but other events bound to the same entity may later try to resume. A particular generic Cormas class (corresponding to the model itself) is automatically created when entities are defined during the first step of the modelling process, and collections of these entities are set as attributes of this class. In addition to the "step" method to be executed each time-step, an "initialisation" method specifying the number of entities to create and their initial state has also to be written.

Defining observer's viewpoints in a flexible ways with CORMAS

The last step allows the user to open some specific tools to define viewpoints on the model. Depending on the current state of the entities, a point of view may be defined. A state is the value of an attribute or the combination of several values. A point of view is a Smalltalk method to be implemented by the observer. This insures a flexibility by allowing to define any combination of values taken by the attributes of the entity. During the simulation, the current visual state of an entity is updated according to the corresponding active point of view method. The selection of point of view methods is available directly from the contextual menu of the spatial grid interface. For spatial entities, the visual state refers to a colour, as for located entities, it refers to a shape, a colour and a size.

Building hierarchical artificial landscapes with CORMAS

In this section, we present first in details the CORMAS spatial entities and then the associated tools allowing to define an artificial landscape as the environment of the agents.

The CORMAS generic spatial entities.

The construction of a spatial support with CORMAS goes throw the definition of an elementary spatial entity, which will represent the smallest homogeneous portion of the environment in the model. The identification of the appropriate spatial resolution is one of the key-feature of the modelling process. It is directly correlated to the objective of the model. The hierarchy of spatial entities proposed by Cormas (see figure 5) is based on a

general abstract root class, named *SpatialEntity*, from which the elementary spatial entity (*Cell*) directly inherits.

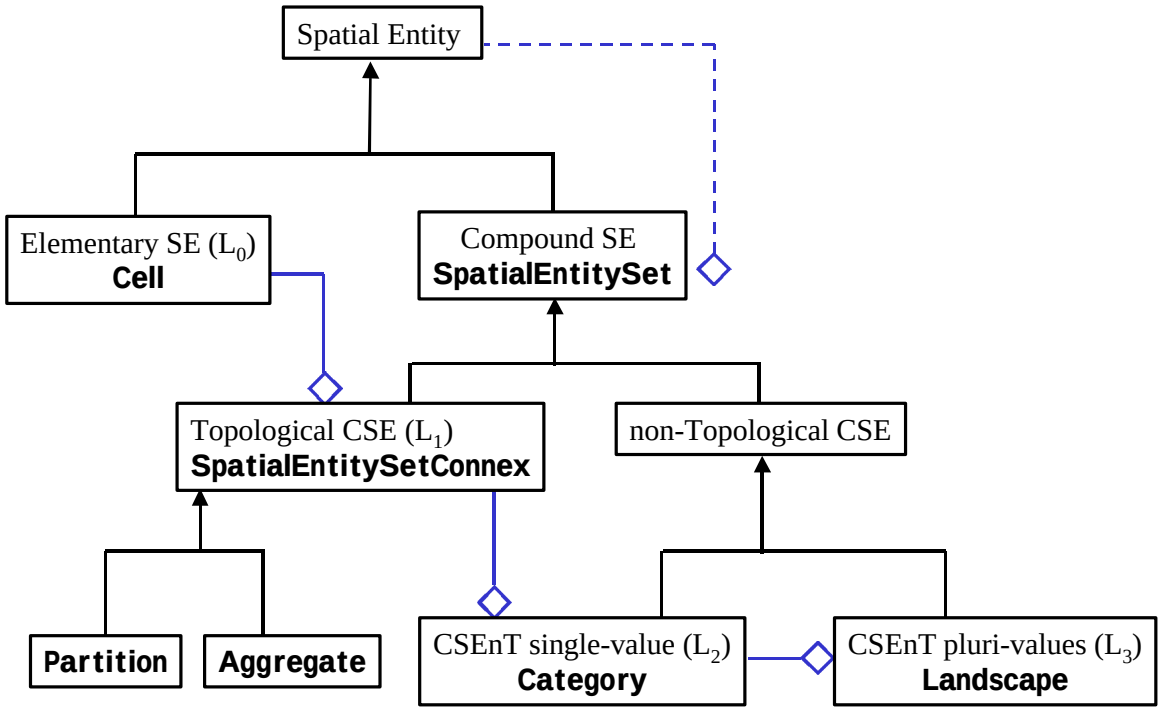


***Figure 5**. Hierarchy of the CORMAS generic spatial entities. Diamonds are used to indicated composition relationships.*

At the abstract level of *SpatialEntity*, some attributes and methods shared by every kind of Cormas spatial entities have been defined. For instance, *theOccupants* is a register of all the inhabiting located entities. Thus any located entities may directly requests the spatial entity it is occupying via its *cell* attribute while in a symmetrical way any CORMAS spatial entities registers directly all its inhabiting agents. Even if this redundancy should alter the performance of the simulation framework, it facilitates the writing of Smalltalk methods.

Some other attributes are related to the topology. From this category, the most important is *neighbourhood,* defined as a collection of adjacent instances of the same class. According to the topological properties of the virtual landscape, and depending whether the spatial entity shape is chosen regular or not (see next section), CORMAS provides methods to automatically update the neighbourhood. Additionally, the *extensiveNeighbourhood* attribute may be used to store a larger collection. This is mainly used to configure specific perception ranges of agents. The usual way to define an extensive neighbourhood is to recursively look for neighbours of the adjacent neighbours. There is a method (*recursiveNeighbourhood: n*) which computes recursively this extensive neighbourhood (within a area of radius *n*) and returns it, as illustrated in figure 6.

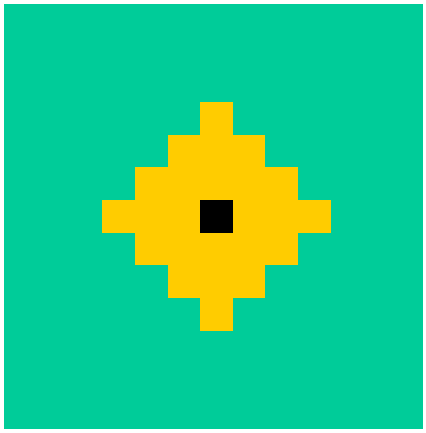


***Figure 6**. An example of extensive neighbourhood of radius 3 for a 4-connex cell.*

The hierarchical organisation of the Cormas generic spatial entities (see figure 5) is based on the *theCSE* attribute that registers the belonging to a higher level spatial entities. Basically, each class of Cormas spatial entity is susceptible to be defined as a spatial component any other Cormas compound spatial entity. All the classes inheriting from *SpatialEntitySet* have a *components* attribute. When a compound spatial entity is defined or evolves, its *components* attribute is updated by using one of the methods *addComponent:*, *addComponents:*, *deleteComponent:* or *deleteComponents:*. Within these methods, each element of the components that has been added or deleted updates its attribute *theCSE*. A method implementing exchange of components between compound spatial entities is based on these primitives:

```
receiveComponents: aCollec from: anotherCompoundSE
        anotherCompoundSE deleteComponents: aCollec.
        self addComponents: aCollec
```

By using these generic method, the updating of the both-ways referencing mechanism based on the couple of attributes *components*/*theCSE* (similar to the one linking the *cell* and *theOccupants* attributes, as mentioned earlier) is transparent to the user.

The spatial entities inheriting from the abstract class *SpatialEntitySetConnex* are built by aggregation of lower level contiguous spatial entities (see figure 5). The spatial environment of CORMAS (see next section) provides generic methods that perform this kind of aggregation. Thus for the construction of *Partition* spatial entities, the following three methods are available:

- **partitions: <**entityC> **madeOf: <**entityE> **attribute: <**attributeName>
  where <entityC> and <entityE> are respectively the higher and the lower levels spatial entities, and <attributeName> is the name of the attribute of the lower level spatial entity on which the aggregation is processed. The result will give as many distinct instances of Partitions as there exists distinct values for this attribute in the whole collection of lower level spatial entity.
- **partitions: <**entityC> **fromSeeds: <**aCollec>
  where <entityC> is the higher level spatial entity, and <aCollec> is a collection of instances of the lower level spatial entity. Starting from these kind of seeds, the aggregation is processed by recursively looking for not yet aggregated items the outside surround of the newly created instances of the *Partition* class.

Figures 7 (a) (b) and (c) illustrate this kind of progressive extension by diffusion from an initial random distribution of 10 seeds in a 30*30 spatial grid (a), after 2 recursions of the algorithm (b), and when the process is completed (c).

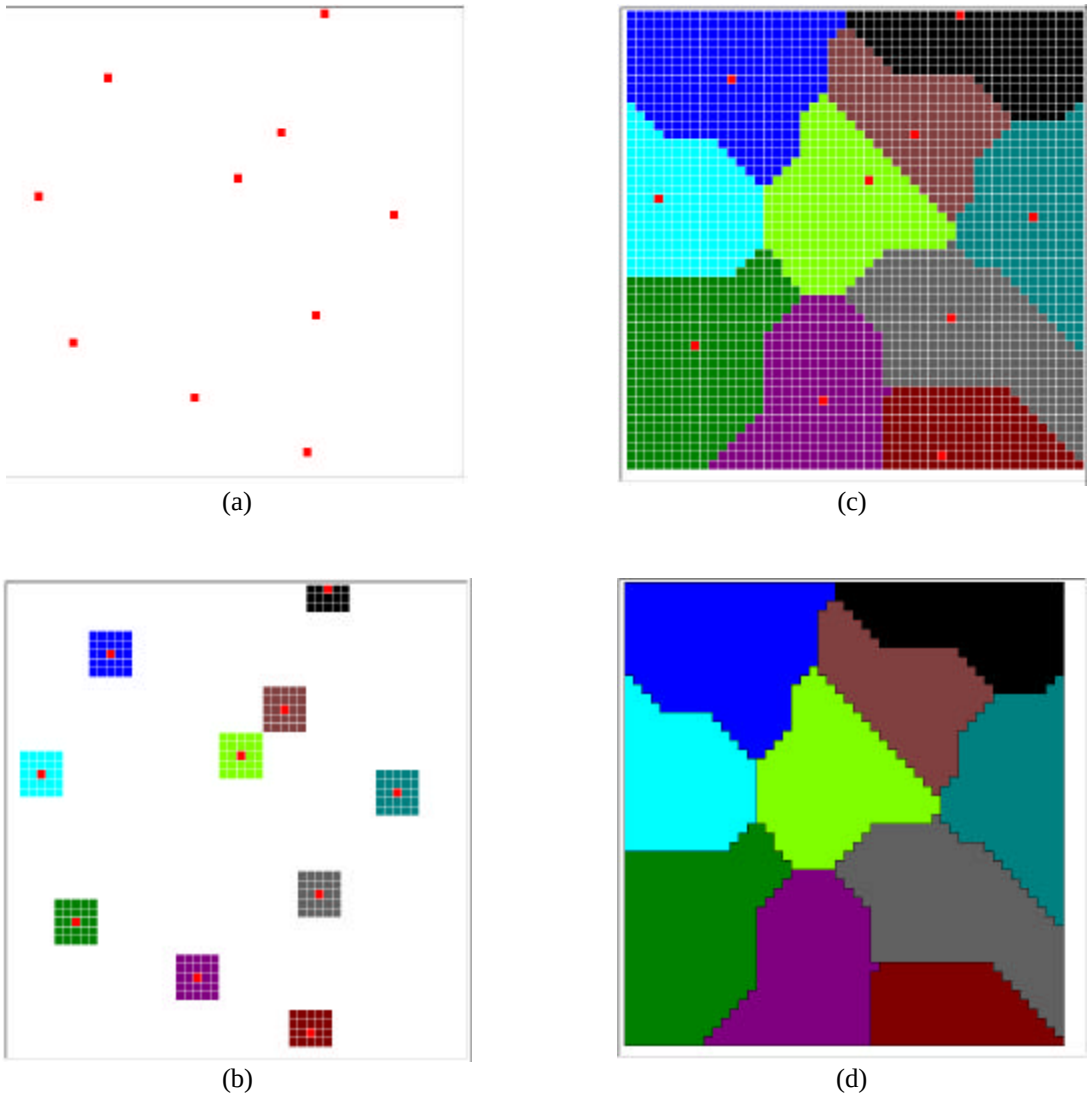


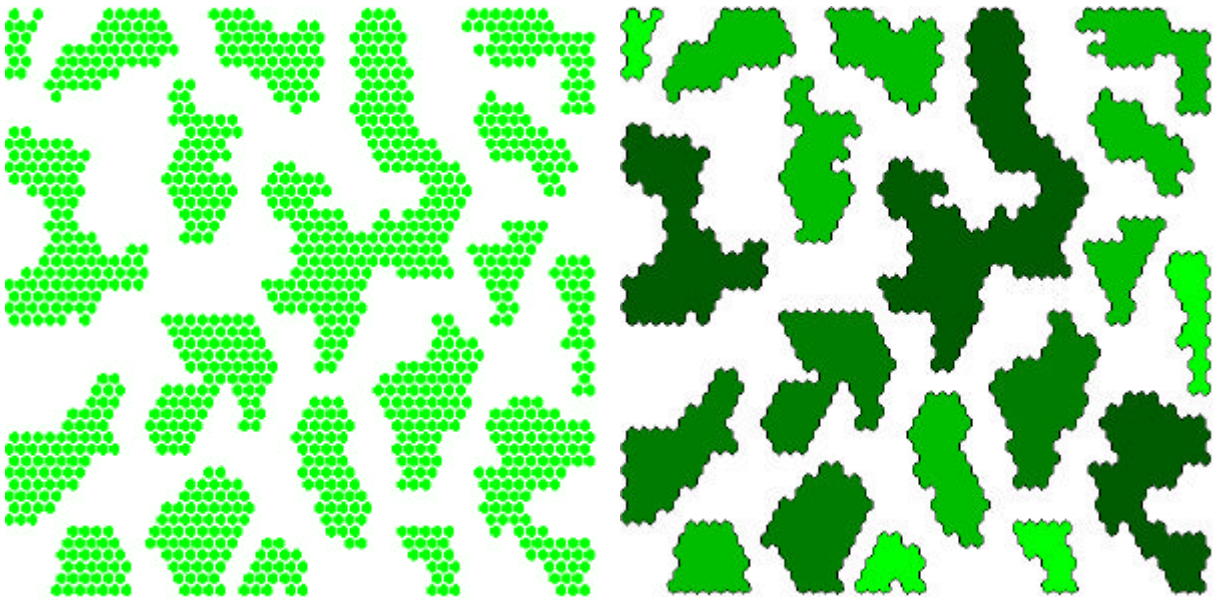
***Figure 7**. An example of the generic method of aggregation by diffusion from seeds.*

Another point has to be noted from figure 7(d): the definition of points of view is available for all the CORMAS spatial entities of a model. The polygonal image of a compound spatial entity is automatically adjusted from the outlines of the lower level spatial entities belonging to its inside surround, and the colour used to display the compound entities are based on the visual state specified in the active point of view method, in the same way than all the other CORMAS entities. In the example showed in figure 7(d), the colour has been randomly assigned when the *Partition* instances were created.

Aggregate is the other compound and connex spatial entity proposed by CORMAS (figure 5). It is mainly used in situations where the aggregation is conditioned by the state of the lower level spatial entities, this state eventually fluctuating dynamically during the simulation. The generic method provided by the spatial environment of CORMAS stands as follow:

- **aggregates:** <entityC> **madeOf:** <entityE> **condition:** <methodName> **minSize:** <x>
  where <entityC> and <entityE> are respectively the higher and the lower levels spatial entities, <methodName> is the name of the condition for any instance of the lower level spatial entity to be aggregated, and <x> is the minimum size to confirm the creation of the instances of the higher level spatial entity.

Figure 8 illustrates this method in a 30*30 spatial grid made of hexagonal cells with a very simple state attribute being either #empty (white) or #forest (gray) (a).



(a) (b)

***Figure 8**. An example of aggregation conditioned by the state of the lower level spatial entities.*

The result of the aggregation of the #forest cells is materialised in figure 8(b) with the viewpoint on the compound spatial entities based on their size (the darker, the bigger).

The spatial environment of CORMAS

By default, a regular spatial grid made of 10x10 cells is opening when clicking on the left icon of the visualisation group box in the CORMAS main interface (see figure 3). The configuration of the spatial grid can be done through its menu.

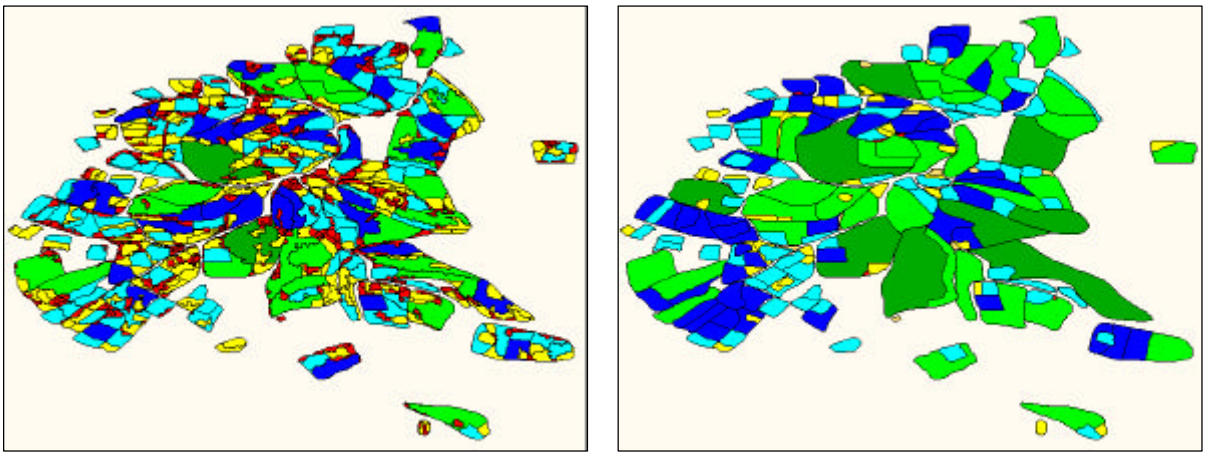
Tesselation

Cormas can represent the space as regular grids or irregular tesselation. Regular grids will be created automatically. Irregular grids will require special programmes to generate or load polygons. Depending whether the “Regular” or “Irregular” option is selected within this submenu, the main menu is adjusted.

Topology

In case of irregular tesselation two options are proposed. One for the generation of the elementary spatial entities using the Voronoi algorithm, and another one to load polygons

exported as text files from Geographical Information System (GIS) softwares like MapInfo or ArcView. A MAS model to assess soil erosion risk in north Thailand (Trébuil, pers. comm.), illustrates this functionality. The result of the transfer of GIS data files of the watershed into the CORMAS environment is shown in figure 9.



(a) (b)

Figure 9*. An example of hierarchical structure based on polygonal spatial entities imported from a GIS.*

Actual maps and key layers of information, such as spatial constraints like slope angles and lengths, have been used, and two spatial entities are considered in the model: homogeneous zones (with homogeneous slopes) and farmers' fields, and are interconnected in the sense that the higher level spatial entity (the field) is compound of homogeneous zones. Their attributes are: *area*, *owner*, *slope* characteristics (orientation, angle, length). Both active point of view methods of the embedded spatial entities are based on range values of their *area* attribute.

In case of regular grids the following classical parameters may be adjusted.

- The size of the grid (number of lines and number of columns).
- The grid boundaries. They are either torroidal, which implies that all the cells of the grid have the same number of neighbouring cells, even those located at the edge of the grid. Phenomena that reach the edge of the grid can continue on the opposite side of the grid. One can visualise this type of space as an inner tube. The other solution (closed) is to consider that cells at the edge of the grid have a number of neighbouring cells limited by borders.
- The cell shape, either rectangular or hexagonal. The hexagonal cell has six neighbours. The connexity of rectangular cells has to be chosen between 4 and 8.

For any combinations of these three parameters, CORMAS automatically computes the neighbourhood of all the cells.

Tools

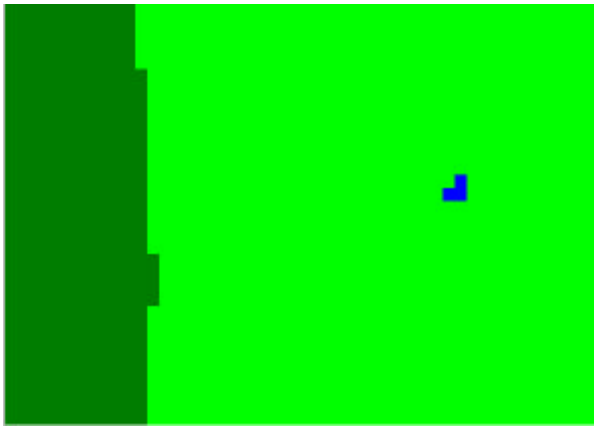
At any time, CORMAS allows the user to change a cell attribute in a easy way. One has first to select an attribute from the Tools submenu, then to enter a new value in the dialog box appearing, and finally to assign this new value to a particular cell with a right-click on it. Additionally, it is possible to save and to load a spatial grid as a text file. When saving a grid, the user is requested about the attributes of the cells to be saved.

Some examples of CORMAS models based on spatial entities

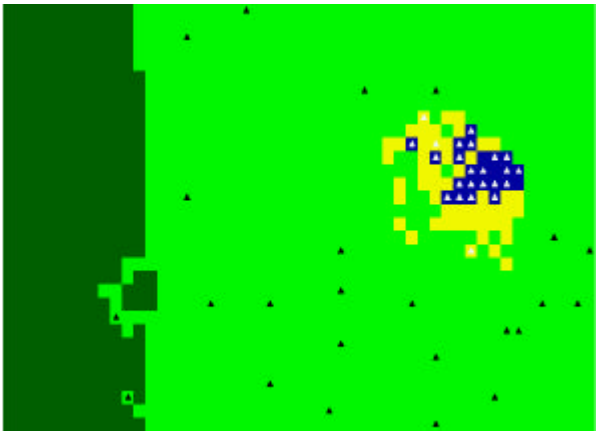
Before presenting examples of models with a hierarchical structure made of CORMAS spatial entities, a first point is addressed here, dealing with the degree of autonomy that may conceptually be assigned to the spatial entities of a model.

Spatial entities as holders of the landscape dynamics.

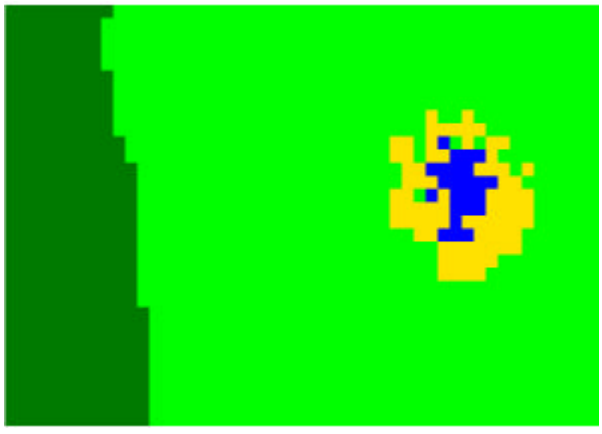
When using MAS to simulate land-use dynamics, the usual way is to define physical agents and their actions on the environment. Another conceptual framework consists in defining geographical entities endowed with specific behaviour. A methodological comparison between these two conceptual approaches have been discussed from a theoretical example by Bousquet et al. (1999). The initial spatial environment (figure 10a) is made of a forest (darker left side of the picture), a savannah (lighter, right side), and a L-shaped village made of 3 cells. All the cells have also a boolean *fertility* attribute.



(a)



(b)



(c)

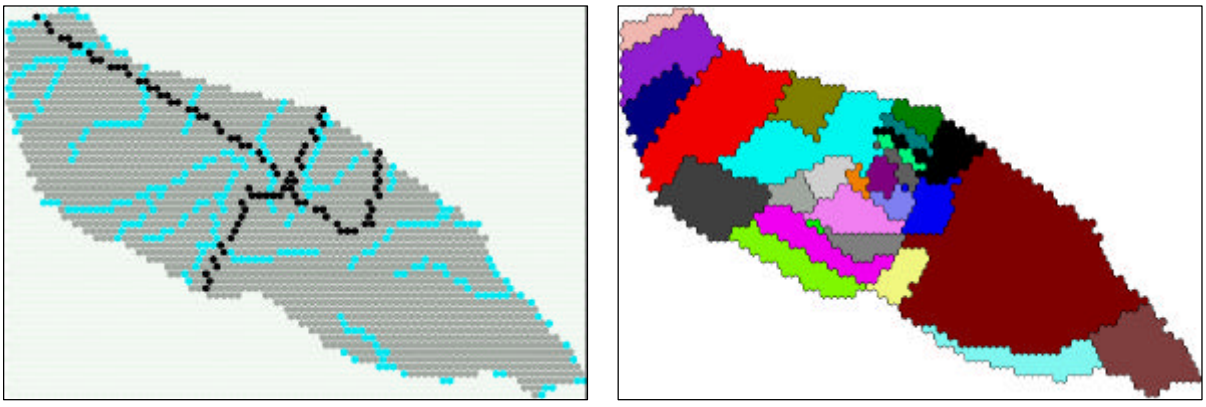
Figure 10*. Land use dynamics in a savannah artificial landscape (a) under the effect of increase of population. (b) corresponds to an implementation via agents representing the actors (farmers – represented by white triangles- and cattles –represented by black triangles) that directly modify the environment, as (c) corresponds to an implementation via exchanges of components (cells) between the higher level spatial entities.*

With the viewpoint of the actors managing the natural resources, the dynamics is implemented by farmer/herders who set up villages in fertile zones and clear fields around the village. At each time-step, a farmer agent (defined with a perception radius of 3 cells) are looking for fertile and still under savannah cells to convert them into a field and add them to the existing agricultural area. When a farmer agent has found three fields, it installs a member of its family close to its dwelling. In doing so, a field cell is transformed into a village cell. In parallel with this process of agricultural development, livestock graze in the savannah around the fields. Their number increase proportionally to the number of farmer agents. At each time step, the cattle agents (defined with a perception radius of 1 cell) are randomly moving to a neighbouring cell being neither a village cell nor a field cell. When they arrive in a forest cell, however, they transform this cell in a savannah cell. An intermediate (20 time-steps) result is given in figure 10b.

With an exclusively spatial viewpoint, the agents are no longer farmers or herds, but geographical entities (savannah, forest, field, village) being defined as CORMAS compound spatial entities made of cells. They are using the exchange of components protocol described earlier. An intermediate result is shown in figure 10c. With this second approach, space is integrated at the overall level of the geographical entity and no longer at the cellular level.

Spatial entities as appropriate level of knowledge and information

The model described here (Bousquet et al., in press) is based on a study of blue duiker hunting (a small antelope) in Djemiong, a forest village in eastern Cameroon. The aim of the study is to understand how the organisation of the hunting activity between villagers constitutes a management system. The model was built based on the life history of the blue-duiker and on the inhabitants' hunting behaviour and incorporates raster data from a GIS. Once blue duikers have reached maturity and found a mate, they demarcate their territory (about 3 ha) where they remain until they die. Each cell on the spatial grid (figure 11a) represents an area of 3 ha. The GIS is divided into three layers: roads, rivers and hunting localities. A file is created for each layer to provide information on each cell. Cormas then imports this data. The cell has an attribute for each layer: water (yes/no), road (yes/no) (cf. figure 11a) and the hunting localities (cf. figure 11b).



(a) (b)

Figure 11*. The Djemiong artificial landscape. (a) each light grey cell represents an area of 3 ha, which is the size of the adult duiker's territory. Cells with water are dark grey, cells with a road are black. (b) the 27 hunting localities in Djemiong.*

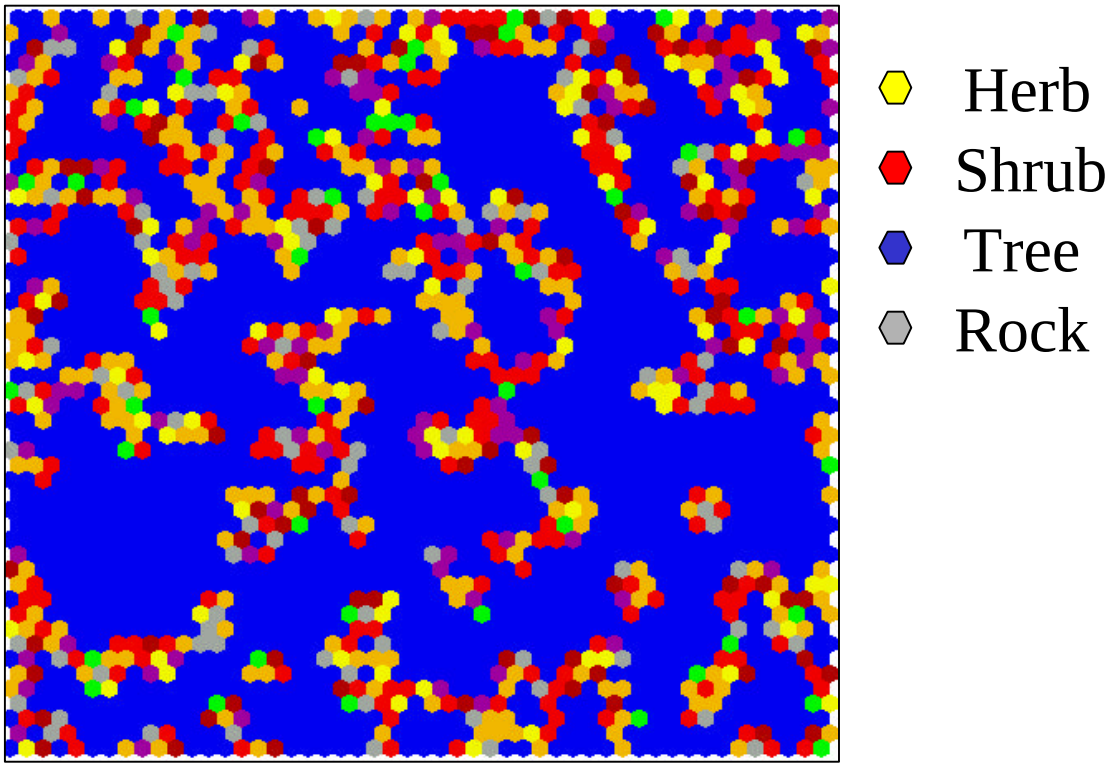
These two levels of spatial entities have been used to define the behaviour of the agents of the model. The perception range of a duiker agent is defined as a 3-order recursive function based on the 4-connex neighbourhood of the cell where it is located (see figure 6). In a weekly time step, a duiker agent can visit any of the 25 cells that make up this area. When a single adult male meets a single mature female, they look for a suitable cell within their common perception range where they can settle. The suitability is defined as follows: the new site should be empty (no other duiker agents present) and have neither water nor road.

On the other hand, the hunting localities have been determined from surveys on the field. Their limits have been defined on a map in consultation with the inhabitants. Twenty-nine hunting localities were identified from the spatial information collected in the survey (see figure 11b). During the hunting season, each hunter sets traps along a path in the forest (trap network).

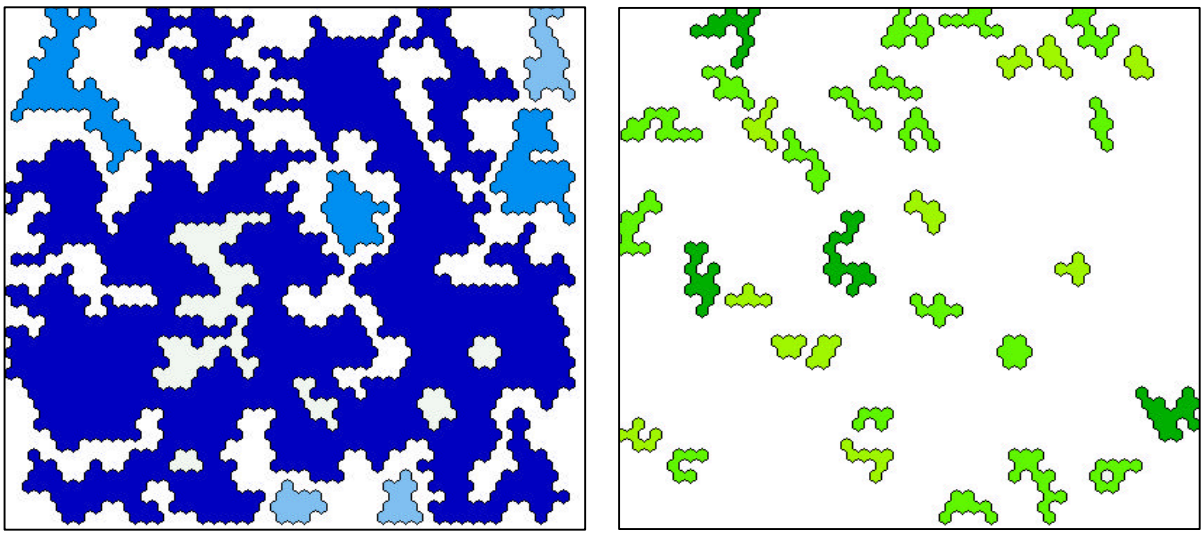
Because the information obtained from the survey is not on the same scale as that used for the hunting process in the model, which was determined by ethological processes, we had to make assumptions about the precise locations of traps paths within a given hunting locality. Several simulation scenarios have been defined on this basis and then compared by running the model (Bousquet et al., in press).

Spatial entities as materialisation of the points of view of agents on their environment

In this model of sylvo-pastoralism in Mediterranean forests still under construction, two agents use resources from the same environment: a shepherd agent, looking for good places to graze its cattle, and a forester agent, more concerned by the quality and quantity of the forests. The CORMAS spatial environment is defined from a conceptualisation of the builder of the model (Etienne, pers. comm.). Three layers of vegetation are taken into account, herb, shrub and tree. All these three layers may be present in the same time on any cell. They are represented by colours (respectively yellow, red and blue), so that simultaneous presence of more than one layer is represented by a colour corresponding to the mixing of the basic colours (see figure 12a).



(a)



(b) (c)

***Figure 12**. Three points of view on a Mediterranean forest. The first one (a), defined at the cell level, represents a conceptualisation of the modeller, whereas (b) and (c) are using compound spatial entities to represent points of view of the agents of the model -a forester and a shepherd- on this shared environment.*

The natural dynamics of the model states that cells with herb are covered with shrub after 3 years, unless being grazed. Additionally, fire events may occur from time to time. The spreading of fire is growing on contact with shrub. The way the shepherd and forester agent will act will have consequences on the landscape structure. The shepherd agent acts with a monthly time-step, as the forester makes decisions once a year (every 12 time-steps).

The two possible management actions of the forester are to clear of undergrowth (to remove shrub) or to sow a cell with herb. Each action costs an amount of money. Three criteria have been defined to estimate the quality of the management scheme: the limitation of the spatial extension of a fire-event, the preservation of a biodiversity index (based on the distribution of the cells colours), and the relative proportion a forest-only (blue) cells. If at least 2 of these 3 criteria are evolving in the right direction, the forester agent receive subsidies. Given its balance of money, the forester agent is only able to undertake a limited number of actions. Specific aggregates of cells based on the criteria may be useful when the forester has to choose the precise cells on which these actions should be undertaken as a priority (see figure 12b), for instance to locate a firebreak around its most valuable forest.

In a similar way, a point of view specific to the shepherd may be built by aggregating cells with herb. This kind of higher level spatial entities (see figure 12c) represent pasture areas to be grazed during a month. Because there is a travel cost from one pasture area to another, the shepherd agent may use these entities to compute interesting yearly pasture route. One of the interest of such a model is then to define communication between the shepherd and the forester agents. For instance the forester may request the shepherd for grazing a particular area. Then the agents have to make the balance between their individual goals and the common interest (Etienne, pers. comm.).

Spatial entities holding and sharing the resources

The aim of the model (Bousquet et al., 2000) described here is to understand the interaction between fuelwood consumption and landscape dynamics. The hypothesis put forward suggests that fuelwood consumption can explain the landscape changes that occur in the Kayanza region of Burundi. A preliminary map was outlined (see figure 13a); agents use fuelwood, have access to different parts of space and have the capacity to exchange.

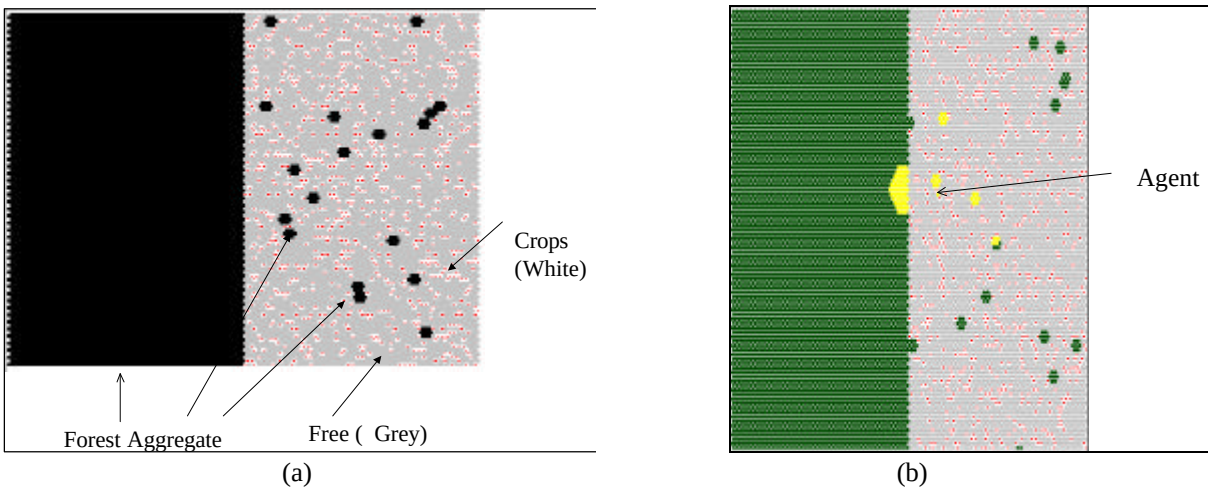


Figure 13*. (a) initial situation (b) visualisation of the perception range of a particular agent.*

The population increases and agent migration—from overpopulated areas to unoccupied plots—is simulated. The impact of changing rules on foraging, exchange and access is then observed on a landscape level. The household agent has specific behaviour patterns for

harvesting wood. The agents calculate how much live wood (consumption of live wood) or dead wood (consumption of dead wood) they need. They then ask the cells in their perception range (wood space) for this amount of wood (see figure 13b). The local consumption corresponds to the amount of wood consumed by agents on an agroforestry cell. The quantity of consumable wood on the cell is the quantity of surplus live wood compared to the maximum production level (75 m^3). In this way, the sustainable management of the agroforestry cell is simulated.

In this model, the aggregate forest is made up of sets of contiguous cells that are in a forest or degraded forest state. The aggregate is capable of defining its border and determining sub-units of cells that are at a certain distance from a given point. Thus, an agent situated at a certain distance from the aggregate forest can ask the latter which cells it can collect wood from. In this way, the aggregate forest controls the distribution of wood and determines the order in which the cells produce wood by using the basic method common to all the CORMAS entities described in section 2.1.1. If the case arises in the model, the most degraded cells (with the least live wood) produce wood first. Thus, the model represents the fact that habits develop and fuelwood harvesting is concentrated in certain parts of the forest.

Conclusion

Rather to raise the question of scale transfer, the methodology that we propose beyond the use of the CORMAS simulation framework enables us to focus on relationships among dynamic processes at several levels. The CORMAS spatial structure may prove useful in trying to slice the “layer cake” of ecological systems diagonally, as Allen et al. (1987) have stressed to be one of the most interesting point to focus on in ecological modelling.

References

Allen, T. F. H., O'Neill, R. V., Hoekstra, T. W., 1987. Interlevel relations in ecological research and management: some working principles from hierarchy theory. Journal of Applied Systems Analysis 14, 63-79.

Barreteau, O., Bousquet, F., in press. SHADOC: a Multi-Agent Model to tackle viability of irrigated systems. Annals of Operations Research.

Baveco, H., Lingeman, 1992. An object-oriented tool for individual-oriented simulation: host-parasitoid system application. Ecological Modelling 61, 267-286.

Bousquet, F., Bakam, I., Proton, H., Le Page, C., 1998. Cormas : Common-Pool Resources and Multi-Agent Systems. Lecture Notes in Artificial Intelligence 1416, 826-837.

Bousquet, F., Barreteau, O., Le Page, C., Mullon, C., Weber, J., 1999a. An environmental modelling approach. The use of multi-agents simulations. In: F. Blasco , A. Weill (Eds.), Adances in Environmental and Ecological Modelling, Elsevier, Paris, pp. 113-122.

Bousquet, F., D'Aquino, P., Rouchier, J., Requier-Desjardins, M., Bah, A., Canal, R., Le Page, C. (1999b). Rangeland herd and and herder mobility in dry intertropical zones: multi-agent systems and adaptation. Paper presented at the VI International Rangeland Congress, Townsville, Australy.

Bousquet, F., Gautier, D., Le Page, C. (1999c). Resource management and scale transfer: the contribution of multiagent systems. Paper presented at the Scaling Methodologies in

Ecoregional Approaches for Natural Resources Management, Ho Chi Minh City, Vietnam.

Bousquet, F., Le Page, C., Antona, M., Guizol, P. (2000a). Ecological scales and social exchanges: the use of multi-agent systems. Paper presented at the IUFRO World Congress, Kuala Lumpur, Malaysia.

Bousquet, F., LePage, C., Bakam, I., Takforyan, A., 2000b. Multiagent simulations of hunting wild meat in a village in eastern Cameroon. Ecological modelling In press.

Epstein, J., Axtell, R., 1996. Growing Artificial Societies. Social Science from the Bottom Up. Brookins Institution Press/ The MIT Press, .

Ferber, J., 1999. Multi-Agent Systems : an Introduction to Distributed Artificial Intelligence. Addison-Wesley, Reading, MA, 509 pp.

Grimm, V., 1999. Ten years of individual-based modelling in ecology: what have we learned and what could we learn in the future ? Ecological Modelling 115, 129-148.

Hraber, P. H., Jones, T., Forrest, S., 1997. The Ecology of Echo. Artificial Life 3, 165-190.

Kareiva, P., Wennergren, U., 1995. Connecting landscape patterns to ecosystem and population processes. Nature 373, 299-302.

Kawata, M., Toquenaga Y., 1994. From artificial individuals to global patterns. TREE 9.

Lhotka, L., 1994. Implementation of individual-oriented models in aquatic ecology. Ecological Modelling 74, 47-62.

Lima, S. L., Zollner, P. A., 1996. Towards a behavioral ecology of ecological landscapes. TREE 11 (3), 131-135.

Liu, J., Ashton, P. S., 1998. FORMOSAIC: an individual-based spatially explicit model for simulating forest dynamics in landscape mosaics. Ecological Modelling 106, 177-200.

Lomnicki, A., 1999. Individual-based models and the individual-based approach to population ecology. Ecological modelling 115, 191-198.

Lorek, H., Sonnenschein, M., 1998. Object-oriented support for modelling and simulation of individual-oriented ecological models. Ecological Modelling 108, 77-96.

Lorek, H., Sonnenschein, M., 1999. Modelling and simulation software to support individual-based ecological modelling. Ecological Modelling 115, 199-216.

Minar, N., Burkhart, R., Langton, C., Askenazi, M. (1996). *The swarm simulation system : a toolkit for building multi-agent simulations*. Available: http://www.santafe.ed/project/swarm.

Resnick, M., 1996. Beyond the Centralized Mindset. Journal of the Learning Sciences 5 (1), 1-22.

Rouchier, J., Bousquet, F., Barreteau, O., Le Page, C., Bonnefoy, J.-L. (2000). Multi-Agent modelling and renewable resources issues: the relevance of shared representations for interacting agents. Paper presented at the Multi-Agent Systems and Agent-Based Simulation, Boston.

Weiss, G. (Ed.), 1999. Multiagent Systems : a Modern Approach to Distributed Artificial Intelligence. MIT Press, 619 pp.

Wooldridge, M., 1999. Intelligent Agents. In: G. Weiss (Ed.), Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence, The MIT Press, Cambridge, Massachusetts, pp. 27-78.

Wu, J., Levin, S. A., 1997. A patch-based spatial modeling approach: conceptual framework and simulation scheme. Ecological Modelling 101, 325-346.

Ziv, Y., 1998. The effect of habitat heterogeneity on species diversity patterns: a community-level approach using an object-orientedn landscape simulation model (SHALOM). Ecological Modelling 111, 135-170.