ELSEVIER

Ecological Modelling 88 (1996) 113–124

ECOLOGICAL MODELLING

The representation of space in an object-oriented computational pelagic ecosystem

Ph. Laval *

URA 716 du C.N.R.S., Station Zoologique, B.P. 28, 06230 Villefranche-sur-Mer, France

Received 3 August 1994; accepted 3 February 1995

Abstract

The HOOD (Hierarchical Object-Oriented Design) methodology, established for software engineering purposes, may be applied to decompose an ecosystem into a hierarchy of entities. Self-reproducing artificial organisms (software entities "living" in the computer memory) may populate the system, allowing to simulate the development of a planktonic population. The artificial creatures exist and move in asynchronous parallelism. Feeding, which was absent in the previous model, permits a regulation of the population growth. Space is an essential reference to be taken into account, because food and other resources are only found at certain locations. Space in turn is also a resource, which should be managed in the software. In the HOOD design, the space object is included in a larger conceptual object, the Environment object. This higher-level object contains a "control object" managing the concurrent access to the resources, as well as the resources themselves. The introduction of spatialized food resources, consumed by the population of artificial organisms, allows a more realistic simulation of the colonization of space by "blooming" pelagic organisms.

Keywords: Artificial life; Marine ecosystems; Object-oriented models; Zooplankton

1. Introduction

In marine pelagic ecosystems, salps (Tunicata: Thaliacea) constitute a group of filter-feeders which, at certain periods, may be predominant. Their importance has been recently reviewed by Fortier et al. (1994). These authors stress the role of salps in the lengthening of carbon turnover time, and make the hypothesis that "the capacity of these large microphages to swarm explosively allows them to control phytoplankton blooms". They also remark that "the actual biotic or abiotic conditions triggering thaliacean swarms are poorly understood". It is therefore useful to try to model the explosion of salp populations.

Equations representing salp growth have already been introduced in classical modelling (i.e. based on differential equations) of pelagic ecosystems (Andersen and Nival, 1986; Braconnot et al., 1988). However, these kinds of models cannot take into account the complexity of the multiple interactions occurring concurrently in ecosystems. In a previous article (Laval, 1995a), it was shown how a software engineering method, HOOD (Hierarchical Object-Oriented Design), could be

* Fax: (+33) 93-763834.

used to construct the framework of an object-oriented ecological simulation, with artificial organisms as lower-level objects. These organisms were simulated Tunicates, represented with self-reproductive concurrent software units, "living" in the computer memory. To simulate the complicated mode of reproduction of salps, a population issued from one initial individual by alternation of sexual and asexual reproduction grew explosively in the computer memory. A Space_Management object provided a referential for the organisms and algorithms for their movements. Objects (clock, counters) needed by the modeller to launch and stop the simulation, and to observe what happened, were included in the design. All objects were arranged in a hierarchical framework representing a simulation of a minimal ecosystem.

This minimal design was sufficient to demonstrate how the HOOD method could be applied to an ecological model. One obvious shortcoming of the corresponding version of the software (CALIFE version 3.40) was that the artificial population, developing without constraint at an exponential rate, invaded eventually all the available space. Like any natural population, the artificial organisms have to have growth limitations if they are used in an ecosystem simulation. Food and other resources should also be introduced in the design, and these "objects" must be seamlessly integrated in the existing framework. This paper explains how space and resources may be inserted in the CALIFE hierarchical object model.

2. What is space, from an object-oriented modelling point of view?

In an object-oriented design, each object is an entity which has a state, a behaviour, and an identity (Booch, 1987, 1991). An object may be a real or an abstract entity. The HOOD method (Delatte et al., 1993) hierarchically decomposes a system in less and less abstract objects, starting from a "root object" which is an abstraction of the entire system. At the bottom of the hierarchy most objects represent real-world entities. This hierarchical decomposition is appealing to ecologists, who are well aware of the hierarchical organization of ecosystems (see for example Odum, 1988; Costanza and Hannon, 1989; Kirsta, 1994).

The word space conveys two meanings: one is space as a geometric referential; the other is room for something. From an object-oriented programming point of view, in the first meaning space is not an object: it is an attribute, a potential property of some objects (Booch, 1991, p. 77). However, in the meaning of "room available for the movements of organisms", it is conceivable to introduce a Space object. This object would contain a representation of the available positions in the system. The occupancy of these positions constitutes the object internal state.

In a computer simulation the Space object should also be mapped to a display device, which is an unavoidable object required by the simulation, in the same way as observation requires an observer. To visualize the course of the simulation, each occupied space position corresponds to the display of a code at a computer screen position.

In the CALIFE software, space is bi-dimensional. The extension to 3 dimensions is straightforward but computationally intensive, with large memory requirements. For the present state of development, a bi-dimensional model is already useful.

3. Mutual exclusion

One originality of the CALIFE software is that organisms behave in parallel: each individual contains its own set of states, driven by a local "internal clock". If two organisms try to update simultaneously the same resource, one of them should have exclusive access to the resource during the update, otherwise the amount of the resource would be incorrectly assessed by the second organism. This is the well-known "mutual exclusion" problem found in any concurrent software (see for example Booch, 1987, p. 190).

From the point of view of concurrency, space should be accessed in mutual exclusion because two organisms cannot occupy simultaneously the same position. However, two organisms occupying close but distinct positions may ask for the value of some physical characteristic at the same destination point; in this case mutual exclusion is not needed. However, if both "decide" to move to this place, mutual exclusion is needed, and only one of them will get the place.

4. Space as a resource, and a resource of resources

In our first approach (Laval, 1995a), space was viewed mainly as a geometric referential. The introduction of spatially distributed food leads to a deeper examination of the kind of things which are spatially distributed in an ecosystem. From an organism's point of view, three categories may be distinguished.

4.1. Physical characteristics

Temperature and salinity are obviously spatially distributed. These characteristics cannot be changed by the organism (at least at a granularity much larger than the organism size); their level remains the same with or without the presence of the organism. In computer jargon, they are read-only resources: at a given position, an organism can only inquire as to their level. This level may be different over space, but also over time. It is easy to see how physical characteristics may be dealt with in an object-oriented model. An organism in any point of the space–time continuum may use a function of the Space object returning the needed value. This function could be based on a previously established model, or a database of observations.

Physical characteristics in pelagic ecosystems are vertically oriented: "Oceans and lakes are anisotropic environments. Their ecosystems are organized along the axis defined by light and gravity." (Margalef, 1979). Light is not a parameter in the current version of CALIFE, but if necessary it could be introduced in the same way as other physical characteristics. Gravity is important when weight is dominant over viscosity, and when turbulence has no appreciable upward component. In the CALIFE Tunicate simulation, gravity accounted for the sinking of cadavers.

In the meaning of "room available", space itself is a physical characteristic, with a particular property: once occupied a position is unavailable for other organisms. All the resources present at that position are locked for the other organisms. Except where room is a scarce resource (e.g. benthic environment), there is properly speaking no competition for space itself; the competition is for the control of resources present at points of space.

4.2. Exploitable resources

If a resource is exploitable by an organism, at a given position the level of the resource may decrease during the residence time of the organism. Exploitation of a resource is an interaction between the occurrence and level of the resource, and the organism's need (internal state) for that resource. After the interaction, the internal state of the organism is modified.

In pelagic ecosystems, biological resources such as food have their own dynamics: planktonic prey, phytoplankton, microzooplankton, bacteria, all reproduce (and die). They are renewable resources. The interaction of an organism with a renewable resource does not irreversibly reduce or exhaust the resource.

4.3. Deposited resources

Some resources may be absent when an organism arrives at a given position, and produced or left by the organism when it leaves the position. They originate from the organism's metabolism, and thus come from an exploitable resource found elsewhere (or earlier) by the organism. In the ocean, this is the case for dissolved organic matter, or faecal pellets. An organism may be viewed as a temporary buffer, able to sequestrate some resource, which may be released elsewhere (or later). It may also be a transformer of a resource. These resources resulting from the metabolism of other organisms may in turn become resources for other organisms in the trophic food web.

5. HOOD design of space and resources

We may now set up the specification and role of a hierarchically organized object containing the different kinds of resources discussed above. The name Environment captures the meaning of this hierarchically organized object better than Space_Management (the name used in CALIFE vers. 3.40). Unfortunately, in the HOOD terminology, the name "environment object" stands for another system software incorporated in a design (Delatte et al., 1993, p. 31); in ecology, however, environment has a well-established meaning which should prevail here. This high-level object must account for the organization and role of spatially distributed resources, and also must solve the technical problem related to the concurrent access to these resources.

The Environment object should thus contain (1) child objects representing the different kinds of spatialized resources, and (2) a control object mediating the concurrent access to these resources, and solving the mutual exclusion problem.

The design of the Environment object may be captured in a HOOD diagram (Fig. 1). This diagram should replace, and is a refinement of, the Space_Management object in fig. 1 of Laval (1995a). The child objects in this diagram are as follows.

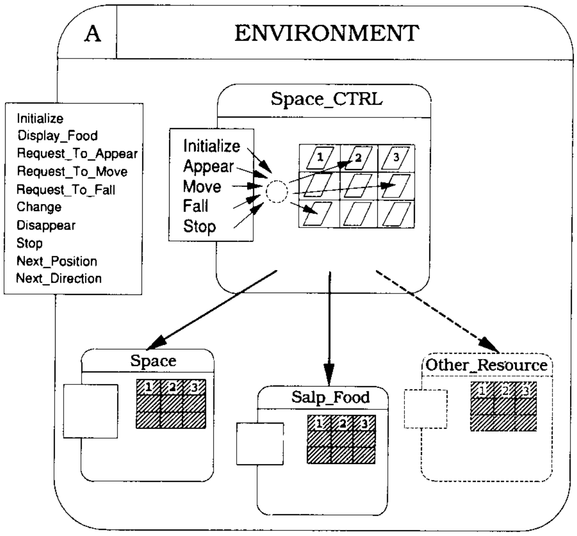

Fig. 1. Simplified HOOD diagram of the Environment object. The Space_CTRL object contains an array of tasks (shown as small parallelograms; HOOD has no graphical representation for tasks). Each task is responsible for a region of space (corresponding to the numbers 1, 2, 3...). Salp_Food is an instantiation of a generic Food package (not shown); the stippled object "Other_Resource" indicates that other resource objects may be instantiated as well. Objects may only use the routines figuring in the interfaces boxes of the other objects at the same level. To simplify the diagram, the subprograms or functions made visible in the interface boxes of Space and Salp_Food are not detailed. The A in the upper left part signals that Environment is an "active" object (in the HOOD terminology), i.e. an object which manages the communications between its child objects.

5.1. The resources objects

5.1.1. The Space object

The Space object encapsulates a local array (X,Y) of positions. These positions contain specific codes indicating if a certain kind of organism is present or not. At the specification level, only two subprograms are needed to inquire what is present, or to change the code, at a given position.

At the implementation level, these two subprograms may be tied to any appropriate means to access the video hardware, either a graphic library providing these two functionalities, or two Ada interfaced assembly routines. In the current version, a simple memory address mapping to the CGA video memory has been used instead. This is a very portable solution, with no memory penalty (the space array is the video memory itself); the software may thus run on any personal computer, but with the drawback of limiting the space to a 25×80 array. Using pixels of different colours to display the organisms, instead of different ASCII characters, would allow the use of a 640×480 space on very common VGA video boards, and up to 1280×1048 on more advanced systems, without any change at higher levels of the program.

5.1.2. The Salp_Food object

Because food is a spatialized resource, a food object should also contain an (X,Y) array, whose locations correspond to those in the array contained in the Space object. There should be specific food objects representing different kinds of food. Objects with similar behaviours constitute a class (Booch, 1991). A class Food is expressed in Ada with a generic package. This generic Food package contains a function, Inquire_Food, returning the amount of food present at a given location, and a procedure, Update_Food, which, given a certain Amount_Requested, returns Amount_Delivered, updating the amount present at the given location. Salp_Food is the name of the generic instance relative to the Tunicate organisms modelled in CALIFE. Other food objects are not yet introduced in the current version of the software (4.50), but may be easily added by instantiating the Food generic template.

In the present version, food is a non-renewable resource: different initial amounts of food exist at given space positions, where they may only be decreased by Tunicates.

5.2. The control object

5.2.1. Concurrent access to resources

The control object manages all concurrent requests, either for food resources located at the same (X,Y) position, or for space occupancy itself (i.e. a location in the video memory) when two or more organisms need to be displayed.

Every move to a location passes through a controller, which serializes (serves one after the other) concurrent requests for the same location. This controller is an Ada task. Using one controller task for the whole spatial domain is clearly inadequate: with a personal computer with only one CPU, one task cannot queue, process, and dequeue all simultaneous requests when the organisms are very numerous. Experiments have shown that this results in freezing the program (Laval, 1995b). On the other hand, using as many controllers as locations would be impracticable: each Ada task has an associated overhead (in memory and processing time) which is not negligible. The solution lies in-between, with a manageable quantity of controllers each responsible of a small region of space. To evaluate this quantity one can consider the worst case, when there are moving organisms in almost each location, all willing to move (as may be the case at the peak of a bloom). With the present configuration, each controller deals with at most 72 requests, a limit which has proved satisfactory. Controllers are arranged in an array in a "control object", named Space_CTRL according to the HOOD terminology.

5.2.2. Concurrent moves

In the CALIFE software, besides the controller tasks, Ada tasks are also used to implement the behaviour of the organisms. Each organism has an attached individual task (Laval, 1995b). Moves, reproduction, physiological functions, are accomplished inside this task. For short, in the following I will name "zooid" every moving organism (live and dead oozooids, live and dead chains). Cadavers are represented in CALIFE because they make an important contribution to the flux of organic matter in the ecosystem. Dead zooids continue to move (they sink, i.e. they move vertically downward) until their complete decomposition.

Zooid moves simulate Tunicate movements. A rather complicated algorithm is used to find an available position for a move. When a position is occupied (by another zooid or by an obstacle), a position at 180° from the current direction is sought. This mimics the behaviour of oozooids and chains, which reverse their swimming direction when they bump to an obstacle (Bone and Trueman, 1983). If the position in this opposite direction is also occupied, the position in a slightly (1/8) differing direction is tried, and so on.

When a moving organism enters another region of space, its request for a free position is handled transparently by the control object: the latter simply directs the request to the controller task in charge of the new region.

5.2.3. De-synchronization of moves

Every zooid moves or reproduces at fixed points in the internal loop inside the task representing its behaviour. This results in quasi-synchronous moves of all the zooids, giving an unnatural jerky aspect to the simulation. Moreover, if all moves occur simultaneously, the controller tasks are likely to be overwhelmed when the zooids are numerous. To smooth the controller requests and the occurrences of moves, a very small variable lag is introduced before each reproduction (sexual and asexual). This is done with a random delay between 0 and 200 milliseconds. On the Ada tasking side, this delay has the advantage of adding a supplementary synchronization point (synchronization points allow the scheduler to switch to another task).

5.2.4. *Problems of scale*

Moving organisms explore the available space, consuming food if it is present. The production of chains allows them to successfully exploit food patches. The results attained so far realistically suggest the spreading of a real population: bursts of oozooids hatch from chains, move and feed, full grown zooids turn into cadavers, cadavers disappear after a decomposition duration, and so on.

Swimming movements are accomplished at every beat of the internal clock, every 1.0 second. This duration was found adequate to simulate the rather slow swimming of salps. It allows enough time to schedule the movements of the other organisms swimming concurrently. It may be set to a different value via the configuration file, but not a too different one. A value of 0.1 s is still possible, but salps then swim like rockets. A longer value, say 5.0 s, leaves the salps immobile most of the time. Their life cycle is also delayed by the same factor, so that the simulation takes much longer. It is important to note that individual movements and life cycle phases do not share the same time scale. However, in a model it is not necessary to represent every scale faithfully; what is important is to allow each organism to search a "representative" sample of the available space. This space is in turn not the real space available to a real salp population. It is a very limited area used to represent it on a computer. In character mode, this area contains 1216 positions. An acceptable trade-off, found by trial and error, was to allow each zooid to explore about 1/40 of the available space during its life. At 1 position explored each second, this corresponds to a mean longevity of about 30 s.

6. Feeding

With the introduction of a Salp_Food object, feeding and some of its effects on reproduction can now be simulated. This is done by means of a local variable (different for oozooids and chains) in each Tunicate task. The state of this variable represents the current level of the Tunicates' internal reserves. Each Tunicate is created with an initial amount I of reserves, representing the reserves present at birth in the eleoblast (Brien, 1948). Chains, representing the collective behaviour of N aggregates, contain N times the amount of reserves found in every aggregate (to simplify, initial food reserves and food requirements of aggregates and oozooids, which have roughly the same size, were made equal). Reproduction costs the equivalent of the metabolism of R food units to the reserves, and may only take place if the current reserves are greater than the amount corresponding to the metabolism of R. For the experiments, R was set so that the cost of reproduction corresponds to I, that is, the initial amount is just insufficient to permit reproduction. Food requirements for an individual are set to the amount R necessary for reproduction: if available at the position reached by a Tunicate, an amount of up to R food units will be removed from the environment. This amount is subtracted to the Salp_Food object (which is updated in mutual exclusion) at the corresponding position, and added to the local reserves of the Tunicate.

In CALIFE vers. 3.40, the swimming direction changed by 1/8 every 8 moves, to mimic the swimming in large circles of real Tunicates (Laval, 1995b). To add some strategy to the food search, in the present version this change is made every 4 moves if the level of resources is greater than 6 units (that is, when the organism is "satiated"), and every 8 moves otherwise (when it is "hungry"): the swimming circles are tighter when there is food, so that the organism stays closer to the food.

7. Computer implementation

The CALIFE program was developed with an Alsys 386-DOS Ada compiler and environment on a 80386 (20 MHz) micro-computer (Laval, 1995a). The version 4.50 of the software represents about 4000 lines of Ada (including 1500 lines of comments), not counting the utility libraries (about 2000 lines of code). The source code of the program is available upon request from the author.

This version is implemented in character mode. The zooids move in a 22 lines × 78 columns space (line 24 is used for displaying the elapsed time and for various messages, and line 25 for user prompts; a frame occupies lines 1 and 23, and columns 1 and 80). For this 22 × 78 "swimming area", a 3 × 10 array of controllers has proved adequate: the overhead introduced by the 30 supplementary tasks is tolerable, and sufficient to manage the simultaneous presence of about 300 zooid tasks occurring at the bloom peak.

8. Results

An initial parameters setting (Table 1) was established with values comparable to the ones found for *Salpa fusiformis*, one of the most common species in the Ligurian Sea, Mediterranean (Braconnot et al., 1988). In the sea, the number of aggregates per chain in this species is about 100, a number which would very quickly fill the available space on the computer. This number was set to 3, a value sufficient (because of the exponential growth) to start up a small population bloom. Twenty runs were made with this configuration, that is, with the same initial position and swimming direction for the first oozooid, the same life cycle parameters, etc. The same food configuration file was used (Fig. 2). In all experiments, all the food was eventually consumed, except at 1 or 2 locations in a very few cases.

Despite the identical starting configuration, the total numbers of oozooids and chains produced is markedly different from one run to the other. To dismiss the objection that this could be attributed only to the presence of the random delay introduced before each reproduction (Section 5.2.3), a first test was made with this delay suppressed (more precisely with a delay of 0.0 second, which preserves the synchronization point). The results were still variable, with a coefficient of variation (standard deviation/mean × 100) of 4.28% for the oozooids and 4.92% for the chains. Moreover, the curves of the numbers of oozooids produced (and chains, not shown) were widely different (Fig. 3). The total number of oozooid and chains

Table 1
Initial parameters setting for the experiments with the random delay suppressed (Fig. 3). Durations in seconds

Simulation duration	140.0
Number of chain generations	2
Number of aggregates per chain	3
For oozooids:	
First chain emission	5.0
Interval between chains	4.0
Longevity	12.0
Cadaver persistency	6.0
For chains:	
Oozooid emission	6.0
Longevity	20.0
Cadaver persistency	6.0

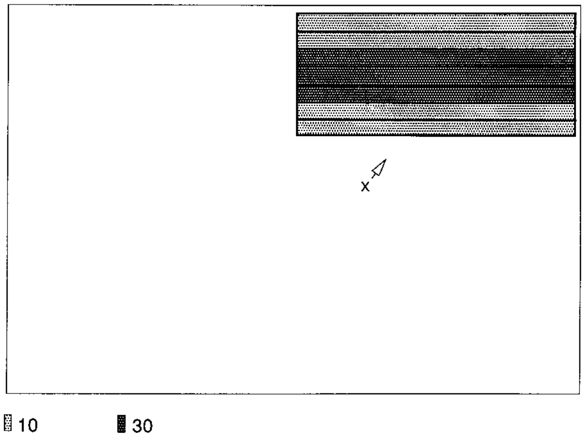

Fig. 2. Initial amounts of food used for all experiments. Food is present in the upper right part of the screen, with amounts of 10 or 30 units at each shaded position. × and arrow: initial position and swimming direction of the first created oozooid.

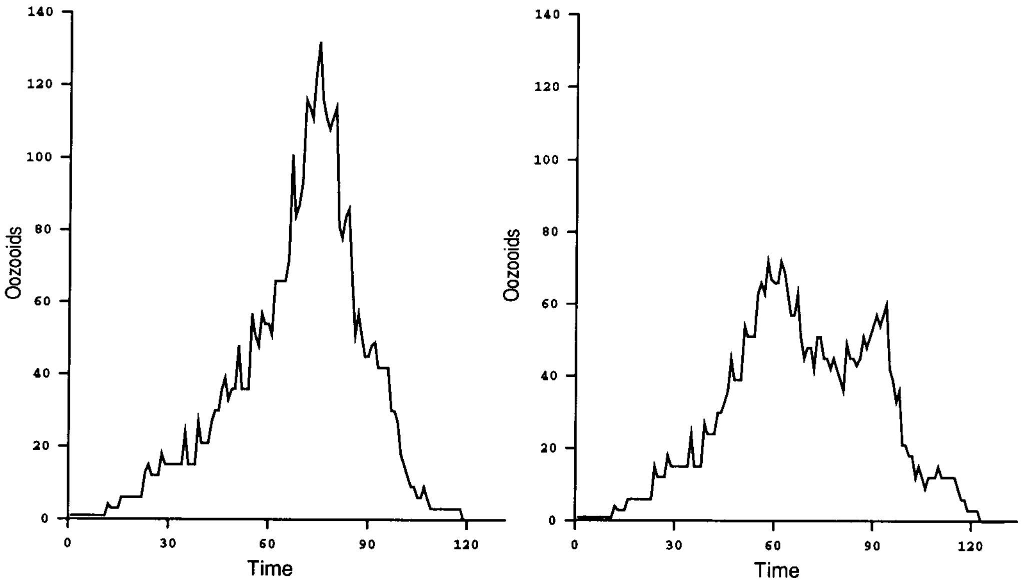

Fig. 3. Two widely different outputs from 20 runs with the random delay disabled (set to 0.0). Three aggregates per chain. Only the numbers of oozooids produced are shown.

produced may almost double in some runs compared to others.

With 5 aggregates per chain, instead of 3, the exponential growth of the population is more apparent: each chain may be seen emitting clutches of 5 oozooids, each of them giving new chains, etc. Introducing a random delay before each reproduction smooth the simulation, and spreads out the work of the scheduler. The coefficients of variation of the numbers of zooids and chains produced are of the same magnitude than with no random delay (4.09% and 4.89%).

Setting the initial oozooid position and direction so that this oozooid is bound to encounter a rich source of food was a way to repeatedly produce blooms. Providing a more oligotrophic environment (Fig. 4) shows how the explosive reproductive strategy of salps allows them to nevertheless exploit these poorer conditions. Going from patch to patch, the population is able to spread each time it meets a favourable environment. In the experiment shown in Fig. 5, the population first invades the right part of the space (Fig. 5A). A number of chains are produced, which, together with the newborn oozooids, eat all the food present locally. Then a few individuals reach some food patches in the left side, where they also reproduce explosively (Fig. 5B). At the end of the simulation, almost all

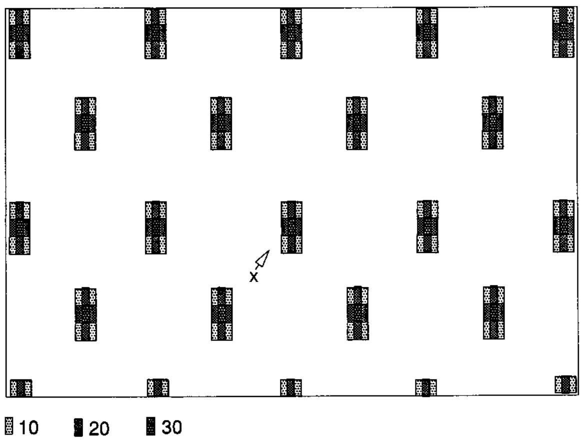

Fig. 4. A patchy, oligotrophic, environment provided for the experiment in Fig. 5. Conventions as in Fig. 2.

the food is consumed. For this experiment, the durations of the life cycle parameters were those of Table 1, multiplied by 2 to obtain a mean longevity of about 30 s.

9. Discussion

The numbers of zooids produced by the CALIFE program resemble times series obtained from

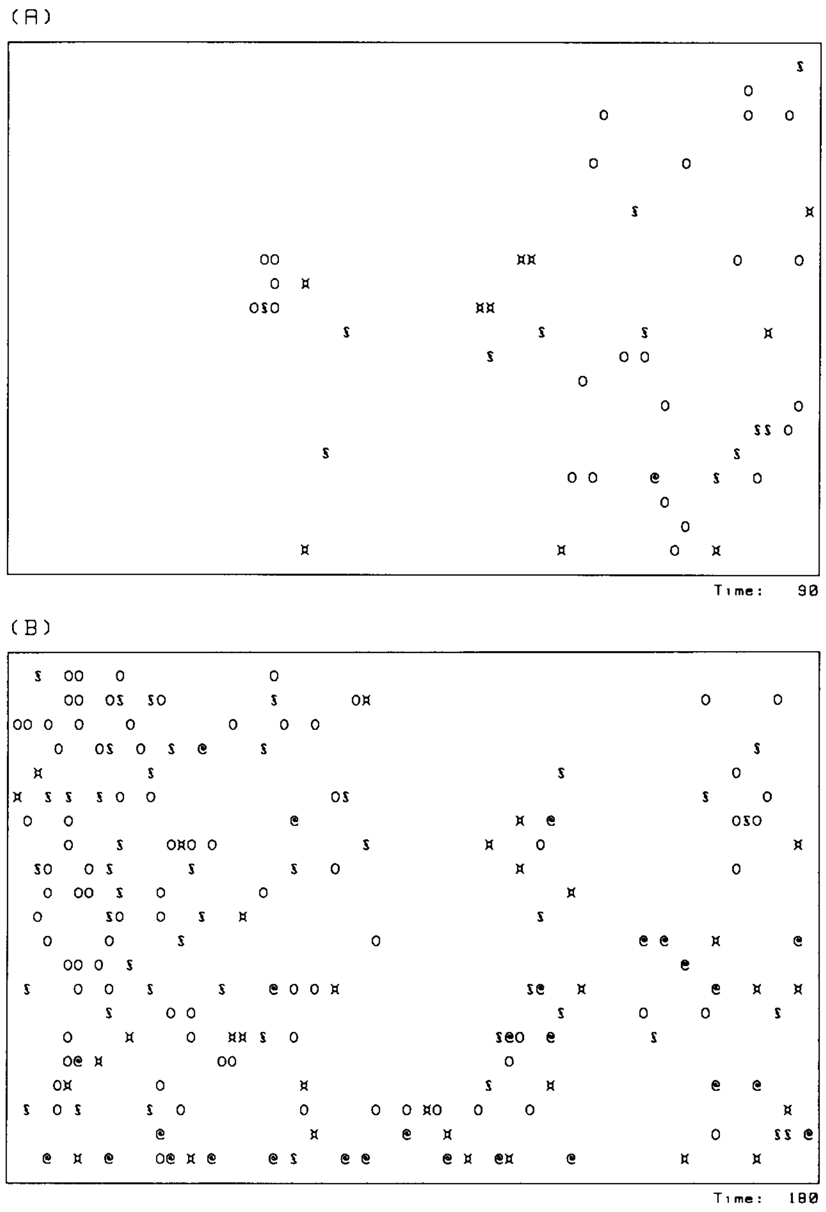

Fig. 5. Development of an artificial salp population in an oligotrophic environment (see Fig. 4). (A) After 90 time units, some zooids encountered food patches in the right part of the area, and the population begins to multiply. On the far left, the 5 oozooids released by a chain may be seen close together. (B) After 180 time units, the population has invaded the left part of the area. The population is less dense in the right part, because all the food was consumed here, and before dying the older individuals were not able to reproduce. Live oozooids are represented by the character "O", oozooid cadavers by "⊗", live chains by "§", and chain cadavers by "@".

plankton samples (see for example fig. 5 of Braconnot, 1971, p. 270). The large variability from one run to the next does not detract from this impression. However, sampling intervals of plankton data are usually wide (at best one or two samples per week), and there are many causes of bias (due to sampling in different water masses, net clogging, inadequate mesh size, etc.), so that a comparison is difficult. The spatial aspect seems also representative of a real salp bloom: the observation of blooming salps from submersibles (personal observations) shows that chains may then be close together, presenting a picture quite similar to the program output.

Even with fixed initial amounts of food at predetermined locations, and an oversimplified food chain reduced to one filter-feeder and its food, some complexity is readily apparent in the model output. No successive runs (with identical parameter settings) gave identical results, an outcome which may disconcert modellers used to sequential programs. The ranges obtained, however, are well within biological bounds. The unpredictable paths followed by the individuals result from their asynchronous behaviour and the numerous changes of swimming direction they make to avoid collisions with each others. These different paths lead them to different regions of space, where the amounts of food may be different. At these locations they accumulate different amounts of reserves, which govern their reproductive success.

The variability of the numbers of oozooids and chains produced by the program helps to explain how a successful exploitation of the casual occurrence of food material may arise. Salps such as *Salpa fusiformis* have a filtration rate of about 0.3 $\mathrm{l\ h^{-1}}$ per individual (Andersen, 1985). A chain of 100 aggregates may thus filter 30 l of water per hour. An oozooid reaching a place where food is abundant gives birth in a few days to a chain of 100 aggregates (Braconnot et al., 1988). These 100 aggregates, if there is enough food, will in turn give 100 oozooids. *S. fusiformis* travel at mean velocities between 1.3 and 6.6 $\mathrm{cm\ s^{-1}}$ if they are not feeding (Bone and Trueman, 1983). *S. aspera*, a related species about the same size as *S. fusiformis*, may undergo daily vertical migrations of 800 m (Wiebe et al., 1979). The strategy of space occupancy of the non-migrant salps seems more or less reflected by the CALIFE software: moving in circles until food is encountered, and then spreading by explosive multiplication. The smallness of the screen in character mode has until now prevented experimentation with gradient attraction, which may play a role in food detection, and with vertical migrations. In the Ligurian Sea, however, and probably in the whole Mediterranean Sea, *S. fusiformis* does not seem to undertake vertical migrations (Laval et al., 1992).

The hardware limitations in character mode (a position represented by one character on the screen) impose a too severe constraint on the space available. These limitations should disappear in pixel mode. This mode, besides another video board and more memory, requires only a few changes in the software, owing to the object-oriented design. In pixel mode, the effect of food patches could be freely investigated because the bulk of the population would not have enough time to reach the screen edges.

The efficiency of Tunicates in exploiting their food resource is the result of a long evolution which has tuned some key parameters to their present ranges of values. The most important parameters are presumably those governing the timing of the different phases of the life cycle, the number of chain generations, the number of aggregates in a chain.

In the CALIFE software an attempt has been made to simulate the effect of these key parameters. If the model is capable of reproducing the blooming of Tunicates, it would be possible to experiment with different values, and to understand how successful strategies developed.

Andersen and Nival (1986) and Braconnot et al. (1988) used differential equations to model the development of salp blooms. In these procedural programs, the flow of control affects globally the data, and equations are applied step by step, in a sequential manner. The deterministic nature of the method precludes the appearance of variability in the results. Ménard et al. (1994) fitted a Markov regression model to ordinal time series of observations. Here transition probabilities allow some variability, as well as dependence from previous (global) states of the population. The CALIFE object-oriented software goes further to express natural variability, because it handles entities concurrently, with control local to each entity; these entities modify independently their own state according to their age, the level of their reserves, the interactions with the other organisms. Considering that sources of variability were kept minimal, this program once further developed, could presumably account for non-linear effects impossible to formalize in manageable equations.

In CALIFE, artificial organisms "bloom" simply because they reproduce exponentially in areas where food is present, and starve when they are away from food. This may be contrasted with artificial ecologies where organisms stick together because they cooperate in the exploitation of a food resource (Assad and Packard, 1992). No such cooperation is likely to be present in relatively primitive organisms such as Tunicates. If there is no active cooperation between individuals, then the bloom size is at most the size of the food patch plus the mean horizontal distance a zooid and its descendants may travel using the reserves acquired in the patch. The CALIFE simulation indicates that if the food distribution is tight, the bloom may not extend very far from the food source.

In CALIFE, food is not a renewable resource. If it were, there would be no constraint for the Tunicate exponential growth. However, this is only a time scale problem: in the ocean, the abundance of food shows a marked seasonality. For a simulation over a longer period of time, it should be possible to make Salp_Food an "active" object, that is an object driven by a task. The Inquire_Food function will then take a Current_Time parameter besides its At_Position parameter. The food object will then have a dynamical behaviour, with food levels updated according to, for example, a phytoplankton growth function.

In the currently highly simplified representation of a Tunicate bloom, reproduction, as well as other metabolic activities, is not represented by its true cost: the production of more aggregates per chain should be dependent of the accumulation of more reserves. In the present version of the software, the number of aggregates per chains is arbitrarily fixed for the course of a simulation. The longevity of oozooids and chains, as well as the other life cycle parameters (see Table 1) are also constant for a given simulation. This was done in order to clearly understand the sources of variability.

The next step will be to improve the representation of the physiology of the artificial salps. To arrive at an energy budget, these artificial organisms should not only simulate swimming, but also respiration and growth, with the corresponding metabolic costs. The initial reserves of the progeny should also depend on the "health" of the parent at the time of reproduction. The number of chain generations obviously depends on the reserves accumulated, but also on the age of the oozooid. The longevity presents the same sort of problem. It is presently fixed in the software, but it should better depend on the balance between metabolic losses (respiration, cost of swimming, cost of reproduction) and gains (by feeding), with a genetic limit. A starved zooid should die not only because it is too old and cannot reproduce (as is presently the case), but because it cannot compensate its metabolic losses.

It is certainly possible to further refine the simulation, but it should be remembered that "a model is not reality, but something that imitates reality at a certain scale" (Morrison, 1991, p. 5).

Interactions with other artificial species would be the next achievement. This would permit the simulation of predator–prey, or parasite–host interactions, which are so important for the study of populations. But before undertaking these highly demanding software works, it was important to set the stage for these improvements. The CALIFE design now contains all the required basic items.

Acknowledgements

This research was supported by the Centre National de la Recherche Scientifique (CNRS/INSU, URA 716 "Écologie du plancton marin"). The author thanks J.C. Braconnot, S. Dallot, and J. Dolan for their helpful comments.

References

Andersen, V., 1985. Filtration and ingestion rates of *Salpa fusiformis* Cuvier (Tunicata: Thaliacea): effects on size, individual weight and algal concentration. J. Exp. Mar. Biol. Ecol., 87: 13–29.

Andersen, V. and Nival, P., 1986. A model of the population dynamics of salps in coastal waters of the Ligurian Sea. J. Plankton Res., 8: 1091–1110.

Assad, A.M. and Packard, N.H., 1992. Emergent colonization in an artificial ecology. In: F.J. Varela and P. Bourgine (Editors), Toward a Practice of Autonomous Systems. Proc. 1st European Conf. on Artificial Life. A Bradford Book, MIT Press, Cambridge, MA, pp. 143–152.

Bone, Q. and Trueman, E.R., 1983. Jet propulsion in salps (Tunicata: Thaliacea). J. Zool., Lond., 201: 481–506.

Booch, G., 1987. Software Components with Ada. Structures, Tools, and Subsystems. Benjamin/Cummings Publ., Menlo Park, CA, 635 pp.

Booch, G., 1991. Object-Oriented Design with Applications. Benjamin/Cummings Publ., Redwood City, CA, 580 pp.

Braconnot, J.-C., 1971. Contribution à l'étude biologique et écologique des Tuniciers pélagiques Salpides et Doliolides. Vie Milieu, Sér. B, 23: 257–286.

Braconnot, J.-C., Choe, S.-M. and Nival, P., 1988. La croissance et le développement de *Salpa fusiformis* Cuvier (Tunicate, Thaliacea). Ann. Inst. Océanogr., Paris, 64: 101–114.

Brien, P., 1948. Embranchement des Tuniciers. In: P.P. Grassé, Traité de Zoologie, 11. Masson et Cie, Paris, pp. 858–859.

Costanza, R. and Hannon, B., 1989. Dealing with the "mixed units" problem in ecosystem network analysis. In: F. Wulff, J.G. Field and K.H. Mann (Editors), Network Analysis in Marine Ecology. Springer-Verlag, Berlin, pp. 90–115.

Delatte, B., Heitz, M. and Muller, J.F., 1993. HOOD Reference Manual 3.1. Masson, Paris and Prentice Hall, London, 175 pp.

Fortier, L., Le Fèvre, J. and Legendre, L., 1994. Export of biogenic carbon to fish and to the deep ocean: the role of large planktonic microphages. J. Plankton Res., 16: 809–839.

Kirsta, Yu.B, 1994. Exchange of information in natural hierarchical systems. Ecol. Model., 73: 269–280.

Laval, Ph., 1995a. Hierarchical object-oriented design of a concurrent, individual-based, model of a pelagic Tunicate bloom. Ecol. Model., 82: 265–276.

Laval, Ph., 1995b. Implementing self-reproducing artificial organisms with Ada. ACM Ada Lett., 15: 46–53.

Laval, Ph., Braconnot, J.C. and Lins da Silva, N., 1992. Deep planktonic filter-feeders found in the aphotic zone with the Cyana submersible in the Ligurian Sea (NW Mediterranean). Mar. Ecol. Progr. Ser., 79: 235–241.

Margalef, R., 1979. The organization of space. Oikos, 33: 152–159.

Ménard, F., Dallot, S., Thomas, G. and Braconnot, J.C., 1994. Temporal fluctuations of two Mediterranean salp populations from 1967 to 1990. Analysis of the influence of environmental variables using a Markov chain model. Mar. Ecol. Progr. Ser., 104: 139–152.

Morrison, F., 1991. The Art of Modeling Dynamics Systems. Forecasting for Chaos, Randomness, and Determinism. J. Wiley and Sons, New York, 387 pp.

Odum, H.T., 1988. Self-organization, transformity, and information. Science, 242: 1132–1139.

Wiebe, P.H., Madin, L.P., Haury, L.R., Harbison, G.R. and Philbin, L.M., 1979. Diel vertical migration by *Salpa aspera* and its potential for large scale particulate organic matter transport to the deep-sea. Mar. Biol., 53: 249–255.