Draft version of the paper: Alfonseca, M., Carro, R.M., de Lara, J. and Pulido, E. (1998): Education in Ecology at the Internet with an Object-Oriented Simulation Language. Proceedings of the EUROSIM'98 Simulation Congress at Espoo, Finland, April 1998, Vol. 2, 118-123.

Education in Ecology at the Internet with an Object-Oriented Simulation Language

Manuel Alfonseca, Rosa Carro, Juan de Lara, Estrella Pulido

Dept. Ingeniería Informática, Universidad Autónoma de Madrid

{Manuel Alfonseca, Rosa Carro, Juan de Lara, Estrella Pulido}@ii.uam.es

Acknowledgement: This paper has been sponsored by the Spanish Interdepartmental Commission of Science and Technology (CICYT), project number TEL97-0306.

Abstract

This paper describes the procedure we have used to generate semiautomatically a course on Ecology for the high-school level, using an extension of the Volterra equations to describe the interaction between different species levels. The simulations used in the course have been written in our own special-purpose object-oriented continuous simulation language (OOCSMP). Our compiler for this language automatically generates Java code and html pages, which must be completed manually with the associated text and icons.

Introduction

System simulation [1] is one of the standard tools of multimedia systems and computer education. In continuous simulation, the appropriate mathematical tool is the set of algebraic-differential equations. Continuous simulation models are usually programmed in a special purpose language, or in general purpose code. Continuous simulation languages use different input interfaces, such as block diagrams [2], bond graphs [3], systems dynamics graphs [4], or mathematical equations [5].

Object-oriented programming originated in the sixties in a discrete simulation language, SIMULA67 [6], which incorporated many of the ideas later included by Alan Kay in the first general purpose object-oriented language, Smalltalk [7], and by Bjarne Stroustrup in C++ [8]. In recent years, object orientation has been added to continuous simulation languages [9]. The language we are using to generate our educational applications (OOCSMP) is our own extension to the old CSMP language (Continuous System Modelling Program), sponsored by IBM [10].

In a previous paper [11], we have described some of the new features of OOCSMP that make it very useful for the simulation of systems consisting of many similar interactuating parts, as in the ecological example we are showing here in detail.

The paper is structured as follows: the first section presents the equations used to model ecosystems; the second describes how the model is programmed in OOCSMP; the process of generation of the Internet course is explained next; the last section describes a specific course on Ecology; finally, the conclusions and our future work are presented.

Simulation of a complex ecosystem

We have made use of an extension of the well-known Volterra equations [12] to represent the dynamics of a complex ecological system. The original equations apply to a two-level two-species ecosystem, where a single predator population interacts with a single prey population in a simple way. The equations are:

$$X' = -m.X + n.X.Y$$
$$Y' = p.Y - q.X.Y$$

where X is the predator population; Y the prey population; and the four constants, m, n, p, q, are all positive. The prey is assumed to feed from an inexhaustible source, therefore, in the absence of the predator (X=0), its number will increase at the rate indicated by constant p. In the presence of the predator, its number will diminish in a proportion to the number of encounters (constant q). In the absence of the prey, the predator will dwindle at the rate given by constant m. In its presence, its number will increase in proportion to the number of encounters (constant n). The system is in equilibrium if the following holds:

```
  X0 = p/q
  Y0 = m/n
```

where X0 and Y0 are the respective initial populations of predator and prey.

Our first extension is to assume that the source of food for the prey (a plant or any other primary producer) is not inexhaustible, but is subject to the same rules as the other two species. We have thus a three-level system, whose behaviour is ruled by the following equations:

```
  X' = -m.X + n.X.Y
  Y' = -p.Y - q.X.Y + c.Y.Z
  Z' =  a.Z - b.Y.Z
```

where all the constants are positive. The generalization is straightforward. Each equation defines the rate of change of a population as the algebraic sum of two kinds of terms: one (call it the "self" term) describes the evolution of the population in the absence of other species, and is proportional to the population of the same species; its coefficient is positive for the primary producer level, negative for all others. Additional terms are proportional to the product of the population of this species by those at immediately upper and lower ecological levels, and they are positive for encounters with a lower level species, negative for encounters with one of the higher ecological level. It is very easy to extend this model to more than three levels, by applying the same concepts.

Our second extension is to allow for several different species at the same ecological level. The only effect of this extension is to include additional equations, as well as new "encounter" terms in the old equations. For two species of prey, with populations Y1 and Y2, the equations would become:

```
  X' = -m.X + n1.X.Y1 + n2.X.Y2
  Y1'= -p.Y1 - q.X.Y1 + c.Y1.Z
  Y2'= -p.Y2 - q.X.Y2 + c.Y2.Z
  Z' =  a.Z - b1.Y1.Z - b2.Y2.Z
```

Programming the model in OOCSMP

For a given number of species, the model described above may be programmed easily in any general-purpose or continuous simulation language. However, as the numbers of levels and species per level increases, the numbers of equations and terms per equation could become unmanageable. In OOCSMP, however, the object-oriented facilities make it possible to model an arbitrary ecological system, with almost no limitations.

We use a hierarchical set of classes to represent the different ecological levels. The top class in the hierarchy, called Species, is defined as indicated in listing 1.

```
************************************
* Definition of the Species class  *
************************************
CLASS Species {
 NAME name
 DATA M, X0, start
 X:=STEP(start)*LIMIT(0,1000,XT)
 XT:=INTGRL(X0,XP)
 XP:=M*X
 PLOT X TIME
}
```

Listing 1: OOCSMP definition of the Species class (file Species.CSM)

There are three attributes in the Species class: the start time (which makes it possible to simulate invasion of an ecosystem by a new species through the use of the STEP block); the initial population at start time

(X0), and the coefficient for the "self" term (M). The equations describe the evolution of a population in the absence of other species, and contain a limitation to a maximum of 1000 and a minimum of zero, to prevent the absurd of negative populations.

The different levels in the ecological system are defined as classes derived from Species, as indicated in listing 2.

```
************************************
* Definition of the Plant class    *
************************************
CLASS Plant : Species {
 DATA AHPl
 XP-= AHPl*X*Herbivore.X
}
************************************
* Definition of the Herbivore class*
************************************
CLASS Herbivore : Species {
 DATA APlH, APrH
 XP+= APlH*X*Plant.X
 XP-= APrH*X*Predator.X
}
************************************
* Definition of the Predator class *
************************************
CLASS Predator : Species {
 DATA AHPr
 XP+= AHPr*X*Herbivore.X
}
```

Listing 2: OOCSMP definition of three ecological levels (file Levels.CSM)

Each class defines only the additional encounter terms with which other species influence its population, and adds the corresponding multiplication attributes. All the species in a given level are represented by their class name. The compiler generates a loop for generic terms of this kind.

A given complete ecological model, combining several species for each level, can now be programmed very easily, as in the example at listing 3.

```
INCLUDE Species.CSM
INCLUDE Levels.CSM
************************************
* Actual species                   *
************************************
* Level 0: plant
Plant    PA ("Plant", 0.4, 100,  0, 0.02)
* Level 1: herbivores
Herbivore HA ("Herb1",-1.28, 20,  0, 0.02, 0.36)
Herbivore HB ("Herb2",-1.28,  5, 10, 0.02, 0.12)
* Level 2: predators
Predator  PrA("Pred1",-7.2,   2,  0, 0.36)
Predator  PrB("Pred2",-4,     2, 10, 0.20)
System := PA, HA, HB, PrA, PrB
System.STEP()
************************************
* Timer and show data              *
************************************
TIMER delta:=0.005,FINTIM:=40,PRdelta:=0.1,PLdelta:=0.1
METHOD ADAMS
```

Listing 3: OOCSMP definition of an ecosystem

Semiautomatic generation of an educational course for the Internet

The OOCSMP models we have developed are translated into standard object-oriented languages by means of a compiler we have built. Depending on the compiling options used, we can tailor the compiler to:

- Generate C++ code and test it under DOS.
- Generate Java code and test it under a Java interpreter or an applet viewer.

- Generate Java code and embed the applet in an html page.
- Automatically add buttons to select between different execution alternatives of the model.
- Provide special windows and widgets to modify the values of the model parameters and make it possible to perform "what if" experiments.

To debug the models, we usually start by generating C++ code, provided with a graphic interface that makes it very easy to adjust the values of the different parameters in the system and shows the results of the simulations as graphical plots.

Once debugged and adjusted, the same compiler, invoked with different options, can be used to generate Java applets and html skeletons. The same model, with a few adjustments, may be used to simulate different ecological systems, and thus generate the several html pages that make up the course.

The automatic preparation of the course ends here. Manual adjustment are needed to fill the html skeletons with explanations, images and cross references to other pages. The resulting set of pages is ready to be made available to the students through the Internet.

A practical course on Ecology

The procedure described in the previous section has been followed to generate a course on Ecology aimed at high-school students. The course consists of five pages presenting the following ecosystems:

- An isolated three-species system (one primary producer, one prey and one predator), with the appropriate parameters to bring the ecosystem out of equilibrium. The student can observe the periodicity of this kind of systems.

- A three-species system in equilibrium.

- A three-species system, originally in equilibrium, which is invaded after some time, first by a new prey, then by a new predator. The first invasion takes the system out of equilibrium. After the second invasion, the whole five-species system reaches a new periodical stability.

- The same system, invaded after some time, first by a new predator, then by a new prey. The periodical stability attained after some time is completely different to the preceding case.

- A five-species system with an user interface that lets the student modify the different parameters to perform experiments. Some of these experiments are suggested by the text of the page, but the student may perform many more.

As mentioned above, the main components of the HTML pages are the simulations of ecosystems presenting different features. The outputs of these simulations are presented graphically in two different ways:

- A plot of the evolution of the populations of the different species along time.

- An iconic representation of the same populations. The first four pages use preestablished icons to represent the different species involved. The last page allows the student to select the desired icons. The number of icons of a given species shown is proportional to its population at any moment. As populations wax and wane, icons appear or disappear.

The plot generator is automatically added by the OOCSMP compiler. The icon generator has been programmed by hand and added manually to the simulation applet. It is our future objective to extend the compiler so that it is able to generate this type of out put representations automatically.

Conclusion

The procedure described here makes it very easy to build Internet educational courses based on the simulation of physical, biological and other systems. The OOCSMP language, designed for the purpose, contains features that simplify programming the models. The compiler we have developed automatically generates C++ and Java code, as well as html skeletons. Some extra effort is required to complete the course, with the addition of textual explanations and an iconic representation of the evolution of the system. However, this effort is small, as compared to the actual development of the model.

The example proposed in the paper (a practical course on Ecology, executable on the Internet) can be found and tested at the following address: html://www.ii.uam.es/~epulido/ecology/simul.htm

In the future, we will extend the language and the compiler to integrate with the model the text for the html pages. The iconic output will also be provided as one of the standard output display options. In this way, the manual effort required to complete the course will become negligible. We also intend to develop a control system with access to a data base, so as to be able to control the performance of each student with the courses we generate. We are currently working to allow the dynamical generation of objects (species, in the ecological example), during the execution of the model.

The ecological model can also be improved by defining ecological niches. Species belonging to the same level, but occupying different niches would not compete among themselves. With this feature, more complicated and real ecological models could be included in the course.

References

[1] Y.Monsef, "Modelling and Simulation of Complex Systems", SCS Int., Erlangen, 1997.

[2] M.Alfonseca, "SIAL/71, a Continuous Simulation Compiler", in "Advances in Cybernetics and Systems", Ed. J. Rose, Gordon and Breach, London, Vol. 3, 1974, 1319-1340.

[3] D.Karnopp, "Bond Graph Models for Electrochemical Energy Storage: Electrical, Chemical and Thermal Effects", Journal of the Franklin Institute, Vol 324, 1990, pp. 983-992.

[4] A.A.Legasto Jr., J.W.Forrester, J.M.Lyneis, editors, "Systems Dynamics", North Holland, 1980.

[5] M.Alfonseca, "APL Continuous System Modelling Program: an Interactive Simu lation Language", Advances in Engineering Software, Vol.1:2, 1979, 73-76.

[6] O.J.Dahl, K.Nygaard, "SIMULA - An ALGOL-Based Simulation Language", Comm. ACM, 9:9, 1966, pp.671-678.

[7] Digitalk Inc., "Smalltalk/V PM", Digitalk Inc., Los Angeles, 1990.

[8] B.Stroustrup, "The C++ Programming Language", Addison-Wesley, Reading (Mass.), 1991-1997.

[9] H.Elmqvist, S.E.Mattson, "An Introduction to the Physical Modeling Language Modelica", Proc. 9th European Simulation Sympossium ESS97, SCS Int., Erlangen, 1997, pp. 110-114. See also http://www.Dynasim.se/Modelica/index.html.

[10] IBM Corp.: "Continuous System Modelling Program III (CSMP III) and Graphic Feature (CSMP III Graphic Feature) General Information Manual", IBM Canada, Ontario, GH19-7000, 1972.

[11] Alfonseca, M.; Pulido, E.; Lara, J.; Orosco, R.: "OOCSMP: An Object-Oriented Simulation Language", Proc. 9th European Simulation Symposium ESS97, SCS Int., Erlangen, 1997, pp. 44-48.

[12] Volterra, V.: "Leçons sur la Théorie Mathématique de la Lutte pour la Vie", Gauthier-Villars, Paris, 1931.