Transactions in GIS, 1996, vol. 1, no. 1, p. 40

Integrating dynamic environmental models in GIS: The development of a Dynamic Modelling language

CEES G WESSELING, DERC-JAN KARSSENBERG, PETER A BURROUGH AND WILLEM P A VAN DEURSEN

C G Wesseling, D Karssenberg and P A Burrough Department of Physical Geography, Utrecht University, PO Box 80.115, 3508 TC, Utrecht, The Netherlands

W P A Van Deursen Resource Analysis, Zuiderstraat 110, 2611 SJ, Delft, The Netherlands

An integrated, executable mathematical modelling language for environmental and ecological applications has been developed to create and run Dynamic Models in a GIS. The modelling language is embedded in the GIS, providing the ability to model complex space-time systems free from the technical burdens of database management and algorithm optimization. The spatial modelling language uses a structured script, which is illustrated here by an example of a model of plant dispersion.

Introduction

In recent years there have been major developments in the quantitative description of natural resources, environment and ecology through the linkage of environmental models to geographic information systems (GIS) (for example, Goodchild et al 1993; Goodchild et al 1996; Burrough 1996). This paper does not cover the scientific aspects of the quantitative modelling of natural processes such as erosion, runoff, ground water pollution or species dispersion (for reviews of this work see Goodchild et al 1993 or Burrough 1996), but describes the main features of a new computer tool for making this kind of modelling easily accessible to environmental scientists.

Current GIS have become indispensable as a source of data and data pre-processing for spatial models, and for analysing and presenting the results. The capability to derive spatial attributes from multiple sources, such as remote sensing, digital elevation models (DEMs), point samples, interpolated surfaces, and digitized versions of existing maps and to store them in a geographic database has revolutionized and simplified the data capture and data supply for the models. The display tools in GIS have provided powerful 2D and 3D visualization of results.

There have been three approaches to linking physical models and GIS, which we term *loose coupling*, *tight coupling*, and *embedded coupling* (Burrough 1996). In *loose coupling* the GIS (and any geostatistical software) is used to retrieve and pre-process the spatial data into the form required by the model structure. The data are written to files which are then used as input to the model program, which may reside on another computer. The model computes the results and returns them as files of point data or areal data which are then displayed (or interpolated and displayed) by the GIS. An example is the initial link between ANSWERS and a raster GIS (De Roo et al 1989).

Loose coupling is appropriate when a standard model is being linked to GIS as an experiment or as part of an exploratory process, or when there are particular computational requirements that are not provided by the GIS. Loose coupling may involve considerable work in changing data formats and data structures, particularly if the model has been obtained or purchased from another institute. The advantage of loose coupling is that existing model code can be used immediately, and this method seems to be favoured in disciplines with a strong tradition of mathematical modelling. The main disadvantages are that the model code cannot be changed easily without specialist knowledge and that much work in data management and format conversion may be necessary.

Tight coupling also involves export of data from GIS to the model with the return of results for display, but the input and output routines for the model have been rewritten in such a way that they can be addressed directly by the interactive tools of the GIS (setting up parameter values via menus, etc.) and the exchange of data is fully automatic.

This is appropriate if a given model is used as a standard for a large number of different applications (for example, the incorporation of MODFLOW in the Intergraph MGE system or the integration of ILWIS and MICRO-FEM (Biesheuvel and Hemker 1993) or TOPMODEL (Romanowicz et al 1993). With properly networked computers it is possible to have the model running on a completely different platform from the GIS without the user realizing the difference. A disadvantage of tight coupling is the considerable investment in

1361-1682/96/0001-0040

programming and data management that is needed and that it is even more difficult than in the case of loose coupling for the user to modify or rewrite the model, should that be required.

Embedded coupling is either 1) where a simple GIS is added to a complex modelling system to display results and provide interactive control or 2) where the model is written using the analytical engine of the GIS, which is often easy for empirical and simple process models, but usually difficult or impossible for dynamic models with complex space-time interactions. The former approach has often been favoured in ecological modelling, which is frequently more exploratory than in other areas and where there has been little perceived need for comprehensive GIS and cartographic facilities (for example, Silvertown et al 1992; Fedra 1996; Goodchild et al 1996).

The great advantage of embedded coupling where the model is written in an integrated programming language provided by the analytical engine of the GIS is that the user can construct his or her own models as required: the disadvantage is that the analytical power of the model is constrained by the power of the user interface. If this consists only of icons, menu commands, or simple English-language type statements then complex mathematical modelling is not possible. Many GIS allow users to write models as sequences of commands in a macro language, but few of these contain the data typing, functionality, and iteration required for dynamic spatial modelling with spatial and temporal data. The aim of the work reported here has been to circumvent these language problems and to provide a powerful spatio-temporal modelling language embedded in an easy-to-use GIS.

Most current GIS command languages permit the user to carry out many kinds of logical and mathematical modelling in the GIS itself, providing the aims of the model are not at odds with the basic assumptions and spatial structures of both GIS and data. Many GIS permit the direct computation of spatial interactions, such as indices of variability within windows of a given size, buffer zones, or fastest path over potential surfaces, which can be of great value for assessing the impacts of roads, computing shortest paths for water movement or pollutant transfer. However, sophisticated use of embedded models in most commercial GIS is currently not easy because they are not designed to deal specifically with spatial-temporal processes or processes in which mass balance and mass transport are integral features.

Dynamic models, with parameters and variables changing over time, are difficult to run in most GIS, because they have been designed for querying and maintaining a static database with static phenomena. Standard GIS do not explicitly allow dynamic phenomena to be stored and analyzed nor do they provide efficient facilities for iteration through time.

True integration, where a user-defined dynamic model runs entirely in the framework provided by the GIS, is only possible if the GIS supports all the operations required for dynamic modelling and provides user access through a proper programming language. In this paper we present such a modelling language, the associated analytical engine and the spatial database that are all part of the new PCRaster version (Van Deursen 1995; Van Deursen and Wesseling 1995).

Language requirements and design issues

Every model is written in some type of formal language, whether it is a series of Structured Query Language (SQL) statements, a FORTRAN or C++ program, a sequence of GIS macros or a formula in a spreadsheet. Even graphically designed flowcharts in which (spatial) operations are chained, (for example, ERDAS Spatial Modeller or STELLA flowcharts) are graphical representations of a programming language. Some of these programming languages are general purpose while others are specially designed for spatial analysis.

A well-designed programming language is essential in every area of software engineering. Our goal has been to develop a special purpose programming language for building dynamic models in a GIS environment. The data manipulated by the language are stored in the GIS database so the language design and the GIS data model design must be consistent with each other.

Environmental models cover a large variety of applications, ranging from simple erosion risk assessment descriptions based on generalized slope length and soil type characteristics to complex pollution models containing diffusion and advection processes. Thus the language must contain a large set of operators from which many types of models can be built. However, the set of operators must be chosen carefully, based on a number of often conflicting requirements. The set of operators should be as small as possible to keep the language easy to learn, but large enough to build all types of dynamic models. The expressive power of an individual operator should enable one to express models as compactly as possible without the risk of being obscure, which often accompanies compactness. These requirements are mainly associated with the level of abstraction that is needed to hide irrelevant details from the user, such as data read and write operations.

First an overview of the structure of the database is presented, followed by a description of the main features of the PCRaster Dynamic Modelling language, illustrated using an example of a model for seed dispersal. It is shown that no specialist programming knowledge is needed for development of a model in the PCRaster Dynamic Modelling language.

The database for dynamic spatial data

Computer languages are constructed around data types and data models. The language is the means by which the information in the data can be analysed. The question *how* to store data seems inevitable when discussing GIS and data models, as can be concluded from the everlasting raster versus vector discussion. More important, however, is the question *what* is stored in the database, and what the characteristics of the entities stored are.

For some models the important entities may include soil type maps and land-use maps. A cross tabulation of these entities is possible whether they are stored as rasters or polygons, but a multiplication of soil class numbers with land use numbers cannot produce anything useful since both entities are nominal data types. Even a comparison (equality) would be nonsense since both nominal data types represent different entities. To overcome these difficulties and to provide functional support of the operations, type information can be included in the database, and a strong type checking mechanism can be applied to the GIS operations.

The PCRaster database can hold raster maps, point data, tables for relations between maps (cross tabulation), and time series for representation of attributes changing over time. All these data representations have types attached. These types describe the special properties of the entities stored and the resulting data representation in the computer. The following data types are recognized:

- boolean
- nominal, subtyped by its legend
- ordinal, subtyped by its legend
- scalar field
- directional field
- vector field
- local drain direction (LDD)

Scalar fields are used to describe intensities and potentials of physical fields, such as temperature, population density, and precipitation. Directional fields apply to attributes that have a circular, continuous scale, such as aspect of the terrain. Vector fields have both magnitude and direction and can be used to represent (horizontal) fluxes and forces, such as movement of ground water and air.

The local drain direction (LDD) data type has been introduced to provide for the definition of the direction of potential flow. It is a data type that supplies the raster database with the topological linkages needed by operators that describe lateral fluxes of fluids or materials. In general, this LDD-map is derived from a Digital Elevation Model (DEM), and represents the direction of surface flow through this elevation map, but any scalar field showing potential differences can be processed to determine a LDD map. For a further discussion on the construction of this data type see Jenson and Domingue (1988) and Moore et al (1993).

Operations on the database entities can only be carried out if the data have the correct data type for the operator concerned. Such a strong type checking scheme assists error detection and helps the user when conceptualizing his/her ideas about the entities used. An additional advantage of typing the entities is that more intelligence, such as polymorphic behaviour, can be built into the operators. For example, an interpolation operator can automatically choose the nearest neighbour algorithm for ordinal data and a bilinear interpolation for scalar data.

The dynamic behaviour of the database, needed for Dynamic Modelling, is modelled by time series indexed on time and location, and by stacks of map layers representing the status of the model at different time steps.

The spatial modelling language: Primitive operators

The PCRaster spatial modelling language is an extension of the ideas behind Map Algebra and the Cartographic Modelling Language proposed by Berry and Tomlin (Berry 1987; Tomlin 1990) but includes new ideas of iterations used in Dynamic Modelling. It follows the same approach as Map Analysis Package (MAP) in the sense that it provides a set of generic operators, which can be used as primitives for the models.

More than 120 spatial and temporal operators are available in the PCRaster system: these include the usual point operators (analytical and arithmetical functions, Boolean operators, operators for relations, comparison, rounding, (random) field generation, etc.), neighbourhood operators for calculations in moving windows of variable size (highpass filtering, edge filtering, moving averages, etc.) and area operators for calculations within specified areas or classes (for instance soil groups) or calculation of distances or cost paths between cells. A rich suite of geomorphological and hydrological operators is available that goes beyond the range of operations generally considered as Map Algebra. These include functions for hill slope and catchment analysis, and the definition of topology for modelling transport (drainage) of material over a local drain direction map with routing functions. The main extension to MAP-type programs is that PCRaster provides time operators for retrieving and storing temporal data in iterative Dynamic Models.

The syntax of the language is based on mathematical equations, which means that the operators can be used and combined in the same way as in mathematical equations. Each equation assigns the value of an expression to a single output which can be a single variable or a whole map (overlay).

The way in which mathematical relations between attributes are written in PCRaster is illustrated by the following example of plant seed dispersal to neighbouring areas at the end of the growing season. Studies have shown that the probability of seeds of a plant growing at one location reaching an unvegetated location decreases exponentially with the distance from vegetated, seed producing locations (Harper 1977; McClanahan 1986):

$$p = 0.1^{d/r} \quad (1)$$

where: p is the probability that seed reaches the unvegetated location ($0 \leq p \leq 0.1$)
d is the distance from vegetated locations (m), and
r is a constant (the distance (range, m) for which $p = 0.1$)

In PCRaster this relation can be used to calculate the probability that each unvegetated, potentially colonizable cell on a map is reached by seed produced at vegetated cells. This is accomplished with two operations:

```
Distance = spread(Plants,0,1);                    (2)
Probability = 0.1 ** (Distance/Range);            (3)
```

The first operation uses the boolean input map Plants which codes cells where the plant is growing as TRUE; all other cells are FALSE. For each cell, the spread operator calculates the distance to the nearest TRUE cell on Plants and assigns this distance to the cell under consideration on the map Distance. The spread operator is an example of the generic implementation of individual operators. Its last two arguments can be used to modify the distance calculation with additional friction maps. The second operation results in the map Probability which for each cell contains the probability that the seed reaches the cell. It uses the input map Distance which resulted from the first operation and the Range for which a constant, non-spatial value may be chosen. For each cell the operator ** raises 0.1 to the power of the quotient of Distance and Range, according to equation 1.

To simulate which unvegetated sites will actually be reached by the seed, a random field generator is used:

```
NewPlants = Probability gt uniform(1)             (4)
```

For each cell, the uniform operator takes a different value from a random uniform distribution between 0 and 1. For each cell, the greater than (gt) operator compares the probability value on the cell with the random value on the cell: if Probability is greater than the random value, the cell is reached by the seed and the cell on the map NewPlants is assigned a boolean TRUE, otherwise a boolean FALSE is assigned to the cell on NewPlants.

The three statements described above can be given from the command line or can be typed in a static Cartographic Modelling script, which is executed by PCRaster, like a batch file. They describe a simple calculation for one time interval, but a dynamic model would show how the dispersal of plants progressed over time. The following section shows how PCRaster provides the functionality to embed these simple statements in a dynamic model that iterates the computations for a series of time steps.

The spatial modelling language: Dynamic Modelling scripts

The structured sequential Dynamic Modelling script

A Dynamic Modelling script consists of separate sections, each of which has a specific function. The division into sections is an essential concept of the Dynamic Modelling language. It tells the computer how to execute a model and it helps the user to structure the components of the model. The basic sections needed for building a sequential Dynamic Model are the ***binding section***, ***areamap section***, ***timer section***, ***initial section*** and the ***dynamic section***. These are shown schematically in Figure 1 and are explained as follows.

The ***binding section*** regulates the database management of the files used throughout the program. In principle, variables mentioned in the Dynamic Modelling script are directly linked to these in the PCRaster database: the names used in the script correspond with those in the database. The ***binding section*** allows for defining a different name in the script: it binds the database filename to the name in the model of a variable. If a model is run several times, each time with different input and resulting output variables, the user only needs to change the database filenames at one position in the ***binding section***, instead of changing all the occurrences in the script. Names in the script can be of any length to improve readability.

The ***areamap section*** defines the geographical attributes of the model area and the spatial resolution (grid size). It uses a clone map to specify the general geographical coordinates, extents and cell size for all maps in the model.

The ***timer section*** regulates the duration and time slice of the model through three parameters, start time (t^{start}), end time (t^{end}) and time slice (dt). The number of time steps in a model is the duration of the model divided by the time slice.

The ***initial section*** sets the initial conditions for the model, including maps and non-spatial attributes. These values may be defined with one or more PCRaster operation. These operations can also calculate the initial conditions from other data in the GIS if these are not already in the database. The initial values are used at the start of the dynamic section for the iteration at the first time step.

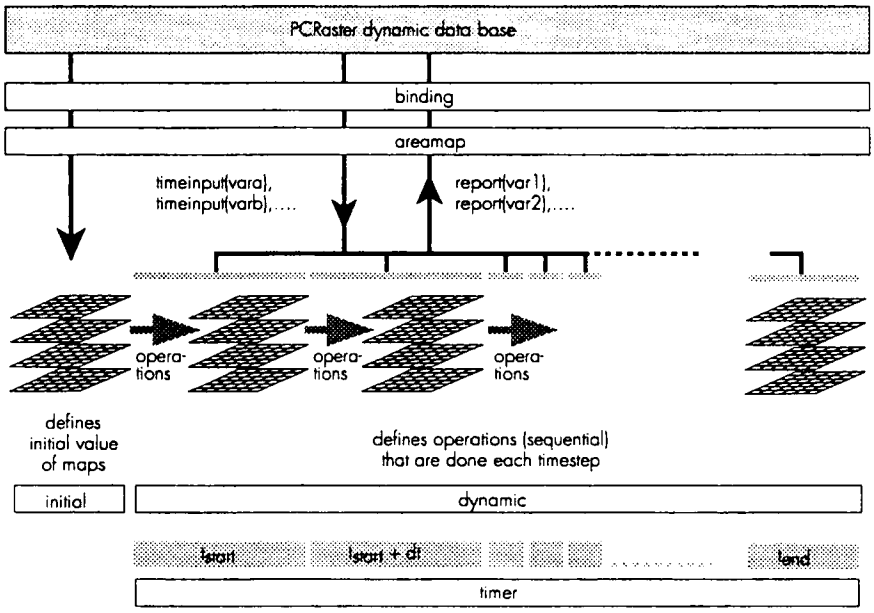

Figure 1. Conceptual picture diagram showing the sections in a sequential dynamic model.

The *dynamic section* defines the operations for each time step *i* that result in a map of values for that time step. Each time step consists of one or more PCRaster operations which are performed sequentially. The results of time step *i* are the input values for time step *i*+1, and so on. This section reads dynamic data from the database or stores model results in the database for each time step.

We illustrate the structure of the dynamic script by considering the problem of the dispersion of plants from a source area to a target area through potential colonization sites over a period of several years. The top two boxes in Figure 2 show a source area (south-east corner, a target area (north-west corner) and two different patterns of potential colonization sites (natularg.map large and well-spaced; natusmal.map small and finely spaced) that can act as stepping stones between source and target. The model illustrates possible dispersion over a period of 15 years.

Table 1 summarizes the script file for the model, which is based upon the exponential seed dispersion equation (Equation 1) described in the previous section. In both scenarios the script file is the same – only the map bound to the model variable Nature (which are maps of potential colonization sites) has been changed to reflect the difference in the distribution of the 'stepping stones'. The *binding section* defines a range of 350 m for the exponential model used (Equation 1). The model simulates 15 years of plant dispersion with a time slice of one year, defined in the *timer section*.

The *initial section* sets the initial distribution of the plant with a boolean map Plants. This boolean map contains TRUE cell values in the source area in the south-east (Figure 2 top) and FALSE cell values in the rest of the map. The initial absence of new plants in the potential colonizing sites is given by the boolean NewPlants map which is totally filled with FALSE (0) cell values. The initial age of the plants is set to zero for all cells (AgeVeg map).

The *dynamic section* of the model script describes the dispersion of the plants. Each year, the first operation calculates the distribution of the plant during the growing season. The plant will grow in a cell if it already grew in that cell during the previous year or if the cell is potentially colonizable and has been colonized as a result of seed dispersal during the previous year. These conditions are described by the boolean logic operators *or* and *and*: a cell on Plants has a value TRUE (vegetated with the plant) if: 1) Plants or NewPlants have a value TRUE *and* 2) Nature has a value TRUE. The operation uses the boolean maps Plants and NewPlants that were generated during the previous time step. The map of potential colonizing sites (Nature) is constant over time and is given in the binding section. The fourth operation results in the map NewPlants. Each year it contains a TRUE value for cells that are reached by the dispersed seed during that year. The remaining cells have a boolean FALSE. The last operation counts the number of years that a plant grows in a cell: it adds one year to a cell of AgeVeg if Plants has a value TRUE on that cell.

Storing model results in the PCRaster database is simply done by preceding an operation with the *report* keyword. Each time step, the seed dispersion model stores the variable AgeVeg in the database with a filename extension of the model time at the time step, resulting in a series (stack) of maps. On the screen this series of maps

can be displayed in animation mode resulting in a dynamic visualization of the spatial development of the plant over 15 years. Figure 2 shows only a selection of a few maps of such a visualization, it gives the age of the plants after 5, 10 and 15 years. The large distances (approximately 1 km) between the nature areas of the left scenario result in slow dispersion of the plant compared to the right scenario where finely dispersed nature areas are located approximately 100 m apart. This is because the range of 350 m for the exponential model (equation 1) results in too small a probability that the seed is transported over 1 km, the distance which is needed for the scenario with the well-spaced nature areas.

The example illustrates that it is possible to build and run an iterative model interactively. Model results can immediately be visualized, without the burden of data exchange. By changing the model or the model parameters in the script different scenarios can be computed and compared. The results of these scenarios could be the start of hypothesis generation or testing.

Integration of the model with the database: Retrieving and storing dynamic data

The power of the high level integration of the GIS and the Dynamic Model is manifested by the simplicity of data exchange between the database and the model which is

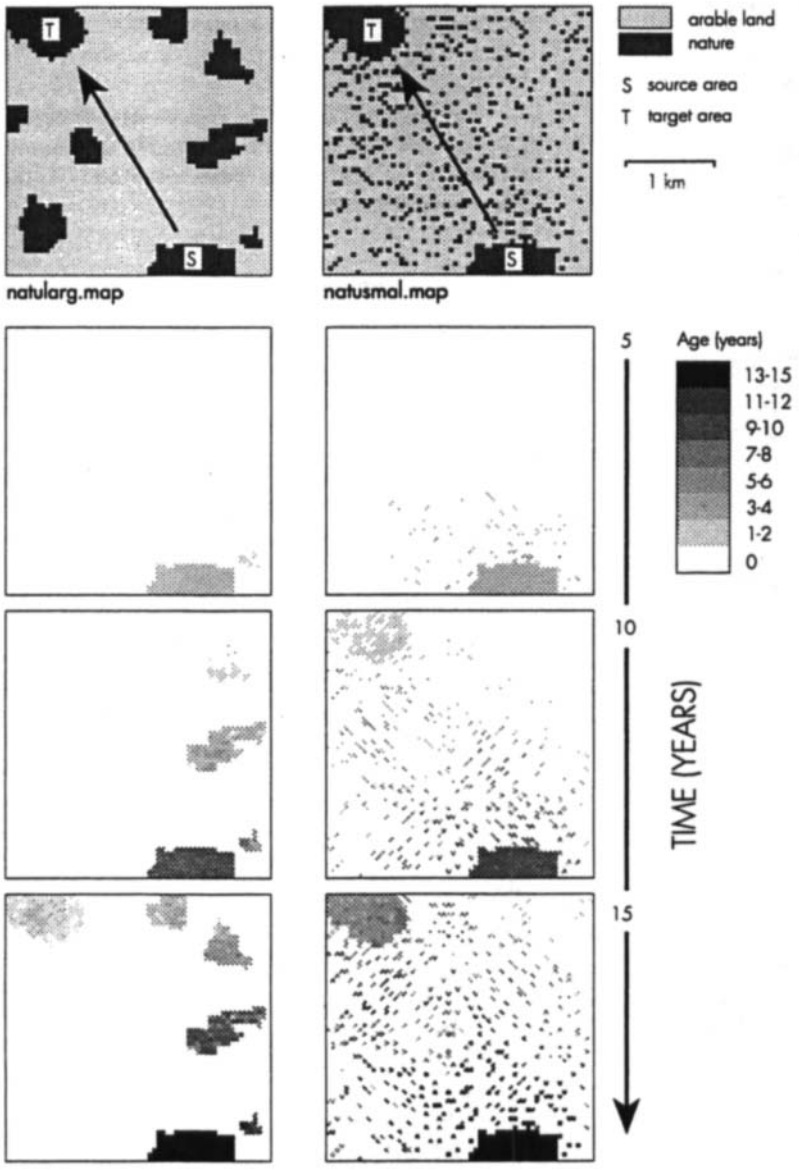

Figure 2. Case studies illustrating the seed dispersion model. Two scenarios are shown (left and right) with the maps natularg.map and natusmal.map linked to Nature in the binding section of the script, respectively. Bottom: Age of plants after 5, 10 and 15 years for both scenarios: reported AgeVeg maps.

done with ordinary operators that have the same syntax as the other (Cartographic Modelling) operators of the PCRaster language.

The previous section showed that the *report* keyword is used to store the results of an operation in the database. If the result of the operation is a map, a stack of map layers is stored, where each map represents one time step. Model results can also be stored in a *timeseries* file. For each time step such a file may contain sampled values of one or more cells on a map or one or more statistical values of all cells or cells in a specified area on a map. For instance, for each time step, the following operation writes to the *timeseries* OldVegAreaTSS the area of the map covered by plants older than 5 years:

```
report OldVegAreaTSS = maparea(AgeVeg gt 5);
```

Retrieval of dynamic data from the database is also simple. It is accomplished with one of the *timeinput* operators. A *timeinput* operator imports specified data to the dynamic section at each time step, independent of the results of the previous time step: it defines a map that is assigned a different set of cell values for each time step. These values can be read from a *timeseries* file or from a stack of maps in the database. For instance, one might assume that the distribution of potentially colonizable areas changes over the period modelled by Table 1. This scenario would be modelled by adding the following operation at the start of the dynamic section:

```
Nature = timeinput(NatureStack);
```

For each time step the operation assigns a boolean map from a stack of maps called NatureStack with the distribution of potentially colonizable areas at the timestep under consideration.

Table 1. Sequential Dynamic Modelling script for simulation of seed dispersal and plant growth. Remarks, ignored by the model, are preceded by a # sign. The first scenario shown in Figure 2 has been calculated with this script; the second scenario has been calculated by linking a different map (natusmal.map) to 'Nature' in the binding section.

```
# model for simulation of plant seed dispersal and plant growth
# over 15 years, with 15 time steps of 1 year

binding
InitialPlants = initpla.map;    # boolean map with distribution of
                                # plants at start of model run
Nature = natularg.map;          # boolean map with nature areas
Range = 350;                    # distance [m] with Probability = 0.1
Probability = probab;           # reported maps with probability that
                                # seed reaches cell and
                                # comes out next year
AgeVeg = ageplants;             # reported maps with
                                # age of vegetation

areamap
placlone.map;                   # clone map of study area

timer
1 15 1;                         # start time, end time, time slice

initial
# distribution of plants at start of model run
Plants = InitialPlants;
# distribution of new plants at start of model run
NewPlants = boolean(0);
# age of vegetation at start of model run
AgeVeg = 0;

dynamic
# distribution of plants in summertime
Plants = (Plants or NewPlants) and Nature;
# distance from nearest cell with plants [m]
Distance = spread(Plants,0,1);
# probability that seed reaches cell and comes out next year
report Probability = 0.1**(Distance/Range);
# cells where plants will grow next year as a result of seed
# dispersal
NewPlants = Probability gt uniformfield(1);
# age of plants
report AgeVeg = if(Plants then AgeVeg+1 else AgeVeg);
```

System implementation

All features of the PCRaster modelling language are implemented in a single command line program named CALC. CALC can be used in any of the following use modes:

1 Computation of a single map. This mode is often used for *ad hoc* analysis. For example: creating a buffer of 200 m around the areas of potential plant colonization:

```
CALC buf.map = spread(natularg.map, 0, 1) lt 200
```

2 Execution of a static Cartographic Model written in an ASCII formatted script file containing a sequence of operations. For example:

```
CALC -f script.mod
```

3 Execution of an ASCII formatted Dynamic Modelling script with the sections *binding, areamap, timer, initial* and *dynamic*. For example:

```
CALC -f plant.mod
```

In all these cases CALC checks the model code for syntax and data type conflicts and then starts the execution. It is important to note that the language is not a just another macro language. A PCRaster Dynamic Modelling script is not a list of actions that is sent to separate autonomic modules of the GIS. Instead, a single program (CALC) reads a script entirely, checks it for errors and then executes it. This approach yields better

error detection facilities and faster execution of the model than macro languages. A disadvantage, compared with macro languages or ordinary programming languages, is that the user cannot easily add extra functionality. Currently this disadvantage is partially avoided by making CALC, and all other modules of PCRaster, command line programs that can be combined with other software in UNIX shell scripts or MS-DOS batch files. For a detailed discussion of the implementation of CALC, see Wesseling et al (1996).

In addition to standard GIS functionality, such as data import/export, hardcopy output and graphical display, the PCRaster package includes two extra packages which are linked following the tight coupling approach. These are ADAM (Heuvelink 1993; Wesseling and Heuvelink 1993) for estimation of error propagation in GIS operations and GSTAT (Pebesma 1996a, 1996b) for *geostatistical interpolation (kriging), conditional* simulation and random field generation.

PCRaster is operational on both UNIX and MS-DOS (80486 or better) platforms with the graphical modules working under X11 and SuperVGA respectively. User manuals are provided on paper or as help functions from the command line.

Versions for other operating system environments (such as MS-WINDOWS 3.11) have not been implemented at present because PCRaster requires a platform that can deal with large data sets and a mixture of command line and graphically event driven software in a robust manner.

Execution speed of display and modelling routines depends on data size and the PCRaster operators included. To give an idea, using a 486DX2 50 MHerz processor and 4 MByte memory, the model described in this paper (Table 1) is executed in 8 seconds and animated without any further data exchange. In combination with the powerful and open script language, this allows the user to build this kind of model interactively.

A student version limited to raster maps not exceeding 60 x 60 cells will be provided on Internet in the spring of 1996. Professional versions will be available on a semi-commercial basis.

Conclusions

This paper introduces the PCRaster Dynamic Modelling language, which is a specially developed programming language embedded in a GIS. The key features are the ability to create dynamic process models with full iteration that include a strong data type checking mechanism to reveal possible errors. The language presented here has been developed at the University of Utrecht over a period of five years. During this period early prototypes of the language have been used for numerous environmental models. Published examples are LISEM, a physically-based hydrological and soil erosion model on a catchment scale (De Roo et al 1994), RHINEFLOW, a water balance model for the river Rhine (Van Deursen and Kwadijk 1993) and Calluna, an ecological model for heathland dynamics (Van Deursen and Heil 1993). Most of the current applications put a strong emphasis on environmental modelling and especially hydrological surface routing. Therefore, the evolution of the language has resulted in a rich set of global functions involving drainage networks. The example presented in this paper shows that the language can also be applied in the field of ecology. Demonstrations of the use of vector fields and associated processes, such as diffusion and advection, are not yet available. Further research and development are needed to explore the effectiveness of the Dynamic Modelling language in areas such as groundwater modelling, where such processes are used extensively.

References

Berry J K 1987 Fundamental operations in computer-assisted map analysis. *International Journal of Geographic Information Systems* 1 (2): 119–36

Biesheuvel A and Hemker C J 1993 Ground water modelling and GIS: Integrating MICRO-FEM and ILWIS. In Kovar K and Nachnebel H P (eds) *Application of Geographic Information Systems in Hydrology and Water Resources Management.* Wallingford, IAHS Publication No. 211: 289–96

Burrough P A [forthcoming] Opportunities and limitations of GIS-based modeling of solute transport at the regional scale. In Corwin D L and Loague K (eds) *Applications of GIS to the Modeling of Non-Point Source Pollutants in the Vadose Zone.* Madison, Soil Science Society of America Special Publication

De Roo A, Hazelhoff L and Burrough P A 1989 Soil erosion modelling using ANSWERS and geographical information systems. *Earth Surface Processes and Landforms* 14 (6–7): 517–32

De Roo A P J, Wesseling C G, Creemers N H D T, Offermans R J E, Ritsema C J and Van Oostindie K 1994 LISEM: A physically-based hydrological and soil erosion model incorporated in a GIS. In J Hart et al (eds) *Proceedings EGIS/MARIS '94.* Utrecht, EGIS Foundation: 207–216

Fedra K 1996 Distributed models and embedded GIS: Strategies and case studies of integration. In Goodchild M F, Steyaert L T, Parks B O, Johnston C, Maidment D R, Crane M and Glendinning S (eds) *GIS and Environmental Modeling: Progress and Research Issues.* Fort Collins, GIS World, Inc.: 413–18

Goodchild M F, Parks B O and Steyaert L T 1993 *Environmental Modeling with GIS.* New York, Oxford University Press

Goodchild M F, Steyaert L T, Parks B O, Crane M P, Johnston C A, Maidment D R and Glendinning S 1996 *GIS and Environmental Modeling: Progress and Research Issues.* Fort Collins, GIS World, Inc.

Harper J L 1977 *The population biology of plants.* New York, Academic Press

Heuvelink G B M 1993 *Error propagation in quantitative spatial modelling applications in geographical information systems.* PhD dissertation, University of Utrecht

Jenson S K and Domingue J O 1988 Extracting topographic structure from digital elevation data for geographic information system analysis. *Photogrammetric Engineering and Remote Sensing* 54 (11): 1593–1600

McClanahan T R 1986 Seed dispersal from vegetation islands. *Ecological Modelling* 43 (4): 301–9

Moore I D, Turner A K, Wilson J P, Jenson K and Band L E 1993 GIS and the land surface – subsurface process modelling. In Goodchild M F, Parks B O and Steyaert L T (eds) *Environmental modelling with GIS.* New York, Oxford University Press: 197–230

Pebesma E J 1996a *GSTAT, geostatistical modelling, prediction and simulation.* Accessible on the Internet at http://www.frw.ruu.nl/pcraster.html

Pebesma E J 1996b *Mapping ground-water quality in the Netherlands.* PhD dissertation, University of Utrecht

Romanowicz R, Beven K, Freer J and Moore R 1993 TOPMODEL as an application module within GIS. In Kovar K and Nachnebel H P (eds) *Application of Geographic Information Systems in Hydrology and Water Resources Management.* Wallingford, IAHS Publication No. 211: 211–23

Silvertown J S, Holtier J, Johnson J and Dale P 1992 Cellular automaton models of interspecific competition for space – the effect of pattern on process. *Journal of Ecology* 80 (3): 527–54

Tomlin C D 1990 *Geographic information systems and cartographic modelling.* Englewood Cliffs, Prentice Hall

Van Deursen W P A 1995 *Geographical information systems and dynamic models: Development and application of a prototype spatial modelling language.* PhD dissertation, University of Utrecht

Van Deursen W P A and Heil G W 1993 Analysis of heathland dynamics using a spatial distributed GIS model. *Scripta Botanica* 21 (1): 17–28

Van Deursen W P A and Kwadijk J C J 1993 RHINEFLOW: An integrated GIS water balance model for the river Rhine. In Kovar K and Nachnebel H P (eds) *Application of Geographic Information Systems in Hydrology and Water Resources Management.* Wallingford, IAHS Publication No. 211: 507–18

Van Deursen W P A and Wesseling C G 1995 *PCRaster Software.* Department of Physical Geography, University of Utrecht. Accessible on the Internet at http://www.frw.ruu.nl/pcraster.html

Wesseling C G, Van Deursen W P A and Burrough P A 1996 A spatial modelling language that unifies dynamic environmental models and GIS. *Proceedings Third International Conference on Integrating GIS and Environmental Modeling, Santa Fe.* Santa Barbara, University of California National Center for Geographic Information and Analysis

Wesseling C G and Heuvelink G B M 1993 *ADAM, an error propagation tool for geographical information systems. Second Beta Version.* Department of Physical Geography, University of Utrecht. Accessible on the Internet at http://www.frw.ruu.nl/pcraster.html