Interacting Agents with Memory in Virtual Ecosystems

Bedřich Beneš
Dep. of Computer Science
ITESM CCM México D.F.
Bedrich.Benes@itesm.mx

Javier Abdul Cordóba
Dep. of Computer Science
ITESM CEM México D.F.
abdul@itesm.mx

Juan Miguel Soto
Dep. of Computer Science
ITESM CEM México D.F.
mguerrer@itesm.mx

ABSTRACT

An agent-based modeling of virtual ecosystems is presented. A virtual ecosystem develops by plant competition according to biologically inspired rules and tends to reach stability. Virtual agents enter the ecosystem and perform actions that favor certain plant species and cause system instabilities. Agents liberate space for some plant species by eliminating the others, they take out old plants if an area is overcrowded, they seed plants, water them, etc. Agents synchronize by message passing to cover the area efficiently and not to interpenetrate their areas of influence. Each agent has a local memory of pending tasks. When a memory overflow arises the agent divide the pending tasks among the nearest agents.

Keywords

Virtual plant ecosystem, visual simulation, agents with memory, procedural modeling

1. Introduction

There are many ways to model shape of a 3D object or of an entire scene. One of the most interesting areas belongs to procedural modeling, where a shape of an object is defined by some action or piece of code. The classical approaches include fractal-based modeling, particle systems, grammar-based techniques, etc. The procedural techniques are more or less sophisticated and with increasing computer powers new things are becoming possible. The new approach introduced in this paper is based on autonomous agents (automata) that interact with 3D objects.

Agents are procedurally driven automata that can perform certain actions on the ecosystem and can communicate. Every agent has its local memory that serves for postponing the work that cannot be done at the moment. The stack of tasks is not infinite and some work could possibly be forgotten if a memory overflow occurs. To avoid this the agents can communicate and share the work.

After the previous work description the ecosystems modeling and development are described. Section 3 deals with ecosystem rendering and the following section describes their development. Section 5 introduces the agents and the next Section 6 deals with the implementation. The last two sections describe results and conclusions.

Journal of WSCG, Vol.11, No.1., ISSN 1213-6972
WSCG'2003, February 3-7, 2003, Plzen, Czech Republic.

2. Previous Work

The first procedural models of plants were deeply rooted in fractals [Oppen86] and their usage in computer graphics were very limited. The formal specification of these techniques - Lindenmayer systems (L-systems) was introduced by Lindenmayer [Linde68], later used by Smith [Smith84], and nowadays are being extensively improved by Prusinkiewicz and his collaborates (see the book [Prusi90] and the tutorial [Jones00]).

Plants were first simulated as closed systems with no interaction with their environment [Aono84, Smith84, Oppen86]. These models do not reflect real conditions that is an important drawback. The light influence of the plant shape was mentioned for example in [Arvo88, Měch96, Prusi93]. The irradiance evaluation is complicated and some simplifications could be used. We have suggested the use of hardware-assisted rendering to calculate the direct illumination of plants leaves in [Beneš96]. So-called Open L-systems introduced by Měch *et al.* [Měch96] presents a formal specification of plant-environment interactions.

Deussen *et al.* [Deuss98] used two level simulation to create a visual model of a virtual plant ecosystem. Our paper is an extension of this approach. Plants compete for space by comparing their ecological neighborhoods. If two neighborhoods interpenetrate the weaker plants is removed. Plants also have their 3D representation that is used to render photo realistic images.

Another virtual ecosystems simulation was recently described by Lane and Prusinkiewicz [Lane02]. Two approaches are described here, the local-to-global and the global-to-local. In the first one the entities are planted, develop, and interact that leads to a certain plant distribution. The rules for competition govern the resulting model. This is a typical *artificial life* approach where the local rules lead to an emergent phenomenon. The global-to-local approach involves user's assistance that defines the initial plant distribution. The theoretical framework introduced in this paper is an extension of L-systems to *multiset L-systems*.

The problem of ecosystem stability was addressed in [Beneš02]. An artificial environmental feedback assures the ecosystems to always grow to a stable state. The system reaches stability even after wrong initialization or after an ecological catastrophe. This approach is explained in Section 4.

We focus on the non-plant agents interacting with ecosystems in this paper. Probably the first approach was a simulation of a bug eliminating some parts of plants in [Prusi95]. Traumatic reiteration was used here. Buds that are eliminated cease production of a hormone that inhibits the other buds from growing. The lack of this hormone is propagated down in the plant structure and causes the closest bud to wake up from its dormant state. The newly growing bud starts producing the hormone that stops the other buds.

An another paper dealing with more elaborated models of insect attacking plants has been recently published [Hanan02]. The paper presents examples of a formal specification by means of L-systems of insects interacting with plant or plants in different ways ranging from a single insect foraging on a plant to insects flying in an ecosystem. The paper also discusses plant response to a damage, behavior modeling, insect perception modeling, etc.

3. Geometry and Rendering

We use two geometrical representations of a plant. The first is necessary to simulate plant competitions whereas the other is required to display the scene by photo realistic rendering. For the first case a plant is represented by its 2D position and the radius of influence called the ecological neighborhood. This is displayed as a set of circles (see in Figure 1).

To get photorealistic images we save scene time samples as scene description files for the Persistence of Vision ray-tracer. Here, the set of 3D geometric primitives represents the scene objects. Bézier surfaces model flower petals and grass blades, generalized cylinders are used for stems, line segments represent tiny leaves, spherical cap is used as a model of the head of the english daisy, etc.

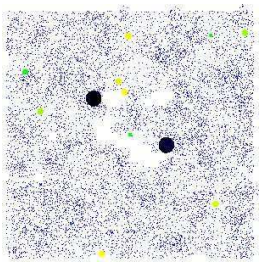

Figure 1: An ecosystem displayed as a set of circles

The scenes are a bit excessive, for example a scene with hundred thousands plants takes more than fifty megabytes of the disc space and requires more than 1GB of memory to be rendered. To diminish these requirements we use instancing. Plants are quantized to groups that are similarly old. For example each plant from the age between zero and fifty days has one representative in the scene. Plants are instanced by transformations in real scene that leads to significant memory saves. Only translations and rotations are used. The visual quality of each scene depends on the number of instances used. Increasing number of instance leads to better quality but increases memory requirements. We have noticed that about hundred representatives that are instanced in scenes with hundred thousands plants do not disturb visually at all. The plant repetition is noticeable if there are less than twenty different objects for the same scenes. It would be interesting to see how the number of instances influences the visual quality of an image and the amount of disc space. An example of a close-up of a scene consisting of forty thousand objects is on Figure 2.

4. Ecosystems

We use the local-to-global approach with an environmental feedback to assure stability of simulations. The simulation algorithm described in [Lane02] and extended by the environmental feedback in [Beneš02] is briefly describe it here.

Figure 2: A model of a virtual ecosystem ray-traced using instancing

An ecosystem is represented as a homogeneous planar continuous area. A scene is described and plants are put to their initial conditions. Plants develop according to the local rules and compete for resources. The competition is simulated by collision detection of the circular ecological neighborhoods of plants. If two circles interpenetrate, the collision is detected, and the weaker plant is eliminated.

Competitions could be classified into two important classes. Plants are competing between the same species and between the others. In all cases so called viability function is evaluated. The function favors plants in the middle of their age. In other words small and fragile as well as old and weak plants have smaller chance to survive. If two plants of the same specie compete the viability function value depends merely on their age. The situation is different for plants of different species.

The environmental feedback simulates the phenomenon of running-off resources if there are more plants of the same plant specie at the same place. This makes them weak and ben-

efits the other plant species in competitions. We measure the average area of all ecological neighborhoods and scale the viability functions correspondingly (see [Beneš02] for details). It means that the winner of the competition of two species depends on their age and frequency. For example if the last representative of one plant specie meet with grass that is overcrowding the environment, the grass has a very small (but non-zero) chance to win the competition. We have never seen any plant specie to extinct in our simulations. The incorrectly initialized ecosystem reaches stability fast and plants just grow and change their places. An example of a top-view of a stable ecosystem is showed in Figure 3. The initially incorrectly initialized ecosystem reaches stability in approximately 100 days and keeps till the end of simulation. The average number of daisies was 50 and there are ten thousands grass blades. The size of the file was less than 2.5MB.

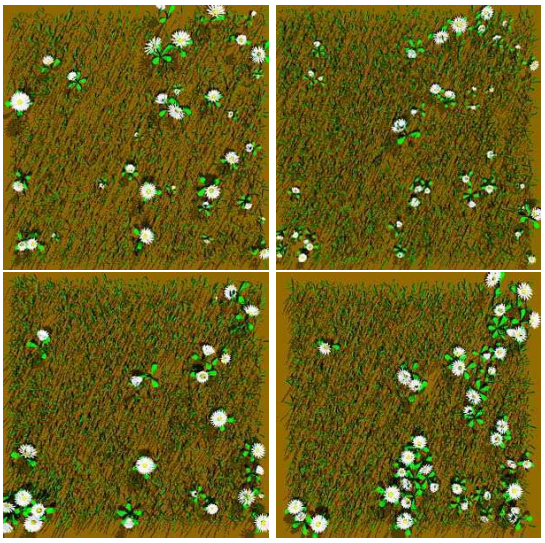

Figure 3: Randomly initialized ecosystem (top left) reaches the stability after one year (top right) and keeps it (down). The number of plants does not vary very much, but they change their positions

5. Agents

This paper extends the concept of one currently submitted paper [Beneš03], where the agents that enters an ecosystem interact in a limited way and do not have memory. Agents entering the field wander by a random walk till they reach a plant to be eliminated. They move to this plant and perform the action. If there is another one close they move to this plant etc. This can cause some plants to be skipped as shown in Figure 4. An agent has a position counter, computes an accumulated average of the skipped plant positions, and broadcasts this information to all agents. If there is an agent with no work it simply takes this direction, because there is certainly something to do. This approach gives efficient results but the missing memory causes agents to wander randomly on the field without any specific purpose.

The new approach introduced here is the agent's memory and the communication. Before describing it, let's mention the way agents are implemented and how they behave.

5.1. Agent Description

An agent is represented by the actual position, area of influence, and the direction it moves. At the beginning the agents appear at the edge of the ecosystem and enter it. For simplicity, the agents will be described as circles with an arrow indicating the direction of their motion. Every agent has the FIFO (first in - first out) memory. The memory has a finite depth, we use ten elements in our implementation.

5.2. Agent Behavior

An agent enters the ecosystem and starts its work that includes eliminating weeds, eliminating plants that are located at incorrect position, eliminating old plants etc. An agent has its initial direction of the motion, that is perpendicular to the edge of the field, and walks in this direction till some task (plant) appears in its area of interest. Then it moves there and performs the action. If a new task appears, it moves there, etc. If there are two

tasks to do, the agent will move to the closest one.

This can cause some tasks to be skipped as shown in Figure 4. The agent has the plants A and B in its region of interest. After moving to the closest one, *i.e.*, A, the plant B leaves the radius and the plant C enters. In the next step the agent will move to the plant C and the plant B would be forgotten. In this case the plant B will be put onto the stack. At the moment there is no plant in the radius of the agent it performs backtracking *i.e.,* takes the position from the top of the stack and moves there. This approach leads to more efficient task distribution than a random walk.

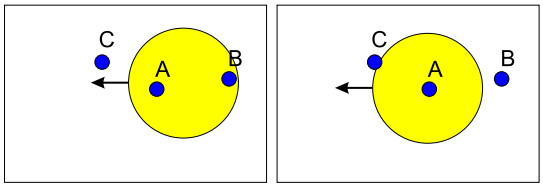

Figure 4: (left) Moving to the plant A causes the plant B to leave the agent's region of interest. The position of the plant B is put on the stack

5.3. Communication

There are two problematic cases. The first is when two agents meet and their radii interpenetrate. It corresponds to the situation when two agents work in the same, or almost the same, area. We solve this situation in the following way. We compare the depth of the stack of both agents and the numbers of plants to be served that each agents should do. The agent that has more things to do leaves this area moving to the position of the plant that is on the top of its stack. This keeps the agent with less work inside the area that has some plants.

Another critical case is when an agent detects the stack overflow. It means there is too much work to do for one agent. In this case the agent calls the closest colleague and they perform synchronization of their tasks *i.e.,* negotiation. The agent with the stack overflow asks the colleague the depth of its stack. Then it sends the half of this length of the plant positions from the *bottom* of its own stack. These plants are too far and it is worth to visit the plants from the top of the stack first.

5.4. Lifetime

Agents enter the ecosystem at the same time once a week and work for several hours. Since different task has different duration we convert the time it to the trajectory that is passed by each agent. We suppose that the agent's motion is done with speed 0.5m/sec and work on one plant takes ten seconds. This work is converted to the same units, *i.e.,* to the passed trajectory. Every agent has its local counter of distance passed. When the length that corresponds to the assigned time is exceeded the agent leaves the field.

6. Implementation

The entire system in implemented in C++ with support of OpenGL and runs on Win32 and UNIX. The output is either displayed using OpenGL or ray-traced.

The critical point of the program is the detection of the collision or proximity of one circle to another one. First, we used k-d trees to perform this task efficiently. Later we noticed that the area is totally filled quite fast, that implies the 2D space is filled homogeneously. This allows us to use simple regular subdivision of the space. We divide the 2D area into squares and detect plants that are inside each of them. It is important to notice that one plant can belong to more than one square. This complicates the tests a bit. We maintain two lists of plants for each square, the list of those that reside there, and of the ones that just enter by their ecological neighborhoods.

The initialization is done in a jittering-like way. We put n_i plants of the i-th specie there. Since the area is divided to m squares, we just fill randomly each square by n_i/m plants.

The program runs sufficiently fast. Simulation of one year development of an area of $100\mathrm{m}^2$ that includes 250 000 plants with the time step four days takes less than three minutes on IBM PC 1GHz.

The scene description files for the ray-tracer are up to 20MB depending on the number of instances. The rendering times on the same computer were up to 20 minutes.

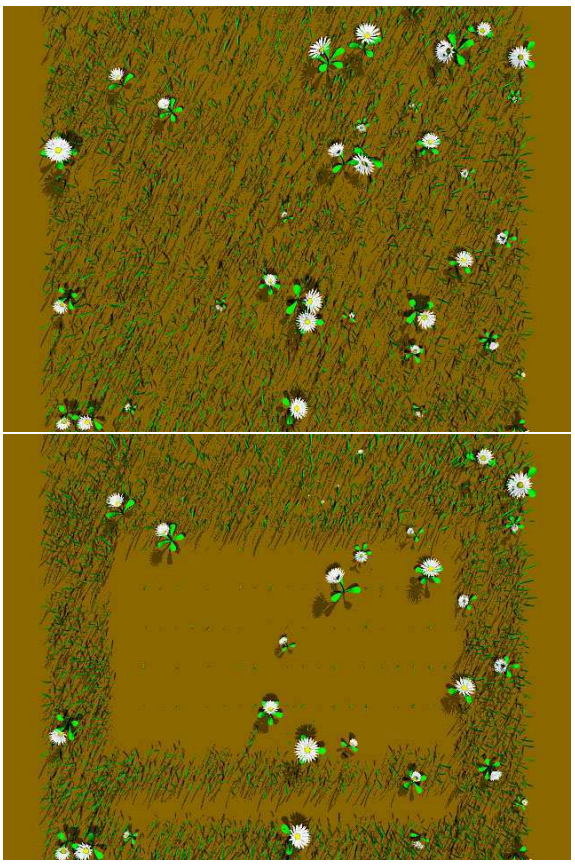

Figure 5: A stable area that contains grass and daisies (up) is cleaned by seven virtual agents from grass and prepared for seeding the daisies

7. Results

Some complex scenes can be obtained only by the development simulation. A typical case is a well-treated garden. We aim to show that this can be simulated really efficiently, fast, and simply by the technique introduced here.

The user seeds some plants and assigns rules to the agents. The rest is the emergent phenomena of the simulation. To demonstrate this we have created the following example.

A lawn with english daisies (see in Figure 5) is decided to contain daisy beds. The lawn is left to grow for forty days to reach stability. Then the agents repeatedly enter and eliminate the grass from the assigned area. The daisies are protected and just the grass is eliminated. Now, in the day 30, the area is prepared to cultivate daisies.

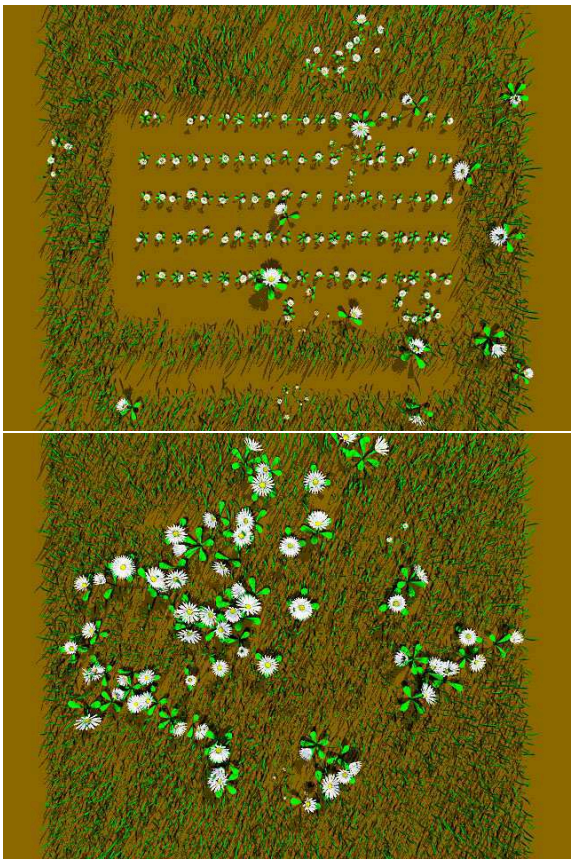

Figure 6: Daisy beds are cultivated and protected against the grass that invades their space from all sides (up). The field is abandoned (down) and after one year the daisy bed disappears

Agents plant daisies and then repeatedly eliminate grass as can be seen in the top image in Figure 6. This image is taken from the day 200 of the simulation. After one more year of the cultivation the garden is abandoned. First the daisies start to grow very fast

to all sides and the grass also invades their space. After one year the daisy bed has disappeared, as can be seen on the last image.

The simulation area was $5m^2$ and the average number of plants was 30 000. The total time of the simulation was less than two minutes.

8. Conclusions

A procedural modeling based on virtual agents is presented. The agents are automata with finite memory that can move over an ecosystem, perform some actions, and communicate. The agents can share their tasks efficiently and avoid collisions and duplication of work. As they move on a virtual field they keep a track of the work that was impossible to do and in a critical moments they can share it among them. Similar way allows to solve collisions when some agents meet at the same place.

There are many possible applications of this technique, but the apparent one is creating a realistic scenes by procedural modeling. Precise, maybe user assisted, definition of the agents and their tasks together with the ecosystem simulation could allow an efficient and realistic simulation of scenes that are difficult or impossible to model by other techniques.

9. REFERENCES

[Aono84] M. Aono and T.L. Kunii. Botanical Tree Image Generation. *IEEE Computer Graphics and Applications*, 4(5):10–34, 1984.

[Arvo88] J. Arvo and D. Kirk. Modeling Plants with Environment-Sensitive Automata. In *Proceedings of Ausgraph'88*, pages 27–33, 1988.

[Beneš96] B. Beneš. An Efficient Estimation of Light in Simulation of Plant Development. In *Computer Animation and Simulation'96*, Springer Computer Science, pages 153–165. Springer–Verlag Wien New York, 1996.

[Beneš02] B. Beneš. A Stable Modeling of Large Plant Environments. In *Proceedings of the ICCVG'02*, pages 94–101. Association for Image Processing, 2002.

[Beneš03] B. Beneš, J.M. Soto, and J.A. Cordoba. Using Procedural Agents in Virtual Plant Ecosystems. *submitted to the TP CG'03*, IEEE, 2003.

[Deuss98] O. Deussen, P. Hanrahan, B. Lintermann, R. Měch, M. Pharr, and P. Prusinkiewicz. Realistic Modeling and Rendering of Plant Ecosystems. In *Proceedings of SIGGRAPH'98*, Annual Conference Series 1998, pages 275–286, 1998.

[Hanan02] J. Hanan, P. Prusinkiewicz, M. Zalucki, and D. Skirvin. Simulation of Insect Movement with Respect to Plant Architecture and Morphogenesis. *Computers and Electronics in Agriculture*, to appear.

[Jones00] H. Jones. Modelling of Growing Natural Forms. In *Eurographics'00 Tutorials*. Springer–Verlag, 2000.

[Lane02] B. Lane and P. Prusinkiewicz. Generating Spatial Distribution for Multilevel Models of Plant Communities. In *Proceedings of Graphics Interface'02*, pages 69–80, 2002.

[Linde68] A. Lindenmayer. Mathematical Models for Cellular Interaction in Development. *Journal of Theoretical Biology*, Parts I and II(18):280–315, 1968.

[Měch96] R. Měch and P. Prusinkiewicz. Visual Models of Plants Interacting With Their Environment. In *Proceedings of SIGGRAPH '96*, volume 30(4) of *Annual Conference Series 1996*, pages 397–410, 1996.

[Oppen86] P. Oppenheimer. Real Time Design and Animation of Fractal Plants and Trees. In *Proceedings of SIGGRAPH '86*, volume 20(4) of *Annual Conference Series 1986*, pages 55–64, 1986.

[Prusi90] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer–Verlag, New York, 1990. With J.S.Hanan, F.D. Fracchia, D.R.Fowler, M.J.de Boer, and L.Mercer.

[Prusi93] P. Prusinkiewicz, J. Hanan, M. Hammel, and R. Měch. L-systems: from the Theory to Visual Models of Plants. *Machine Graphics and Vision*, 2(4):12–22, 1993.

[Prusi95] P. Prusinkiewicz, M. James, R. Měch, and J. Hanan. The Artificial Life of Plants. In *SIGGRAPH '95 Course Notes*, volume 7, pages 1-1–1-38. ACM SIGGRAPH, 1995.

[Smith84] A.R. Smith. Plants, Fractals and Formal Languages. In *Proceedings of SIGGRAPH '84*, volume 18(3) of *Annual Conference Series*, pages 1–10, 1984.