Modeling Virtual Gardens by Autonomous Procedural Agents

Bedřich Beneš,
ITESM Campus Ciudad de México
Bedrich.Benes@itesm.mx

Javier Abdul Cordóba,
Juan Miguel Soto
ITESM Campus Estado de México
{abdul|mguerrer}@itesm.mx

Abstract

Cultivated virtual plant ecosystems modeling by means of simple memoryless procedural agents is presented. The ecosystem growth is a dynamic process with a tendency to chaos. An agent can move in the ecosystem and perform certain actions defined by users in the agent description file. Agents can seed new plants, pull out weeds, water plants, and communicate by message passing to distribute their tasks. An example of a successive garden cultivation in a wild ecosystem is presented. Agents first eliminate weeds, prepare space for and lay sidewalks, plant garden flowers, and protect their development.

The main contribution of this paper is to show that autonomous agents can be used as a tool for user assisted procedural modeling of highly complex scenes.

1. Introduction and Previous Work

Visual simulation and procedural modeling of plants have a long history and an impressive progress in computer graphics. Early models were based on fractals such as [14] and later developed through formal specification to result in Lindenmayer systems (L-systems). L-systems were introduced in [12], first used for modeling of geometrical plant-like structures (called *graftals*) in computer graphics in [19], and later were extensively improved by Prusinkiewicz and his collaborators (see the book [17] and the tutorial [10] for an overview).

An important factor influencing plant development, and therefore its structure, is illumination as mentioned for example in [1, 13, 15].

Greene [7] simulated climbing plants and used approaches similar to those described by Arvo and Kirk [1]. The plant growth is simulated as a random walk and the best-illuminated position on the surface of a supporting tool is calculated. The plants tend to grow on the surface of another object resulting in a "virtual ivy" or similar climbing plant species. These techniques were recently reused in [4]. In these works the plant structure itself is modeled as a result of a work of a set of virtual automata. In this paper, we use external agents to perform tasks that influence the plant development.

Attempts to simulate interaction with an environment have resulted in an extension of L-systems to so called Open L-systems [13]. The Open L-systems allows a formal specification of an interaction of a plant with its environment and the plant itself.

Visual simulations of plant ecosystems were attempted by Deussen *et al.* [6] who use a two level simulation. On the low-level plants interact just by simple competition of their ecological neighborhoods represented as circles. If two circles interpenetrate there is a collision and the weaker plant is eliminated. On the higher level the plants are represented as 3D models that are rendered using some simplifications but resulting in high quality photorealistic images.

Lane and Prusinkiewicz introduce two approaches in [11], the local-to-global and the global-to-local. In the first one the entities are planted, grow, and interact in a way that leads to a certain plant distribution. If the rules for competition and the implementation are correct the resulting model is visually plausible and therefore realistic. The other approach, the global-to-local, could be also called a computer assisted ecosystem simulation. The user specifies a plant distribution as a gray-scale image and the system generates the corresponding 3D model. The theoretical framework introduced in this paper is another extension of the concept of L-systems - *multiset L-systems*.

Probably the first approach to simulation of plants interacting with another biological objects was a simulation of a bug eating some parts of plants in [16]. The phenomenon of a traumatic reiteration (a change in leadership) was used here. The buds that are eliminated by a bug cease production of a hormone that inhibits the other buds from growing. The lack of this hormone is propagated down the plant structure and causes the nearest bud to wake from its dormant state. The newly growing bud starts producing the hormone that stops the other buds. This is simulated by sending signals by the just eliminated bud down through

Proceedings of the Theory and Practice of Computer Graphics (TPCG'03)
0-7695-1942-3/03 $17.00 © 2003 IEEE

the plant structure. The closest bud captures the signal and stops its propagation.

Another paper dealing with more elaborated models of insects attacking plants has been recently published [8]. The paper presents examples of a formal specification by means of L-systems of insects interacting with a plant or plants in different ways. This ranges from a single insect foraging on an individual plant to insects flying in an ecosystem. The paper also discusses plant response to damage, behavior modeling, insect perception modeling.

Agents operating on virtual ecosystems were recently studied in [3, 5]. The previous work shows that agents can be used as efficient tools for modeling phenomena that are difficult to achieve without them, for example agent-farmers that protect a predefined area (garden) against weeds [5]. The process of ecosystem development simulation itself is slow and complicated and sophisticated agents slow down these calculations even more. The main contribution of this paper is an efficient interaction calculation and simple interaction between agents and plants allowing for fast simulation with a high number of agents.

This paper has the following structure. In the next section individual plant models and growth simulations are described. Section 3 deals with ecosystems description and the next section describes in depth agents and their functionality. A brief note about rendering is given in Section 5 and the last two sections describe results and conclusions.

2. Plant Modeling and Growth Simulation

An ecosystem is a set of individual plants in different stages of development. Our model deals with several plant species – we have models of english daisy (*bellis perennis*), wheat, three different kinds of grass, some bush-like plant, campanula (blue bell), yellow tulip, and different procedurally generated trees obtained by the strands model of Holton [9]. Examples of the plants are illustrated in Figure 1. Some of the plants were generated by the Plant Studio software (www.kurtz-fernhout.com).

Geometrical models are represented on two logical levels. At the low level the model of a plant is represented as a set of basic geometric primitives; Bézier surfaces are used to model flowers petals and grass blades, generalized cylinders are models of stems, line segments model tiny leaves, a spherical cap is used as a model of the head of the english daisy, and so on.

At the higher level the plant is represented by its position and the *ecological neighborhood* [6] that is a radius within which the plant resides with no other plant allowed inside. This functional representation is used for plant-plant and plan-agent interactions.

When a plant is initialized the day of its possible death is also determined. It corresponds to a perfect, non-influenced development in ideal conditions. In classical ecosystem simulations the real fate of a plant depends on competition with other plants and external resources, the new approach introduced in this paper is the influence of procedural agents. Both aspects will be described later in the paper.

Figure 1. Examples of plants used in the ecosystem model

3. Ecosystems

An ecosystem is a virtual environment represented as a 2D continuous area inhabited by virtual plants.

The local-to-global approach to model ecosystem development is used [11]. An ecosystem is described, plants are seeded, grow by competing for resources and space, and reproduce. The emergent phenomenon of this artificial life simulation is the spatial plant distribution.

The ecosystem initialization can be done in several ways. We can do it interactively using some drawing program, where the color corresponds to the spatial distribution of the plants [11], or we can plant some seeds at random and let the system grow to a stable state, or we can program the agents to run over the ecosystem and plant seeds.

The system is controlled by discrete time-steps that approximate continuous time. We use the time-step $\Delta t = 1/2$ day, determined by the fastest growing plant in the ecosystem, in this case the wheat. During one time-step the following actions are performed:

1. plants that should go to seed give birth to new seeds,
2. plants out of the ecosystem are eliminated,

3. colliding plants are detected,
4. old plants, as well as the plants that have lost in competitions, are eliminated,
5. agents perform their scheduled actions,
6. all plants grow, and
7. the time-step is increased $t = t + \Delta t$.

The time-step of the ecosystem growth simulation, a half-day, is relatively long for the agents that usually work in a few hours, in other words plants do not grow faster than agents work. We suppose the ecosystem does not develop during the agents' work so we run it as an independent process after the ecosystem simulation step.

The description of the ecosystem development simulation, as presented in this section, is sufficient for understanding of the following sections. We refer readers to [2, 6, 11] for more details of virtual ecosystems.

4. Agents

4.1. Visibility

An agent is represented as a point at the given position $[x, y]$, and the visibility range that is defined by the viewing direction, the visibility angle α, and the distance of visibility d (see Figure 2).

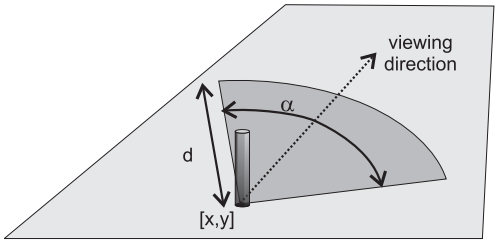

Figure 2. Agent's visibility range

We do not evaluate real objects an agent can see *i.e.,* we do not calculate occlusions that would be very complicated. We suppose that everything within the agent's visibility region can be perceived.

Each agent can be in one of two modes of seeing, the qualitative mode and the quantitative one. In the first an important closest individual plant is detected. This mode is used for example in pulling out weeds, searching for food, picking fruits, and so on.

The second mode is used for quantitative operations, like cutting grass, or seeding new plants in empty areas. An agent calculates the direction of maximal and minimal plant coverage in the following way. The visibility range is subdivided into n strips in polar coordinates corresponding to the angle α/n (we use $n = 7$). For each subarea the number of plants whose centers lie inside is counted. The minimum and maximum numbers are found and the corresponding directions are determined.

The most time demanding task is the determination of plant proximity. To perform it efficiently we use regular subdivision to squares of the ecosystem area. Each square has two lists of plant indices, those whose centers lie inside and those that interpenetrate the square. To determine the closest plants we evaluate the squares occupied by the agent and calculate the proximity of plants from these squares only. In the previous versions we have used k-d trees, but there are two important facts permitting us to use the regular subdivision; the area is completely covered by plants and no two ecological neighborhoods interpenetrate. A simulation of ten years development of a plant ecosystem of an average number of 20k plants with a time-step of 1/2 day on a 2GHz PC runs for less than one minute.

4.2. Actions

As mentioned in Section 4.1 an agent can be in one of two modes, qualitative - where an individual plant is detected, or quantitative - where the occupying number of plants is detected. One can easily imagine an extension of this concept to one logical level up – an agent could detect part of plants, for example old branches of trees could be eliminated, fruits could be harvested, and so on.

Unlike plants an agent can move over the ecosystem and can perform certain actions that include:

- planting a new plant (individual or multitudes),
- killing a plant (pulling out weeds or mowing, depending on the agent's operating mode),
- watering plants,
- placing insecticides, and
- communicating with other agents.

Virtual farmers are not presented in the ecosystem all the time. They can be run on a user's demand interactively or can be scheduled. For example in the spring or late summer they mow in predefined directions. Eliminating weeds is planned once every month. Watering is done every time the ground is drying, and so on.

4.3. Motion

An agent can walk in an ecosystem in one of two different modes – random walk or moving in a certain direction.

The random walk is used for such kind of tasks as pulling out weeds or picking up some plants and the agent is switched to the mode of seeing individual plants. As soon as a plant or plants appear in its visibility range it moves to the closest one and perform the necessary action, then it moves to another one. This can cause overcrowding by agents in

some areas so agent movements need to be synchronized. This is further explained in Section 4.4.

The directional walk is performed for quantitative tasks. An agent walks in a predefined direction that is not changed very much and is less chaotic than a random walk. The walking direction can vary depending on the amount of plants in each subangle of the visibility range (see Section 4.1). Synchronization is easier in this case. A typical example is agents walking in rows and seeding plants.

4.4. Message Passing

Agents communicate by message passing to share their work efficiently. We have noticed in our simulations that in some case agents tend to accumulate in certain areas of an ecosystem. To avoid this we have adopted the approach of *boids* introduced by Reynolds [18]. He used collision avoidance for visual simulation of flying flock of birds. Two moving elements attempt to keep away from each other in at least a predefined distance. We use the method in such a way that each agent before performing its motion searches the neighborhood for another agent. If its step causes violation of the minimal proximity rule, the agent updates its motion to move just the distance not to interpenetrate the working area of the another one.

The above-described step is performed only when an agent has some work to do. If the agent has no plants in its neighborhood it works differently. In this case agents try to share their work and enter into their working areas. It is important to notice that when an agent enters the working area of another agent and finds a plant the mode will be switched back so the agent in the next step will try to leave this area. As a result this approach shares the work efficiently but keeps agents away from each other. This approach has a logical justification. Imagine a group of people picking strawberries. At the moment you do not have any fruit close to you, you will possibly try to move close to someone else who still has some fruits but not too close to invade his or her space.

These two above described approaches help to keep the area efficiently covered by agents.

4.5. Constraints

Every agent has a set of constraints that are applied during the work. These constraints are a set of rules that can be thought of as a program of an agent. One of the rules already described in the Section 4.1 is the mode of work, qualitative or quantitative. The other rules are as follows:

- which plants will be affected,
- which actions will be performed, and
- constraints, determining the working area.

Constraints are described in an external XML configuration file that is parsed and processed by the system and delivered to agents when the simulation starts. This is a simplified version of the description file introduced in [3]. An example follows

$< global >$
t=0;
Δt=0.5;
p_1=daisy;
p_2=grass;
p_3=bush;
p_4=wheat;
p_5=corn;
$< /global >$
$< agent1 >$#farmer
kill: p_1, p_2;
water: p_4, p_5;
constraints: 10<x<20;
20<y<30;
kill: p_3;
water: p_4, p_5;
constraints: 20<x<30;
20<y<30;
$< /agent1 >$
$< agent2 >$#lawnmaker
kill: p_4, p_5;
cut: p_2, p_3;
seed: p_1;
$< position >$
circle[10,10],2;circle[10,15],2;
circle[15,10],2;circle[15,15],2;
$< /position >$
water:p_1, p_2, p_3;
constraints: 0<x<100;
$< /agent2 >$

There are important new parts introduced. First, compared with the agents from [3], the file is simplified. Functions are not used and internal states are missing to allow a significant speedup of the program.

There are new sections of constraints that determine working areas for each agent. It is also important to notice that one agent can do different things in different areas. For example the first agent has two different sections described by its constraints.

Positions for seeding some plant species can be determined by the attribute `position`, as shown in the case of lawnmaker that seeds the daisies only in four circles.

5. Rendering

The simulation algorithm itself works in a 2D area whereas the results are displayed in 3D. A scene description file for the Persistence of Vision raytracer (www.povray.org) is generated during the simulation in the desired time intervals. We also display an OpenGL preview to keep track of the what is happening. This allows the user to stop the process, change some parameters, or run the agents.

Figure 3. Example of an ecosystem rendered using instancing. There are more than 100 000 plants but only 42 originals

The apparent problem is the amount of saved data. The ecosystems are huge and it was difficult or impossible to render these scenes without some additional simplification. We have decided to use instancing to save memory.

Each plant specie is represented by $n = 7$ discrete samples of its lifetime. These samples are models saved in an external file and are used during rendering. When the raytracer description file is to be saved the program makes quantization of the age of each plant and saves the reference to the corresponding instance, so the growth is not continuous but step-wise. To provide an impression of growth, we scale the sample in x, y, and z coordinates between two different samples correspondingly. The plant grows by scaling and every 1/7-th of its lifetime the model switches to another instance. We are aware that we use few different objects in the scene, but the final rendering process is effective and the scene looks realistic. We do not aim to generate animations.

Figure 3 shows a scene with an ecosystem displayed by instancing. The size of the ray-tracer description file is 17MB, the scene consists of 103k plants and only 622k of geometric objects, and 42 different objects. Peak memory used for raytracing was 1.1GB and the total time of rendering in resolution 800×600 pixels with antialiasing on a 2GHz IBM PC was 24 minutes.

An interesting question is if any memory is saved, and if so, how much. Suppose we have a scene that includes daisies, trees, grass, and wheat. Let's denote the number of objects by d, t, g, and w respectively. Let's suppose the size of the object in the memory is $s(x)$, where $x \in \{d, t, g, w\}$. Let's also suppose the size of one instance (*i.e.,* the transformation) is $inst$. Using a non-instanced approach, where each object is completely represented in a memory, gives the total size

$$s_u = d\ s(d) + t\ s(t) + g\ s(g) + w\ s(w)$$

the size of the instanced representation is

$$s_i = inst(d + t + g + w) + s(d) + s(t) + s(g) + s(w).$$

The compression factor is then $c = s_i/s_u$. For one scene we got:

object	size (kB)	number of objects
daisy	105	72
grass	1	3092
trees	733	8
wheat	160	460

The $s_u = 90\ 116$kB, the $s_i = 9\ 228$kB and the compression factor is 90%. For other scenes the compression ranges from 80% to 90% and in general is better for large scenes.

6. Results

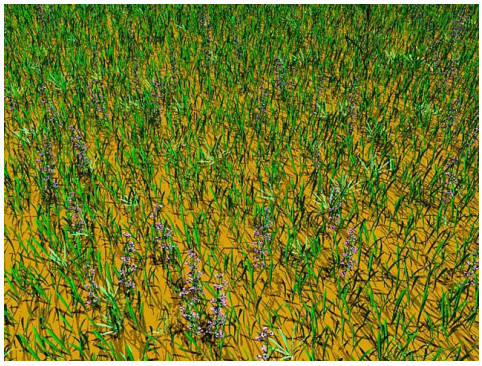

Figure 4. Creation of a virtual garden starts with a wild ecosystem

Two examples of a virtual garden are presented. Gardens occupy area of 16×16 meters and are inhabited by approximately 256k plants. All images were rendered in resolution

800×600 pixels with antialiasing and the rendering time on a 2GHz IBM PC was 35 minutes.

The following example shows the way a virtual garden is cultivated. First, as shown in the close-up in Figure 4, a wild ecosystem with 200k blades of grass, 80 plants of wheat, 2000 plants of gilia, 2000 clusters of grass, and 20k of blue bells is cultivated. Seven agents repeatedly pull out weeds and areas that are supposed to be a sidewalk are cleaned from plants. This cultivation is done in three visits during one week and the agents work approximately two hours. The entire time of simulation was less than two minutes. The result is shown in Figure 5.

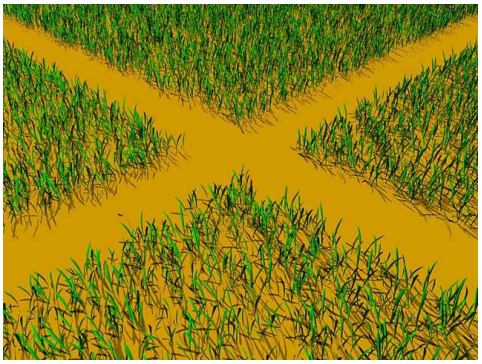

Figure 5. Weeds are pulled and a space for sidewalks is prepared

In the next step (Figure 6) the sidewalks are laid and tulips are planted. This was done during one visit of seven agents in a duration of two hours.

Later, the garden is repeatedly cleared of weeds and tulips are protected and kept in their areas. It means that any tulip that is found outside its predefined space is considered as a weed and eliminated.

In the second example in Figure 7 the same strategy was used but tulips were planted in lines along the sidewalk.

7. Conclusions and Future Work

Modeling of garden creation by virtual autonomous agents is presented. Agents are continuous automata without memory and can perform predefined actions; they can walk and pull out weeds, seed new plants, water them, and undertake other tasks. Agents communicate to deal with the work efficiently. Each agent has its own set of tasks and treats to perform them efficiently in cooperation with the other agents. An external file with a simple description language defines the tasks to be performed and when the program runs these tasks are delivered to the agents.

Figure 6. In the last step tulips are planted and protected

The simulation algorithm runs efficiently on a mid-class PC with up to million plants. Instancing allows for reasonable rendering saving up to 90% of space using the Persistence of Vision ray tracer on scenes having hundreds of thousands of plants.

From the user's point of view the usage of agents is simple. Users defines the tasks that should be performed by agents by writing them to the agent description file, then they define the number of agents, and run the ecosystem development. Agents repeatedly (scheduled or run by users) visit the ecosystem and perform their tasks. Users can save intermediate steps and run different agents to perform different tasks. In this interactive way the ecosystem can be transformed into, for example, a field or a garden.

There are many possible extensions. One of them is the way the constraints are defined. It is not really handy to define the constraints by functions, so some kind of visual input could help to simplify this process, we could use a binary image defining the areas. Another important future work is the speed of the simulation. The process is efficient, but for large scenes we easily loose interactivity. One of the frequent criticisms is the predictability of these approaches. How can we know what will be the result, if it depends on interactions of some agents? One solution is to divide the task into distinct steps and save the intermediate works. User could possibly make undo steps and continue with different rules.

We believe this algorithm is sufficiently clear and easy to implement and can be used to prepare realistic scenes that are impossible or hard to create by hand or by other techniques. This approach gives results that are visually intune with our expectations and we think that this could be possibly used as a tool for observing what could happen with real ecosystems.

8. Acknowledgment

We would like to thank to anonymous referees who greatly helped to improve quality of this paper.

This work is sponsored by the ITESM Grant No.3086.

References

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

[2] B. Beneš. A Stable Modeling of Large Plant Ecosystems. In *Proceedings of the International Conference on Computer Vision and Graphics*, pages 94–101. Association for Image Processing, 2002.

[3] B. Beneš and E. D. Espinosa. Modeling Virtual Ecosystems with Proactive Guidance of Agents. *IEEE Proceedings of the Computer Animation and Social Agents 2003*, IEEE Computer Society, 2003, to appear.

[4] B. Beneš and E. Millán. Virtual Climbing Plants Competing for Space. In N. Magnenat-Thalmann, editor, *IEEE Proceedings of the Computer Animation 2002*, pages 33–42. IEEE Computer Society, 2002.

[5] B. Beneš, J. Soto, and J. Cordoba. Interacting Agents with Memory in Virtual Ecosystems. *Journal of WSCG*, 11(I), pages 49–56, 2003.

[6] 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. ACM Press, 1998.

[7] N. Greene. Voxel Space Automata: Modeling with Stochastic Growth Processes in Voxel Space. In *Proceedings of SIGGRAPH '89*, volume 23(4) of *Annual Conference Series 1989*, pages 175–184. ACM Press, 1989.

[8] 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*, 2002, to appear.

[9] M. Holton. Strands, Gravity and Botanical Tree Imagery. *Computer Graphics Forum*, 13(I), pages 57–67, 1994.

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

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

[12] A. Lindenmayer. Mathematical models for cellular interaction in development. *Journal of Theoretical Biology*, Parts I and II(18), pages 280–315, 1968.

[13] 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. ACM Press, 1996.

[14] 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. ACM Press, 1986.

[15] 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), pages 12–22, 1993.

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

[17] 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.

[18] C. Reynolds. Flocks, Herds and Schools: A Distributed Behavioral Model. In *Proceedings of SIGGRAPH '87*, volume 21(3) of *Annual Conference Series*, pages 25–34. ACM Press, 1987.

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

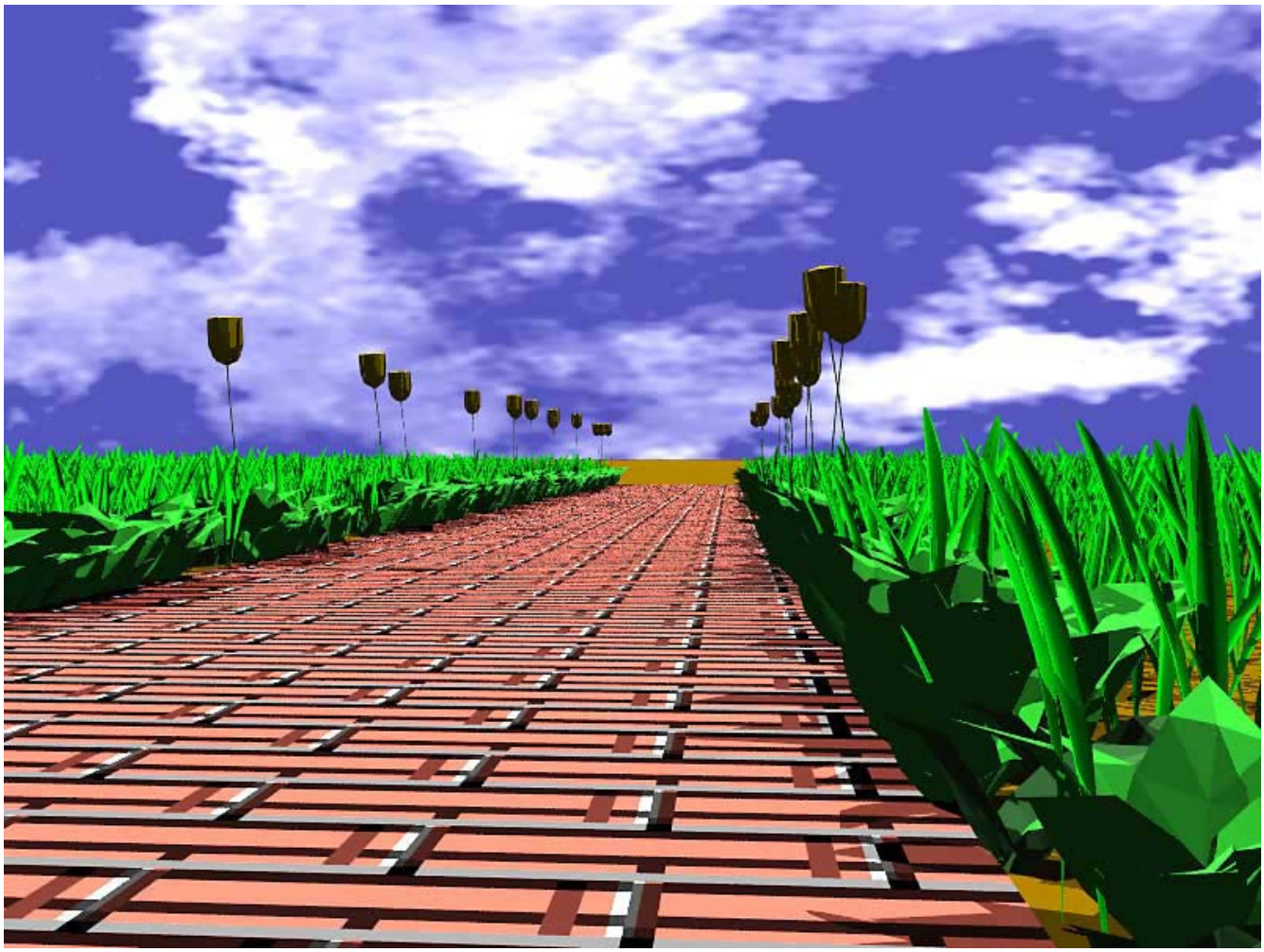

Figure 7. Virtual garden as a result of the work of virtual agents