A Ray Density Estimation Approach to Take into Account Environment Illumination in Plant Growth Simulation

William Van Haevre Fabian Di Fiore Philippe Bekaert Frank Van Reeth

Limburgs Universitair Centrum
Expertise Centre for Digital Media
Universitaire Campus
B-3590 Diepenbeek, Belgium
{william.vanhaevre, fabian.difiore, philippe.bekaert, frank.vanreeth}@luc.ac.be

Figure 1: These figures depict some results of our algorithm to simulate plant growth. In this case the plants were subject to the direct and indirect illumination of the ceiling lamps.

Abstract

Light interaction is one of the most important factors in developing realistic plant models. Plants react to received illumination by bending branches, adapting their growth rate, orienting leaves and flowers, producing larger or smaller leaves, etc.

In this paper, we present a novel approach to simulate plant growth as a response to environment illumination. The basic idea of our algorithm is to simulate light transport in the environment in which plants grow by tracing light particles originating from light sources. Both intensity and mean direction of incident illumination are determined easily. This is based on a ray density estimation of the environment illumination by means of a predominant illumination direction.

An adaptive spatial data structure is used to store the rays along which light particles travel in space. This data structure allows efficient calculation of ray density at locations where the algorithm needs to query incident illumination.

Our approach takes into account both direct and indirect illumination and is an algorithm that is both flexible and accurate. It is easy to implement and more general illumination models can be incorporated in a straightforward manner. Furthermore, using a non-uniform, adaptive data structure for storing the rays, calculation time and storage requirements are kept within reasonable limits.

CR Categories: I.6.3 [Simulation and Modelling]: Applications—Simulation of Natural Phenomena

Keywords: plant modelling, natural phenomena, light interaction, ray density, L-systems, phototropism

1 Introduction

Plant modelling has always been a challenging task. Due to the complex geometry involved, not only the modelling process itself is labour-intensive, also the interaction of the model with its environment yields several im-

portant problems (e.g. spatial constraints) that need to be solved.

Many different techniques have been developed to model realistic plants and their growth behaviour. These vary from the use of particles which sense the environment to search for the best growth direction [Benes and Millan 2002], to casting rays from a specific point in space to find out how this location is occluded from the available light in the scene [Greene 1989; Měch and Prusinkiewicz 1996]. Most of these techniques are based on a heuristic approach. In contrast, we introduce a novel algorithm that for each point of interest calculates the amount of light present in a certain neighbourhood. Our solution is based on the estimation of the ray density and the retrieval of the mean direction of the incident illumination (figure 1).

We use L-systems [Prusinkiewicz and Lindenmayer 1990] to model the plants and their growth movements. However, other methods can be used, as long as they support the interaction of the model with the presented data structure (e.g. marking of growth positions).

This paper is organised as follows. In Section 2 we present a short review on related work and indicate the differences with our method. Section 3 elaborates on the data structure used to represent light in the scene. We demonstrate the use of this structure to locate the required information about the illumination in Section 4. Section 5 explains in detail the representation of the plants themselves and gives a short review on the use of L-systems. Section 6 illustrates a few interactions in which plants might participate by using the obtained information. We end this paper with an evaluation of the results (Section 7) and our conclusions and topics for future research (Section 8).

2 Related Work

For many years, people have been exploring the area of plant modelling, which is situated on the cross-section of the fields of biology and computer graphics.

Early efforts include the work of Aono and Kunii [Aono and Kunii 1984], in which the creation of complex branching patterns is described. Moreover, for the first time, interaction with the environment was simulated. Concurrently, because of the self similarity they provide, fractals have been used to create plant shapes [Bloomenthal 1985; Oppenheimer August 1986].

Particle systems provide yet another approach to visualise certain natural phenomena [Reeves 1983; Reeves and Blau 1985]. Recently, this method and the work of Greene [Greene 1989] were extended, allowing environment sensitive modelling of the competition for space of climbing plants [Benes and Millan 2002].

De Reffye et al. [de Reffye et al. 1988], and, Lintermann and Deussen [Lintermann and Deussen 1999], use a component based approach. The former developed a procedural model while Lintermann and Deussen created an interactive plant modelling program *xfrog* [Deussen and Lintermann 1998] in which components are used to encapsulate data while algorithms are responsible for generating the different plant elements. However, each of these components has its own specific set of parameters to control its behaviour.

There has been a consensus among researchers to turn to L-systems, introduced by Lindenmayer [Prusinkiewicz and Lindenmayer 1990] to describe the shape and behaviour of plants. L-systems provide an intuitive and elegant way to describe plant growth properties. This is elucidated in Section 5.

During the last decade, the behaviour of communities of plants [Lane and Prusinkiewicz 2002] and even of complete ecosystems [Deussen et al. 1998] have been examined. Several papers addressed the questions concerning plant interaction with each other and objects within their environment [Benes and Millan 2002; Van Haevre and Bekaert 2003; Měch and Prusinkiewicz 1996; Prusinkiewicz et al. 1994].

Earlier presented methods estimated the illumination influencing plant growth in a heuristic way. These proposals varied from simple ray-casting techniques towards the sky through the plant model, over the use of a voxel representation [Greene 1989], to techniques that take into account the opacity of the voxels encountered by a light ray to represent the translucency of the plant foliage [Měch and Prusinkiewicz 1996].

Global illumination techniques have been used to take care of the light distribution in and outside the plant models. Radiosity as well as Monte Carlo techniques can be used to achieve this. Our research is highly related with work done by Soler et al. [Soler et al. 2003]. In their work an extension of hierarchical radiosity with clustering is presented and it is used for simulating plant growth taking into account illumination. A more detailed overview of earlier work on plant light interaction can be found in their paper.

In this paper, we introduce a novel approach inspired by photon mapping [Jensen 2001]. Like photon mapping, the algorithm will deal easily with non-diffuse light reflection and translucency as well as complex and dynamic geometry. Unlike photon mapping, we perform density estimation of light rays in an environment rather than estimating the density of photon hits on object surfaces. Moreover, contrary to other simple solutions that

incorporate environment light in plant growth [Benes and Millan 2002; Měch and Prusinkiewicz 1996] and our previous work [Van Haevre and Bekaert 2003], the new algorithm can provide accurate estimation of the illumination, including indirect illumination, with ease.

Plant models are regarded as static objects during light computation. Every growth step allows the plants to change their shape according to the lighting in their neighbourhood. The changes in the plants' shapes influence the lighting in the scene which again influences the plants' shapes etc. In order to speed up this iterative process, we create a data structure that facilitates the following two actions: (i) avoiding full recomputation of the lighting after applying the changes to the plants' shape (Section 3.1); (ii) fast querying of the lighting in the neighbourhood of a plant, needed in order to determine how the shape of the plant is going to evolve (Section 4).

3 The Representation of Environment Illumination

In this section we describe the data structure and calculations required to represent the ray density and the mean incident illumination direction.

Light transport is simulated by tracing light particles (i.e. photons) originating from light sources. As these photons strike a surface they can either be absorbed or reflected back into the scene, according to the material properties of the surface. In the next section, we propose density estimation of the rays (along which particles travel) in order to calculate the intensity and mean direction of illumination in space (i.e. the radiance field).

For a start, a data structure allowing efficient density querying needs to be set up. This data structure should allow the rapid retrieval of rays passing nearest to a query location. Spatial subdivision data structures such as a uniform grid, an octree or a BSP tree, in which spatial cells are tagged with references to the rays passing through them, are well suited for this task.

The use of a uniform grid is prohibitively expensive in terms of memory requirements and inefficient as well, since many light rays are likely to pass through areas with no active plant growth, in which no querying for illumination will be performed. Each ray would intersect a lot of grid cells and, hence, each of the grid cells would have to keep a reference to the ray. Therefore, the use of an octree is recommended, since it is a non-uniform data structure that can be updated dynamically (Section 3.1) and can be constructed with higher detail where required.

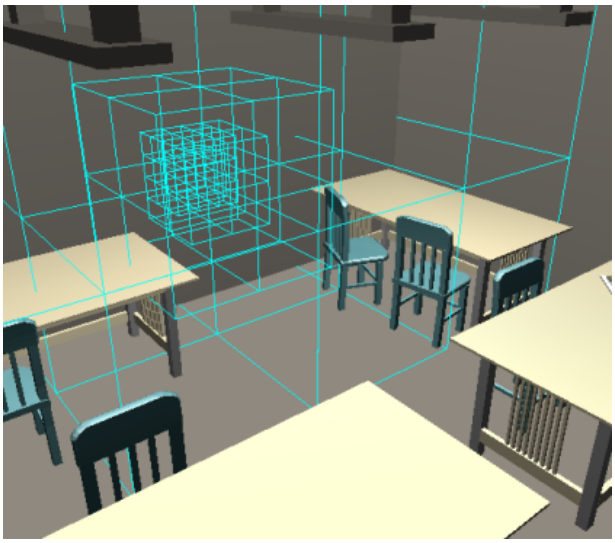

Figure 2: An octree as a spatial subdivision data structure, allowing local storage of references to nearby light rays.

The octree is constructed in a demand-driven way. Whenever the radiance field needs to be queried at a given location, a "point of interest" or "refinement point" p is recorded. Every time such a new point is added to the scene, the octree is refined (if necessary) by splitting cells into uniform octants, until a specific cell width c is reached (see figures 2 and 3).

As light is emitted from the light sources, light rays intersect the existing nodes of the octree. After adding a refinement point and updating the references to the rays, the cell containing the point will hold a reference to all rays passing through this cell. Whenever a refinement point is no longer required, the references to the intersecting rays can be discarded, while keeping the references in the higher layer of the octree for further refinements.

Consequently, octree cells might be created as refinement points are added. After their addition, the cells are updated using the references to rays stored in higher levels of the octree. When light rays are added to the scene, the topology of the structure remains unchanged and only one update of the existing cells is required. Empty cells (which contain no detail points) are deleted recursively. This way, the amount of memory needed is kept to a minimum.

3.1 A Dynamic Update

While the plant's shape is enlarging, several branches, leaves and flowers are added to the scene. As a result, a small number of rays suddenly gets occluded, leaving some invalid paths in the environment. To resolve this

instability, a dynamic update of the rays is required.

Refinement points are added to positions where the plant is growing. Consequently, the topology of the grid is focussed around these growth positions, and the rays that might be blocked by new plant parts are found in the corresponding grid cells of these points. As a path consists of a linked list of rays, it can be broken off as soon as an interruption occurs. Hence, the pruned part of the path is discarded, references to these parts are removed from the octree, and the path is extended by recasting the intersected ray into the scene, taking into account the new plant parts.

Listing 1 illustrates the steps required to update a plant model to the next stadium of its growth, after being initialised:

1. Update the sequence that represents the shape of the plant, using the production rules of a plant's specific L-system that describes the behaviour of the plant.
2. Create a new plant model from this sequence and add refinement points at query positions. As new cells are created, the shape of the octree and the references it contains alter.
3. Remove empty cells from the octree whereas some of them might exist from previous iterations but have become useless for the current plant shape.
4. Remove all ray references in the finest layer of the octree because these are the locations where new plant parts may have been created. Afterwards, a full update of the octree is required to remove the obsolete references to rays within all of its layers.
5. Due to the removal of rays and rays that were created in the path behind it, paths need to be extended again, incorporating the geometry of the newly generated plant parts.
6. Using the constructed octree, illumination intensity and mean incident light direction can be calculated (Section 4) and returned to the L-system by means of "open" parameters.
7. Remove all refinement points as they are no longer required.

```
Grow(Plant)
LSystem.LoadFromfile();
LSystemSequence = LSystem.getAxiom();
for each growth iteration do
    LSystemSequence.update(LSystem);
      //using a plant's specific L-system
    Plant.CreateModel(LSystemSequence);
      //while adding refinement points at query locations
    Octree.RemoveEmptyCells(); (recursively)
      //to free memory
    Octree.RemoveRayReferencesInQueriedCells();
      //top-down to remove all references to obsolete rays
    Rays.update();
      //recast paths where broken
    LSystemSequence.RetrieveOpenParameters()
      //updates the sequence while calculating ray density
      //and incident light direction
    Octree.RemoveRefinementPoints();
      //allowing obsolete cells to be removed in the next iteration
end for
```

Listing 1: Overview of the growth process.

4 Illumination Querying

In this section, we explain how to retrieve the light intensity and mean incident light direction while estimating ray density.

Each ray cast in the scene transports a fraction $\Delta\Phi_i$ of the light present in the scene. Based on a single ray, we cannot trace the dominant direction and intensity of the light flow at a specific point. To calculate this direction and to find an estimation of the light intensity, the ray density is used. This method is inspired by the estimation of the irradiance on a surface, described by Jensen [Jensen 2001].

The ray density at a position can be expressed as the amount of rays intersecting a predefined volume (a sphere in our case) around that point. To estimate the density of rays around a point, several possibilities exist [Silverman 1986], for instance:

1. **Nearest neighbours method:** search for a specific number of the closest rays around the point and calculate the corresponding volume. The radius equals the highest distance found between the point and one of these rays.
2. **Histogram method:** search all rays around the point inside a fixed volume. This means that a fixed search distance is used.

In both cases, the illumination intensity at the query location can be obtained by summing the photon powers $\Delta\Phi_i$ of the rays, per unit of projected query volume surface area. When using a sphere as a query volume, the projected surface area is just the area of a disc with the same radius. When the Russian Roulette reflection algorithm [Jensen 2001] is used while tracing the light paths through the environment, the power $\Delta\Phi_i$ of all the rays can be kept equal and the ray density can be expressed as the amount of rays per unit of projected query volume surface area. This estimation can be used relative to a reference density (which stands for full illumination). In this way one can decide how much illumination is present at a certain point. The reference density can be acquired at a user defined position and should represent the amount of rays per unit of projected query volume surface area for a fully illuminated point.

The dominant illumination direction is found by sum-

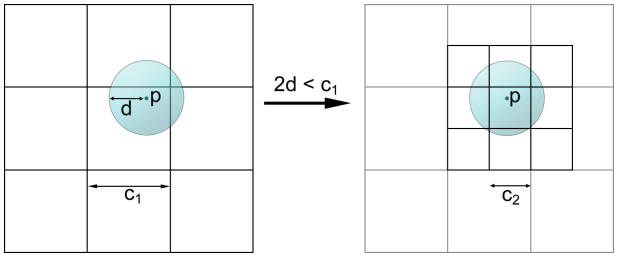

Figure 3: An adaptive octree depth based on a user-specified search distance d used for illumination querying. (Left) $2d$ is smaller than c_1. The octree needs to be divided. (Right) New situation: $2d$ is larger than c_2. No further subdivision is required.

ming vectors with magnitude $\Delta\Phi_i$ (possibly equal for all rays, when Russian Roulette is used) into the ray direction. The result should be normalised .

In order to facilitate the search for rays in the octree we advise employing the histogram method, since the fixed search distance allows a much easier search for rays in the octree. Using a fixed search distance d to estimate the ray density, we can calculate a depth for the octree to obtain a cell width of maximum $2d$ for the lowest layer (the layer containing the refinement points). This way, we can restrict our search for rays to the current cell and its 26 adjacent cells (see figure 3, which for clarity visualises the 2D case).

Some of the required adjacent octree cells might not yet exist. To force their existence, (as we need the references to their intersecting rays), we add the necessary refinement points to the octree. The positions can be calculated from the query position. In this way we locally create a uniform grid which allows us to find the requested light rays by simple look-up. Of course, there is some cost involved in refining the octree at newly inserted refinement points, but often subsequent queries are done in the same neighbourhood, usually spreading the cost over several queries.

¿From the rays intersecting the 26 octree cells, only those intersecting the search volume (a sphere) are retained. This requires checking the distance between the rays and the query position, and comparing this distance with the search sphere radius. Figure 4 shows how an octree in combination with a local uniform grid allows a fast, position oriented search in space for light rays.

In Section 6 we show that many interesting illumination related plant growth effects can be modelled considering little more than the intensity and mean direction of illumination near active locations of growth. More complex interactions can be simulated as well, but they require knowledge about plant specific growth behaviour.

5 The Representation of Plants

The ray density estimation approach proposed in previous section is not connected to a particular plant growth model. Any model can be used, as long as it contains a mechanism to incorporate external data such as illumination. We decided to make use of L-systems [Prusinkiewicz and Lindenmayer 1990; Prusinkiewicz et al. 2001] to model plants. In the following section a small overview is given about L-systems.

An L-system is a parallel string rewriting system that constructs a generally more complex string of characters from a less complex string by using production rules. A production rule is defined as follows:

$$label : lc < pred > rc : cond \rightarrow succ : prob$$

label: a label to mark the production rule.

lc: (optional) a left context after which the predecessor must follow to make the rule applicable.

pred: the predecessor, the character from the original string that is to be replaced by a more complex sequence of characters.

rc: (optional) a right context in front of which the predecessor must be positioned to make the rule applicable.

cond: (optional) a condition which must evaluate to true, based on parameters from *lc*, *pred* or *rc*, to make the rule applicable.

succ: the successor, a string of characters to replace the predecessor.

prob: a stochastic value, indicating the probability for this rule to be selected when several rules are applicable.

Starting with an axiom (a begin sequence) the L-system replaces each of the characters (in parallel) when using these production rules. Each of these symbols needs to get a proper visual interpretation, to obtain a visual model. A well known technique to do this is called "Turtle Interpretation" [Prusinkiewicz and Lindenmayer 1990]. In this case the string of characters acts as a sequence of changes, applicable to the state of a drawing tool (a pen, the "turtle", ...). Each symbol represents a specific change to the state. At any time, the state is characterised by a position p and three mutually perpendicular orientation vectors H, U and L, indicating the heading, the up direction and the direction to the left. Symbols like $+$, $-$, &, $\wedge$, $\backslash$ and $/$ rotate the state. Other symbols can move the state to a different position, for instance when drawing a branch. More information on how to create and use L-systems can be found in [Prusinkiewicz and Hammel 1994; Prusinkiewicz et al. 1996; Prusinkiewicz and Kari 1996; Prusinkiewicz and Lindenmayer 1990].

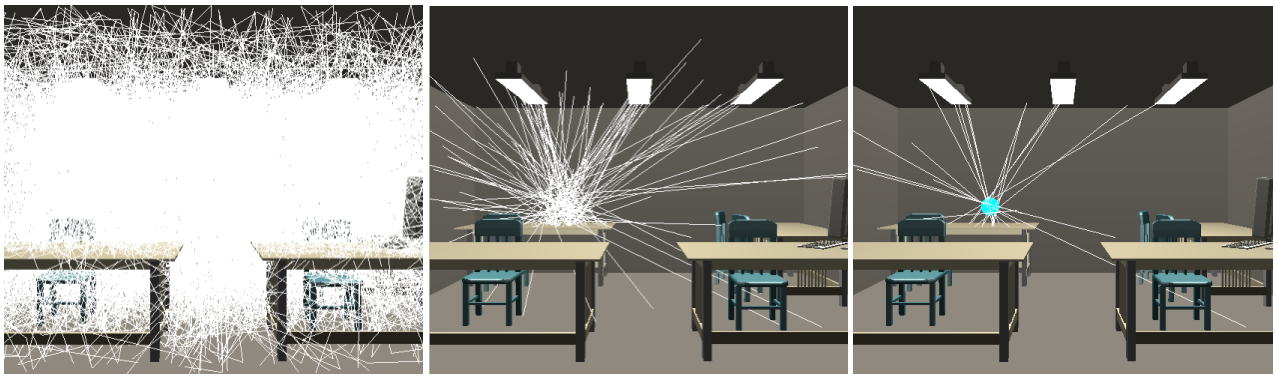

Figure 4: Three incremental refinements of the nearest rays around a query point. (Left) all rays within the scene, (Middle) all rays intersecting the 27 surrounding grid cells, (Right) all nearest rays within search distance d.

Měch and Prusinkiewicz [Měch and Prusinkiewicz 1996] introduced a variant of L-systems, called "open" L-systems, which contain a mechanism for incorporating "external" data in production rules. Special symbols are introduced in the production rules. After rewriting a sequence, the symbols are interpreted and replaced by associated parameter values such as lengths and rotation angles. This way, specific alterations can be made to the model to take into account the requirements of the environment. This is the place where ray density is queried (see Section 4).

6 Plant-Light Interaction

This section demonstrates the ease by which our algorithm can be used. The following examples are not based on specific properties of certain species, but illustrate common aspects of plant growth concerning light interaction. The combination of the existing literature on L-systems together with research for the biological properties of plants allows the creation of growing plants in a natural way within their environment.

In [Kendrik and Kronenberg 1986], Kendrik and Kronenberg cover several aspects of the interaction between plants and the present illumination in an environment. In our research we focused on phototropism, which is the subject of the next section.

6.1 Phototropism

The phenomenon manifested as a sequence of growth movements of plants or a change in their shape in relation (positive or negative) to the incoming light direction is called phototropism.

Several movements are made by a plant while reaching to or moving away from the available illumination. Branches elongate or remain short due to the amount of light present in their environment. Many existing plant species bend towards light sources, to reach for as much of the available light as possible. Others don't or do the opposite. Leaves might position themselves perpendicular to the incoming light direction while gathering energy for the growth of the plant — by means of the chemical process called photosynthesis. Other species, however, orient their leaves away from that direction to prevent overheating, and allow light beams to penetrate more deeply into the canopy. Also, the size of the leaves may correspond to the availability of light at a specific position.

6.2 The Bending of Branches

The bending of branches due to the incoming light direction is a well known phenomenon. Several plants try to shape themselves to increase the number of plant parts that can be reached by light rays. In order to enlarge the total surface of leaves and branches that is illuminated, the branches will bend to occupy free space providing sufficient illumination.

When using L-systems to model the plants, a symbol can be used to interact with the environment [Měch and Prusinkiewicz 1996]. For instance, if F represents a branch, an "open" symbol $R(?,?)$ could be placed behind it. $R(?,?)$ will be interpreted by the application as a request for a new growth direction. The parameters will get a proper value which leads to $R(r,p)$. Later on (as shown in listing 1) the symbol will be replaced by a combination of rotations $\backslash(r)\&(p)/(r)$, due to production rule p_2 (see L-system 1). This way, the state at the end of the branch will have been turned to the calculated direction.

L-system 1

$$\begin{array}{lcl} \omega : A & & \\ p_1 : A & \rightarrow & FR(?,?)A \\ p_2 : R(r,p) & \rightarrow & \backslash(r)\&(p)/(r) \end{array}$$

The parameters r (roll) and p (pitch) are calculated as follows: whenever the symbol $R(?,?)$ is encountered while interpreting the sequence, a refinement point is added to the octree at the current state's position. To find the open parameters for the new sequence, the nearest rays to this position are located. From these rays the mean incident light direction is derived as explained in Section 4. This direction is compared to the current state vectors H_c, U_c and L_c. A roll angle (rotation around H_c) and pitch angle (rotation around L_c) are calculated which transform the current state to a new state that incorporates bending. To suppress the effect of bending too exaggerated, a maximum pitch angle can be defined and given to the application instead of one of the ?'s, to which the calculated pitch angle can be constrained. The combination of the $\backslash(r)$, $\&(p)$ and $/(r)$ commands makes it possible to rotate p degrees away from the heading vector, in any direction, without a distortion of the overall plant topology.

The following set of sequences is an example of what might be produced during a few derivation steps (we have limited the pitch angle to 8 degrees):

$$\begin{array}{lcl} 0 & : & A \\ 0' & : & A \\ 1 & : & FR(?,8)A \\ 1' & : & FR(20,6)A \\ 2 & : & F\backslash(20)\&(6)/(20)FR(?,8)A \\ 2' & : & F\backslash(20)\&(6)/(20)FR(9,5)A \\ 3 & : & F\backslash(20)\&(6)/(20)F\backslash(9)\&(5)/(9)FR(?,8)A \\ 3' & : & F\backslash(20)\&(6)/(20)F\backslash(9)\&(5)/(9)FR(3,8)A \end{array}$$

When visualising such a sequence using turtle graphics, a growing branch is displayed bending locally towards the direction calculated by the application (figure 5).

6.3 Leaf and Flower Orientation

As mentioned before, the leaves provide one of the most important sources of energy for the plant. When they absorb light, the process of photosynthesis produces energy which makes it possible for the plant to grow, to remain strong, to produce new plant parts, etc. In order to sustain this task, an optimal orientation of the leaves in relation to the positions of the light sources becomes very important.

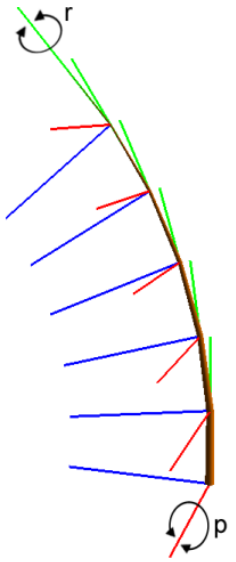

Figure 5: A branch bending towards a calculated direction, using a roll and a pitch angle to change the growth state.

When a leaf is added to the end of a branch or a twig, a refinement point is created and placed at that position into the octree. Then, the nearest rays are located and the mean incident light direction is estimated. Next, a new state is created for this position, based on the current state of the L-system and the estimated incoming light direction. Finally, the changes needed to convert the original state to the desired state are returned to the L-system so they will be applied in the next iteration.

The same idea is valid when positioning flowers or other plant parts.

6.4 Branch, Leaf and Flower Size

Another phototropism we want to illustrate by using our algorithm, concerns the length of the branches and the size of the leaves, in relation to the available light. Several types of plants tend to grow faster (enlarging the branches) in dark spaces while preventing leaves from growing, in order to reach the illuminated areas faster. Once they get out of the dark, they reduce their growth speed and focus on the production of leaves and flowers, to obtain their required energy.

Again, we used a special symbol in our L-system which allows the required interaction with the environment. An L-system containing the main production rules for this behaviour is illustrated below (see L-system 2). (with f_i, g_i and h_i functions in i.)

L-system 2

$$
\begin{array}{lrcl}
\omega : & AI(?)F(0)I(?)W(0) & & \\
p_1 : & A & \rightarrow & I(?)F(0)[+(8)I(?)L(0)]A \\
p_2 : I(i) & < W(0) & \rightarrow & W(f_i) \\
p_3 : I(i) & < L(0) & \rightarrow & L(g_i) \\
p_4 : I(i) & < F(0) & \rightarrow & F(h_i)
\end{array}
$$

In the axiom, the open symbol $I(?)$ appears twice. First before an F (a piece of branch) and next before a W symbol, which marks the position of a flower in the plant. In the first production rule it also appears before an L which in our case is the symbol marking a new leaf.

Whenever $I(?)$ is encountered, a refinement point is added at the position indicated by the current state of the L-system. The nearest rays are located and the ray density is estimated. Depending on this density a number is returned for the open parameter. Subordinate to this parameter, specific changes can be made to the plant model. Rules p_2, p_3, and p_4 demonstrate how the parameters of the module representing a flower, a leaf or a branch can be altered according to the value i that is returned by the application through module I. A plant specific function f,g or h can map this parameter on a suitable value, allowing the adaptation of the plant to darkness and the behaviour to reach for more (or less) illuminated environments by increasing or decreasing the size of its components.

6.5 More Complex Behaviour

All of the previously mentioned changes to the plant model can easily be combined into a more complex behaviour. It simply requires providing an L-system that contains the production rules which describe the needed behaviour. For instance, by combining two techniques described earlier in this paper (Sections 6.2 and 6.4) into one L-system, branches can bend or remain straight depending on the amount of illumination reaching them. The combination of production rules illustrated in L-system 3 ensures the maximum bending angle for module R to be limited to a function f of parameter i, which was returned by the query for ray density proceeding it.

L-system 3

$$
\begin{array}{lcl}
\omega : A & & \\
p_1 : A & \rightarrow & I(?)WF(1)A \\
p_2 : I(i) < W & \rightarrow & R(?, f_i)
\end{array}
$$

It is obvious that other phenomena can be simulated easily using this kind of combinations (e.g. linking the maximum bend angle to the age of the branch, increasing or decreasing the amount of leaves or flowers in relation to the available light at a certain position, ...).

7 Results and Discussion

Figure 6 shows two test scenes. All plant models in each scene are created using the same L-system, indicating its specific topology and growth behaviour. Due to light interaction, the shape of each plant is altered, according to its position in the scene and the location of the light sources.

The corresponding charts indicate the amount of time required for each iteration in the plant growth simulation. They display the calculation speed of a full update of the plant model (including the update of the octree, the refinement points and the rays). We must stress that these results can be improved using a more optimised implementation.

The shape of the curves clearly imply that the first iteration of the growth process relatively demands more calculation time than the subsequent iterations. This is due to the first refinement of the octree, which demands the creation of several octants at different layers and a recursive update of the references to the intersecting rays. The other iterations have a calculation time increasing exponentially in relation to the complexity of the model. The small fluctuations in the illustrated timings are caused by larger changes to the octree which are necessary when the model enters previously non existent parts of the octree. Again, the corresponding refinement requires the creation of several octants at different layers and an update of the corresponding references to the intersecting rays. All timings were acquired on a 1000 MHz PC with 256 MB's of memory.

The first scene in figure 6 contains four plants with flowers, subject to the following rules as they grow: (i) branches bend towards the mean incident light direction; (ii) the maximum bending angle depends on the amount of illumination at each position (as more light reaches the branch, less bending occurs); (iii) flowers are created when a sufficient amount of light is available. Three small bushes were added to the second scene, growing according to the following behaviour: (i) branches bend towards the mean incident light direction; (ii) the size of branches and leaves depends on the amount of incident illumination.

Figure 7 shows a closer view of the resulting plant models. The first row illustrates the plant growing from under the table, the plant on the table bending slightly towards the light sources, and a close view of the plant in the middle of the room growing straight up towards the ceiling. The second row shows three views of the bushes whose

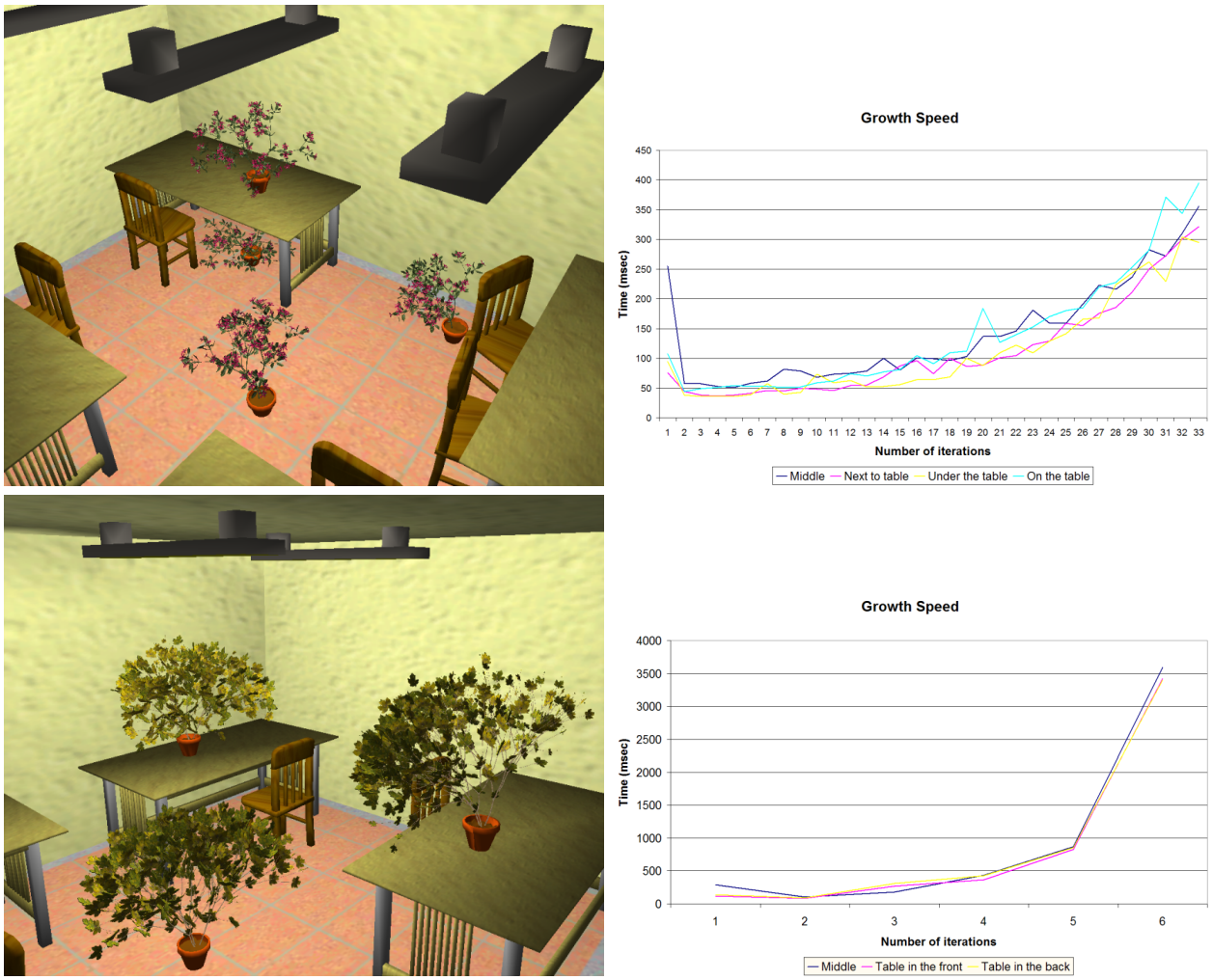

Figure 6: Two examples of a scene in which several plants, created from the same L-system, are growing in various ways due to different environmental properties (under a table, on a table more close to the light sources, next to the wall, etc). For each plant the calculation time required to update the model is depicted in the corresponding chart.

canopy is much denser at more illuminated parts. The last three illustrations are top views of the plants, indicating how the overall shape of the plant is influenced by possible local changes to the model, without the requirement to change the topology of the plant.

Figure 8 displays a few snapshots of an animation depicting the growth of the plant located under the table in figure 6. As the amount of incident illumination increases, the bending of the branches is no longer required and the plant develops itself by creating more flowers.

Our ray density estimation exhibits noise, due to randomness in the light emission. This produces fluctuations in the growth parameters which are hardly visible and hence even provide a small increase to the realism of the grown plants. We believe that this noise can be reduced by using a different estimator (e.g. nearest neighbour).

8 Conclusions

We have presented a novel approach, based on a ray density calculation to get an estimation of the environment illumination by means of a predominant illumination direction. This information can be used in a simulation of plant growth, allowing several movements of the plants to obtain an optimal growth direction. The flexibility and accuracy of the algorithm, together with its low calculation time and limited memory usage ensure an innovative technique, attractive for plant modelling applications.

In the future, we would like to focus on other estimators. The nearest neighbours method, for instance, always guarantees a sufficient number of rays to do statistics. Secondly, we are looking forward to exploit data available from real plant species, and to experiment with new plant representations.

Figure 7: A closer view of the plants depicted in figure 6. (First row) Growing from under a table; bending slightly towards the light sources; in the middle of a room, growing straight up towards the ceiling. (Second row) Bushes whose canopy is much denser at parts were more illumination is present. (Third row) Top views from plants indicating the overall shape of the plant being influenced by local changes to the model without the requirement to change the topology of the plant.

Figure 8: Several snapshots of an animation depicting the growth of the plant located under the table in figure 6. Both the maximum angle — used to bend the branches — and the creation of new flowers depend on the amount of incident illumination at each specific position (see Section 6.5).

Acknowledgements

We gratefully express our gratitude to the European Fund for Regional Development and the Flemish Government, which are kindly funding part of the research reported in this paper. Furthermore, we would like to thank Bjorn Geuns for his appreciated artistic input.

References

AONO, M., AND KUNII, T. 1984. Botanical image tree generation. *IEEE Computer Graphics and Applications 4*, 5, 10–34.

BENES, B., AND MILLAN, E. U. 2002. Virtual climbing plants competing for space. In *Computer Animation Proceedings 2002*, ACM Press, 33–42.

BLOOMENTHAL, J. 1985. Modeling the mighty maple. *Computer Graphics (SIGGRAPH '85 Proceedings) 19*, 3 (July), 305–311.

DE REFFYE, P., EDELIN, C., FRANÇON, J., JAEGER, M., AND PUECH, C. 1988. Plant models faithful to botanical structure and development. *Computer Graphics (SIGGRAPH '88 proceedings) 22*, 4 (Aug.), 151–158.

DEUSSEN, O., AND LINTERMANN, B., 1998. software: xfrog 2.0, www.greenworks.de.

DEUSSEN, O., HANRAHAN, P., LINTERMANN, B., MECH, R., PHARR, M., AND PRUSINKIEWICZ, P. 1998. Realistic modeling and rendering of plant ecosystems. In *SIGGRAPH '98 Conference Proceedings*, ACM Press, ACM SIGGRAPH, 275–286.

GREENE, N. 1989. Voxel space automata: modeling with stochastic growth processes in voxel space. *SIGGRAPH '89 23*, 3 (July), 175–184.

JENSEN, H. W. 2001. *Realistic Image Synthesis Using Photon Mapping (1st edition)*. AK Peters.

KENDRIK, R., AND KRONENBERG, G. 1986. *Photomorphogenesis in Plants*. Kluwer Academic Publishers.

LANE, B., AND PRUSINKIEWICZ, P. 2002. Generating spatial distributions for multilevel models of plant communities. In *Proceedings of Graphics Interface 2002*, 69–80.

LINTERMANN, B., AND DEUSSEN, O. 1999. Interactive modeling of plants. *IEEE Computer Graphics and Applications 19*, 1 (Jan./Feb.), 56–65.

MĚCH, R., AND PRUSINKIEWICZ, P. 1996. Visual models of plants interacting with their environment. In *SIGGRAPH '96 Conference Proceedings*, ACM Press, ACM SIGGRAPH, 397–410.

OPPENHEIMER, P. August 1986. Real time design and animation of fractal plants and trees. *Computer Graphics (SIGGRAPH '86 proceedings) 20*, 4.

PRUSINKIEWICZ, P., AND HAMMEL, M. 1994. Language restricted iterated functions, koch constructions and l-systems. *SIGGRAPH '94 Course Notes*.

PRUSINKIEWICZ, P., AND KARI, L. 1996. Subapical bracketed l-systems. In *Lecture Notes in Computer Science, Volume 1073*, Springer Verlag, 550–564.

PRUSINKIEWICZ, P., AND LINDENMAYER, A. 1990. *The Algoritmic Beauty of Plants*. Springer Verlag.

PRUSINKIEWICZ, P., JAMES, M., AND MĚCH, R. 1994. Synthetic topiary. *Computer Graphics 28*, Annual Conference Series (July), 351–358.

PRUSINKIEWICZ, P., HAMMEL, M., HANAN, J., AND MĚCH, R. 1996. L-systems: From the theory to visual models of plants. In *Proc. 2nd CSIRO Symp. Computational Challenges in Life Sciences*, CSIRO Publishing, P.O. Box 1139, Collingwood 3066, Australia, M. T. Michalewicz, Ed.

PRUSINKIEWICZ, P., MUNDERMANN, L., LANE, B., AND KARWOWSKI, R. 2001. The use of positional information in the modelling of plants. In *SIGGRAPH '01 Conference Proceedings*, ACM Press, ACM SIGGRAPH.

REEVES, W., AND BLAU, R. 1985. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *Computer Graphics (SIGGRAPH '85 proceedings) 19*, 3 (July), 313–322.

REEVES, W. 1983. Particle systems - a technique for modelling a class of fuzzy objects. In *ACM Transactions on Graphics, Vol 2, Nr 2*.

SILVERMAN, B. W. 1986. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall.

SOLER, C., SILLION, F., BLAISE, F., AND REFFYE, P. D. 2003. An efficient instantiation algorithm for simulating radiant energy transfer in plant models. *ACM Transactions on Graphics 2003 22*, 2 (Apr.), 204–233.

VAN HAEVRE, W., AND BEKAERT, P. 2003. A simple but effective algorithm to model the competition of virtual plants for light and space. In *Journal of WSCG*, 464–471.