

Environmental Framework to Visualize Emergent Artificial Forest Ecosystems

Aleš Zamuda and Janez Brest

*Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova ul. 17, 2000 Maribor, Slovenia
(email: ales.zamuda@uni-mb.si, janez.brest@uni-mb.si)*

Abstract

We propose an environmental framework for simulation and visualization of woody plant forests. A complex application software system develops and animates a spontaneous afforestation process within this environment. The system considers several environmental properties and combines computer animation with artificial life. The main goal of the presented software system is to use it in computer animation for synthesis of natural environments and visual analysis of their natural look credibility. The afforestation process is modeled as an ecosystem simulation, where trees struggle for survival based on several growth factors. A detailed description of the procedures for simulating tree growth and the factors that might influence tree growth is provided. All the tree growth simulation procedures and factors are biologically inspired. They have been defined mathematically in the paper by designing a bottom-up agent model which emerges the artificial tree distribution by mediating to the simulation.

A flexible and adaptable procedural 3D model is used to visualize trees. Also, growth of individual trees is animated, from development of branch complexity to per-leaf precision, which allows a very realistic perception of the emerging ecosystem. The visualization of trees is sped up so that the models of trees have progressively lower-details proportional to the distance from a certain point of view. Locations and maturity of visualized trees are obtained from the ecosystem simulation results, and the afforestation process is animated over several centuries. The natural look of the artificial tree distribution is confirmed visually and statistically. Visually, it is confirmed from rendered sequences, and statistically, from graphs of tree species populations. Several patterns emerge permanently, such as the number of trees in the ecosystem simulation increasing exponentially and trees growing in communities.

Keywords: Artificial life, natural phenomena, computational biomodelling, ecosystem simulation, woody plant, forest ecosystem, computer animation, tree modelling.

1. Introduction

In this paper, we present an environmental framework for the simulation and visualization of woody plant forest ecosystems. To be able to visualize forests of woody plants, we must first be able to render individual trees using tree models. After we have visual models of trees, we can place them in an environment according to some distribution and other environmental factors. In this paper, these are obtained by computer simulation for artificial life of woody plants. In the following of this introduction, related work on modelling of trees, ecosystem modelling, artificial life, and landscape visualization is discussed briefly.

The tree modelling has a twenty year tradition in computer graphics, from game engines to cinematic scenery. At the beginning, the focus was on modelling the geometrical structures of individual trees. Manual modelling of tree structure and its leaves is a tedious task, because each branch and leaf position, rotation, size, and texture must be appointed. Therefore, procedural tree models are used instead and

several following techniques for procedural models are available today. Let us briefly refer to some of the strongest representatives and their key distinguishing features. First, these procedural models differ in types of branching structure construction [36]. Also, they differ in the level of detail [2, 4, 39], the flexibility, and pretentiousness of modelling [19, 52], space [32, 47], and time complexity [32] in addition to the animation ability and representation of the built 3D model. The majority of these models try to determine some visible properties of the final 3D model, such as rotation of branches around their central axes. These properties are usually biologically inspired by *phyllotaxis*, i.e. the main influence on the tree's architecture [40].

Ecosystem simulation has been addressed in many fields, from biology to artificial intelligence [16, 21]. Ecosystem simulation is also used by computer graphics to render a 3D ecosystem scene into an image [9, 63]. To, generally, be able to simulate life, artificial life models exist, such as [38, 22, 27, 17]. In simulating life of trees within a forest ecosystem, the spontaneous development [46, 23] is one of the main processes of forest development which must be modelled. Following increases in computer power and computer graphics research, research interest from individual tree models rendering has grown to several trees rendering together in an image. Considering the limitations, the aim in modelling single trees is to obtain natural look when rendering. This aim in rendering several trees still persists, and can be represented by determining several environmental factors, such as locations of trees, their sizes, and interaction of trees with their environment. Manually defining these factors, even perhaps by using some help by general purpose 3D modellers (e.g. Maya, Lightwave), requires specialized knowledge and art, creativity, and labour time by the user. An automated tool can try to partly relieve the user in these requirements. Such tool tries to define tree locations and other considerable factors to get a naturally looking image. Type of trees to render at tree locations could be recognized from photographic imagery [41], or rather obtained by simulating an artificial ecological system and life within it.

The first to treat landscape visualization with forests and shrubs were Weber and Penn [52]. Their trees were represented by a procedural geometrical model and rendered with ray-tracing. As this model featured level of detail adaptation, they were able to visualize few thousand trees on a landscape. Trees on this landscape were manually placed and did not reflect any biological laws of spontaneous afforestation, this being one of considerable drawbacks of their approach.

Chiba et al. [6] represented trees using volumetric textures. An image was compiled by ray marching through voxels (3D image elements) and integrating their density and colour. Trees were placed on landscapes using Poisson disc sampling [7], so that a new tree was successfully added only if its randomly selected location was not too close to the location of some other tree. The simulation was simple and fast, but their rendering process took several hours and the trees were of low detail. Using such an approach they created images with up to twenty thousand trees on a landscape. Their simulation was more of a trial-and-error calculation and still, life was not simulated to obtain tree distribution.

Specialized for rendering ecosystems of woody plants, Deussen et al. [10] built the first ecosystem simulator used in computer graphics to follow some natural laws of spontaneous afforestation. Their highly detailed tree models were created using L-systems [35, 56, 51] and rendered using *ray casting* or *ray tracing*. Non-textured scene overview was possible with Gouraud shading by using OpenGL, but realistic image rendering took several hours and it was not intended for real-time animation. With so many ray-traced objects on scene, they came up with the conclusion that a compact procedural model for geometry generation and the *instancing* of created geometrical tree models must be used for objects to fit in the main memory. Another conclusion they drew is, that for rendered images not to look artificial, the perception of tree distribution in a scene should be authentic. Visually credible distribution of trees was achieved using two different approaches, one is from global to local, the other is from local to global. The first approach was semi-automatic requiring a standard paint program [10], the second was automatic calculation of the distribution, a pure simulation. This second approach, the simulation, included some natural laws of

spontaneous afforestation, the interaction among the trees themselves and the environment. By passing a limited number of parameters to the model, many emerging trees have grown over the landscape according to the modelled natural laws of spontaneous afforestation.

Let us elaborate on laws of interaction Deussen et al. included in their second approach, which actually were bottom-up software agents [15, 1, 44] to thrive the simulation by mediating to the environment. The first such agent was a tendency for the same species of trees to grow in clusters, emerging the effects of *clustering* [3], *clumping* [25], and *under-dispersion* [20]. The second agent Lane and Prusinkiewicz considered in [25] was competition among individual trees in an area of *ecologic neighbourhood* [8], where one tree can influence the growth of others. By intersection of two ecological neighbourhood areas, the involved trees interact. One of the trees becomes dominated and when a tree is being dominated, its growth is slower or it may even die. Such competition principally results in a phenomena of tree distribution known as *self-thinning*, meaning that in the beginning of simulation, trees grow without interference, but as density increases, the dominated trees start dying off. Deussen et al. [10] defined the domination of a tree by comparing its competitive ability to the competitive ability of competing trees within its range. If a tree has inferior competitive ability than a neighbour, it becomes dominated. The competitive ability depends on the age of the tree, domination tolerance, water concentration at its location, and its preference for wet or dry areas. Considering the living conditions, this is a rather simplified growth model, since it only considers the moisture. Lane and Prusinkiewicz [25] introduced a clustering mechanism to this simulation, which enabled trees to grow in communities.

Motivation and objectives of our work are as follows. We want to combine models that are efficient for visualization and allow tuning of the simulation in real-time during animation. We want, that our simulation allows a great number of individual trees to be simulated within a forest, and that after the simulation, the trees get to be placed according to emerging patterns, into a naturally-looking distribution. We want to use programmable agents to thrive the emerging patterns, by mediating to the developing ecosystem. We want the agents to encompass the ecosystem rules, such as environmental resources distribution, competition of trees for these resources, and tree growth. We also want the number of trees in the ecosystem to increase exponentially if there is enough resources, we want trees to grow in communities, and we want tree distribution to be biologically inspired and natural looking. Compared to the existing solutions, we want to propose a compact efficient integration of a procedural tree-model into the forestry landscape visualizer based on ecosystem simulation; build new parameterized procedural tree models within ecosystems; consider several tree living conditions for distribution of trees; have the ability to interactively influence the tree distribution simulation; and have the living conditions computed based on adjustable digital terrain data.

In computer graphics, the existing simplified living condition models only consider the soil moisture [10, 8, 25]. It is true that this is usually a significant environmental impact, but additional, sometimes even more important ones than only the soil moisture, can be found, e.g. in the Ellenberg taxonomy [13]. This is why we decided to build a new ecological model to consider more parameters (indicators). We defined¹ new models for tree growth, their distribution across landscapes, and the calculation of additional living conditions. For simulation, we use real terrain elevation data to compute living conditions and especially their relationships. The relationships between the Ellenberg indicator values are explored primarily in ecological literature, see e.g. [13, 18, 26]. We leave the habitat selection and model parameterization to users of our software tool and only provide them with computational models implementation behind it.

The paper is divided in six sections. The following three sections, from the second to the fourth, pro-

¹We have implemented all the defined models in C++ programming language; our application is available as EcoMod project on SourceForge at <http://sourceforge.net/projects/ecomod/>. See Appendix B for some pseudocodes.

pose our environmental framework to visualize emergent artificial forest ecosystems. The second section proposes the simulation of an ecological system. The third section proposes our models for calculation of living conditions within an ecosystem, where elevation data is used to calculate living conditions such as slope, soil moisture, and intensities of wind and sun. The fourth section explains our modelling of trees and proposes visualization of our simulated ecosystem. Discussion takes place in the fifth section. The paper concludes with a summary, and guidelines for future work. Also, there are appendices for symbols denotation and algorithm pseudocodes.

2. Simulation of an Ecological System

Our ecosystem simulation simulates life of trees within an ecosystem, i.e. the spontaneous afforestation process. With its help, for each tree species, we determine the distribution and sizes of individual trees across a terrain, based on the living conditions within this terrain. These living conditions are *height above sea level*, *slope*, *moisture*, and the intensities of *sun* and *wind*. We begin the simulation with an empty terrain on which, in each simulation step, a certain number of trees for each species is added. In each simulation step, we execute three sub-steps (see Appendix B, Algorithm 1): i.e. the creation of new trees, their growth, and the elimination of dead trees. These three sub-steps are implemented as software agents, that act stochastically². Our simulation step time difference is one year ($\Delta t = 1$ year), so that we can simulate the lives of several hundred thousand trees over several hundred years.

Let us first present the first sub-step of every simulation step in our simulation, the creation of new trees. The sub-step is done for each tree species using different parameters, because each tree species has a specific method of replication. First, $\Delta N_{g,s}$ trees for each species to be added is computed in each year. The amount varies according to the number of seeds from this species cumulated over the previous year, and its particular seed germination factor. When we add new trees, we consider the under-dispersion principle by increasing the probability of chosen locations close to existing trees of the same species.

Now, let us describe the second simulation sub-step, simulating the growth of trees. This sub-step depends upon the *competitive ability*³ of the trees. Using $a_{s,p}$ ($a_{s,p} \in [0, 1]$), we define the competitive ability of p -th tree from the tree species s :

$$a_{s,p} = h_{s,p} p_{persist;s} v_{s,p}, \quad (1)$$

where $h_{s,p} \in [0, 1]$ denotes the height of the tree, $p_{persist;s} \in [0, 1]$ the persistent clinging to life⁴, and $v_{s,p} \in [0, 1]$ the suitability of living conditions for a tree with indices s and p . The persistent clinging to life of a species, $p_{persist;s}$, is used to distinguish between pioneering species such as shrubs and birches and others such as beeches, spruces, and maples. It is a characteristic for the former to retreat if the latter have suitable living conditions in the same area.

The height of a tree depends upon its age and other factors. The first factor to consider is *growth obstruction*, caused through domination by more powerful trees within the neighbourhood. The second factor is *non-uniform growth intensity* throughout a tree's life (young trees grow faster than old ones). For the latter factor, we defined a polyline to approximate an ideal continuous growth curve for the height of a tree depending on its age. Fig. 1 shows the height of a tree over years without growth obstruction using the ideal continuous curve and its approximated polyline. For each tree species, the polyline control points are defined using *average maturity year of species* ($t_{m;s}$), *average year when species is fully grown* ($t_{a;s}$), *average height of a matured species* ($h_{m;s}$), and *maximum potential age of species* ($t_{f;s}$).

²The random number generator we use is *SUPER-DUPER* with notation $LCG(2^{32}, 69069, 0, 1)$ [29].

³The term was introduced by Deussen et. al. [10].

⁴As noted in Appendix A, descriptive indices for main symbol are separated from variable indices with a colon in this paper.

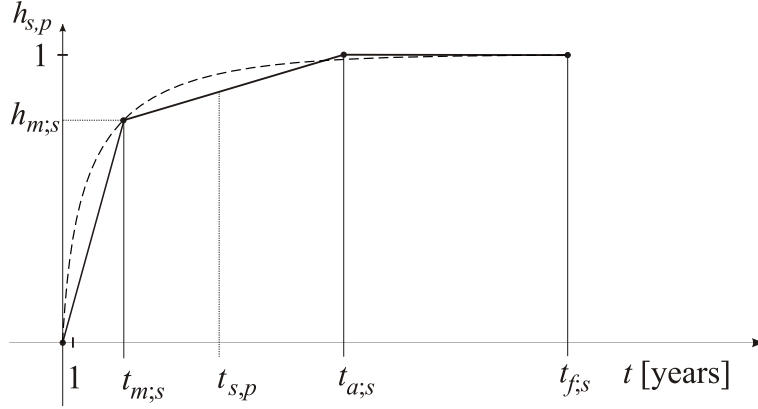


Figure 1: Height of a species through lifetime without growth obstruction.

The suitability of living conditions for a specimen is calculated as follows:

$$v_{s,p} = k_{\bar{y};s,p} m_{s,p} w_{s,p} l_{s,p} s_{s,p}, \quad (2)$$

where $k_{\bar{y};s,p} \in [0, 1]$ denotes the suitability of height above sea level, $m_{s,p} \in [0, 1]$ soil moisture, $l_{s,p} \in [0, 1]$ sun, $w_{s,p} \in [0, 1]$ wind, and $s_{s,p} \in [0, 1]$ slope at growth location. We define the latter by considering *slope harmfulness* ($f_{s;s}$) for each species:

$$s_{s,p} = 1 - (f_{s;s} s_{i,k})^2, \quad (3)$$

where indices i and k select the patch at growth location, to determine the slope. The suitability of the remaining four living condition factors depends on any deviation from the *optimal living conditions* of a species, defined as $\bar{y}_{o;s}$, $m_{o;s}$, $l_{o;s}$, and $w_{o;s}$ in the following equations:

$$k_{\bar{y};s,p} = \begin{cases} 1, & |\bar{y}_{i,k} - \bar{y}_{o;s}| \leq \Delta \bar{y}_{o;s} \\ 1 - \max \{1, |\bar{y}_{i,k} - \bar{y}_{o;s}| f_{\bar{y};s}\}, & \text{otherwise} \end{cases}, \quad (4)$$

$$m_{s,p} = \begin{cases} 1, & |m_{i,k} - m_{o;s}| \leq \Delta m_{o;s} \\ 1 - \max \{1, |m_{i,k} - m_{o;s}| f_{m;s}\}, & \text{otherwise} \end{cases}, \quad (5)$$

$$l_{s,p} = \begin{cases} 1, & |l_{i,k} - l_{o;s}| \leq \Delta l_{o;s} \\ 1 - \max \{1, |l_{i,k} - l_{o;s}| f_{l;s}\}, & \text{otherwise} \end{cases}, \quad (6)$$

$$w_{s,p} = \begin{cases} 1, & |w_{i,k} - w_{o;s}| \leq \Delta w_{o;s} \\ 1 - \max \{1, |w_{i,k} - w_{o;s}| f_{w;s}\}, & \text{otherwise} \end{cases}. \quad (7)$$

Here, *tolerated deviations* $\Delta \bar{y}_{o;s}$, $\Delta m_{o;s}$, $\Delta l_{o;s}$, and $\Delta w_{o;s}$ are auxiliary parameters for defining the classes of suitable living conditions for a species. If a certain living condition exceeds the tolerated deviation, its harmfulness is considered by diminishing a living condition with it, and a *harmfulness factor*. Latter factors are denoted as $f_{\bar{y};s}$, $f_{m;s}$, $f_{w;s}$, and $f_{l;s}$ for each living condition.

From Eqs. 4–7 we can observe the following. If an environmental factor drops below a certain threshold, it will negatively impact a tree's fitness. Its negative impact is linearly scaled in accordance with a weight that reflects the possible harmfulness of the environmental factor.

Listings of tree species living conditions can be obtained from literature. One such listing was prepared by Ellenberg [13] for Central Europe and extended to other areas of the world [18, 26]. Ellenberg divided living conditions within ten classes on a linear scale and classified suitable living conditions for individual species in these classes. Listings of statistical data can be read from a digital library [49] and used in our simulation.

The final, third simulation sub-step copes with the elimination of dead trees. We distinguish between three causes of death: *natural*, *intolerableness of living conditions*, and *suffocation*. We model natural cause of death as following. The year, when a tree is going to die naturally, is predetermined at its creation as a uniformly distributed random number between the $t_{a;s}$ and $t_{f;s}$.

We model the intolerance of living conditions, together with the growth of a tree. A seed only germinates under favorable environmental conditions.

Suffocation is caused if a tree is dominated by neighbouring trees. Generally, certain tree can be dominated by any other tree on terrain. For each tree A , we would eventually have to execute a test to determine a boolean flag of neighbourhood intersection $F_{A,B} \in \{0, 1\}$ where B runs through for all other trees in current population. This would result in an undesirable $O(n^2)$ algorithm. As if this is not enough, the test to be considered is not a simple integer comparison by subtraction, but a more complex 3D geometrical distance test. To evaluate $F_{A,B}$, we compare the radii of both trees:

$$F_{A,B} = \begin{cases} true, & d(A, B) < r_A + r_B \\ false, & otherwise \end{cases}, \quad (8)$$

where r_A and r_B denote the ecological radii of the two trees. If the distance between the trees is smaller than the sum of radii r_A and r_B , the ecological neighbourhoods intersect. The distance $d(A, B)$ between the trees A and B at locations \mathbf{p}_A and \mathbf{p}_B is defined by:

$$d(A, B) = \|\mathbf{p}_A - \mathbf{p}_B\|. \quad (9)$$

Therefore, for some tree A , the domination test in Eq. 8 would have to be done for each other tree than the tree A in the current population⁵. The optimization approach we took to improve the neighborhood search algorithm is localization of the algorithm to lower the order of the improved algorithm. To find trees nearby, we conduct a reference list on each patch to search for nearby trees much faster. The list contains pointers to trees, which grow in this part of terrain, and is constructed using uniform plane partitioning. For each search, all trees in few nearby patches are tested to determine any dominance of a certain tree. We conducted an experiment to present improvement in a real case. It was shown that with 600,000 trees in the 299th simulation year, on average only 540 instead of 600,000 tests were made per a tree. For a case with ideally uniformly distributed locations of trees, the complexity order of this algorithm is $O(n \log n)$.

3. Models for the Computation of Living Conditions in a Terrain

The mesh for the terrain is read from a digital terrain model (DEM) [28] and defined using vertices:

$$\mathbf{p}_{i,k} = [x_i \quad y_{i,k} \quad z_k]^T, \quad i, k \in [0, 99], \quad (10)$$

being sampled for every e.g. 25 m along x and z . $y_{i,k}$ denotes elevation of the (i, k) -th sample. From vertices, quad-like patches are formed, implying 10,000 patches for a sample terrain. From the data of these patches, living conditions are estimated for every patch using speed-efficient calculations. Height above sea level y is normalized for later use with other models:

$$\bar{y} = \frac{y - y_{min}}{y_{max} - y_{min}}, \quad (11)$$

⁵Time complexity order of this algorithm is $20 \cdot O(n^2)$ operations, where n is number of trees and operation considered is addition. Specifically, each test consists of three subtractions, three raisings to second power, three additions (to calculate distance between trees), another addition with raising to second power (to calculate minimal distance for interaction between both species), and a final subtraction for the comparison. In total this is approximately eight additions and four raisings to power of two or approximately a non-parallel execution timing of twenty additions disregarding pipeline speedup.

where y_{min} is the minimum height above sea level of the terrain, and y_{max} is its maximum. The surface normal of a quad (patch) is denoted by $\mathbf{n}_{i,k}$ and calculated as:

$$\mathbf{n}_{i,k} = (\mathbf{p}_{i+1,k} - \mathbf{p}_{i,k}) \times (\mathbf{p}_{i,k+1} - \mathbf{p}_{i,k}) = [n_{x;i,k} \quad n_{y;i,k} \quad n_{z;i,k}]^T, \quad (12)$$

where the first vector spans along neighbouring vertices in the first axis and the second one in the next axis. With $\bar{\mathbf{n}}_{i,k}$, the normalized normal is denoted. Slope is calculated by using y component of $\bar{\mathbf{n}}_{i,k}$:

$$s_{i,k} = 1 - \bar{n}_{y;i,k}. \quad (13)$$

A flat (horizontal) terrain has a slope of zero and a steep (vertical) terrain has a slope of 1. Using algorithms for simulating natural phenomena impact, we determine relative levels for terrain soil moisture, and average wind and sun distribution for the summer's half-year. We created these algorithms ourselves, but they have been compared with existing referential literature. Fig. 2 displays a distributions of these living conditions across terrain, obtained using our models.

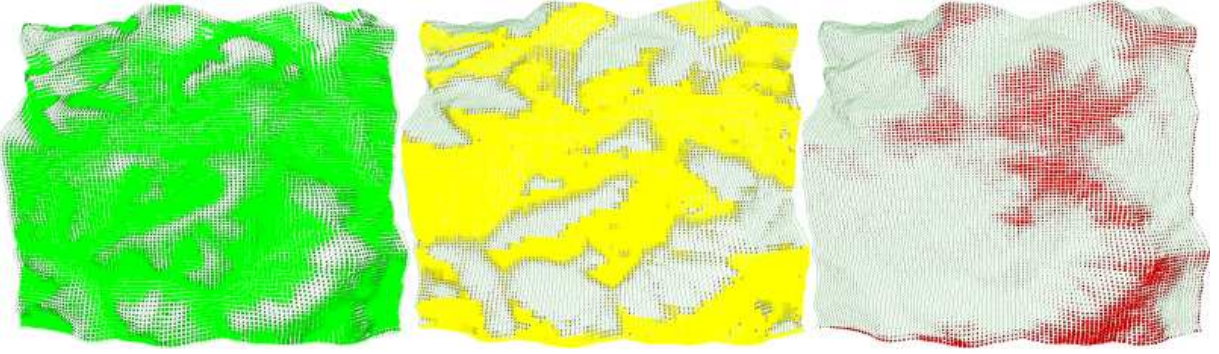


Figure 2: Intensities of soil moisture, wind, and sun for real terrain data obtained using our models on real terrain data.

3.1. Soil Moisture

Our soil moisture intensity calculation is based on factors of rain flow and stagnation. Rain flow is the main affecting factor of soil moisture, depending on slope, soil absorption rate, and terrain topology. The another factor is stagnant water body accumulation, such as rivers and lakes. The latter is included by initializing the respective areas to high moisture levels. We assume that rain fall is uniformly distributed across the whole terrain and, therefore, each patch receives the same quantity of water. Rain water that cannot be absorbed flows to nearby areas. All this results in nonuniform soil moisture distribution.

As explained, rain-water flows to or from neighbouring patches with a magnitude depending on their slope, height variation with neighbouring patches, and the soil absorption rate. Locally this means, the moisture received for certain patches equals the immediately absorbed rain and the absorbed part of the water flowing to this patch from higher patches. Soil absorption rate depends on soil structure and rain type (light rain, rain shower). In our calculation, the rain absorption rate is approximated as a portion of absorbed rain compared to all the rainfall, but a soil absorption rate map can be loaded too.

Patches forming the terrain have different surface areas when viewed in 3D space, but all have same areas in xz plane projections. To explain this assumption, let us remember that rain source is equally distributed across the terrain. If we assume in our model, that rain falls perpendicularly to the xz plane, we can assume that every patch (independent of its size) receives the same quantity of rain $m_{climate}$.

The algorithm for moisture calculation (see Appendix B) begins by initializing all patches with the rain quantity $m_{climate}$ (e.g. $m_{climate} = 1$). For each of these patches, moisture propagation (infiltration),

m_w , is added for all areas of stagnant water body accumulations (e.g. $m_w = 10$):

$$m_{i,k} := \begin{cases} m_{i,k} + m_w, & \bar{y}_{i,k} \leq \bar{y}_w \\ m_{i,k}, & \text{otherwise} \end{cases} \quad (14)$$

Patches containing water accumulations are defined as those for which $\bar{y}_{i,k}$ component is less or equal than normalized height of water level, $\bar{y}_w \in [0, 1]$, the latter being an adjustable parameter of the soil moisture calculation.

For liquid flow calculation, patches are sorted upon their height and iterated through, to imitate the water flow behaviour. Let us assume that currently processed patch is $s_{i,k}$. For this patch, number of flow away sides, $n_{neigh;i,k} \in [0, 4]$, is determined first, by counting lower neighbouring patches. To test whether a patch is lower, its central point is used. Soil absorption rate, k_m , is a controllable parameter, so that a complete water flow off for the patch is defined as:

$$\Delta m_{i,k} = \begin{cases} 0, & n_{neigh;i,k} = 0 \\ \frac{s_{i,k} k_m}{n_{neigh;i,k}}, & n_{neigh;i,k} = 1, 2, 3, 4 \end{cases} \quad (15)$$

This results in stronger water flow off for patches with steeper slope and weaker for those with a more gradual slope. The ran off water portion is added to lower neighbouring patches:

$$m_{i\pm 1,k\pm 1} := m_{i\pm 1,k\pm 1} + \Delta m_{i,k}. \quad (16)$$

All poured off portions are subtracted for original patch:

$$m_{i,k} := m_{i,k} - n_{neigh;i,k} \Delta m_{i,k}. \quad (17)$$

The obtained moisture distribution is further processed by convolving it with a low-pass digital filter (an efficient moving average FIR filter is used). The filter is executed $N_{mblur} = 10$ times to soften and widen the moisture accumulation peaks.

3.2. Wind

Our wind intensity calculation is based on wind-shelter position determination. To calculate wind shelter areas, our algorithm considers two main principles: wind flow is, in many ways, similar to liquid flow, moving horizontally but because of minor differences in temperature it changes its mass and is, therefore, subject to vertical movement too [50]. Both principles result in wind shelter areas being created behind hills, because wind flows forward horizontally, but does not move down quickly enough. Using wind drop speed and forward motion speed, a total angle of wind occlusion can be defined. We constrain the wind occlusion angle search to traversal of $N_w = 10$ [50] patches, each sample being K meters apart.

To calculate the final windiness $w_{i,k}$ on the patch $s_{i,k}$, several computation steps are followed once again. Because we want to determine an average windiness of patches on a yearly basis, wind over terrain is defined as an average wind vector: it has its strength and constant direction. We shall denote this average wind vector as \mathbf{w} , lying in the xz plane.

For each patch, our algorithm (see Appendix B) therefore travels in the opposite direction of wind vector $w_{i,k}$, and determines a maximum wind occlusion angle until the distance when the wind would surely drop enough. If a traversed patch is lower than the patch being processed for windiness calculation, it does not occlude wind impact, but if it is higher, it is possible that it does. Wind occlusion angle is defined as $\varphi_{i,k,w} = \angle(\mathbf{c}_{i,k,w}, \mathbf{c}_{i,k}, \mathbf{c}'_{i,k,w})$, where vertex $\mathbf{c}_{i,k,w}$ is center of the traversed patch, $\mathbf{c}_{i,k} =$

$[c_{x;i,k} \ c_{y;i,k} \ c_{z;i,k}]^T$ center of processed patch, and $\mathbf{c}'_{i,k,w}$ projection of vertex $\mathbf{c}_{i,k,w}$ to the $y = c_{y;i,k}$ plane (Fig. 3). Calculation of wind occlusion angle is defined as:

$$\varphi_{i,k,w} = \arctan \frac{c_{y;i,k,w} - c_{y;i,k}}{\|\mathbf{c}'_{i,k,w} - \mathbf{c}_{i,k}\|}. \quad (18)$$

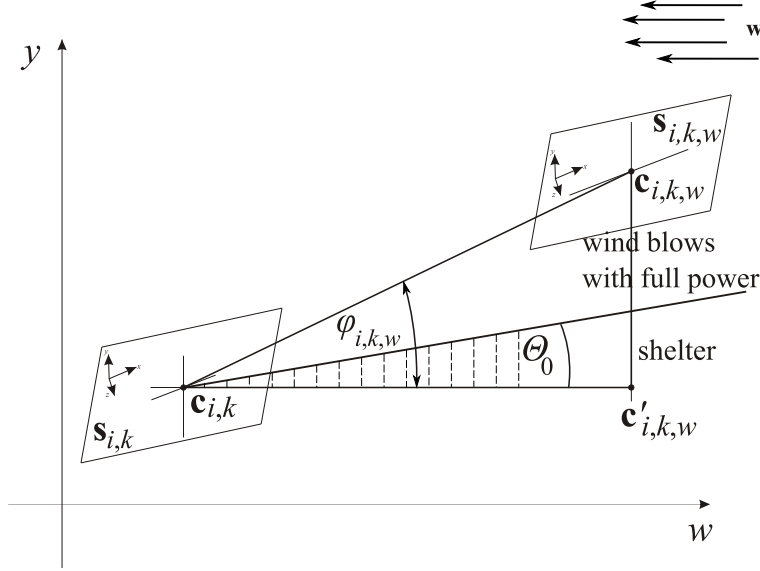


Figure 3: Angle of wind occlusion $\varphi_{i,k,w}$ between patch centers $\mathbf{c}_{i,k}$ and $\mathbf{c}_{i,k,w}$, for patches $\mathbf{s}_{i,k}$ and $\mathbf{s}_{i,k,w}$. Patch $\mathbf{s}_{i,k,w}$ corresponds to w -th sample with central point $\mathbf{c}_{i,k,w}$ from Fig. 4.

Wind occlusion rate is retrieved from minimum wind occlusion angle ($\varphi_{i,k} = \min_w \{\varphi_{i,k,w}\}$). Patch traversal for wind source vector is shown on Fig. 4.

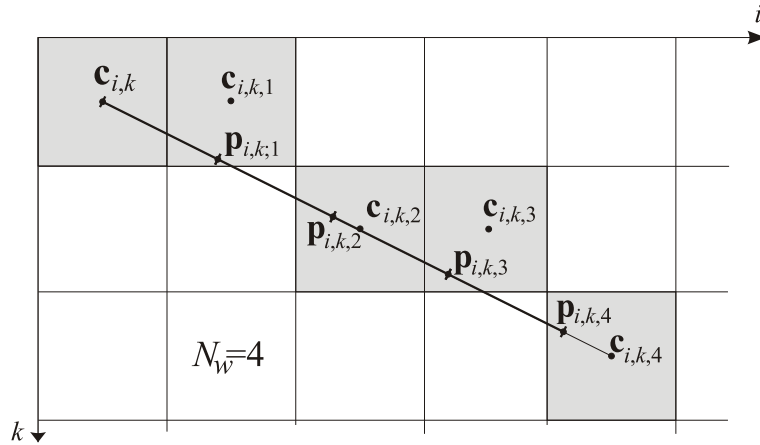


Figure 4: Patch traversal for wind source vector from vertex $\mathbf{c}_{i,k}$ for $N_w = 4$ samples. Traversal direction is defined as $-\mathbf{w}$ and single step length is same as the dimension of a patch, K . By sampling (dashes), appropriate patches and their central vertices $\mathbf{c}_{i,k,w}$ are obtained for wind occlusion angle calculation.

To finalize wind impact calculation $w_{i,k}$, obtained wind occlusion angle $\varphi_{i,k}$ is used and compared against threshold angle $\Theta_0 \in [0, \frac{\pi}{2}]$ (e.g. $\frac{\pi}{4}$). We defined the latter as an angle, where wind impact is fully occluded. Angle Θ_0 is schematically defined on Fig. 3. As can be reasoned from the figure, greater $\varphi_{i,k}$

angle results in smaller $w_{i,k}$, until it reaches Θ_0 threshold, when $w_{i,k} = 1$ equals to full wind impact:

$$w_{i,k} = \begin{cases} \varphi_{i,k}, & \varphi_{i,k} \leq \Theta_0 \\ 1, & \text{otherwise} \end{cases} \quad (19)$$

3.3. Sun

Our sun intensity distribution calculation is based on astronomical models for the Earth's rotation around the Sun. As the Earth rotates around the Sun and by itself, different areas of the terrain on the Earth receive different amounts of the Sun's radiation on surface [31, 5, 43]. What we do is sample the key Sun positions and accumulate the average Sun radiation, similar, but not quite the same, as in [59]. The latter is mainly dependant on the incidence angles of Sun's rays. Another important factor is hill shadows, preventing the Sun's rays breaking through, as in e.g. some valleys.

To deploy our algorithm (see Appendix B) for sunniness distribution calculation, we first need to parameterize it. Cardinal point vectors North \mathbf{p}_N , South \mathbf{p}_S , East \mathbf{p}_E , and West \mathbf{p}_W are prerequisite parameters. Because our model is adjusted for northern hemisphere, Sun is never directly in its zenith at noon, but rather some degrees lower to the South. This incidence angle distortion is caused by declination angle. For the Sun position on the sky sampling, we defined normalized vectors \mathbf{p}_0 , $\mathbf{p}_{\frac{N_l-1}{2}}$, and \mathbf{p}_{N_l-1} for starting, central, and end locations. To calculate all $\mathbf{p}_n, n \in [0, N_l]$ positions, some are interpolated from East to West, as seen on Fig. 5 and 6.

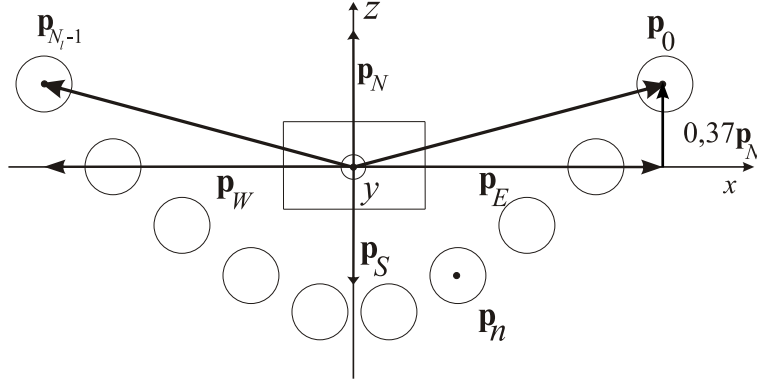


Figure 5: Samples of Sun positions on the sky, used for sunniness calculation of single patch $\mathbf{s}_{i,k}$, in two dimensions.

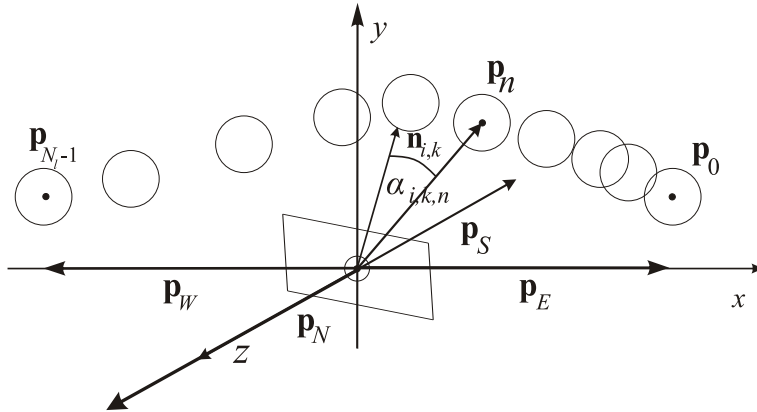


Figure 6: Samples of Sun positions on the sky, used for sunniness calculation of single patch $\mathbf{s}_{i,k}$, in three dimensions.

For our model, vectors \mathbf{p}_0 and \mathbf{p}_{N_l-1} can be defined as:

$$\mathbf{p}_0 = \mathbf{p}_E + 0.37\mathbf{p}_N, \quad (20)$$

$$\mathbf{p}_{N_l-1} = \mathbf{p}_W + 0.37\mathbf{p}_N, \quad (21)$$

where $0.37\mathbf{p}_N$ vector denotes Sun position on the East and West, moved to the North because of appropriate declination angle (derivated in the following subsection, 3.4).

The mentioned declination angle is incorporated into central sample $\mathbf{p}_{\frac{N_l-1}{2}}$ too. Let us define it for 46th parallel of latitude (this is where declination angle equals 46°). Here, the incidence angle at noon $\alpha_{i,k,\frac{N_l-1}{2}} = 90^\circ - 46^\circ = 44^\circ$ results in $\mathbf{p}_{\frac{N_l-1}{2}} = (0, \sin 44^\circ, \cos 44^\circ)$, because the Sun is exactly between East and West at this time. Now we can define the samples for location of the Sun on the sky:

$$\mathbf{p}_n = \begin{cases} \mathbf{p}_0 + \frac{2n}{N_l-1}(\mathbf{p}_{\frac{N_l-1}{2}} - \mathbf{p}_0), n = 0, 1, \dots, \frac{N_l-1}{2} \\ \mathbf{p}_0 + \frac{2(n-\frac{1}{2})}{N_l-1}(\mathbf{p}_{N_l-1} - \mathbf{p}_{\frac{N_l-1}{2}}), n = \frac{N_l-1}{2} + 1, \dots, N_l - 1 \end{cases} \quad (22)$$

We then normalize the calculated location vector \mathbf{p}_n to determine contribution of this sample according to its incidence angle. The latter is defined as $\alpha_{i,k,n}$, as seen from Fig. 6. As it is known, the radiation contribution is proportional according to cosine law, therefore only cosine of the $\alpha_{i,k,n}$ must be obtained. Cosine of the incidence angle is therefore calculated using normal $\bar{\mathbf{n}}_{i,k}$ and current Sun location vector \mathbf{p}_n :

$$\cos \alpha_{i,k,n} = \bar{\mathbf{n}}_{i,k} \cdot \frac{\mathbf{p}_n}{\|\mathbf{p}_n\|}. \quad (23)$$

Suniness contribution of a single sample $l_{basic;i,k,n}$, when Sun is at sample location \mathbf{p}_n , is defined as:

$$l_{basic;i,k,n} = \begin{cases} \cos \alpha_{i,k,n}, & 0 \leq \alpha_{i,k,n} \leq \frac{\pi}{2} \\ l_{ambient}, & \text{otherwise} \end{cases} \quad (24)$$

Contribution amount $l_{basic;i,k,n}$ is weighted, because samples near noon contribute more power, because atmosphere is warmer and clearer so Sun rays penetrate easier. Radiation contribution from single sample is therefore:

$$l_{i,k,n} = \begin{cases} l_{basic;i,k,n} \left(1 + \frac{2n}{N_l-1}\right), & 0 \leq n \leq \frac{N_l-1}{2} \\ l_{basic;i,k,n} \left(3 - \frac{2n}{N_l-1}\right), & \frac{N_l-1}{2} < n \leq N_l - 1 \end{cases} \quad (25)$$

As mentioned, because of the emergence of shadows, some rays can not penetrate into some areas. To determine, if a certain patch is in shadow, a similar algorithm was used as in the determination of wind shelter positions. In this case, traversal is done from a current patch towards the Sun's location. The height of the patches is used during traversal of patches towards \mathbf{p}_n . If the central point $\mathbf{c}_{i,k;n}$ of some n -th patch forms a smaller incidence angle with the central point $\mathbf{c}_{i,k}$ of the starting patch, this patch is preventing the Sun's rays from radiating onto the starting patch. As soon as some occluding patch is found, the search is finished for the current sample. The mentioned incidence angle is defined in the same as for wind calculation, using Eq. 18. If a certain patch is in shadow for some Sun's sampled position, that sample's radiation is diminished according to the ambient light coefficient parameter controllable by the user of the model, $l_{ambient} \in (0, 1)$ (e.g. $l_{ambient} = 0.2$ for non-cloudy days).

Suniness of $\mathbf{s}_{i,k}$ patch is accumulated from all sampled positions of the Sun:

$$l_{i,k} = \sum_{n=0}^{N_l-1} \frac{l_{i,k,n}}{N_l}. \quad (26)$$

3.4. Derivation of Average Sunniness Angle During Summer Time Equinoxes

In the model of sunniness, average angles for sunrise and sunset are taken through the springtime and summertime (from spring equinox 21 March to autumn equinox 23 September) to calculate the \mathbf{p}_0 vector. We obtained a calculated angle of approximately $\alpha = 21.98^\circ$ from North and therefore add to the Sun position vector \mathbf{p}_0 a vector of $0.37\mathbf{p}_N$. Details for the obtained result follow.

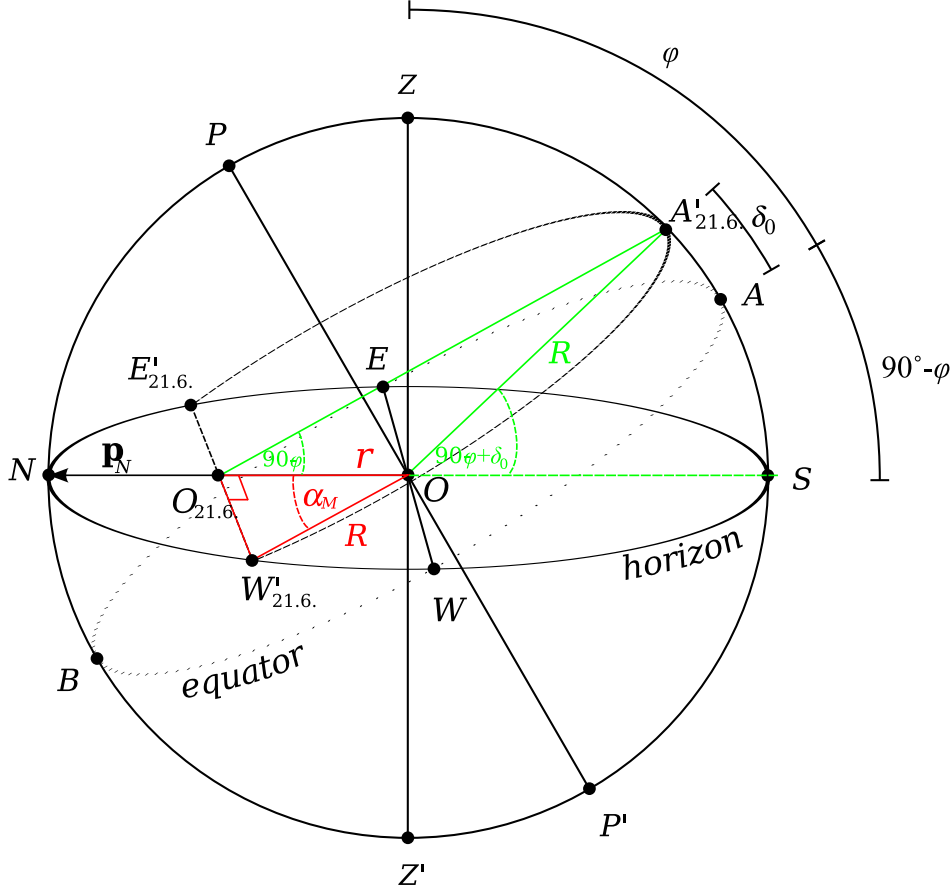


Figure 7: Vertices and rings of the celestial sphere.

It is known that on 21 March and 23 September the sunrise is at exactly at East and the sunset is exactly at West. At that time the Sun at equator is perpendicular to the Earth's surface, resulting in an equinox with equal duration of day and night. From one to another equinox in summer half-year time on the northern hemisphere the Sun rises somewhat more to the North and also goes down somewhat more to the North. This means that the Sun's position viewed from the surface of the Earth traverses more than an angle of 180° . To obtain an average angle of how much the Earth rises towards to the North we must first determine its extreme value α_M which sets time on 21 July, ie. at the summer solstice.

The part of Central Europe (Slovenia), for which our sample calculation is parameterized lies on latitude of 46 degrees ($\varphi = 46^\circ$). In the time of summer solstice the inclination of $\delta_0 = 23.5^\circ$ the angle of Sun on the sky is $90^\circ - 46^\circ + 23.5^\circ = 67.5^\circ$, seen as the angle $\angle A_{21.6}.OS$ on Figure 7. On the latter Figure, O denotes the center of celestial sphere. Z denotes zenith, Z' nadir, P North celestial pole, P' South celestial pole, E , W , S , and N the East, West, South, and North. The line through ZO is called perpendicular axis and the line through PP' is celestial axis. The spline EAW defines the celestial equator, and the ESW denotes the horizon. The spline NPZ defines the celestial meridian. φ denotes the geographical latitude of the viewer. δ_0 denotes the offset in degrees from equinox and is in the time of the summer solstice equal

to 23.5° . Symbols $E'_{21.6}$ and $W'_{21.6}$ denote vertices of sunrise and sunset at the day of summer solstice on a point with latitude $\varphi - \delta_0$. The spline through these points and the $A'_{21.6}$ lies in a plane perpendicular to $EA W$. Red colored triangle gives the studied angle α_M . From this angle, later the average angle α is calculated. The angle α_M is between the vector of Earth's radius R , in the direction to sunrise at the day of summer solstice and vector to the North with length of r .

From the angle $\angle A_{21.6} O_{21.6} S = 44^\circ$ we can calculate using the sine law the ratio between lengths r and R . The ratio between the Earth's radius R and the part of radius which intersects the horizon (ESW : East-West-South-West) is:

$$\frac{r}{\sin \angle O_{21.6} A_{21.6} O} = \frac{R}{\sin \angle O O_{21.6} A_{21.6}}, \quad (27)$$

$$\frac{r}{\sin (180^\circ - (90^\circ - \varphi) - (180^\circ - (90^\circ - \varphi + \delta_0)))} = \frac{R}{\sin (90^\circ - \varphi)}, \quad (28)$$

$$\frac{r}{\sin 23.5^\circ} = \frac{R}{\sin 44^\circ}, \quad (29)$$

$$\frac{r}{R} = \frac{\sin 23.5^\circ}{\sin 44^\circ}. \quad (30)$$

The angle α_M in the lower right-angled triangle between side r and hypotenuse R is:

$$\sin \alpha_M = \frac{r}{R}, \quad (31)$$

$$\alpha_M = \sin^{-1} \frac{r}{R} = \sin^{-1} \frac{\sin 23.5^\circ}{\sin 44^\circ} \approx \sin^{-1} 0.574 \approx 35.031. \quad (32)$$

Although the Earth rotates around the Sun in a elliptical trajectory, we will approximate this rotation with a circle. After this approximation the angle α_M is changed during time with a sine law if we take $t = 0$ as a spring equinox.

To calculate average angle α during summer time half year, we must calculate half of the whole-year sinusoidal period lasting approximately $t_0 = 365.25$ days in total. The angle α amounts in the spring equinox ($t = 0$) to $\alpha = 0$, in the summer solstice ($t = \frac{t_0}{4}$) to $\alpha = \alpha_M$, and in the time of autumn equinox ($t = \frac{t_0}{2}$) to $\alpha = 0$. Because the time sinusoid for Sun traversal is symmetric we must calculate only the average of the first quarter. First we calculate the integral to calculate the average of the angle looked for:

$$\omega = \frac{2\pi}{t_0}, \quad (33)$$

$$\alpha_{sum} = \int_0^{\frac{t_0}{2}} \alpha_M \sin(\omega t) dt = 2\alpha_M \int_0^{\frac{t_0}{4}} \sin(\omega t) dt = 2\alpha_M (-\cos(\omega t)|_{\frac{t_0}{4}} - (-\cos(\omega t)|_0)), \quad (34)$$

$$\alpha_{sum} = 2\alpha_M(0 + 1) \approx 70^\circ. \quad (35)$$

To get average, the result of the integral is divided by time during which the values are accumulated, ie. the summer half of year during equinoxes ($\Delta t = \frac{t_0}{2}$):

$$\alpha = \frac{\alpha_{sum}}{\frac{t_0}{2}} = \frac{70^\circ}{\frac{365.25}{2}} \approx 0.383 \approx 21.980^\circ. \quad (36)$$

To calculate Sun position \mathbf{p}_0 we must add the vector pointing towards North which is of length of the sine of the angle α :

$$\mathbf{p}_0 = \mathbf{p}_E + \sin(\alpha)\mathbf{p}_N \approx \mathbf{p}_E + 0.374\mathbf{p}_N. \quad (37)$$

4. Ecosystem Landscape Visualization

In our application, we can analyse simulation results quantitatively using 2D graphs, e.g. population size of each species, average age of species, or the clustering of species during simulation. In this section we will first present a simulation example to show population dynamics behaviour in our simulation. Then, we will focus on efficiently visualizing simulation results and round up with architectural configuration.

Fig. 8 shows graphs for a case, where simulation lasted for several hundred simulation years. Abscissa increases years by one for each simulated step, while the ordinate counts the population size of each species. Initially, there are enough resources for all species, so their population sizes grow exponentially. Shrubs are a pioneering species so they propagate at a fastest pace. After 200 years, the terrain has long been saturated and only the number of willows changes slightly. Species population sizes are now fairly stabilized, with a total population size of over 600,000 trees in the ecosystem, including all ages of trees, from newly germinated seeds to older trees.

To test simulation stability, we enforce an ecological catastrophe in the year 300 by killing all trees and not changing any simulation random generators. This is an arbitrary catastrophe (it could be e.g. fire). We have set that on average out of 100,000 trees only 100 survive. This is to test and analyze stability of patterns concerning species distribution over time. Also, this tests how the simulation proceeds if we do not begin with an empty terrain but rather already germinated seeds – in this case the shrubs do not reach such great number and are suffocated quicker by stronger trees progressing faster now. The ecosystem recovers with similar population sizes for each species as before the catastrophe. In year 600, we trigger another catastrophe by killing only 99.9% of trees and eliminating all shrubs and willows during this progress. But the ecosystem recovers once again. As can be seen, shrubs get eliminated again, but the willows also. They become extinct because other competitive species populate former growth areas of the willows and they do not have enough seeds to recover.

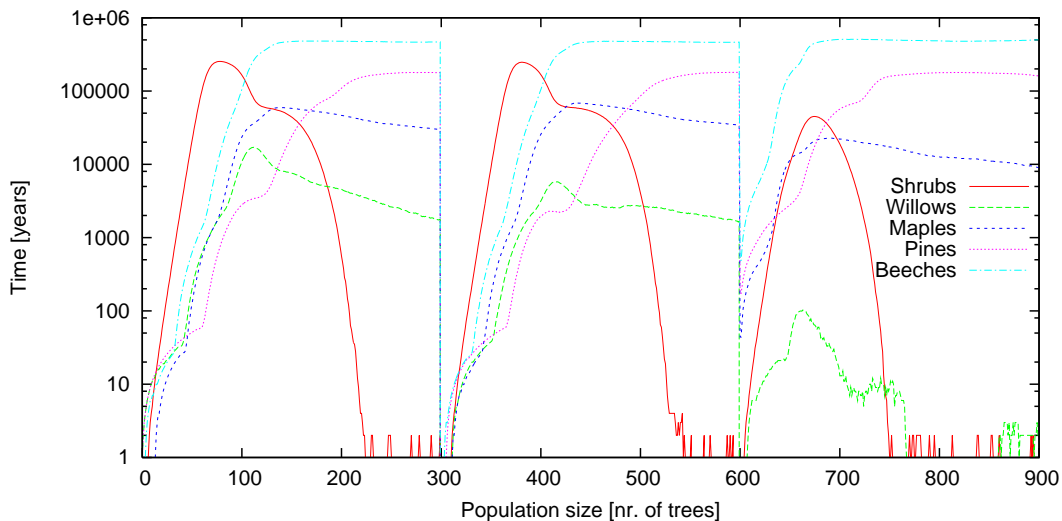


Figure 8: Exponential propagation of trees, stabilization and recovery after catastrophes in years 300 and 600.

The simulation results influenced by integrated agents are visualized using computer graphics by synthesizing rendered images of simulated ecosystem [19, 30, 6, 9, 12, 48, 33, 42]. It is that animation is guided by simulation and the simulation is visualized by animation. We visualize each simulated year by drawing a terrain surface and adding individually different geometrical models of trees at their growth locations.

Several free visualizers for fast terrain rendering using OpenGL can be found on the Web [34, 11]. Several of them use OpenGL extensions [45, 54, 55] for faster real-time rendering of e.g. 100×100 vertex mesh at hundred frames per second. OpenGL shader quality is usually lower than ray-tracing shader quality, but its speed is much better. Memory utilization is also lower with OpenGL because of its depth buffer algorithm. Every geometrical object (e.g. geometry of a tree) is namely disposed right after it is sent to OpenGL server, because it is not needed for rendering any more. OpenGL server also disposes the received geometry right after it has rendered it into image and updated the depth buffer.

To visualize the distribution of trees across terrain, a schematic or a photo-realistic image is rendered. The schematic image is composed of circles being drawn at locations of trees with extent of their ecological neighbourhood area radii. As all circles for same species are also in same colour, this is appropriate for schematic analysis. A photo-realistic image is rendered in order to visualize the distribution of trees across a terrain.

For the image, at each tree location our procedural models are used to instantiate the adapted and individualized geometrical models, for more realistic visualization. The per-species linked visual models are adapted according to the age of a tree. Parameterized procedural models are obtained using our interactive modeller part in order to instantiate these geometrical models. Using the modeller, related data for parameterizable procedural model is acquired by the design of graphs and the use of interactive dialogues, which immediately affect the displayed image of a tree. This certainly eases modelling and allows rapid model creation. As the user parameterizes the procedural model with a limited set of intuitive parameters, the underlying procedural model calculates and instantiates a geometrical model for a tree. The 3D position, size, orientation, and texture are calculated in order to create a geometrical model, for each of up to over thousand branch segments and several ten thousand leaves. Our procedural model can also be designed within an ecosystem with other trees, for easier perception [62]. The ergonomics of the modeller enables us to obtain a new procedural model within twenty minutes or less. In addition, the procedural model enables a fairly controllable, but flexible, animation. Each leaf and each branch segment can be animated in real time by changing some adjustable procedural model parameters, e.g. age of a tree. Those animation effects already included in our modeller are the growth of a tree and the sway of the tree in the wind. Fig. 9 displays some trees, which were visualized using our procedural models. As seen, coniferous and foliage trees can be obtained, which have very different branching structures and leaf distribution.

For the visualization of complete ecosystems, we upgraded our tree modeller with three major improvements and integrated it into the ecosystem simulator, where the tree-visualization part was used to visualize a full range of different trees in the terrain. The first improvement is the use of culling [14] for the removal of objects from the scene, so that the geometry is never computed for culled trees. The second improvement is the level of detail reduction for distant trees. Dependent on reduction degree, less smaller branches are drawn. Our reduction degree is defined by:

$$l_{LOD} = \left\lfloor \sqrt{\frac{\|\mathbf{p}_v - \mathbf{p}_{s,p}\|}{5}} \right\rfloor, \quad (38)$$

where \mathbf{p}_v denotes point of view and $\mathbf{p}_{s,p}$ bounding sphere center of p -th tree from s -th species (distances appear in meters).

The appearance of simplified trees differs little, but visualization process speedup is by up to several hundred times. Using these two improvements, we can already afford real-time animation of several thousand trees. The third improvement pushes visualization performance even further. According to the above reduction degree, differently-complexed geometrical objects are chosen for branch segment and leaf representation. Fully detailed branch segment and leaf representations are drawn with a cone and a quad. A simplified branch segment and leaf are drawn using a line and a vertex.



Figure 9: Trees obtained using our modeller [62].

4.1. Synthesized Images

We present some rendered images, which are synthesized from one of the simulation runs. Four tree species are included: beech, maple, pine, and shrubs. Simulation begins with an empty terrain with no trees. The trees populate the landscape, as a natural consequence if people stop mowing grassland (Fig. 10a). The first few years mostly include these, because shrubs can grow and replicate quickly (Fig. 10b). As they are weak in their fight for space against new dominating species, they retreat and the terrain is populated by larger foliage plants (Fig. 11a). Later, spruces start to spread from the right side (Fig. 11b). After approximately 180 simulated years the terrain is saturated with all species used in simulation. Each of them acquires a suitable area for living conditions, where it can dominate other species. Obtained clustering of species is seen, but some trees from another species are still contained within the envired areas making the image more authentic.

4.2. Architectural Configuration

Architectural configuration of the built application is shown in Fig. 12. EcoMod application is composed of both tree modelling component and ecosystem modelling component. Each component is finely granulated and they interact with each other at top level by the latter using constructed procedural models of the first to create geometrical models for growing trees. As seen from the figure, user parameter input is saved in the databases of procedural model and ecological model parameters. The links of ecological models with their visual representations using procedural geometrical models, are saved in the former database.

5. Discussion

We are presenting an approach for the *modelling of ecosystems*, which combines computer graphics and artificial life in a new way. Aim of this work is an attempt to combine a scientific forest growth model with visualization, potentially useful in computer games, cinematic sequence rendering, scientific data visualization, or other application domains. A cheaper *computer animation* of natural phenomena is

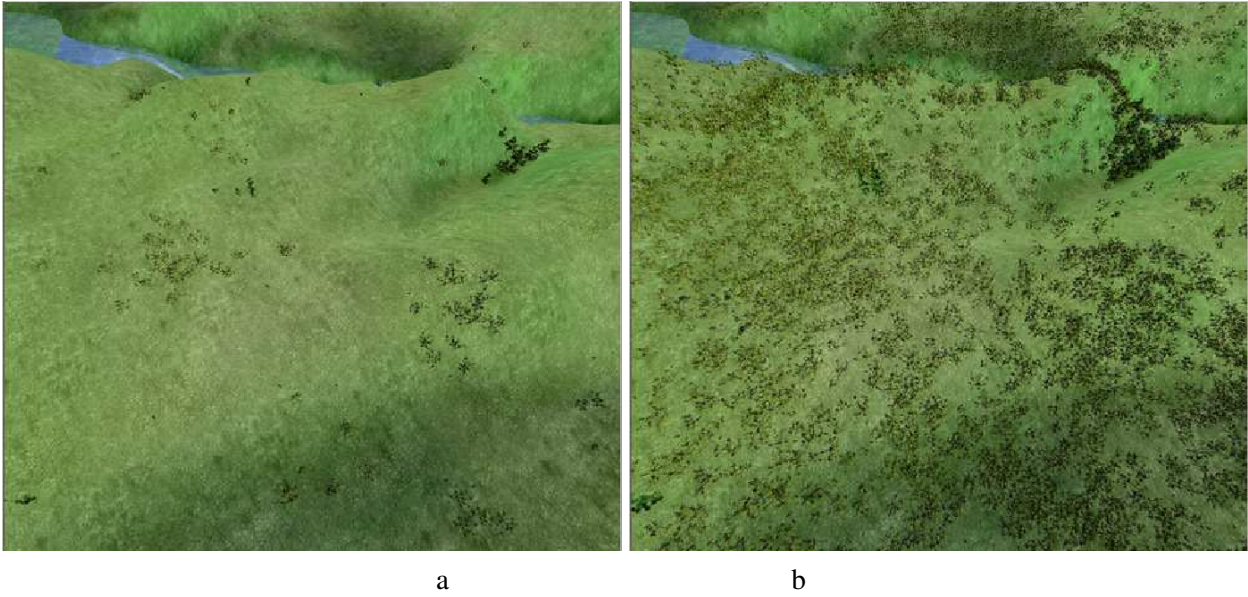


Figure 10: a) Beginning of simulation, b) People stop mowing grassland, shrubs begin to appear.

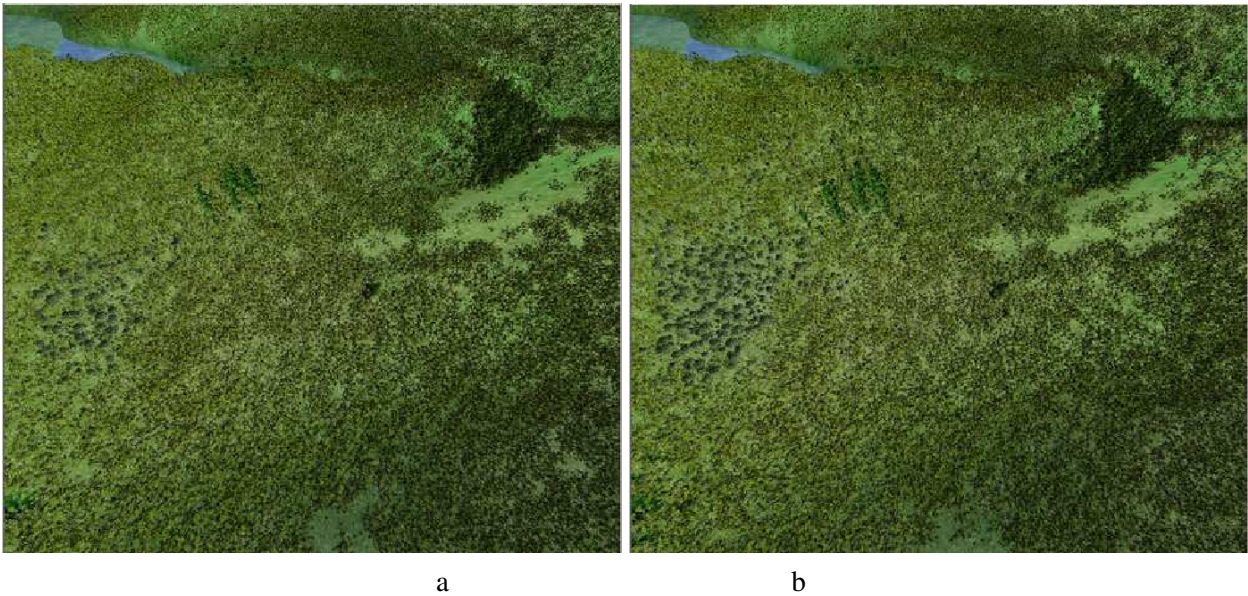


Figure 11: a) Simulation after 60 years: terrain is saturated, bigger tree species begin to prosper and replace shrubs, b) Results after 80 years: pines start to penetrate massively, whereas beech and maple are already distributed finitely across the terrain according to living conditions, and their interaction.

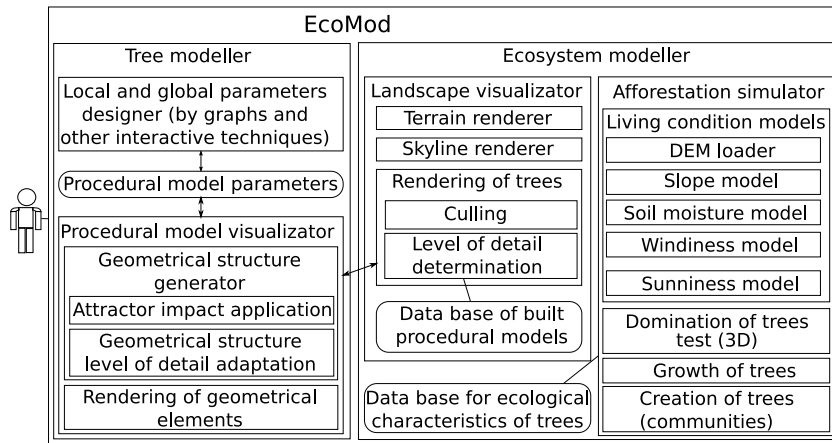


Figure 12: Architectural configuration of our environmental framework application components and implemented algorithms.

achieved by combining both scientific topics. As seen again, computer animation is still and once again the leading economical driver of research into artificial life.

The distribution of trees just *emerges* within the simulation, as this emergence is a principal phenomena of *artificial life*. Simulation is driven by programmable agents which mediate within the development. They are the keepers of rules within the ecosystem, such as the growth of individual trees, competition between neighbouring trees, the dying of trees, the manner of reproduction, and the disposal of living conditions. The emerged forest patterns can be observed on the rendered images presented in Figures 8, 10, and 11.

Let us stress, what are main contributions of our algorithms in relation to existing solutions:

- first, considering efficiency, we integrate a procedural tree-model modeller tool (subject of human-computer interaction), ecosystem simulator (subject of artificial life), and forestry landscape visualization (subject of computer animation) in a single application, with
 - ability to shape a new compact procedural tree model on a real landscape at a selected DEM location, and efficient reduction degree visualization of the built model within it,
 - immediate forest visualization of a simulated ecosystem during simulation, and
 - efficient generalization of ecological neighbourhood area collision detection for domination determination, from 2D into 3D space;
 - open source implementation of a new OpenGL engine with high applicability for reconfigurable vegetation rendering in computer games,
- second, in visualization of an ecosystem, we consider several living conditions besides soil moisture, such as height above sea level, wind, and sun,
 - we construct and define by pseudocodes the algorithms for calculation of living conditions,
 - we generate rendered ecosystems, which have estimated ecological factors calculations based on real DEM terrain data, and
 - we use living condition factors from Ellenberg taxonomy, in forest development simulation for visualization of ecosystems.

The main limitations of our work are:

- since expressed as computationally efficient simulation model, our model can not express the full complexity of a real life ecosystem, only its coarse approximation,
- the simulation model does not take into account animal interaction, human pollution, and human harvesting of forests,
- our simulation model does not take in consideration a temperature indicator separate from light, and only encompasses both in a sunniness indicator; also, our windiness and sunniness calculation does not take in account shadows of independent trees to nearby trees, and
- the simulation model runs have not been verified against some existing real ecosystem data; also, since there is no existing database of indicator values for all Earth's habitats, our forest simulation model has not been parameterized for the whole planet Earth.

6. Conclusion

In the paper, we provided a detailed description of the model for simulating spontaneous forest growth process and the environmental factors that might influence this process. All the tree growth factors and procedure have been defined mathematically, and the simulation could be based on a bottom-up agent model. The software simulation system has been designed for tree distribution during landscape spontaneous afforestation. It considers several environmental properties and combines computer graphics with artificial life. The application is used in computer animation for the synthesis and analysis of natural environments. A flexible and adaptable procedural 3D model is used to visualize trees. Behaviour of the presented tree distribution model was also confirmed visually. The number of trees in the ecosystem increases exponentially, trees grow in communities, and permanently emerging patterns are biologically inspired and natural looking.

There are plenty of guidelines for future work. Simulation could be additionally optimized and run parallel to several computers. The created model for forest growth can be parameterized using Ellenberg taxonomy [13], a well established classification method for living conditions impact on distribution of trees. Our created models of natural laws and their interaction should be fitted to real-world experiments by using large scale global optimization techniques [57, 60, 24, 53] and multi-objective optimization [61, 37] using niching techniques on the objectives [58]. Then, in-depth analysis of complex patterns as emerged from local behaviors and the impact of the underlying tree growth mechanism on the global tree distribution could be analyzed for real-world scenarios. Also, impact of pollution to forest growth could be shown by entering pollutant source locations and strengths. Last, but not least, our work could be used in simulator engines of virtual life for computer games and other virtual realities.

Acknowledgement

Authors would like to acknowledge the appreciated editor and anonymous reviewers which have helped much in improving this paper with their constructive comments and their valuable time.

Appendix A. Symbolical Denotation

Here we list the mathematical symbols defined for the terms used. The indices p and s , e.g. (p, s) , denote a woody plant, i.e. tree, and a species index, respectively. The i and k indices denote terrain patch indices. The indices are separated by comma. We sometimes also use main symbol descriptors in subscript – these are separated by semicolons from indices.

Ecological System Model		Soil Moisture	
$a_{r;p}$	maximum competitive ability of p -th tree	k_m	soil absorption rate
a_A	competitive ability of some tree A	$m_{climate}$	climate average rain quantity per patch
$a_{s,p}$	competitive ability of p -th tree of s -th species	$m_{i,k}$	patch $s_{i,k}$ moisture (rain quantity, filtered)
$d(A, B)$	distance between points \mathbf{p}_A and \mathbf{p}_B	$m_{o;s}$	s -th species optimal moisture
$F_{A,B}$	neighbourhood intersection boolean flag	$\Delta m_{o;s}$	species s tolerated moisture deviation
$f_{\bar{y};s}, f_{s;s},$	harmfulness factors intolerances for height	$m_{s,p}$	tree (p, s) soil moisture suitability
$f_{m;s},$	above sea level, slope, moisture, windiness,	m_w	stagnant water body accumulations area size
$f_{w;s}, f_{l;s}$	and sunniness, respectively	y_{min}	minimal terrain height above sea level
$h_{s,p}$	height in meters, for a tree (p, s)	y_{max}	maximal terrain height above sea level
$h_{m;s}$	species s trees height reached when mature	$\Delta m_{i,k}$	patch $s_{i,k}$ complete water flow off
N_{opt}	maximum distance in number of patches, for two possibly neighbouring trees	N_{mblur}	moisture accumulation peaks filtering count
$\Delta N_{g,s}$	species s new trees count for each generation	$n_{neigh;i,k}$	patch $s_{i,k}$ number of rain flow away sides
\mathbf{p}_A	location of a tree A on terrain, in 3-D space	\bar{y}_w	normalized height of water level
$p_{persist;s}$	persistent clinging to life factor of species s	Wind	
r_A	ecological neighbourhood radius of a tree A	$\mathbf{c}_{i,k,w}$	traversed patch center for w -th wind sample
$t_{s,p}$	age of a tree (p, s)	N_w	wind traversal patch count
$t_{f;s}$	maximum potential age of species s	\mathbf{w}	average wind vector, defined in xz plane
$t_{m;s}$	average maturity year of species s	$w_{i,k}$	patch $s_{i,k}$ windiness
$t_{a;s}$	average year when species s is fully grown	$w_{o;s}$	s -th species optimal windiness
$v_{s,p}$	vigour of a plant (p, s)	$\Delta w_{o;s}$	species s tolerated windiness deviation
		$w_{s,p}$	tree (p, s) patch windiness
Terrain Elevation and Slope		$\varphi_{i,k}$	minimum wind occlusion angle for patch $s_{i,k}$
$\mathbf{c}_{i,k}$	central point of patch $s_{i,k}$	$\varphi_{i,k,w}$	patch $s_{i,k}$ wind occlusion angle, w -th sample
$\mathbf{c}'_{i,k,w}$	w -th wind point sample projection, patch $s_{i,k}$	Θ_0	wind occlusion threshold angle
(i, k)	patch indices for x and z axis, respectively	Sun	
$k_{\bar{y};s,p}$	tree (p, s) suitability of height above sea level	$l_{ambient}$	shadowed patches ambient light coefficient
K	dimension in meters of a patch, in plane xz	$l_{basic;i,k,n}$	potential n -th sunniness contribution for $s_{i,k}$
$\mathbf{n}_{i,k}$	normal vector at point $\mathbf{p}_{i,k}$ of patch $s_{i,k}$	$l_{i,k}$	patch $s_{i,k}$ sunniness
$\mathbf{n}_{x;i,k}, \mathbf{n}_{y;i,k},$	components of terrain patches normal vectors	$l_{i,k,n}$	patch $s_{i,k}$ sunniness for n -th Sun position
and $\mathbf{n}_{z;i,k}$		$l_{o;s}$	s -th species optimal sunniness
$\bar{\mathbf{n}}_{i,k}$	unit normal vector at point $\mathbf{p}_{i,k}$	$\Delta l_{o;s}$	s -th species tolerated sunniness deviation
$s_{i,k}$	terrain patch slope	$l_{s,p}$	tree (p, s) patch sunniness
$\mathbf{s}_{i,k}$	(i, k) -th terrain patch (quadrilateral)	n	Sun position sample number
$s_{s,p}$	tree (p, s) location slope	N_l	number of Sun position samplings
y	height above sea level	$\mathbf{p}_N, \mathbf{p}_S,$	cardinal point vectors pointing North, South,
\bar{y}	normalized height above sea level	$\mathbf{p}_E, \mathbf{p}_W$	East, and West
$\bar{y}_{i,k}$	patch $s_{i,k}$ normalized height above sea level	\mathbf{p}_n	n -th Sun location vector from patch $s_{i,k}$
$\bar{y}_{o;s}$	optimal height above sea level for s -th species	\mathbf{p}_0	Sun starting position at East
$\Delta \bar{y}_{o;s}$	tolerated height above sea level deviation	\mathbf{p}_{N_l-1}	Sun ending position at East
τ	simulation step number (time)	$\alpha_{i,k,n}$	Sun sample incidence angle
Visualization			
l_{LOD}	geometry level of detail		
\mathbf{p}_v	viewpoint		

Appendix B. Algorithms

Here we list pseudocodes of the developed algorithms for our ecosystem simulation, domination and for tree elimination, new tree creation and moisture, wind, and sun intensities calculations.

Algorithm 1 Ecosystem simulation routine.

Require: v – list of all tree species for simulation; r – population of plants in current generation; f – list of all matrices for environmental factors

Ensure: runs step simulation of ecosystem spontaneous afforestation

```
loop
  create new trees for all species( $v, r$ );
  grow all trees( $r, f$ );
  eliminate dead trees( $r$ );
end loop
```

Algorithm 2 Creation of new trees on terrain.

Require: $N_s, \Delta N_{g,s}$ – see Appendix A for denotation; p_s – for each species, probability of nearby growth for new trees

Ensure: new trees $P_{s,n}$ added to list of trees for all species, and to terrain

```
for  $s = 0$  to  $N_s - 1$  do
   $A_s := 0$ ;
  for  $n = 0$  to  $\Delta N_{g,s} - 1$  do
    if  $p_s < \text{random}(0, 1)$  and list length( $P_{s,p}$ )-1  $\geq 0$  then
       $I_{s,n} := \text{random index}(0, \text{list length}(P_{s,p})-1)$ ; {seeding: create list of seeds from nearby parent trees}
    else
       $A_s := A_s + 1$ ; {trees to seed further away}
    end if
  end for
  sort list  $I_{s,n}$  by values of stored indices;
  for  $n = 0$  to list length( $I_{s,n}$ ) - 1 do
     $s_{s,n} := \text{get random neighbouring patch}(I_{s,n})$ ;
     $p_{s,n} := \text{random vertex on patch}(s_{s,n})$ ;
    if the vertex  $p_{s,n}$  is not in water then
       $P_{s,n} := \text{create new tree}(p_{s,n})$ ; {the seed falls on appropriate ground}
    end if
  end for
  for  $n = 0$  to  $A_s - 1$  do
     $s_{s,n} := \text{get random patch}()$ ; {we add trees, which can grow far away from parents}
     $p_{s,n} := \text{random vertex on patch}(s_{s,n})$ ;
    if the vertex  $p_{s,n}$  is not in water then
       $P_{s,n} := \text{create new tree}(p_{s,n})$ ;
    end if
  end for
end for
```

Algorithm 3 Ecological neighbourhood intersection domination determination for tree elimination.

Require: p_A, a_A, r_A – see Appendix A for denotation; (i_A, k_A) – patch indices of where tree A is located; N_{opt} – neighbouring patches search extent; $p_{i,k,n}$ – list of locations of n trees, which are close to patch $s_{i,k}$; $a_{i,k,n}$ – list of competitive abilities of n trees, which are close to patch $s_{i,k}$; $r_{i,k,n}$ – list of radii of n trees, which are close to patch $s_{i,k}$

Ensure: $D_{A,B}$ – domination flags of all trees

```
 $D_{A,B} := \text{false}$ ;
for  $z = k - N_{opt}$  to  $k + N_{opt}$  do
  for  $x = i - N_{opt}$  to  $i + N_{opt}$  do
    for  $n = 0$  to list length( $p_{i,k}$ )-1 do
      if  $|p_A - p_{i,k,n}| < r_A + r_{i,k,n}$  and  $a_A < a_{i,k,n}$  then
         $D_{A,B} := \text{true}$ ; return;
      end if
    end for
  end for
end for
```

Algorithm 4 Moisture intensity calculation on a terrain.

Require: $i_{max}, k_{max}, \mathbf{c}_{i,k}, N, m_{climate}, m_w, s_{i,k}, k_m$ – see Appendix A for denotation

Ensure: $m_{i,k}$ – matrix of elements, moisture intensity for each patch on terrain

```
for  $i = 0$  to  $i_{max} - 1$  do
  for  $k = 0$  to  $k_{max} - 1$  do
     $m_{i,k} := m_{climate}$ ;
    if  $\bar{y}_{i,k} \leq \bar{y}_w$  then
       $m_{i,k} := m_{i,k} + m_w$ ; {add moisture to patches of stagnant water accumulations}
    end if
  end for
end for
sort patches descending by height ( $\mathbf{c}_{i,k}, N$ ); {we get a list, e.g. (34, 13, 17, ...)}
for  $j = 0$  to  $N - 1$  do
  ( $i, k$ ) := get indices ( $i, k$ ) from sorted index  $j$ ; {we get  $0 \rightarrow 34, 1 \rightarrow 13, 2 \rightarrow 17, \dots$ }
   $n_{neigh;i,k} :=$  determine number of flow away sides of patch  $s_{i,k}$ ;
   $\Delta m_{i,k} := m_{i,k} \frac{s_{i,k} k_m}{n_{neigh;i,k}}$ ; {we calculate part of flow away for one patch}
   $m_{i\pm 1, k\pm 1} := m_{i\pm 1, k\pm 1} + \Delta m_{i,k}$ ; {add this part to neighbouring patches}
   $m_{i,k} := m_{i,k} - n_{neigh;i,k} \Delta m_{i,k}$ ; {subtract moisture for current patch}
end for
for  $j = 0$  to  $N_{mblur} - 1$  do
   $m_{i,k} := \frac{m_{i,k} + m_{i-1,k} + m_{i,k+1} + m_{i+1,k} + m_{i,k+1}}{5}$ ; {soften and widen moisture accumulation peaks}
end for
```

Algorithm 5 Wind intensity calculation on a terrain.

Require: $i_{max}, k_{max}, \mathbf{c}_{i,k}, N_w$ – see Appendix A for denotation

Ensure: $w_{i,k}$ – matrix of elements, wind intensity for each patch on terrain

```
for  $i = 0$  to  $i_{max} - 1$  do
  for  $k = 0$  to  $k_{max} - 1$  do
     $\varphi_{i,k} := 0$ ;
    for  $w = 0$  to  $N_w - 1$  do
       $\varphi_{i,k,w} = \arctan \frac{c_{y;i,k,w} - c_{y;i,k}}{\|\mathbf{c}_{i,k,w}' - \mathbf{c}_{i,k}\|}$ ;
       $\varphi_{i,k} := \min(\varphi_{i,k,w}, \varphi_{i,k})$ ; {smaller the wind shelter angle, the more wind is blowing}
    end for
    if  $\varphi_{i,k} \leq S_0$  then
       $w_{i,k} := \varphi_{i,k}$ ;
    else
       $w_{i,k} := 1$ ;
    end if
  end for
end for
```

Algorithm 6 Sun intensity calculation on a terrain.

Require: $i_{max}, k_{max}, N_l, \mathbf{p}_N, \mathbf{p}_S, \mathbf{p}_E, \mathbf{p}_W, l_{ambient}$ – see Appendix A for denotation

Ensure: $l_{i,k}$ – matrix of elements, sun intensity for each patch on terrain

$\mathbf{p}_0 := \mathbf{p}_E + 0, 37\mathbf{p}_N$; {determine Sun start and end visible locations}

$\mathbf{p}_{N_l-1} := \mathbf{p}_W + 0, 37\mathbf{p}_N$;

$\mathbf{p}_{\frac{N_l-1}{2}} = (0, \sin 44^\circ, \cos 44^\circ)$;

for $i = 0$ **to** $i_{max} - 1$ **do**

for $k = 0$ **to** $k_{max} - 1$ **do**

$l_{i,k} := 0$; {initialization of patch Sun light intensity}

for $n = 0$ **to** $N_l - 1$ **do**

if $n \leq N_l - 1$ **then**

$\mathbf{p}_n = \mathbf{p}_0 + \frac{2n}{N_l-1}(\mathbf{p}_{\frac{N_l-1}{2}} - \mathbf{p}_0)$; {pre-noon Sun location samples}

else

$\mathbf{p}_n = \mathbf{p}_0 + \frac{2(n - \frac{1}{2})}{N_l-1}(\mathbf{p}_{N_l-1} - \mathbf{p}_{\frac{N_l-1}{2}})$; {afternoon Sun location samples}

end if

$\cos \alpha_{i,k,n} = \bar{\mathbf{n}}_{i,k} \cdot \frac{\mathbf{p}_n}{\|\mathbf{p}_n\|}$; {sample incidence angle}

if $\cos \alpha_{i,k,n} \geq 0$ **then**

$l_{basic;i,k,n} = \cos \alpha_{i,k,n}$; {direct light radiation to a sample}

if $0 < l < \frac{N_l-1}{2}$ **then**

$l_{i,k,n} = l_{basic;i,k,n} \left(1 + \frac{2n}{N_l-1}\right)$; {light radiation over middle part of day}

else

$l_{i,k,n} = l_{basic;i,k,n} \left(3 - \frac{2n}{N_l-1}\right)$; {light radiation at morning and evening part of day}

end if

if not in shadow(\mathbf{p}_n) **then**

$l_{i,k} := l_{i,k} + \frac{l_{i,k,n}}{N_l}$; {accumulation of all Sun intensity for a patch}

else

$l_{i,k} := l_{i,k} + \frac{l_{i,k,n} l_{ambient}}{N_l}$; {add ambient light intensity in shadowed areas}

end if

end if

end for

end for

end for

References

- [1] Albus, J. S., 1999. The engineering of mind. *Information Sciences* 117 (1-2), 1–18.
- [2] Aono, M., Kunii, T., 1984. Botanical tree image generation. *IEEE Computer Graphics and Applications* 4 (5), 10–34.
- [3] Benes, B., Guerrero, J. M. S., February 2004. Clustering in virtual plant ecosystems. In: *WSCG Proceedings*.
- [4] Bloomenthal, J., 1985. Modeling the mighty maple. In: Barsky, B. A. (Ed.), *SIGGRAPH '85 Conference Proceedings*, San Francisco, CA, 22–26 July 1985. pp. 305–311.
- [5] Braun, J., Mitchell, J., 1983. Solar geometry for fixed and tracking surfaces. *Solar Energy* 31 (5), 439–444.
- [6] Chiba, N., Muraoka, K., Doi, A., Hosokawa, J., 1997. Rendering of forest scenery using 3D textures. *Journal of Visualization and Computer Animation* 8 (4), 191–199.
- [7] Cook, R. L., 1986. Stochastic sampling in computer graphics. *ACM Transactions on Graphics* 5 (1), 51–72.
- [8] Dale, M. R. T., 1999. *Spatial Pattern Analysis in Plant Ecology*. Cambridge Studies in Ecology. Cambridge University Press, Cambridge, UK.
- [9] Deussen, O., Colditz, C., Stamminger, M., Drettakis, G., 2002. Interactive visualization of complex plant ecosystems. In: *Proceedings of the conference on Visualization '02*. pp. 219–226.
- [10] Deussen, O., Hanrahan, P., Lintermann, B., Mech, R., Pharr, M., Prusinkiewicz, P., 1998. Realistic modeling and rendering of plant ecosystems. In: *Proceedings of SIGGRAPH '98*. pp. 275–286.
- [11] DigiBen, 2005. Game tutorials. Internet <http://www.gametutorials.com/gtstore/c-1-test-cat.aspx>; access 26 May 2010.
- [12] Eguchi, T., Hirasawa, K., Hu, J., Ota, N., 2006. A study of evolutionary multiagent models based on symbiosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 36 (1), 179–193.
- [13] Ellenberg, H., Weber, H., Dull, R., Wirth, V., Werner, W., Paulissen, D., 1991. *Zeigerwerte von pflanzen in mitteleuropa*. *Scripta Geobotanica* (18), 1–248.
- [14] Gribb, G., Hartmann, K., 2001. Fast extraction of viewing frustum planes from the world-view-projection matrix. Internet <http://www2.ravensoft.com/users/ggribb/planeextraction.pdf>; access 27 March 2006.
- [15] Grimm, V., Revilla, E., Berger, U., Jeltsch, F., Mooij, W., Railsback, S., Thulke, H., Weiner, J., Wiegand, T., DeAngelis, D., 2005. Pattern-oriented modeling of agent-based complex systems: lessons from ecology. *Science* 310 (5750), 987.
- [16] Guisan, A., Zimmermann, N. E., 2000. Predictive habitat distribution models in ecology. *Ecological Modelling* (135), 147–186.
- [17] Hagrais, H., Sobh, T., 2002. Intelligent learning and control of autonomous robotic agents operating in unstructured environments. *Information Sciences* 145 (1-2), 1–12.
- [18] Hill, M. O., Roy, D. B., Mountford, J. O., Bunce, R. G. H., 2000. Extending ellenberg's indicator values to a new area: algorithmic approach. *Journal of Applied Ecology* (37), 3–15.
- [19] Holton, M., 1994. Strands, gravity, and botanical tree imagery. *Comput. Graph. Forum* 13 (1), 57–67.
- [20] Hopkins, B., 1954. A new method for determining the type of distribution of plant individuals. *Annals of Botany* XVIII, 213–226.
- [21] Keppens, J., Shen, Q., 2006. Granularity and disaggregation in compositional modelling with applications to ecological systems. *Applied Intelligence* 25 (3), 269–292.
- [22] Klein, J., 2003. Breve: a 3D environment for the simulation of decentralized systems and artificial life. *Artificial life* eight, 329.

- [23] Kobler, A., Cunder, T., Pirnat, J., 2005. Modelling spontaneous afforestation in Postojna area, Slovenia. *Journal for Nature Conservation* 13 (2-3), 127–135.
- [24] Korosec, P., Silc, J., Filipic, B., 2010. The differential ant-stigmergy algorithm. *Information Sciences*.
- [25] Lane, B., Prusinkiewicz, P., may 27–29 2002. Generating spatial distributions for multilevel models of plant communities. In: *Proceedings of the Graphics Interface 2002 (GI-02)*. Canadian Information Processing Society, Mississauga, Ontario, Canada, pp. 69–80.
- [26] Lawesson, J. E., Fosaa, A. M., Olsen, E., 2003. Calibration of Ellenberg indicator values for the Faroe Islands. *Applied Vegetation Science* (6), 53–62.
- [27] Lee, M., 2003. Evolution of behaviors in autonomous robot using artificial neural network and genetic algorithm. *Information Sciences* 155 (1), 43–60.
- [28] Li, Z., Zhu, Q., Gold, C., 2005. *Digital Terrain Modeling*. CRC Press, San Francisco.
- [29] Marsaglia, G., 1972. The structure of linear congruential sequences. In: Zaremba, S. K. (Ed.), *Appl. Number Theory numer. Analysis, Proc. Sympos. Univ. Montreal 1971*, 249–285. Academic Press, New York, NY, USA, pp. 249–285.
- [30] Mech, R., Prusinkiewicz, P., 1996. Visual models of plants interacting with their environment. In: *Proceedings of SIGGRAPH '96*. pp. 397–410.
- [31] Norris, D., 1966. Solar radiation on inclined surfaces. *Solar Energy* 10 (2), 72–76.
- [32] Oppenheimer, P. E., 1986. Real time design and animation of fractal plants and trees. *Computer Graphics* 20 (4), 55–64.
- [33] Paunovski, O., Eleftherakis, G., Dimopoulos, K., Cowling, T., 2010. Evaluation of a selective distributed discovery strategy in a fully decentralized biologically inspired environment. *Information Sciences* 180 (10), 1865–1875.
- [34] Pressman, O. E., 2001. Terrain engine. Internet <http://ohad.visual-i.com/exper/exper.htm>.
- [35] Prusinkiewicz, P., Hammel, M., Hanan, J., Měch, R., 1997. L-systems: From the theory to visual models of plants. In: Michalewicz, M. T. (Ed.), *Plants to Ecosystems. Vol. 1 of Advances in Computational Life Sciences*. CSIRO Publishing, P.O. Box 1139, Collingwood 3066, Australia, Ch. 1, pp. 1–27.
- [36] Prusinkiewicz, P., Lindenmayer, A., 1990. *The Algorithmic Beauty of Plants*. Springer-Verlag.
- [37] Qu, B., Suganthan, P., 2010. Multi-objective evolutionary algorithms based on the summation of normalized objectives and diversified selection. *Information Sciences*.
- [38] Ray, T., 1990. Evolution and optimization of digital organisms. *Scientific excellence in supercomputing: The IBM*, 489–531.
- [39] Reeves, W., 1985. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *Proceedings of SIGGRAPH'85*, 313–322.
- [40] Reinhardt, D., Pesce, E.-R., Stieger, P., Mandel, T., Baltensperger, K., Bennett, M., Traas, J., Friml, J., Kuhlemeier, C., 2003. Regulation of phyllotaxis by polar auxin transport. *Nature* (426), 255–260.
- [41] Samal, A., Brandle, J., Zhang, D., 2006. Texture as the basis for individual tree identification. *Information Sciences* 176 (5), 565–576.
- [42] Schut, M., 2010. On model design for simulation of collective intelligence. *Information Sciences* 180 (1), 132–155.
- [43] Sen, Z., 2004. Solar energy in progress and future research trends. *Progress in Energy and Combustion Science* 30 (4), 367–416.
- [44] Shih, T., 2001. Mobile agent evolution computing. *Information Sciences* 137 (1-4), 53–73.

- [45] Shreiner, D., Woo, M., Neider, J., Davis, T., 2004. OpenGL programming guide. Addison-Wesley.
- [46] Stojanova, D., Panov, P., Gjorgjioski, V., Kobler, A., Dzeroski, S., 2010. Estimating vegetation height and canopy cover from remotely sensed data with machine learning. *Ecological Informatics* 5 (4), 256–266.
- [47] Strnad, D., Guid, N., 2004. Modeling trees with hypertextures. *Comput. Graph. Forum* 23 (2), 173–188.
- [48] Symeonidis, A., Valtos, E., Seroglou, S., Mitkas, P., 2005. Biotope: an integrated framework for simulating distributed multiagent computational systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans* 35 (3), 420–432.
- [49] Tichý, L., 2002. JUICE, software for vegetation classification. *Journal of Vegetation Science* (13), 451–453.
- [50] van Lieshout, M. P., 2004. Time dependent energy calculations. In: *Wind flow and Trees*. British Wind Energy Association.
- [51] von Mammen, S., Jacob, C., 2007. Genetic swarm grammar programming: Ecological breeding like a gardener. In: Srinivasan, D., Wang, L. (Eds.), 2007 IEEE Congress on Evolutionary Computation. IEEE Computational Intelligence Society, IEEE Press, Singapore, pp. 851–858.
- [52] Weber, J., Penn, J., 1995. Creation and rendering of realistic trees. *Proceedings of SIGGRAPH '95*, 119–128.
- [53] Weber, M., Neri, F., Tirronen, V., in press. A Study on Scale Factor in Distributed Differential Evolution. *Information Sciences*.
- [54] Wikipedia, 2006. OpenGL — Wikipedia, The Free Encyclopedia. Internet <http://en.wikipedia.org/w/index.php?title=OpenGL&oldid=92006684>.
- [55] Woo, M., et al., 1999. OpenGL programming guide: the official guide to learning OpenGL, version 1.2, 3rd Edition. Addison-Wesley.
- [56] Xiao, P., Vadakkepat, P., Lee, T., 2008. Context-dependent DNA coding with redundancy and introns. *IEEE Transactions on Systems Man and Cybernetics-Part B-Cybernetics* 38 (2), 331–341.
- [57] Yang, Z., Tang, K., Yao, X., 2008. Large scale evolutionary optimization using cooperative coevolution. *Information Sciences* 178 (15), 2985–2999.
- [58] Yu, E., Suganthan, P., in press. Ensemble of niching algorithms. *Information Sciences*.
- [59] Zaksek, K., Podobnikar, T., Ostir, K., 2005. Solar radiation modelling. *Computers & Geosciences* 31 (2), 233–240.
- [60] Zamuda, A., Brest, J., Bošković, B., Žumer, V., 2008. Large Scale Global Optimization Using Differential Evolution with Self Adaptation and Cooperative Co-evolution. In: 2008 IEEE World Congress on Computational Intelligence. IEEE Press, pp. 3719–3726.
- [61] Zamuda, A., Brest, J., Bošković, B., Žumer, V., 2009. Differential Evolution with Self-adaptation and Local Search for Constrained Multiobjective Optimization. In: *IEEE Congress on Evolutionary Computation 2009*. IEEE Press, pp. 195–202.
- [62] Zamuda, A., Brest, J., Guid, N., Žumer, V., 2006. Construction of virtual trees within ecosystems with ecomod tool. In: *Proceedings of IPSI-2006 Slovenia, International Conference on Advances in the Internet, Processing, Systems, and Interdisciplinary Research*. p. 15.
- [63] Zamuda, A., Brest, J., Guid, N., Žumer, V., 2007. Modelling, Simulation, and Visualization of Forest Ecosystems. In: *The IEEE Region 8 EUROCON 2007: International conference on "Computer as a tool", September 9-12, 2007, Warsaw, Poland*. IEEE Press, pp. 2600–2606.