Ecologically Sound Procedural Generation of Natural Environments

Onrust, Benny; Bidarra, Rafael; Rooseboom, Robert; van de Koppel, Johan

DOI
10.1155/2017/7057141

Publication date
2017

Document Version
Final published version

Published in
International Journal of Computer Games Technology

Citation (APA)
Onrust, B., Bidarra, R., Rooseboom, R., & van de Koppel, J. (2017). Ecologically Sound Procedural Generation of Natural Environments. *International Journal of Computer Games Technology*, *2017*(7057141), 1-17. Article 7057141. https://doi.org/10.1155/2017/7057141

Hindawi
International Journal of Computer Games Technology
Volume 2017, Article ID 7057141, 17 pages
https://doi.org/10.1155/2017/7057141

Research Article

Ecologically Sound Procedural Generation of Natural Environments

Benny Onrust,[1] Rafael Bidarra,[1] Robert Rooseboom,[2] and Johan van de Koppel[2]

[1]*Computer Graphics and Visualization Group, Delft University of Technology, Delft, Netherlands*
[2]*Department of Spatial Ecology, Royal Netherlands Institute for Sea Research, Yerseke, Netherlands*

Correspondence should be addressed to Rafael Bidarra; r.bidarra@tudelft.nl

Received 11 February 2017; Accepted 23 April 2017; Published 18 May 2017

Academic Editor: Michael J. Katchabaw

Current techniques for the creation and exploration of virtual worlds are largely unable to generate sound natural environments from ecological data and to provide interactive web-based visualizations of such detailed environments. We tackle this challenge and propose a novel framework that (i) explores the advantages of landscape maps and ecological statistical data, translating them to an ecologically sound plant distribution, and (ii) creates a visually convincing 3D representation of the natural environment suitable for its interactive visualization over the web. Our *vegetation model* improves techniques from procedural ecosystem generation and neutral landscape modeling. It is able to generate diverse ecological sound plant distributions directly from landscape maps with statistical ecological data. Our *visualization model* integrates existing level of detail and illumination techniques to achieve interactive frame rates and improve realism. We validated with ecology experts the outcome of our framework using two case studies and concluded that it provides convincing interactive visualizations of large natural environments.

1. Introduction

The visualization of existing and future natural environments is becoming more important for decision-making, as well as for recreational and scientific communication, as it considerably helps to better understand the various spatial relations in an environment [1, 2]. This is an important topic for ecologists who are focusing on developing ecological models that can predict how an environment develops in the future (see Figure 1). Such models use ecological and geophysical processes to make these accurate predictions. The disadvantage of these models is that the output lacks detail and often can only be used by ecologists. A 3D visualization of this data can be helpful to communicate their work to nonecologists or promote future/existing natural environments to the public in general.

The combination of ecological models, existing geodatasets, and 3D visualizations is becoming more relevant with the so-called "Building with Nature" solutions. Building with Nature is an initiative that focuses on the development of nature combined with other utilities [3]. For example, instead of creating a strong dike to protect the land against water, ecological processes are utilized in the target area to develop natural dunes that can provide protection. This area can be used not only for security, but also for recreation services. Figure 2 shows an example of a Building with Nature project. This project started with placing a lot of sand before the coast (Figure 2(a)), which has evolved, by the end of 2013, into a more unnatural coastline shape (Figure 2(b)) that allowed natural dune beach and dune development through stimulating natural ecological processes in that area. In addition, the dunes were enriched with the extra sand, promoting the coastal protection. Further, vegetation started growing on the sand and provided space for fish, sea mammals, and birds. Finally, there is more space for recreational purposes.

Ecological models are being developed to predict these processes and 3D visualizations could help to explore and communicate these results. This requires that the 3D visualizations are detailed, visually convincing, and easily accessible to the general public. Therefore, the output data from ecological models or geodatasets needs to be translated, in an ecologically correct manner, into an accurate plant distribution.

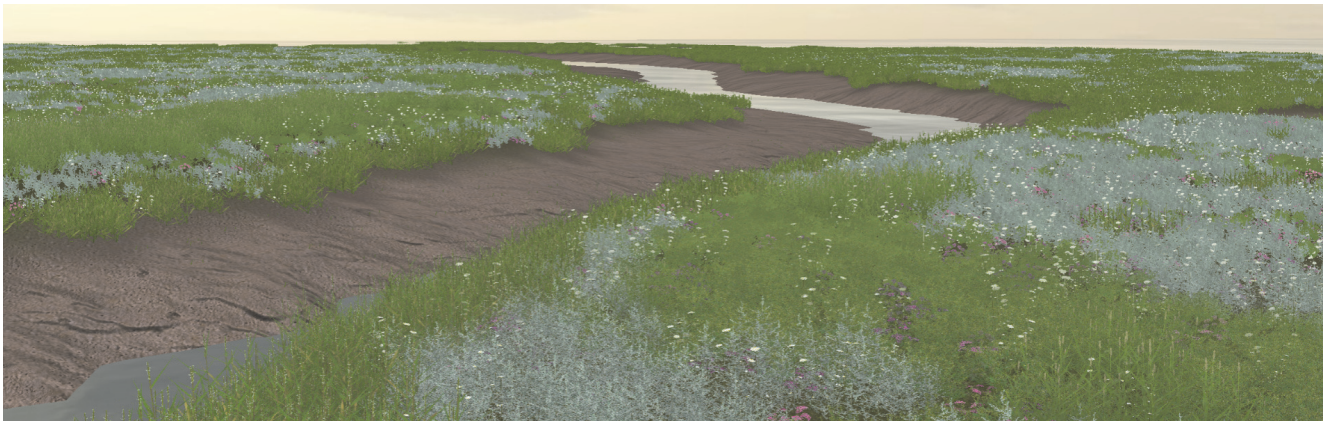


Figure 1: Virtual Paulinapolder: a salt marsh located in the Netherlands, generated, and rendered with our framework.

(a) (b)

Figure 2: An example of a "Building with Nature" project [3]. On (a), a lot of sand is placed, which is transformed by ecological processes in the area shown in (b).

In addition, to promote communication and dissemination, visualizations of such results should be easily and widely accessible, making interactive 3D web visualizations little less than indispensable.

However, both the generation of ecologically sound plant distributions and the generation of detailed 3D environments that are suitable for interactive web-based visualization are far from trivial tasks. The input data from either ecological models or geodatasets do not often contain enough detail to derive exact plant positions nor to obtain a high-density plant distribution with a large variety of species. Therefore, procedural generation techniques have to be used to generate and fill these missing details. Most procedural techniques for natural environment focus either on simulation of ecosystems or on the global generation of ecosystems using state-of-the-art point generation technique to determine plant positions. Both families of techniques lack the ability to correctly translate ecological input data, like coverage or patchiness data of plant species, to a plant distribution with high density and variety. Moreover, most examples of interactive 3D visualization of high-density natural environments focus on desktop applications, which are less useful than web-based applications in the context of ecological management, policy-making, and popular awareness. It is not possible to use these techniques directly in a web environment, because browser-based solutions do not have the same rendering capabilities as desktop-based solutions. Current web visualization approaches focus on natural environments with only the physical terrain or a low plant density/variety.

We present a new approach to generate accurate and sound plant distributions from ecological input maps and interactively visualize its results in a web browser-based context. This article, therefore, answers the following questions: (i) how to generate an ecologically sound plant distribution from ecological input maps and (ii) how to generate a visually convincing interactive 3D web-based visualization of such natural environments with high density and variety of plants. We answer these questions by proposing a framework with (i) a *vegetation model* that combines procedural and ecological modeling techniques to translate landscape maps to an ecologically sound plant distribution and (ii) a *visualization model* that translates the generated plant distribution into a 3D representation suitable for interactive visualization over the web. The vegetation model is able to translate input landscape maps with statistics about coverage and patchiness of plant species to a sound and convincing plant distribution; and the visualization model supports rendering natural environments with high density and variety of plants.

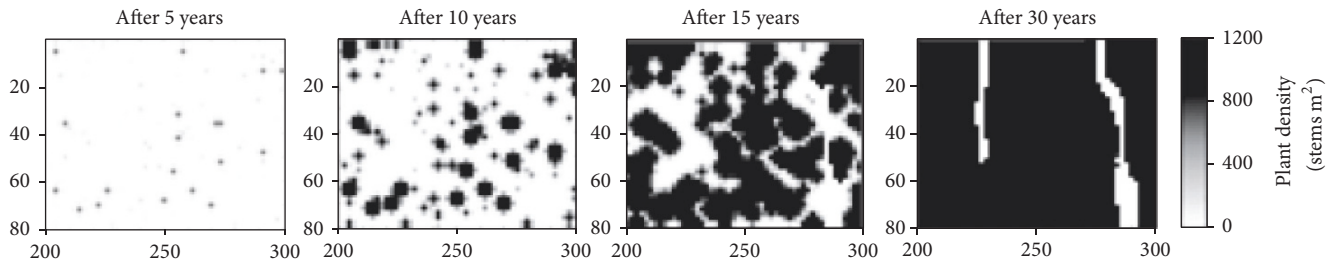


Figure 3: Example outputs of a dynamic ecological model at different time stamps [6].

This article is a significantly extended version of a previous conference paper [4].

2. Related Work

This section provides an overview of techniques related to ecological modeling, procedural ecosystem generation, and interactive 3D visualization of natural environments. An overview of ecological and procedural techniques is included to show the current limitations in generating plant distributions. We also include non-web-based solutions of interactive 3D visualization, because of the limited examples that are available for interactive web-based visualization for natural environments. We do not include a review of generative algorithms to produce individual 3D plant models, as a survey of such techniques has been recently published elsewhere [5].

2.1. Ecological Model Techniques. We divide ecological model techniques into two categories: dynamic and neutral model techniques. Dynamic models simulate ecological and geophysical processes [8], which normally result in raster maps containing information about height, biomass, and/or coverage of certain vegetation at a certain point in time [6, 9, 10]. This data often lacks sufficient details to extract plant positions. Dynamic models make it possible to extract spatial information about future landscapes. Figure 3 shows the output of a dynamic ecological model at different time stamps. This model provides information about the plant density for an area that develops over the years.

Neutral models generate classification grid maps based on coverage and shape metric information per plant species. Shape metric values give information about the patterns/patchiness of a plant species (e.g., a plant species could grow scattered in an area or grow very close to each other). This input data is translated to a single plant species for each grid cell on the input map by using either a MRC (modified random clusters) model [11] or fractal-based model [12]. The disadvantage of neutral model techniques is that, similarly to dynamic models, plant positions can often not be extracted directly from the generated maps. Another disadvantage is that neutral models assume that the conditions for each plant species are the same for the complete environment (hence the term "neutral"). For example, they assume that the coverage value for a plant species is the same at every location in the environment. Often, this assumption does not hold in real-world environments.

2.2. Procedural Ecosystem Generation Techniques. Procedural ecosystem techniques compute virtual plant distributions, and these techniques can be divided into two categories: local-to-global or global-to-local [13]. Techniques from the local-to-global category use multiset L-systems to simulate plant growth and competitions [14]. To obtain a complete ecosystem, it is necessary to iterate through the L-system rules and stop the simulation after a certain amount of iterations. Local-to-global techniques make it possible to model individual behavior for each plant. Complex behavior, such as realistic competition for sun light and soil resources, can be modeled [15]. The disadvantage is that the controllability of these techniques is low, as it is not possible to predict the outcome after the simulation is finished given the input parameters. They are not able to translate maps and statistics about the environment to a realistic plant distribution. Instead, these methods are good in showing interactions between different plants.

The global-to-local techniques do not use a simulation process to calculate a plant distribution and plants are not modeled individually. Instead, positions from plants are calculated directly from a globally defined environment. Hammes [16] uses a method that defines possible ecotypes for an environment. An ecotype is, for example, a forest or desert. Given a height map, the likelihood for each tile for every ecotype is calculated. The ecotype with the highest probability, while accounting for random variation, is selected. Next, plants belonging to that ecotype are scattered randomly in that tile. This method is limited, because plants are randomly placed within a tile and only a single type of plant is used. In addition, the final distribution does not follow the input probability values for each ecotype. Lane and Prusinkiewicz [13] place each plant with a dart-throwing algorithm in combination with probability fields, which increases the likelihood that plants are placed at their preferred location. In addition, each plant can exhibit neighborhood effects on the remaining plants by updating the probability field around it with a negative or positive effect. Again, with this method it is not possible to have the input plant species follow a certain statistical distribution. Alsweis and Deussen [17] generated plant distributions by generating points following the PDD (Poisson Disk Distribution) in combination with Wang tiling to generate all the points efficiently. This method did not investigate how to classify/assign these points to a plant species. On the other hand, the placement of plants with different sizes was convincing.

Weier et al. [18] extended the previous technique by also classifying these points to different plant species, using a combination of the previously discussed methods of Hammes [16] and Lane and Prusinkiewicz [13]. First, a complete point set was generated using the PDD with a Wang tiling technique. Each point receives probability values for each plant species. Next, each point is assigned the plant species with the highest probability, while accounting for some random variation. Finally, a group of points is selected that have a probability value with the highest standard deviation, which are most certain to retain their original plant species classification. These points are used to exhibit a neighborhood effect on their neighboring points. To include this effect in the classification, the classification process is repeated until a number of iterations have been done or when a certain amount of points does not change plant species anymore. The disadvantage of this technique is that the classification process does not translate the input statistical data to the final plant distribution. Also, it is difficult to generate different kinds of plant patterns in the plant distribution with only the neighborhood kernel.

2.3. Interactive 3D Visualization of Natural Environments. 3D rendering of natural environment with high vegetation count is a difficult problem, even with dedicated software and/or hardware, due to the high polygon count and light interaction. There are several desktop-based solutions that are able to render large amounts of plants [19, 20]. Often, these techniques focus on rendering one or two plant species with a high density in the environment, but they achieve interactive frame rates with it. Web-based rendering, which has less rendering capabilities in comparison to desktop-based rendering, on the other hand, does not have techniques proposed, by our knowledge, for the rendering of such large natural environments. Instead, the current focus is more on geovisualizations. In this section, we first discuss several techniques for the rendering of natural environments using desktop-based solutions. Next, we provide a small overview of the current work done for web-based rendering related to natural environments.

One of the first techniques to render many plants in real-time was a method proposed by Deussen et al. [21] that handles complex plant ecosystems by abstracting further away plant objects into single points and lines. Bruneton and Neyret [20] developed a technique, which is able to render a realistic forest representation in real-time with realistic lighting at all scales. They use a z-field representation to render the nearest trees individually and a shader map representation to render far-away trees. The resulting lighting was realistic and suitable for real-time purposes, especially for far-away views. Other techniques focus on the rendering of millions of grass blades in an environment. Boulanger et al. [19] propose a method to render large amounts of grass blades with dynamic lighting. A LOD (level of detail) system divides the grass blades into different representations. Geometry models are used for blades close to the viewer, and blades at moderate distances are represented with vertical and horizontal slices, while far away only the horizontal slice is used. A modification of the alpha blending technique is used to blend the transitions between the LODs. Fan et al. [22] extended the previous method with animations. Although none of these solutions is web browser-based, they provide insight into how to organize the data to maintain real-time performance and to create transition between the different LODs.

The interactive 3D rendering of natural environments on a web browser is a fairly new topic that has not received much attention so far. In current literature we could not find examples of 3D interactive visualization of complex natural environments with high-density vegetation, and only a few techniques aimed at real-time visualization of environments without vegetation using geovisualizations [23, 24]. These visualizations focus on the streaming of geodata to the browser and the organization of the data to achieve interactive frame rates. Data is often organized in groups using quadtree structures to reduce far-away geometry.

3. Basic Approach

Here, we provide the outline of our approach for the generation and web-based visualization of natural environments. The main goal is to generate ecologically sound plant distributions based on various ecological datasets and to interactively visualize these over the web. In the previous section, we have discussed various methods that focus on the procedural generation and/or 3D visualization of natural environments, but often these methods are limited and do not provide satisfactory results. In particular, procedural methods for the generation of plant distributions are mostly unable to correctly process ecological information, such as statistical data on coverage and patchiness, in combination with landscape maps.

Our approach, improving upon fractal-based neutral modeling and procedural point generation techniques, capitalizes on their advantages while avoiding their pitfalls, in order to solve this problem. In addition, most interactive 3D visualizations of natural environments are currently provided in standalone applications, which typically can utilize more GPU features than web browser-based applications. We found no examples of web-based interactive visualizations of large natural environments presenting a large variety of plant species, like those we present here.

Before we go into the details of the framework's structure, we will first define some concepts frequently used in this article and elaborate on the various kinds of input.

3.1. Concepts. The following concepts will be regularly used throughout this article:

(i) Plant species: the species of the plant, for example, oak or birch.

(ii) Plant spacing: the minimal required distance between plants. Often, this is related to the plant radius or size of the specific plant species.

(iii) Plant level: different plant species that are placed in one group, because they have approximately the same

plant spacing. These groups are used in our framework to process multiple plant species simultaneously. The aim of creating this division is to allow the generation of plant distributions that contain plant species with large difference in plant spacing, such as trees and flowers.

(iv) Plant patterns or patchiness: the patterns of the plants of a certain plant species. Plants of a species that exhibit high patchiness grow close together, while plants of a species with low patchiness grow scattered throughout the environment.

(v) Plant coverage: the amount of occupation of a certain plant species in the environment.

3.2. Input. A variety of input data is used at different stages. The following list summarizes all these inputs:

(i) Landscape maps: for calculating the plant distribution and 3D visualization of the environment. During plant distribution generation, landscape maps are used in combination with statistical data of each plant species. In addition, a height map is used to represent the terrain. Landscape maps can be derived from remote sensing sources, or they are generated by ecological models.

(ii) Plant statistical data: statistics about the coverage per plant species and about the patchiness of each plant species, used for calculating the plant distribution. This statistical data is often related to one or more landscape maps. For example, we can have a height map with coverage statistics that are based on the height of that map, so that certain plant species can have higher coverage on high ground and lower coverage on low ground.

(iii) Plant models: one or more 3D models per plant species, used to represent the various plant species at the highest LOD.

(iv) Texture maps: used to represent the various LOD representations and to decorate the rest of the scene.

(v) Other parameters: for example, plant spacing for each plant species, used during plant distribution generation.

3.3. Overview. A global overview of our approach is depicted in Figure 4, visually representing the data pipeline, from input data, through *vegetation model*, to *visualization model*. The input data from existing remote sensing sources or ecological models is translated by the *vegetation model* to a point distribution where each individual point has been classified to correspond to one of the plant types occurring in that environment. The *visualization model* translates that result to an interactive 3D visualization on the web.

3.4. Vegetation Model. The vegetation model consists of plant distributions generated from landscape maps in combination with statistical data about coverage and patchiness of plant species. This is achieved by dividing the model into two separate components: plant position generation and plant species generation. The plant position component generates all possible plant positions from the input landscape maps. The plant species component generates plant species for these points using the landscape maps and the coverage and patchiness statistical data.

3.5. Visualization Model. The visualization model organizes and translates the generated plant distribution to a 3D representation suited for interactive visualization over the web. This stage consists of three phases: the offline phase, the precomputation phase, and the rendering phase. The offline phase occurs before the actual visualization and is done in advance, once and for all; it includes, for example, the plant model generation. The precomputation phase structures the plant distribution from the vegetation model into a LOD scheme of the terrain, organized in a quadtree structure. Finally, the rendering phase renders all the geometry and the various LODs are blended together to obtain smooth transitions.

4. Vegetation Model

This section describes the vegetation model consisting of plant distributions generated from landscape maps in combination with statistical data about coverage and patchiness of plant species. Generation of this model is divided into two main stages: plant position generation and plant species generation. Each of these stages will be discussed separately, and in this discussion, we will assume that the plant sizes of each plant species are equal. Towards the end of the section, we introduce the concept of plant levels, which explains how a plant distribution can be generated where plant species have significant differences in plant sizes.

4.1. Plant Position Generation. The goal of this stage is to generate all possible plant locations in the environment without assigning or creating bias towards any of the plant species. The classification of these positions using the coverage and patchiness statistics of each plant species is handled at the next stage. To obtain all possible plant positions, we adopt the PDD with Wang tiling technique used by Alsweis and Deussen [17] and Weier et al. [18]. This technique makes it possible to randomly generate points with a uniform distribution where each point has a predefined minimal distance to each other: similar to what can be observed in nature. We extended this technique to integrate plant positions of different sizes seamlessly without creating a bias based on the size to any of the plant species. The next paragraphs explain how these plant positions are generated.

4.1.1. Identify Vegetated Tiles. The first step is to identify on an input map of the landscape all the tiles that contain vegetation. This requires the use of a map that provides information on the location of vegetation, for example, Normalized Difference Vegetation Index (NDVI), biomass, or coverage maps. The next step is to threshold the map given

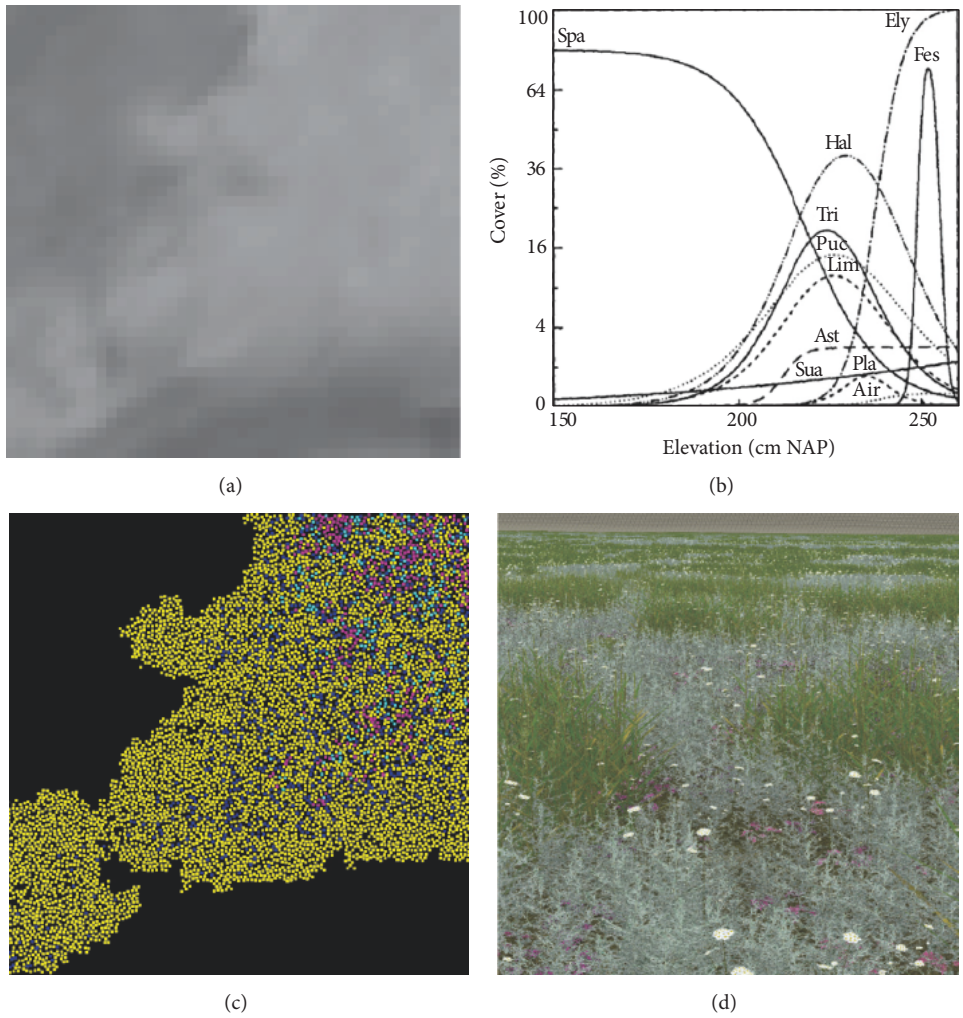


Figure 4: A visual overview of our approach starting with the various input data, such as (a) landscape maps and (b) statistical data [7]. Next, from this data we derive the plant distribution in the vegetation model (c), where each point is associated with plant species. Finally, from this plant distribution we derive the detailed 3D visualization model, suitable for rendering in the browser (d).

a user-defined threshold. Each tile with a value higher than the threshold is marked as vegetated. The resulting output is a binary grid map where, for each tile, it is indicated whether it contains vegetation or not. In Figure 5 an example of this step is shown. An NDVI map given as input is shown on the left, and on the right we show the resulting binary map after comparing each value of the tiles in the grid with the predefined threshold.

Figure 5: NDVI map on which a threshold is used to obtain the tiles that contain vegetation. Threshold is set at 0.08.

4.1.2. Generate Plant Positions from Vegetated Tiles. Points are generated with the PDD and Wang corner tiling technique [25]. Wang corner tiling is used to avoid the corner problem that appears in the regular Wang border tiling technique. A Wang tiling is created using only the tiles on the map that are marked as vegetated. Next, each Wang tile is filled with a PDD [26].

The result of this process is a seamless point distribution where each point has at least a user-defined minimum distance to other points and where only the vegetated tiles on the map contain points. The minimum distance is determined based on the plant size of the plant species. Figure 6 shows example output of plant positions generated from a tile-based map with information about vegetation presence.

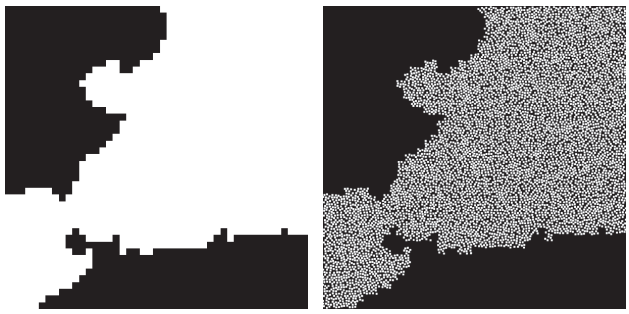


Figure 6: Plant positions generated from the vegetated tile map using the PDD with Wang tiling technique.

4.2. Plant Species Generation. The aim of the plant species generation stage is to classify the generated point distribution. The classification is based on fractal neutral modeling techniques [12]. These techniques are able to classify raster maps using coverage and patchiness statistics for each plant species. As mentioned in Section 2, they are only able to translate *static* coverage and patchiness data correctly. We extended this method by integrating it with the generated point distribution, so that it is able to handle nonstatic statistical coverage and patchiness data. The next paragraphs explain this classification procedure step by step.

4.2.1. Assigning Coverage and Patchiness Data. The first step is to assign each point a single coverage and patchiness value for each plant species in the environment. Each point extracts the appropriate value of each input map; for example, if the input is a height map, each point is assigned a height value based on the location in the map. The extracted values are translated to a coverage value by using the corresponding statistical data, for example, statistical data that contains information about the coverage of each plant species for a certain range of height values.

It is possible that each point receives multiple coverage values for the same plant species; for example, a height map may be augmented with a soil map with related coverage statistics. This means that each point receives for every plant species a coverage value based on the height and a coverage value based on the soil. For the remainder of the classification, these coverage values have to be merged to a single value, so that each point has only a single coverage value for each plant species. We obtain a single coverage value by taking the minimum value, because we assume that the minimum is the limiting growth factor for that plant species. The same process is applied to extract the patchiness values. Patchiness is represented with two values: roughness and patch area, for the size of the patterns.

4.2.2. Fractal Generation. The second step is to calculate a fractal value for every plant species in each point. Fractal values are commonly used to represent different kinds of patterns in nature [12]. The advantage of fractal algorithms is that they calculate a random value for a point that depends on the point location. This makes it possible to generate similar random values for points that are close to each other and dissimilar values for points that are not. This way, we can represent plants that grow close to each other and plants that are scattered throughout the environment. To achieve this based on the patchiness input data, our fractal algorithm must be able to translate the input roughness and patch area values to an individual fractal value for each point for every plant species. In addition, it is possible that the roughness or patch area values are nonstatic values for every plant species, in contrast to those used in neutral modeling techniques.

We use a modified fractal Brownian motion algorithm [26] that is able to generate a fractal value based on the input patchiness data. Normally, fractal values are generated by adding multiple values of Simplex noise with different weights. A base frequency value is defined to determine the clustering of similar Simplex noise values, where a lower value means higher clustering and a higher value a lower clustering. Based on the frequency, the amplitude value is used to generate a new frequency value that in turn is used to calculate a new Simplex noise value that is to be added to the previous calculated values.

The frequency and amplitude value are used to relate our patchiness data: the roughness and patch area. The patch area is related to the frequency, and the roughness is related to the amplitude. The relation between the amplitude and the roughness is basically one-to-one, because when a high roughness value results in a high amplitude value, the patterns become rougher. The reason for this is that a higher amplitude increases the frequency value with a higher value of each iteration, and a higher frequency value means more disperse patterns, which means rougher patterns. The relation between the frequency and the patch area is more difficult and is not one-to-one. Instead, the final fractal value is calculated by generating and adding multiple fractal values with different input frequency value. These input frequency values cover the whole range of patch area values that are available for that plant species in that environment. The final value is calculated based on a weighted average of all these calculated fractal values. The weight of each fractal value depends on the similarity of the input patch area used for that point. This process is required to support nonstatic patchiness data within each plant species. Additional details of this algorithm with examples can be found elsewhere [26]. Figure 7 shows fractal values that are generated for each point position and all plant species. In this case there are four plant species, which means that each point position receives four fractal values equal to the number of plant species. In addition, the example demonstrates the influence of different patchiness statistics on the patterns of each plant species.

4.2.3. Classification. The last step is to classify each point to a plant species using the coverage and fractal values that were assigned to each point in the previous steps. First, an individual threshold value for each plant species is calculated for every point. The threshold value of a point is found by taking an ordered list of the fractal values of all the points of that particular plant species and then using the coverage value of each point as percentile in that list. The fractal value that matches with the position of the particular percentile is the threshold value that is going to be used for that point.

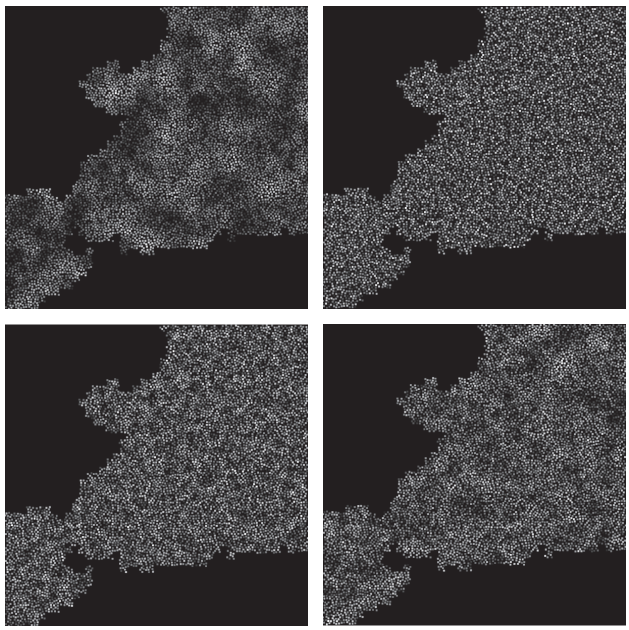


Figure 7: Fractal map for each of four plant species where each point has received fractal values for each plant species based on their patchiness data. Clearly, different kinds of patterns can be identified among the plant species.

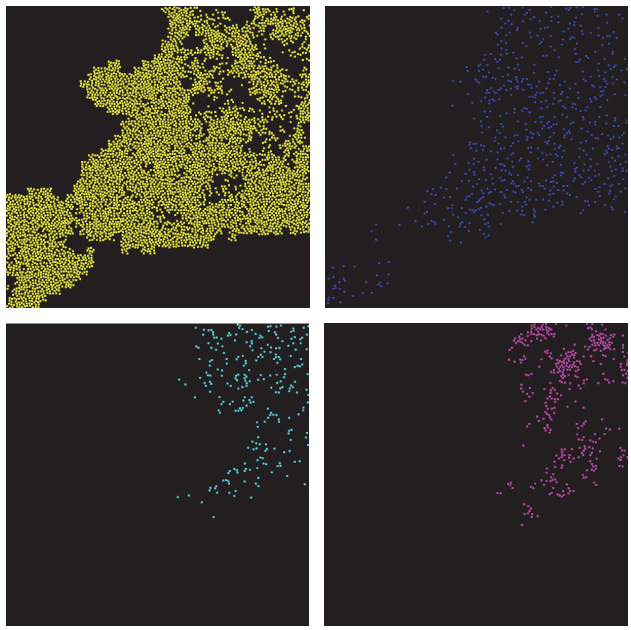


Figure 8: The intermediate result in the classification process where each plant species has been assigned to the available plant positions separately to meet the coverage input statistics.

Now each point has, for every plant species, a separate threshold that is based on the coverage values. Next, for each plant species, the fractal and threshold value of each point are compared. When the fractal value is higher than the threshold value, the point is assigned the corresponding plant species. The result of this step is that each plant species gets assigned a set of points matching the coverage and patchiness input statistics. Figure 8 shows the various points that are classified for each plant species with different coverage statistics.

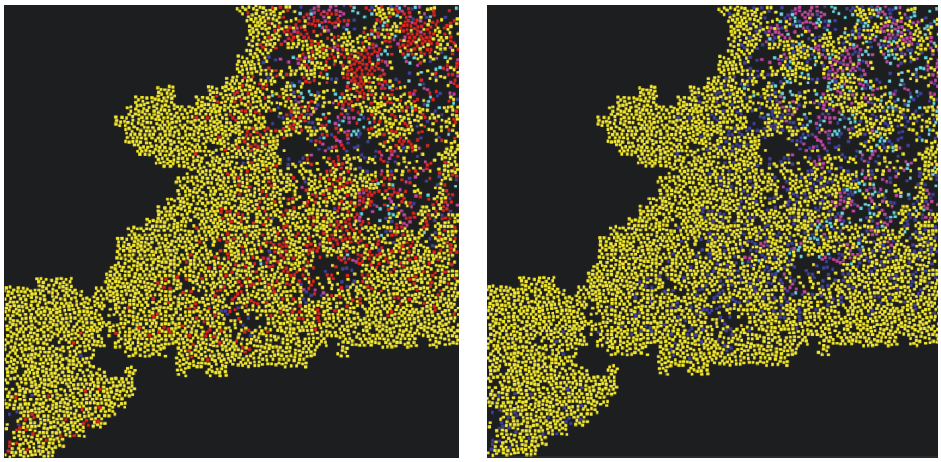
In this process, it may happen that certain points have been assigned to multiple plant species. These conflicts are solved by assigning the plant species that has the highest fractal value, which is determined separately for each conflicted point. Figure 9 shows the classified plant distribution with conflicts and the distribution where the conflicts are solved.

The consequence of this can be that a certain plant species may end up having less coverage than required. Therefore, the remaining nonclassified points are used to add additional coverage to such plant species. Before the remaining points can be classified, it is first necessary to update the coverage values so that each plant species will meet its expected coverage in the final plant distribution. For each unclassified point, a new coverage value is calculated by repeating the first step of the classification component. First, the used coverage statistics are updated for each input map by generating several reference points that are uniformly distributed over the complete range of values of the input map. Next, for each reference point, the total amount of coverage in the intermediate plant distribution is calculated. This is compared to the expected coverage and, by subtracting the current coverage, we get the amount of missing coverage per reference point. Per reference point, all coverage values are normalized. Next, coverage values can be assigned as usual, as in the first step of this stage.

The remaining points are assigned a plant species by repeating the same classification process. The only difference is that the plant species are processed one-by-one on the new remaining point set, so no conflicts are generated. The main reason for this step is to ensure a stopping point for the algorithm; otherwise, conflicts are likely to be generated, and the process may need to be repeated. The plant species with the highest standard deviation in their average patchiness statistics in comparison with the other plant species is processed first. By the end of this process, a complete plant distribution is obtained as shown in Figure 10. The plant distribution is generated following the input statistics about coverage and patchiness as can be seen in the plant distribution.

4.3. Multiple Plant Levels. In the previous sections, we assumed that all plant species have approximately the same plant size. In this section, we describe how our vegetation model can also support plant species that have large differences in plant size, such as trees and flowers, and their interaction. To achieve this, we introduce the concept of plant levels, which basically divides the available plant species in the environment in different groups. The division is based on the plant size, so plant species with approximately the same size are grouped together. The usage of multiple plant levels requires a few extensions in both the plant position generation and plant species generation part, which we describe in this section.

4.3.1. Plant Position Generation. Plant positions for multiple plant levels are generated semi-separately from each other.



(a) (b)

Figure 9: In (a), points in red are the plant positions that have been assigned to multiple plant species. On (b), these conflicts have been resolved by taking the plant species with the highest fractal for a conflicted point.

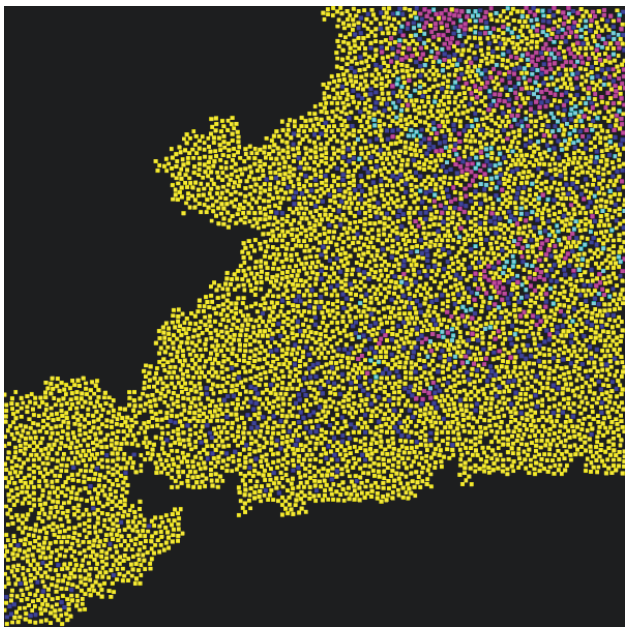


Figure 10: The final plant distribution of the vegetation model.

This means that we start by generating plant positions from the largest plant level (the plant species that have the largest plant size) down to the smallest plant level. A plant level that is processed takes into account the points that are already placed on the map. Since each plant level is (significantly) smaller than the previously processed plant level (and thus the minimal spacing is smaller too), it is guaranteed that the plant level being currently processed can generate plant positions. The only problem is now how to take into account the points that are already generated by the previous plant levels. There are two options: use for these points the minimal distance that they had when they were placed, or use for these points the same minimal distance as for the points generated in the current plant level. We use the last option, because we do not know yet if a point generated for a certain plant species will also be classified to one of the plant species of that plant level. It is possible that during classification a point is not assigned a plant species of that plant level. When that happens, we do not want to waste this point but use it for the processing of plant species of lower plant levels.

With this choice, integration will be seamless, while with the other option the point would be isolated, because it would have a much larger distance to the other points than required. Therefore, the points generated with this algorithm are nonbiased, because the size does not influence the classification process, since it can be changed dynamically without creating artifacts, like isolated points. An example of this is shown in Figure 11, where a point distribution is generated with two plant levels. Red is the larger plant level and blue the smaller. As can be seen, the red points all have a larger distance to each other than the blue points. Nevertheless, the blue points have the same distance to the red points as they have to each other.

4.3.2. Plant Species Generation. Again, the plant levels are processed sequentially, starting with the largest plant level. Each plant level uses the plant positions that are generated for its level as well as the plant positions that have not been classified by the previously processed plant levels. The same classification procedure as explained in the previous section is applied. After the classification of a plant level, it is possible to apply neighborhood influences, like in the work of Lane and Prusinkiewicz [13] and Weier et al. [18]. These neighborhood effects influence the coverage statistics of the neighboring nonclassified points and make it possible to model influences of, for example, trees on the neighboring smaller plants. An output example with neighboring effects is shown in Figure 12 where the largest plant level (consisting of the blue points) has a negative effect on the smaller plants of the other plant level.

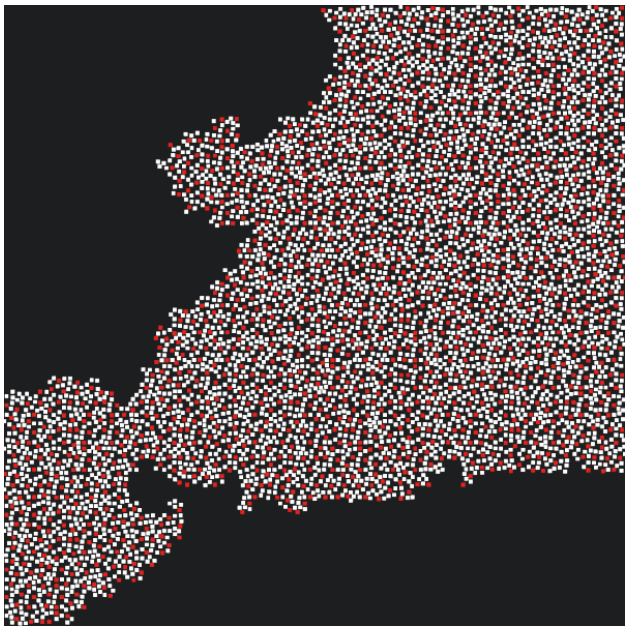


Figure 11: PDD distribution with multiple plant levels. The red points are plants with a larger plant spacing; the white points are plants with a smaller plant spacing.

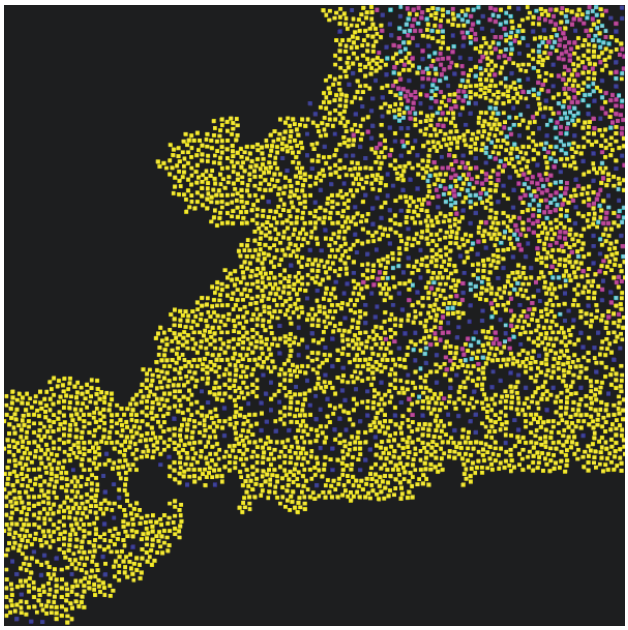


Figure 12: In this example, the blue plant species belongs to the largest plant level, while the other plant species belongs to the smallest plant level. The blue plant species has a negative neighboring effect on the other plant species.

5. Visualization Model

This section explains how the generated plant distribution is organized and translated to a 3D representation that supports visualization over the web at interactive frame rates. We start with explaining the two most important concepts of the model: data organization and transitions between the different LODs. Finally, we give an overview of the complete rendering framework.

5.1. Data Organization. For the purpose of visualization, the input plant distribution has to be translated into a 3D representation. Due to the high density of the distribution and the size of the complete environment, it is not feasible to represent every plant as a detailed 3D model, because that would result in a high geometry complexity, drastically dropping the performance of the visualization, neither would such a detailed representation be necessary, as the amount of details humans see decreases with increasing distance. Therefore, to reduce the geometry, it is necessary to use different LODs (levels of detail) for the plants. This means that a different representation for a plant, other than its 3D model, has to be used, depending on the plant location relative to the viewer.

For our framework, we adopt a level of detail scheme that divides the plant distribution into three zones depending on the location of the viewer. This scheme is similar to a LOD technique proposed for the rendering of millions of grass blades [19]. The first zone, closest to the viewer, consists of complete 3D models, to better convey the impression of a richly detailed environment. In the second zone, further away, plants are represented as billboards, that is, by flat images. To support very large scenes, we also included a third zone, further towards the horizon, where plants are not represented individually, but as a color map applied on the terrain. The switching between zones is dependent on the distance to the viewing point and can be configured with a user-defined threshold.

To use this LOD scheme, we had to solve another problem: it is not feasible to calculate the appropriate LOD representation for each plant, as this would require every frame to iterate over many hundreds of thousands of plants on the CPU. Therefore, neighboring plants are grouped together and a single check is made for the whole group. These groups are generated by dividing the plant distribution and storing it in a quadtree structure [20, 21]. The whole distribution is divided into four equal quads and each of these quads is again divided into four quads. This continues up to a number of iterations defined in the framework. The smallest quads are placed closest to the user and gradually large blocks are used to fill the remaining space. The switch between quads of different sizes depends on a distance threshold.

In each of these quads, the plants have the same LOD representation. Therefore, it is important that quads of different sizes are used and that the smaller quads are placed close to the user, while the larger quads are placed further away, to reduce the geometry complexity. We do not want to place large quads close to the viewer, because close to the user quads are filled with detailed plant models. As a result, a lot of geometry is placed outside the viewing frustum (thus outside the screen), because a large quad close to the user cannot normally fit within the complete screen. Thus, with this quadtree organization, less geometry is processed that is located outside the view of the user. We use the same quadtree structure and organization for the terrain data.

5.2. Transition between LODs. Using different representations for the plant models at fixed distances from the viewer leads to a popping effect, noticed when plant representations switch abruptly between consecutive LODs. This is an unwanted artifact that can distract the viewer from the visualization. Therefore, it is necessary to smooth the transition between the different LODs. In our case, we have two transitions: between plant models and billboards and between billboards and the terrain color map. We use alpha blending for the smoothing procedure for each transition.

The first step to produce a smooth transition between the plant models and billboards is to create a small, configurable, overlapping region where both representations for a plant coexist on the same location. Next, for each position in this transition zone an alpha value is calculated that indicates how much of that representation contributes to the final blended result. This is calculated for both plant models and billboards. As a result, the plant models in the transition border have an alpha value near to one when closest to the viewer, but gradually this value becomes smaller as the plant models get closer to the other border of the transition zone. For the billboards' alpha values, the inverse happens. The calculated alphas of both representations are now used to blend the representations. This is achieved by generating two separate images with one only containing the plant models and the other the billboards. During the generation of these images the alpha values of both representations are mapped to the alpha band of these images. Thus, every pixel of both images has received an alpha value. Finally, by combining both images, a new image is generated, for which the color of each pixel is a combination of the corresponding colors of the billboard image and plant model image.

The process for the transition between billboards and the terrain is similar to that for the transition between plant models and billboards. Again, an overlapping zone is created and for both billboards and terrain, an alpha value is calculated. However, during this transition we do not generate two separate images and combine them. Instead, the calculated alpha values for the billboards are used to represent the transparency of the billboards. This means that billboards gradually fade out, because they become transparent. The alpha value of the terrain is used to blend the terrain LOD color with the original terrain color. The result is that billboards gradually fade out as the distance to the viewer increases and the terrain gradually changes color to that similar to the billboards.

5.3. Rendering Framework. In this subsection, we give an overview of the complete rendering framework that includes the offline, precomputation, and rendering phases.

5.3.1. Offline Phase. The main task in the offline phase is the generation of the 3D plant models, which we perform using L-systems. The main challenge of using L-systems is that they are often hard to master, due to their lack of controllability [5], making it time-consuming to create L-system rules that generate convincing plant models. Therefore, we developed a node-based L-system method, which allows one to create L-system rules by means of a sequence of nodes in a graph. Each of these nodes represents an L-system operation. One of the main advantages of using node-based systems is that the user can follow the content generation flow between the various L-system rules and other relevant data [27]. For this, we used the procedural engine Sceelix [28], which implements the concept of Procedural Content Graphs [29], benefiting from its numerous features: for example, the parameters for each node operation can be dynamically set and can be made dependent on one another throughout the L-system.

5.3.2. Precomputation Phase. The first task in the precomputation phase is to load all the data necessary for the visualization, such as 3D plant models, textures for billboards, a height map to generate the terrain, and the plant locations derived from the plant distribution. These plant locations, organized in a quadtree structure, are used to place both plant models and billboards at correct positions. In order to minimize the "clone effect" of many similar plant models and billboards used in the visualization, each position also contains some additional information about the rotation, scale, and color variance of the plant, aimed at slightly randomizing its appearance. The rotation and scale factor are a random uniform number; the color factor also depends on the scale factor, meaning in practice that the smaller the plant, the darker its color.

The terrain mesh is calculated from the imported height map where each vertex corresponds to the height value of a tile from the height raster map. The triangulation of these vertices is based on using an additional terrain height map that has double the resolution of the original height map, which is obtained by using bicubic interpolation. We do this interpolation outside of the framework and just import it along with the original height map. This additional map is necessary, because we need to decide which triangles are to be generated for each quad. Since we derive the terrain from a height map, the obtained vertices are always laying in strict rows and columns that form quad, each of which can be halved into two possible triangle pairs. The decision on which triangle pair to generate depends on the additional reference height point in the middle obtained from the height map with the double resolution: for each triangle pair, the height of the middle point is calculated based on its two triangles; the triangle pair with the smallest height difference to that reference point is chosen.

Another important task during the precomputation phase for the terrain mesh is to calculate the colors of the color map LOD representation for the terrain. As explained earlier in this section, the color map LOD representation is not represented as separate instances like in the case of the plant models and billboards but is a color on the terrain based on the color of the represented plant. This is calculated by computing for each vertex in the terrain mesh the number of plants and plant species that are within a certain distance from it. The dominant plant species, which has the largest number of plants closest to the specified vertex, is then chosen for that vertex. Finally, the color that is assigned to the vertex is obtained from a separate texture map, which defines the terrain color map for each plant species.

5.3.3. Rendering Phase. In the rendering phase, the actual visualization of the plants, terrain, water, and background is performed. For each frame a check is made to decide which LOD representation should be rendered at a certain plant position using the quadtree structure. The first task in the rendering phase is to go through the quadtree of the plant positions and terrain, determine which quads are visible, and choose which LOD representation they should have. The next task is to render the visible objects and selected representation to the screen. This task is divided into three steps, aimed at achieving a smooth transition between the plant models and billboards. The first two steps pertain the rendering of the two separate images using alpha blending, and the third step performs their combination. The first image is rendered with the plant models and the underlying terrain and water. The second image is rendered with the billboards and the underlying terrain, water, and also the background.

The visualization is enriched with shadows to obtain a more convincing scenery. Shadows are computed for the plant model objects and terrain by using the percentage-closer soft shadow mapping technique [30]. We did not compute any shadows for the billboards, because that would drastically increase the complexity of the visualization. Rather, shadows on the billboards are integrated into the texture of the specific plant species. This requires no additional computation during the rendering phase. Furthermore, in order to introduce additional details on the mesh, both the terrain and water are enriched with normal maps, and they use the environment map of the skybox to create some reflection as well.

6. Implementation

The implementation of our framework involves several modules. The vegetation model is implemented with Python scripts and its output is stored in a text file that is used as input for the visualization model. The visualization model was implemented with WebGL, because it is a cross-platform free web standard that gives access to the low-level 3D graphics API based on OpenGL ES 2.0. In addition, it does not require installing any plug-ins, because it is implemented right into the browser and all major browsers support WebGL. For the actual implementation, we used an existing WebGL framework *three.js* [31]. In the remainder of this section, we will focus on the implementation details for the WebGL rendering.

6.1. Plant Models. Each plant species is represented by one or more 3D plant models. The geometry of these models is stored on the GPU by using VBOs (Vertex Buffer Objects). We only store a single instance of each unique model to reduce the memory footprint. This also means that we do not use a large variety of models for each plant species, because that would result in a large number of models that have to be stored in the GPU. Since we only store one unique instance of each model, we need to store additional data on the GPU to be able to place the different models across the scene. Additional VBOs linked to each plant model are created that contain information about position, scale, rotation, and color. During rendering, each model goes through its linked lists and uses this information to place itself in the environment, a process called geometry instancing. The creation of the various buffers is handled by the three.js framework, but the geometry instancing process was not yet available at the time of implementation. Therefore, this was implemented in the three.js framework by using the WebGL extension *ANGLE_instanced_arrays* with its corresponding functions. (At the time of writing, the instancing process has also become available in the official three.js framework build.)

6.2. Billboards. Efficient and effective methods for generating billboards use geometry shaders so that only a single vertex has to be sent to the GPU, which is then transformed in the geometry shader to various planes [20]. However, currently, geometry shaders are not available in WebGL. There are two alternatives: (i) to create the various planes to represent the billboards beforehand, which means that additional geometry has to be sent to the GPU and processed in the vertex shader, and (ii) to send a single vertex to the GPU and vertex shader and use the *GL_Point* command in combination with the *Gl_PointSize* command available in WebGL. This means that the vertex is directly written as a pixel on the screen based on the provided position with a certain size (e.g., number of pixels) defined with *Gl_PointSize*. The advantage of this method is that the amount of geometry sent to the GPU is still as low as possible and the billboards always face the camera directly, since they are written directly on the screen as a quad. The disadvantage is that some controllability is lost regarding the shape of the billboards because, with this method, billboards are always represented as perfect squares on which a texture is placed. In certain cases, it is possible that the original texture of the billboard is not a perfect square, and the texture has to add additional transparent pixels to the original texture to become a perfect square. This means that potentially a lot of unused transparent pixels are processed in the fragment shader.

We decided to use the second alternative, because we wanted to limit as much as possible the amount of geometry that is sent to the GPU, to boost the frame rate of the visualization. One important step of this implementation of the billboards was to define the size (in pixels) of each billboard on the screen, so that billboards that are further away from the viewer must have a size that is smaller than the billboards nearby.

Finally, each point representing a billboard has a list attached with information about color variation, plant species, texture variation, and position. The plant species and texture variation information is necessary to select the correct billboard texture from the complete texture map. Each plant species has multiple billboard textures based on their plant models obtained from different viewing angles, which are defined using the texture variation information. One of the textures is chosen for each position and during visualization the texture of the billboards at a certain position does not change. The main reason for this is that the use of a single texture at each position is much more efficient than switching between various textures during rendering.

6.3. Terrain and Water. The terrain geometry is put in VBOs and stored on the GPU. The same applies to the water geometry, which is basically represented as one big plane. The shaders that are used to render both the water and terrain use various common techniques such as texture blending, normal, and environmental mapping. Textures are blended based on, for example, the height of terrain to create smooth transitions between the different types of terrain (e.g., sand and grass). Normal maps are used to introduce additional details on the terrain. Environmental mapping is mainly used for water rendering to create reflection on the water.

6.4. LOD Transitions. The implementation of the transitions between the LODs was achieved by using FBOs (Frame Buffer Objects). FBOs make it possible to write and store results instead of rendering directly to the screen. A separate FBO is used to render the plant models and their surroundings and another FBO is used to render the billboards and surroundings. The textures from both FBOs are blended together on the GPU by using a shader, as described in the previous section. This result can then be sent to the screen, or it can be stored in another FBO to apply any subsequent effects.

A smooth transition can only be achieved when regions of the billboards and plant models overlap. During the blending of the two images the shader does not know whether it is blending plants, terrain, or water. Therefore, a small overlap must also be created of the terrain and water. This results in the terrain and water being partly rendered twice.

6.5. Shadows. We only compute shadows that are cast by the plant models on the plant models themselves and on the terrain. Shadows are computed using the percentage-closer soft shadow mapping technique, which is supported by the three.js framework. These shadows are computed by first generating a depth texture of the scene that is stored in an FBO. The depth texture is generated based on the same geometry instancing technique described above.

Finally, shadows are simply approximated for the terrain LOD on which the billboards are placed. Each vertex in the terrain shader has information about the number of plants that are in the neighborhood, and in the fragment shader this number is used to decide on the darkness of the color for that fragment, to represent shadows.

7. Results and Discussion

In this section, we present some results generated by our framework and discuss its rendering performance. Finally, we discuss the validation of these results and of the framework as a whole.

7.1. Input. To test our framework, we generated results for two different regions: (i) an existing area called the Paulinapolder (247500 m^2), a salt marsh located in the South of the Netherlands, and (ii) a fictive area (2025 m^2) based on the output of an ecological model describing a (future) salt marsh. For both areas, we use various landscape maps and statistical data about coverage and patchiness of plants as input to generate plant distributions with the vegetation model. Since the Paulinapolder is an existing area, we can use landscape maps derived from existing geographical datasets. We have used two types of landscape maps: a height map and an NDVI map. In addition, we used coverage and patchiness statistics that are based on the height of the environment. The NDVI map is used to determine the presence of vegetation in the environment. In total, we consider seven different plant species as input.

The ecological model area is based on an ecological model developed by Schwarz [10]. This model generates landscape maps containing information about the height of the environment and a coverage map for a single plant species. The generation of a single map by the ecological model does not necessarily mean that only one plant species grows there. After discussion with the ecologists, we decided to add two additional plant species as input. To be able to process these additional plant species, we use the same coverage and patchiness statistics based on height as used for the Paulinapolder. The coverage map is not directly used as a coverage statistic for any plant species. Instead, it is used to determine where vegetation is located.

7.2. Results. Based on the provided input data, we generated the plant distribution of the vegetation model for each area and translated this plant distribution to the corresponding visualization model. The results of the vegetation model are shown in Figure 13 for the Paulinapolder and in Figure 14 for the ecological model area. Figures 15 and 16 present a global visualization of both environments, respectively. Figure 17 captures the seamless transition between the different LODs, and Figure 18 provides a close-up view of plant models. More results, including a video and the interactive web visualization itself, are available online (https://graphics.tudelft.nl/benny-onrust).

7.3. Performance. We focus on the performance of the visualization model, as one of our main aims was to achieve interactive frame rates on a web browser. The vegetation model is computed offline before the actual rendering and therefore does not influence interactivity nor rendering performance.

The plant distribution generated for the Paulinapolder area has around 700.000 plants. A typical frame of this environment consists of up to 2,7 million triangles, representing the plant models, terrain, water, and background. The rendering times for a typical Paulinapolder scene were 7.3 ms, of which over 60% was spent on the plant models and their shadows and around 30% on the billboards. For the ecological model area, the plant distribution generated consists of around 150.000 plants, and a typical frame consists of around 0,4 million triangles in total. The rendering times for a typical scene of the ecological model were 5.5 ms, of which around 40% was spent on the plant models and their shadows and over 40% on the billboards.

We measured these rendering times on an Alienware Aurora R4 with an Intel Core i7-4820K CPU @3.70 GHz, 16 GB RAM, and NVIDIA GeForce GTX 780, using the

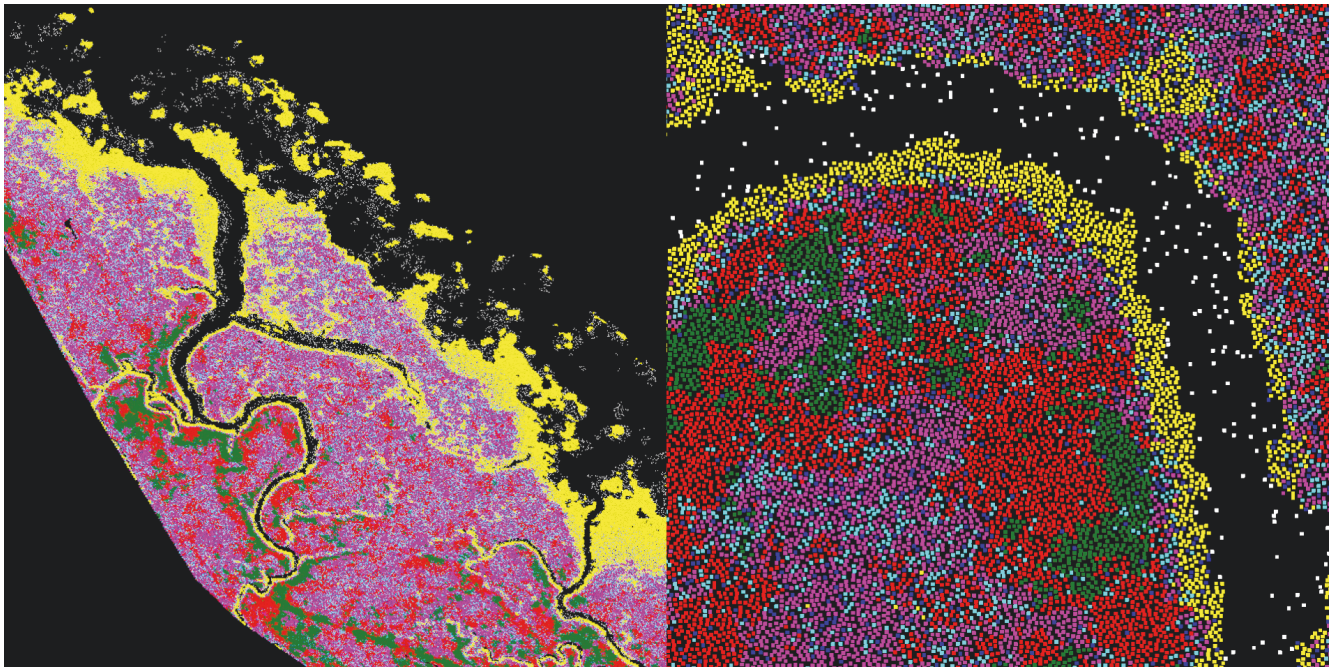


Figure 13: Result of the vegetation model for the Paulinapolder where yellow is Spartina, green is Elymus, red is Atriplex, blue is Aster, teal is Artemisia, pink is Limonium, and white is Salicornia.

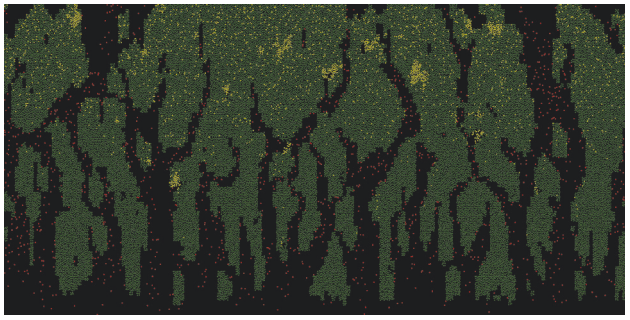


Figure 14: Result of the vegetation model for the ecological model area where green is Spartina, red is Salicornia, and yellow is Aster.

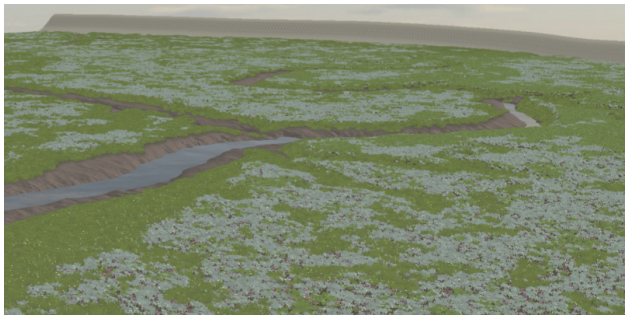


Figure 15: Global overview of the virtual Paulinapolder.

Chrome browser v43.0. Machines with other GPUs offered results in the same order of magnitude, thus always providing frame rates above 50 fps.

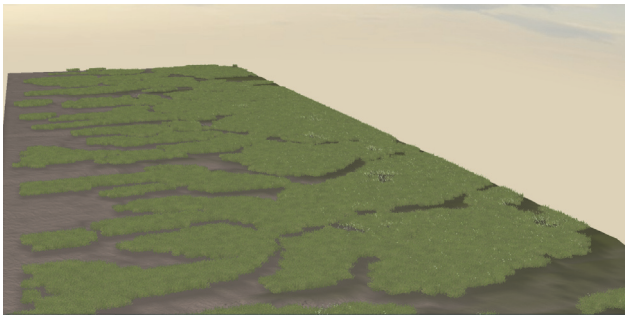


Figure 16: Global overview of the virtual ecological model area.

7.4. Validation. We combine the validation of the Paulinapolder and the ecological model area, because the comments on both areas are generally applicable. We performed two types of validation: (i) a statistical validation of the plant distribution, which calculates whether the input statistical data matches the statistics derived from the generated plant distribution, and (ii) an expert validation, which was performed in collaboration with ecologists during the development of this framework.

For the statistical validation on the generated plant distribution, we created several artificial datasets to investigate certain special cases in the input data. First, we compared the input coverage statistics with the coverage statistics of the generated plant distributions. We did this on a global level, where we calculate a single average coverage value for each plant species and where we compare it with the average coverage value of each plant species in the generated plant

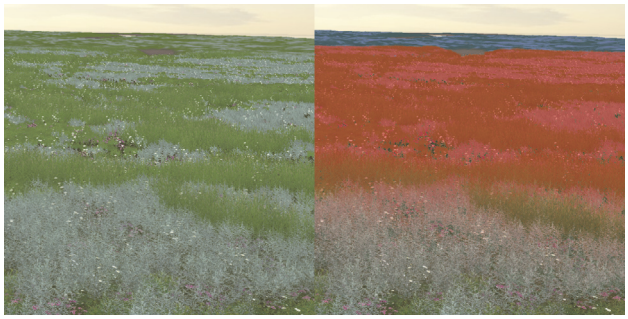


Figure 17: Transition of the various LODs in the virtual Paulinapolder. Green is the regular plant models, red is the billboards, and blue is the terrain color map.

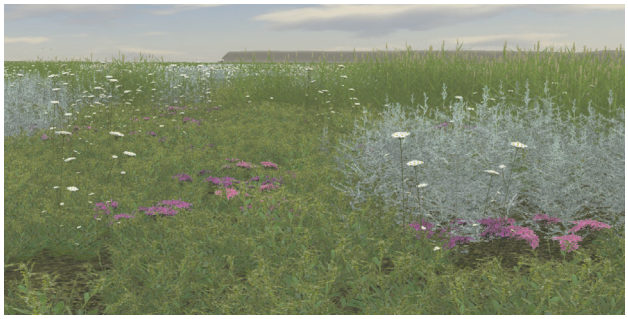


Figure 18: Close-up view of the plant models in the virtual Paulinapolder.

distribution. In addition, we performed the same comparison on a local level where we, for example, calculate the average coverage at certain height points for plant species whose coverage is dependent on height. The exact figures of the various comparisons can be found elsewhere [26]. This validation showed that the input coverage data matches the coverage values of the generated plant distribution both on a global and on a local level.

We also validated our results throughout the development of the framework, by having ecologists visually judge the plant distributions and visualization models obtained. This was done to investigate whether the generated results were convincing and if the data was behaving properly. In general, the ecologists found that the vegetation model was able to convincingly translate ecological data to a plant distribution for both the Paulinapolder and ecological model area. In addition, plausible patterns in the plant distributions were clearly reproduced in the visualization model: they were deemed convincing and different patterns could be clearly identified across the various plant species. Figure 15 shows several such different patterns, ranging from very large patches of plants that grow closely together, to small random patterns of plants that grow scattered throughout the area.

The visualizations themselves were deemed convincing by ecologists, who judged them as proper representations of salt marshes. This was especially the case for the local/middle distance view where the plant models are visible. The far view, that is, the region where plants are represented as billboards or as a color map on the terrain, was considered less convincing than the local/middle view. One of the remarks was that small gaps started to appear in the plant distribution at a certain distance range. Another remark was that the color variation among the various plant species was not entirely satisfactory. However, in this view, the various patterns for different plant species are clearly visible. The transitions between the different LODs were, in most cases, not noticeable by the ecologists, or they were at least not considered distracting.

7.5. Discussion. The aim of this research was to generate an ecologically sound plant distribution from landscape maps and ecological statistical data and to translate it to a convincing interactive 3D visualization over the web. Also, the plant distribution generation solution should be generic, in the sense that it should support different plant species, patterns, and input data. Validation showed that these requirements were well met in general. Using statistical and expert validation, we showed that input ecological maps and statistics were translated to a plant distribution with convincing patterns. Expert validation and performance measurements also indicate that we were able to create convincing real-time 3D visualizations. Based on the validation and implementation, we also found several limitations in the present framework, which we discuss now.

Validation of the vegetation model showed that the coverage statistics were translated correctly to the generated plant distribution and that the resulting patterns were convincing. However, we did not validate whether the input patchiness statistics also match the output statistics, because the regular methods to calculate patchiness statistics assume that the patterns are in a grid format and not in a point format. Therefore, it should still be investigated how to perform this kind of measurements on point sets. In any case, expert validation indicates that the patterns were visually convincing.

The main limitation of the visualization model lies in the representation of the billboards. Billboards are represented as a single point and they are rendered as several pixels on the screen to maximize rendering performance. As a result, billboards face the camera from every viewing angle. When the billboards are viewed globally from a high, bird-eye view, the generated billboards often look less convincing, because the same texture that is normally used to view the billboards horizontally is then being used to view the billboards vertically. In the current visualization, this is often not very noticeable, because all the plants are relatively small and have relatively uniform color distribution, but when the plants become larger and there are more differences among the plants, this will be more noticeable. In addition, it is difficult to automatically set the correct size for each billboard, because size is measured in number of pixels. This gave the problem that certain billboard objects have the correct size in local view, but from certain distances in global view, the billboards become too small and this creates small gaps in the distribution. An example of this is shown in Figure 19.

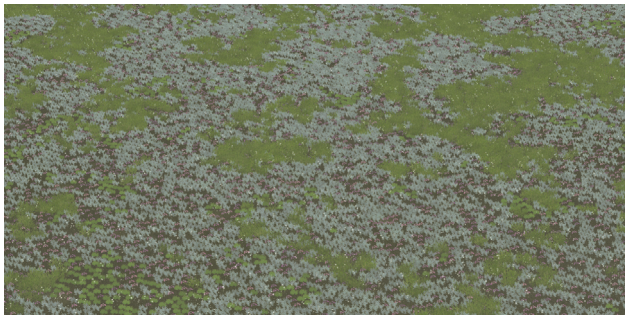


Figure 19: Gaps appear in the plant distribution when viewing the visualization in global view.

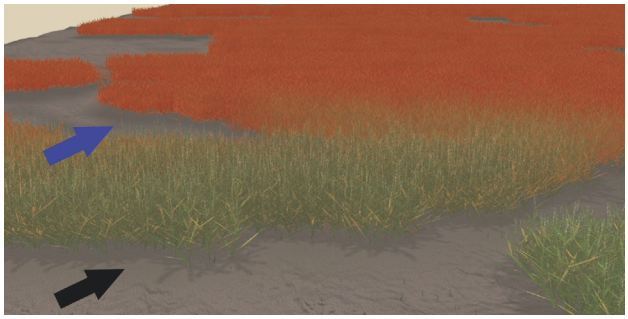


Figure 20: The plant models (green) cast different shadows than the billboards (red). The shape of the plant can be seen in the shadows cast by the plant models (see the shadows pointed by the black arrow), which is not visible in the shadows (see the blue arrow) that have been approximated for the billboards.

Shadows of the billboards are approximated by using baked-in shadows in the texture, and shadows on the terrain are approximated by turning the terrain color slightly darker. This shadow computation does not use the actual shape of the plants. In Figure 20, we can see that the billboards objects have different "shadows" on the ground than the plant models close to the viewer, though it is often difficult to notice these differences and change in shadows. The shadows cast by billboards could be improved by (partly) replacing the current billboards with volumetric billboards [32]. These are able to generate realistic shadows and realistic different viewing angles. However, their efficient implementation requires, for example, the use of a geometry shader, which is not yet available in WebGL. Additional research into this topic could greatly enhance the realism of 3D plant distributions, because it would allow for the generation of more convincing billboards, which are the weakest point in the current visualization model with respect to a convincing appearance.

Another disadvantage is the shadow computation technique for the plant model objects in the visualization. Currently, we use the percentage-closer shadow mapping technique that was directly available in *three.js*. The computed shadows are realistic enough for our purposes, but performance-wise it could be improved by, for example, using a variance shadow mapping technique [33].

Finally, the alpha blending transition between the plant models and billboards is in most cases smooth and there are very limited ghosting or popping effects. When there is a large difference in size between the plants in the visualization, the transition is less smooth for the larger plants. The reason for this is that the transition thresholds are the same for all plant model objects. This could be improved by varying the threshold per plant species. The larger plant species could switch farther to a billboard representation.

8. Conclusion

We developed a new method for the generation of accurate plant distributions from landscape maps and statistical data and for the visualization of the resulting natural environments in an interactive 3D web environment. We presented an implemented framework that addresses the main challenges of creating such plant distribution and of generating and rendering a 3D visualization model that can be browsed at interactive frame rates. For the plant distribution generation, we presented a new model that combines existing procedural plant placement techniques using Poisson Disk Distribution with Wang tiling technique in combination with concepts from neutral modeling techniques. In addition, a visually convincing interactive 3D web visualization was created by using, among others, LOD, shadow mapping, and geometry instancing techniques. We tested our system by generating plant distributions for two case studies, using landscape maps and ecological statistical data. Ecologists validated our results and found them to be most convincing. Statistics showed that our framework is able to translate correctly the input coverage statistics to the output plant distribution.

Our work stands out from previous research, because (i) our plant distribution generation is fully data-driven and (ii) we demonstrated with our interactive visualization WebGL prototype the possibilities of rendering over the web very large natural environments with a high density and variety of plants.

In the future, we would like to investigate whether other representations of billboards improve the visualization at different viewing angles, especially in global view. In addition, we would like to investigate more local and global illumination models to improve performance and realism of lights and shadows in the visualization. To improve the usability of this method, it might be preferable to combine the vegetation and visualization model in one web application, so that the user can easily change the plant distribution in the 3D visualization without having to do offline computations. Furthermore, it might be interesting to extend the framework by including the fauna of the environment, for improved realism [34]. Finally, so far our framework has been tested on environments with only grass-like plant species; we would like to do additional testing for other more forest-like scenes to assess the quality and performance of its results.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to thank Alex Kolpa for helping with the implementation of the L-system plugin in Sceelix.

References

[1] C. Pettit, C. Raymond, B. A. Bryan, and H. Lewis, "Identifying strengths and weaknesses of landscape visualisation for effective communication of future alternatives," *Landscape and Urban Planning*, vol. 100, no. 3, pp. 231–241, 2011.

[2] R. van Lammeren, J. Houtkamp, S. Colijn, M. Hilferink, and A. Bouwman, "Affective appraisal of 3D land use visualization," *Computers, Environment and Urban Systems*, vol. 34, no. 6, pp. 465–475, 2010.

[3] H. J. de Vriend, M. van Koningsveld, S. G. J. Aarninkhof, M. B. de Vries, and M. J. Baptist, "Sustainable hydraulic engineering through building with nature," *Journal of Hydro-Environment Research*, vol. 9, no. 2, pp. 159–171, 2015.

[4] B. Onrust, R. Bidarra, R. Rooseboom, and J. Van De Koppel, "Procedural generation and interactive web visualization of natural environments," in *Proceedings of the 20th International Conference on 3D Web Technology*, pp. 133–141, ACM, 2015.

[5] R. M. Smelik, T. Tutenel, R. Bidarra, and B. Benes, "A survey on procedural modelling for virtual worlds," *Computer Graphics Forum*, vol. 33, no. 6, pp. 31–50, 2014.

[6] S. Temmerman, T. J. Bouma, J. Van de Koppel, D. Van der Wal, M. B. De Vries, and P. M. J. Herman, "Vegetation causes channel erosion in a tidal landscape," *Geology*, vol. 35, no. 7, pp. 631–634, 2007.

[7] J. de Leeuw, L. P. Apon, P. M. Herman, W. de Munck, and W. G. Beeftink, *The Response of Salt Marsh Vegetation to Tidal Reduction Caused by the Oosterschelde Storm-Surge Barrier*, Springer, 1994.

[8] J. Molofsky and J. D. Bever, "A new kind of ecology?" *BioScience*, vol. 54, no. 5, pp. 440–446, 2004.

[9] M. Rietkerk and J. van de Koppel, "Regular pattern formation in real ecosystems," *Trends in Ecology & Evolution*, vol. 23, no. 3, pp. 169–175, 2008.

[10] C. Schwarz, *Implications of biogeomorphic feedbacks on tidal landscape development [Ph.D. thesis]*, Radboud University Nijmegen, 2014.

[11] S. Saura and J. Martinez-Millan, "Landscape patterns simulation with a modified random clusters method," *Landscape Ecology*, vol. 15, no. 7, pp. 661–678, 2000.

[12] W. W. Hargrove, F. M. Hoffman, and P. M. Schwartz, "A fractal landscape realizer for generating synthetic maps," *Conservation Ecology*, vol. 6, no. 1, 2, 2002.

[13] B. Lane and P. Prusinkiewicz, "Generating spatial distributions for multilevel models of plant communities," in *Proceedings of the Graphic Interface*, pp. 69–80, 2002.

[14] O. Deussen, P. Hanrahan, B. Lintermann, R. Mech, M. Pharr, and P. Prusinkiewicz, "Realistic modeling and rendering of plant ecosystems," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 275–286, ACM, 1998.

[15] E. Chng, "An artificial life-based vegetation modelling approach for biodiversity research," *Green Technologies: Concepts, Methodologies, Tools and Applications*, 417, 2010.

[16] J. Hammes, "Modeling of ecosystems as a data source for real-time terrain rendering," in *In Digital Earth Moving*, pp. 98–111, Springer, 2001.

[17] M. Alsweis and O. Deussen, "Wang-tiles for the simulation and visualization of plant competition," in *Advances in Computer Graphics*, vol. 4035 of *Lecture Notes in Computer Science*, pp. 1–11, Springer, Berlin, Heidelberg, 2006.

[18] M. Weier, A. Hinkenjann, G. Demme, and P. Slusallek, "Generating and rendering large scale tiled plant populations," *Journal of Virtual Reality and Broadcasting*, vol. 10, no. 1, 2013.

[19] K. Boulanger, S. Pattanaik, and K. Bouatouch, "Rendering grass terrains in real-time with dynamic lighting," in *Proceedings of ACM SIGGRAPH 2006: Sketches (SIGGRAPH '06)*, August 2006.

[20] E. Bruneton and F. Neyret, "Real-time realistic rendering and lighting of forests," *Computer Graphics Forum*, vol. 31, no. 2, pp. 373–382, 2012.

[21] O. Deussen, C. Colditz, M. Stamminger, and G. Drettakis, "Interactive visualization of complex plant ecosystems," *IEEE*, pp. 219–226, 2002.

[22] Z. Fan, H. Li, K. Hillesland, and B. Sheng, "Simulation and rendering for millions of grass blades," in *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games*, pp. 55–60, ACM, 2015.

[23] B. Fanini, L. Calori, D. Ferdani, and S. Pescarin, "Interactive 3D landscapes on line," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 3816, pp. 453–459.

[24] M. Englert, P. Herzig, S. Wagner, Y. Jung, and U. Bockholt, "X3D-earthbrowser: visualize our earth in your web browser," in *Proceedings of the 18th ACM International Conference on 3D Web Technology*, pp. 139–142, ACM, 2013.

[25] A. Lagae, *Tile-Based Methods in Computer Graphics [Ph.D. thesis]*, Katholieke Universiteit Leuven, 2007.

[26] B. Onrust, *Automatic generation of plant distributions for existing and future natural environments using spatial data [M.S. thesis]*, Delft University of Technology, //graphics.tudelft.nl/benny-onrust/, 2015, https://graphics.tudelft.nl/benny-onrust/.

[27] P. Silva, P. Müller, R. Bidarra, and A. Coelho, "Node-based shape grammar representation and editing," in *Proceedings of the Workshop on Procedural Content Generation for Games (PCG '13), Co-Located with the Eigth International Conference on the Foundations of Digital Games*, 2013.

[28] Sceelix, "The 3D scenes procedural engine," http://www.sceelix.com. Accessed: 1 April 2017.

[29] P. B. Silva, E. Eisemann, R. Bidarra, and A. Coelho, "Procedural content graphs for urban modeling," *International Journal of Computer Games Technology*, vol. 2015, Article ID 808904, 15 pages, 2015.

[30] R. Fernando, "Percentage-closer soft shadows," in *Proceedings of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '05)*, p. 35, ACM, 2005.

[31] R. Cabello, three.js-javascript 3D library, 2010.

[32] P. Decaudin and F. Neyret, "Volumetric billboards," *Computer Graphics Forum*, vol. 28, no. 8, pp. 2079–2089, 2009.

[33] W. Donnelly and A. Lauritzen, "Variance shadow maps," in *Proceedings of the Symposium on Interactive 3D Graphics and Games*, pp. 161–165, 2006.

[34] N. Komodakis, C. Panagiotakis, and G. Tziritas, "3D visual reconstruction of large scale natural sites and their fauna," *Signal Processing: Image Communication*, vol. 20, no. 9-10, pp. 869–890, 2005.