Procedural generation and interactive web visualization of natural environments

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

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

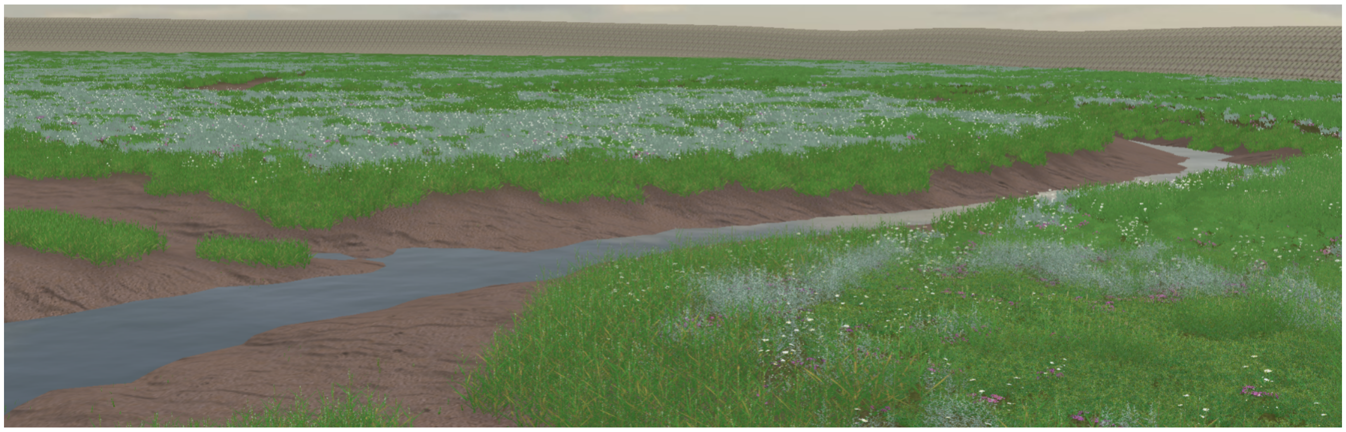

Figure 1: *A visualization of a salt marsh located in the Netherlands, rendered with our framework.*

Abstract

Interactive 3D visualization of natural environments can help ecologists, policy makers and the broad public in general to better understand, promote and protect both existing and developing environments. The creation and exploration of virtual worlds can be very helpful for this purpose. However, current techniques are neither able to generate sound natural environments from ecological data nor do they provide web-based visualizations at interactive rates of such detailed ecological systems. In this paper, we approach the challenge of developing and interactively visualizing in real time ecologically accurate and visually convincing models of complex natural environments over the web. For this, we propose a framework that (i) is able to combine landscape maps and ecological statistical data, translating them to an ecologically sound plant distribution, and (ii) creates a detailed 3D representation of the natural environment and provides for its fully interactive visualization in real-time over the web. The main contribution of our research consists of the real-time web-based visualization of complete and visually convincing natural environments with their high density and variability of individual organisms. The *vegetation model* combines and improves techniques from procedural ecosystem generation and neutral landscape modeling. It is able to generate diverse ecological sound plant distribution directly from landscape maps with statistics about coverage and patchiness of plant species. The *visualization* model uses several existing level-of-detail and illumination techniques to achieve interactive frame rates and improve realism. From the validation of the results with ecology experts we conclude that our framework provides very convincing interactive visualizations of large virtual natural environments.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality I.6.3 [Simulation and Modeling]: Applications

Keywords: WebGL, procedural generation, ecological models

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 [Pettit et al. 2011; van Lammeren et al. 2010]. This is an important topic for ecologists who are focusing on developing ecological models that can predict how an environment develops in the future. 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 non-ecologists or promote future/existing natural environments to the public in general. The combination of ecological models, existing geo-datasets 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 [de Vriend et al. 2014]. For example, instead of creating a complete dike to protect the land against water, ecological process are promoted in that area to develop natural dunes and it's corresponding vegetation for water protection. Not only can this area be used for protection, but also for recreation 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 very detailed, visually convincing and easily accessible to the general public. Therefore, the output data from ecological models or geo-datasets needs to be translated, in an ecologically correct manner, into an accurate plant distribution. 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 from given input data 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 geo-datasets do not often contain enough detail to directly extract the exact plant positions to obtain a high density plant distribution with a lot of variety in species. Therefore, procedural generation techniques are used to generate and fill these missing details. Most procedural techniques for natural environment focus either on simulation of ecosystems [Deussen et al. 1998; Chng 2010] or on the global generation of ecosystems using state-of-the-art point generation technique to determine plant positions [Alsweis and Deussen 2006; Weier et al. 2013]. 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 techniques or examples of interactive 3D visualization of high density natural environments focus on desktop and video applications [Bruneton and Neyret 2012; Boulanger et al. 2006]. It is not possible to use these techniques in a web environment, because browser-based solutions do not have the same rendering capabilities as desktop-based solutions. Current web-based visualization examples mostly focus on generation of natural environment with only terrain or a low plant density [Fanini et al. 2011; Herzig et al. 2013].

Therefore, the main questions answered in this paper are (i) how to generate an ecologically convincing plant distribution from ecological input maps and (ii) how to generate a visually convincing interactive 3D web-based visualization for natural environments with high-density and variety in plants. We answer these questions by proposing a framework with (i) a *vegetation model* that combines procedural and ecological model 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 web-based visualization at interactive frame rates. The vegetation model is able to translate input landscape maps with statistics about coverage and patchiness of plant species to a sound plant distribution; and the visualization model supports rendering natural environment with high-density and variety of plants.

The paper is organized as follow: First, we give an overview of related work regarding ecological modeling, procedural ecosystem generation, and the interactive 3D visualization of natural landscapes. Next, we present the complete overview of our framework. The following two sections discuss respectively the vegetation and visualization model of our framework. In Section 6, we present our WebGL-based implementation of the framework. We present and discuss the results in Section 7. Finally, we draw conclusions and present possible future work.

2 Related Work

This section provides an overview of techniques related to ecological modelling, 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. As to interactive 3D visualization, we also include non-web-based solutions, 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 recent overview of such techniques can be found elsewhere [Smelik et al. 2014].

Ecological model techniques We divide ecological model techniques into two categories: dynamic and neutral model techniques. Dynamic models simulate ecological and geophysical processes [Molofsky and Bever 2004], which result normally in raster maps containing information about height, biomass, and/or coverage of certain vegetation at a certain point in time [Temmerman et al. 2007; Rietkerk and Van de Koppel 2008; Schwarz 2014]. This data often lack details to extract plant positions. Dynamic models give the possibility to extract spatial information about future landscapes. Figure 2 shows the output of a dynamic ecological model with output at different time stamps. 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 type of plant 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 [Saura and Martínez-Millán 2000] or fractal-based model [Hargrove et al. 2002]. 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 name 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. Our vegetation model, will use dynamic models to obtain ecological and spatial information of future landscapes, such as height and coverage data. Neutral model techniques are then combined with procedural techniques to translate the information from the dynamic model to an ecologically correct plant distribution.

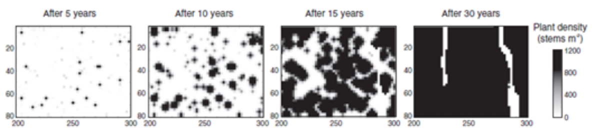

Figure 2: *Example outputs of a dynamic ecological model at different time stamps [Temmerman et al. 2007].*

Procedural ecosystem generation techniques Procedural ecosystem techniques can compute virtual plant distributions and be divided into two categories: local-to-global or global-to-local. Techniques from the local-to-global category use multi-set L-systems to simulate plant growth and competitions [Deussen et al. 1998; Lane and Prusinkiewicz 2002]. 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 give the possibility to model individual behaviour for each plant. Complex behaviour can be modelled such as realistic competition for sun light and soil resources [Chng 2010]. The disadvantage is that the controllability of these techniques is low, because it is not possible to predict the outcome after the simulation is finished given the input parameters. They are not able to to translate maps and statistics about the environment to a correct 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 modelled individually. Instead, positions from plants are calculated directly from a global-defined environment. Hammes [Hammes 2001] 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 is selected with highest probability in combination with random variation. 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 was used. In addition, the final distribution does not follow the input probability values for each ecotype. Lane [Lane and Prusinkiewicz 2002] places each plant with a dart-throwing algorithm in combination with a probability fields to place plants more at their preferred location. In addition, each plant can exhibit neighborhood effects on the remaining plants by updating the probability field around 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 [Alsweis and Deussen 2006] generated plant distributions by generating points following the PDD (Poisson Disk distribution) in combination with Wang tiling to efficiently generate all the points. This method, however, did not investigate how to classify/assign these points to a plant species. The placement of different plants with different sizes was however convincing.

Weier [Weier et al. 2013] extended the previous technique by also classifying these points to different plant species, using a combination of the previously discussed methods of Hammes [Hammes 2001] and Lane [Lane and Prusinkiewicz 2002]. First, a complete point set was generated following the PDD with a Wang tiling technique. Each point receives probability values for each plant species. Next, each point gets assigned the plant species with the highest probability in combination with some random variation. Finally, a group of points is selected that have a probability value with the highest standard deviation, and are, thus, most certain to stay by the same plant species. 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 has been done or when a certain amount of points do not change plant species anymore. The disadvantage of this technique is that the classification process does not translate in the input statistical distribution to the final plant distribution. Also, it is difficult with only the neighborhood kernel to generate different kinds of plant patterns in the plant distribution. In our vegetation model, we will partly adopt the PDD generation and classification process, but we combine it with a neutral modelling technique to be able to correctly translate statistical data about coverage and patchiness of plat species to the plant distribution.

Interactive 3D visualization of natural environments The 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. Deussen [Deussen et al. 2002] proposed a method for rendering of complete complex plant ecosystems by abstracting further away objects into points and lines. Bruneton [Bruneton and Neyret 2012] 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. Wang [Wang et al. 2013] proposes a method that greatly reduces the memory footprint of large detailed forests.

Other techniques focus on the rendering of millions of grass blades. Boulanger [Boulanger et al. 2006] proposes a method to render large amount of grass blades with dynamic lighting. A LOD (level of detail) system divides the grass blades in different representations. Close to the viewer geometry models are used, from moderate distance, blades 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. Allthough, these are non-web based solutions, it provides insight in how to organize the data to maintain real-time performance and to create transition between the different LODs. Fan [Fan et al. 2015] extended this method with animations.

The interactive web-based 3D rendering of natural environment is a fairly new topic and did not get much attention so far. We did not find, in the current literature, any examples of 3D interactive visualization of natural environment with high vegetation count. Current techniques focus on real time generation of complete environments without vegetation focusing on geo-visualizations [Fanini et al. 2011; Herzig et al. 2013]. These visualizations focus on the streaming of geo-data 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 Our Framework

Here, we provide the outline of our framework to provide 3D web-based visualization of natural environments. It is often not possible to automatically extract plant positions and their species from maps of existing and future landscapes, because these maps lack details and are too general. In section 2, we discussed models that try to determine plant positions from these maps in combination with statistical data of the expected plant distribution. This statistical data can contain information about the coverage or grow patterns of the plants. These discussed methods are limited, mainly because they are not able to determine when all the plant positions are given to correctly translate the statistical data to the final distribution. Thus, the input statistics do not match with the statistics of the output distribution. Also, it is difficult to consistently generate different kind of patterns for each plant species or different kinds of patterns for a certain plant species depending on the location in the environment. Our framework solves this first problem by combining existing procedural techniques with neutral modeling techniques. The second problem is how to organize and translate this data for an interactive 3D visualization on the web. There are current techniques that provide solution for the real-time visualization of large natural environment in non-web environments. Often, they only focus on the visualization of single plant species such as grass or trees. Our framework shows how to organize and translate the plant distribution to a 3D representation and web-based visualization.

This sections presents an overview of our framework. First, we introduce some terms frequently-used throughout the paper. Next, we talk about the requirements of our framework and third we discuss the input of our framework. Finally, we give a global overview of the process of this framework. In the subsequent sections, each part of the framework is discussed in more detail. Section 4 discusses the model to generate an ecologically sound plant distribution from landscape maps. Section 5 discusses how the generated plant distribution must be organized and translated for a 3D representation to achieve real-time performance for a web-based environment.

Terms In our framework we regularly use the following terms:

- Plant species: The species of the plant. For example, an oak or birch.
- Plant spacing: The minimum distance required between plants. Plant spacing is often related to the plant size.
- Plant level: Different plant species that are placed in one group, because they have approximately the same plant spac-

ing. These groups are used in the vegetation model to process multiple plant species simultaneously.

- Plant patterns or patchiness: The patterns of the plants of a certain plant species. For example, plants of a certain plant can grow close to each other, having a high patchiness or they could grow scattered through the environment having a low patchiness.

Input The first part of this framework requires landscape maps and coverage statistics to generate a plant distribution. The coverage statistics must be dependent on the landscape maps, e.g. coverage statistics based on the height requires a height map. The second part of our framework requires input that defines the geometry of the 3D representation of the environment.

Thus, the inputs of our algorithm are:

- Landscape maps, for calculating plant distribution and 3D visualization of the environment. For example, a height map for the 3D visualization of the terrain.
- Statistics about the coverage per plant species, and about the patchiness of each plant species.
- One or more 3D models per plant species.

Requirements The goal of this framework is to translate landscape maps obtained from either ecologically or geographical data sets to an ecological sound plant distribution and generate from this distribution a real-time 3D web-based visualization. Therefore, the framework has the following requirements:

- Real-time 3D web-based visualization
- The input coverage and patchiness statistics with various landscape maps should match with the statistics of the generated plant distribution.
- Support for different plant species, sizes, and patterns.

Overview To meet the above requirements, the framework consists of two main components. The first component deals with the generation of plant distribution from landscape maps in combination with statistical data about coverage and patchiness of the plant species. This algorithm consists of two components: a plant position generation and a plant species generation component. The plant position component computes based on the input maps all possible plant positions in the environment. This step takes into account the required minimum distance between the different plants. The plant species generation component classifies the generated plant positions to one of the plant species or nothing. The reason for separating these components is that it gives us the possibility to accurately map the statistical information. Details of this algorithm can be found in Section 4.

The second part is to organize and translate the generated plant distribution to a real-time/interactive 3D web visualization. Simply, putting a 3D model at every generated location in the plant distribution does not work, because of the high density of the plant distribution, which results in high geometry complexity. Therefore, the plant distribution has to be organized into different levels of detail to reduce the complexity of the scene. We organize the distribution into three levels: Models, billboards, and "far-away" field. Models are placed close to the viewer to increase the realism of the scene. Billboards middle distance and the "far-away" field does not generate geometry for a plant, but only colors the underlying terrain. In Section 5 each of these representations and the transitions between them are discussed in more detail.

4 Vegetation Model

This section describes the algorithm to generate a plant distribution from landscape maps in combination with statistical data about coverage and patchiness of the plants in the environment. The algorithm is divided into two main components: plant position generation and plant species generation. In addition, we will discuss separately the multiple plant level support of the algorithm.

4.1 Plant position generation

The goal of this component is to generate all possible plant locations in the environment. In the next component, these positions are classified using the coverage and patchiness statistics of each plant species. To obtain all possible plant positions, we adopt the PDD with Wang tiling technique used by Alsweis [Alsweis and Deussen 2006] and Weier [Weier et al. 2013]. This technique gives the possibility to generate points randomly uniformly with a minimal distance from each other: the same behaviour that can be found in nature. The next paragraphs explain how these plant positions are generated.

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 provide information of where vegetation is located, for example NDVI, biomass, or coverage maps. The next step is to threshold the map given a user-defined threshold. Each tile with a value higher than the threshold is marked as vegetated.

Generate plant positions from vegetated tiles Points are generated with the PDD and Wang corner tiling technique [Lagae 2007]. Wang corner tiling is used to avoid the corner problem that appears in the regular Wang border tiling technique. A Wang tiling is created with only the tiles on the map that are marked as vegetated. Next, each Wang tile is filled with a PDD. The exact procedure of how the Wang tiling is created and used in combination with PDD is described in detail elsewhere [Onrust 2015]. The result of this process is a seamless point distribution where each point has at least a user-defined minimum distance to each other where only the vegetated tiles on the map contains points. The minimum distance is determined based on the plant size.

4.2 Plant species generation

The aim of the plant species generation component is to classify the generated point distribution. The classification is based on fractal neutral modeling techniques [Hargrove et al. 2002]. These 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 correctly translate static coverage and patchiness data. We extend this method to integrate with the generated point distribution, so that it is able to work with non-static statistical coverage and patchiness data. The next paragraphs explain step-by-step this classification procedure.

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. First, 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 multiple coverage values are extracted for the same plant species for a single point when multiple input maps are supplied (e.g. height map and a soil map). These coverage values are merged to a single value by taking the minimum value, because we assume that the minimum is the limiting grow factor for that plant species. The same process is applied for extracting the patchiness values. Patchiness is represented with two values: Hurst, to represent roughness, and patch area, for the size of the patterns.

Fractal generation The second step is to calculate a fractal value for every plant species of each point. Fractal values are used, because these values can be used to represent different kinds of patterns in nature [Hargrove et al. 2002]. The advantage of fractal algorithms is that they calculate a random value for a point that depends on the point location. This gives the possibility to generate similar random values for points that are close to each other or vice-versa. Thus, we can represent plants that grow close to each other or are scattered throughout the environment. Therefore, our fractal algorithm must be able to translate the input Hurst and patch area data to an individual fractal value for each point for every plant species. In addition, it is possible that the Hurst or patch area value are not static value values for every plant species, in contrast to neutral modelling techniques. Therefore, we use a modified fractal Brownian motion algorithm of which the exact details are discussed in [Onrust 2015]. This algorithm is able to generate fractal values for each point individually for every plant species using the Hurst and shape are input parameters.

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, each plant species calculates for every point an individual threshold value. 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 be used by that point.

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 plant species that belongs to both values. The result of this step is that each plant species gets assigned a set of points matching the coverage and patchiness input statistics. However, it may happen that certain points have been assigned to multiple plant species. These conflicts are solved by assigning the plant species with the highest fractal value for these points. The consequence is that certain plant species have less coverage than what is expected. Therefore, the remaining non-classified points have to be classified to compensate for the missing coverage of each plant species

Before the remaining points can be classified, it is first necessary to update the coverage values so that each plant species will have its expected coverage in the final plant distribution. First, the coverage statistics are updated for each input map by generating several reference points that are uniformly distributed over the complete range of the values of the input map. Next, for each reference point, the amount of total coverage is calculated that is obtained in the current plant distribution. This is compared with the expected coverage and, by subtracting the received 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 component.

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 each time on the new remaining point set, so that there are no conflicts generated. The main reason for this step is to stop the algorithm; otherwise, conflicts were most likely to be generated, and the same process repeated. The plant species with the highest standard deviation in their average patchiness statistics in comparison with the other plant species is processed first.

An example result of this complete process is shown in Figure 3 where a plant distribution is generated for an existing salt marsh located in the Netherlands. Different kinds of patterns and locations in the environment for each plant species (each color is a unique plant species) are clearly visible.

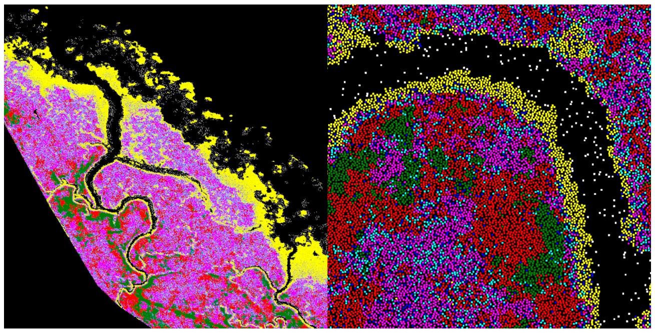

Figure 3: *Result generated by the vegetation model. On the left the complete generated plant distribution is shown for the Paulinapolder salt marsh in the Netherlands. On the right a more detailed view is provided. Each color is a different plant species.*

4.3 Multiple plant levels

In the previous sections, we assumed that all plants have approximately the same size, and thus, that all plants belong to the same plant level. However, the vegetation model also supports plant species with large difference in plant size, such as trees and flowers. This requires some small additions to both the plant position generation and plant species generation part.

Plant position generation Plant positions for multiple plant levels are generated semi-separately from each other. We start with generating plant positions from the largest plant level (i.e. 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. 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 of their plant level, or use for these points the same minimal distance of the current processed plant level. We use the last option, because we do not know yet if a point generated for a certain plant level is also classified to one of the plant species of that plant level. It is possible that during the classification a point is not assigned a plant species of that plant level. Then, we do not want to throw away this point but use it for the processing of plant species of the lower plant levels. With our choice, the integration will be seamless, while with the other option the point would be isolated, because it has much larger distance to the other points than it should have.

Plant species generation Again, the plant levels are processed sequentially from each other, starting with the largest plant level till the smallest plant level. Each plant level uses the plant positions that are generated for his plant level and the plant positions that have not been classified by the previous processed plant levels. The regular classification process is used for each plant level. After the classification of a plant level, there is the possibility to apply neighborhood influences, like in the previous work of Lane [Lane and Prusinkiewicz 2002], and Weier [Weier et al. 2013]. These neighborhood effects influences the coverage statistics of the neighboring non-classified points and it gives the possibility to model influences of for example trees on the neighboring smaller plants.

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. The structure is as follows: first, we explain the data organization of the plant distribution. Next, we discuss the transitions between the different data structures, and finally, we give an overview of the complete rendering framework.

Data organization For the 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 possible 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. Therefore, to reduce the geometry, it is necessary to use different LODs for the plants. This means that a different representation for a plant, other than its 3D model, has to be used, depending for example on the plant location relative to the viewer: it is not necessary to place a detailed 3D model far away from the viewer, because the details of such model cannot be seen anyway. For our framework, we adopt a LOD schema that divides the plant distribution in three zones depending on the location of the viewer. This schema is similar to a LOD technique proposed for the rendering of millions of grass blades [Boulanger et al. 2006]. 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, i.e. 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 texture applied on the terrain. The switching between zones is dependent on the distance to the viewing point, and can be configured as a user-defined threshold.

To use this LOD scheme, we had to solve another problem: it is not feasible to calculate for each plant its appropriate LOD representation, as this would require every frame to iterate over many thousands of plants on the CPU. Therefore, neighbour 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 quad tree structure [Deussen et al. 2002] [Bruneton and Neyret 2012]. The whole distribution is divided in four equal quads and each of these quads is again divided in four quads. This continues up to a number of iterations defined in the framework. A quad at a certain level in the quad tree is selected when it is within a certain distance from the viewer. This gives the possibility to place the smaller quads close to the viewer and larger quads further away. The same quad tree structure is used to organize the terrain data.

Transition between LODs Using different representation for the plant model at fixed distances from the viewer leads to another problem: 'popping'. The popping effect is noticed when plant representations switch between different LODs. This is an unwanted artifact and 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 model and billboard objects, and between billboards and terrain. The smoothing procedure for each transition is explained separately, although both procedures are based on the alpha blending.

The first step to blend both representations is to create a small, configurable, overlapping band 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 the plant is blended. This is calculated both for plant models, Formula (1), and for billboards, Formula (2):

$$a_{plantmodel} = clamp(\frac{(distance - b1)}{(b2 - b1)}, 0.0, 1.0) \quad (1)$$

$$a_{billboard} = 1.0 - clamp(\frac{(distance - b1)}{(b2 - b1)}, 0.0, 1.0) \quad (2)$$

The $distance$ factor is the calculated distance between the viewer's position and the plant position. The factors $b1$ and $b2$ are the distances to the closest border of the transition zone, and to the furthest border, respectively. The result is that plant models in the transition border have an alpha near to one when closest to the viewer, but gradually fade out as they get close 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 both representations. This is achieved by generating two separate images with one only the plant models and the others with billboards, during the generation of these images the alpha values are mapped to the alpha band of these images. Each of this images contains the alpha value for each pixel. Finally, a new image is generated by combining both images. The color of each pixel in the new image is a combination of the colors of the billboard and plant model image using formula 3 where $a_{plantmodel}$ and $a_{billboard}$ are the alpha values, and $p_{plantmodel}$ and $p_{billboard}$ the color value of the same pixel in both images. In the end, this process creates a smooth transition when switching between plant model and billboard objects.

$$pixel_{new} = a_{plantmodel} * p_{plantmodel} + a_{billboard} * p_{billboard} \quad (3)$$

The process for the transition between billboards and the terrain is similar as for the transition between plant model and billboard objects. Again, an overlapping zone is created and for billboards and terrain an alpha value is calculated. The alpha for billboards is calculated this time with Formula (1) and for the terrain with Formula (2). Again, billboards fade out using the alpha value, but the terrain does not use the alpha value to become semi-transparent. Instead, it used the alpha value to blend with the original color of the terrain. The result is that billboards gradually fade out and that the terrain takes over with a color that is similar to that of the billboards.

Rendering framework In this paragraph, we give an overview of the complete rendering framework, including the pre-processing steps, and the rendering steps. The first step is to prepare the scene for the actual rendering. During this step, a terrain mesh is generated from a raster height map, and the different LOD representations for the plants are generated. In addition, both the terrain and plant distribution are divided into a quad tree structure. The 3D plant models are generated offline by using a node-based 3D modeling tool called Construct, [Silva et al. TBP 2015] for which we developed a special L-systems plug-in. Billboards are generated from these models. The third representation is computed during the pre-processing phase by first finding for each vertex in the terrain mesh all the closest plants within a pre-defined distance. Then, for each vertex it is calculated how much each plant species contributes to the final color. The plant species with the maximum contribution is selected. At run-time, the terrain vertices receive the corresponding color.

The rendering step is divided into three parts. The first two parts are the rendering of the separate images for respectively the plant model and billboard objects with their underlying terrain and water. For the plant model objects shadows are computed for the models itself and the underlying terrain by using the percentage-closer soft shadow mapping technique is used [Fernando 2005]. Shadows are not computed for the billboards and their underlying terrain, because we want to minimize the rendering time. Instead, shadows are integrated into the texture of each billboards, and shadows on the ground are approximated by making the terrain color darker at place where plants are located. Finally, during the third part both images are blended together, as described above.

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, the user does not have to install 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* [Cabello 2010].

To reduce the data load to the GPU, we store all the terrain and plant data directly in the GPU by using VBOs (Vertex Buffer Objects). In the VBOs, plants are stored as a single instance with attached a list of their positions, scales, and rotations. During, rendering the stored instance is placed multiple times using the attached lists. This process is called geometry instancing and reduces the memory footprint as well.

The generation of billboards is more difficult, because WebGL currently does not support geometry shaders, which are an efficient way to generate different billboards [Bruneton and Neyret 2012], since a single point is sent to the GPU. An alternative solution would be to generate a complete quad, but this results in sending additional geometry to the GPU, which we want to limit. We decided to represent each billboard as a single point and draw it in the fragment shader as several pixels on the screen. The size of the point (in number of pixels) can be adapted, giving us the possibility to draw large points close to the viewer and smaller further away. Another advantage is that the billboards are always turned to the camera. The disadvantage of this method is that the point is always a perfect square, meaning that we process some unused fragments on the GPU for every billboard, when the billboard texture is not perfectly squared.

The transition between the plant model and billboard objects was achieved by rendering two separate images and then blend them together. WebGL provides support for FBOs (Frame Buffer Objects), which can be used to render results to it instead of to the screen. Therefore, the plant models and billboards are render both to a separate FBO in RGBA format. The RGBA format is important, because the alpha channel is used to blend both FBOs. Next, a final FBO is created which stores the blending result and renders it to the screen.

Shadows are computed with percentage-closer soft shadow mapping technique for both the plant model objects and terrain. The main reason for using this technique is that an implementation was already supported by the *three.js* framework.

7 Results and Discussion

We generated results for an existing area called the Paulinapolder, a salt marsh located in the South of the Netherlands. A plant distribution was generated from our vegetation model by using landscape maps about height and NDVI in combination with statistical coverage and patchiness data for each plant species. Next, with the visualization model, the height map and generated plant distributions are translated into our 3D representation, for interactive web visualization. In this section, we first present the results generated by our framework. Next, we discuss the performance of the visualization, followed by the validation of the results. Finally, we discuss the results and the complete framework.

Results The result of the vegetation model for the Paulinapolder input is shown in Figure 3. Figures 1 and 4 present a global visualization of the virtual Paulinapolder environment. Figure 5 captures the seamless transition between the different LODs. The same figure shows a close-up view of plant models in the environment. More results, including a video and the web visualization, are available (`https://graphics.tudelft.nl/benny-onrust`).

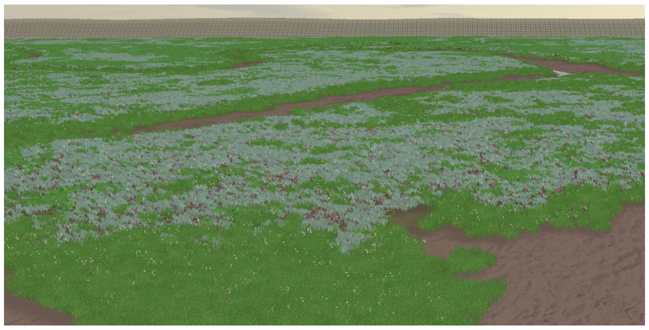

Figure 4: *Global overview of the virtual Paulinapolder.*

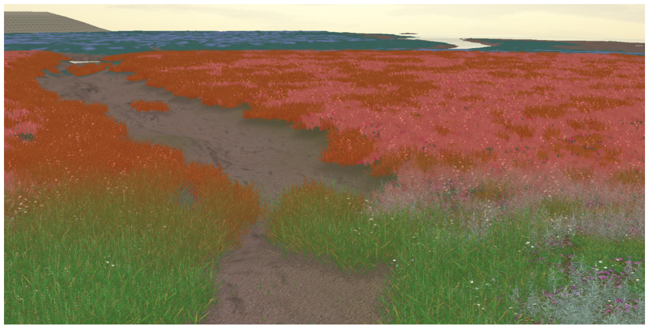

Figure 5: *The seamless transition between the different LODs. Far-away LOD is colored blue. Billboards are colored red, and the plant models have their regular color.*

Performance The complete scene has around 700,000 plants. A typical frame of this scene consists out of 2,2 million faces to render the plant models, terrain, water, and background. In addition, 380,000 points are necessary to represent the billboards. The rendering of a typical frame with a resolution of 1920x1080 takes around 165ms using a NVDIA Quadro k1000m GPU. Of this time, plant models are rendered in 65 ms, and their shadow computation takes 27 ms; billboards are rendered in 38 ms, terrain in 18 ms, and water in 5 ms.

Validation We validated our results by having ecologists judge the visualization. In addition, we calculated the coverage statistics from the generated distribution to show that the input coverage statistics are approximately equal. The validation with the ecologists confirmed that our framework is able to generate different kinds of patterns for each plant species depending in the input statistics. Figure 4 shows several of these different patterns ranging from very large patterns of plants that grow closely together, to small random patterns of plants that grow scattered throughout the area. A disadvantage of the current visualization is that when in global view some billboard objects tend to become too small which creates small gaps in the distribution. The local view of the plant model objects was deemed convincing and the zone transition was not noticeable, or in any case not distracting. The calculated coverage statistics from the generated plant distribution was found to be almost similar as the input coverage statistics [Onrust 2015].

Discussion The aim of this research was to procedural generate an ecological correct plant distribution from ecological maps and statistics. In, addition, the generated plant distribution should be translated to a convincing real-time 3D web visualization. Also, the plant distribution generation solution should be generic meaning that it should support different plant species and, patterns, and input data. The validation showed that these requirements were in general met. Using mathematical validation 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 showed that we were able create a convincing real-time 3D visualization. We also found several limitations in our presented framework.

The main limitation of the visualization model is in the representation of the billboards. Billboards are represented as a single point that are rendered as several pixels on the screen to maximize the rendering performance. This means that billboards are represented as a single slice, directly facing the camera independent from which angle. Billboards that are viewed globally with a birds-eye view do not look convincing, because the same slice that is used to view the billboards horizontally, is also used to view the billboards vertically. In the current visualization, this is often not visible, because all the plants are relatively small and have relatively monotonic color, but when the plants become larger, this will be noticeable. In addition, it is difficult to set automatically the correct size for each billboard for every viewing distance, because size is measured in the number of pixels. Figure 4 shows that the current representations leads to problems, because small gaps appear at the middle-range distance from the viewer around certain plant species.

Other limitation of the billboards is that the shadows of the billboards are approximated by using baked-in shadows in the texture and the shadow cast on the terrain is approximated by turning the terrain color slightly darker. The shadow cast does not use the actual shape of the plants. In Figure 5 it can be seen that the billboards objects have different shadows on the ground than the plant model objects. Shadow computation for billboards could be improved by (partly) replacing the current billboards with the concept of volumetric billboards [Decaudin and Neyret 2009]. 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 in this topic could greatly enhance the realism of 3D plant distributions, because it allows for the generation of more convincing billboards, which are the weakest aspect in the current visualization.

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 [Donnelly and Lauritzen 2006].

Finally, the transition between the plant models and billboards is in most cases smooth and there are 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 later to a billboard representation.

8 Conclusion

We presented the design and implementation of a framework for the generation and rendering of ecologically sound plant distributions and environments. The central issues of plant distribution generation from landscape maps and statistics, and the rendering of the plant distribution to a web visualization with interactive frame rates were addressed with a combination of techniques. 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. A convincing interactive 3D web visualization was created by using existing LODs, shadow mapping, instancing techniques. We tested our system by generating a plant distribution for an existing environment using its landscape maps and statistics. Ecologists validated our results and found the results in 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 improve and research 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. Further, it might be interesting to extend the framework by including the fauna of the environment to improve realism [Komodakis et al. 2005]. Finally, our framework has only been tested on environments with only grass-like plant species; we would like to do additional testing for other more forest-like scenes to investigate the quality and performance of its results.

Acknowledgements

We would like to thank Alex Kolpa for helping with the implementation of the L-system plugin in Construct.

References

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

Borsboom-van Beurden, J., Van Lammeren, R., Hoogerwerf, T., and Bouwman, A. 2006. Linking land use modelling and 3d visualisation. In *Innovations in design & decision support systems in architecture and urban planning*. Springer, 85–101.

Boulanger, K., Pattanaik, S., and Bouatouch, K. 2006. Rendering grass terrains in real-time with dynamic lighting. In *ACM SIGGRAPH 2006 Sketches*.

Bruneton, E., and Neyret, F. 2012. Real-time realistic rendering and lighting of forests. In *Computer Graphics Forum*, vol. 31, 373–382.

Cabello, R., 2010. three. js-javascript 3d library.

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

de Vriend, H. J., van Koningsveld, M., Aarninkhof, S. G., de Vries, M. B., and Baptist, M. J. 2014. Sustainable hydraulic engineering through building with nature. *Journal of Hydro-environment Research*.

Decaudin, P., and Neyret, F. 2009. Volumetric billboards. In *Computer Graphics Forum*, vol. 28, Wiley Online Library, 2079–2089.

Deussen, O., Hanrahan, P., Lintermann, B., Měch, R., Pharr, M., and Prusinkiewicz, P. 1998. Realistic modeling and rendering of plant ecosystems. In *Proceedings of the 25th annual conference on Computer Graphics and Interactive techniques*, ACM, 275–286.

Deussen, O., Colditz, C., Stamminger, M., and Drettakis, G. 2002. Interactive visualization of complex plant ecosystems. In *IEEE*, 219–226.

Donnelly, W., and Lauritzen, A. 2006. Variance shadow maps. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, ACM, 161–165.

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

Fanini, B., Calori, L., Ferdani, D., and Pescarin, S. 2011. Interactive 3d landscapes on line. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 3816*, 453–459.

Fernando, R. 2005. Percentage-closer soft shadows. In *ACM SIGGRAPH 2005 Sketches*, ACM, 35.

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

Hargrove, W. W., Hoffman, F. M., and Schwartz, P. M. 2002. A fractal landscape realizer for generating synthetic maps. *Conservation Ecology 6*, 1, 2.

Herzig, P., Englert, M., Wagner, S., Jung, Y., and Bockholt, U. 2013. X3d-earthbrowser: visualize our earth in your web browser. In *Proceedings of the 18th International Conference on 3D Web Technology*, ACM, 139–142.

Komodakis, N., Panagiotakis, C., and Tziritas, G. 2005. 3d visual reconstruction of large scale natural sites and their fauna. *Signal Processing: Image Communication 20*, 9, 869–890.

Lagae, A. 2007. *Tile-Based Methods in Computer Graphics*. PhD thesis, Katholieke Universiteit Leuven.

Lane, B., and Prusinkiewicz, P. 2002. Generating spatial distributions for multilevel models of plant communities. In *Proceedings of Graphics Interface*, 69–80.

Liu, Q.-X., Herman, P. M., Mooij, W. M., Huisman, J., Scheffer, M., Olff, H., and van de Koppel, J. 2014. Pattern formation at multiple spatial scales drives the resilience of mussel bed ecosystems. *Nature communications 5*.

Molofsky, J., and Bever, J. D. 2004. A new kind of ecology? *BioScience 54*, 5, 440–446.

Onrust, B. 2015. *Automatic generation of plant distributions for existing and future natural environments using spatial data*. Master's thesis, Delft University of Technology.

Pettit, C. J., Raymond, C. M., Bryan, B. A., and Lewis, H. 2011. Identifying strengths and weaknesses of landscape visualisation for effective communication of future alternatives. *Landscape and Urban Planning 100*, 3, 231–241.

Rietkerk, M., and Van de Koppel, J. 2008. Regular pattern formation in real ecosystems. *Trends in Ecology & Evolution 23*, 3, 169–175.

Saura, S., and Martínez-Millán, J. 2000. Landscape patterns simulation with a modified random clusters method. *Landscape ecology 15*, 7, 661–678.

Schwarz, C. 2014. *Implications of biogeomorphic feedbacks on tidal landscape development*. PhD thesis, Radboud University Nijmegen.

Silva, P. B., Eisenmann, E., Bidarra, R., and Coelho, A. TBP 2015. Procedural content graphs for urban modeling. *International Journal of Computer Games Technology*, 12.

Smelik, R. M., Tutenel, T., Bidarra, R., and Benes, B. 2014. A survey on procedural modelling for virtual worlds. In *Computer Graphics Forum*, vol. 33, 31–50.

Temmerman, S., Bouma, T., Van de Koppel, J., Van der Wal, D., De Vries, M., and Herman, P. 2007. Vegetation causes channel erosion in a tidal landscape. *Geology 35*, 7, 631–634.

van Lammeren, R., Houtkamp, J., Colijn, S., Hilferink, M., and Bouwman, A. 2010. Affective appraisal of 3d land use visualization. *Computers, environment and urban systems 34*, 6, 465–475.

Wang, S.-Y., Lin, C.-K., and Tai, W.-K. 2013. Compressing 3d trees with rendering efficiency based on differential data. *IEEE Transactions on Multimedia 15*, 2, 304–315.

Weier, M., Hinkenjann, A., Demme, G., and Slusallek, P. 2013. Generating and rendering large scale tiled plant populations. *Journal of Virtual Reality and Broadcasting 10*, 1.