Using procedural methods to generate realistic virtual rural worlds

Minor Dissertation presented in partial fulfilment of the requirements for the degree of Master of Science in Computer Science

by

Harry Long

Supervised by:

James Gain and Marie-Paul Cani

February 2016

I know the meaning of plagiarism and declare that all of the work in this document, save for that which is properly acknowledged, is my own.

Contents

1 Introduction **5**

1.1 Research Goals . . . 6

1.2 Contributions . . . 7

1.3 Structure . . . 7

2 Background **8**

2.1 Terrains . . . 9

2.2 Rivers & Streams . . . 10

2.3 Vegetation . . . 16

2.3.1 Explicit Placement . . . 16

2.3.2 Probabilistic Placement . . . 18

2.3.3 Simulators . . . 22

2.3.4 Conclusions . . . 26

List of Figures

1.1 Example of procedurally generated content. From top to bottom, left to right: Procedurally generated river stream [DGGK11b], procedurally generated terrain through sketching [GMSe09], procedurally generated plant [SSBR01] 6

2.1 A single iteration of midpoint displacement for the creation of mountains [PHM93]. New vertices y_A, y_B and y_C are created and shifted vertically by a random offset 12
2.2 Single production of midpoint displacement adapted to river generation [PHM93]. Given the initial triangle, four valid split scenarios. 13
2.3 Waterfall classifications [Emi14] 15
2.4 Using explicit placement as input examplars for reproduction [EC15] . . 17
2.5 Reconstructed roadside vegetation using orthophotos [ACV$^+$14b] 17
2.6 Point distributions with associated pair correlation histogram [Emi14] . . 19
2.7 Radial distribution analysis 19
2.8 Vegetation generated using predefined ecosystems [Ham01] 22
2.9 Plant growing towards light source [SSBR01] 24
2.10 Plant placement using an ecosystem simulator modelled by L-Systems . . 25

List of Tables

2.1 Pros and cons of individual techniques 27

Chapter 1

Introduction

Creating detailed virtual worlds can be a tedious task for artists. Indeed, modelling terrain, vegetation, water streams, rivers, water reserves, soil, rocks, buildings and road networks for large virtual worlds "by hand" can be extremely burdensome. This is especially true when realism is a key requirement. The increase in size and complexity of these virtual worlds mirror that of the processing capabilities of computing hardware. As a consequence, the task is only getting worse.

A popular technique to overcome the burden of repetitive tasks is to have them automated. This involves generating algorithms which, given a set of input parameters, generate the required content automatically. This is called *procedural content generation* and has already been successfully applied in different areas of computer graphics including: the generation of non-repetitive textures [aEL99, LLX+01, WLKT09], modelling plants [BPC+12, FZS+08, GFJ+11, Lew99b], generating terrains [SKG+09, GMSe09, DP10], generating river networks [DGGK11b, EC15] and generating city landscapes [GMN14, KM07, PM01] (figure 1)
A common difficulty with these methods, however, is finding the appropriate input parameters for the procedural algorithms. The correlation between the parameters and the resulting content is often unintuitive and, as a consequence, often comes down to iterative trial-and-errors until a "close enough" result is found. To overcome this, interactive techniques are often used in an attempt to make generating the input parameters more intuitive. These range from simple paint tools such as lassos and brushes [EC15] to sketch-based recognition algorithms [GMSe09].

The intent of this thesis is to develop procedural algorithms to automate the generation of virtual rural worlds. The input parameters for the procedural algorithms must be interactive and/or self-explanatory.

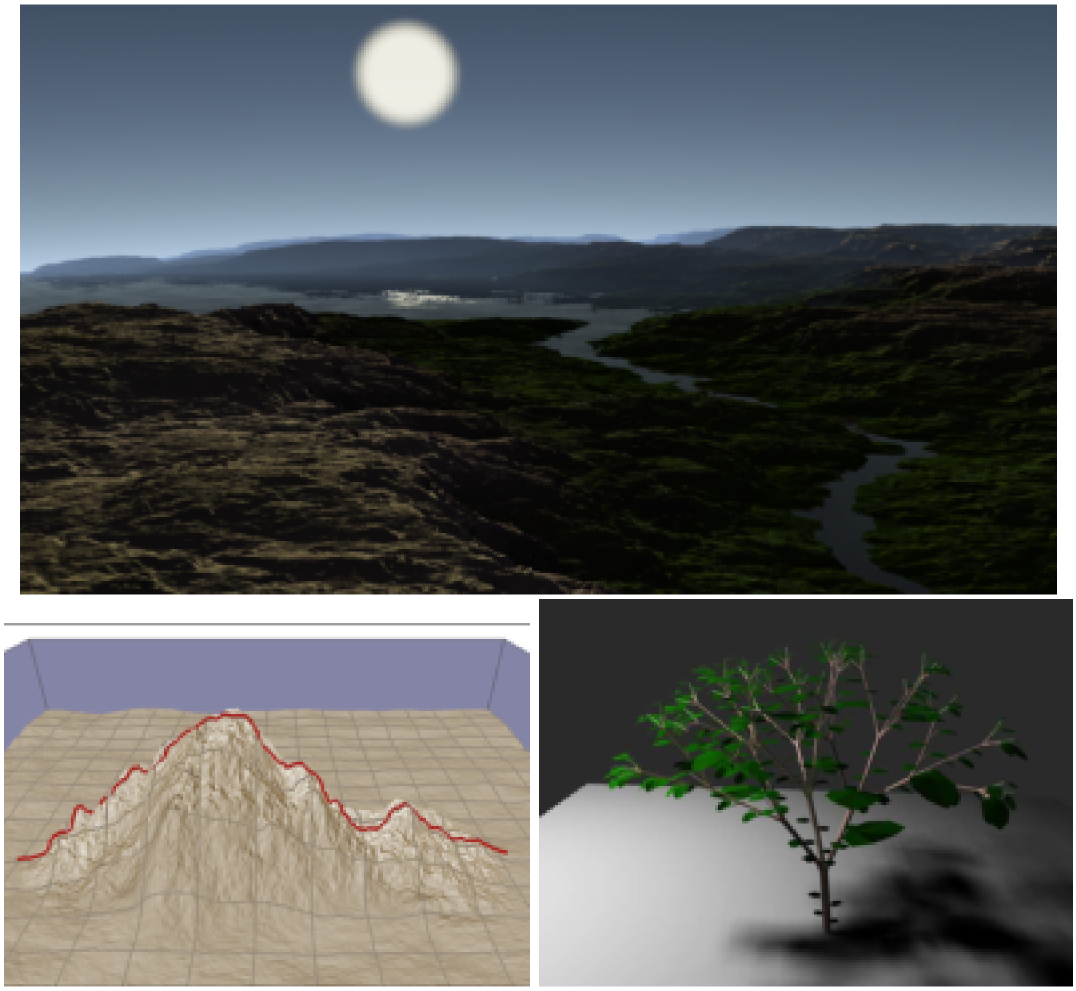

Figure 1.1: Example of procedurally generated content. From top to bottom, left to right: Procedurally generated river stream [DGGK11b], procedurally generated terrain through sketching [GMSe09], procedurally generated plant [SSBR01]

1.1 Research Goals

The research goals for this project are as follows:

- Develop procedural methods to automate the generation of realistic virtual rural worlds.
- Provide intuitive and smart controls.
- When possible, make interactions real-time.

One of the most important aspect of rural landscapes is vegetation. As such, our *first goal* must strongly focus on the insertion of plants. The automation provided should not limit user control and the flexibility of the system. For example, it must be possible to generate worlds with varying elevations, river networks, water sources and vegetation.

For the *second goal*, lots of thought must be put into making all user oriented controls intuitive. To do so, it will be important to research the pros and cons of other graphical applications in terms of control. If need be, multiple prototype controls should be developed in an attempt to find the best suited.

Maintaining a continuous feedback loop between user action and corresponding reaction is extremely important for both user-friendliness and to optimize usage. In an attempt to meet our *third goal* therefore, efficient algorithms must be developed in order to keep there time complexity to a minimum. When suited, these algorithms should be developed to run on the GPU.

1.2 Contributions

State contributions of this thesis

1.3 Structure

Outline structure of the thesis

Chapter 2

Background

This chapter gives an overview of previous work related to our topic. Procedural methods applied to computer graphics is a wide area of research with an exhaustive number of publications. As a consequence, we cannot pretend to review all this work. Instead, we will focus on research which is closely linked to the generation of virtual rural worlds.

Our work will not focus on modelling terrain relief but rather terrain content. As a consequence, material focused on procedurally generating terrain will not be reviewed. In this chapter will focus on the two primary constituents of rural terrains - water and vegetation.

2.1 Terrains

Terrain.

2.2 Rivers & Streams

KELLEY ET AL. –¿ Terrain simulation using a model of stream erosion

TERRAIN ADAPTED TO RIVERS

In the work by Kelly et al. the user specifies, on a horizontal plane, the terrain outline along with the main trunk stream. The terrain outline is used to configure the terrain extremities once ported to a 3 dimensional space. The main trunk stream specifies the path which the highest order water stream should follow on the resulting terrain.
Given the terrain outline and the position of the initial main trunk stream, the system calculates the drainage area the stream is responsible for. If this value surpasses a preconfigured constant, the stream will be split to form new streams, each channelling a smaller drainage area. This constant depends on the type of soil as resistant materials (e.g. stone) will be stronger and less susceptible to splitting than weaker ones (e.g. clay). This splitting process is repeated iteratively until a balance is found where each stream is able to channel their associated drainage area. After which a plausible elevation is calculated for each stream junction and, subsequently, a plausible terrain generated.

Derzapf ET AL. –¿ RIVER NETWORKS FOR INSTANT PROCEDURAL PLANETS

TERRAIN ADAPTS TO RIVER

In their work, Derzapf et al. permit the creation of procedural planets at any scale in real-time. To do so, only a rough representation of the planet is generated and detailed content is produced on-the-fly when the user navigates through it. This has the advantage of keeping memory usage minimum whilst not compromising on realism. To ensure updates are performed in real-time, their algorithms are implemented to take use of the massively parallel architecture of GPUs.
To initialise high-level planets, the system first creates the base mesh with all contained vertices representing the sea. The system then selects a given number of seed continent vertices and allows them to spread until a user-configured land-to-water ratio is reached. At this point, the "rough" planet is a mesh with each vertex assigned one of two labels: continent or sea.
To create the rivers the system iterates through all continental vertices in pseudo-random order to find those that are adjacent to a sea vertex. When such a vertex is found, it acts as the river mouth from which adjacent continental vertices chosen pseudo-randomly are connected. This is performed iteratively to form the river networks.
To assign ground altitudes to the river vertices the system employs the following formula for each river vertex, starting from the river mouth:

$a_v = a_u + e_a l_e \xi, e_a = \frac{a_{max_river}}{l_r}$

Where:

- a_v is the ground altitude of the current vertex.
- a_u is the ground altitude of the previously processed vertex (or zero if v is the first vertex).
- e_a is the average ground elevation.
- l_e is the length of the current vertex.

- $\xi \in [0, 1[$ is a uniformly distributed pseudo-random number.
- $a_{max_{r}iver}$ is the user-configured maximum river altitude.
- l_r is the current river length.

When the ground altitudes have been assigned, the following formula is used iteratively on each river vertex to assign water altitudes:
$w_v = a_v + e_w l_e, e_w = \frac{\epsilon_{river}}{l_{cr}}$
Where:

- w_v is the water altitude of the current vertex.
- a_v is the ground altitude of the current vertex.
- e_w is the average water elevation.
- l_e is the length of the current vertex.
- ϵ_{river} is the maximum river depth.
- l_{cr} is the distance from the current vertex to the river spring.

Once this has been performed, the "rough terrain" is comprised of continental vertices, river vertices and sea vertices, each with a defined altitude.

All randomness in the algorithms depends on a configured seed value, enabling virtual virtual worlds to be easily reproducible. This has the added advantage of permitting networked users to explore the same virtual world with very little data exchange.

SMELIK ET AL. –¿ INTERACTIVE CREATION OF VIRTUAL WORLDS USING PROCEDURAL SKETCHING

RIVER ADAPTS TO TERRAIN, EXPLICIT USER CONTROL

In their work, Smelik et al. create an interactive system which permits users to model a complete virtual world with content ranging from rural features (mountains, rivers, etc.) to man-made ones (buildings, road networks). In order to promote user-intuitiveness, they focus strongly on user controls and develop novel interaction tools which include sketch recognition.
When modelling the virtual world, interactions are split into two modes: **Landscape** and **Feature**. *Landscape mode* permits the designer to paint ecotopes onto the terrain using traditional image editing tools. These ecotopes are predefined by the user and encompass both elevation and soil material information. In *feature mode*, the user is able to place terrain content which include rivers. To place these rivers, the user must sketch vector lines outlining the core path of the river. Based on this information, a suitable course is plotted through the landscape, river banks inserted and other terrain features to which the river takes precedence adapted. For example, if the river is plotted to pass through a forest, trees on the rived bed and bank will be removed.

Pmsinkiewicz ET AL. –¿ A fractal model of mountains and rivers Musgrave et al. –¿ The synthesis and rendering of eroded fractal terrains — Also fractal

TERRAIN ADAPTS TO RIVER

In their work, Pmsinkiewicz et al. use an adapted midpoint-displacement technique to procedurally generate plausible rivers on terrain. Midpoint-displacement is a fractal technique heavily used for the creation of realistic looking terrain height-maps. Given a starting triangle representing the terrain, A, midpoint-displacement iteratively subdivides A into four smaller triangles. Each time new triangle vertices are created they are displaced vertically by a random offset. This process is repeated until a given recursion limit is reached. See figure 2.2 for an example of a single iteration of the process.

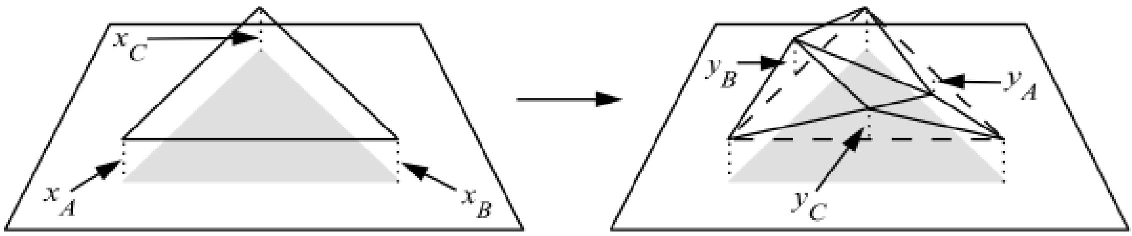

Figure 2.1: A single iteration of midpoint displacement for the creation of mountains [PHM93]. New vertices y_A, y_B and y_C are created and shifted vertically by a random offset

To adapt this method to the generation of rivers on the terrain, the edges of each newly formed triangle are labelled as *entry*, *exit* or *neutral*. An entry edge defines the point of entry for the river into the triangle, an exit edge the point of exit and a neutral edge prevents the river from passing through.

When a production step is applied and a triangle split, the following constraints must be applied:

- An entry edge must split into an entry and a neutral edge.
- An exit edge must split into an exit edge and a neutral edge.
- A neutral edge must split into two neutral edges.
- The newly formed edge-pairs within the triangle must either be "entry/exit" or "neutral/neutral".

See figure 2.2 for example valid productions.

One difficulty with this technique is to ensure two adjacent triangles are coherent once they split. That is, that the exit edge of one coincides with the entry edge of the other edge. To overcome this, the location of the vertices of an edge are used as the key to a random number generating hash table which, depending on its output, determines which segment will be crossed by the river.
When complete, a river path which is guaranteed not to cross itself is created. Post-processing is then used to create realistic terrain surrounding the riverbed.

Belhadj et Audibert –¿ Modeling landscapes with ridges and rivers: bottom up approach

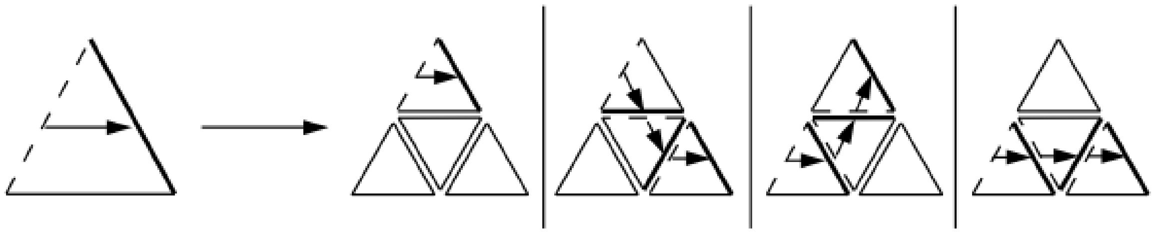

Figure 2.2: Single production of midpoint displacement adapted to river generation [PHM93]. Given the initial triangle, four valid split scenarios.

RIVER ADAPTED TO TERRAIN

In their work Belhadg et Audibert [BA05] automate the creation of ridges and river networks on the terrain. To create the ridges, particle pairs are first placed at random locations on the terrain. These particle pairs are then randomly assigned an axis and, iteratively, both particles distance themselves in opposite directions, perpendicular to this axis. At each iteration, a new height is calculated for the vertex. The heights decrease with distance from the start point following a Gaussian distribution.
To create the river networks, river particles are placed on the top of the newly generated ridges and a simple physical simulation is used to model the motion of these particles on the terrain. The path followed by these particles are tracked and, when two paths intersect, their particle velocity and mass are combined. When all particles have stopped moving the simulation is deemed balances and all particle paths which do not lead to terrain extremities are discarded. The remaining particle paths are kept and form the river network. To initialise the remaining vertices on the terrain, interpolation is performed based on the height of its surrounding river and ridge vertices.

Teoh, Soon Tee –¿ River and coastal action in automatic terrain generation

RIVER ADAPTED TO TERRAIN

The work by Soon Tee [Teo08] permit the procedural generation of terrains, river meanders, river deltas, coastal cliffs and coastal beaches. Similarly to the work by Belhadg et Audibert [BA05], the user starts by creating ridges and mountain ranges that form the base terrain. When this is done, the user is able to place seas, lakes and rivers.
Two methods can be used to place water reserves (lakes and seas) in their system: *flood filling* and *locating substantial local minima*.
Flood filling is a commonly used technique to place water on terrains and requires the user to click a point on the terrain which will act as the seed point for the water surface. This seed will then propagate iteratively to surrounding points which are at a lower height until there are no more lower points to propagate to.
With *locating substantial local minima*, the system automatically detects locations on the terrain which are local minimums and could therefore cater a lake or a sea. The user can then raise the surface height manually.
To place rivers on the terrain the user can either explicitly point out the start of the stream or the system can generate them by performing a simulation based on total rainfall, wind direction and terrain relief. To explicitly point out the start of the stream, the

user must simply click the relevant point on the terrain. A simplified model of gravity is subsequently used where the water is iteratively evacuated to its lowest surrounding point. This process finishes and the river deemed complete when the path reaches a local minima or terrain borders.
The simulation approach requires the user to specify wind direction and maximum rainfall. Then, starting from the source of the wind, the system simulates clouds moving in the direction of the wind with precipitation increasing with altitude until all rainfall has been depleted. The path taken by the rain and consequentially the rivers, is then determined similarly to the method mentioned previously for explicit river instantiation.

A. Emilien et al. −¿ Interactive Procedural Modelling of Coherent Waterfall Scenes

Emilien et al [Emi14] focus their research on the lesser explored area of procedurally generated waterfall scenes. They model waterfalls as three separate segments: running water, free-fall and pool. Running water segments are rivers and streams which are in continuous contact with the terrain. Free-fall segments are the "waterfall" part where there is no longer contact with the terrain. Lastly, pool segments outline the basin formed at the location where the free-falling water reaches the ground.
Given a terrain, the user models running water and pool segments using control points and free-fall segments using parabola. The control points for the running water and pool segments are not constrained to being in contact with the terrain as the terrain will later be adapted to fit these points. The only constraint is that the path must flow downhill. Based on this input, the system automatically calculates plausible water flow intensities. These intensities can be overridden by the user, however, for fine control.
Using the flow path, water flow intensities and waterfall classifications (figure 2.2), the system modifies the terrain to generate plausible river networks and waterfalls.

Gnevaux et al. −¿ Terrain generation using procedural models based on hydrology

Gnevaux et al. [GGG+13] permit users to procedurally generate terrains based on models of hydrology. To so, the user first sketches the contour of the terrain and optionally some river paths within the terrain and output nodes on the contour which will represent the river mouth. If no candidate output nodes are specified by the user, they are determined heuristically from geomorphology data based on the shape of the contour. These seed nodes are then iteratively expanded to form a larger network of rivers. To do so, on each iteration a candidate node is selected for expansion based on elevation and priority. The weighting between elevation and priority is configured by the user and has a large effect on the resulting river network. For example, by prioritising candidate nodes at lower elevations, lowlands will be generated first.
Rewriting grammar is used to perform node expansion. Configured values of $\rho_a, \rho_s and \rho_c$ influence the probability of selecting production rules favouring asymmetric branching, symmetric branching and continuation without branching respectively. The position for the new node is then selected using the following constraints:

- It should be at a minimum distance from existing nodes and edges.
- The new node should be at a greater distance from the terrain contour.
- The new node should be compatible with the elevation constraints of existing nodes.

If a position satisfying these constraints is found, a new node is added at the given position and the process is repeated.

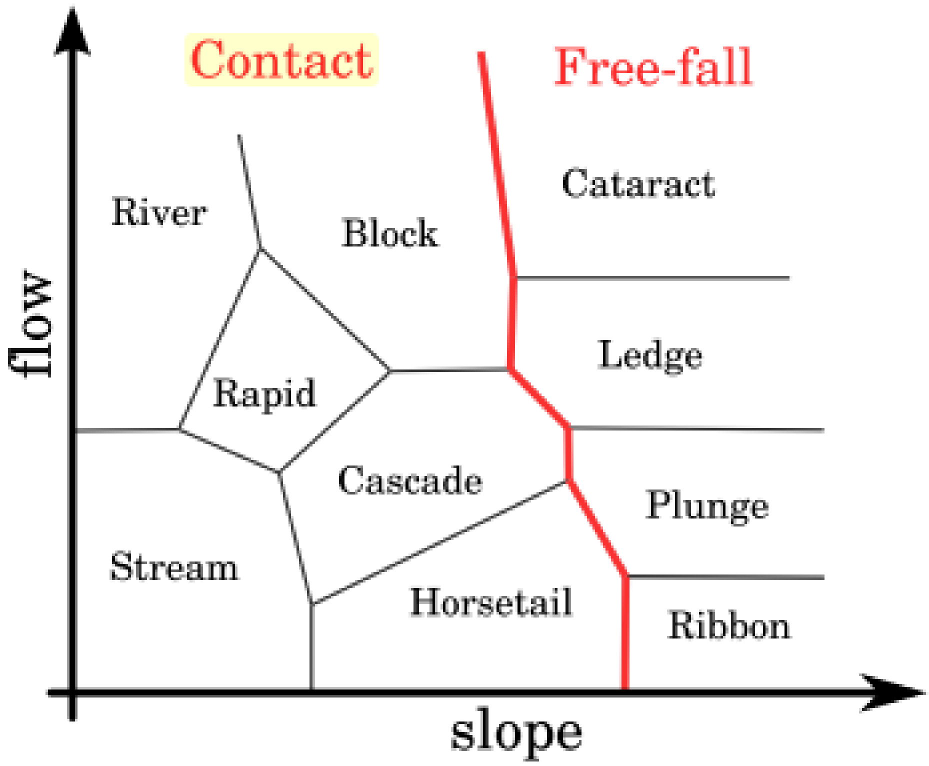

Figure 2.3: Waterfall classifications [Emi14]

The system then creates Voronoi cells centered around the river network nodes. Cells belonging to individual watersheds are then concatenated and plausible heights are generated for all points within those cells in order to keep the river flowing downwards at all times. Based on the calculated flow intensity (based on slope), water drainage and soil type, appropriate rivers are then placed.

t'Ava et al. –¿ Interactive terrain modelling using hydraulic erosion

t'Ava et al. [vBBK08] simulate hydraulic erosion in order to form water networks on the terrain in real time on the GPU. To simulate the movement of water on the terrain, the terrain is first split into equal sized (configurable) columns. Then, given given a seed volume and location for the water, a hydrostatic pipe-model simulation is used where, based on column heights, fluid density and gravitational acceleration, water is iteratively evacuated from each column to one of its surrounding column until a balance is found. Whilst the water is being routed through the terrain *force-based* and *dissolution-based* erosion is simulated. *Force-based* erosion is a direct consequence of the the force of the water on the terrain surface (figure **??**. Dissolution-based erosion is a consequence of the water mass on the terrain surface under the water and is most often characterised by a smoothing effect.

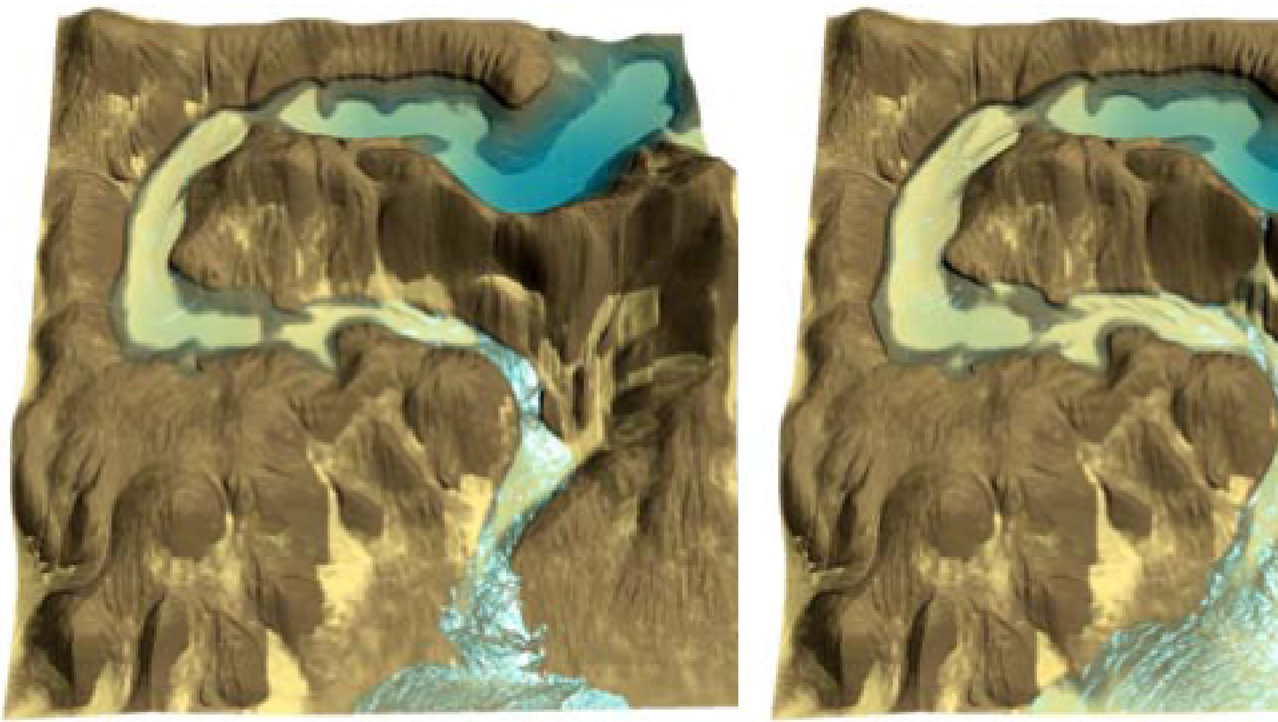

Figure 2.4: Simulation of the effect of force-based erosion caused by running water [vBBK08]

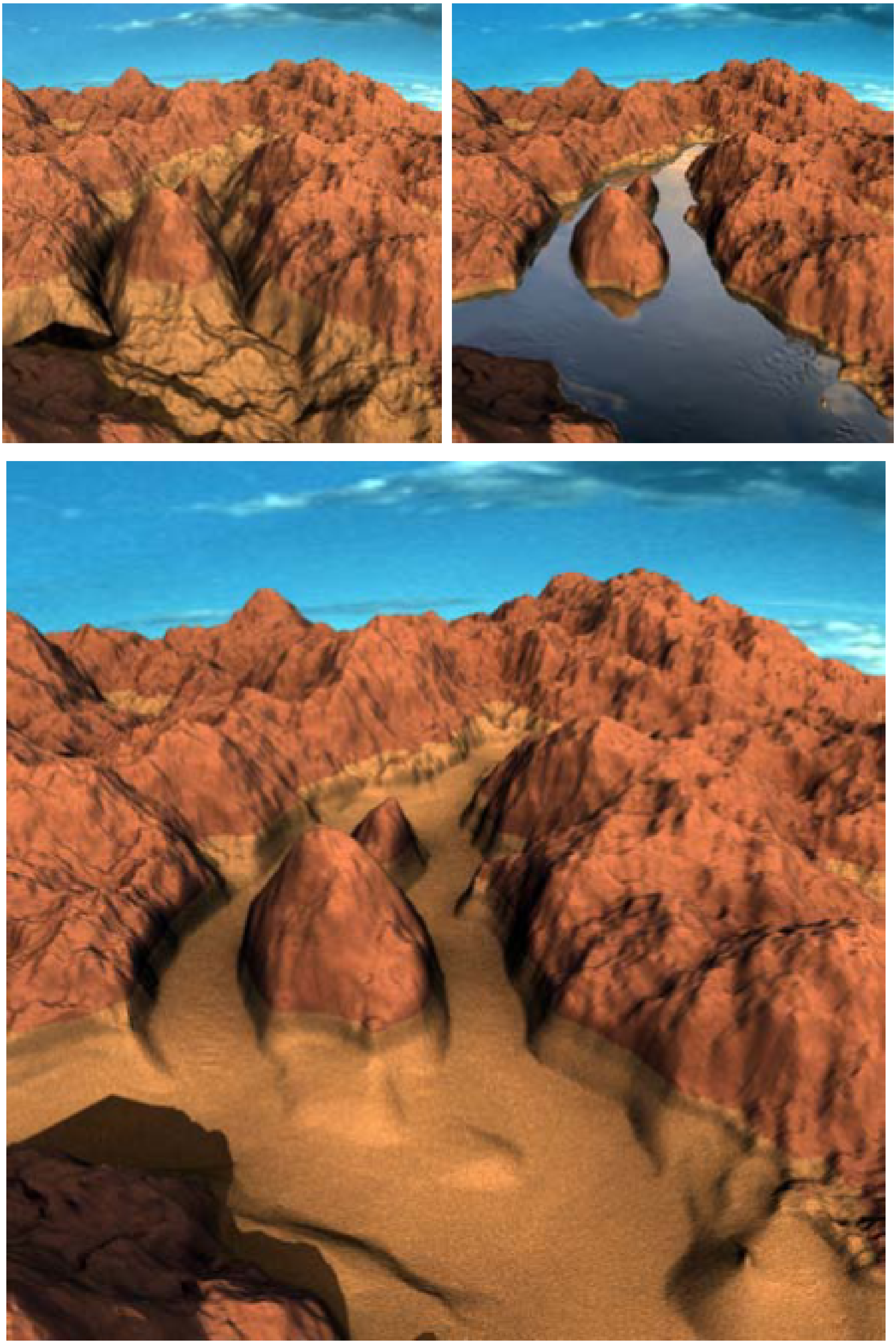

Figure 2.5: Simulation of dissolution-based erosion erosion caused by water movement[vBBK08]

2.3 Vegetation

Vegetation is core to rural landscapes. The species present along with their associated densities create a relationship between ecosystems and areas on earth on which resources are adequate. To ensure realism in virtual environments, much emphasize must be put on efficiently modelling these underlying ecosystems.

This section will review different methods to generate suitable vegetation for virtual worlds. These methods can be split into three main categories: *Explicit Instancing, Probabilistic Instancing* and *Plant Growth Modelling*.
Explicit Placement require explicit user-input to directly or indirectly pinpoint exact locations for individual plant instances.
Probabilistic Placement methods use statistical models to generate suitable vegetation.
Simulators attempt to algorithmically reproduce plants battling for available resources.

We will measure the success of these techniques based on the level of automation they provide, the realism they achieve, their computational cost and their adaptability. Adaptability, here, represents the ease at which a given technique is able to model a number of different vegetative scenarios.

2.3.1 Explicit Placement

Explicit placement methods require input from the user to determine the location and properties of individual plants.

Arnaud et al. [EC15] permit users to insert individual plants manually by simply clicking a given location on the terrain. To overcome the tedious task of manually placing individual plants on large terrains, the system is able to analyse existing distributions for reproduction. For example, to generate a large forest, the user is only required to generate a small subsection which can then be used to reproduce it on any scale (figure 2.3.1)

Similarly, Deussen et al. [DHL+98] allow users to use grayscale raster images as input to specify terrain vegetation. The location of individual plants is determined by pixel location whereas plant properties are correlated to pixel intensity.

In their work focused on improving the realism of roadside landscapes, Andujar et al. [ACV+14b] use orthophotos as input to determine the location and properties of individual plants. Unlike ordinary aerial photographs, aerial orthophotos use normalisation techniques to take into account terrain relief and camera tilt. The result is an image with uniform scale throughout which, similarly to a map, can be used to accurately measure distances between points. These orthophotos are used to measure the distances between plants. To later reproduce the roadside landscape, they use a dart throwing algorithm to place individual plants whilst respecting the appropriate distances.

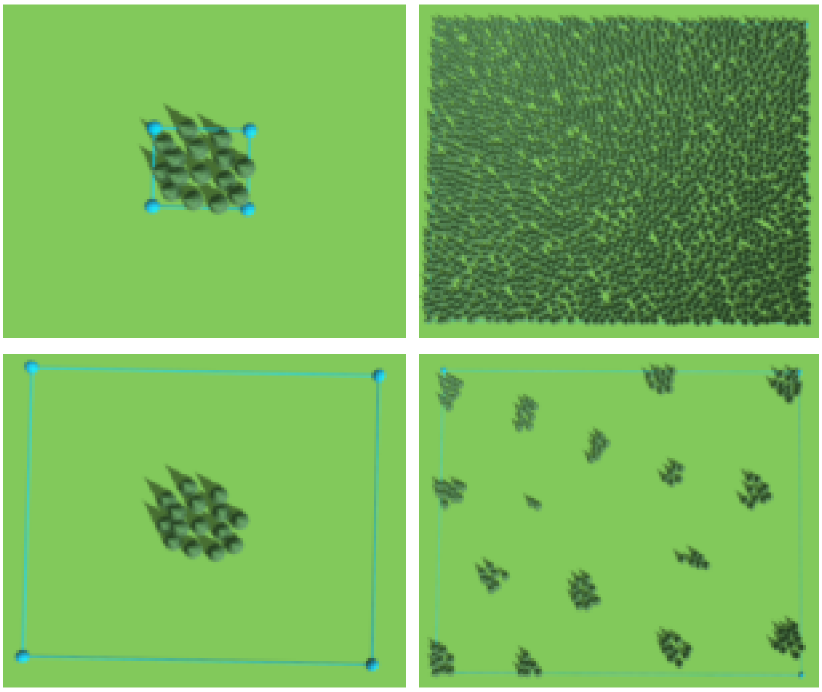

Figure 2.6: Using explicit placement as input examplars for reproduction [EC15]

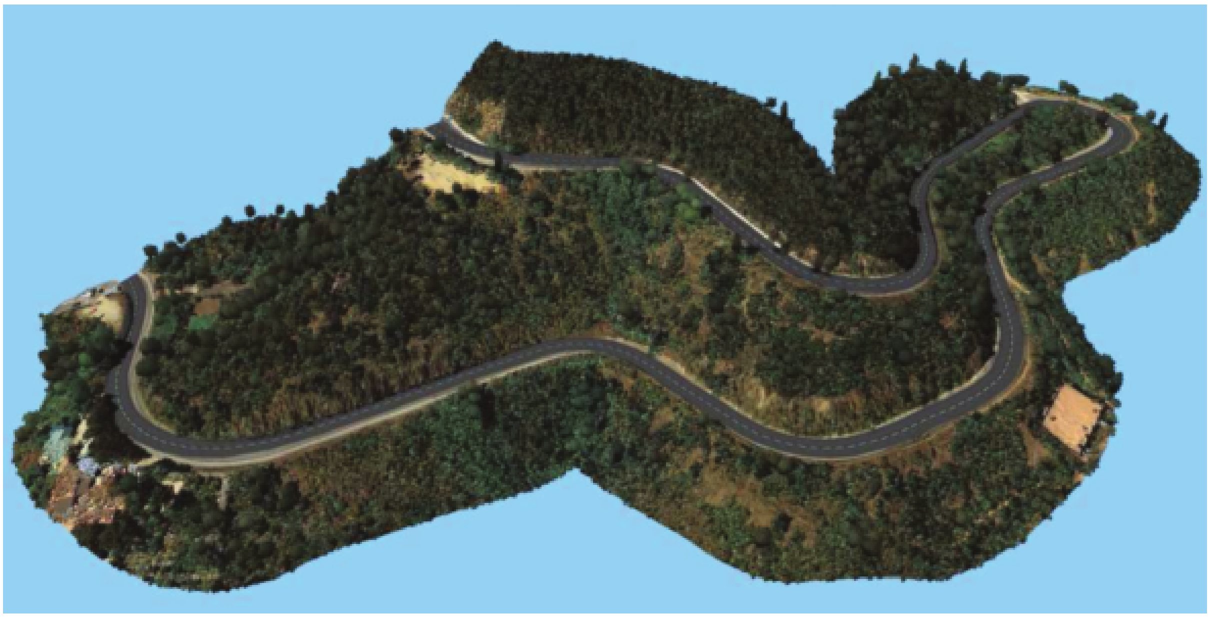

Figure 2.7: Reconstructed roadside vegetation using orthophotos [ACV+14b]

CONCLUSIONS

Explicit placement methods provide significant user control over the resulting virtual world. However, as there is *little to no automation* of this process, it can be very tedious and time consuming for the user. This is especially true when the virtual world being created are very large (e.g. open world video games). An advantage of this limited

automation, however, is that modifications are most often very small and are therefore performed in real-time.
The *adaptability* of these methods are very poor. Running a different scenario would most often involve starting explicitly placing individual plants from scratch.
Creating vegetation for large virtual worlds using these methods is extremely strenuous and, as a consequence, realism is often compromised.

2.3.2 Probabilistic Placement

Probabilistic placement methods use statistical models in an attempt to produce adequate vegetation. These methods can be further split into two sub-categories which are discussed in further detail below: *Radial Distribution Analysis* and *Predefined Ecosystems*. *Radial Distribution Analysis* approaches analyse the underlying distribution of the vegetation for later reproduction. *Predefined Ecosystems* calculate, based on the varying resources of the terrain and a set of predefined ecosystems, the best suited.

RADIAL DISTRIBUTION ANALYSIS

Work by Emilien et al. [EC15], Boudon et al. [BM07] and Lane et al. [LP02] use radial distribution analysis to convert to metric form the underlying plant distributions of input examplars. The data generated by the analysis stage can later be used to synthesise, at any scale, new point distributions which respect the characteristics of the input exemplar. For example, by analysing the positions of individual plants in a small subset of a forest and using it as the input exemplar, it is possible to reproduce it at a much larger scale in order to model its full size counterpart.

Analysis Generating the analytical data involves measuring the distances between individual points of different categories from the input examplar. For plant distribution analysis, the points represent individual plants and the categories represent the different species.

Before performing the analysis, the following parameters are configured:

- $\mathbf{R_{min}}$: The minimum distance from which point distances need to be analysed.
- $\mathbf{R_{max}}$: The maximum distance after which point distances don't need to be analysed.
- **Bin size**: When analysing the distances of given points, it is necessary to aggregate the points which reside at similar distances into bins. The bin size is the range represented by a single bin.

A core part of radial distribution analysis is generating pair correlation histograms for each category pair combination. A pair correlation histogram H_{AB} represents the variation in the distance between points of of category C_A and C_B ranging from R_{min} to R_{max} in *bin size* increments (figure 2.3.2)

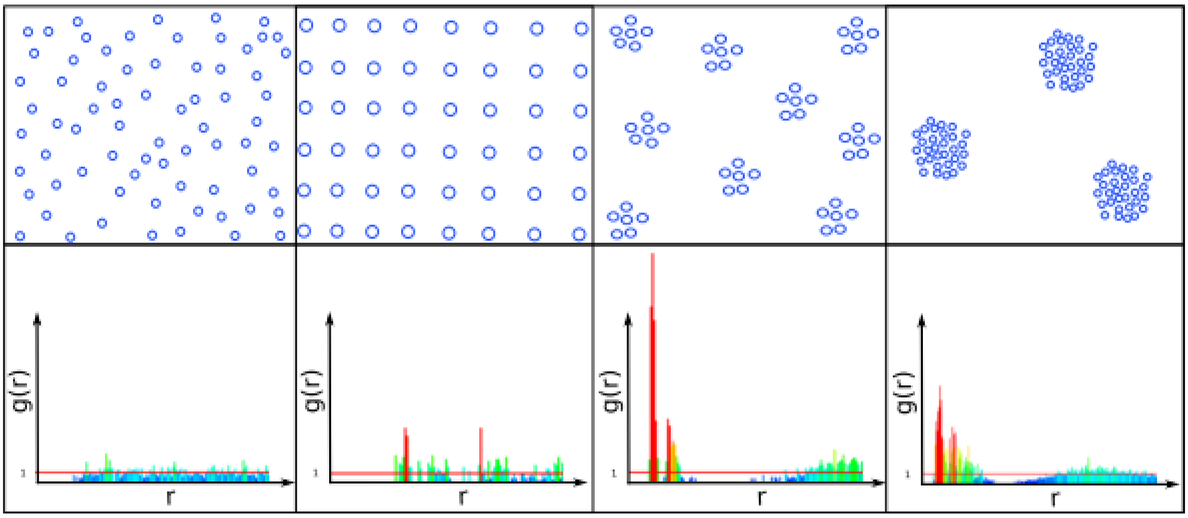

Figure 2.8: Point distributions with associated pair correlation histogram [Emi14]

To generate the pair correlation histogram H_{AB}, the algorithm iterates through each reference point of category C_A and, for each destination point of category C_B at a distance between R_{min} and R_{max}, increments the relevant bin in the histogram. In figure 2.3.2, for example, are being measured the points that lie within the annular shell of radius r with bin size d_r (area d_A).

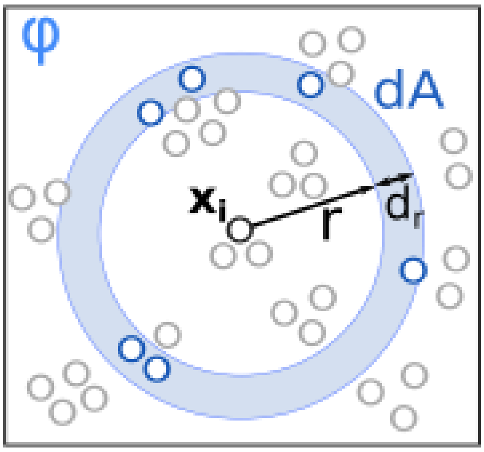

Figure 2.9: Radial distribution analysis

Because of their larger circumference, the coverage area of annular shells get larger as the distance bin being measured increases. In other words, $A_r < A_{r+1}$ where A_r is the area covered by the annular shell starting at distance r. A direct consequence of this is that annular shells at further distances will naturally be prone to containing more points. To counter for this, normalisation is performed based on annular shell area.

The radial distribution analysis function h_{rdf} is as follows:

$$h_{rdf}(k) = \sum_{x_i \in X} \sum_{y_j \in Y \& k d_r \leq d(x_i, y_j) < (k+1) d_r} \frac{A}{d_A n_x n_y}$$

Where:

- *hrdf(k)* is the k-th value of the pair wise histogram.
- X are the points of category X (reference points).
- Y are the points of category Y (target points) .
- d_r is the annular shell width.
- A is the total analysed area.
- n_x and n_y are the number of points of categories x and y respectively. Note that pairwise histograms also need to be calculated for points of the same category. In this situation, category x and category y would be the same.
- d_A is the area of the annular shell being analysed.

Conceptually, this formula calculates the variance from the average density of the target category at incremental distances from points of the reference category.

Reproduction In order to reproduce the distribution of the input exemplar, points are added iteratively whilst matching as closely as possible the corresponding pair correlation histogram data calculated during the analysis stage. Metropolis-Hastings sampling [HLT09] is the most common way to do this. It involves performing a fixed number of point birth-and-death perturbations. A change from the initial arrangement X to the new arrangement *X'* is accepted with probability R, where:

$$R = \frac{f(X')}{f(X}$$

f(X) is the probability density function (PDF) of a given arrangement and is expressed as:

$$f(X) = \prod_{C_{Y_K} \leq C_X} \prod_{x_i \in X} \prod_{y_i \in Y_k} h_{X, Y_k}(d(x_i, y_j))$$

Where:

- C_y and C_x represent categories Y and X, respectively.
- X are all points of category X.

- Y are all points of category Y/
- $h_{X,Y_k}(d(x_i, y_j))$ is the value retrieved from the pairwise histogram of categories X and Y given the distance between points x_i and y_i.

Intuitively, the PDF defines, given a set of points, the aggregate strength of the current distribution.

Because the PDF formula is a product, calculating it for a new layout *X'* with appended/removed point P only involves calculating the PDF for the single reference point P. As a consequence, reproduction can be performed very efficiently. In their work, Emilien and Cani [EC15] are able to perform analysis and reproduction in near real-time.

When using this technique to reproduce a plausible plant distribution, Boudon et al. [BM07] take it one step further by enabling plant crowns to deform based on predefined elasticity parameters. Because the crowns are not constrained to being circular, they can deform to permit facilitate the survival of plants at a lower height.

PREDEFINED ECOSYSTEMS

In their work, Hammes el al. [Ham01] predefine ecosystems along with their preferred environment. These environments are defined in terms of:

- Elevation: All plant species have an upper limit after which temperature or oxygen levels are ill-suited.
- Relative elevation: The local changes in height. Local minimums tend to be valleys and therefore wetter with less illumination. Local maximums, on the other hand, tend to me ridges which are dryer and much more exposed.
- Slope: Gradient has a direct impact on the quality of the soil and therefore the plants which can grow. When slopes get steeper, plants tend to get much smaller as they struggle to get required nutrients from the soil.
- Slope direction: This has a direct effect on sunlight exposure. Southern facing slopes in the northern hemisphere will have a greater exposure to the sun and vice-versa for the southern hemisphere.

All these ecosystems are stored in a database and, when vegetation is to be placed on the terrain, the most suitable ecosystems are chosen based on the terrain properties mentioned above. See figure 2.3.2 for an example landscape generated using this technique.

CONCLUSIONS

Probabilistic Placement permit users to specify only small portions of input data to populate large areas. For the *Radial Distribution Analysis* approach, this input data would be in the form of an input distribution. For the *Predefined Ecosystems* approach, it would be a predefined ecosystem along with its preferred environment. Although this

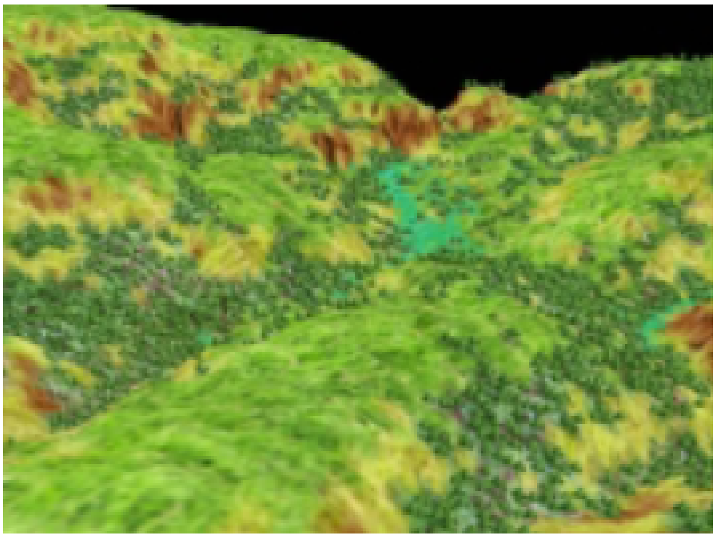

Figure 2.10: Vegetation generated using predefined ecosystems [Ham01]

automation does ease the task for artists, specifying accurate input data is still crucial to produce realistic vegetation. Consequently, although the realism achieved by these methods is generally good, their adaptability is still limited.
Thanks to the use of efficient algorithms, the computational complexity of these methods are often low and real-time updating is achievable.

2.3.3 Simulators

Another approach used to determine vegetation in a given environment is to simulate plants battling for available resources. This approach can be further classified into two subcategories: *Plant Growth Modelling* and *Ecosystem Simulators*.
Plant Growth Modelling techniques go into extensive detail to model the effect of resources on plant growth. The realism is such that it can often be used to model plant growth on earth.
Ecosystem Simulators try to simulate plants competing for available resources when growing. Unlike plant growth modelling which targets botanical realism, these techniques target visual realism. As a consequence, they are more lenient in terms of realism.

PLANT GROWTH MODELLING

These types of simulators attempt to algorithmically reproduce the laws of nature with such precision that they can be used in agronomical sciences and forestry to estimate and maximize crop yield. To achieve this, such simulators go into great detail to model the available resources. For example, work by Soler et al. [SSBR01, SED03] splits single plants into geometrical organs with unique light transmittance and reflectance properties. By doing so, light propagation within the plant can be simulated in order to determine the aggregated photosynthetic potential. This work, along with that of Yan et al. [Yan04], base their simulators on two vital and widely accepted laws of nature:

- *Law of the sum of temperatures*: Plants grow in cycles which vary from days to years depending on the specie. The law of the sum of temperatures states that the frequency of these cycles is proportional to the sum of the daily average of the temperatures.

- *Law of the water use efficiency*: The amount of fresh matter fabricated by a plant is proportional to the water evaporation of the plant. This factor is called the water use efficiency.

Water evaporates during photosynthesis as the plant exchanges water for carbon dioxide. Based on this and the law of water use efficiency outlined above, the amount of fresh matter produced (i.e growth) for a given plant is directly correlated to the amount of photosynthesis performed. Using this, Soler et al. [SSBR01] apply the following formula to calculate the amount of fresh matter, $Q_m(t)$, created by a given plant at time t:

$$Q_m(t) = \sum_{x=1}^{N(t)} \frac{E(x,t)}{R}$$

Where:

- $E(x,t)$ is the potential for matter production of the x-th leaf at the t-th cycle. It is proportional to the incoming radiant energy up to a certain threshold, after which it remains constant.

- R is the hydraulic resistance of the given leaf. This resistance is what limits water evaporation (photosynthesis) and therefore growth. It varies depending in the specie and surface area.

Intuitively, this formula calculates the total available fresh matter, Q_m, that can be produced for an individual plant P at a given time t, by calculating the photosynthesis potential of each individual leaf of P given the current lighting.

Using this, the algorithm iterates through growth cycles with a frequency that is calculated based on the *law of the sum of temperatures* mentioned above. Each growth cycle performs the following two steps:

1. The lighting and therefore photosynthesis potential of each individual leaf of the plant is calculated. This is then used to calculate, as above, the quantity of fresh matter produced.

2. The fresh matter is then distributed to different organs of the plant according to an associated organ strength.

By going into such detail, these simulators produce very realistic simulations of the evolution of plants. For example, to maximize growth, plants are able to grow in direction of the light source (figure 2.3.3).

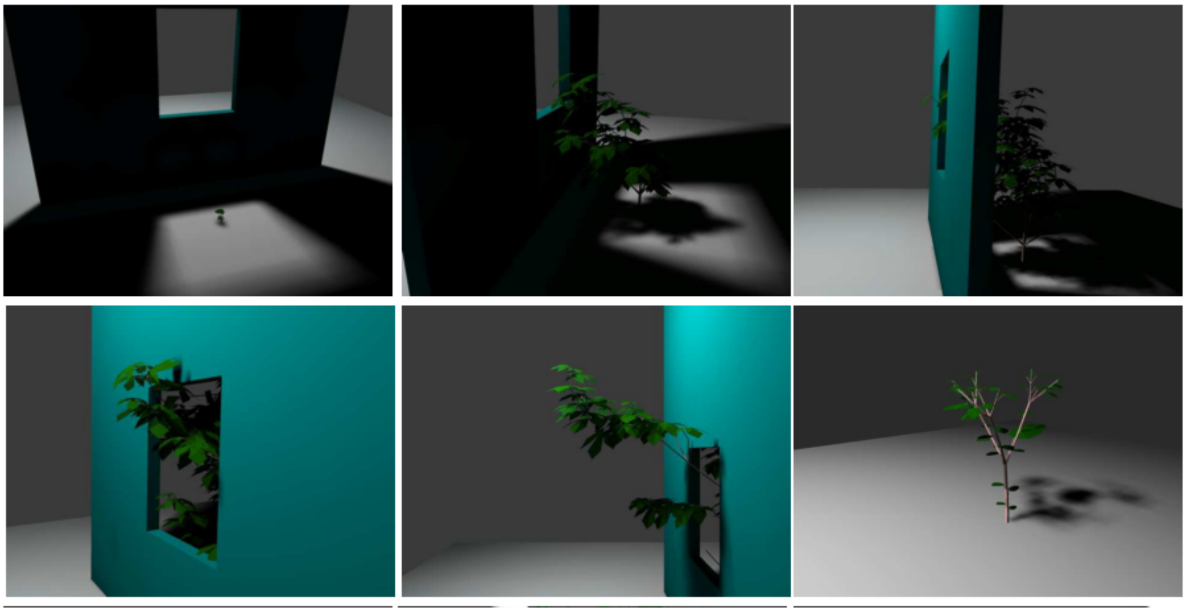

Figure 2.11: Plant growing towards light source [SSBR01]

ECOSYSTEM SIMULATORS

Ecosystem simulators use procedural methods to algorithmically reproduce the competition for resources that occurs in nature during plant growth. In nature, this competition is an extremely complex process and so reproducing it exactly would be infeasible. Instead, a simplified model of this ecological process is implemented. During these simulations, available resources fluctuate and each plants strength is continuously recalculated based on its associated properties. This strength directly affects the plants growth and chance of survival.

Such plant properties include: age; vigor; shade tolerance; humidity requirement and temperature requirements. Amongst others, the resources modelled include: available illumination; available humidity; temperature and slope.

The aim of ecosystem simulators is to determine, given an initial state $\boldsymbol{S_t}$ of the system at time $\boldsymbol{t}$ and a simulation time $\boldsymbol{n}$, the state $\boldsymbol{S_{t+n}}$.

The state of the system represents individual plant instances with associated location and properties.

Lindenmayer systems, commonly referred to as L-systems, use a formal grammar along with a set of production rules to iteratively create larger strings from a starting string called the axiom. Such systems are commonly used to model plants and plant growth [PL90, DC02, BPC$^+$12, PHM93].

An extension to basic L-systems, referred to as open L-systems, adds a communication grammar which permits the set of production rules to behave differently depending on predefined conditions [Pru96]. In their work modelling the growth of struce trees, Berezovskava et al. [BKK97] use different production rules depending on local bud density. This is a simplified representation of buds competing for available light.

By introduction multiset L-Systems, Lane and Przemyslaw [LP02] extend L-systems yet further to model an ecosystem simulator. The production rules for multiset L-systems work in two stages. The first, identical to basic L-Systems, produces a new string given an input string and production rule. The second, splits the resulting string into new sets using a predefined separation symbol. In their work, the different sets represent different plant instances, thus enabling new plants to spawn during the production steps. When building their L-System, Lane and and Przemyslaw [LP02] focus on reproducing three important properties of nature, each distinctly testable to determine the plausibility of the results:

- *Self-thinning*: When plants grow, their resource requirements increase and, as a direct consequence, inter-plant competition for resources increases. Eventually, the competition becomes too intense and resources too scarce leading to more vigorous plants starving smaller plants. At this point, self thinning begins and plant densities decrease.
- *Succession*: Given plant specie A with a fast growth rate and specie B with a slower growth rate but higher shade tolerance. At first, the faster growing specie A will dominate and flourish but, with time, the slower growing but more shade tolerant specie B will flourish and dominate.
- *Propagation*: Plants often propagate in clusters surrounding the seeding plant.

The L-System they implemented contains different production rules to represent the different properties of nature mentioned above. A single simulation and the corresponding output can be seen in figure 2.3.3.

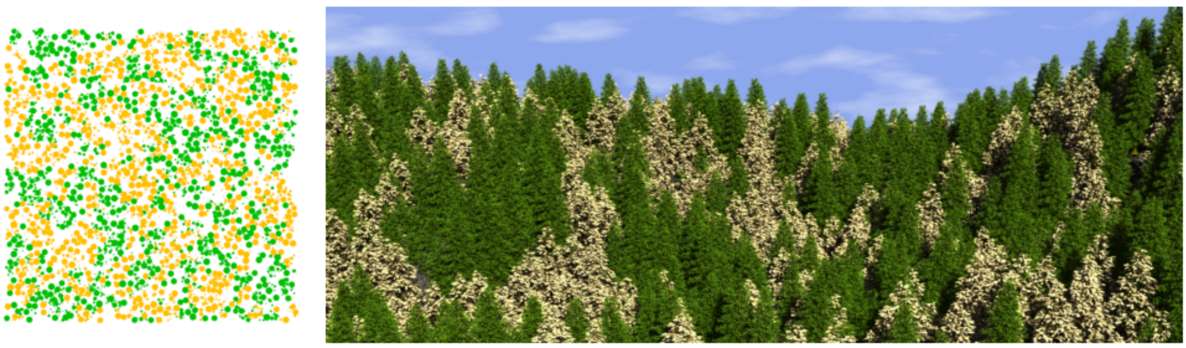

Figure 2.12: Plant placement using an ecosystem simulator modelled by L-Systems [LP02]. *Left:* Result of the simulation where orange circles indicate the positions of poplar trees and green circles the positions of spruce trees. *Right:* Reproduced virtual world where the location of individual plants is deduced from the output of the simulator.

Work by Deussen et al. [DHL+98] also uses L-Systems as the basis for an ecosystem simulator. As an extension to the work by Lane and Przemyslaw [LP02], they introduce the notion of soil humidity and an associated soil per specie humidity preference.

A direct consequence of the automation provided by these ecosystems is that fine control over the final vegetative content is lost. Deussen et al. [DHL+98] overcome this, however, by offering a hybrid approach where the ecosystem simulator is first used to populate the entire terrain and explicit instancing is used thereafter for the detailing .

Another weakness of procedural ecosystems based on L-Systems worth mentioning is that the communication parameter is binary; in the work by Lane et al. [LP02] a plant will be dominated as soon as its radius intersects another larger plant, at which point it will die with a set probability. This probability of death will stay constant and will not increase as this domination increases. Similarly, in the humidity model of Deussen et al. [DHL+98], a plant has a preference for wet or dry areas and there is no notion of a measurable humidity preference range. This could prove problematic to model species which are able to adapt to a multitude of environments with varying resource availability (e.g. grass).

CONCLUSIONS

Probably the main advantage of simulators over other approaches is the level of *automation*. Running simulations is done with easy and requires very little input from the user. Although the adaptability of these methods is also impressive, it is limited by the necessity to configure the properties for individual species. This is especially true for *Plant Growth Modelling* approaches where topological data must be configured. Obtaining topological data often involves real-world analysis of the plants growth cycles.
Computational cost is often high when using simulators. The extent of which is dependant on the level of detail and the number of plants being simulated simultaneously. For example, in the highly detailed simulations of Soler et al. [SSBR01], simulating 45 cycles for a single plant takes approximately 15 minutes.

2.3.4 Conclusions

Which technique (Explicit, Probabilistic or Simulators) to use entirely depends on the requirements of the system. For example, if realism is the key priority then ecosystem simulators able to provide botanical realism would be the most suitable approach. Choosing the technique is therefore all about minimizing the associated compromises. In table 2.3.2 we summarize the pros and cons of the individual techniques based on the following criteria:

- *Automation*: The level of automation the technique provides. That is, how little user input is needed.
- *Realism*: The level of realism with which the technique models real-world ecosystems.
- *Computational efficiency*: The techniques efficiency in terms of computational resource requirements.
- *Adaptability*: How well the technique can adapt to model different scenarios.

	Automation	Realism	Computational Efficiency	Adaptability
Explicit Placement	Poor	Poor	Excellent	Poor
		Probabilistic Placement		
Radial Distribution Analysis	Good	Very Good	Very Good	Fair
Predefined Ecosystems	Good	Fair	Very Good	Poor
		Simulators		
Plant Growth Modelling	Excellent	Excellent	Poor	Fair
Ecosystem Simulators	Excellent	Very Good	Fair	Good

Table 2.1: Pros and cons of individual techniques

Given a set of plant species, available resources and terrain, our system must be able to specify the locations of individual plants. The output must be: visually realistic; easily scalable in order to be able to re-run simulations with different input species; computationally efficient to ensure the effect of user actions appear in close to real-time.
Given these requirements, a hybrid approach is best suited which combines the adaptability and realism of ecosystem simulators with the computational efficiency of probabilistic placement. Computationally expensive ecosystem simulator runs will be performed beforehand in order to acquire the necessary distribution data. This data will then be stored in order for it to be queried at a later stage without having to redo expensive simulations. When placing vegetation in the virtual world, pre-calculated distribution data will be queried and probabilistic instancing used to fill user-defined areas with suited plant species and realistic distributions.

Bibliography

[ACV+14a] C. Andújar, a. Chica, M. a. Vico, S. Moya, and P. Brunet. Inexpensive Reconstruction and Rendering of Realistic Roadside Landscapes. *Computer Graphics Forum*, 00(0):1–18, 2014.

[ACV+14b] C. Andújar, A. Chica, M. a. Vico, S. Moya, and P. Brunet. Inexpensive Reconstruction and Rendering of Realistic Roadside Landscapes. *Computer Graphics Forum*, 33(6):101–117, February 2014.

[aEL99] a.a. Efros and T K Leung. Texture synthesis by non-parametric sampling. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 2:1033 – 1038, 1999.

[AM09] Ratish Agarwal and Dr. Mahesh Motwani. Survey of clustering algorithms for MANET. 16(3):645–678, 2009.

[BA05] F Belhadj and Pierre Audibert. Modeling landscapes with ridges and rivers: bottom up approach. *Proceedings of the 3rd international conference ...*, 1:1–4, 2005.

[BKH97] Doug A Bowman, David Koller, and Larry F Hodges. Travel in Immersive Virtual Environments: An Evaluation Motion Control Techniques of Viewpoint. pages 45–52, 1997.

[BKK97] F. S. Berezovskava, G. P. Karev, and O. S. Kisliuk. A fractal approach to computer-analytical modelling of tree crowns. pages 323–327, 1997.

[BM07] F Boudon and G Le Moguédec. Déformation asymétrique de houppiers pour la génération de représentations paysageres réalistes. *Revue Electronique Francophone d'Informatique*, 1(1):9–19, 2007.

[BPC+12] Frédéric Boudon, Christophe Pradal, Thomas Cokelaer, Przemyslaw Prusinkiewicz, and Christophe Godin. L-py: an L-system simulation framework for modeling plant architecture development based on a dynamic language. *Frontiers in plant science*, 3(May):76, January 2012.

[CHZ00] Jonathan M Cohen, John F Hughes, and Robert C Zeleznik. Harold: A World Made of Drawings. *NPAR 2000*, pages 83–90, 2000.

[CMF98] N. Chiba, K. Muraoka, and K. Fujita. An erosion model based on velocity fields for the visual simulation of mountain scenery. *The Journal of Visualization and Computer Animation*, 9(4):185–194, 1998.

[DB] Gilliam J.P. De Carpenter and Rafael Bidarra. Interactive GPU-based procedural heighfield brushes. 8:55–62.

[DC02] Oliver Deussen and Carsten Colditz. Interactive visualization of complex plant ecosystems. *Proceedings of the conference on Visualization '02 (VIS '02)*, pages 219–226, 2002.

[DGA04] Brett Desbenoit, Eric Galin, and Samir Akkouche. Simulating and modeling lichen growth. 23(3), 2004.

[DGA06] Brett Desbenoit, Eric Galin, and Samir Akkouche. Modeling Autumn Sceneries. 2006.

[DGGK11a] E Derzapf, B Ganster, M Guthe, and R Klein. River Networks for Instant Procedural Planets. 30(7), 2011.

[DGGK11b] E. Derzapf, Björn Ganster, M. Guthe, and Reinhard Klein. River Networks for Instant Procedural Planets. *Computer Graphics Forum*, 30(7):2031–2040, 2011.

[DHL+98] Oliver Deussen, Pat Hanrahan, Bernd Lintermann, Radomir Měch, Matt Pharr, and Przemyslaw Prusinkiewicz. Realistic Modeling and Rendering of Plant Ecosystems. *Conference on Computer Graphics and Interactive Techniques*, pages 275—-286, 1998.

[DP10] Jonathon Doran and Ian Parberry. Controlled Procedural Terrain Generation Using Software Agents. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(2):111–119, June 2010.

[DSS93] Rudy P Darken, John L Sibert, and Computer Science. A Toolset for Navigation in Virtual Environments. pages 157–165, 1993.

[EC15] Arnaud Emilien and Marie-Paule Cani. WorldBrush: Interactive Example-based Synthesis of Procedural Virtual Worlds. *Siggraph '15, TO BE PUBLISHED*, 2015.

[Emi14] Arnaud Emilien. Création interactive de monde virtuels. 2014.

[EPCV14] Arnaud Emilien, Pierre Poulin, Marie-Paule Cani, and Ulysse Vimont. Interactive Procedural Modelling of Coherent Waterfall Scenes. *Computer Graphics Forum*, pages n/a–n/a, 2014.

[Fai88] Plant Models Faithful. ˜l˜ Computer Graphics, Volume 22, Number 4, August 1988. 22(4):151–158, 1988.

[FMWu00] Sven Fuhrmann, Alan M Maceachren, and Westfaelische Wilhelms-universitaet. Testing on Usability: Navigation in Desktop GeoVirtual Environments. 2000.

[FZS+08] Thierry Fourcaud, Xiaopeng Zhang, Alexia Stokes, Hans Lambers, and Christian Körner. Plant growth modelling and applications: the increasing importance of plant architecture in growth models. *Annals of botany*, 101(8):1053–63, May 2008.

[Gal05] E Galin. Real-time Rendering of Realistic-looking Grass. 2005.

[GFJ+11] Y. Guo, T. Fourcaud, M. Jaeger, X. Zhang, and B. Li. Plant growth and architectural modelling and its applications. *Annals of Botany*, 107(5):723–727, April 2011.

[GGG+13] Jean-David Génevaux, Éric Galin, Eric Guérin, Adrien Peytavie, and Bedich Beneš. Terrain generation using procedural models based on hydrology. *ACM Transactions on Graphics*, 32(4):1, 2013.

[GMN14] James Gain, Patrick Marais, and Rudolph Neeser. City Sketching. *WSCG 2014*, pages 1–10, 2014.

[GMSe09] James Gain, Patrick Marais, and Wolfgang Straß er. Terrain sketching. *Proceedings of the 2009 symposium on Interactive 3D graphics and games - I3D '09*, 1(212):1–8, 2009.

[GPGB11] Eric Galin, Adrien Peytavie, Eric Guérin, and Bedich Beneš. Authoring Hierarchical Road Networks. *Computer Graphics Forum*, 30(7):2021–2030, September 2011.

[HA36] Johan Hammes and South Africa. Modeling of ecosystems as a data source for real-time terrain rendering. 1936.

[Ham01] Johan Hammes. Modeling of Ecosystems as a Data Source for Real-Time Terrain Rendering. *Framework*, pages 98–111, 2001.

[HLT09] T Hurtut, PE Landes, and J Thollot. Appearance-guided synthesis of element arrangements by example. *Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering*, pages 51–60, 2009.

[IMT04] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A Sketching Interface for 3D Freeform Design. *SIGGRAPH*, pages 1–8, 2004.

[Jae] Marc Jaeger. Philippe de Reffye (Cirad) & Marc Jaeger (Cirad).

[JR92] M Jaeger and P H D E Reffye. Basic concepts of computer simulation of plant growth. 17(3):275–291, 1992.

[KD01] N M Kapolka and D J Dollhopf. Effect of slope gradient and plant growth on soil loss on reconstructed steep slopes. *International Journal of Surface Mining*, 15(2):86–89, 2001.

[KD10] N.M. Kapolka and D.J. Dollhopf. Effect of Slope Gradient and Plant Growth on Soil Loss on Reconstructed Steep Slopes. *International Journal of Surface Mining, Reclamation and Environment*, 15(2):86–99, August 2010.

[KM07] George Kelly and Hugh McCabe. Citygen: An interactive system for procedural city generation. *Fifth International Conference on Game Design and Technology*, pages 8–16, 2007.

[KMN88] Alex D. Kelley, Michael C. Malin, and Gregory M. Nielson. Terrain simulation using a model of stream erosion. *ACM SIGGRAPH Computer Graphics*, 22(4):263–268, 1988.

[Lau10a] Eddie Lau. Visually appealing water flow over a terrain. 2010.

[Lau10b] Eddie Lau. Visually appealing water flow over a terrain. 2010.

[Lew99a] Philip Lewis. Three-dimensional plant modelling for remote sensing simulation studies using the Botanical Plant Modelling System. 1999.

[Lew99b] Philip Lewis. Three-dimensional plant modelling for remote sensing simulation studies using the Botanical Plant Modelling System. 1999.

[Lew99c] Philip Lewis. Three-dimensional plant modelling for remote sensings studies using Botanical Plant Modeling System, 1999.

[LLX+01] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics*, 20(3):127–150, July 2001.

[LP] Brendan Lane and Przemyslaw Prusinkiewicz. Generating Spatial Distributions for Multilevel Models of Plant Communities.

[LP02] Brendan Lane and Przemyslaw Prusinkiewicz. Generating Spatial Distributions for Multilevel Models of Plant Communities. *Interface*, 2002:69–80, 2002.

[Mae06] Marcelo M Maes. Efficient Animation of Water Flow on Irregular Terrains. pages 107–115, 2006.

[MD07] Xing Mei and Philippe Decaudin. Fast Hydraulic Erosion Simulation and Visualization on GPU. *Pacific Graphics*, 2007.

[MDH] Xing Mei, Philippe Decaudin, and Bao-gang Hu. Fast Hydraulic Erosion Simulation and Visualization on GPU.

[MFC06] Marcelo M Maes, Tadahiro Fujimoto, and Norishige Chiba. Efficient animation of water flow on irregular terrains. *GRAPHITE - International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*, 1(212):107–115, 2006.

[MGG+10] N. Maréchal, E. Guérin, E. Galin, S. Mérillou, and N. Mérillou. Heat Transfer Simulation for Modeling Realistic Winter Sceneries. *Computer Graphics Forum*, 29(2):449–458, May 2010.

[MKM89] F. K. Musgrave, C. E. Kolb, and R. S. Mace. The synthesis and rendering of eroded fractal terrains. *ACM SIGGRAPH Computer Graphics*, 23(3):41–50, 1989.

[Muh01] Andreas Muhar. Three-dimensional modelling and visualisation of vegetation for landscape simulation. *Landscape and Urban Planning*, 54(1-4):5–17, 2001.

[Nik07] Karl J Niklas. Maximum plant height and the biophysical factors that limit it. pages 433–440, 2007.

[OA00] Communications Of and T H E Acm. July 2000/Vol. 43, No. 7 COMMUNICATIONS OF THE ACM. 43(7), 2000.

[OG12] AC Öztireli and Markus Gross. Analysis and synthesis of point distributions based on pair correlation. *ACM Transactions on Graphics (TOG)*, 31(6):170:1–170:10, 2012.

[PBBW95] Randy Pausch, Tommy Burnette, Dan Brockway, and Michael E Weiblen. Navigation and Locomotion in Virtual Worlds via Flight into Hand-Held Miniatures. pages 399–400, 1995.

[PBN+09] C. Pradal, F. Boudon, C. Nouguier, J. Chopard, and C. Godin. PlantGL: A Python-based geometric library for 3D plant modelling at different scales. *Graphical Models*, 71(1):1–21, January 2009.

[PDkFG08] Christophe Pradal, Samuel Dufour-kowalski, Christian Fournier, and Christophe Godin. OpenAlea: a visual programming and component-based software platform for plant modelling. pages 751–760, 2008.

[PHM93] Przemyslaw Prusinkiewicz, Mark Hammel, and Eric Mjolsness. Animation of Plant Development. *SIGGRAPH '93 Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, 93:351–360, 1993.

[PL90] ALP Prusinkiewicz and A Lindenmayer. *The Algorithmic Beauty of Plants*. 1990.

[PM01] Yoav I. H. Parish and Pascal Müller. Procedural Modeling of Cities. *28th annual conference on Computer graphics and interactive techniques*, (August):301–308, 2001.

[PMV+12] C. E. Timothy Paine, Toby R. Marthews, Deborah R. Vogt, Drew Purves, Mark Rees, Andy Hector, and Lindsay a. Turnbull. How to fit nonlinear plant growth models and calculate growth rates: an update for ecologists. *Methods in Ecology and Evolution*, 3(2):245–256, April 2012.

[PPH93] P Pmsinkiewicz, Prusinkiewicz\ Prusinkiewicz, and Mark Hammel. A fractal model of mountains and rivers. *Graphics Interface*, (May):174–180, 1993.

[Pru96] Przemyslaw Prusinkiewicz. Visual Models of Plants Interacting with Their Environment. *SIGGRAPH '96 Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1:397–410, 1996.

[Pru98] Przemyslaw Prusinkiewicz. Modeling of spatial structure and development of plants. *Scientia Horticulturae*, 74(1-2):113–149, 1998.

[RG] Stefan Roettger and Computer Graphics Group. Ndvi-based vegetation rendering.

[SAPa] M. Smith, R. Allen, and L. Pereira. Revised FAO Methodology for crop-water requirements. pages 51–58.

[SAPb] M Smith, R Allen, and L Pereira. Revised FAO methodology for crop water requirements. pages 51–58.

[SED03] Cyril Soler, F R Ed, and Philippe Dereffye. An Efficient Instantiation Algorithm for Simulating Radiant Energy Transfer in Plant Models. 22(2):204–233, 2003.

[SKG+09] Ruben M Smelik, Klaas Jan De Kraker, Saskia A Groenewegen, The Hague, Tim Tutenel, and Rafael Bidarra. A Survey of Procedural Methods for Terrain Modelling . 2009.

[SM08] Karan Singh and James Mccrae. Sketch-Based Path Design. *Proceedings of the Graphics Interface Canadian Inform. Process. Soc.*, pages 95–102, 2008.

[SS98] Dietrich Stoyan and Helga Stoyan. Non-Homogeneous Gibbs Process Models for Forestry A Case Study. 40:521–532, 1998.

[SSBR01] Cyril Soler, FX Sillion, F Blaise, and Philippe De Reffye. A physiological plant growth simulation engine based on accurate radiant energy transfer. Technical report, 2001.

[ST10] Ruben Smelik and Tim Tutenel. Integrating procedural generation and manual editing of virtual worlds. *Proceedings of the 2010 ...*, 2:1–8, 2010.

[STDB11] R. M. Smelik, T. Tutenel, K. J. De Kraker, and R. Bidarra. A declarative approach to procedural modeling of virtual worlds. *Computers and Graphics (Pergamon)*, 35(2):352–363, April 2011.

[STdKB10] Ruben M. Smelik, Tim Tutenel, Klaas Jan de Kraker, and Rafael Bidarra. Declarative Terrain Modeling for Military Training Games. *International Journal of Computer Games Technology*, 2010:1–11, 2010.

[STKB10] R M Smelik, T Tutenel, K J De Kraker, and R Bidarra. Interactive Creation of Virtual Worlds Using Procedural Sketching. *Proceedings of Eurographics 2010: Short Papers. Eurographics Association*, pages 1–4, 2010.

[SWF12a] Wei Sun, Junhui Wang, and Yixin Fang. Regularized k-means clustering of high-dimensional data and its asymptotic consistency. *Electronic Journal of Statistics*, 6(April 2011):148–167, 2012.

[SWF12b] Wei Sun, Junhui Wang, and Yixin Fang. Regularized k-means clustering of high-dimensional data and its asymptotic consistency. *Electronic Journal of Statistics*, 6(April 2011):148–167, 2012.

[Teo08] Soon Tee Teoh. River and coastal action in automatic terrain generation. *CGVR - Computer Graphics and Virtual Reality*, (1):3–9, 2008.

[vBBK08] Ondej Št'Ava, Bedich Beneš, Matthew Brisbin, and Jaroslav Kivánek. Interactive terrain modeling using hydraulic erosion. *EG CA - EuroGraphics Symposium on Computer Animation*, 2008.

[VSLD13] Kenneth Vanhoey, Basile Sauvage, Frédéric Larue, and Jean-Michel Dischler. On-the-fly multi-scale infinite texturing from example. *ACM Transactions on Graphics*, 32(6):1–10, November 2013.

[WLKT09] Li-yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. State of the Art in Example-based Texture Synthesis. *In proceedings of Eurographics 09*, (2):1–25, 2009.

[Xu05] Rui Xu. Survey of Clustering Algorithms. 16(3):645–678, 2005.

[Yan04] H.-P. Yan. A Dynamic, Architectural Plant Model Simulating Resource-dependent Growth. *Annals of Botany*, 93(5):591–602, March 2004.

[ZDL+11] Guo Xin Zhang, Song Pei Du, Yu Kun Lai, Tianyun Ni, and Shi Min Hu. Sketch guided solid texturing. *Graphical Models*, 73(3):59–73, May 2011.

[ZHHV10] Robert C Zeleznik, Kenneth P Herndon, John F Hughes, and Scientific Visualization. SKETCH: An Interface for Sketching 3D Scenes. 1910.