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

List of Figures

List of Tables

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 [? ? ?], modelling plants [? ? ? ?], generating terrains [? ? ?], generating river networks [? ?] and generating city landscapes [? ? ?] (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 [?] to sketch-based recognition algorithms [?].

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.

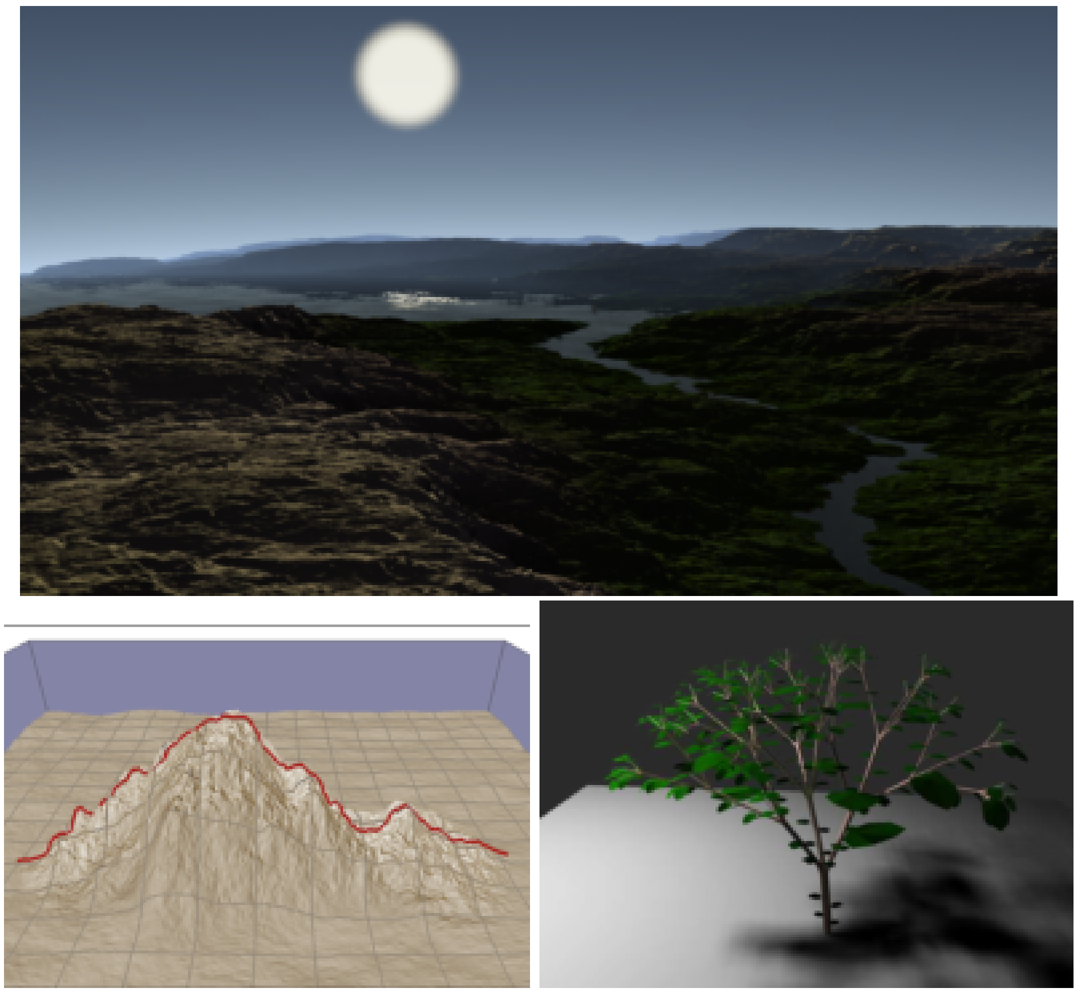

Figure 1.1: Example of procedurally generated content. From top to bottom, left to right: Procedurally generated river stream [?], procedurally generated terrain through sketching [?], procedurally generated plant [?]

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 Rivers & Streams

In this section will be reviewed the various techniques that are used to place rivers and streams on a terrain. We will split the review material into the following categories, each with a dedicated section: *Classification-based*, *Simulation-based*, *Heuristic-based*, *Fractal-based* and *Explicit*.
Classification-based methods use pre-classified data based on real-world analysis to determine the most suited water network given a set of user-defined or terrain-defined constraints.
Simulation-based techniques attempt to simulate natural phenomena such as gravity to determine the water networks on a given terrain.
Heuristic-based techniques use algorithms based on real-world observation in an attempt to produce a plausible river network on the terrain.
Fractal-based techniques use recursive algorithms in their attempt to generate plausible river networks.
Explicit techniques require the user to specify in great detail the path the river should follow on the terrain.

The various techniques will be critiqued based on: the realism of the generated river networks, their computational cost and the level of automation they provide.

2.1.1 Classification-based

Classification-based methods use real-world analysis of river networks to determine, based on terrain parameters (slope, soil type, flow intensity, etc.), the types of rivers best suited (stream, cascade, rapid, etc.) to given landscapes.

Emilien et al [**?**] use classification-based techniques in their research focused 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 parts of the water network in continuous contact with the terrain. *Free-fall* segments are parts which break terrain contact (i.e. waterfall). Lastly, *pool* segments represent the water-basin formed where free-fall segments meet the terrain.
Given a terrain, the user models running water and pool segments by defining control points and free-fall segments by defining a 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 adapt accordingly. The only constraint is that the path must continuously flow downhill. Based on this input, the system calculates plausible water flow intensities which, if required, can be overridden by the user for finer control.
The slope and water flow intensity requirements are then used as input to the waterfall classification (figure 2.1.1) in order to determine realistic waterfall scenes to generate.

By automatically generating plausible waterfall scenes based on trajectory input from the user, the technique strikes a good balance between automation and user control. A consequence of this, however, is that the resulting realism is heavily user-dependent.

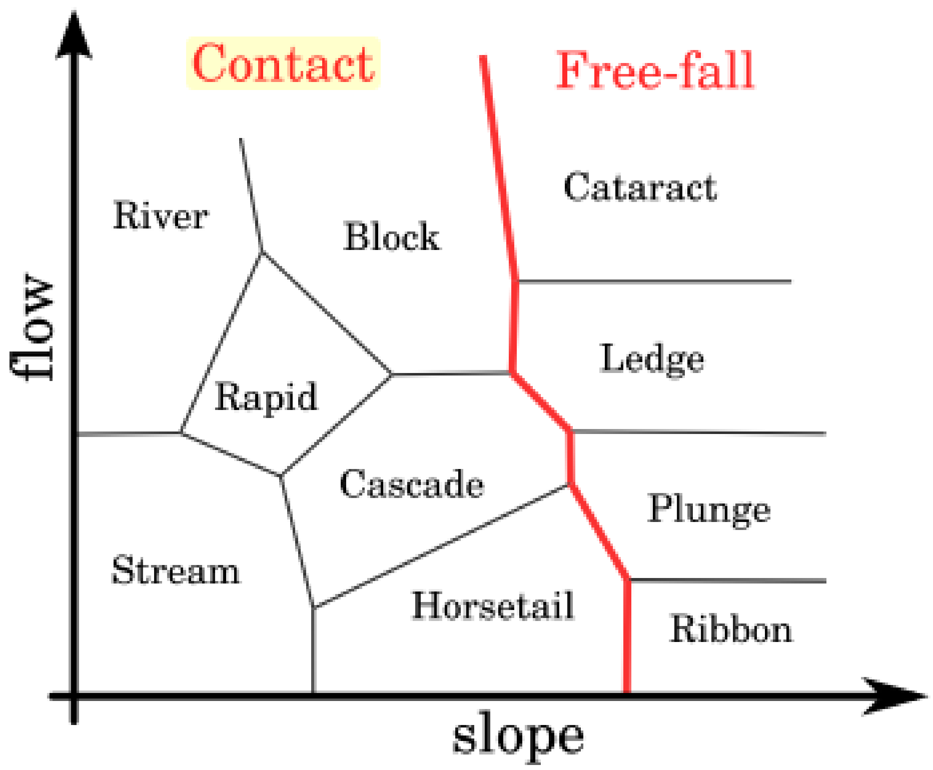

Figure 2.1: Waterfall classifications [?]

In terms of computational complexity, the work by Emilien et al [?] is able to produce complex waterfall scenes in near real-time.

2.1.2 Simulation-based

Simulation-based techniques attempt to reproduce real-world phenomena to generate realistic river networks on given terrains. These simulations vary greatly in the level of detail they attempt to reproduce and can be split into the following sub-categories, each with a dedicated section below: *Gravitation-simulation Erosion-simulation* and *Rainfall simulation*

2.1.2.1 Gravity-simulation

Gravity simulating techniques attempt to determine the path water will take on a terrain by algorithmically replicating the effects of gravity.

In order to generate plausible rivers, Belhadg et Audibert [?] simulate the effect gravity has on water particles placed on the peaks of pre-generated ridges. To create the ridges, particle pairs are first placed at random locations on the terrain. These particle pairs are then randomly assigned a horizontal axis from which they iteratively distance themselves in opposite directions. At each iteration a new vertex is placed and its height decreased from the previous vertex based on a Gaussian distribution. To create the river networks, river particles are placed on the top of these generated ridges and a physical simulation which takes into consideration particle velocity, particle mass and surface friction is used to model the motion of these particles on the terrain. The path followed by these particles is tracked and, when two paths intersect, their particle velocity and mass are combined. When all particles have stopped moving the simulation is deemed balanced and all particle paths which do not lead to terrain extremities discarded. The remaining particle paths are kept and form the core river network.

Similarly, in the work by Soon Tee [**?**], water is placed at specific locations on the terrain either by the user or whilst simulating rainfall. To determine the course the placed water takes on the terrain, water is iteratively evacuated into the surrounding cell with lowest elevation. This continues until a local minima or terrain extremities is reached.

In their work on modelling the effects of hydraulic erosion, t'Ava et al. [**?**] determine the course user-placed water takes on the terrain using a hydrostatic pipe-model simulation. In order to do so, the terrain is split into equal-sized (configurable) columns and the simulation iteratively evacuates water from source to surrounding destination columns based on column elevations, fluid density and gravitational acceleration.

These techniques can produce very plausible results but have the downside of being dependent on the base terrain as their height-field must cater for river networks in the first place. This is not the case, however, for the work by t'Ava et al. [**?**] fow which the gravitation simulation is used as a feedback loop to model terrain erosion. The performance of these methods depend heavily on the level of detail of the underlying water flow simulation. t'Ava et al. [**?**] succeed in generating the water flow in real-time by optimizing their algorithms to use the heavily parallel architecture of GPUs.

2.1.2.2 Erosion-simulation

Erosion-based simulations attempt to produce realistic terrains by modelling the effects of erosion. Erosion results from exogenic processes (water flow, wind, temperature) and is characterised by the removal of soil and rock from one location on earth's surface to be redeposited on another. Earth's landscape is a direct consequence of erosion and reproducing this phenomena accurately is core to procedurally generating accurate landscapes. Both Kelly et al. [**?**] and t'Ava et al. [**?**] attempt to produce plausible terrains by modelling these effects.

In the work by Kelley 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 three-dimensional space. The main trunk stream specifies the path which the highest order water stream should follow on the resulting terrain. Given this terrain outline and the position of the initial main trunk stream, the system iteratively increments the number of nodes which form the main trunk in order to add streams to the network. The number of new nodes added depends on the drainage area (surface area that a stream needs to channel) and the soil type as more resistant soil materials (e.g. stone) will be less influenced by water erosion than weaker ones (e.g. clay).

t'Ava et al. [?] are able to simulate the effects of hydraulic erosion on a terrain in real-time by using the massively parallel architectures of GPUs. Virtual pipettes are used by the user to drop water at required locations on the terrain and a gravitation simulation mentioned previously (??) is used to determine the initial water course on the terrain. Whilst the water is being routed through the terrain, the effects of *force-based* and *dissolution-based* erosion are simulated. *Force-based* erosion is a direct consequence of the the force of the water on the terrain surface (figure 2.1.2.2). 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 2.1.2.2).

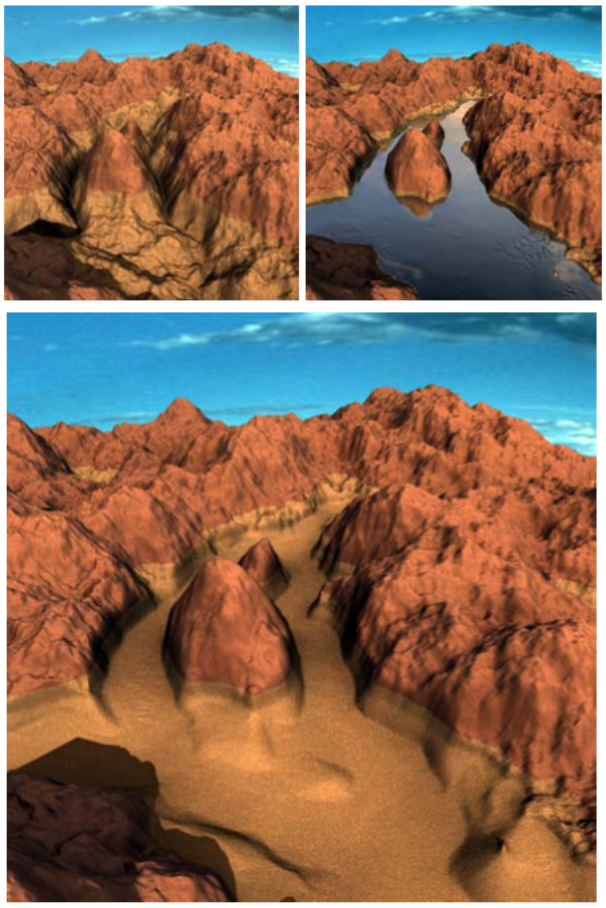

Figure 2.2: Simulation of dissolution-based erosion erosion caused by water movement[?]

Whether modelling erosion indirectly like in the work by Kelley et al. [?] which builds the terrain around models of erosion or directly like the work by t'Ava et al. [?] which simulates the effect of erosion in real-time, both succeed in producing plausible terrains with integrated river networks. Fine-control over the resulting terrain, however, is limited in the work by Kelley et al. [?] due to extensive automation. This is overcome

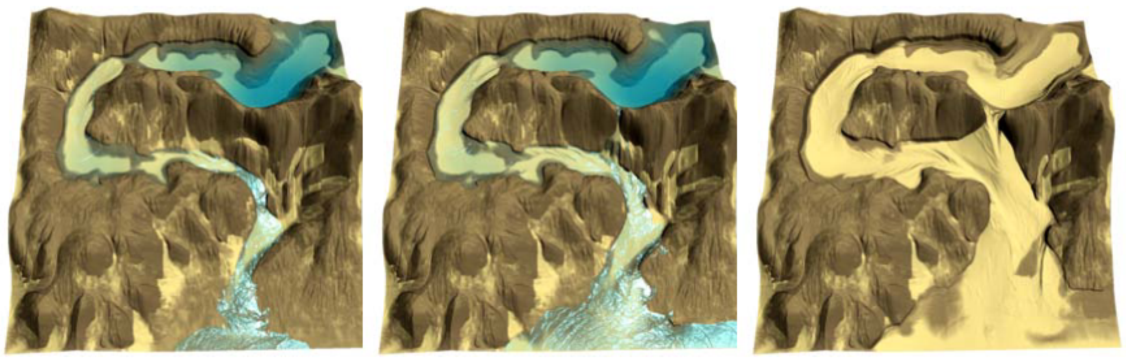

Figure 2.3: Simulation of the effect of force-based erosion caused by running water [?]

in the work by [?] et al. by permitting the user to both place water using a virtual pipette and remodel the terrain relief. In terms of computational cost, t'Ava et al. [?] are able to reproduce the effects of erosion in real-time.

2.1.2.3 Rainfall simulations

In order to determine where on the terrain rivers will appear, work by Soon Tee [?] performs a rainfall simulation to determine both the location and quantity of water at different points on the terrain followed by a gravitation simulation (mentioned above) to determine the course of the water on the terrain. The rainfall simulation 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 a configured velocity. When contact is made with points on the terrain, water is dropped on the corresponding cell. The amount of water dropped increases with altitude and zeroes out when all available rainfall is depleted.

Simulating rainfall in order to determine where water will fall on the terrain and therefore where river networks will form is an original approach and one that successfully generates visually plausible terrains. Requiring only wind direction, wind velocity and maximum rainfall from the user, the system provides a good level of automation. Determining the influence these inputs have on the resulting scene could be unintuitive, however, and require a "trial-and-error" approach. Their algorithm creates the terrain along with the river networks in O(n) time, n representing the number of cells on the terrain.

2.1.3 Heuristic-based

Heuristic approaches attempt to build river and stream networks on terrains by algorithmically reproducing key characteristics based on real-world observations.

Derzapf et al. [?] use such methods in their work based on procedurally generating virtual planets in real-time. To do so, only a very basic mesh-representation of the terrain is generated at first and detailed content is generated on demand as the user navigates

through the virtual world. This method of adaptive rendering permits memory usage to me manageable whilst not compromising on realism. To ensure updates are performed in real-time, their algorithms are designed to make use of the massively parallel architecture of GPUs.
To initialise the base representation of the planets, the system first creates the base mesh with all vertices representing the sea. The system then randomly assigns a certain number of these vertices to act as seed continent vertices to spread until a user-configured land-to-water ratio is reached. To place rivers, similarly to the work by [**?**] et al., they first locate continental points which are on coastal edges to act as river mouths. When such a vertices are found, adjacent continental vertices are iteratively selected pseudo-randomly and connected in order to form the river network.
To assign ground altitudes to connected river vertices the system employs the following formula, starting from the river mouth:

$$a_v = a_u + e_a l_e \xi, e_a = \tfrac{a_{maxriver}}{l_r}$$

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.
- $a_{maxriver}$ 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:

$$w_v = a_v + e_w l_e, e_w = \tfrac{\epsilon_{river}}{l_{cr}}$$

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 user-configured maximum river depth.
- l_{cr} is the distance from the current vertex to the river spring.

All randomness in these algorithms depend on a configured seed value enabling the virtual world to be easily reproducible.

This heuristic approach offers an extensive level of automation and, as a result, fine control over the resulting scene is lost. Rather than generating virtual worlds fitting specific user requirements, it is more suited to generating plausible virtual worlds which fit loose constraints (e.g. maximum river altitude, maximum water depth, river stream must flow downhill, etc.).

2.1.4 Fractal-based

Another technique employed to produce river streams is by employing fractal-based algorithms. Such methods use recursive splitting and string rewriting to determine plausible river networks.

In their work, Pmsinkiewicz et al. use a fractal-based technique based on midpoint-displacement to procedurally generate plausible rivers on a terrain. Midpoint-displacement is most commonly used for procedurally generating realistic terrain height-maps and works follows follows: Given a starting triangle representing a terrain A, midpoint-displacement iteratively subdivides A it 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 2.1.4 for an example of a single iteration of the process.

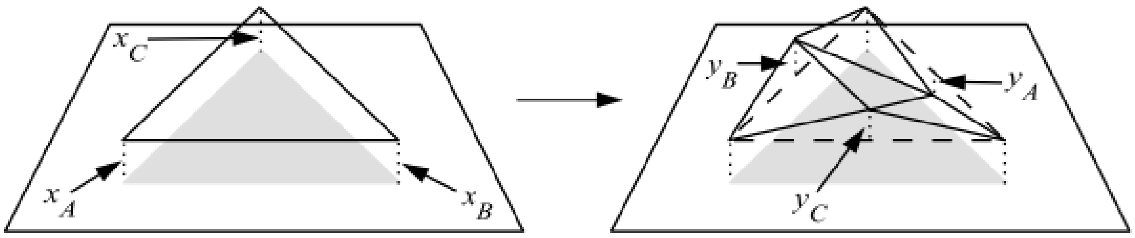

Figure 2.4: A single iteration of midpoint displacement for the creation of mountains [?]. New vertices y_A, y_B and y_C are created and shifted vertically by a random offset

To adapt this method to the generation of rivers on the terrain, rather than vertically displace newly formed triangle vertices, there edges are labelled as *entry*, *exit* or *neutral* (figure **??**). 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".

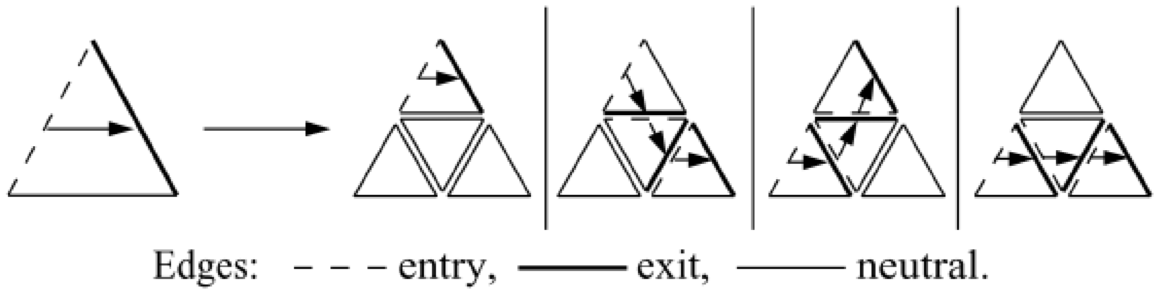

Figure 2.5: Single production of midpoint displacement adapted to river generation [?]. Given the initial triangle, four valid split scenarios.

One difficulty with this technique is to ensure two adjacent triangles are coherent once split. That is, that the exit edge of one coincides with the entry edge of the other. To solve this, the location of edge vertices are used as the key to a random number generating hash table which, based on its output number, determines the segment that will be crossed by the river, if any.

In their work, Gnevaux et al. [?] use fractal-based string rewriting to produce the river networks on the terrain. Once initial nodes have been selected to act as the river mouth, rewriting grammar is used to perform river node expansion. Configured values of $\rho_a, \rho_s and \rho_c$ influence the probability of selecting productions favouring asymmetric branching, symmetric branching and continuation without branching, respectively. The position for the new node is then selected based on 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.

Both these techniques are successful in generating realistic river networks on terrains. The user, however, is limited in the amount of control he has over the resulting rivers. In the work by Gnevaux et al. [?], for example, this is limited to specify the preferred branching behaviours of the rivers. In terms of performance, Gnevaux et al. [?] are able to produce terrains of several hundred square kilometres in a matter of seconds.

2.1.5 Explicit

Explicit techniques use explicit input from the user to determine locations and properties of the river networks to generate.

Flood-filling is such a technique and is used in the work by Soon Tee [**?**] to permit users to place water reserves (e.g. sea, lakes, etc.) by clicking a single point on the terrain. This point which will act as the seed point for the water surface and will propagate iteratively to surrounding points at lower heights until all such points have been depleted.

Smelik et al. also use explicit techniques to create an interactive system permitting users to model a complete virtual world with content ranging from rural features (mountains, rivers, etc.) to man-made ones (buildings, road networks). 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, including rivers. To do so, similarly to the interface provided by Emilien et al. [**?**], the user sketches vector lines outlining the core path of the river and, based on this, a suitable course is plotted through the landscape. Other terrain features to which the river takes precedence adapt accordingly. For example, if the river is plotted to pass through a forest, trees on the rived bed and bank will be removed automatically.

Rather than placing vector-lines, the work by Soon Tee [**?**] and t'Ava et al. [**?**] permits users to click single points on the terrain which will act as the water source. The system then automatically generates a plausible path for the water down slope of the terrain.

As these methods provide very little automation in terms of guaranteeing consistency in the scene, the resulting realism is very much user-dependent. Real-time action-reaction feedback is essential with explicit methods and so the majority of the tools run in real-time.

2.2 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.

2.2.1 Explicit Placement

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

Arnaud et al. [**?**] 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 2.2.1)

Similarly, Deussen et al. [**?**] 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. [**?**] 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.

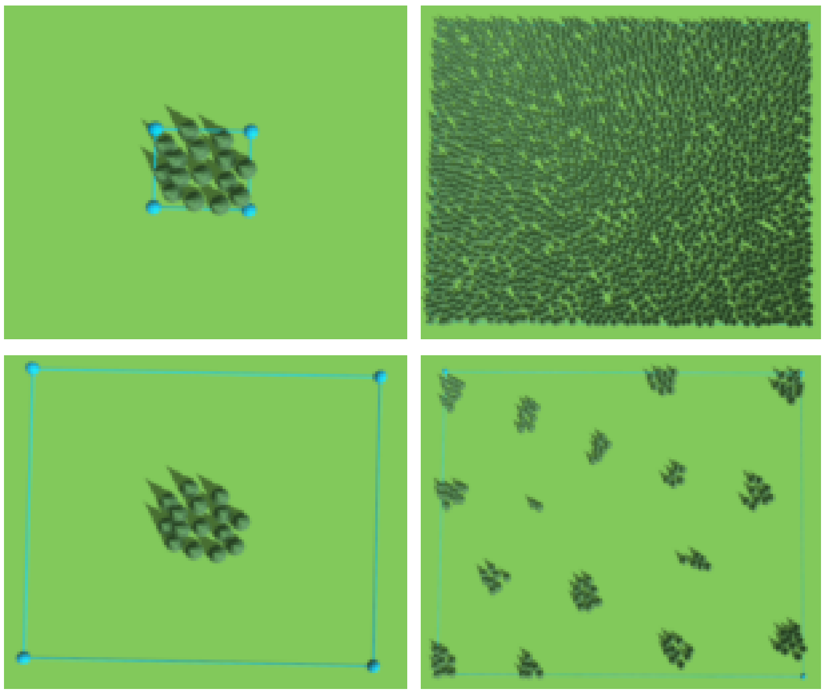
Figure 2.6: Using explicit placement as input examplars for reproduction [?]

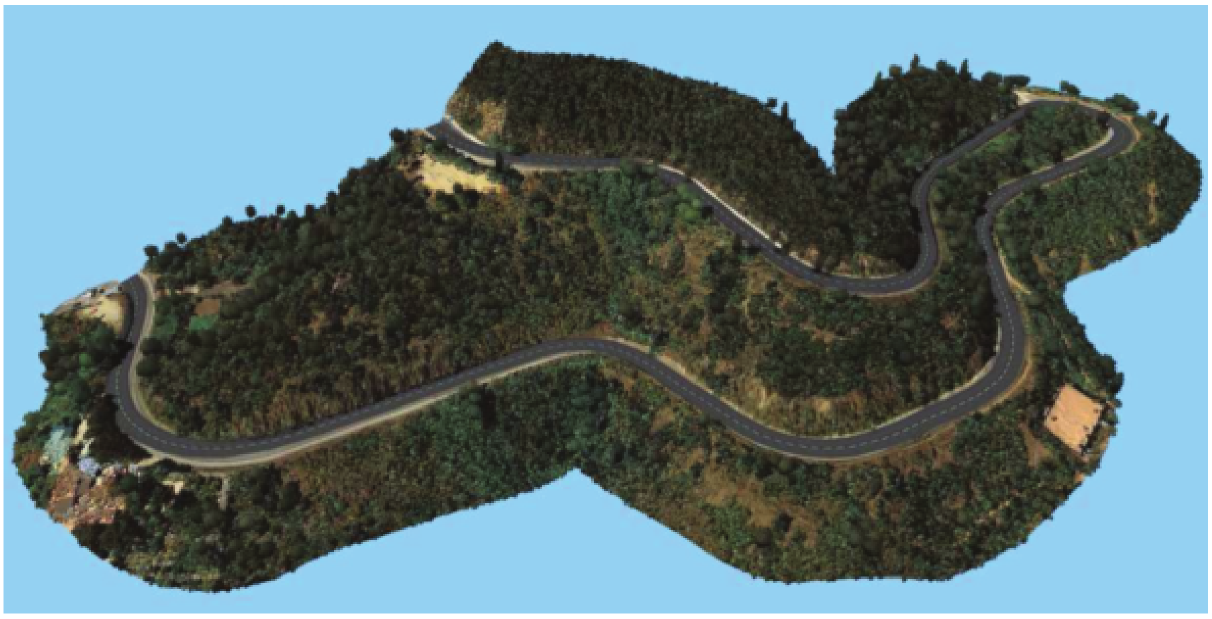
Figure 2.7: Reconstructed roadside vegetation using orthophotos [?]

2.2.1.1 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.

2.2.2 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.

2.2.2.1 RADIAL DISTRIBUTION ANALYSIS

Work by Emilien et al. [**?**], Boudon et al. [**?**] and Lane et al. [**?**] 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 2.2.2.1)

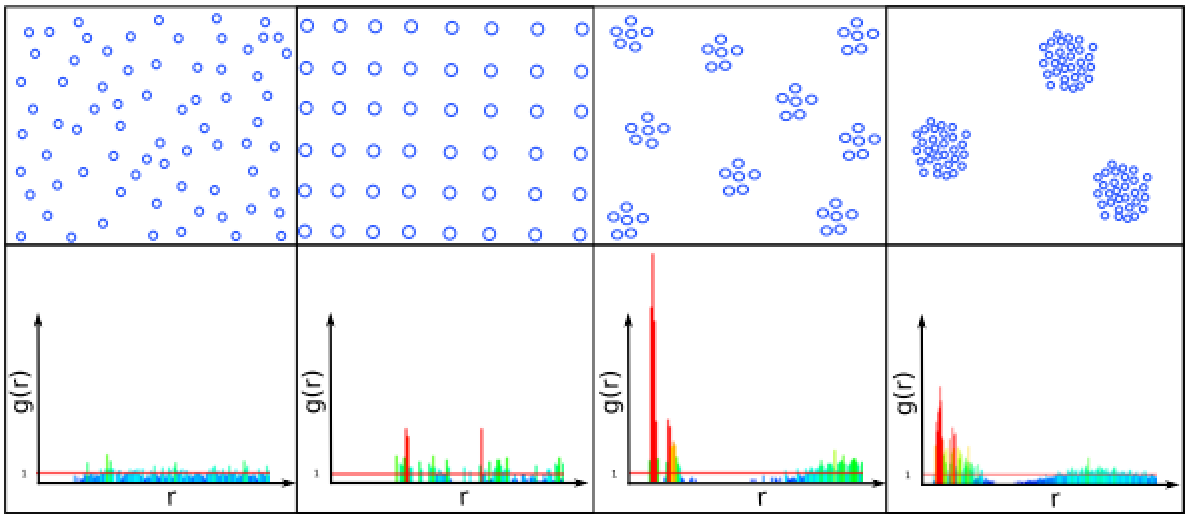

Figure 2.8: Point distributions with associated pair correlation histogram [**?**]

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 2.2.2.1, for example, are being measured the points that lie within the annular shell of radius r with bin size d_r (area d_A).

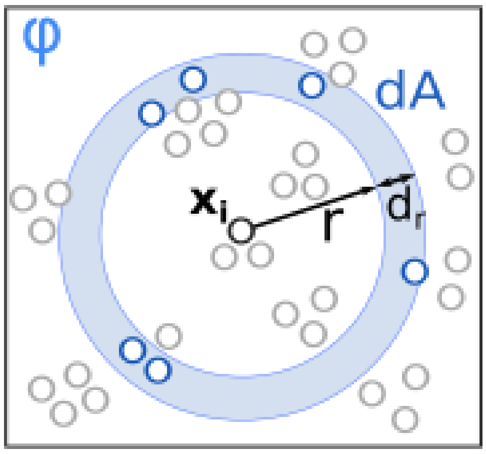

Figure 2.9: Radial distribution analysis

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:

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

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 [**?**] 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:

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

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 [**?**] are able to perform analysis and reproduction in near real-time.

When using this technique to reproduce a plausible plant distribution, Boudon et al. [**?**] 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.

2.2.2.2 PREDEFINED ECOSYSTEMS

In their work, Hammes el al. [**?**] 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 2.2.2.2 for an example landscape generated using this technique.

2.2.2.3 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

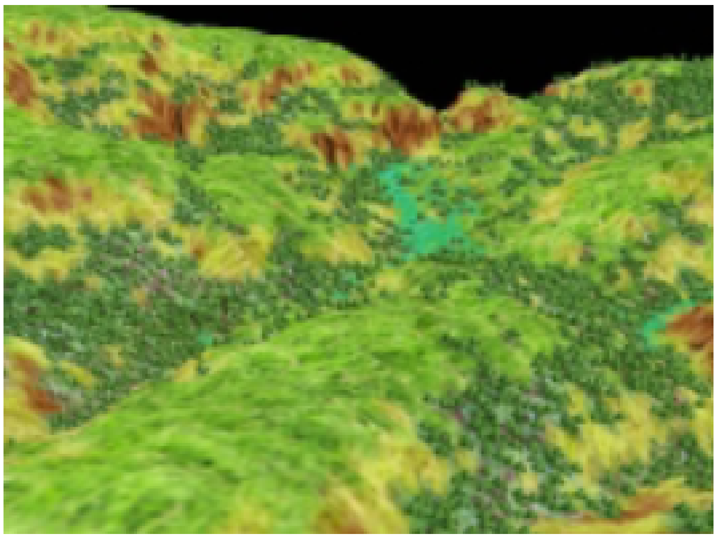

Figure 2.10: Vegetation generated using predefined ecosystems [?]

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.

2.2.3 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.

2.2.3.1 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. [? ?] 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. [?], 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. [**?**] 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 2.2.3.1).

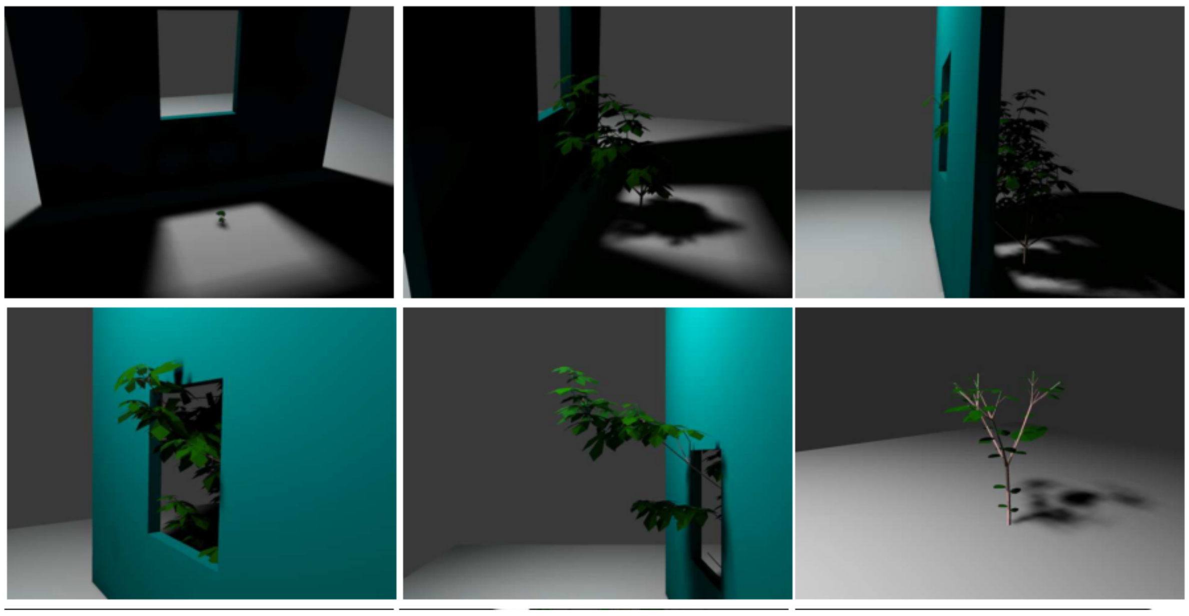

Figure 2.11: Plant growing towards light source [?]

2.2.3.2 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 [? ? ? ?].
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 [?]. In their work modelling the growth of struce trees, Berezovskava et al. [?] 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 [?] 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 [?] 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 2.2.3.2.

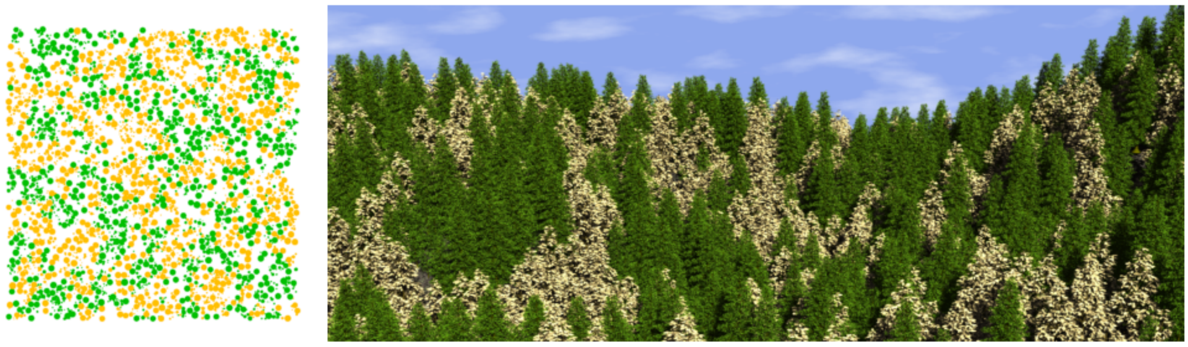

Figure 2.12: Plant placement using an ecosystem simulator modelled by L-Systems [?]. *Left:* Result of the simulation where orange circles indicate the positions of poplar trees and green circles the positions of spruce trees. *Right:* Reproduced virtual world where the location of individual plants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

Work by Deussen et al. [?] also uses L-Systems as the basis for an ecosystem simulator. As an extension to the work by Lane and Przemyslaw [?], 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. [?] 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. [**?**] 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. [**?**], 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).

2.2.3.3 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. [**?**], simulating 45 cycles for a single plant takes approximately 15 minutes.

2.2.4 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 2.2.2 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.

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

	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

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.