# 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

# System Overview

Multiple confined components serving specific purposes constitute the building blocks of the overall system, notably: *the resource gatherer*, *the resource clusterer*, *the plant selector*, *the ecosystem simulator* and *the distribution analyser and reproducer*. The purpose of this chapter is to give an overview of the system, each component and how they fit together. To do so, an overview of each component is given separately. Each clearly stating the components purpose, required inputs and what it outputs. To conclude, figure 3.6 gives an overview of the system, illustrating how the individual components fit together.

## 1.1 Resource Gatherer

Vegetation requires resources to grow, the amount of which uniquely identifies a given specie and associates it with a given climate and, subsequently, locations on earth. Determining resource data is essential, therefore, to generating realistic virtual worlds as it is vital to determining suitable vegetation and distribution thereof. The purpose of the resource gatherer is to determine, for each terrain vertex: *sun exposure*, *soil humidity*, *temperature* and *slope*. Figure 3.1 illustrates the output of the resource gatherer along with the user inputs required to generate them.

## 1.2 Resource Clusterer

Determining a suitable plant distribution for each individual terrain vertex is infeasible due to the associated computational cost. To reduce the amount of plant distributions to calculate, K-means clustering is performed on the terrain to group together points with similar resource properties. The mean value of each cluster is then used to determine suitable vegetation and distribution thereof. Figure 3.2 illustrates the input requirements and output of this component.

## 1.3 Plant Selector

Given the mean value of the individual clusters, the plant selector determines the plants which are able to survive in each terrain cluster and calculates for each of them a suit-

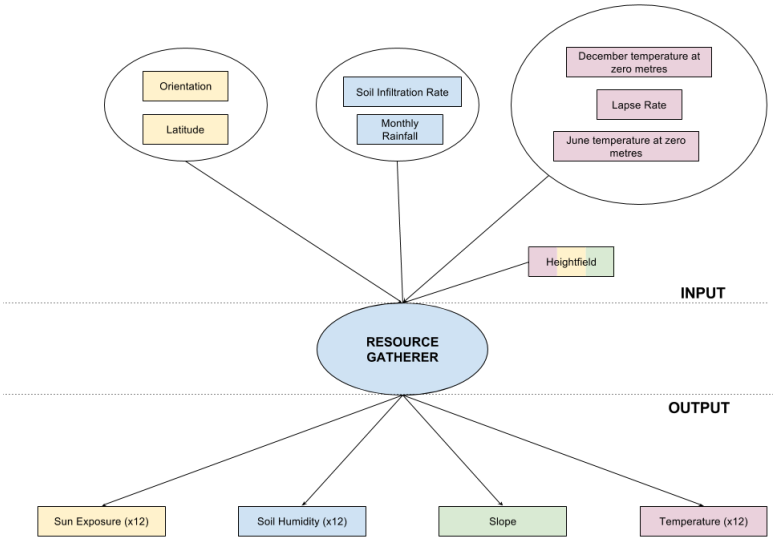



Figure 1.1: Resource gatherer overview with colour coding to correlate input with corresponding output.

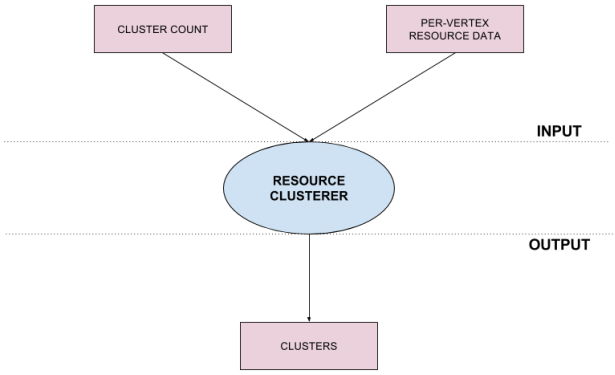



Figure 1.2: Resource clusterer overview.

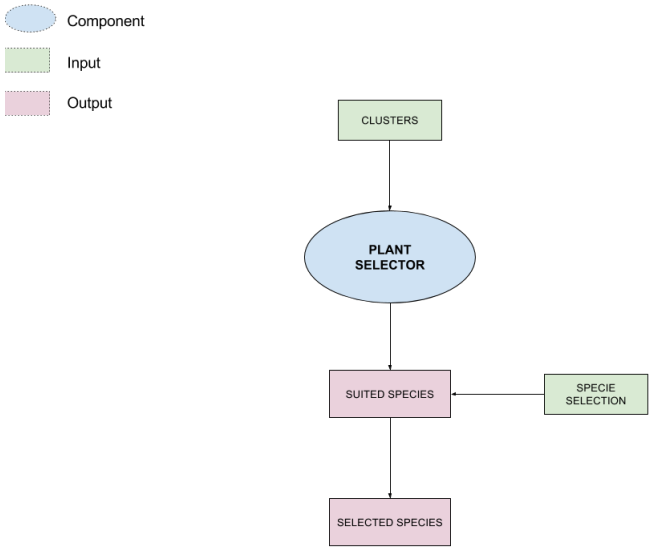



Figure 1.3: Plant selector overview.

ability score. This score depicts how suited a specie is to each individual cluster and is illustrated to the user for informational purposes in order to facilitate the specie selection procedure. Figure 3.3 shows the input requirements and outputs of this component.

## 1.4 Ecosystem Simulator

The ecosystem simulator is used to determine a valid plant distribution given a set of plant species and resources (soil humidity, illumination, slope and temperature). To do so, it simulates plants spawning, growing, battling and dying through time at monthly intervals on a hundred by hundred metre simulation area. Figure 3.4 shows the input requirements and outputs of this component.

## 1.5 Distribution Analyser and Reproducer

Because the ecosystem simulator is computationally expensive, the simulation area is restricted to ten thousand square metres (hundred by hundred metres). In order to place vegetation in clusters with larger surface areas, radial distribution analysis and reproduction is performed. This technique analyses the variation in plant density over distance of an input exemplar in order to generate pair correlation histograms which are used to reproduce distributions matching the characteristics of the input exemplar. Because the reproduction is much less computationally costly than the ecosystem simulator, it is possible to efficiently produce distributions covering much larger areas. Figure 3.5 shows the input requirements and outputs of this component.

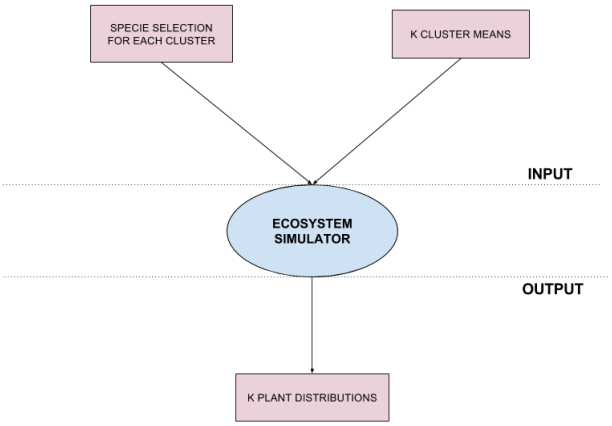



Figure 1.4: Ecosystem simulator overview.

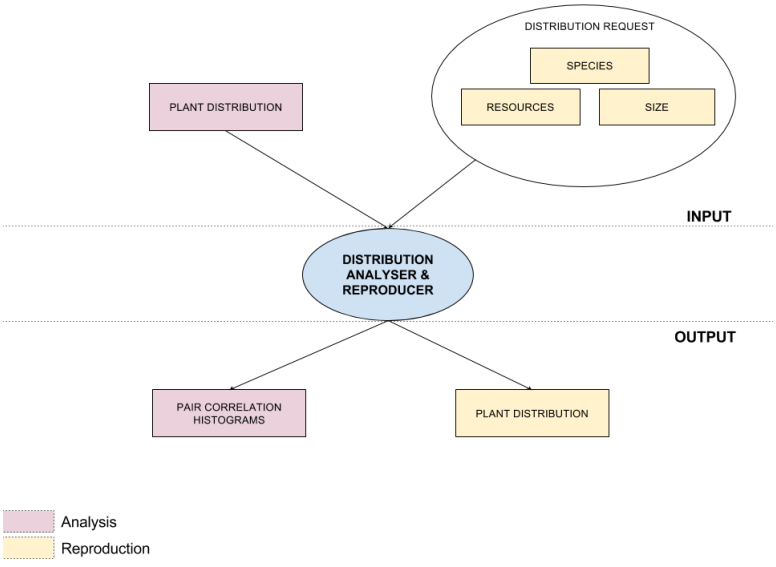



Figure 1.5: Distribution analyser and reproducer overview.

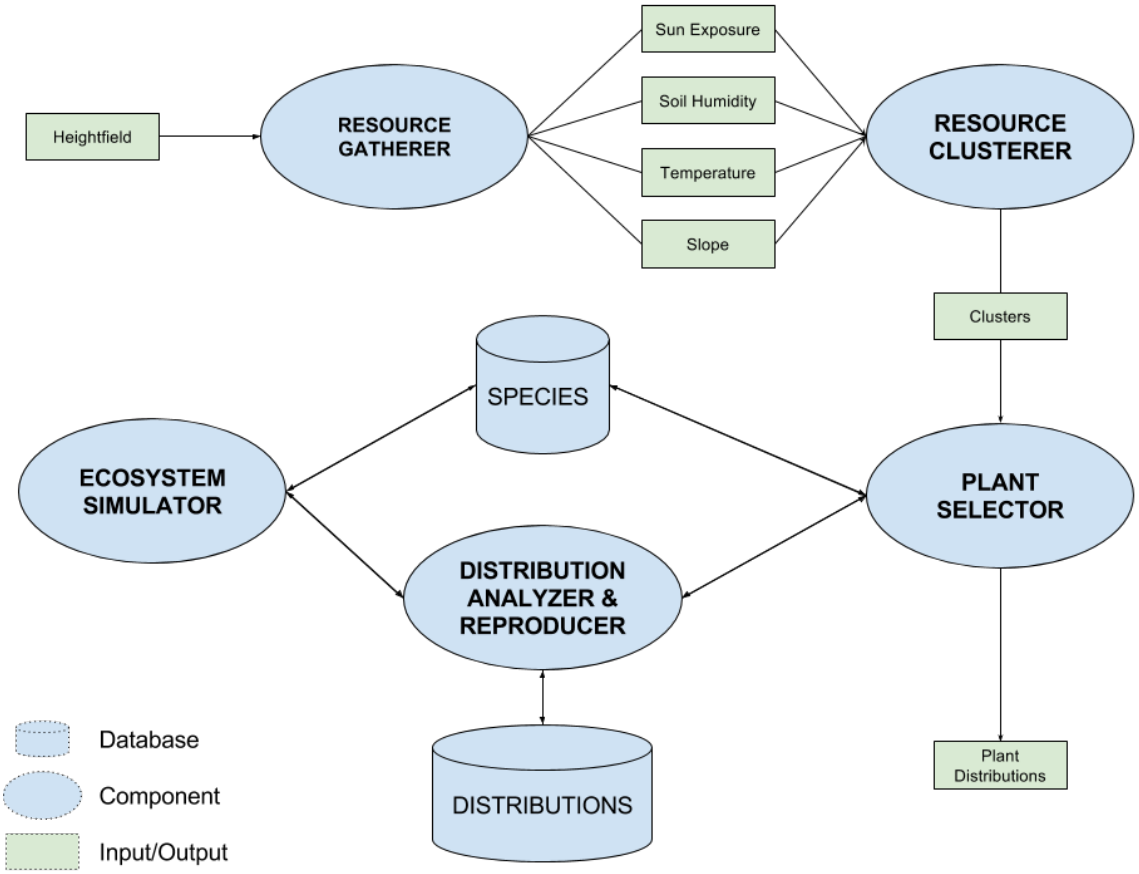



Figure 1.6: System overview

# Chapter 2

# Clustering

Each point on the terrain has a number of associated resource properties (summarised in table 5.1) which are used to depict suitable vegetation and, using an ecosystem simulator, a distribution of said vegetation. This simulator is computationally expensive, however, and running it for each terrain vertex is infeasible. As such, to make the number of ecosystem simulations which need to be run manageable, clustering is performed on the terrain based on the resources associated to the individual points. The type of clustering algorithm used is *K-Means Clustering*, discussed in the dedicated *K-Means Clustering* section. To accelerate the clustering algorithm and strive for real-time results, it is implemented to run on the GPU, details of which can be found in the *GPU Implementation* section. To conclude, the performance and results of the clustering algorithm is discussed.

| **Resource** | **Count** | **Comments** |
|---|---|---|
| Slope | 1 | - |
| Temperature | 12 | Temperature for each month |
| Illumination | 12 | Illumination for each month |
| Soil Humidity | 12 | Soil humidity for each month |

Table 2.1: Resource properties associated with a single point on the terrain.

## 2.1 K-Means Clustering

K-means is a well known and broadly used clustering algorithm [**?** ] which has been used to tackle a wide range of problems including character recognition [**?** ], image segmentation [**? ?** ] and compression [**?** ]. The algorithm groups points into clusters based on the euclidean distance from a set of $K$ cluster means.

Given a set of data points $P$ and a configured value $k$, k-means clustering behaves as follows:

1. Choose $K$ points at random to act as the seed cluster means $C_{mean}$.
2. Iterate through every data point $P_n$ from $P$ and calculate its Euclidean distance with each cluster mean using equation 5.1. Point $P$ becomes a member of its closest cluster.
3. Use the members of each cluster to calculate the new cluster means.
4. Repeat from step 2.

$$D(A, B) = \sum_{n=1}^{nvalues} (value_n(A) - value_n(B))^2 \tag{2.1}$$

Where:

- **A** and **B** are either data points or a cluster mean.
- **nvalues** are the number of numerical values associated with each data point.
- $\mathbf{value_n(A)}$ is the data value $n$ of data point A.

### 2.1.1 Customising the Euclidean Distance Calculation

The Euclidean distance calculation outlined in equation 5.1 works well when all values are in the same dimension. Unfortunately, this is not the case for terrain resource data as illumination, slope, temperature and soil humidity are all measured in different units. This means that a one millimetre change in soil humidity will have as musch of an influence on clustering than a one degree change in temperature. To equalise the effect on clustering across all resources, weighting is performed when calculating the Euclidean distances as illustrated in table 5.2.

| **Value** | **Weighting** |
|---|---|
| Slope | 1 |
| Temperature | 1 |
| Illumination | 1 |
| Soil Humidity | 0.1 |

Table 2.2: Resource weighting when calculating Euclidean distances between.

Equation 5.2 is used to calculate the Euclidean distance between two terrain positions (or cluster means) $A$ and $B$ for clustering.

$$D(A,B) = (S(A)-S(B))^2+\sum_{n=1}^{12}(T_n(A)-T_n(B))^2+(H_n(A)-H_n(B))^2+(0.1\times(I_n(A)-I_n(B))^2) \tag{2.2}$$

Where:

- *S(x)* is the slope of point or cluster mean x.
- $T_n(x)$ is the temperature of point or cluster mean x at month n.
- $H_n(x)$ is the soil humidity of point or cluster mean x at month n.
- $I_n(x)$ is the illumination of point or cluster mean x at month n.

### 2.1.2 Choosing Seed Cluster Means

A downside of classic K-means clustering techniques is that they are non-reproducible. This is because the final clusters depend heavily on the initial points which were chosen to act as the cluster means. As these are selected at random, different runs will result in different clusters. Reproducibility is important here, however, as if the clusters can't be reproduced neither will the final terrain.

A solution to this problem is to initialise the cluster means using pre-determined points on the terrain rather than at random. However, it is also good for the initial seed points to contain distinct resource properties in order for the final clusters to form faster. To attempt to fulfil both these requirements, the seed points are selected at equal distances on the terrain's diagonal. This ensures reproducibility as the same seed points will always be selected for a given terrain and attempts to cater for the second by maximizing seed point terrain coverage.

### 2.1.3 Configuring the Number of Clusters *K*

The value of $K$ will affect the number of ecosystem simulators which need to be run when placing vegetation on the terrain. Although larger values will potentially result in more realistic vegetation distributions, the added simulations will required additional processing time. Choosing a good value for $K$ is therefore about finding a balance between realism and processing time and, as such, depends on user requirements. Because of this, the value of $K$ is required as input from the user before the clustering is performed.

| Storage Type | Data Type | Element Count | Usage |
|---|---|---|---|
| 3-D Texture | Float | W × H × 12 | Monthly Soil Humidity |
| 3-D Texture | Float | W × H × 12 | Monthly Illumination |
| 2-D Texture | Float | W × H | Temperature |
| 2-D Texture | Float | W × H | Slope |
| 3-D Texture | Float | $K \times WorkGroups_x \times 13 \times WorkGroups_y$ | Slope and Humidity Reducer |
| 3-D Texture | Float | $K \times WorkGroups_x \times 2 \times WorkGroups_y$ | Temperature Reducer |
| 3-D Texture | Float | $K \times WorkGroups_x \times 12 \times WorkGroups_y$ | Daily Illumination Reducer |

Table 2.3: Global memory allocations necessary for the GPU implementation of K-Means clustering. $W$ and $H$ are the width and height of the terrain respectively. $WorkGroups_x$ and $WorkGroups_y$ are the horizontal and vertical workgroup count respectively.

## 2.2 GPU Implementation

Performing K-Means clustering is an *O(KN)* problem where K is the number of clusters. As a consequence, given a cluster count, the processing time will increase linearly with terrain area. For large terrains with millions of vertices this could prove time consuming and, consequentially, have a negative effect on user experience. To accelerate the clustering process and maximize user-experience it is implemented to make use of the heavily parallel architecture of the GPU. Below are discussed the details and optimizations of this implementation.

### 2.2.1 Calculating cluster membership

Given the cluster means for iteration $i$, the algorithm must determine to which cluster each terrain vertex belongs. To do so efficiently, each GPU core is associated a unique vertex and is responsible for determining it's cluster membership.

### 2.2.2 Calculating the new cluster means

Once all terrain vertices have been assigned to a cluster, they must be iterated over in order to calculate the new means of each cluster.

Within a work-group (group of cores which are guaranteed to run in parallel and have access to shared memory), when each core calculates it's distance from individual cluster means, it loads its associated resource data into a unique index of fast-access shared memory. It also stores in shared memory the calculated cluster membership. This data is then used to calculate, within each work-group, $k$ new work-group cluster means, in parallel. The $k$ work-group cluster means are then stored to global memory along with

the member count for each cluster.

Finally, the global cluster means are calculated by $k$ cores in parallel using the means calculated for each individual work group along with their associated member counts.

### 2.2.3 Storage Optimization

The temperature on the terrain increases linearly with altitude. As such, even though the temperature changes monthly on the terrain, the temperature difference between two points $P_a$ and $P_b$ will remain constant throughout the year. Calculating the Euclidean distance from a given point $A$ to a cluster mean is identical to calculating the difference between two points on the terrain. As a consequence, it is only necessary to use a single months temperature data to establish terrain clusters, saving vital GPU storage space.

### 2.2.4 Minimizing CPU to GPU data transfers

As mentioned previously, copying data to and from CPU to GPU is a costly process. To prevent these costly transfer operations, all data required for the clustering algorithm (table 5.3) is copied to the GPU at the start of the clustering process and no further transfers are performed until completion.

## 2.3 Performance

As mentioned previously, the user must specify the requested cluster count $k$. In order to find a suitable value for $k$, the user will need to trial a number of different values. To minimize any negative effect on user-experience, therefore, it is important the clustering performs in near real-time, irrespective of terrain size and cluster count.

The performance of the CPU and GPU clustering implementations are analysed below along with an evaluation of the GPU speed-up. In order to evaluate the performance of the different implementations, the clustering time is analysed in relation to terrain size and cluster count. In order to accurately compare their performance, the same terrains are used with identical resources specified. All tests were performed on a machine with specifications outlined in appendix D.

### 2.3.1 CPU Performance

Figure 5.1 shows the clustering time achieved on the CPU for different terrain sizes and number of clusters. From this data, it is possible to conclude that:

- The clustering time increases linearly with the number of clusters to produce.
- The clustering time is proportional to terrain area.

Although the clustering time is reasonable for smaller terrains, this processing time increases sharply with the terrain size. This is especially true when combined with an increase in the number of clusters to generate.

### 2.3.2 GPU Performance

Figure 5.2 illustrates the performance of the same tests run on the GPU. The following conclusions can be made from this data::

- The processing time increases linearly with cluster count but at a significantly slower rate than on the CPU. This is because individual clusters are managed in parallel on the GPU.
- Similarly to the CPU implementation, the processing time increases linearly with terrain area. Unlike the cluster count, however, the rate of increase is comparable to that of CPU implementation. The reason for this is because, although individual terrain vertices are managed in parallel on the GPU implementation, the number of vertices far outweigh the number of GPU cores.

### 2.3.3 GPU Speed-up

Figures 5.3, 5.4 and 5.5 compare the CPU and GPU clustering processing time for square terrains of size 256, 512 and 1024 respectively. These histograms show that the GPU greatly outperforms the CPU, irrespective of terrain size and cluster count. Also visible

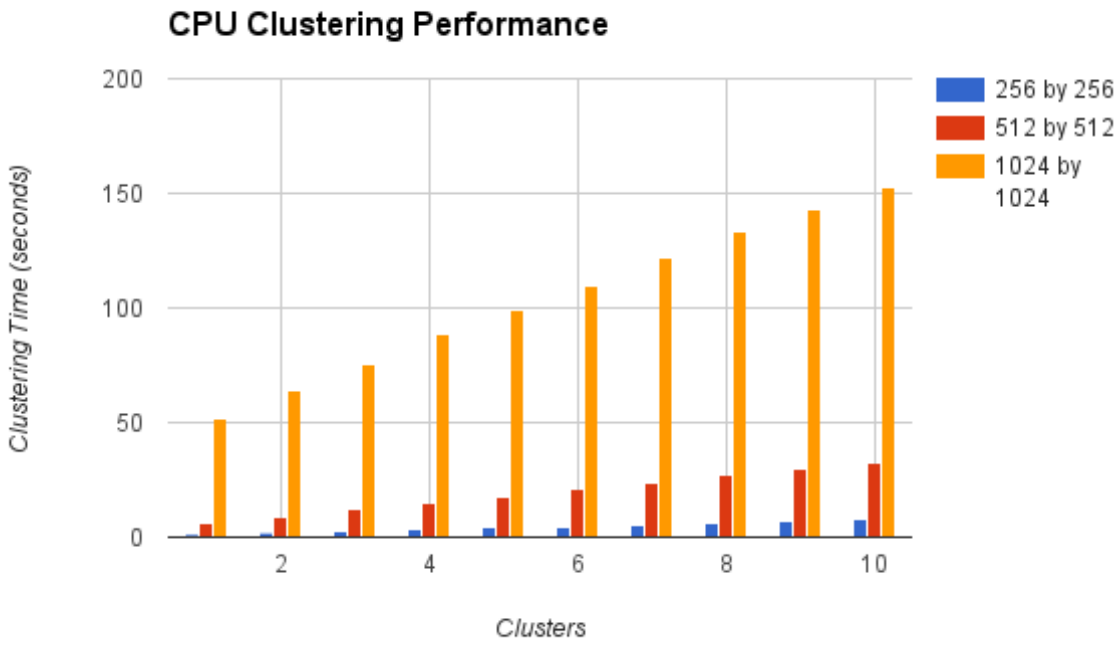



Figure 2.1: Time it takes for the clustering process to complete on the CPU depending on the cluster count. The analysis was performed for terrains of size: 256 by 256 (blue), 512 by 512 (red) and 1024 by 1024 (orange).

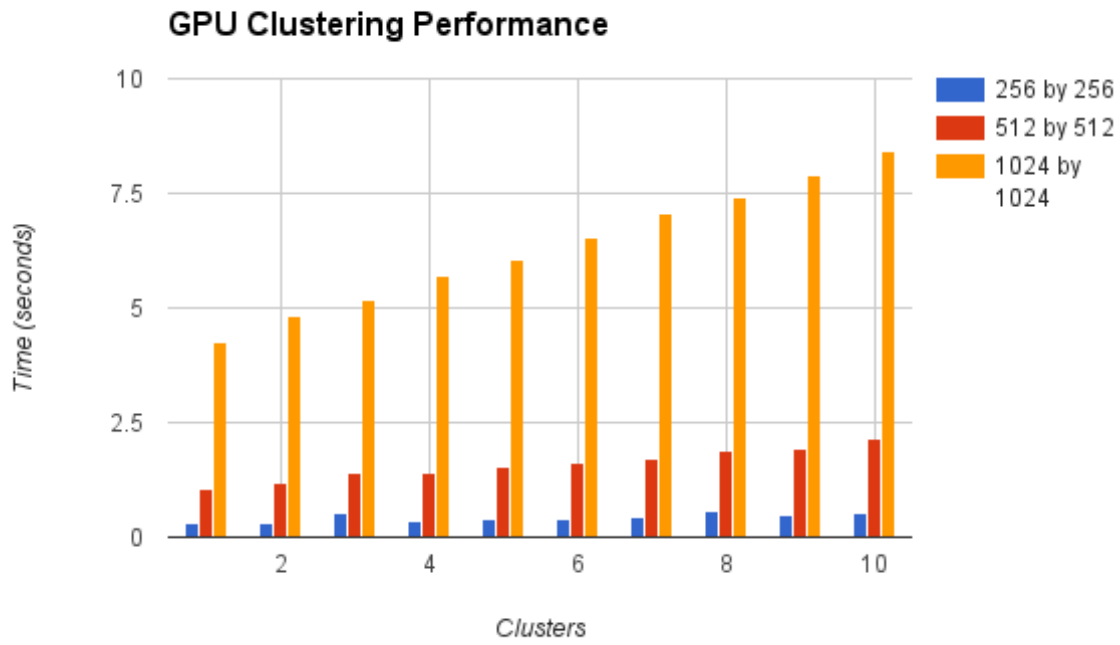



Figure 2.2: Time it takes for the clustering process to complete on the GPU depending on the cluster count. The analysis was performed for terrains of size: 256 by 256 (blue), 512 by 512 (red) and 1024 by 1024 (orange).

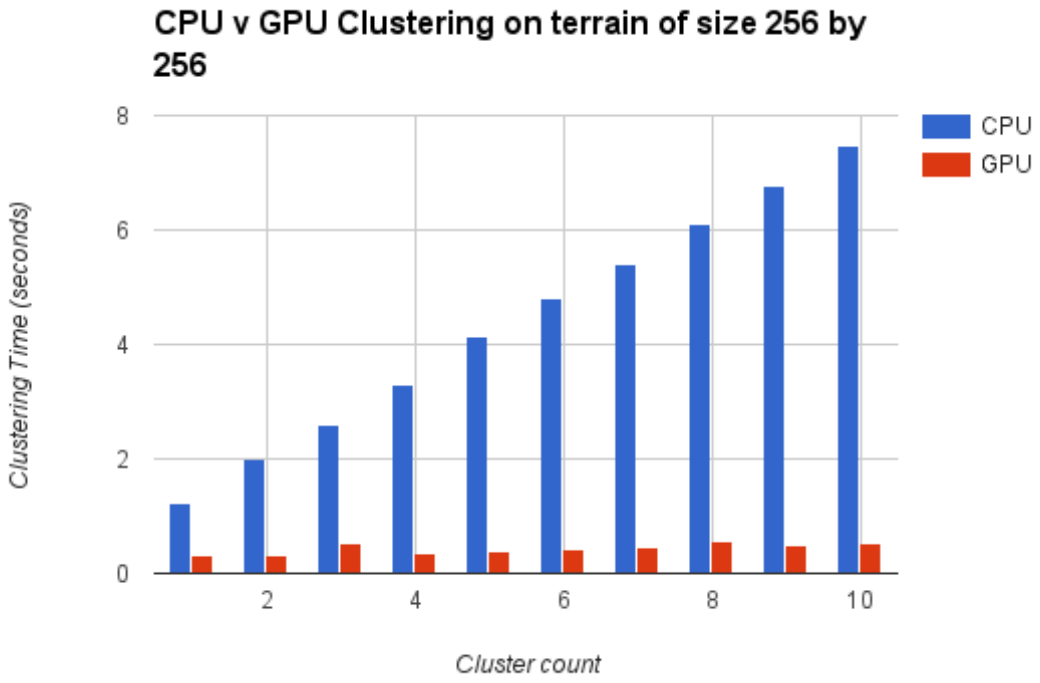



Figure 2.3: Comparison of CPU (blue) and GPU (red) clustering times on a 256 by 256 terrain.

in these graphics is the increased sensitivity to the cluster count of the CPU implementation over the GPU one.

As well as confirming the increased sensitivity to cluster count of the CPU implementation, figure 5.6 which plots the GPU speed-up for each terrain size and cluster count, shows the speed-up also increases with terrain size.

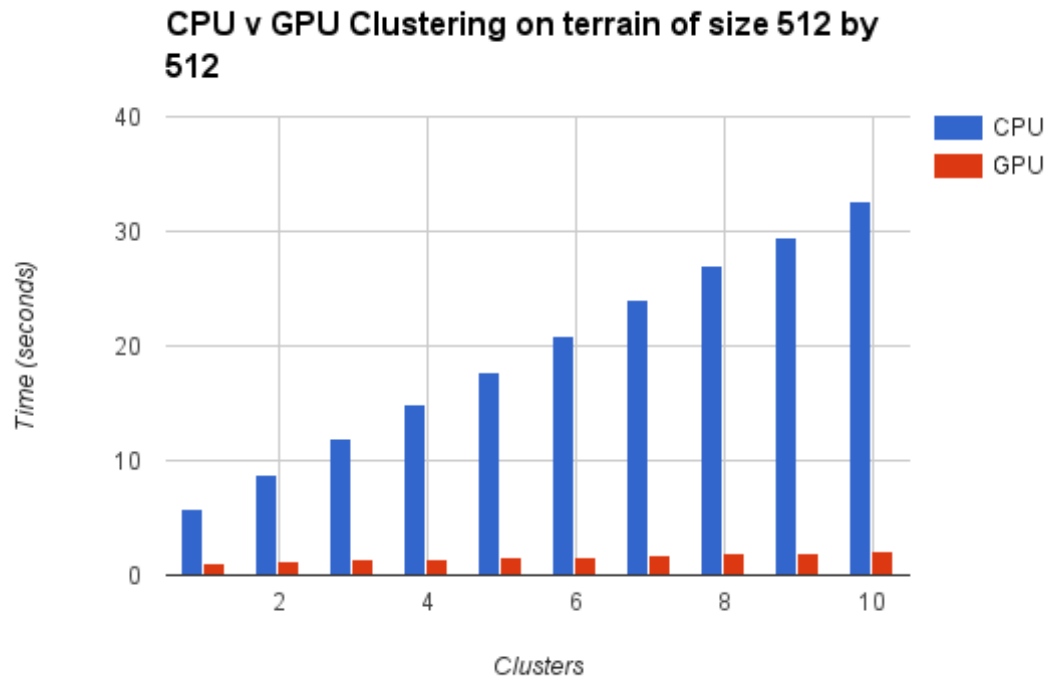



Figure 2.4: Comparison of CPU (blue) and GPU (red) clustering times on a 512 by 512 terrain.

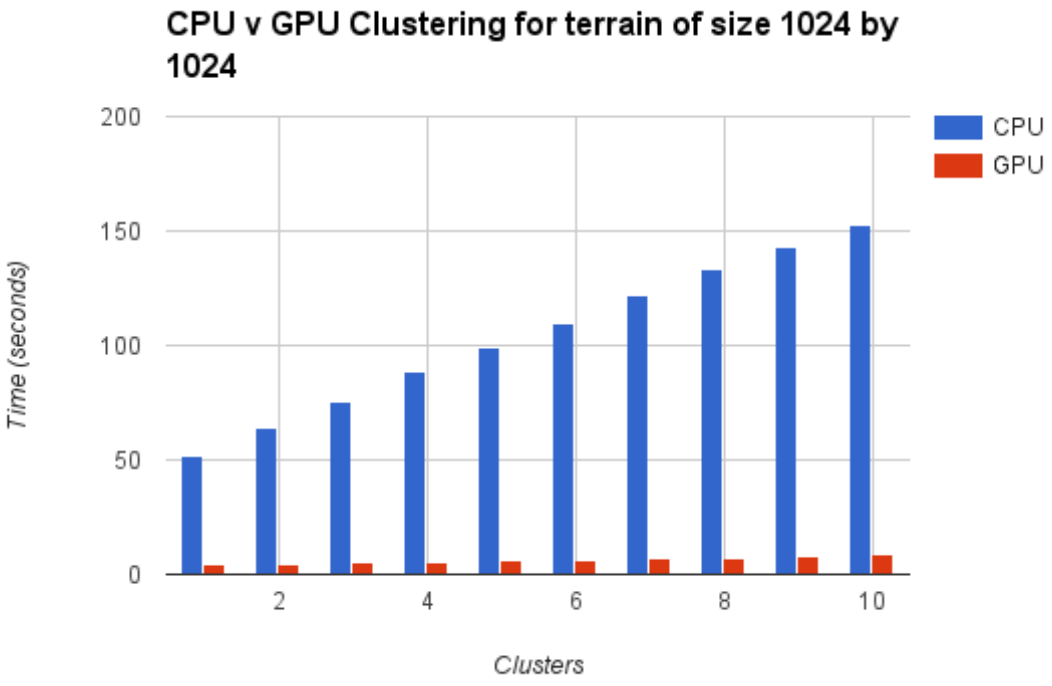



Figure 2.5: Comparison of CPU (blue) and GPU (red) clustering times on a 1024 by 1024 terrain.

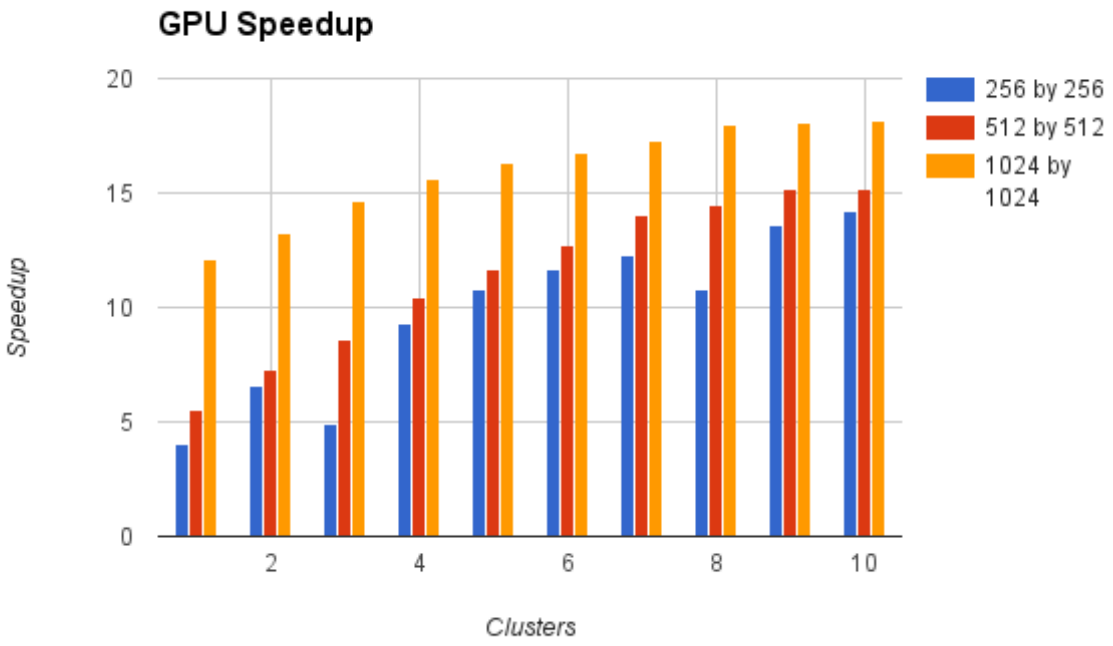



Figure 2.6: Calculated clustering speed-up of the GPU implementation compared to the CPU implementation for square terrains of size 256 (blue), 512 (red) and 1024 (yellow).

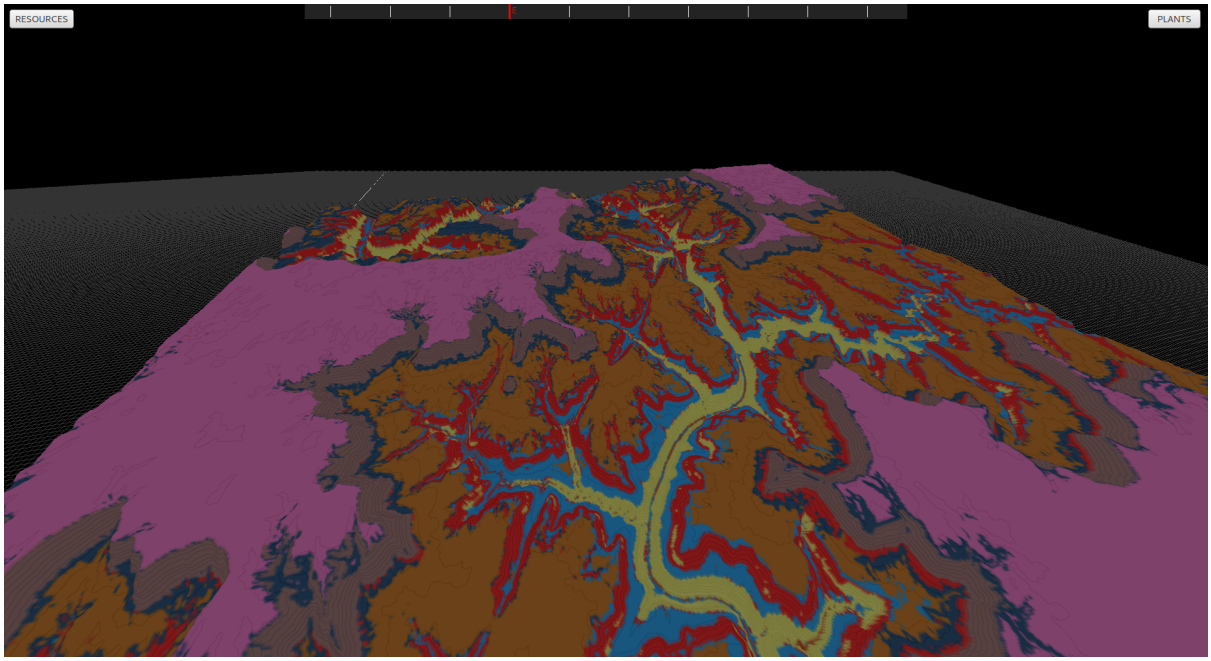
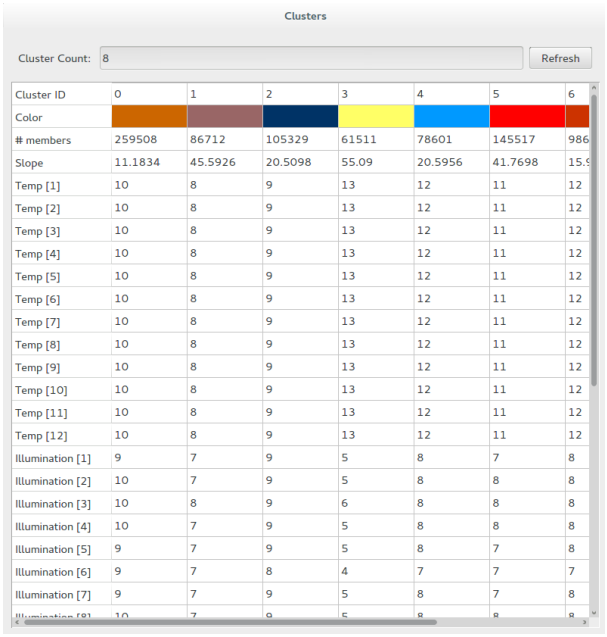



Figure 2.7: Cluster Overlay.

## 2.4 Overlay and Cluster Descriptions

When clustering is complete, each point on the terrain is associated with one of $k$ unique clusters. To make this association apparent to the user a cluster overlay is displayed. The clustering overlay attributes a unique color to each cluster and subsequently each terrain vertex a unique color based on cluster membership (see figure 5.7). Along with the terrain overlay, a dialogue shows the properties (color, member count, resources) of each cluster (see figure 5.8).




| Cluster ID | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Color | | | | | | | |
| # members | 259508 | 86712 | 105329 | 61511 | 78601 | 145517 | 986 |
| Slope | 11.1834 | 45.5926 | 20.5098 | 55.09 | 20.5956 | 41.7698 | 15.9 |
| Temp [1] | 10 | 8 | 9 | 13 | 12 | 11 | 12 |
| Temp [2] | 10 | 8 | 9 | 13 | 12 | 11 | 12 |
| Temp [3] | 10 | 8 | 9 | 13 | 12 | 11 | 12 |
| Temp [4] | 10 | 8 | 9 | 13 | 12 | 11 | 12 |
| Temp [5] | 10 | 8 | 9 | 13 | 12 | 11 | 12 |
| Temp [6] | 10 | 8 | 9 | 13 | 12 | 11 | 12 |
| Temp [7] | 10 | 8 | 9 | 13 | 12 | 11 | 12 |
| Temp [8] | 10 | 8 | 9 | 13 | 12 | 11 | 12 |
| Temp [9] | 10 | 8 | 9 | 13 | 12 | 11 | 12 |
| Temp [10] | 10 | 8 | 9 | 13 | 12 | 11 | 12 |
| Temp [11] | 10 | 8 | 9 | 13 | 12 | 11 | 12 |
| Temp [12] | 10 | 8 | 9 | 13 | 12 | 11 | 12 |
| Illumination [1] | 9 | 7 | 9 | 5 | 8 | 7 | 8 |
| Illumination [2] | 10 | 7 | 9 | 5 | 8 | 8 | 8 |
| Illumination [3] | 10 | 8 | 9 | 6 | 8 | 8 | 8 |
| Illumination [4] | 10 | 7 | 9 | 5 | 8 | 8 | 8 |
| Illumination [5] | 9 | 7 | 9 | 5 | 8 | 7 | 8 |
| Illumination [6] | 9 | 7 | 8 | 4 | 7 | 7 | 7 |
| Illumination [7] | 9 | 7 | 9 | 5 | 8 | 7 | 8 |
| Illumination [8] | 10 | 7 | 9 | 5 | 8 | 8 | 8 |

Figure 2.8: Cluster properties dialogue.

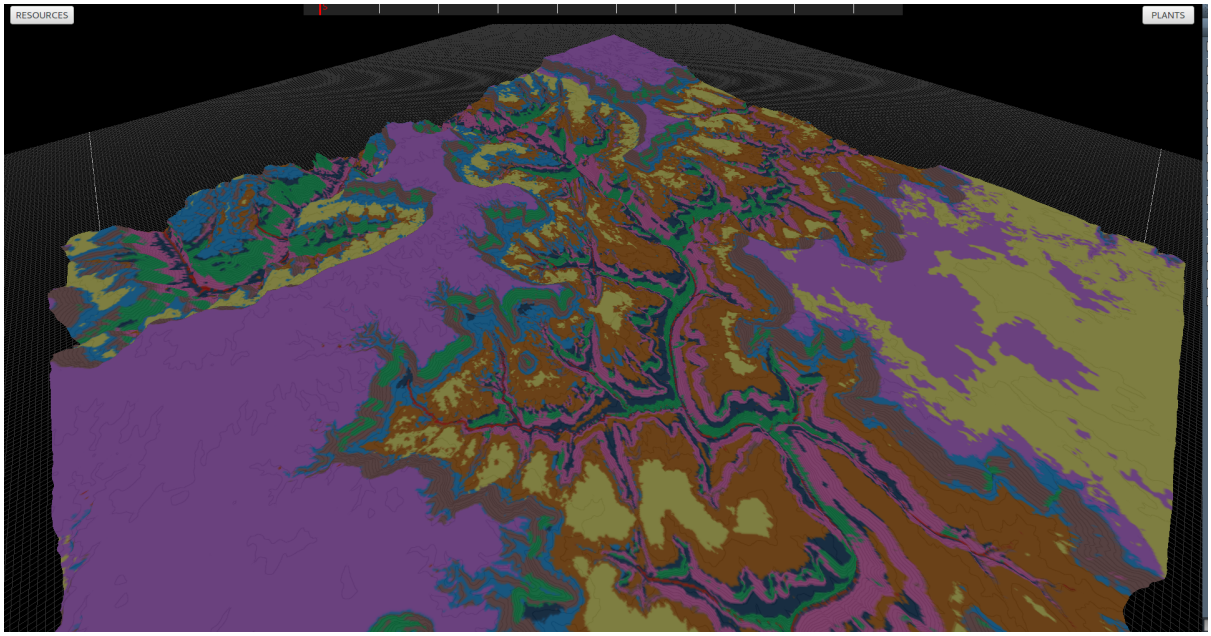



Figure 2.9: Clustering test: Resulting terrain clusters

## 2.5 Results

To test the clustering algorithm, a terrain is loaded, it's resources edited and five clusters produced. These clusters are subsequently analysed to ensure they successfully detect distinct resource features on which to cluster.

The terrain used is a model of the Grand Canyon using data from the US Geological Survey [1]. This terrain is chosen as its canyons and crevasses make ground illumination vary greatly.

The following resource edits were performed on the terrain:

- *Latitude*: Set to zero degrees (equator)
- *Soil Infiltration*: 5 millimetres for all terrain points with a slope under 30 degrees. All points with a slope over 30 degrees were set to 0 to simulate a cliff.
- *Rainfall*: 25 millimetres for every month.
- *Temperature*: 15 degrees at 0 meters in December. 30 degrees at 0 metres in June.

The resulting terrain clusters that form are displayed in figure 5.9 and summarized in appendix A.

Figures 5.10, 5.11, 5.12 and 5.13 show how much each cluster's illumination, temperature, soil humidity and slope vary from the terrain's average. Using this data, is is possible to find the key feature(s) of each individual cluster summarized in table 5.4.

[1] `http://www.usgs.gov`

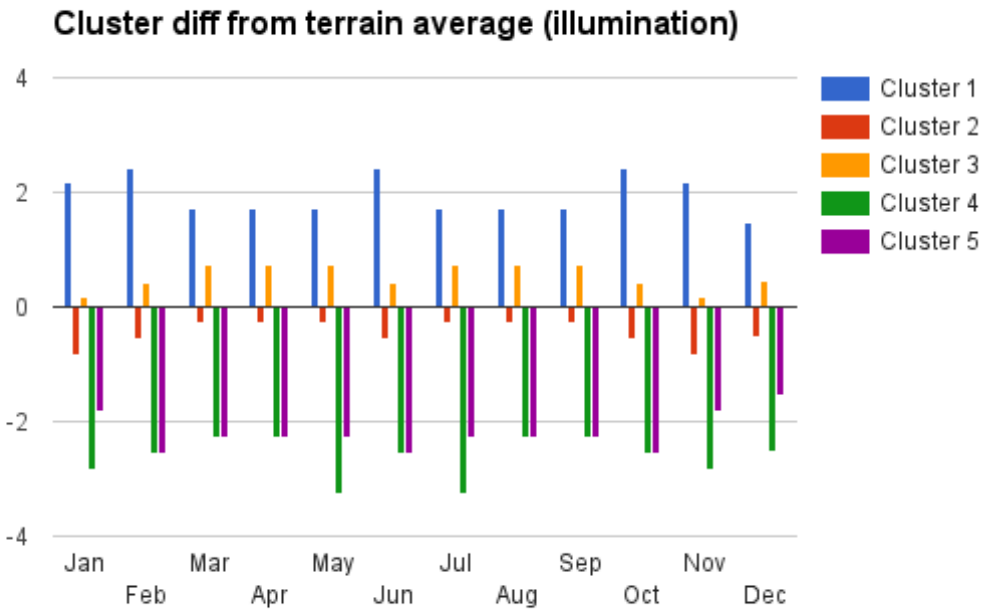



Figure 2.10: Monthly illumination for each cluster and the average over the whole terrain.

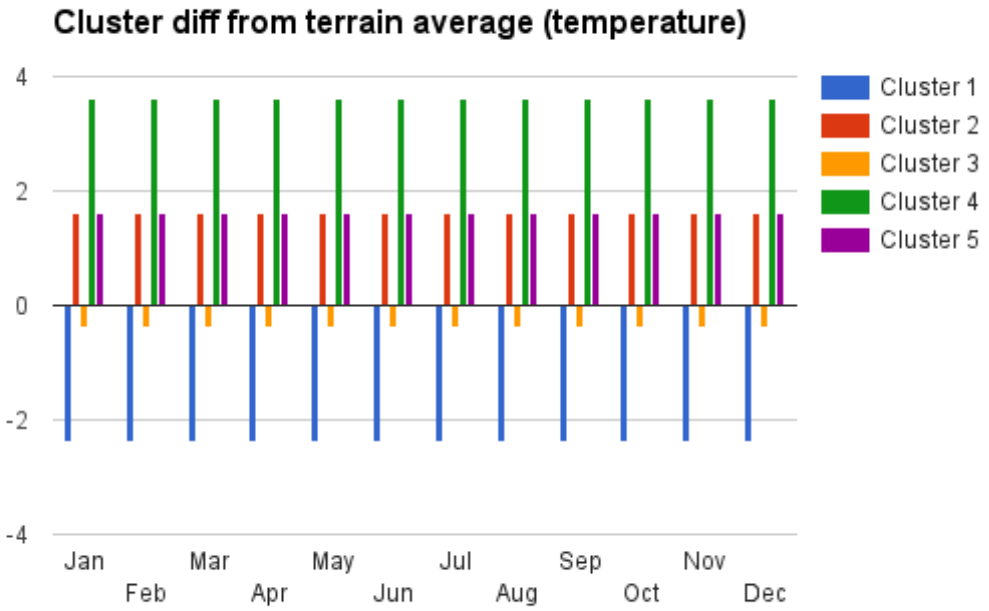



Figure 2.11: Monthly temperature for each cluster and the average over the whole terrain. Cluster 2 has the same values as cluster 4.

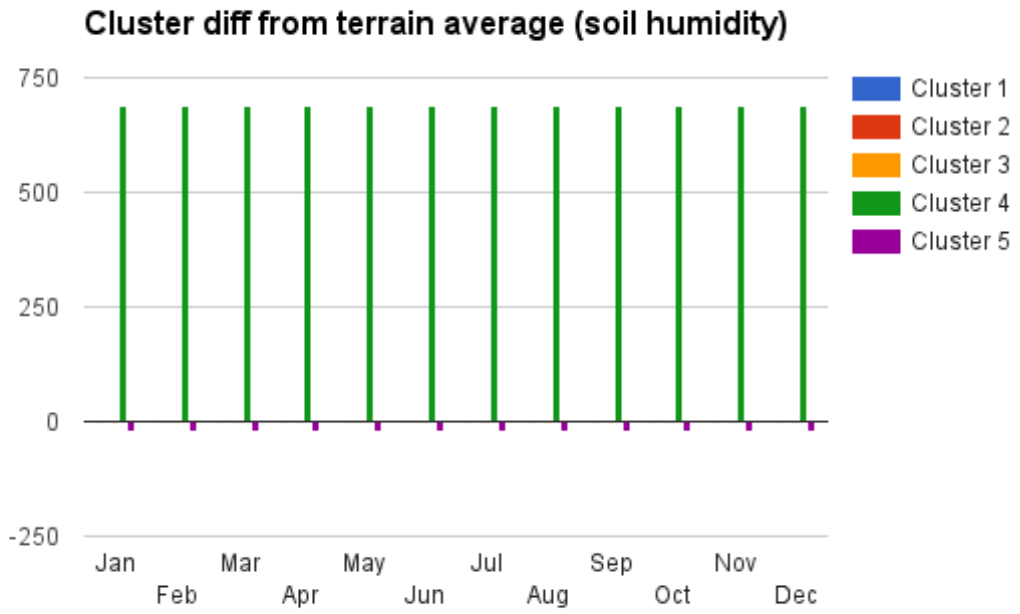



Figure 2.12: Soil humidity for each cluster (same for every month) and the average over the whole terrain.

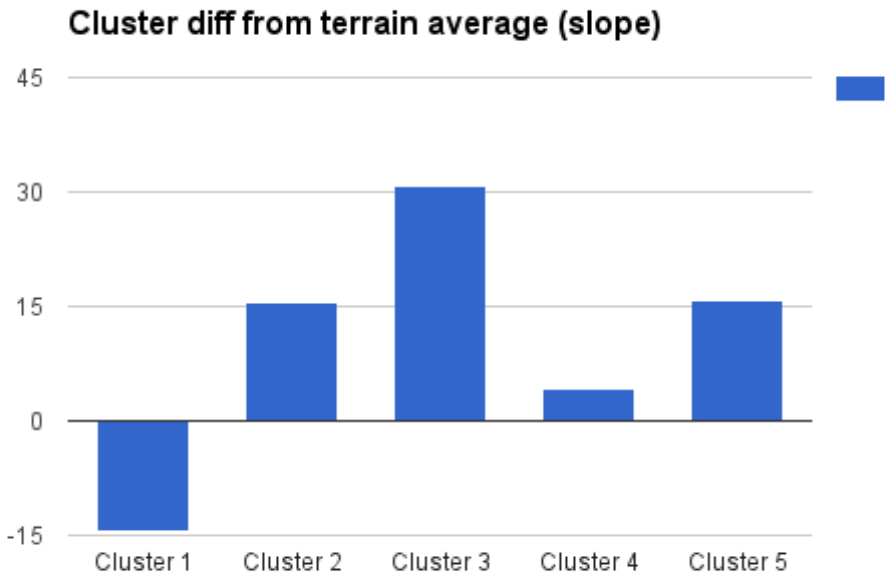



Figure 2.13: Slope for each cluster and the average over the whole terrain.

| Cluster | Illumination | Temperature | Soil Humidity | Slope |
|---|---|---|---|---|
| **1** | $\uparrow$ | $\downarrow$ | $\rightarrow$ | $\downarrow\downarrow$ |
| **2** | $\rightarrow$ | $\uparrow$ | $\rightarrow$ | $\uparrow$ |
| **3** | $\rightarrow$ | $\rightarrow$ | $\rightarrow$ | $\uparrow\uparrow\uparrow$ |
| **4** | $\downarrow\downarrow$ | $\uparrow\uparrow$ | $\uparrow\uparrow\uparrow$ | $\rightarrow$ |
| **5** | $\downarrow$ | $\uparrow$ | $\downarrow\downarrow$ | $\uparrow$ |

Table 2.4: Summary of cluster feature variance from terrain average.

# Chapter 3

# Vegetation

An essential part of rural terrains is vegetation. Available resources determine which plant species are able to grow and to what extent they strive in a given environment. Reproducing this link between species and climate is essential to determine suitable vegetation and, subsequently, generate plausible terrains.

To determine environments suited for given species, they are configured with associated resource requirements as outlined in *Plant Species*. Given these properties, it is possible to automatically filter out ill-suited plants from being suggested to the user. Information about this automatic filtering feature is outlined in *Plant Suitability Filtering*.

Although a multitude of plants can grow in a given environment, some will naturally strive more than others. This can be because resources are more adequate or they have a faster, more aggressive growth rate. To model this intra-specie battle for resources and determine a suitable vegetative state, an ecosystem simulator is used. Details of which can be found in *Ecosystem Simulator*.

The ecosystem simulator is computationally expensive and can take some time to determine a valid distribution. The simulation time is dependent on the number of plant instances, the simulation area and the duration. To accelerate the process, the ecosystem simulator is run on a small area and the resulting distribution analysed in order to efficiently reproduce it on larger areas. A caching system is also used to prevent users from having to run the same costly simulation more than once. Information about these features are discussed in *Plant Distribution Analysis and Reproduction*.

| Property | Value | Unit |
|---|---|---|
| **Slope** | Maximum Slope | Degrees |
| **Growth** | Maximum canopy | Centimetres |
| | Maximum root size | Centimetres |
| | Maximum height | Centimetres |
| **Ageing** | Start of decline | Months |
| | Maximum age | Months |
| **Seeding** | Maximum seeding distance | Metres |
| | Annual seed count | - |
| **Illumination** | Start of prime | hours |
| | End of prime | hours |
| | Minimum | hours |
| | Maximum | hours |
| **Humidity** | Start of prime | millimetres |
| | End of prime | millimetres |
| | Minimum | millimetres |
| | Maximum | millimetres |
| **Temperature** | Start of prime | degrees |
| | End of prime | degrees |
| | Minimum | degrees |
| | Maximum | degrees |

## 3.1 Plant Species

A database is used to store all plant species and their associated properties which are used to determine their ability to grow in given environments and, subsequently, deduce a plausible distribution using the ecosystem simulator. Details about these, how they can be edited and new species added is discussed in *Specie Properties* and *Storing Species* respectively.

### 3.1.1 Specie Properties

When configuring a new specie it is necessary to specify with it a set of properties which are used to link it to given suitable environments. These properties are discussed in detail below and summarized in table 6.1.1.

#### 3.1.1.1 Slope

Steep slopes cause essential water and soil nutrients to run-off, making them less rich and, therefore, less suited to plant growth. The slope angle also causes larger species to struggle to support their own biomass. For this reason, steeper slopes often cater for smaller plant species (grass, shrub, etc.). To model the effect of slope on given plant species, when configuring a new plant specie an associated *maximum slope* must be specified.

#### 3.1.1.2 Growth

To model the growth of a plant specie in the ecosystem simulator, it is necessary to specify: *Maximum height*, *maximum canopy width* and *maximum root size*. Using this along with the specie's ageing properties (see 6.1.1.3), it is possible to simulate the plants vertical growth (height), horizontal growth (canopy) and root coverage. Determining a plant's height and canopy width is also used to determine the shade it project's on other plants during the simulation. Modelling the plant's root growth is used to determine how far the plant can reach to fetch soil water. Note that a maximum canopy width of zero can be specified to model plants with no canopy.

#### 3.1.1.3 Ageing

Biological life-cycle varies greatly between plant species. Whereas annual and biennials have a fixed lifespan of one and two years respectively, perennial plant species can live far longer. To model the life-cycle of different plant species they must be configured with an associated *age of start of decline* and *maximum age*. Using these two values, it is possible to simulate a plant getting weaker and, therefore, becoming more susceptible to domination from it's surrounding plants.

#### 3.1.1.4 Seeding

It is necessary to replicate the spawning of offspring in the ecosystem simulator for two core reasons:

1. *Propagation*: Plants propagate on a terrain by producing new offspring which attempt to spawn and invade different areas.

2. *Succession*: New plants spawn to later succeed older and weaker plants of the same specie.

Depending on the specie, plants spawn new offspring by producing seeds or spores. Although biologically different, both can be considered identical for the sole purpose of modelling propagation and succession.

The number of seeds that a given plant creates annually is specie-dependent and therefore must accompany the specie's configuration.

Different species use different techniques to propagate their seeds. For example, some use fruit as a mechanism to propagate using animals digestive systems, some are coated with a sticky mucous to stick to the fur of animals passing by, some use the wind and some rely solely on gravity to take the seeds to the ground directly below. The seeding mechanism used directly affects the distance which can be achieved between parent and offspring and, to emulate this in the system, a *maximum seeding distance* is configured.

#### 3.1.1.5 Illumination

Illumination has a big impact on plant growth. Whereas some species strive in the shaded undergrowth, others require direct illumination all year round. To model this, the following illumination values must be configured with each plant specie:

- *Minimum daily illumination*: The minimum daily illumination, in hours, at which a plant of this specie can survive.
- *Prime daily illumination range*: The daily illumination range, in hours, which is optimal for the given specie.
- *Maximum daily illumination*: The maximum daily illumination, in hours, at which a plant of this specie can survive.

#### 3.1.1.6 Humidity (rainfall)

Soil water deposited into the soil by either rainfall or existing groundwater is absorbed by plant roots and is vital to its development and survival. Whereas some species have evolved to survive in arid climates with very little water, others require frequent downpours of rain.
To simplify water requirement specifications of different plant species, it is necessary to configure the amount of rainfall necessary as if this was the plant's only source of water (i.e no groundwater). The following values must be configured to grasp a species water requirements:

- *Minimum monthly rainfall*: The minimum monthly rainfall, in millimetres, at which a plant of this specie can survive.
- *Prime monthly rainfall range*: The monthly rainfall range, in millimetres, which is optimal for the given specie.
- *Maximum monthly rainfall*: The maximum monthly rainfall, in millimetres, at which a plant of this specie can survive.

#### 3.1.1.7 Temperature

Another aspect of climates which greatly affect plant growth is temperature and, in order to model a specie's temperature requirements. the following values need to be configured:

- *Minimum temperature*: The minimum temperature, in degrees, at which a plant of this specie can survive.
- *Prime temperature range*: The temperature range, in degrees, which is optimal for the given specie.
- *Maximum temperature*: The maximum temperature, in degrees, at which a plant of this specie can survive.

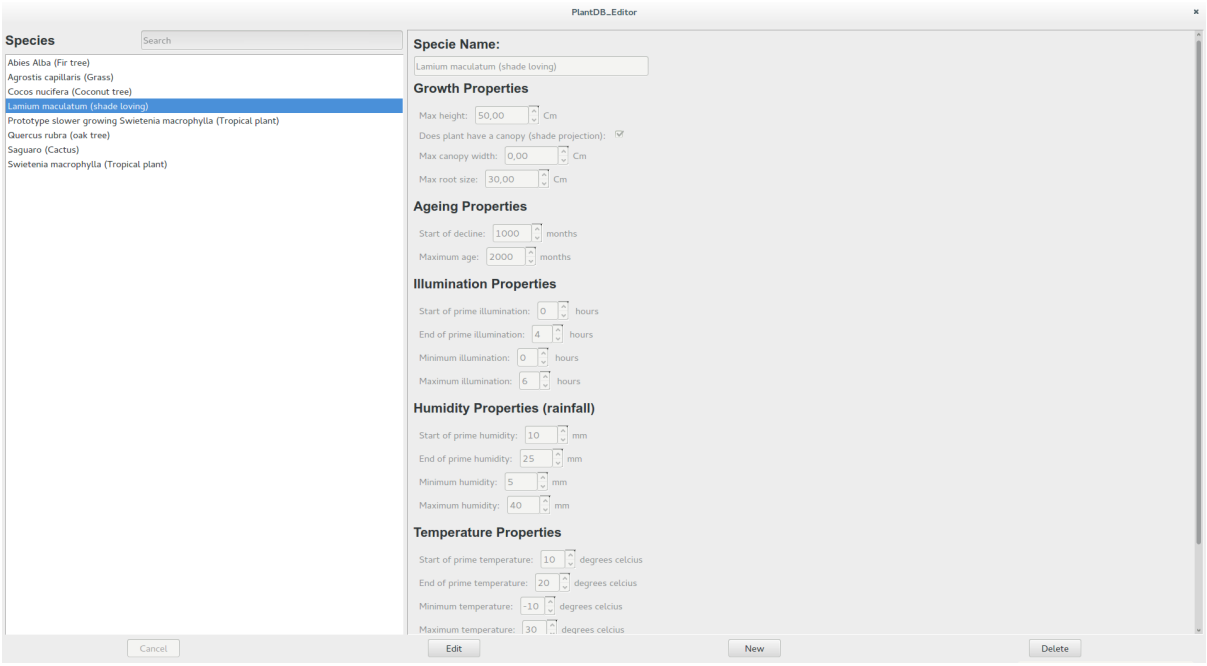



Figure 3.1: Plant database editor tool.

### 3.1.2 Storing Species

All species and associated properties are stored in a database for easy retrieval and filtering. A dedicated tool can be used to interact with the database to add or remove species and edit existing ones (see figure 6.1).

## 3.2 Plant Suitability Filtering

When the terrain clusters have been generated, the user must specify the plant species to plot. The ecosystem simulator is then used to determine a suitable distribution for the species given the resources associated with the individual clusters.

Rather than permit the user to select any plant from the database with which to populate the terrain, a filtering pass is performed in order to suggest only the plants which are able to survive. To determine whether a given specie is suited, a *specie suitability score* is calculated for each specie based on the resources of the given cluster. How this score is calculated is described in *Calculating the Specie Suitability Score* below.

As well as being used to filter out ill-suited species, this suitability score also highlights the species which strive in the given environment and could, as a consequence, prove to be useful information for the user when selecting the plants species to select. Various methods are used to effectively communicate the suitability score of each specie to the user, details of which are discussed in *Communicating the Suitability Score* below.

### 3.2.1 Calculating the Specie Suitability Score

The specie suitability score associated to a given specie $S$ for cluster $C$, $AggregateScore_S(C)$, illustrates how suited specie $S$ is to the environment of cluster $C$ on a range of 0 (ill-suited) to 100 (perfect conditions). To calculate it, the resource requirements of specie $S$ are matched with the resource availability of cluster $C$.

To determine this aggregate score, it is first necessary to determine the specie's suitability to the environment in terms of *slope*, *illumination*, *soil humidity* and *temperature*. A separate score is calculated for each and is discussed separately below.

#### 3.2.1.1 Slope Suitability Score

The slope suitability score determines how well suited the specie is in terms of slope and is calculated as illustrated in equation 6.1.

$$SlopeScore_S(x) = \begin{cases} 100, & \text{if } x \leq max \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

Where: $SlopeScore_S(x)$ is the slope suitability score for specie $S$ given slope $x$; $max$ is the maximum slope configured for specie $S$.

#### 3.2.1.2 Illumination Suitability Score

Because the illumination varies on a monthly basis, to calculate the aggregate illumination score for a given specie is is necessary to calculate the *illumination score* for each month as illustrated in equation 6.2.

$$IllumScore_S(x) = \begin{cases} 0, & \text{if } x \leq min \text{ or } x \geq max \\ \frac{x-min}{prime_{start}-min} \times 100, & \text{if } x \in ]min, prime_{start}[ \\ 100, & \text{if } x \in [prime_{start}, prime_{end}] \\ (1 - \frac{x-prime_{end}}{max-prime_{end}}) \times 100, & \text{if } x \in ]prime_{end}, max[ \end{cases} \tag{3.2}$$

Where: *IllumScore$_S$(x)* is the illumination suitability score for specie $S$ given illumination $x$; *min* is the minimum illumination configured for specie $S$;*max* is the maximum illumination configured for specie $S$; *prime$_{start}$* is the start of the prime illumination range configured for specie $S$;*prime$_{end}$* is the end of the prime illumination range configured for specie $S$.

Given the illumination scores for each month, it is possible to calculate the *average illumination score* using equation 6.3.

$$AvgIllumScore_S(x) = \begin{cases} \frac{\sum_{m=1}^{m=12} IllumScore_S(m)}{12}, & \text{if } IllumScore_S(m) > 0 \text{ for } m \in [1, 12] \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

#### 3.2.1.3 Humidity Suitability Score

The humidity also varies on a monthly basis and, as such, it is also necessary to calculate the humidity score for each month as illustrated in equation 6.4.

$$HumScore_S(x) = \begin{cases} 0, & \text{if } x \leq min \text{ or } x \geq max \\ \frac{x-min}{prime_{start}-min} \times 100, & \text{if } x \in ]min, prime_{start}[ \\ 100, & \text{if } x \in [prime_{start}, prime_{end}] \\ (1 - \frac{x-prime_{end}}{max-prime_{end}}) \times 100, & \text{if } x \in ]prime_{end}, max[ \end{cases} \tag{3.4}$$

Where: *HumScore$_S$(x)* is the humidity suitability score for specie $S$ given humidity $x$; *min* is the minimum humidity configured for specie $S$; *max* is the maximum humidity configured for specie $S$; *prime$_{start}$* is the start of the prime humidity range configured for specie $S$; *prime$_{end}$* is the end of the prime humidity range configured for specie $S$.

Given the humidity scores for each month, it is possible to calculate the *average humidity score* using equation 6.5.

$$AvgHumScore_S(x) = \begin{cases} \frac{\sum_{m=1}^{m=12} HumScore_S(m)}{12}, & \text{if } HumScore_S(m) > 0 \text{ for } m \in [1, 12] \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

#### 3.2.1.4 Temperature Suitability Score

Temperature also varies on a monthly basis and, as such, it is also necessary to calculate the temperature score for each month as illustrated in equation 6.6.

$$TempScore_S(x) = \begin{cases} 0, & \text{if } x \leq min \text{ or } x \geq max \\ \frac{x-min}{prime_{start}-min} \times 100, & \text{if } x \in ]min, prime_{start}[ \\ 100, & \text{if } x \in [prime_{start}, prime_{end}] \\ (1 - \frac{x-prime_{end}}{max-prime_{end}}) \times 100, & \text{if } x \in ]prime_{end}, max[ \end{cases} \tag{3.6}$$

Where: *TempScore$_S$(x)* is the temperature suitability score for specie $S$ given temperature $x$;*min* is the minimum temperature configured for specie $S$; *max* is the maximum temperature configured for specie $S$; *prime$_{start}$* is the start of the prime temperature range configured for specie $S$; *prime$_{end}$* is the end of the prime temperature range configured for specie *S*.

Given the temperature scores for each month, it is possible to calculate the *average temperature score* using equation 6.7.

$$AvgTempScore_S(x) = \begin{cases} \frac{\sum_{m=1}^{m=12} TempScore_S(m)}{12}, & \text{if } TempScore_S(m) > 0 \text{ for } m \in [1, 12] \\ 0, & \text{otherwise} \end{cases} \tag{3.7}$$

#### 3.2.1.5 Specie Suitability Score

The suitability score gives an overview of the specie's suitability and is calculated using equation 6.8

$$Score_S(s, i, h, t) = \begin{cases} \frac{SlopeScore_S(s)+AvgIllumScore_S(i)+AvgHumScore_S(h)+AvgTempScore_S(x)}{4}, & \text{if} \\ TempScore_S(m) > 0 \text{ and } AvgIllumScore_S(m) > 0 \text{ and} \\ AvgHumScore_S(m) > 0 \text{ and } AvgTempScore_S(m) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

### 3.2.2 Communicating the Specie Suitability Score

When all the terrain clusters have been created, the terrain suitability score for each specie in the plant database is calculated in relation to the resources of each individual cluster. If the calculated score is zero for all clusters, the specie is automatically filtered out to prevent the user from selecting it.

Color coding is used to make further use of the specie suitability score and intuitively communicate to the user the species most suited to the environment (see figure 6.2).

When the user selects a given specie, all intermediate scores which were used to calculate the *specie suitability score* are communicated to the user in histogram form (see figure 6.3).

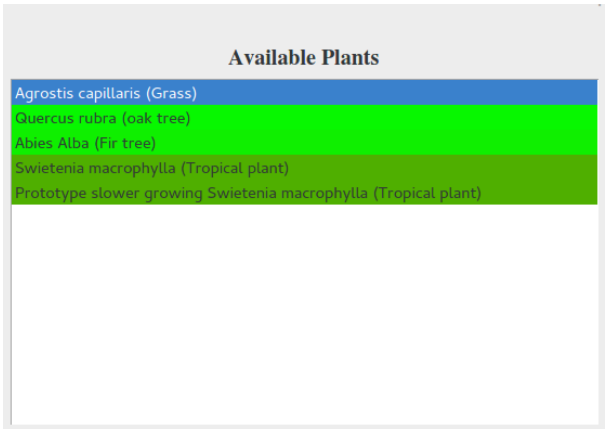



Figure 3.2: Color coding for specie suitability. The greener the highlight the more suited the specie is to the environment.

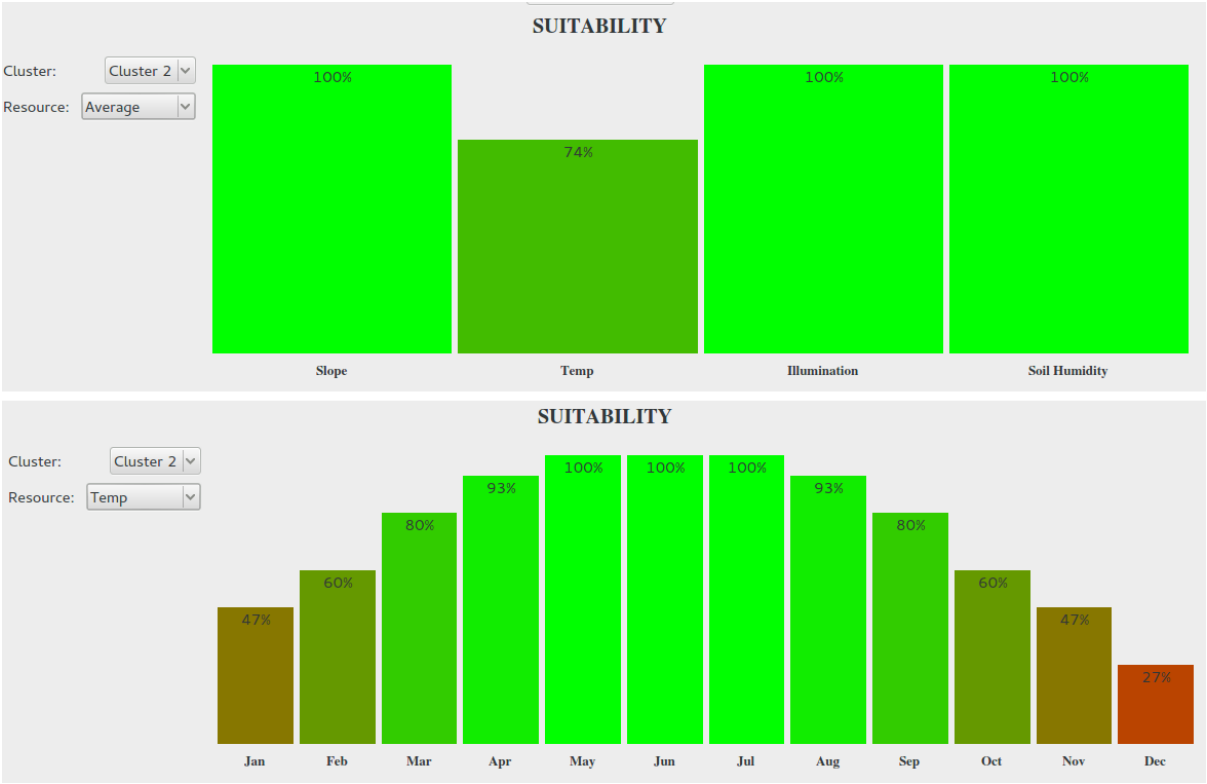



Figure 3.3: Average (top) and temperature (bottom) intermediate specie suitability histograms. Not displayed but also generated are the illumination and humidity intermediate specie suitability histograms.

## 3.3 Ecosystem Simulator

Once the user selects the union of all species that must appear in all clusters of the terrain, it is necessary to determine a valid vegetation distribution for each cluster. To do so, an ecosystem simulator is used similarly to that in the work by Deussen et al [DHL+98] and Lane and Przemyslaw [LP02]. Unlike these other ecosystem simulators, however, it isn't based on L-Systems and models both resource requirements and resource availability in greater detail. The purpose of ecosystem simulator is to determine, given a vegetative state, $S_t$ at time $t$, the vegetative state $S_{t+n}$ at time $t+n$, for any value of n.

To do so, the simulation advances through time in monthly intervals and the strength of all plant instances are re-calculated at each iteration. Their strength depend not only on resource properties of the given month but also surrounding plants as they battle for these resources. Determining the set S = {$P_1$, $P_2$, $P_3$, ...} of plants which compete for resources with plant $P_n$ depends on the spatial reach of $P_n$. Spatial awareness is therefore a key requirement of the simulation which is achieved by splitting the simulation area into a grid of cells as described in *Gridded Simulation Area*.

Within each cell of the gridded simulation area, resources must be distributed to the different plant instances present. How this is done is described in *Resource Distribution*.

Given the resources allocated to each plant instance, it is possible to calculate their strength. This is used as a representation of the plant's health and, consequentially, it's ability to survive and grow. Details about the plant's strength calculation and it's usage are discussed in *Plant Strength Calculation* and *Plant Strength Usage* respectively.

On an annual basis, new plant instances are spawned based on the specie's seeding properties. How this is done is discussed in *Spawning Plants*.

To conclude the discussion on the ecosystem simulator, performance and results will be analysed and discussed.

### 3.3.1 Gridded Simulation Area

The simulation window greatly effects the performance of the ecosystem simulator and, therefore, it is necessary to keep it to a minimum. However, too small a simulation area will fail to accurately model the interaction of larger plant species. Given these constraints, a simulation window of one hundred by one hundred meters is used, accurate to the nearest centimetre. To model plant's interacting and battling for resources, the window is split into cells to form a grid as illustrated in figure 6.4.

The size of individual cells can be configured to increase/decrease the resolution and, therefore, the accuracy of the simulation. As the simulation progresses, plant's grow, their spatial coverage increases, and they enter new grid cells. When a plant enters a new grid cell, it becomes a member of it and cell resources must be distributed to it as well as to all other plants in the cell. The information associated to each individual

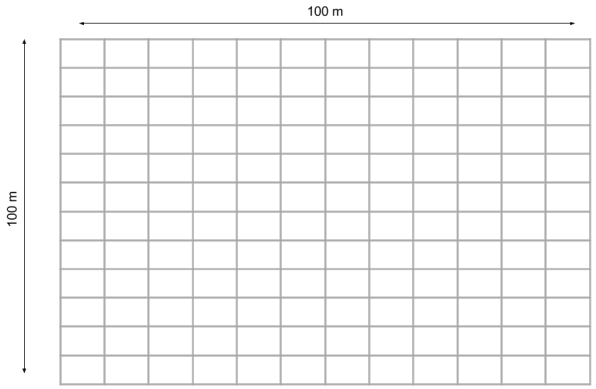



Figure 3.4: Gridded simulation area.

grid cell can be split into two categories: *time-dependent* and *simulation-dependent*. The time-dependent information depends only on the current month, is identical for every grid cell and is comprised of: the *soil humidity* and the *illumination*. The simulation-dependent information changes throughout the simulation as plants spawn, die and grow and is comprised of: the list of plants whose roots intersect the cell and the list of plants whose canopy intersects the cell.

### 3.3.2 Resource Distribution

The strength of each plant in the simulation must be recalculated on a monthly basis. To determine the strength of a given plant, it is necessary to know the illumination and humidity allocated to it along with the temperature. The temperature is not a distributable resource and is identical for all plant instances given the month. The allocated humidity and illumination is determined by averaging the resources distributed to it in each cell it overlaps in the grid. So, before the strengths of individual plant instances can be calculated, first each cell of the grid must be iterated over and illumination and humidity distributed to the plants contained within. How these are allocated is discussed below.

#### 3.3.2.1 Humidity Distribution

Plants grow there roots in order to access the nutrients and moisture available in the surrounding soil. As roots of different plants overlap, they start to compete for these resources.

When distributing the soil humidity of a given grid cell $C_{xy}$ to the set $S = \{P_1, P_2, P_3, ...\}$ of plants which roots intersect the cell, one of three distinct scenarios can occur depending on the total available humidity of the cell: *Abundant humidity*, *sufficient humidity* and *insufficient humidity*.

The humidity is deemed abundant if the available humidity, $H_{available}$, surpasses 300 millimetres. In this situation, the humidity is deemed to be enough for there to be standing water and therefore all plants of $S$ are allocated $H_{available}$.

If $H_{available}$ is less than 300 millimetres, it is necessary to determine whether the humidity is sufficient or insufficient by calculating the requested humidity $H_{requested}$ as outlined in equation 6.9.

$$H_{requested} = \sum MinHumidity(P_n) \text{ for } n \in S \tag{3.9}$$

Where:*MinHumidity($P_n$)* is the minimum humidity requirement of the specie to which plant $P_n$ belongs;$S$ is the set of plants whose roots intersect with the given grid cell.

If $H_{requested}$ is less than $H_{available}$, the humidity is deemed sufficient and the amount allocated to each plant is calculated as described in equation 6.10. If $H_{requested}$ is more than $H_{available}$, however, the humidity is deemed insufficient and the allocation is done following equation 6.11

$$\begin{aligned} H_{allocated}(P_n) &= MinHumidity(P_n) + OverFlow \\ OverFlow &= H_{available} - \sum MinHumidity(P_n) \text{ for } n \in S \end{aligned} \tag{3.10}$$

Where:*$H_{allocated}(P_n)$* is the humidity allocated to plant $P_n$; *MinHumidity($P_n$)* is the minimum humidity requirement of the specie to which plant $P_n$ belongs;$S$ is the set of plants whose roots intersect with the given grid cell.

Intuitively, it allocates each plant with the minimum amount of humidity it requires to survive plus the resulting overflow.

$$\begin{aligned} H_{allocated}(P_n) &= min(MinHumidity(P_n), Vigor(P_n) \times H_{remaining}) \\ Vigour(P_n) &= \frac{RootSize(P_n)}{\sum RootSize(P_x) \text{ for } x \in S} \\ H_{remaining} &= H_{available} - (\sum H_{allocated}(P_x) \text{ for } x \in S_{processed}) \end{aligned} \tag{3.11}$$

Where: The plants of $S$ **must** be iterated over in decrementing order of their vigor as this will affect the water they are allocated; *$H_{allocated}(P_n)$* is the humidity allocated to plant $P_n$; *MinHumidity($P_n$)* is the minimum humidity requirement of the specie to which plant $P_n$ belongs; *Vigour($P_n$)* is the vigor of plant $P_n$ in comparison to other plants present in the cell. It is estimated based on root size; *RootSize($P_n$)* is the root size of plant $P_n$; $S_{processed}$) is the set of plants from $S$ whose water allocation has already been calculated; $S$ is the set of plants whose roots intersect with the given grid cell.

Intuitively, this algorithm prioritises water distribution to more vigorous plant's.

#### 3.3.2.2 Plant Humidity Allocation

To calculate the humidity allocated to plant $P_n$, it is first necessary to determine the set of grid cells $S = \{C_1, C_2, C_3, ...\}$ which it's roots intersect. Given this, the plants humidity allocation is calculated using equation 6.12.

$$H_n = \frac{\sum H_{allocated}(C_x) \text{ for } x \in S}{|S|} \tag{3.12}$$

Where:$H_n$ is the humidity allocated to plant $P_n$; *$H_{allocated}(C_n)$* is the humidity allocated to plant $P_n$ in grid cell $C_n$; $\mid S \mid$ is the number of cells in the set $S$.

Intuitively, the humidity allocated to a given plant is simply the average of the humidity allocated to it in all grid cells it's roots intersect.

#### 3.3.2.3 Illumination Distribution

Plants which are heavily dependent on illumination will often grow a large canopy in order to maximize the leaf area receiving direct sunlight. Doing so also restricts the illumination received by smaller plants in the undergrowth. To model the shade projection of larger plants, illumination is distributed in each cell depending on the plants height and canopy width.

Equation 6.13 is used to allocate illumination amongst the set $S = \{\mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3, ...\}$ of plants which canopy intersects cell $C_{xy}$.

$$Illumination(P_n) = \begin{cases} C_{illumination}, & \text{if } CanopyWidth(P_n) = 0 for x \in S \\ C_{illumination}, & \text{if } Height(P_n) > height(x) for x \in S : x \neq P_n \\ 0, & \text{otherwise} \end{cases} \tag{3.13}$$

Where: *Illumination($P_n$)* is the illumination allocated to plant $P_n$ whose canopy overlaps with current grid cell; $C_{illumination}$ is the monthly illumination of the cell; *CanopyWidth($P_n$)* is the canopy width of plant $\mathrm{P}_n$; *Height($P_n$)* is the height of plant $\mathrm{P}_n$; $S$ is the set of plants whose canopy intersects with the given grid cell.

Intuitively, if all plants present in the given cell are canopy-free (i.e no shade projection), the equation allocates them all the available illumination. If not all plants are canopy-free, the equation allocates illumination only to the tallest canopy plant.

#### 3.3.2.4 Plant Illumination Allocation

Calculating the illumination allocated to a plant $P_n$ is very similar to calculating the humidity allocation only the set of grid cells $S = \{\mathrm{C}_1, \mathrm{C}_2, \mathrm{C}_3, ...\}$ are those which the plants canopy intersects. Given $S$, the plant illumination is calculated using equation 6.14.

$$I_n = \frac{\sum I_{allocated}(C_x) \text{ for } x \in S}{|S|} \tag{3.14}$$

Where: $I_n$ is the illumination allocated to plant $P_n$; $I_{allocated}(C_n)$ is the illumination allocated to plant $P_n$ in grid cell $\mathrm{C}_n$; $\mid S \mid$ is the number of cells in the set $S$.

Intuitively, the illumination allocated to a given plant is simply the average of the humidity allocated to it in all grid cells it's canopy intersect.

### 3.3.3 Plant Strength Calculation

The strength of plant $P_n$ is the minimum of $S_{age}$, $S_{temperature}$, $S_{illumination}$ and $S_{humidity}$ which represent the strength of the plant in terms of it's age, temperature, illumination and humidity respectively. To calculate these values, ranging from negative to positive one hundred, a graph is plotted for each specie based on it's properties as outlined in

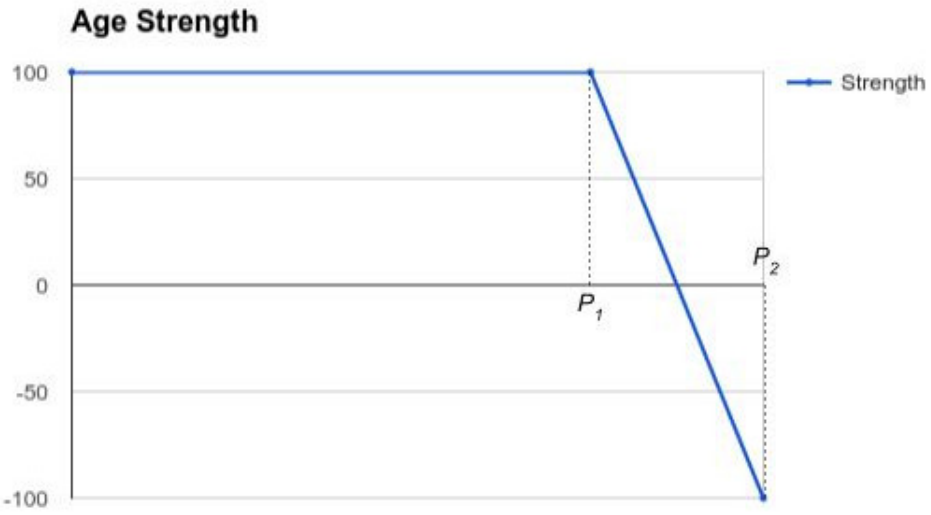



Figure 3.5: Graph used to calculate the age strength of any plant. $\mathbf{P}_1$ is the age of *start of decline* configured for the given specie. $\mathbf{P}_2$ is the *maximum age* configured for the given specie.

figures 6.5, 6.6, 6.6, 6.7 and 6.8. Using these graphs, it is possible to calculate the strength of any plant given it's allocated resources.

### 3.3.4 Plant Strength Usage

The strength of each plant is recalculated on a monthly basis as available resources change and other plants spawn and grow. This value subsequently influences its growth potential and probability of death, as discussed below.

#### 3.3.4.1 Growth Potential

In the simulation, each plant $P$ attempts to grow its roots, its canopy and it's height on a monthly basis. Each specie has an maximum monthly root growth $MaxGrowth_{root}$, canopy growth $MaxGrowth_{canopy}$ and height growth $MaxGrowth_{height}$ which are calculated using the specie's growth and ageing properties as outlined in equations 6.15, 6.16 and 6.17 respectively.

$$MaxGrowth_{root}(S) = \frac{MaxRoot(S)}{Age_{StartOfDecline}(S)} \tag{3.15}$$

Where:$MaxGrowth_{root}(S)$ is the maximum monthly root growth of specie $S$;$MaxRoot(S)$ is the configured maximum root size of the specie $S$;$Age_{StartOfDecline}(S)$ is the age of start of decline configured for specie $S$.

$$MaxGrowth_{canopy}(S) = \frac{MaxCanopy(S)}{Age_{StartOfDecline}(S)} \tag{3.16}$$

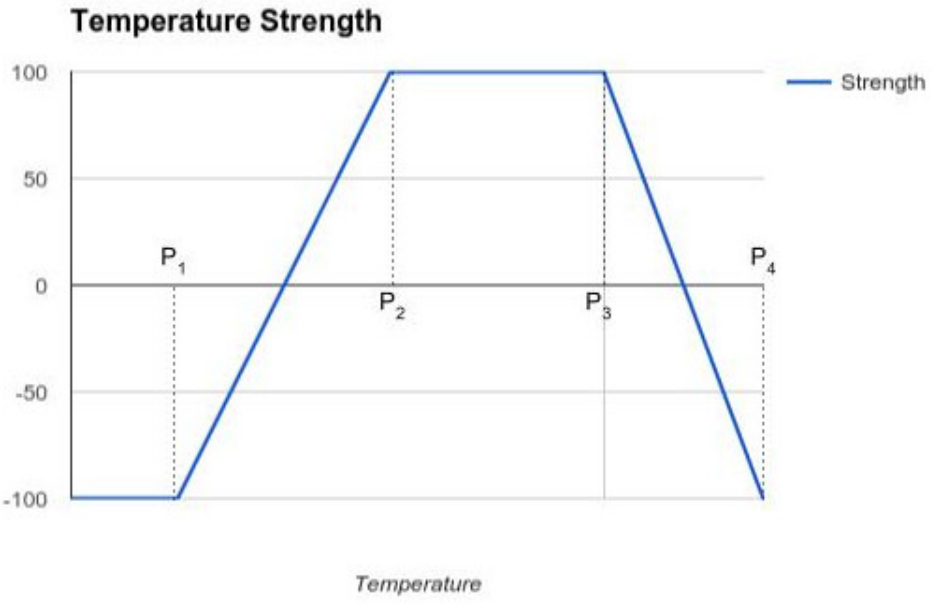



Figure 3.6: Graph used to calculate the temperature strength of any plant. $\mathbf{P}_1$ and $\mathbf{P}_4$ are the *minimum* and *maximum temperature* configured for the given specie. $\mathbf{P}_2$ and $\mathbf{P}_3$ form the *prime temperature range* configured for the given specie.

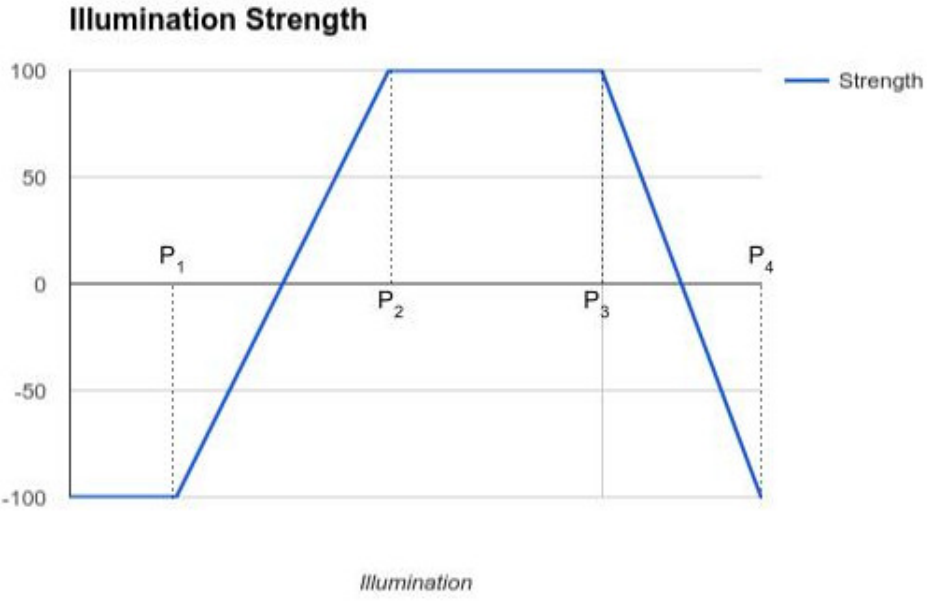



Figure 3.7: Graph used to calculate the illumination strength of any plant. $\mathbf{P}_1$ and $\mathbf{P}_4$ are the *minimum* and *maximum illumination* configured for the given specie. $\mathbf{P}_2$ and $\mathbf{P}_3$ form the *prime illumination range* configured for the given specie.

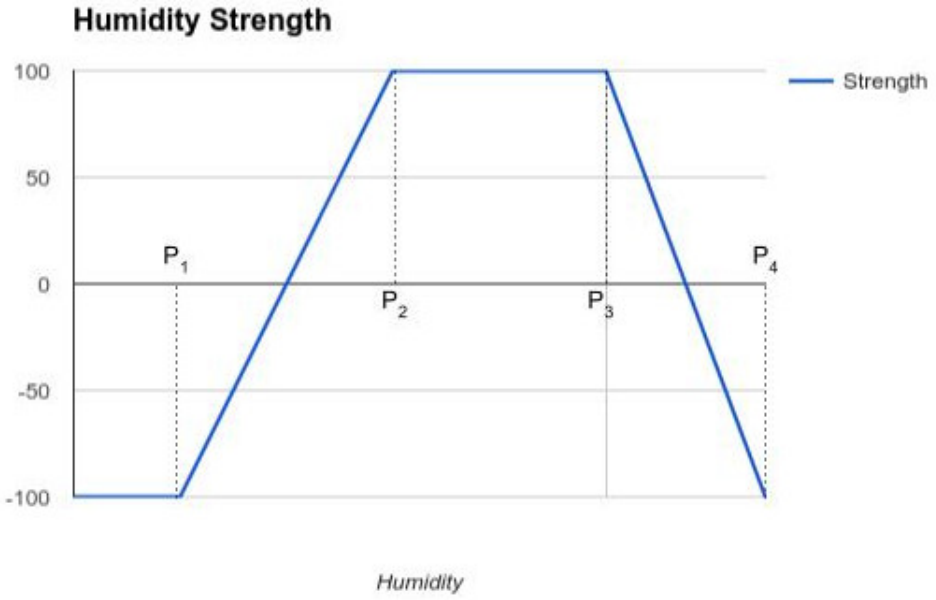



Figure 3.8: Graph used to calculate the humidity strength of any plant. $\mathbf{P}_1$ and $\mathbf{P}_4$ are the *minimum* and *maximum humidity* configured for the given specie. $\mathbf{P}_2$ and $\mathbf{P}_3$ form the *prime humidity range* configured for the given specie.

Where:$MaxGrowth_{canopy}(S)$ is the maximum monthly canopy growth of specie $S$;$MaxCanopy(S)$ is the configured maximum canopy size of specie $S$;$Age_{StartOfDecline}(S)$ is the age of start of decline configured for specie $S$.

$$MaxGrowth_{height}(S) = \frac{MaxHeight(S)}{Age_{StartOfDecline}(S)} \tag{3.17}$$

Where:$MaxGrowth_{height}(S)$ is the maximum monthly height growth of specie $S$;$MaxHeight(S)$ is the configured maximum height of specie $S$;$Age_{StartOfDecline}(S)$ is the age of start of decline configured for specie $S$.

The actual root growth $Growth_{root}$, canopy growth $Growth_{canopy}$ and height growth $Growth_{height}$ is directly dependent on the plant's strength, however, and is calculated using equations 6.18, 6.19 and 6.20 respectively. The maximum growth is achieved only if the plant is at its full strength. Note that no plants grow if there current strength is negative as they are deemed in a *survival state*.

$$Growth_{root}(P, S) = max(0, Strength(P) \times MaxGrowth_{root}(S) \tag{3.18}$$

Where:$Growth_{root}(S)$ is the monthly root growth of plant $P$ of specie $S$;$Strength(P$ is the current strength of $P$;$MaxGrowth_{root}(S)$ is the maximum monthly root growth calculated for specie $S$.

$$Growth_{canopy}(P, S) = max(0, Strength(P) \times MaxGrowth_{canopy}(S) \tag{3.19}$$

Where:$Growth_{canopy}(S)$ is the monthly canopy growth of plant $P$ of specie $S$;$Strength(P$ is the current strength of $P$;$MaxGrowth_{canopy}(S)$ is the maximum monthly canopy growth calculated for specie $S$.

$$Growth_{height}(P, S) = max(0, Strength(P) \times MaxGrowth_{height}(S) \quad (3.20)$$

Where:*Growth$_{height}$(S)* is the monthly height growth of plant $P$ of specie $S$;*Strength(P* is the current strength of $P$; *MaxGrowth$_{height}$(S)* is the maximum monthly height growth calculated for specie $S$.

#### 3.3.4.2 Probability of Death

On a monthly basis, the probability of death of each plant is calculated based on it's strength using equation 6.21 and the plant killed with the said probability. Note that a plant $P$ will only be susceptible to be killed off if it's strength is negative.

$$Probability_{death}(P) = max(0, \frac{-1 \times Strength(P) + counter}{100}) \quad (3.21)$$

Where:*Probabilitydeath(P)* is the probability of death of plant $P$;*Strength(P* is the current strength of $P$;*counter* is a value which increases by ten each month the plant's strength is negative and resets to zero when it becomes positive. This is to prevent plant's from surviving with a continuous negative strength for too long.

### 3.3.5 Spawning Plants

In nature, the spawning of new plants ensures specie *succession* and *propagation*. In order to accurately model the evolution of an ecosystem it is essential to replicate this spawning mechanism. To do so, seeds are produced annually for each specie and are positioned either randomly or at predefined positions. The number of seeds that are produced for a given specie is determined by the specie's *annual seed count* configuration.

Different seeding mechanisms are used in the simulator depending on the number of plant's of the given specie present in the simulation, $S_{count}$, and it's illumination properties.
If $S_{count}$ is greater than zero, existing plant instances are used to determine the location of new plants, irrespective of it's illumination requirements, as described in *Spawning from Existing Plants*.
If $S_{count}$ is zero, the seeding mechanism depends on the specie's illumination requirements. If the specie is shade loving, *canopy seeding* is employed, else *random seeding*. Both are discussed below.

#### 3.3.5.1 Spawning from Existing Plants

To ensure specie propagation, when plant's of the given specie are already present in the simulation window, they are used to determine the location for new plant instances. To do so, $n$ of these plants are selected at random and seeds placed at random within an annular radius $r$ of each. The value of $n$ is the *annual seed count* configured for the current specie. The value of $r$ is the configured *maximum seeding distance* of the specie. Note that if the number of plants of the given specie is less than the number of seeds to produce, a single plant is used to produce multiple seed locations.

This technique effectively ensures *propagation* until the number of plant instances present is larger than the number of seeds to produce. At which point, the *propagation* potential decreases as the plant count increases. This is because as the selection pool for the random plants increases in size, the probability of selecting a plant at a location which will permit propagation decreases. To overcome this and ensure the initial seeding plants that are selected span a wide area of the simulation window, they are selected at from individual simulation grid cells.

#### 3.3.5.2 Canopy Seeding

Shade-loving plants strive in the shaded undergrowth. If shade-loving plants are spawned at random locations in the simulation, the probability of them landing under the canopy of an existing plant is extremely low. To overcome this, shade-loving plants are not spawned at random but under the canopy of randomly selected plants.

#### 3.3.5.3 Random Seeding

This is the most simple form of seeding and is used when no plants of the given specie are present in the simulation and the specie is not shade-loving. The set of locations for new plants to spawn is selected at random within the simulation window.

### 3.3.6 Performance

The number of plants present in the simulation will heavily influence it's performance as the strength of each plant needs to be recalculated on a monthly basis. To test the influence of plant count on simulation time, a simulation is run with a single specie of plant and the monthly processing time analysed alongside the number of plants present. The plant used is grass at is has no canopy and very minimal root coverage, permitting a large number of instances to grow simultaneously (see appendix B for properties of specie). The resources were set to be optimal for maximizing plant count and minimizing intra-plant competition. The simulation is started with only a single instance and, as the simulation progresses and seeding is performed, the number of instances increase. The results are summarized in Figure 6.9 and show that the processing time increases linearly with plant count.

Another simulation property which heavily impacts performance is the root and canopy growth of plant species present. The reason for this is that, as plant's roots and canopy grow, they will cover more simulation grid cells and more calculations will be required per individual cell when the contained resources are distributed. To analyse the impact of plant growth, a base specie $S_{base}$ is created with a given root and canopy growth rate. Then, two species are created $S_{X2}$ and $S_{X3}$ with twice and thrice the growth rates of $S_{base}$ respectively (see appendix B for specie details). An identical simulation is run on each specie in terms of available resources and, on a monthly basis, the number of plants present in the simulation along with the monthly processing time analysed. Given this information, it is possible to track the average monthly processing time per plant throughout the simulation. It is important to normalise based on the number of plants

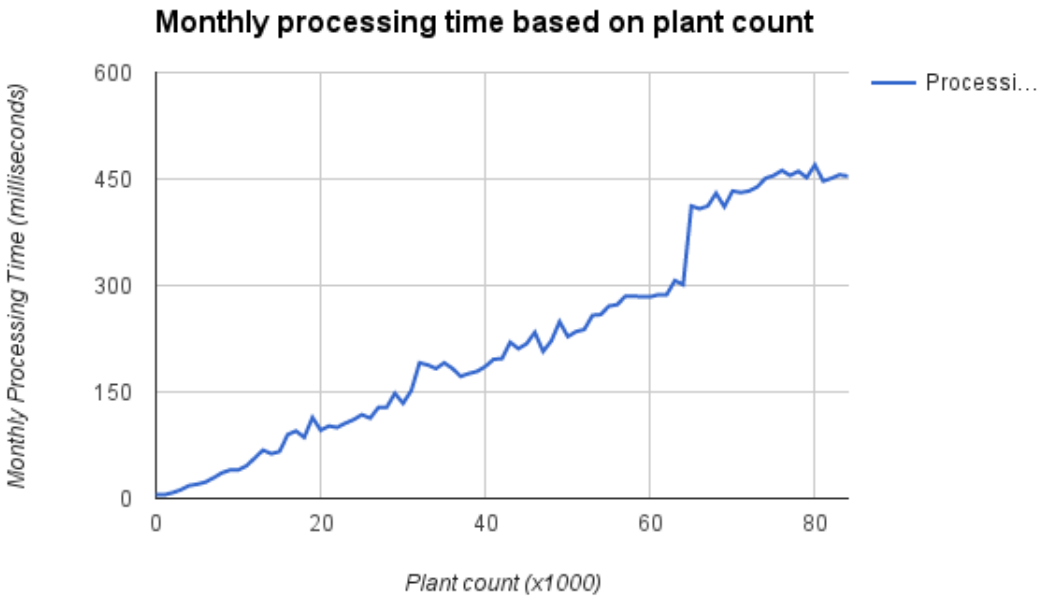



Figure 3.9: Processing time based on plant count. Total simulation time for 100 years: 271 seconds

as the faster growing plants will permit less plants to survive for the simple reason that they will require and be able to access resources from a larger amount of grid cells. As can be seen in the results plotted in figure 6.10, the processing times are similar to start and then increase proportionally to the species growth rate.

### 3.3.7 Results

To test the resulting spatial distribution of plant communities in their work, Lane and and Przemyslaw [LP02] attempt to reproduce three important properties of nature: *Self-thinning*, *succession* and *propagation*.
As 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.
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. This is the *succession* property.
Plants *propagate* in clusters surrounding the seeding plant.

To test the ecosystem simulator, we will first employ the same methodology as Lane and and Przemyslaw [LP02] and attempt to reproduce *self-thinning*, *succession* and *propagation*.

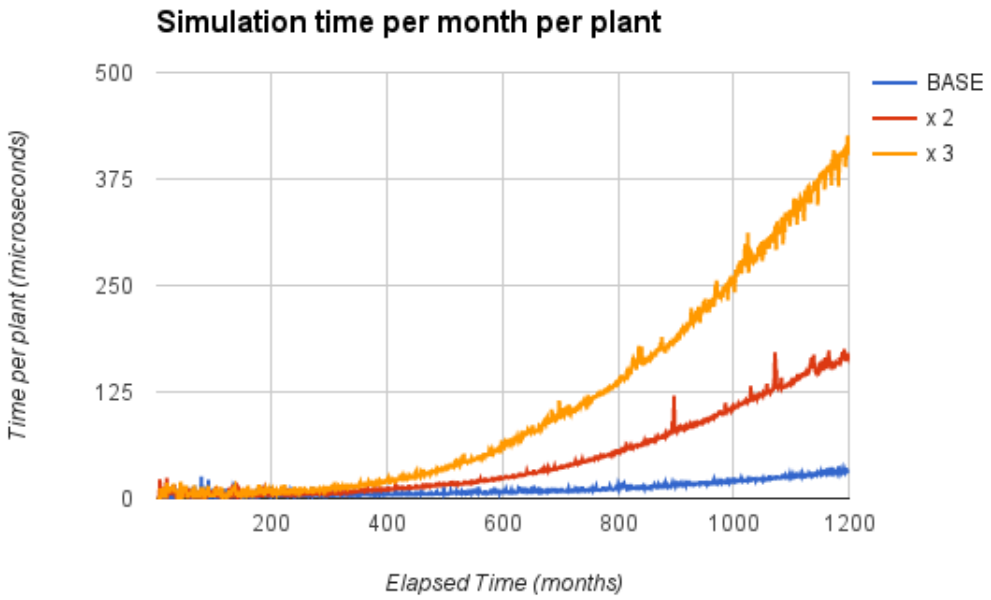



Figure 3.10: Evolution of the monthly processing time normalised based on plant count. The processing time increases as the plant's grow larger as they cover more grid cells. Total simulation time for one hundred years: 49 seconds for $S_{base}$, 122 seconds for $S_{X2}$ and 166 seconds for $S_{X3}$

To ensure a given plant specie strives better in its optimal environment, the same simulation will be run using a single plant specie with varying resource properties and the resulting plant distribution analysed. This test is discussed in *Varying Resources Test* below.

A *shade-simulation test* is performed to ensure plants which depend on direct illumination strive less in the shaded undergrowth of larger plants.

To ensure shade-loving plants are able to propagate and strive under the canopies of larger plants, a *shade-loving plant test* is also performed.

#### 3.3.7.1 Self-thinning Test

*Self-thinning* occurs when the plant biomass surpasses a given tipping point where available resources become insufficient to permit further plant growth. At this point, larger plants start to kill off smaller, weaker plants by stealing their resources.

To test whether self-thinning is successfully modelled in the ecosystem simulator, three simulations are run as described in table 6.1 and the plant count tracked throughout. As described previously, self-thinning occurs because of insufficient resources. By modifying only available humidity in each simulation, it's effect on self-thinning becomes apparent. As can be seen in the results summarized in figure 6.11, the plant count increases at first,

| Simulation | Simulation time (years) | Humidity | Illumination | Temperature |
|---|---|---|---|---|
| 1 | 100 | 25 | 10 | 15 |
| 2 | 100 | 30 | 10 | 15 |
| 3 | 100 | 35 | 10 | 15 |

Table 3.1: Self-thinning test simulation configurations. For simplicity, monthly resources are kept constant.

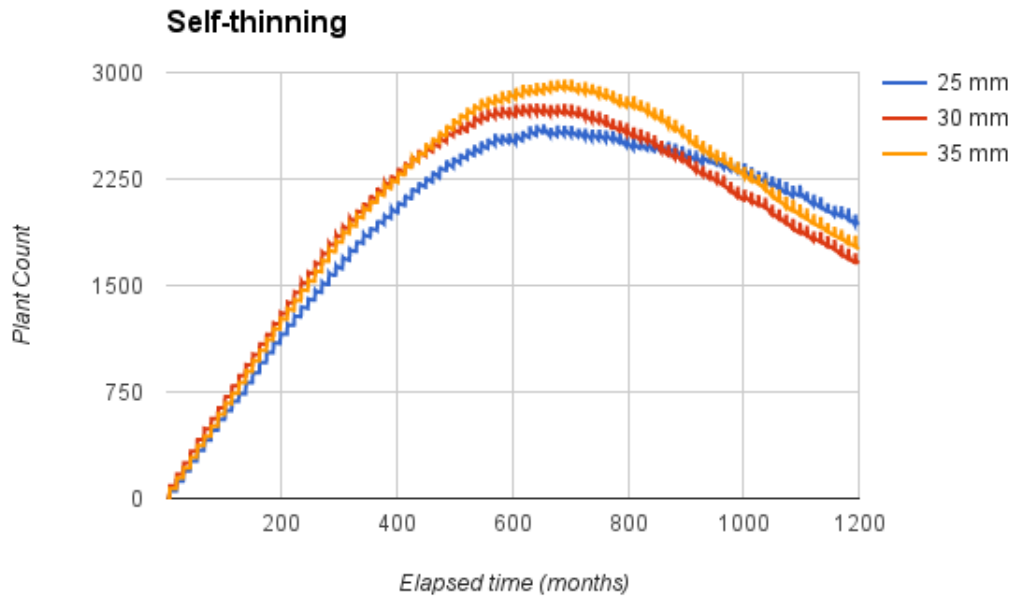



Figure 3.11: Plant count tracked throughout three separate simulations differing only in available humidity.

reaches a maximum and decreases thereafter. This is the exact behaviour of self-thinning. Furthermore, it is apparent that the maximum plant count increases with the humidity available, therefore showing that self-thinning is sensitive to available resources.

#### 3.3.7.2 Succession Test

Succession occurs in an ecosystem due to the different growth rates of the species it contains. To test *succession* in the ecosystem simulator, two plant species $S_{fast}$ and $S_{slow}$ are created differing only in their growth rate and illumination properties (see appendix B for details) and a simulation run with these two species under optimal conditions. During the simulation, the appearance and average size of the two plant species are monitored to determine the dominating specie. The analytical results illustrated in figure 6.12 along with the appearance at ten year intervals displayed in figure 6.13 shows that $S_{fast}$ dominates at first ( 300 months in) followed by $S_{slow}$ ( 500 months in). A balance is found thereafter.

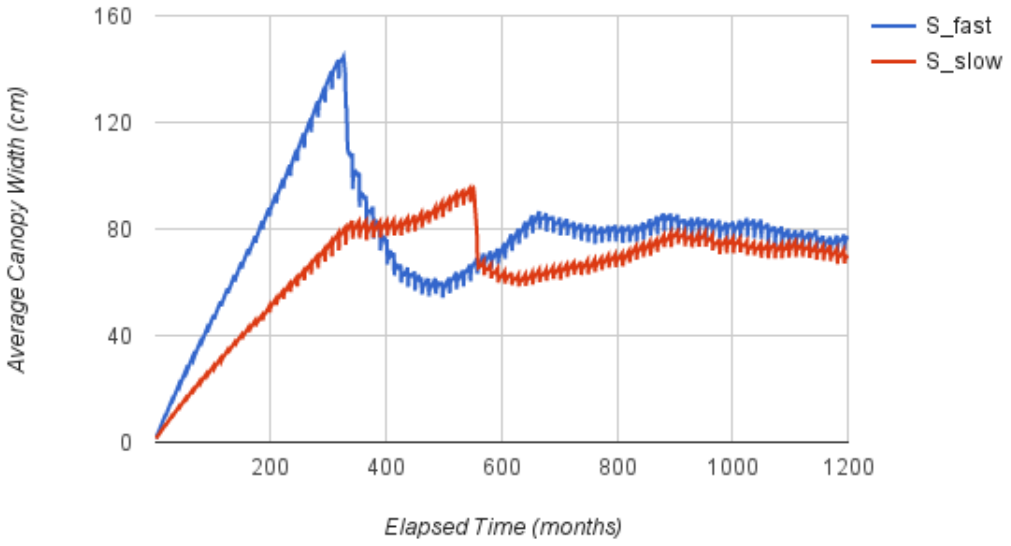



Figure 3.12: Succession Test: Average size of the slow growing $S_{slow}$ (red) and fast growing $S_{fast}$ (blue) throughout a simulation run in optimal conditions.

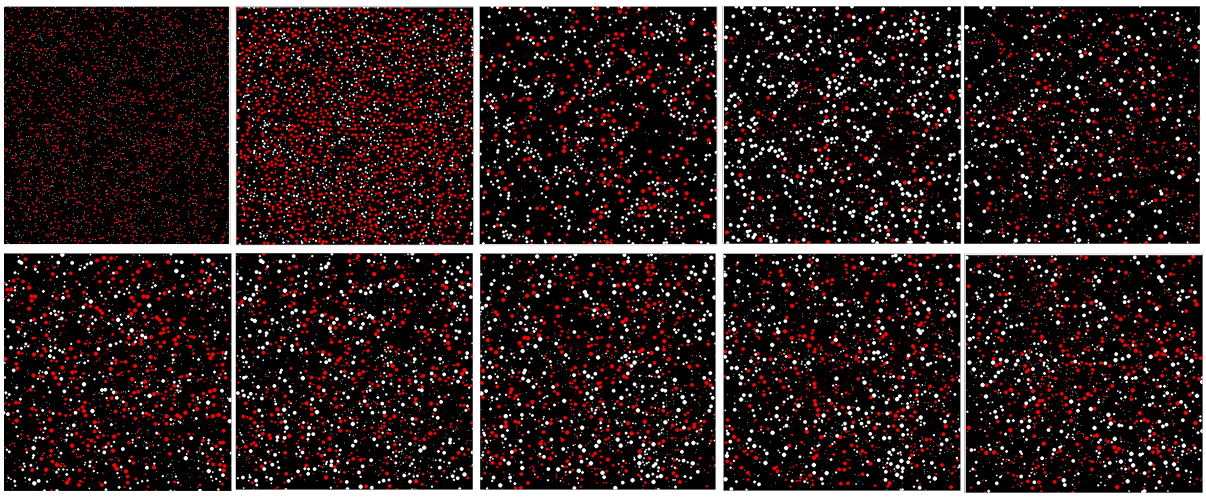


Figure 3.13: Succession Test: Appearance of the slow growing $S_{slow}$ (white) and fast growing $S_{red}$ (blue) at different times during the simulation. From left-to-right, top-top-bottom: 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100 years.

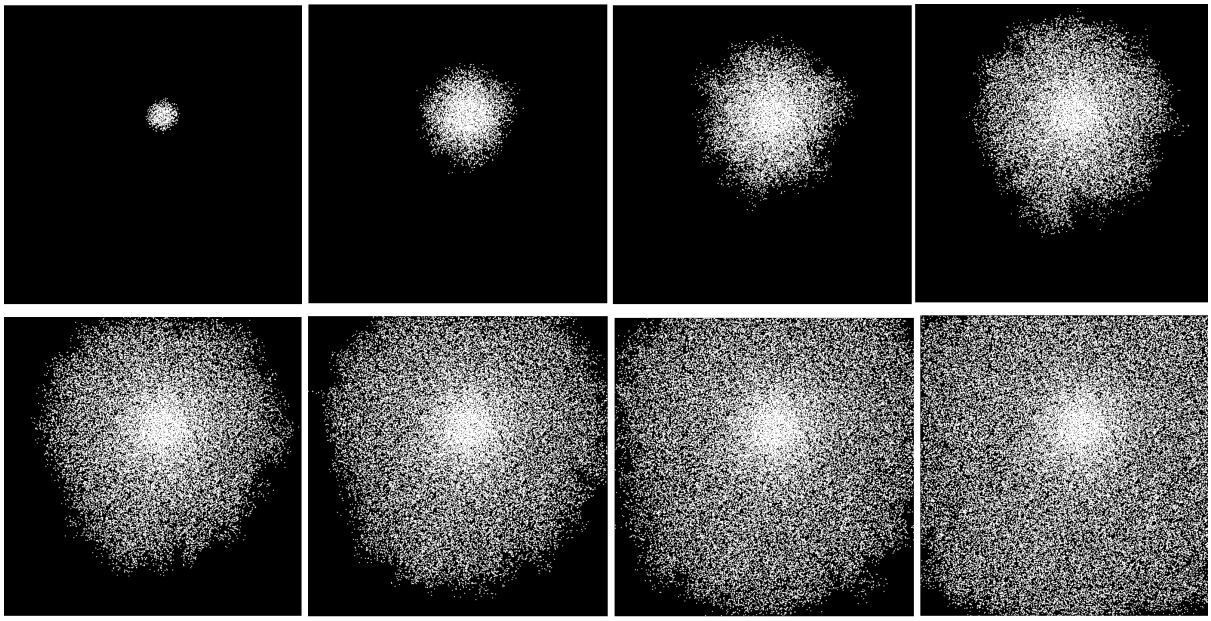


Figure 3.14: Propagation Test: Evolution through time of a simulation starting from a single seed plant of grass. From left-to-right, top-to-bottom: 2, 10, 20, 30, 40, 50, 60, 70 years in.

#### 3.3.7.3 Propagation Test

To ensure propagation is modelled in the ecosystem simulator a simulation is run with a single starting grass seed (see appendix B for specie details) and it's evolution tracked throughout. Figure 6.14 shows that iterative propagation through annual seeding enables a single seed plant to colonize the entirety of the terrain.

#### 3.3.7.4 Varying Resource Test

To ensure a given plant specie strives better when it's habitat is more suited, multiple simulations are run with only specie $S_{base}$ (see appendix B for specie properties) present. As outlined in table 6.2, the simulations vary only in their configured humidity. As seen by the results plotted in figure 6.15, illustrating the average canopy width for a single plant instance throughout the simulations, plants strive better in environments better suited to their resource requirements.

#### 3.3.7.5 Shade Test

Plant's that are heavily dependent on illumination struggle to grow in areas shaded by the canopy of larger plants. To test this is modelled in the ecosystem simulator, a simulation is run with two species: $S_{smallroots}$ and grass (see appendix B for specie details). $S_{smallroots}$ is a custom specie created for the purpose of this test which has very small roots growth. This is important so as to focus on the effects of illumination and minimize the influence of drought. Figure 6.16, which illustrates the state of the simulation after fifty years, shows the grass struggling to grow in areas directly below the canopies of $S_{smallroots}$.

| Simulation | Simulation time (years) | Humidity | Illumination | Temperature |
|---|---|---|---|---|
| 1 | 100 | 22 | 10 | 20 |
| 2 | 100 | 24 | 10 | 20 |
| 3 | 100 | 26 | 10 | 20 |
| 4 | 100 | 28 | 10 | 20 |
| 5 | 100 | 30 | 10 | 20 |
| 6 | 100 | 32 | 10 | 20 |
| 7 | 100 | 34 | 10 | 20 |
| 8 | 100 | 36 | 10 | 20 |
| 9 | 100 | 38 | 10 | 20 |

Table 3.2: Varying resource rest simulation configurations. For simplicity, monthly resources are kept constant.

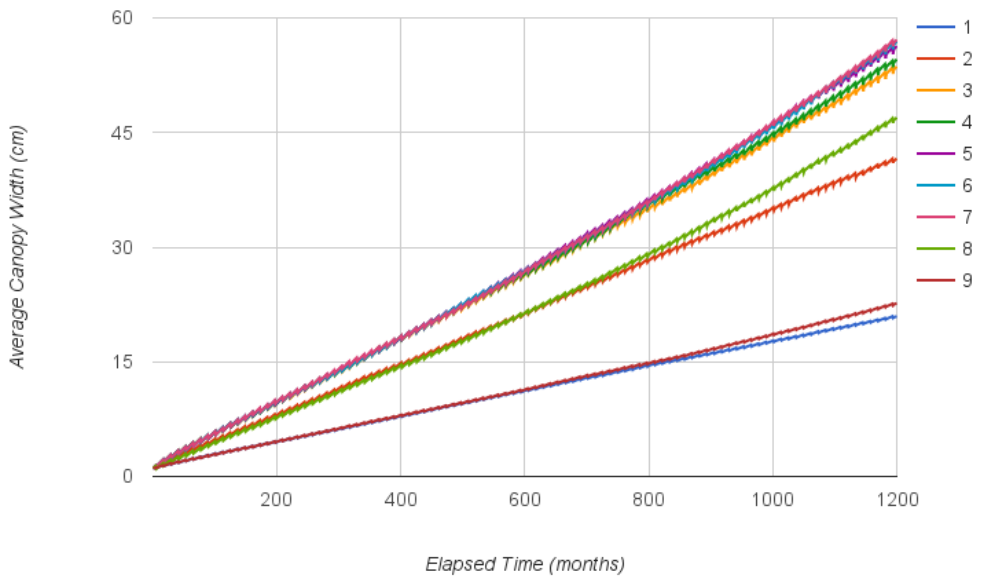



Figure 3.15: Varying Resource Test: Average canopy width of a single plant throughout the simulations outlined in table 6.2.

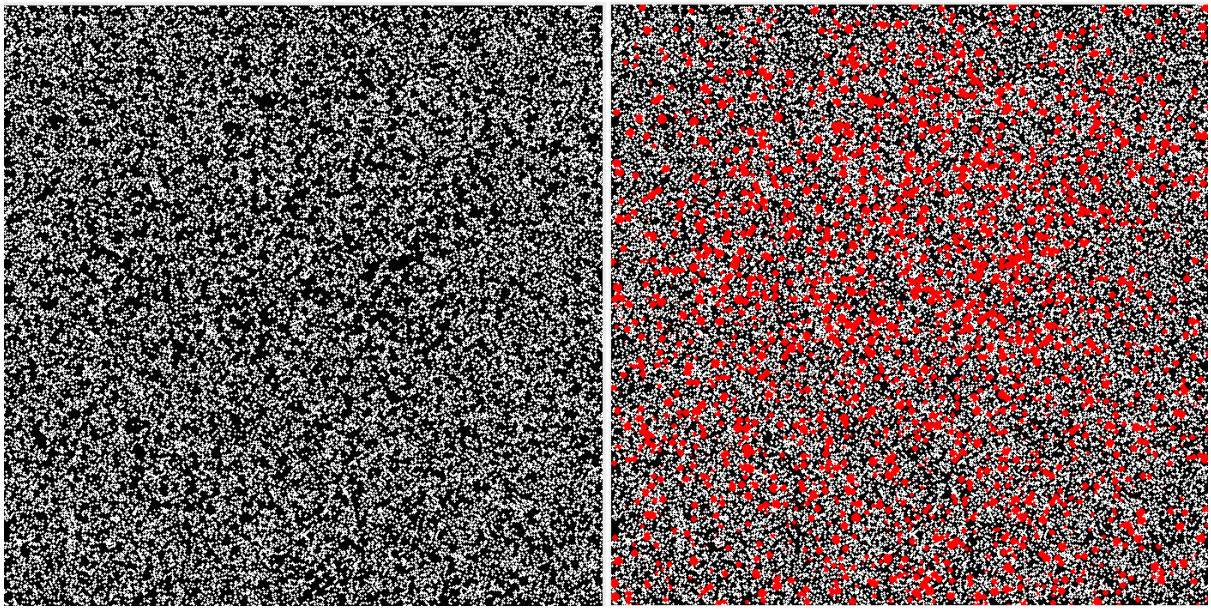


Figure 3.16: Shade Test: Simulation with $S_{smallroots}$ (red), grass (white) after fifty years. Left without rendering instances of $S_{smallroots}$ to visualise the effects of the shade

#### 3.3.7.6 Shade-loving Test

Species which strive in shaded areas and struggle to survive in open spaces directly illuminated by the sun are deemed to be shade-loving. The shade can be caused by the terrain relief or by the shadow cast by the canopy of taller plants. To test whether the ecosystem simulator successfully caters for such plant species, a simulation is run identical to that done in the shade test (section 6.3.7.5) but with shade-loving specie $S_{shadeloving}$ added (see appendix B for specie details). As seen by the snapshot of the simulation after fifty years illustrated in figure .., instances of $S_{shadeloving}$ only appear in areas directly covered by the canopies of $S_{smallroots}$.

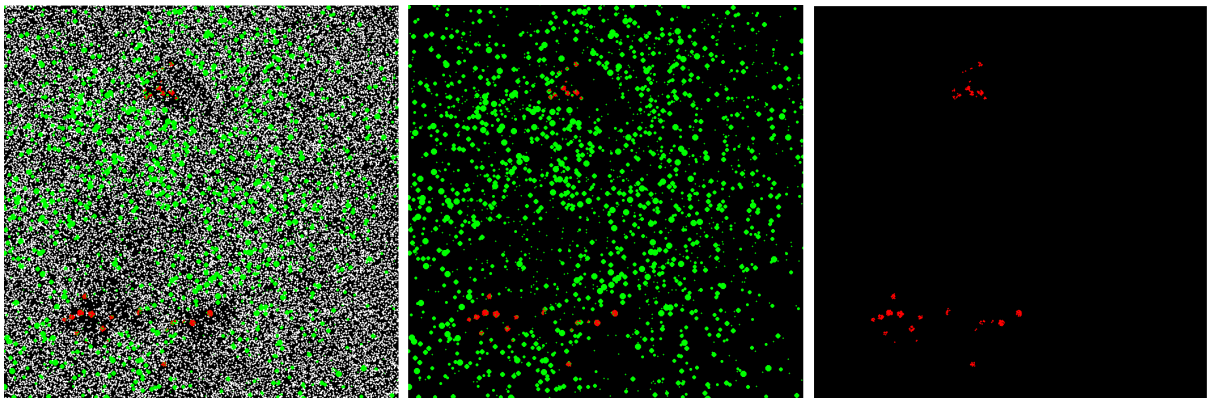


Figure 3.17: Shade Loving Test: Simulation with $S_{smallroots}$ (green), grass (white) and $S_{shadeloving}$ after fifty years. From left-to-right: All species, excluding grass and only $S_{shadeloving}$. As can be seen, $S_{shadeloving}$ strive under the canopies of $S_{smallroots}$.

## 3.4 Plant Distribution Analysis and Reproduction

As mentioned previously, running a simulation using the ecosystem simulator to generate valid plant distributions can be a lengthy process (see section 6.3.6). This processing time depends on the plant count and the resolution of the simulation window. The simulation illustrated in figure 6.9, for example, took four and a half minutes to run.

The area to be covered by vegetation on the terrain, and therefore for which valid plant distributions created, is much larger than the hundred by hundred metre simulation window used in the ecosystem simulator. There are three obvious ways the ecosystem simulator could be used to generate vegetation for larger areas: *Setting the simulation window to the area which must be covered*, *decreasing the resolution of the simulation window* and by *repeating the output of the hundred by hundred metre distribution (tiling)*. Each come with major setbacks, however, as *increasing the simulation* window will further increase the processing time, *decreasing the resolution* would impact the resulting realism and *tiling* would create repetitive vegetation.

To attempt to generate vegetation which is coherent with that produced with the ecosystem but on a larger scale whilst minimizing impacts on performance and realism, the plant distribution output by the ecosystem simulator is analysed for later reproduction. The analysis and reproduction process are discussed in *Analysis* and *Reproduction* respectively.

The analysed distribution data, as well as permitting larger scale reproduction, can also be used as a cache mechanism to bypass future runs of the same ecosystem simulator run. Details on how this is implemented are discussed in *Caching Distribution Data*.

To conclude this section, input exemplars will be analysed and the resulting reproduction(s) evaluated.

### 3.4.1 Distribution Analysis

*Radial distribution analysis*, as described in section 2.2.2.1, is performed on the output of the ecosystem simulator to grasp its core characteristics. Each plant instance acts as a single point and the different species represent the individual categories when performing the analysis. Customizations to the generic analysis algorithm are performed, however, to better suit the purpose of analysing plant distributions. Each are discussed separately below in: *Generating the category hierarchy*, *Category Dependency Analysis* and *Point-size Analysis*. In *Configuration Parameters* are discussed the parameters used. To conclude, the performance of the distribution will be discussed in *Performance*.

#### 3.4.1.1 Generating the category hierarchy

During the reproduction phase, distributions for each category are created sequentially. One a valid distribution is created for a given category, it is static and **does not** change whilst points of other categories are being plotted. For this reason, the category hierarchy plays a vital role and has a big impact on the final distribution. A side effect of this hierarchical approach is that pair-correlation histograms do not need to be generated for

each category pair combinations but only for combinations for which the target category is under or equal to the source category in the hierarchy.

Because taller plants will potentially have a canopy which shades and influences the position of smaller plants, it is important these be generated first during reproduction. For this reason, the hierarchy is generated according to the average height of the represented plant specie in descending order.

#### 3.4.1.2 Category Dependency Analysis

Shade-loving plants will appear under the the shaded canopies of taller plants (see section 6.3.7.6). When analysing the distance of these shade-loving plants to the plant's which shade them during the analysis phase, a new *negative-bin* is created.

During reproduction, the taller plants will be generated first because they are classed higher in the hierarchy (see section 6.4.1.1). However, this does not guarantee all shade-loving plants will be placed in the shaded canopy of other plants as if a shade-loving plant is placed at a distance larger than $R_{max}$ to any other plant instance, it is attributed a strength of one and is therefore deemed valid. It is essential to attribute a strength of one in such condition to permit plant propagation. A solution to this problem would be to make $R_{max}$ large enough to cover the entire simulation window. This is extremely wasteful in terms of computational resources however and, as such, another solution is used here; When the pairwise histograms have been generated for a given category $A$, they are analysed sequentially to check whether or not all instances appear within the *negative-bin* of the other category (specie). The specie $A$ is deemed **dependent** on all categories for which this is true and, during reproduction, will have to be placed within the radius of one of them to be deemed valid.

#### 3.4.1.3 Point-size Analysis

As well as the position of individual plants, an important property of the ecosystem simulator output is plant size. In order to reproduce appropriately sized plants, this must also be analysed. To do so, the *minimum* and *maximum* canopy radius and height for each category are also analysed.

#### 3.4.1.4 Configuration Parameters

The *radial distribution analysis* requires the following configuration parameters: $R_{min}$, $R_{min}$ and *bin-size*. Details on each parameter can be found in section 2.2.2.1.

Increasing the analysis range [$R_{min}$, $R_{max}$] and decreasing the *bin-size* will impact performance but potentially increase the accuracy of the analysis and finding optimal values for these parameters depends on the properties of the points being analysed.
It is unnecessary to have too large of an analysis range as the impact a plant has on it's surrounding is finite. This impact radius varies however and is dependent on specie size. In order to cater for different species of different sizes and therefore with different impact radii, $R_{max}$ is dynamic and limited to **two metres** passed the extremity of the plant's

canopy.
The bin-size doesn't influence performance as severely as the analysis range, however, as it has no impact on the number of points that need to be processed. Smaller bin sizes will result in less points being processed per bin and, therefore, a less accurate representation of the distribution variation with distance. Because smaller bins will result in a smaller number of points, also, the analysis will be more sensitive to noise. A bin size of **twenty centimetres** is used as it strikes a good balance between accuracy and point count per bin.

#### 3.4.1.5 Performance

In order to generate the necessary analysis data, each point (plant) must be iterated over and the distance measured from it to all other points within a radius of $R_{max}$. As a consequence, the analysis time is directly correlated to plant density and, therefore, plant count within the hundred metre analysis window. To determine the correlation between plant density and analysis time, test distributions are generated of various densities using the ecosystem simulator which are subsequently analysed and the time to do so, measured. Because the number of histograms that need to be generated depends on the number of categories to be analysed, one would easily assume that the processing time is also correlated to the category count. This is **not** the case however and only point density influences analysis performance. To demonstrate this, the various plant densities are generated containing one, two and three distinct categories.The results plotted in figure 6.18 indicate an exponential correlation between plant count and processing time and a quicker analysis time when points are split into multiple categories. Although the correlation is exponential, the most extreme scenario (over ninety thousand points of a single category) is processed in a manageable time of just under two seconds.

### 3.4.2 Reproduction

The purpose of the analysed radial distribution data is to later use it to reproduce similar distributions on larger scales. To do so, the same reproduction technique described in section 2.2.2.1 is employed with slight nuances described below, namely: *Matched-density initialization* and *Iterative point moving*.

The radial distribution data generated to test the analysis performance above (section 6.4.2.3) is used to test the reproduction performance.

#### 3.4.2.1 Matched Density Initialization

Rather than employ a birth-and-death technique like that described in section 2.2.2.1, where a point is added, the aggregate strength of the distribution calculated and the new point accepted with a calculated probability, matched-density initialization is employed. This involved generating a fixed number of points for each category so that their density matches that of the analysed density. The only requirement is for the aggregate strength of the distribution to be non-zero. In other words, the distribution does not need to be strongly matched, but valid.

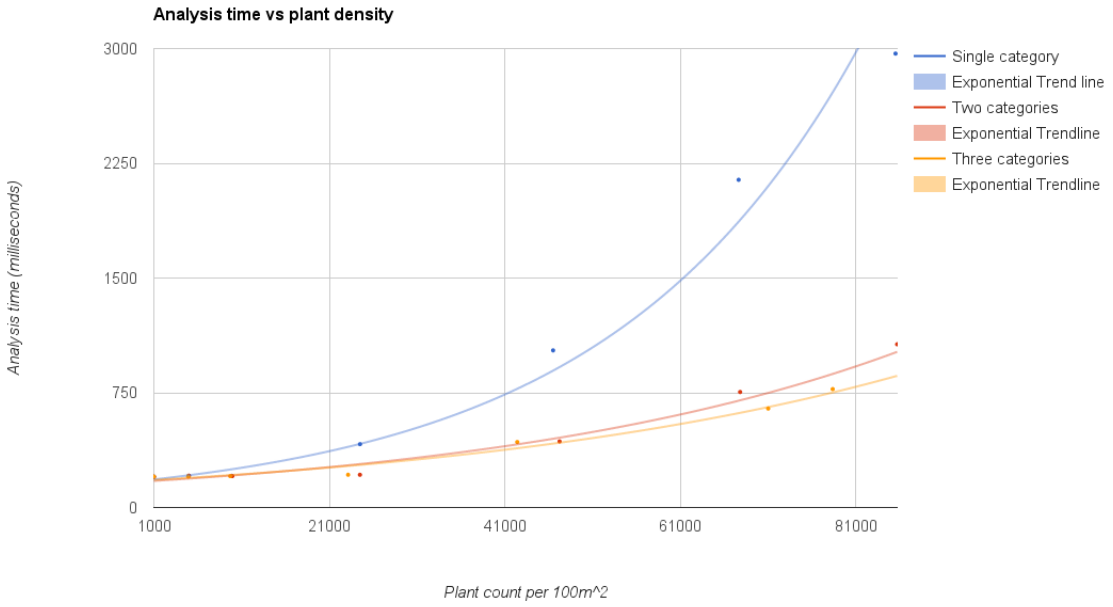



Figure 3.18: Distribution analysis time based on aggregate plant density for single category (blue), two categories (red) and three categories (yellow).

#### 3.4.2.2 Iterative Point Moving

When points of a given category have been initialized and the required density reached, iterative point-moving is performed where each added point is iterated over and moved to two random locations. The new distribution strength is calculated after each move and the best scoring move is accepted with probability $P_{acceptance}$, calculated using equation 6.22.

$$P_{acceptance} = \frac{Strength_{n+1}}{Strength_n} \tag{3.22}$$

Where:$Strength_{n+1}$ is the aggregated distribution strength after the move; $Strength_n$ is the aggregated distribution strength before the move.

#### 3.4.2.3 Performance

The reproduction area and point density are two properties which greatly affect performance. Although both effect the number of points to reproduce and, therefore, the reproduction time, because an increase in density will lead to more points within $R_{max}$ of any given source point, given a fixed plant count, performance should increase with area. To determine to what extent and the correlation between plant density, reproduction area and performance, test reproductions are performed using the analysis data generated when performance testing the analysis stage (section ).

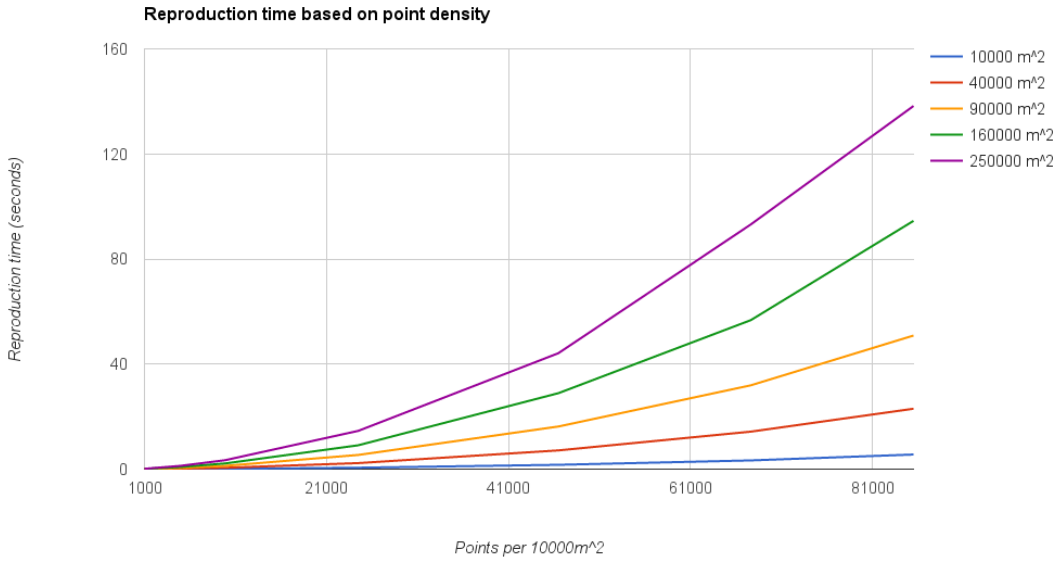



Figure 3.19: Reproduction time based on point density for different reproduction areas.

As seen in figure 6.19 which plots the reproduction performance based on point density for various reproduction areas, the correlation between plant density and reproduction time is exponential. It also shows the increase to be more accentuated when reproducing larger areas. This is expected, however, as larger areas will require more points to be added in order to meet the required density.

Using the test data generated for figure 6.19, it is possible to plot the reproduction time based solely on plant count for various densities (see figure 6.20). It shows the correlation between plant count and reproduction time to be linear and dependent on point density. The reason it is sensitive to plant density is because the denser the points are the the more of them will be within a distance of $R_{max}$ and therefore need ot be taken into consideration when calculating the strength of the distribution. In order to keep reproduction times manageable, the reproduced plant count is limited to half a million. If large areas need to be reproduced with a plant count higher than this limit, repeating/tiling is performed. Appendix C outlines the maximum reproduction areas that can be achieved for different plant densities. Although the repetition (tiling) performed will increase for denser distributions, it will not necessarily be more noticeable as denser distributions will tend to have less distinct patterns and be more closely correlated to random.

### 3.4.3 Caching Distribution Data

In order to prevent repeated costly runs of the ecosystem simulator for identical resource parameters, the analysed distribution data is stored and tracked in a database. This way, if a plant distribution is requested for a simulation which has already been run,

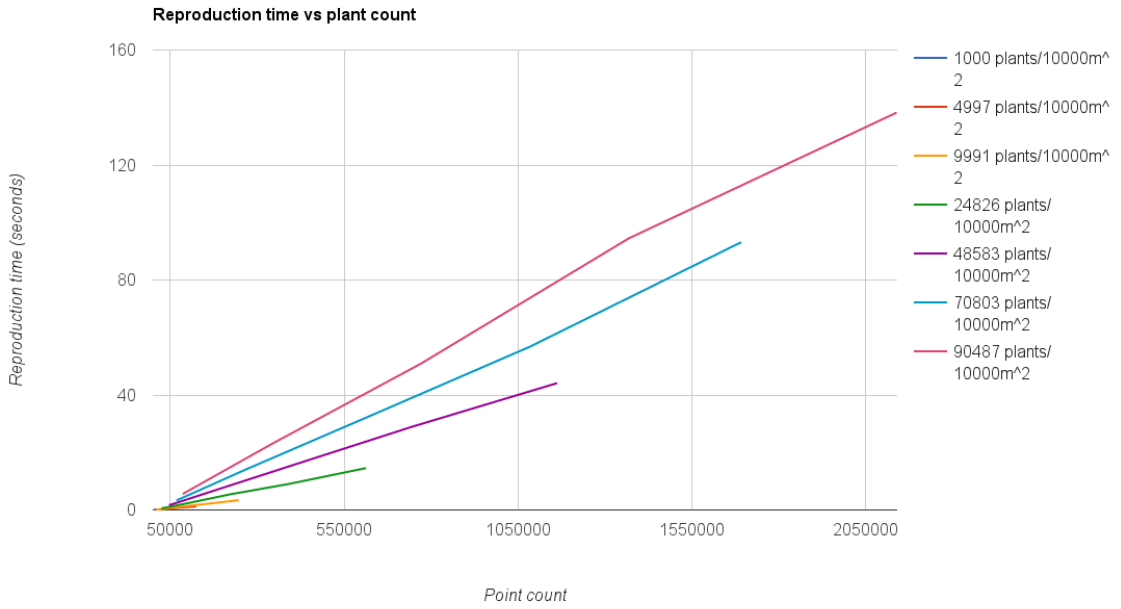



Figure 3.20: Reproduction time based on point count for different densities.

the ecosystem simulator is bypassed entirely and the stored distribution data used. A custom binary file format is used in order to save space when storing the necessary analysed distribution data.

### 3.4.4 Results

The important properties of the input exemplars which must be reproduced are: *inter and intra specie separation*, *plant size* and *specie densities*. To ensure these are accurately reproduced, an ecosystem simulator run is performed containing *shade-loving*, *shade intolerant* and *canopy plants*. The resulting plant distribution is subsequently used as input exemplar to stress test the distribution analyser and reproducer. Figure 6.21 shows an overview and zoomed subsection of the input exemplar along with it's associated reproduction. From this, along with the point count of individual species, it is possible to conclude that point density and point size is accurately replicated. To determine whether intra and inter-specie spacing is accurately reproduced, the reproduction distribution is re-analysed in order to produce the pair correlation histograms of the reproduced distribution. The original and reproduced histograms are then compared to ensure they follow similar trends (see figure 6.22). Important properties to note which are accurately reproduced are:

- No plants appear within the radius of the shade-loving (category 6) and shade-intolerant plants (category 5)
- The density of shade-intolerant plants (category 5) drastically decreases within the radius of canopy plants (category 9).

- The density of shade-loving plants (category 6) drastically decreases within the radius of canopy plants (category 9). Note that in both the original and reproduction, all shade-loving plants appear within the radius of a canopy plants as the categories are dependent (see section 6.4.1.2). The variation in histogram values between the two is caused by the positioning of other nearby canopy plants.

- The density decreases drastically for canopy plants within the canopy of other canopy plants (category 9) as it blocks access to available illumination.

Note that the pair correlation histogram of the shade-loving specie with itself (category 6 and 6) is substantially different between original and reproduction. The reason for this is because during the ecosystem simulator, seeding is performed from existing plant instances which will lead to the clustering of plants under the same canopy. During iterative point-moving (see section 6.4.2.2, the probability of shade-loving plants to cluster under neighbouring canopy plants is very slim, however, and dispersion under all present canopy plants is much more likely.

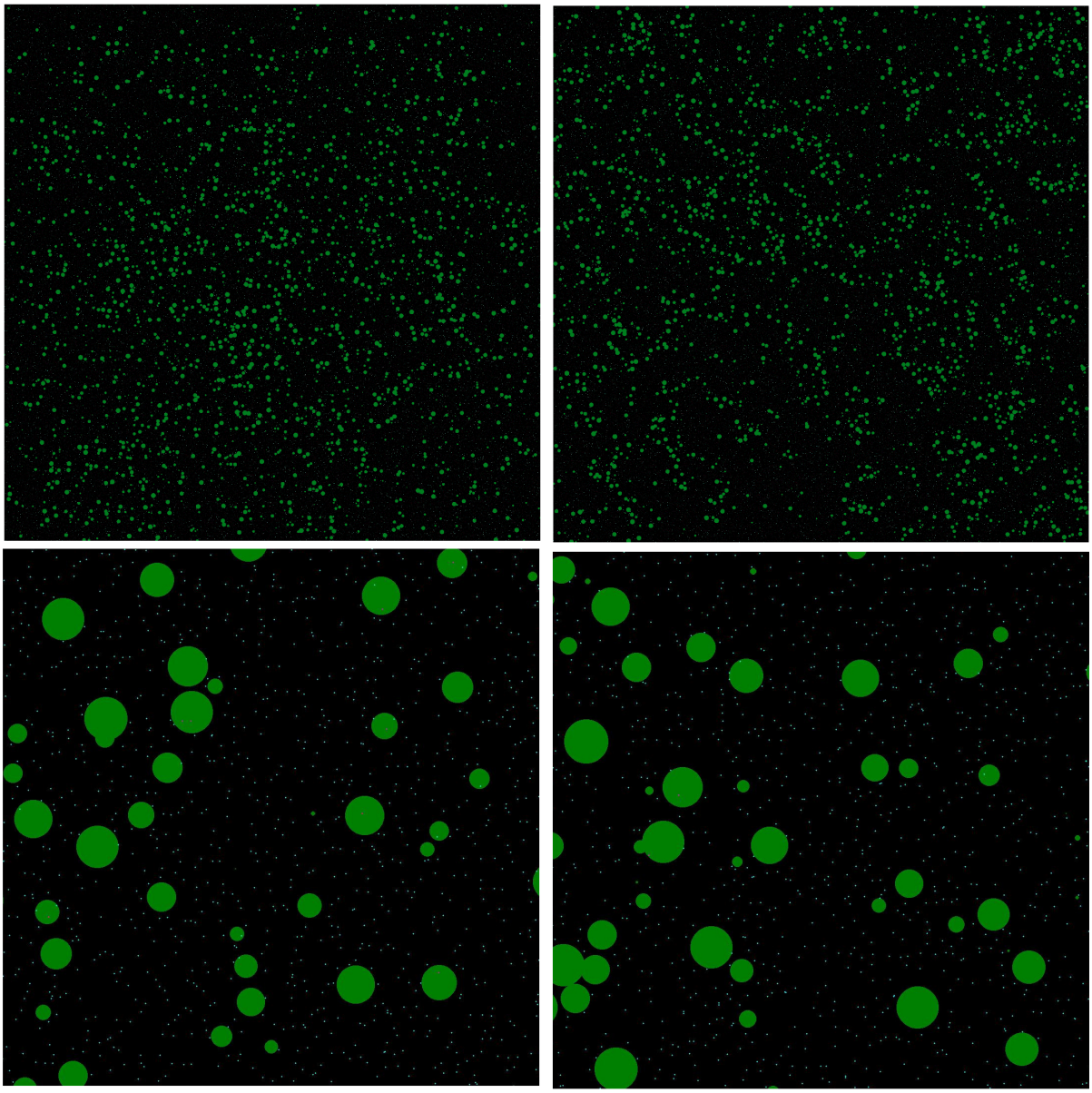


Figure 3.21: Distribution analysis and reproduction test: Input exemplar overview (top-left), reproduction overview (top right), zoomed input exemplar (x 10), zoomed reproduction (x 10).

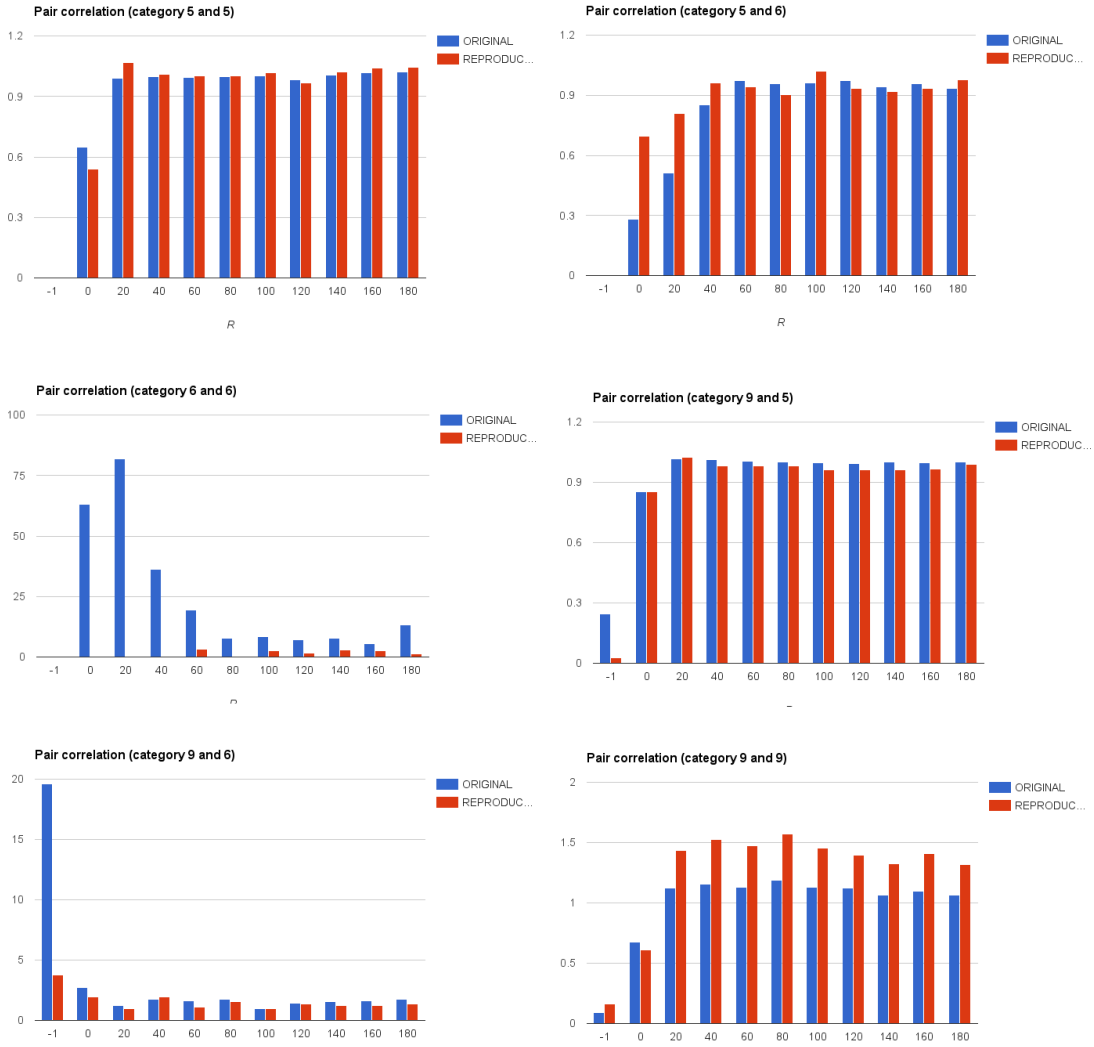



Figure 3.22: Original (blue) and reproduced (red) pair correlation histograms for different bins where category 6 is a shade-loving, category 5 is shade intolerant and category 9 is a canopy specie. Bin sizes of -1 signify the target category is within the radius of the source.

# Appendices

# Appendix A

# Cluster Summary

| | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| Color | | | | | |
| Member count | 314415 (30%) | 225868 (22%) | 225943 (22%) | 9201 (0.8%) | 273149 (26%) |
| Slope | 6.65549 | 15.5914 | 16.2722 | 19.8204 | 46.483 |
| Temperature (Jan) | 19 | 23 | 21 | 25 | 23 |
| Temperature (Feb) | 17 | 21 | 19 | 23 | 21 |
| Temperature (Mar) | 14 | 18 | 16 | 20 | 18 |
| Temperature (Apr) | 12 | 16 | 14 | 18 | 16 |
| Temperature (May) | 9 | 13 | 11 | 15 | 13 |
| Temperature (Jun) | 7 | 11 | 9 | 13 | 11 |
| Temperature (Jul) | 9 | 13 | 11 | 15 | 13 |
| Temperature (Aug) | 12 | 16 | 14 | 18 | 16 |
| Temperature (Sep) | 14 | 18 | 16 | 20 | 18 |
| Temperature (Oct) | 17 | 21 | 19 | 23 | 21 |
| Temperature (Nov) | 19 | 23 | 21 | 25 | 23 |
| Temperature (Dec) | 22 | 26 | 24 | 28 | 26 |
| Illumination (Jan) | 8 | 5 | 6 | 3 | 4 |
| Illumination (Feb) | 9 | 6 | 7 | 4 | 4 |
| Illumination (Mar) | 9 | 7 | 8 | 5 | 5 |
| Illumination (Apr) | 10 | 8 | 9 | 6 | 6 |
| Illumination (May) | 11 | 9 | 10 | 6 | 7 |
| Illumination (Jun) | 12 | 9 | 10 | 7 | 7 |
| Illumination (Jul) | 11 | 9 | 10 | 6 | 7 |
| Illumination (Aug) | 10 | 8 | 9 | 6 | 6 |
| Illumination (Sep) | 9 | 7 | 8 | 5 | 5 |
| Illumination (Oct) | 9 | 6 | 7 | 4 | 4 |
| Illumination (Nov) | 8 | 5 | 6 | 3 | 4 |
| Illumination (Dec) | 6 | 4 | 5 | 2 | 3 |
| Soil Humidity (Jan) | 24.7 | 26.8 | 23.2 | 711.4 | 2.2 |
| Soil Humidity (Feb) | 24.7 | 26.8 | 23.2 | 711.4 | 2.2 |
| Soil Humidity (Mar) | 24.7 | 26.8 | 23.2 | 711.4 | 2.2 |
| Soil Humidity (Apr) | 24.7 | 26.8 | 23.2 | 711.4 | 2.2 |
| Soil Humidity (May) | 24.7 | 26.8 | 23.2 | 711.4 | 2.2 |
| Soil Humidity (Jun) | 24.7 | 26.8 | 23.2 | 711.4 | 2.2 |
| Soil Humidity (Jul) | 24.7 | 26.8 | 23.2 | 711.4 | 2.2 |
| Soil Humidity (Aug) | 24.7 | 26.8 | 23.2 | 711.4 | 2.2 |
| Soil Humidity (Sep) | 24.7 | 26.8 | 23.2 | 711.4 | 2.2 |
| Soil Humidity (Oct) | 24.7 | 26.8 | 23.2 | 711.4 | 2.2 |
| Soil Humidity (Nov) | 24.7 | 26.8 | 23.2 | 711.4 | 2.2 |
| Soil Humidity (Dec) | 24.7 | 26.8 | 23.2 | 711.4 | 2.2 |

Table A.1: Clustering test: Cluster summary.

# Appendix B

# Specie Properties

| **Name** | **Grass** | **S**$_{base}$ | **S**$_{X2}$ | **S**$_{X3}$ | **S**$_{slow}$ | **S**$_{fast}$ | **S**$_{smallroots}$ | **S**$_{shadeloving}$ |
|---|---|---|---|---|---|---|---|---|
| **Max Height (cm)** | 60 | 1500 | 1500 | 1500 | 1500 | 1500 | 1500 | 50 |
| **Max canopy width (cm)** | 0 | 1000 | 2000 | 3000 | 2000 | 2000 | 3000 | 0 |
| **Max root size (cm)** | 20 | 1000 | 2000 | 2000 | 2000 | 2000 | 50 | 30 |
| **Age of start of decline (months)** | 8000 | 1000 | 1000 | 1000 | 500 | 300 | 1000 | 1000 |
| **Maximum age (months)** | 9000 | 2000 | 2000 | 2000 | 600 | 350 | 2000 | 2000 |
| **Start of prime illumination (h)** | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 0 |
| **End of prime illumination (h)** | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 4 |
| **Minimum illumination (h)** | 5 | 6 | 6 | 6 | 3 | 6 | 6 | 0 |
| **Maximum illumination (h)** | 15 | 12 | 12 | 12 | 12 | 12 | 12 | 6 |
| **Start of prime humidity (mm)** | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 10 |
| **End of prime humidity (mm)** | 45 | 35 | 35 | 35 | 35 | 35 | 35 | 25 |
| **Minimum humidity (mm)** | 10 | 15 | 15 | 15 | 15 | 15 | 15 | 5 |
| **Maximum humidity (mm)** | 60 | 45 | 45 | 45 | 45 | 45 | 45 | 40 |
| **Start of prime temperature (degrees)** | 15 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| **End of prime temperature (degrees)** | 25 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| **Minimum temperature (degrees)** | 0 | -5 | -5 | -5 | -5 | -5 | -5 | -10 |
| **Maximum temperature (degrees)** | 40 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
| **Maximum seeding distance (m)** | 3 | 50 | 50 | 50 | 50 | 50 | 50 | 3 |
| **Annual seed count** | 1000 | 100 | 100 | 200 | 100 | 100 | 100 | 1000 |

# Appendix C

# Maximum Distribution Reproduction Areas

| **Points per 10000m$^2$** | **Maximum reproduction area (m$^2$)** |
|---|---|
| 5000 | 1000000 |
| 10000 | 500000 |
| 15000 | 333333 |
| 20000 | 250000 |
| 25000 | 200000 |
| 30000 | 166666 |
| 35000 | 142857 |
| 40000 | 125000 |
| 45000 | 111111 |
| 50000 | 100000 |
| 55000 | 90909 |
| 60000 | 83333 |
| 65000 | 76923 |
| 70000 | 71428 |
| 75000 | 66666 |
| 80000 | 62500 |
| 85000 | 58823 |
| 90000 | 55555 |
| 95000 | 52631 |
| 100000 | 50000 |

Table C.1: Maximum reproduction area for given plant densities

# Appendix D

# Machine Specifications

| | |
|---|---|
| **CPU** | Intel Core i7-4770 CPU @ 3.40GHz |
| **GPU** | Nvidia GeForce GTX 770 |
| **RAM** | |
| **Storage** | 1TB HDD + 60GB SDD |

Table D.1: Specifications of the machine used for all performance testing.

# Bibliography


[ACV+14] 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, feb 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.

[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.

[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, jan 2012.

[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.

[DGGK11] 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, jun 2010.

[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.

[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.

[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, apr 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.

[GMS09] James Gain, Patrick Marais, and Wolfgang Straßer. Terrain sketching. *Proceedings of the 2009 symposium on Interactive 3D graphics and games - I3D '09*, 1(212):1–8, 2009.

[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.

[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.

[Lew99] Philip Lewis. Three-dimensional plant modelling for remote sensing simulation studies using the Botanical Plant Modelling 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, jul 2001.

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

[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.

[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.

[ŠBBK08] Ondej Šťava, Bedich Beneš, Matthew Brisbin, and Jaroslav Kivánek. Interactive terrain modeling using hydraulic erosion. *EG CA - EuroGraphics Symposium on Computer Animation*, 2008.

[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.

[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.

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

[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.

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