**UNIVERSITAT
JAUME•I**

Procedural Generation of Natural Environments

Ph.D. THESIS: CRISTINA GASCH GARCIA

DIRECTOR: DR. MIGUEL CHOVER SELLÉS

CASTELLÓ, JANUARY 2021

Programa de Doctorado en Informática

Escuela de Doctorado de la Universitat Jaume I

PROCEDURAL GENERATION OF NATURAL ENVIRONMENTS

GENERACIÓN PROCEDURAL DE ENTORNOS NATURALES

Memoria presentada por Cristina Gasch Garcia para optar al grado de
Doctor por la Universidad Jaume I

Cristina Gasch Garcia

Miguel Chover Sellés

Castelló de la Plana, Enero 2021

Funding

This work has been supported by the Spanish Ministry of Science and Technology (Projects TIN2016-75866-C3-1-R, TIN2016-75866-C3-1-R y PID2019-106426RB-C32) and by research projects of the University Jaume I (UJI-B2018-56 and UJI-B2018-44).

Resumen

Los entornos naturales son una parte muy importante en la industria de los videojuegos, simuladores y realidad virtual. En los últimos años, la demanda de entornos de gran tamaño y con mayor realismo va en aumento. El principal problema para obtener estos tamaños de entorno reside en tener que crearlos manualmente, ya que esto implica una inversión de tiempo demasiado alta. La generación automática es una solución a este problema, pero este tipo de generación puede llevar al desarrollador a perder el control del resultado final. Por otro lado, otro problema que presentan los entornos naturales de gran tamaño es la gran cantidad de información que incluyen. Un bosque puede estar formado por miles de árboles, que a su vez incluyen otros elementos como hojas, tallos y ramas. Una visualización realista de este tipo de escenas, implica el procesamiento de una cantidad excesiva de información que impide su uso en aplicaciones en tiempo real. En los últimos años varios autores han publicado posibles soluciones a estos problemas, aunque todavía quedan un gran número de problemas por resolver, como los que se abordan en esta tesis.

El objetivo principal de esta tesis ha sido presentar diferentes soluciones a los problemas que plantea la generación procedural controlada de entornos naturales y su visualización interactiva. En cuanto al primer problema, se presenta un nuevo método de generación procedural controlada de terrenos basado en ruido. A continuación, se presenta un método de distribución procedural de elementos naturales,

a partir de varias restricciones biológicas dadas. Por otro lado, para facilitar la visualización de los entornos, se presenta una técnica de simplificación basada en información mutua de los puntos de vista. Esta técnica se ha diseñado especialmente para simplificar el follaje de los elementos naturales. Y por último, a partir de esta técnica, se ha desarrollado un modelo multirresolución que permite modificar de forma continua y en tiempo real el nivel de detalle de los elementos naturales.

Objeto y objetivos de la investigación

El área de estudio de gráficos por computador se centra en generar imágenes visuales de forma sintética con la ayuda de un ordenador. Estas imágenes sintéticas se obtienen a partir de algoritmos y los elementos que componen la escena. Entre estos elementos, se pueden encontrar las texturas, la iluminación y todos los objetos que la forman. Los objetos pueden estar formados por geometría sencilla como podría ser un cubo o más compleja si se trata de representar árboles o terrenos. Para poder modelar estos elementos más complejos como un entorno natural completo, existen sistemas de modelado para ayudar en la tarea. Pero aun así, cuando se necesita que estos tengan un gran tamaño se requiere de una inversión de tiempo excesivo.

Los métodos de generación procedural de contenido tienen como objetivo principal crear contenido de forma automática mediante algoritmos, aportando aleatoriedad a la generación sin necesidad de la intervención humana. De forma que se pueda obtener mayor cantidad de contenido en menor tiempo que en la generación manual. Pero el mayor problema de este tipo de generación reside en que cuanto más complejo sea un algoritmo mayor facilidad hay en que los resultados escapen al control del desarrollador, de forma que estos lleguen a ser poco realistas y se requiera de la intervención manual para arreglarlos. Este problema afecta directamente a la generación de entornos naturales, ya que una generación sin ningún tipo de control

puede obtener resultados con cambios de alturas realmente bruscos, superposición de elementos y requerir interacción humana al final del proceso para poder utilizar los resultados obtenidos.

En los últimos años se han realizado muchos estudios para dar control a la generación procedural en entornos. Muchos de estos trabajos, pese a que se ha automatizado gran parte del proceso en la generación, siguen requiriendo de un nivel demasiado alto de intervención humana. En el caso de aquellos que no requieren ninguna intervención, en algunos de ellos el nivel de control obtenido sigue siendo insuficiente para adaptar los entornos a unos requisitos precisos y en otros, pese a conseguir buenos resultados, se requiere de demasiado trabajo previo por parte del usuario o incluso tener un coste computacional excesivo.

En esta tesis se proponen varios métodos de generación procedural. Por un lado, se propone un método que permite la generación del terreno mediante una función de ruido que combina datos de altura dados como requisitos con datos aleatorios. De esta forma, se consigue un terreno realista gracias al aspecto pseudoaleatorio característico de la naturaleza, pero que incluye todas las restricciones dadas con gran precisión. Además, al tratarse de una función de ruido, el coste para obtener el terreno resultante es muy bajo. Por otro lado, el segundo método propuesto distribuye sobre el terreno obtenido elementos vegetales respetando los requisitos biológicos de altura, pendiente y vecindad de las especies a distribuir. Una vez distribuidos, todos estos elementos crecen respetando el tamaño máximo que puede alcanzar cada especie, maximizando la cobertura del terreno.

Aunque se considera que la generación de un entorno natural realista es importante, en el área de estudio de los gráficos por computador, la visualización en tiempo real tiene como objetivo generar imágenes lo más rápido posible para utilizarlas en aplicaciones interactivas. Renunciando con ello en muchas ocasiones a la calidad de las mismas, repercutiendo en el realismo obtenido. Esto es debido a que cuanto

más detallado sea el modelado geométrico, mayor será el realismo obtenido, pero también aumentará el coste de visualización. Para reducir este problema, se han realizado muchos estudios sobre el nivel de detalle de los modelos, con el objetivo de reducir la geometría de los elementos en función de su importancia en la escena. Para aplicar estos cambios en la geometría de los objetos de forma automática y evitar un cambio brusco se utilizan las técnicas de modelado llamadas técnicas de multirresolución.

La mayoría de los trabajos publicados que utilizan estas técnicas se han enfocado a mallas formadas por superficies continuas, las cuales permiten reducir su complejidad poligonal con un impacto visual mínimo comparándolo con el modelo original. Sin embargo, la parte del follaje de los elementos naturales no está formada por mallas continuas, sino por polígonos independientes que representan cada hoja. En este caso, los modelos multirresolución mencionados generalmente no funcionan correctamente, ya que tienden a eliminar las hojas por completo dejando huecos en el follaje, lo que produce un gran impacto visual. Algunos trabajos surgidos en los últimos años han presentado modelos de simplificación para el follaje de los árboles, que consiguen reducir el impacto visual causado por los otros modelos aunque el realismo de los resultados todavía puede ser mejorado.

El modelo multirresolución propuesto en esta tesis se ha creado utilizando como secuencia de eliminación de hojas, la generada por un algoritmo de simplificación basado en el punto de vista. De esta forma, este modelo permite la simplificación mediante la poda de las hojas, priorizando aquellas que permanecen ocultas desde cualquier punto de vista. Además, mantiene la densidad del follaje mediante el escalado de las hojas cercanas proporcional a la poda realizada y actualiza el nivel de detalle del elemento en tiempo real, evitando así los cambios bruscos en el follaje.

Planteamiento y metodología utilizados

Esta tesis tiene como objetivo la presentación de métodos que permitan la generación de un entorno natural controlado y la optimización de su visualización en tiempo real. Para ello se han realizado las siguientes tareas:

Estado del arte

Antes de definir una línea de trabajo para resolver los objetivos de esta tesis conviene hacer un estudio en profundidad de los trabajos existentes relacionados con lo que se quiere resolver. Una vez hecho el estudio se han evaluado las ventajas e inconvenientes de los trabajos existentes en la literatura y se ha dado una visión de las oportunidades de mejora apropiadas. En el capítulo 2 se presenta el análisis detallado de las siguientes áreas: representación de terrenos, generación de entornos naturales, técnicas de optimización de geometría y modelos multirresolución de follaje.

Propuestas y desarrollo de métodos para la generación controlada y optimización de entornos naturales

Se ha desarrollado un método para la generación procedural controlada de terrenos, un método para distribuir proceduralmente elementos naturales con restricciones, una técnica de simplificación del follaje de los elementos naturales y un modelo multirresolución continuo basado en la técnica de simplificación anterior.

Evaluación y comparación de los resultados

Para la evaluación y comparación de las ideas desarrolladas se han utilizado las herramientas necesarias, tales como motores de juego

y sistemas de cómputo numérico. Los resultados obtenidos han sido analizados y comparados para su validación.

Aportaciones originales

Las técnicas y métodos propuestos en la tesis han permitido realizar aportaciones en el campo de la creación y representación interactiva de entornos naturales complejos, que vienen avaladas por las diferentes publicaciones científicas que se han realizado durante su desarrollo. A continuación, se describen con detalle:

En el capítulo 3 se propone una solución con bajo coste computacional, para facilitar la generación de terrenos de forma automática a partir de un conjunto de restricciones iniciales. Este método basado en una función de ruido permite usar rutas, puntos de interés o áreas para definir valores de altura que se mantendrán en el resultado final. El terreno obtenido incluye todas estas restricciones con gran precisión, manteniendo la continuidad con el resto de valores de altura aleatorios y un aspecto realista.

Para distribuir los elementos naturales en un terreno se necesita que esta se haga teniendo en cuenta las características biológicas de las especies. En el capítulo 4 se presenta un método que permite esta distribución asistida por el usuario, teniendo en cuenta la altura y la pendiente a la que cada especie puede crecer. Además, el método hace crecer cada elemento hasta las medidas reales máximas que pueden alcanzar sin que se cree superposición de las especies, maximizando además, la cobertura del terreno. El método se ha diseñado de forma que en un futuro puedan ampliarse las características biológicas restrictivas de forma sencilla.

La representación realista de bosques de gran tamaño ofrece muchos problemas de eficiencia que dificultan su visualización en tiempo real. Para resolver este problema, en el capítulo 5 se presenta una técnica

de simplificación que utiliza como métrica de error la información mutua entre varios puntos de vista. De esta forma, el orden de poda empieza por las hojas ocultas y termina en las hojas que producen mayor impacto visual al ser eliminadas. Además, en cada proceso de poda, la hoja más cercana se escala en proporción al error cometido, priorizando así la apariencia visual del modelo y su realismo.

Por otro lado, aunque los modelos multirresolución discretos pueden atenuar los cambios bruscos que se producen cuando un elemento se cambia por su versión simplificada, el cambio del número de hojas en el follaje puede causar igualmente un impacto visual. Por tanto, la solución que se presenta en el capítulo 6 es un modelo multirresolución continuo. Es decir, cada vez que la importancia en la escena de un elemento varía, se obtiene el nivel de simplificación requerido a partir de la estructura de datos que define el modelo, actualizando en tiempo real el nivel de detalle del elemento.

Conclusiones obtenidas y futuras líneas de investigación

Del estudio planteado en el capítulo 2 se concluye que existen numerosas técnicas que aportan mejoras en diferentes aspectos, pero todas ellas incluyen desventajas que sugieren oportunidades de mejora. En esta tesis se propone como objetivo utilizar estas oportunidades de mejora para proponer nuevas técnicas para resolver los problemas propuestos.

En el capítulo 3 se presenta un nuevo método de modelado procedural basado en ruido para generar terrenos que combina datos de altura aleatorios con datos restrictivos. El terreno resultante es forzado a pasar por todas estas restricciones con un alto grado de precisión, aumentando su aleatoriedad a medida que se aleja de ellas. El resultado obtenido por el método demuestra que los terrenos resultantes incluyen de forma altamente precisa y con un bajo coste computacional todas las

restricciones dadas, además de la aleatoriedad y el realismo buscado en un terreno natural.

En el capítulo 4 se presenta un método que distribuye elementos naturales sobre un terreno dado de forma procedural, maximizando la cobertura y siguiendo varias restricciones biológicas y del terreno. La representación realista final se obtiene simulando el proceso de crecimiento de las semillas. Los resultados muestran que la distribución resultante cumple los requisitos dados de cada especie, además que, esta mantiene un aspecto visual realista debido a la heterogeneidad del tamaño de los elementos y su posición y debido a las agrupaciones finales obtenidas.

En cuanto a la optimización de la visualización de los elementos naturales, en el capítulo 5 se presenta una técnica de simplificación específica para el follaje que prioriza la apariencia visual. Esta técnica busca reducir la geometría del modelo eliminando hojas completas, priorizando aquellas que producen menor impacto visual al ser eliminadas. Cada vez que una hoja es eliminada, la técnica busca aquella que estuviera más cerca y la escala en proporción al error cometido en la poda. Los resultados demuestran que esta técnica consigue una simplificación con un impacto visual muy bajo, que colocado a una cierta distancia del observador, la diferencia con el modelo original es apenas imperceptible.

Por último, en el capítulo 6 se presenta un modelo multirresolución continuo fácilmente integrable en motores de juegos. Este modelo utiliza la estructura de datos obtenida con la técnica de simplificación anterior para actualizar el nivel de simplificación de los elementos de forma suave y en tiempo real. Los resultados obtenidos muestran cómo este modelo permite mantener la calidad visual mientras el costo de renderizado se reduce significativamente, comparándolo con el renderizado sin simplificación.

Cada uno de los temas tratados en esta tesis concluyó además, en una publicación de revista tal como se indica en el último punto del capítulo 7. La publicación llamada “Procedural modelling of terrains with constraints” corresponde al capítulo 3; “Procedural Modelling of Plant Ecosystems” corresponde al capítulo 4; Para el capítulo 5, se realizó la publicación “Viewpoint-Driven Simplification of Plant and Tree Foliage”; Y por último, el capítulo 6 concluyó en la publicación “Multiresolution Modeling for Foliage Rendering”.

Trabajo futuro

Aunque los métodos presentados en esta tesis aprovechan las oportunidades de mejora dados por los métodos existentes, también presentan algunas carencias que pueden ser subsanadas ofreciendo nuevas líneas de trabajo futuro. A continuación, se presentan algunas de las líneas más interesantes.

En primer lugar, en cuanto a la distribución procedural de elementos vegetales, se podría mejorar considerando el uso de celdas para crear patrones de vegetación y cubrir grandes superficies sin aumentar el costo computacional. Por otro lado, para poder mejorar el realismo en la distribución se podría ampliar las restricciones biológicas en otras como la humedad del suelo, la temperatura y la posición del sol. En línea con esta línea de investigación, se está analizando la opción de poder añadir áreas artificiales como campos de cultivo. Al no tener un comportamiento natural, estas áreas ignorarán otras características como la colocación irregular o la selección de especies según las características del terreno, por lo que se requiere un mayor estudio. Además, el uso de diferentes modelos que representen edades diferentes de cada especie permitiría representar diferentes niveles de crecimiento de forma mucho más realista.

En cuanto a la optimización de la visualización de árboles y plantas, el

método presentado no modifica la estructura del elemento, por lo que se quiere estudiar añadir un esqueleto al mismo para realizar la simulación del viento. Además, un aspecto que necesita ser mejorado en el modelo multirresolución presentado es su integración en la GPU, con la intención de reducir el costo computacional. El modelo presentado demuestra una reducción de costes significativa, pero al ejecutarlo en la CPU, no se puede aprovechar todo el potencial de la tecnología actual y, por lo tanto, todavía se le puede sacar mayor partido.

Por último, una línea en la que se quiere trabajar es unificar todas las partes de esta tesis en un plug-in que se pueda utilizar en motores de juegos para generar un entorno natural completo. Además, se podrían agregar funcionalidades para otras características de los entornos naturales, como la integración de rocas, efectos atmosféricos e incluso vida animal. La idea final de esta última línea de investigación es conseguir un resultado fotorrealista y poder comparar el realismo obtenido utilizando los datos de un entorno natural real con fotografías tomadas del mismo.

Abstract

The representation of natural environments is of great importance in the video games, simulators and virtual reality industry. More and more environments with more details and greater realism are demanded, which requires a too high investment of time to generate them manually. For all this, procedural content creation techniques have been developed that allow generating complex environments, although they do not usually allow control of the final result. However, these environments need real-time visualization techniques that can handle all the generated geometry interactively. In this sense, the main objective of the thesis is to present a series of methods developed with the intention of controlling the final result in procedural generation and improving the visualization of natural environments. On one side, a new method of procedural terrain generation based on noise functions is presented, which results in a pseudo-random terrain that precisely complies with the height restrictions given in advance. In addition to this, a user-assisted process tree and plant distribution method is introduced, which uses biological constraints on plant species. Finally, to improve the visualization of natural environments, a technique is presented to simplify the foliage of natural elements. This technique is based on mutual information from the point of view and a continuous multiresolution model that uses this technique to display different resolutions of the element according to its importance in the scene, allowing interactive visualization of the natural environment.

Keywords: procedural generation, terrain modeling, natural environment, geometric simplification, continuous multiresolution model.

Acknowledgements

This thesis is the result of many years of work, not only of my doctoral studies, but also of all previous years that led me to start this journey. But of course, I would not have been able to get here without the help and support of many people.

First of all, I would like to thank my director Miguel Chover, for his advice, encouragement and supervision even before I started my doctoral studies. And above all for having given me the opportunity to work in the field of computer graphics.

I would also like to thank all people I have worked with in recent years, from whom I have always been able to learn something. With special mention to Inmaculada Remolar, who without her support during all this time would not have gotten here.

On the other hand, I would like to thank a very special person for me, Sari Vidal. Who introduced me to this world and taught me how exciting research can be. If it was not for her, I would never have even started my doctorate and without her support, much less could I have finished it.

Last but not least, to my mother Pilar Garcia, for her patience and love throughout my life. And to my friends who were always there when I needed a break, especially Marcos, Rebeca, David and Oscar, who although they did not understand very well what I was talking about, they listened to me when I needed to talk about my work.

Index

1	Introduction	1
1.1	Objectives of the research	3
1.1.1	Terrain procedural generation	3
1.1.2	Procedural distribution of plant elements	4
1.1.3	Visualization of plant elements	5
1.2	Contributions and overview	7
2	State of the Art	11
2.1	Related work for terrain representation	11
2.1.1	Generation of terrain with constraints	12
2.1.2	Adaptation of the terrain to constraints	13
	Results not controlled by the users	13
	Results controlled by the users	15
2.2	Related work for natural environment generation	17
2.3	Related work for geometry optimization techniques	21
2.3.1	Simplification Methods of Tree Models	21

2.3.2	Simplification Methods Based on Mutual Information	24
2.4	Related work for multiresolution foliage rendering	25
3	Terrain representation	27
3.1	Introduction	28
3.2	Procedural generation of terrain with constraints	31
3.2.1	Case study	33
3.3	Large terrain models	37
3.4	Results and Discussion	38
3.4.1	Details of the used constraints	39
3.4.2	Visual results	40
3.4.3	Efficiency and accuracy	43
3.4.4	Realism	45
3.4.5	Different iterations over the same paths	45
3.4.6	Validating the presented method	46
3.4.7	Cases of failure	48
3.4.8	Discussion	49
3.5	Conclusions	50
4	Procedural distribution of vegetation	53
4.1	Introduction	54
4.2	Plant cover method	57
4.3	Seed dissemination	58

4.4	Labeling the seeds	59
4.4.1	Evaluating the terrain characteristics	59
4.4.2	Areas of influence of a vegetation species	62
4.4.3	Labeling of the seeds according to terrain characteristics and areas of influence	64
4.4.4	Adding neighboring information	65
4.5	Vegetation growth	67
4.6	Results	71
4.6.1	Validation of the method	73
4.6.2	Computational time analysis	77
4.6.3	Visual results	79
4.7	Conclusions	81
5	Simplification techniques for vegetation	85
5.1	Introduction	86
5.2	Viewpoint-based error metric using mutual information	89
5.3	Simplification scheme	92
5.3.1	Leaf Removal Operation	93
5.3.2	Evaluating the Number of Viewpoints	95
5.3.3	Simplification Algorithm	98
5.3.4	Algorithm Complexity	100
5.4	Results	101

5.4.1	Comparing the Simplified Approximations	101
5.4.2	Evaluating the Simplified Approximations	105
5.4.3	Simplification Considering Visual Obstacles	108
5.5	Conclusions	109
6	Multiresolution Foliage Rendering	113
6.1	Introduction	113
6.2	Model overview	115
6.3	Data Structure	116
6.4	Rendering algorithm	117
6.5	Results	121
6.5.1	Tree Rendering	122
6.5.2	Forest Rendering	123
6.5.3	Visual results	125
6.6	Conclusions	126
7	Conclusions and future work	129
7.1	Conclusions	129
7.2	Future work	134
7.3	Publications	136
7.3.1	Journals	136
7.3.2	Conferences	137

Index of figures

1.1	Surface of a natural environment (Terragen).	3
3.1	Graphical representation of the elements involved in obtaining Noise(P).	31
3.2	Procedural texture obtained with the Perlin method and its terrain representation.	33
3.3	Procedural texture obtained with the Perlin method and its terrain representation.	34
3.4	Visual results obtained for the simplest paths and the synthetic figures.	41
3.5	Visual results obtained for the real paths.	42
3.6	Different close views of the resulting terrain that includes the Second real path.	45
3.7	Results obtained after performing different iterations of the method taking the path called <i>Simple</i> as input data. .	46
3.8	Results obtained modifying the number of octaves, the frequency and the persistence.	47
3.9	Example of a case of failure.	49

- 4.1 Placement of seeds after applying a deviation of the initial position. 58
- 4.2 (a) Height-map with altitudes ranging between 800 and 1300 meters. (b) Slope map, which varies between 0 and 90 degrees. 61
- 4.3 (a) *Diffusion points* for three different species, differentiated by three colors (blue, red and yellow). (b) Areas of influence associated with the *diffusion points*. . . 63
- 4.4 (a) Labeling of the training set of seeds. Three vegetation species (blue, red and yellow) have been considered. Black dots represent seeds with no vegetation species assigned. (b) Distribution of the vegetation species obtained by applying the k nearest neighbor classifier on the rest of the points with $k = 3$ 66
- 4.5 Graph analyzing all data of example regarding the three plants belonging to the three species under consideration (A, B, C). Green lines are the distance between the plants and brown lines the maximum radii that each of them can reach until meeting point M 69
- 4.6 (a) The *cawi* obtained is shown for each seed on the terrain. (b) In the detail shown on the right, the different sizes of the circles can be appreciated. 70
- 4.7 Species of trees used, from left to right: *Pinus Sylvestris*, *Abies Alba*, and *Sorbus Aucuparia*. 71
- 4.8 Coverage maps obtained by modifying some features. Vegetation species: *Abies Alba* (red); *Pinus Sylvestris* (blue); *Sorbus Aucuparia* (yellow); *Pinus pinea* (magenta); *Quercus robur* (cyan); *Betula pendula* (green). 75

4.9 Results obtained when applying the distribution in a rectangle of 40×60 meters considering different number of distances.	78
4.10 Evolution of computing costs with respect to the number of trees.	79
4.11 Visual process to obtain the final terrain with trees from the height-map.	80
4.12 Visualization of the result obtained at long, medium and close distances.	83
5.1 Example of foliage simplification performed by the method presented in [1]. (a) Original: 20,350 leaves; (b) simplified: 2348 leaves.	88
5.2 Workflow for obtaining a simplified approximation of a tree, finally formed by 506 polygons. VMI, viewpoint mutual information.	93
5.3 Example of leaves represented by different polygon combinations. (a) Leaf representation by quads; (b) leaf representation by a mesh with several polygons.	94
5.4 Example of resizing a leaf that covers the pruned one (drawn with dashes on the right).	95
5.5 Camera positions that have been used in the test. (a) Dodecahedron: 20 vertices; (b) icosahedron: 12 vertices.	96
5.6 Error according to the number of cameras used. (VNC: Variable number of cameras).	97
5.7 Time cost according to the number of cameras used. (VNC: Variable number of cameras).	97

- 5.8 Trees and bushes used in the experiments that compare both methods. (a) English oak: 20,376 leaves; (b) *Carya illinoensis*: 8140 leaves. 102
- 5.9 Visual comparison between the methods applied to the model tree shown in Figure 5.8a: Top. the stochastic simplification method (SM); Middle. FSA simplification; bottom. the VDS results. From left to right: 50%, 25% and 10%. 103
- 5.10 Visual comparison between the methods applied to the model tree shown in Figure 5.8b: Top. the SM; Middle. FSA simplification; bottom. the VDS results. From left to right: 50%, 25% and 10%. 104
- 5.11 Visual results of the VDS scheme. From top to bottom: *Betula lenta*: original model 16,100 leaves; retaining 10% of leaves: 1,610; and 1%: 161 leaves. *Alnus glutinosa*: : original model 41,155 leaves; retaining 10% of leaves: 4,115; and 1%: 411 leaves. *Betula populifolia*: original model 20,280 leaves; retaining 10% of leaves: 2,028; and 1%: 202 leaves. 106
- 5.12 Visual results of the VDS scheme. From top to bottom: *Sorbus aucuparia lenta*: original model 2,760 leaves; retaining 10% of leaves: 276; and 1%: 27 leaves. *Acer platanoides*: : original model 105,688 leaves; retaining 10% of leaves: 10,569; and 1%: 1,056 leaves. 107
- 5.13 Example of the composition of a scene where different approximations of the same plant are shown (original foliage, 50%, 25% and 10% approximations). 107

5.14 Simplified approximation of foliage when surrounded by a fence. (a) Level of detail formed by 10% of the leaves; (b) detail of the approximation making transparent the fence in front.	109
5.15 Simplified approximation of a plant when it is situated close to walls. (a) Ninety percent simplified; only 10% of leaves remain; (b) back view. Walls have been cut for a better appreciation.	110
6.1 (a) Collapse operation, removing Leaf L.i. (b) Split operation, adding Leaf L.i.	120
6.2 The three vegetation species used in the tests.	121
6.3 Comparison of the FPS obtained with the three vegetable species at different distances.	122
6.4 Analysis of the time required to extract the different levels of detail for the three vegetable species at different distances.	123
6.5 Chart showing the FPS obtained while the amount of trees rendered increases.	124
6.6 Comparison between three different distances. With a reduction of 20%, 70% and 85% respectively.	125
6.7 Forest with nine trees with simplification applied.	126
7.1 Terrains obtained with the method of procedural generation of lands with restrictions.	131
7.2 Result obtained with the method of procedural distribution of plant elements.	132

- 7.3 Result obtained with the simplification technique based on mutual information. Top from left to right: 50%, 75%, 90%. Bottom from left to right: 0%, 50%, 75%, 90%, 95%. 133

Index of tables

3.1	Detail of the data used to test the presented method. . . .	39
3.2	Details of the interest points and areas associated with the real paths.	40
3.3	Time cost (seconds) and error introduced (%) generating the terrains of the hand-made routes and the special figures.	43
3.4	Partial and total cost (seconds) and the error introduced (%) generating large terrains that contain the real paths. .	44
4.1	The selected species data. The minimum and maximum radius that the specie can reach and, regarding the terrain, minimum and maximum height and maximum slope where each it can grow. All these data are measured in meters and the slope in degrees.	72
4.2	Estimated values used to calculate the number of plants per species w in function of the height and the slope (Equation 4.3).	72
4.3	Details of the parameters involved in the experiments performed.	74

5.1	Experimental data of the developed test comparing both simplification methods. VDS, viewpoint-driven simplification.	102
5.2	Experimental data evaluating the results of the presented method.	108

List of Algorithms

1	Process to obtain the terrain with constraints.	37
2	Process of leaf simplification.	100
3	Process to extract the Level of Detail.	119
4	Process to render the Level of Detail.	119

Chapter 1

Introduction

A natural environment could be understood as a world composed of living and non-living beings that interact with each other. These environments are composed of ecological units such as soil, rocks or vegetation, natural resources and physical phenomena such as climate or water, as well as the built environment, that is, one that has been modified by human hand creating buildings or agricultural land.

In the field of research that studies the theory and techniques for the generation of synthetic images with the help of a computer, Computer Graphics, there are different techniques to simulate these virtual environments. These techniques can be divided into manual, that is, those that are mainly characterized by the fact that at least a large part of the environment is created through human, and techniques of automatic interaction, also called procedural generation techniques. Sometimes, use manual techniques because they are artistically rewarding or even because of their therapeutic properties [slayton10] is prefer. But for many other times, these techniques can be tedious or time-consuming, while automatic techniques allow to create highly detailed environments

quickly and efficiently. As a disadvantage, automatic techniques can lead to losing control of the result and require knowledge of the involved algorithms to obtain good visual results.

Videogames, simulators, and virtual reality industries increasingly seek to increase the complexity and realism of natural environments [2]. This trend can be seen in the industry increasingly using realistic forest design tools [3, 4] such as SpeedTree [5], a realistic forest environment generation library. To meet this demand for realism and complexity, the need for the use of procedural generation techniques that include mechanisms that guarantee an adequate result is increasing. The thesis presented is aimed at those occasions in which designers do not have the necessary time required by manual techniques, do not have the necessary knowledge, or seek control of the results including randomness. This thesis seeks to create complex natural environments, minimizing interaction with the user, directing the process in a simpler way, and seeking to prioritize the realism of the results.

In this sense, the main objectives of the thesis are divided into three sub-objectives. The first of all consists of the procedural generation of terrain according to certain restrictions that the user will initially provide, such as GPS routes or certain height values for the mountains. The second objective addresses to the placement of vegetation on the ground, taking into account some characteristics such as the type of trees and plants, as well as their interaction with the height and slopes on which they are located. Finally, techniques are presented to improve the visualization of natural scenes, through the definition of simplification algorithms and interactive vegetation visualization. Because of this, a new continuous multi-resolution model is proposed for the representation of the foliage of this kind of elements.

1.1 Objectives of the research

1.1.1 Terrain procedural generation

When talking about terrain in graphics, it refers to the shape of the relief that the surface of the natural environment has, to which no element has yet been added, as can be seen in Figure 1.1. A way to procedurally generate terrain is through an algorithm that calculates the value of each pixel in a raster image. The result obtained is a map of heights where the color of each pixel indicates the height that the point should reach when represented as terrain.

Figure 1.1. Surface of a natural environment (Terragen).

Generally, it is expected that the resulting heightmap for a natural scene does not contain sharp changes between pixels and is not uniform, that is, it maintains the pseudo-random aspect of nature. When a land is not given any restriction for its generation, it must be modified to be able to add elements that depend on the surface such as rivers, lakes, roads, towns, mountain ranges or mountains with specific heights, etc. However, if it is known what main characteristics the terrain should

have beforehand, an algorithm could be designed to create it taking into account these restrictions. In other words, the algorithm could have as an example of a requirement, include flatter areas to be able to add towns or take into account that it must comprehend paths between elevations with not too vertical inclinations for the roads.

Adding these requirements randomly, could give good results, but there is a risk that they do not make sense or that they do not meet the needs of the designers, as in many solutions given in the literature where the final result is unknown beforehand. Other authors solve this problem through an interface that allows the user to modify the terrain in a simpler way in real time or by adapting the terrain to the desired characteristics until an approximation is achieved.

Most of the previous solutions focus on being able to control the general mountainous shape of the terrain, or on requirements to add specific elements. These solutions are insufficient to define specific characteristics of the terrain and obtain good integration results. For this reason, the thesis presents a method to generate terrain procedurally including initial restrictions on some of its height values. In this way, it is possible to obtain lands that include the randomness of nature but maintaining the restrictions imposed by the user. Furthermore, the developed algorithm allows the generation of different terrain models in their different executions that always maintain the input restrictions and a diverse final appearance.

1.1.2 Procedural distribution of plant elements

In an outdoor scene such as a natural environment, plant elements are very important to give realism. As with the terrain, having to place them by hand can be a very tedious task, especially in large stages where there can be thousands of these elements. Therefore, the procedural placement of these elements is necessary and for this reason it has

been a widely studied field in the literature, although it still remains an open problem.

When observing the real world, natural elements maintain a pseudo-random distribution, in which they tend to cluster by species but still allow the growth of other species in that cluster. In addition to that, each natural element has vital requirements in terms of temperature, humidity or other characteristics that will allow them to grow in certain points but will not in others. These characteristics, called biotic (characteristics that interact with other living elements) and abiotic (characteristics related to non-living elements such as water or temperature) must be taken into account to achieve a high degree of realism in the placement of plant elements. Otherwise, the results will be chaotic and unreal.

Previous work in this field has focused on the placement of small groups of trees or small areas of vegetation. On the other hand, those jobs that seek to cover large areas simply introduce randomness to reduce the computational cost. However, random changes in position or scale of plant elements do not guarantee the creation of realistic environments.

For this reason, this thesis introduces a model that allows plant elements to be placed according to their biological characteristics and also the terrain on which they are located. The system from an initial distribution raises a system of inequalities that allows each element evolve to cover the maximum extension associated with its species. The result is a land covered by vegetation that tends to be grouped by species, just as it happens in the real world.

1.1.3 Visualization of plant elements

Efficient rendering in real-time scenes can be a problem in computer graphics. The biggest problem happens in large scenes, where there

can be hundreds of thousands of trees and plants. Each of these elements is formed by a trunk, all its branches and a large number of leaves, which also, depending on the shape of the leaf, could be formed from a quadruple to a dozen polygons. Taking this into account, to visualize a forest with hundreds or thousands of elements, the total number of polygons to draw can reach millions. Also, by adding all the other features that will increase the cost of rendering, such as lighting or shading, then the time required to be able to draw the entire scene using existing graphics hardware will exceed the time required to render it in real time.

In many solutions billboards and impostors are used to replace the polygons of plant elements and thus reduce the amount of them to be drawn. This solution produces good results over long distances, but as the distance is reduced, the realism decreases due to the emergence of “parallax and clustering artifacts”. Solutions that seek to prioritize visual appearance focus on finding simpler versions of the 3D model, modifying its geometry to reduce the number of polygons that form it.

Techniques of level of detail modeling allow to adapt the resolution of polygon meshes to the importance of objects in the scene. These techniques work well for continuous surfaces like terrain, but most do not work properly when it comes to the foliage of a tree or plant. These types of elements are represented by independent polygons, so these solutions tend to eliminate those of the leaves and therefore reduce the leafiness of the foliage causing a high visual impact. To solve this problem, this thesis presents a simplification technique aimed at the foliage of trees and plants, giving more importance to the visual appearance. To achieve this, on the one hand, a pruning is carried out always looking for the leaf that, when eliminated, produces the minimum error in the visualization from different points of view. Initially, those that are hidden by other leaves, or rather other elements are eliminated, so that at close distances the visual difference with the

original is imperceptible. And on the other hand, as the simplification increases and therefore the error produced, leaves near to the eliminated ones suffer a scaling proportional to the error. In this way, gaps in the foliage are avoided, reducing the visual impact and increasing the realism of the elements at long distances.

The creation of simplified plant models allows their use in discrete multiresolution models. However, these models can produce popping effects when switching from one representation to the next one in level of detail. To solve this problem, the thesis also introduces a continuous level of detail model. This model uses the previously described simplification technique to create its data structure and has an extraction algorithm to be able to obtain the most appropriate approximation at all times in real time. In this way, changes in the foliage are made smoothly according to their importance in the scene, as the observer moves.

1.2 Contributions and overview

The organization used in this thesis is shown below, describing its contributions proposed:

- **Chapter 2: Previous work**

This chapter presents the state of art in the different matters addressed in this thesis. Up front, previous work on the procedural representation of terrain is described to include restrictions. Both those that adapt the land to restrictions, and those that include them when generating lands automatically. Subsequently, the different works on procedural placement of natural elements, the characteristics they take into account and their operation are reviewed. Lastly, the works related to geometry optimization techniques are described, focused on simplification models based on polygons. In addition, those studies that use "mutual

information” as a simplification criteria is reviewed, a criteria used in the proposed algorithm.

- **Chapter 3: Terrain representation**

The method presented in this chapter allows to generate a random terrain in a procedural way, which maintains the initially included requirements in its final result. These requirements make it possible to define a priori, GPS routes, points of interest, rivers, etc. The rest of the terrain height values are calculated based on the restrictions, obtaining more regular heights in the areas near them and increasing their randomness. As they drift away Finally, the resulting heightmap includes all the given constraints, but maintains the pseudo-random appearance that is expected of natural terrain.

The result generated has been validated using the modifiable attributes commonly allowed by procedural terrain generated with noise functions, modifying the number of octaves, the frequency and the persistence. This validation allows us to know if by using the same criteria for customizing the land, the result obtained in the restrictions is altered.

- **Chapter 4: Natural environment generation**

This section describes a method that allows natural elements to be distributed procedurally on any terrain. To achieve a realistic distribution, the combination of biotic and abiotic characteristics is taken into account. Subsequently, in order to simulate the way plants grow in nature, the method uses the k-nearest neighbors classifier to ensure the groupings of species.

The main contribution of the method lies in the use of a system of inequalities to simulate the competitiveness of each element of the plant set. Through this system, it is established which elements die and what size the others reach to maximize the coverage of

the land, taking into account the growth requirements of each tree or plant.

- **Chapter 5: Geometry optimization techniques for plants and trees**

This chapter introduces a new method for simplifying the foliage of trees and plants that prioritizes visual appearance. The method uses as an error metric, a measure of information theory called point-of-view mutual information. This measure uses mutual information to capture the variation in the shape of an element. The method generates a set of views of the foliage to be simplified and calculates in each one of them the error made when eliminating each leaf, using this measure as a simplification criteria.

In this way, the leaf to be deleted is selected and the process is repeated until there are no more leaves left. The method prioritizes the pruning of the leaves that are hidden, preserving the visible parts of the foliage. Once the leaves are eliminated, in each iteration a scaling of the nearest leaf is carried out, which guarantees the maintenance of the leafiness of the foliage.

- **Chapter 6: Multiresolution foliage rendering**

In this chapter, a multiresolution continuous model for the representation of the leaves of trees and plants is presented. This model allows to adapt the geometry to its importance in the scene in an interactive way. The data structures of the model are built from the simplification sequence obtained by applying the techniques from the previous chapter. Every time the distance to the element changes and therefore its importance in the scene changes, these algorithms calculate the current level of simplification and compare it with the previous one. Depending on the result obtained, they add or remove leaves from the foliage representation following the pruning order and scale the rest of the leaves using the corresponding scaling value, extracted from the

data structure.

To facilitate their use and allow obtaining results, the algorithms have been developed in a simple way to be able to easily implement them in a game engine.

Chapter 2

State of the Art

The representation of natural environments is essential in a wide range of applications such as geographic information systems, simulations, or video games. These environments are made up of many elements, but among the most important are the land and vegetation cover. Both elements play a fundamental role in obtaining the necessary realism in the simulation of an ecosystem. For this reason, in recent years a significant number of works on the representation of the terrain and the distribution and optimization of plant elements have appeared in the literature.

This chapter presents the works that make up the state of the art of the three topics related to this thesis: Terrain representation, natural environment generation and geometry optimization techniques.

2.1 Related work for terrain representation

One of the most widely used terrain generation methods is procedural modelling. It is mainly based on the fact that terrains are self-similar [6]

and the most of works that manage this generation method use noise functions to obtain the height values of the terrain [7, 8].

Some of these works have included some constraints to condition the resulting terrain. According to the moment in which these constraints are applied, they can be mainly classified in two groups: methods that generate a terrain that includes the desired elements, and methods in which the final terrain is obtained by applying the required constraints to an already created terrain.

2.1.1 Generation of terrain with constraints

In order to create a realistic terrain automatically, one of the most important characteristics is to be able to generate it including constraints imposed by the user, so that it fits the terrain to be represented as much as possible. For this purpose, different methods have been developed.

Some studies are based on the modification of the method of terrain generation using mid-point displacement [9, 7]. Kamal et al. [10] present a technique that generates a mountain by taking its elevation and base propagation as a constraint. Belhadj [11] presents a more general method in which a set of values constrain the mid-point displacement process. The constraints consist of scattered points of elevation provided by a satellite or other geological data acquisition sources.

A larger-scale method is the one proposed by Doran et al. [12], in which, using agents, constraints are added during the generation of the terrain, thereby creating geographical features such as mountains, beaches, etc. Although it allows for control over the terrain relief, it does not allow us to know the accurate position where these occurrences take place [13].

Some works initially generate the constraints and then model the terrain based on them. This is the case of the procedural approach

presented by Genevaux et al. [14] that uses river networks for large terrain modeling. The landscape is generated by combining procedural terrain and river patches with blending and carving operators, but the user lacks fine control over the result.

The work of Genevaux et al. [15] requires the user to define and distribute many different primitives by hand. They combine feature-based primitives and hierarchical procedural modelling to generate the terrain, including rivers. Guerin et al. [16] extend this model and present a hierarchical sparse terrain model that defines continuous terrains with varying level of detail. The approach represents terrains as elevation functions built from a sparse combination of automatically distributed primitives and, linear combinations of land-form features stored in a dictionary.

Recently, Gasch et al. [17] present a method of automatic terrain creation with constraints that force some height values to concur with a series of points determined by a GPS route. The limitations of this method are given by the size and complexity of the routes, since the accuracy diminished as they became bigger. This fact restricts its use to artificial routes, as tests carried out using real routes result in unrealistic maps.

2.1.2 Adaptation of the terrain to constraints

The following methods adapt an already created terrain to include elements such as roads or rivers. They can be classified depending on the user interaction required.

Results not controlled by the users

This group considers works in which, although users can interactively apply some processes to a terrain to add its natural elements, they do

not have any control on the final result.

A realistic way to modify terrain to simulate soil erosion caused by air or water is by using physics-based algorithms. Kelley et al. [18] propose a method based on empirical erosion models used in geomorphology to simulate the erosion of stream networks, adjusting the surface by triangulation and fractal interpolation. With the aim of improving the representation of relief, the approach proposed by [8] calculates the erosion of the water on the surface depending on the amount of water that would fall in each vertex of a fractal height field and the influence of the thermal weathering. They use a global simulation model.

The work of Nagashima [19] presents an algorithm that adjust valleys and mountains to the rainfall erosion and thermal weathering by using earth layer patterns on their surfaces. Neidhold et al. [20] combine a fluid simulation with an erosion algorithm. This fact enables to create eroded terrain interactively by changing simulation parameters or applying additional water to the scene. All these algorithms produce simulations that are difficult to control and predict, and interactive editing cannot be done for large areas of terrain due to that calculations require a lot of time and memory, and a high computational cost.

To accelerate the simulation of terrain erosion, some works downloaded the simulation to a GPU [21, 22]. Due to memory requirements and GPU limitations, these methods were not suitable for large terrain modeling. In order to simulate large-scale terrain affected by physics-based erosion, Vanek et al. [23] use adaptive tiles on the GPU. They divide the terrain into tiles of different resolutions according to their complexity. Then, each tile is stored as a mip-map texture and a mip-map is built for each tile. During the simulation process, they use the appropriate level of detail depending on the changes in the terrain.

Kristof et al. [24] present an hydraulic erosion simulation using the Smoothed Particle Hydrodynamics (SPH), which implies lower memory

usage and better performance for large scale simulations as compared to earlier approaches. A stochastic and interactive simulation of the impact of various geomorphological agents on terrain erosion in conjunction with the vegetation simulation is presented in the work of Cordonnier et al. [25]. During the simulation process, the input elevation map is modified using a discrete layer model in which different geomorphological and ecological events modify the data stored in these layers.

Some studies allow for the inclusion of roads in the terrain, such as the one proposed by Stachniak et al. [26]. This method requires the initial approximation of the terrain and a function that looks for an acceptable set of deformation operations to apply to a random terrain in order to adapt it to the constraints. The problem is that it is computationally expensive. The work presented by Zhou et al. [27] allows roads, rivers, crests or valleys to be generated. They use a sketch generated by the user that, when linked to the height map of the terrain, produces the terrain following that crest-shaped drawing. This method allows a large terrain to be modified, although the final height cannot be predicted accurately.

Results controlled by the users

This group involves methods where the users have control over the final result of the terrain appearance.

Based on Perlin noise, Scheneider et al. [28] introduce an editing environment where it is possible to edit the terrain by interactively modifying the base functions of the noise generator. Their method replaces the Perlin noise grid with a set of user-drawn greyscale images. Gain et al. [29] allow the user to draw the silhouette and the boundaries of a mountain chain through the propagation of noise, the terrain grows until the constraint is reached. To facilitate user interaction and make it more intuitive, Smelik et al. [30] propose a method in which a top view

of the map is drawn, covering both heights and soil material. Rusnell et al. [31] base the terrain synthesis method on the interpretation of a weighted graph generated from a set of generator vertices.

The work presented by Hnaidi et al. [32] allows a designer to draw 3D curves to control the shape of the terrain generated. Each curve is enriched with different properties, such as elevation and slope angle, which become constraints during the modelling process. Bradbury et al. [33] present a sketch-based toolset that allows the manipulation of virtual terrains. The method works with a layer-based interface. Each layer includes three sub-layers showing, respectively, the high, medium and low frequency bands of the image. The user can mix these bands and apply different edits to each, as desired. The method for the creation of islands developed by Puig-Centelles et al. [34] is also included in this category, in which the resulting island created in real time can be observed through an interface.

Other methods use interfaces to allow the user to modify the terrain [35] or even to carve them in real time [36]. In these approaches, users paint a height-mapped terrain directly in 3D by applying simple terrain raising brushes and also GPU-based brushes that generate several types of noise in real time. The model proposed by Tasse et al. [37] combines the sketching interface and patch-based terrain generation in a hybrid system that gives wider control to the user and produces realistic terrains.

In an attempt to bring interactivity and user control in the synthesis of the terrain, the work of Gain et al. [38] combines user-defined maps with silhouette strokes used to define roughness. They use parallel pixel-based texture synthesis to interactively create the terrain from a database of height field examples, and gain control over the height and slope of the terrain by means of constraint points and curves. This method does not capture the large-scale terrain shape characteristics of real landscapes.

In 2017, Zhang et al. [39] presented a method to procedurally generate rivers and then adapt their routes according to a set of image features that are extracted by user interaction. The method adapts the initially generated rivers to these features by applying an iterative optimization in order to increase the similarity between the resulting rivers and the images.

Modifications made through user interaction have the advantage that it is much easier to know how the final result will be and facilitates trial and error approach. However, the disadvantage is that to get accurate results, the user must spend time creating the terrain. [13].

2.2 Related work for natural environment generation

The procedural generation techniques that deal with the automatic allocation of plants and trees estimate their position using different algorithms. Nevertheless, these techniques failed to create a distribution of plants with enough variety and realism because no ecological input data were considered. Over the years, several works have appeared that deal with this problem.

The biotic and abiotic characteristics of the vegetation species [40, 41] are essential in the process of obtaining the exact positions of the plants in a terrain where a high density and variety of species are required [42]. Onrust et al. [43] classified the techniques of plant species placement in two main groups: based on ecological modeling techniques and based on procedural generation of ecosystems.

Works classified in the first group distribute the plants based on real data related to the terrain or the vegetation species to represent. These techniques can, in turn, be classified in two sub-groups depending on the source of the data used as input. On one hand, dynamic ecological

modeling techniques simulate ecological processes taking into account data that can be extracted from a raster map, such as height, biomass or vegetation cover [44, 45, 46, 47]. On the other hand, neutral ecological modeling techniques [48] generate a distribution of plants on a terrain following a grid structure that is based on some vegetation species information.

According to Onrust et al. [43], the second group is related to the procedural generation techniques that perform an automatic distribution of plants and trees. Works in this group estimate their position using different algorithms. Nevertheless, these techniques usually fail to create a distribution of plants with sufficient variety and realism because no ecological input data were considered. Over the years, several works have appeared that deal with this problem.

Deussen et al. [49] propose the framework of the Open L-Systems model [50] to simulate the individual-based ecosystem models. The growth of an individual ecosystem is represented using the dynamic behavior model of Firbank and Watkinson [51]. This model starts from an initial set of randomly distributed circles with initially random radii that represent plants growing in the ground. In a second phase, control by the user is incorporated through a gray-scale image that represents the density of the plants. Individual plant positions that fit these densities are created using the Floyd-Steinberg error diffusion algorithm [52]. Finally, the points produced by this algorithm are altered slightly to make the distribution more random. Nevertheless, the result of the distribution of the plants is too uniform. Moreover, the cost of the process in terms of time is high, according to Lane and Prusinkiewicz [53].

These authors propose the symbiosis of multiple vegetation species to generate plant distributions. They combine a probability value with a dart-throwing algorithm that allows plants to be placed in their most probable areas. In addition, each plant can produce an inhibition or expansion effect on the remaining plants, updating the probability field

around it. However, the grammatical elaboration method turns out to be quite complex and the computation times relatively high.

A method called FON (Field Of Neighborhood) is proposed by Bauer et al. [54]. It describes a circular zone of influence around the plant, whose radius determines the distance from which it can interact with its neighbors. This radius depends on the soil characteristics, size, and nutritional needs of the plant and the space needed for that nutrition. Based on FON, Alsweis and Deussen [55] propose a method that simulate plant communities of two different species in evolution and later, in [56], they calculate the distribution of the plants by combining Poisson Disc Distributions (PDDs) [57] with Wang-tiles [58], although neither the distribution by species nor the interactions between them are considered.

Weier et al. [59] extend the previous method and generate large areas of vegetation using Wang-tiles of vegetation with reusable distributions that allow a terrain to be covered. Each Wang-tile is generated with PDDs with a variable radius. The tiles are organized in a multilevel structure and treated with a nested KD-tree, which allows them to be reused during visualization. In addition, they propose a solution to represent plants whose area of influence is on the edges of the tiles. The main problem with this proposal is that the size of the discs that line the ground tends to be too homogeneous.

Onrust et al. [60] present a procedural plant placement technique that combines the Poison disk distribution with Wang tiles, establishing an ecological modeling technique. Results were only tested with isolated plants that had problems with the position of the impostors with respect to the camera. The work proposed by Onrust et al. [43] is based on vegetation location information resulting from the combination of several parameters such as the Normalized Difference Vegetation Index (NDVI), height-map, biomass or coverage maps.

Gasch et al. [61] present a method of automatic vegetation placement that takes as its input the height-map of a terrain where a route is represented. The method extracts the values necessary to distribute the vegetation species from the height-map and combines them with a noise distribution based on Perlin noise [62]. So by doing this, it becomes possible to obtain placement patterns with a greater probability of occurrence in favorable points on the map and near the route. However, this method does not take into account the evolution and growth of plants.

Li et al. [48] propose a procedural model based on graphs that generate realistic vegetation distributions with different compositions and configurations adjusted to the metric of a specific landscape created from patches. The metrics used include the number of tree species, their proportions in the landscape, the number of patches that have to be populated by each species and their average shape. The seeds are randomly placed and their species are assigned in order to obtain the final distributions of the trees. The method establishes the distribution of each species with a time efficiency that satisfies the requirements to build large-scale virtual forests.

A more generic method to describe growth and simulation of ecosystems is presented in the Growth-Grammars [63], which allows the user to access a tool to develop growth grammars through an interactive modeling platform, GroIMP. This grammar is executed through programs written in XL (a Java-based growth grammar language) and visualized by XL4C4D (an XL-plugin for CINEMA 4D). Although this method allows populations of plants to be generated in small areas, it does not reuse some components to cover more extensive areas.

Benes et al. [64] present in their paper a method to procedurally distribute vegetation in urban environments. They address the problem of spatial distribution of vegetation by combining a user-guided simulation system with an interactive procedural system that integrates

plants into the urban environment.

Starting from the 3D geometry of an urban layout with initial conditions and parameters, procedural rules are inferred. A manageability level is calculated for each area of the urban space, which defines the amount of influence between the simulation of the wild ecosystem, where plants compete for resources and seeds freely, and the ecosystem managed by the user, where almost it is not allowed sowing and the plants grow only under well defined conditions.

2.3 Related work for geometry optimization techniques

This section reviews the methods that have appeared to reduce the complexity of the sparse component of the trees and plants. Although some representative works are based on points [65, 66, 67] or based on images [68, 69, 70], those that are based on polygons will be reviewed, which is the primitive that the presented method uses. Simplification methods that use the mutual information as an error metric to reduce the polygonal complexity are also reviewed in this section.

2.3.1 Simplification Methods of Tree Models

Among the most important works to reduce the number of leaves while maintaining the visual appearance, those that use an iterative process of foliar collapse stand out.

In 2002, Remolar et al. [1] proposed the first method for foliage simplification that deals with polygons, the Foliage Simplification Algorithm (FSA). Two leaves disappear and are replaced by a new one that preserves an area similar to the original ones, maintaining the visual effect of the foliage. The method uses an error function

to determine which pair of leaves is collapsed, taking the Hausdorff distance between two leaves and their planarity into account. One year later, Zhang et al. [71] presented a new algorithm, progressive leaves union (PLU), which enhances the cost of FSA by refining the error function with additional criteria. They also take the Hausdorff distance into account, but considering that the choice of the pair of leaves to collapse is reduced to leaves that have been collapsed with less than two different leaves (close in position and similar in shape). Neither FSA nor PLU can deal with quadrilaterally-shaped leaves. Based on PLU, Zhang et al. [72] propose a novel method, hierarchical union of organs (HUO). The method includes triangular leaves and introduces a hierarchy into simplification by making use of the fact that the leaves in a crown are constructed by instantiating a phyllotaxy cluster sample with different transformations. In 2009, Deng et al. [73] presented two different algorithms based on HUO, one for broad leaves represented by quadrilaterals and one for thin leaves represented by lines. They introduce leaf density to adapt compression to the local distribution of leaves, so that more visually-relevant details are kept. Two years later, Bao et al. [74] proposed a new leaf modeling method that uses the texture to simplify triangular mesh models of leaves.

Other works that have also appeared, try to reduce the foliage by performing pruning operations. Based on stochastic pruning, Cook et al. [75, 76] proposed a stochastic simplification of leaves with random removal of leaves. The geometry is reduced down to a certain fraction. Next, the remaining geometry is scaled so that the total area of rendered surfaces is equal to the original area. In 2011, Neubert et al. [77] presented an optimized pruning algorithm, improving upon previous techniques by applying model-specific geometry reduction and optimized scaling, as well as view-optimized pruning. They introduced precision and recall (PR) as a measure of quality for rendering complex geometry with pruning. This measure does not consider pixel colors, but the right pixels of a rendered object are set.

View-dependent representation techniques take into account the distance between tree models and viewpoints. These methods diminish the number of leaves in a crown in a non-uniform way, eliminating more leaves in the less visible parts of the tree than in the more visible areas.

In 2011, Gumbau et al. [78] presented a foliage pruning method for real-time vegetation rendering based on stochastic pruning of needless leaves for a given level of detail (LoD). In order to build the LoD scheme, a stochastic pruning method is applied in a pre-process; LoD tree model foliage is divided into a cloud of cells; and each cell's visibility is computed from a set of external viewpoints surrounding the foliage. Following a camera-dependent criterion, the less visible parts of the foliage are detected in real time. The appropriate resolution of the foliage is calculated taking into account both the distance of the tree to the observer, as well as the visibility of the leaves.

Lindstrom and Turk [79] address the problem of visual similarity by developing a purely image-based metric. They determine the cost of an edge collapse operation by rendering the model from several viewpoints. The algorithm compares the rendered images to the original ones and adds the mean-square error in luminance across all the pixels of all the images. Then, all edges are sorted by the total error incurred in the images, and after that, the edge collapse that produces the least error is chosen. Its main disadvantage is the high temporal cost. In 2002, Zhang and Turk [80] proposed an algorithm that defines a visibility function between the surfaces of a model and a surrounding sphere of up to 258 cameras. A table of visibility between the camera positions and surface triangles is computed, obtaining for each triangle a measure of its visibility. In order to guide the simplification process, this visibility measure is combined with the quadric error measure introduced by Garland and Heckbert [81]. The visibility map calculation is a very time-consuming process, so this method has a high cost.

The simplification techniques developed by Lee et al. [82] and Lee

and Kuo [83] are based on pixel-based metrics. They do not consider the tree model leaf density. To solve this problem, in 2009, Lee et al. [84] performed a simplification technique based on the tree leaf density of rendered images viewed from multiple angles in which the simplification process preserves different levels of densities on all partitions of a tree model.

2.3.2 Simplification Methods Based on Mutual Information

In many works, information entropy has been studied to measure the correlation between a set of viewpoints and the visibility of the objects, as can be shown in Viola et al. [85] and Feixas et al. [86]. They suggest that the variation on entropy has a close relationship with the model silhouette during the simplification process. Based on the measures proposed in viewpoint entropy by Vázquez et al. [87, 88] and the viewpoint Kullback–Leibler distance introduced by Sbert et al. in [89] and [90], Castelló et al. [91] proposed a viewpoint-based simplification approach for polygonal meshes driven by an information-theoretic measure, viewpoint mutual information (VMI).

This metric measures the correlation between a viewpoint and the set of polygons of the object or scene. The algorithm applies the best half-edge collapse as a decimation criterion and uses the variation in mutual information to measure the collapse error as a new simplification error metric. VMI decreases the visual error more for simplified meshes than viewpoint entropy, since it is able to maintain the silhouette better, and achieves the best results in visual similarity because it is able to remove all the invisible inner regions. VMI has been used to compute the best camera positions in volume rendering and polygonal meshes.

The works previously analyzed are related to the simplification of general meshes. However, a viewpoint-driven foliage simplification algorithm based on an information theoretic measure was proposed

by Zhang [92] and Zhu et al. [93] that deals with the foliage of the plants. They reshape complex leaves as quadrilaterals. Then, the leaves are separated into clouds of cells for rapidly finding the best leaf pair to simplify. Two information-theoretic measures are evaluated in the simplification algorithm: mutual information and leaf visibility from a set of viewpoints. The simplification process consists of leaf-collapse operations, in which two leaves disappear and a new one appears based on the area maximum.

2.4 Related work for multiresolution foliage rendering

Some methods have appeared in the literature addressed to reduce the complexity of the sparse component of the trees and plants and their visualization. These display methods can be classified mainly in three groups, depending on the method they use to represent the foliage: images, points or polygons.

The first group uses image-based representation to perform real time rendering of plants [94, 95]. They considerably reduce the amount of required geometry, but they are usually used to represent distant plants due to their parallax effect.

Methods based on point/line representation usually combine polygons to construct hybrid models for trees. At close distances, the plant is represented by a polygon. With increasing viewing distance, branch meshes will transform into lines and leaves into points [96, 67].

The last group in the classification uses the polygons to represent the plants, regardless of the distance of the object to the viewer. However, different compression techniques are used to reduce the amount of data to visualize. They usually use different metrics to compute the off-line simplification that allows to build the level of detail structure. These

simplification methods can be based on stochastic, geometry or image metrics. They condition the performance of the multiresolution model.

Traditionally, works in the literature perform a level of detail modelling by means of multiresolution models built from simplification methods based on geometric metrics [97, 98, 99]. However, due to the sparse distribution of the leaves in the foliage, the metrics have to be adapted to this kind of meshes to ensure good performance of the process. Remolar et al. [1] present a simplification method by collapsing points based on the Hausdorff distance, comparing two sets of points, using the shortest distance between a point and the set of points. Deng et al. [99] expand this method to simplify thin or broad leaves, integrating both computations in an only simplification framework.

Other works have appeared that apply stochastic simplification [76, 78] to invisible parts of the foliage for performing the real time rendering. Dong et al. [100] propose a hybrid representation (HR) of a simplified tree model, which allows to adaptively select simplified models according to the resolution of different devices. Zhang et al [39] present a method that deals with forest visualization. They represent trees by means of polygon meshes plus semi-transparent lines. Line models with different transparencies are instanced on the GPU by merging multiple tree samples into a single model.

Image based simplification has been widely analyzed in literature. However, most of the works deal with general meshes [79]. In the previous point, a simplification method based on a viewpoint-driven metric that uses the mutual information in order to choose the leaf to prune is presented. Its results allows the user to obtain different approximations of the foliage of a tree, obtaining a discrete multiresolution model. Therefore, this section uses this method to obtain continuous level of detail of the foliage.

Chapter 3

Terrain representation

Terrain is an essential part of any outdoor environment and, consequently, many techniques have appeared that deal with the problem of its automatic generation, such as procedural modelling. One of the most popular methods to create terrains is Perlin noise, which generates a map of pseudo-random heights with an appearance similar to nature and with a low computational cost. However, the randomness of the method makes it difficult to have any control over the result obtained. In order to solve the problem of lack of control, this chapter presents a new method based on noise that allows for procedural terrain creation including elevation constraints. Consequently, the method establishes the restrictions as fixed values in the heightmap function and creates a system of equations to obtain all points that depend on these restrictions. In this way, the terrain obtained maintains not only the random noise, but also including the desired restrictions.

The chapter also includes how this method on large terrain models without losing resolution or increasing excessively the computational cost is applied. The results show that our method makes it possible to

integrate this kind of constraints with high accuracy and realism while preserving the natural appearance of the procedural generation.

3.1 Introduction

The representation of natural environments is essential in a wide range of applications, such as geographic information systems [101], flight simulations or video games. The terrain is a crucial part of these outdoor environments and its representation has been widely analyzed in the literature [102, 103]. Some studies try to extract real data [104], but this limits the results to the real world and it is not always possible to obtain all real data to be able to represent them. That is why automatic terrain generation is very important, a field of research that is still active. Taking advantage of their fractal dimensions [6, 105], many studies have appeared in the literature that address the automation of its creation process [106], mainly using procedural modelling. This modelling method has been applied to represent many features related to terrains, including roads, trees, villages, cityscapes [107, 108, 109], or even to represent atmospheric phenomena [110]. According to the work of Smelik et al. [13], procedural modelling was initially based on subdivision processes, in which the height map was iteratively created and, by thus, in order to create the elevation, a controlled amount of randomness was added in each iteration.

The first subdivision algorithm on which is based procedural modelling is known as the midpoint offset method [9, 7], in which a new point is generated in every iteration. The elevation of this new point is obtained as a result of the average of the elevations of its neighbours, with the addition of a random displacement which decreases with each iteration. Many other studies, mostly based on noise generators such as the Perlin method [62, 111] have been done in which the height map is obtained by a turbulence function. Perlin noise is fast, but it tends to create terrains

that are uniform and random, often requiring significant post-processing to add interesting features. There are some works that deal with this problem, such as the one presented by Parberry [112] that uses a variant of Perlin noise, called value noise, to generate geotypical procedural terrain based on a spatial analysis of elevation data extracted from GIS data. The advantage of the approaches based on noise generators is that the computational cost is low, due to the fact that every point can be calculated independently of its neighbours.

However, all the methods described above have problems controlling the generated terrain. Although some of them allow small constraints to be applied to the initial parameters, such as the maximum height, the procedural creation of a terrain that fits more specific constraints is more difficult to make efficiently.

Elements such as roads and rivers require a precise control to be able to include them in the final height map. Generally, these elements have specific forms and characteristics that condition the ground shape and including them in the terrain without control can cause unrealistic results. For example, roads cannot be vertical, rivers do not flow upwards, and so forth. To represent large terrains with different geometric features, Bruneton et al. [113] combined a digital elevation model with layered geographic information system (GIS) vector data. Thus, they were able to obtain precise boundaries of roads, rivers, lakes and fields and enforce vector data constraints in the terrain.

According to the work of Smelik et al. [13], the solution provided by the studies that attempt to solve this problem is mostly based on the trial-and-error method. Some of them even need to indicate where the modifications have to be made, something that may not be trivial for users, since it is almost impossible to reliably predict how each operation will affect the final result. Many commercial tools have appeared that allow the user to manually model, generate, and modify the terrain [114, 115, 116, 117]. But, procedural terrain modelling needs to be

controllable in order to simplify content creation.

However, some applications such as video games for training, require including real GPS routes. But for the rest of the terrain, it is required randomness. Furthermore, these applications also need automatic creation methods without user interaction during the creation process. Therefore, the initial problem to be solved was the integration of real GPS routes in procedurally generated terrain.

After studying the previous works, it has been found that noise-based works allow for variance in the final results, but at the moment there is none that includes previously established data on their results. With the problem in mind, it was decided to create a new noise-based method to take advantage of its randomness and characteristics. This method should have a low computational cost and also allow the inclusion of restrictions precisely in the final result.

The method solves an equation system that includes these constraints and generates a custom height map. Terrains generated with this new method are similar in appearance to those created with common noise functions, but including these previously established constraints, that the user can change, to design a random and at the same time custom terrain. The idea of calculating noise functions with constraints can be applied to any problem where the user needs to have some kind of control on the result.

The rest of the chapter is organized as follows. Section 3.2 explains the method created. Section 3.3 considers the creation of large terrain models. Next, Section 3.4 presents the results obtained. Finally, conclusions are included in section 3.5.

3.2 Procedural generation of terrain with constraints

The presented work not only creates a height-map that includes some previously given features, but they are combined with some randomly generated heights for the rest of the terrain, that makes it possible to have a natural appearance. This randomness is generated by means of a noise function. After analyzing some noise generators, Perlin [62, 111] has been used, since its low computational cost allows its use in real time and, in addition, its result maintains the pseudo-random appearance of nature.

According to the work of Perlin, the calculation of the noise in a point typically involves three steps. Initially, a two-dimensional grid is created, and a random value in the range of $[0, 1]$ is assigned to each node (Q). The next step is to determine the nodes that make up the cell in which the point to be calculated is located. And finally, the distance is obtained between the point P and every node Q of that cell. In a two-dimensional grid, this will require the computation of 4 distance vectors. The noise at the given point is finally obtained as the interpolation between the values computed at the vertices of the cell.

Figure 3.1. Graphical representation of the elements involved in obtaining $\text{Noise}(P)$.

Considering a cell with four grid nodes Q_{00} , Q_{01} , Q_{10} , Q_{11} , the noise value of a point P is based on a bilinear interpolation from the noise of the values of these nodes. Figure 3.1 illustrates the points and grid nodes involved in the process. The formula of this bilinear interpolation is presented in Equation 3.1.

$$\begin{aligned} Noise(P) => x \cdot y \cdot Q_{00} + x \cdot (1 - y) \cdot Q_{01} \\ + (1 - x) \cdot y \cdot Q_{10} + (1 - x) \cdot (1 - y) \cdot Q_{11} \end{aligned} \quad (3.1)$$

Turbulence is obtained at a point $Turbulence(P)$ by considering the noise value of that point P on some levels, which are produced by iteratively dividing the grid by 2. If i is the number that controls the quantity of noise levels that have been taken into account to calculate the turbulence, then Equation 3.2 shows the final function proposed by Perlin to calculate it.

$$Turbulence(P) => \sum_i \frac{Noise(P \cdot 2^i)}{2^i} \quad (3.2)$$

Considering the values assigned to every point, a grey-scale image is generated for a determined turbulence. Terrain generation is achieved by interpreting this image as a height map so that the colour of each pixel is used as the value of the terrain elevation. An example of a procedural texture and the terrain that it generates is shown in Figure 3.2.

Perlin method [62] generates random values for each of the Q nodes involved in calculating the noise of P , so the value obtained in that point is also random. This chapter introduces constraints into certain points, for instance P has a value conditioned by the terrain features that has to appear in the final representation. In this case, the Q values that have an influence in P cannot be randomly approximated. Instead of

Figure 3.2. Procedural texture obtained with the Perlin method and its terrain representation.

obtaining its value from the values of Q , our method changes the roles and the values that must be calculated are the points Q that influence P . This calculation is possible, establishing a system of equations using all equations obtained from all fixed values that are required for the given restrictions.

3.2.1 Case study

In order to make it easier to understand the method, the process of establishing the equation system is explained through an example. In this case, the turbulence is obtained combining three noise levels.

Figure 3.3 illustrates the data of the example. Initially, the value of the noise at point P_1 , $Noise(P_1)$ is established with vertices (Q_{00} , Q_{04} , Q_{40} and Q_{44}) on the first noise level (Figure 3.3(a)). According to Equation 3.1, these four vertices are necessary to obtain the noise of P_1 . On the second level (Figure 3.3(b)), the grid is divided and five more vertices appear (Q_{02} , Q_{20} , Q_{22} , Q_{24} and Q_{42}). However, only four of all the current

vertices condition the value of P_1 to obtain the $Noise(P_1)$. They have been marked with a circle to differentiate them from the rest. Finally, on the last considered level (Figure 3.3(c)), all the other vertices that complete the grid appear. This last level is where the vertices have the least distance between them, thereby giving the highest resolution. As on previous level, only three of these new values that need to be found intervene in obtaining $Noise(P_1)$, (Q_{03} , Q_{04} and Q_{13}) in addition to Q_{14} .

(a) Noise level 1.

(b) Noise level 2.

(c) Noise level 3.

Figure 3.3. Procedural texture obtained with the Perlin method and its terrain representation.

Taking into account the equation shown in Equation 3.2, the turbulence $Turbulence(P_1)$ is obtained by adding the noise on each of the three levels considered. The final formula is detailed in Equation

3.3, where C_height is the height value of the constraint, a_i , b_i , c_i and d_i replace xy , $x(1 - y)$, $(1 - x)y$ and $(1 - x)(1 - y)$ respectively, from Equation 3.2. Finally, ten Q vertices condition the value of the turbulence at that point, leaving fifteen that do not intervene and which, as in the Perlin method, will receive a random value.

$$C_height = \sum_{i=0}^3 \frac{Noise(P1 \cdot 2^i)}{2^i} \Rightarrow \frac{a_1 \cdot Q_{00} + b_1 \cdot Q_{04} + c_1 \cdot Q_{40} + d_1 \cdot Q_{44}}{2^0} + \frac{a_2 \cdot Q_{02} + b_2 \cdot Q_{04} + c_2 \cdot Q_{22} + d_2 \cdot Q_{24}}{2^1} + \frac{a_3 \cdot Q_{03} + b_3 \cdot Q_{04} + c_3 \cdot Q_{13} + d_3 \cdot Q_{14}}{2^2} \quad (3.3)$$

In the example used, only one point has been used as a constraint. However, a real case usually has a points set, for example a path with n points. This fact makes it possible to have the equation system 3.4 composed by the equation that obtains every point in the trajectory.

$$\begin{cases} C_height_P_1 = \sum_i \frac{Noise(P_1 \cdot 2^i)}{2^i} \\ \cdot \\ \cdot \\ \cdot \\ C_Height_P_n = \sum_i \frac{Noise(P_n \cdot 2^i)}{2^i} \end{cases} \quad (3.4)$$

This example clearly shows that there is an unbalanced number of unknown-known terms involved in the equation system: n known points on the trajectory versus $(2^{m-1} + 1) \times (2^{m-1} + 1)$ vertices that need to be found, being $m = 3$ in this case. In order to balance this difference without adding more unknown values to the equations system, more restriction points have to be added in the trajectory of existing points. To obtain these points in a coherent way, these new values have been

obtained by making a curve equation with the starting points. This allows us to obtain with great precision additional points that do not change the trajectory of the points set to represent and allow us to solve the system. The additional points number added depends on the unbalanced numbers of unknown-known terms. Therefore, points are added between the pairs of nearby points of each set until the number of equations is equal or greater than the number of unknown values.

Points are added in such a way as to force all the ones that form the trajectory to be equidistant (d_{min}). Having more points in the path with known values makes the equation system have more equations without increasing the number of values to obtain.

But it must be kept in mind that the new added equations can be equivalent to other equations of the system, that is, with linearly dependent results that they have an infinite set of solutions. This happens when the new added point is placed too close to the existing ones. To avoid this problem, the point is discarded and only valid turbulence equations are added to the equations system. For the new valid points, as has been done with the constraints points, its equation from its height value and its turbulence are obtained.

As a result of the resolving process, the values of the Q vertices that directly affect the constraint points are obtained. However, the process still leaves some Q vertices without value. This is due to the fact that they have not been used to obtain the value of the constraints. To finish the process, a random value to these vertices without value is assigned. In this way, the Perlin turbulence matrix that is obtained contains the constraints values given initially, but keeping the randomness in the parts that do not affect these constraints. The pseudocode of the implementation of all this process is shown in Algorithm 1.

Algorithm 1 Process to obtain the terrain with constraints.

```

//Obtain all turbulence equations for the system.
eq_n = 0
for all constraints do
    eq_n = eq_n + 1
    vector_equations(eq_n) = compute_eq(turbulence, C_height))
end for

//If there are more unknowns points than equations add new
//points and obtain its equation
for eq_n less unknow_number do
    eq_n = eq_n + 1
    equation(eq_n) = compute_eq(turbulence, C_height_new_point)
end for
Solve equation system(vector_equations)

//Get the value of nodes that don't influence the constraints
for all Q_nodes do
    if Q_node no value then
        Q_node = rand_value[0 - 1]
    end if
end for

```

3.3 Large terrain models

The number of noise levels involved in obtaining the turbulence limits the size of the grid. If m levels are considered, the resolution of the obtained grid will be $((2^{m-1} + 1) \times (2^{m-1} + 1))$. According to this, the resolution of the grid in the previous example, where 3 levels of noise were considered, is 5×5 nodes. The number of points of the final terrain depends on the resolution of the grid, so the larger a terrain is, the more noise levels have to be considered.

Some considerations have to be taken into account when large terrains are going to be generated. On the one hand, the size of the terrain determines the distance between the nodes on the grid that

represent it. If this distance is too large, all the constraints that fall below this minimum distance will be ignored, thus losing precision in the result. To avoid this loss of information, grids with a great number of points would have to be used in large areas, which would require using a high number of noise levels. Keep in mind that the equations size obtained by each constraint increases proportionally with each calculated noise level. Because of this, the computational cost of solving the system of equations also increases.

Taking these facts into account, large terrain are then represented by dividing large areas into smaller tiles, so that the resolution of them falls within the optimal result. Once the map has been divided, the method solves each tile independently. It must be taken into account that when they are joined again, the values of the vertices at the edges must be the same. Due to this fact, the method solves every one by establishing these values of the vertices at the edges as new constraints, because they have to match with the corresponding vertices of the adjacent tiles. This ensures that the height of the edges is equal in neighbouring tiles.

3.4 Results and Discussion

In order to test the method presented here, several experiments were conducted using a computer with an Intel Core i7-6700k 4.00 GHz processor. As already mentioned, the method uses restrictions formed by a set of points. These points can form routes (synthetics or georeferenced), irregular or shaped areas, for example to add lakes or areas modified by human hands such as farmland. Furthermore, these points can be used to set heights of interest, such as elevations and low ground areas. Their characteristics have been detailed in subsection 3.4.1. During the experimentation, the MATLAB function LU factorization has been used to solve the system of linear equations, because of its computational efficiency.

3.4.1 Details of the used constraints

Different points sets have been used to test the results of the method. They can be classified in two groups according to the terrain that is going to represent: simple or large terrains. Table 3.1 describes the number of points of each route and the normalized height, that is, the difference in meters between minimum and maximum height of terrain.

Table 3.1. Detail of the data used to test the presented method.

Route	Number of points	Normalized height (m)
Simple	43	13
Snake	192	50
First route	443	65
Second route	668	122
Third route	421	44
Fourth route	208	36.5
Lambda	12,154	20
Flower	23,217	40

Regarding the first group, two different points sets have been considered: hand-made and special figures. Starting from the hand-made paths, two routes have been designed to test the method: one of them called *Snake*, because its form (Figure 3.4(a)), and the other one composed by a low number of points, that has been called *Simple* to identify it (Figure 3.4(d)).

Some works related to the topic use synthetic forms that represent special figures to test the results. So, two of them have been used to comprove that this method can get this kind of constraints. Both are black and white figures, which the points that form the figure have been extracted, so that the method uses them as restrictions. They have been obtained from two research works related to the topic: the first one, called *Flower*, from the work presented by Yoon and Lee [118] and the

Table 3.2. Details of the interest points and areas associated with the real paths.

Route	Interest points	Area to be included
First route	0	No
Second route	0	No
Third route	9	No
Fourth route	12	Circle: 38 points

second one, called *Lambda*, from the work from Zhou et al. [27]. All the points sets in this group have been represented immersed in terrains with a resolution of 256×256 .

The problem to be solved by this work asked to be able to use GPS routes as a restriction. Therefore, in this second group of point sets, four real routes extracted from Wikiloc [119] have been used, a mashup where geo-referenced outdoor routes and points of interest from around the world are stored and shared. These routes represent real routes and areas in the province of Castellón (Spain). They are more complex paths, composed by a higher number of points, with some interest points and areas. Table 3.2 completes the information about these routes. The considered resolution of the grid that represents the terrain has been 1024×1024 , so the process of dividing the grid into 16 tiles has been performed to represent them.

3.4.2 Visual results

The visual results obtained by applying the presented method to the different routes are shown in the following figures. Figure 3.4 illustrates the results for the simplest paths and the synthetic figures and finally Figure 3.5 for the real paths. All of them have been distributed in three columns. The first column shows a bird's eye view of the path or the

Figure 3.4. Visual results obtained for the simplest paths and the synthetic figures.

Figure 3.5. Visual results obtained for the real paths.

set points. The second one shows the turbulence generated for the corresponding set of points, where its location has been marked when it was not clear. Finally, the third column shows the terrain representation associated with the turbulence generated. The set of points have also been highlighted in these images.

3.4.3 Efficiency and accuracy

The study was carried out by analyzing aspects related to efficiency, precision and realism. The experimental results have been divided according to the classification of the paths, and they can be seen in Table 3.3 and Table 3.4.

Table 3.3. Time cost (seconds) and error introduced (%) generating the terrains of the hand-made routes and the special figures.

Route	Number Equations	Noise Levels	Total Cost (s)	Error (%)
Snake	192	6	0.12	16.46
		7	0.21	0
		8	0.54	0
Route simple	83	6	0.07	25.28
		7	0.15	0
		8	0.46	0
Flower	14,895	6	15.37	24.29
		7	29.92	0.37
		8	61.21	0.14
Lambda	12,158	6	11.76	28.52
		7	28.03	0.11
		8	58.16	0

Regarding efficiency, every terrain was obtained computing grids of 256×256 . Only the computation of the terrain that contains the real paths has required to generate tiles, because the resolution of the final

Table 3.4. Partial and total cost (seconds) and the error introduced (%) generating large terrains that contain the real paths.

Route	Number Equations	Noise Levels	Tile Cost (s)	Total Cost (s)	Error (%)
First	1979	6	1.01	16.82	12.92
		7	4.26	69.02	8.83
		8	10.80	174.91	5.37
Second	2204	6	1.79	29.20	11.55
		7	5.21	84.95	7.87
		8	11.91	193.28	3.09
Third	1966	6	0.81	12.99	5.9
		7	4.03	65.43	2.77
		8	10.64	172.12	0.58
Fourth	1794	6	0.23	5.01	6.29
		7	3.31	53.72	3.08
		8	9.87	161.02	0.21

terrain was 1024×1024 . Then, the seconds required to compute the tiles only appear in Table 3.4 for these routes.

The grids of 256×256 were obtained with three different numbers of noise levels (6, 7 and 8), which yield different levels of grid resolution (32×32 , 64×64 and 128×128). The number of equations was the same for all the levels in every experiment. This implies that the points used as constraints in the edges are more separated in lower resolutions. Finally, the tables show the total time spent on calculating the terrain.

To analyze the introduced error of the method, the values of the obtained height map were compared to the original values of the route, with a threshold of 0.025%. The final value for the error is obtained from all the points on the route with values that match the map obtained or which are within the permitted error. This accuracy is also shown in the tables.

3.4.4 Realism

The Unity 3D game engine [120] was used to see the final representation of the height maps obtained by the method. Figure 3.6 shows the results obtained for the second real path. The terrain has been textured to improve the realism and the data set used as constraints in the experiments are marked in pink in the figures. These marks show that the generated terrain is adapted to the established constraints.

Figure 3.6. Different close views of the resulting terrain that includes the Second real path.

3.4.5 Different iterations over the same paths

Apart from the data included in the terrain, a large part of the data is randomly obtained according to the traditional Perlin method. However, applying different iterations to the same path does not return the same terrains. This is because the method forces the data to be included in the terrain to have an accuracy, but as the distance of the points of the grid to the path increases, these constraints do not have any influence on the points and they are completely random.

An example of this situation is shown in Figure 3.7. The path used in this experiment has been the one called *Simple*. Two iterations of the method have been performed with this path and the terrains generated have been different in every one of them.

(a) First iteration: Turbulence.

(b) First iteration: Terrain generated.

(c) Second iteration: Turbulence.

(d) Second iteration: Terrain generated.

Figure 3.7. Results obtained after performing different iterations of the method taking the path called *Simple* as input data.

3.4.6 Validating the presented method

Noise-based procedural terrain generators [114] do not allow the use of precise constraints but often have some attributes that their

(a) 5 octaves.

(b) 7 octaves.

(c) Frequency +1.

(d) Frequency +2.

(e) Persistence 0.70.

(f) Persistence 0.99.

Figure 3.8. Results obtained modifying the number of octaves, the frequency and the persistence.

modification allows to alter the obtained result. The most common attributes to vary are the modification of the octaves, modifying the level of detail of the result, the frequency, which allows varying the roughness of the terrain and the persistence, which modifies the height of the crests and the wave valleys. In order to analyze the method presented and to validate that in addition to meeting the proposed objectives it also allows the variations specified by these other methods without losing precision in the restrictions, it has been tested by modifying these attributes. Figure 3.8 shows the results obtained when modifying the attributes. The path used to test this validation has been the one called *Simple*.

First of all, the number of octaves has been changed to obtain different results of the obtained turbulence. Figure 3.8 shows the terrains generated by establishing the number of octaves to 5 and 7. It can be appreciated in the images that the level of detail of the terrain increases at the same time this attribute does. Next, the frequency has been modified. This attribute varies the distance between the points in a level of noise, so, as can be appreciated in the Figure 3.8, the higher the frequency is, the more abrupt the resulting terrain results.

Finally, the persistence factor indicates the weight of every level of noise in the turbulence. As the level of noise increases, its weight is reduced to half. Increasing the value of the persistence, the final result will have higher roughness because the higher levels of noise will have more influence in the final turbulence. Figure 3.8 shows the results obtained by varying the persistence value from 0.70 to 0.99.

3.4.7 Cases of failure

The method needs that the set of points that is going to be included in the final terrain, meets some conditions to obtain a good result. If all the points that form it are too closely situated or consecutive points

represent really different locations, the terrain obtained will be not valid. An example of this situation can be observed in Figure 3.9.

To avoid this situation, the routes have to be pre-processed to ensure a good distribution of the point data and a coherent sequence of point locations. Then, a good result can be obtained by applying this terrain generation method.

Figure 3.9. Example of a case of failure.

3.4.8 Discussion

As shown in the results, the method presented here is capable of generating large terrains that can be adapted to certain features using procedural modelling. This procedural modelling includes some previously fixed data that represent certain constraints and adds to them the random data that characterize this modelling method. Time spent on obtaining the turbulence map that represents the maps is not high, taking into account that this process is not required to be performed in real time: in fact it took 3 minutes in the case of generating 1024×1024 maps formed by tiles with a resolution of 256×256 . This makes the method suitable for quick generation and later visualization in appropriate

real-time applications such as game engines or simulators.

Regarding the accuracy of the data to be included in the final approximation, the generated terrain includes the routes with a precision of more than 85%, in most cases being above 90%. If the accuracy does not reach 100%, except in case of accepting a major error, it is due to the final joining of the tiles. Sometimes, a small deviation takes place at some of the join points. This is not appreciable to the naked eye, but it affects the value of the accuracy.

The realism of the terrains can be observed in Figure 3.6, where close views can be appreciated. There, it is shown the good quality of the results. Moreover, different executions of the method give as result to different terrains, as it has been shown in Figure 3.7. This is one of the advantages of being based on procedural modelling, which is, the possibility of obtaining different terrains that fit the constraints established by the user.

3.5 Conclusions

This chapter presents a new procedural modelling method to generate terrain that allows to combine in the same representation, some given data with other ones randomly generated. Some point set constraints are taken as input, such as GPS routes, figures or points situated in interest zones, and the method produces a random terrain that includes these constraints with a high degree of precision. To obtain this outcome, the method solves a system of equations based on a noise function. As a result, a procedural texture is generated that is used as a height map. The noise generator method used has been presented by Perlin. This noise, traditionally allows for variations in the final result but not the inclusion of precise data.

Results show that generated terrains include the restrictions with a

high level of realism, in its final representation, and ensuring a smooth continuity between the input data and the randomly generated heights. Moreover, the method maintains the random nature of the Perlin noise, because different terrains result from every run of the method. All the process is performed, as results support, with a low computational cost.

In the tests, it has been demonstrated that our method is able to produce large terrains. In these cases, terrains are divided into tiles, each of them is obtained separately, and they are joined at the end. Visual representations of the generated turbulence maps show the good quality of the results that include the given routes and the interest points, without the appearance of jumps in the terrain.

Chapter 4

Procedural distribution of vegetation

Vegetation is essential in the representation of outdoor scenarios to obtain realism. However, this process can result tedious if plants are manually distributed. This chapter presents a method that procedurally distributes plant models that takes into account features of the land and the species to be placed. The method initially distributes seeds on the terrain and after analyzing the requirements of the plants and the characteristics of the terrain, every seed is labeled with the most appropriated vegetation species to grow there. Different data are evaluated to determine the distribution of species: some related to the terrain, such as the height and the slope, and other ones related to the own vegetation species, such as the size that it can reach or the space it requires to survive. In addition, other processes are also simulated to obtain more realism, such as the grouping of plants of the same species that happens in nature: users are allowed to customize the final representation by pointing where some plant populations have to be located. Finally, once every seed has been labelled, the method simulates a growth of the plants taking into account constraints associated with the space that are available for every representation,

restrained by the neighboring plants that surround each one of them. This simulation provides a solution that maximizes the total area covered, due to the fact that space is considered in the growing simulation. Results show the good performance and the quality of the visual results of the presented technique.

4.1 Introduction

The representation of outdoor scenes is currently a very popular field of research due to the vast amount of applications that require them, such as ecosystem simulation [121, 45], video games [122] or geographic information systems [44]. This fact has made the modelling of these scenarios highly researched in recent decades. Terrain representation has been analyzed in a significant number of works, as it is reviewed by Rebollo et al. and Galin et al. [123, 124]. However, it is necessary to add some vegetation to obtain the realism required in the ecosystem simulations.

In the literature, an extensive study on methods to streamline the modeling of plants and trees [125, 126, 127] can be found, as well as procedural methods [128]. Nevertheless, their automatic distribution is still an open problem. Smelik et al. [13] review in their paper some works that use procedural techniques for the creation of natural environments and virtual worlds. Mainly, they estimate the virtual position of the plants from a procedural calculation, obtaining points that determine their distribution. However, they conclude that these techniques achieve unrealistic results, since they lack the ability to translate ecological input data into a distribution of high density and variety plants.

In order to increase the obtained realism of the outdoor scenes, it is necessary to take into account some botanical information in the procedural placement, such as the biotic and abiotic characteristics of the species, as well as the volume that they take up [129, 130, ?]. Smelik

et al. [13] state that an ecological plant distribution with enough density and variety, where every vegetation species is placed in a pre-fixed position is a difficult challenge to achieve [131, 132, 133, 134]. This chapter presents a method that procedurally distributes plants on a terrain, taking into account a combination of some biotic concepts with others related to the environment. Consider these requirements to distribute the vegetation makes it possible to simulate the real world with high realism.

The presented method procedurally generates distributions of vegetation over height-maps, that can be based on real-world elevation data or synthetic data. The distribution algorithm starts with a perturbed regular grid, which represents the potential vegetable species positions where are allocated seeds. Initially, a sub-set of seeds are evaluated to assign a particular vegetable species to them. This assignment is based on the evaluation of a weighted equation that takes into account the elevation and slope of the terrain and the possibility of establishing a community of a determined vegetable species in an area. This community is set by the user, who can customize the obtained results by positioning some points, denominated *diffusion points*, as the center of that area.

The weighted equation considers for each vegetable species, the min/max ranges of elevation and slope where it can grow. These data are obtained from real-world observations. In case the characteristics of the seed position to be labelled do not meet any requirement associated with any vegetable species to disseminate, it is left empty, indicating that nothing can grow at that point.

Then, the remaining seeds are evaluated to be labelled. This process is based on a k-nearest ($k=3$) neighbour classifier, because the process simulates the fact that the vegetable species can have influence on the plants that grow around them. However, it can be assigned no vegetable species if the location is not appropriate.

The final process of the method is addressed to simulate the growth of the plants. Each seed labelled with a vegetable species computes the maximum circular area that can take the plant, constrained by the maximum volume that the vegetable species can grow and by not overlapping the neighbouring circles. If the final circular area is smaller than the minimum one which that species requires to live, the plant is then discarded.

This method differs from others previously presented in the literature that also deal with the distribution and growth of vegetation, in the following aspects:

- The method can easily increase the amount of considered real-world data to enhance the realism of vegetable species assignment. Currently, elevation and slope ranges are taken from real-world observations, but the method could be extended to incorporate other data related to the growth of plants, such as soil characteristics, land-use data, hours of sunlight per day, water sources, etc.
- User input is simple and optional: points with a given radius influence the vegetable species assignment.
- The algorithm uses simple linear equations with a small set of weights, thus making it easy to customize the results.
- The algorithm replicates the grouping/clustering of species observed in the real world.

The rest of the chapter is organized as follows. Section 6.2 briefly describes the procedural method. Section 6.3 analyzes the distribution of the seeds and Section 6.4 describes the labeling process for every seed positioned on the terrain. Section 6.5 describes the growing process that makes it possible to obtain the high realism. Section 6.6 analyzes the

results obtained by testing the method, and finally, section 4.7 presents some conclusions and some observations related to the future work.

4.2 Plant cover method

The main goal of this method is to distribute vegetation procedurally so as to represent synthetic outdoor environments that simulate the real world. To obtain a realistic representation, the method considers different processes: dissemination of the seeds, labeling, and simulation of growth.

After an initial seed placement, each seed has to be labeled as belonging to a vegetation species. This is performed by obtaining the one that is most suitable for that position on the terrain, according to the biotic data for that species. The labels of the neighboring seeds are also taken into account in this labeling process in order to represent groups of plants of a same vegetation species.

Once each seed has been classified, the method simulates the growth process taking into consideration the competition for space with neighboring plants. Plants that do not reach the minimum size to live are eliminated. This fact differentiates this method from other techniques such as PDDs [59], where the terrain is covered following a growth strategy that starts from a point of origin while maintaining a minimum distance between plants.

Moreover, the fact of delimiting the size that each plant can reach and its effect on the inhibition or expansion on the neighboring plants separates the method from those proposed by Lane and Prusinkiewicz [53] and Alsweis and Deussen[55].

4.3 Seed dissemination

Initially, some areas without vegetation are determined by analyzing the features of the terrain. Some criteria to choose these areas can be, for example, zones situated higher than a certain high, or areas where some paths, lakes or rivers are located. In the case of computer-generated terrains, other ways can be considered to achieve this non-vegetation areas.

In this chapter, a Perlin noise map [62] has been created to facilitate this task. It allows to apply a threshold (between 0 and 1) to create areas without vegetation independent of the height of the terrain. In this way, these areas without plants can simulate grasslands, sandy areas, rocky areas, etc. The value of the threshold can be established by the user or even remove it. Figure 4.1 shows a plant distribution on the ground, where areas with no vegetation are represented in black.

Figure 4.1. Placement of seeds after applying a deviation of the initial position.

Then, the method establishes an initial distribution of the seeds over the terrain represented by a height-map. Although plants in nature are grouped according to environmental conditions, at first glance, they

do not have an organized distribution. The initial positioning follows a regular grid distribution where the seeds are separated by a distance of $d_{spacing} = 5$ meters. For this purpose, a relation of $1 \text{ meter} = 1 \text{ pixel}$ in its representation within the placement map has been considered. This assumption has been established after analyzing the biotic data of the vegetation species to be distributed over the terrain (see characteristics of the species in Table 4.1). This measure satisfies the existence of seeds whose distance to their neighboring seeds is greater than the maximum radius of the circle which that species is capable of covering.

In order to break the regular location of the seeds, the method performs a deviation of the seed positions s_p , by modifying their (x, y) coordinates with a random value. This random value will always be less than half the spacing used in the generation of the grid, in order to prevent two elements from overlapping (see Equation 4.1).

$$\begin{aligned} s_{p.x} &= s_p.x + \left(\frac{d_{spacing}}{2} \cdot random(-0.8, 0.8) \right) \\ s_{p.y} &= s_p.y + \left(\frac{d_{spacing}}{2} \cdot random(-0.8, 0.8) \right) \end{aligned} \quad (4.1)$$

The final result of the redistribution can be seen in Figure 4.1(a), and a closer view can be observed in Figure 4.1(b). Both figures show the deviation suffered by the seed positions, thereby eliminating the unrealistic regular appearance and covering all the terrain surface.

4.4 Labeling the seeds

4.4.1 Evaluating the terrain characteristics

The next process is to determine the species to which each seed belongs. This step is performed only in a sub-set of the distributed seed positions. It is then necessary to consider some features of the terrain,

all of which are available from the height-map. Data obtained from soil maps with vegetation coverage statistics would be really valuable information to be included in this process. However, these covering data are not always available, either because they cannot be achieved or because the terrain used is synthetic.

The slope and the altitude of the seed can be obtained from the height-map. Both characteristics are important for analyzing the possibility of each vegetable species to grow at that certain position. The slope has a direct influence on the quality of the soil, therefore, since there are no moisture maps or soil types, this characteristic allows simulating the soil type and its water absorption capacity. The altitude is a characteristic that influences both temperature and sun exposure: the higher the altitude, the lower the temperature and the greater the sun exposure [45].

The altitude of the seed can be obtained directly in meters from the height-map (Figure 4.2(a)). However, the slope where a seed is situated, s_p , is computed using the following equation:

$$slope(s_p) = \arccos \left(\frac{N \cdot N(s_p)}{|N| \cdot |N(s_p)|} \right) \quad (4.2)$$

where N is the vector normal to the horizontal plane and $N(s_p)$ is the normal vector at the point where the seed is situated. The $slope(s_p)$ is measured in degrees. Figure 4.2(b) shows the slope characteristics of the terrain under consideration.

In their work, Galin et al. [124] state that the ability of the species to live is limited by multiple conditions, so that although a vegetable species has a perfect soil, it cannot grow if it does not have enough temperature or light. In this chapter, this fact has been taken into account, so if the position of a seed is not in the range of the altitude or slope values where the considered vegetable species can grow, it will not be able to develop and will die.

Figure 4.2. (a) Height-map with altitudes ranging between 800 and 1300 meters. (b) Slope map, which varies between 0 and 90 degrees.

A database containing information about the range of heights of the slopes where the vegetation species under consideration can live has previously been stored (see Table 4.2). In this way, if the values of either the height or the slope of the point where the seed is situated exceed the allowed range for that species, then that species is discarded and it is not labeled. In regions where a species can grow, two new variables must be estimated $\{w_{height}, w_{slope}\} \in [0, 1]$, which measure the influence of these features in the labeling process.

The CREAM research center [135] has statistics about the distribution of different vegetation species according to the height and the slope on the terrain in the Community of Catalonia (Spain). On observing these data [136], it can be assumed that the number of plants per species follows a Gaussian distribution. This information is used in our method to obtain the distribution of the plants per species in function of these values.

In accordance with this assumption, the plant distribution depending on the height is analyzed. Let V be the set of vegetation species to be distributed and v is a particular species in that set. Since s_p is the

position of the seed and $height(s_p)$ is its height, the distribution of plants per species $w(s_p, v)$ is given by the following equation:

$$w(s_p, v) = a_v \cdot e^{-\frac{(height(s_p) - b_v)^2}{c_v^2}} \quad \forall v \in V \quad (4.3)$$

where for each species $v \in V$, a_v is the value at the highest point of the peak of the curve of the Gaussian distribution, b_v is the average height value, and c_v is an approximation of the standard deviation of the heights for that species. The reference values a_v, b_v, c_v for each species that have been considered in the experiments can be seen in Table 4.2.

Finally, the distribution of plants per species $w(s_p, v)$ has to be normalized to obtain the influence of the height $w_{height}(s_p, v)$ at a particular seed position s_p of belonging to vegetation species v . It is obtained by means of Equation 4.4:

$$w_{height}(s_p, v) = \frac{w(s_p, v) - \min(w(v))}{\max(w(v)) - \min(w(v))} \quad (4.4)$$

where $\min(w(v))$ is the minimum value of $w(s_p, v)$ for species v considering all the seeds on the terrain and $\max(w(v))$ is the maximum.

Assuming the same reasoning for the slope, the value $w_{slope}(s_p, v)$ would be obtained in a similar way to Equation 4.4, but in this case considering the function $slope(s_p)$ that measures the slope of the terrain where the seed is positioned, instead of $height(s_p)$ in Equation 4.3.

4.4.2 Areas of influence of a vegetation species

In a real outdoor environment, it is common to find plant groups of the same vegetable species. In order to simulate these groupings, the method allows adding areas where a specific vegetable species has more influence of growing. These areas can be generated randomly or can be specified by the user. They are established by adding some

points on the terrain called *diffusion points*, considered as the center of a circular area where the grouping is allocated. Figure 4.3(a) shows an example where some *diffusion points* for three different species have been distributed. They have been differentiated by three colors: blue, red and yellow.

The circular areas associated to them, determine the region where plants of that vegetation species have a high probability of growing, being higher as closer to the point is situated. The area of influence of each *diffusion point* is maximized to cover the maximum possible space on the terrain, as can be seen in Figure 4.3(b). However, in order to prevent the borders between species from being very notable, this area can be increased and intersect with others.

Figure 4.3. (a) *Diffusion points* for three different species, differentiated by three colors (blue, red and yellow). (b) Areas of influence associated with the *diffusion points*.

For each seed on the map, the method searches to determine whether it is situated in the area of influence of one or more *diffusion points*. This value will reach its maximum when it is right in one *diffusion point* and will be reduced until reaching the limit of its area of influence, with a value of 0 just when leaving it. Equation 4.5 illustrates the calculation

performed to measure the influence of each *diffusion point* on a seed s_p as follows:

$$w_{diff}(s_p) = \begin{cases} 1 - \frac{d(s_p, diff)}{r(diff)}, & \text{when } d(s_p, diff) \leq r(diff) \\ 0, & \text{otherwise} \end{cases} \quad (4.5)$$

where $d(s_p, diff)$ is the distance of the seed under consideration (s_p) from the *diffusion point* ($diff$) and $r(diff)$ the radius of the area of influence associated to it.

Finally, the influence of each *diffusion point* set by the user for a vegetation species v has to be evaluated on a seed s_p to analyze its classification. This is shown in Equation 4.6.

$$\forall(diff \in v) \quad w_{diff}(s_p, v) = \max(w_{diff}(s_p)) \quad (4.6)$$

4.4.3 Labeling of the seeds according to terrain characteristics and areas of influence

The next step is to obtain the probability of each seed belonging to a particular vegetation species $w_{sp}(s_p, v)$. Equation 4.7 shows how this probability is obtained by performing a linear combination between the influence of the terrain, w_{height} and w_{slope} , and the influence of the *diffusion points* of that species w_{diff} . The user can modify the degree of importance of each of them by managing some coefficients ($c_{height}, c_{slope}, c_{diff}$). By default, the same value is maintained for the terrain features as for the area of influence, i.e., $c_{height} + c_{slope} = c_{diff}$. If the seed that is being evaluated is outside the area of influence of that species, $w_{diff} = 0$ and the probability is obtained according to the

height and the slope.

$$w_{sp}(s_p, v) = C_{height} \cdot w_{height}(s_p, v) + C_{slope} \cdot w_{slope}(s_p, v) + C_{diff} \cdot w_{diff}(s_p, v) \quad (4.7)$$

The vegetation species v that will be assigned to that seed s_p will be the one that gives $w_{species}(s_p, v)$ its maximum value.

$$\forall (v \in V) \quad w_{species}(s_p) = v : \max (w_{species}(s_p, v)) \quad (4.8)$$

V being the set of the different vegetation species that are going to be present in the outdoor environment.

4.4.4 Adding neighboring information

As it was previously said, it is usual for some areas of the terrain to be populated by plants of the same species. Moreover, it has also been demonstrated that the plants can condition the vegetable species that grows near them, because of some facts such as the dissemination of seeds from them. Some works in the literature have simulated this clusters by means of different techniques. For instance, Lane and Prusinkiewicz [53] uses kernel deformation to create an algorithm to label species by neighborhood.

In the method presented here, it has been considered that each seed conditions the label of its neighbors to create this influence. Consequently, this has been taken into account in the labeling process: a k-nearest neighbor classifier [137] has been used, in our case with $k = 3$ neighbors.

To perform this process, a set of seeds are initially considered as an initial set. All these seeds in the initial set are labeled as a particular vegetation species by applying Equation 4.8. Once this set has been evaluated, the process assesses the rest of seeds by analyzing the

label of the three nearest seeds in the initial set. The label is assigned according to the vegetation species with the majority voting among these 3 neighboring seeds. This allows simulating the neighborhood that occurs in a real environment, allowing some plants to condition the species of the nearby seeds.

Figure 4.4(a) shows the labels applied to the seeds in the initial set, a different color being assigned to each vegetation species. If the position of a seed is not appropriate for any of the vegetation species to be distributed, it is marked in black and an empty label is assigned to it. Figure 4.4(b) illustrates the labels of all the seeds on the terrain after performing the labeling process that takes the neighboring into account. This figure shows some areas where a vegetation species has a strong influence, and some others where different vegetation species appear in the middle. This fact simulates the randomness of nature.

Figure 4.4. (a) Labeling of the training set of seeds. Three vegetation species (blue, red and yellow) have been considered. Black dots represent seeds with no vegetation species assigned. (b) Distribution of the vegetation species obtained by applying the k nearest neighbor classifier on the rest of the points with $k = 3$.

4.5 Vegetation growth

The placement of the seeds and the seed labeling is a process carried out independently of the growth of the plant. To simulate the growth of the plants, this method estimates a circle around the seed similar to the one used in the FON model [55, 56]. This circle determines the maximum area that can be reached by the plant that grows from that seed. Its radius is estimated taking into account the fact that the area occupied by the vegetation on the ground must be maximized. The optimization technique presented by Gimeno et al. [138] is applied to meet this restriction. This growing simulation is performed to search the best solution, the optimal radius of each plant on the map. It takes into account the limitations of the vegetable species assigned to the seed in addition to some general restrictions:

- The circle of each plant cannot intersect with any other plant. This is not true in nature, where a certain overlapping between the tree canopies does exist, but this overlap will be performed during the final visualization process.
- The radius of the circle of each plant cannot exceed the maximum radius that the vegetation species is biologically capable of reaching.
- If the circle associated with the plant does not grow until the minimum radius for the species, that plant cannot survive.

Given a distribution of points that represent the seeds, a distance matrix is constructed. Based on this information, the area occupied by a set of plants on a terrain or domain Ω can be estimated. Let $C_r(s_p)$ be any circle centered on the point s_p with radius r and whose area A is bounded by:

$$A(C_r(s_p)) = \pi \cdot r^2 \quad (4.9)$$

Let $P = \{s_{p1}, \dots, s_{pn}\}$ be a set of n isolated seed positions representing the coordinates where plants have been planted. Considering Ω as the possible locations on the terrain, i.e., $\Omega \subset \mathcal{R}^2$, this set $P \subset \Omega$ is associated with the subset of $\mathcal{R}^2$, given in Equation 4.10.

$$D_{\Omega}(P) = \left\{ (r_1, r_2, \dots, r_n) \in \mathcal{R}^n \mid C_{r_i}(s_{pi}) \subset \Omega \text{ for } i = 1, \dots, n \right. \\ \left. \text{and } C_{r_i}(s_{pi}) \cap C_{r_j}(s_{pj}) = \emptyset \text{ for all } i \neq j \right\} \quad (4.10)$$

where $D_{\Omega}(P)$ corresponds to the solution space to which all the radii found by the method belong. The solution $D_{\Omega}(P)$ found from the above equation guarantees that the circular areas of the plants do not intersect each other. These regions called *circular areas without intersection* (*cawi*) in the set P are determined according to Equation 4.11.

$$cawi(P, \Omega) = \max_{(r_1, \dots, r_n) \in D_{\Omega}(P)} \left\{ \pi \cdot \sum_i r_i^2 \right\} \quad (4.11)$$

The following example clarifies the procedure. Let us consider the case of a domain Ω where three plants of different vegetation species are defined: *Pinus Sylvestris* (A), *Abies Alba* (B), and *Sorbus Aucuparia* (C). Their maximum radii of growth are 5.6, 5.7, 2.7 meters and their minimum radii are 0.2, 0.6, 0.7 meters, respectively. Thus, the distances between plants with values of 7, 8, 9 meters are defined. Figure 4.5 illustrates these data by means of a graph that relates all these distances between a plant of each vegetation species and the vector M , which represents the maximum radius that each plant can reach.

Figure 4.5. Graph analyzing all data of example regarding the three plants belonging to the three species under consideration (A, B, C). Green lines are the distance between the plants and brown lines the maximum radii that each of them can reach until meeting point M .

From the graph, the following system of inequalities can be extracted:

$$\left\{ \begin{array}{ll} r_A + r_B & \leq 7, \\ r_A + r_C & \leq 8, \\ r_B + r_C & \leq 9, \\ r_A & \leq 5.6, \\ r_B & \leq 5.7, \\ r_C & \leq 2.7, \\ r_A & > 0, \\ r_B & > 0, \\ r_C & > 0. \end{array} \right. \quad (4.12)$$

This system of inequalities can be identified by $S \cdot r \leq b$, where S is a matrix of zeros and ones that relates the different plants to each other and to the vector M of maximum radii; r is the radii vector to be solved, in this case (r_A, r_B, r_C) ; and b collects the independent terms of the inequalities. In this example, the solution that maximizes the *cawi*

value for the three plants is given in Equation 4.13.

$$cawi(P, \Omega) = (5.6^2 + 1.4^2 + 2.7^2) \cdot \pi = 127.58. \quad (4.13)$$

On observing the system of inequalities, it can be seen that the constraints do not include the minimum radius that each vegetation species requires in order to be alive. The reason for this is to favor the growth of large trees rather than small trees, since these provide the *cawi* value with a greater numerical result. This decision allows plants with a radius lower than their minimum radius to be eliminated in the same way as is performed with a self-thinning model in the work of Lane and Prusinkiewicz [53].

Finally, the whole process is repeated a second time so that the space left by the plants that are removed can be covered by the rest of the plants. Figure 4.6 shows the result of applying this growth method in the terrain used as an example, by solving the system of inequalities. As can be observed, the previous points with the same size shown in Figure 4.4 now appear re-sized with different magnitudes.

Figure 4.6. (a) The *cawi* obtained is shown for each seed on the terrain. (b) In the detail shown on the right, the different sizes of the circles can be appreciated.

4.6 Results

Several experiments were performed using a computer with an Intel Xeon E5 v2 3.70GHz processor. The trees and textures used in the examples belong to the *Landscape Auto-material* plugin [139]. Unity 3D version 2018.2.7 was used to view the resulting outdoor environments with a realistic visualization.

Three different vegetation species have been considered in the experiments: *Pinus Sylvestris*, *Abies Alba*, and *Sorbus Aucuparia* (Figure 4.7). They have similar characteristics in terms of the type of ecosystem they need to live but in turn they differ in size and in the characteristics of the terrain (height and slope) on which they can grow. The data requirements for each of them were acquired from real data [140, 130, 141] and are summarized in Table 4.1. This table informs about the minimum and the maximum radius that a plant of that species can reach, the minimum and the maximum height where it can live, and the maximum slope where that species can appear.

Figure 4.7. Species of trees used, from left to right: *Pinus Sylvestris*, *Abies Alba*, and *Sorbus Aucuparia*.

Table 4.1. The selected species data. The minimum and maximum radius that the specie can reach and, regarding the terrain, minimum and maximum height and maximum slope where each it can grow. All these data are measured in meters and the slope in degrees.

vegetation species	Radius		Height		Max slope
	min	max	min	max	
<i>Pinus Sylvestris</i>	0.2	5.6	1,000	2,000	70
<i>Abies Alba</i>	0.6	5.7	700	2,000	60
<i>Sorbus Aucuparia</i>	0.7	2.7	0	2,300	30

In order to evaluate the influence of the terrain features in the labeling of a seed, the values a_v, b_v, c_v are required for each species to compute Equation 4.3. These values were obtained from the Growth-Grammars [63]. They are shown in Table 4.2, classified according to the height and the slope of the terrain.

Table 4.2. Estimated values used to calculate the number of plants per species w in function of the height and the slope (Equation 4.3).

		<i>Pinus Sylvestris</i>	<i>Abies Alba</i>	<i>Sorbus Aucuparia</i>
Height	a_v	44,200	6,060	2,056
	b_v	1,041	1,553	1,396
	c_v	584.7	230.6	346.7
Slope	a_v	46,582	2,834	1,401
	b_v	36.60	50.21	49.01
	c_v	26.91	27.52	25.57

Regarding the *diffusion points*, the ones that have been considered are those shown in Figure 4.3. The labeling process was performed following a 3 nearest neighbor classifier, as was previously informed. Finally, the system of inequalities has been solved by applying the Simplex optimization technique, since this technique allows to obtain the best solution without restriction in the number of variables, thus allowing to be able to expand these variables in case of requiring it in the future.

4.6.1 Validation of the method

The method has been tested using a height-map with a resolution of 512×512 meters. As has been said in previous sections, the relation established between pixels and meters has been $1 \text{ meter} = 1 \text{ pixel}$. Regarding the height of the terrain, a height level difference of 500 meters has been represented between the lowest and highest area, with a range of values that varies between 800 and 1,300 meters.

Some tests have been performed to validate the procedurally created outdoor environments, specific modifications being applied in each execution. Initially, areas with no vegetation are established by limiting the height values: areas higher than 3,000 meters are considered dry zones. The same map has been used in all the experiments to make it easier to notice the visual differences when some parameter values change.

Three different vegetation species have been considered and differentiated by colors: *Abies Alba* is represented in red, *Pinus Sylvestris* is shown in blue, and *Sorbus Aucuparia* is shown in yellow. Different variations of parameters have been tested in order to see how the distribution changes according to the user preference. All the data involved in the tests are shown in Table 4.3 and the results obtained can be seen in Figure 4.8.

The first test has been called *Basic map* (Figure 4.8(a)). Initially, the values that weight the parameters involved in the labeling process of the training test have been set to $(c_1 + c_2 = c_3)$. They give the same relevance to the terrain characteristics as the areas of influence established by the user. In this case, the values considered were $c_1 = 0.25$, $c_2 = 0.25$ and $c_3 = 0.5$. The visual result shows that the different plants appear grouped by species because the area of influence has played the leading role. As indicated in Table 4.3, with a spacing between the plants of 5 pixels, 7,581 seeds were initially distributed, but after applying the

requirements for growth, 2,145 plants were eliminated. This test has required the resolution of a system of inequalities made up of 7,581 inequalities.

The second test, called *Height relevance* in the table, evaluates the visual results when the influence of the height is increased. The values given to the coefficients in this case were $c_1 = 0.6$, $c_2 = 0.2$, $c_3 = 0.2$. The same distribution and number of seeds were set. These data also produced the same number of inequalities as in the previous test. However, in the end, more plants were eliminated: 2,476. Figure 4.8(b) illustrates the visual result. Plants of different species appear to be more mixed than in the previous test. This is because all of them have a similar range of height where they can grow, as can be seen in Table 4.3.

Table 4.3. Details of the parameters involved in the experiments performed.

Experiment name	Terrain Resolution	Initial Seeds	Inequality Number	Eliminated Seeds
Basic test	512×512	7,581	37,905	2,145
Height relevance	512×512	7,581	37,905	2,476
Increasing Heights	512×512	7,581	37,905	2,434
Increasing spacing	512×512	1,933	9,665	242
Doubling no. species	512×512	7,581	37,905	2,108
Larger map size	1024×1024	35,969	179,845	5,428

The third test was performed by modifying the range of heights of the terrain. Maintaining the variation of 500 meters, the scale of heights was modified from 1,900 to 2,400 meters. Regarding the data shown in Table 4.3, it can be observed that establishing the same number of initial seeds, the number of plants eliminated was quite similar to that of previous tests (2,434). Figure 4.8(c) shows the visual results.

On the one hand, an area without any plant can be appreciated. This

(a) Basic test.

(b) Height relevance.

(c) Higher Heights.

(d) Increasing the spacing.

(e) Doubling no. species.

(f) Larger map size.

Figure 4.8. Coverage maps obtained by modifying some features. Vegetation species: *Abies Alba* (red); *Pinus Sylvestris* (blue); *Sorbus Aucuparia* (yellow); *Pinus pinea* (magenta); *Quercus robur* (cyan); *Betula pendula* (green).

area is the highest one on the map, so, by increasing the height range above the maximum height where the three species can grow, an empty area appears.

On the other hand, the plant represented in yellow, *Sorbus*, have considerably increased the number of instances because the maximum height where it can grow is higher than the other two vegetable species. In the low-lying areas, however, the dominant vegetable species becomes *Pinus Sylvestris* (blue), given that it has the minimum height lower than the red element, *Albies Alba*, and a higher slope value than the other two species.

Finally, it can be appreciated in this figure that despite the fact that there is less area covered by vegetation, the number of eliminated seeds is similar to the rest of the tests. This is because the dominant vegetable species has a smaller maximum size than the other two, therefore, more elements fit in the same area than in the other tests with other dominant species.

Regarding the fourth test, the initial spacing between seeds increased twofold, resulting in 10. This has been called "Increasing spacing". In this case, as more space is required, a lower number of seeds have been distributed: 1,933. This produced only 9,665 inequalities that, after solving them, meant that 242 plants eventually died. Figure 4.8(d) illustrates the visual result of this test. It shows that if plants have more space available around them, most of them can grow to their maximum size, thus reducing the number of those eliminated.

The fifth test doubled the number of vegetation species involved in the experiment, resulting in 6 vegetation species. The identification "Doubling no. species" has been applied in the table. Figure 4.8(e) shows the visual result of the experiment. In this case, the new vegetation species considered are: *Pinus pinea* (represented in magenta), *Quercus robur* (cyan), and *Betula pendula* (green). The characteristics regarding

the crown radius are obtained from Pretzsch et al. [130] and the range of the terrain features (height and slope) have been considered to be similar to the three vegetation species shown in Table 4.1. Figure 4.8(e) shows how the competition increases and the size of the groups of vegetation species are smaller.

Finally, the last test was performed doubling the resolution of the terrain. In this case, a resolution of 1024×1024 meters was established. Considering a double-sized terrain and maintaining the same spacing between seeds, 5 meters, the number of seeds distributed increased to 35,969, and so the number of inequalities reached 179,845. In the same way, the number of plants that are finally deleted increased to 5,428. Figure 4.8(f) shows the visual results obtained by applying the method on this large map. This image shows that there is a greater number of elements, but the competition between them is not very different from the results obtained using the initial map size.

4.6.2 Computational time analysis

The computation time of the method is conditioned by the number of inequalities that have to be solved to simulate the growing process. As stated in section 6.5, they are stored as $S \cdot r \leq b$, where S is the matrix storing 1 or 0, r is the radii vector to be solved, and b represents the independent terms of the inequalities. Some of the inequalities considered control the space each plant occupies according to its maximum radius, while others control the distance from that plant to the rest of the plants distributed over the terrain. On analyzing the size of this matrix, S , it is observed that given a set of n plants, the number of inequalities that should be incorporated if all possible distances are considered is determined by Equation 4.14.

$$2 \cdot n + \frac{n \cdot (n - 1)}{2} \quad (4.14)$$

However, the number of inequalities can be reduced by considering that the resulting value of the plant radius is directly related to the distance to its neighboring plants instead of to all of them. This approach is a critical aspect in order to reduce the number of inequalities and make the method effective at the computational level.

Figure 4.9. Results obtained when applying the distribution in a rectangle of 40×60 meters considering different number of distances.

Several tests have been carried out to choose the appropriate number of neighbors without producing crown overlapping between plants. According to their results, for each vegetation species only the distances related to 5 nearest neighboring plants are required. This approximation reduces the number of inequalities to the value shown in Equation 4.15. Nevertheless, it has been demonstrated experimentally that this number can be reduced by up to 40% if the distances that are repeated between neighbors are eliminated.

$$2 \cdot n + 5 \cdot n \quad (4.15)$$

Figure 4.9 shows an example of the distribution of plants of the vegetation species *Pinus Sylvestris*. The terrain was considered to be a 40×60 meter rectangle where the seed placement followed a uniform distribution. Figure 4.9(a) illustrates the results of this distribution, where all distances are considered, while in Figure 4.9(b) only the 5 nearest

neighbors are taken into account. The figures show that by considering the distance of only the 5 nearest plants, the method associates to each seed a circular area that does not intersect with any other and produces a good growth proposal.

Finally, in order to analyze the computational cost regarding the number of plants, some populations of *Pinus Sylvestris* have been placed over an area of 512×512 meters using this assumption of considering the 5 nearest neighbors. Figure 4.10 analyzes the results of this experiment. It can be observed how the time in seconds tends to grow exponentially with the number of seeds distributed over the terrain.

Figure 4.10. Evolution of computing costs with respect to the number of trees.

4.6.3 Visual results

The process of obtaining the final terrain is illustrated in Figure 4.11, in accordance with the experiment called *Basic test*. The terrain has

been textured to show the placement of the different vegetation species and the difference in growth between the plants. Colored circles have been used to offer a visual representation of the relationship between the coverage obtained for the map under consideration and the final terrain with trees. Figure 4.11(a) displays a top view of this coverage map and Figure 4.11(b) shows a perspective view, which allows the characteristics of the terrain to be observed visually.

(a) Top view of the result.

(b) Perspective view of plant covering.

(c) Final result with trees.

Figure 4.11. Visual process to obtain the final terrain with trees from the height-map.

Finally, Figure 4.11(c) illustrates the final visual result, by changing the circles for the corresponding trees. The realistic model where each plant is represented allows us to value how the characteristics of the terrain

have conditioned the final results. Apart from the initially established areas without vegetation, the feature that has determined the lack of vegetation has been the slope. The height was not relevant because the maximum height considered in the experiments (1,300 meters) has allowed all the species in the test to grow.

Figure 4.12 illustrates the results in more detail. The *Landscape Auto-material* plugin that simulates vegetation and floral fauna has been used in the representation, so the ecosystem generated can be better appreciated. This figure shows three captures of the visual results produced at different distances. Figure 4.12(a) shows the different vegetation species in the outdoor environment at a long distance. Moving closer to the camera, Figure 4.12(b) shows the scenario at a medium distance. Here, the lack of vegetation can be seen in areas with high slopes. Finally, Figure 4.12(c) illustrates a shot from the ground where the good quality of the visual results can be seen.

4.7 Conclusions

This chapter presents a procedural modeling method for natural environments that, based on biological restrictions, distributes plants over a terrain. It starts by performing an automatic positioning of seeds, which are later labeled as a particular vegetation species. Different variables condition these labels, mainly based on the characteristic of the position of the seeds on the ground and on the biological requirements of each vegetation species. The labels of the k -neighboring seeds are also considered to represent the typical grouping of plants present in nature. Users are allowed to partially condition the final labeling process by establishing some areas where these groups of plants have to be allocated.

The final realistic representation is obtained by simulating the growing process from the seeds. Maximum coverage of the terrain is reached by

allowing the plants to grow until the maximum size, according to their biotic data. Intersection between the volumes assigned to each plant is not allowed in the simulation process, although some branches can reach other plants in the final realistic representation.

Results show that the generated outdoor environment maintains the requirements given for each vegetation species. Furthermore, the visual results are realistic and natural due to the heterogeneity in the size and position of the plants, the grouping achieved between species, and the fact that some elements of another species grow among them.

(a) Long distance.

(b) Medium distance.

(c) Close distance.

Figure 4.12. Visualization of the result obtained at long, medium and close distances.

Chapter 5

Simplification techniques for vegetation

Plants and trees are an essential part of outdoor scenes. They are represented by such a vast number of polygons that performing real-time visualization is still a problem in spite of the advantages of the graphic hardware. Some point-based or image-based rendering methods solve this problem. But in some interactive applications the geometric representation of the vegetation is required, so these solutions do not work.

This chapter presents a simplification method that deals with the geometry of the foliage, reducing the number of primitives that represent these objects and making their interactive visualization possible. The method metric is based in a simplification image-based that establishes an order of leaf pruning and reduces the complexity of the canopies of trees and plants. The simplification in the proposed method is based on points of view and uses mutual information to choose the blade to prune. The error introduced every time a leaf is pruned, is compensated by altering the size of the nearest leaf to preserve the leafy appearance of the foliage. When the tree is pruned in this way, the leaves begin to

withdraw from the interior of the tree and in the same way, the method begins to alter the size to the leaves located inside. In addition, this form of pruning avoids the appearance of an unpopulated tree or with leaves that are too unreal in close views, which usually produced when a foliage representation is formed by a reduced number of leaves. Results demonstrate the good quality and time performance of the presented method.

5.1 Introduction

Outdoor scenes are very common in many interactive applications, such as video games or virtual reality walk-throughs. Vegetation is an essential part of these environments, producing a lack of realism if some plants do not appear in the scenes. Tree and plant modeling has been widely researched, and different techniques have appeared in the literature to represent these objects. However, real-time visualization of forest or environments with vegetation is still a problem in spite of the current development of the graphics hardware. The main reason for this, is that tree models are formed by such a vast number of polygons that interactive visualization of scenes with vegetation is practically impossible.

Methods that have appeared to perform their real-time visualization can be mainly classified into two groups: image-based and geometry-based rendering. The methods in the first group, image-based rendering, change the geometry of the plant by an image that represents it. The advantage of these methods is that visualization time is drastically reduced compared to geometry rendering. However, the lack of realism is remarkable when the camera approaches the object. The other group in the classification, the geometry-based rendering, can be divided in turn into two groups: point-based rendering and rendering based on polygons. The first group represents the leaves by points or even lines

when they are situated far from the viewer. The polygon-based rendering allows a realistic visualization even when the camera is really close to the plant. Some applications perform this rendering method because the detail of the leaves in the foliage is required. Its main drawback is the vast amount of polygons, which delays the interactive visualization if an optimization technique is not applied.

Multiresolution modeling [142, 143] has been successfully presented as a solution to the problem of efficient manipulation of highly detailed polygonal surfaces. Discrete multiresolution modeling consists of representing an object by means of multiple approximations or levels of detail (LoDs) where each approximation or LoD represents the original object using a different number of polygons. Interactive applications manage this set of LoDs and change the approximation to be rendered in real time taking into account some parameters, such as the distance to the viewer or the amount of pixels the object occupies in the final image. Game engines, such as Unity 3D [120], allow game designers to use this solution by including a component called LoD Group where different approximations of the same object can be uploaded (LoD1, LoD2, etc.), but only one of them will be rendered in the interactive application, depending on the established conditions. The distance to the camera is the most used criterion to change the rendered LoD.

Simplification methods deal with the task of reducing the amount of geometry in the meshes, without drastically affecting appearance [144, 145]. These methods allow to generate the different LoDs that are necessary to create the set of approximations that compound a discrete multiresolution model [142, 143].

The problem with these simplification techniques is that they have generally been aimed at working with continuous meshes. However, the part of the foliage in the plant elements does not meet this restriction. These elements are formed by independent polygons, the leaves, where the image of the leaf is textured on them. The fact that these methods

do not work correctly with isolated polygons has been proven [66]. After applying an automatic general simplification process, the appearance of the resulting foliage undergoes a too drastic change. Since the number of leaves is less than before, the tree appears less leafy.

Some simplification methods specially addressed to foliage part of the plants have appeared in the literature, such as the one presented in Figure 5.1 by Remolar et al. [1]. Where the authors propose the leaf collapse as a simplification operation: two leaves are transformed into a single one, so that the area of the new leaf is similar to the area initially formed by the two leaves. Other works, such as presented by Zhang and Blaise [71], deal with the same collapse operation; changing the criteria, they select the leaves to be collapsed by some measurements of similarity, such as the distance and diameter of the leaf.

Figure 5.1. Example of foliage simplification performed by the method presented in [1]. **(a)** Original: 20,350 leaves; **(b)** simplified: 2,348 leaves.

Other works apply stochastic pruning to the foliage, such as the works presented by Cook et al. [75, 76]. In their work, some elements are pruned (i.e., eliminated), and the remaining elements are altered to preserve the overall appearance of the scene. The elements that are pruned are

randomly chosen.

This chapter presents a simplification method addressed to the foliage of the plants that is driven by the visual appearance. A viewpoint-based simplification method that manages an information-theoretic measure called viewpoint mutual information (VMI) [91] has been adapted to the geometric simplification of the sparse part of the plants. In this method, a set of cameras have been uniformly distributed around the foliage and the simplification of every leaf is evaluated, checking the introduced error in every one of the images in the image set obtained from the different cameras. Removing the leaves that introduce the lowest error makes it possible to increase the degree of simplification in hidden interiors because they do not have visual impact. This allows us to obtain a better preservation of the visible parts of the model. Another important feature of our algorithm is that the size of some remaining leaves is altered to preserve the leafy appearance of the representation of the tree canopy. Experimental results show that our method yields better visual performance, comparing the results with those produced by applying the method presented in [1].

This chapter is organized as follows. Section 5.2 presents the metric used to choose the leaf that introduces the lowest error. Then, Section 5.3 analyses the presented simplification scheme. The results are shown and evaluated in Section 5.4, and finally, Section 5.5 exposes the conclusions of the work and some proposed future lines of research.

5.2 Viewpoint-based error metric using mutual information

The metric used to simplify the foliage of the plants in the presented foliage simplification method is based on the information-theoretic measure called viewpoint mutual information (VMI). This method use mutual information (MI) in [146] as a shape descriptor for object

recognition, which is suitable for capturing the shape variation. Moreover, works presented in [85, 86] introduced VMI to select the best views, also demonstrating that the VMI depends on the shape of the object and not on the resolution of the mesh.

Next, the metric used to obtain the error in the VMI will be formalized. Let V be a set of viewpoints and O the set of polygons of an object. Viewpoints will be indexed by v and polygons by o . Viewpoint entropy, based on Shannon entropy, has been defined [87] from the relative area of the polygons projected over the sphere of directions centered at viewpoint v .

The marginal probability distribution of V is given by $p(v) = 1/N_v$, where N_v is the number of viewpoints. That is, the same probability is assigned to each viewpoint, although other distributions could be used. Let a_o be the area of the polygon o projected over the sphere, N_O be the number of polygons of the object and $a_t = \sum_{i=0}^{N_O} a_i$ the total area of the projected polygons over the sphere. The conditional probability $p(o|v) = a_o/a_t$ is defined by the normalized projected area of polygon o over the sphere of directions centered at viewpoint v . Finally, the marginal probability distribution of O is given by $p(o) = \sum_{v \in V} p(v)p(o|v) = (1/N_v) \sum_{v \in V} p(o|v)$. Finally, VMI is defined according to Equation (5.1), which represents the degree of correlation between the viewpoint v and the set of polygons O .

$$I(v; O) = \sum_{o \in O} p(o|v) \log \frac{p(o|v)}{p(o)} \quad (5.1)$$

According to [85, 86], the difference shown in Equation (5.2) quantifies for a given viewpoint v the variation in the shape of that polygonal object O to O' , viewed from that point of view.

$$|I(v; O) - I(v; O')| \quad (5.2)$$

The work presented by Castelló et al. [91] evaluates every simplification step that changes a polygonal object O to O' performing edge collapse in the geometry. The error associated with every one of these collapse operations C_e is defined by the sum of variations in VMI for all viewpoints (Equation (5.3)).

$$C_e = \sum_{v \in V} |I(v; O) - I(v; O')| \quad (5.3)$$

In this method, this idea is applied to measure the variation of the shape of the foliage F , not composed by continuous meshes. As in [91], a set of viewpoints V has been considered in order to cover the object from different views. The distribution of the cameras around the foliage has been performed taking into account that VMI is sensitive to the distance of the object. In our case, the considered object is the foliage of a plant, so a simplification operation involves the elimination or removal of a leaf. Taking these facts into account, the error introduced by a leaf removal L_{pru} is defined by the sum of the VMI variations for all the viewpoints v in V :

$$L_{pru}(l) = \sum_{v \in V} |I(v; F) - I(v; F')| \quad (5.4)$$

where F' represents the foliage without the visualization of the leaf l .

Moreover, following the method employed in [91], the technique for computing viewpoint entropy based on projected areas is the hybrid SW-HW histogram. This technique takes advantage of the PCI Express bus symmetry. A different color is assigned to each leaf, composed by a combination of polygons where the image of the leaf is textured, and the whole object is sent for rendering. Next, a buffer read operation is performed, and then, this buffer is analyzed pixel-by-pixel to retrieve data

about its color. Using RGBA color encoding with a byte value for each channel, up to 256^4 polygons can be calculated with only one single rendering pass. This technique is used during the simplification process.

Another concept to consider is the resolution of the images generated by every camera that have to be compared. The method implemented for estimating the error is based on the projected areas of polygons. Depending on the orientation of the leaves in the foliage, problems can appear when this error is obtained, especially in the case of tiny or long and thin triangles whose projections do not cover a single pixel. One possible solution to reduce these errors is to increase the image resolution, but this obviously penalizes the performance of our simplification method. According to the work [91], the optimum resolution of the images used to determine the error is 256×256 . This chapter demonstrates that this resolution is the best option to balance time computing and shape preserving.

5.3 Simplification scheme

Trees and plants can be easily divided in two different parts: the solid component, i.e., the trunk and the branches, and the sparse component, the foliage or leaves. In order to offer a whole simplification of the plant, every part is separately analyzed and simplified. Figure 5.2 illustrates the process followed to achieve a coarser LoD.

The trunk is formed by a set of polygonal meshes. Work presented in [91] performs the viewpoint-based simplification of general meshes driven by the VMI, so this method has been applied to this part of the plant. Branch simplification, as is shown in the bottom of Figure 5.2, illustrates the result obtained applying this method.

Foliage representation is different depending on the tool that has been used to generate the plant. Xfrog [147] and Speedtree [5], two

Figure 5.2. Workflow for obtaining a simplified approximation of a tree, finally formed by 506 polygons. VMI, viewpoint mutual information.

of the most popular software programs that deal with plant modeling, usually represent the leaves using a quad formed by two triangles or a combination of them. Some of these polygon combinations can be appreciated in Figure 5.3. This is a very common representation in trees that have been designed for interactive applications, such as video games.

In order to obtain an LoD of the foliage composed by a lower number of leaves, the simplification operation has to be performed iteratively until the desired number of leaves is achieved. In the case of the presented method, the simplification operation is the leaf removal.

5.3.1 Leaf Removal Operation

The simplification process is based on the leaf removal operation. As well as the work presented in [76], this operation implies eliminating a whole leaf of the foliage. In every iteration, our simplification method

Figure 5.3. Example of leaves represented by different polygon combinations. (a) Leaf representation by quads; (b) leaf representation by a mesh with several polygons.

evaluates the associated cost to prune every one of the leaves in the foliage, comparing the obtained images with the images of the original plant and associating a cost to every pruning L_{pru} (Equation (5.4)). The leaf firstly removed of the representation will be the one that introduces the lowest deviation in the canopy.

Once the leaf that introduces the lowest cost is determined, the next step is to maintain the leafy appearance of the foliage. Then, its nearest leaf is located and its size scaled in order to cover an area that avoids appreciating the gap produced by the pruning in the final approximation. This nearest leaf is the one that makes the geometric distance between the center of the combinations of polygons that represents the leaves lowest. The developed method takes into account the size of the leaf to prune and the deviation L_{pru} that is going to introduce to the final representation after its removal. Let l be the leaf to be pruned and l' its nearest leaf. Let $Size(l)$ be a function that calculates the size of the leaf that it takes as input. Then, the size of the nearest leaf is altered following the equation shown in Equation (5.5). Figure 5.4 illustrates this process in order to get a better comprehension.

$$Size(l') = Size(l) + Size(l) * L_{pru}(l), \quad L_{pru}(l) \in [0, 1] \quad (5.5)$$

Leaves that introduce a very low error in the representation after their removal hardly vary relative to the size of the leaves nearest to them. By contrast, leaves that produce a high error after the disappearance of the visualization, compensate for this change of appearance by increasing the size of the nearest leaves in order to cover the opening that they have produced. In this case, the nearest leaf is resized to cover its own area and that of the pruned leaf.

Figure 5.4. Example of resizing a leaf that covers the pruned one (drawn with dashes on the right).

5.3.2 Evaluating the Number of Viewpoints

The number of viewpoints condition the good quality of the results. However, it has to be considered that the more cameras are used, the higher the computing time to extract the LoD is. Works related to view-dependent simplification [91, 79] demonstrate that in order to get a trade-off between accuracy and cost, it is appropriate to perform the measurements with 20 regularly-distributed viewpoints and rendered 256×256 resolution images. In this chapter, a study has been

performed to analyze the number of cameras that allow to obtain a better simplification. The foliage used in the test shown in Figure 5.2, composed of 20,376 leaves.

Different configurations have been tested in the experiment. First of all, 20 cameras were distributed on the vertices of a dodecahedron of one unit of radius that surrounds the object (shown in Figure 5.5(a)). Next, 12 view-points have been situated around the foliage following the vertices of a icosahedron (Figure 5.5(b)), and finally, a hybrid configuration of the cameras has also been tested, referred to as the variable number of cameras (VNC). This last configuration takes into account the percentage of simplification that is going to be performed, establishing four ranges in the method. Simplifications in the range of [0%, 10%], 20 cameras, [11%, 30%] of simplification, 12 view-points, the range between [31%, 50%] 8 cameras, and finally, until 90% (6) cameras are located around the object.

Figure 5.5. Camera positions that have been used in the test. **(a)** Dodecahedron: 20 vertices; **(b)** icosahedron: 12 vertices.

To measure the quality of the simplified models, another 12 different viewpoints have been distributed around every representation. These cameras are distributed following the points of the icosahedron shown in

Figure 5.6. Error according to the number of cameras used. (VNC: Variable number of cameras).

Figure 5.7. Time cost according to the number of cameras used. (VNC: Variable number of cameras).

Figure 5.5(b). The VMI error was calculated using perspective projection, a 60 degrees field of view, a 3.0 radius for the viewpoint sphere and a 512×512 image resolution with flat shading. This fact makes the evaluation of the introduced error more exact. Results are shown in Figures 5.6 and 5.7.

After testing the presented simplification method, the configuration of the 20 cameras has been found to be fairly satisfactory. The simplification process is an off-line process that is not performed in real time. The most wanted requirement in this kind of process is the quality of the results, regardless of the time required to get them. Figure 5.6 shows how the error of the resulting LoDs obtained by evaluating every leaf removal with 20 cameras is more similar to the original representation than the obtained with a lower number of cameras. Even when the obtained LoD has 90% less leaves than the original foliage, the error between both representations is about 5%. This process took 5000 s to complete, but the quality of the results are worth it.

5.3.3 Simplification Algorithm

The data structure that has been designed to perform the simplification of the foliage is shown next. The main one is the *Foliage*, which is compounded by a list of leaves. The data of every leaf are stored in the structure *Leaf*, where the edges that compound the leaf are referenced. Furthermore, the central point of the area that they occupy is stored in *central_point*, and the size of the leaf is stored in *area_size*. A list of all the edges of the foliage is stored in the structure *Edge*, where each one of the edges keeps the vertices that determine them. Finally, the data structure *Vertices* stores the geometric and the mapping coordinates.

```
Struct Edge
    Vertices[2] vertices;
}
```



```

Struct Vertices
    float[3] vertex;
    float[2] uv;
}
Struct Leaf {
    Edge *edges;
    float[3] central_point;
    float area_size;
}
Struct Foliage {
    Leaf *leaves;
}

```

The pseudocode of the implementation is shown in Algorithm 2. Due to the nature of the mesh that forms the foliage, the leaf removal cost has to be calculated every time a leaf removal is proposed. A removed leaf can modify the visibility of some leaves situated in different areas of the foliage, not only the nearest ones. Then, the VMI has to be computed every time a pruning is evaluated in order to obtain the cost of that simplification operation.

Let *Leaves_Simplification_Level* be the number of leaves that have to finally form the simplified representation of the foliage and *number_Simplification_Level* be a function that counts the number of leaves in the current representation. Every iteration of the algorithm will eliminate the leaf that introduces the lowest cost and will increase the size of the nearest one. Every time a leaf pruning is evaluated, the cost of this operation is checked in order to find the leaf that minimizes it. Then, this simulation is reversed to continue checking the rest of leaves in the representation. When all the leaves have been processed, the one with this minimum cost is pruned and its nearest leaf scaled. This process is repeated until the number of leaves that has to form the simplified LoD (*Leaves_Simplification_Level*) is achieved.

Algorithm 2 Process of leaf simplification.

```

// Initialize the number of cameras
 $n = 20$ 
// Compute the initial VMI for the foliage F
Compute  $I(v, F)$  where  $v = \{1, \dots, n\}$ 
// Initialize the set of leaves to evaluate
 $F' = F$ 
// Repeat the process until the number of leaves reaches the
simplification level
while  $\text{number\_of\_leaves}(F') > \text{Leaves\_Simplification\_Level}$  do
     $\text{min\_cost} = 1000.0$  // Initialize the cost
    for  $l \in F$  do
        remove  $l$  to obtain  $F'$ 
        Compute  $I(v, F')$  where  $v = \{1, \dots, n\}$ 
         $L_{\text{pru}}(l) = |I(v, F) - I(v, F')|$ 
        if  $\text{min\_cost} > L_{\text{pru}}(l)$  then
             $\text{leaf\_to\_prune} = l$ 
             $\text{min\_cost} = L_{\text{pru}}(l)$ 
        end if
        Undo removal
    end for
    // Remove  $\text{leaf\_to\_prune}$ 
     $F' = F' - (\text{leaf\_to\_prune})$ 
    // Obtain the nearest_leaf of  $\text{leaf\_to\_prune}$ 
     $\text{nearest\_leaf} = \text{Near\_Leaf}(F', \text{leaf\_to\_prune})$ 
    // Scale the size of nearest_leaf according to Equation (5.5)
     $\text{nearest\_leaf.area\_size} = \text{Resize}(\text{nearest\_leaf}, l, \text{min\_cost})$ 
end while

```

5.3.4 Algorithm Complexity

The total temporal cost of our algorithm depends on the number of viewpoints V , the number of pixels (image resolution) P and the number of leaves L in the representation. Calculating VMI implies a cost of $O(PV)$. The cost of obtaining a coarser representation depends on the final number of leaves that remain in the final simplification. The pruning

operation only has a cost of $O(PVL)$. Finally, if considered the worst case, that is all leaves are removed, the total complexity is $O(PVL^2)$.

5.4 Results

Some experiments have been carried out in order to test the presented simplification scheme. They have been performed on a computer with an Intel Core i7-6700k 4.00 GHz with 16 GB RAM and NVIDIA GeForce GTX 1080 graphics card.

All the geometric models of plants that have been used in the tests have been modeled using the commercial modeling tool Xfrog [147]. Our method has been implemented taking advantage of the graphics hardware, so for geometric visualization standard OpenGL be have used running on current GPUs. To render our images, vertex buffer objects have been used, and for the off-screen rendering, the OpenGL frame buffer object extension has been employed.

5.4.1 Comparing the Simplified Approximations

In order to evaluate the good quality of the results, some plants have been simplified and compared with the approximations obtained by applying the method presented by Remolar et al. [1], called FSA (foliage simplification algorithm) and an implementation of the stochastic method (SM) presented by Cook and Halstead [75]. The chosen models are shown in Figure 5.8.

Visual comparisons are shown from Figures 5.9 to 5.10. Three levels of detail have been computed for every one of the plant models used in the test: the ones that approximately maintain 50%, 25% and 10% of the leaves. All the involved data are detailed in Table 5.1. The data related to the presented method are identified as VDS (viewpoint-driven simplification). First of all, the geometric details of every one of the

plants that have been used in the test are described in it, adding the number of leaves of the resulting approximations.

Figure 5.8. Trees and bushes used in the experiments that compare both methods. **(a)** English oak: 20,376 leaves; **(b)** *Carya illinoensis*: 8140 leaves.

Table 5.1. Experimental data of the developed test comparing both simplification methods. VDS,viewpoint-driven simplification.

Model		English Oak				Carya illinoensis	
Leaves	Original		20376			8140	
	Final	10175	5087	2035	4070	2035	814
VMI ERROR	FSA	12.01	22.09	45.17	32.42	64.31	77.86
	VDS	3.71	4.48	5.31	5.12	6.64	14.65
	SM	19.75	25.26	32.25	7.54	14.38	37.42
Time (s)	FSA	98.32	101.64	105.93	40.13	45.58	48.23
	VDS	4010.27	4501.14	4938.48	901.23	1018.97	1234.29
	SM	81.36	83.64	85.81	34.08	37.18	39.92

Next, the quality of the simplified models has been measured using the mutual information (VMI). In this case, the final results obtained by both methods have been compared with the original models, taking the

Figure 5.9. Visual comparison between the methods applied to the model tree shown in Figure 5.8a: Top. the stochastic simplification method (SM); Middle. FSA simplification; bottom. the VDS results. From left to right: 50%, 25% and 10%.

Figure 5.10. Visual comparison between the methods applied to the model tree shown in Figure 5.8b: Top. the SM; Middle. FSA simplification; bottom. the VDS results. From left to right: 50%, 25% and 10%.

images from the 12 cameras situated around the approximations and original models. The percentage of introduced error is shown in the table for each one of the methods. It can be observed that the approximations obtained by the VDS method introduce a lower error than obtained by FSA and by SM. In fact, the approximations obtained by the SM method introduce the highest error because of its stochastic performance.

Finally, the computing time is also shown for every simplification. As it can be seen, the cost of the presented method is quite higher than the time employed to obtain the levels of detail with the FSA and with the SM. View-dependent methods have in general a high temporal cost, but as was previously said, This is not really a problem because the calculations are performed in a pre-process [79]. This high cost is compensated by the good quality of the obtained results, as can be seen in the figures.

5.4.2 Evaluating the Simplified Approximations

Apart from the plant models that have been used to compare the VDS method, others have been simplified and the quality of the obtained results also evaluated using the same error measure, the VMI. Visual results are shown in Figures 5.11 and 5.12. Three levels of detail have been obtained for every one of these plant models, the ones that approximately maintain 25%, 10% and 1% of the leaves. However, only the two last LoDs are included in the chapter to evaluate the good visual quality. All the data are detailed in Table 5.2. The geometric details of every one of the plants that have been used in the test are described in it, adding the number of leaves of the resulting approximations. The leaves of the *Sorbus aucuparia* tree are formed by a mesh with several polygons and not only two triangles forming a quad, like the other ones models (Figure 5.3(b)). Results obtained by simplifying this foliage model demonstrate the good performance of the method with models that use this kind of representation for the leaves.

Figure 5.11. Visual results of the VDS scheme. From top to bottom: *Betula lenta*: original model 16,100 leaves; retaining 10% of leaves: 1,610; and 1%: 161 leaves. *Alnus glutinosa*: original model 41,155 leaves; retaining 10% of leaves: 4,115; and 1%: 411 leaves. *Betula popudifolia*: original model 20,280 leaves; retaining 10% of leaves: 2,028; and 1%: 202 leaves.

Finally, the percentage of VMI deviation against the original model is shown for every one of the computed levels of detail. Furthermore, the information of the computing time is also shown in this table for every simplification.

Figure 5.12. Visual results of the VDS scheme. From top to bottom: *Sorbus aucuparia lenta*: original model 2,760 leaves; retaining 10% of leaves: 276; and 1%: 27 leaves. *Acer platanoides*: : original model 105,688 leaves; retaining 10% of leaves: 10,569; and 1%: 1,056 leaves.

Figure 5.13. Example of the composition of a scene where different approximations of the same plant are shown (original foliage, 50%, 25% and 10% approximations).

Table 5.2. Experimental data evaluating the results of the presented method.

Model	Original		Final		VMI Error	Time (s)
	Triangles	Leaves	Triangles	Leaves		
<i>Betula lenta</i>	32,200	16,100	8050	4025	4.27	2412.80
			3220	1610	9.86	2634.06
			322	161	28.42	2782.18
<i>Alnus glutinosa</i>	82,310	41,155	20,576	10,288	10.53	9950.32
			8230	4115	14.54	10,301.56
			822	411	35.47	10,745.12
<i>Betula populifolia</i>	40,560	20,280	10,140	5070	12.29	3137.33
			4056	2028	19.84	3350.38
			404	202	39.14	3432.01
<i>Sorbus aucuparia</i>	49,680	2760	12,420	690	14.12	642.52
			4968	276	29.88	849.12
			486	27	51.40	912.41
<i>Acer platanoides</i>	211,376	105,688	52,844	26,422	9.75	14,112.47
			21,138	10,569	13.86	14,405.18
			2112	1056	37.15	14,484.03

Finally, a proposal of the composition of the different levels of detail shown in Figure 5.9 is illustrated in Figure 5.13. The distance to the camera has conditioned the choice of every approximation. As can be seen in the figure, the reduction of leaves in the representations cannot be appreciated even at the furthest level of detail.

5.4.3 Simplification Considering Visual Obstacles

The last experiment that has been carried out takes some obstacles into account when the simplification is performed. In order to evaluate the good results in these cases, two different situations have been simulated. Firstly, the simplification method has taken as input the foliage of a tree that is partially hidden by a small fence that surrounds it (Figure 5.14). In this case, a set of leaves is completely hidden from some of the viewpoints. This makes the error introduced by pruning these leaves be negligible. If the remaining leaves have to be, for instance, 10% of

the original representation, these non-visible leaves will be the first to be pruned, meaning that the leaves that remain are distributed in the discernible area of the canopy. Figure 5.14b shows the distribution of the leaves in a situation where these visual obstacles have been considered. For a better appreciation, the fence in front is rendered transparent.

Figure 5.14. Simplified approximation of foliage when surrounded by a fence. **(a)** Level of detail formed by 10% of the leaves; **(b)** detail of the approximation making transparent the fence in front.

Other kinds of visual obstacles that have also been tested are the walls situated near the plants. Some parts of the canopy are hidden by them, so leaves located in this area are the first ones pruned when the simplification is performed. Figure 5.15 illustrates this situation. As well as the previous example, only 10% of the leaves have been left at this level of detail. Figure 5.15b shows the back view of the approximation. The walls have been cut in order to appreciate that the leaves in the back area of the foliage have been pruned.

5.5 Conclusions

This chapter has presented a simplification method based on viewpoint mutual information that deals with the sparse part of the plants. The geometry that forms the canopy of the trees and plants is reduced by performing a viewpoint-driven simplification. In order to

Figure 5.15. Simplified approximation of a plant when it is situated close to walls. (a) Ninety percent simplified; only 10% of leaves remain; (b) back view. Walls have been cut for a better appreciation.

decrease the number of rendered leaves to a determined number, the visibility of every leaf in the canopy is evaluated according to viewpoint mutual information. To perform this evaluation, a set of cameras has been distributed around the foliage to evaluate the visual influence of each leaf pruning. Images obtained from these cameras are compared with the images of the original foliage, taken from the same viewpoints. This makes it possible to assign an error to every potential leaf pruning. The leaf that introduces the lowest error is pruned in every iteration.

The leafy appearance of the plant is preserved by resizing the nearest leaf of the removed one. The new size of this nearest leaf depends on the error introduced by the one that disappears. This resizing is addressed to fill the hole that has left the removed leaf, avoiding the pruning appearance. These processes are performed until the desired number of leaves is achieved. Different experiments have been carried out that demonstrate the good performance of the results, obtaining LoDs composed by a very low number of leaves that still maintain similarity with the original representation.

Combining the presented method with a general simplification scheme applied to the meshes that form the trunks and branches makes it

possible to obtain different LoDs of plants and trees that can form a discrete multiresolution model. This set of approximations can be used in real time by video game designers in interactive applications. Real-time rendering can be achieved by changing the rendered representation from one LoD to another according to some criteria, such as the size of the foliage in the final rendering.

Chapter 6

Multiresolution Foliage Rendering

This chapter presents a continuous level of detail representation of foliage of trees. Multiresolution modeling allows to adapt the number of polygons to render to the relevance of the object in the scene. However, foliage is represented by isolated polygons, so most of the multiresolution modeling methods do not work properly with this part of the tree. This chapter presents a multiresolution model that allows to adapt the number of leaves to the relevance of the foliage in the scene. The criterion to select the appropriate leaves to render is based on a previously performed view-driven simplification. To adapt this parameter in real time, data structures and the necessary algorithms that allow us to extract the appropriate number of polygons are presented. Some tests have been developed to evaluate the proposed solution and results show the good performance of the presented continuous level of detail.

6.1 Introduction

Vegetation is an essential part of the outdoor environments in applications where realism is a must. However, the vast amount of

polygons that are required to represent them makes their inclusion in interactive applications a problem. Level of detail modelling has been confirmed as a solution to adapt the number of polygons of the objects in real-time rendering of scenes. This technique manages object representation while maintaining an adequate frame rate in real-time rendering applications. Nevertheless, foliage of the plants are represented by isolated polygons, so most of the multiresolution models in the literature can not properly manage their representation. This problem does not occur when the solid part of the plants, the trunk and the branches, are considered because they are represented by continuous meshes.

According to Ribelles et al. [142], multiresolution models can be classified in discrete and continuous ones. The first group provides the application with a set of individual representations with different levels of detail. This technique has been widely used and it is included in some game engines, such as Unity [120] or Unreal Engine [148]. The continuous multiresolution models allow the application to vary the number of polygons that represent the object in a smooth way [149]. These multiresolution models avoid popping effects because the change from one resolution to another one is made in a continuous way. Both groups of multiresolution models are based on simplification methods that have previously processed the geometry of the object. This simplification process is performed off-line and the results are used to build the data structure that form the appropriate multiresolution models. Simplification methods differ in the metrics they are based on. Because of the fact that foliage is not represented by continuous meshes, the simplification methods have to be adapted to this kind of representation. Some works have been presented that deal with this part of the plants, that differ in the metrics they are based on: based on geometry [1, 99] or based on view-point driven [150].

This chapter proposes a continuous multiresolution model that allows

representing the foliage and adapting the number of polygons according to their importance in the scene. The simplification method used to build this structure of level of detail has been the method presented in the previous chapter.

The proposed multiresolution model, as well as its level of detail extraction algorithm, has been developed to be easily included in current game engines in order to facilitate its use. A plug-in to be included in Unity 3D has been implemented, so the test and experiments to evaluate the method have been performed using this game engine. The obtained results show the good performance of the proposed level of detail model.

This chapter is organized as follows: First, in Section 6.2 briefly describes the model overview. Next, in Section 6.3 it is presented the data structure used to extract the information of the simplified model. Moreover, Section 6.4 analyzes the rendering algorithms that perform the extraction of the appropriate number of leaves that form the foliage representation. Then, in Section 6.5 describes the test that has been performed and analyzes the results obtained, and finally, section 6.6 presents some conclusions and observations related to the future work.

6.2 Model overview

The presented continuous multiresolution model is based on the Viewpoint-Driven Simplification (VDS) of Plant and Tree Foliage obtained in the previous chapter. The model stores the data obtained from the VDS method in a data structure. As the viewer moves across the scene, vegetation elements update their detail based on the distance to the viewer and the size of the area they occupy on the final image. To do this, some algorithms have been developed that access to that data structure and adapt in real time the number of polygons rendered to the appropriate view conditions. Algorithms add or remove leaves from the current foliage representation and scale the rest that remain in the

foliage, reducing the rendering time required to ensure the realism of the scene.

6.3 Data Structure

Due the VDS simplification method is an iterative process, the sequence of deleted leaves can be easily obtained, as well as the leaf modified in each iteration and its scale. From this sequence, a data structure has been designed that stores all the data necessary to draw each of the required levels of detail. It should be noted that the order in which the data is stored in said structure is the order of the drawing of the leaves. This structure is shown below.

```
Struct Leaf {  
    Vertex vertices[4];  
    int leafToSplit;  
    int leafToCollapse;  
    float scale;  
}  
  
vector<leaf> leaves;
```

The *vector leaves* stores the list of all the leaves that form the foliage. The order of storing the leaves is conditioned by the sequence obtained from the VDS algorithm, so it represents the drawing sequence.

All the data required to render a leaf is stored in the *struct Leaf*. It stores the data required both to render the original size and to render when some resizing is required. It is composed by:

- *vertices*: It stores the vertices that form the leaf.

- *leafToModify*: The leaf that has to be modified by scaling its size when this leaf is added (*leafToSplit*) or removed (*leafToCollapse*) from the current representation.
- *modScale*: The amount that must vary the size of the leaf to be modified.
- *myScale*: This variable allows you to save the current scale of the sheet itself. Initially its value is 1.

6.4 Rendering algorithm

Once all the required data are stored in the multiresolution model, the algorithms that extract the appropriate level of detail have to be designed. They take into consideration the distance of the plant to the point of view as well as the size of that plant in the final rendered frame. The target of the criterion is to improve the quality of the render, so the closer the viewer is, the higher the realism has to be. As the camera moves far from the plant, fewer polygons are required to represent the foliage. The same case happens with the number of pixels used to render the final representation of the foliage: the importance of the plant in the frame conditions the number of leaves that is required to ensure the good quality of the final rendering.

These both criteria have been considered and codified in the formula shown in Equation 6.1, that obtained the value *LOD*. It combines the value *distanceToViewer*, obtained from the normalized distance (with a value between 0 and 1) between the center of the foliage and the camera position, and *projectedArea*, that stores the size of the projected area of the foliage in the current frame. Some constants (*a*, *b*) have been added to the equation to allow customizing the relevance of every one of the criteria in the equation.

$$LOD = distanceToViewer * a + projectedArea * b \quad (6.1)$$

This *LOD* value is normalized and allows us to obtain the number of leaves to be finally rendered. Then, this number is compared with the one that is currently being rendered, performing different actions in the case the number of leaves to render is the same, fewer or higher than the current number of leaves rendered. The actions to be performed are conditioned by this comparison, so:

- *Same number of leaves*: The level of detail to render is the same that is being currently rendered, so no actions are required to be performed.
- *Fewer leaves to render*: Some leaves have to be removed from the list to be rendered and some of the ones that remain have to be resized.
- *Higher number of leaves*: Some leaves have to be added to the final representation and some of the ones that are being rendered have to reduce their size in order to undo the scaling.

Algorithm 3 analyzes the process to follow in these cases. Every time the process starts, the algorithm has to evaluate the relevance of the foliage in the scene (*LOD*). This value is used to extract the number of leaves to render in this case *newLeaf*. Once this number has been obtained, it is determined if some leaves have to be added to the current approximation or some leaves have to be removed (*collapse*).

The scaling process is already calculated by the simplification method and saved in the structure used by this model to speed up the process and not have to recalculate the scaling on each leaf. Each leaf, when removed or added, only modifies another leaf (*leafToModify*), therefore, this model uses the leaves themselves to save the scale values that

Algorithm 3 Process to extract the Level of Detail.

```

Function Extract(LOD) do
    newLeaf = totalLeaves * LOD;
    if newLeaf > currentLeaf then
        for i = 0; i < newLeaf - currentLeaf; i ++ do
            leafSplit(currentLeaf);
            currentLeaf ++;
        end for
    else
        for i = 0; i < currentLeaf - newLeaf; i ++ do
            leafCollapse(currentLeaf);
            currentLeaf --;
        end for
    end if
end function

Function leafSplit(currentLeaf)
    leafToSplit = leaves[currentLeaf].leafToSplit;
    aux = leaves[currentLeaf].modScale;
    leaves[currentLeaf].modScale = leaves[leafToSplit].myScale;
    leaves[leafToSplit].myScale = aux;
end function

Function leafCollapse(currentLeaf) do
    leafToCollapse = leaves[currentLeaf].leafToCollapse;
    aux = leaves[currentLeaf].modScale;
    leaves[currentLeaf].modScale = leaves[leafToCollapse].myScale;
    leaves[leafToCollapse].myScale = aux;
end function

```

Algorithm 4 Process to render the Level of Detail.

```

Function Draw(LOD) do
    lastLeave = totalLeaves * LOD;
    for i = 0; i < lastLeave; i ++ do
        drawLeave(i);
    end for
end function

```

affect the other leaf, exchanging them when necessary. First, the model differentiates whether it is a *leafCollapse* or a *leafSplit*. In the first case, the scale value of the leaf to be modified is contained in the leaf to be collapsed. Therefore, the current leaf exchanges the scale of its *modScale* variable with the *myScale* variable of the leaf to be modified, as shown in Figure 6.1(a).

Figure 6.1. (a) Collapse operation, removing Leaf L_i . (b) Split operation, adding Leaf L_i .

In the second case, as seen in the figure 6.1(b), the added leaf retrieves its original scale value from the nearby leaf. Which, in addition, also retrieves the scale from your *modScale* variable. When this process has been performed, the algorithm renders the list of the leaves that form the current level of detail (*Draw(LOD)*).

6.5 Results

The experiments have been performed using a computer with an Intel Core i7-3517U 2.40GHz processor with 8 GB RAM and AMD Radeon HD 8500M/8700M graphics card. All the geometric models of plants that have been used in the tests have been modeled using the commercial modeling tool Xfrog [35]. Unity 3D version 2018.2.7 was used to implement the model multiresolution and to view the result.

Three different vegetation species have been considered in the experiments: *English oak* with 20,146 leaves, *Carya illinoensis* with 8,059 leaves and *Betula populifolia* with 12,140 leaves. These three trees can be seen in Figure 6.2. The trunks and branches used have been simplified by the method presented by [91], but they do not change during the execution of the simplification process presented in this chapter.

Figure 6.2. The three vegetation species used in the tests.

In order to validate the multiresolution model, tests have been carried out with the three plant species separately and with forests formed by these species. These tests have been inspired by the tests proposed by Ribellés et al. [151].

6.5.1 Tree Rendering

The performed test has analyzed each vegetable species individually. Every one of them has been represented and then the camera has been moved away from the model, adapting the level of detail of its representation according to the formula previously presented. Then, the temporal cost of performing the visualization of the approximation has been calculated. This has been measured by obtaining the frames per second (FPS) while the camera increases and decreases the distance to the model. The results obtained can be seen in Figure 6.3. As it can be observed in the chart, initially there is a great difference between the three vegetable species due to the different number of leaves. However, as the camera zooms out, and therefore the level of simplification increases, this difference decreases, further increasing the frames per second.

Figure 6.3. Comparison of the FPS obtained with the three vegetable species at different distances.

Regarding the time used to extract the appropriate level of detail, it has been measured in milliseconds per every distance. The obtained

Figure 6.4. Analysis of the time required to extract the different levels of detail for the three vegetable species at different distances.

results can be seen in Figure 6.4. At the initial point, since the camera is close to the model and therefore there is no simplification, the cost is reduced to the render time itself. Then, as the method starts to reduce the detail of the foliage, the difference between the extraction of required approximation of the three vegetable species is greater. However, as the camera zooms out and the representations reduce considerably their number of leaves, the extraction cost decreases and the three species reduce the difference between them.

6.5.2 Forest Rendering

In order to better analyze the presented model, some tests have been carried out with groups of trees to represent forests. Figure 6.5 shows the results obtained when analyzing the spatial cost of visualizing them while the number of trees is increased. This cost has been measured by obtaining the frames per second (FPS) while the camera moves around

Figure 6.5. Chart showing the FPS obtained while the amount of trees rendered increases.

a circular path that points to the forest center. Each forest is made up of only one vegetable species that adapt their resolution according to the distance to the camera. However, in order to evaluate the good quality of the results, the forest represented with the vegetable species *English oak* has been tested twice: one test with all the geometry, without performing reduction *English oak geometry*, and the other one adapting the detail to the *English oak* criteria.

As it can be seen in the chart, the FPS considerably increases as the detail of the foliage of the rendered plants are adapted to the established criteria. Moreover, it can also be seen that the cost of extracting the appropriate leaves for every frame is not relevant in front of the wide reduction in the visualization time.

6.5.3 Visual results

In order to show how the presented multiresolution model allows simplifying the vegetation elements without reducing their visual quality, Figure 6.6 shows the simplification carried out at three different distances. For this, a tree with 20,000 leaves has been used, shown in 80% of its leaves, 30% and 15%.

Figure 6.6. Comparison between three different distances. With a reduction of 20%, 70% and 85% respectively.

In addition, a view of a small forest is shown in Figure 6.7. This forest contains nine trees with 20,000 leaves each one. After applying the simplification model, the number of total leaves drawn is 49,680. However, the visual difference between the close and distant trees is barely appreciated.

Figure 6.7. Forest with nine trees with simplification applied.

6.6 Conclusions

In this chapter, a multiresolution model for the foliage of plant elements has been presented, based on the Viewpoint-Driven Simplification method. This simplification method establishes a leaf removing order from the most hidden leaves to the outside of the foliage. Also, the method avoids the pruning appearance of the less detailed representation by filling the possible gaps in the foliage by scaling the leaves close to the pruned ones. In this way, the foliage can be highly simplified without being barely noticeable by the viewer when the camera is far from the plant. Data structure and the algorithms that allow to extract the appropriate approximation have been presented too.

Also, the multiresolution model have been designed to be easily implemented in game engines. The extraction time of the appropriate level of detail is very low being, as results demonstrate, less than 1 millisecond in models with more than 40,000 polygons. Also, the results obtained demonstrate that the model allows to adapt the detail of the foliage in a continuous way in a scene. Regarding forest visualization,

also results support that the rendering cost is significantly reduced while maintaining the visual quality.

Chapter 7

Conclusions and future work

The objective of this work has been to make different contributions in the field of procedural generation of natural environments, starting from initial restrictions specified by the user. Specifically, work has been done on: the generation of terrain, the placement of natural elements and the optimization of the real-time display of trees and plants.

The chapter is organized as follows. First, the conclusions obtained in the different sections of the thesis are presented. Here there are some ideas for further research in the future. Finally, it is provided a list of publications that have emerged from this work and the research projects that have allowed the development of the thesis itself.

7.1 Conclusions

Before beginning each of the topics on which this thesis has been worked on, a study of the state of art was carried out on all the works related in the bibliography. This study, which is presented in Chapter 2, includes an analysis on: the techniques of representation and generation

of land, the creation of natural environments, the techniques of geometry optimization and the interactive visualization of trees and plants.

In Chapter 3 a new procedural modeling method for generating terrain was presented which combines random height data with restrictive data regarding routes, points of interest, and even areas. The resulting terrain is forced through all these constraints with a high degree of precision, increasing in randomness as it moves away from them. For this, a system of equations based on a noise function is solved, where each equation belongs to one of the restrictive points. The result of this system allows obtaining the value of all the points that influence the noise function to obtain the restrictive points, allowing those that do not influence, to take random values. The noise function used is the one presented by Perlin, which originally, despite allowing some variations, does not allow the inclusion of precise data.

The result obtained by the method shows that the resulting terrains include in a highly precise way and with a low computational cost, all the restrictions given, in addition to the randomness sought in a natural terrain. As can be seen in Figure 7.1. All terrain points maintain continuity regardless of whether they are restrictive points or random points, fulfilling the objective of obtaining a realistic natural terrain. The proposed model allows obtaining different results in each execution, maintaining in any case the same input restrictions. As an additional point, the method can create large terrains by carrying out an extension of the proposed technique.

In Chapter 4 a method has been presented that distributes natural features over a terrain in a procedural way, maximizing coverage and following various biological and terrain constraints. In the first place, a distribution of the seeds is carried out without considering the species, to afterwards label them with the corresponding species. To do this, the method chooses the label based on the height and slope where it will be placed and the biological characteristics of each species. The

Figure 7.1. Terrains obtained with the method of procedural generation of lands with restrictions.

final representation is obtained by simulating the growth process of the seeds. For this, the method allows them to grow to their maximum size, according to their biological information and the space occupied by the other seeds. The results show that the resulting distribution meets the given requirements of each species and the groups of plants have sufficient heterogeneity and realism. As it can be seen in Figure 7.2, in these groups, a dominant species is always maintained without excluding the other species. In addition to this, limits are blurred between the other groups, avoiding sudden changes of species in the same forest.

Regarding the optimization of the visualization of natural elements, in chapter 5 a simplification technique based on mutual information

Figure 7.2. Result obtained with the method of procedural distribution of plant elements.

from points of view centered on foliage has been presented. This technique seeks to reduce the geometry of the model by eliminating complete leaves and prioritizing those that produce less visual impact when eliminated. To do this, a set of cameras is distributed around the model to evaluate the obtained variation through mutual information from the point of view, when removing each leaf. This allows each iteration to assign an error to each leaf and eliminates the one with the lowest value. This process allows the pruning order to start with those leaves that remain hidden from any view. Each time a leaf is removed, the technique looks for the closest one and escalates it in proportion to the error made via pruning. In this way, the leafiness of the foliage is maintained and any great visual impact between the simplified model and the original is avoided.

This process is carried out iteratively until the desired simplification level is reached. So, on the one hand, a set of models with different levels of detail can be obtained to be used in a discrete multi-resolution model. And on the other hand, a structure that includes the pruning order can be obtained to be used in a continuous multiresolution model.

In Figure 7.3 a foliage model is represented with various simplifications in both up close and at different distances. The upper part shows closely the simplification at 50%, 75% and 90%, and the lower part shows, at different distances, the model at 0% simplification, 50%, 75%, 90% and up to 95%. In this image, how the results reach a simplification with a very low visual impact can be seen, which when placed at a certain distance from the observer, the difference with the original model is barely perceptible.

Figure 7.3. Result obtained with the simplification technique based on mutual information. Top from left to right: 50%, 75%, 90%. Bottom from left to right: 0%, 50%, 75%, 90%, 95%.

To complete this simplification technique, at Chapter 6, a continuous multiresolution model based on the prior art has been presented. This

model extracts from the data structure, obtained in the simplification, the pruning order of the model, the leaf to be scaled in each iteration and the scaling value to be applied. Both, the data structure and the algorithms used, have been presented in the relevant chapter. In addition, they have been designed to be easily integrated into game engines.

In the multiresolution model, each time the observer changes position, and therefore the importance of the element in the scene changes, it obtains the degree of simplification to apply. In case this degree is reduced, it adds the corresponding leaves to be drawn following the pruning order and reduces the size of the nearby leaves using the scaling value. Otherwise, if the degree of simplification increases, it removes the corresponding leaves and increases the size of the nearby leaves to maintain the same level of leafiness. Once the above is obtained, the tree is redrawn, so that the increase or reduction of leaves is carried out in a smooth and continuous way.

The results obtained demonstrate how this continuous multi-resolution model allows us to maintain visual quality while the cost of rendering is significantly reduced, compared to rendering without simplification.

7.2 Future work

From this work several future lines of research emerge. Regarding the placement of plant elements, this methodology could be expanded to cover large areas of vegetation. The use of Wang-tiles with KD-trees [59] has been considered to solve this problem. These multilevel data structures divide the space into instantiated representations where each tile is filled with PDD. It could easily be extended to use Wang-tiles to generate plant distribution patterns.

Another line of research to work on is to improve the placement of vegetation by adding other biological characteristics of the species, such

as soil moisture, temperature, and the position of the sun. In line with this line of research, the option of being able to indicate artificial areas as cultivation fields is being analyzed. By not having a natural behavior, these areas will ignore other characteristics such as irregular distribution or the selection of species according to the characteristics of the terrain, so further study is required. In addition to this, the adaptation of the vegetal representation according to the size obtained is being analyzed. In this work only a single representation has been used, but greater biological precision would be obtained if different plant models were used to represent the different levels of growth for each species.

Regarding the optimization of the visualization of natural elements, on the one hand, to improve the simplification made considering the information on the texturing of the leaves [152] is wanted. And on the other hand, to consider the skeletons to carry out the simulation of the wind in the plants is wanted. By adding some restrictions to the presented method, such as associating each leaf with a skeleton or branch, a model can be achieved that can be animated. The presented method does not add new leaves to the representation, so although some leaves are removed, the rest of the remaining leaves can perfectly maintain the link with the initial skeleton and simulate the movement of the wind.

Furthermore, an aspect that needs to be improved in the multiresolution model presented is its integration in the GPU to accelerate the cost of extraction of the models. The featured model demonstrates a significant cost reduction, but when running on CPU, the full potential of current technology cannot be harnessed.

Lastly, as future work it is expected to unify all the models developed in this thesis in a plug-in that can be used in game engines to generate complete natural environments. In addition to adding functionalities for other characteristics corresponding to natural environments, such as rock integration, atmospheric effects and even animal life. The final idea

of this last line of research is to achieve a photorealistic result and to be able to compare the realism obtained using the data of a real natural environment with photographs taken of it.

7.3 Publications

For assessing the different works presented throughout this PhD dissertation, this section lists the different publications obtained while developing this thesis.

7.3.1 Journals

- Cristina Gasch Garcia, Miguel Chover Sellés, Inmaculada Remolar Quintana, Cristina Rebollo Santamaría. Viewpoint-Driven Simplification of Plant and Tree Foliage. *Entropy*. Num. 20 (4). pp. 213. 2018.
- Cristina Gasch Garcia, Miguel Chover Sellés, Inmaculada Remolar Quintana, Cristina Rebollo Santamaría. Procedural modelling of terrains with constraints. *Multimedia Tools and Applications*. Num. 79. pp. 31125–31146. 2020.
- Cristina Gasch Garcia, José Martínez Sotoca, Miguel Chover Sellés, Inmaculada Remolar Quintana, Cristina Rebollo Santamaría. Procedural Modelling of Plant Ecosystems. *Computers and Graphics*. 2020. Submitted.
- Cristina Gasch Garcia, Miguel Chover Sellés, Inmaculada Remolar Quintana, Cristina Rebollo Santamaría. Multiresolution Modeling for Foliage Rendering. *Entertainment Computing*. 2020. Submitted.

7.3.2 Conferences

- Cristina Gasch Garcia, Miguel Chover Sellés, Inmaculada Remolar Quintana. Procedural Modeling of Terrain from GPS Routes. XXVI Spanish Computer Graphics Conference (CEIG), pages 101-108. The Eurographics Association, 2016.
- Cristina Gasch Garcia, Miguel Chover Sellés, Inmaculada Remolar Quintana. Procedural generation of natural environments with restrictions. In F.J. Melero and N. Pelechano, editors, Spanish Computer Graphics Conference (CEIG), pages 131–139. The Eurographics Association, 2017.
- Cristina Gasch Garcia, Miguel Chover Sellés, Inmaculada Remolar Quintana. Multiresolution Foliage Rendering. Proceedings of the VI Congreso de la sociedad española para las ciencias del videojuego (COSECIVI), pages 78-90. 2020.

Bibliography

- [1] I. Remolar, M. Chover, O. Belmonte, J. Ribelles, and C. Rebollo. Geometric simplification of foliage. *In Eurographics 2002 Short Presentations*, 2002.
- [2] Gamasutra. Art tips for building forests. Available online: https://www.gamasutra.com/blogs/DannyWeinbaum/20170216/291658/Art_Tips_for_Building_Forests.php, 2021. Last accessed January 2021.
- [3] SpeedTree. Games reel 2020. Available online: <https://store.speedtree.com/games-reel-2020>, 2021. Last accessed January 2021.
- [4] Wikipedia. Speedtree: Aplicaciones. Available online: <https://en.wikipedia.org/wiki/SpeedTree>, 2021. Last accessed January 2021.
- [5] Speedtree. Speedtree: 3d vegetation modeling and middleware. Available online: <https://store.speedtree.com>, 2018. Last accessed January 2021.
- [6] Benoit B Mandelbrot. *The fractal geometry of nature*, volume 1. WH freeman New York, 1982.
- [7] Gavin SP Miller. The definition and rendering of terrain maps. *In SIGGRAPH '86: Proceedings of the 13th Annual Conference*

- on Computer Graphics and Interactive Techniques*, 20(4):39–48, 1986.
- [8] F Kenton Musgrave, Craig E Kolb, and Robert S Mace. The synthesis and rendering of eroded fractal terrains. *In SIGGRAPH '89: Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, 19:41–50, 1989.
- [9] Alain Fournier, Don Fussell, and Loren Carpenter. Computer rendering of stochastic models. *Communications of the ACM*, 25(6):371–384, 1982.
- [10] K Raiyan Kamal and Yusuf Sarwar Uddin. Parametrically controlled terrain generation. *In GRAPHITE '07: Proceedings of the 5 International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*, pages 17–23, 2007.
- [11] Fares Belhadj. Terrain modeling: a constrained fractal model. *In AFRIGRAPH '07: Proceedings of the 5 Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, pages 197–204, 2007.
- [12] 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, 2010.
- [13] Ruben M Smelik, Tim Tutenel, Rafael Bidarra, and Bedrich Benes. A survey on procedural modelling for virtual worlds. 33(6):31–50, 2014.
- [14] Jean-David Genevaux, Eric Galin, Eric Guerin, Adrien Peytavie, and Bedrich Benes. Terrain generation using procedural models based on hydrology. *ACM Transactions on Graphics. Proceedings of SIGGRAPH*, 32(4):13, 2013.

- [15] Jean-David Génevaux, Eric Galin, Adrien Peytavie, Eric Guérin, Cyril Briquet, François Grosbellet, and Bedrich Benes. Terrain modelling from feature primitives. *Computer Graphics Forum*, 34(6):198–210, 2015.
- [16] Eric Guérin, Julie Digne, Eric Galin, and Adrien Peytavie. Sparse representation of terrains for procedural modeling. *Computer Graphics Forum*, 35(2):177–187, 2016.
- [17] Cristina Gasch, Miguel Chover, Inmaculada Remolar, and Cristina Rebollo. Procedural modeling of terrain from gps routes. In *Proceedings of the XXVI Spanish Computer Graphics Conference, CEIG '16*, pages 101–108, Goslar Germany, Germany, 2016. Eurographics Association.
- [18] Alex D Kelley, Michael C Malin, and Gregory M Nielson. Terrain simulation using a model of stream erosion. *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, pages 263–268, 1988.
- [19] Kenji Nagashima. Computer generation of eroded valley and mountain terrains. *The Visual Computer*, 9(13):456–464, 1997.
- [20] Benjamin Neidhold, Markus Wacker, and Oliver Deussen. Interactive physically based fluid and erosion simulation. *Proceedings of Eurographics Workshop on Natural Phenomena*, 1:25–32, 2005.
- [21] Xing Mei, Philippe Decaudin, and Bao-Gang Hu. Fast hydraulic erosion simulation and visualization on gpu. *IEEE Computer Society*, pages 47–56, 2007.
- [22] Ondrej Št'ava, Bedrich Beneš, Matthew Brisbin, and Jaroslav Krivánek. Interactive terrain modeling using hydraulic erosion. In *Eurographics/SIGGRAPH Symposium on Computer Animation*, pages 201–210, 2008.

- [23] Juraj Vanek, Bedrich Benes, Adam Herout, and Ondrej Stava. Large-scale physics-based terrain editing using adaptive tiles on the gpu. *IEEE Computer Graphics and Applications*, 31(6):35–44, 2011.
- [24] Peter Krištof, Bedrich Beneš, Jaroslav Křivánek, and Ondrej Štáva. Hydraulic erosion using smoothed particle hydrodynamics. *Computer Graphics Forum*, 28(2), 2009.
- [25] Guillaume Cordonnier, Eric Galin, James Gain, Bedrich Benes, Eric Guérin, Adrien Peytavie, and Marie-Paule Cani. Authoring landscapes by combining ecosystem and terrain erosion simulation. *ACM Trans. Graph 2017.*, 36(4):12 pages, 2017.
- [26] Szymon Stachniak and Wolfgang Stuerzlinger. An algorithm for automated fractal terrain deformation. *Computer Graphics and Artificial Intelligence*, 1:64–76, 2005.
- [27] Howard Zhou, Jie Sun, Greg Turk, and James M Rehg. Terrain synthesis from digital elevation models. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):834–848, 2007.
- [28] Jens Schneider, Tobias Boldte, and Rüdiger Westermann. Real-time editing, synthesis, and rendering of infinite landscapes on gpus. *In Vision, Modeling and Visualization*, pages 145–152, 2006.
- [29] James Gain, Patrick Marais, and Wolfgang Straßer. Terrain sketching. *In I3D '09: Proceedings of the Symposium on Interactive 3D Graphics and Games*, pages 31–38, 2009.
- [30] Ruben Michaël Smelik, Tim Tutenel, Klaas Jan de Kraker, and Rafael Bidarra. Interactive creation of virtual worlds using procedural sketching. pages 29–32, 2010.

- [31] Brennan Rusnell, David Mould, and Mark Eramian. Feature-rich distance-based terrain synthesis. *The Visual Computer*, 25(5):573–579, 2009.
- [32] Houssam Hnaidi, Eric Guérin, Samir Akkouche, Adrien Peytavie, and Eric Galin. Feature based terrain generation using diffusion equation. In *Computer Graphics Forum: Proceedings of Pacific Graphics.*, 29:2179–2186, 2010.
- [33] Gwyneth A Bradbury, Il Choi, Cristina Amati, Kenny Mitchell, and Tim Weyrich. Frequency-based creation and editing of virtual terrain. *Proceedings of the 11th European Conference on Visual Media Production, London, UK.*, page 13–14, 2014.
- [34] Anna Puig-Centelles, Peter AC Varley, Oscar Ripollés, and Miguel Chover. Automatic terrain generation with a sketching tool. *Multimedia Tools and Applications*, pages 1957–1986, 2014.
- [35] Adrien Bernhardt, André Maximo, Luiz Velho, Houssam Hnaidi, and Marie-Paule Cani. Real-time terrain modeling using cpu-gpu coupled computation. In *SIBGRAPI '11: Proceedings of the 24th Conference on Graphics, Patterns and Images*, pages 64–71, 2011.
- [36] Giliam JP De Carpentier and Rafael Bidarra. Interactive gpu based procedural heightfield brushes. In *FDG '09: Proceedings th International Conference on the Foundations of Digital of the 4 Games*, 2009.
- [37] Flora Ponjou Tasse, James Gain, and Patrick Marais. Enhanced texture-based terrain synthesis on graphics hardware. *Computer Graphics Forum*, 31(6):1959–1972, 2012.
- [38] James Gain, Bruce Merry, and Patrick Marais. Parallel, realistic and controllable terrain synthesis. 34(2):105–116, 2015.

- [39] Jian Zhang, Chang-bo Wang, Hong Qin, Yi Chen, and Yan Gao. Procedural modeling of rivers from single image toward natural scene production. *The Visual Computer*, 35(2):223–237, 2019.
- [40] F Bravo, Gregorio Montero González, and M Del Río. Índices de densidad de las masas forestales. *Ecología*, (11):177–187, 1997.
- [41] C.D. Zegers. *Ecología forestal: el bosque y su medio ambiente*. Colección Nueva Técnica: Universidad Austral de Chile. Editorial Universitaria, 1997.
- [42] Miłosz Makowski, Torsten Hädrich, Jan Scheffczyk, Dominik L. Michels, Sören Pirk, and Wojtek Pałubicki. Synthetic silviculture: Multi-scale modeling of plant ecosystems. *ACM Trans. Graph.*, 38(4), jul 2019.
- [43] Benny Onrust, Rafael Bidarra, Robert Rooseboom, and Johan Van De Koppel. Ecologically sound procedural generation of natural environments. *International Journal of Computer Games Technology*, pages 1–17, 01 2017.
- [44] Carlos Andújar, Antoni Chica, MA Vico, S Moya, and Pere Brunet. Inexpensive reconstruction and rendering of realistic roadside landscapes. *Computer Graphics Forum*, 33(6):101–117, 2014.
- [45] James Gain, Harry Long, Guillaume Cordonnier, and M-P Cani. Ecobrush: Interactive control of visually consistent large-scale ecosystems. *Computer Graphics Forum*, 36(2):63–73, 2017.
- [46] S Temmerman, TJ Bouma, J Van de Koppel, D Van der Wal, MB De Vries, and PMJ Herman. Vegetation causes channel erosion in a tidal landscape. *Geology*, 35(7):631–634, 2007.
- [47] Max Rietkerk and Johan Van de Koppel. Regular pattern formation in real ecosystems. *Trends in Ecology & Evolution*, 23(3):169 – 175, 2008.

- [48] Jiaqi Li, Xiaoyan Gu, Xinchu Li, Junzhong Tan, and Jiangfeng She. Procedural generation of large-scale forests using a graph-based neutral landscape model. *ISPRS International Journal of Geo-Information*, 7(3), 2018.
- [49] Oliver Deussen, Pat Hanrahan, Bernd Lintermann, Radomír Měch, Matt Pharr, and Przemyslaw Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 275–286, 1998.
- [50] Radomír Měch and Przemyslaw Prusinkiewicz. Visual models of plants interacting with their environment. *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 397–410, 1996.
- [51] L.G. Firbank and A.R. Watkinson. A model of interference within plant monocultures. *Journal of Theoretical Biology*, 116(2):291–311, 1985.
- [52] R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial greyscale. *Proceedings of the Society for Information Display*, 17:75–77, 1975.
- [53] B. Lane and P. Prusinkiewicz. Generating spatial distributions for multilevel models of plant communities. In *Proceedings of the Graphics Interface 2002*, pages 69–80, May 2002.
- [54] S. Bauer, U. Berger, H. Hildenbrandt, and V. Grimm. Cyclic dynamics in simulated plant populations. *Proceedings. Biological sciences*, 269:2443–50, 2002.
- [55] M. Alsweis and O. Deussen. Modeling and visualization of symmetric and asymmetric plant competition. *Proceedings of the First Eurographics Conference on Natural Phenomena*, pages 83–88, 2005.

- [56] M. Alsweis and O. Deussen. Wang-tiles for the simulation and visualization of plant competition. *Advances in Computer Graphics*, pages 1–11, 2006.
- [57] A. Lagae and P. Dutré. A comparison of methods for generating poisson disk distributions. *Comput. Graph. Forum*, 27:114–129, 2008.
- [58] A. Dietrich, G. Marmitt, and P. Slusallek. Terrain guided multi-level instancing of highly complex plant populations. *IEEE Symposium on Interactive Ray Tracing*, pages 169–176, 2006.
- [59] M. Weier, A. Hinkenjann, G. Demme, and P. Slusallek. Generating and rendering large scale tiled plant populations. *Journal of Virtual Reality and Broadcasting*, 10(1):1860–2037, 2013.
- [60] B. Onrust, R. Bidarra, R. Rooseboom, and J. van de Koppel. Procedural generation and interactive web visualization of natural environments. In *Proceedings of the 20th International Conference on 3D Web Technology*, Web3D '15, pages 133–141, New York, NY, USA, 2015. ACM.
- [61] C. Gasch, M. Chover, and I. Remolar. Procedural generation of natural environments with restrictions. In F.J. Melero and N. Pelechano, editors, *Spanish Computer Graphics Conference (CEIG)*, pages 131–139. The Eurographics Association, 2017.
- [62] Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.
- [63] GROGRA. Grogra.de. Available online: <http://www.grogra.de/>, 2019. Last accessed January 2021.
- [64] Bedřich Beneš, Michel Abdul Massih, Philip Jarvis, Daniel G Aliaga, and Carlos A Vanegas. Urban ecosystem design. pages 167–174, 2011.

- [65] M. Stamminger and G. Drettakis. Interactive sampling and rendering for complex and procedural geometry. *In Proceedings of the Eurographics Workshop on Rendering, London, UK.*, pages 25–27, 2001.
- [66] Oliver Deussen, Carsten Colditz, Marc Stamminger, and George Drettakis. Interactive visualization of complex plant ecosystems. 2002.
- [67] Guillaume Gilet, Alexandre Meyer, and Fabrice Neyret. Point-based rendering of trees. pages 67–72, 2005.
- [68] Aleks Jakulin. Interactive vegetation rendering with slicing and blending. 2000.
- [69] Anton L Fuhrmann, Eike Umlauf, and Stephan Mantler. Extreme model simplification for forest rendering. *NPH*, 5:57–67, 2005.
- [70] P. Decaudin and F. Neyret. Volumetric billboards. *Comput. Graph. Forum.*, 28(2079), 2009.
- [71] X.P. Zhang and F. Blaise. Progressive polygon foliage simplification. *In Proceedings of the International Symposium on Plant Growth Modeling, Simulation, Visualization and Their Applications 2003-PMA'03*, pages 13–16, 2003.
- [72] X.P. Zhang, F. Blaise, and M. Jaeger. Multiresolution plant models with complex organs. *In Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and Its Applications, Hong Kong, China.*, pages 14–17, 2006.
- [73] Q. Deng, X. Zhang, G. Yang, and Jaeger M. Multiresolution foliage for forest rendering. *Comput. Anim. Virtual Worlds*, 22(1), 2009.
- [74] G.B. Bao, H.J. Li, X.P. Zhang, W.J. Che, and M. Jaeger. Realistic real-time rendering for large-scale forest scenes. *In*

Proceedings of the IEEE International Symposium on VR Innovation, Singapore., pages 19–20, 2011.

- [75] Robert L Cook and John Halstead. Stochastic pruning. 2006.
- [76] R.L. Cook, J. Halstead, M. Planck, and D. Ryu. Stochastic simplification of aggregate detail. *ACM Trans. Graph.*, 26(79), 2007.
- [77] B. Neubert, S. Pirk, O. Deussen, and C. Dachsbacher. Improved model and view-dependent pruning of large botanical scenes. *Comput. Graph. Forum*, 30(1708), 2011.
- [78] J. Gumbau, M. Chover, I. Remolar, and C. Rebollo. View-dependent pruning for real-time rendering of trees. *Comput. Graph.*, 35(364), 2011.
- [79] P. Lindstrom and G. Turk. Image-driven simplification. *ACM Trans. Graph*, 19(204), 2000.
- [80] E. Zhang and G. Turk. Visibility-guided simplification. *IEEE Comput. Vis.*, 31(267), 2002.
- [81] M. Garland and P.S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, Los Angeles, CA, USA.*, pages 3–8, 1997.
- [82] J. Lee, J. Peng, and Kuo J. View-dependent visibility estimation for tree models. In *Proceedings of the IEEE International Conference on Multimedia and Expo, Beijing, China.*, pages 2–5, 2007.
- [83] J. Lee and J. Kuo. Fast and flexible tree rendering with enhanced visibility estimation. In *Proceedings of the Tenth IEEE International Symposium on Multimedia, Berkeley, CA, US.*, pages 15–17, 2008.

- [84] J. Lee, Kuo J., and C. Kuo. Enhanced 3d tree model simplification and perceptual analysis. *In Proceedings of the IEEE International Conference on Multimedia and Expo, New York, NY, USA., 2009.*
- [85] Ivan Viola, Miquel Feixas, Mateu Sbert, and Meister Eduard Groller. Importance-driven focus of attention. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):933–940, 2006.
- [86] Miquel Feixas, Mateu Sbert, and Francisco González. A unified information-theoretic framework for viewpoint selection and mesh saliency. *ACM Transactions on Applied Perception (TAP)*, 6(1):1–23, 2009.
- [87] Pere-Pau Vázquez, Miquel Feixas, Mateu Sbert, and Wolfgang Heidrich. Viewpoint selection using viewpoint entropy. 1:273–280, 2001.
- [88] Pere-Pau Vázquez, Miquel Feixas, Mateu Sbert, and Wolfgang Heidrich. Automatic view selection using viewpoint entropy and its application to image-based modelling. 22(4):689–700, 2003.
- [89] Mateu Sbert, Miquel Feixas, Jaume Rigau, Ivan Viola, and Miguel Chover. Applications of information theory to computer graphics part i: Introduction. *Eurographics Tutorial Notes*, pages 625–704, 2002.
- [90] Mateu Sbert, Dimitri Plemenos, Miquel Feixas, and Francisco González. Viewpoint quality: Measures and applications. *In Proceedings Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging 2005, Girona, Spain., pages 18–20, 2005.*
- [91] Pascual Castelló, Mateu Sbert, Miguel Chover, and Miquel Feixas. Viewpoint-driven simplification using mutual information. *Computers & Graphics*, 32(4):451–463, 2008.

- [92] S. Zhang. Foliage simplification based on multi-viewpoints for efficient rendering. *J. Softw.*, 9(1655), 2014.
- [93] Q. Zhu, S. Zhang, L. Zeng, and J. Liu. Multi-resolution foliage faithful to the visual and botanic concepts. *J. Comput. Theor. Nanosci.*, 11(1007), 2014.
- [94] A. Jakulin. Interactive vegetation rendering with slicing and blending. In A. de Sousa and J.C. Torres, editors, *Proc. Eurographics 2000 (Short Presentations)*. Eurographics, 2000.
- [95] J. Lluch, E. Camahort, and R. Vivo. An image based multiresolution model for interactive foliage rendering. *Journal of WSCG'04*, 12(3):507–514, 2004.
- [96] Carsten Dachsbacher, Christian Vogelsgang, and Marc Stamminger. Sequential point trees. In *ACM Transactions on Graphics*, pages 657–662, 2003.
- [97] I. Remolar, M. Chover, J. Ribelles, and O. Belmonte. View-dependent multiresolution model for foliage. *Journal of WSCG'03*, 11(2):370–378, 2003.
- [98] C. Rebollo, I. Remolar, M. Chover, and J. Gumbau. Hardware-oriented visualisation of trees. In *Lecture Notes in Computer Science vol. 4263/2006*, pages 374–383, 2006.
- [99] Q. Deng, X. Zhang, G. Yang, and M. Jaeger. Multiresolution foliage for forest rendering. *Comput. Animat. Virtual Worlds*, 21(1):1–23, 2010.
- [100] T. Dong, S. Liu, J. Xia, J. Fan, and L. Zhang. A time-critical adaptive approach for visualizing natural scenes on different devices. *PLoS ONE*, 2:1–26, 2015.
- [101] C. Rebollo, I. Remolar, J. Gumbau, and M. Chover. Three-dimensional trees for virtual globes. *International Journal of Digital Earth*, 7(10):789–810, 2014.

- [102] F. Ramos, M. Chover, O. Ripollés, and C. Granell. Continuous level of detail on graphics hardware. *In Proceedings of the 13th International Conference on Discrete Geometry for Computer Imagery*, pages 460–469, 2006.
- [103] O. Ripollés, J.F. Ramos, A. Puig-Centelles, and M. Chover. Real-time tessellation of terrain on graphics hardware. *Computers & Geosciences*, pages 147–155, 2012.
- [104] W. Li, D. Han, H. Li, X. Wang, and J. Zhu. Extraction of digital terrain model based on regular mesh generation in mountainous areas. *Multimedia Tools and Applications*, 77:6267–6286, 2018.
- [105] P. Prusinkiewicz and M. Hammel. A fractal model of mountains with rivers. *In Proceedings of Graphics Interface'93*, pages 174–180, 1993.
- [106] C. Rebollo, I. Remolar, M. Chover, and J.F. Ramos. A comparison of multiresolution modelling in real-time terrain visualisation. *Springer-Verlag Berlin Heidelberg*, 3044:703–712, 2004.
- [107] A. Emilien, A. Bernhardt, A. Peytavie, M.P. Cani, and E. Galin. Procedural generation of villages on arbitrary terrains. *Visual Computer*, 28(6-8):809–818, 2012.
- [108] F. Hou, H. Qin, and Y. Qi. Procedure-based component and architecture modeling from a single image. *Vis. Comput*, 32(2):151–166, 2016.
- [109] R. Pajarola and E. Gobbetti. Survey of semi-regular multiresolution models for interactive terrain rendering. *Vis. Comput*, 23(8):583–605, 2007.
- [110] A. Puig-Centelles, O. Ripollés, and M. Chover. Creation and control of rain in virtual environments. *The Visual Computer*, 25(11):1037–1052, 2009.

- [111] K. Perlin. Improving noise. *In Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 681–682, 2002.
- [112] I. Parberry. Designer worlds: Procedural generation of infinite terrain from real-world elevation data. *Journal of Computer Graphics Techniques (JCGT)*, 3(1):74–85, 2014.
- [113] E. Bruneton and F. Neyret. Real-time rendering and editing of vector-based terrains. *Computer Graphics Forum. Special Issue: Eurographics '08*, 27(2):311–320, 2008.
- [114] Planetside Software. Available online: <https://planetside.co.uk/>, 2018. Last accessed January 2021.
- [115] WM WorldMchine Software. Available online: <https://planetside.co.uk/>, 2018. Last accessed January 2021.
- [116] E on Software. Available online: <https://info.e-onsoftware.com/more-info-vue>, 2018. Last accessed January 2021.
- [117] Bundy Soft. Available online: <http://www.bundysoft.com/L3DT/>, 2018. Last accessed January 2021.
- [118] Yoon J.C. and Lee IK. Stable and controllable noise. *Graphical Models*, 70(5):105–115, 2008.
- [119] Wikiloc. Routes of the world. Available online: <https://es.wikiloc.com>, 2018. Last accessed January 2021.
- [120] Unity 3D. Available online: <https://unity3d.com/>, 2018. Last accessed January 2021.
- [121] S. Griffon, A. Nespoulous, J. Cheylan, P. Marty, and D. Auclair. Virtual reality for cultural landscape visualization. *Virtual Reality*, 15:279–294, 2011.

- [122] S. V. Grigore and A. Sabou. Real-time guided procedural terrain generation. *Proceedings of the International Conference on Human-Computer Interaction (RoCHI 2017)*, pages 159–162, 2017.
- [123] C. Rebollo, I. Remolar, M. Chover, and J. F. Ramos. A comparison of multiresolution modelling in real-time terrain visualisation. In *Computational Science and Its Applications – ICCSA 2004*, pages 703–712, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
- [124] Eric Galin, Eric Guérin, Adrien Peytavie, Guillaume Cordonnier, Marie-Paule Cani, Bedrich Benes, and James Gain. A review of digital terrain modeling. *Computer Graphics Forum*, 38, 05 2019.
- [125] Fabricio Anastacio, Przemyslaw Prusinkiewicz, and Mario Costa Sousa. Sketch-based parameterization of l-systems using illustration-inspired construction lines and depth modulation. *Computers & Graphics*, 33(4):440–451, 2009.
- [126] Jamie Wither, Frédéric Boudon, Marie-Paule Cani, and Christophe Godin. Structure from silhouettes: a new paradigm for fast sketch-based design of trees. *Computer Graphics Forum*, 28(2):541–550, 2009. Special issue: Eurographics 2009.
- [127] Steven Longay, Adam Runions, Frédéric Boudon, and Przemyslaw Prusinkiewicz. TreeSketch: Interactive Procedural Modeling of Trees on a Tablet. In Karan Singh and Levent Burak Kara, editors, *Eurographics Workshop on Sketch-Based Interfaces and Modeling*. The Eurographics Association, 2012.
- [128] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The algorithmic beauty of plants*. Springer Science & Business Media, 2012.
- [129] L. Chauchard, González M. Sbrancia R., L. Maresca, Rabino A., and Mazzuchelli M. Applicability of density rules on

nothofagus forests: li. the minimum stocking line and stand-density management diagram. *Bosque*, 22(2):3–10, 2001.

- [130] H. Pretzsch, P. Biber, E. Uhl, J. Dahlhausen, T. Rötzer, J. Caldentey, T. Koike, T. van Con, A. Chavanne, T. Seifert, B. Toit, C. Farnden, and S. Pauleit. Crown size and growing space requirement of common tree species in urban centres, parks, and forests. *Urban Forestry & Urban Greening*, 14(3):466–479, 2015.
- [131] C. Damgaard, J. Weiner, and H. Nagashima. Modelling individual growth and competition in plant populations: Growth curves of *Chenopodium album* at two densities. *Journal of Ecology*, 90:666 – 671, 08 2002.
- [132] S. Aikio and S. Pakkasmaa. Relatedness and competitive asymmetry - implications for growth and population dynamics. *Oikos*, 100:283–290, 02 2003.
- [133] S. Aikio. Competitive asymmetry, foraging area size and coexistence of annuals. *Oikos*, 104:51–58, 2004.
- [134] Bedřich Beneš, Nathan Andrysco, and Ondřej Št'ava. Interactive modeling of virtual ecosystems. *Proceedings of the Fifth Eurographics Conference on Natural Phenomena*, page 9–16, 2009.
- [135] CREAf. Centre de recerca ecològica i aplicacions forestals. Available online: <http://www.creaf.cat> , 2019. Last accessed January 2021.
- [136] CREAf. Centre de recerca ecològica i aplicacions forestals. Available online: <http://www.creaf.uab.es/iefc/pub/Catalunya/MenuEspecies.htm>, 2019. Last accessed January 2021.

- [137] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.*, 13(1):21–27, 2006.
- [138] V. Gimeno, S. Markvorsen, and J. M. Sotoca. A lower bound for the area of plateau foams. *Cornell University*, *arxiv:1906.08537v2*, 2019.
- [139] Vea Games. Landscape auto material. Available online: <https://www.veagames.com/lam-ue4>, 2019. Last accessed January 2021.
- [140] H. Pretzsch, C. Matthew, and J. Dieler. *Allometry of Tree Crown Structure. Relevance for Space Occupation at the Individual Plant Level and for Self-Thinning at the Stand Level*, pages 287–310. Springer, 2012.
- [141] ARBOLAPP. Arbolapp.es. Available online: <http://www.arbolapp.es/>, 2019. Last accessed January 2021.
- [142] J. Ribelles, M.A. López, O. Belmonte, I. Remolar, and M. Chover. Multiresolution modeling of arbitrary polygonal surfaces: A characterization. *Comput. Graph.*, 26:449, 2002.
- [143] P.S. Heckbert and M. Garland. Multiresolution modeling for fast rendering. In *Proceedings of the Graphics Interface*, pages 19–20, 1994.
- [144] D.P. Luebke. A developer’s survey of polygonal simplification algorithms. *IEEE Comput. Graph. Appl.*, 21:24, 2001.
- [145] José Ribelles, Angeles López, Inmaculada Remolar, Oscar Belmonte, and Miguel Chover. Multiresolution modelling of polygonal surface meshes using triangle fans. In *Discrete Geometry for Computer Imagery*, pages 431–443, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

- [146] Jaume Rigau, Miquel Feixas, and Mateu Sbert. Shape complexity based on mutual information. *In Proceedings of the International Conference on Shape Modeling and Applications, Cambridge, MA, USA.*, pages 355–360, 2005.
- [147] Xfrog. Xfrog: Xfrog plants. Available online: <http://xfrog.com/category/xfrogplants.html>, 2018. Last accessed January 2021.
- [148] Epic Games. Unreal engine. Available online: <http://www.unrealengine.com/>. Accessed on January 2021.
- [149] José Ribelles, Miguel Chover, Angeles López, and Joaquín Huerta. Frame-to-frame coherence of multiresolution ordered meshes. *Proc. of CEIG*, 99:91–104, 1999.
- [150] Cristina Gasch, Inmaculada Remolar, Miguel Chover, and Cristina Rebollo. Viewpoint-Driven Simplification of Plant and Tree Foliage. *Entropy*, 20(4):213, March 2018.
- [151] J. Ribelles, M. Chover, A. Lopez, and J. Huerta. A first step to evaluate and compare multiresolution models. In *Eurographics 1999 - Short Presentations*. Eurographics Association, 1999.
- [152] Carlos González, Pascual Castelló, and Miguel Chover. A texture-based metric extension for simplification methods. In *GRAPP (GM/R)*, pages 69–76, 2007.