Universidad Politécnica de Valencia

Departamento de Sistemas informáticos y Computación

Modelos multirresolución para la aceleración de la visualización de entornos naturales

Tesis Doctoral

AUTOR: JAVIER LLUCH CRESPO

DIRECTOR: DR. ROBERTO VIVÓ HERNANDO

VALENCIA, JUNIO 2002

Resumen

La visualización interactiva de grandes conjuntos de datos es uno de los temas que polarizan la investigación actual en el campo de la informática gráfica. Aunque el diseño de hardware gráfico avanza a pasos agigantados también es cierto que cada vez se pretende visualizar de forma interactiva escenas con un mayor número de primitivas. Entre las escenas que exigen la utilización de una gran cantidad de geometría para ser representadas destacan los entornos naturales. Este tipo de escenas se utilizan en aplicaciones de simulación, realidad virtual y juegos. Es esencial la aplicación de métodos que permitan la aceleración de la visualización para obtener un número de imágenes por segundo suficiente para evitar efectos no deseados de parpadeo.

En la presente tesis, se desarrolla un modelo multirresolución que permite la representación de un árbol en distintos niveles de detalle. El nivel con menor detalle representa al árbol con sólo seis texturas que representan el follaje y unos cuantos polígonos para representar a las ramas. El máximo nivel de detalle representará cada hoja como un polígono con una textura que representa a la hoja, y contendrá todas las ramas del árbol. La obtención de los niveles de detalle es automática y permite la visión del árbol desde cualquier posición y orientación. El modelo está preparado para su inclusión en escenas formadas por diferentes tipos de árboles.

Palabras clave: informática gráfica, modelado multirresolución, modelado procedural, visualización basada en la imagen.

Resum

La visualització interactiva de grans conjunts de dades és un dels temes que polaritzen la investigació actual de la informàtica gràfica. Tot i que el disseny de hardware gràfic avança ràpidament, cada vegada més es pretén visualitzar de manera interactiva escenes amb un major nombre de primitives. Entre les escenes que exigeixen la utilització d'un gran quantitat de geometria per representar-se, destaquen els entorns naturals. Aquest tipus d'escenes s'utilitzen en aplicacions de simulació, realitat virtual i jocs. És essencial l'aplicació de mètodes que permiten l'acceleració de la visualització per obtenir un nombre d'imatges per segon suficient per evitar efectes no desitjables de parpelleig.

En la tesi presentada, s'hi desenvolupa un model multirresolució que permet la representació d'un arbre en nivells diferents de detall. El nivell amb el menor detall representa l'arbre amb només sis textures que representen el fullam i una sèrie de polígons per representar les branques. El nivell de detall màxim representa cada fulla com un polígon amb una textura que la fulla i conté totes les branques de l'arbre. L'obtenció dels nivells de detall és automàtica i permet la visió de l'arbre des de qualsevol posició i orientació. El model està preparat per la inclusió en escenes formades per diferents tipus d'arbres.

Paraules clau: informàtica gràfica, modelatge multirresolució, modelatge procedural, visualització basada en la imatge.

Abstract

Interactive rendering of large datasets is one of the topics of current research in the field of computer graphics. Even though progress in graphics hardware design is moving fast, it is also the case that we want to interactively render scenes with increasing numbers of primitives. Among the scenes that require the largest amounts of geometry we focus on natural environments. This type of scenes is commonly used in applications like simulation, virtual reality and games. It is essential that we apply methods that accelerate rendering in order to obtain a frame rate high enough to avoid the undesired flicker effect.

In this dissertation we develop a multiresolution model that allows the representation of a tree with different levels of detail. The lowest level of detail stores six textures to represent the foliage, and a few polygons to represent all the branches of the tree. The highest level of detail represents the leaves as texture-mapped polygons and all the branches as a detailed polygonal model. We can extract levels of detail automatically and render a tree from any viewing position and orientation. The model is ready to be integrated in scenes made of different types of trees.

Keywords: computer graphics, multiresolution models, procedural models, image-based rendering.

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

En primer lugar deseo agradecérselo a mis padres porque han hecho posible que llegue este momento. A mis hermanos Gemma, Vicent y Mar por ser más que hermanos, amigos. A mi cuñado Toni, por mis descansos en el paraíso. A mi sobrina Andrea y mi ahijado Andreu, por ser tan bonicos. A mi prima Maite y mis sobrinas Raquel y Sara, por que es todo un placer cocinarles la paella del domingo. A mis colegas del GIG de la UJI, Riki, Joaquín, Miguel, Michael, Pepe, Oscar e Inma, a los de la SIG de la UPV Emilio, MªJosé, Mon, Paco, Inma, Carlos y Mª Carmen, y a los becarios Sergio, Pedro y Toni, por la ayuda prestada, y la compañía en los viajes por esos mundos de la investigación. A mis amigos Riki, Joaquín, Emilio, Charlie, Leo, Oscar, Palo, Carlos, Gus, Celias, Anas, Helena, Carmen, Jeannette y Cristina, por las tertulias, por su comprensión y por hacerme sentirme especial. Por último, y no por ello menos importante, un agradecimiento especial al sheriff, director y guía espiritual Roberto.

Capítulo 1
Introducción

La visualización interactiva de grandes conjuntos de datos es uno de los temas que polarizan la investigación actual en el campo de la informática gráfica. Aunque el diseño de hardware gráfico avanza a pasos agigantados también es cierto que cada vez se pretende visualizar de forma interactiva escenas con un mayor número de primitivas.

Por ejemplo, si tenemos una escena con 10 millones de polígonos y tenemos un moderno acelerador gráfico que permite visualizar 5 millones de polígonos por segundo, si utilizamos un algoritmo de fuerza bruta obtendremos una imagen cada dos segundos, lo cual es totalmente inaceptable para un sistema interactivo. Si además añadimos efectos de iluminación y texturas a la escena la razón aún disminuirá más.

Por lo tanto, es imposible visualizar este tipo de escenas masivas consiguiendo un número de imágenes por segundo suficiente para que no se produzcan efectos de parpadeo sin utilizar técnicas software que aceleren el proceso. Un ejemplo de este tipo de escenas son las poblaciones vegetales, como la que aparece en la figura 1.1, ya que exige la utilización de un gran número de primitivas para modelar cada uno de los individuos.

La presente tesis está enmarcada dentro de los proyectos que desarrolla la sección de informática gráfica del departamento de sistemas informáticos y computación:

- Proyecto GV97-TI-05-42: Modelización y visualización de entornos naturales virtuales. Aplicación a la simulación de la visión robótica de campos frutales. (1998-99)
- Proyecto TIC99-0510-C02-01: Arquitecturas multirresolutivas de sistemas gráficos procedurales (2000-02)

En el proyecto GV97-TI-05-42, ya concluido, se obtuvieron los siguientes resultados: la evaluación de los sistemas de modelado evolutivos, el desarrollo de un sistema con sensibilidad al contexto para la generación de individuos, la integración del modelo en poblaciones de árboles en agricultura y la inclusión de un agente móvil en la escena. Se desarrollaron dos entornos con tecnología propia: el GREEN y el EVERGREEN.

El primer entorno permite la construcción mediante sistemas paramétricos aleatorios de Lindermayer de individuos particulares incluyendo características de sensibilidad al contexto, poda, flexibilidad modular y multi-interpretación. El segundo entorno se desarrolló con la filosofía de un navegador. La estructuración de una población de árboles en forma de campo agrícola se consigue replicando el individuo, generado por el primer entorno, con ligeras variaciones, la generación de terreno en forma de mapa de alturas, la envoltura del campo en un entorno de cielo y vallas simulado, la inclusión de un agente móvil con características

parametrizables de movimiento y la implementación de la interacción con el agente. Los principales resultados fueron la consecución de tiempos aceptables de navegación mediante el ensayo de diferentes técnicas de nivel de detalle, la interfaz de navegación multiplataforma, la generación de terreno, el rendering de la imagen visualizada por el robot y la animación por la fuerza del viento.

Figura 1.1: Vista de un campo de naranjos

El segundo proyecto, TIC99-0510-C02-01, desarrollándose en la actualidad, tiene como objetivo principal la representación interactiva de entornos virtuales y su transmisión electrónica progresiva, ya que suponen unos de los campos abiertos en la investigación relacionada con la Realidad Virtual. El área es conocida con el nombre general de Sistemas Multirresolución. El objetivo del proyecto se enmarca en esta línea de investigación y está dirigido a la consecución de modelos multirresolución novedosos aplicables a un amplio espectro de modelos gráficos. Para ello se eligieron elementos de la naturaleza, los árboles y plantas, como banco de pruebas, por la alta complejidad geométrica que supone su representación. Esta complejidad viene dada no sólo por el número de polígonos necesario para representarlos, sino también por su topología, que hace difícil la aplicación de las técnicas de simplificación más extendidas.

1.1 Motivación

Trabajar con estos modelos de gran complejidad, formados por cientos de miles de primitivas produce serios problemas, principalmente en tres aspectos:

- **Visualización.** Cada vez son más las aplicaciones gráficas que exigen interactividad con el usuario. En algunas, como la realidad virtual, llegan a ser

críticas ya que además necesitan visualizar una tasa de, aproximadamente, 30 imágenes por segundo, resultando imposible conseguirlo ya que las prestaciones de las mejores estaciones gráficas se ven ampliamente superadas.
- **Almacenamiento.** Debido a la gran cantidad de información utilizada en la representación de los objetos, es necesario un espacio de almacenamiento mayor en disco o memoria, y al mismo tiempo, la recuperación de los datos se hace menos eficiente.
- **Transmisión.** Es evidente que el tiempo necesario para transmitir un modelo es función de su tamaño. Con los anchos de banda actuales transmitir un modelo por una línea de comunicación puede ser muy lento. Además, la explotación de internet para distribuir y compartir datos aumenta la importancia de este problema.

Por lo tanto, la búsqueda de soluciones eficientes para este problema es un campo que en la actualidad ocupa un lugar prominente dentro de la investigación en la informática gráfica. Los modelos que se utilizan actualmente para la visualización de poblaciones vegetales tienen una serie de inconvenientes, que deben ser resueltos para poder generar el mayor número posible de imágenes por segundo y con el mayor grado de realismo.

El modelo más extendido tanto en juegos como en sistemas de simulación y realidad virtual es el de los *billboards*, es decir la utilización de polígonos, que siempre están orientados hacia el observador, y a los que se les aplican texturas de árboles. Este tipo de representaciones tiene la gran desventaja de que cuando se realiza una inspección cercana del polígono, se pierde totalmente el realismo. Por lo tanto, un modelo debe de tener en cuenta tanto la eficiencia de visualización como el grado de realismo que ofrece. Otros modelos con gran realismo, necesitan de un artista 3D para dibujar cada una de las texturas que forman el árbol.

En la presente tesis se va a desarrollar un modelo multirresolución que permite la representación de un árbol en distintos niveles de detalle. El nivel con menor detalle representa al árbol con sólo seis texturas que representan el follaje y unos cuantos polígonos para representar a las ramas. El máximo nivel de detalle representará cada hoja como un polígono con una textura que representa a la hoja, y contendrá todas las ramas del árbol. La obtención de los niveles de detalle es automática.

1.2 Objetivos

El objetivo principal que se plantea conseguir con la realización de la tesis es la aceleración de la visualización de una escena formada por múltiples elementos vegetales, de modo que sea posible una inspección de los elementos desde cualquier posición y orientación dentro de la escena. La situación ideal sería poder simular la visión de una ardilla que se moviera por el interior de la escena.

Para conseguir este objetivo va a ser necesario formular un modelo multirresolución para crear una estructura de datos que represente a los distintos individuos y que permita su visualización y transmisión a través de la red en distintos niveles de detalle dependiendo de ciertos parámetros tales como la distancia al observador o la orientación de éste con respecto al individuo. La búsqueda de la solución al problema se va a enfocar teniendo en cuenta distintos aspectos de la escena:

- **General:** Utilizando técnicas de selección para enviar al visualizador sólo aquellos individuos que estén situados dentro del campo de visión del observador,

y de nivel de detalle para las primitivas que representan elementos de la escena que estén más alejados del observador.
- **Sólo hojas:** Utilización de imágenes precalculadas que representen el follaje de distintas partes del árbol para disminuir el número de polígonos que hay que visualizar.
- **Sólo ramas:** Conversión de la estructura de ramas en una malla poligonal a la que aplicar técnicas de multirresolución y simplificación poligonal. También es posible obtener una estructura multirresolución que represente al conjunto de ramas con distintos números de primitivas.
- **Estructura de datos:** Creación de una estructura de datos que represente un modelo que contenga las ramas y las hojas del modelo.

En definitiva se trata de obtener un nuevo modelo que permita la representación de las ramas y las hojas de un árbol en diferentes niveles de detalle, que la obtención de los niveles de detalle sea automática, que su visualización sea lo más rápida y realista posible, que su tamaño no sea excesivo y que pueda ser transmitido de forma incremental.

1.3 Sumario

A la vista de los objetivos perseguidos, se plantea el desarrollo de la presente tesis en siete capítulos y un apéndice de bibliografía:

- En el segundo capítulo, se repasan los antecedentes necesarios para poder comenzar a trabajar en el desarrollo de la tesis. Se describen las diferentes técnicas de modelado de vegetales, haciendo hincapié en los sistemas de reescritura en general y los sistemas-L en particular. A continuación, se realiza un estudio de la visualización interactiva resaltando los últimos avances en navegación de poblaciones vegetales. Por último, se fijan las líneas de trabajo a seguir.
- En el tercer y cuarto capítulos, se analizan las aplicaciones desarrolladas que han provocado la realización de la tesis. GREEN (GeneradoR de Especies para Entornos Naturales) es un editor y visualizador de elementos vegetales basado en los sistemas-L. EVERGREEN (Entorno de Visualización de Escenas en el exterioR basado en GREEN) es una aplicación que permite navegar en una escena formada por árboles modelados mediante GREEN.
- En el quinto capítulo, se describen los modelos que se han desarrollado en la realización de la tesis. En primer lugar se expone el modelo que permite la representación de las ramas de un árbol mediante una única malla poligonal. A continuación, se describe el modelo de multirresolución procedural para la representación de las ramas del árbol. En el siguiente apartado, se aborda el problema de la representación del follaje, y se trata utilizando un modelo multirresolución basado en la imagen que hace uso de texturas precalculadas para sustituir la masiva geometría que forman las hojas del árbol. Por último, se describe el modelo que permite la representación del árbol completo y se analizan las posibilidades de inclusión del modelo en una población de elementos vegetales.
- En el sexto capítulo, se exponen los resultados de todos los modelos que se han propuesto en el capítulo anterior
- En el último capítulo, se analizan las conclusiones a las que se ha llegado con el desarrollo de la tesis. También se enumeran las publicaciones escritas hasta el momento y que están relacionadas con la tesis. Por último, se hace un análisis de las posibles extensiones que se pueden realizar al trabajo.

- La tesis termina con un apéndice donde se incluyen todas las referencias utilizadas para su elaboración, y otro que describe algunas de las partes que forman el sistema GREEN.

Capítulo 2
Modelado de vegetación. Antecedentes

La utilización de modelos para describir distintos aspectos de una entidad concreta o abstracta está muy extendida en todas las actividades humanas. La función de un modelo es permitir que se visualice y se entienda la estructura y/o el comportamiento de la entidad, además de proporcionar un medio para poder experimentar y predecir los efectos de distintas entradas o posibles cambios del modelo.

En informática gráfica los modelos más comunes son los siguientes:

- Modelos de organización: Son jerarquías que representan, mediante grafos dirigidos, burocracias y taxonomías.
- Modelos cuantitativos: Son ecuaciones que describen sistemas demográficos, climáticos, físicos, matemáticos, etc. Generalmente se representan mediante grafos y gráficos estadísticos.
- Modelos geométricos: Son conjuntos de componentes con sus propiedades geométricas y las distintas interconexiones entre los mismos que describen estructuras arquitectónicas, geográficas, moleculares y otras estructuras químicas, etc. Estos modelos se representan mediante listas, diagramas de bloques, etc.

Los modelos geométricos son los más extendidos dado que la transformación entre distintos modelos y la visualización de los mismos se realiza de forma muy eficiente. Sin embargo la mayoría de los fenómenos naturales no se pueden representar de forma eficiente mediante estos modelos, al menos a gran escala. Por ejemplo, la niebla está formada por gran cantidad de minúsculas partículas de agua, si cada partícula se representa individualmente, el problema se hace intratable, además la percepción de la niebla es borrosa y no como millones de partículas individuales. Esto se debe a que esta percepción se basa en como afecta la niebla a la luz que llega a nuestros ojos, y no en la forma o el lugar donde se encuentren las partículas de agua. Por ello, para representar fenómenos de este tipo se necesitan unos modelos distintos.

Estos modelos deben permitir representar fenómenos complejos de una forma simple, de modo que mediante la utilización de parámetros y ajustes del modelo se puedan representar gran cantidad de objetos. Por ejemplo, a partir de un modelo que representa una especie de un árbol se pueda crear distintos individuos de la misma especie. Los modelos que van a permitir realizarlo se denominan procedurales.

El modelado procedural define un objeto a través de un procedimiento, por ello el modelo descrito puede interactuar con eventos externos que lo modifican. Por ejemplo, un modelo de

una esfera que genera una representación de la misma dependiendo de tres parámetros: centro, radio y otro que indique la resolución de subdivisión de la misma, es un modelo procedural, sin embargo, el conjunto de polígonos que forman la esfera no será procedural.

Las propiedades más importantes son el ahorro de memoria, la capacidad de interactuar con el entorno y la de definir la evolución temporal, por lo tanto son modelos muy compactos.

Los principales modelos procedurales son los siguientes:

- Modelos fractales.
- Modelos basados en gramáticas o sistemas de reescritura.
- Sistemas de partículas.

En el siguiente apartado, se van a comentar los diferentes modelos utilizados por otros autores para el modelado de especies vegetales. En la sección 2.2, se habla del modelado procedural de especies vegetales haciendo hincapié, en el siguiente apartado, en el modelo utilizado en la presente tesis: sistemas paramétricos aleatorios. Posteriormente, se hace un estudio de las técnicas utilizadas en las aplicaciones de navegación interactiva de entornos naturales. En el siguiente apartado, se exponen los principales avances que se han producido en las técnicas específicas que permiten la aceleración de la visualización interactiva de poblaciones vegetales. Por último, se discute la problemática actual y los objetivos que se marcaron en la realización de la tesis.

2.1 Propuestas principales al modelado de vegetación

El modelado de especies vegetales es un campo de la informática gráfica que ha sido objeto de estudio por muchos autores. Entre las distintas aproximaciones que existen se distinguen dos tendencias principales: el modelado basado en principios botánicos y el modelado que busca el realismo visual sin cumplir necesariamente los fenómenos naturales.

Inicialmente, el modelado de especies vegetales tuvo una motivación biológica, los primeros trabajos de Lindenmayer y posteriormente de Prusinkiewicz [Prus90] utilizaron sistemas de reescritura que operaban sobre un conjunto de reglas. Esta aproximación incluye sensibilidad al contexto, así como comportamiento estocástico. También se han realizado extensiones para simular la interacción con el entorno del individuo. El modelo describe las plantas en términos de reglas de crecimiento local, basadas en conocimientos biológicos, lo cual hace que el modelado no sea muy intuitivo, ya que el control de aspectos globales es complicado. A pesar de ello, este modelo tiene otra serie de ventajas, que se comentarán posteriormente, que han llevado a elegirlo para el desarrollo de este trabajo.

Otros modelos (Oppenheimer [Oppe86], Reffye[Reff88] y Holton[Holt94]) basados en principios botánicos proporcionan algunos parámetros que permiten un control global más sencillo.

Por otra parte, entre los modelos que buscan una simulación de formas vegetales sin tener en cuenta de forma estricta los principios botánicos se encuentran: los sistemas de partículas [Reev85], Weber [Webe95] y el modelo más reciente de Lintermann y Deussen [Lint99].

A continuación se va a presentar una breve descripción de los métodos comentados.

2.1.1 Basados en principios botánicos

Oppenheimer [Oppe86] utiliza un modelo fractal que permite generar plantas, árboles, hojas e incluso estructuras inorgánicas como deltas de ríos y copos de nieve. La geometría y la topología del modelo se obtienen mediante parámetros numéricos que son análogos al ADN. El modelo obtenido tiene las siguientes características:

- Parametrización detallada de la relación geométrica entre los nodos del árbol.
- Aplicación de modelado estocástico tanto para los parámetros geométricos como topológicos.
- Modelado estocástico de la corteza del árbol.

El árbol se obtiene mediante un proceso recursivo en el que un nodo del árbol es una rama con uno o más nodos asociados. El proceso se detiene cuando el tamaño de la rama generada alcanza un umbral, entonces se genera una hoja. La parametrización que regula la relación entre los nodos determina el ángulo entre el tronco y las ramas, el tamaño del tronco y las ramas, la medida en que se estrechan las ramas y el número de ramificaciones por rama. Estos parámetros pueden tener variaciones aleatorias para evitar la autosimilaridad fractal y darle al modelo un mayor realismo.

El modelo descrito por Reffye et al [Reff88], se basa completamente en principios botánicos teniendo en cuenta: cómo crecen, cómo ocupan el espacio o dónde y cómo se localizan las hojas, flores y frutos. El objetivo del modelo es obtener imágenes de plantas y árboles lo más fieles a su naturaleza botánica como sea posible y construir un modelo que incluya las leyes botánicas conocidas que explican el crecimiento y arquitectura de las plantas.

La idea general del método es modelar la actividad de los brotes en espacios de tiempo discretos, de modo que un brote en un momento determinado (señal de reloj) puede:

- convertirse en una flor y morir,
- dormirse,
- convertirse en un internodo, al final del cual puede aparecer una o más hojas, con nuevos brotes laterales asociados y un nuevo brote principal, o
- morir

Estos eventos se producen siguiendo unas leyes estocásticas que caracterizan cada variedad o cada especie. Estas leyes también controlarán las propiedades geométricas del modelo. Para simular el crecimiento de la planta se deberán controlar los siguientes parámetros:

- La edad
- La velocidad de crecimiento de las ramas
- El número de posibles brotes de cada nodo
- Las probabilidades de muerte, pausa, ramificación o reiteración de cada brote dadas como función de la edad, tamaño y nivel del mismo
- El tipo de unidad de crecimiento y el número de nodos por unidad de crecimiento dadas como función de los mismos parámetros que antes
- Los parámetros geométricos de un internodo: longitud y diámetro
- Para cada nivel de ramificación, la tendencia de desarrollo, el ángulo de intersección y la filotaxis (distribución de las hojas a lo largo del tallo).

Una vez definidos estos parámetros, la obtención del modelo se realiza recorriendo cada instante de tiempo desde el nacimiento de la planta hasta la edad deseada. Para cada instante

se deben tener en cuenta todos los brotes que estén vivos para aplicarles distintos tests, dependiendo del resultado del test, el brote puede desaparecer de la lista de brotes vivos o puede añadirse un nuevo internodo. El resultado del procedimiento es un conjunto de elementos (internodos, hojas, flores y frutos) que forman la estructura del árbol.

Holton[Holt94] utiliza un modelo procedural que asigna una ruta a cualquier camino que vaya desde la raíz a una hoja del árbol, basándose en el modelo de tuberías (*pipe model*) que ha sido utilizado para explicar la ramificación de los ríos, la formaciones bronquiales de los pulmones y en botánica como un proceso subyacente en la generación de los árboles y plantas. El número de rutas en una rama determinará su grosor, su longitud, el ángulo de ramificación y el número de ramas hijas y hojas que generará. Todas estas características son parametrizables, de modo que dependiendo de los valores de los parámetros se podrá simular fenómenos de gravedad, fototropismo, ortotropismo, plagiotropismo y planartropismo. El modelo resultante es muy realista, aunque el proceso de generación es muy costoso.

2.1.2 No basados en principios botánicos

Una de las primeras técninas que se utilizaron para modelar árboles fueron los sistemas de partículas [Reev85]. Los sistemas de partículas se utilizaron por primera vez para modelar una explosión en la película StarTrek II. Cada partícula de fuego llevaba asociados una serie de parámetros que permiten describir la posición actual de la partícula y otra serie de características tales como la velocidad, trayectoria y posición iniciales, color, tamaño, tiempo de vida, etc. A cada parámetro de cada partícula se le asigna un valor en función de una distribución aleatoria, variando su valor a lo largo del tiempo de forma independiente a las demás partículas.

Sin embargo, los sistemas de partículas que modelan fenómenos naturales como árboles o hierbas son más estructurados, por ello las partículas no son independientes. Cada árbol se dibuja mediante un conjunto de líneas y pequeños círculos, que constituyen las ramas y las hojas. Existen muchas relaciones complejas entre las partículas que representan las ramas y las hojas, ya que todas ellas deben de representar un objeto tridimensional con cohesión.

El modelado del árbol parte del tronco principal y posteriormente se obtiene el resto del árbol mediante la generación recursiva de las ramas hijas. La estructura de datos es un árbol donde cada nodo describe un segmento de rama. Antes de iniciar la generación de las ramas se deben inicializar una serie de parámetros que determinarán las características y dimensiones del árbol. Los valores de los parámetros se definen mediante distribuciones aleatorias que determinarán el tipo del árbol.

La distribución de la longitud de cada rama depende de las dimensiones del árbol (Figura 2.1), para ello se calcula un volumen de inclusión aproximado, cuya forma (cónica o elíptica) variará dependiendo del tipo de árbol que se esté generando. Las ramas hijas heredan muchos de los parámetros de sus padres, aunque otros se ajustan a su tamaño. La recursión termina cuando se alcanza el grosor de rama mínimo o un nivel de recursión máximo.

Cada especie tiene una probabilidad de ramificación distinta. Entre una rama y una rama hija existe una relación madre - hija, sin embargo dos ramificaciones de una misma rama tienen una relación de hermanas y comparten distribuciones de probabilidad. Este algoritmo genera árboles con una estructura muy regular, por ello se realizan post-procesos para modelar la gravedad, viento y fototropismo.

Una vez creada la estructura de ramas se añaden las hojas a aquellas ramas que no tienen ramas hijas. Cada hoja tendrá una serie de parámetros, que se definirán de manera estocástica, para determinar su forma, orientación, espaciado, densidad, color y posición.

Weber [Webe95] propone un método basado en la parametrización de ciertas características de los árboles que se han obtenido mediante la observación de distintas especies. El modelo se parece al de Oppenheimer aunque se evita la autosimilaridad para no limitar el potencial de representación del método. Las ramas hijas heredan ciertas propiedades de la madre pero otras pueden ser completamente distintas.

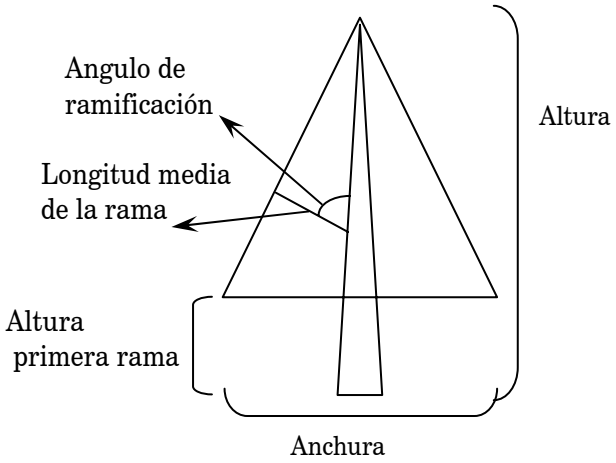


Figura 2.1: Dimensiones del modelo de Reeves

El modelo parte del tronco principal el cual puede dividirse varias veces lo largo de su longitud formando estructuras curvadas similares, las cuales pueden volver a subdividirse. El modelo está formado por ramas (incluyendo el tronco) y hojas. La forma final del árbol se determinará por una serie de parámetros que controlarán:

- La resolución de cada rama
- El número de divisiones de cada rama
- El ángulo de división
- El número de hijos de cada rama
- El radio de cada rama
- El número de hojas de cada rama
- La forma general de la copa mediante poda
- Efectos de viento y gravedad
- La orientación de las hojas

El modelo propuesto por Lintermann [Lint99] está formado por un grafo cuyos nodos contienen información geométrica y estructural. A diferencia de otros modelos de descripción de geometría al estilo de Inventor, los componentes que forman el modelo encapsulan datos y algoritmos para generar los elementos que forman las plantas. La estructura formada permite la representación de objetos naturales ramificados, como plantas y árboles.

Los componentes se pueden dividir en tres categorías, los que:

- producen objetos gráficos como ramas, hojas o primitivas geométricas,
- multiplican algorítmicamente otros componentes, y

- definen técnicas globales de modelado

Todos los componentes tienen una serie de parámetros que controlan su comportamiento. Los componentes se pueden combinar de tres formas:

- Un componente puede ser hijo de otro, por lo que creará su geometría en función de la del padre
- Un componente puede ser las ramas de otro
- Un componente puede ser hijo de un componente multiplicativo, lo que representa tantos componentes como se indique en el factor de multiplicación

Además, puede haber una relación recursiva entre componentes, en este caso se deberá indicar el nivel máximo de recursión.

A continuación se va a realizar un estudio más exhaustivo del modelado de especies vegetales mediante gramáticas, ya que este modelo es el más extendido y el que se ha elegido para la realización del presente trabajo.

2.2 Modelado gramatical de especies vegetales

Los primeros pasos en este campo los dio el biólogo Aristid Lindenmayer que formuló el modelo sistema L [Lind68] para su aplicación en la interacción celular. Este sencillo modelo fue posteriormente desarrollado por Prusinkiewicz [Prus86] para su aplicación a especies vegetales. El modelo se basa en los conceptos de amplificación de datos y reglas de reescritura. Partiendo de un módulo inicial denominado axioma y aplicando en paralelo y de forma sucesiva las reglas, se obtiene una cadena que puede ser interpretada de forma gráfica. Los símbolos que forman el lenguaje pueden tener asociados parámetros. El sistema incluye sensibilidad al contexto que permite la simulación de fenómenos endógenos en los individuos y la posibilidad de aplicar las reglas de forma aleatoria para obtener distintos individuos de la misma especie. Este modelo ha sido uno de los más utilizados y se ha ampliado para la simulación de distintos fenómenos como la interactividad del desarrollo de la planta con su entorno (poda contra una superficie, obstáculos, competencia por la luz). Este apartado se centrará en los avances que se han conseguido en la aplicación de los sistemas L al modelado, simulación y visualización del desarrollo de las plantas. Para ello se seguirán los siguientes pasos:

- Estudio de la arquitectura modular de las plantas y demostración que los aspectos principales de su desarrollo se pueden ver como un proceso de reescritura
- Utilización de la notación de cadenas de símbolos con corchetes para expresar la topología de estructuras ramificadas

2.2.1 Modelos de desarrollo de arquitecturas de plantas

Se considerará una planta como una configuración espacial de unidades discretas de construcción o *módulos,* los cuales se desarrollan a lo largo del tiempo. Típicamente los módulos representan componentes estructurales básicos de una planta, tales como metameros (internudos con una hoja asociada y un brote lateral) y ramas (ver figura 2.2). El objetivo consiste en describir el desarrollo de la planta, y en particular la emergencia de la forma de la planta, como una integración del desarrollo y funcionamiento de los módulos individuales.

La esencia del desarrollo de una planta puede ser vista como un sistema de reescritura que repetidamente reemplaza módulos padre por configuraciones de módulos hijo. Suponiendo que

todos los módulos pertenecen a un conjunto finito de tipos de módulo, el comportamiento de una amplia configuración arbitraria de módulos se puede especificar utilizando un conjunto finito de reglas de reescritura o producciones. Una producción especifica cómo se reemplaza un módulo predecesor único, por una configuración de cero o más módulos sucesores. A este proceso se le denomina derivación y se puede apreciar en la figura 2.3.

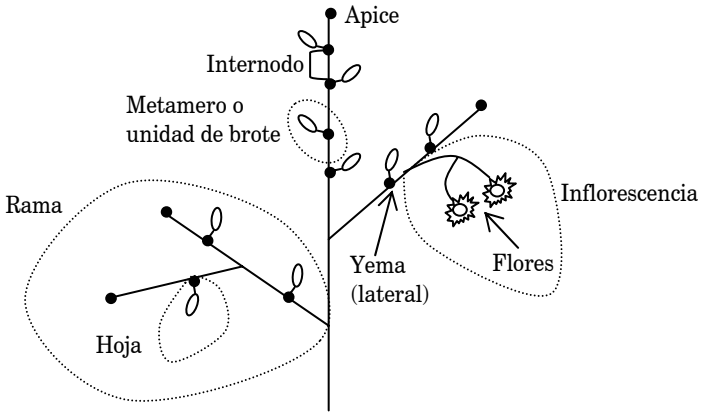


Figura 2.2: Jerarquía de niveles de la organización de una planta.

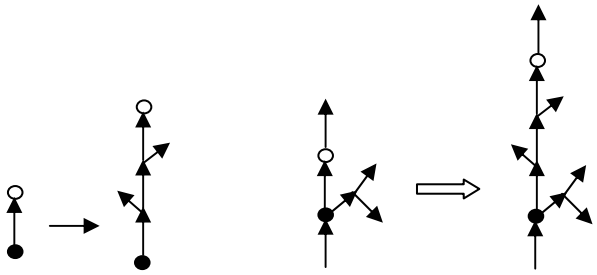


Figura 2.3: Ejemplo de la especificación y aplicación de una producción

2.2.2 Definiciones y notación

A fin de considerar la arquitectura de la planta de una forma abstracta, se utiliza la notación de árbol axial (ver figura 2.4) que complementa la noción de árbol utilizada en teoría de grafos como una estructura de datos, con la noción botánica de rama. Una estructura de datos árbol tiene aristas etiquetadas y dirigidas, y forma caminos desde un nodo específico denominado base hasta los nodos terminales. En un contexto biológico, las aristas son los segmentos de rama. Un segmento seguido de al menos uno o más segmentos se llama internodo. Un segmento terminal se denomina ápice.

Un árbol axial es un caso especial de una estructura árbol, en la que se pueden distinguir por lo menos un segmento recto antecedente en cada nodo. Todas las demás aristas se llaman segmentos laterales. Una secuencia de segmentos se denomina eje, si:

- el primer segmento de la secuencia se origina en la raíz del árbol o como un segmento lateral de un nodo,
- cada segmento siguiente es un segmento recto, y
- el último segmento no es seguido por ningún segmento recto en el árbol.

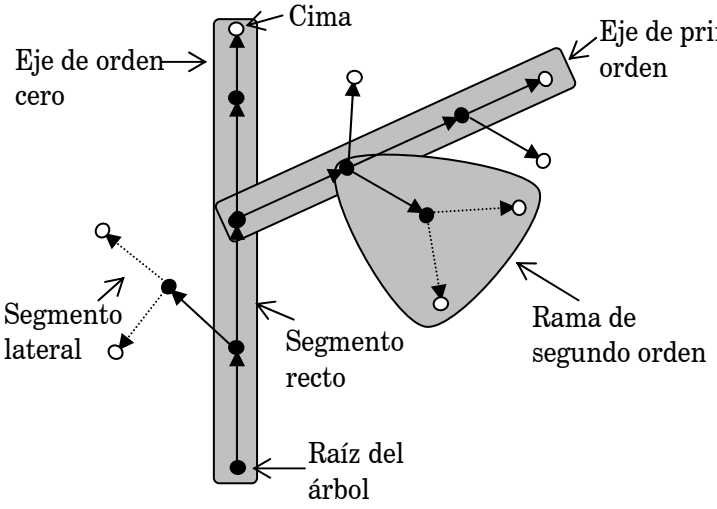


Figura 2.4: Partes de un árbol axial

Junto a todos sus descendientes, un eje forma una rama. Una rama es, por sí misma, un (sub)árbol axial. Los ejes y las ramas están ordenados. Los ejes que se originan en la base de la planta tienen orden cero. Un eje originado como un segmento lateral de un eje padre de orden n, tiene orden n+1. El orden de una rama es el mismo que el orden del eje de menor orden que contenga. El nodo terminal de un eje se denomina copa de la rama.

Para representar árboles axiales, Lindenmayer introdujo la notación de cadenas con corchetes *[]*. Un árbol axial con aristas etiquetadas con símbolos pertenecientes al alfabeto V se representa como una palabra (cadena de símbolos) ω sobre el alfabeto $V_E = V \cup \{[,]\}$:

$$\omega = x_1[\alpha_1]\, x_2[\alpha_2] \ldots\, x_n[\alpha_n]\, x_{n+1}$$

Se asume que las subpalabras $x_1, x_2, \ldots x_n, x_{n+1} \in V^*$ no contienen corchetes, y que las subpalabaras $\alpha_1, \alpha_2 \ldots, \alpha_n \in V_E^*$ están bien anidadas. La palabra $x_1, x_2, \ldots x_n, x_{n+1}$ representa el eje principal (es decir de orden cero), con internodos $x_1, x_2, \ldots, x_n$ y ápice x_{n+1}. Las palabras $\alpha_1, \alpha_2, \ldots, \alpha_n$ representan las ramas de primer orden de ω. Cada rama α_i se puede descomponer de forma similar a ω produciendo un eje de primer orden y posiblemente ramas de segundo orden. Esta descomposición se puede hacer recursivamente. La descomposición de una palabra bien anidada es única, por lo tanto todos los términos introducidos anteriormente no son ambiguos.

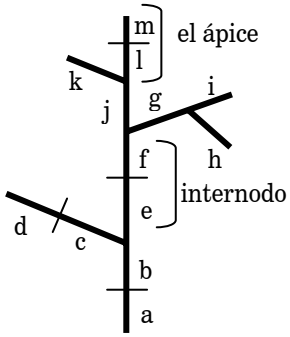


Figura 2.5: Ejemplo de una estructura ramificada

Por ejemplo, el árbol axial de la figura 2.5 se representa mediante la cadena:

$$\omega = x_1[\alpha_1]x_2\ [\alpha_2]\ x_3[\alpha_3]\ x_4$$
$$\omega = ab\,[cd]\,ef\,[g[h]i]\,j\ \ [k]\ \ lm$$

La subpalabra *abefjlm* es el eje principal; $x_1=ab$, $x_2=ef$ y $x_3=j$, son los internodos, y $x_4=lm$ es el ápice. Las subpalabras $\alpha_1=cd$, $\alpha_2= g[h]i$ y $\alpha_3=k$ son las ramas laterales. α_1 y α_3 tienen sólo ápices mientras que α_2 tiene un internodo g, y un ápice i, y una rama lateral h de segundo orden.

Generalmente es muy útil caracterizar los módulos de la planta (y la relación entre ellos) de forma más detallada que utilizando una sola etiqueta. Esto se puede realizar asociando uno o más parámetros numéricos a los símbolos. Un símbolo con parámetros asociados se denomina letra paramétrica, y una cadena de letras paramétricas se denomina palabra paramétrica.

2.2.3 Tortuga intérprete

Las cadenas con corchetes permiten representar la topología de estructuras ramificadas. Para obtener la forma que viene dada por propiedades como la orientación de las ramas y la longitud de los internodos, se debe realizar una interpretación gráfica de las cadenas. Para ello es necesaria la incorporación de información geométrica en la cadena. Una serie de símbolos, con o sin parámetros, se reservan para determinar las propiedades geométricas de la estructura representada. La interpretación mediante tortuga, introducida por Szilard y Quinton [Szil79], y extendida por Prusinkiewicz [Prus86,Prus87] es representativa de esta aproximación.

La cadena se analiza de izquierda a derecha, y cada símbolo se interpreta como una orden para maniobrar una tortuga similar a la utilizada en el lenguaje LOGO [Abel82], pero en tres dimensiones. La tortuga se define mediante su estado que consiste en una posición y una orientación que se define mediante tres vectores denominados *H, L* y *U* indicando la dirección de avance (Heading), la dirección a la izquierda (Left) y la dirección arriba (Up) respectivamente. Estos tres vectores son unitarios, perpendiculares entre si y satisfacen la igualdad $HxL=U$. La orientación de la tortuga puede variarse mediante operaciones de giro sobre cada uno de estos vectores, tal y como se muestra en la figura 2.6 y de acuerdo con la ecuación:

$$[\,H'\ L'\ U'] = [H\ L\ U] \times R$$

donde R es una matriz de rotación 3x3.

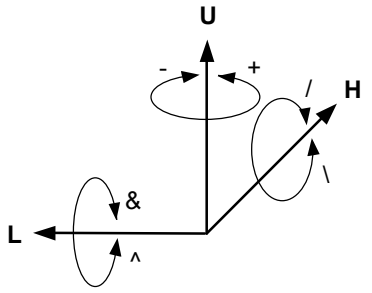


Figura 2.6: Símbolos de control de la tortuga tridimensional

La tortuga inicialmente está situada en el origen de coordenadas, con el vector H apuntando hacia la dirección positiva del eje Y y el vector L apuntando hacia la dirección negativa del eje X. Las órdenes del sistema asociadas al movimiento de la tortuga son las siguientes:

Símbolos de movimiento y dibujo:

F(d),G(d)	Mover la tortuga hacia delante una distancia d > 0. La posición de la tortuga cambia a (x',y',z'), donde $x' = x + d \cdot Hx$, $y' = y + d \cdot Hy$, $z' = z + d \cdot Hz$. Se dibuja un segmento de línea entre los puntos (x,y,z) y (x',y',z').
f(d),g(d)	Mover la tortuga hacia delante una distancia d sin dibujar una línea.
!(a)	Fijar ancho de la rama

Símbolos que controlan la orientación de la tortuga:

+ (α)	Girar a la izquierda un ángulo α según $R_U(\alpha)$
- (α)	Girar a la derecha un ángulo α según $R_U(\alpha)$
& (α)	Subir un ángulo α según $R_L(\alpha)$
^ (α)	Bajar un ángulo α según $R_L(\alpha)$
/ (α)	Rodar a la izquierda un ángulo α según $R_H(\alpha)$
\ (α)	Rodar a la derecha un ángulo α según $R_H(\alpha)$
|	Volverse de espaldas según $R_U(180º)$

Determinadas operaciones asociadas a la generación de una especie vegetal, como por ejemplo finalizar una rama y volver al punto inicial para iniciar una nueva rama, requieren la incorporación al esquema de tortuga de una pila en la que almacenar y recuperar determinados estados de interés de la tortuga. Las órdenes asociadas al manejo de la pila son:

[	Almacenar el estado actual (Push)
]	Recuperar el estado de cabeza de la pila (Pop)

2.3 Modelado mediante sistemas L paramétricos

El presente apartado se centrará en los sistemas L paramétricos, ya que estos están especialmente indicados en aplicaciones de modelado. Estos sistemas son una extensión del concepto de reescritura paralela de cadenas de símbolos a cadenas de palabras. Los primeros modelos que utilizan parámetros son los de Baker y Herman[Bake72], y Lindenmayer [Lind74]. El modelo que se va a seguir en la presente tesis es el desarrollado posteriormente por Prusinkiewicz y Hanan [Prus90, Prus92].

Los sistemas L paramétricos operan sobre *cadenas paramétricas*, definidas como una lista de *módulos* formados por *símbolos* con una serie de *parámetros* asociados. Los símbolos pertenecen al *alfabeto* V y los parámetros pertenecen al conjunto de números reales $\Re$. Un módulo cuyo símbolo sea $A \in V$ y cuyos parámetros sean $a_1,a_2,...,a_n \in \Re$, se denota por $A(a_1,a_2,...,a_n)$. Cada módulo pertenece al conjunto $M = V x \Re^*$, donde $\Re^*$ representa el conjunto de todas las secuencias finitas de parámetros. El conjunto de todas las cadenas de módulos se representa por $M^* = (V x \Re^*)^*$ y el conjunto de las cadenas no vacías por $M^+ = (V x \Re^*)^+$.

Los *parámetros actuales* con valores reales que aparecen en las cadenas son contrastados con los *parámetros formales* usados en la especificación de las producciones del sistema. Estos parámetros formales pueden formar parte de *expresiones* aritméticas y lógicas. Si Σ es un conjunto de parámetros formales, se dice que $C(\Sigma)$ es una expresión lógica que contiene ele-

mentos de Σ y E(Σ) es una expresión aritmética con elementos del mismo conjunto. Ambas expresiones están formadas por parámetros formales y constantes numéricas, combinadas utilizando los operadores aritméticos +, -, *, /, el operador resto %, el operador de exponenciación ^, los operadores relacionales =, <, >, <=, >=, <>, los operadores lógicos &,|,! (*y*, *o* y *no*) y paréntesis. En la construcción de estas expresiones se siguen las reglas sintácticas habituales con la adecuada precedencia de operadores. Los conjuntos de todas las expresiones lógicas y aritméticas formadas por elementos de Σ se denotan por $\mathcal{C}(\Sigma)$ y $\mathcal{E}(\Sigma)$ respectivamente. La expresión lógica vacía ε se supone incluida en $\mathcal{C}(\Sigma)$ y se evalúa siempre a Cierto.

2.3.1 Sistemas de contexto libre

Se define un *sistema-0L paramétrico* como una cuádrupla ordenada $G=(V, \Sigma, \omega, P)$, donde:

- V es el *alfabeto* del sistema.
- Σ es el conjunto de *parámetros formales*.
- $\omega \in M^+$ es una cadena no vacía llamada *axioma*.
- $P \subset (V x \Sigma^*) x \mathcal{C}(\Sigma) x (V x \mathcal{E}(\Sigma)^*)^*$ es un conjunto finito de *producciones*.

Los símbolos : y $\rightarrow$ se utilizan para separar las tres componentes de una producción: el *predecesor*, la *condición* y el *sucesor*. Por ejemplo, la producción

$$A(s,t) : s > t \rightarrow B(s+1)D(s/2,t-2)$$

tiene como predecesor $A(s,t)$, condición $s>t$, y sucesor $B(s+1)D(s/2,t-2)$. El dígito "0" en el término "sistema 0L" significa que la gramática es de contexto libre, y por lo tanto no se considera el contexto del predecesor.

Una producción *coincide* con un módulo de una cadena paramétrica si se cumplen las siguientes condiciones:

- el símbolo del módulo y el símbolo del predecesor coinciden,
- el número de parámetros actuales del módulo coincide con el número de parámetros formales del predecesor y,
- la condición se evalúa a Cierto sustituyendo los parámetros actuales en los formales de la producción.

Una producción que coincide puede *aplicarse* al módulo, generando la cadena de módulos definida por el sucesor de la producción. Durante la evaluación de las expresiones aritméticas, los parámetros actuales se sustituyen en los formales de acuerdo a su posición. Por ejemplo, la producción citada anteriormente coincide con el módulo $A(8,2)$, ya que el símbolo A es el mismo que en el predecesor de la producción, el módulo tiene dos parámetros actuales y el predecesor dos parámetros formales, y la condición $s > t$ se evalúa a Cierto si $s=8$ y $t=2$. El resultado de aplicar la producción es la cadena paramétrica $B(9)D(4,0)$.

Un módulo x que produce una cadena paramétrica χ como resultado de aplicar una producción en un sistema-L se denota por $x \rightarrow \chi$. Dada una cadena paramétrica $\mu = x_1,x_2,...,x_m$, la cadena $\nu=\chi_1,\chi_2,...,\chi_m$ se *deriva directamente* de μ ($\mu \Rightarrow \nu$) si y solo si $x_i \rightarrow \chi_i$ para $i=1,2,...,m$. Una cadena paramétrica ν es generada por G en una derivación de longitud n si existe una secuencia de cadenas $\mu_0,\mu_1,...,\mu_n$ de forma que $\mu_0=\omega$, $\mu_n=\nu$ y $\mu_0 \Rightarrow \mu_1 \Rightarrow ... \Rightarrow \mu_n$.

A continuación se muestra un ejemplo de sistema-L paramétrico. Al igual que en los sistemas-L tradicionales, se asume que un módulo se reemplaza por él mismo cuando no se encuentra ninguna producción coincidente en el conjunto *P*. La especificación de este sistema de ejemplo, así como las cadenas obtenidas en las primeras derivaciones pueden observarse en la figura 2.7.

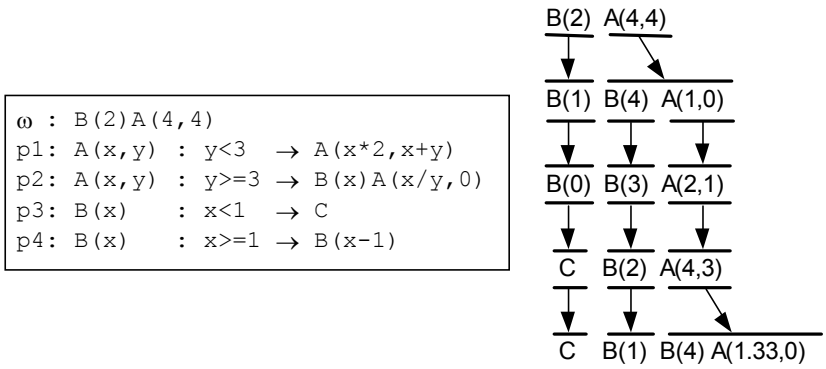


Figura 2.7: Sistema-L paramétrico de ejemplo y primeras derivaciones

No existe ninguna diferencia sustancial entre los sistemas 0L que utilizan corchetes y los que no los utilizan. Sin embargo, debido a la interpretación de los corchetes como delimitadores de las ramificaciones se deben observar las siguientes restricciones:

- pred: cond → succ, donde pred ∈ $(V \times \Sigma^*)$, *cond* ∈ $\mathcal{C}(\Sigma)$, succ ∈ $(V \times \mathbb{E}(\Sigma)^* \cup \{ [,] \})^*$ y succ está bien anidado
- [→ [
-]→]

Estas restricciones reflejan una asunción motivada biológicamente: las ramas solo pueden ser iniciadas por módulos padre individuales.

El siguiente sistema L genera la secuencia de desarrollo de una hoja compuesta:

```
ω : !(1)F(1)
p1: F(s) → G(s)[-!(1)F(s)][+!(1)F(s)]G(s)!(1)F(s)
p2: G(s) → G(2*s)
p3: !(w) → !(3)
```

La estructura se construye sobre dos tipos de módulos los ápices *F* (representados por líneas finas) e internodos *G* (líneas finas). En ambos casos el parámetro *s* determina la longitud de la línea que representa al módulo. Un ápice produce una estructura consistente en dos internodos, dos ápices laterales y una réplica del ápice principal (p1). Un internodo se alarga mediante un factor constante de escalado (p2). La producción p3 se utiliza para que las líneas que representan los internodos aparezcan mas gruesas (3 unidades) que las que representan los ápices (1 unidad). El ángulo de ramificación se fija en *45* grados.

Este ejemplo demuestra que la reescritura paralela, inherente a los sistemas L, captura simultáneamente los cambios que tienen lugar en diferentes partes del organismo. Un paso de derivación corresponde con un intervalo de tiempo. Las diferentes secuencias de desarrollo obtenidas en los consecutivos pasos de derivación se pueden considerar el resultado de la simulación del desarrollo del organismo en intervalos discretos de tiempo, como puede observarse en la figura 2.8

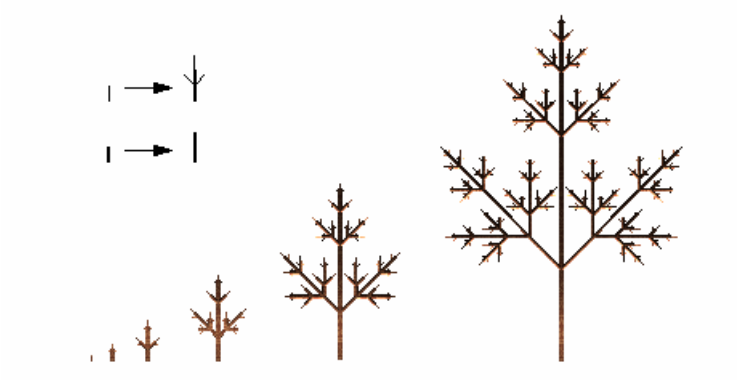


Figura 2.8: Las producciones y la secuencia de desarrollo del modelo de hoja compuesta

Los sistemas L paramétricos proporcionan un marco matemático de trabajo interesante para la exploración de un rango de formas que se pueden obtener de un mismo modelo estructural mediante la modificación de atributos. Mediante la exploración del espacio paramétrico se pueden obtener una gran cantidad de formas producidas incluso por los modelos más simples. Por ejemplo en la figura 2.9 se pueden observar nueve formas distintas obtenidas por el siguiente sistema D0L:

```
ω :A(100,w0)
p1:A(s,w): s>=min → !(w)F(s)
     [+(α1)/(ϕ1)A(s*r1,w*q^e)]
     [+(α2)/(ϕ2)A(s*r2,w*(1-q)^e)]
```

Imagen	r1	r2	α1	α2	ϕ1	ϕ2	w0	q	e	min	n
a	.75	.77	35	-35	0	0	30	.50	.40	0.0	10
b	.65	.71	27	-68	0	0	20	.53	.50	1.7	12
c	.50	.85	25	-15	180	0	20	.45	.50	0.5	9
d	.60	.85	25	-15	180	180	20	.45	.50	0.0	10
e	.58	.83	30	15	0	180	20	.40	.50	1.0	11
f	.92	.37	0	60	180	0	2	.50	.00	0.5	15
g	.80	.80	30	-30	137	137	30	.50	.50	0.0	10
h	.95	.75	5	-30	-90	90	40	.60	.45	25.0	12
i	.55	.95	-5	30	137	137	5	.40	.00	5.0	12

Tabla 2.1: Valores de los parámetros de los modelos que se muestran en la figura 2.9

La producción p1 reemplaza el ápice *A* por un internodo *F* y dos nuevos ápices *A*. Los valores de los ángulos determinan la orientación de estos ápices con respecto al nodo anterior. Los parámetros *r1* y *r2* determinan el gradual decremento de la longitud de los internodos. Las constantes *w0, q* y *e* controlan el grosor de las ramas. El valor de *q* especifica las diferencias de grosor entre las ramas que se generan en un mismo vértice. Finalmente, la condición previene la creación de ramas con una longitud menor a *min*.

A pesar de su aparente diversidad, las formas generadas con el sistema anterior tienen una estructura común: en cada paso de derivación, cada ápice alcanza un internodo terminado por un par de nuevos ápices. Esta es una forma simple del ramificado subapical, un patrón de desarrollo muy común entre las plantas, en el que las nuevas ramas sólo pueden comenzar cerca de ápices de ejes ya existentes. En consecuencia las ramas más bajas, tendrán más tiempo de desarrollarse que las más altas, resultando una estructura basitónica (mayor

desarrollo en la base que la cima figura 2.10a). En la naturaleza, sin embargo, también se pueden encontrar estructuras mesotónicas y acrotónicas, donde el mayor desarrollo se encuentra en el medio y en la cima respectivamente (figura 2.10b y 2.10c). Estos tipos de estructuras se pueden simular mediante sistemas D0L.

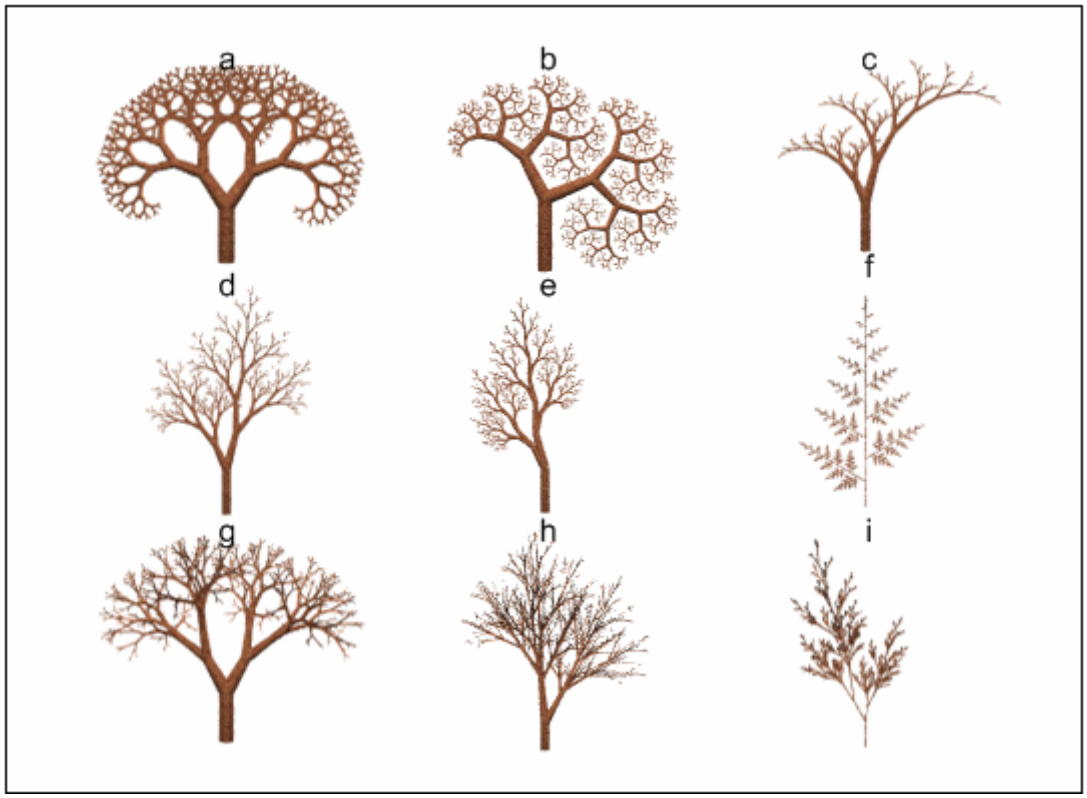


Figura 2.9: Estructuras generadas con el sistema anterior

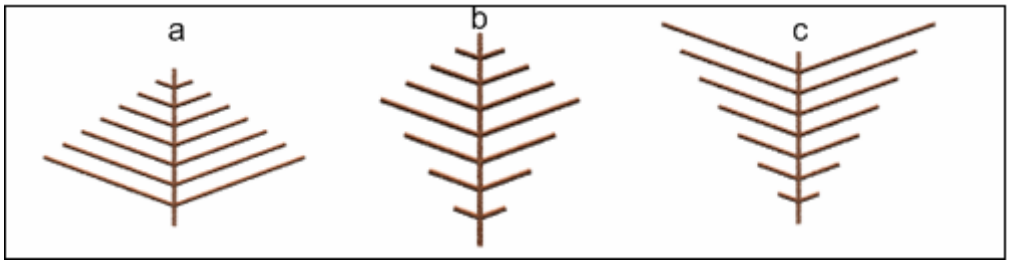


Figura 2.10: Representación esquemática de un patrón de ramificación (a) basitónico (b) mesotónico y (c) acrotónico

2.3.2 Sistemas L paramétricos aleatorios

Hasta ahora se ha visto el desarrollo de una forma como un proceso determinista, de modo que un sistema L siempre generaba la misma cadena. Sin embargo, en el desarrollo de una planta siempre intervienen factores aleatorios. Se pueden distinguir dos casos:

- Los detalles de los mecanismos que controlan el desarrollo no se conocen, y sólo se dispone de datos estadísticos. Estos datos pueden representar: la distribución de la longitud de los internodos, los ángulos de ramificación, probabilidad de ramificación, etc.

- Los mecanismos fisiológicos responsables del control del crecimiento se conocen o se han postulado, pero es conveniente modelar el promedio de los resultados en vez de todos los detalles de operación en modelos construidos en un alto nivel de abstracción.

La simulación de estructuras naturales reales hace necesario introducir variaciones de espécimen a espécimen que mantengan el aspecto general de la estructura a la vez que modifiquen sus detalles. En un sistema estocástico, cada producción tiene una probabilidad de ser aplicada, lo que permite que un mismo símbolo del alfabeto pueda sustituirse por distintas cadenas [Yoko80][Eich80]. Entre los principales trabajos que utilizan sistemas paramétricos estocásticos destacan los realizados por P. Prusinkiewicz sobre el modelado de especies vegetales reales [Prus90] [Prus94]. Aunque los sistemas L estocásticos pueden generar variaciones en la *topología* de la estructura, no permiten las variaciones en los parámetros del sistema que afectan a la *geometría* (longitudes de ramas, ángulos de divergencia, etc.).

Es posible obtener un modelo que permita homogeneizar tanto los cambios en la topología (aplicación estocástica de producciones) como en la geometría (variaciones aleatorias en los parámetros del modelo) mediante una extensión que permita el manejo de variables aleatorias en el sistema. Este modelo denominado Sistemas L Aleatorios o Sistemas Paramétricos Aleatorios fue formulado por [Quir96b] y posteriormente desarrollado para la obtención del sistema de modelado GREEN [Lluc00].

Los Sistemas L Aleatorios (Sistemas RL) [Lluc93][Quir96a][Lluch97][Lluch00] son una extensión de los Sistemas L Paramétricos en los que se incluye la posibilidad de definir un conjunto de variables aleatorias que toman valores en un espacio de probabilidad según una función de distribución determinada [Jone91]. Utilizando expresiones aritméticas y lógicas que combinen los parámetros asociados a los símbolos del alfabeto con las variables aleatorias definidas se consigue que las cadenas generadas por un mismo sistema sean distintas, así como la posibilidad de controlar las variaciones que se produzcan.

Si Σ es el conjunto de parámetros formales del sistema, ϑ es el conjunto de variables aleatorias se dice que $C(\Sigma,\vartheta)$ es una expresión lógica que contiene elementos de Σ y ϑ, y $E(\Sigma,\vartheta)$ es una expresión aritmética con elementos de Σ y ϑ. Ambas expresiones están formadas por parámetros formales, variables aleatorias y constantes numéricas, combinadas utilizando operadores aritméticos, relacionales, lógicos, paréntesis y funciones, evaluándose a cierto la expresión lógica vacía.

Si $C(\Sigma,\vartheta)$ y $E(\Sigma,\vartheta)$ representan a los conjuntos de todas las expresiones lógicas y aritméticas respectivamente, se define un Sistema RL como:

$G<V,\Sigma,\omega,P,\vartheta>$, donde:

- V es el conjunto de símbolos del sistema.
- Σ es el conjunto de parámetros formales.
- $\omega \in M^+$ es una cadena no vacía llamada axioma.
- $P \subset (V \times \Sigma^*) \times C(\Sigma,\vartheta) \times (V \times E(\Sigma,\vartheta) *)^*$ es un conjunto finito de producciones.
- ϑ es el conjunto de variables aleatorias.

Una producción que coincide puede aplicarse al módulo, generando la cadena de módulos definida por el sucesor de la producción. Los parámetros actuales se sustituyen en los formales de acuerdo a su posición. Al realizar esta sustitución se genera un valor real para cada una de las variables aleatorias implicadas en las expresiones correspondientes a los parámetros

actuales. Cada variable se distribuye en un espacio de probabilidad de acuerdo a su función de distribución, mediante la cual se calcula su valor actual.

Por ejemplo, la producción:

```
A(t) : ε → A(t+n)
```

donde ε es la expresión lógica vacía (que se evalúa a cierto) y n una variable aleatoria continua, definida mediante una distribución Uniforme en el rango de extremo inferior -1.0 y extremo superior 1.0, puede aplicarse al módulo A(10.0) para generar el módulo A(10.0+r) donde r es el valor actual de la variable n, comprendido entre (-1.0, 1.0) . El valor final de r depende de la variable n por lo que no se puede determinar a priori, pero se conoce la forma en que se distribuye.

La sintaxis del sistema RL es la siguiente:

```
SYSTEM
DECLARATIONS
  claseVariable nombreVariable=tipoVariable(parámetro[,parámetro...])
  ...
AXIOM
  axioma
RULES
  predecesor : condicionLógica → sucesor
  ...
END
```

donde las palabras en mayúsculas son reservadas.

Existen dos clases de variables aleatorias: Continua y Discreta, dependiendo de su función de distribución. Si la variable aleatoria es Discreta puede ser de los siguientes tipos: Constante, Uniforme, Binomial y Poisson. Si la variable aleatoria es Continua puede ser de los siguientes tipos: Constante, Uniforme, Exponencial y Normal.

Tanto el axioma como el predecesor siguen las mismas reglas que en los sistemas L paramétricos.

La condición lógica estará formada por variables, constantes, parámetros, operadores relaciones, operadores lógicos y expresiones aritméticas, de modo que la producción será aplicada sólo si la condición se evalúa a cierto.

El sucesor consiste en módulos formados por un símbolo del alfabeto al que se le asocian parámetros que se pueden componer de: variables, constantes, parámetros formales o expresiones aritméticas que contengan a todos ellos. El sucesor puede ser la cadena vacía o uno o más módulos.

Los sistemas RL incluyen a los Sistemas L paramétricos estocásticos, de forma que cualquier sistema estocástico puede reescribirse como un sistema RL. Por ejemplo, el sistema estocástico definido por el axioma ω y las producciones p1 y p2:

```
ω A(10.0)
p1 A(t): ε --0.5--> A(t+1.0)
p2 A(t): ε --0.5--> A(t+2.0)
```

puede reescribirse utilizando el sistema RL equivalente que use una variable aleatoria discreta con función de distribución Uniforme con rango de variación entre 0 y 1:

```
SYSTEM
DECLARATIONS
  DISCRETE v = UNIFORM(0,1)
AXIOM
  A(10.0)
RULES
  A(t): v = 0 → A(t+1.0)
  A(t): v = 1 → A(t+2.0)
END
```

Los sistemas RL proporcionan mayor potencia de definición de objetos que los sistemas estocásticos, ya que se puede controlar la forma en que se distribuyen los valores de cada variable aleatoria. Por otra parte, los sistemas RL simplifican la escritura de las producciones del sistema, permitiendo la definición mediante unas pocas producciones de objetos que necesitarían de un número de producciones mucho mayor en un sistema estocástico. A continuación se presenta un ejemplo de un modelo de crecimiento aleatorio de un árbol.

Muchos modelos simples de estructuras ramificadas producen un crecimiento exponencial de las ramas, sin embargo en la realidad se puede observar que este crecimiento exponencial sólo se produce en las fases tempranas del desarrollo del árbol. El modelo que se presenta parte de las siguientes premisas:

- El desarrollo comienza en la estación $k=1$ con la formación de un brote no ramificable
- En las siguientes estaciones de crecimiento, crecen nuevos brotes en las yemas situadas cerca del final de los segmentos generados el último año. Una constante controla el número máximo de brotes que se producirán en la rama madre, $bmax>1$
- Todos los segmentos de rama tienen aproximadamente la misma longitud L y crecen formando una corona semiesférica
- Las hojas se producen en los segmentos terminales (del año actual), formando un nivel semiesférico de hojas cerca del perímetro de la corona. Una constante, σ_{min}, determina el área mínima de hojas que debe ser expuesta a la luz para crear un brote viable.

De acuerdo con estos postulados, el radio de la corona del árbol, después de $k>=1$ estaciones de crecimiento, está limitado por $R_k=L*k$. Una corona semiesférica de este radio tiene una superficie $S_k=2*\Pi*(R_k)^2=2*\Pi*(L*k)^2$, y este valor determina la cota superior del área de la corona que está expuesta directamente a la luz. El número de segmentos de rama N_{k+1} que se añaden al árbol el año $k+1$, está limitado, por un lado, por el número de segmentos creados el año anterior N_k multiplicado por el número máximo de bifurcaciones $bmax$, y por otro lado, por el número máximo de brotes $v_{k+1}= S_{k+1}/\sigma_{min}$ que se podrán producir sin oscurecer excesivamente al resto. Por lo tanto:

$$N_{k+1} = min \{ bmax* N_k , v_{k+1} \} = \{ bmax* N_k, , 2*\Pi*L^2*(k+1)^2/\sigma_{min}\} \qquad (2.1)$$

Se asume que el área mínima de una hoja expuesta a la luz por cada brote, es pequeña en comparación con el área de la corona, $\sigma_{min} << 2*\Pi*L^2$. En un árbol joven (durante las primeras estaciones de crecimiento), el máximo número de nuevos brotes no son suficientes para cubrir la superficie de la corona $bmax*N_k < v_{k+1}$, y por lo tanto el número de nuevos brotes se incrementará de forma exponencial con la edad del árbol: $N_{k+1}=bmax*N_k$. Como el

área de la corona es proporcional sólo al cuadrado de la edad del árbol, a cierta edad t el número potencial de nuevos brotes excederá del número de brotes que se pueden exponer, suficientemente, a la luz directa, es decir $bmax*N_t >= v_t$. Por esto, la ramificación estará limitada por el área de la corona, de modo que el ratio de bifurcación media en la estación k, b_k, siendo $k>=t$, será igual a:

$$b_k = N_{k+1} / N_k = \frac{2*\Pi*L^2*(k+1)^2 / \sigma_{min}}{2*\Pi*L^2*k^2 / \sigma_{min}} = 1 + (2*k+1)/k^2 \qquad (2.2)$$

Existen diferentes patrones de ramificación que pueden satisfacer esta fórmula general. Por ejemplo, si cada segmento del año anterior puede generar uno o dos nuevos brotes, el ratio de segmentos que generan dos brotes será igual a:

$$(N_{k+1} - N_k) / N_k = (2*k+1)/k^2 \qquad (2.3)$$

El siguiente sistema L paramétrico aleatorio cumple esta ecuación:

```
Sistema 2.1
CUniform d 0.0 1.0
ω :FA(1)
  p1:A(k)# d< min{1,(2*k+1)/k2} → /(φ)[+(□)FA(k+1)]-(β)FA(k+1)
  p2:A(k) → /(φ)B-(β)FA(k+1)
```

La generación del árbol comienza con un único internodo F terminado por un ápice A(1). El parámetro del ápice actúa como contador de los pasos de derivación. La producción *p1* crea dos nuevas ramas, mientras que *p2* produce una rama y un nodo dormido *B*, la probabilidad de aplicar *p1* es $p=(2*k+1)/k^2$ y de aplicar *p2* es $q=1-p$. Por lo tanto, hasta la iteración $k=3$, se aplicará *p1*, ya que se asume que es en $k=3$ cuando termina la bifurcación exponencial, a partir de ese momento la probabilidad de bifurcación se determina por la ecuación *(2.3)*. En la figura 2.11 se muestra tres ejemplos de árboles con los ángulos de ramificación $\beta=20^o$, $\alpha=32^o$, $\varphi=90^o$, que generan un modelo de ramificación simpodial (los nuevos brotes no tienen la misma dirección de crecimiento que los precedentes).

Figura 2.11: Ejemplos de árboles generados con el sistema 2.1

2.3.3 Limitación del crecimiento

En los modelos descritos hasta ahora, cada módulo, una vez creado, existe de forma indefinida o genera uno o más hijos, pero nunca desaparece. Los procesos naturales de desarrollo de una planta a menudo comportan la muerte programada de módulos seleccionados. Existen dos formas de simular este fenómeno:

- **Sistemas L sin propagación:** La aproximación original para simular la muerte de un módulo fue el uso de producciones de borrado, utilizando la regla $A \rightarrow \varepsilon$ donde ε representa el módulo vacío.
- **Símbolo de poda:** La tarea de modelado es más complicada cuando una planta elimina una estructura completa, como por ejemplo una rama. Este proceso, denominado abscisión, se puede representar mediante el borrado de todos los símbolos que forman una rama. Para ello se utiliza el símbolo de poda %, que causa la eliminación de todos los símbolos que le siguen que pertenecen a su rama:

$$a\,[\,b\,\%\,[\,c\,d\,]\,e\,[\,\%\,f\,]\,]\,g\,[\,h\,[\,\%\,i\,]\,j\,]\,k \Rightarrow a\,[\,b\,]\,g\,[\,h\,[\,]\,j\,]\,k$$

Con este procedimiento se pueden simular estructuras naturales como por ejemplo palmeras.

2.3.4 Sensibilidad al contexto

La comunicación entre módulos juega un papel crucial en el control del proceso de desarrollo de las plantas. Lindenmayer distinguía dos formas de comunicación:

- por línea de edad, es decir entre padres e hijos
- interacción entre módulos coexistentes

En esta última clase el intercambio de información se puede realizar de formas: endógena (entre módulos adyacentes) o exógena (debida a factores externos). El flujo de agua, hormonas o nutrientes entre módulos son ejemplos de comunicación endógena.

Las producciones de un sistema 0L son de contexto libre, es decir las reglas se aplican sea cual sea el contexto en el que se encuentre el predecesor, y por lo tanto sólo pueden modelar mecanismos de desarrollo controlados por línea de edad. Para poder incluir la interacción endógena de los módulos es necesaria la utilización de sistemas L sensibles al contexto.

En los sistemas L sensibles al contexto la aplicabilidad de una regla depende además del contexto del predecesor, ya que una producción tiene el siguiente formato:

$$ci < pred > cd : cond \rightarrow suc$$

donde *ci, pred* y *cd* son el contexto izquierdo, el predecesor estricto y el contexto derecho. Los únicos elementos obligatorios son del predecesor estricto y el sucesor. Por ejemplo, el siguiente sistema-L se compone del axioma y de tres producciones:

```
SYSTEM
DECLARATIONS
   CONTINUOS p=UNIFORM(0,1)
AXIOM
  A(1)B(3)A(5)
RULES
  A(x): p<=0.4 → A(x+1)
  A(x): p>0.4 → B(x-1)
  A(x) < B(y) > A(z) : y<4 → B(x+z)[A(y)]
END
```

Las dos primeras producciones tienen una aplicación estocástica, de modo que sustituyen el módulo `A(x)` por `A(x+1)o B(x-1)` con una probabilidad de *0.4* y *0.6* respectivamente. La última producción es sensible al contexto y sustituirá el módulo `B(y)` con un contexto izquierdo `A(x)` y

un contexto derecho `A(z)` por `B(x+z)[A(y)]`, siempre que $y<4$. Por lo tanto el primer paso de la derivación podría tener el siguiente resultado:

```
A(1)B(3)A(5) ⇒ A(2)B(6)[A(3)]B(4)
```

En esta derivación se asume que como resultado de la aplicación de una elección aleatoria, la primera producción se ha aplicado al módulo `A(1)` y la segunda al módulo `A(5)`. La tercera producción se ha aplicado al módulo `B(3)`, porque aparece con el contexto requerido y la condición $3 < 4$ es cierta.

En terminología biológica, el contexto izquierdo expresa el flujo de información acropetal, mientras que el contexto derecho describe el flujo basipetal.

Los procesos de desarrollo basados en el flujo endógeno de información que pueden ser se pueden analizar en función de las siguientes características:

- Dirección del flujo de la información: acropetal o basipetal. Pueden haber varias señales propagándose en la misma o distintas direcciones simultáneamente o una detrás de otra.
- Los procesos que tienen lugar en las ramificaciones: Un punto de ramificación se puede ver como un lugar donde la información que llega de diferentes partes se combina, procesa y distribuye a diferentes direcciones.
- Granularidad de la información: La información que se intercambian los módulos puede representar señales discretas (presencia de una hormona) o valores cuantificables (concentración de fotosíntesis).

Las posibles aplicaciones de los sistemas sensibles al contexto es muy amplia, como por ejemplo: Modelos de desarrollo acrotónicos o mesotónicos, desarrollo controlado mediante la distribución de recursos, simulación del ataque de un insecto a una planta, y en general cualquier fenómeno en el que sea necesaria la transmisión de información entre los módulos del modelo. Sin embargo, el modelo no estaría completo si no se tuvieran en cuenta los factores externos a la planta que influyen en su desarrollo. Estos factores se consideran en la próxima sección.

2.3.5 Sensibilidad al entorno

El entorno es un factor clave que afecta a los ciclos de vida de las plantas y grupos de plantas. Consecuentemente, el papel que desempeña el entorno en el desarrollo de una planta es un área importante de estudio. Se debe tener en cuenta este aspecto si se quieren obtener modelos predictivos que se puedan utilizar en aplicaciones que van desde el diseño asistido de jardines y paisajes hasta la determinación de cosechas en la agricultura.

Utilizando el flujo de información entre una planta y su entorno como una forma de clasificación, se pueden distinguir tres formas de interacción:

- La planta se ve afectada por las propiedades globales del entorno, tales como la duración del día controla el inicio de la floración y las temperaturas mínimas y máximas diarias modulan la razón de crecimiento.
- La planta se ve afectada por propiedades locales del entorno, tales como la presencia de obstáculos que controlan el crecimiento de la hierba, y dirigen el crecimiento de las raíces de los árboles, geometría de soporte para enredaderas, la resistencia y temperatura en distintos niveles de la tierra, y geometría predefinida de superficies respecto a las cuales la planta se poda.

- La planta interactúa con el entorno con un ciclo de realimentación de información, donde el entorno afecta a la planta y la planta afecta al entorno recíprocamente. Este tipo de interacción controla el desarrollo de la planta indicando qué partes de la planta influyen en el desarrollo de otras partes de la misma o de otras plantas a través del espacio en el que están creciendo: competencia por el espacio, competencia entre las raíces por los nutrientes y el agua que es transportada en la tierra y competencia por la luz entre los brotes de plantas herbáceas y entre las ramas de los árboles

Inicialmente, los sistemas L se concibieron como sistemas informáticos cerrados, incapaces de simular cualquier forma de comunicación entre la planta modelada y su entorno. El primer paso para la inclusión de factores ambientales se debe a Rozenberg que definió los sistemas L tabulados, que permiten un cambio en el conjunto de las reglas de desarrollo (el conjunto de producciones del sistema L) en respuesta a un cambio en el entorno [Herm75]. Los sistemas L tabulados se aplicaron, por ejemplo, para simular el cambio entre la producción de hojas y la producción de flores por un ápice de una planta debido a un cambio en la duración del día [Frij74]. Los sistemas L paramétricos hacen posible una variante de esta técnica que permite que el entorno afecte al modelo de una forma cuantitativa en vez de cualitativa. Por ejemplo, se puede utilizar un modelo que contenga las temperaturas mínimas y máximas diarias para controlar la razón de crecimiento de las plantas [Hana95].

La interpretación mediante la tortuga del sistema L descrita anteriormente se utiliza para visualizar los modelos como un post-proceso, sin ningún efecto en la operación del sistema. Sin embargo, cuando se tienen en cuenta las condiciones del entorno la posición y orientación de la tortuga es importante para la modelar fenómenos como la detección de colisiones con obstáculos y la exposición a la luz.

La extensión de los sistemas L a la sensibilidad al entorno hace accesible estos parámetros durante el proceso de derivación [Prus94]. La cadena generada se interpreta después de cada paso de derivación y los atributos de la tortuga resultantes de la interpretación se devuelven como parámetros de los módulos reservados de encuesta. Sintácticamente, los módulos de encuesta tienen la forma ?X(x,y,z), donde X= P,H,U o L. Dependiendo del símbolo X, los valores de los parámetros x,y,z representan la posición o uno de los vectores de orientación. Los sistemas L sensibles al entorno se ilustran mediante modelos de escenas de podado. Las funciones de entorno definen formas geométricas, contra las que los árboles se podan. A continuación se presenta un proceso abstracto de desarrollo influenciado por el entorno:

```
ω : A
p1: A → [+B][-B]F?P(x,y)A
p2: B → F?P(X,Y)@OB
p3: ?P(x,y): 4x^2 + (y-10)^2 >10^2 → [+(2y)F][-(2y)F]%
```

El módulo F representa una línea de longitud 1, y los módulos + y – representan giros a izquierda y derecha de 60º. El desarrollo comienza con un módulo A que crea una secuencia de ramas opuestas `[+B][-B]` separadas por internodos F (p1). Las ramas crecen al añadir segmentos F, delimitados por marcas @O (p2). El ápice principal A y los laterales B, crean módulos de encuesta, que devuelven la posición de la tortuga, la cual se utiliza para que cuando un módulo se sitúa fuera de la elipse definida por la ecuación $4x^2 + (y-10)^2 > 10^2$ se crean un par de tentáculos y el resto se poda (p3). El resultado se muestra en la figura 2.12.

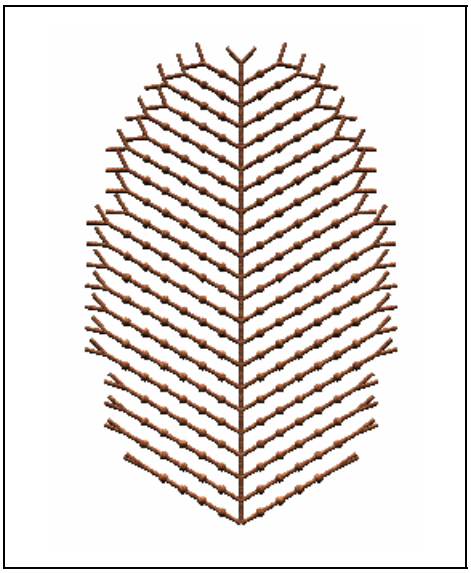


Figura 2.12: Estructura ramificada podada contra una elipse

Una generalización de este concepto son los sistemas L abiertos [Méch96]. Los módulos de comunicación de la forma ?E(x1,...,xm) se utilizan para enviar y recibir información del entorno representada por el valor de los parámetros x1,...,xm. Para este fin, las cadenas resultantes de un paso de derivación se estudian de izquierda a derecha para determinar el estado de la tortuga asociada con cada símbolo. Esta fase es similar a la interpretación gráfica de la cadena, excepto que el resultado no es necesario visualizarlo. Cuando se encuentra un símbolo de comunicación, el proceso crea y envía un mensaje al entorno incluyendo toda o parte de la siguiente información:

- la dirección (posición en la cadena) del módulo de comunicación (es necesario para identificar el módulo cuando se recibe una respuesta del entorno)
- valores de los parámetros xi
- el estado de la tortuga (coordenadas de la posición y del vector orientación, así como algunos otros atributos, tales como la anchura de la línea)
- el tipo y parámetros del módulo siguiente al módulo de comunicación en la cadena

El formato exacto del mensaje se puede definir en un fichero de especificación de la comunicación, compartido entre los procesos que modelan las plantas y el entorno. Por lo tanto, es posible incluir sólo la información necesaria para un modelo particular en los mensajes que se envían al entorno.

Los mensajes de salida del programa de modelado de la planta se transfieren al proceso que simula el entorno utilizando un mecanismo de comunicación entre procesos que proporciona el sistema operativo. El entorno procesa esa información y devuelve los resultados al modelo de la planta utilizando mensajes con el siguiente formato:

- La dirección del módulo de comunicación utilizado.
- Los valores de los parámetros yi que llevan la salida del entorno.

El proceso de la planta utiliza la información recibida para dar valor a los parámetros del módulo de comunicación.

Cuando se han recibido todas las respuestas generadas por el entorno, la cadena resultante se puede interpretar y visualizar, y se puede realizar el siguiente paso de derivación, iniciando un nuevo ciclo de la simulación. Este proceso se muestra en la figura 2.13.

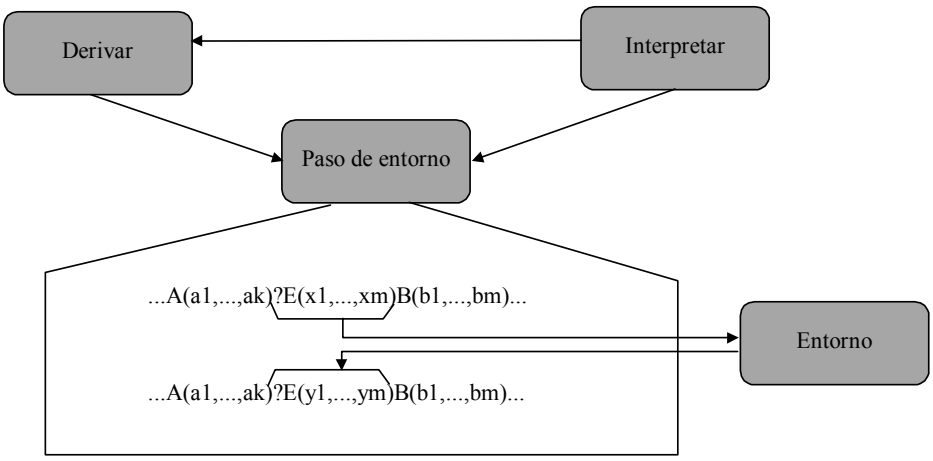


Figura 2.13: Flujo de información durante la simulación de la interacción de una planta con su entorno.

2.4 Visualización interactiva de vegetación

En la actualidad, existe un gran número de aplicaciones donde es necesaria una visualización interactiva de conjuntos de datos muy complejos. Este tipo de aplicaciones se denominan IWA (Interactive walkthrough aplications). Es necesario que el usuario perciba la escena de forma realista y a la vez se debe de cumplir que el número de imágenes generadas por segundo sea suficiente para que no aparezcan efectos no deseados de parpadeo. Estos dos parámetros, realismo y ausencia de parpadeo, como es obvio son difíciles de compatibilizar.

En los sistemas de hoy en día es fácil encontrar escenas con decenas o cientos de millones de polígonos, los cuales no se pueden visualizar de forma interactiva sin parpadeo utilizando las aceleradoras gráficas actuales que son capaces de visualizar hasta 10 millones de polígonos por segundo. Estas tarjetas también son capaces de soportar efectos avanzados de visualización como iluminación, texturación y anti-aliasing en tiempo real. Sin embargo, conforme avanza el hardware gráfico 3D las escenas se hacen más complejas, por ejemplo se pueden tener modelos tan complejos como un Boeing 777 que está formado por 2 millones de partes que incluyen un total de 500 millones de polígonos, si se utilizara un algoritmo de fuerza bruta para visualizarlo sería necesario disponer de una tarjeta que fuera capaz de visualizar 10.000 millones de polígonos por segundo para obtener 20 imágenes por segundo.

Muchas de estas aplicaciones de gráficos en tiempo real y realidad virtual tienen como objetivo la inmersión del usuario en un escenario exterior compuesto por una gran cantidad de fenómenos naturales: paisaje, plantas, árboles, montañas, etc.. En este tipo de aplicaciones el área que puede observar el usuario es muy grande, por lo que es importante que el usuario pueda moverse libremente a lo largo del área sin encontrar zonas artificiales demasiado pronto. El entorno debería de contener mucho detalle (p.ej. las hojas de los árboles) cuando se realice una inspección cercana para obtener un grado de realismo mayor. La edición y visualización de este tipo de escenas son muy complicadas. A continuación se enumeran las principales técnicas desarrolladas por otros autores.

2.4.1 Edición y visualización de poblaciones vegetales

El modelado y visualización de escenas naturales con miles de elementos plantea un buen número de problemas [Prus00] [Hous98]. Se debe modelar el terreno [Abad99], distribuir las plantas de una forma natural, reflejando la interacción de una planta con las demás y con su entorno, se deben sintetizar los modelos geométricos de cada planta, en consonancia con su posición en el interior del ecosistema, para poblar la escena. La escena, que puede estar formada por millones de primitivas, se debe de visualizar de forma eficiente incorporando los efectos de la iluminación en un entorno natural. Por ello es necesario desarrollar un sistema formado por una serie de herramientas que resuelvan estos problemas.

La síntesis realista de imágenes de terrenos cubiertos de vegetación es un problema importante en informática gráfica. Estas escenas incluyen ecosistemas naturales tales como bosques o prados, entornos hechos por el hombre, por ejemplo parques y jardines, y entornos híbridos, tales como tierras reforestadas tras incendios o talas. Los modelos de estos ecosistemas tienen un amplio rango de aplicaciones existentes o potenciales, como por ejemplo la creación de paisajes o jardines asistida por ordenador, la predicción y visualización de los efectos de la tala en el paisaje, la visualización de modelos de ecosistemas para propósitos de investigación o educacionales y la síntesis de escenas para animaciones realizadas por ordenador, simuladores de vuelo o conducción, juegos y arte generado por computador.

En 1985 Reeves y Blau [Reev85] crearon imágenes de bosques y praderas que se utilizaron para la animación realizada por ordenador “The adventures of André and Waly B.”. Reeves y Blau organizan el modelado de la escena como una secuencia de pasos: especificación de un mapa de terreno que proporciona los puntos de elevación de la escena, situación de la vegetación en el terreno de forma procedural o interactiva , modelado de las plantas individuales (hierba y árboles), y visualización de los modelos. Chiba y otros [Chib97] han seguido este esquema general en su trabajo de visualización de bosques, proporcionando la base para las aplicaciones comerciales dedicadas a la síntesis de paisajes [Anim96][Ques97].

La complejidad de la naturaleza hace que esto sea necesario para asignar cuidadosamente los recursos informáticos (tiempo de CPU, memoria y espacio en disco) cuando se desea modelar y visualizar escenas naturales mediante gráficos por ordenador. El tamaño de la base de datos que almacena la escena es un elemento particularmente crítico, ya que el tamaño de los datos geométricos necesarios para representar la escena en detalle es mayor de lo que se puede representar en los ordenadores actuales. Si se analizan las soluciones al problema existentes actualmente, se puede concluir que es importante la búsqueda de un buen equilibrio entre el realismo de las imágenes y la cantidad de recursos necesarios para generarlas.

Las escenas sintetizadas por Reeves y Blau se obtuvieron utilizando sistemas de partículas estructurados, del orden de un millón de partículas por árbol [Reev85]. Para manejar grandes números de primitivas que forman las escenas, los modelos de partículas de cada árbol se generaron de forma procedural y se visualizaron secuencialmente, cada modelo se descargaba de memoria una vez se había visualizado el árbol que representaba. Por esta razón, el tamaño de la memoria necesaria para generar la escena era proporcional al número de partículas de cada árbol, en vez del número total de partículas de la escena. Esta solución requería cálculos de sombreado aproximados, ya que no estaba disponible la información detallada acerca de los árboles cercanos al que se estaba visualizando. El sombreado aproximado también reducía los cálculos del proceso de visualización de la escena.

Otra solución que permite controlar el tamaño de la representación de la escena consiste en reducir los detalles visualmente poco importantes. Se ha realizado un intenso trabajo de investigación de los métodos generales que permiten esta reducción [Hopp97], pero los

resultados obtenidos no se pueden aplicar de forma sencilla a estructuras muy ramificadas como las plantas. Por ello, Weber y Penn [Webe95] plantean una representación multirresolución heurística específica para árboles, que permite la reducción de las primitivas geométricas de los modelos que ocupan una pequeña porción de la pantalla. Marshall y otros [Mars97] también plantean una representación multirresolución de escenas botánicas integrando representaciones poligonales de objetos grandes con aproximaciones de tetraedros de las partes menos representativas de la escena.

Gardner [Gard84] propuso una estrategia diferente para crear escenas naturales visualmente complejas. En este caso el terreno y los árboles se modelaron utilizando un número relativamente pequeño de primitivas geométricas (superficies cuadráticas). La complejidad visual de la escena viene dada por la utilización de texturas procedurales que controlan el color y la transparencia de las copas de los árboles. En una solución similar, los árboles y las plantas se representan mediante mapas de texturas sobre polígonos planos (por ejemplo [Ques97]). Esta solución produce artefactos visibles cuando se cambia la posición del punto de vista. Max [Max95] propuso una solución basada en la imagen más precisa, ya que desarrolló un algoritmo para interpolar entre distintas vistas precalculadas del árbol. En [Max96] se presenta una versión multirresolución de este método que aprovecha la estructura jerárquica de los árboles modelados. Shade y otros [Shad96] presentan un sistema híbrido para navegar en el interior de la escena que utiliza una combinación de geometría y polígonos con textura.

Kajiya y Kay [Kaji89] presentan las texturas volumétricas como un paradigma alternativo para solucionar las limitaciones de los polígonos texturados con mapas de texturas. Neyret [Neyr95][Neyr96] presenta un método para generar terrenos utilizando texturas volumétricas que representan hierba y árboles. Chiba y otros [Chib97] eliminan las deformaciones de las plantas causadas por las curvaturas del terreno permitiendo que las texelas intersecten.

El uso de texturas volumétricas limita el espacio de memoria necesario para representar la escena, porque el mismo texel se puede aplicar en distintas áreas. Esta misma idea se utiliza en la solución más antigua para simplificar escenas visualmente complejas, la instanciación de objetos [Suth63]. De acuerdo con el paradigma de la instanciación, un objeto que aparece varias veces en la escena (posiblemente deformado por una transformación afín), se define solamente una vez, y cada una de sus diferentes instancias se especifican mediante una transformación afín del prototipo. Dado que el espacio necesario para almacenar una transformación es pequeño, el espacio necesario para representar la escena dependerá principalmente del número de objetos diferentes del que se disponga, en vez del número de instancias. Las plantas son objetos particularmente apropiadas para la instanciación, porque se pueden encontrar ocurrencias repetitivas, no sólo en especies de plantas, sino que también en órganos de las plantas y en estructuras ramificadas. Esto permite construir estructuras de datos jerárquicas compactas muy apropiadas para el uso de trazado de rayos rápido, como se plantea en [Kay86][Snyd87].Hart y DeFanti [Hart91][Hart92] extienden el paradigma de la instanciación a estructuras recursivas (autosimiliares). Todas estas referencias contienen ejemplos de escenas botánicas generadas utilizando la instanciación.

La complejidad de las escenas naturales hace que no sólo sean difíciles de visualizar, sino que también difíciles de especificar. Las técnicas de modelado interactivo, disponibles en los paquetes comerciales tales como Alias/Wavefront Studio 8 [Alia96], se centran en la manipulación directa de un número relativamente pequeño de elementos. Por el contrario, un paisaje con plantas puede estar formado por millones de superficies (representando tallos, hojas, flores y frutos) incluidas en complicadas estructuras ramificadas, y posteriormente distribuidas en un ecosistema.

Para poder modelar y visualizar este tipo de escenas Deussen [Deus98] propone el uso de diferentes técnicas como el modelado multinivel que permite modelar el terreno sin tener en cuenta la ubicación de las plantas, y distribuirlas sin tener en cuanta los detalles de cada planta individual. Por otro lado, utiliza un sistema abierto que permite aumentar la complejidad de las escenas.

2.4.2 Técnicas de aceleración de la visualización

En aplicaciones como la realidad virtual o la simulación es necesario que la visualización de la escena se realice de forma interactiva. En todas las aportaciones examinadas en el apartado anterior, la obtención de imágenes se realiza en tiempos que los hace inviables para su utilización en entornos interactivos. Es necesario utilizar técnicas software que permitan manejar escenas complejas y visualizarlas de forma interactiva sin parpadeo utilizando las aceleradoras gráficas actuales.

A continuación se van describir las técnicas más avanzadas que se utilizan los IWA para acelerar la visualización de las escenas. Los frentes donde es posible la mejora para obtener una visualización más rápida son los siguientes:

- Selección por visibilidad (Visibility culling)
- Modelado multirresolución
- Visualización directa de primitivas de mayor orden
- Métodos de visualización basados en la imagen

El principal objetivo de estas técnicas consiste en enviar al hardware gráfico el menor número de primitivas posible, eliminando aquellas partes que observador no puede ver o reduciendo el detalle en aquellos modelos lejanos al observador. Para visualizar de forma interactiva conjuntos grandes de primitivas es necesario utilizar varias de las técnicas que se van a describir a continuación, creando una integración.

Selección por visibilidad
Dado un punto de vista y un campo de visión, de todas las primitivas de la escena sólo una porción es visible. Muchos sistemas eliminan aquellas primitivas que están fuera del volumen de la vista, sin embargo no hacen nada con aquellas que están ocultas por otros objetos. Existen distintas aproximaciones para resolver este problema:

- Dividir la escena en celdas [Lueb95] utilizable sólo en aplicaciones de arquitectura o similares
- Utilizar un Z-buffer jerárquico [Gree93], el hardware no soporta la Z-pirámide
- Selección en el espacio de los objetos para modelos poligonales [Huds97], el conjunto de oclusores se restringe a objetos convexos y no son capaces de hacer selección de partes significativas del modelo
- Mapas de oclusión jerárquicos [Zhan97].

Las características que deben de reunir estos algoritmos son:

- General: que se pueda aplicar a distintos modelos y entornos
- Interactivos: deben de calcularse en una fracción muy pequeña de tiempo
- Selección significativa: deben eliminar porciones significativas del modelo que no son visibles
- Prácticos: deben de poder implementarse en los sistemas actuales y trabajar bien con modelos grandes y reales.

Modelado Multirresolución

Independientemente de cómo se haya generado el modelo poligonal, de la aplicación que lo utilice y del campo en el que se trabaje, almacenar, transmitir y visualizar tal enorme cantidad de datos se hace muy difícil o en algunos casos, imposible. Con el fin de solucionar estos problemas aparece el modelado multirresolución [Heck94]. Este tipo de modelado almacena n aproximaciones distintas del objeto, llamadas niveles de detalle (LOD), siendo n proporcional al tamaño del propio objeto [Pupp97].

Para generar las distintas aproximaciones es necesario el uso de un método de simplificación. Un método de simplificación decide que información es menos relevante y la elimina del modelo, creando una aproximación formada por un menor número de vértices y triángulos. Aplicándolo de forma sucesiva se obtienen las distintas aproximaciones. Los primeros modelos multirresolución, sólo almacenaban un conjunto pequeño de niveles de detalle. Sin embargo, esta solución presenta el problema de la transición entre dos niveles que produce un perceptible efecto de salto en la imagen. Por este motivo, los modelos multirresolución han pasado a albergar n niveles de detalle, donde entre dos representaciones consecutivas la diferencia suele ser de un vértice, una arista o un triángulo.

En los últimos años se ha llevado a cabo una extensa investigación en modelos multirresolución. Algunos de estos modelos están limitados a algún tipo de superficie concreta (paramétrica, 2-D manifold, orientable, etc.) y a veces están concebidos para algún tipo de aplicación concreta (transmisión, realidad virtual, etc.).

Se define el concepto de modelo multirresolución como aquel que permite la representación y manipulación de entidades geométricas con diferentes niveles de detalle [Andú98]. De esta forma, un modelo multirresolución debe establecer las estructuras de datos necesarias para almacenar la información (tanto geométrica como atributos escalares o discretos), y los algoritmos de acceso a dicha estructura de datos para recuperarla de la forma más eficiente posible.

En general, las propiedades que un modelo multirresolución deberá cumplir se resumen en las siguientes:

- El incremento en el tamaño del modelo no sea significativo cuando se aumenta el número de niveles de detalle.
- La extracción de un LOD se realice de forma eficiente, es decir, si el modelo contiene n niveles de detalle la información debe estar organizada de manera que el algoritmo de extracción pueda recuperar los datos en el menor tiempo posible a fin de permitir una visualización interactiva.
- La malla extraída sea continua, entendiendo por continua que no aparezca ningún tipo de rotura o agujero para todas las posibles representaciones.
- La transición de un nivel a otro se realice de forma suave, es decir, al pasar de un nivel de detalle al siguiente (ya sea para incrementar o decrementar detalle) no ha de percibirse ningún tipo de salto en la imagen.
- No haya pérdida de información, es decir, el modelo multirresolución ha de ser capaz de representar con exactitud al modelo original.

Visualización directa de primitivas de mayor orden

Muchas aplicaciones utilizan splines (NURBS) para representar los objetos de la escena. Es muy sencillo teselar estas primitivas para visualizarlas, pero esta técnica tiene un problema ya que las mallas poligonales pueden ser muy burdas si se miran de cerca o pueden hacerse muy

grandes con el consiguiente incremento de cálculo. Existen técnicas para mejorar la visualización de este tipo de primitivas, que permiten la teselación en tiempo real visualizando hasta 1000 superficies splines de forma interactiva [Abi94]. Kumar et al han desarrollado otros algoritmos que utilizan técnicas como la coherencia cuadro a cuadro, selección de visibilidad y teselación incremental de los parches en triángulos para incrementar la eficiencia [Kuma97].

Representaciones basadas en la imagen (Image based rendering)
Un método alternativo de simplificación consiste en reemplazar la geometría por imágenes utilizando: texturas, warping 3D, campos de luz, lumigraphs o LDI (Layered Depth Images). En este caso, las imágenes se pueden visualizar de forma que la ratio sea dependiente del tamaño pero independiente de la complejidad de la escena:

- reemplazar ciertos objetos (tales como objetos lejanos) por polígonos texturados denominados impostores [Shad96]
- realizando algoritmos de visualización que utilizan imágenes en vez de geometría como primitivas básicas [Gort96]

Otra técnica muy reciente [Meye01] permite la visualización de árboles con sombras de forma consistente con las condiciones de iluminación. Para ello utiliza una jerarquía de texturas bidireccionales similar a un campo de luz 6D. También calcula información de visibilidad para realizar más eficientemente el cálculo de las sombras.

Algunas de las técnicas de aceleración comentadas, no se pueden aplicar a escenas formadas por árboles, debido a la complejidad de la topología de los objetos a visualizar. Por ello en el siguiente apartado se presentan las técnicas más recientes que se aplican específicamente a la aceleración de la visualización de árboles y plantas.

2.4.3 Iluminación de escenas en el exterior

Además de la edición y la visibilidad, otro aspecto importante en la visualización de escenas naturales es la iluminación. La simulación de la interacción de las plantas con la luz permite determinar no sólo los atributos de apariencia, como el color, el brillo o la transparencia, sino también simular su crecimiento e interacción con el entorno. Visto desde esta perspectiva, la obtención de imágenes realistas con vegetación en el exterior implica la utilización de modelos de iluminación que incorporen fuentes de luz naturales (fundamentalmente el cielo y el sol) y que tengan en cuenta las características biológicas del tejido de las plantas, todo ello sin perder de vista la elevada complejidad de este tipo de escenas que hace que los algoritmos clásicos de cálculo de iluminación global sean difícilmente aplicables.

Hasta el momento, la mayor parte de los algoritmos utilizados para la iluminación de vegetación, y en general de escenas complejas, tienen como objetivo la eficiencia, como es el caso del método de Max [Max96] que propone el uso de imágenes pre-calculadas que posteriormente se combinan para obtener nuevas imágenes desde distintos puntos de vista. Para las hojas utiliza polígonos texturizados que se obtienen a partir de hojas reales escaneadas. Posteriormente el mismo Max junto con otros investigadores [Max97] presentaron un método de transporte de radiancia paralela al plano para vegetación densa, no aplicable a árboles aislados, en el que asumen que la radiancia depende angularmente de la dirección del flujo de luz y posicionalmente únicamente de la altura respecto del suelo, lo que simplifica el cálculo del transporte de radiancia.

Paralelamente C. Traxler y M. Gervautz [Trax97] utilizaron trazado de rayos para iluminar escenas con vegetación modeladas mediante sistemas paramétricos de reescritura y representadas mediante grafos cíclicos, pero no incorporan fuentes de luz naturales.

Daubert y otros [Daub97] trataron de integrar todos los métodos que ayudan en los cálculos de radiosidad para escenas en el exterior con mapas de terreno, plantas y edificios, en un algoritmo jerárquico que incorpora el cielo y el sol como elementos de la jerarquía. La cúpula celeste se representa como un a semiesfera dividida en cuadriláteros, considerando cada uno de ellos como una fuente de luz en el infinito, mientras que el sol se modela como una fuente de luz paralela.

Chelle y otros presentaron en [Chel98] un estudio relacionado con la radiosidad en escenas con vegetación donde distinguen entre radiosidad lejana y cercana dependiendo de la proximidad de los polígonos entre si. El cielo y el sol se consideran fuentes de luz situadas en el infinito. El cielo se descompone en un conjunto de fuentes direccionales. El método, llamado Radiosidad Anidada, es adecuado para escenas con una distribución homogénea de polígonos.

Como hemos visto hasta el momento, las técnicas para la creación realista de escenas con vegetación han tenido como objetivo principal resolver el problema de la complejidad y la incorporación de fuentes de luz naturales. La incorporación del cielo y el sol como fuentes de luz lleva al desarrollo y aplicación de modelos de cielo como el presentado por Perez, Seals y Michalski [Pere93], que permite el cálculo de la radiancia espectral del cielo en diferentes instantes de tiempo y en diferentes puntos de la Tierra y al método propuesto posteriormente por Preetham y otros [Pree99]. Preetham utiliza la fórmula de Perez incorporando parámetros que permiten modelar el efecto de la atmósfera.

La iluminación (natural o artificial) realista de las plantas, y en concreto de las hojas, no se puede conseguir aplicando para su visualización malas aproximaciones a la naturaleza del tejido foliar. Sin embargo, hace poco que el interés de los investigadores se centra en la interacción de la luz con las plantas. Recientemente Baranoski y Rokne [Bara01] presentaron una aproximación para simular la dispersión de la luz en las hojas, consistente en pre-calcular valores de reflexión y de transmisión cuyos diseño y formulación están basados en métodos de Monte Carlo estándar.

Por último, resaltar que ni los métodos basados en radiosidad ni en trazado de rayos son adecuados para visualizar escenas naturales en el exterior de forma interactiva, donde puede cambiar tanto la posición de observación como las condiciones de iluminación.

2.5 Avances en la navegación interactiva de poblaciones vegetales

Tradicionalmente, se han utilizado técnicas para acelerar la visualización de este tipo de escenas: aceleradoras de hardware gráfico, mapas de texturas, nivel de detalle y selección. Sin embargo, para conseguir manejar estas cantidades enormes de datos y visualizarlas con el mayor realismo posible, también se debe trabajar en tres frentes:

- **Modelado:** Construir un gran número de modelos diferentes puede ser muy costoso. La utilización de mapas de texturas (p.ej.: fotografías) reduce el coste del modelado, pero no son muy realistas cuando el usuario se sitúa cerca de la imagen. La repetición del mismo modelo dentro de la escena se detecta rápidamente y el entorno virtual pierde credibilidad.

- **Requerimientos de almacenamiento:** En este tipo de aplicaciones es necesaria una gran capacidad de almacenamiento, ya que las bases de datos geométricas que se utilizan son enormes. Por lo tanto, es muy importante tener una representación compacta de cada uno de los elementos que forman la escena.
- **Transmisión:** Si la base de datos va a ser distribuida a través de la red, es todavía más importante tener una representación compacta. El rápido crecimiento de las aplicaciones de realidad virtual basadas en Internet, donde los anchos de banda son impredecibles, hacen que sea necesaria una representación compacta que pueda ser transmitida en un tiempo menor.

A continuación se presentan las principales técnicas que otros autores han utilizado para acelerar la visualización de árboles.

2.5.1 Visualización directa de un modelo procedural

Schmalstieg [Schm97] presenta un método que permite la visualización directa de sistemas L, para ello transforma el sistema de reescritura en un grafo cíclico dirigido que no necesita de un modelo geométrico intermedio para ser visualizado. Es decir, el grafo almacena las primitivas gráficas que se derivan de la interpretación del sistema-L. Este tipo de estructura utiliza la memoria de manera eficiente, ya que no es necesario el uso explícito de geometría detallada. Además, la forma compacta en la que se almacenan los datos es muy interesante a la hora de transmitirlos por la red. Los problemas ocasionados cuando se ven de cerca las texturas utilizadas en otras soluciones, se elimina porque al visualizar se utiliza la geometría detallada de las primitivas que forman el modelo, sin unos requerimientos elevados de memoria.

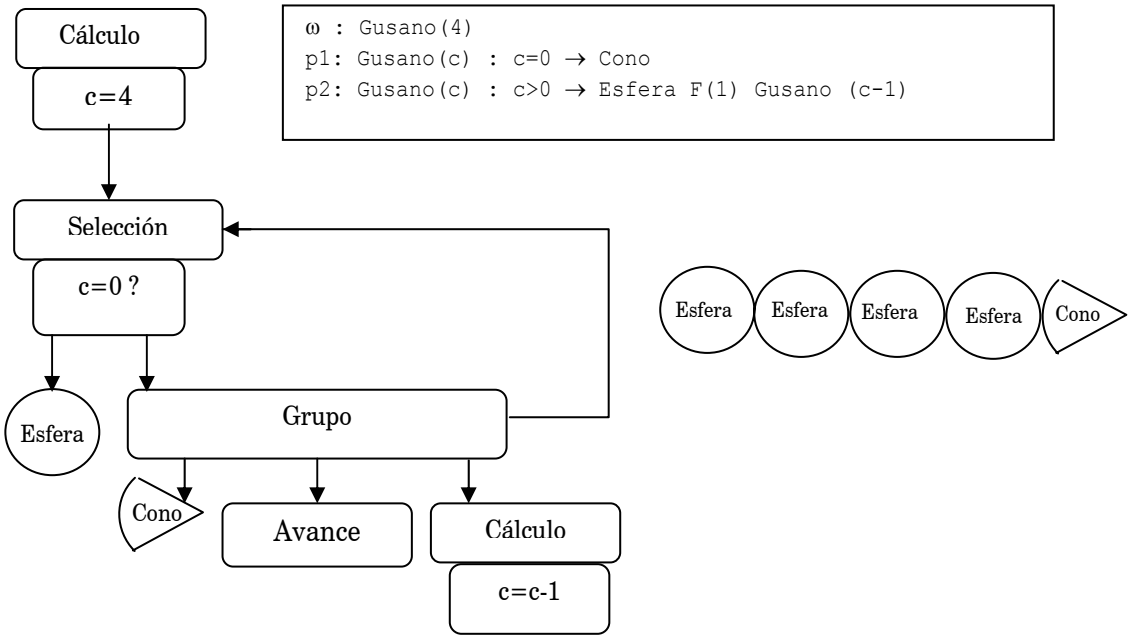
Para realizar la conversión de un sistema L en un grafo cíclico dirigido se deben cumplir las siguientes condiciones:

- Sólo los símbolos terminales tienen significado gráfico
- Sólo los símbolos no terminales pueden ser predecesores de una regla
- Por lo menos hay una regla, para cada no terminal, que deriva sólo en símbolos terminales
- La cadena se podrá visualizar cuando esté formada sólo por símbolos terminales

El grafo se obtiene siguiendo las siguientes reglas:

- El sucesor de cada regla se considera un subgrafo
- Los módulos concatenados se representan como hijos de un nodo de grupo que incluye todos sus hijos
- Para cada módulo no terminal se crea un solo nodo de selección
- Los hijos de un nodo de selección son los subgrafos formados por los sucesores de las reglas de cada módulo
- El nodo de selección del axioma formará la raíz del grafo
- Las expresiones aritméticas pasadas como parámetros a los módulos se transforman en nodos de cálculo.

Por ejemplo si se tiene el siguiente sistema recursivo, se obtiene el grafo:



Una vez se ha generado el grafo, se recorre en profundidad y de izquierda a derecha para visualizar el modelo. Durante el recorrido se deben ir acumulando los estados, es decir las transformaciones se deben ir multiplicando conforme se vayan encontrando. El estado se guardará antes de realizar un recorrido en profundidad y se recuperará una vez se haya recorrido el subgrafo.

Para crear una población de individuos diferentes de la misma especie, se puede utilizar un único modelo. Para ello se deberán de parametrizar las principales características de la especie y aplicar valores aleatorios que varíen en un rango predefinido.

Si se quiere visualizar una gran cantidad de elementos la geometría necesaria será enorme, por lo que será necesaria la utilización de niveles de detalle. Aquí radica el principal problema de este modelo. No se podrían eliminar niveles de recursión en los subgrafos, ya que el resultado que se obtendría sería un individuo en un estado de desarrollo menor y no un nivel de detalle menor. La solución que plantea el modelo consiste en cambiar un subgrafo por una única primitiva del color y de la forma apropiados. Sin embargo, este proceso no es automático y se realiza a mano, lo cual es un proceso muy laborioso.

2.5.2 Texturas volumétricas interactivas

Meyer y Neyret presentan en [Meye98] un método para visualizar de forma interactiva texturas volumétricas [Kaji89, Neyr96], realizando una transformación al Z-buffer de este método inicialmente propuesto para el trazado de rayos. Esta aproximación consiste en cortar una pieza de geometría 3D en una serie de finos niveles. Un nivel es un rectángulo que contiene la visualización de la geometría incluida en ese corte. Estos niveles se utilizan posteriormente como texturas transparentes en el proceso de visualización, que se sitúan sobre la superficie del terreno.

La especificación de un objeto consiste en una malla de triángulos con coordenadas de textura y un vector de altura en los vértices, además de un patrón de textura volumétrica, que está formado por un conjunto de texturas RGBA que representan rodajas horizontales, infinitamente finas, del volumen, como se puede apreciar en la figura 2.14.

Para obtener las texturas se puede utilizar un visualizador estándar en el que se van modificando los planos de recorte delantero y trasero, ajustándolos a las rodajas precalculadas, y situando el punto de vista en la parte superior del patrón. Si el objeto que se utiliza para definir la textura está definido mediante modelado de superficies, será necesario rellenar los huecos que se generan en los contornos calculados.

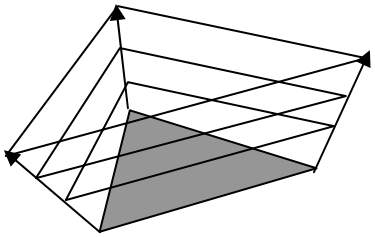


Figura 2.14: Un texel se dibuja utilizando triángulos texturados calculados por extrusión

Un problema importante se presenta cuando el observador se sitúa en una posición en la que las visuales forman un ángulo con los vectores de altura cercano a los 90º. En este caso aparecen agujeros entre las distintas rodajas, lo que conlleva una pérdida total de realismo. La solución que se plantea es la de crear rodajas verticales, además de las horizontales, es decir crear un conjunto de rodajas para cada una de las direcciones principales, multiplicando por tres el número de texturas necesarias.

Sin embargo el principal problema que presenta este método es la uniformidad de los elementos visualizados, ya que todos ellos parten de un mismo patrón, y sólo se modifican si el terreno no es plano. Además, en un terreno muy escarpado los vectores de altura dejan de ser paralelos lo que conlleva una gran deformación de los texeles.

2.5.3 Visualización mediante mezcla de rodajas

Jakulin [Jaku00] presenta una técnica que combina geometría (para el tronco y ramas principales) con visualización basada en la imagen (para la corona). La corona se visualiza utilizando una representación multinivel, de modo que cada nivel está formado por una textura, llamada rodaja como en la aproximación de Meyer. Para cada corona se utilizan varios conjuntos de niveles para facilitar la visualización desde distintos puntos de vista.

La transición entre dos conjuntos de niveles se realiza ajustando la opacidad de las texturas en función del ángulo que forman con la visual, facilitando un cambio suave entre ambas y reduciendo el número de rodajas necesarias.

Un conjunto de rodajas se visualiza, simplemente, dibujando todas las rodajas que lo forman. Las rodajas están situadas en posiciones fijas del espacio, y su orientación es invariante con la posición y orientación del observador, a diferencia de los típicos *billboards*. Las rodajas de un mismo conjunto son paralelas y equidistantes. La solución que se plantea consiste en seis conjuntos de rodajas que forman un ángulo de 60º entre ellas, cada uno formado por 5 rodajas, con lo que se necesitan 30 texturas para visualizar la corona del árbol.

El método expuesto sólo sirve para un observador que se sitúa sobre el terreno, por lo tanto no es posible realizar sobrevuelos sin una pérdida total de realismo, y las soluciones que se plantean son las propuestas en [Meye98]. Otro de los problemas que plantea es el número de texturas necesarias, lo cual puede llevar a un cuello de botella en la memoria de texturas.

Este método trabaja de forma óptima cuando la distancia del observador a los árboles visualizados se sitúa entre los 10 y los 50 metros, donde la discrepancia entre el error producida por el ángulo formado entre las visuales y las texturas es bajo. Si el observador se aproxima a una distancia menor la calidad obtenida no es suficientemente realista.

2.6 Problemática actual y objetivos de la tesis

Tras el estudio de las técnicas existentes en el modelado y visualización interactiva de entornos naturales, es posible determinar las líneas de investigación que hay abiertas en este campo. Teniendo en cuenta estas líneas generales de actuación se pueden marcar los objetivos que se persiguen en la presente tesis y los modelos que se van a seguir para ello.

2.6.1 Líneas de trabajo

Una vez estudiados los métodos que han desarrollado otros autores se pueden extraer las siguientes conclusiones:

- La visualización interactiva de árboles con un cierto nivel de realismo es una tarea compleja y necesaria en gran cantidad de aplicaciones
- Utilizar geometría, únicamente, es inviable con los dispositivos gráficos actuales
- Se debe enviar al hardware gráfico el menor número de primitivas gráficas
- El modelo puede ser visualizado y transmitido en distintos niveles de detalle
- Es necesario obtener métodos que permitan una visualización realista de los árboles desde cualquier posición y orientación del observador
- Es posible encontrar soluciones diferentes para las hojas y para las ramas
- Los modelos utilizados para los elementos naturales de la misma especie deben ser parecidos pero no deben ser el mismo

Por lo tanto, teniendo en cuenta las conclusiones alcanzadas tras el estudio de las técnicas existentes, se puede decir que la solución del problema puede ser abordada desde distintos frentes:

- **General:** Utilizando técnicas de selección para enviar al visualizador sólo aquellos individuos que estén situados dentro del campo de visión del observador y utilizar distintos niveles de detalle para aquellos individuos u otros elementos de la escena que estén más alejados del observador.
- **Sólo hojas:** Utilización de imágenes precalculadas de distintas partes del árbol para disminuir el número de polígonos a visualizar, obtener un modelo multirresolución para extraer distintos niveles de detalle en función de la distancia al observador o para su transmisión incremental.
- **Sólo ramas (polígonos):** La estructura ramificada se puede convertir en una malla poligonal a la que se podrán aplicar técnicas de multirresolución y simplificación.
- **Sólo ramas (cadenas):** Obtener una estructura de datos en el proceso de derivación a la que se le pueda aplicar un modelo multirresolución procedural, que permita la extracción de los niveles de detalle cuando se interpreta la cadena, así como la transmisión de los mismos de forma incremental.
- **Fusión:** Obtener un modelo conjunto que permita la visualización de un conjunto de árboles completos.

2.6.2 Objetivos de la tesis

En función de las conclusiones alcanzadas se establece que para conseguir el objetivo general de la tesis que consiste en acelerar la visualización de una escena formada por múltiples elementos vegetales, siendo posible la inspección cercana de los elementos, sin pérdida de realismo, será necesario alcanzar los siguientes objetivos parciales:

- Obtener una malla poligonal que represente a todas las ramas del árbol
- Obtener un modelo multirresolución procedural, es decir una cadena paramétrica que represente las ramas en distintos niveles de detalle
- Obtener un modelo multirresolución basado en la imagen que represente a las hojas del árbol
- Obtener un modelo que fusione los anteriores
- Transmitir el modelo obtenido de forma incremental

2.6.3 Elección de modelo

Para la consecución de estos objetivos se ha decidido que entre las diferentes líneas de modelado vegetal existentes es, a nuestro juicio, la del modelado basado en sistemas L la que presenta un mayor número de ventajas. Las razones que han llevado a su elección se enumeran a continuación:

- Es el modelo más extendido, ampliamente utilizado y estudiado por distintos grupos de investigación, incluido el nuestro, lo que le ha llevado a un grado de desarrollo muy superior a todos los modelos expuestos anteriormente.
- Se puede modelar la comunicación endógena entre los distintos elementos que forman el árbol, ello permite la simulación de crecimiento, floración, etc. tal y como ocurre en la naturaleza.
- Es posible simular la interacción del individuo con el entorno.
- El conjunto de símbolos que forman la gramática se puede ampliar según las necesidades del modelo, así como su funcionabilidad.
- La interpretación gráfica de los módulos permite una gran flexibilidad a la hora de la visualización del modelo.
- Es posible la obtención de una única malla de polígonos que represente a todas las ramas del árbol.
- El proceso de generación del modelo lleva implícito la posibilidad de crear agrupaciones de ramas y hojas para una posterior aceleración de su visualización
- Es posible la obtención de un modelo multirresolución que represente la estructura de ramas del árbol en distintos niveles de detalle.

Una vez comentados los antecedentes de la investigación y establecidos los objetivos a alcanzar, en los dos próximos capítulos, se va a realizar un análisis de las aplicaciones desarrolladas. Estas aplicaciones han sido el motivo y sirven de base para la realización de la presente tesis

Capítulo 3
GREEN: GeneradoR de Especies para Entornos Naturales

Una vez que se han descrito los objetivos que se pretenden conseguir en la presente tesis, y antes de pasar a explicar como se han conseguido, se va a hacer un estudio de los trabajos previos que se han realizado. Estos trabajos, imprescindibles para conseguir los objetivos planteados, se enmarcan dentro de dos proyectos de investigación financiados:

- "Modelización y Visualización de entornos naturales virtuales. Aplicación a la simulación de la visión robótica de campos frutales", financiado por la Generalitat Valenciana. GV97-TI-05-42. Periodo: 1998/99
- "Arquitecturas multirresolutivas de sistemas gráficos procedurales", financiado por la CYCIT. TIC99-0510-C02-01. Periodo: 2000/02

La herramienta desarrollada en el primer proyecto reseñado se conoce como GREEN: GeneradoR de Especies para Entornos Naturales, y consiste en un entorno de edición y visualización interactiva de árboles y plantas mediante sistemas L, en el presente apartado se van a describir sus principales características. En el segundo proyecto se ha desarrollado una herramienta para la visualización de poblaciones vegetales cuyos individuos han sido editados mediante GREEN. El nombre de esta aplicación es EVERGREEN: Entorno de Visualización de Escenas en el exterioR basado en GREEN. En el capítulo cuatro se explicarán en detalle los principales aspectos de la misma, para pasar en el capítulo cinco a desarrollar las técnicas multirresolución que se han planteado como objetivos de la tesis y que son necesarias para conseguir una navegación interactiva en una escena natural.

3.1 Características generales

Como primer paso en la realización de esta aplicación, se ha diseñado un prototipo en lenguaje smalltalk, el cual permite la derivación e interpretación de un sistema RL. Como resultado de la interpretación se obtiene un fichero en formato INVENTOR que posteriormente puede ser visualizado mediante una aplicación externa.

El sistema diseñado dispone de una estructura de clases cuya clase principal es el propio GREEN. Además de las clases necesarias para el modelado mediante sistemas-RL, como pueden ser las reglas de producción, las variables aleatorias o los módulos que permiten controlar la tortuga, se han incorporado módulos cuya funcionalidad permite derivar una misma cadena de diferentes maneras, lo que hace del GREEN una herramienta muy potente para obtener individuos diferentes partiendo de una misma gramática.

Con este sistema también es posible modelar diferentes características de los árboles o plantas (de las estructuras arborescentes en general), como son la sensibilidad al contexto, el tropismo o la poda.

En la Tabla 3.1 se especifican algunas de las clases más relevantes del Green junto con sus variables de instancia.

Clase	Variables de instancia		
Cadena	módulos		
Derivador	cadena	contexto	sistema
Función	bloqueada	nombre	último
Intérprete	tortuga	cadena	
Módulo	nombre	parámetros	
Regla	condición	predecesor	sucesor
SistemaRL	axioma	funciones	reglas
Tortuga	pila	posición	vectorH vectorL vectorU

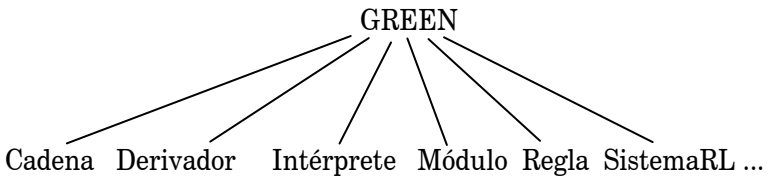


Tabla 3.1: Clases y variables de instancia principales del sistema Green

Los diferentes módulos implementados en este sistema se podrían clasificar dependiendo de si actúan sobre el estado de la tortuga (como giros o avance), o si son módulos que definen la geometría de las diferentes partes del objeto (como cilindro o esfera) o si son módulos que permiten controlar la propia derivación. Entre estos últimos se encuentran los módulos *Switch*, *Péndulo*, *Repetición* o *Poda*.

- El módulo *Switch* utiliza una lista de módulos como parámetros. En cada derivación se deriva el siguiente módulo en la lista. Cuando se llega al final de la lista se vuelve al principio.
- El módulo *Péndulo* permite que una cadena de módulos derive en sentido directo o inverso cada vez que deriva. En cada derivación cambia el sentido.
- El módulo *Repetición* permite derivar una cadena la cantidad de veces que indique un parámetro.
- Por último, el módulo *Poda* elimina de la cadena los módulos que están detrás de él y pertenecen a la misma rama.

El prototipo tiene dos problemas principales, el primero es el tiempo de respuesta, ya que puede tardar muchos minutos en derivar una gramática. Si el número de iteraciones que se solicita es muy grande, el tiempo de respuesta se transforma en horas, lo que hace inviable la edición de ciertos sistemas. El segundo problema reside en la necesidad de utilizar una aplicación externa para poder visualizar los objetos modelados.

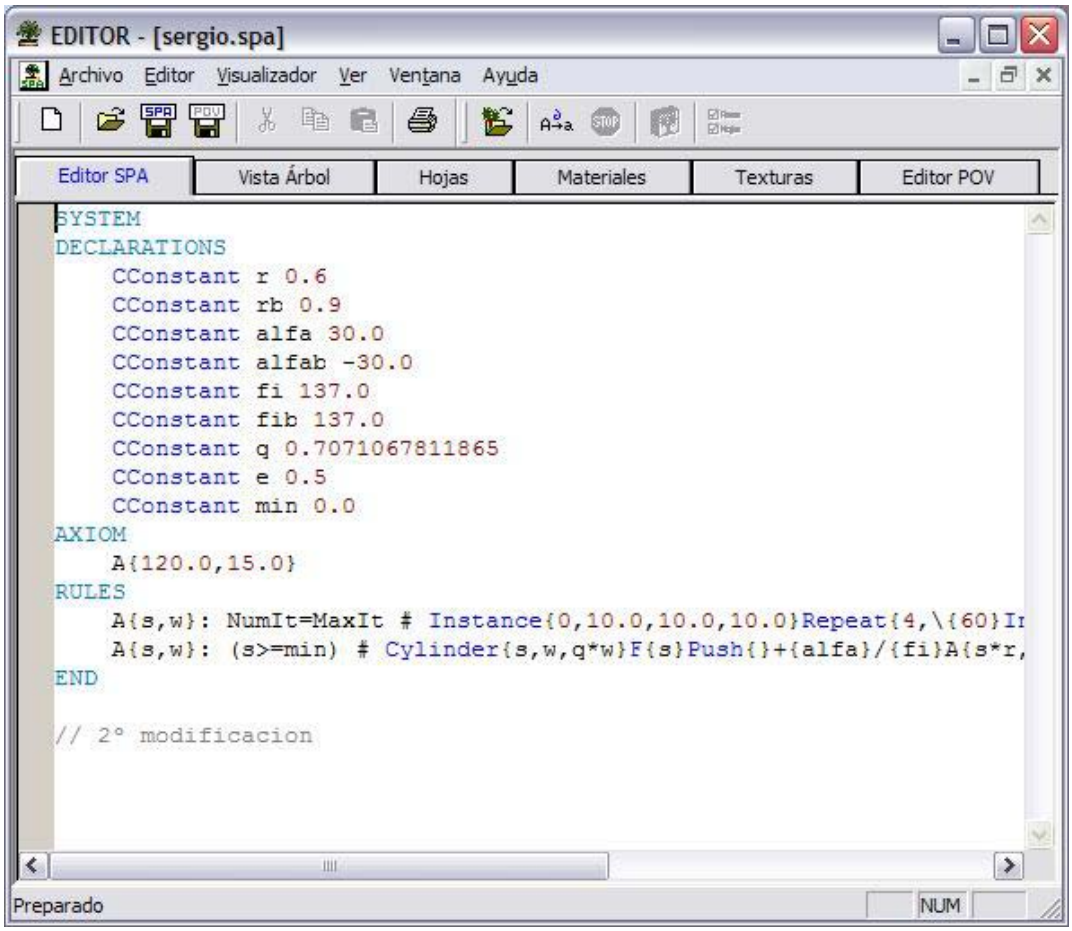


Figura 3.1: Vista general del sistema GREEN

Por lo tanto, una vez el prototipo ha cumplido su objetivo de diseño de las principales funcionalidades de derivación e interpretación de sistemas, se pasa a la fase de implementación de la aplicación. Se decide realizar un sistema compacto con una interfaz gráfica que incluya las siguientes opciones: editor de texto para definir el sistema a interpretar, visualizador interactivo del modelo editado, selector de material y de textura para las hojas y las ramas, y visualización realista mediante POVRay.

El desarrollo de la aplicación se ha realizado utilizando el entorno Visual C++ y las librerías MFC para el interfaz y OpenGL para el visualizador interactivo. En la figura 3.1 se puede apreciar una vista general de la aplicación con las distintas opciones que la componen. En la figura 3.2 se muestra el esquema del funcionamiento general de la aplicación.

3.2 Estructura y especificación de un sistema RL

Un sistemaRL se especifica en un fichero de texto que tiene las siguientes partes, cada una de ellas encabezada por una palabra clave escrita en mayúsculas:

- Cabecera: se especifica por la palabra clave SYSTEM
- Declaración de funciones y variables aleatorias: comienza con la palabra clave DECLARATIONS, en cada línea se especificará una función o variable aleatoria y sus parámetros correspondientes
- Axioma: la declaración del axioma comienza con la palabra clave AXIOM

- Reglas: tras la palabra clave RULES se especificarán todas las reglas o producciones del sistema, una en cada línea.
- Cierre: el final del sistema se indica con la palabra clave END

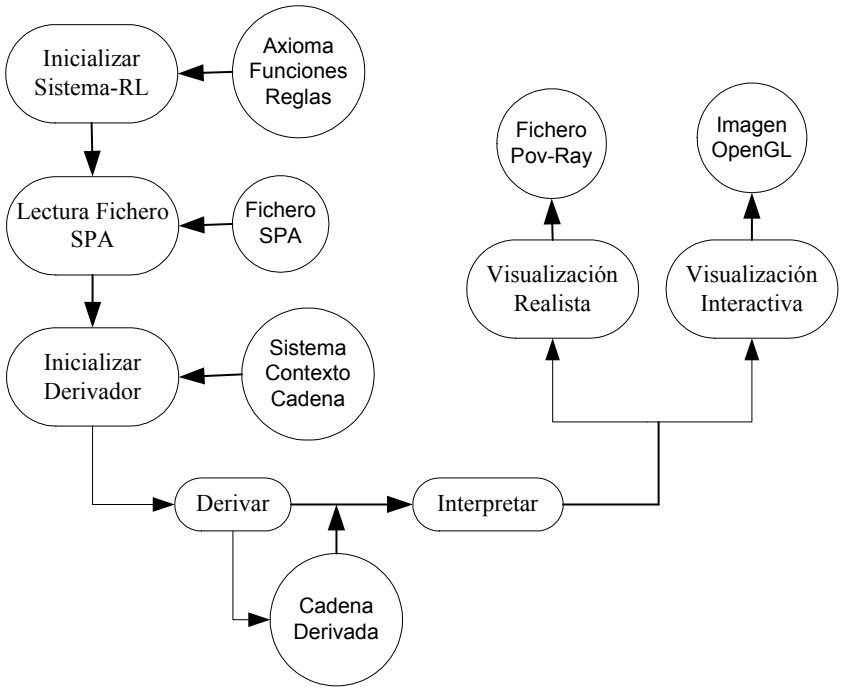


Figura 3.2: Esquema de funcionamiento general de GREEN

A continuación se describe la sintaxis diseñada en notación BNF [Naur60]. Todas las palabras en mayúsculas se consideran reservadas:

```
RL-System::=   SYSTEM [Declaration_Section]
               AXIOM  Axiom
               RULES  Production[Production]...
               END

Declaration_Section::=DECLARATIONS Declaration[Declaration]...
Declaration::=Var_Type Var_Class Var_Name Values
Var_Type::=DISCRETE|CONTINUOUS
Var_Name::=<identifier>
Var_Class::=BINOMIAL|CCONSTANT|CUNIFORM|DCONSTANT|DUNIFORM|EXPONENTIAL|FMAX|FMIN|
FPODA|PESCALAR|POISSON|NORMAL
Values::=<number>[,<number>]...
Parameters::=<identifier>[,<identifier>]...
Axiom::=Module[Module]...
Module::=<identifier>'{' [Values]'}'
Production::=Predecessor[:Condition]#Successor
Predecessor::=[LeftPredecessor<]<identifier>'{'[Parameters]'}'[>RightPredecessor]
LeftPredecessor::= <identifier>'{'[Parameters]'}'[ <identifier>'{'[Parameters]'}']
RightPredecessor::= <identifier>'{'[Parameters]'}'[ <identifier>'{'[Parameters]'}']
Successor::=Successor_Item[Successor_Item]...
Successor_Item::=Successor_List|Multiple_Item|Bracketed_Item
Multiple_Item::={Multiple_Value,Succesor_List}
Multiple_Value::=<identifier>|<number>
Bracketed_Item::= '['Succesor_List']'
Successor_List::=Parametric_Module[Parametric_Module]...
Condition::=<logical_exp>
Parametric_Module::=<identifier>'{'[Parametric_List]'}'
Parametric_List::=<arithmetic_exp>[,<arithmetic_exp>]...
```

Los elementos <identifier>, <number>, <logical_exp> y <arithmetic_exp> representan los conceptos de *identificador*, *valor numérico constante*, *expresión lógica* y *expresión aritmética* respectivamente, habituales en los lenguajes de programación de alto nivel.

A continuación se representa esquemáticamente la sintaxis diseñada:

SYSTEM
DECLARATIONS
tipoVariable nombreVariable parámetros
•
tipoVariable nombreVariable parámetros
AXIOM
axioma
RULES
predecesor izq < predecesor > predecesor der : condicionLógica # sucesor
•
predecesor izq < predecesor > predecesor der : condicionLógica # sucesor
END

En el anexo B se pueden consultar el tipo de variables aleatorias y los parámetros que necesitan, así como el nombre y significado de cada uno de los módulos con significado gráfico.

Por otra parte, las expresiones que forman la condición lógica siguen la sintaxis empleada en C++ para describir expresiones (salvo los operadores *AND(&&)*, *OR(||)* y el operador de *igualdad*, que se representan mediante los símbolos **&**, **|**, **=**, respectivamente):

*(A+B), (A-B), (A*B), (A/B), (A^B), (A&B), (A|B), (!A), (A<B), (A>B), (A=B), (A<=B), (A>=B).*

Asimismo, se permite el empleo de las funciones trigonométricas seno y coseno, especificándose de la siguiente manera:

cos(A), sin(A).

Además de los parámetros indicados en los módulos, se pueden emplear todas las variables aleatorias y funciones declaradas en la parte DECLARATIONS para formular una expresión, tanto en la condición, como en los parámetros de los módulos del sucesor.

Hay que hacer una mención especial a las funciones *FMax, FMin o FPoda* (por ejemplo, podemos tener en la declaración la línea "*Fmin min*"), ya que son funciones que aceptan parámetros cuando se utilizan en las expresiones. Así, la expresión "*min(x,y)*" calculará cual es el mínimo entre los valores *x* e *y*, y tomará dicho valor en la expresión. En la declaración de la función de poda se indica un fichero en el que se habrá definido un volumen mediante modelado CSG. A la función de poda se le pasa como parámetro una posición y devuelve cierto si la posición está dentro del volumen o falso si no lo está.

Por último, citar que en todos los S.P.A. existen dos variables implícitas especiales, **NumIt** y **MaxIt**, que hacen referencia a la iteración actual y al número máximo de iteraciones respectivamente. A continuación se presenta un ejemplo de un sistema paramétrico aleatorio en el que se pueden apreciar la mayoría de características comentadas.

```
SYSTEM
DECLARATIONS
 CUniform d 0. 1.
 CConstant alfa 50.
 CConstant beta 30.
 CConstant teta 90.
 FPoda poda poda1.pod
 FMin min
AXIOM
       Cylinder{1.,0.1,0.1}F{1.}A{1.}?P{1.,1.,1.}
RULES
   A{k}>?P{x,y,z}:(NumIt<MaxIt)&(k=0.)#A{1.}

   A{k}>?P{x,y,z}:(NumIt<MaxIt)&(d<=min(1.,(((2.*k)+1)/(k*k))))&(!poda(x,y,z))#
   /{teta}Push{}+{alfa}Cylinder{1.,0.1,0.1}F{1.}A{k+1.}?P{x,y,z}Pop{}
   -(beta}Cylinder{1.,0.1,0.1}F{1.}A{k+1}

   A{k}>?P{x,y,z}:(NumIt<MaxIt)&(!poda(x,y,z))#
   /{teta}B{k+1.,k+1.}-(beta}Cylinder{1.,0.1,0.1}F{1.}A{k+1.}

   A{k}>?P{x,y,z}:(NumIt<MaxIt)#T{}%{}

   F{t}>S{}:(NumIt<MaxIt)#S{}Cylinder{1.,0.1,0.1}F{1.}

   B{m,n}>-{h}Cylinder{i,j,k}:(NumIt<MaxIt)#B{m+1.,n}

END
```

3.3 Derivación

El proceso de derivación consiste en leer un fichero de gramática, donde se incluye la especificación del sistema RL, y aplicar las reglas que lo forman a cada uno de los módulos que forman la cadena de entrada tantas iteraciones como el usuario indique. Partiendo del axioma definido en el fichero, la cadena generada en una iteración se toma como entrada de la siguiente.

Antes de comenzar a explicar los algoritmos que se han desarrollado para realizar el proceso de derivación se deben tener claros los siguientes conceptos:

- Variables aleatorias: toman un valor cada vez que se deriva un módulo, por lo que cuando se está buscando si un módulo coincide con alguna regla, su valor no puede variar. Si están bloqueadas no pueden cambiar de valor.
- Contexto: es un diccionario de parejas, palabra clave y valor, que incluye los valores que se han calculado para las variables aleatorias y para los parámetros actuales.
- Expresiones: en una expresión se pueden utilizar funciones, variables aleatorias, constantes y parámetros. Pueden aparecer en la condición (expresión lógica) o en los parámetros de los módulos de los sucesores (expresión aritmética). Evaluar una expresión consiste en obtener su valor teniendo en cuenta los valores almacenados en el contexto.
- Parámetros: los parámetros formales del predecesor toman valores en función de los correspondientes parámetros actuales del módulo a derivar, incluyendo estos valores en el diccionario.

A continuación se van a comentar las distintas fases que componen el proceso de obtención de la cadena derivada, teniendo en cuenta la estructura de clases definida en el apartado 3.1 y el formato de fichero que se ha explicado en el apartado anterior.

3.3.1 Inicialización del sistema RL

Para poder empezar el proceso, es necesario crear un nuevo sistemaRL e inicializar sus variables de instancia que almacenan: el axioma, el diccionario de funciones y la lista de reglas. Todas ellas estarán inicialmente vacías.

3.3.2 Lectura del fichero del sistema

Una vez se ha creado el sistemaRL se deben asignar a las variables de instancia los valores que el usuario especifique en el fichero de la gramática, para ello es necesario realizar un proceso de lectura, teniendo en cuenta el formato de descripción de los sistemas RL.

En primer lugar se leerán la variables aleatoria y funciones del sistema, si las hay, teniendo en cuenta el tipo, ya que el número de parámetros a leer será diferente. Después se leerá el axioma del sistema. Por último se realizará la lectura de las reglas, leyendo el predecesor, condición y sucesor. En la lectura del predecesor se debe determinar si la regla es de contexto libre o no, y en caso de que no lo sea, se debe leer el predecesor izquierdo, el derecho o ambos.

3.3.3 Inicialización del derivador

Una vez se ha leído el sistemaRL se debe crear un derivador, asignarle el sistema leído e inicializar la cadena actual con el axioma y el contexto con el contexto inicial que incluye las variables aleatorias del sistema y las variables implícitas: **NumIt** y **MaxIt**.

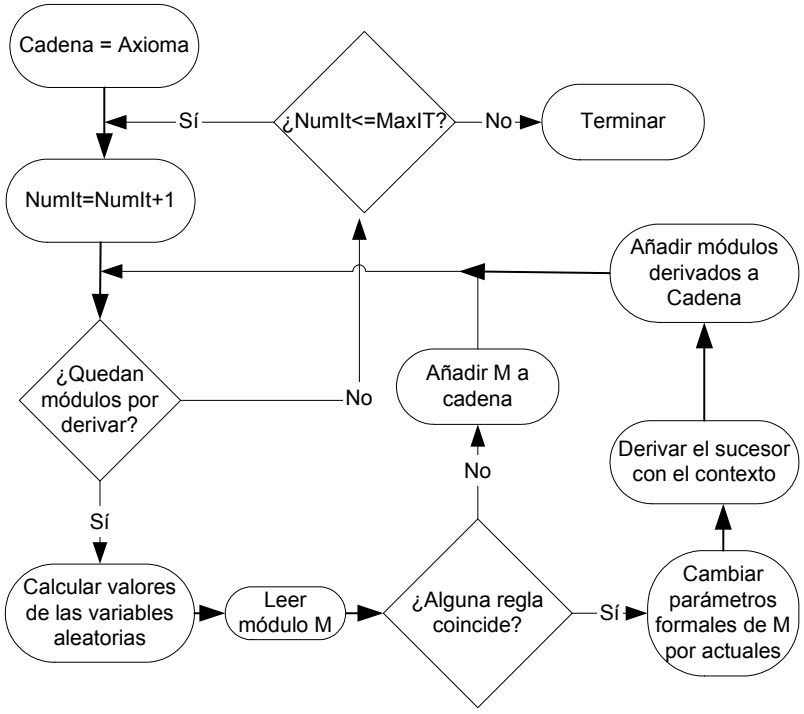


Figura 3.3: Esquema de funcionamiento del proceso de derivación

3.3.4 Proceso de derivación

El proceso de derivación se puede comenzar cuando se han inicializado todos los elementos necesarios. El proceso se realizará tantas veces como el usuario lo solicite, almacenando en el derivador las cadenas intermedias que se vayan obteniendo. En la figura 3.3 se muestra el esquema de funcionamiento de este proceso.

En primer lugar se llamará al procedimiento *DerivaVeces,* el cual será el encargado de inicializar la variable del sistema **MaxIt** con el número de iteraciones que se debe derivar el sistema. A continuación se presentan los algoritmos que realizan el proceso de derivación.

Procedimiento DerivaVeces (**MaxIt**)Clase Derivador (*D→derivaVeces(n_iteraciones)*)
Añadir al contexto **MaxIt**
Repetir **MaxIt** veces
D→deriva
Fin Procedimiento

Procedimiento deriva Clase Derivador
inicial = **D→cadenas[NumIt]**
Incrementamos el valor de **NumIt** en una unidad.
derivada = nueva Cadena // *donde se guardan los módulos del proceso de derivación*
Desbloqueamos las funciones de **D→contexto** // *Pueden cambiar de valor*
Para todos los módulos **m** de **inicial** hacer:
Realizamos las funciones de **D→contexto** // *Se calcula un valor*
Bloqueamos las funciones de **D→contexto** // *Pueden cambiar de valor*
Buscamos la regla que hace match con **m**
rv=*sistema→reglaValida_enEste_izquierda_derecha(m,contexto,inicial)***(2)**
Desbloqueamos las funciones de **D→contexto**
FPara

Si se ha encontrado **rv** que haya hecho match
Se añaden a **D→contexto** los parámetros actuales de **m**
rv→predecesor→contextoActual(inicial,pos_ini,NULL) **(3)**
Derivamos el sucesor en **D→contexto** y añadimos la cadena resultante de la derivación a la cadena **derivada**
rv→sucesor→deriva(contexto) **(4)**
Sino //*No se puede aplicar ninguna regla*
Añadir a **derivada** una copia del módulo **m**
FSi

Procesar **derivada** mediante *interpretaDeriva* teniendo en cuenta la poda **(5)**
Añadir los módulos resultantes de dicho proceso a **cadenas[NumIt]**

Fin Procedimiento

(2) Se debe buscar la primera regla del sistema que cumpla que el predecesor coincide con el módulo a derivar, el contexto izquierdo y derecho del módulo coinciden con el del predecesor de la regla y la condición se evalúa a cierto

Función reglaValida(Modulo **m**,Diccionario **contexto**,Cadena **inicial**)Clase SistemaRL:Regla

Para toda regla **r** del Sistema

```
        Si r hace match para el módulo m y contexto
            Si la condición es cierta en contexto
                r→condicion→cierta(contextoRegla) (2')
                Devolver r
            FSi
        FSi
    FPara
    Devolver Regla vacía

Fin Procedimiento
```

(2') Comprobar que la condición es cierta en el nuevo contexto

```
Función cierta(contexto) Clase Condición : Booleano

    Devuelve el resultado de evaluar la expresión en contexto

Fin Procedimiento
```

(3) Obtener el contexto actual, incluyendo los parámetros formales del módulo a derivar con su valor actual

```
Función contextoactual(cadena) Clase Cadena: Contexto

    Si las dos cadenas tienen el mismo número de módulos
        Para cada módulo m
            Para toda pareja de parámetros (formal – actual)
                m→equivalencias(moduloactual)
                Añadir asociación a contexto
            FPara
        FPara
        Devolver contexto
    Sino
        Devolver Diccionario vacío
    FSi

Fin Procedimiento
```

(4) Derivar el sucesor en el contexto actual

```
Función deriva(contexto) Clase Regla : Cadena

    Para todo módulo m del sucesor
        Derivar m en contexto
        m→deriva(contexto) //Depende de las subclases (4')
        derivada = Añadir el resultado de la derivación
    FPara
    devolver derivada

Fin Procedimiento
```

(4') Derivar un módulo en el contexto actual, si el módulo es especial derivará de forma diferente

Función deriva(**contexto**) Clase Módulo : Cadena
// *para módulos no especiales, si son especiales depende de su función*
Crear un nuevo módulo **m2** de la misma sub clase que **m**
Poner el nombre de **m** en **m2**
Para todo **p** parámetro del módulo
p→deriva(contexto) //devuelve una cadena o una expresión evaluada
Añadir el resultado a parámetros de **m2**
FPara

devolver **m2** con los parámetros evaluados

Fin Procedimiento

(5) Paso intermedio de derivación para aplicar la función de poda, es necesario ir interpretando a la vez que se deriva para poder dar valores a los símbolos de interrogación ?PHLU

Función interpretaDeriva (**tortuga**) Clase Módulo : Cadena

Si **m** tiene significado gráfico
Modificar **tortuga**
Sino si es un módulo de pregunta (?PHLU)
Asignar a los parámetros el valor del vector adecuado almacenado en **tortuga**
FSi
Devolver **m** como una cadena

Fin Procedimiento

3.4 Interpretación

Como resultado del proceso de derivación se obtiene una cadena paramétrica, la cual puede ser interpretada gráficamente para poder visualizar el objeto modelado. Un intérprete estará formado por:

- una cadena, que proviene del proceso de derivación y debe ser interpretada
- una tortuga, para interpretar los módulos con significado gráfico

Cada módulo tendrá su significado, que se puede consultar en el anexo B, pero a la hora de interpretar los podemos dividir en tres grandes grupos:

- Aquellos que no tienen significado para la interpretación, entre los que se encuentran los símbolos no terminales, los de encuesta y los de control de derivación
- Los símbolos que modifican la tortuga, es decir los de giros, avance y manejo de la pila de estados
- Los que generan un elemento visible: cilindros, esferas y formas

En la fase de interpretación los primeros se ignorarán, los segundos modificarán la tortuga del intérprete y los terceros generarán una salida en el fichero o crearán un nodo en la estructura de datos que se utiliza en la visualización.

El primer paso consiste en crear un nuevo interprete, asignándoles las variables de instancia: la cadena contendrá la última cadena generada en la derivación y la tortuga estará inicializada.

Posteriormente todos los módulos de **cadena** se interpretarán de forma secuencial, y en función de si el usuario ha elegido realizar la visualización mediante POVRay o de forma interactiva, el resultado de la interpretación será diferente.

3.4.1 Interpretación creando un fichero de POVRay

Para facilitar la creación de una imagen en POVRay se ha incluido en la aplicación un editor de texto que incluye una plantilla de fichero POVRay, en la que se han definido la cámara, el cielo, la iluminación, los ficheros de texturas y colores necesarios y la declaración de un polígono para ser utilizado como hoja. El fichero creado en el proceso de interpretación contendrá la geometría y el aspecto del objeto generado y por lo tanto se deberá incluir al final de la plantilla.

Los símbolos que generan líneas en el fichero son los siguientes:

- *Cylinder (altura, radioinf, radiosup)*: crea un cono truncado indicando el punto inicial (posición de la tortuga), radio inicial, punto final (se avanza la tortuga en la dirección del vector *H*, las unidades indicadas en altura) y radio final. También se puede indicar el material del que está formado
- *Sphere (radio)*: crea un objeto esfera, con centro en la posición de la tortuga y los dos radios iguales al radio definido en el parámetro del módulo
- *Instance (índice, esc1, esc2, esc3)*: crea un objeto predefinido, se indica el índice del objeto y se cambia de tamaño en función de los factores de escala indicados como parámetros, se orienta en función de los vectores *H, L* y *U* de la tortuga, y se traslada a la posición de la tortuga

En la figura 3.4 se puede observar el editor del fichero de configuración de POVRAY que incorpora la aplicacioón. El formato del fichero POVRay se puede consultar en (www.povray.org) y en la tabla 3.2 se enumeran las líneas de fichero que se escriben cuando se interpreta cada uno de los símbolos.

Símbolo	PovRAY
Cylinder	cone { punto1 radio1, punto2 radio2 texture { T_Wood1 } }
Sphere	sphere{punto radio1, radio2 }
Instance	object { Forma[forma] scale<sc1,sc2,sc3> matrix< vL.x, vL.y, vL.z, vH.x, vH.y, vH.z, vU.x, vU.y, vU.z ,0.0 ,0.0 ,0.0> translate punto}

Tabla 3.2: Línea de fichero POVRay producida por la interpretación de los símbolos

3.4.2 Interpretación creando una estructura de datos

Para realizar una visualización del objeto modelado de forma interactiva, se debe generar una estructura de datos, en el proceso de interpretación. La estructura de datos utilizada para la visualización es un árbol. Los nodos que forman el árbol almacenan los siguientes elementos:

- *int m_nivel*: Nivel de profundidad del nodo en el árbol
- *int m_numHijos*: Número de hijos del nodo
- *CNodo* m_hijoIzq*: Puntero al hijo izquierdo del nodo
- *CNodo* m_hermanoDer*: Puntero al hermano derecho del nodo
- *CNodo* m_padre*: Puntero al padre del nodo
- *CElemento* m_elemento*: Rama u hoja que contiene el nodo

Cada elemento tiene asociado el tipo (rama u hoja), la matriz de transformación que escala, orienta y traslada el elemento y el material y la textura que definen su aspecto final. Para incluir un nuevo nodo en el árbol se utiliza la función:

void NuevoHijo(int tipo, float params[], int np, Matriz mt)

donde *tipo* indica el tipo de elemento que almacena el nodo (rama u hoja), *params* almacena los parámetros que definen la geometría del elemento, *np* indica el número de parámetros que almacena *params* y *mt* es la matriz de transformación del elemento, que se forma a partir de los vectores y la posición de la tortuga.

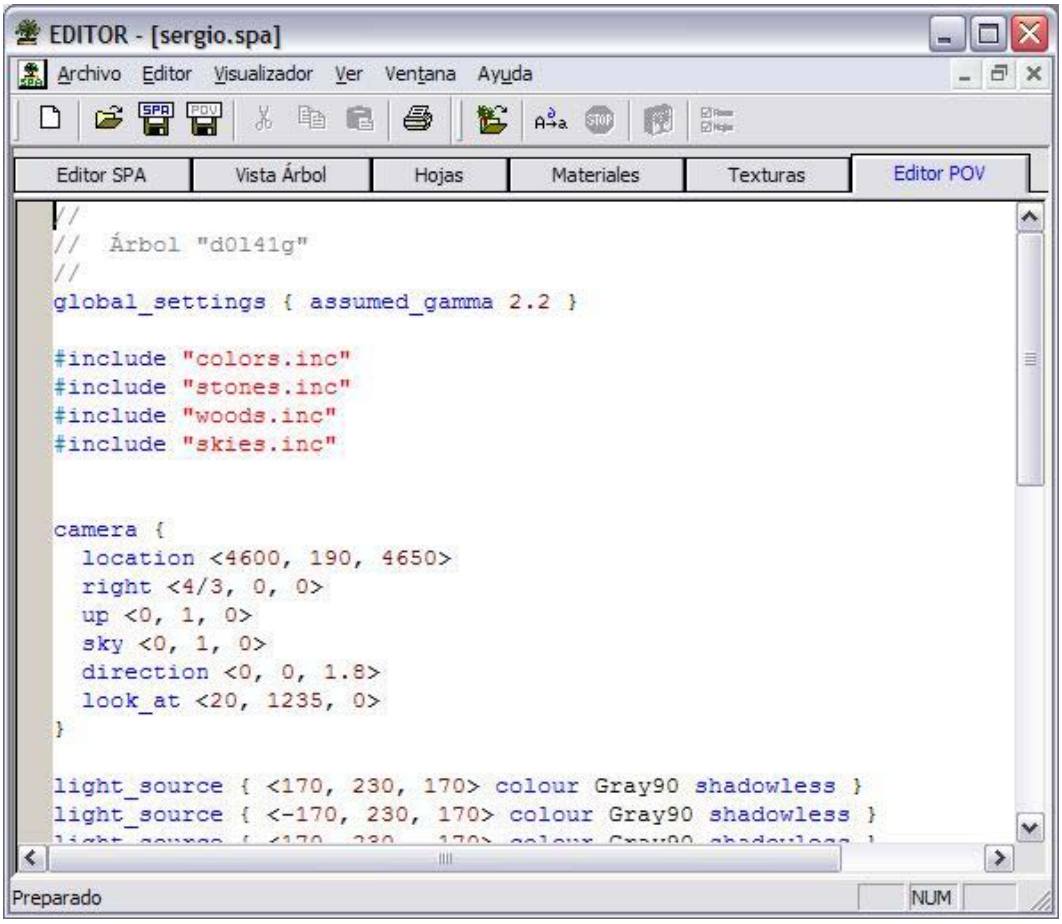


Figura 3.4: Editor del fichero POV, con plantilla

3.5 Visualización

El fichero generado es directamente visualizable mediante la aplicación POVRay, para ello se debe de ejecutar la plantilla que incluya el fichero del que se desea obtener una imagen. En la figura 3.5 se puede observar una imagen generada con POVRay a partir de un fichero generado por GREEN.

Para visualizar de forma interactiva, se recorren todos los nodos de la estructura de datos, se le asigna el material asociado al elemento y se visualiza transformado por la matriz asociada. Las hojas se pueden representar mediante polígonos con material o con texturas. Para seleccionar la forma del polígono se utiliza el cuadro que se puede observar en la figura 3.6. Para seleccionar el material que utilizan las ramas y las hojas se presenta el cuadro de diálogo que aparece en la figura 3.7. En la figura 3.8 se puede observar la ventana de selección de textura. Y por último, en la figura 3.9 se muestra la ventana de visualización interactiva de un árbol generado por GREEN.

Figura 3.5: Imagen generada mediante POVRay de un modelo editado en GREEN

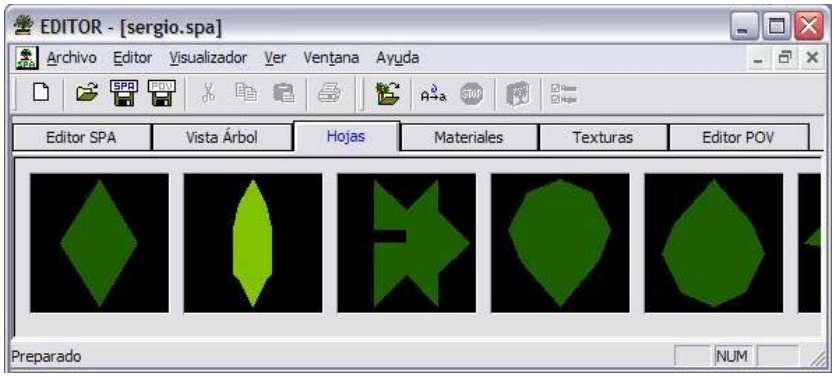


Figura 3.6: Selección del polígono de las hojas

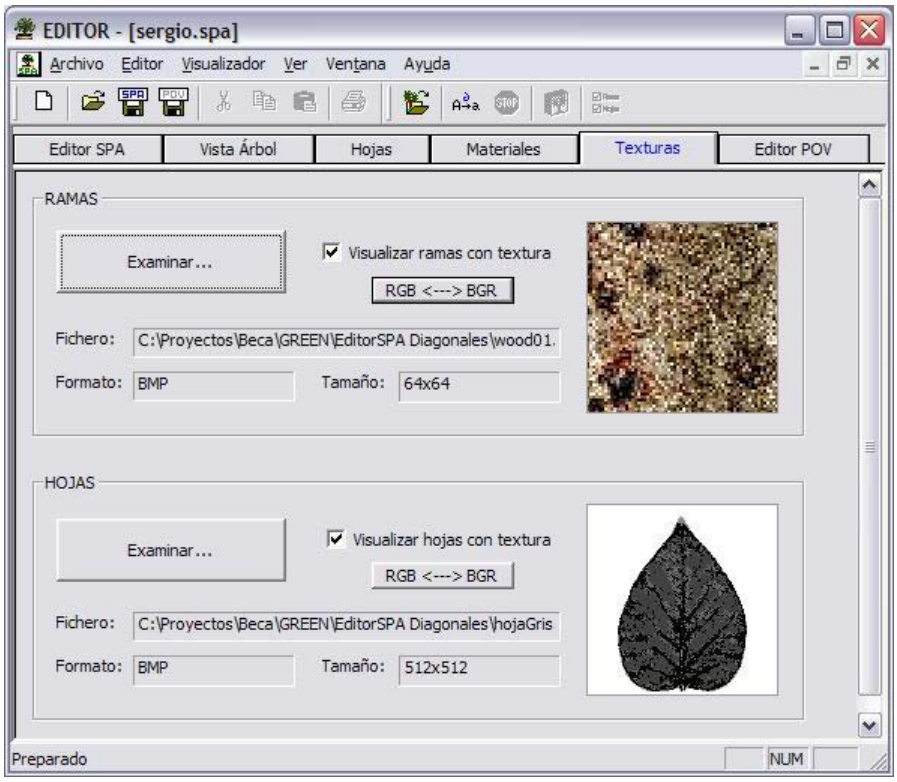


Figura 3.7: Selección de la textura de las hojas y de las ramas

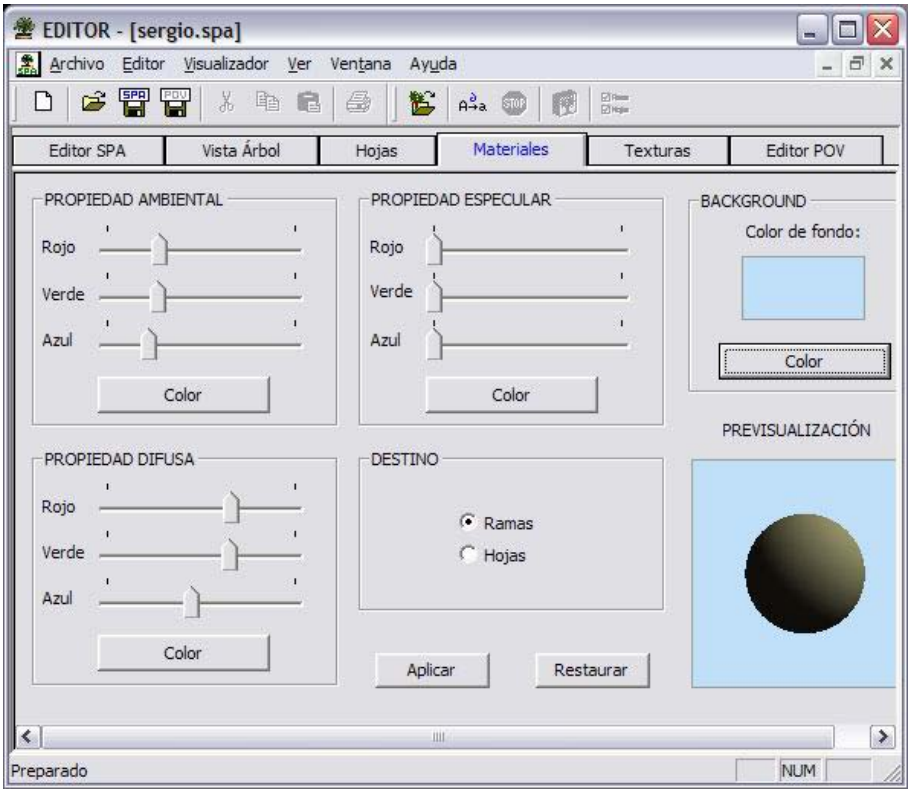


Figura 3.8: Selección del material de los elementos

Figura 3.9: Visualización interactiva de GREEN

3.6 Otras aplicaciones

Se ha realizado un estudio de otras aplicaciones desarrolladas por otros grupos de investigación, con la intención de realizar una comparación con la herramienta desarrollada. Las aplicaciones analizadas son: TREE Professional, Xfrog y Lstudio. Todas ellas tienen un coste que no supera los 600 €. Se han dejado fuera del estudio aquellas aplicaciones comerciales, de coste mucho más alto, y que tienen una funcionalidad mucho mayor, como por ejemplo la edición de jardines o ecosistemas, la realización de animaciones sobre el entorno editado, la simulación de fenómenos de erosión, sequía, etc. Un ejemplo de este tipo de aplicaciones es AMAP (www.cirad.fr).

3.6.1 Tree Profesional

Tree Professional (www.onyxtree.com) es una herramienta de modelado paramétrico y visualización realista de árboles, palmeras y otros tipos de plantas. Una vez generado el árbol se puede almacenar en varios formatos; como un fichero de parámetros, como una imagen BMP ó TGA, y como un modelo 3D.

No se trata de una simple librería de distintas especies de plantas, aunque el programa viene acompañado de un gran número de árboles de hoja ancha y coníferas, palmeras y arbustos ya modelados. Se trata de un modelador de árboles real, que permite modelar virtualmente cualquier tipo de árbol gracias a la potencia descriptiva de los métodos de modelado paramétricos y que este programa incorpora. Al contrario de lo que ocurre en el modelado convencional, en el paramétrico no se comienza desde cero, ya que el conocimiento que se tiene sobre la morfología y crecimiento de los árboles está codificado de antemano.

El modelado del árbol se basa en manipular algunas de sus características (parámetros) esenciales como la altura, la curvatura y la densidad de las ramas, el tipo y color de las hojas. Modificando este tipo de parámetros se pueden obtener distintas especies de plantas, variaciones de las mismas especies, árboles en diferentes etapas de su crecimiento o en diferentes estaciones. Los parámetros son almacenados en el modelo mientras se trabaja sobre ellos de manera que los cambios permanecen siempre presentes.

A pesar de los numerosos parámetros, es bastante sencillo manejar este programa. Los parámetros están agrupados lógicamente y se pueden acceder a diferentes niveles de detalle de modelado. Estos niveles permiten concentrarse en una clase de elementos, lo cual es una ayuda en el proceso de modelado.

Además incluye la iluminación ambiental proveniente del sol, siendo capaz de representar todos los matices de la reflexión del sol, translucidez, cambios de color y las múltiples sombras que se producen cuando el sol proyecta su luz sobre el árbol. Todos estos cálculos se realizan en unos pocos segundos. También es posible utilizar los modelos desarrollados en animaciones ya que se puede exportar su geometría a un formato 3D estándar como 3DS, DXF, etc. Es posible ajustar el número de polígonos generados.

Otra característica importante consiste en la técnica Direct Image Síntesis desarrollada en la aplicación, que permite mezclar el modelado y la visualización del árbol en único proceso. La generación del árbol está dirigida por un complejo sistema de reglas basadas en el conocimiento acerca de la anatomía y el crecimiento de los árboles, y en los fundamentos de la pintura. Esta técnica permite una generación más rápida del árbol y una visualización más realista.

En la figura 3.10 se puede apreciar el aspecto general de la aplicación, las partes principales son el menú, la ventana de visualización, los iconos y el panel de parámetros. En el menú se puede acceder a las típicas funciones de fichero y edición, también es posible elegir el tipo de árbol que se quiere modelar: de hoja ancha, conífera o palmera, y por último se pueden seleccionar distintas opciones de visualización como el fondo y auto-redibujado.

Los iconos situados en la parte inferior de la ventana de visualización permiten definir: los elementos que se van a visualizar (tronco, ramas, hojas, etc), cómo se van a visualizar (líneas, alámbrico, sólido), inclusión de luz solar (es configurable) y sombras y visualización de la envoltura del árbol. Por último aparecen un grupo de iconos que permiten cortar, hacer zoom, desplazar y redibujar.

El panel de control de los parámetros permite acceder y modificar los valores de los distintos parámetros que permiten el modelado del árbol. Los parámetros diferirán en función del tipo de árbol elegido (árbol de hoja ancha, conífera o palmera) y del elemento a modificar (tronco y ramas de primer orden, resto de ramas, tamaño y color del tronco y las ramas, y follaje).

Las conclusiones a las que se han llegado tras un estudio de las características del sistema de modelado se resumen a continuación, resaltando aquellas cuestiones favorables y negativas que se han encontrado en el análisis de la aplicación. A favor de esta aplicación se puede decir:

- El interfaz de usuario es muy completo en cuanto a opciones de modelado se refiere.
- Es sencillo de utilizar y se puede diseñar un árbol con una apariencia realista de forma rápida.

- La malla poligonal puede emplearse en proyectos de animación o en entornos de tiempo real que utilicen multirresolución poligonal.

Por el contrario, tiene algunos aspectos negativos entre los que cabe destacar:

- El interfaz de usuario adolece de opciones de visualización.
- Tan solo se pueden modelar tres niveles de ramas.
- El redibujado del modelo es lento, puede tardar varios segundos tras un cambio.
- No permite simular la evolución del individuo, ni procesos internos de intercambio de información entre las distintas partes del árbol.
- No es posible modelar interacción con elementos externos.
- Las imágenes que genera no son de gran calidad, pero permite exportar el modelo 3D a formatos muy utilizados como el 3DS, C4D o DFX.

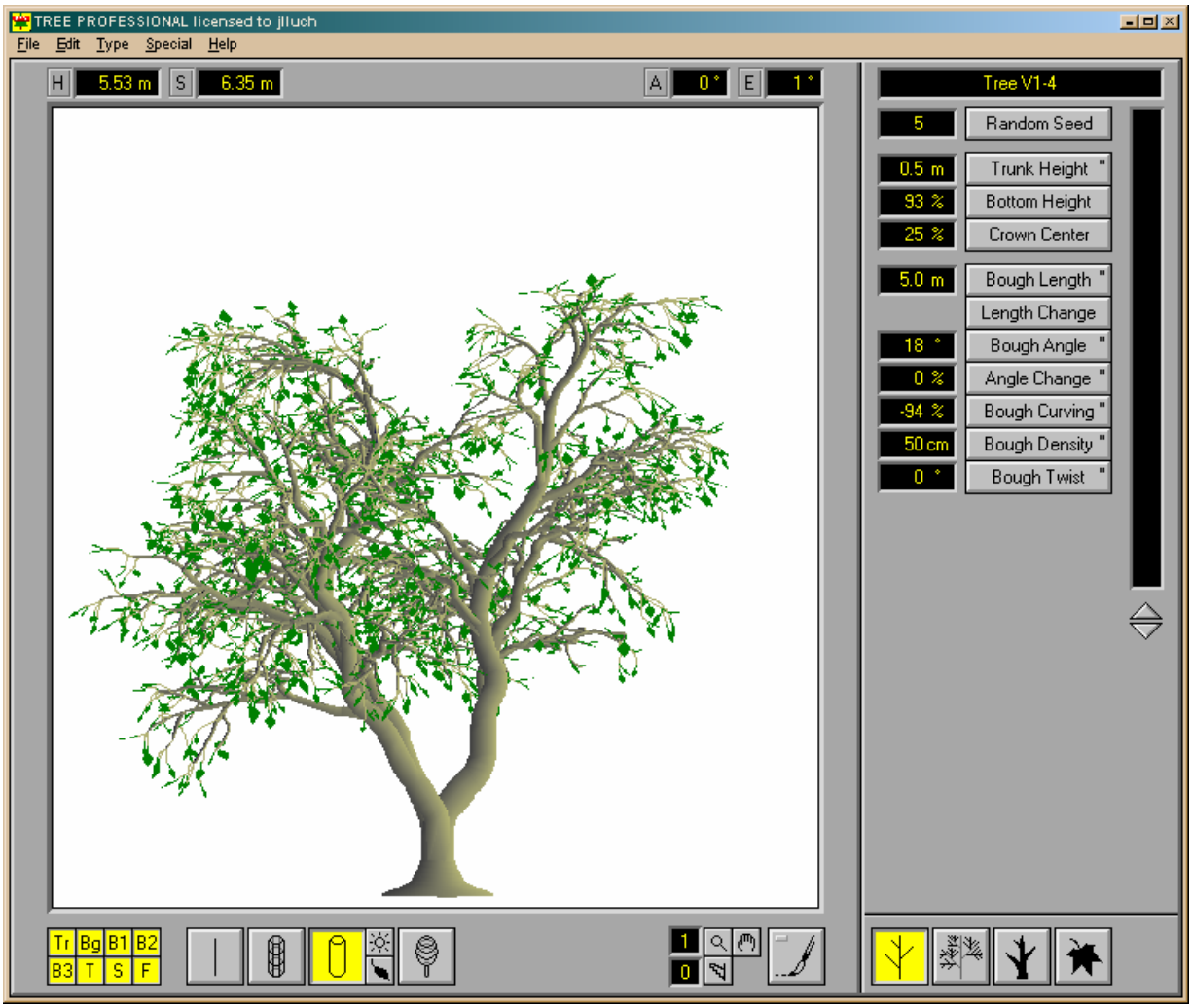


Figura 3.10: Aspecto general de Tree Profesional 5

3.6.2 Xfrog

Esta aplicación esta basada en los trabajos realizados por Lintermann [Lint99], comentados en el capítulo anterior. Con este método es posible la generación de diferentes objetos ramificacdos como árboles, plantas, arbustos, flores e incluso elementos no botánicos. La especificación de un determinado modelo se realiza mediante un grafo, a través del cual se

obtiene la geometría resultante. Los elementos del grafo son un conjunto de componentes geométricos y estructurales.

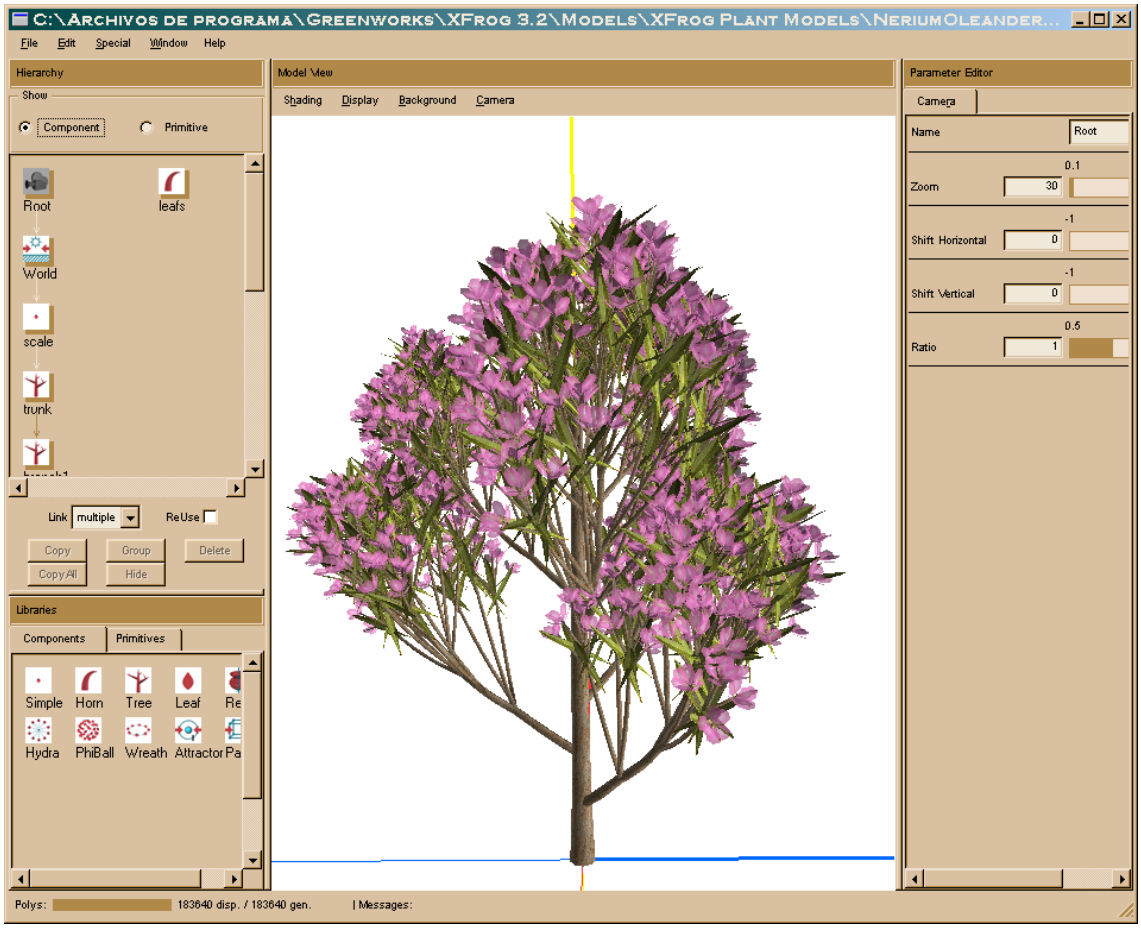


Figura 3.11: Aspecto general de Xfrog

El usuario percibe de forma inmediata la forma del modelo que está creando, es decir cada vez que modifica el grafo. El modelado está orientado a la unión de diferentes componentes, combinando objetos geométricos (primitivas) con información de cómo organizar la geometría. Se pueden obtener modelos complejos, de forma sencilla, combinando diferentes componentes y asignándoles primitivas.

Los componentes encapsulan datos y algoritmos que permiten la generación de los elementos que forman la planta. Existen tres grupos de componentes con diferentes funciones. Los primeros generan objetos gráficos como tallos, hojas o primitivas geométricas; los segundos, multiplican el resto de componentes; y los terceros, aplican técnicas globales de modelado.

La aplicación tiene el aspecto que aparece en la figura 3.11. La ventana contiene el menú principal en la parte superior y se divide en las siguientes sub-ventanas:

- Jerarquía, donde se visualiza el grafo que se está editando.
- Vista del modelo, donde se visualiza el modelo que se está creando.
- Editor de parámetros, donde se pueden modificar los parámetros de las siguientes partes del modelo.
- Librerías, donde aparecen los componentes y primitivas que se pueden utilizar como elementos del grafo.

El menú principal contiene las típicas opciones, de archivo, edición, ventana y ayuda, y la ventana de visualización permite las típicas opciones de navegación, por lo que no merecen mayor comentario. A continuación se analizan el resto de componentes de la aplicación.

Editor de jerarquía: Es el lugar donde se puede editar el grafo de componentes. El nodo raíz es un nodo cámara que permite definir la vista, por lo tanto, cada componente que se enlace con el nodo raíz aparecerá en la ventana de visualización del modelo. La funciones que se pueden realizar son las siguientes:

- Creación de componentes: consiste en incluir un componente en el grafo.
- Enlazado de componentes: Si arrastramos un componente sobre otro que esté en el grafo, quedan enlazados inmediatamente, siendo el componente arrastrado hijo del otro. Un componente puede tener varios hijos, pero un único padre.
- Tipos de enlaces: Un componente contiene información que indica como colocar la geometría en el espacio. Cada componente tendrá uno o varios orígenes donde se colocará la geometría asociada al componente. En cada origen es posible crear geometría o enlazar un nuevo componente. Existen dos tipos de enlaces, simple y múltiple. El enlace simple conectará el nuevo componente al último origen del nodo padre. El enlace múltiple conectará el nuevo componente en cada origen del nodo padre.

Ventana de librerías: permite el acceso a todos los elementos de construcción, componentes y primitivas. Para incluir un nuevo componente en la estructura, sólo es necesario pinchar el componente requerido y arrastrarlo al grafo. Existen dos categorías de elementos:

- Componentes: Existen nueve tipos de componentes, *simple*, *horn*, *tree*, *leaf*, *revo*, *phiball*, *hydra*, *wreath* y *attractor*. El componente *simple* se usa para crear una instancia de una primitiva. El componente *horn* utiliza una primitiva tubo con forma de cuerno que se puede utilizar para modelar los tallos de las hojas y flores. El componente *tree* se utiliza para crear árboles, y crea un tubo en forma de cuerno con una fila de orígenes en la que se crea una nueva primitiva. El componente *leaf* permite la creación de hojas mediante la edición de un polígono. El componente *revo* genera un volumen de revolución. El componente *hydra* permite multiplicar otros componentes en forma de estrella. El componente *phiball* multiplica otros componentes en forma esférica. El componente *wreath* multiplica otros componentes en forma circular. Por último, el componente *attactor* se utiliza para deformar la geometría utilizando esferas invisibles que el modelo no puede atravesar.
- Primitivas: La primitivas que se pueden asociar a un componente son: vacía, caja, esfera, cono, cilindro, toro, rectángulo, círculo, tubo, área y atractor.

Editor de parámetros: en esta ventana es posible modificar todos los parámetros asociados a los diferentes componentes y primitivas que forman el modelo. Está dividido en cuatro partes, la primera permite la edición de los parámetros propios del elemento, la segunda permite cambiar parámetros básicos como la posición, la tercera contiene todos los parámetros referentes a la primitiva que se ha asignado al componente y en la cuarta se encuentran los parámetros que caracterizan las propiedades del material de la primitiva seleccionada.

Tras realizar este estudio de la aplicación se puede decir que los aspectos más favorables de la misma son los siguientes:

- El interfaz de usuario es muy sencillo de utilizar ya que está basado en la técnica de pinchar y arrastrar.
- Crear y modificar un grafo es muy sencillo.
- Se pueden crear imágenes en los formatos JPG y PNG.
- La malla poligonal se puede guardar en diversos formatos entre los que se encuentran: WRL, DXF y RIB. Esto permite utilizar los modelos 3D en otros entornos.
- El redibujado del árbol es muy rápido, ya que el visulizador se basa en OpenGL.

Por otro lado, los aspectos más negativos que se han encontrado se enumeran a continuación:

- Para la obtención de un árbol realista se necesita bastante experiencia y tiempo para establecer el valor de todos los parámetros que lo componen.
- No permite simular la evolución del individuo, ni procesos internos de intercambio de información entre las distintas partes del árbol.
- La interacción con elementos externos se modela de forma muy restringida.

3.6.3 Lstudio

Para terminar con este estudio de las aplicaciones desarrolladas por otros grupos se ha realizado un análisis del Lstudio, que es una aplicación desarrollada por el grupo de investigación de P. Prusinkiewicz y que recoge todos los avances técnico que han desarrollado en el campo de los sistemas-L.

Como se puede apreciar en la figura 3.12, la aplicación se divide en 8 ventanas que permiten definir: el sistema-L en un fichero de texto, los parámetros de la cámara sintética, los parámetros de animación, el color de los elementos, la superficie de una hoja mediante puntos de control, el contorno de las ramas, funciones y un fichero de texto. En la figura 3.13 se puede observar un árbol modelado mediante Lstudio.

Figura 3.12: Vista general de la aplicación L-Studio

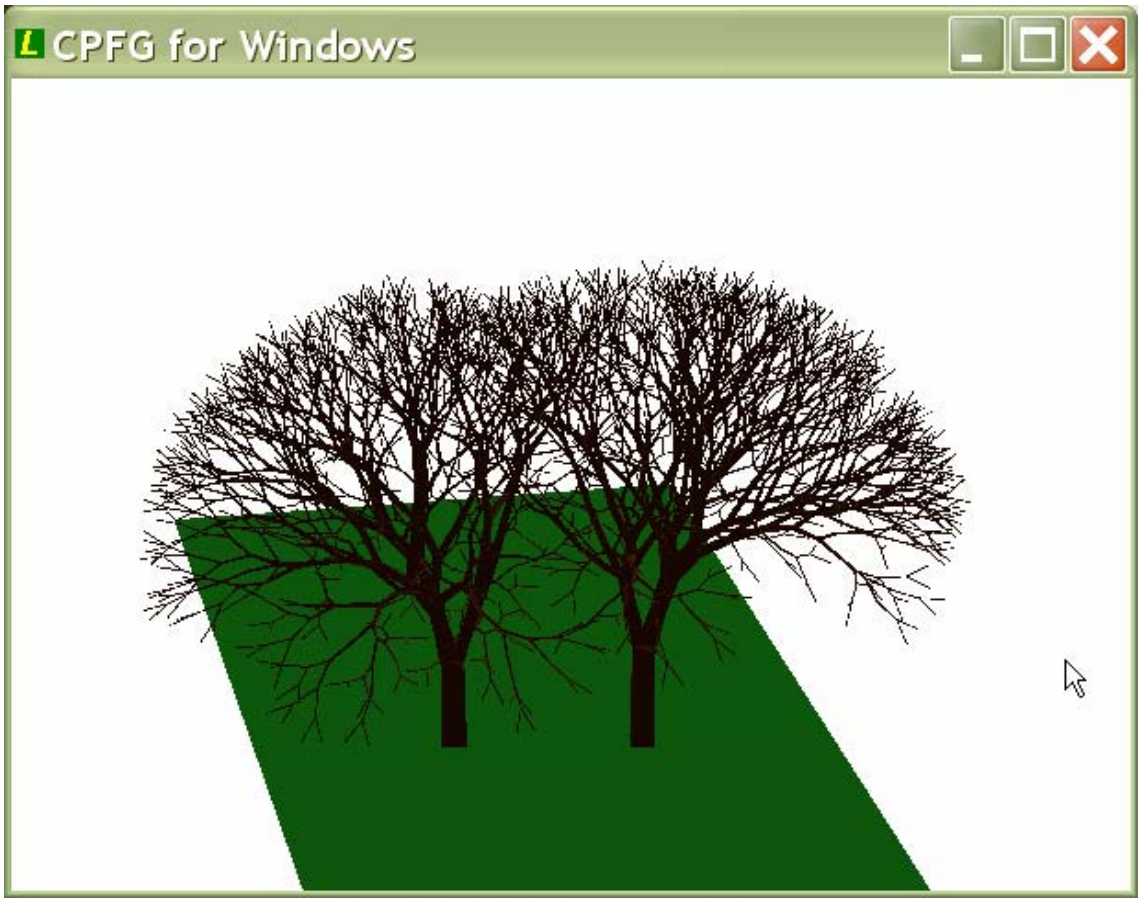


Figura 3.13: Arboles modelados mediante L-Studio

Las posibilidades de edición de modelos es muy amplia, ya que permite: sensibilidad al contexto, sensibilidad al entorno, poda, tropismo, sub-sistemas, homomorfismos y sistemas abiertos [Méch98]. Las principales características de la aplicación se enumeran a continuación:

- Definición de variables y vectores
- Existen tres tipos de estructuras: asignación de una expresión a una variable, condicional mediante *if o if-else* y repetitiva mediante *while o do-while*
- Posibilidad de ejecutar instrucciones al principio o final de la derivación o al final de cada iteración
- Funciones predefinidas: matemáticas, trigonométricas, de entrada y salida, para animación y variables aleatorias.
- Sub sistemas-L que permiten dividir la estructura del sistema en partes más pequeñas e independientes
- Homomorfismos que permiten definir un conjunto de producciones que sólo se aplicarán en el proceso de interpretación, lo que permite cambiar la apariencia final sin modificar la lógica del modelo
- Descomposiciones que hacen posible sustituir un módulo de la cadena en diferentes componentes, de modo que después de cada iteración de la derivación y antes de la interpretación se sustituyen los módulos correspondientes a la descomposición
- Utilización de splines y cilindros generalizados para representar los órganos de las plantas
- Animación de modelos, tanto en desarrollo como de cámara

- Posibilidad de utilizar sistemas abiertos, en los que el entorno se simula con procesos generados por programas realizados por el usuario en lenguaje C.

Por lo tanto, se puede concluir que esta es una aplicación muy completa, ya que reúne una gran cantidad de aspectos relacionados con el modelado de las plantas que no contienen ninguna otra de las aplicaciones estudiadas. Tan sólo se pueden nombrar dos inconvenientes, el primero se refiere a la complejidad que tiene la definición de los ficheros de los sistemas-L. La sintaxis contiene más de 60 símbolos interpretables, además de la posibilidad de incorporar estructuras de control similares a las utilizadas en el lenguaje C. También hay que decir que esta complejidad está relacionada con el amplio repertorio de opciones que ofrece el sistema. El segundo inconveniente es la imposibilidad de definir texturas a la hora de visualizar los elementos gráficos que forman el modelo. Sin embargo es posible exportar el modelo a un editor, estilo 3D Studio, donde se pueden incluir texturas para obtener una visualización más realista.

3.7 Estudio comparativo

El principal objetivo que se persiguió cuando se decidió realizar la aplicación fue tener un sistema de modelado propio con el que poder realizar las pruebas de los futuros trabajos en los que el grupo de investigación decidiera trabajar. De este modo, se contaría con los fuentes de un modelador de especies vegetales, ya que no es posible adquirir los editores desarrollados por otros grupos incluyendo los programas fuentes, lo que es normal debido a la gran cantidad de trabajo necesario para su desarrollo. Las aplicaciones de dominio público a las que se tuvo acceso como por ejemplo, *Lparser* (www.xs4all.nl/~ljlapre), no reunían la calidad mínima necesaria.

La aplicación desarrollada se diferencia de las dos primeras en que el modelo que se utiliza son los sistemas-L, con las ventajas e inconvenientes que ello conlleva, las cuales se enumeran en la sección 2.6.3. Como se puede suponer, es más parecida a la tercera, incluso se podría decir que es un sub-conjunto de ella. Las únicas mejoras que incorpora es la utilización de texturas de hojas reales y de cortezas para las ramas. Por supuesto, se han desarrollado los algoritmos que se describen en el capítulo 5 para que los modelos obtenidos puedan ser visualizados de forma rápida en la aplicación que se ha desarrollado para la navegación en entornos naturales. En la tabla 3.3 se muestra un resumen de las principales características de cada una de las aplicaciones.

Además, los sistemas-RL, no sólo se han utilizado para la obtención de árboles y plantas, sino que también es posible obtener: Objetos geométricos generados mediante sustitución de formas [Quir95] [Quir96a] [Lluch99], atractores caóticos [Quir94], formaciones de coral [Quir97]. Incluso pueden ser utilizados para obtener visualizaciones no realistas [Camp01].

Características / Aplicación	GREEN	LSTUDIO	XFROG	TREE
Fuentes disponibles	Sí	No	No	No
Modelado paramétrico	Sí	Sí	Sí	Sí
Interfaz de Usuario	Medio	Bajo	Alto	Medio
Complejidad de modelado	Media	Alta	Alta	Media
Simulación del desarrollo	Sí	Sí	No	No
Simulación de procesos internos	Sí	Sí	No	No
Interacción con el entorno	Sí	Sí	Restr.	No
Modelado mediante funciones matemáticas	Sí	Sí	No	No
Utilización de aleatoriedad	Sí	Sí	No	No
Modelado mediante sub-componentes	No	Sí	Sí	No
Modelado de otros elementos	Sí	No	No	No
Utilización de Splines	No	Sí	No	No
Generación de malla poligonal	Sí	Sí	Sí	Sí
Utilización de Texturas	Sí	No	No	No
Visualización interactiva apoyado sobre OpenGL	Sí	Sí	Sí	No

Tabla 3.3: Comparativa de GREEN con las tres aplicaciones de modelado de árboles estudiadas

Capítulo 4
EVERGREEN: Entorno de Visualización de Escenas en el exterioR basado en GREEN

Una vez terminada la aplicación de modelado es necesaria la implementación de otra aplicación que permita la navegación de un usuario en el interior de una población formada por varios individuos. Esta aplicación recibe el nombre de EVERGREEN, ya que los modelos que se van a utilizar en ella, estarán diseñados mediante GREEN. La gran cantidad de geometría que es necesaria para visualizar este tipo de escenas, hace inviable que el número de individuos visualizados simultáneamente sea muy grande. Por lo tanto, esta aplicación se convertirá en un perfecto banco de pruebas donde analizar las técnicas que se proponen en la presente tesis para acelerar la visualización de poblaciones vegetales.

El desarrollo de la aplicación se enmarca dentro del proyecto: "Arquitecturas multirresolutivas de sistemas gráficos procedurales", financiado por la CYCIT, TIC99-0510-C02-01durante el periodo 2000/02. Los objetivos que se persiguen en la tesis se encuadran dentro del proyecto, en el que el primer paso consiste en la implementación de la aplicación que se va a comentar en este capítulo. Esta aplicación se mejorará considerablemente cuando se apliquen las técnicas multirresolución que se comentan en el capítulo siguiente.

EVERGREEN es un sistema que permite leer un modelo desarrollado en GREEN y visualizar copias del mismo, dispuestas en forma matricial a semejanza de un campo de árboles frutales. Se han añadido otros elementos visuales para incrementar el realismo de la escena: vallas, suelo, cielo y niebla. En este capítulo se va a realizar un estudio de las principales características de la aplicación.

En la siguiente sección se presenta la arquitectura del sistema, comentado las diferentes partes que lo forman y el esquema general de funcionamiento. En la segunda sección se analizan las técnicas para especificación de terreno. A continuación, se hace un análisis de cómo se modelan y se instancian las plantas para obtener un conjunto de individuos a visualizar. En la siguiente sección se comentan las técnicas de distribución de árboles existentes. En la sección quinta se comenta como se definen los diferentes elementos que forman el entorno. Posteriormente, se analizan las técnicas que se han desarrollado en el visualizador y se analizan los resultados que se han obtenido. Por último, se hace un repaso a las principales aplicaciones desarrolladas por otros grupos.

4.1 Arquitectura del sistema

La aplicación ha sido desarrollada en Visual C++ y utilizando OpenGL como librería de apoyo a la visualización. Para diseñar el interfaz gráfico se ha utilizado la librería GLUI que está basada enteramente en OpenGL. Por lo tanto no se han utilizado otras librerías adicionales que podrían ralentizar la visualización de la escena. En la figura 4.1 se puede observar una vista general de la aplicación.

El sistema se ha desarrollado de forma modular de manera que se pueda desarrollar cualquiera de las partes que lo forman por separado. Los principales módulos son: el campo, el terreno, el agente, el modelo de árbol y los elementos adicionales de la escena. Para cada módulo se ha desarrollado una clase, en la tabla 4.1 se pueden consultar las principales variables de instancia que componen cada una de las clases.

Clase	Variables de instancia
Terreno	ancho largo malla texturaSuelo texturaValla
Agente	posición visual
Arbol	traslación rotación precisión ramas hojas texturaRama texturaHoja
Ambiente	nieblaON texturaFondo direcciónLuz radioEsfera
Campo	filas arbolesPorFila arboles agente terreno ambiente

Tabla 4.1: Clases principales de EverGreen

Antes de pasar a comentar cada uno de los elementos que forman la aplicación, se debe comentar el funcionamiento general de la misma. Para ello se puede consultar el diagrama que se presenta en la figura 4.2.

Para la descripción del campo serán necesarios determinar los siguientes datos: el tamaño del terreno (ancho y largo), las texturas a utilizar en el suelo y la valla y la distribución de los árboles. Posteriormente se utilizarán los modelos editados en GREEN como entrada de la aplicación. Cada modelo se situará y orientará dentro del campo.

Seguidamente, se deben tener en cuenta otros elementos que forman la escena como: la iluminación, la esfera envolvente y la textura que utiliza y la posibilidad de aplicación de niebla en el ambiente. Una vez determinados todos estos elementos, la escena ya está formada, por lo que será necesario introducir un observador en la misma. Para ello, se utiliza el agente, indicándole una posición y una visual. Cualquier modificación de los elementos descritos supone un cambio de la visualización de la escena, que se tendrá en cuenta de forma interactiva.

En resumen, las tareas que se deben realizar para modelar una escena de estas características son las siguientes: especificación del terreno, modelado de las plantas, instanciación de los modelos, distribución de los árboles, definición del entorno y visualización interactiva de la escena.

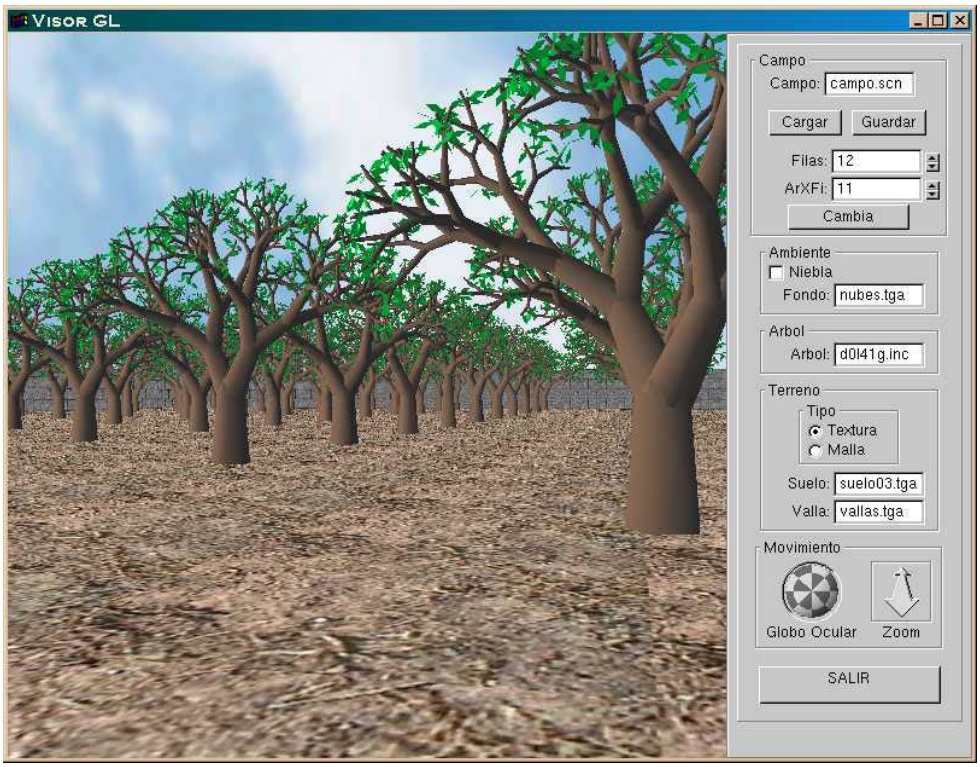


Figura 4.1: Vista general de EVERGREEN

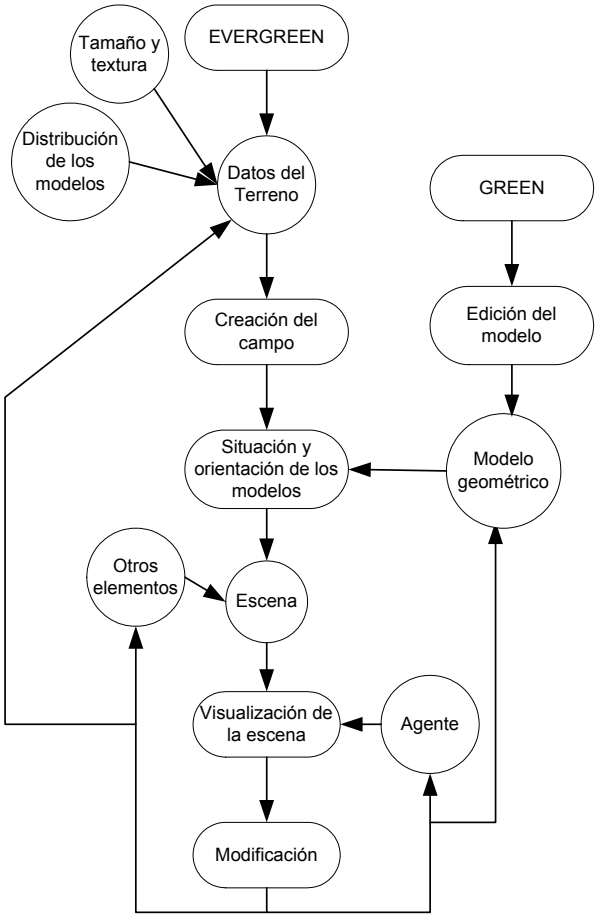


Figura 4.2: Esquema general de funcionamiento del EVERGREEN

4.2 Especificación del terreno

El modelado de una escena natural comienza con la especificación del terreno donde se va a situar la escena. El objetivo de este paso consiste en determinar los datos referentes a la elevación, orientación local y características adicionales del terreno. Dependiendo del realismo que se le quiera conferir a la aplicación estas características tendrán un mayor o menor grado de complejidad. Así, será posible determinar el contenido de agua, de nutrientes y de luz de cada una de las partes del terreno para determinar el vigor de las plantas que crecen en cada punto del mismo.

El terreno puede representar zonas reales, extrayendo la información de los sistemas de información geográfica disponibles o puede realizarse de forma sintética. Entre las técnicas más utilizadas para la obtención de terrenos sintéticos destacan: mapas de altura pintados a mano [Will91], métodos de generación de terreno fractal [Musg89] y modelos basados en la simulación de la erosión del suelo [Musg89].

En nuestra aplicación no se han desarrollado técnicas especiales para la especificación del terreno, ya que el objetivo principal de la misma es la visualización interactiva con un cierto grado de realismo, y no el realismo en sí. Por lo tanto se ha considerado innecesario la adición de elementos que puedan incrementar el tiempo necesario para la visualización.

El terreno está formado por una malla de cuadriláteros, a la que se le aplica una textura que puede ser elegida por el usuario, para obtener distintos tipos de terreno. A esta malla se le puede aplicar un mapa de alturas de modo que el terreno no aparezca completamente plano. El cálculo de la altura de cada uno de los vértices de la malla se realiza de manera aleatoria, con una pequeña variación uniforme, ya que el terreno que se desea visualizar es un campo de árboles frutales.

4.3 Modelado e instanciación de las plantas

Los modelos geométricos que representan plantas suelen tener un gran tamaño. Por ejemplo, un fichero que almacene un árbol con su geometría totalmente detallada puede ocupar hasta 10 Mbytes, por lo tanto una escena con un millar de estos elementos necesitará 10 Gbytes para ser almacenada. Debido al gran tamaño de las escenas, estas no se pueden almacenar completamente en la memoria del sistema. Una primera reducción consiste en utilizar niveles de detalle, simplificando la geometría de los objetos que ocupen poco espacio en la pantalla, sin embargo con esto no es suficiente.

La instanciación [Suth63] se ha utilizado de forma muy extendida en este tipo de problemas. De acuerdo con este paradigma, los objetos geométricos que son transformaciones afines idénticas se convierten en instancias de un único objeto. Sin embargo, en este tipo de escenas donde existe gran cantidad de elementos se puede extender la instanciación a aquellos elementos que son parecidos a otros, aunque no necesariamente iguales. De modo, que un conjunto de plantas similares se representan mediante instancias de una planta representativa. También es posible crear instancias de los elementos que forman las plantas: hojas, ramas, frutos y flores.

Brownbill [Brown96] realiza un estudio de la instanciación aproximada y analiza los resultados que obtiene entre el tamaño de la geometría utilizada y la distorsión percibida entre la imagen original y la obtenida tras realizar la instanciación, obteniendo reducciones de hasta 50 veces sin que el impacto visual sea relevante. También Smith [Smit84] observó que el

conjunto de números aleatorios utilizados para generar una montaña fractal o árboles generados mediante sistemas de partículas se podían reducir a unos cuantos valores representativos sin que afectara de forma significativa a la complejidad percibida en las imágenes generadas.

Los modelos utilizados en la aplicación han sido diseñados mediante la aplicación GREEN, que ha sido objeto de estudio en el capítulo 3. A los modelos diseñados se les pueden aplicar variaciones aleatorias para obtener individuos diferentes de la misma especie, tan sólo aplicando la derivación del modelo tantas veces como se desee. También es posible obtener un individuo en diferentes fases de desarrollo en función del número de iteraciones que se apliquen al sistema. Una vez se obtiene el modelo final, se puede utilizar un fichero para almacenar la geometría del mismo, para ser utilizado en la navegación interactiva. El formato de fichero requerido es el mismo que se utiliza para visualizar el árbol mediante POVRay, y que se comenta en la seccion 3.4.1.

En la aplicación desarrollada se utilizan unos pocos individuos diferentes que se repiten en la escena aplicándoles transformaciones, traslaciones para situarlos en la posición que van a ocupar en el terreno y rotaciones y escalados para modificar la orientación y el tamaño de los modelos. Las rotaciones permitirán que al visualizar desde un único punto de vista distintas instancias de un mismo individuo con diferentes orientaciones éste parezca diferente, ya que la parte del árbol que será visible será distinta.

4.4 Distribución de los árboles

La tarea de la distribución de la población sobre el terreno se puede realizar utilizando diferentes métodos que ofrecen distintos grados de control del usuario, de tiempo necesario para especificar la distribución y de la validación biológica de los resultados. Las técnicas de distribución se dividen en las basadas en el individuo y las de ocupación del espacio [Gree97].

Las técnicas de ocupación de espacio describen la distribución de las densidades de una especie de planta sobre el terreno. En síntesis de imagen, esta distribución se puede obtener utilizando dos técnicas:

- Especificación explícita: La distribución de las densidades de las plantas se mide en el campo (contando las plantas que ocupan unos puntos de muestra) o creadas de forma interactiva, mediante un programa de dibujo.
- Generación procedural: Las distribuciones de las densidades de las plantas se obtienen simulando interacciones entre poblaciones de plantas utilizando modelos ecológicos, tales como autómatas celulares [Gree97] o procesos de reacción-difusión [Higg96].

Las técnicas basadas en el individuo proporcionan la situación y los atributos de cada planta individual. También se distinguen dos aproximaciones:

- Especificación explícita: Las posiciones y atributos de la planta representan los datos del campo, por ejemplo representando una escena real o especificándolos el usuario de forma interactiva.
- Generación procedural: Las posiciones y atributos de las plantas se obtienen utilizando un modelo de generación de patrones de puntos [Wu97], que crea una distribución de puntos con las propiedades estadísticas deseadas o utilizando un modelo de población basado en el individuo que se aplica para simular las interacciones entre las plantas de un ecosistema [Sorr93].

En EVERGREEN la distribución de las plantas se realiza utilizando una técnica de generación procedural basada en el individuo, ya que en un campo de árboles frutales, éstos se distribuyen de forma matricial, ocupando toda la extensión del terreno. Es posible aplicar una variación aleatoria para romper la uniformidad de la matriz.

4.5 Definición del entorno

A la hora de visualizar la escena se deben tener en cuenta dos factores importantes, el primero es, sin duda, acometer la complejidad de la escena, y el segundo simular la iluminación, los materiales, los efectos atmosféricos y el entorno en el que se desarrolla la escena. Este segundo factor es muy importante si se desea incrementar el realismo de la escena. En todas las aplicaciones en las que se necesitan los entornos naturales, ya sean de realidad virtual, de juegos, de simulación, de representación de sistemas de información geográfica o representaciones artísticas, es necesario obtener un cierto grado de realismo. Sin embargo en función de la aplicación se deberá definir el grado de realismo que se desea obtener. No será posible obtener el mismo realismo en la obtención de una imagen fija o una animación para un sistema de información geográfica que en una navegación interactiva de ese mismo sistema. En los sistemas más complejos es posible definir todo tipo de elementos adicionales como cielo, montañas, ríos, lagos y elementos artificiales como carreteras, caminos o casas.

En la aplicación desarrollada se permite, además del suelo, la definición del cielo, el muro y efectos de niebla. El cielo consiste en una esfera que envuelve completamente el terreno que forma la escena. A esta esfera se le aplica una textura que puede ser seleccionada por el usuario. Rodeando el terreno se puede definir una textura para el muro de separación. También es posible conseguir efectos de niebla en la escena, que el usuario puede configurar.

4.6 Visualización interactiva de la escena

La visualización interactiva del campo es la parte más importante de la aplicación y a la que se le va a prestar más atención ya que las principales aportaciones de la tesis se encuentran en la gran mejora que se obtiene en los ratios de imágenes por segundo conseguidos, aplicando las técnicas desarrolladas y que se comentarán en el próximo capítulo.

La estructura de datos del campo que se debe visualizar está formada por los siguientes elementos:

- Un conjunto de árboles: en realidad son instancias de un único árbol formado por un conjunto de ramas y de hojas
- Un terreno: es una malla de cuadriláteros, plana o con un mapa de alturas asociado, a la que se le aplica una textura repetida. Además, se define la textura de los muros de separación
- Un ambiente: contiene la esfera y la textura que representa al cielo y los parámetros de la niebla y las fuentes de luz utilizadas
- Un agente: es el observador y está definido por una posición y una visual. El observador puede navegar libremente por el terreno, teniendo todas las funciones que tendría una cámara de un robot.

La visualización de todos los elementos que forman el terreno y el ambiente se realiza utilizando polígonos texturados. Estos elementos no suponen un incremento de carga de visualización significativa, debido a que el número de polígonos necesarios para su

representación comparado con el número de polígonos utilizados en los árboles, es muy pequeño. Por lo tanto, es en este último aspecto donde se debe realizar el mayor esfuerzo a la hora de obtener simplificaciones.

Las estructuras de datos utilizadas para representar a los árboles, son dos listas de elementos. La primera para representar las ramas, y la segunda para representar las hojas. Cada segmento de rama se visualiza como un cilindro y cada hoja como un polígono.

Las primeras pruebas se realizaron utilizando un algoritmo de fuerza bruta, obteniendo unos resultados bastante malos, como era de esperar por otra parte. Con escenas formadas por menos de ocho árboles se obtenían una par de imágenes por segundo como máximo. Por ello, era necesaria la utilización de técnicas de aceleración. Se han desarrollado dos técnicas conocidas como visualización por selección y nivel de detalle, dejando para una versión posterior el desarrollo de técnicas más avanzadas que permitan incrementar de forma significativa la aceleración de la visualización de la escena.

4.6.1 Visualización por selección

Uno de los principales problemas que se presentan a la hora de visualizar una escena con gran cantidad de geometría es el cuello de botella que se forma en el hardware gráfico debido a que se envía un mayor número de polígonos de los que es capaz de visualizar. Por lo tanto una forma sencilla de acelerar la visualización de una escena es no enviar al visualizador aquella información que quede fuera del campo de visión del observador.

La técnica consiste en seleccionar aquellas primitivas que están dentro del campo de visión y enviarlas al hardware gráfico. El problema principal reside en determinar qué primitivas están dentro del campo de visión, y en realizar este cálculo de la forma más rápida posible.

En el problema que estamos abarcando, es decir la visualización de un conjunto de árboles distribuidos de forma matricial, se puede hacer uso de la coherencia y utilizar los puntos de la matriz para determinar si un árbol está o no dentro del campo de visión. Para ello se deberá tener en cuenta la posición del punto de vista *PV*, la dirección de visualización *V*, el campo de visión *CV*, la razón de aspecto *R* y la posición del árbol que se va a visualizar *P*.

El ángulo de visión horizontal θ de una cámara sintética, se puede calcular en función del ángulo vertical o campo de visión y la razón de aspecto de la ventana de proyección, de modo que $\theta = 2*atan(R*tan(CV/2))$. El punto donde se va a colocar un árbol no estará dentro del campo de visión si el ángulo que forma el vector que une el punto de vista con ese punto, *PV-P*, con la dirección de visualización V es mayor que $\theta/2$, como se puede observar en la figura 4.3. Por lo tanto, calculando el producto escalar entre los vectores unitarios en esas dos direcciones es suficiente para determinar si el punto *P* está dentro del campo de visión o no lo está.

Esta técnica a la vez de sencilla es muy eficiente de calcular, pero presenta un problema. Como el único punto que se calcula para determinar si un árbol se visualiza es aquel sobre el que se planta, en el momento que este punto sale del campo de visión el árbol desaparece de la visualización. Del mismo modo, si este punto entra en el campo de visión el árbol se visualiza. Esto lleva a que los árboles aparezcan o desaparezcan de forma repentina cuando se navega por la escena. Este problema se podría solucionar utilizando los cuatro puntos extremos de la cara que hay sobre el terreno de la caja de inclusión. Esto supondría un incremento en los cálculos necesarios para determinar la selección, pero se solucionaría el problema de la aparición o desaparición repentina de árboles.

Por otra parte el método va a permitir que el número de imágenes por segundo que se van a generar no dependa del número de árboles que forman la escena, sino de los que están dentro del campo de visión.

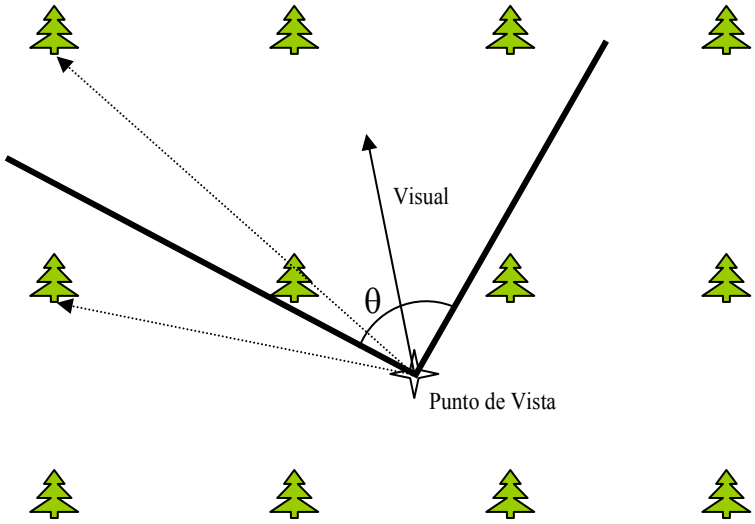


Figura 4.3: Selección de los elementos que están dentro del campo de visión

4.6.2 Niveles de detalle

Una vez se ha conseguido simplificar la cantidad de geometría que se envía al hardware gráfico, se pueden aplicar técnicas que permitan simplificar la geometría que se va a visualizar para ello se ha elegido la técnica de niveles de detalle. Si para representar un objeto, en lugar de utilizar una única representación, se utilizan un conjunto de ellas, tenemos lo que se denominan *niveles de detalle*. Cada una de estas representaciones utiliza un menor número de polígonos.

Los niveles de detalle empezaron a utilizarse con el objetivo puesto en aumentar las prestaciones del sistema gráfico. Por ejemplo, resulta absurdo representar un objeto con miles de polígonos si éste se encuentra muy alejado del observador. En este caso sería conveniente utilizar en su lugar una representación del objeto formada por pocos polígonos, liberando al hardware gráfico de procesar esos miles de polígonos. A medida que la distancia a dicho objeto vaya disminuyendo, se puede ir aumentando el nivel de detalle hasta llegar a la mejor representación.

La visualización en tiempo real se podría definir de forma sencilla como la capacidad de generar entre 20 y 30 imágenes por segundo, cantidad esta suficiente como para que el ojo humano no perciba ningún tipo de parpadeo. El disponer de niveles de detalle de los objetos que integran la escena, permite balancear la carga de forma que de acuerdo a las prestaciones del hardware que se utilice, se pueda mantener la tasa de imágenes por segundo adecuada.

En la figura 4.4 se muestran 3 niveles de detalle del modelo de un conejo intentando simular el nivel de detalle a utilizar según la distancia al observador.

Los distintos factores que se han utilizado [Krus97] para realizar la selección de un determinado nivel de detalle son los siguientes:

- Distancia: Dado un objeto, respecto del observador, a mayor distancia menor será su tamaño. Para hacer una rápida estimación se suelen englobar a los objetos con cajas o esferas.

- Incidencia: Ciertos objetos, según el ángulo de la vista, pueden verse de formas distintas. Por ejemplo una puerta vista de perfil, aún estando cerca del observador, necesitaría muy pocos polígonos para representarla.
- Movimiento: Un objeto animado no es necesario que se represente con grandes detalles ya que estos apenas serán perceptibles.
- Visión Periférica: Dibujar con más detalle aquellos objetos que están en el centro del campo de visión y con menos detalle aquellos que quedan alrededor de dicho centro.
- Tarea: Utilizar niveles de detalle altos en aquellos objetos que se están utilizando para hacer una determinada tarea, y niveles bajos para aquellos objetos que no se utilicen.
- Silueta: Utilizar una mayor resolución de la malla en las partes extremas, que definen la silueta del objeto modelado.

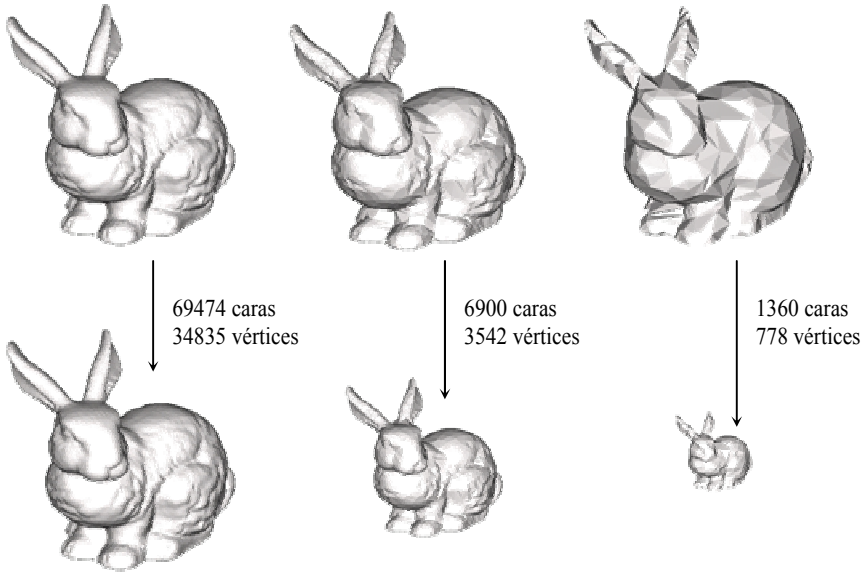


Figura 4.4: Tres niveles de detalle del modelo del conejo.

Cuando se está visualizando un objeto, y mediante un criterio de selección se determina que se ha de cambiar de nivel de detalle, es posible percibir un salto entre la representación anterior y la nueva. Este problema ha sido tratado y su solución pasa por una de entre las dos siguientes:

- Transparencia: Además de visualizar el nivel que toque, dibujar de forma transparente el nivel anterior (el que se estaba visualizando) a fin de que la mezcla elimine la percepción del salto.
- Morphing: Con este nombre se denominan las técnicas que son capaces de convertir un objeto en otro de forma suave.

La transición entre niveles de detalle es uno de los principales problemas con el que se enfrenta el uso de esta técnica. El hecho de almacenar sólo un número determinado de niveles produce que en la visualización se aprecien saltos cuando un objeto pasa de un nivel a otro, y además que no siempre sea posible adaptar la resolución correcta al tener un número discreto de representaciones.

El coste espacial es elevado debido a tener que almacenar de forma independiente todos y cada uno de los niveles de detalle. Esto obliga a que el número de niveles sea pequeño, normalmente entre 5 y 10.

Los niveles de detalle que se utilizan en la aplicación se refieren a la forma en la que se representan las ramas de los árboles en función de la distancia del árbol al observador. Para determinar la precisión con la que se van a visualizar las ramas se hace uso de la distribución matricial de los árboles. Si el observador está situado en la fila *i* y columna *j,* para un árbol que ocupa una *fila* y *columna* determinada, se calculará la precisión como diferencia entre la precisión máxima y la mayor distancia de fila o de columna entre la posición del observador y la situación del árbol, como se puede apreciar en la ecuación 4.1.

Precisión = PRECISION_MAX - máximo (absoluto (i - fila), absoluto (j - columna)) *(4.1)*

En la precisión máxima el cilindro que representa la rama estará formado por 12 puntos. La resolución de los cilindros se irá disminuyendo en función la precisión con la que se dibuje, hasta llegar al mínimo nivel de detalle donde los cilindros estarán representados mediante líneas.

4.7 Resultados

La aplicación desarrollada permite la visualización en tiempo real de unas decenas de árboles distribuidos en forma matricial, incluyendo un suelo texturado, un muro y una esfera envolvente con una textura para simular el cielo. Todos estos elementos son configurables y pueden ser cambiados en tiempo real por el usuario, así como el número de filas de árboles y el número de árboles por fila. En la figura 4.1 se pueden apreciar las distintas opciones disponibles en la aplicación.

En las pruebas realizadas utilizando los algoritmos de selección y de nivel de detalle comentados anteriormente se obtienen menos de 10 fps en escenas formadas por 50 árboles. Cada árbol está formado por 3072 hojas y 1023 cilindros, lo que hace un total de 30696 triángulos. Por este motivo, se decide estudiar alternativas que permitan la visualización de escenas formadas por cientos de árboles. Las técnicas desarrolladas se describen en el capítulo siguiente.

A continuación se hace un estudio de otras aplicaciones que permiten la visualización de entornos naturales en tiempo real.

4.8 Otras aplicaciones

Existe un gran número de aplicaciones donde es necesaria la visualización de vegetación en tiempo real. Las más importantes son aplicaciones de realidad virtual, de simulación, juegos y de representación de los datos de un sistema de información geográfica. La mayoría de estas aplicaciones prestan poca atención a la representación de los árboles y utilizan técnicas de *billboard*, es decir, texturas fotorrealistas transparentes que giran para encararse en la dirección de observación. También se pueden aplicar las texturas transparentes a dos polígonos cruzados. Estas dos técnicas se observan en la figura 4.5.

Otras aplicaciones, sobre todo juegos, utilizan técnicas intermedias entre la utilización de uno o dos polígonos y toda la geometría. Una de las técnicas más utilizadas en los juegos de multijugador on-line son las plantas modeladas a mano. Para la creación de los modelos se

necesita un artista 3D que crea cada uno de los árboles de la escena de forma individual. Generalmente, un árbol está formado por el tronco modelado por tubos y cada rama se compone de unos pocos polígonos utilizados como *billboards*.

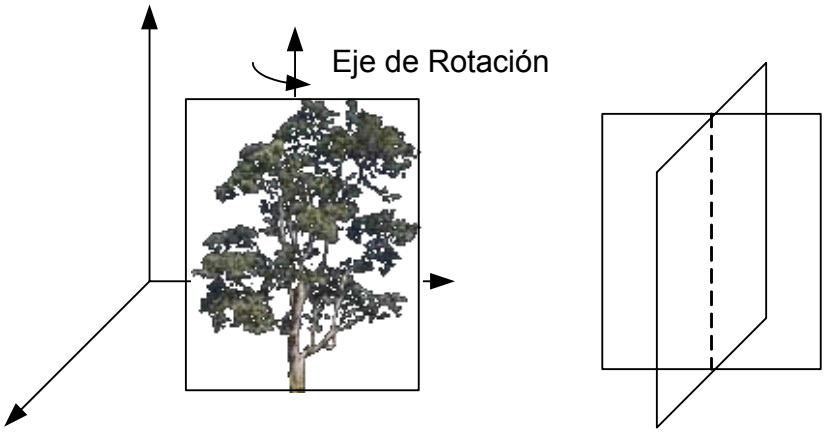


Figura 4.5: *Billboard* de un polígono giratorio y de dos polígonos en cruz.

Algunas aplicaciones tan solo permiten una visualización realista del terreno, para obtener imágenes fijas que pueden tardar varios minutos en generarse. Como por ejemplo Envision (forsys.cfr.washington.edu/envision.html) del servicio forestal de los Estados Unidos, (Pacific Northwest Research Station) que se puede apreciar en la figura 4.6.

Otras aplicaciones realizan una visualización poco realista de los elementos vegetales, ya que su principal objetivo consiste en el modelado del crecimiento de un bosque, utilizando como entrada las medidas reales tomadas en la naturaleza. En la figura 4.7 se puede observar una imagen de la aplicación TreeView desarrollada en la Technical University of Munich (www.wwk.forst.tu-muenchen.de/info/vis/3D_html).

Entre las aplicaciones comerciales que permiten la visualización de gran cantidad de vegetación en tiempo real cabe destacar el sistema Blueberry 3D (www.blueberry3d.com). Este sistema tiene un coste de unos 2500 euros. La aplicación permite visualizar un gran número de plantas, se pueden utilizar unas seis especies diferentes y se visualizan mediante *billboards* si están lejanas o mediante modelos con unos pocos polígonos cuando el observador se aproxima a ellas. Los arbustos aparecen de repente y en la mayoría de los casos los *billboards* son muy diferentes del modelo poligonal. De modo que cuando el observador se va acercando a una planta el *billboard* que la representa desaparece de forma gradual y se transforma en una planta diferente. La técnica de *morphing* utilizada causa problemas con otros objetos de la escena. En la figura 4.8 aparece una imagen de la aplicación.

Otra aplicación que permite la visualización de un buen número de plantas es TreeFX, de la que se ha podido obtener poca información ya que está en desarrollo (www.3dpipeline.com). En realidad no es una aplicación propiamente dicha sino que es un motor de rendering de tiempo real que se distribuye con una librería de árboles y tipos de terreno. La técnica que utiliza es la de *billboards* para cada grupo de ramas. Utiliza técnicas de mezclado para suavizar el cambio entre distintos grupos de texturas. La visualización cercana de los elementos pierde bastante realismo. Si se examinan las plantas desde arriba o desde abajo se descubren las texturas planas que se utilizan como impostores de las ramas. En la figura 4.9 se puede observar una vista de la aplicación.

Por último, resaltar la aplicación desarrollada por Jakulin [Jaku00] cuya técnica fue comentada en la sección 2.5.3. y que se conoce como visualización mediante mezcla de rodajas. En la figura 4.10 aparece una muestra de la misma.

Otras aplicaciones como AMAP (www.cirad.fr) están dedicadas a la obtención de imágenes realistas, pero que no permiten la navegación en tiempo real de los entornos editados. Entre las opciones que pueden incluir este tipo de aplicaciones son las de generar animaciones, simular ecosistemas o jardines o simulaciones biológicas, como por ejemplo incendios de bosques y su reforestación. El precio de este tipo de aplicaciones suele superar los 1.500 €.

Se puede obtener más información referente a la edición y visualización de entornos naturales en www.vterrain.org.

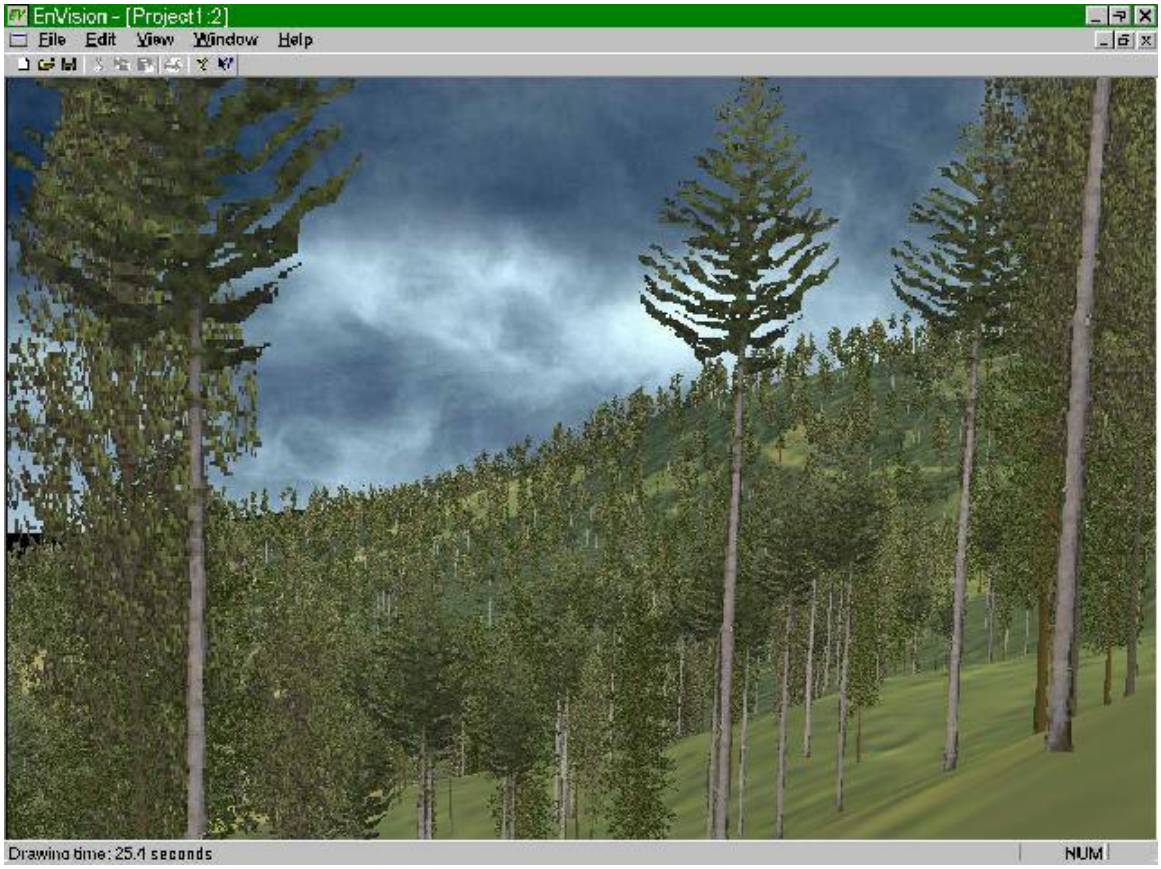


Figura 4.6: Sistema EnVision desarrollado por el departamento forestal de EEUU

Figura 4.7: Visión general del sistema TreeView

Figura 4.8: Sistema BlueBerry3D

Figura 4.9: Vista del sistema TreeFx

Figura 4.10: Vista de la aplicación de visualización mediante mezcla de rodajas

Capítulo 5
Técnicas multirresolución para la aceleración de la navegación interactiva de entornos naturales

Tras el desarrollo de la aplicación EVERGREEN y a la vista de los resultados obtenidos se considera imprescindible el desarrollo de otra serie de técnicas que permitan la aceleración de la visualización de la escena natural. Estas técnicas van a formar el cuerpo principal de la tesis y por lo tanto son aportaciones originales. Las técnicas desarrolladas van a abordar el problema desde tres puntos de vista distintos: las ramas, las hojas y el árbol en su totalidad.

Para conseguir una simplificación de las ramas desde el punto de vista geométrico se ha desarrollado un algoritmo que permite que la interpretación de la cadena del lenguaje, que representa a las ramas del árbol, tenga como resultado una malla de polígonos que puede ser simplificada en función de la distancia del observador. También, se ha desarrollado una técnica que a partir de una especificación procedural de las ramas del árbol, es decir una cadena del lenguaje, permite obtener distintos niveles de detalle.

Para las hojas se ha desarrollado una técnica para la aceleración de su visualización que consiste en obtener imágenes precalculadas de grupos de hojas que se utilizan para sustituir a las mismas en el proceso de visualización, reduciéndose drásticamente el número de polígonos necesario para representar las hojas. El algoritmo proporciona distintos niveles de detalle del follaje del árbol.

Por último, se ha desarrollado un algoritmo para englobar las técnicas anteriores que sirva para obtener distintos niveles de detalle de un árbol completo, es decir, un modelo que tenga en cuenta a las ramas y a las hojas simultáneamente. Este modelo podrá ser transmitido de forma incremental.

A continuación, se presenta el método que permite representar las ramas de un árbol mediante una única malla poligonal, solucionando los problemas de visibilidad y continuidad que aparecen cuando las ramas se representan mediante uniones de primitivas. En la segunda sección se formula el modelo de multirresolución procedural, que modela de forma multirresolutiva una estructura ramificada mediante una cadena de símbolos paramétricos. En el siguiente apartado se presenta el modelo de representación del follaje de un árbol mediante impostores. En la sección cuarta, se explica como obtener un modelo conjunto que represente tanto las ramas como las hojas del árbol. Por último, se explican los aspectos más importantes a tener en cuenta cuando se introduce el modelo en una población de árboles.

5.1 Representación de una estructura ramificada mediante una malla poligonal

Como se ha descrito en la sección 2.1 de la presente tesis existe una gran variedad de modelos para la generación de árboles, sin embargo ninguno de ellos crea una estructura geométrica homogénea para visualizar los objetos generados. El resultado final de los métodos estudiados son uniones de distintos modelos geométricos que representan cada rama o segmento de rama. Las primitivas más utilizadas son: conos truncados, cilindros generalizados o mallas de polígonos para cada rama.

La utilización de un conjunto de primitivas para representar ramas curvadas puede generar discontinuidades, también existen problemas a la hora de representar las bifurcaciones ya que se solapan los polígonos que representan distintas ramas con los correspondientes problemas de visibilidad (véase figura 5.1). Es por ello que se plantea la obtención de una única malla de polígonos para representar la estructura del árbol [Lluch01a],[Lluch01b].

La obtención de una malla única permitirá la utilización de un método de simplificación estándar para disminuir el número de polígonos que la forman. Por otro lado, será posible crear un método de simplificación que se adapte a la estructura ramificada que se desee representar, en el que la simplificación de los polígonos que forman parte de las ramas de niveles superiores sea prioritaria.

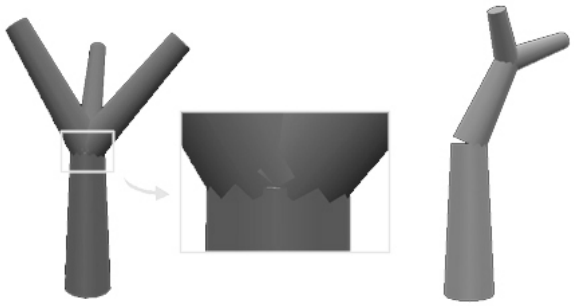


Figura 5.1: Problemas de visibilidad y continuidad resueltos con la malla única

5.1.1 Algoritmo

La construcción de un árbol formado por una única malla de triángulos a partir de la interpretación de una cadena resultante de la derivación de un Sistema-RL de generación de especies vegetales, atraviesa tres fases:

- La fase de representación del árbol.
- La fase de obtención del modelo poligonal.
- La fase de refinamiento de los nudos del árbol.

El resultado de esta última fase será una única malla de triángulos 3D que representará el árbol construido con el Sistema-RL.

La representación del árbol

La interpretación de la cadena se realiza mediante una modificación del intérprete desarrollado en GREEN (sección 3.4), de modo que la orientación de la tortuga puede variarse mediante operaciones de giro sobre cada uno de los ejes ortogonales 3D asociados a la tortuga, mientras que el movimiento siempre se genera en la dirección marcada por el eje local de la tortuga H (*heading*). En cada movimiento, la tortuga crea un contorno (elíptico o circular) en

la posición inicial y otro en la posición final. El tamaño y orientación de dicho contorno dependerá del grosor e inclinación de la rama que se esté construyendo en ese momento.

Para el almacenamiento de los contornos que va generando la tortuga, se utiliza una estructura jerárquica de tipo árbol. Cada nodo de dicho árbol estará compuesto por los siguientes elementos (ver figura 5.2):

- el correspondiente contorno generado por la tortuga,
- una matriz de transformación que contendrá las operaciones de rotación y traslación que permitieron a la tortuga pasar del nodo anterior (nodo padre) al actual,
- un puntero al siguiente nodo hermano,
- un puntero al primer nodo hijo y
- un puntero al nodo padre.

El nodo padre es aquel del cual partió la tortuga (punto de inicio del movimiento) para poder llegar hasta el nodo actual (punto fin de movimiento). Todos los nodos hijos de un mismo padre están unidos mediante una lista enlazada en orden de su generación. Un nodo padre tiene únicamente enlace directo con el primer nodo hijo que engendró (Figura 5.2).

Partiendo de esta estructura jerárquica obtenida a partir de la interpretación de la cadena de entrada, se generará una malla de triángulos que respetará la forma de los contornos y la topología establecida por la estructura obtenida en esta fase.

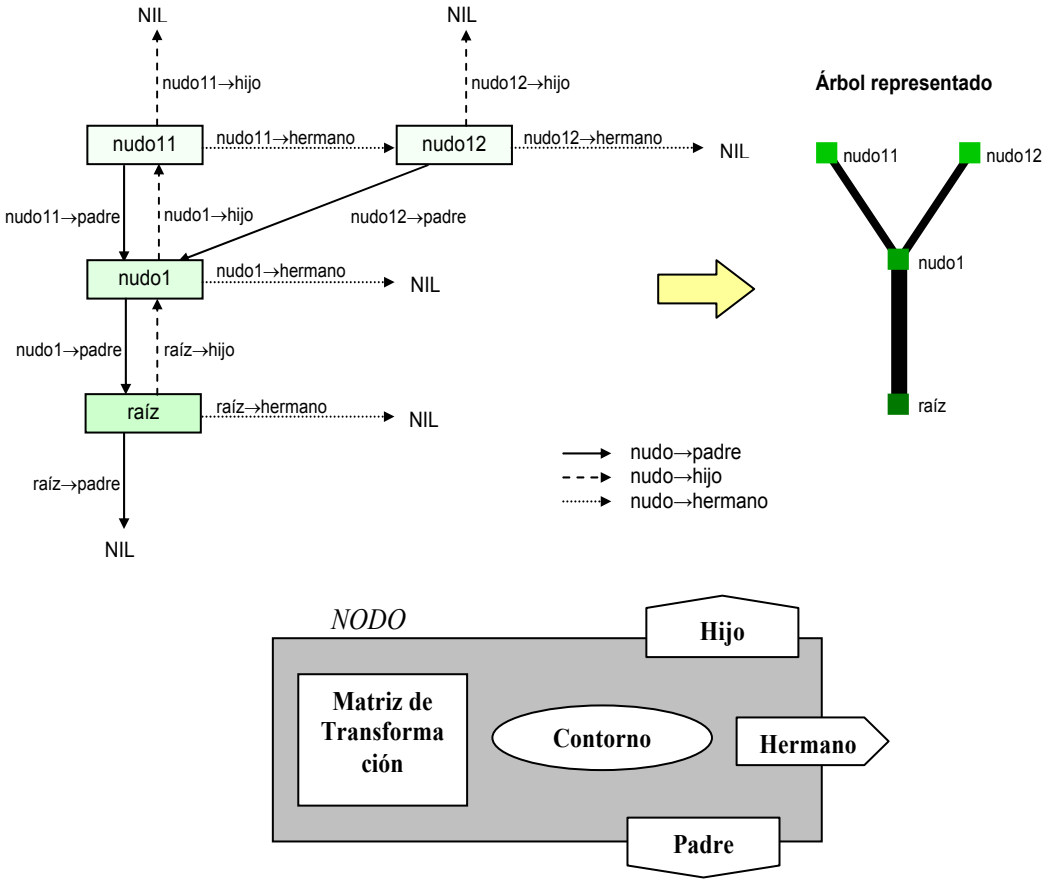


Figura 5.2: Enlaces y contenido de un nodo.

Obtención del modelo poligonal

Para la construcción del modelo poligonal se utiliza una librería desarrollada por nuestro grupo de investigación [Fern99] que permite la obtención de una malla de triángulos a partir de una secuencia de contornos. La reconstrucción de superficies a partir de un conjunto de contornos planos es un problema que se intenta resolver desde los años 70. El interés que despierta el desarrollo de este tipo de algoritmos se debe a las aplicaciones que tienen tanto en ingeniería (Diseño Asistido por Ordenador) como en medicina. Es este último campo el que más ha potenciado su desarrollo con la aparición, a principio de los años 70, de instrumentos de exploración no invasivos como la resonancia magnética (MRI) o la tomografía axial computerizada (TAC).

La librería desarrollada no sólo ha resuelto los problemas que, en este tipo de reconstrucción, identificaron Meyers y Skinner [Meye92] sino que, además, mejora la eficiencia de otros métodos de reconstrucción publicados anteriormente [Choi94][Fuch97][Meye92][Xu96]. La librería de reconstrucción se aplica en la generación de mallas de triángulos entre pares de secciones consecutivas. En el caso del algoritmo que se presenta, la sección inferior siempre estará constituida por un único contorno que formará parte de un nodo padre. Por otro lado, la sección superior estará formada por todos los nodos hijo del nodo padre actual. La construcción de las secciones y el resultado del mallado se pueden observar en la figura 5.3.

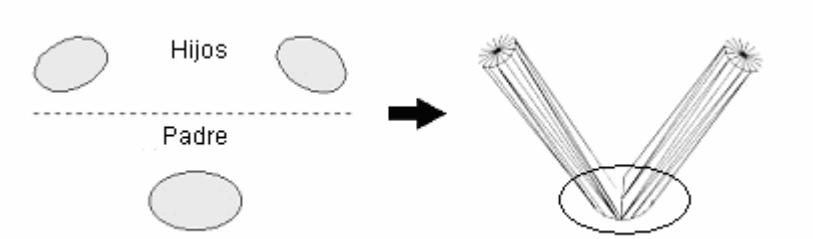


Figura 5.3: Generación de la malla de triángulos entre dos secciones adyacentes.

Como muestra la figura 5.3, la aplicación directa de la librería de reconstrucción para la generación de la malla de triángulos a partir de la estructura generada en la fase de representación del árbol, presenta una geometría visualmente errónea en el punto de unión con el nodo padre. Para mejorar este resultado se realiza un refinamiento de la estructura como paso previo a la obtención del modelo poligonal.

Se ha desarrollado un algoritmo denominado *refinado por intervalos* que va a permitir corregir los errores geométricos que aparecen en los puntos de unión de las ramificaciones (nudos del árbol). Este método permite añadir un conjunto de nuevos contornos, equiespaciados en altura, a la estructura entre el nodo padre y los hijos. Estos contornos se añaden desde el punto de unión de las ramificaciones hasta que éstas se separan completamente (ver figura 5.4).

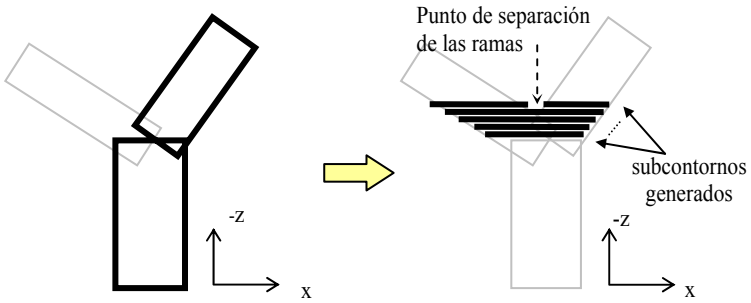


Figura 5.4: Inserción de subcontornos realizado por el método de refinado por intervalos.

Refinamiento de las uniones

Para realizar el refinado por intervalos, se supone que las ramas de los árboles se representan mediante contornos que definen cilindros o conos truncados de altura y anchura iguales a la longitud y grosor de la rama a la que representan. Teniendo en cuenta esta suposición, la generación de los contornos intermedios sigue una secuencia de 3 pasos: cálculo de los parámetros de la elipse, cálculo de un contorno único e inclusión en la estructura de datos.

Hay que puntualizar que todos los cálculos que se realizarán en adelante asumen que el cilindro inicial tiene su base en el plano *XY*, centrado en el origen, y se prolonga en el eje *Z*. El cálculo de los parámetros de la elipse (radio menor y mayor, centro y rotación) que se obtiene al cortar un cilindro mediante un plano horizontal a una altura específica, se va realizar de la siguiente forma. En primer lugar se calculan los radios de la elipse (ver figura 5.5, dibujo a), el radio menor (r_e) de la elipse es igual al radio del cilindro (r_c) y el radio mayor (R_e) se calcula en función del ángulo α que forma la inclinación del cilindro con el eje *X* (5.1).

$$r_e = r_c,\ R_e = \frac{r_e}{sen(\alpha)} \tag{5.1}$$

Seguidamente, para colocar la elipse en su posición correcta se debe realizar una rotación de la elipse y una traslación desde el origen hasta el lugar que le corresponde. Para calcular el centro de la elipse, es necesario determinar la altura del primer contorno. Para ello, se utiliza como incremento (h) la longitud de las aristas que forman el polígono que define el contorno del cilindro que representa a la rama (todas tienen la misma longitud). Este incremento se suma a la altura del cilindro de la rama padre. Las alturas de los siguientes contornos serán equidistantes. El valor del incremento que se utilice puede ser diferente, dependiendo si se quieren obtener un mayor o menor número de subcontornos.

Como se conoce la altura a la cual se va a generar la elipse, se pueden calcular las coordenadas de su centro, suponiendo que la base del cilindro esté situada en el origen (5.2).

$$\left.\begin{aligned} c_x &= s_x + \gamma \cdot v_x \\ c_y &= s_y + \gamma \cdot v_y \\ c_z &= s_z + \gamma \cdot v_z \end{aligned}\right\} \xrightarrow{(s_x,s_y,s_z)=(0,0,0)} \left.\begin{aligned} c_x &= \gamma \cdot v_x \\ c_y &= \gamma \cdot v_y \\ c_z &= \gamma \cdot v_z \end{aligned}\right\} \tag{5.2}$$

El valor de la coordenada c_z es conocido (figura 5.5b), ya que es el incremento h multiplicado por el número del subcontorno que se está generando, se puede despejar γ y obtener el resto de componentes del centro (5.3).

$$c_z = h * n\_subcontorno \rightarrow \ \gamma = \frac{c_z}{v_z} \tag{5.3}$$

El contorno se trasladará al centro calculado anteriormente, pero antes se debe determinar la orientación del mismo. Para rotar la elipse se utilizará el ángulo que forma la proyección del vector director del cilindro sobre el plano XY con el eje horizontal. La traslación se realizará al centro de la elipse calculado anteriormente. En la figura 5.5c se muestra cuál es el ángulo de giro que es necesario aplicar sobre la elipse y que resulta ser (5.4).

$$\vec{v} \bullet \vec{R} = \left\|\vec{v}\right\| \cdot \left\|\vec{R}\right\| \cdot cos(\delta), \delta = arccos\left(\frac{v_x}{\sqrt{v_x^2 + v_y^2 + v_z^2}}\right) \tag{5.4}$$

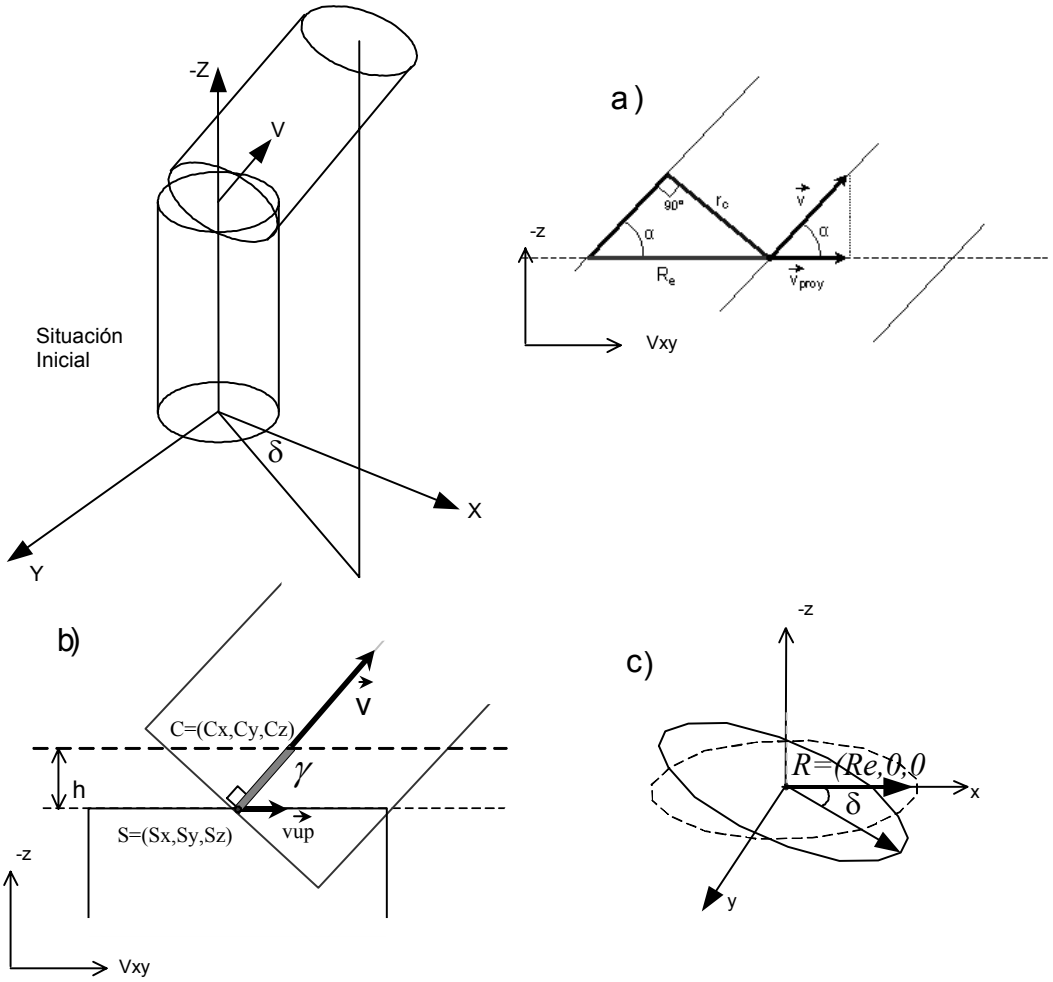


Figura 5.5: Obtención de los parámetros de la elipse.

Obtención de un único contorno

A partir de las elipses que se hayan obtenido es necesaria la obtención de un único contorno, para cada uno de los intervalos fijados desde el nodo padre hasta el punto en el que se separan todas las ramificaciones (ver figura 5.6).

Para la generación del contorno único se hace necesario detectar las intersecciones que se producen entre las elipses. El cálculo de intersecciones resulta más sencillo si se asume que las secciones elípticas están aproximadas por polígonos, definidos por una resolución que determinará el número de vértices que forman dicho polígono. Los pasos a seguir son los siguientes:

- Primero se han de ordenar las secciones por altura de menor a mayor
- A continuación, se busca un punto que pertenezca a una de las elipses y sea externo al resto
- Para la generación del contorno único se necesita una estructura que almacene qué puntos se han visitado durante este proceso
- Además, se asumirá que los vértices de todas las elipses están ordenados en el mismo sentido, horario o antihorario
- Se escoge la primera arista de la elipse actual comenzando por el punto actual. La arista se compara con las aristas del resto de elipses. Si no se detecta ninguna intersección con las mismas, se almacena el punto actual en el contorno único, se

marca como visitado y se pasa a la siguiente arista de la elipse actual. Si se ha detectado una intersección se guarda la información relativa al punto actual, se pasa al siguiente punto, que en este caso, será el punto final de la arista intersectada. Por esto, la elipse actual pasará a ser la elipse que contiene dicha arista (figura 5.7).

Puede ocurrir que se llegue a un punto ya visitado y no haber detectado ninguna intersección. Esto quiere decir que el contorno único sólo lo formará una elipse. Esta situación se da cuando hay elipses contenidas dentro de otras elipses mayores (véase figura 5.8-a), o cuando las elipses se encuentran separadas (véase figura 5.8-b). Cuando todas las elipses estén separadas se terminará el proceso de creación de contornos.

Inserción del contorno en la estructura de datos

Por último, se construye un nodo con el contorno generado en el paso anterior. Éste se añade a la estructura del árbol generada por la interpretación de la cadena de entrada al presente algoritmo.

La cantidad de contornos insertados dependerá exclusivamente de la inclinación de los cilindros. Si la inclinación de éstos es pequeña, respecto al plano definido por el contorno del nodo padre, el método detectará en pocas iteraciones la ausencia de intersecciones entre las elipses y, por tanto, el número de nodos a insertar será menor. En cambio, si los cilindros presentan un ángulo de inclinación grande, el punto de separación estará muy alejado de la base de estos y, por tanto, se tendrán que insertar muchos nodos intermedios para el mismo valor de intervalo.

En el capítulo siguiente se analizarán con más profundidad los resultados obtenidos por el algoritmo, analizando los tiempos y costes de almacenamiento necesarios para las distintas representaciones.

Figura 5.6: Generación de un contorno único a partir de la detección de intersecciones entre aristas.

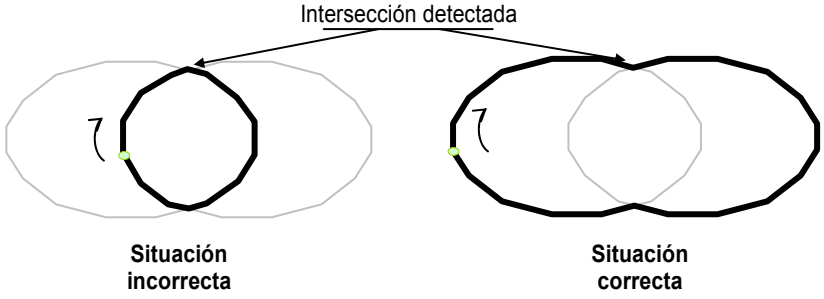


Figura 5.7: El contorno debe comenzar en un punto externo al resto de elipses

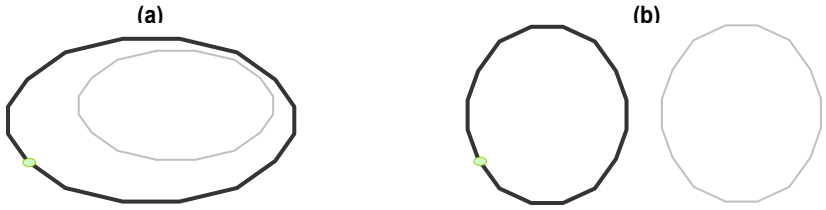


Figura 5.8: Casos especiales a tratar en la generación de un contorno único.

5.2 Modelo multirresolución procedural para la representación de las ramas

Una alternativa a la aplicación de técnicas poligonales de simplificación para la obtención de un modelo multirresolución de las ramas es la *multirresolución procedural* o gramatical. Se quiere obtener una estructura multirresolución de las ramas del árbol transformando la cadena obtenida como derivación del sistema-L en otra que, interpretada adecuadamente, represente niveles de detalle continuos. La cadena obtenida en la derivación ha sido generada siguiendo reglas de crecimiento por lo que no resulta apropiada para la extracción directa de niveles de detalle visuales. No es posible extraer directamente de esta cadena subcadenas cuya interpretación pueda decirse que es un nivel de detalle del árbol completo.

Se plantea pues la búsqueda de un modelo de multirresolución gramatical interpretable proceduralmente y la transformación de la cadena original en este modelo. El modelo debe ser compacto, y el coste de almacenamiento debe ser comparable al utilizado por todas las ramas del árbol sin niveles de detalle. Se quiere, además, que la diferencia entre un nivel de detalle y el siguiente sea una rama, por lo que podemos decir que el modelo será continuo considerando la rama como unidad mínima de representación.

La posibilidad de utilizar las distintas cadenas que se obtienen en el proceso de derivación tampoco conduce al resultado visual perseguido. Si bien cada cadena puede ser interpretada gráficamente sin problemas, su representación secuenciada conduce a fotografías del árbol en distintos grados de crecimiento. Este criterio evolutivo no es apropiado con la idea de disponer de árboles geométricamente más simples cuanto más alejados del observador se hallen. Por ejemplo, la altura real del árbol no se alcanzaría hasta el máximo nivel de detalle, dando la impresión que el árbol "crece" conforme se acerca.

Al igual que sucede con la multirresolución poligonal que extrae un modelo más o menos simple dado un cierto nivel de detalle, la multirresolución procedural extrae una cadena de mayor o menor longitud que representa en mayor o menor medida la cadena original para ese nivel de detalle. La extracción del mayor nivel de detalle debe ser equivalente al modelo original, en este caso la cadena derivada, es decir, la equivalencia debe darse entre las interpretaciones gráficas de las cadenas.

5.2.1 Antecedentes

En la sección 2.4.2 se hace una breve introducción a la multirresolución poligonal, que es la más extendida en el campo en el que se está trabajando. Sin embargo, existen otro tipo de modelos multirresolución como los basados en wavelets o los basados en texturas. Todos ellos tienen en común que el modelo representa a un objeto mediante un conjunto de aproximaciones con diferentes niveles de detalle y permiten recuperar cualquiera de ellos bajo demanda. Hasta donde se conoce, no existen referencias a la multirresolución procedural en el

sentido que aquí se ha definido, compartiendo este modelo la escalabilidad de los anteriores y siendo especialmente apropiado para estructuras ramificadas.

Para conseguir la cadena multirresolución se van a seguir los siguientes pasos:

1. Construcción de una estructura de datos en árbol multicamino a partir de la cadena derivada. Se nombrará a la estructura de datos "a-TAD" para diferenciarlo del árbol vegetal que se intenta graficar.
2. Generación de la cadena multirresolución (multicadena) a partir del recorrido del a-TAD.
3. Determinación del modo de interpretación de la multicadena para la extracción de un nivel de detalle concreto (acceso aleatorio) o del nivel de detalle anterior y siguiente al actualmente representado.
4. Generación del modelo gráfico del nivel demandado.

Cada uno de ellos se describe en detalle en los apartados que continúan.

5.2.2 Estructura de datos intermedia

Para obtener la cadena multirresolutiva va a ser necesario generar una estructura de datos intermedia. La estructura de datos que se va a utilizar es un árbol multicamino por generalidad, aunque en la mayoría de los casos reales se trate de un árbol binario.

Antes de definir qué debe almacenar esta estructura de datos, es necesario plantearse cuales son los objetivos que debe cumplir el modelo. A continuación, se enumeran los requisitos principales que debe reunir el modelo que se va a desarrollar:

- El resultado del proceso debe ser una cadena paramétrica.
- La interpretación de esta cadena generará los distintos niveles de detalle.
- Los niveles de detalle deben ser crecientes, es decir el nivel n+1 se debe obtener a partir del nivel n, de este modo los niveles de detalle podrán ser transmitidos de forma incremental.
- El número de módulos de la cadena paramétrica obtenida no debe ser significativamente mayor que el número de módulos de la cadena de entrada.
- Los primeros niveles de detalle deben ser suficientes para obtener la estructura general de las ramas del árbol modelado, es decir, el árbol se creará de forma balanceada.
- Los sucesivos niveles deben incrementar el detalle del modelo, de modo que la interpretación de la cadena completa genere todas las ramas del árbol.

Como hipótesis, se asume que el sistema-L derivado produce una cadena bien construida. Decimos que está bien construida si la interpretación gráfica de la misma por una tortuga 3D conduce a una estructura ramificada conectada y progresiva (la tortuga no retrocede). Así, la cadena de entrada está formada por símbolos con sentido gráfico: giros, avances, primitivas y símbolos de manejo de la pila; todos ellos bien construidos. Además, se supone que todos los símbolos no terminales han sido eliminados de la misma sin pérdida de generalidad, ya que no aportan nada al modelo al no ser interpretables. Llamaremos a esta cadena original o *cadena de entrada*.

La forma que tendrá una cadena de entrada queda restringida a la definición siguiente:

cadenagráfica = {avance|giro|primitiva}
ramal=cadenagrafica{"["ramal"]"}

cadena de entrada=ramal

donde se ha representado las inserciones en la pila de estado por el símbolo "[" y las de extracción por el símbolo "]". Aunque la definición gramatical de las cadenas de entrada produce símbolos redundantes (inserciones y extracciones innecesarias) resulta lógicamente conveniente la separación de ramales por símbolos de pila. Nótese también que se está considerando gráficamente como rama la cadena gráfica entre dos símbolos de pila. Por tanto, el nacimiento de una rama siempre marca el final de otra ajustándose a la definición dada.

Por ejemplo, supóngase que se tiene la cadena del lenguaje 5.5, donde *Cn* corresponde a la cadena de símbolos gráficos necesaria para generar la rama *n*.

C1[C2[C3[C4][C5[C6][C7]]][C9[C11][C10]][C8]][C12[C13][C14][C15[C16]]] *(5.5)*

Obviamente, la interpretación de la cadena 5.5 puede producir una estructura ramificada como la que aparece en la figura 5.9 pues no es más que una representación lineal de un a-TAD donde los corchetes hacen las veces de paréntesis. Esta estructura de datos será un paso intermedio para la obtención de la cadena multirresolución.

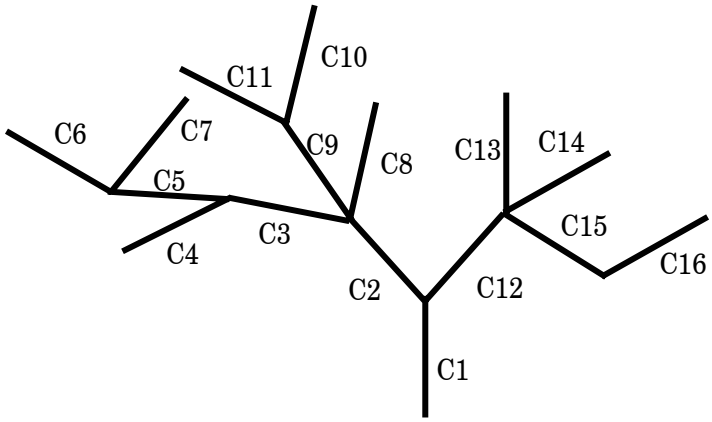


Figura 5.9: Conjunto de ramas obtenido tras interpretar la cadena 5.5

A fin de simplificar el algoritmo de generación del árbol, se sustituyen redundancias en extracciones e inserciones de pila seguidas "][" por un nuevo símbolo cuya interpretación no afecta a la pila "|" cargando simplemente el estado del tope de pila en la tortuga. Así, la cadena 5.5 queda transformada de la manera siguiente:

C1[C2[C3[C4|C5[C6|C7]]|C9[C11|C10]|C8]|C12[C13|C14|C15[C16]]] *(5.6)*

Los nodos (objetos dato en el a-TAD) se van a corresponder físicamente con lo que se ha definido como rama, es decir, los tramos del árbol entre dos bifurcaciones. Las aristas del a-TAD corresponden con las relaciones entre las ramas del árbol vegetal, relaciones que permitirán el recorrido del a-TAD. Los nodos almacenarán la siguiente información (Figura 5.10):

- identificador único del nodo,
- cadena de módulos paramétricos con significado gráfico que permiten representar la rama,
- una medida dada por una métrica aplicada al árbol,
- un estado, que indica si el nodo ha sido incluido en la cadena de salida o no,
- enlaces al padre, al siguiente hermano y al primer hijo,

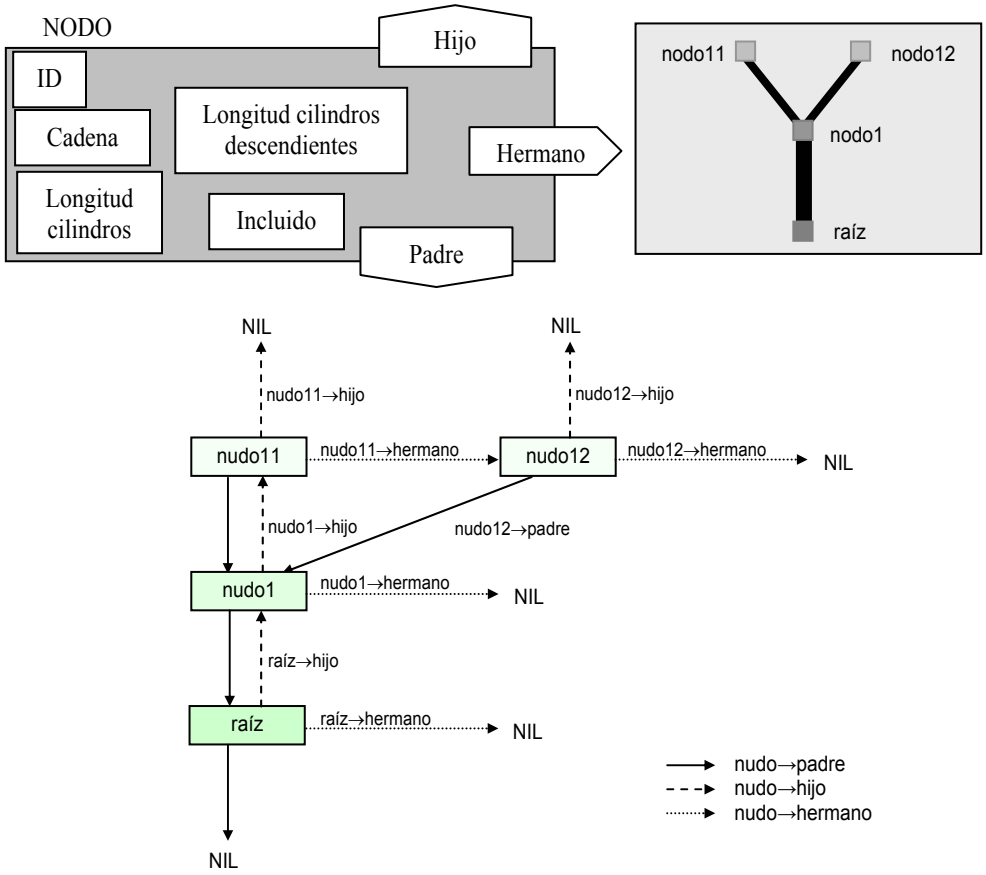


Figura 5.10: Contenido de un nodo y enlaces de la estructura de datos

El interés de asociar una medida a cada nodo del árbol viene dado por la necesidad de establecer un orden en la representación de las ramas. Este orden será, en definitiva, el que se seguirá para construir la secuencia de niveles de detalle. Pueden existir multitud de criterios que establezcan ese orden, sin embargo, el evolutivo no es uno de ellos, lo que justifica la reordenación en el árbol-TAD.

Para establecer el criterio, se atenderá a las definiciones siguientes:

Definición 5.1. Sea C el conjunto de las cualidades atribuibles a una rama como la textura, la distancia al suelo o cualquier otra. Se define $H(C)$ como el conjunto de valores que pueden tomar esas cualidades y $h(c)$ el subconjunto incluido en $H(C)$ de los valores que puede tomar la cualidad c perteneciente a C.

Definición 5.2. Sea A un a-TAD construido a partir de una cadena de entrada. Sea N_i un nodo perteneciente a A. Llamaremos $\{N\}_i$ al conjunto de nodos hijos del nodo N_i y $a(N_i)$ al conjunto de nodos descendientes de N_i.

Obviamente $N_i \cup a(N_i)$ es un a-TAD.

Definición 5.3. Sea N_i un nodo de A. Sean n,m dos nodos cualesquiera de $\{N\}_i$. Diremos que Q_c: $\{N\}_i \Rightarrow h(c)$ es una *métrica* de la cualidad c sobre A si y solo si existe R tal que $Q_c(m) \mathrm{R} Q_c(n)$ siendo R una relación de orden.

Lógicamente, la métrica se convierte en el criterio que va a permitir reordenar la cadena al establecer prioridades en el recorrido del a-TAD. Se pueden pensar en métricas diferentes, aunque las de mayor éxito serán aquellas que atiendan a criterios visuales. Aquí se ha elegido una métrica de densidad de descendientes partiendo del principio de que resulta más urgente el dibujo de aquellos nodos que mayor ramificación presentan en sus descendientes ponderada por la longitud de los mismos.

Definición 5.4. Se define la *longitud de rama* como una cualidad del conjunto *C*. El valor de la longitud de rama *l(N)* se mide por los avances de la tortuga en la cadena asociada al nodo *N*.

Definición 5.5. Sea *N* un nodo de un a-TAD *A*. Se define *longitud del ramal L(N)* como la suma de los valores de las longitudes de cada una de las ramas de *a(N)* más la longitud de la rama asociada a *N*. Es decir,

$$L(N) = l(N) + \sum_{n \in \{N\}} L(n) \qquad (5.7)$$

Teorema 5.1. *L* es una métrica de la longitud de rama sobre *A*.

En efecto, elegidos dos nodos cualesquiera de un nodo no terminal será posible calcular el valor de *L* para cada uno de ellos recorriendo cada subárbol y acumulando el avance de la tortuga, lo que establece una ordenación. En el caso de que el nodo sea terminal no tendrá hijos, por lo que la decisión de qué hijo elegir es innecesaria.

Otras métricas visualmente aceptables son las siguientes:

Longitud del camino máximo $$P(N) = l(N) + \max_{n \in \{N\}}(P(n)) \qquad (5.8)$$

Número de ramas hijas $$B(N) = \mathrm{car}\{N\} \qquad (5.9)$$

Número de ramas descendientes $$R(N) = B(N) + \sum_{n \in \{N\}} R(n) = \mathrm{car}\, a(N) \qquad (5.10)$$

La métrica nos permite almacenar una medida en cada nodo. Esta medida corresponderá a las longitudes de todas las ramas descendientes incluida la del nodo. Tomando la cadena de entrada, el algoritmo que construye el a-TAD con la métrica propuesta es el siguiente:

```
Se crea la raíz del a-TAD y se considera el nodo actual
Se recorre la cadena de entrada módulo a módulo
En caso de que el módulo sea
    módulo gráfico:
        se incrementa la medida en la longitud del avance del nodo actual
        se añade a la cadena del nodo actual
    [:
        se añade un hijo que pasa a ser el nodo actual
    |:
        se incrementa la medida del nodo padre en la propia
        se añade un hermano que pasa a ser el nodo actual
    ]:
        se incrementa la medida del nodo padre en la propia
        el nodo padre pasa a ser el actual
```

En la figura 5.11 se muestra la estructura de datos obtenida al aplicarse el algoritmo a la cadena de ejemplo 5.5. En cada nodo aparece la cadena asociada y la medida aplicando la métrica propuesta, suponiendo que en cada cadena la longitud de la rama es 1.

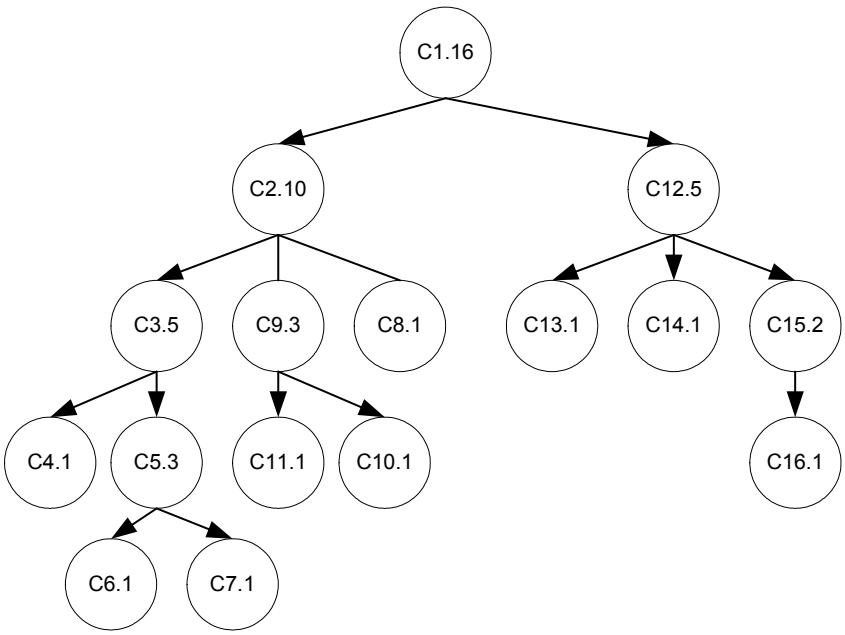


Figura 5.11: Estructura de datos obtenida a partir de la cadena 5.5

5.2.3 Algoritmo para la obtención de la cadena multirresolución

Una vez generada la estructura de datos intermedia, ya es posible analizarla para obtener una cadena multirresolución. Para ello, se realizarán recorridos que comienzan en el nodo raíz y terminan en cada una de las hojas. Cada uno de estos caminos generará un nivel de detalle distinto, teniendo en cuenta que las cadenas que han sido incluidas en un nivel de detalle no podrán repetirse en otro nivel.

La construcción de un nivel de detalle determinado puede empezar en cualquier punto del a-TAD. Como el estado de la tortuga en el punto de comienzo se debe obtener con un acceso directo y no a través de una pila, la estructura de datos que se necesita para almacenar los estados es ahora una lista indexada. Esto quiere decir que para la interpretación de la cadena multirresolución son necesarios dos nuevos símbolos para el manejo del almacén de los estados de la tortuga que sustituyen al de inserción "[" y al de extracción "]". Los dos nuevos símbolos son *SAVE(id)*, que almacena el estado de la tortuga con un identificador *id* y *RESTORE(id)*, que recupera el estado almacenado con identificador *id*. Los identificadores de los estados son únicos.

Teniendo en cuenta todos estos detalles para la creación de una cadena de módulos que contenga los distintos niveles de detalle, será necesario recorrer la estructura generada anteriormente de la siguiente forma:

- La creación del primer nivel de detalle consiste en recorrer el árbol desde la raíz hasta alcanzar una hoja, el criterio de elección del hijo vendrá dado por la métrica elegida. Se deben activar los interruptores de los nodos incluidos en la cadena de salida, para que no se repitan.

- Delante de los módulos de la cadena asociada a un nodo se incluirá un módulo SAVE poniendo como parámetro el identificador del padre del nodo, de este modo se almacenará el estado de la tortuga para poderlo recuperar posteriormente. Sólo será necesario si falta algún hermano del nodo actual por incluir en la cadena de salida.
- Para generar los siguientes niveles se realizará la búsqueda, desde el nodo raíz, del primer nodo no incluido en la cadena de salida, que tenga un valor de la métrica mayor (dependiendo de la métrica podría escogerse el mínimo). El nivel comenzará con un módulo RESTORE, asignando como parámetro el identificador del nodo padre del primer nodo del nivel. Este módulo recuperará el estado de la tortuga que permite continuar la creación del árbol.
- Después de incluir cada nivel se realizará un proceso de actualización de los valores de la métrica de cada nodo en función de los que han sido incluidos en el nivel correspondiente.

El algoritmo tomará como entrada el a-TAD y como salida creará una cadena de módulos multirresolución que incluirá los distintos niveles de detalle. Esta colección de módulos se denomina *multicadena*.

Definición 5.6. Sean dos nodos n,m pertenecientes al a-TAD A tal que m pertenece a $a(n)$. Se define $X(n,m)$ *camino* entre n y m como la colección ordenada de nodos que lleva desde n hasta m siguiendo los enlaces padre a hijo. $X(n,n)=n$.

Lógicamente se cumple que $X(n,m)$ esta incluido en $X(raíz,m)$.

Definición 5.7. Se define la *longitud del camino* $lX(n,m)$ como $lX(n,m) = \sum_{k \in X(n,m)} l(k)$.

El algoritmo extrae caminos que terminan siempre en un nodo terminal. Los caminos están formados por nodos que todavía no han sido extraídos. Cuando todos los nodos terminales han sido extraídos formando parte de algún camino, el proceso de formación de la *multicadena* acaba.

Se ha elegido como métrica la propuesta en (5.7) aunque para cualquier otra se procede de forma similar. Para el caso de la longitud $L(N)$, y para cada extracción de un camino (nivel de detalle), se debe actualizar el valor de la métrica para cada nodo del camino que tiene como origen la raíz del a-TAD y como fin el último nodo del camino extraído. Supóngase que el camino extraído es $X(n,m)$ y que si N es un nodo de A, $N.L$ es la medida obtenida para ese nodo al aplicar la métrica del apartado anterior. La actualización de las medidas en el árbol seguirá las reglas siguientes:

$$\begin{aligned} &\forall k \in X(raíz,m) - X(n,m) \quad k.L \leftarrow k.L - lX(n,m) \\ &\forall k \in X(n,m) \quad k.L \leftarrow k.L - lX(k,m) \end{aligned} \qquad (5.11)$$

Esto es, para aquellos nodos desde la raíz al padre del nodo primero de la secuencia extraída se decrementa el mismo valor, a saber, la longitud del camino extraído. Para los nodos del camino se decrementa en la longitud del camino que lleva desde ese nodo al nodo terminal.

La extracción del camino en cada iteración sigue la regla de buscar un nodo terminal m no extraído todavía, de manera que las métricas de cada uno de los nodos que forman el camino desde la raíz sea la máxima entre sus respectivos hermanos. Basándose en las definiciones anteriores se propone el teorema siguiente:

Teorema 5.2. Sea *m* perteneciente a *A* un nodo terminal no incluido en la cadena de salida. Sea *k* un nodo cualquiera de *X(raíz,m)* y *j* perteneciente a {*k*} el hijo de *k* que cumple que *j.L=max(q.L)* con *q* perteneciente a {*k*}. Si *j* pertenece a *X(raíz,m)* entonces *X(n,m)* incluido en *X(raíz,m)* es el camino que representa el siguiente nivel de detalle siendo *n* el primer nodo del camino no incluido en la cadena de salida.

En efecto, para cada padre empezando desde la raíz se elige el hijo *j* con la métrica mayor. Como sólo puede haber un hermano en cada paso, el camino estará formado por los nodos con métrica mayor, que es lo que se pretende. De él se elegirá la secuencia de nodos no incluidos todavía en la multicadena para formar el siguiente nivel.

El algoritmo necesario para la obtención de la cadena multirresolutiva quedará de la siguiente forma:

```
Procedimiento GenerarMultiCadena
    mientras queden nodos en el a-TAD no extraídos (o mientras raíz.L distinto de 0)

        // empezando por la raíz formar el camino que maximiza la métrica
        añadir raíz a camino
        nodo = raíz
        mientras nodo no sea terminal
            añadir al camino el hijo del nodo con métrica mayor que sus hermanos
            nodo=hijo escogido

        // extraer del camino la subcadena a añadir a la multicadena
        nodo=primero del camino
        mientras nodo esté marcado como incluido en multicadena
            nodo=siguiente de camino
        si nodo distinto de raíz entonces añadir a subcadena 'RESTORE(id del nodo padre)'
        desde nodo hasta final del camino
            añadir cadena del nodo a la subcadena
            incrementar longitud del camino en la longitud de este nodo
            si nodo tiene más de un hijo entonces
                añadir a subcadena 'SAVE(id del nodo)'
        añadir subcadena a multicadena

        // corregir las métricas
        para cada nodo del camino extraído anteriormente
            decrementar nodo.L en longitud del camino
        para cada nodo del camino extraído en esta ocasión (de forma ordenada)
            decrementar nodo.L en longitud del camino
            decrementar longitud del camino en la longitud propia
            marcar nodo como incluido

Fin Procedimiento GenerarMultiCadena
```

La cadena multirresolución obtenida para el ejemplo 5.5 es la siguiente:

C1 SAVE(1) C2 SAVE(2) C3 SAVE(3) C5 SAVE(5) C6 RESTORE(2) C9 SAVE(9) C11 RESTORE(1) C12 SAVE(12) C15 C16 RESTORE(3) C4 RESTORE(5) C7 RESTORE(9) C10 RESTORE(12) C13 RESTORE(2) C8 RESTORE(12) C14

5.2.4 Interpretación de la cadena multirresolución

Una vez se ha obtenido la multicadena será necesario realizar un proceso de interpretación para generar los distintos niveles de detalle. Aunque es posible dotar de continuidad a los niveles de detalle de tal manera que sólo se diferencien en una rama, la consideración de caminos como niveles de detalle sucesivos es visualmente más conveniente. La razón de la afirmación anterior es doble. Por un lado, en detalle grueso, se dibujarán rápidamente los principales ramales del árbol dando la impresión de completitud. Por otro, en detalle fino, la variación de un nivel al siguiente se acerca mucho a la continuidad (sólo una rama de diferencia). Esos hechos pueden apreciarse en la distancia entre módulos RESTORE en la multicadena del apartado anterior. Así pues, se deben tener en cuenta las siguientes consideraciones:

- Los niveles de detalle están separados por los módulos RESTORE.
- La interpretación será la misma que una cadena paramétrica excepto que en vez de utilizar una pila para almacenar estados se utilizará una lista indexada.
- La interpretación del módulo SAVE consiste en almacenar el estado de la tortuga en un vector en la posición indicada por el parámetro
- La interpretación del módulo RESTORE consiste en recuperar el estado almacenado en la posición del vector indicada por el parámetro.

El algoritmo que permite generar el árbol con los distintos niveles de detalle y que tiene en cuenta todas las características comentadas de la cadena multirresolución es el siguiente:

```
Algoritmo interpretarMulticadena
    para cada módulo de la multicadena
        caso de que el módulo sea
            primitiva:
                dibujar la forma
            movimiento de tortuga:
                acumular la transformación
            SAVE:
                almacenar la transformación en la posición id
            RESTORE:
                recuperar la transformación de la posición id
fin
```

A priori, el número de niveles de detalle tal como se ha planteado coincide con el número de módulos RESTORE más 1 de la multicadena. Bajo este supuesto, si se ha dibujado el nivel n, para dibujar el nivel n+1 simplemente habrá que interpretar el siguiente RESTORE y el resto de módulos hasta el siguiente RESTORE o el final de la cadena. Lógicamente es necesario un apuntador en la cadena que indique el siguiente módulo a interpretar. Debe entenderse que un nivel de detalle se construye añadiendo a lo ya interpretado en el nivel anterior la subcadena del nivel actual. En el ejemplo anterior:

nivel 1: C1 SAVE(1) C2 SAVE(2) C3 SAVE(3) C5 SAVE(5) C6
nivel 2: nivel 1 + RESTORE(2) C9 SAVE(9) C11
nivel 3: nivel 2 + RESTORE(1) C12 SAVE(12) C15 C16
nivel 4: nivel 3 + RESTORE(3) C4
nivel 5: nivel 4 + RESTORE(5) C7
nivel 6: nivel 5 + RESTORE(9) C10
nivel 7: nivel 6 + RESTORE(12) C13
nivel 8: nivel 7 + RESTORE(2) C8

nivel 9: nivel 8 + RESTORE(12) C14

La figura 5.12 muestra los tres primeros niveles de detalle.

La multicadena puede compilarse con una interpretación previa de manera que los módulos SAVE sean sustituidos por un vector de estados VSAVE. De esta manera la multicadena queda compuesta únicamente por módulos gráficos, de transformación y de recuperación del estado. También es posible generar VSAVE durante la formación de la multicadena con una variante simple del algoritmo dado. Un inconveniente de esta estrategia es que para poder transmitir la multicadena a fin de interpretarla en destino hay que enviar por separado y primero el vector VSAVE completo.

El acceso directo a un nivel de detalle concreto no presenta ningún inconveniente al ser aditivo el modelo multirresolución (un nivel es el anterior más una cadena adicional).

La reducción de niveles de detalle tampoco plantea problemas. Por ejemplo, se podría desear reducir en el ejemplo anterior los 9 niveles a 3. Para ello basta con incluir un nuevo símbolo de comienzo de nivel CNIVEL antes de algunos de los RESTORE. Así, varios niveles antiguos se convertirían en uno sólo comprendiendo varias operaciones de RESTORE hasta el próximo CNIVEL.

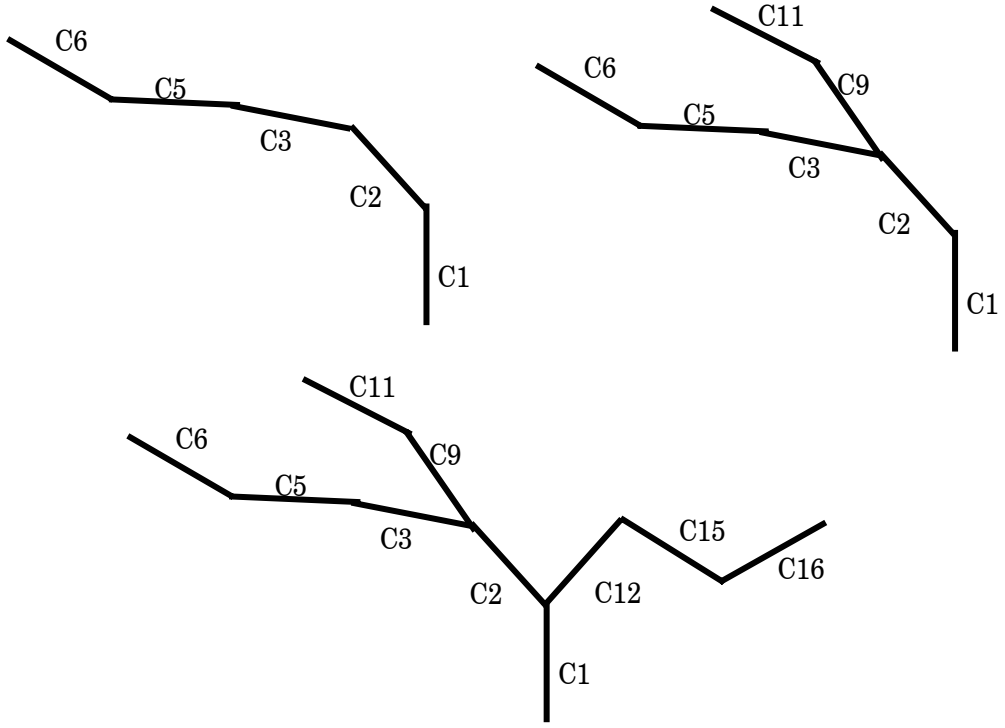


Figura 5.12: Primeros niveles de detalle

Una vez solucionada la visualización de la estructura de las ramas del árbol, tanto de forma geométrica como poligonal, es necesario plantear una solución para la visualización en distintos niveles de detalle del follaje. A continuación, se presenta un modelo que resuelve este problema.

5.3 Modelo multirresolución basado en la imagen para la representación de las hojas

El modelo que se presenta a continuación [Lluch01c] propone la visualización del follaje del árbol mediante la utilización de imágenes precalculadas, que sustituyen las hojas que están dentro de una caja de inclusión que representa a un conjunto de ramas. Los mejores resultados se obtienen cuando el tamaño de las hojas es pequeño en relación con el tamaño total del vegetal a visualizar. Esto es así ya que cuanto más pequeñas sean las hojas, la caja de inclusión englobará un mayor número de ellas y por lo tanto la cantidad de geometría sustituida será mayor. El modelo se podrá utilizar tanto en árboles como en arbustos que cumplan esta condición.

El modelo propuesto permitirá la visualización del árbol en distintos niveles de detalle en función de la distancia del observador. Es posible visualizar el individuo a cualquier distancia y desde cualquier orientación. El nivel con el menor detalle está representado por una única caja que representará las hojas de todo el árbol y el máximo nivel de detalle se representará utilizando un polígono con textura para cada una de las hojas.

La transmisión del modelo a través de la red se puede hacer de forma progresiva enviando las imágenes que forman cada uno de los niveles de detalle, o también se puede enviar un fichero de texto de unos pocos bytes donde se define el sistema-L y generar en el destino los niveles de detalle.

El método mejora las técnicas más recientes desarrolladas por otros autores para la aceleración de la visualización de árboles, y que han sido expuestas en la sección 2.5. El proceso de obtención del modelo multirresolución parte de un sistema-L. Una vez derivada la cadena paramétrica se realiza la fase de interpretación, durante la cual se creará la estructura de datos necesaria para la obtención de las imágenes que se utilizarán como impostores en la visualización del modelo. A continuación se explica como se crea esta estructura de datos.

5.3.1 Estructura de datos

Dadas las características de este tipo de modelos se ha elegido una estructura de datos jerárquica en forma de árbol. El árbol abstracto (grafo) está compuesto por nodos y aristas. Para representar esta estructura se ha optado por la utilización de listas enlazadas de nodos, donde cada nodo contiene un enlace al primer hijo, denominado hijo izquierdo, otro enlace a su primer hermano, denominado hermano derecho y un enlace a su nodo padre. Existe un nodo especial denominado raíz, que no tiene ni nodo padre ni nodos hermanos. Mediante este método de representación es posible simbolizar cualquier tipo de estructura ramificada. Este tipo de estructura es ideal para almacenar, y posteriormente visualizar, la información del árbol modelado mediante el sistema L correspondiente.

Además de los enlaces citados, cada nodo contendrá la definición de la geometría del elemento gráfico que representa: un cilindro, una esfera, un polígono, etc.; y la información referente a la caja de inclusión del subárbol que parte de dicho nodo. También se almacenará el índice del nivel de detalle del nodo y el número de hijos que tiene. En la figura 5.13 se muestra la estructura del nodo.

Cada vez que se interpreta un módulo que identifica a un elemento gráfico, se añade un nodo al grafo indicando el tipo de elemento y sus características. Si se trata del primer nodo del grafo entonces éste se creará como el nodo raíz, sino se creará como un hijo. Si en el nivel actual todavía no había ningún nodo, entonces se tratará del hijo izquierdo. Pero si ya existía

algún nodo, entonces el nuevo nodo se añadirá como hermano derecho del último nodo del nivel actual.

Los módulos de control de pila determinarán el comienzo y el final de las cajas de inclusión que se van a generar. El módulo *PUSH*, guarda el estado en la pila e incrementa en uno el nivel del grafo, y el módulo *POP*, recupera el estado de la pila pasando a un nivel menor del grafo.

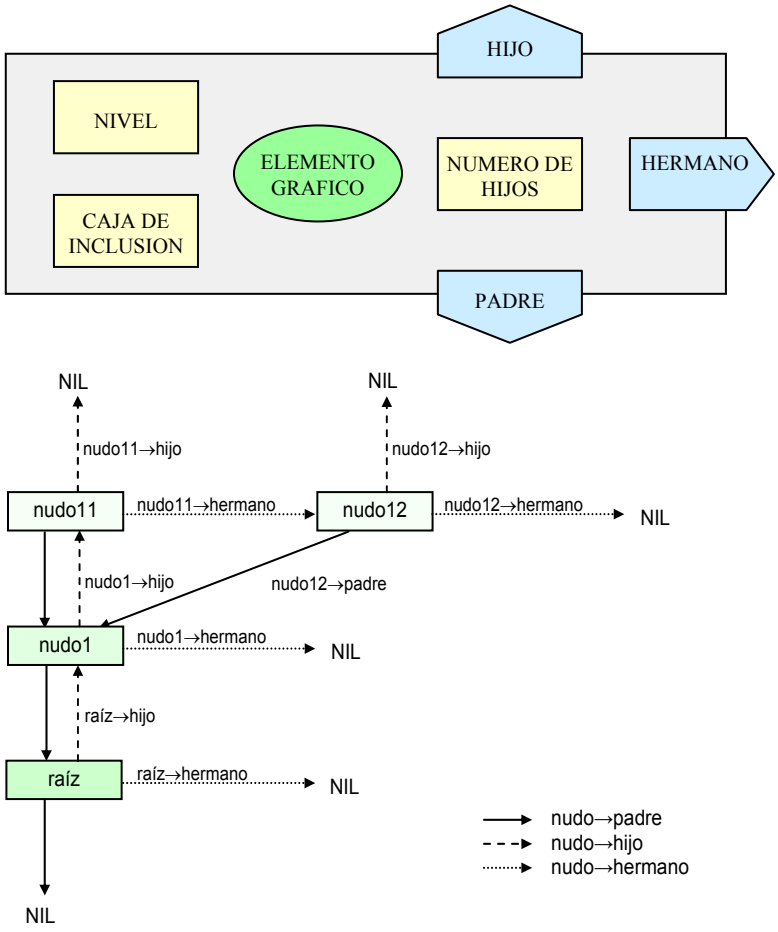


Figura 5.13: Elementos de un nodo del grafo y estructura de los enlaces

5.3.2 Creación de la estructura de datos

Se van a utilizar cajas de inclusión que contendrán un conjunto de ramas del árbol. Para definirlas es suficiente con obtener el extremo inferior (*min*) y el extremo superior (*max*) de la caja, y la matriz de transformación que la orientará y la situará en su correspondiente posición. Cada nodo del grafo tendrá asociada una caja de inclusión que contiene todos los elementos de ese nodo y de todos los nodos que forman el subgrafo que parte del mismo nodo.
Paralelamente a la creación del grafo se calcula la estructura de cajas. Los módulos *PUSH* y *POP* determinan el punto de inicio y final de un nuevo ramal, y por lo tanto, de una nueva caja de inclusión. Existe un límite en el número de niveles que se profundizará en el grafo para calcular las cajas de inclusión y que se denomina *máximo nivel de detalle.*

En un instante cualquiera del proceso de interpretación, se define como caja *abierta* aquella cuyas dimensiones no son todavía definitivas, es decir, aún falta por analizar parte del ramal y es posible que el volumen de la caja se incremente. También se define la *última* caja como

aquella que se ha creado en último lugar. Y por último, el *nivel* determina la profundidad actual en el grafo.

Algoritmo de generación de las cajas de inclusión

Una vez explicados los conceptos más importantes, se puede exponer el algoritmo que permite la obtención de la caja de inclusión que le corresponde a cada nodo:

Procedimiento generarCajas
  Recorrer todos los módulos de la cadena
    En caso de que el módulo sea:

PUSH	Crear nueva caja Inicializar *min*, *max* Marcar como *última* y *abierta* Incrementar *nivel*
POP	Cerrar *última* caja Guardar dimensiones de la caja
CILINDRO, FORMA	Actualizar *max*, *min* de todas las cajas *abiertas*
AVANCE	Modificar posición de la tortuga
GIRO	Modificar orientación de la tortuga

Fin Procedimiento generarCajas

El proceso se realiza de la siguiente manera, cuando el intérprete encuentra un módulo de tipo *PUSH* se crea una nueva caja abierta inicializando sus extremos a cero, marcándose como *última*. Desde este momento y hasta que aparece un módulo *POP*, se actualizan las dimensiones de todas las cajas abiertas. Una vez finalizado este proceso cada nodo del grafo tendrá asociada una caja de inclusión. Se debe tener en cuenta que debido a que los sistemas L producen cadenas bien anidadas, el número de módulos PUSH es el mismo que de módulos *POP*, de este modo, la caja perteneciente a un nodo, incluirá a todas las cajas que pertenezcan a los nodos de los niveles inferiores del subgrafo que parte de ese nodo.

Cada una de estas cajas de inclusión contendrá una parte de la geometría del árbol modelado. Por lo tanto, ahora es posible calcular una serie de imágenes que representen la geometría que se encuentra en el interior de la caja. A continuación se comenta como se realiza el proceso de generación de estas imágenes.

Generación de las texturas precalculadas

En las texturas sólo se va a plasmar la geometría que representa a las hojas y no a las ramas debido a que aparecen problemas de discontinuidad cuando se cambia de nivel de detalle que son difíciles de resolver. Esto ocurre porque las proyecciones de las ramas de los niveles de detalle consecutivos no ocupan la misma posición cuando se visualizan las texturas que les corresponden a cada uno de ellos.

Cuando se cierra una caja se almacenan las dimensiones de la misma en la estructura de datos. Esta caja define un volumen que envuelve completamente un ramal del árbol. Así que, se puede aprovechar esta característica para generar un conjunto de imágenes que represente la geometría que se encuentra dentro de ese volumen, para posteriormente sustituir en el proceso de visualización dicha geometría por las imágenes.

La primera aproximación consiste en generar una imagen por cada cara de la caja, obtenida como una proyección ortográfica de la geometría sobre el plano que forma esa cara. Para

conseguirlo se coloca la cámara en el centro de la cara en estudio con el vector *look* orientado hacia el centro de la caja y el vector *up* orientado según la vertical de la caja, después se proyecta el ramal contenido en la caja y por último se almacenan los píxeles calculados en una textura. Por lo tanto, es necesario transformar el sistema de coordenadas de la cámara en el sistema de coordenadas de la cara para la cual se desea generar la textura, ajustando y recortando el *viewport* de la cámara al tamaño de dicha cara. En la figura 5.14 se puede observar de manera gráfica este proceso.

Tras este proceso se habrán almacenado en disco seis imágenes por caja, esto supone un coste de almacenamiento muy alto, sobre todo si las texturas tienen un tamaño que no sea el adecuado y si se han generado cajas para todos los ramales del árbol. La memoria de texturas de las tarjetas gráficas más utilizadas está limitada a 32 Mbytes, y a 64 Mbytes en otras de un coste superior. Por lo tanto, hay que analizar estas dos situaciones y buscar un compromiso entre calidad y coste.

Figura 5.14: Generación de las texturas precalculadas.

Tamaño de las texturas

Para determinar el tamaño más adecuado para las imágenes hace falta estudiar el número de píxeles que van a cubrir en la pantalla cuando sean proyectadas. Las cajas de inclusión que pertenecen a un mismo nivel del grafo, representan un nivel de detalle de visualización, de modo que las cajas de un nivel incluirán a todas las de los niveles inferiores. Por lo tanto, si se visualizan las texturas que pertenecen a una caja de inclusión, no se visualizarán las que estén incluidas en ella. De este modo, si el observador se encuentra lo suficientemente alejado, el follaje del árbol se representará con las texturas calculadas para la caja de inclusión que abarca a todo el árbol. Debido a que la distancia al árbol es grande, las texturas ocuparán un pequeño grupo de píxeles. Cuando el observador esté más próximo al árbol, se utilizarán las texturas de las cajas de inclusión más pequeñas que representan los diferentes ramales y aunque, en este caso el tamaño que ocupa el árbol en pantalla es mayor, las texturas que sustituyen los ramales son menores, por lo que ocupan un número de pixeles similar a las de los niveles de detalle superiores.

Este desarrollo indica que el tamaño de las texturas puede ser constante para todos los niveles de detalle, independientemente de las dimensiones de las cajas de inclusión que representen. Para comprobar cual es el tamaño ideal para almacenar las texturas se debe tener en cuenta que las imágenes deben ser cuadradas y que el tamaño debe ser potencia de 2. En principio, se descartan las texturas de tamaños superiores a 128x128 por ser demasiado grandes, y la memoria de textura es limitada, además la mejora visual que se obtiene respecto a la utilización de texturas de 128x128, es inapreciable. Por otro lado, se decide que las texturas de

tamaño menor a 64x64 píxeles son demasiado pequeñas y producen distorsiones en la proyecciones de las hojas. Por lo tanto, los tamaños que se van a utilizar son 64x64 y 128x128 pixeles. Las primeras se utilizarán si se desea representar una variedad de árboles mayor, en concreto cuatro veces más que con las texturas de 128x128, sin embargo, si se desea obtener mejor calidad, se utilizarán las segundas.

Utilizando texturas de 128x128 píxeles y 16 bits de profundidad, se puede comprobar que para almacenar las texturas de un árbol de tipo binario completo donde cada nodo tiene dos hijos, con 8 niveles de profundidad, serían necesarios 96Mbytes de memoria para almacenar las texturas correspondientes a un árbol. Es por ello necesario reducir el número de niveles para los que se va a calcular las texturas.

8 niveles → número de nodos (cajas) = $2^8 - 1 = 255$ cajas
6 caras (texturas) por caja × 255 cajas = 1530 texturas
tamaño de textura = 128 píxeles × 128 píxeles × 2 bytes = 64 Kbytes
*1530 texturas × 64 Kbytes por textura ≅ **96 Mbytes***

Número de niveles
En este caso se ha de tener en cuenta el espacio que ocupa un ramal del árbol con relación al tamaño de la textura que lo debe sustituir. Si cubre más píxeles de los que cubriría la textura, ésta debe ser escalada a un tamaño mayor, con lo que aparece *aliasing*. Para evitar esto, habría que profundizar un nivel más en ese ramal y generar las texturas para los ramales hijos.

Para reducir el coste espacial del modelo, se ha desarrollo una estrategia adaptativa para la generación de texturas, que tiene en cuenta la topología del árbol para determinar el máximo número de niveles a generar. Dicha estrategia consiste en calcular la relación entre el volumen de la caja del nodo para el cual se están generando las texturas y el volumen de la caja del nodo raíz. Si esta relación es menor que un umbral, ese nivel ya no se divide. Utilizando el valor del umbral se puede ajustar el número de niveles del árbol de cajas de inclusión, y en consecuencia el tamaño total de las texturas que representan el follaje del árbol. El tamaño máximo que suelen ocupar las texturas con este tipo de estrategia es de 6 Mbytes, aunque si se utilizan texturas de 64x64 este tamaño se reduce a menos de 2 Mbytes.

El siguiente punto describe el algoritmo de visualización del modelo que se ha desarrollado, además de comentar algunos aspectos de aceleración y de mejora de la calidad visual.

5.3.3 Visualización del modelo

La visualización de árboles en entornos interactivos o de tiempo real, resulta una tarea muy poco eficiente debido a la complejidad geométrica de este tipo de modelos. Por ello se ha investigado la forma de sustituir esta gran cantidad de información geométrica por otras representaciones más sencillas, pero intentando no disminuir la calidad visual obtenida.

Cuando el observador se encuentra a una distancia considerable del árbol, se sustituye el follaje del mismo por las imágenes asociadas a las cajas de inclusión del nodo raíz. Para ello se emplea un polígono por cada cara de la caja, colocándolos perpendicularmente sobre el eje central de la caja, tal y como se indica en la figura 5.15. Como en un momento cualquiera de la visualización el observador no será capaz de ver todas las caras de la caja, se realiza back-face culling.

En el instante en que el observador se acerque un poco más al árbol, se pasará a visualizar las imágenes asociadas a las cajas del siguiente nivel del grafo. Este proceso se repite según el

observador se encuentre cada vez más próximo al árbol. Se debe tener en cuenta que los polígonos texturizados están orientados según la dirección del ramal al cual representan, por lo que la representación es coherente con la topología del árbol.

Cabe destacar que la comprobación de la distancia a la cual se encuentra el observador no se realiza con respecto a todo el árbol sino a cada una de las cajas que se estén visualizando, dependiendo también del tamaño de las mismas, por lo que, en un momento determinado, habrá ramales que se encuentren representados por cajas de un nivel de detalle diferente al de otros ramales. Este es el motivo por el cual esta técnica se presenta como un método multirresolución discreto y con nivel de detalle dependiente de la vista.

Figura 5.15: Visualización en aspas de las texturas pertenecientes a una caja de inclusión.

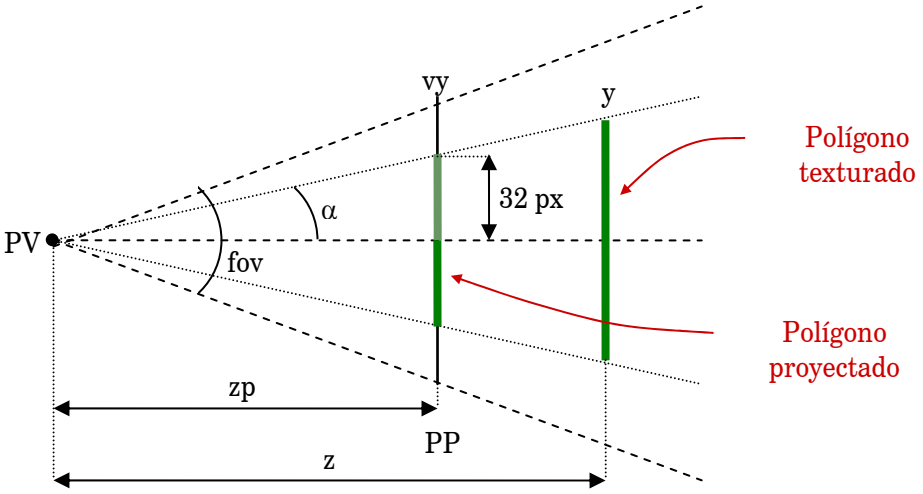


Figura 5.16: Determinación de la distancia máxima de proyección de la textura.

Para determinar la distancia a la que se debe cambiar de un nivel de detalle a otro, se va a calcular para cada caja la distancia a la que la textura correspondiente se proyectará ocupando el máximo número de píxeles posible. El área ocupada por la proyección se maximizará cuando el polígono a proyectar está perpendicular al vector LOOK. En estas condiciones se puede determinar la distancia entre el observador y la caja cuando la proyección ocupa un área de 128x128 ó 64x64 pixeles (figura 5.16). De este modo, cuando la textura esté situada a esta distancia y sea perpendicular al vector LOOK, no será necesario realizar escalados que deriven en efectos de *aliasing*. Por lo tanto, la distancia donde un nivel de detalle tiene que cambiar al siguiente o al anterior, será función de este valor que se ha calculado. Se debe tener en cuenta,

que el tamaño de la proyección será menor si la textura no es perpendicular al vector LOOK, lo cual ocurrirá en la mayoría de los casos, por ello se debe de aplicar un factor de corrección para determinar la distancia del LOD. Cada caja de inclusión tendrá asociada una distancia, que se precalculará, para acelerar el proceso de visualización. A continuación se muestra el algoritmo de visualización que se ha desarrollado.

```
Procedimiento dibujar
  dibujarNodo (raiz)
Fin Procedimiento dibujar

Procedimiento dibujarNodo (nodo)
   Obtener distancia desde la caja del nodo al observador
     Si distancia > distancia almacenada en la caja
        Dibujar texturas de la caja
     Sino
         Si no se ha alcanzado el máximo nivel de detalle
           Para cada hijo dibujar(hijo)
         Sino
            Dibujar las hojas de la caja como polígonos individuales
Fin Procedimiento dibujarNodo
```

Los resultados que se consiguen mediante esta técnica son aceptables pero se pueden mejorar. Existen dos problemas principales a solucionar. En primer lugar, la sensación de volumen que se obtiene al visualizar las texturas no es suficiente. Y en segundo, la transición de un nivel de detalle al siguiente no se realiza de forma suave. A continuación se comentan algunas mejoras que se han aplicado al método inicial.

Mejoras en la visualización

Para conseguir mayor volumen se utilizó una técnica similar a la propuesta por Meyer [Meye98]. Esta técnica consiste en dividir el volumen de la caja en capas (*slices*) paralelas entre sí, para cada una de las tres direcciones principales de la caja. La generación de las texturas precalculadas es similar a la situación inicial, lo único que hace falta es ajustar los planos de recorte para que se ajusten a cada una de las capas (figura 5.17).

Mediante esta aproximación se consigue una sensación de volumen mucho mayor, así como una transición entre niveles de detalle muy suave. El problema ahora es que la cantidad de memoria necesaria para almacenar las texturas es mucho mayor, tantas veces como subdivisiones de la caja se efectúen. Este hecho limita todavía más el número de árboles diferentes que se pueden visualizar en un escenario, lo cual no es muy recomendable.

Figura 5.17: División de la caja por capas en los planos principales.

La solución definitiva por la que se ha optado se encuentra a medio camino entre las dos alternativas anteriores. En este caso se utilizan los planos diagonales de las cajas para generar las texturas como se puede observar en la figura 5.18.

Con este método se obtiene una sensación de volumen adecuada, una transición suave entre niveles de detalle y el número de imágenes que se generan por caja es el mismo que en la situación inicial.

Figura 5.18: División de la caja por sus planos diagonales.

Por último, hay que comentar que es necesario conseguir una transición suave entre las diferentes caras de cada caja y entre los distintos niveles de detalle, para que no se produzcan saltos en la visualización cuando se pasa de mostrar una cara a la contigua, o de un nivel de detalle al siguiente o al anterior. Para evitar estos saltos en la visualización se utiliza una técnica denominada *blending*. La solución consiste en modificar el canal alfa de los polígonos visualizados de manera progresiva, según se mueve el observador. Para el cálculo del valor del canal alfa se han definido tres regiones que rodean al plano, como se ve en la figura 5.19.

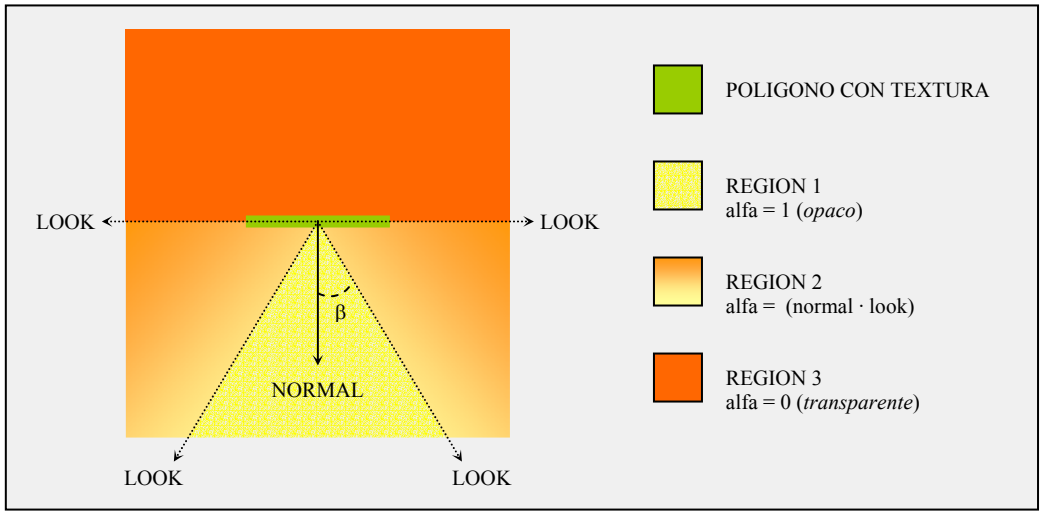


Figura 5.19: Cálculo del valor del canal alfa dependiendo de la posición del observador.

En la región 1, donde el observador se encuentra encarado al plano, el canal alfa valdrá 1 (polígono opaco). En la región 2 es donde se produce la transición de opaco a transparente, para la cual, el canal alfa se calcula como el producto escalar entre la normal del plano que forma el polígono y el vector *look* del observador. Dentro de la región 3 no se dibujará el polígono, ya que es totalmente transparente.

En el capítulo siguiente se comentan los principales resultados que se han obtenido con la técnica descrita. Una vez presentado el modelo multirresolución para la representación del follaje del árbol, a continuación se presenta un modelo conjunto que tiene en cuenta el modelo completo, es decir ramas y hojas.

5.4 Modelo multirresolución para la representación del árbol

Una vez presentadas las soluciones parciales para las ramas y para las hojas se plantea crear un modelo que englobe a ambos y que permita la representación y transmisión por la red del árbol completo. El modelo es una fusión de los dos últimos modelos expuestos, es decir las ramas se representan mediante los niveles de detalle obtenidos a partir de la cadena multirresolución, y las hojas se representan mediante la estructura de imágenes precalculadas obtenidas mediante la estructura de cajas de inclusión que se genera durante la interpretación de la cadena.

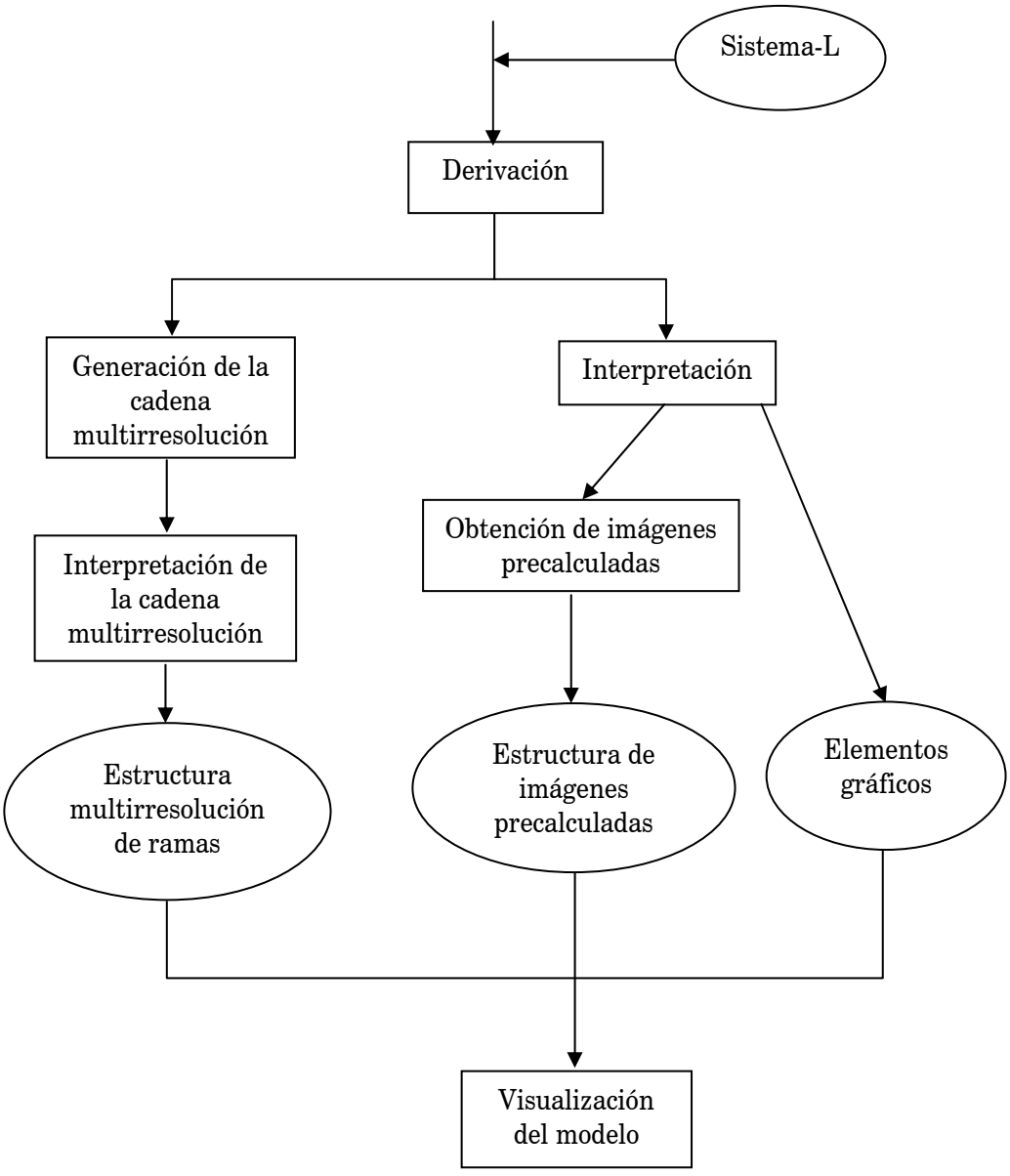


Figura 5.20: Fases para la obtención del modelo multirresolución conjunto

Los pasos que se van a seguir para obtener el modelo completo son los siguientes:

- La obtención del modelo parte de un sistema-L
- Una vez se ha derivado el sistema se obtiene una cadena de módulos paramétricos
- Esta cadena se utiliza en dos procesos:

    - Por un lado, mediante la interpretación de la cadena se obtendrán los elementos gráficos que representan al modelo y la estructura de cajas de inclusión que contienen a cada uno de los niveles de ramificación del árbol. Con esta estructura de cajas de inclusión se calculan las imágenes que representan a las hojas incluidas dentro de las cajas y que posteriormente se utilizarán como textura para sustituir la geometría correspondiente.
    - Por otro lado, la cadena derivada se utiliza para obtener la cadena multirresolución con la que se podrá construir la estructura de datos que representará en diferentes niveles de detalle, todas las ramas del árbol. Para ello se realizará el proceso de interpretación de la cadena multirresolución.
- Por último, utilizando las tres estructuras de datos generadas se puede realizar la visualización multirresolución del individuo modelado.

En la figura 5.20 se muestra gráficamente las distintas fases de obtención del modelo conjunto.

El modelo necesita una superestructura de datos que englobará a las dos estructuras que representan a las ramas y a las hojas en distintos niveles de detalle. Esta superestructura está orientada a la visualización interactiva del individuo en un entorno formado por elementos naturales. La estructura de datos incluye la siguiente información:

- Fichero donde está almacenado el sistema-L
- Matriz de transformación, para situar, orientar y definir el tamaño del árbol en la escena
- Estructura de datos que almacena las hojas
- Estructura de datos que almacena las ramas

La estructura de datos que almacena las hojas contiene la información referente a la visualización de las hojas:

- Textura que representa a la hoja
- Material de la hoja
- Tipo de visualización: si se visualizan o no, y si se visualizan, si lo hace en modo multirresolución
- Estructura de datos de tipo árbol abstracto que almacena las cajas de inclusión

Cada nodo de esta estructura de datos almacena la información referente a la caja de inclusión a la que pertenece y a los elementos (hojas) que engloba, para su posterior visualización. Cada caja de inclusión tendrá asociadas las texturas que la representan y la distancia a la que se debe visualizar.

Por otro lado, la estructura de datos que almacena las ramas contendrá la siguiente información, necesaria para su visualización:

- Textura que envuelve las ramas
- Material de las ramas
- Tipo de visualización: si se visualizan o no, y si se visualizan, si lo hacen en modo multirresolución
- Estructura de datos de tipo lista que almacena los diferentes niveles de detalle

La estructura de datos que almacena los niveles de detalle de las ramas, será una lista de punteros. Cada uno de estos punteros enlazará con el primer nodo que forma el nivel de

detalle. Los nodos contendrán la lista de elementos (tramos de ramas) que representan a cada rama.

En la figura 5.21 se muestra la estructura de datos que almacena toda la información necesaria para representar el árbol multirresolución.

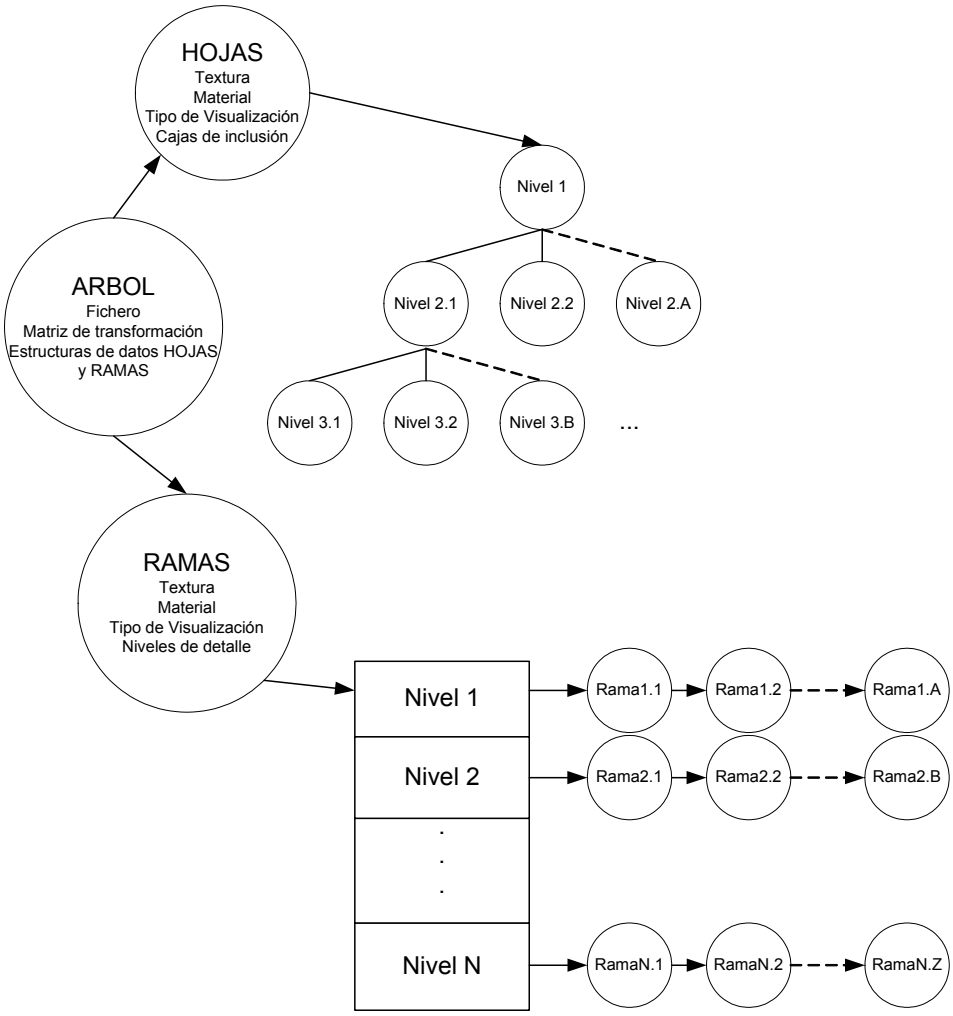


Figura 5.21: Estructuras de datos que permiten almacenar las ramas y las hojas del árbol multirresolución para su visualización en una escena

5.4.1 Transmisión del modelo

El modelo se puede transmitir de forma progresiva. Por un lado, se puede enviar la cadena multirresolución que representa las ramas del árbol, en su totalidad o por niveles de detalle. La cadena multirresolución se almacena en un fichero de texto cuyo tamaño no alcanza los 100Kbytes. Sin embargo, si este fichero se comprime su tamaño se reduce a menos de 10Kbytes. Por otro lado, se pueden enviar las imágenes que pertenecen a cada uno de los niveles de detalle por separado. El tamaño total de las texturas está entre 2 y 6 Mbytes. Si se comprime el tamaño se reduce a entre 100 y 400 Kbytes en término medio.

En situaciones críticas de bajo ancho de banda se puede solucionar enviando el fichero de definición del sistema L. Este fichero ocupa unos pocos bytes. En este caso, el proceso de derivación e interpretación del sistema se realizará en el destino, por lo que la generación de

las texturas para las hojas y de la cadena multirresolución para las ramas se realizará en el destino. Por ejemplo, si se tiene el sistema:

```
SYSTEM
DECLARATIONS
AXIOM
    A{120.0,20.0}
RULES
    A{s,w}: NumIt=MaxIt # Instance{0,10,1.0,16}Repeat{4,\{60}Instance{0,10,1.0,16}}
    A{s,w}:(s>=min) #
             Cylinder{s,w,0.707*w}F{s}Push{}+{30.0}/{137.0}A{s*0.6,w*0.707}Pop{}
             Push{}-{30.0}/{137.0}A{s*0.9,w*0.707}Pop{}
END
```

Si se realizan 11 iteraciones, se genera un fichero que almacena la cadena multirresolución que ocupa 72 Kbytes, y comprimido 7 Kbytes. Por otro lado, se generan 204 texturas, que si se utiliza un tamaño de 64x64 píxeles y 16 bits de profundidad, ocupan 1.8 Mbytes. Si se comprimen ocupan en total 151 Kbytes. El tamaño de los diferentes niveles de detalles se puede consultar en la tabla 5.1. En la figura 5.22 se muestran las imágenes generadas en los primeros niveles de detalle.

Nivel	Ocupación (Kbytes)	
	Sin comprimir	Comprimido
Nivel 1 : 12 texturas	96	10
Nivel 2 : 24 texturas	192	20
Nivel 3 : 48 texturas	384	38
Nivel 4 : 72 texturas	768	52
Nivel 5 : 24 texturas	192	15
Nivel 6 : 24 texturas	192	16
Total	1781	151

Tabla 5.1: Ocupación de las texturas que representan a los diferentes niveles de detalle

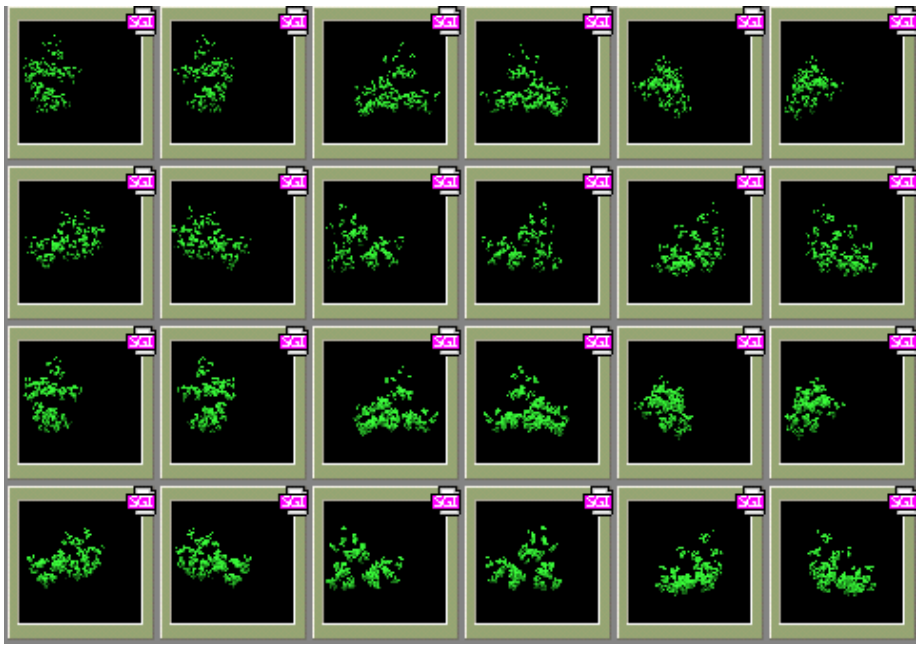


Figura 5.22: Imágenes generadas para los primeros niveles de detalle

5.5 Integración del modelo en poblaciones

En los puntos anteriores se ha descrito con detalle un modelo que permite la representación de los elementos de un árbol, hojas y ramas, de forma multirresolutiva. Sin embargo, no se puede terminar este capítulo sin comentar algunos aspectos que son importantes cuando el modelo se incorpora en un entorno natural. No es lo mismo visualizar un único árbol que visualizar un conjunto de ellos. Las cuestiones fundamentales a tener en cuenta ya se han planteado en el capítulo 4, y se pueden resumir de la siguiente forma, teniendo en cuenta como se puede utilizar el modelo presentado para resolverlas:

- Almacenamiento de la escena
- Modelado e instanciación de plantas
- Visualización interactiva de la escena

En cuanto al almacenamiento de la escena, se debe tener en cuenta que el modelo que se ha expuesto es muy compacto, ya que la representación del árbol se realiza mediante un sistema-L que se almacena en un fichero que ocupa unos pocos bytes, y si se desea almacenar el árbol generado, se pueden utilizar formatos de imagen comprimidos.

Aún consiguiendo un tamaño reducido de almacenamiento se debe tener en cuenta que una escena puede estar formada por miles de elementos, es interesante utilizar la instanciación para obtener todos aquellos elementos que sean parecidos a uno dado, por lo tanto es posible representar gran cantidad de árboles utilizando unos pocos como patrón a los que se les aplica transformaciones: giros y escalados. Otra forma sencilla de obtención de modelos parecidos, pero no iguales utilizando el modelo propuesto, consiste en modificar la posición o la orientación de las texturas calculadas. Se pueden cambiar las texturas calculadas para una caja por las calculadas para otra, o se pueden girar las cajas de inclusión con respecto a su eje principal y las texturas que se observan desde el punto de vista serán distintas, para cada árbol.

Por último, el modelo que se ha presentado permite la multirresolución del individuo modelado y por lo tanto acelera la visualización del mismo. Sin embargo, es posible aplicar técnicas que tengan en cuenta la interacción de los distintos individuos que forman la escena. Por ejemplo, las cajas de inclusión que se han calculado pueden ser utilizadas como oclusores a la hora de calcular la visibilidad de la escena, en el caso de que la rama representada sea lo suficientemente frondosa, esto se puede calcular con una función de densidad para cada una de las cajas que forman el individuo.

En el siguiente capítulo se muestran los resultados que se han obtenido en la investigación, y para cada uno de los modelos que se han presentado anteriormente.

Capítulo 6
Resultados obtenidos

Una vez se han desarrollado los diferentes modelos propuestos en la presente tesis, se va a realizar un estudio de los resultados que se han obtenido con cada uno de ellos. Este capítulo se va a dividir en una sección por cada uno de los modelos, analizando tanto los resultados visuales como los cuantitativos. Para realizar las pruebas se ha utilizado un sistema basado en un Pentium IV 1,5 Ghz con 256 Mbytes de memoria RAM y una tarjeta gráfica GeForce 3.

En primer lugar se muestran los resultados que ha dado el modelo que permite representar una estructura ramificada mediante una única malla poligonal. En la segunda sección, se aborda el problema de la representación de la estructura ramificada desde el punto de vista procedural. A continuación, se analizan los resultados que ha generado el modelo multirresolución para la visualización del follaje del árbol mediante impostores precalculados. Por último, se hace un estudio del modelo completo que permite la representación multirresolución de un árbol, incluyendo las ramas y el follaje.

6.1 Modelo de representación de una estructura ramificada mediante una malla poligonal

El resultado del algoritmo descrito será una única malla de triángulos que representará en 3D el árbol descrito por la cadena de símbolos que se toma como entrada. Como se aprecia en la figura 6.1 se solucionan los problemas, tanto de continuidad como de visibilidad, que aparecen cuando se representan las ramas como una unión de primitivas.

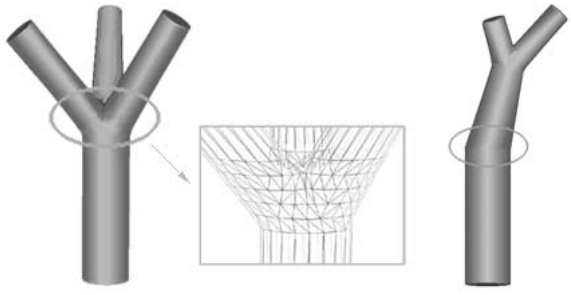
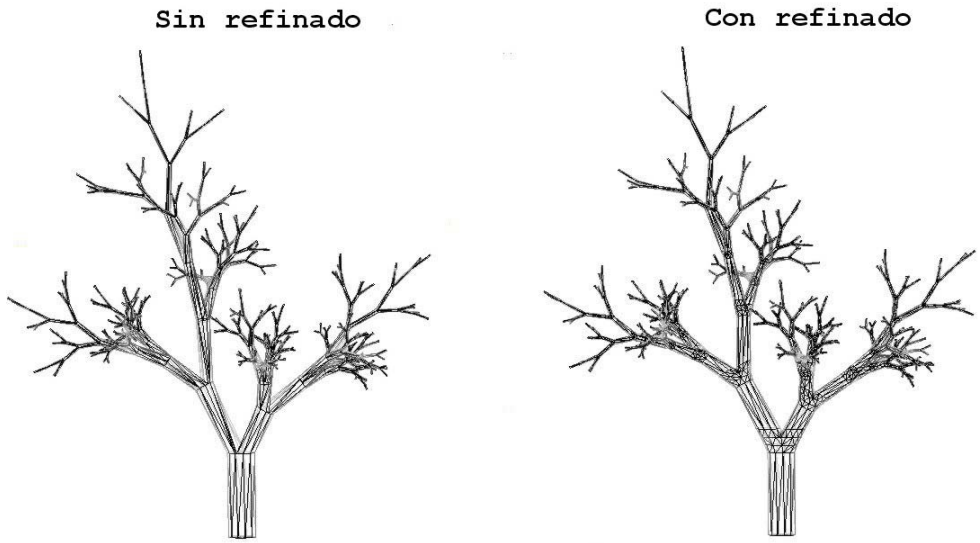


Figura 6.1: La malla única da soluciones a los problemas de visibilidad de representaciones anteriores.

Para analizar los resultados del modelo, se han utilizado varios sistemas-L. Para la obtención del modelo que aparece en la figura 6.2 se ha utilizado el siguiente sistema:

```
SYSTEM Ejemplo
DECLARATIONS
   CConstant r 0.6
   CConstant rb 0.9
   CConstant alfa 30.0
   CConstant alfab -30.0
   CConstant fi 137.0
   CConstant fib 137.0
   CConstant q 0.707
   CConstant e 0.5
   CConstant min 0.0
AXIOM
   A{120.0,20.0}
RULES
   A{s,w}: NumIt=MaxIt # Instance{0,10,1.0,16}Repeat{4,\{60}Instance{0,10,1.0,16}}
   A{s,w}: (s>=min) # Cylinder{s,w,q*w}F{s}
                        Push{}+{alfa}/{fi}A{s*r,w*q}Pop{}
                        Push{}+{alfab}/{fib}A{s*rb,w*q}Pop{}
END
```

En dicha figura se presentan los resultados que muestran el modelo tanto sin aplicar el método de refinamiento como aplicando el método de refinamiento, y visualizado en alámbrico y en sombreado. El resultado de la reconstrucción con refinamiento es visualmente muy superior al método de reconstrucción sin aplicar dicha mejora. En la tabla 6.1 se muestra una comparativa del número de triángulos generados con el método de refinamiento por intervalos con respecto al método directo. Con esta tabla se demuestra que la mejora visual que consigue el método de refinamiento por intervalos se realiza a costa de aumentar, en el peor de los casos, del orden de 20 veces el coste temporal y 3 veces el coste espacial, aunque hay que tener en cuenta que este proceso sólo se ha de realizar una vez (para generar el modelo poligonal).



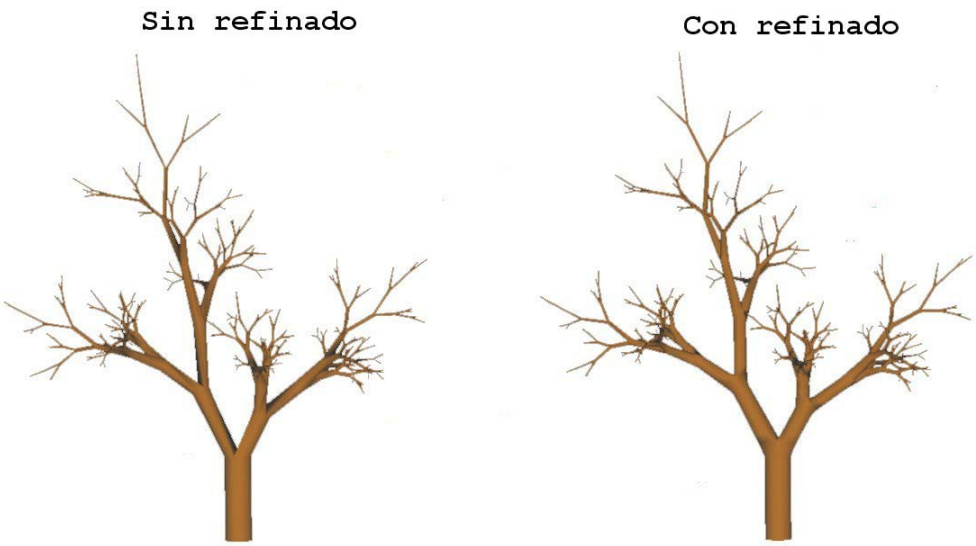


Figura 6.2: Resultado de aplicar el algoritmo desarrollado a una cadena generada mediante un sistema-RL. A la izquierda se muestra el árbol obtenido sin aplicar el refinamiento por intervalos, a la derecha se aprecia el mismo árbol una vez aplicado el refinamiento de los nodos. Arriba se ve la malla poligonal en alámbrico y en las imágenes de abajo se ha realizado el sombreado de los polígonos.

Figura 6.3: Aplicación de un algoritmo de simplificación a la malla que representa el árbol, obteniéndose tres niveles de detalle, para su utilización en un mundo VRML.

Vértices del contorno	Polígonos generados		Tiempo requerido	
	Árbol original	Refinado por intervalos	Árbol original	Refinado por intervalos
5	4998	9708	10s 475ms	36s 752ms
9	5028	10142	10s 615ms	40s 037ms
12	5343	13762	12s 118ms	1m 12s 023ms
15	8435	24482	27s 450ms	3m 44s 092ms

Tabla 6.1: Comparación de costes espaciales y temporales del árbol reconstruido, aplicando o no el algoritmo de refinado, con distintas resoluciones para el contorno.

Este incremento del coste espacial exige que, para la inclusión de estos elementos para la generación de campos virtuales, objetivo principal de este tipo de reconstrucciones, precise la aplicación de procesos de decimación que respeten la topología de dichas mallas. Para las reconstrucciones obtenidas con el algoritmo descrito en la presente tesis, se está aplicando un algoritmo de decimación [Schr92] que consigue una reducción de hasta el 80% en el número de triángulos que representan la malla sin que con ello se vea afectada la mejora visual conseguida con el método de refinamiento por intervalos. En la figura 6.3 se puede observar un mundo VRML, donde se han incluido modelos con tres niveles de detalle.

6.2 Modelo multirresolución procedural para la representación de las ramas

Utilizando el mismo sistema-L se han hecho pruebas con el modelo de multirresolución procedural, para obtener distintos niveles de detalle del mismo modelo. A continuación se presenta la cadena multirresolución obtenida tras 3 iteraciones:

```
Cylinder{120.,15.,10.60}F{120.}
Save{1}+{-30.}/{137.}
Cylinder{108.,10.60,7.49}F{108.}
Save{5}+{-30.}/{137.}
Cylinder{97.2,7.49,5.30}F{97.2}
Recover{1}
+{30.}/{137.}
Cylinder{72.,10.60,7.49}F{72.}
Save{2}
+{-30.}/{137.}
Cylinder{64.8,7.49,5.30}F{64.8}
Recover{5}
+{30.}/{137.}
Cylinder{64.8,7.499,5.30}F{64.8}
Recover{2}
+{30.}/{137.}
Cylinder{43.2,7.49,5.30}F{43.2}
```

En la tabla 6.2 se hace una comparación con los símbolos y polígonos necesarios para representar distintos niveles de detalle. Partiendo de un árbol generado tras 11 iteraciones del sistema se presentan en la figura 6.4 el árbol completo y distintos niveles de detalle del mismo. Como se puede observar, la estructura general de las ramas del árbol se aprecia en cualquiera

de los niveles de detalle, pero la reducción del número de polígonos necesario para representarlos son notablemente inferiores en los niveles de detalle más bajos. En la figura 6.5 se muestran los niveles de detalle en la distancia que les corresponde, sin hojas y con hojas. En estas últimas imágenes se puede apreciar mejor como los niveles de detalle obtenidos son adecuados para el modelo ramificado que representan.

(#) LOD	# Símbolos	# Polígonos
(4) Mínimo	168	172
(50) Medio-Bajo	1183	872
(200) Medio-Alto	3198	2103
(511) Máximo	5113	3252
Cadena paramétrica	6134	3252

Tabla 6.2: Tabla comparativa, de símbolos y polígonos.

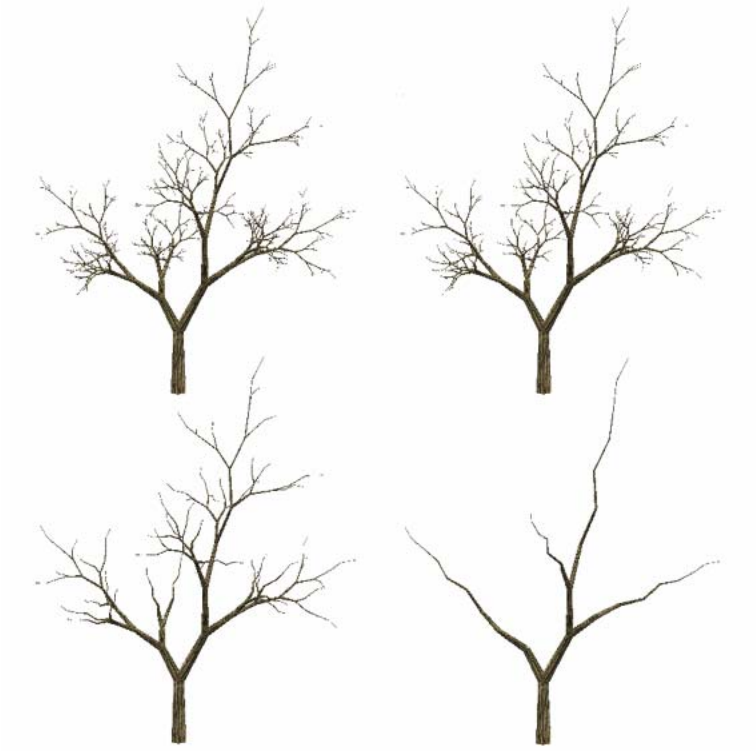


Figura 6.4: Árbol total y diferentes niveles de detalles

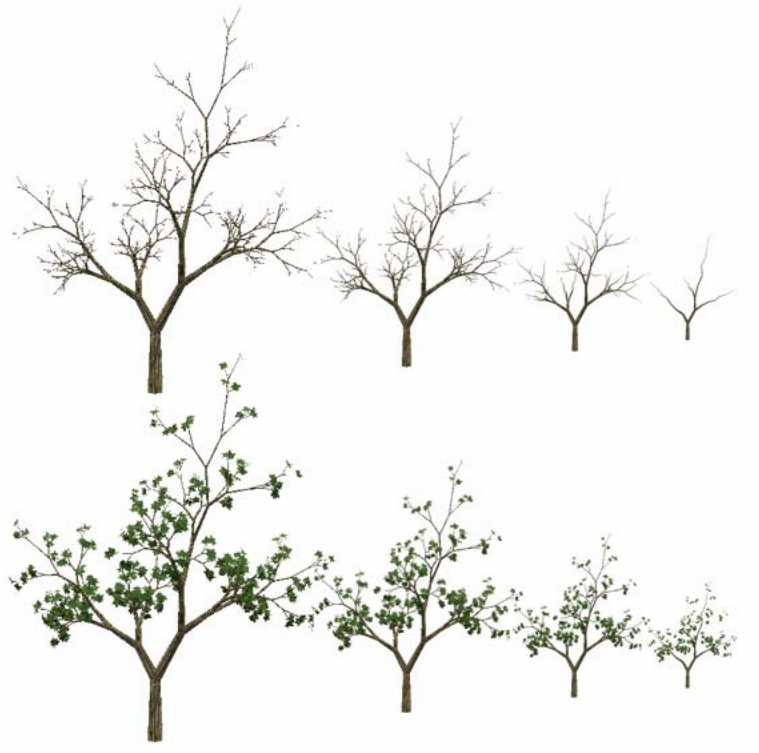


Figura 6.5: Los diferentes niveles de detalles de la figura 6.3 situados a la distancia apropiada, en la imagen inferior se puede apreciar el mismo ejemplo pero con las hojas correspondientes.

6.3 Modelo multirresolución basado en la imagen para la representación de las hojas

Utilizando el mismo sistema anterior se puede obtener el conjunto de imágenes que representan el follaje en diferentes niveles de detalle. La producción de las imágenes del modelo tarda menos de un minuto. El tamaño máximo de las imágenes es de 6Mbytes, aunque utilizando texturas de 64x64 píxeles, el tamaño se reduce a menos de 2 Mbytes. Por lo tanto, es posible visualizar hasta 16 modelos distintos utilizando una tarjeta gráfica con 32 Mbytes.

En la figura 6.6 se muestra un modelo utilizando sólo geometría y las distintas configuraciones propuestas. Se puede apreciar que las diferencias visuales entre el uso de geometría o la disposición diagonal de texturas es mínima. Sin embargo, el follaje de la primera está formado por 20000 polígonos texturados, mientras que en la segunda sólo son necesarios 50 polígonos.

En la figura 6.7, se puede observar una inspección cercana de un modelo. Las hojas individuales que están cercanas al observador se representan mediante un polígono cada una, es posible que otras hojas del árbol se visualicen utilizando las imágenes precalculadas. En la figura 6.8, se puede apreciar otra de las características de la técnica propuesta, los modelos pueden ser visualizados desde cualquier dirección sin perder realismo.

Finalmente, en la tabla 6.3, se presentan los resultados obtenidos en imágenes por segundo y número de polígonos necesarios para visualizar una escena formada por distintos números de elementos, visualizando: a) el follaje con sólo geometría, b) utilizando nuestra técnica para visualizar el follaje y c) el follaje con la técnica desarrollada y todas las ramas con geometría.

Se pueden alcanzar sólo 5 fps en escenas formadas por una veintena de árboles, cuando se visualizan todas las ramas del árbol, ya que el número de polígonos necesario para representarlas es muy grande. Se observa que los ratios de visualización bajan drásticamente cuando se visualizan las ramas. Por ello, es necesario aplicar el modelo conjunto para poder obtener mejores resultados.

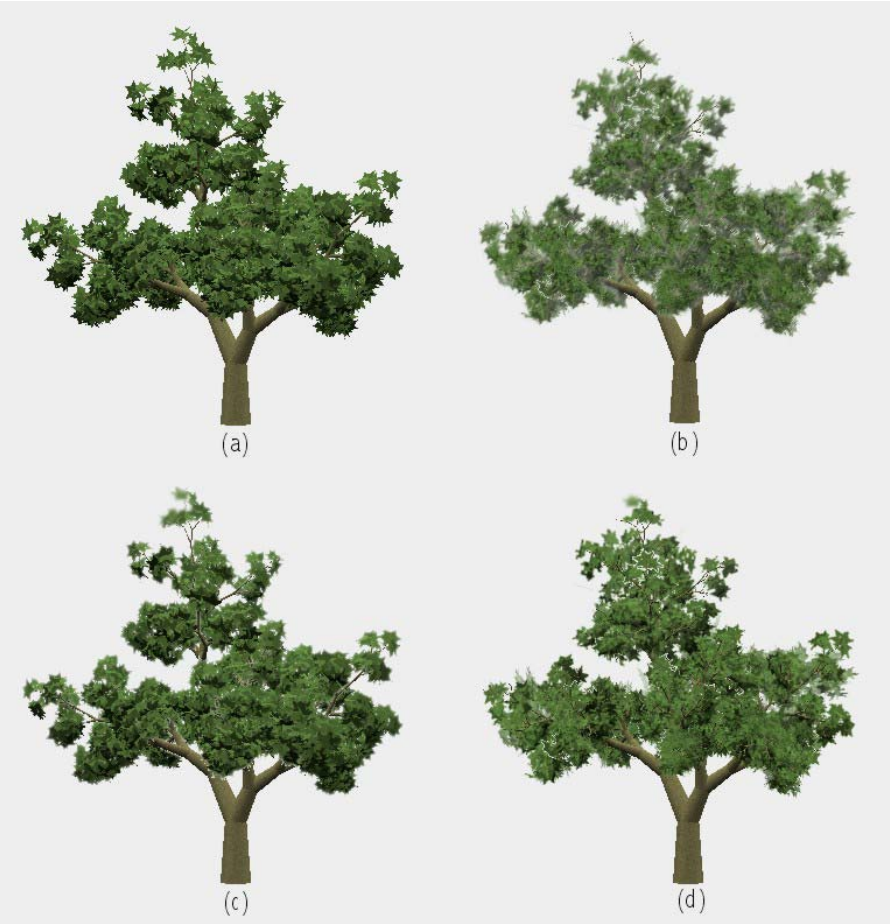


Figura 6.6: Árbol visualizado por: a) geometría con 20.000 polígonos, b) texturas en aspas con 50, c) texturas en capas con 200 y d) texturas en diagonales con 50 polígonos para el follaje

Figura 6.7: Inspección cercana de un árbol

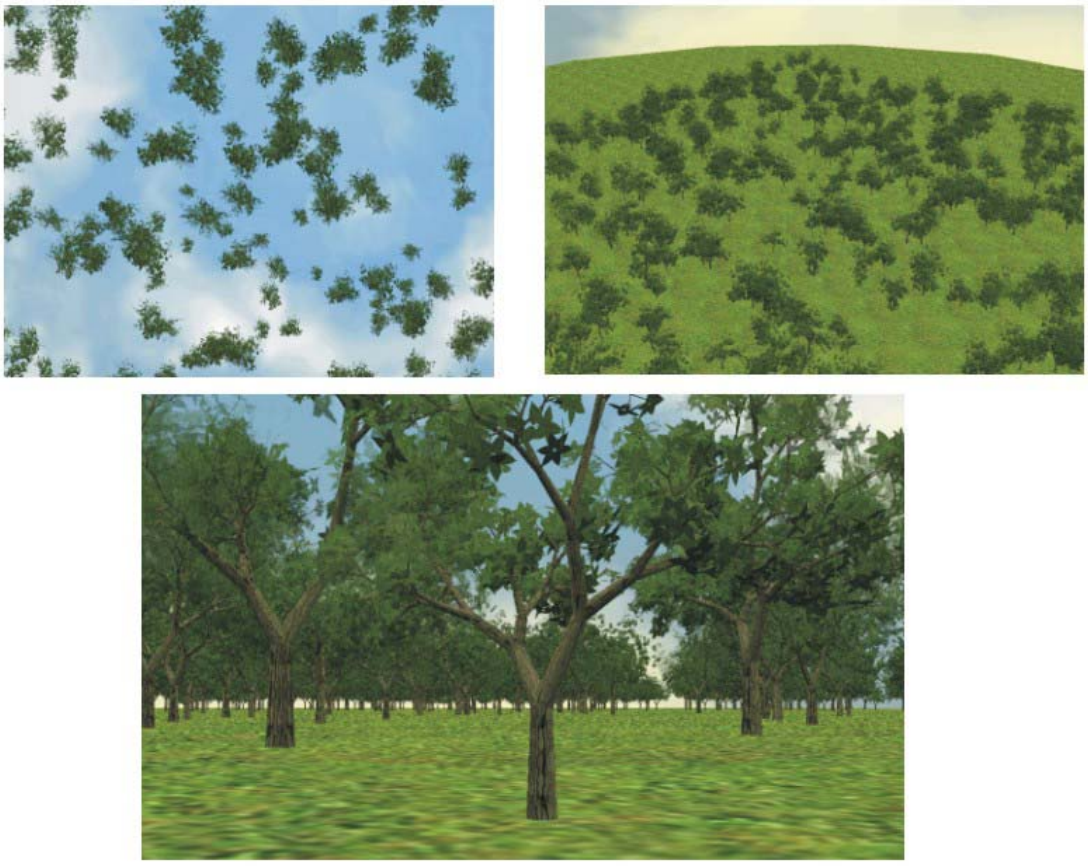


Figura 6.8: Conjunto de árboles visualizados desde distintos puntos de vista.

# Arboles		100	250	500	750	1000	1250	1500
FPS	a)	3,00	1,72	1,36	1,25	1,10	0,80	0,00
	b)	78,00	58,00	34,00	24,00	17,80	15,28	13.5
	c)	2,73	1,65	1,25	1,23	1,00	0,50	0,00
Polígonos	a)	397.600	994.000	1.988.000	2.966.090	3.956.120	4.970.000	5.964.000
	b)	1.000	2.500	5.000	7.000	10.000	14.000	16.500
	c)	644.924	1.615.000	3.256.000	4.886.000	6.515.000	8.143.000	9.772.000

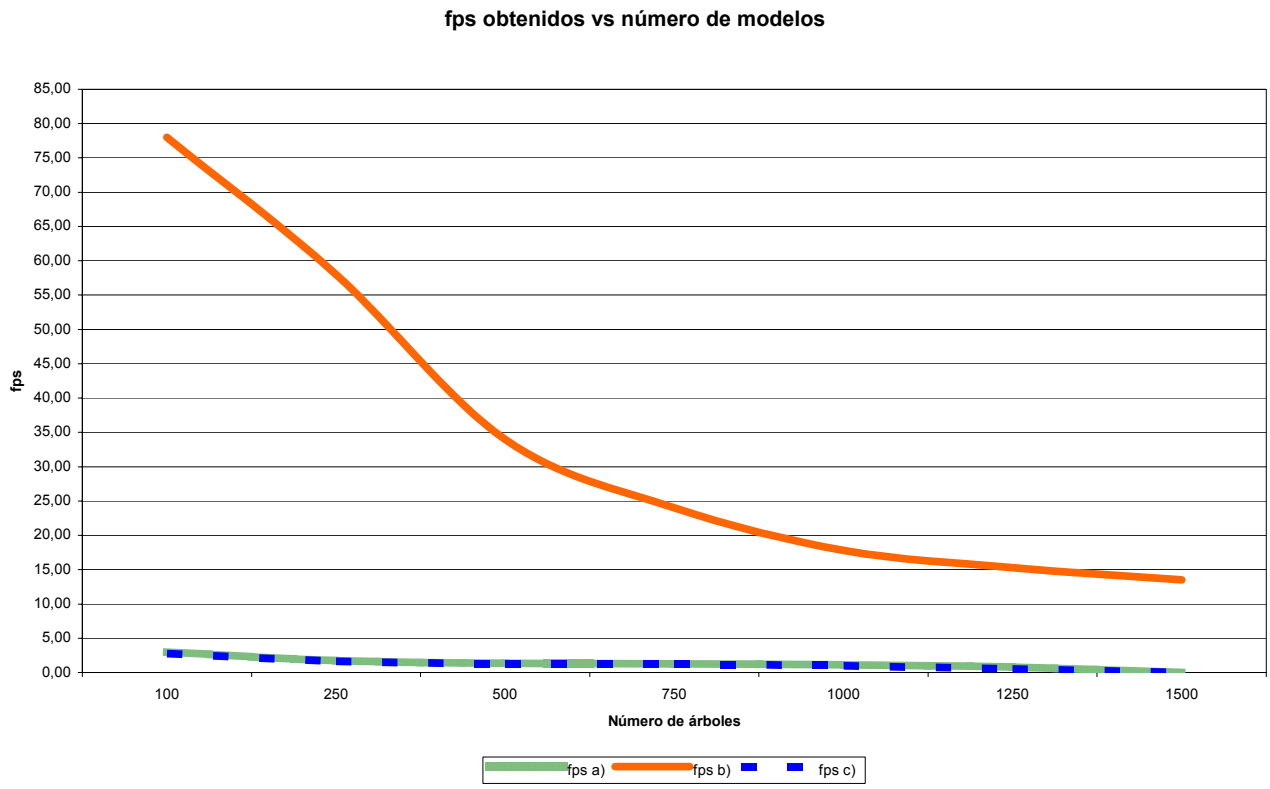


Tabla 6.3: Comparativa de resultados obtenidos, en imágenes por segundo y polígonos necesarios, para visualizar n árboles utilizando: a) visualización del follaje con polígonos, b) visualización del follaje con el modelo multirresolución y c) visualización de todas las ramas, y el follaje con el modelo multirresolución.

6.4 Modelo multirresolución para la representación del árbol

Para terminar el capítulo de resultados, se ha utilizado el mismo banco de pruebas que en el modelo anterior, pero ahora las ramas se van a representar utilizando el modelo de multirresolución procedural. En la figura 6.9 se observan diferentes niveles de detalle del modelo. En la figura 6.10 se puede apreciar una población de elementos. En la tabla 6.4 se muestran los resultados obtenidos si se trabaja con toda la geometría, con toda la geometría de las ramas y texturas para las hojas o con el modelo multirresolución propuesto.

Figura 6.9: Diferentes niveles de detalle del modelo

Figura 6.10: Población de árboles visualizada con el modelo multirresolución

# Arboles		100	250	500	750	1000	1250	1500
FPS	a)	2,00	1,30	1,00	0,60	0,10	0,10	0,10
	b)	2,73	1,65	1,25	1,23	1,00	0,50	0,10
	c)	76,00	46,70	24,40	18,45	15,00	10,5	8,00
Polígonos	a)	1.048.000	2.620.000	5.240.000	7.860.000	10.480.000	13.100.000	15.720.000
	b)	644.924	1.615.000	3.256.000	4.886.000	6.515.000	8.143.000	9.772.000
	c)	8.000	18.000	36.048	57.828	82.820	100.454	120.640

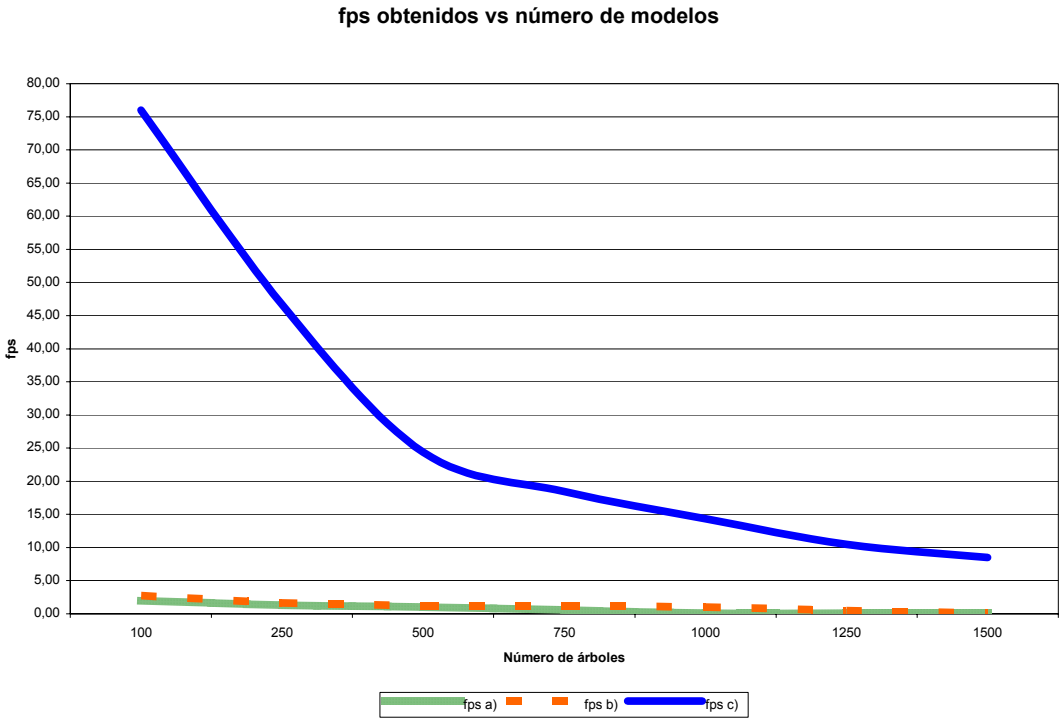


Tabla 6.4: Comparativa de resultados obtenidos para visualizar n modelos utilizando: a) todas las ramas y las hojas con geometría, b) todas las ramas, y las hojas con el modelo multirresolución de texturas y c) el modelo multirresolución conjunto para las ramas y las hojas

Se puede observar que el incremento del rendimiento es considerable. Se utilizan 120.640 polígonos para representar una escena formada por 1500 árboles utilizando el modelo propuesto, además la escena es posible visualizarla a 8 fps. Cuando no se aplica el modelo multirresolución propuesto, son necesarios 130 veces más polígonos para representar la misma escena, lo que implica la imposibilidad de realizar una visualización interactiva.

Para terminar la exposición de resultados, se presenta la tabla 6.5 donde se realiza una comparativa entre las principales características del método expuesto, el método desarrollado por Jakulin y el que utiliza el TREEFX.

	MultiArbol	Jakulin	TreeFx
Generación automática	Sí	Sí	No
Memoria ocupada por las texturas	<4 MBytes	>6 Mbytes	N/D
Inspección cercana	Sí	No	No
Inspección desde cualquier orientación	Sí	No	No
Transmisión incremental	Sí	Sí	No
FPS con 1000 modelos	>15	< 10	>15
Utilización de Billboards	No	No	Sí
Utilización de alpha blending	Sí	Sí	No

Tabla 6.5: Comparativa entre el método desarrollado y los propuestos por Jakulin y TREEFX.

Capítulo 7
Conclusiones y líneas abiertas

El principal esfuerzo que se ha realizado en esta tesis ha consistido en resolver el problema de la visualización interactiva de una población de árboles. El objetivo principal ha sido desarrollar un modelo multirresolución que permita representar un árbol, tanto las ramas como las hojas, en distintos niveles de detalle. La extracción de los niveles se realiza de forma eficiente y el coste de almacenamiento, es relativamente bajo, ya que los objetos que modela son muy complejos.

La visualización del modelo se realiza sin que se perciban, excesivamente, efectos de salto cuando se cambia entre distintos niveles de detalle. El modelo es extensible para aplicar técnicas que permitan visualizar poblaciones de elementos y se puede transmitir de forma progresiva. Por último, cabe destacar que se mejoran los modelos propuestos hasta el momento, ya que consiguiendo un número de imágenes por segundo similar, es posible la inspección de la escena desde cualquier posición y orientación, sin perder realismo.

A continuación, se enumeran las principales aportaciones de la tesis, resaltando las publicaciones que se han escrito en el desarrollo de la misma. Por último, se hace un estudio de las líneas abiertas

7.1 Principales aportaciones

Los proyectos de investigación que desarrolla la sección de informática gráfica del DSIC/UPV en el campo de edición y visualización de entornos virtuales, han desembocado en la realización de las aplicaciones GREEN y EVERGREEN. Con la primera se ha conseguido un editor de especies vegetales basado en sistemas-L. Con la segunda, se ha obtenido un navegador de entornos naturales, cuyos elementos se han editado mediante GREEN.

Una vez finalizadas estas aplicaciones, se hace necesaria la aceleración de la visualización interactiva en EVERGREEN. Por ello, se plantea la obtención de un modelo multirresolución que permita la visualización de los árboles en distintos niveles de detalle. Este modelo se ha conseguido en diferentes fases. En primer lugar, se decide que la solución debe darse por separado para las hojas y las ramas del modelo. Se realizan los siguientes modelos:

- Modelo poligonal para las ramas
- Modelo procedural para las ramas
- Modelo basado en la imagen para las hojas
- Modelo conjunto para las ramas y las hojas

A continuación se relacionan las conclusiones a las que se ha llegado con cada uno de los modelos propuestos.

Modelo de representación de una estructura ramificada mediante una malla poligonal

Este modelo permite representar de forma homogénea una estructura ramificada: árboles, plantas, sistema circulatorio, tuberías, etc. La representación homogénea del modelo nos permite:

- Solución a problemas como discontinuidades y solapamiento de geometría
- Implementación de modelos dinámicos sobre la malla: efectos de viento
- Aplicación de métodos de simplificación de mallas de polígonos

Se pueden aplicar métodos estándar de simplificación de mallas para conseguir diferentes niveles de detalle. Sin embargo, sería interesante obtener un método de simplificación que se adapte a la forma ramificada de la malla. Este método tendría en cuenta que los polígonos que forman parte de las ramas de los niveles superiores aportan menos detalle que los polígonos del tronco. Por ello, la simplificación de los primeros debe ser prioritaria.

La obtención de niveles de detalle mediante la aplicación de métodos de simplificación tiene como inconvenientes el incremento del coste espacial y la discontinuidad entre los distintos niveles que puede producir efectos visuales no deseados al realizar la transición entre dos niveles. Por lo tanto, la obtención de un modelo multirresolución poligonal que tuviera en cuenta el método de simplificación propuesto, sería muy interesante.

Para mejorar la potencia descriptiva del método también se plantea la utilización de curvas paramétricas para representar los contornos y las trayectorias de las ramas.

Modelo multirresolución procedural para la representación de las ramas

Se ha formulado un nuevo modelo multirresolución procedural que consigue representar un objeto ramificado en distintos niveles de detalle mediante una cadena de símbolos paramétricos. Esta cadena se obtiene a partir del sistema-L que modela el objeto. El número de símbolos de la cadena resultante es similar al que tiene la cadena de entrada, e incluso menor. La cadena multirresolución contiene los distintos niveles de detalle de forma incremental. Un nivel de detalle se construye a partir de los anteriores.

A partir de la interpretación de la cadena multirresolución se obtiene una estructura de datos que permite extraer de forma eficiente cada uno de los niveles de detalle para su visualización. Las pruebas que se han realizado con el modelo han sido muy positivas. Se generan las diferentes resoluciones con una gran diferencia en el número de polígonos necesarios para su visualización. La estrategia de generación de los niveles de detalle permite que el objeto se genere de forma balanceada. De modo que el aspecto visual de las ramas del árbol se mantiene en todos los niveles de detalle.

Modelo multirresolución basado en la imagen para la representación de las hojas

Esta técnica multirresolución para la visualización del follaje de un árbol se obtiene de forma automática y permite la visualización del árbol desde cualquier posición y orientación. Estas características lo diferencian de otras técnicas propuestas por otros autores.

El modelo define distintos niveles de detalle para visualizar las hojas que forman parte de un grupo de ramas. Es posible visualizar un árbol en el que se utilicen distintos niveles de detalle al mismo tiempo. Cada grupo de ramas selecciona su nivel de detalle en función de su

distancia al observador. Por lo tanto, la técnica multirresolución desarrollada es dependiente de la vista. Además, la cantidad de memoria de textura empleada es independiente de la frondosidad del árbol.

Modelo multirresolución para la representación del árbol
A partir de los dos modelos anteriores, se ha diseñado un modelo multirresolución que permite representar tanto las ramas como las hojas. Por lo tanto el modelo incorpora las características de ambos.

Es posible transmitirlo de forma progresiva. Por un lado, es posible enviar la cadena multirresolución que representa las ramas. Es posible enviar las cadenas que forman cada nivel de detalle por separado. Por otro lado, se pueden enviar las imágenes que pertenecen a cada uno de los niveles de detalle. El tamaño del modelo comprimido no llega a los 300 Kbytes. En situaciones críticas de bajo ancho de banda se puede solucionar enviando el fichero de definición del sistema L. Este fichero ocupa unos pocos bytes. En este caso, la generación de los niveles de detalle se realizará en el destino.

Es posible la navegación interactiva de una escena formada por 1500 elementos, consiguiendo 8 imágenes por segundo. Este rendimiento se puede incrementar si se utilizan *billboards* para los elementos más lejanos. Los saltos que se producen al cambiar de niveles de detalle, se minimizan al utilizar técnicas de blending. Es posible la visualización de la escena desde cualquier posición y orientación, a diferencia de otros sistemas que sólo permiten la visualización en determinadas circunstancias.

7.2 Publicaciones relacionadas con la tesis

A continuación se citan todas las publicaciones originadas hasta el momento, que se han realizado con resultados que tienen relación con el trabajo desarrollado en la tesis.

"Generación de árboles y plantas mediante Sistemas-L"
Ricardo Quirós, Javier Lluch, Roberto Vivó.
II Congreso Español de Informática Gráfica (CEIG) 1992, San Sebastián.

"Aplicaciones de los Sistemas Paramétricos Aleatorios a la generación de árboles y plantas herbáceas"
Javier Lluch, Ricardo Quirós y Bernardo Castellanos
III Congreso Español de Informática Gráfica CEIG 1993, Granada.

"Generación y visualización de atractores caóticos"
Ricardo Quirós, Javier Lluch, Miguel Chover, Carlos Larrañaga.
CONGRESO: IV Congreso Español de Informática Gráfica CEIG 1994, Zaragoza
ISBN: 84-600-9034-5.

"Aplicación de los métodos de desplazamiento y textura a la síntesis de árboles y plantas herbáceas"
Miguel Chover, Javier Lluch, Ricardo Quirós, Joaquín Huerta.
IV Congreso Español de Informática Gráfica CEIG 1994, Zaragoza
ISBN: 84-600-9034-5.

"Fractales de Sustitución Geométrica mediante Sistemas Paramétricos Aleatorios"
Ricardo Quirós, Javier Lluch, Joaquín Huerta, Roberto Vivó.
V Congreso Español de Informática Gráfica CEIG 1995, Palma de Mallorca.

"Texture, displacement and inmmersion: A model for tree rendering."
Chover,M., Vivó,R. , Quirós,R., Lluch,J.
WSCG'95 Third International Conference on Computer Graphics and Visualization, Pilsen, Feb.95, Republica Checa.
ISBN: 80-7087-187-6.

"Geometric substitution with rewriting rule systems."
R.Quirós, J.Lluch, M.Chover, R.Vivó.
Computer and Graphics, vol. 20, no.5, 1996, Pergamon Press Inc.

"Modelado y Visualización de formaciones de coral"
R. Quirós, J. Lluch, M.J. Vicent, J. Huerta
VII Congreso Español de Informática Gráfica CEIG 1997, Barcelona.
ISBN: 84-8498-429-X

"Aplicación de los sistemas de reescritura a la síntesis de imagen"
J. Lluch, M. Chover, R. Quirós, R. Vivó
Novática Vol 130 1997

"Sustitución Geométrica interactiva mediante sistemas paramétricos aleatorios"
J. Lluch, M. Chover, R. Quirós, R. Vivó
Novática Vol 140 1999

"GREEN: Gramáticas RL para la edición de entornos naturales"
María José Vicent, Javier Lluch, Ricardo Quirós, Roberto Vivó
IX Congreso Español de Informática Gráfica CEIG 1999, Jaén.
ISBN: 84-89869-81-2

"Aplication for 3D Interactive Gavimetric Modeling"
Abad, F. García-Consuegra J. Lluch,J.
AGILE'99 2ND AGILE Conference, Rome, Apr.99, Italia.

"GREEN: A new tool for modelling natural elements"
LLuch J., Vicent M.J., Vivó R., Quirós R.
WSCG'2000 International Conference on Computer Graphics and Visualization, Pilsen, Feb. 2000, Rep. Checa
ISBN: 80-7082-6012-6

"Modelado y Visualización de Elementos Naturales Basado en Técnicas de Reescritura. Aplicación a la Animación de Árboles por la Acción de Fuerzas Vectoriales Externas"
M.J.Vicent, J.Lluch, R. Vivó
XII Congreso Internacional de Ingeniería Gráfica, Valladolid 2000
ISBN: 84-8448-008-9

"Non-Photorealistic Rendering of Plants and Trees"
C.Campos, R.Quirós, J.Huerta, M.Chover, J.Lluch, R.Vivó
International Conference on Augmented, Virtual Environments and Three-Dimensional Imaging ICAV3D, Mykonos, Junio 2001

"Visualización artística de árboles y plantas"
C.Campos, R.Quirós, J.Huerta, M.Chover, J.Lluch, R.Vivó

XI Congreso Español de Informática Gráfica, CEIG'01, Gerona, Julio 2001
ISBN: 84-8458-061-X

"Modelado de estructuras ramificadas mediante malla poligonal única"
J.Lluch, M.J.Vicent, C.Monserrat, S.Fernández
XI Congreso Español de Informática Gráfica, CEIG'01, Gerona, Julio 2001
ISBN: 84-8458-061-X

"A single polygonal mesh representing a botanical tree surface"
J.Lluch,M.J.Vicent, C.Monserrat, S.Fernández
International Conference on Augmented, Virtual Environments and Three-Dimensional Imaging ICAV3D, Mykonos, Junio 2001

"The modelling of branched structures using a single polygonal mesh"
J.Lluch, M.J.Vicent, S.Fernandez, C.Monserrat, R.Vivó
IAESTED Visualization, Imaging, and Image Processing, Marbella, Septiembre 2001
ISBN: 0-88986-309-1

"Modelo multirresolución para la visualización de follaje en tiempo real"
Lluch Crespo, Javier; Fernández, Sergio;Vivó,Roberto
Technical Report, II-DSIC-24/01, Universidad Politécnica de Valencia, Diciembre 2001

7.2.1 Proyectos de investigación

El trabajo realizado en la tesis también está enmarcado en dos proyectos de investigación desarrollados por nuestro grupo de investigación, en colaboración con el grupo de informática gráfica de Universidad Jaime I de Castellón:

"Modelización y visualización de entornos naturales virtuales. Aplicación a la simulación de la visión robótica de campos frutales."
Generalitat Valenciana, GV97-TI-05-42, 1998-99

"Arquitecturas multirresolutivas de sistemas gráficos procedurales"
CICYT, TIC99-0510-C02-01, 2000-02

7.3 Líneas de trabajo futuro

Por último, y para terminar la memoria se van a presentar las posibilidades de extensión que ofrece el trabajo desarrollado.

- Modelo multirresolución para la malla única: El modelo que permite generar una única malla para representar todas las ramas del árbol, presenta como inconveniente el número de polígonos necesario para representarlo. Sería interesante obtener un método de simplificación poligonal que tenga en cuenta la topología ramificada. El modelo debería comenzar a simplificar aquellos polígonos que pertenezcan a las ramas de los niveles superiores, y simplificar en último lugar aquellos polígonos que forman parte del tronco.
- Incremento del realismo de la escena: Para la generación de las texturas que representan las hojas no se tiene en cuenta la iluminación global de la escena, se podría investigar un método que incorpore a las texturas precalculadas información referente a la accesibilidad de la luz solar a las mismas. También es posible utilizar la

proyección de las texturas sobre el terreno para crear un mapa de sombras arrojadas. Estos detalles incrementarían considerablemente el realismo de la visualización.

- Extensión del modelo para la visualización de poblaciones: La aplicación de técnicas avanzadas de oclusión al modelo es un campo interesante en el que avanzar para obtener un mayor número de imágenes por segundo. La estructura de cajas desarrollada se podría utilizar para conseguir los oclusores.
- Visualización no fotorrealista: Se presenta como una alternativa a la visualización convencional y permite un aumento en la expresividad de las imágenes generadas. Es posible visualizar de forma no fotorrealista árboles modelados mediante sistemas-L [Camp01]. El método se puede aplicar en la animación tradicional, los video juegos y las presentación de proyectos de arquitectura e interiorismo, y si se mejora su eficiencia en aplicaciones de tiempo real.

Anexo A
Bibliografía

[Abad99] Abad, F. García-Consuegra J., Lluch J., "Aplication for 3D Interactive Gavimetric Modeling", AGILE'99 2ND AGILE Conference, Rome, Apr.99, Italia.

[Abel82] H. Abelson and A. A. diSessa. Turtle geometry. M.I.T. Press, Cambridge, 1982.

[Abi94] S.S. Abi-Ezzi and S. Subramaniam. "Fast dynamic tessellation of trimmed nurbs surfaces." Computer Graphics Forum, 13(3):107-126, 1994. Proc. of Eurographics'94.

[Alia96] Alias/Wavefront; a division of Silicon Graphics Ltd. Studio V8. SGIprogram, 1996.

[Andú98] C. Andujar, Simplificación de Modelos Poliédricos, Report LSI-98-1-T, Universitat Politécnica de Catalunya, Llenguatges i Sistemes Informátics, 1998

[Anim96] AnimaTek, Inc. AnimatTek's World Builder. PC program, 1996.

[Bake72] R. Baker and G. T. Herman. Simulation of organisms using a developmental model, parts I and II. Int. J. of Bio-Medical Computing, 3:201--215 and 251--267, 1972

[Bara01] G. Baranoski,J. Rokne, *Efficiently simulating scattering of light by leaves*, The Visual Computer 17, pg.491-505, Springer Verlag 2001

[Brown96] A. Brownbill. "Reducing the storage required to render L-system based models". Master's thesis, University of Calgary, October 1996.

[Camp01] C.Campos, R.Quirós, J.Huerta, M.Chover, J.Lluch, R.Vivó, "Non-Photorealistic Rendering of Plants and Trees". International Conference on Augmented, Virtual Environments and Three-Dimensional Imaging ICAV3D, Mykonos, Junio 2001

[Chel98] M. Chelle, B. Andrieu, K, Bouatouch, *Nested radiosity for plant canopies*, The Visual Computer, Springer Verlag , pg.109-125, 1998

[Chiv97] N. Chiba, K. Muraoka, A. Doi, and J. Hosokawa. Rendering of forest scenery using 3D textures. *The Journal of Visualization and Computer Animatio*n, 8:191–199, 1997.

[Choi94] Choi, Y.; Park, K.H.: A heuristic triangulation algorithm for multiple planar contours using an extended double branching procedure, The Visual Computer 10, 1994, 372-387.

[Daub97] K. Daubert, H. Schirmacher, F.X. Sillion, G. Drettakis, *Hierarchical Lighting Simulation for Outdoor Scenes*, Rendering Techniques'97, Springer Computer Science , pg.227-238, 1997

[Deus98] O.Deussen, P.Hanrahan, B.Lintermann, R.Mech, M.Pharr and P.Prusinkiewicz, "Realistic modeling and rendering of plant ecosystems". SIGGRAPH'98, pag. 275-286, 1998.

[Eich80] P. Eichhorst and W. J. Savitch. Growth functions of stochastic Lindenmayer systems. Information and Control, 45:217--228, 1980.

[Fern99] S. Fernández: Sistema de interpolación 3D de una gramática RL para la edición de entornos naturales, Facultad de Informática,1999, Universidad Politécnica de Valencia, 46022 Valencia (Spain).

[Frij74] Frijters D., Lindenmayer, A., "A model for the growth and flowering of Aster novae-angliae on the basis table (1,0)L-systems", In L-systems, G.Rozenberg and A.Saloma, Eds, Lecture Notes in CS 15. Springer-Verlag, Berlin, 1974, pag.24-52.

[Fuch77] Fuchs, H.; Kedem, Z.M.; Uselton, S.P.: Optimal surface reconstruction for planar contours, Commun. ACM 20, 1977, 693-702.

[Gard84] G. Y. Gardner. "Simulation of natural scenes using textured quadric surfaces". *Computer Graphics (SIGGRAPH 84 Proceedings*), 18(3):11–20, 1984.

[Gort96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. "The lumigraph". SIGGRAPH 96 Conference Proceedings, pages 43-54.

[Gree93] N. Greene, M. Kass, and G. Miller. Hierarchical z-buffer visibility. InProc. of ACM Siggraph, pages 231-238, 1993.

[Gree97] D. G. Green. "Modelling plants in landscapes". In M. T. Michalewicz, editor, *Plants to ecosystems. Advances in computational life sciences* I, pages 85–96. CSIRO Publishing, Melbourne, 1997.

[Hana95] J. Hanan "Virtual Plants - Integrating achitectural and physiological plant models." In Proceedings of ModSim 95 pag 44-50

[Hart91] J. C. Hart and T. A. DeFanti. Efficient anti-aliased rendering of 3D linear fractals. *Computer Graphics (SIGGRAPH 91 Proceedings*), 25:91–100, 1991.

[Hart92] J.C. Hart. The object instancing paradigm for linear fractal modeling. In *Proceedings of Graphics Interface 9*2, pages 224–231, 1992.

[Heck94] Heckbert, P.S., Garland, M., "Multiresolution Modeling for Fast Rendering", Proceedings of Graphics Interface '94, Banff Alberta Canada, Mayo 1994, pag. 43-50.

[Herm75] G.T. Herman, G. Rozemberg, "Developmental systems and languages" Noth-Holland, Amsterdam, 1975.

[Higg96] S. I. Higgins and D. M. Richardson. "A review of models of alien plant spread". *Ecological Modelling*, 87:249–265, 1996.

[Holt94] M. Holton, ``Strands, Gravity, and Botanical Tree Imagery", Computer Graphics Forum, Vol. 13, No. 1, 1994, pp. 57-67.

[Hopp97] H. Hoppe. View-dependent refinement of progressive meshes. In *SIGGRAPH 97 Conference Proceeding*s, pages 189–198, August 1997.

[Hou98] House,D. Et al "Visualizing a real forest", IEEE Computer Graphics and Aplications, 18,1 (Jan-Feb 98), 56-65.

[Huds97] T. Hudson, D. Manocha, J. Cohen, M. Lin, K. Ho, and H. Zhang. "Accelerated occlusion culling using shadow frusta". In Proc. Of ACM Symposium on Computational Geometry, pages 1-10, 1997.

[Jaku00] A. Jakulin. Interactive Vegetation Rendering with Slicing and Blending EUROGRAPHICS 2000

[Jone91] H. Jones, D. Saupe, "Stochastic Methods and Natural Phenomena", Eurographics'91 Tutorial Note 2, 1991

[Kaji89] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional tex-tures. *Computer Graphics (SIGGRAPH 89 Proceedings*), 23(3):271–289, 1989.

[Kay86] T. L. Kay and J. T. Kajiya. Ray tracing complex scenes. Computer Graphics (SIGGRAPH 86 Proceedings), 20(4):269–278, 1986.

[Krus97] Mike Krus, Patrick Bourdot, Francoise Guidnel, and Guillaume Thibault. Levels of Detail & Polygonal Simplification. Crossroads: The ACM Student Magazine. Summer 1997.

[Kuma97] Kumar, S. Et al "Accelerated walkthrough of large spline models". In proc. Of ACM Symposium on Interactive 3D Graphics

[Lind68] A. Lindenmayer, "Matemathical models for cellular interaction in development", Journal of theorical biology, 1968, pag. 280—315

[Lind74] A. Lindenmayer. Adding continuous components to L-systems. In G. Rozenberg and A. Salomaa, editors, L Systems, Lecture Notes in Computer Science 15, pages 53--68. Springer-Verlag, Berlin, 1974.

[Lint99] Lintermann B., Deussen, O. "Interactive modeling of plants" IEEE Computer Graphics and Aplications, 19,1 (Jan-Feb 99), 56-65.

[Lluc93] J. Lluch, R. Quiros, B. Castellanos, "Aplicaciones de los Sistemas Paramétricos Aleatorios a la generación de árboles y plantas herbáceas", III Congreso Español de Informática Gráfica (Actas del Congreso), Granada, 1993, Junio, pag. 195--207

[Lluc94] J. Lluch, "Fractales de sustitución geométrica mediante Sistemas Paramétricos Aleatorios", Trabajo de 6 Créditos, Dep. Sist. Informáticos y Computación, Universidad Politécnica de Valencia, 1994

[Lluc97] J. Lluch, M. Chover, R. Quirós, R. Vivó, "Aplicación de los sistemas de reescritura a la síntesis de imagen", Novática Vol 130 1997

[Lluc99] J. Lluch, M. Chover, R. Quirós, R. Vivó, "Sustitución Geométrica interactiva mediante sistemas paramétricos aleatorios", Novática Vol 140 1999

[Lluc00] LLuch J., Vicent M.J., Vivó R., Quirós R., "GREEN: A new tool for modelling natural elements", WSCG'2000 International Conference on Computer Graphics and Visualization, Pilsen, Feb. 2000, Rep. Checa

[Lluch01a] J.Lluch,M.J.Vicent, C.Monserrat, S.Fernández, "A single polygonal mesh representing a botanical tree surface", : International Conference on Augmented, Virtual Environments and Three-Dimensional Imaging ICAV3D, Mykonos, Junio 2001

[Lluch01b] J.Lluch, M.J.Vicent, S.Fernandez, C.Monserrat, R.Vivó, "The modelling of branched structures using a single polygonal mesh", IAESTED Visualization, Imaging, and Image Processing, Marbella, Septiembre 2001, Ed. Acta Press, ISBN: 0-88986-309-1

[Lluch01c] Lluch Crespo, Javier; Fernández, Sergio;Vivó,Roberto, "Modelo multirresolución para la visualización de follaje en tiempo real", Technical Report II-DSIC-24/01, Universidad Politécnica de Valencia, Diciembre 2001

[Lueb95] D. Luebke and C. Georges. Portals and mirrors: Simple, fast evaluation of potentially visible sets. In ACM Interactive 3D Graphics Conference, Monterey, CA, 1995.

[Mars97] D. Marshall, D. S. Fussel, and A. T. Campbell. Multiresolution rendering of complex botanical scenes. In *Proceedings of Graphics Interface 9*7, pages 97–104, May 1997.

[Max95] N. Max and K. Ohsaki. "Rendering trees from precomputed Z-buffer views." In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques 9*5, pages 74–81 and 359–360. Springer Wien, 1995.

[Max96] N. Max "Hierarchical Rendering of Trees from Precomputed Multi-Layer Z-Buffers." In Rendering Techniques'96, Springer Computer Science, pages 165--174. SpringerVerlag Wien New York, 1996.

[Max97] N. Max, C. Mobley, B. Keating, W. En-Hua, "Plane-Parallel Radiance Transport for Global Illumination in Vegetation", Rendering Techniques'97, Springer Computer Science 1997, pg.239-250

[Méch96] R. Méch and P. Prusinkiewicz, ``Visual Models of Plants Interacting with their Environment," Computer Graphics (Proc. Siggraph 96), ACM Press, NewYork, 1996, pp. 397- 410.

[Méch98] R.Mech. “CPFG Versin 3.4 User’s Manual”, May 1998.

[Meye92] D. Meyers, S. Skinner, K. Sloan: Surfaces from contours, ACM Transactions on Graphics 11, July 1992, 228-258.

[Meye98] A. Meyer and F. Neyret. Interactive volumetric textures. Eurographics Rendering Workshop 1998, 157–168, June 1998.

[Meye01] A. Meyer, F. Neyret and P.Poulin. Interactive Rendering of Trees with Shading and Shadows. Eurographics Rendering Workshop 2001, June 2001

[Musg89] F. K. Musgrave, C. E. Kolb, and R. S. Mace. “The synthesis andrendering of eroded fractal terrains”. *Computer Graphics (SIGGRAPH89 Proceedings*), 23(3):41–50, 1989.

[Naur60] P. Naur et al., “Report on the algorithmic language ALGOL 60”, Communications of the ACM, 3(5) , 1960, pag. 299--314

[Neyr95] F. Neyret. A general and multiscale model for volumetric textures. In *Proceedings of Graphics Interface 9*5, pages 83–91, 1995.

[Neyr96] F. Neyret. Synthesizing verdant landscapes using volumetric textures. In X. Pueyo and P. Schro¨ oder, editors, *Rendering Techniques 9*6, pages 215–224 and 291, Wien, 1996. Springer-Verlag.

[Oppe86] P.E. Oppenheimer, ``Real-Time Design and Animation of Fractal Plants and Trees,'' Computer Graphics (Proc. Siggraph 86), Vol. 20, ACM Press, NewYork, 1986, pp. 55-64.

[Pere93] R. Perez, R. Seals, J. Michalski, *All-Weather model for sky luminace distribution-Preliminary configuration and validation*, Solar Energy Vol. 50 nº3, pg.235-245, Pergamon Press , 1993

[Pree99] J, Preetham, P. Shirley, B. Smits, *A Practical Analytic Model for Daylight*, Computer Graphics Proceedings, 1999, pg.91-100, SIGGRAPH 99

[Prus86] P. Prusinkiewicz, “Graphical applications of L-Systems”, Proccedings of Graphics Interface'86, 1986, pag. 247--253

[Prus87] P. Prusinkiewicz. Applications of L-systems to computer imagery. In H. Ehrig, M. Nagl, A. Rosenfeld, and G. Rozenberg, editors, Graph grammars and their application to computer science; Third International Workshop, pages 534--548.

[Prus90] P. Prusinkiewicz and A. Lindenmayer. The algorithmic beauty of plants. Springer-Verlag, New York, 1990. With J. S. Hanan, F. D. Fracchia, D. R. Fowler, M. J. M. de Boer, and L. Mercer. Springer-Verlag, Berlin, 1987. Lecture Notes in Computer Science 291.

[Prus92] P. Prusinkiewicz and J. Hanan. L-systems: From formalism to programming languages. In G. Rozenberg and A. Salomaa, editors, Lindenmayer systems: Impacts on theoretical computer science, computer graphics, and developmental biology, pages 193--211. Springer-Verlag, Berlin, 1992.

[Prus94] P. Prusinkiewicz, M. James, R. Mech, “Synthetic Topiary”, Computer Graphics, 1994, pag. 351—358

[Prus00] P. Prusinkiewicz, “Modeling of plants and plant ecosystem”, Communications of the ACM,Vol 43, 7.July 2000

[Pupp97] Puppo, E., Scopigno, R., "Simplificacion, LOD and Multiresolution Principles and Applications", EUROGRAPHICS'97, 1997, vol. 16, no. 3

[Ques97] P. Heckbert. Color image quantization for frame buffer display. *Computer* Questar Productions, LLC. World Construction Set Version 2. PC program, 1997.

[Quir94] R. Quirós, J. Lluch, M. Chover, C. Larrañaga, “Generación y Visualización de atractores caóticos”, Actas del IV Congreso Español de Informática Gráfica, Zaragoza, 1994, Junio

[Quir95] R. Quirós, J. Lluch, J. Huerta, M. Chover, “Fractales de sustitución geométrica mediante Sistemas Paramétricos Aleatorios”, Actas del V Congreso Español de Informática Gráfica, Palma de Mallorca, 1995, Junio

[Quir96a] R. Quirós, J. Lluch, M. Chover, R. Vivó, “Geometric substitution using Random L-Systems”, Computers & Graphics, vol. 20, no. 5, 1996

[Quir96b] R. Quirós, “Aportación al modelado geométrico de elementos naturales mediante sistemas de reescritura”, Tesis doctoral, 1996, Septiembre

[Quir97] R. Quirós, J. Lluch, M.J.Vicent, J. Huerta, “Modelado y visualización de formaciones de coral”, Actas del VII Congreso Español de Informática Gráfica, Barcelona, 1997, Junio

[Reff88] P. de Reffye et al., ``Plant Models Faithful to Botanical Structure and Development," Computer Graphics (Proc. Siggraph 88), Vol. 22, ACM Press, New York, 1988, pp. 151-158.

[Reev85] W. T. Reeves and R. Blau. Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems. Proceedings of SIGGRAPH 85, in Computer Graphics, 19, 3, July 1985, pages 313–322.

[Schm97] D. Schmalstieg, M. Gervautz, Modeling and Rendering of Outdoor Scenes for Distributed Virtual Environments .Proceedings of ACM Symposium on Virtual Reality Software and Technology 1997 (VRST'97), pp. 209-216,Lausanne, Switzerland, Sep. 15-17, 1997

[Schr92] W.J. Schroeder, J.A: Zarge, W.Du, “Decimation of Triangle meshes" SIGGRAPH´92 in Computer Graphics, 26 (2): 65-70, July, 1992.

[Shad96] Jonathan Shade, Dani Lischinski, David Salesin, Tony DeRose, and John Snyder. “Hierarchical image caching for accelerated walkthroughs of complex environments”. SIGGRAPH 96 Conference Proceedings,pages 75-82.

[Smit84] A. R. Smith. Plants, fractals, and formal languages. *Computer Graphics (SIGGRAPH 84 Proceedings*), 18(3):1–10, 1984.

[Snyd87] J. M. Snyder and A. H. Barr. Ray tracing complex models containing surface tessellations. *Computer Graphics (SIGGRAPH 87 Proceed-ings*),21(4):119–128, 1987.

[Sorr93] K. A. Sorrensen-Cothern, E. D. Ford, and D. G. Sprugel. “A model of competition incorporating plasticity through modular foliage and crown development”. *Ecological Monograph*s, 63(3):277–304, 1993.

[Suth63] I. E. Sutherland. Sketchpad: Aman-machine graphical communication system. Proceedings of the Spring Joint Computer Conference, 1963.

[Szil79] A. L. Szilard and R. E. Quinton. An interpretation for DOL systems by computer graphics. The Science Terrapin, 4:8--13, 1979.

[Trax97] C. Traxler, M. Gervautz, *Efficient ray tracing of complex natural scenes*, proceedings of Fractal 97, 4th international Multidisciplinary Conference, Denver (Colorado) 1997

[Webe95] J. Weber and J. Penn, ``Creation and Rendering of Realistic Trees," Computer Graphics (Proc. Siggraph 95), ACM Press, New York, 1995, pp. 119-128.

[Will91] L. Williams. “Shading in two dimensions”. In *Proceedings of Graphics Interface 9*1, pp 143–151, June 1991.

[Wu97] H. Wu, K. W. Malafant, L. K. Pendridge, P. J. Sharpe, and J. Walker.”Simulation of two-dimensional point patterns: application of a lattice framework approach”. *Ecological Modelling*, 38:299–308, 1997.

[Xu96] Xu, M.; Tang, Z.; Deng, J.; Zhang, S.: A new algorithm for contours connection, SPIE 2644, 1996, 341-348.

[Yoko80] T. Yokomori. Stochastic characterizations of EOL languages. Information and Control, 45:26--33, 1980.

[Zhan97] H. Zhang, D. Manocha, T. Hudson, and K. Ho. “Visibility culling using hierarchical occlusion maps”. Proc. of ACM SIGGRAPH'97, 1997.

Anexo B
Green

Variables aleatorias reconocidas:

tipoVariable::= Binomial n p | *CConstant v* | *CUniform min max* | *DConstant v* | *DUniform min max* | *Exponential f* | *FMax nombre* | *FMin nombre* | *FPoda nombre fichero* | *Normal media varianza* | *PEscalar nombre* | *Poisson landa*

Módulos que forman las cadenas:

Modulo::= nombreModulo { *parámetrosSepPorComas* }

nombreModulo::= F | *Cylinder* | *Sphere* | *Instance* | *Material* | *Repeat* | *Switch* | *Pendulo* | / | \ | & | ^ | + | - | % | *Pop* | *Push* | *?P* | *?H* | *?L* | *?U*

Significado de cada uno de los módulos:

Módulo	Significado
F{avance}	Avanza la tortuga en la dirección del vector H tantas unidades como indica el parámetro.
Cylinder{altura,radio1,radio2}	Construye un cilindro (para representar las ramas del árbol) a partir de la posición actual de la tortuga en la dirección del vector H. El primer parámetro especifica la altura del cilindro y los dos siguientes especifican, respectivamente, los radios inferior y superior del cilindro.
Sphere{radio}	Construye una esfera centrada en la posición actual de la tortuga y con el radio que se especifica en el parámetro.
Instance{índice,f1,f2,f3}	Construye una forma predeterminada (en general, un polígono o un conjunto de polígonos, que se emplearán para representar las hojas, las flores y los frutos) que varía en función del índice que se obtiene en el primer parámetro. Los restantes parámetros especifican la escala en los ejes (X, Y, Z) de dicha forma.
Repeat{número, módulos}	Repite (deriva) la lista de módulos especificada en su segundo parámetro (en realidad es una cadena que contiene dicha lista de módulos) tantas veces como indica la expresión que se obtiene en el primer parámetro.

Switch{módulos, módulos,...}	Deriva cada vez una de las listas de módulos (cadenas) que se le pasa como parámetro, empezando la primera vez por el primer parámetro, y derivando los siguientes de uno en uno tras cada llamada al método deriva de dicho módulo.
Pendulo{módulos,módulos,...}	Deriva todas las listas de módulos (cadenas) que tiene como parámetros, recorriéndolos la primera vez que se deriva en sentido adelante-atrás, y en las siguientes llamadas recorriéndolos todos en sentido contrario al de la llamada anterior.
/{θ} \{θ}	Ladear la tortuga hacia la izquierda o la derecha, respectivamente, un ángulo θ alrededor del eje H de la tortuga.
&{θ} ^{θ}	Inclinar la tortuga hacia arriba o abajo, respectivamente, un ángulo θ alrededor del eje L de la tortuga.
+{θ} -{θ}	Girar la tortuga hacia la izquierda o la derecha, respectivamente, un ángulo θ alrededor del eje U de la tortuga.
%{}	Se emplea para iniciar la fase de poda de la derivación.
Push{} Pop{}	Apila o desapila, respectivamente, el estado de la tortuga (formado por sus variables de instancia: posición y vectores de orientación H, L, U).
?P{x, y, z}	Pregunta por la posición actual de la tortuga y responde almacenando las componentes (x, y, z) de la posición de la tortuga en los respectivos parámetros.
?H{x, y, z} ?L{x, y, z} ?U{x, y, z}	Pregunta por el vector de orientación H, L o U, respectivamente, actual de la tortuga y responde almacenando las componentes (x, y, z) de dicho vector de la tortuga en los respectivos parámetros.