INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

Agentes en ecosistemas Virtuales

TESIS QUE PARA OPTAR EL GRADO DE
MAESTRO EN CIENCIAS DE LA COMPUTACIÓN
PRESENTA

JAVIER ABDUL CÓRDOBA GÁNDARA

Asesor:	DR. BEDŘICH BENEŠ	
Comité de tesis:	DR. ISAAC JUAN RUDOMÍN GOLDBERG	
	DRA. NYDIA SUPPEN REYNAGA	
Jurado:	DR. ISAAC JUAN RUDOMÍN GOLDBERG	Presidente
	DRA. NYDIA SUPPEN REYNAGA	Secretario
	DR. BEDŘICH BENEŠ	Vocal

Atizapán de Zaragoza, Edo. Mex., Enero de 2004

RESUMEN

En la presente tesis se estudia la influencia causada por agentes sobre un modelo de simulación de ecosistemas. Los agentes pueden ser visualizados como hombres que siembran, riegan, cortan plantas, pero nuestro enfoque es lo suficientemente general para pensar también en animales, insectos, eventos en el medio ambiente como lluvia, erosión, condiciones del terreno, etc.

La principal meta que se persigue es crear un marco de trabajo lo más general posible para especificar el comportamiento tanto de los elementos internos al ecosistema (plantas) como de los agentes. De este modo es posible traducir los datos estadísticos sobre ecosistemas reales como entradas a la simulación que se presenta, tanto en las plantas como en los agentes.

Se presenta la problemática encontrada en el desarrollo del simulador y las técnicas usadas para obtener los resultados deseados. Como resultados se obtienen imágenes y animaciones que no fueron creadas de un modo procedural y determinístico, sino elaborada a partir de la conducta y el conocimiento de los agentes afectando un medio dinámico.

El trabajo contribuye tanto a las áreas de las Ciencias Computacionales como a la Biología, mostrando como se puede simular un modelo complejo a partir de unas cuantas variables características y llegar a imágenes fotorrealistas.

AGRADECIMIENTOS

Antes que nada quisiera agradecer a mi asesor el Dr. Bedřich Beneš por guiarme en este trabajo, prácticamente liberándome de él.

A mis padres, Gloria y Pablo.

Gracias a mis hermanas, Nayeli y Zaire, cuya lista de correcciones fue más detallada de lo que yo imaginaba.

A los amigos que siempre me brindaron su apoyo, Yedy, Jonás, Roberto. Y a todos los que no recuerdo en estos cinco minutos de gloria.

Índice general

Índice de figuras

Índice de cuadros

1. INTRODUCCIÓN

1.1. MOTIVACIÓN

La simulación de ecosistemas es un problema interesante desde el punto de vista computacional, dada la complejidad que estos mecanismos tienen en la realidad. Sería imposible reflejar cada uno de los detalles que la naturaleza ha puesto en los sistemas biológicos.

En el presente trabajo se estudia la distribución espacial de los individuos dentro de un ecosistema. El fin es conseguir imágenes lo más realistas posibles, a partir de un modelo de competencia por el espacio que ocurre en las plantas.

El problema ha sido tratado con anterioridad y se ha comprobado su dificultad tanto por la cantidad de objetos que un ecosistema puede tener, como por lo difícil que resulta obtener imágenes que la ilustren.

Un ecosistema por muy pequeño que parezca contiene miles y hasta millones de individuos, en el presente trabajo nos limitamos al subconjuto de las plantas, y específicamente se parte de un modelo estable de simulación de competencia por el espacio para estudiar la influencia de agentes externos.

Es un hecho común que los sistemas biológicos no existen totalmente aislados, sino que deben interactuar con otros para poder sobrevivir. En nuestro trabajo deseamos estudiar la manera como agentes externos pueden modificar el transcurso normal del desarrollo de un ecosistema. Esta tarea se desea hacer del modo lo más cercano posible a la realidad y así obtener distribuciones espaciales, y posteriormente imágenes, que sean comparables a los procesos naturales.

La investigación presentada puede ocuparse para crear jardines virtuales que hayan sido originados de un modo aleatorio, pero de acuerdo a un comportamiento predefinido. Se pueden simular eventos ocurridos en un caso específico, ya que se desea la flexibilidad de poder especificar comportamiento real tanto de las plantas como de los agentes.

1.2. PROBLEMA

El objetivo de este trabajo es simular ecosistemas y la interacción de agentes dentro de ellos.

El ecosistema se modela como un espacio planar y la competencia entre plantas de distintas especies. Cada planta puede ser vista como un agente muy limitado. La simulación alcanza la estabilidad dado que los procesos de crecimiento y muerte de las plantas mantienen la población en los límites del espacio físico que se ocupa. Las plantas están parametrizadas de tal modo que se pueden recrear ciertas características que se encuentran en la realidad, tales como tiempo promedio de vida, probabilidad de liberar semillas en cierto espacio, área que ocupa cada individuo en el terreno, etc.

Además de ver a las plantas como micro-agentes, se desea introducir a otros que son externos al ecosistema. Como ejemplos podemos mencionar granjeros, insectos, animales, etc., realizar ex-

perimentos con la manera en que alteran el ecosistema y su estabilidad, al favorecer o no alguna especie de planta. Se desea que los agentes muestren un comportamiento inteligente guiado por una conducta que el usuario pueda especificar.

Cada agente consiste en un autómata dirigido mediante el procesamiento de un estado interno, de intenciones y hábitos, además de una retroalimentación que genera aprendizaje. Los agentes pueden comunicarse entre sí información acerca de sus tareas y además contienen una memoria limitada sobre lo que han visitado y el trabajo que les queda por hacer.

Existe una amplia bibliografía sobre agentes, sin embargo el enfoque de este trabajo es simplificar toda la Inteligencia Artificial (IA) necesaria para abstraer una comportamiento inteligente y usar el modelo en la figura 3.17 en la página 45 [23], basado en sensores, el motor de razonamiento bosquejado y una serie de comportamientos definidos por el usuario que describan el comportamiento del ecosistema en sí.

1.3. CONTRIBUCIÓN DE LA TESIS

La principal contribución de este trabajo es en el área de simulación de ecosistemas ya que a pesar de que existen distintas herramientas para crear modelos realistas de plantas, para simular su crecimiento y para estudiar algunos efectos de la naturaleza en las poblaciones, no existe casi nada escrito sobre la influencia de una multitud de agentes al interactuar con los elementos naturales del ecosistema, como por ejemplo granjeros que trabajan preparando campo para ser sembrado.

En este sentido nuestro trabajo se acerca a la simulación de multitudes y su interacción, solo que en este caso el objetivo es especificar cierto comportamiento general y que los agentes de algún modo realizen el trabajo sin especificar lo que cada uno debe hacer para contribuir a la meta general.

Por otro lado en el área de la simulación biológica, es posible estudiar las propiedades de una población muy específica si se tienen los datos estadísticos adecuados y así hacer simulaciones que sirvan más desde el punto de vista biológico que computacional.

Las imágenes que aquí se obtienen no son creadas proceduralmente mediante un algoritmo que decida la colocación exacta de las plantas, en cambio, es una simulación estocástica la que permite, hasta cierto punto, recrear los procesos reales que afectan a los ecosistemas y obtener imágenes con características muy realistas.

1.4. ORGANIZACIÓN DE LA TESIS

La tesis está organizada de la siguiente manera, en el capítulo 2 se da un repaso del trabajo previo tanto sobre el cómputo para ecosistemas, como de la IA requerida para realizar la tarea que nos interesa, esto es, la teoría sobre agentes con comportamiento y simulación de multitudes. Posteriormente, en el capítulo 3 se describen las técnicas usadas para recrear la simulación y tambien para darles *vida* a los agentes, así como la especificación de su comportamiento. En el capítulo 4 se describen brevemente algunos detalles sobre el software realizado, más tarde en el 5 se muestran

los resultados obtenidos y finalmente en el 6 se cierra con las conclusiones y se bosqueja el rumbo futuro que puede seguir esta investigación.

2. TRABAJO PREVIO

2.1. INTRODUCCIÓN

En este capítulo se describe en primer lugar la manera como se ha apoyado el modelado de ecosistemas mediante la computación, posteriormente se da un repaso de la IA requerida para la simulación de agentes.

También se mencionan trabajos sobre multitudes y sobre la inteligencia requerida para mostrar un *comportamiento*. Todo esto tiene como objetivo mostrar la liga entre estas dos áreas y resaltar lo novedoso de nuestro enfoque.

2.2. ECOSISTEMAS

La simulación de ecosistemas se puede atacar teniendo como objetivos de planeación, modelado y visualización, a continuación se explica qué es lo que existe en el área de las ciencias computacionales en cada caso.

2.2.1. PLANEACIÓN

En este rubro se encuentra software que ayuda a tomar decisiones forestales y agrícolas, por ejemplo para rotar cultivos, administrar la manera eficiente de cortar los árboles y obtener mayor ganancia, hacer la tierra más productiva, etc. [1, 10, 11], algunas de las herramientas hacen uso de Geographic Information Systems (GIS) [13] y hacen un uso muy básico de reglas de planeación dado que están más enfocados a una visión global del espacio plantado.

2.2.2. MODELADO

Aquí encontramos métodos generalmente basados en datos estadísticos y que ayudan a tomar decisiones más a detalle con enfoque a la sostenibilidad del ecosistema, conservación del hábitat y monitoreo del crecimiento de las especies [4, 5, 6, 14, 49]. Se constrasta el enfoque global en el que importan parámetros en promedio o bien en alguna distribución, contra el particularizado donde además de obtener resultados para una comunidad (volumen a cortar, producción total esperada, etc.) se desea modelar el desarrollo independiente de un individuo [12].

En un bosque con una sola especie normalmente se usan tablas simples que indican cómo cosechar, sin embargo cuando se mezclan especies se requiere una solución más completa para la administración forestal. Las tablas de cosecha proporcionan información muy general y asumen distribuciones uniformes de crecimiento para programación de la poda, sólo proporcionan información como diámetro promedio, altura, volumen total, etc. Las necesidades con especies mixtas

van más allá, se requiere a nivel individual información cualitativa y estructural para poder evaluar valores ecológicos.

Las principales variables que se van a medir son el desarrollo del diámetro y altura del árbol, así como la distribución al reproducirse. El tamaño de las coronas de los árboles también debe tomarse en cuenta para reflejar las necesidades de espacio que tienen para crecer las distintas especies.

Para construir el modelo se pueden tomar los siguientes enfoques [12]:

- Estadístico, basado en datos obtenidos en observaciones a largo plazo, son relativamente precisos en diámetro y altura.
- Eco-fisiológico, se modelan los procesos que intervienen en el crecimiento del árbol como la luz, nutrientes usados, fotosíntesis.
- Estructural, principalmente toman el conocimiento botánico que se tiene de las especies.

Los datos estadísticos solamente son válidos para el área ecológica de donde se tomaron así que conviene validarlos con alguna base de utlilidad antes de fiarse del modelo. Los datos faltantes se pueden recrear usando alguna distribución estadística conocida y a partir de los parámetros específicos de la región.

En el caso de estas simulaciones es importante tomar en cuenta el índice de competencia [77].

$$C_{66} = \frac{\sum_{i=1}^{N} ks66_i}{A}$$

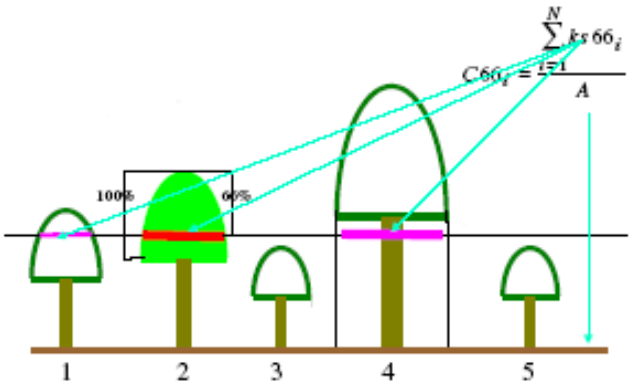


Figura 2.1: Índice de competencia de un árbol pequeño

Esto es, la suma del área que ocupan los demás árboles en el terreno a la altura de $2/3$ (o 66 %) del árbol en cuestión, dividido entre el área del terreno que se está modelando. En la figura 2.1 se puede ver un ejemplo de un árbol donde los demás son más altos y por lo tanto su área sumada será mayor, por lo tanto el índice será mayor, alternativamente en 2.2 en la página siguiente se puede observar que cuando el árbol es más alto, el área de los demás no afecta tanto y la competencia se reduce para él. El área al 66 % es tomada dado que allí comienza la sombra, además

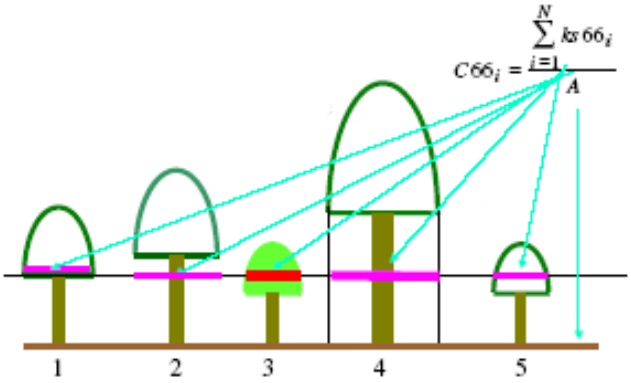


Figura 2.2: Índice de competencia para un arbol alto

allí es la anchura máxima [32, 33] y se ha mostrado empíricamente que la correlación del índice de competencia con el crecimiento es máxima también a esa altura [25].

Para representar la distribución de la población se usan esquemas en donde se presenta una visión superior del terreno plantado y el área de cobertura de la planta 2.3. De este modo se representa también la competencia por el área ecológica que es en donde la planta se puede desarrollar.

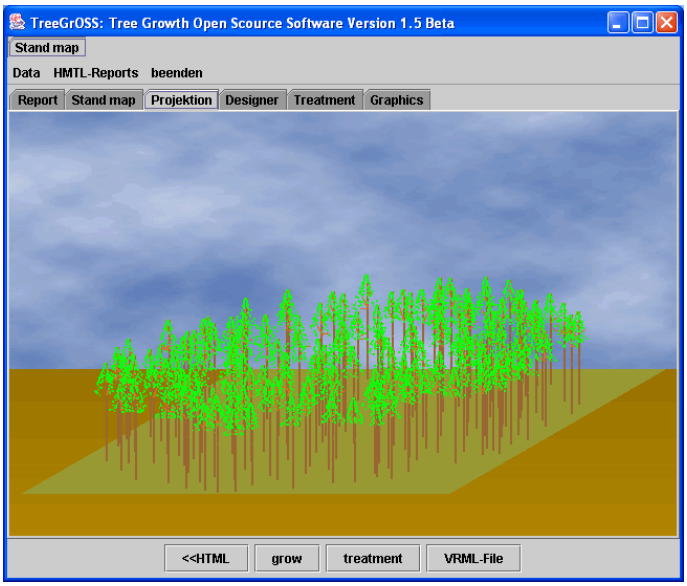


Figura 2.3: Proyección en 3D del campo en TreeGross

2.2.3. VISUALIZACIÓN

Desplegar el resultado de la simulación de un ecosistema puede servir para alcanzar diversos objetivos:

- Con fines ornamentales, en la planeación de jardines.

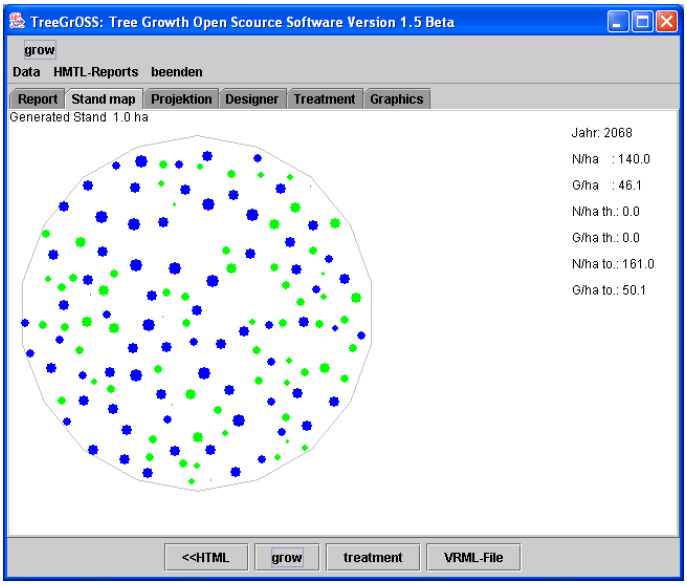


Figura 2.4: Representación de un terreno con árboles de distintas edades

- En la educación, mostrar los resultados de un fenómeno importante que afecta a una comunidad de plantas, un ecosistema extinto.
- Visualizar las operaciones de decisiones forestales como cortar o replantar zonas de un bosque, e incluso monitorear desastres como incendios [8].
- En planeación urbana, diseño arquitectónico, la interacción con habitaciones humanas.
- En arte y entretenimiento, el ecosistema con un modelo subyacente puede aparecer más realista con poca o ninguna interacción del artista.

Existen dos vertientes que se han seguido en el problema de dibujar ecosistemas, el primero se enfoca al dibujo de plantas individualmente y el segundo al de ecosistemas en conjunto.

En el primer caso los métodos usados son sistemas L, los basados en partículas y los de hilos.

Sistemas L Los *sistemas L* (o de Lindenmayer) [63] son capaces de simular de modo realista la manera como ocurre el crecimiento en las plantas [42, 45, 62, 63]. y recientemente se han extendido mediante los sistemas L abiertos para poder incluir la interacción de las plantas con el medio ambiente [54, 61] ya que los anteriores solamente servían para modelar las plantas de modo aislado [17, 58, 70].

Un sistema L es muy parecido a lo que en teoría de lenguajes se llama derivación del lenguaje; a partir de la descripción de las reglas de la gramática (llamados axiomas), y una serie de sustituciones repetitivas, se obtienen cadenas de los símbolos especificados. La característica particular de estos sistemas es que las cadenas son interpretadas como componentes de la planta o bien como operaciones geométricas en el caso de usar geometría de tortuga [16].

Partículas Los sistemas de partículas [18, 64] están basados en ejecutar ciertas acciones tomando en cuenta una detección discreta del medio circundante, por ejemplo la planta puede seguir una guía y rodearla como lo haría una enredadera, también se puede hacer lo mismo siguiendo la luz, etc. El espacio se divide en voxeles [21, 39] y de acuerdo al comportamiento predefinido en un autómata (seguir regiones de humedad, evitar obstáculos, etc.). De este modo también se puede simular la influencia de la luz sobre el crecimiento de la planta, dado que es posible evaluar la cantidad de iluminación en los voxeles [18, 19, 54, 59].

Hilos La técnica de hilos [41] usada para modelar árboles como una serie de hilos que se alargan por las ramas, extiende el modelo de tubería [37, 47] generalmente relacionado con estructuras naturales como venas y ríos. En esta técnica existe una zona donde se condensan los hilos (tronco) pero se van separando en manojos conforme se extienden las ramas y termina en los hilos (o ramas) más delgadas. De este modo se pueden ir modelando ciertas fuerzas relacionadas con el crecimiento de los árboles como la gravedad, el equilibrio del tronco y otras que van determinando la forma que toma el tronco.

Simplificaciones La síntesis de una imagen de un ecosistema es una tarea complicada dado el orden de magnitud de los polígonos que se generan (millones de polígonos) por esta razón se ha buscado distintos métodos para hacer más sencillo este proceso.

El instanciamiento [35, 71] es un método en el que los objetos que aparecen varias veces en la imagen solamente se procesan una vez y las veces que se repiten se convierten en instancias suyas, solamente se varían transformaciones geométricas sencillas como trasladar, rotar y escalar. Este método simplifica el uso de ray tracing y se puede jerarquizar para incrementar más la velocidad [43]. En [34] el espacio se divide en celdas, mismas que se recorren a distintos niveles de resolución y se van rendereando conforme a los niveles de detalle necesarios. En [35] se combinan varias técnicas en las distintas etapas del desarrollo del ecosistema: el dibujo del terreno mediante un editor gráfico, la distribución de las plantas ya sea de modo manual o mediante simulación o una combinación, la reducción de la complejidad de la escena usando instanciamiento aproximado ya sea de plantas completas o de partes de las plantas y se da un ejemplo de los resultados que se obtienen en la figura 2.5 en la página siguiente.

Al simular plantas mediante partículas [15] lo que se ha hecho es que al procesar secuencialmente la imagen, se van usando los objetos y de inmediato se descartan, así que no es necesario conservar todos los elementos en memoria y se disminuye la complejidad. Por otra parte se pueden utilizar estrategias de Level Of Detail (LOD) y trabajar con modelos en donde se degrada la exactitud geométrica como en [76] donde se hace un estudio de las características geométricas del dibujo de árboles y plantas, pero al componer una imagen con muchos objetos (árboles) se cortan detalles a la distancia y las imágenes no se degradan ostensiblemente (fig. 2.6 en la página siguiente).

La creación de los modelos puede simplificarse dependiendo del LOD requerido [46], esto es llamado un algoritmo de multiresolución procedural y usa sistemas L paramétricos, donde se incluye la información de la relevancia de los componentes de un árbol para poder dibujarse al

Figura 2.5: Paisaje logrado al combinar varias técnicas de dibujo de plantas

detalle adecuado a lo que se requiere renderear (Fig. 2.7 en la página siguiente).

Figura 2.6: Un bosque modelado a partir de un estudio de las características geométricas de las plantas

Para reducir aún más la complejidad de la imagen se pueden usar impostores y en vez de dibujar el modelo completo se usa una imagen previamente compuesta [72, 73], e incluso se ha trabajado en técnicas para interpolar entre distintos puntos de vista de las imágenes y así obtener mayor realismo al no todas ser idénticas y usar transformaciones en las características de la imagen además de las transformaciones geométricas [48, 50].

En el caso de tener una iluminación compleja, este método resulta pobre, dado que recomponer las texturas resulta de nuevo difícil. De este modo se puede optar por precalcular las distancias y usar el z-buffer para combinar una serie de imágenes predefinidas con las características que se desean del ambiente [51], ver figura 2.8 en la página 18. El método de z-buffer ha sido extendido usando un muestreo aleatorio y así reduciendo la complejidad de los datos que se han de analizar [75].

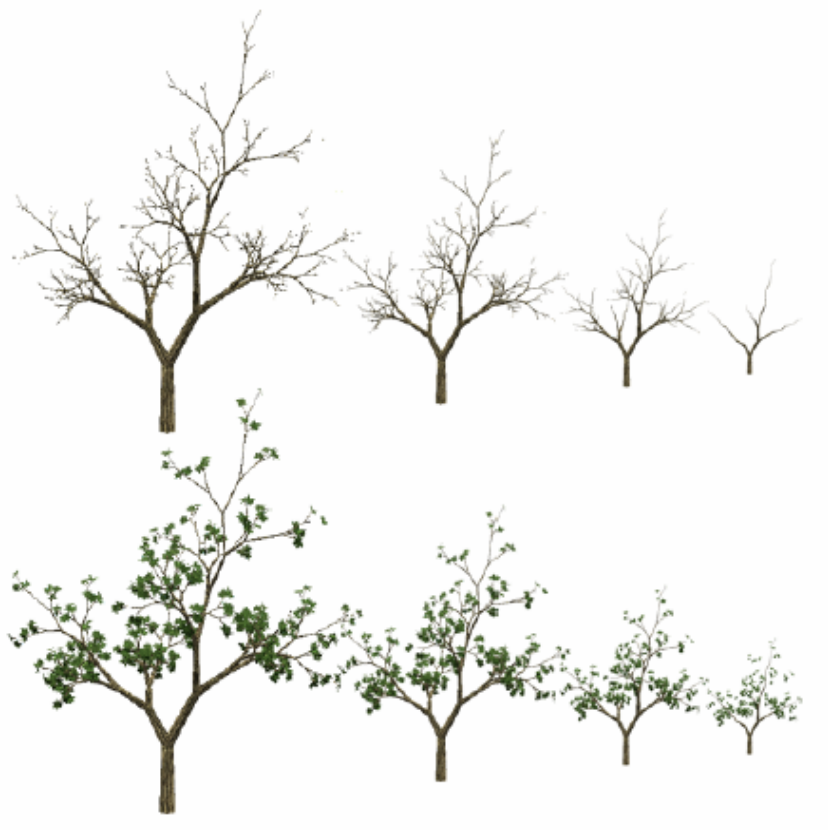


Figura 2.7: Árboles rendereados a distintos niveles de resolución dependiendo de la distancia

En [36] se usa un enfoque híbrido, por una parte se usa un modelo basado en física para calcular la respuesta de las ramas de un árbol al viento, y otro procedural en el que a menor nivel de detalle los comportamientos no requieren tanta exactitud. El principal problema al que se enfrenta en este caso es a una transición suave (sin saltos en la animación) al cambiar entre un modelo y otro. El resultado que se obtiene es a un nivel interactivo usando 256 árboles (figura 2.9 en la página 19).

2.3. AGENTES PARA GRÁFICAS COMPUTACIONALES

Para la simulación de ecosistemas se utilizan diversas herramientas de IA desde los modelos con neuronas artificiales [53] en las que se hace analogía de operadores booleanos contra una red de neuronas artificiales interconectadas. Más tarde se relacionan los pesos de las salidas de las neuronas con el concepto de aprendizaje [40, 56]. También los métodos de generar-y-probar o de búsqueda exahustiva donde se determina el paso siguiente en un universo de búsqueda (uno intermedo a la solución del problema) y se hace la prueba para verificar si es correcto o se debe descartar. Sin embargo estos métodos rápidamente caen en la explosión exponencial así que se desarrollaron técnicas para acortar el universo de búsqueda y así encontrar soluciones más rápido (heurísticas). [52, 57, 68, 69].

En cuanto a la representación del conocimiento, se usan métodos como los de *frames* que son un medio para organizar el conocimiento a gran escala, originalmente pensado para clasificar ob-

Figura 2.8: Dos imágenes recalculadas a partir del modificar los valores del z-buffer

jetos en problemas de visión computacional. En este enfoque los datos y clases se organizan en una jerarquía similar a una taxonomía biológica, muy adecuado a los ecosistemas reales. El conocimiento acerca de cómo los eventos afectan los campos existentes en el *frame* está encapsulado en él mismo [55].

Algunos sistemas de simulación tienen bases de conocimiento y predicción estadística que se asemeja a los sistemas expertos. Estos contienen una serie de reglas y hechos aportados por expertos humanos, que sirven para encontrar una solución a cierta pregunta que se hace al sistema sobre un campo de conocimiento muy acotado. Sin embargo un problema identificado es que es difícil abstraer el conocimiento de los expertos, dado que muchas veces existen inconsistencias u omisiones que son resueltas hasta el momento de tomar una resolución.

Al parecer el concepto que capta mejor la esencia de un agente, es el encontrarse a sí mismo ubicado dentro de un medio y estar *consciente* de su independencia.

Entre métodos alternativos para el control requerido en agentes se puede encontrar la arquitectura de subsunción que originalmente se usó en robots [31]. Esta arquitectura no están basada en la descomposición de módulos funcionales (i.e. sensores, percepción, modelado, planeación, ejecución de tareas, control de motores y efectores), sino directamente en conexiones desde los sensores hacia una arquitectura en capas y paralela y de allí a los efectores. Lo que se busca con

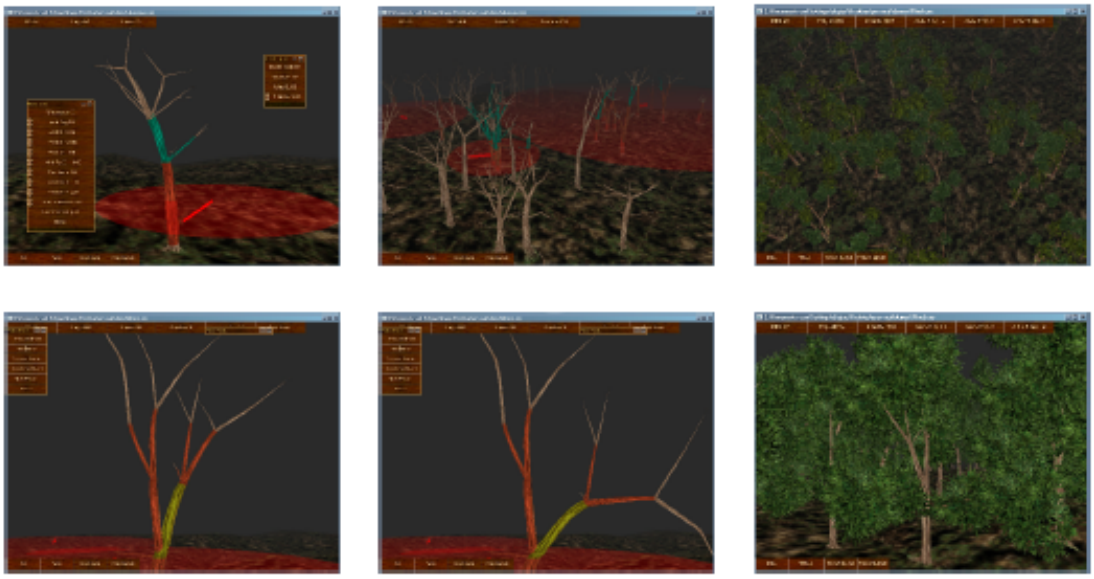


Figura 2.9: Bosque interactivo logrado al combinar un modelo de animación procedural con otro basado en física

este enfoque es cumplir con varias metas al mismo tiempo, tener a sensores simultáneos, que el sistema sea robusto en el sentido de que una modificación en la construcción del robot o del mundo no requiera reconstruir todo y también ser extensible para futuras mejoras al agente.

Se parte de ubicar niveles de competencia y capas en la arquitectura que estén enfocadas a darle al agente las capacidades deseadas. Niveles de competencia son, por ejemplo, que el robot evite colisiones con objetos estáticos o dinámicos, que se mueva sin golpear con todo, seguir una ruta usando las capacidades anteriores, aceptar cambios en los objetos del ambiente, etc.

La implementación [30] consiste en una serie de módulos interconectados, basados en un Augmented Finite State Machine (máquina de estado finito aumentado) (AFSM) compuesto. Cada módulo tiene una serie de entradas y salidas con un *buffer* para almacenar la entrada más reciente (si llegan nuevas se pierde la anterior). Cada módulo tiene además una entrada supresora por medio de la que otro módulo puede tomar precedencia en la señal y una salida inhibidora por medio de la cual otro módulo puede reemplazar totalmente la salida normal durante un tiempo específico.

Un conjunto de módulos normalmente sirve para llegar a un nivel de competencia y dado que los módulos son de consecutiva complejidad, además asumen (de ahí el nombre *subsumption*) la competencia de los niveles inferiores. Para añadir más niveles de competencia la máquina creada se aumenta usando las entradas supresoras y salidas inhibidoras, pero nada se cambia en los niveles inferiores.

Esta arquitectura [27, 28, 29] implica que no es necesaria hacer una representación o abstracción del mundo real para poder operar en él, sino que el mundo en sí podía ser la representación en el que el agente se ubica. Esto es llamado *Physical Grounding Hipothesis* i.e. para construir un sistema inteligente es necesario que su representación esté sustentada en el mundo físico. Esto se puede interpretar para agentes de software como el mundo en el que vaya a ”vivir” el agente. La

hipótesis apoya el hecho de que sea superflua una representación simbólica proponiendo que el mundo es el mejor modelo y al parecer la arquitectura de algún modo contiene el modelo de modo implícito.

No se require un lenguaje simbólico para representar el mundo sino que las capas tienen acceso libre a los sensores (múltiples) y desde allí se opera para crear los comportamientos deseados en las capas de competencia que se establezcan. El modelo trabaja con bastante rapidez ya que la implementación consiste en una serie de máquinas de estado finito y el procesamiento se hace en paralelo una vez que se insertan en un robot o agente.

Por otro lado existe el argumento de la evolución, dado que la naturaleza se tardó tanto tiempo en dotarnos de las facultades básicas, como caminar, sensar, interactuar con el mundo, precisamente en esas tareas es en lo que hay que invertir más. La inteligencia que se busca es parecida a la de un insecto, pero uno del mundo real, que interactúe con su entorno y tenga las capacidades básicas de subsistencia.

2.4. ANIMACIÓN BASADA EN COMPORTAMIENTO

La animación consiste en una secuencia de imágenes producidas por medios sintéticos a partir de variar múltiples parámetros en una simulación.

Según Reynolds [65] la animación, puede ser de dos tipos:

1. Metas específicas definidas, por ejemplo en producción comercial cuando se requiere que los actores cumplan con un script predefinido.

2. Guiada por conducta, cuando la animación se produce como resultado de una simulación, al experimentar con el ”¿qué pasaría si...?”. En el presente trabajo este tipo de simulación va a ser el que nos interesa.

El uso inteligencia artificial para guiar lo que se simula, resulta ser una evolución de las técnicas de animación tradicional [60], donde las metas se deben descomponer en una serie de subtareas y se han de ir cumpliendo una a una para completar la meta original.

Un nivel más de abstracción es especificar las metas en un script donde se define un escenario para dejar que la animación se vaya ejecutando según los mecanismos de más bajo nivel (las subtareas). Se deben superar los retardos al resolver la simulación y tratar de llegar a un nivel de interacción de tiempo real para que el animador reciba retroalimentación inmediata sobre lo que modifica en el modelo.

Un paso más adelante es la animación guiada por comportamiento, que trata de simular las características de percepción/decisión de los elementos que participan en la simulación y hacerlos interactuar. Este tipo de simuladores pueden clasificarse según el tipo de percepción que hacen y las decisiones que toman. La percepción del medio en que los agentes interactúan se realiza mediante sensores que bien pueden operar sobre el ambiente o de otro modo interactuar con la representación que se tiene sobre él. Se pueden usar algoritmos reactivos que tomen una decisión

directamente a partir de lo que se percibe, o bien simular un proceso psicológico, etológico o a partir de la observación de la naturaleza.

En este punto del procesamiento es donde se ocupan las técnicas de IA como sistemas expertos, redes de conocimiento, optimización, planeación, autómatas, aprendizaje, etc.

En [74] se presenta una plataforma basada en bitmaps en la que los colores codifican espacios, comportamientos, puntos de interés, etc. es decir, una representación del mundo específica para el agente pero que ayuda en las tareas básicas de cálculo de rutas, detección de colisiones y en el modelo presentado en [23] presenta la interacción de agentes con un ambiente virtual de ecosistemas usando un esquema de sensores–efectores.

A partir de la animación basada en comportamiento también se ha sugerido un nivel más [38], quedando por niveles de abstracción como sigue:

1. Geométrica
2. Cinemática
3. Física
4. Basada en comportamiento
5. Cognitiva

Sobre la animación cognitiva, se tiene control sobre lo que los caracteres animados (o agentes) saben, en cómo lo aprenden y cómo se puede usar esa información para planear tareas complejas. De este modo se puede programar (en Congnitive Modeling Language (CML)) un script enfocado a lo que el animador desea que suceda y los agentes decidirán a través de su propio conocimiento el cómo va a ocurrir. Para llegar a esta meta se divide el *modelado cognitivo* en dos subtareas: *especificación del dominio de conocimiento* y *dirección de caracter*, diferenciando el cómo el agente percibe o sensar su medio y el razonamiento que debe llevar a cabo para cumplir con su meta. En los resultados se pueden ver comportamientos muy complejos y cómo los agentes deciden acerca de las tareas que han de realizar (fig. 2.10 en la página siguiente).

El CML es el lenguaje que se desarrolló, se basa en el cálculo situacional, pero en una versión adaptada a robótica. Mediante esta herramienta lógica lo que se trata de hacer es determinar el estado (o *situación*) de las cosas en el mundo (o universo del discurso) y se realizan aserciones sobre lo que es cierto y sobre el cómo una acción cambiaría la situación. Todo lo anterior se especifica usando el formalismo de la lógica de primer orden. Por ejemplo para especificar propiedades de objetos se usan ***fluentes***, por ejemplo $blanco(x, s)$ especifica que el objeto x es $blanco$ en la situación s. Luego se pueden especificar cómo acciones modifican las situaciones $s' = pintar(x, s)$, para decir que la situación s' ha sido modificada habiendo ejecutado $pintar$ en la situación s. Ahora bien, usando este cálculo rápidamente se cae en el ***frame problem***, dado que para ejecutar cualquier acción existe una cadena muy larga de condiciones que hay que tomar en cuenta, para modelar un mundo ”realista”.

Para resolver este problema se usan fluentes Interval-Valued Epistemic (IVE) en el que usando aritmética de intervalos, se codifica la incertidumbre sobre el mundo dentro de un rango sobre el

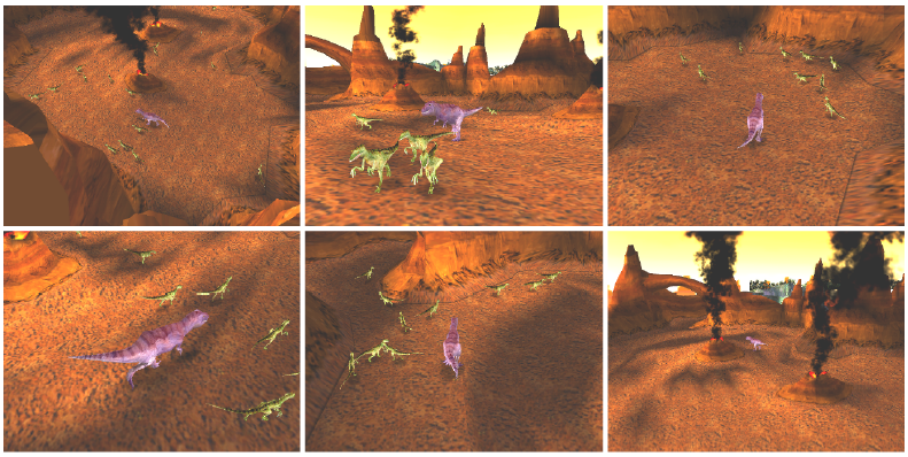


Figura 2.10: Animación de un T-Rex persiguiendo a Raptors, animación cognitiva

que se pueden hacer aseveraciones lógicas correctas. Por ejemplo en cierto modelo se podría decir que un objeto tiene una velocidad en el intervalo de $(10, 50)$ en una dirección también representada mediante un intervalo. Esta técnica es de uso común, por ejemplo, en planeación y juegos.

De este modo combinando el CML con la representación del mundo mediante los fluentes IVE se obtiene como resultado que se pueden crear scripts en el que el agente a partir de un limitado conocimiento del mundo puede ”razonar” sobre él y tomar decisiones para llegar a un objetivo predefinido.

Es de notar que en los resultados que se obtienen incluso la cámara es un agente que tiene un conocimiento sobre el mundo, un comportamiento y que puede tomar decisiones usando su motor de inferencia en CML.

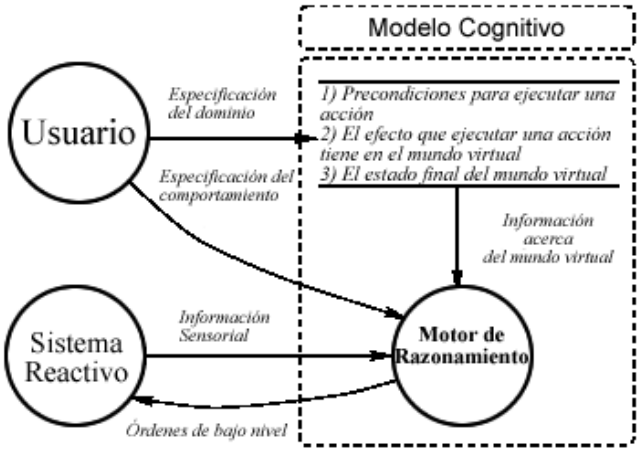


Figura 2.11: Interacción entre el modelo cognitivo y el sistema reactivo de conducta

Usando la animación basada en un modelo cognitivo se pueden especificar metas a alto nivel y los agentes a partir de un motor de conocimiento razonan sobre las conductas que deben tomar para llegar al objetivo que se les plantea.

2.5. BOIDS Y MULTITUDES

Reynolds [65] propone un sistema llamado Actor/Scriptor Animation System (ASAS) como una extensión de Lisp. Como metas básicas, el sistema buscaba paralelismo, independencia y sincronización en tareas, todo enfocado al control de distintos objetos interactuando en una animación.

ASAS tiene una influencia muy parcada de la inteligencia artificial y en particular de la teoría de agentes. Como una extensión de Lisp, incluyó una serie de operadores básicos, que dada la expresividad sintáctica y semántica del lenguaje, pueden combinarse para obtener operadores más abstractos y complejos. Algunos de los objetos, por ejemplo pueden ser definidos proceduralmente y otros guiados por los datos, pero ambos teniendo características geométricas necesarias para ser desplegados y además ser modificados usando operadores comunes. Así los actores o agentes tienen cierta homogeneidad en su representación, pero individualidad en su estado propio.

ASAS incluyó el concepto de paso de mensajes para comunicarse entre objetos –llamados actores allí–. Básicamente un actor es lo que hoy llamamos instancias de clase, código que cuando se ejecuta contiene ciertas propiedades y métodos para operar con sus datos. En el script de animación un actor representa un elemento visible y su conducta. Aquí es donde la paralelización entra en juego como una separación de los procesos que toman lugar. En Lisp un actor se implementa conceptualmente como una ”cerradura” o ”proceso”.

En el campo de la simulación basada en conducta ASAS es muy fuerte ya que se da una independencia relativa a los actores y se espera que reaccionen como si fueran caracteres reales, hablando entre ellos usando mecanismos de paso de mensajes y ajustando sus variables locales al ambiente global.

Lo que ASAS proporciona es una integración de un sistema de gráficas computacionales a un lenguaje de programación muy expresivo. El resultado sufre del aspecto orgánico que produce la influencia de un modelador, dado que todo se define proceduralmente. Además el sistema introduce el concepto de un actor o bien agente y destaca la importancia del control en los elementos de la animación. El sistema fue usado para construier animaciones muy vistosas e imágenes como las que se mostraron en la película TRON de Disney y sienta un precedente de un sistema con integración de lenguajes en campos de dominio distintos.

Reynolds [66] intenta simular las complejas interacciones y movimiento que se puede observar en agentes biológicos agregados en la naturaleza. Se realiza usando un modelo conductual distribuido en el que no existe estado global, sino una combinación de actores que tienen alguna percepción local del ambiente y cierta conducta programada sobre cómo reaccionar. Los resultados muestran cómo una simple interacción produce una simulación muy convincente sobre multitudes reales.

Las características de las multitudes (término aquí usado para referirse a la agregación de animales en agua, aire o tierra) incluye:

- Sincronización relativa a pesar de la falta de un estado global.
- Elementos discretos producen en conjunto una apariencia muy fluida.
- Es simple en gran escala pero muy complejo en los detalles.

- Aparenta ser el resultado de un comportamiento aleatorio pero resulta ser sincronizado.
- Uno puede pensar que existe un sistema de control central pero esto es imposible o imprácti-co.
- La conducta global está basada en la percepción local de cada agente.

Este modelo asume que la manada resulta de la interacción entre agentes. Se dan las reglas básicas de conducta y una percepción de su estado local o ambiente muy cercano (simulando mecanismos de percepción). Cada elemento será llamado *boid* como una contracción de *bird–oid* (se podría traducir como *pajaroide*).

Figura 2.12: Particulas simulando una parvada

La simulación está basada en una generalización de sistemas de partículas. Tradicionalmente se trabaja con puntos u objetos puntuales, pero aquí una partícula representa un objeto (un actor), cada uno con variables locales, sistema de coordenadas local, orientación y conducta. Otra diferencia fundamental es que en un sistema de partículas hay un estado global y aquí la conducta depende tanto de variables externas e internas. Este *estado* es encapsulado como una estructura de datos almacenada en el objeto. La conducta es dada como una serie de reglas y/o condiciones. Aquí se manifiesta el concepto de *actores*, estructuras que incluyen procesos, procedimientos y estado. Se comunican por medio de paso de mensajes, esencialmente como si fueran entidades independientes o *computadoras virtuales*.

Para construir un banco virtual de *boids* se parte de un conjunto de objetos con la capacidad de volar, pero además se agregan varias conductas virtuales:

- Evitar colisiones con los boids más cercanos.
- Imitar la velocidad de los vecinos.
- Tratar de conservarse cerca de las entidades circundantes.

Figura 2.13: Boids (*bird-oids*)

2.6. RESUMEN

Se ha podido comprobar en este capítulo que existe ya mucho trabajo en la simulación de ecosistemas, con distintos fines e incluso herramientas comerciales que ayudan en distintas etapas de la planeación de campos sembrados. También se repasaron trabajos realizados para la simulación de multitudes y de agentes inmersos en ambientes virtuales, con percepción y características sensoriales sobre su medio, así como efectores que les permiten retroalimentar este medio e interactuar entre ellos. El enfoque de *boids* representó un trabajo fundamental ya que a partir de ese estudio han surgido numerosas aplicaciones, demostrando que no es necesario tener un control total sobre los elementos de una simulación.

3. SOLUCIÓN DEL PROBLEMA

3.1. SIMULACIÓN

3.1.1. ECOSISTEMAS

Para modelar el ecosistema usamos el modelo local-a-global de vida artificial [44] en el que se representan las características de la población desde los individuos para obtener resultados en el ecosistema completo. Cada planta tiene reglas de comportamiento que hacen que *viva*, afectada por otras plantas y algunas condiciones externas. La planta puede básicamente crecer, reproducirse y morir, durante su *vida* surge el fenómeno de distribución espacial y competencia por el espacio y/o recursos.

La simulación consiste en un área plana donde se colocan un cierto número de plantas de modo aleatorio uniforme y se les da un valor de edad distinto de zero, también aleatorio. En [35] se explica cómo obtener distintas distribuciones aleatorias con propiedades específicas, no sólo la uniforme. Esto puede ser incorporado con facilidad en nuestra implementación. Las plantas irán creciendo de edad y de área (vecindad ecológica) hasta su madurez, después liberan semillas (una sola vez en su vida) y el ciclo continúa. Inicialmente el sistema no es estable, pero poco a poco converge al equilibrio de población entre las especies, como se puede observar en la sección 3.1.3 en la página 31. Las plantas crecen de acuerdo al algoritmo 1.

Algoritmo 1 Simulación del ecosistema

mientras $Tsimulacion < Ttotal$ **hacer**
  para todo p donde p forma parte del conjunto de plantas **hacer**
    liberar semillas si p no lo ha hecho y está en edad de hacerlo
    si p está fuera del ecosistema **entonces**
      eliminar p
    fin si
    detectar colisión entre p y las demás plantas $\rightarrow$ el perdedor muere
    si $tmuerte > Tsimulacion$ **entonces** {es decir, la planta ha muerto}
      eliminar p
    fin si
    hacer crecer p
  fin para
  el tiempo de la simulación se incrementa $Tsimulacion = Tsimulacion + \Delta t$
fin mientras

Debido a que cada planta lanza sus semillas dentro de cierto radio aleatorio, y luego las nuevas plantas crecen, compiten entre ellas y vuelven a iniciar el cliclo, ocurre en la simulación la

formación de racimos o *clusters* de las plantas de la misma especie, tal como se muestra en la figura 3.1.

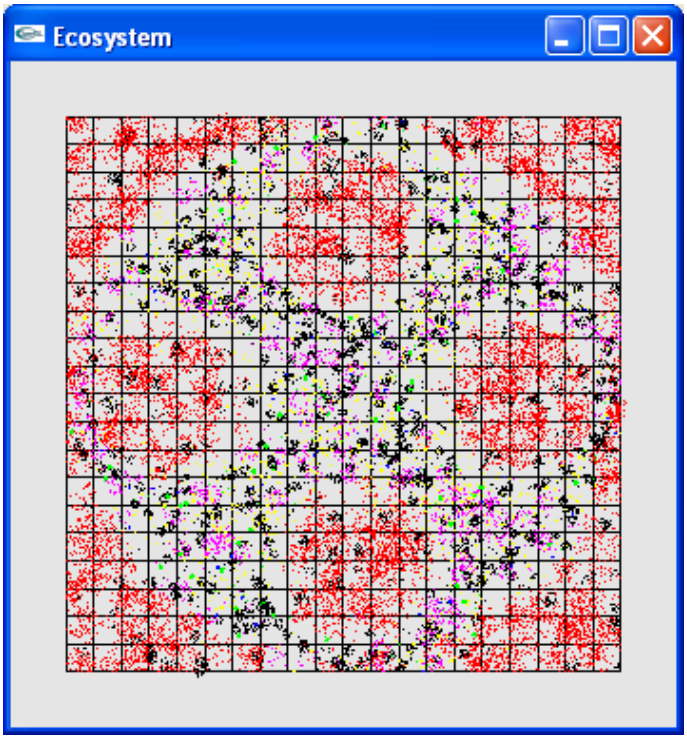


Figura 3.1: Formación de racimos

El número de especies que se incorporan en la simulación es arbitrario. En la sección 3.3.3 en la página 49 se describe cómo incorporar nuevas especies con distintos parámetros de crecimiento.

3.1.2. VIABILIDAD Y COMPETENCIA

Para simular la interacción entre las plantas y el ambiente existen distintos niveles de precisión, el máximo detalle sería tomar en cuenta todos los componentes de la planta tallo, ramificaciones, hojas, etc., lo cual resulta muy complejo computacionalmente. Dado lo anterior se simplifica el modelo y cada planta se representa con un círculo que simboliza la *vecindad ecológica* que ocupa. El radio del círculo está directamente relacionado con el área normalizada (ver ecuación 3.2 en la página siguiente), esto es, conforme la planta va creciendo, su vecindad ecológica aumenta.

La competencia por el espacio entre las plantas se da entre la misma especie y entre plantas de distintas especies. En la figura 3.2 en la página siguiente se observa una colisión, la detección se hace usando la fórmula 3.1.

$$\text{colisión} = \begin{cases} 1 & \text{si} \sqrt{r1^2 + r2^2} < r1 + r2 \\ 0 & \text{en caso contrario} \end{cases} \tag{3.1}$$

Cuando se detecta una colisión es imposible que ambas plantas involucradas continúen con vida, así que una tiene que morir de acuerdo al cálculo de viabilidad. Este parámetro indica la la

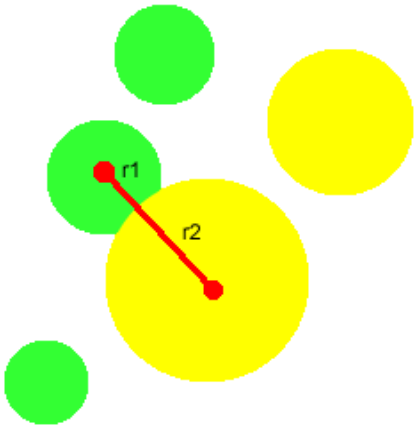


Figura 3.2: Colisión del área ecológica entre dos plantas

probabilidad de la planta de sobrevivir en caso de dos compitiendo por el espacio. Cuando se trata de plantas de la misma especie, la viabilidad es simple de calcular, ya que que sólo depende de su edad normalizada calculada como aparece en la ecuación 3.2. Ésta es una manera estándar de comparar en qué porcentaje de su vida se encuentra la planta con independencia de su especie, así al enfrentarse no importa tanto la edad absoluta, sino en relación a la edad total que la planta vive, en qué etapa se encuentra y así caracterizarla como joven o vieja.

$$\overline{edad} = \frac{edad}{nacimiento - muerte} \tag{3.2}$$

$$v(t) = \begin{cases} \overline{edad} & \text{si } \overline{edad} < 1/2 \\ 1 - \overline{edad} & \text{en caso contrario} \end{cases} \tag{3.3}$$

Originalmente [20] la función de viabilidad favorece a las plantas que están a la mitad de su vida, así las plantas jóvenes o viejas tienen menor probabilidad de ganarle a una que pasa por la mitad de su vida (ecuación 3.3 y figura 3.3).

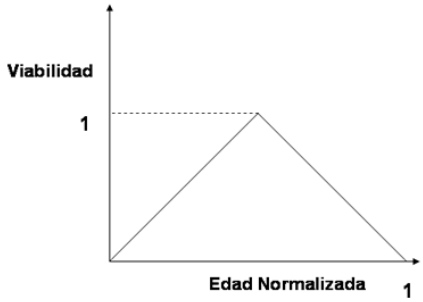
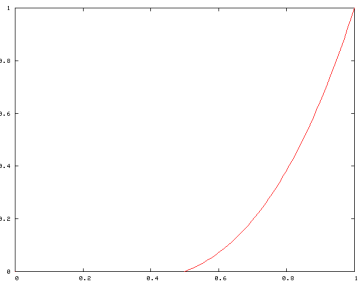


Figura 3.3: Función de viabilidad donde la planta es más fuerte a la mitad de su vida

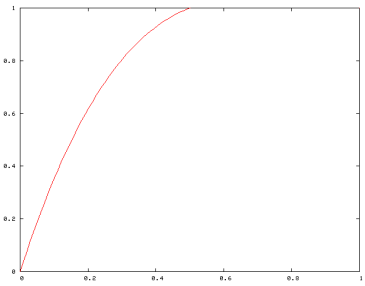
En [23] se hace una extensión en que la función de viabilidad es variable. Este esquema se puede incorporar fácilmente en nuestras plantas. Por lo tanto la función se hace dependiente de la especie además de la edad. Así se pueden modelar algunas especies que tengan mayor probabilidad

de sobrevivir como plagas, plantas más delicadas con una viabilidad menor, comparar tolerancia a los parámetros climáticos, etc. En la tabla 3.4 en la página siguiente se grafican algunas funciones usadas con sus correspondientes ecuaciones.

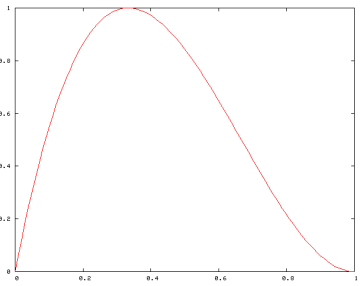
Para incorporarlas al modelo se hace un muestreo de la función correspondiente en valores de la edad normalizada. La función modela de esta manera cierto comportamiento de la planta y también es posible pensar en una evolución o aprendizaje a partir de la experiencia durante la simulación a través del tiempo. Esto servirá posteriormente para modelar interacciones más complejas como por ejemplo reacciones químicas entre las especies o compatibilidad entre unas y otras plantas, plagas, parásitos, enfermedades, en el caso de los agentes, respuesta hacia ciertas especies, veneno, gusto por ciertas plantas, intenciones, etc. La manera como los agentes incorporan este conocimiento se describe en la sección 3.3.2 en la página 48.



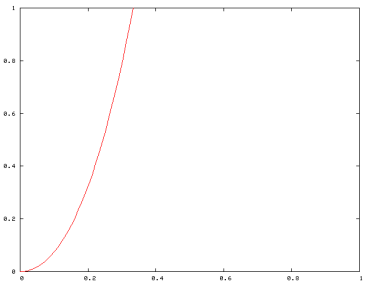
(a) $f_a(x) = 2x^3 - x^2$



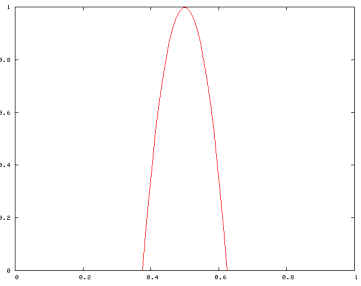
(b) $f_b(x) = 2x^3 - 5x^2 + 4x$



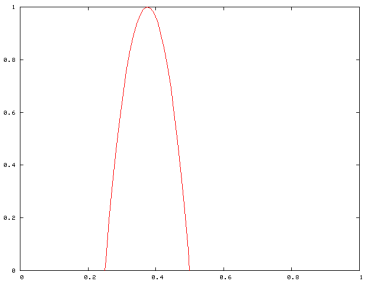
(c) $f_c(x) = 6.75x^3 - 13.5x^2 + 6.75x$



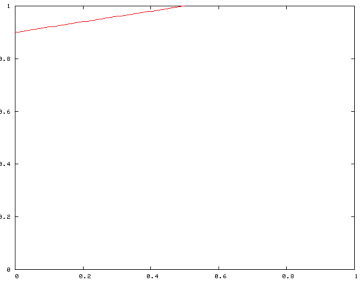
(d) $f_d(x) = -6.75x^3 + 6.75x^2$



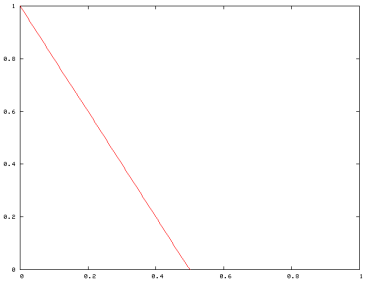
(e) $f_e(x) = -64x^2 + 64x - 15$



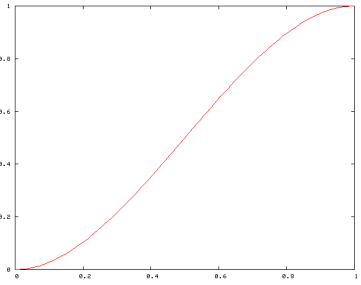
(f) $f_f(x) = -64x^2 + 48x - 8$



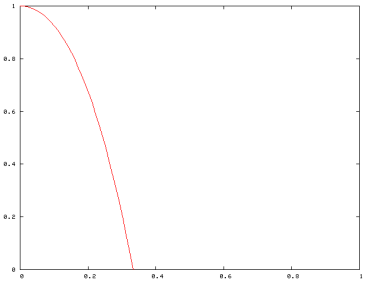
(g) $f_g(x) = -0.2x + 0.9$



(h) $f_h(x) = -2x + 1$



(i) $f_i(x) = -2x^3 + 3x^2$



(j) $f_j(x) = 6.75x^3 - 6.75x^2 + 1$

Figure 3.4: Tabla de funciones para la viabilidad de las plantas

En el caso de ser dos plantas de distinta especie las que colisionan, se toma en cuenta además la funcion de viabilidad el área de la vecindad ecológica de todo el grupo de la misma especie. Esto se hace porque es necesario incorporar retroalimentación negativa al sistema dado que la simulación puede hacerse inestable si una especie comienza a dominar el área dado el fenómeno de formación de racimos y abuso del espacio descrito en la sección 3.1.1 en la página 26. En los sistemas biológicos lo que sucede es un fenómeno llamado sucesión [44], que alterna el dominio de las especies y evita que sea una la que domine durante todo el tiempo. Cuando una especie ocupa mayor vecindad ecológica los recursos comienzan a escasear debido a su uso excesivo, por lo tanto la viabilidad de la especie disminuye, beneficiando con ello a la proliferación de plantas de las demás especies y una alternancia en quién ocupa el área del terreno. De acuerdo a experimentos se ha obtenido que cuando el área total de la vecindad ecológica de una especie aumenta, la viabilidad de esta especie debe ser reducida en realción al área total de la vecindad ecológica de las demás especies, como podemos observar en la ecuación 3.4.

$$v_k(t) = \frac{\sum_{i=1, i\neq k}^{n} a_i}{\sum_{i=1}^{n} a_i} v(t) = \frac{a_1 + a_2 + \ldots + a_{k-1} + a_{k+1} + \ldots + a_n}{a_1 + \ldots + a_n} v(t) \tag{3.4}$$

De este modo el ambiente afecta a la viabilidad de las plantas contra otras especies y se equilibran las probabilidades de colisiones en general. Cuando se especifican los valores iniciales (ver sección 3.3.3 en la página 49) es necesario tomar en cuenta esta retroalimentación del ambiente ya que no solamente cuenta qué tantas semillas libera la planta, sino también su radio para el cálculo del área que ocupa la especie. Esto puede repercutir en que la viabilidad disminuye demasiado y el afectar un parámetro resulte contraproducente para la planta que se desea modelar.

3.1.3. POBLACIÓN

En [20] se muestra que el modelo que usamos cumple con las propiedades de estabilidad y formación de racimos. En esta sección mostramos los parámetros que definen el desarrollo de la población de plantas. De acuerdo a estos es posible relacionar la información biológica disponible y especificar los valores para las especies que se desean modelar obtener así resultados realistas. Esto aunado a las funciones de viabilidad variables, permite una versatilidad en la construcción del modelo. Las variables se enlistan en el cuadro 3.1 en la página siguiente y sus respectivos parámetros de referencia se especifican en el cuadro 3.2 en la página siguiente.

Esto corresponde a la simulación estable mostrada en la figura 3.5 en la página 33. Todas las simulaciones que se ilustran parten de una población de una sola planta de cierta especie y se monitorea cómo evoluciona su crecimiento. Las gráficas tienen como unidad el promedio de población en 200 días para considerar el fenómeno de sucesión explicado en la sección 3.1.2 en la página 27. Así se presentan gráficas continuas pero que muestran cómo los valores iniciales modifican la forma de la gráfica, aunque con el tiempo el modelo tiende a ser estable.

A continuación se altera el valor inicial de uno de los parámetros para dar una interpretación de acuerdo a la gráfica de referencia mostrada. Esto corresponde a las distintas posibilidades que se tienen para crear modelos de plantas reales y su comportamiento a partir de tablas biológicas. A partir de las gráficas siguientes y su interpretación se pueden hacer analogías con la realidad y

Variable	Significado
Radio	Es el área de cobertura biológica que ocupa la planta.
Multiplicador	Es un factor que se multiplica por el número aleatorio de semillas que libera una planta para ajustar su difusión.
Edad	Cuánto dura una planta (en días).
Semillas	La cantidad máxima de semillas que pueden ser liberadas.

Cuadro 3.1: Variables esenciales para modelar plantas

Parámetro	**Valor de referencia**
Radio	0.05
Multiplicador	10
Edad	365
Semillas	4

Cuadro 3.2: Valores de referencia para los parámetros de la simulación

crear plantas con un comportamiento definido. Aunado a la viabilidad que se expone en la sección 3.1.2 en la página 27 y a tomar en cuenta la retroalimentación negativa como se expone en la sección 3.1.1 en la página 26 se tiene una amplia libertad para definir el comportamiento requerido.

Radio

En la gráfica 3.6 en la página siguiente se observa que conforme aumenta el radio, la cantidad de plantas promedio disminuye. Esto se explica dado que la simulación está acotada en espacio, entonces si cada planta ocupa más espacio, deben caber menos. En la realidad este fenómeno también ocurre pero con varios recursos, la tierra, los nutrientes, el agua, el sol, etc. aunque en nuestra simulación solamente se considere la vecindad ecológica. En las simulaciones se demuestra que la simplificación resulta útil. Es importante considerar que cuando una planta ocupa un espacio mayor lo más lógico es que también su edad sea mayor y su viabilidad sea alta durante mayor tiempo. Si no se toma en cuenta lo anterior se puede tener un modelo que considere árboles, pero que estén muriendo a causa de la proliferación de pasto.

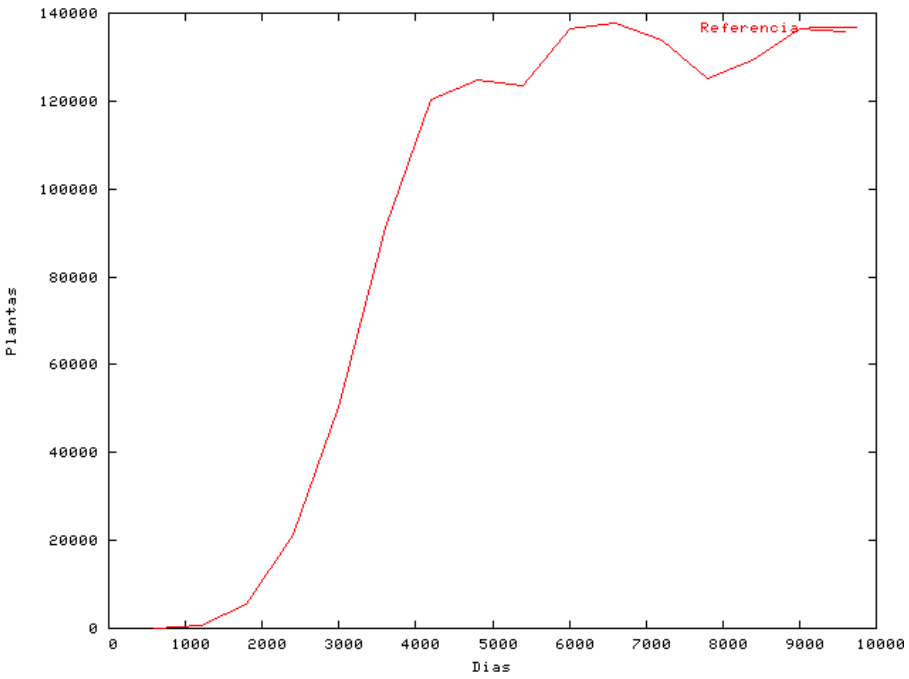


Figura 3.5: Parámetros de referencia (promedio en 200 días)

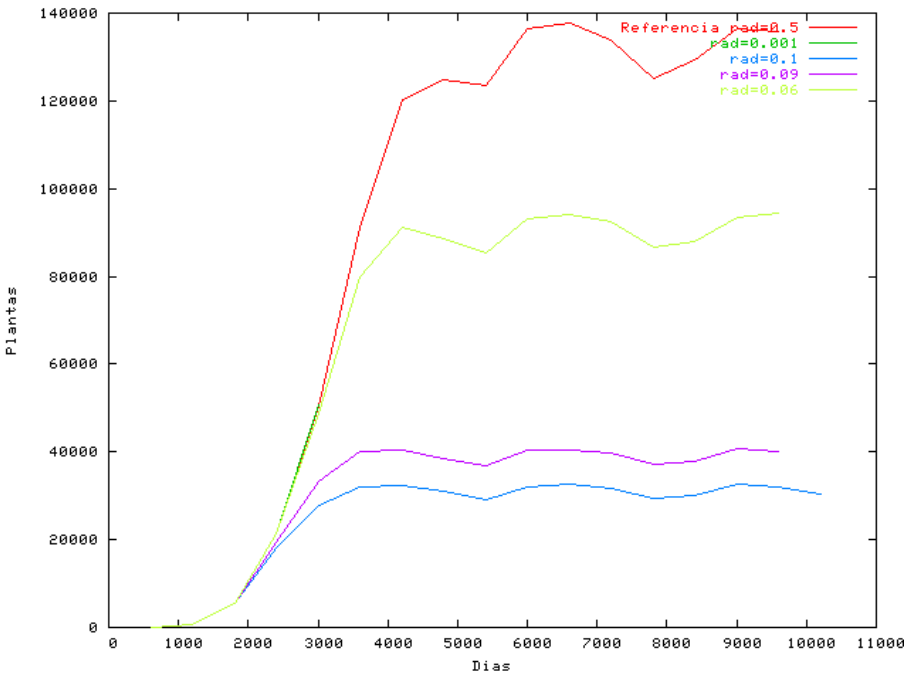


Figura 3.6: Variación del radio (promedio en 200 días)

Multiplicador

Este parámetro (gráfica 3.7 en la página siguiente) modifica la cantidad de semillas aleatorias, por lo tanto al ser mayor causa que la simulación se vaya a su punto de equilibrio más rápido

y de modo inverso cuando es menor, que la simulación se retarde. En la naturaleza no existe un parámetro similar, solamente se incorpora para estabilizar la simulación más rápido. Esto es debido a que el número de semillas puede no ser el suficiente y la simulación tarde más en obtener resultados equivalentes a si se incrementa al número total de semillas que inicialmente se lanza por planta.

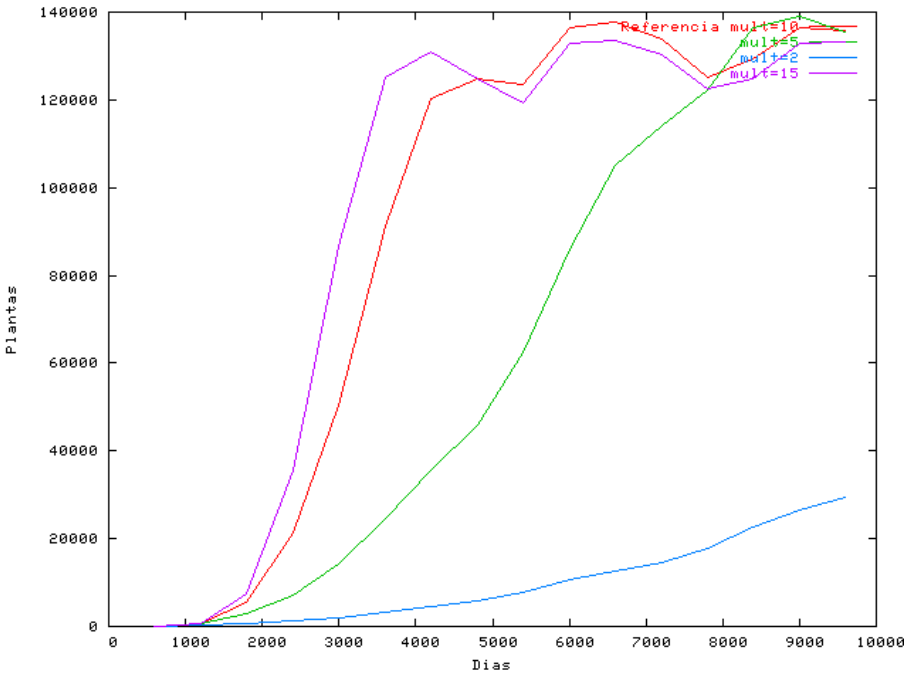


Figura 3.7: Variación del multiplicador (promedio en 200 días)

Edad

La edad (gráfica 3.8 en la página siguiente) comprime o alarga los ciclos en los que la población se recicla, al mismo tiempo que retrasa alcanzar el punto de equilibrio. En el caso de una simulación con varias especies, las edades se asignan aleatoriamente, por lo que al parecer la población no tendría por qué seguir el fenómeno de sucesión, sin embargo la formación de racimos hace que poco a poco las especies vayan ocupando ciertas zonas y se van estabilizando los periodos de nacimiento y muerte.

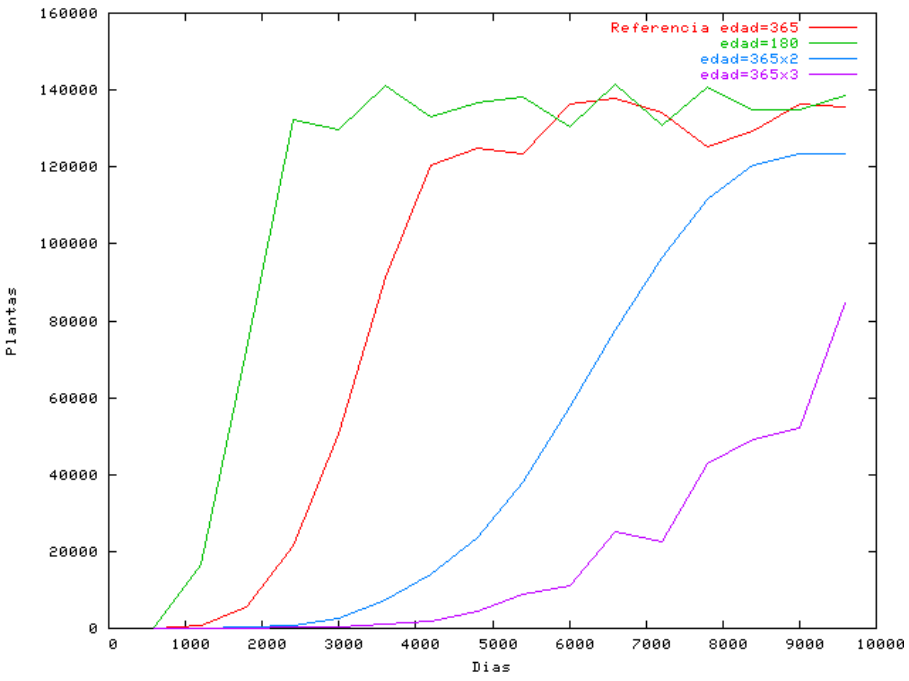


Figura 3.8: Variación de la edad (promedio en 200 días)

Semillas

La cantidad de semillas (figura 3.9 en la página siguiente) controla la cantidad promedio de plantas que existirán de plantas en la simulación. Este parámetro está relacionado con el radio, pero al ser cíclica la población, lo que provoca es que crezcan los picos y valles a los que llega la cantidad de plantas. En las gráficas se muestran solamente los promedios. También podemos observar que la formación de racimos es más mientras mayor sea la cantidad de semillas lanzadas por planta.

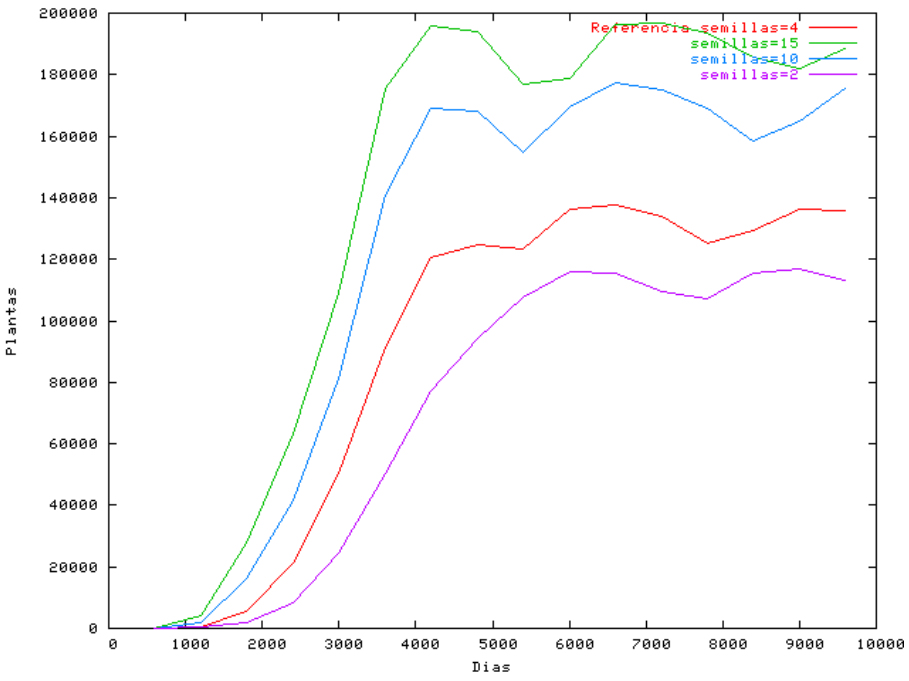


Figura 3.9: Variación de la cantidad máxima de semillas (promedio en 200 días)

3.1.4. MODELOS E IMÁGENES FINALES

Se usan dos representaciones para el ecosistema, la primera de baja calidad se usa para tener retroalimentación inmediata sobre la competencia entre las plantas. La segunda es una visualización fotorrealista.

Para la primera se usa un plano 2D donde cada planta está representada como un círculo con un color que indica la especie de la planta y el radio corresponde a su vecindad ecológica (varía con la edad). Los círculos se colocan en el lugar que corresponde a la planta (ver figura 3.10 en la página siguiente). En esta representación es muy fácil notar las colisiones y lo que ocurre. Posteriormente esta representación se extiende incluyendo a los agentes en la sección 3.2 en la página 38.

En el segundo caso el programa genera muestreos en ciertos valores del tiempo que se pueden determinar previamente en el script en Extensible Markup Language (XML) (ver sección 3.3.3 en la página 49) y el resultado se gurada archivos de descripcion de la escena que se pueden procesar con POVray [7].

Los archivos crecen de acuerdo al número de plantas que contiene el ecosistema, por ejemplo, una escena con unas cien mil plantas puede medir unos cincuenta megas de espacio en disco y para renderear ocupar más de 1GB. Para disminuir este consumo se usa instanciamiento [20, 22], lo que consiste en agrupar plantas de edad similar, por ejemplo las plantas de cierta especie que están en el primer 20 % de su vida se representan con una sola imagen que a su vez puede ser escalada para alcanzar mayor realismo. El porcentaje de vida se mide usando la edad normalizada (explicada en la sección 3.1.2 en la página 27). Las plantas que son representadas con este modelo son *instanciadas* y se les aplican operaciones geométricas sencillas, traslación asignada por el simulador y rotación arbitraria asignada como un valor inicial. La diferencia visual resulta mínima

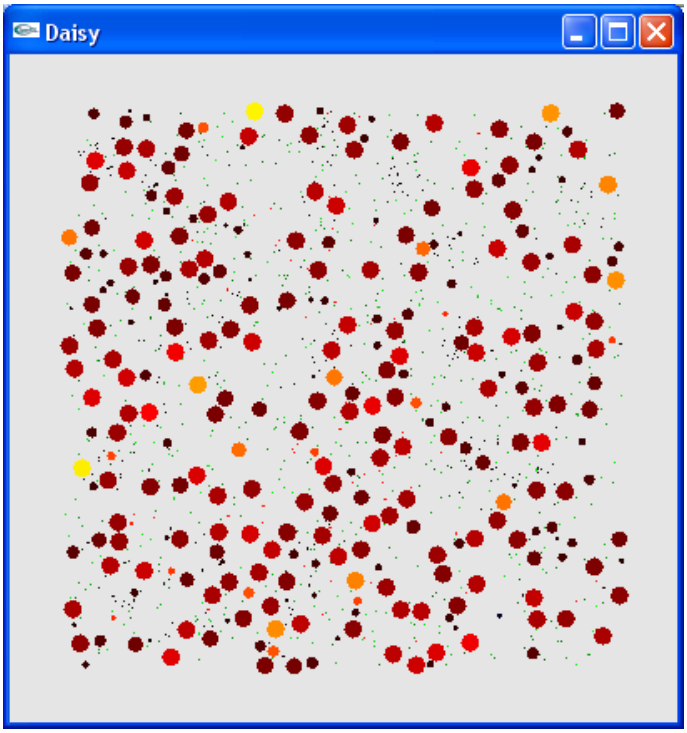


Figura 3.10: Representación de las plantas como círculos en un plano

para el ojo humano. En nuestra implementación usamos cinco instancias por especie, si fuera necesario fácilmente se pueden agregar más, pero hemos notado que la calidad visual no varía lo suficiente y el gasto en memoría es bastante.

Usando instanciamiento se llega a un factor de compresión mayor o igual al 50 % [20].

Algunos de los modelos que hemos usado en las ilustraciones fueron creados con el software PlantStudio [9] que utiliza sistemas-L para modelar las plantas. En la figura 3.11 podemos observar algunos de ellos.

Figura 3.11: Modelos usados en la recreación de ecosistemas

En el caso de los agentes los modelos fueron creados usando el software Poser4 de Metacrations o mediante algunas imágenes obtenidas en internet. Dado que realmente eran con fines ilustrativos

se usó la técnica de *billboarding* en POVray para poder mostrar las imágenes de un modo más o menos realista. En la figura 3.12 en la página siguiente se pueden ver algunos granjeros trabajando dentro del ecosistema.

Al crear escenas en 3D se puede ganar en desempeño al insertar objetos en 2D de tal modo que aparezcan como si fueran tridimensionales. Esta es la idea básica del billboarding.

Un billboard es parecido a un señalamiento en una carretera. Se define un sólido rectangular y se le aplica una textura. La meta es hacer que los objetos 2D aparezcan como 3D. El efecto se obtiene rotando la primitiva a la cual se aplica la textura de tal modo que siempre muestra la cara a la cámara. No importa si la imagen que se quiere convertir en billboard no es rectangular. Algunas porciones del billboard pueden hacerse transparentes y así una figura irregular también puede usarse. En el caso de POVray esto se hizo indicando que el color de fondo de las imágenes tiene una transmisión del 100 % de la luz, es decir, es transparente.

Muchos juegos emplean esta técnica en sprites animados (imágenes 2D). Por ejemplo, cuando el usuario se mueve en un laberinto 3D, puede ver armas o recompensas que puede recoger, normalmente estas son imágenes bidimensionales que se colocan como texturas en una primitiva rectangular, también se utiliza en los juegos para dibujar árboles, arbustos, nubes, etc. objetos cuya geometría no es totalmente indispensable.

Cuando una imagen se aplica al billboard, la primitva rectangular debe ser primero rotada de tal modo que la normal sea perpendicular al vector de visión desde la cámara, después de eso se traslada a su posición y posteriormente se puede aplicar la textura. El billboarding trabaja mejor con objetos simétricos, especialmente con aquellos que lo son al eje vertical. También se requiere que el punto de vista del usuario no varíe mucho en Z dado que desde arriba se hace muy obvio que el objeto es 2D en vez de 3D.

Dado que el punto de vista de un granjero es relativamente el mismo en Z esta técnica resulta muy adecuada para crear recorridos en la escena desde el punto de vista de un hipotético granjero.

3.2. AGENTES

Los agentes son autómatas guiados proceduralmente. Cada uno puede realizar acciones en el ecosistema y puede comunicarse con los demás, poseen memoria para posponer trabajo que no puede realizar de inmediato, a continuación se hace una descripción detallada de cómo se caracterizan a nuestros agentes y en la siguiente sección se detalla cómo se especifica el comportamiento que realizan.

3.2.1. PERCEPCIÓN

El agente está representado esquemáticamente por un círculo que a su vez simboliza su área de visión o influencia. Al comenzar su ciclo de acción, cada agente se coloca en una orilla del ecosistema con dirección perpendicular hacia dentro. Posteriormente el agente camina buscando un objeto parte de su comportamiento. Para poder encontrar lo que el agente busca, es necesario realizar una búsqueda, que se describe posteriormente en la sección 3.2.3 en la página 41. La

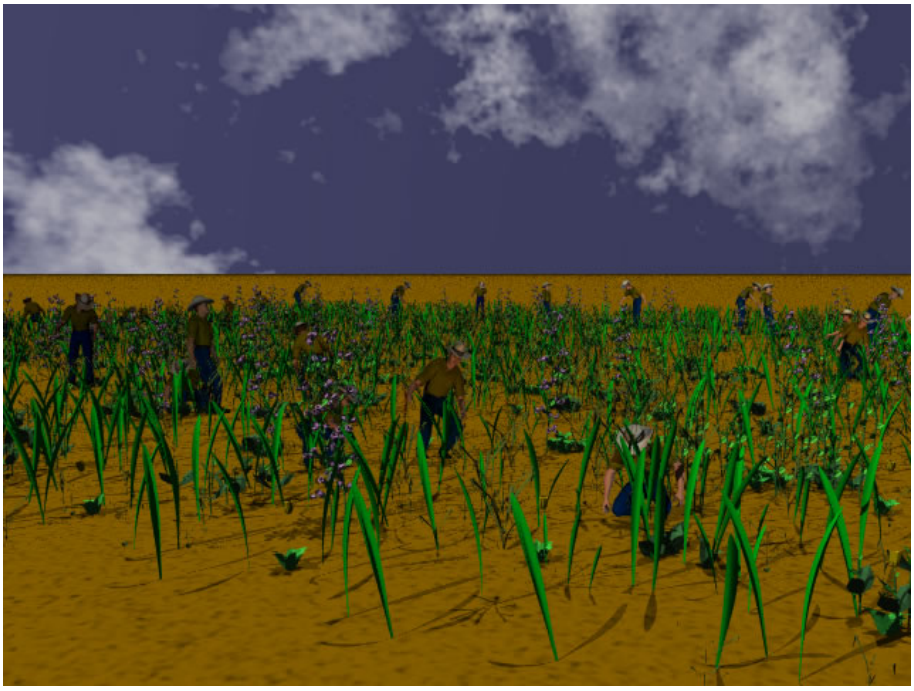


Figura 3.12: Granjeros dentro del ecosistema, imagen que usa billboarding

percepción de las plantas en las que el agente está interesado se limita mediante un ángulo de visión coo se observa en la figura 3.13.

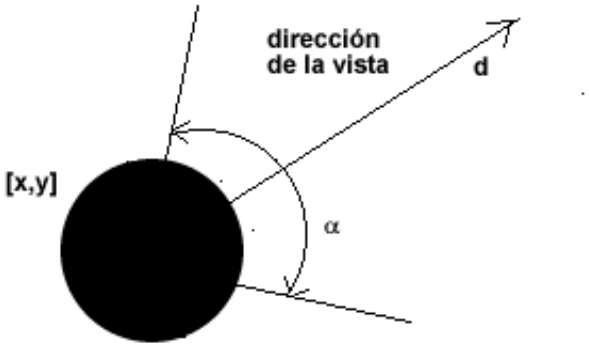


Figura 3.13: Distancia y ángulo de visión de un agente

El agente camina sin obstáculos y tiene una percepción global de la simulación, sin embargo solo actúa sobre el punto de interés (planta) más cercano. En su camino hacia una planta puede encontrar otras que sean de su interés y se almacenan en memoria para posteriormente regresar.

Los agentes se pueden parametrizar según el radio de acción que tienen, es decir, cada agente solamente va a encontrar los objetos que estén dentro de su rango de visión. Mientras ninguno aparezca dentro del rango el agente sigue caminando según la última dirección hacia la que caminó, y en su camino puede o no encontrar lo que busca. En el caso de que el agente siga caminando y salga de los límites del terreno, o del área de acción según sea el caso, pueden pasar dos cosas, se

cambia la dirección hacia el sentido contrario o bien se vuelve a colocar el agente en una posición aleatoria dentro del espacio en donde debe trabajar. Por los experimentos realizados y según diversas topologías de las áreas de acción con las que se ha trabajado, resulta más eficiente volver a colocarlo en una posición arbitraria y con una dirección también aleatoria.

Los agentes pueden ser limitados en cuanto a las especies que ocupan, y también poseen una función de viabilidad que los representa, estas funciones se explican para las plantas en la sección 3.1.2 en la página 27 sin embargo, funcionan exactamente igual en los agentes. Esto modifica la manera como los agentes perciben el espacio ya que pueden ignorar por completo a alguna especie, caminar hacia una planta y decidir de modo semi-aleatorio si la van a afectar a la planta o no, o bien la acción por omisión que es afectar con una probabilidad de 1 a la especie de planta que está definida en su comportamiento.

Los agentes pueden o no percibirse unos a otros. Sin embargo sí se comunican las tareas que tienen pendientes, ya sea la actual o las que están en memoria. Así se evita realizar tareas innecesarias, como por ejemplo en el caso de que un agente se encuentre caminando hacia una planta que será cortada antes que él lo haga, por otro agente aún más cercano.

En el esquema actualmente desarrollado es relativamente fácil incorporar percepción del medio basada en mediciones, por ejemplo, dado que el agente tiene un rango de visión, es posible que el agente haga subdivisiones discretas de lo que observa mientras camina sin tarea definida, de tal modo que siga las zonas con mayor o menor concentración de plantas y que posteriormente tenga mayor probabilidad de cortar, sembrar o cual sea lo que esté definido en su estado interno y comportamiento.

3.2.2. MOVIMIENTO

Durante el ciclo de visita de los agentes al campo o ecosistema, se detiene el tiempo de simulación. Los agentes están limitados en cuanto a la distancia máxima que pueden recorrer, en la sección 3.3.3 en la página 49 se detalla cómo se parametriza esto para cada tipo de agente que se puede colocar en la simulación. Si pensamos que el agente se mueve con una velocidad uniforme, entonces este parámetro también puede representar el tiempo que el agente pasa trabajando en el campo. Moviendo este parámetro se puede definir qué tanto afecta el agente al campo y dado que su camino no es determinístico, podemos definir qué tan bien se realiza su tarea. En la ilustración 3.14 en la página siguiente podemos observar que estos valores determinan qué tan definida es la frontera de acción de los agentes.

La aplicación puede mostrar el camino que los agentes realizan. El movimiento de los agentes es guiado por el algoritmo 2 en la página 42. Cuando los agentes usan memoria se puede observar un movimiento en el que el aente recorre una zona y regresa hacia ciertos puntos de la trayectoria que ha recorrido, oscilando dentro de una región. Cuando el agente no usa la memoria su movimiento se parece más a una caminata aleatoria, dado que cuando no tiene una tarea definida simplmente sigue recorriendo el plano en la misma dirección hasta que encuentra algo qué hacer o bien sale del plano y es necesario cambiar su dirección hacia dentro o bien colocarse en una posición nueva para poder continuar. La segunda opción es más eficiente dado que en el caso de

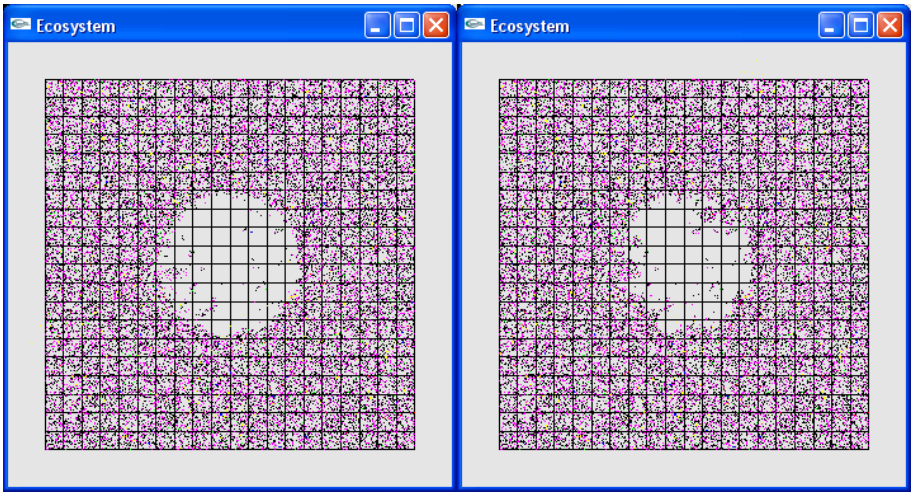


Figura 3.14: Distintos valores para la distancia máxima que recorre el agente hacen que su tarea sea más irregular

simplemente cambiar la dirección y cuando la zona tiene poca probabilidad de encontrar una planta de interés el aente puede ciclarse sin encontrar otra nueva planta y su trabajo se desperdicia.

3.2.3. BÚSQUEDA

Uno de los principales problemas en nuestra simulación es que se requiere detectar las colisiones entre las planta, pero dado que cada planta (o agente) tiene sólo la información de su propio estado, cada ciclo es necesario que se actualice la información sobre la posición de sus vecinos. Esto origina una búsqueda $O(n^2)$ ya que cada planta debe verificar la posición de todas las demás. Para atacar el problema se usaron dos métodos.

Árboles K-dimensionales

Estos árboles son la generalización de los árboles binarios, pero para n dimensiones. En un árbol binario convencional la inserción se realiza como se observa en el algoritmo 3 en la página siguiente, con un costo $O(log(n))$ y la búsqueda de un elemento es $O(log(n))$ lo que reduce considerablemente el tiempo de la operación más común. El caso que se requiere es una búsqueda de dos dimensiones, es decir $K = 2$ y una presentación generalizada de los algoritmos de inserción, borrado, optimización, se puede consultar en [24]. El resultado de almacenar las plantas en un árbol K-dimensional es una partición del espacio como el que se muestra en la figura 3.15 en la página 43.

Estos árboles poseen además varios operadores de búsqueda como son exacta, parcial, por región y de vecinos. Al ir creando cuadros, resulta que cada partición es un nodo del árbol. El principal problema al que nos enfrentamos al usar esta técnica está también documentada en [24]: la eliminación. Dado que cada ciclo existen plantas que mueren, se puede optar por eliminarlas del árbol o bien reconstruirlo por completo. Debido a que la operación de eliminar conlleva un

Algoritmo 2 Movimiento de los agentes

mientras Se alcanza la distancia total especificada **hacer**
  para todo a onde a es un agente **hacer**
    si a fuera de la zona de acción **entonces**
      Cambiar dirección de a hacia dentro de la zona o escoger nueva posición
    fin si
    si Planta en memoria **entonces**
      Dirigirse hacia planta en memoria
    en otro caso
      a_i busca a la planta más cercana y dirigirse hacia ella.
    fin si
    si Planta dentro del rango de visión **entonces**
      Ir a la planta y ejecutar acción {Aquí el agente debe *decidir*}
      Actualizar distancia, dirección
      Continuar con siguiente a_i
    fin si
    Seguir caminando en la misma dirección y actualizar distancia
    si Otra planta dentro del rango de acción **entonces**
      si a tiene memoria disponible **entonces**
        Almacenar en memoria
      en otro caso
        Comunicar tareas con agentes cercanos
      fin si
    fin si
  fin para
fin mientras

Algoritmo 3 Inserción en un árbol binario

procedimiento insertar (elemento e en árbol binario t)
si $t =$ nulo **entonces**
  $t = nuevoNodo(e)$
si no, pero $t.contenido < e$ **entonces**
  $insertar(e, t.ramaderecha)$
en otro caso
  $insertar(e, t.ramaizquierda)$
fin si

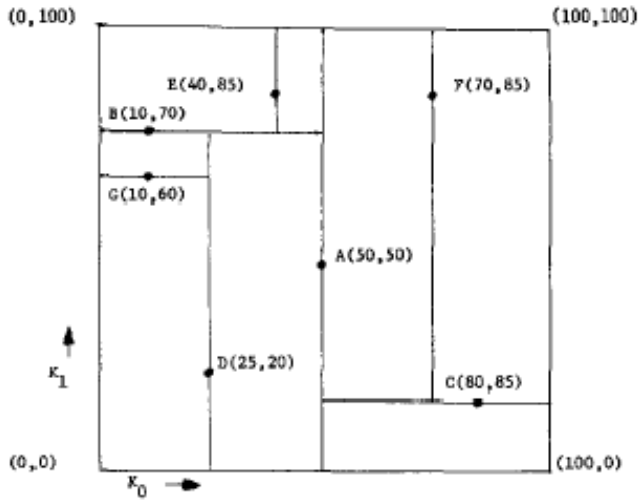


Figura 3.15: Partición del espacio según un árbol K-dimensional

movimiento en la estructura jerárquica del árbol, se optó por la reconstrucción total, sin embargo a pesar de que la inserción es $O(log(n))$ no se obtuvieron los resultados esperados a comparación con una operación de $O(k)$ que resulta cuando se almacenan las plantas en un simple arreglo.

Subdivisión regular del espacio

Este algoritmo está documentado en [67] como el método de reja para búsqueda en rangos, dentro de los algoritmos geométricos. El algoritmo es una búsqueda sencilla de $O(n)$ para cada planta, sin embargo se añade información relevante sobre la posición de la planta o el agente. Así al hacer una búsqueda sólo se hace en las casillas adyacentes, es decir, en las que posiblemente se produzca una colisión. Para esto es necesario que cada ciclo se cree una estructura que contiene las casillas que abarca la planta o agente, pero el crear esta estructura es de $O(k)$, y una vez construida las plantas hacen una búsqueda de $O(n)$, lo que daría de nuevo una búsqueda $O(n^2)$ sin embargo al reducir considerablemente el espacio total de búsqueda el tiempo de procesamiento es menor. Una esquematización de este método se encuentra en la figura 3.16 en la página siguiente.

3.2.4. MEMORIA

Cada agente tiene una memoria de las plantas que encuentra en su recorrido hacia la planta más cercana, esta memoria es recorrida hacia atrás una vez que ha realizado la tarea más prioritaria, es decir, funciona como un pila.

Si el estado del agente cambia durante un ciclo de trabajo, por ejemplo se cansa o termina la distancia o tiempo que debe trabajar, es necesario que estas tareas sean reubicadas en otro de los agentes. Para poder realizar esto se usa a las plantas como un *blackboard* general en donde ellas mismas tienen una bandera que indica si algún agente ha mostrado interés en cortarlas, de este modo se evitan ciclos innecesarios y que los agentes estén buscando a las mismas plantas de modo

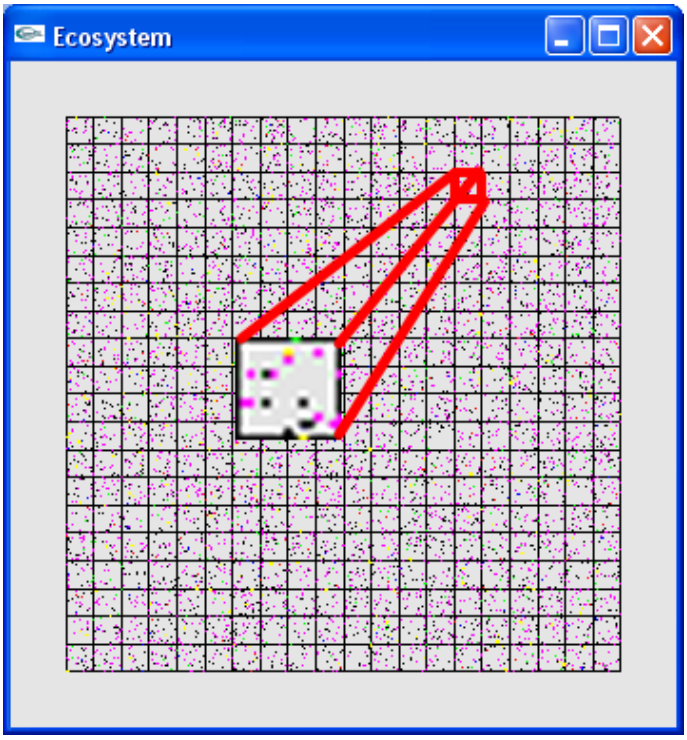


Figura 3.16: Búsqueda usando casillas resultado de subdividir el espacio

ineficiente.

También puede ocurrir que cuando el agente decida recorrer las tareas que tiene en memoria, las plantas ya no sean de su interés ya sea por la distancia o por las intenciones que tiene, entonces debe comunicar esto al agente más cercano, o bien liberar algunas de las plantas ya que ocurre una sobrecarga de la pila de tareas y saca las que están más lejanas, es decir las que están al fondo, las marca como libres –para que otro agente pueda interesarse en ellas– y continúa su camino hacia la planta de interés en ese momento.

3.3. COMPORTAMIENTO

El comportamiento de los agentes se origina a través de los estímulos que llegan mediante una serie de sensores, que generan señales de estímulo y se transportan al generador de intenciones. Aquí se combinan con una serie de estados internos que corresponden a emociones como son flojera, cansancio, hambre, etc. Al final los hábitos son la última parte que recibe el generador de intenciones y se representan como una serie de funciones que describen la probabilidad de que una acción sea tomada. Una vez que se decide lo que se hará en el ciclo actual, el generador de respuesta vuelve a alimentar los estados internos, por ejemplo, el de una señal de caminar puede incrementar el cansancio al mismo tiempo que los efectores realizan esta acción.

El agente percibe el ambiente mediante una serie de sensores, actualmente la implementación está basada en un sensor de visibilidad, sin embargo se pueden agregar otros que respondan a olores, humedad, etc. sensaciones que se especifiquen en el campo.

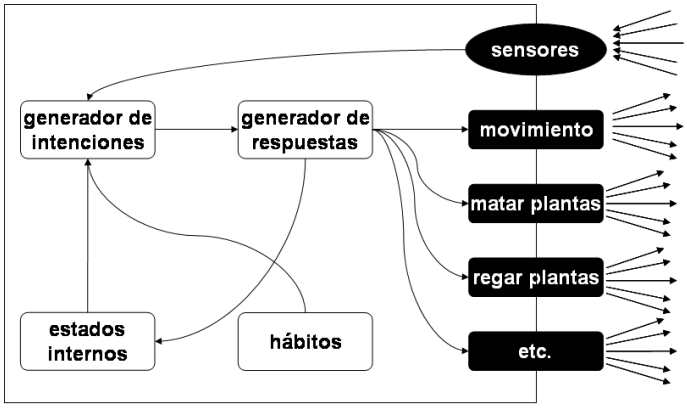


Figura 3.17: Modelo de estímulo–respuesta de los agentes

El área que cada agente percibe está acotada por un ángulo de visión como se observa en la figura 3.13 en la página 39, se tiene un ángulo α, una distancia d y además se pueden considerar variaciones en el ambiente que modifiquen la efectividad de este campo, como pueden ser el día y la noche, las condiciones climáticas, si hay niebla, etc. o algún estado interno como podría ser el cansancio, el hambre, etc.

Es necesario especificar el conjunto de características que cada conjunto de agentes va a tener. Del mismo modo que se modelan las plantas, el agente recibe ciertos valores iniciales de visión, función de viabilidad para calcular la probabilidad con la que realizará una acción con determinada especie de plantas, etc.

Se puede definir un rango de acción como podemos observar en las imágenes en la figura 3.18. Se cuenta con operadores para especificar áreas en círculos, rectángulos, operaciones de conjuntos básicas con estos elementos como unión y complemento y además se puede integrar un bitmap para que el agente solamente actúe en el rango que se haya especificado en un programa de dibujo como Photoshop, ver ilustración 3.19 en la página siguiente.

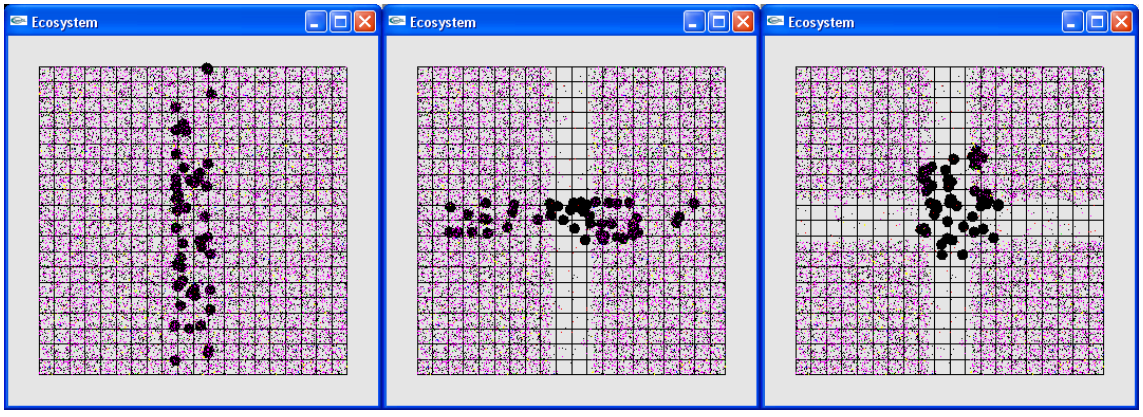


Figura 3.18: Agentes trabajando dentro de un área específica

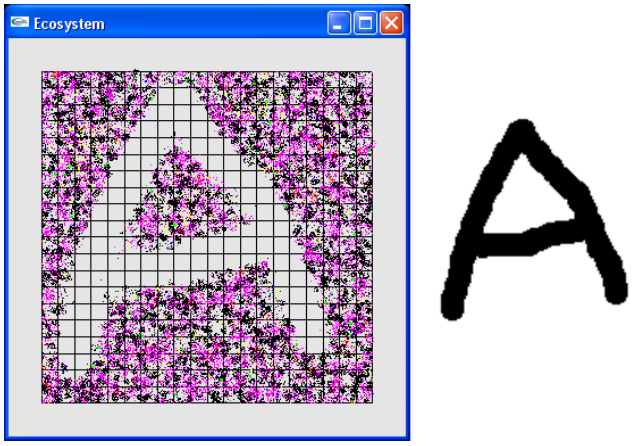


Figura 3.19: Agentes que trabaja en el área especificada mediante un bitmap

El objetivo perseguido al dotar a los agentes de la capacidad de trazar una figura especificada mediante un mapa de bits, es que se puedan conseguir imágenes parecidas a los *crop circles* en Inglaterra [2, 3], fenómeno que no ha sido explicado del todo. Podemos observar algunos de estos dibujos en el cuadro 3.20 en la página siguiente. Lo que es interesante de esto es una geometría regular y la armonía visual que representan estos enigmáticos dibujos. En nuestro trabajo una vez que se ha creado el bitmap, los agentes trabajan de modo independiente.

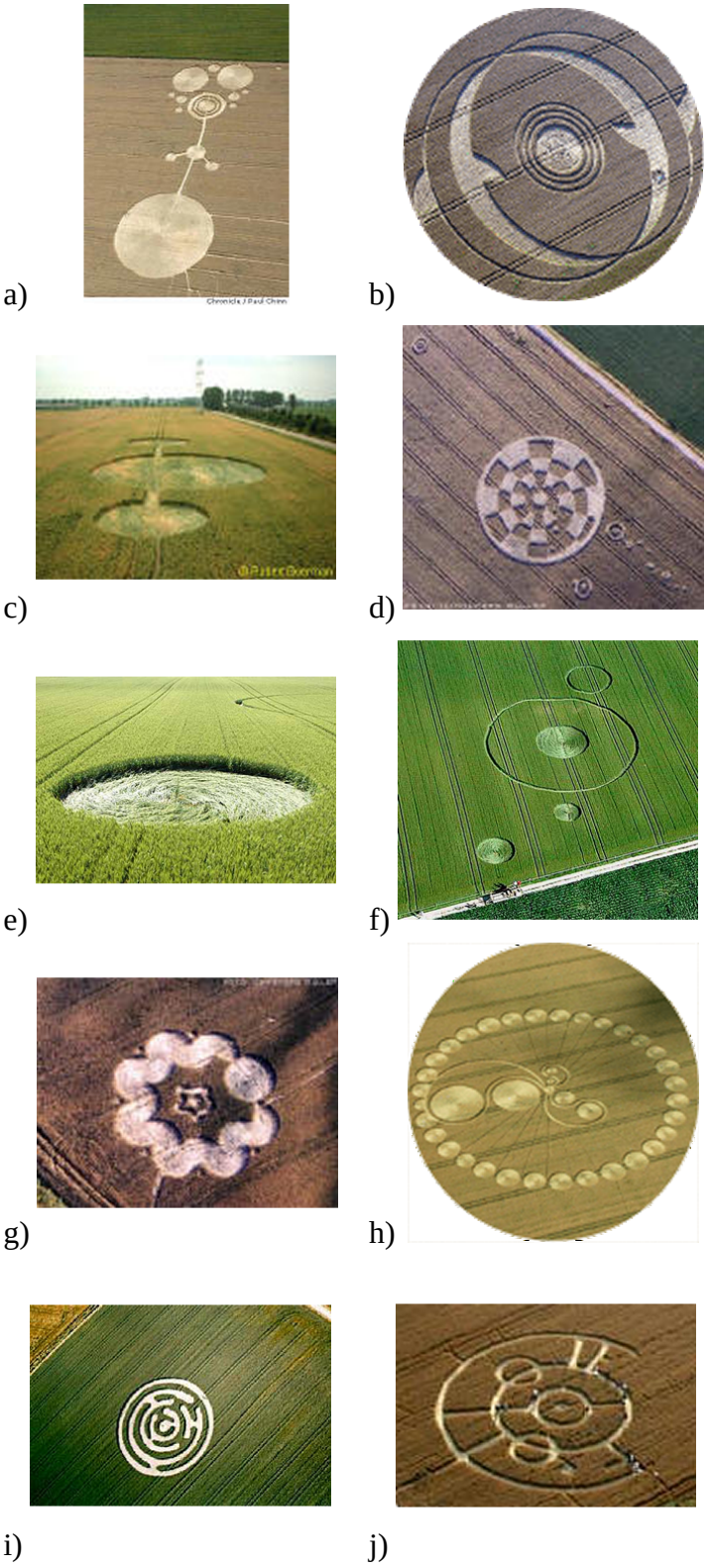


Figura 3.20: Distintos diseños de *crop circles*

3.3.1. ACCIONES

Las operaciones básicas que un agente puede realizar con las plantas están ilustradas en el cuadro 3.3.

cortar	Equivale a la muerte de una planta
sembrar	Se incorpora una nueva planta de determinada especie
inhibir	El crecimiento de la planta se *retarda*
excitar	El crecimiento de la planta se *acelera*

Cuadro 3.3: Operaciones de los agentes

El rango de comportamiento que se puede especificar para los agentes es bastante amplio, se puede generar la implementación para pensar en agentes como granjeros que tienen las conductas de sembrar y cortar plantas, animales que afectan a las ya sembradas, etc. Al incorporar las operaciones de inhibir y excitar el crecimiento se puede también simular plagas, enfermedades, fenómenos naturales, y cualquier otro fenómeno que pueda afectar de un modo no definitivo el desarrollo de las plantas. Como también se puede especificar un área en la que los agentes actúan, incluso las propiedades del terreno pueden ser simuladas con el esquema propuesto.

Un ejemplo de un comportamiento no obvio para los agentes es la lluvia: al disminuir el rango de visión a un radio muy pequeño y hacer que los agentes se reposicionen arbitrariamene en el espacio, es posible imaginar cada gota como un microagente que cae en una posición totalmente aleatoria, y si una planta está dentro de su radio de acción la afecta excitando su crecimiento, luego se reposiciona aleatoriamente y así, incluso seleccionando una distribución estadística o un porcentaje de crecimiento distinto para cada planta de tal modo que la lluvia pueda afectar de distinto modo a los individuos de la población.

3.3.2. INTENCIONES

Cada agente tiene un comportamiento básico definido que consta de una serie de *conductas*, cada una de ella consiste en la especificación de una especie de planta, la acción que se aplicará, el área en que la conducta será válida y una función de probabilidad de que este evento ocurra una vez que se encuentre a la planta en cuestión.

Estos parámetros se definen para cierto conjunto de agentes, del mismo modo que la distancia máxima (o tiempo si la velocidad es constante) que cada agente va a recorrer. También se define aquí el rango de visión del agente. Se pueden definir varias conductas y así existirán agentes que estén interesados en cortar varias plantas del mismo modo que en regar otras. Dado que se pueden especificar distintas áreas de acción en las conductas, se ha definido que cuando existen varias, el área total es igual a la unión de las sub-áreas.

El motor de intenciones del agente funciona como una máquina de estados bastante sencilla, ya que no son muchas las tareas que puede realizar, pero dada la especificación arbirtraria y la función de probabilidad con la que el agente decide si ejecutar una acción o no, es posible definir una conducta bastante real. Estas funciones funcionan de modo análogo a las que se explican en la

sección 3.1.2 en la página 27, son muestreos discretos sobre funciones continuas, solamente que el agente no tiene vida media sino que se puede definir una operación análoga al ciclo en el que el agente trabaja mediante la variable que indica cuánta distancia o tiempo lleva de su jornada en determinado momento. Así es posible definir que un agente comience con un nivel de actividad muy alto y poco a poco se canse, o que sea más eficiente a la mitad del día, etc.

Estas funciones de probabilidad de éxito o fracaso también pueden provocar retroalimentación en las intenciones de los agentes si se define alguna respuesta deseada, de tal modo que el afectar a cierta especie de planta cause una reacción que se integre al conocimiento global mediante las funciones de viabilidad.

3.3.3. ESPECIFICACIÓN DEL COMPORTAMIENTO

Para dar valores iniciales a los distintos parámetros de la simulación se seleccionó el formato XML dado que resulta muy sencillo darle una sintaxis a la relación entre los elementos y sus propiedades. En diversos precedentes [38, 65] la solución ha sido desarrollar un lenguaje completo que pueda expresar la complejidad de la tarea, sin embargo XML presenta como ventajas las siguientes:

- Una sintaxis conocida, así que los autores pueden comprender fácilmente los patrones que se representan.
- Es bastante sencillo comparado con las sintaxis de otros lenguajes.
- Simplifica la tarea de extraer información automáticamente.
- Es muy fácil de agregar nuevos elementos a partir de la base que ya está hecha, sin modificar el código del parser actual.
- Un único documento fuente puede dar origen a varios otros formatos, una vez que se estandariza una versión, se pueden crear ampliaciones para casos específicos.
- Permite separar contenido y forma de presentación, incluso serviría para generar otros documentos además parametrizar la simulación.

En versiones preliminares se analizó la alternativa de usar archivos que únicamente relacionaban una variable con el valor actual para la simulación en curso. La sintaxis que se usó fue la de archivos INI popularizada por diversas aplicaciones [7], sin embargo la expresividad necesaria complicaba el mantenimiento de los archivos.

Por otro lado el desarrollar un lenguaje era un gasto superfluo debido a que una sintaxis conocida basta para determinar los valores iniciales y la estructura jerárquica de los elementos en la simulación, crear una nueva no representa una ventaja importante.

Los elementos incluidos en la especificación en XML actual son:

Cuadro 3.4: Elementos del archivo XML

Elemento	Definición y atributos	
`simulation`	Es el elemento más externo, especifica los parámetros básicos y generales que aplican al modelo.	
	start	Define el momento de inicio de la simulación, dado en días.
	deltaT	Indica el nivel de detalle de avance en el tiempo, puede servir para correr simulaciones que sean más representativas en relación a la vida de las plantas, ya que al especificar su edad, este valor hará que los eventos ocurran más o menos rápido. Se mide en días al igual que el valor de comienzo y fin.
	end	Este parámetro indica hasta qué momento la simulación finaliza, cuando el tiempo llega a este momento el programa termina.
	area	Indica el tamaño del grid que se usará, el área abarca desde $[-A, A]$ tanto en x como en y, donde A es el valor proporcionado.
`event`	Ocurre dentro de un `simulation`, asocia un acontecimiento a un momento en el tiempo (medido en días). tiene dos parámetros:	
	days	Es el número de día en el que este evento es lanzado, si la simulación avanza en pasos que no corresponden exactamente a este valor, el evento ocurre cuando se supera el valor especificado.

Continúa en la siguiente página…

Elemento	Definición y atributos	
`plant`	**save**	Indica que inmediatamente después de lanzar este evento, un archivo será salvado con el estado de la simulación. El archivo tiene el nombre ”data_N.inc” donde N es el día en que ocurre el evento.
	repeat	Indica que este evento se ha de repetir N veces, donde N es el valor proporcionado.
	Dentro de un `event` puede ocurrir un `agentstart` con un único parámetro *id* que especifica el agente que se va a lanzar. De modo redundante existe un elemento `save` que recibe un parámetro *fname* y que sirve para salvar en determinado archivo (si se repite, el archivo se sobreescribe).	
	Se especifica dentro de un `simulation` y sirve para definir los parámetros de las especies que se van a simular. Se puede definir su comportamiento como se muestra en [23] por medio de la función de viabilidad y supervivencia de la especie, ver sección 3.1.2 en la página 27	
	id	Este parámetro define el número que se usa para referirse a esta planta en los otros elementos.
	name	Valor informativo, es el que se imprime en el menú.
	fname	Es el nombre del archivo ”fname.inc” que se usará al generar el resultado de la simulación, este archivo debe de incluirse previamente para renderear en Povray.
	initial	El número inicial de plantas que existen, este número es el que se coloca por omisión en el menú pero puede ser modificado en el programa.

Continúa en la siguiente página...

Elemento	Definición y atributos	
	viability	El nombre de la función de viabilidad que se usará dentro del programa, esto está fijo en el código, hay funciones desde la A a la K y corresponden a las que se presentan en [23].
	rad	Es el área de cobertura biológica que ocupa la planta. Consultar la sección 3.1.3 en la página 31.
	mult	Es un factor que se multiplica por el número aleatorio de semillas que libera una planta para ajustar su difusión, para mayor detalle ver la sección 3.1.3 en la página 31.
	age	El número de días que vive una planta, ver sección 3.1.3 en la página 31.
	seeds	La cantidad máxima de semillas que pueden ser liberadas, es una referencia ya que el número exacto es aleatorio. Consultar la sección 3.1.3 en la página 31.
`agentspec`	Por medio de este elemento se especifican los tipos de agentes que se usarán en la simulación, su comportamiento en términos de las tareas básicas definidas y la manera como la ejecutarán en el espacio simulado.	
	id	El número que sirve de referencia a este tipo de agentes.
	name	Campo informativo para los mensajes que se imprimen.
	span	Cantidad de agentes que serán generados.
	maxpath	Es el máximo que dura la ruta total del agente, se puede pensar como la distancia máxima que el agente camina.

Continúa en la siguiente página...

Elemento	Definición y atributos	
	vision	Es la distancia hasta la cual el agente tiene percepción de su entorno, cuando una planta está fuera de este rango es ignorada.
`behaviour`	El comportamiento del agente, esto consiste de una acción que está definida para cierta especie de plantas y limitada por cierto espacio (hasta abarcar el campo completo. No tiene parámetros, pero sirve para englobar a un conjunto de `kill`, `seed`, `excite` o `inhibit` relacionados con varios `constraint`.	

`kill`
`seed`
`excite`
`inhibit`
	Definen qué acción ejecuta el agente. Se debe especificar qué especie de planta es la afectada y en su caso un porcentaje o monto a excitar o inhibir el crecimiento.	

	plantId	Qué especie de planta es la afectada en este behaviour.
	amount	Para las acciones de `excite` e `inhibit` es el porcentaje de la edad de la planta que el crecimiento será retardado o acelerado.
`constraint`	Detalla el espacio en el que los agentes actuarán, si es omitido entonces los agentes ocuparán todo el espacio posible en el campo. No tiene atributos, solamente sirve para agrupar las figuras que se pueden escoger como parte del comportamiento.	
`rectangle`	Especifica un rectángulo, sus atributos son *x*, *y*, *height*, *width* y *amount*. Éste último especifica la cantidad de plantas que se siembran por especie en el caso de que la acción sea `seed`.	
`circle`	Círculo definido por los atributos *x*, *y*, *radius* y *amount* (que aplica solo para `seed`).	

Continúa en la siguiente página…

Elemento	Definición y atributos
`lines`	Sembrar en líneas como un granjero, se define mediante la coordenada superior izquierda *x*, *y* datos del rectángulo que se abarca *width*, *height*, y un parámetro para determinar cada cuánto se hacen las líneas *deltarows*. No hay monto de plantas a sembrar pero aleatoriamente se siembran aproximadamente quinientas por línea.
`image`	Usando el atributo *fname* indica un archivo que se ha de tomar de entrada para mapear a la región que afectan los agentes. La imagen ha de ser en formato PCX en escala de grises y de 128x128 pixeles.

Existen algunas combinaciones de elementos con sub-elementos en el XML que la sintaxis permite pero que no tienen sentido y que deben ser revisados por el programa, un ejemplo de esto son las acciones contra el área en el que deben aplicarse, ver la tabla 3.5.

	`circle`*	`rectangle`*	`lines`	`image`
`kill`	X	X		X
`seed`	X	X	X	
`excite`	X	X		
`inhibit`	X	X		

*Soporta el operador `negative`

Cuadro 3.5: Concordancia de acciones con operadores para limitar el área

3.3.4. EJEMPLOS DE XML

En el cuadro 3.6 observamos un ejemplo de los valores iniciales que se dan a la simulación, dura 1000 días y tiene un evento en el día 200, el agente 4 va a comenzar su actividad y cuando termine se va a guardar el estado en un archivo para posteriormente renderearlo. La simulación avanza de 3 en 3 días.

```
<simulation start="0" deltaT="3" end="1000" area="8">
    <event days="200" save="1">
        <agentstart id="4"/>
    </event>
</simulation>
```

Cuadro 3.6: Parámetros generales de la simulación

A continuación (cuadro 3.7) se muestra la especificación para una de las especies de plantas. Una definición así produce que se agrege un elemento en el menú para dar valores iniciales de población. Aquí se usa la función K definida en la sección 3.1.2 en la página 27.

```
<plant
    id="0"
    name="Blade"
    fname="blade"
    initial="3000"
    viability="funcionk"
    rad="0.001"
    mult="50"
    age="730"
    seeds="15"
/>
```

Cuadro 3.7: Especie: ”Blade”

El cuadro 3.8 es una especificación de un agente que corta en círculo, podemos observar distintos elementos como el nombre, rango de visión, agentes que se generan, recorrido máximo, etc. Así como especificación de una figura en la sección `constraint` y un comentario como es estándar en xml.

```
    <agentspec id="4" name="cortar en circulo"
     span="50" maxpath="10" vision="0.5">
        <behaviour>
            <kill plantId="0"/>
            <kill plantId="1"/>
            <kill plantId="2"/>
            <kill plantId="3"/>
            <kill plantId="4"/>
            <kill plantId="5"/>
            <kill plantId="6"/>
            <constraint>
            <!-- <image fname="prueba.pcx" negative="1"/> -->
                <circle x="0" y="0" r="3"/>
            </constraint>
        </behaviour>
</agentspec>
```

Cuadro 3.8: Especificación de un agente

3.4. RESUMEN

En el presente capítulo se han mostrado los distintos elementos que componen la solución presentada. Se parte de cómo se ha modelado a las plantas, especificando las adiciones en cuanto a comportamiento y cálculo de viabilidad, muestras de cómo las variables fundamentales determinan la simulación y de cómo se pueden introducir fácilmente nuevos modelos a las imágenes de tal modo que es posible crear distribuciones espaciales de muy diversos tipos, con resultados muy distintos también.

Se han detallado las características de los agentes, caracterizándolos como semi-reactivos, que tienen una tarea común pero no se especifica el script específico para cada uno. Los agentes tienen memoria y se comunican mediante mecanismos muy sencillos. También se ha mostrado de qué modo se puede incorporar aprendizaje en la manera como reaccionan con el ecosistema y se ha bosquejado la metodología que se debe seguir para determinar su comportamiento y crear las imágenes que se muestran en los resultados en la sección 5 en la página 59.

4. IMPLEMENTACIÓN

4.1. CARACTERÍSTICAS DEL PROGRAMA

El programa desarrollado se implementó primero en C++ para definir clases y métodos para los agentes y plantas. Cada elemento del ecosistema es un agente y como tal tiene comportamiento y ciertas operaciones comunes como la manera como se dibuja a baja o alta resolución, etc. Usando este enfoque se implementó el algoritmo de árboles k-dimensionales para resolver las colisiones como se explica en la sección 3.2.3. La simulación se volvió lenta debido a las operaciones de creación y eliminación de los objetos. Los máximos niveles de simulación consistían en unas 15000 plantas.

Posteriormente se reimplementó regresando al esquema original donde se tiene un gran arreglo en memoria y las operaciones se realizan a más bajo nivel marcando banderas y elementos de los arreglos. De este modo se llegó a simular y a obtener datos para posteriormente renderear, hasta de unas 200000 plantas.

Se integró un parser de XML de acuerdo a lo especificado en la sección 3.3.3 en la página 49, esto lo puede consultar el usuario desde el inicio de la simulación ya que es un parámetro requerido el tener el archivo de entrada con los parámetros iniciales. Una vez que el programa verifica la sintaxis y consistencia de los datos, imprime el resultado en la pantalla como podemos observar en la figura 4.1 en la página siguiente.

Posteriormente accesamos a la pantalla para ajustar el número de población inicial que se desea. Este menú es dinámico y depende de las especies que se definan en el archivo de entrada en XML, podemos observarlo en la figura 4.2 en la página siguiente.

Aquí el usuario puede seleccionar si se imprimirá la actividad de los agentes, si desea visualizar a las plantas y dar inicio o detener la simulación. También se proporciona un botón para reinicializar la simulación y para generar un archivo de descripción de la escena en el POVray.

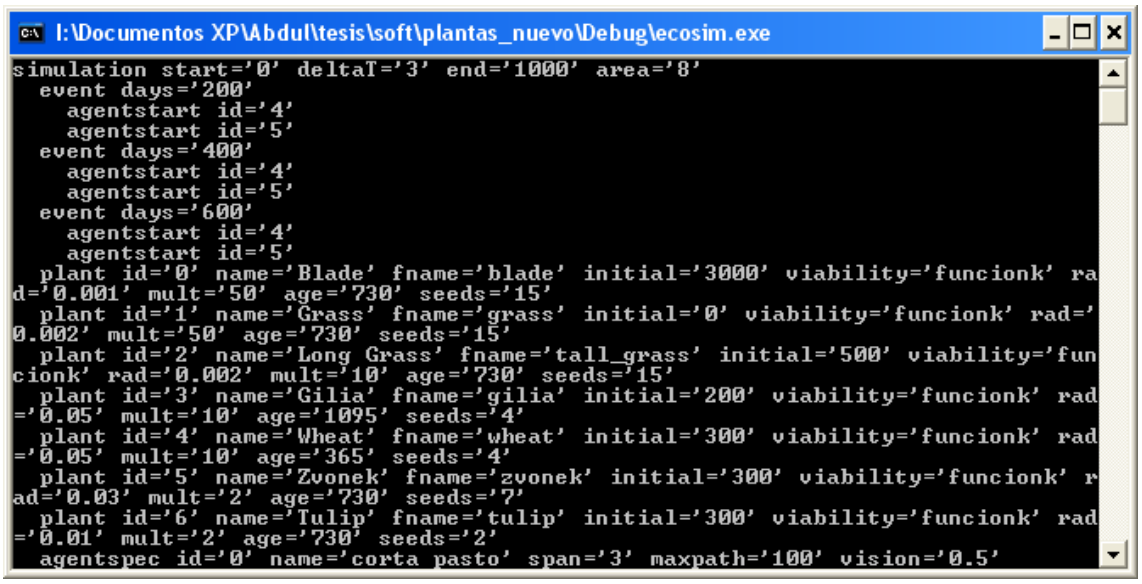


Figura 4.1: Resultado de parsear y verificar sintaxis del archivo xml

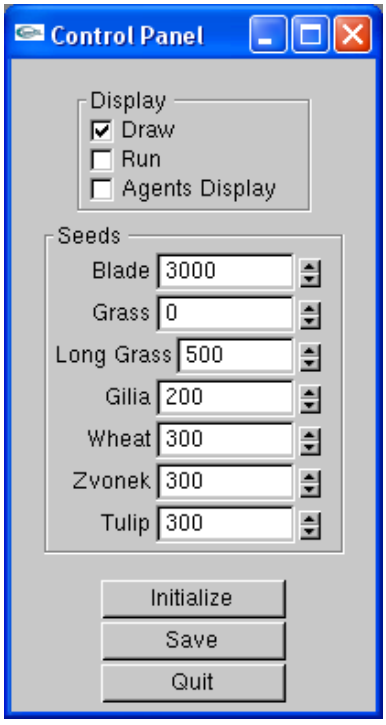
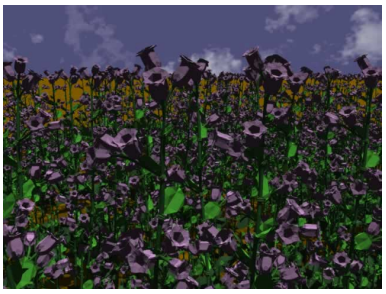


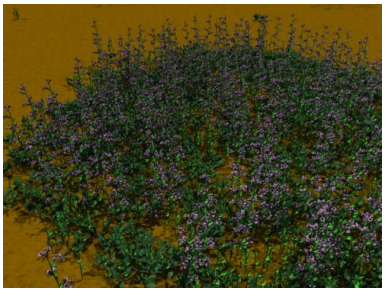
Figura 4.2: Menú del programa desde donde se pueden alterar los valores iniciales de población

5. RESULTADOS

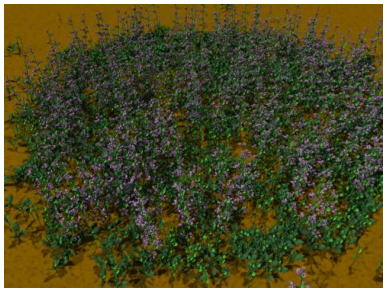
En la serie de ilustraciones 5.1 tenemos un campo que ha sido cortado después de 500 días. La población inicial fue de 2000 plantas, en este caso solmente una especie *campanas*. Después de ese tiempo se envían a 50 agentes con un radio de visión de 0.5 metros a cortar siguiendo un círculo de radio de 4 metros. El campo es cuadrado y con lados de 16 metros. Como resultado se tienen alrededor de 10000 plantas en un patrón circular. Se puede observar que algunas siguen creciendo fuera del área de interés, esto es por lo estocástico de la simulación. Los resultados son más realistas que si el corte se determina proceduralmente discriminando por zonas.



(a) Dentro del campo.



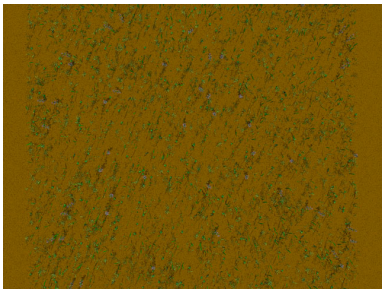
(b) Toma aérea de frente



(c) Toma aérea lateral

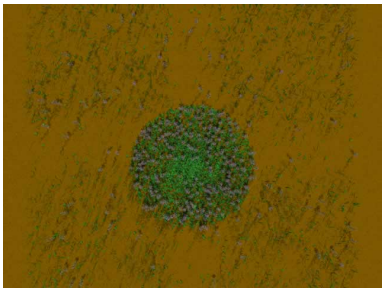
Figura 5.1: Terreno sembrado con campanas, los agentes han cortado en círculo.

En la figura 5.2 en la página siguiente Observamos una simulación más compleja. Aquí el campo inicialmente tiene 3000 hojas de pasto corto y 2000 de otra especie de pasto que ocupa más espacio, 300 unidades de trigo, 300 de tulipanes y 300 de campanas. La simulación se deja correr durante 200 días. Luego se envía a 50 granjeros a que corten en patrones rectangulares formando dos caminos que cruzan el campo, la simulación de esto la podemos ver en la figura 3.18 en la página 45. La parte central (un círculo) se deja sin cortar. Inmediatamente después se envía a los granjeros a que siembren 500 campanas adicionales en el centro. Después de otros 200 días se vuelven a enviar a los granjeros a que ejecuten las mismas tareas y tenemos la imagen 5.2(d) en la página siguiente donde ya han brotado flores de las campanas.

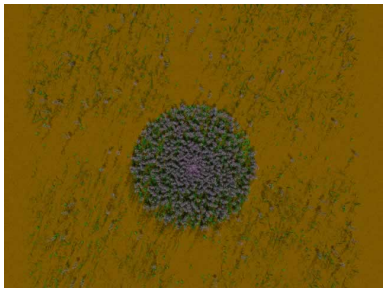


(a) Etapa inicial

(b) Un poco más adelante, las plantas crecen desordenadamente.



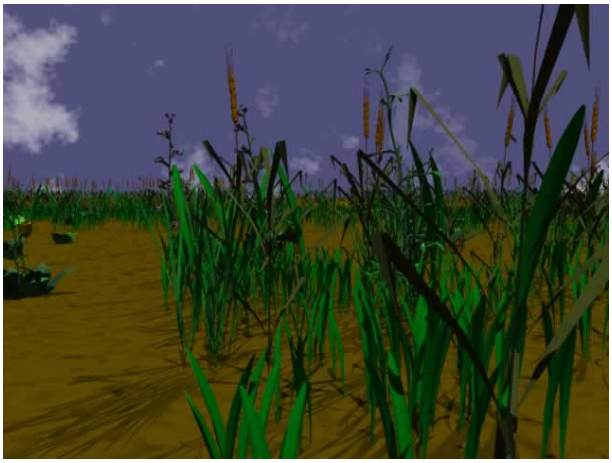
(c) Se cortan caminos horizontal y verticalmente y en el círculo central se deja crecer más pasto y se siembran campanas.



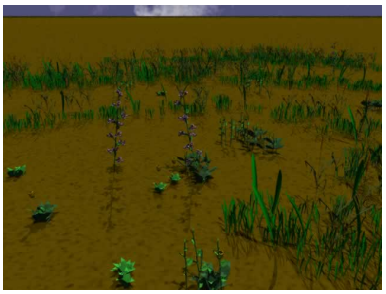
(d) Las campanas comienzan a brotar.

Figura 5.2: Etapas de preparación de un jardín.

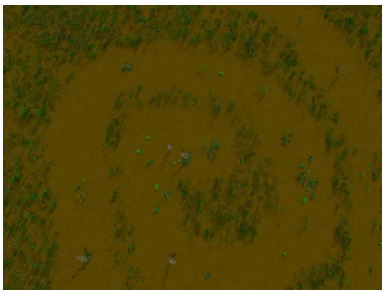
La ilustración 5.3 en la página siguiente muestra tomas aéreas de un terreno en el que los granjeros han trabajado siguiendo una imagen como patrón. Sus tareas han sido podar el pasto en la parte oscura y dejar el paso libre a las flores en esa misma zona. El estado en el que se muestra el terreno es después de 600 días de simulación, los agentes se enviaron tres veces, cada 200 días. La población a los 600 días fue de 23000 plantas distribuidas en cuatro especies.



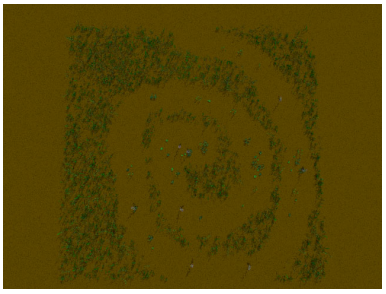
(a) Dentro del terreno



(b) Desde la estatura de una persona no es muy notorio el patrón



(c) Al comenzar a elevarse la figura se aclara

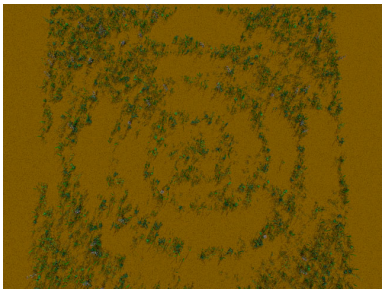


(d) Vista abarcando todo el terreno

(e) Imagen desde la que se organizó la tarea de los agentes

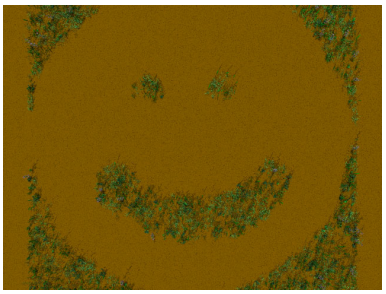
Figura 5.3: Vuelo sobre un campo sembrado en espiral

En la tabla de figuras 5.4 en la página siguiente se muestran otros resultados donde a partir de la imagen se ha generado el jardín de la izquierda. La cantidad de planta varía de 20 a 30 mil plantas.



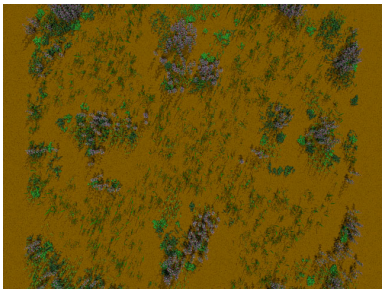
(a) Imagen después de 600 días

(b) Mapa de bits

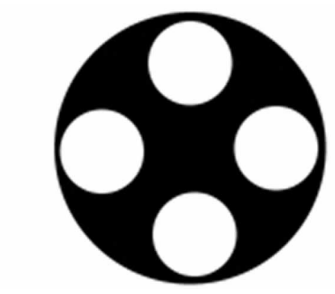


(c) Imagen después de 700 días

(d) Mapa de bits



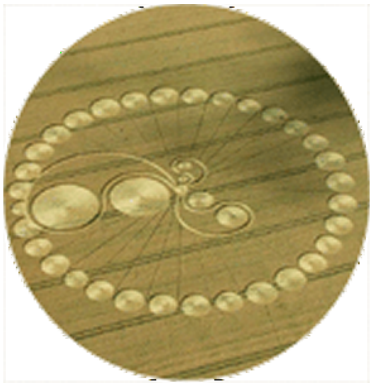
(e) Imagen después de 224 días



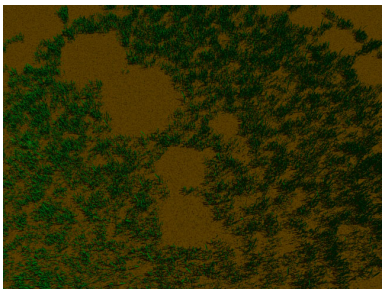
(f) Mapa de bits

Figura 5.4: Campos cortados siguiendo el patrón a la derecha

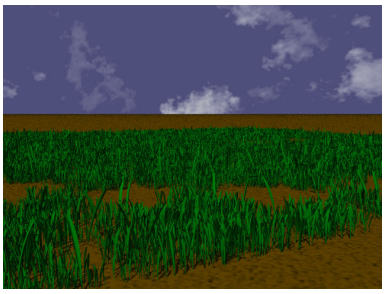
Finalmente se compara un crop circle como los mostrados en la sección 3.3 en la página 44 y el resultado de una simulación basada en el mismo trazo sobre el terreno. Se dibujan 130000 plantas de la misma especie, la población inicial fue de 10000 y el simulador corrió durante 700 días con los agentes visitando cada 100 días del 300 al 700 (fig. 5.5 en la página siguiente).



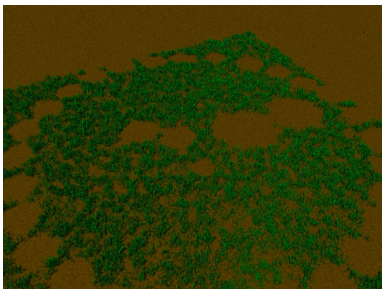
(a) Imagen original



(b)



(c)



(d)

Figura 5.5: Comparación de un crop circle real con uno sintético

6. CONCLUSIONES

6.1. RESUMEN DE LOS RESULTADOS

Hemos mostrado un modelo de interacción de agentes externos con el desarrollo de ecosistemas. Se han generado imágenes en las que la distribución espacial no solo depende de las variables internas, sino también de un proceso inteligente de agentes con un comportamiento definido.

Los agentes desarrollados para esta tarea tienen características especiales ya que debido a lo exigente de la simulación en cuanto a cantidad de elementos, se tuvo que buscar una implementación que pudiera funcionar aún con una saturación del terreno, con el fin de incrementar la capacidad en número de plantas que se pueden incorporar y así obtener fotorrealismo en los resultados.

Mediante el simulador generado y la parametrización que se obtuvo, es posible modelar una amplia gama de comportamiento tanto en las plantas como en los agentes. Es posible crear escenarios con una apariencia muy realista y en poco tiempo. Los modelos así obtenidos son dinámicos en el sentido de que consideran el crecimiento de las plantas. Es posible mediante los agentes incorporar eventos que puedan modificar su desarrollo.

Dado que los agentes pueden seguir un área en particular para la ejecución de sus tareas, es posible crear imágenes muy parecidas a los *crop circles* y en general diseños en la creación de jardines.

6.2. INVESTIGACIÓN FUTURA

Existen varios métodos de búsqueda y partición del espacio explicados en la sección 3.2.3 en la página 41, sin embargo el foco principal de este trabajo no era buscar uno óptimo sino uno que funcionara bien para las simulaciones a realizarse. Para poder elevar el número de plantas en el campo es necesario usar un método más eficiente y valdría la pena comparar otros métodos multidimensionales, que además tenga la característica de que se puede eliminar de modo sencillo ya que la operación que más tarda en este momento en cada ciclo es reconstruir el arreglo de plantas según las que han muerto y además reconstruir la reja que sirve para la detección de colisiones.

La interpretación de los parámetros iniciales de las plantas no se ha comparado con modelos biológicos reales, este trabajo puede ser extendido si se toman datos estadísticos en la naturaleza y se ajustan a la simulación para verificar su consistencia.

Los agentes se han visto como granjeros, a pesar de que se han definido otros comportamientos, solamente se ha explotado para la generación de imágenes el que corresponde a cortar y sembrar plantas. Queda como trabajo futuro estudiar las características de los comportamientos de inhibir y excitar el crecimiento ya que estos además pueden modelar otros fenómenos en la naturaleza, no solamente *macroagentes*.

Del mismo modo, no se ha estudiado la posibilidad de *microagentes* como podrían ser insectos, lluvia, enfermedades que modifiquen a un nivel de detalle distinto las características de las plantas.

Las funciones de viabilidad no están definidas para los agentes actualmente, si se hace se puede completar el trabajo de los microagentes como fenómenos de la naturaleza.

Se bosquejó en la sección 3.1.2 en la página 27 la manera como se pueden alterar las funciones de viabilidad, sin embargo este trabajo no está hecho y de este modo se puede pensar en un aprendizaje de los agentes cuya función evolucione a través del tiempo. Esto puede modelar el que las plantas se adapten a condiciones naturales, que los agentes sean animales que reaccionan de alguna manera al interactuar con las plantas como desarrollar preferencias o aversiones dependiendo de la especie.

Bibliografía

[1] 3LOG. http://www.3log.com/index.shtml.

[2] Crop Circle Connector. http://www.cropcircleconnector.com.

[3] Crop Circles Archive. http://www.cropcircle-archive.com/.

[4] D. R. Systems Inc. http://www.drsystemsinc.com/.

[5] Forest Growth Model for Northwest Germany. http://www.nfv.gwdg.de/A/BwinPro/bwinpro.php.

[6] HALCO. http://www.halcosoftware.com/software.html.

[7] Persistence of vision raytracer. http://www.povray.com.

[8] Phoenix: An Adaptable Planner for a Complex, Real-time Environment. http://eksl-www.cs.umass.edu/research/phoenix.html.

[9] Plantstudio. http://www.kurtz-fernhout.com.

[10] Softree. http://www.softree.com/.

[11] Tradetec. http://www.tradetec.com/.

[12] TreeGross – Tree Growth Open Source Software. http://treegross.sourceforge.net.

[13] Woodlands. http://www.tradetec.com/.

[14] WWW-Server for Ecological Modelling. http://dino.wiz.uni-kassel.de/ecobas.html.

[15] The adventures of andré and wally. Lucasfilm Ltd., 1982.

[16] ABELSON, H., AND DISESSA, A. *Turtle Geometry*. MIT Press, 1986.

[17] AONO, M., AND KUNII, T. L. Botanical tree image generation. *IEEE Computer Graphics and Applications 4*, 5 (May 1984), 10–29, 32–34.

[18] ARVO, J., AND KIRK, D. Modeling plants with environment-sensitive automata. *Ausgraph'88* (1988), 27–33.

[19] BENEŠ, B. An efficient estimation of light in simulation of plant development. In *Computer Animation and Simulation '96* (1996), R. Boulic and G. Hegron, Eds., Eurographics, Springer-Verlag Wien New York, pp. 153–165. Proceedings of the Eurographics Workshop in Poitiers, France, August 31–September 18, 1996.

[20] BENEŠ., B. A stable modeling of large plant ecosystems. *International Conference on Computer Vision and Graphics* (2002).

[21] BENEŠ, B., CÓRDOBA, J. A., AND SOTO, J. M. A real-time visual models of plantas interacting with virtual environment. *Workshop on Intelligent Vitual Environments* (2002).

[22] BENEŠ, B., CÓRDOBA, J. A., AND SOTO, J. M. Interacting agents with memory in virtual ecosystems. *The 11-th International Conference in Central Europe on Computer Graphics* (February 2003).

[23] BENEŠ, B., AND ESPINOZA, E. D. Modeling virtual ecosystems with the proactive guidance of agents. *Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA '03)* (2003).

[24] BENTLEY, J. L. Multidimensional binary search trees used for associative searching. *Commun. ACM 18*, 9 (1975), 509–517.

[25] BIGING, G., AND DOBBERTIN, M. Evaluation of competition indices in individual tree growth models. *Forest Science 41*, 2 (1995), 360–377.

[26] BROOKS, R. A. *Cambrian Intelligence: the early history of the new AI*. A Bradford Book, 1999.

[27] BROOKS, R. A. Elephants don't play chess. In *Cambrian Intelligence: the early history of the new AI* [26].

[28] BROOKS, R. A. Intelligence without representation. In *Cambrian Intelligence: the early history of the new AI* [26].

[29] BROOKS, R. A. New approaches to robotics. In *Cambrian Intelligence: the early history of the new AI* [26].

[30] BROOKS, R. A. A robot that walks: emergent behaviors from a carefully evolved network. In *Cambrian Intelligence: the early history of the new AI* [26].

[31] BROOKS, R. A. A robust layered control system for a mobile robot. In *Cambrian Intelligence: the early history of the new AI* [26].

[32] BURGER, H. Baumkrone und zuwachs in zwei hiebsreifen fichtenbeständen. *Mitt. Schweiz. Anst. f. d. forstl. Versuchsw 21* (1939), 147–176.

[33] BURGER, H. Kronenaufbau gleichaltriger nadelholzbestände. *Mitt. Schweiz. Anst. f. d. forstl. Versuchsw 21* (1939), 5–58.

[34] CHAMBERLAIN, B., DEROSE, T., LISCHINSKI, D., SALESIN, D., AND SNYDER, J. Fast rendering of complex environments using a spatial hierarchy. In *Graphics Interface '96* (1996), W. A. Davis and R. Bartels, Eds., Canadian Human-Computer Communications Society, pp. 132–141.

[35] DEUSSEN, O., HANRAHAN, P., LINTERMANN, B., MECH, R., PHARR, M., AND PRUSINKIEWICZ, P. Realistic modeling and rendering of plant ecosystems. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), ACM Press, pp. 275–286.

[36] DI GIACOMO, T., CAPO, S., AND FAURE, F. An interactive forest. pp. 65–74.

[37] FRITSCH, F., AND SALISBURY, E. *Plant Form and Function*. G. Bell & Sons Ltd, 1965.

[38] FUNGE, J., TU, X., AND TERZOPOULOS, D. Cognitive modeling: knowledge, reasoning and planning for intelligent characters. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 29–38.

[39] GREENE, N. Voxel space automata: modeling with stochastic growth processes in voxel space. *SIGGRAPH 89, ACM Computer Graphics, July 1989 23*, 3 (1989), 175–184.

[40] HEBB, D. O. *The Organization of Behavior*. Wiley, 1949.

[41] HOLTON, M. Strands, gravity and botanical tree imagery. *Computer Graphics Forum 13*, 1 (Mar. 1994), 57–67.

[42] JAMES, M., HAMMAL, M., HANAN, J., MECH, R., AND PRUSINKIEWICS, P. *CPFG Version 2.7 User's Manual*.

[43] KAJIYA, J. T., AND KAY, T. L. Rendering fur with three dimensional textures. *Computer Graphics (SIGGRAPH '89 Proceedings) 23*, 3 (July 1989), 271–280.

[44] LANE, B. Models of plant communities for image synthesis, June 2001.

[45] LINTERMANN, B., AND DEUSSEN, O. Interactive modelling and animation of branching botanical structures. *Computer Animation and Simulation '96* (1996), 139–151. Proceedings of the Eurographics Workshop in Poitiers, France, August 31–September 18, 1996.

[46] LLUCH, J., CAMAHORT, E., AND VIVÓ, R. Procedural multiresolution for plant and tree rendering. In *Proceedings of the 2nd international conference on Computer graphics, virtual Reality, visualisation and interaction in Africa* (2003), ACM Press, pp. 31–38.

[47] MACDONALD, N. *Trees and Networks in Biological Models*. John Wiley, Chichester, 1983.

[48] MACIEL, P. W. C., AND SHIRLEY, P. Visual navigation of large environments using textured clusters. In *Symposium on Interactive 3D Graphics* (1995), pp. 95–102, 211.

[49] MASON, E. G. Forestry. http://www.fore.canterbury.ac.nz/EUAN.HTM.

[50] MAX, N., AND OHSAKI, K. Rendering trees from precomputed Z-buffer views.

[51] MAX, N., AND OHSAKI, K. Rendering trees from precomputed Z-buffer views. In *Proceedings of the 6th Eurographics Workshop on Rendering* (1995).

[52] MCCARTHY, J., MINSKY, M. L., ROCHESTER, N., AND SHANNON, C. E. A proposal for the dartmouth summer research project on artificial intelligence, August 31, 1955.

[53] MCCULLOCH, W., AND PITTS, W. H. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* (1943).

[54] MĚCH, R., AND PRUSINKIEWICZ, P. Visual models of plants interacting with their environment. 397–410. held in New Orleans, Louisiana, 04-09 August 1996.

[55] MINSKY, M. A framework for representing knowledge. In *The Psichology of Computer Vision* (Winston PH), vol. 17, McGraw Hill.

[56] MINSKY, M., AND PAPERT, S. Perceptrons: An introduction to computational geometry.

[57] NEWELL, A., AND ERNST, G. W. The search for generality. *Information Processing 1* (1965), 17–24.

[58] OPPENHEIMER, P. E. Real time design and animation of fractal plants and trees. D. C. Evans and R. J. Athay, Eds., vol. 20, pp. 55–64.

[59] P. PRUSINKIEWICZ, J. HANAN, M. H., AND MĚCH, R. L-systems: from the theory to visual models of plants. *Machine Graphics and Vision* (1993), 2(4):12–22.

[60] PARISY, O., AND SCHLICK, C. A physically realistic framework for the generation of high-level animation controllers. In *Proceedings of the 2nd international symposium on Smart graphics* (2002), ACM Press, pp. 47–54.

[61] POWER, J. L., BRUSH, A. J. B., PRASINKIEWIC, P., AND SALESIN, D. H. Interactive arrangement of botanical L-system models (color plate S. 234). 175–182.

[62] PRUSINKIEWICZ, P., JAMES, M., AND MĚCH, R. Synthetic topiary. 351–358. ISBN 0-89791-667-0.

[63] PRUSINKIEWICZ, P., LINDENMAYER, A., HANAN, J. S., ET AL. *The algorithmic beauty of plants*. Springer-Verlag, New York, 1990.

[64] REEVES, W. T., AND BLAU, R. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *Computer Graphics (SIGGRAPH '85 Proceedings) 19*, 3 (July 1985), 313–322.

[65] REYNOLDS, C. W. Computer animation with scripts and actors. *Computer Graphics 16*, 3 (July 1982), 289–296.

[66] REYNOLDS, C. W. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (1987), ACM Press, pp. 25–34.

[67] SEDGEWICK, R. *Algorithms in C*. Computer Science. Addison-Wesley, 1990.

[68] SHANNON, C. E. Automatic chess player. *Scientific American 182* (1950), 48–51.

[69] SHANNON, C. E. Programming a digital computar for playing chess. *Philosophical Magazine 41*, 256 (1950), 75.

[70] SMITH., A. R. Plants, fractals, and formal languages. *Computer Graphics 18(3)* (1984), 1–10.

[71] SUTHERLAND, I. *A Man Machine Graphical Communication System*. PhD thesis, Massachusetts Institute of Technology, Jan. 1963.

[72] TECCHIA, F., AND CHRYSANTHOU, Y. Real-Time rendering of densely populated urban environments. pp. 83–88.

[73] TECCHIA, F., LOSCOS, C., AND CHRYSANTHOU, Y. Image-based crowd rendering. *IEEE Computer Graphics and Applications 22*, 2 (Mar./Apr. 2002), 36–43.

[74] TECCHIA, F., LOSCOS, C., CONROY, R., AND CHRYSANTHOU, Y. Agent behaviour simulator (abs): A platform for urban behaviour development. University College London, Department of Computer Science. This work was in part supported by the EPSRC project, GR/R01576/01 and the EPSRC Interdisciplinary Research, Centre equator.

[75] WAND, M., FISCHER, M., PETER, I., AUF DER HEIDE, F. M., AND STRASSER, W. The randomized z-buffer algorithm: Interactive rendering of highly complex scenes. In *SIGGRAPH 2001, Computer Graphics Proceedings* (2001), E. Fiume, Ed., ACM Press / ACM SIGGRAPH, pp. 361–370.

[76] WEBER, J., AND PENN, J. Creation and rendering of realistic trees. 119–128. held in Los Angeles, California, 06-11 August 1995.

[77] WENSEL, L., MEERSCHAERT, W., AND BIGING, G. Tree height and diameter growth models for northern california conifers. *Hilgardia, University of California 55*, 8 (1987).