Uso de la metodología GAIA para modelar el comportamiento de personajes en un juego de estrategia en tiempo real

Use of GAIA for modeling the behavior of characters in a real time strategy game

Norman Muñoz, Carlos Cobos[*]*, William Rivera, Jaime López, Martha Mendoza*

Grupo de I+D en Tecnologías de la Información, Departamento de Sistemas, Universidad del Cauca. Sector Tulcán, Edificio FIET, Oficina 422, Popayán, Colombia

(Recibido el 13 de enero de 2009. Aceptado el 15 de febrero de 2010)

Resumen

Los juegos modernos en su permanente búsqueda por capturar más la atención de sus jugadores, requieren de técnicas de Inteligencia Artificial que les permita a los personajes asumir comportamientos más cercanos o parecidos a la realidad. Para lograr esto, existen muchas técnicas que se pueden utilizar, pero en los últimos años el uso de agentes inteligentes ha tomado gran importancia, ya que las propiedades de los personajes encajan perfectamente con la definición de dicha técnica. En este artículo, se presentan los principales resultados de investigación y la forma como se usó la metodología GAIA para el modelamiento del comportamiento de los personajes en un juego de estrategia de tiempo real llamado Independencia (basado en la Revolución de los Comuneros de la Independencia de Colombia).

----- *Palabras clave*: GAIA, agentes, modelamiento, juego de estrategia

Abstract

The modern games in its permanent search to get players attention in a greater way require artificial intelligence techniques that allow the characters to acquire behaviors closer or similar to reality. To do this, there are many techniques that can be used, but recently the use of intelligent agents has gained great importance, since the characters properties fit perfectly with the definition of this technique. In this paper, we present the main results of research and the way how the GAIA methodology was used for modeling the

* Autor de correspondencia: teléfono: + 57 + 2 + 820 98 00 ext. 2119, fax: + 57 + 2 + 820 98 00 ext. 2102, correo electrónico: ccobos@unicauca.edu.co. (C. Cobos).

behavior of the characters in a real time strategy game called Independencia (based on the "Comuneros" Revolution of the Colombian Independence).

----- *Keywords:* GAIA, agents, modeling, strategic game

Introducción

Existen muchas técnicas para el modelado del comportamiento de los personajes en los juegos modernos [1], destacándose las máquinas de estados finitos (MEF, permiten modelar el comportamiento de un sistema a través de un número limitado de estados y un conjunto de transiciones de estado), los agentes y el scripting (lenguaje de programación de cuarta generación que simplifica alguna tarea compleja para un programa particular; su alcance puede variar desde un simple archivo de configuración hasta un complejo lenguaje interpretado en tiempo de ejecución [1]), sin demeritar la importancia de las redes neuronales y la lógica difusa, entre otras. Sin embargo, las tres primeras se han impuesto en esta área, debido a su fácil codificación y a los excelentes resultados que ofrecen en el corto tiempo [2].

En este artículo se presentan los principales resultados de investigación de un proyecto que buscó el desarrollo de un video juego de estrategia en 3D basado en un hecho histórico de la campaña de Independencia de la República de Colombia denominado INDEPENDENCIA 1.0 [3]. Este videojuego se basó en la revolución de los Comuneros y muestra la campaña desde el Socorro (Santander - Colombia) hasta la Capital bajo las órdenes del general Juan Francisco Berbeo. En la figura 1 se muestra una imagen de algunos de los personajes que fueron implementados en el juego de estrategia, Tres campesinos a pie y uno a caballo, confrontando a dos peones realistas (conquistadores).

A continuación se presentan trabajos relacionados con los juegos, el concepto de agente y un resumen de GAIA [4]. En la siguiente sección, se presenta la forma como se usó GAIA para el análisis y diseño del juego, y luego se muestran los resultados de la investigación: arquitectura general de un motor orientado a objetos, visualizador de modelos, editor de terrenos, buscador de caminos y el juego en si mismo. Finalmente, se muestran las conclusiones y el trabajo futuro.

Figura 1 Personajes del juego de estrategia

Trabajos relacionados

El interés por dotar a los personajes de los videojuegos de un comportamiento más próximo al del ser humano no es nuevo. John Laird y Michael Van Lent [5] de la Universidad de Michigan utilizan técnicas de Inteligencia Artificial, como los agentes software, para lograr este propósito. Es precisamente John Laird uno de los investigadores que ha hecho grandes aportes en el tema, ya que en su grupo de investigación ha desarrollado un proyecto (SOAR/Games Project)[5] basado en agentes que provee a los investigadores de un conjunto de ambientes para hacer pruebas en áreas como aprendizaje de máquinas, arquitecturas inteligentes y diseño de interfaces; y a los desarrolladores de juegos de un ambiente para hacer juegos más divertidos al poder crear agentes más inteligentes y reales [6]. Sin embargo no todos los programadores de videojuegos de estrategia deciden usar agentes para dotar de inteligencia a sus personajes, en tal sentido, pueden usar otras técnicas como por ejemplo el uso de grafos [7].

De todas maneras, si un desarrollador desea que el comportamiento de los personajes (o unidades) sea más próximo a la realidad, es conveniente utilizar técnicas de Inteligencia Artificial. Al

respecto Nayerek propone la utilización de agentes autónomos para el modelado del comportamiento de personajes en videojuegos, para probar que es posible aplicar esta técnica para resolver problemas como el manejo de tiempo real, dinámica, recursos y conocimiento incompleto [8]. Un buen ejemplo de la utilización de agentes en juegos es «Empire Earth» de Sierra [9] en el cual la Inteligencia Artificial se compone de varios agentes llamados administradores [10]. Juegos como Civilization [11], Balance of Power [12] y Populous [13] usan la tecnología de agentes para reaccionar a las acciones del jugador tal como lo haría un ser humano, logrando que los jugadores piensen que se trata de un oponente real.

Este trabajo, es un caso exitoso del uso de GAIA y de los agentes de software para modelar los personajes de un juego de estrategia, en el que además, se convirtieron los modelos basados en GAIA a una implementación en Visual C++ usando un motor de juegos orientado a objetos que también se desarrolló en el proyecto (dicho motor se explica en forma detallada en [14]).

Marco conceptual

Un agente es un sistema de computador capaz de realizar acciones independientes en beneficio de su propietario o usuario, es decir, son autónomos. Por esto, un agente es capaz de actuar independientemente, exhibiendo control sobre su estado interno, para lo cual hace uso de su percepción interna del ambiente en el que se desenvuelve [1]. Generalmente, los agentes en un juego son conjuntos de MEF que trabajan cada uno en un problema particular y se comunican con los demás.

GAIA es una metodología para realizar los procesos de análisis y diseño orientados a agentes. Esta metodología hace uso de una serie de modelos de análisis y diseño (ver [4] para más detalle). La etapa de análisis en GAIA busca el entendimiento del sistema y su estructura como una *organización*. Una organización puede ser vista como un conjunto de roles que mantienen ciertos tipos de interacción con otros roles en el sistema. Un rol se define en GAIA por cuatro atributos: las responsabilidades, los permisos, las actividades y los protocolos. Las responsabilidades, a su vez, pueden fraccionarse en dos categorías: las propiedades de vida y las propiedades de seguridad. Las responsabilidades determinan la funcionalidad y son tal vez los atributos claves asociados con un rol. Los permisos identifican los recursos que el rol tiene disponibles para realizar sus responsabilidades. Por lo general, estos tienden a ser recursos de información y con cada recurso habrá asociados unos derechos vinculados como por ejemplo: leer, cambiar o generar el recurso. Las actividades representan acciones privadas u operaciones que el agente puede completar sin la necesidad de interactuar con otros agentes en el sistema. Los protocolos definen las formas en que un agente asumiendo dicho rol podría interactuar con otros agentes. Durante el análisis con GAIA se generan dos tipos de modelos: de roles y de interacción.

El Modelo de Roles: Permiten identificar los distintos tipos de roles que se encuentran en el sistema. Con cada rol hay asociados unos permisos y unas responsabilidades. Las responsabilidades son de dos tipos, de vida y de seguridad. Con estos conceptos, se puede especificar un esquema de rol para cada tipo de rol en la organización y conformar así el modelo de roles. Un esquema de rol presenta una descripción del rol, sus protocolos y actividades, sus permisos y las responsabilidades de vida y seguridad.

El Modelo de Interacción: Es una serie de definiciones de protocolo, una para cada tipo distinto de interacción entre los roles. Cada definición de protocolo consiste de un propósito de la interacción, los roles iniciador y receptor de la misma, las entradas y salidas y el procesamiento ejecutado.

El proceso de diseño en GAIA involucra la transformación de los modelos de análisis a un nivel de abstracción suficientemente bajo para que sea posible implementar los agentes haciendo uso de las técnicas de diseño tradicionales, tales como las que hacen uso de la orientación a

objetos. En la etapa de diseño, GAIA hace uso de tres tipos de modelos: el modelo de agentes, el modelo de servicios y el modelo de conocidos.

El Modelo de Agentes: Permite definir los diferentes tipos de agente que se podrán encontrar en el sistema y el número de instancias que se tendrán de cada uno en tiempo de ejecución. Un tipo de agente es un conjunto de roles, es decir, un tipo de agente puede asumir uno o más roles aunque lo contrario no es cierto. El modelo de agentes se construye mediante un árbol de tipos de agentes, en el cual los nodos hoja corresponden a los roles y los restantes a los tipos de agentes. Para cada tipo de agente se debe definir un cuantificador de instancia, el cual precisa el número de instancias que se tendrán en el sistema en tiempo de ejecución (ver [4] para más detalles)

El Modelo de Servicios: Expone los servicios que cada tipo de agente va a implementar, entendiendo por servicio, cierta funcionalidad. Cada servicio es derivado de las actividades y protocolos, así como de sus propiedades de vida y seguridad encontradas en la etapa de análisis. El modelo de servicios se compone de las propiedades de cada uno de los servicios: las entradas, las salidas, las precondiciones y las poscondiciones. Las entradas y salidas proceden en forma directa del modelo de protocolos. Las precondiciones y las poscondiciones constituyen limitantes en los servicios y son derivadas de las propiedades de un rol.

El Modelo de Conocidos: Permite precisar los enlaces de comunicación que existen entre tipos de agentes e identificar posibles problemas de embotellamiento surgidos por el uso de estos canales de comunicación. El modelo es un grafo dirigido, en donde cada nodo corresponde a un tipo de agente y las aristas se relacionan con los caminos de comunicación. Así pues, un grafo **A** → **B**, muestra que hay un camino de **A** a **B**, pero no necesariamente de **B** a **A**.

El análisis y diseño de independencia

En Independencia se decidió tomar GAIA como metodología para el modelado de la inteligencia de los personajes del juego, ya que brinda dos fortalezas principales: 1) es orientada a agentes, lo cual permitiría posteriormente hacer uso de los mismos como eje central de la lógica del juego. 2) no impone restricciones en la implementación, brindando al desarrollador la posibilidad de escoger la estructura de implementación deseada [4]. Esta decisión no sólo permitió el modelado del comportamiento de los personajes, sino que permitió explorar una técnica de vanguardia en el mundo de los juegos y que está siendo adoptada por grandes casas de desarrollo de juegos como Sierra y su título Empire Earth.

GAIA se usó únicamente para el modelado del comportamiento de los personajes y todos los personajes que intervienen en la historia del juego representan integrantes de un ejército. Cada uno de ellos se denomina una Unidad, y en el caso especial de Juan Francisco Berbeo, es denominada Prócer. El comportamiento que puede ser asumido por éstos actores se resume en el diagrama de **casos de uso** que se muestra en la figura 2, en el cual se observa que el actor Unidad puede vigilar, caminar, atacar a un enemigo, recibir un ataque, escuchar el grito comunero y morir, mientras que el actor Prócer puede asumir todo el comportamiento de una Unidad y adicionalmente puede lanzar el grito comunero.

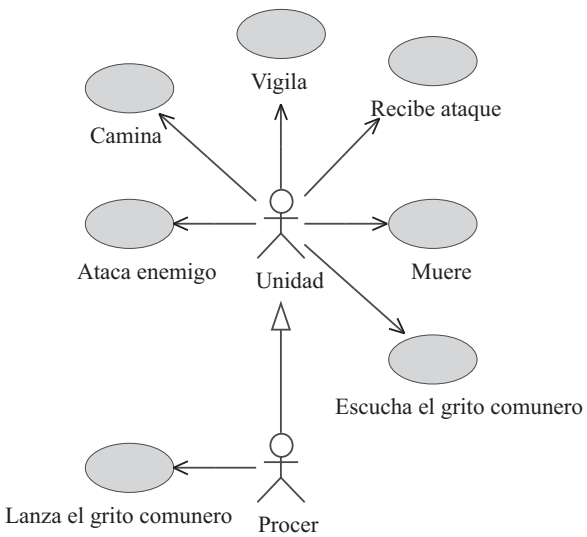

Figura 2 Casos de uso para el comportamiento de los personajes

Cuando una unidad se encuentra en estado de vigilancia, constantemente verifica si hay enemigos en el área dentro de su rango de visión y de ser así, decide atacar a la unidad enemiga visualizada. Si la unidad decide atacar, se mueve hasta que el objetivo se encuentre dentro del rango de ataque y cuando suceda lo anterior, la unidad hace que su enemigo reciba el ataque. Cuando una unidad recibe un ataque, disminuye sus puntos de vida de acuerdo al poder de ataque de la unidad que realiza el ataque, combinado con la protección o armadura de la unidad atacada. Si luego de recibir un ataque, los puntos de vida de la unidad son menores o iguales a cero, la unidad muere. Luego de realizar un ataque, la unidad debe esperar un tiempo dado para poder realizar el siguiente ataque, dicho tiempo está determinado por la velocidad de ataque de la unidad. Adicionalmente, una unidad puede caminar desde su posición actual a una nueva posición deseada en el terreno, para lo cual la unidad calcula la ruta e inicia el recorrido.

El tipo especial de unidad denominada Prócer, tiene la habilidad de lanzar un grito de batalla que incentiva a los combatientes comuneros para que ataquen con más empeño. En el momento en que el prócer decide lanzar el grito, determina si ha pasado un tiempo suficiente desde la última vez que lo hizo para poder volver a lanzarlo. De ser así, las unidades de su ejército que están al ‹alcance del grito› pueden escucharlo e incrementan su poder de ataque durante un lapso de tiempo determinado por la duración del grito.

Con esta información sobre el funcionamiento del sistema y siguiendo la metodología GAIA se pudo determinar que en el juego todos los personajes que intervienen juegan el rol de una *Unidad*. Este rol se detalla en la tabla 1 mediante su *esquema de rol*.

Tabla 1 Modelo de roles. Esquema del rol Unidad

Esquema del Rol:	***Unidad***

Descripción:
Este rol involucra el vigilar el área del terreno donde se encuentra la unidad hasta donde alcanza su rango de visión para saber si hay unidades enemigas y de ser así se acerca hasta donde su rango de ataque sea suficiente para atacar al enemigo y lo hace. En el combate la unidad puede perder todos sus puntos de vida así que puede morir. Desde el momento de su creación, la unidad sabe si puede incentivar a las unidades amigas cercanas haciendo uso del grito comunero, es decir, si la unidad se puede comportar como un prócer.

Protocolos y Actividades:

Actividades:

- **Vigilar**: Consiste en revisar dentro del rango de visión de la unidad si hay unidades enemigas. De ser así, la unidad puede pasar a la actividad de atacar.
- **Caminar**: Esta actividad implica el desplazarse de un lugar a otro dentro del terreno teniendo en cuenta los obstáculos del terreno y la posición de las diferentes unidades dentro de él.
- **Morir**: Cuando una unidad está siendo atacada, pierde progresivamente sus puntos de vida. Si estos puntos se hacen cero o menos, la unidad pasa a la actividad morir, la cual impide que la unidad se mueva, vigile, ataque o incentive a sus tropas.

Protocolos

- **Atacar**: Este protocolo se realiza cuando la unidad ha detectado una unidad enemiga y esta última se encuentra dentro del rango de ataque de la primera. Esta actividad consiste en la disminución de los puntos de vida de la unidad enemiga hasta que alguna de las dos muera.
- **GritoComunero**: Esta actividad es realizada por la unidad cuando tiene la capacidad de ser Prócer y consiste en incrementar la capacidad de ataque propia y la de las unidades amigas cercanas durante un cierto intervalo tiempo.

Esquema del Rol:	*Unidad*
Permisos:	
changes supplied Terreno	// Terreno donde se encuentra la unidad
Losas	// Inf. de una losa (cuadro que hace parte del terreno)
changes supplied Mapa	// Información de obstáculos en el terreno
Mapa[fila][columna]	// Obstáculo en una losa
reads supplied EsProcer	// Indica si la unidad es un prócer
Responsabilidades:	
Liveness:	
Unidad = ((Vigilar.[Caminar].[Atacar].[Morir])||[GritoComunero])^{w}	
Safety:	
puntosVida > 0	// Condición de vida de la unidad
EsProcer = (TRUE | FALSE)	// Indica si la unidad es un prócer

De la tabla 1 se puede resaltar lo siguiente: ***Terreno*** es un recurso que representa la información geográfica del terreno donde esta la unidad. ***Mapa*** es un recurso que representa la información acerca de los obstáculos en el terreno. Esta información es utilizada por las unidades cuando desean desplazarse de un lugar a otro dentro del terreno; por tal motivo esta información se encuentra en cambio constante. Y ***EsProcer*** es un recurso de sólo lectura para la unidad, el cual le indica si puede, cada cierto intervalo de tiempo y a petición del usuario, incentivar a las unidades amigas que se encuentren cercanas a ella emitiendo el grito comunero. El tipo de unidad con esta capacidad es llamada Prócer.

Luego de identificar los roles, era necesario definir cómo iban a interactuar los agentes entre sí dependiendo del rol que cada uno desempeña en el sistema. Esto se logró especificando los protocolos asociados a cada rol mediante el **modelo de interacción**. Para el caso del rol Unidad se definieron los siguientes protocolos: Atacar y GritoComunero.

El Protocolo ATACAR: Este protocolo (tabla 2) se puede visualizar desde las perspectivas de la unidad que ataca y de la unidad que recibe el ataque. Desde el punto de vista de la unidad que ataca, esté protocolo es utilizado cuando la unidad visualiza a una unidad enemiga dentro del espacio abarcado por su rango de visión. Por otro lado, la unidad que recibe el ataque hace uso de este protocolo cuando una unidad enemiga ha completado un ataque en contra de ella; el resultado de esto es la disminución de los puntos de vida de la unidad receptora en una cantidad que depende del valor del ataque, llegando posiblemente a pasar al estado de muerte si sus puntos de vida llegan a ser menores o iguales a cero.

Tabla 2 Definición del protocolo Atacar

Atacar		*UnidadEnemiga*
Unidad	*Unidad*	
Cuando una unidad enemiga se encuentra dentro del rango de ataque de la unidad iniciadora, esta última lanza un ataque contra la primera.		
↓		
RecibirAtaque		
Unidad	*Unidad*	*ValorAtaque*
La unidad que recibe el ataque disminuye sus puntos de vida en un valor que depende del valor del ataque.		*PuntosVida*

El Protocolo GRITOCOMUNERO: Este protocolo (tabla 3) es básicamente una habilidad especial que puede tener una unidad (denominada prócer) y consiste en hacer que las unidades amigas cercanas a ella incrementen su nivel de ataque durante un tiempo determinado.

Tabla 3 Definición del protocolo GritoComunero

LanzarGritoComunero		***El usuario pide al prócer lanzar el grito.***
Unidad	***Unidad***	
Si una unidad es un prócer, la unidad puede hacer uso del grito comunero para incentivar a las unidades amigas cercanas para que ataquen con más ahínco.		***Las unidades cercanas escuchan el grito.***
↓		
EscucharGritoComunero		
Unidad	***Unidad***	***Tiempo***
Cuando el prócer emite el grito comunero, las unidades amigas que estén cerca a él, pueden escucharlo, así que éstas incrementarán sus puntos de ataque durante un tiempo determinado.		***PuntosAtaque***

El siguiente paso buscó determinar lo diferentes tipos de agentes que tiene el sistema. Dado que únicamente se identificó el rol *Unidad*, se definió un solo tipo de agente que se denominó *AgenteUnidad* y que podrían coexistir varias instancias (cuantificador +) de este tipo de agente en un instante dado en tiempo de ejecución (ver el **modelo de agentes** en la figura 3).

A continuación se buscó especificar los servicios (o funcionalidad) que debían implementar cada tipo de agente. Para esto se tuvo en cuenta la información de las actividades, protocolos y propiedades de vida y de seguridad de la etapa de análisis. De esta manera se llegó a la creación del **modelo de servicios** para el tipo de agente *AgenteUnidad* que se presenta en la tabla 4.

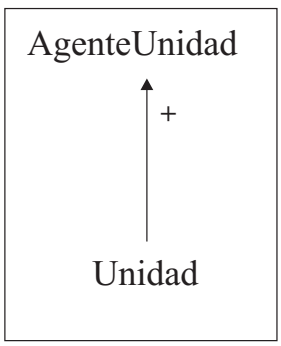

Figura 3 Modelo de agentes

Tabla 4 Modelo de servicio

Servicio	***Entradas***	***Salidas***	***Precondiciones***	***Post-Condiciones***
Vigilar	Mapa	Actividad o Protocolo	PuntosVida > 0	La actividad de la unidad es Vigilar, Atacar o Morir
Caminar	Destino	Posición actualizada	PuntosVida > 0 Λ Camino ≠ nil	La posición debe estar dentro del terreno.
Morir	-	-	PuntosVida <= 0	-
Atacar	UnidadEnemiga	La unidad enemiga recibe el ataque.	PuntosVida > 0 Λ UnidadEnemiga ≠ nil Λ TiempoUltimoAtaque > t	-
Recibir Ataque	ValorAtaque	PuntosVida	PuntosVida > 0	Puntos de vida disminuidos
EmitirGritoComunero	-	Notificación a las unidades amigas cercanas.	PuntosVida > 0 Λ TiempoUltimoGrito > t	No se puede emitir el grito comunero nuevamente hasta que transcurra un tiempo t.
EscucharGritoComunero	Tiempo	PuntosAtaque	PuntosVida > 0	Puntos de ataque incrementados

Para finalizar el diseño se establecieron los enlaces de comunicación que se podían dar entre los distintos tipos de agentes. Debido a que los personajes del juego no son entes aislados y están en continua interacción con los demás, se realizó el **modelo de conocidos** (figura 4) donde se muestra la comunicación entre los agentes del tipo *AgenteUnidad* con otros del mismo tipo.

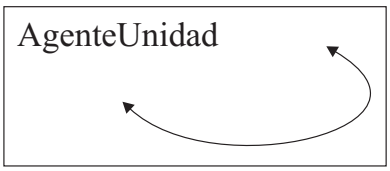

Figura 4 Modelo de conocidos

Terminado el análisis y diseño del comportamiento de los personajes del juego usando GAIA se tuvo claro la existencia de un tipo de agente denominado AgenteUnidad, del cual ya conocíamos su comportamiento, propiedades y modo de interacción. Esto facilitó la posterior integración con el diseño UML de los demás componentes del juego y finalmente su implementación mediante clases en lenguaje C++. Todo fue posible gracias a que la metodología desliga el análisis y el diseño de los agentes, de la plataforma y herramientas usadas para la implementación.

Debido a que en este videojuego el comportamiento de los personajes es relativamente simple, la implementación de los agentes usando C++ fue posible y no tuvo una alta complejidad. Sin embargo, si se implementan características más complejas como comportamiento social y económico de la sociedad dentro del juego, su implementación con C++ puede llegar a volverse compleja y difícil de mantener por lo cual sería conveniente analizar la posibilidad de implementar los agentes usando lenguajes declarativos o combinar el resultado de GAIA con otras técnicas de inteligencia artificial como el scripting.

Principales productos de la investigación

Para el desarrollo de video juegos es preciso tener en cuenta varios conceptos, uno de ellos, el guión, que normalmente es desarrollado por un profesional de la comunicación, las artes o la sociología. Desde la perspectiva técnica, es preciso tener en cuenta los prerrequisitos matemáticos, entre ellos, la teoría de vectores, matrices, rayos y rotación de ejes, los cuales son de gran utilidad para trabajar en espacios tridimensionales. Un concepto básico son las losas, que permiten la conformación del terreno del juego. Teniendo definidas las losas, se debe pensar en cómo se va a hacer para desplazar a una unidad (personaje) desde un punto A hasta un punto B del mapa, y más aún, gracias a las propiedades de las losas, hay que establecer si hay obstáculos que puedan alterar dicho desplazamiento. A este problema se le conoce como la búsqueda de camino. Otro tema importante, consiste en definir la API que se usará para el desarrollo del video juego, entre ellas: DirectX y OpenGL. Teniendo en cuenta que estas APIs no ofrecen un modelo orientado objetos para su uso. Se diseño y desarrollo un motor orientado a objetos como base de la implementación del proyecto. En la figura 5 se puede apreciar una vista general del motor (más detalles en referencia [14]).

El desarrollo de un juego es un trabajo eminentemente interdisciplinario. En este sentido, el trabajo que el diseñador de los personajes realizó, necesitó de una herramienta que permitiera validar si los modelos construidos en un software de terceros, se podía visualizar adecuadamente con las opciones dadas por el motor de juegos. Por lo anterior, se construyó una aplicación para la visualización y prueba de los modelos, la cual se muestra en la figura 6.

Además, como los modelos gráficos desarrollados por el diseñador no sólo correspondían a personajes del juego sino también a objetos del ambiente, como por ejemplo, casas, árboles, iglesias, entre otros, y además que se necesitaba tener claro como estos elementos se organizaban en el entorno del juego se necesitó desarrollar una herramienta software (basada en el motor) que permitiera crear o definir el terreno del juego (figura 7). Logrando con ella ubicar los objetos de ambientación y definir propiedades adicionales a las losas, como por ejemplo el nivel de la superficie (colinas, llanuras, entre otros).

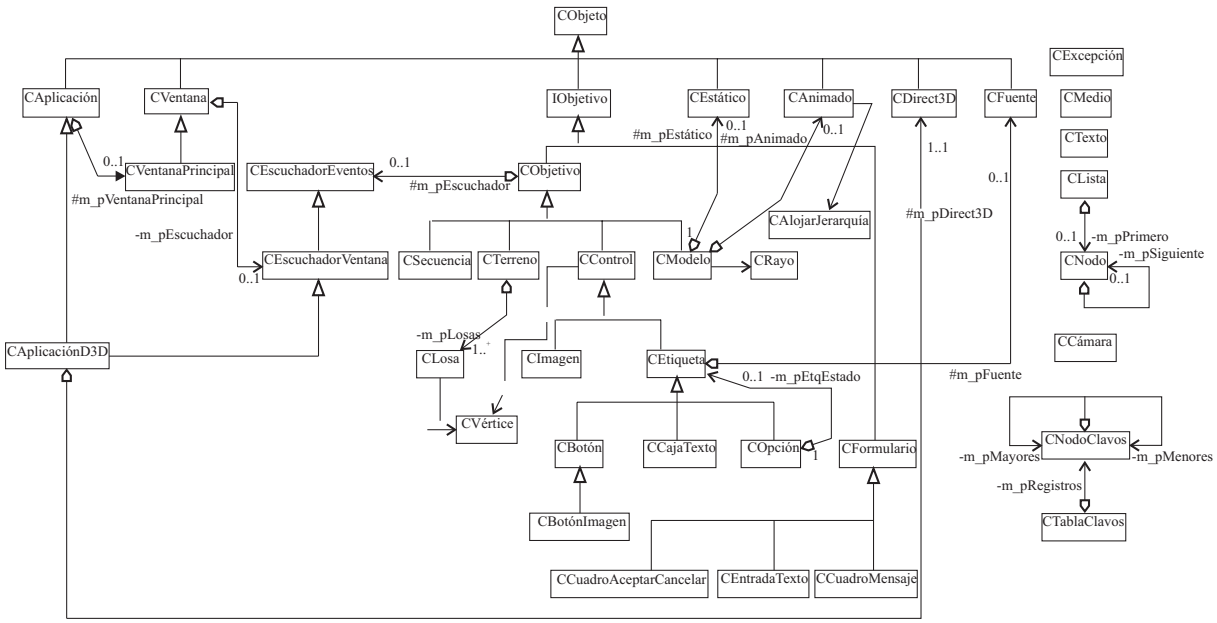

Figura 5 Vista general del motor 3D orientado a objetos

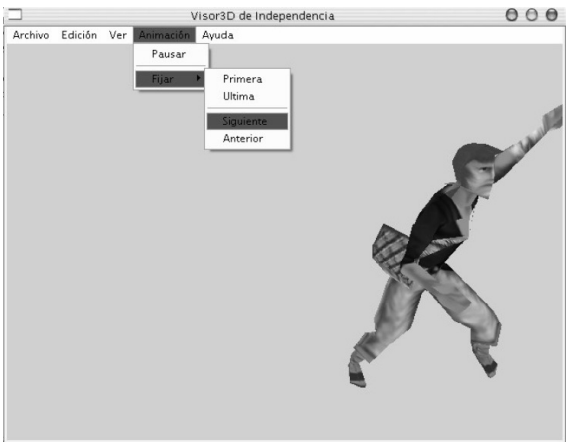

Figura 6 Ventana principal del visor de modelos

Figura 7 Editor de terrenos

Terminada la construcción de modelos (personajes y ambientación) y la definición del terreno, se necesitó solucionar el problema de la búsqueda de caminos. En la literatura se reporta el algoritmo A* (A estrella) como la base de solución de este problema [7], puesto que encuentra el camino más corto entre dos puntos en un grafo construido a partir del terreno y de los elementos que se encuentran en el juego. Con el objetivo de probar el algoritmo se creó una aplicación (figura 8), y se debieron realizar varias optimizaciones del mismo, teniendo en cuenta las colisiones de los distintos personajes sobre el terreno (barreras dinámicas) y además por que este algoritmo sería ejecutado por todos los personajes que se estaban moviendo en el juego en tiempo real. En nuestra implementación de A*, se optimiza la búsqueda del camino restringiendo el máximo numero de losas que se exploran, además si el destino esta ocupado y/o no se puede llegar por que existen obstáculos fijos o dinámicos, el personaje se desplaza hasta la losa libre más cercana al punto

de destino. En la figura 8 se puede apreciar un personaje que desea trasladarse a un punto en el terreno, en este caso no se encuentra un camino posible debido a que el destino se encuentra encerrado totalmente por obstáculos.

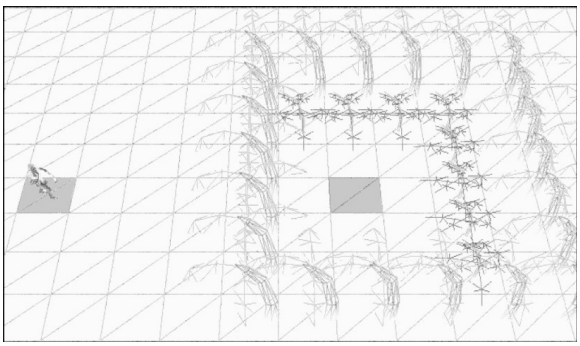

Figura 8 Buscador de caminos

Con todo lo anterior, es decir, diseñado e implementado el motor, resueltos los problemas de comunicación con el diseñador, contando con el guión, el algoritmo de búsqueda de caminos, el terreno, y el análisis y diseño de todos los personajes en el juego (a través de la metodología GAIA) se procedió a la implementación del juego en si mismo, el cual se desarrolló como un juego monousuario donde el jugador puede controlar a cada uno de los personajes usando el dispositivo apuntador (mouse) para desplazarlos por todo el terreno e ir descubriendo en el camino a otros personajes que se unen a la «marcha»; encontrándose en la ruta con enemigos a quienes debe confrontar y le permiten generar una mayor habilidad para enfrentar al comandadote de los conquistadores. A cada personaje, mediante scripting, se le definieron unas características como por ejemplo puntos de vida, velocidad de desplazamiento y destreza en la pelea, las cuales son tenidas en cuenta en la implementación del agente; así por ejemplo el grito comunero del agente Prócer incrementa la destreza en la pelea de los personajes amigos que estén cerca a él. En la figura 9 se muestra una animación del juego, donde Juan Francisco Berbeo (Personaje jugador) y tres campesinos (otros agentes inteligentes) salen del Socorro (población ubicada en Santander - Colombia) y empiezan la marcha de los Comuneros.

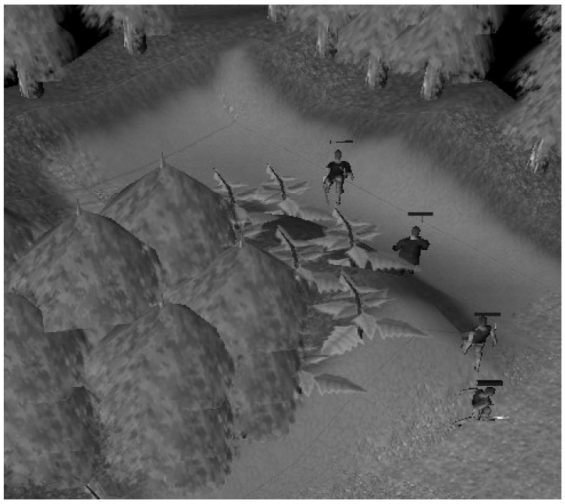

Figura 9 Animación de independencia

Conclusiones y trabajo futuro

Teniendo en cuenta que GAIA es una metodología orientada a desarrollar agentes que colaboran entre sí para alcanzar los objetivos del sistema [15], pensando más en que van a existir varios tipos de agentes cada uno desempeñando un rol y que en este caso de estudio sólo se identificó un agente como tal, es posible que hubiera sido más propicio trabajar con agentes autónomos en lugar de hacerlo con sistemas multiagentes, muy a pesar de los buenos resultados obtenidos. Un agente autónomo es un único agente que incluye temas como sentido, modelado de emociones, motivación, personalidad y acciones de selección y planeamiento, que fue realmente lo que se hizo con el agente AgenteUnidad definido en el juego Independencia. Sin embargo si la idea es que todo el sistema se modele con agentes, es muy probable que no se defina únicamente un agente y en ese caso si se requiera de un sistema multiagente como GAIA, MAS-CommonKADS [16] u otra metodología de este tipo.

Se resalta en el videojuego la utilización de una metodología de agentes para la definición del comportamiento de los personajes del juego. En este sentido se debe considerar que GAIA ayudó a definir los agentes que se utilizarían dentro del juego sin necesidad de entrar en detalles de implementación y que los productos

obtenidos gracias a esta metodología se pudieron diseñar muy fácilmente usando UML. Existió, entonces, una complementariedad entre GAIA, UML y Programación Orientada a Objetos que permitió que la definición e implementación del comportamiento de las unidades del juego estuviera a la altura y vanguardia de grandes títulos de juegos en esta misma categoría.

Este juego muestra la capacidad de los ingenieros Colombianos, en el desarrollo de proyectos software NO tradicionales, y que pueden a futuro ser una base sólida de ingresos para el País.

En cuanto al trabajo futuro en el área, se empezó a explorar un proyecto relacionado con la forma de modelar el comportamiento de personajes de videojuegos, no sólo de estrategia, sino de otros géneros (aventura), usando metodologías diferentes a GAIA, para determinar cuál es la más apropiada y con ello llegar a la definición posterior de un framework, que permita crear agentes usando un lenguaje de programación declarativo y que sean fácilmente incorporados dentro de un videojuego, para que definan el comportamiento de sus personajes. Además se han iniciado proyectos que permitan incorporar efectivamente los juegos en la formación de niños con respecto a la prevención y tratamiento de enfermedades.

Agradecimientos

Este trabajo fue cofinanciado por COLCIENCIAS a través de su proyecto de I+D en ETI con código 1103-14-14897 y la Vicerrectoría de Investigaciones de la Universidad del Cauca.

Referencias

1. P. Sweetser. *Current AI in Games: A review.* School of ITEE, University of Queensland. Australia. 2002. http://www.itee.uq.edu.au/~penny/_papers/Game. Al Review.pdf. Consultada el 6 de noviembre de 2006.

2. P. Baillie-De Byl. *Programming Believable Characters for Computer Games*. Rockland (MA). 2004. pp. 465.

3. N. Muñoz, W. Rivera. *Independencia: Juego de Estrategia en 3D basado en hechos importantes de la Campaña Libertadora de Colombia.* Trabajo de grado. Ingeniería de Sistemas. Universidad del Cauca. Popayán. Colombia. 2005. pp. 104.

4. M. Wooldridge, N. Jennings, D. Kinny. *The Gaia Methodology for Agent-Oriented Analysis and Design.* Ed. Kluwer Academic Publishers. Boston. pp. 1-27.

5. M. Van Lent, J. Laird, J. Buckman, J. Hartford, S. Houchard, K. Steinkraus, R. Tedrake. *Proceedings of the Sixteenth National Conference on Artificial Intelligence.* July 18-22. 1999. Orlando (FL). 1999. pp. 929-930.

6. M. Van Lent, J. Laird. "Developing an Artificial Intelligence Engine". *Proceedings of the game developers Conference*. March 16-18. 1999. San José (CA). pp. 577-588.

7. T. Barron. *Strategy Game Programming with DirectX 9.0*. Wordware Publishing Inc. Plano. Texas. (USA). pp. 538.

8. A. Nareyek. *Intelligent Agents for Computer Games.* http://www.ai-center.com/home/alex/ 2000. Consultada el 12 de febrero 20 de 2008.

9. Empire Earth, Sierra. http://empireearth.sierra.com/es/. Consultada el 22 de febrero de 2008.

10. M. Ponsen. *Improving Adaptative Game AI with Evolutionary Learning.* http://www.kbs.twi.tudelft.nl/docs/MSc/2004/Ponsen_Marc/thesis.pdf. Consultada el 15 de enero de 2008.

11. Civilization, http://www.civ3.com/. Consultada el 22 de febrero de 2008.

12. Balance of power. http://www.balance-of-power.ch/. Consultada el 22 de febrero de 2008.

13. Popopulus http://www.mobygames.com/game/populous. Consultada el 22 de febrero de 2008.

14. N. Muñoz, C. Cobos, W. Rivera. "Motor Gráfico y Multimedia Orientado a Objetos para el Desarrollo de Juegos de Estrategia en 3D". *Revista Enlace Informático.* Vol. 4. 2005. pp. 57-68.

15. Grupo de agentes de software: ingeniería y aplicaciones. *Metodologías de agentes.* http://grasia.fdi.ucm.es/ingenias/Spain/index.php. 2007. Consultada el 1 de diciembre de 2007.

16. C. A. Iglesias. *Definición de una metodología para el desarrollo de sistemas multiagentes*. Tesis doctoral. Departamento de Ingeniería de Sistemas Telemáticos. Universidad Politécnica de Madrid. España. 1998. pp.31-49.