Metodología para la construcción de personajes basados en Agentes Software en videojuegos de aventura 3D

Trabajo de Grado

Diana María Pérez Mera
Luís Alfredo Córdoba Hoyos

Director: Msc. Erwin Meza Vega

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Grupo de I+D en tecnologías de la información
Departamento de Sistemas
Popayán, Julio de 2009

Agradecimientos

Por el apoyo que recibimos durante todo el desarrollo de este trabajo queremos expresar agradecimientos a:

- **Msc. Erwin Meza Vega** Director del trabajo, por su valiosa colaboración durante la realización de este proyecto.

- **Ing. Esp. Jaime Adalberto López V.** Codirector del trabajo, por su apoyo incondicional durante el desarrollo de la investigación y sus valiosos consejos en la realización del videojuego.

- A nuestros docentes quienes nos guiaron durante todo el transcurso de nuestra carrera y aportaron su conocimiento para nuestra formación personal y académica.

Dedicatoria

"Especialmente a mis padres y hermanas por todo su amor, comprensión y apoyo durante todos los momentos de mi vida.
A mi esposo Jaime por brindarme todo su cariño y amor incondicional
A mi hijo por ser el motor de mi vida
A mis amigos por compartir conmigo alegrías y tristezas durante el transcurso de mi carrera."

Diana María Pérez Mera

"A todas aquellas personas que siempre estuvieron ahí, brindándome su apoyo. En especial:
Mi Madre y mi Padre
Mis hermanos y demás familiares
Mis amigos y Compañeros"

Luis Alfredo Córdoba Hoyos

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCIÓN

El juego es una actividad presente en todos los seres humanos que ha sido identificada como un patrón fijo en el comportamiento de las personas [1]. Su diversidad demuestra que cumple con una función primordial a lo largo del ciclo vital de cada individuo. Habitualmente se le asocia con la infancia, pero lo cierto es que se manifiesta a lo largo de toda la vida del hombre, incluso hasta en la ancianidad [2]. Generalmente, el juego es relacionado con diversión, satisfacción y ocio, pero su trascendencia es mucho mayor, ya que a través del juego se transmiten valores, normas de conducta, se resuelven conflictos, se educa a los jóvenes y se desarrollan múltiples facetas de la personalidad [1]. Desde siglos anteriores han existido diferentes teorías psicológicas sobre el juego, por ejemplo en el siglo XIX algunos lo consideraban como el resultado de un exceso de energía acumulada y mediante el juego se gastan las energías sobrantes (Teoría del excedente de energía) [2]. Otros, por el contrario, sostenían que las personas tienden a realizar actividades difíciles y trabajosas que producen fatiga, de las que descansan mediante otras actividades como el juego, que producen relajación (Teoría de la relajación), también se concebía el juego como un modo de ejercitar o practicar los instintos antes de que éstos estén completamente desarrollados. Pero a pesar del cambio en el pensamiento de las personas a través del tiempo, se llegaba a la misma conclusión, el juego consistía en un ejercicio preparatorio para el desarrollo de funciones que son necesarias para la época adulta [1].

Hoy en día los juegos han evolucionado, tal vez buscando la manera de adaptarse a cambios ocurridos en áreas, por ejemplo en la tecnología, brindando de esta manera, nuevas formas de entretenimiento entre las cuales se destacan los “videojuegos”. Un videojuego es un programa informático, creado expresamente para divertir, basado en la interacción entre una persona y un aparato electrónico donde este se ejecuta. Estos recrean entornos virtuales en los cuales el jugador puede controlar a un personaje o cualquier otro elemento de dicho entorno, para conseguir uno o varios objetivos por medio de unas reglas determinadas [3]. Estos son un medio de distracción con una gran popularidad, tanto así, que parte de las generaciones actuales buscan, durante su tiempo libre, distraerse en algún videojuego experimentando un mundo virtual y enfrentando experiencias y retos que normalmente no se presentan en la vida diaria [4].

Un estudio realizado por la universidad de Indiana [5], concluye que los videojuegos son considerados una mala influencia para aquellas personas que los juegan por contener escenas de violencia y por la adicción que generan. Pero una investigación realizada por el Observatorio del Videojuego de la Universidad Europea de Madrid (UEM) [6] señala que esta forma de entretenimiento no es tan negativa, pues aumenta la sociabilidad y fomenta el desarrollo de “habilidades directivas”. Por otro lado, los videojuegos ayudan a despertar habilidades como la capacidad de superación, destreza y sobre todo trabajo en equipo y son muy buenos para la formación especialmente de los jóvenes [7][3].

Teniendo en cuenta lo anterior se han creado diferentes tipos de videojuegos, cada uno de estos, ayuda a que el jugador desarrolle ciertas habilidades dependiendo de cuál es su tipo de videojuego favorito. Actualmente los videojuegos de aventura son los de mayor preferencia [8] debido a que estos poseen diferentes características que llaman la atención de los jugadores, tales como argumentos bien elaborados, buenos efectos visuales y sonoros, permitiendo al jugador formar parte de las historias recreadas y por último, estos contienen un alto nivel de inteligencia artificial [9]. Los personajes que protagonizan los videojuegos de aventura deben ser atractivos al jugador ya que gran parte del tiempo de juego se basa en la interacción de estos con el usuario y con otros personajes mediante acciones y diálogos, se les debe asignar un papel que permita fijarle una identidad, también se debe saber que acciones va a realizar para poder darle una personalidad que justifique sus actos, si va a ser el héroe, la princesa, el villano, el tutor del héroe o el escudero (personajes propios de un juego de aventura). También se les debe asignar características como nombre, imagen, localización, relación con los demás personajes y definir los comportamientos con respecto a su entorno, dotando al personaje de acciones como la proactividad, reactividad y características sociales [10]. Las características mencionadas son de gran importancia dentro del videojuego, porque es a partir de la variedad y creatividad con que se construyen los personajes, que la historia del juego obtiene su riqueza y valor argumental.

En los juegos los personajes pueden actuar de manera individual como lo hacen los personajes llamados PNJ (personajes no jugables), que son controlados automáticamente por técnicas como inteligencia artificial. En la actualidad, estos son controlados mediante máquinas de estado finito o scripts para que puedan interactuar en un entorno artificial dentro de los videojuegos [11]. Estas técnicas entregan a los personajes un número determinado de acciones y reacciones que tienen que ser previamente conocidas y programadas por el desarrollador, esto hace que el juego tenga poca *jugabilidad* porque una vez que el jugador ha explorado todas las alternativas pre-programadas, el juego deja de ser útil y por lo tanto entretenido [11].

Los personajes de los juegos también pueden trabajar de manera grupal o cooperativa, centrando la atención en cómo construir juegos basados en grupos de personajes que se interrelacionen entre sí, brindando mayor dinamismo al juego. En este caso, los personajes pueden tener influencia unos con otros, no solo de manera explícita, sino también por la actuación sobre el entorno, obligando a estudiar el entorno con detalle para detectar qué acciones realizadas por un personaje pueden afectar a otro. Esto aumenta la complejidad del desarrollo, debido a que no existe ninguna metodología o modelo que permita un desarrollo grupal de personajes que se relacionan entre sí (solo existen técnicas para crear personajes individuales) [12]. En este sentido existen técnicas de inteligencia artificial que no han sido muy exploradas dentro del desarrollo de los personajes de un videojuego, una de estas técnicas son los agentes software los cuales se sitúan en un entorno, el cual perciben y modifican a través del tiempo, mientras persiguen sus propios objetivos basados en su experiencia y/o aprendizaje [11].

Todas las propiedades que brindan los agentes (autonomía, reactividad, iniciativa, habilidad social, etc.) serían de gran ayuda en la construcción de personajes en un videojuego porque, no solo permiten modelar comportamientos sino que también ayudan a desarrollar las características sociales que deben tener los personajes, facilitando la comunicación de estos con su entorno [13], de esta manera surge la pregunta de investigación: ¿Cómo se puede facilitar la creación de personajes en videojuegos de aventura 3D, basados en agentes software?

Para contestar esta pregunta y ofrecer una alternativa de solución a los problemas mencionados, se desarrolló el presente trabajo a través de la ejecución de los objetivos:

OBJETIVO GENERAL

Proponer una metodología basada en las metodologías de diseño de agentes software, para la construcción de personajes en videojuegos de aventura 3D

OBJETIVOS ESPECÍFICOS

1. Adaptar una metodología de diseño de agentes que permita incorporar las características que poseen los agentes en la construcción de videojuegos de aventura 3D.

2. Construir un API[1] basada en la metodología como un módulo software sobre una librería gráfica o un motor de juegos.

3. Validar la metodología mediante la construcción de un nivel de juego para computador que cumpla con los siguientes requisitos:

3.1 Utilice la implementación de la metodología propuesta (agentes-personajes).

3.2 La historia debe encontrarse basada en el tratamiento de la tuberculosis pulmonar.

3.3 Emplee un escenario acorde a la historia del juego con ambientación, como por ejemplo: edificaciones, personajes genéricos 3D, ayuda contextual y manejo de terrenos, entre otros.

3.4 Sea elaborado en modo “single player”, es decir, que lo pueda jugar un único jugador a la vez.

A lo largo del presente documento se abordan los conceptos teóricos y prácticos que se necesitaron para llevar a cabo el proyecto. El documento se encuentra organizado en diez partes como se explica a continuación:

Introducción
Este capítulo da una visión general del trabajo desarrollado, los objetivos que cumple y una ubicación del contexto en general.

[1] Application Programming Interface (Interfaz de Aplicación Programable)

Capítulo I: Marco Teórico
Este capítulo contiene las bases teóricas sobre las cuales se encuentra enmarcado el proyecto, ellas contemplan los videojuegos, agentes software, motores de juegos y la tuberculosis.

Capítulo II: Metodología para la construcción de personajes
Este capítulo se presenta el modelo desarrollado, el cual contiene la utilización de los agentes software para crear personajes de videojuegos de aventura y de esta manera facilitar el diseño de los mismos.

Capítulo III: Aplicación de la metodología para la construcción de personajes en un videojuego de aventura en 3D
Este capítulo contiene el desarrollo y documentación de las actividades definidas en las etapas de Conceptualización, análisis, modelo de diseño, diseño, codificación y prueba de cada agente, integración y operación y mantenimiento, además contiene una descripción de los problemas y las soluciones que se presentaron en el desarrollo del proyecto.

Capítulo IV: Descripción del API para implementación de agentes/personajes
Este capítulo contiene el desarrollo y documentación de una interfaz de aplicación programable para dotar de las características propias de los agentes a los personajes de un juego de aventura 3D.

Capítulo V: Resultados Obtenidos
Este capítulo presenta los productos y principales aportes obtenidos a partir de la ejecución del presente proyecto.

Capítulo VI: Conclusiones, Recomendaciones y Perspectivas
Esta parte describe las conclusiones a las cuales se llegará una vez culminado el desarrollo del proyecto. También contiene las respectivas recomendaciones para futuros trabajos en esta temática.

Bibliografía
Este ítem indica la bibliografía de los documentos utilizados.

Anexo A
Se compone de un grupo de conceptos relevantes para compresión de este proyecto.

Anexo B
Se profundiza en la evaluación sobre las diferentes metodologías de agentes a partir de los criterios propuestos en este documento. Tomando como marco referencial la bibliografía revisada de estas metodologías.

Anexo C
Contiene la aplicación completa de la metodología, en la construcción del Video Juego “Simón y la tuberculosis – Nivel 2”, con una descripción completa de los personajes.

Articulo

Ensayo construido a partir de los conceptos manejados en el presente documento, enfocado en una descripción general de la metodología propuesta.

CAPITULO I
MARCO TEORICO

Este capítulo contiene las bases teóricas sobre las cuales se encuentra enmarcado el proyecto, ellas contemplan los videojuegos, agentes software, motores de juegos y la tuberculosis.

1.1. Videojuegos

Los videojuegos son un medio de entretenimiento, el cual se ha abierto camino en el mercado desde finales de los años 70 convirtiéndose en una de las más grandes y solventes industrias en el mundo del entretenimiento que ha crecido a la par con la evolución del software y los avances tecnológicos en materia del hardware[14]. Hoy en día existe una gran variedad de videojuegos, por lo que se ha visto la necesidad de crear una forma de clasificación para éstos de manera que pudieran ser agrupados a partir de distintos ejes como pueden ser: la temática, el argumento, la plataforma, entre otros [15]. Un género importante en el mundo de los videojuegos es el de aventura, ya que este puede ser visto como otro tipo de medio narrativo como los libros o las películas en los cuales el jugador adopta el papel de protagonista de una historia, viéndose obligado a sumergirse en el argumento de ésta, para lograr alcanzar los objetivos del juego [16]. En los juegos de aventura la historia forma parte de los elementos más importantes; sin embargo, para los filósofos Carl Jung y Sigmund Freud no podría existir una historia sino existen los personajes, ya que una historia es una narración de los sucesos que ocurren a un determinado personaje en la búsqueda de una determinada solución, simplificando un poco la definición [17]. Es por esto que para la creación de juegos de aventura debe existir un cuidado especial al crear los personajes sin dejar de lado los demás elementos indispensables para la creación de un videojuego como el diseño de los ambientes, la visualización de terrenos, los efectos de sonidos, el diseño de niveles entre otros. Con respecto al área de la programación es importante tener en cuenta el motor de juegos, considerado como el suporte del juego, cuando se hace referencia al tema de la codificación [18]. En un motor de juegos es posible encontrar una gran cantidad de código que puede ser reutilizado facilitando la definición de sonidos, gráficos y otras características del juego.

1.1.1. Desarrollo de videojuegos

En los primeros años de la historia de los videojuegos, casi todo su desarrollo era realizado por una sola persona, la cual se encargaba de realizar todos los procedimientos necesarios para su construcción. Esta persona era la encargada de generar la historia, las imágenes, la programación, la música y todas las demás tareas que se deben realizar para construir un videojuego. A través de años de experimentación y avances tecnológicos, el desarrollo de videojuegos ha evolucionado hacia el desarrollo basado en equipos de trabajo [19]. En la actualidad, los equipos de desarrollo de videojuegos cuentan con personal especializado en diferentes disciplinas, los cuales se encargan de realizar tareas específicas en el proceso de la elaboración del juego, entre los cuales se encuentran [19]:

- **Guionistas:** Los guionistas son los responsables de escribir la historia del juego, los desafíos a enfrentar y los misterios a resolver. Por lo general, ayudan a definir el origen del juego, como los personajes del juego, los diálogos, y el nivel de división [19].

- **Diseñadores de Nivel:** Son los encargados de crear y usar las herramientas necesarias para definir cada uno de los niveles del juego, de acuerdo con las premisas definidas por el equipo de programación y la historia escrita por los guionistas [19].

- **Artistas:** "Artistas" es una categoría amplia, que para el caso de los videojuegos abarca: los creadores del arte conceptual, los creadores del arte computacional, las personas responsables de la creación de las texturas para los modelos 3D, las personas encargadas de la definición del color, entre otros. Estas personas son las encargadas de crear los pantallazos de apertura, las imágenes del menú, además pueden encargarse de la realización del arte para el mercadeo del videojuego [19].

- **Modeladores:** Estas personas tienen a cargo la responsabilidad de crear los modelos 3D para el videojuego siguiendo los conceptos y el arte computacional [19].

- **Animadores:** Crear un modelo 3D es diferente que animarlo, es por esta razón que muchos equipo de desarrollo cuentan con personal especializado en este campo. Los cuales también se encargan de crear las escenas y secuencias de video que se muestran al iniciar y en los puntos esenciales del juego, como cuando el jugador gana un desafío, o al principio o al final de un nivel [19].

- **Músicos:** Esta es también una categoría amplia que comprende desde las personas encargadas de la reproducción de los sonidos de fondo, los efectos ambientales y las voces de los diferentes personajes del videojuego [19].

- **Programadores:** Son las personas encargadas de la redacción del código del juego, incluyendo todos los cálculos físicos y matemáticos necesarios para satisfacer los efectos deseados para el videojuego. La metodología planteada en este proyecto pretende definir un conjunto de procedimientos, que permitan facilitar sus funciones a través de la organización de la información relevante para esta categoría en cuanto a la construcción del juego [19].

Todas estas personas conforman el grupo de desarrollo de un videojuego. Aunque en ocasiones, una sola persona debe asumir varios roles, dependiendo de la complejidad del juego y de los recursos con los que se cuente. No obstante, es necesario que el equipo de desarrollo realice una serie de procesos o tareas en un orden establecido, ya que algunos de los miembros del equipo necesitan de los resultados de los otros miembros, para poder desempeñar con éxito sus funciones [23].

1.1.2. Tipos de videojuegos

En la actualidad existe una gran variedad de videojuegos, dentro de los cuales muchos comparten características fundamentalmente relacionadas con la temática principal, el argumento, la estructura del juego, la plataforma, las habilidades lúdicas perseguidas [24]. Los videojuegos que combinan estas características reciben el nombre de híbridos. En el Anexo A de este proyecto se presenta una clasificación de los videojuegos en base a una propuesta presentada por el profesor Pere Marques Graells de la Universidad Autónoma de Barcelona, en la cual se identifican seis tipos de videojuegos [29]: Arcades - Aventura y Rol - Simuladores - Estrategia – Deportes - Puzzles y preguntas.

1.1.3. Personajes

Autores como William Archer opinan [31] que la creación de personajes no puede ser regulada por recomendaciones teóricas. A pesar de que en la actualidad es común encontrar una gran cantidad de manuales y guías para escribir historias, la creatividad no debe ser atada a un conjunto de reglas o pasos. Y no es lo que pretende este proyecto, dado que la idea es dar un enfoque "mostrar un camino a seguir" lo que no quiere decir que esta sea la única forma, es solo una opción para crear personajes en un videojuego de aventura [31].

Cuando se habla de personajes en tres dimensiones (3D), y generalmente se asocia este concepto exclusivamente a la parte física del personaje en lo que se refiere a la altura, anchura y profundidad. El término de tres dimensiones también es empleado por algunos autores para ofrecer una descripción detallada del personaje determinando su carácter físico, psicológico y sociológico, esta definición es aplicada a todos los personajes de un juego ya sea o no el personaje jugador [17] (la descripción de los personajes de un videojuego es ampliada en el Anexo A).

1.2. Agentes Software

1.2.1. ¿Qué es un Agente?

Aunque no hay una definición unificada en cuanto a qué es un agente, un intento de unificar los esfuerzos para el desarrollo de esta tecnología puede encontrarse en **FIPA** (*Foundation for Intelligent Physical Agents*) [46] donde se definen los agentes como entidades de software dotadas de un grupo de propiedades, entre las que se destacan: el ser capaz de actuar autónomamente en un ambiente, comunicarse directamente con otros agentes, estar condicionado por un conjunto de tendencias u objetivos, manejar recursos propios, ser capaz de percibir su ambiente y tomar de él una representación parcial, además, ser una entidad que posee habilidad y ofrece servicios, que puede reproducirse, etc. [46] De forma general, varios autores reconocen en los agentes diversas propiedades, entre las que se destacan el ser autónomos, reactivos, pro-activos y tener habilidad social [38] (la descripción de los agentes se amplía en el Anexo A).

1.2.2. Propiedades de los Agentes

A lo largo de la construcción de la teoría de agentes diferentes autores han propuesto diferentes características para los agentes software, entre las cuales se encuentran [29]:

- **Inteligencia:** Es difícil de definir con precisión. Cuando se aplica a los agentes, se podría definir la inteligencia como algunas capacidades de alto nivel, es decir, la capacidad del agente para razonar sobre su conocimiento, aprender e inferir nuevos conocimientos y planificar las acciones consecuentes para alcanzar sus objetivos. En este caso, la inteligencia tiene el mismo sentido que el término “inteligencia artificial” [49] [29].

- **Autonomía:** Existen diferentes formas de entender el término “autonomía”. En el sentido más común (absoluto), la autonomía se refiere a lo impredecible del comportamiento del agente: cuanto más impredecible sea, más autonomía aparecerá [29]. Sin embargo, la autonomía absoluta no se aplica muy bien a los agentes, ya que de acuerdo con la mayoría de definiciones, un agente debe servir para algunos propósitos que lo limitan. Un tipo más útil de autonomía es la “autonomía social”, que se refiere a la autonomía del agente cuando es analizado en una sociedad de agentes. No obstante, las relaciones sociales más comunes como la cooperación, coordinación y el compromiso, limitan al agente. Finalmente, la ejecución de la autonomía se refiere a la capacidad del agente para elegir y llevar a cabo la acción adecuada en el momento oportuno (técnicas de planificación). De hecho, al diseñar un agente, se tiene que encontrar un fino equilibrio entre autonomía y servidumbre. Un nivel inferior puede consistir en un agente que represente a un usuario, como el bien conocido PDA (Asistente Personal Digital) [49] [29].

- **Racionalidad:** Se aplica esencialmente a los agentes autónomos e inteligentes. Los agentes racionales tienden a elegir y realizar acciones que maximicen su esperada utilidad en función de sus percepciones y su propio conocimiento. La racionalidad también implica que la acción elegida sea consistente con los deseos y creencias del agente [49] [29].

- **Movilidad:** Es la capacidad que tiene un agente para iniciar su ejecución, y desplazarse a otro entorno (llevando consigo datos y códigos) para continuar su ejecución [29]. El término "movilidad" dividió a la comunidad de agentes en dos escuelas: la de los que argumentan que la movilidad no es un aspecto esencial (la comunidad de multi-agentes DAI, esencialmente académicos) y la de los que afirman que los agentes móviles son el futuro de los agentes (comunidad de programadores orientados a objetos) [29]. La programación basada en agentes móviles genera un nuevo paradigma que realza la flexibilidad y eficacia del diseño y ejecución de las aplicaciones distribuidas, y reduce el ancho de banda de red requerido [49].

- **Comunicación:** Es un aspecto crucial para un agente, que por definición tiene que comunicarse durante su vida con la entidad en nombre de la que actúa y con los otros agentes con los cuales necesita colaborar [29]. El nivel de interacción/comunicación depende en gran medida del nivel de conocimiento del agente, que se desenvuelve entre "datos", "información" y "conocimiento". Los datos se corresponden con el nivel más básico, que puede ser el contenido de una variable, un archivo, etc. La información está mucho más estructurada y puede consistir en la descripción de parte de un equipo de telecomunicación, un documento XML (eXtensible Markup Language) electrónico, o un correo electrónico (con datos asociados) [29]. El conocimiento consiste en información estructurada y reglas lógicas. Se pueden asociar con cada nivel de información los lenguajes de comunicación y protocolos adecuados. Por ejemplo, un agente puede usar FTP (Protocolo de Transferencia de Archivos) para enviar un archivo, o llamar al método de otro agente para obtener o transmitir información. Igualmente, el HTTP (HyperText Transfer Protocol) se puede utilizar para transmitir descripciones XML [49].

- **Sociabilidad:** los agentes son capaces de interaccionar con otros agentes (humanos o no) a través de un lenguaje de comunicación entre agentes [29].

- **Reactividad:** los agentes son capaces de percibir estímulos de su entorno y reaccionar a dichos estímulos [29].

- **Proactividad, iniciativa:** los agentes no sólo son entidades que reaccionan a un estímulo, también cuentan con un carácter emprendedor, que les brinda la posibilidad de actuar guiados por sus objetivos [29].

- **Veracidad:** asunción de que un agente no comunica información falsa a propósito [29].

- **Benevolencia:** asunción de que un agente está dispuesto a ayudar a otros agentes, si esto no entra en conflicto con sus propios objetivos [29].

1.2.3. Metodologías de Agentes

Las técnicas convencionales de ingeniería de software (Proceso Unificado), no tienen en cuenta las necesidades de especificación de los Sistemas Multi-agente, como: la planificación de tareas, intercambio de información con lenguajes de comunicación orientados a agentes, movilidad del código o motivación de los componentes del sistema [46]. Por esta razón se formularon nuevas metodologías basadas en agentes, tales como MaSe, Zeus, INGENIAS, MAS-CommonKADS, GAIA, entre otras. Estas metodologías parten de un modelo informal, en la mayoría de casos de cómo debe ser un Sistema Multi-agente y proponen guías para su construcción. Las primeras metodologías, consistían en una lista breve de pasos a seguir. En la actualidad, aunque es posible encontrar varios progresos en el intento de integrar metodologías orientadas a agentes con la ingeniería del software clásica, aún no se alcanza la madurez que se puede encontrar en metodologías convencionales como el Proceso Unificado [46]. Algunas de las metodologías más usadas actualmente son:

- **MAS-CommonKADS:** Esta metodología extiende CommonKADS aplicando ideas de metodologías orientadas a objetos, en la producción de Sistemas Multi-agentes. La metodología CommonKADS está construida en función a su modelo de experiencia y está pensada para desarrollar sistemas expertos que interactúen con el usuario [36]. De hecho considera sólo dos agentes básicos: el usuario y el sistema. MAS-CommonKADS extiende los modelos de CommonKADS, para tener en cuenta la posibilidad de que dos o más componentes del sistema interactúen [36].

- **MaSE:** (Multi-agent systems Software Engineering) parte del paradigma orientado a objetos y asume que un agente es una especialización de un objeto [39]. La especialización consiste en que los agentes se coordinan unos con otros vía conversaciones y actúan proactivamente para alcanzar metas individuales y del sistema. En MaSE los agentes son una abstracción conveniente, que puede o no poseer inteligencia [21]. En este sentido, los componentes inteligentes y no inteligentes se gestionan igualmente dentro del mismo armazón.

- **ZEUS:** Consta de una herramienta y una metodología de forma similar a MaSE. Desde su aparición, ZEUS se ha convertido en referencia de cómo debe ser una herramienta para el desarrollo de Sistemas Multi-agentes [34]. Sobre todo, por la forma en que combinan los distintos resultados de investigación en agentes (planificación, ontologías, asignación de responsabilidades, relaciones sociales entre agentes) en un sistema ya definido. De hecho, la aplicación genera incluso programas para arrancar el sistema especificado e incluye herramientas de monitorización como el Visor de Sociedad que muestra los agentes existentes y sus relaciones, la herramienta de control para ver o modificar remotamente el estado de los agentes y los Generadores de Informes para obtener estadísticas de funcionamiento e informes de actuación de la sociedad de agentes [35].

- **GAIA:** Es una metodología para el diseño de sistemas basados en agentes cuyo objetivo es obtener un sistema que maximice alguna medida de calidad global. GAIA pretende ayudar al analista a ir sistemáticamente desde unos requisitos iniciales a un diseño que, según los autores, esté lo suficientemente detallado como para ser implementado directamente [50].

- **INGENIAS:** Está basada de la propuesta metodológica de MESSAGE, que ha tenido un gran impacto en la comunidad dedicada al estudio de los agente software. El motivo de presentar INGENIAS en lugar de MESSAGE, es que sus autores consideran INGENIAS como la evolución de las ideas de MESSAGE. INGENIAS profundiza en los elementos mostrados en el método de especificación, en el proceso de desarrollo, además de incorporar nuevas herramientas de soporte y ejemplos de desarrollo [51].

1.3. Técnicas de inteligencia Artificial

A continuación, se describen algunas de las técnicas de Inteligencia Artificial que son usadas con frecuencia en la programación de videojuegos.

1.3.1. Máquinas de Estado Finito (FSM)

Las Máquinas de Estado Finito también conocidas como Autómatas Finitos Deterministas, dependiendo del autor al que se haga referencia [52]. Podrían ser definidas como un formalismo que consiste en dos conjuntos:

- Un primer conjunto que representa los escenarios o configuraciones dentro de los que se puede sumergir la IA.

- Un segundo conjunto de transacciones que son las condiciones que conectan los estados en forma dirigida.

Básicamente, las máquinas de estado finito representan el cerebro de la entidad como un grupo de posibles acciones (estados) y la forma de cambiar de una acción a otra.

Las máquinas de estado finitos son una de las técnicas más utilizadas para la creación de juegos con Inteligencia Artificial, ya que estas pueden ser vistas como el cerebro de una entidad virtual, que describe diferentes acciones usando estados y encadenando ésos con transacciones de manera significativa [52]. Muchos videojuegos utilizan esta técnica desde hace mucho tiempo debido a que son intuitivos, fáciles de codificar, de buen desempeño y pueden representar una amplia gama de comportamientos.

1.3.2. Lógica Difusa

Lógica difusa se refiere a un tipo de lógica que no se centra en reconocer valores verdaderos y falsos. Más bien, utiliza expresiones que no son totalmente ciertas ni totalmente falsas. Esta lógica se aplica a conceptos que pueden tomar un valor cualquiera de veracidad dentro de un conjunto de valores que oscilan entre dos valores (la verdad absoluta o la falsedad total) [52]. Cuando se hace referencia del término difuso se hace referencia al objeto que se está estudiando, el cual expresa la falta de definición del concepto al cual se aplica. La lógica difusa permite tratar con información imprecisa, como el peso medio o temperatura alta, en términos de conjuntos borrosos que se combinan en reglas para definir acciones: por ejemplo "si la temperatura es muy alta, enfriar mucho" [52]. De esta manera, los sistemas basados en lógica difusa combinan variables de entrada definidas en términos de conjuntos difusos, por medio de grupos de reglas que producen uno o varios valores de salida.

1.3.3. Algoritmos Genéticos

Un algoritmo genético es una técnica de programación donde son apropiados los comportamientos de la evolución biológica y utilizados como estrategia para resolución de problemas. Dado un problema específico a resolver, la entrada del algoritmo genético es un conjunto de soluciones potenciales a ese problema, codificadas de alguna manera, y una métrica llamada función de aptitud que permite evaluar cuantitativamente a cada (solución) candidata [52]. Estas candidatas pueden ser soluciones que ya se sabe que funcionan, con el objetivo de que el algoritmo genético las mejore, pero se suelen generar aleatoriamente. Luego el algoritmo genético evalúa cada candidata de acuerdo con la función de aptitud. En un acervo de candidatas generadas aleatoriamente, por supuesto, la mayoría no funcionarán en absoluto, y serán eliminadas. Sin embargo, por puro azar, unas pocas pueden ser prometedoras y mostrar actividad, aunque sólo sea una actividad débil e imperfecta, hacia la solución del problema [52].

Las candidatas prometedoras son conservadas y les es permitido reproducirse. Se realizan múltiples copias de ellas, pero las copias no son perfectas; se introducen cambios aleatorios durante el proceso de copia. Luego, esta descendencia digital prosigue con la siguiente generación, formando un nuevo acervo de soluciones candidatas, y son sometidas a una ronda de evaluación de aptitud. Las candidatas que han empeorado o no han mejorado con los cambios en su código son eliminadas nuevamente; pero, por puro azar, las variaciones aleatorias introducidas en la población pueden haber mejorado a algunos individuos, convirtiéndolos en mejores soluciones del problema, más completas o más eficientes. Se repite el proceso de selección y copian estos individuos vencedores hacia la siguiente generación con cambios aleatorios, este procesos es repetido hasta encontrar la solución adecuada. Las expectativas son que la aptitud media de la población se incrementará en cada ronda y, por tanto, repitiendo este proceso cientos o miles de veces, pueden descubrirse mejores soluciones al problema [52].

Un buen uso de esta tecnología en los juegos es para simular entornos artificiales. En lugar de mantenimiento de los mismos elementos del medio ambiente una y otra vez, se puede crear elementos (como pequeños programas) para evolucionar más fuertes, más inteligentes, más rápidos y elementos (u objetos) que puede interactuar con el medio ambiente y el jugador [52].
El uso de redes neuronales y algoritmos genéticos es muy común en entornos de aprendizaje, y los árboles de decisión y los autómatas son utilizados para definir el comportamiento de los personajes, pero esto no significa que es necesario utilizarlo en todos los juegos que se creen. Además, es importante tener en cuenta que estos no son los únicos campos de la inteligencia artificial, existen varios más, muchos no se mencionaron debido al enfoque de este proyecto, lo que no les resta importancia.

1.4. Motores de Juegos

Un motor de juegos es una Interfaz de Programación de Aplicaciones (Application Program Interface, API) o librería de funciones, que se encarga, entre muchas otras cosas, del manejo de los gráficos, de interpretar las acciones del usuario en el teclado, mouse o joystick, comunicándole al juego que el jugador desea que su personaje salte, camine, etc [18]. En otras palabras, el motor de juegos es la tecnología que mueve al juego [18]. Esta es la razón por la cual el desarrollo de un juego debe estar soportado en la selección de un buen motor de juegos, que considere el aspecto gráfico, la comunicación humano-computador, la dinámica y la flexibilidad del juego en tiempo real.

A continuación, se describen los elementos básicos comunes de un Motor de Juegos [18]:

- **Motor de Renderizado y Animación:** Proporciona funciones de renderizado (2D - 3D – Sprites 3D), para realizar esto se apoya en librerías y API gráficas como OpenGL y DirectGraphics. Este motor se encarga de la definición de una capa de abstracción entre Hardware, Software y el juego en sí, además de la visibilidad, el mapeado, la gestión de mallas 3D, etc. Todas estas funciones son las que permiten tener la impresión de ver imágenes en 3D en la pantalla (en dos dimensiones) de un monitor, que no está diseñado para proyectar imágenes en 3D.

- **Grafo de Escena:** Esta parte es muy importante para el motor de juegos, y se basa en una estructura de datos utilizada para el desarrollo de gráficos basados en vectores, además de la construcción de videojuegos. Es la encargada de ordenar la representación lógica y espacial de una escena gráfica.

- **Motor de Física:** Este software es el encargado de simular los diferentes modelos de física newtoniana, a través del uso de variables del tipo velocidad, masa, etc. Su objetivo principal consiste en simular y predecir los efectos bajo diversas situaciones de lo que ocurre en la vida real o en un mundo de fantasía.

- **Motor de Sonidos**: Es el encargado de la reproducción de la banda sonora, los efectos de sonido (disparos, explosiones, voces de los personajes, etc.), en forma sincrónica con las acciones dentro del videojuego.

- **Gestión de Redes:** Permiten a los juegos utilizar la componente de red como: los juegos en línea, los juegos multi-jugador, etc. Apoyado en los diferentes protocolos de comunicación.

De los componentes mencionados, el motor de renderizado y el motor escena son los más comunes en los motores de juegos, aunque existe la posibilidad de algunos motores no cuenten con algunos de los elementos mencionados [18]. Es por esta razón, que resulta indispensable tener claras las necesidades del videojuego, al momento de escoger el motor con el que se va a trabajar. Teniendo en cuenta que al utilizar un motor que vincule la mayor cantidad de elementos necesarios para el desarrollo de un videojuego, se economiza significativamente el tiempo de implementación, ya que no es necesario reinvertir esfuerzos en el desarrollo de partes que ya se encuentran implementadas y probadas [18]. Aunque usar un motor resultar una limitante ya que solo se puede trabajar con las estructuras definidas en el mismo [18].

CAPITULO II
METODOLOGÍA PARA LA CONSTRUCCIÓN DE PERSONAJES

En este capítulo es presentado un compendio de información, a partir del cual se plantea la propuesta metodológica para la construcción de personajes en videojuegos de aventura 3D. Para ello, es necesario definir qué procesos manejan actualmente las metodologías para la construcción de videojuegos. También son analizadas algunas de las metodologías de sistemas multi-agentes, con el fin de encontrar una que facilite la construcción de videojuegos. A partir de la información analizada, se propone una metodología que reúne los pasos esenciales al construir un videojuego de aventura en 3D tomando como base las metodologías de sistemas multi-agentes.

2.1. Definición.

Según el diccionario de la Real Academia de la Lengua Española el termino **metodología** se encuentra definido como: "Método o procedimiento que se usa en una investigación científica o en una exposición doctrinal". Sin embargo, existen autores que han apropiado este término dotándolo de las características específicas de las ramas de la ciencia en que se desempeñan, como es el caso de Rumbaugh, quien desde una perspectiva orientada al desarrollo de software, define metodología como [19]: *"Una metodología de ingeniería software es un proceso para la producción organizada del software, empleando una colección de técnicas predefinidas y convenciones en las notaciones. Una metodología se presenta normalmente como una serie de pasos, con técnicas y notaciones asociadas a cada paso. Los pasos de la producción del software se organizan normalmente en un ciclo de vida consistente en varias fases de desarrollo"* [19].
De acuerdo con Pressman [20], las principales actividades de una metodología son:

• La definición y descripción del problema que se desea resolver.

• El diseño y la descripción de una solución que se ajuste a las necesidades del usuario.

• La construcción de la solución.

• La prueba de la solución implementada.

Entre los requisitos que debe cumplir una metodología se pueden citar los siguientes [21]:

- La metodología está *documentada*: el procedimiento de uso de la metodología está contenido en un documento o manual de usuario.
- La metodología es *repetible*: cada aplicación de la metodología es la misma.
- La metodología es *enseñable*: los procedimientos descritos tienen un nivel suficientemente detallado y existen ejemplos para que personal cualificado pueda ser instruido en la metodología.
- La metodología está basada en *técnicas probadas*: la metodología implementa procedimientos fundamentales probados u otras metodologías más simples.
- La metodología ha sido *validada*: la metodología ha funcionado correctamente en un gran número de aplicaciones.
- La metodología es *apropiada* al problema que quiere resolverse.

2.2. Metodologías para el desarrollo de Videojuegos

Al comparar la gran cantidad de estudios realizados con respecto a las áreas relacionadas con las ciencias de la computación como el desarrollo de bases de datos o la creación de páginas web, es posible apreciar como la comunidad dedicada al desarrollo de videojuegos se encuentra iniciando el proceso de articulación de los conceptos y técnicas específicas de su medio, de manera que sea posible establecer una metodología general para el desarrollo de videojuegos [24]. En los sitos web de las diferentes comunidades de desarrolladores de videojuegos como Gamasutra [25], GameDev [26], entre otras, es posible encontrar, propuestas de metodologías para la construcción de videojuegos [23]. Estas propuestas tienen un enfoque muy marcado hacia las estrategias de mercadeo, o la producción, desde un punto de vista empresarial o administrativo [23]. Estas propuestas metodológicas abarcan de forma muy general los diferentes procedimientos llevados a cabo por los miembros del equipo de trabajo. Sin embargo, existen algunos métodos donde el enfoque es totalmente artístico y se enmarcan varias tareas específicas, que permiten la construcción de un documento denominado comúnmente como "Documento de Diseño" [27]. En éste, se plantean un conjunto de procesos de desarrollo comunes para la creación de un videojuego, entre los cuales es usual encontrar el: *storyline*[2], el diseño de personajes, el diseño de escenarios y la definición de reglas [27]. Pero, en la definición de estos procesos frecuentemente es ignorado brindar un enfoque que facilite el trabajo de los miembros del equipo de programación, lo que puede verse reflejado en la no implementación de algunos de los aspectos

[2] Storyline: Historia principal de un drama o trabajo literario.

del juego o el retraso a la hora de entregar la totalidad del juego [23]. Teniendo en cuenta que existen varias metodologías de desarrollo de software, que plantean procedimientos que permiten la integración de las propuestas artísticas existentes para el desarrollo de videojuegos. Integración que brindaría la posibilidad de definir procesos que involucren a todo el equipo de trabajo, de manera que, cada miembro pueda entender los conceptos generales del videojuego a realizar independientemente desde su formación.

2.3. Revisión de metodologías para agentes.

Existen varios enfoques para el desarrollo de software entre los cuales es relevante mencionar el orientado a objetos, orientados a aspectos, orientado a agentes, entre otros. Para el caso particular de los videojuegos (en especial los de aventura) es necesario describir las relaciones (conversaciones, colaboraciones, etc.) entre los personajes de la historia, además de las relaciones con su entorno [12]. Basados en la definición de agentes de Wooldridge y Jennigns [29] donde un agente debe poseer las propiedades de habilidad social, re-actividad y pro-actividad [29]. Resulta conveniente aprovechar la similitud en las propiedades definidas de los agentes y los personajes de un videojuego, para analizar la historia del videojuego desde una perspectiva orientada a agentes software.

A continuación, en la **Tabla 1 - Revisión de Metodologías** se presentan las metodologías de agentes estudiadas, además se expone una breve descripción de los diferentes modelos que posee cada una de ellas:

METODOLOGIA	DESCRIPCION	MODELOS
MASE [21]	Se concibe como una abstracción del paradigma orientado a objetos donde los agentes son especializaciones de objetos. En lugar de simples objetos con métodos que pueden invocarse desde otros objetos, los agentes se coordinan unos con otros vía conversaciones y actúan pro activamente para alcanzar metas individuales y del sistema.	➢ *Capturar los objetivos:* Es la primera fase de la metodología. Toma las especificaciones iniciales del sistema y las transforma en un conjunto de metas del sistema. ➢ *Transformar metas en roles:* En este paso se transforman las metas estructuradas en la fase anterior, de manera que resulten más útiles para la construcción de sistemas MultiAgentes, convirtiéndolas en: los roles y sus tareas asociadas. ➢ *Aplicar casos de uso:* Tras la creación de los roles, se asocian las tareas para cada rol. Cada meta asociada a un rol puede

METODOLOGIA	DESCRIPCION	MODELOS
		tener tareas que definan los detalles de cómo se logra. ➢ *Crear clases de Agentes:* Llegada esta fase, las clases agente se identifican a partir de los roles. El resultado es el Diagrama de Clases Agente, que muestra las clases agente junto con las conversaciones que mantienen dichos agentes. ➢ *Construir conversaciones:* Una conversación bajo MASE, define el protocolo coordinado entre dos agentes. La metodología define específicamente una conversación como dos diagramas de comunicación de Clases, uno para el iniciador de la conversación y otro para el que responde. ➢ *Ensamblar clases de agentes:* En esta etapa de MASE se construye la parte interna de una clase agente. ➢ *Diseño del sistema:* Es la fase final de la metodología MASE. En esta fase es donde se instancian las clases agente. Se utiliza un diagrama de desarrollo para mostrar el número, tipo y localidad de los agentes dentro del sistema.
GAIA [31]	GAIA es una metodología para el diseño de sistemas basados en agentes cuyo objetivo es obtener un sistema que maximice alguna medida de calidad global. GAIA pretende ayudar al analista a ir sistemáticamente desde unos requisitos iniciales a un diseño que, según los autores, esté lo suficientemente detallado como para ser implementado directamente.	➢ *Roles:* Los modelos de roles organizacionales describen de manera precisa todos los roles que constituyen la organización computacional, en términos de sus funcionalidades, actividades y responsabilidades, y los términos de sus protocolos y patrones de interacción. ➢ *Interacciones:* Corresponde a todos los principios, procesos,

METODOLOGIA	DESCRIPCION	MODELOS
		protocolos, estrategias de comunicación, políticas sociales y estructuras que permiten a los agentes transportar información ➢ *Agentes:* El modelo de agentes define el tipo de agentes que creará el sistema, tanto durante su desarrollo como durante la ejecución. ➢ Servicios: Identifica los servicios que son necesarios para llevar a cabo el rol del agente, como su nombre indica.
MESSAGE [32]	Propone el análisis y diseño del Sistemas MultiAgente (SMA) desde cinco puntos de vista para capturar los diferentes aspectos de un SMA: el de Organización, el de Tareas/Objetivos, los objetivos que persiguen y las tareas implicadas en el proceso; el de Agente, que contiene una descripción detallada y extensa de cada agente y rol dentro del SMA.	➢ *Organización:* captura la estructura global del sistema. ➢ *Objetivos/Tareas:* determina qué hace el SMA y sus agentes constituyentes en términos de los objetivos que persiguen y las tareas implicadas en el proceso. ➢ *Agentes/Roles:* contiene una descripción detallada y extensa de cada agente y rol dentro del SMA ➢ *Interacción:* trata las interacciones a distintos niveles de abstracción. ➢ *Dominio:* actúa como repositorio de información (para entidades y relaciones) concernientes al dominio del problema.
INGENIAS [33]	INGENIAS es considerado como la evolución de las ideas propuestas por la metodología MESSAGE. Teniendo encuentra que en INGENIAS se profundiza en los elementos mostrados en: el método de especificación, los procesos de desarrollo, además, incorporar nuevas herramientas de soporte y ejemplos de desarrollo.	➢ Agentes: Se usa para describir agentes particulares excluyendo las interacciones con otros agentes. Se centra en la funcionalidad del agente y en el diseño de su control. ➢ Tarea: Se basa en encontrar las consecuencias que tiene la ejecución de tareas y por qué se deberían ejecutar. Justifica la ejecución de tareas

METODOLOGIA	DESCRIPCION	MODELOS
		basándose en objetivos. ➢ Interacciones: Define las interacciones entre los agentes o entre agentes y humanos, se definen a alto nivel, en diseño se detalla el protocolo de interacción. ➢ Entorno: Define las entidades del entorno del SMA con las que vaya a interactuar.
ZEUS [34][35]	La metodología ZEUS propone un desarrollo en cuatro etapas: el análisis del dominio, el diseño de los agentes, la realización de los agentes y el soporte en tiempo de ejecución. Las etapas soportadas por la herramienta son la de *realización de los agentes* y la de *soporte en tiempo de ejecución*. Las etapas anteriores se basan en el uso de roles para analizar el dominio y en su asignación a agentes.	➢ Roles: Cada rol describe una posición y un conjunto de responsabilidades en un cierto contexto. Los roles deben: - Ser modulares - Alta cohesión - No disponer de responsabilidades extrañas - Ser completo (no triviales) - Bajo acoplamiento.
MAS – CommonKADS [36]	Esta metodología extiende CommonKADS aplicando ideas de metodologías orientadas a objetos para su aplicación a la producción de SMA. La metodología CommonKADS gira alrededor del modelo de experiencia y está pensada para desarrollar sistemas expertos que interactúen con el usuario. De hecho, considera sólo dos agentes básicos: el usuario y el sistema. MAS-CommonKADS extiende los modelos de CommonKADS para tener en cuenta la posibilidad de que dos o más componentes del sistema interactúen.	➢ Agentes: Específica las características de un agente: sus capacidades de razonamiento, habilidades, servicios, sensores, efectores, grupos de agentes a los que pertenece y clase de agente. Un agente puede ser un agente humano, software, o cualquier entidad capaz de emplear un lenguaje de comunicación de agentes. ➢ Tarea: Describe las tareas que los agentes pueden realizar: los objetivos de cada tarea, su descomposición, los ingredientes y los métodos de resolución de problemas para resolver cada objetivo. ➢ Experiencia: Describe el conocimiento necesitado por los agentes para alcanzar sus

METODOLOGIA	DESCRIPCION	MODELOS
		objetivos. Sigue la descomposición de *CommonKADS* y reutiliza las bibliotecas de tareas genéricas. ➢ Organización: es una herramienta para analizar la organización humana en que el sistema multiagente va a ser introducido y para describir la organización de los agentes software y su relación con el entorno. ➢ Diseño: Mientras que los otros cinco modelos tratan del análisis del sistema multiagente, este modelo se utiliza para describir la arquitectura y el diseño del sistema multiagente como paso previo a su implementación.
MASINA [37]	(Multi Agent Systems in Automation). Esta metodología usa como base a MAS-CommonKADS, proponiendo algunas extensiones, modificaciones y sustituciones de los modelos definidos en MAS-CommonKADS. MASINA permite plasmar elementos fundamentales del área de SMA, como representar la noción de inteligencia en un agente (modelado de agentes inteligentes), a nivel colectivo, los mecanismos de coordinación entre agentes (planificación emergente, resolución de conflictos, etc.), la comunicación directa o indirecta, entre otras cosas.	➢ Agentes: describe las características de cada agente, con dos extensiones particulares, una que permite usar modelos de referencia para especificar un agente dato, y otra para modelar los niveles de abstracción del Sistema Multi-agentes ➢ Tareas: representa las actividades realizadas por los agentes para alcanzar sus objetivos, en nuestra extensión es posible especificar tareas que requieren el uso de técnicas inteligentes, así como describir el macro algoritmo de la tarea ➢ Inteligencia: describe los conocimientos, mecanismo de aprendizaje y métodos de razonamiento necesarios para que los agentes cumplan sus objetivos

METODOLOGIA	DESCRIPCION	MODELOS
		➢ Comunicación: agrupa las interacciones en conversaciones y define los mecanismos para que esas interacciones se lleven a cabo coherentemente ➢ Coordinación: describe los actos de habla (interacciones).

Tabla 1 - Revisión de Metodologías

Es importante aclarar que el objetivo de esta revisión bibliográfica sobre las metodologías mencionadas, no es determinar cuál es la mejor metodología de agentes, ya que todas ellas poseen sus ventajas e inconvenientes. La idea es determinar cuál de estas metodologías presenta las bases apropiadas, de forma que sea posible adaptar de manera ágil y eficiente; los conceptos y las necesidades al desarrollar un videojuego de aventura en 3D. Durante la revisión bibliográfica se tuvieron en cuenta de las metodologías: MASE, ZEUS, GAIA, MASINA, INGENIAS y MAS-CommonKADS, sus etapas de análisis, diseño e implementación de cada metodología, evaluadas en base a criterios como [38]: ¿se encuentra bien documentada?, ¿existe un documento donde se explique el procedimiento de uso de la metodología?; ¿Es fácil de aplicar al desarrollo de videojuegos?, ¿los procesos se adaptan de manera adecuada al desarrollo de juegos de aventura?; ¿Sus modelos son fáciles de comprender y aplicar?; ¿se encuentra bien argumentada?, ¿sus bases y teorías se encuentran apoyadas en fundamentos probados u otras metodologías? (Una descripción más detallada de esta revisión se encuentra en el Anexo B de este proyecto) [38].

La metodología MASE plantea un proceso de desarrollo completo y detallado con un conjunto de ideas muy similares a las planteadas en la arquitectura BDI (Beliefs, Desires, Intentions), además cuenta con una herramienta de apoyo llamada agentTool, que permite trabajar de forma gráfica con los modelos propuestos. No obstante, existe muy poco material de referencia con respecto a esta metodología [39]. GAIA provee una metodología basada en el paradigma de agentes con una robusta estructura. Sin embargo, se encuentra definido de tal forma que resulta complejo pasar de la capa de diseño a la de implementación, además no cuenta con una herramienta de desarrollo [31]. MESSAGE es una metodología relativamente reciente, que intentó unificar los resultados de varias de las metodologías de agentes, se encuentra basada en una serie de definiciones de UML lo que hace fácil de entender para muchos de los ingenieros, con el inconveniente de que no llegó a concretarse totalmente dejando vacíos en su modelo de diseño y la ausencia de una herramienta de desarrollo [33]. La metodología ZEUS se centra en agrupar en roles la funcionalidad del sistema, restando importancia a otros de sus modelos planteados. Además, cuenta con una muy completa herramienta de desarrollo, aunque no se profundiza en la aplicación de la herramienta dentro del proceso de desarrollo [34]. MASINA es una metodología basada en MAS-CommonKADS con modificaciones que le permiten incorporar comportamientos inteligentes. Aunque no ha sido liberada en su totalidad cuenta con procesos bien definidos, no cuenta con una herramienta de desarrollo pero están trabajando en ello [37]. INGENIAS es considerada como la evolución de MESSAGE, donde se definen meta-modelos (una descripción de alto nivel de qué elementos tiene el modelo) con los cuales se describe el sistema, los procesos

que se proponen pueden resulta demasiado excesivos al tratarse de desarrollo reducidos, aunque cuenta con una muy buena documentación su aplicación resulta relativamente compleja, cuenta con una herramienta de desarrollo muy completa [33]. Para finalizar, la metodología MAS-CommonKADS extiende de CommonKADS, y fue de las primeras metodologías en aplicar la producción de Sistemas Multi-Agentes. Se basa en los sistemas de conocimiento lo que facilita su compresión al personal de múltiples disciplinas, sus procesos de desarrollo permiten adaptarse al tamaño del problema, además de encontrase muy bien documentados [36]. Aunque existen herramientas de desarrollo de acceso limitado, se encontraron algunos trabajos que facilitan la implementación. Entre estos, existe un framework elaborado en un proyecto del programa de Ingeniería de Sistemas de la Universidad del Cauca, el cual permite una fácil adaptación al framework de XNA 2.0 para el desarrollo de Juegos[61]. La Tabla 2 – Evaluación , cuantifica los datos resumidos en este apartado, los cuales se encuentran basados en la revisión metodología del anexo B del presente proyecto [38]. A partir de esta revisión bibliográfica pudo definirse basado en criterios encontrados en trabajos realizado para comparar metodologías [19] [64] y la información para la creación de videojuegos [15], se determina trabajar con la metodología MAS-CommonKADS, ya que sus características se adecúan de manera, mucho más completa al desarrollo de este proyecto.

METODOLOGIA	ANALISIS	DISEÑO	DESARROLLO	PROMEDIO
MASE	3	3.5	2.5	**3**
GAIA	2.5	2.5	0	**1.7**
MESSAGE	2	2	0	**1.3**
INGENIAS	3.2	3.5	3.5	**3.4**
ZEUS	1.5	1.5	4.3	**2.4**
MAS-CommonKADS	**4.5**	**4.5**	**2**	**3.7**
MASINA	2.5	2.5	0	**1.7**

Tabla 2 – Evaluación de Metodologías

2.4. Diseño de la metodología

La definición de la metodología para la construcción de personajes se llevó a cabo mediante el estudio y la investigación de varias metodologías para la construcción de agentes [40]. Como resultado de dicho estudio se determina como base para la construcción de la metodología a MAS-CommonKADS. En este apartado se revisan cada uno de los elementos, modelos, y diagramas, con el fin de entenderlos y extenderlos cubriendo las necesidades presentadas al construir videojuegos de aventura 3D. MAS-CommonKADS es una extensión metodológica de su predecesor CommonKADS [36], la cual se enfoca en la construcción de sistemas expertos. La extensión orientada a agentes aprovecha los principios del paradigma orientado a objetos que tienen ya una buena base con diferentes metodologías. Esta metodología ha sido la primera en plantear la integración del SMA con un modelo de ciclo de vida de software, más concretamente el espiral dirigido por riesgos. El modelo del ciclo de vida, para el desarrollo de sistemas multi-agente presentado por MAS- CommonKADS enumera las siguientes fases [36]:

Conceptualización: Se encarga de recopilar toda la información existente del dominio del problema con el fin de organizarla, para describir la totalidad el problema de una forma muy general y usando lenguaje natural. El hecho de que esta etapa se encuentre definida en lenguaje natural permite la integración de varias de las tareas definidas por los "artistas" sin generar cambios significativos en las mismas.

Análisis**:** Se determinan todos los requisitos del sistema partiendo del enunciado del problema definido en la etapa de conceptualización.

Diseño**:** Especificación de los requisitos de la fase de análisis que se puede lograr mediante el desarrollo del modelo de diseño. Se determinan las arquitecturas tanto de la red multiagente como de cada agente.

Las fases enunciadas anteriormente están compuestas por los siguientes modelos:

Modelo de Agente (AM)*:* Específica las características de un agente: sus capacidades de razonamiento, habilidades, servicios, sensores, efectores, grupos de agentes a los que pertenece y clase de agente. Un agente puede ser un agente humano, software, o cualquier entidad capaz de emplear un lenguaje de comunicación de agentes [36].

Modelo de Organización (OM)*:* Es una herramienta para analizar la organización humana en que el sistema multi-agente va a ser introducido y describe la organización de los agentes software y su relación con el entorno [36].

Modelo de Tareas ™: Describe las tareas que los agentes pueden realizar: los objetivos de cada tarea, su descomposición, los elementos y los métodos de resolución de problemas para resolver cada objetivo [36].

Modelo de la Experiencia (EM): Describe el conocimiento necesitado por los agentes para alcanzar sus objetivos. Sigue la descomposición de *CommonKADS* y reutiliza las bibliotecas de tareas genéricas [36].

Modelo de Comunicación (CM): Describe las interacciones entre un agente humano y un agente software. Se centra en la consideración de factores humanos para dicha interacción [36].

Modelo de Coordinación (CoM): *D*esarrolla y describe las interacciones entre los agentes de un sistema multi-agente. La objetivo de este modelo es definir los servidos ofrecidos por un agente, brindando la posibilidad de realizar tareas para otros agentes [36].

Modelo de Diseño (DM): Mientras que los otros seis modelos tratan del análisis del sistema multi-agente, este modelo se utiliza para describir la arquitectura y el diseño del sistema multi-agente como paso previo a su implementación [36].

La **Figura 1 – Modelos MAS-CommonKADS** muestra gráficamente la relación que existe entre cada uno de los modelos propuestos por MAS-CommonKADS.

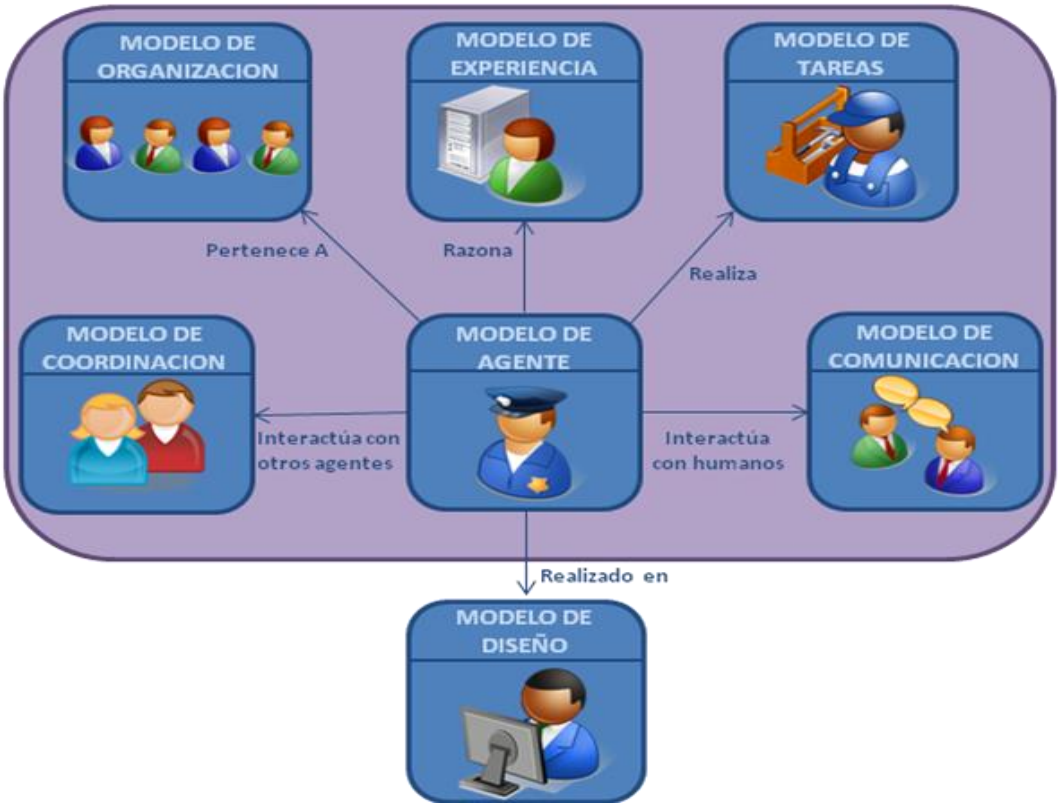


Figura 1 – Modelos MAS-CommonKADS

En el siguiente apartado se profundizará en cada una de las fases de la metodología MAS-CommonKADS, mostrando los resultados de introducir en sus procedimientos los conceptos y tareas propios de la construcción de videojuegos de aventura 3D.

2.4.1. Conceptualización

Esta sección presenta la información del dominio del problema con el fin de organizarla de forma tal que queden sentadas las bases para la construcción. Teniendo en cuenta que se encarga de formalizar todas las ideas generales sobre la construcción, describe la totalidad el problema de una forma muy general y usando lenguaje natural [36]. Para este proyecto, en la fase de concepción se define a gran escala las características generales del videojuego, la cuales forman la base para todo el proceso de construcción de un videojuego de aventura pero se encuentran enfocadas en mayor proporción a una concepción artística del juego [19]. Existen varios temas acerca del videojuego que deben estar bien definidos antes de iniciar con la construcción con el fin de prevenir la generación de ambigüedades en fases futuras. Esta es la razón por la que el desarrollo de esta fase es de suprema importancia, ya que constituye los cimientos de toda la construcción del videojuego de aventura [31]. De igual forma, en la fase de conceptualización se definen varias tareas y a partir de la integración de los resultados obtenidos de estas tareas se

logra concebir la creación del **Documento de Conceptualización del Videojuego** (Documento entregable de esta fase). Entre las tareas específicas de esta fase se han definido las siguientes:

2.4.1.1. Storyline

En este apartado se debe realizar una descripción general de la historia en la que se centra el videojuego, que permitirá obtener una vaga idea de su concepción [31]. El storyline puede ser visto como una representación general en la cual se resumen los hechos más relevantes que se llevarán a cabo durante el transcurso del videojuego. En la mayoría de los casos, el storyline se construye en forma de cuento o historia para facilitar su comprensión, como lo aconsejan varios autores [15][31] [19].

2.4.1.2. Análisis del Argumento Base

Una vez definido el storyline, se realiza una descripción formal que tiene una estructura orientada a un guión cinematográfico, comic u obra de teatro [31] y que facilita la definición de cada una de las acciones los personajes dentro de la historia. Es importante resaltar que estas acciones son definidas a partir de una orientación literaria [19] y se definen usando una secuencia de escenas (Ver *Tabla 3 - Descripción de Escena)*.

ESCENA		
CASO:		
CONTROLES:		
PANTALLA:		
NOTAS:		
ACCIONES:		
HITS DE CASO:		
RESTRICCIONES Y POSICIONES DEL MAPA:		
PERSONAJES DEL MÓDULO:		
DIALOGOS FIJOS:		
DIALOGOS LIBRES:		

Tabla 3 - Descripción de Escena

Los elementos presentados en la *Tabla 3 - Descripción de Escena* se describen a continuación:

- **Caso:** Se define una numeración que permita seguir la secuencia en que se ejecutan las escenas.
- **Controles:** Se deben listar los controles que se encuentran habilitados durante la ejecución de la escena.
- **Pantalla:** Descripción en lenguaje natural, de la imagen mostrada durante la escena.
- **Notas:** Se detallan las actividades de particular importancia durante esta escena.

- **Acciones:** Se describen los actos más importantes que tiene lugar durante el tiempo en que transcurre la escena.
- **Hit de Caso:** Son declarados en lenguaje natural los eventos de mayor importancia en la escena.
- **Restricciones y Posiciones del Mapa:** Si existe, se presenta una imagen descriptiva del mapa y la ubicación de los objetos en el.
- **Personajes del Módulo:** Son listados todos personajes que tiene puesta en escena durante este periodo.
- **Diálogos Fijos:** Se relatan las conversaciones entre los personajes durante el transcurso normal de la escena.
- **Diálogos Libres:** Se presentan un conjunto de diálogos alternativos en caso de existir una variante en el curso normal de la escena.

2.4.1.3. Definición de Reglas

Cada juego cuenta con un conjunto de reglas las cuales enmarcan y estructuran su definición. Estas reglas pueden ser definidas como pautas abstractas encargadas de la descripción funcional del sistema [41]. Las reglas permiten establecer cualidades particulares asociadas a un juego, y operan estableciendo un límite a las actividades que pueden desarrollar los jugadores dentro del juego obligándolos a tomar caminos específicos para llegar a las metas y garantizando que todos tomen camino definidos [42].

Las siguientes características se encuentran presentes en las reglas de un videojuego:

- E**xplícitas e inequívocas:** Son completas y carecen de ambigüedad. Del mismo modo, las normas tienen que ser totalmente explícitas con respecto a lo que se quiere transmitir [42].

- **Compartidas por todos los jugadores**: En un juego con muchos jugadores es necesario que todos compartan el mismo número de reglas [42].

2.4.1.4. Definición del Mundo

Cuando se hace referencia al mundo de un videojuego, se deben tener en cuenta los elementos que conforman la ambientación de sus distintas escenas. Para esto es necesario realizar una descripción detallada, que permita la definición de aspectos como la escala que determinará el tamaño del terreno, los distintos objetos y su ubicación dentro del mundo [42]. Es posible realizar una definición del mundo mediante la creación de gráficos que ilustren el orden de los objetos en las determinadas escenas, los cuales servirían de guía al momento de organizar los elementos en el mundo.

2.4.1.5. Casos de Uso de los Personajes

Una actividad realizada por un personaje del juego representa el papel de un caso de uso, dado que describe las posibles interacciones o usos de un **usuario y/o agente** con el sistema. Los usuarios y/o agentes del sistema se denominan *actores*, y representan a las entidades catalogadas como externas al sistema. Los casos de uso se pueden combinar, indicando que un caso de uso extiende otro caso o usa un caso previo. Para este proyecto, el objetivo de definir estas actividades es identificar las acciones descritas en la historia del videojuego que realizan los personajes. Por esta razón se trabajara con actores sino con *personajes*.
El análisis mediante casos de uso sigue varios pasos para lograr su identificación, estos han sido adaptados para lograr identificar las actividades que realizan los personajes [43]:

- Identificar los personajes: Interesa identificar los papeles que desempeñan los elementos que interactúan dentro del videojuego. Cada papel se considera un personaje diferente. Normalmente este proceso requerirá varias iteraciones.

- Identificar los casos de uso: En este caso se enumeran las actividades realizadas por los personajes. Para facilitar desarrollo de esta tarea se recomienda el uso de las siguientes preguntas: ¿Cuáles son las principales tareas o funciones realizadas por cada personaje? ¿Qué información del entorno adquiere, produce o cambia cada personaje? ¿Informa algún personaje al videojuego sobre cambios externos en el entorno del videojuego? ¿Qué información desea cada personaje del entorno del videojuego? ¿Desea algún personaje ser informado de cambios inesperados?

- Agrupar los casos de uso: si aparecen variaciones del mismo tema estas son agrupadas (por ejemplo: 'Pelear usando arma', 'pelear usando las manos', 'pelear usando las piernas').

- Determinar las interacciones de cada caso de uso identificado: En particular se definen aspectos como: que personaje inicia cada caso de uso, existen precondiciones que deben ser ciertas para que un determinado caso de uso pueda comenzar y la conclusión lógica de la transacción.

- Describir los casos de uso: Los casos de uso se suelen describir informalmente empleando lenguaje natural o derivaciones de la notación gráfica propuesta por Jacobson [43].

- Considerar todas las excepciones que pueden ocurrir al llevar a cabo una transición y especificar cómo afectan al caso de uso.

- Buscar relaciones entre casos de uso: factorizar partes comunes e indicar si un caso de uso agrega las interacciones de otro (relación "usa") o añade información de otro caso (relación "extiende").

2.4.1.5.1. Ciclo de Desarrollo de los Casos de Uso

A continuación se muestran los diferentes ítems que deben ser desarrollados para definir la etapa de Conceptualización dentro de la metodología MAS-CommonKADS [43]:

- Identificación de los personajes (actores)
- Descripción de los actores
- Identificación de los casos de uso
- Descripción de los casos de uso

En esta metodología las actividades son orientadas a la identificación de personajes, permitiendo definir sus características más relevantes e identificar las funciones que realizan, las cuales son representadas a través de los casos de uso.

2.4.1.5.1.1. Identificación de Personajes

Se describe cada uno de los atributos que deben tener los personajes al momento de identificarlos, se propone una plantilla presentada en la Tabla 4 – Identificación Personaje/Agente [17][47]:

IDENTIFICACION PERSONAJES/AGENTES	
Id	
Nombre	
Personajes con los que interactúa:	
Descripción:	
Tipo:	
Referencias cruzadas	

Tabla 4 – Identificación Personaje/Agente

Los elementos presentados en la Tabla 4 – Identificación Personaje/Agente se describen a continuación:

- **ID:** código con el que se identifica al personaje, debe ser único y es escogido libremente.
- **Nombre:** identificación del personaje con una cadena de texto corta, debe ser única.
- **Personajes con los que interactúa:** enumera los personajes con los cuales tiene una interacción durante la realización de la historia.
- **Comportamiento:** reacciones generales que debe tener el personaje.
- **Tipo:** principal, secundario, extra.
- **Referencias cruzadas:** relación de este artefacto con los creados anteriormente.

2.4.1.5.1.2. Descripción de los Personajes

Se profundiza en una descripción de las características de los personajes aumentando algunos atributos importantes en lo concerniente a la definición del personaje, se propone una plantilla en la Tabla 5 - Descripción Personaje/Agente [17][47]:

DESCRIPCION PERSONAJES/AGENTES	
ID:	
Nombre:	
Personajes con los que interactúa:	
Comportamiento:	
Dimensión física o fisiológica:	
Dimensión social:	
Dimensión psicológica:	
Tipo:	
Metas: **Motivación:**	
Intención:	
Objetivo:	
Evolución:	
Referencias cruzadas:	

Tabla 5 - Descripción Personaje/Agente

Los elementos presentados en la Tabla 5 - Descripción Personaje/Agente se describen a continuación:

- **ID:** código con el que se identifica al personaje, debe ser único y es escogido libremente.
- **Nombre:** identificación del personaje con una cadena de texto corta, debe ser única.
- **Personajes con los que interactúa:** enumeración de los personajes con los que interactúa.
- **Comportamiento:** descripción de las diferentes reacciones del personaje ante diferentes situaciones o demás personajes.
- **Dimensión física o fisiológica:** a través del lenguaje natural se detallan características morfológicas y fisiológicas del personaje como: sexo, edad, descripción física (peso, altura), apariencia, defectos deformidades, enfermedades);
- **Dimensión social:** haciendo uso del lenguaje natural se detallan características sociológicas del personaje como: clase social, ocupación, educación, religión, raza, nacionalidad, filiación política;
- **Dimensión psicológica:** en lenguaje natural se detallan características psicológicas del personaje como: historia familiar, Vida sexual, autoestima, actitud frente a la vida, habilidades, cualidades, inteligencia.
- **Tipo:** descripción del tipo al que pertenece (principal, secundario, extra).
- **Metas:**
  - **Motivación:** razón por la cual el personaje quiere o necesita lograr el objetivo
  - **Intención:** describe si el personaje tiene una buena o mala finalidad.

- **Objetivo:** meta a lograr u objeto a obtener por el personaje

- **Evolución:** acciones presentadas a lo largo de la historia que podrían cambiar actitudes o comportamientos del personaje.
- **Referencias cruzadas:** relación de este artefacto con los creados anteriormente.

2.4.1.5.1.3. Identificación de los casos de uso de los personajes

Se describen cada uno de los atributos de las actividades que desarrollan los personajes, se propone una plantilla en la tabla Tabla 6 – Caso de Uso [43]:

CASOS DE USO	
Id:	
Nombre de la Actividad:	
Personajes:	
Resumen:	
Referencias Cruzadas:	

Tabla 6 – Caso de Uso

Los elementos presentados en la Tabla 6 – Caso de Uso se describen a continuación:

- **Id:** código con el que se identifica la actividad, debe ser único y es de nombrado libre.
- **Nombre de la Actividad:** identificación de la actividad usando una cadena de texto.
- **Personajes:** lista de los personajes, en la cual debe identificarse el encargado de iniciar la actividad.
- **Resumen:** síntesis de la actividad.
- **Referencias Cruzadas:** relación de este artefacto con los creados anteriormente.

2.4.1.5.1.4. Descripción de los casos de uso

Como su nombre lo indica, se debe realizar una descripción más detallada de cada uno de los atributos de las actividades, aumentado algunos importantes [20]. La descripción de los casos de uso, consta de dos tipos: una representación textual en la que se hace referencia en la Tabla 7 – Descripción Caso de Uso y una representación grafica en la que se usan los elementos mostrados en la Tabla 8 – Representación Gráfica Casos de Uso

- **Representación Textual:**

DESCRIPCION CASOS DE USO	
Id:	
Nombre de la Actividad:	
Precondiciones:	
Personajes:	
Resumen:	
Tipo:	
Curso normal de los eventos:	
Excepciones:	
Post-condiciones:	
Referencias Cruzadas:	

Tabla 7 – Descripción Caso de Uso

Los elementos presentados en la Tabla 7 – Descripción Caso de Uso se describen a continuación:

- **Id:** código con el que se identifica la actividad, debe ser único y es de libre nombrado.
- **Nombre de la Actividad:** identificación de la actividad usando una cadena de texto.
- **Personajes:** lista de los personajes, en la cual debe identificarse el encargado de iniciar la actividad.
- **Precondiciones:** condición que debe ser cierta para que la actividad pueda ser iniciada.
- **Resumen:** breve descripción de la secuencia de interacciones entre los personajes.
- **Tipo:** clasificación de las actividades en:
  - Primarias: representan las actividades comunes más importantes.
  - Secundarios: representan actividades menores o raros.
  - Opcionales: representan actividades que pueden no abordarse.
- **Curso normal de los eventos:** Descripción detallada e informal de las interacciones, referenciando las posibles excepciones.
- **Excepciones:** Descripción de cada comportamiento inesperado y los posibles caminos que puede tomar.
- **Postcondiciones:** Descripción de las condiciones en las que debe encontrarse el sistema una vez cumplido el caso de uso.
- **Referencias Cruzadas:** relación de este artefacto con los creados anteriormente.

- **Representación Gráfica:**

Con esta representación se explica gráficamente un conjunto de casos de uso del videojuego, los personajes y la relación entre estos y los casos de uso. A continuación se presentan los símbolos con los que se van a representar cada uno de los objetos del análisis Tabla 8 – Representación Gráfica Casos de Uso [43]:

Personaje-Agente	Jugador-Agente	Relación	Caso de Uso
		<<Tipo de relación>> 	

Tabla 8 – Representación Gráfica Casos de Uso

El objetivo de esta representación es ofrecer un diagrama contextual que permite conocer rápidamente los personajes del sistema y las formas básicas en que se utilizan.

2.4.2. Modelo de Comportamiento

En MAS-CommonKADS se plantean un modelo de agentes, cuyo objetivo es definir las características especificas de cada uno de los agentes que interactúan en el sistema y un modelo de tareas donde se describen las labores que cada agente puede realizar [36]. Para la construcción de videojuegos de aventura basados en personajes el uso de estos modelos no es necesario, ya que los agentes y sus tareas se encuentran definidos desde la primera etapa (conceptualización). Teniendo en cuenta que estos son representados a través de los personajes del videojuego, y que cada uno se asocia con un conjunto de tareas o casos de uso que pueden realizar, resulta más importante centrar la atención en el análisis de los comportamientos para cada uno de los agentes. En este proyecto se propone un nuevo modelo el cual es denominado **modelo de comportamiento** [36]**.**

En el modelo de comportamiento se debe realizar el mapeo de los casos de uso a las características que poseen los agentes. Utilizando como base la definición de agente de **Wooldridge y Jennings** [36], se ha decido que utilizar las propiedades de reactividad, proactividad y habilidad social, debido a que a lo largo del desarrollo de la historia del videojuego, el personaje/agente debe ser flexible y adaptarse a los cambios que ocurren en su entorno [36]. Estas propiedades se explican a continuación:

- *Reactivo:* es aquel agente que mantiene una interacción constante con su ambiente, y responde a los cambios que ocurren en este (en un tiempo es que la respuesta es útil).

- *Proactivo:* es el hecho de que un agente tome la iniciativa, sea mediante sugerencias o acciones anticipadas para alcanzar sus objetivos.

- *Habilidad social:* en agentes es la habilidad de interactuar con otros agentes (y posiblemente con humanos) por medio de algún tipo de lenguaje de comunicación de agentes, y quizás cooperar con otros.

A continuación se propone una plantilla en la Tabla 9 **–** Comportamiento, y luego se presenta la descripción de algunos atributos más importantes que se deben tener en cuenta al mapear cada caso de uso presente en dicha tabla:

- **Representación Textual**

COMPORTAMIENTO	
Id:	
Comportamiento:	
Caso de uso:	
Tipo:	
Referencias Cruzadas:	

Tabla 9 – Comportamiento

Los elementos presentados en la Tabla 9 – Comportamiento se describen a continuación:

- **ID:** identificador para cada caso de uso.
- **Comportamiento:** nombre del comportamiento
- **Caso de uso:** el nombre del caso uso que está relacionado con el comportamiento.
- **Tipo:** pueden ser:
  - Reactivo: se describe la reacción que lo dispara.
  - Proactivo: tiempo de duración y/frecuencia.
  - Social: agentes relacionados.
- **Acción que lo dispara**: en caso de que el tipo sea reactivo
- **Referencias Cruzadas:** relaciones con los artefactos realizados anteriormente.

2.4.2.1. Notación gráfica

Los diagramas de interacción son una forma de realizar una representación gráfica de los casos de uso, y describen el curso particular de los eventos de los casos ocurridos entre, los jugadores que interactúan directamente con el videojuego y los eventos generados por los personajes [36]. En este diagrama el tiempo avanza de arriba hacia abajo, y el ordenamiento de los eventos debería seguir el orden indicado en el caso de uso. En este caso las interacciones también determinan el comportamiento de los personajes/agentes, mostrando cuál es su reacción cuando actúan sobre ellos. Igualmente muestran cómo el comportamiento se determina en función de los objetivos de los personajes/agentes y las tareas a ejecutar. Se puede concluir que existe un importante vínculo entre interacciones, objetivos y tareas. Estos diagramas se ilustran en la Figura 2 – Diagramas de Interacción de personajes, la Figura 3 – Diagramas de Interacción de jugadores.

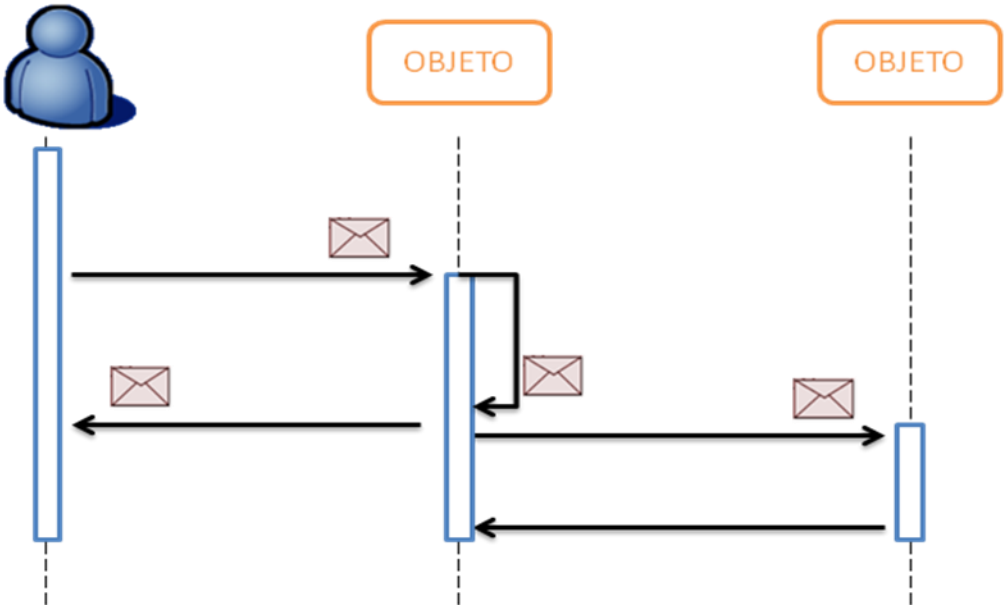


Figura 2 – Diagramas de Interacción de personajes

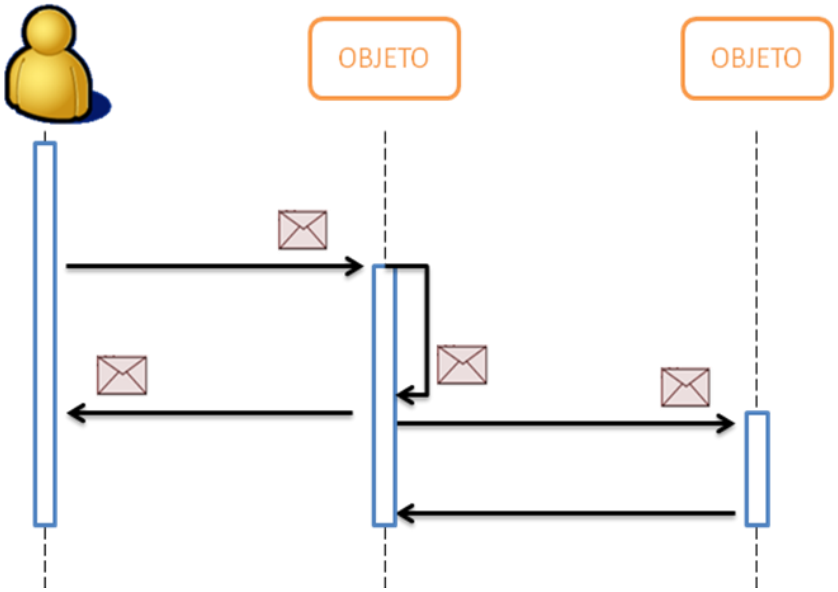


Figura 3 – Diagramas de Interacción de jugadores

2.4.3. Modelo de Experiencia

El modelo de experiencia es el encargado de administrar el conocimiento de los personajes y del videojuego en general, permitiendo definir su nivel de complejidad y determinar la secuencia de comportamientos de los distintos actores del juego de acuerdo con las acciones realizadas por el jugador [53]. Para alcanzar el objetivo de este modelo, es necesario definir una técnica que permita dotar al videojuego con las características planteadas, entre las técnicas que se pueden revisar para la implementación de este aspecto del juego se encuentran: las Máquinas de estado finito, los algoritmos genéticos, las redes neuronales, entre otras [53]. Es necesario tener en cuenta que algunas de estas técnicas se adaptan con mucha más facilidad que otras al desarrollo de videojuegos, aunque todas podrían ser utilizadas para este fin, es necesario resaltar que la profundización en este tema sobrepasa los alcances de este proyecto. Por esta razón, se define como opcional el carácter de esté modelo, dado a que existen diversas formas de abarcar el tema [53].

Hoy en día, las técnicas de Inteligencia Artificial están siendo utilizadas para el desarrollo de videojuegos para que aborden el tema de la dificultad, pero, ¿que podría ser considerado como Inteligencia artificial en un videojuego? una definición puede ser “la simulación por computadora de comportamientos inteligentes” [52]. No obstante, es necesario especificar aún más el significado de inteligencia, el cual podría ser tomado como “el comportamiento que exhibe una gran capacidad de adaptación y resolución de problemas”, o “la conducta más próxima a la de los seres humanos” [52]. Aunque los humanos no siempre nos comportemos de una forma brillante, hay cualidades en nuestro comportamiento que nos hacen inteligentes.

En los juegos es muy común que se tome como referencia la primera definición de inteligencia, lo que produce resultados completamente irreales. (Esto se puede ilustrar Figura 4 – Búsqueda**,** en el problema de búsqueda de ruta del punto A al Punto B evitando los obstáculos). Existen muchos algoritmos que dan una solución correcta a este problema, analizando el mapa y trazando un camino que conecte ambos puntos y además evite los obstáculos, y algunos hasta determinaran el camino más corto. Pero, esta actitud es irrealista al compararla con los seres humanos, ya que estos no trazan rutas óptimas y a menudo cometen errores muy complejos recorriendo laberintos. Por ejemplo un humano podría ir desde A hasta B siguiendo la trayectoria sólida (no esquivaría los obstáculos sino que se desplazaría sobre ellos), argumentando que no tenía un conocimiento previo del camino. Pero si un juego utilizara esté método podría ser algo muy inteligente pero no muy realista. Se debe tener en cuenta el mantener un grado razonable de imperfección al momento incorporar la Inteligencia Artificial, de modo que no resulte frustrante para el jugador humano [52].

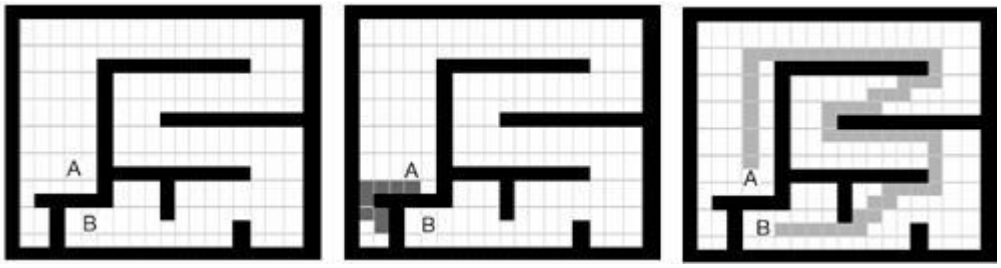


Figura 4 – Búsqueda

Es prudente no olvidar que lo llamativo de un juego son sus retos, y por esta razón los juegos requieren algo de "inteligencia". La "inteligencia" es la que proporciona el grado de dificultad, que causarán una atracción en el jugador [52]. Algo similar ocurren en las series de aventura, en las cuales una vez que el protagonista logra descifrar la estrategia del enemigo y lo derrota, pierde interés en él. Así, la IA busca generar equilibrio entre la conducta que le permita ser a la vez tanto evolucionada como sofisticada, y un comportamiento que sea más o menos humano (en el sentido menos óptimo de la palabra). Existen varias técnicas que pueden ser utilizadas para asegurar que la IA no es sólo "la solución del problema de los robots", sino que además suministra la cantidad correcta de complejidad para desafiar y cautivar al jugador [18].

Estructura de un sistema de IA

Se analiza una entidad IA (Inteligencia Artificial) como puede ser un enemigo en *Quake* o un ejército *Age of Empires II* [52]. La comprensión de los principales bloques de construcción ayudará a la definición posterior de la estructura y el código de sus sistemas [52]. Se pueden plantear dos formas generales de ver la inteligencia artificial de un juego[52]. La primera y la más común es el agente, que es prácticamente un juego de personajes del mundo. Normalmente se trata de enemigos, pero también puede ser personajes no-jugador, compañeros, o cualquier ser animado en el campo. Para este tipo de entidades, debe seguir una estructura biológica, de modo que sea posible modelar estos comportamientos de una manera realista. Si se estructuran estos sistemas de manera similar al cerebro humano, es fácil identificar los siguientes cuatro elementos o normas[52]:

- Sensor o sistema de entrada
- Memoria de trabajo
- Razonamiento y el análisis básico
- Acción o el sistema de producción

Existen casos en los que no es necesario utilizar todos estos elementos dependiendo específicamente de la aplicación, pero es común encontrar el uso de muchos de ellos en las diferentes entidades. La definición de estos elementos permite determinar el enfoque que tomara la entidad IA.

Sensores

Todas las entidades IA deben estar conscientes de su entorno (conocer su mundo) para de esta manera utilizar esta información en el razonamiento o fase de análisis. El sentido y la forma en que este mundo depende en gran medida del tipo de juego que está creando. Para comprender esto, se ejemplificará mediante el análisis para una entidad IA de un personaje del juego Quake [52].

En el juego *Quake*, un individuo enemigo necesita saber [52]:

¿Dónde está el jugador y donde se le busca?
¿Cuál es la geometría de los alrededores?

Por lo tanto, el modelo de este mundo es relativamente sencillo. En ese juego, el sistema visual es un laberinto donde los obstáculos son elementos muy parecidos a los del mundo real (muros, agua, etc). Donde se puede ver al jugador si este se encuentra en el rango de la visión humana, y se utilizan algoritmos de detección de colisiones para establecer los obstáculos. Esta fase es de esencial importancia al momento de recopilar la información que impulsará los análisis posteriores.

Memoria

Almacenar los datos en IA es a menudo complicado, dado que los conceptos que se almacenan no son sencillos. Se almacenan datos como puntos y orientaciones, y se utilizan valores numéricos para describir el si esta dentro del "estado" de la IA. Si el personaje está caminando, es igual a uno; Si él está en corriendo, es igual a dos; así sucesivamente. Ahora, ¿cómo se puede almacenar información más abstracta, como el balance de poder de un punto anterior? Y ¿qué hay acerca en una ruta? ¿Cómo guardar una ruta de manera que recuerde cómo ir del punto A hasta el B? Algunas de estas estructuras de datos son no son triviales [52].

Análisis / Razonamiento básico

El análisis / razonamiento es la parte de la IA en la cual realmente se utilizan los datos obtenidos por medio de los sentidos y la memoria, permitiendo el análisis de la configuración actual y la toma de decisiones. Uno de los métodos más empleados para tales tareas son máquinas de estados finitos y sistemas basados en reglas. La toma de una decisión puede ser lenta o rápida dependiendo del número de alternativas y de los datos sensoriales a evaluar. Por ejemplo, jugar ajedrez es un proceso lento debido a la gran cantidad de movimientos que se pueden realizar en un tablero dada una configuración inicial, mientras que el movimiento de un personaje en Quake es realmente sencillo ya que tiene un número limitado de opciones (por lo general, se desplazan en cuatro direcciones, saltan y disparan) [52].

Afortunadamente, muchos juegos requieren procesos de toma de decisiones para unas pocas opciones, lo que hace su desarrollo más fácil. Es común encontrar una gran cantidad de juegos que tienen algoritmos relativamente simples pero aun así son elogiados por su IA.

Acción / Output System

La inteligencia, por más sofisticada que sea, debe transmitir acciones y respuestas, por esto generalmente se trata de averiguar lo que sucede dentro del cerebro de una criatura virtual. Es esencial asociar las rutinas con las acciones de las subrutinas de IA, para poder tener una sensación de verdadera inteligencia[52]. Es necesario determinar de forma clara cómo se utiliza la inteligencia por el sistema (videojuego), ya que la personalidad de los personajes es transmitida a través de sus comportamientos y es posible que al exagerar demasiado sus acciones resulten obvias. Esto proporciona al jugador una percepción de un mayor grado de inteligencia, cuando en realidad la complejidad real de las rutinas básicas de IA resulta ser baja. Un buen ejemplo sería el juego Súper Mario Bross [52]. En este juego se tiene un mundo lleno de todo tipo de criaturas (tortugas, dragones, entre otras), donde las acciones de todas estas son muy similares. Igualmente, el comportamiento de estas criaturas (enemigos en el juego) era simple: perseguir o repetir el mismo ciclo de movimientos. Sin embargo, mediante la creación de la de la historia de los movimientos de cada uno de ellos, la personalidad y la inteligencia se perciben de una mejor forma.

2.4.4. Modelo de Coordinación

El modelo de coordinación permite profundizar en las interacciones entre los personajes/agentes, que se agrupan en *conversaciones* [17]. Se define una conversación como un conjunto de interacciones iniciadas con el fin de alcanzar un objetivo. Cada interacción entre dos personajes/agentes se realiza mediante el envío de un mensaje que tiene asociado un acto de habla. En una conversación se pueden distinguir los siguientes papeles desempeñados por los agentes: *iniciador,* agente que inicia la conversación; y *participantes*, agentes involucrados en la conversación.

Se pueden identificar dos razones para iniciar una conversación [36]: necesitar ayuda o proporcionar ayuda. De la misma manera, un agente puede responder a una conversación por dos causas [34]: suministrar ayuda o aceptar ayuda. Esta ayuda o petición de cooperación puede ser de dos tipos: intercambio de información o solicitud de realización de una tarea. Si se asume que los agentes no son altruistas, la petición de ayuda implica un coste, y la oferta de servicios será asimilable a la publicidad, donde que los agentes compiten por la atención del usuario [49].
Las conversaciones se pueden tipificar de tres formas:

POR LA FORMA COMO SE ELABORA LA CONVERSACIÓN

- Conversaciones *primitivas:* Se dice que una conversación es primitiva cuando sólo el agente iniciador de la conversación envía un mensaje y recibe (o no) una respuesta a dicho mensaje.

- Conversaciones *complejas*. El resto de conversaciones se denominan complejas, y suelen requerir el conocimiento de un protocolo que rija la conversación.

POR EL NÚMERO DE PARTICIPANTES

- Conversaciones bilaterales: es realizada entre dos agentes.

- Conversaciones multilaterales, entre más de dos agentes.

POR LA FORMA COMO SE INICIAN

- Primarias, si son iniciadas directamente por el agente.

- Secundarias, si se inicia como resultado de otra conversación.

El modelo de coordinación se estructura en torno a los conceptos de agente/personaje y tareas (casos de uso) e introduce los siguientes conceptos:

- *Mensaje:* estructura de datos que intercambian los agentes para comunicarse.

- *Acto de habla:* intención del emisor del mensaje al transmitir el contenido del mensaje.

- *Servicio:* prestación realizada por un agente para satisfacer las necesidades de otro agente.

- *Intervención:* intercambio de un mensaje entre un agente y otro agente. Puede haber tres tipos de interacciones básicas: síncrona (bloqueante hasta recibir respuesta), asíncrona (sin respuesta) y diferida (envío no bloqueante y recepción de respuesta).

- *Conversación:* conjunto de interacciones cuyo fin es la realización de un servicio. Se distinguen dos tipos de conversaciones: conversaciones de información y conversaciones de servicio. Las conversaciones de información tienen como objetivo informar a un agente de algún hecho u obtener una determinada información (consulta). Las conversaciones de servicio tienen como objetivo la realización de un servicio.

- *Protocolo:* normas asociadas a las conversaciones. Un protocolo determina las interacciones que se llevan a cabo en una conversación.

2.4.4.1. Conversación

Las conversaciones constituyen el elemento central del modelo de coordinación. La existencia las demás entidades de este modelo depende de la existencia de una conversación a la que están ligadas. Una conversación es un conjunto de interacciones cuyo fin es la consecución de un objetivo. Si un agente es capaz de mantener varias conversaciones simultáneas, creará un ejemplar de la entidad conversación para cada conversación activa. Si sólo es capaz de mantener una conversación, sólo podrá tener un ejemplar de conversación activo. En la Tabla 10 – Conversación se propone una plantilla que permite una descripción de cada conversación.

CONVERSACION	
ID Mensaje	
Per-formativa	
Agente Emisor	
Agente Receptor	
Agente a quien se enviaría Respuesta	
Contenido: texto a enviar	
Lenguaje	
Codificación	
Ontología	

Tabla 10 – Conversación

Los elementos presentados en la Tabla 10 – Conversación se describen a continuación:

- **ID Mensaje**: Identificador único para los mensajes.
- **Per-formativa**: Tipo de mensaje que se está enviando.
- **Agente Emisor**: Nombre del agente que emite el mensaje.
- **Agente Receptor**: Nombre del agente que recibe el mensaje.
- **Agente a quien se enviaría Respuesta**: Nombre del agente a quien se debe responder.
- **Contenido**: Describe la información que se va a enviar.
- **Lenguaje**: Nombre del sistema en que se encuentra expresado el mensaje.
- **Codificación**: Define el tipo de codificación usado en este mensaje.
- **Ontología**: Define el esquema conceptual de los agentes.

Para realizar las interacciones entre personajes/agentes, se debe tener en cuenta la comunicación entre ellos, ya que es la clave para obtener todo el potencial del paradigma de agentes [54]. Los agentes emplean un lenguaje de comunicación (Agent Communication Language, ACL) para comunicar información o conocimiento y de esta manera realizar tareas que un solo agente no podría realizar [54].

2.4.4.2. Notación Gráfica

Para establecer una conversación entre agentes es necesario definir previamente el protocolo que se va a seguir durante la conversación. Un protocolo de interacción es una descripción detallada del tipo y orden de los mensajes involucrados en una conversación entre agentes [54]. Un agente puede participar simultáneamente en múltiples diálogos con diferentes agentes y con diferentes protocolos de interacción. En este caso se ha escogido el protocolo FIPA Contract Net, teniendo en cuenta que es considerado un estándar de comunicación de agentes [54]. Las siguientes imágenes presentan el formato básico de comunicación entre los personajes usando este protocolo (Figura 5 – Protocolo de comunicación entre personajes y Figura 6 – Protocolo de comunicación entre personajes y jugadores).

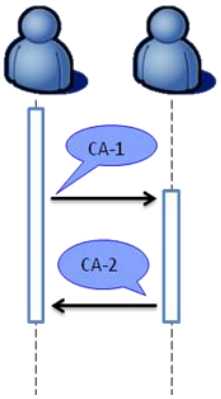


Figura 5 – Protocolo de comunicación entre personajes

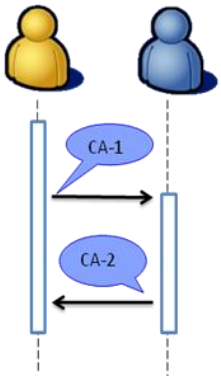


Figura 6 – Protocolo de comunicación entre personajes y jugadores

Las siguientes imágenes muestran los diferentes hilos de interacción que se pueden presentar entre los agentes usando el protocolo FIPA Contract Net: Figura 7 – Mensajes concurrentes, Figura 8 – Mensajes con decisión inclusivo y Figura 9 – Mensajes con decisión exclusivos:

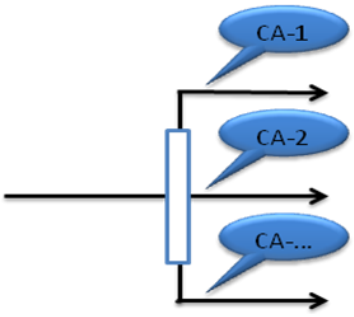


Figura 7 – Mensajes concurrentes

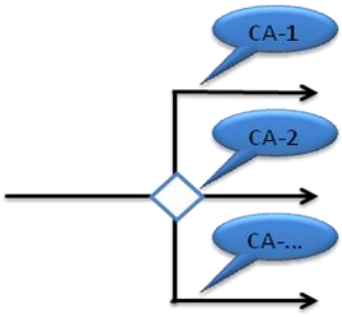


Figura 8 – Mensajes con decisión inclusivo

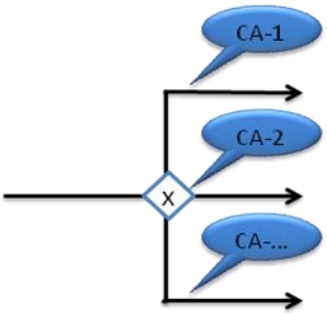


Figura 9 – Mensajes con decisión exclusivos

2.4.5. Modelo de Organización

El modelo de organización permite el modelado de las relaciones estructurales entre personajes/agentes. Este modelo describe tanto la organización del jugador como la sociedad de personajes/agentes, definiendo sus relaciones estáticas existentes. Para ello es necesario tener una clara definición de la relación de herencia entre agentes. En los modelos orientados a objetos basados en clases, la relación de herencia significa que la clase derivada hereda los atributos y métodos de la clase padre. Pero el caso de estudio actual está enfocado hacia los personajes/agentes es necesario definir la relación de herencia entre clases de agentes, entendiendo que una clase agente es la generalización de un conjunto de agentes con la misma arquitectura, en la cual se definen los servicios (tanto los ofrecidos como los requeridos), las habilidades, los sensores y actuadores, creencias y objetivos compartidos por este tipo de agentes. La relación de herencia se puede evidenciar cuando se tiene un conjunto de agentes/personajes los cuales comparten objetivos, creencias, planes y servicios propios de una misma clase, pero es necesario que a un agente/personaje en especial se le puedan agregar objetivos, creencias, planes y servicios particulares, basados en la historia.

A continuación se recomienda un conjunto de situaciones en las cuales es necesario desarrollar un ejemplar del modelo de organización, debido a las sociedades agente/personaje [31]:

- Cuando sean identificados varios agentes/personajes con características en común, de forma que sea posible determinar las relaciones de herencia.

- Cuando los agentes/personaje interactúan con el entorno, de manera que sea posible modelar los objetos del entorno y su interacción con los agentes/personajes.

- Cuando sean encontrados agentes/personajes en los cuales se pueden determinar una relación de autoridad, que simbolice poder o estructura entre los agentes/personajes, esta relación debe hallase reflejada en el modelo.

- Cuando sean identificados grupos o coaliciones de agentes/personajes o de clases agente/personaje, la estructura y relaciones de estos grupos debe mostrarse reflejada en el modelo.

- Cuando sean encontradas relaciones entre agentes/personajes y agente/jugador es preciso modelar dicha relación.

2.4.5.1. Notación Gráfica

Se usa la misma notación gráfica propuesta por MAS-CommonKADS para mostrar las relaciones entre los agentes como lo muestra la Figura 10 – Notación Gráfica del Modelo de Organización. Las asociaciones entre los agentes y otros son empleadas para describir relaciones de autoridad o poder. La relación de grupo se realiza encerrando a varios agentes en una caja o mediante la creación de una agregación y colocando los elementos del agente que representa al grupo.

En la representación gráfica del agente se muestra la parte interna o estado mental, y/o la parte externa, que es la relación con el entorno y los demás agentes. Por último, otros objetos relevantes del problema deben ser mostrados en el diagrama usando la notación de OMT [36].

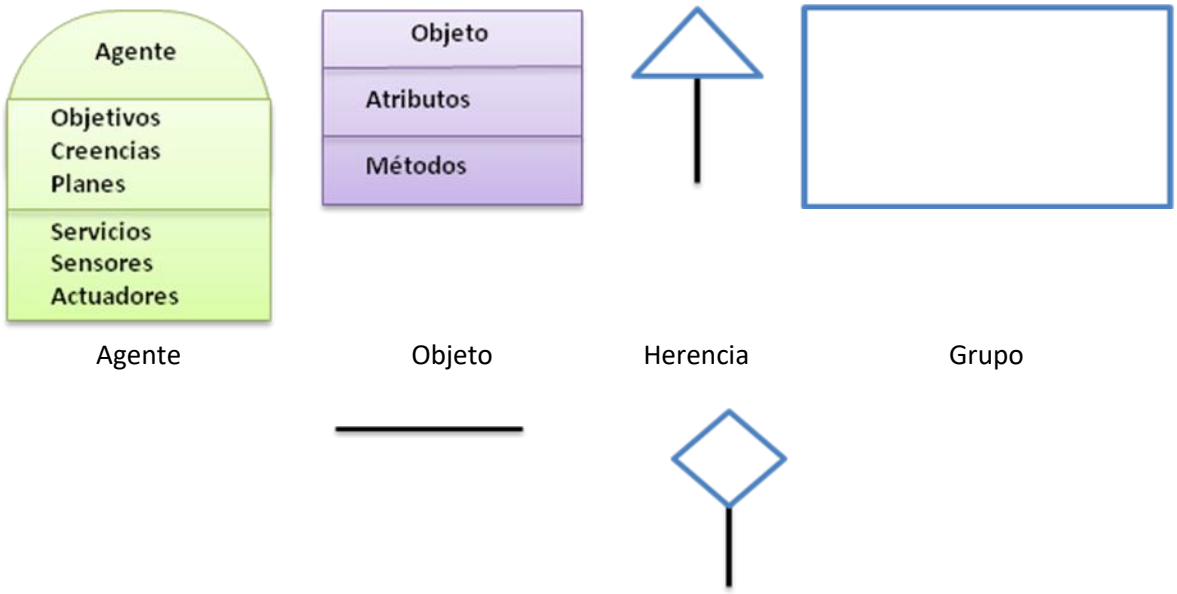


Figura 10 – Notación Gráfica del Modelo de Organización

2.4.6. Modelo de Diseño

En el presente numeral se presenta el modelo de diseño el cual tiene como objetivo principal la descripción de los componentes de los distintos modelos de la etapa de análisis de forma tal que se facilite su definición en un lenguaje de programación.

Para el modelo de diseño de MAS-CommonKADS existen tres clases de decisiones de diseño: diseño de la red, el cual permite a los agentes tener una concepción uniforme de la red, diseño de agentes, donde los agentes son descompuestos en subsistemas, y el diseño de la plataforma, donde se recogen las decisiones de hardware y software, documentando decisiones de bajo nivel sobre el lenguaje de implementación seleccionado. Para este proyecto no se usa la decisión de diseño de la red, teniendo en cuenta que para los Juegos de aventura el uso de la red (multi-jugador) se hace extremadamente complejo debido a la estrecha dependencia entre el juego y la historia, donde el Personaje/Jugador debe cumplir ciertos objetivos definidos de tal forma que el juego se siga según el curso definido en la historia [16].

2.4.6.1. Diseño de Agentes

En este ítem se documentan las distintas funciones de cada módulo que deben ser implementadas según la arquitectura seleccionada para el agente. Para llevar a cabo esta tarea será necesario el uso de la Tabla 11 – Componentes de diseño del agente, definida a partir de los principales característica de diseño de agentes considerados en la metodología Mas-CommonKADS:

- **Sistema-Personaje/Agente:** El sistema agente representa un agente que va a ser implementado y está siendo diseñado. Puede corresponderse con uno o varios ejemplares del modelo de agente.

SISTEMA PERSONAJE/AGENTE	
Id:	
Nombre del Personaje/Agente:	
Arquitectura:	
Lenguaje:	
Subsistemas:	
Referencias Cruzadas	

Tabla 11 – Componentes de diseño del agente

Los elementos presentados en la Tabla 11 – Componentes de diseño del agente se describen a continuación:

  - **Nombre:** Se define el nombre del agente, haciendo alusión al ejemplar del modelo de agente desarrollado.
  - **Arquitectura:** Tipo de arquitectura del agente.
  - **Lenguaje:** Nombre del lenguaje de descripción de agente empleado, por ejemplo ADL.
  - **Subsistema:** Subsistemas asignados al agente.

- **Subsistema**: Subsistema o módulo de un agente, se propone la plantilla de la Tabla 12 – Componentes de diseño del subsistema .

SUBSISTEMA PERSONAJE/AGENTE	
Id:	
Nombre del Subsistema:	
Funcionalidades:	
Implementa:	
Diseño-Detallado:	
Referencias Cruzadas:	

Tabla 12 – Componentes de diseño del subsistema

Los elementos presentados en la Tabla 12 – Componentes de diseño del subsistema se describen a continuación:

- **Nombre:** Nombre del subsistema que se está diseñando.
- **Tipo:** Tipo de subsistema. Se distinguen varios tipos: de interacción con el usuario (derivado del modelo de comunicación), de comunicación con el resto de agentes (derivado del modelo de coordinación), de interacción con el exterior (derivado del modelo de agente, tarea y coordinación), de ejecución de tareas (derivado del modelo de tareas y agente), de planificación y reacción (agrupa los planes para alcanzar los objetivos asignado y responder a los estímulos recibidos y enviados en el módulo de interacción con el exterior). Estos módulos son muy dependientes de la arquitectura de agente seleccionada.
- **Funcionalidades:** Descripción de las funcionalidades implementadas.
- **Implementa**: Tareas implementadas (del modelo de tareas).
- **Diseño-Detallado**: Nombre de la entidad Diseño-Detallado en que se describe el subsistema.

- **Diseño detallado subsistema** Descripción en un lenguaje de diseño detallado de un subsistema de un Personaje/Agente, se propone la plantilla de la Tabla 13 –Diseño detallado.

DISEÑO DETALLADO DEL SUBSISTEMA PERSONAJE/AGENTE	
Id:	
Nombre del Subsistema:	
Arquitectura:	
Lenguaje:	
Descripción:	
Referencias Cruzadas:	

Tabla 13 –Diseño detallado

Los elementos presentados en la Tabla 13 –Diseño detallado se describen a continuación:

- **Nombre**: Nombre del subsistema que se está diseñando.
- **Lenguaje:** Lenguaje de diseño detallado empleado.
- **Descripción**: Referencia a la descripción del subsistema empleando el lenguaje de diseño detallado.

2.4.6.2. Diseño de Plataforma

El propósito del diseño de la plataforma es permitir la documentación de las decisiones de bajo nivel acera del lenguaje de implementación seleccionado, el software y el hardware empleado y los usuarios finales del sistema. En la Tabla 14 – Plantilla: Plataforma, se plasman los principales componentes del diseño de la plataforma considerados en la metodología Mas-CommonKADS:

PLATAFORMA	
Id:	
Nombre de la Plataforma:	
Descripción:	
Lenguaje:	
Hardware:	
Software:	
Usuario:	
Referencias Cruzadas:	

Tabla 14 – Plantilla: Plataforma

Los elementos presentados en la Tabla 14 – Plantilla: Plataforma se describen a continuación:

- **Plataforma:** La plataforma recoge todas las decisiones del entorno de implementación y operación del sistema multi-agente.
  - **Nombre:** Nombre de la plataforma o sistema en que se está diseñando.
  - **Descripción:** Justificación de las decisiones tomadas.
  - **Lenguaje:** Nombre de los lenguajes de implementación empleados.
  - **Hardware Requerido:** Requisitos del hardware en que se desarrolla el sistema (tipos de máquinas, conexiones, etc.).
  - **Software Requerido:** Requisitos de configuración software en que se desarrolla la plataforma (sistema operativo, nombres de las aplicaciones, versiones, etc.).
  - **Usuario:** Usuarios de la plataforma, extraídos de los modelos de agente de la organización.

Se debe tener en cuenta que MAS-CommonKADS precisan la existencia de dos grandes niveles para la abstracción del diseño de sistemas multi-agentes, que son:

- *Nivel de programación de agente:* El cual tiene como objetivo la definición de la forma detallada posible de la conducta de cada uno de los componentes de la arquitectura del agente mediante el uso de un lenguaje de programación de agentes.

- *Nivel de descripción de agente:* En este nivel se utiliza un lenguaje el cual describe los componentes del agente pero sin especificar como son realizados, debido a que esta labor se realiza al momento de implementar.

A continuación se presentan un conjunto de actividades que deberán ser consideradas para la creación del modelo de diseño. Se debe realizar un diseño de los Personajes/Agentes, definiendo la arquitectura adecuada para cada uno, y por último el diseño de la plataforma de los agentes.

Pasos para el diseño de Personajes/Agentes

- *Determinación de la arquitectura de los agentes:* El objetivo de este paso consiste en establecer la arquitectura adecuada para cada Personaje/Agente. Luego se realiza la descripción de la arquitectura del agente, en sus respectivas plantillas. Aun cuando no se ha definido una forma de determinar que cual arquitectura es la mejor para cada agente, existen varios aspectos que podrían ser de mucha ayuda en esta labor, como:

    - Aspectos derivados de las plataformas disponibles para la arquitectura: los recursos o normas de la organización pueden requerir que los agentes sean programados en un lenguaje o con unos protocolos determinados [36].

    - Necesidad de deliberación y reacción: los agentes con capacidades de razonamiento y/o reacción frente al entorno deberán ser modelados con una arquitectura que facilite estas características. La cuestión de utilizar una arquitectura puramente reactiva, puramente deliberativa, o una híbrida, puede basarse en la experiencia de los desarrolladores [36].

    - Necesidad de paralelismo en la ejecución de acciones: un aspecto relevante para la selección de una arquitectura es si un agente queda bloqueado o no al realizar una tarea [36].

    - Problemas de tiempo real: también se debe tener en cuenta si se requiere que se garantice un tiempo de ejecución para las tareas realizadas por el agente [36].

    - Necesidad de movilidad: otro aspecto a tener en cuenta es la necesidad de movilidad de los agentes para trabajar con recursos de otras máquinas [36].

- *Diseño detallado de los módulos:* Con este paso lo que busca es una visión detallada y específica de cada Personaje/Agente y de los principales módulos. Es muy común el uso de un lenguaje de agentes para su especificación.

Pasos para el diseño de la plataforma

Basándose en el diseño, la arquitectura de agente seleccionada y en los requisitos no funcionales, se lleva a cabo la selección del sistema operativo y de la plataforma software y hardware. Las tres decisiones están relacionadas, por lo que no se impone un orden entre ellas.

- *Selección del lenguaje de implementación:* En esta paso se documentan las razones por las cuales se selecciona determinado lenguaje de implementación. Basados en un diseño validado de agentes. Existen varias razones determinantes al momento de la elección del lenguaje como lo son:

    - Política general de la organización, recursos disponibles, interoperabilidad con otros sistemas de la organización y previsión de su desarrollo en años posteriores[36].

- Experiencia de los programadores [36].

- Disponibilidad de plataformas multiagente para la arquitectura de agente deseada en ese lenguaje [36].

- *Selección de los recursos software:* El objetivo de este paso es la documentación de la selección de recursos software. La selección de recursos software está basada en el estudio de las herramientas disponibles en la organización o si existe la necesidad de adquirir nuevas para realizar los agentes. De cada herramienta se debe indicar el nombre, requisitos software y hardware, la versión y las condiciones de operación.

- *Selección de los recursos hardware:* En esta actividad se centra en documentar el hardware necesario para ejecutar el sistema multiagente desarrollado. Deberá documentarse qué hardware es empleado para ejecutar el sistema multiagente y en qué plataformas hardware se puede ejecutar el sistema desarrollado.

CAPITULO III
APLICACIÓN DE LA METODOLOGÍA PARA LA CONSTRUCCIÓN DE PERSONAJES EN UN VIDEOJUEGO DE AVENTURA EN 3D

Este capítulo contiene el desarrollo y la documentación de las actividades definidas en las etapas de Conceptualización, análisis y diseño del segundo Nivel del videojuego "Simón & la Tuberculosis".

3.1. Conceptualización

En esta fase se definen los fundamentos argumentales del videojuego, creando documentos como la historia o el storyline. Es importante señalar que para el desarrollo del Item 3.1. y todos sus subItem se contó con el apoyo del estudiante del programa de diseño grafico de la Universidad del Cauca Alvaro Bacca y forma parte de su proyecto de grado [42]. Además, dada la extensión del presente capítulo el contenido completo de la aplicación de la metodología se consignó en el Anexo C, conservando en capítulo a la ejemplificación de la aplicación de la metodología en el videojuego de aventura "Simón y la Tuberculosis".

3.1.1. StoryLine – Simón y la tuberculosis – Nivel 2

Todo comienza en un pequeño pueblo, San José de las piedras, donde vive Simón nuestro protagonista; un niño muy especial con un destino increíble, aun desconocido para él. San José de las piedras ha sido escenario de importantes eventos pero el que compete a esta historia es uno en especial. Y este se centra en la magia dentro del corazón de un niño y en como su pericia sin igual, le otorgan la capacidad de cambiar el rumbo de toda la gente de su pueblo e indiscutiblemente la de muchas más personas en un futuro.

Hace 30 años hubo una gran batalla, entre un invasor desconocido y todos los pueblos de la región, siendo San José el último bastión de los habitantes de estos pueblos y en el único lugar donde lograron quedar algunos sobrevivientes ahora esclavos, bajo el reinado de aquel desconocido tirano.

Las fuerzas conjuntas de toda la región lucharon por largos 20 años hasta quedar totalmente reducidos, quedando pueblos más pequeños alrededor totalmente destruidos reducidos a cenizas. Todo comenzó por la ambición y maldad de un extraño rey invasor, el cual usaba unas técnicas

que siempre fueron muy peculiares y extrañas, inexplicablemente a su paso imponía ciertos cambios en el espacio, y mucha gente caía enferma y en su mayoría, obligados a persistir en prácticas que empeoraban su salud hasta la muerte.

Así han pasado 10 años, desde que el invasor ganó la guerra y se autoproclamó rey. La única explicación de que no terminó con la vida de todas las personas es que de alguna manera también necesita de ellas y para él es divertido verlas enfermas. En este pueblo, el único reducto de resistencia queda en una casita con un gran sótano, donde una humilde familia intenta sin muy buenos resultados salvar a enfermos graves de la extraña enfermedad. Simón es el menor de la casa y junto a su mamá y a sus abuelitos ayuda en esta laboriosa misión. Un día su mamá, por un descuido se vió infectada por el peligroso virus y todos en su casa se asustaron mucho porque no había mayor modo de ayudarla y sin su ayuda tampoco podría ayudarse a otro montón de personas. La situación empeoró cuando simón por querer ayudar ingenuamente le dijo a alguien extraño que necesitaban ayuda y así quedaron al descubierto, el hombre resultó ser un espía del rey invasor y al descubrirlos este último se puso furioso.

El rey invasor determinó desalojar a todos los enfermos de la casa; frente a tanto dolor Simón no supo qué hacer, más que salir corriendo a la tumba de su padre y pedir desesperadamente ayuda, sin saber a quién. Cuando de pronto, apareció un gracioso ser mágico a quien le fue otorgado el permiso para intervenir, al escuchar las potentes súplicas del niño. Simón se quedo atónito al ver aquel ser, mientras este le decía: "Simón, he venido a ayudarte porque así se me ha permitido, mi nombre es Orah y mi misión eterna es mantener el tiempo en orden, en este momento no es mucho lo que puedes hacer por las personas pero existe una oportunidad, solo una y si así deseas no importan los riesgos, está en tus manos…"

Simón al escuchar esas palabras se lleno de alegría y aceptó sin vacilar su misión, al cual el ser mágico respondió: "Muy bien valiente jovencito, lo que podemos hacer es viajar en el tiempo y regresar al momento crucial en donde tú puedes ayudar, ni un segundo antes ni un segundo después, cierra los ojos… antes de irnos quiero que sepas algo; vas a encontrar pruebas difíciles en tu camino por cada prueba que resuelvas con éxito estaremos a un paso de descubrir la manera de evitar que todas las personas enfermen y poder liberarnos de ese maléfico mal que nos aqueja, en esta misión no estarás solo, yo te acompañare estando siempre a tu lado, oculto bajo el disfraz, de un perro cachorro."

Así Simón y Orah emprenden su mágico viaje al pasado. Orah guiara del uso de pistas a Simón y le ayudará a interpretar ciertas cosas que él no entienda. Simón se enfrentará por medo de retos con los capitanes del ejército invasor y a medida que se enfrenta a estos recoge unos cristales que todos unidos desaparecen al malo final, así Simón vuelve a su tiempo y se llena de felicidad al ver todo bonito y nadie con la enfermedad. Su felicidad es tal que todo queda en secreto y se dedica a disfrutar cada día de su vida.

Así, descubren completamente la tuberculosis y las personas pueden lograr un mundo mejor[3].

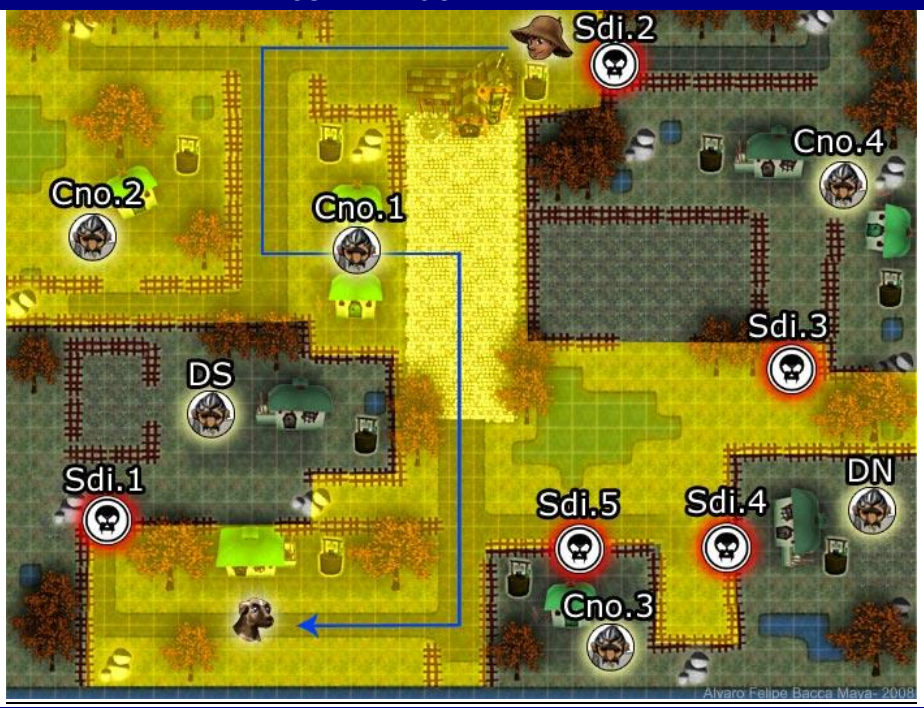
[3] StoryLine desarrollado por el estudiante de diseño grafico Alvaro Felipe Bacca Maya.

3.1.2. Análisis del argumento base

En el caso de los mapas, las zonas amarillas son las zonas donde se desarrolla la *acción*. Y en el caso de los enemigos y amigos los cuales pueden distinguirse por los siguientes íconos:

Figura 11 – Notación de los personajes para el argumento base

ESCENA: Inicio	
CASO:	**1**
CONTROLES:	Funcionan completamente(Con excepción 1.1.C)
PANTALLA:	Simón en el escenario de juego.
NOTAS:	- Los soldados no atacan a Simón en los conectores argumentales, por lo tanto no es un final para el juego. - Sdi.3 se encuentra de espaldas a la zona donde Simón se puede mover.
ACCIONES:	- Simón debe ir hasta su casa para encontrarse con Orah cachorrita.
HITS DE CASO:	**1.1.A.** Orah le envía un mensaje "mágico" a Simón, donde le dice que se dirija hasta su casa. **1.1.B.** Simón debe ir a su casa enviado por el mensaje de Orah. * Dado el caso el jugador puede dar una vuelta por el territorio permitido antes de iniciar el diálogo. **1.1.C.** Simón se encuentra con Orah cachorrita.

<table>
<tr><th colspan="5">ESCENA: Inicio</th></tr>
<tr><td>RESTRICCIONES Y POSICIONES DEL MAPA:</td><td colspan="4"> 
</td></tr>
<tr><td>PERSONAJES DEL MÓDULO:</td><td>- Simón
- Orah</td><td>-Cno.1
-Cno.2</td><td>- Sdi.1
- Sdi.2</td><td>- Sdi.3
- Sdi.4</td></tr>
<tr><td>DIALOGOS FIJOS:</td><td colspan="4">

Caso1.1.A:
Antes de iniciar el recorrido y sin necesidad de que el jugador presione click sobre ella:
1- Orah: Simón!, lo lograste!. Has viajado 25 años en el tiempo hasta la época en que todo comenzó. Necesitas hacer algo antes de continuar: Ve a la casa donde vivirás en un futuro, allí te encontrarás con la amiga que te comenté hace un momento, ella te guiará en tu misión.
2- Orah: Mucha suerte pequeño Simón, recuerda que el futuro de tu gente está en tus manos…
3- Simón: Pero… Orah!? Te volveré a ver?
4- Orah: Más pronto de lo que te imaginas… adiós.
5- Simón: Orah?... mejor me doy prisa.
Caso1.1.B:
Comentarios-pistas que dan los ciudadanos (diálogos libres). De hecho uno de estos debe darle a Simón una nota de inventario.
Caso1.1.C:
6- Orah pequeña: Hola holita hola!!!.... Oh por todos los dioses has llegado!! Tú debes ser... Simón?
7- Simón: Pero… quien eres tú!??! Acaso….
8- Orah pequeña: Ji ji ji, si si si !!! Me presento: Mi nombre es Orah y soy nueva por aquí… no te cuento más de mi porque estoy en una

</td></tr>
</table>

<table>
<tr><th colspan="3">ESCENA: Inicio</th></tr>
<tr><td></td><td colspan="2">misión secreta ya sabes… de todas maneras creo que debes saber más de mí de lo que yo misma sé
9- Simón: Ah… “Vieja amiga”… ahora veo.. en realidad no “tan” vieja!
10- Orah pequeña: De que hablas Simón?.
11- Simón: Oh! No no de nada, es sólo que alguien me dijo que tú eras una buena amiga…
12- Orah pequeña: Aaah!.. quizás estés hablando de aquella perrita parecida a mí que se me apareció anoche!! No te imaginas el susto que tuve… sin embargo no sé… hay algo en ella que me causo mucha confianza, además si es un ser mágico debe ser igual a mi…
13- Orah pequeña: Aunque… no es por nada, pero yo soy más linda y joven! Ji ji ji… aquella estaba viejita y despeinada! Espero nunca ser así cuando grande!!!
14- Simón: Eh?! … mmmm.. no sabes nada… Oye Oritah.. te puedo decir Oritah?
15- Orah pequeña: Si, de ahora en adelante dime Oritah… no me molesta en lo absoluto... que me querías decir?
16- Simón: Mira, necesito tu ayuda, he viajado hasta aquí para intentar detener algo muy malo que va a suceder.
17- Oritah: Aaaa.. SI! Algo me dijo aquella perra señora, no te preocupes creo que me dió todas las indicaciones necesarias así cuenta conmigo!!!
18- Oritah: Para empezar, creo que debemos ir a visitar a Don Segundo (DS)… escuché cuando le decía a un vecino que algo extraño le está sucediendo y que al parecer los guardias de aquel brujo curandero no han podido hacer nada para detener la peste que cada día es más poderosa!
19- Oritah: Don Segundo vive aquí arriba por el camino de la izquierda, sígueme.

• En este momento Oritah sigue el camino y Simón la sigue automáticamente.</td></tr>
<tr><td>DIALOGOS LIBRES:</td><td>Los diálogos “1ª” se refieren a cuando simón se ha acercado por primera vez al personaje no jugador. Y los diálogos “1B” es cuando se acercan por segunda vez a estos mismos.(1.1.B)

1- 1ª- Cno.1: Nunca te había visto por aquí pequeño!, vienes del norte!? Porque si es así llamaré a los guardias del curandero!
1- 1B- Cno.1: Por tu bien no</td><td><u>Sdi.2 // Sdi.3 // Sdi.4</u>

Sí Simón se acerca a cualquier Sdi.* por primera vez o varias veces seguidas con el mismo. (1.1.B):
1- Sdi.*: ¿eeh… Hola niño!.. bonito perrito je, je... Mmm… Lo siento pero el paso por este lugar no está permitido... por lo de la peste tu sabes…
Sí Simón se acerca a un Sdi.* Inmediatamente después de haber</td></tr>
</table>

ESCENA: Inicio		
	dudes del alcalde y de aquel magnífico brujo curandero, están haciendo lo mejor para salvarnos de la peste del norte! 2- **1ª- Cno.2:** Deberías de estar con los otros hombres trabajando en la granja! Ya estás en edad y realmente necesitan más ayuda… como verás ya soy un poco viejo y ya no tengo fuerzas para hacerlo. 3- **1B- Cno.2:** Sabes? El alcalde da algunos permisos firmados para que los forasteros como tu puedan moverse por la ciudad… Tengo uno. Tómalo, de pronto te sirve niño forastero y toma esta nota que encontré, quizás también te sirva. **(1.1.B) Al entregarle las notas a Simón el personaje Cno.2 ya no vuelve a decir el diálogo 1B.**	**hablado con otro(1.1.B):** *2-* **Sdi.*:** Querido pequeñín… como te dijo antes mi amigo, no puedes pasar… Es por efectos del cuidado que debemos tener según las indicaciones del maestro brujo… tu entiendes cierto? **(1.1.B) Sí Simón recoge el permiso que le da el Cno.2 en este caso (el permiso no sirve hasta el caso 1.2).** 3- **Sdi.*:** Lo siento pequeñín pero ese pase que tienes no sirve para seguir por ésta sección del pueblo, muy probablemente te funcione en otra parte. **Sdi.1 No se dirige a Simón en ningún momento pues se encuentra de espaldas a él siempre en este caso (1.1)**

Tabla 15 – Escena: inicio

3.1.3. Definición de reglas

El videojuego tiene un carácter progresivo, en el cual el sistema de reglas puede ser planteado de tal manera que el usuario pueda fácilmente evolucionar a través de las pruebas. En un caso determinado podría proponerse un juego con modelo de "desbloqueo" de pruebas. Por ejemplo, al resolver la prueba número uno, debe habilitarse un menú que ofrezca la posibilidad a los jugadores de elegir la siguiente prueba, entre la número: dos, tres o repetir la número uno. Según sus criterios sin estar ligado a un orden especifico, y una vez superadas estas tres pruebas debe posibilitarse la elección entre las pruebas número: cuatro, cinco y seis del mismo nivel (de manera similar a los juegos mentales o "*braingames"*).

Igualmente, se puede plantear un nivel que puede ser repetido indefinidamente, utilizando pruebas que motiven al jugador a tener el mayor puntaje, puede hacer que el usuario adquiera un

carácter menos crítico frente a las reglas y más propenso a aceptarlas (*Jesper Juul*), esto además le daría más "vida" y justificaría la repetición de las pruebas que le gusten al jugador.

3.1.3.1. Controles

Las entradas deben ser totalmente controlables sólo con el teclado, ya que la complejidad en el manejo de los personajes es relativamente sencillo. Inicialmente, las teclas que se han configurado para el control del personaje de Simón son:

Los controles de desplazamiento: La Figura_12 – Controles de desplazamiento muestra los botones del teclado que permiten controlar los movimientos del Simón sobre el terreno.

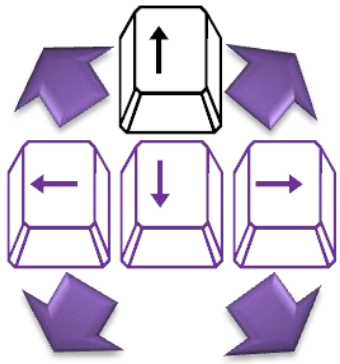


Figura_12 – Controles de desplazamiento

- La tecla Esc: Despliega una ventana de menú, y además mantiene el juego pausado.

Figura_13 – Botón de Esc

Simón no requiere saltar, por tal motivo debe hacer los recorridos evitando obstrucciones en el camino como piedras y/o cercos.

3.1.3.2. Estados del Personaje

- El único personaje jugador es Simón, Su movimiento en todas las direcciones de la pantalla. Tanto horizontal, vertical como diagonal (por ejemplo: *superior + izquierda*).

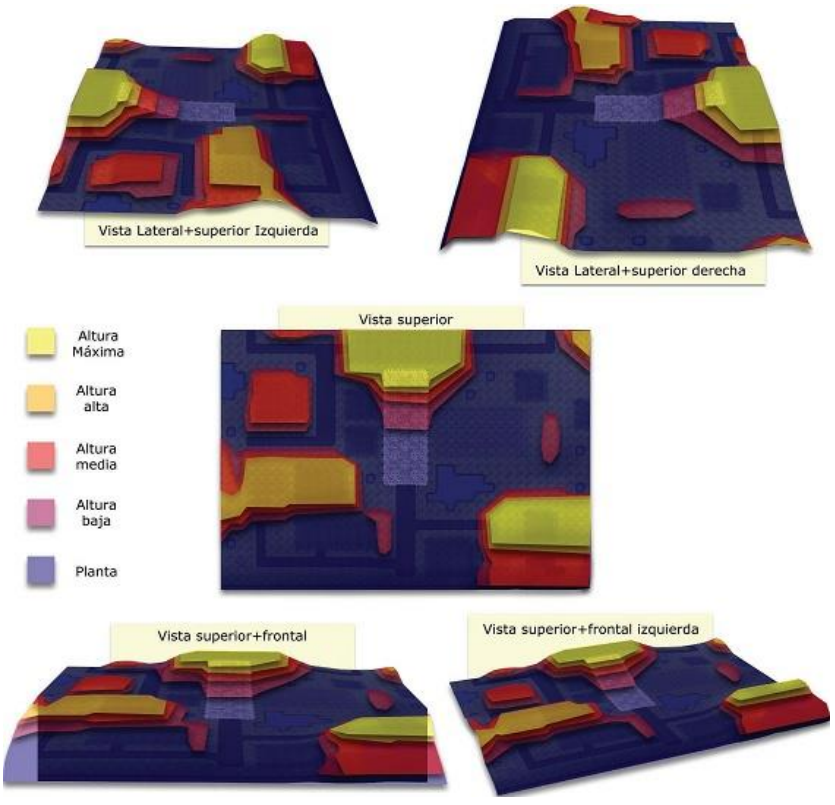
- El movimiento de los diálogos se muestra mientras los diálogos son dispuestos en la pantalla (sin importar el personaje que está hablando) acompañados de una cara ilustrada del personaje que habla.

- El movimiento de "acción" puede ser particularmente útil en el caso recoger un objeto o presionar sobre algún personaje antes de hablar con él y cualquier evento relacionado directamente con *accionar*.

- Simón no requiere saltar, por tal motivo debe hacer los recorridos evitando obstrucciones en el camino como piedras y cercos.

3.1.4. Definición del mundo

Para el videojuego "Simón y la Tuberculosis" – nivel 2 se define un terreno abierto, en el que los sucesos tienen lugar en el pueblo y todos los escenarios planteados se llevan a cabo al aire libre en el pueblo San José de las Piedras. La (Figura_14 – Bocetos del terreno) es un boceto del terreno en el cual se enfatiza su sistema montañoso (alturas), y los diferentes caminos del pueblo.



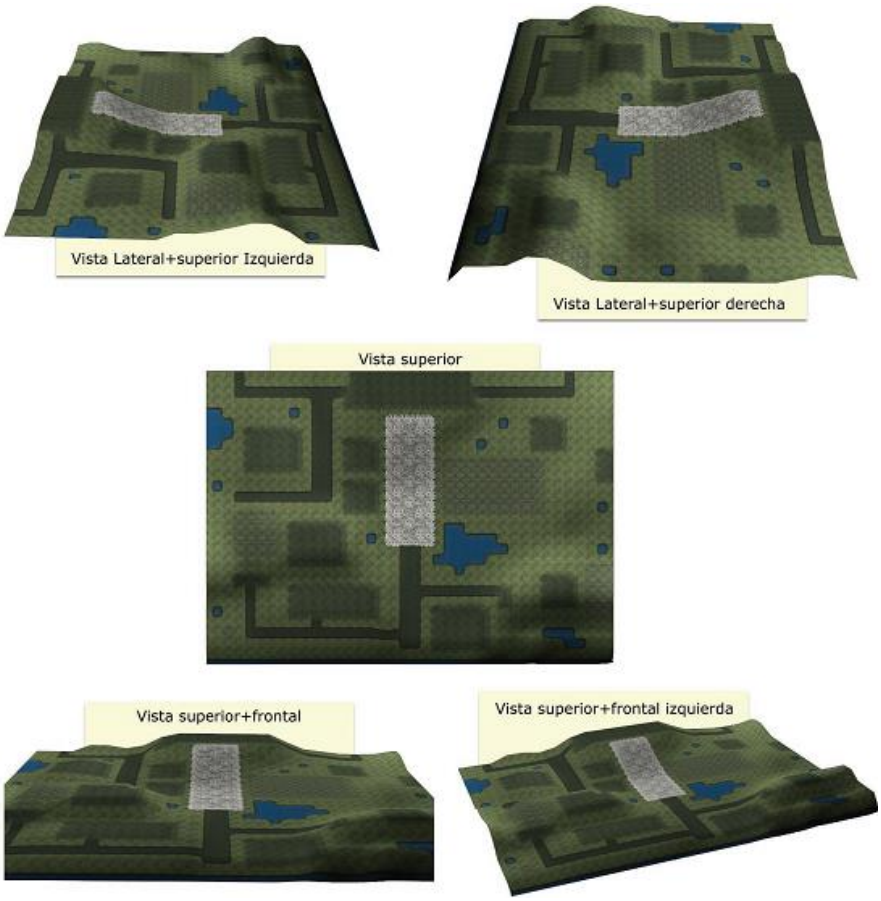


Figura_14 – Bocetos del terreno

El terreno cuenta con los siguientes objetos (estos objetos se encuentran distribuidos en el terreno de la forma en que lo ilustra la Figura_15 – Ubicación Unidades Terreno) que son considerados para el jugador como obstáculos al momento de desplazarse, entre los cuales se tiene:

- Las Casa de campesinos
- El Castillo del alcalde
- Aljibes
- Cercas
- Piedras
- Arboles

Figura 15 – Ubicación Unidades Terreno

3.1.5. Identificación de Personajes

A partir de la comprensión del storyline **"Simón & La Tuberculosis"**, se identificaron los siguientes personajes: Simón, Oritah, Mamá de Simón, Soldados, Campesinos y Don Segundo. Esta selección se realiza con base en la importancia de los personajes en la historia, además se tiene en cuenta que existan relaciones directas entre el personaje y Simón (Personaje jugador). En este ítem a modo de ejemplo se muestra solo la plantilla con la información del personaje "Simón", (los otros personajes identificados se encuentran descritos en el Anexo B de este trabajo).

IDENTIFICACION PERSONAJES/AGENTES	
Id	IDPP01
Nombre	SIMON
Bocetos	
Personajes con los que interactúa:	- ORITAH - CAMPESINOS - SOLDADOS - DON SEGUNDO
Descripción:	Simón es el personaje-jugador de la historia, este personaje es la representación del usuario en el juego (es el personaje que el Usuario controla). Simón realiza un viaje a través del tiempo, para regresar al pasado con la tarea de prevenir una enfermedad que en el futuro está acabando con su familia y los campesinos que habitan su pueblo. Para ello cuenta con la guía de un personaje acompañante el cual está representada en una cachorrita llamada Oritah.
Tipo:	Principal
Referencias cruzadas	Análisis del argumento base

Tabla 16 – Identificación Simón

Además de Simón, en el videojuego entrarán en escena personajes como: Orah, Don Segundo, Campesinos y soldados, los cuales se encuentran descritos en el Anexo B.

3.1.6. Descripción de los Personajes

A continuación se profundiza en los comportamientos y funciones de cada uno de los personajes.

DESCRIPCION PERSONAJES/AGENTES	
ID:	DSPP01
Nombre:	SIMON
Boceto:	
Personajes con los que interactúa:	- ORITAH - CAMPESINOS - SOLDADOS - DON SEGUNDO
Comportamiento:	- Encontrar a Oritah. - Encontrar a Don Segundo. - Obtener la gema.
Dimensiones Física – Social - Psicológica:	Simón es un niño tímido, que aspira poder ayudar a su comunidad en la batalla contra la peste que aqueja su mundo. Así como lo fue su padre, Simón quiere ser un héroe, sin embargo su corta edad no le permite ser gran protagonista, si de guerra física se tratara, su gran ventaja es su inteligencia e ímpetu. Es de pocos amigos pero a través de la historia, descubrirá que es tan valioso en si mismo que dejara atrás la timidez y será un gran líder. Sin embargo ahora su gran curiosidad, pasión e imaginación, sumado con las inmensas ganas de ayudar a su gente, son sus únicas pero valiosas armas para salvar a su pueblo.
Tipo:	Principal
Referencias cruzadas:	StoryLine – Simón y la tuberculosis – Nivel 2 Tabla 16 – Identificación Simón

Tabla 17 – Descripción Simón

3.1.7. Identificación de los casos de uso de los personajes

En este numeral se realizará una breve descripción de las actividades realizadas por los personajes (se muestra como ejemplo la actividad "Aguardar").

CASOS DE USO	
Id:	CU01
Nombre de la Actividad:	Aguardar
Personajes:	Oritah – Aldeanos – Don Segundo - Soldados
Resumen:	Los personajes no jugador mantienen en espera de que Simón se acerque a ellos para ejecutar alguna acción. La acción ejecutada depende del personaje al que se acerque Simón.
Referencias Cruzadas:	Tabla 15 – Escena: inicio

Tabla 18 – Caso de Uso Aguardar

3.1.8. Descripción de los casos de uso

A continuación se describen a fondo cada uno de los atributos de las actividades desarrolladas en los casos de uso (por ejemplo, para la actividad "Aguardar"):

DESCRIPCION CASOS DE USO	
Id:	DCU01
Nombre de la Actividad:	Aguardar
Precondiciones:	Simón debe estar fuera de un perímetro definido a partir del personaje no jugador.
Personajes:	Don Segundo – Oritah – Soldado – Aldeano
Resumen:	Todos los personajes no jugador tienen este comportamiento mientras y es el que les permite estar alerta de cuando se encuentren con Simón.
Tipo:	Primario
Curso normal de los eventos:	1. Al iniciar el juego todos los personajes no jugador deben esperar a encontrarse con Simón para ejecutar sus comportamientos pro-activos.
Excepciones:	
Post-condiciones:	Permite que los personajes no jugador definan sun comportamiento reactivo antes de acivas los comportamientos reactivos.
Referencias Cruzadas:	Tabla 18 – Caso de Uso Aguardar

Tabla 19 – Descripción Caso de Uso Aguardar

3.1.8.1. Representación gráfica de los casos de uso

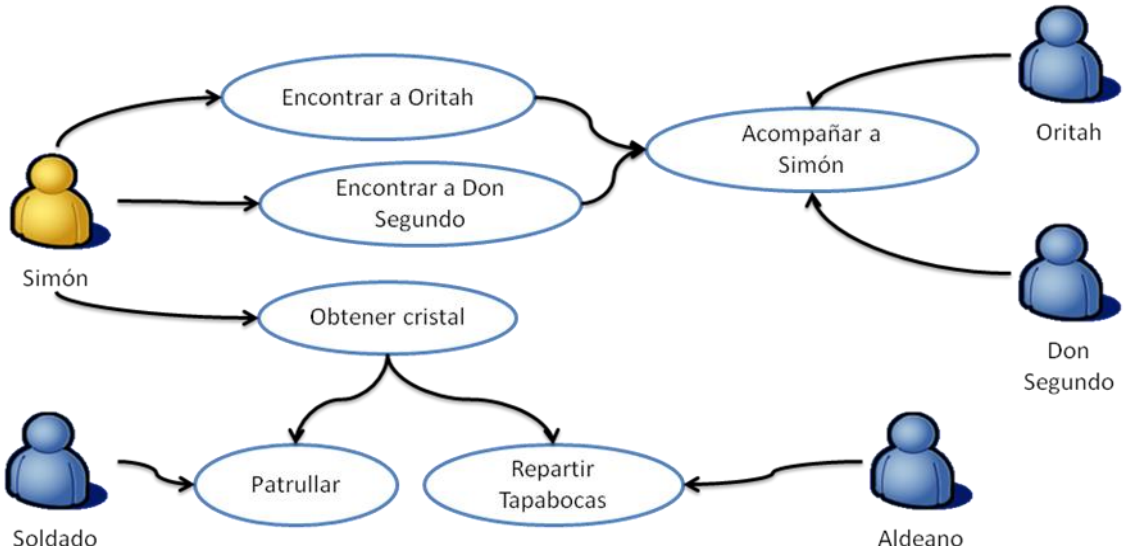


Figura 16 – Diagrama Casos de Usos 1

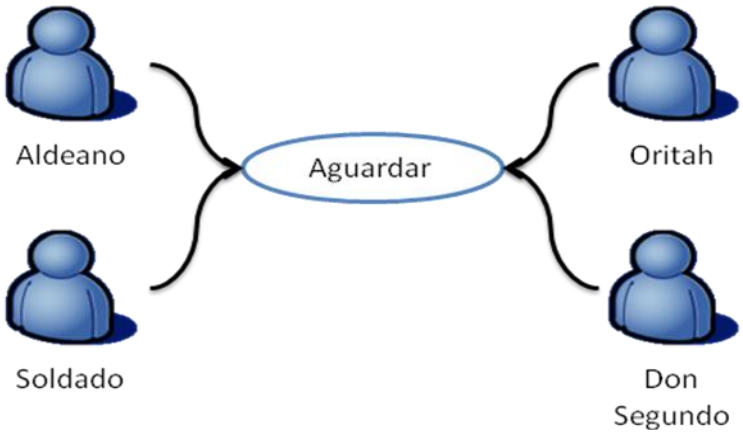


Figura 17 – Diagrama Casos de Uso 2

3.2. Modelo de Comportamiento

COMPORTAMIENTO	
Id:	CMP01
Comportamiento:	Aguardar
Caso de uso:	CASO DE USO AGUARDAR
Tipo:	Proactivo
Referencias Cruzadas:	Tabla 19 – Descripción Caso de Uso Aguardar

Tabla 20 – Comportamiento Aguardar

3.2.1. Notación gráfica

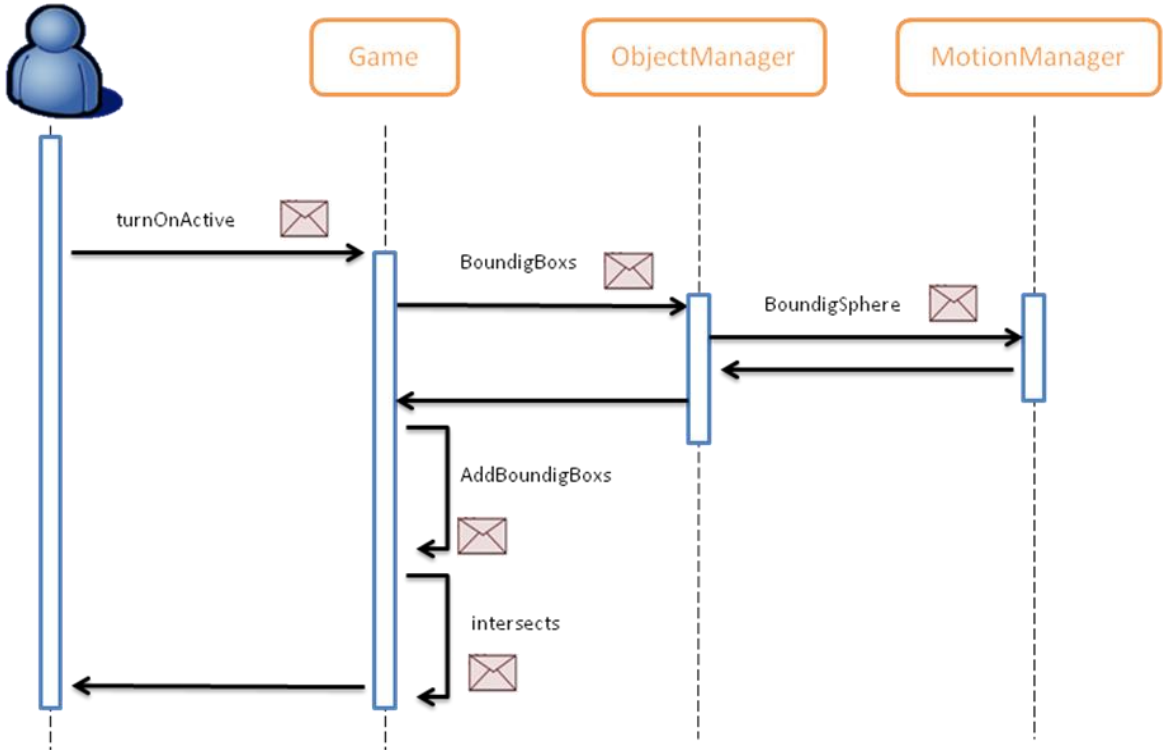


Figura 18 – Diagrama del comportamiento aguardar

3.3. Modelo de Coordinación

MENSAJE	
ID Mensaje	MSJ01
Per-formativa	Acept_Proposal
Agente Emisor	Orah
Agente Receptor	Simón
Agente a quien se enviaría Respuesta	Orah
Contenido: texto a enviar	Simón!, lo lograste!. Has viajado 25 años en el tiempo hasta la época en que todo comenzó. Necesitas hacer algo antes de continuar. Ve a la casa donde vivirás en un futuro, allí te encontrarás con la amiga que te comenté hace un momento, ella te guiará en tu misión.
Lenguaje	Español
Codificación	lenguaje natural
Ontología	No aplica. Porque se encuentra codificada en lenguaje natural.

Tabla 21 – MSJ01

MENSAJE	
ID Mensaje	MSJ02
Per-formativa	Inform
Agente Emisor	Orah
Agente Receptor	Simón
Agente a quien se enviaría Respuesta	Orah
Contenido: texto a enviar	Mucha suerte pequeño Simón, recuerda que el futuro de tu gente está en tus manos…
Lenguaje	Español
Codificación	lenguaje natural
Ontología	No aplica. Porque se encuentra codificada en lenguaje natural.

Tabla 22 – MSJ02

MENSAJE	
ID Mensaje	MSJ03
Per-formativa	Inform_If
Agente Emisor	Simón
Agente Receptor	Orah
Agente a quien se enviaría Respuesta	Simón
Contenido: texto a enviar	Pero… Orah!? Te volveré a ver?
Lenguaje	Español
Codificación	lenguaje natural
Ontología	No aplica. Porque se encuentra codificada en lenguaje natural.

Tabla 23 – MSJ03

MENSAJE	
ID Mensaje	MSJ04
Per-formativa	Agree
Agente Emisor	Simón
Agente Receptor	Orah
Agente a quien se enviaría Respuesta	Simón
Contenido: texto a enviar	Orah?... mejor me doy prisa.
Lenguaje	Español
Codificación	lenguaje natural
Ontología	No aplica. Porque se encuentra codificada en lenguaje natural.

Tabla 24 – MSJ04

MENSAJE	
ID Mensaje	MSJ05
Per-formativa	Query_If
Agente Emisor	Orah
Agente Receptor	Simón
Agente a quien se enviaría Respuesta	Orah
Contenido: texto a enviar	Más pronto de lo que te imaginas… adiós.
Lenguaje	Español
Codificación	lenguaje natural
Ontología	No aplica. Porque se encuentra codificada en lenguaje natural.

Tabla 25 – MSJ05

CONVERSACION	
ID Mensaje	CNV01
Descripción	Esta conversación se lleva a cabo durante la primera escena, se realiza una breve introducción.
Personaje/Agente	**Personaje/Agente**
Orah	Simón
Mensaje	
Tabla 21 – MSJ01 Tabla 22 – MSJ02 Tabla 24 – MSJ04	Tabla 23 – MSJ03 Tabla 25 – MSJ05

Tabla 26 – Conversación Saludo

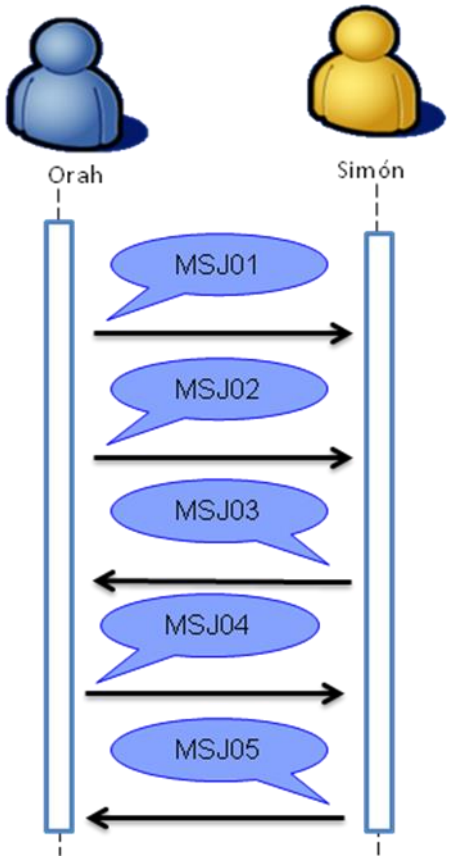


Figura 19 – Diagrama Conversación Saludo

3.4. Modelo de Organización

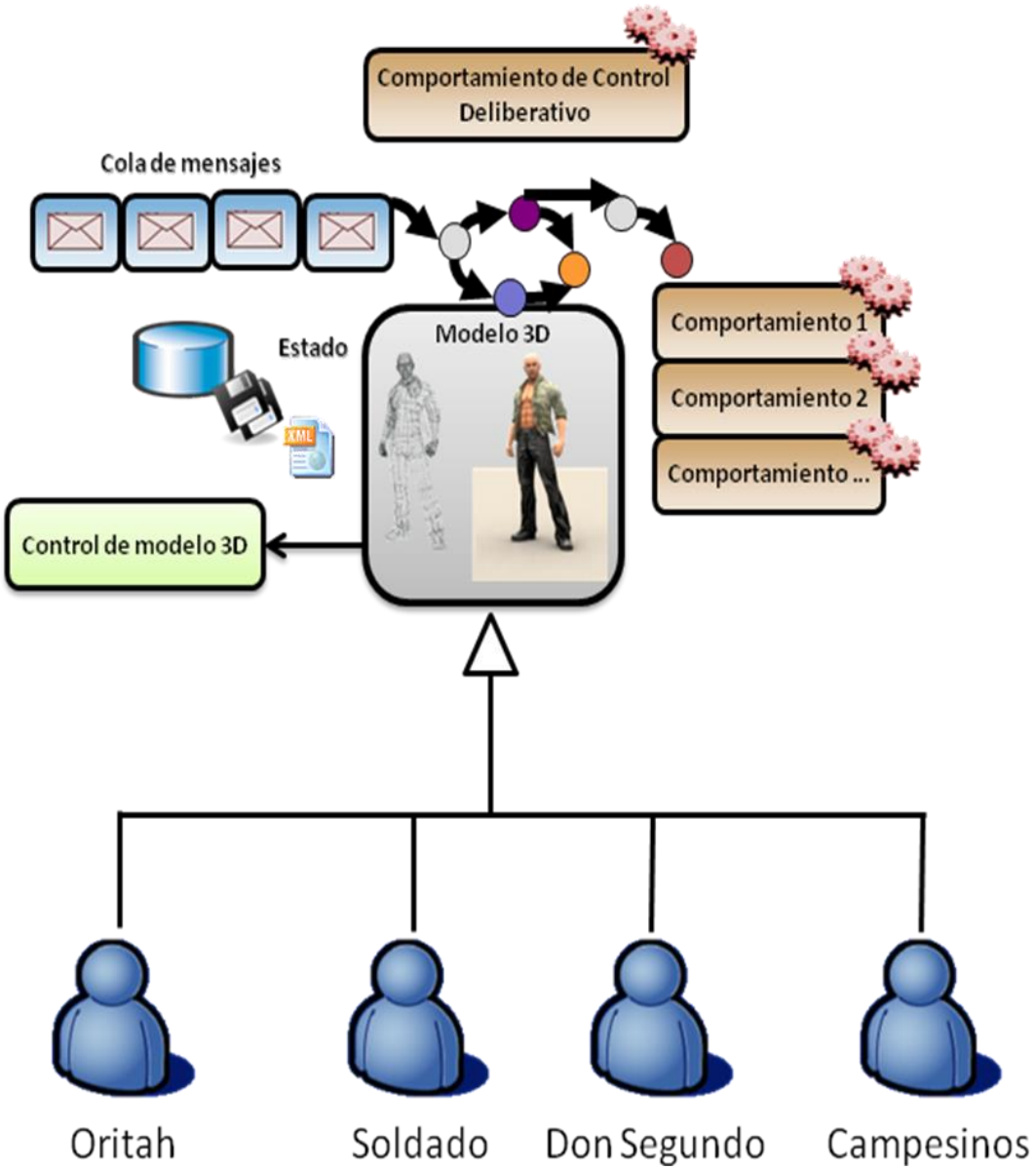


Figura 20 – Modelo de Organización "Simón y La Tuberculosis"

3.5. Modelo de Diseño

En el presente ítem se presenta el modelo de diseño el cual tiene como objetivo principal la descripción de los componentes de los distintos modelos de la etapa de análisis de forma tal que se facilite su definición en un lenguaje de programación.

3.5.1. Diseño de Agentes

SISTEMA PERSONAJE/AGENTE	
Id:	AG001
Nombre del Personaje/Agente:	Orah
Arquitectura	Deliberativo
Lenguaje	C# - XNA
Sub-Sistema	No tiene asociado un Sub-Sistema
Referencias Cruzadas	

Tabla 27 – Arquitectura Orah

3.5.2. Diseño de Plataforma

PLATAFORMA	
Id:	PL001
Nombre de la Plataforma:	Jatt.Agentes
Descripción:	Para la implementación de los agentes se desarrollo una API.
Lenguaje:	C#
Hardware:	• Procesador Pentium a 600MHz, recomendado 1GHz (un gigahertz) • Sistema Operativo Windows 2000 o superior. • 192Mb en Ram, recomandado 256Mb • 1.3Gb de disco duro disponible • Monitor de 800x600 a 256 colores, recomendado 1024x768 a 16 bits.
Software:	Visual Studio C# 2005 Express Editon
Usuario:	Orah – Don Segundo – Soldado - Campesino
Referencias Cruzadas:	

Tabla 28 – Arquitectura de la plataforma

CAPITULO IV DESCRIPCIÓN DEL API PARA IMPLEMENTACIÓN DE AGENTES/PERSONAJES

Este capítulo contiene el desarrollo y la documentación de una interfaz de aplicación programable (API) para dotar de las características propias de los agentes a los personajes de un juego de aventura 3D.

El uso de agentes software se ha extendido por los distintos campos de la ciencias de la computación y cada vez son más los desarrollos que implementan esta técnica, debido a las ventajas sustanciales que ofrecen. Se pueden encontrar desde desarrollos Web hasta aplicaciones de escritorio, con muy variados enfoques [75]. Existen grupos de investigación que se han enfocado al estudio de estos agentes proponiendo metodologías y definiendo modos de implementación; los avances en este campo han sido tan significativos que en el presente se cuenta con protocolos como los presentados por FIPA que mejoran los procesos de desarrollo [55].

Pero uno de los principales limitantes de esta tecnología, es que la mayoría de sus avances puestos públicos se encuentran desarrollados en Java, y las plataformas que los soportan se basan en los estándares de la Foundation for Intelligent Physical Agents (FIPA)[53], lo que dificulta el trabajo al tratar de adicionar nuevas características a los agentes para trabajar con sistemas más específicos. Debido a la naturaleza del proyecto y a su desarrollo usando el framework XNA[58] como librería gráfica, se requería hacer uso de herramientas que permitan controlar el ciclo de vida de los agentes de una forma compatible con el framework de .NET.

Una vez sentadas las restricciones que se tenían para la implementación de la librería se tomó como base el trabajo realizado en “Agentes Inteligentes para adaptar la presentación de contenidos para cursos de educación en línea – STI+AI” [61] , puesto que en este trabajo se desarrolló una librería preliminar para el control del ciclo de vida de agentes. La Figura 21 – Modelo de la librería de agentes usada en STI+AI muestra un esquema del trabajo con la librería mencionada.

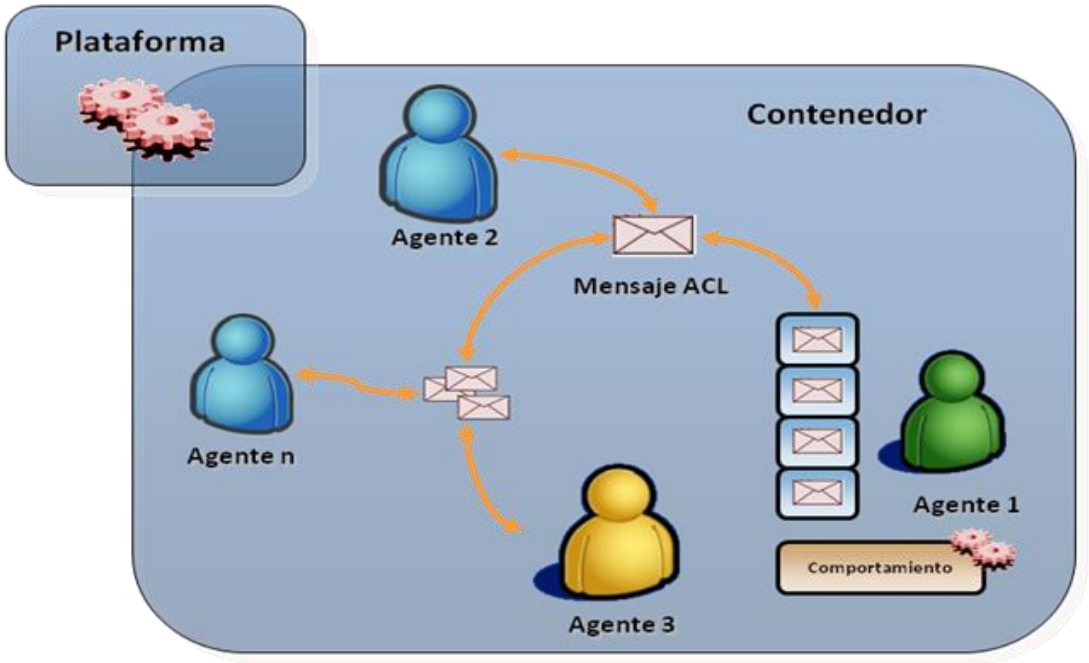


Figura 21 – Modelo de la librería de agentes usada en STI+AI

Esta librería preliminar permite la creación de contenedores, agentes y comportamientos, pero no permite el almacenamiento de estados, ni considera la deliberación para la ejecución de agentes; también carece de un sistema de registro de excepciones que facilite encontrar errores tanto en tiempo de desarrollo, como en tiempo de despliegue. Esta característica es muy importante porque un error ocurrido al interior de una conversación no se ve reflejado directamente sobre la interfaz de usuario debido a que la comunicación se realiza usando mensajes asincrónicos e hilos de programación que ofuscan los errores presentados y que dificultan la detección de inconvenientes

4.1. Adaptación del API

La idea de realizar un API para la construcción de personajes en un juego basada en una librería para la creación de agentes requiere de la adaptación de los elementos genéricos a un esquema particular de personajes para juegos de tipo aventura. Para llevar a cabo este objetivo se modificaron los siguientes elementos:

4.1.1. Modelo base de agente

El modelo base del agente se compone de tres módulos que permiten realizar las siguientes acciones: deliberación interna de un agente para realizar la ejecución de un comportamiento en particular dentro del poll de comportamientos creados, almacenamiento de estado (con persistencia mediante formato XML) y por último un módulo para control de registro de

excepciones cuya notificación puede ser enviada por diversos métodos (e-mail, archivos planos, etc.).

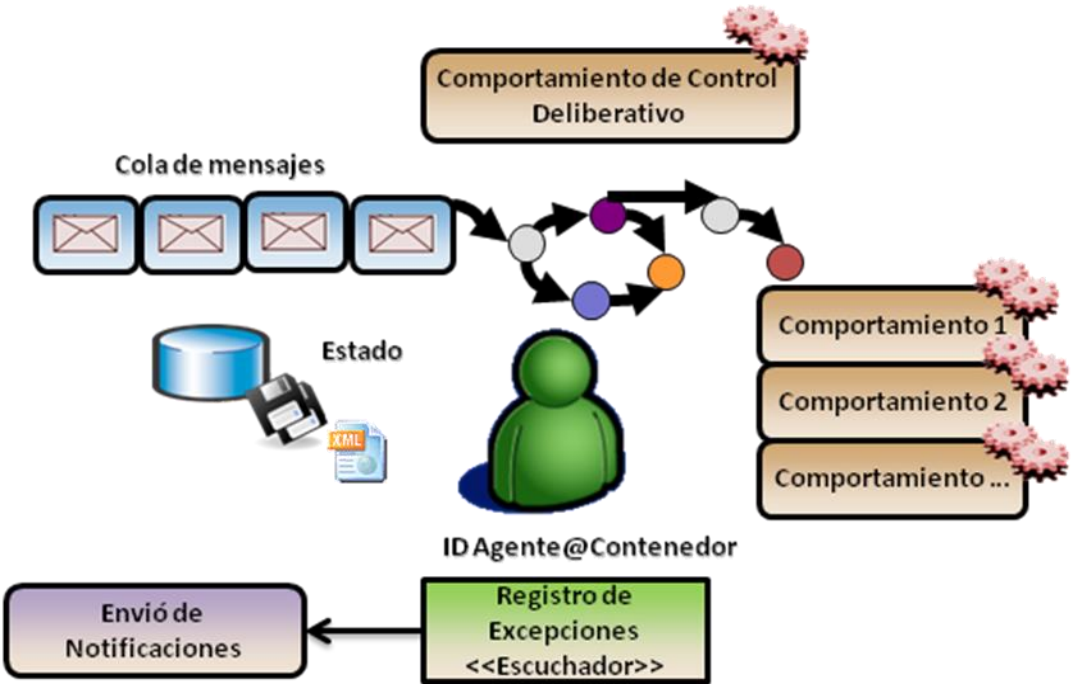


Figura 22 – Modelo de Agente mejorado

La Figura 22 – Modelo de Agente mejorado muestra la implementación mejorada para un agente, el cual contiene:

- **Comportamiento de Control (Nuevo):** Permite implementar un mecanismo de control deliberativo basado en el modelo de experiencia para elegir el comportamiento a ejecutar basado en el estado actual y en la experiencia almacenada en el mismo estado. Es mediante este mecanismo que se implementa la verdadera pro-actividad del agente.

- **Lista de comportamientos:** Posibilita la creación de un conjunto de conductas que ejecuta el agente ya sea de manera proactiva (activadas por el comportamiento de control) o de manera reactiva (activadas por la comunicación o acción de otro agente o el entorno)

- **Cola de mensajes:** Almacena los mensajes enviados por los agentes que se encuentran en el mismo entorno de ejecución. Es importante aclarar que estos mensajes no son iguales a los enviados de un objeto a otro en el sentido estricto de la programación orientada a objetos, sino que constituyen actos de habla y que siguen el estándar y las per-formativas de FIPA.

- **Identificador del agente:** Mediante este atributo los agentes pueden enviar y recibir mensajes mediante el directorio facilitador que se ha implementado en la plataforma, la nomenclatura recomendada para nombrar un agente es **NombreAgente@ContenedorDelAgente.**

- **Estado del agente (Nuevo)**: Este elemento permite almacenar estructuras complejas que representen no solo el estado sino también la experiencia adquirida por el agente dotándolo de mayor inteligencia y dinamismo.

- **Registro de excepciones (Nuevo):** Permite realizar un registro mediante un log de errores, esto facilita no solo el trabajo en la etapa de codificación sino que permite encontrar situaciones inesperadas en la ejecución, lo que redunda en un sistema de agentes con mayor calidad.

- **Envio de notificaciones (Nuevo)**: Posibilita el almacenamiento de las excepciones registradas mediante el módulo anterior pero desacopla el sistema de almacenamiento usando, lo que permite enviarlas a través de e-mail o registrarlas en un disco, etc.

4.1.2. Modelo base del personaje

Una vez realizadas las mejoras en la clase base del agente se debía crear un componente que no solo representara cualquier tipo de agente, sino un personaje para los videojuegos, es así como nace la clase Personaje, que tiene una relación de herencia directa con el modelo de agente.

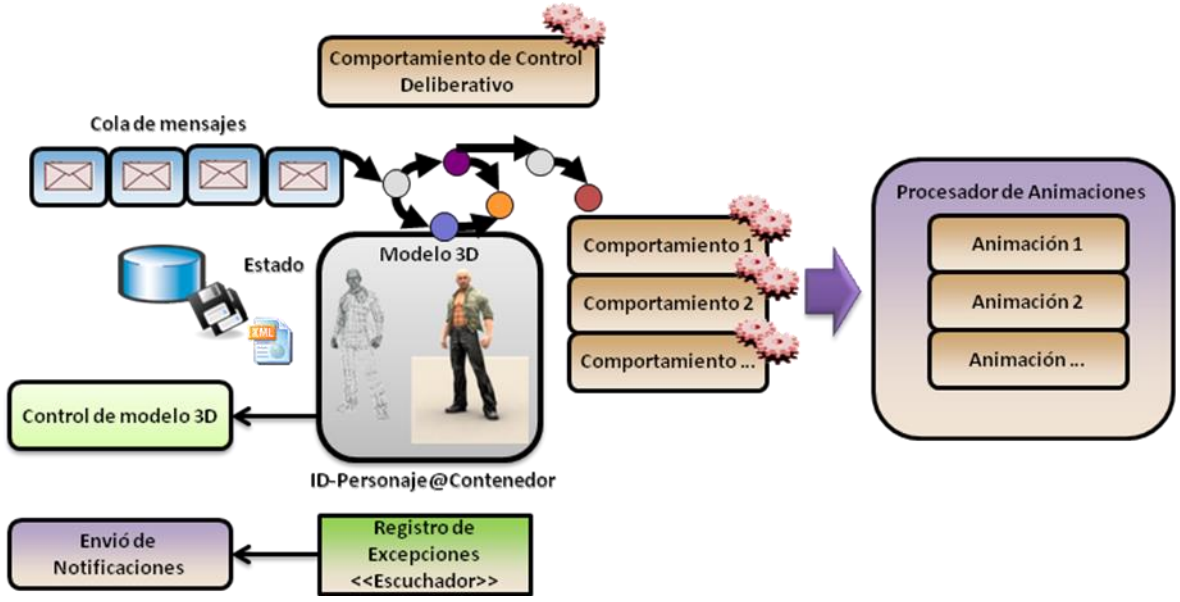


Figura 23 – Modelo de personaje

La Figura 23 – Modelo de personaje muestra los componentes desarrollados para el API de personajes, a continuación se lista los nuevos elementos del modelo:

- **Modelo 3D:** Representa un sistema que permite el dibujado de un objeto grafico en un plano en tres dimensiones, controlando la posición, desplazamiento, rotación, manejo objetos envolventes para manejo de colisiones entre objetos.

- **Control de modelo 3D:** Permite controlar la entrada de información que efectúa el movimiento del agente no solo para personajes jugadores sino también para personajes no jugadores.

- **Procesador de animaciones:** Librería creada para extender los modelos tridimensionales ofreciendo un procesador que transforma modelos **Simples 3D** en **Modelos Animados** para ofrecer al desarrollador mayor facilidad al reproducir una animación. Esta librería están estrechamente relacionado con los comportamientos puesto que son estos quienes usan las animaciones para representar el estado en el que se encuentra un personaje.

- **Animaciones**: Módulo mediante el cual se da la sensación de movimiento a los personajes cuando están ejecutando un comportamiento, es importante resaltar que dentro de un comportamiento se pueden ejecutar varias animaciones.

4.1.3. Comportamientos independientes y pro-actividad

Las tareas que realiza un agente se estructuran en comportamientos, que permiten que el agente sea implementado como un hilo (thread) de control. Estos comportamientos se desacoplan del API logrando una mayor flexibilidad en la implementación mediante el uso de delegados y eventos. La programación basada en comportamientos debe realizar los siguientes pasos:

- Determinar qué debe ser capaz de hacer el agente.
- Asociar cada funcionalidad con un comportamiento y un conjunto de animaciones.
- Programar el comportamiento de control.
- Enlazar la funcionalidad programada al delegado del comportamiento.
- Dejar que la API programe la ejecución del comportamiento, basado en los estados del agente.

Se puede pensar que los comportamientos como hilos de ejecución. No obstante, a diferencia de los hilos estos no se ejecutan en cada momento, es tarea del programador definir un comportamiento adecuado de control que le permitan al programador de tareas (scheduler) ejecutar una acción según las necesidades del personaje. El scheduler de la librería permite que cada agente se equipare únicamente a un único hilo de control, lo que posibilita el ahorro de procesador y memoria.

4.2. Arquitectura de la Librería

El desarrollo de la librería para el control del ciclo de vida de personajes se realizó mediante la construcción de 3 capas: renderizado, lógica del personaje y servicios. Este desarrollo representa un factor de gran utilidad en la construcción, dado que permite la división de responsabilidades, flexibilidad, escalabilidad y descomposición de las capas en otras menos densas. La Figura 24 – Arquitectura del API para personajes muestra la arquitectura desarrollada para la Librería.

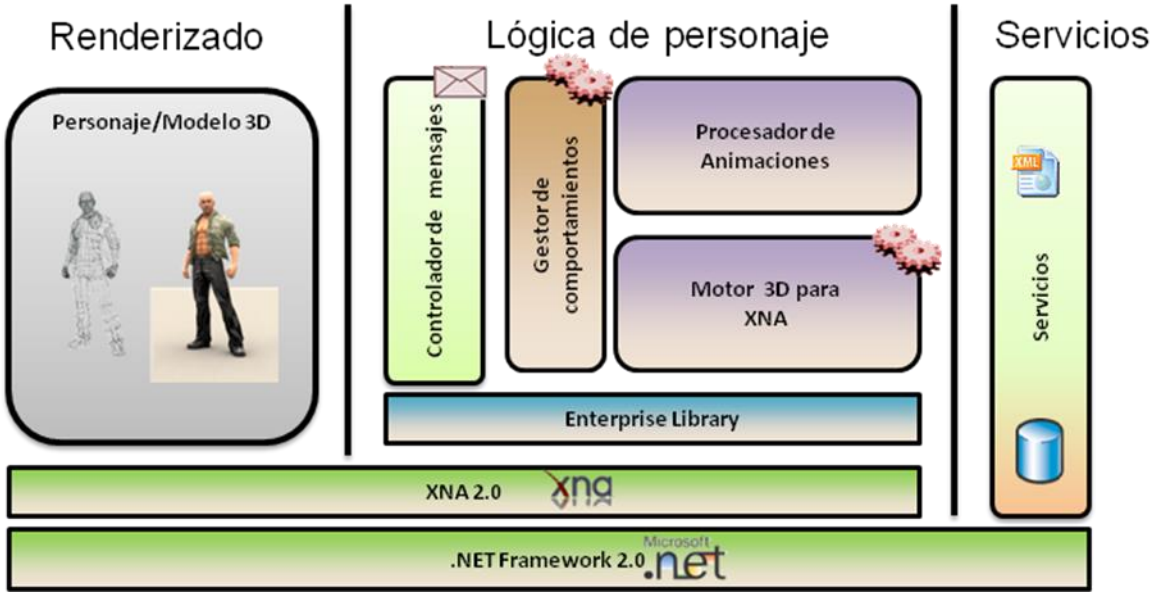


Figura 24 – Arquitectura del API para personajes

4.2.1. Capa de Renderizado

Esta capa permite realizar toda la parte gráfica relacionada con el personaje, realiza la carga de modelos 3D en formatos .x y .fbx, controlando la forma como se dibujan los objetos mediante las matrices de proyección, vista y mundo. Se encarga también de aplicar los efectos de texturas y de luz para el objeto así como de identificar que objetos del mundo se encuentran cerca, bien sea para realizar una colisión e impedir el paso o para lanzar una conversación.

Esta capa también es la encargada de realizar las transformaciones requeridas en el modelo como escalar, rotar, trasladar. Además permite la aplicación de efectos como el de niebla, que permite que no se dibujen los objetos hasta no estar a una distancia corta de ellos, minimizando la carga de dibujado para la tarjeta gráfica. Todo este módulo es soportado sobre el framework de XNA, lo que posibilita no solo el dibujado en equipos de escritorio sino también en consolas XBOX que son más atractivas para los jugadores.

4.2.2. Lógica de personaje

Esta capa concentra sus esfuerzos en ofrecer los servicios para que el personaje controle los sus comportamientos está conformada por los siguientes módulos:

Controlador de mensajes: Este módulo se basa en la teoría de actos de habla para agentes y permite realizar conversaciones primarias. Las *conversaciones* son iniciadas directamente por el agente, mientras que las conversaciones secundarias, se inicia como resultado de otra

conversación. Las performativas que maneja el sistema son tomadas del proyecto STI+IA y se listan en la **Tabla 29 - *Lista de Performativas que soporta la API*** [61].

PERFORMATIVAS	PROPÓSITO
Accept_Proposal	**Informa que se acepta la conversación previa para la realización de una acción.**
Agree	**Informa que se acepta la acción.**
Cancel	**Informa que se debe cancelar la acción.**
Call_for_Proposal	**Informa sobre el llamado a una oferta de acción.**
Confirm	**Confirma la realización de la acción.**
Disconfirm	**No confirma la realización de la acción.**
Failure	**Informa si se presenta un fallo.**
Inform	**Mensaje de información.**
Inform_If	**Mensaje de información previo a una pregunta.**
Inform_Ref	**Mensaje de información previo a una conversación.**
Not_Understood	**Informa que el mensaje no se ha comprendido.**
Propagate	**Permite identificar cuáles son los agentes a los cuales debe ser propagado el mensaje.**
Propose	**Informa el propósito de una acción.**
Proxy	**Informa el Agente objetivo del mensaje.**
Query_If	**Respuesta a una pregunta.**
Query_Ref	**Respuesta a una referencia.**
Refuse	**Denegación a la petición y explicación de la denegación.**
Reject	**Rechazo de la petición.**
Proposal	**Oferta de acción.**
Request	**Respuesta a una acción.**
Request_When	**Petición cuando se realiza una acción.**
Request_Whenever	**Petición siempre que se realice una acción.**
Subscribe	**Suscripción a un servicio.**

Tabla 29 - *Lista de Performativas que soporta la API*

Gestor de comportamientos: Permite a un personaje ejecutar una serie de acciones de manera proactiva y reactiva, de la misma manera que los seres vivos realizan diferentes acciones cuando se enfrentan a estímulos diversos. Este módulo permite al personaje gestionar de forma explícita diferentes comportamientos y tiene tres aspectos principales:

- La definición de comportamientos alternativos para el personaje
- Un comportamiento de control que otorga las condiciones particulares que deben producir un cambio de comportamiento
- Los mecanismos que producen el cambio.

Procesador de animaciones: Este módulo permite realizar animaciones basadas en esqueletos y huesos para un modelo 3D, pero también permite la reproducción de animaciones basadas en KeyFrames; este módulo se implementó puesto que XNA no posee por defecto un manejo de animaciones propio. El desarrollo de este componente se hizo a través de un procesador que transformara los modelos simples en modelos complejos con información adicional lo que hace posible la reproducción animaciones.

Motor 3D para XNA: Este módulo fue construido para apoyar el dibujado y manejo de escenarios en tres dimensiones en XNA y se explica con mayor detalle en el ítem 5.3.1.2.3 Motor gráfico.

4.2.3. Servicios

Esta capa ofrece asistencia a los módulos que se encuentran sobre la capa de servicios, los principales servicios son:

- Persistencia basada la hidratación y des-hidratación del estado de los personajes con en estándares XML y XSD.
- Configuración del registro de excepciones y módulo de registro basado en Enterprise Library.
- Listener para registro de excepciones en formato plano o envío por correo electrónico.

CAPITULO V
RESULTADOS OBTENIDOS

Este capítulo se presentan los principales aportes y productos obtenidos a partir de la ejecución del presente proyecto.

5.1. Propuesta metodológica para la construcción de videojuegos de aventura 3D

El primer producto obtenido a partir de la elaboración de este proyecto, es una propuesta metodológica para la construcción de videojuegos de aventura en 3D que se basa en la metodología para la construcción de Sistemas multi-agentes MAS-CommonKADS. Esta metodología fue adaptada de tal manera que permitiera la integración de los procedimientos de las metodologías para la construcción de videojuegos de aventura, en los cuales existe una fuerte tendencia a orientarse hacia la parte artística o administrativa de la construcción de videojuegos dejando de lado la implementación del juego. En esta propuesta se presentan varios modelos en los cuales se plantean una serie de pasos orientados a estructurar la construcción de un videojuego partiendo de una perspectiva artística para llegar a una codificación. Esta estrategia centra los esfuerzos del grupo de trabajo de acuerdo con sus habilidades y tareas dentro del grupo.

De forma que resulte más fácil referenciar la propuesta metodológica, será denominada GAME-MAS-CK. En la propuesta se plantean tres etapas, que son: Conceptualización, Análisis y Diseño. Dentro de estas etapas se proponen los modelos ilustrados en la Figura 25 - Modelos metodología propuesta.

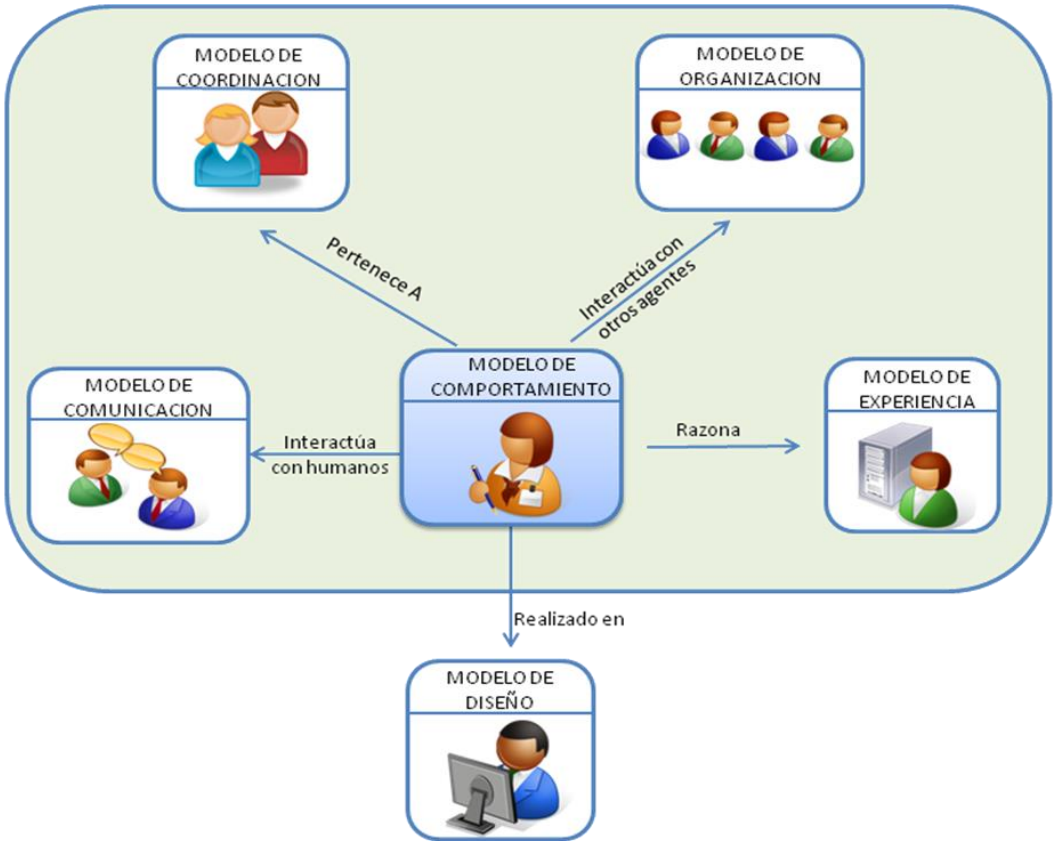


Figura 25 - Modelos metodología propuesta

La primera etapa, denominada Etapa de Conceptualización, tiene como objetivo generar una descripción preliminar del videojuego a partir de la construcción del S*toryline.* El Storyline puede ser concebido como historia o un cuento, que contiene en términos generales las bases conceptuales para la construcción del juego. Una vez finalizada la construcción del storyline, es posible dar inicio a la realización del *Análisis del argumento base,* donde se describen con mayor detalle las posibles escenas que se presentarán organizadas de forma secuencial. En cada escena se deben plasmar las acciones más relevantes realizadas por cada uno de los personajes. El producto obtenido una vez realizada esta tarea es muy parecido a un guión cinematográfico, una tira cómica o una obra de teatro, y permite a los miembros del equipo de desarrollo tener una visión mucho más clara del videojuego a construir. La siguiente tarea de esta etapa es la *Definición de reglas*, que tiene por objetivo limitar y estructurar la definición del juego a través de pautas abstractas que permitan una descripción funcional del mismo. Las reglas configuran los límites a las actividades que pueden desarrollar los jugadores en el juego. También es necesario realizar una tarea que permita la *Definición del mundo,* donde se describan los distintos objetos que forman parte de los escenarios propuestos. Igualmente, es preciso especificar la forma del terreno. Las tareas mencionadas hasta el momento, son realizadas utilizando lenguaje natural y apoyadas en ilustraciones, ya que esta etapa es realizada en mayor parte por los artistas del equipo de trabajo los cuales tienen una tendencia a considerar que los esquemas en esta etapa limitan la libre expresión de la creatividad. Es importante resaltar que al final de la etapa se propone una última tarea denominada *casos de uso de los personajes,* en la cual se identifican los

personajes más relevantes del las escenas del juego basados en la importancia de sus interacciones con el personaje principal. Posteriormente, se realiza una descripción más detallada de cada uno de estos personajes, se generan bocetos los mismos; una vez definidos los personajes, es posible iniciar con la identificación y descripción de los casos de usos de manera que queden especificadas las actividades realizadas por cada uno de estos personajes dentro del juego. La etapa de conceptualización de GAME-MAS-CK absorbe los modelos de agentes y el de tareas de MAS-CommonKADS, ya que para la construcción de videojuegos de aventura los personajes se consideran como agentes. La segunda, llamada etapa de análisis plantea los siguientes modelos los cuales se adaptan de la metodología MAS-CommonKADS de manera que tengan una orientación hacia la construcción de videojuegos de aventura. Los modelos propuestos en GAME-MAS-CK son:

- **Modelo de comunicación:** Este modelo permite definir las comunicaciones entre los distintos personajes/agentes del juego, determinando cuales pueden tener un canal de comunicación directo y si ese canal es unidireccional o bidireccional. En este modelo se ilustra la comunicación entre los personajes/agentes y el jugador.

- **Modelo de comportamiento:** En este modelo se realiza un mapeo de los casos de usos de los personajes planteados, de forma que estos puedan ser entendidos a partir de las propiedades planteadas para los agentes software, en especial: habilidad social, re-actividad y pro-actividad. Las propiedades de los agentes se pueden identificar con facilidad en los personajes de un videojuego de aventura, debido a que las relaciones con los elementos de su entorno se encuentran descritas de forma explicita en el *storyline* y más detalladas en el *análisis del argumento base*, ambas tareas del la anterior etapa.

- **Modelo de experiencia:** El modelo de experiencia, en el caso de los videojuegos de aventura, es el encargado de definir la complejidad inmersa en el videojuego. Esto se ve reflejado en la forma en que se comportan los personajes no jugadores de un videojuego. Para implementar esta complejidad en la actualidad se usan varias técnicas, de acuerdo con las necesidades del juego. Algunas de las técnicas más comunes son las máquinas de estados finitos, las redes neuronales, los árboles de decisión y algoritmos genéticos. Esta metodología no pretende ligarse a ninguna de estas técnicas, pero es necesario que durante el análisis se establezca de forma clara la complejidad del juego, para luego escoger la técnica que mejor se ajuste a las necesidades.

- **Modelo de coordinación:** En el modelo de coordinación se realiza una descripción más detallada de las interrelaciones de los personajes/agentes las cuales se encuentran agrupadas por conversaciones. Una conversación puede ser vista como un conjunto de interacciones iniciadas con el fin de alcanzar un objetivo. Cada interacción es realizada mediante el envío de mensajes entre dos personajes/agentes, y debe tener asociado un acto de habla. Estas conversaciones pueden iniciarse o responderse por los siguientes motivos: necesitar, proporcionar, suministrar o aceptar ayuda. A su vez, la ayuda o petición de cooperación puede ser de dos tipos: intercambio de información o solicitud de realización de una tarea.

- **Modelo de organización: En** este modelo se definen las relaciones estructurales entre los personajes/agentes, permitiendo una descripción de la organización del personaje/jugador y de la sociedad de personajes/agentes a partir de la identificación de las relaciones estáticas existentes. Estas relaciones son una extensión de la relación de herencia definida en la programación orientada a objetos, donde una clase agente es la generalización de un conjunto de agentes con la misma arquitectura. La arquitectura define sus servicios, habilidades, sensores, actuadores, creencias y objetivos compartidos por todos los agentes. La relación de herencia se puede evidenciar cuando se tiene un conjunto de agentes/personajes que comparten objetivos, creencias, planes y servicios propios de una misma clases, pero es necesario que un agente/personaje en especial se le puedan agregar objetivos, creencias, planes y servicios particulares, según las necesidades descritas en la historia.

Modelo de diseño: El modelo de diseño es el único modelo planteado para la tercera etapa o etapa de diseño. Se describen los componentes de los distintos modelos de la etapa de análisis, de manera que resulte más fácil su definición en un lenguaje de programación. En este modelos se definen dos clases de decisiones de diseño: el diseño de agentes, donde se documentan las distintas funciones de cada módulo, que deben ser implementadas según la arquitectura seleccionada para el agente; el diseño de la plataforma, permite la documentación de las decisiones de bajo nivel, acera del lenguaje de implementación seleccionado, el software y el hardware empleado, además se definen los usuarios finales del sistema.

5.2. API para la implementación de Personajes/Agentes

Con el objetivo de facilitar la implementación de juegos basados en agentes, en el desarrollo de este proyecto se planteó una API, soportada en el framework para el desarrollo de juegos XNA. La API propuesta relaciona conceptos de la programación de agentes con los conceptos de programación necesarios para la construcción de videojuegos.

La API cuenta con un modelo base para los agentes, el cual permite realizar: deliberación interna de un agente, para realizar la ejecución de un comportamiento en particular dentro del poll de comportamientos creados; el almacenamiento de estado y por último un módulo para control de registro de excepciones cuya notificación puede ser enviada por diversos métodos (e-mail, archivos planos, etc.). Esta API también cuenta con un modelo base del personaje, que hereda las propiedades implementadas para los agentes y adiciona características propias de un personaje de videojuego, como son: el modelo 3D, el control del modelo 3D, el procesador de animaciones y las animaciones.

Además, la API estructura las tareas realizadas por los agentes en comportamientos, los cuales permiten la implementación del agente como hilo de control (thread), estos comportamientos pueden ser desligados de la API, proporcionando mayor flexibilidad en la implementación. La arquitectura sobre la cual se construyó la API consta de tres capas denominadas la Capa de renderizado, encargada de toda la parte gráfica relacionada con el personaje; la capa de lógica del

personaje, que tiene como función ofrecer los servicios que permiten al personaje controlar sus comportamientos; y la capa de servicios, cuyo objetivo es ofrecer asistencia a los módulos de las capas superiores.

Es importante resaltar que para la construcción de esta API se tomó como base la API desarrollada en el proyecto STI+IA [61] y además de adaptar la funcionalidad existente a la construcción de videojuegos, se adicionaron características como: el comportamiento del control, el estado del agente, el registro de excepciones y el envío de notificaciones.

5.3. Videojuego

A continuación se presentan los aspectos que se consideraron en el momento de implementar el juego "Simón y la Tuberculosis – nivel 2", como videojuego caracterizado por la investigación, exploración, la solución de rompecabezas y la interacción con personajes.

5.3.1. Arquitectura del videojuego

La Figura 26 – Arquitectura del video juego muestra la arquitectura del videojuego que se realiza usando la metodología MAS-COMMON-KADS y la API para la construcción de personajes.

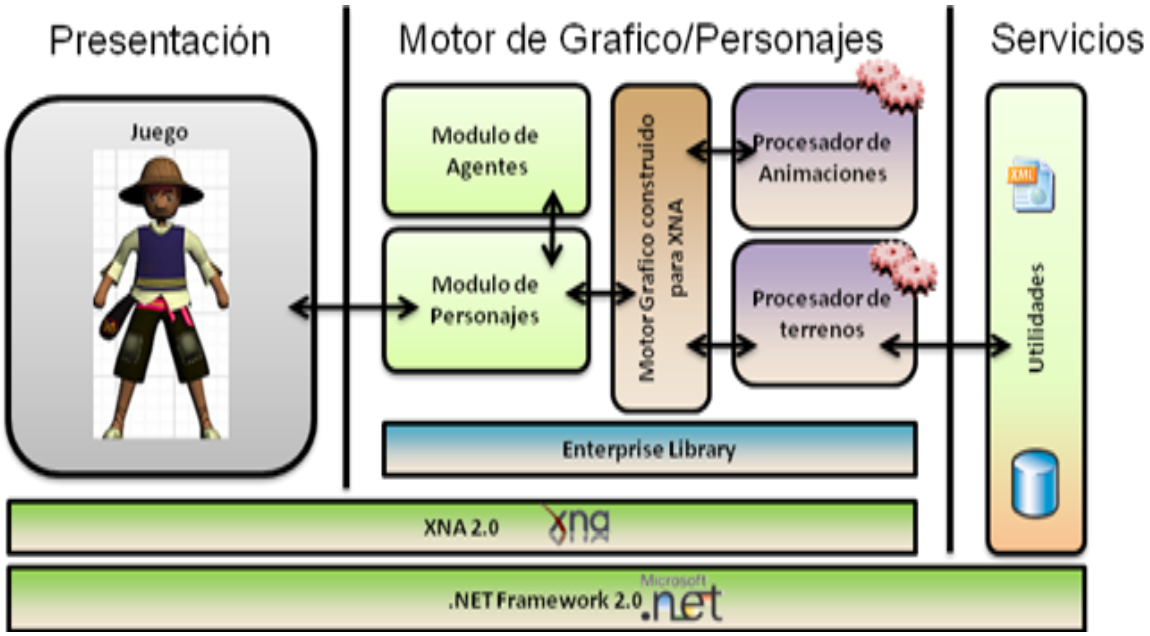


Figura 26 – Arquitectura del video juego

Si bien las arquitecturas de los videojuegos no son muy similares a las de un sistema de información, se realizaron esfuerzos para desarrollar una arquitectura multinivel donde se nota una marcada separación entre la presentación, la lógica del juego y unos servicios adicionales, dando como resultado una arquitectura con 3 niveles distribuidos así:

5.3.1.1. Capa de presentación Juego

Esta capa contiene un único módulo que representa la interfaz gráfica del juego, y realiza las veces de control del videojuego. Dentro de ella se cuenta con los recursos gráficos como *Textura*, *Modelos 3D*, gráficos de menú, etc. Esta capa fue desarrollada usando el patrón *vista-controlador* para realizar una separación visible entre la capa de presentación y la capa de motor gráfico y de personajes.

5.3.1.2. Capa de Motor Gráfico y personajes

Esta capa contiene varios módulos que hacen las veces de lógica del juego y que fueron desarrollados de manera genérica para permitir su reutilización en trabajos futuros, los elementos con que cuenta esta capa son:

5.3.1.2.1. Módulo de agentes

Este módulo constituye en sí mismo una librería para el control de ciclo de vida de agentes genéricos, y fue desarrollado como una biblioteca de clases que hace uso del los patrones y buenas prácticas de Microsoft a través del uso de Enterprise Library. El módulo contiene una adaptación del framework de JADE[56], y hace uso de los estándares FIPA ACL[55] para la comunicación entre agentes.

5.3.1.2.2. Módulo de personajes

Este módulo constituye el API (en forma de una biblioteca de clases) para la creación de personajes y es el resultado de combinar el módulo de agentes con el motor gráfico. Esta combinación produce un sistema que permite crear de personajes que usan modelos gráficos y se comportan como agentes con características como la reactividad, proactividad y la habilidad social.

5.3.1.2.3. Motor gráfico

Este motor permite controlar los aspectos lógicos del desarrollo de juegos en 3D, y controla los objetos tanto estáticos como en movimientoo y el manejo de cámaras. También se encarga de controlar el manejo de menús gráficos, control del mini mapa del juego y control del inventario de gemas y de objetos recogidos por el personaje jugador.

5.3.1.2.4. Procesador de animaciones

Este elemento fué desarrollado como un módulo especial denominado ContentPipeLine, que permite tomar un recurso y transfórmalo en otro. Para este proyecto se realizó la transformación de un modelo simple a un modelo complejo con animaciones, que permitieran dar la sensación de movimiento en cada uno de los modelos del juego. De esta forma se logró una mejora en la presentación del juego.

5.3.1.2.5. Procesador de Terrenos

Al igual que el procesador de animaciones, este modulo fue creado como un *ContentPipeLine*. Su función consiste en proporcionar los mecanismos para permitir la carga de un archivo de texturas (denominado mapa de alturas), y transformarlo en un modelo 3D para construir el terreno en el cual se desarrolla el juego.

5.3.1.3. Capa de Servicios

Esta capa contiene un módulo de utilidades que permite el trabajo con los archivos XML, que constituyen una base fundamental en el trabajo. Los archivos XML permiten realizar cambios de escenario en el juego sin tener que realizar cambios en la programación del mismo.

5.3.2. Sistema de coordinación de cámaras del video juego

El juego cuenta con un sistema de tres posiciones para el manejo de la cámara, lo que permite cambiar la perspectiva visual mientras el personaje se desplaza por el terreno. Esto se implementó mediante el cambio de configuración del método setLookAt, calculando el sistema de coordenadas con vectores de 3 dimensiones: profundidad (eje z), proyección (eje x) y altura (eje y). La Figura 27 – Sistema de coordinación de cámara ilustra el sistema de coordenadas de la cámara ubicada en el sistema coordenadas del mundo. Es importante resaltar que los vectores que componen el sistema de visión de la cámara deben ser vectores unitarios y perpendiculares. Se pueden usar los vectores unitarios para representar direcciones, porque el tamaño de los vectores no cambia la dirección de visualización.

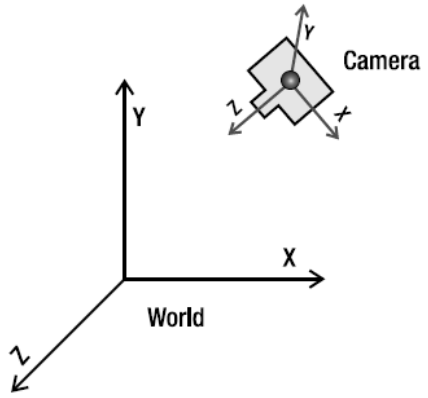


Figura 27 – Sistema de coordinación de cámara

El vector de profundidad es el que ubica la posición de la cámara con respecto a la posición del objeto que se quiere visualizar, y se puede calcular restando a la posición de la cámara la posición con la posición del objetivo. El vector de altura define qué tan alto debe ubicarse la cámara y es usado para orientar el ángulo de la cámara. Por ejemplo, se puede usar el vector (0, 1, 0) para orientar la cámara como respecto al eje *y* del mundo.

Otro punto importante para resaltar es que para fijar las diferentes posiciones de la cámara, se definieron tres vectores de altura que no necesariamente son perpendiculares al vector de profundidad, pero si perpendiculares al vector de proyección. Si se requiere que el vector de altura sea perpendicular al vector de proyección, después de calcular el vector de proyección se debe calcular un nuevo vector de altura con el producto cruzado entre los vectores restantes.
Esos tres vectores forman el sistema de coordenadas de la cámara y son usados cada vez que sea necesario para transformar la cámara basada en estos tres ejes.

5.3.2.1. Cámara en Tercera Persona

Como la mayoría de juegos de aventura, Simón y la Tuberculosis usa un sistema de cámaras en tercera persona, es decir, que el personaje principal (Simón) es seguido por la cámara mientras este se encuentra en movimiento. Además la distancia en la que la cámara sigue un objeto debe ser constante, este mecanismo da la impresión de que el objeto está vinculado a la cámara. Para hacer que la cámara siga a simón es necesario definir algunos parámetros, como la posición de persecución (la posición que la se ubica la cámara para seguir al personaje); dirección (la dirección utilizada para seguir al objeto); velocidad y la distancia máxima y mínima entre la cámara y el objeto.

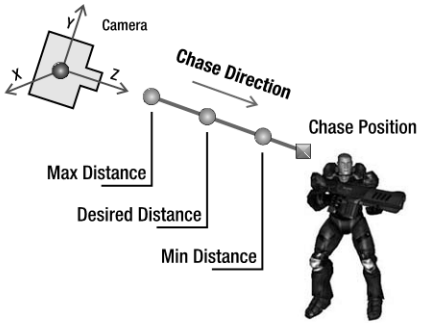


Figura 28 - Cámara en Tercera Persona

5.3.3. Armado del terreno de juego

El terreno es uno de los aspectos más importantes en un juego de aventura, puesto que es sobre éste donde se va a desarrollar la trama del juego. Existen básicamente dos técnicas para desarrollar un terreno de juego: la primera es el manejo de lozas (tiles), que requieren fraccionar el piso del juego en porciones como baldosas de un tamaño fijo y cada una con su respectiva textura. Este modelo implica un desgaste del diseñador, puesto que se debe conseguir que las porciones sean el menor número posible y que se permita la reutilización en varias partes del juego. Esta técnica es muy usada para escenarios donde el terreno es plano y en juegos en dos dimensiones. La segunda técnica consiste en la creación de un modelo o malla del terreno, generando las alturas de manera aleatoria o a través de un mapa de alturas y colocando una única textura en el terreno. Esta práctica es más adecuada, puesto que genera terrenos con alturas irregulares y con la ventaja de poder hacer diseños de texturas mucho más vistosos que no establecen límites al diseñador.

5.3.3.1. Mapa de alturas

El mapa de alturas es una imagen en 2-D en la cual se almacenan las alturas del terreno. Son usualmente guardados en imágenes de 8 bits en escalas de grises. Cada punto de la imagen almacena la posición en la que se encuentra la altura del terreno.

Para construir el terreno a partir de un mapa de alturas, se construye una grilla de vértices con las mismas dimensiones del mapa. Se utiliza el valor del color de cada punto (Píxel) como la altura de un vértice sobre la grilla de vértices. Además, cada vértice de la grilla contiene otros atributos necesarios para dibujar, como la normal, una grilla de vértices de 6 x 6 creada sobre el plano XZ del mundo, donde la altura del vértice es relativa al eje y del mundo.

En la grilla de vértices se debe definir una distancia entre cada par de vértices (vertical y horizontalmente). Para este proyecto este valor equivale a 20 unidades, y se conoce como la "escala del bloque". Entre menor sea la distancia entre los vértices más suave será la transición entre la altura de los vértices y la grilla de vértices pero reduce el tamaño de la grilla, mientras que al aumentar la distancia entre los vértices el tamaño de la grilla es incrementado produciendo fuertes transiciones entre las alturas de los vértices. De esta manera, si la distancia entre cada par de vértices (vertical y horizontalmente) es 1 metro, el tamaño total del terreno generado seria de 255 x 255 metros.

Como el mapa de alturas del terreno es almacenado generalmente en imágenes de 8 bits, sus valores de altura pueden tomar valores entre 0 y 255, donde 0 (color negro) representa la altura más baja para un vértice y 255 (color blanco) representa la altura máxima. Se puede aumentar este intervalo usando una variable, que es multiplicada por el valor por defecto de la altura, incrementando su rango.

Los mapas de alturas tienen una desventaja y es que al construir un modelo a partir de la realización de cálculos matemáticos para determinar la altura, el rendimiento del juego se disminuye considerablemente. Se debe garantizar que el terreno debe ser generado con el menor

desgaste posible por parte del sistema. Por esta razón dentro del juego se creó un preprocesador, que toma la imagen del mapa de alturas y la transformara en un modelo 3d completo. Para mejorar el rendimiento, este proceso sólo lo realizara la primera vez que se compila el juego y no cada vez que se ejecute, lo que garantiza que el modelo será cargado muy rápidamente. La Figura 29 – Funcionamiento del procesador de terrenos del juego muestra la forma como se realiza la transformación.

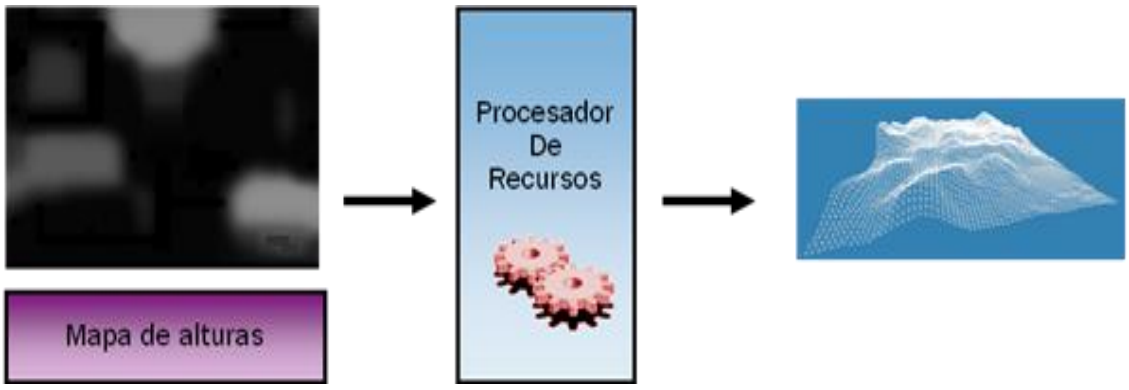


Figura 29 – Funcionamiento del procesador de terrenos del juego

5.3.4. Armado del escenario de juego

El armado del mundo dentro de un juego de aventura requiere que sea dinámico[60] y que permita dibujar los objetos en diferentes posiciones (o escenario), cada una de las cuales podría considerarse un nivel o hacer parte del mismo. Los juegos actuales permiten a los usuarios construir sus propios escenarios, o modificar los existentes[18], generando diversos mundos y retos.

Al iniciar con el desarrollo del juego el diseñador no tenia tenía muy claro las posiciones de los objetos y los nombres de los modelos que se deseaban posicionar en la escena del nivel, es por ello que se debió buscar un mecanismo que permitiera hacer cambios y de manera fácil configurar los elementos del juego, posibilitando que una vez compilado el juego estos elementos se reconfiguraran de acuerdo con deseo no solo de desarrolladores sino también diseñadores y jugadores.

Para hacer cambios en la configuración de videojuegos se usó la técnica de archivos de configuración en XML. Estos permiten la carga de información creando objetos en memoria a partir de documentos XML que es un lenguaje estándar, en texto plano y auto descriptivo. A continuación se presenta un ejemplo de un archivo, donde se muestran dos modelos posicionados dentro de la escena del juego:

```
<?xml version="1.0" encoding="utf-16"?>
<ArrayOfStaticObject xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <StaticObject>
    <Enabled>true</Enabled>
    <UpdateOrder>0</UpdateOrder>
    <Position>
      <X>-1400</X>
      <Y>-600</Y>
      <Z>-700</Z>
    </Position>
    <Translation>
      <X>0</X>
      <Y>0</Y>
      <Z>0</Z>
    </Translation>
    <Rotation>
      <X>0</X>
      <Y>0</Y>
      <Z>0</Z>
    </Rotation>
    <ObjectName>House 2/house2</ObjectName>
  </StaticObject>
  <StaticObject>
    <Enabled>true</Enabled>
    <UpdateOrder>0</UpdateOrder>
    <Position>
      <X>-1400</X>
      <Y>-600</Y>
      <Z>-2200</Z>
    </Position>
    <ObjectName>House_2/house2</ObjectName>
  </StaticObject>
</ArrayOfStaticObject>
```

5.3.5. Problemas presentados en el desarrollo del video juego

5.3.5.1. Manejo de colisiones en 3D.

Uno de los principales problemas presentados en el desarrollo del juego fue la detección de colisiones en espacios tridimensionales. Esta detección se diferencia de la detección de colisiones que se realiza en dos dimensiones, que en su forma más básica se fundamenta en la intersección de dos cuadrados para determinar si dos objetos se encuentran uno con el otro, como lo muestra la Figura 30 – Colisiones en 2D.

Figura 30 – Colisiones en 2D

El manejo de colisiones en tres dimensiones se puede realizar usando volúmenes contenedores de los objetos para intentar encontrar un punto donde los dos elementos se encuentran ocupando el mismos espacio. Los volúmenes más usados y que se tienen de manera predeterminada en las librerías graficas son: el cilindro, el cubo, y la esfera, en caso de la librería XNA se cuenta con los objetos *BoundingBox, BoundingSphere* y *Plane*. El propósito de estos objetos es envolver los modelos en 3D y hacer una colisión usando elementos más simples sin tener que hacer uso de las objetos en sí, puesto que realizar miles o millones de cálculos matemáticos de un conjunto de objetos envolventes para un modelo de un personaje de un juego baja el rendimiento del mismo.
En el caso de modelos compuestos por varios *mesh* o mallas, es posible envolver cada una de ellas usando un volumen propio, lo que facilita la detección de colisiones. Para el caso de los modelos genéricos creados para el juego, estos no tienen un conjunto de mallas, y solo cuentan con un único objeto que debe ser envuelto totalmente usando uno de los volúmenes ya mencionados.
En proceso de cubrimiento del objeto se hace uso del eje central del objeto y se revisa el tamaño para formar el volumen. Para el caso del juego desarrollado se tuvieron que resolver dos problemas:

- Los modelos del juego no tenían bien posicionado el centro de masa, lo que impedía usar las funciones de la librería para obtener de manera automática un volumen que envuelve el objeto.

  Este problema se solucionó cargando los modelos en el mundo y mostrando el modelo con una caja envolvente sobre él, y luego se almacenaron los objetos envolventes en un archivo XML con las coordenadas de la caja (BoundingBox) que contenía el objeto usando como base la posición del elemento. Estos datos fueron añadidos al archivo de posicionamiento como un nuevo elemento llamado **<BOX>**, y se modificaron las posiciones para que envolvieran correctamente al objeto como se muestra en el siguiente ejemplo:

```
<?xml version="1.0" encoding="utf-16"?>
<ArrayOfStaticObject xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <StaticObject>
    <Enabled>true</Enabled>
    <UpdateOrder>0</UpdateOrder>

    <Rotation>
      <X>0</X>
      <Y>0</Y>
      <Z>0</Z>
    </Rotation>
    <Box>
    <Min>
      <X>-1200</X>
      <Y>-400</Y>
      <Z>-300</Z>
    </Min>
    <Max>
      <X>-1800</X>
      <Y>-1000</Y>
      <Z>-900</Z>
    </Max>
    </Box>
    <ObjectName>House 2/house2</ObjectName>
  </StaticObject>
</ArrayOfStaticObject>
```

➢ El segundo problema fue realizar la colisión entre dos volúmenes. Para este problema inicialmente se consideró inicialmente la esfera (BoundingSphere) como la opción más adecuada, debido a la facilidad que existe para generarla usando una instancia del modelo en XNA. Sin embargo, la esfera no se adapta a todos los objetos estáticos del juego, como por ejemplo la cerca. El sólido ideal usado como mecanismo de colisión es el cubo, debido a que la mayoría de los objetos del juego encajan muy bien en él. Una vez seleccionado este sólido como contenedor de los objetos del juego, se procedió a realizar pruebas de colisión, detectando que existían problemas en la versión 2.0 de XNA para colisionar dos objetos tipo cubo (BundingBox). La documentación indica que los volúmenes manejados por XNA cuentan con un método (Intersect) que determinaba si dos objetos podrían ser colisionados; Al hacer las pruebas usando este mecanismo se detectó que método propio de framework de XNA no funciona cuando se hace la colisión de dos volúmenes tipo cubo. La solución a este problema consistió en realizar la intersección de dos tipos deferentes de volúmenes: la esfera y el cubo, ya que el juego maneja dos controladores para los tipos estáticos y los objetos en movimiento. Se usó la esfera como contenedor de los objetos en movimiento y el cubo como el contenedor de los objetos estáticos.

5.4. Cumplimiento de objetivos

A continuación se presentan las tablas resumen que consignan cada uno de los objetivos del proyecto, los productos esperados, los productos obtenidos, los medios de verificación y algunas estrategias u observaciones.

Objetivo:	**Objetivo General**
Descripción del Objetivo:	Proponer una metodología basada en las metodologías de diseño de agentes software, para la construcción de personajes en videojuegos de aventura 3D.
Productos Esperados:	Descripción de la Metodología para la creación de personajes GAME-MAS-CK Aplicación de la metodología
Resultados Obtenidos:	Documento con el análisis de las metodologías existentes para Agentes y descripción del desarrollo de la metodología GAME-MAS-CK que enfoca los modelos de MAS-CommonKADS a la creación de videojuegos, donde fue necesaria la integración de productos como: StoryLine, Análisis del argumento base, definición de reglas, definición del mundo. También se plantearon una serie de plantillas donde se estructura la información necesaria de cada uno de los modelos. Además se construyo una API basada en la metodología que permite la reutilización de código relacionado con agentes software y la construcción de videojuegos de aventura 3D. Aplicación de la metodología en la creación de personajes en el videojuego sobre el tratamiento de la tuberculosis (Simón y la tuberculosis – Nivel 2).
Medios de Verificación:	En el Capítulo 2 de este documento se encuentra la metodología propuesta para la realización de los personajes de un videojuego. En el Capítulo 3 de este documento se encuentran la aplicación de la metodología para el desarrollo de los personajes del videojuego sobre el

Objetivo:	Objetivo General
	tratamiento de la tuberculosis.
Estrategias y/o Observaciones:	Metodología para el desarrollo de personajes de un videojuego: se estudiaron las diferentes disciplinas que se reúnen en la creación de un videojuego para de esta manera determinar los diferentes pasos que se deben seguir en el desarrollo del mismo y los entregables que se debe tener al finalizar cada una de las etapas de la construcción del videojuego, logrando un mejor entendimiento de cada uno de los integrantes del equipo del videojuego. Después se realizó una revisión de las metodologías de agentes existentes, evaluándolas y comparándolas entre sí, tomando como base para el desarrollo la metodología MAS-CommonKADS. Esta metodología se seleccionó debido a que que es la más se adapta a las necesidades de desarrollo de un videojuego. No obstante, las otras metodologías también fueron tenidas en cuenta en menor medida para complementarla metodología elegida como base. Aplicación de la Metodología: Esta aplicación se llevó a cabo como un caso de estudio dentro del videojuego Simón y la tuberculosis, realizando un estudio de las características que debían tener cada uno de los personajes del juego y realizando cada una de las etapas por las que debe pasar para su construcción.

Objetivo:	1. Objetivo Especifico
Descripción del Objetivo:	Adaptar una metodología de diseño de agentes que permita incorporar las características que poseen los agentes en la construcción de videojuegos de aventura 3D.
Productos Esperados:	Descripción de la Metodología para la creación de personajes GAME-MAS-CK Aplicación de la metodología Video juego de aventura 3D API para la construcción de personajes/agentes
Resultados Obtenidos:	Descripción de la Metodología para la creación de personajes GAME-MAS-CK. Donde se modificaron los procedimientos de MAS-CommonKADS integrando productos y elementos propios del videojuego a la metodología como: StoryLine, Análisis del argumento base, definición de reglas, definición del mundo. Además modelos como el de agentes y el de tareas propios de MAS-CommonKADS, se integraron a los procesos de la fase de conceptualización y creó un nuevo modelo denominado modelo de comportamiento. Se plantearon plantillas para los procesos definidos en la metodología los cuales tienen con fin estructurar la información necesaria en cada proceso. Aplicación de la metodología en la creación de personajes del videojuego sobre el tratamiento de la tuberculosis. API que permita incorporar las características de los agentes a los personajes de un juego de aventura 3d

Objetivo:	1. Objetivo Especifico
Medios de Verificación:	En el Capítulo 2 de este documento se encuentra la metodología propuesta para la realización de los personajes de un videojuego. En el Capítulo 3 de este documento se encuentran la aplicación Metodología para el desarrollo de los personajes del videojuego sobre el tratamiento de la tuberculosis. En el capítulo 4 se realiza una descripción el API para la construcción de personajes dotándolos de las características propias de los agentes como reactividad, proactividad y habilidad social.
Estrategias y/o Observaciones:	Metodología para el desarrollo de personajes de un videojuego: se estudiaron las diferentes disciplinas que se reúnen en la creación de un videojuego para de esta manera determinar los diferentes pasos que se deben seguir en el desarrollo del mismo y los entregables que se debe tener al finalizar cada una de las etapas de la construcción del video juego, logrando un mejor entendimiento de cada uno de los integrantes del equipo del videojuego. Después se realizó una revisión de las metodologías de agentes existentes, evaluándolas y comparándolas entre sí. Se tomó como base para el desarrollo la metodología Mas-commonKADS, puesto que es la más se adapta a las necesidades de desarrollo de un videojuego. Aplicación de la Metodología: Esta aplicación se realizó como un caso de estudio dentro del videojuego Simón y la tuberculosis, realizando un estudio de las características que debían tener cada uno de los personajes del juego y realizando cada una de las etapas por las que debe pasar para su construcción. API para la construcción de personajes/agentes: Se realizó una implementación de una librería para construcción de agentes usando los estándares FIPA ACL, combinando este trabajo con el motor grafico produciendo una librería que permite la creación de personajes que usan modelos gráficos y se comportan como agentes con características como la reactividad, proactividad y la habilidad social.

Objetivo:	2. Objetivo Especifico
Descripción del Objetivo:	Construir un API[4] basada en la metodología como un módulo software sobre una librería gráfica o un motor de juegos.
Productos Esperados:	Librería reutilizable para la construcción de personajes en videojuegos de aventura 3D.
Resultados Obtenidos:	Documento con la descripción de la implementación del API. Librería que permite la reutilización de código relacionado con los agentes en la construcción de videojuegos. Esta fue incluida en el motor de videojuego utilizado para crear el juego “Simón & la tuberculosis”.
Medios de Verificación:	En el capítulo 4 se realiza una descripción el API para la construcción de personajes dotándolos de las características propias de los agentes como reactividad, proactividad y habilidad social.

[4] Application Programming Interface (Interfaz de Aplicación Programable)

Objetivo:	2. Objetivo Especifico
Estrategias y/o Observaciones:	API para la construcción de personajes/agentes: Se realizó una implementación de una librería para construcción de agentes usando los estándares FIPA ACL, combinando este trabajo con el motor gráfico seleccionado. El resultado es una librería que permite la creación de personajes que usan modelos gráficos y se comportan como agentes con características como la reactividad, proactividad y la habilidad social.

Objetivo:	3. Objetivo Especifico
Descripción del Objetivo:	Validar la metodología mediante la construcción de un nivel de juego para computador que cumpla con los siguientes requisitos: ✓ Utilice la implementación de la metodología propuesta (agentes-personajes). ✓ La historia debe encontrarse basada en el tratamiento de la tuberculosis pulmonar. ✓ Emplee un escenario acorde a la historia del juego con ambientación, como por ejemplo: edificaciones, personajes genéricos 3D, ayuda contextual y manejo de terrenos, entre otros. ✓ Sea elaborado en modo "single player", es decir, que lo pueda jugar un único jugador a la vez.
Productos Esperados:	Videojuego sobre el tratamiento de la tuberculosis
Resultados Obtenidos:	Documento que describe como se uso la metodología propuesta para la construcción del nivel para el videojuego "Simón & la tuberculosis". Dentro de la solución del videojuego se entregan las siguientes librerías: Motor 3D para XNA Procesador de animaciones Procesador de terrenos basado en mapas de alturas Modelos 3D genéricos y especializados para el videojuego Nivel del Videojuego "Simón & la tuberculosis".
Medios de Verificación:	En el Capítulo 5 de este documento se encuentra la descripción detallada de la implementación del nivel del videojuego como los resultados obtenidos de este proyecto.
Estrategias y/o Observaciones:	Implementación del nivel del videojuego "Simon y la tuberculosis": Se realizó una investigación de los videojuegos de aventura determinando los aspectos más importantes a tener en cuenta para desarrollar un videojuego de este tipo enfocándose principalmente en las características y comportamientos de los personajes que interactúan dentro de este. Se realizó un estudio detallado de aspectos a tener en cuenta sobre el tratamiento de la tuberculosis pulmonar para así poder realizar un guión acorde con la temática. Se estudiaron todos los aspectos relacionados con la animación en 3D de modelos dentro del framework para videojuegos XNA 2.0

CAPITULO VI
CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

La metodología Mas-CommonKADS presenta la estructura y las bases teóricas necesarias para construir una propuesta metodológica que permita el modelado y diseño de videojuegos de aventura en 3D basado en Agentes Software. Teniendo en cuenta que su etapa de conceptualización se define con base en el lenguaje natural, Mas-CommonKADS ofrece la posibilidad de integración de varias de las tareas necesarias para crear de videojuegos que usualmente son desarrollados por personal con un enfoque artístico.

A pesar de que la industria de los videojuegos ha sido catalogada como una de las mayores recaudadoras del mundo del entretenimiento en los últimos años, resulta notable la reducida cantidad de información formal relacionada con el desarrollo de videojuegos, desde una perspectiva que permita abordar las diferentes tareas del equipo de trabajo. Este proyecto plantea una propuesta metodológica donde se incluyen tanto productos y conceptos del equipo artístico como del equipo de desarrollo del juego, facilitando la comunicación entre los miembros del equipo.

La propuesta metodológica planteada en este proyecto, construida con base en la metodología MAS-CommonKADS, presenta los argumentos suficientes para ser tomada en cuenta al momento de realizar el desarrollo de un videojuego de aventura en 3D. Este trabajo constituye una contribución al área de desarrollo de videojuegos, ya que ofrece los componentes teóricos y prácticos apropiados para la construcción de videojuegos de aventura 3D, al incorporar las características de los agentes software a los personajes de los videojuegos.

La API construida a partir de la metodología propuesta para la implementación Agentes/Personajes se encuentra basada en los protocolos de comunicación de FIPA para agentes, lo cual permite dotar a los personajes de un juego de características como: reactividad, proactividad y habilidad social. Proporcionando la posibilidad de reutilizar el código relacionado con las características de agentes y renderizado de modelos 3D al programar videojuegos con personajes sobre el framework para la construcción de videojuegos XNA.

La API JATT para la construcción de videojuegos cuenta con un modulo de utilidades que permite hacer cambios en la visualización de los modelos 3D sobre el terreno de juego sin necesidad de recompilar el juego reduciendo los tiempos de codificación del juego.

En este sentido, es importante resaltar que el proceso de desarrollo de un videojuego es una tarea multidisciplinaria, en la cual, no solo es necesario contar con el trabajo de un equipo de desarrolladores, sino que también es necesario el apoyo de personal con conocimiento en áreas como: Comunicación Social, Música, Diseño Gráfico, entre otras, dependiendo la magnitud del desarrollo. Los procesos planteados en la propuesta metodológica permiten la integración de los productos desarrollados por los miembros del equipo de desarrollo, facilitando la comprensión en el momento de implementar el videojuego.

La metodología GAME-MAS-CK propone una construcción de videojuegos orientada a agentes, donde se integraron productos propios de la construcción de videojuegos como: StoryLine, Analisis del argumento base, definición de reglas, definición del mundo. Además cuenta con una serie de plantillas que sirven como medio de apoyo para estructurar y organizar la información necesaria de cada uno de los modelos con los que cuenta la GAME-MAS-CK.

6.2. Recomendaciones

Si se planea trabajar en el desarrollo de videojuegos en especial de aventura en 3D, la propuesta metodológica planteada en este proyecto, debería pensarse en la creación de una herramienta gráfica que sirva de apoyo en el modelado de los distintos artefactos de análisis y diseño planteados.

Igualmente se debe investigar acerca de los protocolos de comunicación de agentes que soportan el servicio de transporte de mensajes por la red, con el fin de conseguir los personajes creados en la plataforma se comuniquen con otros personajes logrando la creación de juegos de uso masivo a través de internet.

Si se desea abordar trabajos relacionados con la construcción de un videojuego es recomendable la creación de un grupo de trabajo interdisciplinario, que permita realizar una correcta etapa de conceptualización del juego ya que de este trabajo depende el éxito o fracaso del mismo.

6.3. Trabajos Futuros

El desarrollo de este trabajo se puede considerar como un estudio introductorio a la investigación relacionada con los videojuegos, en tanto este trabajo es de los pioneros en la construcción de proyectos encaminados a definir un compendio de tareas que permitan la construcción de un videojuego, permitiendo que la investigación en futuros trabajos se enfoque en aquello "que se va a construir" antes que en "cómo hacerlo". En función de lo anterior se propone temas para investigar como:

- La construcción de videojuegos de aventura que manejen en su historia temas educativos logrando que los mismos cuenten con una función formativa sin dejar de ser un medio de entretenimiento.

- La integración de técnicas de inteligencia artificial a juegos de aventura en 3D, con el fin de aumentar la complejidad en el comportamiento de los personajes y del videojuego.

- La creación de una herramienta que apoye gráficamente la construcción de los modelos planteados en la metodología propuesta.

- Integrar características de multi-player en la construcción de videojuegos basados en historias, utilizando el paradigma de agentes software en su proceso de desarrollo.

BIBLIOGRAFIA

[1] **Ausubel, P. y Sullivan V.** El desarrollo infantil. 3. Aspectos lingüísticos, cognitivos y físicos. Barcelona. Paidós.1983**.**

[2] **Bijou, Sydney W.** (1982): Psicología del desarrollo infantil. México, D. F. Trillas.

[3] **Calvo, A.** Ocio en los noventa: los videojuegos. Tesis Doctoral Palma de Mallorca: Universidad de les Illes Balears. 1997.

[4] **Licona, Ana Liliam.** Los videojuegos como actividad lúdica del siglo XXI: el caso de Latinoamérica.

[5] **Meix, joan Isern.** La mala influencia de los juegos violentos. http://www.meristation.com/v3/des_noticia.php?id=8531&pic EEUU. 2002. visitada el 14 de marzo de 2007.

[6] Universidad europea de Madrid. Observatorio del videojuego http://www.uem.es/web/cin/cin2/observatorio/ visitada el 14 de marzo de 2007.

[7] **Mancini, Pablo.** Videojuegos, sociabilidad y aprendizaje. http://videojuegos.educ.ar/vj/usos-pedagogicos/videojuegos_sociabilidad_y_apr.php visitada el 14 de marzo de 2007.

[8] **Canovas, Guillermo.** Estudio realizado por protégeles. Videojuegos, menores y responsabilidad de los padres. Diciembre del 2005.

[9] **Rodríguez, Elena.** Jóvenes y Videojuegos: Espacio, significación y conflictos (FAD), Los tipos agrupados de videojuegos. España. 2002. http://www.injuve.mtas.es/ .

[10] Intentando crear videojuegos. Metodología SCUMM2 (personajes). España 2005. http://admin.blog.mundo-omepet.com/index.php?op=Default&postCategoryId=2&blogId=1 visitada el 16 de marzo de 2007.

[11] **González González Alfredo; Baqueiro Espinosa Omar, Meza Cota Emmanuel** Metodologías de la I.A. (Agentes autónomos y Redes neuronales Supervisadas) aplicadas a NPC's (Non player characters), Universidad Autónoma de Baja California Sur. Departamento de Sistemas Computacionales. 2004. http://www.redcientifica.com/doc/doc200401210112.html visitada el 5 de marzo del 2007.

[12] **Chandler, Rafael.** It Builds Character: Character Development Techniques in Games. http://www.gamasutra.com/features/20050810/chandler_01.shtml 2005 visitada el 14 de marzo de 2007.

[13] **Ferber, J. y Gutknecht, O.** *A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems*. Actas de conferencia. Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98), IEEE CS Press. 1998.

[14] **G. Martin, Ignacio.** *La Industria del Videojuego: Un Negocio en Alza.* Madrid: CEPREDE, 2006, Vol. XI.

[15] **Crawford, Chris.** *The Art of Computer Games.* Washington: Washington State University Vancouver, 2000.

[16] **Tanguay, David.** *A guide to create the ideal adventure game.* California: Adventure Classic Gaming, 2006.

[17] **Egri, Lajos.** *Art of Creative Writing.* Ohio: UArt, 1998.
[18] **Zerbst, Stefan y Düvel, Oliver.** *3D Game Engine Programming.* Boston. Series Editor, 2004.
[19] **Rumbaugh, J, y otros.** *Object-Oriented Modeling and Design*. Prentice-Hall, 1991.
[20] **Pressman, Roger.** *Ingeniería del Software.* Mexico : Mac Graw Hill, 2005.
[21] **Samara Ruiz, Maria Eguimendia.** *Metodología para sistemas multiagente, MaSE.* Vol. 1.
[22] **Despain, Wendy**. Professional Techniques for Video Game. A K Peters Ltd. 1 Edición, 2008.
[23] **Doug, Church**. The Aim Of Game Design Methods. [En línea] 03 de Marzo de 2008. http://www.gamasutra.com/features/20030303/kreimeier_01.shtml
[24] **Doug, Church**. Formal Abstract Design Tools. Originally *Game Developer* magazine, Vol 3, Issue 28, July 1999. http://www.gamasutra.com/features/19990716/design_tools_01.htm
[25] **GAMASUTRA –** The Art & Business of Making Games. [Visitada el: 25 de junio de 2007] www.gamasutra.com
[26] **GAMEDEV.NET – all your game development needs.** [Visitada el: 26 de junio de 2007] www.gamedev.net
[27] **Diseño de videojuegos.** [En línea] 03 de Marzo de 2008. [Citado el: 25 de Octubre de 2007.] http://ultragamer.wordpress.com/
[28] **Feil, John y Scattergood, Marc.** *Beginning Game Level Design.* Boston. Thomson Course Technology PTR, 2005.
[29] **Jennings, Nicholas. Wooldridge, Michael.** *Intelligent Agents: Theory and Practice* Manchester : Knowledge Engineering Review, 1994.
[30] **Carrasco Polaino, Rafael.** *Propuesta De Tipología Básica De Los Videojuegos De PC Y Consola* 7, Villanueva. ICONO 14, 2006, Vol. I.
[31] **Ouarda Bourass, Miguel Angel Almasa.** *Metodologia GAIA.* 0506. Revista TAIA, Vol. 1.
[32] **Caire, G., Leal, F., Chainho, P., Evans, R., Garijo, F., Gomez-Sanz, J. J., Pavon, J., Kerney, P., Stark, J.,and Massonet, P.** Eurescom P907: MESSAGE - Methodology for Engineering Systems of Software Agents. [En línea] [Citado el: 14 de Junio de 2007.] http://www.eurescom.de/public/projects/P900-series/p907/default.asp. 2002.
[33] **GRASIA.** Grupo de Agentes de Software: Ingeniería y aplicaciones. *Página Principal INGENIAS.* [En línea] [Citado el: 14 de Junio de 2007.] http://grasia.fdi.ucm.es/ingenias/Spain/ejemplos/ejemplo_actividades/analisis-inicio.php.
[34] **Collis, Jaron y Divine, Ndunu.** *ZEUS Technical Manual.* Estados Unidos: British Telecommunication plc., 1999.
[35] **Nwana, Hyacinth, Ndumu, Divine y Lee, Lyndon.** ZEUS: An Advanced Tool-Kit for Engineering Distributed Multi-Agent Systems. *CiteSeer.IST.* [En línea] [Citado el: 22 de Septiembre de 2007.] http://citeseer.ist.psu.edu/nwana98zeus.html.
[36] **Iglesias Fernández, Carlos Ángel.** *Definición de una Metodología Para el Desarrollo de Sistemas Multiagente.* Madrid-España: Universidad Politécnica de Madrid, 1998.
[37] **Aguilar, J. Hidrobo, F. y Cerrada M.** A Methodology to Specify Multiagent Systems. Lecture Notes in Artificial Intelligence, 4496:92–101, 2007.
[38] **Franklin, Stan y Graesser, Art.** *Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages.* United States : Springer-Verlag, 1996.
[39] **DeLoach, Scott.** *Analysis and Design using MaSE and AgentTool.* Ohio MAICS, 2001. Vol. XII.
[40] Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.18 (2003), pp. 51-63. ISSN: 1137-3601. © AEPIA.

[41] **Salen, Katie. Zimmerman, Eric.** Rules of Play: Game Design Fundamentals. The MIT Press, 2004

[42] **Bacca, Alvaro.** "Simon Y La Tuberculosis: Experiencia Aplicativa Del Videojuego Como Herramienta Emergente Para Educar". Tesis de Pregrado – Diseño Grafico. Universidad del Cauca. 2008.

[43] **Larman, Craig.** *UML y Patrones.* Prentice Hall. 2003.

[44] **Sheldon, Lee.** *Character Development and Storytelling for Games.* Boston. Thomson Course Technology PTR, 2005.

[45] **Handler Miller, Carolyn.** *Digital Storytelling.* Oxford. ELSEVIER, 2004.

[46] **Parunak, Van Dyke and Odell, James**. *Engineering Artefacts for Multi-Agent Systems*, ERIM CEC. (1999).

[47] **Baiz Quevedo, Frank.** La construcción del personaje. [En línea] [Citado el: 9 de septiembre de 2007.]

[48] **Ferber, J. Gutknecht, O.** *A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems. Actas de conferencia. Proceedings of the Third International Conference on Multi-Agent Systems.* Boston. IEEE CS Press, 1998.

[49] **Russell, Stuart.** *Inteligencia Artificial: un enfoque moderno.* México. Prentice - Hall, 1996.

[50] **Wooldridge, Michael. Jennings, Nicholas y Kinny, David.** *The Gaia Methodology for Agent-Oriented Analysis and Design***.** 27, Boston: Kluwer Academic Publishers, 2000, Vol. I.

[51] **Gómez-Sanz, J.** Modelado de Sistemas Multi-Agente. PhD thesis, Departamento de Sistemas Informáticos y Programación, Universidad Complutense Madrid, 2002.

[52] **Bourg, David y Seeman, Glenn.** *AI for Game Developers .* United States : O'Reilly Media, 2004.

[53] **Kirmse, Andrew.** *Game Programming Gems 4.* Massachusetts. Charles River Media.

[54] **Ferreiro Barreiro, Divina.** *Estandar FIPA "Foundation For Intelligent Physical Agents".* Vigo, 2002.

[55] **Vieira, Sebastian.** *FIPA - ACL.* Boston s.n., 2000.

[56] **Botía, Juan A.** *Construcción de Agentes Software con JAVA+JADE* Madrid, 2005.

[57] **Catá, Jordi y Méndez, Alex.** *Comparativa de Motores Gráficos para Videojuegos.* Madrid, 2002.

[58] **Nischke, Benjamin.** *Professional XNA Game Programming: For Xbox 360 and Windows.* Hannover, Wrox Press, 2007.

[59] **Aguilar, J., Rivas F., Hidrobo F., Cerrada M***. Análisis y Diseño de Sistemas Multiagente Usando la Metodología MASINA.* Reporte Técnico – Universidad de los Andes (CEMISID), January 2004.

[60] **OGRE.** Ogre 3D. *Open source Graphics Engine.* [En línea] 03 de Marzo de 2007. [Citado el: 25 de Octubre de 2007.] http://www.ogre3d.org.

[61] **Lopez, Jaime, Yuly Botina.** Agentes Inteligentes para adaptar la presentación de contenidos para cursos de educación en línea**.** Tesis de Pregrado Universidad del Cauca. 2008.

[62] **Patricia Charlton, Yan Chen, Fredrik Espinoza, Abe Mamdani, Olle Olsson, Jeremy Pitt,Fergal Somers, and Annika Waern**. An open agent architecture supporting multimediaservices on public information kiosks. In Proceedings of PAAM'97, London, U.K 1997).

[63] **AMASE**: Agent-based Mobile Access to Information Services [En línea] 18 de Diciembre de 2001. [Citado el: 09 de Diciembre de 2008.] http://cordis.europa.eu/infowin/acts/analysys/products/thematic/agents/ch3/amase.htm

[64] **ABROSE**: A Cooperative Multi-Agent Based Framework for Electronic Marketplace [En línea] 18 de Diciembre de 2001. [Citado el: 09 de Diciembre de 2008.] http://cordis.europa.eu/infowin/acts/analysys/products/thematic/agents/ch3/abrose.htm

[65] **Chess D., Harrison C., Kershenbaum A,** "Mobile Agents: Are They A Good Idea?". IBM Research Division. T. J. Watson Research Center. 1995. [Citado el: 09 de Diciembre de 2008] URL - http://citeseer.ist.psu.edu/28994.html.

[66] **Mejía Salaza Maria Helena**, Aplicación de la Metodología GAIA, ANFORA ISSN 01216538 Volumen 19 Serie 19 Págs. 119 - 138 , Universidad Autónoma de Manizales, 2004

[67] **Ierache, Jorge Salvador.** *Elaboración de una aproximación metodológica para el desarrollo de software orientado a sistemas multiagente.* Madrid-España : s.n., 2003.

[68] **Gómez Sanz, Jorge J.** Metodologías para el desarrollo de sistemas multi-agente, Facultad de Informática, Universidad Complutense, Avda. Complutense Madrid, ISSN: 1137-3601, 2003

[69] **Morales Pedro Cuesta, Gómez Rodríguez Alma María, Rodríguez Martínez Francisco Javier**, Desarrollo de un Sistema Multi-agente con MaSE y JADE, Revista de la Asociación de Técnicos de Informática, ISSN 0211-2124, Nº. 170, 2004

[70] **Henao Cálad Mónica**, CommonKADS-RT: Una Metodología para el Desarrollo de Sistemas Basados en el Conocimiento de Tiempo Real, Tesis doctoral, Valencia, España. Abril de 2.001

[71] **Sánchez Crespo, Daniel Dalmau.** *Core Techniques and Algorithms in Game Programming.* Barcelona: New Riders Publishing, 2003.

[72] **J. Rumbaugh, M. Blaha, W. Premerlani and V.F Eddy.** *Object-Oriented Modeling and Design.* s. Prentice Hall, 1991.

[73] **Grasia, grupo de agentes software: Ingeniería y Aplicaciones.** Grupo de agentes software: Ingeniería y Aplicaciones. [En línea] [Citado el: 13 de agosto de 2007.] http://grasia.fdi.ucm.es/Spain/index.php.

[74] **Francisco Jose Gallego Duran, Faraon Llorens Largo, Ramon Rizo Aldeguer.** *Breve Análisis de algunas metodologías de diseño de SMA.* San Vicente del Raspeig s.n., 2004.

[75] **Dautenhahn, K.; Bond, A.H.; Canamero, L.; Edmonds, B. Socially** Intelligent Agents Creating Relationships with Computers and Robots Series, Vol. 3 ISBN: 978-1-4020-7057-0. 2006.