UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Sistema de creación de mundos virtuales:

Módulo de gráficas y simulación

Carlos Antonio Villagrán Juárez

Guatemala

2010

Sistema de creación de mundos virtuales:
Módulo de gráficas y simulación

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Sistema de creación de mundos virtuales: Módulo de gráficas y simulación

Trabajo de graduación presentado por Carlos Antonio Villagrán Juárez

para optar al grado de Licenciado en Ingeniería en Ciencias de la Computación

Guatemala

2010

Vo. Bo.:

(f)_____

(Ing. Sergio Izquierdo)

Tribunal:

(f)_____

(Ing. Sergio Izquierdo)

(f)_____

(Ing. Luis Barrera)

(f)_____

(Ing. Erik Véliz)

Fecha de aprobación: Guatemala, 27 de mayo de 2010

PREFACIO

En este trabajo se desarrolló uno de los módulos del Sistema de Creación de Mundos Virtuales, el Módulo de Gráficas y Simulación. Dicho módulo, es el encargado de representar el mundo de forma gráfica y manejar la interacción con el usuario.

El sistema nació de la idea de dos estudiantes de computación que querían desarrollar un proyecto que, a futuro, pudiera ser algo que beneficie a toda la humanidad. Así mismo, se deseaba utilizar en un sólo proyecto la mayoría de los conocimientos aprendidos durante los estudios en la universidad. Así nació el sistema, en tiempos donde las tecnologías interactivas están siendo utilizadas cada vez más por todas las personas.

Inicialmente el proyecto estaba enfocado directamente a crear mundos virtuales para asistir a maestros en la enseñanza, sin embargo, debido a complicaciones, se cambió el enfoque hacia la creación de mundos virtuales en general.

El proyecto se realizó durante 18 meses bajo el curso de Taller de Trabajo Profesional. Inicialmente era un proyecto planeado para 12 meses el cual fue extendido debido a retrasos en el desarrollo.

Agradezco a mi familia, por apoyarme durante el desarrollo de este proyecto y durante mi vida, y nunca dudar de mí ni de lo que me propongo. Por darme todo lo que me han dado, desde la educación hasta los consejos que me han ayudado a construir mi vida y proponerme nuevas metas.

Agradezco a mis amigos, por ser sinceros conmigo y decirme lo que realmente piensan. Por todos los momentos que hemos pasado juntos y que han ayudado a mi vida. Por motivarme a intentar nuevas cosas y enseñarme a apreciar las que ya he logrado. También agradezco a los que me apoyaron brindándome los recursos y materiales que se utilizaron en la creación de esta herramienta.

Agradezco a mis compañeros de trabajo, por ser comprensivos en los momentos en que lo necesito. Por enseñarme a ser parte de la sociedad y a desarrollar productos funcionales. Por hacerme pensar sobre temas más allá del trabajo y reflexionar sobre mis decisiones.

Agradezco a la Universidad del Valle, por enseñarme muchos de los conocimientos que hoy poseo sobre computación y otras áreas. Por dejarme pensar libremente y permitirme presentar este proyecto como trabajo de graduación. Por darme las herramientas y las oportunidades que han influido mucho en lo que soy y lo que puedo hacer.

Agradezco a mi compañera de este trabajo por haber tenido confianza en mí y en este proyecto. También por haberme ayudado en todos los momentos que lo necesitaba, tanto en asuntos técnicos como en asuntos personales. Por último, por tener una gran exigencia por que todo se haga bien y por haber aceptado mis decisiones.

ÍNDICE

LISTADO DE GRÁFICOS

RESUMEN

En este sistema, un mundo virtual es una representación gráfica tridimensional que simula un ambiente. Esta representación contiene entidades virtuales y permite que un usuario interactué con estas a través de un personaje que lo representa dentro de la simulación.

La idea principal del Sistema de Creación de Mundos Virtuales es proveer las herramientas para construir mundos virtuales que puedan ser utilizados por múltiples usuarios a través de internet.

El sistema está dividido en dos módulos, en este trabajo se realizó el de gráficas y simulación. Se utilizó otro módulo, el módulo de administración de información y comunicaciones, para obtener los datos que definen un mundo virtual, sus entidades y personajes. Estos datos se necesitan para representar gráficamente los mundos.

En este módulo se representan de forma gráfica los mundos, se manejan los dispositivos de entrada y la interacción con el usuario. También corresponde a este módulo realizar las simulaciones de eventos y fuerzas físicas de un mundo virtual.

Parte de este módulo contempla también el desarrollo de la interfaz gráfica que el usuario empleará para identificarse y para consultar información sobre el mundo y su personaje. Esta interfaz se asemeja a la de una aplicación tradicional, ya que hace uso de ventanas para mostrar la información.

El resultado del desarrollo de este módulo es una aplicación multiplataforma mediante la cual el usuario utiliza el mundo virtual.

Este trabajo de graduación se presenta bajo la modalidad de Trabajo Profesional. Se realizó durante el año 2009 y parte del año 2010.

1. INTRODUCCIÓN AL SISTEMA

La palabra “mundo” en este contexto se refiere a un ambiente, el cual es considerado autónomo por parte de sus habitantes. La palabra “virtual” se refiere a algo que no existe, capaz de provocar el efecto de algo existente (Bartle 2003).

Los mundos virtuales son implementados por computadoras y simulan un ambiente. Algunas de las entidades en este ambiente actúan directamente bajo el control de personas individuales, siendo estos los “habitantes” del mundo. Generalmente una persona tiene un único habitante (llamado personaje o avatar) a quien controla y con quien se identifica. Con sus acciones, los habitantes realizan cambios en el ambiente. Varias personas pueden afectar el mismo ambiente al mismo tiempo. Por esta razón se dice que el mundo es compartido o multiusuario (Bartle 2003).

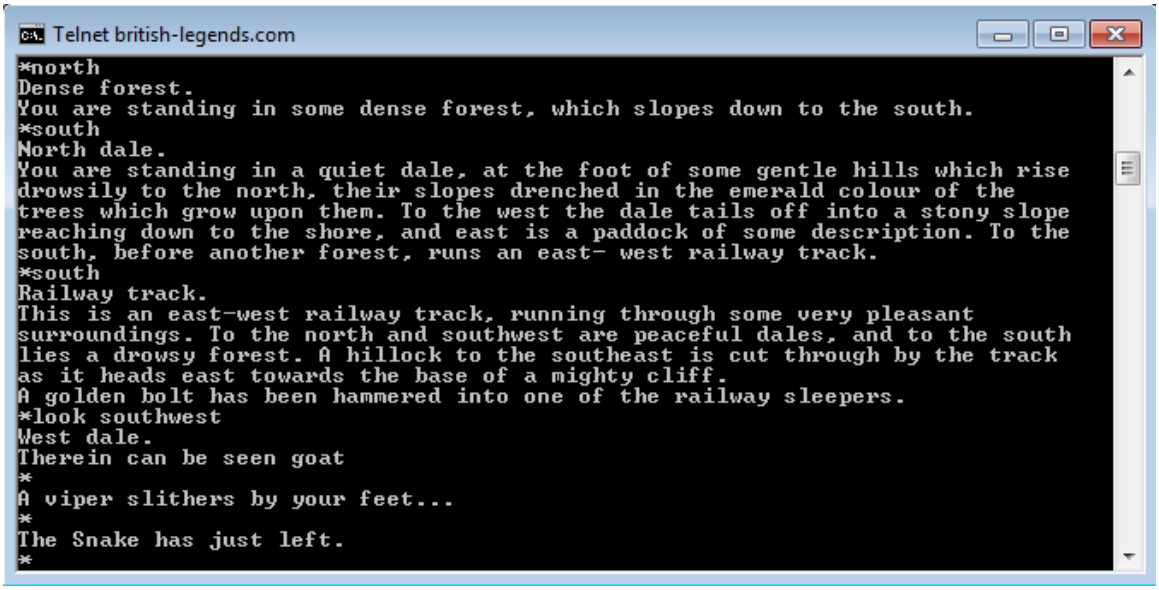
Para cumplir con la característica de que el mundo debe ser autónomo, el ambiente debe tener una serie de reglas por las cuales se rige y evoluciona. Estas reglas son las que permiten a los usuarios modificar el estado del mundo.

Un mundo virtual debe ser persistente, lo que significa que debe seguir funcionando aunque no se encuentren habitantes dentro de él. Otra de las características de un mundo virtual es que las respuestas a la interacción de los usuarios con el mismo se producen en tiempo real. Esto quiere decir que el sistema responde dentro del tiempo esperado por el usuario, según lo que tardaría realizar la operación en el mundo real. En este caso el tiempo de respuesta debe ser inmediato para la mayoría de operaciones.

1.1. Historia

Los mundos virtuales surgieron a finales de los años 70 como juegos de computadora originalmente conocidos como MUD's (Multi User Dungeons), pues MUD era el nombre del primero de estos en prosperar (Bartle 2003). Estos juegos proponían aventuras en un mundo de fantasía, de las cuales el usuario era partícipe a través de su personaje. Inicialmente, los ambientes y los eventos que ocurrían dentro de ellos eran descritos utilizando texto (Figura 1). Los MUD's lograron demostrar que aunque no se utilizara una tecnología de despliegue sofisticada, la representación rica y consistente de un espacio virtual puede ser vívidamente experimentada en la imaginación del usuario (Mitchel 1995).

Figura 1 Captura de pantalla de MUD



Debido a la aparición del Internet y a la evolución de la computación en los siguientes años, los MUD's se propagaron y varios programadores empezaron a implementar sus propios mundos virtuales. Uno de los que vale la pena mencionar es TinyMUD, el cual no tenía las características de un juego, sino permitía que los usuarios crearan elementos dentro del mundo virtual. Las personas pasaban el tiempo creando

ambientes y objetos y hablando de sus creaciones (Bartle 2003). Los mundos virtuales inspirados en TinyMUD se llamaron "mundos virtuales sociales", y a diferencia de los "mundos virtuales de juego" no tienen una trama definida, no requieren que el usuario realice misiones ni reciba recompensas establecidas, sino se utilizan para que los usuarios creen sus propios objetos y se comuniquen y convivan con el resto de habitantes.

En la actualidad, los mundos virtuales basados en texto se han convertido en espacios digitales en tercera dimensión, con millones de usuarios, funcionamiento político y social complejo y con una gran variedad de características en su población (Peachey, *et al*. 2010). Como máximo exponente actual de los "mundos virtuales sociales" se puede mencionar a Second Life, desarrollado por Linden Lab (Figura 2). Al igual que TinyMUD, la base de Second Life es el contenido creado por los usuarios, para esto incluye herramientas que permiten construir objetos tridimensionales a partir de formas básicas y además herramientas de programación que permiten diseñar contenido interactivo. Las personas pueden crear y distribuir objetos, incluso ponerlos a la venta y cambiar el dinero obtenido en el mundo virtual por dinero real. Los usuarios de Second Life aprovechan este sistema como medio de socialización, para hacer negocios, como medio de expresión artística, para educación y muchas otras actividades (Gollub 2007).

Los "mundos virtuales de juego" o mejor conocidos como MMORPG (Massive Multiplayer Online Role Playing Games) se han convertido en una industria multimillonaria del entretenimiento (Peachey, *et al*. 2010). Estos juegos siguen siendo una historia de fantasía donde el usuario crea un personaje que tiene un rol o trabajo específico y un conjunto de habilidades. Dentro del mundo virtual existen misiones y tareas que los personajes deben realizar para cumplir con su rol dentro del mundo. Al hacer las actividades el personaje puede subir de nivel, incrementar sus habilidades,

conseguir nuevas armas y equipo y así especializarse en su trabajo y poder obtener mejores recompensas.

Normalmente se necesita que varias de las actividades sean realizadas en equipos donde participan usuarios con diferentes roles, por lo que se necesita que además de especializarse en las habilidades de su personaje, el usuario también sea capaz de integrar estas habilidades con las de sus compañeros de equipo (Gee s.f.). Uno de los MMORPG's más famosos es World of Warcraft (Figura 3) desarrollado por Blizzard Entertainment, que en la actualidad cuenta con más de doce millones de subscriptores (Peachey, *et al*. 2010).

Figura 2 Captura de pantalla del mundo virtual social: Second Life

Figura 3 Captura de pantalla del MMORPG World of Warcraft

1.2. Mundos virtuales y educación

Desde hace algunos años, varias universidades e instituciones académicas de nivel superior utilizan mundos virtuales sociales, como Second Life para realizar actividades educativas. El interés por esta herramienta radica en que además de permitir la socialización y el trabajo en equipo, permite que el usuario pueda construir objetos dentro del mundo virtual. Esto ha sido aprovechado por los educadores, quienes han utilizado Second Life para construir campus y aulas virtuales, bibliotecas y museos (Kemp y Livingstone 2006). También han utilizado esta herramienta para que los estudiantes realicen proyectos que serían muy difíciles de realizar en el mundo real, como la implementación de un negocio o tienda virtual (Mason 2007), la creación de mini ciudades, y el entrenamiento para manipulación de equipo sofisticado (Kemp y Livingstone 2006).

La mayoría de interés hacia los mundos virtuales en el campo de la educación ha sido para los mundos virtuales sociales, como Second Life, pues Second Life es completamente libre de una narrativa, no tiene un escenario ni trama definida, por lo que los maestros tienen la libertad de construir sus propias metáforas y crear escenarios específicos (Kemp y Livingstone 2006). Esta es una gran ventaja para los educadores, pero si se utilizara un MMORPG específicamente diseñado para la educación, la trama, el escenario y los elementos de juego serían muy útiles como herramienta educativa. Dentro de un juego el usuario encuentra objetivos que debe cumplir dentro de la simulación, para alcanzar estos objetivos el jugador debe reconocer los problemas, determinar las reglas de la simulación y decidir qué elementos del juego puede utilizar para resolver estos problemas (Gee s.f.). Específicamente se mencionan las siguientes características de un video juego que pueden ser útiles para la educación :

- El crear una simulación temática de un sistema complejo y hacer que el usuario sea parte, y esté dentro de la simulación, hace que él sea capaz de entender el sistema como un todo y no sólo aprender eventos que parezcan locales o que suceden al azar (Gee s.f.).
- La mente humana funciona como un simulador, capaz de imaginar un escenario y colocar a la persona que lo imagina en diferentes roles dentro de la acción. Tal como se hace también dentro de un juego de video. Según el autor James Paul Gee, el aprendizaje se da al realizar generalizaciones de las diferentes experiencias y simulaciones de un sujeto. Por lo que los videojuegos son una herramienta natural para el aprendizaje (Gee s.f.).
- Los objetivos de un juego pueden alcanzarse por múltiples rutas, pero al decidir qué camino tomar, el usuario debe considerar las consecuencias y evaluar los diferentes trayectos para determinar cuál, según él, es el mejor camino a seguir. Al ejecutar su plan, si no obtiene los resultados esperados, el usuario debe de volver a empezar el proceso. Por lo que para complementar las tareas del

juego se requiere la formulación de hipótesis, experimentación y análisis de los resultados (Mayo 2009).

- En un juego, la inteligencia puede distribuirse dentro de los diferentes elementos del mismo. Por ejemplo, el hacer que el personaje tenga ciertas habilidades necesarias para completar una tarea, permite al usuario empezar la actividad con cierto grado de efectividad, sin que él propiamente tenga el completo conocimiento de lo que va a realizar, sino que lo tenga parcialmente a través del personaje. Utilizando el personaje el jugador aprende luego de prueba, error y retroalimentación (Gee s.f.).

1.3. Sistema de creación de mundos virtuales

La idea original de este proyecto era crear un MMORPG que pudiera ser utilizado para educación, pero debido a la dimensión muy grande del plan y a la falta de especialistas en el área de educación en el equipo de trabajo, se decidió reducir el alcance del proyecto a la realización de una herramienta de software que permita crear mundos virtuales. Con esta herramienta de software se podrá diseñar el MMORPG en el futuro.

El sistema de creación de mundos virtuales se realizó en dos módulos: Módulo de administración de la información y comunicaciones y módulo de gráficas y simulación. En la siguiente sección de este documento se especifican las características que tiene el sistema de creación de mundos virtuales en conjunto.

2. ESPECIFICACIÓN DEL SISTEMA

El sistema de creación de mundos virtuales tiene como objetivos:

- Crear la infraestructura de software que permita definir mundos virtuales.
- Crear una interfaz gráfica que permita definir mundos virtuales.
- Crear una interfaz gráfica que permita utilizar los mundos virtuales.

Para esto se definen los siguientes componentes del sistema.

2.1. Componentes del sistema

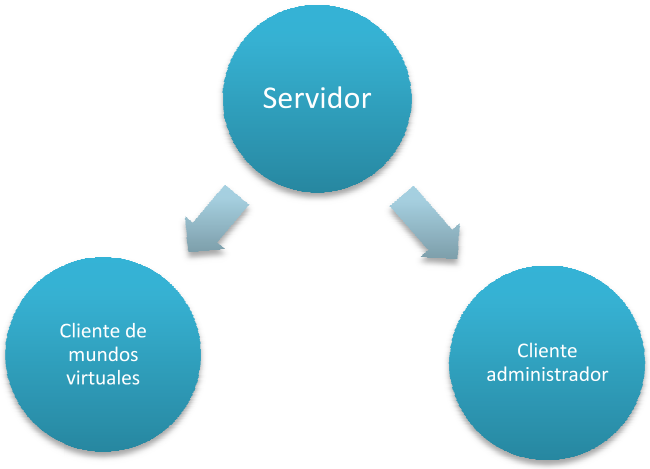
2.1.1. Servidor. El servidor es el lugar en el que "residen" los mundos virtuales, mantiene la copia que hace que el mundo virtual sea común a todos los usuarios. Los clientes se conectan al servidor a través de una red y obtienen de él la información del estado del mundo virtual y de los demás clientes conectados.

2.1.2. Cliente de mundos virtuales. Es una interfaz gráfica en tres dimensiones que permite que un usuario interactúe con un mundo virtual. Obtiene los datos del estado del mundo virtual en el servidor, y con éstos representa el ambiente que es explorado por el usuario.

2.1.3. Cliente Administrador. Es una interfaz gráfica con el propósito de crear y administrar los mundos virtuales. Esta interfaz está disponible sólo para el administrador del sistema.

Todos los clientes interactúan con el mismo servidor. La interacción entre los componentes del sistema se muestra en la Figura 4 .

Figura 4: Relación entre los componentes del sistema



2.2. Definiciones

En el sistema de creación de mundos virtuales se utilizan las siguientes definiciones.

2.2.1. Mundo virtual. Es la simulación computarizada de un ambiente. En este caso es una representación gráfica tridimensional que contiene entidades y personajes, y permite que un individuo interactúe con estos a través de un avatar. La interacción entre varios individuos también es posible, compartiendo el mismo mundo virtual.

Los mundos virtuales cumplen leyes. Éstas pueden ser leyes físicas, como la gravedad, o leyes de la simulación, como una secuencia de actividades que deben realizarse para dar por terminada una misión.

Los mundos pueden ser compartidos por todos los usuarios, y también instancias (copias) para un grupo de usuarios específico.

2.2.2. Entidad del mundo virtual (Ítem). Es cualquier objeto que se encuentre dentro del mundo virtual y no sea directamente controlado por el usuario. Se incluyen

los objetos de escenario y decoración, los objetos que interactúan con los avatares y los personajes controlados por computadora NPC (Non Player Character). Los ítems pueden tener características y propiedades específicas que definen su funcionalidad. Las entidades del mundo virtual pueden moverse dentro del ambiente siguiendo una trayectoria definida.

2.2.3. Avatar (Personaje). Es una representación del usuario como un modelo en tres dimensiones dentro del mundo virtual. Es capaz de recorrer el espacio tridimensional y de interactuar con el mismo.

La interacción se realiza mediante mensajes de texto, animaciones e ingreso de datos por medio de mouse y teclado. De esta forma el avatar podrá realizar actividades y misiones dentro del mundo virtual.

Los personajes pueden poseer ítems. Los ítems pueden ser necesarios para realizar actividades dentro de la simulación o decorativos, como ropa y accesorios.

Los personajes tienen propiedades que definen como estos interactúan con el mundo virtual. Estas pueden ser físicas, como velocidad, fuerza y peso, y de control, como puntuación.

2.2.4. Actividades. Las actividades que se realizan dentro de la simulación son:

- Exploración del mundo virtual: El personaje puede recorrer el mundo virtual y observar los ítems que se encuentran dentro de él.
- Información acerca de los ítems y NPC: Los ítems y NPC pueden darle información al usuario acerca de lo que estos representan dentro del mundo virtual.
- Misiones: Los ítems y NPC pueden pedirle al personaje que realice determinadas actividades a cambio de una recompensa. La recompensa puede ser un aumento en su puntuación o un ítem.
- Obtención de ítems: El personaje puede colectar ítems.

Las actividades pueden realizarse de manera individual, o en conjunto con varios personajes a la vez.

2.3. Descripción general del sistema

2.3.1. Requerimientos funcionales

2.3.1.1. Sistema

- Permitir la creación de mundos virtuales.
- Comunicar los datos de los mundos virtuales y usuarios a través de internet.
- Permitir la actividad de múltiples mundos virtuales al mismo tiempo.

2.3.1.2. Mundo virtual

- Representar el ambiente de la simulación en tres dimensiones.
- Cumplir leyes predefinidas. físicas y de simulación.
- Responder a la interacción con el usuario según la definición en la sección 2.2.3.
- Permitir la interacción de varios usuarios en el mismo mundo virtual.
- Implementar las actividades según su definición en la sección 2.2.4.

2.3.1.3. Personajes

- Representar al usuario en el mundo virtual.
- Explorar el mundo virtual en tres dimensiones.
- Realizar actividades dentro del mundo virtual según la definición de la sección 2.2.4.
- Interactuar con los elementos del ambiente y con otros personajes a través de texto, de animaciones y de ingreso de datos a través de teclado y mouse.
- Adquirir y administrar ítems.
- Adquirir y administrar misiones.

2.3.2. Tipos de usuarios. El sistema está dirigido a dos tipos de usuarios:

- Administrador del sistema: Encargado de administrar el sistema en general y crear los mundos virtuales.
- Usuario general: Utiliza los mundos virtuales.

2.3.3. Ambiente operativo. El software está destinado a utilizarse en computadoras personales y debe poder ser ejecutado sobre los siguientes sistemas operativos: Windows, Linux, Mac OS.

2.3.4. Requerimientos no funcionales

- El software debe ser desarrollado con herramientas de distribución gratuita, ya que no se cuenta con recursos para la adquisición de herramientas de desarrollo de software.
- El software debe ser multiplataforma y de fácil instalación.

2.4. Requerimientos detallados de información

2.4.1. Mundos. Un mundo debe tener:

- Un nombre que lo identifique.
- Un estado que indica si es:
  - Plantilla: Puede ser utilizado para generar copias de él mismo.
  - Instancia: Copia individual del mundo para un grupo de usuarios específico.
  - Único: No tiene copias y es compartido por todos los usuarios.

Un mundo puede tener:

- Propiedades
  - Leyes que físicas, como la gravedad.
  - Tipo de terreno.
  - Tipo de cielo.
- Ítems y NPC.

- Personajes de los usuarios.
- Regiones: Definen áreas específicas dentro del mapa. Se forman por un conjunto de puntos dentro del espacio tridimensional. Los puntos son de la forma [X, Y, Z].

2.4.2. Ítems. Un ítem debe tener:

- Un nombre que lo identifique.
- Un estado que indica si es:
  - Plantilla: Puede ser utilizado para generar copias de él mismo.
  - Instancia: Copia individual del mundo o personaje que lo posee.
  - Único: Instancia común para todos los mundos y personajes que lo utilizan.

Un ítem puede tener:

- Una posición dentro del mundo virtual: Representada por una coordenada del espacio tridimensional en forma de punto [X, Y, Z] y un vector de rotación, de la forma [RX, RY, RZ].
- Propiedades.
- Un modelo tridimensional asociado.
- Indicador de la posición con respecto al ítem o personaje al que pertenece.
- Un conjunto de eventos a los que responde.
- Una trayectoria.
- Otros ítems.

2.4.3. Trayectoria. Está formada por una serie de posiciones, por las cuales un ítem debe moverse constantemente.

Una trayectoria tiene:

- Un nombre.
- Un estado, que puede ser plantilla, instancia o única.
- Un conjunto de puntos [X, Y, Z] , [RX, RY, RZ] que determinan la posición y rotación del objeto dentro del mundo.

2.4.4. Personajes. Un personaje tiene:

- Un nombre que lo identifica.
- Una posición dentro del mundo virtual: Representada por una coordenada del espacio tridimensional en forma de punto [X, Y, Z] y un vector de rotación, de la forma [RX, RY, RZ].
- Un estado que indica si es:
  - Plantilla: Puede ser utilizado para generar copias de él mismo.
  - Instancia: Copia individual del usuario.
- Propiedades:
  - Un modelo tridimensional asociado.
  - Peso del objeto, para aplicarle gravedad.
  - Velocidad, para caminar.
  - Fuerza para saltar.
  - Puntos: Adquiridos por completar actividades.
- Ítems
- Misiones

2.4.5. Misiones. Las misiones tienen:

- Un nombre.
- Un estado. Que puede ser plantilla, instancia y único.
- Un conjunto de eventos, que hay que cumplir para dar por terminada la misión.
- Un indicador del progreso de la misión, para cada personaje que la obtenga.

2.4.6. Eventos. Un evento es un conjunto de condiciones que al cumplirse ejecutan una cadena de acciones.

Los eventos tienen:

- Un estado: Indica si el evento está activo o inactivo.
- Condiciones: Las condiciones que deben cumplirse.

- Relación entre las condiciones: Permite establecer una relación lógica entre las condiciones listadas.
- Un conjunto de acciones, que se ejecutan al cumplirse las condiciones según la relación.
-

2.4.6.1. Condiciones de los eventos. Las condiciones de los eventos necesitan de:

- Un tipo: Identifica a la condición y la información que debe de ser analizada para que la condición se cumpla. Ej. Mouse_Click
- Parámetros: Los datos que se utilizan en el análisis de la información.

2.4.7. Acciones. Realizan operaciones que modifican las propiedades y funcionalidades del mundo y sus entidades.

Las acciones tienen:

- Operación a realizar: Identifica la operación a realizar. Ej: Agregar ítem a un personaje.
- Parámetros: Los datos a utilizar en la operación.

2.4.8. Usuarios. Los usuarios tienen:

- Nombre de usuario.
- Nombre y apellido.
- Género.
- Fecha de nacimiento.
- Estado, que indica si está activo o inactivo.
- Rol: Indica el papel del usuario dentro del sistema y determina los permisos que este posee.

2.5. Limitaciones del proyecto

- Eventos y acciones: Por el momento se cuenta con un conjunto predefinido de eventos y acciones que el sistema detecta y ejecuta. Estos son los básicos utilizados en la creación del mundo virtual de pruebas. En el futuro se piensa ampliar el conjunto de eventos y acciones o incluso proveer de alguna forma en la que los usuarios del sistema puedan definir estos eventos y acciones.
- Seguridad Informática: Para hacer más sencillo el trabajo no se toman en cuenta todas las medidas de seguridad que se deberían de implementar en un proyecto en producción. Los aspectos de seguridad en los que se trabajan son los básicos para cumplir con la funcionalidad del proyecto.
- SDK: Se obtuvo como sugerencia la implementación de un SDK (Software Development Kit) para permitir que se desarrollen mundos virtuales y funcionalidad extra para el sistema por medio de programación. Pero, por el momento no se piensa desarrollar este SDK, pues la idea del proyecto es poder crear los mundos virtuales por medio de una interfaz gráfica y no mediante programación.

2.6. Diseño preliminar

2.6.1. Herramientas de desarrollo

- Lenguaje de programación: Java. Permite que el cliente sea ejecutado en cualquier plataforma.
- Ambiente de Desarrollo de Software: Eclipse. De distribución gratuita y de código abierto.

2.6.2. Arquitectura del sistema El sistema se separa en dos módulos:

2.6.2.1. Módulo de administración de la información y comunicaciones

Este módulo es el encargado de almacenar la información de los mundos virtuales en el servidor. También se encarga de crear un método para hacer que esta información pueda ser accedida por los clientes. Además implementa el cliente administrador definido en la sección 2.1.3.

2.6.2.2. Módulo de gráficas y simulación El encargado de proveer la interfaz gráfica tridimensional que representa los mundos virtuales, definida en la sección 2.1.2, simular las fuerzas físicas y eventos que suceden en el mundo y de manejar la interacción con el usuario.

3. INTRODUCIÓN AL MÓDULO

Para representar mundos virtuales de forma visual se hace uso de las gráficas por computadora, estas se utilizan para mostrar objetos y ambientes tridimensionales en una pantalla.

El sistema de creación de mundos virtuales está dividido en dos módulos. Esta división se realizó agrupando funciones del sistema que tienen objetivos en común y así poderse trabajar por separado simultáneamente.

El módulo de administración de la información y comunicaciones se encarga de almacenar los datos necesarios para la representación gráfica de los mundos. Este también pone a disposición un método para que estos datos sean accesibles por el otro módulo.

El módulo de Gráficas y Simulación se encarga de representar los datos del mundo y de los personajes en una pantalla. Este también permite que los usuarios interactúen con los objetos del mundo.

En este trabajo se llevó a cabo el desarrollo del módulo de gráficas y simulación. Se utilizó el módulo de administración de la información y comunicaciones para obtener los datos del mundo virtual y sus componentes mediante un servicio web.

Se creó una aplicación desarrollada en el lenguaje de programación Java. También se utilizó un motor de gráficas, desarrollado en el mismo lenguaje y que además hace uso de OpenGL, para simplificar el desarrollo de la representación gráfica de los mundos en tercera dimensión

4. DEFINICIÓN DE REQUERIMIENTOS

La intención de esta aplicación es proveer el medio para que el usuario pueda relacionarse con el mundo virtual. En esta aplicación se representa gráficamente el mundo virtual y sus componentes, también es en donde el usuario es representado por un personaje que le permite interactuar con el mundo.

4.1. La aplicación debe:

- Responder a interacción con el usuario.
- Mostrar una interfaz gráfica.
- Manejar los dispositivos de entrada.
- Considerar el desempeño en equipos de bajo nivel.
- Obtener datos de los mundos por medio del módulo de administración de la información y comunicaciones.
- Ser desarrollada en el lenguaje de programación Java y el ambiente de desarrollo Eclipse.
- Ser multiplataforma.

4.2. Al representar el mundo virtual se debe:

- Representar al usuario con un personaje.
- Representar el terreno.
- Representar el cielo.
- Representar los ítems del mundo.
- Ejecutar animaciones en los ítems animados y en personajes.
- Cumplir leyes predefinidas, físicas y de simulación.
- Delimitar el área del mundo virtual que los personaje pueden explorar
- Detectar colisiones.
- Permitir la interacción de varios usuarios en el mismo mundo virtual.
- Identificar cuando los personajes entran al mundo virtual y cuando salen.
- Controlar la posición en el mundo de todos los personajes conectados..

- Dar misiones a los personajes a través de los NPC.
- Dar información a los personajes acerca de los ítems y NPC’s en el mundo virtual.

4.3. Un usuario que utilice la aplicación debe poder:

- Identificar o crear un usuario del sistema.
- Crear un personaje.
- Escoger un personaje.
- Escoger un mundo virtual.
- Explorar el mundo virtual en tres dimensiones.
- Caminar y saltar.
- Realizar actividades dentro del mundo virtual.
- Obtener información acerca del personaje, sus misiones e ítems.
- Interactuar con los elementos del ambiente y con otros personajes.
- Hacer uso del chat para comunicarse con otros personajes.
- Adquirir y administrar ítems.
- Adquirir y administrar misiones.
- Equipar ítems.
- Salir de la aplicación.

5. OBJETIVOS

5.1. Objetivos generales

- Representar mundos virtuales de forma gráfica.
- Permitir la interacción entre el usuario y el mundo virtual.

5.2. Objetivos específicos

- Acceder a los datos del mundo virtual mediante el módulo de administración de la información y comunicaciones.
- Manejar los dispositivos de entrada y salida del computador para permitir la interacción con el usuario.
- Desarrollar la aplicación para que sea multiplataforma.
- Utilizar herramientas de desarrollo gratuitas.

6. MARCO TEÓRICO

6.1. Gráficas por computadora.

Es cualquier uso de una computadora para crear o manipular una imagen (Sherley 2005). Existen varios conceptos importantes dentro del área de las gráficas por computadora que se deben entender.

6.1.1 Modelar. Es crear las especificaciones de las propiedades de forma y apariencia de un objeto de manera que puedan ser guardadas en una computadora.

6.1.2 Renderizar. Es un término heredado del arte y se trata de crear imágenes sombreadas de modelos tridimensionales.

6.1.3 Animación. Es una técnica para crear una ilusión de movimiento a través de una secuencia de imágenes.

6.1.4 Modelos geométricos 3D. Son una parte clave de los programas de gráficas por computadora. Estos modelos describen objetos mediante primitivas matemáticas, como esferas, cubos, conos y polígonos. Otros modelos se representan por triángulos con vértices compartidos, estos se conocen como una malla formada de triángulos (Triangle mesh). (Sherley 2005). Algunos de los usos de las gráficas por computadora son:

6.1.5 Video juegos. utilizan modelos 3D sofisticados y algoritmos de renderizado.

6.1.6 Diseño Asistido por Computadora CAD (Computer Aided Design). Uno de los mayores usos de las gráficas por computadora es en los procesos de diseño, especialmente en los sistemas de ingeniería y de arquitectura. Los objetos primero se representan por una armazón de cables (wireframe) que muestra el bosquejo general de la figura y sus objetos internos. De esta forma se permite ver rápidamente los efectos que tienen algún cambio en el objeto.

6.1.7 Arte por computadora. Los artistas gráficos usan una gran variedad de métodos de computación que proveen medios para facilitar el diseño de formas y animaciones específicas de objetos.

Los sistemas de simulación de brochas, lápices, etc. son utilizados para simular diferentes herramientas que se utilizan en el arte tradicional. Normalmente se utilizan mecanismos de entrada como un lápiz inalámbrico sobre un panel en donde se simula escribir como si fuera un papel o un lienzo real.

Los artistas también utilizan asistencia de funciones matemáticas para crear efectos visuales, mediante operaciones que crean gráficas al azar o que forman patrones. (Hearn y Baker 1996)

6.1.8 Simulaciones virtuales. Los modelos generados por computadora se pueden utilizar para asistir en la educación y para asistir a los usuarios en el aprendizaje del uso de sistemas. Para esto se crean sistemas especializados que crean una representación visual de los componentes y de las condiciones necesarias para aprender un tema específico. El ambiente simulado provee de ventajas que no se tienen en los sistemas reales, como la habilidad de repetir un caso especifico las veces que sea necesario. (Bartle 2003).

6.1.9 Interfaz Gráfica. La mayoría de aplicaciones de computadora proveen una interfaz gráfica. Una forma muy común es mediante un sistema de manejo de ventanas que permite al usuario tener múltiples vistas al mismo tiempo. En cada ventana se pueden observar datos y representaciones distintas. Las interfaces gráficas proveen una forma fácil y cómoda para que el usuario interactúe con el sistema utilizando los dispositivos de entrada. (Hearn y Baker 1996)

6.2. Java.

Es un lenguaje de programación de alto nivel, es compilado lo cual significa que para ejecutarse en una computadora, el código de alto nivel primero debe transformarse a instrucciones binarias a ser ejecutadas por una máquina. A diferencia de otros

lenguajes, Java al compilar no produce instrucciones binarias específicas para la máquina en que se utilizará. Java produce código binario para una plataforma conocida como Máquina Virtual de Java (Java Virtual Machine). El código binario no es ejecutado directamente por la máquina en donde se ejecuta el programa, si no que es ejecutado por un programa que emula la máquina virtual de Java. Por lo tanto, el código que es generado en otros lenguajes varía de sistema en sistema mientras que en Java es el mismo por lo que se puede escribir un programa que podrá ser ejecutado en múltiples plataformas sin necesidad de cambios.

Hay un problema con esta ventaja que Java provee. El problema es que no se puede esperar que los programas que son emulados por otro programa, como lo hace Java, se ejecuten tan rápidamente como los que se ejecutan directamente en la máquina real. Sin embargo se han hecho mejoras para hacer que Java sea más competitivo en términos de velocidad. (Cooper 2000)

6.3. OpenGL.

Consiste de unos 150 comandos distintos que se utilizan para especificar los objetos y las operaciones necesarias para producir aplicaciones interactivas en tercera dimensión.

Está diseñado para ser optimizado e independiente del hardware para ser implementado en múltiples plataformas de hardware. Para lograr esto no se incluyen comandos para realizar tareas de manejo de ventanas ni para manejar los dispositivos de entrada, por lo que es parte del trabajo del usuario hacer estas funciones para el sistema en donde se desee ejecutar la aplicación. OpenGL tampoco provee de comandos de alto nivel para describir modelos en tercera dimensión. Esos comandos permitirían describir objetos relativamente complicados, como automóviles, partes del cuerpo, etc. En OpenGL, el desarrollador debe de construir los modelos complejos utilizando un pequeño conjunto de figuras geométricas. (Puntos, líneas, polígonos). (Neider y Tom 1997)

6.4. Motor gráfico.

El objetivo de un motor gráfico es proveer facilidades para realizar operaciones. El trabajo de un motor gráfico es realizar todas las operaciones de bajo nivel, como comunicarse con los dispositivos de gráficas, asignar las propiedades de dibujado, transformar los modelos y lidiar con funciones matemáticas complejas, como la rotación de matrices. Este trabajo es necesario hacerlo, pero no debe desviar la atención del desarrollador que tiene otros objetivos específicos para el uso de la aplicación.

Al utilizar un motor gráfico se proveen muchas ventajas, aparte de simplificar el desarrollo. Provee funciones optimizadas para realizar ciertas operaciones que de haber sido programadas por el desarrollador de la aplicación no hubieran tenido tan alta prioridad ya que ese no era el objetivo principal del desarrollo. Otro factor es el hecho de que algunos motores gráficos dan la posibilidad de crear aplicaciones multiplataforma, esto quiere decir que la aplicación ejecutable podrá ser utilizada en diferentes sistemas sin necesidad de hacer cambios. (Zerbst y Duvel 2004)

6.5. Servicio web.

Es cualquier servicio que esté disponible a través de Internet, usa un sistema de mensajes XML estandarizado y no depende de un sistema operativo o lenguaje de programación especifico.

Un servicio web puede tener las siguientes propiedades:

- Se describe a sí mismo. Cuando se publica un servicio se debe publicar también una interfaz pública del servicio. Como mínimo debe tener una documentación que se pueda ser leída por los desarrolladores para utilizar el servicio fácilmente.
- Ser descubrible. Cuando se publica un servicio web debe ser relativamente simple la forma en que se avisa que se ha publicado.

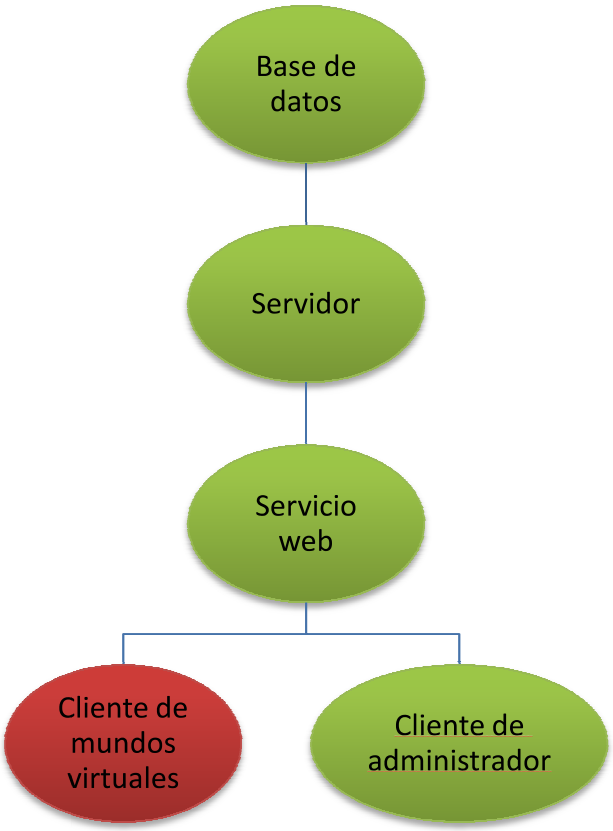
Lo que los servicios web proveen es un modelo centrado en las aplicaciones, esto permite la integración de aplicaciones de forma rápida mediante un estándar de cómo se deben realizar las comunicaciones entre éstas. (Cerami 2002)

7. DESARROLLO DE LA APLICACIÓN

La aplicación se desarrolló en el lenguaje de programación Java, ya que este lenguaje resuelve el problema de hacer que la aplicación sea multiplataforma. Se utilizó el ambiente de desarrollo Eclipse por ser una herramienta gratuita que permite el desarrollo de aplicaciones en Java.

La aplicación creada es un cliente que obtiene los datos necesarios de los mundos virtuales utilizando el servicio web que provee el módulo de administración de la información y comunicaciones. En la Figura 5 se puede observar la relación de la aplicación con el resto del sistema.

Figura 5 Estructura del sistema



7.1. Interacción módulo de administración de la información y comunicaciones.

El primer paso para poder crear el cliente, que representa visualmente el mundo y sus componentes, fue obtener los datos y características de los mismos.

Para hacer la interacción se utilizó el servicio web que provee el módulo de administración de la información y comunicaciones. Para esto se crearon clases en donde se almacenan las propiedades y los métodos específicos que se necesitan para crear el mundo virtual. Estas clases están en una biblioteca común que utilizarán todos los clientes que se conectan al servicio web. El nombre de esta biblioteca es "ClientLibrary" y dentro de ella se encuentran las clases que almacenan los datos del mundo y sus componentes.

Dentro de la biblioteca "ClientLibrary" se encuentra una clase creada especialmente para comunicarse con el servidor. Esta clase permite utilizar el servicio web como si fuera una clase más dentro de la biblioteca, sin importar que sus operaciones se realicen en el servidor. Esta clase se llama ServerConnection y se utiliza dentro de la aplicación tanto para obtener datos del mundo como para mandarlos.

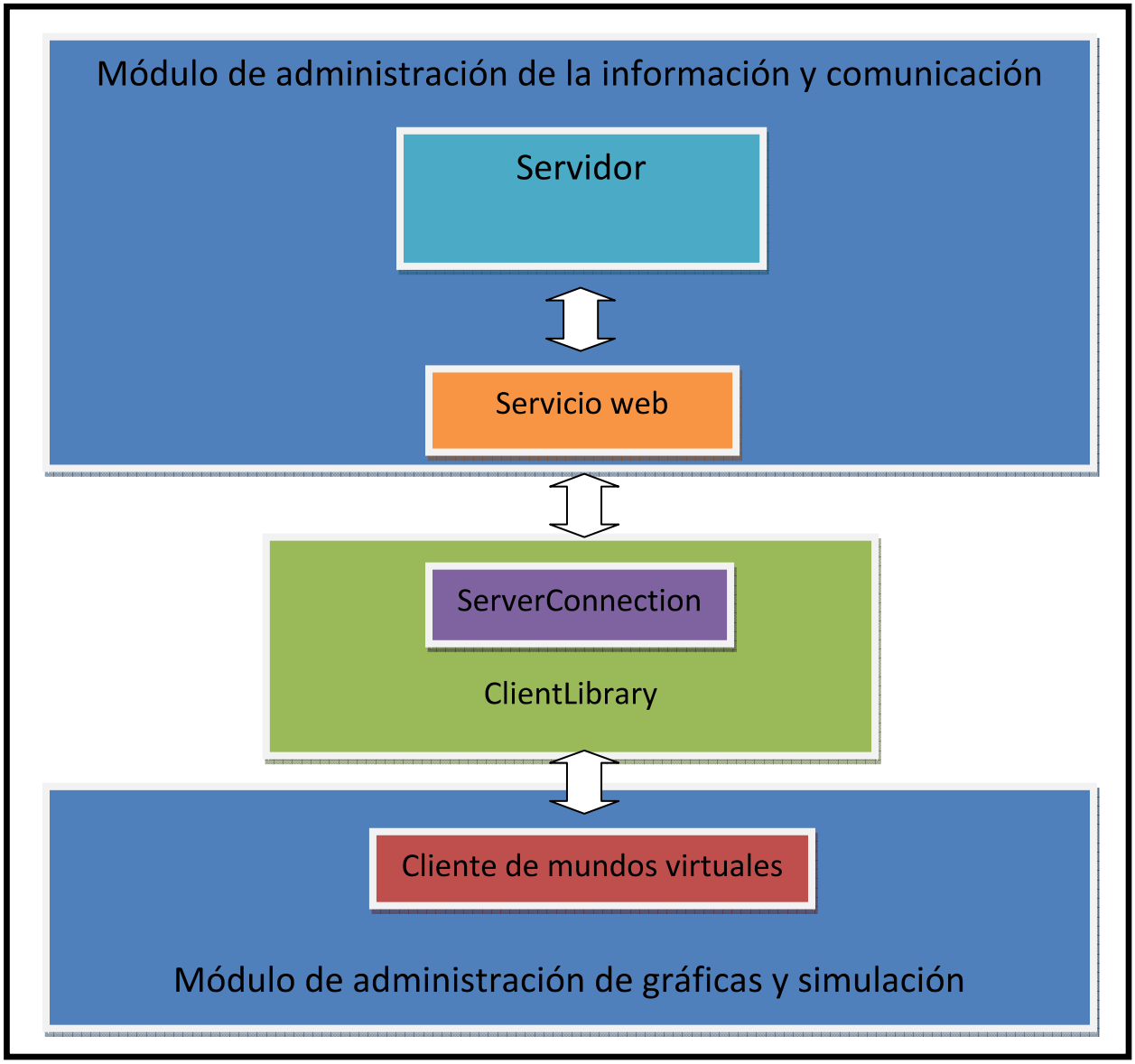
A continuación se muestra, en la Figura 6, un esquema de la interacción entre ambos módulos del proyecto utilizando la clase mencionada anteriormente.

7.2. Despliegue de gráficos en pantalla.

Cuando ya se puede acceder a los datos del mundo virtual, el siguiente paso es encontrar la manera de dibujar los objetos en pantalla. En el campo de las gráficas por computadora existen muchas formas de lograr esto. Una de ellas es mediante la biblioteca OpenGL. Sin embargo el uso de esta biblioteca, que aunque facilita el manejo de los dispositivos de salida y provee comandos avanzados para dibujar en pantalla, aún resulta ser bastante complicada si se requiere representar objetos complejos como personajes, y es aún más difícil el manejo de una gran cantidad de objetos a la vez. Por

esta razón se optó por buscar una herramienta que simplificara el desarrollo de la aplicación en la parte gráfica.

Figura 6 Interacción entre módulos

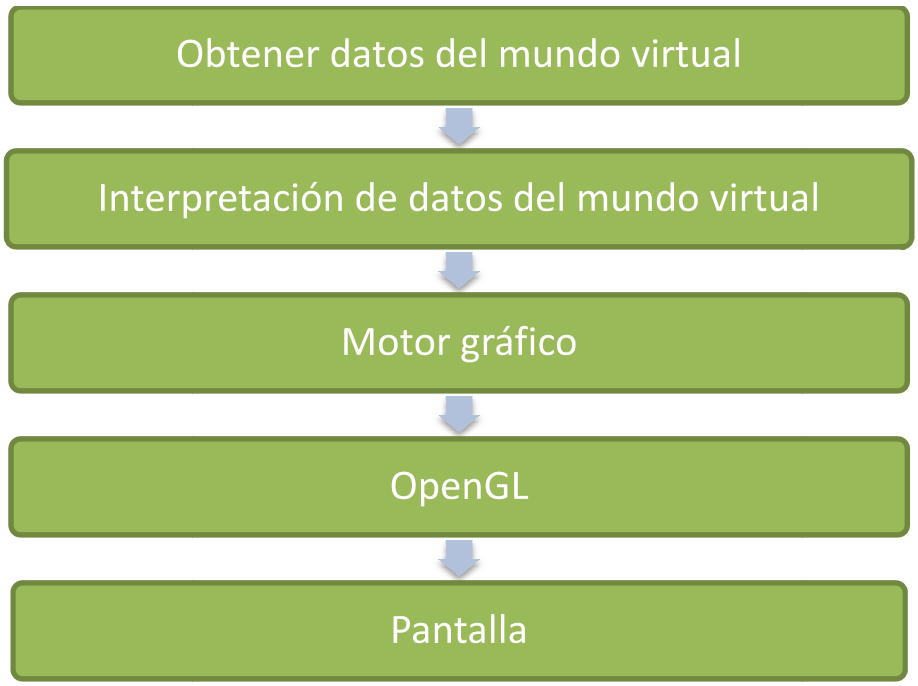


Un motor gráfico es una herramienta que provee facilidades y simplifica el trabajo de desarrollo de ambientes en tres dimensiones, su función principal es proveer funciones que realizan operaciones complicadas sin que el desarrollador tenga que entender los detalles y las complejidades que se efectúan al realizarlas. Debido a que uno de los requerimientos del módulo es que fuera desarrollado en Java, se buscó un

motor de gráficas que estuviera desarrollado en el mismo lenguaje de programación para poder tener una buena integración con este módulo y el resto del sistema.

A continuación se muestra un diagrama de cómo forma parte un motor gráfico en el desarrollo del proyecto.

Figura 7 Etapas de desarrollo de la representación gráfica



Uno de los objetivos principales de la aplicación es obtener los datos del mundo virtual y representarlos en pantalla.

Como se observa en la Figura 7, sin la ayuda de un motor gráfico, parte del trabajo de desarrollo sería transformar los datos interpretados del mundo virtual directamente a OpenGL y como se mencionó, esto presenta un nivel más de dificultad en la programación.

La mejor opción fue un motor de gráficas totalmente gratuito llamado JMonkeyEngine(JME2), que se encuentra en desarrollo en la actualidad por una comunidad de programadores. Este motor se encuentra actualmente estable en la versión 2 y se está desarrollando la versión 3.

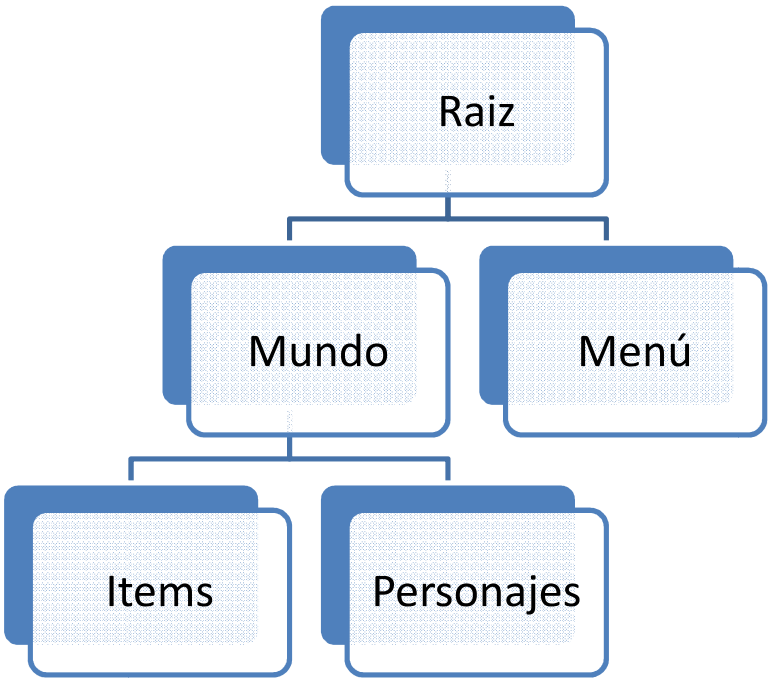
Este motor provee herramientas para el desarrollo de ambientes en tercera dimensión. Algunas de las que se utilizaron en este módulo son: generación de terrenos, importación de objetos 3D y el manejo de dispositivos de entrada.

La manera en que se manejan los objetos en este motor es muy interesante, utiliza una estructura de árbol para representar la jerarquía de los objetos y agruparlos. En la Figura 8 podemos observar la estructura mencionada.

El nodo raíz no representa ningún objeto, se utiliza como referencia al árbol, este nodo se encuentra activo en todo momento. Cada nodo puede representar un objeto en el mundo virtual o ser padre de otros nodos. En la Figura 8, el nodo raíz tiene dos hijos, el menú y el mundo. El mundo representa al mundo virtual y el menú es una ventana que da opciones al usuario. También se puede observar que el mundo tiene dos hijos, ítems y personajes.

Las transformaciones gráficas que se le aplican a un nodo padre se aplican automáticamente a los nodos hijos. Por lo que si, por ejemplo, movemos de posición el nodo de mundo también sus nodos hijos, ítems y personajes, se moverán con él. Este efecto se aplica también a cambio de tamaño y a rotaciones. Esto permite que el manejo de múltiples objetos sea más simple que si se tuviera que manejar cada objeto por separado.

Figura 8 Estructura de árbol de nodos del motor gráfico



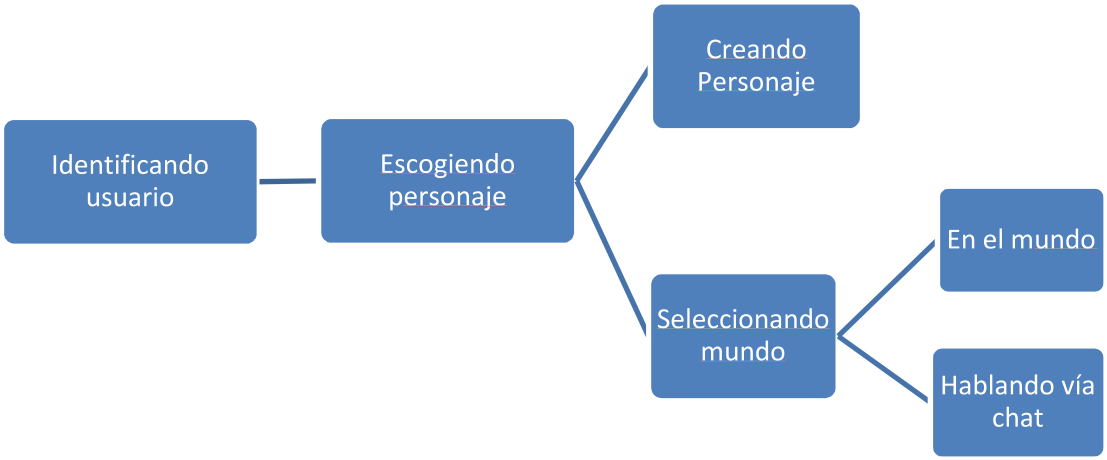
7.3. Manejo de estados de juego.

Un estado de juego es como una pequeña aplicación o sub módulo dentro de otra aplicación. Cada estado de juego tiene un fin específico por lo que su definición varía según su función. A continuación se muestran algunos de los estados de juego por los que pasa la aplicación, más adelante se describirá cada estado detalladamente.

En la Figura 9 se encuentra el diagrama de estados por los que pasa la aplicación al iniciar. Inicialmente la aplicación se encuentra en el estado de identificamdo usuario, cuando se ha identificado el usuario se pasa al estado de escogiendo personaje. En este estado el usuario puede elegir entre utilizar un personaje existente o crear uno nuevo. Si el usuario escoge la opción de crear un nuevo personaje entonces se pasará al estado de creando nuevo personaje. Al terminar la creación del nuevo personaje, se regresará al estado de escogiendo personaje. En este estado el usuario tiene nuevamente dos opciones, crear otro personaje o seleccionar uno existente. En este caso se puede

observar que aunque el usuario ya ha utilizado este estado antes, la decisión que tomó de crear un personaje no afecta las opciones que se le presentan ahora.

Figura 9 Diagrama de estados de juego



De regreso al diagrama, el usuario se encuentra en el estado de escogiendo personaje, si ahora el usuario elije un personaje creado se pasará al estado de seleccionando mundo en donde el usuario seleccionará el mundo que desea utilizar. Hasta el momento se ha pasado por condiciones en donde solo hay un estado activo en la aplicación al mismo tiempo.

Al tomar la decisión de qué mundo virtual utilizar, se pasará al estado del mundo virtual, también estará activo el estado del chat. En este momento existen dos estados activos que deben poder ser utilizados por el cliente y sus representaciones gráficas deben ser dibujadas en pantalla al mismo tiempo. La aplicación debe poder hacer funcionar los estados que estén activos en todo momento. También se debe tener una forma de agregar, remover y modificar los estados según se requiera.

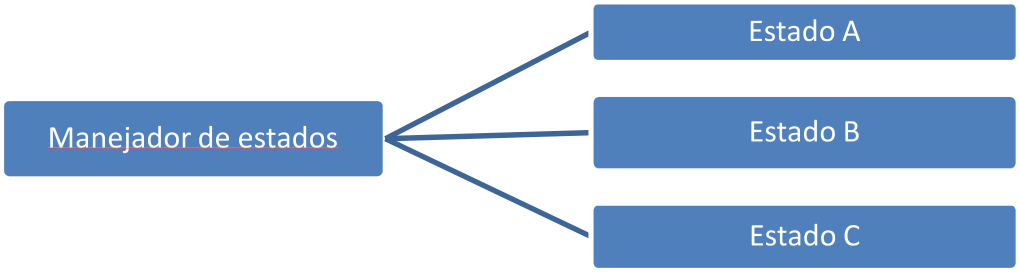
Para manejar los estados se creó una estructura en forma de lista que tiene los estados que están activos. Estos pueden cambiar según las acciones del usuario. Cada estado es diferente debido a que cada uno realiza diferente funciones y despliega diferente información en pantalla, sin embargo todos tienen las siguientes propiedades:

- Status: Indica el estado en que se encuentra (activo, remover, cargando, por agregar).
- Nombre: Indica el tipo del estado.(identificando usuario, seleccionando mundo, etc.)
- Prioridad de entrada: Indica cómo se comporta el manejo de dispositivos de entrada cuando este estado está activo.

Como puede haber múltiples estados activos en el mismo momento, el manejador de estados debe de controlar qué estado tiene derecho a utilizar la entrada. Esto se logra utilizando la propiedad de “Prioridad de entrada” que tiene cada estado. Si un estado tiene mayor prioridad que otro, el usuario solo podrá interactuar con ese estado hasta que el estado se remueva o se active un estado con mayor prioridad. Existen estados que tienen la misma prioridad de entrada, en este caso los estados comparten los dispositivos de entrada y el usuario puede interactuar con ambos.

En la Figura 10 se puede observar una representación del manejador de estados. El cuadro con la etiqueta manejador de estados representa la lista de estados que se encuentran operando dentro de la aplicación. Los otros cuadros indican los estados que están dentro de esta lista.

Figura 10 Manejador de estados

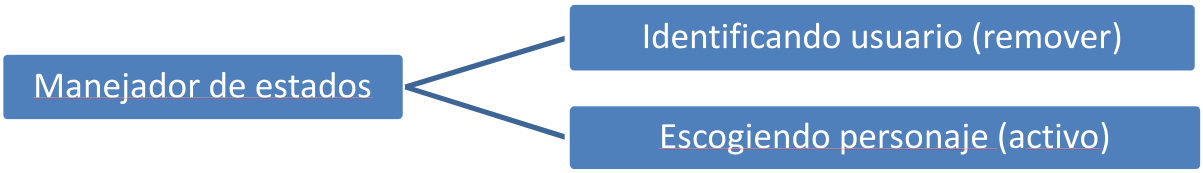


A continuación se muestra un ejemplo de cómo funciona el manejador de estados que se creó para la aplicación.

Figura 11 Ejemplo manejador de estados 1

En la Figura 11 se observa el inicio de la aplicación, encontramos el estado de identificando usuario, este estado se encuentra activo.

Figura 12 Ejemplo manejador de estados 2



En la Figura 12 se puede observar que al haber identificado el usuario, el estado de escogiendo personaje se agrega y el estado de identificando usuario debe ser eliminado.

Figura 13 Ejemplo manejador de estados 3

En la Figura 13 se observa que el estado de identificando usuario, ya ha sido borrado y sólo queda activo el estado de escogiendo personaje.

Lo mostrado en las figuras anteriores es sólo un ejemplo de lo que hace el manejador de estados, existen muchas combinaciones de estados que pueden estar activos al mismo tiempo.

El manejador de estados se crea al iniciar la aplicación, posteriormente se encuentra actualizando constantemente los estados dentro de la aplicación. En la figura 14 se muestra el ciclo por el que pasa el manejador de estados durante toda la ejecución del programa.

El manejador de estados es una de las partes principales de la aplicación, más adelante se mostrará cómo se integra con toda la aplicación para mantener actualizado en todo momento los objetos del mundo y sus componentes.

7.4. Estados de ventanas.

Para crear la interfaz gráfica en donde el usuario puede consultar información, ingresar texto, etc. se creó una interfaz en dos dimensiones. Esta interfaz de ventanas fue creada utilizando una biblioteca extra del motor de gráficas JME2 conocida como GBUI, Graphic Banana User Interface, que permite crear interfaces de ventanas y sus componentes comunes, como lo son listas, espacios de texto, etiquetas, etc.

Figura 14 Ciclo de manejador de estados

Figura 15 Estado de información del cliente

En la Figura 15 se muestra una imagen de captura de pantalla de la aplicación en funcionamiento. Se pueden observar tres estados activos al mismo tiempo. El estado que tiene terreno y cielo es el estado del mundo virtual. Sobre éste se encuentra, en la esquina inferior izquierda, el estado del chat. Por último sobre estos dos se encuentra el estado de información del personaje. Debido a que el último estado tiene mayor prioridad sobre la entrada que los otros dos, el usuario solo puede interactuar con este estado.

Ambos estados, el estado de chat y el de información del personaje fueron creados utilizando la biblioteca, GBUI, mencionada anteriormente. Estos estados se asemejan más a una aplicación tradicional de ventanas. Se pueden observar etiquetas de texto, botones, y listas de opciones. Se crearon de esta forma ya que por la naturaleza de la información que representan en pantalla, en su mayoría texto, no es necesario representarlo en forma tridimensional, además, a diferencia de los objetos 3D, no dependen de la dirección hacia donde este viendo el usuario dentro del mundo virtual, ya que siempre se representan de la misma manera.

Utilizando estas herramientas se crearon los siguientes estados que pueden ser usados dentro de la aplicación:

- **AlertDialogState:** Sirve para desplegar un mensaje al usuario.
- **CharacterState:** Es el estado que contiene la información del personaje.
- **ChatState** Es el estado del chat que está activo cuando se está en un mundo virtual.
- **ConversationDialogState:** Se utiliza para desplegar una conversación entre un ítem y el personaje principal.
- **InventoryState:** Se utiliza para mostrar los ítems que posee el personaje principal.
- **LoadingState:** Se muestra mientras está cargando otro estado.

- **LoginState:** Es el estado inicial en donde el usuario ingresa su nombre de usuario y su contraseña.
- **MenuState:** Se muestra en el mundo virtual, contiene las opciones de ver información del personaje principal, ver inventario, ver misiones, regresar al mundo o salir del mundo actual.
- **NewCharacterState:** Es el estado en donde se puede crear un nuevo personaje.
- **OtherCharacterState:** Se muestra la información del personaje de otro usuario.
- **PictureDialogState:** En este estado se muestran fotografías, se espera que sean fotografías con material que sirva para terminar misiones.
- **QuestState:** Es el estado en donde el personaje puede ver el historial de sus misiones.
- **SelectCharacterState:** Es el estado en donde el usuario selecciona al personaje que utilizará en el mundo virtual.
- **SelectWorldState:** Es el estado en donde se selecciona el mundo virtual al cual se desea entrar con el personaje.
- **StartQuest:** En este estado se muestra una misión en donde el personaje puede optar por aceptarla o rechazarla.
- **TestDialogState:** En este estado se muestra una pregunta y se dan varias opciones para que el usuario conteste.

7.4.1. Ciclo de vida. El ciclo de vida individual de estos depende de la función de cada uno, sin embargo el ciclo general es:

- Crear la ventana
- Crear componentes dentro de la ventana
- Crear manejador de entrada

Posteriormente ingresa a un ciclo de ejecución hasta ser detenido. El ciclo se muestra en la figura 16

Figura 16 Ciclo de vida de estados de ventanas

7.5. Estado de mundo virtual.

Este es el estado más importante de la aplicación, también es el más complejo. Se compone de dos grupos de componentes que debe representar. Los primeros son los componentes gráficos y los segundos son los componentes de simulación. A continuación se muestra como se desarrolló este estado.

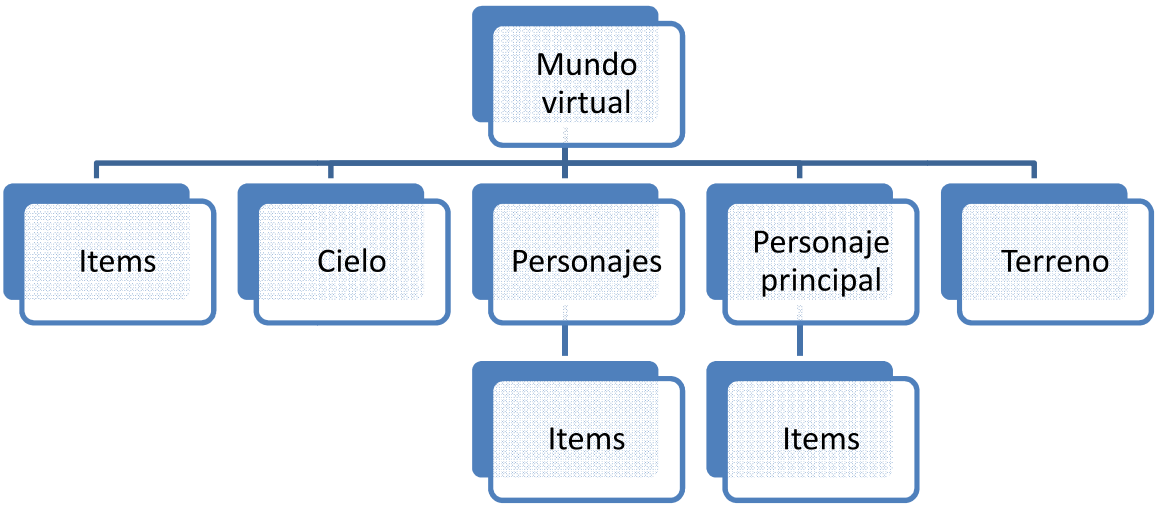
7.5.1. Componentes gráficos

En la figura 17 se puede observar cuales son así como su orden dentro del mundo.

7.5.1.1. Terreno. El terreno es el objeto más grande que hay dentro del mundo, los define tanto su forma como las texturas y colores que tiene en cada posición.

En la Figura 18 se puede observar una imagen de captura de pantalla de un mundo virtual de prueba creado con esta aplicación. En esta imagen se puede observar el terreno. Hay distintos niveles, desde el agua, que es lo más profundo, hasta la punta de las montañas, que se observan en la parte más lejana, que son la parte más alta.

Figura 17 Componentes gráficos del mundo virtual



Para crear el terreno se utilizó una herramienta que provee el motor de gráficas. Esta herramienta permite generar terreno a partir de una fotografía. La idea principal es que cada pixel de la fotografía representa la profundidad del terreno en esa área, en donde negro es la profundidad más baja y blanco es la más alta, los tonos de gris entre estos dos colores representan alturas entre lo más alto y lo más bajo dependiendo de su tonalidad. En la Figura 19 se muestra un mapa del mundo virtual de prueba. En la Figura 20 se muestra la imagen que se utilizo para generar el terreno de este mundo.

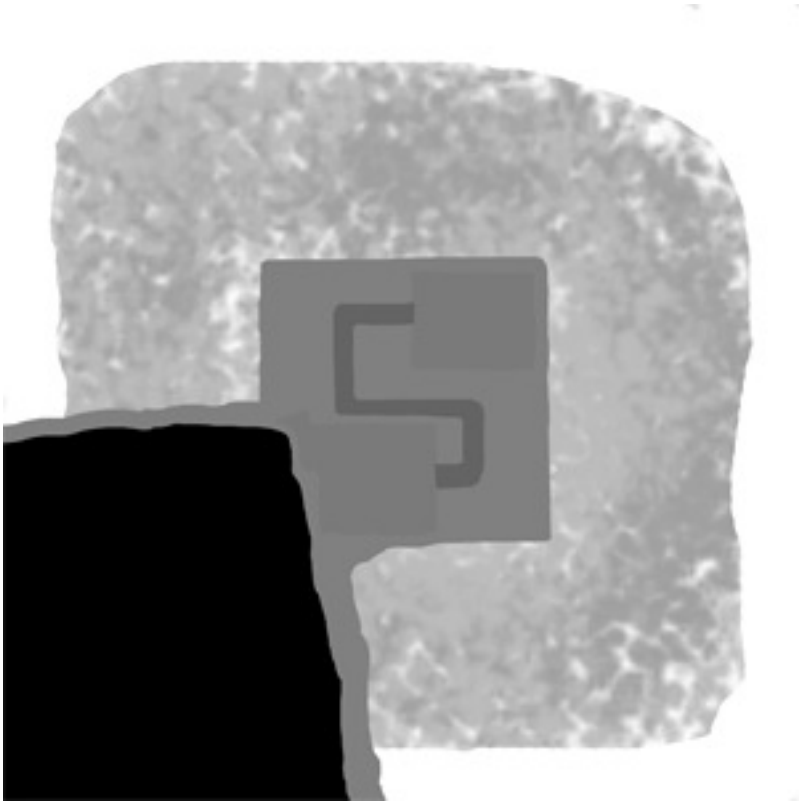
En el centro de la Figura 20 se puede observar que una imagen parecida al mapa, esto es porque el área que representa el mapa se encuentra en el centro del terreno.

Figura 18 Mundo virtual

El área negra que se observa en la imagen del terreno representa el área de menos altura. Esta es el área que se mira como agua en la Figura 18. En la imagen del terreno, en los alrededores del centro se puede ver el patrón que representan las montañas.

Figura 19 Mapa de mundo virtual

Figura 20 Terreno de mundo virtual



Utilizando la herramienta de generación de terreno mediante los tonos de grises se creó la forma del terreno. El siguiente paso es ponerle el color y textura.

Para esto se utiliza otra herramienta que provee el motor de gráficas, un método de aplicar texturas diferentes al terreno generado dependiendo de la altura en la que se encuentre.

En el mundo de prueba se aplicó la textura de agua a la altura más baja, textura de grama a la textura de altura media y textura de montaña a la altura más alta.

7.5.1.2. Cielo. El cielo es un cubo que se encuentra alrededor del usuario. El cubo se mueve procurando que el usuario siempre esté en el centro del mismo. Con esto se logra dar el efecto de que el cielo se encuentra lejano y es inalcanzable.

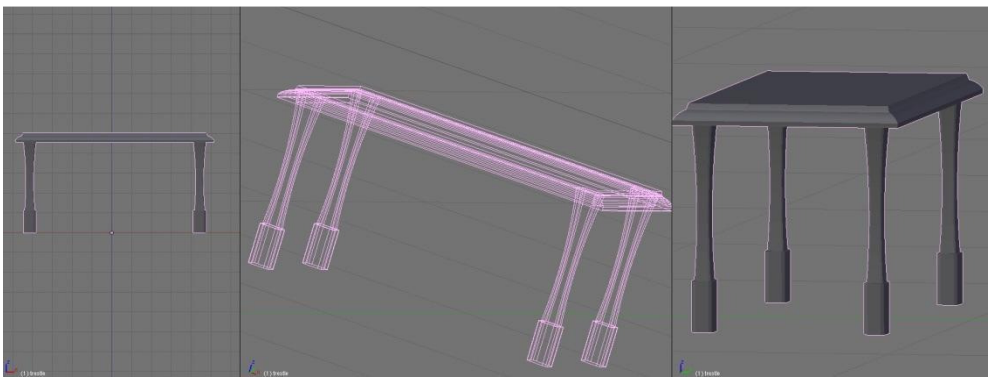
Se le coloca una textura de cielo en la parte de adentro al cubo. En la Figura 21 se puede observar un ejemplo de cómo son las texturas que se le aplican al cubo del cielo. Esta imagen debe tener el tamaño y los cortes adecuados para que la representación del cielo sea efectiva.

Figura 21 Textura del cielo

7.5.1.3. Ítems. Los ítems son las representaciones de los objetos y de los personajes que no son controlados por usuarios.

Existe un gran número de formatos en los cuales se guardan objetos 3D. La mayoría consiste de una lista de vértices que forman las figuras geométricas que componen el objeto en tres dimensiones y los datos de las texturas y colores que este tiene.

Figura 22 Ítem - Blender



En esta aplicación se utilizan dos formatos:

- OBJ: Este es un formato que se compone de dos archivos:
  - OBJ: Contiene la información de los vértices que componen el objeto.
  - MTL: Contiene la información de los materiales, como texturas y colores que se debe aplicar al objeto y en qué partes.
- JME-XML: Este formato está definido por los creadores del motor de gráficas JME2 y utiliza XML para definir los vértices y las texturas en un mismo archivo. También puede contener animaciones en el caso que sea un objeto animado.

Para utilizar estos formatos se hizo uso de herramientas que provee el motor de gráficas, tanto para leer los formatos OBJ como los XML. A ésta herramienta se le proporciona la dirección del archivo con el modelo 3D y esta crea un nodo que puede ser utilizado con facilidad y posicionado dentro del mundo.

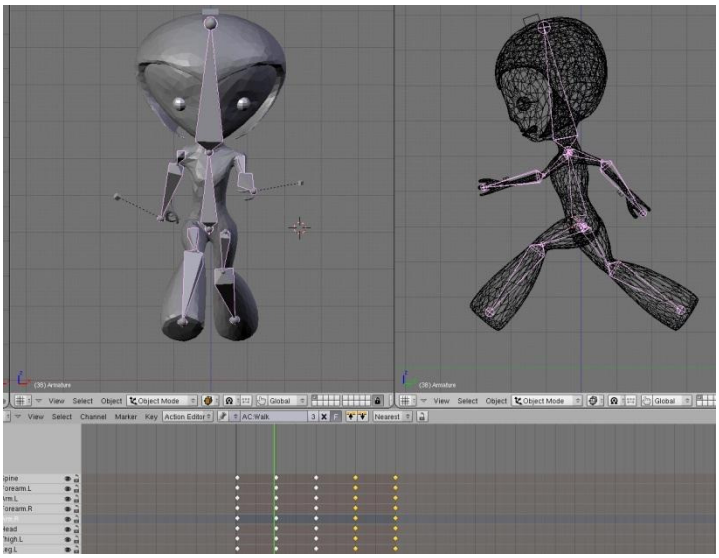
Para poder editar los modelos y las animaciones se hizo uso de una aplicación de modelación 3D. El nombre de esta aplicación es Blender, es una aplicación gratuita. En la Figura 22 se puede observar parte de la interfaz de esta aplicación así como un ítem que se utilizó en un mundo de pruebas.

Esta aplicación permite grabar los modelos en muchos formatos y uno de ellos es el formato OBJ. Mediante una secuencia de comandos proveídos por los creadores del motor de gráficas, Blender es capaz de grabar los modelos en archivos de formato XML. Archivos que pueden ser utilizados con las herramientas del motor de gráficas.

7.5.1.4. Personajes. Representan a los usuarios dentro del mundo virtual. A diferencia de los ítems, para los personajes únicamente se utiliza el formato JME-XML. Esto se debe a que todos los personajes tienen animaciones, como las de caminar y saltar.

Para crear estas animaciones es necesario colocar un esqueleto a los objetos que representan a los personajes. Después se le graban secuencias de movimientos utilizando una aplicación de modelado y animación 3D. En la Figura 23 se puede observar un personaje creado para un mundo de pruebas. En la figura, se puede observar que el personaje tiene un esqueleto simulado y está realizando la animación de caminar. Se utilizó la herramienta Blender para hacer estas transformaciones.

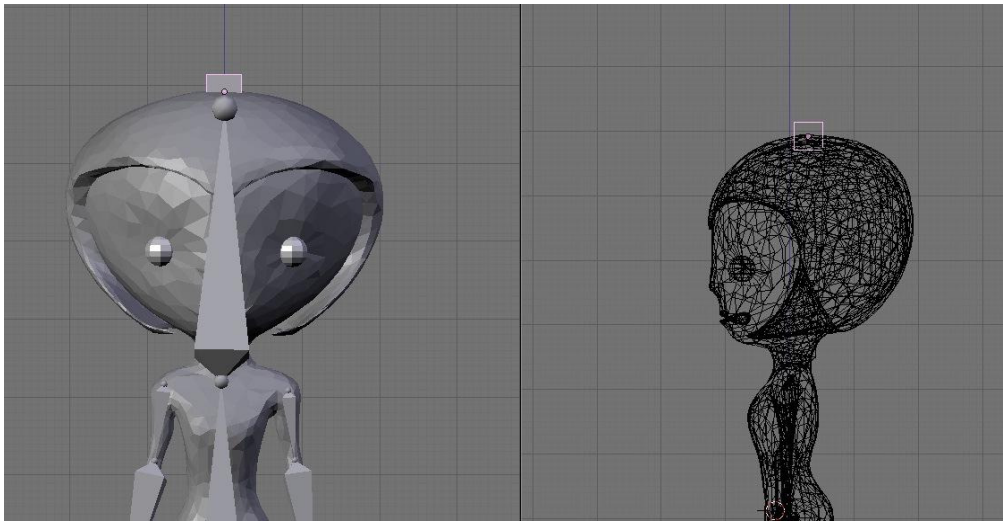
Figura 23 Personaje - Animaciones - Blender



Además de animaciones, los personajes también tienen la capacidad de equipar otros ítems en puntos específicos. En la Figura 24 se muestra un punto de montaje, este se encuentra representado por un cubo sobre la cabeza del personaje.

Al introducir al personaje al mundo virtual este cubo se remueve y se coloca, si es necesario, el ítem que el usuario desee. Como por ejemplo un sombrero. También hay puntos de montaje en ambas manos en donde se pueden colocar ítems.

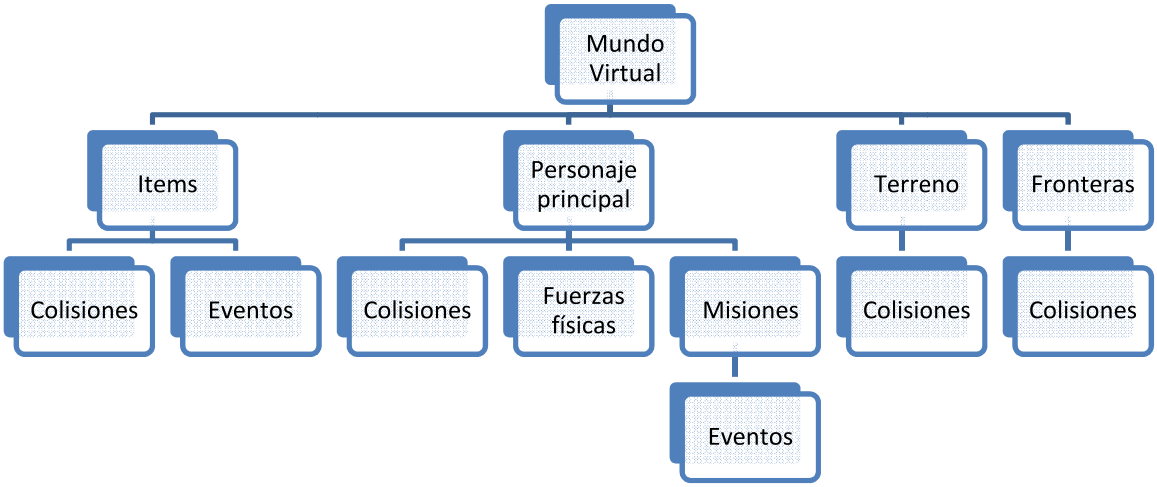
Figura 24 Personaje - Puntos de montaje - Blender



El personaje principal es el que el usuario esté utilizando en la aplicación, a diferencia de los otros personajes, el usuario puede, mediante los dispositivos de entrada, realizar funciones específicas con este personaje, como explorar el mundo, saltar y caminar.

7.5.2. Componentes de simulación. En la Figura 25 se pueden observar, así como su orden dentro del mundo.

Figura 25 Componentes de simulación del mundo virtual



7.5.2.1. Fuerzas físicas. Se aplican fuerzas físicas solamente al personaje principal de cada cliente. Esto significa que la aplicación solo es encargada de aplicar las fuerzas al personaje que representa al usuario de esta aplicación. De esta forma se mejora el desempeño de la aplicación y se prevé de cualquier problema que podría ocurrir por algún retraso en la sincronización con los otros clientes.

Las fuerzas que se aplican causan una aceleración en el personaje, la cual genera una velocidad. Estos cálculos se realizan usando conceptos simples de cinemática, en específico de movimiento rectilíneo uniformemente variado.

7.5.2.2. Eventos. Son una serie de condiciones que al cumplirse ejecutan una cadena de acciones.

Los ítems del mundo virtual tienen eventos, al cumplirse estos eventos se ejecutan acciones que cambian los datos del mundo o del personaje. Estos eventos pueden ser: encontrarse a una distancia de un ítem especifico, presionar sobre un ítem específico, etc. Cada mundo virtual puede tener los eventos que desee para llevar a cabo el objetivo del mundo.

Los eventos se revisan constantemente para que cuando estos ocurran, se ejecuten sus acciones. Las acciones pueden ser desde cambiar o activar un estado hasta cambiar una propiedad del personaje o del mundo. Las acciones también varían según el objetivo del mundo virtual.

7.5.2.3. Colisiones. Un objeto colisiona con otro cuando sus modelos tridimensionales se sobreponen en el espacio, sin embargo hay modelos que son muy complejos. En este caso, calcular cuando se sobreponen con otro puede generar mucha carga para la aplicación y causar que se ejecute de manera incomoda para el usuario. Por esta razón se utilizan cajas de colisión. Una caja de colisión es una caja que rodea al

objeto, de esta forma en vez de realizar los cálculos con la gran cantidad de lados que posee un objeto complejo, se realizan utilizando las cajas que los rodean.

En la aplicación desarrollada, las colisiones se revisan únicamente entre el personaje principal del cliente y los ítems y las fronteras del mundo. De esta forma cada cliente se encarga de revisar y asegurarse que su personaje principal no se sobreponga con otros objetos. Esto mejora grandemente el desempeño de la aplicación ya que solo se debe hacer cálculos de colisión con un solo personaje.

Los cálculos de las colisiones se realizaron utilizando las herramientas que el motor de graficas provee para revisar colisiones.

7.5.2.4. Fronteras. Delimitan el área que el personaje puede explorar dentro del mundo. Para lograr esto se crearon paredes invisibles que son tratadas como si fueran ítems del mundo que los personajes no pueden atravesar.

7.5.2.5. Misiones. Las misiones crean eventos adicionales que el cliente debe de revisar constantemente para verificar cuando se cumplen. El cliente revisa los eventos únicamente del personaje principal. Al cumplirse, el evento funciona de la misma manera que los eventos que tienen los ítems.

7.5.3. Cámara. La cámara sigue al personaje, el movimiento del personaje toma en cuenta la dirección de la cámara.

Para realizar esto se utilizó una herramienta de creación de cámara que provee el motor gráfico. Mediante ésta se puede configurar: la distancia máxima a la que que puede estar la cámara del personaje, la distancia mínima a la que puede estar la cámara del personaje y cómo debe reaccionar la cámara al presionar los botones del ratón.

7.5.4. Luces. Se implementaron dos luces direccionales que apuntan en direcciones opuestas, una tiene más fuerza que la otra para crear efectos de sombreado en los objetos.

Para esto se utilizaron las herramientas de iluminación que el motor gráfico provee, que permiten especificar la dirección de la luz y su intensidad.

7.5.5. Sincronización de datos. Se deben de mantener sincronizados los datos del mundo, entre estos las posiciones y sus propiedades de los personajes e ítems. Para esto se utilizaron tres hilos que corren simultáneamente dentro del estado del mundo.

El primero es el hilo que mantiene una lista de posiciones por la cual ha pasado el personaje principal. Se utiliza una lista para no sincronizar la posición del personaje cada vez que este se mueve, sino enviar un conjunto de posiciones de una sola vez y así no sobrecargar el servidor. Con la lista de posiciones se puede reproducir el movimiento del personaje en otros clientes. Esta lista se borra cada vez que es enviada mediante el servicio web, de este modo puede ser utilizada nuevamente.

El segundo es el hilo que sincroniza las posiciones, tanto del personaje principal como de los personajes de otros usuarios que se encuentran en el mundo. Éste envía la lista de posiciones que se crea en el hilo mencionado anteriormente y recibe la lista de posiciones de los demás personajes. El tiempo que transcurre entre cada sincronización con el cliente es configurable.

El último hilo se utiliza para sincronizar el resto de datos del mundo y de los personajes que no han sido actualizados por los hilos anteriores. El tiempo que transcurre entre cada sincronización es configurable.

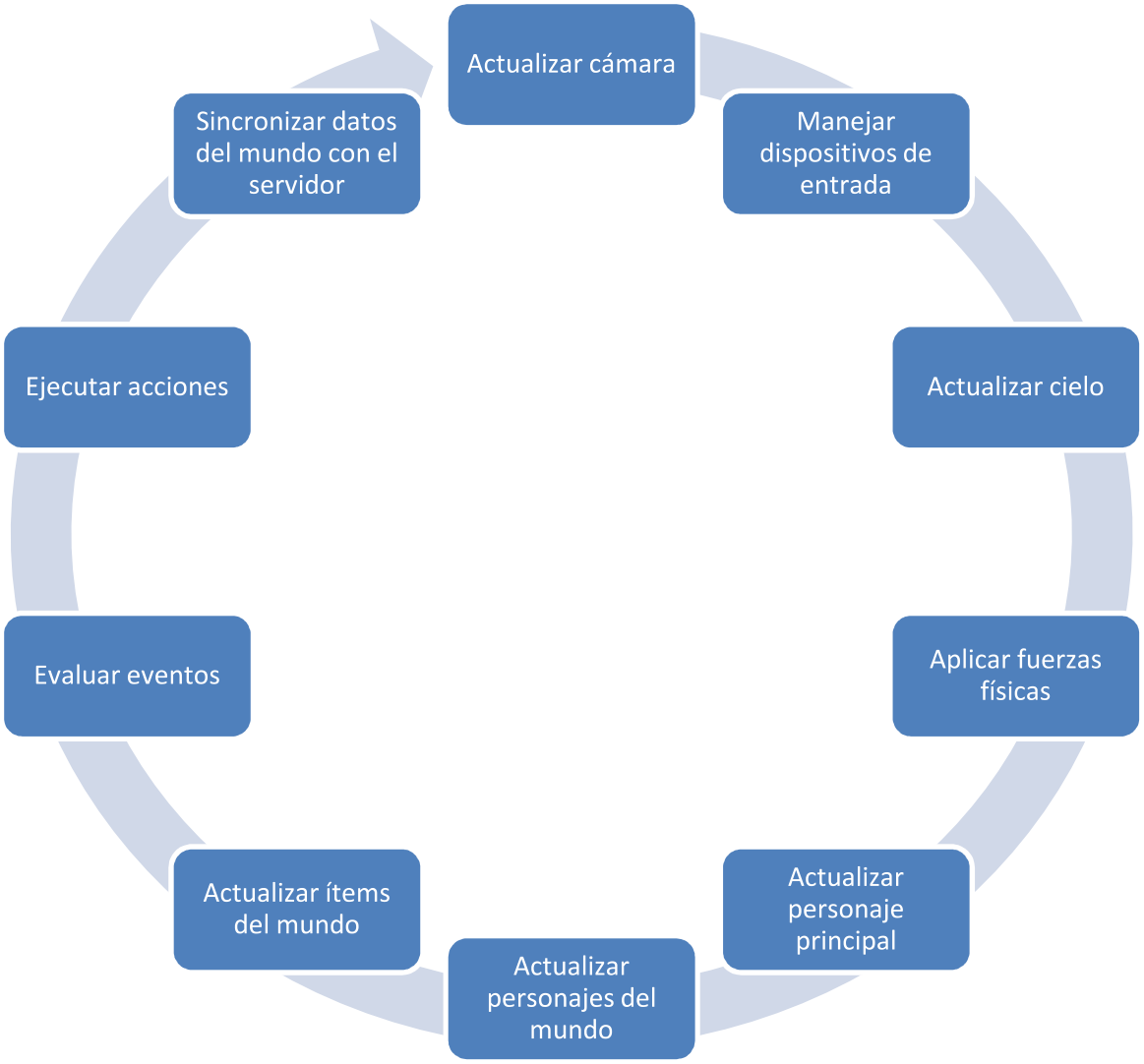
7.5.6. Ciclo de vida: Al iniciar el estado del mundo virtual, se crean sus componentes en el siguiente orden:

- Cámara
- Terreno
- Luces

- Ítems del mundo
- Regiones
- Personajes del mundo
- Personaje principal
- Cielo

Al terminar de crear estos componentes el estado ingresa un ciclo de ejecución hasta que es detenido. El ciclo se muestra en la Figura 26

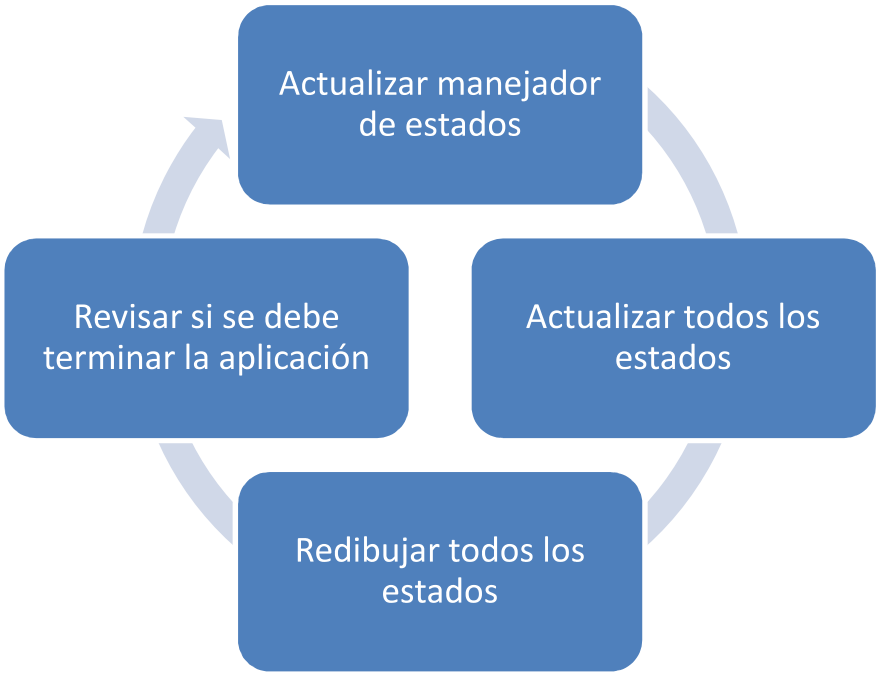
Figura 26 Ciclo de vida del mundo virtual



7.6. Ciclo de vida de la aplicación.

Al iniciar se muestra una ventana con la configuración que se desea utilizar la aplicación. Esto es la resolución de la aplicación, el modo de pantalla (modo pantalla completa o modo ventana). Después de que el usuario escogió la configuración, se crea la aplicación con estos datos. Al iniciar se crea el manejador de datos y se agrega el estado de identificar usuario. Finalmente la aplicación entra a un ciclo de ejecución hasta que sea detenida. El ciclo se muestra en la Figura 27.

Figura 27 Ciclo de vida de la aplicación



Aquí se concluye el desarrollo de la aplicación. La aplicación se puede utilizar en una computadora que tenga instalada la máquina virtual de Java y una conexión al servicio web, ya sea por Internet o por medio de una red local.

8. RESULTADOS

El resultado de este trabajo es la creación de una aplicación que cumple con los requerimientos definidos del módulo de gráficas y simulación.

Se creó también un mundo de prueba en donde se utilizó la aplicación para representar un mundo virtual específico y se hizo una demostración de su funcionamiento en público.

Se creó este documento que sirve como una guía del desarrollo de este módulo. Los archivos utilizados y el código fuente desarrollado en esta aplicación se encuentran en los apéndices de este documento.

9. DISCUSIÓN

El uso de un motor gráfico facilitó el desarrollo de la aplicación para representar de forma gráfica los mundos virtuales. Se utilizó el motor gráfico JmonkeyEngine debido a que, además de estar desarrollado en Java, proveía las herramientas necesarias para el desarrollo de este módulo. Otra ventaja es que su uso no es tan complicado y no necesita un nivel de conocimientos excesivamente alto, tanto de gráficas por computadora como de matemáticas, para poder utilizar la mayoría de sus herramientas.

Sin embargo, en algunos casos las opciones que proporcionaba el motor gráfico no eran tan simples como se requería, como lo fue en el caso del manejo de los ángulos de rotación de los objetos. Estos estaban representados por una estructura conocida como quaternions, los cuales no son lo suficientemente simples para trabajar con ellos y se deben de realizar transformaciones extra para utilizarlos.

Otra desventaja de haber utilizado el motor gráfico es que, en algunas ocasiones, este restringía las opciones que se podían utilizar en el desarrollo de la aplicación. Como por ejemplo en la importación de objetos animados, la herramienta que provee el motor gráfico utiliza un formato de archivo XML y esto no podía ser modificado sin necesidad de hacer grandes cambios al motor.

En el uso de los objetos animados que estaban dentro de un XML, no se podía duplicar este objeto para poderse reutilizar sin tener que leer el archivo otra vez. Esto ocasionaba que un personaje o un ítem que ya había sido cargado tuvieran que volverse a cargar aunque fuera exactamente el mismo. El problema era que al duplicarse éste no

duplicaba sus texturas ni el controlador de la animación, por lo que la copia se mostraba sin texturas y con la misma animación que el original.

En algunas ocasiones la aplicación se encontraba con demasiada carga y se quedaba sin actualizarse por un momento la imagen. Al terminar de procesar esa carga, la aplicación trata de compensar por los cuadros que no se actualizaron lo cual crea un efecto de movimiento más rápido de lo normal, tanto en las animaciones como a la velocidad en que se movían los personajes.

Otro problema al utilizar los objetos animados es que muchas veces se quería utilizar el mismo modelo pero con texturas diferentes, el objeto animado solo permitía el uso de una textura y debido a que la dirección se encontraba dentro del XML que no podía ser editado ya que se utilizaba por múltiples modelos, se debía de crear una copia del archivo XML para poder cambiar de textura al objeto.

La biblioteca “ClientLibrary” que se utilizó para la interacción con el módulo de administración de la información y comunicaciones fue de gran ayuda, ya que con esto se tuvo acceso a los datos del servidor de la misma forma en que se accedían los datos locales.

10.CONCLUSIONES

La aplicación representa de forma gráfica el mundo utilizando un motor de gráficas y creando representaciones visuales para cada uno de los objetos, condiciones y personajes que lo componen.

Se permite la interacción entre los usuarios y el mundo virtual mediante el mouse y el teclado. La aplicación responde a la interacción del usuario cambiando de estados o cambiando los objetos dentro del mundo virtual.

Se accede a los datos del mundo virtual mediante el módulo de administración de la información y comunicaciones utilizando la biblioteca de clases compartida por todos los clientes “ClientLibrary” y las clases de datos que contiene.

Se controla el flujo de la aplicación mediante el manejador de estados, con éste se tiene claro el orden en que los estados deben aparecer y a qué estados se puede llegar cuando se está en determinado estado.

Se manejan los dispositivos de entrada y salida en cada estado, también se maneja la prioridad de entrada que tiene cada estado en el momento que se encuentran activos múltiples estados.

Se provee una forma de utilizar la aplicación en cualquier plataforma mediante el uso de un motor gráfico desarrollado en Java y que utiliza OpenGL, esto hace que la aplicación pueda ser ejecutada en un gran número de sistemas de software y de hardware.

Se utilizaron herramientas de desarrollo gratuitas, ya que se utilizo Java, OpenGL, el IDE Eclipse para la programación, servicios web también creados en Java y sobre la plataforma Linux. Se utilizo el programa Blender para la creación de los objetos 3D, los puntos de montaje y las animaciones de los personajes y los ítems animados.

11.RECOMENDACIONES

La cantidad de modelos y la complejidad de estos influye en el desempeño de la aplicación, tanto en el desempeño gráfico como en la cantidad de información que debe ser obtenida del servidor. Se recomienda que se utilice el menor número de objetos y que tengan la menor complejidad posible para que la aplicación funcione de una manera que sea cómoda para el usuario.

Se recomienda que el tamaño de las imágenes que utilicen los modelos sea potencia de dos, ya que algunos dispositivos no soportan otros tamaños y deben de cambiar el tamaño de las imágenes cuando son utilizadas y esto impacta en el desempeño de la aplicación.

Si se desea tener muchos ítems animados diferentes se recomienda que las diferencias se hagan utilizando ítems equipados a esos ítems y no utilizando modelos diferentes para cada uno. Utilizar modelos diferentes hará que el tamaño de la aplicación sea mayor y este será más difícil de transportar.

Se recomienda que se utilice, cuando sea posible, un mismo ítem estático en diferentes posiciones en vez de utilizar dos ítems estáticos, de esta forma solo se debe leer el modelo una sola vez del archivo y la siguiente se hará una copia en memoria. Esto no funciona de la misma manera con los ítems animados.

Se recomienda que no se creen mundos virtuales de gran tamaño, esto puede hacer que el desempeño de la aplicación sea bajo. Si es posible dividir el mundo en pequeños mundos y que el efecto de la simulación sea el mismo se debe de hacer.

Debido a que el sistema inicialmente estaba enfocado a la creación de mundos virtuales para uso en educación, existen varias funcionalidades que pueden ser de gran utilidad para enseñanza. Entre estas están los estados de prueba, los estados que dan información al usuario, entre otros, por lo que se recomienda su uso en aplicaciones educativas

12.BIBLIOGRAFÍA Y REFERENCIAS

Bartle, Richard. *Designing Virtual Worlds.* New Riders Publishing, 2003.

Cerami, Ethan. *Web Services Essentials.* O'Reilly, 2002.

Cooper, James W. *A Beginners Intrdouction to Java 2.* IBM Thomas J Watson Research Center, 2000.

Gee, James Paul. «Why are Video Games Good for Learning?» *Academic Advanced Distributed Learning Co-Lab.* http://www.academiccolab.org/resources/documents/MacArthur.pdf (último acceso: 16 de Septiembre de 2010).

Gollub, Rachel. "Second Life and Education." *Crossroads* (ACM) 14, no. 1 (2007).

Hearn, Donald, and M. Pauline Baker. *Computer Graphics C version.* Prentice Hall, 1996.

Kemp, Jeremy, and Daniel Livingstone. "Putting a Second Life Metaverse Skin on Learning Management Systems." *Second Life Education Workshop 2006.* San Francisco, 2006.

Mason, Hilary. "Experiential Education in Second Life." *Second Life Education Workshop 2007.* 2007.

Mayo, Merrilea J. "Video Games: A Route to Large-Scale STEM Education?" *Science Magazine*, no. 323 (Enero 2009).

Mitchel, Don. *From MUD's to Virtual Worlds.* 23 de Marzo de 1995. http://www.mentallandscape.com/Papers_95vworlds.htm (último acceso: 12 de Agosto de 2010).

Neider, Jackie, and Davis Tom. *The official Guide to Learning OpenGL Version 1.1.* Addinson-Wesley Publishing Company, 1997.

Peachey, Anna, Julia Gillen, Livingstone Daniel, and Sarah Smith-Robins, . *Researching Learning in Virtual Worlds.* Springer London, 2010.

Sherley, Peter. *Fundamentals of Computer graphics.* A.K. Peters, LTD, 2005.

Zerbst, Stefan, y Oliver Duvel. *3d Game Engine Programming.* Boston. MA: Thomson, 2004.

APÉNDICE I. GLOSARIO

Compilador: Es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación. Usualmente el segundo lenguaje es lenguaje de máquina. Este proceso de traducción se conoce como compilación.

Eclipse: Es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse).

Emulador: Un software que permite ejecutar programas de computadora en una plataforma (arquitectura hardware o sistema operativo) diferente de aquella para la cual fueron escritos originalmente. A diferencia de un simulador, que sólo trata de reproducir el comportamiento del programa, un emulador trata de modelar de forma precisa el dispositivo que se está emulando.

Estructura de árbol: Es una forma de representar la jerarquía de una estructura de forma gráfica. Se llama una estructura de árbol ya que la representación clásica se parece a un árbol. Tiene una raíz única. Esta raíz puede tener un número indefinido de hijos, o ramas. A las ramas que no tienen ningún hijo se le conoce como hojas.

Frame: Se denomina frame en inglés, a un fotograma o cuadro, una imagen particular dentro de una sucesión de imágenes que componen una animación. La continua sucesión de estos fotogramas producen a la vista la sensación de movimiento, fenómeno dado por las pequeñas diferencias que hay entre cada uno de ellos.

Interfaz gráfica: Conocida también como GUI (del inglés *graphical user interface*) es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador.

Java: Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Lenguaje de programación: Es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado de un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación.

Módulo: Es un componente auto controlado de un sistema, dicho componente posee una interfaz bien definida hacia otros componentes; algo es modular si está construido de manera tal que se facilite su ensamblaje, acomodamiento flexible y reparación de sus componentes.

Motor de gráficas: es un término que hace referencia a una serie de rutinas de programación que permiten el diseño, la creación y la representación de un mundo virtual.

Multiplataforma: Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas.

Mundo virtual: Se trata de la simulación de mundos o entornos, denominados virtuales, en los que el hombre interacciona con la máquina en entornos artificiales semejantes a la vida real.

OpenGL: (Open Graphics Library) es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. La interfaz consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples, tales como puntos, líneas y triángulos.

Render: *Renderizado* (*render* en inglés) es un término usado en jerga informática para referirse al proceso de generar una imagen desde un modelo. Este término técnico es utilizado por los animadores o productores audiovisuales y en programas de diseño en 3D.

En términos de visualizaciones en una computadora, más específicamente en 3D, la renderización es un proceso de cálculo complejo desarrollado por un ordenador destinado a generar una imagen 2D a partir de una escena 3D. En el proceso de renderización la computadora *interpreta* la escena en tres dimensiones y la plasma en una imagen bidimensional.

Servicio web: Es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos.

Sistema: Es un conjunto de elementos dinámicamente relacionados formando una actividad para alcanzar un objetivo

Wireframe: Algoritmo de renderización del que resulta una imagen semitransparente, de la cual sólo se dibujan las aristas de la malla que constituye al objeto. De ahí su nombre.

Casi nunca se emplea en la representación final de una imagen, pero sí en su edición, debido a la escasa potencia de cálculo necesaria (comparada con otros métodos).

Para conseguir una imagen en wireframe sólo tenemos que tener en cuenta las posiciones de los puntos en el espacio tridimensional y las uniones entre ellos para formar los polígonos.

XML: Siglas en inglés de *Extensible Markup Language* (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

APÉNDICE II: ESTRUCTURA DE ARCHIVOS DE LA APLICACIÓN

A continuación se muestra la estructura de archivos de la aplicación que se creó en este módulo.

- **Core:** Es en donde se encuentran las clases principales de la aplicación.
  - **GameCore.java**: En esta clase se encuentran los datos principales de la aplicación, se pueden acceder desde cualquier otra clase de la aplicación.
    - **Display:** Es el sistema de render que se utiliza para dibujar los objetos en pantalla.
    - **UserCredentialTR:** Es la información necesaria del usuario que está utilizando este cliente, para ser autenticado por el servicio web. Al inicio contiene datos del usuario "visitante" para poder autenticarse con el servicio web.
    - **Camera:** Es la cámara actual que se está utilizando, solo puede haber una activa en cada momento.
    - **UserDT:** es la información sobre el usuario y sus datos.
    - **CharacterDT:** es la información del personaje del usuario que se está utilizando actualmente en este cliente.

- **GameSettings:** Se encuentran los datos del cliente como el ancho de la ventana, alto de la ventana y si esta en modo de pantalla completa o no.
- **Finished:** Indica si el cliente debe de cerrarse.
- **UpdateRenderState:** Se utiliza para saber cuándo se debe de actualizar el estado de render. Sirve para solo hacer una vez la actualización del estado de render por frame y no múltiples veces si se cumplen dos casos que necesiten actualizar este estado.

- **NaokState.java:** Esta clase es la que heredan todos los estados de juego. Contiene los datos y los métodos que todos los estados deben tener.
  - **Camera:** La cámara de este estado. Se utiliza en los estados de mundos virtuales.
  - **InputType:** Es el tipo de entrada de este estado, indica si se comparte la entrada con otros estados o si tiene prioridad sobre ella.
  - **StateName:** Indica que tipo de estado es.
  - **InputEnabled:** Indica si este estado tiene acceso a la entrada actualmente o si otro estado tiene más prioridad que este.
  - **UniqueName:** Identifica a este estado con un nombre único ya que pueden existir varios estados del mismo tipo al mismo tiempo.
- **NaokStateManager.java:** Esta clase se utiliza para manejar los estados del juego que están activos en el cliente en todo momento.

  - **StateList:** Contiene una lista de los estados que están activos, cargando, por activarse o por removerse.
  - **RootNode:** Representa el nodo padre de todos los nodos que deben ser dibujados en pantalla.
  - **ChatState:** Es el estado de chat que está activo en este momento.

- **UserClientMain.java:** Es la clase principal del cliente, es la clase en donde inicia. Se encuentra en un ciclo continuo hasta que se da la instrucción de

cerrar el cliente. Dibuja todos los nodos que sean hijos del nodo principal, de esta forma se mantiene la pantalla redibujándose y actualizando la representación visual.

- **Data:** Aquí se encuentran todos los recursos utilizados en el cliente, son recursos como imágenes, texturas, modelos 3D, imágenes de cursores, etc.

- **Input:** Se encuentran las clases encargadas del manejo de entrada del cliente dentro del estado del mundo virtual.

    - **JumpAction.java:** Se encarga de hacer al personaje saltar.
    - **MenuAction.java**: Se encarga mostrar el estado de menú.
    - **MoveBackAction.java**: Se encarga hacer caminar al personaje hacia atrás.
    - **MoveForwardAction.java**: Se encarga hacer caminar al personaje hacia adelante.
    - **MoveLeftAction.java**: Se encarga de hacer caminar el personaje hacia la izquierda.
    - **MoveRightAction.java**: Se encarga de hacer caminar al personaje hacia la derecha.
    - **NaokChatListener.java**: Se encarga de manejar el input del estado del chat dentro del mundo virtual
    - **UserClientHandler.java**: Se encarga de manejar el input dentro del estado del mundo virtual.

- **Node:** Contiene los nodos de los objetos que deben ser desplegados en pantalla. Sus nombres contienen el sufijo GR, esto significa que son la representación gráfica de un objeto DT.

- **CharacterGR.java:** Contiene la información para la representación gráfica de un personaje.

  - **Model:** Es el nodo del modelo 3D a utilizar para este personaje.
  - **CanJump:** Especifica si un personaje puede saltar en este momento o no.
  - **IsWalking:** Especifica si un personaje está caminando o no.
  - **CharacterDT:** Es el CharacterDT del personaje a quien representa.
  - **Positions:** Es una lista de posiciones en las que el personaje ha estado. Sirve para poder sincronizar las posiciones con los otros clientes que estén en el mundo en el mismo momento.
  - **Forces:** Son las fuerzas que se le aplican a este personaje, como la gravedad.
  - **Velocity:** Es la velocidad que un personaje tiene en dado momento.
  - **LeftHandItemID:** El ítem que tiene equipado en la mano izquierda.
  - **RightHandItemID:** El ítem que tiene equipado en la mano derecha.
  - **HeadItemId:** El ítem que tiene equipado en la cabeza.

- **CharacterGRSet.java:** Contiene un conjunto de representaciones gráficas de personajes CharacterGR.

- **ItemGR.java:** Contiene la información para la representación gráfica de un ítem.

  - **Model:** Es el nodo del modelo 3D a utilizar para este ítem.
  - **Arrow.:** Es el nodo que contiene la flecha que apunta a este ítem si es parte de una misión o tiene alguna conversación.
  - **ItemDT.:** El ItemDT del ítem al que representa.

        - **ActionArrow:** Indica qué tipo de flecha es la que tiene este ítem apuntándole.
        - **ArrowRotation:** Se utiliza para hacer rotar la flecha que apunta a este ítem y mantener un efecto de movimiento.
        - **AnimationControler:** Contiene el control para cambiar de animaciones si el ítem es un ítem que tiene animaciones.
        - **NameBNode:** Se utiliza para mostrar el nombre del ítem sobre él.

    - **ItemGRSet.java:** Contiene un conjunto de representaciones gráficas de ítems ItemGR.
    - **RegionGR.java:** Contiene la información para la representación gráfica de una región. Estas son utilizadas para las fronteras.

    - **RegionGRSet.java:** Contiene un conjunto de representaciones gráficas de regiones RegionGR.
- **State:** Se encuentran las clases que representan estados de juego. Todas heredan de la clase NaokState que se encuentra en la carpeta Core.
    - **AlertDialog.java:** Sirve para desplegar un mensaje al usuario.
    - **BackgroundWorld.GR.java:** Se utiliza este estado mientras se está en los estados de ingresar usuario, seleccionar personaje, etc.
    - **CharacterState.java:** Es el estado que contiene la información del personaje.
    - **ChatState.java** Es el estado del chat que está activo cuando se está en un mundo virtual.
    - **ConversationDialogState.java:** Se utiliza para desplegar una conversación entre un ítem y el personaje principal.
    - **InventoryState.java:** Se utiliza para mostrar los ítems que posee el personaje principal.
    - **LoadingState.java:** Se muestra mientras está cargando otro estado.

- **LoginState.java:** Es el estado inicial en donde el usuario ingresa su nombre de usuario y su contraseña.
- **MenuState.java:** Se muestra en el mundo virtual, contiene las opciones de ver información del personaje principal, ver inventario, ver misiones, regresar al mundo o salir del mundo actual.
- **NewCharacterState.java:** Es el estado en donde se puede crear un nuevo personaje.
- **OtherCharacterState.java:** Se muestra la información del personaje de otro usuario.
- **PictureDialogState.java:** En este estado se muestran fotografías, se espera que sean fotografías con material que sirva para terminar misiones.
- **QuestState.java:** Es el estado en donde el personaje puede ver el historial de sus misiones.
- **SelectCharacterState.java:** Es el estado en donde el usuario selecciona al personaje que utilizará en el mundo virtual.
- **SelectWorldState.java:** Es el estado en donde se selecciona el mundo virtual al cual se desea entrar con el personaje.
- **StartQuest.java:** En este estado se muestra una misión en donde el personaje puede optar por aceptarla o rechazarla.
- **TestDialogState.java:** En este estado se muestra una pregunta y se dan varias opciones para que el usuario conteste.
- **WorldGR.java:** Este es el estado en donde se representa el mundo virtual.
  - **WorldDT:** Es el WorldDT que representa**.**
  - **Ítems:** Es el conjunto de ítems que contiene el mundo.
  - **Characters:** Es el conjunto de personajes que contiene el mundo, no contiene al personaje que representa al usuario del cliente.
  - **Character:** Es el personaje que está utilizando el usuario que está utilizando este cliente.
  - **InputHandler:** Es el manejador de la entrada de este estado.

- **TerrainBlock:** Es el nodo que contiene la representación gráfica del terreno en tres dimensiones.
- **PositionThread:** Es el hilo que se utiliza para mantener sincronizadas las posiciones de los personajes de este mundo virtual.
- **QueuePositionThread:** Es el hilo que se utiliza para guardar una lista de las posiciones en donde ha estado un personaje.
- **UpdateDataThread:** Es el hilo que se utiliza para mantener los datos del mundo sincronizados con otros clientes.
- **Skybox:** Es el nodo que contiene la representación gráfica del cielo en tres dimensiones.
- **Chaser:** Es la cámara que sigue al personaje principal por el mundo virtual y lo mantiene en el centro de la pantalla.
- **Regions:** Es el nodo que contiene la representación gráfica de las regiones en tres dimensiones.
- **CurrentMouseCursor:** Indica la imagen que utiliza el cursor del mouse.
- **Interpolation1, Interpolation2, Interpolation3:** Se utilizan para dividir la carga de las actualizaciones que se deben hacer constantemente en grupos.

- **Support:** Contiene clases que pueden ser utilizadas en el cliente que no son específicas a una clase en particular.
  - **CommonMethods.java:** Contiene métodos comunes que se utilizan en el cliente.
  - **ForceGR.java:** Representa una fuerza que se le aplica al personaje.
  - **ForceGRSet.java:** Es un conjunto de representaciones de fuerzas ForceGR.
  - **GameClientConfiguration.java:** Contiene valores de configuración que se utilizan dentro del cliente.
  - **TextLabel2D.java:** Es el nodo que contiene la representación gráfica del texto del nombre de los personajes y algunos ítems.

- **VelocityGR.java:** Es la representación de la velocidad en todas las direcciones que tiene el personaje principal.

- **Thread:** Contiene los hilos que se utilizan en el cliente y corren cuando hay un mundo virtual activo.
  - **InitStateThread.java:** Se utiliza este hilo para mostrar el estado en donde se indica que otro estado está siendo cargado.
  - **QueueCharacterPositionThread.java:** Este hilo es el encargado de guardar en la lista de posiciones del personaje principal las posiciones en donde el personaje ha estado.
  - **UpdateDataThread:.java:** En este hilo se actualizan los datos del mundo excepto las posiciones de los personajes.
  - **UpdatePositionThread.java:** En este hilo se sincronizan las posiciones de los personajes de este mundo

APÉNDICE III: CÓDIGO FUENTE