CENTRO DE CIENCIAS BÁSICAS

"TESIS"

MODELADO ARQUITECTÓNICO DE ESCENARIOS 3D SEMI-INMERSIVOS IMPLEMENTADOS A TRAVÉS DE GAME ENGINES

Que para obtener el grado de:

MAESTRO EN CIENCIAS, SISTEMAS Y DE LA INFORMACIÓN

presenta:

Irving Rafael Acuña Mayorga

COMITÉ DE TESIS

Director:

Dr. Francisco J. Álvarez Rodríguez

Comité Tutorial:

Dr. Juan Muñoz López

M. en C. Arturo Barajas Saavedra

Cd. Universitaria, Noviembre de 2010

Aguascalientes, Ags. 04 de Noviembre de 2010

Por este conducto autorizamos al tesista:

IC Irving Rafael Acuña Mayorga

La impresión de su documento final de tesis titulado *"Modelado Arquitectónico de Escenarios 3D Semi-Inmersivos Implementados a través de Game Engines"*, ya que cumple con los requisitos de contenido y forma exigidos en la Universidad Autónoma de Aguascalientes.

Dr. Francisco J. Álvarez Rodríguez

Director de Tesis

Dr. Juan Muñoz López

Codirector

MC Arturo Barajas Saavedra

Codirector

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES
Conmemoración del Bicentenario del inicio de la Independencia de y del Centenario de la Revolución Mexicana

Centro de Ciencias Básicas

IC. IRVING RAFAEL ACUÑA MAYORGA
PASANTE DE LA MAESTRÍA EN CIENCIAS EXACTAS, SISTEMAS Y DE LA INFORMACIÓN
P R E S E N T E .

Estimado (a) Alumno (a) Acuña:

Por medio de este conducto me permito comunicar a Usted que habiendo recibido los votos aprobatorios de los revisores de su trabajo de tesis y/o trabajo práctico titulado: **"Modelado Arquitectónico de Escenarios 3D Semi-Inmersivos Implementados a través de Game Engines",** hago de su conocimiento que puede imprimir dicho documento y continuar con los trámites para la presentación de su examen de grado.

Sin otro particular me permito saludarle muy afectuosamente.

A T E N T A M E N T E
Aguascalientes, Ags., 5 de noviembre de 2010
"LUMEN PROFERRE"
EL DECANO

DR. FRANCISCO JAVIER ÁLVAREZ RODRÍGUEZ

c.c.p.- Archivo

AGRADECIMIENTOS

Es gracias a aquellas personas que nos apoyan constantemente, que no buscan tener crédito alguno y que solo desean lo mejor para nosotros que un trabajo de gran esfuerzo puede ser llevado a cabo.

Es por ello que deseo agradecer y reconocer la colaboración de aquellas personas que hicieron que este trabajo pudiese ser llevado a cabo.

Gracias a toda mi familia, que estuvieron apoyándome y motivándome constantemente para seguir adelante, especialmente a mi primo Olin, que no solo me apoyo moralmente, sino que además contribuyó con el diseño 3D para la realización del prototipo de esta investigación, así como a mi hermano Johnathan, que logró mantenerme con la frente en alto en los momentos más difíciles.

Gracias a mi director y sinodales que dedicaron su tiempo para guiarme en el desarrollo de un trabajo de investigación, así como por sus aportaciones para crear un documento del cual me siento satisfecho y orgulloso.

Y gracias a todas aquellas personas que me inspiraron y me motivaron para continuar estudiando y crecer tanto profesionalmente como personalmente.

RESUMEN / ABSTRACT

Cuando un equipo virtual colabora remotamente requiere que cada uno de sus miembros perciban un escenario virtual de la misma forma aún cuando este se esté modificando en tiempo real, para ello es necesario presentarles a todos un reflejo del objeto de trabajo con el detalle suficiente para que todos puedan percibirlo de igual manera. Por lo tanto se requiere que los programas de software que se desarrollan para simular escenarios virtuales en 3D permitan que varios usuarios puedan interactuar simultáneamente y de forma semi-inmersiva con los objetos que les son presentados por estos sistemas.

Para solucionar dicho problema, se propone diseñar una arquitectura de software para el desarrollo de aplicaciones 3D, en donde distintos usuarios puedan interactuar en un escenario virtual compartido simultáneamente con distintos objetos, implementando técnicas semi-inmersivas de realidad virtual, servicios distribuidos, procesamiento multi-hilo y las funciones básicas de un game engine.

El modelo arquitectónico facilita el desarrollo de software 3D multi-usuario y permite al usuario final visualizar e interactuar de forma más simple y natural con el entorno virtual, donde los principales beneficiados de la investigación serán, entre otros: los desarrolladores de videojuegos 3D, permitiéndoles introducir mecanismos de control que permitan a los jugadores interactuar con los videojuegos de maneras más simples y naturales; a las instituciones médicas, dándoles la posibilidad de visualizar e interactuar con órganos reconstruidos tridimensionalmente; a los arquitectos e ingenieros civiles, concediéndoles visualizar e interactuar con edificaciones virtuales de manera natural; así como también a los estudiantes, presentándoles una alternativa de aprendizaje más compleja, interesante y motivacional.

ÍNDICE DE CONTENIDO

ÍNDICE DE FÍGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Descripción del Contexto del Problema de Investigación

El desarrollo y mejora de nuevas tecnologías interactivas ha impactado significativamente en las ciencias y las artes tradicionales. Debido a que el nacimiento de nuevas disciplinas y la evolución de técnicas de interacción han generado una gran necesidad de nuevas formas de comunicación, por lo que áreas como la realidad virtual (RV) han afectado de forma considerable aspectos culturales y educativos [22].

La realidad virtual es una de las áreas de investigación y desarrollo más reciente en la industria de la computación. Sus aplicaciones potenciales van desde el diseño de interiores hasta simulación de vuelos aeronáuticos. Existen diversas formas de emplear la tecnología de realidad virtual, teniendo como premisa crear medios más intuitivos para que humanos y computadores trabajen juntos [22].

Para entender mejor el texto anterior hay que comprender primeramente lo que la realidad virtual es, la cual es definida típicamente en términos tecnológicos como una simulación en diferentes escenarios virtuales en tres dimensiones que responden a los movimientos humanos [23]. A este fin le auxilian sensores o dispositivos de hardware (computadoras, lentes, audífonos, guantes sensibles al movimiento, etc.) que el usuario manipula de tal manera que el escenario percibido es asociado con el escenario virtual (EV) y no con el real. El proceso de manipulación es controlado por una computadora central que recopila la información de los movimientos físicos y los aplica sobre el EV [24].

Sin embargo, a pesar de que lo definido anteriormente es válido no llega a satisfacer textualmente la idea que se tiene de la realidad virtual, por lo que una definición más sencilla y más apropiada se menciona enseguida:

"*La realidad virtual es aquella forma de trabajo donde el hombre puede interactuar totalmente con la computadora, generando ésta espacios virtuales donde el humano puede*

desempeñar sus labores y donde el humano se comunica con la computadora a través de dispositivos de interacción" [36].

Utilizando gafas u otros dispositivos se ve y se entra en una presentación o espacio virtual creado por computadora de una realidad alternativa en la que se participa. Al mover la cabeza o dar órdenes, esta escena virtual queda dominada y cambia armónicamente. La cabeza o la mano parecen ser transportadas a y expuestas al moverse dentro de la escena generada por computadora.

La realidad virtual no es intimidatoria ni es del dominio exclusivo de adictos a los videojuegos y a la tecnología. Sus aplicaciones tampoco están restringidas a lo puramente tecnológico o científico. Es un medio creativo de comunicación al alcance de todos.

Cabe recordar que la realidad virtual explota todas las técnicas de reproducción de imágenes y las extiende, usándolas dentro del entorno en el que el usuario puede examinar, manipular e interactuar con los objetos expuestos. Una vez diferenciado lo que es real de lo virtual podemos es posible dar una definición bastante específica de lo que es realidad virtual:

"La realidad virtual es la manipulación de los sentidos humanos (siendo actualmente el tacto, la visión y la audición) por medio de entornos tridimensionales sintetizados por computadora en el que uno o varios participantes acoplados de manera adecuada al sistema de computación interactúan de manera rápida e intuitiva, de tal manera que la computadora desaparece de la mente del usuario dejando como real el entorno generado por la computadora".

Para el desarrollo de RV es necesario el uso de cuatro tecnologías [25]:

- Los dispositivos visuales que dan un grado de inmersión al usuario en el EV y bloquean las sensaciones del mundo real.
- Los sistemas de renderización gráfica que cambian las imágenes continuamente.

- Los sistemas de rastreo que reportan continuamente la posición y orientación del usuario.
- Los sistemas de construcción y mantenimiento que se encargan de los modelos realistas y detallados del EV.

La *inmersión* es un concepto clave en los sistemas RV, donde los usuarios se vuelven parte del mundo simulado, ó bien, también puede definirse como el sentimiento de realidad que tiene un usuario en un EV. El término "inmersión" es una descripción de tecnología, por lo que puede ser medido, de esta forma el grado de inmersión se incrementa por agregar componentes que ayuden a dar la sensación de lo "real" [24].

La *presencia* es otro concepto importante, el cual se deriva de una experiencia con alto grado de inmersión. Se define como la sensación psicológica de "estoy ahí", en el EV [24].

De acuerdo a las definiciones anteriores, se usa para fines propios de esta investigación el término *semi-inmersivo*, el cual es definido como el sentimiento de realidad que el usuario tiene en un EV utilizando los mínimos componentes de hardware posibles, obteniendo una alta percepción del mundo virtual, pero sin perder la percepción del mundo real.

No obstante, aunque la RV es una tecnología ampliamente usada hoy en día, no existen métodos o técnicas de desarrollo propios para el diseño de herramientas de software de este tipo [3][10]. Por lo que la ausencia de una herramienta de software para el desarrollo de aplicaciones de Realidad Virtual conduce a demorar los tiempos de realización del producto y a la posibilidad del uso ineficiente de las bibliotecas básicas de gráficos como son la OpenGL y la DirectX. Esto último puede afectar considerablemente la calidad de la visualización en cuanto a velocidad (cuadros por segundo) o la calidad de la imagen. Además de presentar defectos significativos, tales como pueden ser la falta de interactividad y desplazamiento en los EV [28][29].

1.2. Relevancia y Justificación

Cuando un equipo virtual colabora remotamente requiere que cada uno de sus miembros perciban un escenario virtual de la misma forma, aún cuando este se esté modificando en tiempo real, para ello es necesario presentarles a todos un reflejo del objeto de trabajo con el detalle suficiente para que todos puedan percibirlo de igual manera. Por lo tanto se requiere que los programas de software que se desarrollan para simular escenarios virtuales en 3D permitan que varios usuarios puedan interactuar simultáneamente y de forma semi-inmersiva con los objetos que les son presentados por estos sistemas.

Se han identificado diversos factores que han sido presentados por los desarrolladores de software en este tipo de sistemas, entre los cuales se incluyen la administración consistente de información distribuida; garantizar la interactividad en tiempo real; contender con el limitado ancho de banda de la red, de los recursos de renderización y de los tiempos de cómputo del procesador. Donde esta última se debe a las nuevas manufacturas de hardware, que implementan arquitecturas de procesador multi-núcleo, presentando un gran reto a los desarrolladores de software por no estar familiarizados con la programación concurrente y su alta complejidad [21] [12].

Los beneficiados más claros son para los usuarios finales interesados en las aplicaciones diseñadas bajo el esquema de la arquitectura a desarrollar en esta investigación, sin embargo, además de los usuarios finales, existe otra gama de personas beneficiadas por dicha arquitectura:

- *Los desarrolladores de videojuegos 3D*: el crecimiento de la industria de los videojuegos provocó una reciente tendencia en dicho mercado, la cual está llevando a desarrollar controles con mayor complejidad, como lo bien suelen ser demasiados botones en un simple control. Es por ello que la industria busca introducir mecanismos de control alternativos que permitan a los jugadores interactuar con los videojuegos de maneras más simples y más naturales, de tal forma que puedan tener ricos juegos inmersivos [4].

- *Las instituciones médicas*: ya que la posibilidad de visualizar e interactuar con órganos reconstruidos tridimensionalmente mientras están inmersos en un escenario virtual, provee a los médicos una forma muy natural de investigar la anatomía de un paciente [5].
- *Los arquitectos e ingenieros civiles*: debido a que el visualizar edificios o construcciones dentro de un escenario virtual e interactuar de manera natural con el entorno 3D, permite a los clientes concebir de una manera más grafica el proyecto a realizar [4].
- *En la enseñanza académica*: usar programación de juegos 3D no es solo útil para trabajar con objetos espaciales, lo cual es requerido para algunos cursos de gráficos por computadora, sino que también ayuda a la motivación de los estudiantes, ya que ellos tienden a percibir los juegos 3D como algo complejo e interesante [30].

1.3. Descripción General del Enfoque, Métodos y Técnicas de Investigación

De acuerdo con los trabajos de Mora [32][33] y de Hevner [31] se considera para este documento el propósito de investigación de tipo diseño conceptual descriptivo, ya que se desarrolló una arquitectura especializada a un problema en específico, con el fin de poder generar múltiples sistemas virtuales con una finalidad en particular. El tipo de investigación es tipo aplicada específica descriptiva, ya que se evaluó el modelo arquitectónico por medio de la medición de un sistema virtual desarrollado bajo las especificaciones planteadas por la arquitectura.

El modelo de investigación propuesto consiste de cuatro fases para el desarrollo de la arquitectura propuesta ([32]):

- *Formulación del Problema de Investigación*: En esta etapa inicial se definieron primeramente el contexto, el problema y la relevancia de investigación. Posteriormente se establecieron los objetivos, la hipótesis/premisas y las preguntas

de investigación. Lo anterior se realizó por medio de una serie de iteraciones donde se fue limitando tanto el problema de investigación así como el alcance de la propia investigación.

- *Fase de Revisión Bibliográfica*: En esta fase se concentraron las actividades de recolección y análisis de estudios (teorías, arquitecturas, modelos, métodos, técnicas, tecnologías, entre otros) relacionados a esta investigación, determinando sus contribuciones y limitaciones. Como ejemplo de lo anterior se concentra en el estado del arte de este trabajo documentación como: diseño 3D; game engines; sistemas distribuidos; metodologías para el desarrollo de arquitecturas, además de arquitecturas para el diseño de game engines distribuidos y el manejo de sistemas multi-hilo en una arquitectura de un game engine.
- *Fase de Desarrollo*: Esta fase tuvo como objetivo el desarrollo del artefacto conceptual de la investigación, o bien, el diseño arquitectónico planteado a desarrollar. Para ello se utilizó MEDARISH, la cual es una guía para el desarrollo de arquitecturas de software, propiciando así a plasmar en UML el objeto conceptual desarrollado en este trabajo, facilitando así su entendimiento.
- *Fase de Evaluación*: En este punto de la investigación se evaluó la arquitectura por medio de la comparación de un prototipo de software diseñado y construido bajo los esquemas de construcción de la propia arquitectura contra los estándares de la misma arquitectura.

Figura 1. Modelo General de Investigación

1.4. Descripción de los Capítulos de la Tesis

En esta sección de la tesis se ha planteado el contexto general del problema de investigación. En los capítulos siguientes se plantea el desarrollo de la investigación de la siguiente manera:

- En el capítulo 2 se hace una descripción detallada sobre el problema de investigación, así como los objetivos, hipótesis, premisas y la metodología utilizada para el proyecto de investigación.
- En el capítulo 3 se describen las bases teóricas relacionadas con los "game engines" y los sistemas distribuidos.
- El capítulo 4 plasma la arquitectura propuesta por esta investigación, la cuál es una solución al problema planteado en los primeros capítulos de este documento.
- El capítulo 5 plantea la evaluación a la arquitectura de investigación, así como los resultados obtenidos en el proyecto de investigación.
- Por último en el capítulo 6 se encuentran las conclusiones obtenidas de la investigación, así como recomendaciones sobre futuros proyectos de investigación.

2. ESTRUCTURA DE LA INVESTIGACIÓN

2.1. Descripción Específica del Problema de Investigación

El principal problema que enfrenta esta investigación es satisfacer la necesidad de facilitar la abstracción de escenarios virtuales que permitan la interacción de usuarios remotos con los objetos del escenario a través de medios electrónicos de comunicación.

La solución propuesta consiste en utilizar las funciones básicas de los ”game engines”, ya que estos además de contar con módulos para la creación de mundos virtuales, también permiten hacer uso de componentes reutilizables, reduciendo tanto la complejidad y el tiempo de desarrollo de videojuegos en 3D.

Al dividir la carga de trabajo de un sistema por medio de la integración de algoritmos de paralelización, el rendimiento de los sistemas incrementa significativamente, por lo que es posible realizar procesos más complejos sobre un procesador, así como también una mejor administración de los datos que fluyen entre los distintos usuarios remotos.

Utilizar un modelo con un alto nivel alto de abstracción, como el que proporciona una arquitectura de software permitirá generar una solución que pueda reutilizarse para resolver problemas similares presentados en el desarrollo de diferentes programas de realidad virtual.

2.2. Pregunta/Objetivo General de Investigación

Bajo los elementos planteados anteriormente para resolver la problemática de investigación, surge la siguiente *pregunta de investigación*:

¿Qué elementos, características, funciones e interrelaciones deben considerarse para generar un diseño arquitectónico para sistemas virtuales 3D que permitan a usuarios remotos interactuar simultáneamente y de forma semi-inmersiva?

Para responder dicha pregunta y resolver el problema de investigación planteado se presenta el *objetivo general* de esta investigación:

- Diseñar una arquitectura de software que implementé las tecnologías de los "game engines" para permitir la creación de sistemas virtuales donde los usuarios remotos puedan interactuar de forma simultánea y semi-inmersiva en los escenarios 3D.

Así mismo, a partir del anterior objetivo se desprenden los siguientes *objetivos específicos* de investigación:

- Caracterizar un game engine, que tome en cuenta algoritmos de paralelización.
- Establecer un marco de trabajo que contemple incorporar nuevas tecnologías de hardware, algoritmos y nuevos elementos para evolucionar y mejorar los sistemas virtuales distribuidos.
- Establecer un marco de trabajo que permita la interacción simultánea de los múltiples usuarios remotos.
- Establecer un marco de trabajo que contemple incorporar diferentes tecnologías semi-inmersivas.

2.2.1. Hipótesis, Proposición, Premisa o Conjetura Principal de la Investigación

Premisa principal de investigación:

- Un modelo arquitectónico de software simplifica el desarrollo de programas de software que simulen escenarios virtuales en 3D en donde varios usuarios podrán interactuar simultáneamente y de forma semi-inmersiva con los objetos que les son presentados por estos sistemas.

Premisas adicionales:

- Los algoritmos multi-hilo implementados en una arquitectura de game engine incrementan el desempeño para ejecutar algunas tareas.
- Los sistemas distribuidos permiten incrementar el número de usuarios interconectados simultáneamente sin que el sistema se colapse.
- El uso de dispositivos semi-inmersivos incrementa la sensación de realidad en un sistema virtual.

2.3. Descripción Específica del Enfoque, Métodos y Técnicas de Investigación usadas

Enfoque de la investigación: El enfoque específico que esta investigación tomó es el de *diseño conceptual descriptivo*, ya que se generó un artefacto conceptual constituido por un *modelo arquitectónico* con la finalidad de crear sistemas virtuales de software donde se permite la interacción simultánea y de forma semi-inmersiva de usuarios remotos.

Técnicas de recolección de la información: Revisión bibliográfica de estudios realizados. Debido a la alta abstracción del objeto de investigación resulta muy difícil la creación de dicha arquitectura sin tener fundamentos previos, es por ello que la revisión bibliográfica utiliza los avances de estudios similares como pilares para el desarrollo del objeto de estudio. También se contempla la recolección de los datos arrojados por los prototipos desarrollados en la evaluación, lo que permite un mejor análisis de la arquitectura y la mejora de ésta.

Técnicas de evaluación: La evaluación de la investigación es por *prueba de concepto*, por medio de la aplicación de la arquitectura en casos de uso. Se realizó por medio de la comparación y medición entre el software desarrollado bajo la arquitectura propuesta en este trabajo contra la misma arquitectura propuesta.

3. MARCO TEÓRICO

3.1. Revisión de Teorías Base Usadas

3.1.1. Objetos 3D

Un *gráfico 3D* se origina mediante un proceso de cálculos matemáticos sobre entidades geométricas tridimensionales producidas en una computadora, cuyo propósito es conseguir una proyección visual en dos dimensiones para ser mostrada en una pantalla [40].

El entorno 3D consiste en una representación de coordenadas en un plano cartesiano, donde las variables X, Y y Z determinan la localización de cualquier objeto en el espacio. El centro de dicho espacio cartesiano es el origen, el cual tiene las variables iguales a cero (Figura 2 - lado izquierdo) [37][38][39].

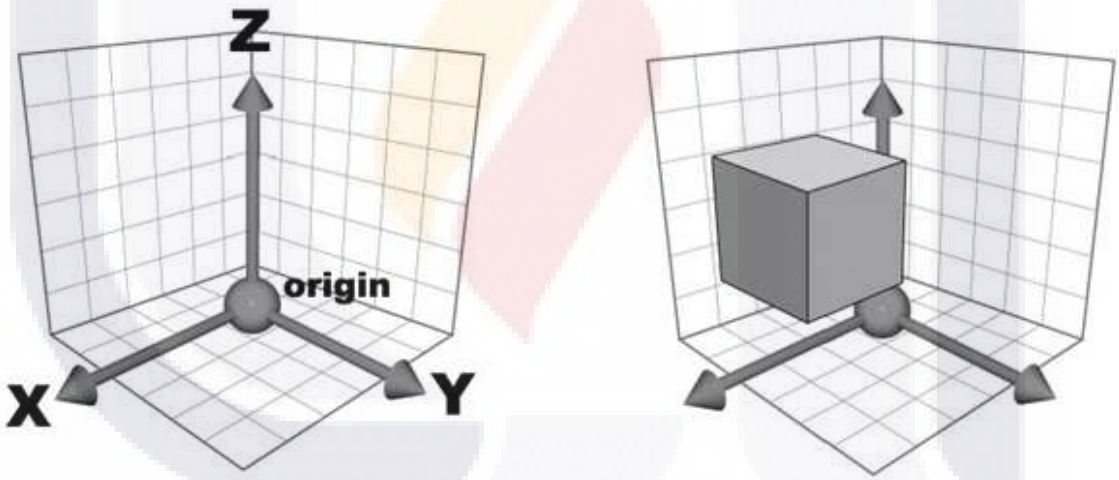


Figura 2. Sistema de Coordenadas 3D

Existen 2 tipos de operaciones básicas que realizan los objetos virtuales [37][38]:

- *Traslación*: Esta operación se ejecuta sobre el sistema de coordenadas del mundo virtual (escena), y consiste en cambiar el valor de las variables X, Y y/ó Z para posicionar el objeto con respecto al origen de la escena (Figura 2 - lado derecho).
- *Rotación*: Esta operación es ejecutada en el sistema de coordenadas propio del objeto virtual en donde el origen es el pivote o centro del objeto virtual y consiste

en la rotación propia del objeto con respecto a los ángulos de las variables X, Y y/ó Z (Figura 3).

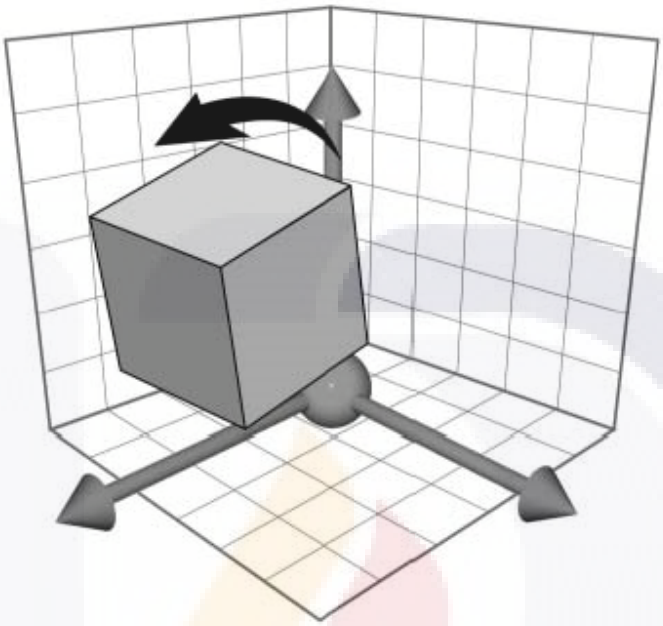


Figura 3. Rotación de un Objeto Virtual

Existen 5 etapas para la creación de un objeto 3D [40]:

1. *Modelado*: Consiste en dar forma a objetos individuales para usarlos posteriormente en la escena. Para ello se utilizan diferentes técnicas matemáticas:
    a. *Polimallas ó Polígonos*: Este tipo de modelado es el más común y básico de los modelos geométricos usados en 3D, donde es considerado un polígono como cualquier forma plana y cerrada (con su primer y último vértices unidos). Los polígonos pueden tener cualquier número de vértices (siempre y cuando excedan de 2), pero en el ámbito del modelado 3D es mejor manejar polígonos de 3 o 4 vértices debido a que es mucho más fácil el suavizado del objeto virtual [37][38].

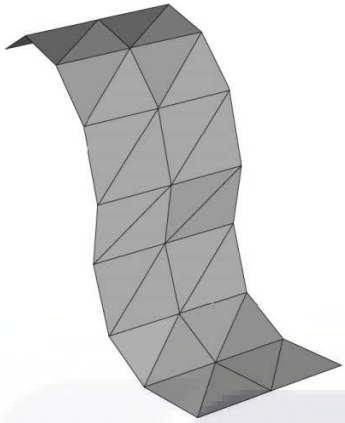


Figura 4. Polimalla

A pesar de que un polígono puede ser renderizado, un polígono por sí mismo no tiene volumen, espesor y rigidez, por lo que cuando varios polígonos se conectan, compartiendo lados y vértices se forma una polimalla, la cual es moldeada para dar forma a un objeto virtual (Figura 4) [37].

b. *NURBS*: A diferencia de los polígonos, en este tipo de modelado se trabaja con curvas definidas matemáticamente. Es decir, el modelado se realiza por medio de interpretaciones de formulas matemáticas, por lo que es bastante fácil y rápido manejar formas con curvas suaves, sin embargo no es apropiado para el manejo de ángulos rectos, además de que el tiempo del renderizado es más tardado (Figura 5) [38][39].

Figura 5. Ejemplo de NURBS

2. *Texturizado*: Después de realizar el modelado, el objeto virtual queda de color uniforme y liso, es por medio de texturas que se adhiere el realismo necesario al objeto [38][40].

3. *Sombreado e Iluminación*: Es la parte fundamental de la escena para imprimirle realismo, por medio de creación de diversas luces colocadas en diferentes puntos [38][40].
4. *Animación*: Los objetos virtuales pueden ser animados de diferentes maneras:
    a. *Transformaciones básicas en los tres ejes (XYZ)*: Rotación y traslación [40].
    b. *Forma*:
        i. *Esqueletos*: Se asigna un esqueleto a un objeto, el cual afecta directamente el movimiento a las porciones correspondientes del modelo [40].
        ii. *Deformadores*: Se utilizan cajas de deformación que produzcan deformaciones sinusoidales [40].
        iii. *Dinámicas*: Simulaciones de ropa, cabello y de objetos [40].
5. *Renderización*: Proceso llevado a cabo por una computadora para generar una imagen 2D que el usuario pueda apreciar de forma automática a partir de un modelo tridimensional existente. Una animación es una serie de renderizaione secuenciales [38][40].

En la Tabla 1. Aplicaciones para Modelado 3D muestra una lista de las diferentes aplicaciones que existen para el modelado 3D [40]:

Nombre	Compañía
Maya	Autodesk (antes alias|wavefront)
SOFTIMAGE|XSI	Autodesk
3DStudio MAX	Autodesk (antes AVID y antes de Microsoft)
LightWave	Newtek
Blender	Blender (OpenSource)
Cinema 4D	Maxon
Houdini	Side Effects
Rhinoceros	Rhino
Pov-ray	Povray
Cheetah 3D	Cheetah 3D

Tabla 1. Aplicaciones para Modelado 3D

3.1.2. Game Engine

Un *game engine* es un tipo de middleware que facilita el rápido desarrollo de un videojuego [12] o bien es un sistema de diseño de software el cual integra una colección de diversos objetos de código de computadora, con la finalidad de desarrollar videojuegos [9][10].

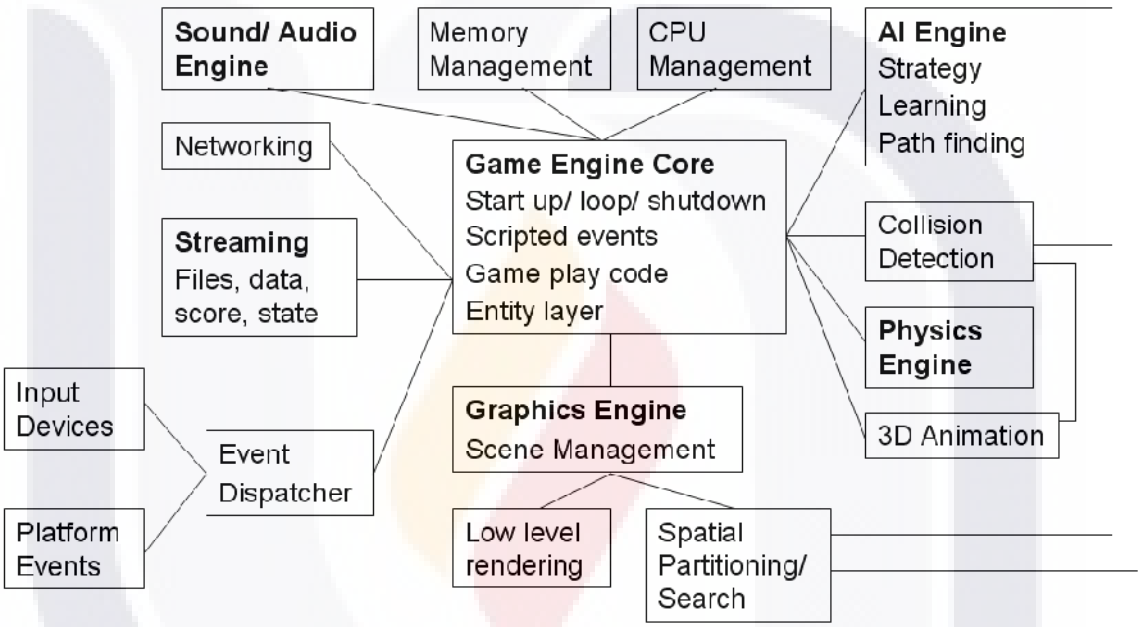


Figura 6. Diseño de alto nivel de un Game Engine [4]

Frecuentemente un game engine es diseñado con una arquitectura basada en componentes (Figura 6), la cual facilita el intercambio entre diferentes modos de interacción con la aplicación [9][12]. Dicha arquitectura permite al sistema proveer distintos servicios, como lo son los siguientes módulos [8][9][12]:

- Módulo gráfico en 2D o 3D.
- Módulo de animación.
- Módulo de física.
- Módulo de detección de colisiones.
- Módulo de entrada y salida de datos.

- Módulo de sonido.
- Módulo de inteligencia artificial.
- Módulo de comunicación en red.
- Módulo de bases de datos.
- Módulo de la interfaz de usuario (GUI).
- Módulo de funciones de guardar/cargar.

Figura 7. Arquitectura de un Game Engine basada en niveles

Sin embargo, existen otras arquitecturas para los videojuegos multijugador, pudiendo estar basadas en modelos centralizados (cliente servidor) [11][4], o bien distribuidos (peer-to-peer), donde la principal diferencia entre ellas consiste en el tráfico de los paquetes en la red [11]. Un game engine comúnmente es diseñado bajo una arquitectura cliente-servidor compuesta de tres capas (existen arquitecturas cliente-servidor de hasta n capas): donde la capa de servidor es la encargada de administrar la red, la construcción de los escenarios

virtuales, análisis de bases de datos, almacenamiento persistente, entre otras funciones mas; la capa compartida administra la red a bajo nivel, la detección e intersección de colisiones, las simulaciones, la física y la animación 3D, entre otros más; por último, la capa de cliente es responsable de la predicción y corrección de la red, del código del juego del cliente, la animación 3D completa, los sonidos y el render 3D (Figura 7) [4].

Hay algunos "game engines" que solo cuentan con la capacidad de render 3D en tiempo real, lo que conlleva a los programadores de videojuegos tener que desarrollar el resto de las funcionalidades o componentes, estos "game engines" son llamados "3D engines". Los "game engines" actuales permiten la representación de mundos virtuales 3D por medio de una representación orientada a objetos, facilitando el diseño del juego y una eficiente renderización de vastos escenarios virtuales [9].

Existen algunas librerías de bajo nivel que de igual manera permiten el desarrollo de videojuegos, tales como SDL, Allegro Library o ClanLib. Estas son comúnmente usadas en juegos que proveen al hardware independiente acceder a otros componentes de hardware [9].

Los "game engines" normalmente son clasificados de acuerdo a los diferentes géneros que hay, donde esta clasificación comercial permite proveer optimizaciones y características específicas para cada uno de los géneros, pero usualmente a la expensa de la flexibilidad [2]. Entre los principales géneros se encuentran los siguientes [2][4][9][10]:

- *FPS Engines (First Person Shooter Engine)*: Simula un ambiente en 3D donde los usuarios ven mundo a través de los ojos de su carácter. Regularmente (mas no necesariamente obligatorio) el objetivo es crear un EV donde el jugador tenga que matar a otros jugadores o NPC's (Caracteres Virtuales).
- *RTS Engines (Real-Time-Strategy Engines)*: Por lo general simulan un ambiente 3D desde una perspective isométrica. Consisten en permitir al usuario tanto crear como destruir un gran número de edificios y unidades (por lo general tropas militares) en tiempo real y no por turnos.

- *RPG Engines (Role Playing Game Engines)*: Estos engines pueden utilizar gráficos 2D, Isometric Engines o EV 3D en su totalidad. Tienen la finalidad de permitir a los usuarios asumir el rol de sus caracteres en una situación ficticia, donde cada una de sus decisiones y acciones repercutirá en diferentes posibles situaciones ficticias.
- *MMOG Engines (Massively Multiplayer Game Online Engines)*: Son similares a alguno del resto de los engines, la diferencia radica en que soportan cientos de miles de jugadores simultáneamente.

Debido a la gran facilidad de diseño de escenarios 3D que tienen los "game engines", otras áreas ajenas a los videojuegos han mostrado gran interés por el mencionado middleware. Entre estas áreas destacan la ciencia, medicina, efectos especiales para la industria cinematográfica, las aplicaciones militares [10][4], los museos virtuales [3] y el diseño arquitectónico [4].

Los "game engines" tienen como principal característica ser una base para una herramienta de visualización y ahorrar significativamente el tiempo de implementación por reutilizar la funcionalidad implementada en los videojuegos. Los "game engines" modernos ofrecen un render rápido y de calidad, además del soporte interactivo y multi-usuario que es difícil de obtener usando herramientas existentes de visualización [4].

Entre otras de las características de los "game engines" es el hecho de que es posible desarrollar nuevos módulos en lenguajes cotidianos de programación. La Tabla 2 muestra una serie de características, capacidades y restricciones de diez "game engines", tanto comercial como código libre. Es posible observar que la gran mayoría de ellos tienen el lenguaje C++ como una extensión del sistema, lo que permite a los desarrolladores crear sus propias funciones cuando le lenguaje propio del sistema no cuenta con dichas funciones [4].

No.	Game Engine	Licencia	Extensibilidad de Ambiente	Extensiones del Lenguaje de Programación
1	Torque Game Engine	Commercial	Very Good	Torque Script, C++
2	Doom Engine 3	Commercial	Very Good	Scripting Language, C++
3	Unreal Engine 2	Commercial	Very Good	Scripting Language, C++
4	EasyWay	Open Source	Poor	Java
5	Sauerbraten (Cube 2)	Open Source	Good	Scripting Language, C++
6	Similus	Open Source	Poor	C++
7	Simple Direct Engine	Open Source	Poor	C#
8	Ghost Engine	Open Source	Poor	C++
9	Raydium	Open Source	Poor	C
10	Drome	Open Source	Good	C++

Tabla 2. Características generales de los Game Engines [3]

Comúnmente la manera para desarrollar un EV es utilizando una herramienta gráfica, nombrada *Virtual Reality Engine* (VRE), la cual ha sido por años el estándar para representaciones graficas de computadora en 3D en la ciencia y la arquitectura. Sin embargo requiere un alto costo para su desarrollo, ya que necesita equipo costoso y un alto nivel educativo por parte del personal informático [3][10]. Usualmente esta tarea es implementada por la técnica no inmersiva, Virtual Reality Modeling Language (VRML), la cual es bastante difícil de codificar ya que requiere altas habilidades de programación [3][34]. Es por ello que los “game engines” se presentan como una alternativa para desarrollar un EV, ya que presentan los mismo resultados de una manera mucho más sencilla [3][10].

Como limitaciones los “game engines” tienen la característica de separar la visualización en mundos 3D estáticos y objetos dinámicos con los que el usuario puede interactuar. Otra limitante es el tamaño máximo de los mapas, ya que usualmente los “game engines” trabajan con mapas pequeños. Esto porque el diseño de aplicaciones de visualización con “game engines” requiere más trabajo que una librería de visualización [4], además de que una alta calidad en las imágenes 3D consume una alta cantidad de

recursos del procesador, por lo que en ocasiones es necesario tener una computadora con hardware bastante óptimo y costoso [3][10].

3.1.3. Sistemas Distribuidos

Los *sistemas distribuidos* son sistemas en el que los componentes de hardware y software se encuentran separados físicamente, pero se mantienen unidos por medio de una red de comunicación [13][15]. En adición a la anterior definición Coulouris afirma que la comunicación entre dichos componentes es realizada únicamente mediante el paso de mensajes [19].

Los sistemas distribuidos cuentan con las siguientes características:

- Procesan información de forma multi-procedural, evitando confiar en la comunicación compartida de un sistema central [14].
- Cuentan con diversas características particulares, como lo son concurrencia de los componentes, carencia de un reloj global y fallos independientes de los componentes [19].
- Un sistema distribuido es percibido como un único e integrado sistema de computación, a pesar de que los componentes se encuentran dispersos geográficamente [13][15].

Existen además diversos desafíos que hay que tomar en cuenta al momento de diseñar un sistema distribuido [19], como lo son:

- La *heterogeneidad*, que no es más que la existencia de variedad y diferencias entres los componentes que conforman a un sistema distribuido.
- La *extensibilidad*, que es la característica que determina si un sistema distribuido puede ser extendido, es decir, al cual se le puedan agregar nuevos servicios.
- La *seguridad*, el cual es el aspecto preocupado por la seguridad de los recursos.

- La *escalabilidad* es el factor que determina si un sistema distribuido conserva la efectividad original al incrementarse significativamente en el número de usuarios y recursos.
- El *tratamiento de fallos* ocurre cuando hay fallas tanto en el hardware y el software y producen resultados incorrectos o detener algún servicio en su totalidad.
- La *transparencia*, la cual oculta al usuario la separación de los componentes, de tal forma que se perciba al sistema como un todo y único sistema integrado.

La arquitectura de un sistema distribuido es la encargada de asignar la responsabilidad de los componentes, así como su ubicación física, sus implicaciones principales están en la fiabilidad, seguridad y prestación del sistema resultante [19].

Existen dos tipos de arquitecturas tradicionales que permiten a múltiples usuarios participar simultáneamente en una misma aplicación: *cliente-servidor* y *peer-to-peer*. Sin embargo existe otro tipo de arquitectura que trata de cumplir con el mismo fin que las anteriores: *multiservidores* [13] [16] [17][18][19].

La arquitectura cliente-servidor (Figura 8) se compone de un único servidor, el cuál provee uno o varios servicios a distintos clientes de forma simultánea [13] y de manera individual [17]. En este tipo de arquitectura cada cliente envía mensajes al servidor, ya sea para la solicitud de un servicio, o bien para entablar una comunicación con otros cliente, de esta manera el servidor puede filtrar los mensajes para reducir la cantidad de mensajes necesarios que se dirigen a cada cliente a través de la red. Sin embargo uno de los principales problemas de esta arquitectura es que a medida que el número de clientes incrementa también lo hace el número de paquetes necesarios en la red de forma significativa, llevando así al servidor a un estado en el que la administración y el manejo de los mensajes demoran considerablemente [14].

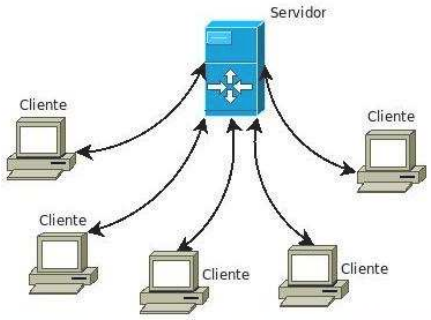


Figura 8. Arquitectura Cliente-Servidor [13]

La arquitectura peer-to-peer (Figura 9) se conforma de distintos clientes interconectados entre sí, donde cada nodo tiene las mismas responsabilidades que los demás, de esta manera la interacción entre los equipos es cooperativa con el objetivo de fragmentar y realizar una tarea en particular. Una de las ventajas que se presenta en una arquitectura de este tipo es el gasto mínimo de comunicación debido a la conexión directa entre los clientes [13][14][17], sin embargo una gran cantidad de clientes simultáneos conlleva a tener una saturación de paquetes en la red, por lo que la red se vuelve difícil de administrar y manejar [18].

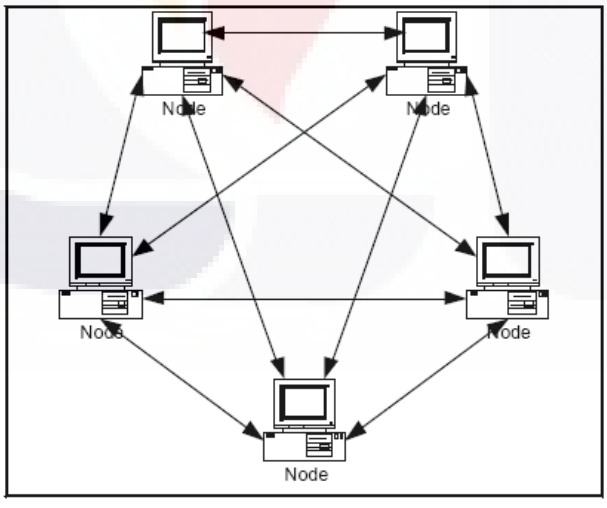


Figura 9. Arquitectura Peer-to-Peer [17]

La arquitectura Multiservidor (Figura 10) tiene como finalidad corregir los problemas de comunicación comúnmente presentados en una arquitectura cliente-servidor [16][18]. Esto se realiza por medio de conectar distintos servidores entre sí, de manera que se dividen

el número de objetos sobre el cual se basa el servicio y se distribuyen entre ellos [13]. En el área de la realidad virtual, una función muy particular es la de separar en regiones un escenario virtual, y asignar cada región a un servidor diferente, de manera que la probabilidad de saturar la carga de trabajo de un servidor se reduce significativamente ya que los clientes pueden ser conectados dinámicamente a los diferentes servidores [18].

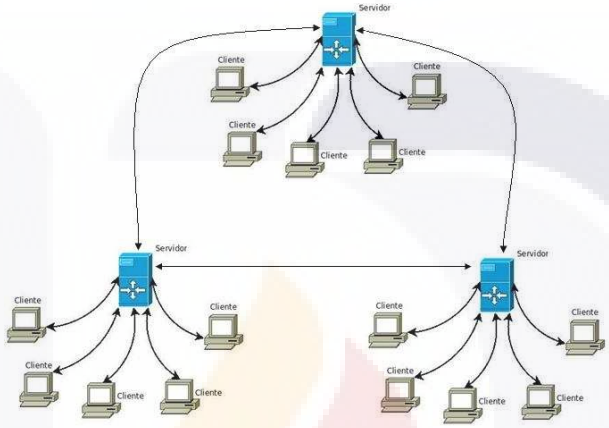


Figura 10. Arquitectura Multiservidor [13][17]

Los sistemas distribuidos deben de considerar para un buen funcionamiento diversas características como lo son [13]:

- El tiempo de respuesta entre componentes.
- Disponer de los recursos adecuados.
- Fiabilidad en el sistema.
- Tolerancia a fallas.
- Seguridad (Confidencialidad, Integridad y disponibilidad).

Un factor importante a considerar en el diseño de un sistema distribuido, es el tipo de interacción que habrá entre los componentes que conforman dicho sistema, para el cual se establece un modelo de interacción, en donde se definen las reglas y políticas de comunicación que tendrán los componentes [19]. Existen dos variantes de este modelo de interacción:

- *Sistemas distribuidos síncronos*: En este tipo de modelo simple existe una fuerte restricción sobre el tiempo. En donde se conocen el tiempo mínimo y máximo para la entrega de mensajes entre componentes. Este tipo de sistemas es comúnmente usado cuando existe la necesidad de que los mensajes sean entregados en un tiempo determinado [19].
- *Sistemas distribuidos asíncronos*: En este tipo de sistemas no existe ninguna presuposición de recursos, es decir, no existen limitaciones sobre la velocidad de procesamiento, así como tampoco es necesario un tiempo límite para la entrega de mensajes [19].

En el área de la RV es bastante común manejar EV, dentro de los cuáles interactúan una gran cantidad de usuarios, entre ellos destacan los MMOG (Massive Multiplayer Online Games), simulaciones militares, escenarios de entrenamiento, encuentros virtuales, entre otros. Dichas aplicaciones para soportar esa gran cantidad de usuarios simultáneos necesitan una gran topología de red incremental [35]. La arquitectura de red cliente-servidor es muy común en los sistemas virtuales debido a que el servidor puede realizar el filtrado de mensajes para ser manejado por cada cliente. Usualmente esta arquitectura funciona para pocos clientes, sin embargo, conforme el número de clientes aumenta también lo hace la cantidad de mensajes, saturando la carga de trabajo del servidor [17][18]. Para evitar ese problema se utiliza la arquitectura multi-servidor, en donde la cantidad de usuarios se divide entre los diferentes servidores o bien, cada servidor tiene una responsabilidad diferente en el mundo virtual [17]. La topología peer-to-peer tiene como ventajas la reducción del costo de comunicación, así como la distribución del trabajo entre los distintos nodos, sin embargo presenta dos fuertes problemas: la saturación de la red debido a la gran cantidad de paquetes en circulación y la poca facilidad de incremento que posee la topología [17][35].

3.2. Diseño de Arquitecturas de Software

Existen diferentes metodologías con diferentes procedimientos para el diseño de arquitecturas de software, a continuación se describen algunas de ellas:

MEDARISH

La **ME**todología para el **D**iseño de **A**rquitecturas de **R**eferencia que **I**ntegra **S**istemas **H**eredados (MEDARISH) [41] establece un proceso que describe los pasos necesarios para la construcción de una arquitectura de software, tomando en cuenta un conjunto estructurado de requerimientos y restricciones relacionados a un dominio en particular.

Esta metodología logra el perfeccionamiento y la evolución de los modelos arquitectónicos por medio de iteraciones, generando un ciclo en el cual los modelos arquitectónicos de referencia se encuentran en constante evolución adaptándose a nuevas necesidades.

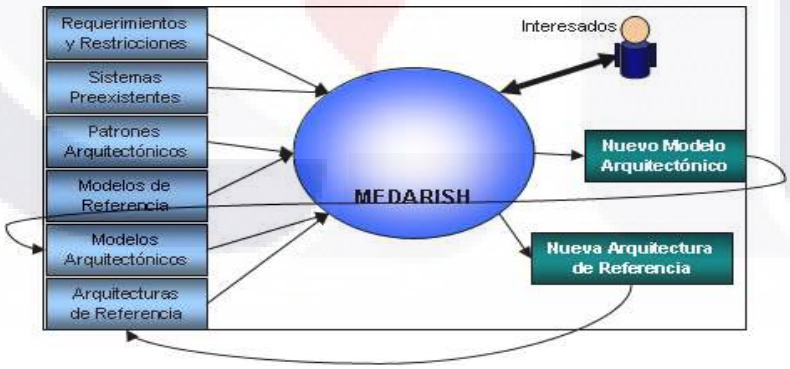


Figura 11. Diseño Conceptual de MEDARISH

Además de contar con requerimientos y restricciones, también se cuenta con otros elementos de entrada, los cuales conllevan a generar modelos arquitectónicos específicos

para la construcción de sistemas y modelos arquitectónicos de referencia para el diseño de nuevas arquitecturas (Figura 11).

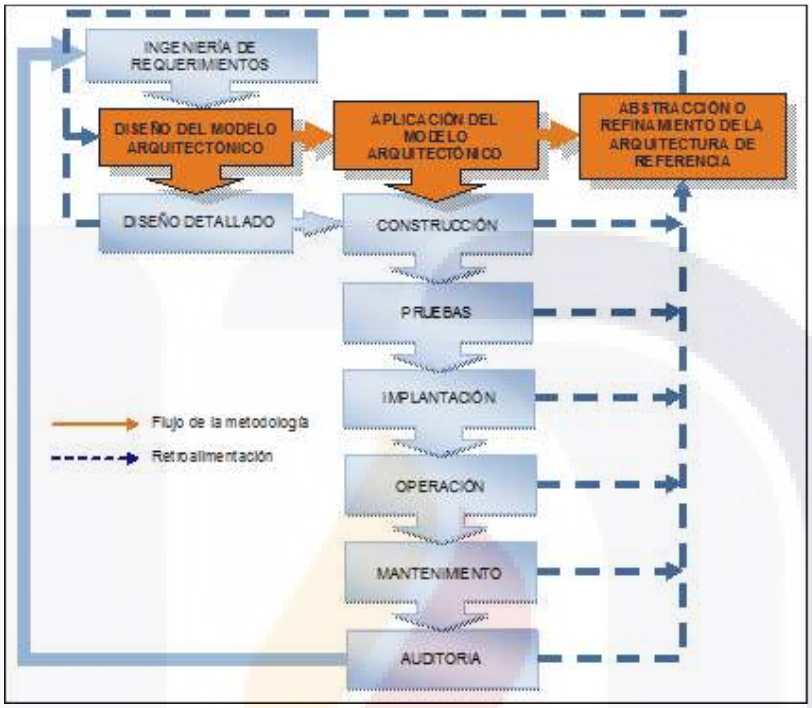


Figura 12. Fases de Metodología MEDARISH

El proceso para la creación de arquitecturas que describe la metodología puede dividirse en tres grandes fases (Figura 12):

1. Diseño del modelo arquitectónico.
2. Aplicación del modelo arquitectónico.
3. Abstracción y refinamiento de la arquitectura de referencia.

La primera fase comprende las actividades necesarias para tomar y validar un conjunto de requerimientos y restricciones que provienen de la etapa de ingeniería de requerimientos, abstraer los elementos arquitectónicos nuevos y heredados, diseñar, documentar y validar un modelo arquitectónico.

En la segunda fase el modelo arquitectónico se aplica en la construcción del sistema. Una vez construido el sistema se hace una evaluación de la utilidad que tuvo el modelo arquitectónico para construir el sistema final y se efectúan mediciones para determinar la variabilidad de la arquitectura del sistema real contra la arquitectura diseñada originalmente.

La tercera fase comprende las actividades necesarias para abstraer el nuevo modelo arquitectónico de referencia cuando no había uno existente o para crear una nueva versión de la arquitectura de referencia que perfeccione aquella de la que se partió inicialmente.

Proceso de Arquitecturas de Software

Rozanski y Woods [42] describen una serie de actividades con el fin de definir una arquitectura de software:

1. Acordar el alcance y contexto, las restricciones y los principios arquitectónicos básicos.
2. Identificar y enganchar a los interesados.
3. Identificar y utilizar los escenarios arquitectónicos.
4. Utilizar los estilos arquitectónicos y los patrones.
5. Producir modelos arquitectónicos.
6. Documentar la arquitectura.
7. Validar la arquitectura.

3.3. Principales Trabajos Relacionados

El enfoque de este proyecto, el cual se orienta en desarrollar un modelo arquitectónico nos lleva a revisar trabajos enfocados con el desarrollo de diferentes arquitecturas relacionadas con el desarrollo de escenarios virtuales, entre los que destacan:

Multi-Threaded Game Engine Desing

El trabajo de James Tulip [12] discute los diferentes temas, enfoques y compromisos que se deben de considerar en el diseño de un game engine multi-hilo. Se plantea claramente la problemática que existe hoy en día con la demanda de recursos que los sistemas tienen sobre los procesadores multi-núcleo. Habla sobre como las actuales arquitecturas de "game engines" no estan diseñadas para obtener el máximo rendimiento de los procesadores multi-núcleo. Para corroborar lo anterior, se hicieron varios monitoreos de algunos videojuegos ejecutándose sobre diferentes procesadores, tanto simples como multi-núcleo, obteniendo que las diferencias en cuanto a tiempo no son realmente significativas (Tabla 3).

Juegos de Computadora	**Desempeño (Cuadros Por Segundo)**		
	Procesador de 1 núcleo Athlon 64 3800+	**Procesador de 2 núcleos Athlon 64 X2 3800+**	**Decremento en desempeño con 2 núcleos**
Serious Sam 2	106	93	-13 (14%)
Quake 4	99	86	-13 (15%)
Far Cry	75	67	-8 (18%)

Tabla 3. Disminución de desempeño con 2 núcleos en juegos actuales

Además de lo anterior Tulip identifica cinco factores para la programación concurrente [12]:

- *Ley de Amdahal*: Establece el límite fundamental para ganar velocidad potencial en relación al procesamiento concurrente.
- *Paralelismo de Tareas y Datos*: El paralelismo de tareas consiste en separar y ejecutar de forma independiente y asíncrona las tareas, las cuales pueden ser coordinadas por puntos de sincronización definidos. El paralelismo de datos establece que los datos se rompan en pedazos independientes, donde el mismo procesamiento es aplicado independientemente y de forma paralela a cada uno de los pedazos.
- *Sincronización*: La palabra "sincronización" en el contexto de la concurrencia es usado en el sentido de "coordinado".
- *Granularidad*: En este contexto, se refiere a granularidad a la cantidad de trabajo ejecutado por tarea concurrente.
- *Balanceo de Carga*: Se refiere al concepto donde la carga de procesamiento permitida a las tareas en ejecución en paralelo deben de ser similares.

Por último Tulip afirma que la mezcla de paralelismo en tareas y datos es un acercamiento óptimo a la explotación del paralelismo en los videojuegos. Sin embargo en el contexto de arquitecturas de "game engines" multi-hilo no es difícil encontrar las oportunidades para paralelización, sino el problema es el controlar la programación y ejecución de las muchas simultáneas operaciones.

Complex Systems-Based High-Level Architecture for Massively Multiplayer Games

Charles River [20] introduce en su libro una arquitectura de videojuegos basada en las propiedades emergentes de los sistemas complejos para determinar el comportamiento determinista en los juegos MMP (Massively Multiplayer).

Hay que comprender primeramente que los sistemas complejos son sistemas altamente estructurados, los cuales tienen un número de elementos interactuantes que están organizados en estructuras que varían en tamaño y profundidad. Esas estructuras están

sujetas a procesos de cambio que no pueden ser descritos por una regla o reducidos a un nivel de explicación. Los sistemas complejos pueden ser distinguidos por la presencia de ciclos de retroalimentación, entornos abiertos y límites del sistema indeterminados.

River afirma que los MMP presentan muchas de las propiedades y comportamientos que pueden ser encontrados en aplicaciones industriales de sistemas complejos, lo cual es atribuido a la alta interacción entre miles de usuarios, NPCs, grandes mundos persistentes, un número no determinístico de estados del juego y múltiples entradas impredecibles por parte de los usuarios con diferentes antecedentes socioeconómicos, étnicos y culturales.

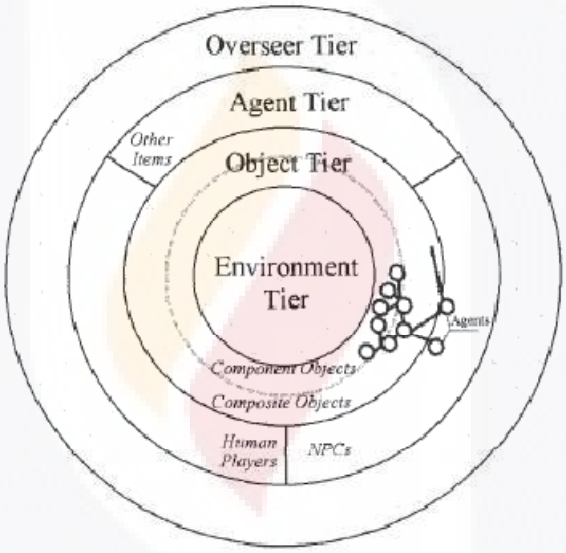


Figura 13. Arquitectura de Alto Nivel basada en Sistemas Complejos

La arquitectura multicapa basada en sistemas complejos (Figura 13) consiste de cuatro capas:

- *Capa de ambiente*: Define las propiedades globales del mundo del juego virtual que es simulado. Incluye la simulación, gráficos, audio, física, inteligencia artificial, manejadores de eventos y los recursos de hardware.
- *Capa de objeto*: Consiste de componentes y composiciones de componentes. La emergencia inicia con los componentes de la capa objeto y se propaga hacia arriba a través de la interacción y la agregación componentes.

- *Capa de agente*: Representa todos los objetos del juego que tienen funciones de alto nivel en el juego. Incluye jugadores humanos, NPCs y otros objetos que pueden presentar un grado significativo de inteligencia artificial.
- *Capa supervisora*: Es la última y más significativa capa en la arquitectura de River. Hace uso de agentes para implementar múltiples gobernadores artificiales del juego para proveer las políticas que tendrán influencia sobre los objetos del juego en el ambiente del juego.

Las arquitecturas multicapa tradicionales exhiben ortogonalidad entre las capas para reducir el acoplamiento entre ellas, es decir las operaciones de cada capa pueden usarse en cualquier dirección con cualquier otra capa. En la arquitectura propuesta por River la ortogonalidad se presenta entre la capa ambiente y el resto de las capas, esto para asegurar que la estabilidad del ambiente es independiente de la evolución natural de las otras capas, mientras que las capas de objeto, agente y supervisor están diseñadas para trabajar sinérgica y cohesivamente para permitir el incrementar la interacción y la emergencia con los objetos del juego.

3.4. Resumen de Contribuciones y Limitaciones de los Principales Trabajos Relacionados

A continuación se presenta un resumen de los trabajos que muestran las diferentes arquitecturas de los “game engines”, los cuales presentan diferentes perspectivas que son útiles para el cumplimiento del objetivo de esta investigación.

Hay que mencionar que el análisis siguiente mostrará tanto las contribuciones como las limitaciones de cada trabajo, con el fin de determinar el grado en que cada estudio es significativo para satisfacer el problema de investigación planteado.

Trabajo:	Multi-threaded Game Engine Design.
Propuesto por:	James Tulip [12].
Contribuciones:	Establece las métricas para medir el desempeño de un game engine multi-hilo; Identifica los factores clave para la programación concurrente; propone los elementos que deben ejecutarse paralelamente.
Limitaciones:	No plantea en qué momento debe realizarse la paralelización de tareas.

Trabajo:	Game Programming Gems 6.
Propuesto por:	Charles River [20].
Contribuciones:	Plantea una arquitectura de “game engines” que se adapta los constantes cambios de un MMP; el ambiente virtual y su contenido deben ejecutarse paralelamente a los usuarios y a los objetos que interactúan dentro del EV.
Limitaciones:	La arquitectura se concentra bajo aspectos que cambian constantemente, por lo que su estructura es demasiado compleja al igual que el funcionamiento de cada una de sus capas, de tal manera que su implementación es demasiado compleja.

4. DESARROLLO

En este capítulo se presenta la propuesta de un diseño arquitectónico que facilita el desarrollo de sistemas virtuales 3D que permiten a múltiples usuarios remotos interactuar simultáneamente y de forma semi-inmersiva entre sí.

La arquitectura propuesta integra componentes de las diferentes arquitecturas de "game engines" para crear un modelo de referencia que pueda ajustarse a las diversas necesidades de los usuarios y a los diferentes entornos donde puede aplicarse la realidad virtual.

Para desarrollar la arquitectura se utilizó la metodología MEDARISH, la cual sirve para el diseño, tanto de arquitecturas de software como de arquitecturas de referencia

La aplicación de MEDARISH sobre esta investigación consistió solamente en la ejecución de las dos primeras fases: diseñando un modelo arquitectónico detallado y llevándolo a la aplicación por medio del desarrollo de un sistema de realidad virtual semi-inmersivo, dando como resultado final una arquitectura específica y aplicada para el desarrollo de sistemas virtuales semi-inmersivos distribuidos remotamente. Sin embargo aún queda pendiente la generalización del modelo arquitectónico generado para poder crear una arquitectura de referencia y así generar en un futuro diferentes arquitecturas específicas para cada una de las situaciones que existen alrededor del contexto de la realidad virtual.

4.1. Arquitectura Multi-Hilo y Semi-Inmersiva para Sistemas Virtuales Distribuidos Implementando Game Engines

La arquitectura propuesta en esta investigación integra componentes de varias arquitecturas de "game engines" para poder ajustarse a los diferentes entornos donde puede aplicarse la realidad virtual, además de contemplar algoritmos multi-hilos para incrementar

el rendimiento de los sistemas virtuales y dispositivos semi-inmersivos de hardware con el fin de aumentar la sensación de estar dentro del mundo virtual.

Como primer aspecto para el diseño de la arquitectura se contempló la parte física, y es debido a que la investigación se enfoca en el diseño de un sistema distribuido que se optó por adoptar una arquitectura multi-servidor (Figura 14), ya que de acuerdo a la literatura este tipo de configuración permite distribuir los mundos virtuales de un mismo sistema entre los servidores existentes, distribuyendo la carga de red de los múltiples usuarios conectados.

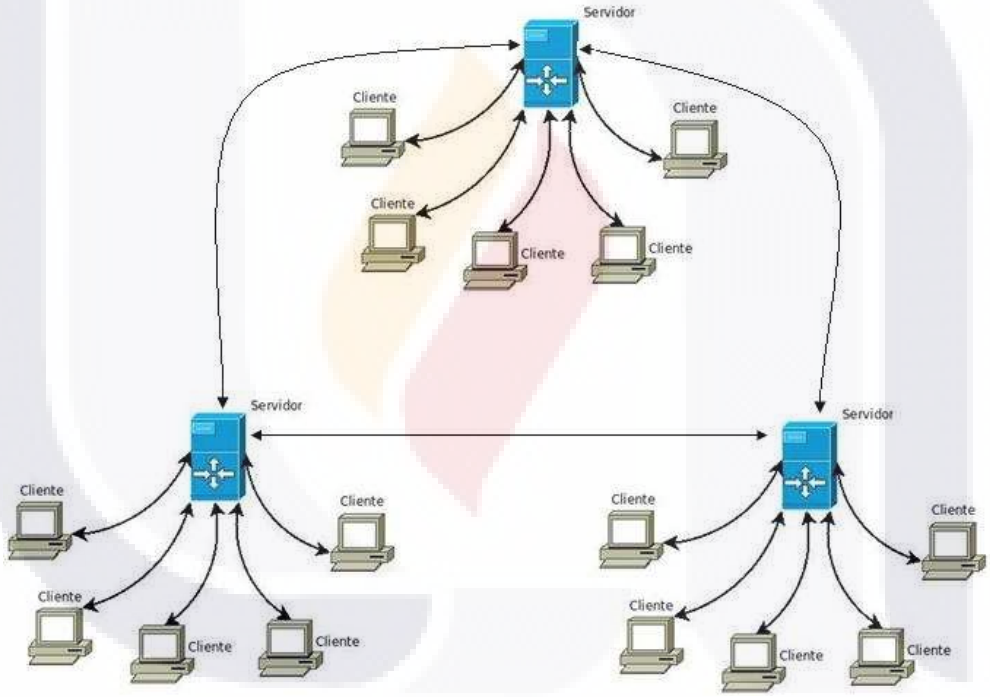


Figura 14. Arquitectura Multi-Servidor

El siguiente paso realizado fue la determinación de las tareas que se ejecutan en cada equipo de la arquitectura, para ello se tomó como referencia las funciones de los “game engines” de acuerdo a una arquitectura cliente-servidor (Figura 15).

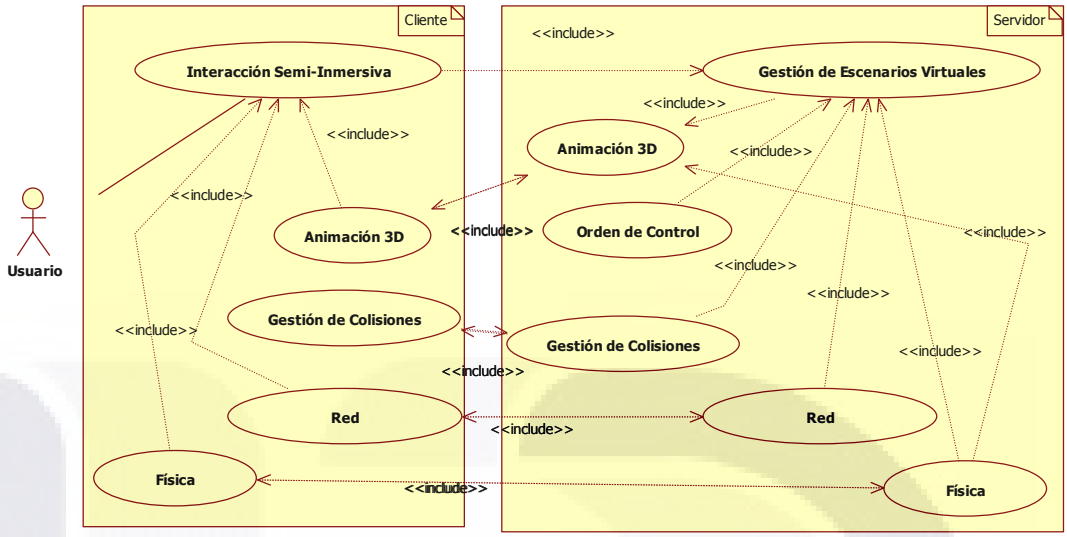
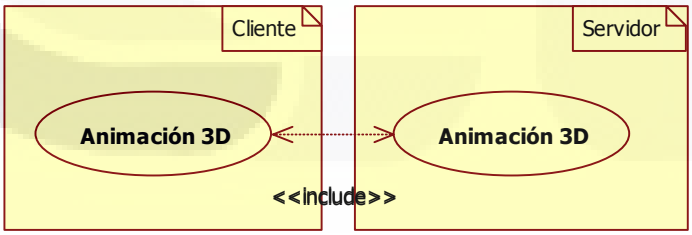


Figura 15. Caso de Uso Global

A continuación se describen las especificaciones de los casos de uso más relevantes que fue necesario tomar en cuenta:

1. *Nombre de Caso de Uso*: Interacción Semi-Inmersiva
    a. *Descripción corta*: Este caso de uso permite al usuario interactuar de forma semi-inmersiva con el mundo virtual.
    b. *Actores Involucrados:* Usuario
    c. *Pre condiciones:* Tener un escenario virtual creado del lado del servidor.
    d. *Pos condiciones:* Se establece un modo de interacción entre el mundo real con el mundo virtual.
    e. *Flujo Básico.*
        i. El usuario interactúa con el sistema cliente por medio de dispositivos de hardware para la entrada y salida de datos en el sistema.

2. *Nombre de Caso de Uso*: Animación 3D
    a. *Descripción corta*: Este caso de uso permite al usuario percibir el contenido de un mundo virtual de forma visual.
    b. *Pre condiciones*: Tener establecida una interacción semi inmersiva del lado del cliente; tener un escenario virtual creado del lado del servidor.
    c. *Pos condiciones*: Genera una representación visual del escenario virtual.
    d. *Flujo Básico.*
        i. El modulo animación 3D servidor indica los movimientos que realiza algún personaje u objeto virtual dentro del escenario virtual.
        ii. El modulo animación 3D cliente debe interpretar las indicaciones del modulo animación 3D servidor y mostrarlas visualmente al usuario por medio del modulo interacción semi-inmersiva.



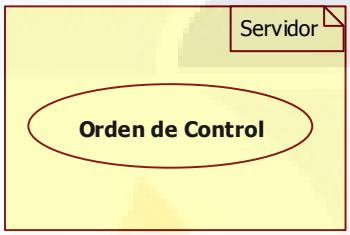
3. *Nombre de Caso de Uso*: Orden de Control
    a. *Descripción corta*: Este caso de uso permite ubicar geográficamente a cada usuario, NPC y objeto virtual dentro del mundo virtual.
    b. *Pre condiciones*: Tener un escenario virtual creado del lado servidor.

c. *Pos condiciones*: Establece una nueva posición y un nuevo estado para cada usuario, NPC y objeto virtual.

d. *Flujo Básico.*

i. Inicia cuando el modulo gestión de escenarios virtuales recibe las acciones realizadas por cada usuario.

ii. Determina la nueva posición geográfica, la nueva dirección hacia donde ve y la nueva acción que realiza por cada personaje virtual.



4. *Nombre de Caso de Uso*: Gestión de Colisiones

a. *Descripción corta*: Este caso de uso permite identificar si existen colisiones en el mundo virtual.

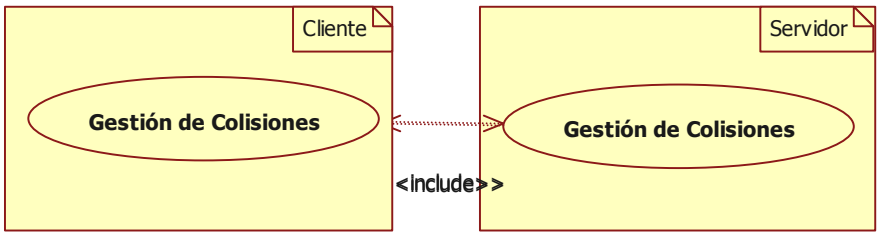
b. *Pre condiciones*: Tener un escenario virtual creado en el lado del servidor.

c. *Pos condiciones*: Determinar la posible existencia de colisiones dentro del escenario virtual.

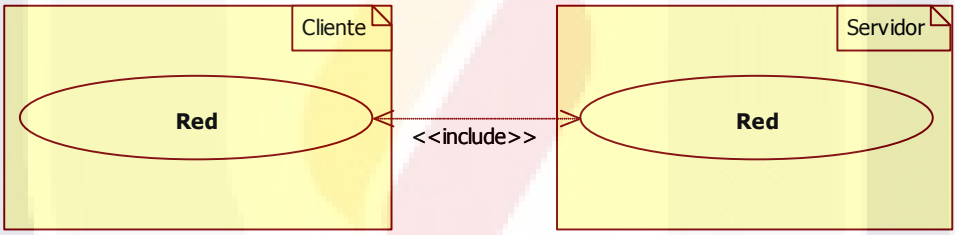
d. *Flujo Básico.*

i. Inicia cuando existe una colisión dentro del mundo virtual.

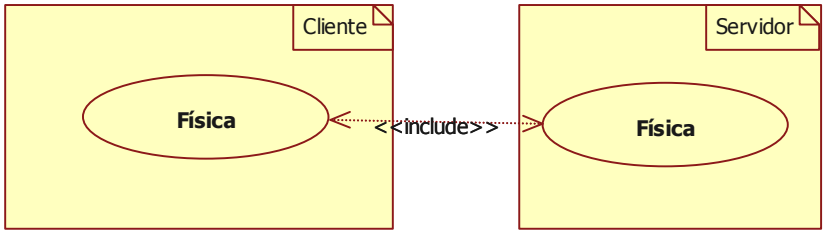
ii. Se le informa al modulo gestión de escenarios virtuales sobre las colisiones existentes y las acciones consecuentes de dichas acciones.



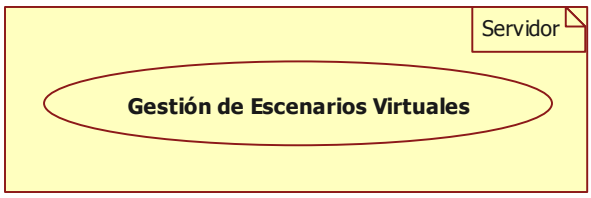
5. *Nombre de Caso de Uso*: Red
    a. *Descripción corta*: Este caso de uso permite al cliente y al servidor tener una comunicación constante.
    b. *Pre condiciones*: Tener un escenario virtual creado del lado del servidor; tener una interacción semi-inmersiva del lado del cliente.
    c. *Pos condiciones*: Se establece un canal de comunicación entre el sistema cliente y el sistema servidor.
    d. *Flujo Básico.*
        i. Inicia cuando el sistema servidor crea una sesión de red.
        ii. El sistema cliente puede acceder solamente a un servidor con una sesión de red existente.
        iii. Mientras la sesión de red exista se mantiene una comunicación constante entre el sistema cliente y el sistema servidor.



6. *Nombre de Caso de Uso*: Física
    a. *Descripción corta*: Este caso de uso permite aplicar las leyes de la física dentro del mundo virtual.
    b. *Pre condiciones*: Tener un escenario virtual creado del lado del servidor; tener una interacción semi-inmersiva del lado del cliente.
    c. *Pos condiciones*: Establece las leyes de la física dentro del mundo virtual.
    d. *Flujo Básico.*
        i. Inicia cuando se aplica alguna ley de la física dentro del mundo virtual.
        ii. El movimiento tanto de los personajes virtuales como de los objetos que yacen ahí son limitados por las leyes de la física que aplican en ese escenario virtual.



7. *Nombre de Caso de Uso*: Gestión de Escenarios Virtuales
    a. *Descripción corta*: Este caso de uso permite administrar un escenario virtual.
    b. *Pre condiciones*: Tener una interacción semi-inmersiva creada; tener creado un canal de comunicación; tener establecidas las política de la física a aplicar; tener determinadas la existencia de colisiones; tener establecidas las nuevas posiciones y acciones para cada personaje virtual.
    c. *Pos condiciones*: Administra el contenido de un escenario virtual.
    d. *Flujo Alternativo*: En caso de no existir un escenario virtual genera uno nuevo.
    e. *Flujo Básico.*
        i. Inicia cuando el servidor ejecuta en el sistema un escenario virtual.
        ii. Establece las políticas y restricciones dentro del sistema virtual.
        iii. Gestiona el contenido del escenario virtual.
        iv. Auxilia al modulo animación 3D servidor para generar una representación visual del escenario virtual.



De acuerdo a lo planteado en los párrafos anteriores y a lo aportado por River [20] se establece el siguiente modelo arquitectónico general (Figura 16), donde su propuesta consta de tres capas, cuyo objetivo es dividir las principales funciones de los sistemas por niveles.

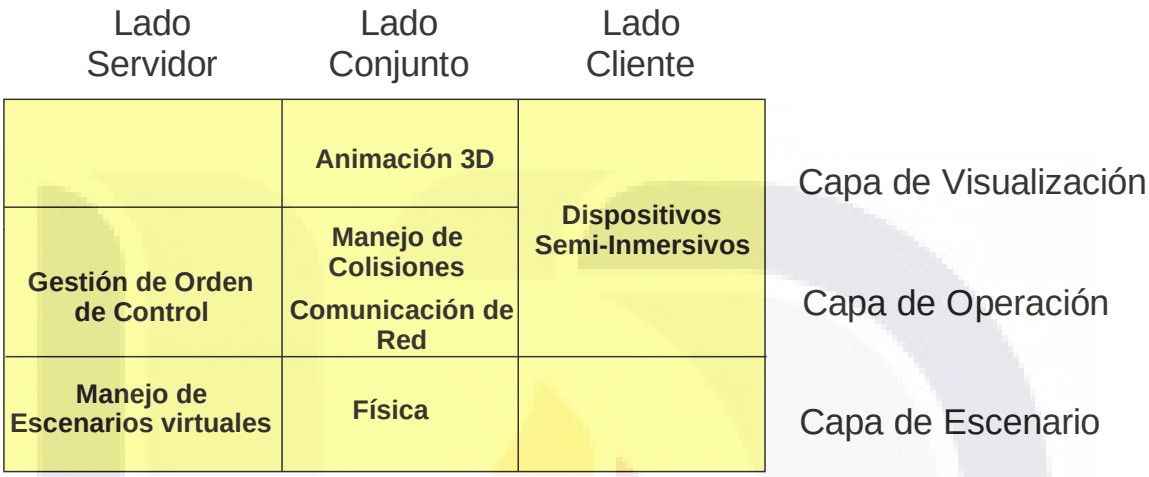


Figura 16. Diseño de Alto Nivel de una Arquitectura Multi-Hilo y Semi-Inmersiva para Sistemas Virtuales Distribuidos Implementando Game Engines

La capa de visualización abarca las tareas directamente relacionadas con el usuario final, por ello se contempla la entrada y salida de datos por medio de dispositivos semi-inmersivos. Además de lo que es la animación 3D, lo cual es lo que percibe el usuario finalmente.

La capa de operación también trabaja con los dispositivos semi-inmersivos, solo que aquí se hacen los procesos para interpretar los datos de entrada y crear los datos de salida. Esta capa también trabaja la comunicación de datos entre los diversos clientes y el servidor. Por último, el servidor actualiza las nuevas posiciones de los objetos virtuales después de haber verificado la existencia de colisiones entre dichos objetos virtuales.

La capa de escenario realiza tanto la administración de los escenarios virtuales como la aplicación de la física, permitiendo crear políticas y reglas totalmente independientes al usuario.

El paso procedente fue establecer de acuerdo al modelo anterior los módulos necesarios para poder llevar a cabo la construcción de un sistema virtual distribuido semi-inmersivo (

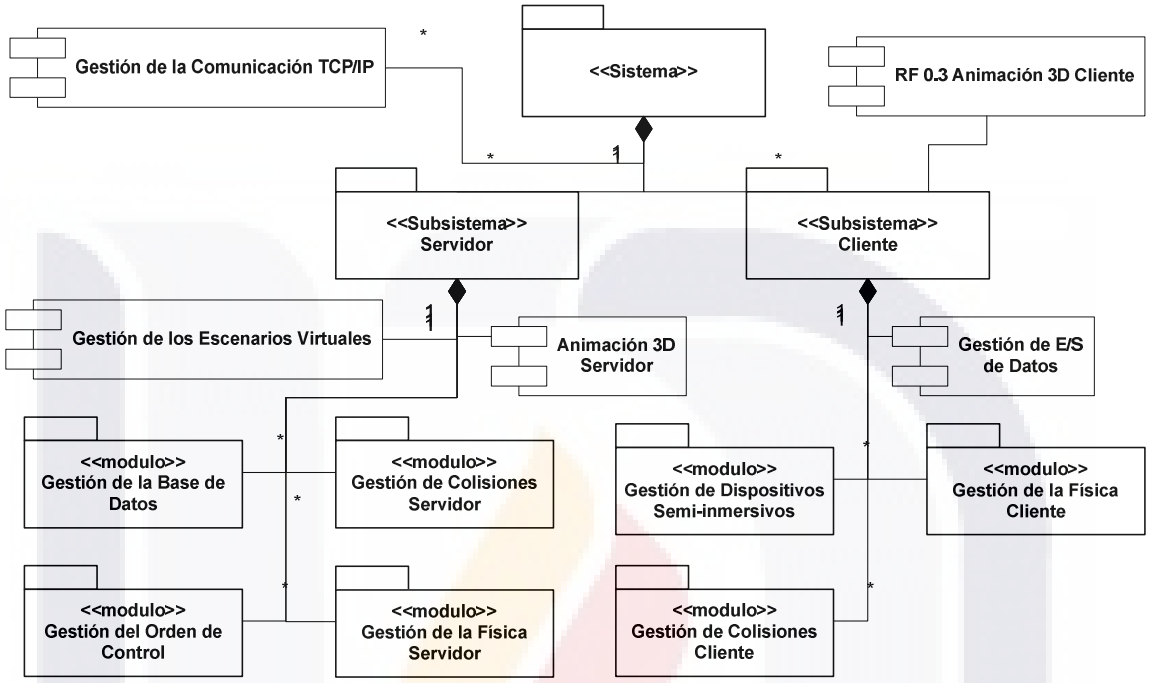


Figura 17).

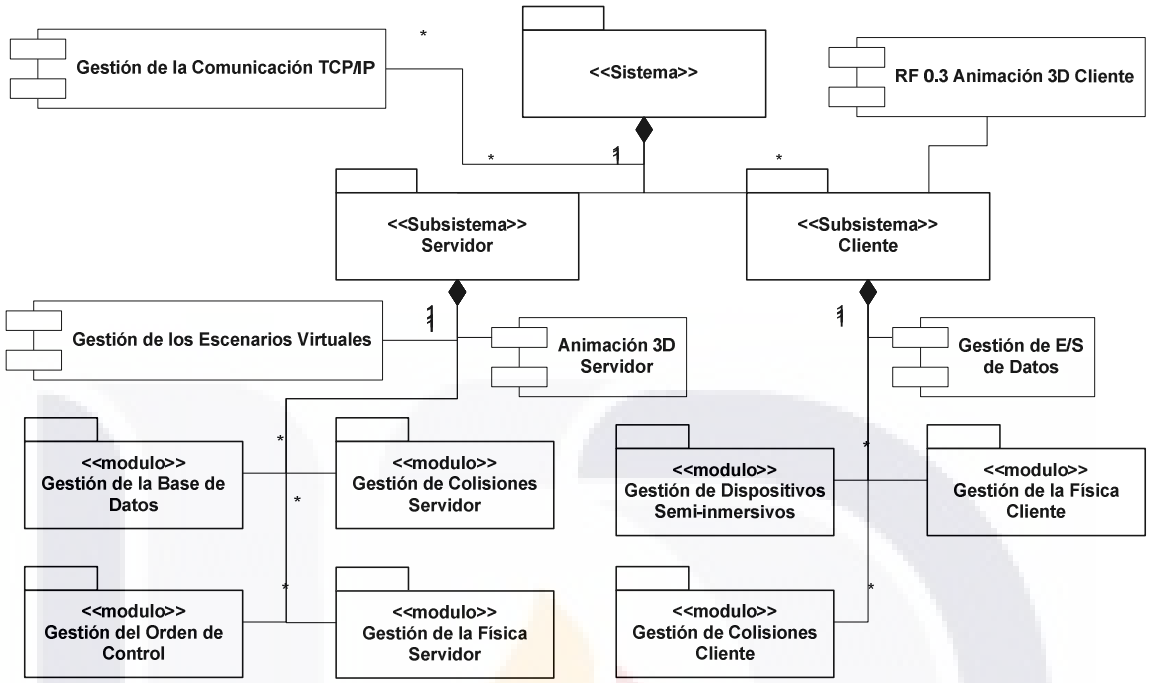


Figura 17. Diagrama de Módulos

Todos los módulos están contenidos en un sistema principal, el cual se divide en 2 subsistemas, además de utilizar un componente con las funciones necesarias para llevar a cabo la comunicación entre los subsistemas.

1. *Servidor*: Este subsistema contiene un componente para la gestión de los escenarios virtuales y otro para la animación en la parte del servidor. Además cuenta con 4 módulos para diversas gestiones: base de datos, orden de control, física y colisiones.

2. *Cliente*: Este subsistema cuenta con un componente para llevar a cabo la animación en el cliente y otro para la administración de entrada y salida de datos. Además cuenta también con 3 módulos para las gestiones de: física, dispositivos semi-inmersivos y colisiones.

Como complemento del paso anterior se especificaron los componentes que comprenden cada uno de los módulos ya establecidos, donde los componentes del

subsistema *Servidor* pueden ser observados gráficamente en la

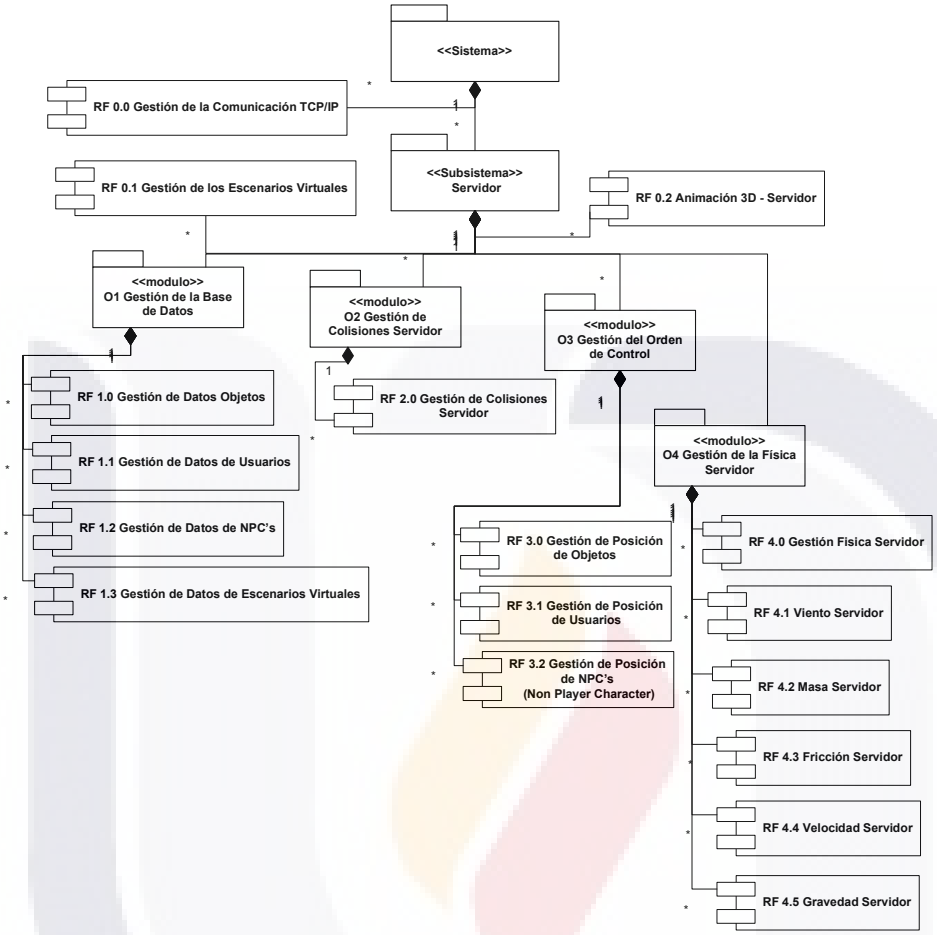


Figura 18.

1. *Módulo gestion de la base de datos*: Contiene aquellos componentes que proporcionan las funcionalidades para el manejo de la base de datos: Gestión de datos de objetos, gestión de datos de usuarios, gestión de datos de NPC's y gestión de datos de escenarios virtuales.

2. *Módulo de gestión de colisiones Servidor*: Aquí se incluyen los componentes que administran las colisiones en el lado del servidor: Gestión de colisiones servidor.

3. *Módulo de gestión de orden de control*: Contiene los componentes que proporcionan la funcionalidad para llevar la administración de la posición: Gestión

de posición de objetos, gestión de posición de usuarios, gestión de posición de NPC's.

4. *Módulo de la física Servidor*: Contiene diversos componentes con la funcionalidad para aplicar las leyes de la física a un escenario virtual: Gestión física servidor, viento servidor, masa servidor, fricción servidor, velocidad servidor y gravedad servidor.

Los componentes del subsistema *Cliente* pueden ser apreciados gráficamente en la Figura 19.

1. *Módulo de gestión de E/S de datos*: Contiene aquellos componentes que proporcionan la funcionalidad para la administración de entrada y salida de datos: Gestión de E/S de datos y gestión de dispositivos semi-inmersivos.

2. *Módulo de gestión de colisiones Cliente*: Aquí se incluyen los componentes que administran las colisiones en el lado del cliente: Gestión de colisiones cliente.

3. *Módulo de la física Cliente*: Contiene diversos componentes con la funcionalidad para aplicar las leyes de la física al usuario dentro del escenario virtual: Gestión física cliente, viento cliente, masa cliente, fricción cliente, velocidad cliente y gravedad cliente.

Las descripciones específicas de cada módulo y componente son explicadas en la Tabla 4.

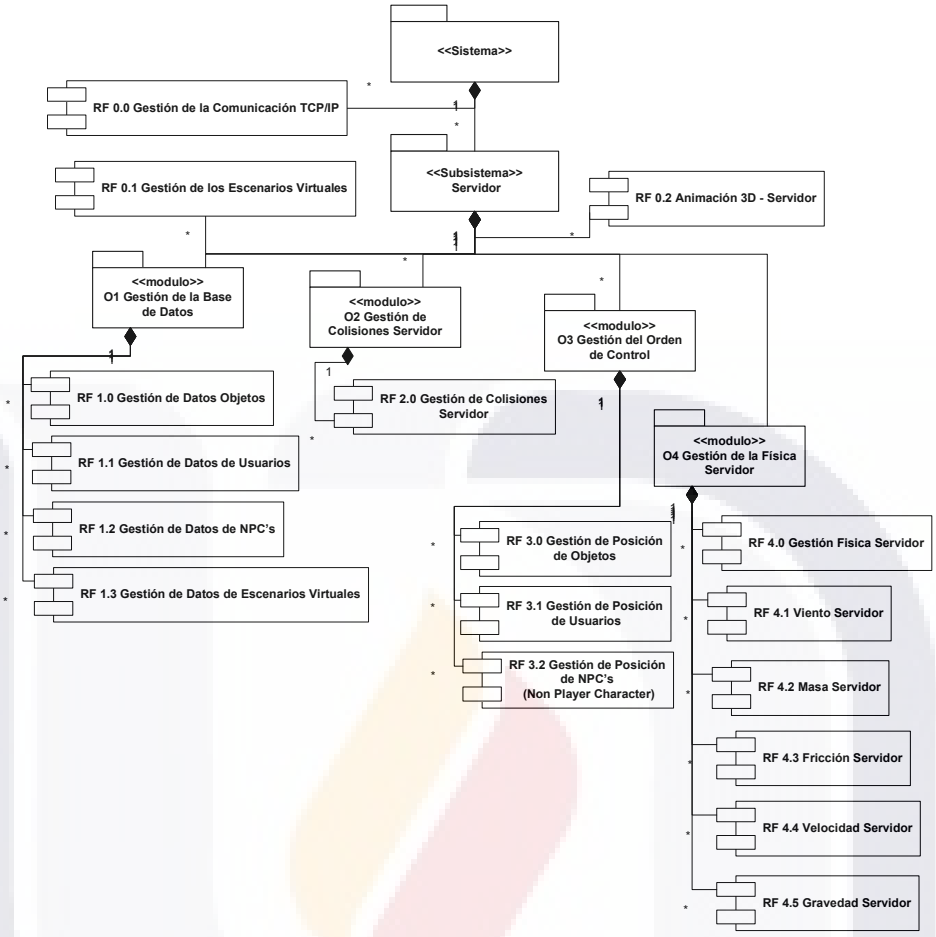


Figura 18. Diagrama de Componentes - Servidor

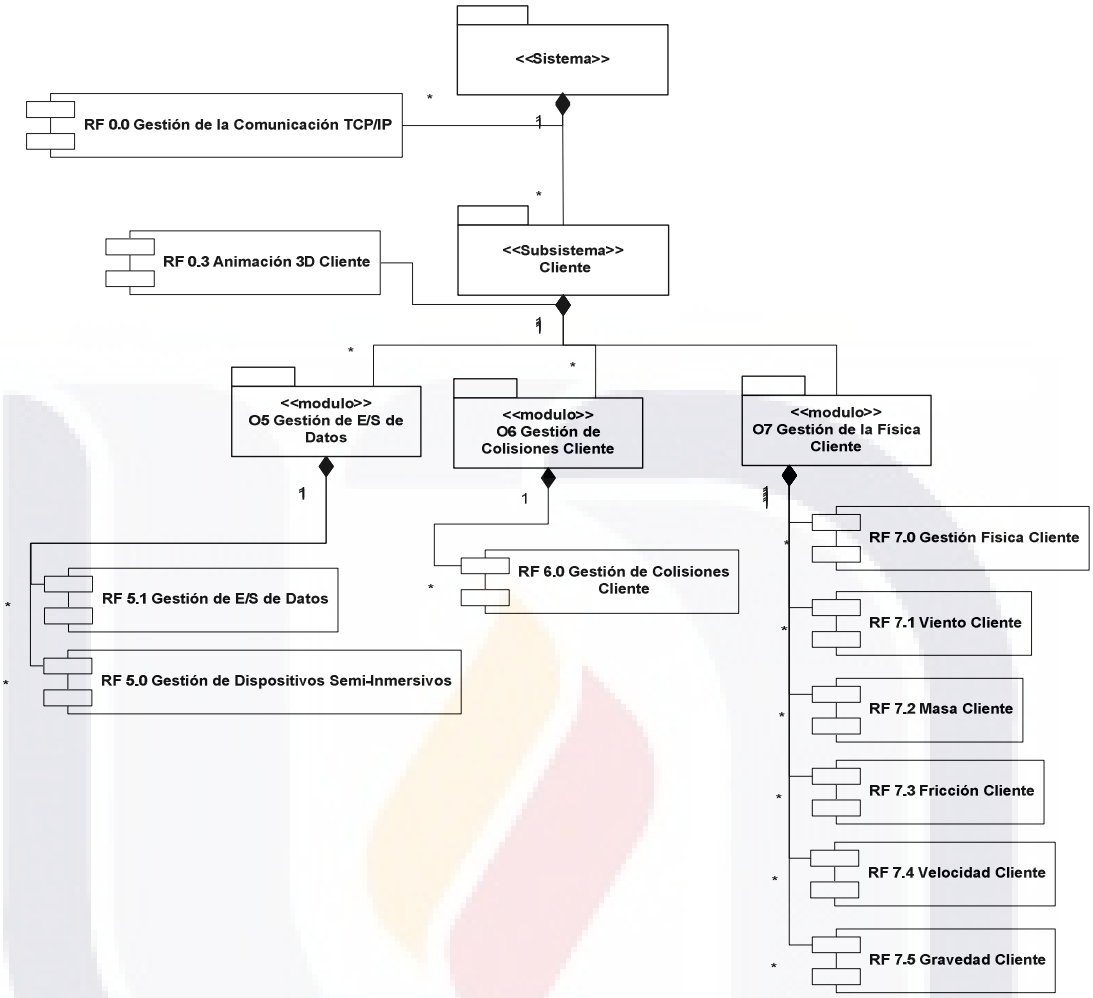


Figura 19. Diagrama de Componentes - Cliente

ID	Modulo/Componente	Responsabilidad
RF 0.0	Gestión de la Comunicación TCP/IP	Contiene la funcionalidad necesaria para comunicar los diferentes subsistemas a través de la red.
RF 0.1	Gestión de los Escenarios Virtuales	Contiene la funcionalidad necesaria para la creación y la administración de un escenario virtual.
RF 0.2	Animación 3D - Servidor	Contiene la funcionalidad necesaria para determinar cómo debe ser visualmente el ambiente virtual representado.
RF 0.3	Animación 3D - Cliente	Contiene la funcionalidad necesaria para crear visualmente el ambiente virtual, así como todo lo que contiene (usuarios, objetos y NPC's).
O1	Gestión de la Base de	Modulo que se encarga de almacenar y organizar los

	Datos	diferentes tipos de datos.
O2	Gestión de Colisiones Servidor	Modulo que se encarga de detectar cuando hay una colisión, así como la acción derivada de esta.
O3	Gestión del Orden de Control	Modulo encargado de gestionar la posición de los usuarios, objetos y NPC's dentro del escenario virtual.
O4	Gestión de la Física Servidor	Modulo encargado de determinar las fuerzas físicas que influyen en un escenario virtual.
O5	Gestión de E/S de Datos	Modulo encargado de administrar los datos introducidos por el usuario.
O6	Gestión de Colisiones Cliente	Modulo que se encarga de detectar cuando hay una colisión, así como la acción derivada de esta.
O7	Gestión de la Física Cliente	Modulo de encargado de aplicar las fuerzas físicas en un escenario virtual.
RF 1.0	Gestión de Datos de Objetos	Contiene la funcionalidad necesaria para la administración de los datos particulares de los objetos virtuales.
RF 1.1	Gestión de Datos de Usuarios	Contiene la funcionalidad necesaria para la administración de los datos particulares de los usuarios.
RF 1.2	Gestión de Datos de NPC's	Contiene la funcionalidad necesaria para la administración de los datos particulares de los NPC's.
RF 1.3	Gestión de Datos de Escenarios Virtuales	Contiene la funcionalidad necesaria para la administración de los datos particulares de los escenarios virtuales.
RF 2.0	Gestión de Colisiones Servidor	Contiene la funcionalidad necesaria para determinar la existencia de una colisión.
RF 3.0	Gestión de Posición de Objetos	Contiene la funcionalidad necesaria para determinar tanto la posición como la orientación de los objetos virtuales dentro del escenario virtual.
RF 3.1	Gestión de Posición de Usuarios	Contiene la funcionalidad necesaria para determinar tanto la posición como la orientación de los usuarios dentro del escenario virtual.
RF 3.2	Gestión de Posición de NPC's	Contiene la funcionalidad necesaria para determinar tanto la posición como la orientación de los NPC's dentro del escenario virtual.
RF 4.0	Gestión Física Servidor	Contiene la funcionalidad necesaria para coordinar la física aplicada en el escenario virtual.
RF 4.1	Viento Servidor	Contiene la funcionalidad necesaria para establecer la

		velocidad y dirección del viento.
RF 4.2	Masa Servidor	Contiene la funcionalidad necesaria para establecer la masa que tienen los cuerpos dentro de un escenario virtual.
RF 4.3	Fricción Servidor	Contiene la funcionalidad necesaria para establecer la fricción que tienen los cuerpos dentro de un escenario virtual.
RF 4.4	Velocidad Servidor	Contiene la funcionalidad necesaria para establecer la velocidad a la que se desplazan los cuerpos dentro de un escenario virtual.
RF 4.5	Gravedad Servidor	Contiene la funcionalidad necesaria para establecer la fuerza de gravedad dentro de escenario virtual.
RF 5.0	Gestión de Dispositivos Semi-Inmersivos	Contiene la funcionalidad necesaria para leer y mostrar datos a través de dispositivos semi-inmersivos.
RF 5.1	Gestión de E/S de Datos	Contiene la funcionalidad necesaria para leer y mostrar datos a través de dispositivos comunes de E/S.
RF 6.0	Gestión de Colisiones Cliente	Contiene la funcionalidad necesaria para determinar la existencia de una colisión.
RF 7.0	Gestión Física Cliente	Contiene la funcionalidad necesaria para aplicar la física en el cliente.
RF 7.1	Viento Cliente	Contiene la funcionalidad necesaria para aplicar la velocidad y dirección del viento.
RF 7.2	Masa Cliente	Contiene la funcionalidad necesaria para aplicar la masa que tienen los cuerpos dentro de un escenario virtual.
RF 7.3	Fricción Cliente	Contiene la funcionalidad necesaria para aplicar la fricción que tienen los cuerpos dentro de un escenario virtual.
RF 7.4	Velocidad Cliente	Contiene la funcionalidad necesaria para aplicar la velocidad a la que se desplazan los cuerpos dentro de un escenario virtual.
RF 7.5	Gravedad Cliente	Contiene la funcionalidad necesaria para aplicar la fuerza de gravedad dentro de escenario virtual.

Tabla 4. Diagrama de Responsabilidades

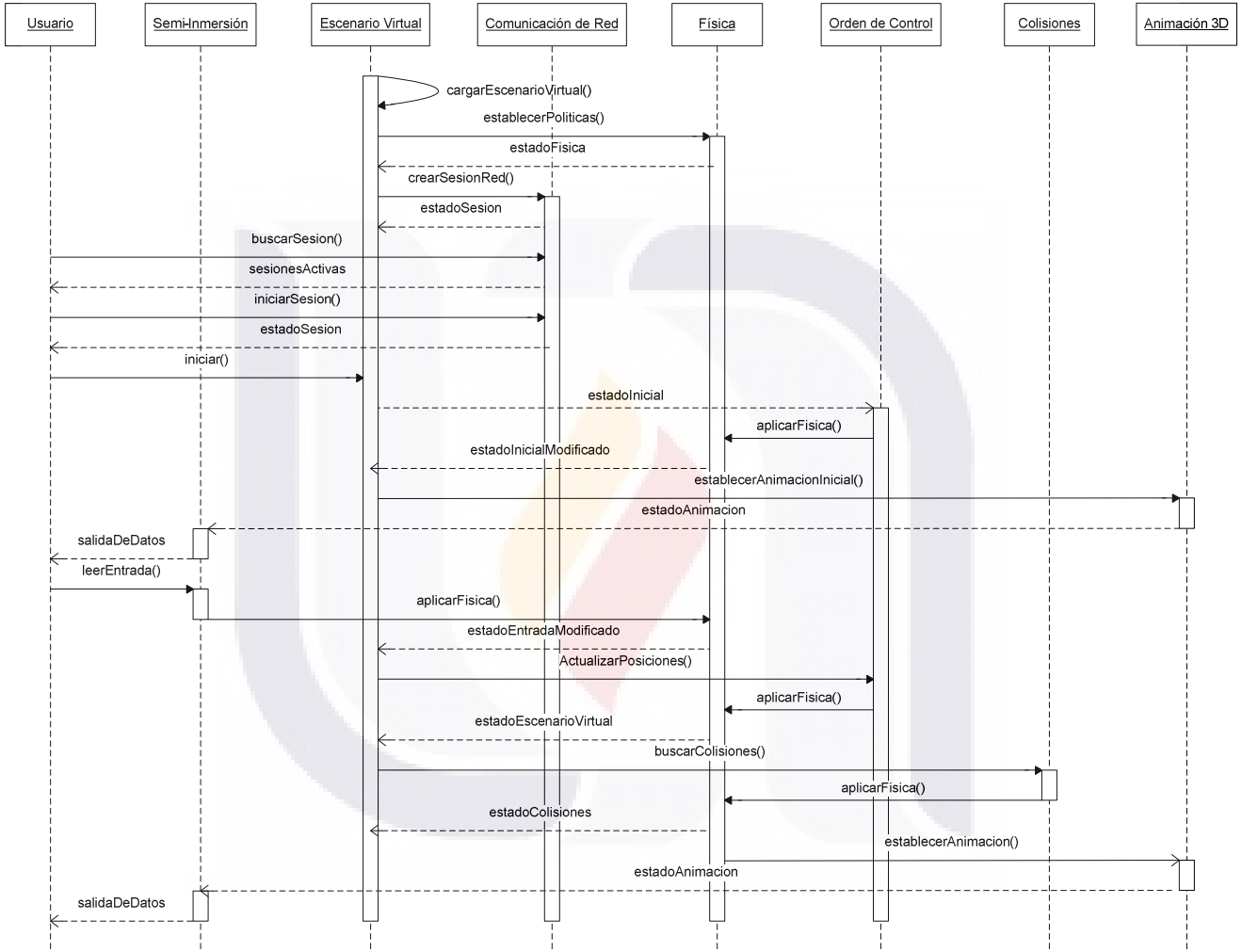


Figura 20. Diagrama de Secuencia

La Figura 20 muestra tanto el tipo de mensajes y el momento en que estos enviados son entre los componentes ya establecidos. En seguida se listan las tareas y el orden en que son ejecutadas:

1. El proceso inicia cuando el modulo de escenario virtual carga un escenario virtual.
2. Se establecen las políticas de las leyes de la física que se aplicarán.
3. Se crea una sesión de red en el escenario virtual para que los usuarios remotos puedan acceder al escenario virtual.
4. El usuario busca sesiones de red activas para acceder al sistema virtual.
5. Una vez encontrada una sesión de red el usuario ingresa al sistema virtual.
6. Una vez que el usuario ha accedido, el sistema virtual lo introduce dentro del escenario virtual.
7. El escenario virtual envía el estado inicial del usuario al modulo de orden de control.
8. Se aplican las leyes de la física al personaje recién introducido al mundo virtual.
9. El escenario virtual informa al modulo de animación 3D sobre las acciones de movimiento iniciales del personaje virtual, el cual muestra la representación inicial al usuario del mundo virtual.
10. El usuario introduce por medio de dispositivos de hardware semi-inmersivos nuevas acciones para realizar dentro del mundo virtual.
11. Se aplican las leyes de la física a las acciones realizadas por el usuario.
12. El modulo de orden de control actualiza las nuevas posiciones de los entes virtuales.
13. Se buscan y detectan colisiones.
14. Se aplican las leyes de la física a cada colisión encontrada.
15. Se determinan las nuevas acciones realizadas por los entes virtuales.
16. El modulo de animación de 3D nuevamente hace un proceso de render para representar gráficamente el escenario virtual.
17. se repite nuevamente todo desde el paso 10.

La anterior vista permite conocer cómo es la interacción en un momento dado dentro de un sistema virtual semi-inmersivo distribuido.

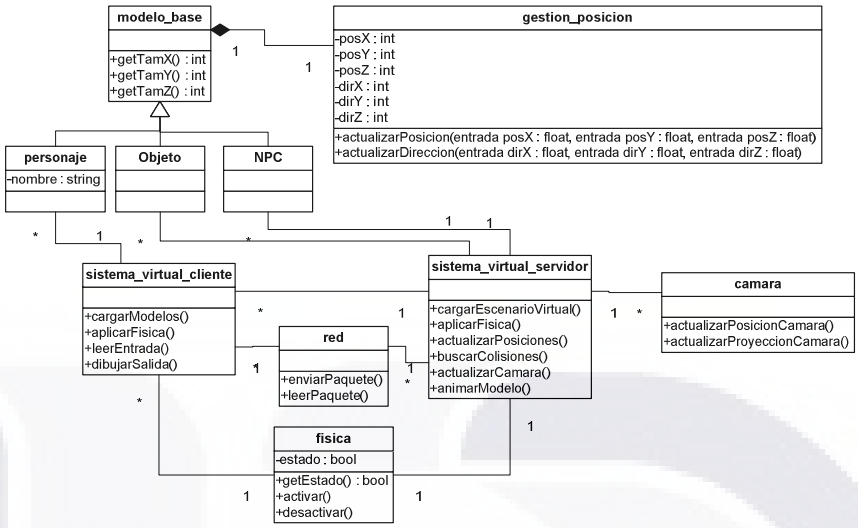


Figura 21. Diagrama de Clases

La Figura 21 muestra las principales clases con las que cuenta un sistema virtual semi-inmersivo distribuido, al igual que las relaciones que hay entre ellas, las cuáles se listan y describen a continuación:

- *modelo_base*: Clase encargada de permitir la creación de un ente virtual.
- *gestión_posición*: Clase que compone a *modelo_base* encargada de gestionar la posición de un ente virtual.
- *personaje*: Clase que hereda de *modelo_base* encargada de crear un personaje virtual para el usuario.
- *objeto*: Clase que hereda de *modelo_base* encargada de crear un objeto virtual.
- *NPC*: Clase que hereda de *modelo_base* encargada de crear un NPC.
- *sistema_virtual_cliente*: Clase encargada de interactuar con el usuario y proporcionarle la información al servidor.
- *sistema_virtual_servidor*: Clase encargada de administrar el escenario virtual y los cambios que hay dentro del mundo virtual.
- *camara*: Clase encargada de determinar la perspectiva de visión que tendrá cada usuario.

- *red*: Clase encargada de crear un canal de comunicación entre el servidor y los múltiples usuarios remotos.
- *fisica*: Clase encargada de aplicar las leyes de la física.

Como paso siguiente se definieron los archivos necesarios para la comunicación entre los diferentes equipos físicos de un sistema distribuido que implemente un sistema virtual semi-inmersivo, esto se aprecia en la Figura 22, en la cual es posible observar que existen tres grupos principales: *servidor*, *cliente* y *base de datos*. Dentro de estos grupos los módulos crean diferentes tipos de archivos para el paso de mensajes que existen entre ellos.

Implementar esta estructura de datos muestra su conveniencia al tener un sistema de realidad virtual distribuido entre múltiples equipos físicos, ya que de esta manera solo viajan a través de la red aquellos paquetes con información realmente importante para el funcionamiento del sistema, reduciendo tanto la carga de red como la cantidad de paquetes recibidos a procesar en el servidor.

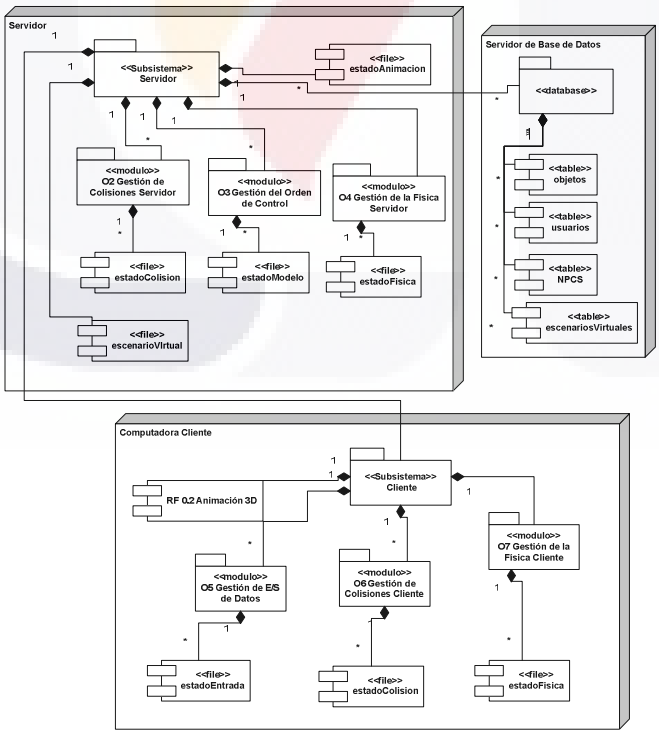


Figura 22. Diagrama de Grupo de Datos

Como punto final se describe en los subcapítulos siguientes el proceso llevado a cabo para transformar una arquitectura semi-inmersiva para sistemas virtuales distribuidos implementando "game engines" a una arquitectura multi-hilo y semi-inmersiva para sistemas virtuales distribuidos implementando "game engines".

4.2. Árbol de Paralelización de Tareas

Tulip [12] menciona que para crear una arquitectura multi-hilo es necesario encontrar las tareas que serán paralelizadas dentro del modelo arquitectónico, es por ello que para cumplir con los objetivos de este estudio se ha limitado a las tareas que afectan directamente la interacción del usuario con el escenario virtual, formando lo que se llama árbol de paralelización de tareas (Figura 23). Este árbol se divide en tres campos principales de tareas: *usuario, red y control.*

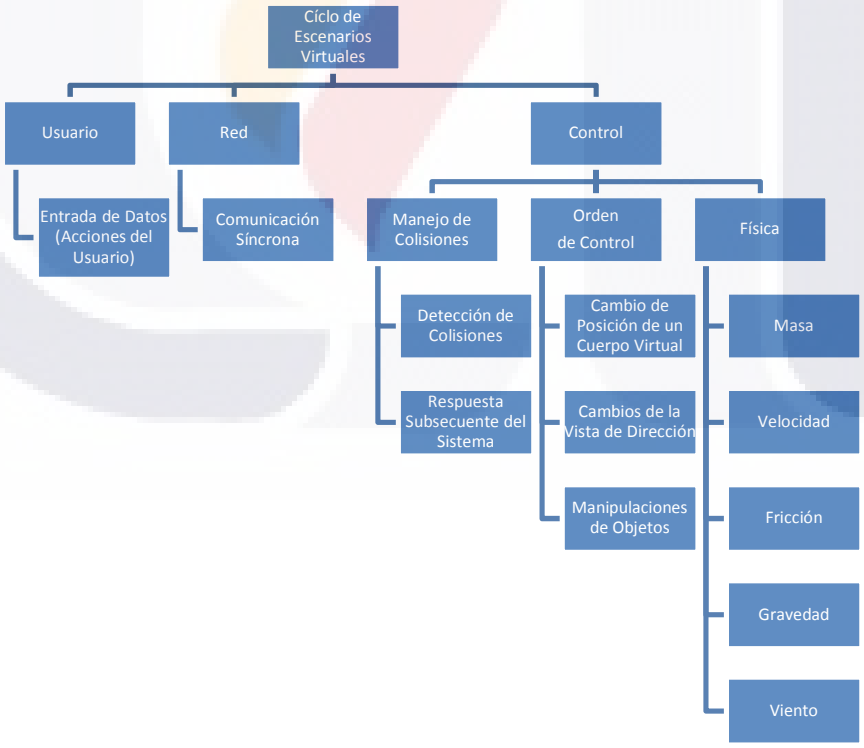


Figura 23. Árbol de Paralelización de Tareas

1. El campo *usuario* contempla a su vez las *entradas de datos*, las cuales pueden ser también interpretadas como las acciones que el usuario realiza dentro del escenario virtual.
2. El campo *red* abarca solo la administración de paquetes de red que existen entre el servidor y el cliente.
3. El campo de *control* considera tres aspectos importantes, los cuales son: *manejo de colisiones*, *orden de control* y *física*.

El primero de estos tiene dos importantes tareas, detectar colisiones y determinar una respuesta subsecuente a esta colisión. El segundo aspecto tiene como objetivo gestionar tanto el cambio de posición de un ente virtual (usuario u objeto) como el de dirección, además de la manipulación que ejerce el usuario sobre un objeto virtual. El último aspecto contempla las reglas generales de la física que afectan a las personas y los objetos en la vida real, como lo son la masa, velocidad, fricción, gravedad y la resistencia al viento.

4.3. Algoritmo para el Balanceo de Carga

Una vez que se han establecido las tareas a paralelizar, es necesario determinar un algoritmo que realice esta tarea, de tal forma que la carga de trabajo quede distribuida casi equitativamente entre los múltiples núcleos o procesadores que tenga la computadora.

Debido a que la revisión bibliográfica no presentó un algoritmo apropiado para la investigación, se propuso uno que consiste en asignar el paquete a procesar al núcleo o procesador con menos carga de trabajo. En seguida se muestra el pseudocódigo utilizado:

Donde:

- T = Paquete a Procesar.
- entero :: min ( ) = regresa el núcleo o procesador que tiene menos carga de trabajo
- asignar (entero, Paquete) = asigna un paquete a un núcleo o procesador.
- Paquete :: sigPaquete ( ) = obtiene el siguiente paquete a Procesar

```
Balanceo Carga
{
        minimo : entero
        hacer mientras ( T ≠ null )
        {
                T ← sigPaquete ( )
                minimo ← min ()
                asignar ( minimo, T )
        }
}
```

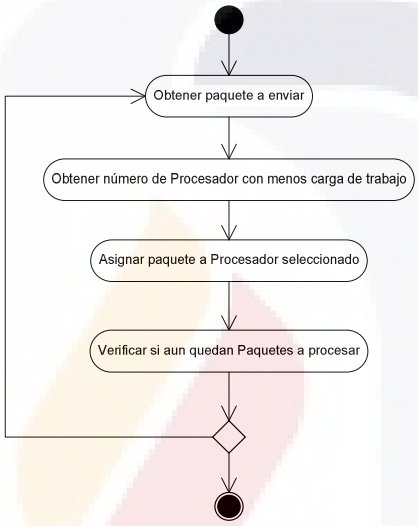


Figura 24. Diagrama de Actividades

La Figura 24 muestra las actividades realizadas por el anterior algoritmo:

1. Mientras existan paquetes a procesar se realizará lo siguiente:
    a. *Obtener el siguiente paquete a procesar*: En esta actividad se obtiene un paquete generado por el árbol de paralelización de tareas y se asigna a la cola de tareas.
    b. *Obtener procesador o núcleo con la mínima carga de trabajo*: En esta actividad se obtiene ya sea el procesador o el núcleo (dependiendo del hardware sobre el cual el sistema es ejecutado) con la mínima carga de trabajo proporcionada por los paquetes que tiene asignados y ejecutándose.

c. *Asignar el paquete a procesar al procesador o núcleo con la mínima carga de trabajo*: En esta actividad se asigna el paquete en cola al procesador obtenido en la actividad anterior para su ejecución inmediata y balancear la carga de trabajo entre todos los núcleos o procesadores.
d. *Verificar si aún quedan paquetes a procesar*: En esta actividad se determina si aún existen paquetes del árbol de paralelización de tareas que deban ser procesados de forma paralela.

El anterior algoritmo a pesar de la simplicidad que tiene ofrece como ventaja el distribuir casi equitativamente los paquetes a procesar, permitiendo tener un mayor rendimiento sobre el sistema virtual.

4.4. Aplicación del Algoritmo de Balanceo de Carga sobre el Árbol de Paralelización de Tareas

Ya establecidos tanto las tareas a paralelizar como el algoritmo para la distribución de cargas, se procede a la directa aplicación sobre el código. Para ello primero se hace una relación de los procesos que serán paralelizados con un identificador para facilitar su identificación (Tabla 5).

ID	TIPO DE PROCESO
1	Entrada de Datos (Acciones del Usuario)
2	Comunicación Síncrona
3	Detección de Colisiones
4	Respuesta Subsecuente del Sistema
5	Cambio de Posición de un Cuerpo Virtual
6	Cambios de la Vista de Dirección
7	Manipulaciones de Objetos
8	Masa
9	Velocidad

10	Fricción
11	Gravedad
12	Viento

Tabla 5. Procesos a Paralelizar

Como ejemplo de la aplicación del algoritmo de balanceo de carga se presentan dos diferentes simulaciones con dos y cuatro procesadores respectivamente, donde cada simulación posee una cantidad aleatoria de paquetes.

Simulación con 2 Procesadores

PROCESO	TIPO	TAMAÑO	NO. PROCESADOR
1	11	27	0
2	9	13	1
3	9	49	1
4	11	54	0
5	10	59	1
6	4	96	0
7	5	97	1
8	11	77	0
9	5	59	1
10	7	30	0
11	1	55	1
12	4	95	0
13	6	43	1
14	8	5	1
15	10	61	0
16	5	73	1

Total Procesador 0: 440
Total Procesador 1: 453

Simulación con 4 Procesadores

PROCESO	TIPO	TAMAÑO	NO. PROCESADOR
1	11	20	0
2	9	86	1
3	9	87	2
4	11	68	3
5	10	99	0
6	4	5	1
7	5	93	3
8	11	59	2
9	5	27	1
10	7	40	1
11	1	25	0
12	4	5	0
13	6	12	2
14	8	5	0
15	10	16	0
16	5	31	1
17	5	99	2
18	2	34	0
19	7	90	3
20	9	3	1

Total Procesador 0: 204
Total Procesador 1: 192
Total Procesador 2: 257
Total Procesador 3: 259

La simulación muestra que se asigna una serie de paquetes consecutivos a un diferente procesador, el cual es determinado por tener la menor carga de trabajo en ese momento. Al final se observa la carga final casi equitativa de cada procesador.

5. RESULTADOS Y LIMITACIONES

En este capítulo se presentan los resultados obtenidos, así como también las limitaciones encontradas en la realización del estudio de investigación.

5.1. XNA

Se escogió este Game Engine para el desarrollo del prototipo de un sistema 3D semi-inmersivo por la alta flexibilidad y la gran cantidad de librerías que contiene, así como la fácil importación de librerías externas, como fue necesario para este proyecto importar librerías para el uso de dispositivos semi-inmersivos. Es por ello que es necesario explicar primeramente lo que el XNA es, ya que es la herramienta principal con la que se desarrollo el sistema 3D semi-inmersivo.

XNA (Figura 25) es una colección de librerías creadas por Microsoft para el desarrollo de videojuegos para las plataformas Xbox 360, Windows y Zone. Su desarrollo está basado en .NET Framework 2.0, y tiene como ventaja proveer un manejo optimizado para la ejecución de videojuegos [43].

Figura 25. Diferentes Capas de XNA [43]

En otras palabras XNA es una plataforma de desarrollo de videojuegos sobre DirectX, la cual integra funcionalidades específicas que facilitan el desarrollo de los mismos. La Figura 26 muestra los componentes que integran el XNA, donde los recuadros verdes representan el contenido del mismo, los recuadros morados indican lo que el programador tiene que agregar, y el recuadro azul los componentes externos desarrollados por terceras personas.

Figura 26. Contenido de XNA [43]

5.2. Implementación de XNA

Es debido a la alta flexibilidad y a la alta cantidad de librerías que tiene el XNA que se optó como herramienta base para el desarrollo del prototipo de software. Su implementación consistió en la utilización de las librerías que contiene dentro núcleo del framework por medio de métodos de fácil implementación. Lo anterior permitió utilizar a placer las librerías para el desarrollo del modelo propuesto en la Figura 16 sin intervenir en el flujo del proceso de la arquitectura.

5.3. Evaluación

Como se menciona en el capítulo 1 de este documento, la evaluación se realizó por medio del desarrollo de un sistema de realidad virtual implementando los estándares establecidos por la arquitectura propuesta en esta investigación. Esta evaluación se realiza en dos partes, en la construcción del sistema basado en el modelo arquitectónico, y en la evaluación de conformidad del sistema con el modelo.

5.3.1. Proyecto Piloto

Como se menciona anteriormente fue necesaria la construcción de un sistema 3D semi-inmersivo, en donde se tomó como herramienta principal de desarrollo el XNA, ya que además de facilitar el desarrollo de aplicaciones, también permite establecer el cómo funciona, a diferencia de la gran mayoría de los "game engines", que solo proporcionan las librerías, impidiendo indicar el flujo de trabajo.

El prototipo desarrollado consiste en la simulación de múltiples naves espaciales desplazándose entre asteroides. Cada nave es controlada por un usuario remoto diferente, el cual interactúa con el sistema por medio de un dispositivo semi-inmersivo. Este dispositivo se compone de unos lentes y un sensor infrarrojo, lo cual hace posible cambiar la perspectiva de visión del usuario dependiendo de su posición física real, transformando el monitor del usuario en una ventana hacia el mundo virtual.

5.4. Construcción del Sistema Basado en el Modelo Arquitectónico

Para el desarrollo del sistema 3D semi-inmersivo se tomó como base la arquitectura presentada en este documento, para ello se tomaron las diferentes vistas genéricas que contiene y se adecuaron para cumplir las especificaciones del sistema 3D semi-inmersivo.

Como primer cambio se optó por tomar una arquitectura cliente-servidor (Figura 27), en lugar de una multi-servidor, esto debido a que la cantidad de usuarios conectados simultáneamente era demasiado pequeña, por lo que un solo servidor responde perfectamente a las solicitudes generadas por los clientes.

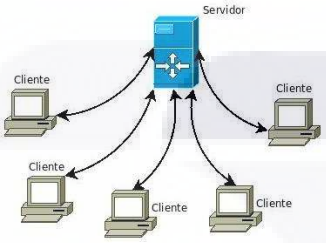


Figura 27. Arquitectura Cliente – Servidor para el Prototipo

El siguiente cambio que se generó fue tomar el diagrama de módulos de la arquitectura propuesta y eliminar el módulo de física, debido a la alta complejidad que requiere su implementación a nivel de codificación, así como también el módulo de base de datos, ya que para el sistema actual no es requerido (

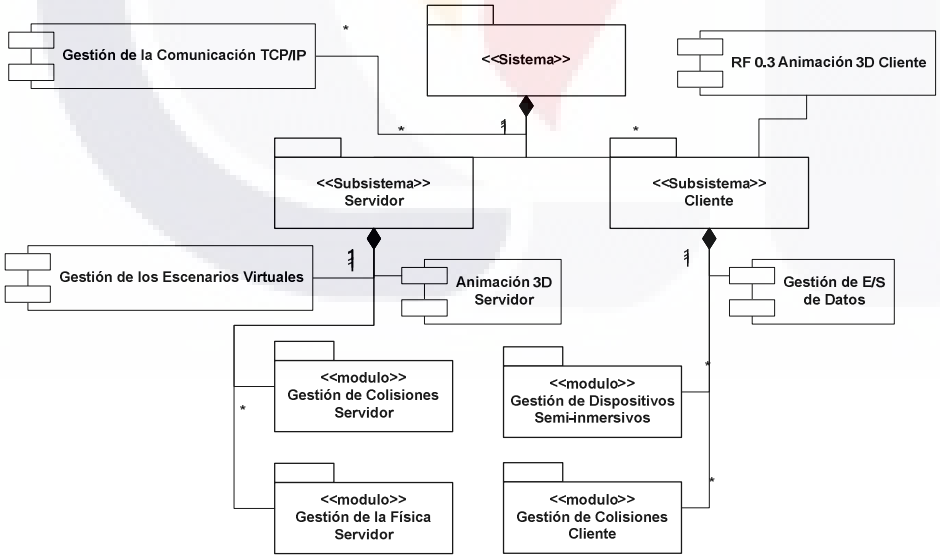


Figura 28). Lo que demuestra que la arquitectura de estudio no deja ser solo un guía de apoyo en el desarrollo de sistemas 3D semi-inmersivos.

Al igual que el diagrama de la arquitectura el sistema principal se divide en 2 subsistemas: servidor y cliente. De igual manera Se utiliza un componente con las funciones necesarias para llevar a cabo la comunicación entre los subsistemas.

1. *Servidor*: Este subsistema contiene un componente para la gestión de los escenarios virtuales y otro para la animación en la parte del servidor. Además cuenta con 2 módulos para diversas gestiones: orden de control y colisiones.
2. *Cliente*: Este subsistema cuenta con un componente para llevar a cabo la animación en el cliente, cuenta también con 2 módulos para las gestiones de: animación 3D y colisiones.

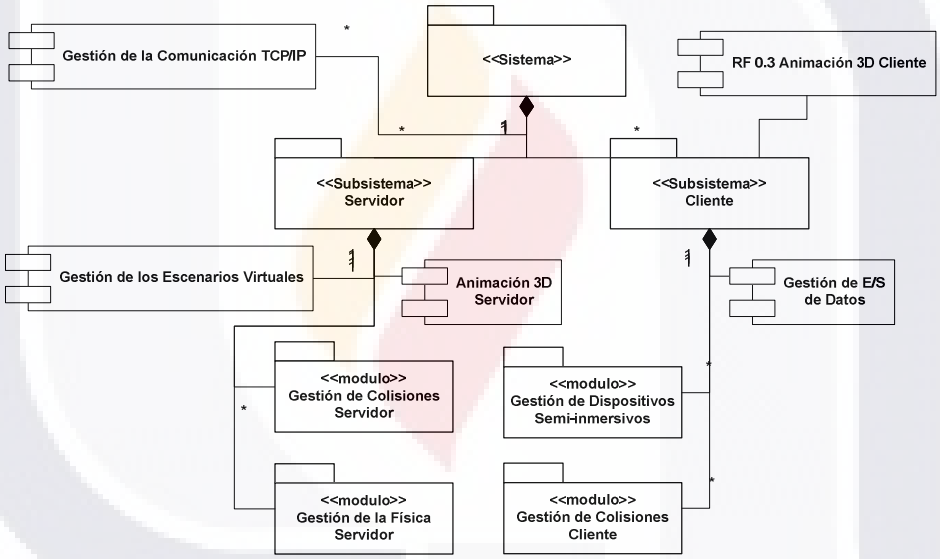


Figura 28. Diagrama de Módulos para el Prototipo

En seguida se muestra el diagrama de clases diseñado para el prototipo desarrollado en esta investigación (Figura 29).

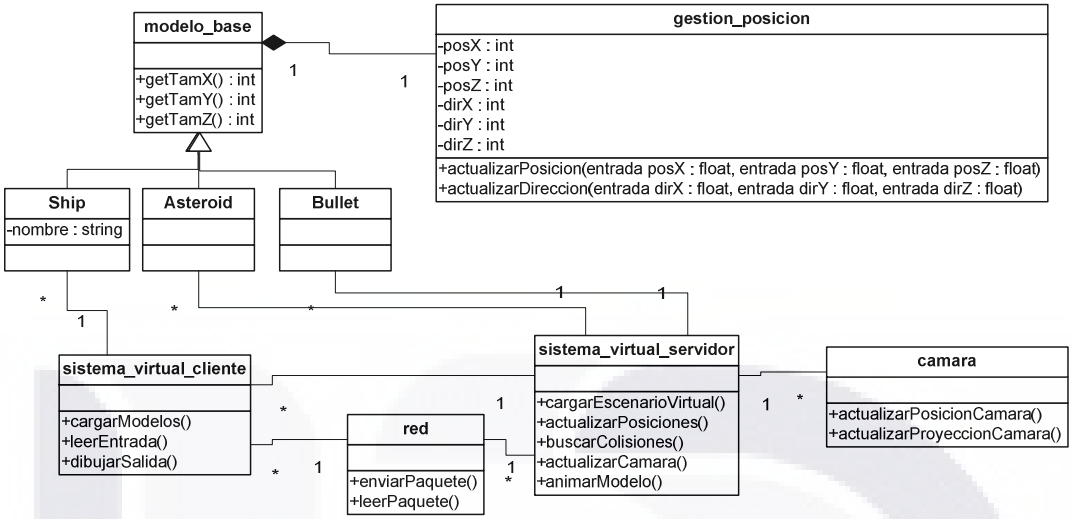


Figura 29. Diagrama de Clases del Prototipo

Como es posible observar en los diagramas anteriores, existe un gran parecido al modelo genérico que se presenta en el anterior capítulo, por lo que es posible afirmar que la arquitectura de esta investigación funciona de manera apropiada como una guía en el desarrollo de sistemas 3D semi-inmersivos.

La Figura 30 muestra una pantalla del prototipo desarrollado, en la cual es posible apreciar el uso de coordenadas del dispositivo semi-inmersivo (lentes mencionados anteriormente) para el manejo de la perspectiva de visión que tendrá el usuario.

También es posible observar la carga de trabajo que hay en los núcleos del procesador del CPU, en donde de acuerdo a la implementación del algoritmo muli-hilo la carga de trabajo en ambos procesadores es casi equitativa.

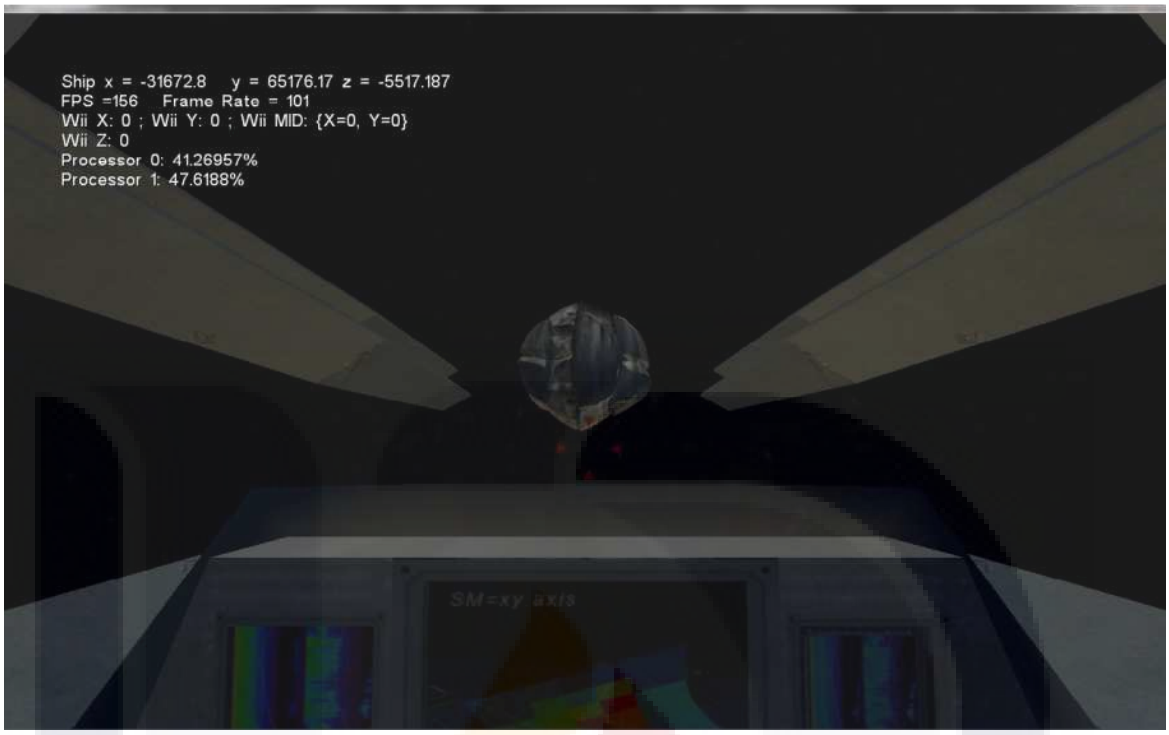


Figura 30. Prototipo de Sistema 3D Semi-Inmersivo

5.5. Evaluación de Conformidad del Sistema con el Modelo

Para realizar esta evaluación, se tomó nuevamente como guía la metodología MEDARISH [41], la cual nos proporciona una serie de pasos a realizar para comprobar que el sistema desarrollado este conforme a los estándares del modelo arquitectónico. En la Figura 31 es posible observar los pasos mencionados anteriormente, los cuales consisten en comparar la arquitectura del sistema contra la arquitectura del modelo, procediendo a la obtención de diferencias, determinando por último si se encuentran en un rango valido o no.

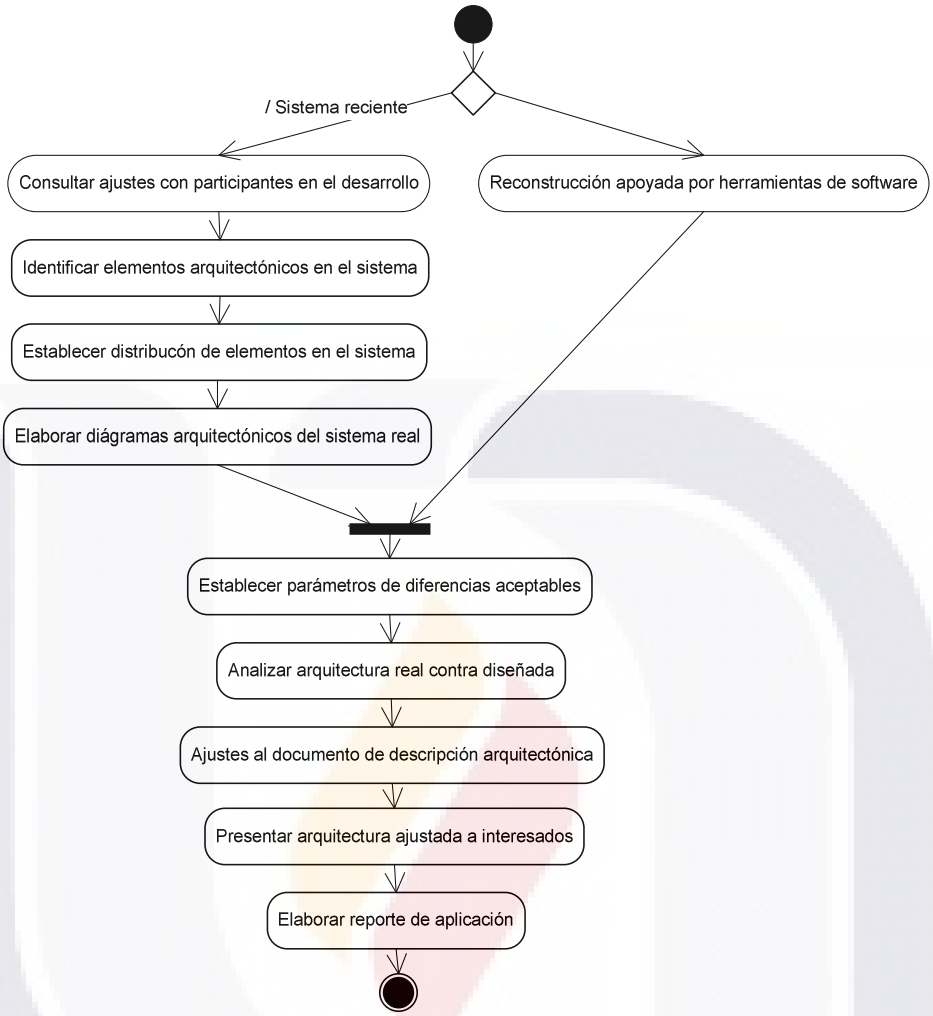


Figura 31. Evaluación del Sistema Contra el Modelo Arquitectónico

Tomando como base lo establecido por MEDARISH es posible afirmar que el sistema desarrollado cumple con los estándares establecidos, ya que como se muestra en la sección anterior (subcapítulo 5.3), al comparar ambas arquitecturas es posible apreciar que en la arquitectura del sistema solo se procedió a la eliminación de algunos módulos. Otro punto importante a analizar es el uso de algoritmos multi-hilo, los cuales son posibles de observar en la Figura 30, donde se muestra la carga de los 2 núcleos de un procesador bastante similares.

5.6. Limitaciones

Las limitaciones encontradas en esta investigación son las siguientes:

- Esta investigación a pesar de estar enfocada al desarrollo de sistemas de 3D semi-inmersivos distribuidos por medio de “game engines”, sigue siendo bastante general.
- El modelo arquitectónico es una guía para el desarrollo de sistemas 3D, sin embargo se requieren conocimientos previos técnicos sobre el contexto de “game engines” y realidad virtual para la utilización de la misma, ya que de otra manera no podrá explotarse debidamente.
- No se enfoca a nivel de codificación, por lo que una mala implementación de los algoritmos multi-hilo puede llevar al decremento del rendimiento del CPU debido a la gran cantidad de hilos existentes.

6. CONCLUSIONES

6.1. Objetivos Alcanzados

Se desarrollo de manera satisfactoria una arquitectura de software que implementa las tecnologías de los "game engines" para permitir la creación de sistemas virtuales donde los usuarios remotos puedan interactuar de forma simultánea y semi-inmersiva en los escenarios 3D.

De lo anterior es posible afirmar que el objetivo principal de investigación se ha cumplido satisfactoriamente, de igual manera al observar detalladamente las diferentes vistas de la arquitectura desarrollada en esta investigación es correcto afirmar el cumplimiento de los siguientes objetivos específicos:

- Es posible caracterizar un game engine, que tome en cuenta algoritmos de paralelización.
- Es posible establecer un marco de trabajo que contemple incorporar nuevas tecnologías de hardware, algoritmos y nuevos elementos para evolucionar y mejorar los sistemas virtuales distribuidos.
- Es posible establecer un marco de trabajo que permita la interacción simultánea de los múltiples usuarios remotos.
- Es posible establecer un marco de trabajo que contemple incorporar diferentes tecnologías semi-inmersivas.

Con los resultados obtenidos en los capítulos anteriores es posible aseverar que un modelo arquitectónico de software simplifica el desarrollo de programas de software que simulen escenarios virtuales en 3D en donde varios usuarios podrán interactuar simultáneamente y de forma semi-inmersiva con los objetos que les son presentados por estos sistemas.

Además de lo anterior también es correcto ratificar que:

- Los algoritmos multi-hilo implementados en una arquitectura de game engine incrementan el desempeño para ejecutar algunas tareas, ya que un proceso grande ejecutado por un solo núcleo en un determinado tiempo puede ser procesado en un tiempo menor si es dividido en múltiples procesos pequeños, lo que permite ejecutar una mayor cantidad de paquetes y procesos de forma más eficiente..
- Los sistemas distribuidos permiten incrementar el número de usuarios interconectados simultáneamente sin que el sistema se colapse, esto debido a que las diferentes funciones del sistema se distribuyen entre los distintos equipos físicos interconectados en un momento dado.
- El uso de dispositivos semi-inmersivos incrementa la sensación de realidad en un sistema virtual, esto se debe a que con cada dispositivo utilizado es más fácil percibir el mundo virtual e interactuar dentro de este como si fuese el real.

6.2. Conclusiones de los Métodos de Investigación Empleados

Los métodos de investigación conceptual empleados en este proyecto han probado ser herramientas bastante apropiadas para llevar a cabo el desarrollo de una arquitectura bastante compleja que abre nuevas áreas de estudio dentro de la ingeniería de software, como complementa algunas ya existentes.

El uso de MEDARISH resulto ser una guía bastante apropiada para el desarrollo de una arquitectura de software especializada, así como el uso de UML facilitó plasmar las diferentes vistas de la arquitectura en una representación común.

6.3. Conclusiones de Aprendizaje Personal

El proceso de aprendizaje personal que sigue un proceso de maestría desarrolla aptitudes no solo metodológicas, sino también personales, que permiten crear trabajos de investigación así como también aplicar el conocimiento fuera del campo del campo de la investigación, ya que las diferentes habilidades del investigador también actúan en la vida cotidiana permitiendo ser una mejor persona, algunas de estas habilidades son las de ser observador, buscar diferentes soluciones a un problema en concreto, ser disciplinado, entre otras.

Una aportación obtenida durante el estudio del posgrado, fue la de descubrir la gran gama de conocimientos sin investigar aún, permitiéndome ser un pionero en campos muy particulares del saber, proveyendo nuevas tecnologías que en un futuro no muy lejano mejoraran la calidad de vida.

6.4. Contribuciones de Investigación

El trabajo de esta investigación es considerado que contribuye al área de ingeniería de software, enfocado al nivel conceptual de diseño de sistemas por medio de arquitecturas de software, aplicado al campo de la realidad virtual

La aportación de esta investigación se realiza por medio de la creación de una arquitectura de software que implementa algoritmos multi-hilo y tecnologías de “game engines” para permitir el desarrollo de sistemas virtuales distribuidos que utilicen dispositivos de hardware semi-inmersivos para un ámbito de realidad virtual.

La arquitectura cubre el proceso completo desde el punto de vista conceptual para el desarrollo de sistemas virtuales distribuidos semi-inmersivos, proporcionando diferentes vistas de software basadas en UML para facilitar el tener una amplia gama de visión con respecto al desarrollo de un sistema virtual distribuido semi-inmersivo.

Otra de las características de la arquitectura es la facilidad de adaptación que posee para desarrollar un sistema con características especiales y ajenas a la arquitectura, ya que permite la integración de nuevos módulos, así como la eliminación de aquellos que no son requeridos en un momento dado.

Otra contribución generada por esta investigación es el *algoritmo de balanceo de carga*, el cual no solo puede ser aplicado en el contexto de esta investigación, sino que además puede ser utilizado en cualquier sistema que requiera distribuir una gran carga de trabajo entre los múltiples núcleos o procesadores de una computadora.

6.5. Trabajo a Futuro

La arquitectura propuesta ha probado ser funcional para el desarrollo de sistemas virtuales distribuidos y semi-inmersivos, sin embargo aun falta crear una arquitectura de referencia que permita crear diferentes arquitecturas especializadas.

Otro punto importante es la realización de pruebas tanto con un número excedente de usuarios como con diferentes sistemas utilizando diferentes dispositivos semi-inmersivos para la mejora de la arquitectura de esta investigación.

Otro aspecto necesario a mejorar es la administración de los hilos, ya que la manera implementada en esta investigación a pesar de ser factible puede llegar a tener grandes problemas de rendimiento si no se tiene un cuidado con la prioridad asignada a estos durante su ejecución.

Por último se piensa realizar un game engine que caracterice dentro de sí la arquitectura generada en esta investigación, de tal manera que sea posible generar diferentes sistemas virtuales distribuidos y semi-inmersivos de forma más simple y más apropiada.

7. GLOSARIO

Ambiente virtual: es el escenario que rodea al usuario y a los objetos virtuales. El ambiente virtual tiene atributos que lo definen y puede tener comportamiento. Entre los atributos que puede tener un ambiente virtual están las luces y los sonidos [26].

Comportamiento: es un conjunto de reacciones de un objeto que actúa en respuesta a un estímulo procedente de su medio externo, y es observable objetivamente [26].

Escena: es la imagen de un Mundo Virtual que el usuario visualiza en un momento dado [26].

Gesto: está definido por la posición y orientación, de la mano y los dedos del usuario, en un momento determinado. El usuario interactúa con el mundo virtual a través de gestos.

Inmersión: puede definirse como la presentación de pistas sensoriales que convencen perceptivamente a los usuarios de que ellos están rodeados por el ambiente generado por computadora. Para elevar la sensación de inmersión del usuario dentro del mundo virtual, se deben representar fielmente comportamientos físicos de los objetos como la gravedad y las colisiones entre los objetos [26].

Dispositivos de entrada: Son dispositivos que permiten la interacción entre el humano y el Mundo Virtual. Entre estos dispositivos periféricos se puede mencionar el guante de datos, joystick y sistemas de reconocimiento de voz [26].

Dispositivos hápticos: Son dispositivos de entrada y salida que pueden medir la posición y fuerza de la mano del usuario y otras partes del cuerpo cuando se esté manipulando un ambiente virtual [26].

Dispositivos visuales: presentan a los ojos del usuario el mundo 3D generado por la computadora [26].

Hardware gráfico y de cómputo: Los sistemas gráficos y de cómputo se refieren al hardware usado para controlar la operación completa del ambiente virtual [26].

Herramientas de software: Algunas herramientas de software para el desarrollo de aplicaciones de Realidad Virtual son librerías, sistemas de aplicaciones ó ambientes para desarrollo completo, integrando cada aspecto de la creación de una aplicación de RV (modelación, codificación y ejecución) en un paquete [27].

Mundo Fantástico: Es el que nos permite realizar tareas irreales, como volar o atravesar paredes. Es el típico entorno que visualizamos en los videojuegos, pero también proporcionan situaciones interesantes para aplicaciones serias, como puede ser observar un edificio volando a su alrededor o introducirnos dentro de un volcán [27].

Mundo Muerto: Es aquel en el que no hay objetos en movimiento ni partes interactivas, por lo cual solo se permite su exploración. Suele ser el que vemos en las animaciones tradicionales, en las cuales las imágenes están pre calculadas y producen una experiencia pasiva [27].

Mundo Real: Es aquel en el cual los elementos tienen sus atributos reales, de tal manera que si miramos un reloj, marca la hora. Si pulsamos las teclas de una calculadora, se visualizaran las operaciones que esta realiza y así sucesivamente [27].

Mundo Virtual: está compuesto por el ambiente virtual y todos los objetos virtuales contenidos dentro de él (el Mundo Virtual vacío tiene un Ambiente Virtual por defecto a pesar de no contener objetos) [26].

Navegar: se dice que el usuario navega dentro del mundo virtual cuando cambia su posición y/o orientación dentro de este [26].

Objeto virtual: es un modelo abstracto de un objeto real que tiene atributos que lo definen y puede tener comportamiento propio. Un objeto virtual puede tener asociado luces y sonidos como parte de sus atributos. El objeto virtual está definido por una geometría generalmente asociada a un conjunto de polígonos [26].

Renderización: Proceso llevado a cabo por una computadora para generar una imagen que el usuario pueda apreciar de forma automática a partir de un modelo tridimensional existente.

Sistemas de rastreo: dispositivos que proveen información sobre la posición y orientación de un objeto[26].

Sistema de realidad virtual: es un conjunto de dispositivos de hardware y software que ubican al participante en un ambiente generado por computadora que aparenta ser real. Este cuenta con una interfaz entre la computadora y, los sistemas perceptivos y musculares del usuario.

Sistemas de sonido: Dispositivos usados para la generación de sonido 3D (sonidos localizados) dentro del mundo virtual. Los sonidos localizados pueden ser asociados a objetos o pueden ser usados para mejorar la sensación de inmersión en el ambiente [26].

8. BIBLIOGRAFÍA

[1] Bishop, Lars, et al. **Designing a PC Game Engine.** Computer Graphics in Entertainament 1998.

[2] Anderson, Eike F., et al. ***The Case for Research in Game Engine Architecture.*** FuturePlay 2008.

[3] Mateevitsi, Victor, et al. ***A Game-Engine Based Virtual Museum Authoring and Presentation System.*** 3rd International Conference on Digital Interactive Media Entertainment and Arts.

[4] Wünsch, Burkhard C., et al. ***A Framework for Game Engine Based in Visualisations.***

[5] Zhang, Xiayou y Gracanin, Denis. ***Streaming Web Services For 3D Portal Applications***. ACM, 2008.

[6] Schou, Torben y Gardner, Henry. ***A Wii Remote, a Game Engine, Five Sensor Bars and a Virtual Reality Theatre.*** ACM.

[7] Smith, J. David y Graham, T. C. Nicholas. ***Use of Eye Movements for Video Game Control***. ACE 06, June ACM.

[8] Gallo, Luigi, et al. ***3D Interaction with Volumetric Medical Data: Experiencing the Wiimote***. Ambi-sys 2008. ACM.

[9] ***Game Engines***.

http://en.wikipedia.org/wiki/Game_engine. Consulta 02/03/2009

[10] Stang, Bendik. ***Game Engines Features and possibilities***. IMM DTU, 2003.

[11] Caltagirone, Sergio, et al. ***Architecture for a Massively Multilplayer Online Role Playing Game Engine***. CCSC: Northwestern Conference. December 2002.

[12] Tulip, James, et al. ***Multi-Threaded Game Engine Desing***. ACM

[13] Curiel, Mariela. ***Sistemas Distribuidos***.

http://www.ldc.usb.ve/~mcuriel/Cursos/sop3/Tema1.pdf. Consulta 06/03/2009

[14] Yonezawa, Akinori y Hewitt, Carl. ***Modeling Distributed Systems***.

[15] ***Computación Distribuida***.

http://es.wikipedia.org/wiki/Computaci%C3%B3n_distribuida. Consulta 06/03/2009

[16] Roberts, D. J., et al. ***A Real-time, Predictive Architecture for Distributed Virtual Reality***

[17] Rhalib, Abdennour El y Merabti, Madjid. ***Agents-Based Modeling for a Peer-to-Peer MMOG Architecture.*** ACM Computers in Entertainment, Vol. 3, No. 2, April 2005, Article 3B.

[18] Ng, Beatrice, et al. ***A Multi-Server Architecture for Distributed Virtual Walkthrough.*** ACM VRST'02, November 11–13, 2002, Hong Kong.

[19] Coulouris, George, et al. **Sistemas Distribuidos Conceptos y Diseño.** Addison Wesley.

[20] Media, Charles River. ***Game Programming Gems 6*.** Thomson

[21] Wang, Weihua, et al. ***SmartCU3D: a Collaborative Virtual Environment System with Behavior Based Interaction Management.*** ACM VRST' 01, November 15-17, 2001, Banff, Alberta, Canada.

[22] Roussou, Maria. ***Immersive Interactive Virtual Reality in the Museum*.** Athenas, Grecia

[23] Steuer, Jonathan. ***Defining Virtual Reality: Dimensions Determining Telepresence***. Journal of Communication. 1993.

[24] Thalmann, Nadia, et al. ***Virtual Reality Software and Technology***.

[25] Brooks, Frederick. ***What's Real About Virtual Reality?***. IEEE, 1999.

[26] Matte, Alfredo. ***Una Herramienta para Generar Mundos Virtuales Inmersivos***. 2001.

[27] ***Realidad Virtual***

Disponible en: http://html.rincondelvago.com/realidad-virtual_1.html Consulta *17/12/2010*

[28] Cory, Clark A., et al. ***3D Computer Animated Walkthroughs for Architecture, Engineering, and Contruction Applications.*** International Conference Graphicon. Nizhny Novgorod, Russia. 2001.

[29] Igarashi, Takeo, et al. ***Path Drawing for 3D Walkthrough.***

[30] Joao Bernardes, et al. ***Integrating the Wii Controller with enJine: 3D Interfaces Extending the Frontiers of a Didactic Game***. ACM Comput. Entertain. February 2009.

[31] Hevner, A., et al. ***Design Science in Information System Research.*** MIS Quaterly Vol. 28 No. 1, USA, 2004.

[32] Mora, Manuel. ***Descripción del Método de Investigación Conceptual, versión 2.*** Reporte Interno, UAA, Aguascalientes, México, 2004.

[33] Mora, Manuel, et al. ***The case for Conceptual Research in Information Systems.*** G. Grant & T. Felix (Eds), Electronic Proceedings of the 2008 International Conference on Information Resources Management (Conf-IRM). Ontario, Canada. 2008.

[34] Jahn, Gonzalo Vélez. ***Museos Virtuales – Presente y Futuro.*** Primera Conferencia Venezolana sobre Aplicación de Computadoras en Arquitectura, Caracas, 1999.

[35] Anthes, Christoph, et al. ***An Adaptive Network Architecture for Close-Coupled Collaboration in Distributed Virtual Environments***, GUP Linz. Austria, 2004.

[36] Pimental, Ken y Teixeira, Kevin. ***Virtual Reality: Through the New Looking Glass***. Intel – Mc Graw Hill, Usa, 1993.

[37] De la flor, Mike. ***The Carrara Studio 3 Handbook.*** Charles River Media, 2004.

[38] ***Modelado 3D.***

http://abc.mitreum.net/wp-content/uploads/clase2-parte1-teoria.pdf. Consulta *29/03/2010*

[39] ***Fundamentos Básicos de Modelado 3D.***

http://www.cristalab.com/tutoriales/fundamentos-basicos-de-modelado-3d-c148l. Consulta *29/03/2010*

[40] ***Gráficos 3D por Computadora.***

http://es.wikipedia.org/wiki/Gr%C3%A1ficos_3D_por_computadora. Consulta *29/03/2010*

[41] Muñoz López, Juan. ***Metodología para Diseñar Modelos Arquitectónicos de Referencia que Integran Sistemas Heredados***. Universidad Autónoma de Aguascalientes, 2008.

[42] Rozanski y Woods. ***Software Systems Architecture***. Addison Wesley, Pearson Education Inc., U. S. A., 2005.

[43] **¿Qué es XNA?**

http://aprendiendoxna.wordpress.com/primeros-pasos/%C2%BFque-es-xna/ Consulta *01/08/2010*