Universidad de las Ciencias Informáticas

Facultad 5

"PERSONALIZACIÓN DE ESCENARIOS DE APRENDIZAJE 3D PARA LABORATORIOS VIRTUALES"

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor: Daiver Gé Ramírez

Tutor: Lic. Luis Gabriel Viciedo Caraballoso

La Habana, Junio 2011
"Año 53 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas, a hacer uso del mismo en su beneficio.

Para que así conste firmamos el presente a los ____ días del mes de _____del año _____.

Autor:

Daiver Gé Ramírez

_____

Tutor:

Luis Gabriel Viciedo Caraballoso

_____

DATOS DE CONTACTO

Tutor:

Nombre y Apellidos: Luis Gabriel Viciedo Caraballoso.

Institución: Universidad de las Ciencias Informáticas.

Título: Licenciado en Educación, especialidad Física.

Categoría Docente**:** Profesor Auxiliar.

E-mail: viciedo@uci.cu

Se desempeña como Profesor Auxiliar desde el año 2000. Es Diseñador del Aprendizaje en el proyecto Laboratorios Virtuales del Centro de Informática Industrial (CEDIN) y encargado de la capacitación y tutoría de los estudiantes que están en el ciclo básico y profesional de la carrera que pertenecen al proyecto mencionado. Ha realizado tutorías en más de 15 trabajos de diploma en la carrera de Ingeniería Informática y se ha desempeñado en tribunales de eventos científicos y de cambios de categoría docente en la UCI. Tiene más de 20 publicaciones sobre el tema de Laboratorios Virtuales y Ambientes de Aprendizaje Interactivos desde el año 2008.

Agradecimientos

A mi mamá, que siente todo lo que hago y me da fuerzas para continuar.

A Isidro, por confiar en mí y darme apoyo en todo momento.

A mi hermano, por ayudarme todos estos años de mi carrera.

A mi abuelita, por alentarme y apoyarme durante todo este tiempo.

A mi familia, por la preocupación y el apoyo durante estos cinco años.

A mis amigos, por estar a mi lado y convertirse en mi familia en la UCI.

A mi tutor, por su ayuda e interés para realizar este trabajo.

A Alexey, por su ayuda para llevar a cabo este trabajo.

DEDICATORIA

A mis padres, mi hermano y mi familia por estar siempre orgullosos de mí.

Especialmente a mi abuelita querida CHACHI.

Daiver

RESUMEN

La Realidad Virtual ha alcanzado gran auge en la actualidad, convirtiéndose en un importante campo de desarrollo e investigación, especialmente en el área de la educación, donde consiste en un recurso didáctico del que los profesores se pueden servir para motivar y atraer la atención de los estudiantes a través de los gráficos tridimensionales de calidad y de alto grado de interactividad ofrecida por los sistemas virtuales.

En el proyecto PROLAVI de la Facultad 5, se desarrollan laboratorios virtuales con fines educativos donde es necesario un mecanismo para realizar cambios de diseño en estos escenarios virtuales que no comprometan la aplicación principal desarrollada por el proyecto y sean realizados de manera rápida y simple.

En este trabajo se propone dar solución a ese problema. Para ello se hace una investigación sobre los temas relacionados con los laboratorios virtuales, lenguajes script que pueden ser introducidos en las aplicaciones para un mejor desempeño y la compilación en tiempo de ejecución, de los que se explican sus características y conceptos fundamentales. Finalmente se demuestra que el uso de lenguajes script en las aplicaciones es de gran ayuda para su funcionamiento, aún más cuando se trata de configuraciones y cambios de diseño.

PALABRAS CLAVE

Escenarios virtuales, lenguajes script, Realidad Virtual,.

ÍNDICE DE CONTENIDO

INTRODUCCIÓN

La Informática es una de las ciencias que mayor impacto ha tenido en el desarrollo económico y social a nivel mundial desde finales del siglo pasado hasta el presente. Su evolución ha sido vertiginosa desde su nacimiento, permitiendo el surgimiento de diferentes áreas dentro de la misma, que son el centro de atención de los científicos, especialistas y del mercado.

Una de estas áreas es la Realidad Virtual, donde las industrias de juegos y simulación han ocupado los planos vanguardistas de desarrollo e investigación.

En los últimos años el desarrollo de la Realidad Virtual en el mundo ha sido muy amplio, esto puede observarse en diferentes ramas como, los videojuegos, la medicina, en el ámbito militar y en el área educativa.

La Realidad Virtual es una tecnología especialmente adecuada para la enseñanza, debido a su facilidad para captar la atención de los estudiantes mediante su inmersión en mundos virtuales relacionados con las diferentes ramas del saber, lo cual puede ayudar en el aprendizaje de los contenidos de cualquier materia.

Cuba, centrando su esfuerzo en el desarrollo de la industria del software, ha comenzado a dar sus primeros pasos en esta área. Con el objetivo de insertar a Cuba en el mercado del software a nivel mundial y para la informatización del país se crea en el año 2002 la Universidad de las Ciencias Informáticas (UCI), la cual desde sus inicios tiene como objetivo revolucionar la industria del software en Cuba, alcanzando hoy en día notables logros en el ámbito internacional.

La UCI, en aras de convertir la Informática en una de las ramas más productivas de la nación cubana y asumir con orgullo y placer el reto de informatización de la sociedad cubana, se ha convertido en un centro científico donde se vincula la docencia con el proceso productivo, en el desarrollo de varios perfiles informáticos.

Es por eso que en la Facultad 5 de la Universidad de las Ciencias Informáticas (UCI) surge el proyecto Laboratorios Virtuales con el propósito de desarrollar escenarios de aprendizaje en 3D, mediante la creación de prácticas de laboratorios virtuales para algunas asignaturas relacionadas con el mundo de la informática, como el Ensamblaje de Computadoras, Diseño e Instalación de una red local (LAN) y Administración y Configuración de una red local (LAN).

Un laboratorio virtual es un sistema computacional que pretende aproximar el ambiente de un laboratorio tradicional. Los experimentos se realizan paso a paso, siguiendo un procedimiento similar al de un laboratorio tradicional: se visualizan instrumentos y fenómenos mediante objetos dinámicos, imágenes o animaciones. Se obtienen resultados numéricos y gráficos, tratándose

éstos matemáticamente para la obtención de los objetivos perseguidos en la planificación docente de las asignaturas. (1)

Actualmente el proyecto se encuentra en la elaboración de tres laboratorios virtuales donde se desea personalizar los escenarios de aprendizajes de los mismos mediante la utilización de lenguajes de extensión, previendo que en la fase de despliegue ocurran problemas tales como, si se desea hacer algún cambios en el diseño de cualquiera de los objetos esto implica tiempo extra en los cronogramas de desarrollo de la aplicación informática. Los cambios de diseño comprometen la integridad de la aplicación principal desarrollada. Los laboratorios virtuales son una herramienta de auto aprendizaje donde los alumnos alteran las variables de entradas y configuran nuevos experimentos, por lo que estos escenarios se encuentran en constantes cambios y la inserción de nuevos componentes en la aplicación se hace desde la aplicación principal, lo cual compromete sus funcionalidades.

El **problema científico** a resolver es cómo ofrecer una solución informática para la realización de cambios en los escenarios de aprendizaje de los laboratorios virtuales, que no comprometan la integridad de la solución principal desarrollada.

Por lo tanto constituye **objeto de estudio** de este tema la Integración de soluciones informáticas con lenguajes de extensión.

Para centralizar aún más el trabajo se realiza el siguiente **campo de acción:** Configuración de escenarios de aprendizaje con lenguajes de extensión.

Una vez definido lo antes expuesto se plantea el siguiente **objetivo general**: Realizar cambios en los escenarios de aprendizaje en laboratorios virtuales sin comprometer la aplicación principal.

Para el total cumplimiento del mismo se trazan las siguientes tareas de investigación:

- Establecer el estado del arte sobre la utilización de programación con lenguajes de extensión en el área de los videojuegos.
- Caracterizar la importancia de los lenguajes de extensión y justificar la selección final del lenguaje con el cual desarrollar el procedimiento informático que se propone.
- Caracterizar los formatos de los escenarios de aprendizaje de los laboratorios Virtuales.
- Dominar las especificaciones técnicas de los laboratorios virtuales que se desarrollan.
- Identificar las herramientas necesarias para la personalización de los escenarios de aprendizaje con el lenguaje script seleccionado.

- Diseñar y graficar los cambios a ejecutar en los escenarios de aprendizaje de la aplicaciones desarrolladas por el proyecto.
- Implementación de las técnicas estudiadas que muestren como dar solución a algunos de los problemas encontrados.
- Establecer las limitaciones de la solución informática desarrollada.

Durante la investigación se propone el uso **de métodos teóricos y empíricos** siguientes:

Teóricos:

- Análisis histórico lógico: Es imprescindible el estudio de la evolución internacional comprobada en el empleo del videojuego con fines de aprendizaje, sus riesgos y situación actual.
- Análisis y síntesis de los fundamentos teóricos, didácticos y metodológicos de la práctica de estudio y el laboratorio como tipos de clase en la educación superior cubana.

Empíricos:

- La observación de los tipos de videojuegos enfocados al aprendizaje y los ambientes de aprendizaje que se pueden conformar con estas tecnologías.
- Entrevistas a estudiantes con el rol de programadores dentro del proyecto para indagar sobre las necesidades del proyecto.

El presente documento está estructurado de la siguiente manera: resumen, introducción, tres capítulos de contenido, conclusiones, recomendaciones, referencias bibliográficas, bibliografía, glosario de términos y anexos.

En el Capítulo 1: *Fundamentación Teórica,* se hace un estudio donde se investiga acerca de los Ambientes de aprendizaje interactivos, sus características, ventajas y desventajas. Se hace un amplio recorrido por los diferentes lenguajes de extensión que pueden ser aplicados a estos Ambientes de aprendizaje para su mejor funcionamiento.

En el Capítulo 2: *Soluciones técnicas,* se proponen las soluciones técnicas para la creación del Demo representativo con el lenguaje script seleccionado, así como las funcionalidades que debe tener el mismo.

En el Capítulo 3: *Descripción de la propuesta,* se tratan los temas relacionados con la solución final del Demo y su aplicación al proyecto PROLAVI.

CAPÍTULO I "FUNDAMENTACIÓN TEÓRICA"

1.1 Ambientes de aprendizaje interactivos

En la actualidad existen un gran número de Ambientes de aprendizaje interactivos que han sido situados dentro del ámbito académico para el servicio de la sociedad, los cuales han sido de gran utilidad por los conocimientos que aportan.

Un **ambiente de aprendizaje interactivo** es un espacio electrónico, donde se simulan en 2D o 3D situaciones problemáticas provenientes de un diseño de aprendizaje, donde deben cumplirse objetivos instructivos y educativos de un programa de estudio, en que estudiantes y tutores colaboran en escenarios simulados y cumplen tareas experimentales o de entrenamiento, asumiendo un rol propio de su profesión y donde son evaluados por ello.

Muchas aplicaciones multimedia, elaboradas con disímiles tecnologías informáticas, pueden considerarse como ambientes de aprendizajes interactivos. En nuestro caso, estamos considerando solo aquellas aplicaciones informáticas que asumen estrategias y experiencias provenientes de la industria de los videojuegos y de la Realidad Virtual, dadas las potencialidades de aplicación en el área del aprendizaje en contextos educativos. Como ejemplos de estas aplicaciones, podemos mencionar: simuladores de vuelos y de conducción, entrenadores médicos, programas de visualización científica, instrumentación virtual, los juegos serios y laboratorios virtuales, donde la principal forma de interacción ocurre en escenarios con objetos simulados en 2D o 3D.

1.2 Laboratorios Virtuales

Para el estudio y desarrollo de los laboratorios virtuales se han dividido en tres grupos fundamentales teniendo en cuenta sus características fundamentales:

- Laboratorios virtuales software: Están desarrollados como un programa independiente y para ser ejecutados en los ordenadores, su servicio no requiere de un servidor Web.
- Laboratorios virtuales Web: Este tipo de laboratorios se basa en un software que depende de los recursos de un servidor determinado.
- Laboratorios remotos: Estos laboratorios requieren de equipos servidores específicos que les den acceso a las maquinas a operar de forma remota, y no pueden ofrecer su funcionalidad de forma local.

En la Universidad de las Ciencias informática debido a que estos sistemas informáticos están encaminados específicamente al aprendizaje de los estudiantes, adoptan la siguiente definición de Laboratorio Virtual:

"Son escenarios de aprendizaje electrónico concebido para la colaboración y la experimentación dentro de un proceso de enseñanza-aprendizaje, con objeto de desarrollar habilidades, investigación o realizar otras actividades creativas. Igualmente permite elaborar, consultar y difundir resultados mediante tecnologías de información y comunicaciones". (2)

1.2.1 Ventajas y desventajas de los Laboratorios Virtuales

A partir de la definición anterior, se pueden enunciar algunas ventajas que presentan los Laboratorios Virtuales:

- Son más económicos, siendo una alternativa barata y eficiente donde los estudiantes pueden simular y observar fenómenos como si manipularan objetos reales.
- Acerca y facilita a un mayor número de alumnos a la realización de experimentos, aunque el alumno y el experimento no coincidan en el espacio.
- Es una herramienta de auto aprendizaje, donde el alumno altera las variables de entrada, configura nuevos experimentos, aprende el manejo de instrumentos y personaliza el experimento.
- Los estudiantes aprenden mediante prueba y error, sin miedo a sufrir o provocar un accidente, sin avergonzarse de realizar varias veces la misma práctica, ya que pueden repetirlas sin límite; sin temor a dañar alguna herramienta o equipo.

Estos Laboratorios Virtuales no solo presentan ventajas, también presentan inconvenientes que a continuación se muestran algunos de los más destacados.

- El Laboratorio Virtual no puede sustituir la experiencia práctica, aunque realice una simulación altamente parecida a la realidad se debe considerar una herramienta complementaria para aumentar el rendimiento de los estudiantes.
- En el Laboratorio Virtual se corre el riesgo de que el alumno se comporte como un mero espectador por lo que es necesario que se realicen las actividades de forma organizada y progresiva para alcanzar los objetivos planteados de la práctica realizada.
- El alumno no utiliza elementos reales en los Laboratorios Virtuales, lo que provoca una pérdida parcial de la visión de la realidad. Además, no siempre se dispone de la simulación adecuada para el tema que el profesor desea trabajar.

1.3 Laboratorios Remotos

La creciente complejidad de las actividades prácticas de los laboratorios y el desarrollo de las TIC y la computación, han hecho que los laboratorios virtuales evolucionen, transformándose en Laboratorios Remotos. Estos son sistemas basados en instrumentación real del laboratorio (no prácticas simuladas), que permiten al estudiante realizar actividades prácticas de forma local o remota, transfiriendo información entre el proceso y el estudiante de manera uni o bidireccional. El alumno utiliza y controla los recursos disponibles en el laboratorio, a través de estaciones de trabajo de una red local (intranet) o bien a través de internet. (1)

1.3.1 Ventajas de los Laboratorios Remotos

Algunas ventajas que presentan los Laboratorios Remotos:

- Permite aprovechar los recursos, tanto humanos como materiales de los Laboratorios Tradicionales.
- El Laboratorio Remoto amplía la oferta horaria del alumno en su formación. Son un recurso extremadamente rentable en la formación.
- El alumno no necesita disponer del software de simulación.
- Los Laboratorios Remotos ofrecen la posibilidad de controlar de forma remota las aplicaciones basadas en instrumentos virtuales, donde destacan la modularidad y el carácter abierto de los objetos dinámicos de implementación.
- No todo son ventajas, también existen inconvenientes. A continuación se muestran las más destacadas.
- La experimentación en tiempo real exige períodos de muestreo relativamente pequeños, requiriendo el uso de recursos que por lo general, resultan costosos, además de la necesidad de disponer de sistemas operativos de tiempo real.
- Todas las actuaciones sobre los sistemas deben poder realizarse utilizando entradas y salidas digitales y analógicas. Tanto el hardware como el software han de ser suficientemente robustos para que no fallen en ningún momento.

1.4 Juegos Didácticos

En el Informe Horizont del año 2011, aparecen analizadas las seis tecnologías informáticas emergentes que deben impactar la enseñanza universitaria en los próximos cinco años. Entre ellas aparece el aprendizaje basado en juegos. En este sentido, se explica el impacto del juego en el desarrollo cognitivo del adulto joven y la posibilidad de resolución de problemas, la colaboración

entre pares, además de la exploración de nuevos roles y la experimentación en ambientes simulados.

La sensación de trabajar hacia objetivos claros y sin ambigüedades, la posibilidad de soluciones a situaciones problemáticas simuladas, asumiendo un rol determinado, la interacción social en la red académica, entre otras, son características provenientes de los juegos, que pueden ser transferidas al proceso de aprendizaje del estudiante universitario.

El acto didáctico define la actuación del profesor para facilitar el aprendizaje de los estudiantes. Su naturaleza es esencialmente comunicativa. Los juegos didácticos como base del sustento para el aprendizaje y como ejercicio para poder incrementar el coeficiente intelectual, favorecen la agilidad mental y disminuyen en si el uso de otros métodos ortodoxos que empobrecen la calidad de una idea bien impartida.

Acorde a las características de los juegos didácticos: la participación, el dinamismo, el entretenimiento, la interpretación de papeles y la competencia estos también constan de tres fases:

- Introducción, ideada con el objetivo de iniciar al "jugador", dependiendo de algunas normas y pautas a seguir.
- Desarrollo, se lleva a cabo la actuación del jugador según las reglas del juego.
- Culminación, cuando el jugador llega a la meta que no es más que el resultado de todo aquello que aprendió o ejercitó durante el desarrollo del juego.

Los juegos didácticos se dividen en las siguientes categorías:

- Juegos Creativos.
- Juegos Profesionales.
- Juegos Didácticos.

Juegos Creativos: Permiten desarrollar en los estudiantes la creatividad, estimulan la imaginación y la producción de nuevas ideas.

Juegos Profesionales: Estimulan la rapidez y la motivación para entender y aprender. Se centran en la materia que se va a impartir.

Juegos Didácticos: Resulta un método muy eficaz, sobre todo para la enseñanza problémica.

Existen diversas variantes de cómo emplear estos:

- **De tipo Competitivo**: Encuentro de conocimientos, olimpiadas.
- **De tipo Profesional**: Análisis de situaciones concretas, análisis de casos, interpretaciones de papeles, simulación.

Algunos juegos didácticos que aparecen en los ambientes de aprendizaje interactivos son los llamados Serius Games.
No existe una única definición del término **"Juego Serio",** aunque se entiende que hace referencia a juegos usados en ámbitos como la formación, la publicidad, la simulación o la educación.

Los Juegos Serios están dirigidos a una gran variedad de público, desde estudiantes de educación primaria y secundaria a profesionales y consumidores. Los juegos serios pueden ser de cualquier género, usar cualquier tecnología de juegos y estar desarrollados para cualquier plataforma. Algunos los consideran un tipo de entretenimiento educativo, aunque el grueso de la comunidad se resiste a utilizar este término.

Un juego serio puede ser una simulación con la apariencia de un juego, pero está relacionado con acontecimientos o procesos que nada tienen que ver con los juegos, como pueden ser las operaciones militares o empresariales (aunque muchos juegos populares de entretenimiento están basados en operaciones militares o empresariales). Los juegos están hechos para proporcionar un contexto de entretenimiento y auto fortalecimiento con el que motivar, educar y entrenar a los jugadores. Otros objetivos de estos juegos son el marketing y la publicidad.

Al tiempo que los juegos serios están pensados para formar o educar a los usuarios, lo están también para entretener. Los desarrolladores de videojuegos tienen experiencia a la hora de crear juegos divertidos y atractivos ya que su sustento depende de ello. En el curso de los eventos y los procedimientos que se simulan, los desarrolladores automáticamente inyectan dosis de entretenimiento y jugabilidad a sus aplicaciones.

1.5 Clasificación de los Juegos Serios

Advergaming

Los advergames son videojuegos interactivos que permiten una exposición continuada del usuario ante la marca o producto publicitado. Como ejemplo de estos juegos esta el juego de motos online que se desarrollo para Repsol YPF para promocionar un portal Web.

Juegos educativos y de entretenimiento

Son juegos que tratan de formar el aprendizaje asistido. Los juegos educativos más comunes son los orientados a niños, normalmente clasificados por materias y edades.

En los juegos de entretenimiento, las personas son capaces de moverse e interactuar libremente, experimentando la vida como simulaciones, como si estuvieran realmente allí. La experiencia

interactiva y realista ayuda a las personas a aprender más rápidamente, recordar mejor los procedimientos y desarrollar las competencias esenciales. Ejemplos de estos juegos son: Fire Fighter Safety, Virtual Safety training y Education Plataform.

Salud

Los juegos sobre la salud también llamados "juegos de bienestar "o "health games", tratan temas relacionados con la prevención de enfermedades, la vida sana, y el ejercicio entre otros. Es el tipo de juego más investigativo por los claros beneficios sociales que aportan.

Un ejemplo claro es Cath the Sperm, un juego que se trata de concienciar sobre el uso del condón, o Emergencia 112, que enseña cómo se efectúan los primeros auxilios adecuadamente. Otros ejemplos son: Remission y Virtual Reality Medical Center.

Políticos y Sociales

Estos juegos son diseñados para educar al público sobre sus derechos y obligaciones como ciudadanos, para fomentar los comportamientos cívicos y las normas de convivencias. Las ONG los utilizan para denunciar injusticias o para promover la conciencia social.

Un ejemplo de estos juegos es Food Force.

La industria del videojuego con más de treinta años de experiencia ha aportado tecnologías suficientes para ser empleadas con éxito en el aprendizaje y en desarrollo de competencias en la rama ingenieril, pero solo en los últimos años han sido tomadas en cuenta por los educadores y los especialistas en Tecnología Educativa.

1.6 Escenario virtual de aprendizaje

La incorporación de las TIC a la educación es cada vez más acelerada, está produciendo una serie de cambios y transformaciones en las formas en que se representan y se llevan a cabo los procesos de enseñanza y aprendizaje (E-A). Estos cambios pueden observarse en los entornos tradicionales de educación formal, pero también en la aparición de los nuevos entornos educativos basados total o parcialmente en las TIC, como las denominadas Comunidades Virtuales de Aprendizaje (CVA).

Un escenario virtual del aprendizaje es concebido como la interfaz visual donde se realiza el proceso de aprendizaje con los objetivos y contenidos previos y donde se materializan los niveles de la habilidad o de la competencia que se pretende consolidar. Es donde acontecen las interacciones entre los objetos simulados y los participantes, que han sido objeto de un diseño del aprendizaje por parte del profesor y los especialistas informáticos. No necesariamente incluye simuladores tridimensionales, pero siempre son recomendables niveles de ayuda en forma de textos, imágenes, sonido y video, encaminados en la estrategia de aprendizaje prevista. (3)

Las fases presentes en el escenario de aprendizaje, corresponden a los hitos a realizar en los escenarios de aprendizaje y fueron definidos por el diseño de instrucciones o del aprendizaje, que pautan el cumplimiento gradual de los objetivos del laboratorio virtual.

Se pueden encontrar en un escenario de aprendizaje tres o más escenas, de acuerdo a las situaciones problemáticas que son presentadas al estudiante, en función de casos de estudio, o cualquier otro diseño de aprendizaje, asumido por el profesor. Esto depende del modelo pedagógico asumido y del nivel de la habilidad que se pretenda fortalecer.

1.7 Diseño del escenario virtual

Para la realización del diseño de un escenario virtual para la enseñanza y el aprendizaje se debe tener en cuenta entre otros aspectos la selección de los recursos tecnológicos y la planificación de los usos de dichos recursos, además, hacer un seguimiento de los usos que los participantes hacen de estos recursos y de su evolución, así como una valoración de nivel de logro de los objetivos educativos para los que fueron diseñados, y proceder a una reconstrucción y adaptación en consecuencia del diseño original.

Por lo general los escenarios virtuales de la enseñanza y el aprendizaje deben incorporar los aspectos que se mencionan a continuación:

- Un espacio para la creación, gestión y entrega de secuencias de actividades de aprendizaje, con propuestas realizadas por el profesor que los estudiantes puedan seleccionar y desarrollar.
- Una serie de dispositivos que permitan a los estudiantes identificar las características y las variables relativas de la exigencia de la tarea propuesta, de tal manera que puedan ajustar su forma de abordar la tarea tanto de manera individual como grupal y colaboración.
- Una serie de funciones automáticas que proporcionen información tanto al profesor como a los estudiantes sobre quién hace qué, cómo, cuando, con quién y con qué resultados, de manera que sea posible poner en marcha procesos de autorregulación y ofrecer ayudas al aprendizaje tanto de naturaleza individual como grupal.
- Una estructura dinámica que permita pasar con rapidez y facilidad del trabajo individual al trabajo grupal, conservando la identidad y especificidad de ambos espacios de trabajo, y que permita al profesor entregar devoluciones en ambos planos.

1.8 Lenguajes de Programación

Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por maquinas. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se aprueba, se compila y se mantiene el código fuente de un programa informático se le llama programación. (4)

1.8.1 C++

El C++ tiene su origen en 1980 y su nombre fue propuesto por Rick Mascitti es un lenguaje de programación resuelto de la extensión del C, agregándole la característica de que permite la manipulación de objetos, luego se le agregaron facilidades de programación genérica, una de las posibilidades que brinda C++ como lenguaje es que permite la sobrecarga de operadores y crear nuevos tipos. Otras de las ventajas que presenta son:

- Muy potente y robusto, por lo que se utiliza mucho para la creación de sistemas complejos, desde programas “Hello World” hasta Sistemas Operativos.
- Puede compilar y ejecutar código C.
- Tiene una gran cantidad de documentación en internet y código de ejemplo.
- Es un lenguaje muy portable por lo que se puede utilizar en varias plataformas de desarrollo.
- Es muy rápido en ejecución.

También presenta algunas desventajas señaladas a continuación:

- El uso de DLLs es muy complejo.
- Manejo de punteros y memoria, es una ventaja porque permite un mejor control de memoria, pero la inexperiencia de los desarrolladores o la pérdida de costumbre puede conllevar a un desastre.
- No es recomendable para el desarrollo de la Web.

1.9 Lenguajes interpretados

Estos tipos de lenguajes de programación, no requieren un código a ser compilado, ya que consisten en scripts que son interpretados en tiempo real por un intérprete, lo cual permite maximizar la eficiencia de los programas, en la mayoría de los casos. Entre los principales programas de este tipo que podemos encontrar, tenemos: Java, Perl, Python, Ruby, ASP, Bash, entre otros. (5)

Por lo general, los lenguajes interpretados son de alto nivel y están orientados a objetos y eventos, lo que facilita la programación web y la programación cliente/servidor, por lo cual, actualmente son lenguajes con mucho auge en el ámbito informático

Teóricamente, cualquier lenguaje puede ser compilado o ser interpretado, por lo que esta definición es aplicada puramente debido a la práctica de implementación común y no a alguna característica en particular del lenguaje.

1.9.1 Antecedentes históricos de los lenguajes interpretados

En los comienzos de la computación, el diseño del lenguaje fue fuertemente influenciado por la decisión de usar como modo de ejecución, la compilación o la interpretación. Por ejemplo algunos lenguajes compilados requieren que los programas deban indicar explícitamente el tipo de dato de una variable en el momento en que sea declarada o al ser usada por primera vez. Por otro lado, algunos lenguajes interpretados toman ventajas de los aspectos dinámicos de la interpretación para hacer tales declaraciones innecesarias.

Inicialmente, los lenguajes interpretados eran compilados línea por línea, cada línea era compilada a medida que estaba a punto de ser ejecutada, y si un loop o una subrutina hiciera que ciertas líneas se ejecutaran múltiples veces, ellas deberían ser recompiladas repetidamente. Esto ha llegado a ser mucho menos común. La mayoría de los lenguajes interpretados usan una representación intermedia, que combina tanto la compilación como la interpretación.

La representación intermedia puede ser compilada de una vez por todas, cada vez que se vaya a ejecutar o cada vez que un cambio en el código fuente es detectado antes de la ejecución.

1.9.2 Ventajas y desventajas de los lenguajes interpretados

Los lenguajes interpretados dan a los programas cierta flexibilidad adicional sobre los lenguajes compilados. Además de ser más fáciles de implementar en intérpretes que en compiladores incluyen otras características como:

- Independencia de plataforma.
- Reflexión y uso reflexivo del evaluador.
- Presentan tipos dinámicos.
- Facilidad en la depuración, es más fácil obtener información del código fuente en lenguajes interpretados.
- Pequeño tamaño del programa puesto a que los lenguajes interpretados tienen flexibilidad para elegir el código de instrucción.
- Presentan un ámbito dinámico.
- Gestión de memoria automática.

La ejecución del programa por medio de un intérprete es usualmente mucho menos eficiente que la ejecución de un lenguaje compilado. No es eficiente en tiempo porque, cada instrucción debe pasar por una interpretación en tiempo de ejecución, o el código tiene que ser compilado a una representación intermedia antes de cada ejecución. Otra desventaja es la necesidad de un intérprete en la máquina local para poder hacer la ejecución posible.

1.10 Lenguajes interpretados de uso común

A continuación son mencionados los lenguajes interpretados que fueron estudiados y tenidos en cuenta para la realización de la propuesta debido a las características que presentan y su usabilidad a nivel mundial en la producción de software.

1.10.1 ActionScript 3.0

ActionScript es un lenguaje de programación orientado a objetos, utilizado en especial en aplicaciones web animadas realizadas en el entorno Adobe Flash. Este lenguaje ofrece un modelo de programación robusto que resulta familiar y fácil de comprender a desarrolladores con conocimientos básicos sobre programación orientada a objetos.

Algunas de las ventajas que presenta son puestas a continuación:

- ActionScript 3.0 aumenta las posibilidades de creación de scripts de las versiones anteriores de ActionScript.
- Fue diseñado para facilitar la creación de aplicaciones muy complejas con conjunto de datos voluminosos y bases de código reutilizables y orientadas a objetos.
- El código ActionScript 3.0 puede ejecutarse a mayor velocidad que el anterior ActionScript.

1.10.2 Bash

Bash es un programa informático cuya función principal es interpretar órdenes. Está basado en la shell de Unix y es compatible con POSIX.

Bash es el intérprete predeterminado en la mayoría de los sistemas GNU/Linux, además de MacOSXTiger, y puede ejecutarse en la mayoría de los sistemas operativos tipo Unix, aunque también ha sido llevado a Microsoft Windows por diferentes por algunos proyectos.

1.10.3 Basic4GL

Basic4GL es uno de los tantos lenguajes interpretados que existen hoy en día en el mundo, este tiene como característica fundamental el uso de una sintaxis similar a los dialectos tradicionales del BASIC y ofrece IDE y una cuidadosa comprensiva y depuración.

Basic4GL no es diseñado para competir con lenguajes de programación como por ejemplo C++, fue pensado para sustituir viejos idiomas como QBasic o BASIC DE GFA. Este lenguaje fue diseñado para funcionar en Windows como sistema operativo pero una versión de este se está desarrollando para Linux.

1.10.4 BeanShell

BeanShell es un programa scripting basado en la sintaxis de java. Al igual que ocurre con Python o Perl podemos crear aplicaciones completas, puesto que BeanShell usa directamente una máquina virtual de java y puede usar todas las librerías de java disponible. Así mismo se pueden hacer fácilmente scripts para operaciones simples como en el lenguaje interpretado Bash.

1.10.5 Javascript

Javascript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque también existe una forma de Javascript del lado de servidor. También es reconocido su uso en aplicaciones externas a la web, como documentos PDF y aplicaciones de escritorios por mencionar algunas.

Javascript se diseñó con una sintaxis similar a C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y Javascript no están relacionados y tienen semánticas y propósitos diferentes.

1.10.6 Logo

Logo es un lenguaje de alto nivel, de muy fácil aprendizaje, razón por la cual suele ser el lenguaje de programación preferido para trabajar con niños y jóvenes. Fue diseñado con fines didácticos basado en las características del lenguaje Lisp. A pesar de que no fue creado con la finalidad de usarlo para enseñar programación, puede usarse para enseñar la mayoría de los principales conceptos de la programación, ya que proporciona soporte para manejo de listas, archivos y entrada/salida. Logo cuenta con más de 180 intérpretes y compiladores, además de ser uno de los pocos lenguajes de programación con instrucciones en español en algunos intérpretes.

1.10.7 PHP

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz grafica usando las bibliotecas Qt o GTK.

Aunque presenta un inconveniente que al ser un lenguaje que se interpreta en ejecución para ciertos usos puede resultar un inconveniente que el código fuente no pueda ser oculto. La ofuscación es una técnica que puede dificultar la lectura del código pero no la impide y en ciertos casos representa un costo en tiempos de ejecución.

1.10.8 Python

Python es un lenguaje de programación de alto nivel cuya filosofía hace hincapiés en una sintaxis muy limpia y que favorece un código legible.

Se trata de un lenguaje de programación multiparadigma ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico, es fuertemente tipado y es multiplataforma.

1.10.9 Lua

Lua es un lenguaje de programación imperativo, estructurado y bastante ligero que fue diseñado como lenguajes de script con una semántica extendible. Lua es un lenguaje de extensión, suficientemente compacto para usarse en diferentes plataformas. En Lua las variables no tienen tipo, sólo los datos y pueden ser lógicos, enteros, números con puntos flotantes o cadenas. Estructuras de datos como matrices, conjuntos, tablas, hash, listas y registros pueden ser representados utilizando la única estructura de datos de Lua; la tabla.

1.11 Ogre

Acrónimo del inglés Object Oriented Graphics Rendering Engine, es un motor de renderizado en tres dimensiones orientado a escenas y escrito en el lenguaje C++ y licenciado bajo la licencia LGPL (Lesser General Public License), además de contar con una comunidad muy activa. Sus librerías evitan la dificultad de la utilización de capas inferiores de librerías Gráficas como OpenGl y DirectX, además provee una interfaz basada en objetos del mundo y otras clases de alto nivel.

Ogre no solo fue desarrollado con el fin de la realización de juegos, sino también para simulaciones, aplicaciones de negocios o cualquier otro uso, además, para el caso de los juegos fue diseñado para explotar al máximo las posibilidades materiales de las tarjetas 3D, todo esto a través de una interfaz orientada a objetos; ofrece un sistema de partículas de gestión de recursos muy potente, además de una serie de funcionalidades muy interesantes.

1.11.1 Funcionalidades de Ogre

Funcionalidades:

- Productividad.
- Fácil de usar con interfaz orientada a objetos diseñada para minimizar el esfuerzo requerido al renderizar escenas 3D y separar lo de la implementación 3D como OpenGL o DirectX.
- Plataformas y soporte de APIs 3D.
- Posee soporte para DirectX y OpenGL.
- Soporte para Windows en todas sus versiones, GNU/Linux y Mac.
- Construcción en Visual C++ y Code::Blocks en Windows y g++ en GNU/Linux.
- Materiales y soporte de Shaders
- Mayas
- Animaciones
- Funcionalidades de la escena
- Efectos especiales

1.12 Entorno de Desarrollo Integrado (IDE)

Es un programa o sistema informático que está integrado por un grupo de herramientas de programación, un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Puede ser específicamente para un solo lenguaje de programación o varios al mismo tiempo. Proveen un marco de trabajo amigable y fácil de configurar para numerosos lenguajes de programación, puede funcionar también como un sistema en tiempo de ejecución lo que permite utilizar el lenguaje de manera interactiva, independiente del trabajo orientado a archivos de texto. (6)

Existen varios IDE los cuales pueden ser usados para el desarrollo de los Laboratorios Virtuales entre ellos se encuentran: Eclipse, C++ Builder 6, Qt Creator, y CodeBlock.

1.12.1 Microsoft Visual Studio

El Microsoft Visual Studio (VS) es in IDE, que existe solo para Windows, fue creado por Microsoft Corporation y soporta varios lenguajes de programación, tales como: ASP .NET, Visual J#, Visual C++, Visual C#. En la actualidad se le están agregando extensiones para muchos más. Es un IDE que permite a los desarrolladores crear sitios Web, aplicaciones de escritorio y para dispositivos móviles. Desde el lanzamiento de la versión 6.0 ha alcanzado un gran desarrollo, brindándole en cada versión más facilidades al programador. El VS 2010 es la versión más reciente la cual ya presenta herramientas para la creación de aplicaciones para Windows 7, entre las características que más destacan es la capacidad de utilizar más de un monitor, y compila las características de todas las ediciones comunes, Professional, Team Studio, Test, conocida como Visual Studio Ultimate

El VS permite al desarrollador:

- Programar sus aplicaciones disfrutando de un entorno altamente productivo, que además cuenta con diseñadores gráficos.
- Desarrollar y depurar aplicaciones multicapa de servidor desde un mismo entorno unificado de desarrollo.

Es un IDE que le brinda una interfaz amigable al desarrollador pero esta bajo licencia privativa, por lo tanto se ve restringido su uso.

1.13 Metodología de desarrollo de Software

Las metodologías de desarrollo de software son un conjunto de procedimientos, métodos y técnicas que ayudan a organizar el trabajo y la documentación con el objetivo de facilitar el desarrollo del producto. Es decir, son una guía de pasos a seguir, donde además indican qué persona y qué papel debe tener la misma para realizar una actividad específica en el desarrollo del software. Además detallan la información que se debe producir después de terminada una actividad. (7)

1.13.1 RUP (Rational Unified Process)

Se selección RUP (Proceso Unificado de Desarrollo) como metodología para realizar el análisis y diseño de la aplicación, porque es una metodología que acumula muchos años de experiencia en el desarrollo de software, además de estar rigurosamente probada a nivel mundial con el desarrollo

de innumerables sistemas de software. Dentro de las principales características de esta metodología y por lo cual se escogió se encuentran:

Guiado por casos de uso: Los casos de usos reflejan lo que los usuarios futuros necesitan y desean, constituye la guía fundamental establecida para las actividades a realizar durante el proceso de desarrollo del sistema.

Centrado en arquitectura: La arquitectura muestra la visión común del sistema completo.

Iterativo e Incremental: RUP divide el proyecto en fases de desarrollo, propone además que cada una de ellas se desarrolle en iteraciones, las cuales aportan un incremento en el proceso de desarrollo y terminan con el complimiento del punto de control trazado en la fase.

1.14 UML (Unified Modeling Language)

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones de conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas de ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar.

Un modelo UML nos indica que es lo que supuestamente hará el sistema, pero no como lo hará. Mediante las combinaciones de sus elementos gráficos UML nos permite conformar una serie de diagramas de manera estándar con lo que podemos generar un anteproyecto fácil de entender para cualquier persona involucrada en el proceso de desarrollo.

Con la notación UML podemos compartir y comunicar el conocimiento de una arquitectura a la combinación simultánea de cinco perspectivas:

- **Definir:** Fijar, determinar, decidir, explicar un concepto a través de sus atributos distintivos. Señalar sus límites y dar una idea exacta de lo que es esencial y de lo que es circunstancial.
- **Organizar:** Establecer unos recursos, disponer de un orden de responsabilidades y formalizar unas reglas de relación y actuación; todo ello orientado a conseguir un propósito.
- **Visualizar:** Representar mediante imágenes y/o símbolos el contenido y la organización de los conceptos que configuran un sistema. Hacer visible su naturaleza y su complejidad.
- **Actuar:** Pensar y tomar decisiones de manera ágil y sistemática, siguiendo un método; éste a su vez, define el modo de actuar sobre la base de relación de un conjunto de actores, actividades, entregables y certificaciones posibles en un escenario concreto.
- **Certificar:** Comprobar de manera fehaciente que in entregable es completo, coherente y usable para el propósito que ha sido creado.

El resultado, es una mayor comprensión y claridad sobre la naturaleza de los objetos, eventos y hechos que tienen consecuencias dentro de un dominio.

1.15 Herramienta CASE: Visual Paradigm

Las herramientas CASE (Computer Aided Software Engineering, ingeniería de software Asistida por Ordenador) son las aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y dinero. Estas herramientas contribuyen de manera directa en todos los aspectos del ciclo de vida de desarrollo de software en tareas como el proceso de realizar el diseño del proyecto, cálculo de costes, implementación de parte del código automático con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Para el modelado de los artefactos y diagramas generados a lo largo del ciclo de vida de proyecto se decidió emplear Visual Paradigm en su versión 6.4, pues su uso está muy estandarizado a nivel mundial y constituye una herramienta multiplataforma muy madura y acabada.

Visual Paradigm para UML es una herramienta profesional y fácil de utilizar, que soporta el ciclo de vida completo de desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Ayuda a una más rápida construcción de aplicaciones de calidad a un menor coste. Permite dibujar a todos los tipos de diagramas de clases, permite la realización de ingeniería tanto directa como inversa, generar el código desde diagramas y generar la documentación automática en varios formatos Web o Formato de Documento Portable (pdf), permite el control de versiones. Además, la herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto. Proporciona también abundantes tutoriales, demostraciones interactivas y proyectos de UML. Además es robusta y portable.

1.16 Conclusiones del capítulo

Durante el presente capítulo, donde se encuentra la base de estudio del tema en el cual estará inmerso nuestro trabajo, se realizó una profunda investigación acerca de los ambientes de aprendizaje interactivos, también se investigó sobre temas relacionados con las herramientas encargadas de estos entornos rigiéndose por un guión instruccional previamente definido para un trabajo más fácil de los desarrolladores.

CAPÍTULO II "SOLUCIONES TÉCNICAS"

2.1 Objetivo de la propuesta

Uno de los principales problemas que presenta el proyecto PROLAVI de Laboratorios Virtuales de la Facultad 5 son los cambios de diseño a realizar en los escenarios virtuales del aprendizaje a petición de cliente a la hora de ser instalado el producto, estos cambios de diseño implican cambios también en la aplicación principal desarrollada por el grupo de trabajo. Por lo que el objetivo de este trabajo es proponer al proyecto de un mecanismo capaz de realizar cambios de diseño en tiempo de ejecución mediante la utilización de lenguajes interpretados donde la aplicación principal no sea sometida a cambios.

Para el desarrollo de este trabajo se propone realizar un Demo, que en primer lugar permita visualizar una escena utilizando Ogre3D. Posteriormente se pasa a realizar cambios de diseño en dicha escena utilizando el lenguaje interpretado LUA y a mostrar gráficamente como ocurren estos cambios en tiempo de ejecución. Para dar solución a esta propuesta en la aplicación debe de estar incluida la biblioteca Luabind la cual permite realizar estos cambios en la aplicación de manera rápida y sencilla.

2.2 Historia de Lua

Lua fue creado en 1993 por Roberto lerusalimschy, Luiz Henrique de Figueredo y Weldemar Celes, miembros del Grupo de Tecnología de Computación Gráfica (Tecgraf) en la Pontificia universidad Católica de Río de Janeiro. Las versiones de Lua anteriores a la 5.0 fueron distribuidas bajo una licencia similar a la BSD, de la versión 5.0 en adelante se utiliza la licencia MIT, compatible con la GPL.

Lua ha sido usado en muchas aplicaciones comerciales y no comerciales, cuyo número aumenta cada año.

2.3 Características de Lua

Lua es un lenguaje interpretado pequeño y rápido que tuvo como meta inicial ser un lenguaje interpretado extensible dentro de las aplicaciones de software. El éxito favorable que ha tenido Lua como lenguaje script permite la utilización del mismo en muchos juegos y en aplicaciones de software. (8)

La semántica que presenta puede ser extendida y modificada redefiniendo funciones de las estructuras de datos utilizando meta tablas. También ofrece un buen soporte para la programación

orientada a objetos, programación funcional y programación orientada a datos. Lua ofrece soporte para funciones de orden superior, colector de basura

El hecho de poder ser extensible, simple, eficiente y presentar una portabilidad buena lo convierte en un lenguaje script pequeño pero poderoso.

Siendo un lenguaje de extensión, Lua no tiene noción de programa principal (main): sólo funciona embebido en un cliente anfitrión, denominado programa contenedor o simplemente anfitrión (host). Éste puede invocar funciones para ejecutar un trozo de código Lua, puede escribir y leer variables de Lua y puede registrar funciones C para que sean llamadas por el código Lua. A través del uso de funciones C, Lua puede ser aumentado para abarcar un amplio rango de diferentes dominios, creando entonces lenguajes de programación personalizados que comparten el mismo marco sintáctico. La distribución de Lua incluye un programa anfitrión de muestra denominado Lua, que usa la biblioteca de Lua para ofrecer un intérprete de Lua completo e independiente.

Lua es software libre, y se proporciona, como es usual, sin garantías, como se establece en su licencia.

2.4 Funcionamiento interno

Los programas en Lua no son interpretados directamente, sino compilados a código bytecode, que es ejecutado en la máquina virtual de Lua. El proceso de compilación es normalmente transparente al usuario y se realiza en tiempo de ejecución, pero puede hacerse con anticipación para aumentar el rendimiento y reducir el uso de la memoria al prescindir de compilador. También es posible la compilación en tiempo de ejecución utilizando LuaJIT.

2.5 Compilación en tiempo de ejecución

La compilación en tiempo de ejecución, también conocida como traducción dinámica, es una técnica para mejorar el rendimiento de sistemas de programación que compila bytecode a código de máquina nativo en tiempo de ejecución. La compilación en tiempo de ejecución se construye a partir de dos ideas anteriores relacionadas con los entornos de ejecución: la compilación a bytecode y la compilación dinámica. (9)

En un sistema que use compilación a bytecode como por ejemplo Smalltalk, Perl, GNU CLISP o las primeras versiones de Java, el código fuente es traducido a un código intermedio llamado bytecode. El bytecode no es el código máquina de ninguna computadora en particular, y puede por tanto ser portable entre diferentes arquitecturas. El bytecode es entonces interpretado, o ejecutado por una máquina virtual.

Un entorno de compilación dinámica es aquél en el que el compilador puede ser usado durante la ejecución. Por ejemplo, la mayoría de los sistemas Commons Lisp tienen una función compile que permite compilar nuevas funciones creadas durante la ejecución del programa. Aunque ventajoso en la depuración interactiva, la compilación dinámica es menos útil en un sistema de explotación desatendido. Este método es más común en emuladores modernos y frecuentemente comerciales que requieren mucha velocidad, como el Qemu y el VirtualPC (PC) o el Executor (Macintosh 68k).
En el entorno de compilación en tiempo de ejecución, la compilación a bytecode es el primer paso, reduciendo el código fuente a una representación intermedia portable y optimizable. El bytecode se despliega en el sistema de destino. Cuando dicho código se ejecuta, el compilador en tiempo de ejecución lo traduce a código de máquina nativo. Esto puede realizarse a nivel de fichero o de funciones, compilándose en este último caso el código correspondiente a una función justo cuando va a ejecutarse.
El objetivo es combinar muchas de las ventajas de compilación a código nativo y a bytecode: la mayoría del "trabajo pesado" de procesar el código fuente original y realizar optimizaciones básicas se realiza en el momento de compilar el bytecode, mucho antes del despliegue: así, la compilación a código de máquina del programa resulta mucho más rápida que partiendo del código fuente. El bytecode desplegado es portable, a diferencia del código máquina para cualquier arquitectura concreta. Los compiladores dinámicos son más fáciles de escribir, pues el compilador a bytecode ya realiza buena parte del trabajo.

2.6 Aplicaciones de Lua

Lua ha sido usado para procesar datos de entrada a sistemas complejos, configurar aplicaciones, controlar hardware y muchas otras cosas:

- En el gestor de ventanas Ion es posible utilizar Lua para personalizar la apariencia y extender su funcionalidad.
- El gestor de ventanas Awesome en su versión 3 utiliza Lua para su fichero de configuración.

Lenguajes como Flash, java, Lua y otros, son empleados en distintos sistemas operativos, lo cual consigue un ahorro de costes, al simplificar el trabajo de desarrollo de un nuevo programa de software, al añadirlos como partes "prefabricadas" que incluso al adaptar o portar el programa a nuevos usos, por ejemplo de videoconsolas a sistemas operativos como Android y otros, no necesitan ser modificados o mínimamente, convirtiéndolo en un programa de software de calidad nuevo, a un coste de desarrollo muy reducido.

También Lua es uno de los lenguajes de programación más utilizados para homebrews de la consola PSP de Sony debido a su sencillez. Van desde aplicaciones para añadir complementos u

otros programas fácilmente a la consola a entornos de ventanas excelentes y videojuegos muy completos.

Por otro lado es utilizado para los productos de la compañía canadiense desarrolladora de software Indigo Rose, en AutoPlay Media Studio; estos programas no fueron desarrollados en Lua, sino que utilizan Lua para generar y crear scripts, ya sea por un asistente o por la pericia del programador.

Debido a que Lua compilado es pequeño, veloz y tiene licencia permisiva ha ganado seguidores entre los desarrolladores de videojuegos. Algunos usos de Lua en videojuegos son mencionados a continuación:

- World of Warcraft, donde el usuario tiene la posibilidad de personalizar casi completamente la interfaz creando añadidos que permiten informarle de cualquier cosa en s correspondiente carpeta Interface, en la que WOW.exe tiene el intérprete de Lua y ejecuta en su interface el Addon creado en Lua.
- Este lenguaje es usado también en la interface del videojuego Half-Life 2, pudiendo modificarlo casi completamente mediante un mod tipo “sandbox” llamado Garry’s Mod que está escrito completamente en Lua.
- Existe también un mod para Half-Life 2 llamado Fortress Forever que permite configurar altamente los mapas.
- El RTS a gran escala Supreme Comamander, el cual es modificable por el usuario en casi todos sus aspectos.
- El juego RPG Tibia, modificable casi totalmente (poderes, mapas, entre otros) junto con XML.
- Parte de S.T.A.L.K.E.R-Shadow Of Chernobyl, permitiendo al jugador modificar armas, armaduras y varios aspectos del juego.
- Worms 4: Mayhem utiliza Lua y XML para definir misiones y desafíos.
- Ragnarok Online usa Lua para programar la inteligencia artificial de los homúnculos.
- Regnum Online usa Lua para la mayoría de scripts del juego como interfaz, modo de juego, acciones, entre otros.
- En el caso de Blitzkrieg se usa el lenguaje Lua en los editores de mapas e incluso los puede escribir uno mismo.
- En StepMania se usa Lua para desarrollar la implementación de animaciones del entorno gráfico, y asimismo la ejecución de comandos internos relacionados con la jugabilidad.

2.7 Razones para usar Lua

Después de estudiado los diferentes lenguajes script existentes en la actualidad se decidió utilizar Lua para la confección del Demo representativo por las características que presenta el miso.

Lua es un lenguaje de extensión que ha estado presente en un gran número de aplicaciones informáticas de gran importancia, su uso a nivel mundial viene dado por las características que presenta y que a continuación son mencionadas:

- **Portabilidad**: Lua corre prácticamente en todas las plataformas conocidas.
- **Simplicidad:** Presenta un único tipo de datos (tablas), un solo tipo numérico (generalmente double) y es orientado a objetos.
- **Pequeño tamaño:** Su peso es menor de 200 KB y su distribución completa cabe en un disquete.
- **Eficiencia:** Muchas marcas independientes muestran a Lua como uno de los lenguajes interpretados más rápidos con tipado dinámico.

Además de estas características, Lua posee otras cualidades que son tomadas a consideración para su uso y son presentadas a continuación:

Lua tiene la característica de ser robusta, rápida, portable, pequeña y libre.

2.8 API C

Actualmente, es muy común desarrollar partes de las aplicaciones utilizando un lenguaje script para un mejor desempeño. Existen lenguajes como Python, Perl y Ruby que poseen alguna forma de comunicarse con C.

Lua no es diferente y dispone de una API para que el código Lua pueda comunicarse con el código C y vice-versa.

Lua es un lenguaje extensible, pues podemos definir funciones en C y usarlas en programas de Lua. Al mismo tiempo, Lua también es un lenguaje de extensión, una vez que es posible imbuir Lua en una aplicación C.

Del mismo modo que la API de C permite que las funciones de Lua sean usadas en el código C, es posible también hacer llamadas a funciones de C en el programa realizado en Lua.

2.9 Bibliotecas de Lua

Las bibliotecas de Lua normales proporcionan funciones útiles que se llevan a cabo directamente a través del API de C. Algunas de estas funciones proporcionan los servicios esenciales al idioma; otros proporcionan el acceso a los servicios "externos"; y otros podrían llevarse a cabo en el propio Lua.

Todas las bibliotecas se llevan a cabo a través del API del C oficial y se proporcionan como los módulos de C separados. Actualmente, Lua tiene las bibliotecas normales siguientes:

- Bibliotecas básicas que incluyen sub-bibliotecas.
- Empaquetamiento de bibliotecas.
- Manipulación de cadenas.
- Manipulación de tablas.
- Funciones matemáticas (sin, log, entre otras).
- Bibliotecas de entrada y salidas.

Salvo las bibliotecas de paquetes, cada librería proporciona todas sus funciones como los campos de una tabla global o como los métodos de sus objetos.

2.10 Luabind

Teniendo en cuenta el estudio realizado anteriormente utilizaremos la biblioteca Luabind para la realización del trabajo, pues Luabind permite la utilización de funciones de C en el interpretador de Lua. Así que en los scripts de Lua se pueden usar funciones que se han escrito en C. Luabind es una API para C++ que encapsula mas limpiamente la registración de funciones y agrega registración de clases, soporte para templates, sobrecarga de operadores, entre otras.

Luabind es autorizada y se distribuye con la licencia de software del alza (BSL1.0) y la licencia del MIT.

Entre los rasgos más importantes que pueden ser extraídos de Luabind encontramos los siguientes.

Luabind soporta:

- Sobrecarga de funciones libres, así como la sobrecarga de funciones que son miembros del lenguaje.
- Luabind permite la utilización de una herencia sencilla en sus clases.

- Presenta una política para los parámetros y el retorno de valores.
- Permite llamar funciones de Lua a C++ y la exportación de clases de C++ enteras a Lua.

Luabind ha sido probada para trabajar en los compiladores siguientes:

- Visual Studio 7.1
- Intel C++ 6.0 (Windows)
- GCC 2.95.3 (cygwin)
- GCC 3.0.4 (Debian/Linux)
- GCC 3.1 (SunOS 5.8)
- GCC 3.2 (cygwin)
- GCC 3.3.1 (cygwin)
- GCC 3.3 (Apple, MacOS X)
- GCC 4.0 (Apple, MacOS X)

La principal desventaja que tiene la utilización de esta librería es que el tiempo de compilación aumentará para el archivo que hace el registro, por lo que es recomendado que se coloque todo en el mismo archivo .cpp.

2.11 Exportación de clases a Lua

Una de las características que presenta Luabind y en la que enfatizaremos en nuestro trabajo es la exportación de clases enteras de C++ a Lua a través de módulos, incluyendo constructores con sus argumentos como se muestra a continuación:

```
module(L)
[
    def("f", &f),
    def("g", &g),
    class_<A>("A")
        .def(constructor<int, int>),
    def("h", &h)
];
```

Los siguientes módulos forman parte de la biblioteca estándar de Lua:

- **coroutine:** posee las operaciones relacionadas con co-rutinas.
- **string:** contienen funciones que manipulan funciones de caracteres.
- **table:** manipulación de tablas.
- **math:** módulo con las funciones matemáticas.

- **io:** biblioteca de entrada y salida (E/S).
- **package:** biblioteca de módulos.
- **os:** Implementa funciones del sistema operacional.
- **debug:** biblioteca de depuración.

Para realizar los lazos entre C++ y Lua es necesario usar la función luabind::def() como a continuación se muestra:

```
template<class F, class policies>
void def(const char* name, F f, const Policies&);
```

- Name es el nombre que la función tendrá en el script de Lua.
- F es el indicador de la función que se quiere registrar.
- Las políticas es parámetro que es usado para describir cómo van a ser usados los valores de retorno por la función, este es un parámetro opcional.

A continuación se muestra un ejemplo de cómo sería la registración de estas funciones si se quisiera registrar la función float std::sin(float):

```
module(L)
[
    def("sin", &std::sin)
];
```

2.12 Reglas del negocio

En la aplicación correspondiente a la propuesta que se hace en este trabajo, los cambios de diseño son realizados en una escena creada en Ogre3D, porque durante la investigación del trabajo se llegó a la conclusión de que el proyecto se encuentra realizado con esta herramienta y por lo tanto dichos cambios serán realizados a través de un script realizado en LUA y mostrados en tiempo real.

2.13 Captura de Requisitos

Los requisitos son condiciones o capacidades que deben ser alcanzadas por un sistema para satisfacer las necesidades del cliente. Se pueden clasificar como funcionales y no funcionales. Los funcionales representan las capacidades o funciones que el sistema de debe de cumplir, y los no funcionales son las propiedades o cualidades que el producto debe de tener. Estos facilitan el entendimiento entre clientes y desarrolladores del sistema a elaborar. A continuación se muestran los requisitos funcionales y no funcionales a tener en cuenta para la construcción del sistema.

2.14 Requisitos Funcionales

Los requisitos funcionales que identifican este sistema son:

1. El sistema debe permitir la visualización de la escena a modificar.
   1.1 Permitir crear una escena y ubicar dentro de la misma el objeto al cual se le van a aplicar los cambios de diseño.
2. El sistema debe permitirle al programador realizar cambios de diseño y ser guardados en la escena para ser visualizados posteriormente.
   2.1 Permitir realizar cambios de textura al objeto dentro de la escena.
   2.2 Permitir realizar cambios de dimensión al objeto dentro de la escena.
   2.3 Permitir realizar cambios de color al objeto dentro de la escena.
   2.4 Visualizar los cambios de diseño realizados por el programador después de ser guardados en el script.
3. El sistema debe permitir cargar el script realizado en lenguaje LUA para la realización de los cambios de diseño.
   3.1 El sistema debe brindarle facilidad al programador de aplicar cambios de diseño mediante la utilización del script.

2.15 Requisitos no Funcionales

1. **De software:**
   - Sistema Operativo Windows XP o superior y cualquier distribución de Linux.
2. **Requisitos de Hardware:**
   - 512 MB de RAM como mínimo.
   - Procesador Pentium 4 o superior.
3. **Restricciones en el diseño y la implementación:**
   - Se utilizará OGRE3D para la realización de la escena.
   - Se hará uso de la biblioteca Luabind.
4. **Requisitos de soporte:**
   - Soporte para los Sistemas Operativos Windows XP y Linux
5. **Usabilidad.**
   - Será fácil de usar por los programadores aunque sea la primera vez que se enfrenten al lenguaje script seleccionado.

2.16 Modelo de negocio

En la figura 1, a continuación se representa el modelo de negocio para facilitar la comprensión de la solución propuesta, donde se representan los principales conceptos y relaciones existentes entre ellos que a continuación se presentan mediante un **glosario de términos:**

El **programador** es aquella persona que posee los privilegios para acceder a la aplicación y realizar los cambios solicitados por el cliente.

La **escena** es el lugar donde van a estar los objetos que van a ser objeto de cambios.

El **script** es el archivo que va a ser programado bajo en lenguaje interpretado LUA y va ser utilizado en la escena.

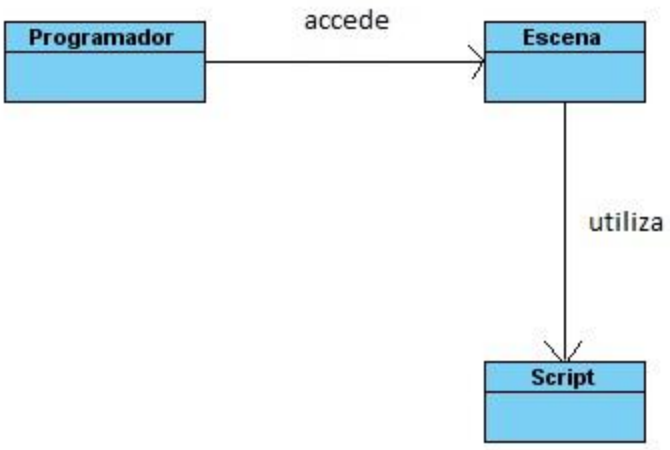

Figura 1 Diagrama del modelo del negocio

2.17 Modelo de Casos de Uso de Sistema

El programador es el único actor del sistema que interactúa con la aplicación, iniciando todos los casos de uso del sistema. En la figura No.2 se observan los casos definidos para representar el flujo de los eventos que ocurren dentro de la aplicación y su interacción con los actores.

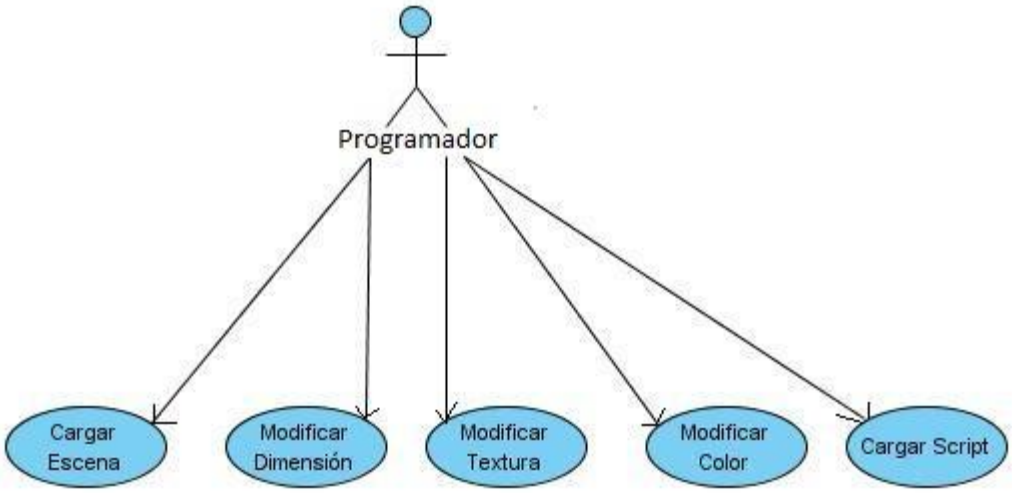

Figura 2 Diagrama de casos de uso del sistema.

2.18 Descripción de actores

Tabla No.1: Actor del sistema

Actor	Descripción
Programador	Es el encargado de generar la escena, realizar los cambios en la misma, así como cagar el script en el que van a estar recogidos estos cambios.

2.19 Casos de Uso del sistema

Tabla No. 2: Caso de Uso Cargar Escena.

CU-1	Cargar Escena
Actor	Programador
Descripción	El programador es el que realiza los pasos para Cargar la escena a la cual se le van a aplicar los cambios.
Referencia	R1

Tabla No. 3: Modificar Dimensión.

CU-2	Modificar Dimensión.
Actor	Programador
Descripción	El programador es el que selecciona el cambio de diseño “Modificar Dimensión” que va a hacer aplicado sobre el objeto en la escena.
Referencia	R1

Tabla No. 4: Caso de Uso Modificar Textura.

CU-2	Modificar Textura.
Actor	Programador
Descripción	El programador es el que selecciona el cambio de diseño “Modificar Textura” que va a hacer aplicado sobre el objeto en la escena.
Referencia	R1

Tabla No. 5: Caso de Uso Modificar Color.

CU-2	Modificar Color.
Actor	Programador
Descripción	El programador es el que selecciona el cambio de diseño “Modificar Color” que va a hacer aplicado sobre el objeto en la escena.
Referencia	R1

Tabla No. 6: Caso de Uso Cargar Script.

CU-3	Cargar Script

Actor	Programador
Descripción	El programador cargar el script necesario para aplicar los cambios en el objeto en la escena creada con anterioridad.
Referencia	R3

2.20 Descripción de los Casos de Uso

A continuación se describe como se lleva a cabo y se ejecuta un caso de uso determinado, en términos de las clases de análisis y de sus objetos en interacción.

Tabla No. 7: Descripción textual del Caso de Uso Cargar Escena.

Caso de Uso:	**Cargar Escena**
Actores:	Programador
Resumen:	El caso de uso se inicia cuando el programador crea y visualiza la escena a la cual se le aplicarán los cambios de diseño.
Precondiciones:	Debe de estar integrada la biblioteca Luabind para realizar los cambios de diseño.
Referencias	R1
Flujo Normal de Eventos	
Acción del A	**Respuesta del Negocio**
1) El programador ejecuta la aplicación para ejercer su trabajo sobre ella.	1.1) El sistema muestra la escena con los datos iniciales y el objeto al cual se le van a aplicar los cambios.
Poscondiciones	

Tabla No. 8: Descripción textual del Caso de Uso Modificar Dimensión.

Caso de Uso:	**Modificar Dimensión**
Actores:	Programador
Resumen:	El programador podrá observar en la escena los cambios de dimensión que son realizados al objeto que se encuentra dentro de la misma.
Precondiciones:	Debe de estar integrada la biblioteca Luabind para realizar los cambios de diseño.
Referencias	R1
Flujo Normal de Eventos	
Acción del A	**Respuesta del Negocio**
2) El programador configura los cambios referidos a la dimensión del objeto que se quieren aplicar en la aplicación.	1.2) El sistema muestra la escena con los datos los cambios realizados después de guardados los mismos.
Poscondicio-nes	

Tabla No. 9: Descripción textual del Caso de Uso Modificar Textura.

Caso de Uso:	**Modificar Textura**
Actores:	Programador
Resumen:	El programador podrá observar en la escena los cambios de textura que son realizados al objeto que se encuentra dentro de la misma.
Precondicion	Debe de estar integrada la biblioteca Luabind para realizar los cambios de diseño.

es:	
Referencias	R1
Flujo Normal de Eventos	
Acción del A	**Respuesta del Negocio**
3) El programador configura los cambios referidos a la textura del objeto que se quieren aplicar en la aplicación.	1.3) El sistema muestra la escena con los datos los cambios realizados después de guardados los mismos.
Poscondicio-nes	

Tabla No. 10: Descripción textual del Caso de Uso Modificar Color.

Caso de Uso:	**Modificar Color**
Actores:	Programador
Resumen:	El programador podrá observar en la escena los cambios color que son realizados al objeto que se encuentra dentro de la misma.
Precondicion es:	Debe de estar integrada la biblioteca Luabind para realizar los cambios de diseño.
Referencias	R1
Flujo Normal de Eventos	
Acción del A	**Respuesta del Negocio**
4) El programador configura los cambios referidos al color del objeto	1.4) El sistema muestra la escena con los datos los cambios realizados después de guardados los mismos.

que se quieren aplicar en la aplicación.	
Poscondicio-nes	

Tabla No. 11: Descripción textual del Caso de Uso Cargar Script.

Caso de Uso:	Cargar Script.
Actores:	Programador
Resumen:	La aplicación debe de ser capaz de cargar el script realizado en el lenguaje script LUA con las opciones para realizar los cambios de diseño que son requeridos por el cliente.
Precondicion es:	Debe de estar integrada la biblioteca Luabind para realizar los cambios de diseño.
Referencias	R3
Flujo Normal de Eventos	
Acción del A	**Respuesta del Negocio**
5) El programador selecciona el script con las opciones de incluidas de los cambios de diseño.	1.5) El sistema muestra el script con los datos para ser cambiados y posteriormente son guardados para ser mostrados en la escena.
Poscondicio-nes	

2.21 Conclusiones del capítulo

En este capítulo completó la fase del Modelo de Negocio, donde se han descrito los procesos con la especificación de sus actividades y actores involucrados, así como los casos de uso del negocio. Se realizó además el levantamiento de requisitos funcionales y posteriormente la presentación de los requisitos no funcionales, que constituyen el paso fundamental para adentrarse en este modelo y el correcto funcionamiento del sistema a implementar. Se presentó el Diagrama de Casos de Uso del Sistema y la perspectiva descripción de cada uno de los casos de uso que intervienen en el mismo, haciendo evidente la forma en la que quedará estructurado el sistema.

Capítulo III "Descripción de la propuesta"

Muchas veces las personas presentan problemas en la programación cuando surge la necesidad de ajustar cualquier elemento dentro del programa o para establecer sus valores. Para la solución de estos problemas se requiere de la reconstrucción del código una y otra vez, que gasta mucho tiempo, por lo que surge la idea de incorporarle un lenguaje script al programa. De esta manera se podía cambiar, guardar y volver a ejecutar el programa ahorrando gran cantidad de tiempo.

En estos momentos el grupo de desarrollo del proyecto PROLAVI se encuentra trabajando en la elaboración de tres laboratorios virtuales: Ensamblaje de computadoras, Diseño e instalación de una red de área local (LAN) y Configuración y Administración de una red de área local (LAN), en los cuales si se desea realizar cambios de diseño se ve comprometida la aplicación principal desarrollada por el grupo de trabajo.

Para lograr que estos cambios de diseño no comprometan la integración principal desarrollada por el proyecto se pueden incluir en estas aplicaciones el uso de lenguajes de extensión como es el caso de Lua, debido a las ventajas que le ofrece al proyecto que se encuentra usando Ogre en su integración.

La incorporación de Lua al proyecto se hace de manera tal que los cambios de diseño puedan ser observados al instante.

3.1 Laboratorio Virtual Arquitectura de Computadoras

Este laboratorio virtual, como su nombre lo indica, consiste en que el estudiante debe de ser capaz de armar una computadora, la misma cuenta con 2 fases: en la primera se ensamblan los elementos internos de la computadora, tales como: memoria RAM, micro-procesador, tarjeta madre, entre otros; la segunda fase tiene el mismo principio de funcionamiento pero con las partes externas de la computadora, ejemplo: mouse, teclado, y los cables, entre otros.

La principal funcionalidad de este laboratorio virtual es el manejo de objetos de la escena, mediante los cuales el usuario va conformando una computadora, con los elementos que contiene el panel de herramientas, independientemente de la fase en que se encuentre desarrollado el laboratorio virtual, y en común para todas se muestra un conjunto de herramientas que pueden ser utilizadas en el ensamblaje de la computadora, por ejemplo: destornilladores, pinzas y otros. El reporte que se genera contiene los dispositivos que se conectaron a la computadora, los errores cometidos en cada dispositivo conectado y el tiempo en que se conectó finalmente la tarjeta madre dentro del case.

3.2 Laboratorio Virtual Diseño e instalación de una Red Local (LAN)

Este laboratorio está dividido en 3 fases, en cada una de ellas se persiguen diferentes objetivos; la primera fase, consiste en confeccionar un cable de red, para el cual se brindan un grupo de herramientas y finalmente se hace una prueba para comprobar el estado del cable; la fase numero 2, consiste en diseñar una red local en dos dimensiones, la principal funcionalidad de esta fase es permitirle al usuario confeccionar una red, colocando los dispositivos y conectándolos entre ellos; la última fase de este laboratorio tiene como principal funcionalidad diseñar una red en un local (Edificio), donde e usuario coloca los distintos dispositivos de red y los conecta entre ellos, desplazándose por todo el edificio. La interfaz le muestra al usuario un panel lateral donde se encuentran todos los equipos que puede colocar en el diseño de la red, tanto para la fase 2, como para la fase 3, y en caso de la fase 1 se muestran también los tipos de cables que pueden ensamblar el usuario, además en común se muestran para cada una de ellas, las herramientas que se pueden utilizar en las mismas, ejemplo; Pinzas y otros. Al final el usuario puede exportar el estado de la red diseñada

3.3 Laboratorio Virtual Administración y Configuración de una red Local (LAN)

Este laboratorio virtual está dividido en varias actividades, el mismo comienza cargando la red que se diseña en el Laboratorio Virtual de Diseño e Instalación de la red. Se le brinda la posibilidad de administrar y configurar los diferentes terminales de red (Computadoras), todo esto mediante un diálogo que le muestra al usuario las distintas opciones de lo que puede configurar en uno u otro sistema operativo (GNU/Linux MS Windows).

3.4 Descripción del caso de estudio

Para demostrar que es posible incluir el lenguaje de extensión Lua en el desarrollo de aplicaciones de Realidad Virtual que demanden un alto nivel de diseño se tomará como ejemplo un pequeño Demo donde existen objetos gráficos que deben responder a los cambios de diseño que se realicen sobre él y desde el script realizado en Lua. Para lograr esto, se realiza una aplicación donde se encuentra integrado Ogre y Lua/Luabind, y la misma es capaz de visualizar los cambios realizados en la escena.

A continuación se describe detalladamente el ejemplo. En el Demo se pueden realizar una serie de cambios que se encuentran previamente definidos y estos pueden ser visualizados en la escena.

Los posibles cambios a realizar son los siguientes:

- Modificar color.
- Modificar textura.
- Modificar Dimensiones.

Por supuesto, para una aplicación de grandes dimensiones como las realizadas en el proyecto PROLAVI el trabajo se tornaría un poco complejo debido a la complejidad y diversidad que estos presentan es por eso que para probar esta técnica se realiza un Demo demostrativo.

3.5 Integración de Lua con C++

A diferencia de otros lenguajes script, Lua fue hecho para su integración con C++ y la manera en que lo hace es bastante simple.

Lo primero para integrar Lua con C ++ es abrir un handler para Lua, este handler se usa en todas las funciones relacionadas con Lua y provee una forma fácil para realizar funciones escritas en C++ y llamadas desde Lua:

Código:

```
/*abrimos nuestro handler*/
lua_State *ls = lua_open();
if(ls == NULL){
        /*error*/
}
```

Luego que terminamos de usar Lua o antes de salir del programa necesitamos cerrar nuestro handler con la función lua_close:

Código:

```
/*cerramos nuestro handler*/
lua_close(ls);
```

Para lograr interpretar el código de Lua es necesario el uso de tres funciones: **lua_dostring**, **lua_dofile y lua_dobuffer**. La función lua_dobuffer se utiliza para interpretar el código compilado de Lua, mientras que lua_dostring se utiliza para interpretar código Lua sin compilar, pero antes debemos cargar la biblioteca base antes de llamar a la función lua_dostring como se muestra a continuación:

Código:

```
extern "C" {
#include "lua.h"
#include "lualib.h"
#include "lauxlib.h"
}
```

Para el comenzar el trabajo con Lua desde una aplicación debemos designar ciertas funciones en el script, para que se ejecuten en el programa, esto se hace de la siguiente manera:

1. Escribimos la función que se va a ejecutar.
2. La registramos en Lua.
3. Cada vez que Lua encuentre el nombre de la función que registramos, ejecuta el código de la función.

Lua es un lenguaje con asignación dinámica de tipos de datos, es decir que una variable es int, string, float dependiendo del contexto en que se use, por lo que para pasar datos entre nuestro script y nuestro programa, estos deben ser convertidos siempre a C. Para realizar este trabajo existen las funciones: lua_tonumber, lua_tostring, lua_tocfunction, entre otras.

Hasta ahora se ha visto la llamada de funciones escritas en C++ desde un script Lua, pero para tener completa la integración es necesario llamar funciones escritas en Lua desde nuestra aplicación. Para poder hacer esto necesitamos:

1. Poner la función en la pila.
2. Poner los argumentos en la pila.
3. Llamar la función.

Para realizar este llamado de funciones necesitamos algo parecido al fragmento de código que aparece a continuación:

Código:

```
lua_getglobal(ls,"FuncionEnLua"); //ponemos nuestra funcion en el tope
lua_pushnumber(ls,arg1);          //luego nuestros argumentos
lua_pushnumber(ls,arg2);

if(lua_pcall(ls,2,1,0) != 0){

}
```

La función lua_getglobal, además fuera de ayudarnos en las llamadas a funciones en Lua, nos ayuda también a obtener el valor de datos globales en nuestro script.

La compilación del script Lua, se lleva a cabo mediante el compilador de Lua llamado Luac y se utiliza de la siguiente manera:

Luac –o script.bin script.lua

La misma función que interpreta un archivo de script, es la misma que ejecuta el archivo compilado, es decir lua_dofile.

Código:

```
lua_dofile(ls,"script.bin");
```

3.6 Integración de Ogre y Lua/Luabind

Como propuesta de la implementación para la integración, será creada una clase llamada "Lazos", utilizando como lenguaje de programación C++. Esta clase tendrá definidos los métodos que permitirán la exportación de las clases a Lua con sus constructores incluidos y sus funciones para la realización de los lazos entre C++ y Lua, debido a que la biblioteca que utilizamos en la propuesta permite esta unión entre los dos lenguajes.

En la figura 3, a continuación se muestra un ejemplo de cómo quedaría la exportación de clases con sus constructores:

```
void bindColourValue( lua_State* L )
{
    module(L)
    [
        class_<ColourValue>("ColourValue")
        .def(constructor<>())
        .def(constructor<Real, Real, Real, Real>())
        .def(constructor<Real, Real, Real>())
        .def(tostring(self))
        .def_readwrite( "r", &ColourValue::r)
        .def_readwrite( "g", &ColourValue::g )
        .def_readwrite( "b", &ColourValue::b )
        .def_readwrite( "a", &ColourValue::a )
        .def( "saturate", &ColourValue::saturate )


    ];
```

Figura 3 Exportación de clases.

Además en el Demo aparecen otras clases principales como es el caso de “Aplicación”, la misma es la encargada de cargar la escena de Ogre en la que van a hacer visualizados los cambios de diseño.

La clase correspondiente al estado de Lua “EstadoLua” es la encargada de iniciar el trabajo con las bibliotecas de Lua, además de cargar el script y chequear si son correctas las funciones que se encuentran dentro de él.

3.6 Resultados

Tras haber estudiado los diferentes usos que tienen los lenguajes de extensión en el área de los videojuegos, de acuerdo a las ventajas que brindan los mismos, se puede afirmar que su uso en el desarrollo de los laboratorios virtuales potenciaría muchos beneficios, ya que ganarían las funcionalidades que brindan los lenguajes interpretados a los videojuegos, tales como son las configuraciones y la personalización de sus ambientes, adecuándose a un usuario específico o a las necesidades de los clientes. También sería más factible utilizar uno de estos lenguajes interpretados, por su sencillez a la hora de programar y de entender sus códigos, en la integración de los laboratorios virtuales que es algo muy engorroso de realizar hoy día.

CONCLUSIONES

Se realizó un estudio de las necesidades del proyecto de Laboratorios Virtuales de la facultad con el objetivo de analizar qué tipo de técnicas podían ser incluidas en la aplicación principal para no comprometer su integridad.

Además se hizo un estudio de los diferentes lenguajes script, especificándose sus ventajas y desventajas para así seleccionar el más adecuado para utilizar de acuerdo a las necesidades del proyecto, y se concluyó usando el lenguaje script Lua en una aplicación demostrativa, donde prueba que es un lenguaje que se adapta muy bien a las necesidades del proyecto para realizar los cambios de diseños en los escenarios virtuales con que se cuenta.

Como resultado de la investigación se propone el uso del lenguaje script Lua para ser utilizada en el proyecto de Laboratorios Virtuales, del cual se logró especificar su funcionamiento y sus características.

RECOMENDACIONES

Se recomienda:

- Utilizar el presente trabajo como punto de partida para profundizar sobre los temas relacionados con el lenguaje script Lua y su inclusión en aplicaciones para lograr un mejor desempeño de las mismas.
- Aplicar la técnica que se propone en el proyecto de Laboratorios Virtuales para lograr mejores resultados y simplificar el trabajo que realiza grupo de trabajo.
- Que se tenga en cuenta esta investigación para que en los nuevos proyectos relacionados con Realidad Virtual y más específicamente los próximos laboratorios virtuales a desarrollar por el proyecto PROLAVI, se aplique esta técnica.

REFERENCIAS BIBLIOGRAFICAS.

1. **L.Rosado y Herreros, J.R.** *Nuevas aplicaciones didácticas de los laboratorios virtuales y remotos en la enseñanaza Física.* 2005.

2. **Viciedo Caraballoso, Luis Gabriel y Gonzáalez campistruz, Jaime.** *Estrategias para la implementación de laboratorios virtuales con fines de aprendizaje con el manejo del lenguaje de programación Lua.* 2011.

3. **García Pérez, yamileidis y Dominguez Soria, Juan Carlos.** *Desarrollo de instrumentos virtuales para un laboratorio virtual de Obstetricia.* 2010.

4. Wikipedia La Enciclopedia Libre. [En línea] http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n.

5. wikipedia. [En línea] http://es.wikipedia.org.

6. Wikipedia. *Wikipedia.* [En línea] http://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado.

7. scribd. [En línea] http://www.scribd.com.

8. Lua. [En línea] http://www.lua.org.

9. wikipedia . [En línea] http://es.wikipedia.org.

10. Culturacion Artículos tecnológicos educativos. *Culturacion Artículos tecnológicos educativos.* [En línea] http://culturacion.com.

11. Sitio Web Wikipedia. [En línea] http://es.wikipedia.org/wiki/Lenguaje_de_programacon.

12. **Avila, Patricia.** *Ambientes Virtuales de Aprendizaje, una nueva experiencia.* 1999.

13. **Emerich, Paul.** *Beginning Lua with World of Warcraft Addons.* 2009.

14. **G. Booch, I. Jacobson y J. Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* 2000.

15. **Marchena, Daniel.** *Entornos Virtuales de Aprendizaje.* 2008.

16. **Hanson, Kami y Shelton, Brett E.** *Design and Development of Virtual Reality: Analysis of Challenges Faced by.* 2008.

17. **Sanz Pardo, Annette y Martínez Vázquez, Juan Luis.** *EL USO DE LOS LABORATORIOS VIRTUALES EN LA ASIGNATURA BIOQUIMICA COMO ALTERNATIVA PARA LA APICACION DE LAS TECNOLOGIAS DE LA INFORMACION Y LA COMUNICACION.* 2005.

18. **Monge Nájera, Julian y Méndez Estrada, Victor Hugo.** *Ventajas y desventajas de los laboratorios virtuales en educación a distancia.* 2007.

19. Rasterbar Software. [En línea] http://www.rasterbar.com/.

20. The Code Project. [En línea] http://www.codeproject.com/.

BIBLIOGRAFÍA

1. **L.Rosado and Herreros, J.R.** *Nuevas aplicaciones didácticas de los laboratorios virtuales y remotos en la enseñanaza Física.* 2005.

2. **Viciedo Caraballoso, Luis Gabriel and Gonzáalez campistruz, Jaime.** *Estrategias para la implementación de laboratorios virtuales con fines de aprendizaje con el manejo del lenguaje de programación Lua.* 2011.

3. **García Pérez, yamileidis and Dominguez Soria, Juan Carlos.** *Desarrollo de instrumentos virtuales para un laboratorio virtual de Obstetricia.* 2010.

4. Wikipedia La Enciclopedia Libre. [Online] http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n.

5. wikipedia. [Online] http://es.wikipedia.org.

6. Wikipedia. *Wikipedia.* [Online] http://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado.

7. scribd. [Online] http://www.scribd.com.

8. Lua. [Online] http://www.lua.org.

9. wikipedia . [Online] http://es.wikipedia.org.

10. Culturacion Artículos tecnológicos educativos. *Culturacion Artículos tecnológicos educativos.* [Online] http://culturacion.com.

11. Sitio Web Wikipedia. [Online] http://es.wikipedia.org/wiki/Lenguaje_de_programacon.

12. **Avila, Patricia.** *Ambientes Virtuales de Aprendizaje, una nueva experiencia.* 1999.

13. **Emerich, Paul.** *Beginning Lua with World of Warcraft Addons.* 2009.

14. **G. Booch, I. Jacobson and J. Rumbaugh.** *El Proceso Unificado de Desarrollo de Software.* 2000.

15. **Marchena, Daniel.** *Entornos Virtuales de Aprendizaje.* 2008.

16. **Hanson, Kami and Shelton, Brett E.** *Design and Development of Virtual Reality: Analysis of Challenges Faced by.* 2008.

17. **Sanz Pardo, Annette and Martínez Vázquez, Juan Luis.** *EL USO DE LOS LABORATORIOS VIRTUALES EN LA ASIGNATURA BIOQUIMICA COMO ALTERNATIVA PARA LA APICACION DE LAS TECNOLOGIAS DE LA INFORMACION Y LA COMUNICACION.* 2005.

18. **Monge Nájera, Julian and Méndez Estrada, Victor Hugo.** *Ventajas y desventajas de los laboratorios virtuales en educación a distancia.* 2007.

19. Rasterbar Software. [Online] http://www.rasterbar.com/.

20. The Code Project. [Online] http://www.codeproject.com/.

Figura 4 Fase 1 L.V Arquitectura de Computadores (Partes Internas).

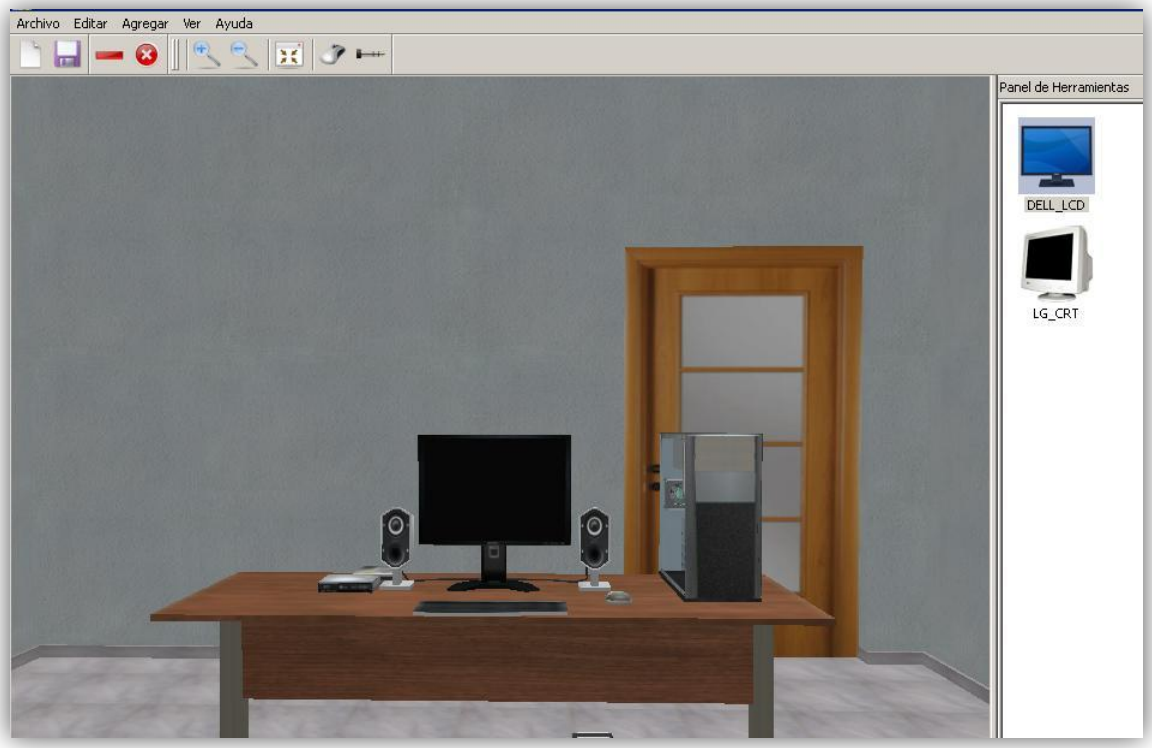

Figura 5 Fase 2 L.V Arquitectura de Computadores (Partes Externas).

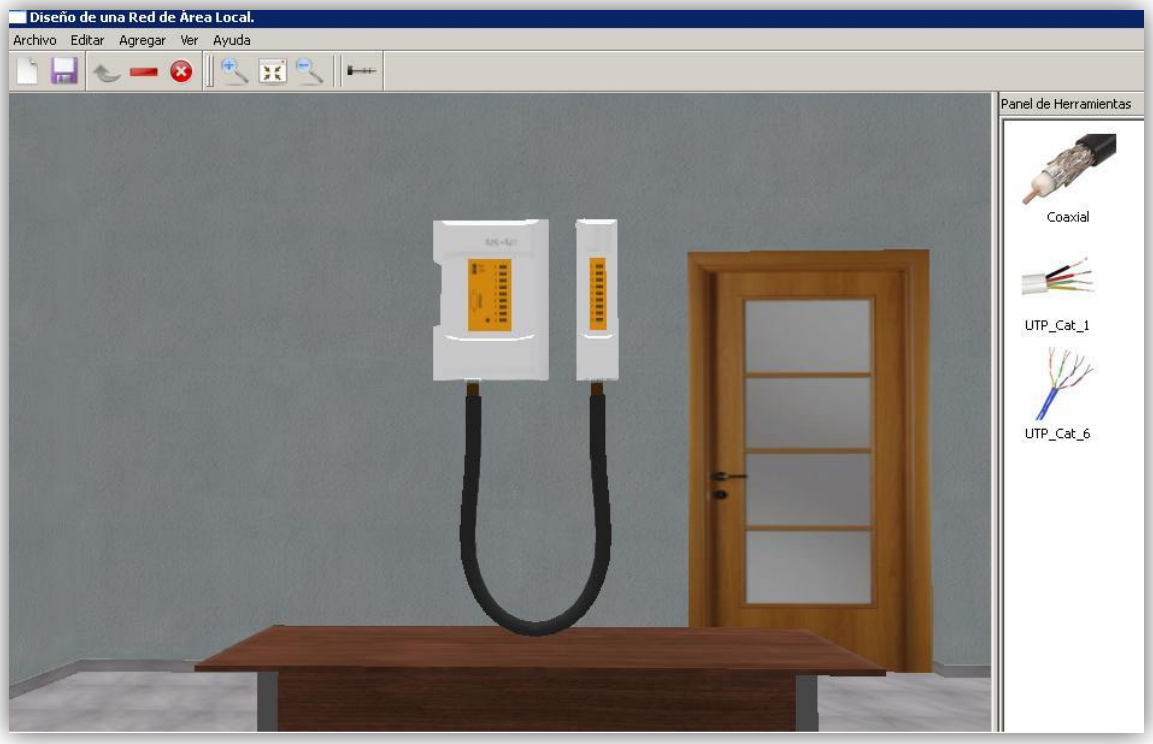

Figura 6 Fase 1 L.V Diseño e Instalación de una red local (LAN).

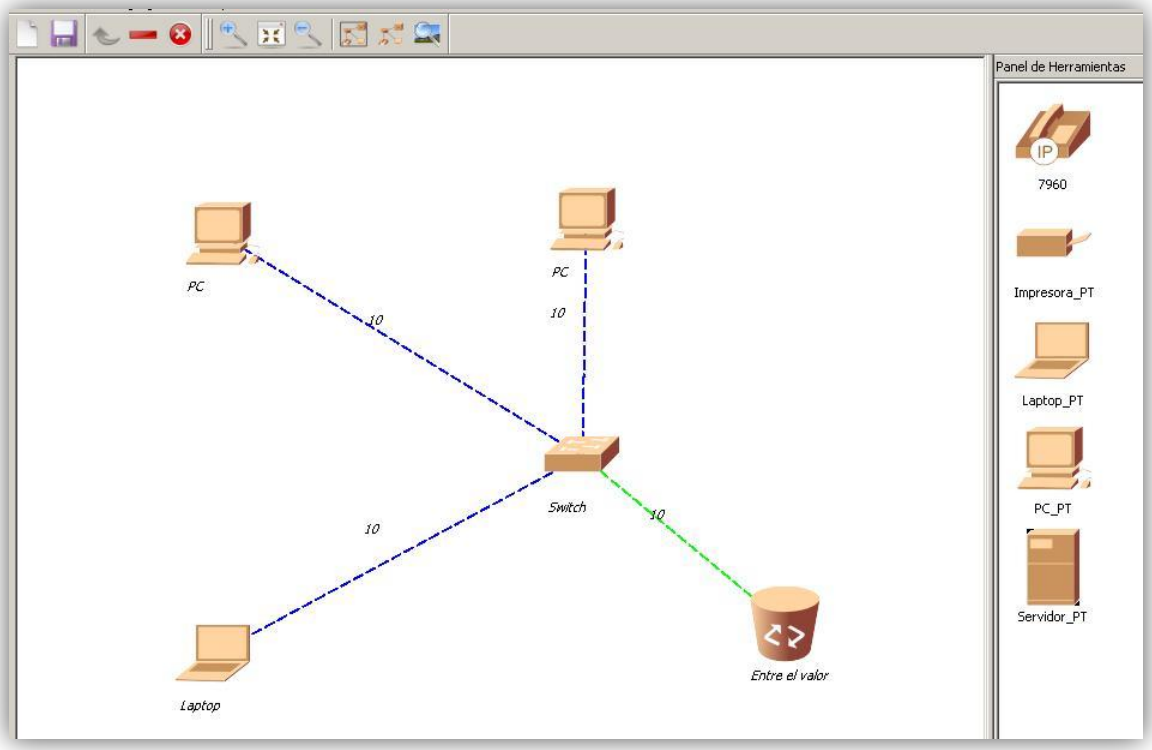

Figura 7 Fase 2 L.V Diseño e Instalación de una red local (LAN).

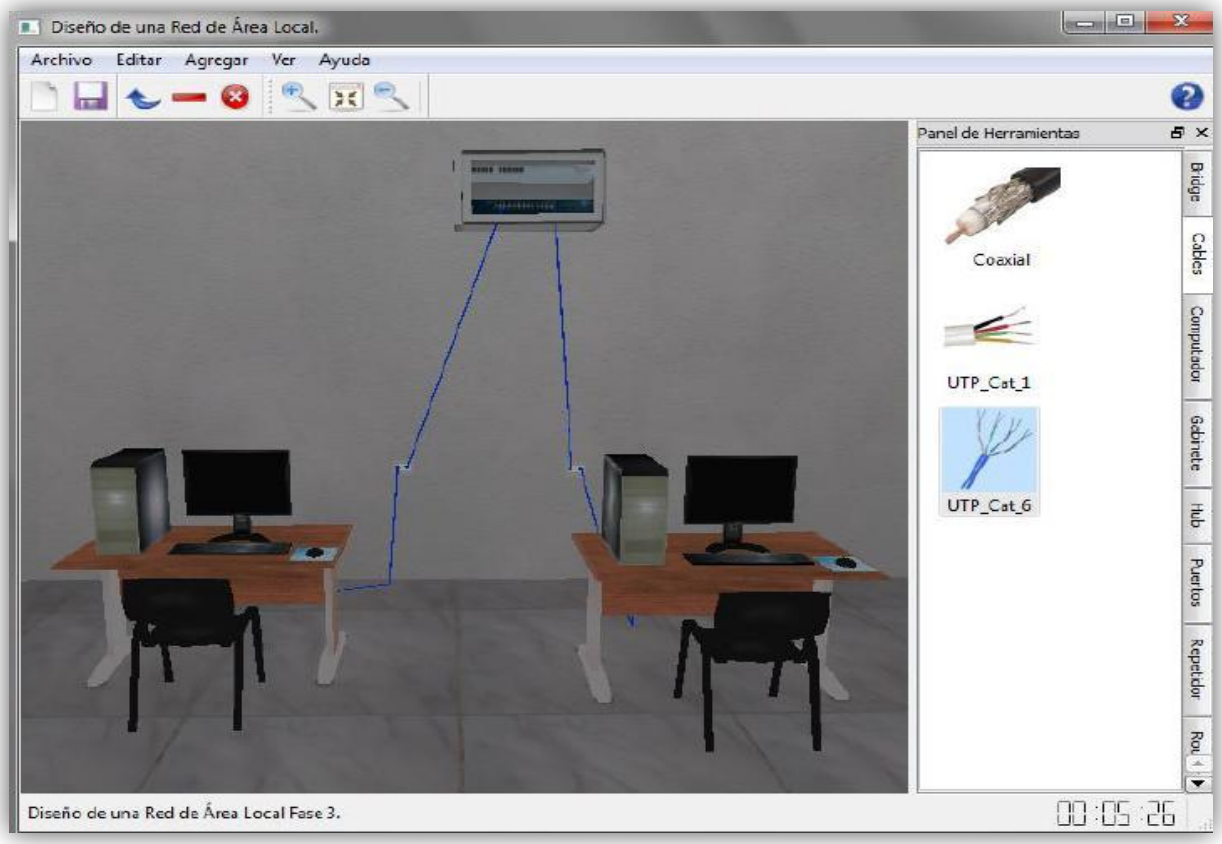

Figura 8 Fase 3 L.V Diseño e Instalación de una red local (LAN).

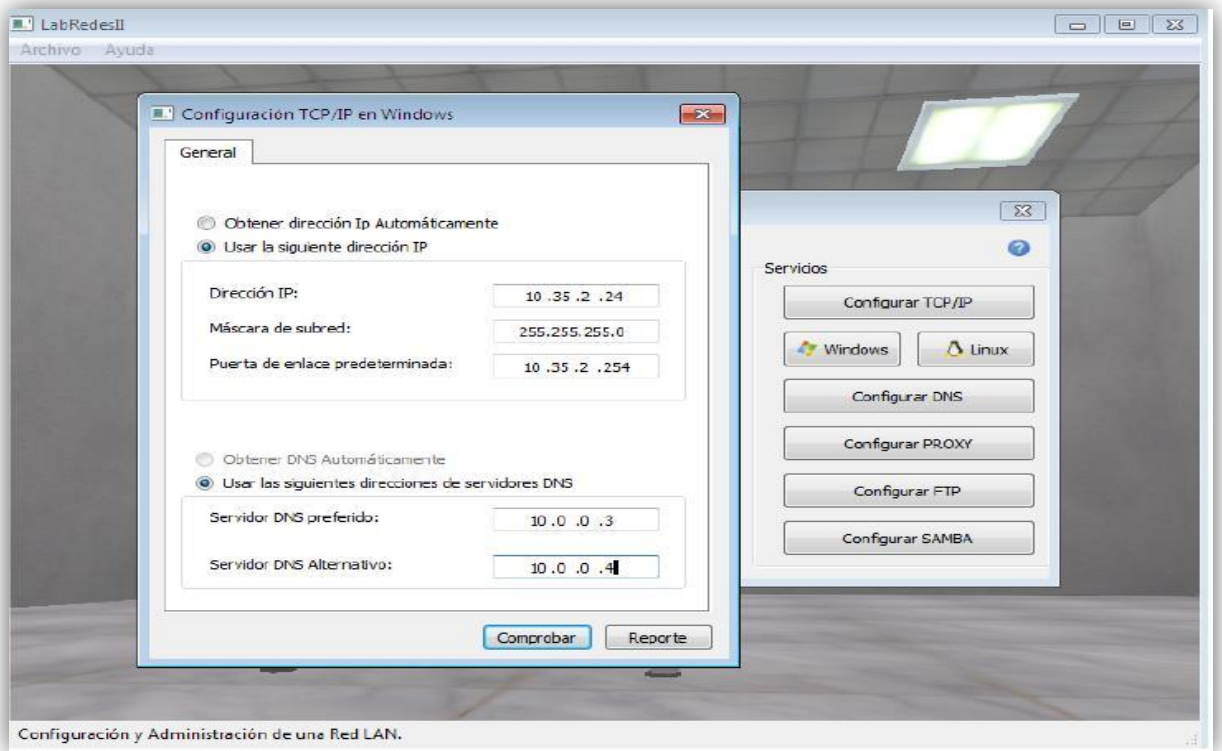

Figura 9 L.V Administración y Configuración de una red local (LAN).

Figura 10 Juegos que usan Lua.

Figura 11 Uso de Lua en Sistemas Operativos.