Universidad de las Ciencias Informáticas

Facultad 1

Herramienta de autor para el diseño de entornos tridimensionales.

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

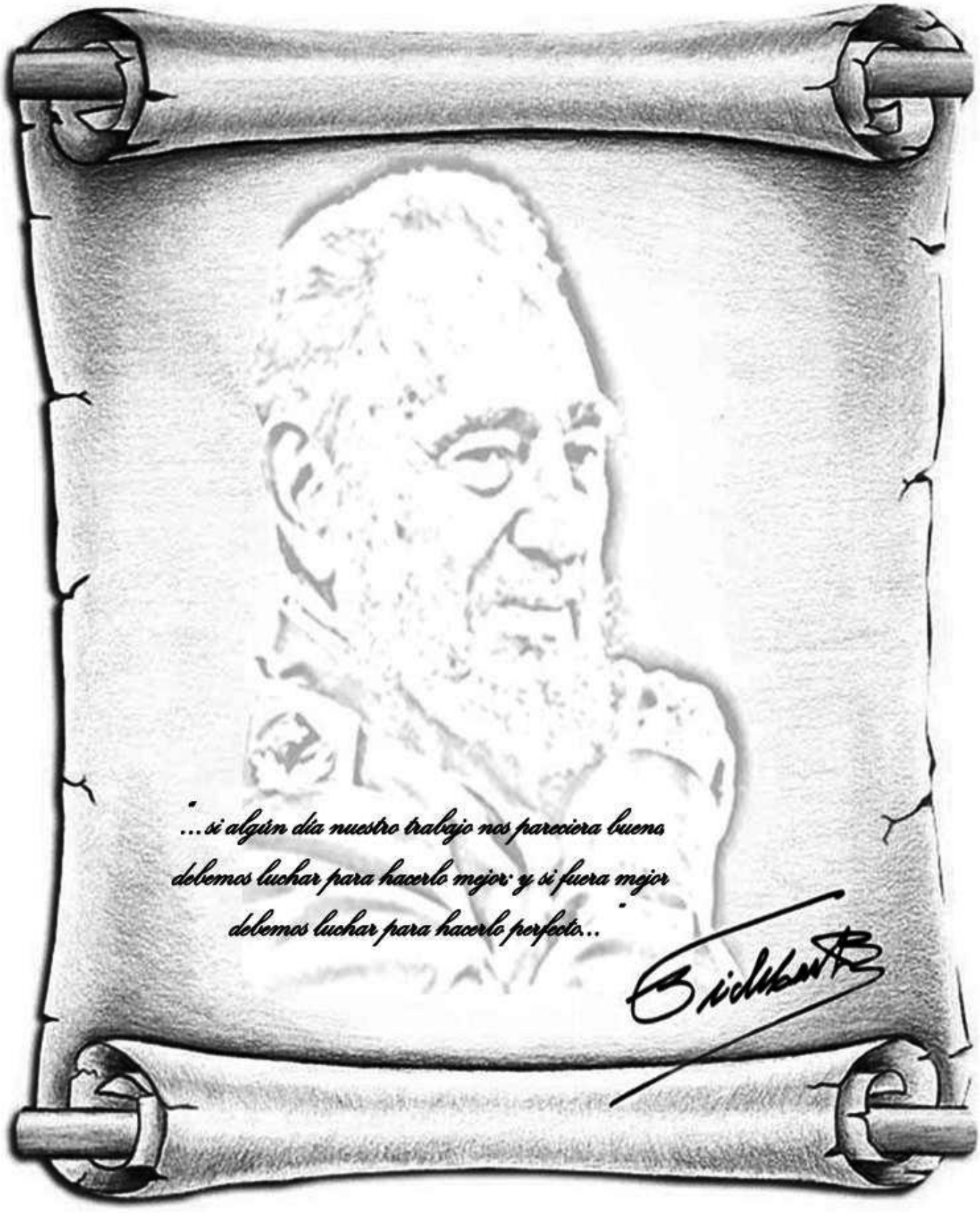
Autor: Fernando Núñez Rocío

Tutores: MSc. Dunia Suárez Ferreiro

Ing. Elizabeth Reinosa Aliaga

La Habana, 2013

Pensamiento



Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al Centro de Informatización Universitaria de la Universidad de las Ciencias Informáticas, para que haga el uso que estime pertinente con este trabajo.

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

_____
Fernando Núñez Rocío

_____ _____
MSc. Dunia Suárez Ferreiro ing. Elizabeth Reinosa Aliaga

Agradecimientos

Mamita linda, quiero agradecerte por siempre brindarme tu amor, apoyo, preocupación y sabiduría. Gracias por darme todo en la vida y hacerme la persona que soy hoy. Eres lo más grande que tengo y espero que lo sepas.

A mi abuelito querido "René ", que dios lo tenga en la gloria, por ser un padre para mí y más que eso mi amigo.

A toda mi familia por estar siempre ahí y brindarme siempre su apoyo incondicional, especialmente a mi tía Casilda y a mi abuela Argelia.

A mi papá y a mi hermanita que aunque estén lejos de mí siempre me han brindado todo su apoyo y han estado al tanto de lo que sucede conmigo.

A mi lindísima ex novia que aún adoro, Raydelmis y a toda su familia por estar siempre ahí y brindarme su apoyo incondicional en todo momento.

A mis hermanos en la universidad: Alexander (el flaco), Joel, Edito, Danelia, Adriel, Henry, Roberto y Félix.

A esas personas que si no fuera por ellas hoy no sería ingeniero: Alexander (el flaco), Joel, Edito, Danelia, Adriel, Yannier (el pintos).

A mis tutores por la comprensión y el apoyo que me han dado.

A la profesora Yenisleydi Cariaga Cristo por toda su ayuda.

A todos los que de una forma u otra han contribuido a la realización de este sueño.

Gracias.

Dedicatoria

A la memoria de mi abuelo querido, que siempre soñó con verme graduado y convertido en un profesional.

A mi madre querida que tanto se esforzó por convertir ese sueño en realidad, por ser siempre mi guía y ejemplo a seguir, por su paciencia y dedicación y por su apoyo incondicional.

Resumen

En la Universidad de las Ciencias Informáticas (UCI), el uso de las Tecnologías de la Información y las Comunicaciones (TIC) es bastante generalizada. Aunque se utilizan como medios de apoyo a la clase y para la auto-preparación de estudiantes y profesores, estas tecnologías pueden ser mejor aprovechadas si se emplean herramientas educativas que permitan apoyar el proceso de enseñanza-aprendizaje (PEA) de las asignaturas de difícil comprensión para los educandos, en particular las pertenecientes a la disciplina Técnicas de Programación de Computadoras (TPC).

Como parte de un proyecto de innovación educativa (Herramienta Educativa sobre Software libre para las asignaturas Programación), en la Facultad 1 se desarrolló un sistema que permite la ejercitación de contenidos a través de juegos didácticos estilo tablero. Este tipo de juegos no motiva completamente a los estudiantes a partir del tercer año de la carrera, por lo que es necesario incluir nuevos tipos de entornos con un carácter tridimensional, específicamente en juegos de estrategia.

La presente investigación propone el desarrollo de una herramienta de autor para el diseño de entornos tridimensionales que se incluirán en juegos didácticos para la ejercitación de contenidos. Además se exponen los principales artefactos diseñados durante el proceso de desarrollo, así como la verificación del correcto funcionamiento del sistema a partir de los resultados obtenidos con las pruebas realizadas.

Palabras clave: aplicaciones Web, herramientas de autor, entornos tridimensionales

Abstract

At the University of Informatics Sciences (UCI), the use of Information Technology and Communications (ICT) is quite widespread. Although used as a means of support for the classroom and for self-preparation of students and teachers, these technologies can be better exploited if used educational tools that can support the teaching-learning process (PEA) of the subjects that are hard to understand, particularly those belonging to the discipline of Computer Programming Techniques (TPC).

As part of an educational innovation project (Educational Tool on Free Software for programming courses), Faculty 1 developed a system that allows to exercise the content through didactic games style board. This type of games does not motivate students completely, so it is necessary to include new types of environments with a three dimensional character, specifically in strategy games.

This research proposes the development of an authoring tool for the design of three dimensional environments that will be include in the didactic games to exercise the content. It also presents the main artifacts designed during the development process, as well as verifying the correct operation of the system from the results obtained from the tests.

Keywords: web application authoring tools, three-dimensional environments

Índice

Índice de Figuras

Índice de Tablas

Introducción

En la actualidad, el auge de las Tecnologías de la Información y las Comunicaciones (TIC) ha tenido su impacto en diferentes esferas sociales, siendo el sector educacional uno de los beneficiados. Este desarrollo ha dado paso al surgimiento y evolución de nuevas formas de acceso al conocimiento. Un ejemplo indudable de esto lo constituyen las páginas web, las cuales presentan la mayor parte de la información a través de textos, imágenes y multimedia. En el ámbito educacional de manera creciente se utilizan herramientas educativas de apoyo al proceso de enseñanza aprendizaje, resultando conveniente la integración de la tercera dimensión debido a que el mundo real es tridimensional, por lo que al reducir el entorno web a sólo dos dimensiones se pierde información que resulta valiosa para una simulación del mismo.

Con el paso de los años, estudiantes y profesores de todos los niveles han afrontado un problema común: algunas de las áreas de la educación son difíciles de asimilar y de enseñar. En busca de una solución a este problema se ha concentrado el interés en una importante rama de la computación, la realidad virtual. Esta nueva tendencia tiene significativas aplicaciones en la educación pues hay indicios de que estimula de manera considerable el proceso de aprendizaje. Este estímulo ocurre a través del efecto de inmersión que genera el ordenador y gracias al cual los estudiantes pueden interactuar completamente con un ambiente artificial utilizando los sentidos del oído, la vista y el tacto. Una característica importante de los escenarios virtuales es que brindan a sus usuarios la posibilidad de interactuar como participantes activos. [1]

Al analizar el proceso de enseñanza aprendizaje en la Universidad de las Ciencias Informáticas, específicamente en la materia de Programación, se aprecia que en varios casos las clases impartidas por los profesores no son aprovechadas de la mejor forma posible por parte de los estudiantes. Es válido aclarar que la disciplina Técnicas de Programación de Computadoras (TPC) pertenece al núcleo básico-específico de la especialidad y las asignaturas que la componen abordan contenidos esencialmente complejos, los cuales constituyen la causa de insatisfactorios resultados docentes. Estas asignaturas exigen del estudiante un nivel de pensamiento lógico avanzado ya que deben desarrollar habilidades para el diseño e implementación de algoritmos y para ello se requiere una buena base, en niveles y asignaturas previas, que permita asimilar más fácilmente los nuevos contenidos. [2]

Según Suárez Ferreiro, a partir de entrevistas realizadas a directivos y profesores de las asignaturas mencionadas anteriormente, de los resultados obtenidos en los controles y de la observación diaria del comportamiento en las clases, se llegó a la conclusión que existen estudiantes que no se sienten motivados ni muestran interés por la asignatura de Programación. Esto se manifiesta a partir de la poca

atención y participación en clases, la no realización de tareas, la obtención de bajas notas en las evaluaciones sistemáticas, así como algunas ausencias injustificadas a los turnos de clases.[2]

Teniendo en cuenta los elementos antes expuestos, un colectivo de profesores de la Facultad 1 de la UCI, han buscado nuevas alternativas para incentivar en los educandos el interés por el estudio. Como parte de estas estrategias se desarrolló una aplicación llamada SMProg, que permite la ejercitación de contenidos de manera didáctica. Esta herramienta cumple con las expectativas de un grupo de estudiantes (primero y segundo años), pero los juegos estilo tableros (que son los que brinda el software SMProg) no motivan de igual forma a estudiantes de niveles superiores (tercero, cuarto y quinto), ya que los intereses de estos están más centrados en los juegos de estrategias con carácter tridimensional.

La primera versión de SMProg presenta varias deficiencias, entre las que se pueden mencionar:

- Solo se incluyen en esta versión preguntas de selección múltiple, de selección única, de verdadero y falso y de completar, no es posible implementar algoritmos.
- En la etapa de diseño no se facilita al profesor la especificación de niveles de ayuda para cada problemática (el estudiante no cuenta con niveles de ayuda según su capacidad para dar respuesta a las interrogantes planteadas).
- Presenta una arquitectura de escritorio, o sea, la aplicación debe estar instalada en la computadora que se va a utilizar, siendo esta una de las mayores desventajas.[2]

Partiendo de la situación problemática existente, se identifica el siguiente **problema de la investigación:** ¿Cómo facilitar la creación de recursos educativos para la ejercitación de contenidos en la disciplina Técnicas de Programación de Computadoras, en la Universidad de las Ciencias Informáticas?

Se define como **objeto de estudio**, el proceso de desarrollo de aplicaciones tridimensionales, enmarcándolo en el **campo de acción** de las herramientas de autor para diseños tridimensionales.

Partiendo de dicha premisa se define como **objetivo general** de la investigación: Desarrollar una herramienta de autor que permita el diseño de entornos tridimensionales con la finalidad de ser utilizados en juegos didácticos para la ejercitación de contenidos en la disciplina Técnicas de Programación de Computadoras en la Universidad de las Ciencias Informáticas.

El objetivo general está desglosado en los siguientes **objetivos específicos**:

1. Construir el marco teórico de la investigación relacionado con el desarrollo de herramientas de autor para el diseño de entornos tridimensionales.

2. Desarrollar una arquitectura que facilite el diseño de entornos tridimensionales.
3. Verificar el correcto funcionamiento de la Herramienta de autor para el diseño de entornos tridimensionales.

Para cumplir con el objetivo propuesto se desarrollan las siguientes **tareas de investigación**:

1. Caracterización del estado del arte acerca de las herramientas de autor libres para el diseño de entornos tridimensionales.
2. Identificación de los recursos informáticos a desarrollar para la implementación de la herramienta de autor.
3. Definición de la arquitectura del sistema.
4. Desarrollo de la herramienta de autor para el diseño de entornos tridimensionales.
5. Realización de las pruebas correspondientes al software.
6. Diseño de al menos cinco entornos tridimensionales y almacenarlos en formato xml.

Para un mejor desarrollo de la investigación se usaron los siguientes **métodos científicos**:

- **Del nivel teórico:**
  - Método histórico-lógico: Su utilización permitió investigar los antecedentes del objeto de estudio, además de la evolución de diferentes sistemas desde su surgimiento hasta la actualidad.
  - Método de modelación: Mediante este método se podrá representar toda la información obtenida hasta el momento mediante diagramas, ya sea de clases o de diseño, los cuales permitirán reflejar las relaciones y cualidades de la aplicación Web que se quiere desarrollar.
  - Método analítico-sintético: Se utilizó durante la búsqueda y análisis de conceptos y documentos, que permitieron la obtención de los elementos necesarios, de una manera resumida, para la comprensión del objeto de estudio.
- Del nivel empírico
  - Método de la observación: Este método se usó para, observar el funcionamiento de herramientas y tecnologías que se podían ser útiles en el desarrollo del Trabajo de Diploma, para determinar las características, ventajas y desventajas de las mismas y la posibilidad

de su utilización.

En el presente trabajo se realiza la propuesta de una herramienta educativa para el diseño de entornos tridimensionales. Esta herramienta facilita la creación de juegos didácticos para la ejercitación de contenidos de la disciplina de TPC. Además, es interactiva a través de componentes visuales, lo cual facilita el trabajo de diseño de los entornos tridimensionales por parte de los profesores, atendiendo a las necesidades específicas de cada estudiante.

El presente trabajo de diploma está compuesto por un resumen, una introducción, tres capítulos, conclusiones, recomendaciones, así como las referencias bibliográficas y la bibliografía, donde se da cumplimiento a los objetivos planteados en el trabajo. A continuación se describen los principales aspectos abordados en cada uno de los capítulos:

Capítulo 1: Fundamentación teórica relativa a los entornos tridimensionales.

Se tratan conceptos y definiciones fundamentales, se realiza un estudio de herramientas y tecnologías que pueden ser útiles para el desarrollo del Trabajo de Diploma, además se definen los lenguajes y la metodología que se utilizará para la realización de la investigación.

Capítulo 2: Características de la Herramienta de autor para el diseño de entornos tridimensionales.
Este capítulo tiene como objetivo principal describir las características fundamentales del sistema, para dar solución a la problemática existente. Luego se exponen los requerimientos funcionales y no funcionales a cumplir por la aplicación. Además es posible apreciar los diagramas de casos de uso del sistema y la descripción de cada uno de ellos así como el diagrama de clase y de colaboración.

Capítulo 3: Implementación y prueba de la Herramienta de autor para el diseño de entornos tridimensionales.
En este capítulo se ilustra la manera en que los elementos de diseño se implementan en términos de componentes. Además se realizan las comprobaciones pertinentes para verificar el correcto funcionamiento del software.

Capítulo 1. Fundamentación teórica relativa a los entornos tridimensionales.

En el presente capítulo se detallan las tendencias existentes en el desarrollo de aplicaciones web con tecnologías tridimensionales (3D) y las ventajas que brindan en el campo de la educación. También se proporciona una breve explicación de algunos conceptos que servirán de guía para un mejor entendimiento de la investigación. Se realiza un estudio acerca de diferentes herramientas y métodos que se utilizan en el diseño de entornos 3D. Además, se especifica la metodología a utilizar, así como los lenguajes de programación y las herramientas que se proponen para el desarrollo del sistema.

1.1 Conceptos relacionados

1.1.1 Mundos virtuales

Un mundo virtual es un tipo de comunidad virtual en línea que simula un mundo o entorno artificial inspirado o no en la realidad, en el cual los usuarios pueden interactuar entre sí a través de personajes o avatares, y usar objetos o bienes virtuales. Se puede decir que hay tres tipos básicos de mundo virtuales que pueden existir por separados como también mezclados entre ellos:

- Mundo Muerto: es aquel en el que no hay objetos en movimiento ni partes interactivas, por lo cual sólo se permite su exploración. Suele ser el que se ve en las animaciones tradicionales, en las cuales las imágenes están pre calculadas y producen una experiencia pasiva.
- Mundo Real: es aquel en el cual los elementos tienen sus atributos reales, de tal manera que si se mira un reloj, marca la hora. Si se pulsan las teclas de una calculadora, se visualizan las operaciones que esta realiza y así sucesivamente.
- Mundo Fantástico: es el que permite realizar tareas irreales, como volar o atravesar paredes. Es el típico entorno que se visualiza en los videojuegos, pero también proporcionan situaciones interesantes para aplicaciones serias, como puede ser observar un edificio volando a su alrededor o introducirse dentro de un volcán. [3]

Se puede decir entonces que, un mundo virtual es un espacio digital que simula el entorno físico de la realidad y que puede ser utilizado para llevar a cabo algunas actividades realizadas por el hombre, entre ellas la educación. Un sistema es educativo cuando se utiliza para desarrollar una actividad planificada, intencionada y estructurada, que pretende lograr el desarrollo integral del individuo. La educación implica aprendizaje, pero el aprendizaje estricto (adquisición de conocimientos) no siempre implica educación.

Los mundos virtuales pueden ser incluidos en la educación de forma exitosa. Para facilitar su creación pueden utilizarse herramientas de autor.

1.1.3 Herramientas de autor en la educación

Una herramienta de autor en la educación es un programa de ordenador diseñado para facilitar, a profesores no especializados en informática, la creación de material educativo. En cierta manera, evita la complejidad de la programación tradicional y permite la creación de "lecciones electrónicas" a cualquier instructor interesado y que esté dispuesto a dedicar unas cuantas horas a actualizar sus conocimientos y herramientas didácticas. [4]

Hasta ahora, algunos profesores, pese a manejar con frecuencia herramientas informáticas: correo electrónico, bases de datos y hojas de cálculo, tratamiento de textos, entre otros consideraban que la creación de su propio material educativo multimedia escapaba de su capacidad por falta de formación en programación de aplicaciones, principalmente, y por tratarse de un proceso complejo que conlleva altos costes económicos, largos tiempos de desarrollo, y de difícil accesibilidad para muchas instituciones educativas. Esta es la situación que la aparición de estos sistemas llamados de autor intenta corregir. [4]

Las herramientas de autor para la educación permiten:

- Realizar software sin apenas conocimientos de programación, ya que el profesor puede gestionar los contenidos ignorando por completo un lenguaje de programación.
- Facilidad de uso.
- Máxima contextualización.
- Adaptación a los alumnos.
- Integración curricular.

1.1.4 Aplicaciones Web

Se denomina aplicación web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un Servidor web a través de Internet o de una intranet mediante un navegador [6]. En otras palabras, es una aplicación (Software) que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador.

Las aplicaciones web son populares debido a lo práctico del navegador web como Cliente ligero, a la independencia del Sistema operativo, así como a la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales. [5]

Ventajas de las aplicaciones web

Desde el punto de vista del usuario, no es necesario instalar ningún software adicional que no sea un navegador web. Las actualizaciones las realiza el desarrollador en su servidor y cada vez que un usuario se conecte tendrá la última versión disponible. No ocupa espacio en el disco duro de la máquina ya que se ejecutan a través de la red y no se corre el riesgo de incompatibilidades con los sistemas operativos.

Para un mejor aprovechamiento y utilización del sistema y para que se encuentre disponible en cualquier momento en toda la red universitaria, después de haber reflejado las ventajas del uso de las aplicaciones web; se decide desarrollar una aplicación web en la cual se permita la ejercitación de contenidos de manera didáctica en la asignatura de Programación. Para ello es necesario contar con una herramienta que facilite el diseño de los entornos tridimensionales a utilizar en los juegos. [5]

1.1.5 Servidor web

Apache Tomcat Como servidor Web de aplicaciones fue escrito en Java que es el lenguaje de programación propuesto para implementar la aplicación. Funciona en cualquier sistema operativo que tenga instalado la Máquina Virtual de Java, es usado como servidor Web autónomo en entornos con alto nivel de tráfico y alta disponibilidad, es un software de código abierto, funciona como un contenedor de servlets (objetos que corren dentro del contexto de un servidor de aplicaciones y extienden su funcionalidad), además implementa las especificaciones de los servlets y de Java Server Pages (JSP) de Sun Microsystem. [6]

1.2 Herramientas y tecnologías para el desarrollo de entornos tridimensionales

En la actualidad existen numerosas aplicaciones que proporcionan a los usuarios nuevas maneras de crear entornos tridimensionales. Muchos de estos programas necesitan licencias del proveedor para ser usados, además de estar regidos por una plataforma de desarrollo, lo cual limita su uso. Otro de los aspectos a tener en consideración es que la mayoría de estas aplicaciones están diseñadas como aplicaciones de escritorio, lo que impone trabas para usarlas en entornos de desarrollo web.

A continuación se realizará un estudio de algunas herramientas para el desarrollo de entornos tridimensionales, donde quedarán expuestas sus principales características, ventajas y desventajas, así como también se valorará la posibilidad de usarlas como base para desarrollar una herramienta de autor que facilite el diseño de entornos tridimensionales y que pueda integrarse a SMProg. Estas herramientas se listan a continuación:

1.2.1 Interfaz de Programación de aplicaciones (API) para el desarrollo de entornos tridimensionales

Java3D: Es un Interfaz de Programación de Aplicaciones (API) orientado a objetos para el lenguaje Java, permite la confección de aplicaciones tridimensionales. Las que posibilitan construir objetos 3D, visualizar objetos 3D y controlar el comportamiento de los objetos 3D.

Java3D es una interfaz que encapsula programación de gráficos usando un verdadero concepto orientado a objetos. Aquí, un escenario se construye utilizando un grafo de escena que es una representación de los objetos que tienen que ser mostrados. En este grafo se describen la geometría de los objetos tridimensionales y sus propiedades (colores, texturas, movimientos, entre otros), su ubicación dentro de la escena, orientación, fuentes de luz y lugar en donde está situado el observado. [7]

Ventajas:

- Al ser una extensión de la API de Java, es independiente de la plataforma.
- Es un API de código abierto.
- Interfaz de alto nivel.
- La visualización se basa en las API OpengGL y DirectX.
- La escena se construye creando un grafo.
- La aplicación 3D puede ser un applet.

Desventajas:

- El API oculta detalles de cómo se visualiza la escena.
- Los componentes de Java3D son pesados.
- No es tan rápido como una aplicación en código nativo en OpenGL o DirectX.

WebGL: Es una serie de estándares para manejar gráficos 3D por medio de aceleración de hardware en los navegadores web sin necesidad de plugins. El sistema funciona a través de JavaScript y su capacidad para comunicarse con OpenGL ES 2.0 es muy elevada. [8]

Es un estándar que es apoyado por Mozilla (Firefox), Google y Opera. Centra su funcionamiento a través de WebKit, que no es más que el motor que se encuentra detrás de los navegadores Safari y Chrome. Su potencial está encaminado a la implementación de juegos 3D en los navegadores, además de desarrollar aplicaciones educativas haciendo uso de los estándares para gráficos 3D de OpenGL. Su principal desventaja radica en que se encuentra actualmente en desarrollo, por lo que su utilización es muy limitada.

Sandy3D: Es un motor disponible en tres versiones: AS2, AS3 y haXe. Es conocido por su gran empleo en la creación de aplicaciones web 3D, debido a que tiene un renderizado excelente para escenas interiores y contiene avanzados efectos de iluminación dentro de los que se encuentra Phong, Gouraud y CelShading. Es compatible con Adobe Flash Player desde su versión 7 hasta la 10, además de tener compiladores que manejan varios tipos de formatos 3D como son: Collada, 3DS, ASE, MD2. Su principal desventaja es que para manejar transparencia de caras y el agregado de texturas de mapa de bits (bitmap) es necesario utilizar un volumen considerable de código, además de ofrecer una baja calidad de la escena en cuanto a texturizado se trata. De manera general los modelos construidos tienen una escasa definición debido a que poseen pocos polígonos. Este motor pertenece a una comunidad libre, pero en el año 2008, tras un cambio de la perspectiva en su desarrollo, fueron agregados módulos con restricciones monetarias y de licencia para su uso. [9]

Google O3D: Es una API de código libre basada en JavaScript para crear mundos 3D. La API de O3D maximiza el rendimiento de la GPU para la programación del lenguaje de sombreado de forma directa, proporciona un plug-ins para el navegador que añade capacidades gráficas dentro del motor de renderizado. Su principal desventaja consiste en que aún se encuentra en desarrollo. [10]

1.2.2 Herramientas de autor para el diseño de entornos tridimensionales

Blender: Es un programa dedicado a la creación de imágenes y animaciones en tercera dimensión. Ofrece una amplia gama de opciones en lo referente a modelado, renderización, animación, postproducción y creación de productos 3D, destacándose un motor de juegos 3D integrado. [11]

A continuación se especifican las principales ventajas que posee:

- Multiplataforma, libre, gratuito y con un tamaño de origen realmente pequeño comparado con otros paquetes de 3D, dependiendo del sistema operativo en el que se ejecuta.
- Motor de juegos 3D integrado.
- Junto a las herramientas de animación se incluyen cinemática inversa, deformaciones por armadura o cuadrícula, vértices de carga y partículas estáticas y dinámicas.

Dentro de las desventajas se tienen que:

- La interfaz del software es un tanto “diferente” cuando la comparamos con otros programas, por lo que no es fácil de entender a simple vista todo el abanico de características que ofrece, ni siquiera para los que ya conocen programas de este tipo.

Unity3D: Es una herramienta de creación de entornos 3D, aplicaciones interactivas, aplicaciones web y juegos en 3D. Está basado en JavaScript y se debe descargar el plugins para poder ver el contenido en cualquier browser. [12]

Dentro de las principales ventajas se tiene que:

- No afecta el rendimiento de la máquina independientemente de las características de la misma.
- Se puede programar en C#, JavaScript, .NET y se puede comunicar mediante sockets al flash player, por lo que se puede mezclar con tecnologías Adobe.
- Para el desarrollo de proyectos se puede usar el editor de Unity, aunque también cualquier otro editor para JavaScript, llámese Dreamweaver, Eclipse, etc. ya que las librerías Unity se integran a un proyecto JavaScript.
- Cuenta con un editor exclusivo para 3D con el cual se puede testear sin necesitad de compilar, siendo de gran ventaja porque permite ver cómo se comporta el entorno.

Como desventajas se tiene que:

- La documentación de las clases no es muy clara y algunas veces hay que perder mucho tiempo experimentando para entender el comportamiento de una clase en particular.
- Se puede descargar la versión Trial para desarrollar, lo demás es bajo licencia.

1.2.3 Juegos didácticos aplicados a la educación

Jclic: Es un proyecto de código abierto formado por un conjunto de aplicaciones informáticas que sirven para realizar actividades educativas. Está desarrollado sobre la plataforma Java por lo que funciona en diversos entornos y sistemas operativos. El formato para almacenar los datos de las actividades es .xml. Este módulo está formado por cuatro aplicaciones, una herramienta de autor para diseñar y publicar asignaturas; una herramienta que permite realizar las actividades sin tener conexión a internet o con el EVA relativo; un applet que permite “incrustar” las actividades de JClic en una página Web (útil para poder interactuar directamente con un Entorno Virtual de Aprendizaje(EVA)) y una herramienta para obtener reportes sobre los resultados obtenidos por los estudiantes en las diferentes actividades realizadas. Su alcance es para el nivel primario y secundario ya que las actividades que se pueden diseñar son del tipo de rompecabezas, asociaciones, ejercicios de texto y palabras cruzadas. Este tipo de juegos didácticos está bien diseñado para estos niveles de enseñanza pero para el nivel universitario se requiere de un alcance mayor. [13]

Squake: Es un entorno de desarrollo basado en el lenguaje orientado a objetos SmallTalk, el cual ha sido implementado sobre estándares libres [14]. Permite el diseño de Juegos de diversos tipos (historias, sencillos videojuegos, animaciones, asociaciones de textos), así como la creación de multimedias y simulaciones interactivas (tutoriales en forma de historias). Squeake fue diseñado para niños entre 6 y 16

años, pero también ha sido utilizado en otras enseñanzas. Incluye varios proyectos de desarrollo como Scratch, Alice y Croquet. En el curso 2008-2009 en la UCI se utilizó el Scratch como apoyo para el diseño de algoritmos como parte de las actividades del curso de nivelación Introducción a la algoritmización durante el curso 2008-2009.

1.2.4 Conclusiones del estudio realizado

Con el estudio realizado se llega a la conclusión de que a pesar de que las mismas facilitan o hacen posible la gestión de entornos tridimensionales, poseen varias desventajas o inconvenientes que no resultan factibles para el desarrollo de la aplicación que se describe en la presente investigación. Son varios los criterios que pueden señalarse: algunos software necesitan licencias del proveedor para ser usados y están regidos por una plataforma de desarrollo, lo cual limita su uso. Además algunas de estas herramientas no fueron pensadas para utilizarse con fines educativos por lo que su interfaz, al estar diseñada para usuarios avanzados, impide que el profesor interactúe fácilmente con el software. Las herramientas educativas analizadas son para temáticas específicas o para la enseñanza en otros niveles.
Otro de los aspectos a tener en consideración y una de las principales deficiencias encontradas es que muchos de estos sistemas están diseñados como aplicaciones de escritorio, lo que impone trabas a la hora de usarlos en entornos de desarrollo web.

1.3 Proceso Unificado de Desarrollo de Software (RUP)

Las metodologías de desarrollo de software surgen por la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental para desarrollar un producto de software. Se puede decir que un proceso de desarrollo de software es el conjunto de actividades destinadas a guiar los esfuerzos de las personas implicadas en un proyecto de desarrollo.

El Proceso Unificado de Desarrollo de Software (RUP), nombrado así por sus siglas en inglés (Rational Unified Process), es la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Esta no presenta pasos que estén firmemente establecidos, sino que cuenta con un conjunto de flujos de trabajo adaptables al contexto y necesidades de cada organización. RUP es una de las metodologías tradicionales, que se centra en la definición detallada de los procesos y tareas a realizar. Además propone una extensa documentación ya que pretende prever todo de antemano.

Esta metodología tiene como meta, asegurar que la producción de software sea de alta calidad y que cumpla con las necesidades de los usuarios dentro de las restricciones. RUP cuenta con una forma organizada y disciplinada de asignar tareas y responsabilidades. Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por casos de uso.

A continuación se detallan dichas características:

- **Guiado por Casos de Usos:** Los casos de uso reflejan lo que los futuros usuarios necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos, es decir, los casos de usos representan los requisitos funcionales. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

- **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo y desarrollarlo.

- **Iterativo e Incremental:** Este modelo plantea la implementación del proyecto a través iteraciones. Estas iteraciones se logran dividiendo el proyecto en pequeñas partes con lo cual se pueden definir objetivos por cumplir en cada iteración y así poder ir completando este iteración por iteración. Se tienen varias ventajas, entre ellas se puede mencionar la de tener pequeños avances que son entregables al cliente, el cual puede probar mientras se está desarrollando otra iteración del proyecto, así el mismo va creciendo hasta quedar completo.

RUP divide el desarrollo en 4 fases que detallan su ciclo de vida. La fase de inicio define el alcance del proyecto. La fase de elaboración permite determinar la arquitectura óptima. La fase de construcción, tiene como objetivo lograr la capacidad operacional inicial del producto, o sea, la implementación. La fase de transición, se realiza con el fin de desplegar el resultado del trabajo. [15]

RUP es una metodología configurable para satisfacer diferentes tipos de proyectos. En este caso se le harán algunas modificaciones para ajustarla a las necesidades del proyecto de Innovación educativa, el cual está conformado por un equipo de desarrollo reducido compuesto principalmente por estudiantes, con poca experiencia en el desarrollo de software y de trabajo en equipo. Los roles y responsabilidades están bien determinados y el tiempo disponible para el desarrollo es escaso. Además, no existe un cliente bien definido, las restricciones y requerimientos se van identificando a medida que avanza el desarrollo de la investigación y se cuenta con escasos recursos materiales para el desarrollo.

Por las razones antes expuestas, solo se generarán los artefactos imprescindibles de los que propone RUP en cada una de sus fases, tales como: modelo de casos de uso del sistema, diagramas de colaboración (interacción), de clases, despliegue, componentes y casos de prueba.

1.4 Lenguaje Unificado de Modelado (UML)

UML es un lenguaje que proporciona un vocabulario y una o varias reglas para permitir una comunicación. Este lenguaje se centra en la representación gráfica de un sistema e indica cómo crear y leer los modelos.

UML es un estándar del Grupo de Gestión de Objeto OMG (Object Management Group por sus siglas en inglés) diseñado para visualizar, especificar, construir y documentar software orientados a objetos. El modelado en la construcción de software es de mucha importancia para lograr un mejor entendimiento de lo que se está haciendo. Permite descubrir oportunidades de simplificación y reutilización, ya que facilita los planos de un sistema y pueden ser detallados debido a que la modelación se basa en una representación visual y la combinación de diversos elementos gráficos que permiten la creación de diagramas. UML permite comunicar la estructura de un sistema complejo, y especificar el comportamiento deseado del mismo. [16]

UML estandariza nueve tipos de diagramas para representar gráficamente un sistema desde distintas perspectivas. Este lenguaje es muy utilizado en varias metodologías de desarrollo y presenta una estrecha relación con la escogida en el presente trabajo de diploma, debido a que RUP utiliza UML para la representación visual además de que manipula todos los diagramas propuestos por este.

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones:

- Visualizar: UML permite expresar gráficamente un sistema de manera que otro lo pueda entender.
- Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden ser aprovechados para su futura revisión. [17]

El lenguaje UML es perfecto para el modelado de sistemas orientados a objetos ya que incluye la representación de abstracciones, herencias, polimorfismos, encapsulamientos, envío de mensajes, asociaciones y agregaciones. Permite detectar con facilidad las dependencias y dificultades implícitas del sistema. Se pueden modelar tanto sistemas de software como de hardware.

1.5 Visual Paradigm

Las herramientas Ingeniería de Software Asistida por Computación (CASE) son un conjunto de programas y ayudas que brindan asistencia técnica a analistas, ingenieros de software y desarrolladores para el análisis de requisitos, modelado visual y documentación durante casi todo el ciclo de vida de un proyecto de software. La selección de esta herramienta está estrechamente relacionada con la metodología de software y el lenguaje de modelado a utilizar.

Visual Paradigm para UML es una herramienta fácil de usar, que soporta todo el ciclo de vida del software: análisis, diseño orientado a objetos, construcción, prueba y despliegue. Permite dibujar todos los tipos de diagramas, realizar la conocida ingeniería inversa y generar código y documentación a partir de diagramas

realizados. “Es un software que ofrece a los analistas del sistema todas las herramientas necesarias para capturar y organizar los requisitos”. [18]

Visual Paradigm es capaz de integrarse con diversos entornos de desarrollo integrados (IDE por sus siglas en inglés) como son: NetBeans (de Sun), Jdeveloper (de Oracle), Eclipse (de IBM) y JBuilder (de Borland). Posee ingeniería inversa para: JAVA, .NET, XML e Hibernate. Visual Paradigm se ha convertido en una de las herramientas de modelado más usadas para el desarrollo de sistemas en plataformas libres, ya que aunque posee una licencia comercial, se puede obtener una versión gratuita (licencia para Community Edition).

Visual Paradigm como herramienta CASE ofrece:

- Entorno de creación de diagramas para UML 2.1.
- Diseño centrado en casos de uso y enfocado al negocio lo cual permite generar un software de mayor calidad.
- Uso de un lenguaje estándar, común para todo el equipo de desarrollo, lo cual facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código permanecen sincronizados durante todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones.
- Disponibilidad de integrarse a los principales IDE’s.
- Disponibilidad en múltiples plataformas tanto en sistemas operativos Windows como en las distribuciones de GNU\Linux”. [18]

Se utiliza esta herramienta pues permite el modelado orientado a objetos y basado en el Lenguaje Unificado de Modelado (UML); es multiplataforma, posee gran disponibilidad de integración con varios entornos de desarrollos integrados que están bajo licencias de software libre como NetBeans. Es sencilla de instalar y fácil de utilizar y actualizar.

1.6 Lenguaje de programación

“Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis que pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes”. [19]

Para el desarrollo de esta aplicación se utiliza el lenguaje de programación Java, creado por Sun Microsystems. Una de las características más importantes es que los programas “ejecutables”, creados por el compilador de Java, son independientes de la arquitectura. Se ejecutan indistintamente en una gran variedad de equipos con diferentes microprocesadores y sistemas operativos. Permite escribir Applets (pequeños programas que se insertan en una página HTML) y se ejecutan en el ordenador local. Se pueden escribir aplicaciones cliente/servidor, aplicaciones distribuidas en redes locales y en Internet etc.

Es fácil de aprender y está bien estructurado.

A continuación se describen otras características importantes del lenguaje Java:

- Es intrínsecamente orientado a objetos.
- Funciona perfectamente en red.
- Aprovecha características de la mayoría de los lenguajes modernos evitando sus inconvenientes, en particular los del C++.
- Presenta gran cantidad funcionalidades gracias a sus librerías (clases).
- No tiene punteros manejables por el programador, aunque los maneja interna y transparentemente.
- El manejo de la memoria no es un problema, la gestiona el propio lenguaje y no el programador.
- Genera aplicaciones con la menor cantidad de errores posibles.

Se puede concluir que Java es el lenguaje más apropiado para desarrollar la herramienta, debido a la portabilidad de su plataforma, que está basada principalmente en que el compilador genera un código binario conocido como *bytecode* el cual es interpretado por la Máquina Virtual de Java (JVM por sus siglas en inglés: *Java Virtual Machine*), y esta a su vez se encarga de ejecutar dicho código. Esto permite que un programa escrito en Java pueda ser ejecutado en cualquier ordenador independientemente de la plataforma que este use, ya sea Windows o Linux. Se puede incrustar en otros programas en forma de applets o componentes.

Un applet es un componente de Java que se ejecuta en el contexto de otro programa. El más común de los ejemplos es la incrustación de estos en los navegadores web. El applet a diferencia de otros programas no puede ejecutarse de manera independiente, sino en un contenedor. Permite también ejecutar el archivo .jar (creado luego de la compilación de la aplicación) en cualquier plataforma.

1.7 NetBeans como Entorno de Desarrollo Integrado (IDE)

NetBeans es un entorno de desarrollo integrado (IDE) de código abierto creado por Sun Microsystems en junio del 2000. Es una herramienta que permite a los programadores escribir, depurar y ejecutar programas. Fue hecho principalmente para Java, pero puede servir para cualquier otro lenguaje de programación.

NetBeans es un producto libre y gratuito sin restricciones de uso, lo que lo hace conveniente para el desarrollo del producto final de la presente investigación. Permite la creación de aplicaciones tanto de escritorio como web. Es posible su uso en diversos sistemas operativos como: Windows, GNU\Linux, Solaris, Mac OS, lo que lo convierte en un IDE multiplataforma.

NetBeans proporciona una estructura para los proyectos que se crean, proponiendo un esqueleto para la organización del código. Permite la integración de los lenguajes de programación web como son HTML, JavaScript y CSS. Además de la integración con disímiles marcos de trabajo. [20]

1.8 OpenGL como biblioteca gráfica

OpenGL es la biblioteca gráfica 3D por excelencia, puede utilizarse en casi todos los Sistemas Operativos. Desarrollada originalmente por Silicon Graphics Incorporated (SGI), es una biblioteca de gráficos que ofrece al programador una API multilenguaje y multiplataforma. OpenGL se propone para el empleo de hardware informático diseñado y optimizado para la visualización y manipulación de gráficos 3D, soportando iluminación y sombreado, mapeado con texturas, animación y otros efectos especiales. [21]

1.9 XML

Lenguaje de Marca Extensible (XML) es un lenguaje para estructurar cualquier tipo de datos, haciendo más fácil para el ordenador el proceso de generar y controlar estos, lo cual lo ha convertido en un estándar para la comunicación entre procesos y plataformas, y especialmente para el desarrollo de frameworks o APIs como Gtk, Qt o Motif, que tienen como objetivo definir las interfaces gráficas de usuario.

Contiene características que aseguran su validez durante mucho tiempo, es independiente de plataforma, no pertenece a ninguna firma concreta de software, soporta internacionalización [22]. Durante el desarrollo de la herramienta que se describe en la presente investigación los archivos xml resultarán de mucha importancia, ya que estos se utilizarán para guardar las escenas diseñadas, y además permitirán al profesor cargar y visualizar escenas 3D.

1.10 Consideraciones finales del Capítulo

En este capítulo se valoró la importancia que reviste para el desarrollo de esta aplicación usar herramientas bajo licencias de software libre, lo cual tuvo una gran influencia a la hora de hacer la selección de las mismas.

Con el estudio de las herramientas homólogas se llega a la conclusión de que a pesar de que las mismas hacen posible la gestión de entornos tridimensionales, poseen varias desventajas o inconvenientes que no resultan factibles para el desarrollo de la aplicación que se describe en la presente investigación, de ahí la necesidad de desarrollar una herramienta para facilitar el diseño de entornos tridimensionales que se integre con SMProg.

Capítulo 2. Características de la Herramienta de autor para el diseño de entornos tridimensionales.

El presente capítulo tiene como objetivo principal describir las características fundamentales del sistema, para dar solución a la problemática existente. Se exponen los requerimientos funcionales y no funcionales a cumplir por la aplicación, el diagrama de casos de uso del sistema con sus descripciones, así como la representación de las clases y los diagramas de colaboración.

2.1 Descripción del problema

En la Universidad de las Ciencias Informáticas existe una herramienta llamada SMProg, que permite la ejercitación de contenidos de manera didáctica. Esta herramienta cumple con las expectativas de un pequeño grupo de estudiantes (primero y segundo), pero los juegos estilo tableros (que son los que brinda el software SMProg) no motivan de igual forma a estudiantes de niveles superiores (tercero, cuarto y quinto), ya que los intereses de estos están más centrados en los juegos de estrategias con carácter tridimensional. Esta aplicación presenta una arquitectura de escritorio, que implica un despliegue específico, siendo esta una de las mayores desventajas.

2.2 Introducción de la aplicación SMProg 2.0

SMProg 2.0 es una aplicación Web que contiene varias herramientas de autor que le permiten al profesor diseñar los entornos de juegos didácticos (ya sean de estilo tablero o 3D), así como el conjunto de problemas asociados a temas de una asignatura específica. Además facilita la ejercitación de contenidos a los estudiantes a partir de la selección de un entorno de juego de interés y de un conjunto de problemas de un tema en particular, las evaluaciones obtenidas por el estudiante durante el juego serán almacenadas en la base de datos, para que el profesor pueda realizar análisis posteriores de las mismas. La presente investigación se centra en la implementación de una herramienta de autor para el diseño de entornos 3D que se integra al sistema SMProg.

2.3 Solución Propuesta

Se propone el desarrollo de una Herramienta de autor que gestione el diseño de entornos tridimensionales en plataformas web, para de esta forma facilitar a los profesores incentivar en los estudiantes de niveles superiores (tercero, cuarto y quinto) el interés por asignaturas de difícil comprensión. Esta aplicación deberá tener en su interfaz visual, además de la pantalla OpenGL (entorno tridimensional), una paleta de componentes donde estarán situados los objetos, para posteriormente ubicarlos en el entorno de forma tridimensional y de esta manera ir conformando la escena. Finalmente el software deberá permitir guardar

la escena creada en un formato xml y abrirla posteriormente cuando se requiera. Esta herramienta deberá estar montada en una plataforma web.

2.4 Levantamiento de requisitos

En la primera fase de la metodología seleccionada, se realiza un levantamiento exhaustivo de requerimientos, que es imprescindible para la realización segura de todo proyecto de software. Los requisitos pueden ser funcionales o no funcionales.

Los primeros son aquellas capacidades o condiciones que la aplicación debe cumplir, ellos definen el comportamiento interno del software (cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica). Los últimos son las propiedades o cualidades que el sistema debe tener, estos requisitos son los que permiten que el sistema sea atractivo, usable, rápido y confiable.

2.4.1. Requisitos funcionales

RF1. Crear escena.

RF2. Editar escena.

RF2.1 Ubicar los objetos 3D en el entorno.

RF2.2 Modificar los objetos 3D.

RF2.3 Eliminar los objetos 3D.

RF3. Guardar escena en un archivo xml.

RF4. Abrir escena desde un archivo xml.

2.4.2. Requisitos no funcionales

- Apariencia o Interfaz Externa.
  - Interfaz amigable para el usuario.
- Portabilidad.
  - El sistema debe ser multiplataforma.
- Legales.
  - Las herramientas de desarrollo deben ser libres o de código abierto. Para las que no lo sean, deben tener las licencias avaladas para su uso.
- Hardware Recomendado para PC Servidor.
  - 40 o más GB de capacidad en el disco duro.

- Microprocesador de 2.4 GHz o superior.
- Memoria RAM de 2 GB o superior.
- Tarjeta de Red de 100 Mbps o superior.

➢ Hardware recomendado para PC Cliente.

- 15 o más GB de capacidad en el disco duro.
- Memoria RAM de 512 MB o superior (256 MB mínimo).
- Instalación de un navegador web.

2.5 Modelo de casos de uso del sistema

En la aplicación que se describe en la presente investigación, el profesor será el único actor del sistema, ya que es el que interactuará con el mismo y a su vez será el iniciador de todos los casos de uso presentes en la aplicación. En la figura No. 1 se observan los casos de usos definidos para representar el flujo de los eventos, los cuáles permitirán obtener un resultado de valor apreciable durante la gestión de la información.

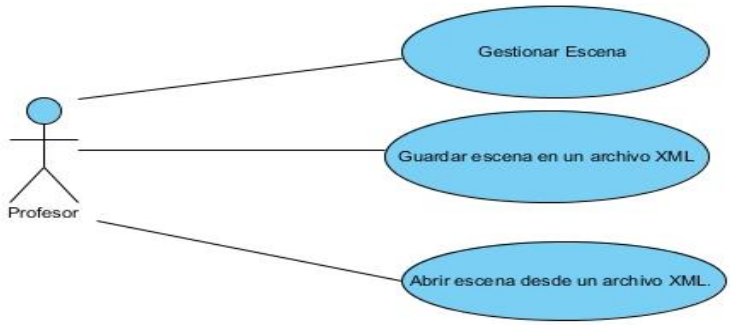


Figura # 1 Diagrama de casos de uso del sistema.

2.5.1 Actores del sistema

A continuación se realiza una breve descripción del actor que interviene en el sistema. En este caso solo interviene el profesor que permite diseñar Juegos Didácticos de carácter tridimensional para la ejercitación y evaluación de contenidos.

Actor	Descripción
Profesor	El profesor es el encargado de diseñar escenas 3D para los Juegos Didácticos.

Tabla # 1 Descripción del actor del sistema.

2.5.2 Descripción de los casos de uso del sistema

CU-1	Gestionar Escena
Actor	Profesor
Resumen	El caso de uso inicia cuando el profesor abre la aplicación y decide crear o editar (Ubicar los objetos 3D en el entorno, Eliminar los objetos 3D, Modificar los objetos 3D) una escena y culmina cuando una o varias acciones se realizan de manera satisfactoria.
Complejidad	Alta
Prioridad	Crítico
Referencias	RF1, RF2
Precondiciones	La escena debe estar previamente creada.
Postcondiciones	-

Flujo normal de eventos

Sección 1.” Crear escena**”**

	Actor	**Sistema**	
1	Selecciona la opción “Archivo”.	2	Muestra un menú.
3	Selecciona la opción “Nueva escena”.	4	Muestra la nueva escena.

Sección 2.” Editar escena”

Sección 2.1 “Ubicar los objetos 3D en el entorno**”**

	Actor	**Sistema**	
1	Selecciona en el menú de la barra de herramientas el objeto que desea ubicar en la escena.		
2	Da clic izquierdo en el lugar de la escena	3	Muestra una ventana para

	que desea ubicar el objeto.		asignarle propiedades al objeto seleccionado antes de ubicarlo en la escena.
4	Asigna las propiedades al objeto y selecciona la opción “Aceptar”.	5	Muestra el objeto pintado en la escena.
Sección 2.” Editar escena” **Sección 2.2 “**Modificar los objetos 3D**”**			
	Actor	**Sistema**	
1	Da clic izquierdo en el objeto ubicado previamente en la escena.	2	Muestra una ventana para asignarle nuevas propiedades al objeto seleccionado.
3	Asigna las nuevas propiedades al objeto y selecciona la opción “Aceptar”.	4	Modifica las propiedades del objeto seleccionado y lo muestra en la escena.
Sección 2.” Editar escena” **Sección 2.3 “**Eliminar los objetos 3D**”**			
	Actor	**Sistema**	
1	Da clic derecho en el objeto ubicado previamente en la escena.	2	Muestra una ventana con la opción de eliminar o no el objeto seleccionado.
3	Selecciona la opción “Aceptar”.	4	Elimina el objeto de la escena.
Flujos alternos			
Sección 2.”Editar escena**”** **Sección 2.1”**Ubicar los objetos 3D en el entorno”			
4. Se dejan campos en blanco o se entra cualquier parámetro que no sea un número			

entero o decimal.			
	Actor		**Sistema**
4.1 4.2	Deja campos en blanco. No entra un número entero o decimal.	5.1	Lanza una excepción. “Datos incorrectos”.
Flujos alternos			
Sección 2.”Editar escena” **Sección 2.2”** Modificar los objetos 3D”			
3. Se dejan campos en blanco o se entra cualquier parámetro que no sea un número entero o decimal.			
	Actor		**Sistema**
3.1 3.2	Deja campos en blanco. No entra un número entero o decimal.	4.1	Lanza la excepción “Datos incorrectos”.
Flujos alternos			
Sección 2.”Editar escena” **Sección 2.3”** Eliminar los objetos 3D”			
1. Se da clic derecho sobre la escena sin haber en ese lugar un objeto.			
	Actor		**Sistema**
1.1	Da clic derecho sobre la escena sin haber en ese lugar un objeto.	2.1	Lanza una excepción.” Seleccione el objeto que desea eliminar”.
CUS-2	Guardar escena en un archivo xml.		
Actor	Profesor		

Resumen	El caso de uso inicia cuando el profesor selecciona la opción “Archivo” y escoge la opción “Guardar” y culmina cuando se ha guardado la escena de manera satisfactoria.
Complejidad	Alta
Prioridad	Crítico
Referencias	RF3
Precondiciones	La escena debe de estar previamente creada.
Postcondiciones	La escena se guardó correctamente en un archivo xml.

Flujo normal de eventos

	Actor		**Sistema**
1	Selecciona la opción “Archivo”.	2	Muestra un menú.
3	Selecciona la opción “Guardar”.	4	Muestra una ventana que permite al profesor escoger donde desea guardar la escena.
5	Selecciona la opción “Aceptar”.	6	Guarda la escena en un archivo xml.

CU-3	Abrir escena desde un archivo xml.
Actor	Profesor
Resumen	El caso de uso inicia cuando el profesor selecciona la opción “Archivo” y escoge la opción “Abrir” y culmina cuando se ha abierto la escena de manera satisfactoria.
Complejidad	Alta
Prioridad	Crítico
Referencias	RF4
Precondiciones	Debe de haber al menos una escena creada, aunque sea en blanco.

Postcondiciones	De existir algún objeto en la escena que se desea abrir, este será dibujado en la pantalla OpenGL.

Flujo normal de eventos			
	Actor	**Sistema**	
1	Selecciona la opción “Archivo”.	2	Muestra un menú.
3	Selecciona la opción “Abrir”.	4	Muestra una ventana que permite al profesor escoger el lugar del cual desea abrir la escena.
5	Selecciona el archivo que desea abrir y selecciona la opción “Aceptar”.	6	Abre la escena desde un archivo xml y la visualiza.

Flujos alternos			
5. No se selecciona un archivo con extensión .xml.			
	Actor		**Sistema**
5.1	Selecciona un archivo con una extensión distinta a .xml.	5.2	Muestra el mensaje “El archivo seleccionado debe ser .xml”.

Tabla # 2 Descripción de los casos de uso del sistema.

2.6 Arquitectura de software

La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución. Una arquitectura bien definida y robusta valida que el software tenga las mejores condiciones ya que es la columna vertebral de la aplicación. [23]

2.6.1 Patrón Modelo Vista Controlador (MVC)

Patrón de diseño de arquitectura de software muy empleado actualmente. Este sugiere la separación del software en tres estratos (Modelo, Vista y Controlador). [24]

Modelo: El modelo en sí son los datos puros que puestos en el contexto del sistema proveen de información al usuario o a la aplicación misma. (Información almacenada en una base de datos o en xml junto con las reglas de negocio que transforman esta información).

Vista: Es la representación del modelo en forma gráfica, disponible para la interacción con el usuario. En el caso de una aplicación WEB, la "Vista" es una página HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones.

Controlador: Es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información y modificando el Modelo en caso de ser necesario. (Código que obtiene datos dinámicamente y genera el contenido HTML).

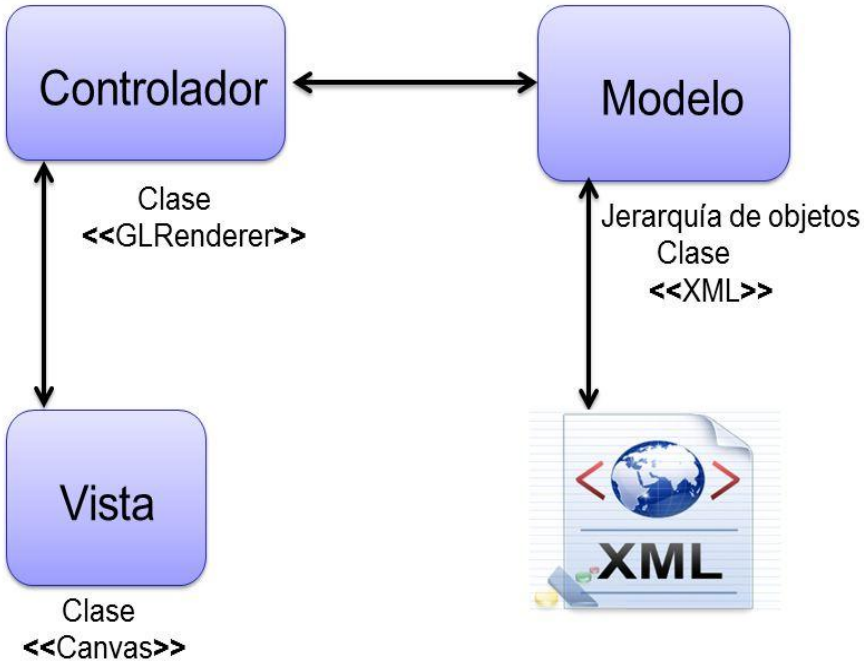


Figura # 2 Patrón de arquitectura Modelo Vista Controlador.

A continuación se mencionan un conjunto de ventajas que el patrón MVC aporta y que se tuvieron en consideración a la hora de hacer su selección:

- La separación del Modelo y la Vista, es decir, separar los datos de su representación visual.
- Facilita agregar nuevos tipos de datos según sea requerido por la aplicación, ya que son independientes del funcionamiento de las otras capas.
- Crea independencia de funcionamiento.

- Facilita el mantenimiento en caso de errores.
- Permite el escalamiento de la aplicación en caso de ser requerido.

2.6.2 Patrones de diseño

Un patrón de diseño describe una estructura recurrente de componentes que se comunican para resolver un problema general de diseño en un contexto particular. Abstrae e identifica los aspectos claves de una estructura de diseño común, lo que los hace útiles para crear un diseño orientado a objetos reutilizable. Identifica las clases e instancias participantes, sus roles y colaboraciones y la distribución de responsabilidades. [25]

Para la realización del diseño de la aplicación se tienen en cuenta los patrones GRASP (Patrones Generales de Software para Asignación de Responsabilidades), los cuales describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. [26]

A continuación se muestra una selección de algunos patrones que serán utilizados durante la realización del diseño de la herramienta propuesta: [26]

Experto: es un patrón que se usa al asignar responsabilidades en diseños orientados a objetos. Expresa que siempre se debe asignar una responsabilidad al experto en información, o sea, a la clase que cuenta con la información necesaria para llevar a cabo la funcionalidad.

Este patrón se puede evidenciar en la clase controladora que funciona como experta en información. Por ejemplo, la clase CC_GLRenderer es la única experta para llevar acabo la funcionalidad de pintar cada uno de los objetos.

Alta Cohesión: este patrón es importante durante todas las decisiones de diseño. Constituye un objetivo subyacente a tener en cuenta continuamente. Es un principio evaluativo que aplica un diseñador mientras evalúa todas las decisiones de diseño. Dentro de los beneficios que este patrón aporta, se pueden mencionar:

- Incrementa la claridad y facilita la comprensión del diseño.
- Simplifica el mantenimiento y las mejoras.
- Soporta a menudo bajo acoplamiento.

Este patrón se puede apreciar en todas las clases que están dentro del paquete Objetos, ya que se ajustan a manejar sólo la responsabilidad para la que fueron creadas (Ej. la clase silla, la única responsabilidad que desempeña es la crear el objeto silla, los mismo pasa con el resto de los objetos).

Bajo Acoplamiento: Este patrón es el encargado de dar soporte a las bajas dependencias y al incremento de la reutilización. Su objetivo principal es diseñar las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

Este patrón se puede apreciar en todas las clases que están dentro del paquete Objetos, ya que estas no dependen de otra clase para cumplir la responsabilidad para la que fueron creadas (Ej. La clase silla, no depende de ninguna otra clase para crear un objeto de tipo silla, lo mismo pasa con el resto de los objetos).

2.7 Diagrama de Clases del Diseño

Un diagrama de clases, es un diagrama estático que describe la estructura de un sistema, mostrando sus clases, atributos y las relaciones que existen entre ellos [27]. Para el diseño de la herramienta de autor que se describe en la presenta investigación se identificaron un conjunto de clases que modelan la solución de las funcionalidades a implementar.

A continuación se muestra el Diagrama de Clases que describe al sistema:

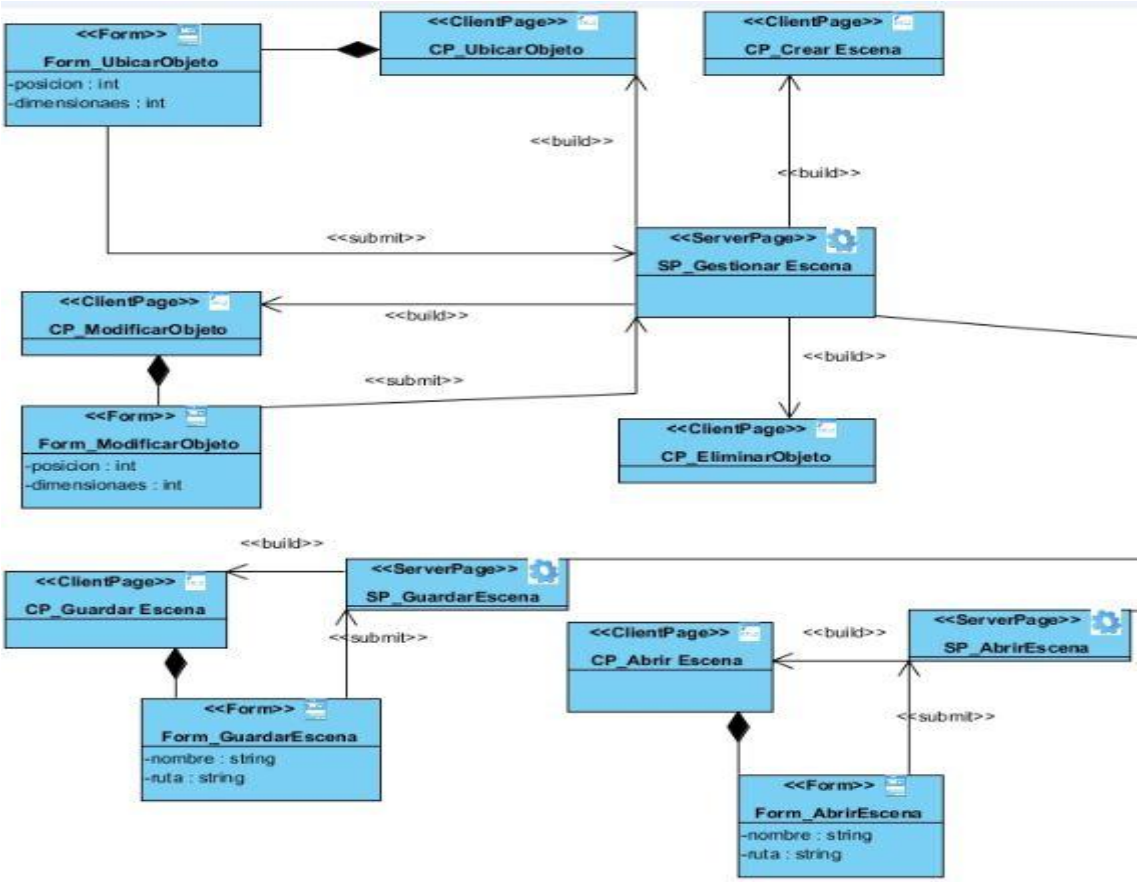


Figura # 3 Vista.

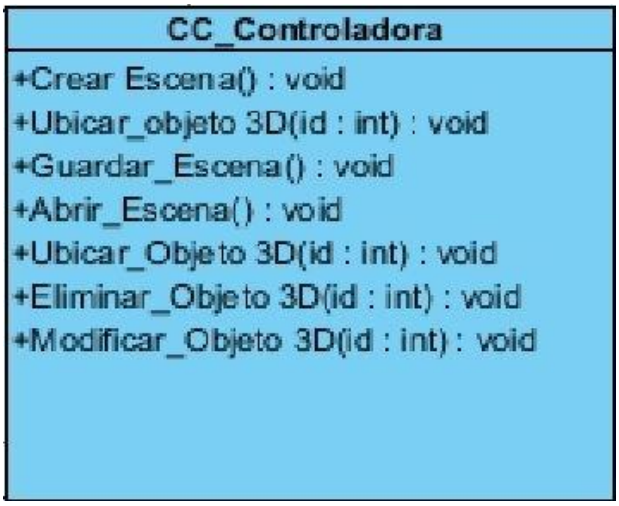


Figura # 4 Controlador.

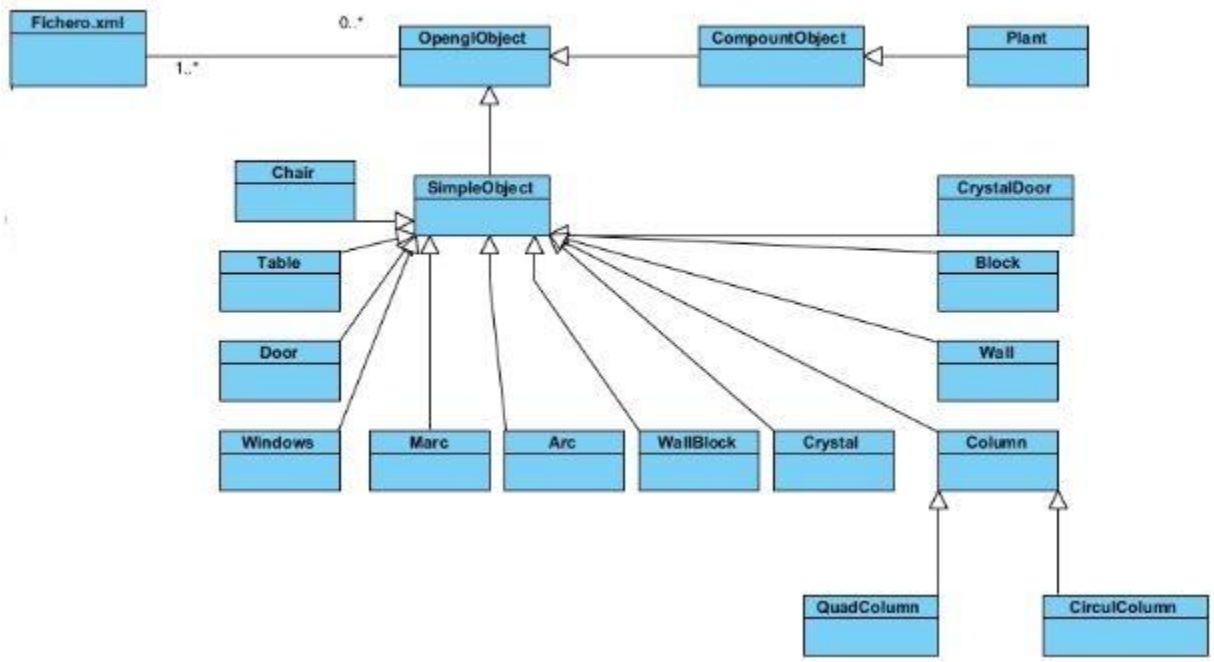


Figura # 5 Modelo.

2.8 Diagramas de Interacción

Un diagrama de colaboración es un diagrama de interacción que resalta la organización estructural de los objetos que envían y reciben los mensajes. [28]

A continuación se muestran algunos Diagramas de Colaboración que describen el flujo de acciones del sistema. Para informarse del resto de los diagramas de colaboración: **Consultar Anexo 1.**

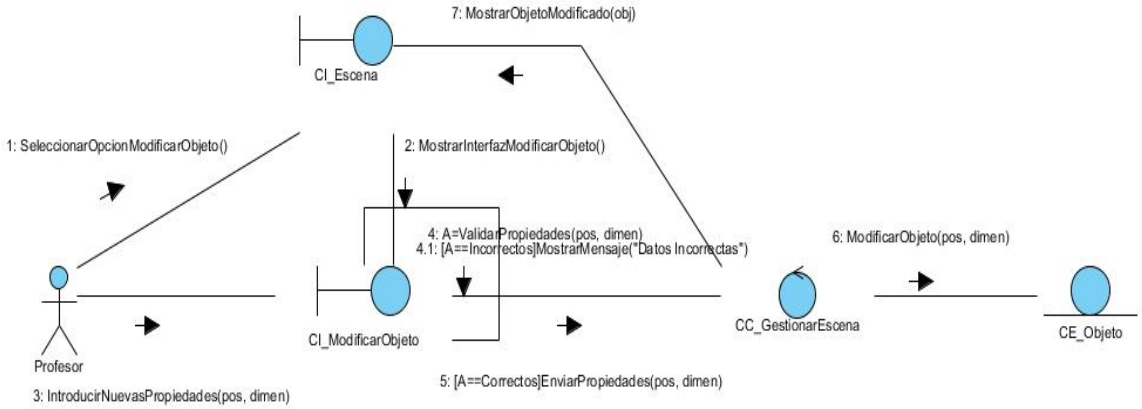


Figura # 6 Diagrama de Colaboración del CU Gestionar Escena (RF2.2 Modificar los objetos 3D).

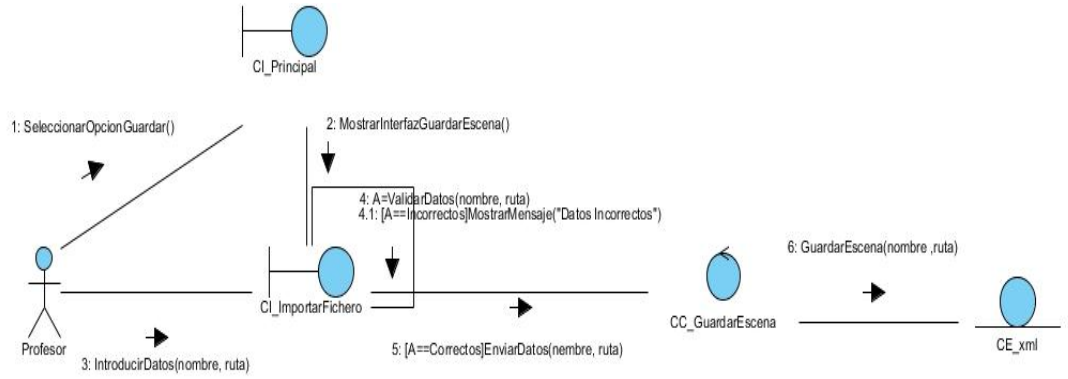
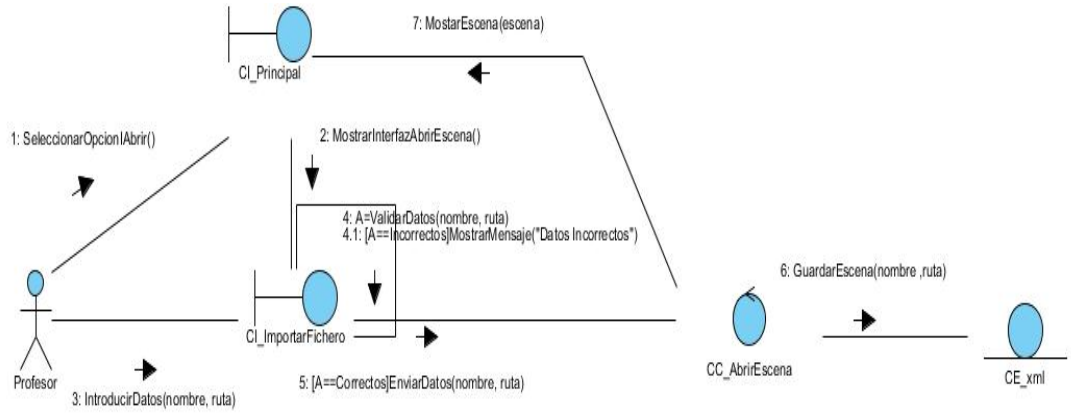


Figura # 7 Diagrama de Colaboración del CU Guardar escena.



Figura# 8 Diagrama de Colaboración del CU Abrir escena.

2.9 Consideraciones finales del Capítulo

Con la realización de este capítulo ha quedado planteada la propuesta de solución de la Herramienta de autor para el diseño de entornos tridimensionales. Los requisitos funcionales y no funcionales serán la base para construir un módulo que cumpla con todas las restricciones y objetivos especificados, haciendo uso de los patrones de diseño y arquitectónicos. Además, con los diagramas de interacción se muestra una panorámica de cómo funcionará la aplicación.

Capítulo 3. Implementación y prueba de la Herramienta de autor para el diseño de entornos tridimensionales.

Después de haber realizado el análisis y diseño del sistema, se ha capturado la mayor parte de la arquitectura del mismo y se ha creado una visión del modelo de implementación. En este capítulo se ilustra la manera en que los elementos de diseño se implementan en términos de componentes. Además se realizan las comprobaciones pertinentes para validar el correcto funcionamiento del software, con el objetivo de encontrar y corregir posibles errores que atenten contra el buen funcionamiento de la aplicación. Los resultados serán registrados, observados y comparados con los objetivos inicialmente trazados.

3.1 Diagrama de Componentes

El Diagrama de Componentes muestra los elementos de diseño de un sistema de software. Además, permite visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. [29]

A continuación se muestra el Diagrama de Componentes perteneciente al sistema Herramienta de autor para el diseño de entornos tridimensionales.

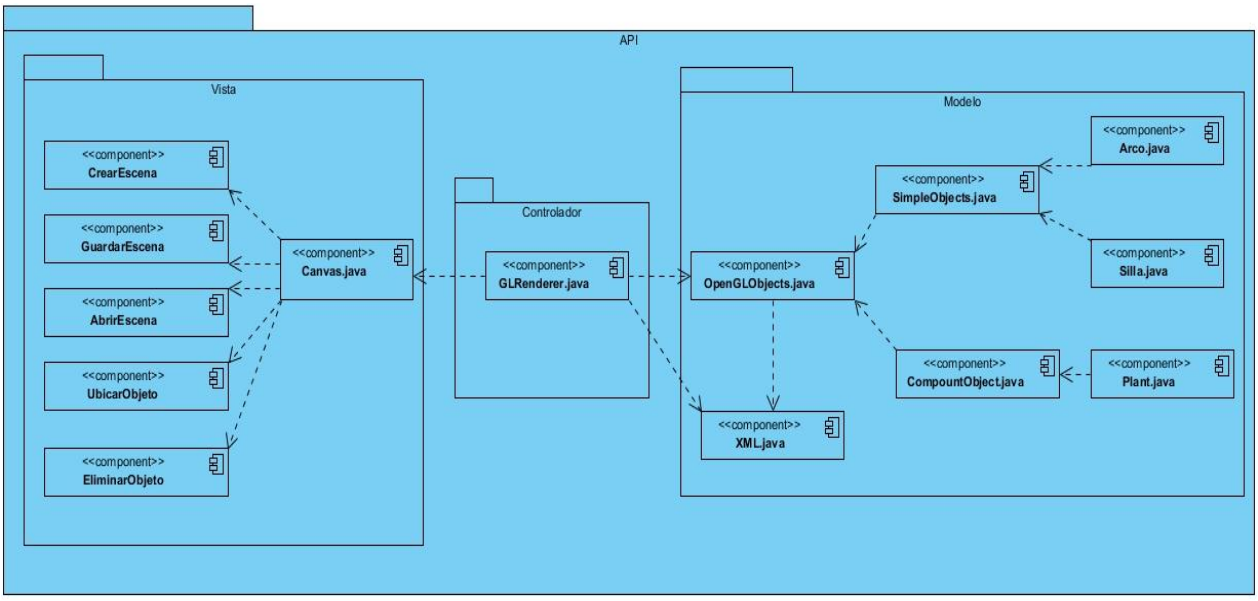


Figura # 9 Diagrama de Componentes.

La vista la proporciona la clase Canvas.java que está dentro del subsistema Vista, esta a su vez establece comunicación con la clase GLRenderer.java situada en el subsistema Controlador, y es la encargada de

dibujar cada uno de los objetos, la misma se relaciona con el subsistema Modelo en el que se encuentran cada uno de los objetos que son dibujados.

3.2 Estándares de codificación

Con el propósito de hacer el código uniforme y mejorar el rendimiento de la aplicación, fue necesario establecer estándares de codificación. Dichos estándares no son más que un conjunto de reglas específicas a una lengua, que reducen apreciablemente el riesgo de que los futuros desarrolladores introduzcan errores en el código existente. Estos estándares permitirán que el sistema sea totalmente independiente del autor o cualquier persona que interactúe con el código, además servirán de referencia a otros programadores para futuras integraciones o mantenimiento.

3.2.1 CamelCase

Es un estándar que se puede aplicar a varios tipos de lenguajes de programación incluyendo Java, .Net, C y C++. La implementación del sistema se desarrollará en el lenguaje de programación Java, usando la librería GLUT del mismo lenguaje, para la representación gráfica de objetos 3D. Existen dos tipos de CamelCase: Lower CamelCase y Upper CamelCase escogiendo el primero para la estandarización del código utilizado durante la implementación de la Herramienta de autor para el diseño de entornos tridimensionales. [30]

A continuación se especificarán algunas reglas a seguir por dicho estándar:

- Para la declaración de los métodos en el código del sistema se realizará con la primera letra en mayúscula.
- Los nombres de las clases deben ser sustantivos o frases de sustantivo que no puedan usar prefijos.
- Las clases y paquetes del sistema se nombran en inglés.
- Las clases quedarán organizadas por los paquetes que representen su funcionalidad en el sistema.
- Para entrada de datos se recomienda no poner prefijos. Por ejemplo en el caso de parámetros para métodos no sería necesario declararlos como p_Nombre.

3.3 Modelo de despliegue

El diagrama de despliegue permite modelar la disposición física o topología de un sistema. Muestra el hardware usado y los dispositivos instalados en él, además de las conexiones físicas entre este y las relaciones entre componentes. Es utilizado para capturar los elementos de configuración del procesamiento y las conexiones entre ellos. También se utiliza para visualizar la distribución de los

componentes de software en los nodos físicos. El mismo está integrado por: nodos, dispositivos y conectores [31]. El diagrama de despliegue propuesto está compuesto por cuatro nodos: PC_Cliente, PC_Servidor de aplicaciones, Servidor directorio activo (UCI) y el Servidor de base de datos. Este será el despliegue de la integración de todos los módulos que conformarán la nueva versión de la herramienta SMProg. La presente investigación se centra solamente en la conexión segura por medio del protocolo <<HTTPS>> del profesor desde una PC_Cliente hasta la PC_Servidor Web, donde será capaz de diseñar entornos 3D.

A continuación se muestra el diagrama de despliegue de la aplicación SMProg:

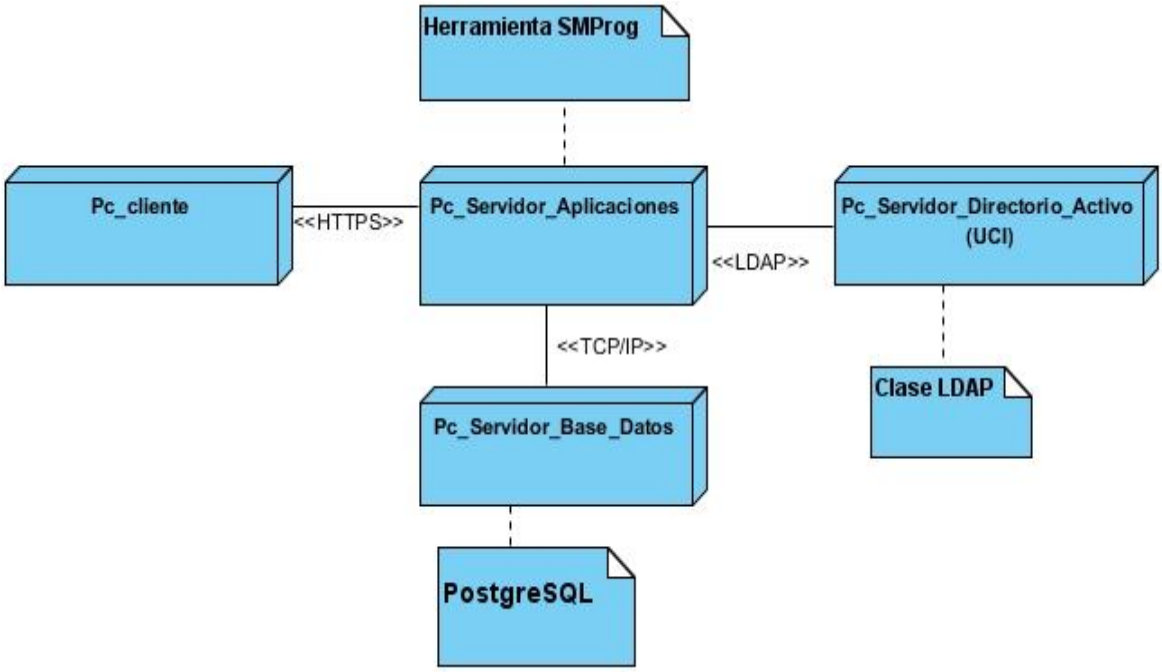


Figura # 10 Diagrama de despliegue.

3.4 Pruebas de Software

Las pruebas son actividades a través de las cuales un sistema o componente es ejecutado bajo ciertas condiciones o requisitos que permiten determinar de forma parcial o completa si el funcionamiento es el esperado o si existen errores que atenten contra el buen desempeño de la aplicación.

3.4.1 Pruebas de Funcionalidades

Este tipo de prueba se realizó por el método de caja negra, el mismo está enfocado a “mirar” en el exterior de lo que se prueba**,** o sea, estas se llevan a cabo sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa [32]. Para la realización de esta prueba se hizo uso

del artefacto Diseño de Casos de Prueba basado en casos de uso definido por la metodología RUP. Consultar **Anexo 2**.

3.4.2 Resultados de las pruebas de funcionalidades

A partir del diseño y ejecución de los casos de prueba se detectaron 15 no conformidades tales como: existencia de objetos que no eran dibujados en la posición especificada, objetos que eran guardados de un tamaño y al ser cargados en la escena tomaban otra dimensión, redimensión de la pantalla una vez que se pintaban los objetos y errores ortográficos, entre otros.

De estas no conformidades fueron corregidas 10 en la primera iteración, dándole paso a una segunda iteración de pruebas, en la cual se evaluaron nuevamente las funcionalidades y se detectaron 5, las cuales fueron corregidas. En la tercera iteración no se encontró ninguna no conformidad obteniéndose resultados satisfactorios para esta etapa del desarrollo del proyecto.

En la figura #11 se muestran las no conformidades detectadas durante el proceso de pruebas y las iteraciones que se realizaron para solucionarlas.

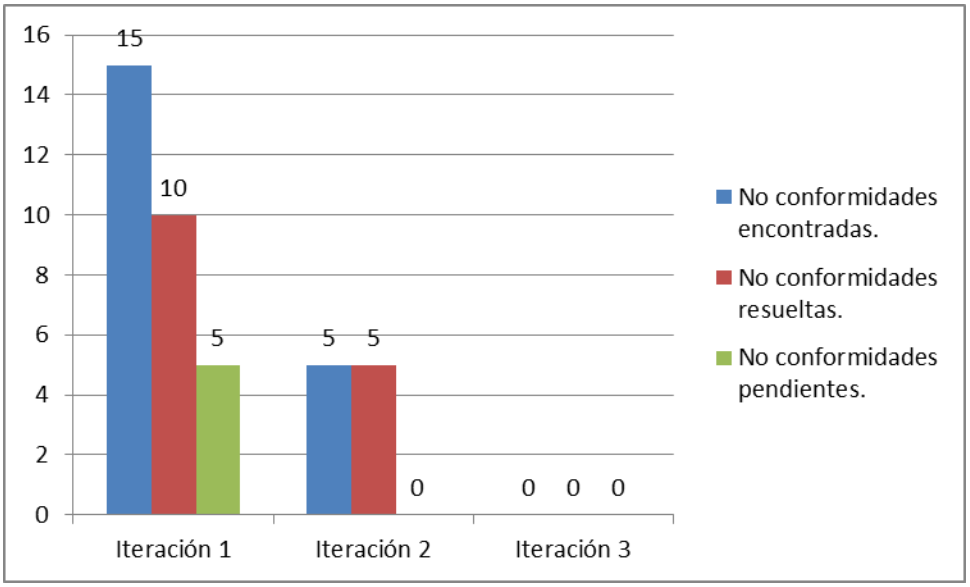


Figura # 11 Iteraciones realizadas durante el proceso de pruebas

3.5 Interfaces principales del sistema

Una vez realizada la aplicación se obtuvieron las siguientes vistas, las cuales permiten visualizar algunas de las funcionalidades del producto.

La siguiente imagen muestra la interfaz principal de la aplicación, desde donde el usuario ejecutará todas las acciones que desee.

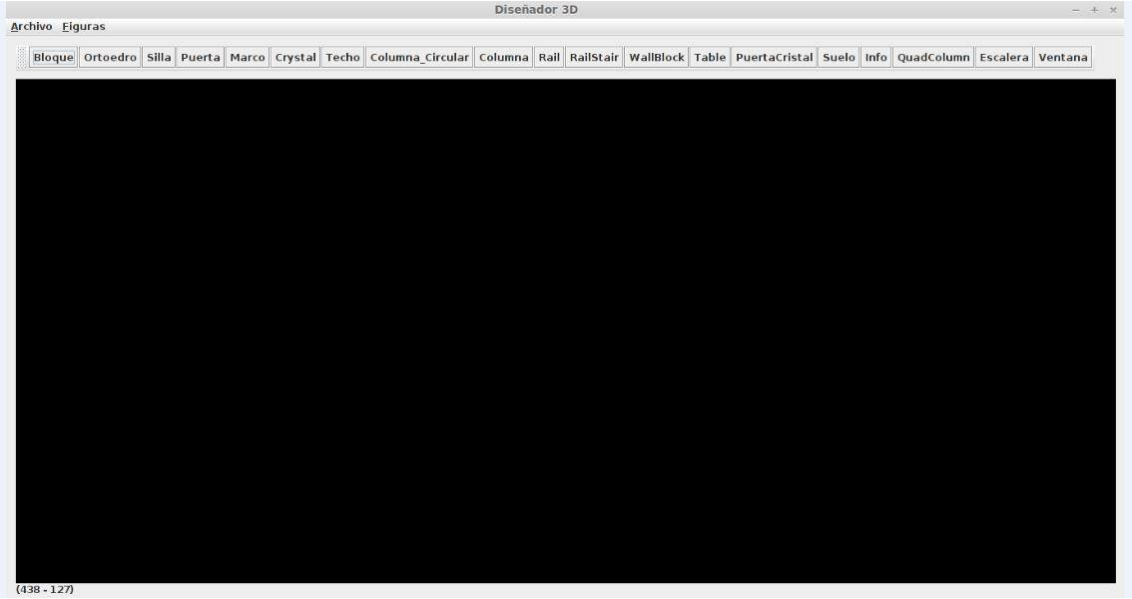


Figura # 12 Interfaz Principal.

La siguiente interfaz pertenece a la funcionalidad Ubicar objeto en el entorno 3D, en la cual se introducen los datos correspondientes a las dimensiones y el posicionamiento que tendrá el objeto en la escena.

Figura # 13 Funcionalidad Ubicar los objetos 3D en el entorno.

La interfaz que a continuación se muestra pertenece a la funcionalidad Abrir escena desde un archivo xml, la cual permite cargar una escena previamente creada y mostrarla.

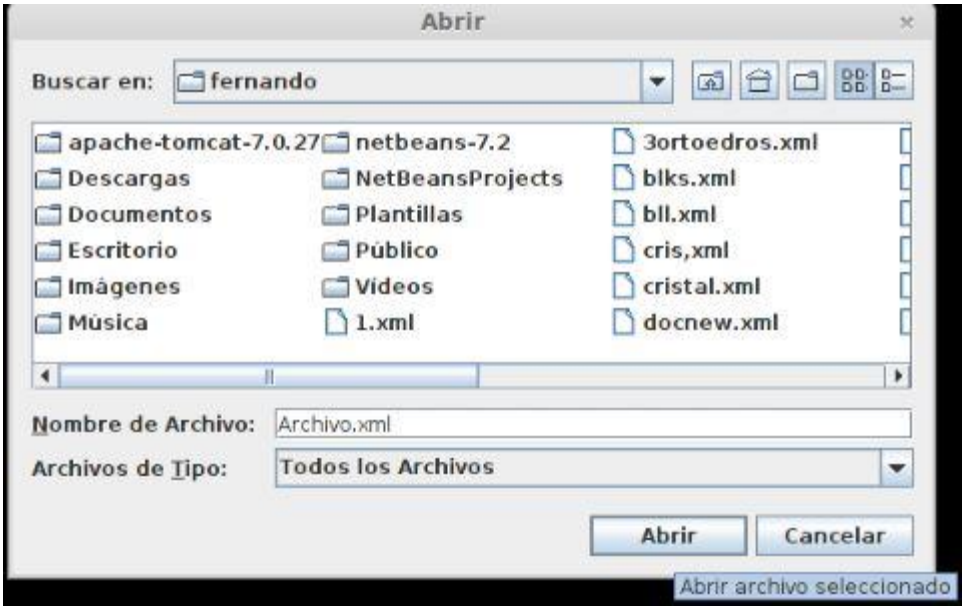


Figura # 14 Interfaz Abrir escena.

Por último se muestra la interfaz de la funcionalidad Guardar escena en un archivo xml, donde se selecciona el lugar donde se desea guardar la escena creada.

Figura # 15 Funcionalidad Guardar escena.

3.6 Consideraciones finales del capítulo

En este capítulo se logró una presentación del sistema a través de los tipos de contenidos que va a contener el mismo. Con las pruebas realizadas se comprobó el correcto funcionamiento de la herramienta de autor para entornos tridimensionales y se garantiza su integración con el sistema SMProg 2.0.

Conclusiones

Una vez desarrollada la Herramienta de autor para el diseño de entornos tridimensionales se pueden plantear las siguientes conclusiones:

- A partir del estudio realizado sobre algunas herramientas de autor para el diseño de entornos tridimensionales, se evidenció que las mismas no eran factibles para su utilización en la UCI y la necesidad del desarrollo de una herramienta de autor que se integre al sistema SMProg 2.0.
- Con el correcto desarrollo de la arquitectura del sistema se facilita el diseño de entornos tridimensionales por parte de los profesores.
- Se verificó el correcto funcionamiento de la aplicación a través de las pruebas realizadas al software, las mismas arrojaron un resultado positivo en cuanto a la calidad de la herramienta de autor y al cumplimiento de las expectativas trazadas inicialmente.

Recomendaciones

Se recomienda:

- Asociar texturas a los objetos 3D.
- Incluir la ventana OpenGL en una aplicación web.
- Permitir el desplazamiento a través de la escena en tiempo de diseño.

Referencia Bibliográfica

[1] **R. Talancón Díaz y L. R. Suárez Batista.** «Sistema para la visualización de imágenes y modelos 3D médicos en la WEB con fines educativos». Universidad de las Ciencias Informáticas. La Habana, Cuba, 2010.138.

[2] **D. Suárez Ferreiro.** «Herramienta para la creación y uso de juegos didácticos en la enseñanza de la Programación Trabajo». Universidad de las Ciencias Informáticas. La Habana, Cuba, 2011.55.

[3] Definición, características y tipos. [En línea]. [Citado el 29 diciembre 2011]. Disponible en Web: http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/3D/Realidad%20Virtual/web/definicion.html.

[4] **D. G. Níkleva.** «LAS HERRAMIENTAS DE AUTOR COMO HERRAMIENTA DOCENTE». [En línea]. [Citado el 18 diciembre 2012]. Disponible en Web: http://congresos.um.es/ticbiomed/index/search/authors/view?firstName=DIMITRINKA&middleName=G.&lastName=N%C3%8DKLEVA&affiliation=UNIVERSIDAD%20DE%20GRANADA&country=ES.

[5] Programación de aplicaciones web de **Sergio Luján Mora** - Editorial Club Universitario. [En línea]. [Citado el 20 diciembre 2012]. Disponible en Web: http://www.editorial-club-universitario.es/libro.asp?ref=367.

[6] **EAVES, J. y GODFREY, R. J. W.** *Apache Tomcat Bible.* Wiley India Pvt. Limited, 2003. 817 p. ISBN 9788126504305.

[7] **M. Jiménez.** « Escenarios virtuales WEB3D: Simulación con VRML». JAVA3D y X3D. 2004.

[8] Khronos to Create New Camera Control Open Standard API - Khronos Group Press Release. [En línea]. [Citado el 12 enero 2013]. Disponible en Web: http://www.khronos.org/news/press/releases/khronos-webgl-initiative-hardware-accelerated-3d-graphics-internet.

[9] Sandy is a Flash 3D engine, available in 3 versions: AS2, AS3 and haXe. [En línea]. [Citado el 8 noviembre 2012]. Disponible en Web: http://www.flashsandy.org/.

[10] Google joins effort for 3D Web standard with new plugin, API | Ars Technica. [En línea]. [Citado el 2 febrero 2013]. Disponible en Web: http://arstechnica.com/information-technology/2009/04/google-releases-3d-graphics-plugin-for-browsers/.

[11] Blender.org - Home of the Blender project - Free and Open 3D creation software. [En línea]. [Citado el 12 febrero 2013]. Disponible en Web: http://www.blender.org/.

[12] **Edwards, Lin.** Superior 3D Graphics for the WEB a Step Closer. septiembre 22, 2009.

[13] zona Clic - JClic. [En línea] [Citado el: 23 febrero 2013.] Disponible en Web: http://clic.xtec.cat/es/jclic/.

[14] Squeak, un entorno de aprendizaje diferente. [En línea] [Citado el: 23 febrero 2013.] Disponible en Web: http://squeak.educarex.es/Squeakpolis/.

[15] **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso Unificado de Desarrollo de Software.* La Habana: Féliz Varela, 2004.

[16] Object Management Group - UML. [En línea]. [Citado el: 22 febrero 2013]. Disponible en Web: http://www.uml.org/.

[17] **Enrique Hernández Orallo**. Disca. *Disca.* [En línea] [Citado el: 23 febrero 2013.]. Disponible en Web: http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

[18] Increase productivity and enhance communication and collaboration efficiency by using UML. [En línea] [Citado el: 12 diciembre 2012.]. Disponible en Web: http://www.visual-paradigm.com.

[19] Java Help Center. [En línea] [Citado el: 24 diciembre 2012.]. Disponible en Web: http://www.java.com/en/download/help/index.xml.

[20] NetBeans IDE - Features. [En línea] [Citado el: 24 diciembre 2012.]. Disponible en Web: http://netbeans.org/features/index.html.

[21] Tutoriales Programación: OpenGL - ABCdatos. [En línea]. [Citado el: 3 abril 2013]. Disponible en Web: http://www.abcdatos.com/tutoriales/programacion/opengl.html.

[22] **DOUG TIDWELL**. Introduccion a XML. [En línea]. [Citado el: 2 marzo 2013]. Disponible en Web: http://es.scribd.com/doc/17800576/Introduccion-a-XML.

[23] **Pressman, R. S.**Software Engineering. A practitioner´s approach. NY, Mc Graw Hill, 2010, 238-239 p.

[24] **Pavón, Juan.** *Estructura de las Aplicaciones Orientadas a Objetos El patrón Modelo-Vista-Controlador (MVC).* 2009. p. 12.

[25] Patrones y Antipatrones: una Introducción - Parte I. [En línea]. [Citado el: 2 abril 2013]. Disponible en Web: http://msdn.microsoft.com/es-es/library/bb972242.aspx.

[26] **Andrés Grosso**. Patrones GRASP. [En línea]. [Citado el: 3 abril 2013]. Disponible en Web: http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

[27] **Pressman, R. S.** Software Engineering. A practitioner´s approach. NY, Mc Graw Hill, 2010, 852.

[28] Diagrama de colaboración. [En línea]. [Citado el: 2 abril 2013]. Disponible en Web: http://www.arghys.com/general/diagrama-de-colaboracion.html.

[29] **Schmuller, Joseph** UML_en_24_hors. 2003. pag 168-173.

[30] **PEDRO R. MARCOS**. Escritura estilo CamelCase. [En línea]. 29 enero 2013. [Citado el: 2 mayo 2013]. Disponible en Web: http://www.pedrormarcos.com/escritura-estilo-camelcase/.

[31] **Pressman, R. S**.Software Engineering. A practitioner´s approach. NY, Mc Graw Hill, 2010, 852 p.

[32] **PRESSMAN, R. S. A. y MURRIETA, J. E. M**. *Ingeniería del software: un enfoque práctico.* 6 ed. Mcgraw Hill/Interamericana Editores, 2006. 958 p. ISBN 9789701054734.

Bibliografía

1. **R. Talancón Díaz y L. R. Suárez Batista.** «Sistema para la visualización de imágenes y modelos 3D médicos en la WEB con fines educativos». Universidad de las Ciencias Informáticas. La Habana, Cuba, 2010.138.
2. **D. Suárez Ferreiro.** «Herramienta para la creación y uso de juegos didácticos en la enseñanza de la Programación Trabajo». Universidad de las Ciencias Informáticas. La Habana, Cuba, 2011.55.
3. Definición, características y tipos. [En línea]. [Citado el 29 diciembre 2011]. Disponible en Web: http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/3D/Realidad%20Virtual/web/definicion.html.

4. **D. G. Níkleva.** «LAS HERRAMIENTAS DE AUTOR COMO HERRAMIENTA DOCENTE». [En línea]. [Citado el 18 diciembre 2012]. Disponible en Web: http://congresos.um.es/ticbiomed/index/search/authors/view?firstName=DIMITRINKA&middleName=G.&lastName=N%C3%8DKLEVA&affiliation=UNIVERSIDAD%20DE%20GRANADA&country=ES.

5. Programación de aplicaciones web de **Sergio Luján Mora** - Editorial Club Universitario. [En línea]. [Citado el 20 diciembre 2012]. Disponible en Web: http://www.editorial-club-universitario.es/libro.asp?ref=367.

6. **EAVES, J. y GODFREY, R. J. W.** *Apache Tomcat Bible.* Wiley India Pvt. Limited, 2003. 817 p. ISBN 9788126504305.

7. **M. Jiménez.** « Escenarios virtuales WEB3D: Simulación con VRML». JAVA3D y X3D. 2004.
8. Khronos to Create New Camera Control Open Standard API - Khronos Group Press Release. [En línea]. [Citado el 12 enero 2013]. Disponible en Web: http://www.khronos.org/news/press/releases/khronos-webgl-initiative-hardware-accelerated-3d-graphics-internet.
9. Sandy is a Flash 3D engine, available in 3 versions: AS2, AS3 and haXe. [En línea]. [Citado el 8 noviembre 2012]. Disponible en Web: http://www.flashsandy.org/.
10. Google joins effort for 3D Web standard with new plugin, API | Ars Technica. [En línea]. [Citado el 2 febrero 2013]. Disponible en Web: http://arstechnica.com/information-technology/2009/04/google-releases-3d-graphics-plugin-for-browsers/.

11. blender.org - Home of the Blender project - Free and Open 3D creation software. [En línea]. [Citado el 12 febrero 2013]. Disponible en Web: http://www.blender.org/.

12. **Edwards, Lin.** Superior 3D Graphics for the WEB a Step Closer. septiembre 22, 2009.
13. zonaClic - JClic. [En línea] [Citado el: 23 febrero 2013.] Disponible en Web: http://clic.xtec.cat/es/jclic/.

14. Squeak, un entorno de aprendizaje diferente. [En línea] [Citado el: 23 febrero 2013.] Disponible en Web: http://squeak.educarex.es/Squeakpolis/.

15. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso Unificado de Desarrollo de Software.* La Habana : Féliz Varela, 2004.

16. Object Management Group - UML. [En línea]. [Citado el: 22 febrero 2013]. Disponible en Web: http://www.uml.org/.

17. **Enrique Hernández Orallo**. Disca. *Disca.* [En línea] [Citado el: 23 febrero 2013.]. Disponible en Web: http://www.disca.upv.es/enheror/pdf/ActaUML.PDF.

18. Increase productivity and enhance communication and collaboration efficiency by using UML. [En línea] [Citado el: 12 diciembre 2012.]. Disponible en Web: http://www.visual-paradigm.com.

19. Java Help Center. [En línea] [Citado el: 24 diciembre 2012.]. Disponible en Web: http://www.java.com/en/download/help/index.xml.

20. NetBeans IDE - Features. [En línea] [Citado el: 24 diciembre 2012.]. Disponible en Web: http://netbeans.org/features/index.html.

21. Tutoriales Programación: OpenGL - ABCdatos. [En línea]. [Citado el: 3 abril 2013]. Disponible en Web: http://www.abcdatos.com/tutoriales/programacion/opengl.html.

22. **DOUG TIDWELL**. Introduccion a XML. [En línea]. [Citado el: 2 marzo 2013]. Disponible en Web: http://es.scribd.com/doc/17800576/Introduccion-a-XML.

23. **Pressman, R. S.**Software Engineering. A practitioner´s approach. NY, Mc Graw Hill, 2010, 238-239 p.

24. **Pavón, Juan.** *Estructura de las Aplicaciones Orientadas a Objetos El patrón Modelo-Vista-Controlador (MVC).* 2009. p. 12.

25. Patrones y Antipatrones: una Introducción - Parte I. [En línea]. [Citado el: 2 abril 2013]. Disponible en Web: http://msdn.microsoft.com/es-es/library/bb972242.aspx.

26. **Andrés Grosso**. Patrones GRASP. [En línea]. [Citado el: 3 abril 2013]. Disponible en Web: http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

27. **Pressman, R. S.** Software Engineering. A practitioner´s approach. NY, Mc Graw Hill, 2010, 852.

28. Diagrama de colaboración. [En línea]. [Citado el: 2 abril 2013]. Disponible en Web: http://www.arghys.com/general/diagrama-de-colaboracion.html.

29. **Schmuller, Joseph** UML_en_24_hors. 2003. pag 168-173.

30. **PEDRO R. MARCOS**. Escritura estilo CamelCase. [En línea]. 29 enero 2013. [Citado el: 2 mayo 2013]. Disponible en Web: http://www.pedrormarcos.com/escritura-estilo-camelcase/.

31. **Pressman, R. S**.Software Engineering. A practitioner´s approach. NY, Mc Graw Hill, 2010, 852 p.

32. **PRESSMAN, R. S. A. y MURRIETA, J. E. M**. *Ingeniería del software: un enfoque práctico.* 6 ed. Mcgraw Hill/Interamericana Editores, 2006. 958 p. ISBN 9789701054734.

33. **Muneta, Martínez, et al.** Últimas Tendencias en Gráficos WEB3D para Internet. 2002.

34. **Macías, Jiménez.** Escenarios virtuales WEB3D: Simulación con VRML, JAVA3D y X3D. 2004.

35. **Hilera, José R., Otón, Salvador and Martínez, Javier.** Aplicación de la realidad virtual en la enseñanza a través de internet. 2000.

36. **Brutzman, Don and Daly, Leonard.** *X3D: Extensible 3D Graphics for WEB authors.* San Francisco: Elsevier Inc, 2007.

37. **Cardona, Jordi R.** X3D-Edit Free Software For X3D Code Edition And Worldbuilding. mayo 13, 2008.

38. **Torossi, Gustavo.** El Proceso Unificado de Desarrollo de Software. 2005.

Anexos

Anexo1: Diagramas de Colaboración

A continuación se muestran los restantes Diagramas de Colaboración que describen el flujo de acciones del sistema:

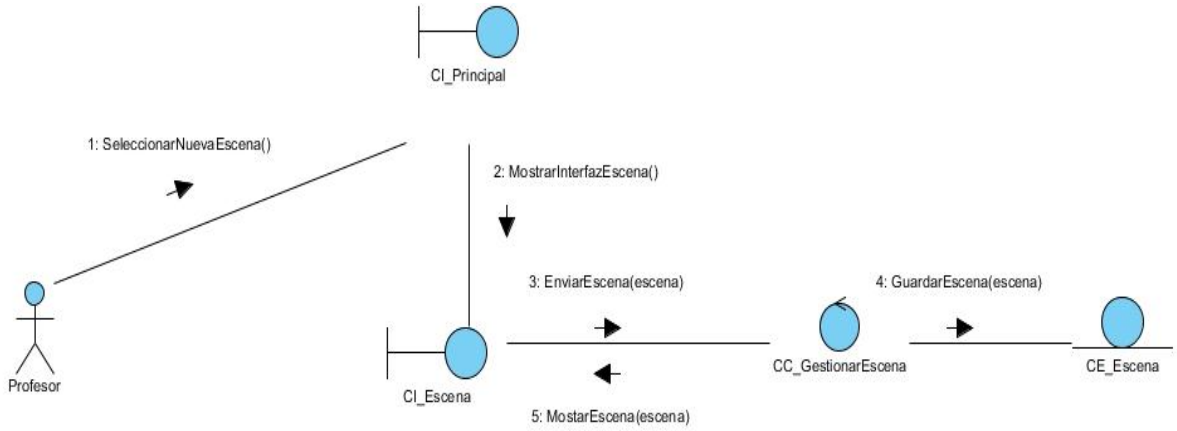


Figura # 16 Diagrama de Colaboración del CU Gestionar Escena (RF1 Crear escena).

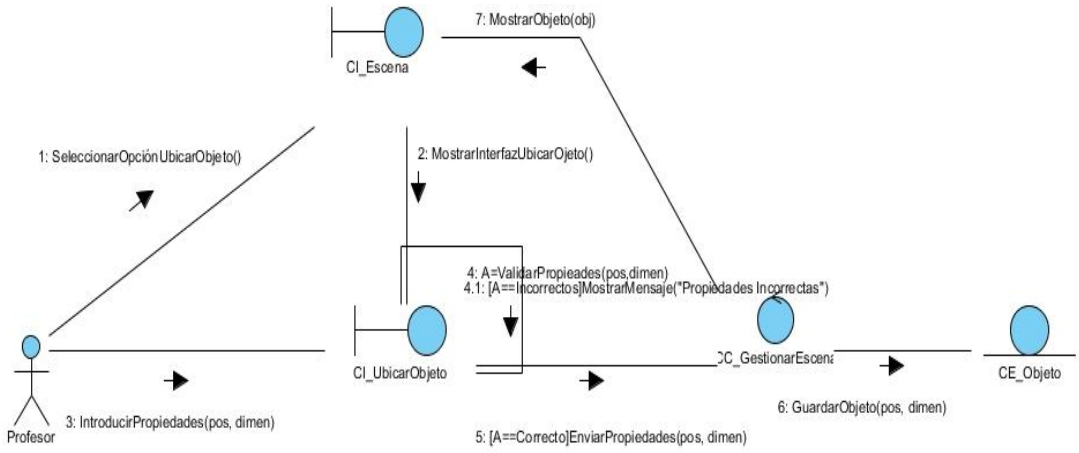


Figura # 17 Diagrama de Colaboración del CU Gestionar Escena (RF2.1 Ubicar los objetos 3D en el entorno.

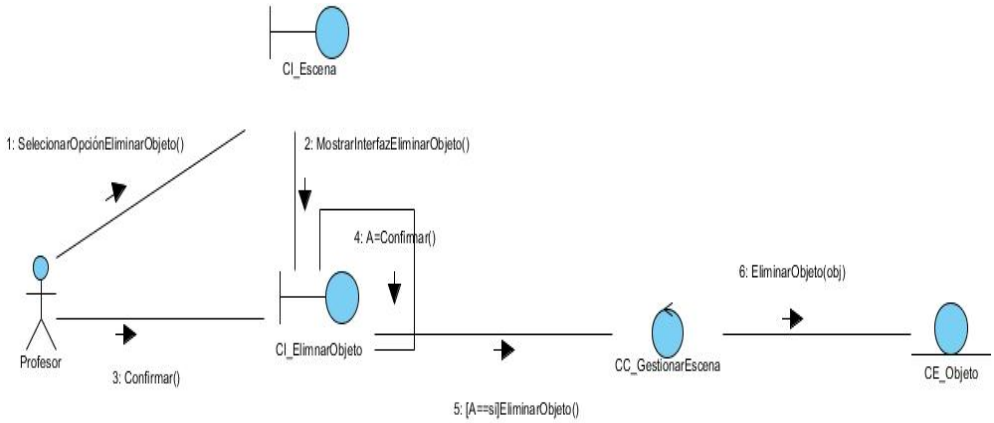


Figura # 18 Diagrama de Colaboración del CU Gestionar Escena (RF2.3 Eliminar los objetos 3D).

Anexo2: Diseño de Casos de Prueba basado en Casos de Uso

A continuación se muestran los Diseños de Casos de Prueba basado en Casos de Uso que describen el flujo de acciones del sistema:

Escenario	Descripción	Variable 1 Alto	Variable 2 Ancho	Variable 3 Largo	Variable 4 Radio	Variable 5 Ángulo	Variable 6 Espesor	Variable 7 Base	Variable 8 Marco	Variable 9 Superficie	Variable 10 X	Variable 11 Y	Respuesta del sistema	Flujo central
EC 1.1 Modificar objeto 3D con algún campo incorrecto.	*El usuario introduce un dato incorrecto.*	*I* *null o distinto de número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*El sistema muestra el mensaje de error:* *Datos incorrectos.*	*El usuario debe dar clic izquierdo sobre el objeto que desea modificar y el sistema levantará la interfaz para modificar los datos que desee. Luego el usuario llenará los campos y el sistema muestra un mensaje en dependencia de los datos que sean*
		V *número*	*I* *null o distinto de número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*		
		V *número*	*V* *número*	*I* *null o distinto de número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*		
		V *número*	*V* *número*	*V* *número*	*I* *null o distinto de número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*		
		V *número*	*V* *número*	*V* *número*	*V* *número*	*I* *null o distinto de número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*		
		V *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*I* *null o distinto de número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*		
		V *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*I* *null o distinto de número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*		

		V número	V número	V número	V número	V número	V número	V número	I null o distinto de número	V número	V número	V número			introducidos.
		V número	V número	V número	V número	V número	V número	V número	V número	I null o distinto de número	V número	V número			
		V número	V número	V número	V número	V número	V número	V número	V número	V número	I null o distinto de un número entero.	V número			
		V número	V número	V número	V número	V número	V número	V número	V número	V número	V número	I null o distinto de un número entero.			
EC 1.2 Modificar objeto 3D con todos los datos correctamente.	El usuario introduce todos los datos correctamente.	V	V	V	V	V	V	V	V	V	V	V	El sistema modifica el objeto en el entorno.	El usuario debe dar clic izquierdo sobre el objeto que desea modificar y el sistema levantará la interfaz para modificar los datos	
		número	número	número	número	número	número	número	número	número	número	número			

														que desee. Luego el usuario llenará los campos y el sistema realiza los cambios.

Tabla # 3 Diseño de Casos de Prueba basado en Casos de Uso (RF2.2 Modificar Objeto 3d).

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	*Alto*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
2	*Ancho*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
3	*Largo*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
4	*Radio*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
5	*Ángulo*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
6	*Espesor*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
7	*Base*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
8	*Marco*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
9	*Superficie*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
10	*X*	*campo de texto*	*No.*	*Los datos introducidos deben ser números enteros.*
11	*Y*	*campo de texto*	*No.*	*Los datos introducidos deben ser números enteros.*

Tabla # 4 Descripción de las variables del Diseño de Casos de Prueba basado en Casos de Uso (RF2.2 Modificar Objeto 3d).

Escenario	Descripción	Variable 1 Alto	Variable 2 Ancho	Variable 3 Largo	Variable 4 Radio	Variable 5 Ángulo	Variable 6 Espesor	Variable 7 Base	Variable 8 Marco	Variable 9 Superficie	Variable 10 X	Variable 11 Y	Respuesta del sistema	Flujo central
EC 1.1 Ubicar objeto 3D en el entorno con datos incorrectos.	*El usuario introduce un dato incorrecto.*	*I* *null o distinto de número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*El sistema muestra el mensaje de error:* *Datos incorrectos.*	*El usuario debe seleccionar el objeto que desea pintar y dar clic en el lugar donde desea ubicarlo. El sistema levantará la interfaz para introducir los datos del objeto. Luego el usuario llenará los campos y el sistema muestra un mensaje en dependencia de los datos que sean introducidos.*
		V *número*	*I* *null o distinto de número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*		
		V *número*	*V* *número*	*I* *null o distinto de número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*		
		V *número*	*V* *número*	*V* *número*	*I* *null o distinto de número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*		
		V *número*	*V* *número*	*V* *número*	*V* *número*	*I* *null o distinto de número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*		
		V *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*I* *null o distinto de número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*	*V* *número*		
		V	*V*	*V*	*V*	*V*	*V*	*I*	*V*	*V*	*V*	*V*		

		número	*número*	*número*	*número*	*número*	*número*	*null o distinto de número*	*número*	*número*	*número*	*número*		
		V número	*V número*	*V número*	*V número*	*V número*	*V número*	*V número*	*I null o distinto de número*	*V número*	*V número*	*V número*		
		V número	*V número*	*V número*	*V número*	*V número*	*V número*	*V número*	*V número*	*I null o distinto de número*	*V número*	*V número*		
		V número	*V número*	*V número*	*V número*	*V número*	*V número*	*V número*	*V número*	*V número*	*I null o distinto de número*	*V número*		
		V número	*V número*	*V número*	*V número*	*V número*	*V número*	*V número*	*V número*	*V número*	*V número*	*I null o distinto de número*		
EC 1.2 Ubicar objeto 3D en el entorno con los datos correctos.	*El usuario introduce todos los datos correctamente.*	*V número*	*V número*	*V número*	*V número*	*V número*	*V número*	*V número*	*V número*	*V número*	*V número*	*V número*	*El sistema dibuja el objeto en el entorno.*	*El usuario debe seleccionar el objeto que desea pintar y dar clic en el lugar donde desea ubicarlo. El sistema levantará la interfaz para introducir los datos del objeto. Luego el*

															usuario llenará los campos y el sistema dibuja el objeto en el entorno.

Tabla # 5 Diseño de Casos de Prueba basado en Casos de Uso (RF2.1 Ubicar Objeto 3d en el entorno).

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	*Alto*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
2	*Ancho*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
3	*Largo*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
4	*Radio*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
5	*Ángulo*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
6	*Espesor*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
7	*Base*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
8	*Marco*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
9	*Superficie*	*campo de texto*	*No.*	*Los datos introducidos deben ser números.*
10	*X*	*campo de texto*	*No.*	*Los datos introducidos deben ser números enteros.*
11	*Y*	*campo de texto*	*No.*	*Los datos introducidos deben ser números enteros.*

Tabla # 6 Descripción de las variables del Diseño de Casos de Prueba basado en Casos de Uso (RF2.1 Ubicar Objeto 3D en el entorno).

Glosario de Términos

API – *Application Programming Interface* (Interfaz de Programación de Aplicaciones).

CASE - *ComputerAided Software Engineering* (Ingeniería de Software Asistida por Computación): Son aplicaciones informáticas para aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

HTML - *Hypertext Markup Language* (Lenguaje de Marcado Hipertextual): lenguaje de marcado para hipertextos, utilizado para la confección de páginas Web.

IDE - *Integrated Development Environment* (Entorno Integrado de Desarrollo): es un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

OMG - Object Management Group (Grupo de Gestión de Objeto): es una asociación sin fines de lucro formada por grandes corporaciones, muchas de ellas de la industria del software, como IBM, Apple, Sun Microsystems y HP. Es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, tales como UML, XMI, CORBA. Es una organización que promueve el uso de tecnología orientada a objetos mediante guías y especificaciones para las mismas.

PEA - Proceso de Enseñanza-Aprendizaje.

RUP - *Rational Unified Process* (Proceso Unificado de desarrollo): Metodología para el desarrollo de *Software*.

TPC - Técnicas de Programación de Computadoras.

TIC - Tecnologías de la Información y las Comunicaciones.

UML - *Unified Modeling Language* (Lenguaje Unificado de Modelado): Prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación.

Web: Sistema para presentar información en internet basado en hipertexto.

XML - *Extensible Markup Language* (Lenguaje de Marca Extensible): es un metalenguaje que nos permite definir lenguajes de marcado adecuados a usos determinados. Es un estándar internacionalmente reconocido y no pertenece a ninguna compañía por lo que su utilización es libre.