UNIVERSIDAD DE LAS CIENCIAS INFÓRMATICAS

FACULTAD 5

Título: Implementación con cálculo de distancias de Escenario Virtual Tridimensional Interactivo Web.

Tesis en opción al título académico de:
Ingeniero en Ciencias Informáticas.

Autor: Carlos Vallejo Vigoa

Tutor: Gilberto Cao Tarrero

Ciudad de la Habana

Junio de 2011

Curso 2010-2011

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores del trabajo titulado:

Implementación con cálculo de distancias de Escenario Virtual Tridimensional Interactivo Web y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____	_____
Autor: Carlos Vallejo Vigoa	Tutor: Ing. Gilberto Cao Tarrero
Firma	Firma

DATOS DE CONTACTO

Tutor: Ing. Gilberto Cao Tarrero.

Edad: 25

Ciudadanía: cubano

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Ciencias Informáticas.

Categoría Docente:

E-mail: gcao@uci.cu

Graduado en la Universidad de las Ciencias Informáticas en el año 2009. Se desempeñó en el período 2009-2010 como líder de la línea de productos Visualización Web del Centro de Informática Industrial (CEDIN). Desde el 2010 dirige el desarrollo del módulo Laboratorio Virtual del software educativo “La naturaleza y el hombre” en el Centro de Tecnologías para la Formación (FORTES).

DEDICATORIA

A mis padres, hermana, abuelos, familia y amigos.

AGRADECIMIENTOS

A mi mamá: Mami a ti, por desde pequeño mostrarme el camino a seguir y dejarme andarlo solo, por estar siempre a mi lado, cuando te he necesitado y por muchas cosas más que no cabrían en esta página pero que estoy seguro tú sabes cuales son, mi compromiso en esta vida es contigo, orgulloso de ser tu hijo.

A mi papá: A ti papá por ser ejemplo cada día, de superación y sacrificio, yo sé que no he seguido tu ejemplo al pie de la letra, pero estoy haciendo el intento, orgulloso de ti.

A mi hermana: Enana a ti también te agradezco, por estar siempre fajada conmigo, por ser mí hermana menor, te quiero mucho.

Abuela Aida: A ti, la más divertida de mis abuelas, va dedicado este trabajo, aunque hace poco nos dejaste, estoy seguro de que estarías muy orgullosa de mi cuando me vieras graduado, así como yo estoy muy orgulloso de la abuela que tuve. Siempre estarás conmigo.

Abuela Mirian: Abuela, por complacerme en todo, porque eras tú quien se quedaba dormida cuando me leías cuentos para que yo me durmiera. Por ser todo lo que la palabra abuela encierra y la que todos quisieran tener.

Abuela Caro: A ti abuela, que eres la que menos veo y la que más lejos vives, por darte esos viajes conmigo hasta Soroa a pesar de tu edad cuando yo era chico, por darme todo tu cariño cuando estas junto a mí, por ser madre de mi madre y de mis tíos y tías.

Abuelo Pepe: A ti mi abuelo, por ser ejemplo de respeto y sacrificio, por la experiencia que nos das a todos los que estamos cerca de ti, por ser el padre de mi padre y de mi tía, muy orgulloso de ti.

A toda mi familia: Porque de todos he aprendido algo que me ha permitido llegar a donde estoy hoy, es una lástima que casi todos vivan lejos, me hubiera gustado mucho tenerlos más cerca a todos.

Amigos y amigas: A todos los del barrio, los que crecieron conmigo y aún lo hacen, a los de los camilitos, los más fieles que he conocido, y a todos con los que he compartido durante estos cinco años en la UCI, me llevo muy buenos recuerdos de todos ustedes.

RESUMEN

El desarrollo de las Tecnologías Informáticas de las Comunicaciones (TIC) ha posibilitado la utilización de aplicaciones informáticas para mostrar geografías en la Web. Con el desarrollo de las tecnologías Web y la realidad virtual se ha incrementado en número de sistemas informáticos que permiten mostrar elementos tridimensionales en los navegadores, contribuyendo al bajo consumo de hardware y a una mayor accesibilidad de este tipo de aplicaciones. La combinación de todos estos elementos nos da la posibilidad de llevar a la Web, espacios geográficos tridimensionales que permiten una mayor visualización de los mismos para la mejor comprensión del entorno.

Con el objetivo de que las personas tengan acceso a los antes expuesto es necesario desarrollar e implementar un escenario virtual tridimensional interactivo Web que permita la visualización de la geografía de la Universidad de las Ciencias Informáticas, mostrando algunos aspectos de esta en tres dimensiones y permitiendo el cálculo de distancias.

En el presente documento se efectúa un detallado análisis y diseño del sistema, así como un estudio de las herramientas informáticas más actuales para permitir el desarrollo de una aplicación de este tipo.

Palabras clave: Web, tridimensional, visualización.

ÍNDICE

INTRODUCCIÓN

El rápido y ascendente desarrollo de la informática en los últimos tiempos ha traído consigo que cada vez más personas y más esferas de nuestra vida se vean ligadas a ella, se crean oportunidades de trabajo, se abren nuevos mercados y vías de comercio, se establecen comunicaciones, y esto como consecuencia ha traído un rápido desarrollo y mayor calidad de las soluciones y aplicaciones que se presentan en la red.

Dentro de la amplia gama de aplicaciones que surgen en Internet a diario se ha notado un crecimiento de las Aplicaciones Enriquecidas de Internet, que son aplicaciones Web que combinan las ventajas que ofrecen las aplicaciones Web y las aplicaciones de escritorio tradicionales. En este grupo de aplicaciones ha comenzado a crecer el número de los llamados escenarios o paseos virtuales que muestran un sitio o recinto con un gran realismo. Estos que en su primer momento eran construidos a base de fotografías y secuencias de video y que solo mostraban dos dimensiones; han evolucionado y en la actualidad la gran mayoría se encuentran hechos con modelos tridimensionales que dan una visión más exacta y real, además de permitir una mayor interacción y desplazamiento del usuario por recintos, locales y hasta ciudades, creadas virtualmente.

La Universidad de las Ciencias Informáticas, surgida en el marco de la batalla de ideas librada por nuestro país, no está exenta al desarrollo y evolución de las tecnologías de la información y las comunicaciones, ha venido aplicando un modelo de formación en el que se vincula la docencia con el proceso productivo, está dividida en varios centros productivos distribuidos por las facultades, uno de ellos es el Centro de Informática Industrial (CEDIN) de la facultad 5 que se encuentra dividido en cuatro departamentos. Uno de ellos es Entornos Virtuales dentro del cual se encuentra la línea de Visualización Web, la cual tiene como objetivo enriquecer el conocimiento y aplicaciones sobre lo referente a tecnología Web en la UCI y nuestro país y se ha propuesto comenzar a adentrarse y a trabajar el uso de la tecnología Web.

Estas aplicaciones cada vez cuentan con detalles más realistas y precisión en su realización, pero un detalle interesante seria conocer la distancia entre objetos o lugares dentro de un escenario virtual, pocas veces visto o casi nunca visto anteriormente, por la gran cantidad de posibilidades que esto generaría.

En estos momentos por la importancia que tiene la UCI para muchas personas e instituciones, nacionales e internacionales se hace necesario desarrollar y aplicar a la Web un escenario tridimensional interactivo de un espacio de nuestra universidad. Permitiendo el cálculo de distancia entre puntos o sitios de referencia.

Dada esta **situación problemática** se formula el siguiente **problema científico:** ¿Cómo crear un escenario tridimensional interactivo Web que posibilite el cálculo de distancias?

Se constituye como **objeto de estudio:** las Aplicaciones Enriquecidas de Internet que incluyan escenarios tridimensionales interactivos.

Por lo que se propone como **objetivo general de la investigación:** Implementar un escenario tridimensional interactivo Web donde se posibilite el cálculo de distancia con altos niveles de eficiencia, flexibilidad y portabilidad.

Se plantea como **campo de acción:** Cálculo de distancias en escenarios tridimensionales Web.

Por lo que se deben llevar a cabo para dar cumplimiento a los objetivos planteados las siguientes **tareas de investigación**:

- Análisis del proceso de desarrollo de los escenarios tridimensionales interactivos.
- Comparación de las aplicaciones y tecnologías de desarrollo Web que permitan crear un modelo tridimensional interactivo.
- Investigación de algoritmos y fórmulas para calcular distancia.
- Construcción de los modelos tridimensionales que serán usados.
- Selección de la biblioteca que permita el satisfactorio desarrollo de la aplicación.
- Identificación de patrones y estilos que posibiliten del desarrollo satisfactorio de la aplicación.
- Implementación de las funcionalidades para dar cumplimiento a los requisitos que posee la aplicación.
- Realización de pruebas para verificar el correcto funcionamiento, cohesión y fortaleza de la aplicación.

Métodos científicos de investigación usados**:**

Métodos Teóricos:

Analítico - Sintético:

- Este método permitirá realizar una selección de las teorías y la bibliografía ya existentes sobre escenarios tridimensionales interactivos de donde se puedan obtener aspectos que puedan complementar o aportar elementos a la investigación, luego, realizar una síntesis de la misma.

Análisis Histórico - Lógico:

- Para analizar el proceso o etapas por el que han transitado los escenarios virtuales Web desde su surgimiento hasta la actualidad, tecnologías y aplicaciones usadas para la construcción de los mismos.

Métodos Empíricos:

Observación:

- Para analizar la forma en que se presentan los escenarios virtuales en la Web, las principales funcionalidades que estos presentan.

Este trabajo está estructurado en cuatro capítulos, en el primer capítulo se aborda la fundamentación teórica del presente trabajo, se hace un estudio sobre las herramientas existentes para desarrollar un escenario virtual tridimensional interactivo Web y se seleccionan las que van a ser utilizadas, exponiendo el porqué del uso de las mismas. En el capítulo dos se exponen algunas características del sistema, así como las soluciones técnicas que este presenta u algunos temas referentes a la metodología de desarrollo utilizada. En el tercer capítulo se expone el diseño utilizado para el desarrollo de la aplicación y se explican las iteraciones por las cuales transcurrió la aplicación durante la fase de implementación de la misma. En el cuarto capítulo se exponen las pruebas realizadas al sistema y los resultados alcanzados en estas.

Capítulo 1: Fundamentación teórica.

1.1 Introducción al capítulo.

En el presente capítulo se hace un estudio general de los escenarios tridimensionales interactivos Web, donde se abordaran los principales conceptos y términos encontrados en la investigación así como las herramientas y tecnologías utilizadas.

1.2 Aplicaciones Enriquecidas de Internet.

Para entender que son las Aplicaciones Enriquecidas de Internet primeramente debemos conocer el concepto de Aplicación (Termino Informático) y Aplicación Web que se presentan a continuación.

Aplicación:

Una aplicación, en términos de informática es un tipo de programa informático diseñado como herramienta para permitir y facilitar a un usuario realizar uno o diversos tipos de trabajos. (Parcher, 1997)

Estas son usadas para la automatización de ciertas tareas, como por ejemplo algunas de las más sencillas que se han realizado para la redacción de textos, realizar cálculos y juegos; como son el Bloc de Notas, la Calculadora y el Solitario desarrollados por Microsoft. Y algunas más complejas como serían herramientas de diseño, reproductores multimedia y programas de mantenimiento tales como Adobe Photoshop de Adobe, Reproductor de Windows Media de Microsoft y TuneUp Utilities de TuneUp Software. Para solo citar algunas de la amplia gama de aplicaciones que existen.

En ocasiones son encontradas varias aplicaciones con distintas funcionalidades formando un paquete o suite, tal es el caso de Adobe Suite y Microsoft Office.

Aplicación 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. 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. (Soriano, 2011)

En inglés se denomina “browser-based aplication”, es decir, aplicación basada en navegadores. Son programas que se diseñan para funcionar a través de un navegador de internet, es decir, son aplicaciones que se ejecutan de forma online. (Pereda, 2007)

Las aplicaciones Web son bastante populares debido a la facilidad de uso que presentan, ya que existe una independencia tanto de sistema operativo como de rendimiento y memoria en cuanto al hardware necesario para acceder a ellas. Son relativamente fáciles de actualizar, mantener, y no es necesaria su instalación en el ordenador del usuario. La disponibilidad de estas suele ser relativamente alta y es posible acceder a ellas prácticamente desde cualquier ordenador con acceso a Internet.

Ejemplo de estas podemos encontrar: tiendas en línea, blogs y webmails.

Aplicaciones Enriquecidas de Internet (RIA):

RIA es acrónimo de *Rich Internet Applications* (Aplicaciones de Internet Enriquecidas).

En las aplicaciones para Internet los requerimientos de usabilidad y “experiencia de usuario” están evolucionando. Es por esto que las limitaciones que presentan los interfaces puramente HTML y el modelo de programación HTTP con constantes llamadas al servidor no son adecuadas para muchas aplicaciones. Debido a esto Internet está cambiando, ha dejado de ser estática y un nuevo tipo de aplicaciones está tomando el relevo de las tradicionales aplicaciones Web: son las llamadas RIA (Rich Internet Applications). Las RIA son más atractivas visualmente, más interactivas y mucho más satisfactorias.

Una Aplicación Rica en Internet es enteramente un nuevo grupo de experiencias Web que involucra interacción, velocidad y flexibilidad. En los entornos RIA no se producen recargas de página, ya que desde el principio se carga toda la pagina y los datos necesarios para que aplicación funcione. Sólo se produce comunicación con el servidor solo cuando los datos son requeridos (on-demand), cuando se necesitan datos externos como datos de una base de datos o de otros ficheros externos.

Estas surgen debido a algunas ineficiencias que presentan las aplicaciones Web:

- Recarga continua de páginas que produce un tráfico muy alto entre el cliente y el servidor que a veces llega a recargar la página.

- Ofrecen menos funcionalidades que las aplicaciones de escritorio.
- Poca capacidad multimedia, ya que se hace necesario un reproductor externo para reproducir los contenidos.

Las aplicaciones de tipo RIA son un nuevo tipo de aplicaciones con más ventajas que las tradicionales aplicaciones Web. Esta surge como una combinación de las ventajas que ofrecen las aplicaciones Web y las aplicaciones de escritorio tradicionales. (Soriano, 2011)

Este tipo de aplicación trabaja en función de acortar la brecha entre las aplicaciones Web y las de escritorio.

¿Por qué Aplicaciones “Enriquecidas” de Internet?

Las aplicaciones Web se ejecutan solamente dependiendo de un navegador Web estándar, las RIA se ejecutan dependiendo del navegador pero en la mayoría de los casos necesitan además la instalación de un software o plugin en el ordenador donde se vallan a ejecutar. La causa por la cual usar este software es que hay funcionalidades no soportadas por los navegadores y su función es enriquecer las aplicaciones Web ofreciendo dichas funcionalidades. La principal funcionalidad que brindan estos “plugin” a los navegadores va a ser el trabajo interactivo con gráficos. El uso de estos, nos va a permitir reproducir música en línea, cargar y reproducir videos en la Web

En la actualidad existe una variedad de plataformas para crear aplicaciones de tipo RIA, entre las cuales podemos encontrar, Flash y Flex de Adobe, AJAX, Open Laszlo (herramienta Open Source), SilverLight de Microsoft y JavaFx Script de Sun Microsystems que son las más destacadas.

Ventajas de las RIA:

- No necesitan instalación (Si es necesario tener actualizado el navegador Web).
- Pueden ser utilizadas desde cualquier PC con conexión a Internet.
- No dependen de un sistema operativo específico.
- Mucha menos posibilidad de infección por virus respecto a las aplicaciones escritorio.

- Ofrecen aplicaciones interactivas que se ejecutan del lado del cliente sin enviar información al servidor.

1.3 Escenarios virtuales.

Los escenarios virtuales son simulaciones de espacios físicos reales o no reales, a los cuales el ser un humano no puede acceder por ninguno de sus 5 sentidos. Estos son generados por programas computacionales y son mostrados mediante máquinas que sean capaces de proyectar una realidad virtual.

En la actualidad estos se encuentran presentes en nuestra cotidianidad. Pueden ser observados en el fondo de muchas de las películas de ciencia ficción que muestran viajes espaciales, en videojuegos, incluso en el parte meteorológico de los noticieros.

1.3.1 Paseos o Visitas virtuales.

Los paseos o visitas virtuales son una simulación de locaciones existentes o inexistentes, en su mayoría existentes; usualmente construida a base de fotografías panorámicas o secuencias de video. Donde el usuario situado desde un punto puede girar 360 grados horizontalmente y 180 grados verticalmente y observar el entorno que le rodea, todo esto sin poder interactuar con el entorno y observándolo solamente en dos dimensiones.

Son una forma de publicidad altamente efectiva y atractiva para los usuarios, debido al gran atractivo visual que presentan. Aumentan notablemente la permanencia del usuario en la página, y en consecuencia, su atracción e interés por el lugar. El uso de esta tecnología aumenta notablemente el número de visitas a una página.

1.3.2 Paseos o Visitas virtuales tridimensionales interactivos.

Los paseos virtuales tridimensionales, son una simulación virtual de una locación existente o inexistente construidos a partir de un modelo tridimensional generado por computadora y diseñado por una persona o grupo de personas, surgen como una mejora de los anteriores paseos virtuales brindando a los usuarios y desarrolladores la posibilidad de poder interactuar con el entorno, los usuarios ya podrían desplazarse en un mundo tridimensional y observar los objetos en sus tres dimensiones dándoles mayor capacidad visual por el parecido a la realidad que estos presentan. Por otra parte los desarrolladores tienen más

posibilidades cuando debieran realizarse cambios en el entorno pues solo tendrían que generar otro modelo tridimensional.

Las representaciones en tres dimensiones ya son una auténtica realidad en Internet, en la actualidad podemos visitar y pasear por lugares como las ciudades de Nueva York y Paris, desplazarse por las calles e incluso ingresar en museos, hoteles y cafeterías

Usos y aplicaciones:

- Son un potente instrumento de visualización de la información, cualquiera que sea su naturaleza. Ofrecen multitud de ventajas a la hora de transmitir e interpretar información y se muestran como una herramienta de fácil manejo para el usuario.
- Visualizar una representación análoga a la realidad facilita la transmisión y compresión de los conceptos espaciales.
- Dada la heterogeneidad de usos de los mismos se utilizan en áreas tan diversas como la televisión, el turismo, los museos, las propiedades inmobiliarias, las nuevas infraestructuras o los planes urbanísticos.
- Permite valorar una obra civil (puentes, presas, viaductos, etc.) o un complejo proyecto urbanístico en su conjunto, viendo de qué manera afecta al entorno (impacto ambiental).
- Una representación del territorio en 3D es una herramienta formativa óptima para instruir en conceptos territoriales, ya que posibilita una experiencia de inmersión en los lugares que se pretende conocer.
- Puede ser utilizado tanto por un alumno en los primeros niveles de enseñanza escolar reconociendo la geografía de su país, hasta por un empresario para recorrer un recinto ferial u observar lo que podría ser la nueva sede de su empresa.

1.4 Metodologías de desarrollo de software.

Las metodologías de desarrollo de software son un marco de trabajo usado para estructurar, planificar y controlar un proyecto de desarrollo, que permite llevarlo a cabo con altas posibilidades de éxito. Los mismos definen un conjunto de actividades, acciones, tareas, fundamentos y productos de trabajo para desarrollar software de calidad. (Pressman, 2009)

1.4.1 RUP

Proceso Unificado de Rational o Rational Unified Process en inglés es un proceso de desarrollo de software que captura las mejores prácticas del conocimiento de líderes en ingeniería de software y proporciona a los equipos de desarrollo guías, estándares y recomendaciones para la construcción de software de alta calidad. Las mejores prácticas de desarrollo de software están documentadas como principios clave.

El Proceso Unificado de Rational es un modelo de proceso moderno que proviene del trabajo en el UML y el asociado Proceso de Desarrollo de Software. Reúne elementos de todos los modelos de procesos genéricos, iteraciones de apoyo e ilustra buenas prácticas en la especificación y el diseño. (Sommerville, 2005)

RUP es uno de los procesos más generales que existe, está enfocado a cualquier tipo de proyecto así no sea de software, se basa en la documentación generada en cada una de sus cuatro fases: 1. Intercepción (puesta en marcha), 2. Elaboración (definición, análisis y diseño), 3. Construcción (implementación), 4. Transición (fin del proyecto y puesta en marcha). En las cuales se ejecutaran varias iteraciones según el tamaño del proyecto. Ver anexo 1.

RUP se basa en Casos de Uso para describir lo que se tiene y las funcionalidades que se esperan del software, está orientado a la arquitectura del sistema a implementarse, documentándose de la mejor manera, basándose en UML (Unified Modeling Language - Lenguaje de Modelado Unificado).

1.4.2 XP

Las metodologías ágiles tienen un origen reciente en el entorno de la ingeniería de software comparada con las metodologías pesadas. Su origen está ligado a los constantes inconvenientes que se presentaban en proyectos con algunas características, en los cuales la utilización de metodologías pesadas era el motivo del fracaso. Estas proponen procesos que se adaptan y progresan con el cambio, llegando incluso hasta el punto de cambiar ellos mismos a diferencia de las metodologías pesadas que potencian la planificación detallada del desarrollo de software a largo plazo, pero que pueden venirse abajo frente a los cambios.

En la década del 90 surge XP (“eXtrme Programming” en inglés y “Programación Extrema” en español). Más que una metodología, XP se considera una disciplina, la cual esta sostenida por valores y principios propios de las metodologías ágiles. Se basa en la simplicidad, la comunicación y el reciclado continuo de código además en el trabajo orientado directamente al objetivo, basándose para esto en las relaciones interpersonales y en la velocidad de reacción para la implementación y para los cambios que puedan surgir durante el desarrollo del proceso.

XP resalta una serie de valores y principios que se deben de tener en cuenta y que deben ser llevados a la práctica, durante el desarrollo del proyecto. Más que una metodología XP se considera una disciplina sostenida por valores y principios propios de las metodologías ágiles. Existen cuatro valores puntuales para el desarrollo basado en metodologías ágiles que se exponen a continuación.

- Comunicación: En la metodología XP es muy importante que exista un ambiente de colaboración y comunicación al interior de equipo de desarrollo, así como en la interacción de este con el cliente. En XP la interacción con el cliente es tan estrecha, que es considerado parte del equipo de desarrollo.
- Simplicidad: Este valor se aplica en todos los aspectos de la programación extrema. Desde diseños muy sencillos donde lo más relevante es la funcionalidad necesaria que requiere el cliente, hasta la simplificación del código mediante la refactorización de actividades complejas, solo se desarrolla lo que el cliente demanda, de la forma más sencilla.
- Coraje: El equipo de desarrollo debe estar preparado para enfrentarse a los continuos cambios que se presentarán en el transcurso de la actividad. Cada integrante debe tener el valor de exponer los problemas o dudas que halle en la realización del proyecto. Aun con estas variaciones, las jornadas de trabajo deben proporcionar el máximo de rendimiento.
- Retroalimentación: Se presenta desde el comienzo del proyecto, ayuda a encaminarlo y darle forma. Este se presenta en los dos sentidos, por parte del equipo de trabajo hacia el cliente, con el fin de brindarle información sobre la evolución del sistema, y desde el cliente hacia el equipo en los aportes a la construcción del proyecto.

XP define Historias de Usuario como base del software a desarrollar. Estas historias van a ser descritas por el cliente, a partir de estas se crea un plan de liberación o entrega del software entre el equipo de desarrollo y el cliente. Para cada entrega se discutirán los objetivos de la misma, y se definirán las

iteraciones que deben ser de poco tiempo de duración (días o semanas) necesarias para cumplir con los objetivos de la entrega. Al final de cada iteración se muestra el resultado al cliente para que este los juzgue. Si el cliente queda satisfecho se definirán las siguientes iteraciones del proyecto, si esto no sucede se adaptara el plan de entrega hasta que el cliente lo apruebe y el software quede a su gusto.

Lo primero que se debe realizar antes de comenzar cada iteración será escribir las pruebas que se van a realizar antes de cada entrega para comprobar el correcto funcionamiento del software. Esto es muy importante en la metodología XP debido a que se apuesta por iteraciones cortas donde el cliente puede ver el resultado.

La implementación en XP se produce siempre en parejas, lo cual no se corresponde con el desarrollo de esta aplicación, pues el desarrollo de la misma será realizado por una sola persona, pero de todas formas describiremos el porqué de este desarrollo en pares pues es fundamental en el desarrollo de software basado en XP.

XP plantea que en la implementación deben de trabajar dos programadores en un ordenador, lo cual incrementaría la calidad del código. Pues trabajando en parejas se obtiene un diseño de mejor calidad, un código más organizado y con menores errores que si se trabajase solo, además de que un compañero puede ayudar a solucionar inconvenientes en tiempo de codificación, los cuales aparecen frecuentemente.

En esta metodología se enfatiza mucho los aspectos relacionados con las pruebas, clasificándolas en diferentes tipos y funcionalidades específicas, indicando quién, cuándo y cómo deben ser implementadas y ejecutadas. Solo se debe liberar una nueva versión del producto si este ha superado las pruebas en su totalidad. De no suceder esto se utilizara el resultado de las mismas para identificar el error y solucionarlo.

¿Por qué utilizar XP?

Para el desarrollo de la aplicación se decidió utilizar la metodología XP porque es una de las metodologías de desarrollo de software más exitosa en la actualidad utilizada para proyectos a corto plazo, equipos de trabajo de pocas personas y cuyo plazo de entrega sea en un periodo corto de tiempo. XP intenta minimizar el riesgo de fallo del proceso por medio de la disposición permanente de un representante del cliente a disposición del equipo de desarrollo, lo cual es muy positivo para el desarrollo de nuestra

aplicación ya que en este caso el cliente y el equipo de desarrollo son la misma persona. En el anexo 3 se muestra una tabla realizada por IBM sobre entrevistas a personas acerca de XP y en el anexo 4 se muestra una tabla comparativa entre las metodologías RUP y XP.

1.5 Herramientas CASE.

Las herramientas CASE (Ingeniería del Software Asistida por Computadora) son un amplio abanico de diferentes tipos de programas para ayudar a las actividades del proceso de software, como el análisis de requerimientos, el modelado de sistemas, la depuración y las pruebas. En la actualidad, todos los métodos vienen con tecnología CASE asociada, como los editores para las notaciones utilizadas en el método, módulos de análisis que verifican el modelo del sistema según las reglas del método y generadores de informes que ayudan a crear la documentación del sistema. Las herramientas CASE también incluyen un generador de código que automáticamente genera código fuente a partir del modelo del sistema y de algunas guías de procesos para los ingenieros de software. (Sommerville, 2005)

El uso de estas herramientas nos va a ayudar en el proceso de desarrollo del software en tareas como el proceso de realizar el diseño de un proyecto, cálculo de costes, detección de errores, entre otras y tiene como objetivos mejorar la productividad en el desarrollo y mantenimiento del software, crear una buena planificación para el desarrollo de proyectos, reducir el tiempo y costo en el desarrollo de aplicaciones.

Para el desarrollo de la presente aplicación se tuvieron en cuenta dos de estas herramientas, por las funcionalidades y facilidades que estas brindan, las cuales serán caracterizadas en este epígrafe para luego seleccionar la más idónea para ayudar en el desarrollo de este proyecto.

1.5.1 Rational Rose

Rose es una herramienta CASE para el modelado visual mediante UML de sistemas de software, permitiendo de esta forma especificar, analizar y diseñar sistemas antes de codificarlos. Posibilita la generación de código a partir de los modelos. La ingeniería de código (directa e inversa) es posible para ANSI C++, Visual C++, Visual Basic 6, Java, J2EE/EJB, CORBA, Ada 83, Ada 95, Bases de datos: DB2, Oracle, SQL 92, SQL Server, Sybase, Aplicaciones WEB. (Rational Rose Enterprise, 2007)

1.5.2 Visual Paradigm for UML

Visual Paradigm for UML (VP-UML) es una herramienta CASE orientado al trabajo con UML. Soporta los principales estándares de la industria tales como Lenguaje de Modelado Unificado (UML), *Systems Modeling Language* (SysML), entre otros. Brinda un conjunto de herramientas a los equipos de desarrollo de *software* necesario para la captura de requisitos, la planificación de controles, el modelado de clases, modelado de paquetes y modelado de datos. (Headquarters)

La herramienta está diseñada para una gran variedad de usuarios, incluyendo Ingenieros de Software, Analistas de Sistema, Arquitectos de Sistemas que esten interesados en construir sistemas de software.

¿Por qué utilizar Visual Paradigm for UML?

Visual Paradigm soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

1.6 Servidores Web.

Un servidor Web es un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario de Internet. El servidor web se encarga de contestar a estas peticiones de forma adecuada, entregando como resultado una página web o información de todo tipo de acuerdo a los comandos solicitados.

Gracias a los avances en conectividad y la gran disponibilidad de banda ancha, hoy en día es muy común establecer los servidores Web dentro de la propia empresa, sin tener que recurrir a caros alojamientos en proveedores externos. Esto es posible gracias a Apache, uno de los mejores y el más utilizado entre los servidores Web que existen, aunque existen otros de gran calidad como es el caso de Cherokee y Tomcat. Apache ha construido una gran reputación entre los servidores Web gracias a su gran estabilidad,

confiabilidad y el gran aporte del grupo de voluntarios que planean y desarrollan todo lo relativo a esta plataforma, desde la documentación hasta el mismo código en sí.

1.6.1 Apache

Apache es un servidor Web HTTP de código abierto que está disponible para las plataformas Windows, Linux y MacOS. Apache HTTP Server presenta muchas características y funciones que lo califican como un servidor robusto y rápido por lo cual fue elegido para albergar la aplicación. Posee gran popularidad entre los usuarios, lo que nos va a facilitar conseguir la ayuda y soporte al presentarse un problema.

1.6.2 Cherokee

Cherokee es un servidor Web multiplataforma, surge de la mano del desarrollador Álvaro López Ortega, quien buscaba desarrollar un servidor Web que fuera bastante rápido y funcional, pero a la vez no tan grande y pesado como el Apache. En la actualidad este servidor se presenta de manera libre y es desarrollado y mantenido por una amplia comunidad de desarrolladores. Factores que actúan a su favor, pero que no hacen frente al extenso tiempo de existencia del servidor Web Apache.

¿Por qué utilizar Apache?

- Corre en una multitud de sistemas operativos, lo que lo hace prácticamente universal.
- Apache es una tecnología gratuita de código fuente abierta. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este software de manera que si queremos ver que es lo que estamos instalando como servidor, lo podemos saber, sin ningún secreto, sin ninguna puerta trasera.
- Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor Web Apache. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que los instalemos cuando los necesitemos. Otra cosa importante es que cualquiera que posea una experiencia decente en la programación de C o Perl puede escribir un módulo para realizar una función determinada.
- Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.

- Tiene una alta configurabilidad en la creación y gestión de logs. Apache permite la creación de ficheros de log a medida del administrador, de este modo puedes tener un mayor control sobre lo que sucede en tu servidor.

1.7 Lenguajes de programación.

Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. (O'Reilly, 2010)

Los lenguajes de programación nos permiten 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.

1.7.1 ActionScript 3.0.

El lenguaje de programación usado para el desarrollo del escenario tridimensional interactivo es ActionScript en su versión 3.0 pues permite incrementar el performance y facilitar el desarrollo de complejas aplicaciones orientadas a objetos. Es un lenguaje más consistente y acorde con los estándares de la industria.

ActionScript 3.0 se basa en dos partes fundamentales: el núcleo del lenguaje y el API para Flash Player. Esta primera describe los elementos básicos del lenguaje como son los tipos de datos, expresiones, ciclos y condiciones; esta parte del lenguaje no ha cambiado mucho respecto a versiones anteriores. Y la segunda parte que describe las clases de ActionScript que serán corridas por el Flash Player.

Es un lenguaje de programación orientado a objetos, utilizado en especial en aplicaciones Web animadas realizadas en el entorno Adobe Flash. Fue lanzado con la versión 4 de Flash, y desde entonces hasta ahora, ha ido ampliándose poco a poco, hasta llegar a niveles de dinamismo y versatilidad muy altos en la versión 11 (Adobe Flash CS5) de Flash.

Ventajas de usar ActionScript 3.0:

Este lenguaje aumenta las posibilidades de creación de scripts de las versiones anteriores de ActionScript. Se ha diseñado para facilitar la creación de aplicaciones muy complejas con conjuntos de datos voluminosos y bases de código reutilizables y orientadas a objetos. Aunque no se requiere para el contenido que se ejecuta en Adobe Flash Player 9, ActionScript 3.0 permite introducir mejoras de rendimiento. El código ActionScript 3.0 puede ejecutarse con una velocidad diez veces mayor que el código ActionScript heredado.

¿Por qué utilizar ActionScript 3.0?

Se decidió utilizar ActionScript 3.0 por los beneficios que reporta a la aplicación y a continuación se exponen algunos de los siguientes:

- Seguridad: El código será más fácil de mantener, eliminando ambigüedades y permitiendo una escritura más clara y concisa.
- Simplicidad: El lenguaje será lo suficientemente intuitivo como para que el programador pueda escribir código sin tener que consultar el manual de referencia constantemente.
- Performance: El lenguaje habilitará a los programadores para escribir complejos programas que funcionen de forma más eficiente y responsable.
- Compatibilidad: El lenguaje estará provisto de una mayor compatibilidad con los estándares de la industria.

1.8 Bibliotecas 3D.

El termino biblioteca se una en la informática para definir un conjunto de subprogramas utilizados para desarrollar software. Las bibliotecas contienen código y datos, que proporcionan servicios a programas independientes, es decir, pasan a formar parte de éstos. Las bibliotecas en su mayoría no son ejecutables, sino que son un conjunto o paquete de archivos que sirven de apoyo o herramienta para la creación de software.

1.8.1 Sandy.

Es una potente librería de animación 3D desarrollada para Flash. Nos provee una gran variedad de utilidades para dotar a cualquier aplicación o juego realizado en Flash de interesantes efectos

tridimensionales. En el año 2005, Sandy fue el primer motor 3D en surgir para Flash. Fue desarrollado en ActionScript 2.0 que era el lenguaje script que estaba disponible para Flash en el momento en que surge. Por la naturaleza interpretada de ActionScript 2.0 y las capacidades limitadas de renderizar gráficos del Flash Player, las representaciones de este motor eran bastante básicas y limitadas.

1.8.2 PaperVision3D.

En una librería 3D para Flash de código abierto, liberada para Flash Player. Esta aparece en 2006 seguido de la biblioteca Sandy con la cual difiere en dos cosas fundamentales: PaperVision3D provee un acercamiento más simple a la hora de crear los elementos tridimensionales y que fue creado con ActionScript 3.0, un lenguaje más potente que la versión 2.0 del mismo.

En realidad no se instala. Se trata de una librería de clases de ActionScript de la que se servirá nuestra aplicación para implementar sus funciones.

1.8.3 Away3D.

Es un motor gráfico 3D de código abierto que surge como una rama del motor PaperVision3D en 2007, que rápidamente comenzó a evolucionar en una dirección propia pretendiendo estabilidad y facilidad de uso. Depende mucho de las contribuciones gratuitas de diseñadores y desarrolladores de la comunidad Flash del mundo.

Away3D es una potente biblioteca, para el desarrollo de elementos tridimensionales orientados a la Web, permite cargar una variedad de modelos 3D exportados desde una amplia gama de herramientas de modelado 3D. Permite la creación de estructuras tridimensionales sin necesidad de importarlas como es el caso de cubos, pirámides, conos, esferas, así como la aplicación de texturas a estos modelos.

¿Por qué utilizar Away3D?

- Más actualizaciones y consistencia que cualquier otro motor para Flash.
- Su código abierto permite la contribución para mejoras y corrección de errores.
- Permite importar una gran variedad de formatos de archivos tridimensionales.
- Actualmente su paquete de clases es más completo que el de los otros motores gráficos.

1.9 Framework de desarrollo.

Framework en el mundo del desarrollo de sistemas software no es más que una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta.

Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.

1.9.1 Flex 3.

Este framework agrupa una serie de tecnologías publicadas desde Marzo de 2004 por Macromedia para dar soporte al desarrollo y despliegue de Aplicaciones Enriquecidas de Internet, fue inicialmente liberado como una aplicación de la J2EE o biblioteca de etiquetas JSP que compilaba el lenguaje de marcas Flex (MXML) y ejecutaba mediante ActionScript aplicaciones Flash. Con Flex 3 los desarrolladores pueden usar los lenguajes XML, MXML y ActionScript 3.0 para describir interfaces de usuario y capturar las interacciones del usuario, tiene varios componentes y características que aportan funcionalidades tales como Servicios Web, objetos remotos, arrastrar y soltar, columnas ordenables, gráficas, efectos de animación y otras interacciones simples. Flex incluye una rica librería de clases de componentes para crear interfaces de usuario para ser usadas en aplicaciones web.

Proceso de desarrollo de una aplicación Flex:

(Datos tomados del archivo de ayuda de la versión 2.0 Beta 3)

- Definir un interfaz de aplicación usando un conjunto de componentes pre-definidos.
- Ordenar estos componentes en el diseño de la interfaz de usuario.
- Usar estilos y temas para definir el diseño visual.
- Añadir comportamiento dinámico (una parte de la aplicación interactuando con otra, por ejemplo).
- Definir y conectar a servicios de datos según sea necesario (servicios http).
- Compilar el código fuente en un archivo SWF que funcione en el reproductor Flash.

¿Por qué utilizar Flex?

- Soporta la creación de archivos estáticos que son compilados, y que pueden ser distribuidos en línea sin la necesidad de tener una licencia de servidor.
- El cliente solo carga la aplicación una vez, mejorando así el flujo de datos frente a aplicaciones basadas en HTML (PHP, ASP, JSP, CFMX), las cuales requieren de ejecutar plantillas en el servidor para cada acción.
- El lenguaje y la estructura de archivos de Flex buscan el desacoplamiento de la lógica y el diseño.
- El servidor Flex también actúa como un Gateway permitiendo al cliente comunicarse con servicios web XML y objetos remotos.
- Las aplicaciones desarrolladas sobre la plataforma Flex pueden interactuar con otras tecnologías del lado servidor.

1.10 Entorno de desarrollo.

Un entorno de desarrollo integrado (IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Algunos pueden soportar uno o varios lenguajes de programación, como es el caso de Visual Estudio y Eclipse.

1.10.1 Adobe Flash Builder 4.

Adobe Flash Builder (anteriormente Adobe Flex Builder) es un entorno de desarrollo integrado escrito en la plataforma Eclipse destinado para el desarrollo de Aplicaciones de Enriquecidas de Internet (RIA) multiplataforma y contenidos, utilizando el marco de trabajo de código abierto de Flex. Particularmente para la plataforma de Adobe Flash. Presenta nuevas capacidades para permitir codificación más rápida y productiva así como mejoras de desempeño.

Características de Adobe Flash Builder:

Además de los editores de código empotrados para MXML y ActionScript, Flash Builder soporta un editor WYSIWYG para modificar aplicaciones y componentes MXML. Flash Builder incluye un depurador interactivo integrado permitiendo a los desarrolladores pasar a través de la ejecución de código mientras inspeccionan las variables y visualizan las expresiones. La versión de 4 añadió soporte para análisis de

rendimiento. La vista del profiler muestra información estadística acerca de donde y cuanta memoria está siendo consumida además del tiempo de ejecución de llamadas a funciones.

¿Por qué utilizar Adobe Flash Builder 4?

- Está diseñado para ayudar a los desarrolladores de software a crear rápidamente Aplicaciones Enriquecidas de Internet.
- Está diseñado para crear aplicaciones multiplataforma utilizando el marco de trabajo de código abierto de Flex.
- Incluye compatibilidad con la codificación inteligente, la depuración y el diseño visual, y presenta potentes herramientas de prueba que agilizan el desarrollo y hacen que las aplicaciones tengan un rendimiento más elevado.

1.11 Herramientas de modelado 3D.

La informática hay ramas que se encargan de la creación de representaciones graficas mediante el uso del ordenador. Muchas son las aplicaciones o sectores a las que van a ir dirigidos estos diseños. Cada uno de estos sectores, cuenta con programas específicos adaptados a las necesidades concretas de los profesionales que los van a emplear, cada uno con sus ventajas y desventajas frente a los demás, pero con la posibilidad de realizar un trabajo de calidad. Las aplicaciones de modelado y animación en 3D poseen un extenso campo de aplicación, que va desde la publicidad, la cinematografía, la realización de videojuegos y arquitectura.

En este epígrafe se exponen algunas de estas herramientas, destacando el uso de Google SketchUp en la creación de la aplicación.

1.11.1 Blender.

Blender es un programa dedicado a la creación de gráficos y animaciones en tres dimensiones. Una de las mayores ventajas de Blender es que se trata de software libre, multiplataforma y gratuito. Si instalación es muy sencilla y el instalador no ocupa mucho espacio en disco duro. Integra un motor 3D para juegos con características como la detección de colisiones, recreaciones dinámicas y lógica. Un aspecto muy positivo a señalar es la comunidad que existe alrededor de este. Uno de los aspectos que más marcan

negativamente a esta aplicación es lo poca intuitiva que es su interfaz. Brinda la posibilidad de exportar los modelos 3D hacia los formatos *.3ds, *.dae, *.fbx, *.obj, y *.x3d.

1.11.2 Autodesk 3ds Max.

Autodesk 3ds Max es un programa para la creación de gráficos y animaciones tridimensionales, es uno de los programas de animación 3D más utilizados en la actualidad reconocido en el ámbito de los videojuegos. Dispone de una sólida capacidad de edición contando con más de 100 herramientas de modelado, escultura y manipulación de objetos. Existen infinidad de plugins que facilitan la creación de trabajos. Es un programa muy potente y estable, compatible con una gran variedad de formatos.

Pero presenta algunas desventajas como que solo está disponible para Windows, es decir no es multiplataforma, presenta un precio de adquisición bastante alto y los requerimientos de hardware que el programa necesita para su funcionamiento son bastante elevados.

1.11.3 AutoCAD.

AutoCAD es una aplicación de diseño para dibujo en dos y tres dimensiones, orientado al diseño arquitectónico. Trabaja imágenes de tipo vectorial, aunque también lo hace con mapas de bits. Posibilita el trabajo en dos dimensiones mediante el dibujo de figuras básicas o primitivas como líneas, arcos, rectángulos para y tridimensional mediante texturas y sólidos. Está orientado a la producción de planos. Es un programa bastante robusto y estable que está disponible para las plataformas Windows y MacOS. Brinda la posibilidad de exportar archivos con formato *.dwg,*.dwf,*.dxf,*.fbx.

1.11.4 Google SketchUp.

La mayoría de las herramientas de modelado 3D requieren al menos una base de conocimientos de dibujo. Sin embargo, Google SketchUp está pensado para que pueda ser utilizado por cualquier persona, con conocimientos básico de informática, ya que es una herramienta muy intuitiva, sobre todo en comparación con otros programas de dibujo 3D.

Google SketchUp es una aplicación para realizar diseños y modelado tridimensional, encaminado a la arquitectura e ingeniería civil. Permite una rápida conceptualización de entornos arquitectónicos,

especialmente edificaciones, siendo esto posible gracias a las nuevas herramientas incorporadas como son unir, intersecar, sustraer, recortar, y dividir sólidos.

Su instalación es muy fácil. Está disponible para las plataformas Windows y Mac OS. Actualmente existen dos versiones, la básica y la profesional. La primera es gratuita en su totalidad y la más indicada para principiantes. La versión profesional, tiene un coste muy elevado, y posee una serie de opciones inexistentes en la versión básica.

Algunas de las desventajas de esta versión básica de Google SketchUp es que carece de soporte técnico y presenta limitaciones respecto a otras herramientas. Tampoco puede exportar a 3D Studio Max o AutoCAD como sería de esperar, aunque si puede importar de estos y otros programas.

¿Por qué utilizar Google SketchUp?

- Es el más adecuado para principiantes.
- La versión básica, que es con la que trabajaremos es totalmente gratis.
- Es un programa intuitivo y relativamente fácil de usar lo que le da cierta ventaja sobre las otras herramientas de modelado tridimensional.
- Es idóneo para proyectos arquitectónicos y modelado de edificaciones.
- La aplicación incluye entre sus recursos un paquete de tutoriales interactivos para aprender a diseñar y modelar en el propio ambiente del programa.

1.12 Matemáticas y algoritmos.

En este epígrafe se expondrán algunas fórmulas matemáticas y algoritmos que se hacen necesarios para dar solución al presente trabajo.

1.12.1 Distancia entre dos puntos.

La distancia entre dos puntos del espacio euclídeo equivale a la longitud del segmento de recta que los une, expresado numéricamente.

Distancia entre dos puntos bidimensionales.

La distancia entre dos puntos $A(x_1,y_1)$ y $B(x_2,y_2)$ sería igual a:

$d = \text{sqrt}((x_2-x_1)^2 + (y_2-y_1)^2)$

Distancia entre dos puntos tridimensionales.

La distancia entre dos puntos $A(x_1,y_1,z_1)$ y $B(x_2,y_2,z_2)$ sería igual a:

$d = \text{sqrt}((x_2-x_1)^2 + (y_2-y_1)^2 + (z_2-z_1)^2)$

1.12.2 Algoritmo de caminos mínimos (Dijkstra).

El algoritmo de Dijkstra o algoritmo de caminos mínimos, es un algoritmo para determinar el camino más corto en un grafo dirigido y con pesos en cada arista. Consiste en ir explorando todos los caminos más cortos que parten de un vértice origen y que llevan a todos los demás vértices. El algoritmo se detiene cuando se obtiene el camino más corto.

Utilizaremos el algoritmo de Dijkstra pues presenta una complejidad o tiempo de ejecución de $O(n^2)$ que es menos al de Floyd $O(n^3)$.

Descripción del algoritmo:

Teniendo un grafo dirigido ponderado de N nodos no aislados, sea x el nodo inicial, un vector D de tamaño N guardara al final del algoritmo las distancias desde x al resto de los nodos.

1. Inicializar todas las distancias en D con un valor infinito relativo ya que son desconocidas al principio, exceptuando la de x que se debe colocar en 0 debido a que la distancia de x a x sería 0.
2. Sea a = x (tomamos a como nodo actual).
3. Recorremos todos los nodos adyacentes de a, excepto los nodos marcados, llamaremos a estos vi.
4. Si la distancia desde x hasta vi guardada en D es mayor que la distancia desde x hasta a sumada a la distancia desde a hasta vi; esta se sustituye con la segunda nombrada, esto es: si (Di > Da + d(a, vi)) entonces Di = Da + d(a, vi)
5. Marcamos como completo el nodo a.

6. Tomamos como próximo nodo actual el de menor valor en D (puede hacerse almacenando los valores en una cola de prioridad) y volvemos al paso 3 mientras existan nodos no marcados.

Una vez terminado al algoritmo, D estará completamente lleno.

1.13 Conclusiones del capítulo.

En el presente capítulo se expusieron las herramientas, metodologías y bibliotecas existentes que facilitan el desarrollo de un Escenario Virtual Tridimensional Interactivo Web, destacando las que serán usadas para el desarrollo de la presente aplicación y el porqué de su selección; así como otros aspectos relacionados al proceso de desarrollo del sistema. En el siguiente capítulo se hará una descripción del sistema, así como las fases por las cuales debe de transitar una solución informática al estar guiada por una metodología ágil como lo es XP.

Capítulo 2: El Sistema, basado en XP.

2.1 Introducción al capítulo.

En el presente capítulo se van a describir las características del sistema, y se mostrará la propuesta de solución para posibilitar un mayor entendimiento del funcionamiento de la aplicación. Se abordan temas teóricos sobre la metodología XP por la cual esta guiado el proceso de desarrollo de esta aplicación, donde se detallaran cada una de las fases por las cuales transitará nuestro sistema.

2.2 Descripción del Sistema.

La descripción inicial del sistema es una idea que generalmente se plantea antes de ser presentado el problema al equipo de desarrollo, que va a permitir al cliente tener conocimiento de las prestaciones y apariencia del software que desean. Esta descripción va a variar a medida que se realicen encuentros entre los clientes y el equipo de desarrollo.

En este trabajo el cliente forma parte del equipo de desarrollo uno de los requisitos fundamentales para llegar al éxito de un proyecto guiado por el uso de la metodología XP, lo cual facilita la determinación del alcance y el tiempo de entrega de la aplicación.

La aplicación a desarrollar es un Escenario Virtual Tridimensional Interactivo Web que permita:

- Girar la cámara 360 grados horizontal y verticalmente en el escenario.
- Acercar y alejar la cámara en el escenario.
- El cálculo, de distancia y camino mínimo entre puntos del escenario.

Para la interactividad del usuario con la aplicación, este debe girar la cámara horizontal y verticalmente mediante el uso de las teclas direccionales representadas en el teclado mediante las figuras: ←, ↑, →, ↓, de la zona de navegación del teclado; las cuales indican la dirección del movimiento (rotar vertical izquierda, rotar horizontal arriba, rotar vertical derecha, rotar horizontal abajo). Acercar y alejar la cámara hacia una dirección del escenario será posible a través de la utilización de las teclas PageUp y

PageDown. El cálculo de distancia y camino mínimo entre puntos del escenario se realizará mediante la selección de los mismos por el usuario usando para ello el botón izquierdo del mouse.

El sistema implementado está conformado por una pantalla principal donde se visualiza el escenario y 3 cuadros de texto para mostrar la información del mismo. En el primer cuadro de texto que se encuentra a la izquierda-arriba de la pantalla se mostrara la selección realizada por el usuario, en el segundo cuadro de texto que se encuentra en el centro-arriba de la pantalla se mostrarán los resultados de las acciones realizadas y el tercer cuadro de texto ubicado a la derecha-arriba de la pantalla se mostraran algunas instrucciones necesarias para la interacción con la aplicación. Ver anexo 5.

2.3 Requerimientos del sistema.

El término requisito o requerimiento se usa en la industria del software de una manera constante, en algunos casos un requisito es una declaración simplemente de alto nivel, abstracta que especifica funciones a cumplir o restricciones del sistema. Al otro extremo es una función formal y detallada de una función del sistema.

Los requisitos brindan una idea de las funciones que un sistema debe cumplir, así mismo como características que este va a presentar.

2.3.1 Requerimientos funcionales.

Éstas son declaraciones de servicios que el sistema debería proveer, cómo debería reaccionar el sistema para las entradas particulares y cómo debería comportarse el sistema en situaciones particulares. En algunos casos, los requisitos funcionales también explícitamente pueden declarar lo que el sistema no debería hacer. (Sommerville, Ingenieria de Software 8va Edicion, 2005)

Del cumplimiento de estas funcionalidades depende en alto grado el desarrollo exitoso de la aplicación de software.

A continuación se mencionan los requisitos funcionales que debe cumplir el Escenario Virtual Tridimensional Interactivo Web que vamos a desarrollar:

RF1: Acercar y alejar la cámara.

RF2: Rotar la cámara horizontal y verticalmente.

- **RF2.1:** Rotar la cámara verticalmente hacia la izquierda.
- **RF2.2:** Rotar la cámara horizontalmente hacia arriba.
- **RF2.3:** Rotar la cámara verticalmente hacia la derecha.
- **RF2.4:** Rotar la cámara horizontalmente hacia abajo.

RF3: Calcular camino mínimo entre puntos tridimensionales del escenario.

RF4: Calcular distancia entre dos puntos del escenario.

2.3.2 Requerimientos no funcionales.

Éstas son restricciones en los servicios o las funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, restricciones en el proceso de desarrollo y las normas. Los requisitos no funcionales a menudo se aplican al sistema como un todo, usualmente no se aplican a las características individuales del sistema o los servicios. (Sommerville, Ingenieria de Software 8va Edicion, 2005)

A continuación se detallan algunos de los requisitos no funcionales que debe cumplir la aplicación:

Usabilidad: La aplicación podrá ser utilizada a través de la red. La utilización del sistema no requiere de amplios conocimientos de informática por parte de los usuarios, solo un básico conocimiento del uso del teclado y el mouse. Cuenta con una ayuda para facilitar el uso de la misma.

Eficiencia o Rendimiento: Para la utilización de la aplicación debe contarse con un Microprocesador igual o superior a los Intel Pentium IV y poseer una capacidad de más de 512 MB de memoria RAM.

Fiabilidad: La tasa de fallos del sistema deberá ser nula.

Portabilidad: El sistema deberá funcionar en los sistemas operativos Windows XP, Windows Vista, Windows 7, cualquier distribución de Linux, y MAC. Siendo además posible el acceso al sistema a través de cualquier navegador actualizado o con la instalación del Adobe Flash Player 10 o una versión superior.

Interfaz o Apariencia: El sistema está diseñado para una resolución de pantalla de 1024x768 pixeles.

2.4 Proceso guiado por XP.

En un proceso guiado por la metodología XP se deben potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo y propiciando un buen clima de trabajo. XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes y la simplicidad en las soluciones implementadas.

Para desarrollar la presente aplicación se decidió tomar como guía la metodología XP, por las características que esta presenta y la adaptabilidad de la aplicación a esta metodología.

2.4.1 Fase: Exploración.

El ciclo de vida de la metodología XP consiste de seis fases: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

Exploración es la primera de estas fases y XP propone que a partir de esta comienza la construcción del producto, en esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. (Penades, 2008)

En esta fase los programadores son quienes estiman el tiempo de desarrollo basado en la información obtenida de las entrevistas con los clientes. Estas estimaciones suelen sufrir variaciones cuando se analizan detalladamente en cada iteración, ya que son primarias y vienen basadas en datos de muy alto nivel. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

2.4.1.1 Historias de Usuario.

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software, estas vienen siendo como los casos de uso en una metodología RUP. En las historias de usuario el cliente describe brevemente las características que el sistema debe poseer, escritas generalmente en un lenguaje natural y sin mucha terminología técnica.

El trabajo con las historias de usuario es bastante flexible y dinámico, debido a que estas pueden reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas en el transcurso del proyecto. Cada historia de usuario debe ser comprensible y delimitada para que pueda ser implementada en período de tiempo relativamente corto (semanas). (Penades, 2008)

Las historias de usuario son descompuestas en tareas de programación y asignadas a los programadores para ser implementadas durante una iteración. Cada historia de usuario conduce a la creación de un test de aceptación, que servirá para comprobar la correcta implementación de esta tarea.

Para la creación de una plantilla para contener las historias de usuarios se sugiere crear una ficha en la cual se encuentren los siguientes contenidos: nombre, numero de historia, usuario, tipo (nueva, corrección, mejora) estimación de esfuerzo en días, riesgo, prioridad técnica, descripción y comentarios.

Historias de Usuario de la presente aplicación:

Historia de Usuario	
Nro: 1	**Nombre de la HU:** Acercar y alejar la cámara.
Usuario: General.	**Estimación:** 7 días.
Tipo: Nueva	**Iteración:** 1ra.
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Alto.
Descripción: Será posible a través de la utilización de las teclas PageUp y PageDown	
Comentarios: Referencia al RF1: Acercar y alejar la cámara.	

Tabla: Historia de Usuario 1

Historia de Usuario	
Nro: 2	**Nombre de la HU:** Rotar la cámara horizontal y verticalmente.
Usuario: General.	**Estimación:** 14 días.
Tipo: Nueva	**Iteración:** 2da.
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Alto.
Descripción: Se debe controlar el desplazamiento mediante el uso de las teclas ←, ↑, →, ↓, de la zona de navegación del teclado; las cuales indican la dirección del movimiento (rotar a la izquierda, rotar hacia arriba, rotar a la derecha, rotar hacia abajo).	
Comentarios: Referencia al RF2: Rotar la cámara horizontal y verticalmente.	

Tabla: Historia de Usuario 2

Historia de Usuario	
Nro: 3	**Nombre de la HU:** Calcular camino mínimo entre puntos tridimensionales del escenario.
Usuario: General.	**Estimación:** 14 días.
Tipo: Nueva	**Iteración:** 3ra.

Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Alto.
Descripción: El usuario se le mostrara el camino mínimo y distancia a recorrer entre los dos puntos tridimensionales seleccionados por él.	
Comentarios: Referencia al RF3: Calcular camino mínimo entre puntos tridimensionales del escenario.	

Tabla: Historia de Usuario 3

Historia de Usuario	
Nro: 4	**Nombre de la HU:** Calcular distancia entre dos puntos del escenario.
Usuario: General.	**Estimación:** 14 días.
Tipo: Nueva	**Iteración:** 4ta.
Prioridad en negocio: Alta.	**Riesgo en desarrollo:** Alto.
Descripción: El cálculo de distancia se realizará mediante la selección de los dos puntos por el usuario usando para ello el botón izquierdo del mouse.	
Comentarios: Referencia al RF4: Calcular distancia entre dos puntos del escenario.	

Tabla: Historia de Usuario 4

2.4.2 Fase: Planificación de la entrega.

Esta es generalmente una fase corta, donde el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. (Penades, 2008)

A continuación se presenta una tabla que refleja la estimación de esfuerzo por historias de usuario.

Historias de Usuario	**Puntos Estimados**
Acercar y alejar la cámara.	1
Rotar la cámara horizontal y verticalmente	2
Calcular camino mínimo entre puntos tridimensionales del escenario	2
Calcular distancia entre dos puntos del escenario.	2
Total	7

Tabla: Estimación de esfuerzo por historias de usuario.

El plan de entregas se realiza en base a las estimaciones de tiempos realizadas por los desarrolladores, esto es uno de los temas más delicados del desarrollo de un proyecto de software, pues el cliente espera que para la fecha señalada su producto deba estar terminado. Por eso se hace muy importante tener bien claros las necesidades del cliente, la capacidad del equipo de desarrollo para completar las tareas en el período de tiempo con que se cuenta. En esta etapa debe de aumentar la curva de comunicación entre clientes y desarrolladores, pues pueden surgir cambios que afecten a la estimación de tiempo del proyecto y el cliente debe de ser informado inmediatamente, para llegar a un acuerdo y hacer un reajuste en la

entrega del proyecto., ya que no sería bueno defraudar al cliente luego de haber ocupado parte de su tiempo y recursos.

Entregable	Final 1ra Iteración 2da Sem Abril	Final 2da Iteración 4ta Sem Abril	Final 3ra Iteración 3ra Sem Mayo	Final 4ta Iteración 3ra Sem Mayo
EVTIW	**Versión 0.1**	**Versión 0.2**	**Versión 0.3**	**Versión 1.0**

Tabla: Plan de Entregas.

2.4.3 Fase: Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción. (Penades, 2008)

En el desarrollo de esta aplicación, esta fase va a contar con cuatro iteraciones implementándose en cada una de ellas una historia de usuario que deben de ser implementadas en orden de prioridad, que no será así, pues para la realización de un escenario virtual tridimensional interactivo web que posibilite el cálculo de distancias, es más conveniente implementar el escenario para luego realizar el cálculo de distancias, dejándose las historias de usuario de mayor importancia para la última iteración. Al final de cada iteración se liberará una versión beta de la aplicación para ser mostrada al cliente y antes de pasar a la siguiente se le realizaran pruebas a la versión obtenida y solo después de que pase estas pruebas, se pasara a la próxima iteración.

A partir del comienzo de la primera iteración se comenzara con el diseño e implementación de la aplicación, cuestiones que serán el tema del próximo capítulo y donde se abordaran en profundidad.

A continuación se presenta en la siguiente tabla el plan de iteraciones de esta aplicación.

Iteración	Historias de Usuario a implementar	Duración de la iteración
Iteración # 1	Rotar la cámara horizontal y verticalmente.	1 Semana
Iteración # 2	Acercar y alejar la cámara al escenario.	2 Semanas
Iteración # 3	Calcular camino mínimo entre puntos tridimensionales del escenario.	2 Semanas
Iteración # 4	Calcular distancia entre dos puntos del escenario.	2 Semanas

Tabla: Iteraciones.

2.4.4 Fases: Producción, Mantenimiento y Muerte del Proyecto.

En este epígrafe abordaremos de manera resumida estas tres fases ya que la fase de Mantenimiento y Muerte del Proyecto, no se llevaran a cabo, puesto que esta aplicación informática se plantea como solución para un trabajo de tesis y no para un proyecto comercial.

La fase de **Producción** requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Estas pruebas y revisiones serán realizadas en el capítulo cuarto del presente trabajo. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

Las ideas que en esta fase son propuestas y las sugerencias serán documentadas para su posterior implementación que podría ser durante la fase de mantenimiento.

En la fase de **Mantenimiento** se plantea que: Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura. (Penades, 2008)

La **Muerte del Proyecto** sucede cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

2.5 Conclusiones del capítulo.

En este capítulo se han analizado los procesos por los cuales debe transitar la aplicación a desarrollar y se ha dado una breve descripción del sistema, se ha elaborado una propuesta que satisface las necesidades del cliente y se desarrollaron las dos primeras fases propuestas por la metodología usada, obteniéndose de estas las historias de usuario que van a ser implementadas durante el ciclo de implementación y que facilitaron al cliente el entendimiento de los requerimientos que debe de cumplir la solución informática a realizar. También se obtuvo un plan de iteraciones y un plan de entrega, para guiar el proceso de diseño e implementación que van a ser profundizados en el siguiente capítulo.

Capítulo 3: Diseño e Implementación.

3.1 Introducción al capítulo.

XP propone que el diseño debe ser claro y lo más sencillo posible, pues de un buen diseño depende el correcto funcionamiento lógico del sistema y la implementación será la columna vertebral de cualquier sistema guiado por XP, de ahí la importancia que se le otorga. En el presente capítulo se describe detalladamente los aspectos relacionados al diseño de la aplicación y las cuatro iteraciones realizadas durante el proceso de implementación.

3.2 Diseño

En un proceso guiado por la metodología XP se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. El diseño crea una estructura que organiza la lógica del sistema, un buen diseño permite que el sistema crezca con cambios en un solo lugar. Los diseños deben de ser sencillos, si alguna parte del desarrollo es compleja debería dividirse en varias partes. La complejidad innecesaria y el código extra debe ser removido inmediatamente, ya que esto da la posibilidad de ganar en tiempo y eficiencia. (Solis, 2003)

3.2.1 Descripción de la Arquitectura.

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 la aplicación tenga las mejores condiciones ya que es la columna vertebral de la aplicación. (Reinoso, 2007)

Actualmente la arquitectura está en desarrollo. Este año se defendieron 3 tesis, las cuales están destinadas a definir una arquitectura para el desarrollo de estas aplicaciones basadas un estilo arquitectónico DSSA o arquitectura de software de dominio específico. Como no existe una arquitectura definida para el desarrollo de este tipo de aplicaciones, se determinó usar una variante básica del estilo cliente-servidor.

En la actualidad la arquitectura más encontrada en el desarrollo de aplicaciones de tipo RIA es la Cliente-Servidor que define dos tipos de entidades diferenciadas: clientes y servidores, cada una cumpliendo funciones diferentes. Donde el cliente inicia el dialogo a través de peticiones realizadas al servidor, el cual da respuesta a las peticiones recibidas, este constituye la parte pasiva ya que espera las peticiones de los clientes y procesa las mismas para enviar las respuestas. La ubicación del cliente es independiente a la ubicación del servidor y generalmente los clientes se encuentran en ordenadores personales.

La aplicación desarrollada, presenta una arquitectura Cliente-Servidor ya que los archivos fuente son almacenados en el servidor (archivos de extensión: *.swf, *.html) y la llamada inicial será a través de una estación cliente con un navegador Web enriquecido con Adobe Flash Placer 10 o una versión superior. Ver anexo 6.

La función de Adobe Flash Placer es manejar los cambios que se producen en la pantalla, produciendo las modificaciones requeridas solamente sobre aquellos elementos que necesitan ser actualizados.

3.2.2 Patrones de diseño.

Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo ni siquiera dos veces de la misma forma. (Alexander, 1979)

Un patrón en programación orientada a objetos es una descripción de diversos objetos y clases preparados para resolver un problema de diseño general aplicado a un contexto específico. ActionScript es un lenguaje de programación cada vez más sofisticado y con muchas posibilidades. La versión 3 mejora el rendimiento y ofrece nuevas inclusiones como el uso de expresiones regulares y nuevas formas de empaquetar las clases. Y si usamos los patrones de diseño, la optimización del código será mucho más efectiva, ya que podremos reutilizarlos según nos convenga. La utilización de patrones de diseño nos permite solucionar problemas comunes en la programación, optimizar código y aumentar la productividad.

Para lograr una mayor calidad en el diseño, se tuvieron en cuenta un conjunto de patrones los cuales proporcionan respuesta a un conjunto de problemas similares:

Prototipo: Se manifiesta cuando se crean nuevos objetos clonándolos de una instancia ya existente, en nuestra aplicación se muestra al crear un objeto contenido dentro del paquete “primitives “ de Away3D, como es el caso del plano y los cubos creados.

Experto en Información: Este patrón está relacionado con la asignación de responsabilidades y nos plantea que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. Se pone de manifiesto en la clase “Demo8.as”, donde los métodos implementados van a trabajar con la información de esta clase.

Creador: Este patrón nos ayuda a identificar quién debe ser el responsable de la creación nuevos objetos o clases y plantea que la nueva instancia deberá ser creada por la clase que: Tiene la información necesaria para realizar la creación del objeto, o usa directamente las instancias creadas del objeto, o almacena o maneja varias instancias de la clase contiene o agrega la clase. Este patrón se pone de manifiesto en la mayoría de las aplicaciones de programadas ActionScript 3 ya que este crea instancias de clases importadas desde las librerías de Flash o de Away3D.

3.2.3 Diagrama de clases del sistema.

El diagrama de clases provee una vista del sistema describiendo las clases y atributos que lo componen, así como las relaciones entre ellos. Provee una amplia variedad de usos desde modelar la estructura de datos del dominio específico hasta detallar nuestro sistema. (Visual Paradigm)

Los diagramas de clase son importantes no solo para la visualización, especificación y documentación del modelo estructural, sino también para la construcción de sistemas ejecutables. Ingeniería hacia delante e ingeniería inversa.

A continuación se representa el diagrama de clases que representa a nuestra aplicación, para un mejor entendimiento de la misma.

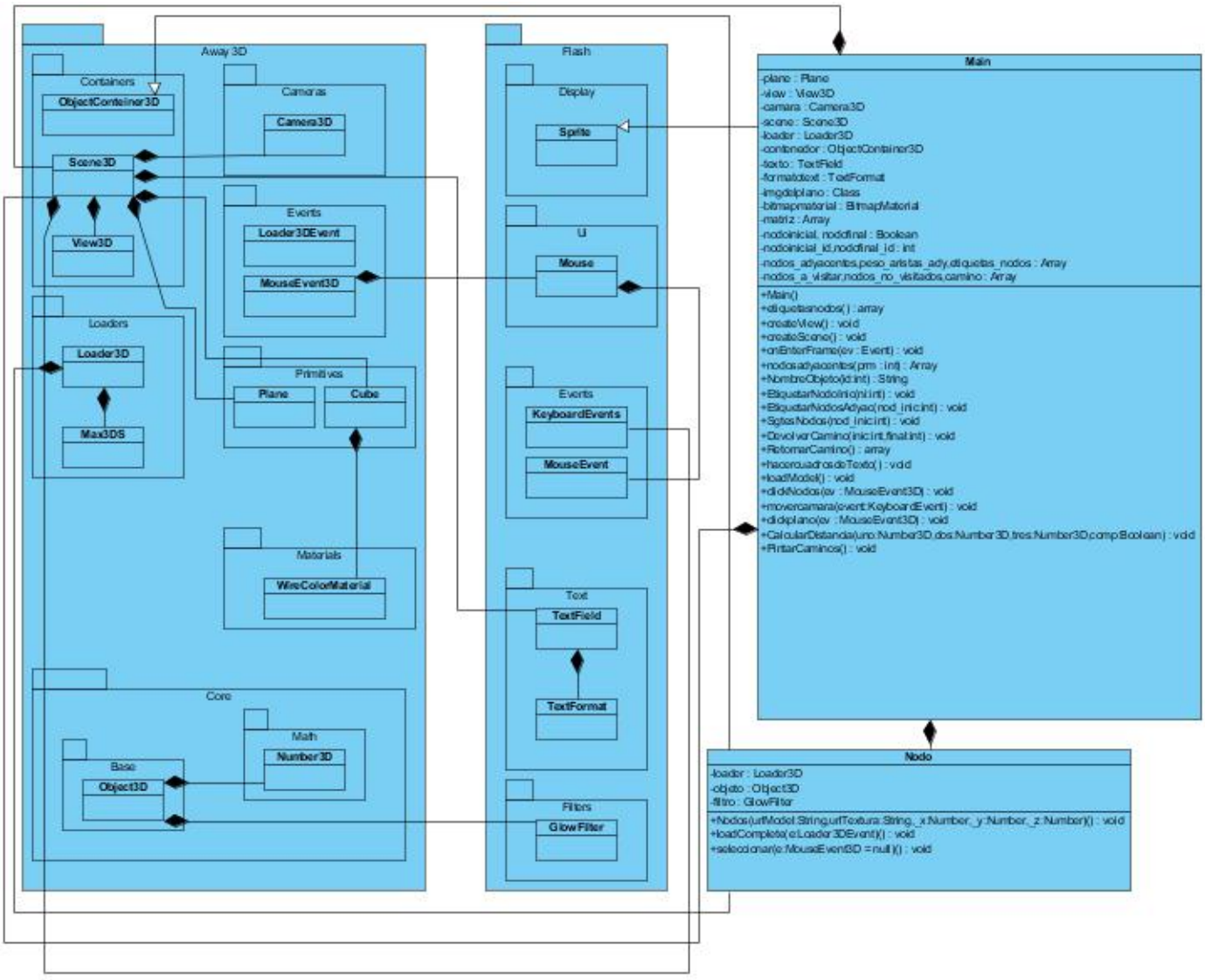

Figura: Diagrama de Clases del Sistema.

3.3 Implementación.

En las metodologías pesadas, la implementación es un proceso al cual solo se llega después de largas fases de análisis y diseño de las que queda una gran cantidad de documentación a partir de la cual el proceso de implementación es relativamente sencillo. En XP el proceso es muy diferente. Prácticamente desde un principio se inicia con la codificación, favoreciendo el logro del objetivo de estar haciendo entregas frecuentes al cliente. (Luis Miguel Echeverry, 2007)

La implementación es la única actividad de la que no se puede prescindir porque sin código fuente no hay programa. Por tanto es necesario codificar y plasmar las ideas a través del código.

Para una producción efectiva la metodología XP plantea una serie de prácticas básicas que deben tenerse en cuenta durante la implementación para alcanzar el éxito en un proyecto. A continuación se hace mención de estas:

Recodificación: La recodificación es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad y simplificarlo. Esto a veces nos puede llevar a hacer más trabajo del necesario, pero a la vez estaremos preparando nuestro sistema para que en un futuro acepte nuevos cambios y pueda albergar nuevas características.

Programación en Pares: Todo el código de producción lo escriben dos personas frente al ordenador, con un sólo ratón y un sólo teclado. Cada miembro de la pareja juega su papel: uno codifica en el ordenador y piensa la mejor manera de hacerlo, el otro piensa más estratégicamente. Las principales ventajas de introducir esta práctica es que muchos errores son detectados conforme son introducidos en el código. El emparejamiento puede ser dinámico combinando diferentes niveles de experiencia y áreas de trabajo.

Nota: Esta práctica no se lleva a cabo durante la implementación de nuestro trabajo, puesto que el mismo solo está compuesto por una sola persona.

Propiedad Colectiva: Cualquiera que crea que puede aportar valor al código puede cambiarlo, ningún miembro del equipo es propietario del código. Si hacemos el código propietario, y necesitamos de su autor para que lo cambie entonces estaremos alejándonos cada vez más de la comprensión del problema. Esta práctica motiva a todos a contribuir con nuevas ideas en todos los segmentos del sistema, evitando a la vez que algún programador sea imprescindible para realizar cambios en alguna porción del código.

Integración Continua: El código debe ser integrado como mínimo una vez al día, y realizar las pruebas sobre la totalidad del sistema, las cuales deben ser aprobadas para que el nuevo código sea integrado definitivamente. La integración continua a menudo reduce la fragmentación de los esfuerzos de los desarrolladores por falta de comunicación sobre lo que puede ser reutilizado o compartido.

Cliente In-situ: Un cliente real debe sentarse con el equipo de programadores, estar disponible para responder a sus preguntas, resolver discusiones y fijar las prioridades. Gran parte del éxito del proyecto XP se debe a que es el cliente quien conduce constantemente el trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada.

40 Horas Semanales: Se debe trabajar un máximo de 40 horas por semana. No se deben trabajar horas extras en dos semanas seguidas. Si esto ocurre, probablemente está ocurriendo un problema que debe corregirse. El trabajo extra desmotiva al equipo. Los proyectos que requieren trabajo extra para intentar cumplir con los plazos suelen al final ser entregados con retraso. Es preferible replantear los plazos a trabajar horas extras.

Estándares de Codificación: Si los programadores van a estar tocando partes distintas del sistema, intercambiando compañeros, haciendo recodificación, debemos de establecer un estándar de codificación aceptado e implantado por todo el equipo. XP enfatiza la comunicación de los programadores a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación. Los estándares de programación mantienen el código legible para los miembros del equipo, facilitando los cambios.

En el código de la aplicación desarrollada se usaron los siguientes estilos de código:

Todas las variables creadas deben ser escritas en letra minúsculas, ejemplo:

private var matriz:Array;

private var plane:Plane;

El nombre de todos los métodos creados, debe de comenzar con letra mayusculas:

public function NombreObjeto(id:int):String

public function Etiquetasnodos():Array

Al abrir y cerrar bloques de código las líneas en las que se encuentran las llaves, no va más código, ejemplo:

```
public function CreateScene() : void
{
scene = new Scene3D();
}
```

Ante la existencia de líneas de código muy largas, estas van a ser fraccionadas después de un operador:

```
texto2.text = "Controles:" + " " + "Flecha Arriba y Abajo: Rotar cámara horizontal"+
        "Flecha Izq y Der: Rotar cámara vertical"+ "  " + "PageUp y Page Down: Acercar y Alejar cámara";
```

De la correcta aplicación de todas estas prácticas definidas por XP, depende el cumplimiento de los requerimientos especificados en las historias de usuario del epígrafe 2.4.1.1. En los siguientes epígrafes del presente capítulo se exponen detalladamente las cuatro iteraciones generadas en la fase de Planificación.

3.3.1 1ra Iteración.

Al finalizar esta primera iteración ya debe de quedar implementada la Historia de Usuario: “Rotar la cámara horizontal y verticalmente” y una primera versión de la aplicación, que le permitirá al usuario tener una visión del escenario de la aplicación y rotar la cámara a través del plano principal mediante la utilización de las teclas ←, ↑, →, ↓. Al concluir esta primera iteración se obtendrá una estructura básica del sistema.

Durante la implementación de esta primera iteración se llevaron a cabo las siguientes tareas:

Tarea 1: Crear clase principal.

Tarea 2: Crear la vista.

Tarea 3: Crear la escena.

Tarea 4: Crear la cámara.

Tarea 5: Crear funcionalidad que permita el movimiento de la cámara alrededor del escenario.

Tarea 6: Crear el plano sobre el cual va a estar contenido el escenario.

Tarea 7: Tratamiento de imagen 2D que conformara el mapa del plano.

Tarea	
Nro. De Tarea: 1	**Nro. de HU:** 1
Nombre Tarea: Crear clase principal.	
Tipo de Tarea: Desarrollo.	**Estimación:** 1 día.
Fecha de Inicio: 1/4/2011	**Fecha de Fin:** 4/4/2011
Responsable: Carlos Vallejo Vigoa.	
Comentarios: Crear la clase principal y posibles métodos a implementar posteriormente.	

Tabla: Tarea 1, perteneciente a Historia de Usuario 1.

Tarea	
Nro. de Tarea: 2	**Nro. de HU:** 1
Nombre Tarea: Crear clase principal.	
Tipo de Tarea: Desarrollo.	**Estimación:** 1 día.
Fecha de Inicio: 4/4/2011	**Fecha de Fin:** 5/4/2011
Responsable: Carlos Vallejo Vigoa.	

Comentarios: Crear la vista.

Tabla: Tarea 2, perteneciente a Historia de Usuario 1.

Tarea	
Nro. de Tarea: 3	**Nro. de HU:** 1
Nombre Tarea: Crear la escena.	
Tipo de Tarea: Desarrollo.	**Estimación:** 1 día.
Fecha de Inicio: 5/4/2011	**Fecha de Fin:** 6/4/2011
Responsable: Carlos Vallejo Vigoa.	
Comentarios: Crear la vista.	

Tabla: Tarea 3, perteneciente a Historia de Usuario 1.

Tarea	
Nro. de Tarea: 4	**Nro. de HU:** 1
Nombre Tarea: Crear la cámara	
Tipo de Tarea: Desarrollo.	**Estimación:** 1 día.
Fecha de Inicio: 6/4/2011	**Fecha de Fin:** 7/4/2011
Responsable: Carlos Vallejo Vigoa.	
Comentarios: Crear la cámara.	

Tabla: Tarea 4, perteneciente a Historia de Usuario 1.

Tarea	
Nro. de Tarea: 5	**Nro. de HU:** 1
Nombre Tarea: Crear funcionalidad que permita el movimiento de la cámara alrededor del escenario.	
Tipo de Tarea: Desarrollo.	**Estimación:** 1 día.
Fecha de Inicio: 7/4/2011	**Fecha de Fin:** 8/4/2011
Responsable: Carlos Vallejo Vigoa.	
Comentarios: Implementar la funcionalidad que permita el movimiento alrededor del escenario, mediante el uso de las teclas ←, ↑, →, ↓.	

Tabla: Tarea 5, perteneciente a Historia de Usuario 1.

Tarea	
Nro. de Tarea: 6	**Nro. de HU:** 1
Nombre Tarea: Crear el plano sobre el cual va a estar contenido el escenario.	
Tipo de Tarea: Desarrollo.	**Estimación:** 1 día.
Fecha de Inicio: 8/4/2011	**Fecha de Fin:** 11/4/2011
Responsable: Carlos Vallejo Vigoa.	
Comentarios: Implementación del plano sobre el cual va a estar contenido el plano	

Tabla: Tarea 6, perteneciente a Historia de Usuario 1.

Tarea

Nro. de Tarea: 7	**Nro. de HU:** 1
Nombre Tarea: Tratamiento de imagen 2D que conformara el mapa del plano.	
Tipo de Tarea: Desarrollo.	**Estimación:** 1 día.
Fecha de Inicio: 11/4/2011	**Fecha de Fin:** 12/4/2011
Responsable: Carlos Vallejo Vigoa.	
Comentarios: Tratamiento de imagen 2D en Adobe Photoshop que conformara el mapa del plano.	

Tabla: Tarea 7, perteneciente a Historia de Usuario 1.

3.3.2 2da Iteración.

En esta 2da iteración se implementará la Historia de Usuario: “Acercar y alejar la cámara al escenario”, esta historia de usuario genera varias tareas, mediante las cuales al finalizar su cumplimiento dejaran conformado el escenario en su totalidad, faltándole las funcionalidades contenidas en las historias de usuario números 3 y 4. Al finalizar esta iteración el escenario contara con nuevas funcionalidades y el cliente podrá observarlo en su totalidad.

Las tareas a llevar a cabo en esta iteración serán:

Tarea 8: Acercar y alejar la cámara del escenario.

Tarea 9: Crear modelos tridimensionales para ser cargados en el escenario.

Tarea 10: Crear las texturas para aplicar a modelos tridimensionales.

Tarea 11: Aplicar texturas a los modelos tridimensionales.

Tarea

Nro. de Tarea: 8	**Nro. de HU:** 2
Nombre Tarea: Acercar y alejar la cámara al escenario.	
Tipo de Tarea: Desarrollo.	**Estimación:** 1 día.
Fecha de Inicio: 12/4/2011	**Fecha de Fin:** 13/4/2011
Responsable: Carlos Vallejo Vigoa.	
Comentarios: Implementar la funcionalidad que permita acercar y alejar la cámara al escenario mediante el uso de las teclas PageUp y PageDown.	

Tabla: Tarea 8, perteneciente a Historia de Usuario 2.

Tarea	
Nro. de Tarea: 9	**Nro. de HU:** 2
Nombre Tarea: Crear modelos tridimensionales para ser cargados en el escenario.	
Tipo de Tarea: Desarrollo.	**Estimación:** 5 días.
Fecha de Inicio: 13/4/2011	**Fecha de Fin:** 20/4/2011
Responsable: Carlos Vallejo Vigoa.	
Comentarios: Crear los modelos tridimensionales que van a ser utilizados en el EVTIW, mediante la herramienta Google SketchUp. Los modelos deben de tener la menor cantidad de polígonos posibles.	

Tabla: Tarea 9, perteneciente a Historia de Usuario 2.

Tarea	
Nro. de Tarea: 10	**Nro. de HU:** 2

Nombre Tarea: Crear texturas para aplicar a los modelos tridimensionales.	
Tipo de Tarea: Desarrollo.	**Estimación:** 5 días.
Fecha de Inicio: 20/4/2011	**Fecha de Fin:** 27/4/2011
Responsable: Carlos Vallejo Vigoa.	
Comentarios: Crear las texturas que van a ser aplicadas a los modelos tridimensionales mediante el uso de la herramienta Adobe Photoshop.	

Tabla: Tarea 10, perteneciente a Historia de Usuario 2.

Tarea	
Nro. de Tarea: 11	**Nro. de HU:** 2
Nombre Tarea: Aplicar texturas a los modelos tridimensionales.	
Tipo de Tarea: Desarrollo.	**Estimación:** 3 días.
Fecha de Inicio: 27/4/2011	**Fecha de Fin:** 29/4/2011
Responsable: Carlos Vallejo Vigoa.	
Comentarios: Aplicar texturas a modelos tridimensionales mediante la herramienta 3D's Max.	

Tabla: Tarea 11, perteneciente a Historia de Usuario 2.

3.3.3 3ra Iteración.

En esta iteración se implementara la Historia de Usuario: "Calcular camino mínimo entre puntos tridimensionales del escenario", la cual genera tareas que agregan funcionalidades a la aplicación.

Tareas generadas en la tercera iteración:

Tarea 12: Calcular camino mínimo entre los puntos tridimensionales.

Tarea 13: Revisar el código generado por la primera y segunda iteración.

Tarea	
Nro. de Tarea: 12	**Nro. de HU:** 3
Nombre Tarea: Calcular camino mínimo entre los puntos tridimensionales.	
Tipo de Tarea: Desarrollo.	**Estimación:** 12 días.
Fecha de Inicio: 29/4/2011	**Fecha de Fin:** 17/5/2011
Responsable: Carlos Vallejo Vigoa.	
Comentarios: Implementación de una versión del algoritmo Dijkstra que devuelva el camino a seguir, además de la longitud del camino a seguir.	

Tabla: Tarea 12, perteneciente a Historia de Usuario 3.

Tarea	
Nro. de Tarea: 13	**Nro. de HU:** 3
Nombre Tarea: Revisar el código generado por la primera y segunda iteración.	
Tipo de Tarea: Desarrollo.	**Estimación:** 2 días.
Fecha de Inicio: 17/5/2011	**Fecha de Fin:** 19/5/2011
Responsable: Carlos Vallejo Vigoa.	
Comentarios: Revisión del código generado en las iteraciones primera y segunda para evitar excesos.	

Tabla: Tarea 13, perteneciente a Historia de Usuario 3.

3.3.4 4ta Iteración.

Al finalizar esta cuarta iteración quedarán implementadas las cuatro historias de usuario y a su vez todas las funcionalidades del sistema y se obtendrá una versión final y estable de la aplicación.

Esta iteración genera las siguientes tareas:

Tarea 14: Calcular distancia entre dos puntos del escenario.

Tarea 15: Revisar el código generado por la tercera y cuarta iteración.

Tarea 16: Optimizar el código generado durante el desarrollo de la aplicación.

Tarea	
Nro. de Tarea: 14	**Nro. de HU:** 4
Nombre Tarea: Calcular distancia entre dos puntos del escenario.	
Tipo de Tarea: Desarrollo.	**Estimación:** 6 días.
Fecha de Inicio: 19/5/2011	**Fecha de Fin:** 27/5/2011
Responsable: Carlos Vallejo Vigoa.	
Comentarios: Los puntos serán tridimensionales, por lo tanto se debe obviar la variable sobre la cual no está contenido el plano. Hallar proporción a la distancia real.	

Tabla: Tarea 14, perteneciente a Historia de Usuario 4.

Tarea	
Nro. de Tarea: 15	**Nro. de HU:** 4
Nombre Tarea: Revisar el código generado por la tercera y cuarta iteración.	

Tipo de Tarea: Desarrollo.	**Estimación:** 2 días.
Fecha de Inicio: 27/5/2011	**Fecha de Fin:** 31/5/2011
Responsable: Carlos Vallejo Vigoa.	
Comentarios: Revisar el código generado por la tercera y cuarta iteración para buscar excesos de códigos innecesarios.	

Tabla: Tarea 15, perteneciente a Historia de Usuario 4.

Tarea	
Nro. de Tarea: 16	**Nro. de HU:** 4
Nombre Tarea: Optimizar el código generado durante el desarrollo de la aplicación.	
Tipo de Tarea: Desarrollo.	**Estimación:** 6 días.
Fecha de Inicio: 31/5/2011	**Fecha de Fin:** 8/5/2011
Responsable: Carlos Vallejo Vigoa.	
Comentarios: Analizar el código generado durante el desarrollo de la aplicación en busca de optimizarlo para un mejor funcionamiento.	

Tabla: Tarea 16, perteneciente a Historia de Usuario 4.

3.4 Conclusiones del capítulo.

Al concluir este capítulo, se obtiene una primera versión estable y funcional del sistema, basada en el diseño realizado y mediante la implementación de las historias de usuario y las tareas generadas por estas. En el siguiente capítulo se expondrán las pruebas realizadas al sistema para ayudar a identificar y corregir fallos en caso de que existan.

Capitulo 4: Pruebas

4.1 Introducción al capítulo.

XP enfatiza mucho los aspectos relacionados con las pruebas, clasificándolas en diferentes tipos y funcionalidades específicas, indicando quien, como y cuando estas deben ser implementadas y ejecutadas.

Del buen uso de las pruebas depende el éxito de otras prácticas, tales como la propiedad colectiva del código y la refactorización. XP plantea que se debe ser muy estricto con las pruebas. Solo se deberá liberar una nueva versión del producto si este ha pasado con el cien por ciento la totalidad de las pruebas.

Una de las bases de la metodología XP es el uso de pruebas para comprobar el correcto funcionamiento del código implementado, esto permite que se obtengan soluciones de una mayor calidad, así como aumentar la seguridad al evitar errores no deseados a la hora de refactorizar y modificar. (Beck, 2000)

4.2 Pruebas Unitarias.

La producción de código está guiada por las pruebas unitarias. Las pruebas unitarias se establecen antes de escribir el código y se ejecutadas constantemente ante cada modificación al sistema. (Beck, 2000)

Estas pruebas son aplicadas a todo el código no trivial que se halla implementado, condicionando la liberación de una clase solo cuando esta tenga asociada sus pruebas correspondientes. Las pruebas unitarias deben ser construidas por los programadores empleando algún mecanismo que permita automatizarlas para ahorrar tiempo.

El empleo de pruebas unitarias hace más fácil la liberación continua de versiones, ya que al implementar nuevas funcionalidades estas pruebas se llevarían a cabo de forma automática para saber que la nueva versión no contiene errores.

A la aplicación desarrollada, no se le aplicaron pruebas de este tipo, pero se creó este epígrafe para destacar la existencia y la posibilidad de realizar otros tipos de pruebas.

4.3 Pruebas de aceptación.

Las pruebas de aceptación, también llamadas pruebas funcionales son supervisadas por el cliente. Estas permiten confirmar que al final de cada iteración las historias de usuario han sido implementadas correctamente, cada historia de usuario deberá tener al menos una prueba de aceptación. Para que una historia de usuario se considere aprobada, deberá pasar todas las pruebas de aceptación elaboradas para dicha historia.

Las pruebas de aceptación son pruebas de caja negra, que representan un resultado esperado de una determinada transacción con el sistema. En caso de que existan fallos el cliente deberá determinar la prioridad de resolución de los mismos. El objetivo final de éstas es garantizar que los requerimientos han sido cumplidos y que el sistema es aceptable

A continuación se exponen las pruebas de aceptación realizadas a la aplicación.

Caso de prueba de aceptación	
Código: HU1_P1	**Nro. de HU:** 1
Nombre HU: Rotar la cámara horizontal y verticalmente.	
Descripción: Probar la funcionalidad de la HU número 1.	
Condiciones de ejecución: Aplicación ejecutándose en un navegador enriquecido con Adobe Flash Placer en su versión 10 o una superior.	
Entrada/Pasos de ejecución: Se presionan las teclas ←, ↑, →, ↓.	
Resultado Esperado: La cámara debe rotar en esas direcciones según la tecla presionada.	
Evaluación de la prueba: Satisfactoria.	

Tabla: Caso de prueba de aceptación, código HU1_P1, perteneciente a Historia de Usuario 1.

Caso de prueba de aceptación	
Código: HU2_P1	**Nro. de HU:** 2
Nombre HU: Acercar y alejar la cámara al escenario.	
Descripción: Probar la funcionalidad de la HU número 2.	
Condiciones de ejecución: Aplicación ejecutándose en un navegador enriquecido con Adobe Flash Placer en su versión 10 o una superior.	
Entrada/Pasos de ejecución: Se presionan las teclas PageUp y PageDown	
Resultado Esperado: La cámara debe acercarse y alejarse según la tecla presionada.	
Evaluación de la prueba: Satisfactoria.	

Tabla: Caso de prueba de aceptación, código HU2_P1, perteneciente a Historia de Usuario 2.

Caso de prueba de aceptación	
Código: HU3_P1	**Nro. de HU:** 3
Nombre HU: Calcular camino mínimo entre puntos tridimensionales del escenario.	
Descripción: Probar la funcionalidad de la HU número 3.	
Condiciones de ejecución: Aplicación ejecutándose en un navegador enriquecido con Adobe Flash Placer en su versión 10 o una superior. Deben estar seleccionados los dos elementos tridimensionales entre los cuales se va a calcular el camino mínimo.	
Entrada/Pasos de ejecución: Seleccionar dos elementos tridimensionales.	
Resultado Esperado: El cuadro de texto de la aplicación encargado de mostrar los resultados debe de mostrar el camino mínimo a seguir, así como la distancia total del camino recorrido.	

Evaluación de la prueba: Satisfactoria.

Tabla: Caso de prueba de aceptación, código HU3_P1, perteneciente a Historia de Usuario 3.

Caso de prueba de aceptación	
Código: HU4_P1	**Nro. de HU:** 4
Nombre HU: Calcular distancia entre dos puntos del escenario.	
Descripción: Probar la funcionalidad de la HU número 4.	
Condiciones de ejecución: Aplicación ejecutándose en un navegador enriquecido con Adobe Flash Placer en su versión 10 o una superior. Deben estar seleccionados los dos puntos del plano entre los cuales se va a calcular la distancia.	
Entrada/Pasos de ejecución: Seleccionar dos puntos del plano.	
Resultado Esperado: El cuadro de texto de la aplicación encargado de mostrar los resultados debe de mostrar la distancia entre los dos puntos.	
Evaluación de la prueba: Satisfactoria.	

Tabla: Caso de prueba de aceptación, código HU4_P1, perteneciente a Historia de Usuario 4.

4.4 Conclusiones del capítulo.

Al concluir este capítulo quedan expuestas las pruebas realizadas a la aplicación arrojando resultados satisfactorios en su totalidad. El resultado de estas pruebas brindará al cliente seguridad y confiabilidad sobre las funcionalidades implementadas al sistema.

CONCLUSIONES

Durante el desarrollo del presente trabajo se hizo un estudio de las tendencias actuales para la visualización de elementos tridimensionales en la Web, lo cual nos permitió seleccionar las herramientas para el desarrollo de una aplicación de este tipo. La correcta selección de la metodología, IDE de desarrollo, lenguaje de programación, biblioteca de código, nos posibilitó la correcta visualización de la aplicación en navegadores Web y la fácil interacción del usuario. Aunque a la fecha de finalización de esta tesis se estaban comenzando a dar pasos que permitirán la visualización de elementos 3D en la Web sin la necesidad de instalar ningún software adicional como es el caso de WebGL que es una especificación estándar que está siendo desarrollada actualmente para cumplir esta función y cuya primera versión estable dio a luz el 11 de febrero de 2011 según el sitio de sus desarrolladores (www.khronos.org), también es el caso de HTML5 que es la quinta versión de este importante lenguaje que no es aún un lenguaje recomendado, pero los navegadores más usados como es el caso de Firefox, Internet Explorer, Chrome de Google y Safari han comenzado a interpretar funcionalidades de este lenguaje.

Al llegar a este punto podemos destacar que se cumplieron los objetivos propuestos en la investigación, arrojando a su vez una serie de resultados, plasmados a continuación:

- Se realizó un estudio de forma satisfactoria de algunos sistemas informáticos que permiten la visualización de elementos tridimensionales en la Web, centrándonos en los paseos virtuales.
- Se realizó un estudio de las principales herramientas y tecnologías más usadas a nivel mundial para el desarrollo de este tipo de aplicaciones.
- Se realizó un estudio de las metodologías de desarrollo, seleccionando XP para llevarla a cabo durante el ciclo de elaboración de la aplicación.
- Se implementó un escenario tridimensional interactivo Web utilizando las herramientas seleccionadas que nos van a permitir realizar aplicaciones de este tipo
- Se obtuvo un sistema que permite calcular distancias y caminos mínimos dentro de un escenario tridimensional interactivo Web.

El Escenario Virtual Tridimensional Interactivo Web brinda al usuario la posibilidad de visualizar la geografía de la Universidad de las Ciencias Informáticas en un navegador Web, posibilitando la visualización de algunos elementos tridimensionales que conforman esta geografía. También brinda la

posibilidad de calcular distancias dentro de este escenario y calcular caminos mínimos entre los objetos tridimensionales mostrados. Sentando las bases para la posible creación en un futuro de un sistema similar que abarque la universidad en su totalidad.

De esta manera se concluye que se cumplió el objetivo de la investigación al desarrollar la presente solución informática cumpliendo todas las metas trazadas al comienzo de la investigación.

RECOMENDACIONES

Una vez terminada la aplicación se puede constatar que se cumplieron los objetivos trazados al comienzo de la investigación, este proceso arrojo como resultado una primera versión de un sistema que podría brindar una gran información visual de la geografía de la universidad a través de la Web, por cuanto se hacen las siguientes recomendaciones:

- Se recomienda para el mejoramiento de la aplicación incluir otros elementos tridimensionales al mapa, de otros puntos de referencia de la universidad como podría ser el caso de instalaciones como el rectorado, la piscina, las instalaciones del área de infraestructura productiva de la universidad.
- Se recomienda que la creación de las texturas de los modelos tridimensionales usados sea realizada por un grupo de diseño de nuestra universidad para darle el aspecto más real posible.
- Se recomienda para versiones posteriores de la aplicación darle dimensiones al mapa que se correspondan con las reales, para que los cálculos generados sean iguales a los de la realidad.

REFERENCIAS BIBLIOGRÁFICAS

1. *Rational Rose Enterprise*. (2007). Obtenido de Rational Rose Enterprise.
2. Alexander, C. (1979). *The Timeless Way of Building.*
3. Beck, K. (2000). *Extreme Programming Explained.*
4. Headquarters, C. (s.f.). *Visual Paradigm*. Recuperado el 2011, de http://www.visual-paradigm.com/product/vpuml
5. Luis Miguel Echeverry, L. E. (2007). *Caso práctico de la Metodología Agil XP al desarrollo de software.*
6. O'Reilly, M. (2010). *Learning Phyton.Fourth Edition.*
7. Parcher, L. R. (1997). Glosario de Termino Informaticos. *FAC vol 26*.
8. Penades, P. L. (2008). *Metodologías Agiles para el desarrollo de Software: eXtreme Programming.*
9. Pereda, J. M. (24 de Agosto de 2007). Recuperado el 2011, de http://jmpereda.wordpress.com
10. Pressman, R. S. (2009). Software Engineering. En R. S. Pressman, *Software Engineering 7ma Edicion.*
11. Reinoso, B. (2007). *Introduccion a la Arquitectura de Software.*
12. Sanchez, M. M. (2009). *Metodologias de desarrollo de software.*
13. Solis, M. C. (2003). *Una explicación de la programación extrema.*
14. Sommerville, I. (2005). *Ingenieria de Software 8va Edicion.*
15. Sommerville, I. (2005). Ingenieria del Software. En I. Sommerville.
16. Soriano, G. J. (2011). *Desarrollo de Aplicaciones III:Aplicaciones RIA.*
17. *Visual Paradigm*. (s.f.). Recuperado el 2011, de http://www.visual-paradigm.com/VPGallery/diagrams/Class.html

BIBLIOGRAFÍA CONSULTADA

1. *Rational Rose Enterprise*. (2007). Obtenido de Rational Rose Enterprise.
2. Alexander, C. (1979). *The Timeless Way of Building.*
3. Beck, K. (2000). *Extreme Programming Explained.*
4. Moock, Colin.(2007). Essential ActionScript 3.0.
5. Luis Miguel Echeverry, L. E. (2007). *Caso práctico de la Metodología Agil XP al desarrollo de software.*
6. O'Reilly, M. (2010). *Learning Phyton.Fourth Edition.*
7. Parcher, L. R. (1997). Glosario de Termino Informaticos. *FAC vol 26*.
8. Penades, P. L. (2008). *Metodologías Agiles para el desarrollo de Software: eXtreme Programming.*
9. Elst, Peter.Jacobs,Sas. (2007). Object Oriented Action Script 3.0
10. Pressman, R. S. (2009). Software Engineering. En R. S. Pressman, *Software Engineering 7ma Edicion.*
11. Reinoso, B. (2007). *Introduccion a la Arquitectura de Software.*
12. Sanchez, M. M. (2009). *Metodologias de desarrollo de software.*
13. Solis, M. C. (2003). *Una explicación de la programación extrema.*
14. Sommerville, I. (2005). *Ingenieria de Software 8va Edicion.*
15. Peters, Keith.(2009). AdvancED ActionScript 3.0 Animation.
16. Soriano, G. J. (2011). *Desarrollo de Aplicaciones III:Aplicaciones RIA.*
17. Sanders,B William.(2007) ActionScript 3.0 Design Patterns.

ANEXOS

Anexo 1

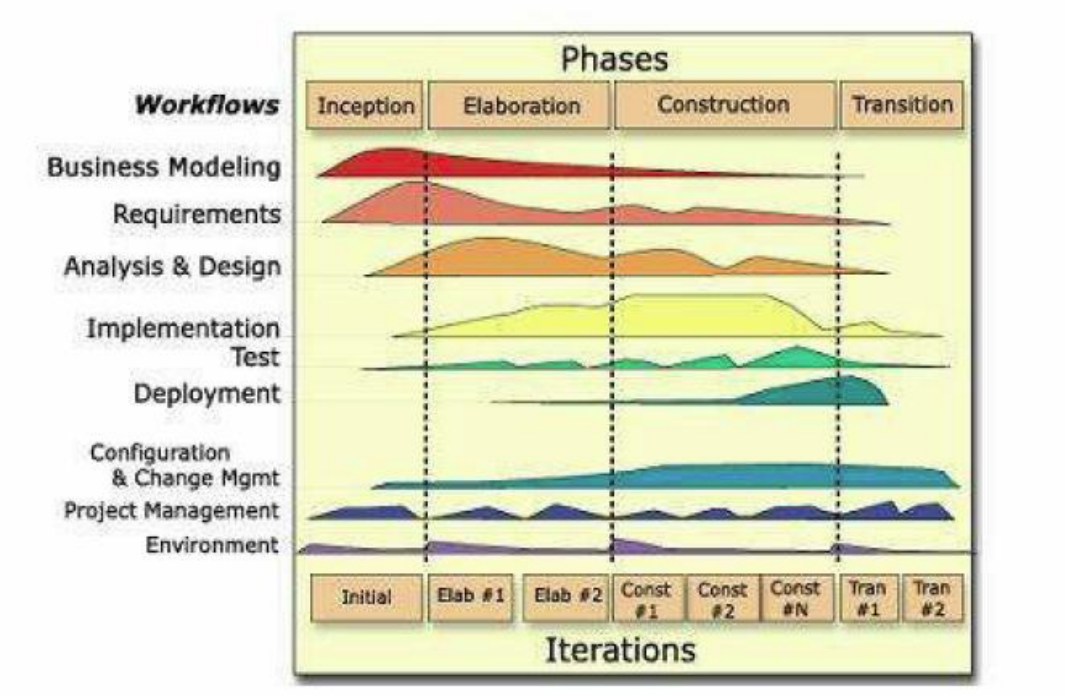

Figura: Fases e Iteraciones de RUP.

Anexo 2

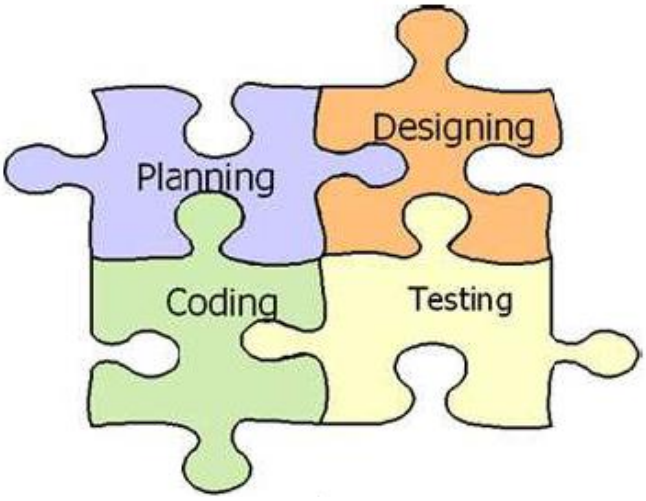

Figura: Representación gráfica de la Metodología XP.

Anexo 3

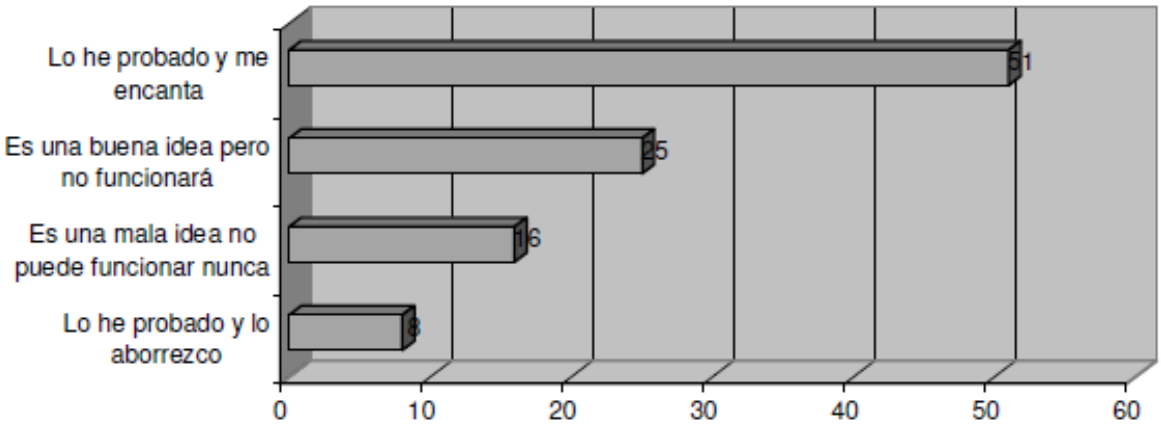

Figura: Entrevistas a desarrolladores sobre XP. (Solis, 2003)

Anexo 4

Puntos a comparar	RUP	XP
Tamaño de los equipos	Pensado para proyectos y equipos grandes, con roles designados y con una duración extendida.	Proyectos cortos con equipos pequeños y rotativo en cuanto a roles.
Obtención de requerimientos	Basado en casos de uso donde se describen los requerimientos de la aplicación desde el punto de vista del usuario. Los casos de uso son descritos en lenguaje técnico, complejo a veces para el entendimiento del cliente.	Basado en historias de usuario donde se describen los requerimientos de la aplicación desde el punto de vista del usuario, las historias de usuario definen los detalles técnicos sin adentrarse en los detalles de la implementación, son descritas con un lenguaje natural de manera que el cliente pueda entender perfectamente lo que se ha escrito.
Carga de Trabajo	Se hace un proceso pesado por estar demasiado basado en la documentación.	Proceso ligero.
Relación con el cliente	Se le presenta al cliente los artefactos generados al final de cada fase, para que sean evaluados por este y se pueden generar las iteraciones necesarias para la siguiente fase.	Comunicación fluida con el cliente a través de su representante. Al final de cada iteración el cliente recibe una porción funcional del software para mantenerse informado e intervenir rápidamente si el desarrollo no cumple

		sus necesidades.
Desarrollo	Basado en iteraciones. Los programadores tienen mayor carga de trabajo.	Basado en iteraciones. Los programadores tienen menor carga de trabajo.
Puntos Flacos	Ya que es un proceso general y muy grande se hace complicado para proyectos y equipos pequeños, pues deben repartirse 32 roles y generar muchos artefactos, lo cual significa un incremento de tiempos y costos.	El cliente debe designar una persona para estar totalmente involucrada en el proceso de desarrollo, lo cual puede implicar que esta persona deje de hacer sus funciones, por lo cual se considera la utilización de esta metodología para desarrollos internos. Es poco probable que el cliente pueda prescindir a tiempo completo de uno de sus empleados lo cual incurriría en un coste adicional para el cliente.

Tabla: Tabla comparativa entre RUP y XP.

Anexo 5

Imagen: Vista de la aplicación.

Anexo 6

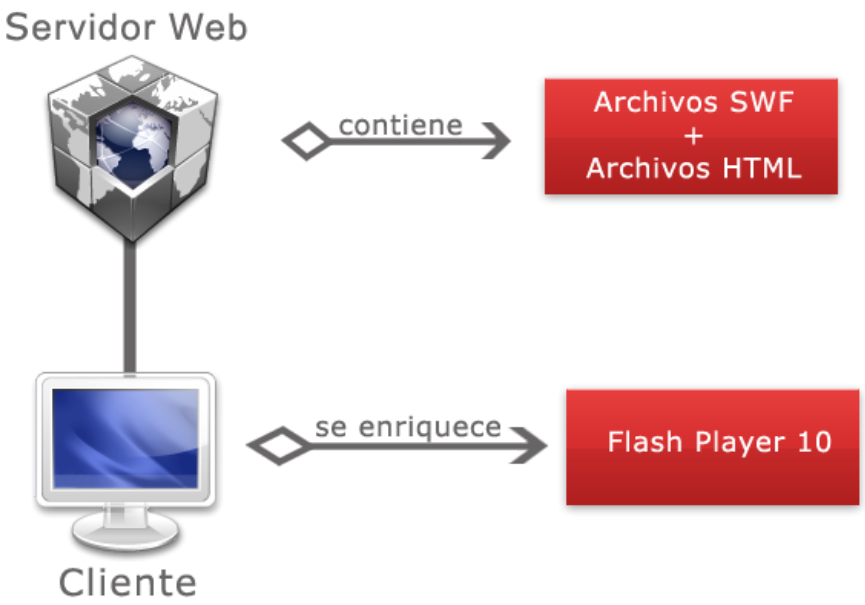

Figura: Representación gráfica de la arquitectura utilizada en la aplicación.

GLORSARIO DE TÉRMINOS Y SIGLAS

ActionScript: Es un lenguaje de programación orientado a objetos (OOP), utilizado en especial en aplicaciones Web animadas realizadas en el entorno Adobe Flash.

Apache: Es un servidor Web HTTP de código abierto.

ASP: Es un framework para aplicaciones Web desarrollado y comercializado por Microsoft.

Away3D: Es un motor gráfico 3D de código abierto, escrito para la plataforma Adobe Flash en Action Script 3.0.

Bibliotecas: En ciencias de la computación, una biblioteca (del inglés *library*) es un conjunto de subprogramas utilizados para desarrollar software.

Editor WYSIWYG: Es el acrónimo de What You See Is What You Get (en inglés, "lo que ves es lo que obtienes"). Se aplica a los procesadores de texto y otros editores de texto con formato (como los editores de HTML).

Flash Player: Adobe *Flash Player* es una aplicación en forma de reproductor multimedia creado inicialmente por Macromedia y actualmente distribuido por Adobe Systems.

Flex 3: Es un término que agrupa una serie de tecnologías publicadas Macromedia para dar soporte al despliegue y desarrollo de Aplicaciones Enriquecidas de Internet.

Framework: La palabra inglesa "framework" define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular.

Herramientas CASE: Son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software.

HTTP: HyperText Transfer Protocol (Protocolo de transferencia de hipertexto) es el método más común de intercambio de información en internet.

JSP: Tecnología Java para la creación de páginas web con programación en el servidor.

Linux : Es uno de los términos empleados para referirse a la combinación del núcleo o kernel libre similar a Unix denominado *Linux.*

MacOS: del inglés Macintosh Operating System, en español Sistema Operativo de Macintosh) es el nombre del sistema operativo creado por Apple.

MXML: Es un lenguaje descriptivo desarrollado para la plataforma FLEX de Adobe.

Navegador: Un *navegador* o *navegador* web (del inglés, web browser) es un programa que permite ver la información que contiene una página web.

PaperVision3D: Es un motor gráfico 3D de código abierto.

PHP: Es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas.

Servidor Web: Un *servidor web* es un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente.

UML: Es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software.

Web: Se refiere a la World Wide Web (también conocida como «la Web»), el sistema de documentos (o páginas web) interconectados por enlaces de hipertexto, disponibles en Internet.

Webmail: Un webmail es un cliente de correo electrónico, que provee una interfaz web por la que acceder al correo electrónico.

XML: Son las siglas de Extensible Markup Language. Es un sistema estándar de codificación de información.