Editor de Mapas Isométricos en línea

IsoMap 3D

Isometric 3D Framework

Autores:

Heriberto Angel Martín Ramos
Jorge Lisandro Ruiz Valenzuela

Tutores:

Msc.Lidiexy Alonso Hernández
Ing.Yausell Ruiz Marine

Universidad de las Ciencias Informáticas

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos 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 _____.

_____ _____

Firma del Autor: Heriberto A Martin Ramos. Firma del Autor: Jorge Lisandro Ruiz Valenzuela.

_____ _____

Firma del Tutor: Msc. Lidiexy Alonso Hernández Firma del Tutor: Ing. Yausell Ruiz Marine

Dedicatoria

A mis padres que con tanto amor y sacrificio hicieron posible que pudiera alcanzar el éxito en mis estudios.

Jorge Lisandro

A mis padres, a mi abuela Lila y a mi abuelo Ángel, que ya no está con nosotros pero le recuerdo siempre.

Heriberto Angel

Agradecimientos

Mi agradecimiento más profundo a mis padres que siempre me han apoyado a lo largo de la vida, gracias por su dedicación.

A mi hermana por ser la persona más importante como amiga, que siempre me apoyo en las decisiones que he tomado y por ser mi guía en la vida. Gracias por tu amor y dedicación.

A mi novia Daylen por el amor, dedicación y apoyo que me brinda cada día.

A mis amigos Reyneris, Frank, Mayrelis, Lien, Annier, Reinier, Aliannys, Yenia, Pablo y todos mis compañeros de cuarto que me han apoyado a lo largo de mi carrera.

A mis tutores por la ayuda que me brindaron en la elaboración de este trabajo.

A mi compañero de tesis, que sin él no hubiera sido posible la culminación de este trabajo. Gracias por tu amistad.

A quienes me dieron su amistad, su corazón, dedicación y apoyo a lo largo de mi carrera.

A la UCI por haberme acogido en estos cinco años y formarme como Ingeniero.

Jorge Lisandro

Agradezco a mis padres y abuelos por haber sido ejemplos a seguir en mi vida y por haberme apoyado siempre para lograr alcanzar mis metas.

A mi hermanito.

A mi tía Nélida y mi primito Tato.

A mi papá Jesús por creer siempre en mí.

A mi familia de La Habana, mis tíos y primos que fueron pilares fundamentales en mi infancia y que han luchado conmigo en muchas etapas y en estos 5 años, sin ellos nada hubiera sido igual.

A mi primo Alexey, que tantas carreras ha dado conmigo, gracias por tu amistad y tu ayuda incondicional.

A mi novia Mónica por su ayuda y apoyo en todo momento, su cariño y su dedicación sin límites.

A su familia que me ha acogido como a un hijo.

A mis amigos de Sta Clara, Rafa, Abraham, Suleidy, Carlos, Rosely, Edel, que han sido una familia más.

A los amigos que conocí en la UCI y a los que vienen conmigo desde la vocacional, Norbert, Rafi, Osnel, William, Sergio, Isuel, Yuri, Abelito, Jorge Luis, Daymel, David, Ibrael, Reyneris, Frank, PJ, los dos Reinier, Donic, Yuniet, Carlos Felipe, Idelkys, Lorenzo.

A mis compañeros de la Facultad 1 y los de la Facultad 5.

A mi compañero de tesis por luchar junto conmigo hasta el final.

A mis tutores por su guía, consejo y dedicación a este trabajo.

Al tribunal por sus aportes críticos y objetivos.

A Jorge y al equipo de Primavera.

A todos los que se me han quedado sin mencionar y han sido parte de mi día a día en estos 5 años y en este trabajo.

A todos ellos, muchas gracias

Heriberto Angel

"El orgullo de los mediocres consiste en hablar siempre de sí mismos; el orgullo de los grandes hombres es no hablar nunca de ellos."

Voltaire

Resumen

En la actualidad, el desarrollo de aplicaciones web es una tarea cada vez más compleja debido a la variedad de tecnologías existentes para la concreción de soluciones, tomando como punto de partida la Web 2.0 que es la representación de la evolución de las soluciones tradicionales hacia soluciones web orientadas al usuario final. Siguiendo estos paradigmas se abre un nuevo camino para la creación de entornos tridimensionales de bajo consumo de hardware que mejoren el diseño y la versatilidad del software en línea. La Universidad de las Ciencias Informáticas (UCI) se encuentra enfrascada en aprovechar estas posibilidades para desarrollar juegos sobre la plataforma web, es por ello que la presente investigación se encuentra centrada en la elaboración de un marco de trabajo que permita la creación y visualización de mapas isométricos tridimensionales, proveyéndolos de interactividad para su utilización en el desarrollo de juegos en línea.

Palabras clave: Web 2.0, juegos en línea, editor de mapas, Away3D, mapas isométricos, modelos 3D.

Índice

IsoMap 3D

Isometric 3D Framework

Introducción

Introducción

Siguiendo los estilos actuales de diseño e implementación de aplicaciones para internet mediante el uso de un conjunto de tecnologías como AJAX (del inglés "Asynchronous JavaScript and XML") y RIAs (del inglés "Rich Internet Applications"), aparecen novedosas alternativas para el desarrollo software en línea, surgen los conceptos de aplicación en la nube, Web 3.0 y de todo lo que puede lograrse integrando estas nuevas tendencias. [HOYTECNOLOGÍA 2010]

Los clásicos estándares del desarrollo de aplicaciones se encuentran hoy en un proceso de migración orientado a servicios en la red, con una total independencia de los sistemas de escritorio. Esta es una de las razones de que las grandes compañías desarrolladoras estén centradas en nuevos entornos dirigidos a la creación de aplicaciones para internet cada vez más potentes.

Uno de los principales obstáculos de esta nueva rama da al traste con el objetivo de encontrar una tecnología factible para manejar y potenciar gráficos en tercera dimensión (3D) dentro de los entornos web. Consecuentemente surgen alternativas como Gráficos 3D Extensibles (del inglés "Extensible 3D Graphics", en adelante X3D), Google O3D, y una mediana gama de motores para el manejo de gráficos 3D que utilizan la plataforma Flash potenciada por Adobe Flash Player[1] como contenido embebido en los navegadores.

La Universidad de las Ciencias Informáticas (UCI), se encuentra enfrascada en la búsqueda de alternativas para lograr resultados palpables en esta rama. No obstante ha venido presentando una serie de inconvenientes en cuanto a desarrollo de aplicaciones en línea, tales como:

- Engorroso desarrollo de entornos tridimensionales y su vinculación a los navegadores.

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

- El tiempo de implementación de los entornos es amplio, dando a lugar que los proyectos no se entreguen a tiempo, trayendo como consecuencia un atraso de manera general en el proyecto.

- La integración de los videojuegos a la plataforma web, especialmente la creación de mapas y entornos, es compleja.

Partiendo de este análisis se identifica la necesidad de solucionar en el menor plazo posible las deficiencias ya expuestas, que entorpecen el desarrollo de proyectos con requisitos similares en nuestra universidad. En lo adelante todos los esfuerzos de los investigadores estarán encaminados a resolver el **problema** que queda formulado con la siguiente interrogante:

¿Cómo facilitar la creación de mapas isométricos que sirvan de base para una plataforma de desarrollo en línea?

Según las condiciones dadas y como elemento imprescindible para planificar la investigación, el **objeto de estudio** radica en las plataformas de desarrollo de juegos. Por lo que se determina como **campo de acción** los módulos de creación y edición de mapas isométricos para juegos en línea.

Como **objetivo general** de la presente investigación se propone implementar una aplicación web que permita crear y editar mapas isométricos para el desarrollo de juegos en línea.

Para dar solución al problema planteado en la investigación se definen las siguientes **tareas de la investigación:**

- Determinación de las necesidades de la Facultad 5, para el desarrollo de juegos en línea.

- Descripción de las tecnologías, tendencias actuales y otros elementos relacionados con los entornos 3D sobre plataformas web, para la selección de las alternativas más adecuadas.

- Identificación de la metodología de desarrollo de software, lenguajes de programación y sistema de gestión de datos, para una adecuada organización del trabajo en equipo y la optimización de los procesos de implementación del software.

- Selección de una plataforma de desarrollo de entornos 3D en la web que permita dar solución al problema planteado.

- Desarrollo de las funcionalidades del editor de mapas isométricos.

Durante el desarrollo de esta investigación fue necesario profundizar en el estudio de los temas abordados, por lo que se utilizaron varios métodos teóricos:

Analítico-Sintético: para la búsqueda y análisis de los conceptos y documentos necesarios que permitieron la obtención de los elementos para la comprensión del objeto de estudio.

Análisis-Histórico Lógico: permitió que se analizara el desarrollo histórico del objeto de estudio y encontrar todas las publicaciones posibles editadas en Cuba y en el mundo sobre la creación de mapas embebidos en los navegadores.

Como método empírico:

Observación: Para observar y verificar los problemas que existen en la universidad por falta de una aplicación que permita crear mapas en un entorno web.

IsoMap 3D

Isometric 3D Framework

Capítulo 1

Fundamentación teórica

Capítulo 1: Fundamentación teórica

El presente capítulo describe las bases científicas sobre las que se desarrollará el aporte práctico de la investigación: una aplicación que facilite la creación de mapas isométricos que constituya la base para una plataforma de desarrollo en línea.

Se introducirá el tema de las tendencias tecnológicas más avanzadas y homólogas al objeto de estudio. También se explicarán detalladamente aspectos a tener en cuenta en la selección de las herramientas, tecnologías y metodologías a utilizar en la búsqueda de mejoras en la creación de mapas isométricos en línea.

Además se expone el estado del arte de las soluciones existentes, así como su viabilidad y efectividad para los proyectos en los que fueron implementadas.

1.1. **Conceptos generales relacionados**

Componentes visuales: Todo objeto gráfico que esté orientado a almacenar respuestas a acciones de entrada sobre la interfaz de usuario. Igualmente puede ser de interacción o solamente de lectura de información. Ejemplos comunes son: los botones, los cuadros de textos, formularios de inscripción, botones de ejercicios, dibujos y otros.

Mapa isométrico: Mapa o dibujo tridimensional que se ha realizado con los ejes inclinados formando un ángulo de 30° con la horizontal. Se caracterizan por emplear una proyección ortogonal, es decir que los elementos más lejanos –a diferencia de la proyección en perspectiva– poseen el mismo tamaño que los elementos más cercanos. [CARLBORN and PACIOREK 1978]

En Línea: Es una categoría que se le da a las aplicaciones o procesos que se ejecutan a partir del uso de una computadora conectada a la red de redes.

Motor Gráfico 3D: conjunto de clases o bibliotecas que proporcionan funciones de renderizado 2D y 3D. Se suele apoyar en APIs[2] gráficas como OpenGL o DirectX y

[2] **API**: Interfaz de Programación de Aplicaciones (del inglés “Application Programming Interface).

definen una capa de abstracción entre el hardware, el sistema operativo y el videojuego. El trabajo de un motor gráfico es el de realizar todas las tareas de bajo nivel tales como comunicarse con el adaptador gráfico, administrar la escena, transformar la geometría del mundo 3D y lidiar con diversos procesos matemáticos que afectan su comportamiento. [OGRE 2009]

Todos los motores gráficos incluyen editores, incluso los más recientes tienen editores en tiempo real. Estos permiten posicionar objetos, crear terrenos, vegetación, incluso sonido y luces y ver en tiempo real cómo se modifica el juego o aplicación. Existen motores 3D libres para uso no comercial, motores gratis para uso comercial y motores 3D comerciales. Estos últimos son muy rápidos para desarrollar aplicaciones pero tienen la limitación del pago de licencias para su uso.

SDK: Herramienta Estándar de Desarrollo ("Estándar Development Kit" por sus siglas en inglés) conjunto de herramientas, generalmente bibliotecas y compiladores de código que permiten a un equipo de desarrollo construir de una manera rápida, softwares para una plataforma específica. No incorporan un IDE[3], son una interfaz de programación a la cual se le pueden crear entornos de desarrollo que la aprovechen para agilizar aún más la producción. [SDK 2010]

Plataforma Flash: Nombre oficial que recibe el conjunto de herramientas, aplicaciones y bibliotecas de componentes de Adobe y se encuentra centrada en Adobe Flash Player como reproductor final de los archivos SWF. De las herramientas se encuentran en un primer plano el IDE de Adobe Flash y Adobe Flex Builder. El lenguaje programación que utiliza es ActionScript en la versión 3.0 (AS3). [INCORPORATED 2009a]

Software Privativo: Se refiere a todo aquel software o fragmento de software cuya licencia comercial, de uso o publicación, no permite compartir el código fuente del mismo, o sea, restringe las libertades para su uso.

[3]**IDE**: Entorno Integrado de Desarrollo (del inglés "Integrated Development Environment").

Software Libre: Hace referencia a la libertad de los usuarios o desarrolladores de tener acceso libre al código fuente para copiarlo, leerlo, distribuirlo, cambiarlo y mejorarlo a su gusto.

1.2. **Análisis de las soluciones existentes**

Uno de los aspectos de marcada importancia para el desarrollo de mapas tridimensionales es la creación de componentes visuales. Muchas de las herramientas utilizadas en la actualidad necesitan una licencia del proveedor para su utilización, por lo que las hace poco rentables debido a sus elevados precios.

Las alternativas libres y gratuitas no escasean, pero es engorroso encontrar las más factibles. Esta es la razón por la que ha sido necesario acudir a soluciones homólogas de código abierto:

X3D: Es un estándar para gráficos vectoriales definido por una norma ISO, que puede emplear tanto una sintaxis similar a la de XML como una del tipo de Lenguaje de Modelado de Realidad Virtual (del inglés: "*Virtual Reality Modelling Language*", en adelante VRML). X3D amplía VRML con extensiones de diseño y la posibilidad de emplear XML para modelar escenas completas en tiempo real. También es utilizado frecuentemente para el procesamiento de imágenes y la representación del medio ambiente de manera 3D. Soporta varias codificaciones de formato de archivos y lenguajes de programación, ofreciendo una interoperabilidad sin igual para los datos 3D y una significativa flexibilidad en la manipulación, la comunicación y las escenas que muestra de forma interactiva.[LEONARD and BRUTZMAN 2008]

WebGL: WebGL es una serie de estándares para manejar gráficos 3D, por medio de aceleración de hardware, en los navegadores web sin necesidad de plugins. El sistema funciona a través de JavaScript y su capacidad para comunicarse con OpenGL ES 2.0 es muy elevada. [GUTIÉRREZ 2010]

Es un estándar que es apoyado por Mozilla (Firefox), Google y Opera. Centra su funcionamiento a través de WebKit, que no es más que el motor que se encuentra detrás de los navegadores Safari y Chrome. Su potencial está encaminado para la

implementación de juegos 3D en los navegadores, además de desarrollar aplicaciones educativas haciendo uso de los estándares para gráficos 3D de OpenGL. Su principal desventaja radica en que se encuentra actualmente en desarrollo, por lo que su utilización es muy limitada.

OpenSpace: Es un marco de trabajo potente para el rápido desarrollo de mundos virtuales isométricos multijugador para la plataforma flash. Su motor gráfico aumenta el nivel de las funcionalidades del lenguaje AS3 y SmartFoxServer[4] [SMARTFOXSERVER 2010], ofreciendo un nivel impredecible de funcionalidades y optimización para que el mundo virtual parezca único al espectador. Permite crear avanzadas arquitecturas de mapas, incluyendo puentes y pasos por encima, con capas de fondo separadas para un mayor control sobre la apariencia gráfica del mapa; además de maximizar el renderizado en la creación de largos mapas. Sólo puede mostrar un máximo de 20 jugadores en un mismo mapa y puede exportar mapas con una dimensión máxima de 15 x 15 tiles, para mapas muy extensos el tamaño es reducido a una dimensión inferior a la máxima permitida. [OPENSPACE 2010]. El marco de trabajo tiene el inconveniente de ser propietario y se necesita de licencia para su utilización.

Sandy3D: Es un motor disponible en tres versiones: AS2, AS3 y haXe. Es popularmente conocido por su gran empleo en la creación de aplicaciones web 3D debido a que tiene un renderizado excelente para escenas interiores y contiene avanzados efectos de iluminación dentro de los que se encuentra Phong, Gouraud y CelShading. Es compatible con Adobe Flash Player desde su versión 7 hasta la 10, además de tener compiladores que manejan varios tipos de formato 3D como son: Collada, 3DS, ASE, MD2. Su principal desventaja es que para manejar transparencia de caras y el agregado de texturas de mapa de bits (bitmap) es necesario utilizar un volumen considerable de código, además de ofrecer una baja calidad de la escena en cuanto a texturizado se trata [SANDY3D 2010]. Aunque

[4]**SmartFoxServer**: es una plataforma para el rápido desarrollo masivo de aplicaciones multiusuario y juegos con Adobe Flash, Flex, Air, Java, Android, .Net, Unity3D, Silverlight y otros.

este motor pertenece a una comunidad libre, en año 2008, tras un cambio de la perspectiva de desarrollo del mismo, fueron agregados módulos con restricciones monetarias y de licencia para su uso.

Alternativa3D: Es un motor 3D basado en la plataforma Adobe Flash que posibilita mostrar mundos tridimensionales, juegos, visitas virtuales o simplemente objetos en el navegador. Una de las mayores ventajas es que utiliza Flash, que se encuentran en la mayoría de los navegadores. Hace uso de la corrección perspectiva mediante la triangulación dinámica, detecta colisiones con el uso de la simulación de la física y es capaz de reutilizar capas cuando se cambia o se modifican regiones mediante un nuevo trazado debido a que son independientes de los polígonos. Provee una amplia iluminación dinámica y sombras (puntuales y direccionales). Además contiene un automapeado UV (horizontal y vertical) que posibilita calcular las coordenadas de los objetos en el espacio y así modificarlos en cualquier momento, ya sea modificar las propiedades de los objetos o aplicarles otras texturas. Su principal desventaja radica en que es un software propietario y se necesita de licencia para su utilización. [ALTERNATIVA3D 2009]

1.3. **Tendencias Tecnológicas**

Los IDEs “lo que ves es lo que obtienes” (del Inglés: “*What do you see it what do you get*”, en adelante WYSIWYG), son históricamente los de mayor impacto y resultado han demostrado tener en el desarrollo de elementos visuales.

WYSIWYG se logra a partir de la emulación gráfica, en tiempo de desarrollo, del código fuente antes de ser compilado o interpretado. De esta forma es posible crear y editar componentes en un entorno visual con facilidad y con las características gráficas deseadas desde un inicio, sin tener que recurrir a la supuesta satisfacción, que por lo general resulta de hacer el mismo trabajo durante horas solamente utilizando código. La mayoría de las herramientas de este tipo son capaces de ser utilizadas por usuarios comunes sin el más mínimo conocimiento de programación.

En el análisis realizado se observa que algunas de las soluciones existentes no ofrecen una óptima calidad de desarrollo de elementos visuales, además de ser

privativas; y otras simplemente no tienen vinculación con AS3 en tiempo de desarrollo.

A diferencia de esto, Adobe Flash, herramienta líder entre los entornos de desarrollo para Flash Player y la principal guía a seguir por los proyectos de software libre, ofrece una interfaz muy amigable en tiempo de desarrollo en el interior del ambiente de trabajo para lograr WYSIWYG. Cabe agregar que los componentes visuales son editados de manera gráfica y el código que los representa, paradójicamente, se mantiene oculto a los desarrolladores. A esto se suma que, si se utiliza código solamente, con un objetivo similar, las bibliotecas de clases de AS3 ofrecen una interfaz de dibujo que no permite controlar, luego de creados, los detalles gráficos para la edición de los mismos, lo que obliga a eliminarlos visualmente y volverlos a trazar con los datos nuevos.

1.4. **Tecnologías**

Según lo analizado hasta el momento, el principal factor que propicia el establecimiento de desarrollo de las soluciones existentes de software libre, es el momento de la integración de los entornos de desarrollo visual con los compiladores de de código ActionScript. La incompatibilidad de muchos de los IDEs WYSIWYG con el lenguaje de programación para la plataforma Flash solo se limita a la creación de archivos de tipo SWF (Shockwave Flash) carentes de interactividad y calidad de los elementos visuales, aspectos muy importantes para la creación de juegos en línea.

Con las alternativas descritas anteriormente, aunque algunas utilizan Adobe Flash Player, los equipos de desarrollo de las mismas deciden utilizar entornos de desarrollo en línea y así lograr una gestión controlada de la información persistente. Adobe Flash Player está creado para acceder a la red bajo ciertas restricciones y de manera limitada al sistema de archivos local. Razón por la cual resulta contradictorio proponerse utilizarlo para crear una alternativa libre. Aun así, se crean plataformas de desarrollo que aprovechan los cambios legales que ha sufrido y las

novedosas soluciones que han ido apareciendo, las cuales se describen a continuación:

En Febrero del 2008, Adobe Systems Inc. (Adobe) liberó, bajo la licencia Mozilla Public License (en adelante MPL), el SDK de Flex. Se ha especulado que la decisión fue tomada debido al amplio avance del uso de AJAX, práctica estándar, muy utilizada para la creación de RIAs con software libre y que presenta una gran competencia a la plataforma Flash de Adobe en la red. El SDK libre de Flex está compuesto por el compilador de AS3, el depurador y la biblioteca de componentes visuales. [INCORPORATED 2009b]

Conjuntamente con esto se hace pública la primera versión estable de PaperVision3D que posteriormente fuera base para la evolución de otros motores gráficos como Away3D, herramienta capaz de recrear entornos 3D mediante el uso del SDK libre de Flex, posibilitando así el empleo de los navegadores para su visualización. [INCORPORATED 2009b]

De esta manera, la integración de Adobe Flex SDK y Away3D se presenta como la mejor alternativa para resolver el problema en cuestión. El uso combinado de los mismos está respaldado con una documentación organizada que lleva Adobe para el desarrollo de su plataforma. Las distintas publicaciones en línea sobre las capacidades, flexibilidad y escalabilidad de los proyectos basados en Flex y Away3D son enormes. A continuación se muestran características que se tuvieron en cuenta para tomar la decisión.

1.4.1. **Away3D**

Away3D es un motor gráfico libre para el desarrollo de aplicaciones embebidas en los navegadores. Posee un alto acoplamiento con lenguajes como AS que le permiten visualizar elementos creados en este tipo de código, ya sea de tipo tridimensional, bidimensional o simplemente un componente visual. [AWAY3D 2010]. Entre sus características de mayor relevancia se encuentran:

- *Visualización de objetos 3D*: contiene una amplia gama de funcionalidades que lo hacen ser muy potente en cuanto a la recreación de objetos tridimensionales en el espacio, además de soportar un elevado número de polígonos en una misma escena. Es capaz de cargar archivos con el formato 3ds, Collada, AS, md2 y muchos otros.

- *Robustez del núcleo gráfico de Away3D para el manejo de objetos 3D*: Away3D es el resultado de la evolución de motores gráficos como PaperVision, por lo que su núcleo provee altas capacidades para manejar objetos 3D que van desde primitivas básicas y una posición de cámara básica hasta objetos muy complejos como personajes, autos y rostros de personajes en movimiento con una elevada cantidad de polígonos.

- *Respaldo hacia los proyectos desarrollados con el motor a nivel mundial*: existen una amplia variedad de marcos de trabajo y lenguajes que son utilizados por las comunidades de desarrollo para extender las capacidades y potencialidades del motor a lo largo del mundo, principalmente promovidas por Adobe con el uso de su SDK libre de Flex.

1.4.2. **Adobe Flash Builder**

Adobe Flash Builder es conocido como el IDE por excelencia para crear contenidos interactivos para la WEB y el escritorio. Sus creadores han logrado construir una plataforma capaz de enfrentar una amplia variedad de proyectos. Entre las principales características que se tuvieron en cuenta para seleccionarlo se encuentran:

- *Fidelidad de Flash Builder como emulador WYSIWYG*: Muy pocas aplicaciones son capaces de mostrar mejores pre visualizaciones de sus resultados que ella misma. La interfaz de programación de Flash Builder es capaz de crear gráficos tan complejos como los que se producen desde el propio IDE de Adobe Flash. [BUILDER 2010].

- *Potencia del lenguaje de programación*: AS3 ha sido el resultado de un análisis muy profundo de sus versiones anteriores y de la unión de muchos esfuerzos por crear un lenguaje altamente robusto para controlar interactividad, acceso a servicios de comunicación y una gestión organizada de contenido multimedia. [BUILDER 2010].

- *Respaldo aceptable de proyectos de software libre a nivel mundial*: Existen una gran variedad de bibliotecas de AS3 que ayudan a extender las soluciones finales de proyectos a lo largo del mundo. Uno de sus principales promotores es Adobe. [BUILDER 2010].

Como desventaja principal se encuentra:

- *Adobe Flash Builder no es un software totalmente libre:* aun cuando se utilicen herramientas de Adobe que exporten en el formato libre SWF, adobe limita la utilización este software con fines no comerciales, se necesitan claves de activación del mismo para la explotación plena de sus componentes. [BUILDER 2010].

1.5. **Marco de trabajo seleccionado para el desarrollo**

Un marco de trabajo es una estructura de soporte definida mediante la cual un proyecto de software puede ser organizado y desarrollado. Brinda un conjunto de funcionalidades y bibliotecas de código que permiten la agilización del desarrollo de un proyecto, reutilizar los componentes y el código a gran escala.

1.5.1. **Adobe Flex Open Source SDK**

Flex es una tecnología reciente y de gran aceptación en el desarrollo de RIAs y aplicaciones para Flash Player, que en muy poco tiempo ha inundado la red con soluciones que oscilan entre sencillas aplicaciones y grandes sistemas empresariales, gracias a la calidad de su arquitectura en general. Su gran impacto vino impulsado por la aparición de la versión 2.0. El marco de trabajo actual está compuesto por una biblioteca de componentes de interfaz de usuario que ofrecen

un alto grado de interactividad y la capacidad de comunicarse con los sistemas de gestión de bases de datos más comunes. [FLEX 2009].

El SDK libre de Flex es desarrollado por Adobe y mantenido por diversas comunidades a lo largo del mundo que crean continuamente nuevos proyectos y mejoran los existentes. La principal línea de desarrollo de éstas es la elaboración de nuevos componentes para las comunicaciones y de marcos de trabajos arquitectónicos para la implementación y perfeccionamiento de aplicaciones en línea.

Su principal debilidad reside en que, al ejecutarse sobre Flash Player, se encuentra obstaculizado por las limitantes de acceso que presenta el reproductor, la cual se elimina con la aparición de Away3D a inicios del 2008 haciendo posible que se pueda seleccionar a Flex como uno de los marcos de trabajo a utilizar. [FLEX 2009].

Flex es clave para la creación de la presente solución debido a que aporta la biblioteca de componentes de interfaz de usuario necesaria para la gestión interna del software a desarrollar y, a su vez, el compilador y el depurador necesario para la programación del proyecto.

1.6. **Lenguaje de implementación**

Para la implementación de la solución se utilizará AS3, el lenguaje de programación de las herramientas antes mencionadas y de Away3D. A continuación se describen las características fundamentales del mismo y que se tuvieron en cuenta a la hora de proponer la solución existente.

1.6.1. **ActionScript 3.0**

- *Programación Orientada a Objetos*: Desde la versión 2.0 de AS es posible crear proyectos Flash Player totalmente orientados a objetos. Aún así, la biblioteca interna de esta versión tenía la característica de ser muy inestable en tópicos como seguridad y rendimiento. AS3, creado a mediados del año 2006, es considerado una revolucionaria evolución de la versión anterior, acompañada de

una serie de conceptos nuevos tomados de otros lenguajes de programación como Java y C++. [ACTIONSCRIPT 2010]

- *Alto rendimiento*: El rendimiento de AS3 es hasta 10 veces superior si se compara con sus anteriores versiones. La nueva máquina virtual para procesamiento de, se encuentra incorporada en las versiones de Adobe Flash Player 9 y superior. De igual forma se continúa incluyendo la versión anterior de la máquina virtual AS2 pese a las debilidades, debido a que cientos de miles de contenidos en la red continúan utilizando AS2. [GROSSMAN 2006]

- *Tratamiento de eventos orientado a objetos*: El tratamiento de eventos siempre ha sido una de las desventajas que los desarrolladores han tenido para crear proyectos Flash Player orientados a objetos. AS3 contiene en su marco de trabajo un paquete de clases exclusivamente dedicado a la gestión de eventos de una forma muy organizada y novedosa. Es una de las características que más resaltan sus creadores .[ACTIONSCRIPT 2010]

- *Alta integración con XML*: AS3 cumple con la reciente especificación de ECMAScript para XML ("ECMAScriptfor XML", en adelante E4X), por tanto incorpora el novedoso estándar para el tratamiento de XML. E4X es otra de las características que más ha propiciado la aceptación de AS3, debido a que la versión 2.0 contaba con una pobre interfaz de trabajo con el estándar de datos, lo que llevó a la creación de soluciones alternativas y a un replanteamiento del problema por parte de los creadores del lenguaje en esta nueva versión. [GROSSMAN 2006]

1.7. **Selección de un IDE para el desarrollo**

Los entornos de desarrollo de código AS3 para Flash Player no abundan. A continuación aparecen las alternativas estudiadas para la selección del mismo:

1.7.1. **Flash Develop**

Creado por el finlandés Mika Palmu como editor de código para MTASC[5] [TWEEN 2008] y SWFMill[6] [FISHER 2008], ha sido moderadamente utilizado por las comunidades de desarrolladores de aplicaciones para Flash Player a lo largo del mundo. Su alta difusión por la red y su calidad técnica, está dada por la pobre capacidad de manejo de código que presenta el IDE de Adobe Flash, siendo una alternativa factible inclusive para los propios creadores de la herramienta de Adobe. Recientemente ha sido adaptado para soportar haXe y el SDK libre de Flex. Una de sus principales desventaja es que está desarrollado utilizando la plataforma .NET de Microsoft, por lo que sólo funciona en la familia de sistemas operativos de la multinacional .[PALMU 2008].

1.7.2. **Adobe Flex Builder para Linux**

Adobe Flex Builder para Linux, la versión para Linux del popular entorno de desarrollo de Adobe, basado en Eclipse, aún se encuentra en un estado de pruebas que llega casi a un (1) año. Comparado con la versión actual para Windows y MacOS, la principal desventaja que tiene es que no cuenta con el entorno gráfico para la creación de las aplicaciones Flex. Aún continúa en desarrollo y no posee una versión comercial. Actualmente se encuentra publicada para descargar y usar con 400 días de prueba antes de expirar. Incorpora la capacidad de crear proyectos AIR[7] [AIR 2008] y Flex, así como depurarlos en tiempo de desarrollo y generar los proyectos finales. [INCORPORATED 2009b]. Debido a las capacidades técnicas

[5] **MTASC**: *MotionTween ActionScript Compiler* es el primer compilador de ActionScript libre. Creado por Nicolas Cannasse en el año 2005.

[6] **SWFMill**: Es un compilador a modo consola para crear archivos SWF y proveer con librerías a otros proyectos.

[7] **AIR**: Adobe Integrated Runtime una herramienta capaz de portar al escritorio cualquier aplicación Flash Player y proveerla de las funcionalidades para comunicarse con el sistema operativo cliente y para acceder a los servicios de redes.

recién expuestas y a la exigencia del uso de sistemas operativos libres, Adobe Flex Builder para Linux es la mejor alternativa para desarrollar la solución planteada.

1.8. **Metodologías de desarrollo**

La metodología a seleccionar en cualquier proceso de desarrollo de software es uno de los temas más complicados, ya que son muchos los factores que se deben incluir en el análisis. A continuación se hace un estudio de las metodologías más utilizadas y que se adapta a todo tipo de proyectos.

1.8.1. **Proceso Unificado de Desarrollo**

El Proceso Unificado de Desarrollo *(del inglés "Rational Unified Process", en adelante RUP)* es una la metodologías robustas más popular que existe. Sus principales características son:

- *Guiado por Casos de uso*: Los cuales sirven para describir el comportamiento del sistema y elaborar los casos de prueba con los que se comprueba que el sistema desarrollado cumple con los requerimientos del cliente. [LARMAN 2004a]
- *Centrado en la arquitectura*: La arquitectura del sistema es la columna vertebral de todo el desarrollo del mismo. Cada iteración gira en torno a la misma, fortaleciendo y corrigiendo sus características. [LARMAN 2004a].
- *Iterativo e incremental*: En cada ciclo de iteración se produce una nueva versión del software. [LARMAN 2004b]

Utiliza UML como lenguaje de modelado y cuenta con varias fases de trabajo en las cuales se desarrolla una serie de flujos fundamentales del desarrollo del proyecto. [KRUCHTEN 2002]

1.8.2. **Programación Extrema**

La Programación extrema *(del inglés "eXtreme Programming", en adelante XP)* es la metodología más popular de las metodologías ágiles. Sus principales características son:

- *Retroalimentación con el cliente*: Conceptualmente, al menos uno de los miembros del equipo de trabajo del proyecto, es un cliente. Esto propicia una constante interacción del mismo con el producto en desarrollo. [SHORE and WARDEN 2007].
- *Cortas iteraciones*: En cada iteración, se obtiene un producto listo para entregar y que tiene valor para el cliente. La entrega continua de resultados compromete a ambas partes en la evolución del proyecto e indirectamente influye de forma positiva en la calidad del producto final. [SHORE and WARDEN 2007]
- *Muy flexible a cambios*: Una de las características más reconocidas de XP. La constante retroalimentación con los clientes permite prever futuros cambios y evita llegar a momentos que paralizan el desarrollo del producto. [SHORE and WARDEN 2007].

Cuenta con una serie de prácticas que producen un aumento de la productividad del equipo de trabajo, tales como la programación en pares, reuniones diarias y planes de entrega a corto plazo, por mencionar algunos. Al igual que RUP, también utiliza UML si el equipo de desarrollo lo decide.

1.9. **Selección de la metodología de desarrollo**

En ambas metodologías se puede hacer un recorte amplio de roles y artefactos para adaptar el proyecto a equipos de trabajos compuestos por solo dos (2) personas. RUP es conocido por la robustez de su proceso de desarrollo a largo plazo y XP por la rapidez a corto plazo de las entregas. Es debido a ello y a las cuestiones que aparecen a continuación, que XP es la metodología seleccionada para el desarrollo de la presente solución:

- El período de desarrollo es corto: El desarrollo de la solución se limita cuatro (4) meses de trabajo continuo solamente.
- El cliente forma parte del equipo de desarrollo.
- Las dimensiones del proyecto son pequeñas.

Uno de los objetivos específicos es publicar versiones de pruebas para obtener retroalimentación de diferentes probadores de la universidad y así optimizar el producto. Para ello es necesario tener un período de entregas corto, respaldado por iteraciones cortas y un proceso de desarrollo ágil.

1.10. **Herramienta CASE para el modelado**

Las herramientas CASE (del inglés “Computer Asisted Software Enginering”, ingeniería de software asistida por computadora) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas ayudan en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código, compilación automática, documentación o detección de errores entre otras.

1.10.1. **Visual Paradigm para UML**

Como herramienta para el modelado de la solución se ha seleccionado Visual Paradigm para UML (en adelante Visual Paradigm), una de las líderes del mercado de las llamadas herramientas CASE.[SIERRA 2007].A continuación se ofrece una lista de las características principales que se tuvieron en cuenta para la selección del mismo:

- *Soporte para la versión 2.1 de UML*: La metodología XP utiliza muchos de los diagramas que provee UML. La versión más reciente de este lenguaje es la de mayor uso a nivel mundial y la que más documentación posee.[SIERRA 2007].

- *Interoperabilidad entre diagramas*: Es capaz de exportar los diagramas de un modelo a otro con mucha facilidad. Esto ahorra considerables cantidades de tiempo.[SIERRA 2007].

- *Generación de código AS3 desde los diagramas*. Uno de los diagramas más utilizados de UML en XP es el diagrama de clases del diseño para definir, iterativamente, la flexibilidad de la arquitectura y generar el código a partir de éste. Visual Paradigm es una de las pocas herramientas capaz de generar esta versión del lenguaje AS3.[PARADIGM 2008].

1.11. **Consideraciones finales**

En el presente capítulo se realizó un estudio sobre el estado del arte, así como una búsqueda de las herramientas existentes con características similares a la solución. De esta manera se logra apreciar los recientes cambios que ha sufrido la plataforma Flash, especialmente en las licencias de Flex SDK y la aparición de Away3D, las cuales propician un ambiente ideal para combinarlos y crear un marco de trabajo para la visualización 3D sobre entornos en línea .Por tanto la universidad dispondrá de una alternativa para revolucionar el proceso de creación de este tipo de productos.

IsoMap 3D

Isometric 3D Framework

Capítulo 2

Presentación de la solución propuesta

Capítulo 2: Presentación de la solución propuesta

El presente capítulo tiene como objetivo principal describir las características fundamentales del sistema. Luego se expondrán de una manera clara, los requerimientos a cumplir por el mismo.

2.1. **Descripción del ambiente de desarrollo**

La descripción inicial de cualquier solución informática es una idea que generalmente se viene meditando antes de una reunión formal con el equipo de desarrollo de software. Es una visión que permite a los clientes imaginar y percibir una utilidad inmediata de uso de la misma, ya que, son ellos mismos quienes proponen esta idea a partir de sus necesidades. De ahí que sea muy común escuchar, en los primeros encuentros, peticiones informales de requisitos y representaciones de la interfaz de usuario, las cuales aportan mucho sobre la concepción que tienen del mismo.

XP propone, para los primeras reuniones con ellos, realizar un sencillo levantamiento de requisitos, en el cual, se tomen todos los aspectos relacionados con sus necesidades. En estos encuentros se pueden incluir prácticas muy efectivas de captura de requerimientos como:

- Entrevistas frecuentes para aclarar dudas que de manera continua puedan ir apareciendo.
- Tormenta de ideas entre el equipo de desarrollo y el equipo de trabajo del cliente.
- Observación de los procesos a informatizar manteniendo una presencia lo más imperceptible posible para no obstaculizarlos.
- Juegos de rol donde el cliente simule ser un usuario que interactúa con el sistema, este segundo simulado por el desarrollador.

A continuación se especifica el alcance del producto a desarrollar de manera similar a la que un cliente describiría en reuniones anteriores con el mismo. Generalmente los detalles técnicos no se incluyen en esta descripción, pero cabe aclarar que el cliente los conoce y no por esto dejan de ser requerimientos, por lo que deben ser cumplidos para lograr satisfacer sus necesidades:

Como en la mayoría de los entornos de desarrollo para la creación de mapas, el sistema contará con un editor gráfico en el cual se podrá seleccionar los tipos de superficies deseados, ver las propiedades y descripción de las mismas, así como una vista previa de la selección. Teniendo además, en la esquina superior derecha, un panel que muestra la vista previa de la selección realizada. Seguidamente un panel desplegable que contiene los distintos tipos de superficies deseadas con sus propiedades y descripciones incluidas.

Es necesario aclarar los términos tipo de superficies y propiedades que no son más que la selección de las superficies deseada para aplicar al mapa y las propiedades de las superficies respectivamente.

Para aplicar una superficie deseada sólo se hace click sobre la misma para seleccionarla y mostrar una vista previa de ella. Para aplicarla en el mapa como tal, nuevamente se hace click en el área donde se desea insertar. Las superficies podrán insertarse de forma consecutiva en el espacio sin necesidad de volverlas a seleccionar.

Para aplicar las diferentes superficies, la herramienta centrará su funcionamiento en un marco de trabajo gráfico propio y libre que permitirá a los desarrolladores utilizarlo posteriormente en otros proyectos independientemente del entorno de desarrollo en que programen.

El panel desplegable contiene además los distintos modelos que pueden ser aplicados al mapa, teniendo en cuenta las propiedades y dimensiones de los objetos, entre ellos se encuentran:

- Casas: Selección de las diferentes modelos de casas existentes.

- Edificios: Selección de los modelos de edificios existentes.

En la parte superior derecha se encuentra el visor de los objetos seleccionados. Este componente posibilita visualizar una vista previa de todos los objetos que se seleccionen de los paneles desplegables, ya sean superficies o modelos.

Una vez terminado el mapa, el usuario podrá guardar el mismo con los últimos cambios realizados. Este proceso se realizará mediante la creación de un archivo XML, garantizando así su reusabilidad para futuros cambios en el mapa, además de garantizar la compatibilidad con los navegadores web.

En la parte derecha a los paneles de vista previa y selección de los objetos a insertar en el mapa se encuentra un componente que contiene botones, ellos realizan las siguientes funciones:

- Desplazar la cámara hacia la dirección deseada (derecha, izquierda, arriba, abajo, centrar la cámara).

- Rotar el mapa 45 grados en sentido de las manecillas del reloj o viceversa.

Este panel también permite un ajuste de acercamiento o alejamiento de la cámara respecto al mapa en cuestión. La Figura 1 muestra un boceto general de la aplicación anteriormente descrita:

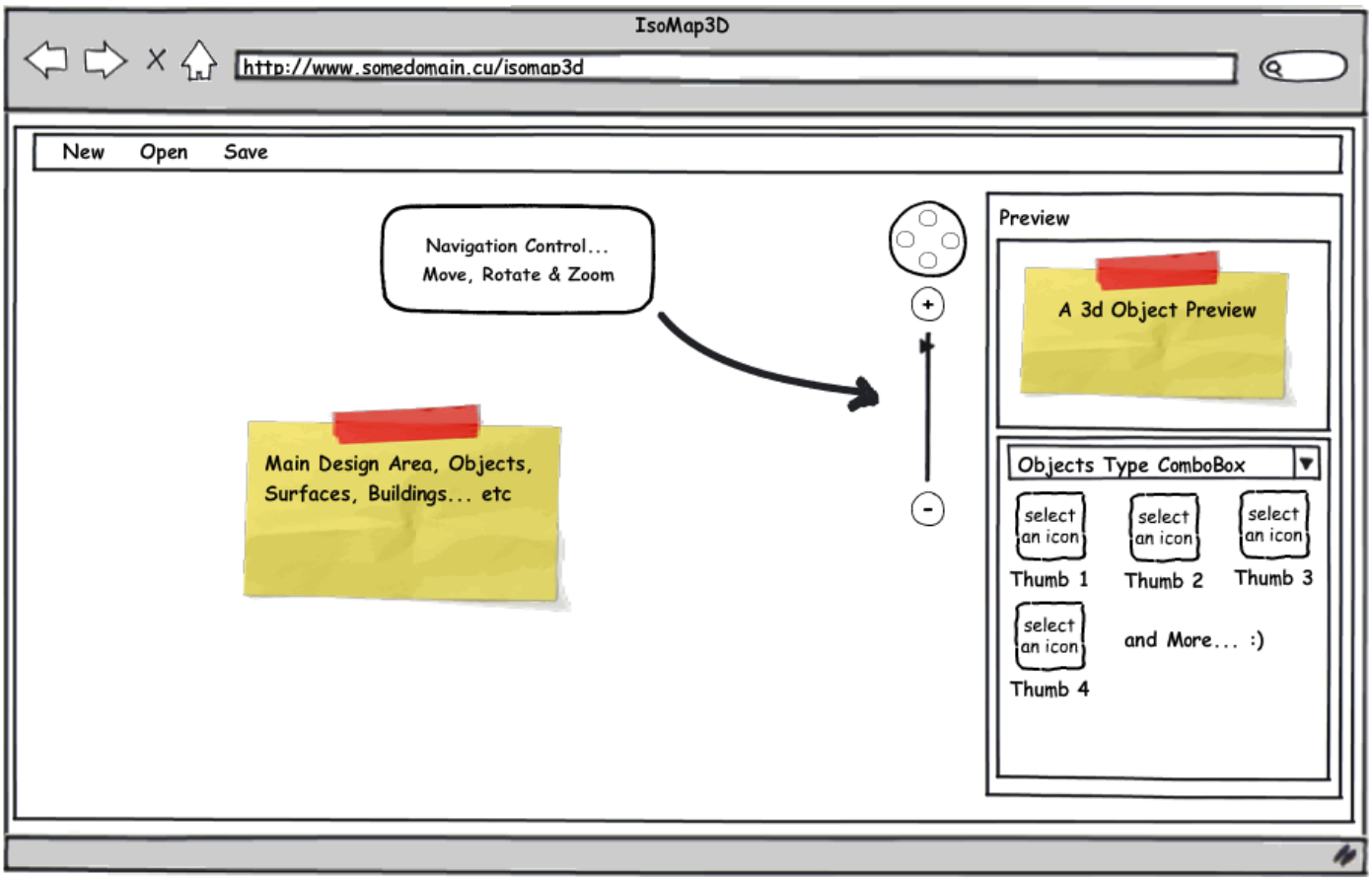

Figura 1. Boceto conceptual de la aplicación.

2.2. **Clientes relacionados con el sistema**

Los clientes relacionados con el sistema son todos aquellos que obtienen un resultado del mismo.

Personas relacionadas con el sistema.	**Justificación.**
	El usuario interactúa directamente con el sistema en la creación de mapas. En caso de no tener, o tener pocos conocimientos sobre el trabajo con la

Usuario.	aplicación, puede acceder a los tutoriales de instrucción en línea para adquirir los conocimientos básicos de trabajo con la misma. Puede ser tanto un diseñador gráfico sin conocimientos de programación en AS como un desarrollador que utiliza la herramienta para crear mapas 3D y luego dotarlos de interactividad en Flex.

Tabla 1. Clientes relacionadas con el sistema.

2.3. **Lista de reservas del producto**

La lista de reservas del producto está compuesta por una serie de requisitos que representan las tareas que el sistema debe realizar para trabajar correctamente, siempre que estas sean las que realmente el cliente quiere. Es por ello que, redactar un listado de requerimientos bien detallados y que posteriormente puedan ser probados, es muy importante.

R1. Emplazar objetos en el mapa.

R1.1- Seleccionar la categoría Superficie.

R1.2- Insertar las superficies de terreno seleccionadas.

R1.3- Seleccionar la categoría Edificios.

R1.4- Insertar las edificaciones seleccionadas.

R2. Utilizar un núcleo gráfico propio para la creación de mapas.

R3. Editar la posición de los objetos.

R3.1- Editar la posición de la superficie seleccionada.

R3.1.1- Rotar la superficie seleccionada.

R3.2- Editar la posición de la edificación seleccionada.

R3.2.1- Rotar la edificación seleccionada.

R4. Ajustar la visión de la cámara respecto al mapa.

R4.1- Mover la cámara hacia la derecha.

R4.2- Mover la cámara hacia la izquierda.

R4.3- Mover la cámara hacia la arriba.

R4.4- Mover la cámara hacia la abajo.

R4.5- Centrar la cámara.

R5. Mostrar una vista previa del objeto seleccionado.

R6. Rotación del mapa.

2.4. **Características no funcionales del sistema**

Los aspectos no funcionales son propiedades o cualidades que el producto debe tener según las necesidades del cliente y las buenas prácticas de programación. Son características del mismo que lo hacen atractivo, usable, rápido y confiable.

2.4.1. **Diseño e implementación**

- Utilizar AS3 para el desarrollo del producto.
- Utilizar programación orientada a objetos.
- Aplicar patrones de diseño, principalmente para el desarrollo de editor visual y del núcleo gráfico propio. Para el núcleo gráfico es imprescindible tener en cuenta aspectos como el rendimiento y la robustez.

- Utilizar una metodología de desarrollo ágil que permita la entrega de versiones de manera regular.
- Utilizar un estándar de ficheros propio y libre para los mapas basado en XML.

2.4.2. **Apariencia o interfaz externa**

- Utilizar componentes gráficos de Flex para el desarrollo de la interfaz de usuario.

2.4.3. **Usabilidad**

- El sistema debe proporcionar una interfaz sencilla para que todos los usuarios puedan utilizarla sin necesidad de poseer habilidades técnicas.

2.4.4. **Ayuda y documentación en línea**

- La documentación del núcleo gráfico debe ser realizada utilizando una herramienta generadora de documentación.

2.4.5. **Software**

- El sistema debe ser independiente de plataforma. Debe ejecutarse tanto en Microsoft Windows como en GNU/Linux.

2.5. **Consideraciones finales**

En este capítulo se han abordado los aspectos referentes a la concepción del producto a construir, así como las características funcionales y no funcionales que debe cumplir. Teniendo esta información, es entonces que se pasa a la fase de planificación y exploración.

IsoMap 3D

Isometric 3D Framework

Capítulo 3

Exploración y planificación

Capítulo 3: Exploración y planificación

En este capítulo se hace alusión a las fases de exploración y planificación, las cuales son las dos primeras de la metodología de desarrollo XP. El objetivo principal de éstas es conocer el alcance del producto a desarrollar y estimar los tiempos de entrega de cada versión. Se expone, además, los artefactos que se generan a partir de los requerimientos expuestos por el cliente en el capítulo anterior.

3.1. **Exploración**

La exploración es la etapa del proceso de desarrollo de software que propone XP para comenzar la construcción de un producto. Una vez que los clientes entregan su propuesta al equipo de trabajo, comienza el análisis en grupo, las horas en los pizarrones, las tormentas de ideas y la conceptualización del software. Un aspecto clave en el esclarecimiento de las dudas sobre los procesos a automatizar está dado por la integración del cliente al equipo de desarrolladores desde el mismo inicio del proyecto. [MARCHESI and SUCCI 2003].

3.2. **Historia de Usuario**

Las historias de los usuarios (HU) son como los casos de uso de RUP, con la diferencia de que no deben ser descritos en más de tres líneas e idealmente es el cliente quien las redacta y prioriza. Además de esto, se le suma un tiempo estimado de desarrollo que lo define el propio equipo del proyecto. En esencia, no son más que las ideas del cliente organizadas y agrupadas de acuerdo a su funcionalidad, teniendo en cuenta un orden que permita priorizar sus necesidades, así como definir las que resultan críticas o claves en el momento de desarrollo de la solución.

Es necesaria la claridad de la descripción ya que en ella radica el éxito del proyecto. La comprensión errónea o la falta de comunicación entre el cliente y el equipo de desarrollo son las principales causas de fracaso de los proyectos. Es por ello que las HU juegan un papel tan importante en este proceso y se les dedica totalmente

una fase en el ciclo de vida de un proyecto XP. A continuación aparecen descritas las HU de la presente solución:

Historia de Usuario	
No.: 1 Nombre: Crear elementos gráficos utilizando un núcleo de visualización de la aplicación.	
Usuario: Usuario	
Prioridad en el Negocio: Alta.	**Nivel de Complejidad**: Alta.
Estimación: 3 semanas	**Iteración Asignada**: 1
Descripción: El usuario puede crear cualquier tipo de elemento gráfico tridimensional utilizando un marco de trabajo o el núcleo gráfico de AS3.	
Información adicional (Observaciones): Da cumplimiento al requisito R2. "Utilizar un núcleo gráfico propio para la creación de mapas". El uso del núcleo gráfico es independiente al entorno de desarrollo. Puede ser utilizado por cualquier usuario que lo importe a un proyecto de AS3 o Flex. El entorno de desarrollo centra su funcionamiento en él.	

Historia de Usuario	
No.: 2 Nombre: Ubicar objetos en el mapa.	
Usuario: Usuario	
Prioridad en el Negocio: Media.	**Nivel de Complejidad**: Media.
Estimación: 3 semanas.	**Iteración Asignada**: 2
Descripción: El usuario selecciona uno (1) de los cuatro (4) objetos de	

superficie o una (1) edificación que desea insertar. Posteriormente inserta el objeto para la conformación del mapa ideado.
Información adicional (Observaciones): Da cumplimiento al requisito R1. "Emplazar objetos en el mapa".

Historia de Usuario	
No.: 3 Nombre: Edición de posición de objetos.	
Usuario: Usuario	
Prioridad en el Negocio: Media.	**Nivel de Complejidad**: Media.
Estimación: 3 semanas.	**Iteración Asignada**: 3
Descripción: El usuario selecciona el objeto que se encuentra en el mapa y desea editar su posición (superficies o edificaciones). Posteriormente edita la posición del objeto que se encuentra en el mapa en el sentido de las manecillas del reloj.	
Información adicional (Observaciones): Da cumplimiento al requisito R3. "Editar la posición de los objetos".	

Historia de Usuario	
No.: 4 Nombre: Ajuste de visión de cámara.	
Usuario: Usuario	
Prioridad en el Negocio: Media.	**Nivel de Complejidad**: Media.

Estimación: 3 semanas.	**Iteración Asignada**: 3
Descripción: El usuario mediante un componente desplaza la posición de la cámara en la dirección deseada (arriba, abajo, derecha, izquierda), además de poder centrarla desde cualquier posición que se encuentre la misma respecto al plano del mapa.	
Información adicional (Observaciones): Da cumplimiento al requisito R4. "Ajustar la visión de la cámara respecto al mapa".	

Historia de Usuario	
No.: 5 Nombre: Vista previa.	
Usuario: Usuario	
Prioridad en el Negocio: Baja.	**Nivel de Complejidad**: Baja.
Estimación: 2 semana.	**Iteración Asignada**: 4
Descripción: El usuario puede visualizar una vista 3D del objeto que selecciona.	
Información adicional (Observaciones): Da cumplimiento al requisito R5. "Mostrar una vista previa del objeto seleccionado".	

Historia de Usuario	
No.: 6 Nombre: Rotar Mapa.	
Usuario: Usuario	
Prioridad en el Negocio: Media.	**Nivel de Complejidad**: Media.

Estimación: 2 semana.	**Iteración Asignada**: 4
Descripción: El usuario puede rotar el mapa en sentido de las manecillas del reloj o contrario a las manecillas del reloj.	
Información adicional (Observaciones): Da cumplimiento al requisito R6. "Rotación del mapa".	

Para la duración de las semanas que se tuvieron en cuenta en las estimaciones de las historias de usuario anteriores, es necesario aclarar que una (1) semana equivale a los cinco (5) días laborales de la misma. Se considera válida esta aclaración ya que generalmente, los cálculos erróneos de estimación de los tiempos de desarrollo se realizan en base a los siete (7) días de la semana.

Mes de ejemplo						
Domingo	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Días feriados.	Fin de semana.	Próximo mes.	Días laborales.

Tabla 2. Análisis de los días laborales de un mes de ejemplo.

3.3. **Planificación**

Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar todas las historias de usuario, partiendo del tiempo asignado a cada una en la fase de exploración. Una vez terminado esto, se procede a organizarlas en las iteraciones correspondientes, teniendo en cuenta la prioridad especificada por el cliente y del tiempo de desarrollo de cada una.

3.3.1. **Iteraciones**

Una iteración no es más que un mini-proyecto que se realiza. Al finalizar cada iteración, se obtiene un resultado concreto con valor parcial para el cliente, teniendo en cuenta quedará totalmente satisfecho al finalizar la última iteración, ya que es la que concluye el producto acordado inicialmente.

En la organización de las iteraciones se debe evitar extender más de un (1) mes laboral; por lo general se proponen entre 20 y 21 días. Los plazos de entrega o retroalimentación atentan contra el cumplimiento de los objetivos del cliente, ya que sometidos a pequeños cambios de manera constante. Esto sucede a pesar de que cuando un miembro de su equipo de trabajo se encuentre incluido en el equipo de desarrollo.

La mejor forma de evitar lo anterior es haciendo verificaciones del sistema en el entorno de despliegue donde radicará finalmente la solución. Las pruebas de los usuarios finales, las correcciones a las que se somete la aplicación en las revisiones y el análisis a partir de casos de prueba, son la mejor forma de verificar que el software avanza por el camino correcto. En resumen, alargar una iteración por más de un (1) mes, es una práctica muy negativa para el desarrollo de un producto.

Iteraciones	Orden de las Historias de Usuario a implementar	Cantidad de tiempo de Trabajo
Iteración 1	1- Crear elementos gráficos utilizando un núcleo de visualización de la	3 semanas

	aplicación.	
Iteración 2	2- Ubicar objetos en el mapa.	3 semanas
Iteración 3	3- Edición de posición de objetos 4- Ajuste de visión de cámara.	3 semanas
Iteración 4	5- Vista previa. 6- Rotar Mapa.	2 semanas

Tabla3. Historias de usuario organizados por iteraciones.

3.4. **Plan de entregas**

El plan de entregas es el compromiso final del equipo de desarrollo con los clientes. Es una cuestión de vital importancia para el negocio entre ambas partes, ya que la entrega tardía de la solución, repercute notablemente de manera negativa en el desarrollo del producto creando insatisfacción en el cliente. La estimación es uno de los temas más complicados del desarrollo de un proyecto de software y es por ello que resulta esencial tener bien claros los requerimientos del cliente y el estilo de trabajo del equipo de desarrollo para realizar una entrega de la solución con un máximo de calidad.

En el más extremo de los casos, la honestidad debe prevalecer en lugar de la incertidumbre y saber decir cuando se puede terminar el proyecto a tiempo o no. Es mejor ser claros con los clientes que defraudarlos luego de haber malgastado su tiempo y sus recursos.

La siguiente tabla muestra el control de versiones que se debe tener al final de cada iteración, que se realizan en la cuarta semana de los meses de enero, febrero, abril y mayo respectivamente.

Entrega	1ra iteración	2da iteración	3ra iteración	4ta iteración
IsoMap3D	----	Versión 0.1	Versión 0.3	Versión 0.5

Tabla 4. Control de versiones y de entrega.

3.5. **Consideraciones finales**

En el este capítulo se realizó la documentación de la primera etapa del ciclo de vida de la solución propuesta. Se describieron claramente las historias de usuario, el plan de iteraciones y el plan de entregas.

IsoMap 3D

Isometric 3D Framework

Capítulo 4

Implementación y pruebas

Capítulo 4: Diseño del sistema, implementación y pruebas

La metodología XP no propone concisamente los artefactos a utilizar en la implementación de la solución. Dicha metodología los deja en manos del equipo de desarrollo, dependiendo de las capacidades de comunicación, la decisión de generar tantos tipos de diagramas UML como se necesiten, y así, facilitar el proceso de desarrollo. En el presente capítulo se hace alusión al diseño del sistema, los diagramas utilizados, las tareas generadas por cada historia de usuario y al proceso de pruebas utilizado.

Con el objetivo de facilitar la comprensión del presente trabajo, se explican a continuación algunas características y observaciones que se tuvieron en cuenta para el desarrollo del mismo y que se consideraron válidas para ser incluidas en el inicio del capítulo.

El marco de trabajo de Adobe Flex.

Flex incorpora en su núcleo un marco de trabajo desarrollado y compuesto por archivos de AS3 que abarcan una amplia gama de funcionalidades. Estas contienen desde servicios de redes hasta componentes de interfaz de usuario, utilidades y otros. Al crear un nuevo proyecto Flex todas estas clases son compiladas a través del propio compilador del SDK y los ficheros son generados.

Se ha seleccionado para la realización de la solución, como se explicó en el Capítulo 1, debido a que brinda una biblioteca de componentes gráficos para el desarrollo de la interfaz visual de la misma y el compilador que permite la configuración y programación de los eventos asociados a cada componente. De igual manera realiza aportes de forma indirecta, a la concepción de los componentes visuales que se pretenden obtener en las historias de usuario Nº 1 y 2. Un punto de partida favorable para la materialización de un núcleo gráfico propio

de la aplicación en cuestión, es el hecho de que Flex esté desarrollado sobre ActionScript y a su vez genere este tipo de código.

Para el trabajo con el marco de trabajo de Adobe Flex, fue seleccionado el entorno de desarrollo de sus creadores ya que provee un ambiente integrado muy práctico, donde convergen las bibliotecas, el compilador y el depurador, siendo este último un factor muy importante en tiempo de desarrollo para la realización de las pruebas.

Motor Away3D.

Away3D tiene una amplia integración con código AS3 que permite la visualización de componentes generados con AS3 para mostrar elementos y objetos 3D en tiempo real. Este motor cuenta con una amplia documentación, comunidades de usuarios, ejemplos prácticos, aplicaciones muy utilizadas en la creación de juegos en línea y un robusto soporte.

Adobe Flex Builder permite utilizar el motor Away3D para la creación y visualización de objetos 3D generados a partir de código AS y que el proceso de depuración sea aún más rápido.

Visual Paradigm para UML.

Visual Paradigm está orientado al trabajo con UML que, aunque no fue creado directamente para una metodología ágil, permite ser utilizado para el trabajo con artefactos importantes como son los diagramas de clases, los de colaboración, de paquetes, de despliegue, entre otros.[SCOTT 2007]. Adicionalmente, Visual Paradigm permite generar código AS a partir de los diagramas de clases, ofreciendo una gran flexibilidad al proceso de desarrollo del núcleo y la arquitectura de la aplicación.

4.1. **Diagrama de clases del sistema**

El diagrama de clases del sistema describe la arquitectura y las relaciones entre cada una de estas clases. Es la vista más detallada de la solución.

En la figura 7 se muestra el diagrama de clases del sistema. Se observa la relación entre las clases principales del núcleo gráfico de IsoMap3D (Suface, Building, GridMap) y su integración mediante la extensión de clases del núcleo de Away3D. Las ventanas de visualización (Viewports) son extendidas propiamente de la clase general *UIComponent* perteneciente a Flex SDK, lo que define que podrán ser agregadas como componentes visuales a cualquier solución deseada. Los componentes de navegación y control de objetos (NavControls, ObjectControl) se encuentran directamente relacionados con la clase *ControlEvent*, la cual extiende de la clase general *Event* perteneciente también a Flex SDK y manejará los eventos propios del núcleo de IsoMap3D.

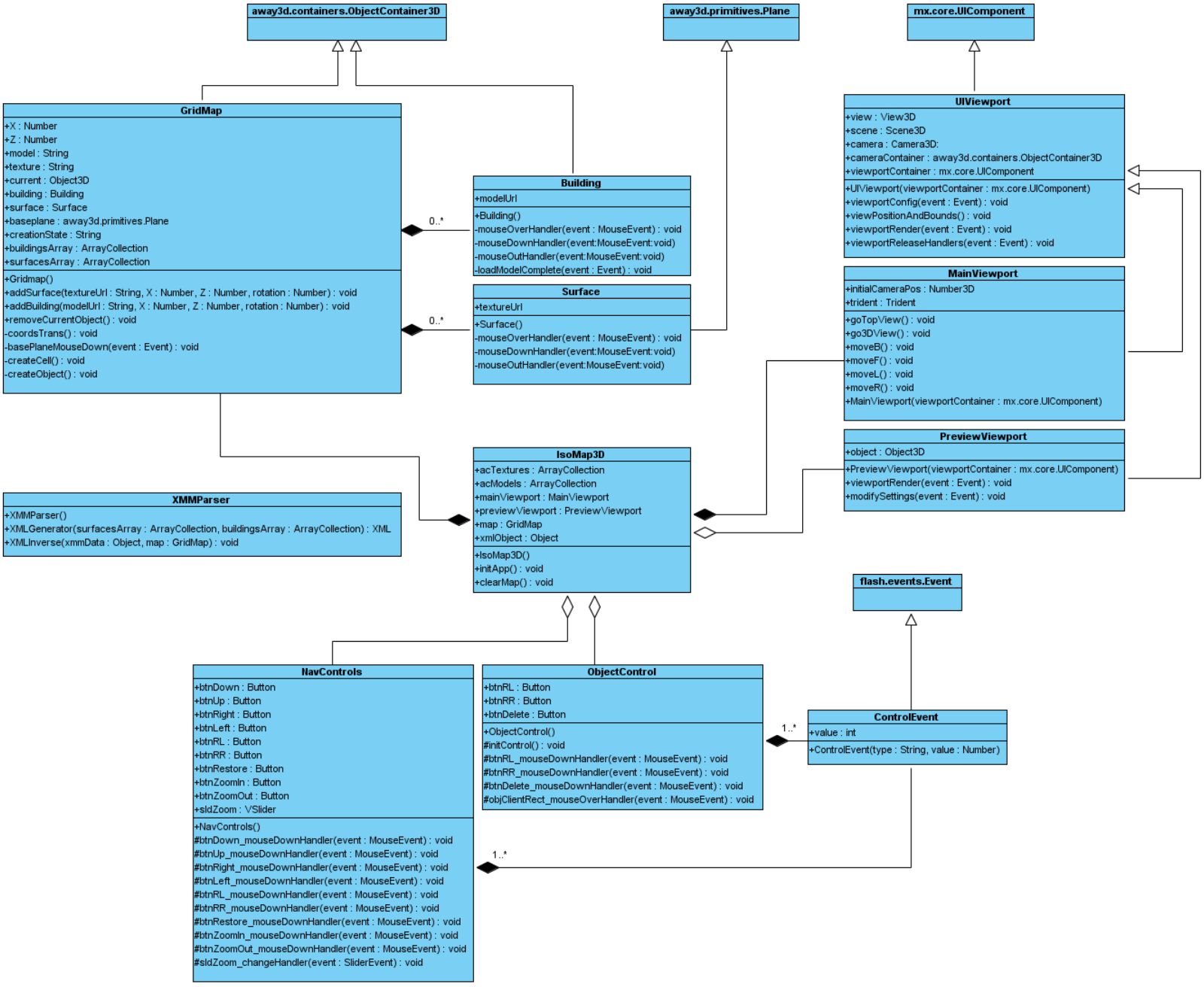

Figura 2. Diagrama de clases del sistema.

4.2. **Paquetes del sistema**

Para organizar el código en cada módulo, se creó una arquitectura centrada en el núcleo gráfico siguiendo la estructura convencional que se utiliza en los proyectos libres de AS y Java. La estructura de paquetes es inversa a la que se asigna a los nombres de dominio de los sitios Web. Por ejemplo, si se crea un proyecto para *www.misitio.cu*, la estructura de los paquetes es *cu* como paquete principal, *misitio* como secundario e internamente se organizaría el proyecto. Finalmente quedaría *cu.misitio* como el paquete raíz.

Siguiendo la estructura anteriormente planteada, el paquete de clases de la arquitectura de la solución quedaría como se muestra en la Figura 5.

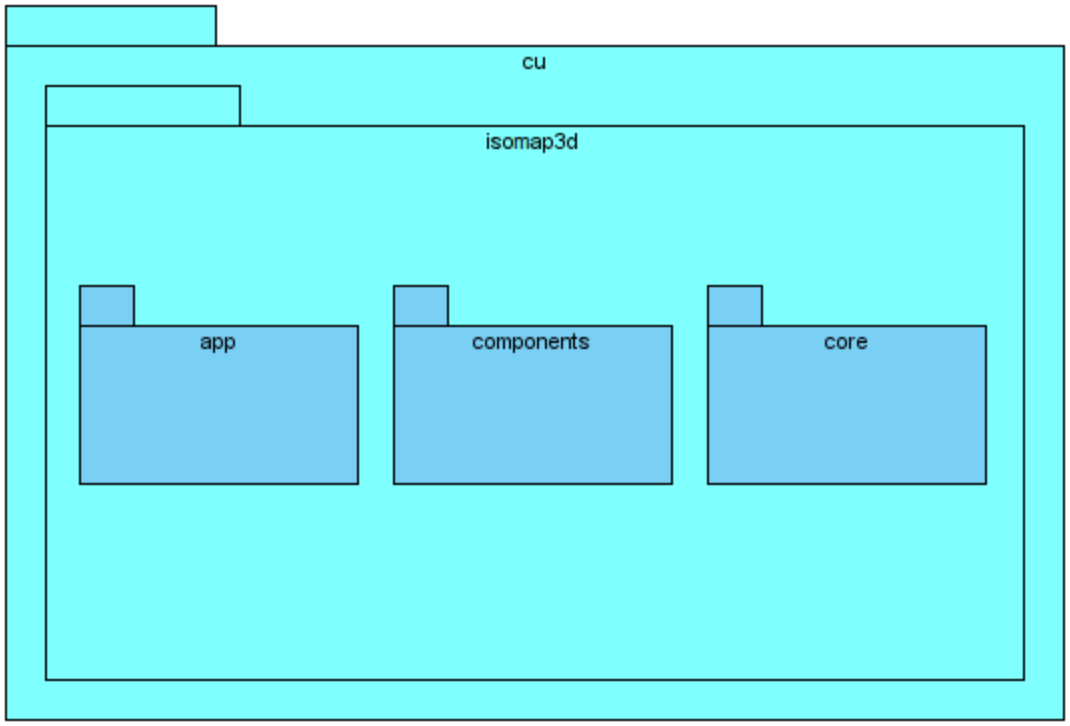

Figura 3. Diagrama de paquetes. Arquitectura de la solución propuesta.

core: Es el núcleo gráfico de la solución. Puede ser utilizado independientemente de los otros paquetes para la creación y edición de gráficos 3D utilizando código AS.

app: Es el paquete de la aplicación. Contiene la vista principal y los estilos de la aplicación. Toda la gestión de la misma se realiza internamente, por lo que puede ser modificado con la seguridad de que los cambios sólo afectarán a la aplicación y no al núcleo gráfico.

components: Contiene un componente para la navegación y otro para el control de objetos, además incluye una clase para el lanzamiento y captura de eventos directamente relacionados con estos componentes. Es la base para la extensión de nuevos controles.

Es necesario aclarar el IDE Adobe Flex Builder está programado sobre código AS con una estructura Modelo-Vista-Controlador fractal (en adelante MVC) basado en el marco de trabajo Cairngorm, este último hace que en la vista sea simple manejar los componentes del programa, ya que se permite enlazar fácilmente los objetos del modelo y escala muy bien múltiples servicios y/o iteraciones similares, por tanto la plataforma estructura sus paquetes haciendo uso del mismo. Debido a ello la aplicación contiene la estructura MVC evidenciando así el patrón Observador el cual es clave para la implementación del sistema.[KRASNOSHCHOK 2009].

4.3. **Relación paquete-módulo**

Adobe Flex permite definir relaciones directas entre módulos, por lo que cada módulo puede obtener información de otro a través de interfaces previamente definidas o accediendo al módulo principal. El uso abusivo de esta relación tiende a que exista un alto acoplamiento entre todas las partes del sistema, una práctica muchas veces riesgosa y que hoy en día la principal causa de la poca flexibilidad de los proyectos creados con Flex.[BUNTEL 2010].

Debido a esto se decidió que el módulo principal iniciara todas las instancias del patrón MVC partiendo de su relación con otros, mientras que los demás módulos utilizan exclusivamente el paquete de utilidades.

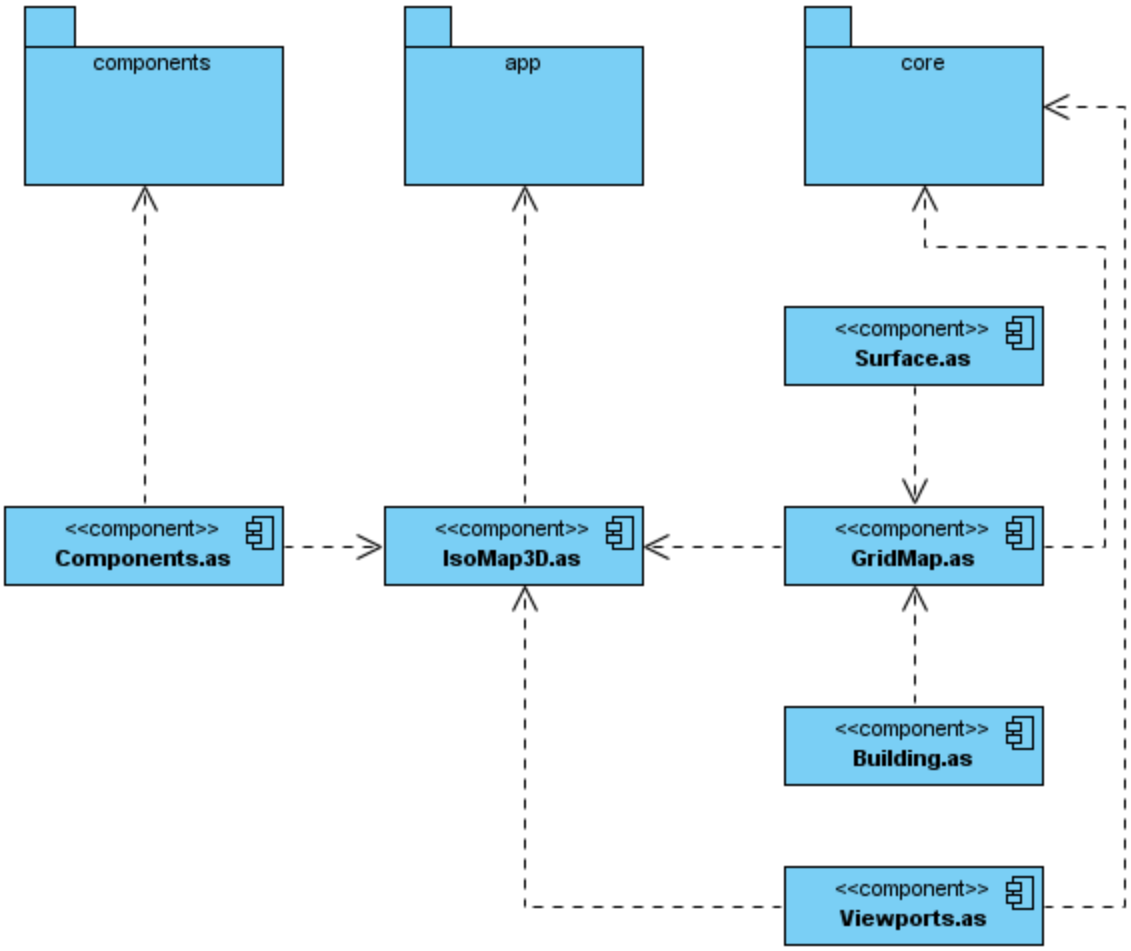

Figura 4.Diagrama de paquetes y módulos. Relaciones.

4.4. **Descripción de la Arquitectura.**

Para el desarrollo de la interfaz de usuario se utilizó una arquitectura modular soportada sobre el patrón Modelo Vista Controlador (en adelante MVC). Flex permite crear aplicaciones modulares, las cuales no son más que diferentes paquetes (ya sea para la agrupación de clases, imágenes y otros.), comunicándose entre sí, práctica que favorece la flexibilidad de las soluciones al no estar centralizado todo en un mismo lugar. Además permite que nuevos módulos sean integrados a la herramienta sin necesidad de modificar directamente el núcleo de la misma.

Para ello se confeccionó un diagrama de componentes de la solución formado por los módulos de Flex que se muestra en la figura 2.

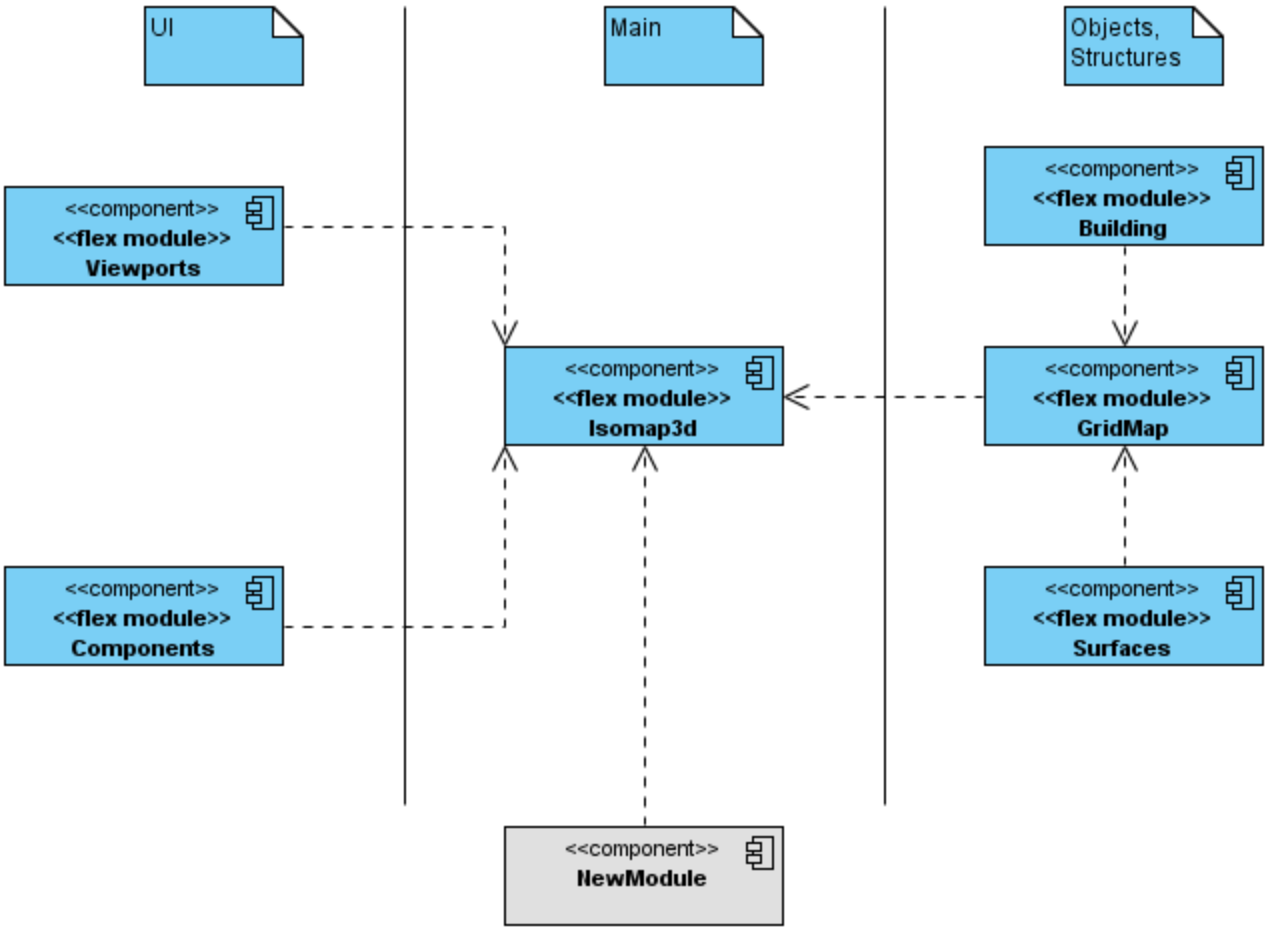

Figura 5. Diagrama de componentes. Módulos de Flex.

En la figura se puede observar que al final del diagrama aparece NewModule, que representa un módulo futuro que se le puede agregar a la solución en vista a aumentar las capacidades del núcleo.

Los módulos se dividen en tres (3) grupos, los cuales se describen a continuación:

4.5. **Módulo central**

- **Isomap3D**: Es el módulo principal de la aplicación. Contiene todas las instancias del Modelo Vista Controlador y a su vez, se encarga de la gestión de carga de los otros módulos.

4.6. **Componentes de interfaz de usuario (Viewports y Components)**

Los módulos de interfaz de usuario de la aplicación son los que se integran al ambiente de desarrollo en forma de componentes visuales. Ver Figura 3.

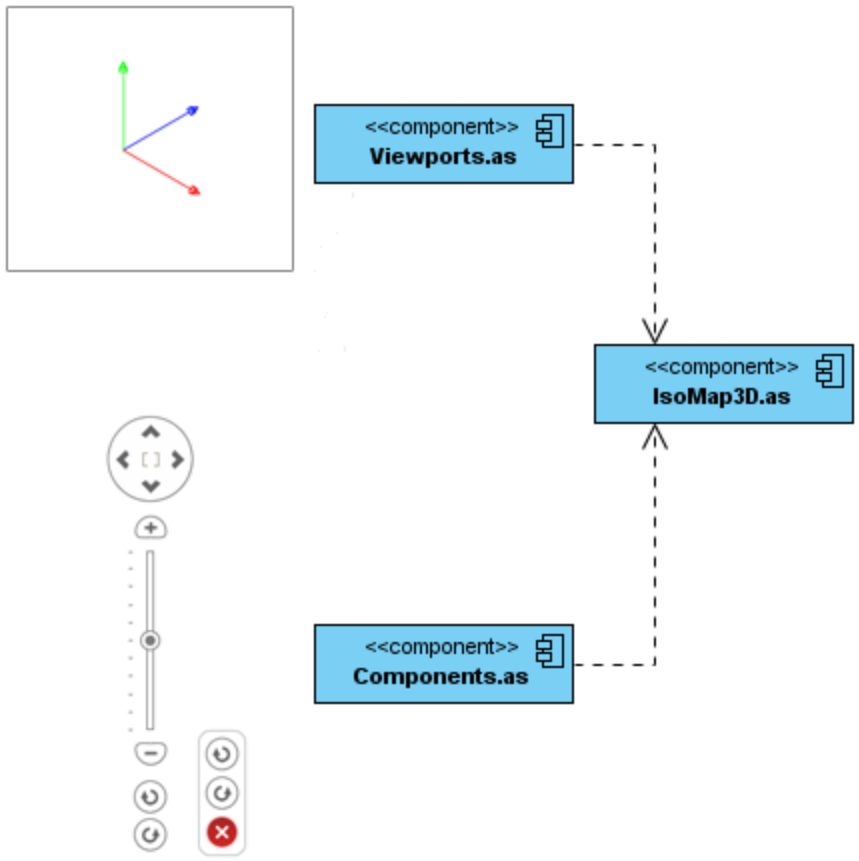

Figura 6. Diagrama de componentes. Componentes de interfaz de usuario.

- **Viewports**: Módulo de visualización. Su función principal es contener y mostrar los objetos gráficos insertados y seleccionados desde una lista desplegable.
- **Components**: Contiene los módulos de interfaz de usuario, generalmente botones, definen la posición de la cámara y orientación del objeto en el espacio.

4.7. **Superficies y Edificios.**

Los módulos de estructuras y superficies son los que componen el mapa en su totalidad como se muestra en la figura 4.

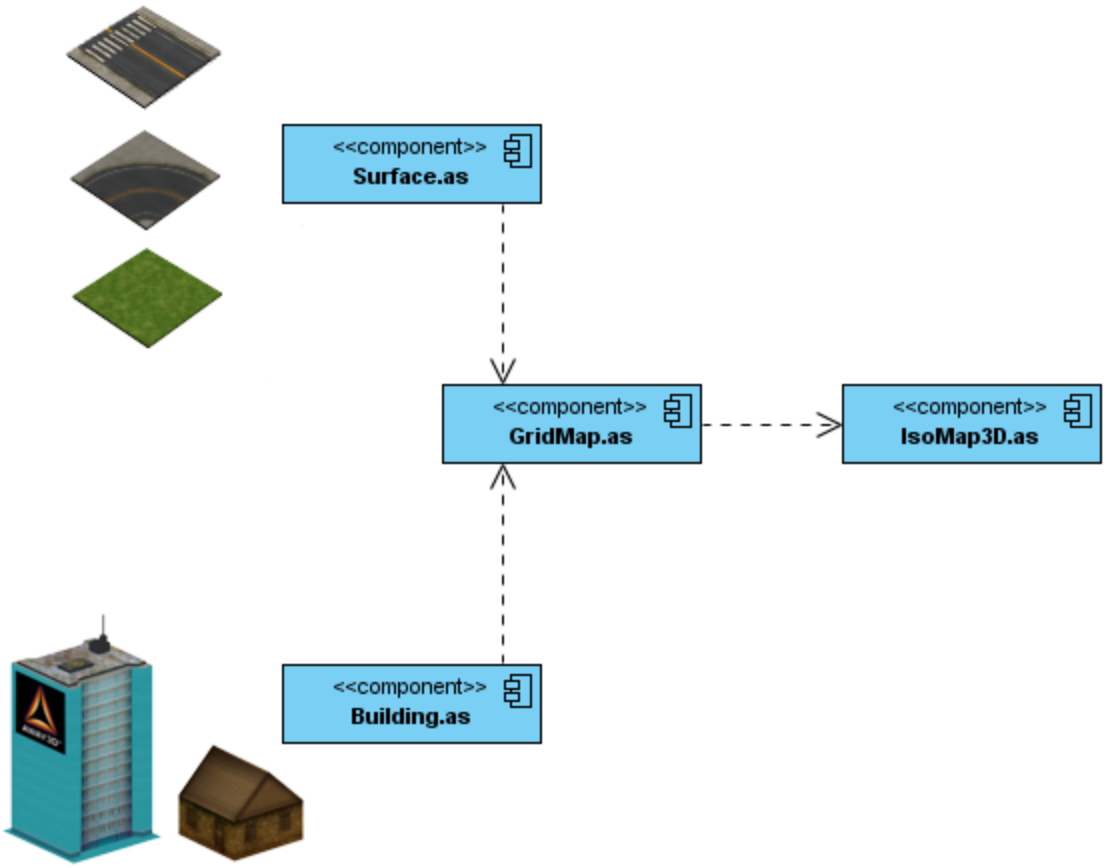

Figura 7. Diagrama de componentes. Superficies y Edificios.

- **Surfaces**: Módulo de carga de superficies, encargado de iniciar la carga de objetos 3D de tipo *surface* de la aplicación mientras se muestra una vista previa del mismo.

- **Buildings**: Módulo de carga de edificios, encargado de iniciar la carga de los objetos 3D de tipo *building* de la aplicación mientras se muestra una vista previa del mismo.

- **GridMap**: Módulo encargado de recrear los objetos 3D que son situados sobre el área de trabajo (Surfaces y Buildings) que en su totalidad conforman el mapa deseado.

4.8. **Implementación.**

La metodología XP propone comenzar la implementación de la solución partiendo de una arquitectura lo más flexible posible, con el propósito de que los desarrolladores puedan reestructurar el sistema sin cambiar su comportamiento y así remover duplicaciones de código, mejorar la comunicación, simplificar el código, o agregar flexibilidad. Debido a ello la solución tiene una arquitectura simple y muy bien definida.

4.8.1. **Primera iteración.**

El principal objetivo de la primera iteración es desarrollar la historia de usuario número uno (1): Crear elementos gráficos utilizando un núcleo de visualización de la aplicación. Esta iteración no finaliza con un producto visual capaz de ser probado, ya que su resultado es una librería gráfica en forma de código y servir de base para las próximas iteraciones.

Para ello se trazaron tres (3) tareas, que se describen a continuación:

- Tarea Nº 1: Estándar de código y de comentarios del núcleo.
- Tarea Nº 2: Manejo de los *tiles*[8]de la aplicación.
- Tarea Nº 3: Clase principal del núcleo: GridMap.

Tareas	
Número tarea: 1	**Número de HU**: 1
Nombre de la tarea: Estándar de código y de comentarios del núcleo.	
Tipo de tarea: Configuración.	**Estimación**: 3 días.

[8]Tiles: Pequeñas áreas cuadradas o mosaicos que componen una rejilla regular.

Fecha de inicio: 11 noviembre de 2009 **Fecha de fin**: 14 noviembre 2009.

Programador responsable: Jorge Lisandro Ruiz.

Descripción: Redactar el estándar de código siguiendo los estándares utilizados internacionalmente en proyectos de ActionScript 3.0. Consultar documentación en la red para la definición del mismo.

Tareas

Número tarea: 2 **Número de HU**: 1

Nombre de la tarea: Manejo de los *tiles* de la aplicación.

Tipo de tarea: Desarrollo. **Estimación**: 10 días.

Fecha de inicio: 15 noviembre de 2009 **Fecha de fin**: 25 de noviembre 2009

Programador responsable: Heriberto Martin.

Descripción: Inicio de la programación para la detección de objetos que serán añadidos al mapa en forma de *tiles*.

Tareas

Número tarea: 3 **Número de HU**: 1

Nombre de la tarea: Clase principal del núcleo: GridMap.

Tipo de tarea: Desarrollo	**Estimación**: 10 días.
Fecha de inicio: 25 de noviembre 2009	**Fecha de fin**: 5 de diciembre 2009.
Programador responsable: Heriberto Martin.	

Descripción: Implementar la clase principal del núcleo.

Paquete: cu.isomap3d.core

Clases: GridMap.

4.8.2. **Segunda iteración.**

La segunda iteración persigue el objetivo de obtener la primera versión de un producto que cumpla con la historia de usuario número dos (2): Ubicar objetos en el mapa. Una vez creada la base para la selección e inserción de elementos en el mapa el núcleo de la aplicación está preparado para insertar tantos objetos como sea necesario.

Las tareas necesarias para esta iteración son las siguientes.

- Tarea Nº 4: Interfaz de usuario Viewport, MainViewport y UIViewport.
- Tarea Nº 5: Selección de tiles del GridMap.
- Tarea Nº 6: Inserción de superficies.
- Tarea Nº 7: Inserción de edificios.

Tareas

Número tarea: 4	**Número de HU**: 2
Nombre de la tarea: Interfaz de usuario Viewport, MainViewport, UIViewport.	
Tipo de tarea: Desarrollo	**Estimación**: 5 días.
Fecha de inicio: 7 de Febrero 2010	**Fecha de fin**: 13 de Febrero 2010
Programador responsable: Jorge Lisandro Ruiz.	

Descripción: Diseño de la interfaz de usuario e implementación de los módulos de visualización.

Paquete: cu.isomap3d.core

Clases: MainViewport, UIViewport.

Módulo: Viewports.

Tareas

Número tarea: 5	**Número de HU**: 2
Nombre de la tarea: Selección de tiles del GridMap.	
Tipo de tarea: Desarrollo	**Estimación**: 2 días.
Fecha de inicio: 14 de Febrero 2010	**Fecha de fin**: 16 de Febrero 2010
Programador responsable: Heriberto Martin.	

Descripción: Implementación de las funcionalidades del proceso de selección de tiles del núcleo gráfico para emplazar objetos en el GridMap.

Paquete: cu.isomap3d.core

Clases: GridMap.

Tareas

Número tarea: 6	**Número de HU**: 2
Nombre de la tarea: Inserción de superficies.	
Tipo de tarea: Desarrollo	**Estimación**: 5 días.
Fecha de inicio: 17 de Febrero 2010	**Fecha de fin**: 22 de Febrero 2010
Programador responsable: Heriberto Martin.	

Descripción: Implementación de las funcionalidades del proceso de inserción de superficies del núcleo gráfico a emplazar en el GridMap, que posteriormente serán mostradas a través del Viewport principal de la aplicación.

Paquete: cu.isomap3d.core

Clases: Suface

Módulo: Surface

Tareas	
Número tarea: 7	**Número de HU**: 2
Nombre de la tarea: Inserción de edificios.	
Tipo de tarea: Desarrollo	**Estimación**: 2 días.
Fecha de inicio: 22 de Febrero 2010	**Fecha de fin**: 27 de Febrero 2010
Programador responsable: Heriberto Martin.	
Descripción: Implementación de las funcionalidades del proceso de inserción de edificios del núcleo gráfico que serán mostrados a través del Viewport principal una vez insertados en el GridMap. **Paquete**:cu.isomap3d.core **Clases**: Building. **Módulo**: Building	

4.8.3. **Tercera iteración.**

Esta iteración está encaminada a dar solución a las historias de usuario número tres (3) y cuatro (4): Edición de posición de objetos y Ajuste de visión de cámara respectivamente. Una vez concluidas las mismas el usuario podrá editar la posición de los objetos seleccionados así como ajustar el campo de visión de la cámara a través de controles de navegación en el sentido que desee.

Para ello se determinaron las siguientes tareas:

Historia de usuario número tres (3):

- Tarea Nº 8: Interfaz de usuario Object3DControl.
- Tarea Nº9: Rotar superficie seleccionada.
- Tarea Nº10: Rotar edificio seleccionado.

Historia de usuario número (4):

- Tarea Nº 11: Interfaz de usuario NavigationControls.
- Tarea Nº 12: Desplazar la cámara hacia arriba y hacia abajo.
- Tarea Nº 13: Desplazar la cámara hacia la derecha e izquierda.
- Tarea Nº 14: Centrar la cámara hacia el origen de coordenadas y ajuste de acercamiento.

Tareas	
Número tarea: 8	**Número de HU**: 3
Nombre de la tarea: Interfaz de usuario ObjectControl.	
Tipo de tarea: Desarrollo	**Estimación**: 2 días.
Fecha de inicio: 1 de Abril 2010	**Fecha de fin**: 3 de Abril 2010
Programador responsable: Jorge Lisandro Ruiz.	
Descripción: Diseño de la interfaz de usuario ObjectControl que permite manipular la orientación de los objetos seleccionados en el mapa.	

Paquete: cu.isomap3d.components

Clases: ObjectControl, ControlEvent

Módulo: Components.

Tareas

Número tarea: 9	**Número de HU**: 3
Nombre de la tarea: Rotar superficie seleccionada.	
Tipo de tarea: Desarrollo	**Estimación**: 5 días.
Fecha de inicio: 4 de Abril 2010.	**Fecha de fin**: 9 de Abril 2010.
Programador responsable: Heriberto Martin.	

Descripción: Implementación de las funcionalidades de rotación de una superficie seleccionada.

Paquete: cu.isomap3d.components, cu.isomap3d.core

Clases: Surface, ObjectControl

Módulo: Components, Surface

Tareas

Número tarea: 10	**Número de HU**: 3

Nombre de la tarea: Rotar construcción seleccionada.

Tipo de tarea: Desarrollo	**Estimación**: 5 días.
Fecha de inicio: 10 de Abril 2010.	**Fecha de fin**: 15 de Abril 2010.

Programador responsable: Heriberto Martin.

Descripción: Implementación de las funcionalidades de rotación a favor de las manecillas del reloj y viceversa de los objetos de tipo *building*.

Paquete: cu.isomap3d.components, cu.isomap3d.core

Clases: Building, ObjectControl

Módulo: Components, Building

Tareas

Número tarea: 11	**Número de HU**: 4

Nombre de la tarea: Interfaz de usuario NavigationControls.

Tipo de tarea: Desarrollo	**Estimación**: 3 días.
Fecha de inicio: 16 de Abril 2010.	**Fecha de fin**: 19 de Abril 2010.

Programador responsable: Jorge Lisandro Ruiz.

Descripción: Diseño de la interfaz de usuario para el control de navegación que permite el paneo de cámara mediante botones y a su vez

ajustar el acercamiento (zoom).

Paquete: cu.isomap3d.components

Clases: NavControls

Módulo: Components.

Tareas

Número tarea: 12	**Número de HU**: 4
Nombre de la tarea: Desplazar la cámara hacia arriba y hacia abajo.	
Tipo de tarea: Desarrollo	**Estimación**: 3 días.
Fecha de inicio: 20 de Abril 2010.	**Fecha de fin**: 23 de Abril 2010.
Programador responsable: Heriberto Martin.	

Descripción: Implementación de las funcionalidades para el movimiento de la cámara, cuyo objetivo es que el usuario pueda desplazarse hacia arriba y hacia abajo mientras construye el mapa deseado.

Paquete: cu.isomap3d.components, cu.isomap3d.core

Clases: NavControls, MainViewport

Módulo: Components, Viewports

Tareas

Número tarea: 13	**Número de HU**: 4
Nombre de la tarea: Desplazar la cámara hacia la derecha e izquierda	
Tipo de tarea: Desarrollo	**Estimación**: 2 días.
Fecha de inicio: 24 de Abril 2010.	**Fecha de fin**: 26 de Abril 2010
Programador responsable: Heriberto Martin.	

Descripción: Implementación de las funcionalidades de movimiento de la cámara con dirección a la derecha e izquierda.

Paquete: cu.isomap3d.components, cu.isomap3d.core

Clases: NavControls, MainViewport

Módulo: Components, Viewports

Tareas

Número tarea: 14	**Número de HU**: 4
Nombre de la tarea: Centrar la cámara hacia el origen de coordenadas y ajuste de acercamiento.	
Tipo de tarea: Desarrollo	**Estimación**: 5 días.
Fecha de inicio: 27 de Abril 2010	**Fecha de fin**: 2 de Mayo 2010.

Programador responsable: Heriberto Martin.

Descripción: Implementación de las funcionalidades que permiten centrar la cámara desde cualquier posición en que se encuentre y ajustar el acercamiento (zoom) para una mejor vista del mapa construido por el usuario.

Paquete: cu.isomap3d.components, cu.isomap3d.core

Clases: NavControls, MainViewport

Módulo: Components, Viewports

4.8.4. **Cuarta iteración.**

En esta última iteración está orientada a dar solución a las historias de usuario número cinco (5) y seis (6), las cuales se denominan Vista previa y Rotar mapa respectivamente.

Para ello se plantean las siguientes tareas ingenieriles:

Historia de usuario número 5:

- Tarea Nº 15: Selección del objeto deseado (Surface o Building).
- Tarea Nº 16: Mostrar vista previa del objeto seleccionado.

Historia de usuario número 6:

- Tarea Nº 17: Interfaz de usuario NavigationControls para la rotación.
- Tarea Nº 18: Rotación del mapa.

Tareas

Número tarea: 15	**Número de HU**: 5
Nombre de la tarea: Selección del objeto deseado (Surface o Building)	
Tipo de tarea: Desarrollo	**Estimación**: 1 día.
Fecha de inicio: 3 de Mayo 2010	**Fecha de fin**: 4 de Mayo 2010
Programador responsable: Heriberto Martin.	

Descripción: Implementación de la funcionalidad que le permita al usuario seleccionar un objeto del panel.

Paquete: cu.isomap3d.app

Clases: IsoMap3D

Módulo: IsoMap3D

Tareas

Número tarea: 16	**Número de HU**: 5
Nombre de la tarea: Mostrar vista previa del objeto seleccionado.	
Tipo de tarea: Desarrollo	**Estimación**: 3 días
Fecha de inicio: 5 de Mayo 2010	**Fecha de fin**: 8 de Mayo 2010
Programador responsable: Heriberto Martin.	

Descripción: Implementación de la funcionalidad mostrar vista previa del objeto una vez seleccionado.

Paquete: cu.isomap3d.core

Clases: PreviewViewport

Modulo: Viewports

Tareas

Número tarea: 17	**Número de HU**: 6
Nombre de la tarea: Interfaz de usuario NavigationControls para la rotación del mapa	
Tipo de tarea: Desarrollo	**Estimación**: 2 días.
Fecha de inicio: 9 de Mayo 2010	**Fecha de fin**: 11 de Mayo 2010
Programador responsable: Jorge Lisandro Ruiz.	

Descripción: Diseño de la interfaz de usuario NavigationControls para la rotación del mapa mediante botones.

Paquete: cu.isomap3d.components

Clases: NavControls

Módulo: Components.

Tareas

Número tarea: 18	**Número de HU**: 6
Nombre de la tarea: Rotación del mapa.	
Tipo de tarea: Desarrollo	**Estimación**: 5 días
Fecha de inicio: 12 de Mayo 2010	**Fecha de fin**: 17 de Mayo 2010
Programador responsable: Heriberto Martin.	
Descripción: Implementación de la funcionalidad para la rotación del mapa en sentido de las manecillas del reloj o viceversa en un ángulo de 45 grados. **Paquete**: cu.isomap3d.components, cu.isomap3d.core **Clases**: NavControls, MainViewport **Módulo:** Isomap3D.	

4.9. **Pruebas.**

La metodología XP cuenta con una característica conocida como TDD, desarrollo dirigido por pruebas (del inglés *Test Driven Development*).[MARCHESI and SUCCI 2003].TDD se enfoca en la implementación orientada a pruebas. El código debe ser probado paso a paso y obtener un resultado funcional. Puede confundirse con el término “pruebas de caja blanca” y, hasta cierto punto, es así: Las pruebas de caja blanca son realizadas a los métodos u operaciones para medir la funcionalidad del mismo tomando como indicador la validez para el cliente.[MARCHESI and SUCCI 2003]. TDD, por otro lado, se aplica antes de comenzar a implementar cada paso de la tarea en desarrollo asumiendo que la prueba es insatisfactoria desde un inicio y una vez que se haya cumplido de la forma más sencilla posible, la lógica del código

a probar, se asume como cumplida. Luego se realiza un proceso conocido informalmente como “refactorización” de código, el cual consiste en limpiarlo, organizarlo y adaptarlo a los patrones. En esencia, TDD se centra en la lógica del código y las pruebas de caja blanca en la del negocio. Las pruebas TDD también son conocidas como pruebas unitarias o de unidad.

ActionScript no cuenta con un marco de trabajo de pruebas TDD, por lo que es necesario utilizar el depurador de Flex Builder o la consola de salida en última instancia. Una vez terminada cada tarea, se prueba la funcionalidad de la misma, ya desde el punto de la validez de esta para el cliente. Esta prueba es realizada por parte del desarrollador. Luego se realizan las pruebas conocidas como de “caja gris”, en la cual se valida el comportamiento general de la solución en ambientes adversos como la pérdida de conectividad con los servicios de redes, pocos recursos de hardware y sobrecarga del mismo. Como paso definitorio el usuario final, realiza las pruebas de “caja negra” o aceptación.

4.9.1. **Pruebas de aceptación.**

Las pruebas de aceptación son realizadas por el propio cliente en compañía de uno de los representantes del equipo de desarrollo y se orientan a las funcionalidades del sistema. Su objetivo es comprobar, desde la perspectiva del usuario final, el cumplimiento de las especificaciones de la lista de reservas del producto.

A continuación, aparecen las pruebas de aceptación realizadas a la solución propuesta:

Caso de Prueba de Aceptación.	
Código: HU1_P1	**Historia de Usuario**: 1
Nombre: Crear un plano.	
Descripción: Prueba para la creación de un plano utilizando el núcleo	

gráfico de la aplicación.

Condiciones de Ejecución: El usuario debe tener un proyecto creado en Flex Builder.

Entrada/Pasos de ejecución: Se importan las clases del núcleo gráfico y las clases del motor para realizar planos, se agregan las dimensiones para crear el plano. Se crea un nuevo plano y se compila el código.

Resultado esperado: el plano es creado y mostrado sin errores.

Evaluación de la prueba: Resultado satisfactorio.

Caso de Prueba de Aceptación.

Código: HU1_P2 **Historia de Usuario**: 1

Nombre: Crear una superficie.

Descripción: Prueba crear una superficie.

Condiciones de Ejecución: el usuario debe tener un proyecto creado en Flex Builder.

Entrada/Pasos de ejecución: Se importan las clases del núcleo gráfico, se crea un plano con dimensiones de 10x10 y se le aplica una textura. Se crea una superficie. Luego se compila el código.

Resultado esperado: La superficie es creada y mostrada sin errores.

Evaluación de la prueba: Resultado Satisfactorio.

Caso de Prueba de Aceptación.

Código: HU1_P3 **Historia de Usuario**: 1

Nombre: Crear un GridMap.

Descripción: Prueba crear un GridMap.

Condiciones de Ejecución: El usuario debe tener creado un proyecto en Flex Builder.

Entrada/Pasos de ejecución: Se importan las clases del núcleo gráfico y se determinan una serie de puntos que formen una rejilla de dimensión de entre los espacios de 10x10. Se crea la rejilla y se compila en código.

Resultado esperado: Creación de un GridMap sin errores.

Evaluación de la prueba: Resultado Satisfactorio.

Caso de Prueba de Aceptación.

Código:HU2_P1 **Historia de Usuario**:2

Nombre: Insertar una superficie.

Descripción: Prueba para insertar una superficie.

Condiciones de Ejecución: –

Entrada/Pasos de ejecución: El usuario selecciona la superficie deseada y hace click sobre el área de trabajo para insertar la superficie.

Resultado esperado: Se inserta la superficie.

Evaluación de la prueba: Resultado Satisfactorio.

Caso de Prueba de Aceptación.

Código: HU2_P2 **Historia de Usuario**: 2

Nombre: Insertar un edificio.

Descripción: Prueba para insertar un edificio.

Condiciones de Ejecución: –

Entrada/Pasos de ejecución: El usuario selecciona el edificio deseado y hace click sobre el área de trabajo para insertar el edificio.

Resultado esperado: Se inserta el edificio.

Evaluación de la prueba: Resultado Satisfactorio.

Caso de Prueba de Aceptación.

Código: HU3_P1 **Historia de Usuario**: 3

Nombre: Editar la posición de un edificio.

Descripción: Prueba cambiar la orientación de un edificio en el mismo lugar.

Condiciones de Ejecución: El usuario debe tener insertado un edificio.

Entrada/Pasos de ejecución: El usuario selecciona el edificio deseado y hace click derecho sobre el objeto, mediante el control de objetos rota en sentido de las manecillas del reloj o viceversa el edificio seleccionado

Resultado esperado: La orientación del edificio cambia.

Evaluación de la prueba: Resultado Satisfactorio.

Caso de Prueba de Aceptación.

Código: HU4_P1 **Historia de Usuario**: 4

Nombre: Mover la cámara.

Descripción: Prueba para mover la cámara hacia arriba, abajo, derecha, izquierda y centrar.

Condiciones de Ejecución: –

Entrada/Pasos de ejecución: El usuario mediante el control de navegación hace click sobre el botón que indica la dirección deseada. Posteriormente la cámara vuelve a su posición original haciendo uso del botón central del panel.

Resultado esperado: La cámara se mueve en la dirección deseada.

Evaluación de la prueba: Resultado Satisfactorio.

Caso de Prueba de Aceptación.

Código: HU5_P1	**Historia de Usuario**: 5

Nombre: Vista previa de un edificio.

Descripción: Prueba para mostrar la vista previa de un edificio.

Condiciones de Ejecución: –

Entrada/Pasos de ejecución: El usuario selecciona un edificio para ver sus características.

Resultado esperado: Se muestra el edificio en el panel de vista previa.

Evaluación de la prueba: Resultado satisfactorio.

Caso de Prueba de Aceptación.

Código: HU6_P1	**Historia de Usuario**: 6

Nombre: Rotar mapa.

Descripción: Prueba para rotar un mapa construido.

Condiciones de Ejecución: El usuario debe tener un mapa construido.

Entrada/Pasos de ejecución: El usuario mediante el control de navegación puede rotar el mapa hacia la derecha o a la izquierda. El usuario hace click en el botón de rotación.

Resultado esperado: el mapa rota hacia la derecha o a hacia la izquierda.

Evaluación de la prueba: Resultado satisfactorio.

4.10. **Consideraciones finales.**

En el presente capítulo se abordaron los temas referentes a la implementación de la solución y la estrategia de pruebas a seguir durante la elaboración del mismo. Se presentó el diseño de la arquitectura y las tareas que se llevaron a cabo para construirla.

Las pruebas realizadas a la solución se terminaron en el tiempo acordado dando un resultado satisfactorio, lo que demuestra que la efectividad del desarrollo dirigido por pruebas juega un papel fundamental en el proceso de desarrollo de software con una metodología ágil.

Conclusiones

La flexibilidad de la aplicación proporciona una base robusta para la construcción de jugos en línea y un ambiente de desarrollo extensible. Con el uso de la metodología XP combinado con los aportes arquitectónicos que provee UML, la programación en parejas y las entregas a corto plazo, se obtuvo una mayor eficiencia del equipo de desarrollo y a su vez un producto funcional.

El lenguaje AS3 y el SDK libre de Flex integrado con Away3D propiciaron crear elementos interactivos de una manera rápida, eficiente y con un grado de robustez elevado, demostrando la posibilidad de visualizar en los navegadores elementos tridimensionales en tiempo real.

Las pruebas de caja negra demostraron que IsoMap3D permite la creación y visualización de mapas isométricos tridimensionales, garantizando un nivel elevado de interactividad para su utilización en el desarrollo de juegos en línea de una manera rápida y eficiente.

Recomendaciones

Los autores de la presente investigación recomiendan:

- Optimizar el rendimiento y los procesos que gestiona la aplicación.
- Añadir nuevas funcionalidades para la visualización de las propiedades y características de las edificaciones y superficies.
- La incorporación de nuevos edificios y tipologías de superficies a las categorías de *building*, *surface* y *others* de la lista desplegable, así como funcionalidades que permitan modificar la textura de un objeto dado.

Glosario de términos

Adobe System Incorporated: Empresa norteamericana de software con sede en California. Fue fundada en diciembre de 1982 por John Warnock y Charles Geschke. Destaca en el mundo del software por sus programas de edición de páginas web, vídeo e imagen digital.

Marco de trabajo: Un marco de trabajo, en el desarrollo de software, es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas de código y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

IDE: Un Ambiente o Entorno de Desarrollo Integrado ("Integrated Development Enviroment" por sus siglas en inglés) es una herramienta que integra un conjunto de aplicaciones para crear software u otro tipo de contenidos digitales, ya sea imágenes, audios, videos, etc. En el desarrollo de software su principal objetivo es automatizar tareas, empaquetar, compilar, ejecutar y generalmente depurar el código, por mencionar algunos ejemplos. Por otro lado, en los IDEs para la creación de contenidos visuales se ofrece una serie de herramientas de dibujo, trazado, edición de propiedades, edición de color y otros.

On-line: Concepto de no más de 20 años de aparición que significa "en línea". Es una categoría que se le da a las aplicaciones o procesos que se ejecutan a partir del uso de una computadora conectada a la red de redes.

RIA: Aplicaciones Enriquecidas de Internet (del inglés "Rich Internet Applications") 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 tradicionales.

UML: El Lenguaje Unificado de Modelado (“Unified Modeling Language” por sus siglas en inglés) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Aplicación en la nube: Según el IEEE Computer Society, es un paradigma en el que la información se almacena de manera permanente en servidores en Internet y se envía a cachés temporales de cliente, lo que incluye equipos de escritorio, centros de ocio, portátiles, entre otros. Cada día, más empresas apuestan por la famosa nube virtual, que permite acceder a multitud de servicios y aplicaciones en Internet sin necesidad de tener el software instalado en el ordenador, es decir, los datos y archivos de los usuarios no estén en sus equipos ni dependan de su sistema operativo, sino que permanezcan almacenados en servidores y centros de datos accesibles en línea.

Web 2.0: La Web 2.0 es la transición que se ha dado de aplicaciones tradicionales hacia aplicaciones que funcionan a través de la web enfocada al usuario final. Se trata de aplicaciones que generen colaboración y de servicios que reemplacen las aplicaciones de escritorio.

Web 3.0: Este concepto es utilizado para describir la evolución del uso y la interacción en la red a través de diferentes caminos. Ello incluye: la transformación de la red en una base de datos, un movimiento para hacer que los contenidos sean accesibles por múltiples aplicaciones, el empuje de las tecnologías de inteligencia artificial, la web semántica, la web geoespacial, o la web 3D; mejorar la interoperabilidad entre los sistemas informáticos usando agentes inteligentes, que no son más que programas en las computadoras que buscan información sin operadores humanos.

Referencias Bibliográficas

ACTIONSCRIPT. *ActionScript* 2010. [Febrero 9, 2010]. Disponible en: http://www.actionscript.org/resources/categories/Articles/Product-Reviews/

AIR. *Adobe Air*, 2008. [Disponible en: www.adobe.com/es/products/air/

ALTERNATIVA3D. *Alternativa Platform*, Alternativa3D, 2009. [Enero 19, 2010]. Disponible en: http://alternativaplatform.com/en/about/

AWAY3D. *Away3D Flash Engine*, 2010. [Enero 21, 2010]. Disponible en: http:www.away3d.com/documentation

BUILDER, A. F. *Adobe Flash Builder 4*, Adobe, 2010. [Disponible en: http://www.adobe.com/products/flashbuilder/

BUNTEL, T., Adobe Labs, 2010. [Abril 2, 2010]. Disponible en: www.adobe.com/technologies/product/flasbuilder

CARLBORN, I. and J. PACIOREK. Proyecciones geométricas planas. en: *Las proyecciones geométricas planas y transformaciones de visualización.* ACM, 1978. 10.p.

FISHER, D. *swf2xml and xml2swf*, SWFMill, 2008. [Enero 8, 2010]. Disponible en: http://swfmill.org/

FLEX, A. *Adobe Flex*, Adobe, 2009. [Diciembre 8, 2010]. Disponible en: http://www.adobe.com/products/flex/

GROSSMAN, H. *ActionScript 3.0 overview*, Adobe, 2006. [Febrero 10, 2010]. Disponible en: http://www.adobe.com/devnet/actionscript/articles/actionscript3overview.html

GUTIÉRREZ, C. *WebGL: 3D para tu browser*, FayerWayer, 2010. [Enero 19, 2010]. Disponible en: http://www.fayerwayer.com/2009/09/webgl-3d-para-tu-browser/

HOYTECNOLOGÍA. *Aplicaciones en la nube 2010.*, 2010. [Mayo 5, 2010]. Disponible en: http://www.hoytecnologia.com/noticias/Diez-aplicaciones-nube-para/148666

INCORPORATED, A. S. *Adobe Flash Player*, Adobe, 2009a. [Deciembre 6, 2009]. Disponible en: http://www.adobe.com/products/flash/about/

---. *Adobe Open Source-Confluence*, 2009b. [Febrero 11, 2010]. Disponible en: http://opensource.adobe.com/wiki/display/site/Home

KRASNOSHCHOK, A. *Flex Implementation of MVC*, 2009. [Marzo 13, 2010]. Disponible en: http://www.slideshare.net/Antonos/mvc-pattern-flex-implementation-of-mvc-

KRUCHTEN, P. *A Software Development Process for a Team of One*. 2002. p.

LARMAN, C. Análisis y diseño. en: *UML y Patrones. Introducción al análisis y diseño a objetos*. Editorial Felix Varela, 2004a. 2:191.p.

---. *UML y Patrones. Introducción al análisis y diseño a objetos*. Editorial Felix Varela, 2004b. p.

LEONARD and BRUTZMAN. *X3D: Extensible 3D Graphics Standard*, 2008. [Febrero 21, 2010]. Disponible en: http://x3dgraphics.com/readmore

MARCHESI, M. and G. SUCCI. *Extreme Programming and Agile Procces in Software Engineering*. 4ta Conferencia Internacional, Genova, 2003. 68 p.

OGRE. *Ogre Engine*, Ogre3D, 2009. [Noviembre 3, 2009]. Disponible en: http://www.ogre3d.org/about

OPENSPACE. *OpenSpace Overview*, OpenSpace, 2010. [Febrero 24, 2010]. Disponible en: http://www.openspace-engine.com/overview

PALMU, M. *FlashDevelop Open Source Flash*, 2008. [Febrero 27, 2010]. Disponible en: http://osflash.org/flashdevelop

PARADIGM, V. *UML CASE Tools - Free for Learning UML, Cost-Effective for Business Solutions*, 2008. [Diciembre 6, 2009]. Disponible en: http://www.visual-paradigm.com/product/vpuml/

SANDY3D. *Sandy3D Overview*, Sandy3D, 2010. [Marzo 5, 2009]. Disponible en: http//www.sandy3d.com/about/

SCOTT, A. *XP and the UML? Clearly the Wrong Question to be Asking* 2007. [Marzo 17, 2010]. Disponible en: http://www.agilemodeling.com/essays/xpUML.htm

SDK, F. *Adobe Flex SDK*, 2010. [Mayo 26, 2010]. Disponible en: http://opensource.adobe.com/wiki/display/flexsdk/Flex+SDK

SHORE and WARDEN. *The Art of Agile Development* Primera Edición. 2007. p.

SIERRA, D. *Visual Paradigm For UML*, 2007. [Mayo 8, 2010]. Disponible en: http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml

SMARTFOXSERVER. *What is SmartFoxServer?*, SmartFoxServer, 2010. [Abril 27, 2010]. Disponible en: http://www.smartfoxserver.com/overview.php

TWEEN, M. *Motion Tween*, Motion Tween, 2008. [Junio 3, 2008]. Disponible en: http://www.mtasc.org/