



**UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN**

**DISEÑO E IMPLEMENTACIÓN DE UN VISUALIZADOR DE EQUIPOS GPS EN MAPAS
DE 3 DIMENSIONES UTILIZANDO SISTEMAS DE INFORMACIÓN GEOGRÁFICA**

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

MARCELO ALEJANDRO MUÑOZ VERGARA

**PROFESOR GUÍA:
NANCY VIOLA HITSCHFELD KAHLER**

**MIEMBROS DE LA COMISIÓN:
BENJAMIN EUGENIO BUSTOS CARDENAS
MAX ISAKSON EILER**

**SANTIAGO DE CHILE
MARZO 2008**

Resumen

El objetivo general del presente trabajo de título es diseñar e implementar un visualizador de equipos GPS, que realice el rendering de un mapa (superficie) de 3 dimensiones utilizando un Sistema de Información Geográfica (GIS) como fuente de datos. Esta herramienta debe simplificar y optimizar la interacción entre el usuario y el servidor.

Al momento de realizar este trabajo, no existían muchas aplicaciones que representen la información geográfica en 3 dimensiones, sin embargo se está presenciando un constante crecimiento de los GIS 3D y su interacción con los CAD (herramientas de diseño asistido por computador). La realidad virtual y la computación gráfica, los influyen y acerca unos a otros cada vez más. En este contexto, los modelos 3D de superficies pueden asistir en forma interactiva los procesos de toma de decisiones, como en este caso, el manejo de una flota equipada con equipos GPS.

El desarrollo se comenzó determinando las herramientas que pudieran agilizar el trabajo y los componentes que formaron parte de la aplicación. Primero se seleccionó un Sistema de Información Geográfico (Servidor GIS) que permite la obtención de datos geográficos mediante consultas simples. Luego se diseñó un componente proxy que permite la interacción entre el Servidor GIS y la interfaz gráfica, que procesa las distintas capas de información en forma independiente y optimiza el manejo de datos para que el posterior rendering de superficie sea más liviano. A continuación se diseñó la interfaz Web en 3 dimensiones con Adobe Flash utilizando librerías de código abierto para generar la representación virtual de los objetos. Finalmente se integra un sistema externo que provee la información de posicionamiento de los móviles GPS dentro de la aplicación.

El resultado final fue una aplicación desarrollada como prototipo inicial de un visualizador portable que realiza rendering en 3 dimensiones de superficies de manera óptima permitiendo una navegación intuitiva utilizando el mouse y teclado. Se integró, también, exitosamente un sistema de información GPS logrando representar a los móviles posicionados sobre un mapa.

El trabajo realizado cumplió con los objetivos propuestos en esta memoria. El prototipo desarrollado es el punto de partida para el desarrollo de una aplicación de mayores proporciones que incluirá funcionalidades tales como mejoramiento de la interfaz, integración de cartografía nueva, mejoras en la utilización de la información cartográfica, implementación de algoritmos complementarios de optimización de datos, entre otros.

Agradecimientos

A la profesora Nancy Hitschfeld por ayudarme a alcanzar esta meta tan ansiada y comprenderme.

A Tastets System por creer en mi trabajo y permitirme desarrollarlo sin cuestionamientos.

A Dios.

Índice

1.	Introducción.....	6
1.1.	Antecedentes Generales.....	6
1.2.	Justificación	8
1.3.	Objetivos generales	10
1.4.	Objetivos específicos	10
1.5.	Contenido de la memoria	10
2.	Antecedentes	12
2.1.	Servidor de Mapas.....	12
2.2.	Flash	14
2.2.1.	Usando ActionScript 3 (AS3)	15
2.2.2.	E4X.....	16
2.2.3.	Librerías 3D para Flash	17
2.2.4.	La alternativa gratuita, SVG.....	19
2.3.	Dispositivos GPS	20
2.4.	Trabajos Relacionados	22
2.5.	Requerimientos de la aplicación y solución propuesta.....	25
3.	Análisis y Diseño.....	26
3.1.	Arquitectura.....	26
3.1.1.	Servidor GIS	26
3.1.2.	Servidor Proxy de Información.....	27
3.1.3.	Interfaz 3D Flash.....	27
3.1.4.	Integración con sistemas GPS.....	28
3.2.	Diagrama de Clases.....	28
3.2.1.	Interfaz 3D	28
3.2.2.	Servidor Proxy de Información.....	32
3.3.	Manejo de Escalas y Peticiones de Información Geográfica	33
4.	Implementación.....	35
4.1.	Ambiente de Trabajo.....	35
4.1.1.	Eclipse	35
4.1.2.	Flex 2 SDK.....	36
4.1.3.	FlashDevelop 3.0 Beta2.....	36
4.1.4.	SVN	36
4.2.	Servidor GIS	37
4.2.1.	Instalación de Geoserver	37
4.2.2.	Carga de información (Shapefiles).....	37
4.2.3.	Estudio de Features.....	38
4.2.4.	Consultas WFS.....	40
4.2.5.	Test de Conexión a Servidor GIS	41
4.3.	Servidor Proxy de Información.....	42
4.3.1.	Procesamiento por capas	43

4.3.2.	Conversión de geometría espacial a píxeles	44
4.3.3.	Generalización de puntos	44
4.4.	Interfaz 3D	48
4.4.1.	Implementación de controles básicos	48
4.4.2.	Implementación de llamadas a servidor Proxy manejando proyecciones y escalas	51
4.5.	Integración del Sistema GPS.	52
4.5.1.	Registrar sesión de usuario	52
4.5.2.	Actualización de posiciones	52
5.	Discusión y conclusiones	54
5.1.	Resultados	54
5.2.	Conclusiones	54
5.3.	Trabajos futuros	56
6.	Bibliografía	57

1. Introducción

Hace un par de años atrás, los Sistemas de Información Geográfica (GIS) experimentaron un gran auge, pues comenzaron a ser utilizados por gobiernos centrales y locales, corporaciones de servicio público, sistemas vehiculares de navegación, estudios de estrategia publicitaria, etc. En conjunto, el aumento de la potencia del hardware, la disminución de costos de los servidores y la aparición de aplicaciones de código abierto han logrado que los GIS se masifiquen. Una de las muestras más populares de su uso es Google Earth.

Si a lo anterior agregamos la necesidad de las empresas de conocer en todo momento la ubicación de sus móviles, ya sea por seguridad del chofer, evitar desvíos, prever atrasos, ahorrar recursos, optimizar rutas entre otros, tenemos los requerimientos necesario para iniciar este proyecto.

Además, podemos agregarle valor a esta aplicación si se crea en 3 dimensiones, que permite una manipulación más natural del sistema y mejorar en la toma de decisiones y análisis, además al ser Web (Flash) es posible accederla desde cualquier parte.

1.1. Antecedentes Generales

Un Sistema de Información Geográfica es un sistema que es utilizado para ingresar, almacenar, recuperar, manipular, analizar y obtener datos referenciados geográficamente o datos Geoespaciales, a fin de brindar apoyo en la toma de decisiones sobre planificación y manejo del uso del suelo, recursos naturales, medio ambiente, transporte, instalaciones urbanas y otros registros administrativos. Esta memoria utilizará, en especial, la capacidad de manejar información de calles para visualización de rutas, posicionamiento de geodatos y visualización de equipos GPS.

Un GIS posee componentes claves que pueden dividirse en:

- Los sistemas computacionales, que proveen la capacidad de almacenar, manejar, capturar, analizar, modelar y **visualizar** los datos, a través de procedimientos de software y hardware.
- La interfaz de configuración que permite seleccionar la información, fijar los estándares, diseñar esquemas de actualización eficientes y analizar las “salidas” de los sistemas existentes.

- Datos obtenidos a partir de mapas digitalizados, fotografías aéreas, imágenes de origen satelital, tablas estadísticas y documentos con información geográfica. Estos datos pueden clasificarse en datos gráficos y en atributos (datos temáticos) por zona. Los datos gráficos pueden almacenarse como datos vectoriales o matriciales (raster).

En resumen, un GIS ayuda a mantener los datos geospaciales, mapas y estadísticas actualizadas y consistentes, facilitando la toma de decisiones, y como están basados en estándares, permiten que su información sea compartida en un contexto espacial.

Un GIS necesita fuentes de información, una muy difundida y actual es la que proveen los equipos GPS.

El GPS o Sistema de Posicionamiento Global es un Sistema de Navegación basado en Satélites (GNSS) el cual permite establecer la posición de un objeto en el mundo, con una precisión variable dependiendo del equipo utilizado que va desde unos pocos metros hasta centímetros usando GPS diferencial.

El GPS funciona mediante una red de satélites en órbita sobre el globo que permite al equipo calcular la posición en que éste se encuentra por medio de una triangulación entre los satélites, es decir, se basa en determinar la distancia de cada satélite respecto al punto de medición. Conocidas las distancias, se determina fácilmente la propia posición relativa respecto a los tres satélites. Conociendo además las coordenadas o posición de cada uno de ellos por la señal que emiten, se obtiene la posición absoluta o coordenadas reales del punto de medición. Esta información combinada con la de un GIS puede generar mapeos de superficies de gran precisión o la posición en el mundo casi exacta de una persona, o un vehículo entre otros.

Dentro de las empresas que ofrecen servicios GPS, existen las dedicadas a controlar vehículos, que ofrecen una gestión más simple sobre una flota, permitiendo el control sobre la ubicación, rutas y costos. Estos datos relacionados con un GIS aportan una poderosa herramienta de regulación del trabajo. Para esta memoria se utilizará la información de posicionamiento.

Por ahora el almacenamiento y procesamiento de información está bastante resuelto, así como las fuentes que generen datos, pero existe una carencia... la visualización. ¿Porqué tener mapas e imágenes estáticas en 2 dimensiones cuando los datos son de forma vectorial, es decir en 3 dimensiones? ¿Porqué restringirse a 2 dimensiones, si la 3ra dimensión es cada vez más utilizada para agilizar el análisis y toma de decisiones?

He aquí la motivación de este trabajo de memoria.

1.2. Justificación

El presente trabajo de memoria pretende crear un visualizador en 3 dimensiones que permita la representación dinámica de mapas en un contexto espacial (más natural) de la información y la fácil interacción entre los datos del servidor y el usuario.

Una visualización en 3D facilita el análisis y manejo de información geoespacial (mapas, imágenes, rutas, entre otros), y aunque el ser humano piense en 2 dimensiones, el concepto de volumen (3D) es fácil de adquirir.

Estamos presenciando un constante crecimiento de los GIS 3D y su interacción con el CAD. La realidad virtual y la computación gráfica, los influyen y acerca unos a otros cada vez más. En este contexto, los modelos urbanos 3D pueden asistir en forma interactiva en cualquiera de las etapas de los procesos de toma de decisiones. De esta forma, tablas o atributos pueden atarse a datos gráficos 3D. Los GIS añaden a estos modelos urbanos 3D los beneficios de los atributos más todas las posibilidades del análisis espacial. Consultas de cualquier tipo, links a hipertextos, documentos sobre normativa y planeamiento enriquecen la información ofrecida por este “mundo virtual”. Por último, considerando aspectos relacionados a la comunicación, el diseño interactivo y las tomas de decisiones se ven facilitadas mejor en un ambiente 3D que 2D. [14][15]

Actualmente la mayor parte de la información almacenada en los GIS se representa a través de un mapa o una imagen que contiene los datos, calles, ríos, información poblacional, entre otros. Estos datos son relacionados con información del usuario [5]. Por ejemplo, las empresas proveedoras de servicios GPS despliegan la ubicación de un equipo a través de un símbolo dibujado sobre la calle en que se encuentra por medio de una imagen generada por un el servidor. Pero, ¿qué ocurre cuando se requiere un mayor detalle de la ubicación? Si la aplicación lo permite, se realiza un acercamiento. El problema con el acercamiento, es que realmente es una segunda llamada al servidor pidiendo la generación de una segunda imagen. Si la necesidad cambia, y se necesita desplazar el mapa el procedimiento es el mismo, el servidor genera una nueva imagen con el contenido solicitado. Este proceso se vuelve un poco lento y además utiliza más recursos de servidor y red de lo necesitado. Al tener una aplicación que trabaje con los datos (no con imágenes) y que presente un ambiente que permita la manipulación simple, la interacción con el mapa se vuelve más rápida y permite una mayor comprensión del entorno de un objeto en estudio dentro de un mapa [1].

Otro problema que existe actualmente cuando se desea actualizar un dato del mapa, ya sea el nombre de una calle, la ubicación de un edificio, la profundidad de un río, etc. Las herramientas que trabajan con imágenes separan, la selección de datos a actualizar, de la interfaz de ingreso de datos, lo que hace el proceso muy complejo tanto para el usuario como para el servidor. Con una vista 3D de un

modelo virtual la tarea se convertiría en sólo seleccionar el objeto a corregir y modificarlo.

Como la cantidad de información a manejar es considerable, la aparición de filtros ayudaría con la selección de la información necesaria. Por ejemplo, si un camión que posee un GPS estuviese ubicando dentro de la ciudad, tal vez sólo sería interesante para mi negocio conocer las calles del área en que está posicionado y no la información fluvial. Si el mapa fuese una imagen, esta operación (ocultar una capa) involucra la generación de una nueva imagen, en cambio en 3 dimensiones, sólo sería ocultar una capa de la información que se obtuvo con anterioridad.

Además de filtros, es posible aplicar técnicas de generalización. Esta consiste en reducir el detalle o simplificar la realidad. Un mapa no puede mostrar todos los detalles y si uno pudiera serían muy difíciles de interpretar. Las fotos aéreas o imágenes satelitales son clara muestra de esto. Principalmente esta técnica se basa en tres procesos; simplificación, selección y, clasificación y simbolización. La simplificación determina el atributo del volumen, dividiendo los objetos en volúmenes (3D), áreas (2D), líneas (1D), y puntos (0D), así como la escala disminuye, los objetos reducen progresivamente su dimensión. Esto se denomina "colapsado". La selección se basa en *agregado*, fusión de varios elementos, mayormente común con áreas; *eliminación*, que consiste en remover elementos como puntos y líneas; y *suavizado*, que es la remoción de detalles en la forma o el delineado en líneas y áreas. El proceso de clasificación y simbolización se basa en la agrupación de datos, nominal o por nombre, ordinal o jerarquía del objeto, e intervalo, es decir cantidad o tamaño. Una vez agrupados es posible aplicarle alguna técnica en especial.

Como solución natural a los problemas antes mencionados surge como respuesta Google Earth, pero esta aplicación no esta orientada en gran manera a los datos geoespaciales, más bien lo hace a imágenes satelitales. Por otra parte no es posible una modificación en su implementación para permitir cruce de información de un sistema GIS con datos del usuario tan fácilmente.

La herramienta escogida para implementar la aplicación es Flash, aunque este no soporta imágenes en 3 dimensiones. Entonces, ¿Porqué elegirlo para el desarrollo? Flash soporta en forma nativa el dibujo vectorial en 2 dimensiones, lo que puede extrapolarse fácilmente a 3 dimensiones por medio de librerías y así aprovechar la capacidad manipular información vectorial y matricial de los GIS, aunque es necesario realizar optimizaciones con técnicas gráficas.

Flash consume pocos recursos y no necesita de máquinas virtuales ni instalaciones adicionales y funciona en la mayoría de los Browser. Es portable y su comportamiento es independiente de la plataforma, además logra un estilo visual único.

Se puede agregar que un futuro sería posible portar la aplicación a Flash Lite, que es una característica de los nuevos equipos portátiles, en especial de celulares.

1.3. Objetivos generales

Diseñar e implementar un visualizador de equipos GPS, que realice un rendering de un mapa (superficie) de 3 dimensiones utilizando un Sistema de Información Geográfica como fuente de datos. Esta herramienta debe simplificar y optimizar la interacción entre el usuario y el servidor. Los objetivos específicos que se desprenden de este objetivo general son los siguientes:

1.4. Objetivos específicos

- Implementar un entorno que realice rendering de superficies en 3 dimensiones en cualquier Web Browser del cliente utilizando Flash. En particular, debe implementar optimizaciones gráficas para la representación del mapa. Debe permitir el manejo del modelo a través de funcionalidades de desplazamiento (panning), zoom, rotación y traslación, entre otras funcionalidades.
- Utilizar la información vectorial y matricial del servidor GIS para generar el contenido visual de las superficies y objetos dentro del proceso de rendering.
- Utilizar la información proporcionada por servicios de control de flota de vehículos por medio de equipos GPS. Relacionar estos datos con la información del GIS y visualizarlos.
- Realizar optimizaciones de generalización, es decir implementar la capacidad de reducir la cantidad de detalles característicos de los objetos en un mapa o presentación cartográfica, con el fin de hacer el modelo final menos complejo.

1.5. Contenido de la memoria

En la siguiente memoria se describirán los pasos que se han seguido y las decisiones que se han tomado para la construcción inicial de este visualizador en 3 dimensiones.

Inicialmente se describen las tecnologías utilizadas en la construcción de la aplicación, las decisiones tomadas para optar por ellas y el rol que toman en la arquitectura.

En el siguiente capítulo se detalla cómo se utilizan los componentes antes descritos, para levantar la arquitectura básica que permitirá su interacción, a través de XML como medio principal de comunicación entre el servidor de mapas, el servidor Proxy que realiza la conversión de la información geográfica a coordenadas espaciales del visualizador Flash y los Dispositivos GPS. Dentro del mismo capítulo, se muestran los diagramas de clase asociados a los componentes desarrollados.

El cuarto capítulo describe la parte práctica del trabajo, es decir, mostrar las herramientas utilizadas, y como se implementaron los componentes de la arquitectura antes propuesta. Por otro lado, se muestra como se realizó la integración con el sistema externo que maneja los dispositivos GPS para obtener la información.

Finalmente se presentan los resultados obtenidos de la representación final del trabajo. Se muestran también las conclusiones obtenidas y los trabajos que quedan pendientes, ya que este es el inicio de un desarrollo mayor, pero genera las bases y marca las directrices a seguir.

2. Antecedentes

En el siguiente capítulo se detallaran las características principales de los componentes utilizados para desarrollar la aplicación y las razones de porque se prefirieron frente a otras alternativas.

2.1. Servidor de Mapas

En general un GIS tiene la capacidad de dar respuesta a las siguientes preguntas:

¿Dónde está el objeto A?

¿Dónde está A con relación a B?

¿Cuántas ocurrencias del tipo A hay en una distancia D de B?

¿Cuál es el valor que toma la función Z en la posición X?

¿Cuál es la dimensión de B (Frecuencia, perímetro, área, volumen)?

¿Cuál es el resultado de la intersección de diferentes tipos de información?

¿Cuál es el camino más corto (menor resistencia o menor costo) sobre el terreno desde un punto (X1, Y1) a lo largo de un corredor P hasta un punto (X2, Y2)?

¿Qué hay en el punto (X, Y)?

¿Qué objetos están próximos a aquellos objetos que tienen una combinación de características?

¿Cuál es el resultado de clasificar los siguientes conjuntos de información espacial?

El modelo de datos de un GIS está basado en estructuras que representan un conjunto de pautas para convertir el mundo real en objetos espaciales representados por objetos lógicos, que consisten en atributos temáticos o semánticos y geometría topológica [1].

Existen dos modelos de datos:

- Modelo basado en vectores. Este utiliza puntos, líneas o áreas discretas que corresponden a objetos discretos con nombre o número de código de atributos.

La ventaja con respecto al modelo raster es precisión, menor volumen de datos, topología completa, rápida conversión y recuperación.

Como desventaja vemos una estructura complicada, con dificultades de superposición, actualización e ingreso de datos.

- Modelo basado en matrices (raster). Este utiliza celdas de una grilla espaciada regularmente en una secuencia específica. Cada celda se denomina "píxel".

Sus ventajas son una estructura simple, de fácil superposición y modelado, fácil integración con imágenes.

Sus desventajas con el gran volumen de datos que genera, baja precisión dificultad para el análisis y lenta conversión.

Independientemente del modelo elegido, las entidades del mundo real se representan en objetos con atributos asociados correspondientes a datos temáticos. A partir de estos datos, los objetos pueden ser separados por capas como se observa en la Figura 1 [6].

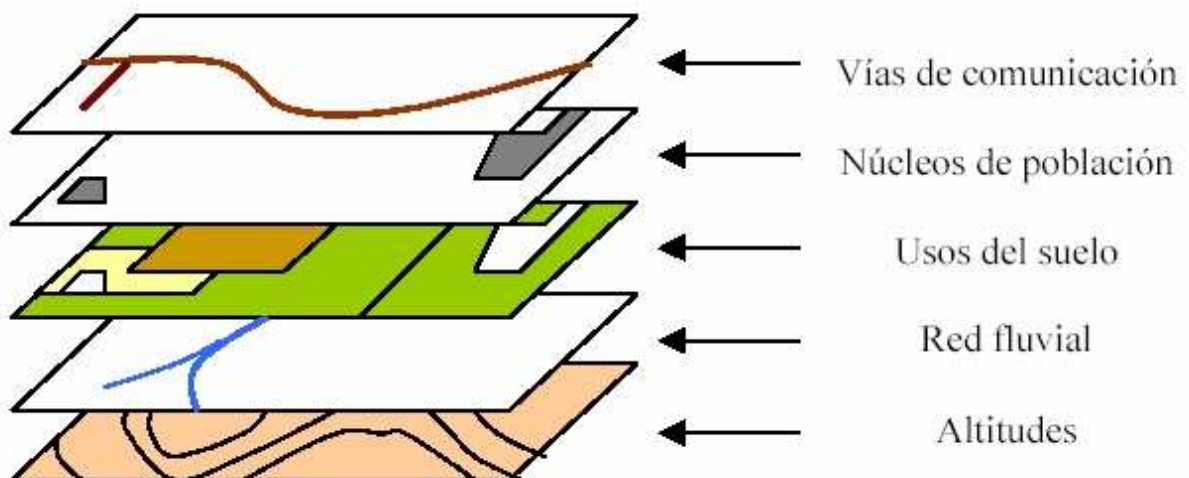


Figura 1

Estos modelos de datos se han generalizado bastante, apareciendo soluciones que los implementan como PostGis, una extensión a Postgresql el cual posee tipos y primitivas geométricos que permiten manejar como un elemento más de la base de datos, información georreferenciada asociada a la relacional. Como complemento al modelo, para la base de datos se ha desarrollado una serie de scripts que permiten exportar esta información en otros formatos. Esta combinación constituye por sí misma un GIS completo. Existen alternativas como MapServer (escrito en C++) o el popular GeoServer (Java) [2].

GeoServer es un servidor *Web Feature Server* (WFS, interfaz estándar de sistemas OpenGIS) de código abierto que permite conectar datos del usuario con información Geoespacial. Además permite la publicación de datos geoespaciales a través de imágenes/mapas utilizando Web Map Server (WMS), obtención de datos a través del WFS, la actualización de información a través del WFS-T (*Web Feature Server-Transactional*) que provee un versionamiento de información y posee herramientas de administración muy simples que permiten una configuración en detalle. La meta principal de este software es permitir un fácil uso de estándares, para que datos interoperables sean compartidos masivamente. Éste es el GIS que se utilizará, dada la existencia de una gran y activa comunidad, de código abierto, y, a diferencia de MapServer, soporta ambientes de producción de alta demanda. Está escrito en Java y existen diversas herramientas como GeoTool, que lo complementan [9].

Geotool es una librería de manipulación de información geoespacial a través de procedimientos estándar, desarrollada también en Java.

2.2. Flash

Flash alude tanto al programa que permite la edición multimedia, como al reproductor de los archivos compilados (Flash Player), escrito y distribuido por Macromedia hasta el 2005, luego por Adobe. Utiliza gráficos vectoriales, imágenes raster, sonido, programación a través de scripts, vídeo y audio. En sentido estricto, Flash es el entorno y Flash Player es el programa de máquina virtual utilizado para ejecutar los archivos generados con Flash.

Los archivos de Flash, tienen generalmente la extensión de archivo .SWF. Los archivos de Flash aparecen muy a menudo como animaciones en páginas Web y sitios Web multimedia, y más recientemente Aplicaciones de Internet Ricas (RIA).

En versiones recientes, Flash ha ido más allá de las animaciones simples, convirtiéndolo en una herramienta de desarrollo completa, para crear principalmente elementos multimedia e interactivos para Internet.

El contenido de la aplicación se almacena en un archivo .fla, que al compilarlo une la película con los scripts, creando el ejecutable, o el .swf, Este archivo se publica y es accesible desde cualquier Flash Player, que por hoy en día está dentro de las aplicaciones más utilizada en Internet [13].

Una de las características de Flash es el modelo de seguridad sandbox, es decir que las aplicaciones que están reproduciéndose en un navegador disponen de recursos muy restringidos y limitados, por ejemplo, no es posible acceder a archivos del disco. A partir de la versión 7 del reproductor, las aplicaciones Flash sólo pueden obtener recursos (como imágenes, sonidos, fuentes, entre otros) desde el dominio que fueron publicados, a menos que sea permitido explícitamente en la configuración de seguridad.

2.2.1. Usando ActionScript 3 (AS3)

ActionScript es el lenguaje nativo para la creación de Script en Flash, desde la versión 2.0, pasa de ser de programación estructurada a orientada a objetos y además nos encontramos con un lenguaje más estricto y más amplio donde es posible crear clases propietarias.

Al iniciar esta memoria ActionScript 3.0 no era demasiado popular, basta con mencionar que la librería 3D utilizada aún no tenía una versión estable disponible en AS3, pero era necesario correr el riesgo por la mantenibilidad del código y la similitud que tiene con Java. Además, Adobe, publicita que es más optima que la versión 2.0 y es compatible con otras aplicaciones como Flex.

La sintaxis cambio bastante en comparación a la versión anterior.

El siguiente código imprime "Hola mundo" de la escena:

```
CreateTextField ( "saludo", 0, 0, 0, 100, 100);
saludo.text = "Hola, mundo!";
```

Esto crea un campo de texto en profundidad 0, en la ubicación 0, a 0 píxeles de la pantalla, con 100 píxeles de ancho y alto. A continuación, al parámetro "text" del componente "saludo" se le asigna el valor "Hola, mundo!" que se visualizará automáticamente en el reproductor.

Al crear una clase externa en ActionScript 2.0, el ejemplo anterior podría ser escrito así:

```
class com.example.Greeter extends MovieClip
{
    public function Greeter()
    {
        var txtHello:TextField = this.createTextField("txtHello", 0, 0, 0, 100, 100);
        txtHello.text = "Hola, mundo!";
    }
}
```

En ActionScript 3,0 los programas son un poco más grandes y más complejos debido a la estructura intrínsecamente distinta de la aplicación, al incremento de la separación en el código y al lenguaje de programación del IDE Flash.

```

package com.example
{
import flash.text.TextField;
import flash.display.Sprite;

public class Greeter extends Sprite
{
public function Greeter()
{
var txtHello:TextField = new TextField();
txtHello.text = "Hello World";
addChild(txtHello);
}
}
}

```

Como se puede ver, tiene mayor orden, legibilidad, es tipado (no estricto) y permite la aplicación de patrones actuales.

2.2.2. E4X

Una de las razones por la que se optó utilizar AS3 es la existencia natural de E4X.

ECMAScript for XML (E4X) es una extensión de lenguajes de programación utilizado para trabajar nativamente XMLs con ECMAScript, que trata al archivo como si fuese un objeto, facilitando el acceso al documento XML en una forma que imita su estructura. El objetivo es proporcionar una alternativa a las interfaces DOM que utilizan una sintaxis sencilla de consulta para acceder a XMLs. Cada rama del documento es tratado como una primitiva (enteros, booleanos, Strings, entre otros). Este es un ejemplo:

```

var bodega = <contenido nombre="Bodega central">
  <item tipo="bicicletas" precio="100000" cantidad="6"/>
  <item tipo="patines" precio="25000" cantidad="10"/>
  <item tipo="skates" precio="60000" cantidad="3"/>
</ contenido >;

alert(bodega.item.@tipo == "patines").@cantidad);
alert(bodega.@nombre);
for each( var precio in bodega..@precio ) {
  alert(precio);
}

```

De esta forma se recorre el documento para obtener la cantidad de patines y los precios de cada ítem.

Dada la cantidad de Webservice que son utilizados para la comunicación, E4X es una herramienta que facilita el desarrollo.

2.2.3. Librerías 3D para Flash

Existen variadas alternativas en desarrollo para ser usadas como librerías 3D. Las más reconocidas son Papervision, Sandy 3D y WireEngine.

Flash, en forma nativa, no tiene la capacidad de generar gráficos en 3 dimensiones, ni utiliza las capacidades de los procesadores gráficos de las nuevas tarjetas asociados a OpenGL o DirectX. Las librerías 3D de Flash, básicamente simulan la tercera dimensión a través de proyecciones de vectores dibujados mediante primitivas de Flash, lo que encarece un poco el cálculo a nivel de CPU principal a diferencia de las librerías antes mencionadas (OpenGL y DirectX) que están optimizadas a nivel de Chip.

Seleccionar la librería a utilizar no fue simple. La primera librería en ser descartada fue Papervision, ya que en general, las revisiones existentes destacan de la calidad del rendering generado en desmedro del rendimiento y para la aplicación a desarrollar lo importante es un rápido cálculo y generación de superficies, ya que los mapas serán referenciales y no necesitan representar plenamente la realidad.

La segunda librería testada y utilizada fue Sandy 3D. Para entender la estructura del mundo 3D, debemos tener una manera de describirlo. Normalmente se utiliza el concepto de un escenario gráfico o un árbol. La escena describe la relación entre las diferentes partes que componen el mundo. Tradicionalmente, el mundo mismo es representado por un objeto "mundo" o "universo". El árbol que representa este mundo, está compuesto por nodos, ramas y hojas. La raíz en el mundo es el objeto principal Node, o Locale.

Las ramas se generan a partir de un único Root Node (nodo raíz), que contiene otros nodos con diferentes propiedades y funciones. En Sandy, un nodo es un Group (grupos), que agrupa objetos 3D visibles o transformaciones que se aplican a sus hijos.

Hay tres tipos de nodos en Sandy, el Group, el TransformGroup y la Object3D.

El Grupo y TransformGroup son nodos de la rama, lo que significa que pueden tener hijos como un subconjunto del árbol o rama. Un grupo es un objeto general de la agrupación, mientras que un TransformGroup se utiliza para aplicar transformaciones, tales como la rotación o la traslación a todos sus hijos.

Un Object3D representa un objeto 3D visible en el mundo. Es un nodo Hoja, es decir, que no puede tener hijos. La Figura 2 muestra gráficamente lo anteriormente explicado.

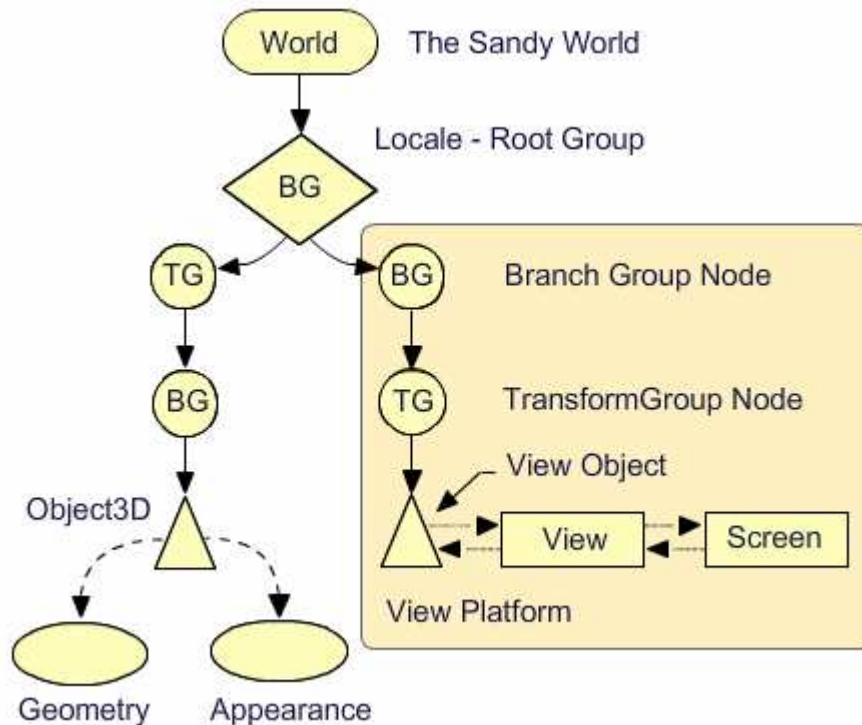


Figura 2

Si se está familiarizado con Java 3D se reconoce la estructura, pues Sandy 3D está basado en Java 3D y posee una API muy similar.

Esta estructura permite una fácil mantención y comprensión de la escena. La librería fue utilizada en el comienzo del desarrollo hasta que se detectaron problemas de rendimiento y complejidad para crear polígonos irregulares.

Los polígonos eran generados a partir de primitivas ofrecidas por la librería tales como cajas, cilindros, pirámides, esferas, entre otros. Para definir un polígono irregular de N lados, figura que es utilizada para definir las superficies del mapa, era necesario crear una primitiva propia. Para este proceso había que determinar a través de vectores, cada vértice del polígono, definir las normales con respecto a cada vértice y las normales de material, que son vectores que definen la orientación de las texturas (se apliquen o no). Aparte de la dificultad de definir una primitiva, este esquema recarga el cálculo, por lo que se descartó.

Finalmente la librería usada es WireEngine3D versión 3.0.10. Esta librería básicamente posee el mismo concepto de mundo o escena 3D, pero maneja sólo un nivel de nodos, es decir un solo grupo, lo que agiliza el cálculo. La calidad del rendering es simple y posee algunos problemas con las texturas, lo que se ve compensado con la capacidad de manejar más de 1000 objetos en pantalla sin grandes pérdidas de rendimiento [11].

Inicialmente esta librería había sido descartada, ya que la única versión que existía era para ActionScript 2.0 y durante el trabajo apareció la versión estable para ActionScript 3.0 con varias optimizaciones.

Otra ventaja, es la facilidad para dibujar polígonos irregulares, simplemente se define un grupo de vectores 3D (Coordenadas X, Y y Z) y un material de superficie, que puede ser un color, con esto el “motor” es capaz de hacer rendering de la figura.

La calidad del rendering es definida en el Mundo 3D (escena) que además posee una cámara, que a través de la modificación de sus parámetros, es posible cambiar el Fov (field of view, campo de visión) y proyección, aplicar el algoritmo de Frustrum Culling que elimina los objetos que no serán dibujados por estar ocultos a la cámara, (entre otras técnicas para optimizar la aplicación), aplicar transformaciones, zoom, etc.

Una anécdota interesante, a la hora de la implementación, fue la posibilidad de contactar al desarrollador de esta librería, quien prestó ayuda a la hora de resolver algunos problemas relacionados al rendering y configuración del Framework.

Cabe destacar que estos proyectos de software libre cuentan, cada uno, con una comunidad que participa activamente haciendo crecer el proyecto

2.2.4. La alternativa gratuita, SVG

Inicialmente la opción de utilizar SVG era bastante fuerte, ya que GeoServer es capaz de hacer rendering de mapas en formato SVG, es decir un dibujo vectorial por defecto.

SVG es un formato para crear gráficos vectoriales, corresponde con las siglas Scalable Vector Graphics, que viene a significar Gráficos Vectoriales Escalables.

Este formato está basado en XML y su desarrollo está a cargo del consorcio W3C (World Wide Web Consortium) y permite generar tanto contenido estático y dinámico a través de scripts de animación y es posible generar renderers bastante complejos.

Uno de los problemas es que es soportado nativamente por pocos navegadores. Inicialmente viene incluido para Firefox, para Microsoft Explorer debe ser instalado como plugin.

Pero el mayor inconveniente es que el código generado por GeoServer es rígido y no es posible modificarlo fácilmente, es decir no se tendría control sobre la imagen generada y no sería posible agregarle información.

Una alternativa estudiada fue utilizar Deng, componente Flash que es capaz de interpreta SVG y representarlo, pero el resultado es el mismo, un rendering generado no manipulable.

2.3. Dispositivos GPS

El Sistema de Posicionamiento Global GPS (siglas en inglés de Global Positioning System), es un método de posicionamiento y navegación basado en las señales transmitidas por la constelación de satélites NAVSTAR (siglas en inglés de Navigation Satellite Timing And Ranging), que son recibidas por receptores portátiles en Tierra. Las señales múltiples que se reciben simultáneamente provenientes de las sucesivas posiciones de los satélites, se utilizan para resolver las ambigüedades y permitir con esto, la determinación de la posición tridimensional del punto por conocer. En la actualidad, NAVSTAR utilizar 27 satélites, 24 operativos y 3 de respaldo.

A continuación una aproximación al funcionamiento de un GPS obtenido desde “guiabasica.com” [12]:

“Los navegadores, captan la señal de más de una satélite, dependiendo del tipo de GPS que utilizemos pueden captar de 9 a 13 satélites simultáneamente, y basándose en las coordenadas que nos ofrecen, realizan una triangulación que nos retorna el punto exacto en que nos encontramos.

Pasaremos a explicarlo de una manera más técnica:

Como con cualquier otro sistema de radiolocalización, si calculamos la distancia a tres transmisores de posición conocida, podemos triangular nuestra posición en cualquier lugar de la Tierra.

Así pues, como primera premisa es necesario que nuestro receptor G.P.S. conozca la posición exacta de los satélites de la constelación.

Supongamos que tomamos nuestra distancia respecto a un satélite y resulta ser de 18.000 km. Desde el punto de vista del satélite, nuestra posición estará en algún punto de una esfera de 18000 km. de radio alrededor de él (Figura 4).

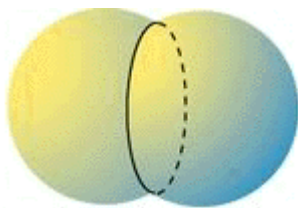


Figura 3

Ahora hacemos la misma medida de cara a otro satélite, y es de 17000 km. Por tanto, nuestra posición estará en la intersección de la anterior esfera y otra actual de radio 17000 km alrededor de este satélite. La intersección de ambas esferas es una circunferencia (Figura 3), luego con dos mediciones nos ubicamos en algún punto de dicha circunferencia.

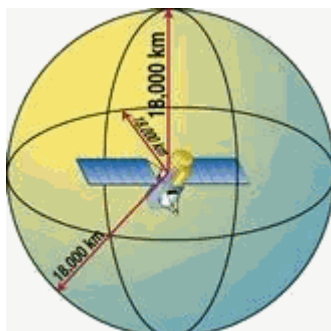


Figura 4

Si además hacemos la misma medida con respecto a un tercer satélite y resulta ser de 20000 km, por ejemplo, precisaremos nuestra posición a la intersección de las tres esferas (Figura 5). La intersección de 3 esferas da lugar a dos puntos, de los cuales uno de ellos será una solución sin sentido para nuestro receptor. En la práctica es necesaria una medida adicional respecto a un cuarto satélite para precisar la posición”.

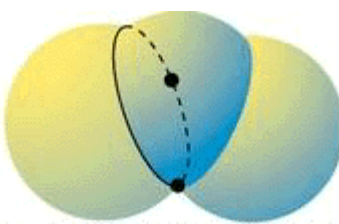


Figura 5

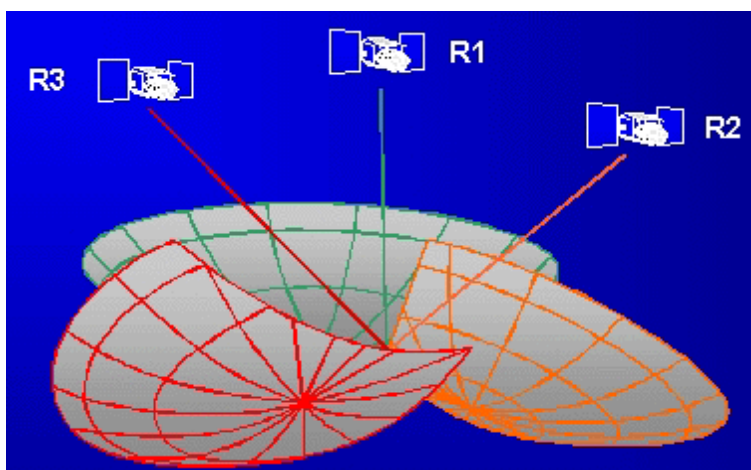


Figura 6

Los equipos utilizados para este trabajo reportan su posición a un servidor central mediante una red GRPS (celular). El servidor almacena las posiciones y su sistema es capaz de procesar la información para ver la localización en línea, obtener informes de gestión y alimentar sistemas externos a través de Webservices (Figura 7), entre otras funcionalidades.

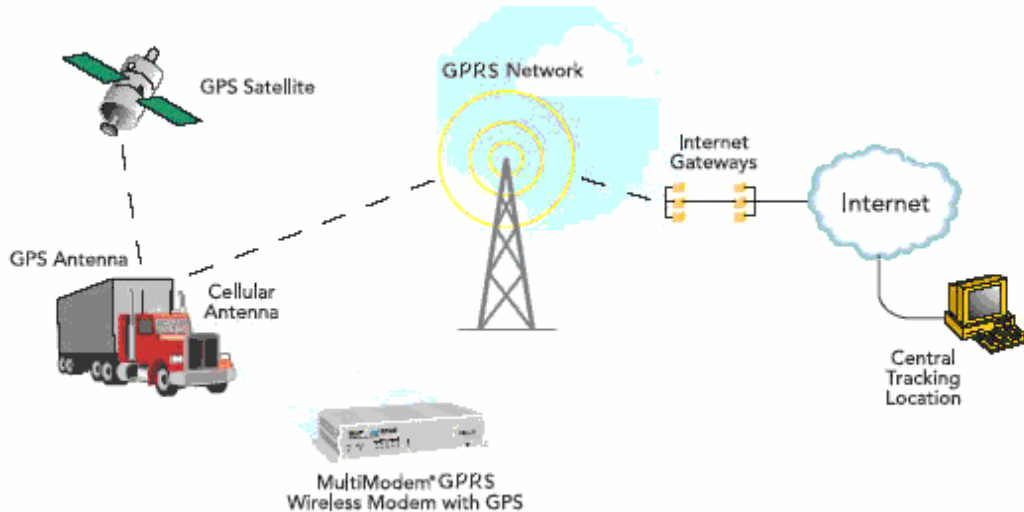


Figura 7

La información es reportada en forma discreta al servidor. Aunque el equipo sepa su posición en todo momento, éste envía los datos dependiendo de la configuración del equipo, que puede ser cada cierto número de minutos transcurridos o cada cierta cantidad de metros recorridos. Si la configuración es demasiado espaciada, como una gran cantidad de metros y tiempo, las posiciones serán más espaciadas y el camino reflejado no será tan fidedigno como uno que reporte más seguido, es decir con un espaciado menor, la información reflejará un camino más continuo.

¿Por qué no reportar la posición, entonces, cada 1 segundo o cada un metro? Recordemos que la información viaja sobre la red GPRS, por lo que hay un cobro asociado y esa cantidad de tráfico sería muy costosa. En este trabajo, la finalidad es representar la posición en el mapa sin importar la continuidad, por lo que no es relevante.

2.4. Trabajos Relacionados

A continuación algunas aplicaciones que permiten explotar la información contenida en un GIS, pero que no constituyen un GIS en sí mismas.

- 3D GIS Viewer

Es un visualizador 3D GIS Web desarrollado por Telemorphic inc., que da la posibilidad al usuario de disfrutar las capacidades de una aplicación de escritorio a través de un Browser. Este visualizador genera mapas de elevación intentando recrear terrenos de forma realista. (Figura 8)

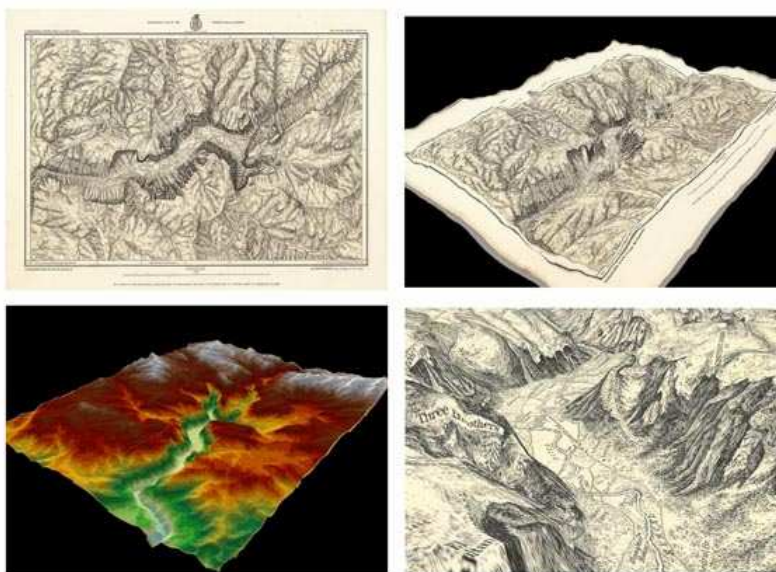


Figura 8

Esta aplicación está enfocada al modelamiento de terreno, y permite una revisión histórica de fallas como valles, mesetas, etc. Posee gran detalle y permite navegar por el relieve obtenido de los mapas de manera similar a los video juegos, sobrevolándolo.

El problema es que no es capaz de interactuar directamente con un servidor GIS, sino que dado el volumen de datos que se necesitan para una visualización, debe tenerlos pre-cargados.

Otra desventaja, es que requiere para su funcionamiento una tarjeta de video con aceleración, mientras que Flash es soportado con cualquier tipo de computador.

- Mapserver

Es una extensión a Apache que permite presentar páginas HTML e imágenes raster generadas a partir de una conexión a una la base de datos PostGis o accediendo directamente a un fichero shapefile (glosario de terminos incorporarlo). Mediante un API en varios lenguajes (PHP, Perl, Python, Java, etc.) y un lenguaje de configuración propio, mapscript, permite construir aplicaciones Web interactivas. Utilizando la librería MING permite generar ficheros en formato Macromedia Flash .SWF

Aunque es capaz de manejar bastantes formatos, no posee la capacidad de generar visualizaciones 3D.

- Quantum GIS (QGis)

Aplicación de escritorio de visualización en 2D de recursos GIS, permite organizar la información por capas, desarrollada con el framework QT. Permite editar la base de datos PostGis o visualizar ficheros Shapefile, con ficheros Raster como mapa de fondo. (Figura 9)

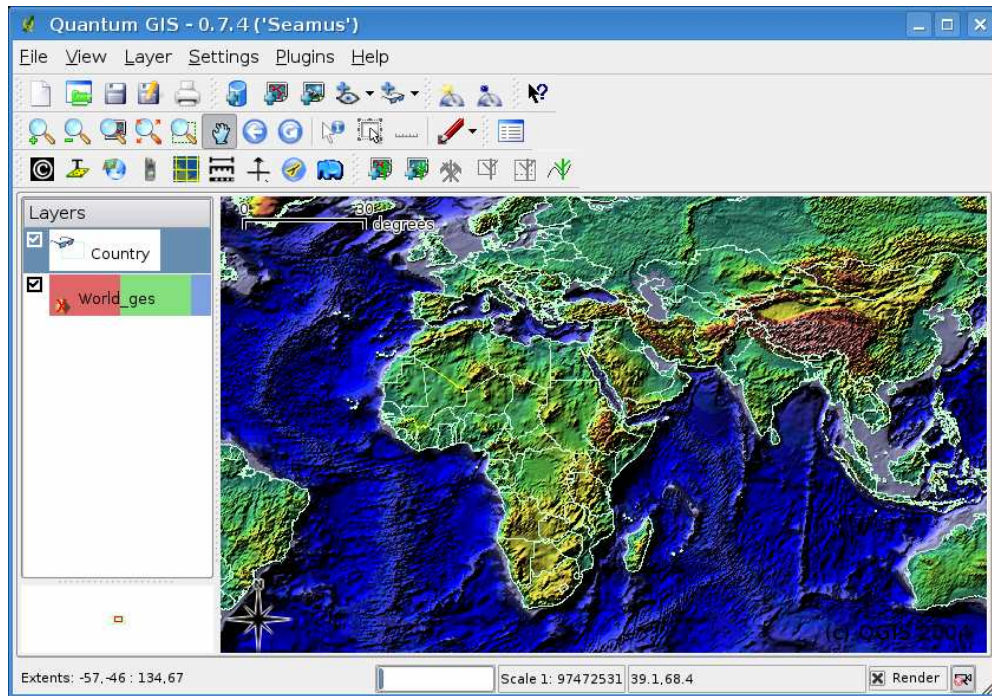


Figura 9

Es multiplataforma y permite manejar formatos raster y vectoriales.

Una de sus mayores ventajas es la posibilidad de usar Quantum GIS como entorno gráfico de SIG GRASS, haciendo más amigable el uso de este último potente entorno. Tal como el sistema anterior, no es capaz de generar visualización 3D, restando fluidez al uso del entorno [10].

- UDIG

Aplicación de escritorio de visualización en 2D de recursos GIS, permite organizar la información por capas, desarrollada para el framework Eclipse RCP (Rich Client Platform) sobre plataforma Java. Permite editar Gis propietarios, la base de datos PostGis o visualizar ficheros Shapefile (ficheros estándar para formato Geoespacial), con ficheros Raster como mapa de fondo.

2.5. Requerimientos de la aplicación y solución propuesta

La aplicación tiene los siguientes requerimientos, y junto a ellos, se incluye la solución propuesta.

- Rendimiento

Uno de requisitos esenciales es enfocar la aplicación a obtener un alto desempeño, que es posible de lograr, ya que está enfocado a un usuario que no necesita demasiada precisión en la representación, sino que lo relevante para él es saber, por ejemplo, a cuantas calles se encuentra su móvil (que tiene instalado el equipo GPS) de un determinado cliente que debe visitar dentro de algunos minutos. Si agregamos que muchos datos geográficos y demográficos (también manejados por los GIS) no son necesarios para la identificación física de un GPS, es posible disminuir el volumen de datos necesarios.

La técnica de generalización de información será implementada para eliminar información innecesaria que no es posible observar.

- Portabilidad

La aplicación debe estar optimizada para que la mayoría de los computadores la soporte sin problemas, y que no sea necesario ningún tipo de hardware específico y costoso. Estos requerimientos se cumplen utilizando Flash.

- Confiabilidad en la información

Los datos visualizados deben evidenciar los datos más actuales en el servidor, y si éstos cambian, la visualización también debe hacerlo, por lo tanto la información visual de un equipo GPS debe representar la última o actual posición de éste en un mapa.

- Usabilidad

La aplicación debe entregar las herramientas necesarias para realizar una visita del mapa en todas direcciones a través de desplazamientos, zoom, rotación y traslación. Como no es posible obtener todos los datos de una superficie, debe manejar un “buffer” (datos almacenados en memoria) de información a medida que se visualizan nuevas zonas del mapa.

- Correctitud y robustez

Cualquier actualización permitida y efectuada en el mapa, debe reflejarse y almacenarse en el servidor. Como Flash no posee buenas herramientas de conexión o manejo de datos, la interacción con el servidor será a través de WebServices, quienes actuarán como interfaz de comunicación.

3. Análisis y Diseño

En este capítulo se muestra la estrategia utilizada para obtener la representación de un equipo GPS sobre el rendering de un mapa en 3 dimensiones. Se expone la arquitectura, que muestra como interactúan los principales componentes. Luego se describen las clases que implementan estos componentes. También se describe la integración del sistema externo que proporciona la información de los móviles GPS y finalmente la estrategia asumida para manejar el comportamiento de la aplicación en relación a las diferentes escalas.

3.1. Arquitectura

A continuación se muestra el esquema general de la relación entre las componentes y manera de comunicarse para lograr visualizar los móviles sobre un mapa 3D (Figura 10).

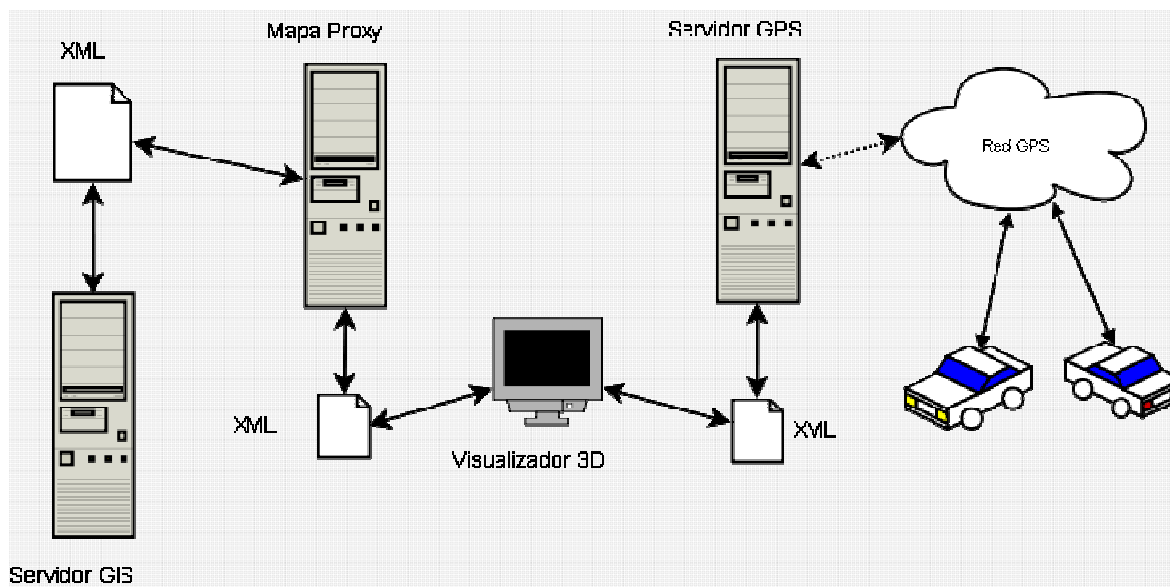


Figura 10

3.1.1. Servidor GIS

El servidor GIS fue el primer componente instalado. La principal función de este es proporcionar la información geográfica de la zona que se necesite. Esta información es obtenida a través de una petición por parte del Mapa Proxy.

Específicamente, la petición es generada por el visualizador 3D, quién a través del Mapa Proxy calcula las coordenadas necesarias para generar el rendering del

mapa. Los datos son enviados al servidor GIS por medio de una consulta WFS basada en un XML, el resultado es enviado mediante otro XML al Mapa Proxy, quien será el encargado de procesarlo.

Un punto a destacar es que aunque exista un estándar de comunicación y consulta con el servidor GIS es necesario conocer previamente la estructura de la capa consultada, ya que la petición se debe hacer sobre características específicas como la geometría, población o puntos de interés. Estas características están dadas por los Shapefiles, que serán explicados en el capítulo 4.

3.1.2. Servidor Proxy de Información

El Mapa Proxy es un agente que es utilizado para procesar previamente la información geográfica proporcionada por el GIS y provee un Webservice por cada capa de consulta. Inicialmente es gatillado por una petición GET desde el visualizador 3D que indica la capa consultada, la escala y el Bounding (rectángulo que encierra la zona consultada), y éste la traduce a una consulta WFS.

La necesidad de procesar la información antes de ser desplegada se debe a la capacidad de cálculo limitada de Flash. Un cálculo demoroso sólo quita prioridad al rendering, lo que se refleja como una aplicación lenta y menos fluida.

La extensión E4X es bastante limitada en el parsing de XML con namespaces complejos, por lo que una consulta directa del visualizador 3D al servidor GIS no era una opción a considerar.

3.1.3. Interfaz 3D Flash

La interfaz 3D Flash es la encargada de interactuar con el usuario y mostrar la representación de toda la información entregada por los componentes del sistema, es decir, los móviles GPS y los mapas filtrados.

El visualizador 3D se carga en el cliente utilizando el plug-in Flash de los navegadores y obtiene la información que debe desplegar en base a Webservices que transfieren XMLs conocidos.

Inicialmente se genera una llamada, por cada capa, al mapa Proxy, indicando el bounding box de la zona que se desea consultar y una escala predeterminada, al recibir la respuesta la procesa y genera el rendering de las capas. Seguidamente genera una llamada al servidor de información GPS para ingresar como usuario del sistema, generando una sesión en el servidor, luego trae las posiciones de los móviles asociados a la cuenta y los despliega en el mapa.

La aplicación maneja dos tipos de coordenadas, las del mundo real en grados de latitud y longitud, y la interna en "píxeles". Estas coordenadas tienen una relación lineal entre sí, por lo que la conversión es casi directa.

Cuando la información es obtenida desde el servidor, ésta queda almacenada dentro de la aplicación como “caché” para disminuir al máximo la transferencia de datos y aumentar el desempeño. Dado que la información gráfica realmente es vectorial, la carga no es demasiada.

Finalmente teniendo la información geográfica y posicional de los vehículos, sólo falta “intersectarlas” para lograr un despliegue de ambas a través de un rendering de 3 dimensiones.

3.1.4. Integración con sistemas GPS

El sistema de información GPS existía mucho antes de comenzar el proyecto y es perteneciente a una empresa del rubro (Tastets System Ltda.). La infraestructura es bastante grande y su descripción ha sido simplificada en el capítulo 2.3 (Dispositivos GPS). No sólo maneja la información de posiciones, sino que además maneja una compleja estructura de usuarios, asociados a perfiles dependiendo de la empresa y tipo de plan. A modo de simplificar la aplicación y la integración misma, es posible visualizarla como la cuenta de un usuario asociada a diversos móviles para los cuales posee información de su localización

Como anteriormente se mencionó, la posición reportada de los equipos GPS es discreta, por lo que no es necesario consultarla en cada segundo, sino que es posible obtenerla cada ciertos intervalos sin que la información deje de representar la realidad.

El sistema de información GPS provee Webservices que entregan información sobre los vehículos a un usuario del sistema.

3.2. Diagrama de Clases

Esquema de la disposición de clases y explicación de las más representativas.

3.2.1. Interfaz 3D

La clase más importante de la interfaz es la visualizador3D. Ésta es la encargada de centralizar y sincronizar las peticiones desde la representación visual hacia el servidor Proxy y viceversa.

A continuación el diagrama general de clases con los métodos más relevantes (Figura 11):

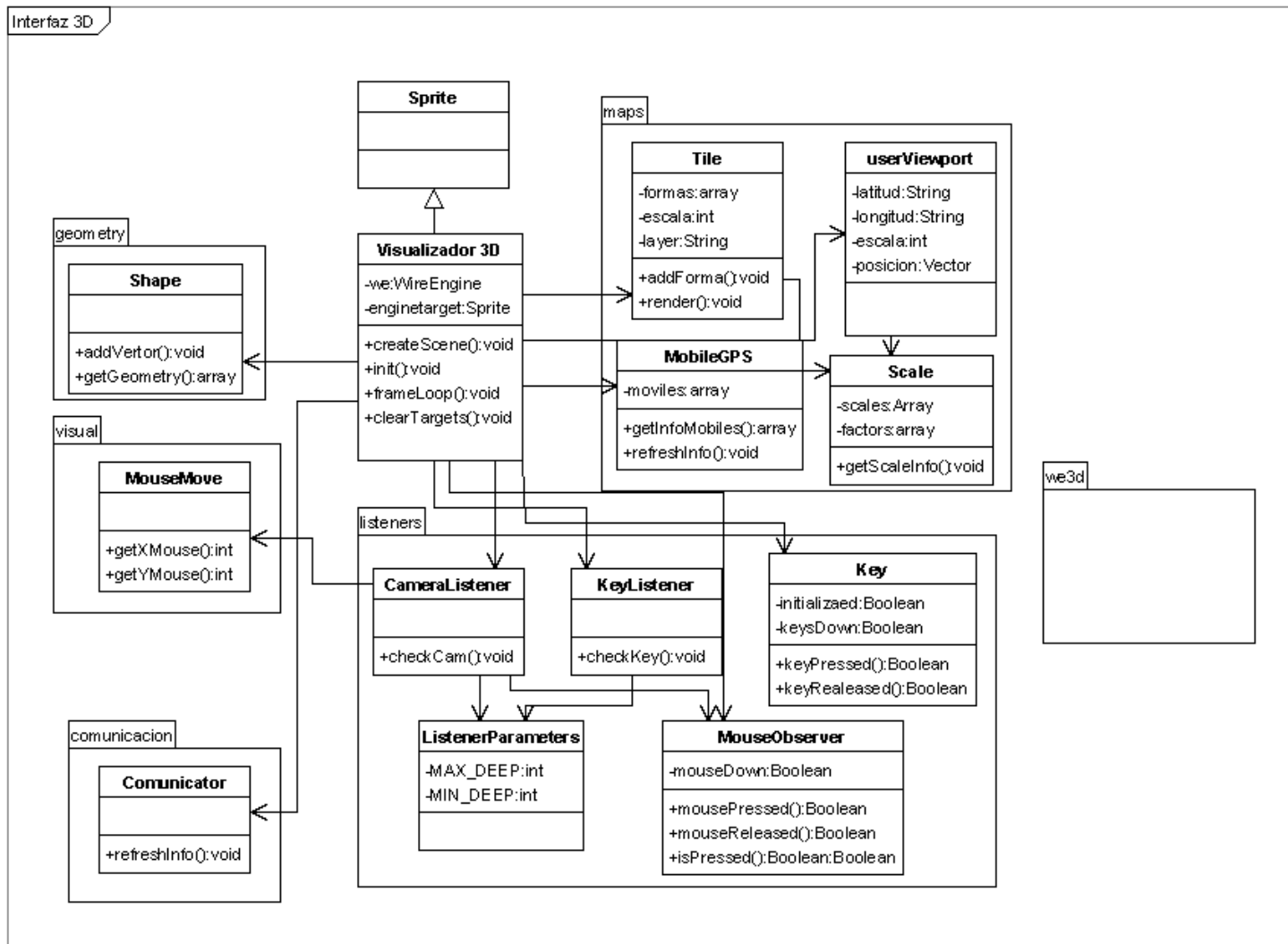


Figura 11

- La clase **visualizador3D** es el *core* de la aplicación. Hereda de la clase `Sprite`, nativa de Flash, esto hace que sea un objeto capaz de ser incluido en la “línea de tiempo” de Flash, es decir poseerá un evento `ENTER_FRAME` que será gatillado una vez por cada ciclo de rendering (frame). Esta clase utiliza la librería `we3d` (`WireEngine 3D`) para generar los polígonos de superficie. El método `CreateScene()` es el encargado de iniciar todos los objetos 3D. Previamente en `init()` se fijan los parámetro del rendering.
- La clase **Shape** simplemente contiene un arreglo de vectores con 3 dimensiones. El conjunto completo de vectores describe un polígono irregular o una línea (dependiendo del tipo) que luego será representado en la interfaz. Todo vector es agregado por `addVector()`.
- La clase **MouseMove** es estática y está encargada de medir el desplazamiento del Mouse dentro de la interfaz 3D. Internamente va guardando la última posición del Mouse (x,y) en píxeles, posee métodos que calculan la diferencia y la guardan en variables estáticas que pueden ser consultadas globalmente
- La clase **MouseObserver** también es estática y es la encargada de detectar el Mouse está presionado o no. También posee variables estáticas que pueden ser consultadas globalmente.
- La clase **Key** es análoga a la anterior, pero detectando el teclado.
- Clases **CameraListener** y **KeyListeners** utilizan las clases anteriormente descritas para calcular el movimiento que debe tener la cámara dependiendo del movimiento del Mouse y la Tecla presionada. Estas clases interactúan directamente con las librerías `we3d`.
- La clase **Tile** es la que contiene toda la información geográfica de una capa. Posee las coordenadas del bounding box de la zona requerida y la escala. Utilizando la clase **Comunicator** obtiene la información desde el Mapa Proxy y la deja en memoria para no ser cargada nuevamente. Una vez cargada la información, el método `rendering` logra realiza la diagramación 3D utilizando `we3d` cargándolo en la interfaz 3D. Cualquier objeto agregado por `addForma()` será desplegado. El método `Render()` es ejecutado en cada ciclo.
- La clase **Scale** es muy importante para el funcionamiento del mapa. Ésta contiene la información de los umbrales que marcan la transición entre una escala y otra, lo que gatilla la recarga de información, cambio de parámetros en el visualizador, como la velocidad a la que avanza el Mouse y velocidad de rotación, comportamiento de las capas, entre otros.

- La clase **UserViewport** es la encargada de manipular los datos internos de coordenadas de píxeles y convertirlas a coordenadas geográficas. Este cálculo permite determinar las posiciones de los vehículos en el mapa.
- La clase **MobileGPS** mantiene actualizada la posición de los móviles, a través del método `refreshInfo()` y se encarga de realizar el rendering en pantalla.
- Las clases **Parameters** y **ListenerParameters** definen los parámetros globales de la aplicación como la URL de conexión, tamaño del Tile, profundidad máxima y mínima, entre otros.

3.2.2. Servidor Proxy de Información

El servidor Proxy es el componente que maneja la información particular de cada capa. Conoce la estructura de cada capa y la forma de transformar la escala real a la virtual. Al mismo tiempo es el encargado de recibir las peticiones de la Interfaz 3D y traducirlas a consultas WFS.

A continuación se muestra el diagrama de clases (Figura 12), que es simple, sin embargo efectivo.

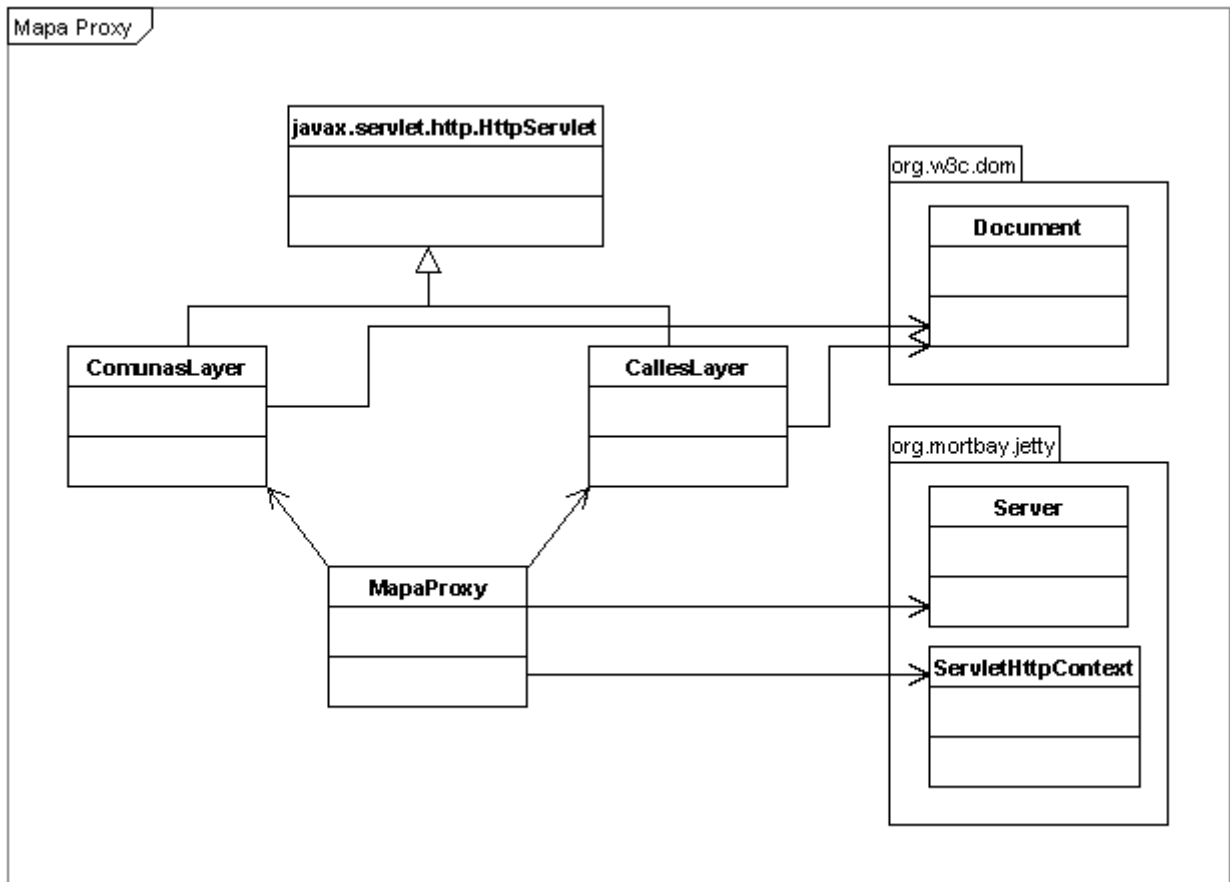


Figura 12

- La clase MapProxy es la principal. Esta clase inicia un servidor de Webservices (realmente Servlets) utilizando las librerías de Jetty y publica un Servlet por cada layer existente, en este caso el ComunasLayer y el CallesLayer.
- Las clases ComunasLayer y CallesLayer funcionan de forma similar. Ambas heredan de HttpServlet, por lo que deben implementar los métodos doGet y doPost para responder a las peticiones generadas por el visualizador. Ambas utilizan las librerías w3c para trabajar los archivos XML resultantes

provenientes del Servidor GIS y para generar el archivos XML de respuesta.

3.3. Manejo de Escalas y Peticiones de Información Geográfica

A continuación se explica la estrategia utilizada para manejar el cambio de escala y las peticiones de información geográfica al servidor GIS. El cambio de escala se hace necesario al momento de observar distintos niveles de detalles de las capas existentes. Por ejemplo, a una escala muy grande, la lejanía es mayor y no es necesario tener los detalles de las calles, en cambio a una escala menor sería bueno apreciar todos los detalles que rodean el área en que se encuentra el móvil en estudio.

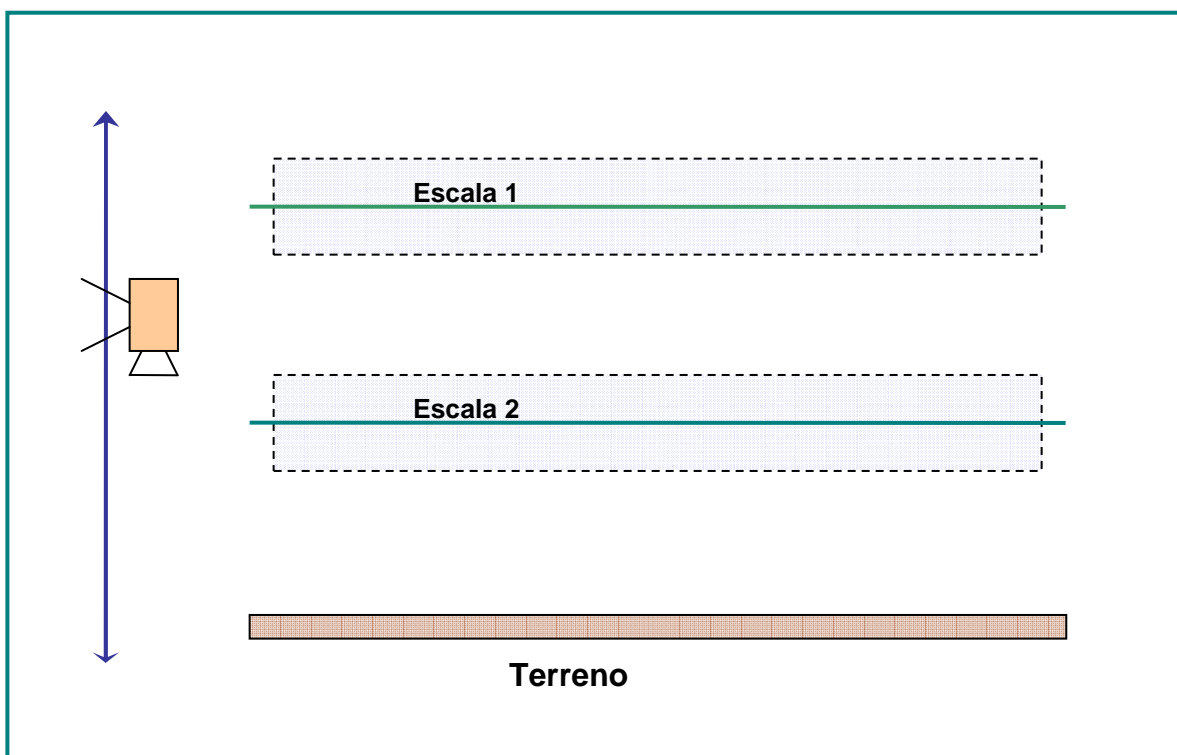


Figura 13

La Figura 13 muestra el esquema general del tratamiento de las escalas con respecto a la posición de la cámara.

A la izquierda de la figura se aprecia la cámara y su trayectoria. Las líneas verdes representan los límites de distancia en los que se aplicará una escala, el rectángulo café es el terreno generado (en verdad es el plano donde se realiza el rendering de información geográfica) y los rectángulos segmentados son las zonas de transición de escalas.

La estrategia del cambio de escalas es la siguiente:

- Inicialmente la cámara tiene una escala determinada. A medida que se cambia el zoom, cambia la distancia al terreno.
- Al llegar a una zona de transición de escalas (la segmentada en la Figura 13) el visualizador comienza a pedir la información geográfica de la siguiente escala. El bounding que consulta corresponde al apuntado por el centro de la cámara más una “delta” que depende de la escala.
- Mientras el visualizador está en la zona de transición de escalas, la información de las dos escalas es visible y se sobrepone. Si la cámara sigue avanzado y cambia su distancia con el terreno, se oculta la información de la escala anterior.
- Mientras se está en una determinada escala, la velocidad de la cámara y precisión de la información es diferente. Entre más cercana la cámara al terreno, hay más detalle y se tiene la sensación de menos velocidad. En forma contraria, al alejarse el desplazamiento es más rápido y las formas son menos precisas.
- El cambio de escala también afecta al Mapa Proxy. La generalización (simplificación del detalle) cambia dependiendo de la escala. En el siguiente capítulo se detallará en la sección de Generalización de puntos (4.3.3), pero como avance se puede mencionar que a mayor escala (más lejos) la generalización es mayor, ya que el detalle necesitado es menor también.

4. Implementación

Este capítulo describe los aspectos más relevantes asociados a la implementación de la aplicación. Estos incluyen:

- Las diferentes herramientas asociadas al desarrollo y el motivo de su elección.
- Utilización del servidor GIS, su configuración, pruebas de funcionamiento y agregar información geográfica por Shapefiles.
- Utilización del protocolo de publicación y consulta de datos de información geográfica (WFS) a través de un servidor GIS basado en WMS.
- Implementación del servidor Proxy, como se produce la comunicación con el servidor GIS y el tratamiento que tiene con la información de cada capa (layer) conocida.
- Como se implementa la generalización de puntos, que permite la simplificación de datos para la optimización de la aplicación.
- Implementación de la interfaz gráfica, controles asociados y manejo de proyecciones.
- Integración con el sistema GPS para ingreso del usuario y obtención de la información para la representación de los móviles.

4.1. Ambiente de Trabajo

A continuación se describen las herramientas utilizadas para la programación y prueba del código.

4.1.1. Eclipse

Eclipse es una plataforma de software de código abierto independiente de una plataforma o un lenguaje. Ésta se prefirió para desarrollar el Mapa Proxy, ya que está basado en Java, al igual que Jetty un “pequeño” servidor Apache, más liviano que se utiliza en el desarrollo. Eclipse permite una depuración bastante precisa y rápida, y es gratuito.

4.1.2. Flex 2 SDK

Inicialmente en este documento se habló de la preferencia por el desarrollo de la aplicación visual con la tecnología de Flash y en especial utilizando ActionScript 3.0 dada su similitud con Java en el manejo de clases, recolector de basura, mantenibilidad en el futuro y la existencia nativa de la extensión E4X.

Al iniciar esta memoria existió la duda entre utilizar los IDE de desarrollo perteneciente a Adobe (y Macromedia), como Flash MX2004 que sólo soporta ActionScript 2.0 y los recientes Adobe Flash CS3 y Adobe Flex que soportan ActionScript 3.0. La principal diferencia entre Flash CS3 y Flex es en que el primero está enfocado al desarrollo artístico de películas y Flex en aplicaciones RIA, Rich Internet Applications (Aplicaciones Ricas de Internet) que poseen más ventajas que las tradicionales aplicaciones Web y se presentan como una combinación de las ventajas que ofrecen las aplicaciones Web y las aplicaciones tradicionales de escritorio. Finalmente se utilizó Flash CS3 para la compilación.

A mediados del trabajo, Adobe liberó el SDK de Flex con lo que fue posible compilar código de ActionScript 3.0 y MXML (no utilizado en este memoria) obteniendo el mismo resultado que si fuese desarrollado con los IDEs de Adobe, pero en forma gratuita.

4.1.3. FlashDevelop 3.0 Beta2

Como se mencionó anteriormente, con el SDK de Flex es posible compilar el código ActionScript 3.0 para obtener ejecutables Flash.

FlashDevelop fue el IDE escogido para el desarrollo del código.

Aunque existen diversos plugins para escribir código ActionScript en Eclipse, estos no son especializados y tienen problemas con las clases propietarias.

FlashDevelop es un IDE gratuito que se especializó inicialmente en ActionScript 2.0 y a partir de su versión 3.0 de ActionScript 3.0. Tiene un manejo más avanzando en la estructuración de código y posee un plugin para integrar el Flex SDK casi nativamente al trabajo.

4.1.4. SVN

Subversion es un programa para el control de versiones que reemplazó al CVS dadas sus deficiencias. Es software libre y se le conoce también como SVN. Una característica importante de Subversion es que, a diferencia de CVS, los archivos no poseen versiones independientes, sino que el número de revisión corresponde a un único dígito de la versión que identifica el estado común de todos los archivos del repositorio en cierto punto del tiempo.

Aunque este software no fue vital para el desarrollo, si fue necesario para el manejo de una única versión estable.

4.2. Servidor GIS

El servidor GIS es un componente esencial dentro de este trabajo. No fue fácil decidirse por uno, ya que existen diversas alternativas, pero uno de los factores que ayudó a la elección de GeoServer fue su fácil instalación y configuración, además está disponible para varios sistemas operativos, como Windows y Linux.

4.2.1. Instalación de GeoServer

GeoServer posee versiones auto instalables tanto para Windows como para Linux, además de existir la versión WAR para instalarla dentro de un contenedor de Servlets J2EE.

La instalación básica no pasa más allá de ejecutar el instalador e ir seleccionando “siguiente” hasta finalizar. El único requisito previo es tener instalado Java 2 Software Development Kit (J2SDK o JDK) y tener definida la variable de entorno JAVA_HOME.

Si al finalizar la instalación la configuración está correcto, se levanta el servicio de GeoServer y queda disponible en la máquina en el puerto 8080 y posible configurarlo a través de la dirección <http://localhost:8080/geoserver/>

Para lograr un mayor rendimiento al cargar información es posible utilizar como fuente de información bases de datos MySQL o Postgres (PostGIS). La optimización del servidor GIS no es el fin de este trabajo, ya que la información será cargada en la memoria de la aplicación, por lo que sólo de recurrirá a los ShapeFiles, que cumplen la función de proveer datos geográficos. Los Shapefiles son archivos de información geográfica generados por herramientas cartográficas.

4.2.2. Carga de información (Shapefiles)

Un Shapefile es un archivo de formato vectorial donde se guarda la localización de los elementos geográficos y los atributos asociados a ellos. El formato carece de capacidad para almacenar información topológica. Un Shapefile básico posee al menos tres archivos que lo conforman:

- **.shp** - es el archivo que almacena las entidades geométricas de los objetos.
- **.shx** - es el archivo que almacena el índice de las entidades geométricas.
- **.dbf** - el dBASE, o base de datos, es el archivo que almacena la información de los atributos de los objetos.

Además de estos tres archivos requeridos, opcionalmente se pueden utilizar otros para mejorar el funcionamiento en las operaciones de consulta a la base de datos, información sobre la proyección cartográfica, o almacenamiento de metadatos. Estos archivos son:

- **.sbn** y **.sbx** - Almacena el índice espacial de las entidades
- **.fbn** y **.fbx** - Almacena el índice espacial de las entidades para los shapefiles que son inalterables (solo lectura)
- **.ain** y **.aih** - Almacena el índice de atributo de los campos activos en una tabla o el tema de la tabla de atributos.
- **.prj** - Es el archivo que guarda la información referida a sistema de coordenadas.
- **.shp.xml** - Almacena los metadatos del shapefile.

Para este trabajo, los Shapefiles que se utilizaron han sido facilitados por particulares para estudio y poseen información geográfica de las **comunas** de Santiago y otros con las **calles**.

Insertar los Shapefiles a GeoServer no es tarea difícil. Simplemente se ingresa a la página de administración, se agrega un "Datasource" con el nombre de los archivos y luego se crean los Feature Types correspondientes, pero para esto es necesario conocer la estructura de los Shapefiles, ya que la información depende del cartógrafo que lo creó.

4.2.3. Estudio de Features

Como se dijo en el punto anterior, una capa Shapefile trae información propia, y al ser cargada como una colección de datos de un determinado tipo (Feature Type) la información queda indexada dentro del servidor GIS para realizar búsquedas y facilitar su acceso. Aquí el ejemplo de la capa de calles:

```

<?xml version="1.0" encoding="utf-8"?>
<wfs:FeatureCollection      xmlns="http://www.opengis.net/wfs"      xmlns:wfs="http://www.opengis.net/wfs"
xmlns:gps="http://localhost"  xmlns:gml="http://www.opengis.net/gml"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"  xsi:schemaLocation="http://www.opengis.net/wfs  http://localhost:8080/geoserver/schemas/wfs/1.0.0/WFS-
basic.xsd http://localhost http://localhost:8080/geoserver/wfs/DescribeFeatureType?typeName=gps:calles">
  <gml:boundedBy>
    <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
      <gml:coordinates xmlns:gml="http://www.opengis.net/gml" decimal="." cs="," ts=" ">-70.67100454,-33.47668731 -
70.65696646,-33.37130632</gml:coordinates>
    </gml:Box>
  </gml:boundedBy>
  <gml:featureMember>
    <gps:calles fid="calles.161">
      <gml:boundedBy>
        <gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
          <gml:coordinates xmlns:gml="http://www.opengis.net/gml" decimal="." cs="," ts=" ">-70.66204764,-33.45358169 -
70.66139151,-33.45343673</gml:coordinates>
        </gml:Box>
      </gml:boundedBy>
      <gps:the_geom>
        <gml:MultiLineString srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
          <gml:lineStringMember>
            <gml:LineString>
              <gml:coordinates xmlns:gml="http://www.opengis.net/gml" decimal="." cs="," ts=" ">-70.66204764,-
33.45358169 -70.66139151,-33.45343673</gml:coordinates>
            </gml:LineString>
          </gml:lineStringMember>
        </gml:MultiLineString>
      </gps:the_geom>
      <gps:FNODE_>43821</gps:FNODE_>
      <gps:TNODE_>43748</gps:TNODE_>
      <gps:LPOLY_>16777</gps:LPOLY_>
      <gps:RPOLY_>17008</gps:RPOLY_>
      <gps:LENGTH>63.078</gps:LENGTH>
      <gps:STGOC1_>60446</gps:STGOC1_>
      <gps:STGOC1_ID>134491</gps:STGOC1_ID>
      <gps:CIUDAD>Santiago</gps:CIUDAD>
      <gps:CLASE>CALLE</gps:CLASE>
      <gps:COMUNA>Santiago</gps:COMUNA>
      <gps:COM_DER>Santiago</gps:COM_DER>
      <gps:COM_IZQ>Santiago</gps:COM_IZQ>
      <gps:FIN_DER>1910</gps:FIN_DER>
      <gps:FIN_IZQ>1911</gps:FIN_IZQ>
      <gps:IDENTIF>41075</gps:IDENTIF>
      <gps:INI_DER>1962</gps:INI_DER>
      <gps:INI_IZQ>1955</gps:INI_IZQ>
      <gps:NOMBRE>TOESCA</gps:NOMBRE>
      <gps:ALIAS></gps:ALIAS>
      <gps:CLASE_RECL>CALLE</gps:CLASE_RECL>
      <gps:CLASE2>4</gps:CLASE2>
    </gps:calles>
  </gml:featureMember>
  ..... (continúa)

```

En el XML de Feature Type anterior están destacadas en negro las características más relevantes para este estudio.

El tag de <gml:featureMember> encierra a un elemento, en este Shapefile es una calle.

El tag <gml:boundedBy> determina el polígono que encierra esta calle. Esta propiedad se utiliza para realizar búsquedas eficientes por coordenadas. <gps:the_geom> es una propiedad agregada y tiene la información geográfica de la "línea" que describe la calle.

Más abajo se aprecia otras propiedades más simples como <gps:COMUNA> o <gps:NOMBRE> que representan información propia de la calles.

Las propiedades FIN_DER, FIN_IZQ, IDENTIF, etc. son índices para relacionar las calles. No son relevantes para este trabajo.

Siempre es posible realizar búsquedas por cualquiera de estos campos, ya sea para localizar una dirección o una calle en particular. Para este trabajo sólo son relevantes los datos geográficos, ya que el mapaProxy a determinada escala consultará por las calles ubicadas en un bounding box.

Para la capa de comunas es análogo, sólo que no cuenta con tantas propiedades características de cada comuna y sí posee más información geográfica para optimizar búsquedas.

4.2.4. Consultas WFS

WFS (Web Feature Server) es un servicio estándar, que ofrece un interfaz de comunicación que permite interactuar con los mapas servidos por el estándar WMS (referirse a la sección de Antecedentes 2.1), como por ejemplo, editar la imagen que nos ofrece el servicio WMS o analizar la imagen siguiendo criterios geográficos. Las consultas WFS son utilizadas para interactuar con las características del mapa. Hay tres tipos principales fuentes de almacenamiento de los SIG:

- Bases de datos relacional.
- GML / XML o archivos XML nativos de bases de datos que contienen GML.
- Shape files.

Cuando un WFS recibe una solicitud GetFeature lo primero que hace es averiguar sobre qué datos debe interactuar. Luego descompone el filtro para generar la consulta según el tipo de datos almacenados. Si el tipo de datos es MySQL, la consulta debería ser una sentencia SQL, si se trata de una base de datos XML la consulta debe ser XPATH, etc.

WFS extrae el nombre de la característica (feature) del filtro y busca la configuración de esta característica particular en el archivo.

Un punto importante para las consultas WFS es que se tiene que saber los nombres de las características que contienen las coordenadas mínima y máxima (o bounding box de coordenadas).

A continuación un ejemplo de consulta para el área entre las coordenadas geográficas (72.102613,-30.212597), (-70.361859,-34.512517), (-72.102613,-30.212597), (-70.361859,-34.512517):

```
<?xml version="1.0" encoding="iso-8859-1"?>
<wfs:GetFeature outputFormat="GML2"
xmlns:gml="http://www.opengis.net/gml"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:ogc="http://www.opengis.net/ogc">
  <wfs:Query typeName="gps:calles">
    <wfs:PropertyName> gps:NOMBRE, </wfs:PropertyName>
    <wfs:PropertyName> gps:the_geom </wfs:PropertyName>
    <ogc:Filter>
      <ogc:BBOX>
        <ogc:PropertyName>thew_geom</ogc:PropertyName>
        <gml:Box>
          <gml:coordinates>
            -72.102613,-30.212597
            -70.361859,-34.512517
            -72.102613,-30.212597
            -70.361859,-34.512517
          </gml:coordinates>
        </gml:Box>
      </ogc:BBOX>
    </ogc:Filter>
  </wfs:Query>
</wfs:GetFeature>
```

Esta consulta pide las características gps:NOMBRE y gps:the_geom de la capa de la capa gps:calles dentro de la región metropolitana.

4.2.5. Test de Conexión a Servidor GIS

Para comprobar que el Servidor este funcionando sin problema y la información geográfica sea la correcta, basta con hacer un llamado al servicio de descripción de las features existentes a través de cualquier navegador:

```
http://urlServidor:8080/geoserver/wfs?service=WFS&request=GetCapabilities
```

Al accionar este link, se obtendrá un extenso documento XML en que la información importante es la siguiente:

```
<FeatureType>
  <Name>gps:calles</Name>
  <Title>calles_Type</Title>
  <Abstract>Generated from calles </Abstract>
  <Keywords>calles test</Keywords>
  <SRS>EPSG:4326</SRS>
  <LatLongBoundingBox minx="-70.69996571282091" miny="-
33.47858034038367" maxx="-70.57075435909162" maxy="-
33.365950489851215"/>
</FeatureType>
<FeatureType>
  <Name>gps:comunas</Name>
  <Title>comunas_Type</Title>
  <Abstract>Generated from comunas</Abstract>
  <Keywords>comunas</Keywords>
  <SRS>EPSG:4326</SRS>
  <LatLongBoundingBox minx="-70.70018005371094" miny="-
33.47858428955078" maxx="-70.57069396972656" maxy="-
33.36591720581055"/>
</FeatureType>
```

Esto demuestra que el servidor está funcionando y la información está cargada.

4.3. Servidor Proxy de Información

Como se describió en el capítulo 3.1.2. el servidor Mapa Proxy está encargado de procesar previamente la información geográfica que recibirá el visualizador 3D y así disminuir la carga de la aplicación Flash.

La forma más tradicional de obtener la información en aplicaciones Flash es a través de Webservices. Como esta aplicación es sencilla no se justificaba la instalación de un servidor empresarial o inclusive un servidor Apache. Por esto se estudió una alternativa más liviana llamada Jetty.

Jetty es un servidor HTTP y un contenedor de Servlets escrito en Java. Se publica como un proyecto de software libre bajo la licencia Apache 2.0. Debido a su pequeño tamaño, Jetty puede ser implementada para ofrecer servicios Web en una aplicación Java estándar.

Actualmente la versión más reciente de Jetty es la 6.1., pero es casi tan pesada como un servidor Apache, por esto se utilizó la versión 4.1. que posee menos funcionalidades, pero satisface los requerimientos.

A continuación un cuadro comparativo de las versiones de Jetty

Version	Java	HTTP	Servlet	JSP	Status	Notes
Jetty-6.1	1.4-1.5	HTTP/1.1 RFC2616	2.5	2.1 or 2.0	Stable	Async SSL, AJP, cometd, testing
Jetty-6	1.4-1.5	HTTP/1.1 RFC2616	2.5	2.1 or 2.0	Deprecated	Continuations, IOC, NIO, dynamic buffers, smaller, faster, better
Jetty-5.1	1.2-1.5	HTTP/1.1 RFC2616	2.4	2.0	Stable	J2EE 1.4 Compliance tested, optimizations, geronimo integration.
Jetty-5.0	1.2-1.4	HTTP/1.1 RFC2616	2.4	2.0	Deprecated	Schema, JettyPlus
Jetty-4.2	1.2-1.4	HTTP/1.1 RFC2616	2.3+	1.2	Mature	JettyPlus
Jetty-4.1	1.2-1.4	HTTP/1.1 RFC2616	2.3	1.2	Deprecated	JAXP1.1, AJP13(mod_jk)
Jetty-4.0	1.2	HTTP/1.1 RFC2616	2.3	1.2	Deprecated	
Jetty-3.1	1.2	HTTP/1.1 RFC2068	2.2	1.1	Ancient	JMX
Jetty-3.0	1.2	HTTP/1.1 RFC2068	2.2	1.1	Fossilized	
Jetty-2.4	1.1	HTTP/1.0 RFC1945	2.1	1.0	Legendary	
Jetty-1.0	1.0	http/1.0 RFC1945	-	-	Mythical	

Tabla 1: Comparación versiones Jetty

Para iniciar el servicio que utiliza Jetty sólo hay que crear una aplicación que cargue clases específicas para el funcionamiento y publique los servlets correspondientes.

4.3.1. Procesamiento por capas

Por cada capa existente, se implementó un servlet capaz de entender la información. Como se mencionó en el capítulo 4.3, cada capa es distinta de otras y posee información propia ingresada por el cartógrafo creador.

Cada servlet implementa los métodos doGet y doPost, que son los encargados de recibir los parámetros de consulta.

Dentro de la petición llegan los parámetros que definen el bounding box de consulta, tales como longitud y latitud tanto máxima como mínima del recuadro. El

layer (capa) toma estos datos y los traduce a una consulta WFS que filtra por la característica **gps:the_geom**.

Un extracto del código de la capa de comunas es:

```
String query =
    "<ogc:Filter xmlns:ogc='http://ogc.org/' xmlns:gml='http://www.opengis.net/gml'">"
    + "<ogc:BBOX>"
    + "<ogc:PropertyName>the_geom</ogc:PropertyName>"
    + "<gml:Box srsName='http://www.opengis.net/gml/srs/epsg.xml#4326'">"
    + "<gml:coordinates>"+minLon+", "+minLat+"
    +maxLon+", "+maxLat+"</gml:coordinates>"
    + "</gml:Box>" + "</ogc:BBOX>" + "</ogc:Filter>";
```

Una vez creada la consulta se realiza la petición de datos al servidor GIS. Se obtiene un documento XML con la información y se procede a convertir los puntos geográficos a coordenadas para el rendering.

4.3.2. Conversión de geometría espacial a pixeles

La conversión de geometría fue simplificada, observando la cercanía de la cámara al terreno, es posible considerar los cálculos como lineales.

Como se sabe, la Tierra tiene una forma aproximadamente esférica, esta corrección debe ser considerada al convertir datos geográficos. Pero ya que el trabajo se centrará en Santiago, se puede suponer el terreno como plano, pues los rendering serán cercanos al terreno.

Para traducir las coordenadas geográficas en pixeles, se realiza el cálculo siguiente:

```
double originalLat = Double.parseDouble(xy[1]);
double originalLon = Double.parseDouble(xy[0]);
double lat = FACTOR * (originalLat - minLat);
double lon = FACTOR * (originalLon - minLon);
```

Como se observa, son sólo operaciones lineales. La variable FACTOR depende de la escala, ya que da la separación entre puntos

4.3.3. Generalización de puntos

La información geográfica obtenida desde el servidor puede ser demasiado detallada, lo que genera una gran cantidad de puntos, traducándose en polígonos extremadamente detallados geoméricamente en el Visualizador. Un polígono de

estas características hace que el proceso de rendering sea más lento innecesariamente, ya que tanto nivel de detalle no es perceptible en una interfaz como la trabajada.

A partir de este detalle es que se aplican técnicas de generalización, para reducir el número de puntos sin perder detalle.

El algoritmo es el siguiente:

- Se revisa cada punto del polígono. Se asume que los puntos están en orden secuencial.
- Si se tiene un punto y el siguiente está a una distancia menor a un radio (dependiendo de la escala) estos se eliminan y no serán dibujados.

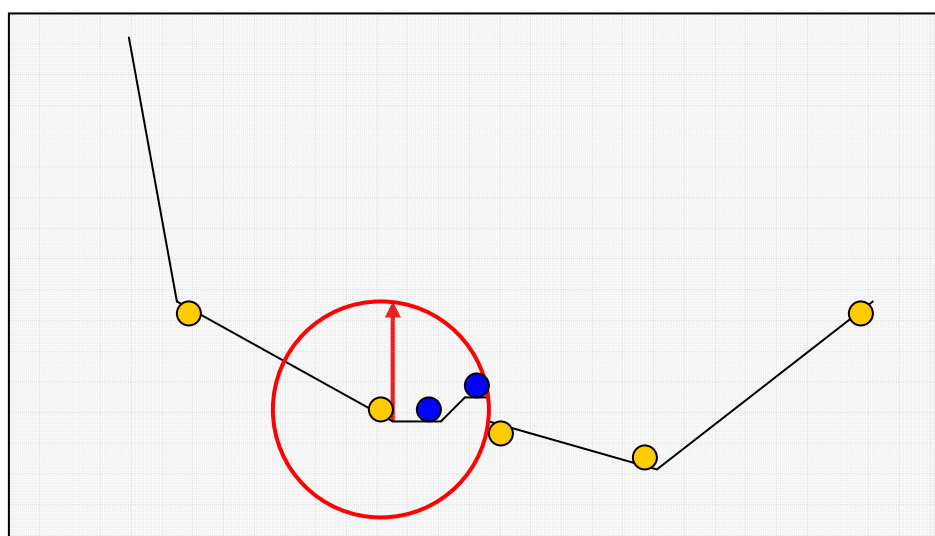


Figura 14

En la Figura 14, se aprecia que según el algoritmo, los puntos que se eliminarán serán los azules.

El siguiente cuadro (Tabla 2) muestra el número de puntos que originalmente conformaban las comunas y el resultado de filtrarlos, por ejemplo, con un radio de 20 píxeles. La determinación del número de píxeles que tendrá el radio se realiza en forma cualitativa, es decir se observa que las figuras no pierdan la forma original. No hay un algoritmo fijo, ya que dependiente del detalle con que se hayan generado los datos geográficos.

Comuna	Original	Generalizado
CONCHALI	131	63
INDEPENDENCIA	127	54
PROVIDENCIA	158	75
RECOLETA	166	69
SANTIAGO	278	115
NUNOA	221	92

Tabla 2: Puntos Filtrados

El resultado es el esperado. El número de puntos disminuyó en más de un 50% (visible en la Figura 16). Entre más grande el radio, menos puntos aparecerán.

El mayor problema de este algoritmo es que al eliminar puntos, se pierde precisión en los bordes de los polígonos, aunque es aceptable, ya que el rendering de objetos aumenta bastante su velocidad.

A continuación la comparación entre una imagen de la vista general de las comunas de Santiago generada con GeoServer (Figura 15) y otra por el visualizador 3D (Figura 16).

En la imagen del visualizador se aprecian “manchas” azules dentro del mapa, estos son causados por la generalización, ya que las comunas son definidas por polígonos independientes y algunos de los puntos que definen los bordes fueron descartados. Este problema podría evitarse si se verificaran los puntos comunes entre comunas, que a cambio disminuiría el rendimiento.

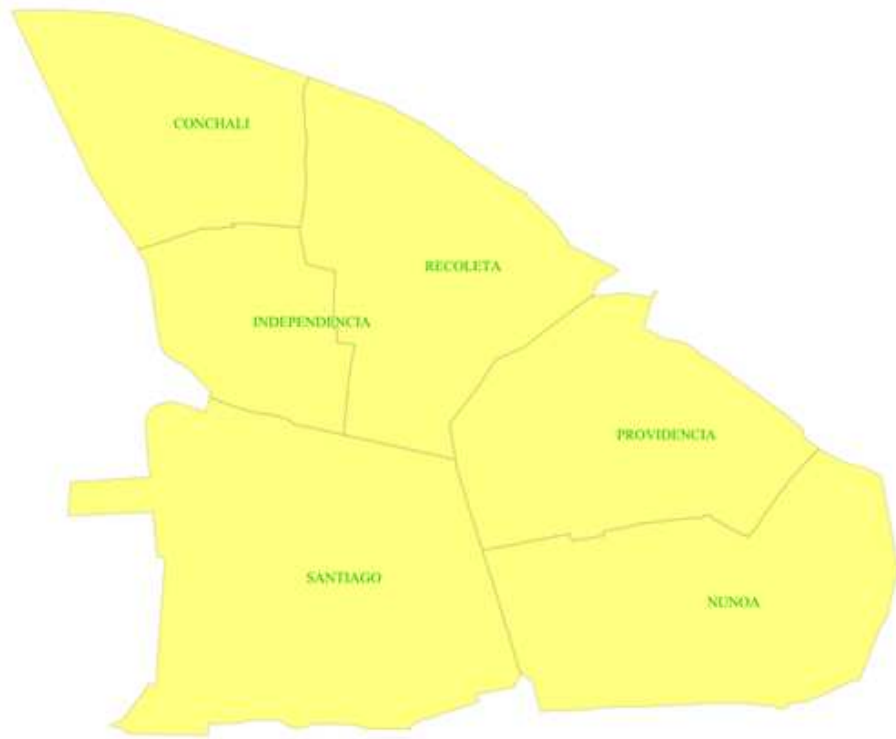


Figura 15

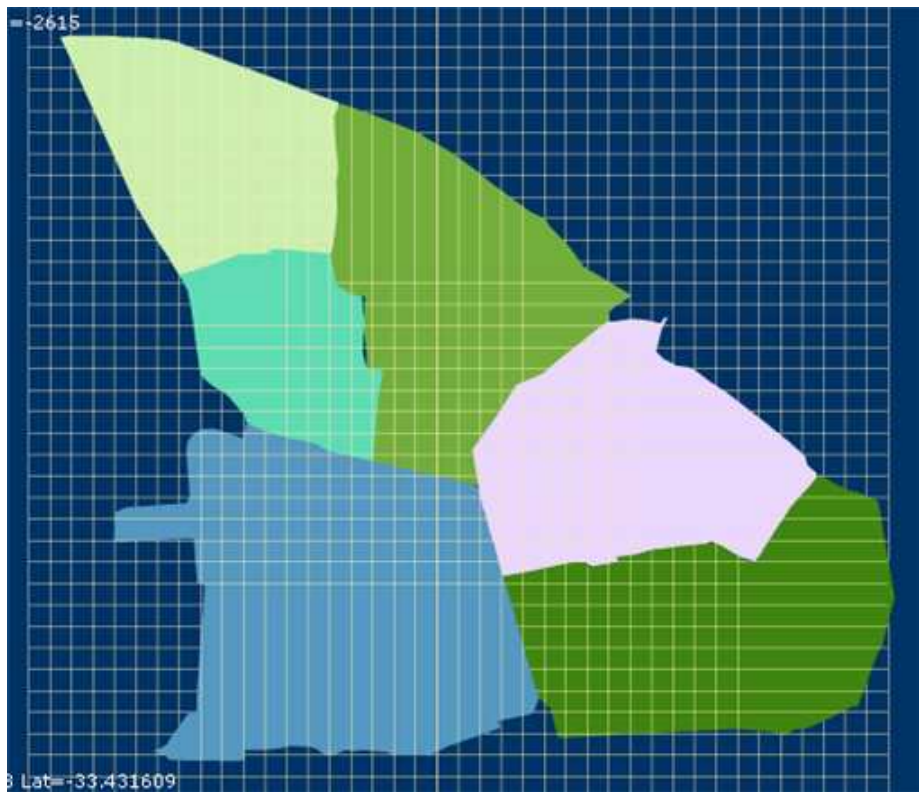


Figura 16

4.4. Interfaz 3D

Este capítulo trata la forma en que se coordinan la parte visual con el manejo de datos y la petición al módulo Mapa Proxy.

4.4.1. Implementación de controles básicos

Los controles de la aplicación tratan de emular el comportamiento de Google Earth, ya que es bastante natural y permite una navegación fluida sobre un terreno en 3 dimensiones (en verdad son dos dimensiones, un plano proyectado).

Antes de implementar los comportamientos de la interfaz, se definieron clases auxiliar (tipo Helpers) que son capaces de detectar movimiento del Mouse, cantidad de movimiento, presión del botón del Mouse (Flash sólo puede utilizar el botón primario) y presión del teclado. Estas clases implementan listeners de eventos y son del tipo estáticas, visibles globalmente por la aplicación.

Antes de comenzar hay que saber que la librería WireEngine 3D posee un objeto llamado Camera3D que dentro de sus funciones disponibles ofrece desplazamiento con respecto al sistema de referencia global o propio de la cámara, al igual que la rotación.

Los cálculos de las transformaciones a aplicar se realizan a través de la clase Matrix4x4 que representa una matriz de 4x4 e implementa las funciones de escalamiento, translación, rotación, entre otros.

El primer comportamiento implementado fue el *panning* o desplazamiento (Figura 17), que se refiere al movimiento horizontal o rotatorio de la cámara. Panning deriva de “panorama” o visión panorámica.

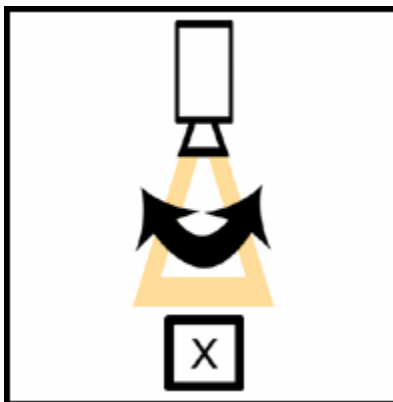


Figura 17

Para calcular la cantidad de desplazamiento (o rotación), se toma la diferencia de píxeles entre la posición actual del Mouse y la anterior, con esto ya es posible

realizar cambios en la posición de la cámara, atenuado por un factor. A continuación un extracto del código:

```
public function checkCam(mouseX : Number, mouseY : Number) : void {
    var difX : Number = -MouseMove.getXmouse(mouseX);
    var difY : Number = MouseMove.getYmouse(mouseY);
    if(MouseObserver.isPressed()){
        if(Key.isDown(Keyboard.CONTROL)){
            cam.rotateLocalX(-0.002*difY);
        }
        else if(Key.isDown(Keyboard.SHIFT)){
            cam.rotate(0,0,-0.005*difX);
        }
        else {
            var factor_altura : Number = (cam.position.z)*0.0025;
            var factor : Number = -0.4*factor_altura;

            cam.moveLocal(factor*difX,factor*difY);
        }
        ListenerParamenters.checkCam(this.cam);
    }
}
```

Como se observa en el código, la cámara (variable cam) ofrece funciones que realizan el panning correspondiente. Para hacer el control más dinámico, el comportamiento del Mouse cambia al mantener presionadas las teclas CONTROL o SHIFT.

Los valores de los factores aplicados fueron obtenidos empíricamente, es decir a prueba y error, observando el comportamiento de la aplicación a 50 Frames por Segundos (FPS). El valor de los 50 FPS es constante, ya que es definido por Flash como parámetro.

Una parte interesante es el cálculo del factor_altura. Esta variable logra que a mayor altura, la cámara se desplace a mayor velocidad y a menor altura, menor velocidad. La cámara siempre pasa por la función checkCam, que determina si no se ha desplazado fuera de los rangos que la escala permite.

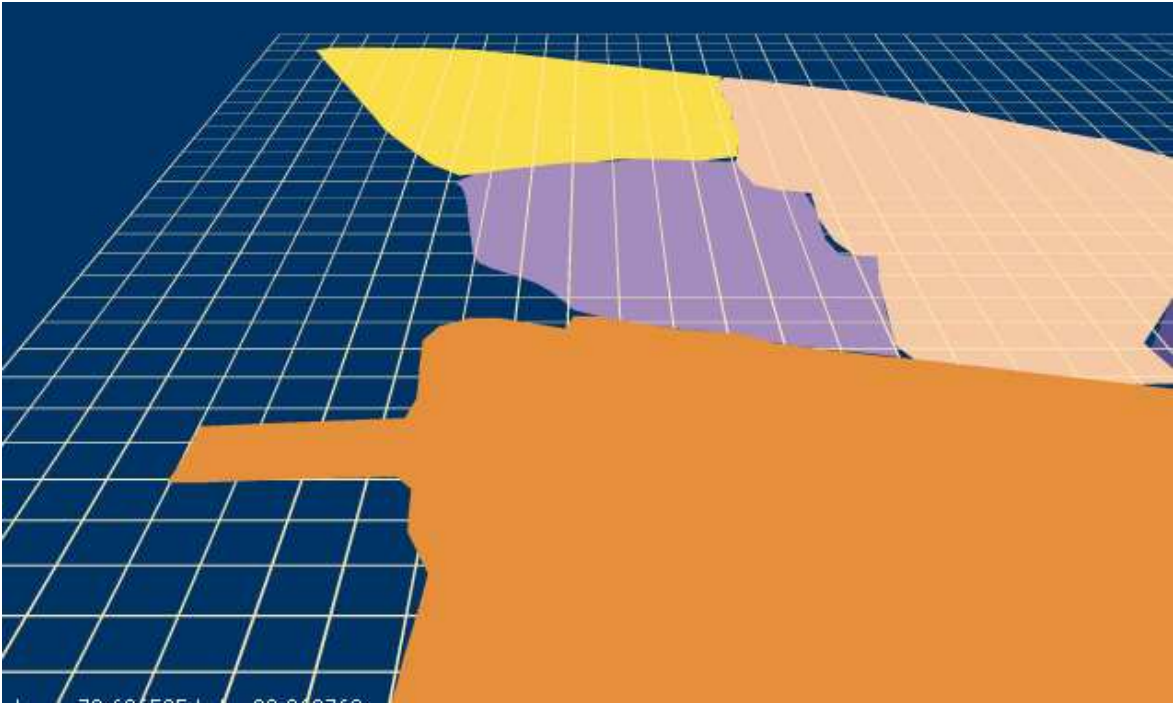


Figura 18

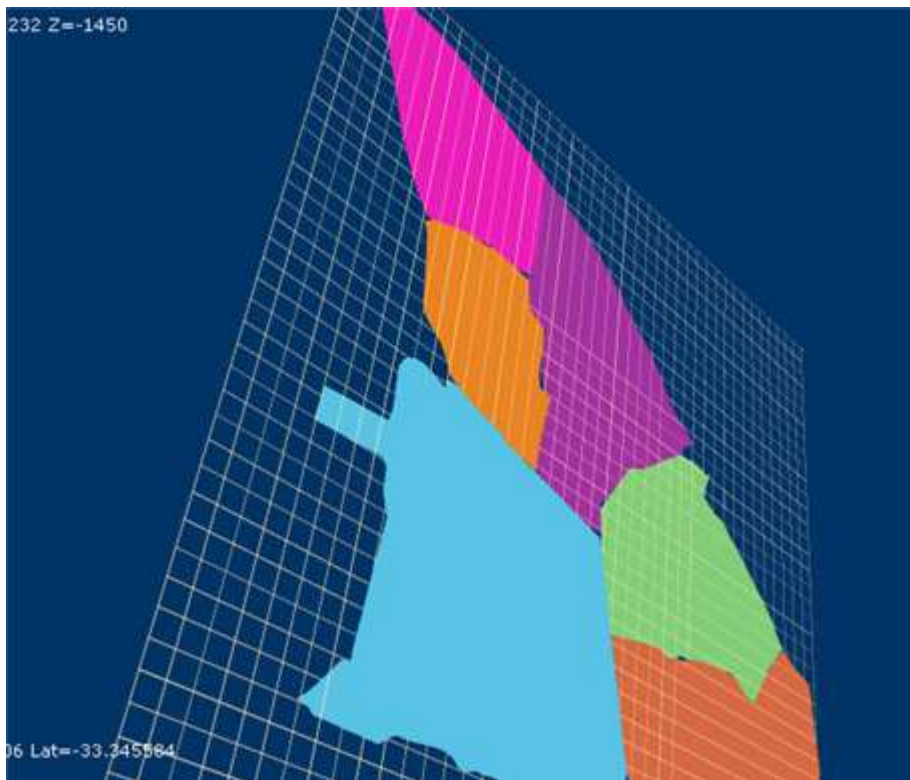


Figura 19

La Figura 18 muestra el mapa de las comunas rotado sobre el eje X. La Figura 19 muestra un zoom (alejamiento) y rotación con respecto a la cámara.

El resumen de las funciones se aprecia en el siguiente cuadro:

Acción	Mouse	Teclado	Resultado
Panning de cámara	Mover Mouse en cualquier eje y presionar Botón	No	El mapa se desplaza como si fuese arrastrado
Rotación en eje X de la cámara	Mover Mouse arriba y abajo y presionar Botón	Presiona CONTROL	La cámara da el efecto de mirar hacia arriba y hacia abajo.
Zoom	No	RePag y AvPag	Cámara se acerca y se aleja del objetivo
Rotación en eje Y de la cámara	Mover Mouse en cualquier eje y presionar Botón	Presiona SHIFT	La cámara da el efecto de girar en el centro.

Con estos controles básicos, ya es posible lograr una navegación fluida.

Cabe notar que el zoom no es realmente un enfoque óptico, sino que la cámara se va a acercando al objetivo. Se realizó de esta forma, ya que la perspectiva se perdía por limitaciones de la librería.

4.4.2. Implementación de llamadas a servidor Proxy manejando proyecciones y escalas

La implementación se realizó de la siguiente forma. Cuando la cámara está en una determinada escala, el visualizador realiza una petición GET al Mapa Proxy. Cuando se obtiene el XML con los puntos geográficos, estos se dibujan y se guardan en la aplicación asociados a la escala y un bounding box que los contiene, dentro de un arreglo.

Cuando la cámara se encuentra entre las distancias de transición de escala (Figura 13) se pide la nueva información con la nueva escala y se realiza el rendering de las dos escalas al mismo tiempo (se sobreponen), si la escala siguiente es menor, la información obtenida tendrá mayor detalle, ya que la generalización será menor. Esta información también será guardada en el arreglo de información geográfica.

Antes de realizar una consulta Web, siempre se revisa si la información correspondiente a la escala y bounding box no existen previamente en memoria.

4.5. Integración del Sistema GPS.

La integración del sistema GPS es bastante sencilla, ya que ha sido simplificada a la identificación de un usuario y luego la obtención de posiciones. La condición previa es tener creado un usuario dentro del sistema y que éste tenga asociados distintos móviles.

4.5.1. Registrar sesión de usuario

El primer paso es llamar al Webservice de login y pasar dentro de la petición un XML con el nombre de usuario y su contraseña. Como respuesta se obtendrá otro XML que contendrá un Token (llave única) generado por el servidor que servirá para las transacciones.

El XML enviado es del tipo

```
<params>
  <usuario>mmunoz</usuario>
  <password>mmunoz</password>
</params>
```

El resultado será:

```
<USUARIO nombre_usuario="Marcelo M." id_usuario="5566"
username="mmunoz">

<SESION>A6CCCECB3493C2F10237171E21FB45BADC8A40983BE24DD3BE5188C3940B2652
</SESION>
  <EMPRESA razon_social="TASTETS SYSTEM LTDA:" id_empresa="45"/>
  <PAIS huso_horario="-3" idioma="es_CL" nombre_pais="Chile"/>
  <PERFILES>
    <INFORME_EVENTOS valor="1"/>
    .....
  </PERFILES>

</USUARIO>
```

El tag SESSION contiene el Token necesario para la futura interacción con el servidor. Este Token cambia cada vez que se realiza un nuevo ingreso del usuario.

4.5.2. Actualización de posiciones

Para actualizar constantemente las posiciones dentro del visualizador 3D es necesario crear un Thread que realice la petición al servidor cada cierto tiempo.

La alternativa de que el visualizador sea notificado cada vez que un móvil cambie su posición no es posible, ya que Flash carece de las herramientas para hacerlo (Sockets, Manejo de puertos, etc.).

Dentro de Flash tampoco existe el concepto de Thread, pero sí el de eventos asíncronos y Timers, por lo que la implementación consta de un Timer (reloj dentro de flash que ejecutará una función después de un número determinado de milisegundos) que cada 30 segundos envía una petición al servidor.

La petición tiene un documento XML con el Token de sesión, un ID de consulta y la lista de móviles pertenecientes al usuario. El ID de consulta se obtiene cada vez que se realiza una nueva consulta. Este ID es utilizado para traer sólo las posiciones realmente nuevas, es decir las mayores a ese ID. Los números identificadores de los móviles están contenidos dentro del documento XML obtenido al ingresar al sistema GPS.

```
<params>
  <sesion>
AA963202B688FB4C46E5D29BA91019B02F9453E948430E5FF3F436EDB6D67340
  </sesion>
  <id_consulta>1460275675</id_consulta>
  <lista_moviles>896,833,11810,797,16874,16871,16872,16873,13130,16875,
16876,12750,36400,777,13170,792,30331,30332,829,9150</lista_moviles>
</params>
```

El resultado es un documento XML con las posiciones nuevas para cada móvil.

```
<EVENTOS>
  <EVENTO id_consulta="618567572" id_evento="4" velocidad="0.0"
fecha="26-07-2007 10:41:22" evento="Posicion cada 00:10:00 hh:mm:ss"
latitud="-33.420717" id_movil="777" rumbo="0.0" longitud="-
70.607617"/>
  <EVENTO id_consulta="618447665" id_evento="4" velocidad="0.0"
fecha="03-04-2007 13:20:06" evento="Posicion cada 00:10:00 hh:mm:ss"
latitud="-33.421017" id_movil="792" rumbo="0.0" longitud="-
70.607317"/>
.....
</EVENTOS>
```

Los demás datos pueden ser ignorados para este trabajo.

Cuando las nuevas posiciones llegan se transforman a pixeles y se almacenan en un arreglo de objetos móviles, luego se representan en el mapa por una figura. Al momento de escribir esta memoria la figura utilizada es un triángulo simple.

5. Discusión y conclusiones

5.1. Resultados

Como resultado de este trabajo se obtiene una aplicación computacional que puede:

- Generar una representación vectorial en 3 dimensiones del mapa de Santiago y sus calles a medida que el usuario se acerca a una posición determinada.
- Navegar el mapa resultante puede en forma natural con el mouse y el teclado. Realizar distintas maniobras como acercamientos, rotaciones y desplazamientos.
- Observar los móviles en tiempo real de cualquier usuario de la empresa Tastets System Ltda. que están ubicados dentro de Santiago.
- Utilizar (teóricamente) cualquier tipo de servidor GIS que tenga el estándar de consulta WFS.
- Simplificar los objetivos a ser visualizados para dar mayor rendimiento al rendering.
- Utilizarse en cualquier navegador que posea Flash 9.0 independiente del sistema operativo.

5.2. Conclusiones

La aplicación realizada intenta emular el funcionamiento, aunque sea de una manera básica, de una aplicación como Google Earth, lo que es bastante ambicioso, ya que están involucrados muchos sistemas (componentes) asociados.

El principal desafío de este trabajo es la integración de tres sistemas y crear la arquitectura que logre la interacción entre un servidor de Mapas, una aplicación Flash y un sistema pre-existente como lo es el de GPS.

El trabajo con servidores de mapas no es fácil. Los estándares de consulta, como WFS, son complejos y no existen demasiados ejemplos si se requiere preguntar por algo complejo. WFS es útil para consultas simples.

La decisión de implementar esta aplicación en Flash fue acertada. La compatibilidad comprobada entre diversos navegadores y sistemas operativos,

permite que el desarrollo sea portable y posible de ejecutar rápidamente sin grandes requerimientos. La liberación de Flex SDK para un desarrollo gratuito y la existencia de IDEs libres de gran potencia logran que uno no dependa de una empresa en particular y permite un desarrollo de bajo costo. La aparición de ActionScript 3.0 como lenguaje similar a Java permite aplicar programación orientada a objetos (OOP) que expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

La existencia de librerías 3D en ActionScript 3.0 facilitó mucho el desarrollo y cálculo. Es algo equivalente a utilizar librerías como OpenGL, que internamente poseen todas las herramientas para lograr un rendering preciso y consistente, sin tener que preocuparse por implementar los cálculos correspondientes.

El rendimiento de las aplicaciones Flash se ve afectado, en mayor medida, por procesos pesados de cálculo y rendering. Una aplicación que responde lenta frente al usuario, terminará por hartarlo. La aplicación de algoritmos de generalización por parte de un agente externo aumentó considerablemente el rendimiento, entre tiempo de respuesta y velocidad de rendering.

La integración con un sistema externo, como la plataforma GPS, obliga a tener restricciones, como por ejemplo tener una cuenta de usuario creada con anterioridad y tener móviles asociados.

Dada la extensión del trabajo, no fue posible cumplir cabalmente con todos los requerimientos no funcionales propuestos, como la usabilidad. Se puso énfasis en el rendimiento, aplicando técnicas y algoritmos para aumentar el desempeño, tales como la generalización de puntos y utilización de la memoria para almacenar la información geográfica para disminuir las llamadas a los servidores. El desempeño es la característica por la que se desea que la aplicación destaque sobre las aplicaciones de mapas basadas en imágenes fijas.

La portabilidad es satisfecha “gratuitamente” por Flash, que posee una portabilidad comentada en el capítulo 2.

La confiabilidad de la información se vio afectada moderadamente por el procesamiento de los datos por algoritmos que favorecen el rendimiento, aunque no es demasiado importante, ya que el fin de la aplicación es entregar sólo información referencial.

La usabilidad sólo se logró en una pequeña medida emulando los controles de Google Earth, pero como se comentará, en la siguiente sección, faltan herramientas.

La correctitud y la robustez se lograron en la medida deseada a través de la implementación de la comunicación a través de los Webservices en todos los componentes de la arquitectura.

5.3. Trabajos futuros

La aplicación desarrollada hasta el momento es un prototipo inicial que debe ser mejorada en los siguientes aspectos:

- La interfaz, aún es muy básica. El rendering logrado es el más básico permitido por la librería. Podrían ser agregados imágenes de materiales al terreno, tales como concreto, tierra, pasto entre otros y mejorar la representación de los móviles. La información representada es sólo gráfica y los nombres o descripciones son olvidados.
- Agregar herramientas que faciliten la obtención de información por parte del usuario, como reglas, buscador de direcciones, seguimientos de móviles, entre otros. Además falta un mayor feedback para el cliente.
- La librería WireEngine 3D posee varias configuraciones para realizar el rendering, es posible agregar luces, aplicar optimizaciones, como Frustrum Culling que elimina los objetos que no serán dibujados por estar ocultos a la cámara, y mejorar la proyección, para obtener una mejor calidad de rendering. Por ahora sólo se utiliza la configuración por omisión, que realiza un rendering sin luces y con proyecciones lineales.
- La cartografía podría ser mejorada en cuanto a la cantidad de datos y agregar más áreas geográficas, para no abarcar solamente la región metropolitana.
- Depurar el programa en profundidad, ya que se pueden producir errores dentro de la aplicación, como problemas de conexión con el servidor de Mapas. La información geográfica se almacena en memoria, pero ésta no tiene límites, por lo que la aplicación después de usarse por largo tiempo puede “colgarse”.
- Agregar más algoritmos de generalización en el mapa Proxy. El algoritmo de eliminación de puntos por cercanía no es suficiente y puede provocar imprecisiones en líneas o elementos muy complejos.
- Optimizar el servidor GIS. La instalación inicial se realizó por omisión y el desempeño obtenido no es de los mejores. Si la fuente de datos se cambia a base de datos el rendimiento mejora automáticamente. Si agregamos un cambio de parámetros de número de conexiones, manejo de memoria, etc. puede mejorar aún más.
- Aprovechar más las consultas WFS, estas permiten obtener información más precisa sobre el terreno, las calles, ubicación de un sector, entre otros. Esta información podría ser asociada a los móviles, por ejemplo, para poder consultar que móvil de la flota está más cerca de la calle Beaucheff.

6. Bibliografía

1. Dr. Shunji Murai.
Journal Selper. "SIG Manual Base Vol.1: Conceptos Fundamentales."
Universidad de Tokio. 1999
2. Paul Ramsey
"The State of Open Source GIS"
Refractions Research Inc. Technical Report. 2006
3. David M. Mark, Nicholas Chrisman, Andrew U. Frank, Patrick H. McHaffie, John Pickles. "The GIS History Project"
University at Buffalo. Department of Geography. 1999
4. Mark, D. M., Initiative 13,
"User Interfaces for Geographic Information Systems," Closing Report.
Santa Barbara, CA: National Center for Geographic Information and Analysis,
Closing Report Series. 1996
5. Richards, T.B., Charles Croner, Gerard Rushton, Carol Brown, Littleton Fowler.
"Geographic Information Systems and Public Health: Mapping the future".
Public Health Reports 1999.
6. Chris Holmes and Allan Doyle, Mick Wilson
"Towards a Free and Open Source Spatial Data Infrastructure"
<http://www.eogeo.org/Projects/gsdi8paper-folder>
Fecha de Acceso: 14-09-2007
7. Prehco Project
"The use of GPS and GIS technologies to increase the quality of the sample and decreasing supervision costs: the experience of the prehco Project."
University of Puerto Rico and Center for Demography and Ecology of the
University of Wisconsin – Madison. 2003
8. Matthias Blech, Chris Holmes
"Structure of RelationshipDataStore"
Empresa de soluciones soluciones moviles y GIS, ConTerra.
9. Gabriel Ortiz
Web de recursos de usuarios de sistemas de información geográfica.
<http://www.gabrielortiz.com/>
Fecha de Acceso: 10-10-2007

10. Quantum GIS: Open Source Geographic Information System

<http://qgis.org/content/view/24/104/>

Fecha de Acceso: 21-11-2007

11. Open Source Flash

<http://osflash.org/>

Fecha de Acceso: 21-11-2007

12. Funcionamiento de un navegador GPS

<http://guiabasica.com/hardware/navegadores-gps-4.html>

Fecha de Acceso: 21-11-2007

13. Estadísticas Flash

http://www.adobe.com/products/player_census/flashplayer/

Fecha de Acceso: 21-11-2007

14. Potencialidades del GIS 3D y los Modelos Urbanos Interactivos

Arq. Javier Alberto Martínez.

Grupo GiS, Facultad de Arquitectura Planeamiento y Diseño, UNR. Rosario, Argentina. 2000

15. The Three Dimensional Visualization & Analysis of Geographic Data

http://maps.unomaha.edu/Peterson/gis/Final_Projects/1996/Swanson/GIS_Paper.html

Fecha de Acceso: 21-11-2007