CEIG'09, San Sebastián, Sept. 9-11 (2009)
C. Andújar and J. LLuch (Editors)

Técnicas Multirresolución para Visualización de Información Vectorial 3D en SIG

Rafael Gaitán, Javier Lluch, Francisco Abad, M. Carmen Juan

Abstract

La representación de datos vectoriales (carreteras, lagos, ciudades, etc) es una parte fundamental en los sistemas de información geográfica, tanto para visualizar la información como para procesos de análisis. La gran cantidad de información adquirida en formato vectorial provoca que, en la mayoría de los casos, la visualización sea masiva o incluso que no quepa toda la información en memoria principal. En este trabajo se presentan algunas técnicas para la representación eficiente de datos vectoriales en 3D. El trabajo se centra en las técnicas utilizadas para datos vectoriales de tipo punto y plantea su extensión para polilíneas y polígonos. Abarca desde las estructuras de datos multirresolución empleadas y los algoritmos de generación hasta la visualización en sistemas de información geográfica 3D.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.6]: Methodology and Techniques–Graphics data structures and data types—Computer Graphics [I.3.8]: Applications—Information Systems [H.0]: General—

1. Introducción

Los recientes avances en diseño, adquisición y simulación 3D han provocado que los conjuntos de datos sean cada vez mayores, tanto que incluso hay casos en los que se excede el tamaño de la memoria principal y por lo tanto también se exceden las capacidades de dibujado del hardware gráfico actual. Debido a esto, es necesario hacer uso de técnicas como la multirresolución y la simplificación para poder reducir la cantidad de geometría a dibujar.

Aún así los modelos multirresolución necesitan trabajar con conjuntos de datos que no excedan el tamaño de la memoria principal. Para resolver el problema nuevos modelos se han desarrollado capaces de almacenar escenas en memoria secundaria (*out-of-core*). Estos modelos se pueden visualizar de manera interactiva haciendo simplificación de geometría dependiente de la vista. Las inmediatas aplicaciones de este tipo de modelos van desde la visualización de grandes bloques de terreno, pasando por videojuegos, simulación o incluso en dispositivos móviles donde los recursos aún son más limitados.

La información geográfica vectorial es una pieza fundamental en los Sistemas de Información Geográfica(SIG). Dicha información puede ser almacenada, procesada y consultada haciendo uso de bases de datos. Cada punto, línea o polígono se codifica como una entidad geográfica como pueden ser carreteras, edificios, vegetación, ríos, fronteras de países, etc. Una de las ventajas de usar información vectorial es que la precisión de los datos es independiente del punto de vista.

Cuando nos trasladamos a la visualización 3D, esa información puede provocar que no se pueda discernir que se está representando, al dibujar todos los elementos de forma masiva. También podemos encontrarnos con que la gran cantidad de información produzca problemas de rendimiento y de memoria. Debido a esto se hace necesario el uso de técnicas de paginación (*out-of-core*) y de multirresolución para resolverlos.

El trabajo que se presenta es un desarrollo para conseguir un conjunto de herramientas para la visualización multirresolución de información geográfica vectorial aplicado concretamente a puntos, líneas y polígonos. El sistema que se presenta es capaz de generar una estructura de datos con capacidad para ser consultada y extraer múltiples niveles de detalle de un conjunto de datos. La extracción de información se realiza en base a un área de extensión y un factor de profundidad o de detalle.

El modelo multirresolución actual ha sido integrado en nuestra librería de representación de terrenos paginados, aunque potencialmente puede ser ampliado para bases de da-

© The Eurographics Association 2009.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

DOI: 10.2312/LocalChapterEvents/CEIG/CEIG09/065-073

tos geográficas o de manera general en aplicaciones SIG. El trabajo desarrollado hasta ahora se limita a la multirresolución de puntos aunque se dan las bases para su extensión a polilíneas y polígonos.

El documento se estructura como sigue. Primero se hace un breve repaso del trabajo al conjunto de trabajos relacionados en técnicas multirresolución, representación de información vectorial y técnicas de *clustering*. A continuación se desarrolla en profundidad el modelo multirresolución para puntos haciendo hincapié en las estructuras de datos, la generación de éstas y por último en la visualización. Finalmente mostramos algunos resultados, las conclusiones y cuáles van a ser las líneas a seguir en nuestro trabajo.

2. Trabajos relacionados

La representación multirresolución se usa habitualmente para obtener diferentes niveles de detalle. El uso del término multirresolución se refiere a la precisión (o nivel de detalle) de una geometría, o lo que es lo mismo, a la densidad (tamaño y número) de las celdas que componen la malla, en el caso de multirresolución de modelos polígonales. Una representación multirresolución de un objeto nos permite obtener diferentes aproximaciones del objeto, por ejemplo la superficie exterior de un objeto sólido, o un parche de terreno representado en forma de malla geométrica, o un conjunto de puntos que definen el color y la forma.

Trabajos relacionados que tratan de dar solución al problema mediante la representación de puntos multirresolución se pueden encontrar en [RL00,DVS03]. El primero crea una jerarquía de puntos, que permite transimitir los datos por red, obteniendo desde un primer momento una aproximación al objeto que se va refinando progresivamente. El segundo también hace uso de una jerarquia de puntos, pero el recorrido de ellos se realiza de manera secuencial, permitiendo implementar la mayor parte del algoritmo en GPU.

La simplificación dependiente de la vista [ESV99] permite realizar cambios en la jerarquía multirresolución dependiendo de los parámetros del observador, iluminación y movimiento. A cada *frame* el sistema adapta la estructura de la malla para obtener el nivel de detalle adecuado. El principal problema que presentan estas estructuras de datos es que incrementan el tamaño de los datos y además requieren que se mantengan en memoria principal.

Las técnicas de paginación (o *out-of-core*) resuelven estos problemas para conjuntos de datos mayores que la memoria principal [EsjC00]. La idea de estos sistemas es almacenar los datos geométricos en el disco. Posteriormente el modelo se preprocesa para obtener un conjunto de estructuras de datos de tipo árbol con las geometrías, preparadas específicamente para tener un rendimiento eficiente de entrada/salida. Estos árboles son estructuras jerárquicas multirresolución. Durante la navegación, esta técnica mantiene todos los árboles de geometría en el disco. En memoria principal almacena únicamente aquellos activos que son necesarios para visualizar el nivel de detalle actual, más algunos árboles precargados que probablemente van a ser necesarios en un futuro cercano.

Las entidades geográficas se expresan a menudo como vectores. Los diferentes tipos de entidades geográficass existentes se expresan por diferentes tipos de geometría, en general puntos, líneas o polígonos. Cada una de estas geometrías están enlazadas a una fila de una tabla en una base de datos geográfica. Esta fila contiene una serie de atributos y la geometría que lo representa.

La representación de información vectorial en 3D puede ser llevada a cabo de dos maneras. La primera consiste en *rasterizar* los vectores y pegarlos como textura al terreno. La segunda representa la información usando primitivas geométricas.

Autores como [OK02, BN08] hacen uso de la *rasterización* de las primitivas, permitiendo fácilmente generar niveles de detalle siguiendo la misma estrategia de subdivisión del terreno. El principal problema que presenta este tipo de aproximación es la dificultad de edición y modificación de los datos de manera interactiva.

En [ASZ07] se presenta una estrategia de representación de primitivas geométricas (por ejemplo haciendo uso de primitivas OpenGL), sin embargo tiene el problema de incrementar demasiado el conjunto de primitivas a dibujar cuando se quieren evitar problemas con el contorno del terreno. Otros autores utilizan modelos basados en niveles de detalle [WKW*03, AAJ06, SGK05]. En [MS07] hacen uso de volúmenes y del *stencil buffer* para representar correctamente los vectores sobre el terreno.

En [TTZ*08a, TTZ*08b] se explica un framework completo (*OSGVirtualPlanets*) para visualización de terreno 3D y de representación de información vectorial 3D. La librería propuesta para representación de terreno es capaz de manejar múltiples fuentes de datos rasterizadas y de elevación sin embargo en la representación vectorial no aplican ninguna técnica multirresolución, tan solo hacen uso de algunas técnicas de recortado para disminuir el conjunto de datos.

Las técnicas de *clustering* [Ber02] se llevan utilizando desde hace bastante tiempo para realizar análisis de los datos geográficos. Por ejemplo en [IhA03] hacen uso de un algoritmo específico para encontrar grupos de objetos geográficos de manera eficiente. También se han llevado a cabo trabajos para manejar conjuntos de datos masivos como se puede apreciar en [SCZ98]. En *clustering* jerárquico, que es donde se centra nuestro trabajo, también existen métodos para acelerar el cálculo del resultado [Ols95].

3. Representación multirresolución de puntos vectoriales

Para conseguir una representación multirresolución el algoritmo se divide en tres partes: (i) generación de las estruc-

© The Eurographics Association 2009.

turas de datos multirresolución, (ii) consulta y extracción de datos de las estructuras generadas, (iii) finalmente visualización del conjunto de puntos seleccionado.

3.1. Estructura de datos

Proponemos el uso de una estructura de datos tipo árbol binario para representar la multirresolución de los puntos. Las estructuras tipo árbol son conocidas por ser bastante eficientes y bastante usadas en otros sistemas que hacen uso de niveles de detalle continuos. También este tipo de estructura de datos permite incorporar un sistema de paginación para conjuntos de datos mayores que la memoria principal.

Para entender la generación del árbol binario hacemos uso de conceptos usados en *clustering* jerárquico. Los métodos de *clustering* normalmente dan como resultado un conjunto de *clusters*, es decir tratan de buscar qué elementos de los datos tienen características comunes. Éstas pueden ser por proximidad o por ciertos atributos que poseen los elementos.

El método que se presenta modifica el concepto de generación de *clustering* jerárquico considerando cada uno de los nodos del árbol como un *cluster* que a su vez contiene otros *clusters*. Las hojas del árbol son propiamente el dato (puntos en este caso). De esta forma podemos recorrer el árbol desde la raíz (menor nivel de detalle) hasta las hojas (mayor nivel de detalle).

Un *cluster* se define por (i) cúales son sus hijos (otros *clusters*), que en un principio serán dos para los nodos intermedios y las hojas no contendrán elementos hijos, tan solo el valor del punto, (ii) la mediana (*medoid*) del *cluster*, (iii) su caja de inclusión (*bounding box*) y (iv) el radio de la esfera mínima que contiene su caja de inclusión.

La caja de inclusión de un *cluster* se determina por la posición de los *clusters* que contiene. Dicho cálculo se propaga desde las hojas por lo que un nodo intermedio del árbol siempre tendrá una caja de inclusión mayor que la de sus hijos.

La posición en el espacio de un *cluster* se determina por su *medoid*. El *medoid* es el punto con menor distancia al centro de la caja de inclusión del conjunto de los *clusters* hijo. Al igual que con la caja de inclusión el valor del *medoid* se propaga desde las hojas, por lo que un *medoid* de un nodo intermedio del árbol siempre seleccionará un punto real del conjunto de datos. La razón por la que se usan *medoids* es debido a que estamos en un ámbito geográfico y lo que se va a representar debe ser fiel a la realidad. Toda la información del *cluster* está relacionada con índices a los valores reales. La Figura 1 representa esquemáticamente lo que se acaba de explicar.

Para la generación del árbol binario el sistema permite desarrollar diferentes algoritmos de generación (*cluster builders*) así como el uso de diferentes algoritmos para el cálculo de distancias (*distance method*). Actualmente el único método de cálculo de distancias implementado es el de distancia euclidea tomando como valor para el cálculo la posición geográfica de la característica (*feature*) de tipo punto.

Actualmente se han desarrollado dos métodos de generación, en ambos casos los cálculos hacen uso de la distancia euclidea y en el caso de los *clusters* intermedios toman el *medoid* como valor a usar en el cálculo.

Algorithm 1 Algoritmo Bottom-Up para la generación del árbol binario multirresolución

```
clusterList ⇐ Points
while clusterList.size() > 2 do
    Cluster_1 ⇐ ExtractElement(clusterList)
    Cluster_2 ⇐ ExtractElement(clusterList)
    distance ⇐ Distance(Cluster_1, Cluster_2)
    mergeCluster ⇐ (Cluster_1, Cluster_2)
    for i = 0 to clusterList.size() do
        for j = i + 1 to clusterList.size() do
            Cluster_i ⇐ clusterList[i]
            Cluster_j ⇐ clusterList[j]
            distance_{i,j} = Distance(Cluster_i, Cluster_j)
            if distance_{i,j} < distance then
                distance ⇐ distance_{i,j}
                mergeCluster ⇐ (Cluster_i, Cluster_j)
            end if
        end for
    end for
    clusterList ⇐ mergeCluster
    RemoveElement(mergeCluster.cluster_0)
    RemoveElement(mergeCluster.cluster_1)
end while
Cluster_1 ⇐ ExtractElement(clusterList)
Cluster_2 ⇐ ExtractElement(clusterList)
rootCluster ⇐ (Cluster_1, Cluster_2)
```

- **Método *Bottom-Up*:** Este método aplica el algoritmo más sencillo que existe en *clustering* jerárquico. El algortimo comienza considerando cada uno de los puntos como un *cluster* hoja. A continuación extrae un *cluster* del conjunto de datos y calcula la distancia con cada uno de los otros *clusters*. Durante el proceso de cálculo se selecciona el *cluster* de menor distancia. Finalmente crea un *cluster* que los agrupa e inserta el nuevo *cluster* en el conjunto de datos, eliminando los dos elementos seleccionados para no volver a calcular con ellos. Este proceso se repite hasta que únicamente queda el *cluster* raíz. En el algoritmo 1 se describe en pseudo-código.
- **Método Top-Down:** Este método aplica un algoritmo de inserción en el árbol binario. El algoritmo comienza considerando todos los puntos como un *cluster*. Primero toma dos *clusters* (C_{h1},C_{h2}), y los agrupa en otro *cluster* (C_p). A continuación procede a buscar en que punto del árbol debe insertar un nuevo *cluster* (C_i). Para la búsqueda atraviesa

© The Eurographics Association 2009.

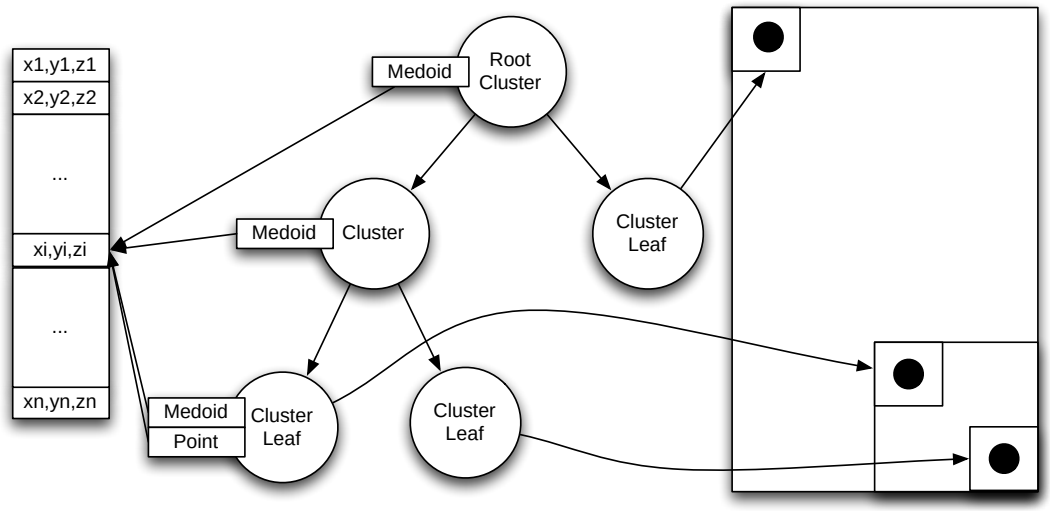


Figura 1: *Representación del árbol binario de clusters, nótese que las hojas contienen el valor del punto que coincide con el del medoid. A la derecha se muestra la representación conceptual de los puntos geográficos y los rectángulos representan el* cluster *al que pertenece.*

C_p y calcula la distancia entre C_i y sus hijos C_{h1} y C_{h2} quedándose con la menor distancia (d_{min}). Si d_{min} es menor que el radio calculado para la esfera de inclusión correspondiente a C_p entonces atraviesa el *cluster* hijo de menor distancia y repite el proceso. En caso contrario agrupa los *clusters* C_p y C_i. Si el *cluster* a procesar es una hoja, entonces procede a fusionarlos directamente. En caso de que el *cluster* a fusionar sea la raíz del árbol los agrupa y actualiza la raíz con el nuevo *cluster*. El algoritmo 2 ilustra lo que se acaba de describir.

Los *clusters* que se extraen en ambos algoritmos se realiza sin ningún tipo de heurística de selección, por lo que se obtiene el primer elemento de la lista. Una vez que la estructura ha sido generada, puede ser almacenada en disco. Actualmente sólo se soportan aquellos conjuntos de datos que quepan en memoria pricinpal, aunque se puede fácilmente extender a un sistema de generación *out-of-core*.

3.2. Proceso de consulta

El proceso de consulta consiste en extraer un conjunto de puntos del árbol multirresolución. El método acepta un área de interés y un factor de profundidad normalizado. El área de interés se utiliza para poder recortar el árbol y quedarse sólo con los *clusters* dentro de la extensión.

Una vez recortado el árbol y determinado el conjunto de *clusters* inicial, el número de *clusters* a extraer se determina haciendo uso del factor de profundidad establecido. Un valor de 0, devuelve la resolución por defecto, es decir devuelve

Algorithm 2 Algoritmo Top-Down para la generación del árbol binario multirresolución

$clusterList \Leftarrow Points$
$C_{h1} \Leftarrow ExtractElement(clusterList)$
$C_{h2} \Leftarrow ExtractElement(clusterList)$
$currentRoot \Leftarrow (C_{h1}, C_{h2})$
while $clusterList.size() > 2$ **do**
  $C_i \Leftarrow ExtractElement(i, clusterList)$
  $C_m \Leftarrow FindClusterToMerge(currentRoot, C_i)$
  $MergeClusters(currentRoot, C_i, C_m)$
end while
$rootCluster \Leftarrow currentRoot$
procedure FINDCLUSTERTOMERGE(C_p, C_i)
  $C_{h1}, C_{h2} \Leftarrow GetChildren(C_p)$
  $d_{min} \Leftarrow Distance(C_i, C_{h1})$
  $minCluster \Leftarrow C_{h1}$
  $distance_{i,h2} = Distance(C_i, C_{h2})$
  if $distance_{i,h2} < d_{min}$ **then**
    $d_{min} \Leftarrow distance_{i,h2}$
    $minCluster \Leftarrow C_{h2}$
  end if
  $radius = GetRadiusBound(C_p)$
  if $distance < radius$ **then**
    $FindClusterToMerge(minCluster, C_i)$
  end if
  return C_p
end procedure

© The Eurographics Association 2009.

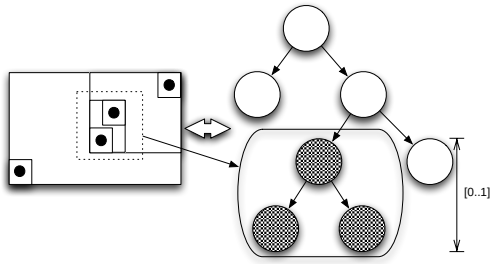


Figura 2: *Ejemplo esquemático de una consulta en un árbol multirresolución de puntos. A la izquierda se encuentran los puntos geográficos y a la derecha su representación en forma de clusters. El recuadro punteado indica el área de extensión de la consulta y el resultado se marca en sombreado en el árbol. El factor de profundidad para extraer mayor o menor resolución queda indicado en el rango de valores de de la imagen.*

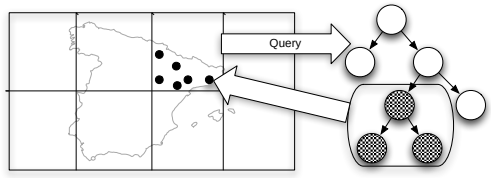


Figura 3: *Proceso de dibujado de una capa de puntos con el sistema multirresolución.*

únicamente los puntos asociados a los *clusters* que encajan dentro del área de extensión sin profundizar más en el árbol.

El factor normalizado de profundidad se basa haciendo uso del número de niveles de profundidad del árbol desde el menor nivel obtenido por la consulta hasta las hojas. Un factor de profundidad mayor que 0 provoca que cada *cluster* con hijos se atraviese hasta que se alcanza la profundidad adecuada. La figura 2 muestra gráficamente lo que se acaba de explicar.

Una vez determinados los *clusters* dentro del área de interés y a la profundidad requerida, se hace uso del *medoid* de los *clusters* para obtener los puntos geográficos reales.

3.3. Representación de puntos en 3D

Normalmente, el conjunto de datos se representa sobre extensiones de terreno. El algoritmo funciona perfectamente sobre el trabajo previo realizado en la librería OSGVirtualPlanets, aunque puede ser integrado en cualquier otro sistema capaz de representar puntos geográficos o incluso puede ser integrado en bases de datos geográficas para mejorar la transferencia de datos.

La integración con el trabajo previo sobre terrenos se basa en añadir un nuevo tipo de capa vectorial al sistema. Cuando se añade una capa vectorial al terreno, los *tiles* que entran dentro de la extensión de la capa realizan peticiones mediante eventos de salida. Cada evento generado contendrá la extensión del *tile* y un factor de profundidad asociado.

Una vez recogido el evento, se genera una consulta al modelo multirresolución de la capa vectorial. Haciendo uso de la extensión del *tile* y del factor de profundidad se extraen los puntos y se genera un grafo de escena con esa información. Posteriormente, se genera un evento de entrada y en la fase adecuada se asocia el grafo de escena al *tile*. La figura 3 muestra el proceso de dibujado con la librería OSGVirtualPlanets.

El sistema permite representar los puntos de dos maneras, la primera es haciendo uso directamente de puntos de OpenGL mediante un grafo de escena en OpenSceneGraph y activando, por ejemplo, la extensión de *point sprites* para conseguir mejorar el aspecto visual.

La otra forma hace uso del nodo *Overlay* de OpenSceneGraph, para dibujar los puntos pegados al terreno. El nodo *Overlay* es capaz, dado el tipo proyección en el que se encuentra el terreno, de generar un *render* a textura de una parte del grafo de escena y pegarlo sobre el terreno. Este método generalmente se prefiere cuando los puntos no tienen elevación o el terreno tiene alguna capa de elevación asociada.

4. Extensión a polilíneas y polígonos

Para polilíneas y polígonos las estructuras de datos a utilizar van a ser similares, es decir, utilizaremos un árbol binario para representar la jerarquía multirresolución. Dado que los elementos a representar poseen conectividad, se hace necesario definir un orden entre los puntos que lo representan.

Habitualmente, las características vectoriales GIS vienen determinadas por un orden implícito. Las polilíneas definen su orden por la aparición de los vértices, es decir el primer vértice está conectado con el segundo y así sucesivamente. Los polígonos definidos en sentido antihorario indican que el área de interés es el interior del polígono (*polígono interior*) mientras que los definidos en sentido horario representan agujeros dentro de otros polígonos (*polígono exterior*). En este trabajo tan sólo nos centramos en polígonos interiores.

El nodo raíz del árbol representará la polilínea o polígono de menor resolución. En el caso de las polilíneas será únicamente el segmento con mayor distancia entre dos puntos del elemento. Para los polígonos será el triángulo, cuyos segmentos serán los de mayor distancia entre los puntos del polígono.

Tanto los nodos intermedios como las hojas del árbol representarán diferencias con respecto al nodo padre, es decir, almacenarán (i) la altura (h) del triángulo que se formaría al

© The Eurographics Association 2009.

Tabla 1: *Resumen de resultados con el conjunto de datos de test*

Capa	Número de puntos	Tiempo Bottom-Up	Tiempo Top-Down	Tiempo Medio de Extracción
cities.shp	2533	104,308*sec*	0,010*sec*	$2{,}134e-05sec$
poblac_4326.shp	30223	∞	0,13*sec*	0,00029*sec*
gaz.shp	156638	∞	0,869732*sec*	$6{,}228e-05sec$
schools_usa.shp	339667	∞	1,57*sec*	0,00038*sec*
points_usa.shp	1087819	∞	6,23*sec*	0,0013*sec*

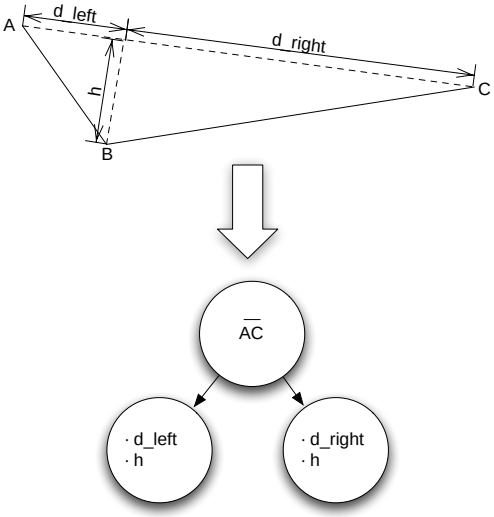


Figura 4: *Estructura de datos para polilíneas y polígonos basado en diferencias. El nodo raíz, contiene almacenado el segmento de menor resolución $\bar{AC}$. Los nodos hijo almacenan los valores h y d_{left}, d_{right} necesarios para la obtención de los segmentos $\bar{AB}$ y $\bar{BC}$.*

unir el origen de un segmento con el final del otro, y también (ii) la distancia (d_{left}, d_{right}) al punto de intersección del segmento que corresponde con la altura y el segmento del nodo padre. Dichos valores, junto al nodo padre permitirán obtener el segmento de línea que corresponde a esa diferencia. Para poder entender mejor la estructura de datos la figura 4 explica en forma esquemática el significado de cada uno de los elementos.

La generación del árbol multirresolución podrá ser parametrizada, aunque la primera aproximación se basará en simplificar segmentos de menor distancia. Almacenando las diferencias hasta el nodo raíz.

La consulta será similar a la explicada con los puntos, es decir, se recortará el área de interés, y posteriormente se calcularán los segmentos a dibujar. Además se tendrá que realizar un post-proceso donde se añadan puntos en el caso de que algún segmento atraviese los límites del área de consulta.

Uno de los problemas que se tiene que resolver con esta técnica es el recorte de los polígonos y de las polilíneas en los bordes de la consulta. En muchos casos se obtendrán entidades que ocuparán varias consultas (o áreas de terreno), en este caso se prevee la adición de puntos en los bordes, conservando el orden topologíco original. Otro tipo de problemas que se debe resolver es el solapamiento de múltiples entidades que se tengan que representar en la misma consulta.

5. Resultados

La figura 5 muestra capturas del sistema en funcionamiento. Para las pruebas se han cargado diferentes capas de puntos. Dos que contienen las poblaciones del mundo, una con las más importantes (*cities.shp*) y otra con más detalle (*gaz.shp*). También se ha probado con un conjunto de datos que contiene la mayoría de las poblaciones de España (*poblac_4326.shp*). Finalmente se han hecho pruebas con una capa de puntos de interés (*poi_usa.shp*) y otra de las escuelas e institutos en los Estados Unidos de América.

La imagen 5(a) muestra una vista general de la tierra con unas pocas poblaciones, en la imagen 5(b) se ha realizado zoom sobre un área y se puede observar la aparición de poblaciones al aumentar el nivel de detalle. La prueba se ha realizado con las capas *cities.shp* y *poblac_4326.shp*.

Las figuras 5(d) y 5(c) muestra una captura con dos niveles de detalle diferente donde se pueden apreciar las cajas de inclusión de los *clusters* asociados a los puntos. La situación de los puntos se determina por el *medoid* asociado al *cluster*.

La tabla 1 muestra un resumen de los resultados obtenidos al aplicar cada uno de los algoritmos de generación. También se muestran los tiempos medios de consulta a la estructura de datos multirresolución de los diferentes conjuntos de datos. El *frame-rate* durante la navegación se ha mantenido en su mayor parte por encima de $60fps$.

Los resultados han sido obtenidos en un MacBook Pro con un procesador Intel Core 2 Duo a $2{,}5GHz$ y $4Gb$ de RAM. Debido al coste del algoritmo *Bottom-Up* no ha sido posible calcular el tiempo de los conjuntos de datos con un número de puntos elevado.

© The Eurographics Association 2009.

El banco de pruebas utilizado es una base de datos pregenerada de *OpenScenGraph* de manera que cada *tile* cargado genera una consulta a la estructura multirresolución. La petición de consulta se realiza en un hilo de ejecución independiente aprovechando la arquitectura que OSG ofrece.

Las figuras 5(e) y 5(f) muestran el sistema haciendo uso del nodo *Overlay* para poder proyectar los puntos sobre el terreno. Además el nivel de detalle depende del punto de vista de la misma forma que en el otro método de dibujado.

Para poder comparar el resultado visual, se han realizado varias pruebas sin la utilización de nuestro sistema multirresolución para puntos. La imagen de la figura 6 se ha tomado únicamente con la capa *poblac_4326*, que contiene un número masivo de puntos en una región relativamente pequeña, dando como resultado que no se pueda distinguir entre los diferentes elementos.

Las figuras 7(a) y 7(b) muestran los datos *gaz.shp* y *poi_usa.shp*. La primera de ellas se ha tomado haciendo uso del algoritmo multirresolución explicado, la segunda muestra el resultado sin hacer uso del sistema.

6. Conclusiones y trabajo futuro

Hemos desarrollado un modelo multirresolución para la visualización de información vectorial para sistemas de información geográfica. El sistema permite generar una base de datos multirresolución capaz de extraer diferentes niveles de detalle de una capa vectorial de puntos procesada. También se definen las bases para la extensión de los algoritmos para capas de polilíneas y polígonos.

El sistema es capaz de visualizar más de una capa al mismo tiempo, extrayendo diferentes niveles de detalle para cada una de ellas. Además se ha integrado con el framework OSGVirtualPlanets aunque puede ser integrado en otros sistemas al estar completamente desarrollado en C++.

Próximamente se implementarán las estructuras de datos multirresolución para polilíneas y polígonos, además de añadir nuevos métodos para el cálculo de distancias y generación de modelos multirresolución. La idea es probar nuevos métodos y compararlos con el trabajo ya desarrollado y con otros sistemas existentes.

Otra línea de trabajo consiste en añadir edición interactiva en capas vectoriales multirresolución, teniendo en cuenta la simbología con que es aplicada y poder integrarlo en aplicaciones de información geográfica.

7. Agradecimientos

Este trabajo está siendo financiado por la *Conselleria d'Infraestructures i Transport* de la *Generalitat Valenciana* (Spain).

References

[AAJ06] ANUPAM AGRAWAL M. R., JOSHI R.: Geometry-based mapping and rendering of vector data over lod phototextured terrain models. In *14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)* (2006), pp. 1–8.

[ASZ07] A. SCHILLING J. B., ZIPF A.: Vector based mapping of polygons on irregular terrain meshes for web 3d map services.

[Ber02] BERKHIN P.: *Survey Of Clustering Data Mining Techniques*. Tech. rep., Accrue Software, San Jose, CA, 2002.

[BN08] BRUNETON E., NEYRET F.: Real-time rendering and editing of vector-based terrains. In *Eurographics* (2008), vol. 27, pp. 311–320.

[DVS03] DACHSBACHER C., VOGELGSANG C., STAMMINGER M.: Sequential point trees. *ACM Trans. Graph. 22*, 3 (2003), 657–662.

[EsjC00] EL-SANA J., JEN CHIANG Y.: External memory view-dependent simplification. *Computer Graphics Forum 19* (2000), 139–150.

[ESV99] EL-SANA J., VARSHNEY A.: Generalized view-dependent simplification. In *Computer Graphics Forum* (1999), pp. 83–94.

[IhA03] INSTITUTE K.-H. A., HEINRICH ANDERS K.: A hierarchical graph-clustering approach to find groups of objects. In *In ICA Commission on Map Generalization, 5th Workshop on Progress in Automated Map Generalization* (2003), pp. 05–10.

[MS07] MARTIN SCHNEIDER R. K.: Efficient and accurate rendering of vector data on virtual landscapes. *Journal of WSCG 15* (2007), 1–3.

[OK02] OLIVER KERSTING J. D.: Interactive 3d visualization of vector data in gis. In *Proceedings of the 10th ACM international symposium on Advances in geographic information systems* (2002), pp. 107–112.

[Ols95] OLSON C. F.: Parallel algorithms for hierarchical clustering. *Parallel Computing 21* (1995), 1313–1325.

[RL00] RUSINKIEWICZ S., LEVOY M.: QSplat: A multiresolution point rendering system for large meshes. In *Proceedings of ACM SIGGRAPH 2000* (July 2000), pp. 343–352.

[SCZ98] SHEIKHOLESLAMI G., CHATTERJEE S., ZHANG A.: Wavecluster: A multi-resolution clustering approach for very large spatial databases. In *VLDB '98: Proceedings of the 24rd International Conference on Very Large Data Bases* (San Francisco, CA, USA, 1998), Morgan Kaufmann Publishers Inc., pp. 428–439.

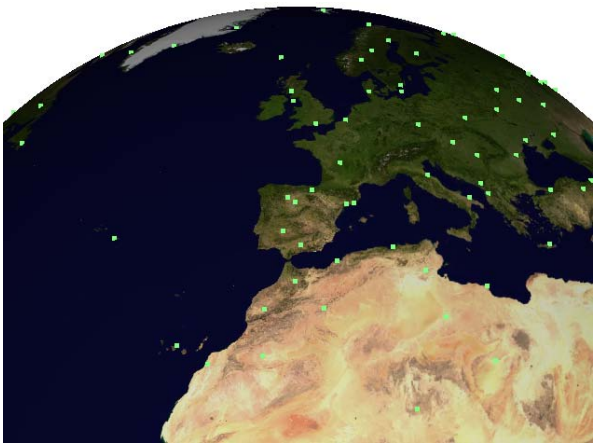
[SGK05] SCHNEIDER M., GUTHE M., KLEIN R.: Real-time rendering of complex vector data on 3d terrain models. In *In Proceedings of The 11th International Conference on Virtual Systems and Multimedia* (2005), pp. 573–582.

[TTZ*08a] TEN M., TORRES J., ZARZOSO J., GAITÁN R., LLUCH J.: Visualización esférica de terreno 3d para sistemas de información geográfica. In *CEIG 2008: Congreso Español de Informática Gráfica* (2008), Eurographics Association, pp. 1–10.

[TTZ*08b] TORRES J., TEN M., ZARZOSO J., GAITÁN R., LLUCH J.: Representación vectorial 3d en un sistema de información geográfica. In *CEIG 2008: Congreso Español de Informática Gráfica* (2008), Eurographics Association, pp. 253–256.

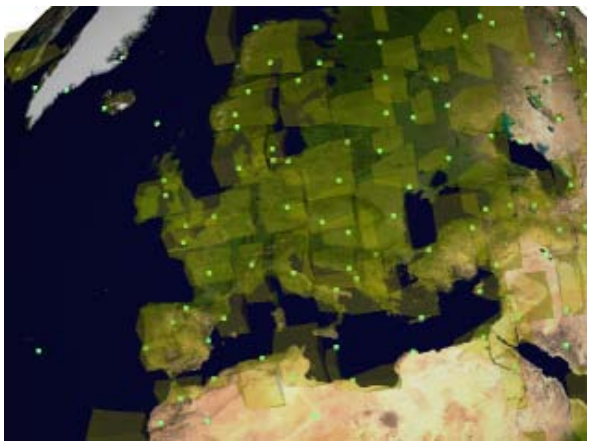
[WKW*03] WARTELL Z., KANG E., WASILEWSKI T., RIBARSKY W., FAUST N.: Rendering vector data over global, multi-resolution 3d. In *Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization 2003* (2003), pp. 213–222.

© The Eurographics Association 2009.



(a) Vista alejada de España con unas pocas poblaciones.

(b) Detalle de España representando poblaciones de mayor nivel de detalle.



(c) Detalle de una sección de la tierra mostrando las cajas de inclusión de los clusters con poco nivel de detalle.

(d) Detalle de una sección de la tierra mostrando poblaciones de resolución intermedia, además de la caja de inclusión del cluster asociado.

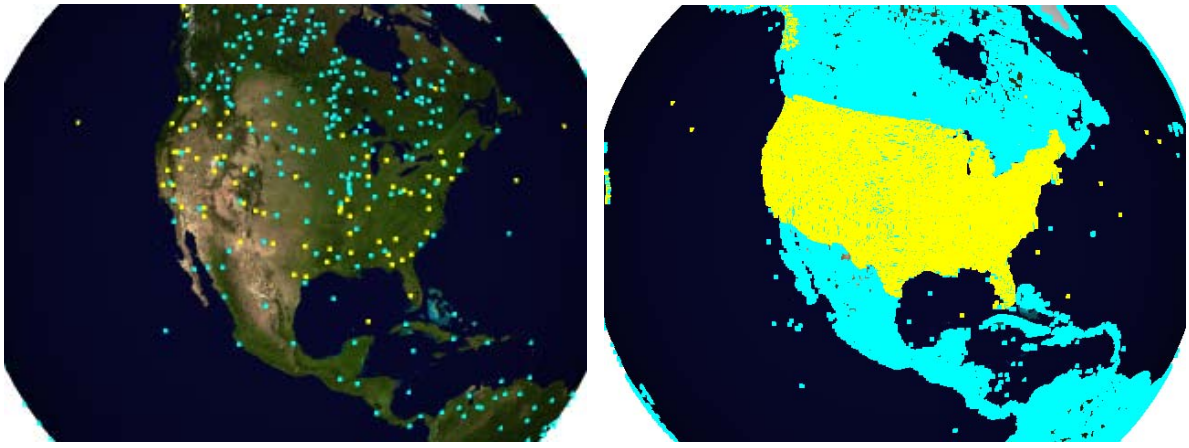
(e) Ejemplo de uso del modo de dibujado mediante el nodo Overlay. Vista alejada con menor nivel de detalle.

(f) Ejemplo de uso del modo de dibujado mediante el nodo Overlay. Detalle con mayor resolución en los puntos.

Figura 5: *Vistas con diferente zoom a la tierra mostrando los cambios sobre las capas vectoriales de puntos multirresolución.*

© The Eurographics Association 2009.

Figura 6: *Captura del conjunto de datos poblac_4326 sin hacer uso del sistema multirresolución. Nótese que no se puede distinguir un punto de otro debido a la gran masa de puntos acumulada en la región.*



(a) Vista de los datos con poco nivel de detalle. (b) Imagen mostrando mayor nivel de resolución en los datos.

Figura 7: *Dos vistas de una región aplicando el algoritmo multirresolución(izquierda y a la derecha sin aplicarlo. El sistema tiene las capas con poblaciones del mundo y puntos de interés en Estados Unidos de América.*

© The Eurographics Association 2009.