REVISIÓN

Método para la visualización 3d de modelos digitales de elevación.

Ing. Edwin Antonio Mero Lino. [edwin.mero@unesum.edu.ec]
Universidad Estatal del Sur de Manabí. Ecuador.

Ing. María Mercedes Ortiz Hernández. [mariaortizh2009@hotmail.com]
Universidad Estatal del Sur de Manabí. Ecuador.

Ing. Cristian José Alava Mero. [cristian.alava@unesum.edu.ec]
Universidad Estatal del Sur de Manabí. Ecuador.

Ing. Jimmy Leonardo Gutiérrez García. [jimmy.gutierrez@unesum.edu.ec]
Universidad Estatal del Sur de Manabí. Ecuador.

Resumen

La visualización de información geográfica sobre la Web representa un área de la ciencia que ha ganado importancia con el desarrollo científico técnico. Los métodos comúnmente utilizados realizan el procesamiento de la información en los servidores generando imágenes que son enviadas a las estaciones clientes para su visualización. Sin embargo, la forma de representar la información genera altos tráficos de datos desde las estaciones servidoras hacia las estaciones cliente y no son explotados los recursos de cómputos con que cuentan las estaciones clientes. La presente investigación describe una solución a la problemática planteada a partir de la implementación de un modelo para la visualización eficiente de superficies de terrenos en tres dimensiones sobre la plataforma Web. El método propuesto explota los recursos disponibles en el ordenador cliente, logrando visualizaciones de alta calidad con decenas de tramas por segundo. Para la visualización de la información se realiza una división lógica del terreno en cuadrantes, permitiéndole al ordenador hacer uso de una estructura de datos espacial (fttree) para manipular las peticiones al servidor, así como el control de memoria operativa de los datos asociados a aquellos cuadrantes que se están visualizando. Se describe, además, un modelo de triangulación basado en el Quadtree Restringido (RQT) y se detalla el proceso de simplificación y prevención de grietas.

Palabras claves: GPU; quadtree restringido; triangulación multiresolución; visualización 3D.

Recibido: 15/01/2020 | **Aceptado**: 31/05/2020

Method for 3d visualization of digital elevation models.

Abstract

The visualization of geographic information on the Web represents an area of science that has gained importance with scientific and technical development. The methods commonly used

perform the processing of information on the servers generating images that are sent to the client stations for viewing. However, the way of representing the information generates high data traffic from the server stations to the client stations and the computation resources used by the client stations are not exploited. The present investigation describes a solution to the problematic raised from the implementation of a model for the efficient visualization of land surfaces in three dimensions on the Web platform. The proposed method exploits the resources available in the client computer, achieving high quality visualizations with tens of frames per second. For the visualization of the information, a logical division of the terrain is made in quadrants, allowing the computer to make use of a spatial data structure (fttree) to manipulate the requests to the server as well as the operational memory control of the data associated with those quadrants that are being visualized. It also describes a triangulation model based on the Restricted Quadtree (RQT) and details the process of simplification and prevention of cracks.

Keywords: GPU; restricted quadtree; multiresolution triangulation; 3D visualization.

Introducción

Los visores de información geoespacial en 3D para la Web han logrado la visualización de superficies a través del envío masivo de imágenes a distintas resoluciones. Estas imágenes deben arribar a la estación cliente en un tiempo mínimo para que la experiencia del usuario resulte fluida. La mayoría de los visores toman como base los estándares abiertos especificados por el Open Geospatial Consortium (OGC), como son los casos de Web Terrain Service (WTS) y Web Perspectiva View Service (WPVS), los cuales se enfocan en la construcción de imágenes en servidores con hardware especializado a tal efecto.

Esta arquitectura propone que la mayor parte del procesamiento sea realizada por los servidores, los cuales devuelven la vista como resultado para la representación (Over & al, 2010). Estos métodos eran eficientes hace 5 años atrás, pero hoy en día con tecnologías como HTML5 y WebGl, que brindan al navegador web la capacidad de representar gráficos en tres dimensiones acelerados por hardware (GPU) y con el alto poder de cómputo que presentan las estaciones clientes actuales. Estas tecnologías no explotan dichas cualidades. Representan un enorme costo en la adquisición de recursos computacionales para el servidor. Para atacar este problema han surgido modelos que aprovechan las potencialidades del cliente para representar y visualizar la información geográfica (Zhu, Tan, & Chan, 2003).

Para alcanzar una visualización interactiva de terrenos han surgido modelos que se basan en representar la superficie como un conjunto de puntos que forman polígonos contiguos y no

solapados a partir de fuentes de datos denominadas Modelos Digitales de Elevación (MDE), con el objetivo de ser visualizados mediante una API gráfica.

El modelo de triangulación es el núcleo para cada sistema de visualización de superficies de terreno. La mejor forma de proveer una representación eficiente es reducir tanto como sea posible la complejidad y número de primitivas geométricas usadas para la representación de la escena (De Berg & al., 2000). Sin embargo, la reducción de geometría provoca una inferior calidad de visualización, por lo que debe ser controlada por un error. En este caso, la simplificación es realizada solo cuando el error de la aproximación esté debajo de un umbral especificado (Pajarola, 1998), (Escrivà, Jaime, Gutiérrez, & Beltrán-Meneu, 2017).

Otra manera para incrementar el rendering es el uso de niveles de detalles (LOD). Este concepto corresponde al modo que el ser humano percibe los objetos a diferentes distancias. De acuerdo con ello, los objetos cercanos tienen mayor detalle que los lejanos (Vega, 2013). Al representar una superficie de forma adaptable con múltiples LODs, de acuerdo con la posición y orientación del observador, se logra un rendimiento máximo en la escena. Estos modelos se conocen como triangulación multi-resolución (To, WH., & Green, 2001),(Acosta, 2018).

Entre las estructuras más utilizadas se encuentran las basadas en árboles binarios de triángulos (Evans, Kirkpatrick, & Townsend, 2001), r-trees y los quadtree (Corral, Vassilakopoulos, & Manolopoulos, 1999), (Samet, 1990). Según la literatura consultada los métodos basados en quadtrees han demostrado ser excepcionalmente eficientes para la construcción y representación de mallas multi-resolución. Uno de los primeros métodos eficientes basados en quadtree es presentado en Lindstrom & al, (1996) y Fuentes et al., (2018). El algoritmo usa una compacta y eficiente malla regular para la representación y aplica una característica del espacio-pantalla como umbral para delimitar el máximo error de la imagen. Se presenta un modelo de triangulación basadas en restricciones del quadtree para lograr una visualización sin grietas, además incluye una métrica de error espacio-objeto para controlar la aproximación de la superficie (Pajarola, 1998),(Hernández Muñoz et al., 2018).

Aun así, representar en su totalidad la superficie en una sola estructura es altamente ineficiente, sobre todo en escenas donde solo se observa una parte del mapa. Esto hace necesario aplicar otro tipo de aproximaciones como refiere Ulrich, (2002) donde, en lugar de una única malla de elevaciones, se subdivide esta en mallas más pequeñas, pero con mayor nivel de detalle denominadas cuadrantes. La geometría de los cuadrantes se procesa de forma que en tiempo de ejecución se seleccionan aquellos que se visualizarán según el punto de vista del

observador. Se pueden encontrar otras aproximaciones similares, algunas incluso con paginación para las texturas y datos de elevación (De Boer, 2000), (Cignoni & al., 2003).

En el presente trabajo se propone un modelo web que permite la representación en tiempo real de superficies de terreno, explotando los recursos disponibles en la estación cliente, lográndose visualizaciones de alta calidad con decenas de tramas por segundo (FPS).

Desarrollo

Como solución al envío masivo de imágenes y para alcanzar una visualización interactiva de superficies continuas en la web, se realiza un enfoque hacia los modelos anteriormente mencionados, utilizando eficientemente los recursos de cómputo disponibles en las estaciones clientes. Por consiguiente, se utiliza un mecanismo de cuadrantes para la gestión dinámica de la escena y se sustituye el envío de imágenes por los MDE a partir de los cuales se construye una red de triángulos derivada del RQT, para su posterior visualización a través del API gráfico.

El reto principal es actualizar continuamente los datos dentro del cliente sin sobrecargar la memoria de este ni afectar la visualización. Para ello, se hace uso de técnicas de recorte sobre el campo de visualización, eliminando los cuadrantes que estén fuera del área visible (Vega, 2013), (Herrera et al., 2018). Esta técnica evita la carga de datos irrelevantes o redundantes disminuyendo el tráfico de red, así como el consumo de memoria en el cliente (“Selección adaptativa” tal como muestra la Figura 1). Los cuadrantes que serán utilizados para la visualización de la escena se envían de forma asíncrona desde el servidor hasta el cliente (“Carga Asíncrona”).

Para garantizar una visualización fluida mientras el usuario navega por la escena, es necesario mantener en el lado del cliente un conjunto de cuadrantes que podrían incluirse potencialmente (debido a movimientos de la cámara en el proceso de navegación) en la escena; estos son conservados en una caché del lado del cliente (Tile Cache). Una vez obtenidos los datos por parte del cliente se construye la red de triángulos y se procede a visualizar (“Construcción de la Malla” y “Rendering”). La única tarea del servidor es leer los datos solicitados y transmitirlos al cliente (“Gestión de solicitudes”).

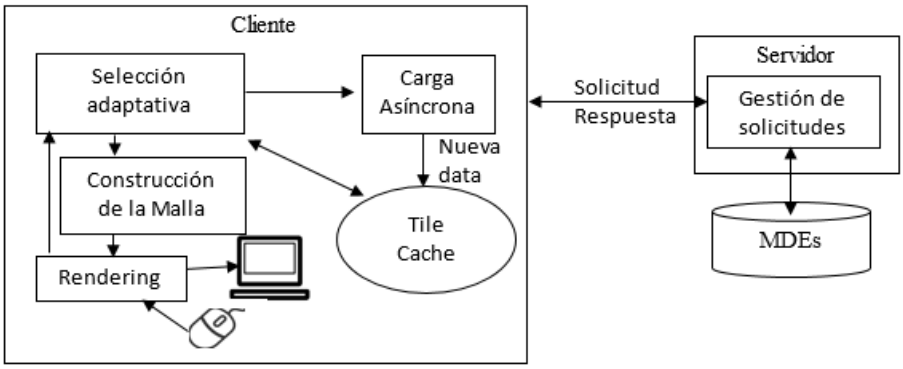

Figura 1. Arquitectura del Sistema.

Almacenamiento en el Servidor

Para almacenar la información de los MDE en el servidor se calcula previamente la matriz de error utilizando el proceso descrito en Pajarola, (2002). Dado que el valor de error asociado a cada punto de un cuadrante se mantiene constante durante toda la visualización, se calculan estos una sola vez y se almacenan junto con los valores de altura, evitando cálculos innecesarios en el futuro. Los datos se almacenan en memoria externa utilizando archivos en formato JSON. A fin de minimizar la actividad en el disco, se realiza un procesamiento de los cuadrantes donde se optimiza su organización. Este proceso tiene como datos de entrada las siguientes características del MDE:

- **ncols**: cantidad de columnas de la matriz de alturas.
- **nrows**: cantidad de filas de la matriz de alturas
- **xllcorner** y **yllcorner**: coordenadas georeferenciables del punto en la esquina inferior izquierda del mapa.
- **xdim:** distancia entre los puntos en el eje X.
- **ydim**: distancia entre los puntos en el eje Y.
- **nodata_value**: valor para representar alturas de puntos que no pudieron ser registradas.

A partir de esta información y la dimensión n de los cuadrantes en los que se dividirá el terreno, se determinan parámetros tales como: el número de elementos conformadores de las columnas ($\mathrm{n\_col\_tile}$) y las filas ($\mathrm{n\_row\_tile}$) de la matriz de cuadrantes que representa la superficie del mapa, el ancho (t_x) y alto (t_y) de cada cuadrante. Todo esto es expresado por las siguientes ecuaciones:

$$t_x = (n-1) * xdim \quad (1)$$

$$t_y = (n-1) * ydim \quad (2)$$

$$\mathrm{rc} = \begin{cases} 0, \mathrm{ncols\ MOD\ n} = 0 \\ 1, \mathrm{ncols\ MOD\ n} \neq 0 \end{cases} \quad (3)$$

$$rr = \begin{cases} 0, nrows \text{ MOD } n = 0 \\ 1, nrows \text{ MOD } n \neq 0 \end{cases} \quad (4)$$

$$n\_col\_tile = \frac{ncols}{n} + rc \quad (5)$$

$$n\_row\_tile = \frac{nrows}{n} + rr \quad (6)$$

Si la superficie del mapa no conforma exactamente una matriz con regiones de dimensión, entonces es necesario agregar otro cuadrante para incluir los puntos sobrantes y completarlo con puntos artificiales que no influyan en la visualización (ej.: poniéndoles altura nodata_value). La inclusión de un cuadrante adicional o no, es representado por – ecuación (3) – para cuando es necesaria una columna de cuadrantes adicional y – ecuación (4) – en caso de necesitarse una fila de cuadrantes extra. En la Figura 2 se muestra un ejemplo de división para una malla de 5x7. Nótese la inclusión de dos columnas de puntos que permitirán dividir la malla resultante en dos cuadrantes de 5x5.

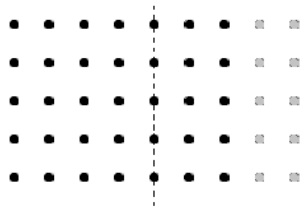

Figura 2. División de una malla de 5x7 en dos cuadrantes. El Segundo cuadrante se completa con puntos artificiales (grises).

La información de cada uno de estos cuadrantes se organiza y almacena por separado; teniendo para cada punto del cuadrante su valor de altura y el error calculado.

Selección adaptativa

En cada momento debe garantizarse que estén cargados en memoria aquellos cuadrantes que se están visualizando o aquellos que potencialmente pueden formar parte de la escena. El proceso de selección de los cuadrantes candidatos a mantenerse en memoria puede ser costoso computacionalmente. Con el objetivo de minimizar el tiempo de selección se organizan los objetos en forma jerárquica, reduciendo la complejidad en tiempo de búsqueda a un orden logarítmico, para ello se hace uso de la estructura de datos espacial FTTree, propuesta en Valle-Martínez & Ortiz-Rojas, (2011) como variante de división recursiva del espacio para dimensiones arbitrarias.

Construcción del FTTree.

El punto de partida para la creación del FTTree lo constituye toda la región del mapa con límites superior izquierdo en y el límite inferior derecho en, que representan el número de cuadrantes que corresponden a las filas y columnas en las que fue dividido el mapa.

El proceso de construcción del FTTree comienza calculando la relación entre el ancho y la altura de las nuevas regiones que serán creadas, y es llamado recursivamente mientras una región sea divisible horizontalmente y/o verticalmente, dependiendo de sus medidas. Las regiones que no puedan ser divididas (nodos hojas) representan los cuadrantes del mapa (Mar, Santana, & Gulín, 2017). Durante este proceso se determina el área que representará cada nodo en la visualización, la cual estará indicada por cuatro puntos y calculados de la siguiente forma:

$$\text{left} = x_0 * (t_x) \quad (7)$$

$$\text{rigth} = (x_1 + 1) * (t_x) \quad (8)$$

$$\text{up} = (\text{n\_rows\_tile} * (t_y)) - y_0 * (t_y) \quad (9)$$

$$\text{bot} = (\text{n\_rows\_tile} * (t_y)) - (y_1 - 1) * (t_y) \quad (10)$$

$$p_0 = [\text{left}, \text{up}],\ p_1 = [\text{rigth}, \text{up}],\ p_2 = [\text{left}, \text{bot}],\ p_3 = [\text{rigth}, \text{bot}] \quad (11)$$

Donde left representa la posición izquierda del área, up la posición superior del área, rigth la posición derecha del área, bot la posición inferior del área, (x_0, y_0) el punto ubicado en la esquina superior izquierda del cuadrante, (x_1, y_1) el punto ubicado en la esquina inferior derecha del cuadrante. Los valores t_x y t_y son calculados utilizando las ecuaciones (1) y (2), respectivamente.

Técnicas de recorte o culling.

La técnica de recorte propuesta es el *frustum-culling* (Vega, 2013). El frustum se encuentra delimitado por 6 planos: plano de recorte lejano, plano de recorte cercano, plano superior, plano inferior, plano izquierda y plano derecho.

Para definir la ubicación de un objeto basta con comprobar de qué lado del plano este está localizado. Si el objeto se encuentra al frente de los seis planos, indica entonces que está ubicado dentro del frustum. Se pueden identificar tres casos de acuerdo con este criterio.

- El objeto se ubica completamente dentro del frustum.
- El objeto se ubica parcialmente dentro del frustum.
- El objeto no se encuentra dentro del frustum.

Los nodos del árbol que coincidan con el primer caso, de no estar cargados o almacenados en cache, serán solicitados al servidor, sin necesidad de continuar descendiendo en la jerarquía. En el segundo caso, se debe continuar hasta encontrar la mínima parte que esté dentro del frustum. Los nodos que no estén en el frustum serán borrados de la memoria (en caso de que hayan sido cargados anteriormente). Cada vez que ocurre un cambio significativo en la escena

(traslaciones, aumento o disminución del nivel de detalle), se hace una consulta al FTTree para determinar qué cuadrantes necesitan ser cargados y cuáles liberados de la memoria interna de la computadora (ver Figura 3) (Arias-Naranjo & al., 2014).

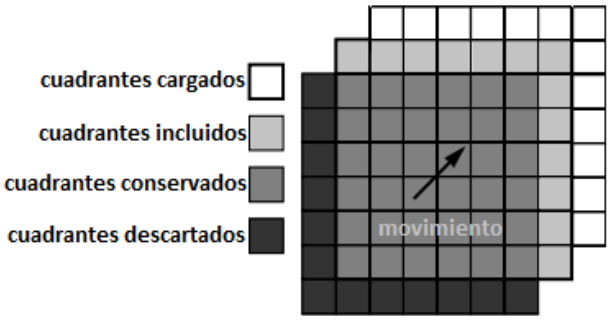

Figura 3. Gestión de la escena. Representación del tratamiento de los cuadrantes a medida que se desplaza el punto de vista de la cámara.

Construcción de la red de Triángulos usando quadtree restringido.

El término quadtree, según Samet, (1990) es usado para describir una clase de estructuras de datos jerárquicas cuya propiedad común es la descomposición recursiva del espacio. La triangulación mediante quadtrees es una triangulación jerárquica y adaptativa para superficies de terrenos, donde la información es representada mediante mallas regulares. Una malla regular es una matriz de puntos los cuales se encuentran distribuidos a distancias constantes por ambas dimensiones (Arias-Naranjo & al., 2014).

Dado que un quadtree irrestricto que satisfaga la tolerancia de aproximación puede presenciar grietas en la representación tridimensional de la superficie (ver Figura 4 a), en Pajarola, (1998) se adiciona una restricción a este proceso, de forma tal que los cuadrantes adyacentes o bloques del quadtree, solo pueden diferir por un nivel en la jerarquía del quadtree. Para hacer cumplir esta regla se inserta un concepto de grafo de dependencias, donde cada uno de los vértices depende de otros dos, del mismo o siguiente nivel, en la jerarquía del árbol. Estas reglas aseguran que cuando un vértice sea seleccionado para formar parte de la triangulación, sus dependencias también deben ser incluidas. Realizando este proceso se previene la aparición de grietas (ver Figura 4 b).

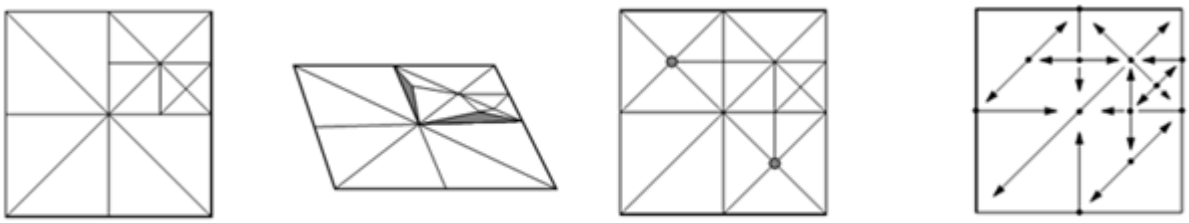

Figura 4. a) Grietas entre cuadrantes. b) Triangulación mediante el QuadTree restringido.

Construcción del RQT

El *quadtree* es construido implícitamente en la matriz de alturas $P_{ij}; i,j = 0..n-1$ que representa un cuadrante de dimensión $n = 2^k + 1$, donde su profundidad queda dada por $maxlevel = log_2(n-1)$.

Para emular las relaciones de padre e hijos entre nodos, desde la misma matriz, fue necesario hacer uso de un conjunto de fórmulas introducidas en Over & al, (2010). De esta forma las operaciones para acceder a los elementos del *quadtree* son de orden constante (O(1)) en términos de operaciones de máquina.

El nivel l en que se encuentra un punto P_{ij} dentro del árbol se determina utilizando las siguientes fórmulas:

$$t(i) = (2^{\lceil log_i 2 \rceil} - i) \oplus i \qquad (12)$$

$$d_l(i) = (t \oplus 2^{\lceil log_2 t \rceil}) \oplus (2t) \qquad (13)$$

$$l = \left(log_2\left(\frac{n-1}{d_l(i)}\right), log_2\left(\frac{n-1}{d_l(j)}\right)\right) \qquad (14)$$

Otro dato importante es conocer si un punto P_{ij} forma parte del conjunto de nodos centrales de determinado nivelo no. Esta condición es utilizada para dividir el *quadtree* en sus diferentes niveles. Un nodo del *quadtree* es divisible siempre que contenga un punto central para el nivel en que se encuentra. En caso contrario se considera el nodo una hoja. La siguiente ecuación es utilizada para determinar la pertenencia de P_{ij} a dicho conjunto:

$$P_{ij} \in L_l^{center} \Leftrightarrow \frac{i}{d_l} MOD\ 2 \neq 0 \wedge \frac{j}{d_l} MOD\ 2 \neq 0 \qquad (15)$$

Donde $d_l;\ d_l = (n-1)/2^l$ es la distancia entre los puntos del nivel l y L_l^{center} representa al conjunto de nodos centrales del nivel en cuestión.

Para realizar la selección de los puntos que estarán incluidos en la triangulación se compara el error de cada punto con un umbral determinado. Todo punto cuyo error esté por debajo del

umbral establecido, se considera no relevante para la triangulación y no es incluido en la misma.

Para este algoritmo se utiliza una estrategia *bottom-up*, comenzando por los nodos del último nivel y se van adicionando los puntos hasta llegar a aquellos en el nivel cero del árbol (los cuatro puntos en las esquinas de la matriz), (Arias-Naranjo & al., 2014), para ello, se verifican primeramente los puntos que no son centros en un mismo nivel junto con sus dependencias y posteriormente los puntos que son centros, con sus respectivas dependencias (Pajarola, 2002).

Un último paso importante es la extracción de la triangulación, luego de construida, para su futura visualización. La extracción de los triángulos utilizando abanicos es una de las variantes utilizadas para realizar este proceso. Se comienza utilizando el nodo centro del nivel uno y construyendo un abanico que contenga los cuatro cuadrantes en los que se divide el árbol en dicho nivel. Si uno de los cuadrantes se encuentra subdivido en cuadrantes más pequeños se desciende en la estructura y se construye otro abanico tomando como punto de referencia el centro de las cuatro regiones del cuadrante inicial (Arias-Naranjo & al., 2014).

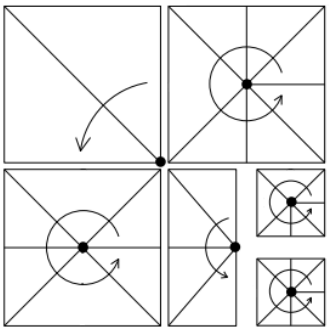

Figura 5. Construcción de los abanicos de triángulos.

Para evaluar el desempeño en tiempo real del modelo presentado se desarrolló un componente implementado en el lenguaje JavaScript, utilizando como interfaz gráfica WebGL y lenguaje de shader GLSL, con peticiones al servidor a través de Ajax. Se tomaron muestras de distintas fuentes de datos disponibles para su libre uso por la comunidad científica, de las cuales se interesa medir la cantidad de Millones de Triángulos por Segundo (MTPS) y la cantidad de tramas repintadas por segundos (FPS), en un sistema con las siguientes prestaciones:

- **Procesador**: Intel Core i3-2120 3.30 GHZ
- **Memoria Ram**: 4 GB
- **Tarjeta gráfica**: Intel HD Graphics 2000
- **Navegador Web**: Mozilla Firefox v35.0

Los datos utilizados tienen dimensiones de $2k \times 2k$ (alrededor de 4 millones de puntos) y $4k \times 4k$ (alrededor de 16 millones de puntos). Estos corresponden a determinadas áreas de la superficie terrestre con características de relieve representativas (valles, regiones montañosas, llanuras).

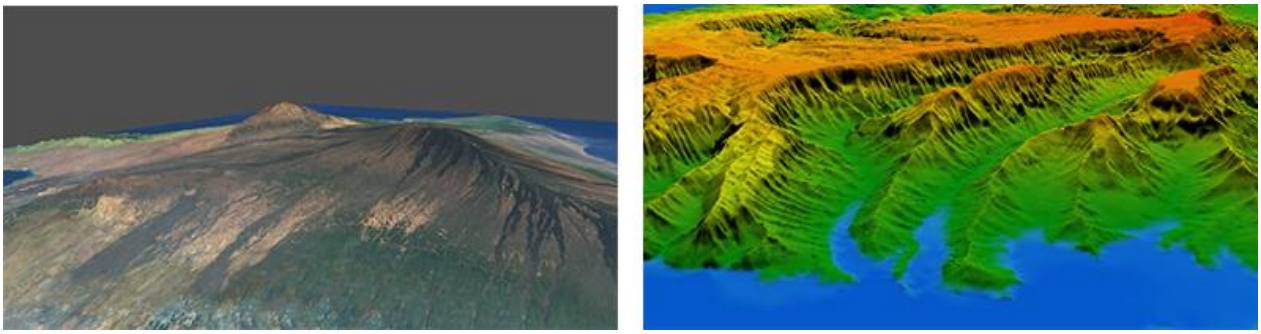

Figura 6. Ejemplo de vistas obtenidas con el componente. Archipiélago de Hawaii. Big Island texturizada (Izquierda), Kauai (derecha).

La prueba consiste en realizar un recorrido sobre la superficie del mapa, de forma tal que sea visualizada la mayor parte de este. En cada instante o trama del proceso se obtiene la cantidad de triángulos visualizados, pudiendo determinar el promedio de triángulos representados durante el recorrido, así como la cantidad de tramas por segundo.

MDE	Orden	Textura	No. de Tramas	MTPS	FPS
Big Insland	4k x 4k	2k x 2k	1977	63	51
Kauia	4k x 4k	2k x 2k	1835	75	45
Costa Rica	2k x 2k	2k x 2k	2088	49	55
Baia Mare	2k x 2k	2k x 2k	2043	18	59

Tabla 1. Resultados de las pruebas realizadas para un tamaño de imagen de 1024x768 píxeles.

Los resultados obtenidos muestran un desempeño promedio de 52.5 FPS. Teniendo en cuenta que los navegadores web solo soportan un máximo 60 FPS en sus funciones de repintado (*RequestAnimationFrame*) y el mínimo necesario para una representación continua es aproximadamente 30 FPS, se puede afirmar que mediante el método propuesto y la implementación de estructuras de datos y algoritmos que exploten los recursos de cómputo de la estación cliente, es posible la visualización de superficies de terreno en tres dimensiones desde la web y en tiempo real.

Conclusiones

1. El método presentado brinda una alternativa de representación tridimensional sobre la Web, de superficies de terrenos, que explota los recursos computacionales de la estación cliente. Al ser necesario el servidor solo para suministrar los datos se logra una independencia del componente desarrollado; permitiendo incluso visualizar datos que se encuentren en la estación cliente, sin necesidad de conectarse a un servidor.

2. El uso de WebGL y GLSL permite el aprovechamiento de los recursos de video existente en la estación de trabajo, aumentando de esta forma la eficiencia en la visualización. Existe una dependencia con el navegador que se esté utilizando; estos son los que restringen la versión de WebGL y GLSL a utilizar.
3. El quadtree restringido es una estructura que, para lograr una visualización sin grietas, necesita de la introducción de una gran cantidad de puntos auxiliares, aumentándose de esta forma la cantidad de primitivas a representar y, por consiguiente, disminuyendo el rendimiento del componente. La utilización de estructuras de datos que hagan uso de un número menor de primitivas permitirá la carga de terrenos de mayores dimensiones y aumenta la taza de tramas por segundo.

Referencias bibliográficas

Acosta, J. L. B. (2018). Visualización de textos 2D para entornos 3D. Desarrollo de Aplicaciones Interactivas Multimedia (48060).

Arias-Naranjo, G., & al., e. (2014). Gestión eficiente de modelos digitales de elevación para su visualización 3D utilizando procesamiento multinúcleo. Revista Cubana de Ciencias Informáticas, Vol. 8, 14-28.

Cignoni, P., & al., e. (2003). Planet-sized batched dynamic adaptive meshes (P-BDAM). Proceedings of the 14th IEEE Visualization 2003 (VIS'03), 20.

Corral, A., Vassilakopoulos, M., & Manolopoulos, Y. (1999). Algorithms for joining R-trees and linear region quadtrees. International Symposium on Spatial Databases, 251-269.

De Berg, M., & al., e. (2000). Computational geometry. Computational geometry. s.l. : Springer.

De Boer, W. H. (2000). Fast terrain rendering using geometrical mipmapping Unpublished paper. from http://www. flipcode. com/articles/article geomipmaps. pdf.

Escrivà, M., Jaime, A., Gutiérrez, A., & Beltrán-Meneu, M. (2017). Geometría 3D y talento matemático: Uno.

Evans, W., Kirkpatrick, D., & Townsend, G. (2001). Right-triangulated irregular networks. . Springer: Algorithmica, Vol. 30, 264-286.

Fuentes, A. R., Garcia, I. G., Reyes, Y. M., Perdomo, G. N., Guanche, P. M., & Palumbo, A. (2018). SONOHISTEROGRAFÍA 3D. Seram.

Hernández Muñoz, Ó., Sánchez Ortiz, M. A., Calvo Manuel, A. M., Martín, M., Adolfo, P., Legido García, M. V., . . . de las Heras Vera, D. (2018). Aplicaciones de las nuevas herramientas de visualización de modelos 3D y realidad virtual en Internet a la docencia de la conservación y restauración del patrimonio, el diseño, y las bellas artes.

Herrera, S. I., Sanz, C. V., Morales, M. I., Palavecino, R., Maldonado, M., Irurzun, I., . . . Suárez, G. I. (2018). M-learning con Realidad Aumentada basada en Objetos 3D. Paper presented at the XX Workshop de Investigadores en Ciencias de la Computación (WICC 2018, Universidad Nacional del Nordeste).

Lindstrom, P., & al, e. (1996). Real-time, continuous level of detail rendering of height fields. Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, 109-118.

Mar, O., Santana, I., & Gulín, J. (2017). Competency assessment model for a virtual laboratory system and distance using fuzzy cognitive map. Revista Investigación Operacional, 38(2), 170-178.

Over, M., & al, e. (2010). Generating web-based 3D City Models from OpenStreetMap: The current situation in Germany. 6, s.l. . Computers, Environment and Urban Systems, Vol. 34, 496-507.

Pajarola, R. (1998). Large scale terrain visualization using the restricted quadtree triangulation. Visualization'98. Proceedings, 19-26.

Pajarola, R. (2002). Overview of quadtree based terrain triangulation and visualization. Technical Report UCI-ICS TR 02-01.

Samet, H. (1990). Applications of Spatial Data Structures.

To, D., Lau. , WH., R., & Green, M. (2001). An adaptive multiresolution method for progressive model transmission. . Teleoperators and Virtual Environments Vol. 10, 62-74.

Ulrich, T. (2002). Rendering massive terrains using chunked level of detail control. SIGGRAPH Course Notes, Vol. 3.

Valle-Martínez, Y., & Ortiz-Rojas, J. (2011). Sistemas de Información. Representación de Superficies de Terrenos para su Visualización en Tres Dimensiones. Ciencias de la Información, Vol. 42, 57-64.

Vega, G. E. (2013). Algoritmo de recortes y de niveles de detalles para el incremento de la velocidad de visualización de modelos 3D en dispositivos de bajo coste. . 3ciencias: cuadernos de desarrollo aplicados a las TIC, Vol. 2(No.2).

Zhu, C., Tan, E. C., & Chan, K. (2003). 3D Terrain visualization for Web GIS. Map Asia, 13-15.