CEIG - Spanish Computer Graphics Conference (2017)
F. J. Melero and N. Pelechano (Editors)

Direct Volume Rendering of Stack-Based Terrains

Alejandro Graciano[1] , Antonio J. Rueda[1] y Francisco R. Feito[1]

[1]Departamento de Informática, Escuela Politécnica Superior, Universidad de Jaén, España
{graciano, ajrueda, ffeito}@ujaen.es

Figura 1: *Un modelo original y diferentes ejemplos de operaciones visuales aplicadas: ocultamiento de capas, aplicación de una ortofotografía como textura, visualización de catas y corte transversal (de izquierda a derecha)*

Abstract

Traditionally, the rendering of volumetric terrain data, as many other scientific 3D data, has been carried out performing direct volume rendering techniques on voxel-based representations. A main problem with this kind of representation is its large memory footprint. Several solutions have emerged in order to reduce the memory consumption and improve the rendering performance. An example of this is the hierarchical data structures for space division based on octrees. Although these representations have produced excellent outcomes, especially for binary datasets, their use in data containing internal structures and organized in a layered style, as in the case of surface-subsurface terrain, still leads to a high memory usage. In this paper, we propose the use of a compact stack-based representation for 3D terrain data, allowing a real-time rendering using classic volume rendering procedures. In contrast with previous work that used this representation as an assistant for rendering purposes, we suggest its use as main data structure maintaining the whole dataset in the GPU in a compact way. Furthermore, the implementation of some visual operations included in geoscientific applications such as borehole visualization, attenuation of material layers or cross sections has been carried out.

CCS Concepts

•***Computing methodologies*** → ***Volumetric models;*** *Scientific visualization; Graphics processors;*

1. Introducción

El modelado y visualización de terrenos es un aspecto fundamental de muchas aplicaciones geocientíficas, en campos como la Hidrología, Sismología, Vulcanología o Estratigrafía, pero también es de utilidad en otros ámbitos no científicos como el diseño de videojuegos y simuladores o la producción de películas de animación y efectos especiales. Tradicionalmente, la representación de terrenos se lleva a cabo por medio del modelado de la geometría de su superficie a partir de modelos digitales de elevación (DEMs, por sus siglas en inglés). Habitualmente, se suele realizar una mejora de su aspecto visual introduciendo sombreados, texturas o códigos de colores para diferentes materiales o alturas. La principal desventaja de usar DEMs es que está limitado a un único valor de elevación por cada celda del raster (lo que se conoce como 2.5D). Como consecuencia, los DEMs no son aptos para modelar fenómenos complejos de la superficie como arcos de roca o cuevas. Además la mejora de las tecnologías de adquisición de datos [MSGE14] [GWZ*16] ha proporcionado datos geológicos del subsuelo que requieren de un modelo más general que los DEMs. Por ello, las aplicaciones geocientíficas [MHW*12] y los sistemas de información geográfica en 3D [GRA16] [Pit16] usan representaciones basadas en vóxels para modelar este tipo de datos. Sin embargo, esta representación volumétrica plantea un problema que puede ser relevante durante el procesamiento y la visualización de modelos de alta resolución: su gran consumo de memoria.

Un enfoque más eficiente es extender los DEMs para que, en lugar de almacenar un único valor por cada coordenada *xy*, almacene una secuencia vertical de intervalos. Cada uno de estos intervalos

DOI: 10.2312/ceig.20171207

será un *stack* de vóxels compactados de igual atributo. Este planteamiento no es nuevo y fue introducido por Benes y Forsbach con el término de *Representación de Terrenos Basada en Stacks* (SBRT) en el contexto de la simulación de terrenos [BF01]. Una de las ventajas de esta representación es que mantiene la simplicidad de los DEMs y, por lo tanto, su facilidad a la hora de implementar operaciones raster. Además, el uso de esta representación tanto para una visualización eficiente como para realizar operaciones sobre ella, sin la necesidad de usar estructuras auxiliares adicionales, es importante para muchas aplicaciones geocientíficas.

En este artículo proponemos el uso de esta representación para la visualización de terrenos y sus estructuras geológicas, adaptando la técnica de renderizado de volúmenes raycasting. Nuestro método de visualización permite operaciones tales como cortes transversales, atenuación de capas de materiales o la visualización selectiva de catas geológicas en tiempo real. La estructura se introduce de forma compacta y completa en la memoria de la GPU. Esto evita el trasiego de datos entre la CPU y la GPU, considerado uno de los cuellos de botella en cualquier sistema de visualización.

Durante el artículo distinguiremos entre terrenos volumétricos/3D y estructuras geológicas o del subsuelo. Los primeros se refieren a la superficie del terreno y a elementos 3D como cuevas o acantilados, mientras que las segundas se refieren a componentes geológicos o del subsuelo como materiales o acuíferos.

El resto del artículo se organiza de la siguiente manera: en la Sección 2 se realiza una revisión de los trabajos relacionados. En la Sección 3 se describe la representación de terrenos basada en stacks. En la Sección 4 se explica el proceso de renderizado seguido, así como una serie de operaciones visuales implementadas. La Sección 5 realiza un análisis de un conjunto de esquemas de memoria en GPU para la representación. Finalmente, concluimos el trabajo en la Sección 6.

2. Trabajos relacionados

Son numerosos los trabajos centrados en la visualización tanto de la superficie como de características volumétricas de terrenos. La forma clásica de representar y visualizar la superficie terrestre es transformando un DEM a una red irregular de triángulos (TIN) y enviándola directamente a la GPU para su renderizado. Ejemplos como [LH04] [DSW09] [NSOJA11] [NKF*16] muestran este enfoque.

En cuanto a la visualización de terrenos volumétricos, se pueden seguir dos estrategias: por un lado se puede realizar una conversión a mallas de triángulos usando métodos como Marching Cubes (MC) [LC87]. La conversión se puede realizar sobre modelos de vóxel directamente [Gei08] o sobre representaciones implícitas [SBD15], entre otros. Esta solución tiene como principal desventaja que, si se quisieran realizar un conjunto de operaciones de visualización en tiempo real; por ejemplo, añadir a la visualización de un conjunto de capas de materiales un corte transversal o un conjunto de catas geológicas (ver Figura 1), implicaría el recálculo de nuevas superficies o el mantenimiento de geometría redundante que podría ocupar mucho más espacio que una SBRT. Otra estrategia corresponde a la utilización de técnicas de visualización directa de volúmenes (DVR) como raycasting. Como ejemplo, existen trabajos haciendo uso de texturas 3D [PSH*09] [NKP14] [HFG*12], de vectores multiescalares [WYZG11] o a partir de superficies de difusión [TSNI10].

Para realizar una revisión más detallada sobre el estado del arte del modelado y visualización de terrenos 3D y del subsuelo, remitimos al lector al trabajo de Natali *et al.* [NLP12].

Hasta ahora, la SBRT ha sido usada en algunos trabajos centrados principalmente en la visualización a nivel de superficie. Benes y Forsbach, precursores de esta representación, la utilizaron para modelar la erosión termal sobre terrenos. En este trabajo se realizó un renderizado del terreno a partir de mapas de altura [BF01]. Peytavie *et al.* [PGGM09] presentaron una visualización más realista proponiendo una representación híbrida en la que la SBRT (referida como *representación de stacks de capas de materiales*) servía como estructura soporte para la generación de una representación implícita de la superficie del terreno. Este trabajo fue extendido por Loffler *et al.* [LMS11] al acelerar la generación de la representación implícita, obteniendo tiempos reales de visualización para modelos de alta resolución. Natali *et al.* [NLP12] presentaron un sistema para la enseñanza de conceptos geológicos a partir de esbozos, siendo los primeros en beneficiarse de la SBRT para la representación de elementos geológicos del subsuelo. Sin embargo, el uso de mapas de altura excluye elementos como acantilados, arcos de piedra, cuevas, acuíferos o yacimientos petrolíferos.

En los trabajos anteriores, la representación basada en stacks juega un papel secundario como estructura auxiliar para generar otras representaciones. En este artículo proponemos su uso como estructura principal para la representación de elementos geológicos tanto de la superficie como del subsuelo. Además, describimos un algoritmo de renderizado directo en GPU en tiempo real con una buena calidad visual para aplicaciones geocientíficas.

3. Representación de terrenos basados en stacks

Esta representación se puede considerar como una generalización de los mapas de altura. Como se ha descrito anteriormente, mientras que los mapas de altura contienen un único valor de altura por coordenada *xy*, la SBRT almacena un conjunto de intervalos. Cada uno está formado por una altura inicial y un atributo, de forma similar a una codificación run-length (Figura 2).

Además de para codificar terrenos volumétricos, esta representación es ideal para modelar un conjunto de datos obtenidos a partir de catas geológicas. En una cata se extrae una muestra física del terreno en forma de cilindro en la que se suceden los distintos materiales del subsuelo. Una forma común de obtener un modelo del subsuelo a partir de un conjunto de catas es por medio de procedimientos de inter-extrapolación dando como resultado un modelo por capas [Tur06]. Una representación de terrenos basada en stacks se ajusta a la perfección a este tipo de modelos ya que cada celda puede almacenar los datos proporcionados por una cata, incluyendo tanto materiales (agua, petróleo, arcilla, roca, etc.) como propiedades geológicas (densidad, permeabilidad, resistividad, etc.).

En la Tabla 1 se muestra una comparación de la memoria consumida por una representación basada en stacks, un modelo de vóxel

Tabla 1: *Comparación del consumo de memoria*

Dataset	Número de vóxels	Uso de memoria (MB)			Compresión (%)	
		Modelo de vóxel	Octree (6 niveles)	SBRT	SBRT - Modelo de vóxel	SBRT - Octree
Terreno 1	200×250×320	61.035	20.004	2.628	95.694	86.631
Terreno 2	200×250×185	35.286	10.810	3.291	90.672	69.553

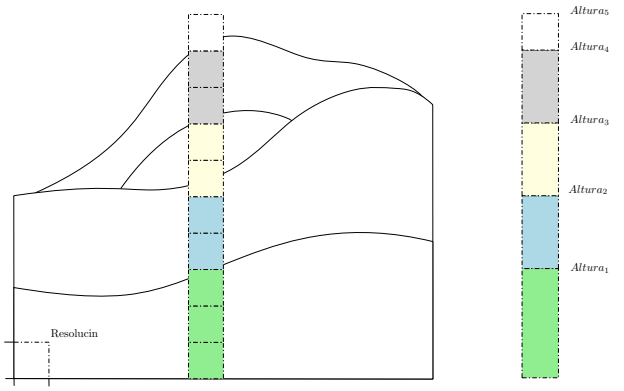

Figura 2: *Esquema de una SBRT. Los vóxeles pertenecientes a una posición xy específica y el stack resultante*

y un octree para dos datasets con datos volumétricos del subsuelo. En el caso de los octrees, la compactación se realiza a partir de vóxeles de un mismo material en regiones cúbicas, no permitiéndose ninguna tolerancia de error en la compactación de éstos o en las estructuras de stacks. Los resultados muestran cómo la SBRT tiene un consumo de memoria claramente menor que los otros métodos: inferior al 10 % del de un modelo de vóxel equivalente y entre un 13 % y un 30 % del requerido por un octree. Una cuestión que debe considerarse es que generalmente los octrees codifican datasets con dimensiones potencia de dos. Esto implica que el dataset Terreno 1 (Tabla 1) tiene que ser ampliado a un dataset de 512 × 512 × 512 dimensiones lo que influye notablemente en el consumo de memoria. A pesar de esto, el uso de una representación basada en stacks para otro tipo de datos volumétricos como los médicos no es adecuado, ya que no están organizados en una disposición por capas.

Otra desventaja de usar estructuras jerárquicas como el octree para la representación de terrenos es su poca utilidad en algunas aplicaciones geocientíficas como las simulaciones físicas [ŠBBK08], ya que tras su ejecución es posible que la estructura deba ser completamente recalculada. Por el contrario, una SBRT puede ser usada directamente para simulación y visualización de una manera directa sin la necesidad de un preprocesamiento en ningún caso. Además, las estructuras jerárquicas no soportan de una forma adecuada la mezcla o solapamientos entre materiales, lo que se conoce como materiales heterogéneos [CRTGFFH15]. La extensión de una SBRT para permitir el modelado de estos materiales es mucho más sencilla que la de una estructura jerárquica ya que se podrían realizar a nivel de stack. Por último, permitir una visualización de catas geológicas (Sección 4.3.3), operación muy común en las aplicaciones geocientíficas, necesitaría de un recálculo de la jerarquía cada vez que se añadiera una nueva cata a la visualización. Sin embargo, esta operación se realiza en tiempo real con una SBRT.

4. Método de visualización

El método descrito en esta sección está basado en el conocido algoritmo de visualización en GPU raycasting. La idea principal es lanzar rayos desde la cámara virtual a través de cada pixel del buffer de salida, componiendo los colores muestreados a lo largo del recorrido [HKRs*06]. A continuación se describen las principales peculiaridades de la adaptación del raycasting para el renderizado de una SBRT:

- Como geometría proxy decidimos usar la caja envolvente para mantener la simplicidad de la solución y porque se ajusta bien a la forma de los datasets de terrenos.
- Los datos volumétricos son codificados en una o dos estructuras (texturas o buffers) en las que primero se realiza extracción del stack para después iterar por sus intervalos (ver Sección 4.2).
- Como método de composición del muestreo (también llamado alpha-blending), se puede utilizar un orden front-to-back o back-to-front. Decidimos usar el primero ya que es más propicio a optimizaciones como la terminación de rayo anticipada.
- Realizamos un avance del rayo adaptativo. Cuando el rayo se encuentre en un espacio vacío, la longitud del salto es igual a la resolución de las celdas del terreno, mientras que para evitar artefactos y una vez el rayo ha colisionado con un material no vacío, aumentamos la frecuencia de muestreo.
- El criterio de parada del algoritmo está basado en la opacidad compuesta. Si la opacidad es mayor que un umbral establecido $(1-\varepsilon)$, se puede asumir que las nuevas contribuciones a la composición serán irrelevantes.

4.1. Cálculo de los vectores normales a la superficie

Una vez se ha muestreado un material, es necesario realizar una correspondencia con un color. Este procedimiento es normalmente llevado a cabo a través de una Función de Transferencia (FT) [CMR08]. En nuestro caso, almacenamos la FT en una tabla de consulta de valores discretos. Para dotar de más realismo a la visualización se ha añadido un modelo de iluminación a la escena. La iluminación es llevada a cabo mediante un método BRDF usando un modelo Lambertiano difuso. La ecuación 1 define este método. Los componentes i_a e i_d controlan la intensidad de la iluminación ambiente y difusa respectivamente, k_d es la reflectancia del modelo

Lambertiano, $\vec{L}$ es el vector dirección del objeto hacia la fuente de luz y $\vec{N}$ el vector normal a la superficie en un punto p.

$$I_p = i_a + k_d(\vec{L}.\vec{N})i_d \quad (1)$$

El cálculo de los vectores normales se realiza normalmente a partir de datos volumétricos que representan valores de intensidad, como por ejemplo es el caso de las imágenes médicas. El enfoque típico es calcular el gradiente de los valores de intensidad a partir de procedimientos como diferencias centrales [HKRs*06] en el espacio del objeto. Este enfoque puede dar como resultado modelos con una superficie con problemas de aliasing para datos binarios (vóxel ocupado o no ocupado). Otra forma sencilla de calcular el vector normal a un vóxel en este espacio es realizando una convolución 3D respecto a él. Cada celda vecina contiene un vector unitario que une el centro de la máscara de convolución con ésta, siendo el resultado de la convolución la suma de aquellos vectores que se encuentran en una celda no ocupada (que estén en el límite de la superficie). La suavidad de la superficie y el tiempo consumido por el procedimiento es directamente proporcional al tamaño de la máscara. En el caso de su utilización en una SBRT este método podría causar un cuello de botella en el rendimiento ya que por cada nuevo acceso en un modelo de voxel original se debería, en el peor de los casos, ubicar un nuevo stack y recorrerlo secuencialmente. Muchos de los trabajos centrados en el cálculo del gradiente de la superficie para datos discretos toman un enfoque basado en el espacio de imagen [YCK92] [KCOY03]. En este caso, las normales se calculan obteniendo los gradientes verticales y horizontales del mapa de profundidad. Nuevamente, el resultado puede provocar aliasing ya que un mismo vóxel cercano a la cámara puede ser rasterizado por varios pixeles. Por el contrario, un enfoque basado en el espacio del objeto tomaría en cuenta la geometría real de los datos.

La estrategia que proponemos es híbrida: calculamos el vector normal con el método basado en el espacio del objeto explicado anteriormente, pero en el espacio de imagen. Usamos una estrategia en diferido en la que, en una primera pasada en la GPU, obtenemos el color devuelto por la FT y almacenamos en un mapa de profundidad la distancia de la cámara a la colisión del rayo, y en una segunda calculamos los vectores normales y componemos el color final de los píxeles. A partir del mapa de profundidad realizamos una proyección inversa para obtener la posición 3D del centro de las celdas de la máscara. Un vóxel se considera ocupado si no está ocluido por el modelo real. Esto es, si al proyectar de nuevo el valor obtenido por la proyección inversa, se obtiene un valor de profundidad menor que la distancia de la cámara a la posición de la proyección inversa (Figura 3). El tiempo requerido por esta estrategia es únicamente dependiente del tamaño del buffer de salida. Nótese que el número de vectores normales que se pueden calcular con este método es limitado en comparación con los enfoques basados en gradiente. Sin embargo, en la práctica, son suficientes para una buena calidad visual en aplicaciones científicas. La Figura 4 resume este enfoque.

Algunos autores sugieren realizar un precálculo de los vectores normales y mandarlas en un buffer al método de raycasting [SWBG06] [BLD12]. Sin embargo, esto implicaría un aumento significativo de la memoria consumida en comparación con las técnicas explicadas anteriormente.

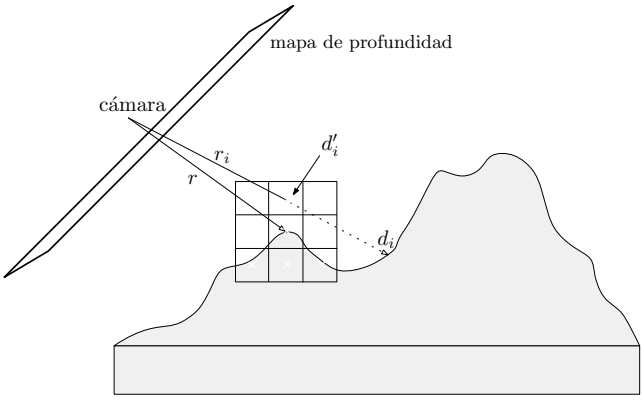

Figura 3: *Cálculo de un vector normal en un espacio de imagen. El rayo r_i produce dos valores de profundidad, uno real (d_i) y uno estimado (d'_i) con los que se calculan los valores de la máscara de convolución. En este caso el vóxel que proyecta d'_i formaría parte de la superficie ya que el proyectado en d_i se encuentra detrás.*

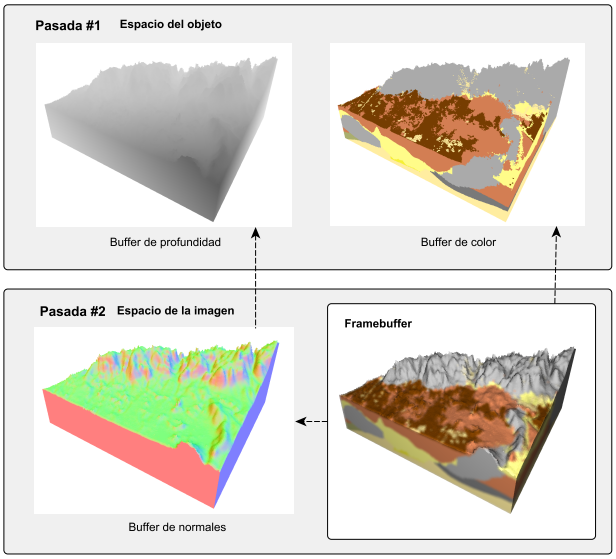

Figura 4: *Estrategia híbrida usada para el cálculo del sombreado. Las líneas punteadas indican una relación de uso*

4.2. Codificación de la SBRT y sampleado

Un aspecto que produce un gran impacto en el rendimiento de cualquier método de visualización es el acceso a las texturas. Acceder a un texel es una de las operaciones que más tiempo consumen en la GPU. Afortunadamente, el acceso a textura exhibe una coherencia espacial y temporal que puede ser explotada para mejorar su rendimiento. Las arquitecturas de las GPUs modernas usan sistemas de cacheado para las texturas siguiendo una estrate-

gia LRU [AMHH09]. Además, para una mayor mejora se pueden utilizar estrategias de *swizzling* [WYC14]. Estas estrategias buscan mejorar la coherencia espacial organizando los datos en secuencias no lineales como las curvas Peanno-Hilbert o de Morton. También existen soluciones en las que los datos se ordenan de forma favorable al raycasting [JGY*12].

A continuación, mostraremos diferentes esquemas de memoria en GPU para almacenar la representación de terrenos basada en stacks. El procedimiento para muestrear un material particular se divide en dos fases: la identificación del stack y su iteración. Estas dos fases son comunes en todas las estrategias planteadas; la única diferencia es el número de texturas (o buffers) para codificar cada una de ellas.

En un primer enfoque (Enfoque 1, Figura 5.b) mantenemos una pareja de texturas 2D: una textura de índices y otra textura de intervalos. El objetivo de la primera es servir como índice espacial para los stacks almacenados en la textura de intervalos. La textura de índices tiene la misma dimensión que la cuadrícula soporte de la SBRT ($grid_{ancho} \times grid_{alto}$); de esta forma, un stack particular se obtiene tras proyectar las componentes xz de la posición del rayo en esta cuadrícula. El texel obtenido tras esa proyección contiene un puntero (un índice 1D) al inicio de un stack y su tamaño en la textura de intervalos en sus componentes R (R_1) y G (G_1). La textura de intervalos almacena los stacks en un orden por filas. En esta textura también usamos las componentes R y G del texel. En la componente R (R_2) codificamos el atributo o material y en la G (G_2) la altura acumulada del intervalo. Por lo tanto, para obtener el intervalo dada la altura de la posición del rayo (y), se debe iterar G_1 veces desde la posición R_1 y comparar esta altura con G_2.

En el segundo enfoque sólo usamos una textura 2D en la que se codifican dos niveles; una para los índices y otra para los intervalos. La textura se construye obteniendo primero el número máximo de intervalos de entre todos los stacks, m. Luego, se crean regiones de $\lceil\sqrt{m}\rceil \times \lceil\sqrt{m}\rceil$ dimensiones, donde $\lceil x \rceil$ es el operador de redondeo superior. Estas regiones actúan como *supertexels* que proporcionan un esquema de pseudoíndices. Por lo tanto, las dimensiones reales de la textura se calculan como:

$$\begin{aligned} textura_{ancho} &= grid_{ancho} * \lceil\sqrt{m}\rceil \\ textura_{alto} &= grid_{alto} * \lceil\sqrt{m}\rceil \end{aligned} \tag{2}$$

Esta estrategia permite una forma sencilla de obtener un determinado stack ya que todas las regiones son cuadradas. De forma similar al Enfoque 1, la posición del rayo tiene que ser proyectada en la cuadrícula de soporte, pero en lugar de obtener un texel, debemos obtener el inicio de una región. Los intervalos de la región serán, por lo tanto, iterados de forma similar al enfoque anterior. Esta textura también codifica el atributo y la altura de un intervalo en las componentes R-G. De forma alternativa, los stacks se pueden almacenar en regiones no cuadradas buscando rectángulos ($rec_{ancho} \times rec_{alto}$) con área mínima. Este problema de optimización se puede formular de la siguiente forma:

$$\begin{aligned} \underset{rec_{ancho}, rec_{alto}}{\text{argmin}} \quad & rec_{ancho} + rec_{alto} \\ \text{sujeto a} \quad & rec_{ancho} * rec_{alto} >= m \\ & rec_{ancho} > 0 \\ & rec_{alto} > 0 \end{aligned} \tag{3}$$

Dentro de este enfoque, la organización basada en rectángulos es la preferida cuando m no es valor primo. La Figura 5.c ilustra este enfoque.

Otro posible enfoque es el propuesto por Natali *et al.* [NKP14]. Usaron una textura 3D de $grid_{ancho} \times grid_{alto} \times m$ dimensiones en la que los intervalos son almacenados a lo largo del eje z (Enfoque 4, Figura 5.e). En cambio, nosotros usaremos una textura 2D de $(grid_{ancho} * m) \times grid_{alto}$ dimensiones para comprobar si su coherencia espacial mejora a la textura 3D. (Enfoque 3, Figura 5.d). El uso de una textura 1D podría ser insuficiente para almacenar todos los intervalos debido a su restricción de tamaño. El proceso de identificación de un stack es similar a los anteriormente explicados.

A pesar de que en los métodos de DVR lo más habitual es usar texturas para almacenar los datos volumétricos; las actuales APIs gráficas proporcionan otras maneras de enviar y almacenar datos en la GPU como los Uniform Buffer Objects (UBO). Reciente, en su versión 4.3, OpenGL ha incluido un nuevo método para almacenar grandes cantidades de datos en la GPU: los Shader Storage Buffer Objects (SSBO). Realmente, los SSBOs pueden ser vistos como una versión mejorada y más flexible de los UBOs. Las implementaciones de OpenGL deben soportar UBOs de al menos 16 KB (sólo de lectura) mientras que el tamaño mínimo que un SSBO debe permitir es de 128 MB (tanto de lectura como de escritura). Además, la especificación de los SSBOs permite almacenar en ellos un array de tamaño no fijo. Esta característica es interesante ya que una SBRT puede ser codificada como un array de intervalos de forma natural. De forma conjunta a los SSBOs, OpenGL introdujo la organización std430 que empaqueta arrays escalares de una forma mucho más eficiente que la tradicional std140 [SWJN16]. Basándonos en esta estructura, proponemos un enfoque para codificar una SBRT en la memoria de la GPU (Enfoque 5). De forma similar al Enfoque 1, usamos dos SSBOs, uno para índices y otro para intervalos (Figura 5.f).

Para aclarar estos enfoques usaremos un ejemplo ilustrativo. Dado un terreno con una representación basada en stacks, con una cuadrícula soporte de $grid_{ancho} \times grid_{alto}$ celdas y un número máximo de intervalos (m) de 5 (Figura 5.a), los patrones de memoria propuestos se describen a continuación: en el Enfoque 1, la textura de índices tiene las mismas dimensiones que la cuadrícula del terreno ($grid_{ancho} \times grid_{alto}$) y los intervalos se añaden de forma secuencial en la textura de intervalos. El tamaño de esta textura se calcula obteniendo primeramente el número total de intervalos de la SBRT. Finalmente, asignando este valor a m en la Ecuación 3 se calcula las dimensiones de la textura. En contrapartida, si el valor es primo se puede usar la Ecuación 2. Asimismo, este valor puede ser aumentado para poder usar la Ecuación 3. El Enfoque 1 puede desperdiciar algo de memoria ya que las últimas celdas de la última fila normalmente quedan sin ser ocupadas. Para el Enfoque 2 es necesario calcular el tamaño de los *supertexels*. El stack más grande

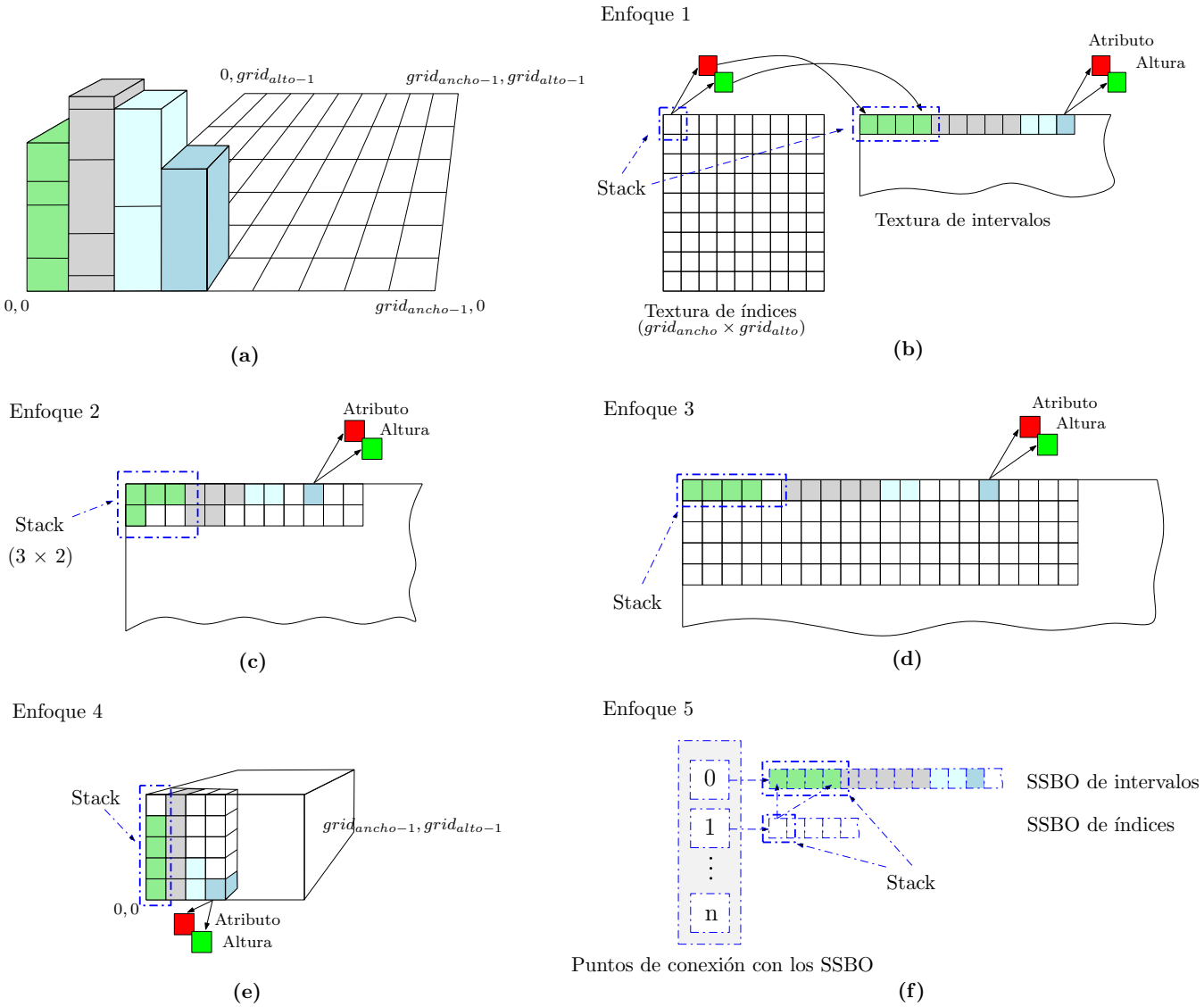

Figura 5: *Patrones de memoria planteados (b, c, d, e, f) para una representación de terrenos basada en stacks (a). Al ser el mayor stack de 5 intervalos, aplicando la Ecuación 3 se obtienen regiones de (3 × 2) texels para el Enfoque 2*

determina este valor (5 intervalos en este ejemplo). Por lo tanto, la textura contendrá regiones de 3 × 2 texels. Los Enfoques 3 y 4 son similares. Finalmente, para el Enfoque 5, dos arrays que contienen los índices y los intervalos se almacenan en dos SSBOs separados. El ejemplo se muestra en la Figura 5.

4.3. Operaciones visuales

En esta subsección se describen un conjunto de operaciones comunes en aplicaciones geocientíficas y su implementación en nuestro sistema. Este tipo de operaciones como la visualización selectiva de catas geológicas o la ocultación de capas de materiales, proporcionan a los geocientíficos herramientas para poder tomar decisiones a simple vista.

4.3.1. Visualización de estratos

Cada una de las capas de materiales puede ser atenuada o eliminada de la visualización para poder inspeccionar estructuras internas de una forma más clara. Esto se lleva a cabo simplemente modificando la opacidad del color del material en la tabla de consulta. Al usar raycasting para el renderizado, no es necesario generar una nueva geometría al ocultar capas, en contraste con otros métodos. En la Figura 6 se muestra un ejemplo.

4.3.2. Visualización de cortes transversales

Otra forma de visualizar estructuras internas es vía cortes transversales geológicos. Estos cortes suelen ser verticales aunque nuestro sistema permite el uso de cualquier plano orientado libremente

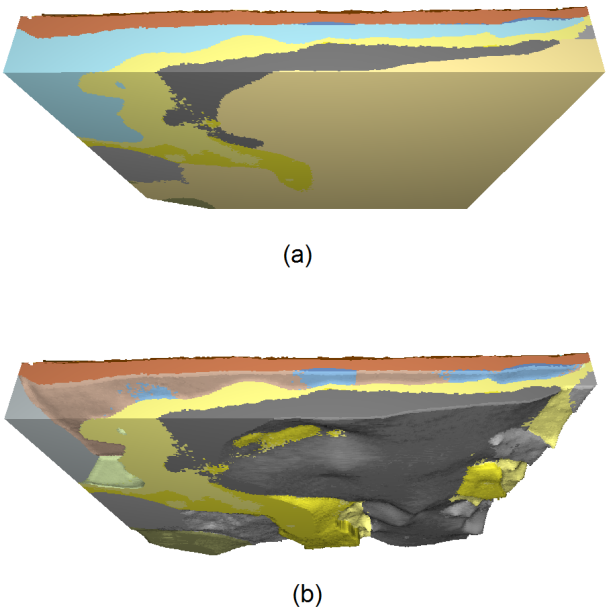

Figura 6: *Ejemplo de atenuación de capas de materiales (b) de un dataset original (a)*

en el espacio 3D. A partir de estos cortes, los geocientíficos son capaces de estudiar la distribución de los tipos de roca, su edad o relación entre las distintas capas. Cuando se elige un plano de corte, el raycasting comenzará desde la colisión del rayo con el plano. La Figura 7 ilustra esta característica.

4.3.3. Visualización de catas geológicas

Los datos de catas geológicas se obtienen perforando el núcleo de roca y contienen información sobre la litología, espesor de los estratos o propiedades físicas de los materiales que son muy relevantes para los estudios geológicos. Como se ha dicho anteriormente, cada uno de los stacks de una SBRT puede codificar una cata. Por tanto para implementar la funcionalidad de visualización de catas geológicas basta con etiquetar los stacks correspondientes. La Figura 8 muestra un ejemplo de una visualización de catas geológicas.

4.3.4. Aplicación de texturas al terreno

El sistema también proporciona una operación para añadir una textura al terreno a partir de ortofotografías o mapas topográficos. El color del pixel se obtiene directamente de la imagen y de forma opcional se puede combinar con el color del material. En la Figura 9 se muestra un ejemplo.

5. Análisis del rendimiento

Para medir el rendimiento de los esquemas de memoria propuestos en la Sección 4.2, se realizó un conjunto de experimentos. Medimos y comparamos los marcos por segundos (fps, por sus siglas en inglés) renderizando los diferentes esquemas así como la cantidad de memoria requerida. Los experimentos se realizaron sobre tres datasets con diferentes dimensiones de cuadrícula y tamaño de stacks; 200×250 con un número máximo de 10 intervalos por stack (Dataset A), 400×500 con 15 intervalos máximos (Dataset B) y 800×1000 con 16 intervalos máximos (Dataset C), ver Figura 10. Estos datos se obtuvieron de la base de datos DINOloket que proporciona datos del subsuelo de distintos lugares de Los Países Bajos [GMV*13]. Para medir el impacto de la dirección del ray-

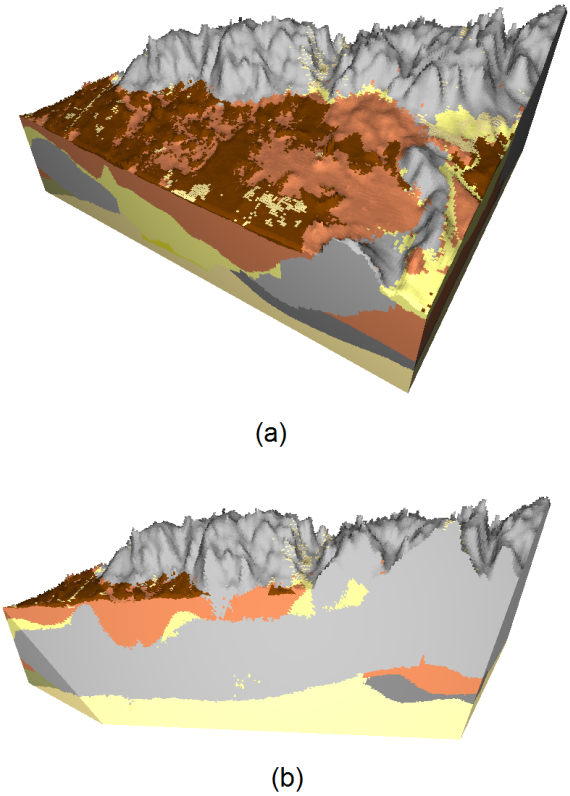

Figura 7: *Corte transversal (b) de un modelo original (a)*

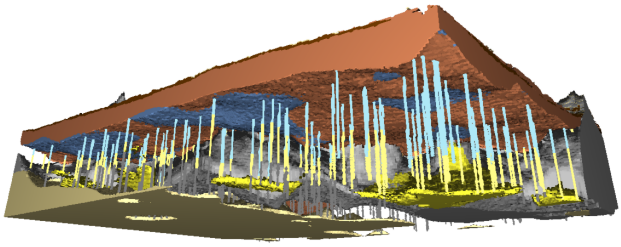

Figura 8: *Ejemplo de visualización de catas geológicas mostradas como cilindros de materiales (azul y amarillo)*

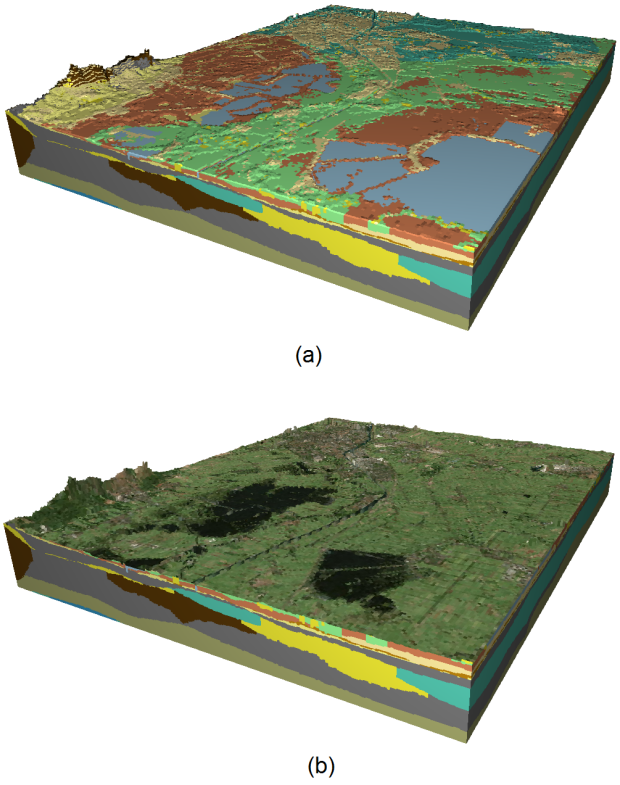

Figura 9: *Ejemplo de aplicación de una ortofotografía. Modelo original (a) y modelo con ortofotografía aplicada como textura (b)*

casting, se hizo el renderizado desde diferentes posiciones de la cámara virtual.

La configuración hardware usada consistió en un PC equipado con un procesador Intel Core i7-4790 a 3.60 GHz, con 16 GB de memoria RAM y una tarjeta gráfica NVIDIA GeForce GTX 970. El sistema ha sido implementado en C++ con OpenGL 4.5 como librería gráfica 3D. El método se ha ejecutado en una ventana de 1920 $\times$ 1080 píxeles.

La Figura 11 muestra los fps medios medidos para cada posición de la cámara: los fps se midieron desde 16 posiciones distintas para cada dataset usando cada uno de los enfoques planteados.

Puede verse que los Enfoques basados en una única textura (Enfoques 2, 3 y 4) tienen un rendimiento peor que el resto debido a la falta de una organización propicia para el cacheado. Por otro lado, los Enfoques 1 y 5 obtienen los mejores resultados al utilizar estructuras unidimensionales, lo que puede favorecer a un prefetching y a un cacheado a nivel de stack. El Enfoque 5 mejora ligeramente el rendimiento del Enfoque 1, obteniendo fps comprendidos entre 279 y 61 mientras que los del Enfoque 1 están comprendidos entre 256 y 55.

El Enfoque 3 presenta ciertas restricciones. Este enfoque no puede ser usado para datos especialmente grandes. Las tarjetas gráficas soportan texturas de un tamaño máximo, siendo el límite de la usada en los experimentos de 16384$\times$16384 para una textura 2D. Por esto, para codificar un dataset con una dimensión de cuadrícula de 1024$\times$1024 y un número máximo de intervalos de 16, este enfoque necesitaría al menos una textura 2D de 17408$\times$1024, que excedería los límites de la tarjeta.

Los requerimientos de memoria utilizada y de memoria necesaria por los enfoques pero que no almacenan datos reales (memoria desperdiciada) se muestran en la Tabla 2, siendo los Enfoques 1 y 5 los más eficientes empaquetando los datos. En esos enfoques, el almacén de intervalos (textura o SSBO) no necesita un tamaño fijo para identificar el inicio de un stack debido al uso del índice espacial. No obstante, como se puede apreciar, el Enfoque 1 requiere una cantidad extra de memoria para el Dataset A que puede ser ignorada. Esto se debe a que el número máximo de intervalos, 309433 en este caso, es un valor primo (necesitando una textura de 2D de tamaño 479$\times$646). Los resultados del resto de enfoques indican un notable desperdicio de memoria.

Con todo esto, podemos afirmar que el enfoque preferido de entre los probados es el basado en SSBOs (Enfoque 5) al obtener los mejores resultados tanto en tiempo de visualización como en consumo de memoria.

En cuanto a resultados visuales cabe destacar que no existen diferencias entre la visualización mostrada por cada uno de los enfoques.

6. Conclusiones

En este trabajo se ha presentado un sistema de visualización en tiempo real para modelos geológicos tanto a nivel de superficie como del subsuelo. Para conseguir esto, se hace uso de una representación de terrenos basada en stacks. Se ha destacado su bajo requerimiento de memoria en comparación con otras estructuras como los modelos de vóxel o los octrees, validando su uso para el manejo y la visualización de geomodelos. También se han probado distintos esquemas para almacenar esta representación de manera eficiente en la GPU, alcanzando resultados bastante aceptables para aplicaciones interactivas. En general, el Enfoque 5 planteado (basado en SSBOs) puede ser considerado el mejor en términos de rendimiento. Además los Enfoques 1 (basado en dos texturas) y 5 obtuvieron los mejores resultados en cuanto a memoria usada y relación usada/desperdiciada. Todo esto, unido a los grandes requerimientos de memoria del resto de enfoques, hace del Enfoque 5 el preferido. Conviene señalar, hasta donde sabemos, que el uso de SSBOs para almacenar datos volumétricos aún no ha sido explorado en la literatura. De forma adicional, nuestro artículo presenta un método simple y eficiente para calcular vectores normales a la superficie en el espacio de imagen. Para concluir, hemos mostrado la integración de la SBRT con algunas operaciones visuales de interés en aplicaciones geocientíficas.

Sin embargo, nuestro método de renderizado puede mejorarse de varias formas. Un aspecto importante es la calidad visual. Por ejemplo, aunque el método usado para calcular los vectores normales puede ser adecuado para muchas aplicaciones, no proporciona un sombreado del todo realista. Debido a que se están visualizando datos discretos, se puede usar algún método de filtrado como un filtro de paso bajo en zonas de cambios de material. Como ya se

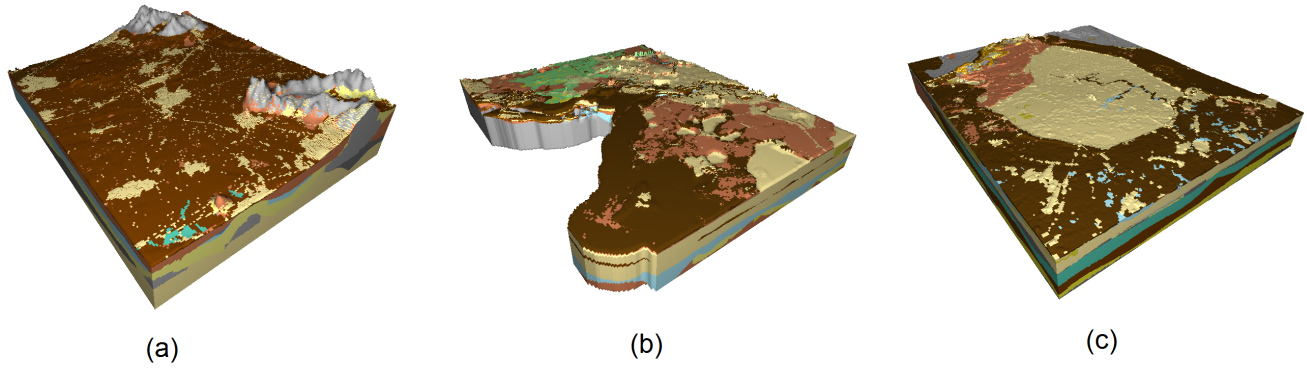

Figura 10: *Datasets usados en los experimentos. Dataset A a la izquierda (a), Dataset B en el centro (b) y Dataset C a la derecha (c)*

Tabla 2: *Requerimientos de memoria de los enfoques planteados en GPU*

	Dataset A			**Dataset B**			**Dataset C**		
	Memoria desperdiciada (MB)	**Memoria consumida (MB)**	**Porcentaje de desperdicio (%)**	**Memoria desperdiciada (MB)**	**Memoria consumida (MB)**	**Porcentaje de desperdicio (%)**	**Memoria desperdiciada (MB)**	**Memoria consumida (MB)**	**Porcentaje de desperdicio (%)**
Enfoque 1	$3{,}81 \times 10^{-6}$	1.5619	$2{,}44 \times 10^{-4}$	0.0000	6.1972	0.0000	0.0000	29.0643	0.0000
Enfoque 2	0.7269	1.9073	38.1133	6.7727	11.4441	59.1809	25.8673	48.8281	52.9762
Enfoque 3	0.7269	1.9073	38.1133	6.7727	11.4441	59.1809	25.8673	48.8281	52.9762
Enfoque 4	0.7269	1.9073	38.1133	6.7727	11.4441	59.1809	25.8673	48.8281	52.9762
Enfoque 5	0.0000	1.5618	0.0000	0.0000	6.1972	0.0000	0.0000	29.0643	0.0000

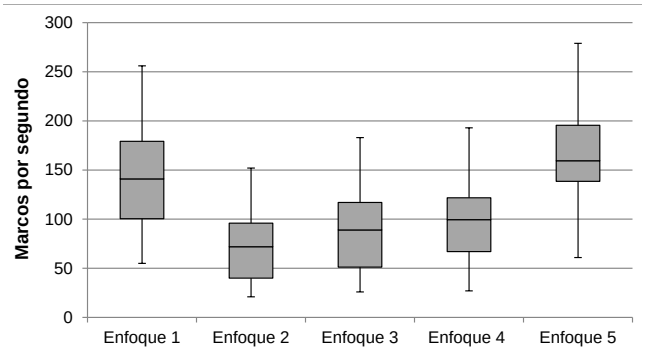

Figura 11: *Resultados medios de tiempo de visualización*

ha indicado, en este trabajo se ha optado por un método de visualización simple, por lo que cualquier mejora en alguna de las fases puede ser añadida.

7. Agradecimientos

Este artículo ha sido parcialmente subvencionado por la Universidad de Jaén a través de la beca predoctoral Acción 15 y por el Fondo Europeo de Desarrollo Regional (FEDER) a través del proyecto TIN2014/58/218/R.

Referencias

[AMHH09] AKENINE-MÖLLER T., HAINES E., HOFFMAN N.: *Real-time rendering*. A. K. Peters, Ltd., 2009. 5

[BF01] BENES B., FORSBACH R.: Layered data representation for visual simulation of terrain erosion. In *Proceedings Spring Conference on Computer Graphics* (2001). doi:10.1109/SCCG.2001.945341. 2

[BLD12] BAERT J., LAGAE A., DUTRÉ P.: Out-of-Core Construction of Sparse Voxel Octrees. *Computer Graphics Forum* (2012). doi:10.1111/cgf.12345. 4

[CMR08] CABAN J. J., MEMBER S., RHEINGANS P.: Texture-based Transfer Functions for Direct Volume Rendering. *IEEE transactions on Visualization and Computer Graphics 14*, 6 (2008), 1364–1371. doi:10.1109/TVCG.2008.169. 3

[CRTGFFH15] CONDE-RODRÍGUEZ F., TORRES J.-C., GARCÍA-FERNÁNDEZ Á.-L., FEITO-HIGUERUELA F.-R.: A comprehensive

framework for modeling heterogeneous objects. *The Visual Computer* (2015). doi:10.1007/s00371-015-1149-0. 3

[DSW09] DICK C., SCHNEIDER J., WESTERMANN R.: Efficient geometry compression for GPU-based decoding in realtime terrain rendering. *Computer Graphics Forum 28*, 1 (2009), 67–83. doi:10.1111/j.1467-8659.2008.01298.x. 2

[Gei08] GEISS R.: Generating complex procedural terrains using the gpu. In *GPU Gems 3*, Nguyen H., (Ed.). Addison-Wesley, 2008, pp. 7–37. 2

[GMV*13] GUNNINK J. L., MALJERS D., VAN GESSEL S. F., MENKOVIC A., HUMMELMAN H. J.: Digital Geological Model (DGM): A 3D raster model of the subsurface of the Netherlands. *Geologie en Mijnbouw/Netherlands Journal of Geosciences 92*, 1 (2013), 33–46. doi:10.1017/S0016774600000263. 7

[GRA16] GRASS DEVELOPMENT TEAM: *Geographic Resources Analysis Support System (GRASS GIS) Software, Version 7.0*. Open Source Geospatial Foundation, 2016. URL: http://grass.osgeo.org. 1

[GWZ*16] GUO J., WU L., ZHOU W., JIANG J., LI C.: Towards Automatic and Topologically Consistent 3D Regional Geological Modeling from Boundaries and Attitudes. *ISPRS International Journal of Geo-Information 5*, 2 (2016), 17. doi:10.3390/ijgi5020017. 1

[HFG*12] HOLLT T., FREILER W., GSCHWANTNER F., DOLEISCH H., HEINEMANN G., HADWIGER M.: SeiVis: An Interactive Visual Subsurface Modeling Application. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (2012), 2226–2235. doi:10.1109/TVCG.2012.259. 2

[HKRs*06] HADWIGER M., KNISS J. M., REZK-SALAMA C., WEISKOPF D., ENGEL K.: *Real-time Volume Graphics*. A. K. Peters, Ltd., 2006. 3, 4

[JGY*12] JÖNSSON D., GANESTAM P., YNNERMAN A., DOGGETT M., ROPINSKI T.: Explicit Cache Management for Volume Ray-Casting on Parallel Architectures. In *EG Symposium on Parallel Graphics and Visualization (EGPGV)* (2012), Eurographics, pp. 31–40. 5

[KCOY03] KADOSH A., COHEN-OR D., YAGEL R.: Tricubic Interpolation of Discrete Surfaces for Binary Volumes. *IEEE Transactions on Visualization and Computer Graphics 9*, 4 (2003), 580–586. doi:10.1109/TVCG.2003.1260750. 4

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1987), SIGGRAPH '87, ACM, pp. 163–169. URL: http://doi.acm.org/10.1145/37401.37422, doi:10.1145/37401.37422. 2

[LH04] LOSASSO F., HOPPE H.: Geometry clipmaps: terrain rendering using nested regular grids. *ACM Transaction on Graphics 1*, 212 (2004), 769–776. URL: http://portal.acm.org/citation.cfm?id=1015799, doi:10.1145/1186562.1015799. 2

[LMS11] LÖFFLER F., MÜLLER A., SCHUMANN H.: Real-time Rendering of Stack-based Terrains. In *Vision, Modeling, and Visualization (2011)* (2011), Eisert P., Hornegger J., Polthier K., (Eds.), The Eurographics Association. doi:10.2312/PE/VMV/VMV11/161-168. 2

[MHW*12] MITASOVA H., HARMON R. S., WEAVER K. J., LYONS N. J., OVERTON M. F.: Scientific visualization of landscapes and landforms. *Geomorphology 137*, 1 (2012), 122–137. doi:10.1016/j.geomorph.2010.09.033. 1

[MSGE14] MATEO LÁZARO J., SÁNCHEZ NAVARRO J. Á., GARCÍA GIL A., EDO ROMERO V.: 3D-geological structures with digital elevation models using GPU programming. *Computers & Geosciences 70* (sep 2014), 138–146. doi:10.1016/j.cageo.2014.05.014. 1

[NKF*16] NIESSNER M., KEINERT B., FISHER M., STAMMINGER M., LOOP C., SCHÄFER H.: Real-Time Rendering Techniques with Hardware Tessellation. *Computer Graphics Forum 35*, 1 (2016), 113–137. doi:10.1111/cgf.12714. 2

[NKP14] NATALI, KLAUSEN T. G., PATEL D.: Sketch-based modelling and visualization of geological deposition. *Computers and Geosciences 67* (2014), 40–48. doi:10.1016/j.cageo.2014.02.010. 2, 5

[NLP12] NATALI M., LIDAL E., PARULEK J.: Modeling terrains and subsurface geology. In *Eurographics 2013-State of the Art Reports* (2012), pp. 155–173. doi:10.2312/conf/EG2013/stars/155-173. 2

[NSOJA11] NOGUERA J. M., SEGURA R. J., OGÁYAR C. J., JOAN-ARINYO R.: Navigating large terrains using commodity mobile devices. *Computers and Geosciences 37*, 9 (2011), 1218–1233. doi:10.1016/j.cageo.2010.08.007. 2

[PGGM09] PEYTAVIE A., GALIN E., GROSJEAN J., MERILLOU S.: Arches: A framework for modeling complex terrains. *Computer Graphics Forum 28*, 2 (2009), 457–467. doi:10.1111/j.1467-8659.2009.01385.x. 2

[Pit16] PITNEY BOWES INC.: Mapinfo Engage3D. http://www.pitneybowes.com/uk/location-intelligence/geographic-information-system/mapinfo-engage3d.html, 1996–2016. 1

[PSH*09] PATEL D., STURE Ø., HAUSER H., GIERTSEN C., EDUARD GRÖLLER M.: Knowledge-assisted visualization of seismic data. *Computers and Graphics (Pergamon) 33*, 5 (2009), 585–596. doi:10.1016/j.cag.2009.06.005. 2

[ŠBBK08] ŠT'AVA O., BENEŠ B., BRISBIN M., KŘIVÁNEK J.: Interactive terrain modeling using hydraulic erosion. *EuroGraphics Symposium on Computer Animation* (2008), 201–210. doi:10.2312/SCA/SCA08/201-210. 3

[SBD15] SCHOLZ M., BENDER J., DACHSBACHER C.: Real-time isosurface extraction with view-dependent level of detail and applications. *Computer Graphics Forum 34*, 1 (2015), 103–115. doi:10.1111/cgf.12462. 2

[SWBG06] SIGG C., WEYRICH T., BOTSCH M., GROSS M.: GPU-Based Ray-Casting of Quadratic Surfaces. *Symposium on Point-Based Graphics* (2006), 59–65. URL: http://dl.acm.org/citation.cfm?id=2386396, doi:10.2312/SPBG/SPBG06/059-065. 4

[SWJN16] SELLERS G., WRIGHT JR R., N. H.: *OpenGL SuperBible. Comprehensive Tutorial and Reference*, 7 ed. Addison Wesley, 2016. 5

[TSNI10] TAKAYAMA K., SORKINE O., NEALEN A., IGARASHI T.: Volumetric modeling with diffusion surfaces. *ACM Transactions on Graphics 29*, 6 (2010), 1. doi:10.1145/1882261.1866202. 2

[Tur06] TURNER A. K.: Challenges and trends for geological modelling and visualisation. *Bulletin of Engineering Geology and the Environment 65*, 2 (2006), 109–127. doi:10.1007/s10064-005-0015-0. 2

[WYC14] WANG J., YANG F., CAO Y.: Cache-aware sampling strategies for texture-based ray casting on gpu. In *Large Data Analysis and Visualization (LDAV), 2014 IEEE 4th Symposium on* (Nov 2014), pp. 19–26. doi:10.1109/LDAV.2014.7013200. 5

[WYZG11] WANG L., YU Y., ZHOU K., GUO B.: Multiscale vector volumes. *ACM Transactions on Graphics 30*, 6 (2011), 1. doi:10.1145/2070781.2024201. 2

[YCK92] YAGEL R., COHEN D., KAUFMAN A.: Normal estimation in 3 D discrete space. *The Visual Computer 8*, 5-6 (1992), 278–291. doi:10.1007/BF01897115. 4