Visualización de Terrenos con Tarjetas de Video Programables GPU

Autor: Lucas Enrique Guaycochea

Tesis de Grado de Ingeniería en Informática
Fecha Defensa: 13 Dic 2011
Facultad de Ingeniería – Universidad de Buenos Aires
Ciudad Autónoma de Buenos Aires, Argentina
lguaycochea@fi.uba.ar

Director: Horacio Antonio Abbate

Facultad de Ingeniería – Universidad de Buenos Aires
Ciudad Autónoma de Buenos Aires, Argentina

Visualización de Terrenos con Tarjetas de Video Programables GPU

Lucas E. Guaycochea & Horacio A. Abbate

Facultad de Ingeniería, Universidad de Buenos Aires

Abstract. La visualización de terrenos es un tema importante y popular en el área de la computación gráfica. El problema que se presenta es lograr una visualización en tiempo real de terrenos extensos aptos para aplicaciones interactivas como simuladores de vuelo. El interés reside en administrar eficientemente la gran cantidad de datos utilizados para modelar la superficie de un terreno extenso y detallado.
La solución presentada consiste en el diseño y la implementación de una técnica simple, que obtiene como resultado una visualización precisa con un buen rendimiento. La técnica propuesta está basada en la técnica Geometry Clipmaps que pertenece al estado del arte del tema. Además, la misma brinda una visualización donde el error de aproximación, producto de utilizar resoluciones menores a la máxima disponible, que es percibido por el usuario se encuentra acotado a un valor umbral que se considera despreciable.
Por último, la implementación aprovecha el poder de procesamiento de las GPU modernas utilizando la librería gráfica Direct3D 10. Los resultados obtenidos permiten la navegación en tiempo real de grandes extensiones de terreno, consumiendo poco procesamiento de CPU.

1 Introducción

En las últimas dos décadas las aplicaciones gráficas para la plataforma PC han tenido un crecimiento exponencial, impulsadas principalmente por la industria de videos juegos. Actualmente se han alcanzado resultados visuales que son casi indistinguibles de la realidad. Sin embargo, los resultados logrados no implican que el crecimiento del área se haya detenido, sino que se sigue trabajando en mejorar distintos detalles para obtener aún más realismo. La rama de las ciencias de la computación que se centra en el estudio y la evolución de las aplicaciones gráficas es llamada computación gráfica.

Dentro del área de la computación gráfica podemos diferenciar dos grandes ramas que se caracterizan por el tiempo disponible para generar las imágenes o animaciones. Por un lado, se encuentran aquellas aplicaciones donde el tiempo no es un limitante y, por ende, se puede invertir el tiempo necesario de procesamiento para generar las imágenes con el realismo deseado, como por ejemplo en la confección de películas animadas. Por otro lado, existen aquellas donde la visualización se debe realizar en

tiempo real, como ocurre en videos juegos, en simuladores de entrenamiento y en aplicaciones interactivas con representación visual de escenarios virtuales o sintéticos. Estas aplicaciones requieren que cada cuadro este listo para mostrarse con la suficiente rapidez como para no afectar la ilusión de continuidad de movimiento en la animación, proveyendo así una adecuada experiencia de usuario interactiva.

El crecimiento de la calidad de las aplicaciones, mencionado en el comienzo, fue viabilizado por el avance de la tecnología. El progreso permitió un incremento en forma sostenida de las capacidades tanto de almacenamiento como de procesamiento; desde los primeros grandes sistemas que consistían en hardware gráficos especiales, hasta las placas gráficas o tarjetas de video que son comunes hoy en día. En la actualidad, las tarjetas de video permiten programar distintas partes de su procesamiento. Esta cualidad permite utilizarlas con distintos fines, implementando algoritmos que serán ejecutados por las unidades de procesamiento gráficas (GPUs) propias de la tarjeta.

Un tema particularmente importante en computación gráfica es el relacionado con el modelado y la visualización de terrenos. Existen importantes campos de aplicación donde es necesario representar escenarios basados en terrenos, como ser: video juegos, cine de animación, simuladores de entrenamiento y sistemas de información geográfica, entre otros. Para visualizar terrenos muy amplios, con imágenes de alta calidad y en tiempo real es necesario enfrentar varios problemas característicos.

Actualmente, para la representación de terrenos reales se dispone de la información de la altura de distintos puntos de la superficie terrestre con una granularidad que llega a un espaciamiento de decenas de metros entre puntos consecutivos. Esta gran cantidad de datos es aún un inconveniente para las capacidades de procesamiento de las tarjetas gráficas actuales si se desean resultados interactivos.

El desafío en la visualización de terrenos consiste en representar fielmente los detalles del terreno, manteniendo la fluidez de generación de imágenes necesaria para brindar animaciones con continuidad visual. Esto implica encontrar un punto de compromiso entre la granularidad (cantidad de puntos y polígonos) con la que se va a modelar un terreno y la velocidad con la que puede procesarse esa cantidad de datos para generar imágenes con la fluidez deseada.

Las soluciones que se han propuesto para resolver el desafío siguen una misma línea de trabajo. Esta consiste en representar el terreno utilizando distintos niveles de detalle en diferentes regiones del mismo, con el fin de disminuir la cantidad de datos (o geometría) a dibujar. Estas soluciones son conocidas como algoritmos o técnicas de nivel de detalle o, en inglés, level-of-detail (LOD). La ***Sección 2*** propone un breve repaso sobre los trabajos más significativos en el tema.

Finalmente, el objetivo de este trabajo es presentar una técnica que resuelva el problema de la visualización de grandes extensiones de terreno. La ***Sección 3*** muestra el diseño de una técnica que provee buenos resultados visuales de calidad y es eficiente en cuanto al procesamiento requerido. Además, el diseño tiene en cuenta las capacidades de las GPUs actuales (aquellas que responden al Shader Model 4.0 [1]), para lograr una implementación eficiente.

La ***Sección 4*** muestra los resultados obtenidos de la implementación de la solución propuesta que esta enmarcada a ser utilizada en aplicaciones de simulación o video

juegos, típicamente un simulador de vuelo. En este tipo de aplicaciones (simuladores) la calidad gráfica lograda en dirección a proveer representaciones fluidas y realistas es un factor fundamental.

Por último, la ***Sección 5*** consiste en las conclusiones del presente trabajo.

2 Trabajos previos

La visualización o renderizado de terrenos presenta aspectos característicos. El terreno es representado utilizando una malla de vértices que modelan su superficie. Una primera aproximación al renderizado de escenas, y particularmente terrenos, es ingresar de manera completa la representación disponible del terreno en el pipeline gráfico, como si fuera cualquier otro objeto. Esta técnica para renderizar terrenos, al no realizar ningún procesamiento particular sobre el mismo, es conocida como fuerza bruta.

Las primeras tarjetas gráficas especializadas o GPUs ofrecían la capacidad de renderizar miles de vértices manteniendo un rendimiento aceptable para aplicaciones interactivas, pero el modelado detallado de terrenos excedía esa capacidad. Utilizar una técnica de fuerza bruta limita el tamaño y los detalles de los modelos de terrenos soportados.

El problema de brindar visualizaciones ricas en detalle de mayores extensiones de terreno seguía vigente. Dado las capacidades de procesamiento del hardware y el realismo idealmente deseado para las representaciones, las soluciones deben buscar el punto de compromiso que conjuguen una performance y una calidad visual adecuada.

Las soluciones que se han propuesto para resolver el desafío siguen una misma línea de trabajo. Esta consiste en representar el terreno utilizando distintos niveles de detalle en diferentes regiones del mismo, con el fin de disminuir la cantidad de datos (o geometría) a dibujar. Estas soluciones son conocidas como algoritmos o técnicas de nivel de detalle o, en inglés, level-of-detail (LOD). Las características comunes a estas soluciones son presentadas en el ***Anexo I***.

En los últimos 15 años, se han publicado varios trabajos que proponen distintas soluciones. Durante la década del ’90, y hacia finales de la misma, se han publicado trabajos significativos en el tema que sentaron las bases de los trabajos posteriores y actuales. En este período, las soluciones se caracterizan por tener como objetivo común minimizar la cantidad de vértices a utilizar en cada cuadro, y a la vez obtener una alta calidad en las imágenes producidas. Dentro de las varias publicaciones, se destacan: Lindstrom et al. [2], Duchaineau et al. [3] y Hoppe [4].

En [2], los autores presentan una solución representando el terreno a través de una malla regular y utilizando, como criterio de elección del nivel de detalle a utilizar, un valor umbral para la proyección del error en pantalla, definido por el usuario. La estrategia es bottom-up y consiste en dos pasos: un primer paso de simplificación a nivel de bloques y, luego, una mayor simplificación vértice por vértice. Por último, provee un nivel de detalle sin discontinuidades espaciales y, según sus autores, un mallado cercano al óptimo.

Duchaineau et al. [3] publicaron en 1997, una técnica de nivel de detalle muy popular que fue bautizada como ROAM. El algoritmo que es presentado en el trabajo utiliza un árbol binario de triángulos para la representación del terreno. Utiliza un par de colas con prioridad para manejar operaciones de división o *split* y de combinación o *merge* de triángulos. El algoritmo aprovecha la coherencia entre cuadro y cuadro, de manera tal que sólo deben realizarse operaciones sobre el mallado obtenido en el cuadro anterior. Por último, brinda una visualización con errores proyectados acotados a partir de un pre-procesamiento donde calcula el error de modelado absoluto.

En 1998, Hugues Hoppe presenta un trabajo [4] donde extiende el uso de su VDPM (View-Dependent Progressive Mesh) framework [5] sobre aplicaciones de renderizado de terrenos. El framework consta de una representación de un modelo en mallas de detalle progresivo y dependiente de parámetros de la visualización. Aplicado al renderizado de terrenos, permite representar la superficie de un terreno utilizando una malla arbitraria de triángulos a partir de una estructura jerárquica de transformaciones geométricas de refinamiento progresivo. Esta estructura permite obtener eficientemente aproximaciones de la malla original en distintos niveles de detalle. Además, el trabajo introduce el uso de *geomorphs* para eliminar el artefacto de *popping*, e introduce un cálculo exacto del error de aproximación en las representaciones utilizadas realizado previo al tiempo de ejecución

Luego, con el comienzo de una nueva década, en el 2000, surge una revolución en el hardware gráfico, es decir, en las tarjetas de video. Sus capacidades de procesamiento comienzan a aumentar en forma exponencial y, además, incorporan la posibilidad de ejecutar programas específicos a nivel de vértices y fragmentos de una forma muy eficiente. Dada la evolución experimentada por las GPU, las soluciones debieron modernizarse para aprovechar las capacidades que fueron incorporadas. Si bien la disminución de la cantidad de triángulos sigue siendo necesaria, ya no es útil la minimización de los mismos llevado a cabo por un gran esfuerzo de cómputo de la CPU. Las técnicas debían aprovechar el paralelismo introducido por las unidades de procesamiento disponibles en las GPU programables. Varios trabajos sobre el renderizado de terrenos mencionan el cambio de criterio y la necesidad de adaptar las soluciones al nuevo paradigma de programación de aplicaciones gráficas [6-8].

Por último, en esta Sección, se presentan dos soluciones modernas que encuentran sus bases en las técnicas anteriormente descriptas. Las mismas pertenecen al estado del arte en el área de visualización de terrenos, debido a su eficiencia y relativa simpleza de implementación.

El trabajo de de Boer [8] es, cronológicamente, el primer trabajo que aborda el renderizado de terrenos teniendo en cuenta las capacidades de las nuevas GPUs. El autor propone dividir el terreno en bloques y luego decidir el nivel de detalle con el que será representado cada bloque en cada instante. Contrariamente a las técnicas ROAM [3] y VDPM aplicado a terrenos [4] que deciden el nivel de detalle en base a un análisis realizado a nivel de los triángulos, y de manera similar a la primera parte de la técnica presentada por Lindstrom et al. [2], el algoritmo presentado decide la resolución de distintas regiones del terreno en un análisis más burdo, a nivel de bloques. Como los trabajos precedentes, para la decisión del nivel de detalle, utiliza un valor umbral para el máximo valor admisible del error proyectado. El algoritmo es

bautizado como GeoMipMaps, en analogía a la técnica de *mipmapping* utilizada con texturas que es bien conocida en el área de la computación gráfica. La analogía consiste en la posibilidad de representar cada bloque con distintos niveles de detalle precalculados, cada uno conteniendo la mitad de la resolución del anterior. La elección del nivel de detalle en cada bloque se realiza utilizando la distancia a la cámara (como uno de los parámetros para el cálculo del error proyectado).

Lossaso y Hoppe presentan un trabajo donde minimizan el trabajo de procesamiento en la CPU. Los autores bautizan esta técnica como Geometry Clipmaps [9], ya que también recurren a una analogía a la técnica de *mipmapping* de texturas. La técnica consiste en una representación del terreno mediante un conjunto de grillas anidadas y centradas alrededor del observador. Cada grilla es representada con un nivel de detalle diferente, disminuyendo la resolución en las grillas externas. La información del terreno que es cubierto por las grillas es almacenada en memoria de video. La información de las grillas es actualizada de manera incremental cuando el observador se desplaza. De esta manera, sólo se renueva una parte de los datos, utilizando y aprovechando la coherencia cuadro a cuadro.

El nivel de detalle o resolución de cada una de las grillas del conjunto se mantiene constante. De este modo, el nivel de detalle de las distintas regiones del terreno sólo se basa en la distancia 2D al observador. Por ende, el algoritmo es independiente de las características particulares del terreno a representar; es decir, no se analizan errores de modelado ni sus proyecciones en pantalla. Por este motivo, una característica principal del trabajo es que la independencia del algoritmo respecto a las particularidades de los terrenos, permite un rendimiento constante. Esto es consecuencia de que la carga de procesamiento realizada por la CPU no varía significativamente durante el tiempo de ejecución.

Por último, un año después de la publicación del trabajo que introduce los Geometry Clipmaps, Assirvatham y Hoppe escriben un artículo donde exponen la implementación de la mayoría de las operaciones en la GPU [10].

3 Técnica de Renderizado de Terrenos Propuesta

3.1 Representación del Terreno

La solución presentada está basada en Geometry Clipmaps y pretende agregarle la habilidad de analizar el error introducido por el uso de niveles de detalle. Esto es realizado subdividiendo cada grilla anidada en bloques o *tiles* (en español, baldosas). Luego, se decide si alguno de los bloques necesita ser refinado para garantizar una representación del terreno con error acotado (en inglés, *error-bounded*).

A partir de Geometry Clipmaps, el terreno es representado usando un conjunto de grillas o *patches* (en español, parches) centrados alrededor de la posición del observador. Cada parche representa una región del terreno con distinta resolución o nivel de detalle. El nivel más detallado es el nivel cero (L = 0), donde el espacio entre los puntos de elevación de la superficie se encuentran en la máxima resolución. La superficie cubierta por cada nivel es cuatro veces mayor que el nivel que lo precede, es decir que

el intervalo de muestreo del primero duplica el del segundo. El intervalo de muestreo de un parche es 2^L veces el intervalo de muestreo de la máxima resolución disponible, para L = 0, 1, 2….

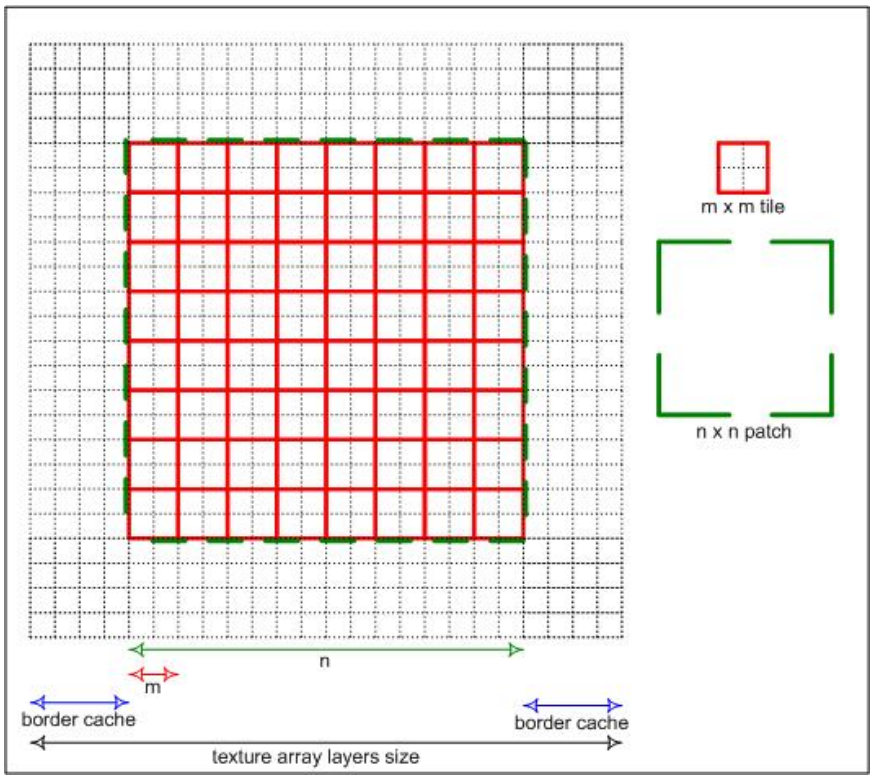

Fig. 1. Tamaños utilizados para los parches, bloques y márgenes. Pueden apreciarse además la relación entre esas estructuras. Se usa para la figura n = 17 y m = 3

En consecuencia, todos los parches o grillas tienen la misma cantidad de puntos o vértices, *n x n*. A diferencia de Geometry Clipmaps, se elige tomar $n = 2^k + 1$ (con *k = 1, 2, 3…*), ya que es necesario para poder dividir cada parche en bloques. La elección de ese valor para *n* requiere manejar nueve casos posibles para las posiciones relativas entre parches de niveles de detalles consecutivos.

En lo que respecta a los recursos a utilizar, los datos de elevación de los vértices de cada parche serán cargados en una textura y muestreados en los programas de vértices. Considerando mejorar el rendimiento, además de los vértices utilizados para renderizar el parche de *n x n*, se cargan en las texturas datos de elevación alrededor de esta región, formando un margen que es utilizado como caché. El tamaño de este margen es una cantidad de vértices igual a una potencia de dos en cada dirección (hacia arriba, abajo, izquierda y derecha). De esta manera, la actualización de los

datos de cada parche recién ocurrirá cuando el caché no contenga los datos necesarios para el renderizado. Cuando esto ocurra, se actualizará una franja completa igual al tamaño del margen, optimizando el uso del ancho de banda de comunicación entre CPU y GPU. Además, esto aumenta la coherencia cuadro a cuadro, ya que evita la actualización frecuente de columnas o filas individuales en las texturas. Por último, la utilización de este margen, y su actualización completa, puede ser aprovechada si los datos de elevación se encuentran comprimidos por esquemas de compresión en bloques.

Como es realizado en Geometry Clipmaps, la actualización de las texturas se realiza de manera incremental utilizando la estrategia de acceso y direccionamiento toroidal.

Hasta este punto, se describió la representación básica del terreno similar a lo utilizado en Geometry Clipmaps. Sin embargo, esta representación no es suficiente, ya que no permite analizar la percepción de los errores de modelado para asegurar una representación con error acotado. Entonces, para agregar esta propiedad a la técnica presentada, se decide dividir cada parche en bloques. Estos bloques permiten analizar las características locales del terreno dentro de cada parche, decidiendo si el nivel de detalle del parche es suficiente o no. El análisis realizado será descripto en la subsección siguiente.

En consecuencia, se utilizan bloques cuadrados de tamaño *m x m*, donde $m = 2^j + 1$ (con *j = 1, 2, 3…*). La solución necesita que cada parche sea divido en, al menos, 8 x 8 bloques. Esto significa que $k - j \geq 3$. La necesidad surge de ubicar la posición relativa del centro de cada parche, que ahora deberá coincidir con la esquina de algún bloque propio. Por otro lado, las texturas que contienen las elevaciones de cada parche, deben almacenar datos de una cantidad entera de bloques, por ende el tamaño del margen utilizado como caché, en cada sentido, debe ser de $2i(m-1)$ vértices de ancho (con *i* = 1, 2, 3…).

Finalmente, en la Fig.1 pueden apreciarse las relaciones entre los tamaños de las distintas grillas utilizadas para los parches, los bloques, los márgenes y las texturas.

3.2 Análisis del Error Proyectado

La técnica presentada propone dividir cada parche original de Geometry Clipmaps en bloques y, entonces, decidir si el nivel de detalle del parche es apropiado para cada bloque. La meta es que la decisión del nivel de detalle a utilizar pueda asegurar visualizaciones del terreno donde los errores introducidos no sean apreciables.

Cuando la superficie de un terreno es representada con una malla de menor resolución que la disponible, se introducen algunos errores de aproximación o de modelado. El error de modelado se define como la distancia vertical que surge de la diferencia de altura de un vértice presente en la malla de máxima resolución respecto de la posición virtual que toma cuando es removido en un nivel de detalle de menor resolución. Este error generalmente es medido directamente en coordenadas del mundo, y por esto es conocido en la bibliografía como *world-space error*.

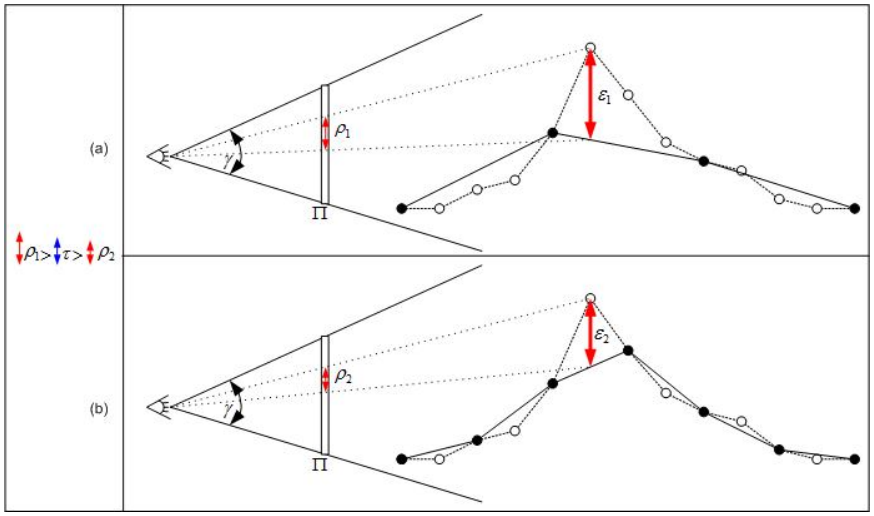

Fig. 2. Los errores de modelado ε_1 y ε_2, originados por representaciones utilizando dos niveles de detalle diferentes, son proyectados en el plano de visualización Π. (a) Nivel 2 (L=2): la proyección ρ_1 del máximo error de modelado excede el valor umbral τ. (b) Nivel 1 (L=1): la proyección ρ_2 del máximo error de modelado es menor que el valor umbral τ.

El error de modelado puede ser medido en términos relativos o absolutos. En algunos trabajos, como por ejemplo [2] y [8], el error es medido en términos relativos entre niveles de detalles sucesivos. Esta forma de medición no permite asegurar representaciones con errores acotados. Si la técnica de nivel de detalle desea brindar una solución con error acotado, la medición de errores relativos debe ser correctamente saturada [11]. La saturación de errores asegura que, en un mismo bloque, el máximo error de modelado presente en una representación de menor detalle nunca es menor que al usar un nivel de mayor detalle. Sin embargo, es más preciso calcular el error de modelado de manera absoluta respecto de la máxima resolución disponible, como es realizado en ROAM [3].

Hugues Hoppe presenta una manera de calcular el error de modelado de manera absoluta y exacta [4]. Propone el cálculo del error no sólo en los vértices de la máxima resolución que hayan sido omitidos, sino también en las intersecciones de los lados de los triángulos de menor detalle con las aristas de los triángulos presentes en la máxima resolución. La técnica presentada pretende este cálculo exacto del error. Sin embargo, al utilizar bloques que son simplificados en resoluciones sucesivas de potencias de dos, los lados de los triángulos de menor detalle coinciden con las aristas presentes en la máxima resolución. Por lo tanto, solo basta con medir el error de modelado en los vértices presentes en la máxima resolución que fueron omitidos.

Respecto al cálculo del error de modelado, la solución requiere que sea realizado anteriormente al tiempo de ejecución. La técnica desarrollada lo realiza como un preprocesamiento realizado off-line, y guarda los resultados en un pequeño archivo en

disco. Luego, en tiempo de carga el archivo es cargado del disco y dispuesto en memoria de CPU para evitar las latencias de lectura de disco en tiempo de ejecución. El cálculo es realizado a partir de dividir el terreno en bloques de m x m, realizando sucesivas pasadas con los bloques en sucesivas resoluciones. Se calcula el máximo error de modelado absoluto y exacto de cada bloque y se almacena en un archivo en disco. Luego, en tiempo de ejecución, a medida que el observador se desplaza y los parches son actualizados incrementalmente, se consultan los errores precalculados (ya dispuestos en la memoria principal) de los bloques necesarios.

Por otro lado, el máximo error de modelado de cada bloque, en un nivel de detalle particular, es una entrada necesaria para analizar si los errores de aproximación son perceptibles por el usuario. Entonces, en tiempo de ejecución, tras calcular la menor distancia del observador al bloque y utilizando los parámetros de la visualización, se calcula el tamaño del error proyectado en píxeles (ver cálculo en ***Anexo I***).

Finalmente, la aplicación es configurada definiendo la máxima cantidad de pixeles tolerables para el error proyectado (como se encuentra en coordenadas de la pantalla, en la bibliografía es llamado *screen-space error*). Este valor constituye un valor umbral con el cual se puede comparar. Si la proyección del máximo error de modelado de un bloque en pantalla excede el valor umbral, entonces esa región de terreno necesita ser representada con una resolución mayor (Fig. 2).

3.3 Algoritmo de Renderizado

Esta sección describe la estrategia de renderizado abordada por la técnica desarrollada. Se explica el algoritmo utilizado para generar cada cuadro en tiempo de ejecución en un alto nivel.

En primer lugar, dada la posición del observador, se calcula la posición central de cada parche y se actualiza la información de elevación en cada textura si es necesario. La posición central de cada parche debe coincidir con la posición de un vértice presente en una representación del terreno utilizando el mismo nivel de detalle del parche. Además deben tenerse en cuenta las posiciones relativas entre los parches como fue explicado en la primera sub-sección de esta sección. Por ende, se comienza por calcular la posición central desde el parche de menor nivel de detalle utilizado. Luego, si existe desplazamiento en la posición central en cada parche respecto del cuadro anterior, y las elevaciones necesarias no se encuentran en la textura cargada en la GPU, se procede a actualizar las texturas necesarias.

En segundo lugar, se toman los bloques de cada parche (excluyendo aquellos bloques presentes en los márgenes) para ser testeados contra el volumen de visualización (técnica conocida como *frustum-culling*). Este testeo es al menos necesario, ya que reduce en un cuarto la carga del pipeline gráfico para el renderizado de geometría (calculado sobre un campo de visualización horizontal de 90 grados). El testeo se realiza utilizando una caja (*axis-aligned bounding box* (AABB)) para definir el volumen ocupado por cada bloque.

En tercer lugar, el análisis del error proyectado se realiza sobre aquellos bloques que no fueron descartados en el paso anterior. Se toma el máximo error de aproximación de cada bloque y se lo proyecta en pantalla para ser comparado con el valor um-

bral definido. Si el error proyectado excede el valor umbral, ese bloque necesita ser representado con una mayor resolución. El bloque debe ser representado con un nivel de detalle cuyo máximo error de aproximación, al proyectarse, no exceda el valor umbral. Este bloque, que será llamado ***bloque refinado***, es renderizado utilizando una resolución dos, cuatro, ocho, etc. veces mayor según sea necesario (experimentalmente, raramente es necesario más de dos niveles de refinamiento). En consecuencia, los bloques refinados tendrán $(r+1)*(m-1)+1$ vértices de lado, donde r es el nivel de refinamiento ($r = 1, 2, 3…$).

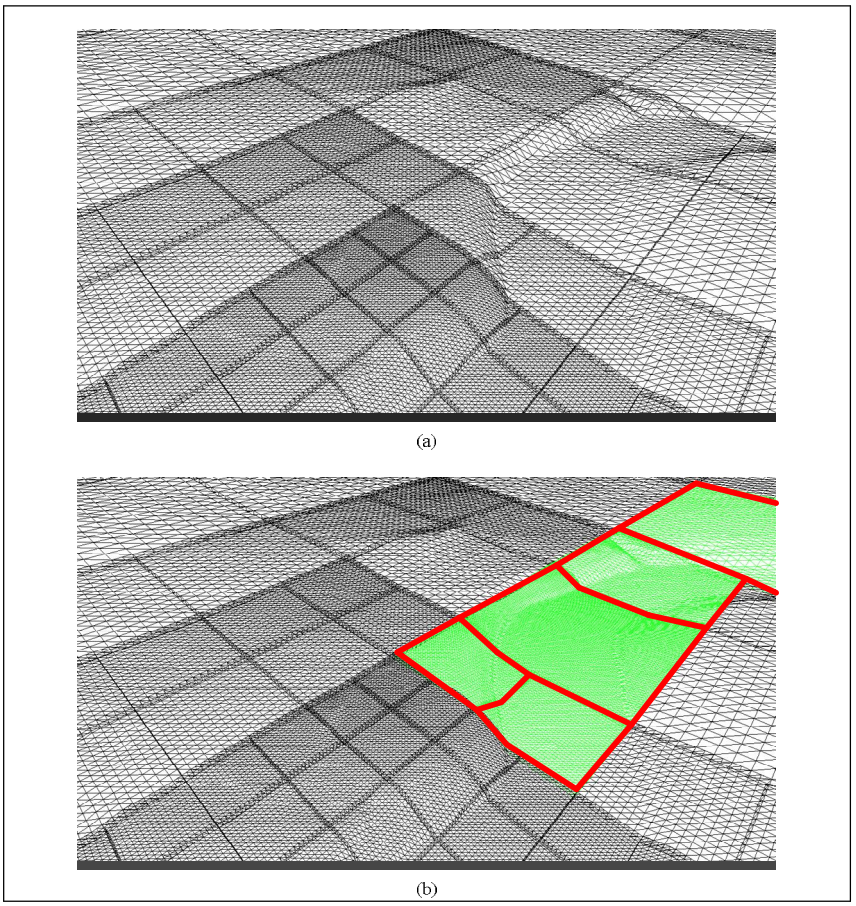

Fig. 3. Dos cuadros distintos renderizados por la técnica propuesta (usando n = 129, m = 17). (a) Parches anidados alrededor del usuario y diferentes posiciones relativas entre niveles sucesivos. (b) Los parches que necesitaron refinamiento son marcados en rojo y dibujados en verde.

Este último paso también incluye cargar en memoria de GPU (utilizando texturas) las elevaciones necesarias para el renderizado de los bloques refinados. La técnica

propone utilizar una cantidad fija de texturas para cada nivel de refinamiento, y administrar su utilización utilizando la menos usada recientemente. Esta política de administración de memoria es llamada ***least-recently used*** o LRU.

Por último, el algoritmo dibuja los bloques utilizando la técnica de *instancing* [1]. Por ende, dibuja primero los bloques que no necesitan refinamiento, y luego los bloques refinados agrupados por nivel de refinamiento. La Fig. 3 muestra dos cuadros renderizados por la técnica propuesta: en (a) pueden verse los distintos parches anidados y divididos en bloques; mientras que en (b) se muestran los bloques refinados marcados en rojo.

3.4 Resolución de Artefactos

Al representar los terrenos con nivel de detalle pueden provocarse artefactos que degradan la continuidad espacial y temporal en las representaciones de los terrenos.

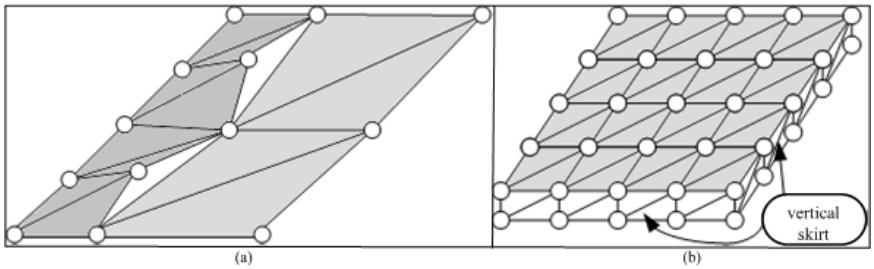

Fig. 4. (a) Las grietas que pueden aparecer entre bloques de distinto nivel de detalle. (b) *Polleras* o *skirts* usadas alrededor de los bloques para ocultar la aparición de grietas.

Por un lado, la discontinuidad espacial se manifiesta en grietas en la superficie del terreno y pueden ocurrir en los bordes de dos regiones representadas con niveles de detalle diferente. Este problema surge en las representaciones de nivel de detalle que no presentan un mallado continuo, como la descripta en esta tesis, al originar vértices T. Soluciones que proponen niveles de detalle continuos como [2] y [3] no presentan este inconveniente.

Tanto las técnicas de nivel de detalle basadas en el uso de bloques, como las que utilizan grillas anidadas, tienen que resolver este problema. Las grietas pueden aparecer en bloques adyacentes de distinta resolución, ya que no comparten la misma cantidad de vértices en sus bordes (Fig. 4(a)). Varios trabajos en la bibliografía del tema, al presentar variaciones en los métodos basados en el uso de bloques, han propuesto distintas alternativas para evitar la aparición de grietas entre bloques vecinos. Algunos trabajos [7-8, 12] resuelven el problema modificando la conexión de los vértices de uno de los bloques (generalmente en el bloque de mayor resolución), evitando la presencia de vértices T. En este trabajo se decide no utilizar estas estrategias ya que requieren conocer para cada bloque el nivel de detalle de sus vecinos, agregando complejidad al algoritmo y la necesidad de manejar varios casos. Por otro lado, Ulrich [6] describe en su trabajo algunas técnicas para resolver el problema agregando geometría

para cubrir las posibles grietas. Las opciones que enumera son agregar alrededor de cada bloque vértices para formar: (i) “pestañas” hacia abajo con un cierto ángulo y longitud, que da a llamar *flanges*; (ii) “cintas verticales” de una determinada altura, que llama *vertical ribbons*; y (iii) “polleras” verticales, también de una determinada altura, en inglés, *skirts*. Mayor detalle de estas opciones pueden verse en el trabajo citado. Geometry Clipmaps cubre el perímetro de cada parche utilizando triángulos de área nula. Sin embargo, esta técnica sólo es válida cuando los bloques difieren en solo un nivel de detalle. Finalmente, en la técnica presentada se opta por utilizar “polleras” verticales alrededor de cada bloque. Estas polleras coinciden en su borde superior con el borde del bloque y tienen una extensión vertical definida por parámetro (Fig. 4(b)). Además, como las grietas introducidas se mantienen por debajo del valor umbral establecido, esta estrategia elegida es eficiente.

Por otro lado, las técnicas de nivel de detalle deben evitar que ocurra el artefacto de *popping*. Para prevenir este efecto, las soluciones suelen interpolar gradualmente la geometría de una región cuando ocurre un cambio en el nivel de detalle. Hoppe en [4] bautizó esta estrategia como *geomorphing*. Como la técnica presentada asegura un error acotado para la representación, los cambios de geometría se mantendrán por debajo del valor umbral definido y, por definición, imperceptibles. Dada esta razón, la técnica no aplica ninguna estrategia específica para evitar los saltos o *pops*.

4 Resultados Obtenidos

La implementación de la técnica descripta fue realizada para hacer uso de las funcionalidades provistas por las GPUs que responden al Shader Model 4.0 [1], utilizando la biblioteca gráfica Direct3D 10 [13].

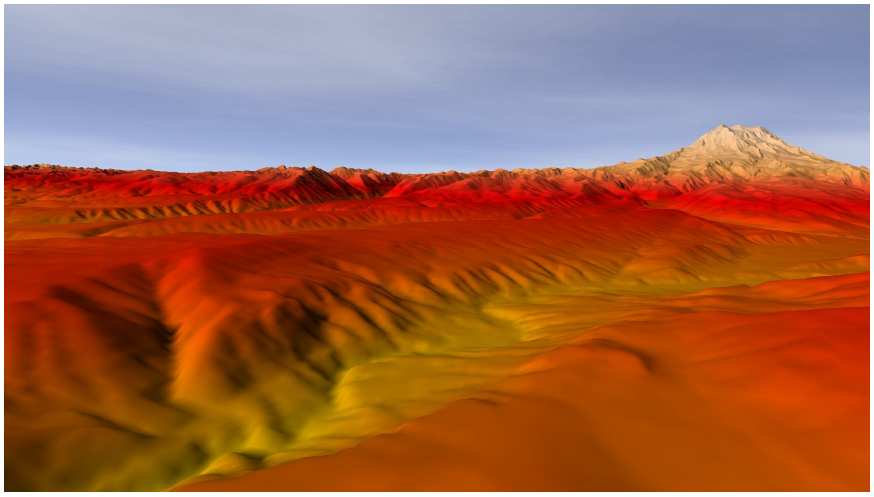

Fig. 5. Resultado del renderizado del terreno en un vuelo sobre el terreno “Puget Sound”.

La mayoría de los trabajos en el tema utilizan un terreno de referencia para mostrar y comparar sus resultados. Este conjunto de datos bien conocido es llamado "*Puget Sound area*". El mismo esta compuesto por una grilla regular de 16.385 x 16.385 puntos con un intervalo de muestreo de 10 metros, cubriendo por lo tanto una superficie de 163,85 Km. x 163,85 Km. El tamaño de la superficie es adecuado para aplicaciones como simuladores de vuelo. Los valores de altura de cada punto son codificados utilizando 16 bits (65536 valores discretos posibles) con una resolución vertical de 0,1 metro. Este conjunto de datos se dispone en un archivo de imagen PNG de un canal de 16 bits.

La ejecución de la implementación fue realizada en una PC, sobre sistema operativo Windows 7, con 2.8 GHz Intel® Core™2 Quad, 4 GB de memoria principal y una tarjeta de video Nvidia GeForce GTX 280 con 2GB de memoria de video.

El framework fue configurado para correr en una ventana *fullscreen* con una resolución de 1920 x 1080 píxeles. El campo de vista horizontal del observador utilizado es de 90 grados. La elección del valor umbral para los errores proyectados fue de 5 píxeles (0,5% de la resolución vertical). De esta manera, se obtuvo una perfomance promedio de 120 cuadros por segundo, lo que implica un promedio de 8 milisegundos para la generación de cada cuadro. Por otro lado, cuando el observador vuela cercano a zonas de detalles de alta frecuencia, el rendimiento promedio cae a unos 50 cuadros por segundo. Debido a la carga de datos necesaria para representar bloques refinados, algunos cuadros pueden llegar a requerir unos 40 milisegundos para su generación. La Fig. 5 muestra una captura de pantalla mostrando la visualización obtenida sobre el terreno mencionado.

La Tabla 1 y Fig. 6 resumen los resultados obtenidos tras ejecutar cinco corridas en las condiciones especificadas. Los cinco experimentos consistieron en un vuelo sobre el terreno a una velocidad de 340 metros por segundo, típicamente desde el centro del terreno hacia la zona de la montaña bien detallada. Se midió el tiempo insumido en la generación de cada cuadro durante un vuelo de entre dos y tres minutos. Como puede verse en la Fig. 6 el 70% de los tiempos medidos se encuentran en el intervalo entre 5 y 15 milisegundos (44,35% en el intervalo 5–10 ms., y 25.44% en el intervalo 10–15 ms.). Por último, el tiempo máximo registrado fue de 49,5 ms.

Tabla 1. Resultados de las cinco corridas ejecutadas.

	Run 1	Run 2	Run 3	Run 4	Run 5
Tiempo de Vuelo	132 seg.	189 seg.	180 seg.	166 seg.	140 seg.
Cuadros Totales	10960	15545	13564	13118	12575
Max. Frame Time	49.5 ms.	44.1 ms.	40.7 ms.	41.5 ms.	36.0 ms.
Frame Time in 5-10 ms	49.91%	43.94%	37.14%	41.31%	49.46%
Frame Time in 10-15 ms	24.80%	14.91%	31.21%	27.19%	29.11%

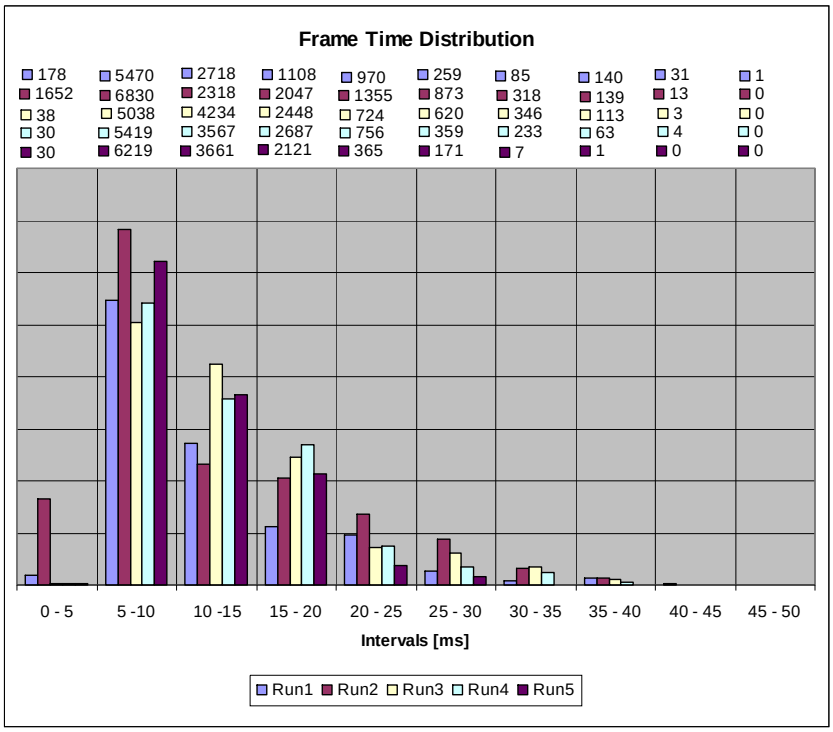

Fig. 6. Distribución de las mediciones de los tiempos de generación de cada cuadro en las cinco corridas ejecutadas.

5 Conclusiones

La solución desarrollada brinda una técnica de renderizado de terrenos apta para aplicaciones interactivas o tiempo real, como ser videojuegos o simuladores de vuelo. Con mayor precisión, la técnica apunta a ser utilizada en simuladores de vuelo, donde los requerimientos son más estrictos. La visualización producida debe representar fielmente terrenos del mundo real logrando una inmersión total del usuario dentro del simulador. Por ende, la técnica propuesta no muestra distorsiones ni artefactos, brindando una representación visualmente exacta del terreno.

La principal contribución de este trabajo es la presentación de una técnica que combina aspectos de las técnicas que pertenecen al estado del arte para brindar una solución sencilla y con un buen rendimiento. La técnica presentada está basada en Geometry Clipmaps, que expone ideas y algoritmos que optimizan el procesamiento requerido obteniendo una alta performance. Más aún, la técnica presentada resuelve la limitación que presenta Geometry Clipmaps para representar superficies con detalles de alta frecuencia sin introducir cambios en la superficie que sean notorios al usuario. Para esto, se opta por incorporar el análisis de la proyección del error de aproxima-

ción propio de las representaciones que utilizan nivel de detalle. La estrategia consiste en mantener acotado el error percibido por debajo de un valor que pueda considerarse despreciable. Para lograrlo, el análisis del error es realizado tras dividir las regiones del terreno en bloques, lo que es propuesto en las técnicas basadas en bloques, como Geometrical MipMaps.

Finalmente, la técnica es implementada utilizando software y hardware actuales, aprovechando las novedades útiles para optimizar la performance. Los resultados de los experimentos muestran un buen rendimiento de la técnica presentada. El poco tiempo de procesamiento de CPU y GPU que consume, permite que la técnica sea incorporada en un motor gráfico a ser utilizado para el desarrollo de aplicaciones interactivas.

Referencias

1. Patidar, S., Bhattacharjee, S., Singh, J.M., and Narayanan, P.: *"Exploiting the shader model 4.0 architecture"*. Technical Report, Center for Visual Information Technology, IIIT Hyderabad, 2006.
2. Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L.F., Faust, N., and Turner, G.A.: "Real-time, continuous level of detail rendering of height fields". Proceedings ACM SIGGRAPH, ACM SIGGRAPH 1996, pp. 109–118, 1996.
3. Duchaineau, M., Wolinsky, M., Sigeti, D.E., Miller, M.C., Aldrich, C., and Mineev-Weinstein, M.B.: "ROAMing terrain: Real- time optimally adapting meshes". Proceedings IEEE Visualization, pp. 81–88, 1997.
4. Hoppe, H.: "Smooth view-dependent level-of-detail control and its application to terrain rendering". Proceedings IEEE Visualization, Computer Society Press, pp. 35–42, 1998.
5. Hoppe, H.: "View-dependent refinement of progressive meshes". Computer Graphics, SIGGRAPH '97 proceedings, pp. 189-198, 1997.
6. Ulrich, T.: "Rendering Massive Terrains Using Chunked Level of Detail Control", SIGGRAPH '02 Course Notes, 2002.
7. Wagner, D.: "Terrain Geomorphing in the Vertex Shader", ShaderX2, Shader Programming Tips and Tricks with DirectX 9, Wordware Publishing, 2004.
8. De Boer, W.H.: "Fast Terrain Rendering Using Geometrical Mipmapping". Unpublished Paper, Available at http://www.flipcode.com/ articles/article_geomipmaps.pdf, 2000.
9. Losasso, F., and Hoppe, H.: "Geometry Clipmaps: Terrain Rendering Using Nested Regular Grids". ACM SIGGRAPH 2004 Papers, pp. 769–776, 2004.
10. Asirvatham, A., and Hoppe, H.: "Terrain Rendering Using GPU-based Geometry Clipmaps", GPU Gems 2, pp. 27–46, 2005.
11. Pajarola, R., and Gobbetti, E.: "Survey of Semi-regular Multiresolution Models for Interactive Terrain Rendering". The Visual Computer nº23, 583-605, 2007.
12. Snook, G: "Simplified Terrain using Interlocking Tiles". Game Programming Gems 2, Charles River Media, Inc., 377-383, 2001.
13. Blythe, D.: "The Direct3D 10 system". *ACM SIGGRAPH 2006, Papers on SIGGRAPH '06*, ACM Press, 2006.
14. Luebke, D., Reddy, M., Cohen, J.D., Varshney, A., Watson, B., and Huebner, R.: "Level of Detail for 3D Graphics". Morgan Kaufmann Publishers, 2003.

6 *Anexo I:* Terrenos con Nivel de Detalle

La incorporación de la visualización de mayores extensiones de terreno de mayor calidad en distintas aplicaciones, es posible a partir del ajuste de la resolución en distintas regiones del modelo del terreno. La estrategia consiste en representar zonas con mayor o menor resolución a partir de la percepción posible de sus detalles. Tomando las propiedades de la visión en perspectiva, zonas cercanas al observador requerirán mayor resolución que aquellas zonas más alejadas. Asimismo, regiones con accidentes geográficos más pronunciados serán representados con mayor resolución respecto de regiones de la superficie del terreno llanas o de leves ondulaciones.

La idea o estrategia presentada, entonces, da lugar a las técnicas de nivel de detalle (en inglés *level-of-detail*, abrevieándose *LOD*). Para el caso de la visualización de terrenos, se desarrollaron técnicas para manejar el nivel de detalle o resolución con el cual se representarán las distintas regiones del modelo de la superficie. Esta elección de los niveles de detalle a utilizar debe realizarse de manera dinámica en tiempo de ejecución, es decir, cuadro por cuadro. La resolución elegida varía a partir de, por ejemplo, la posición del observador o cámara, entre otros criterios (Fig. 7).

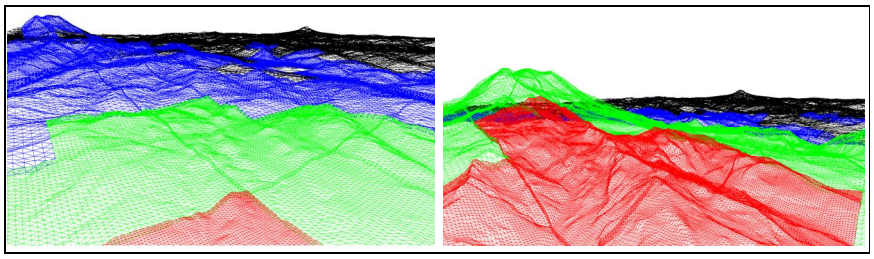

Fig. 7. Un mismo terreno representado con mallas con diferente nivel de detalle. El nivel de detalle de distintas regiones varía según la posición del observador.

A continuación, en este anexo, se explicarán los distintos aspectos involucrados en las técnicas de nivel de detalle. Los aspectos referidos son los criterios para elegir el nivel de detalle adecuado, la estrategia de los algoritmos y los inconvenientes a resolver cuando se utilizan estas técnicas.

6.1 Criterios para la Elección del Nivel de Detalle

Dentro de la vasta bibliografía en el tema, los trabajos publicados muestran que la elección del nivel de detalle a utilizar puede depender de distintos parámetros. El criterio utilizado puede involucrar cálculos sencillos y de baja precisión, como también algoritmos más precisos y complejos. Los criterios más precisos utilizan como entrada mayor cantidad de parámetros. Asimismo, dentro de la bibliografía también puede encontrarse cómo los distintos criterios pueden combinarse. A continuación se desarrollarán tres de los criterios más utilizados.

Distancia al observador.

La manera más sencilla de elegir el nivel de detalle es utilizando la distancia al observador. Por ejemplo, si un terreno es dividido en regiones cuadradas de *n x n* vértices, la misma podrá ser representada con menor cantidad de vértices: $n/2$ x $n/2$, $n/4$ x $n/4$, etc. (Fig. 8). A partir de la distancia del observador al centro de cada región, usualmente proyectada al plano XY, se elige el nivel de detalle con el cual debe ser representada. En otras palabras, a partir de la distancia al observador se elige la cantidad de puntos con la que la superficie de cada región del terreno es modelada.

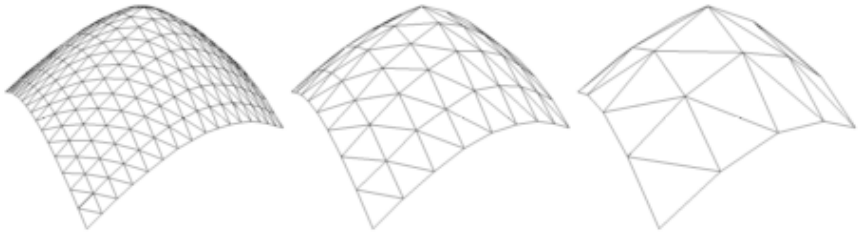

Fig. 8. Una misma región de terreno representada en distintas resoluciones.

Error de modelado.

Un criterio más preciso considera la rugosidad del terreno. El criterio anterior, utilizando sólo la distancia al observador, concibe representaciones de los terrenos ignorando las características particulares del mismo. Ese criterio no considera si cada región representa una superficie llana, con pocos detalles o de baja frecuencia, o si, por el contrario, modela una porción de terreno con accidentes geográficos más pronunciados, con mayores detalles o detalles de alta frecuencia. Por lo tanto, este criterio más preciso tiene en cuenta, no sólo la distancia al observador, sino también, las características de la superficie de cada región.

Si un mismo terreno, o región del mismo, es modelado con distinta cantidad de puntos, existe una diferencia entre las superficies resultantes. Si una de las superficies es tomada de referencia, se dice que las diferencias encontradas por una representación alternativa constituyen un error geométrico o de modelado. Este error geométrico o de modelado puede encontrarse en coordenadas del objeto, o tras su correspondiente transformación del mundo, en coordenadas del mundo.

Particularmente, analizamos el caso donde una porción de terreno es representada con menor resolución a partir de un sub-muestreo de los vértices utilizados en una representación de mayor resolución o mayor densidad de vértices. La altura de un punto de la superficie coincidente con un vértice omitido en una representación de menor detalle, cambiará y será resultado de una combinación o interpolación de las alturas de los vértices presentes. La diferencia de alturas (valor de la coordenada z) para el mismo punto en los modelados de distinto detalle constituye el error de modelado para ese punto (Fig. 9).

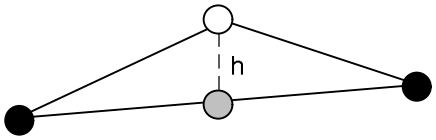

Fig. 9. Error geométrico o de modelado. Vista desde el costado del error en la altura producido al omitir el vértice blanco en un bloque de menor detalle, su posición imaginaria quedará en el segmento que une los vértices negros. El cambio en la altura y error de modelado es h.

El error geométrico o de modelado puede calcularse de manera relativa, entre representaciones de niveles de detalle sucesivos, o de manera absoluta, tomando siempre como referencia el modelado de mayor resolución que contiene todos los vértices disponibles.

Por ejemplo, un criterio que tiene en cuenta el error de modelado presentado junto con la distancia al observador, posee valores máximos para los errores de modelado aceptables según distintos rangos de distancias. Los errores máximos admisibles aumentan con la distancia. Finalmente, el nivel de detalle seleccionado para cada región involucrará un error de representación por debajo del valor umbral establecido para la distancia al observador en la que se encuentre.

Error percibido o proyectado.

Hasta el momento, los criterios enumerados no tienen en cuenta el mecanismo de la visión humana. La manera de obtener modelos de terrenos simples y de la mayor calidad visual posible es utilizar un criterio que aproveche las características de la transformación de proyección perspectiva.

Los triángulos que representan la superficie del terreno en un modelo tridimensional, tras ser procesadas en el pipeline gráfico, son proyectados en la pantalla. El proceso de rasterizado definirá el conjunto de píxeles para dibujar en la pantalla cada triángulo. De esta manera se puede analizar si los detalles de un modelo son perceptibles por el usuario.

El criterio, entonces, consiste en evaluar el tamaño en píxeles de la proyección del error geométrico o de modelado introducido anteriormente. El modelado de una zona del terreno con un determinado nivel de detalle u otro, tiene por consecuencia errores geométricos. Por lo tanto, el criterio de elección pasa por evaluar la proyección del error de modelado, medida en unidades del espacio de pantalla, y decidir si el mismo es aceptable o no.

Por defecto, si el tamaño del error proyectado en pantalla es menor a un píxel, el mismo no será apreciable y el nivel de detalle en cuestión es aceptable. Sin embargo, es común que las aplicaciones o soluciones de visualizaciones y renderizado de terrenos permitan elegir un valor máximo de píxeles para que un error se considere despreciable. Al establecer un valor máximo o umbral, si la proyección del error de modelado no excede dicho valor, entonces el nivel de detalle que se quiere utilizar es aceptable.

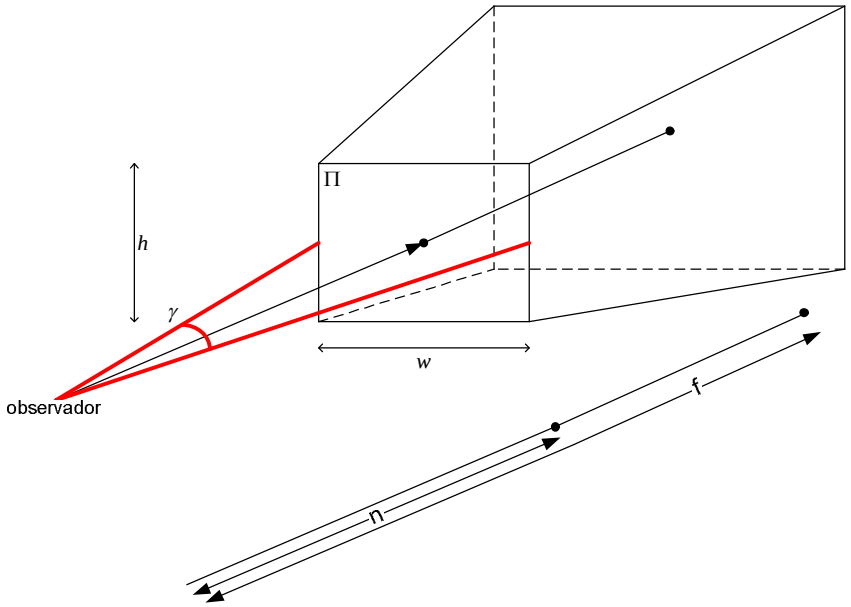

Fig. 10. Volumen de Visualización

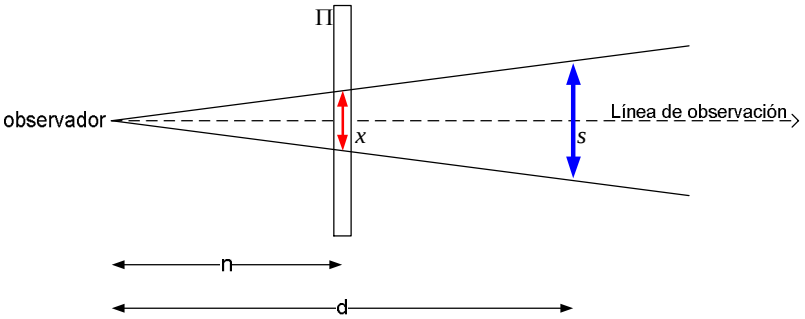

Fig. 11. Semejanza de triángulos para el cálculo de la proyección de un segmento sobre el plano de visualización.

Este criterio es el más preciso de los vistos, e involucra a los anteriores. Utiliza el error de modelado y necesita la distancia de dicho error a la posición del observador, para realizar el cálculo de la proyección y mapeo a la pantalla.

Finalmente, el cálculo de la proyección es explicado a continuación. El volumen de visualización o *view-frustum*, utilizando una proyección en perspectiva, es definido a partir de: (i) el campo de visión horizontal γ (en grados), que refiere a la amplitud visual del observador o del lente de la cámara; (ii) el tamaño de la ventana de visuali-

zación, ancho w_{screen} y alto h_{screen} (en píxeles); (iii) la distancia n al centro del plano de proyección Π (en unidades del mundo); y (iv) la mayor distancia de lo visualizado o distancia *f* al plano trasero del volumen de visualización (en unidades del mundo) (Fig. 10). Entonces, a partir de los parámetros de visualización, se quiere calcular la proyección x en el plano Π de un segmento de longitud s que se encuentra en el mundo a una distancia *d* del observador. De manera conservadora, se asume que el segmento a proyectar tiene una orientación ortogonal a la dirección de observación. Por semejanza de triángulos (Fig. 11), se obtiene (1). Por ende, el resultado obtenido de x se encuentra en unidades del mundo.

$$\frac{s}{d} = \frac{x_{world}}{n} \Rightarrow x_{world} = n\frac{s}{d} \tag{1}$$

Si se quiere obtener la cantidad de píxeles que representarán la proyección x del segmento s, debemos calcular el tamaño de la ventana de visualización en coordenadas del mundo, ancho w_{world} y alto h_{world} (2).

$$\left.\begin{array}{l} tg(\gamma/2) = \dfrac{w_{world}/2}{n} \\ aspect = \dfrac{w_{screen}}{h_{scren}} \end{array}\right\} \Rightarrow \begin{cases} w_{world} = 2n\tan(\gamma/2) \\ h_{world} = \dfrac{w_{world}}{aspect} \end{cases} \tag{2}$$

Asumiendo que la proyección x se encuentra en el centro del plano Π, se calcula el tamaño relativo de dicha proyección en la ventana de visualización y luego se transforma en pixeles, obteniendo x_{screen} en píxeles. Se calculan, entonces, dos casos, $x_{screenh}$ asumiendo que el segmento se encuentra horizontal (3), y $x_{screenv}$ asumiendo que el segmento se encuentra vertical (4).

$$x_{screenh} = \frac{x_{world}}{w_{world}} w_{screen} \Rightarrow x_{screenh} = \frac{s}{2d\tan(\gamma/2)} w_{screen} \tag{3}$$

$$x_{screenv} = \frac{x_{world}}{h_{world}} h_{screen} \Rightarrow x_{screenv} = \frac{s.aspect}{2d\tan(\gamma/2)} h_{screen} \tag{4}$$

Por último, el error de modelado es proyectado en la pantalla, de manera conservadora, utilizando el procedimiento y las ecuaciones explicadas. El método conservador asume que el error de modelado se representa con un segmento ortogonal a la dirección de observación y en posición vertical en el centro de la ventana de visualización.

6.2 Simplificación vs. Refinamiento

La representación de un terreno con nivel de detalle implica reducir la complejidad del modelo sin perder calidad visual. El objetivo es obtener una representación más simple manteniendo la misma apreciación del terreno. En el procedimiento de obtener un mallado más simple, la motivación puede estar dada por: (i) mantener cantidad de primitivas o vértices a utilizar por debajo de un valor establecido; (ii) mantener el error de representación menor a un valor umbral que se considere aceptable; y por último, (iii) una combinación de las anteriores. Independientemente de la motivación utilizada, las soluciones pueden utilizar estrategias diferentes. Las técnicas de nivel de detalle pueden estar concebidas a partir de dos estrategias diametralmente opuestas.

Una primera estrategia consiste en comenzar por un modelo simplificado del terreno, un modelo burdo o de baja resolución, y luego aplicar operaciones de refinamiento hasta alcanzar una representación del terreno con un nivel de detalle aceptable, según la motivación y el criterio de elección utilizado. Esta estrategia, que parte de un modelo de poca resolución llegar a una representación de mayor detalle, es llamada de *refinamiento* o, en inglés, *top-down.*

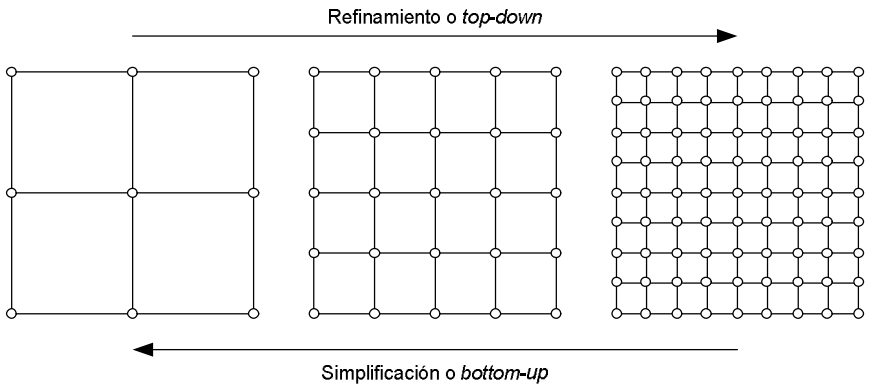

Fig. 12. Simplificación vs. Refinamiento. Una misma región de terreno es representada con mayor o menor detalle.

Por el contrario, otra estrategia utilizada es aquella que comienza por la representación del terreno en su máxima resolución y, tras ejecutar varias operaciones de simplificación, obtiene un mallado del terreno más simple, con el nivel de detalle suficiente que satisface la motivación y el criterio de elección usado. Esta estrategia es llamada de simplificación o, en inglés, *bottom-up* (Fig. 12).

Las nociones de arriba y abajo (*top and bottom)* surgen de imaginar un terreno y sus regiones en una estructura de árbol, donde la raíz de la estructura contiene un modelo simplificado del terreno. El terreno modelado en la raíz es dividido en *n* regiones representadas en un mayor detalle en *n* nodos hijos de la raíz. Sucesivamente, se subdivide cada región en regiones más pequeñas y de mayor detalle, hasta alcanzar una profundidad donde el conjunto de los nodos hoja del árbol constituyen el modelo

del terreno en su máxima resolución (Fig. 13). Si se imagina el árbol con el nodo raíz en la parte superior y las ramificaciones hacia abajo, las estrategias toman su nombre a partir de la manera en que los nodos del árbol son recorridos. En la simplificación, se parte desde abajo, desde los nodos hojas y se procede simplificando y subiendo en la jerarquía de nodos (*bottom-up*). En cambio, en el proceso de refinamiento, se parte desde la raíz, desde arriba, y se refina el mallado bajando por los nodos del árbol (*top-down*).

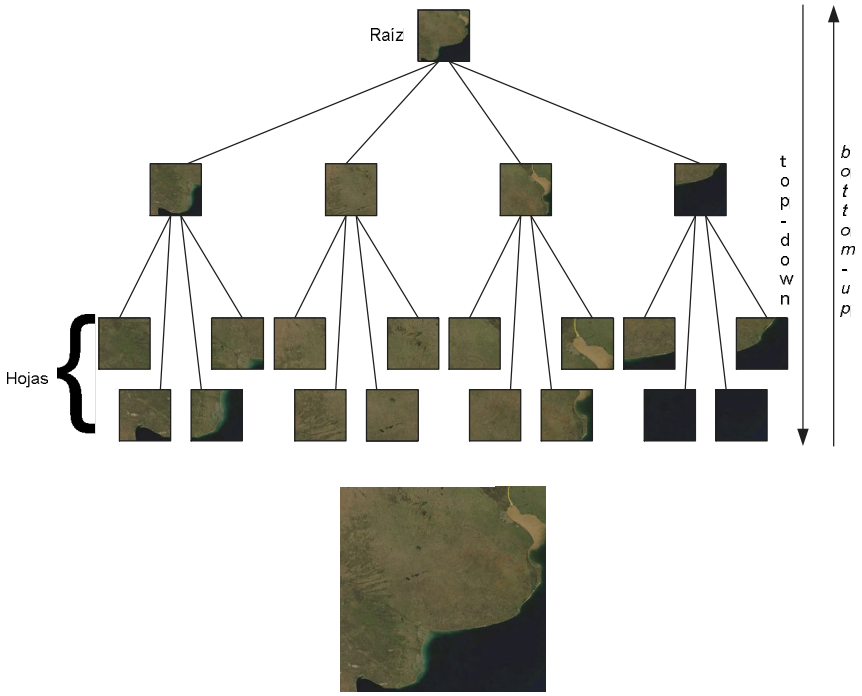

Fig. 13. Estructura de árbol utilizada para manejar el nivel de detalle en la representación de terrenos. La raíz contiene el modelo completo del terreno en una baja resolución. Los nodos hoja contienen regiones del terreno con la máxima resolución disponible. Utilizando todos los nodos hojas se obtiene el terreno con la máxima resolución.

Los algoritmos o técnicas de nivel de detalle, aplican estas estrategias cuadro por cuadro para decidir la resolución con la cual será representado el terreno, según los criterios utilizados. Generalmente, la estrategia de refinamiento produce resultados más precisos, logrando la mínima cantidad de vértices para la calidad deseada. Sin embargo, esta estrategia suele demandar mayor disponibilidad de recursos, tanto de procesamiento como de memoria. Por otro lado, la estrategia de simplificación es más simple y sencilla de implementar, demandando menor cantidad de recursos, aunque produzca mallados con mayor cantidad de vértices que el mínimo posible.

Finalmente, también existen técnicas que usan estrategias híbridas, que mantienen un nivel detalle para el terreno y, cuadro por cuadro, de manera incremental, proceden con operaciones de refinamiento y/o simplificación según sea necesario en diferentes regiones del terreno.

6.3 Consecuencias del Uso de Nivel de Detalle

El uso de estas técnicas de nivel de detalle introduce efectos no deseados o artefactos apreciables en la visualización de los terrenos. Los terrenos representados con estas técnicas pueden presentar en ocasiones discontinuidades espaciales y temporales. La representación de regiones adyacentes con distintos niveles de detalle puede dar lugar a la aparición de *cracks* (o grietas) en la superficie de terreno. Por otro lado, la elección dinámica del nivel de detalle a utilizar, por ejemplo en una misma región, puede ocasionar que los cambios de geometría sean perceptibles por el usuario.

La discontinuidad espacial ocurre cuando triángulos adyacentes pertenecen a distintos niveles de detalle. Las grietas aparecen en los bordes de las regiones representadas con distinto nivel de detalle, ocasionados por aquellos vértices que se encuentran presentes en el nivel de detalle mayor, y ausentes en el otro (Fig. 4(a)). Estos vértices son llamados *vértice-T*, ya que el mismo se encuentra interconectado con una conexión en T (*T-junction)*. Generalmente, la aparición de un crack es ocasionada porque la altura del *vértice-T* no coincide con la altura con el cual es representada esa posición en el nivel de menor detalle, producto de la interpolación de los vértices presentes en este último.

Por otro lado, otro artefacto, que produce un efecto similar a cracks en el terreno, es también originado por los vértices-T. En el caso en el que la altura interpolada coincida con la altura original del vértice, también pueden observarse pequeños agujeros o cracks en la superficie del terreno. Estos agujeros son originados por diferencias de redondeo de punto flotante producidos entre las operaciones del pipeline gráfico. En la bibliografía, el efecto es llamado *bleeding tears* (lágrimas de sangre) [14].

Temporalmente, también pueden apreciarse discontinuidades. El artefacto es llamado *"popping"* (del verbo inglés *to pop*, saltar). El mismo se refiere a la percepción de *saltos ó pops* en la superficie del terreno, es decir, un cambio brusco y repentino en la geometría del terreno. Esto ocurre cuando hay un cambio en el nivel de detalle utilizado para la representación de una misma zona de terreno a medida que el observador se desplaza cuadro por cuadro.

Finalmente, el cambio de nivel de detalle de una región de un cuadro a otro, también puede producir efectos apreciables en el sombreado del terreno, dado que con la geometría también cambian las normales de las primitivas que son usadas en los cálculos de iluminación convencionales.