Representación multiresolución de sólidos mediante una jerarquía de planos envolventes

F.J. Melero, P. Cano y J.C. Torres
Dpto. Lenguajes y Sistemas Informáticos
E.T.S. Ing. Informática y de Telecomunicación
C/ Periodista Daniel Saucedo Aranda s/n, Granada
e-mail: fjmelero@ugr.es, pcano@ugr.es, jctorres@ugr.es

Resumen

En el presente trabajo se presenta una nueva estructura de datos que permite representar un sólido mediante una jerarquía de planos espacialmente indexados. Se hace uso de un árbol para realizar la jerarquización tridimensionalmente indexada. Los planos que correponden a los nodos de dicho nivel forman una envolvente convexa del sólido en el volumen comprendido por el nodo.

1. Introducción

La creación y posterior manipulación de grandes modelos tridimensionales es uno de los aspectos de la informática gráfica que centran mayores esfuerzos entre la comunidad científica internacional [4]. En general, las soluciones propuestas hacen uso de índices espaciales y diversas técnicas multiresolución, estando orientadas a la visualización en tiempo real del modelo tridimensional. Estas técnicas pueden ser combinadas con el uso de impostores [8] o texturas tridimensionales [2] para mejorar la calidad del modelo visualizado.

Para la detección de colisiones, es común el uso de árboles para la indexación espacial del modelo: octrees [9] , BSP-Trees, KD-Trees, SP-Octrees [3] y otros muchos. Tambíen se usan jerarquías de volúmenes envolventes (BVH, Bounding Volume Hierarchies), siendo estos volúmenes de diversos tipos: cajas envolventes alineadas a los ejes (AABB, [1]), cajas envolventes alineadas al objeto (OOBB)[5], k-DOPs [7], etc...

Algunas aproximaciones a la multiresolución permiten la visualización adaptativa [6], o incluso la aparición de detalles del modelo que permanecen ocultos cuando son inapreciables.

2. Construcción del árbol

Para la construcción del árbol se sigue una estrategia *bottom-up*, que pasamos a detallar.

2.1. Creación del índice espacial

El índice espacial se construye a partir del AABB del objeto. Esta caja se subdivide por cada dimensión en 2^l celdas, siendo l el mayor nivel de profundidad del árbol. Esta subdivisión nos conduce a un grid tridimensional de celdas.

A continuación se determina en qué celda del grid espacial se encuentra cada uno de los vértices del modelo, y se procede a la creación del árbol.

2.2. Creación del árbol

Cada celda del grid tridimensional a cualquier resolución -esto es, cada nodo del árbol- se identifíca únivocamente por su código Morton extendido -en adelante *oct-code*-, que tiene, entre otras propiedades, la de proporcionar una ordenación primero en anchura para los nodos identificados con dicho código, y la posibilidad de extraer a partir del octcode de un nodo n la ruta completa que conduce desde el nodo raíz al dicho nodo n.

La creación de los nodos hoja de nuestro árbol es inmediata, ya que no son más que las celdas del grid tridimensional que contienen al menos uno de los vértices del objeto. Cada nodo hoja de nuestro árbol contiene como datos de partida los polígonos a los que pertenecen los vértices incluidos en su celda equivalente del grid espacial.

Haciendo uso de la propiedad anteriormente indicada del octcode, referente a la definición de la ruta desde el nodo raíz hasta el objetivo, se construyen los nodos internos del árbol (fig. 2). Es de destacar que siguiendo este proceso, se consigue una optimización del uso de memoria, ya que no se crea ni un nodo que no vaya a contener información posteriormente.

Como último paso, se realiza una compactación del árbol, asegurando que todos los nodos hoja cumplen un criterio (p.ej. tener al menos 20 polígonos). Esta operacion nos proporciona una mejor distribución de los nodos hoja, apenas ocupándose los dos últimos niveles del árbol en modelos de menos de 1M polígonos.

3. Construcción de la jerarquía de planos

Otra propiedad interesante de la estructura propuesta tiene es que se conoce en el momento de la carga del modelo el máximo número de planos a utilizar: el número de polígonos del modelo. La lista de planos se construye en paralelo a la lectura del archivo con el modelo tridimensional, lo que permite utilizar una tabla con todos los planos y almacenar en los nodos del árbol únicamente los índices correspondientes.

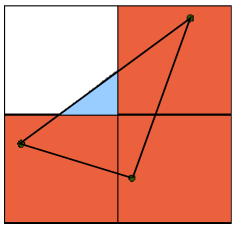

Figura 1: El triángulo es asignado a cuatro nodos

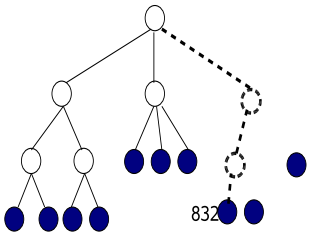

Figura 2: Construyendo el árbol

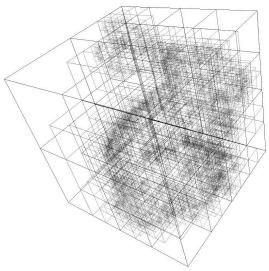

Figura 3: Jerarquía de nodos para el modelo bunny de 69K polígonos

3.1. Asignación de planos

Notando p_f al plano definido por el polígono f, asignaremos dicho plano a los nodos hoja que contienen a los vértices que forman el polígono f y a todos los nodos internos ancestros de estos nodos hoja. Esta asignación inicial de planos a nodos internos se realiza para evitar situaciones como las que se muestran en 2D en la figura 1.

3.2. Construcción de la envolvente

Para determinar el conjunto de planos H_n que forman la envolvente de cada nodo interno n se seleccionan aquellos planos tales que no dejan a ningún punto de los que forman la envolvente de todos los nodos hijos de n en el semiespacio exterior a dicho plano.

Para un nodos hoja l, en lugar de considerar los puntos de los nodos hijos, se consideran los puntos de la geometría final del modelo asociados a dicho nodo l.

3.3. Creación de nuevos nodos

Para evitar que ocurra lo que se muestra en la figura 1, una vez se han completado todas las envolventes en los nodos internos, se recorren y se completa su descendencia con unos nodos hoja "secundarios"que son aquellos que resultan atravesados por los planos almacenados según el criterio anteriormente indicado. Estos nodos hoja no contienen geometría final, pero sí una envolvente convexa de la parte de la frontera del modelo que le corresponde.

4. Resultados

En la versión completa de este artículo se detallarán los resultados obtenidos con esta estructura, y podrán observarse la rapidez y calidad de las representaciones

obtenidas, así como su aplicación a la detección de colisiones.

5. Agradecimientos

Este trabajo ha sido parcialmente financiado por el Ministerio de Ciencia y Tecnólogia (MCYT) bajo el proyecto TIN2004-06326-C03-02.

Referencias

[1] V. D. Bengen. Efficient collision detection of complex deformable models using aabb trees. *Journal of Graphics Tools*, 2(4):1–13, 1997.

[2] D. Benson and J. Davis. Octree textures. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 785–790, New York, NY, USA, 2002. ACM Press.

[3] P. Cano, J. Torres, and F. Velasco. Progressive transmission of polyhedral solids using a hierarchical representation. 2003.

[4] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno. Adaptive tetrapuzzles: efficient out-of-core construction and visualization of gigantic multiresolution polygonal models. *ACM Transaction on Graphics*, 23(3):796–803, 2004.

[5] S. Gottschalk, M. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. *Proceedings SIGGRAPH'96*, pages 171–180, 1996.

[6] H. Hoppe. View-dependent refinement of progressive meshes. *Computer Graphics*, 31(Annual Conference Series):189–198, 1997.

[7] J. Klosowski, M. Held, J. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *IEEE transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[8] F. Melero, P. Cano, and J. Torres. Combining sp-octrees and impostors for multiresolution visualization. *Computer and Graphics*, 2005.

[9] H. Sammet and R. Webber. Hierarchical data structures and algorithms for computer graphics. *IEEE Comp. Graphics and Applications*, 8(3):48–68, 1983. OCTREE.