Técnicas para la aceleración de la visibilidad en apilamientos

Miguel Escrivá, Rafael Gaitán, Javier Lluch, Emilio Camahort, Roberto Vivó

Sección de Informática gráfica

Departamento de Sistemas Informáticos y Computación

Universidad Politécnica de Valencia

Camino de Vera, s/n. 46022 Valencia

{mescriva, rgaitan, jlluch, camahort, rvivo}@dsic.upv.es

Resumen

En la actualidad los gráficos en tres dimensiones están siendo aplicados a un número creciente de tareas aunque no se suelen emplear en monitorización de procesos industriales, donde se suelen usar métodos 2D tradicionales.

En este artículo describimos las técnicas para el cálculo de la visibilidad (eliminación de geometria, back-face culling, portal rendering) y como las empleamos en la visualización de objetos apilados. Con el uso de estas técnicas hemos incrementado el número de imágenes por segundo hasta 18 veces.

1. Motivación

Los recientes avances en diseño 3D, en adquisición de datos y en simulación están generando conjuntos de datos de geometría cada vez más grandes, incluso llegando a exceder el tamaño de la memoria principal y superando las capacidades de visualización del hardware gráfico actual.

La representación interactiva de grandes cantidades de datos es una necesidad fundamental para varias aplicaciones gráficas. Aunque la capacidad de proceso de gráficos está aumentando cada día, su eficiencia no ha podido continuar con el aumento rápido en complejidad de los modelos. Para tratar este problema, se han desarrollado técnicas para el cálculo de la visibilidad [2] y así reducir la cantidad de geometría que es necesaria para dibujar la escena, manteniendo la exactitud de la imagen.

Nuestro trabajo se centra en el estudio de técnicas de aceleración de la visualización de grupos de objetos apilados para el uso en la supervisión de procesos industriales y exploración virtual. Integramos técnicas previamente desarrolladas, con las necesidades de una terminal de carga marítima.

Los objetos apilados son muy comunes en la industria, sobre todo en las empresas relacionadas con la logística y/o almacenaje. Las terminales marítimas de contenedores son un caso particular de este tipo de organizaciones. Los grandes conjuntos de datos catacterísticos de estas industrias presentan un reto al desarrollo de sistemas de simulación gráfica.

En la siguiente sección se realiza un resumen sobre la determinación de la visibilidad y qué tipos de algoritmos existen. Después se explicarán las técnicas de aceleración que se han usado en nuestro trabajo para acelerar la visualización de objetos apiladados. Por último veremos los resultados que se han obtenido y a qué conclusiones hemos llegado, así como algunas ideas para desarrollos futuros.

2. Antecedentes

Varias soluciones algorítmicas han sido propuestas para superar la diferencia entre la potencia del hardware actual y la complejidad del conjunto de la geometría[6, 12, 11, 3]; y suelen incrementar el número de imágenes por segundo debido a la disminución del número de triángulos enviados al hardware gráfico.

La determinación de la visibilidad es el proceso

en el cual se determina qué objetos son visibles y cuáles no. Los triángulos más rápidos son los que no se muestran, es decir, los que no son vistos por el usuario.

3. Técnicas de aceleración

Los grandes conjuntos de datos empleados en las terminales de contenedores hacen indispensable el uso de métodos de aceleración gráfica para generar simulaciones en tiempo real.

Nuestro sistema intenta resolver los problemas que rodean la visualización de grandes volúmenes de datos y que están sujetos a una colocación específica, como pueden ser objetos apilados en una terminal de contenedores. La visualización debe ser interactiva de modo que el usuario pueda tomar decisiones en tiempo real.

Nuestro sistema[10] está construido sobre proyectos éstandar de código abierto que proporcionan compatibilidad y un alto rendimiento. Para el subsistema gráfico, usamos OpenSceneGraph[1, 9].

OpenSceneGraph es un toolkit gráfico de alto nivel y portable para el desarrollo de aplicaciones gráficas de alto rendimiento tales como simuladores de vuelo, juegos, realidad virtual o visualización científica.

3.1. Eliminación de contenedores no visibles

Cuando hablamos de objetos apilados, hay mucha geometría que nunca es visible. Cuantos más objetos apilados y/o uno junto a otros, más partes de estos objetos no serán visibles.

Nuestro sistema modela una terminal de contenedores con una superficie de $1800m^2$ y unos 23000 contenedores. Muchos de estos contenedores no son visibles porque tienen otros contenedores apilados al lado. Por lo que las caras de los contenedores que no son visibles desde ningún ángulo son eliminadas del grafo de escena antes de la visualización. Esto nos permite visualizar sólo la superficie de cada manzana (que nosotros llamamos *manto*), y consigue una reducción importante del número de triángulos y un incremento del número de imágenes por segundo.

3.2. Back-face culling

Es fácil comprobar que de las seis caras de un contenedor sólo podemos ver como mucho tres en un determinado momento, así que con esta técnica es posible reducir el número de triángulos a dibujar en más de un 50 %.

3.3. Portal rendering

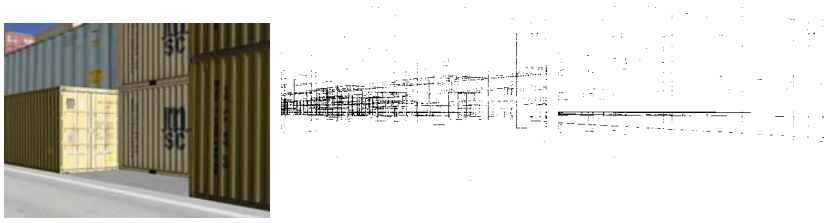
Hemos implementado soporte para portales[8] como otro sistema de *visualización*. Nuestro sistema puede usar *portal rendering*[5, 4] para acelerar la visualización. Los contenedores van a ser *nuestras paredes*(figura 1), que usamos como oclusores.

Cuando la aplicación arranca y despues de obtener información de los contenedores de la base de datos, esta se usa para crear sectores y portales automáticamente usando el algoritmo *Breaking the walls*[7].

La terminal de contenedores es un almacén enorme y si se crean los sectores para toda la terminal a la vez puede ser un proceso costoso. Lo que se ha hecho es aprovecharse de la estructura de la terminal. Como hemos dicho, la terminal de contenedores está formada por manzanas. Así pues en un primer paso se han creado los sectores y portales que nos permitirán movernos por las manzanas de contenedores (Figura 2).

Con esto conseguimos independizar una manzana de otra. Pero hay que tener en cuenta que con esta aproximación estamos suponiendo que las manzanas de contenedores están totalmente llenas de contenedores, que rara vez ocurre. Lo normal es que en una manzana falten contenedores lo que puede provocar que se vean contenedores que están en otra manzana. Así pues, hay que hacer un segundo paso y crear sectores y portales para cada una de las manzanas (Figura 2). Esto nos permitirá seguir atravesando portales cuando una manzana tenga huecos y dibujar los contenedores que se vean por esos huecos.

Para tratar el problema de que no todas las pilas de contenedores tienen la misma altura, se han de crear portales a diferentes alturas. En una terminal de contenedores, al igual que ocurre en otros almacenes, hay un límite de objetos que se pueden apilar. Así pues, una vez encuentras una zona de la terminal que debería de ser un sector, has de ver qué pilas de contenedores delimitan con esta zona y crear un



(a) Sólido (b) Sin portales (c) Con portales

Figura 1: Diferencias del uso o no de portal rendering. La figura 1(a) corresponde con la imágen en sólido, la 1(b) muestra la misma escena en alámbrico sin usar portal rendering y la 1(c) con el uso de portal rendering.

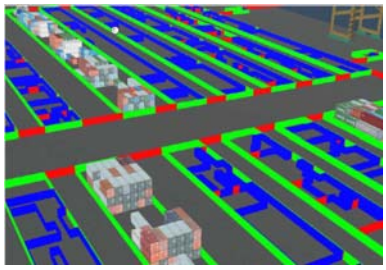


Figura 2: En esta imágen muestra como se han creado los sectores y portales.

portal en esa pila hasta el límite máximo permitido para apilar los objetos.

4. Resultados

En este apartado vamos a comentar qué mejoras hemos obtenido y qué conclusiones podemos extraer de los resultados recogidos.

Para las pruebas se ha almacenado una animación de un recorrido por la terminal de contenedores de 80 segundos de duración.

En la figura 3, podemos ver una tabla a modo de resumen donde se indica el número de imágenes por segundo medio obtenido y el incremento que se obtiene al usar cada una de las técnicas empleadas.

Puede observarse cómo el uso del manto provoca un incremento de alrededor un 63 % y el incremento de Back-Face Culling es de un 98,77 %. El uso de portales nos permite obtener un número de imágenes por segundo sostenido tal como puede verse en

Algoritmo	Frame-rate	Incremento
Todos los triángulos	3,27	—
Uso del manto	5,33	×1,62
Backface culling	6,5	×1,98
Uso de portal rendering	60,65	×18,5

Figura 3: Tabla de incremento del número de imágenes por segundo

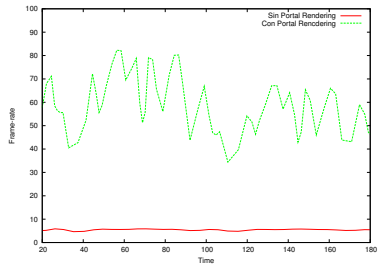


Figura 4: Comparación del número de imágenes por segundo con y sin el uso de portales

la figura 4.

El uso de portales en nuestro sistema multiplica el número de imágenes por segundo más de 18 veces. Se debe a que hemos reducido considerablemente el número de triángulos a visualizar, como se aprecia en la figura 5.

5. Conclusiones y trabajo futuro

Hemos presentado un sistema de simulación multiplataforma capaz de representar una terminal

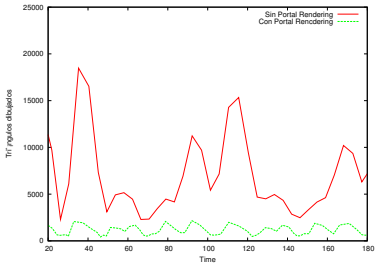


Figura 5: Número de triángulos dibujados

marítima de contenedores en 3D.

Se han visto diferentes técnicas para el cálculo de la visibilidad (eliminación de geometria, back-face culling, portal rendering) y como las empleamos en la visualización de objetos apilados. Con el uso de estas técnicas se ha conseguido incrementar el número de imágenes por segundo hasta 18 veces.

Estamos trabajando en mejorar la calidad de las imágenes generadas aún más usando técnicas basadas en shaders. Otras áreas interesantes son la del uso de oclusores y la mejora del algoritmo de portales para sistemas con bajas especificaciones tales como PDAs o teléfonos móviles; o escenas de tamaño aún mayor. Algunas de las ideas para este tipo de dispositivos son centrarse en lo que al usuario le interesa ver, es decir si el usuario está navegando por un pasillo de la terminal, centrarse en las manzanas de contenedores adyacentes a ese pasillo que es lo importante para él.

6. Agradecimientos

El proyecto Virtainer está siendo subvencionado por el Ministerio de Ciencia y Tecnología (TIC2002-04166-C03-01), y también por el proyecto GameTools (IST-2-004363-STP). Agradecemos a los equipos de Virtainer y Gama su colaboración. También estamos agradecidos a Marítima Valenciana por su apoyo a nuestros esfuerzos de investigación.

Referencias

[1] D. Burns and Robert Osfield. Open scene graph - an open source solution for real time image rendering, 2003.

[2] Daniel Cohen-Or, Yiorgos Chrysanthou, Claudio T. Silva, and Fredó Durand. A survey of visibility for walkthrough applications. In *EUROGRAPHICS'00, course notes, 2000.*, 2000.

[3] Jihad El-Sana, Neta Sokolovsky, and Claudio T. Silva. Integrating occlusion culling with view-dependent rendering. In *Proc. of Visualization 2001*, 2001.

[4] Niklas Elmqvist. *Axis-Aligned Bounding Box Culling For Portal Rendering*. Chalmers Medialab, 1999.

[5] Niklas Elmqvist. *Introduction to Portal Rendering: Indoor Scene Rendering Mode Easy*. Chalmers Medialab, 1999.

[6] Subodh Kumar, Dinesh Manocha, William Garrett, and Ming Lin. Hierarchical backface computation. In *Xavier Pueyo and Peter Schröder, editors, Eurographics Rendering-Workshop*, pages 235–244, 1996.

[7] Alon Lerner, Yiorgos Chrysanthou, and Daniel Cohen-Or. Breaking the walls: Scene partitioning and portal creation. In *Pacific Graphics 2003*, 2003.

[8] David P. Luebke and Chris Georges. Portals and mirrors: Simple, fast evaluation of potentially visible sets. In *Proceedings 1995 Symposium on Interactive Computer Graphics*, pages 105–106, 1995.

[9] OSG Community. *OpenSceneGraph - Open Source high performance 3D graphics toolkit*, 2004. http://www.openscenegraph.org.

[10] Sección de Informática Gráfica - DSIC. *Virtainer: Technologies for the virtual interaction with groups of objects in ordered stacks*, 2004. http://www.virtainer.org.

[11] Hansong Zhang. *Effective Occlusion Culling for the Inveractive Display of Arbitrary Models*. PhD thesis, Department of Computer Science, University of North Carolina at Chapel Hill, 1998.

[12] Hansong Zhang and Kenneth E. Hoff III. Fast backface culling using normal masks. In *Symposium on Interactive 3D Graphics*, pages 103–106, 1997.