UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 5
ENTORNOS VIRTUALES

Título: "Selección de visibilidad mediante Frustum Culling y Coherencia Temporal"

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores: Diosky Rodríguez Nuñez
Gabriel Martínez Molina

Tutor: Ing. Elieser García Ramos.
Cotutor: Ing. Omar Correa Madrigal.

Ciudad de la Habana, Junio de 2010

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____del año _____.

Diosky Rodríguez Nuñez

_____

Firma del Autor

Gabriel Martínez Molina

_____

Firma del Autor

Ing. Elieser García Ramos

_____

Firma del Tutor

Ing. Omar Correa Madrigal

_____

Firma del Cotutor

Agradecimientos

A mis padres por su apoyo incondicional y sacrificio, por tener paciencia para educarme e inculcarme los mejores valores que puede tener un ser humano.

A mis suegros por ser unos padres más, por estar siempre pendiente de mí y de mi universidad, por apoyarme en todo momento y confiar en mí.

A mi esposa por confiar siempre en mí, por apoyarme en todo momento y darme lo más grande que tengo en mi vida a mi hermoso hijo.

A toda mi familia por apoyarme y estar siempre al tanto de mis estudios.

A mis profesores que me han enseñado y apoyado en todos estos años de universidad.

A todos mis amigos por apoyarme en todo momento y estar pendiente de mí.

Diosky.

A mi mamá y mi padrastro por hacerme la persona que soy y estar siempre a mi lado en todos lo momentos.

A mi hermano por ser lo que más quiero en este mundo.

A mi papá por todo lo que significa para mí.

A mi familia y amistades.

A la universidad y a María por brindarme los mejores días de mi vida.

A mis tutores por su apoyo en la realización de este trabajo.

A todos los que de una forma u otra aportaron su granito de arena.

Y en especial a todos mis niños, por todo lo que somos..

Gabriel.

Dedicatoria

Le dedico este trabajo a mis padres, a mis suegros, a mi esposa y mi hermoso nené, a todos muchas gracias por su esfuerzo, dedicación y por confiar siempre en mí.

Diosky.

Este trabajo se lo dedico a mis tres padres y a mi hermano por ser lo que más quiero en la vida, a todas mis amistades, a mi familia y a Diosky por confiar en mí.

Gabriel.

Resumen

El desarrollo de los videojuegos es un tema muy tratado en el mundo actual, la industria de videojuegos ha tenido en los últimos años un gran auge y altas tasas de crecimiento. La Universidad de las Ciencias Informáticas (UCI), ha incursionado en ese campo, para esto se está desarrollando en la facultad cinco de dicha universidad un motor gráfico llamado SceneToolKit (STK), que permite la realización de estos videojuegos.

En el presente trabajo se propone un algoritmo para la optimización del proceso de visualización utilizando la técnica de View Frustum Culling y la Coherencia Temporal para ser incluido en el motor gráfico antes mencionado.

El objetivo consiste en desarrollar una técnica que mediante la utilización de la Coherencia Temporal permita simplificar la técnica de View Frustum Culling. Es importante señalar que no existe actualmente un algoritmo igual a este, hay otros que usan la Coherencia Temporal pero no explotan el sello de tiempo. Siendo intención de la investigación utilizar el mismo en la STK, sin descartar que se pueda integrar a otros motores gráficos.

Índice

Introducción

El desarrollo de los videojuegos es un tema muy tratado en el mundo actual. La industria de videojuegos ha experimentado en los últimos años altas tasas de crecimiento, debido al desarrollo de la computación, capacidad de procesamiento, imágenes más reales y la estrecha relación entre películas de cine y los videojuegos.

Nuestro país ha avanzado notablemente en ese ámbito y la Universidad de las Ciencias Informáticas (UCI) ha tenido un papel fundamental y decisivo. Los videojuegos que se realizan en la universidad, están muy relacionados con las prestaciones de la máquina (RAM, procesador), entre otras. Se desea realizar juegos con una óptima calidad visual, pero que no consuman tantos recursos, porque no en todos los lugares donde se van a implantar se cuenta con una tecnología de punta capaz de soportar muchas prestaciones de la máquina, y aunque se tuviera esa tecnología, los videojuegos cada día exigen que sean más reales, lo cual llevaría a un incremento de estas prestaciones.

La facultad 5 de la Universidad de las Ciencias Informáticas es la encargada del desarrollo de varios de estos videojuegos y para lograrlos entre otras técnicas ha desarrollado el motor gráfico SceneToolKit (STK). El cual es usado para la creación de videojuegos, este motor gráfico tiene numerosas funcionalidades, de sonido, tratamiento de imágenes, visualización de objetos entre otras. Este último es un proceso muy trabajoso, ya que tiene en cuenta muchos aspectos en el entorno del juego. Dentro de las técnicas de visualización de los objetos en la escena hay una muy importante llamada View Frustum Culling que es donde se escogen qué objetos del entorno son los posibles a mostrar en ese instante, esta técnica no tiene en cuenta el movimiento de la cámara, ni su velocidad para hacer dicha selección.

Analizando detenidamente la situación problemática antes expuesta se ha definido el siguiente **problema científico**: ¿Cómo tener en cuenta el movimiento de la cámara, y su velocidad, para optimizar el proceso de selección de visibilidad?

De esta forma y teniendo en cuenta lo anteriormente planteado, el **objeto de estudio** de este trabajo consiste en los métodos de visibilidad para motores gráficos.

El **campo de acción del trabajo** comprende el estudio de las técnicas de selección de visibilidad View Frustum Culling y Coherencia Temporal**.**

El **objetivo general** de este trabajo es elaborar una técnica capaz de optimizar el proceso de selección de visibilidad, mediante las técnicas View Frustum Culling y Coherencia Temporal.

Para dar solución al problema existente se proponen las siguientes tareas a cumplir:

1. Realización de un estudio sobre las tendencias y técnicas más utilizadas para la visualización de objetos en pantalla.
2. Realización del diseño e implementación de la técnica View Frustum Culling para acoplarla a la Coherencia Temporal.
3. Realización del diseño e implementación del algoritmo de Coherencia Temporal para optimizar el proceso de selección de visibilidad en la técnica de View Frustum Culling.
4. Realización del diseño e implementación del algoritmo de Traslación para las diferentes tendencias de la cámara en traslación.
5. Realización del diseño e implementación del algoritmo de Rotación para las diferentes tendencias de la cámara en rotación.
6. Integración de los algoritmos de Traslación y Rotación con el algoritmo Frustum Culling para un acoplamiento entres estos.

Por lo antes planteado se tiene la siguiente **idea a defender:** Si se tiene en cuenta la técnica de Coherencia Temporal, y la de View Frustum Culling se optimizará el proceso de visualización de los objetos en la escena.

Métodos de investigación

1. Método Histórico Lógico: Para realizar un análisis de los principales déficits de visualización mantenidos en las distintas versiones del motor gráfico (STK), fundamentalmente en la técnica View Frustum Culling.

Capítulo 1

Fundamentos teóricos actuales sobre la selección de visualización.

1.1 Optimización de la visualización en entornos virtuales.

1.1.1 Tendencias a optimizar la visualización.

La demanda en aumento de realismo tanto en la complejidad de las escenas como en la velocidad de las animaciones hace que los programadores tengan que hacer un esfuerzo excesivo para lograr nuevas técnicas de optimización de la visualización. Paralelo a esto se tiene un crecimiento en las capacidades de cómputo del hardware lo que deja las puertas abiertas a otras técnicas de optimización donde se encuentran las siguientes:

Utilización de estructuras de datos espaciales: Las estructuras de datos espaciales nos permiten almacenar objetos de la escena, así como organizar el espacio multidimensional de esta. Algunas de las ventajas que trae consigo el uso de estas estructuras vienen dadas por el alto grado de organización que presenta, facilitando las búsquedas dentro de los mismos. Entre las estructuras de datos más óptimas se encuentran Octree, K_Dtree, Bsp_tree, Rejilla Regular y Grafos de Escena.

Utilización de las técnicas de determinación de visibilidad: Con el uso de estas técnicas se logra un mayor rendimiento de los recursos disponibles. Estas se encargan de seleccionar cuales son los polígonos potencialmente visibles para visualizarlos en la escena y descartar en la medida de lo posible aquellos que sean innecesarios en la misma. Entre las técnicas de visualización más usadas están aquellas encargadas de descartar un grupo de polígonos en dependencia de las etapas en que se encuentren, entre ellas encontramos: Backface Culling, Detail Culling, Occlusion Culling y View Frustum Culling, entre otras.

1.1.2 Espacios de optimización.

Dentro de los espacios de optimización encontramos el espacio imagen o bidimensional, el que presenta dos dimensiones (ancho y largo) y se representan imágenes en dos dimensiones, este espacio es una proyección del espacio objeto o tridimensional, el que, como su nombre lo indica tiene tres dimensiones (ancho, largo y profundidad) y se pueden representar objetos tanto planos como volumétricos.

1.2 Proceso de selección de visibilidad a alto nivel.

El proceso de **selección de visibilidad a alto nivel** es muy complejo y costoso, cuenta con una serie de técnicas encargadas de enviar a la Unidad de Procesamiento Gráfico (por sus siglas en ingles GPU) la menor cantidad de objetos y polígonos a procesar sin que afecten la calidad de la imagen, permitiendo que se libere carga de la escena y así incrementar el rendimiento de nuestro programa. A continuación se explicará brevemente el proceso de selección de visibilidad a alto nivel con las técnicas más usadas.

Una de las técnicas más usadas es la del View Frustum Culling. Para ello cada objeto es encuestado con los planos del frustum (el frustum es una clase de pirámide truncada en entornos 3D que representa lo que visualiza la cámara, Fig. 1) y el objeto que se encuentre fuera de esta es eliminado del conjunto de posibles objetos a visualizar, evitando el posterior procesado de estos.

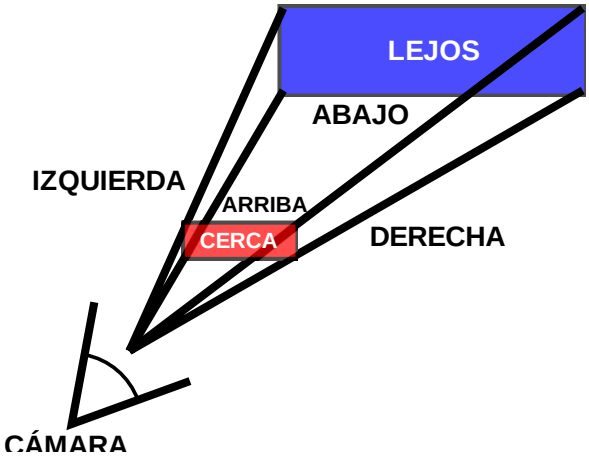


Fig. 1: Frustum 3D.

Posteriormente se realiza un análisis que trata de eliminar aquellos objetos que están ocultos detrás de otros como se muestra en la figura 2, y por lo tanto no contribuyen a la imagen final pero sí son procesados, reduciendo considerablemente el rendimiento de nuestro programa, esta técnica es conocida como Occlusion Culling.

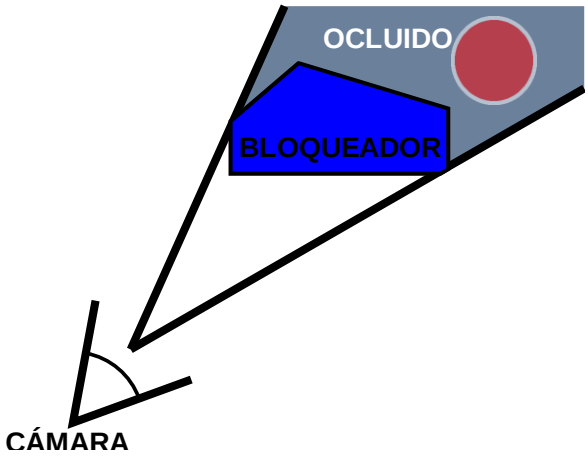


Fig. 2: Occlusion Culling.

Otra de las técnicas usadas para la selección de objetos a renderizar es la de Niveles de Detalles (LOD) (Levels of Details). Los LODs o Niveles de Detalle fueron introducidos por primera vez por Clark en 1974 en el desarrollo de simuladores de vuelo interactivos. La idea básica tras los LODs consiste en usar versiones más simples de un objeto para las distancias lejanas porque si pretendemos pintar un objeto lejano que posee por ejemplo 10000 polígonos, y en pantalla ocupa solamente 10x5 pixeles, ganaremos un gran incremento de velocidad si lo dibujamos con otro modelo equivalente de 100 polígonos, siendo difícil notar la diferencia [1], Fig. 3.

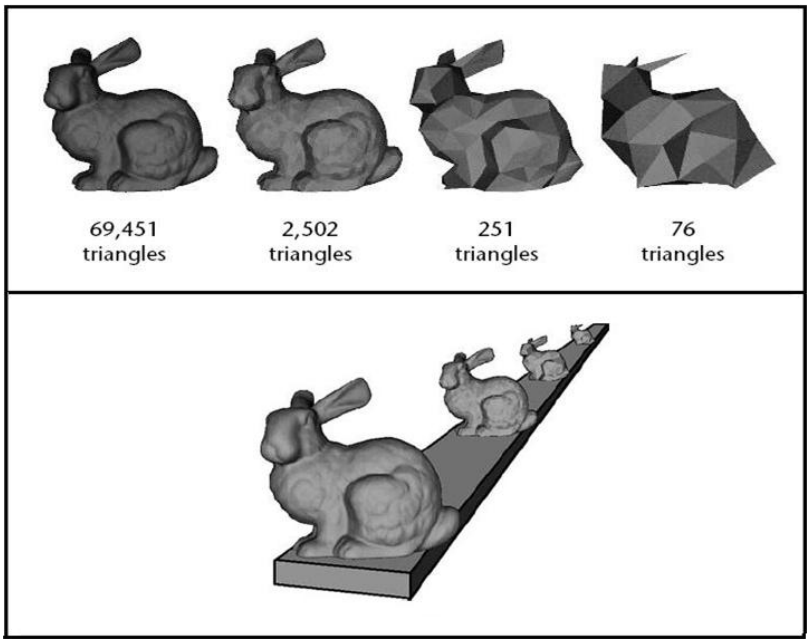


Fig. 3: Niveles de detalles.

Por lo antes planteado se puede deducir que la técnica de View Frustum Culling es una de la más importante para la selección de objetos a visualizar, debido a que con una mayor optimización de esta las demás se verán favorecidas ya que no tendrán que encuestar objetos innecesarios.

1.3 Selección mediante la técnica View Frustum Culling.

1.3.1 Tipos de frustum.

Existen 2 tipos de frustum (2D y 3D), los cuales se utilizan para marcar el área o volumen que se quiere representar. A continuación se explican brevemente en que consisten.

El volumen de visión o frustum 3D, está formado por una pirámide truncada de seis planos (cerca, lejos, arriba, abajo, izquierda, derecha) de forma tal que los objetos que se encuentren fuera de éste no son visibles por la cámara y no se enviarán al pipeline gráfico, Fig. 1.

El área de visión o frustum 2D, es el resultado de la proyección del View Frustum 3D sobre el plano XY. Este frustum está delimitado por rectas como se muestra en la Fig. 4 [6] al igual que el volumen de visión 3D los objetos que se encuentren fuera de estas rectas son descartados.

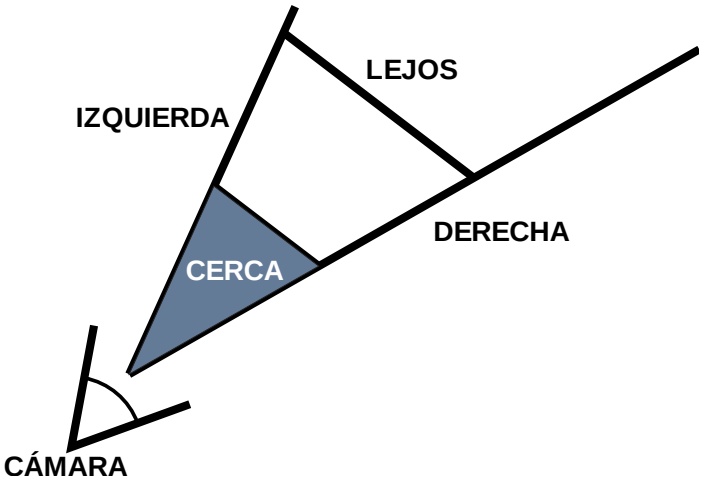


Fig. 4: Frustum 2D.

1.3.2 Tipos de volumen de encierro.

Los volúmenes envolventes o volúmenes de encierro son objetos contenedores de los objetos de la escena, estos tienen como objetivo simplificar su geometría y así reducir el coste de operaciones como el recorte, visibilidad y detección de colisiones. La jerarquía de volúmenes envolventes es muy usada para acelerar el renderizado de la escena, cada nodo de una estructura jerárquica es un volumen envolvente (VE) que incluye parte de la escena. Si durante el recorrido de una estructura jerárquica un VE se encuentra fuera del frustum los contenidos de éste no son procesados, reduciendo así el tiempo de la técnica View Frustum Culling, lo cual permite un incremento del microprocesador y liberar procesos para otras tareas. [5]

De los volúmenes envolventes los más utilizados en la creación de los grafos de escenas son los AABB (Axis Aligned Bounding Boxes), Esferas Frontales (Bounding Spheres) y OBB (Oriented Bounding Boxes).

AABB: Son cajas alineadas por tres valores X, Y, Z máximos y tres X, Y, Z mínimos, esto tiene como ventaja que el cálculo de la normal y la prueba de inclusión de un punto en el objeto son fáciles de obtener, el problema de este volumen recae en que no se ajusta de la mejor manera a los objetos, Fig. 5.

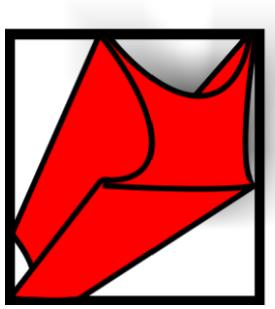


Fig. 5 AABB

Esferas Frontales: Los volúmenes envolventes de esfera están definidos por un punto y un radio, son muy rápidos a la hora de encuestar el volumen de encierro, ya que solo hay que verificar si la esfera está interceptando el frustum. El problema radica en que si el frustum intercepta la esfera y los objetos contenidos no se ven, se van a estar pintando sin estar dentro del campo de visión, Fig 6.

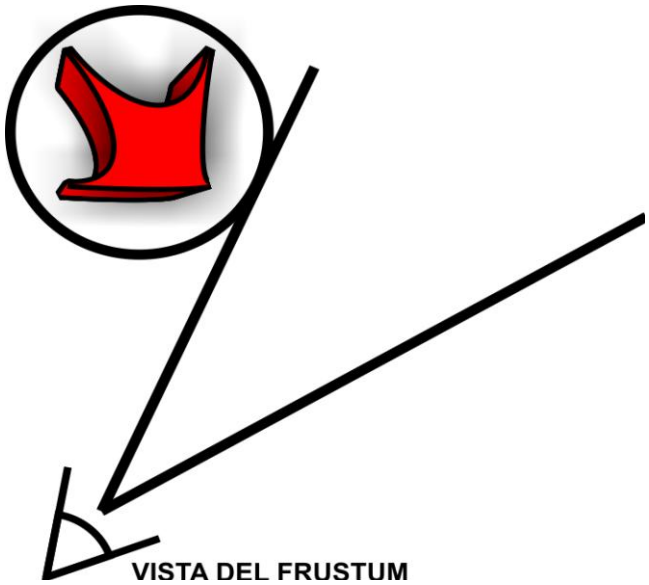


Fig. 6: Esferas frontales (bounding spheres).

OBB: Son cajas definidas por cuatro u ocho puntos orientados a los objetos y se adaptan más a su forma, permitiendo que la encuesta sea más exacta, pero a la vez más compleja, Fig. 7.

•Problema: Encontrar la orientación de la caja que mejor se ajusta al objeto.

•Idealmente se desea el mínimo volumen para la caja, pero esto es muy costoso

Fig 7 OBB

1.3.3 Estructuras Jerárquicas.

1.3.3.1 Grafo de escena.

Es un grafo dirigido y acíclico de nodos que contiene los datos que definen un escenario virtual y controlan su proceso de dibujado. Esta colección de nodos formando un grafo o árbol con la restricción de que un nodo puede tener varios hijos pero solo un único padre, por lo que una operación aplicada a un nodo se propaga a todos sus descendientes, Fig 8. También se puede afirmar que un grafo de escena permite crear una estructura jerárquica de la escena formada por diversos tipos de nodos que ejercen influencia sobre sus nodos hijos.

Ventajas:

1. Contribuye a establecer una organización lógica de la escena.
2. Establece dependencias jerárquicas entre distintos sistemas de referencia.
3. Posibilita el proceso de selección entre múltiples niveles de detalles.
4. Posibilita el proceso automático del Culling (eliminación automática de los objetos que se encuentran fuera del campo de visión.

5. Facilita el control de la escena por parte del usuario.
6. Hace más cómodo el acceso a las bibliotecas gráficas de bajo nivel (OpenGl en este caso).

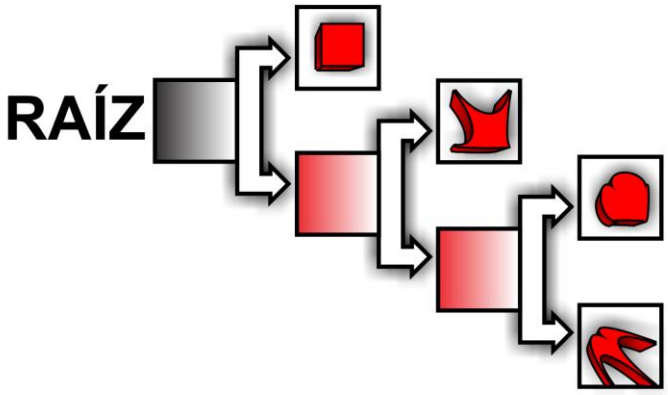


Fig. 8: Grafo de escena.

1.3.3.2 Quadtree.

Esta estructura se utiliza mayormente para representar escenas en dos dimensiones, ya que se basa en la descomposición recursiva del espacio, donde el centro de subdivisión está siempre en un punto y al insertar un elemento nuevo el espacio queda dividido en cuatro cuadrantes y así respectivamente con cada elemento que se inserte representando cada nodo un cuadrante de la escena, Fig. 9.

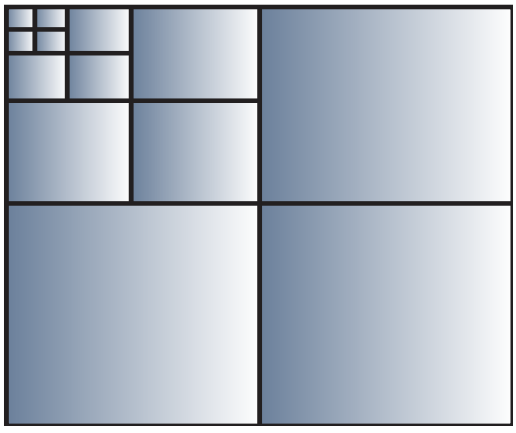


Fig. 9: Quadtree.

Ventajas:

1. Agiliza cálculos sobre las zonas.
2. Ofrecen una resolución variable representando únicamente aquellos detalles característicos de dicha resolución.[8]
3. Respeta la descomposición disjunta del espacio y tienen un grado mayor de independencia de datos. [8]

Desventajas:

1. No es óptimo cuando debe almacenar gran cantidad de diferentes píxeles existentes en una imagen con muchos colores, dado que podría tomar un tamaño excesivamente grande.
2. Por ser esta estructura de datos de memoria principal, al querer representar imágenes muy grandes, muchas veces no puede ser almacenado en dicha memoria.
3. El espacio se descompone en bloques de tamaño uniforme.

1.3.3.3 Octree.

Esta estructura fue introducida por Andrew Glassner en 1984 para la subdivisión del espacio objeto. A diferencia del Quadtree esta estructura se utiliza para representar escenas en tres dimensiones debido a que al insertar un nuevo elemento se subdivide el espacio en ocho cubos de igual tamaño, donde cada cubo a la vez puede ser subdividido. Al igual que el Quadtree cada nodo representa un cubo de la escena guardando toda la información de ésta. En general, la subdivisión de los espacios mediante Octree se basa en colocar varios voxels (octantes) pequeños en los sitios donde hay muchas geometrías, y por el contrario, en los lugares que contengan pequeñas y pocas geometrías se utilizan pocos y grandes voxels, Fig. 10.

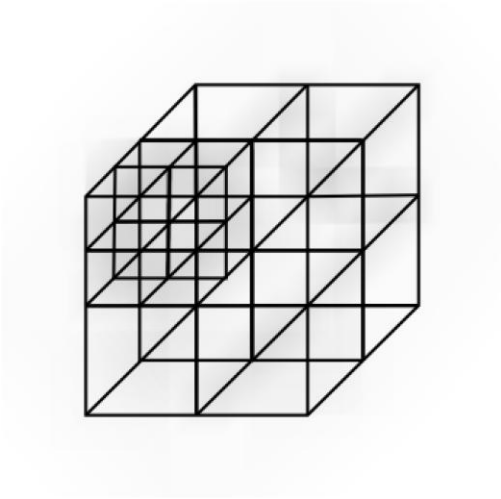


Fig. 10: Octree.

Esta técnica no es muy beneficiosa para tratar escenas dinámicas, pues la actualización de la estructura de datos no es trivial como en otros casos, especialmente si es necesario reconstruir el Octree sobre la marcha, lo cual sucede muy a menudo. [13]

Ventajas:

1. El orden implícito que se puede establecer entre los nodos facilita el proceso de visualización del modelo definiendo su orden de representación.
2. Permite optimizar la representación interna de escenas. [9]
3. Permite un manejo más eficiente de la memoria y capacidad del sistema. [10]
4. Permite el cambio de detalles de manera más suave y automática. [11]

Desventajas:

1. Es difícil de implementar.
2. No se acomodan bien para las escenas dinámicas puesto que reconstruir un Octree incluso parcialmente es muy costoso. [13]

1.3.4 Estructuras no jerárquicas.

1.3.4.1 Rejilla regular.

Esta estructura explota el espacio objeto realizando una división regular de este como se muestra en la Fig.11, la cual divide la escena en celdas cuadradas todas del mismo tamaño. Se trata de un arreglo bidimensional o tridimensional dependiendo del espacio que explote y son muy apropiadas para escenas dinámicas si se combinan con volúmenes envolventes. Esta estructura presenta algoritmos sencillos para su utilización y permite el acceso directo a posiciones intermedias de la escena.

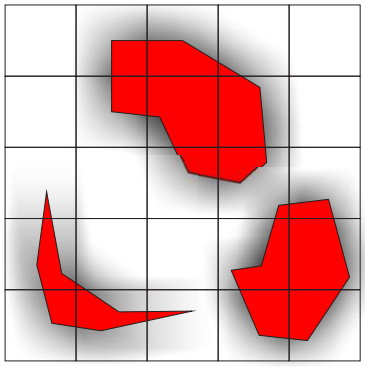


Fig. 11 Rejilla Regular

Ventajas:

1 Presenta algoritmos sencillos para su manejo.

2 Permite el acceso directamente a posiciones intermedias de la escena.

Desventajas:

1. Al ser homogénea no se adapta a las diferentes concentraciones de primitivas que pueda haber en una escena. es importante la elección del tamaño debido a que si los tamaños de las celdas es pequeño, existirán celdas con una cantidad excesiva de primitivas. Y si las celdas son muy

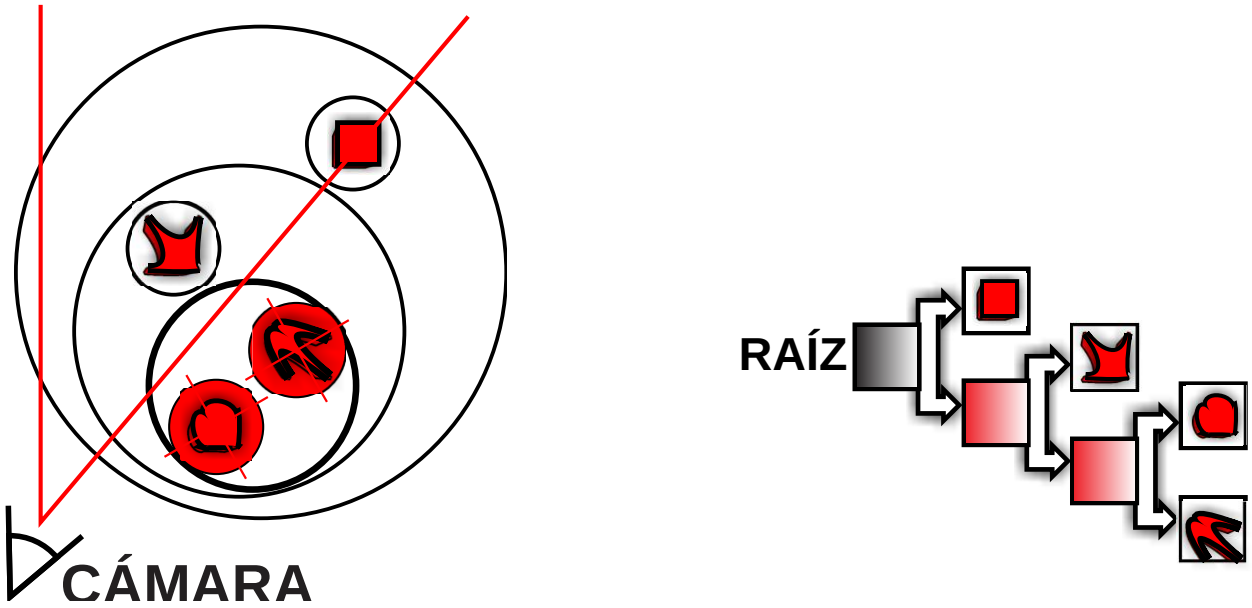
grandes existirá un gran número de celdas vacías que requieren mayor tiempo para ser atravesadas. [12]

2. Debido a su homogeneidad, no se adaptan a las distintas concentraciones de de primitivas que puedan existir en una escena. [12]

1.3.5 Técnica View de Frustum Culling.

La selección de visibilidad mediante la técnica View Frustum Culling se basa en la creación de un volumen de observación en forma de una pirámide de base cuadrada creada desde la posición del observador, que está formada por los planos, izquierdo, derecho, cerca, lejos, arriba y abajo. El objetivo de esta técnica es descartar los objetos que se encuentran fuera del volumen de visualización, o sea, antes de dibujar la escena para el frame actual se prueba cada objeto para ver si está total o parcialmente dentro del campo de visión. Los objetos que no se encuentren en este campo serán descartados y borrados de la lista de posibles objetos a dibujar.

Para una mayor optimización del proceso de visualización, se pueden usar diferentes estructuras de datos espaciales, como Grafos de Escena, Octree, Quadtree, o Rejillas Regulares, entre otras, Fig.12. También se pueden aplicar los Bounding Volumen (Volúmenes Envolventes) para calcular con mayor eficiencia la intersección entre los objetos y el volumen de observación.



12: Grafo de escena.

1.4 Coherencia Temporal.

La Coherencia Temporal está estrechamente relacionada con el tiempo. Su ventaja radica en la forma en que se aprovecha la permanencia de algún objeto en un espacio determinado en cierto período de tiempo. Dentro de esta encontramos los algoritmos de Volúmenes de Fronteras Temporales (TBV) y la técnica Coherencia de Cuadro [4].

1.4.1 Volúmenes de fronteras temporales.

Reconstruir una estructura espacial cada vez que se mueve un objeto es muy costoso, lo mejor sería actualizarla, con esta problemática se enfrentaron Sudarsky y Gotsman y propusieron el algoritmo de los Volúmenes de Fronteras Temporales (por sus siglas en inglés TVB), el cual plantea que lo ideal para evitar estas reconstrucciones de estructuras de datos sería ignorar la actualización de los objetos cierto período de tiempo, teniendo en cuenta solamente el espacio donde se pueden encontrar según el tiempo y si son visibles o no [4] , Fig. 13.

Fig. 13: TVB con un tiempo de validez de 1s.

Para construir el TBV para un objeto se necesitan algunos conocimientos sobre su movimiento. Afortunadamente muchos objetos tienen algunas restricciones que se pueden usar, si sólo el máximo de velocidad o el máximo de aceleración son conocidos, el TBV puede ser construido. Estos son construidos en tiempo de ejecución y en aplicaciones donde el flujo de eventos es desconocido de antemano.

En general, la principal ventaja de este método radica en que es compatible con otras técnicas, y en la mayoría de los casos, resulta muy beneficioso por su alta sensibilidad a la salida.

1.4.2 Coherencia de cuadro (Frame Coherency).

La coherencia de cuadro se basa en la reutilización de los resultados de los frames anteriores para una mayor aceleración del frame actual, por ejemplo: Si un objeto es visible en el actual cuadro, éste puede ser visible en los subsiguientes, según cierto período de tiempo y de esta manera se evita el cálculo de visibilidad permitiendo con esto que la técnica View Frustum Culling optimice su proceso, ya que los objetos a encuestar serían menos.

1.5 Algoritmo prueba de coherencia TR.

Este algoritmo fue introducido por Assarsson y Tomas Möller donde incluyen la coherencia de traslación y rotación para objetos estáticos. [7]

Este algoritmo plantea que si se está realizando una rotación pura, y si por ejemplo un objeto se encuentra fuera del volumen de visión por el plano izquierdo, éste seguirá estando fuera en el próximo frame si la cámara rota a la derecha a una amplitud de 180 grados – el ángulo de apertura de la cámara.

Cuando la cámara realiza una traslación pura en una cantidad Δd (Distancia), todos los objetos se alejan o se acercan a los diferentes planos del Frustum en esa misma cantidad Δd.

Si calculamos la proyección de Δd a lo largo de las normales de cada uno de los planos y teniendo además las distancias críticas de cada objeto a dichos planos, podemos establecer lo siguiente:

Si la proyección Δd'1 que es la proyección de Δd en plano 1 del Frustum es mayor que la distancia L11, que es la distancia crítica del objeto 1 al plano 1 del Frustum, podemos decir que la posición relativa del objeto 1 con respecto al plano 1 ha cambiado, es decir, si el objeto 1 estaba dentro del Frustum con respecto al plano 1 ahora está afuera y viceversa.

Esta prueba se realiza para cada uno de los planos del Frustum y cada uno de los objetos. Si se cumple para algún objeto y algún plano del Frustum, entonces este objeto ha cambiado su posición relativa con respecto a ese plano del Frustum. Luego, si se detecta que el objeto está afuera con respecto a algún plano del Frustum, entonces el objeto está definitivamente fuera del campo de visión, Fig.14.

Para una mejor comprensión del algoritmo antes planteado se utilizó un Frustum 2d y se delimitaron solamente 2 de los planos.

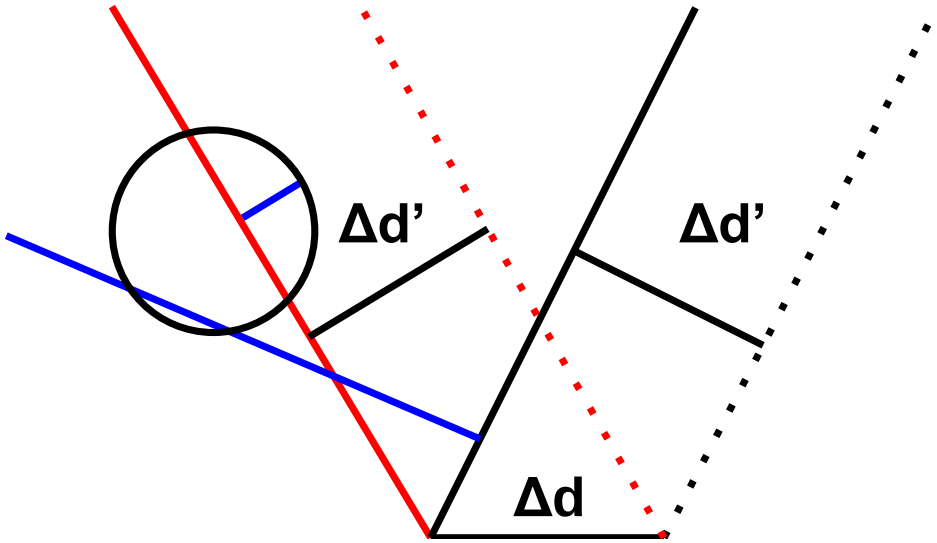


Fig.14: Traslación pura del volumen de visión, plano izquierdo de color rojo y plano derecho de color negro.

Las líneas en color azul representan las distancias del volumen de encierro con respecto a la normal de cada plano, esto significa que esta es la distancia mínima para que el objeto cambie de posición relativa a estos planos.

Deficiencias del algoritmo.

Como se pudo observar el algoritmo planteado solo tiene en cuenta para el caso que la cámara esté realizando una rotación pura, que un objeto seguirá estando fuera del volumen de visión si la cámara rota en sentido contrario una amplitud menor a 180 grados – el ángulo de apertura, pero si cambia la dirección de rotación el algoritmo no es eficiente, tampoco tienen en cuenta el tiempo en que el objeto puede seguir estando fuera del volumen de visión.

En el caso que la cámara esté realizando una traslación pura, en cada frame tienen que encuestar a todos los objetos con todos los planos y verificar si la proyección de la distancia en cada plano es mayor que la distancia crítica al plano correspondiente, tampoco tiene en cuenta el tiempo que estos objetos serán visible o no.

Capítulo 2

Soluciones técnicas

Introducción.

En este capítulo se proponen las soluciones técnicas para lograr optimizar el proceso de visibilidad con la técnica View Frustum Culling en cuanto a cantidad de objetos a encuestar, y se explicarán las ventajas que traerá consigo su uso.

Se detallan los pasos de los algoritmos de A. Traslación y A. Rotación, así como la aplicación de estos algoritmos en la determinación de la visibilidad de los objetos en la escena.

En la actualidad los motores gráficos generalmente usan la técnica de View Frustum Culling, para conocer que objetos deberá dibujar el motor de render en la escena para esto encuestan a todos los objetos de la escena verificando cual debe renderizar y cual no, lo que trae como consecuencia una mayor demora en el dibujado de esta. El motor de render STK (motor gráfico para el cual se realizó la técnica propuesta en este capítulo), también hace uso de esta técnica.

Después de realizar un estudio detallado de algunas de las diferentes técnicas de selección de visibilidad de objetos en escenas, y de cómo dibuja los objetos el motor gráfico STK, se creó la técnica propuesta en este capítulo, que es novedosa ya que explota el sello de tiempo de los objetos, que es el tiempo que van a estar presentes los objetos en la escena(en caso de los objetos visibles en ese instante) o el tiempo que se demorarán estos en entrar al volumen de visión(en caso de no verse en ese instante).

2.1 Características de la solución propuesta y requisitos a cumplir por las escenas para su correcto funcionamiento.

Antes de explicar el funcionamiento de la técnica propuesta, se hace necesario exponer los requisitos previos que deben cumplir las diferentes escenas en que se use, y con estas definir los tipos de escenas para las cuáles es factible y para cuáles no.

2.1.1Características de la solución propuesta.

1. La solución propuesta se basa en las características de la dinámica de la cámara y en la coherencia temporal de los objetos. La coherencia temporal es el tiempo que va a tener un objeto para entrar o salir del campo de visión de la cámara.
2. Se utilizan algoritmos independientes para tratar a los objetos en dependencia del movimiento de la cámara en la escena.
3. No se tienen en cuenta objetos dinámicos, ni objetos que cambien su estado de estáticos a dinámicos en tiempo de ejecución de la escena, debido a que la herramienta no tiene implementado una biblioteca física capaz de ofrecer datos de los objetos en movimiento o posibles a moverse.

2.1.2 Requisitos que deben cumplir las escenas para el correcto funcionamiento de la solución propuesta.

1. No pueden existir en la escena objetos estáticos que se vuelvan dinámicos en plena ejecución (objetos balas).
2. La escena no debe llevar a la cámara a realizar grandes cambios en su traslación y orientación. Esto conllevaría a la realización de demasiados cálculos constantemente y lejos de optimizar, esta técnica podría complejizar el proceso de selección de visibilidad.
3. No deben crearse o entrar objetos inesperadamente a la escena. O sea, todos los objetos de la escena se deben crear en la carga de esta. En caso de que pase esto, el objeto no se tendría en cuenta hasta que se haga un cambio de tendencia y se calculen nuevamente todos los tiempos de los objetos de la escena.
4. No se deben realizar transformaciones a los objetos que afecten su posición o tamaño. Por ejemplo, escalar un objeto en tiempo de ejecución, ya que este cambio no se tendrá en cuenta hasta que no se vuelva a calcular su tiempo de coherencia nuevamente.
5. No pueden existir en la escena objetos en movimiento debido a lo antes mencionado en el punto 3 de las características de la solución propuesta (ver epígrafe 2.2.1).

Después de analizar las características de la técnica propuesta y los requisitos que deben cumplir las escenas para su correcto funcionamiento se puede concluir que la técnica propuesta no debe aplicarse en

aquellos entornos donde existan objetos dinámicos o en los que se les realicen transformaciones en tiempo de ejecución a los objetos. Tampoco es recomendable su uso en escenas donde la cámara realice grandes cambios en su traslación u orientación, por ejemplo, en casos de juegos de acción en primera persona como Quake, Medalla de Honor etc.

Debido a que el entorno donde se use esta técnica se debe analizar en función de las características de la dinámica de la cámara, es más recomendable su uso en escenas donde la cámara mantenga una estabilidad en su movimiento y no predominen los objetos dinámicos, como en casos de paseos virtuales, entornos urbanos, entornos virtuales etc.

2.2 Soluciones propuestas.

Teniendo en cuenta las deficiencias del algoritmo mencionado en el epígrafe 1.5 se propone como solución una técnica que se base en el tiempo que un objeto puede encontrarse visible o no, sin restringir el movimiento de la cámara, manteniendo por separado los algoritmos de traslación y rotación.

2.3 Estructuras de datos utilizadas para manipular los objetos en la escena.

Para la implementación de los algoritmos explicados a continuación se hizo uso de una estructura de datos capaz de agilizar el proceso de visualización de los nodos de una escena. Esta estructura es el Grafo de Escena; se dio uso a esta estructura debido a que presenta una gran cantidad de ventajas en la organización de la escena. Sus características están explicadas en el epígrafe. 1.3.3.1.

Se usó la esfera como volumen de encierro de los objetos en la escena, debido a que es muy rápido encuestar con los planos del frustum con esta, y por consiguiente agiliza el proceso de visualización.

Se usó la cola con prioridad para guardar las estructuras que contienen a los nodos con sus respectivos tiempos, explicados en este capítulo.

2.4 Técnica para la selección de visibilidad en el volumen de visión (View Frustum Culling).

Para determinar si un objeto es visible en la escena en un frame determinado, el volumen de encierro del objeto pasa por varias técnicas de selección de visibilidad, la primera es View Frustum Culling, es la más importante ya que es aquí donde se determina si el objeto se encuentra dentro o fuera del volumen de visión, si el volumen de encierro del objeto está dentro se continúa con las demás técnicas, como por

ejemplo Oclusión Culling, Niveles de Detalles entre otras, y si está fuera ya podemos descartarlo y evitar todas las demás técnicas. A continuación se explicará más detalladamente la técnica View Frustum Culling.

Por cada volumen de encierro en la escena y en cada frame se encuesta con los 6 planos de la cámara (frustum 3d), como se explicó anteriormente se usará como volumen de encierro la esfera, si un volumen de encierro se encuentra total o parcialmente por delante de cada uno de los planos (en el mismo sentido de la normal de los planos), se puede decir que el objeto está dentro del área de visión. En la Fig.16 se hizo una proyección en un frustum 2d para mayor comprensión de lo ante expuesto, ver Fig.16.

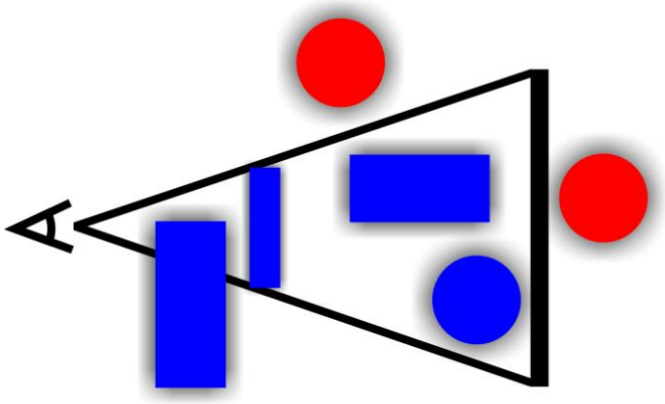


Fig. 16 Frustum 2D

Como se puede apreciar en esta figura los volúmenes de encierro en color rojo están fuera del campo de visión y por ende para estos volúmenes no se continúa con el proceso de visualización. Por el contrario a los verdes si se les continúa aplicando las demás técnicas explicadas en el capítulo anterior.

2.5 Método de detección de tendencia.

La tendencia de la cámara viene dada por el movimiento que esté realizando, es decir, si la cámara está trasladándose solamente, se detecta que hay una tendencia de traslación, después que es detectada esta tendencia se ve en que dirección se traslada, ya que esta tendencia, se divide en seis subtendencias (hacia la izquierda, hacia la derecha, hacia arriba, hacia abajo, hacia delante, hacia atrás), para dar solución al problema planteado no se tuvo en cuenta las tendencias en diagonal. En cada una de estas

subtenencias solo se tienen en cuenta los objetos que se encuentren dentro del volumen de visión de la cámara y aquellos que se encuentren fuera del plano por donde van a entrar al volumen de visión de la cámara si no existiera un cambio de tendencia los demás objetos se descartan. Este procedimiento se explicara en el próximo epígrafe más detalladamente.

Si la cámara está rotando solamente se detecta que hay una tendencia de rotación, igual que con la tendencia de traslación si se traslada, una vez detectada esta tendencia se determina en qué sentido está rotando ya que se divide en cuatro subtendencias (izquierda, derecha, arriba, abajo). En estas tendencias solo se tienen en cuenta los objetos que se encuentran dentro del volumen de visión de la cámara, y los que puedan entrar a este en un momento determinado si no existe un cambio de tendencia los demás objetos se descartan. Si la cámara está estática, los objetos se tratan con la última tendencia presente en la cámara.

2.6 Algoritmo Coherencia Temporal de Traslación "A. Traslación".

Primeramente hay que tener en cuenta para donde se traslada la cámara, ya que a los objetos que son visibles se les calcula el tiempo de coherencia por el plano donde van a salir, y a los no visibles por el plano que van a entrar al campo de visión de la cámara. Por ejemplo si la cámara se traslada hacia la izquierda, los objetos que se encuentren dentro de su campo de visión se les calcula el tiempo de coherencia por el plano derecho ya que es por donde van a salir, y a los objetos que no son visibles y se encuentren fuera del plano izquierdo se les calcula el tiempo de coherencia por ese plano. Los demás objetos de la escena se descartan ya que nunca van a llegar a ser visibles hasta que no halla un cambio de tendencia en el movimiento de la cámara, Fig. 18.

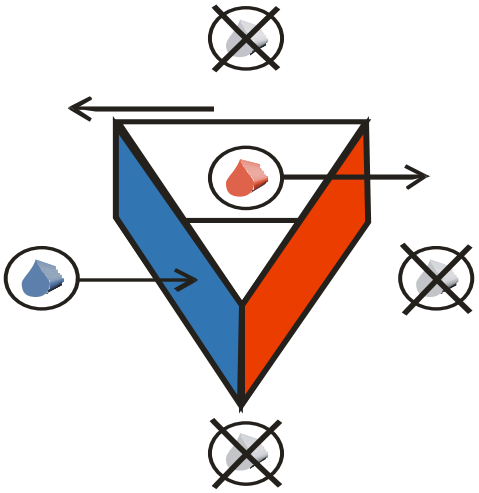


Fig. 18. Representación de la traslación hacia la izquierda.

Algoritmo en pasos:

Terminología usada:

Plano crítico: Plano por donde va a entrar o salir del campo de visión el objeto.
Punto crítico: Punto por donde va a entrar o salir del campo de visión el objeto.
RV: Radio del volumen envolvente.
CV: Centro del volumen envolvente.
TC: Tiempo de coherencia.
Vtc: Velocidad de traslación de la cámara.
Np: Normal del plano crítico.

***Pasos a seguir cada vez que se llame al método**:*

1. *Determinar el plano crítico de acuerdo a la tendencia de la cámara.*

2. *Determinar punto crítico.*
   *Punto crítico = CV + RV * Np*

3. *Determinar la distancia entre el punto crítico y el plano crítico.*

4. *Determinar el TC.*
   TC = Paso 3 / Vtc

2.7 Algoritmo Coherencia Temporal de Rotación “A. Rotación”.

Para determinar los objetos que entran o salen del Frustum cuando la cámara está rotando, solo se tendrá en cuenta la rotación de los objetos respecto a la cámara, condición que solamente cumplen los objetos estáticos; Los dinámicos suman su velocidad de traslación, lo que hace más complejo predecir por qué punto puede colisionar con el plano crítico; los objetos dinámicos en una escena se encuentran en menor proporción que los objetos estáticos, y no es conveniente complejizar el proceso por tan pocos objetos, y por consiguiente salen del análisis para este algoritmo.

Algoritmo de rotación en pasos.

Primeramente se detecta el tipo de tendencia de rotación que presenta la cámara, ya que a los objetos visibles se les calcula el tiempo de coherencia por el plano donde van a salir, y a los no visibles por el plano por donde van a entrar al volumen de visión de la cámara. Para determinar los objetos que van a llegar a ser visibles en un momento determinado si la cámara no cambia de tendencia, se comprueba con los planos por donde entran y salen los objetos del volumen de visión de la cámara; si los dos planos dejan fuera al objeto, en algún momento, éste llegará a ser visible en algún momento por lo que no se puede descartar. Los objetos que no sean visibles y que no cumplan esta última condición no se tienen en cuenta. Ejemplo de esto se muestra en la Fig.19. Esta figura muestra el caso específico de la rotación hacia arriba, donde los planos izquierdo y derecho de normales N1y N2 hacia adentro, dejando dentro de volumen de visión de la cámara al objeto de color rojo, este objeto va a ser enviado a la cola de los elementos visibles después que se le calcule su tiempo de coherencia con el plano de abajo que es por donde va a salir del plano de visión de la cámara. Mientras que estos mismos planos con su normal hacia afuera dejan fuera al objeto de color azul, este objeto en algún momento llegará a ser visible si se mantiene esta tendencia por lo que se le calcula su tiempo de coherencia con el plano de arriba que es por donde va a entrar al volumen de visión de la cámara, posteriormente se envía a la cola de los elementos no visibles. Los objetos de color negro, se descartan y no se tienen en cuenta ya que nunca van a llegar a ser visibles, si se mantiene esta tendencia.

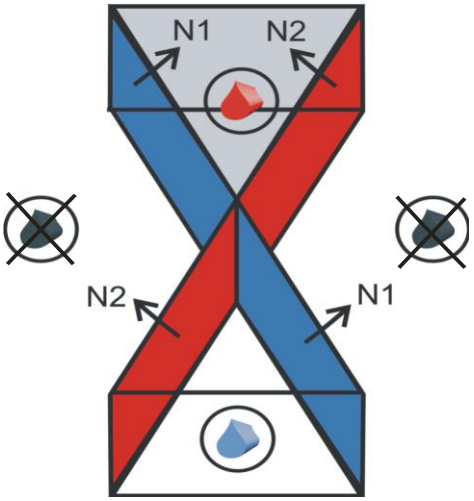


Fig. 19

Después se crea el plano de rotación (plano que se creará en tiempo de ejecución para calcular el tiempo de coherencia), para lo que se tomará de ejemplo el plano izquierdo de la cámara.

Para crear el plano izquierdo de la cámara hacen falta 3 puntos, como se muestra en la Fig.20.

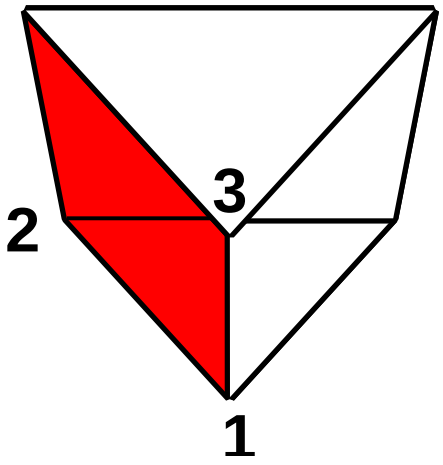


Fig. 20: Plano izquierdo de la cámara.

Para determinar el tiempo de coherencia en rotación, se hace necesario determinar el ángulo que se formaría entre el plano crítico y este mismo plano rotado al interceptar el volumen de encierro (plano de rotación).

Para la creación del plano de rotación se usarán los siguientes puntos, Fig. 21.

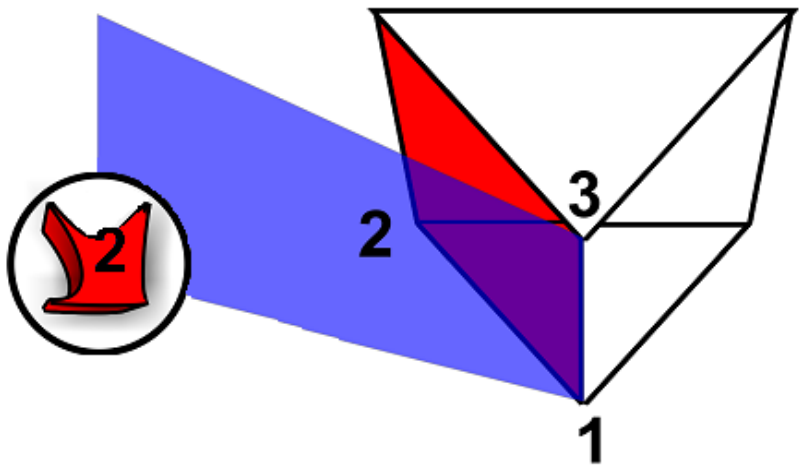


Fig. 21: Plano de rotación (azul).

1. Posición de la cámara.
2. Punto del centro del volumen de encierro.
3. En caso de rotación izquierda o derecha, el punto tres sería el vector arriba de la cámara y en caso de que fuera rotación arriba o abajo, sería el vector izquierda de la misma.

Algoritmo en pasos:

Terminología usada:

Plano crítico: plano por el cual el objeto puede entrar o salir del volumen de visión.
C: centro del volumen de encierro del objeto.
Cc: centro de la cámara
R: radio del volumen de encierro.
N: normal del plano crítico.
Nr: normal de plano de rotación
TC: tiempo de coherencia.

***Pasos a seguir cada vez que se llame al método**:*

1. *Crear el plano de rotación con el volumen de encierro correspondiente.*

2. *Determinar el ángulo comprendido entre el plano crítico y el plano de rotación*

    - *Para esto tenemos que crear el plano de rotación y con éste determinar la normal.*
    - *Si el objeto se encuentra fuera del volumen de visión el ángulo se formaría entre la normal del plano de rotación y la normal del plano crítico multiplicado por (-1), para invertirle el sentido.*
    - *Si el objeto está dentro del volumen de visión hay que determinar si el plano crítico es el izquierdo o el de abajo ya que si son estos hay que multiplicar por (-1) la normal del plano de rotación para poder formar correctamente el ángulo deseado.*
    - *Después que se tienen las dos normales correctamente, se calcula el ángulo entre las dos normales, y posteriormente para determinar el ángulo deseado, a 180 grados le restamos el ángulo entre las dos normales.*

Para una mejor comprensión de lo antes planteado se hizo una vista aérea de la Fig. 21 y se delimitaron los ángulos necesarios.

El ángulo 1 es el ángulo comprendido entre las dos normales.

El ángulo 2 es el ángulo de rotación de la cámara hasta interceptar el centro volumen de encierro del objeto.

Los ángulos que forman las normales con sus respectivos planos son de 90°.

Como la suma total de los ángulos de un cuadrilátero es 360 grados, el ángulo que se desea calcular es 180 grados – el ángulo 1, Fig. 22.

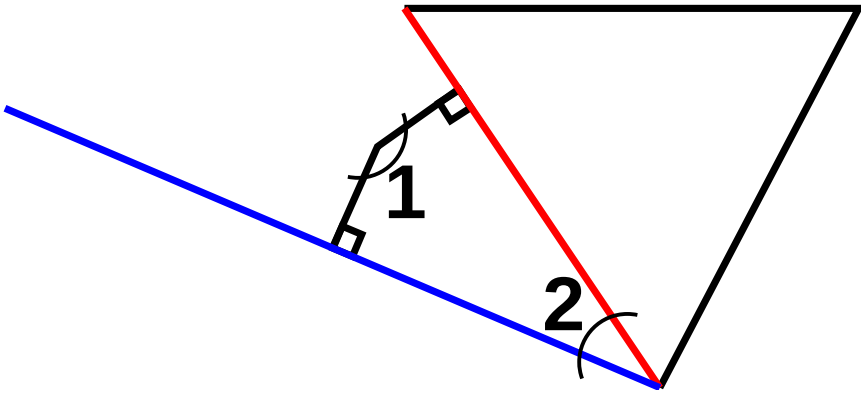


Fig. 22: Vista aérea de la figura 21.

Como el ángulo calculado fue utilizando un plano que se creó con los puntos antes mencionados, el ángulo de barrido de la cámara para interceptar el volumen de encierro no es éste, debido a que es el ángulo con respecto al centro, lo que significa que sería el ángulo para que la cámara interceptara al centro del volumen de encierro y no su extremo, Fig. 23.

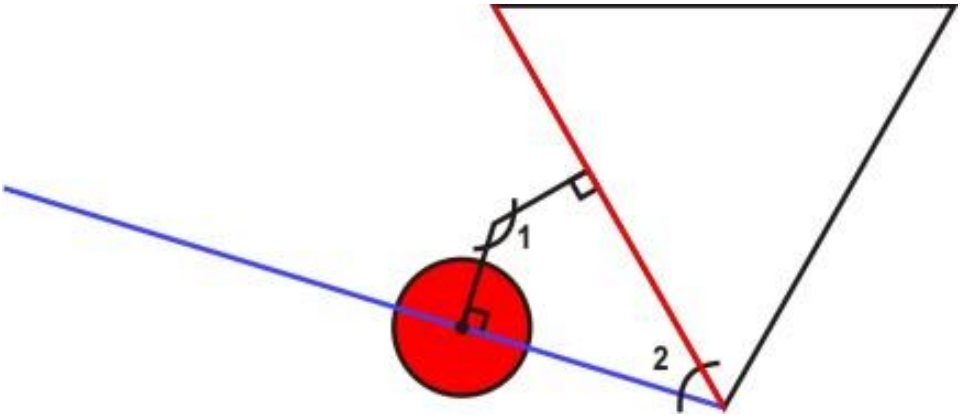


Fig. 23: Ángulo con respecto al centro del volumen de encierro.

3. *Determinar el ángulo entre el plano crítico y el plano rotado, pero restándole el ángulo excedente para obtener correctamente el ángulo que tendría que barrer la cámara para interceptar el volumen de encierro.*

Para obtener el ángulo excedente, hay que tener en cuenta el radio y la distancia del volumen de encierro con respecto a la posición de la cámara, Fig. 24.

Para determinar este ángulo se aplica la ley de los senos que plantea lo siguiente.

$\text{sen } \alpha = r / h$

donde:

r es el radio del volumen de encierro.

h es la distancia del centro del volumen de encierro a la posición de la cámara.

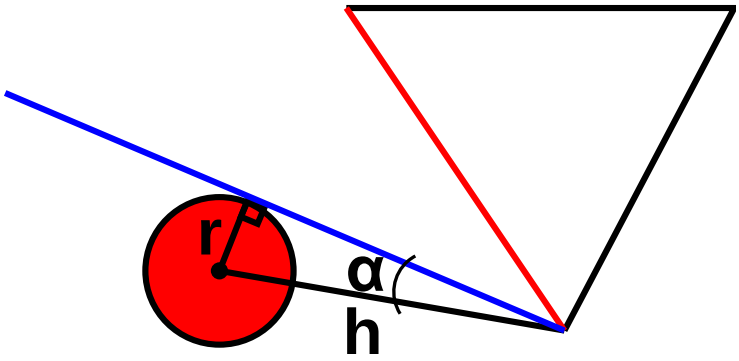


Fig. 24: α es el ángulo excedente.

4. *Determinar el tiempo que el plano crítico se tomaría para interceptar el volumen de encierro correspondiente.*
   Para esto se divide el ángulo obtenido en el paso 3 entre el tiempo de rotación de la cámara.

A continuación se plantea la fórmula para determinar el tiempo de coherencia en el caso del algoritmo A. Rotación. Para un mayor entendimiento ver la figura 25, donde α es el ángulo entre el vector centro del volumen envolvente –posición de la cámara y el plano donde el volumen envolvente va a hacer contacto con el plano crítico, 2 es el ángulo entre el plano crítico y el plano de rotación y ω es la velocidad angular de la cámara.

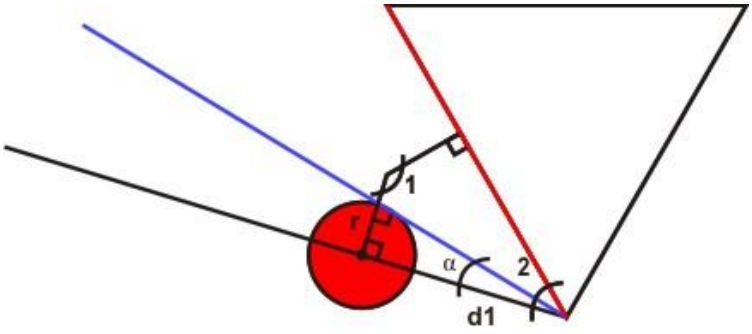


Fig. 25 Ángulo de barrido de la cámara para interceptar el objeto.

La ecuación para determinar el tiempo de coherencia basándose en la figura 25 quedaría de la siguiente forma.

$$TC = \frac{2 - \alpha}{\omega}$$

2.8 Aplicación de los algoritmos “A. Traslación” y “A. Rotación”.

Los algoritmos de A. Traslación y A. Rotación se aplican distintamente en dependencia de la tendencia que presente la cámara (ver epígrafe 2.7). En el estado inicial de la aplicación, se usará la técnica de View Frustum Culling, debido a que la cámara no presenta tendencia alguna. Cuando se detecta alguna tendencia se procede a realizar el algoritmo correspondiente a ésta. En caso de que la cámara cambie de tendencia, se espera un tiempo prudencial verificando la estabilidad del movimiento de la cámara, y si es estable se procede a activar la tendencia correspondiente, se vuelven a calcular los tiempos de coherencias con el algoritmo correspondiente y se llenan las colas. En los próximos frames se procede a decrementar los tiempos de coherencia de los objetos de acuerdo a la pertenencia en las colas. Si en algún momento se produce un cambio de tendencia, se limpian las colas donde se guardan los objetos y se comienza todo desde el inicio. A continuación se mostrará el algoritmo en pasos.

Algoritmo en pasos.

Estructuras de datos empleadas:

CP-OV: Cola con prioridad para los objetos visibles.
CP-ONV: Cola con prioridad para los objetos no visibles

Aclaraciones:

- *En el primer frame las colas están vacías.*
- *Las colas serán de estructuras de datos donde se almacenarán el tiempo de coherencia, un decrementador de tiempo y el nodo.*

Pasos a seguir por frame.

1. *Cuando la cámara cambie de tendencia se espera un tiempo, de acuerdo a como se mantuvo el movimiento de la cámara en ese intervalo se activa la tendencia correspondiente.*

2. *Después que se active la tendencia, se calcula el tiempo de coherencia a los objetos con el algoritmo correspondiente de acuerdo a su movimiento, y se van insertando en sus respectivas colas.*

3. *En el próximo frame se decrementan los tiempos de los objetos visibles y del primer objeto de la CP_ONV.*

4. *Si la tendencia de la cámara es traslación, se aplica el algoritmo de A. Traslación correspondiente.*

   Si la tendencia de la cámara es rotación, se le aplica el algoritmo de A. Rotación correspondiente.

5. *Si en algún momento hay un cambio de tendencia, se limpian las colas y se hace todo desde el paso 1.*

Conclusiones.

En este capítulo se han expuesto las soluciones técnicas para resolver el problema planteado. Se definieron los algoritmos a implementar: A. Traslación, A. Rotación y la aplicación que muestra las potencialidades de los dos algoritmos citados anteriormente, y que será la solución a implementar para determinar la selección de visibilidad de los objetos en la escena.

Capítulo 3

Resultados

Introducción.

En el presente capítulo se expone el caso de prueba realizado y se muestran los resultados del mismo después de aplicado a la solución propuesta y a la técnica de selección de visibilidad View Frustum Culling, (esta es la técnica más comúnmente utilizada por los motores gráficos para seleccionar los posibles objetos a visualizar en la escena). También se llegará a conclusiones dependiendo el rendimiento de las técnicas sometidas a pruebas.

3.1 Descripción del caso de prueba.

El caso de prueba creado para ser aplicado a la técnica de selección de visibilidad View Frustum Culling y a la técnica propuesta, se basa en las características de la escena (Debido a que la técnica que se propone depende de las características que deben tener las escenas explicadas en el epígrafe 2.2.2) y en la variación de la cantidad de objetos existentes en esta y su visibilidad.

Este caso de prueba se le realiza a las dos técnicas en igualdad de condiciones con el objetivo de ver el rendimiento de estas en cuanto a cantidad de ciclos de render que se le realicen a la escena en un segundo o lo que es lo mismo, frames por segundo (fps).

Consiste en la creación de una escena con las características óptimas para el correcto funcionamiento de la técnica propuesta, las pruebas se realizarán con esferas de radio 5 y 5 * 5 en la creación de los anillos de paralelo y meridiano. A esta escena se le varía la cantidad de objetos que va a contener, y se evalúa la cantidad y variación de fps dependiendo de la cantidad de objetos visibles en cada momento.

3.2 Resultados de los casos de pruebas.

En este epígrafe se muestran las gráficas y las conclusiones de los resultados de la aplicación del caso de prueba expuesto en el epígrafe anterior a las técnicas View Frustum Culling y Frustum Culling + Coherencia Temporal.

La tabla que se muestra a continuación muestra las prestaciones de la PC donde se realizaron las pruebas a las dos técnicas. De visualización.

Tipo de ordenador	Portatil
Marca	Haier
Modelo	H53
Procesador	Intel(R) Core(TM) Duo CPU 1.86 Ghz
Memoria RAM	1 Gb.
Chipset	ATI Radeon Xpress 200M, 128 MB
Sistema operativo	Microsoft Windows XP Professional Version 2002 Service Pack 3.

3.2.1 Representación gráfica del caso de prueba para 5 objetos.

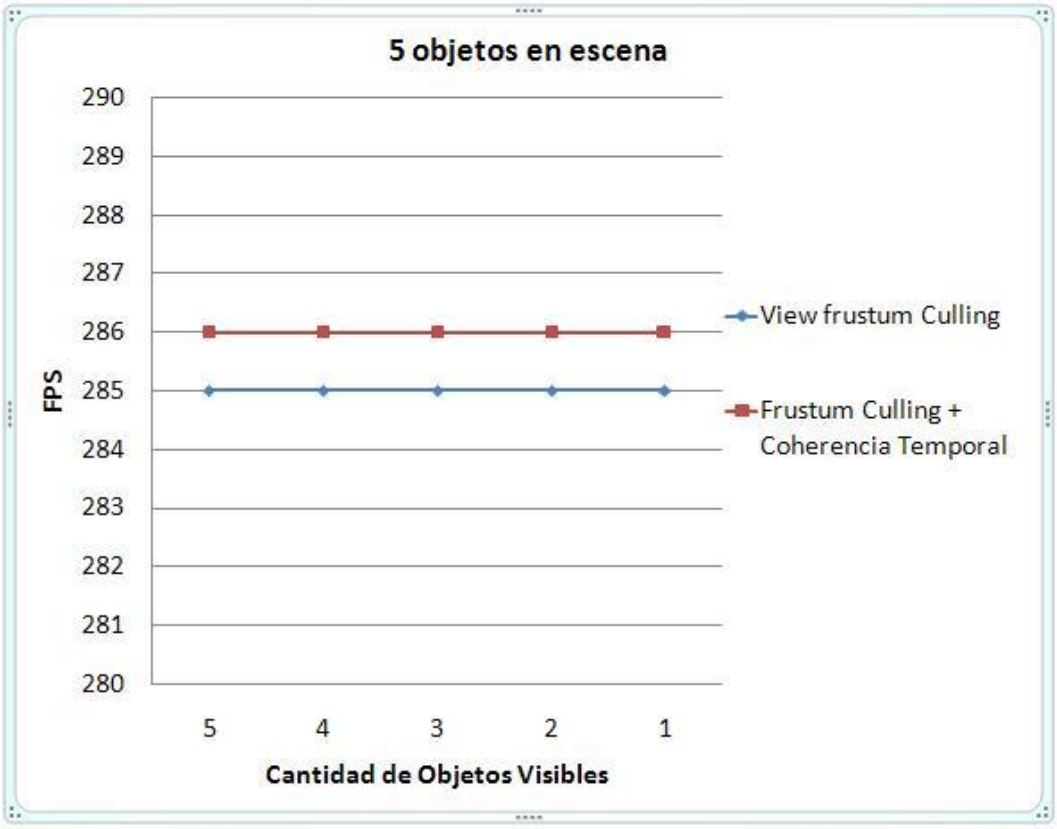


Fig. 26: Representación gráfica del caso de prueba número 1.

Como se puede apreciar en la gráfica anterior, cuando existen pocos objetos en la escena, la técnica de Frustum Culling + Coherencia Temporal mantiene el mismo rendimiento que la de View Frustum Culling. Esto se debe a que la última técnica mencionada debe encuestar menor cantidad de objetos e iguala en tiempo de procesamiento a la primera mencionada.

3.2.2 Representación gráfica del caso de prueba para 50 objetos.

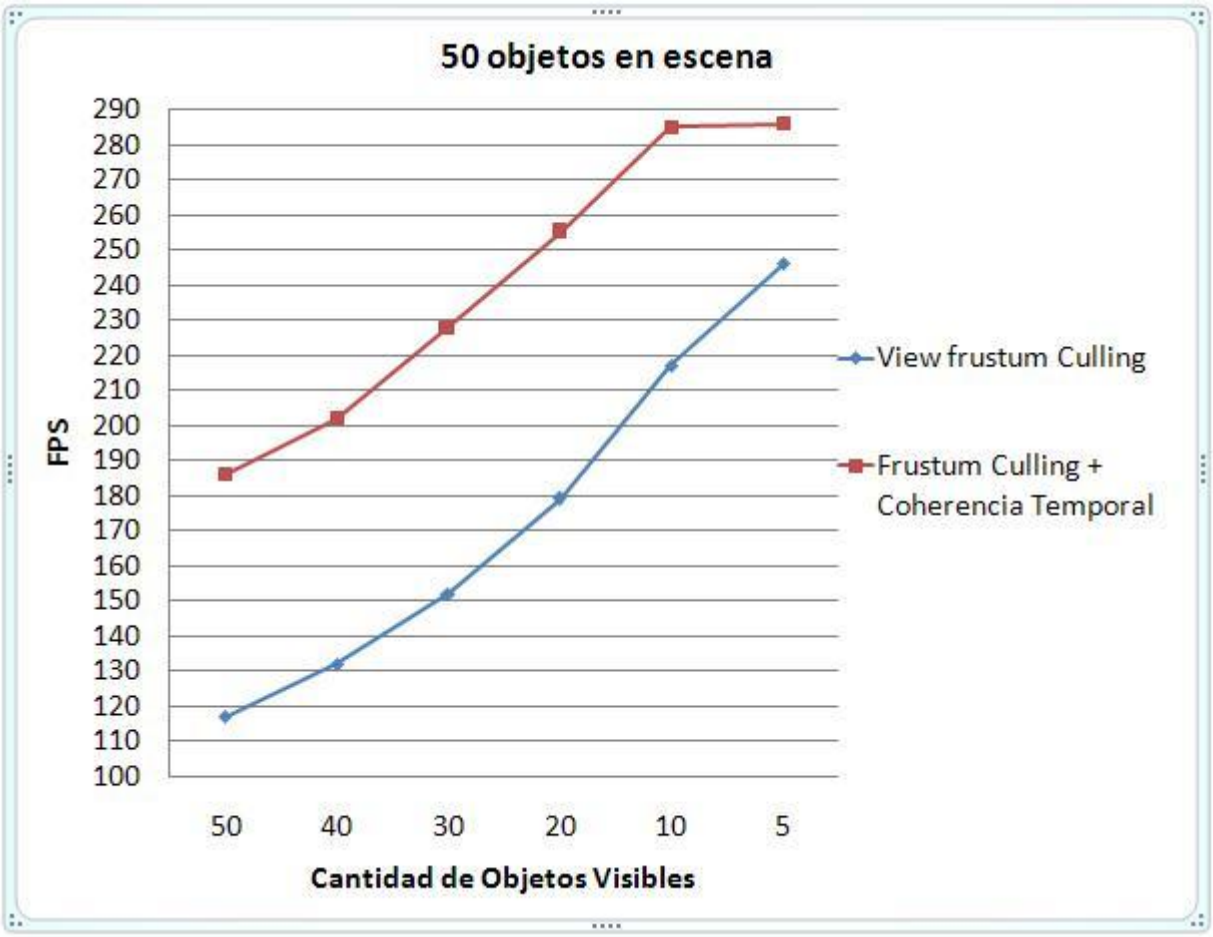


Fig. 27: Representación gráfica para el caso de prueba número 2.

En la gráfica anterior se puede apreciar como con cincuenta objetos visibles en la escena existe una diferencia notable en cuanto a la cantidad de fps. Esto se debe a que el la técnica View Frustum Culling tiene que realizar mayor cantidad de encuestas por frames, mientras que la de Frustum Culling + Coherencia Temporal, solo tiene en cuenta a los objetos que entran o salen del volumen de visión de la cámara.

3.2.3 Representación gráfica del caso de estudio para 140 objetos.

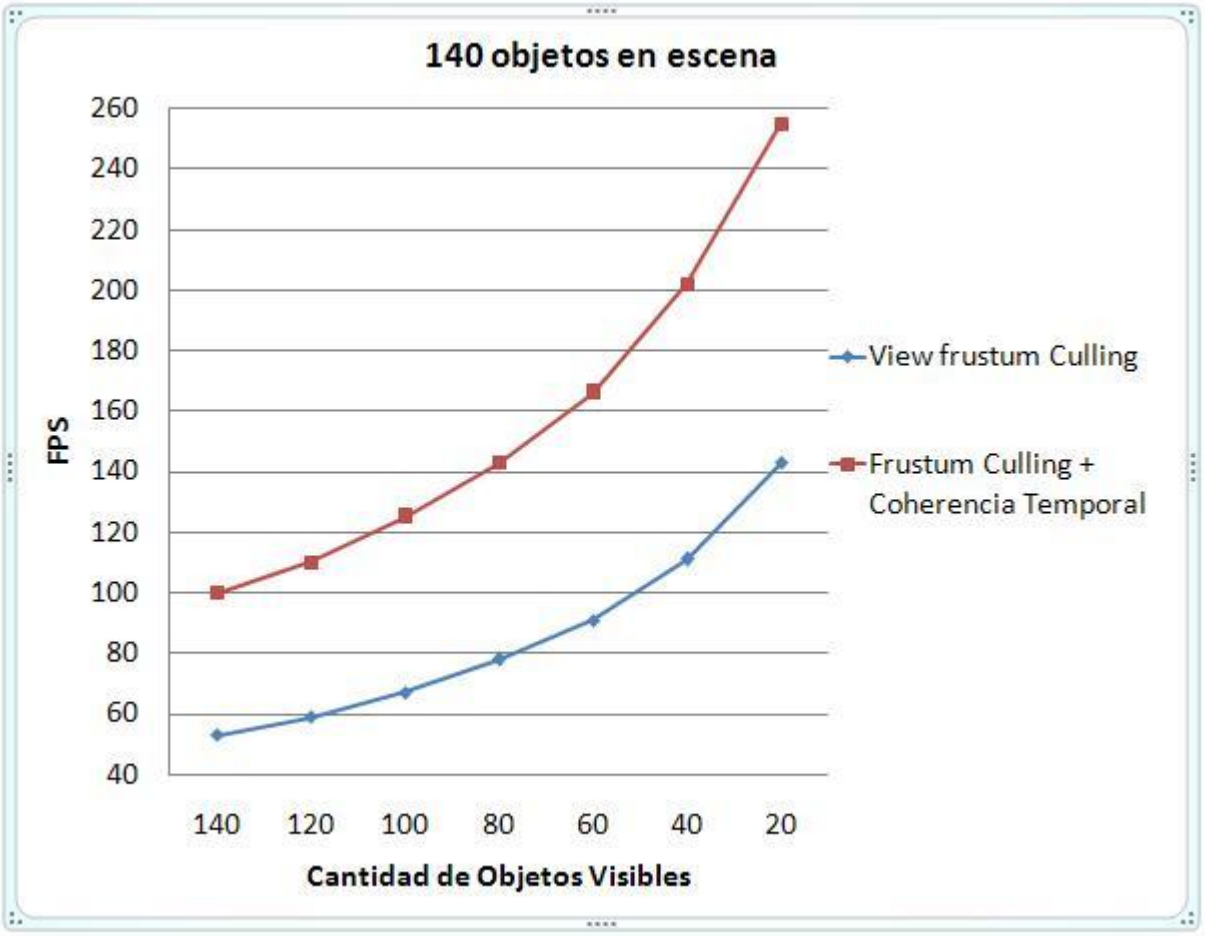


Fig. 27: Representación gráfica del caso de prueba número 3.

Al igual que la gráfica anterior, la creada con los datos resultantes de la aplicación del caso de prueba cuando existen 140 objetos en la escena nos demuestra que tanto la técnica View Frustum Culling como la View Frustum Culling + Coherencia Temporal tienden a disminuir su rendimiento con el aumento del número de objetos visibles en la escena, y que la técnica propuesta tiene un mayor rendimiento en cuanto a la cantidad de fps debido a que mientras existan mayor cantidad de objetos en la escena la otra técnica tiene que realizar mayor cantidad de encuestas disminuyendo su rendimiento.

3.2.4 Representación gráfica del caso de prueba para 240 objetos.

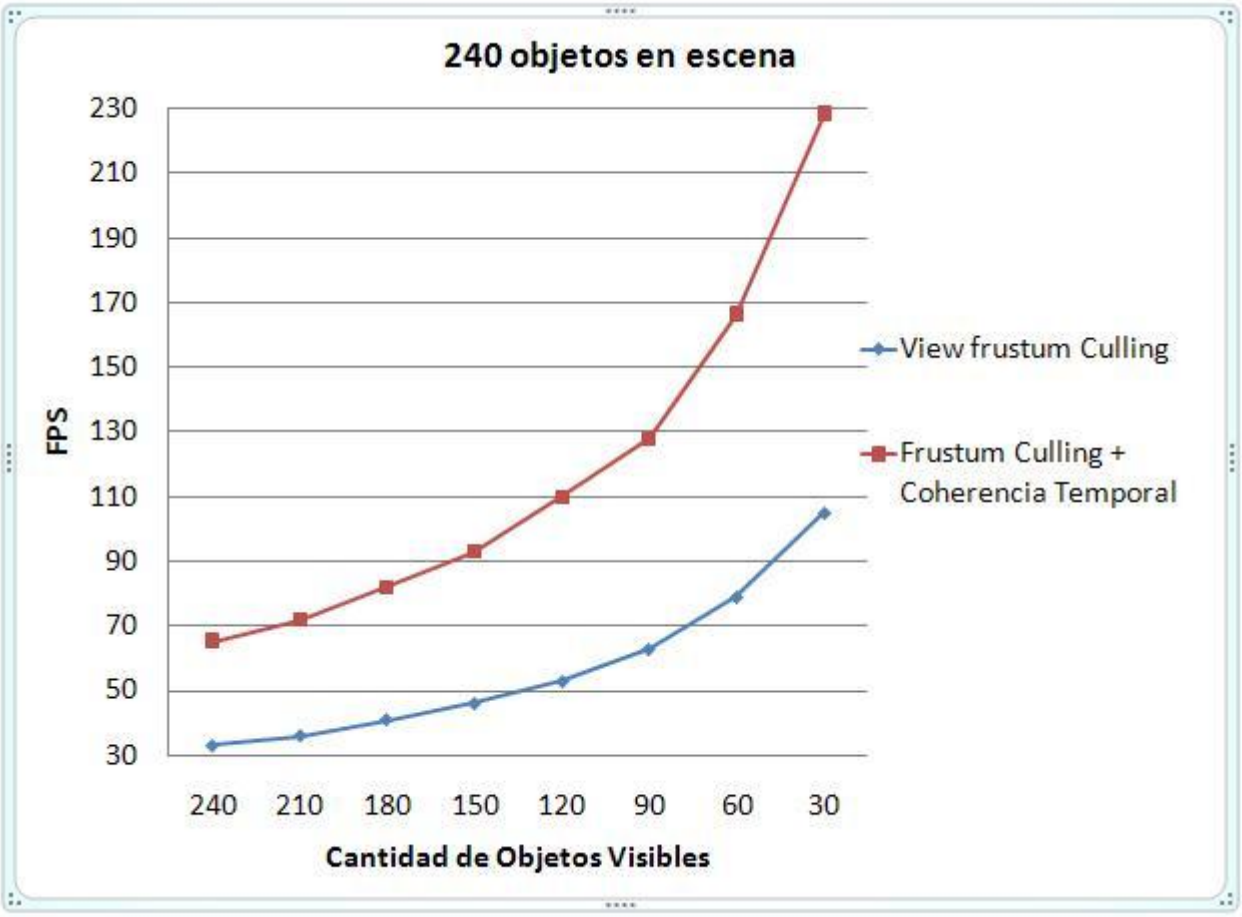


Fig. 28: Representación gráfica del caso de prueba número 4.

En los resultados de la gráfica para el caso de prueba cuando existen 240 objetos en la escena, se puede apreciar un mayor rendimiento en cuanto a la cantidad de fps por parte de la técnica View Frustum Culling + Coherencia Temporal. Hay que notar que el rendimiento de ambas va disminuyendo mientras va aumentando el número de objetos en la escena, ya que tienen que visualizar más objetos.

3.2.5 Representación gráfica del caso de prueba para 540 objetos.

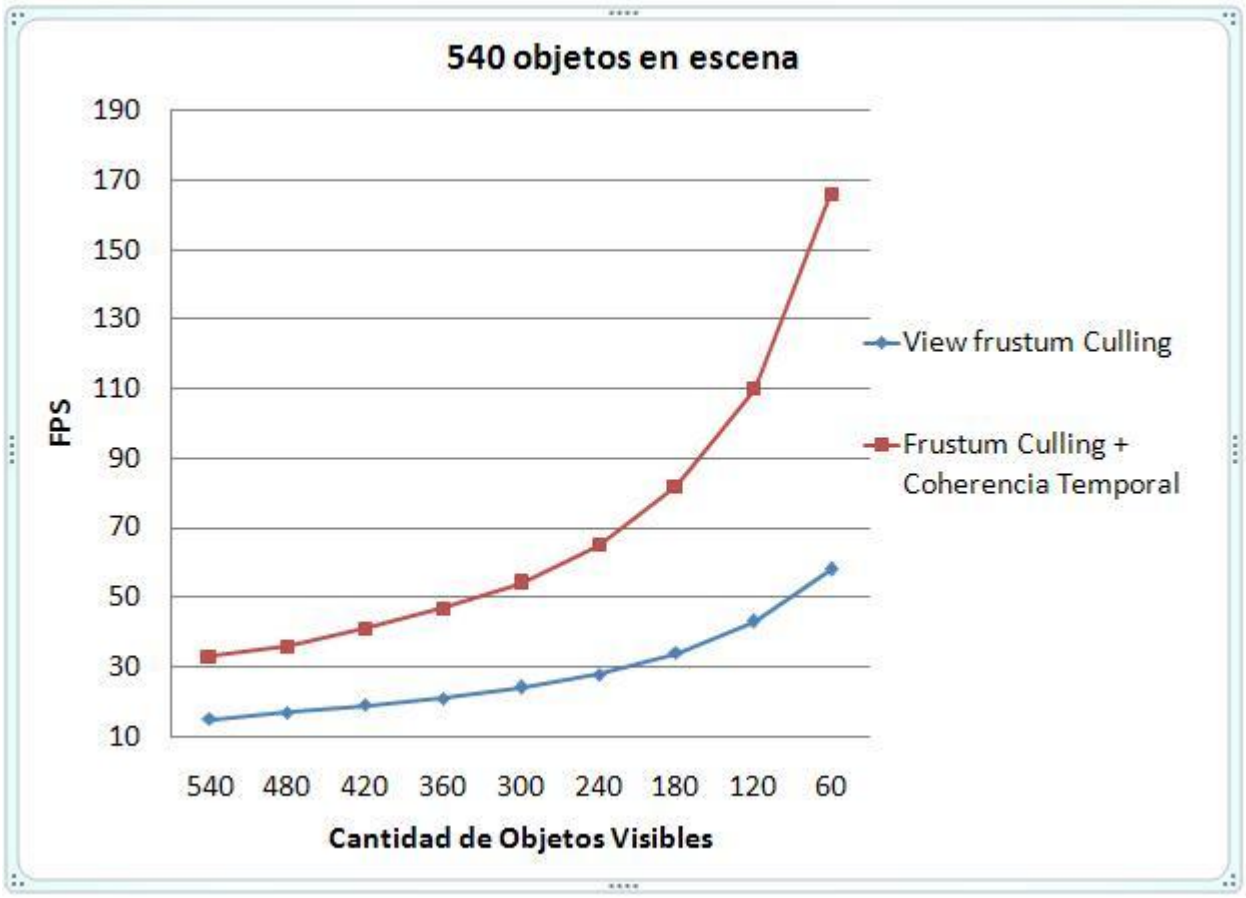


Fig. 29: Representación gráfica del caso de prueba número 5.

En los resultados de la gráfica para el caso de prueba cuando existen 540 objetos en la escena, se puede apreciar un mayor rendimiento en cuanto a la cantidad de fps por parte de la técnica View Frustum Culling + Coherencia Temporal. Se puede notar que el rendimiento de ambas va disminuyendo gradualmente con el aumento del número de objetos en la escena.

3.2.6 Representación gráfica del caso de prueba para 1000 objetos.

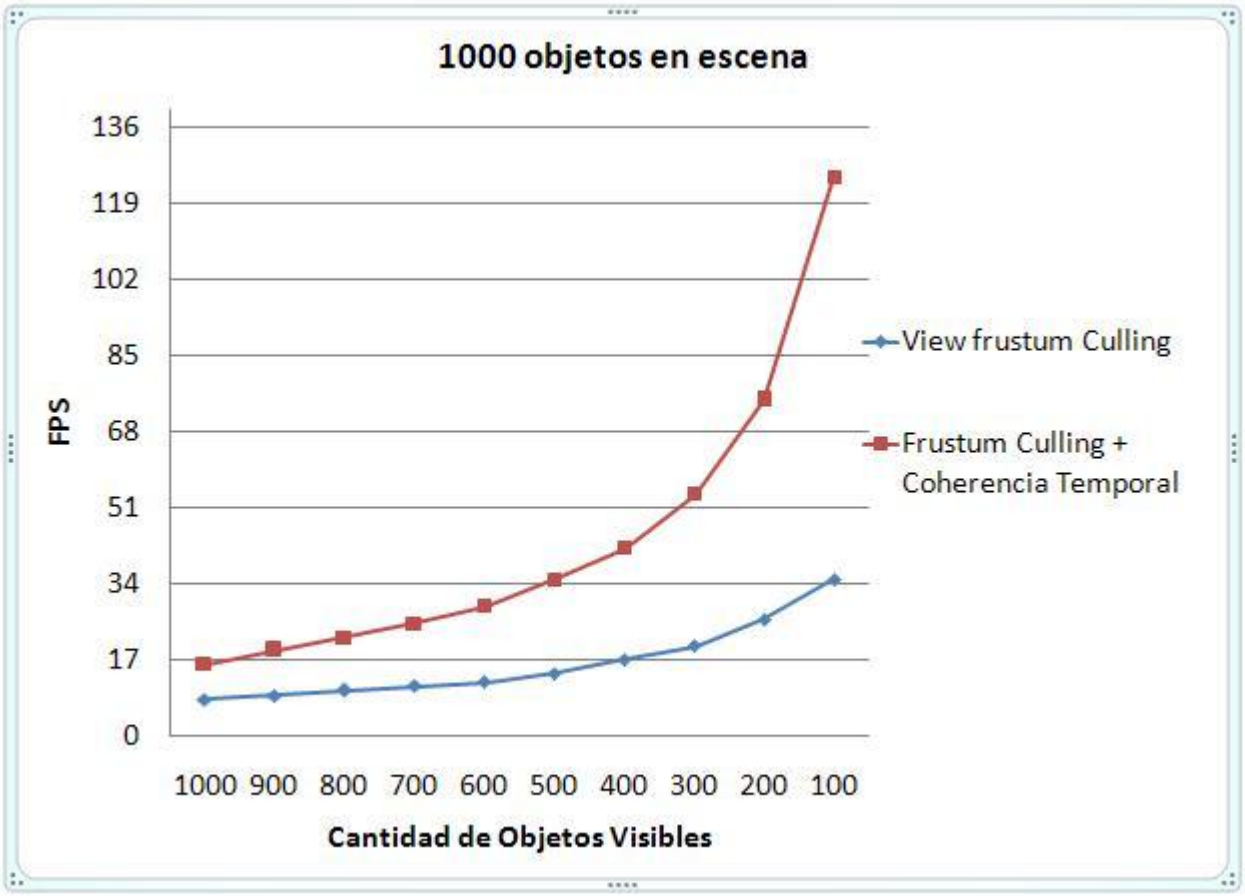


Fig. 30: Representación gráfica del caso de prueba número 6.

Al igual que en los casos de prueba anteriores, con la aplicación del caso de prueba con 1000 objetos en la escena, se puede apreciar un mayor rendimiento de la técnica View Frustum Culling + Coherencia Temporal y además se puede ver como esta técnica mantiene la misma cantidad de fps por cantidad de objetos visibles en la escena sin que la cantidad de objetos de esta afecte su rendimiento mientras que la técnica View Frustum Culling si disminuye su rendimiento debido a que tiene que encuestar mayor cantidad de objetos.

3.2.7 Representación gráfica del caso de prueba para 2000 objetos.

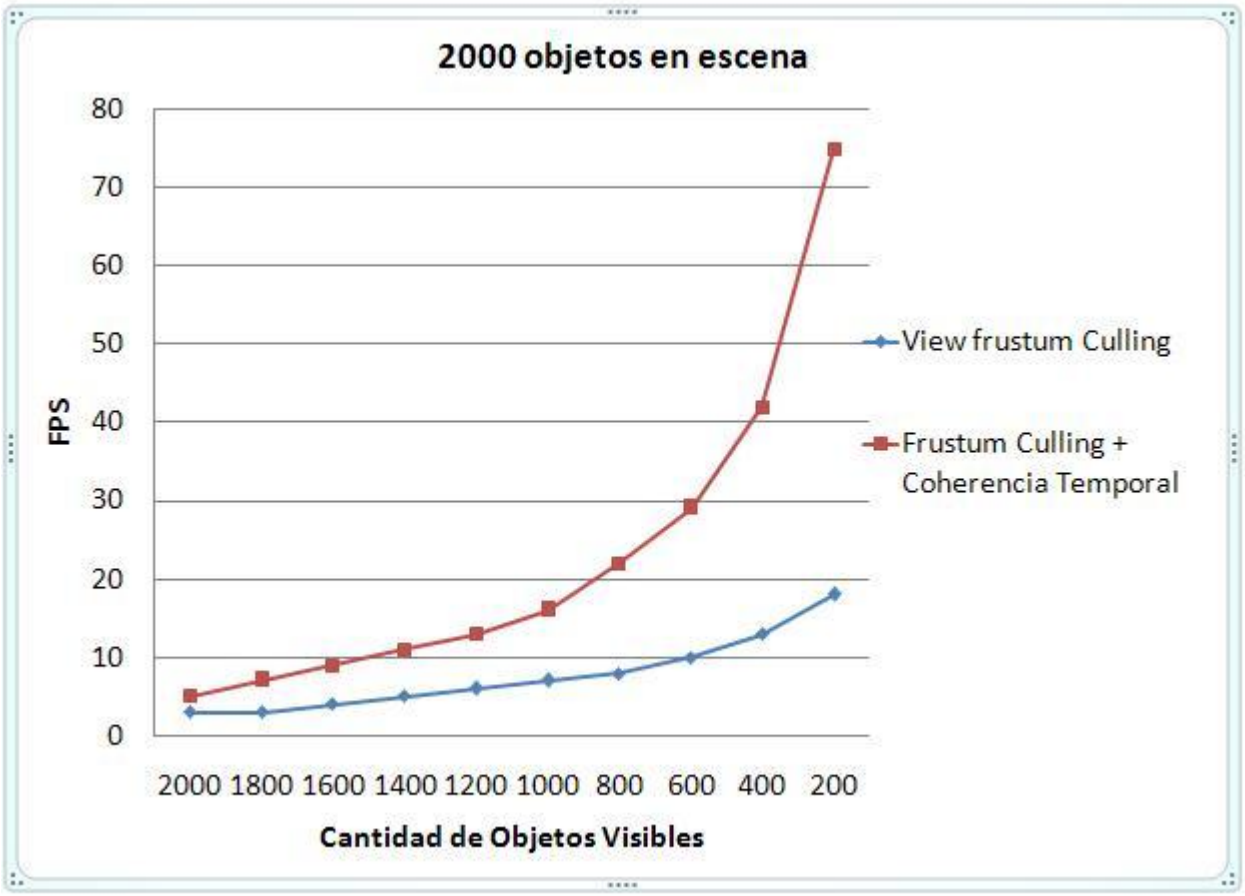


Fig. 31: Representación gráfica del caso de prueba número 7.

Al igual que en los casos de prueba anteriores, con la aplicación del caso de prueba con 2000 objetos en la escena, se puede apreciar un mayor rendimiento de la técnica View Frustum Culling + Coherencia Temporal, además se puede apreciar como con gran cantidad de objetos visibles en la escena el rendimiento de las dos técnicas llegan a ser similares

.

Conclusiones

Como conclusión de los resultados del caso de prueba expuesto en el epígrafe anterior se puede afirmar que mientras incremente la cantidad de objetos en la escena y disminuya el número de objetos visibles, la técnica View Frustum Culling + Coherencia temporal tiene un crecimiento mayor en cuanto a fps que la de View Frustum Culling. Este incremento va a ser mayor a medida que existan más objetos en escena debido a que por ejemplo una escena de 6000 objetos que se visualicen en pantalla 200, la técnica de View Frustum Culling tendría que encuestar a los 6000 objetos y solo pintar 200 mientras que con la técnica de Frustum Culling + Coherencia Temporal se trataría la escena como si solo tuviera los 200 objetos que se van a pintar.

Si se analiza detalladamente las gráficas anteriores se puede llegar a la conclusión de que la técnica View Frustum Culling + Coherencia Temporal solo calcula el tiempo de coherencia a todos los objetos en escena inicialmente, después de calculados los tiempos es indiferente la cantidad de objetos en escena debido a que esta técnica solo trata los objetos que expiren su tiempo de coherencia sin que importe la cantidad real en escena. Por ejemplo, en la gráfica 28, en el caso que se visualicen solo 120 objetos la aplicación correría a 110 fps y en la gráfica 27 si solo se visualizan 120 objetos la aplicación correrá a 110 fps, por lo que si se crea alguna escena no importa la cantidad de objetos que tenga, cuando se visualicen 120 objetos, la aplicación correrá a 110 fps. Esto no pasa así con la técnica de View Frustum Culling debido a que mientras mayor cantidad de objetos tenga la escena más encuestas tiene que hacer esta técnica para solo pintar una parte de la cantidad total de objetos lo que lleva a un descenso de los fps.

Conclusiones Generales.

Como resultado del presente trabajo de diploma:

- Se elaboró una técnica capaz de optimizar el proceso de selección de visibilidad basadas en las técnicas View Frustum Culling y Coherencia Temporal, apoyándose en las tendencias de movimiento de la cámara.
- Se probó la técnica propuesta en el motor de juego STK dando como resultado un incremento de los fps de la aplicación.

Debido a esto se considera que el trabajo cumplió con los objetivos planteados

Recomendaciones

Tomando como base la investigación realizada, quedan algunas recomendaciones que pueden servir de punto de partida para mejorar aún más el resultado obtenido en el mismo.

- Implementar las tendencias de traslaciones y rotaciones restantes.
- Implementar la técnica View Frustum Culling + Coherencia Temporal para objetos en movimiento.

Referencias bibliográficas

[1] Animación en Tiempo Real (Técnicas de Incremento de Velocidad) Disponible en: http://dis.um.es/grupos/sig/08BI/TiempoReal.pdf

[2]VARVANA, M. Dynamic Scene Occlusion Culling, 2003. [Disponible en: http://www.tml.tkk.fi/Opinnot/Tik-111.500/2003/paperit/VarvanaMyllarniemi.pdf]

[3]SUDARSKY, O. and C. GOTSMAN. Dynamic Scene Occlusion Culling, 1999. 5: 217-223.

[5] Optimized View Frustum Culling Algorithms for Bounding Boxes

[6] OMAR CORREA MADRIGAL, MARIEL NOGUEIRA COLLAZO Manipulación Eficiente de Objetos Dinámicos en Escenas Urbanas

[7] Ulf Assarsson and Tomas M¨oller Optimized View Frustum Culling Algorithms for Bounding Boxes. [Disponible en: http://www.google.com.cu/url?sa=t&source=web&cd=1&ved=0CBQQFjAA&url=http%3A%2F%2Fwww.cs.lth.se%2Fhome%2FTomas_Akenine_Moller%2Fpubs%2Fvfcullbox.pdf.gz&rct=j&q=Assarsson+y+Tomas+M%C3%B6ller+algorithm+coherency+rotation+and+traslation&ei=06MMTNGeE8T38Ab5s_WLBw&usg=AFQjCNFDeEQSnqEzF8PYD2yahL8na9bxUQ]

[8] JESÚS CELADA PÉREZ, Procesamiento y representación de la Información Geográfica. [Disponible en: http://www.conc2008login.ibge.gov.br/pesquisador/doc/geo/1.A.5_Procesamiento_y_representacion_de_la_IG.pdf]

[9] F.J. MELERO, P. CANO, J.C. TORRES, Visualización interactiva de SP-Octrees utilizando impostores. [Disponible en: http://lsi.ugr.es/~fjmelero/invest/mel04b.pdf]

[10] JOSE EMILIO LABRA GAYO, Representaciones gráficas y Mundos Virtuales infinitos en las Prácticas de Programación Lógica y Funcional. [Disponible en: http://www.di.uniovi.es/~labra/FTP/Papers/jenui03.pdf]

[11] CHACÓN MORENO, D.2000.Estudio y análisis de la teoría dela multirresolución en el modelado de sólidos.

[12] LIDIA ORTEGA ALVARADO, Navegación en Entornos Virtuales, Avances en Sistemas de Información espacial. [Disponible en: http://wwwdi.ujaen.es/~lidia/ASIE/nev.pdf]

[13] JOSÉ ALEJANDRO HIDALGO NAVARRO, Animación en Tiempo Real (Técnicas de Incremento de Velocidad). [Disponible en: http://dis.um.es/grupos/sig/08BI/TiempoReal.pdf]

Revistas

[4] Manipulación Eficiente de Objetos Dinámicos en Escenas Urbanas Disponible en: **[** http://biblioteca.reduc.edu.cu/biblioteca.virtual/cgi/CD-ROM/otros/UCIENCIA%202007%20(E)/ponencias/trvt/PDF/Manipulacion%20eficiente%20de%20objetos%20dinamicos-2052907776/Manipulacion%20eficiente%20de%20objetos%20dinamicos.pdf]

Glosario de términos

Algoritmo: Conjunto de reglas bien definidas para la solución de un problema en un número finito de pasos.

Estructuras de datos espaciales: Tipo de organización de datos diseñado para gestionar información espacial.

Frame: Cada una de las imágenes que componen una animación.

Frustum: Volumen de visión (en caso de frustum 3d) de la cámara en forma de una pirámide de base cuadrada creada desde la posición del observador, que está formada por los planos, izquierdo, derecho, cerca, lejos, arriba y abajo, en el caso que se esté en el espacio objeto y en caso de que se esté en el espacio imagen, está formado por los planos izquierdo, derecho, cerca y lejos.

Motor gráfico: Es la parte de un programa que controla, gestiona y actualiza los gráficos en tiempo real.

Pipeline gráfico: Método de renderización que consiste en aceptar primitivas tales como puntos, líneas y polígonos, y convertirlas en píxeles que serán dibujados en pantalla.

Videojuego: Es un programa informático, creado expresamente para divertir, basado en la interacción entre una persona y un aparato electrónico donde se ejecuta el videojuego.

View Frustum Culling: Técnica de selección de visibilidad basada en la creación de un volumen de observación en forma de una pirámide de base cuadrada creada desde la posición del observador, con el objetivo de descartar los objetos que se encuentran fuera del volumen de visualización de la cámara.