Frustum Culling

Visualización y Entornos Virtuales

1 Introducción

El objetivo de esta nueva tarea es la implementación de la técnica denominada *frustum culling* mediante la cual los objetos que la cámara no ve, no se dibujan (es decir, no se envia esta información al motor OpenGL).

1.1 Cámara y pirámide de visión

La cámara con proyección perspectiva define una pirámide truncada denominada *frustum*. Si el renderizado se realiza mediante esta cámara, se ve lo que se encuentra dentro de esta pirámide y no se ve lo que se encuentra fuera de ella. Ver Figura 1.

El *frustum* consta de seis planos: `left, right, bottom, top, near` y `far` que forman parte de los atributos de la cámara (en el sistema de coordenadas del mundo). Por tanto, cada vez que movemos la cámara hay que recalcular las ecuaciones de estos planos.

```
class Camera {
...
protected:
        Plane *m_fPlanes[MAX_CLIP_PLANES];
        // Frustum planes. A 6 elements of type plane.
        // Order: (l,r,b,t,n,f)
        // Note: normals point inside the frustum.
...
} camera;
```

Cuando se realiza el cambio de vista, la función `Camera::setViewTrfm()` llama a la función `updateFrustumPlanes()`, la cual se encarga de adaptar los parámetros de los seis planos del *frustum*.

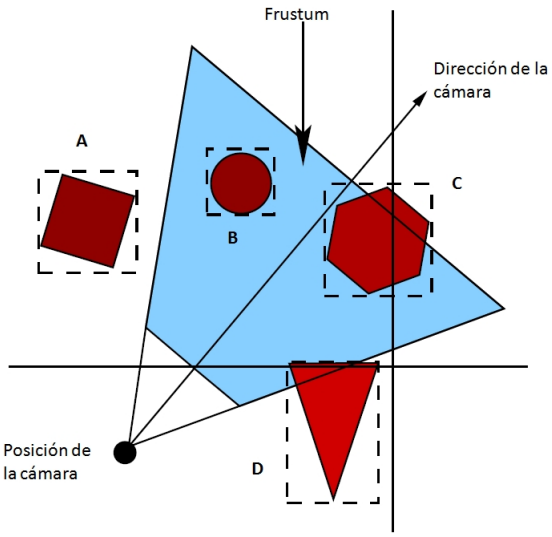

Figure 1: *Frustum Culling*

1.2 Implementación del *Frustum culling*

Para realiar el *Frustum culling* tenemos que implementar dos funciones:

- La función `Scene/node.cc`:

```
void Node::frustumCull(Camera *cam);
```

- La función `Camera/camera.cc`:

```
int Camera::checkFrustum(const BBox *theBBox, unsigned int
                         *planesBitM);
```

En la primera función (`Node::frustumCull`), dada una cámara (de la cual conocemos las ecuaciones del plano *frustum*), tenemos que verificar si los nodos del árbol de la escena se encuentran dentro o fuera del *frustum* y adaptar el atributo `m_isCulled` de la clase `Node`. Si este atributo es 1, el nodo (y el sub-árbol a partir de él) se encuentra fuera del *frustum* y no se renderiza. Si es 0, por el contrario, se visualiza el nodo.

Esta función llama a la función de la cámara, `Camera::checkFrustum(...)`, que devuelve uno de los siguientes tres valores:

- +1 si el `BBox` se encuentra fuera del *frustum*.
- 0 si el `BBox` corta al *frustum*.
- -1 si el `BBox` se encuentra dentro del *frustum*

siendo `BBox` el *bounding box* (`BBox`) de un nodo.

Por ejemplo, el objeto **A** de la Figura 1 se encuentra fuera del *frustum* (+1), el objeto **B** se encuentra dentro del *frustum* (-1) y los objetos **C** y **D** cortan al *frustum* (0).

El segundo parámetro de la función es `planesBitM:` Este parámetro no es estrictamente necesario y, si queréis, no tenéis que utilizarlo. Es una máscara en la cual el bit i-ésimo toma el valor 1 si el `BBox` se encuentra dentro del i-ésimo plano *frustum*.

Una vez implementadas estas dos funciones, solamente faltan dos cosas para calcular el *frustum culling*:

1. En la función `Display()` antes de dibujar la escena, `Render(theCamera)`, debe llamarse a la función `frustumCull(theCamera)`.

2. En la función `Node::draw()`, si el atributo `m_isCulled` del nodo es 1, no se dibuja el objeto.

2 A entregar

Implementación **bien documentada** en la que se explique detalladamente cómo se ha realizado.

3 Bibliografía

Gribb&Hartmann2001 Gribb, Gil, and Klaus Hartmann, *Fast Extraction of Viewing Frustum Planes from the World-View-Projection Matrix*, June 2001.

Dirección del Artículo

4 Anexo: Cálculo de los parámetros de los planos del frustum

A continuación presentamos las ecuaciones que calculan los planos del *frustum* de una cámara (Gribb&Hartmann2001).

Sea $\mathbf{V}$ la matriz que realiza la transformación de la vista y $\mathbf{P}$ la matriz de proyección perspectiva. La matriz $\mathbf{M}$ es la composición de estas dos transformaciones: $\mathbf{M} = \mathbf{PV}$.

$$\mathbf{M} = \begin{pmatrix} \mathbf{m}_{0,} \\ \mathbf{m}_{1,} \\ \mathbf{m}_{2,} \\ \mathbf{m}_{3,} \end{pmatrix}$$

Sean el punto $\mathbf{s} = (s_x, s_y, s_z, s_w)^T$ (donde $s_w = 1$), y su transformación $\mathbf{t} = (t_x, t_y, t_z, t_w)$:

$$\mathbf{t} = \mathbf{Ms} = \begin{pmatrix} \mathbf{m}_{0,} \cdot \mathbf{s} \\ \mathbf{m}_{1,} \cdot \mathbf{s} \\ \mathbf{m}_{2,} \cdot \mathbf{s} \\ \mathbf{m}_{3,} \cdot \mathbf{s} \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \\ t_z \\ t_w \end{pmatrix}$$

Después de la transformación, seguramente $t_w \neq 1$, debido a la proyección perspectiva. Por tanto, todas las coordenadas del punto $\mathbf{t}$ hay que dividirlas por el factor t_w. Como resultado, se tiene el punto $\mathbf{u} = (u_x, u_y, u_z, 1)$.
Si el punto original $\mathbf{s}$ se encuentra en la pirámide del *frustum*, las coordenadas$-1 \leq u_i \leq 1$ donde $i \in x, y, z$; es decir, será un punto $\mathbf{u}$ que se encuentra dentro del cubo unitario.
A partir de estas ecuaciones, podemos obtener las ecuaciones de los planos del *frustum*.

Por ejemplo, analizaremos la parte derecha del plano izquierdo del cubo,$-1 \leq u_x$:

$$-1 \leq u_x \Leftrightarrow -1 \leq \frac{t_x}{t_w} \Leftrightarrow t_x + t_w \geq 0 \Leftrightarrow (\mathbf{m}_{0,} \cdot \mathbf{s}) + (\mathbf{m}_{3,} \cdot \mathbf{s}) \geq 0 \Leftrightarrow (\mathbf{m}_{0,} + \mathbf{m}_{3,}) \cdot \mathbf{s} \geq 0$$

Las últimas ecuaciones, $(\mathbf{m}_{3,} + \mathbf{m}_{0,}) \cdot \mathbf{s} \geq 0$, representan el plano o semi-plano izquierdo del *frustum*.

Hay que pasar el plano izquierdo del cubo a las coordenadas del mundo. Además, nótese que $s_w = 1$, y por tanto, es la ecuación de un plano delantero. Ya que la normal del plano mira hacia fuera del *frustum*, hay que cambiar el signo de la ecuación del plano.

Finalmente:

$$\begin{aligned} -(\mathbf{m}_{3,} + \mathbf{m}_{0,}) \cdot (s_x, s_y, s_z, 1)^T &= 0 \text{ [left]} \\ -(\mathbf{m}_{3,} - \mathbf{m}_{0,}) \cdot (s_x, s_y, s_z, 1)^T &= 0 \text{ [right]} \end{aligned}$$

$$
\begin{aligned}
-(\mathbf{m}_{3,} + \mathbf{m}_{1,}) \cdot (s_x, s_y, s_z, 1)^T &= 0 \text{ [bottom]} \\
-(\mathbf{m}_{3,} - \mathbf{m}_{1,}) \cdot (s_x, s_y, s_z, 1)^T &= 0 \text{ [top]} \\
-(\mathbf{m}_{3,} + \mathbf{m}_{2,}) \cdot (s_x, s_y, s_z, 1)^T &= 0 \text{ [near]} \\
-(\mathbf{m}_{3,} - \mathbf{m}_{2,}) \cdot (s_x, s_y, s_z, 1)^T &= 0 \text{ [far]}
\end{aligned}
$$