Capítulo IV
Visualización de BSP-OctTrees

4.1 Introducción

Aunque la visualización de un modelo geométrico en 2D es prácticamente trivial, la visualización de una representación geométrica en 3D implica principalmente los siguientes problemas:

- La eliminación de partes ocultas, es decir, evitar que las secciones del objeto a visualizar que deben verse de acuerdo al punto de observación queden obstruidas por las secciones del objeto que no deben verse (por ejemplo, las partes posteriores de un objeto deben quedar ocultas mientras que las partes frontales deben quedar visibles).
- Cálculo de las coordenadas correctas en pantalla, es decir, a partir de las coordenadas reales del objeto 3D obtener las coordenadas en las que se visualizará en una pantalla 2D. En otras palabras, el objeto debe trasladarse y proyectarse en la pantalla.
- La utilización de algunos efectos visuales que hagan la visualización más realista, como puede ser la aplicación de texturas o la utilización de diferentes técnicas de iluminación.

Aunque el modelo propuesto hereda la interesante propiedad de los OctTrees clásicos de poder ser visualizados con eliminación correcta de partes ocultas gracias a su estructura regular compuesta por cubos que no se intersectan entre sí, además de que el mecanismo para elegir una enumeración correcta de los nodos del árbol ya fue explicado en **[Sección2.4.3]**, aún así esto no es suficiente para determinar el orden de pintado de las caras internas de cualquier nodo que no sea clásico. Sin embargo, en el modelo propuesto

esto implica únicamente conocer cómo debe ser visualizado el árbol BSP contenido en los nodos BSP, lo cual es un procedimiento bastante sencillo, además de también requerir un procedimiento para reconstruir la frontera de un árbol BSP, que también resulta simple.

Respecto al problema de calcular las coordenadas correctas en 2D, es decir, la traslación y proyección de un objeto en pantalla, y respecto a la utilización de cualquier efecto visual, existen infinidad de mecanismos descritos en la literatura, algunos de ellos descritos en [Argüelles00], por lo que el presente trabajo no detalla mucho al respecto.

4.2 Visualización de árboles BSP

El procedimiento es muy sencillo. Para cada nodo en un árbol BSP debe determinarse su posición relativa respecto al observador, utilizando la ecuación del plano de soporte contenida dentro del nodo. Si el observador está enfrente del nodo, el hijo izquierdo (que contiene el subconjunto de caras $C(H^-)$ descrito en **[Sección3.2]**) debe ser visualizado recursivamente primero, luego el nodo mismo, y por último el hijo derecho (que contiene el subconjunto de caras $C(H^+)$ descrito en **[Sección3.2]**). Por otro lado, si el observador está atrás del nodo, el hijo derecho debe ser visualizado recursivamente primero, luego el nodo mismo, y finalmente el hijo izquierdo.

4.2.1 Reconstrucción al modelo de fronteras

Para poder visualizar un nodo de un árbol BSP, no sólo es necesario poder determinar el orden correcto de visualización, descrito en **[Sección4.2]**, sino que, como ya se ha mencionado en repetidas ocasiones, también es necesario obtener las fronteras del nodo. Para obtener las fronteras de un árbol BSP, existe un método muy simple descrito en [Thibault87], [Naylor90]. Sin embargo, para entender dicho método debe explicarse primero el concepto de la inserción de un polígono en un árbol BSP. Esto consiste en particionar dicho polígono por medio de la ecuación de soporte contenida en el nodo raíz de dicho árbol, e insertar los dos fragmentos resultantes en los hijos del nodo, es decir, el fragmento que quedó en el semiespacio negativo del plano de soporte se inserta en el hijo izquierdo, mientras que el fragmento del semiespacio positivo se inserta en el derecho. Esta operación debe repetirse recursivamente hasta llegar a algún nodo hoja (celda), en cuyo caso no se realiza partición alguna, sino que se toma como salida del procedimiento el fragmento completo. En otras palabras, la inserción de un polígono en un árbol BSP produce la partición del polígono en *n* fragmentos, donde *n* es a lo más el número celdas del árbol BSP. Dichos fragmentos pueden clasificarse en dos subconjuntos: los fragmentos *in*, que son los obtenidos en los nodos hoja izquierdos (las celdas *in* **[Sección2.3.1]**), y los fragmentos *out*, que son los obtenidos en los nodos hoja derechos (las celdas *out* **[Sección2.3.1]**). Desde luego, un fragmento insertado en un nodo interno podría no ser particionado por el plano de soporte del mismo, en cuyo caso la inserción recursiva continúa únicamente a través uno de los dos hijos del nodo interno, dependiendo de la posición del fragmento respecto al plano.

Ahora bien, para obtener las fronteras de un árbol BSP, deben realizarse los siguientes cuatro pasos para cada todos y cada uno de los nodos del árbol:

- Se obtiene primeramente una representación poligonal de la ecuación del plano de soporte del nodo, es decir, del plano que intersecta la región R del nodo. Esto puede lograrse de dos maneras:
  - ◊ En el modelo general, se utiliza un cubo centrado en el origen cuyo tamaño es lo suficientemente grande para contener completamente al objeto poliédrico. Así pues, se procede a proyectar uno de los lados de este cubo sobre el plano de soporte cuya representación poligonal desea obtenerse. El lado elegido es aquél cuya proporción aritmética de su área con la del área de su proyección está más cercana a la unidad. Para lograrlo, se elige el lado cuya normal tiene el menor ángulo respecto a la normal del plano de soporte.
  - ◊ En el caso de un árbol BSP contenido dentro de un nodo BSP, puede utilizarse el octante mismo del nodo BSP y directamente intersectar la ecuación del plano de soporte del cada nodo del árbol BSP contra dicho octante, de manera semejante a como se obtienen las fronteras de un nodo Cara en el modelo Extendido [Argüelles00].
- Dada esta representación poligonal, se inserta en el nodo raíz del árbol BSP original, utilizando el procedimiento descrito anteriormente, pero siguiendo únicamente la ruta hacia el nodo actual, es decir, descartando los fragmentos que serían insertados en nodos hijos que no forman parte de dicha ruta. El proceso de inserción se detiene al llegar al nodo actual, siendo que en **[Naylor90]** se denomina a este último fragmento

como el subplano de soporte del nodo, que no es otra cosa que la representación poligonal de un plano que matemáticamente es infinito. Claro está que también es posible seguir el proceso inverso: insertar la representación poligonal obtenida en el primer paso en el nodo actual y seguir la ruta hacia el nodo raíz utilizando algún tipo de apuntador de los nodos hijo al nodo padre. Como esta última ruta es desde luego única, se evita de esta manera el problema de no saber qué ruta seguir desde el nodo raíz hasta el nodo actual, es decir, decidir si se inserta en el hijo izquierdo o derecho de cada nivel.

- Dado el subplano de soporte del nodo, se procede a insertar dicho polígono en la rama izquierda del nodo actual (H^-) utilizando el procedimiento normal, para posteriormente descartar los fragmentos *out* producidos e insertar los fragmentos *in* restantes en la rama derecha (H^+). Los fragmentos *out* resultantes de esta segunda inserción describen las caras de la frontera del sólido cuya ecuación de soporte es **idéntica** a la ecuación de soporte contenida en el nodo analizado.
- Por último, simplemente se realiza la operación inversa al tercer paso, es decir, el mismo subplano de soporte del nodo obtenido en el segundo paso debe insertarse en la rama derecha del nodo actual (H^+), los fragmentos *out* se descartan, y los fragmentos *in* deben insertarse en la rama izquierda (H^-). Los fragmentos *out* resultantes de esta segunda inserción describen las caras de la frontera del sólido cuya ecuación de soporte es **inversa** a la ecuación de soporte contenida en el nodo analizado.

Es importante comentar que dado que todos los fragmentos obtenidos dentro de un nodo de un árbol BSP son coplanares, el orden de visualización entre ellos es irrelevante.

4.3 Visualización de BSP-OctTrees

El algoritmo requerido recorre el árbol de la misma forma descrita para los OctTrees clásicos **[Sección2.4.3]**, dependiendo de la posición relativa del observador. Cuando se llega a un nodo BSP, puede ser visualizado simplemente recorriendo su árbol BSP en la forma descrita en **[Sección4.2]**. Los nodos Blancos no necesitan ser visualizados ya que son nodos externos al objeto, y los nodos Negros tampoco necesitan ser visualizados ya que éstos son internos al objeto y siempre quedan totalmente cubiertos por nodos BSP.

Respecto a la posición relativa del observador, ésta puede calcularse una única vez antes de iniciar el proceso de visualización de la estructura, utilizando los ángulos de rotación respecto a los ejes *x*, *y*, *z* que se deseen, mediante la multiplicación de las coordenadas de la posición inicial del observador por la siguiente matriz [Argüelles00]:

$$\left[\begin{array}{c|c|c} \cos(y)\cos(z) & \cos(x)\,\mathrm{sen}(z)+\mathrm{sen}(x)\,\mathrm{sen}(y)\cos(z) & \mathrm{sen}(x)\,\mathrm{sen}(z)-\cos(x)\,\mathrm{sen}(y)\cos(z) \\ -\cos(y)\,\mathrm{sen}(z) & \cos(x)\cos(z)-\mathrm{sen}(x)\,\mathrm{sen}(y)\,\mathrm{sen}(z) & \mathrm{sen}(x)\cos(z)+\cos(x)\,\mathrm{sen}(y)\,\mathrm{sen}(z) \\ \mathrm{sen}(y) & -\mathrm{sen}(x)\cos(y) & \cos(x)\cos(y) \end{array}\right]$$

4.4 Ventajas y desventajas

El modelo propuesto simplifica de manera notable el procedimiento de reconstrucción de fronteras que era necesario en el modelo Extendido, ya que éste último

requería de un procedimiento especializado para cada nodo Extendido. El algoritmo de reconstrucción en sí no presenta problemas, excepto que si el árbol BSP dentro de un nodo BSP tuvo errores de precisión matemática éstos serán mucho más notables que si errores equivalentes se producían en el modelo Extendido. Sin embargo, como ya se mencionó, este problema se presenta únicamente cuando el árbol BSP presenta errores. Para árboles BSP correctos, éstos poseen la bien conocida propiedad de que no existe ningún caso en el que puedan obtenerse errores de visualización.

Además, la eliminación de partes ocultas siempre es correcta, sin importar la complejidad del nodo BSP, siempre y cuándo éstos sean creados utilizando la restricción descrita en **[Sección3.3.3]**, a diferencia del modelo Extendido donde no siempre es suficiente con conocer el orden de recorrido de los nodos del OctTree.