El pipeline gráfico

Para llevar a cabo una representación virtual de un ambiente tridimensional, se realiza un modelado del escenario. Dicho modelo incluye la representación geométrica de los objetos presentes, las condiciones de iluminación de la escena y el lugar dentro de la misma desde dónde es apreciada. Cada objeto es representado a través de combinaciones, más o menos complejas, de figuras geométricas, el color o las texturas de sus superficies, y su posición y orientación en la escena. Luego se configuran las condiciones de iluminación, es decir posición, color, tipo y demás atributos de cada fuente de luz presente en la escena. Por último, se determina la posición y dirección del punto de vista o de la cámara a través de la cual se visualiza el ambiente.

La visualización del ambiente en un dispositivo, como la pantalla de un monitor, es posible gracias a software y hardware especializado. Dicha especialización permite dibujar eficientemente sobre una pantalla bidimensional la proyección, comúnmente en perspectiva, de una escena tridimensional (ver Figura 3.1).

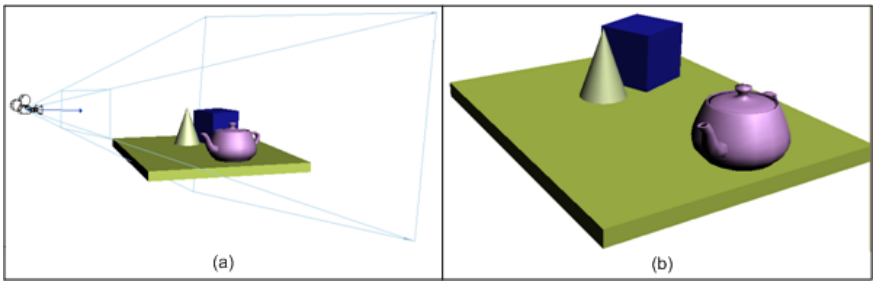

Figura 3.1 – A partir del modelado de una escena tridimensional (a) se obtiene una imagen o cuadro (b). En (a) se ve el modelado de la escena utilizando objetos y posicionando una cámara. Además, puede verse el volumen de visualización de la cámara. En (b) se muestra lo visualizado por la cámara: la escena proyectada (en perspectiva) en el plano frontal del volumen de visualización.

El conjunto de operaciones que se realiza para, a partir del modelado del escenario, lograr una imagen para dibujar en la pantalla es llamado *pipeline gráfico.* En computación gráfica, al proceso de generar una imagen

bidimensional, para mostrarla en un dispositivo, se lo conoce como *renderizar*[1] [Akenine-Möller et al. 2008, 11].

La denominación pipeline hace referencia a que las operaciones se encuentran secuenciadas y que el conjunto de ellas puede verse como una línea de producción de una fábrica. Cada operación corresponde a una estación de trabajo. Cada estación, o etapa del pipeline, toma como entradas las salidas de la estación anterior, y produce salidas que sirven de entradas para la subsiguiente. Esta relación entre las operaciones permite cierto grado de paralelismo. Las estaciones pueden estar todas trabajando simultáneamente si el pipeline es correctamente alimentado; claro esta, cada etapa sobre un conjunto de datos diferente.

La navegación interactiva de ambientes virtuales implica la necesidad de generar en el dispositivo de salida una secuencia de imágenes (también llamadas cuadros) que brinden una sensación de continuidad en el desplazamiento o movimiento. El pipeline gráfico debe renderizar cuadros con una velocidad de al menos 15 cuadros por segundo para que el ser humano no note discontinuidades en la animación [Akenine-Möller et al. 2008, 1].

El rendimiento del pipeline gráfico, o de aquellas aplicaciones que lo utilizan para la visualización de ambientes virtuales, es medido comúnmente a partir de la velocidad con la que se renderizan los cuadros. Las tarjetas gráficas que implementan las funcionalidades del pipeline gráfico junto con el software que las acompaña (los controladores o drivers) son fabricadas de manera genérica. De esta manera, las distintas etapas son optimizadas por los fabricantes por igual. Por lo tanto, el rendimiento es determinado por la manera en que la aplicación utiliza el pipeline gráfico. Finalmente, como el pipeline gráfico consiste en una secuencia de etapas que trabajan paralelamente, el rendimiento resultante está limitado por la etapa que requiere mayor tiempo de procesamiento, la cual es denominada *"cuello de botella"*.

Comentado [LG1]: Ojo que el cuello de botella puede ser la CPU tambien. Esto es el preparado de los datos para alimentar el pipeline. Tal vez puede ponerse como una ETAPA mas y describirla

En computación gráfica, para introducir los datos al pipeline gráfico se utilizan librerías gráficas que hacen uso de los distintos hardware gráficos y

[1] Renderizar es una castellanización del verbo en ingles "to render", cuya traducción literal es traducir o interpretar. Puede considerarse que la acción consiste en traducir un modelo tridimensional a una imagen bidimensional.

brindan una abstracción común del mismo. Las librerías gráficas más populares son DirectX y OpenGL, en sus diferentes versiones.

A continuación, se expondrá una breve explicación de las operaciones que ocurren en un pipeline gráfico genérico y los datos que son ingresados y utilizados en cada etapa.

Vértices y primitivas

Los objetos de la escena son modelados mediante una malla de triángulos. Las librerías gráficas permiten especificar para cada objeto los vértices de dichos triángulos y las conexiones entre ellos.

La información incluida en cada vértice puede variar dependiendo de la aplicación, pero comúnmente contiene:

- **Su posición**: las coordenadas tridimensionales de la ubicación de dicho vértice según un eje de referencia determinado en el modelo del propio objeto.
- **La normal**: la dirección del vector normal al mallado al que pertenece el vértice en esa posición. Es usada para calcular la incidencia de las distintas fuentes de luz y para determinar la cara frontal de los triángulos que define.
- **El color**: puede contener la información del color del vértice.
- **Las coordenadas de textura**: alternativamente al color, si el mismo debe ser tomado de una textura, las coordenadas para realizar el muestreo dentro de la textura correspondiente.

Además del conjunto de vértices que representarán un objeto, debe introducirse la información de cómo deben usarse los mismos para armar los triángulos o primitivas. Esta información es simplemente una secuencia de los índices de los vértices descriptos anteriormente.

Transformaciones

Las posiciones de los vértices que modelan cada objeto son, usualmente, relativas a un eje de referencia propio de cada objeto. Esto permite que el mismo modelo sirva para representar distintas instancias del objeto dentro de una escena.

Una primera etapa del pipeline consiste en transformar las coordenadas de los vértices en coordenadas normalizadas del dispositivo. Para esto a cada vértice se le aplica una secuencia de transformaciones utilizando matrices de dimensión 4x4 y coordenadas homogéneas (véase una buena explicación en el OpenGL Programming Book, Apéndice G). Las posiciones de los vértices son afectadas entonces por 3 transformaciones:

- **Mundo:** A partir de la posición, orientación y escala de la instancia del objeto en la escena, se arma una matriz que lleva las coordenadas de los vértices del sistema de referencia propio del objeto al sistema de referencia de la escena, común a todos los objetos.
- **Vista:** Para mostrar lo que es percibido desde un punto de vista, se aplica una matriz que hace coincidir la posición de la cámara con el origen de coordenadas. Además, se hace coincidir la dirección de la cámara con el eje Z positivo, quedando el eje Y hacia arriba y el eje X hacia la derecha. Así las posiciones de los vértices de los objetos quedan relativas a la ubicación y orientación de la cámara.
- **Proyección:** El espacio que es visualizado de la escena es definido por un volumen de visualización. Los objetos dentro de ese volumen son proyectados en la cara delantera de dicho volumen para obtener la imagen a mostrar. Utilizando una proyección perspectiva, el volumen de visualización tiene la forma de la parte inferior de una pirámide de base rectangular truncada, en inglés, *frustum*. Finalmente, la transformación de proyección convierte ese volumen de visualización en un volumen de visualización canónico, un cubo con sus vértices extremos en (-1, -1, -1) y (1, 1, 1).

Ensamblado, Recorte y Mapeo a pantalla

A continuación, los vértices transformados son ensamblados para formar las primitivas (puntos, líneas o triángulos) descriptas a partir de la secuencia de índices. En esta etapa, las primitivas que quedan fuera del volumen canónico de visualización son descartadas del pipeline. Aquellas que se encuentran parcialmente dentro, son recortadas reemplazando los

vértices que quedan fuera por vértices en las intersecciones de la primitiva con el cubo canónico.

Luego, las primitivas son mapeadas a coordenadas de pantalla a partir de la posición y dimensión de la ventana en la cual se visualizará la escena. El mapeo consiste de un escalado seguido de una traslación, afectando a las coordenadas X e Y (no afecta a la coordenada Z).

Las coordenadas de pantalla X e Y junto con las coordenada Z (valor de profundidad) son pasadas a la siguiente etapa de rasterización.

Rasterización

El objetivo de esta etapa es determinar el conjunto de pixeles que formarán parte de la imagen final a visualizar. Esta etapa usa un buffer bidimensional para la generación progresiva de la imagen o cuadro final,que es llamado frame-buffer. La primera operación consiste en determinar los pixeles necesarios para la representación de cada una de las primitivas a visualizar. El proceso, también conocido como escaneo de conversión (en inglés, *scan conversion*), genera una estructura transitoria llamada fragmento. Cada fragmento contiene los atributos ingresados con los vértices junto a la posición. El valor de los atributos es obtenido mediante una interpolación lineal de los valores presentes en cada vértice de la primitiva, utilizando como pesos la posición del fragmento relativa a los vértices.

Dos o más primitivas pueden generar fragmentos en la misma posición. El fragmento que debe dibujarse es aquel que se encuentra más cercano al observador, es decir, aquel que no se encuentra tapado por otro. El valor de la coordenada Z, o valor de profundidad, es utilizado para dirimir el fragmento que se convertirá en píxel. Para esto se utiliza una estructura de datos llamada Z-buffer o buffer de profundidad (en inglés, *depth buffer*) que almacena el valor de profundidad del último fragmento dibujado en el frame-buffer. Un fragmento es descartado si su valor de profundidad es mayor al que se encuentra en el Z-buffer para la misma posición.

Comentado [LG2]: Ver si hace falta aclarar que los fragmentos pueden ser procesados en paralelo, de manera independiente uno de otro.

El color del fragmento candidato a convertirse en un píxel de la imagen final es decidido en esta etapa. El color puede obtenerse del atributo que acompaña a cada vértice o puede muestrearse de una textura en particular,

a partir de las coordenadas de textura para ese fragmento. Más aún, operaciones de iluminación y transparencia pueden contribuir en el color final de un píxel.

Finalmente, los fragmentos dibujados en el frame-buffer son mostrados en la pantalla del dispositivo.

Temas pendientes:

- Aliasing – ver si se puede introducir algo de aliasing aca o sino mas adelante
- Iluminación – por ahí se puede hablar mejor de la iluminación y del uso de las normales.
- Viewport o visor. Ventana de recorte o de visualizacion
- Pipeline programable. Mas adelante?
- Back-Face Culling – Por ahí se puede mencionar que las primitivas cuya normal se encuentran en la misma direccion q la camara se descartan porq estan dadas vueltas. Habria que hablar del winding cuando se arma un triangulo.