CC52B - Computación Gráfica

Profesor Patricio Inostroza

Profesor Auxiliar Eduardo Graells

21 de Abril de 2006

Clase Aux #5

Problema 1

(Fuente: Aux #5, Otoño 2004, Por Alfredo Cofré).

Se desea crear una emocionante secuencia de persecución de naves espaciales, tales como la de la *figura 2.1*. Para esto, se dispone de una nave espacial A, la cual se encuentra ubicada tal como en la *figura 2.2 y 2.3*, la cual se debe clonar y convertir en la nave B. Para convertir la nave A en la nave B, se deben usar matrices de transformación 3D. La nave **debe girar sobre la punta de su ala derecha**, que es fácilmente identificable, puesto que tiene una pelota verde encima. **Se desea que este punto, que originalmente (en la nave A) se encuentra en las coordenadas (1, -0.5, -1.75), se ubique finalmente (en la nave B) en el punto (5.2, -0.5, 1.75). Además, la nave B debe estar girada en 90ª en torno al eje X c/r a la nave A. La nave B tiene la mitad del tamaño de la nave A, tanto en los ejes X, Y y Z**. Encuentre la matriz de transformación.

Figura 2.1: Emocionante persecución espacial

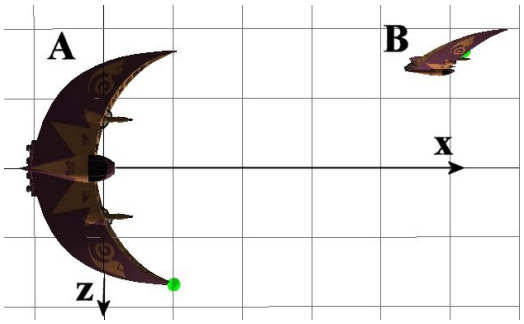

Figura 2.2

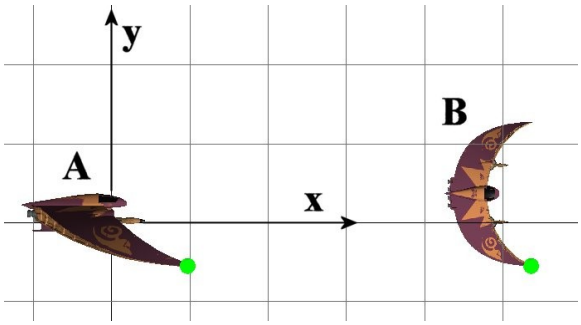

Figura 2.3

Solución

Lo primero que debemos hacer es trasladar la nave A desde el punto verde al origen. Como las coordenadas de dicho punto son (1, -0.5, -1.75), debemos trasladarlo (-1, 0.5, 1.75). Veamos la forma de la matriz de traslación:

$$M_T = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0.5 \\ 0 & 0 & 1 & 1.75 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La matriz de rotación en torno al eje X en un ángulo de 90° es:

$$M_{R(90°)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -sen(\alpha) & 0 \\ 0 & sen(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Después escalamos el objeto a la mitad:

$$M_S = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Y por último trasladamos a su ubicación final:

$$M_{T_2} = \begin{bmatrix} 1 & 0 & 0 & 5.2 \\ 0 & 1 & 0 & -0.5 \\ 0 & 0 & 1 & 1.75 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Así, nuestro punto P' (alias "el punto verde de la nave B") es (recuerden que se expresa en coordenadas homogéneas):

$$P' = M_{T_2} * M_S * M_R * M_T * \begin{bmatrix} 1 \\ -0.5 \\ -1.75 \\ 1 \end{bmatrix}$$

OJO: Recuerden que en algunos libros las matrices tienen una notación diferente (se usan las traspuestas). En ese caso se deben multiplicar en el orden inverso a como lo hacemos nosotros.

Ahora veremos cómo funciona la pila de matrices de OpenGL (Stack Matrix). Para esto vamos a ilustrar esta funcionalidad al observar un sistema solar.

¿Qué es la pila de matrices? Inicialmente nuestro sistema de coordenadas está definido por una matriz de identidad inicial. A esta matriz le podemos premultiplicar matrices de transformación, cambiando el sistema de los objetos que grafiquemos posteriormente a la transformación. La pila de matrices es un conjunto de matrices que nos permite tener más de una matriz y volver hacia matrices anteriores, o guardar la actual y crear una matriz nueva igual a ella.

¿De qué me sirve esto? Ilustremos con el siguiente ejemplo: Tengo un objeto Sol, un objeto Tierra, un objeto Luna y un objeto Marte. El Sol se encuentra en el origen de mi sistema, la Tierra gira alrededor del Sol (y rota sobre su propio eje horizontal, que está inclinado 23°) y la Luna gira alrededor de la Tierra (asumamos que respecto al eje "horizontal" del planeta). De Tierra y Marte conocemos la distancia hasta el Sol, el ángulo que forman (desde una posición inicial) en la trayectoria elíptica alrededor de la gran estrella, y también sus ángulos de rotación respecto a su propio eje. De la Luna conocemos la distancia hasta la Tierra y también el ángulo que ha rotado alrededor de nuestro planeta.

¿Cómo hacemos para graficar esto? Podemos hacer todo "por fuerza bruta", es decir, calcular todas las coordenadas actuales respecto al origen para cada uno de los objetos. También podemos hacer traslaciones y rotaciones, y luego deshacerlas (esto es lo más intuitivo), y pasar a graficar el siguiente objeto.

Pero con matrices es mucho más sencillo. El proceso sería así:

- Graficamos el sol
- Apilamos la matrix actual. (Esto es equivalente a sacarle una copia y dejarla debajo de la copia, y considerar la copia como matrix actual a la cual se le aplicaran las transformaciones): **`glPushMatrix()`**.
- Rotamos el ángulo que ha girado la Tierra alrededor del Sol (el eje depende de cómo definimos el plano en el cual se encuentran los planetas. Consideremos el

plano XY, por lo que rotaría en torno al eje Z).

- Trasladamos la distancia hasta el Sol, rotamos los 23° de inclinación en torno al eje horizontal (como no buscamos rigurosidad física, podemos elegir el eje X o eje Y como eje de rotación) y graficamos la Tierra.
- Rotamos el ángulo que ha girado la Luna alrededor de la Tierra y trasladamos la distancia hacia ella. Graficamos la Luna.
- **Desechamos la matriz actual: `glPopMatrix()`.**
- Ahora repetimos el proceso para Marte (y el resto de los planetas).

¿Qué sucede cuando desechamos la matriz actual? Se descarta la matriz que tenemos y se retoma la que viene en la pila, que está tal cual como fue apilada. Como verán esto es mucho más barato que rotar de vuelta, trasladar de vuelta, y volver a rotar de vuelta. Y nos permite establecer sistemas de coordenadas independientes para cada objeto que queramos graficar sin preocuparnos del "sistema de coordenadas" padre. Esto es muy útil para aplicaciones de brazos o piezas mecánicas que son articuladas (por dar un ejemplo).

Ejemplo de función graficar:

```
GLvoid graficar()
{
     glPushMatrix();
         glTranslatef(this->posicionX(), this->posicionY(), this->posicionZ());
         glRotatef(this->rotacion(), 0.0f, 0.0f, 1.0f);
         glRotatef(this->inclinacion(), -1.0, 0.0f, 0.0f);
         gluSphere(this->quadratic, this->_radio, 32, 32);
         list<Satelite *>::iterator iterador;
         for (iterador = lunas.begin(); iterador != lunas.end(); iterador++)
              (*iterador)->graficar(); //por si tiene varias lunas
     glPopMatrix();
};
```