Espacio proyectivo (Espacio de coordenadas homogéneas)

Justificación teórica de la representación interna de vectores y Trasformaciones en OpenGL

Consideramos las duplas $\{G_k, T_k\}_{k=1, 2, \ldots, n}$, formadas por una *nxn* - matriz G_k y un vector-columna $T_k \in R^n$. Nombramos esas duplas como '*movimientos*' en el sentido que cada dupla $\{G_k, T_k\}$ genera un mapeo del espacio R^n a si mismo mediante la transformación

$$v \rightarrow G_k v + T_k \text{ para todo } v \in R^n.$$

Introducimos de la manera formal el mapeo $\{G_k, T_k\} \rightarrow \begin{bmatrix} G_k & T_k \\ \overline{0} & 1 \end{bmatrix}$ de cada movimiento $\{G_k, T_k\}$ a la matriz de bloques $\begin{bmatrix} G_k & T_k \\ \overline{0} & 1 \end{bmatrix}$, donde $\overline{0}$ es el vector-renglón de dimensión *n* formado por ceros.

Igual, de una manera formal mapeamos todos vectores $v \in R^n$ a un espacio de vectores con un componente escalar adicional y asignemos valor de este componente con 1:

$$v \rightarrow \begin{bmatrix} v \\ 1 \end{bmatrix}$$

Supongamos que en este espacio como un postulado se cumple la siguiente propiedad: *para todo escalar w≠0,* $\begin{bmatrix} v \\ 1 \end{bmatrix} = \begin{bmatrix} wv \\ w \end{bmatrix}$. Tal espacio se denota $\mathbf{RP}^n$ or $\mathbb{P}_n(\mathbb{R})$ y se llama *espacio proyectivo o espacio de las coordenadas homogéneas*.

Nota: En términos geométricos el espacio proyectivo $\mathbf{RP}^n$ se puede interpretar como el espació de las rectas en $\mathbf{R}^n$ donde las rectas paralelas se declaran como equivalentes.

Se puede mostrar matemáticamente que para todo $v \in R^n$ el resultado de superposición de la serie de movimientos al *v*,

$$\mathbf{G_n(G_{n-1}(G_{n-2}(\ldots G_2(G_1 v + T_1) + \ldots + T_{n-2}) + T_{n-1}) + T_n}$$

al ser mapeado de la manera acaba mencionada al espacio proyectivo:

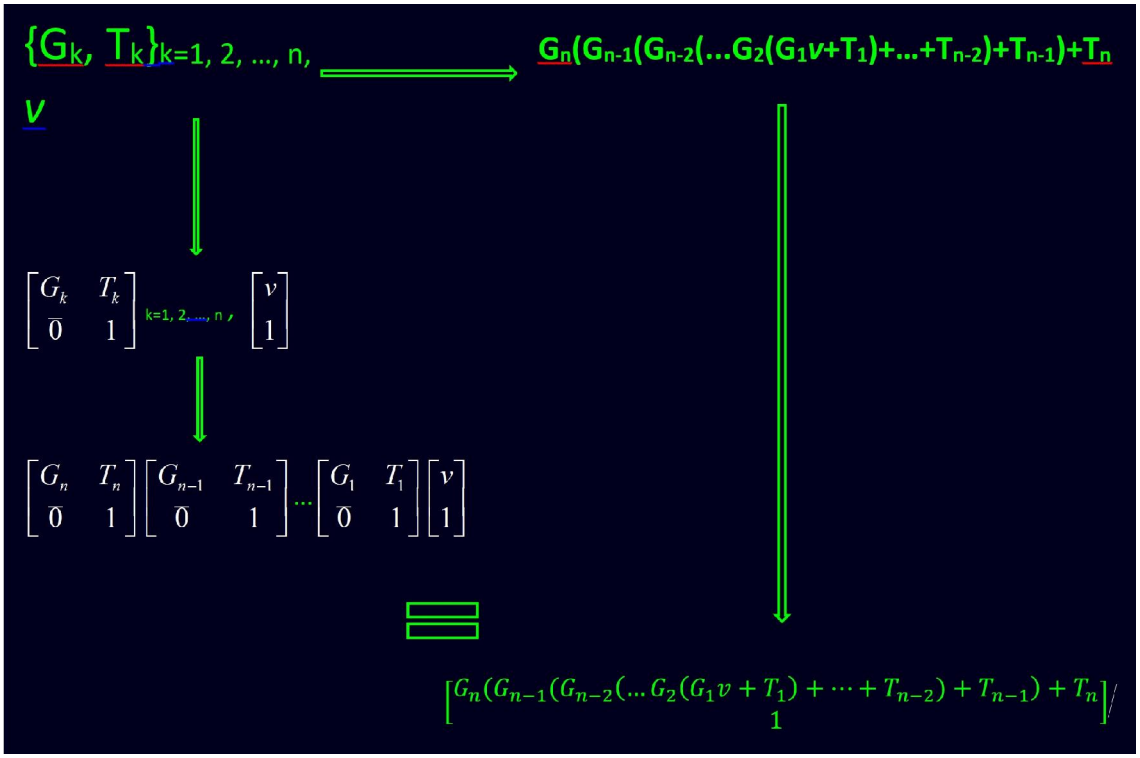
$$\mathbf{G_n(G_{n-1}(G_{n-2}(\ldots G_2(G_1 v + T_1) + \ldots + T_{n-2}) + T_{n-1}) + T_n} \rightarrow \begin{bmatrix} G_n(G_{n-1}(G_{n-2}(\ldots G_2(G_1 v + T_1) + \cdots + T_{n-2}) + T_{n-1}) + T_n \\ 1 \end{bmatrix}$$

coincide con

$$\begin{bmatrix} G_n & T_n \\ \overline{0} & 1 \end{bmatrix} \begin{bmatrix} G_{n-1} & T_{n-1} \\ \overline{0} & 1 \end{bmatrix} \cdots \begin{bmatrix} G_1 & T_1 \\ \overline{0} & 1 \end{bmatrix} \begin{bmatrix} v \\ 1 \end{bmatrix}.$$

Es decir, *la superposición de movimientos* $\{G_k, T_k\}_{k=1, 2, \ldots, n}$ *aplicada a los vectores del espacio tradicional* $\boldsymbol{R}^n$ *puede ser sustituida por el producto de las matrices* $\begin{bmatrix} G_k & T_k \\ \overline{0} & 1 \end{bmatrix}_{k=1, 2, \ldots, n}$ *aplicado al resultado del mapeo* $\boldsymbol{R}^n \rightarrow \mathbf{RP}^n$ *mediante la regla* $v \rightarrow \begin{bmatrix} v \\ 1 \end{bmatrix}$.

Esta propiedad se presenta gráficamente a continuación: no importa cuál de los dos caminos de flechas elegimos en la esquina superior-izquierda, el resultado final será el mismo. Pero el camino que inicia de la flecha vertical tecnológicamente es mejor por las razones presentadas a continuación.



Para mayores detalles, leer inicio del archivo
http://newton.uam.mx/xgeorge/uea/graficacionII/Homogenious%20coordinates.doc

Relación de la representación interna de la geometría con la técnica de manipulación de funciones OpenGL

Externamente, el diseñador de una aplicación gráfica piensa en términos geométricos ordinarios. Este se trata tanto con los objetos geométricos (modelos) como con las trasformaciones.

Sin embargo, un llamado de la función que manipula con objetos transforma automáticamente los vectores tipo {x,y,z} en R^n a la representación interna de vectores en **RP^n**, es decir a los vectores de tipo {x,y,z,1} con tal propiedad que para todo $w \neq 0$ se cumple equivalencia w*{x,y,z,1} = {wx, wy, wz, w} = {x,y,z,1}.

Igualmente, cada transformación (por ejemplo, glRotate, glTranslate, etc.) que refiere a las trasformaciones expresados en términos intuitivos comprensibles genera implícitamente una transformación interna representada mediante una matriz de la dimensión 4x4 (en lugar de 3x3!) la

cual se usa de misma manera que las matrices tipo $\begin{bmatrix} G_k & T_k \\ \bar{0} & 1 \end{bmatrix}$ consideradas arriba, es decir mediante la multiplicación.

Luego, en cada instante de la vida de una aplicación gráfica existe la matriz corriente M_c. Esta matriz se construye como producto de todas transformaciones anteriores. En términos expuestos, se puede decir que

$$M_c = \begin{bmatrix} G_{n-1} & T_{n-1} \\ \bar{0} & 1 \end{bmatrix} \cdots \begin{bmatrix} G_1 & T_1 \\ \bar{0} & 1 \end{bmatrix}$$

Si programador quiere aplicar una nueva transformación, $\begin{bmatrix} G_n & T_n \\ \bar{0} & 1 \end{bmatrix}$, al resultado de las transformaciones anteriores y posteriormente regresar a la **M_c** actual, él guarda **M_c** en pila de transformaciones mediante llamado glPushMatrix(), luego llama la función que construye $\begin{bmatrix} G_n & T_n \\ \bar{0} & 1 \end{bmatrix}$ y automáticamente aplícala multiplicando por **M_c**. Es decir la nueva matriz corriente sería igual a $\begin{bmatrix} G_n & T_n \\ \bar{0} & 1 \end{bmatrix} M_c$. Si el programador después de aplicar esta matriz corriente actualizada quiere regresar a la anterior, él llama glPopMatrix().

Es decir, la tecnología basada en las coordenadas homogéneas implementada en OpenGL permite no preocuparse sobre toda la sucesión de las transformaciones anteriores, sino solo sobre la nueva transformación y la corriente.