Atributos de las primitivas gráficas

Prof. Wílmer Pereira
http://www.ldc.usb.ve/~wpereira

Atributos de colores, líneas y polígonos

Los sistemas gráficos mantiene una lista de atributos con sus valores lo que constituye una máquina de estados. Siempre parten con valores por defecto.

- Los valores de los atributos pueden ser consultados.
- Cambiar cada atributo sólo afecta a lo que se pinta a partir de la modificación del estado.
- Dentro de un mismo, glBegin(X) ... glEnd(), se pueden cambiar los valores de los atributos cuantas veces desee el programador.

Colores ...

- Los componentes RGB generan un poco más de 16 millones de colores donde las escalas de grises se logran cuando los tres componentes tiene el mismo valor.
- Se pueden definir tablas de colores indixadas o tablas de colores en un arreglo.

Arreglos de vértices y colores

La asociación de estos dos arreglos permite, rápidamente, pintar objetos con diferentes colores, sin cambiar el color por defecto

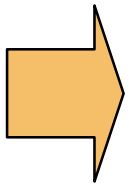
- El siguiente código define un cubo con todos los vértices de su cara frontal en azul y los vértices de la cara posterior en rojo:

```
typedef GLint vertex3 [3], color3 [3];
vertex3 pt [8] = { {0, 0, 0}, {0, 1, 0}, {1, 0, 0}, {1, 1, 0}, {0, 0, 1},
                   {0, 1, 1}, {1, 0, 1}, {1, 1, 1} };
color3 hue [8] = { {1, 0, 0}, {1, 0, 0}, {0, 0, 1}, {0, 0, 1}, {1, 0, 0},
                   {1, 0, 0}, {0, 0, 1}, {0, 0, 1} };

glEnableClientState (GL_VERTEX_ARRAY);
glEnableClientState (GL_COLOR_ARRAY);

glVertexPointer (3, GL_INT, 0, pt);   // El tercer parámetro es el
                                      // espaciado entre colores
glColorPointer (3, GL_INT, 0, hue);
```

- Hay varios *buffers* de color como:



GL_COLOR_BUFFER_BIT
GL_COLOR_ARRAY
GL_POST_COLOR_MATRIX_TABLE …

Mezclado de Colores

Cuando hay solapamiento y transparencia, el color del objeto en primer plano se puede fusionar con el color del objeto de fondo, habilitando la mezcla (*blend*).

- Para activar y desactivar el mezclado

  glEnable(GL_BLEND)
  glDisable(GL_BLEND)

- El color resultante, así como el factor de transparencia resultante, será:

$$(S_r R_s + D_r R_d\,,\; S_g G_s + D_g G_d\,,\; S_b B_s + D_b B_d\,,\; S_a A_s + D_a A_d)$$

Donde:

(R_s, G_s, B_s, A_s)	Color del componente fuente
(R_d, G_d, B_d, A_d)	Color del componente destino
(S_r, S_g, S_b, S_a)	Factor de mezclado de la fuente
(D_r, D_g, D_b, D_a)	Factor de mezclado del destino

Los factores de mezclado se definen con: glBlendFunc(MezcladoFuente,MezcladoDestino)

Atributos de puntos y líneas

El tamaño y color de los puntos y líneas es configurable con: glPointSize(Tamaño), glLineWidth (Tamaño) y glColor*(RGBA).

- Cada punto es un glVertex*(Coordenadas) dentro del par glBegin(X)/glEnd()
- Además en las líneas se puede definir el estilo con:

  glLineStipple(FactorRepeticion , Patron)

  Por ejemplo un patrón 0x00FF sería 8 píxeles pintados y 8 píxeles con el color de fondo. El patrón por defecto es 0xFFFF. Por supuesto se usa entre:

  ```
  glEnable(GL_LINE_STIPPLE)
          …
  glDisable(GL_LINE_STIPPLE)
  ```

- Por último puede haber interpolado de colores con:

  glShadeModel(X) donde X puede ser GL_SMOOTH o GL_FLAT

Atributos de llenado

Inicialmente se debe hacer el llenado de polígonos convexos aunque también es posible llenar: círculos, elipses y áreas delimitadas por curvas

- El llenado puede ser con: patrones, texturas, un único color, mezclado de colores (si hay transparencia) o hueco (sólo bordes)…
- También se puede configurar el aspecto de los bordes: color, ancho y estilo (como cualquier línea) independiente del aspecto del llenado.
- Por último se puede mostrar sólo los vértices sin bordes ni relleno …
- En caso de relleno con patrón, se define en un arreglo y se activa el modo con:

```
glPolygonStipple(ArregloPatron);
```

y se habilita el llenado con:

```
glEnable(GL_POLYGON_STIPPLE);
…
glDisable(GL_POLYGON_STIPPLE);
```

Teselado o Diseño Alambrado

- Se puede mostrar sólo los bordes o sólo los vértices … Además en caso de figura en 3D definir la cara frontal y posterior:

```
glPolygonMode(Cara , Modo)
```

Los argumentos Cara pueden ser: GL_FRONT o GL_BACK y los argumentos de Modo: GL_LINE, GL_POINT o GL_FILL

- Para rellenar y pintar los bordes independientemente, se invoca la rutina de generación de polígonos dos veces:

```
glColor3f(1.0 , 0.0 , 0.0);
    /* invocar rutina de polígono */
glColor3f(0.0 , 1.0 , 0.0);
glPolygonMode(GL_FRONT , GL_LINE)
     /* invocar de nuevo rutina de polígono */
```

Llenado de polígonos en 3D

Configurar llenado y bordes independientemente puede generar efectos de costura entre el interior del polígono y sus bordes

- El problema está en como se calcula la profundidad para los bordes que es diferente para el llenado. Para ello se define una estrategia única de pintado para bordes y llenado con:

```
glEnable(GL_POLYGON_OFFSET_FILL)
glPolygonOffset(Factor1 , Factor2)
```

Donde Factor1 y Factor2 son parte de una fórmula única de profundidad:

*OffsetProfundidad = Factor1*PendienteMax + Factor2*Cons*

Los valores habituales son 0.75 y 1.0 respectivamente …

Recuperación de los valores de los atributos

El estado de cada atributo se recuperar en un arreglo para ser reutilizado o simplemente para verificación de errores. Se puede recuperar a través de una pila por grupos con glPushAttrib(X) y glPopAttrib()

- Para recuperar individualmente, las funciones son:

  glGetBooleanv(X , Y)
  glGetIntegerv(X , Y) …

  Donde X y Y son el atributo y un arreglo que recupera los valores, respectivamente. Por ejemplo:

  glGetFloatv(GL_CURRENT_COLOR , ValoresColores)

  Los valores de atributos pueden ser:

  GL_POINT_SIZE
  GL_CURRENT_RASTER_POSITION
  GL_LINE_WIDTH_RANGE …