OpenGL Utility Toolkit (GLUT)

(e-mail: Miguel.Lozano@uv.es)

21 de junio de 2001

Introducción

OpenGL es un API (Application Programming Interface) orientada al desarrollo de aplicaciones gráficas en general. Uno de los mayores logros conseguidos en la especificación de OpenGL fue la independencia entre ésta y el sistema de ventanas, altamente dependiente del sistema operativo y por tanto del hardware. De esta forma se lograba independendizar el API gráfico del hardware, dejándo en manos del sistema manejador de ventanas (Windows, X11, Motif, ...) la comunicación directa con el hardware. El problema entonces derivó en encontrar el eslabón necesario entre el API gráfico (OpenGL) y los posibles manejadores de ventanas (X11, Motif, Windows,...), dedicados básicamente a la creación y manejo de los eventos que éstas podían recibir.

En este contexto aparece la OpenGL Utility Toolkit (GLUT), cuyo objetivo no es otro que proporcionar ese eslabón, mediante un interface de programación (en C y Fortran) orientado a escribir programas OpenGL independientes del sistema de ventanas que utilice nuestro hardware. Podemos entender las GLUT como una capa situada por encima del sistema de ventanas de forma que éste permanecerá oculto para la OpenGL y mediante su sencillo API de programación, podremos crear y manejar ventanas donde dibujar con OpenGL. El principal objetivo de la arquitectura planteada (Figura 1) es conseguir que la portabilidad del código gráfico (GLUT+OpenGL) a otras plataformas sea automática.

Filosofía de diseño

GLUT, al igual que OpenGL, está basada en la filosofía de "máquina de estados". Las aplicaciones parten de un estado inicial, compartido por la mayoría de ellas y definido a través de ciertas variables de estado, de forma que las acciones ejecutadas dependerán del estado en el que se encuentre la misma. Por esta razón, resulta lógico que la aplicación pueda modificar los valores de las variables de estado, asegurando así el correcto resultado de sus acciones.

Las funciones GLUT comienzan por `glu-`, son sencillas, tienen pocos parámetros y nunca devolverán ningún tipo de dato dependiente del sistema de ventanas

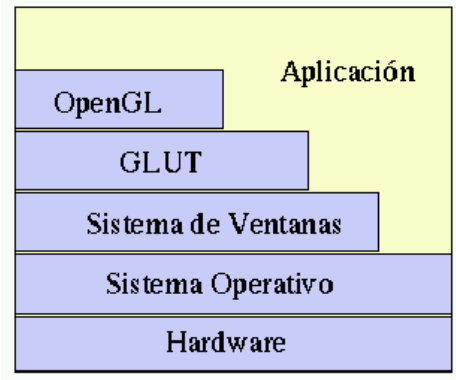

Figura 1: Jerarquía de las aplicaciones gráficas basadas en GLUT.

(manejadores, punteros, ...) , dado que como comentábamos anteriormente uno de los objetivos de la GLUT es encapsular al mismo.

GLUT está organizada en diferentes sub-APIs en base a distintas funcionalidad. A continuación repasaremos aquellos que sean necesarios para la realización de las prácticas de este curso.

Inicialización

Todas las rutinas de inicialización en GLUT vienen marcadas por el prefijo `glutInit-` y marcarán el estado inicial de la aplicación. La orden de inicialización `glutInit(..)` se encarga de negociar una sesión con el servidor de ventanas y posteriormente inicializar la librería GLUT. Esta orden debe invocarse una sóla vez en el programa y antes de la misma no podrá aparecer ninguna otra función de GLUT que no sea de inicialización. A continuación repasaremos el uso normal de las principales rutinas de inicialización.

Uso:
`void glutInit(int **argcp, char **argv);`
argcp es un puntero a la variable argc de la función main (sin modificar).
argv es la variable argv de la función main (sin modificar).

`void glutInitWindowPosition(int x, int y);`
`void glutInitWindowSize(int width, int height);`
x , y -> posición en pixels de la ventana (esquina superior izquierda).
width, height -> ancho y alto en píxels de la ventana.

`void glutInitDisplayMode(unsigned int mode);`
mode es el modo de display, OR lógica entre los siguientes valores:

GLUT_RGBA

Selecciona una ventana en modo RGBA, es el valor por defecto.

GLUT_RGB
: Alias de GLUT_RGBA.

GLUT_INDEX
: Seleciona una ventana en modo de índice de colores.

GLUT_SINGLE
: Selecciona una ventana en modo buffer simple, es el valor por defecto.

GLUT_DOUBLE
: Selecciona una ventana en modo buffer doble.

GLUT_ACCUM
: Selecciona una ventana con un buffer de acumulación.

GLUT_ALPHA
: Selecciona una ventana con componente alpha en el buffer de color.

GLUT_DEPTH
: Selecciona una ventana con buffer de profundidad.

GLUT_STENCIL
: Selecciona una ventana con un stencil-buffer o buffer de recorte.

GLUT_MULTISAMPLE
: Selecciona una ventana con soporte multimuestra.

GLUT_STEREO
: Selecciona una ventana estéreo.

GLUT_LUMINANCE
: Selecciona una ventana con un modelo de color basado en luminancia.

Procesado de Eventos

Una vez se ha completado la inicialización, los programas basados en GLUT entrarán en el bucle de procesado de eventos del cual ya no retornarán. La rutina que realiza el bucle comentado es

```
void glutMainLoop(void);
```

y lógicamente sólo deberá invocarse una vez dentro de nuestro programa. Para que el programa responda a los eventos adecuados, deberemos indicar a GLUT cúales son los eventos que queremos considerar y asignarles las funciones de *callback* necesarias. La sintaxis para registrar eventos asociándoles su funciones de callback es la siguiente:

```
glut[Display|Reshape|Mouse|...|Idle]Func(void (*func)(params...)
);
```

Por ejemplo, para controlar los eventos de ratón de una ventana GLUT, antes de llamar al bucle de eventos deberíamos incluir la siguiente línea:

```
...
glutMouseFunc(miRaton); // Asociamos una función de callback al Mouse
...
glutMainLoop();
}
```

La función de callback en este caso tendrá 4 parámetros, el boton, su estado y la posición x,y.

```
void miRaton(int boton, int estado, int x, int y)
{
if (boton == GLUT_LEFT_BUTTON && estado== GLUT_DOWN)
{
    ..... // Acciones a realizar al pulsar el botón izquierdo.
}
if (boton == GLUT_LEFT_BUTTON && estado == GLUT_UP)
{
     ...../// Acciones a realizar al soltar el botón izquierdo.
}
}
```

Dos funciones de callback fundamentales en aplicaciones GLUT son `glutDisplayFunc` y `glutIdleFunc`. La primera registra la función de dibujado de nuestra ventana, de forma que contendrá todas las ordenes de dibujado OpenGL necesarias para nuestros propósitos.

Uso:

```
void glutDisplayFunc(void (*func)(void) );
```

La segunda informa a GLUT a cerca de la rutina que deberá invocar en caso de no tener ningún evento que procesar. Esto es muy útil para realizar determinados cálculos y se hace necesario, por ejemplo, si queremos obtener imágenes animadas, donde necesitaremos actualizar ciertas variables para "animar" nuestras imagenes.

Uso:

```
void glutIdleFunc(void (*func)(void) );
```

Con lo visto hasta ahora, podemos definir un programa base al que se ajustarán nuestras aplicaciones GLUT.

```
int main(int argcp, char **argv)
{
   /* Poner el tamaño y posición de la ventana */
      glutInitWindowSize(SIZE_X, SIZE_Y);
      glutInitWindowPosition(0, 0);

   /* Seleccionar el tipo de modo de display:
      Buffer simple y color RGBA */
      glutInitDisplayMode(GLUT_RGBA | GLUT_SINGLE);

   /* Inicializar el estado de GLUT */
```

```
      glutInit(&argcp, argv);

   /* Abrir una ventana */
      glutCreateWindow("Laboratorio de Informática Gráfica");

   /* Sistema de Coordenadas 2D Ortogonal (Min_x, Max_x, Min_y,
Max_y) */
      gluOrtho2D(Min_x,Max_x,Min_y,Max_y);

   /* Registrar funciones Callback */
      glutDisplayFunc(miDraw)
      glutIdleFunc(miIdle)

   /* Iniciar el procesado de eventos */
      glutMainLoop();

   return 0;
};
```

Las únicas rutinas que no conocíamos hasta ahora son

- `glutCreateWindow("Laboratorio de Informática Gráfica");`

que pertence al sub-API de "Manejo de Ventanas" (no contemplado en este resumen). Esta rutina creará la ventana dado un nombre y un contexto OpenGL definido previamente.

- `gluOrtho2D(Min_x,Max_x,Min_y,Max_y);`

que pertenece a la librería auxiliar de utilidades de OpenGL (GLU). Esta rutina introducirá un sistema de coordenadas 2D Ortogonal cuyas dimensiones quedarán especificadas en los parámetros de la misma.

Material de Prácticas

Las prácticas se realizarán sobre el sistema operativo Linux donde estarán instaladas las libreriás GLUT y OpenGL. Para la realización de las mismas puede partirse del fichero "base.c" y compilarlo con el "Makefile" adjunto. Estos ficheros estarán en el directoriao de la asignatura /iilabs/lig???.

Referencias

El manual completo de las GLUT, así como diferentes versiones (fuentes y binarios), ejemplos, ..., pueden encontrarse en:

http://reality.sgi.com/mjk/glut3/glut3.html

http://www.opengl.org/developers/documentation/glut.html