NOTAS SOBRE GLUT

Antes de adentrarnos plenamente en el abordaje de OpenGL, es conveniente conocer los fundamentos de la programación con GLUT. Pero dado que su estudio no es el objetivo de este curso, nos centraremos sólo en los conceptos principales para poder trabajar con esta librería de herramientas. No obstante, podrán profundizar estos temas en la especificación de GLUT, que se detalla al final del documento.

CONTENIDOS

1. Introducción a GLUT

GLUT (OpenGL Utility Toolkit) es una API con formato ANSI C, pensado para escribir programas OpenGL, independientes del sistema de ventanas utilizado. Es una biblioteca de utilidades para programas OpenGL, que principalmente proporciona diversas funciones de entrada/salida con el sistema operativo.

Entre las funciones que ofrece se incluyen declaración y manejo de ventanas e interacción por medio de teclado y mouse. También posee rutinas para el dibujado de diversas primitivas geométricas (tanto sólidas como en modo wireframe), que incluyen cubos, esferas y teteras. También tiene soporte para la creación de menús emergentes.

La versión original de GLUT fue escrita por Mark J. Kilgard.

Los dos objetivos de GLUT son permitir la creación de código más portable entre diferentes sistemas operativos (GLUT es multiplataforma) y hacer OpenGL más simple.

Todas las funciones comienzan con el prefijo glut; son simples y requieren de pocos parámetros.

Asimismo, GLUT es una máquina de estados, es decir, que el sistema tiene un conjunto de variables de estado que durante la ejecución del programa fijan las características que debe tener la aplicación en cada momento.

Dependiendo de la funcionalidad podemos distinguir funciones para:
- Inicialización
- Control de ventanas
- Control de menús
- Procesamiento de eventos
- Registro de funciones callback
- Obtención del estado
- Visualización de fuentes de texto
- Dibujo de formas geométricas

A continuación, se comentan algunas de las particularidades y convenciones de GLUT:
- En una ventana, para GLUT y para el sistema operativo (cursor, posición de la ventana), el origen de coordenadas (0,0) se encuentra en la esquina superior izquierda. Esto es inconsistente con OpenGL, cuyo origen está en la esquina inferior izquierda, pero ello no supone ningún problema, como veremos más adelante.
- Los identificadores enteros en GLUT (utilizados para ventanas, menús, etc.) empiezan en 1 y no en 0.
- Las definiciones de las funciones se encuentran en el archivo de encabezado glut.h.

2. Inicialización

Las rutinas que empiezan con el prefijo *glutInit-* se utilizan para inicializar el estado de GLUT. Algunas de estas rutinas son:

- **void glutInit (int argc,char** argv)**

argc: apuntador al parámetro argc de la función main del programa.

argv: parámetro argv de la función main.

glutInit inicializa la librería GLUT, pudiéndosele pasar algún parámetro por línea de comandos a través de argc y argv.

Esta rutina debe ser llamada una única vez al principio del programa.

• **void glutInitWindowSize (int ancho, int alto)**

ancho: ancho de la ventana en píxeles.

alto: altura de la ventana en píxeles.

Esta rutina sirve para indicar el tamaño inicial de las ventanas que se creen.

• **void glutInitWindowPosition (int x, int y)**

x: posición en *x* de la ventana en píxeles.

y: posición en *y* de la ventana en píxeles.

Con esta función fijamos la posición inicial de las ventanas que se creen.

• **void glutInitDisplayMode (unsigned int modo)**

modo: modo de display. Estos modos serán estudiados en la materia en unidades posteriores. Es una composición mediante conectores | de algunos de los valores siguientes:

GLUT_RGBA: selecciona una ventana en modo RGBA. Es el valor por defecto.

GLUT_RGB: ocurre lo mismo que en GLUT_RGBA.

GLUT_INDEX: selecciona una ventana en modo de índice de colores. Se impone sobre GLUT_RGBA.

GLUT_SINGLE: selecciona una ventana en modo buffer simple. Es el valor por defecto.

GLUT_DOUBLE: selecciona una ventana en modo buffer doble. Se impone sobre GLUT_SINGLE. El doble buffer se utiliza para animaciones.

GLUT_ACCUM: selecciona una ventana con un buffer acumulativo.

GLUT_ALPHA: selecciona una ventana con una componente alfa del buffer de color.

GLUT_DEPTH: selecciona una ventana con un buffer de profundidad.

GLUT_STENCIL: selecciona una ventana con un buffer de estarcido.

GLUT_LUMINANCE: selecciona una ventana con un modelo de color en tonos de gris.

Esta función fija el modo de display inicial con que se crearán las ventanas. Los dos valores que suelen darse en la mayoría de las aplicaciones son GLUT_RGBA, para fijar un modelo de color RGB con componente alpha, y GLUT_DOUBLE, para seleccionar una ventana con doble buffer.

A modo de ejemplo, mostramos a continuación el esqueleto básico de un programa en GLUT con las inicializaciones pertinentes (archivo fuente número 1):

```
#include <GL/glut.h>
int main(int argc,char** argv) {
  // Inicializa la libreria GLUT
  glutInit(&argc,argv);
  // Selecciona modo de display: RGB y simple buffer
  glutInitDisplayMode(GLUT_RGB|GLUT_SINGLE);
  // Fija tamaño y posición inicial de las ventanas
  glutInitWindowSize(480,360);
  glutInitWindowPosition(50,50);
  return 0;
}
```

Al compilar el ejemplo, podremos notar que aún no es posible visualizar la ventana. La visualización la llevará a cabo más adelante.

3. Control de ventanas

Una vez inicializada la librería, podemos crear ventanas en la aplicación. Al hacerlo, la librería devuelve un entero, que es el identificador de ventana.

GLUT utiliza el concepto de ventana activa, ya que en cada momento de la ejecución del programa sólo es activa la última ventana que ha sido creada. Funciones para forzar el redibujado, intercambiar los buffers, fijar la posición y el tamaño actúan sobre la misma.

Sólo unas pocas funciones, como crear subventana, activar y/o destruir ventana ventana, hacen uso del identificador de ventana para conocer sobre cuál deben actuar.

GLUT proporciona un identificador de ventana, que es único para toda la aplicación.

Algunas de las principales funciones para el control de ventanas se detallan a continuación:

- **int glutCreateWindow (char *nombre)**

nombre: cadena de caracteres con el nombre de la ventana.
Devuelve un entero, que es el identificador único de la ventana.
Esta función crea una ventana principal. Cuando se genera una ventana, la ventana activa pasa a ser esta nueva ventana. Toda ventana lleva implícito un contexto de visualización de OpenGL. Este contexto es el que permite que la ventana creada con GLUT (o en su caso, con cualquier entorno de ventanas) se *ponga en contacto* con OpenGL.
Las ventanas creadas con esta función no son todavía visualizadas en su interior. La visualización la llevará a cabo una función de respuesta al evento de redibujado de la ventana. Trataremos este concepto más adelante.
El tamaño de la ventana, su posición y modo de display vienen dados por las especificaciones iniciales hechas mediante las funciones glutInit-, que vimos anteriormente.

- **void glutSetWindow (int idVentana)**

idVentana: identificador de la ventana.
Fija la ventana identificada por idVentana como ventana activa.

- **int glutGetWindow (void)**

Devuelve el identificador de la ventana activa.

- **void glutDestroyWindow (int idVentana)**

idVentana: identificador de la ventana.
Destruye la ventana identificada por idVentana.

- **void glutSwapBuffers (void)**

Intercambia los buffers de la ventana actual. Se utiliza en el modo de doble buffer. Esta importante característica de OpenGL (y de GLUT por extensión) hace que se reduzcan al mínimo los parpadeos durante el redibujado, especialmente si hay animaciones. El tema buffer será estudiado en la materia en unidades posteriores.

- **void glutPositionWindow (int x, int y)**

x, y: posición *(x,y)* en píxeles de la ventana.
Solicita el cambio de la posición de la ventana actual en la pantalla. Si la ventana actual es principal, *(x,y)* se toma con referencia al origen de la pantalla.

- **void glutReshapeWindow (int ancho, int alto)**

ancho: ancho de la ventana en píxeles.
alto: altura de la ventana el píxeles.
Solicita el cambio del tamaño de la ventana actual, suministrándole el ancho y el alto especificados.

Otras funciones referentes a las ventanas se muestran a continuación, junto con una breve descripción.

- **void glutPostRedisplay (void)**

Solicita el redibujado de la ventana actual.

- **void glutFullScreen (void): Solicita que la ventana actual sea maximizada.**

Sólo funciona para ventanas principales.

- void glutSetCursor (int cursor): cambia la imagen del cursor cuando sobrevuela la ventana actual.

A continuación, presentamos una ampliación del ejemplo 1, introduciendo la creación y el manejo básico de las ventanas (Archivo fuente número 2):

```
#include <GL/glut.h>
int main(int argc,char** argv) {
  // Identificadores para ventanas
  int idVentana1, idVentana2;
  // Inicializa la libreria GLUT
  glutInit(&argc,argv);
  // Selecciona modo de display: RGB y simple buffer
  glutInitDisplayMode(GLUT_RGB|GLUT_SINGLE);
  // Fija tamaño y posición inicial de las ventanas
  glutInitWindowSize(480,360);
  glutInitWindowPosition(0,0);
  // Crea las ventanas
  glutCreateWindow ("Ventana 1");
  idVentana1=glutGetWindow();
  glutInitDisplayMode(GLUT_RGBA|GLUT_SINGLE);
  glutInitWindowSize(200,500);
glutInitWindowPosition(500,100);
  glutCreateWindow("Ventana  2");  //  crea  el  main
window
  idVentana2=glutGetWindow();
  // Setea la ventana 1 como actual
  glutSetWindow (idVentana1);
  glutSetCursor(GLUT_CURSOR_NONE);
  // Destruye la ventana 1
  // glutDestroyWindow (idVentana1);
  glutMainLoop(); // entra en loop de reconocimiento
de eventos
  return 0;
}
```

La función glutMainLoop, que la explicaremos en detalle más adelante, es utilizada para que el programa ingrese en el loop de reconocimiento de eventos.

4. Control de menús

GLUT contiene menús básicos, con menús flotantes desplegables, que aparecen al hacer click en uno de los tres botones del mouse, en la posición donde se encuentra el cursor.

Los menús vienen identificados por un valor entero que asigna el sistema cuando se crea el menú, de una manera similar a lo que sucede con las ventanas. Para responder a las acciones del menú es necesario escribir una función de respuesta (callback) que las realice.

A esta función de respuesta se le ingresa como parámetro un número entero, que identifica el ítem del menú que ha sido seleccionado y que la función debe utilizar para decidir la acción a llevar a cabo.

Tal y como ocurre con las ventanas, en el caso de los menús también se aplica el concepto de menú activo, de forma que las funciones sucesivas a las que se llame actuarán sobre él en forma directa.

- **int glutCreateMenu (void (*funcion) (int valor))**

funcion: función *callback* que se llama cuando se selecciona un ítem del menú.
El valor que se le pasa a la función identifica al ítem del menú.
Devuelve un identificador único para el menú.
Esta función crea un menú flotante asociado a la ventana actual y devuelve un identificador único para el mismo. Este menú pasa a ser el menú activo.

- **void glutAddMenuEntry (char *nombre, int valor)**

nombre: cadena de caracteres que aparece en la opción del menú.
valor: identificador para el ítem del menú. Este valor es el que se pasa a la función de respuesta al menú.
Añade una entrada al final del menú activo y la cadena que se visualiza es el nombre. Cuando el usuario selecciona esta opción, se llama a la función de respuesta con el valor como parámetro.

- **void glutAddSubMenu (char *nombre, int idMenu)**

nombre: cadena de caracteres que aparece en la opción del menú principal.
idMenu: identificador del submenú asociado al ítem.
Crea un ítem del menú y le asocia un submenú que se despliega al ser seleccionada la entrada del menú principal. El submenú tiene que ser creado previamente como cualquier otro menú.

• **void glutAttachMenu (int boton)**
boton: botón del mouse al que asociamos el menú.
Asocia el menú activo a un botón del mouse en la ventana activa, de manera que el menú se despliega cuando se pulsa el botón del mouse. El botón puede valer GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON o GLUT_RIGHT_BUTTON, para asociarlo al botón izquierdo, central o derecho, respectivamente.

En el siguiente ejemplo (archivo fuente número 3), desplegamos un menú principal y un submenú asociado:

```
void RespuestaMenuColor(int value){}
void RespuestaMenuPrincipal(int value){}

int main(int argc,char** argv) {
  // Identificador de la ventana
  int idVentana;

  // Inicializa la librería GLUT
  glutInit(&argc,argv);
  // Selecciona modo de display: RGB y simple buffer
  glutInitDisplayMode(GLUT_RGB|GLUT_SINGLE);
  // Fija tamaño y posición inicial de las ventanas
  glutInitWindowSize(480,360);
  glutInitWindowPosition(50,50);
  // Crea la ventana
  idVentana=glutCreateWindow ("Ejemplo 3");

  // Identificadores de los menús
  int menuColor, menuPrincipal;
  // Crear submenú y asociarle su función de respuesta
  menuColor=glutCreateMenu (RespuestaMenuColor);
  glutAddMenuEntry("Rojo", 1);
  glutAddMenuEntry("Verde", 2);
  glutAddMenuEntry("Azul", 3);
  // Crear menú principal y asociarle su función de
respuesta
  menuPrincipal                                    =
glutCreateMenu(RespuestaMenuPrincipal);
  glutAddMenuEntry("Luz", 1);
  glutAddMenuEntry("Material", 2);
  glutAddSubMenu("Color", menuColor);
  // Asociar el menú al botón derecho del mouse
  glutAttachMenu(GLUT_RIGHT_BUTTON);
  // Código del programa
  // ...

  glutMainLoop(); // entra en loop de reconocimiento
de eventos
  return 0;
}
```

5. Procesamiento de eventos

GLUT tiene incorporado su propio ciclo de proceso de eventos. Para ello es necesario llamar a ese ciclo con la función glutMainLoop.

Esta función implementa un ciclo infinito que se encarga de consultar la cola de eventos y de responder a cada uno de ellos, ejecutando una función. La función que se ejecuta cuando se produce un evento debe ser escrita por el programador y, además, registrada como una función de respuesta (callback). De este modo, GLUT identifica a qué función llamar para responder a un determinado evento.

• **void glutMainLoop (void)**
Esta rutina debe llamarse una vez en el programa. Puesto que implementa un ciclo infinito (el ciclo sólo termina cuando se produce un evento de cerrar

aplicación), la función no termina nunca. Por ello, es necesario registrar previamente las funciones *callback* y, además, establecer que la llamada a esta función sea la última de main.

6. Registro de funciones de respuesta

El registro de las funciones de respuesta a eventos (llamadas callbacks) debe realizarse antes de llamar a la función glutMainLoop. Para ello, se utiliza un conjunto de funciones con sintaxis común: void glut[Event]Func (tipo (*function) (parámetros)), que indican que para responder al evento *Event* debe utilizarse la función *function.*

Existen tres tipos de eventos: de ventana, de menús y globales.

Los eventos de ventana, como su nombre lo indica, están asociados a las ventanas y tratan cuestiones como el redibujado, redimensionamiento y la opresión de un botón del mouse.

Los eventos de menú responden a la selección de las opciones de un menú.

Los eventos globales no están asociados a ninguna ventana ni menú y se refieren, fundamentalmente, al control de temporizadores y acciones idle.

El orden en que se registran las funciones de respuesta a los eventos es indefinido.

7. Funciones de respuesta a eventos de ventana

- **void glutDisplayFunc (void (*funcion) (void))**

funcion: función de respuesta al evento.
Registra la función que responde al evento de redibujado de la ventana activa. GLUT no proporciona una función de respuesta por defecto para el redibujado, por lo que es obligatorio escribir una función de este tipo para cada ventana creada. Si se crea una ventana y no se registra la función de respuesta al redibujado, se produce un error.

- **void glutReshapeFunc (void (*funcion) (int ancho, int alto))**

funcion: función de respuesta al evento .
Registra la función de respuesta al evento de redimensionamiento de la ventana activa. La función de respuesta debe tener como parámetros el ancho y la altura de la ventana tras el redimensionamiento. GLUT dispone, en este caso, de una función de respuesta por defecto, que se utiliza cuando no se registra ninguna función para este evento.

- **void glutKeyboardFunc (void (*funcion) (unsigned char tecla, int x, int y))**

funcion: función de respuesta al evento.
Se utiliza para registrar la función que responde a eventos del teclado sobre la ventana activa. La función de respuesta debe tener como parámetros la tecla que se ha oprimido (su caracter ASCII) y la posición *(x,y)* del puntero del mouse en ese momento, relativa a la ventana. No es posible detectar las teclas de modificadores directamente (CTRL, ALT, SHIFT). Debemos utilizar para ello la función glutGetModifiers. Si no se registra ninguna función, los eventos de teclado son ignorados.

- **void glutMouseFunc (void (*funcion) (int boton, int estado, int x, int y))**

funcion: función de respuesta al evento.
Registra para la ventana activa la función de respuesta a eventos del mouse. Los eventos de mouse se producen tanto cuando se hace click, como cuando se suelta un botón del mismo. Los parámetros de la función de respuesta deben ser el botón (GLUT_LEFT_BUTTON, GLUT_MIDDLE_BUTTON o GLUT_RIGHT_BUTTON), el estado del botón (GLUT_UP o GLUT_DOWN) y la posición *(x,y)* del puntero, relativa a la ventana.
Cuando existe un menú asociado a un botón del mouse, el evento de oprimir ese botón es ignorado, prevaleciendo el menú sobre otras funcionalidades. Como en el caso del teclado, es posible detectar el uso de modificadores con la función glutGetModifiers. Si no registra una función de respuesta a eventos del mouse, éstos son ignorados.

• **void glutMotionFunc (void (*funcion) (int x, int y))**
funcion: función de respuesta al evento.
Registra, para la ventana activa, la función de respuesta a movimientos del mouse cuando se mantiene oprimido algún botón del mismo. Los parámetros *(x,y)* indican la posición del apuntador en coordenadas relativas a la ventana.
• **void glutPassiveMotionFunc (void (*funcion) (int x, int y))**
funcion: función de respuesta al evento.
Registra, para la ventana activa, la función de respuesta a movimientos del mouse cuando no se mantiene oprimido ningún botón del mismo. Los parámetros *(x,y)* indican la posición del apuntador en coordenadas relativas a la ventana.
• **void glutSpecialFunc (void (*funcion) (int tecla, int x, int y)):**
Registra, para la ventana activa, el evento de pulsar una tecla especial. El parámetro tecla puede ser GLUT_KEY_Fn para teclas de función (n=1, 2, ..., 12), GLUT_KEY_RIGHT, GLUT_KEY_LEFT, GLUT_KEY_UP, GLUT_KEY_DOWN, GLUT_KEY_HOME, GLUT_KEY_END, GLUT_KEY_PAGE_UP, GLUT_KEY_PAGE_DOWN.

Finalmente, se presenta un ejemplo (archivo fuente número 4) donde se registra la función glutDisplayFunc para responder a eventos de redibujado de la ventana.

```
void Display_cb() {
  // envia al monitor la ventana vacia
  glutSwapBuffers();
}

int main(int argc,char** argv) {
  // Inicializa la librería GLUT
  glutInit(&argc,argv);
  //  Selecciona  modo  de  display:  RGB  y  double
buffering
  glutInitDisplayMode(GLUT_RGB|GLUT_DOUBLE);
  // Fija tamaño y posición inicial de las ventanas
  glutInitWindowSize(480,360);
  glutInitWindowPosition(50,50);
  glutCreateWindow("Ejemplo 4");
  //declara los callbacks
  glutDisplayFunc(Display_cb);
  glutMainLoop(); // entra en loop de reconocimiento
de eventos
  return 0;
}
```

Primeramente, se realiza la inicialización de GLUT, que provee memoria y recursos necesarios.

Algo importante a tener presente es que, por cuestiones de eficiencia, se debe solicitar solamente lo que se va a utilizar. En este ejemplo, en relación al buffer de color, no se solicita el canal Alpha, ya que se dibujarán figuras totalmente opacas (se requiere el canal Alpha, a través de GLUT_RGBA, si se usarán transparencias o alguna forma de mezcla de colores).

Este criterio se aplica también al solicitar simple buffer o doble buffer. Siempre que se pretenda trabajar en tiempo real, se requiere doble buffer. En tal caso, mientras se percibe el buffer *de adelante* (front buffer), todo el dibujo se realiza en el buffer *de atrás* (back buffer) y, cuando está listo el proceso de carga del buffer, se intercambian los buffers con glutSwapBuffers; el de atrás pasa al frente y se actualiza la pantalla. Con la técnica de doble buffer se evita el parpadeo (flickering), el cual se produce en las animaciones que son efectuadas con un solo buffer, debido a que se está dibujando sobre lo que se está visualizando.

En unidades posteriores se explicarán en detalle los conceptos vinculados a buffers.

8. Funciones de respuesta a eventos globales

Estas funciones serán estudiadas en unidades posteriores de la materia:

• **void glutIdleFunc (void (*funcion) (void))**
funcion: función de respuesta al evento.
Registra la función de respuesta al evento idle. Este evento se produce cada vez que el sistema no tiene ningún otro evento que atender. En OpenGL se suele utilizar para hacer animaciones. La función que da respuesta al evento debe ser lo más pequeña posible para evitar mermas en la capacidad de interacción de la aplicación.
• **void glutTimerFunc (unsigned int miliseg, void (*funcion) (int valor), int valor)**
miliseg: número de milisegundos del temporizador.
funcion: función de respuesta al evento.
valor: valor que debe utilizarse para llamar a la función de respuesta.
Registra tanto un temporizador, como la función de respuesta al mismo. Debemos indicar el tiempo en milisegundos, un valor de identificación del temporizador y la función que responde al mismo, cuyo único parámetro debe ser el identificador del temporizador.

9. Obtención del estado

GLUT incorpora un conjunto de funciones que devuelven las diferentes variables de estado.

• **int glutGetModifiers (void)**
Devuelve un valor que indica el modificador que se ha tecleado.
Devuelve la tecla modificador que ha sido oprimida: GLUT_ACTIVE_SHIFT, GLUT_ACTIVE_CTRL o GLUT_ACTIVE_ALT.

La librería GLUT permite realizar otras acciones, referentes a la visualización de texto, al manejo del color y al diseño de algunas primitivas geométricas.

Debido a que utilizaremos GLUT como herramienta de apoyo, pero no como objetivo principal del curso, obviaremos esta parte.

Además, OpenGL realiza también el manejo de estos elementos. No obstante, no está demás consultar como referencia el manual de especificación de GLUT.

REFERENCIAS

Kilgard, M. J. *The OpenGL Utiliy Toolkit (GLUT) Programming Interface.* Silicon Graphics, Inc, 1996 [en línea] [OpenGL] http://www.opengl.org/resources/libraries/glut/glut-3.spec.pdf