Una experiencia docente en la enseñanza de OpenGL en la asignatura de Informática Gráfica

José Luis Vicén Cruz
Eduardo Falces Larraga
Rafael Embid Romero

Departamento de Informática. Escuela Universitaria Politécnica de La Almunia
C/ Mayor, s/n (Zaragoza)
E-Mail: {joselo, eduardo} @eupla.unizar.es

Resumen

En este trabajo ofrecemos una visión general de la experiencia que hemos tenido impartiendo la asignatura de Informática Gráfica, en la que se ha introducido por primera vez las librerías de OpenGL como herramienta de desarrollo bajo programación visual en Windows'95. OpenGL es un software desarrollado por Silicon Graphics que rápidamente se ha convertido en el estándar de la industria para aplicaciones de gráficos en 3D de alta calidad. Está disponible para una gran variedad de plataformas y sistemas operativos, incluyendo Windows'95 y Windows NT, IBM OS/2, Open VMS y X Windows. Su facilidad de aprendizaje y su potencia lo convierten en un elemento muy útil a la hora de poner en práctica los conocimientos adquiridos en la parte teórica de la asignatura.

1 Introducción

Con la aparición de los últimos sistemas operativos (Windows'95, Windows NT, Solaris, etc), el desarrollo de programas con lenguajes visuales ha cobrado una importancia fundamental. Atrás quedaron las pantallas en modo texto y la programación directa de las tarjetas de vídeo. Hoy día es preferible el poder aprovechar los interfaces gráficos de los modernos sistemas operativos, utilizando las API's y las extensiones de los mismos para el acceso al hardware gráfico, sin importarnos cuál sea éste. OpenGL nos permite evolucionar hacia estos interfaces de forma rápida y segura, y su cualidad multiplataforma y multisistema es una garantía que no ha pasado desapercibida en la industria, en donde se utiliza masivamente como primera herramienta de programación de gráficos.

2 Organización de la asignatura

La asignatura de Informática Gráfica se ofrece como asignatura optativa en el tercer curso de Ingeniería Técnica en Informática de Sistemas, y se imparte durante tres horas semanales, de las cuales aproximadamente la mitad corresponde a prácticas y ejercicios, y la otra mitad corresponde al contenido teórico propio de la asignatura. Por todo ello, la asignatura posee una carga lectiva de 9 créditos, repartidos a partes iguales entre la teoría y la práctica.

El programa teórico desarrollado en esta asignatura incluye los siguientes apartados:

- Hardware y software gráfico:
  Tecnologías de impresión, tecnologías de pantallas, sistemas de representación por barrido, controladores de vídeo y dispositivos de entrada para la interacción con el operador.

- Geometría y operaciones de visión en 2D:
  Coordenadas homogéneas, transformaciones bidimensionales, composición de transformaciones, sistemas de coordenadas, cambio de sistema, transformación del área de visión y recorte de líneas.

- Geometría y operaciones de visión en 3D:
  Representación de la geometría tridimensional, transformaciones tridimensionales, proyecciones ortográficas y perspectivas, puntos de fuga, volúmenes de visión y recorte de líneas en tres dimensiones.

- Curvas y superficies:

Descripción geométrica de las curvas, implicitación de ecuaciones paramétricas, técnicas de interpolación y aproximación (Bezier, B-Spline), curvas racionales y superficies de forma libre.

- Realismo visual y modelado sólido:

Eliminación de líneas/superficies ocultas, técnicas de visibilidad, sombreado, modelos de color, rendering, representación sólida, bases de la teoría de modelado sólido, representación facetada, espacial, por contornos, operaciones booleanas, cálculos de intersecciones, etc.

En cuanto a la parte práctica de la asignatura, se ha optado por una serie de programas bajo MsDos y Windows'95. Como primer lenguaje de programación utilizamos C++ y C++ Builder, aunque en las prácticas bajo Windows'95 se admite también como herramienta el Borland Delphi.

Las prácticas bajo MsDos se realizan durante el primer parcial, buscando una programación de base a bajo nivel, accediendo directamente a la tarjeta de vídeo (en concreto el chip S3 Trio 64+) y explotando sus características. Se exigen sendos programas de visualización y transformaciones en 2D y 3D, en modo VESA de alta resolución (800x600 ó 1024x768 a 256 colores) con construcción de un interface gráfico adecuado y uso obligatorio del ratón.

Las prácticas bajo Windows'95 se realizan durante el segundo parcial, y es aquí en donde se introducen las librerías de OpenGL para la programación. Se exigen tres escenas de objetos tridimensionales, una de ellas renderizada con texturas y propiedades de materiales asignados. Todas ellas permiten la interacción con el usuario, pudiendo mover el punto de vista, o la situación de los objetos, además de permitir la deformación y transformación de los mismos, y la aplicación de puntos de iluminación. La última de ellas añade la opción de ser una escena animada.

3 ¿Por qué OpenGL ?

OpenGL es una librería diseñada para crear imágenes y animaciones a partir de código, fácilmente portable a otros sistemas y plataformas hardware, y optimizada para una muy rápida ejecución.

Incorpora todos los conceptos vistos en la parte teórica de la asignatura, además de otros avanzados que destacamos a continuación :

- Mapeado de texturas :

Capacidad de aplicar una imagen bitmap a una superficie.

- Z-Búffer :

Capacidad para calcular distancias desde la situación del observador. Permite eliminar automáticamente superficies ocultas.

- Doble Búffer :

Soporte de pantalla virtual para una animación fluída y de calidad.

- Efectos de iluminación y sombreado :

Cálculo del ángulo de incidencia de una fuente de luz sobre las superficies de los objetos. Esto determina el sombreado de los mismos, que además ofrece la posibilidad de escoger el tipo de sombreado que queremos aplicar (facetado, Gouraud, Phong, etc).

- Propiedades de los materiales :

Capacidad para modificar el brillo o reflectividad de los objetos, pudiendo dar efectos de metalizado, transparencia, pulimiento, etc.

- Canales alfa :

Esta es una característica importante, que consiste principalmente en poder dar la propiedad de invisibilidad a un color determinado, por lo que dicho color no interferiría con los colores que se encuentran detrás de él. Es muy útil cuando tenemos objetos que deben dejar ver en parte lo que se encuentre a más distancia, visto desde la posición del observador.

- Matrices de transformación :

Capacidad de poder cambiar la localización, tamaño y perspectiva de un objeto en 2D o 3D en el espacio.

OpenGL es un interface diseñado, mantenido y mejorado por la "OpenGL Architecture Review Board (ARB)", que engloba a importantes compañías de software y hardware, como son Digital Equipment Corporation, Evans and Sutherland, IBM, Intel, Intergraph, Microsoft y Silicon Graphics. Además hay en el mercado una gran variedad de tarjetas de vídeo que interpretan directamente las instrucciones de OpenGL, lo que permite aumentar la rapidez de ejecución entre veinte y cincuenta veces, que para una imagen de menos de 100.000 polígonos crea una ilusión real de movimiento.

4 Arquitectura de OpenGL

Las funciones de OpenGL están incluídas en Windows NT como parte del API, y como DLL's en Windows'95, y constan de unas 150 funciones base, más algunas nuevas bajo Win32 diseñadas para ayudar a la creación de una ventana para un renderizado con OpenGL. Para una aplicación bajo W'95 la estructura es la siguiente :

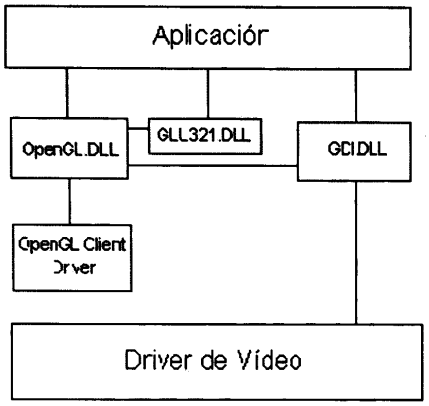

Figura 1 : Arquitectura OpenGL

Una implementación particular depende tanto de los drivers utilizados como de la tarjeta gráfica instalada. Por supuesto, es preciso una buena combinación hardware - software para optimizar el rendimiento de la aplicación.

5 Sintaxis de OpenGL

Las funciones de OpenGL comienzan todas con el prefijo "gl" y un sufijo que indica el tipo de argumentos que pasamos como parámetros, que las distingue del resto de funciones del lenguaje de programación que se utilice.

Por ejemplo, las funciónes que representan vértices son las siguientes :

- glVertex2s(GLshort, GLshort), que indica que se trata de un vértice en 2D, con argumentos de tipo short.
- glVertex3f(GLfloat, GLfloat), que indica que se trata de un vértice en 3D, con argumentos de tipo float.
- glVertex3fv (GLfloat[ ]), que indica que se trata de un vértice en 3D con un argumento que es un vector de float.

Se pueden crear primitivas (objetos complejos a partir de objetos sencillos) utilizando las funciones **"glBegin()"** y **"glEnd()"**. Un sencillo ejemplo de aplicación sería el siguiente :

```
#include <lo_necesario.h>

main( ) {

  InicializarVentanaWindows() ;

  glClearColor (0.0, 0.0, 0.0, 0.0) ;
  glClear( GL_COLOR_BUFFER_BIT) ;
  glColor3f( 1.0, 1.0, 1.0) ;
  glOrtho (0.0, 1.0, 0.0, 1.0, -1.0, 1.0) ;
  glBegin(GL_POLYGON) ;
    glVertex3f( 0.25, 0.25, 0.0) ;
    glVertex3f( 0.75, 0.25, 0.0) ;
    glVertex3f( 0.75, 0.75, 0.0) ;
    glVertex3f( 0.25, 0.75, 0.0) ;
  glEnd() ;
  glFlush() ;

  ActualizarLaVentana_ChequearEventos() ;
```

Este ejemplo dibujaría un cuadrado blanco sobre un fondo negro.

Como se puede observar, salvo la inicialización y actualización de la ventana gráfica, que depende exclusivamente del sistema operativo utilizado, todo lo demás pertenece a OpenGL.

Para generar animaciones se puede seguir un procedimiento como el siguiente :

```
Abrir_Ventana( ) ;
for ( i = 0 ; i < MaxFrame ; i++) {
   Borrar_Ventana( ) ;
   Dibujar_Frame( i ) ;
   Esperar_1/24seg( ) ;
}
```

Para una animación fluída es necesario utilizar el doble búffer, apoyándonos en la función **glSwapBuffers(Display *dpy, Win Window),** que intercambia la pantalla real y la pantalla virtual.

Para el manejo de matrices de transformación tenemos las funciónes **glPushMatrix()** y **glPopMatrix()**, que permiten definir y almacenar matrices, además de las funciones de transformación, como **glRotatef(spin, float, float, float)**, etc, que efectúan directamente las transformaciones sobre los objetos seleccionados.

Por último, se dispone también de algunas primitivas predefinidas, como por ejemplo el cubo y la esfera, tanto en malla de alambre como en modelo sólido, definidas respectivamente por las siguientes funciones :

void glutWireCube(Gldouble size) ;
void glutSolidCube(Gldouble size) ;
void glutWireSphere(Gldouble radio, caras, niveles) ;
void glutSolidSphere(Gldouble size, caras, niveles) ;

Podríamos continuar, pero el objetivo es únicamente dar una idea de cómo trabaja OpenGL, y de las posibilidades que ofrece.

6 Conclusiones

OpenGL es un buen aliado a la hora de practicar con los conceptos más avanzados de la Informática Gráfica, ya que reduce notablemente el tiempo de programación, lo que permite que los ejercicios prácticos sean más complejos y realistas, hecho que motiva al alumno y fomenta su creatividad.

El resultado de esta experiencia nos ha indicado que un porcentaje muy elevado de alumnos realizan los ejercicios con un nivel de perfección por encima del mínimo exigido, investigando ellos mismos algunas de las posibilidades que ofrece OpenGL, y que por problemas de tiempo no se explican en clase.

La tendencia a seguir en los próximos años será la de ir aumentando progresivamente el espacio dedicado a la programación en Windows'95 con OpenGL, (que podría ser complementada con librerías DirectX) e ir reduciendo el tiempo dedicado a la programación bajo MsDos.

Bibliografía

[1] Fostner, Ron. *"Programming with OpenGL"*. Dr.Bobb's Journal (julio 1995)

[2] Neider, Jackie, Tom Davis, and Mason Woo. *"OpenGL programming Guide: The official Guide to Learning OpenGL, Release 1."* Addison-Wesley, 1993 (conocido como "Red Book").

[3] OpenGL Architecture Review Board. *"OpenGL Reference Manual: The Official Reference for OpenGL"* Addison-Wesley, 1992. (conocido como "Blue Book").

[4] Prosise, Jeff. *"Advanced 3-D Graphics for Windows NT 3.5: Introducing the OpenGL Interface, part I"* Microsoft Systems Journal (Octubre 1994).

[5] Crain, Dennis . *"Windows NT OpenGL: Getting Started"* (Abril 1994) Microsoft Developer Network CD. (Este es el primer artículo sobre OpenGL bajo Windows).