Introducción a OpenGL

P. J. Martín, A. Gavilanes
Departamento de Sistemas Informáticos y Computación
Facultad de Informática
Universidad Complutense de Madrid

OpenGL

- OpenGL (de Open Graphics Library) es una especificación. Los fabricantes de hardware desarrollan librerías que implementan las funciones especificadas, y que deben superar tests de conformidad.
- SGI estableció la especificación inicial en 1992.
- Era revisada por el ARB (OpenGL Architectural Review Board), conjunto de empresas (hasta 2003, Microsoft incluida) interesadas en el desarrollo de una API.
- En 2006, el control pasó al Grupo Khronos (Nvidia, Apple, AMD, …), y dentro de él, a OpenGL ARB Working Group.
- Microsoft lanzó en 1995 DirectX, principal competidor de OpenGL.
- Proyecto Fahrenheit (1997).

OpenGL

- OpenGL 2.0 en 2004 incluía soporte para vertex shaders y fragment shaders, en lo que se llamaría GLSL (OpenGL Shading Language).
- OpenGL 3.0 en 2006 permitió la introducción de extensiones.
- En 2007, OpenGL se somete a una limpieza: (Longs Peak) se determina el modelo de deprecación; (Mt. Evans) se introducen geometry shaders.
- OpenGL 4.5 en 2014 es la última versión. Es soportada, entre otras, por las tarjetas Nvidia GeForce 400 y posteriores.
- Desarrollo de bindings para uso en otros lenguajes (JOGL para Java).
- WebGL (para navegadores web), OpenGL ES (para sistemas integrados).

Por qué OpenGL pre-shader

- Física clásica vs Teoría de la relatividad.
- Las partes programables de la tubería gráfica añaden complejidad al código.
- En primeros cursos de informática gráfica es mejor:
  - centrarse más en el fenómeno que en su implementación, facilita el desarrollo posterior de shaders (por ejemplo, en el modelo de iluminación);
  - es mejor entender antes el significado físico de comandos como `glTranslatef()`, que empezar con su significado matemático.
- Código OpenGL heredado.

Por qué OpenGL pre-shader

```
glBegin(GL_POLYGON);
   glVertex3f(20.0, 20.0, 0.0);
   glVertex3f(80.0, 20.0, 0.0);
   glVertex3f(80.0, 80.0, 0.0);
   glVertex3f(20.0, 80.0, 0.0);
glEnd();
```

```
// Drawing routine.
void drawScene(void)
{
   glClear(GL_COLOR_BUFFER_BIT);

   glDrawArrays(GL_TRIANGLE_STRIP, 0, 4);

   glFlush();
}

// Globals
static Vertex squareVertices[] =
{
   { { 20.0, 20.0, 0.0, 1.0 }, { 0.0, 0.0, 0.0, 1.0 } },
   { { 80.0, 20.0, 0.0, 1.0 }, { 0.0, 0.0, 0.0, 1.0 } },
   { { 20.0, 80.0, 0.0, 1.0 }, { 0.0, 0.0, 0.0, 1.0 } },
   { { 80.0, 80.0, 0.0, 1.0 }, { 0.0, 0.0, 0.0, 1.0 } }
};
```

OpenGL

- Es un SW (en forma de librería C) que **permite la comunicación entre el programador y el HW** de la máquina para el diseño de gráficos.
- Es independiente de la plataforma.
- Es independiente del sistema de ventanas. Para usar OpenGL conviene utilizar una API auxiliar para el manejo de ventanas (glut, freeglut, FLTK,…).
- No dispone de comandos de alto nivel para describir escenas 3D. En OpenGL debemos construir el gráfico a partir de sus **primitivas geométricas**:
  - ✓ **Puntos, Líneas y Polígonos**

OpenGL

- OpenGL es una **máquina de estados**. Tenemos una colección de variables de estado que vamos modificando.
- OpenGL **está preparado para trabajar en red**. En un ordenador (servidor) podemos ejecutar nuestro programa OpenGL y en otro (cliente) podemos mostrar el gráfico que hayamos diseñado.
- OpenGL se distribuye con la librería GLU (**OpenGL Utility Library**), que está construida a partir de OpenGL y suministra comandos de alto nivel para el dibujo de gráficos 3D.

GLUT versus Freeglut

- GLUT dejó de tener soporte en 1999
  - GLUT 3.7
- Freeglut es un proyecto activo
  - Freeglut 2.8.1
- Principales diferencias:
  - Comportamiento del bucle principal
  - Cierre de una ventana
  - Cambios en los callbacks soportados
  - Renderizado de cadenas de caracteres

Freeglut

1. Funciones de inicialización
   - **glutInit**: inicializa freeglut
   - **glutInitWindowPosition**, **glutInitWindowSize**: establece la ubicación y el tamaño de las ventanas que se construyan. Se usa como referencia la esquina superior izquierda de la pantalla, las x crecen hacia la derecha, las y hacia abajo
   - **glutInitDisplayMode**: establece qué características de OpenGL se soportan (doble buffer, color RGBA para los píxeles)

2. Funciones para el manejo de ventanas
   - **glutCreateWindow**: construye una ventana
   - **glutDestroyWindow**: destruye una ventana
   - **glutSetOption:** con la constante GLUT_ACTION_ON_WINDOW_CLOSE se establece qué sucede al cerrar una ventana. GLUT_ACTION_CONTINUE_EXECUTION es el valor para continuar la ejecución del resto de las ventanas

3. Funciones para el registro de callbacks
   - ❑ Eventos para la ventana actual
   - **glutDisplayFunc:** registra el método para repintar la ventana
   - **glutReshapeFunc:** registra el método para redimensionar la ventana
   - **glutKeyboardFunc:** registra el método que se ejecuta al pulsar caracteres ASCII
   - **glutSpecialFunc:** registra el método que se ejecuta al pulsar teclas especiales
   - **glutMouseFunc:** registra el método que se ejecuta al pulsar con el ratón

   - ❑ Eventos globales
   - **glutTimerFunc:** registra el método que se ejecutará cuando pase un intervalo de tiempo dado

Freeglut

```
int main(int argc, char *argv[]) {
        …
        int my_window; //my window's identifier

        //Initialization
        glutInitWindowSize(WIDTH, HEIGHT);
        glutInitWindowPosition(140, 140);
        glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
        glutInit(&argc, argv);

        //Window construction
        my_window = glutCreateWindow( "Freeglut 2D-project" );

        //Callback registration
        glutReshapeFunc(resize);
        glutKeyboardFunc(key);
        glutDisplayFunc(display);
        …
```

Freeglut

4. Bucle principal para el procesamiento de eventos
   - **`glutMainLoop`**: comienza el bucle principal
   - **`glutMainLoopEvent`**: ejecuta una iteración del bucle
   - **`glutLeaveMainLoop`**: detiene la ejecución del bucle
5. Funciones de display
   - **`glutPostRedisplay:`** marca la ventana actual exigiendo que vuelva a repintarse
   - **`glutSwapBuffers:`** intercambia los buffers de la ventana actual si se usa doble buffer
6. Funciones para la gestión de menús
   - **`glutCreateMenu:`** crea un menú y registra el método que se ejecuta al seleccionar una de sus entradas
   - **`glutAddMenuEntry:`** inserta una nueva entrada en el menú actual
   - **`glutAddSubMenu:`** inserta un submenú en el menú actual

Freeglut

```
    …
    //OpenGL basic setting
    intitGL();

    //Freeglut's main loop can be stopped executing (**)
    //while ( continue_in_main_loop )
    //  glutMainLoopEvent();

    //Classic glut's main loop can be stopped after X-closing the window,
    //using the following freeglut's setting (*)
    glutSetOption(GLUT_ACTION_ON_WINDOW_CLOSE,
                  GLUT_ACTION_CONTINUE_EXECUTION);

    //Classic glut's main loop can be stopped in freeglut using (*)
    glutMainLoop();

    //We would never reach this point using classic glut
    system("PAUSE");

    return 0;
}
```