PRÁCTICA OPENGL

Manejo de eventos en OpenGL

Gráficos por computador – Grado en Ingeniería de Computadores

Universidad Rey Juan Carlos | 30 de abril de 2022

Pablo Sánchez y Álvaro Martínez

Índice del documento

Contenido

Tabla de ilustraciones

Introducción y ficheros

En esta memoria nos disponemos a explicar el desarrollo de la practica de **manejo de eventos en OpenGL** de la asignatura de *gráficos por Computador* del *Grado de Ingeniería de computadores* de la *Universidad Rey Juan Carlos*.

La práctica consiste en desarrollar una serie de objetos simulando un sistema planetario, además de aceptar cambios en el sistema por eventos creados por el usuario (click en diversas teclas).

El paquete *.zip* entregado contiene los siguientes ficheros:

- "**planet.cpp**": fichero con el código fuente.
- "**planet.exe**": ejecutable del programa.
- Memoria de la práctica.

Para su desarrollo hemos usado el programa de Dev-C++.

Estado inicial del programa

Desde el aula virtual de la universidad se nos proporcionó un fichero con el esqueleto del programa, con el siguiente resultado tras ejecutarlo:

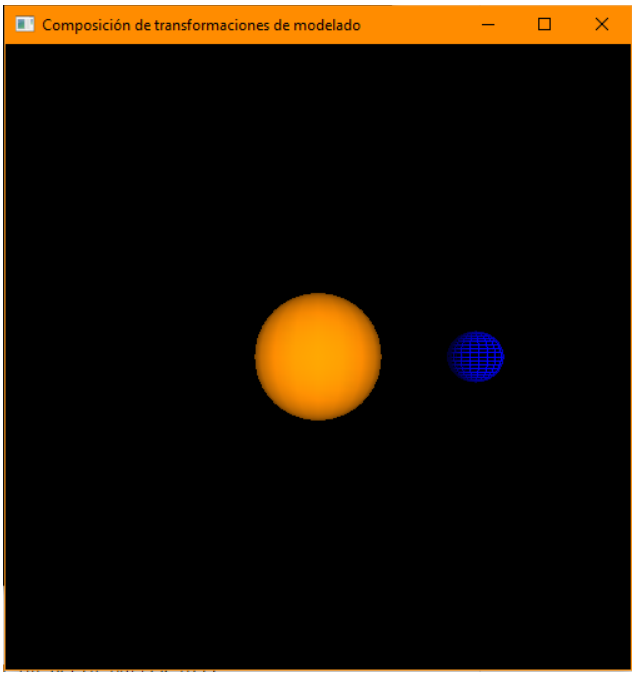

Ilustración 1: programa inicial

Simplemente contenía el sol y el planeta azul, sin ningún movimiento ni eventos por teclado.

Inclusión de satélites

Para poder crear una nueva esfera sin perder la posición sobre el eje de coordenadas necesaria para poder general luego más figuras que orbiten el solo debemos guardar la posición relativa con la sentencias *glPushMatrix()* y *glPopMatrix().*

La primera de estas funciones guarda en la pila de la matriz la posición, mientras que la segunda lo borra. Al ser una pila la anterior posición sería sobre la que nos encontraríamos, esto nos permite dibujar el planeta verde alrededor del sol para luego deshacernos de esa posición y cargar la anterior para así poder generar el siguiente planeta y el satélite.

Para que los cambios en los días y los años muevan los planetas y el satélite alrededor del astro hemos incluido las variables *year* y *day* en las funciones *glRotatef()*.

Obtenemos el resultado deseado añadiendo el siguiente código:

Código de planetas

```
void display()
{
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glEnable(GL_LIGHTING);
	glEnable(GL_DEPTH_TEST);
	glEnable(GL_LIGHT0);

	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	glTranslatef (0.0, 0.0, -5.0);

	glEnable(GL_COLOR_MATERIAL);
	glColor3f(0.99, 0.55, 0.01);

	glPushMatrix();

	glPushMatrix(); // Sol
	glRotatef (90.0, 1.0, 0.0, 0.0);
	glutSolidSphere (1.0, 20, 20);
	glPopMatrix();

	glPushMatrix(); // Planeta verde

	glRotatef(year, 0.0, 0.0, 1.0);
	glTranslatef(0.0, 1.5, 0.0);
	glColor3f(0.0, 1.0, 0.0);
	glutSolidSphere(0.35, 20, 20);

	 glPopMatrix();

	glPushMatrix();

	// Planeta azul
	glRotatef(year, 0.0, 1.0, 0.0);
```

```
        glTranslatef (3.5, 0.0, 0.0);
        glRotatef(day, 0.0, 1.0, 0.0);
        glColor3f (0.0, 0.0, 1.0);
        glutWireSphere (0.4, 20, 20);

        // Satelite rojo del planeta
        glTranslatef(1.0, 0.0, 0.0);
        glRotatef(day, 0.0, 1.0, 0.0);
        glColor3f(1.0, 0.0, 0.0);
        glutWireSphere(0.2, 10, 5);

        glPopMatrix();

        glPopMatrix();

        glDisable(GL_COLOR_MATERIAL);
        glDisable(GL_LIGHT0);
        glDisable(GL_LIGHTING);
        glDisable(GL_DEPTH_TEST);

        glFlush();

        if (rotate) {
          yearAdd();
          dayAdd();
    }
}
```

El uso del último *if* lo explicaremos más adelante.

Con estos cambios obtenemos los siguientes resultados:

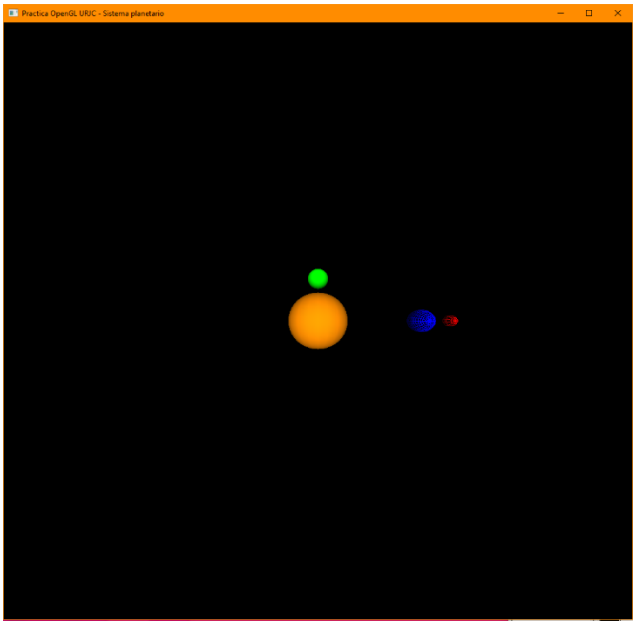

Ilustración 2: inclusión de satélites

Podemos observar que además de haber añadido los satélites, también hemos cambiado la vista del programa debido a que, en ocasiones, el planeta azul se acercaba demasiado a la cámara y esto hacía que su satélite (círculo rojo) se saliese del plano de visión.

También le hemos cambiado el nombre a la ventana de ejecución.

Eventos por teclado

A la hora de realizar los eventos de teclado hemos agregado la función auxiliar *glutSpecialFunc(specialKeyboard).*

Dentro de esta función hemos creado una estructura *switch* para que dependiendo de la tecla que se pulse el sistema solar se mueva de una forma u otra.

Obsérvese en el código que se muestra en la siguiente tabla como los eventos por teclado están a la espera de que se pulse una serie única de teclas, estas son:

- Flechas de teclado.
- Teclas F1 y F2.

Para que el usuario que presione las teclas vea los cambio al final de la estructura hemos incluido la función *glutPostRedisplay(),* esto permite que se actualice las posiciones de los cuerpos que se muestran por pantalla.

Obtenemos el resultado deseado añadiendo el siguiente código:

Código de eventos por teclado

```
void specialKeyboard(int key, int x, int y)
{
	switch (key)
	{
	case GLUT_KEY_UP:
		dayAdd();
		break;
	case GLUT_KEY_DOWN:
		daySubtract();
		break;
	case GLUT_KEY_LEFT:
		yearSubtract();
		daySubtract();
		break;
	case GLUT_KEY_RIGHT:
		yearAdd();
		dayAdd();
		break;
	case GLUT_KEY_F1:
		rotate = true;
		break;
	case GLUT_KEY_F2:
		rotate = false;
```

```
            break;
        default:
            break;
        }
        glutPostRedisplay();
}
```

Para que el programa pueda ejecutarse de manera correcta y volver a "pintar" los objetos si se ha pulsado la tecla F1, hemos incluido una función externa:

Ejecución teclas F1 y F2

```
void timer (int i) {
   glutTimerFunc(100, timer, i);
   glutPostRedisplay();
}
```

Y hemos añadido en el método *main* la siguiente línea:

```
glutTimerFunc(1000, timer, 1);
```

Antes de ejecutar el *loop* principal del programa. También, para saber si la tecla F1 ha sido pulsada, hemos creado una variable estática externa con la que se hacen las comprobaciones.

Finalmente, la condición *if* mencionada anteriormente simplemente comprobará cada *frame* si la variable *rotate* es igual a *true*. De ser así, deberá de ejecutar las funciones de rotar por día y rotar por año.

Conclusiones

Personalmente, nos ha parecido una de las prácticas más interesantes y diferentes hasta ahora de la carrera. Ha sido una práctica bastante interesante que nos ha permitido conocer cómo funciona OpenGL y como se pueden construir entornos 3D a una abstracción de bajo nivel de programación.

Fue todo un reto lograr entender cómo usar las funciones para insertar en las matrices, girar y rotar los objetos declarados.

También, hemos estado atascados algunos días ya que el planeta azul y su satélite rojo no se "escondían" detrás del sol, esto paso ya que estábamos usando primer el código de apoyo subido al aula virtual. En el momento en el que lo cambiamos, todo funcionó correctamente.

Referencias

- Dev-C++: https://es.wikipedia.org/wiki/Dev-C%2B%2B
- OpenGL docs: https://www.opengl.org/
- Temario del aula virtual