

Práctica 2: Transformaciones y eventos. Sistema planetario.

Informática Gráfica

Máster en gráficos, juegos y realidad virtual.

1 Objetivos

En esta práctica se pretenden como objetivos principales los siguientes:

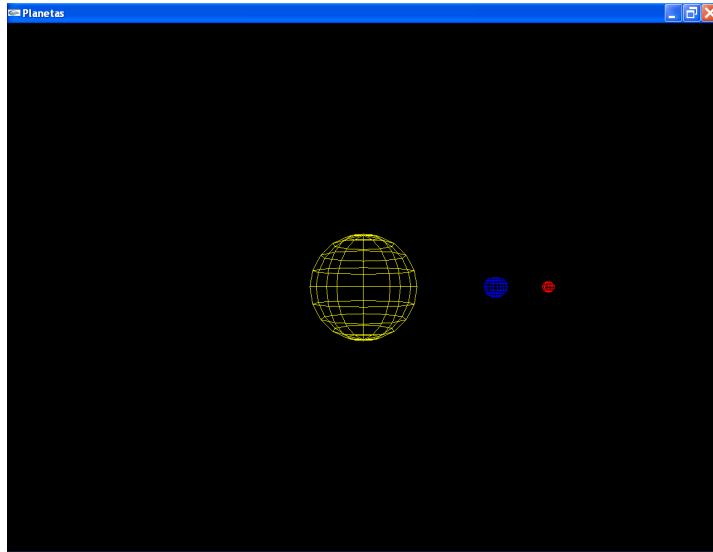
- manejo de las funciones de rotación y traslación en OpenGL.
- manejo de los distintos sistemas de referencia locales y globales.
- manejo de eventos con GLUT.

2 Creación del sistema planetario.

Como resultado final de esta práctica, se deberá crear un sistema planetario formado por tres planetas: Sol, Tierra y Luna. Deben cumplirse los siguientes requisitos:

- En primer lugar, se deberá crear un sistema planetario que conste de tres planetas: Sol, tierra y luna. El Sol deberá estar situado en el centro de la escena. Cada uno de los planetas deberá ser colocado en la escena mediante las matrices adecuadas. Se deberá usar la pila de matrices (funciones `glPushMatrix()` y `glPopMatrix()` para ir almacenando las posiciones relativas de los planetas y posteriormente utilizarlas en las transformaciones).
- Cada planeta será representado por una esfera, de radio y color a elección del alumno. Se desea que el Sol sea el mayor y la Luna el menor de los planetas.
- El observador (cámara) deberá estar situado en una posición perpendicular a la órbita de los planetas.
- Se deberá simular los movimientos de traslación y rotación. La Tierra deberá rotar y a su vez realizar el movimiento de traslación en torno al Sol. La Luna deberá realizar los mismos movimientos respecto a la Tierra. Todos los planetas orbitarán en el mismo plano.
- El movimiento de los planetas se asociará a las teclas "R": movimiento de rotación de la tierra y de la luna, y traslación de la luna en torno a la tierra, y "T": movimiento de traslación de la tierra en torno al sol, en este caso la luna realiza el mismo movimiento de traslación en torno al sol. Cada vez que se pulsen estas teclas debe aumentarse el ángulo (de rotación o traslación) en 5 grados.

El resultado deberá tener un aspecto semejante al de la figura:



3 Ejemplo evento teclado: rotación de un cubo.

El siguiente programa se presenta como un ejemplo de la utilización de eventos de teclado mediante GLUT. Se representará un cubo y pulsando la tecla R el cubo se irá rotando.

```
#include <windows.h>
#include <gl\gl.h>
#include <gl\glu.h>
#include <GL\glut.h>

GLfloat angulo=0.0;

void funcionDePintado (void)
{
    //color de fondo
    glClearColor(0.5, 0.5, 0.5, 1.0);
    glClear (GL_COLOR_BUFFER_BIT);

    glLoadIdentity();
    glTranslatef(0.0, 0.0, -5.0);
    glRotatef(angulo, 0.0, 0.0, 1.0);

    //color de dibujo
    glColor3f (0.0, 1.0, 0.0);
    glutWireCube(2.0);

    glFlush();
}
```

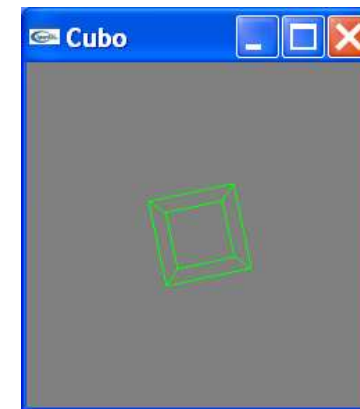
```
void funcionDeReescalado(GLsizei w, GLsizei h)
{
    glMatrixMode (GL_PROJECTION); //activar la
    //matriz de proyeccion
    glLoadIdentity ();
    gluPerspective(90.0, (float)w/(float)h, 2.0, 7.0);

    glViewport (0, 0, w, h);
    glMatrixMode (GL_MODELVIEW);
}

void funcionRotar(unsigned char key, int x, int y)
{
    if(key=='r')
    {
        angulo+=2.0;
        if(angulo>=360) angulo=0.0;
        funcionDePintado();
    }
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(250,250);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Cubo");
    glutReshapeFunc(funcionDeReescalado);
    glutKeyboardFunc (funcionRotar);
    glutDisplayFunc(funcionDePintado);
    glutMainLoop();

    return 0;
}
```



4 Ejemplo animación: rotación de un cubo.

En este ejemplo se crea un cubo igual al del apartado anterior, pero la rotación se realiza de manera continua, creando una animación.

```
#include <windows.h>
#include <gl\gl.h>
#include <gl\glu.h>
#include <gl\glut.h>

GLfloat angulo=0.0;

void funcionDePintado (void)
{
    //color de fondo
    glClearColor(0.5, 0.5, 0.5, 1.0);
    glClear(GL_COLOR_BUFFER_BIT);

    glLoadIdentity();
    glTranslatef(0.0, 0.0, -5.0);
    glRotatef(angulo, 0.0, 0.0, 1.0);

    //color de dibujo
    glColor3f (0.0, 1.0, 0.0);
    glutWireCube(2.0);

    glFlush();
    glutSwapBuffers();
}

void funcionDeReescalado(GLsizei w, GLsizei h)
{
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    gluPerspective(90.0, (float)w/(float)h, 2.0, 7.0);
    glViewport (0, 0, w, h);
    glMatrixMode (GL_MODELVIEW);
    /* restaurar la matriz de modelo-vista como activa*/
}

void funcionIdle(){
    angulo+=0.5;
    if(angulo>=360) angulo=0.0;

    Sleep(50);

    funcionDePintado();
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(250,250);
    glutInitWindowPosition(0,0);
    glutCreateWindow("Cubo");
    glutReshapeFunc(funcionDeReescalado);
    glutIdleFunc(funcionIdle);
    glutDisplayFunc(funcionDePintado);
    glutMainLoop();

    return 0;
}
```