Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №6 по курсу «Компьютерная графика»

Студент: Ф.М. Шавандрин
Преподаватель: Г.С. Филиппов
Группа: М8О-308Б-19
Дата: 20.12.2021
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №6

Создание шейдерных анимационных эффектов в OpenGL.

Задача: Для поверхности, созданной в ЛР№5, обеспечить выполнение следующего шейдерного эффекта:
Вариант 7: Анимация. Координата X изменяется по закону X=sin(t) для всех вершин.

Описание

Для выполнения этой работы я использовал библиотеку PyOpenGL. Это аналог обычного OpenGL, но для языка Python. Для создания анимации в программу я добавил вечный цикл, который изменяет координату x по синусоидальному закону. Этот цикл выполняется в отдельном потоке, который я создаю с помощью стандартной библиотеки threading для Python. В остальном же программа полностью соответствует описанию предыдущей лабораторной работе.

Исходный код

```
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *

import numpy as np
import sys
import threading
import time

# параметры освещения
light_pos = (20, 30, 30)  # положение источника света
light_intensity = 5  # интенсивность света
reflection = 115  # параметр отражения
# фоновое освещение - окружающее освещеие, которое всегда будет придавать объекту некоторый оттенок
ambient = [0.8, 0.0, 0.0, 0.5]
# диффузное освещение - имитирует воздействие на объект направленного источника света
diffuse = [1.0, 0.0, 0.0, light_intensity]
# зеркальный свет - устанавливает цвет блика на объекте
specular = [1.0, 0.0, 0.0, light_intensity]

# вращение
x_rotation = 0
y_rotation = 0
z_rotation = 0

# начальные значения
approximation = 10
a = 1
c = 3
size = 1

def init():
    glClearColor(255, 255, 255, 1.0)  # белый цвет для первоначальной закраски
    glClearDepth(1.0)
    glEnable(GL_DEPTH_TEST)
    glDepthFunc(GL_LEQUAL)
```

```
    glHint(GL_POLYGON_SMOOTH_HINT, GL_NICEST)
    glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST)
    glEnable(GL_NORMALIZE)
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambient)  # определяем текущую модель
освещения
    glEnable(GL_LIGHTING)  # включаем освещение
    glLightModelf(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE)


def draw():
    u = np.linspace(0, 1, approximation)
    t = np.linspace(0, 2 * np.pi, approximation)
    for i in range(approximation - 1):
        for j in range(approximation - 1):
            x1 = a * np.sinh(u[i]) * np.cos(t[j])
            y1 = a * np.sinh(u[i]) * np.sin(t[j])
            z1 = c * np.cosh(u[i])
            x2 = a * np.sinh(u[i]) * np.cos(t[j + 1])
            y2 = a * np.sinh(u[i]) * np.sin(t[j + 1])
            z2 = c * np.cosh(u[i])
            x3 = a * np.sinh(u[i + 1]) * np.cos(t[j + 1])
            y3 = a * np.sinh(u[i + 1]) * np.sin(t[j + 1])
            z3 = c * np.cosh(u[i + 1])
            x4 = a * np.sinh(u[i + 1]) * np.cos(t[j])
            y4 = a * np.sinh(u[i + 1]) * np.sin(t[j])
            z4 = c * np.cosh(u[i + 1])
            glBegin(GL_QUADS)
            glVertex3fv([x1, y1, z1])
            glVertex3fv([x2, y2, z2])
            glVertex3fv([x3, y3, z3])
            glVertex3fv([x4, y4, z4])
            vec1 = [x2 - x1, y2 - y1, z2 - z1]
            vec2 = [x4 - x1, y4 - y1, z4 - z1]
            n = np.cross(vec1, vec2)
            glNormal3fv(n)
            glEnd()


def display():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
    gluLookAt(10, 10, 10, 0, 0, 0, 0, 0, 1)
    glTranslatef(size, size, size)
    init_lighting()
    glRotatef(x_rotation, 1, 0, 0)
    glRotatef(y_rotation, 0, 0, 1)
    glRotatef(z_rotation, 0, 1, 0)

    glPushMatrix()  # сохраняем текущее положение "камеры"
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, diffuse)
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, specular)
    glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, 128 - reflection)
    draw()
    glPopMatrix()  # возвращаем сохраненное положение "камеры"
    glutSwapBuffers()  # выводим все нарисованное в памяти на экран


def init_lighting():
    glEnable(GL_LIGHT0)  # включаем один источник света
       glLightfv(GL_LIGHT0,  GL_POSITION,  light_pos)    #  определяем  положение
источника света

    l_dif = (2.0, 2.0, 3.0, light_intensity)
```

```
    glLightfv(GL_LIGHT0, GL_DIFFUSE, l_dif)
    l_dir = (light_pos[0], light_pos[1], light_pos[2], 1.0)
    glLightfv(GL_LIGHT0, GL_POSITION, l_dir)

    # делаем затухание света
    attenuation = float(101 - light_intensity) / 25.0
        distance  =  np.sqrt(pow(light_pos[0],  2)  +  pow(light_pos[1],  2)  +
pow(light_pos[2], 2))
    constant_attenuation = attenuation / 3.0
    linear_attenuation = attenuation / (3.0 * distance)
    quadratic_attenuation = attenuation / (3.0 * distance * distance)
    glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, constant_attenuation)
    glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, linear_attenuation)
    glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, quadratic_attenuation)


def reshape(width, height):
    glViewport(0, 0, width, height)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluPerspective(60.0, float(width) / float(height), 1.0, 60.0)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()
    gluLookAt(0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1, 0.0)


def specialkeys(key, x, y):
        global  x_rotation,  y_rotation,  z_rotation,  size,  approximation,
light_intensity
    if key == b'w':
        x_rotation += 5  # вращаем на 5 градусов по оси X
    if key == b's':
        x_rotation -= 5  # вращаем на -5 градусов по оси X
    if key == b'a':
        y_rotation += 5  # вращаем на 5 градусов по оси Y
    if key == b'd':
        y_rotation -= 5  # вращаем на -5 градусов по оси Y
    if key == b'q':
        z_rotation += 5  # вращаем на 5 градусов по оси Z
    if key == b'e':
        z_rotation -= 5  # вращаем на -5 градусов по оси Z
    if key == b'=':
        size += 1  # увеличиваем размер на 1
    if key == b'-':
        size -= 1  # уменьшаем размер на 1
    if key == b'm':
        approximation += 1  # увеличиваем число ребер на 1
    if key == b'n':
        approximation -= 1  # уменьшаем число ребер на 1
        approximation = max(3, approximation)
    if key == b'g':
        light_intensity += 5  # увеличиваем интенсивность света на 5
        light_intensity = min(100, light_intensity)
    if key == b'l':
        light_intensity -= 5  # уменьшаем интенсивность света на 5
        light_intensity = max(-100, light_intensity)

    glutPostRedisplay()  # вызываем процедуру перерисовки

def animation_function():
    global x_rotation
    while True:
        t = np.linspace(0, 2 * np.pi, 1000)
        for val in t:
```

```
            x_rotation = np.sin(val) * 50
            glutPostRedisplay()
            time.sleep(0.01)

def main():
      glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)   # используем
двойную буферизацию и формат RGB
    glutInitWindowSize(500, 500)
    glutInitWindowPosition(0, 0)
    glutInit(sys.argv)  # инициализируем opengl
    glutCreateWindow("cg lab 6")
    glutDisplayFunc(display)  # определяем функцию для отрисовки
    glutReshapeFunc(reshape)  # определяем функцию для масштабирования
     glutKeyboardFunc(specialkeys)  # определяем функцию для обработки нажатия
клавиш
    init()
    t = threading.Thread(target=animation_function)
    t.daemon = True
    t.start()
    glutMainLoop()


if __name__ == "__main__":
    print("Rotation:")
    print("OX: W S")
    print("OY: A D")
    print("OZ: Q E")
    print()
    print("Change figure size: - +")
    print("Change approximation: n m")
    print("Change light intensity: l g")
    main()
```

Результат работы

Выводы

Выполнив шестую лабораторную работу по курсу «Компьютерная графика», я познакомился со способом создания простейших анимаций с помощью OpenGL и реализовал одну из таких анимаций для своей фигуры.