Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Московский государственный технический университет имени Н.Э.Баумана (национальный исследовательский университет)»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Джабаров Р.А. ИУ9-32Б

Компьютерная графика.

Лабораторная работа №7

Москва

2023

CODE:

import glfw

from OpenGL.GL import \*

from OpenGL.GLU import \*

from OpenGL.GLUT import \*

import numpy as np

import time

import math

M\_PI = 3.1415926535

delta = 0.05

dt = 0.01

t = 0

velo = -0.1

num = 5000

g = 9.81

pos = 1.

last\_pos = 1.

current\_time = 0

accumulator = 0

tex\_enable = True

shift = 0

WIN\_WIDTH, WIN\_HEIGHT = 800, 800

def main():

global tex, num

if not glfw.init():

return

window = glfw.create\_window(WIN\_WIDTH, WIN\_HEIGHT, "Lab7", None, None)

if not window:

glfw.terminate()

return

glfw.make\_context\_current(window)

glfw.set\_key\_callback(window, key\_callback)

tex = read\_texture("/Users/xb0yx2k20/Documents/photo\_5456474844414725958\_y.jpg")

# make\_list(8)

make\_arrays()

while not glfw.window\_should\_close(window):

display(window)

glfw.destroy\_window(window)

glfw.terminate()

def read\_texture(filename):

from PIL import Image

img = Image.open(filename)

img\_data = np.array(list(img.getdata()), np.int16)

texture\_id = glGenTextures(1)

glBindTexture(GL\_TEXTURE\_2D, glGenTextures(1))

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT)

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT)

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR)

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR)

glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE)

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGBA, img.size[0], img.size[1], 0, GL\_RGBA, GL\_UNSIGNED\_BYTE, img\_data)

return texture\_id

def make\_list(id):

global M\_PI

glNewList(id, GL\_COMPILE)

c = 0

v = - M\_PI/2

u = - M\_PI + 0.1

glColor3fv((0.5, 0.5, 0.5))

glBegin(GL\_LINES)

while (v < M\_PI/2):

while (u < M\_PI):

x = 0.3 \* math.cos(v)\*math.cos(u)

y = 0.7\*math.cos(v)\*math.sin(u)

z = 0.5\*math.sin(v)

glVertex3f(x, y, z)

glColor3fv((0.5, 0.5, 0.5))

x = 0.3\*math.cos(v+delta)\*math.cos(u)

y = 0.7 \* math.cos(v+delta)\*math.sin(u)

z = 0.5\*math.sin(v+delta)

glVertex3f(x, y, z)

glColor3fv((0.5, 0.5, 0.5))

c += 1

if c == 50:

c = 0

u += delta

u = - M\_PI

v += delta

glEnd()

glEndList()

def make\_arrays():

global M\_PI

verticies = []

c = 0

v = - M\_PI/2

u = - M\_PI + 0.1

while (v < M\_PI/2):

while (u < M\_PI):

x = 0.3 \* math.cos(v)\*math.cos(u)

y = 0.7\*math.cos(v)\*math.sin(u)

z = 0.5\*math.sin(v)

x\_ = 0.3\*math.cos(v+delta)\*math.cos(u)

y\_ = 0.7 \* math.cos(v+delta)\*math.sin(u)

z\_ = 0.5\*math.sin(v+delta)

verticies.extend([.0, .0, .5, .5, 0, x, y, z,

.5, 1., .5, .5, 0, x\_, y\_, z\_,

1., 1., .5, .5, 0, 0, 0, 1,])

c += 1

if c == 50:

c = 0

u += delta

u = - M\_PI

v += delta

glInterleavedArrays(GL\_T2F\_C3F\_V3F,

2 \* 4 + 3 \* 4 + 3 \* 4,

np.asarray(verticies, dtype=np.float32))

def display(window):

global pos, last\_pos, shift, current\_time, dt, t, accumulator, velo, tex, num

glEnable(GL\_TEXTURE\_2D)

# if tex\_enable:

# glEnable(GL\_TEXTURE\_2D)

# else:

# glDisable(GL\_TEXTURE\_2D)

glClearColor(0.5, 0.5, 0.5, 0.5)

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)

glLightModeli(GL\_LIGHT\_MODEL\_TWO\_SIDE, GL\_TRUE)

glEnable(GL\_DEPTH\_TEST)

glEnable(GL\_LIGHTING)

glEnable(GL\_LIGHT0)

glDisable(GL\_NORMALIZE)

glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, [1, 1, 1, 1])

glLightfv(GL\_LIGHT0, GL\_POSITION, [0.8, -1., 0., 1])

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, [0.8, 0.7, 0.2])

glLightf(GL\_LIGHT0, GL\_CONSTANT\_ATTENUATION, 0.0)

glLightf(GL\_LIGHT0, GL\_LINEAR\_ATTENUATION, 0.2)

glLightf(GL\_LIGHT0, GL\_QUADRATIC\_ATTENUATION, 0.4)

new\_time = glutGet(GLUT\_ELAPSED\_TIME)

frame\_time = (new\_time - current\_time)

if frame\_time > 0.25:

frame\_time = 0.25

current\_time = new\_time

accumulator += frame\_time

while (accumulator >= dt):

last\_pos = pos

velo -= g \* dt

pos += velo \* dt

if pos <= -0.40:

pos = -0.40

velo = (1 - 10) \* velo / (1 + 10)

t += dt

accumulator -= dt

alpha = accumulator / dt

pos = pos \* alpha + last\_pos \* (1 - alpha)

glPushMatrix()

glTranslatef(0, pos, 0)

glTranslatef(shift, 0, 0)

glRotatef(-90, 1., 0., 0)

glRotatef(45, 0., 1., 1)

glDrawArrays(GL\_POINTS, 0, num \* 3)

glPopMatrix()

glfw.swap\_buffers(window)

glfw.poll\_events()

if np.abs(velo) < 5e-2 and pos == -0.4:

glfw.set\_window\_should\_close(window, True)

def key\_callback(window, key, scancode, action, mods):

global tex\_enable, num, shift

if action == glfw.REPEAT or action == glfw.PRESS:

if key == glfw.KEY\_SPACE:

tex\_enable = not tex\_enable

if key == glfw.KEY\_LEFT:

num -= 1

if num == 0:

num = 1

if key == glfw.KEY\_RIGHT:

num += 1

if key == glfw.KEY\_UP:

shift += 0.1

if key == glfw.KEY\_DOWN:

shift -= 0.1

start\_time = time.perf\_counter()

main()

print('total time:', time.perf\_counter() - start\_time)

I. Теория:

OpenGL:

OpenGL (Open Graphics Library) - это кросс-платформенный интерфейс программирования приложений (API) для работы с графикой.

Он предоставляет функции и возможности для рендеринга трехмерных объектов, настройки освещения, работы с текстурами и другими графическими эффектами.

GLFW:

GLFW (Graphics Library Framework) - это библиотека, предоставляющая функции для создания и управления окнами, обработки событий и ввода.

Она используется в данном коде для создания окна, обработки нажатий клавиш и вызова функций обратного вызова.

GLUT:

GLUT (OpenGL Utility Toolkit) - это набор инструментов для работы с окнами, обработки событий и управления вводом в OpenGL.

В данном коде GLUT используется для создания окна, установки функции обратного вызова для обработки нажатий клавиш и вызова функции glutMainLoop для запуска основного цикла приложения.

Текстуры:

В данном коде используется текстура для отображения изображения на объекте.

Функция read\_texture загружает изображение из файла и создает текстуру с помощью функций glGenTextures, glBindTexture и glTexImage2D.

Текстура привязывается к объекту с помощью функции glEnable(GL\_TEXTURE\_2D).

В функции display текстура включается или выключается в зависимости от значения переменной tex\_enable с помощью функций glEnable(GL\_TEXTURE\_2D) и glDisable(GL\_TEXTURE\_2D).

II. Практическая реализация:

Инициализация окна и OpenGL:

В функции main происходит инициализация GLFW с помощью функции glfw.init.

Создается окно с помощью функции glfw.create\_window и устанавливается текущее контекстное окно с помощью функции glfw.make\_context\_current.

Устанавливается функция обратного вызова glfw.set\_key\_callback для обработки нажатий клавиш.

Загрузка текстуры:

В функции read\_texture происходит загрузка изображения из файла и создание текстуры с помощью функций библиотеки PIL и функций OpenGL.

Текстура привязывается к объекту с помощью функций glTexParameteri, glTexImage2D и glTexEnvf.

Создание списка вершин:

В функции make\_list создается список вершин для отображения геометрии объекта.

С помощью функций glNewList и glEndList создается новый список.

В цикле происходит генерация вершин объекта и их добавление в список с помощью функций glVertex3f и glColor3fv.

Создание массивов вершин:

В функции make\_arrays создается массив вершин для отображения геометрии объекта.

В цикле происходит генерация вершин объекта и их добавление в массив vertices с помощью функции extend.

Основной цикл отображения:

В функции display происходит отрисовка сцены.

Вначале устанавливаются настройки OpenGL: цвет очистки, режимы освещения, материалы и источники света с помощью функций glClearColor, glLightModeli, glEnable, glDisable, glMaterialfv, glLightfv и других.

Затем вычисляется время между кадрами и обновляется положение объекта на основе физического движения с помощью функции display.

В конце происходит отрисовка объекта с помощью функции glDrawArrays, а затем обновление окна с помощью функций glfw.swap\_buffers и glfw.poll\_events.

Если объект остановился на дне (проверяется по значению pos), окно закрывается с помощью функции glfw.set\_window\_should\_close.

Обработка нажатий клавиш:

В функции key\_callback происходит обработка нажатий клавиш.

Если нажата клавиша пробела, включение/выключение текстуры с помощью переменной tex\_enable.

Если нажата клавиша влево, уменьшение значения num (количества сегментов объекта).

Если нажата клавиша вправо, увеличение значения num.

Если нажата клавиша вверх, сдвиг объекта вправо.

Если нажата клавиша вниз, сдвиг объекта влево.

III. Заключение:

Данный код демонстрирует использование OpenGL, GLFW и GLUT для отображения объекта с текстурой и его взаимодействия с помощью клавиш. Он также реализует физическое движение объекта (падение с гравитацией) и обновляет положение объекта на каждом кадре. Код позволяет изменять количество сегментов объекта, включать/выключать текстуру и сдвигать объект влево/вправо.

**Методы оптимизации**:

1.Вместо использования дисплейного списка, применяются команды для генерации фигуры.

2.Фигура представляется в виде массива точек, где каждая точка содержит координаты, цвет и текстурные координаты.

3.Оптимизация осуществляется путем объединения необходимых массивов в один массив с использованием команды glInterleavedArrays вместо отдельных команд glVertexPointer/glNormalPointer/glColorPointer.

4.Используются связанные примитивы для упрощения отрисовки.

5.Нормализация векторов нормалей отключается.

6.Применяются эффективные форматы хранения изображений, такие как GL\_UNSIGNED\_BYTE.

7.Размер окна уменьшается для повышения производительности.

8.Минимизируется количество лишнего кода между командами glBegin/glEnd.

|  |  |
| --- | --- |
| Номер метода | Время в сек |
| - | 7.3 |
| 1 | 1.8 |
| 2 | 1.7 |
| 4 и 5 | 6.5 |
| 7 и 8 | 6.4 |
| 2 ,4, 5, 7 | 1.5 |