Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №4-5 по курсу «Компьютерная графика»

Студент: И. П. Моисеенков
Преподаватель: Г. С. Филиппов
Группа: М8О-308Б-19

Дата: 18.12.2021
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №4-5

Ознакомление с технологией OpenGL.

Задача: Создать графическое приложение с использованием OpenGL. Используя результаты Л.Р.№3, изобразить заданное тело (то же, что и в л.р. №3) с использованием средств OpenGL 2.1. Использовать буфер вершин. Точность аппроксимации тела задается пользователем. Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей. Реализовать простую модель освещения на GLSL. Параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме.

Вариант 11: Прямой усеченный круговой конус.

1 Описание

Для выполнения этой работы я воспользовался библиотекой PyOpenGL. Это аналог обычного OpenGL, но для языка Python.

В программе предусмотрена возможность вращения фигуры по всем осям, изменения размера фигуры, изменения интенсивности источника рассеянного освещения и изменения параметра аппроксимации. Все это выполняется при нажатии определенных клавиш на клавиатуре. Настройки управления выводятся пользователю в консоли при запуске программы.

Здесь, как и в прошлой работе, параметр аппроксимации равен количеству боковых граней многогранника, которым мы аппроксимируем усеченный конус.

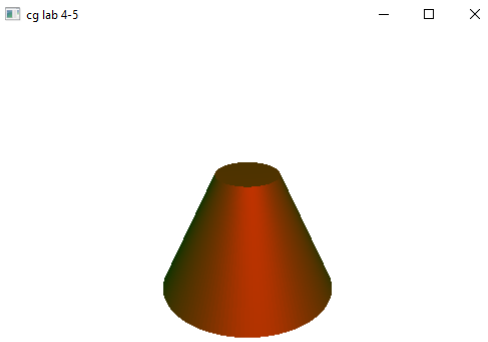
2 Код для отрисовки фигуры и настройки освещения

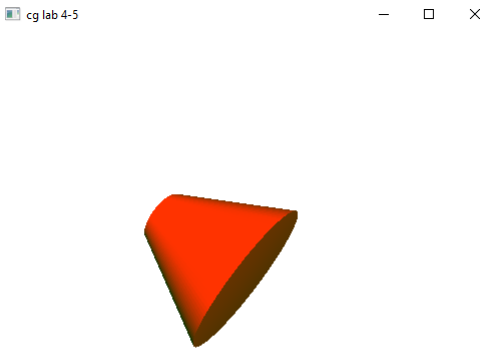
```
def draw():
    global r_lower, r_upper, height, approximation
    v = []
    for i in range(approximation):
        v.append([r_lower * np.cos(2 * np.pi * i / approximation),
                  r_lower * np.sin(2 * np.pi * i / approximation), 0])
        v.append([r_upper * np.cos(2 * np.pi * i / approximation),
                  r_upper * np.sin(2 * np.pi * i / approximation), height])

    sides = [[v[i % (2 * approximation)],
              v[(i + 1) % (2 * approximation)],
```

```
            v[(i + 3) % (2 * approximation)],
            v[(i + 2) % (2 * approximation)]] for i in range(0, approximation * 2 - 1,
                2)]

    top_side = [v[i] for i in range(1, approximation * 2, 2)]
    bottom_side = [v[i] for i in range(0, approximation * 2 - 1, 2)]

    glBegin(GL_QUADS)
    for side in sides:
        n = np.cross(np.array(side[3]) - np.array(side[1]),
                     np.array(side[0]) - np.array(side[1]))
        glNormal3fv(n)
        for vert in side:
            glVertex3fv(vert)
    glEnd()

    glBegin(GL_POLYGON)
    n = np.cross(np.array(top_side[2]) - np.array(top_side[1]),
                 np.array(top_side[0]) - np.array(top_side[1]))
    glNormal3fv(n)
    for vert in top_side:
        glVertex3fv(vert)
    glEnd()

    glBegin(GL_POLYGON)
    n = np.cross(np.array(bottom_side[2]) - np.array(bottom_side[1]),
                 np.array(bottom_side[0]) - np.array(bottom_side[1]))
    glNormal3fv(n)
    for vert in bottom_side:
        glVertex3fv(vert)
    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_rot, 1, 0, 0)
    glRotatef(y_rot, 0, 0, 1)
    glRotatef(z_rot, 0, 1, 0)

    glPushMatrix()
    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()
```

```
60      glPopMatrix()
61      glutSwapBuffers()
62
63
64  def init_lighting():
65      glEnable(GL_LIGHT0)
66      glLightfv(GL_LIGHT0, GL_POSITION, light_pos)
67
68      l_dif = (2.0, 2.0, 3.0, light_intensity)
69      glLightfv(GL_LIGHT0, GL_DIFFUSE, l_dif)
70      l_dir = (light_pos[0], light_pos[1], light_pos[2], 1.0)
71      glLightfv(GL_LIGHT0, GL_POSITION, l_dir)
72
73      attenuation = float(101 - light_intensity) / 25.0
74      distance = np.sqrt(pow(light_pos[0], 2) + pow(light_pos[1], 2) + pow(light_pos[2],
            2))
75      constant_attenuation = attenuation / 3.0
76      linear_attenuation = attenuation / (3.0 * distance)
77      quadratic_attenuation = attenuation / (3.0 * distance * distance)
78      glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, constant_attenuation)
79      glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, linear_attenuation)
80      glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, quadratic_attenuation)
```

3 Результат работы





4 Выводы

Выполнив 4-5 лабораторную работу по компьютерной графике, я познакомился с технологией OpenGL. Это действительно удобная библиотека для отрисовки фигур и гибкой настройки ее параметров - освещения, цвета, свойств материала и т.д.