**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

**отчет**

**по лабораторной работе №6**

**по дисциплине «Компьютерная графика»**

**Тема: Формирования реалистических изображений с использованием простых моделей освещения одним или двумя точечными источниками.**

**Вариант 1**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 7301 |  | Гарцев Е.А.  Литвинов К.Л. |
| Преподаватель |  | Матвеева И. В. |

Санкт-Петербург

2020 г.

**Цель работы**

Научиться формировать реалистические изображения с использованием простых моделей освещения.

**Задание**

Сформировать для наблюдателя изображение освещенных многоугольников или поверхностей, сформированных при выполнении темы 5 без выявления теней. Известными параметрами являются цвет каждого многогранника, интенсивность излучения точечного источника, его положение и положение наблюдателя в пространстве. Обеспечить поворот окрашенной поверхности вокруг осей X и Y или изменение координат источников освещения и наблюдателя.

**Математическая модель**

При выполнении темы 5 был сформирован куб. Для правильного освещения данного многоугольника нам необходимы: положение точечного источника света, положение камеры наблюдателя и нормаль к каждой грани.

Для того, чтобы найти нормаль, необходимо взять 3 точки плоскости, не лежащие на одной прямой. По их координатам построить 2 вектора, которые будут находиться на плоскости: v1 и v2 (например, от 1 точки ко 2 и от 1 точки к 3). После чего найти нормально по формуле:

При этом нормаль должна быть внешней, а не внутренней, иначе будут освещаться внутренние стороны граней.

Зная координаты трёх вершин грани можно определить абсолютное значение нормали, используя выражение:

При вычислении нормалей у них может получаться разный размер, что так же может влиять на характер освещённости.

**Ход работы**

Данная работа выполнялась на языке Python с использованием PyQT для реализации интерфейса, PyOpenGL для удобного отображения сцены, а также системных встроенных библиотек sys, design, math и numpy.

В программе реализовано следующее:

1. Возможность задать положение точечного источника света с помощью координат.
2. Возможность задать положение камеры с помощью координат.

При введении всех необходимых данных пользователю на дисплей будет выводиться куб с корректным освещением всех видимых граней многогранника.

**Выводы**

В процессе выполнения данной лабораторной работы был получен опыт корректного формирования реалистических изображений с помощью простых моделей освещения.

**Листинг кода**

*Файл lab6.py*

import sys

from OpenGL import GL, GLU

from PyQt5.QtWidgets import QApplication, QMainWindow, QPushButton, QMessageBox

from PyQt5.QtGui import QIcon

from PyQt5.QtCore import pyqtSlot

import design

import math

from numpy import matmul

import  cube\_lights

class App(QMainWindow, design.Ui\_MainWindow):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.setupUi(self)

        self.cameraButton.clicked.connect(self.cameraButtonClicked)

        self.lightButton.clicked.connect(self.lightButtonClicked)

    def cameraButtonClicked(self):

        try:

            x = float(self.xCameraLine.text())

            y = float(self.yCameraLine.text())

            z = float(self.zCameraLine.text())

        except ValueError:

            msg = QMessageBox()

            msg.setIcon(QMessageBox.Critical)

            msg.setText("Ошибка")

            msg.setInformativeText('Неправильно введены координаты камеры')

            msg.setWindowTitle("Ошибка")

            msg.exec\_()

            return

        cube\_lights.eye = (x, y, z)

        self.openGLWidget.update()

    def lightButtonClicked(self):

        try:

            x = float(self.xLightLine.text())

            y = float(self.yLightLine.text())

            z = float(self.zLightLine.text())

        except ValueError:

            msg = QMessageBox()

            msg.setIcon(QMessageBox.Critical)

            msg.setText("Ошибка")

            msg.setInformativeText('Неправильно введены координаты света')

            msg.setWindowTitle("Ошибка")

            msg.exec\_()

            return

        cube\_lights.lightPosition = (x, y, z, 1)

        self.openGLWidget.update()

if \_\_name\_\_ == '\_\_main\_\_':

    app = QApplication(sys.argv)

    ex = App()

    ex.show()

    sys.exit(app.exec\_())

*Файл cube\_lights.py*

from OpenGL.GL import \*

from OpenGL.GLU import \*

from OpenGL.GLUT import \*

from PyQt5 import  QtWidgets

import math

from numpy import matmul

window = 0

faces = [[[0, 0, 2], [2, 0, 2], [2, 2, 2], [0, 2, 2]],

         [[0, 0, 2], [0, 2, 2], [0, 2, 0], [0, 0, 0]],

         [[0, 0, 0], [0, 2, 0], [2, 2, 0], [2, 0, 0]],

         [[2, 0, 0], [2, 2, 0], [2, 2, 2], [2, 0, 2]],

         [[2, 0, 0], [2, 0, 2], [0, 0, 2], [0, 0, 0]],

         [[2, 2, 2], [2, 2, 0], [0, 2, 0], [0, 2, 2]]]

eye = (6, 6, 6)

lightPosition = (4.0, 5.0, 5.0, 1.0)

lightColor = (1.0, 1.0, 1.0, 1.0)

point = [0, 0, 0]

ambient = (0.5, 0.5, 0.5, 1)

def getCanonicalForm(pointOne, pointTwo, pointThree):

    x1 = pointOne[0]

    y1 = pointOne[1]

    z1 = pointOne[2]

    x2 = pointTwo[0]

    y2 = pointTwo[1]

    z2 = pointTwo[2]

    x3 = pointThree[0]

    y3 = pointThree[1]

    z3 = pointThree[2]

    result = [0, 0, 0, 0]

    result[0] = (y2-y1)\*(z3-z1)-(z2-z1)\*(y3-y1)

    result[1] = -((x2-x1)\*(z3-z1)-(z2-z1)\*(x3-x1))

    result[2] = (x2-x1)\*(y3-y1)-(y2-y1)\*(x3-x1)

    result[3] = -x1\*result[0] - y1\*result[1] - z1\*result[2]

    return result

def getPointInside():

    global point

    count = 0

    for face in faces:

        count += len(face)

        temp = list(zip(\*face))

        for i in range(3):

            point[i] += sum(temp[i])

    point = [element/count for element in point]

def getT(plane, pointOne, pointTwo):

    a = plane[0]

    b = plane[1]

    c = plane[2]

    d = plane[3]

    x1 = pointOne[0]

    y1 = pointOne[1]

    z1 = pointOne[2]

    x2 = pointTwo[0]

    y2 = pointTwo[1]

    z2 = pointTwo[2]

    t = (-d-a\*x1-b\*y1-c\*z1)/(a\*(x2-x1)+b\*(y2-y1)+c\*(z2-z1))

    return t

def initFigure():

    global faces, point, eye

    glLineWidth(2)

    for face in faces:

        cannonicalFrom = getCanonicalForm(face[0], face[1], face[2])

        t = getT(cannonicalFrom, point, eye)

        if 0 <= t <= 1:

            glBegin(GL\_POLYGON)

            glNormal3f(cannonicalFrom[0], cannonicalFrom[1], cannonicalFrom[2])

            for vertex in face:

                glVertex3dv(vertex)

            glEnd()

class CubeLightsWidget(QtWidgets.QOpenGLWidget):

    def initializeGL(self):

        self.initGL(800, 800)

    def initGL(self, Width, Height):

        glClearColor(1, 0, 1, 1.0)  # Цвет фона

        glClearDepth(1.0)

        glDepthFunc(GL\_LESS)

        glEnable(GL\_DEPTH\_TEST)

        glShadeModel(GL\_SMOOTH)

        glMatrixMode(GL\_PROJECTION)

        gluPerspective(45.0, float(Width) / float(Height), 0.1, 100.0)

        glMatrixMode(GL\_MODELVIEW)

        glEnable(GL\_CULL\_FACE)

        glEnable(GL\_LIGHTING)

        glEnable(GL\_LIGHT0)

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

        glEnable(GL\_NORMALIZE)

    def paintGL(self):

        global ambient, lightPosition, lightColor

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

        glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, ambient)

        glLoadIdentity()

        gluLookAt(eye[0], eye[1], eye[2], 0, 0, 0, 0, 1, 0)

        glLightfv(GL\_LIGHT0, GL\_POSITION, lightPosition)

        glLightfv(GL\_LIGHT0, GL\_SPOT\_DIRECTION, (0.0, 0.0, 1.0))

        glLightfv(GL\_LIGHT0, GL\_SPECULAR, (0.0, 0.0, 0.0, 1.0))

        glLightfv(GL\_LIGHT0, GL\_DIFFUSE, lightColor)

        glMaterialfv(GL\_FRONT, GL\_DIFFUSE, lightColor)

        glMaterialfv(GL\_FRONT, GL\_AMBIENT, (0.7, 0.7, 0.7, 1.0))

        initFigure()

        glPointSize(5)

        glBegin(GL\_POINTS)

        glColor3d(0,1,1)

        glVertex3d(lightPosition[0], lightPosition[1], lightPosition[2])

        glEnd()

*Файл design.py*

# -\*- coding: utf-8 -\*-

# Form implementation generated from reading ui file 'design.ui'

#

# Created by: PyQt5 UI code generator 5.14.1

#

# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_MainWindow(object):

    def setupUi(self, MainWindow):

        MainWindow.setObjectName("MainWindow")

        MainWindow.resize(800, 600)

        self.centralwidget = QtWidgets.QWidget(MainWindow)

        self.centralwidget.setObjectName("centralwidget")

        self.openGLWidget = CubeLightsWidget(self.centralwidget)

        self.openGLWidget.setGeometry(QtCore.QRect(90, 50, 451, 471))

        self.openGLWidget.setObjectName("openGLWidget")

        self.gridLayoutWidget = QtWidgets.QWidget(self.centralwidget)

        self.gridLayoutWidget.setGeometry(QtCore.QRect(576, 70, 219, 171))

        self.gridLayoutWidget.setObjectName("gridLayoutWidget")

        self.gridLayout = QtWidgets.QGridLayout(self.gridLayoutWidget)

        self.gridLayout.setContentsMargins(0, 0, 0, 0)

        self.gridLayout.setObjectName("gridLayout")

        self.zLightLine = QtWidgets.QLineEdit(self.gridLayoutWidget)

        self.zLightLine.setObjectName("zLightLine")

        self.gridLayout.addWidget(self.zLightLine, 4, 2, 1, 1)

        self.xLightLine = QtWidgets.QLineEdit(self.gridLayoutWidget)

        self.xLightLine.setObjectName("xLightLine")

        self.gridLayout.addWidget(self.xLightLine, 4, 0, 1, 1)

        self.xCameraLine = QtWidgets.QLineEdit(self.gridLayoutWidget)

        self.xCameraLine.setObjectName("xCameraLine")

        self.gridLayout.addWidget(self.xCameraLine, 1, 0, 1, 1)

        self.yCameraLine = QtWidgets.QLineEdit(self.gridLayoutWidget)

        self.yCameraLine.setObjectName("yCameraLine")

        self.gridLayout.addWidget(self.yCameraLine, 1, 1, 1, 1)

        self.zCameraLine = QtWidgets.QLineEdit(self.gridLayoutWidget)

        self.zCameraLine.setObjectName("zCameraLine")

        self.gridLayout.addWidget(self.zCameraLine, 1, 2, 1, 1)

        self.label = QtWidgets.QLabel(self.gridLayoutWidget)

        self.label.setObjectName("label")

        self.gridLayout.addWidget(self.label, 0, 0, 1, 3)

        self.lightButton = QtWidgets.QPushButton(self.gridLayoutWidget)

        self.lightButton.setObjectName("lightButton")

        self.gridLayout.addWidget(self.lightButton, 5, 0, 1, 3)

        self.yLightLine = QtWidgets.QLineEdit(self.gridLayoutWidget)

        self.yLightLine.setObjectName("yLightLine")

        self.gridLayout.addWidget(self.yLightLine, 4, 1, 1, 1)

        self.label\_2 = QtWidgets.QLabel(self.gridLayoutWidget)

        self.label\_2.setObjectName("label\_2")

        self.gridLayout.addWidget(self.label\_2, 3, 0, 1, 3)

        self.cameraButton = QtWidgets.QPushButton(self.gridLayoutWidget)

        self.cameraButton.setObjectName("cameraButton")

        self.gridLayout.addWidget(self.cameraButton, 2, 0, 1, 3)

        MainWindow.setCentralWidget(self.centralwidget)

        self.menubar = QtWidgets.QMenuBar(MainWindow)

        self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 22))

        self.menubar.setObjectName("menubar")

        MainWindow.setMenuBar(self.menubar)

        self.statusbar = QtWidgets.QStatusBar(MainWindow)

        self.statusbar.setObjectName("statusbar")

        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)

        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):

        \_translate = QtCore.QCoreApplication.translate

        MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))

        self.label.setText(\_translate("MainWindow", "Введите координаты камеры"))

        self.lightButton.setText(\_translate("MainWindow", "Ввести координаты света"))

        self.label\_2.setText(\_translate("MainWindow", "Введите координаты света"))

        self.cameraButton.setText(\_translate("MainWindow", "Ввести координаты камеры"))

from cube\_lights import CubeLightsWidget