**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

**отчет**

**по лабораторной работе №3**

**по дисциплине «Компьютерная графика»**

**Тема: Формирования различных поверхностей с использованием её пространственного разворота и ортогонального проецирования на плоскость при её визуализации (выводе на экран дисплея).**

**Вариант 3**

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

Санкт-Петербург

2020 г.

**Цель работы**

Научиться формировать различные поверхности с использованием пространственного поворота и корректно проецировать их на плоскости.

**Задание**

Сформировать поверхность Безье для различного задающего многогранника. Обеспечить поворот сформированной поверхности вокруг осей X и Y.

**Математическая модель**

Формируется бикубическая поверхность Безье. Для этого в произвольном порядке заданы 16 точек. Чтобы сформировать плоскость, используется формула:

p(u, v) =

где Pi,j – задающие критические точки, B – многочлены Бернштейна, а – направления параметризации.

Для того, чтобы найти многочлен Бернштейна используют формулу:

где – биномиальный коэффициент, который более привычно видеть в виде , благодаря чему мы пользуемся формулой разложения при помощи факториалов.

После того, как поверхность построена, для её поворота применяем матрицы поворота для каждой точки многоугольника, после чего заново строим поверхность.

**Ход работы**

Данная работа выполнялась на языке Python с использованием PyQT для реализации интерфейса, PyOpenGL для корректного отображения в трёхмерном пространстве, а также системных встроенных библиотек sys, design, numpy, math и random.

В программе реализовано следующее:

1. Задание произвольных 16 точек многогранника.
2. Возможность задать многогранник со случайно сгенерированными точками.
3. Отображение построенной поверхности Безье на дисплей.
4. Возможность повернуть построенную поверхность на заданный угол по осям X и Y.

**Выводы**

В процессе выполнения данной лабораторной работы был получен опыт формирования различных поверхностей с использованием пространственного поворота.

**Листинг кода**

*Файл lab3.py*

**import sys**

**from OpenGL import GL, GLU**

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

**from PyQt5.QtGui import QIcon**

**from PyQt5.QtCore import pyqtSlot**

**import design**

**import math**

**from numpy import matmul**

**import openglwidget as mygl**

**import random**

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

**def \_\_init\_\_(self):**

**super().\_\_init\_\_()**

**self.setupUi(self)**

**self.openGLWidget**

**self.rotateXButton.clicked.connect(self.rotateXPushed)**

**self.rotateYButton.clicked.connect(self.rotateYPushed)**

**self.setButton.clicked.connect(self.setPushed)**

**self.randomButton.clicked.connect(self.randomPushed)**

**self.tableWidget.setRowCount(len(mygl.xyzs)\*4)**

**self.tableWidget.setColumnCount(3)**

**self.tableWidget.setHorizontalHeaderLabels(['x', 'y', 'z'])**

**self.tableWidget.setVerticalHeaderLabels(['P'+str(i+1) for i in range(len(mygl.xyzs)\*4)])**

**k, l = 0, 0**

**for i in range(self.tableWidget.rowCount()):**

**if k % 4 == 0 and k != 0:**

**k = 0**

**if l % 4 == 0 and l != 0:**

**l = 0**

**self.tableWidget.setItem(i, 2, QTableWidgetItem(str(mygl.xyzs[k][l][0])))**

**self.tableWidget.setItem(i, 0, QTableWidgetItem(str(mygl.xyzs[k][l][1])))**

**self.tableWidget.setItem(i, 1, QTableWidgetItem(str(mygl.xyzs[k][l][2])))**

**k += 1**

**l += 1**

**self.tableWidget.resizeColumnsToContents()**

**@pyqtSlot()**

**def rotateXPushed(self):**

**try:**

**teta = float(self.angleXlineEdit.text())**

**except ValueError:**

**msg = QMessageBox()**

**msg.setIcon(QMessageBox.Critical)**

**msg.setText("Ошибка")**

**msg.setInformativeText('Неправильно введён угол поворота')**

**msg.setWindowTitle("Ошибка")**

**msg.exec\_()**

**return**

**teta = teta \* math.pi / 180**

**# Находим нормализированный вектор**

**sqrtSum = math.sqrt((0 - 0)\*\*2 + (0- 0)\*\*2 + (3 - 0)\*\*2)**

**normX = (0 - 0) / sqrtSum**

**normY = (0 - 0) / sqrtSum**

**normZ = (3 - 0) / sqrtSum**

**# Создаём матрицу**

**rotate = []**

**rotate.append([])**

**rotate[0].append(math.cos(teta) + normX\*\*2\*(1-math.cos(teta)))**

**rotate[0].append(normX\*normY\*(1 - math.cos(teta)) - normZ\*math.sin(teta))**

**rotate[0].append(normX\*normZ\*(1 - math.cos(teta)) + normY\*math.sin(teta))**

**rotate.append([])**

**rotate[1].append(normY\*normX\*(1 - math.cos(teta)) + normZ\*math.sin(teta))**

**rotate[1].append(math.cos(teta) + normY\*\*2\*(1 - math.cos(teta)))**

**rotate[1].append(normY\*normZ\*(1-math.cos(teta)) - normX\*math.sin(teta))**

**rotate.append([])**

**rotate[2].append(normZ\*normX\*(1-math.cos(teta)) - normY\*math.sin(teta))**

**rotate[2].append(normZ\*normY\*(1 - math.cos(teta)) + normX\*math.sin(teta))**

**rotate[2].append(math.cos(teta) + normZ\*\*2\*(1-math.cos(teta)))**

**for i in range(len(mygl.xyzs)):**

**mygl.xyzs[i] = matmul(mygl.xyzs[i], rotate)**

**self.openGLWidget.update()**

**@pyqtSlot()**

**def rotateYPushed(self):**

**try:**

**teta = float(self.angleXlineEdit.text())**

**except ValueError:**

**msg = QMessageBox()**

**msg.setIcon(QMessageBox.Critical)**

**msg.setText("Ошибка")**

**msg.setInformativeText('Неправильно введён угол поворота')**

**msg.setWindowTitle("Ошибка")**

**msg.exec\_()**

**return**

**teta = teta \* math.pi / 180**

**# Находим нормализированный вектор**

**sqrtSum = math.sqrt((3 - 0) \*\* 2 + (0 - 0) \*\* 2 + (0 - 0) \*\* 2)**

**normX = (3 - 0) / sqrtSum**

**normY = (0 - 0) / sqrtSum**

**normZ = (0 - 0) / sqrtSum**

**# Создаём матрицу**

**rotate = []**

**rotate.append([])**

**rotate[0].append(math.cos(teta) + normX \*\* 2 \* (1 - math.cos(teta)))**

**rotate[0].append(normX \* normY \* (1 - math.cos(teta)) - normZ \* math.sin(teta))**

**rotate[0].append(normX \* normZ \* (1 - math.cos(teta)) + normY \* math.sin(teta))**

**rotate.append([])**

**rotate[1].append(normY \* normX \* (1 - math.cos(teta)) + normZ \* math.sin(teta))**

**rotate[1].append(math.cos(teta) + normY \*\* 2 \* (1 - math.cos(teta)))**

**rotate[1].append(normY \* normZ \* (1 - math.cos(teta)) - normX \* math.sin(teta))**

**rotate.append([])**

**rotate[2].append(normZ \* normX \* (1 - math.cos(teta)) - normY \* math.sin(teta))**

**rotate[2].append(normZ \* normY \* (1 - math.cos(teta)) + normX \* math.sin(teta))**

**rotate[2].append(math.cos(teta) + normZ \*\* 2 \* (1 - math.cos(teta)))**

**for i in range(len(mygl.xyzs)):**

**mygl.xyzs[i] = matmul(mygl.xyzs[i], rotate)**

**self.openGLWidget.update()**

**@pyqtSlot()**

**def setPushed(self):**

**k, l = 0, 0**

**try:**

**for i in range(self.tableWidget.rowCount()):**

**if k % 4 == 0 and k != 0:**

**k = 0**

**if l % 4 == 0 and l != 0:**

**l = 0**

**mygl.xyzs[k][l][0] = float(self.tableWidget.item(i, 2).text())**

**mygl.xyzs[k][l][1] = float(self.tableWidget.item(i, 0).text())**

**mygl.xyzs[k][l][2] = float(self.tableWidget.item(i, 1).text())**

**k += 1**

**l += 1**

**except Exception:**

**msg = QMessageBox()**

**msg.setIcon(QMessageBox.Critical)**

**msg.setText("Ошибка")**

**msg.setInformativeText('Неправльно заданы координаты многогранника')**

**msg.setWindowTitle("Ошибка")**

**msg.exec\_()**

**return**

**self.openGLWidget.update()**

**def randomPushed(self):**

**for i in range(self.tableWidget.rowCount()):**

**self.tableWidget.setItem(i, 2, QTableWidgetItem(str(random.randint(-15, 15))))**

**self.tableWidget.setItem(i, 0, QTableWidgetItem(str(random.randint(-15, 15))))**

**self.tableWidget.setItem(i, 1, QTableWidgetItem(str(random.randint(-15, 15))))**

**self.tableWidget.resizeColumnsToContents()**

**if \_\_name\_\_ == '\_\_main\_\_':**

**app = QApplication(sys.argv)**

**ex = App()**

**ex.show()**

**sys.exit(app.exec\_())**

*Файл openglwidget.py*

**from OpenGL import GL, GLU**

**from PyQt5 import QtWidgets**

**from bezSurface import \***

**xyzs = [**

**[**

**[-15, 0, 15],**

**[-15, 5, 5],**

**[-15, 5, -5],**

**[-15, 0, -15]**

**],**

**[**

**[-5, 5, 15],**

**[-5, 5, 5],**

**[-5, 5, -5],**

**[-5, 5, -15]**

**],**

**[**

**[5, 5, 15],**

**[5, 5, 5],**

**[5, 5, -5],**

**[5, 5, -15]**

**],**

**[**

**[15, 0, 15],**

**[15, 5, 5],**

**[15, 5, -5],**

**[15, 0, -15]**

**],**

**]**

**class OpenGLWidget(QtWidgets.QOpenGLWidget):**

**def initializeGL(self):**

**GL.glClearDepth(1.0)**

**GL.glDepthFunc(GL.GL\_LESS)**

**GL.glEnable(GL.GL\_DEPTH\_TEST)**

**GL.glMatrixMode(GL.GL\_PROJECTION)**

**GL.glLoadIdentity()**

**GLU.gluPerspective(45, 800/600, 0.1, 400.0)**

**GL.glTranslatef(-3.0, -3.0, -80)**

**GL.glMatrixMode(GL.GL\_MODELVIEW)**

**GL.glEnable(GL.GL\_COLOR\_MATERIAL)**

**def paintGL(self):**

**myBez = Make\_bez(xyzs)**

**us = [i / 100 for i in range(0, 101, 4)]**

**vs = [i / 100 for i in range(0, 101, 4)]**

**bezierPoints = myBez.bezierSurface(us, vs)**

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

**GL.glLoadIdentity()**

**GLU.gluLookAt(1, 1, 3, -2.5, 0, -3,0,1.5,0)**

**GL.glColor3f(1,1,1)**

**# Рисуем оси**

**GL.glBegin(GL.GL\_LINES)**

**GL.glVertex3f(0, 0, 0)**

**GL.glVertex3f(0, 0, 40)**

**GL.glEnd()**

**GL.glBegin(GL.GL\_LINES)**

**GL.glVertex3f(0, 0, 0)**

**GL.glVertex3f(40, 0, 0)**

**GL.glEnd()**

**GL.glBegin(GL.GL\_LINES)**

**GL.glVertex3f(0, 0, 0)**

**GL.glVertex3f(0, 40, 0)**

**GL.glEnd()**

**# Рисуем горизонтальные линии многоугольника**

**GL.glColor(1, 0, 0)**

**GL.glBegin(GL.GL\_LINE\_STRIP)**

**for i in range(len(xyzs)):**

**for j in range(len(xyzs)):**

**GL.glVertex3f(xyzs[i][j][0], xyzs[i][j][1], xyzs[i][j][2])**

**GL.glEnd()**

**# Рисуем вертикальные линии многоугольника**

**GL.glBegin(GL.GL\_LINE\_STRIP)**

**for i in range(len(xyzs)):**

**for j in range(len(xyzs)):**

**GL.glVertex3f(xyzs[j][i][0], xyzs[j][i][1], xyzs[j][i][2])**

**GL.glEnd()**

**# Рисуем горизонтальные линии поверхности Безье**

**GL.glColor(0, 1, 0)**

**GL.glBegin(GL.GL\_LINE\_STRIP)**

**for i in range(len(bezierPoints)):**

**for j in range(len(bezierPoints)):**

**if bezierPoints[i][j] == 0:**

**continue**

**GL.glVertex3f(bezierPoints[i][j][0], bezierPoints[i][j][1], bezierPoints[i][j][2])**

**GL.glEnd()**

**# Рисуем вертикальные линии поверхности Безье**

**GL.glBegin(GL.GL\_LINE\_STRIP)**

**for i in range(len(bezierPoints)):**

**for j in range(len(bezierPoints)):**

**GL.glVertex3f(bezierPoints[j][i][0], bezierPoints[j][i][1], bezierPoints[j][i][2])**

**GL.glEnd()**

*Файл bezSurface.py*

**from math import factorial**

**class Make\_bez:**

**def \_\_init\_\_(self, points):**

**self.points = points**

**self.n = len(points) - 1**

**def bezierSurface(self, us, vs):**

**result = [[0 for i in range(len(us))] for i in range(len(vs))]**

**#result = []**

**for u in us:**

**#resultX = []**

**for v in vs:**

**sumX = 0**

**sumY = 0**

**sumZ = 0**

**for i in range(self.n+1):**

**# Finding bernstein for u point**

**binomialU = factorial(self.n) / (factorial(i)\*factorial(self.n-i))**

**bernsteinU = binomialU \* u\*\*i \* (1 - u)\*\*(self.n-i)**

**for j in range(self.n+1):**

**# Finding bernstein for v point**

**binomialV = factorial(self.n) / (factorial(j)\*factorial(self.n-j))**

**bernsteinV = binomialV \* v\*\*j \* (1 - v)\*\*(self.n-j)**

**# Changing x point**

**sumX += self.points[i][j][0] \* bernsteinU \* bernsteinV**

**# Changing y point**

**sumY += self.points[i][j][1] \* bernsteinU \* bernsteinV**

**# Changing z point**

**sumZ += self.points[i][j][2] \* bernsteinU \* bernsteinV**

**result[round(u\*(len(us)-1))][round(v\*(len(vs)-1))] = [sumX, sumY, sumZ]**

**#result.append([sumX, sumY, sumZ])**

**return result**

*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(1400, 900)**

**self.centralwidget = QtWidgets.QWidget(MainWindow)**

**self.centralwidget.setObjectName("centralwidget")**

**self.openGLWidget = OpenGLWidget(self.centralwidget)**

**self.openGLWidget.setGeometry(QtCore.QRect(10, 0, 800, 600))**

**self.openGLWidget.setMinimumSize(QtCore.QSize(581, 461))**

**self.openGLWidget.setObjectName("openGLWidget")**

**self.tableWidget = QtWidgets.QTableWidget(self.centralwidget)**

**self.tableWidget.setGeometry(QtCore.QRect(820, 0, 481, 301))**

**self.tableWidget.setObjectName("tableWidget")**

**self.tableWidget.setColumnCount(0)**

**self.tableWidget.setRowCount(0)**

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

**self.gridLayoutWidget.setGeometry(QtCore.QRect(830, 360, 492, 84))**

**self.gridLayoutWidget.setObjectName("gridLayoutWidget")**

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

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

**self.gridLayout.setObjectName("gridLayout")**

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

**self.label.setObjectName("label")**

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

**self.angleYlineEdit = QtWidgets.QLineEdit(self.gridLayoutWidget)**

**self.angleYlineEdit.setObjectName("angleYlineEdit")**

**self.gridLayout.addWidget(self.angleYlineEdit, 1, 1, 1, 1)**

**self.angleXlineEdit = QtWidgets.QLineEdit(self.gridLayoutWidget)**

**self.angleXlineEdit.setObjectName("angleXlineEdit")**

**self.gridLayout.addWidget(self.angleXlineEdit, 1, 0, 1, 1)**

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

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

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

**self.rotateXButton = QtWidgets.QPushButton(self.gridLayoutWidget)**

**self.rotateXButton.setObjectName("rotateXButton")**

**self.gridLayout.addWidget(self.rotateXButton, 2, 0, 1, 1)**

**self.rotateYButton = QtWidgets.QPushButton(self.gridLayoutWidget)**

**self.rotateYButton.setObjectName("rotateYButton")**

**self.gridLayout.addWidget(self.rotateYButton, 2, 1, 1, 1)**

**self.setButton = QtWidgets.QPushButton(self.centralwidget)**

**self.setButton.setGeometry(QtCore.QRect(810, 310, 245, 32))**

**self.setButton.setObjectName("setButton")**

**self.randomButton = QtWidgets.QPushButton(self.centralwidget)**

**self.randomButton.setGeometry(QtCore.QRect(1070, 310, 245, 32))**

**self.randomButton.setObjectName("randomButton")**

**MainWindow.setCentralWidget(self.centralwidget)**

**self.menubar = QtWidgets.QMenuBar(MainWindow)**

**self.menubar.setGeometry(QtCore.QRect(0, 0, 1400, 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 угол поворота вокруг оси X"))**

**self.label\_2.setText(\_translate("MainWindow", "Введите угол поворота вокруг оси Y"))**

**self.rotateXButton.setText(\_translate("MainWindow", "Повернуть по X"))**

**self.rotateYButton.setText(\_translate("MainWindow", "Повернуть по Y"))**

**self.setButton.setText(\_translate("MainWindow", "Задать многограник"))**

**self.randomButton.setText(\_translate("MainWindow", "Случайный многограник"))**

**from openglwidget import OpenGLWidget**