МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Компьютерная графика»

Тема: «Кубические сплайн»

Студентка гр. 6382 _____ Черкасова Е.И.

Студент гр. 6303 _____ Доброхвалов М.О.

Преподаватель _____ Герасимова Т.В.

Санкт-Петербург

2019

Задание.

Интерполяционный многочлен Лагранжа по 5 точкам.

Общие сведения.

Сплайны - это гладкие (имеющие несколько непрерывных производных) кусочно-полиномиальные функции, которые могут быть использованы для представления функций, заданных большим количеством значений и для которых неприменима аппроксимация одним полиномом. Так как сплайны гладки, экономичны и легки в работе, они используются при построении произвольных функций для:

- моделирования кривых;
- аппроксимации данных с помощью кривых;
- выполнения функциональных аппроксимаций;
- решения функциональных уравнений.

Здесь кратко излагаются некоторые основные положения и использования сплайнов в 3D графики.

Важным их свойством является простота вычислений. На практике часто используют сплайны вида полиномов третьей степени. С их помощью довольно удобно проводить кривые, которые интуитивно соответствуют человеческому субъективному понятию гладкости.

Интерполяцио́нный многочле́н Лагра́нжа — многочлен минимальной степени, принимающий данные значения в данном наборе точек. Для $n+1$ пар чисел (x_0, y_0), (x_1, y_1),…, (x_n, y_n), где все x_j различны, существует единственный многочлен $L(x)$ степени не более n, для которого $L(x_j) = y_j$.

В простейшем случае ($n=1$)— это линейный многочлен, график которого — прямая, проходящая через две заданные точки.

$$L(x) = \sum_{i=0}^{n} y_i l_i(x)$$

,

где li

$$l_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} = \frac{x - x_0}{x_i - x_0} \cdots \frac{x - x_{i-1}}{x_i - x_{i-1}} \cdot \frac{x - x_{i+1}}{x_i - x_{i+1}} \cdots \frac{x - x_n}{x_i - x_n}$$

Ход работы.

Функция lagrange используется из библиотеки scipy.interpolate, она позволяет найти коэффициенты интерполяционного многочлена Лагранжа.

```
def get_polynomial(self):
    points_x = [point[0] for point in self.int_points]
    points_y = [point[1] for point in self.int_points]
    return lagrange(points_x, points_y)
```

Далее выполняется функция расчета точек полученной функции, и ее отрисовка по массиву рассчитанных точек. Запускается таймер обновления экрана и обработчик событий мыши. Когда пользователь кликает на точку, проверяется, «точность» нажатия и точка начинает движение. Когда пользователь отпускает кнопку мыши точка устанавливается на данную позицию и расчет происходит заново.

Функции обработчики мыши:

```
def mousePressEvent(self, event):
    on_pressed = event.pos()
    for index, point in enumerate(self.int_points):
        if not self.in_move and abs(point[0]-on_pressed.x()) < 5 and \
                abs(point[1]-on_pressed.y()) < 5:
            self.focus_point = index
            self.in_move = True


def mouseMoveEvent(self, event):
    if self.in_move:
        new_position = event.pos()
        self.int_points[self.focus_point] = [new_position.x(), new_position.y()]
        self.polinomial = self.get_polynomial()
        self.update()


def mouseReleaseEvent(self, *args, **kwargs):
    self.in_move = False
    self.focus_point = -1
```

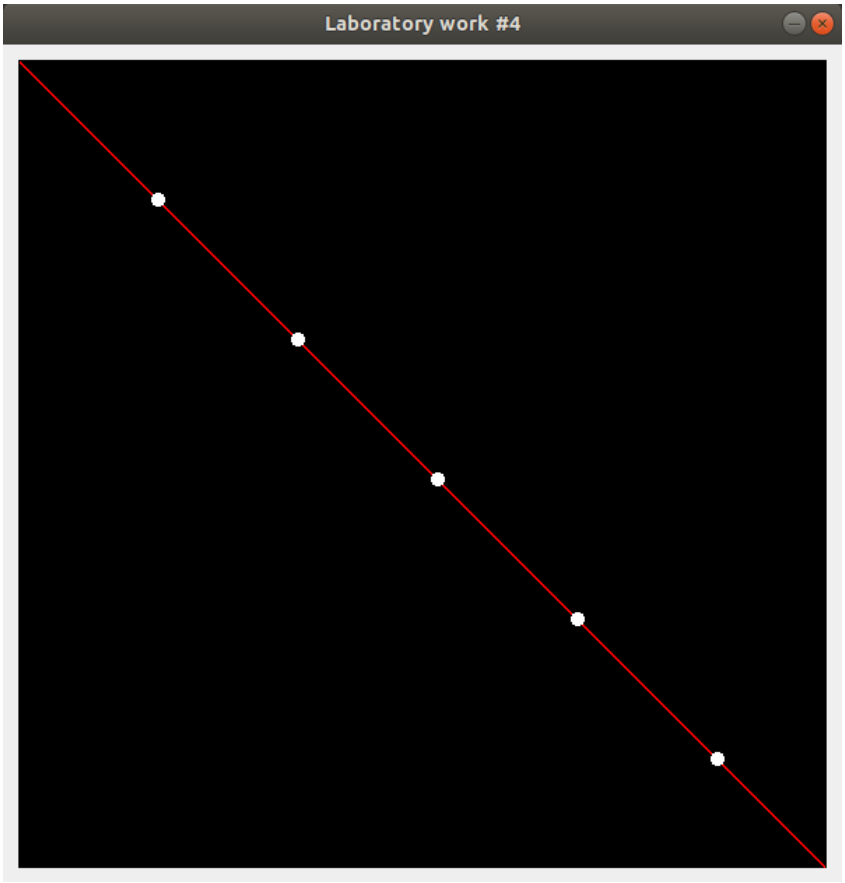
Рисование точек происходит на интервале [100; 500].

Модуль **controller.py,** в котором находится основная логика программы, имеет следующий вид:

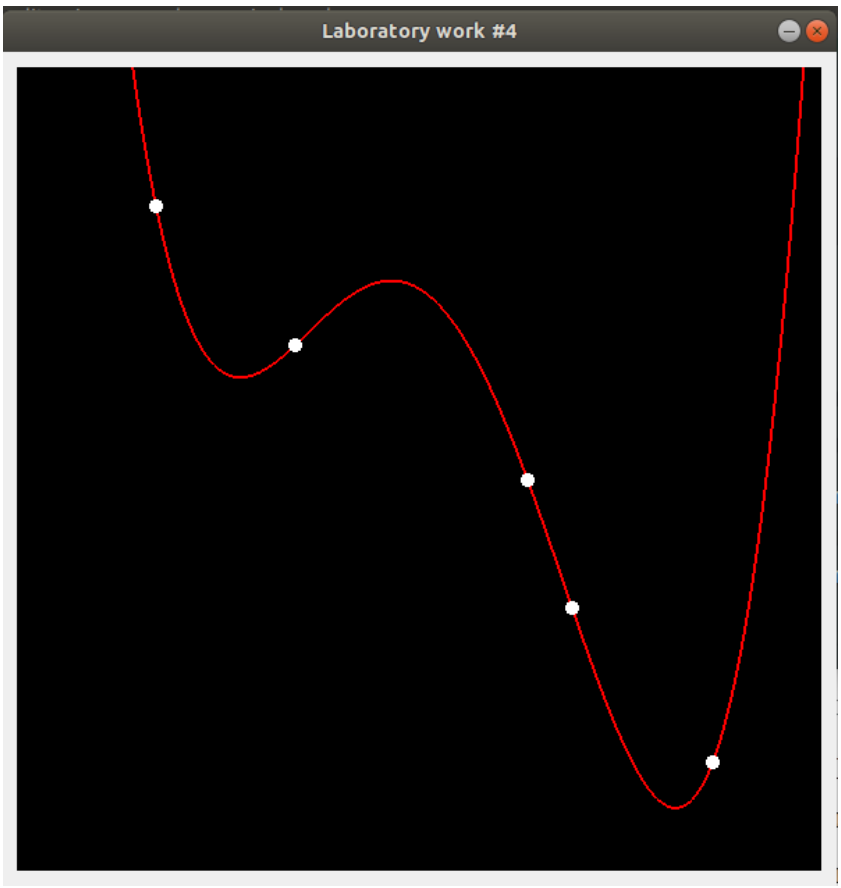
```
from OpenGL.GL import *

from PyQt5.QtOpenGL import QGLWidget
from scipy.interpolate import lagrange

class GLWidget(QGLWidget):

    def __init__(self, parent):
        super(GLWidget, self).__init__(parent)
        self.in_move = False
        self.focus_point = -1
        self.int_points = [[100., 100.], [200., 200.], [300., 300.], [400., 400.], [500., 500.]]
        self.polinomial = self.get_polynomial()

    def initializeGL(self):
        """It is called once before the first call to paintGL() or resizeGL(),
        and then once whenever the widget has been assigned a new QGLContext """
        glMatrixMode(GL_PROJECTION)
        glLoadIdentity()
        glOrtho(0, self.width(), self.height(), 0, -1, 1)
        glMatrixMode(GL_MODELVIEW)

    def paintGL(self):
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT)
        glLoadIdentity()
        glLineWidth(2)
        interp_x = [value for value in range(1, self.width())]
        glColor3f(1,0,0)
        glBegin(GL_LINE_STRIP)
        for i in range(len(interp_x)):
            glVertex2f(interp_x[i], self.polinomial(interp_x[i]))
        glEnd()
        glPointSize(10)
        glEnable(GL_POINT_SMOOTH)
        glBegin(GL_POINTS)
        glColor3f(1,1,1)
        for point in self.int_points:
            glVertex2f(*point)
        glEnd()

    def resizeGL(self, w, h):
        glViewport(0, 0, w, h)
        glOrtho(0, w, 0, h, -1.0, 1.0)
        glLoadIdentity()

    def mousePressEvent(self, event):
        on_pressed = event.pos()
        for index, point in enumerate(self.int_points):
            if not self.in_move and abs(point[0]-on_pressed.x()) < 5 and \
                    abs(point[1]-on_pressed.y()) < 5:
                self.focus_point = index
                self.in_move = True
```

```
def mouseMoveEvent(self, event):
    if self.in_move:
        new_position = event.pos()
        self.int_points[self.focus_point] = [new_position.x(), new_position.y()]
        self.polinomial = self.get_polynomial()
        self.update()


def mouseReleaseEvent(self, *args, **kwargs):
    self.in_move = False
    self.focus_point = -1


def get_polynomial(self):
    points_x = [point[0] for point in self.int_points]
    points_y = [point[1] for point in self.int_points]
    return lagrange(points_x, points_y)
```

Тестирование.



После перемещения точки пользователем:



Вывод.

В процессе выполнения лабораторной работы была разработана программа, строящая интерполяционный многочлен Лагранжа. Написана на языке программирования Python, протестирована в операционной системе Ubuntu. При выполнении работы были приобретены навыки работы с графической библиотекой OpenGL.