**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 сплайны

|  |  |  |
| --- | --- | --- |
| Студенты гр. 6304 |  | Корытов П.В.  Пискунов Я.А. |
| Преподаватель |  | Герасимова Т.В. |

Санкт-Петербург

2018

**Цель работы.**

Реализовать интерактивное приложение, отображающее заданные полиномиальные кривые

При этом для кривых, состоящих из нескольких сегментов, должно быть обеспечено свойство непрерывной кривизны. Программа должна позволять пользователю: интерактивно менять положение контрольных точек, касательных, натяжений.

Вариант 23. Кривая Безье 3-й степени (выполнить два варианта – с использованием стандартной функции рисования кривой и без).

В отчете д.б. представлена реализуемая в программе формула, описан алгоритм построения и показаны основные характеристики кривой.

**Выполнение работы**

Инструменты разработки остались прежними:

* Для разработки использована IDE PyCharm
* Используемый ЯП – Python, интерпретатор версии 3.6.7
* Для разработки интерфейса использован байдинг фреймворка Qt на Python, поддерживаемый сообществом – PyQt5
* Формы разрабатываются в Qt Designer и конвертируются в python-код с помощью утилиты pyuic5
* Для работы с OpenGL используется библиотека pyopengl
* Разработка проводится как на платформе Linux, так и на Windows

Основа программы также осталась без изменений.

Интерактивность виджета OpenGL обеспечена заданием собственных обработчиков событий для нажатия, движения мыши и для отпускания мыши:

**self**.openGLWidget.mouseMoveEvent = **self**.onOpenGLWidgetMouseMoved

**self**.openGLWidget.mousePressEvent = **self**.onOpenGLWidgetMousePressed

**self**.openGLWidget.mouseReleaseEvent = **self**.onOpenGLWidgetMouseReleased

При создании виджета создается набор элементов управления (класс ControlElement). В классе содержатся координаты и методы работы с координатами – изменение, вычисление попадания данных координат в фигуру и т.п.

При нажатии все элементы, которые были под мышкой, запоминаются:

**def** onOpenGLWidgetMousePressed(**self**, event):

width, height = **self**.openGLWidget.width(), **self**.openGLWidget.height()

point = np.array((event.x() / width, 1 - (event.y() / height)))

**for** control **in** **self**.controls:

**if** control.has\_point(\*point):

**self**.move\_controls.append((control, point - control.center()))

При движении мышки – соответственно меняются координаты этих элементов:

**def** onOpenGLWidgetMouseMoved(**self**, event):

**if** **len**(**self**.move\_controls) == 0:

**return**

width, height = **self**.openGLWidget.width(), **self**.openGLWidget.height()

point = (event.x() / width, 1 - (event.y() / height))

**for** control, delta **in** **self**.move\_controls:

control.set\_center(\*(point - delta))

**self**.openGLWidget.update()

А при отпускании – очистка массива move\_controls

def onOpenGLWidgetMouseReleased(self, event):

self.move\_controls.clear()

Сама кривая рисуется следующим образом:

1. Инициализация. На вход подается N+1 точек. Через эти точки строятся линии. Получается N наборов точек.
2. Происходит одновременный перебор N набора точек. Т.е. на каждом шаге имеется N точек с из наборов с одинаковым индексом.

Берется среднее от каждой пары соседних точек (т.е. 0 и 1, 1 и 2, …, N-1 и N). Получается N-1 новых точек

Из новых точек создается N-1 новых наборов точек.

1. Если N=1, то отобразить набор точек, иначе повторить шаг 2.

Реализация этого алгоритма в программе:

**def** paintBezierCommon(**self**, points):

lines = np.array([[(p2 - p1) / 100 \* i + p1 **for** i **in** **range**(100)]

**for** p1, p2 **in** **zip**(points, points[1:])])

**while** **len**(lines) > 1:

lines = np.array([[

(p2 - p1) / 100 \* i + p1 **for** i, p1, p2 **in** **zip**(**range**(100), line1, line2)

]

for **line1**, **line2** in **zip**(**lines**, **lines**[1:])

])

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

[GL.glVertex2d(\*p) for p in lines[0]]

GL.glEnd()

GL.glFinish()

**Тестирование**

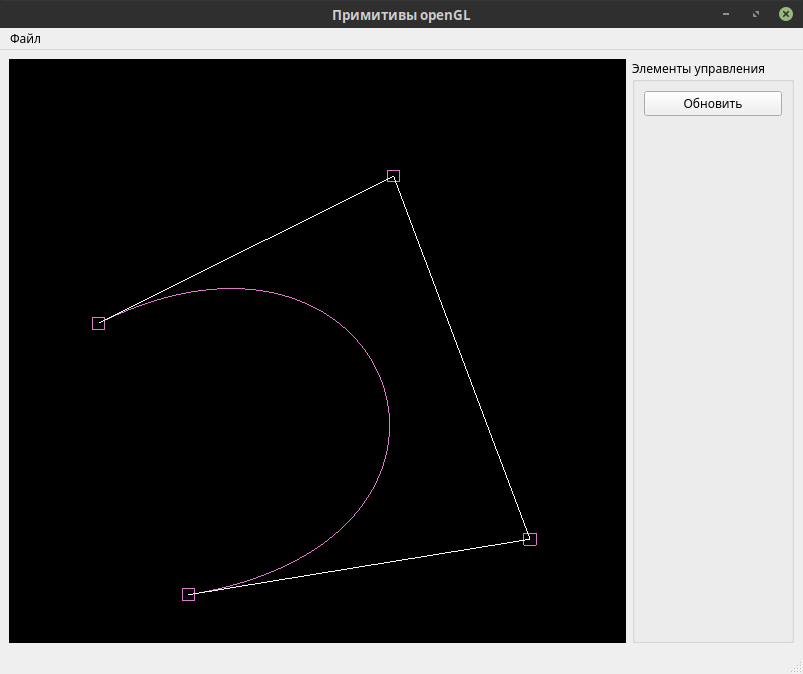


Рисунок 1 - Построенная кривая

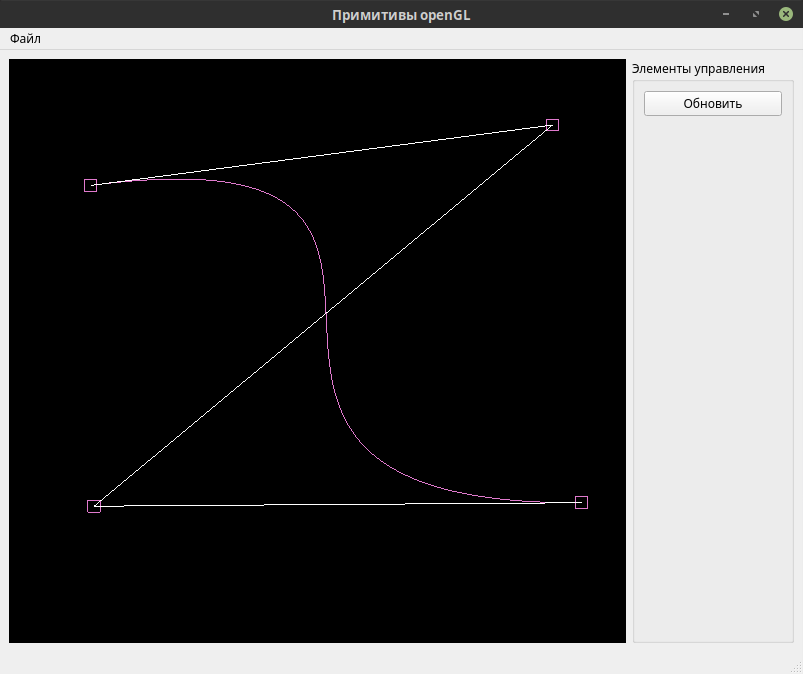


Рисунок 2 - изменение положения точек

**Выводы**

Исследованы возможности обеспечения интерактивности виджетов OpenGL и рисования кубический сплайнов с помощью этого средства.

**ПРИЛОЖЕНИЕ А**

**Код программы**

import random

import numpy as np

from PyQt5.QtWidgets import QMainWindow, QApplication

from OpenGL import GL, GLU, GLUT

import logging

from ui.bezier import Ui\_BezierWindow

from main import cart2pol, pol2cart, random\_rgb

class ControlElement:

p = 0.02

def \_\_init\_\_(self, x, y, color=None):

self.x, self.y = x, y

self.color = random\_rgb() if color is None else color

def get\_points(self):

yield self.x - self.p / 2, self.y - self.p / 2

yield self.x + self.p / 2, self.y - self.p / 2

yield self.x + self.p / 2, self.y + self.p / 2

yield self.x - self.p / 2, self.y + self.p / 2

def set\_center(self, x, y):

self.x, self.y = x, y

def center(self):

return np.array((self.x, self.y))

def has\_point(self, x, y):

p = np.array((x, y))

p\_s = np.array([p for p in self.get\_points()])

return np.all((

np.all(p\_s[0] <= p),

np.all(p\_s[2] >= p)

))

def paint(self):

GL.glBegin(GL.GL\_LINE\_LOOP)

GL.glColor4d(\*self.color)

for point in self.get\_points():

GL.glVertex2d(\*point)

GL.glEnd()

# noinspection PyPep8Naming

class MainWindow(QMainWindow, Ui\_BezierWindow):

controls = []

move\_controls = []

def \_\_init\_\_(self, parent=None):

super().\_\_init\_\_(parent)

self.setupUi(self)

self.openGLWidget.initializeGL()

self.openGLWidget.paintGL = self.paintGL

self.openGLWidget.mouseMoveEvent = self.onOpenGLWidgetMouseMoved

self.openGLWidget.mousePressEvent = self.onOpenGLWidgetMousePressed

self.openGLWidget.mouseReleaseEvent = self.onOpenGLWidgetMouseReleased

self.updateButton.clicked.connect(self.generateControls)

self.generateControls()

def onOpenGLWidgetMousePressed(self, event):

width, height = self.openGLWidget.width(), self.openGLWidget.height()

point = np.array((event.x() / width, 1 - (event.y() / height)))

for control in self.controls:

if control.has\_point(\*point):

self.move\_controls.append((control, point - control.center()))

def onOpenGLWidgetMouseReleased(self, event):

self.move\_controls.clear()

def onOpenGLWidgetMouseMoved(self, event):

if len(self.move\_controls) == 0:

return

width, height = self.openGLWidget.width(), self.openGLWidget.height()

point = (event.x() / width, 1 - (event.y() / height))

for control, delta in self.move\_controls:

control.set\_center(\*(point - delta))

self.openGLWidget.update()

def loadScene(self):

width, height = self.openGLWidget.width(), self.openGLWidget.height()

GL.glViewport(0, 0, width, height)

GL.glMatrixMode(GL.GL\_PROJECTION)

GL.glLoadIdentity()

GL.glOrtho(0, 1, 0, 1, -1, 1)

GL.glMatrixMode(GL.GL\_MODELVIEW)

GL.glLoadIdentity()

def generateControls(self):

color = random\_rgb()

self.controls = [ControlElement(np.random.random(), np.random.random(), color) for \_ in range(4)]

self.openGLWidget.update()

def paintGL(self):

self.loadScene()

try:

self.drawStuff()

except Exception as exp:

print(exp)

pass

def drawStuff(self):

[c.paint() for c in self.controls]

self.paintBezier(\*[c.center() for c in self.controls[0:4]])

self.paintControlLines()

def paintControlLines(self):

GL.glColor4d(1, 1, 1, 1)

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

[GL.glVertex2d(\*c.center()) for c in self.controls]

GL.glEnd()

GL.glFinish()

def paintBezier(self, p1, p2, p3, p4):

self.paintBezierCommon((p1, p2, p3, p4))

def paintBezierCommon(self, points):

lines = np.array([[(p2 - p1) / 100 \* i + p1 for i in range(100)]

for p1, p2 in zip(points, points[1:])])

while len(lines) > 1:

lines = np.array([[

(p2 - p1) / 100 \* i + p1 for i, p1, p2 in zip(range(100), line1, line2)

]

for line1, line2 in zip(lines, lines[1:])

])

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

[GL.glVertex2d(\*p) for p in lines[0]]

GL.glEnd()

GL.glFinish()

if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QApplication(sys.argv)

window = MainWindow()

window.show()

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