Московский Авиационный Институт

(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа №7**

**по курсу «Компьютерная графика»**

|  |  |
| --- | --- |
| Студент: | Марков А.Н. |
| Группа: | М80-308Б-18 |
| Преподаватель: | Филиппов Г.С. |
| Оценка: |  |
| Дата: |  |

Москва

2020

**1. Постановка задачи.**

Написать программу, строящую полиномиальную кривую по заданным точкам. Обеспечить возможность изменения позиции точек и, при необходимости, значений касательных векторов и натяжения.

Вариант №5: Кривая Безье 2-й степени.

**2. Решение задачи.**

Кривая Безье 2-й степени строится в соответствии со следующей формулой:



,

где P0, P1, P2 — опорные точки.

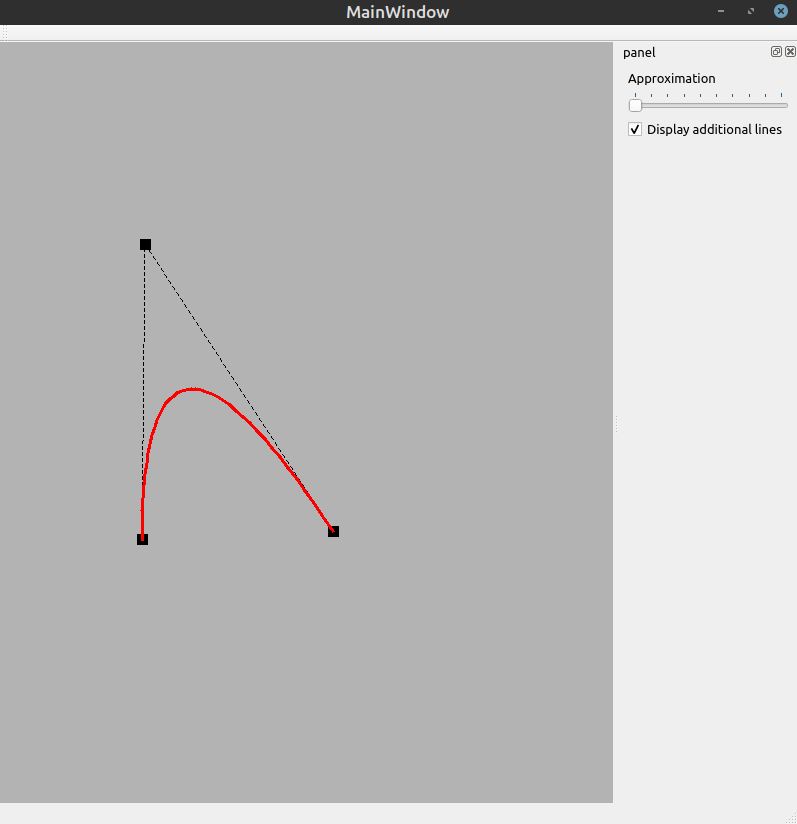
Для решения задачи я решил использовать C++ и фреймворк Qt, в котором использовал библиотеку Qpainter.

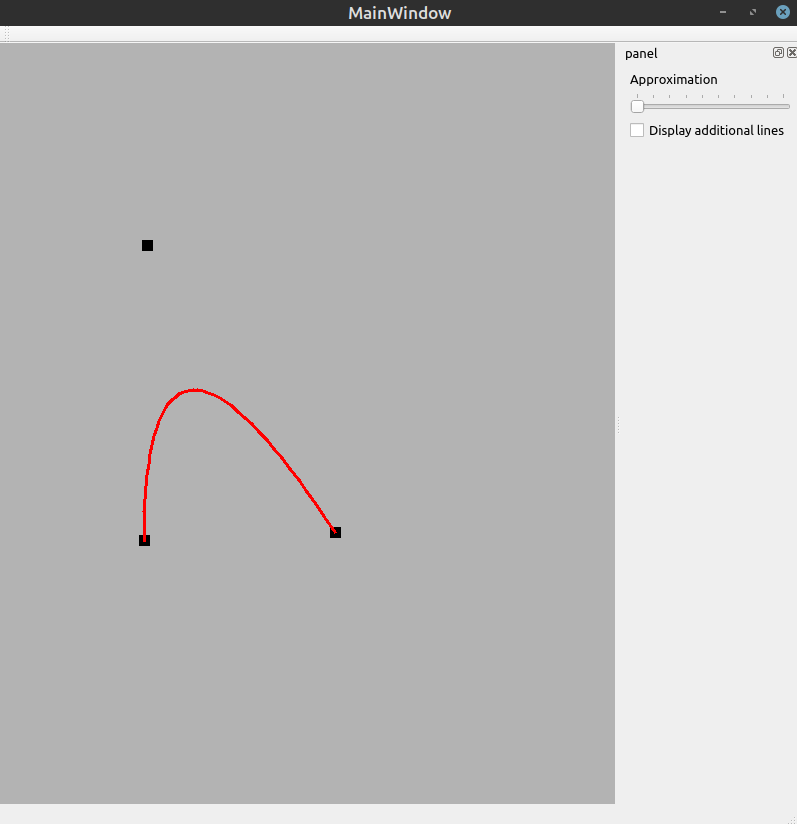
Кривая Безье в моей программе строится следующим образом:

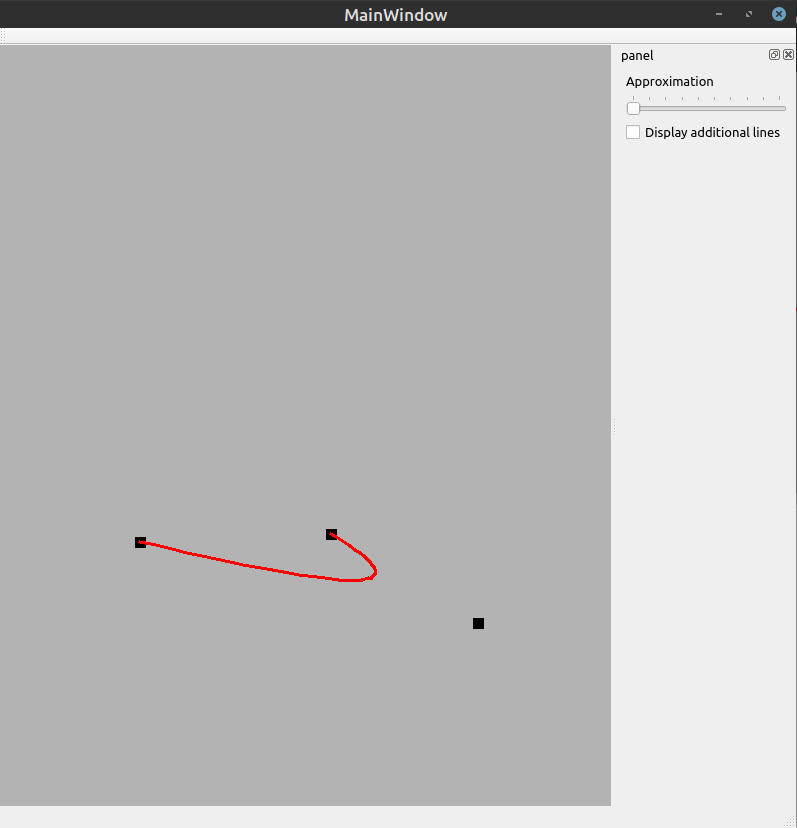
* вычисляется x координата кривой для всех значений 0 <= t <= 1 с некоторым шагом;
* вычисляется y координата кривой для всех значений 0 <= t <= 1 с некоторым шагом.

Таким образом вычисляются точки кривой, которые затем соединяются прямыми.

**3. Демонстрация работы программы.**







**4. Листинг программы.**

**view.h**

#ifndef VIEW\_H

#define VIEW\_H

#include <QWidget>

#include <QPointF>

#include <vector>

const unsigned int MAX\_CNT\_POINTS = 3;

class View : public QWidget

{

Q\_OBJECT

public:

explicit View(QWidget \*parent = nullptr);

void set\_step(double s);

void change\_display\_additional\_line();

protected:

void paintEvent(QPaintEvent \*);

void resizeEvent(QResizeEvent \*);

void mousePressEvent(QMouseEvent \*);

void mouseMoveEvent(QMouseEvent \*);

private:

double step = 0.025;

bool displayAdditionalLines = false;

unsigned int cntPoints;

std::vector<QPointF> points;

signals:

public slots:

};

#endif // VIEW\_H

**view.cpp**

#include "view.h"

#include <QMouseEvent>

#include <QPainter>

#include <QPolygon>

#include <cmath>

#include <QResizeEvent>

const int SQUARE\_SIZE = 10;

View::View(QWidget \*parent) : QWidget(parent) {

QPalette pal = palette();

pal.setColor(QPalette::Window, QColor(179, 179, 179));

setPalette(pal);

setAutoFillBackground(true);

cntPoints = 0;

points.resize(MAX\_CNT\_POINTS);

}

void View::set\_step(double s) {

step = s;

}

void View::change\_display\_additional\_line() {

if (displayAdditionalLines) {

displayAdditionalLines = false;

} else {

displayAdditionalLines = true;

}

}

void View::paintEvent(QPaintEvent \*) {

QPainter ptr{this};

ptr.setPen(QColor(0, 0, 0));

if (cntPoints != 0 && displayAdditionalLines) {

ptr.setPen(Qt::DashLine);

for (unsigned int i = 0; i < cntPoints - 1; i++) {

ptr.drawLine(static\_cast<int>(points[i].x()) + SQUARE\_SIZE / 2,

static\_cast<int>(points[i].y()) + SQUARE\_SIZE / 2,

static\_cast<int>(points[i + 1].x()) + SQUARE\_SIZE / 2,

static\_cast<int>(points[i + 1].y()) + SQUARE\_SIZE / 2);

}

ptr.setPen(Qt::SolidLine);

}

ptr.setBrush(QColor(0, 0, 0));

for (unsigned int i = 0; i < cntPoints; i++) {

QPolygon pol(QRect(static\_cast<int>(points[i].x()),

static\_cast<int>(points[i].y()),

SQUARE\_SIZE, SQUARE\_SIZE));

ptr.drawPolygon(pol);

}

if (cntPoints == 3) {

// draw Bezier curve

QPen newPen(QColor(255, 0, 0), 3);

ptr.setPen(newPen);

double prevX = points[0].x() + SQUARE\_SIZE / 2, prevY = points[0].y() + SQUARE\_SIZE / 2;

for (double t = step; t < 1.; t += step) {

double x = std::pow((1. - t), 2.) \* (points[0].x() + SQUARE\_SIZE / 2) +

2. \* t \* (1. - t) \* (points[1].x() + SQUARE\_SIZE / 2) +

std::pow(t, 2.) \* (points[2].x() + SQUARE\_SIZE / 2);

double y = std::pow((1. - t), 2.) \* (points[0].y() + SQUARE\_SIZE / 2) +

2. \* t \* (1. - t) \* (points[1].y() + SQUARE\_SIZE / 2) +

std::pow(t, 2.) \* (points[2].y() + SQUARE\_SIZE / 2);

ptr.drawLine(static\_cast<int>(prevX),

static\_cast<int>(prevY),

static\_cast<int>(x),

static\_cast<int>(y));

prevX = x;

prevY = y;

if (t + step >= 1.) {

x = points[2].x() + SQUARE\_SIZE / 2;

y = points[2].y() + SQUARE\_SIZE / 2;

ptr.drawLine(static\_cast<int>(prevX),

static\_cast<int>(prevY),

static\_cast<int>(x),

static\_cast<int>(y));

}

}

}

}

void View::resizeEvent(QResizeEvent \*e) {

if (e->oldSize().width() == -1 || e->oldSize().height() == -1) {

return;

}

double coef\_x = width() / static\_cast<double>(e->oldSize().width());

double coef\_y = height() / static\_cast<double>(e->oldSize().height());

for (unsigned int i = 0; i < cntPoints; i++) {

points[i].rx() \*= coef\_x;

points[i].ry() \*= coef\_y;

}

update();

}

void View::mousePressEvent(QMouseEvent \*e) {

if (e->button() == Qt::RightButton) {

if (cntPoints < 3) {

points[cntPoints] = e->pos();

cntPoints++;

}

}

update();

}

void View::mouseMoveEvent(QMouseEvent \*e) {

for (unsigned int i = 0; i < cntPoints; i++) {

if (e->pos().x() >= points[i].x() - 20 && e->pos().x() <= points[i].x() + 20 &&

e->pos().y() >= points[i].y() - 20 && e->pos().y() <= points[i].y() + 20) {

points[i] = e->pos();

break;

}

}

update();

}

**5. Выводы**

Выполнив данную лабораторную работу я получил представление о том как строятся кривые Безье и реализовал отрисовку данной кривой программно на языке C++ средствами Qt.