**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

**Лабораторная работа № 7**

Тема: Построение плоских полиномиальных кривых.

Студент: Кондратьев Егор Алексеевич

Группа: 80-306

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

1. **Постановка задачи**

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

Кривая Безье 2-й степени.

1. **Описание программы**

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.

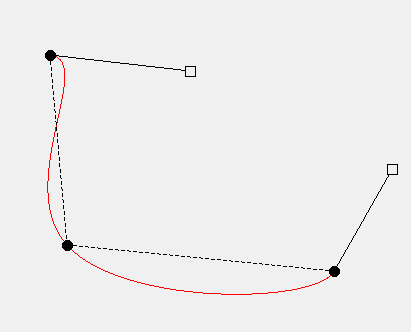
Кривая Безье в моей программе строится следующим образом:

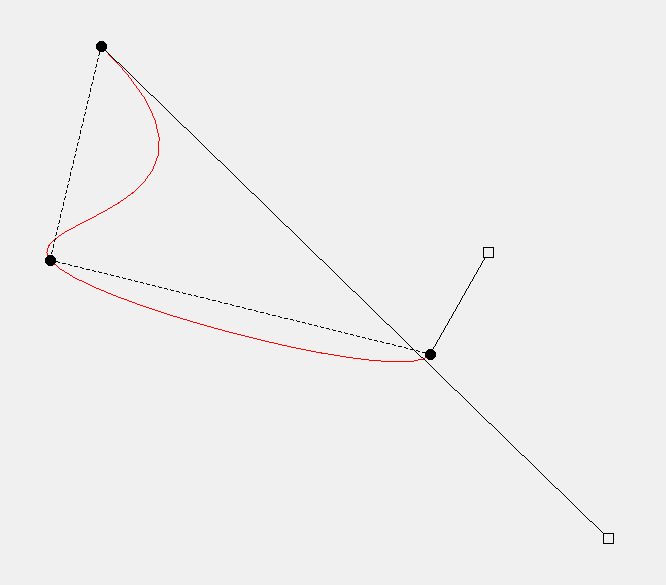
* вычисляется x координата кривой для всех значений 0 <= t <= 1 с некоторым шагом;
* вычисляется y координата кривой для всех значений 0 <= t <= 1 с некоторым шагом.

Таким образом вычисляются точки кривой, которые затем соединяются прямыми.

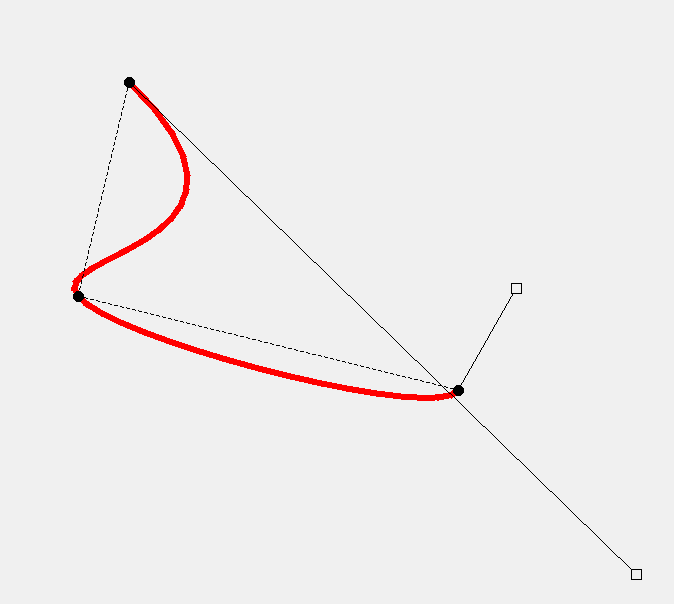
1. **Результаты выполнения**

Исходное положение



изменение положения

Утолщение линий



1. **Листинг программы**

Алгоритм работы состоит из нескольких шагов:

Обработка событий нажатия клавиш и взаимодействия с виджетом

У spinewidget, который унаследован от QWidget переопределено несколько функций отвечающих за взаимодействие виджета с пользователем : mousePressEvent(QMouseEvent\* e), mouseMoveEvent(QMouseEvent\* e), mouseReleaseEvent(QMouseEvent\* e). С помощью этих событий осуществляется общение программы и пользователя, например: можно добавить точку сплайна нажатием на виджет, перенести ее или полностью очистить виджет нажатием на Delete

Пример реализации события нажатия на виджет

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

if(e->buttons() == Qt::LeftButton){

if(!spline.add(QPointF(centrizePointCord(cordsOnCenter(e->pos()))))){

selectedPoint = spline.findPointOrRail(centrizePointCord(cordsOnCenter(e->pos())));

if(selectedPoint != spline.getPoints().end()){

isSelected = true;

if(!isCatch(selectedPoint->getRail().x()+selectedPoint->x(),

selectedPoint->getRail().y()+selectedPoint->y(),

centrizePointCord(cordsOnCenter(e->pos())).x(),

centrizePointCord(cordsOnCenter(e->pos())).y())){

isRailSelected = false;

} else {

isRailSelected = true;

}

}

}

}

}

Вычисления сплайна

Вычисление сплайна возможно только тогда, когда известны все три точки сплайна и две производные на его концах, тогда вызывается метод, в котором запрограммировано решение системы уравнений.

Вычисление полиномов сплайна непрерывной кривизны

void SplineClass::calculatePolynoms(Polynom& fP, Polynom& sP, int p0, int p1, int p2, int r0, int r2){

float& a1 = fP.a;

float& a2 = sP.a;

float& b1 = fP.b;

float& b2 = sP.b;

float& c1 = fP.c;

float& c2 = sP.c;

float& d1 = fP.d;

float& d2 = sP.d;

//qDebug() << p0 << ' ' << p1 << ' '<< p2 << ' '<<r0 << ' '<< r2 << '\n';

d1 = p0; //+

d2 = p1; //+

c1 = r0;

float n = -c1;

float h = d2-d1-c1;

float q = p2 - d2;

float z = r2;

float x = 3\*q + 2\*n - z;

a1 = (x-5\*h)/4;

b1 = (9\*h - x)/4;

b2 = 3\*a1 + b1;

c2 = 3\*a1 + 2\*b1 - n;

a2 = q - b2 - c2;

}

void SplineClass::calculate(){

calculatePolynoms(fSegment.xPolynom,

sSegment.xPolynom,

points[0].x(), points[1].x(),

points[2].x(), points[0].getRail().x(), points[2].getRail().x());

calculatePolynoms(fSegment.yPolynom,

sSegment.yPolynom,

points[0].y(), points[1].y(),

points[2].y(), points[0].getRail().y(), points[2].getRail().y());

}

Зная, что полиномы нормализованы и принимают значения от 0 до 1, можно, задав точность отрисовки, вычислять точки полиномов и соединять их линиями.

Отрывок кода, отвечающего за визуализацию сплайна

if(spline.isValid()){

QPen pen(Qt::red);

pen.setWidth(splineWidth);

painter.setPen(pen);

spline.calculate();

float step = 1/(float)spline.getPresicion();

QPointF prev = points[0];

for(float i = step;i < 1.0f;i += step ){

QPointF next = spline.fSegment.claculatePoint(i);

painter.drawLine(subLineFunction(toNormalCord(prev)), subLineFunction(toNormalCord(next)));

prev = next;

}

painter.drawLine(subLineFunction(toNormalCord(prev)), subLineFunction(toNormalCord(spline.fSegment.claculatePoint(1))));

prev = points[1];

for(float i = step;i < 1.0f;i += step ){

QPointF next = spline.sSegment.claculatePoint(i);

painter.drawLine(subLineFunction(toNormalCord(prev)), subLineFunction(toNormalCord(next)));

prev = next;

}

painter.drawLine(subLineFunction(toNormalCord(prev)), subLineFunction(toNormalCord(spline.sSegment.claculatePoint(1))));

}

**ЛИТЕРАТУРА**

1. QT documentation [Электронный ресурс]. URL: <https://doc.qt.io/>

(дата обращения: 28.11.2021).