Министерство образования и науки РФ

ФГБОУ ВПО «Тамбовский государственный технический университет»

Кафедра « »

Лабораторная работа №2

по дисциплине «Геометрическое моделирование»

Исследование аппроксимирующей кривой Безье

Выполнил: студент гр. −31 Д. Ю.

Проверил: С. А.

Тамбов,

***Цели и задачи****.*

Изучение возможности построения кривых на основе материалов Безье.

***Теория.***

Теорию кривых Безье разработал П. де Кастело в 1959 году и, независимо от него, П. Безье в 1962 году. Для построения кривой Безье N-ого порядка необходимо N+1 точек, две из которых определяют концы кривой, а остальные N-1 называются опорными [1]. В компьютерной графике наибольшее распространение получили квадратичные кривые Безье, строящиеся по трём точкам, и кубические кривые Безье, строящиеся по четырём точкам. Квадратичные кривые Безье используются, например, в шрифтах TrueType при определении контуров символов. API Windows позволяет строить только кубические кривые Безье.

***Решение задачи****.*

Используется одна формула, в общем виде выглядит следующим образом:

, где m – количество опорных точек, i = 0…m, P – опорная точка [x; y], t - коэффициент (0…1).

Пример расчетов для точек

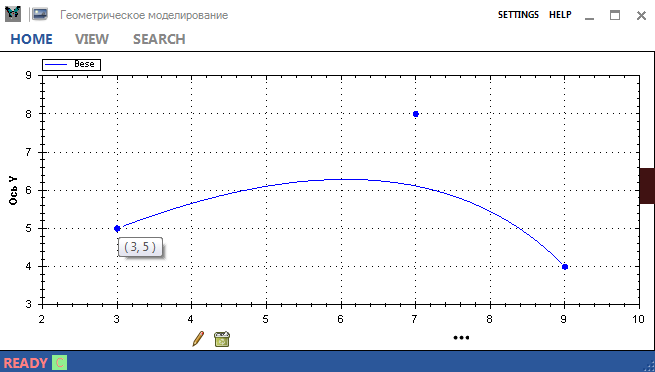
При m=3 формула 1 имеет вид

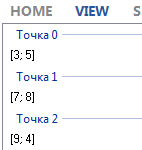
Итого, при t=0.01, P(t)=[3.11; 5.08];

Итого, при t=0.02, P(t)=[3.23; 5.16];

Этот процесс повторяется до тех пор, пока t меньше единицы.

Проверить полученный результат расчетов первых двух итераций, а заодно и правильность кода программы, можно указав те же три точки для построения (Рисунок 1):

**Рисунок 1 Тестирование программы по точкам**

Поставленные точки выводятся в отдельную вкладку View и группируются по очередности добавления на холст (Рисунок 2):

**Рисунок 2 Координаты точек на холсте**

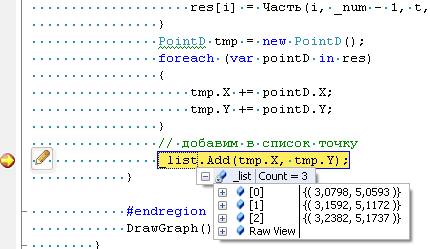
Расчеты первых итераций можно сверить с результатами работы в программе (Рисунок 3).

Рисунок 3 Остановка на второй итерации t

Значения близки к расчетным, это подтверждает правильность реализации алгоритма.

Расчеты сводятся к одному методу расчета произведений для точки и опорному методу расчета C.

Построение (матчасть) начинается с события

/// <summary>

/// Точки сняты, рассчитать формулу

/// </summary>

**private** void btnBuild\_Click**(object** sender**,** EventArgs e**)**

**{**

\_pointArr **=** **new** PointD**[**\_lpointArr**.**Count**];**

\_lpointArr**.**CopyTo**(**\_pointArr**);**

\_num **=** \_pointArr**.**Length**;**

var res **=** **new** PointD**[**\_num**];**

\_list**.**Clear**();**

#region построить график

**for** **(**double t **=** 0.01**;** t **<=** 1**;** t**+=**0.01**)**

**{**

**for** **(**int i **=** 0**;** i **<** \_num**;** i**++)**

**{**//для i-точки вычислить C\*t\*(1-t)\*P

res**[**i**]** **=** Часть**(**i**,** \_num**-**1**,** t**,** \_pointArr**[**i**]);**

**}**

PointD tmp **=** **new** PointD**();**

**foreach** **(**var pointD **in** res**)**

**{**

tmp**.**X **+=** pointD**.**X**;**

tmp**.**Y **+=** pointD**.**Y**;**

**}**

// добавим в список точку

\_list**.**Add**(**tmp**.**X**,** tmp**.**Y**);**

**}**

#endregion

DrawGraph**();**

**}**

/// <summary>

/// для i-точки вычислить C\*t\*(1-t)\*P

/// </summary>

**private** PointD Часть**(**int i**,** int m**,** double t**,** PointD P**)**

**{**

double ti **=** .0**,** tni **=** .0**;**

**if** **(**t **==** 0.0 **&&** i **==** 0**)** ti **=** 1.0**;**

**else** ti **=** Math**.**Pow**(**t**,** i**);**

**if** **(**m **==** i **&&** t **==** 1.0**)** tni **=** 1.0**;**

**else** tni **=** Math**.**Pow**((**1 **-** t**),** **(**m **-** i**));**

double tmp **=** Koefficient\_C**(**i**,** m**)\***ti**\***tni**;**

**return** **new** PointD**(**tmp**\***P**.**X**,** tmp**\***P**.**Y**);**

**}**

**private** **delegate** double \_C**(**int i**,** int m**);**

/// <summary>

/// Рассчитать коэффициент C для i-точки из m-точек

/// </summary>

**private** double Koefficient\_C**(**int i**,** int m**)**

**{**

// C(i)(m)= m!/(i!(m-i)!)

\_C res **=** **(**\_i**,** \_m**)** **=>**

**{**

double res\_m **=** 1**,** res\_i **=** 1**,** res\_im **=** 1**;**

**for** **(**int l **=** 1**;** l **<=** \_m**;** l**++)**

res\_m **=** l**\***res\_m**;**

**for** **(**int l **=** 1**;** l **<=** \_i**;** l**++)**

res\_i **=** l **\*** res\_i**;**

**for** **(**int l **=** 1**;** l **<=** **(**\_m**-**\_i**);** l**++)**

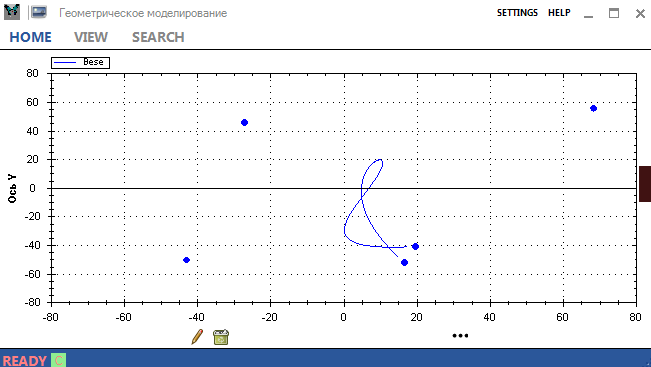
res\_im **=** l **\*** res\_im**;**

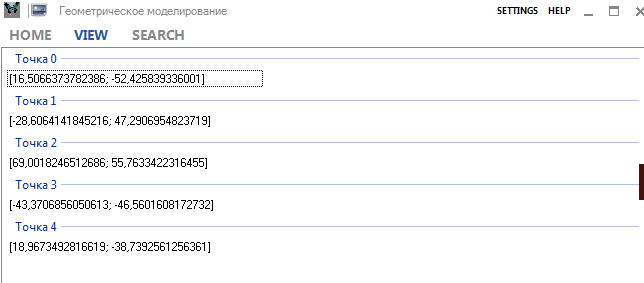
**return** res\_m**/(**res\_i**\***res\_im**);**

**};**

**return** res**(**i**,** m**);**

**}**

***Тест «&» пятью точками***

**Рисунок 4 Построение амперсанда**

**Рисунок 5 Координаты точек**

***Вывод.***

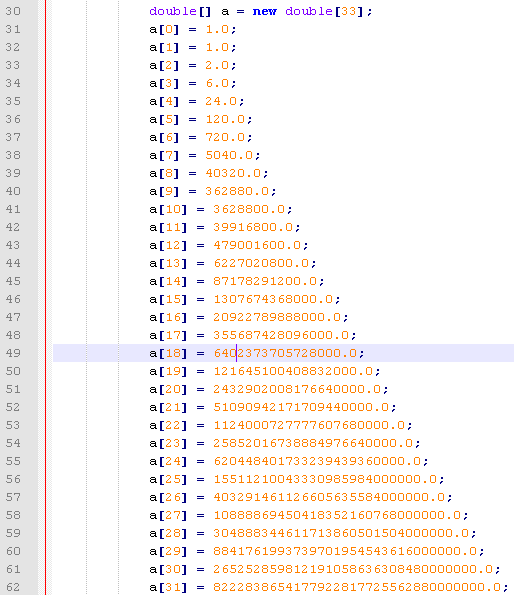
Рисуется все что взбредет в голову. Как минус можно отметить работу, основанную на факториалах, для больших объемов точек это становиться значимой нагрузкой на ПК. Одним из решений, направленных на оптимизацию этого участка можно предложить замену явного расчета факториала числа его заранее вычисленным значением (Рисунок 6).

Рисунок 6 Замена операции расчета факториала числа

***Список литературы.***

1. Теоретические основы изображения кривых Безье, http://www.delphikingdom.com/asp/viewitem.asp?catalogid=881