**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

**отчет**

**по лабораторной работе №2**

**по дисциплине «Компьютерная графика»**

**Тема: "Формирование различных кривых с использованием ортогонального проектирования на плоскость визуализации”**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6307 |  | Лазарев С. О. |
| Преподаватель |  | Матвеева И. В. |

Санкт-Петербург

2019

**Задание:**

Сформировать на плоскости кривую Безье на основе задающей ломаной, определяемой 3 и большим количеством точек. Обеспечить редактирование координат точек задающей ломаной с перерисовкой сплайна Безье.

**Теоретическая часть:**

**Координаты кривой описываются в зависимости от параметра t⋲[0,1]**

Для двух точек:

P = (1-t)P1 + tP2

Для трёх точек:

P =(1−t)2P1 + 2(1−t)tP2 + t2P3

Для четырёх точек:

P = (1−t)3P1 + 3(1−t)2tP2 +3(1−t)t2P3 + t3P4

Вместо Pi нужно подставить координаты i-й опорной точки (xi, yi).

Эти уравнения векторные, то есть для каждой из координат:

x = (1−t)2x1 + 2(1−t)tx2 + t2x3

y = (1−t)2y1 + 2(1−t)ty2 + t2y3

Вместо x1, y1,x2, y2, x3, y3 подставляются координаты трёх опорных точек, и в то время как t пробегает множество от 0 до 1, соответствующие значения (x, y) как раз и образуют кривую.

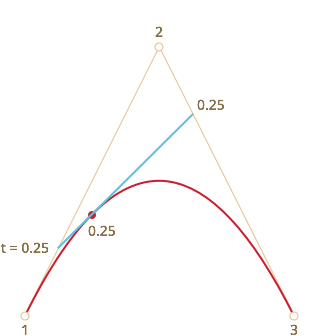
**Реализация:**

1. Рисуем опорные точки.
2. Строятся отрезки между опорными точками 1 → 2 → 3… .
3. Параметр t пробегает значения от 0 до 1.

Для каждого из этих значений t:

На каждом из отрезков берётся точка, находящаяся от начала на расстоянии от 0 до t пропорционально длине. Так как отрезков – два, то и точек две штуки.

Например, при t=0 – точки будут в начале, при t=0.25 – на расстоянии в 25% от начала отрезка, при t=0.5 – 50%(на середине), при t=1 – в конце отрезков.



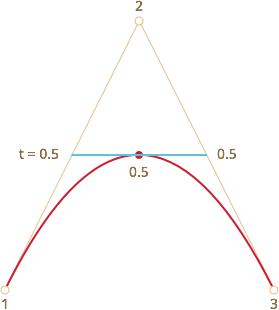


Рис. 1. Работа алгоритма.

Этот алгоритм рекурсивен. Для каждого t из интервала от 0 до 1 по этому правилу, соединяя точки на соответствующем расстоянии, из 4 отрезков делается 3, затем из 3 так же делается 2, затем из 2 отрезков – точка, описывающая кривую для данного значения t.

Так как алгоритм рекурсивен, то аналогичным образом могут быть построены кривые Безье и более высокого порядка: по пяти точкам, шести и так далее. Однако на практике они менее полезны. Обычно используются 2-3 точки, а для сложных линий несколько кривых соединяются. Это гораздо проще с точки зрения поддержки и расчётов.

Приложение написано на DartLang с использованием фреймворка Flutter. Устройство iPhone X.

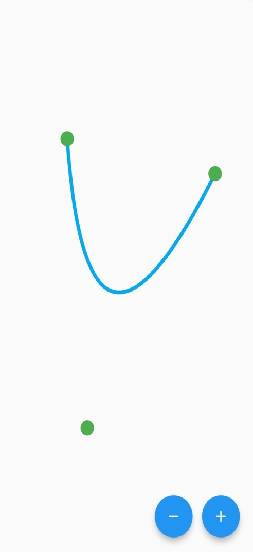


Рис. 2. Пример с 3 точками

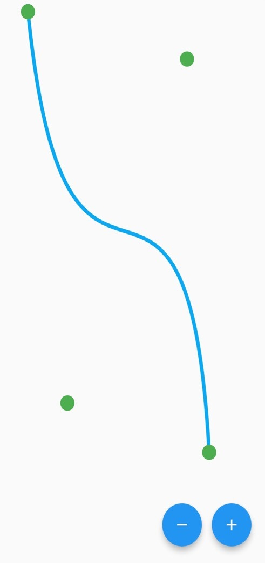


Рис. 3. Пример с 4 точками

**Код:**

import 'package:flutter/material.dart';

import 'dart:math';

import 'dart:ui';

void main() {

runApp(

MaterialApp(

home: Scaffold(

body: HomePage(),

),

),

);

}

List<Offset> offsets = new List<Offset>();

class HomePage extends StatefulWidget {

\_HomePageState createState() => \_HomePageState();

}

class \_HomePageState extends State<HomePage> {

@override

void initState() {

super.initState();

offsets = [

new Offset(100, 150),

new Offset(300, 200),

new Offset(200, 300)

];

}

@override

Widget build(BuildContext context) {

return Stack(

children: <Widget>[

CustomPaint(

child: getStack(),

painter: ShapesPainter(),

),

Positioned(

bottom: 40,

right: 20,

child: new FloatingActionButton(

tooltip: 'Добавить точку',

onPressed: () {

setState(() {

offsets.add(new Offset(100, 100));

});

},

child: new Icon(Icons.add),

),

),

Positioned(

bottom: 40,

right: 90,

child: new FloatingActionButton(

tooltip: 'Удалить точку',

onPressed: () {

if (offsets.length > 3)

setState(() {

offsets.removeLast();

});

},

child: new Icon(Icons.remove),

),

)

],

);

}

Widget getStack() {

List<Widget> list = new List<Widget>();

for (int i = 0; i < offsets.length; i++)

list.add(new PointShape(offsets[i], i));

return Stack(children: list);

}

}

class PointShape extends StatefulWidget {

Offset initPos;

int number;

PointShape(Offset pos, int nums) {

this.initPos = pos;

this.number = nums;

}

\_PointShapeState createState() => \_PointShapeState();

}

class \_PointShapeState extends State<PointShape> {

@override

Widget build(BuildContext context) {

return Positioned(

left: widget.initPos.dx,

top: widget.initPos.dy,

child: GestureDetector(

onPanUpdate: (details) {

setState(() {

offsets[widget.number] = Offset(

offsets[widget.number].dx + details.delta.dx,

offsets[widget.number].dy + details.delta.dy);

widget.initPos = offsets[widget.number];

});

},

child: Container(

alignment: Alignment.center,

decoration: new BoxDecoration(

borderRadius: new BorderRadius.circular(100),

color: Colors.green,

),

height: 20,

width: 20,

),

),

);

}

}

class ShapesPainter extends CustomPainter {

@override

void paint(Canvas canvas, Size size) {

List<Offset> arrayCords = new List<Offset>();

arrayCords = getBezierCurve(offsets);

var path = new Path();

path.moveTo(arrayCords[0].dx + 10, arrayCords[0].dy);

for (int i = 0; i < arrayCords.length; i++) {

path.lineTo(arrayCords[i].dx + 10, arrayCords[i].dy + 10);

}

canvas.drawPath(

path,

Paint()

..color = Colors.lightBlue

..style = PaintingStyle.stroke

..strokeWidth = 5.0,

);

path.close();

}

@override

bool shouldRepaint(CustomPainter oldDelegate) => true;

}

f(number) {

if (number <= 0) {

return 1;

} else {

return (number \* f(number - 1));

}

}

double getBezierBasis(i, n, t) {

f(n) {

if (n <= 1)

return 1;

else

return n \* f(n - 1);

}

return (f(n) / (f(i) \* f(n - i))) \* pow(t, i) \* pow(1 - t, n - i);

}

List<Offset> getBezierCurve(List<Offset> arr) {

var step = 0.01;

var res = new List<Offset>();

for (double t = 0; t < 1 + step; t += step) {

if (t > 1) {

t = 1;

}

double ytmp = 0;

double xtmp = 0;

for (var i = 0; i < arr.length; i++) {

double b = getBezierBasis(i, arr.length - 1, t);

xtmp += arr[i].dx \* b;

ytmp += arr[i].dy \* b;

}

res.add(new Offset(xtmp, ytmp));

}

return res;

}