Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Інститут комп’ютерних наук та інформаційних технологій
Кафедра програмного забезпечення

ЗВІТ

Про виконання лабораторної роботи №2
з дисципліни «Комп’ютерна графіка»
на тему “Програмування кривої Безьє”

Лекторка:
доцент кафедри ПЗ
Левус Є.В.

Виконала:
студ. групи ПЗ-22
Гільфанова К.С.

Прийняла:
старший викл. кафедри ПЗ
Івасько Н. М.

«___» _____ 2024 р.
∑ = _____

Львів
2024

Тема роботи. Програмування кривої Безьє

Мета роботи. Навчитися програмувати алгоритми побудови кривої Безьє

Теоретичні відомості

1. Чим визначається крива у векторній графіці з точки зору користувача?

У векторній графіці крива визначається множиною точок, що виражаються за певними алгоритмами. Щодо кривої Безьє, то вона визначається вершинами багатокутника, який задає форму кривої. Такий підхід створює для користувача більш природнє сприйняття, тобто зрозуміти, як керувати формою кривої буде більш інтуїтивно зрозумілим. Користувачу потрібно буде задати контрольні та опорні точки, перші – для керування формою, а другі безпосередньо належать кривій і визначають її початок на кінець. На виході буде отримано криву, що задається виразом

$$B(t) = \sum_{i=0}^{n} P_i * b_{in}(t)$$

де параметр t ∈[0;1], n-степінь полінома, який характеризується n+1 контрольними точками (вершинами), $P_i\ (x_i, y_i)$ – і-та контрольна точка, і - порядковий номер точки (вершини), - базисні функції кривої Безьє, названі також поліномами Бернштейна.

2. Які є формули для обчислення точок кривої Безьє? Запишіть формулу для скалярного представлення.

Є декілька основних формул, що дозволяють обчислити точки кривої Безьє. Серед них:

- Параметрична формула

  За параметричною формулою крива задається виразом:

$$B(t) = \sum_{i=0}^{n} P_i * b_{in}(t)$$

де параметр t ∈[0;1], n-степінь полінома, який характеризується n+1 контрольними точками (вершинами), $P_i\ (x_i, y_i)$ – і-та контрольна точка, і -

порядковий номер точки (вершини), - базисні функції кривої Безьє, названі також поліномами Бернштейна, які визначаються виразами:

$$\mathbf{b}_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i},$$
$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

- Рекурсивна формула

  За рекурсивною формулою де Кастельє крива Безьє визначається двома іншими кривими, побудованими на підмножинах контрольних точок:

$$B_{P_0P_1\ldots P_n}(t) = (1-t) * B_{P_0P_1\ldots P_{n-1}}(t) + t * B_{P_1P_2\ldots P_n}(t)$$

  У свою чергу кожна з двох кривих також будується на основі інших двох кривих, побудованих на відповідних підмножинах точок, і т.д. Отримаємо такі рекурсивні формули для знаходження точки кривої Безьє при значенні параметру t_0:

$$\begin{cases} P_i^{(0)} = P_i, \ i = \overline{0,n} \\ P_i^{(j)} = P_i^{(j-1)}(1-t_0) + P_{i+1}^{(j-1)} \cdot t_0, \ j = \overline{1,n}, \ i = \overline{0,n-j} \end{cases}$$

- Матрична формула

  Щоб представити криву можна використати вираз:

$$B(t)=T*N*P$$

  T – вектор параметру, T=[$t^n t^{n-1} \ldots t\ 1$]

  P – вектор вершин характеристичної ламаної, $P^T = [P_0\, P_1\, \ldots P_{n-1}\, P_n]$

$$N = \begin{bmatrix} \binom{n}{0}\binom{n}{n}(-1)^n & \binom{n}{1}\binom{n-1}{n-1}(-1)^{n-1} & \dots & \binom{n}{n}\binom{n-n}{n-n}(-1)^0 \\ \binom{n}{0}\binom{n}{n-1}(-1)^{n-1} & \binom{n}{1}\binom{n-1}{n-2}(-1)^{n-2} & \dots & 0 \\ & \dots & & \\ \binom{n}{0}\binom{n}{1}(-1)^1 & \binom{n}{1}\binom{n-1}{0}(-1)^0 & \dots & 0 \\ \binom{n}{0}\binom{n}{0}(-1)^0 & 0 & \dots & 0 \end{bmatrix},$$

За матричною формою криву Безьє можна записати так:

$$\mathbf{B}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \mathbf{M}_B \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix}$$

M – матриця коефіцієнтів

Скалярне представлення для одного виміру (наприклад, x-координата) може бути виражене окремо для кожного виміру:

$$x(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i P_{ix}$$

де P_{ix} - x-координата i-тої контрольної точки.

3. Опишіть структуру матриці коефіцієнтів кривої Безьє, де розміщено нульові, від ємні, додатні елементи?

Матриця коефіцієнтів кривої Безьє має розмірність (n+1) × (n+1), де n - степінь кривої. Кожен елемент матриці обчислюється за допомогою біноміального коефіцієнта і ступеня t:

$$B_{i,j} = \binom{n}{i} (1-t)^{n-i} t^i$$

Таким чином, нульові, від'ємні та додатні елементи матриці залежать від ступеня t та біноміальних коефіцієнтів. У випадку біноміального коефіцієнта, від'ємні та додатні значення можуть з'явитися в залежності від співвідношення

між і та n−і.

Постановка завдання

Створити редактор кривої Безьє, який має такий функціонал:

- введення і редагування вершин характеристичної ламаної,
- побудова кривої за параметричною формулою,
- додавання нової точки для характеристичної ламаної,
- контроль коректності введених даних,
- виведення необхідних підказок, повідомлень,
- виконання індивідуального варіанту.

Індивідуальний варіант

Візуалізувати криву Безьє за рекурсивною формулою; відобразити керуючі точки одним кольором, а опорні – іншим; вивести координати певної кількості (вводиться користувачем) точок кривої; обчислити значення поліномів Бернштейна bin(t) у заданому діапазоні значень параметру t (і та діапазон вводиться користувачем).

Текст програми

Файл: canva.dart

```
import 'dart:math';
import 'dart:ui';

import 'package:flutter/material.dart';
import 'package:lab2_cg/point.dart';

class myCanva extends CustomPainter {
  Color paral = Colors.black, plane = Colors.grey;
  List<Point> points=[];
  List<Offset> controlPoints=[];
  List<Offset> pointsOfCurve=[];
  late Canvas mCanvas;
  late Size mSize;
  bool drawBez = false;
  bool drawPar=false;
  double step=0.1;
  double i=1;
  double widthOfCell = 1;
  double heightOfCell = 1;

  @override
  void paint(Canvas canvas, Size size) {
```

```
    final Paint paint = Paint()
      ..color = plane
      ..strokeCap = StrokeCap.round
      ..strokeWidth = 1.0;
    this.mCanvas = canvas;
    this.mSize = size;
    widthOfCell = mSize.width / 20;
    heightOfCell = mSize.height / 20;

    paintCoordinatePlane();
    if(drawBez){
      controlPoints=convertPoints(points);
      drawBezier(controlPoints);
    }
    if(drawPar){
      controlPoints=convertPoints(points);
      drawControlPoints(controlPoints);
      pointsOfCurve=convertPoints(computeBezierCurvePoints(points,
step));
      drawCurvePoints(pointsOfCurve);
    }

  }

  Point convertPointsBack(Offset point){
    return Point((point.dx -mSize.width/2)/widthOfCell,
        (point.dy -mSize.height/2)/heightOfCell);
  }

  List<Offset> convertPoints(List<Point> list){
    List<Offset> res=[];
    for(int i=0;i<list.length;i++){
      res.add(Offset(mSize.width / 2 + list[i].x *
          widthOfCell,
          mSize.height / 2 - list[i].y * heightOfCell));
    }
    return res;
  }

  void drawControlPoints(List<Offset> points){
    Paint paint = Paint()
      ..color = Color(0xFF303F9F)
      ..strokeCap = StrokeCap.round
      ..strokeWidth = 5.0;

    mCanvas.drawLine(points.first, points.first, paint);
    mCanvas.drawLine(points.last, points.last, paint);

    paint = Paint()
      ..color = Colors.amberAccent
      ..strokeCap = StrokeCap.round
      ..strokeWidth = 5.0;
```

```
    for(int i=1;i<points.length-1;i++){
      mCanvas.drawLine(points[i], points[i], paint);
    }
  }

  void drawCurvePoints(List<Offset> points){
    Paint paint = Paint()
      ..color = Colors.black
      ..strokeCap = StrokeCap.round
      ..strokeWidth = 0.5;

    for(int i=0;i<pointsOfCurve.length;i++){
      mCanvas.drawLine(pointsOfCurve[i], pointsOfCurve[i], paint);
    }
  }

  void drawBezier(List<Offset> points) {
    if (points.length < 2) {
      return;
    }
    for(double i=0;i<1;i+=step){
      Offset t=recursiveBezier(points, i);
      pointsOfCurve.add(t);
    }

    drawControlPoints(points);
    drawCurvePoints(points);
  }


  Offset recursiveBezier(List<Offset> points, double t){
    if(points.length==1){
      // pointsOfCurve.add(points[0]);
      return points[0];
    }
    List<Offset> nextPoints = [];

    for (int i = 0; i < points.length - 1; i++) {
      Offset pointOnCurve = points[i] * (1 - t) + points[i + 1] *
t;
      nextPoints.add(pointOnCurve);
    }
    return recursiveBezier(nextPoints, t);
  }

  List<Point> computeBezierCurvePoints(List<Point> controlPoints,
double step) {
    if (controlPoints.length < 2) {
      throw ArgumentError('At least two control points are
required.');
    }

    List<Point> curvePoints = [];
```

```
    for (double t = 0.0; t <= 1.0; t += step) {
      Point point = computeBezierPoint(t, controlPoints);
      curvePoints.add(point);
    }

    return curvePoints;
  }

  static Point computeBezierPoint(double t, List<Point>
controlPoints) {
    int n = controlPoints.length - 1;
    double x = 0.0;
    double y = 0.0;

    for (int i = 0; i <= n; i++) {
      double coefficient = bernsteinCoefficient(n, i, t);
      x += coefficient * controlPoints[i].x;
      y += coefficient * controlPoints[i].y;
    }

    return Point(x, y);
  }



  static double bernsteinCoefficient(int n, int i, double t) {
    return binomialCoefficient(n, i).toDouble() * pow(1 - t, n -
i) * pow(t, i);
  }

  static int binomialCoefficient(int n, int k) {
    if (k < 0 || k > n) {
      return 0;
    }

    if (k == 0 || k == n) {
      return 1;
    }

    return factorial(n) ~/ (factorial(k) * factorial(n - k));
  }

  static int factorial(int n) {
    if (n == 0 || n == 1) {
      return 1;
    }

    return n * factorial(n - 1);
  }

  void drawText(Color color, String text, double x, double y) {
    final TextPainter textPainter = TextPainter(
```

```
      textDirection: TextDirection.ltr,
      textAlign: TextAlign.center,
    );

    textPainter.text = TextSpan(
      text: text,
      style: TextStyle(
        color: color,
        fontSize: 12.0,
      ),
    );

    textPainter.layout();
    textPainter.paint(
        mCanvas, Offset(x - textPainter.width / 2, y -
textPainter.height / 2));
  }

  String getStringBernsteinCoefficient(int i, double t){
    String res='';
    if(i>points.length-1) return 'Invalid i';
    for(int i=0;i<points.length;i++){
      res+=bernsteinCoefficient(points.length-1, i, t).toString();
      res+='; ';
    }
    return res;
  }

  String getCoords(int q){
    String res='';
    if(q>pointsOfCurve.length-1) return 'Invalid quantity';
    int s=pointsOfCurve.length~/q;
    for(int i=0;i<pointsOfCurve.length;i+=s){
      res+=convertPointsBack(pointsOfCurve[i]).toString();
    }
    return res;
  }

  void paintCoordinatePlane() {
    final Paint paint = Paint()
      ..color = plane
      ..strokeCap = StrokeCap.round
      ..strokeWidth = 1.0;
    final double arrowLength = 20.0;
    final double arrowWidth = 10.0;

    mCanvas.drawLine(Offset(0.0, mSize.height / 2),
        Offset(mSize.width, mSize.height / 2), paint);
    mCanvas.drawLine(Offset(mSize.width / 2, 0),
        Offset(mSize.width / 2, mSize.height), paint);

    // Draw arrowhead at the end of x-axis
    Path arrowPath = Path();
```

```
    arrowPath.moveTo(
        mSize.width - arrowLength, mSize.height / 2 - arrowWidth /
2);
    arrowPath.lineTo(mSize.width, mSize.height / 2);
    arrowPath.lineTo(
        mSize.width - arrowLength, mSize.height / 2 + arrowWidth /
2);
    mCanvas.drawPath(arrowPath, paint);

    // Draw arrowhead at the end of y-axis
    arrowPath.moveTo(mSize.width / 2 - arrowWidth / 2,
arrowLength);
    arrowPath.lineTo(mSize.width / 2, 0.0);
    arrowPath.lineTo(mSize.width / 2 + arrowWidth / 2,
arrowLength);
    mCanvas.drawPath(arrowPath, paint);

    for (int i = 1; i < 20; i++) {
      mCanvas.drawLine(Offset(i * widthOfCell, (mSize.height +
arrowWidth) / 2),
          Offset(i * widthOfCell, (mSize.height - arrowWidth) /
2), paint);
      mCanvas.drawLine(Offset((mSize.width - arrowWidth) / 2, i *
heightOfCell),
          Offset((mSize.width + arrowWidth) / 2, i *
heightOfCell), paint);
      if (i != 10) {
        drawText(
            plane, (i - 10).toString(), i * widthOfCell,
mSize.height / 2 + 15);
        drawText(plane, (-i + 10).toString(), mSize.width / 2 +
15,
            i * heightOfCell - 2);
      }
    }
    drawText(plane, '0', mSize.width / 2 + 15, mSize.height / 2 +
15);
    drawText(plane, 'Ox', mSize.width-15, mSize.height/2+15);
    drawText(plane, 'Oy', mSize.width/2+15, 10);
  }

  @override
  bool shouldRepaint(CustomPainter oldDelegate) {
    return false;
  }

}
```

Файл: control_panel.dart

```
import 'dart:ffi';
```

```
import 'package:flutter/material.dart';
import 'package:lab2_cg/point.dart';
import 'coord_field.dart';
import 'styles.dart';
import 'canva.dart';


class ControllPanel extends StatefulWidget {
  myCanva canva;
  ControllPanel({required this.canva});
  int choosenPoint = 0;
  int quant = 2;
  CoordField t=CoordField(hintText: 'Enter t', controller:
TextEditingController(),);
  CoordField q=CoordField(hintText: 'Enter quantity', controller:
TextEditingController(),);
  CoordField i = CoordField(hintText: 'Enter i', controller:
TextEditingController(),);
  List<CoordField> listOfFieldsX = [CoordField(hintText: '0X:
1.0', controller: TextEditingController(),),CoordField(hintText:
'1X: 1.0', controller: TextEditingController(),)];
  List<CoordField> listOfFieldsY = [CoordField(hintText: '0Y:
1.0', controller: TextEditingController(),),CoordField(hintText:
'1Y: 1.0', controller: TextEditingController(),)];

  @override
  State<ControllPanel> createState() => _ControllPanelState();
}

class _ControllPanelState extends State<ControllPanel> {

  @override
  Widget build(BuildContext context) {
    return Column(children: [
      Expanded(
        child: Container(
          padding: EdgeInsets.all(5),
          margin: EdgeInsets.all(5),
          decoration: kFrame,
          child: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Expanded(
                flex: 3,
                child: Center(
                  child: Text(
                    'Quantity: ${widget.quant}',
                    style: kSubtitle,
                  ),
                ),
              ),
              Expanded(
                child: IconButton(
```

```
                    style: kButtonStyle,
                    onPressed: () {
                      setState(() {
                        if(widget.quant>2){
                          if(widget.choosenPoint==widget.quant-1){
                            widget.choosenPoint--;
                          }
                          widget.quant--;
                          widget.listOfFieldsX.removeLast();
                          widget.listOfFieldsY.removeLast();
                        }
                      });
                    },
                    icon: Icon(
                      Icons.remove,
                      color: Colors.white,
                    ),
                  ),
                ),
                SizedBox(width: 20,),
                Expanded(
                  child: IconButton(
                    style: kButtonStyle,
                    onPressed: () {
                      setState(() {
                        widget.quant++;

widget.listOfFieldsX.add(CoordField(hintText: '${widget.quant-1}X:
1.0', controller: TextEditingController(),));

widget.listOfFieldsY.add(CoordField(hintText: '${widget.quant-1}Y:
1.0', controller: TextEditingController(),));
                      });
                    },
                    icon: Icon(
                      Icons.add,
                      color: Colors.white,
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
        Expanded(
          child: Container(
            padding: EdgeInsets.all(5),
            margin: EdgeInsets.all(5),
            decoration: kFrame,
            child: Row(
              children: [
                Column(
                  mainAxisAlignment: MainAxisAlignment.center,
```

```
                children: [
                  IconButton(
                    icon: Icon(Icons.arrow_upward),
                    onPressed: () {
                      setState(() {
                        if(widget.choosenPoint+1<widget.quant){
                          setState(() {
                            widget.choosenPoint++;

widget.listOfFieldsX[widget.choosenPoint].controller =
TextEditingController();

widget.listOfFieldsY[widget.choosenPoint].controller =
TextEditingController();
                          });
                        }
                      });
                    },
                  ),
                  SizedBox(
                    height: 20,
                    child: Text(widget.choosenPoint.toString()),
                  ),
                  IconButton(
                    icon: Icon(Icons.arrow_downward),
                    onPressed: () {
                      if(widget.choosenPoint>0){
                        setState(() {
                          widget.choosenPoint--;

widget.listOfFieldsX[widget.choosenPoint].controller =
TextEditingController();

widget.listOfFieldsY[widget.choosenPoint].controller =
TextEditingController();
                        });
                      }
                    },
                  ),
                ],
              ),
              Expanded(child:
widget.listOfFieldsX[widget.choosenPoint]),
              Expanded(child:
widget.listOfFieldsY[widget.choosenPoint]),

            ],
          ),
        ),
      ),
      Expanded(
        child: Container(
          padding: EdgeInsets.all(5),
```

```
          margin: EdgeInsets.all(5),
          decoration: kFrame,
          child: Row(
            children: [
              Expanded(
                child: Container(
                  margin: EdgeInsets.all(5),
                  child: Column(
                    crossAxisAlignment:
CrossAxisAlignment.stretch,
                    mainAxisAlignment:
MainAxisAlignment.spaceEvenly,
                    children: [
                      TextButton(
                    style: kButtonStyle,
                    onPressed: () {
                      setState(() {
                        widget.canva.step=widget.t.val;
                        if(widget.canva.drawBez){
                          widget.canva.points.clear();
                          widget.canva.pointsOfCurve.clear();
                          widget.canva.drawBez=false;
                        }
                        for(int i=0;i<widget.quant;i++){

widget.canva.points.add(Point(widget.listOfFieldsX[i].val,
widget.listOfFieldsY[i].val));
                        }
                        widget.canva.drawBez=true;
                      });
                    },
                    child: Text(
                      'Recursive formula',
                      style: kTextStyle,
                    ),
                  ),
                      TextButton(
                        style: kButtonStyle,
                        onPressed: () {
                          widget.canva.step=widget.t.val;
                          setState(() {
                            if(widget.canva.drawPar){
                              widget.canva.points.clear();
                              widget.canva.pointsOfCurve.clear();
                              widget.canva.drawPar=false;
                            }
                            for(int i=0;i<widget.quant;i++){

widget.canva.points.add(Point(widget.listOfFieldsX[i].val,
widget.listOfFieldsY[i].val));
                            }
                            widget.canva.step=widget.t.val;
                            widget.canva.drawPar=true;
```

```
                        });
                      },
                      child: Text(
                        'Parametric formula',
                        style: kTextStyle,
                      ),
                    ),
                  ],
                ),
              ),
            ),
            Expanded(
              child: Container(
                margin: EdgeInsets.all(5),
                child: widget.t,
              ),
            ),
          ],
        ),
      ),
      Expanded(
        child: Container(
          padding: EdgeInsets.all(5),
          margin: EdgeInsets.all(5),
          decoration: kFrame,
          child: Row(
            children: [
              Expanded(
                child: Column(
                  children: [
                  widget.i,
                  TextButton(
                    style: kButtonStyle,
                    onPressed: () {
                      String
text=widget.canva.getStringBernsteinCoefficient(widget.i.val.toInt
(), widget.t.val);
                      showDialog(
                        context: context,
                        builder: (BuildContext context) {
                          return AlertDialog(
                            title: Text('Bernstein
Coefficients:'),
                            content: Text(text),
                            actions: [
                              TextButton(
                                style: kButtonStyle,
                                onPressed: () {
                                  Navigator.of(context).pop();
                                },
                                child: Text('Close', style:
kTextStyle,),
```

```
                              ),
                            ],
                          );
                        },
                      );
                    },
                    child: Text(
                      'Get coefficients',
                      style: kTextStyle,
                    ),
                  ),
                ],
                crossAxisAlignment: CrossAxisAlignment.stretch,)
              ),
              SizedBox(width: 10,),
              Expanded(
                  child: Column(
                    children: [
                      widget.q,
                      TextButton(
                        style: kButtonStyle,
                        onPressed: () {
                          String
text=widget.canva.getCoords(widget.q.val.toInt());
                          showDialog(
                            context: context,
                            builder: (BuildContext context) {
                              return AlertDialog(
                                title: Text('Points of curve:'),
                                content: Text(text),
                                actions: [
                                  TextButton(
                                    style: kButtonStyle,
                                    onPressed: () {
                                      Navigator.of(context).pop();
                                    },
                                    child: Text('Close', style:
kTextStyle,),
                                  ),
                                ],
                              );
                            },
                          );
                        },
                        child: Text(
                          'Get coordinates',
                          style: kTextStyle,
                        ),
                      ),
                    ],
                    crossAxisAlignment:
CrossAxisAlignment.stretch,)
              ),
```

```
            ],
          ),
        ),
      ),
    ]);
  }
}
```

Файл: coord_field.dart

```
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'styles.dart';
import 'point.dart';

class CoordField extends StatelessWidget {
  String hintText;
  double val=1;
  TextEditingController controller;
  CoordField({required this.hintText, required this.controller});

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(4.0),
      child: TextField(
        controller: controller,
        inputFormatters: <TextInputFormatter>[
          FilteringTextInputFormatter.allow(RegExp(r'^-?[0-
9]*\.?[0-9]*$')),
        ],
        textAlign: TextAlign.center,
        onChanged: (value) {
          try{
            hintText=hintText.substring(0, hintText.length-
val.toString().length);
            if(value!='-')
              val=double.parse(value);
            hintText+=val.toString();
          }
          catch(e){
            val=-999;
          }
        },
        decoration:
        kTextFieldDecoration.copyWith(hintText: hintText),
      ),
    );
  }
}
```

Файл: main.dart

```
import 'package:flutter/material.dart';
import 'main_screen.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: mainScreen(),
    );
  }
}
```

Файл: main_screen.dart

```
import 'package:flutter/material.dart';
import 'canva.dart';
import 'package:animated_text_kit/animated_text_kit.dart';
import 'control_pannel.dart';
import 'styles.dart';

class mainScreen extends StatefulWidget {
  const mainScreen({super.key});

  @override
  State<mainScreen> createState() => _mainScreenState();
}

class _mainScreenState extends State<mainScreen> {
  late myCanva canva;
  @override
  void initState() {
    super.initState();
    canva=myCanva();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(
        children: [
          SizedBox(
            child: DefaultTextStyle(
              child: AnimatedTextKit(
```

```
                animatedTexts: [TypewriterAnimatedText('bezier
curve')],
              ),
              style: TextStyle(
                color: Color(0xFF212121),
                fontSize: 40.0,
                fontWeight: FontWeight.w800,
              ),
            ),
          ),
          Expanded(
            child: Row(
              children: [
                Expanded(
                  flex: 3,
                    child: Container(
                        padding: EdgeInsets.all(5),
                        margin: EdgeInsets.all(5),
                        decoration: kFrame,
                        child: CustomPaint(
                            size: Size(900,900),
                            painter: canva))),
                Expanded(
                  flex: 2,
                  child: Container(
                    child: ControllPanel(canva: canva,),
                  ),
                )
              ],
            ),
          ),
        ],
      ),
    );
  }
}
```

Файл: point.dart

```
    class Point{
  double x,y;
  Point(this.x, this.y);

@override
String toString(){
  return ('($x; $y); ');
}

}
```

Файл: styles.dart

```
import 'package:flutter/material.dart';

const kTextFieldDecoration = InputDecoration(
  hintText: 'Enter a value',
  hintStyle: TextStyle(color: Color(0xFF757575)),
  border: OutlineInputBorder(
    borderRadius: BorderRadius.all(Radius.circular(32)),
  ),
  enabledBorder: OutlineInputBorder(
    borderSide: BorderSide(color: Color(0xFF303F9F), width: 1.0),
    borderRadius: BorderRadius.all(Radius.circular(32.0)),
  ),
  focusedBorder: OutlineInputBorder(
    borderSide: BorderSide(color: Color(0xFF303F9F), width: 2.0),
    borderRadius: BorderRadius.all(Radius.circular(32.0)),
  ),
);

var kButtonStyle = TextButton.styleFrom(
    backgroundColor: Color(0xFF303F9F),
    textStyle: TextStyle(color: Colors.white));

const kTextStyle=TextStyle(
  color: Colors.white,
);

var kFrame=BoxDecoration(
    borderRadius: BorderRadius.all(Radius.circular(32)),
    border: Border.all(color: Color(0xFF9E9E9E)));

const kSubtitle=TextStyle(
  color:Color(0xFF212121),
  fontWeight: FontWeight.w700,
  fontSize: 20
);

const kButtonText=TextStyle(
color: Color(0xFF303F9F)
);
```

Результати виконання програми

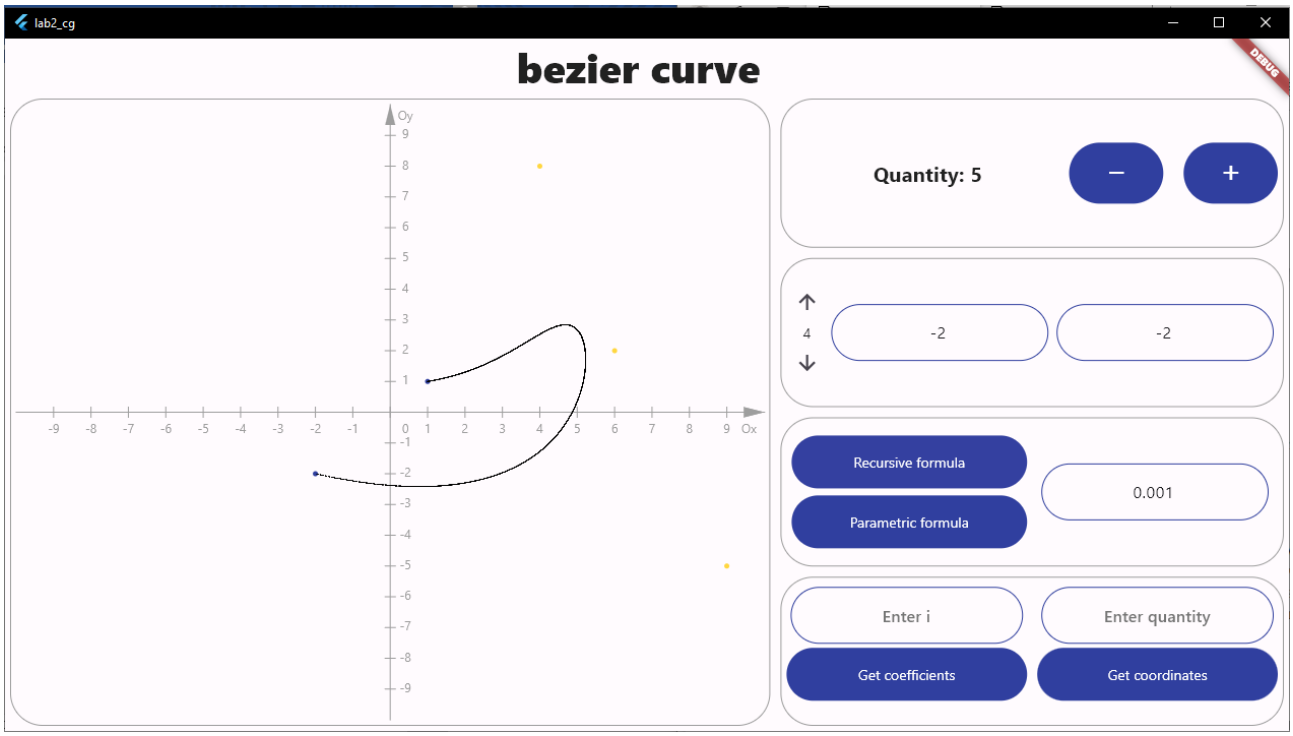

Рис. 1. Зображення кривої Безьє четвертого порядку

Рис. 2. Вивід коефіцієнтів Бернштейна

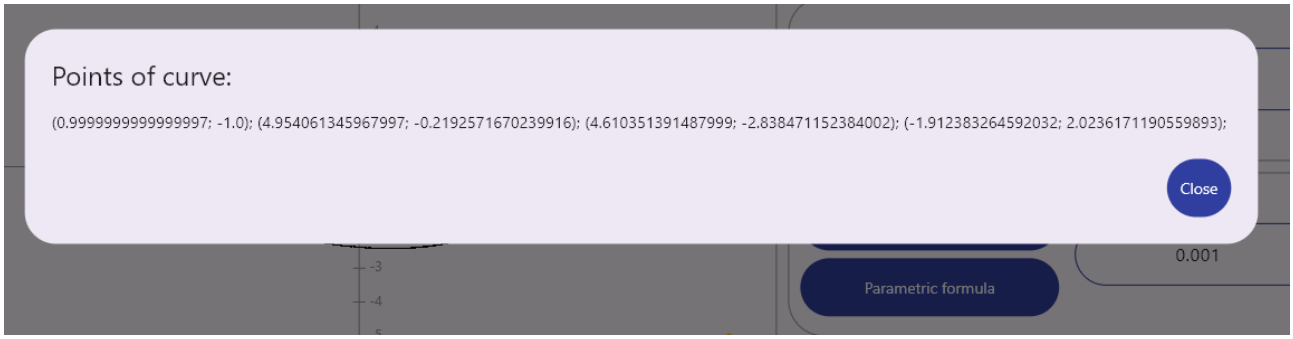

Рис. 2. Вивід координат точок кривої

Висновки

Під час виконання лабораторної роботи я дізналася, що таке крива Безьє, які існують способи її побудови, зрозуміла принцип роботи їх, запрограмувала рекурсивну та параметричну формули та з їх допомогою візуалізувала криву.