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

ЗВІТ

Про виконання лабораторної роботи №1
з дисципліни «Комп’ютерна графіка»
на тему “Побудова двовимірних фігур засобами мови програмування”

Лекторка:
доцент кафедри ПЗ
Левус Є.В.

Виконала:
студ. групи ПЗ-12
Гільфанова К.С.

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

«___» _____ 2024 р.
∑ = _____

Львів
2024

Тема роботи: Побудова двовимірних фігур засобами мови програмування.

Мета роботи: Навчитись будувати двомірні зображення з допомогою графічних примітивів мови програмування.

Теоретичні відомості

1. Переваги обраної технології

Мною було обрано Canvas у Flutter, так як це є доволі зручний інструмент для зображення графічних елементів. Flutter дозволяє легко втілити та налаштувати графічні компоненти, що призводить до створення інтерфейсу, який не лише функціональний, але й відзначається високим рівнем дизайну.

За допомогою Canvas у Flutter, я можу інтегрувати різноманітні графічні ефекти, анімації та інші візуальні елементи у мої додатки. Це дозволяє мені легко налаштовувати розміщення та стилізацію графічних об'єктів, створюючи при цьому інтерфейс, який відповідає сучасним стандартам та вимогам користувачів.

У Flutter можна легко створювати інтерфейс, що буде естетичним та простим у користуванні, тому мені було зручно використовувати даний фреймворк для даного завдання.

Під час виконання завдання було використано методи стандартного класу CustomPainter для зображення ліній, що сполучають 2 задані мною точки, заливки фігур, заданих певними точками, а також зображення тексту.

2. Наведіть приклади застосування тривимірної КГ?

Тривимірна комп'ютерна графіка знаходить широке застосування в сучасних технологіях, зокрема в системах віртуальної, доповненої та змішаної реальності. Одним із прикладів використання її може бути сфера тренажерів, де віртуальні середовища створюються для тренування певних навичок або ситуацій, наприклад, пілотажу літака чи хірургічних вправ. У сфері освіти та науки тривимірна графіка використовується для створення інтерактивних та навчальних програм. В індустрії розваг тривимірна комп'ютерна графіка відіграє важливу роль у створенні реалістичних та захоплюючих візуальних ефектів для відеоігор, фільмів та анімацій. Сучасні блокбастери і відеоігри завдячують тривимірній графіці за можливість створювати вражаючі світи та персонажі, які

занурюють глядача чи гравця в атмосферу та динаміку подій.

3. Що таке растр та його роздільна здатність?

Растр - це графічне зображення, яке представлене виглядом матриці пікселів (точок), де кожен піксель має свій колір і позначається конкретним значенням. Растрові зображення зберігаються у вигляді сітки пікселів, де кожен піксель має визначені координати та колір.

Роздільна здатність (або роздільна здатність зображення) вказує на кількість пікселів, які можуть бути відображені на одиницю довжини чи площі. Вона вимірюється в пікселях на дюйм (dpi) для друку або точок на дюйм (ppi) для екрану.

4. У чому полягає обробка зображень?

Обробка зображень включає у себе перетворення вхідних зображень, при цьому результатом також є зображення. Прикладами такої обробки можуть бути удосконалення контрасту, чіткості, корекція кольорів, редукція кольорів, згладжування, зменшення шумів тощо. В якості вихідних матеріалів для обробки можуть виступати космічні знімки, скановані зображення, радіолокаційні та інфрачервоні зображення і так далі. Основними завданнями обробки зображень є поліпшення якості зображення залежно від конкретного критерію (наприклад, реставрація або відновлення), а також здійснення спеціальних перетворень, що радикально змінюють його вигляд. В останньому випадку обробка зображень може виступати як проміжний етап для подальшого його розпізнавання.

Постановка завдання

Написати програму згідно індивідуального варіанту вибраною мовою програмування з використанням її базових графічних примітивів. Програма має відповідати таким вимогам:

5. Відображення системи координат з початком у центрі області виведення з відповідними підписами та позначками (початок, одиничний відрізок, напрям, назва осей).
6. Задання фігур за введеними координати, що відповідають координатам відповідної побудованої декартової системи, а не

координатам області виведення (Canvas).

7. Оптимальний ввід користувачем координат фігури з автоматичним обчисленням за можливості інших координат для уникнення зайвих обчислень користувачем.
8. Передбачити можливість некоректного введення даних.
9. Зручний інтерфейс користувача.

Індивідуальний варіант

Побудувати декілька паралелограмів за введеними координатами вершин (заливка – будь-який випадковий колір) лише у другій координатній чверті із автоматичним відображенням діагоналей та однієї з висот паралелограма. Забезпечити можливість вибору кольору діагоналей

Текст програми

Файл: canva.dart

```
import 'package:flutter/material.dart';
import 'point.dart';
import 'control_pannel.dart';
import 'point.dart';
import 'dart:math';

class myCanva extends CustomPainter {
  Color paral = Colors.black, plane = Colors.grey;
  late Canvas mCanvas;
  late Size mSize;
  bool drawPar = false;
  double widthOfCell = 1;
  double heightOfCell = 1;
  List<Shape> allFigures = [];

  @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 (drawPar) {
      for (int i = 0; i < allFigures.length; i++) {
        allFigures[i].draw(mCanvas, mSize, widthOfCell,
```

```
heightOfCell);
      }
    }
  }

  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));
  }

  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);
  }

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

class Shape {
  Color diag1, diag2, background;
  List<Point> points = [];
  Shape({required this.points, required this.diag1, required
this.diag2, required this.background});

  double findLenght(int p1, int p2) {
    double n = sqrt(pow(points[p2].x - points[p1].x, 2) +
        pow(points[p2].y - points[p1].y, 2));
    //print('lenght of $p1 $p2 $n');
    return n;
  }

  bool findCos(int p1, int m, int p2) {
    double a = findLenght(p1, m), b = findLenght(m, p2), c =
findLenght(p1, p2);
    double x = (a * a + b * b - c * c) / (2 * a * b);
    return x > 0 ? false : true;
  }

  int findStupidAngle() {
```

```
    if (findCos(0, 1, 2)) return 1;
    return 0;
  }

  Point findHeight() {
    int p1 = findStupidAngle();
    double k=findK(p1+1, p1+2), b=findB(p1+1, p1+2);
    double x=(-b*k+points[p1].x+points[p1].y*k)/(1+k*k);
    Point p2=Point(x, k*x+b);
    return p2;
  }

  double findK(int coord1, int coord2) {
    if (points[coord1].x - points[coord2].x == 0) return 0;
    double b = (points[coord1].y - points[coord2].y) /
        (points[coord1].x - points[coord2].x);
    return b;
  }

  double findB(int coord1, int coord2) {
    if (points[coord2].x - points[coord1].x == 0) return
points[coord1].y;
    double n = -((points[coord2].y - points[coord1].y) *
            points[coord1].x /
            (points[coord2].x - points[coord1].x)) +
        points[coord1].y;
   // print('$coord1 $coord2: n');
    return n;
  }

  void draw(
      Canvas mCanva, Size mSize, double widthOfCell, double
heightOfCell) {
    //drawPar=true;
    Paint paint = Paint()
      ..color = background
      ..strokeCap = StrokeCap.round
      ..strokeWidth = 1.0;

    // Координати чотирьох точок чотирикутника
    Offset point1 = Offset(mSize.width / 2 + points[0].x *
widthOfCell,
        mSize.height / 2 - points[0].y * heightOfCell);
    Offset point2 = Offset(mSize.width / 2 + points[1].x *
widthOfCell,
        mSize.height / 2 - points[1].y * heightOfCell);
    Offset point3 = Offset(mSize.width / 2 + points[2].x *
widthOfCell,
        mSize.height / 2 - points[2].y * heightOfCell);
    Offset point4 = Offset(mSize.width / 2 + points[3].x *
widthOfCell,
        mSize.height / 2 - points[3].y * heightOfCell);
```

```
    // Малюємо заливкований прямокутник
    Path path = Path()
      ..moveTo(point1.dx, point1.dy)
      ..lineTo(point2.dx, point2.dy)
      ..lineTo(point3.dx, point3.dy)
      ..lineTo(point4.dx, point4.dy)
      ..close(); // З'єднуємо останню точку з першою

    mCanva.drawPath(path, paint);

    paint=Paint()
      ..color = Colors.black;

    //паралелограм
    for (int i = 0; i < points.length; i++) {
      int i2 = i + 1;
      if (i2 > 3) i2 = 0;

      mCanva.drawLine(
          Offset(mSize.width / 2 + points[i].x * widthOfCell,
              mSize.height / 2 - points[i].y * heightOfCell),
          Offset(mSize.width / 2 + points[i2].x * widthOfCell,
              mSize.height / 2 - points[i2].y * heightOfCell),
          paint);
    }


    //висоти і продовдення висоти
    Point p1=points[findStupidAngle()], p2=findHeight(),
p3=points[findStupidAngle()+2];
    mCanva.drawLine(
        Offset(mSize.width / 2 + p1.x * widthOfCell,
            mSize.height / 2 - p1.y * heightOfCell),
        Offset(mSize.width / 2 + p2.x * widthOfCell,
            mSize.height / 2 - p2.y * heightOfCell),
        paint);
    mCanva.drawLine(
        Offset(mSize.width / 2 + p3.x * widthOfCell,
            mSize.height / 2 - p3.y * heightOfCell),
        Offset(mSize.width / 2 + p2.x * widthOfCell,
            mSize.height / 2 - p2.y * heightOfCell),
        paint);


    //діагоналі
    paint = Paint()..color = diag1;
    mCanva.drawLine(
        Offset(mSize.width / 2 + points[0].x * widthOfCell,
            mSize.height / 2 - points[0].y * heightOfCell),
        Offset(mSize.width / 2 + points[2].x * widthOfCell,
            mSize.height / 2 - points[2].y * heightOfCell),
        paint);
```

```
    paint = Paint()..color = diag2;
    mCanva.drawLine(
        Offset(mSize.width / 2 + points[1].x * widthOfCell,
            mSize.height / 2 - points[1].y * heightOfCell),
        Offset(mSize.width / 2 + points[3].x * widthOfCell,
            mSize.height / 2 - points[3].y * heightOfCell),
        paint);

  }
}
```

Файл: control_panel.dart

```
import 'package:flutter/material.dart';
import 'coord_field.dart';
import 'package:flutter_colorpicker/flutter_colorpicker.dart';
import 'styles.dart';
import 'canva.dart';
import 'parallelBrain.dart';

class ControllPanel extends StatefulWidget {
   myCanva canva;
  ControllPanel({required this.canva});
  static late List<CoordField> coordFields4X = [
    CoordField(hintText: '1 point: X'),
    CoordField(hintText: '2 point: X'),
    CoordField(hintText: '3 point: X'),
    CoordField(hintText: '4 point: X'),
  ];
  static late List<CoordField> coordFields4Y = [
    CoordField(hintText: '1 point: Y'),
    CoordField(hintText: '2 point: Y'),
    CoordField(hintText: '3 point: Y'),
    CoordField(hintText: '4 point: Y')
  ];

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

class _ControllPanelState extends State<ControllPanel> {
  late ParallelBrain brain;
  Color pickerColor = Color(0xff443a49);
  Color currentColor = Color(0xff443a49);
  Color currentColor2 = Color(0xff443a49);

  @override
  void initState() {
    super.initState();
    brain=ParallelBrain(context: context, canva: widget.canva);
  }
```

```
  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Expanded(
          flex: 9,
          child: Container(
            padding: EdgeInsets.all(5),
            margin: EdgeInsets.all(5),
            decoration: kFrame,
            child: Column(
              children: [
                Text(
                  'Enter 4 coordinates:',
                  style: kSubtitle,
                ),
                Expanded(
                  child: Row(children: [
                    Expanded(
                      child: ControllPanel.coordFields4X[0],
                    ),
                    Expanded(
                      child: ControllPanel.coordFields4Y[0],
                    )
                  ]),
                ),
                Expanded(
                  child: Row(
                    children: [
                      Expanded(child:
ControllPanel.coordFields4X[1]),
                      Expanded(child:
ControllPanel.coordFields4Y[1]),
                    ],
                  ),
                ),
                Expanded(
                  child: Row(
                    children: [
                      Expanded(
                          child: ControllPanel.coordFields4X[2]),
                      Expanded(
                          child: ControllPanel.coordFields4Y[2]),
                    ],
                  ),
                ),
                Expanded(
                  child: Row(
                    children: [
                      Expanded(
                          child: ControllPanel.coordFields4X[3]),
                      Expanded(
                          child: ControllPanel.coordFields4Y[3]),
```

```
                    ],
                  ),
                ),
                SizedBox(
                  height: 5,
                ),
                Expanded(
                  child: TextButton(
                      style: kButtonStyle,
                      onPressed: () {
                        setState(() {
                          brain.addPoints(4, currentColor,
currentColor2);
                        });
                      },
                      child: Text(
                        'Check and draw',
                        style: kTextStyle,
                      )),
                ),
              ],
            ),
          ),
        ),
        Expanded(
          flex: 8,
          child: Container(
            padding: EdgeInsets.all(5),
            margin: EdgeInsets.all(5),
            decoration: kFrame,
            child: Column(
              children: [
                Text(
                  'Enter 3 coordinates:',
                  style: kSubtitle,
                ),
                Expanded(
                  child: Row(children: [
                    Expanded(
                      child: ControllPanel.coordFields4X[0],
                    ),
                    Expanded(
                      child: ControllPanel.coordFields4Y[0],
                    )
                  ]),
                ),
                Expanded(
                  child: Row(
                    children: [
                      Expanded(child:
ControllPanel.coordFields4X[1]),
                      Expanded(child:
ControllPanel.coordFields4Y[1]),
```

```
                    ],
                  ),
                ),
                Expanded(
                  child: Row(
                    children: [
                      Expanded(child:
ControllPanel.coordFields4X[2]),
                      Expanded(child:
ControllPanel.coordFields4Y[2]),
                    ],
                  ),
                ),
                SizedBox(
                  height: 5,
                ),
                Expanded(
                  child: TextButton(
                      style: kButtonStyle,
                      onPressed: () {
                        brain.addPoints(3, currentColor,
currentColor2);
                      },
                      child: Text(
                        'Calculate and draw',
                        style: kTextStyle,
                      )),
                ),
              ],
            ),
          ),
        ),
        Expanded(
          flex: 4,
          child: Container(
            padding: EdgeInsets.all(5),
            margin: EdgeInsets.all(5),
            decoration: kFrame,
            child: Row(
              children: [
                Expanded(
                  child: Column(
                    children: [
                      Text(
                        'Color of 1 diagonal',
                        style: kSubtitle,
                      ),
                      Row(
                        children: [
                          Expanded(
                            child: TextButton(
                              style: kButtonStyle,
                              child: Text(
```

```
                                'Choose color',
                                style: kTextStyle,
                              ),
                              onPressed: () {
                                showDialog(
                                  context: context,
                                  builder: (BuildContext context)
{
                                    return AlertDialog(
                                      title: const Text(
                                        'Pick a color!',
                                        style: kSubtitle,
                                      ),
                                      content:
SingleChildScrollView(
                                        child: ColorPicker(
                                          pickerColor:
pickerColor,
                                          onColorChanged: (value)
{
                                            pickerColor = value;
                                          },
                                        ),
                                      ),
                                      actions: <Widget>[
                                        ElevatedButton(
                                          child: const Text(
                                            'Got it',
                                            style: TextStyle(
                                                color:
Color(0xFF303F9F)),
                                          ),
                                          onPressed: () {
                                            setState(() =>
                                                currentColor =
pickerColor);

Navigator.of(context).pop();
                                          },
                                        ),
                                      ],
                                    );
                                  },
                                );
                              },
                            ),
                          ),
                          Expanded(
                            child: Container(
                              //margin: EdgeInsets.all(5),
                              width: 50.0,
                              height: 50.0,
                              decoration: BoxDecoration(
```

```
                                shape: BoxShape.circle,
                                color: currentColor,
                              ),
                            ),
                          ),
                        ],
                      ),
                    ],
                  ),
                ),
                Expanded(
                  child: Column(
                    children: [
                      Text(
                        'Color of 2 diagonal',
                        style: kSubtitle,
                      ),
                      Row(
                        children: [
                          Expanded(
                            child: TextButton(
                              style: kButtonStyle,
                              child: Text(
                                'Choose color',
                                style: kTextStyle,
                              ),
                              onPressed: () {
                                showDialog(
                                  context: context,
                                  builder: (BuildContext context)
{
                                    return AlertDialog(
                                      title: const Text(
                                        'Pick a color!',
                                        style: kSubtitle,
                                      ),
                                      content:
SingleChildScrollView(
                                        child: ColorPicker(
                                          pickerColor:
pickerColor,
                                          onColorChanged: (value)
{
                                            pickerColor = value;
                                          },
                                        ),
                                      ),
                                      actions: <Widget>[
                                        ElevatedButton(
                                          child: const Text(
                                            'Got it',
                                            style: kButtonText),
                                          onPressed: () {
```

```
                                                setState(() =>
                                                    currentColor2 =
pickerColor);

Navigator.of(context).pop();
                                                  },
                                                ),
                                              ],
                                            );
                                          },
                                        );
                                      },
                                    ),
                                  ),
                                  Expanded(
                                    child: Container(
                                      //margin: EdgeInsets.all(5),
                                      width: 50.0,
                                      height: 50.0,
                                      decoration: BoxDecoration(
                                        shape: BoxShape.circle,
                                        color: currentColor2,
                                      ),
                                    ),
                                  ),
                                ],
                              ),
                            ],
                          ),
                        ),
                      ),
                    ],
                  ),
                )
              ],
            );
  }
}
```

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=-999;
  CoordField({required this.hintText});

  @override
```

```
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(4.0),
      child: TextField(
        inputFormatters: <TextInputFormatter>[
          FilteringTextInputFormatter.allow(RegExp(r'^-
?\d*\.?\d*')),
        ],
        textAlign: TextAlign.center,
        onChanged: (value) {
          try{
            val=double.parse(value);
          }
          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('Parallelogram')],
              ),
              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: parallelBrain.dart

```
import 'dart:math';

import 'package:flutter/material.dart';
import 'package:lab1/canva.dart';
import 'control_pannel.dart';
import 'point.dart';
import 'styles.dart';
import 'canva.dart';

class ParallelBrain {
  BuildContext context;
  myCanva canva;
  ParallelBrain({required this.context, required this.canva});

  bool ifThisIsIn2Q(List<Point> points){
    for(int i=0;i<points.length;i++){
      if(points[i].x>0 ||points[i].y<0){
        exceptionDialog('Incorrect input', 'It isn`t in second
quarter');
        return false;
      }
    }
    return true;
  }

  void addPoints(int num, Color col1, Color col2) {
    List<Point> tempPoints = [];
    try {
      for (int i = 0; i < num; i++) {
        if (ControllPanel.coordFields4X[i].val != -999 &&
            ControllPanel.coordFields4Y[i].val != -999) {
          tempPoints.add(Point(ControllPanel.coordFields4X[i].val,
              ControllPanel.coordFields4Y[i].val));
        } else {
          throw Exception();
        }
      }
      if(num==3) tempPoints.add(findLastPoint(tempPoints));
      if(ifThisIsIn2Q(tempPoints)) {
        canva.allFigures.add(Shape(points: tempPoints,
            diag1: col1,
```

```
            diag2: col2,
            background: getRandomColor()));
        checkIfParall();
      }
    } catch (e) {
      exceptionDialog('Incorrect data', 'Not points are
inputted');
    }
  }

  Point findLastPoint(List<Point> points){
    Point O=Point((points[0].x+points[2].x)/2,
(points[0].y+points[2].y)/2);
    return Point(O.x*2-points[1].x, O.y*2-points[1].y);
    //double x=O.x*2-points[1].x;
  }

  double k(int coord1, int coord2) {
    if (canva.allFigures.last.points[coord1].x -
            canva.allFigures.last.points[coord2].x ==
        0) return 0;
    double b = (canva.allFigures.last.points[coord1].y -
            canva.allFigures.last.points[coord2].y) /
        (canva.allFigures.last.points[coord1].x -
            canva.allFigures.last.points[coord2].x);
    print('k $coord1 $coord2: $b');
    return b;
  }

  double b(int coord1, int coord2) {
    if (canva.allFigures.last.points[coord2].x -
            canva.allFigures.last.points[coord1].x ==
        0) return canva.allFigures.last.points[coord1].y;
    double n = -((canva.allFigures.last.points[coord2].y -
                canva.allFigures.last.points[coord1].y) *
            canva.allFigures.last.points[coord1].x /
            (canva.allFigures.last.points[coord2].x -
                canva.allFigures.last.points[coord1].x)) +
        canva.allFigures.last.points[coord1].y;
    print('b $coord1 $coord2: $n');
    return n;
  }

  void checkIfParall() {
    try {
      if (k(1, 0) == k(3, 2) &&
          k(3, 0) == k(1, 2) &&
          b(1, 0) != b(2, 3) &&
          b(3, 0) != b(1, 2)) {
        canva.drawPar = true;
      } else if (k(2, 0) == k(3, 1) &&
          k(3, 0) == k(2, 1) &&
          b(2, 0) != b(3, 1) &&
```

```
          b(3, 0) != b(2, 1)) {
        canva.allFigures.last.points = [
          canva.allFigures.last.points[0],
          canva.allFigures.last.points[2],
          canva.allFigures.last.points[1],
          canva.allFigures.last.points[3]
        ];
        canva.drawPar = true;
      } else {
        throw Exception();
      }
    } catch (e) {
      exceptionDialog('Incorrect data',
          'Input correct points. Fields are empty or this figure
isn`t parallelogram');
    }
  }

  Color getRandomColor() {
    Random random = Random();
    return Color.fromRGBO(
      random.nextInt(256), // червоний
      random.nextInt(256), // зелений
      random.nextInt(256), // синій
      1.0, // прозорість
    );
  }

  void exceptionDialog(String title, String text) {
    showDialog(
      context: context, // Use the member variable
      builder: (BuildContext context) {
        return AlertDialog(
          title: Text(
            title,
          ),
          content: Text(text),
          actions: <Widget>[
            TextButton(
              onPressed: () {
                Navigator.of(context).pop();
              },
              child: Text(
                'OK',
                style: kButtonText,
              ),
            ),
          ],
        );
      },
    );
  }
}
```

Файл: point.dart

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

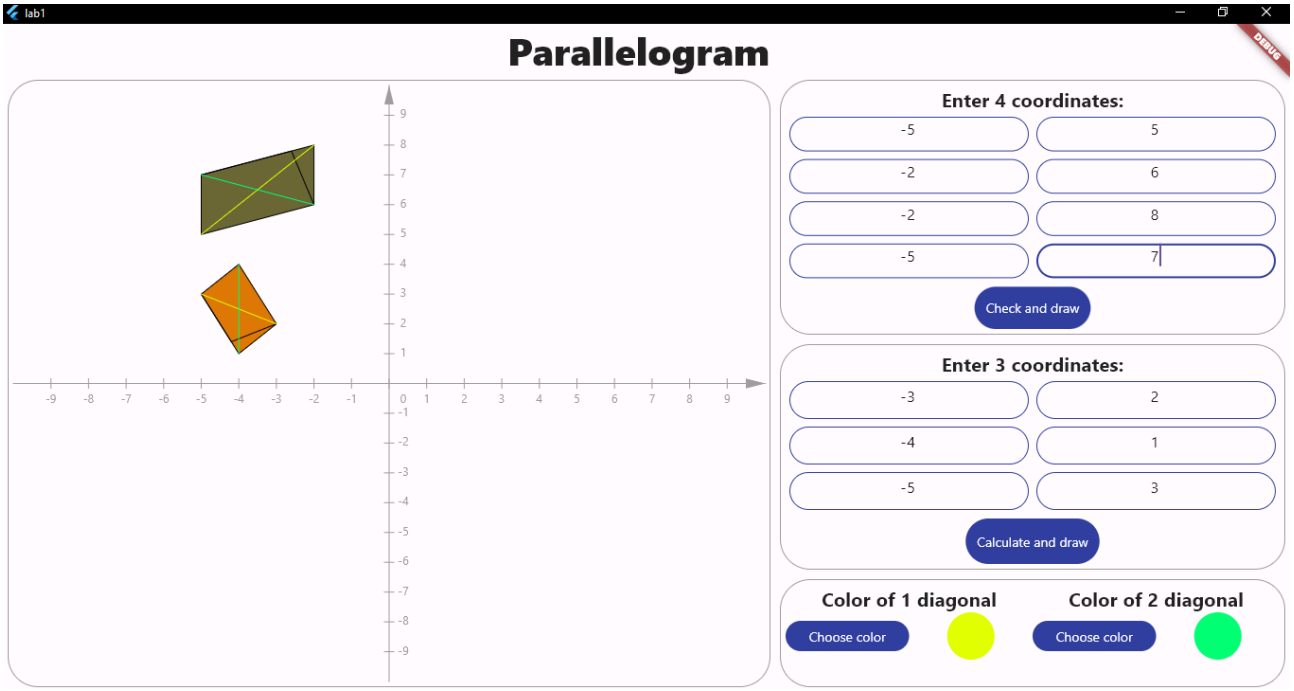

Рис. 1. Зображення паралелограма за введеними 3 і 4 координатами

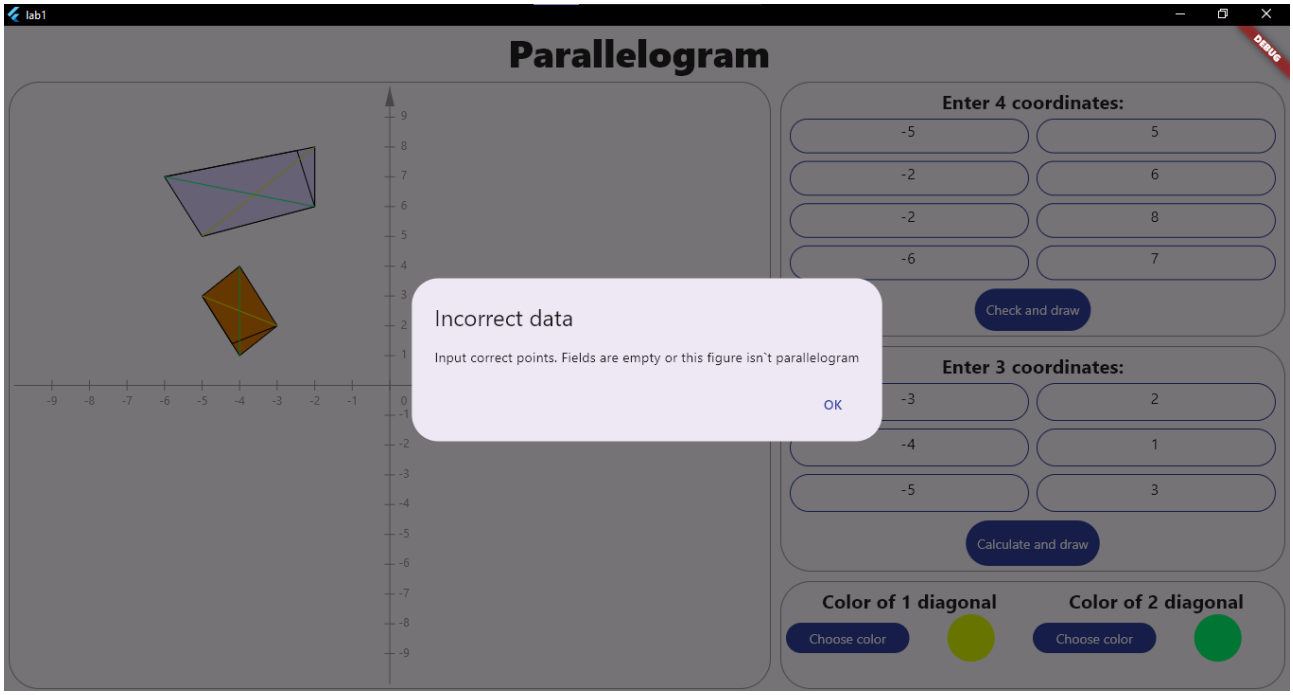

Рис. 2. Перевірка на некоректний ввід даних

Висновки

Під час виконання лабораторної роботи я навчилася будувати двомірні зображення, використавши для цього графічні примітиви мови Dart, пригадала основи лінійної алгебри, застосувавши свої знання на практиці, навчилася працювати з інструментом Canvas в фреймворку Flutter.