Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ: «Информатика и системы управления»
КАФЕДРА: «Теоретическая информатика и компьютерные технологии»

РК №1
«Визуализация двух отрезков и определение их пересечения»

по курсу
«Разработка мобильных приложений»

Выполнил:
студент группы ИУ9-72Б
Караник А.А.

Проверено:
Посевин Д.П.

Москва, 2024

Цель работы

Цель состоит в том, чтобы реализовать приложение на Flutter, которое отображает два отрезка по заданным координатам и определяет, пересекаются ли они, изменяя цвет: красный - если пересекаются, зеленый - если нет.

Реализация

Исходный код программы:

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Отрезки на плоскости',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: LineSegmentScreen(),
    );
  }
}

class LineSegmentScreen extends StatefulWidget {
  @override
  _LineSegmentScreenState createState() => _LineSegmentScreenState();
}

class _LineSegmentScreenState extends State<LineSegmentScreen> {
  final TextEditingController _x1Controller = TextEditingController();
  final TextEditingController _y1Controller = TextEditingController();
  final TextEditingController _x2Controller = TextEditingController();
  final TextEditingController _y2Controller = TextEditingController();
  final TextEditingController _x3Controller = TextEditingController();
  final TextEditingController _y3Controller = TextEditingController();
  final TextEditingController _x4Controller = TextEditingController();
  final TextEditingController _y4Controller = TextEditingController();

  bool? isIntersecting;

  int orientation(Point p, Point q, Point r) {
    num val = (q.y - p.y) * (r.x - q.x) - (q.x - p.x) * (r.y - q.y);
    if (val == 0) return 0;
    return (val > 0) ? 1 : 2;
  }

  bool doIntersect(Point p1, Point q1, Point p2, Point q2) {
    int o1 = orientation(p1, q1, p2);
    int o2 = orientation(p1, q1, q2);
    int o3 = orientation(p2, q2, p1);
    int o4 = orientation(p2, q2, q1);

    if (o1 != o2 && o3 != o4) return true;

    return false;
```

```
}

void checkIntersection() {
  double x1 = double.parse(_x1Controller.text);
  double y1 = double.parse(_y1Controller.text);
  double x2 = double.parse(_x2Controller.text);
  double y2 = double.parse(_y2Controller.text);
  double x3 = double.parse(_x3Controller.text);
  double y3 = double.parse(_y3Controller.text);
  double x4 = double.parse(_x4Controller.text);
  double y4 = double.parse(_y4Controller.text);

  Point p1 = Point(x1, y1);
  Point p2 = Point(x2, y2);
  Point p3 = Point(x3, y3);
  Point p4 = Point(x4, y4);

  bool intersect = doIntersect(p1, p2, p3, p4);

  setState(() {
    isIntersecting = intersect;
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Проверка пересечения отрезков'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        children: [
          Row(
            children: [
              Expanded(
                  child: TextField(
                      controller: _x1Controller,
                      decoration: InputDecoration(labelText: 'x1'))),
              Expanded(
                  child: TextField(
                      controller: _y1Controller,
                      decoration: InputDecoration(labelText: 'y1'))),
            ],
          ),
          Row(
            children: [
              Expanded(
                  child: TextField(
                      controller: _x2Controller,
                      decoration: InputDecoration(labelText: 'x2'))),
              Expanded(
                  child: TextField(
                      controller: _y2Controller,
                      decoration: InputDecoration(labelText: 'y2'))),
            ],
          ),
          Row(
            children: [
              Expanded(
                  child: TextField(
                      controller: _x3Controller,
                      decoration: InputDecoration(labelText: 'x3'))),
              Expanded(
```

```
                    child: TextField(
                        controller: _y3Controller,
                        decoration: InputDecoration(labelText: 'y3'))),
              ],
            ),
            Row(
              children: [
                Expanded(
                    child: TextField(
                        controller: _x4Controller,
                        decoration: InputDecoration(labelText: 'x4'))),
                Expanded(
                    child: TextField(
                        controller: _y4Controller,
                        decoration: InputDecoration(labelText: 'y4'))),
              ],
            ),
            ElevatedButton(
              onPressed: checkIntersection,
              child: Text('Построить'),
            ),
            SizedBox(height: 20),
            Expanded(
              child: CustomPaint(
                painter: LinePainter(
                  x1: double.tryParse(_x1Controller.text) ?? 0,
                  y1: double.tryParse(_y1Controller.text) ?? 0,
                  x2: double.tryParse(_x2Controller.text) ?? 0,
                  y2: double.tryParse(_y2Controller.text) ?? 0,
                  x3: double.tryParse(_x3Controller.text) ?? 0,
                  y3: double.tryParse(_y3Controller.text) ?? 0,
                  x4: double.tryParse(_x4Controller.text) ?? 0,
                  y4: double.tryParse(_y4Controller.text) ?? 0,
                  isIntersecting: isIntersecting ?? false,
                ),
                child: Container(),
              ),
            ),
          ],
        ),
      ),
    );
  }
}

class LinePainter extends CustomPainter {
  final double x1, y1, x2, y2, x3, y3, x4, y4;
  final bool isIntersecting;

  LinePainter({
    required this.x1,
    required this.y1,
    required this.x2,
    required this.y2,
    required this.x3,
    required this.y3,
    required this.x4,
    required this.y4,
    required this.isIntersecting,
  });

  @override
  void paint(Canvas canvas, Size size) {
    final paint = Paint()
      ..color = isIntersecting ? Colors.red : Colors.green
```

```
      ..strokeWidth = 3;

    canvas.drawLine(Offset(x1, y1), Offset(x2, y2), paint);

    canvas.drawLine(Offset(x3, y3), Offset(x4, y4), paint);
  }

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

Результаты

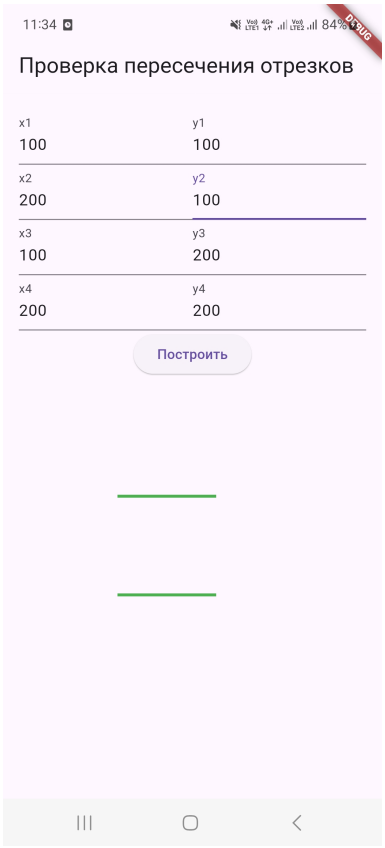

Рис. 1: результаты

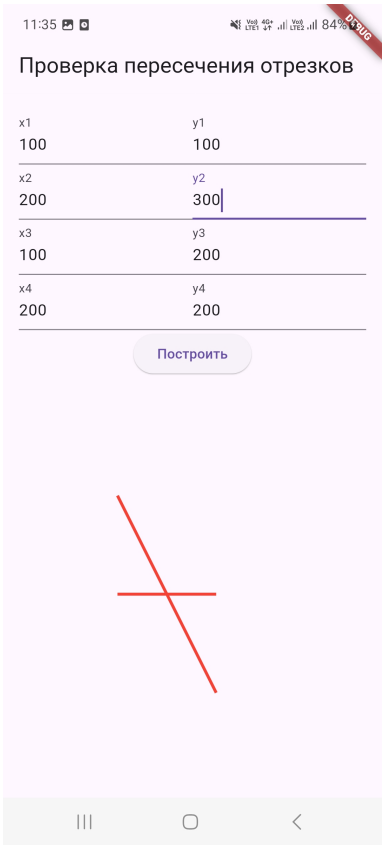

Рис. 2: результаты

Вывод

В результате выполнения лабораторной работы было создано приложение на Flutter, которое успешно визуализирует два отрезка на плоскости по заданным координатам. Программа корректно определяет пересечение отрезков и меняет их цвет.