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»

Лабораторная работа № 1

по курсу «Разработка мобильных приложений»

«Графический пользовательский интерфейс в Dart»

Студентка группы ИУ9-72Б Самохвалова П. С.

Преподаватель Посевин Д. П.

Москва 2023

1 Цель работы

Научиться создавать приложения с графическим пользовательским интерфейсом с использованием фреймворка Flutter на языке программирования Dart.

2 Задание

В течение лабораторной работы нужно разработать программу, рисующую на экране мобильного устройства одно из изображений, перечисленных в таблице ниже. Программа должна иметь графический пользовательский интерфейс, через который пользователь может задавать параметры изображения. Изображение должно перерисовываться автоматически при изменении любого параметра. Значения параметров, обозначенных в таблице латинскими буквами, представляют собой неотрицательные целые числа. Когда в описании изображения говорится о выборе цвета, подразумевается выбор из нескольких предопределённых альтернатив (например, красный, зелёный или синий).

Индивидуальный вариант:

Прямоугольный треугольник выбранного цвета с катетами a и b, вписанный в окружность.

3 Практическая реализация

Исходный код программы представлен в листинге 1.

Листинг 1: Графический пользовательский интерфейс в Dart

```
1  import 'dart:ui';
2  import 'package:flutter/material.dart';
3  import 'dart:math' as math;
4
5  void main() => runApp(MyApp());
6
7  class MyApp extends StatelessWidget {
8    @override
9    Widget build(BuildContext context) {
10     return MaterialApp(
11       title: 'Flutter Custom Painter',
12       theme: ThemeData(
```

```
13          primarySwatch: Colors.blue,
14        ),
15        home: MyPainter(),
16      );
17    }
18  }
19
20  class MyPainter extends StatefulWidget {
21    @override
22    _MyPainterState createState() => _MyPainterState();
23  }
24
25  class _MyPainterState extends State<MyPainter> {
26    var _catetA = 100.0;
27    var _catetB = 100.0;
28    Color selectedColor = Colors.teal;
29
30    @override
31    Widget build(BuildContext context) {
32      return Scaffold(
33        appBar: AppBar(
34          title: Text('Triangle'),
35        ),
36        body: SafeArea(
37          child: Column(
38            crossAxisAlignment: CrossAxisAlignment.start,
39            children: <Widget>[
40              RadioListTile(
41                title: Text('Red'),
42                value: Colors.red,
43                groupValue: selectedColor,
44                onChanged: (value) {
45                  setState(() {
46                    selectedColor = value!;
47                  });
48                },
49              ),
50              RadioListTile(
51                title: Text('Yellow'),
52                value: Colors.yellow, //unique value
53                groupValue: selectedColor ,
54                onChanged: (value) {
55                  setState(() {
56                    selectedColor = value!;
57                  });
58                },
```

```
59              ),
60              RadioListTile(
61                title: Text('Green'),
62                value: Colors.green,
63                groupValue: selectedColor,
64                onChanged: (value) {
65                  setState(() {
66                    selectedColor = value!;
67                  });
68                },
69              ),
70
71              Expanded(
72                child: CustomPaint(
73                  painter: ShapePainter(_catetA, _catetB, selectedColor),
74                  child: Container(),
75                ),
76              ),
77
78              Padding(
79                padding: const EdgeInsets.only(left: 16.0),
80                child: Text('Cathet A'),
81              ),
82              Slider(
83                value: _catetA,
84                min: - MediaQuery.of(context).size.height / 2,
85                max: MediaQuery.of(context).size.height / 2,
86                onChanged: (value) {
87                  setState(() {
88                    _catetA = value;
89                  });
90                },
91              ),
92
93              Padding(
94                padding: const EdgeInsets.only(left: 16.0),
95                child: Text('Cathet B'),
96              ),
97              Slider(
98                value: _catetB,
99                min: - MediaQuery.of(context).size.width / 2,
100               max: MediaQuery.of(context).size.width / 2,
101               onChanged: (value) {
102                 setState(() {
103                   _catetB = value;
104                 });
```

```
105                 },
106               ),
107
108             ],
109           ),
110         ),
111       );
112   }
113 }
114
115 // FOR PAINTING POLYGONS
116 class ShapePainter extends CustomPainter {
117
118   final double catetA;
119   final double catetB;
120   final Color shapeColor;
121   ShapePainter(this.catetA, this.catetB, this.shapeColor);
122
123   @override
124   void paint(Canvas canvas, Size size) {
125     var paint = Paint()
126       ..color = shapeColor
127       ..strokeWidth = 5
128       ..strokeCap = StrokeCap.round
129       ..style = PaintingStyle.stroke;
130
131     var catA = catetA;
132     var catB = catetB;
133     var hypo = math.sqrt(math.pow(catA, 2) + math.pow(catB, 2));
134     var rad = hypo / 2 ;
135
136     var startAX = size.width / 2;
137     var startAY = size.height / 2;
138     Offset startA = Offset(startAX , startAY);
139     var endAX = size.width / 2;
140     var endAY = size.height / 2 - catA;
141     Offset endA = Offset(endAX, endAY);
142     canvas.drawLine(startA, endA, paint);
143
144     var startBX = size.width / 2;
145     var startBY = size.height / 2;
146     Offset startB = Offset(startBX , startBY);
147     var endBX = size.width / 2 + catB;
148     var endBY = size.height / 2;
149     Offset endB = Offset(endBX, endBY);
150     canvas.drawLine(startB, endB, paint);
```

```
151
152       canvas.drawLine(endA, endB, paint);
153
154       Offset center = Offset((endAX + endBX) / 2, (endAY + endBY) / 2);
155       canvas.drawCircle(center, rad, paint);
156
157     }
158
159     @override
160     bool shouldRepaint(CustomPainter oldDelegate) {
161       return false;
162     }
163 }
```

4 Результаты

Результаты работы программы представлены на рисунках 1 – 5.

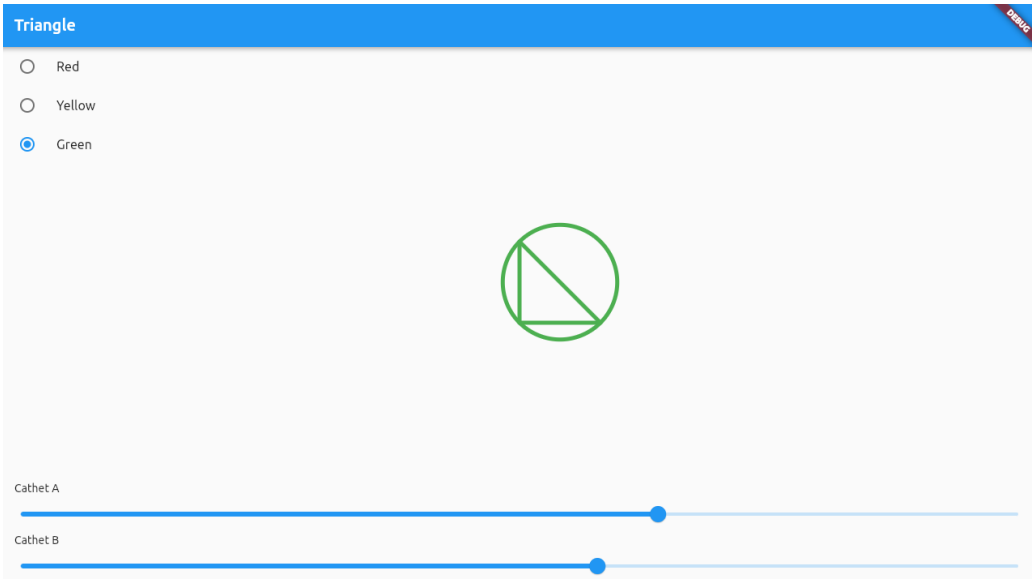

Рис. 1 — Интерфейс приложения

Ссылка на DartPad:
https://dartpad.dev/?id=560f204e649dac497f0a50255914808e

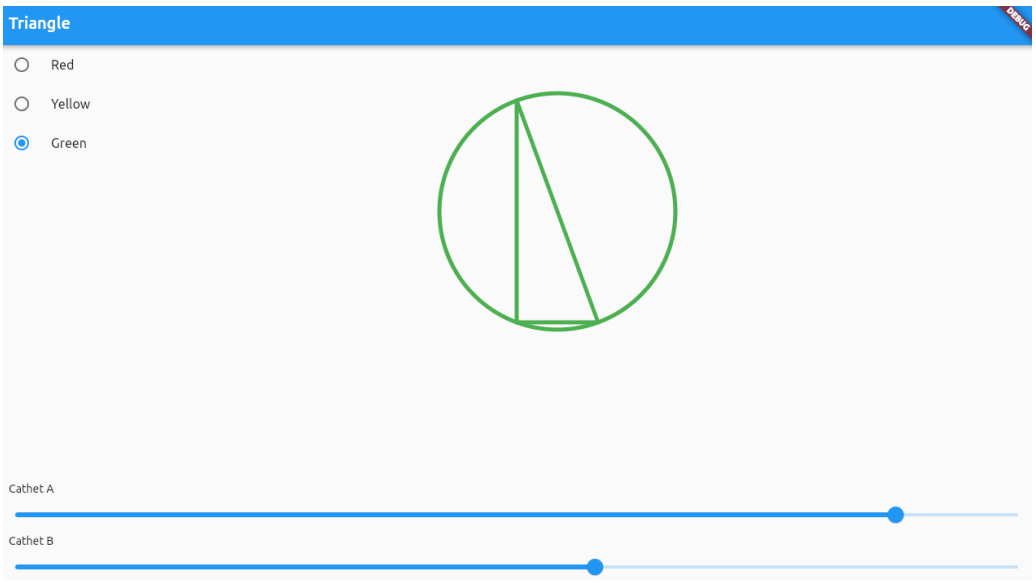

Рис. 2 — Изменение длины катета

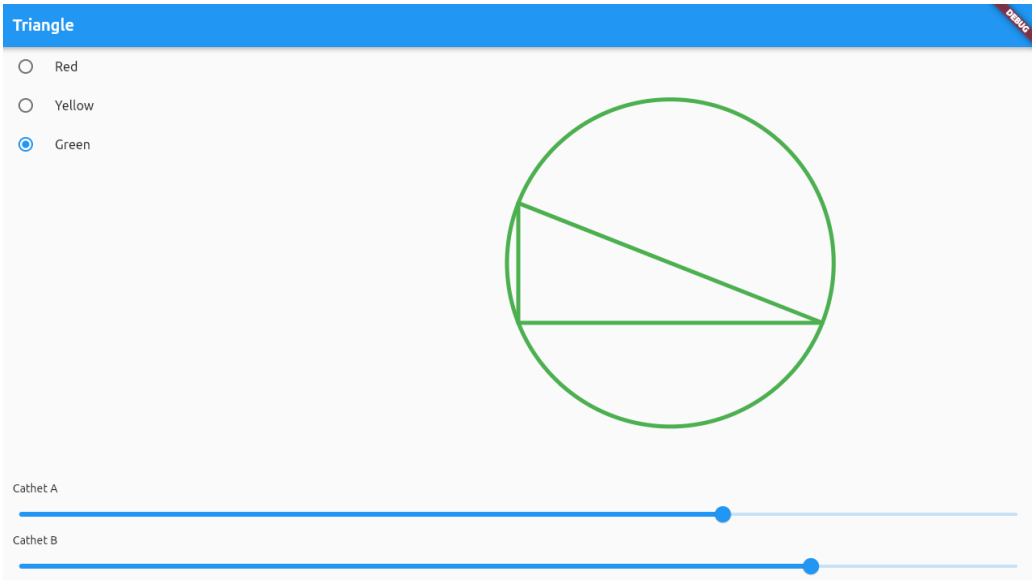

Рис. 3 — Изменение длины катета

5 Выводы

В результате выполнения лабораторной работы было создано приложения с графическим пользовательским интерфейсом с использованием фреймворка Flutter на языке программирования Dart.

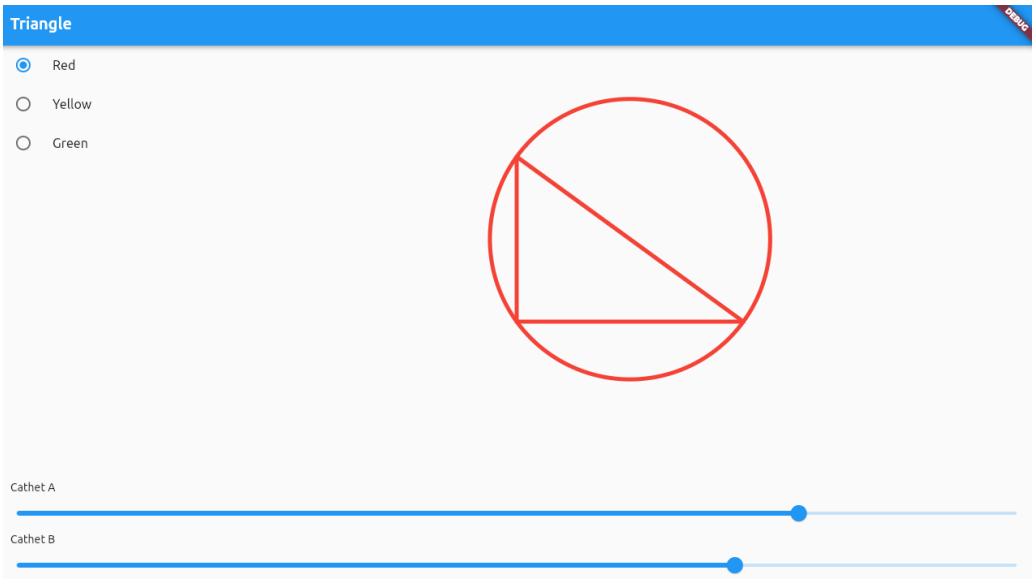

Рис. 4 — Изменение цвета

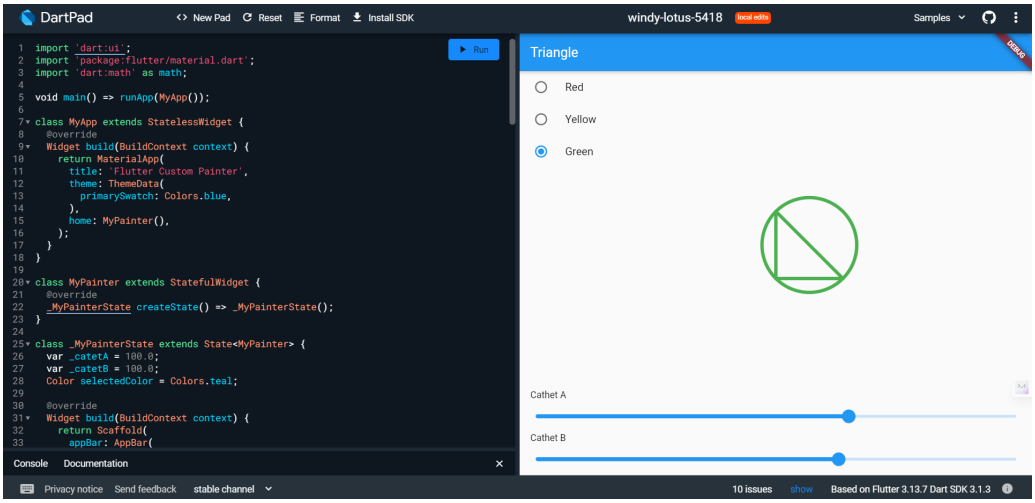

Рис. 5 — Результат в DartPad