**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

**отчет**

**по лабораторной работе №1**

**по дисциплине «Компьютерная графика»**

**Тема: "Исследование математических методов представления и преобразования графических объектов на плоскости и в пространстве ”**

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

Санкт-Петербург

2019

**Задание:**

Поворот плоского объекта относительно произвольной точки плоскости на заданный угол. Необходимо предусмотреть возможность редактирования положения точки.

**Теоретическая часть:**

Для поворота вектора (x, y) формула поворота будет выглядеть следующим образом: X = x \* cos(a) - y \* sin(a); Y = y \* cos(a) + x \* sin(a);

Для поворота вектора (x, y) вокруг произвольной точки (x0, y0) на плоскости формула следующая: X = x0 + (x - x0) \* cos(a) - (y - y0) \* sin(a);  
Y = y0 + (y - y0) \* cos(a) + (x - x0) \* sin(a);

**Реализация:**

Добавим произвольную точку и при её перемещении будем переносить многоугольник относительно этой точки как нового начала координат. Для поворота приведем каждый вектор составляющий многоугольник к началу координат (0; 0) выполним поворот с помощью матрицы поворота и перенесем обратно к выбранному ранее началу координат.

Для изменения координат точки и угла поворота используйте слайдеры (ползунки).

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

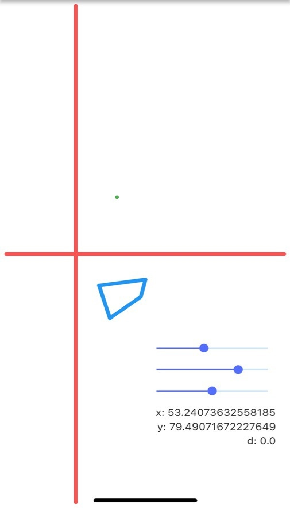


Рис. 1. До поворота

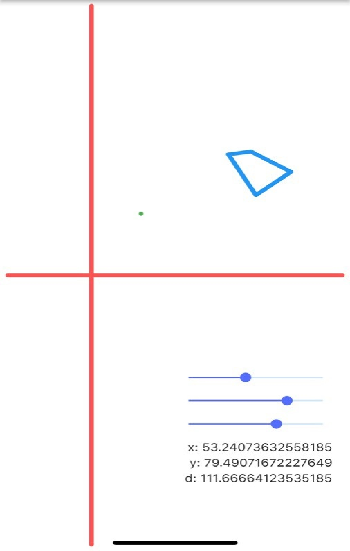


Рис. 2. После поворота на 90 градусов

**Код:**

import 'package:flutter/material.dart';

import 'dart:ui';

import 'dart:math' as math;

double x1, x2, x3, x4, x5, x6, x7, x8;

double y1, y2, y3, y4, y5, y6, y7, y8;

double dotX, dotY, ugol;

void main() => runApp(new MyApp());

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return new MaterialApp(

title: 'Graphics',

home: HomePage(),

theme: new ThemeData(

canvasColor: Colors.transparent,

),

);

}

}

class HomePage extends StatefulWidget {

final Widget child;

HomePage({Key key, this.child}) : super(key: key);

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

}

class \_HomePageState extends State<HomePage> {

final xController = TextEditingController();

final yController = TextEditingController();

final ugolController = TextEditingController();

@override

void initState() {

x1 = 30;

x2 = 90;

x3 = 90;

x4 = 84;

x5 = 84;

x6 = 44;

x7 = 44;

x8 = 30;

y1 = -44;

y2 = -36;

y3 = -36;

y4 = -60;

y5 = -60;

y6 = -90;

y7 = -90;

y8 = -44;

dotX = 0;

dotY = 0;

ugol = 0;

super.initState();

}

@override

void dispose() {

// Clean up the controller when the Widget is disposed

xController.dispose();

yController.dispose();

ugolController.dispose();

super.dispose();

}

@override

Widget build(BuildContext context) {

return new Scaffold(

backgroundColor: Colors.white,

appBar: AppBar(

title: Text('Поворот относительно точки под углом'),

centerTitle: true,

backgroundColor: Colors.blueAccent,

),

body: Padding(

padding: EdgeInsets.all(8.0),

child: Stack(

children: <Widget>[

CustomPaint(

painter: ShapesPainter(),

child: Container(

height: 700,

),

),

new Positioned(

bottom: 200,

right: 5,

child: Slider(

value: dotX,

activeColor: Colors.indigoAccent,

min: 0,

max: 125,

onChanged: (value) {

setState(() {

dotX = value;

calculate();

});

},

),

),

new Positioned(

bottom: 170,

right: 5,

child: Slider(

value: dotY,

activeColor: Colors.indigoAccent,

min: -170.0,

max: 170,

onChanged: (value) {

setState(() {

dotY = value;

calculate();

});

},

),

),

new Positioned(

bottom: 140,

right: 5,

child: Slider(

value: ugol,

activeColor: Colors.indigoAccent,

min: -360.0,

max: 360,

onChanged: (value) {

setState(() {

ugol = value;

calculate();

});

},

),

),

new Positioned(

bottom: 100,

right: 10,

child: new Text("x: " + dotX.toString() + '\n'),

),

new Positioned(

bottom: 80,

right: 10,

child: new Text("y: " + dotY.toString() + '\n'),

),

new Positioned(

bottom: 60,

right: 10,

child: new Text("d: " + ugol.toString() + '\n'),

),

],

),

),

);

}

void calculate() {

double kukol = ugol \* math.pi / 180;

x1 = 30 - dotX;

x2 = 90 - dotX;

x3 = 90 - dotX;

x4 = 84 - dotX;

x5 = 84 - dotX;

x6 = 44 - dotX;

x7 = 44 - dotX;

x8 = 30 - dotX;

y1 = -44 - dotY;

y2 = -36 - dotY;

y3 = -36 - dotY;

y4 = -60 - dotY;

y5 = -60 - dotY;

y6 = -90 - dotY;

y7 = -90 - dotY;

y8 = -44 - dotY;

double temp = x1;

x1 = x1 \* math.cos(kukol) - y1 \* math.sin(kukol);

y1 = temp \* math.sin(kukol) + y1 \* math.cos(kukol);

temp = x2;

x2 = x2 \* math.cos(kukol) - y2 \* math.sin(kukol);

y2 = temp \* math.sin(kukol) + y2 \* math.cos(kukol);

temp = x3;

x3 = x3 \* math.cos(kukol) - y3 \* math.sin(kukol);

y3 = temp \* math.sin(kukol) + y3 \* math.cos(kukol);

temp = x4;

x4 = x4 \* math.cos(kukol) - y4 \* math.sin(kukol);

y4 = temp \* math.sin(kukol) + y4 \* math.cos(kukol);

temp = x5;

x5 = x5 \* math.cos(kukol) - y5 \* math.sin(kukol);

y5 = temp \* math.sin(kukol) + y5 \* math.cos(kukol);

temp = x6;

x6 = x6 \* math.cos(kukol) - y6 \* math.sin(kukol);

y6 = temp \* math.sin(kukol) + y6 \* math.cos(kukol);

temp = x7;

x7 = x7 \* math.cos(kukol) - y7 \* math.sin(kukol);

y7 = temp \* math.sin(kukol) + y7 \* math.cos(kukol);

temp = x8;

x8 = x8 \* math.cos(kukol) - y8 \* math.sin(kukol);

y8 = temp \* math.sin(kukol) + y8 \* math.cos(kukol);

x1 += dotX;

x2 += dotX;

x3 += dotX;

x4 += dotX;

x5 += dotX;

x6 += dotX;

x7 += dotX;

x8 += dotX;

y1 += dotY;

y2 += dotY;

y3 += dotY;

y4 += dotY;

y5 += dotY;

y6 += dotY;

y7 += dotY;

y8 += dotY;

}

}

class ShapesPainter extends CustomPainter {

Paint \_paint = new Paint()

..color = Colors.blueAccent

..strokeCap = StrokeCap.round

..isAntiAlias = true

..strokeWidth = 5.0;

@override

void paint(Canvas canvas, Size size) {

// ось x

canvas.drawLine(

(new Offset(0, size.height / 2)),

new Offset(size.width, size.height / 2),

\_paint..color = Colors.redAccent);

// ось y

canvas.drawLine(

(new Offset(size.width / 4, 0)),

new Offset(size.width / 4, size.height),

\_paint..color = Colors.redAccent);

// Центр

var x = size.width / 4;

var y = size.height / 2;

// Точка

canvas.drawPoints(

PointMode.points,

[

Offset(x + dotX, y - dotY),

],

\_paint..color = Colors.green);

// Прямоугольник

doRect(canvas, x, y);

}

// A ------- B

// | |

// | |

// D ------- C

doRect(Canvas canvas, double x, double y) {

canvas.drawLine(new Offset(x + x1, y - y1), new Offset(x + x2, y - y2),

\_paint..color = Colors.blue); // AB

canvas.drawLine(new Offset(x + x3, y - y3), new Offset(x + x4, y - y4),

\_paint..color = Colors.blue); // BC

canvas.drawLine(new Offset(x + x5, y - y5), new Offset(x + x6, y - y6),

\_paint..color = Colors.blue); // CD

canvas.drawLine(new Offset(x + x7, y - y7), new Offset(x + x8, y - y8),

\_paint..color = Colors.blue); // DA

}

@override

bool shouldRepaint(CustomPainter oldDelegate) => true;

}