Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення

**ЗВІТ**

Про виконання лабораторної роботи №3

з дисципліни «Комп’ютерна графіка»

на тему “Побудова фрактальних зображень”

**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.** Побудова фрактальних зображень.

**Мета роботи.** Вивчити алгоритми побудови фракталів та навчитися їх програмно реалізовувати.

**Теоретичні відомості**

Фрактал Жюлія є фракталом, який генерується ітеративним обчисленням комплексних чисел за допомогою специфічного математичного виразу. Цей фрактал є частиною ширшої сім'ї фракталів, відомих як "множина Мандельброта". Зв'язок між ними полягає в тому, що кожна точка на площині Мандельброта визначає початкову точку для фракталу Жюлія.

У фракталі Жюлія використовується математична функція f(z)=z\*z+c, де z - це комплексне число, що ітеративно оновлюється, а c - це константа, яка визначається за допомогою координат на площині.

Для кожної точки фракталу обчислюється кількість ітерацій, яку потрібно зробити, щоб визначити, чи належить точка множині, і відповідно до цього призначається колір. Цей процес дозволяє візуалізувати складні структури фракталу, відображаючи збіжність або розбіжність обчислень.

Множина Мандельброта, з іншого боку, визначається як множина значень параметра c, для яких ітераційний процес збігається, тобто залишається обмеженим. Кожна точка на площині Мандельброта представляє собою початкову точку для фракталу Жюлія.

Фрактал Жюлія залежить від комплексних чисел - чисел, які мають як реальну, так і "уявну" (імагінарну) складову i, де i визначається як квадратний корінь з -1. Комплексне число формально можна виразити як:

c =r+bi

Де c - комплексне число, r - реальна складова, а b - уявна складова. Щоб створити обмежений набір, спочатку ми створюємо математичну функцію f(z), яка приймає комплексне число. Простим прикладом такої функції є наступне рівняння:

z=z\*z+c

Де c є сталим комплексним числом. Комплексне число z може бути оновлено ітеративно (тут визначено як F(z)):

* Ініціалізація змінної комплексного числа z.
* Ітеративне оновлення значення z на основі функції f(z).

Часто ми встановлюємо порог, щоб запобігти нескінченній ітерації, який може бути одним або обома наступними:

* Ми перевищили значення z (у вищенаведених прикладах ітерація припиняється, коли абсолютне значення z перевищує
* Ми перевищили попередньо визначену кількість ітерацій. На основі будь-якого з цих методів z може бути визначено як обмежений або необмежений (ітерація прямує до нескінченності).

Використана колірна схема генерує кольори таким чином, що на основі обраного користувачем кольору генерується новий, завдяки додаванню до його компонент R, G, B певного числа. Це число вираховується залежно до кількості ітерацій, які були пройдені перш ніж функція почне прямувати до нескінченості. Кількість ітерацій переводиться в число від 0 до 255. Саме воно додається до R, G та B, помноженого на 1, 2 та 3 відповідно.

**Постановка завдання**

Написати програму для візуалізації фракталів та створити з її допомогою галерею фракталів. Програма повинна відповідати таким вимогам:

1. Універсальність алгоритмів побудови фракталів\*.
2. Масштабування зображень алгебраїчних фракталів.
3. Збереження фрактальних зображень у файлах.
4. Зручний інтерфейс користувача

**Індивідуальний варіант**

Фрактал Дракон Хартера-Хейтуея, фрактал Жюлія f(z)=z\*z+c

**Текст програми**

Файл: canva.dart

import 'dart:math';  
import 'dart:ui';  
  
import 'package:flutter/gestures.dart';  
import 'package:flutter/material.dart';  
import 'package:lab3\_2/main\_screen.dart';  
  
enum Direction { Left, Right }  
  
class myCanva extends CustomPainter {  
 Color startColor = Colors.black, plane = Colors.grey;  
 late Canvas mCanvas;  
 late Size mSize;  
 double widthOfCell = 1;  
 double heightOfCell = 1;  
 int depth = 0;  
 double lenght = 10, x = -5, y = 0, angle = 0;  
 late double rX, rY;  
 double coord=2;  
 double mouseX=0, mouseY=0;  
 double ca = - 0.70176, cb = - 0.3842;  
 double resol=10;  
  
 @override  
 void paint(Canvas canvas, Size size) {  
 final Paint paint = Paint()  
 ..color = startColor  
 ..strokeCap = StrokeCap.round  
 ..strokeWidth = 1.0;  
 this.mCanvas = canvas;  
 this.mSize = size;  
 widthOfCell = mSize.width / 20;  
 heightOfCell = mSize.height / 20;  
  
 rX = size.width / 2 + x \* widthOfCell;  
 rY = size.height / 2 - y \* heightOfCell;  
  
 if(!mainScreen.type){  
 drawDragonTriangle(canvas, paint, depth, Direction.Right, rX, rY,  
 lenght \* widthOfCell, angle, startColor);  
 }else{  
 drawJuliaFractal(canvas, mSize);  
 }  
  
 }  
  
 Future<void> drawDragonTriangle(  
 Canvas canvas,  
 Paint paint,  
 int level,  
 Direction turnTowards,  
 double x1,  
 double y1,  
 double dx,  
 double dy,  
 Color startColor) async {  
 if (level <= 0) {  
 Path path = Path();  
 path.moveTo(x1, y1);  
 path.lineTo(x1 + dx, y1 + dy);  
 double nx = dx / 2;  
 double ny = dy / 2;  
 double dx2 = -ny + nx;  
 double dy2 = nx + ny;  
 double x2, y2;  
  
 if (turnTowards == Direction.Right) {  
 x2 = x1 + dx2;  
 y2 = y1 + dy2;  
 } else {  
 x2 = x1 + dy2;  
 y2 = y1 - dx2;  
 }  
 path.lineTo(x2, y2);  
 path.close();  
 paint.color = startColor;  
  
 canvas.drawPath(path, paint);  
 } else {  
 double nx = dx / 2;  
 double ny = dy / 2;  
 double dx2 = -ny + nx;  
 double dy2 = nx + ny;  
 double x2, y2;  
  
 List<Future<void>> futures = [];  
  
 if (turnTowards == Direction.Right) {  
 futures.add(drawDragonTriangle(canvas, paint, level - 1, Direction.Right, x1, y1,  
 dx2, dy2, startColor.withOpacity(0.1)));  
 x2 = x1 + dx2;  
 y2 = y1 + dy2;  
 futures.add(drawDragonTriangle(canvas, paint, level - 1, Direction.Left, x2, y2,  
 dy2, -dx2, startColor.withOpacity(0.3)));  
 } else {  
 futures.add(drawDragonTriangle(canvas, paint, level - 1, Direction.Right, x1, y1,  
 dy2, -dx2, startColor.withOpacity(0.6)));  
 x2 = x1 + dy2;  
 y2 = y1 - dx2;  
 futures.add(drawDragonTriangle(canvas, paint, level - 1, Direction.Left, x2, y2,  
 dx2, dy2, startColor.withOpacity(0.9)));  
 }  
  
 await Future.wait(futures);  
 }  
 }  
  
 double map(double value, double min1, double max1, double min2, double max2) {  
 return min2 + ((value - min1) / (max1 - min1)) \* (max2 - min2);  
 }  
  
  
 void drawJuliaFractal(Canvas canvas, Size size) {  
 final int maxIterations = 50;  
 final double startX = 0;  
 final double endX = size.width;  
 final double startY = 0;  
 final double endY = size.height;  
  
 final int chunkSize = 50;  
  
 Future<void> parallelExecution() async {  
 List<Future> futures = [];  
  
 if(mouseY!=0 || mouseX!=0){  
 ca=map(mouseX, 0, size.width, -1, 1);  
 cb=map(mouseY, 0, size.height, -1, 1);  
 }  
  
 for (double x = startX; x < endX; x += chunkSize) {  
 for (double y = startY; y < endY; y += chunkSize) {  
 futures.add(\_computeChunk(x, y, chunkSize, maxIterations, canvas, size));  
 }  
 }  
  
 await Future.wait(futures);  
 print('end of painting with resol: $resol ca:$ca, cb: $cb');  
 }  
  
 parallelExecution();  
 }  
  
 Future<void> \_computeChunk(double startX, double startY, int chunkSize, int maxIterations, Canvas canvas, Size size) async {  
 for (double x = startX; x < startX + chunkSize; x+=resol) {  
 for (double y = startY; y < startY + chunkSize; y+=resol) {  
 double a = map(x.toDouble(), 0, size.width.toDouble(), -coord, coord);  
 double b = map(y.toDouble(), 0, size.height.toDouble(), -coord, coord);  
  
  
 int n = 0, z = 0;  
 while (n < maxIterations) {  
 double aa = a \* a;  
 double bb = b\*b;  
 double ab2=2\*a\*b;  
  
 a = aa - bb + ca;  
 b = ab2 + cb;  
  
 if (aa\*aa+bb\*bb > 100) {  
 break;  
 }  
 n++;  
 }  
  
 double col = map(n.toDouble(), 0, maxIterations.toDouble(), 0, 255);  
 if (n == maxIterations) col = 0;  
  
 Paint paint = Paint()..color = generateColorShade(startColor, col.toInt());  
 Rect rect=Rect.fromLTWH(x+resol, y+resol, x, y);  
 canvas.drawRect(rect, paint);  
  
 }  
 }  
 }  
  
 Color generateColorShade(Color baseColor, int intensity) {  
 // Отримуємо значення каналів RGB базового кольору  
 int red = baseColor.red;  
 int green = baseColor.green;  
 int blue = baseColor.blue;  
  
 red = (red + intensity).clamp(  
 0, 255); // Обмежуємо значення до діапазону [0, 255]  
 green = (green + 3\*intensity).clamp(0, 255);  
 blue = (blue + 2\*intensity).clamp(0, 255);  
  
 // Повертаємо новий колір  
 return Color.fromARGB(255, red, green, blue);  
 }  
  
 @override  
 bool shouldRepaint(CustomPainter oldDelegate) {  
 return false;  
 }  
}

Файл: control\_panel.dart

import 'package:flutter/material.dart';  
import 'package:lab3\_2/main\_screen.dart';  
import 'coord\_field.dart';  
import 'package:flutter\_colorpicker/flutter\_colorpicker.dart';  
import 'styles.dart';  
import 'canva.dart';  
  
class ControllPanel extends StatefulWidget {  
 myCanva canva;  
 ControllPanel({required this.canva});  
  
 @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(  
 mainScreen.type?'Resolution: ${1/widget.canva.resol}':  
 'Depth: ${widget.canva.depth}',  
 style: kSubtitle,  
 ),  
 ),  
 ),  
 Expanded(  
 child: IconButton(  
 style: kButtonStyle,  
 onPressed: () {  
 setState(() {  
 if(mainScreen.type){  
 widget.canva.resol++;  
 }  
 else if (!mainScreen.type && widget.canva.depth > 0) {  
 widget.canva.depth--;  
 }  
 });  
 },  
 icon: Icon(  
 Icons.remove,  
 color: Colors.white,  
 ),  
 ),  
 ),  
 SizedBox(  
 width: 20,  
 ),  
 Expanded(  
 child: IconButton(  
 style: kButtonStyle,  
 onPressed: () {  
 setState(() {  
 if(mainScreen.type&& widget.canva.resol>1){  
 widget.canva.resol--;  
 }  
 else if (!mainScreen.type) {  
 widget.canva.depth++;  
 }  
 });  
 },  
 icon: Icon(  
 Icons.add,  
 color: Colors.white,  
 ),  
 ),  
 ),  
 ],  
 ),  
 ),  
 ),  
 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(  
 mainScreen.type?'Size: ${widget.canva.coord}':  
 'Lenght: ${widget.canva.lenght}',  
 style: kSubtitle,  
 ),  
 ),  
 ),  
 Expanded(  
 child: IconButton(  
 style: kButtonStyle,  
 onPressed: () {  
 setState(() {  
 if (!mainScreen.type && widget.canva.lenght > 1) {  
 widget.canva.lenght --;  
 }  
 else if(mainScreen.type && widget.canva.coord > 0){  
 widget.canva.coord\*=1.5;  
 }  
 });  
 },  
 icon: Icon(  
 Icons.remove,  
 color: Colors.white,  
 ),  
 ),  
 ),  
 SizedBox(  
 width: 20,  
 ),  
 Expanded(  
 child: IconButton(  
 style: kButtonStyle,  
 onPressed: () {  
 setState(() {  
 if (!mainScreen.type) {  
 widget.canva.lenght ++;  
 }  
 else if(mainScreen.type){  
 widget.canva.coord/=1.5;  
 }  
 });  
 },  
 icon: Icon(  
 Icons.add,  
 color: Colors.white,  
 ),  
 ),  
 ),  
 ],  
 ),  
 ),  
 ),  
 if(!mainScreen.type)  
 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(  
 'Angle:',  
 style: kSubtitle,  
 ),  
 ),  
 ),  
 Expanded(  
 child: IconButton(  
 style: kButtonStyle,  
 onPressed: () {  
 setState(() {  
 widget.canva.angle-=10;  
 });  
 },  
 icon: Icon(  
 Icons.subdirectory\_arrow\_left\_rounded,  
 color: Colors.white,  
 ),  
 ),  
 ),  
 SizedBox(  
 width: 20,  
 ),  
 Expanded(  
 child: IconButton(  
 style: kButtonStyle,  
 onPressed: () {  
 setState(() {  
 widget.canva.angle+=10;  
 });  
 },  
 icon: Icon(  
 Icons.subdirectory\_arrow\_right\_rounded,  
 color: Colors.white,  
 ),  
 ),  
 ),  
 ],  
 ),  
 ),  
 ),  
 Expanded(  
 child: Container(  
 padding: EdgeInsets.all(5),  
 margin: EdgeInsets.all(5),  
 decoration: kFrame,  
 child: Row(  
 children: [  
 Expanded(  
 flex: 1,  
 child: Center(  
 child: Text(  
 mainScreen.type?'Constant:':  
 'Start:',  
 style: kSubtitle,  
 ),  
 ),  
 ),  
 Expanded(  
 child: CoordField(  
 hintText: mainScreen.type?'Enter a:':  
 'Enter x',  
 onChanged: (val) {  
 setState(() {  
 if(mainScreen.type){  
 widget.canva.ca=val;  
 }  
 else{  
 widget.canva.x = val;  
 }  
 });  
 },  
 )),  
 Expanded(  
 child: CoordField(  
 hintText: mainScreen.type?'Enter b:':  
 'Enter y',  
 onChanged: (val) {  
 setState(() {  
 if(mainScreen.type){  
 widget.canva.cb=val;  
 }  
 else{  
 widget.canva.y = val;  
 }  
 });  
 },  
 )),  
 ],  
 ),  
 ),  
 ),  
 Expanded(  
 child: Container(  
 padding: EdgeInsets.all(5),  
 margin: EdgeInsets.all(5),  
 decoration: kFrame,  
 child: 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: widget.canva.startColor,  
 onColorChanged: (value) {  
 setState(() {  
 widget.canva.startColor = value;  
 });  
 },  
 ),  
 ),  
 actions: <Widget>[  
 ElevatedButton(  
 child: const Text(  
 'Got it',  
 style: TextStyle(  
 color: Color(0xFF303F9F),  
 ),  
 ),  
 onPressed: () {  
 Navigator.of(context).pop();  
 },  
 ),  
 ],  
 );  
 },  
 );  
 },  
 ),  
 ),  
 Expanded(  
 child: Container(  
 width: 50.0,  
 height: 50.0,  
 decoration: BoxDecoration(  
 shape: BoxShape.circle,  
 color: widget.canva.startColor,  
 ),  
 ),  
 ),  
 ],  
 ),  
 ),  
 ),  
 ]);  
 }  
}

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 'dart:typed\_data';  
import 'dart:ui' as ui;  
import 'dart:ui';  
import 'dart:io';  
import 'package:path/path.dart' as path;  
  
import 'package:flutter/material.dart';  
import 'package:flutter/rendering.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});  
 static bool type = false;  
 @override  
 State<mainScreen> createState() => \_mainScreenState();  
}  
  
class \_mainScreenState extends State<mainScreen> {  
 late myCanva canva;  
  
 var styleDrago = kButtonStyleActive, styleJulia = kButtonStyle;  
  
 @override  
 void initState() {  
 super.initState();  
 canva = myCanva();  
 }  
  
 Future<void> getImage(myCanva canva) async {  
 final PictureRecorder recorder = PictureRecorder();  
 canva.paint(Canvas(recorder), canva.mSize);  
 final Picture picture = recorder.endRecording();  
  
 ui.Image image = await picture.toImage(  
 canva.mSize.width.toInt(), canva.mSize.height.toInt());  
 ByteData? byteData = await image.toByteData(format: ui.ImageByteFormat.png);  
  
 if (byteData != null) {  
 final String filePath =  
 path.join('D:/user/Downloads', 'canvas\_image.png');  
  
 final File file = File(filePath);  
 await file.writeAsBytes(byteData.buffer.asUint8List(), flush: true);  
  
 print("Image saved at: $filePath");  
 }  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 body: Column(  
 children: [  
 Row(mainAxisAlignment: MainAxisAlignment.spaceAround, children: [  
 Expanded(  
 flex: 10,  
 child: Container(  
 alignment: Alignment.center,  
 padding: EdgeInsets.all(2),  
 child: SizedBox(  
 child: DefaultTextStyle(  
 child: AnimatedTextKit(  
 animatedTexts: [TypewriterAnimatedText('fractals')],  
 ),  
 style: TextStyle(  
 color: Color(0xFF212121),  
 fontSize: 40.0,  
 fontWeight: FontWeight.w800,  
 ),  
 ),  
 ),  
 ),  
 ),  
 Expanded(  
 child: Container(  
 padding: EdgeInsets.all(3),  
 child: TextButton(  
 style: styleJulia,  
 onPressed: () {  
 setState(() {  
 styleJulia = kButtonStyleActive;  
 styleDrago=kButtonStyle;  
 mainScreen.type = true;  
 });  
 },  
 child: Text(  
 'julia',  
 style: kTextStyle,  
 ),  
 ),  
 ),  
 ),  
 Expanded(  
 child: Container(  
 padding: EdgeInsets.all(3),  
 child: TextButton(  
 style: styleDrago,  
 onPressed: () {  
 setState(() {  
 styleJulia = kButtonStyle;  
 styleDrago=kButtonStyleActive;  
 mainScreen.type = false;  
 });  
 },  
 child: Text(  
 'drago',  
 style: kTextStyle,  
 ),  
 ),  
 ),  
 ),  
 Expanded(  
 child: Container(  
 padding: EdgeInsets.all(3),  
 child: IconButton(  
 style: kButtonStyle,  
 onPressed: () {  
 //canva.saveImage();  
 getImage(canva);  
 },  
 icon: Icon(  
 Icons.save\_alt\_outlined,  
 color: Colors.white,  
 ),  
 ),  
 ),  
 ),  
 ]),  
 Expanded(  
 child: Row(  
 children: [  
 Expanded(  
 flex: 3,  
 child: Listener(  
 onPointerDown: (PointerDownEvent event){  
 setState(() {  
 canva.mouseX=event.localPosition.dx;  
 canva.mouseY=event.localPosition.dy;  
 });  
 },  
 child: Container(  
 padding: EdgeInsets.all(5),  
 margin: EdgeInsets.all(5),  
 decoration: kFrame,  
 child: ClipRect(  
 child: CustomPaint(  
 size: Size(1000, 1000),  
 painter: canva,  
 ),  
 ),  
 ),  
 ),  
 ),  
 Expanded(  
 flex: 2,  
 child: Container(  
 child: ControllPanel(  
 canva: canva,  
 ),  
 ),  
 )  
 ],  
 ),  
 ),  
 ],  
 ),  
 );  
 }  
}

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**

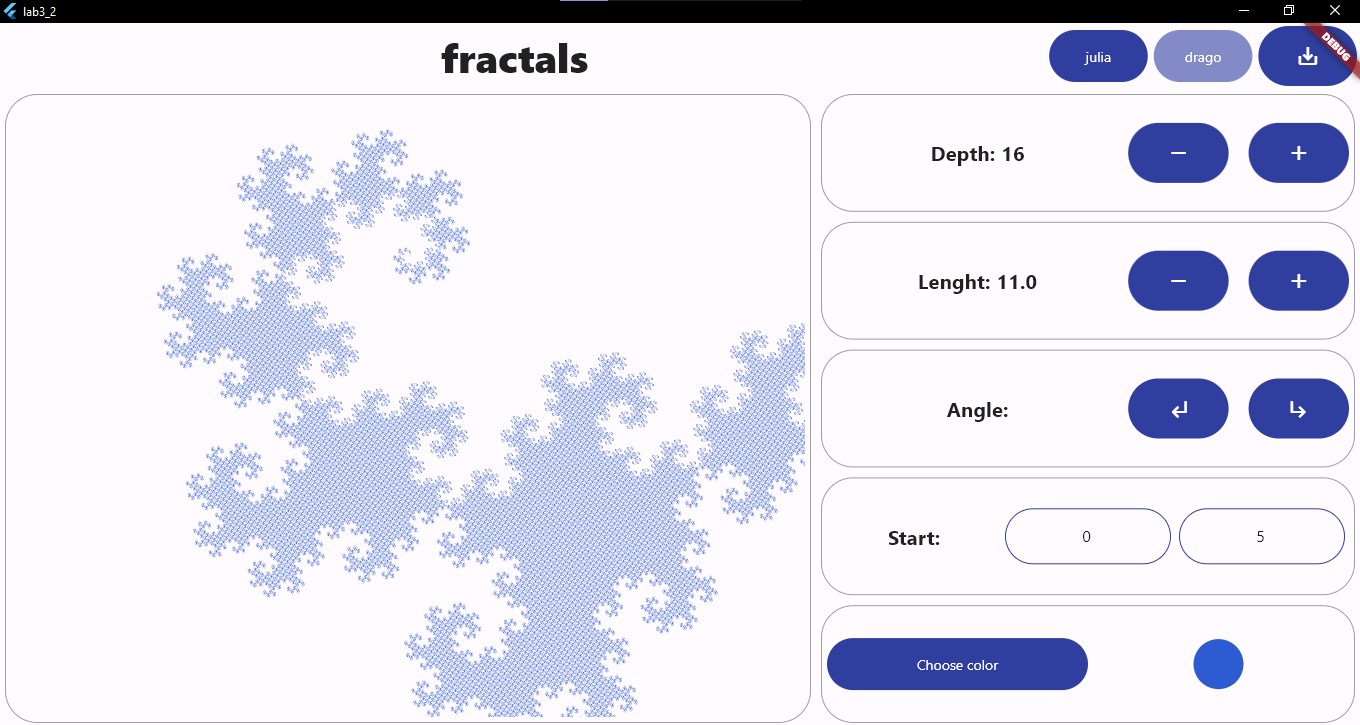
****

Рис. 1. Фрактал Дракон Хартера-Хейтуея

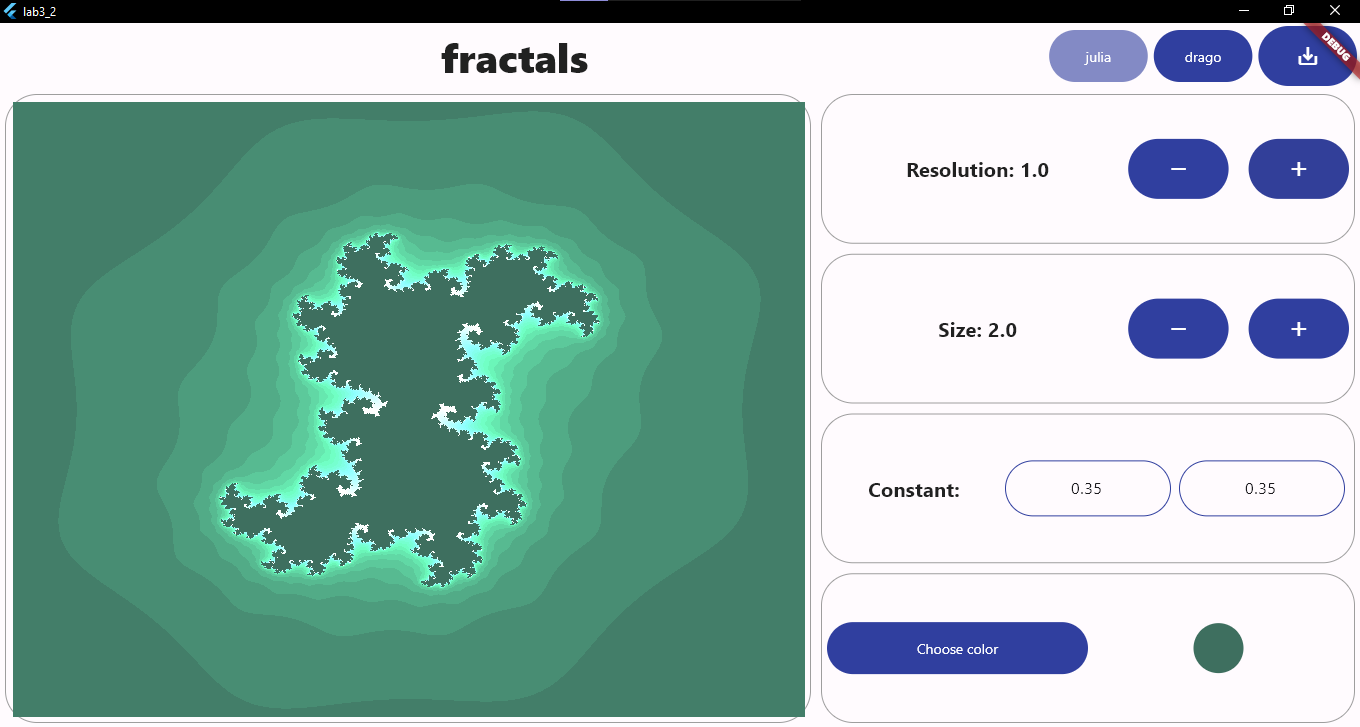


Рис. 2. Фрактал Жюлія

**Висновки**

Під час виконання лабораторної роботи я поглибила свої знання про фрактали. Вивчила принцип побудови геометричних і алгебраїчних фракталів. Дослідила глибше тему пов’язану з множиною Мандельброта та фракталом Жюлія, зв’язок одного та іншого та навчилася зображати їх.