Temă: mulţimi Fatou, mulţimi Julia

1. Clasa JuliaGreen. Să considerăm clasa JuliaGreen dată de exemplu la curs pentru metoda locului final şi să schimbăm numărul de iteraţii `nrIter = 100` în `nrIter = 101`.

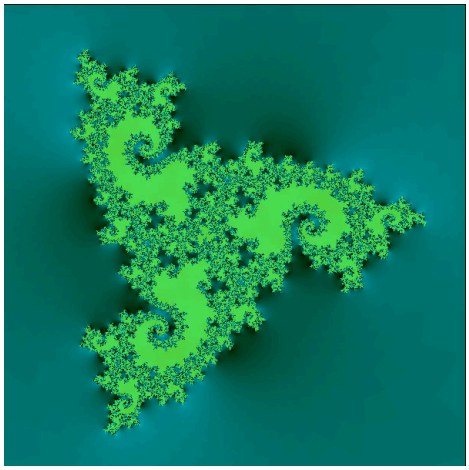

FIGURA 1. `Class JuliaGreen, nrIter = 100`

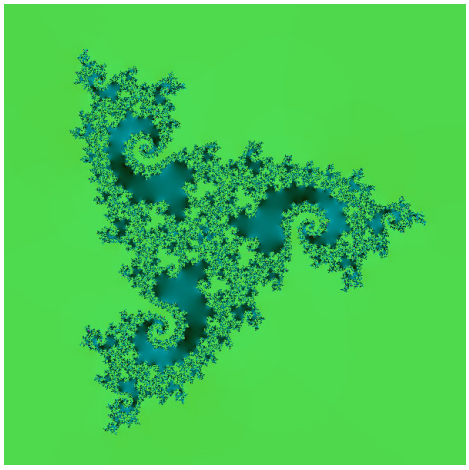

FIGURA 2. `Class JuliaGreen, nrIter = 101`

```
public class JuliaGreen : ComplexForm
{
    Complex i = new Complex(0, 1);
    Complex c = new Complex(0.45, 0.2);

    Complex f(Complex z)
    {
        Complex u = z * z * z;
        return (u + c) / (u - c);
    }
```

```
    public override void makeImage()
    {
        setXminXmaxYminYmax(-2.7, 3.5, -3.1, 3.1);
        Color col;
        Complex z;
        int k, nrIter = 101;
        foreach (Pixel px in bmpPixels)
        {
            z = getZ(px);
            for (k = 0; k < nrIter; k++)
            {
                z = f(z);
                if (z.Ro2 > 1.0e20) break;
            }
            if (z.Ro2 > 1.0e10) col = Color.Black;
            col = getColor(Math.Abs(getI(z.Re)) +
            Math.Abs(getJ(z.Im)) + 1);
            setPixel(px, col);

            if (px.J == 0 && !resetScreen()) return;
        }
    }
}
```

a) Ce se poate deduce din faptul că cele două nuanţe de verde îşi schimbă locurile între ele?

b) Să setăm `nrIter = 1000`. Explicaţi de ce acum, în comparaţie cu cazul `nrIter = 100`, au dispărut umbrele din zona verde închis.

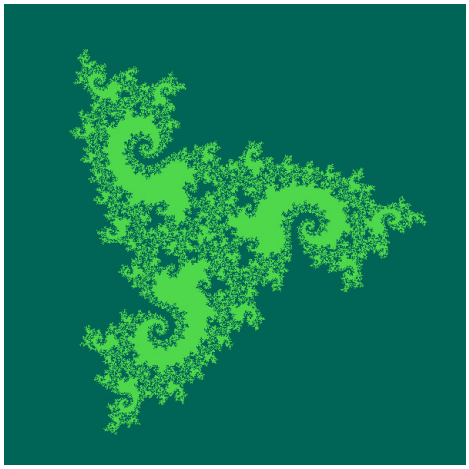

FIGURA 3. `Class JuliaGreen, nrIter = 1000`

2. Clasa JuliaBazine. Să considerăm clasa

```
public class JuliaBazine : ComplexForm
{
    Complex i = new Complex(0, 1);
    Complex c = new Complex(-0.21, -0.7);

    Complex f(Complex z) { return z * z + c; }
```

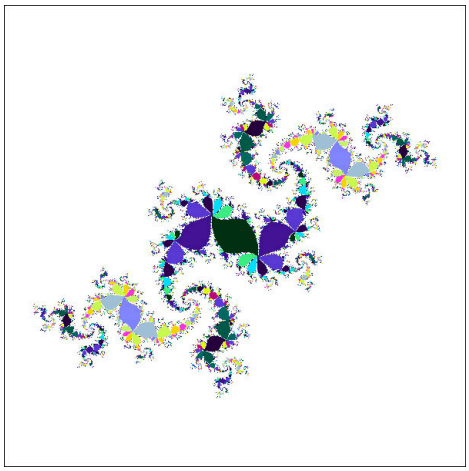

FIGURA 4. Class JuliaBazine

```
    public override void makeImage()
    {
        setXminXmaxYminYmax(-1.5, 1.5, -1.5, 1.5);
        Color col;
        int nrIter = 1001;
        foreach (Pixel px in bmpPixels)
        {
            Complex z = getZ(px);
            for (int k = 0; k < nrIter; k++)
            {
                z = f(z);
                if (z.Ro2 > 1.0e10) break;
            }
            if (z.Ro2 > 1.0e9) col = Color.White;
            else
            {
                col = getColor(10 * Math.Abs(getI(z.Re)) +
                10 * Math.Abs(getJ(z.Im)) + 200);
            }
            setPixel(px, col);
            if (px.J==0 && !resetScreen()) return;
        }
    }
}
```

care colorează prin metoda locului final punctele Fatou ale funcţiei $f_c(z) = z^2 + c$, cu $c = -0.21 - 0.70i$. Sunt puse în evidenţă componentele conexe ale mulţimii Fatou, componenta albă corespunde punctului de la infinit, care în acest caz este un punct fix atractiv, iar componentele colorate corespund unei orbite periodice atractive.

Incercaţi să determinaţi, variind numărul de iteraţii `nrIter`, câte puncte distincte are această orbită periodică.

3. Clasa JuliaPlina2. Clasa următoare

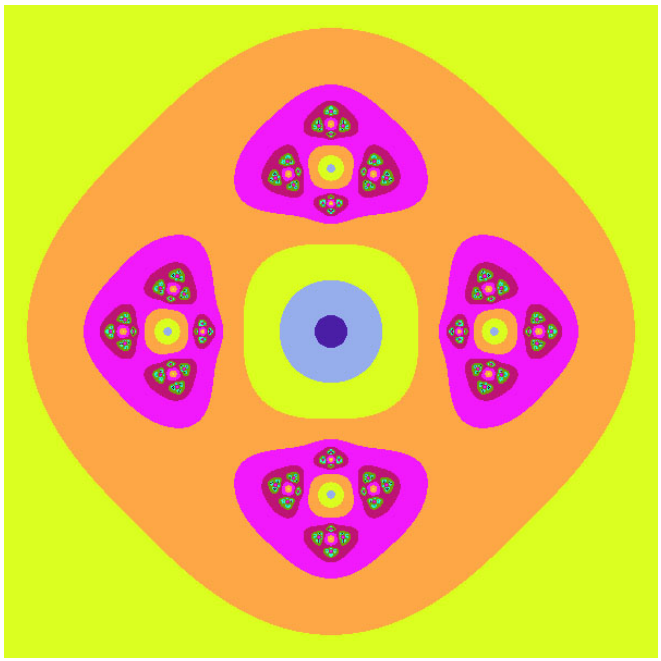

FIGURA 5. Class JuliaPlina2

```
public class JuliaPlina2 : ComplexForm
{
    Complex f(Complex z)
    {
        Complex u = z * z;
        return u - 1 / u;
    }
    public override void makeImage()
    {
        setXminXmaxYminYmax(-2, 2, -2, 2);
        Complex z;
        int k, nrIter = 1001;
        foreach (Pixel px in bmpPixels)
        {
            z = getZ(px);
            for (k = 0; k < nrIter; k++)
            {
                z = f(z);
                if (z.Ro2 > 10000) break;
            }
            setPixel(px, getColor(100 * k));
            if (px.J == 0 && !resetScreen()) return;
        }
    }
}
```

reprezintă grafic mulţimea Julia plină ataşată funcţiei $f(z) = z^2 - 1/z^2$. Incercaţi să explicaţi autosimilaritatea evidentă care apare în figura obţinută.

4. Clasa JuliaSierpinski. In figura 6 este reprezentată cu negru mulţimea

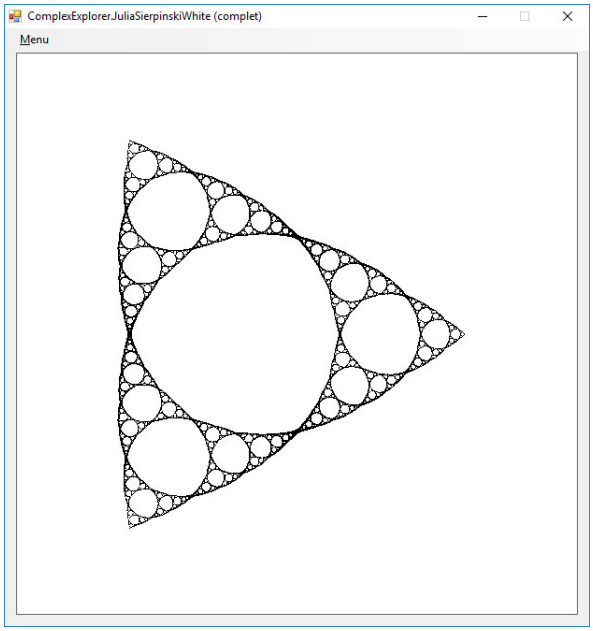

FIGURA 6. `Class JuliaSierpinskiWhite`

Julia ataşată funcţiei $f(z) = \frac{2(z^3-2)}{3z}$ ca frontieră a bazinului de atracţie al punctului de la infinit, bazin colorat cu alb. Incercaţi să implementaţi o clasă `JuliaSierpinski` care să traseze acest desen. Adăugaţi mai multe culori:

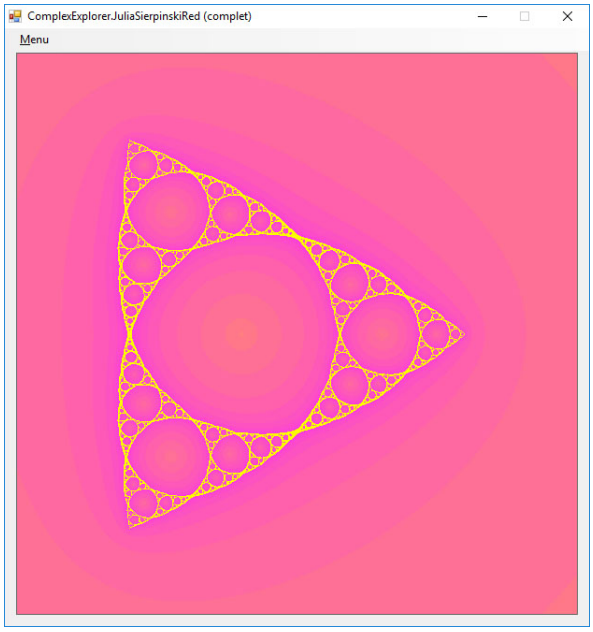

FIGURA 7. `Class JuliaSierpinskiRed`

5. Clasa JuliaNautilus. Adaptaţi clasa `JuliaRandom` pentru a reprezenta mulţimea Julia ataşată funcţiei

$$f(z) = \frac{z^3 - z}{\omega z^2 + 1}$$

pentru $\omega = 1.001(\cos\frac{\pi}{12} + i\sin\frac{\pi}{12})$.

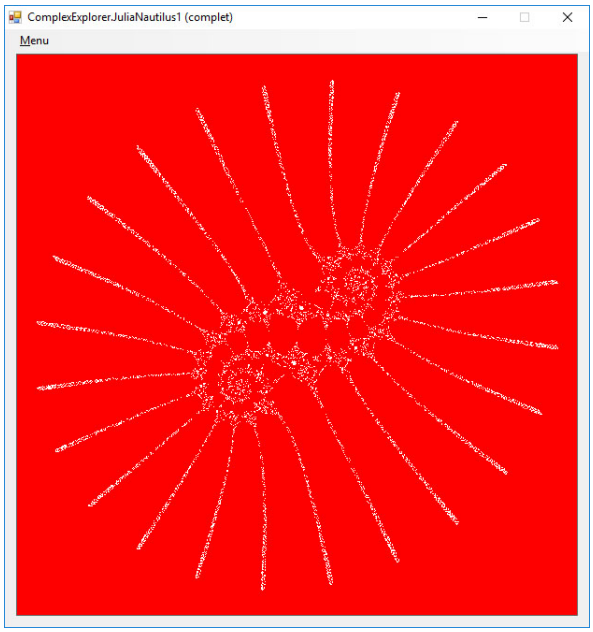

FIGURA 8. `Class JuliaNautilus1`

Incercaţi apoi să analizaţi acest exemplu prin metoda locului final:

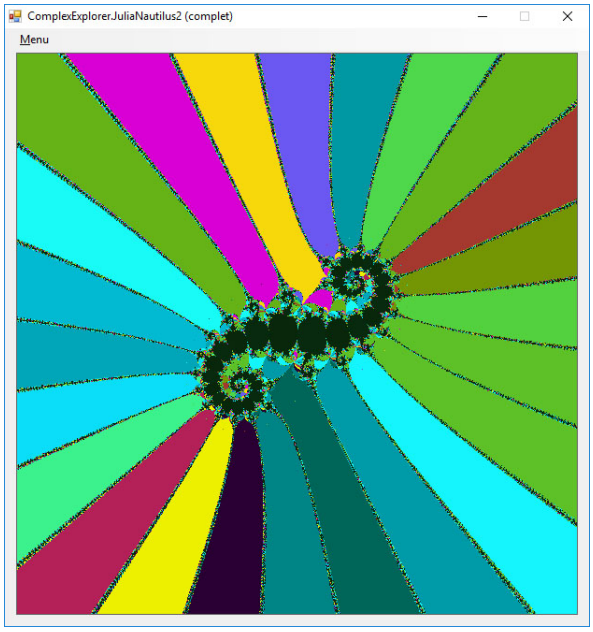

FIGURA 9. `Class JuliaNautilus2`

6. Clasa JuliaFinal. Funcţia `makeImage()` următoare colorează prin metoda locului final punctele Fatou ale funcţiei $f(z) = (z^3+c)/z$ pentru $c = 0.001$. Aflaţi semnificaţia celor patru culori.

```
public class JuliaFinal : ComplexForm
{

    static Complex c = 0.001;
    Complex f(Complex z)
    {
        return (z * z * z + c) / z;
    }
    public override void makeImage()
    {
        setXminXmaxYminYmax(-1.1, 1.1, -1.1, 1.1);
        double infinit = 1.0e10;
        int  nrIter = 106;
        foreach (Pixel px in bmpPixels)
        {
            Complex z = getZ(px);
            for (int k = 0; k < nrIter && z.Ro2 < infinit; k++)
            {
                z = f(z);
            }
            Color col = Color.Red;
            if (z.Ro2 < infinit) col= getColor(Math.Abs(getI(25 * z.Re) +
            getJ(15 * z.Im) + 650));
            setPixel(px, col);
            setPixel(z, col);
            if (px.J == 0 && !resetScreen()) return;
        }
    }
}
```

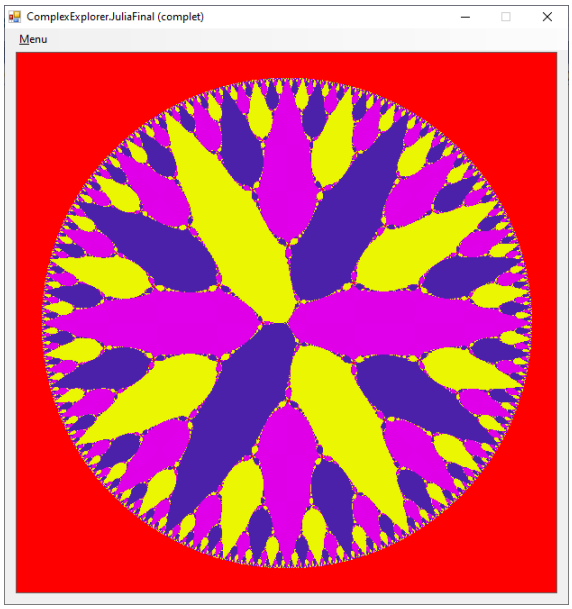

FIGURA 10. Class JuliaFinal

7. Clasa Glynn. Desenaţi mulţimea Julia plină asociată lui $f(z) = z^{1.5} + c$, pentru $c = -0.2$.

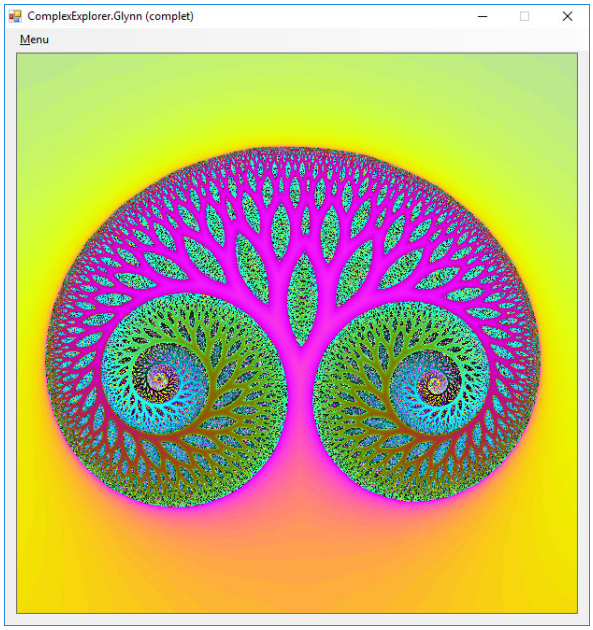

FIGURA 11. `Class Glynn`

Pentru calculul puterii z^α cu $z \in \mathbb{C}$ şi $\alpha \in \mathbb{R}$, folosiţi determinarea principală a funcţiei putere

$$z^\alpha = [\rho(\cos\theta + i\sin\theta)]^\alpha = \rho^\alpha(\cos\alpha\theta + i\sin\alpha\theta),$$

adică funcţia următoare:

```
Complex pow(Complex z, double alfa)
{
    return Complex.setRoTheta(Math.Pow(z.Ro, alfa),z.Theta*alfa);
}
```

Încercaţi şi pentru $f(z) = z^{1.75} + c$, cu $c = -0.375$.