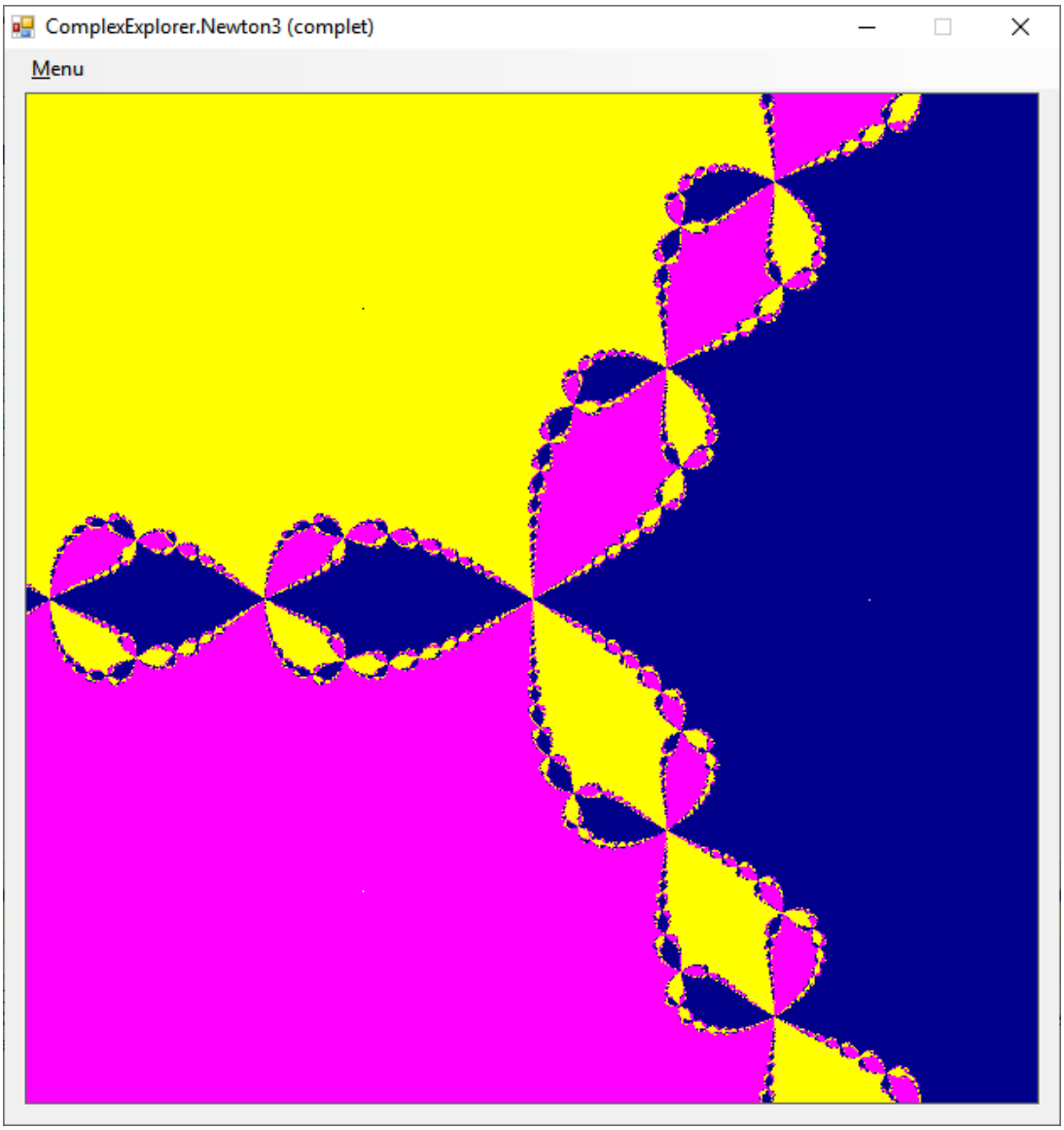
Curs 12

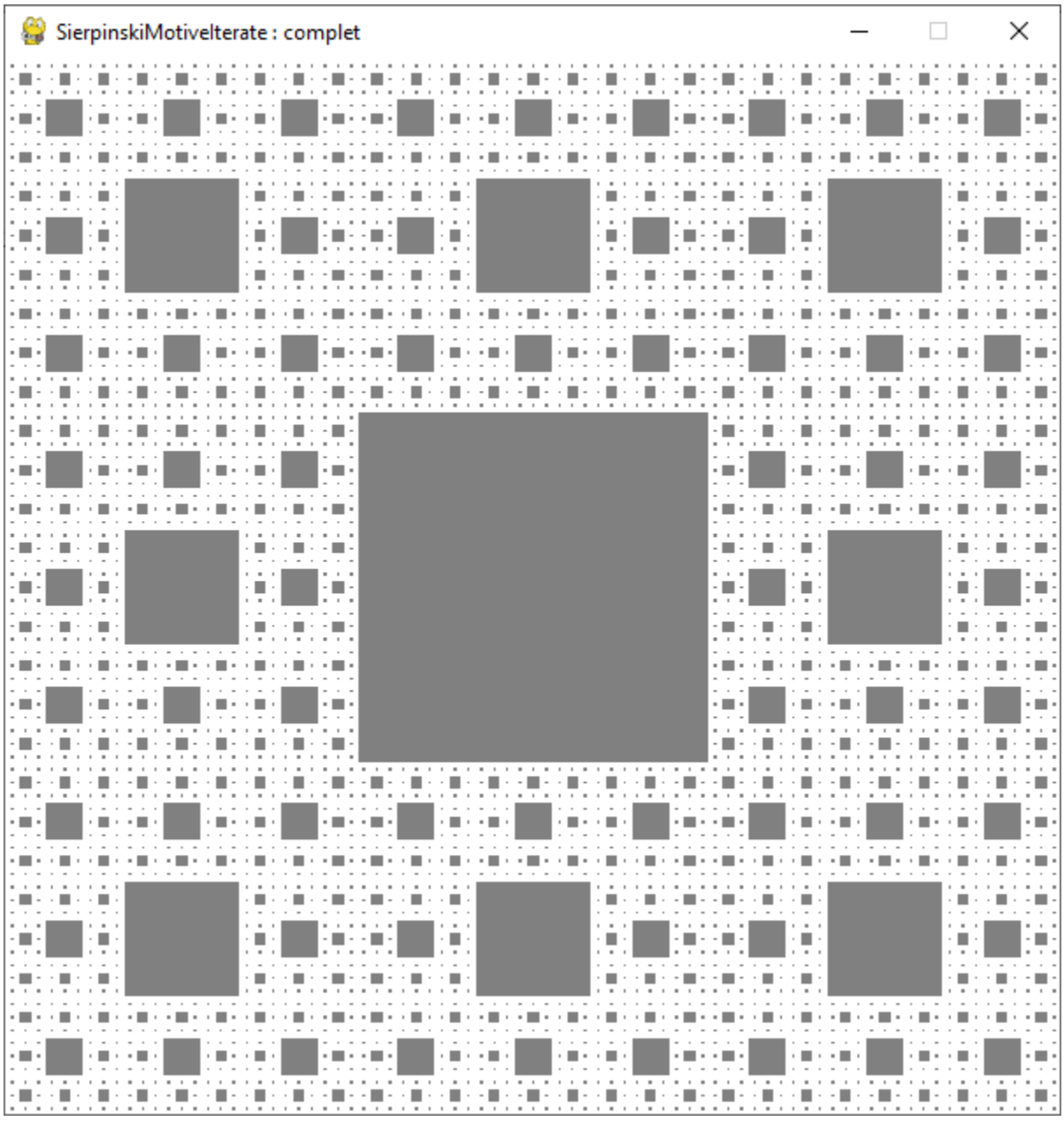
(*plan de curs*)

1. Teorema lui Banach, metoda lui Newton, problema lui Cayley, teorema lui Julia.



```
import ComplexPygame as C
import Color
import math



def Newton3():
    eps0 = C.fromRhoTheta(1.0, 0.0 * math.pi / 3.0)
    eps1 = C.fromRhoTheta(1.0, 2.0 * math.pi / 3.0)
    eps2 = C.fromRhoTheta(1.0, 4.0 * math.pi / 3.0)

    def f(z):
        if z == 0:
            return 1.0e100
        else:
            return (2 * z * z * z + 1) / (3 * z * z)

    C.setXminXmaxYminYmax(-1.5, 1.5, -1.5, 1.5)
    nrIter = 300
    for coloana in C.screenColumns():
        for zeta in coloana:
            col = Color.Black
            z = zeta
            for _ in range(nrIter):
                if C.rho(z - eps0) < 0.1:
                    col = Color.Darkblue
                    break
                if C.rho(z - eps1) < 0.1:
                    col = Color.Yellow
                    break
                if C.rho(z - eps2) < 0.1:
                    col = Color.Fuchsia
                    break
                z = f(z)
            C.setPixel(zeta, col)
        if C.mustClose(): return
    C.setAxis(Color.White)
    C.refreshScreen()




if __name__ == '__main__':
    C.initPygame()
    C.run(Newton3)
```

2. Desene recurente.



```
import ComplexPygame as C
import Color

class Patrat:
    colturi = [-1 - 1j, -1 + 1j, 1 + 1j, 1 - 1j]

    def __init__(self, q, r):
        self.q = q
        self.r = r

    def show(self, col):
        C.fillNgon([self.q + self.r * u for u in Patrat.colturi], col)

    def getXminXmaxYminYmax(self):
        x0, y0 = self.q.real, self.q.imag
        r = self.r
        return x0 - r, x0 + r, y0 - r, y0 + r


def Directii():
    for h in [-1, 0, 1]:
        for k in [-1, 0, 1]:
            if h == 0 and k == 0: continue
            yield complex(h, k)


def traseaza(li, col):
    for p in li:
        p.show(col)


def SierpinskiRecursiv():
    def deseneazaRec(P, niv):
        if C.mustClose(): return
        if niv <= 0: return
        P.show(Color.Index(900 - 50 * niv))
        for u in Directii():
            deseneazaRec(Patrat(P.q + 2 * u * P.r / 3, P.r / 3), niv - 1)

    q0 = 1 + 1j
    r0 = 2
    P = Patrat(q0, r0)
    C.setXminXmaxYminYmax(*P.getXminXmaxYminYmax())
    deseneazaRec(P, 5)
    return
```

```
def SierpinskiMotiveIterate():
    def transforma(li):
        # pentru fiecare patrat P, baza,
        # aplicam motivul de 8 ori
        return [Patrat(P.q + 2 * u * P.r / 3, P.r / 3) for P in li for u in
Directii()]

    q0 = 1 + 1j
    lat0 = 2
    P = Patrat(q0, lat0)
    C.setXminXmaxYminYmax(*P.getXminXmaxYminYmax())
    fig = [P]
    traseaza(fig, Color.Index(650))
    for k in range(6):
        fig = transforma(fig)
        traseaza(fig, Color.Index(650 + 50 * k))
        # C.wait(10)
        if C.mustClose():
            return
    return


def SierpinskiTransformariIterate():
    w0 = 1 + 1j
    r0 = 2
    centre = [w0 + 2 * u * r0 / 3 for u in Directii()]

    def transforma(li):
        # pentru fiecare centru w
        # aplicam transformarea P => Patrat(w + (P.q - w0)/ 3, P.lat / 3))
        return [Patrat(w + (P.q - w0) / 3, P.r / 3) for w in centre for P in li]

    P = Patrat(w0, r0)
    C.setXminXmaxYminYmax(*P.getXminXmaxYminYmax())
    fig = [P]
    traseaza(fig, Color.Index(650))
    for k in range(6):
        fig = transforma(fig)
        traseaza(fig, Color.Index(650 + 50 * k))
        if C.mustClose():
            return
    return
```

```
def SierpinskiTernar():
    # desenam patratul lui Sierpinski
    # prin scriere ternara x=0.ccccccccc

    C.setXminXmaxYminYmax(0, 1, 0, 1)
    for z in C.screenAffixes():
        x, y = z.real, z.imag
        nivMax = 5
        k = 1
        for k in range(nivMax):
            x *= 3
            y *= 3
            ckx = int(x)  # cifra de pe locul k a lui x
            cky = int(y)  # cifra de pe locul k a lui y
            if ckx == 1 and cky == 1:
                break
            x -= ckx
            y -= cky
        C.setPixel(z, Color.Index(650 + 50 * k))
    return


if __name__ == '__main__':
    C.initPygame()
    C.run(SierpinskiRecursiv)
    C.run(SierpinskiMotiveIterate)
    C.run(SierpinskiTransformariIterate)
    C.run(SierpinskiTernar)
```