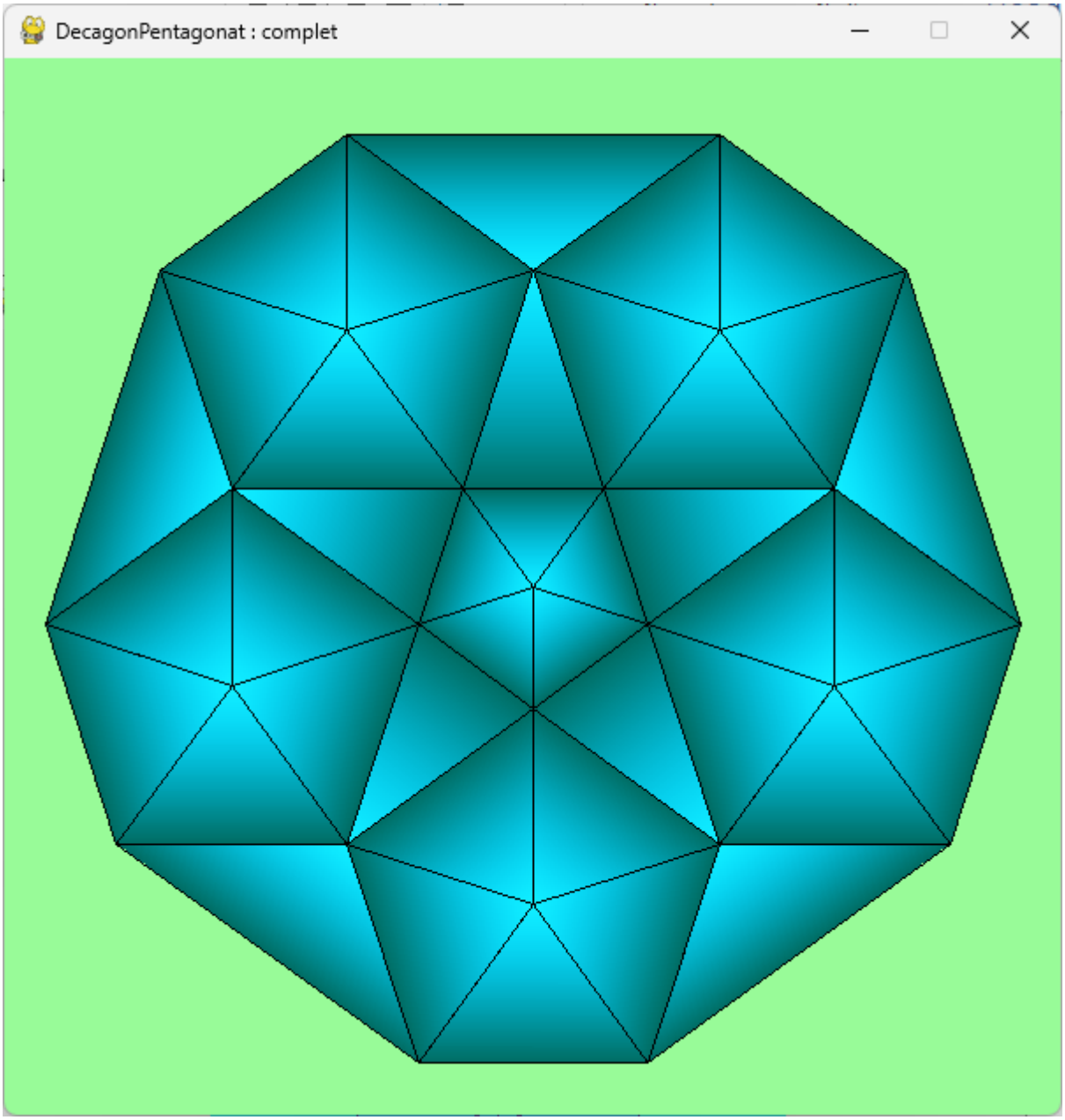
Curs 08

(*plan de curs*)

1. DecagonPentagonat
2. Funcţii analitice



```
import ComplexPygame as C
import Color
import math
def DecagonPentagonat():

    def aria2(a, b, c):
        # dublul ariei triunghiului a, b, c
        return abs(((b - a) * (c - a).conjugate()).imag)

    def umpleTriunghi(a, b, c):
        # coloreaza triunghiul a, b, c
        idx0 = 850
        deltaIdx = -100
        sigmaABC = aria2(a, b, c)
        for z in C.screenAffixes():
            sigmaA = aria2(z, b, c)
            sigmaB = aria2(z, c, a)
            sigmaC = aria2(z, a, b)
            if sigmaA + sigmaB + sigmaC < sigmaABC + 0.001:
                # z este in triunghiul a,b,c
                idx = int(idx0 + sigmaA * deltaIdx / sigmaABC)
                C.setPixel(z, Color.Index(idx))
        C.drawNgon([a, b, c], Color.Black)
        C.refreshScreen()

    def umpleNgon(ngon):
        # coloreaza n-gonul primit
        # q = centrul de greutate
        N = len(ngon)
        q = sum(ngon) / N
        # adaugam primul varf la coada:
        ngon.append(ngon[0])
        for k in range(N):
            umpleTriunghi(q, ngon[k], ngon[k + 1])

    def unNgonQA(q, a, N):
        # returneaza varfurile n-gonului regulat cu centrul q si primul varf a
        # parcurs in sens trigonometric
        alfa = 2.0 * math.pi / N
        return [q + C.fromRhoTheta(1, k * alfa) * (a - q) for k in range(N)]


    C.setXminXmaxYminYmax(0, 10, 0, 10)
    C.fillScreen(Color.Palegreen)
    q = 5+5j
    a = q + C.fromRhoTheta(3, -math.pi / 10)
    decagon = unNgonQA(q, a, 10)
    lstExt = []  # lista exterioara
    lstInt = []  # lista interioara
    for k in range(0, 10, 2):
```

```
            penta = unNgonQA(decagon[k], decagon[k + 1], 5)
            umpleNgon(penta)
            lstExt.append(penta[3])
            lstExt.append(penta[4])
            lstExt.append(penta[0])
            lstInt.append(penta[1])
            lstInt.append(penta[0])
        umpleNgon([lstInt[k] for k in range(0, 10, 2)])

        # schimbam numerotarea varfurilor  din lista exterioara:
        # mutam primul varf la coada
        lstExt.append(lstExt[0])
        del lstExt[0]
        for k in range(1, 15, 3):
            umpleTriunghi(lstExt[k], lstExt[k-1], lstExt[k + 1])

        # completam lista interioara:
        lstInt.append(lstInt[0])
        for k in range(1, 10, 2):
            umpleTriunghi(lstInt[k], lstInt[k - 1], lstInt[k + 1])

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