Wprowadzenie do Grafiki Maszynowej NS
Marcin Głód
164004
grupa II

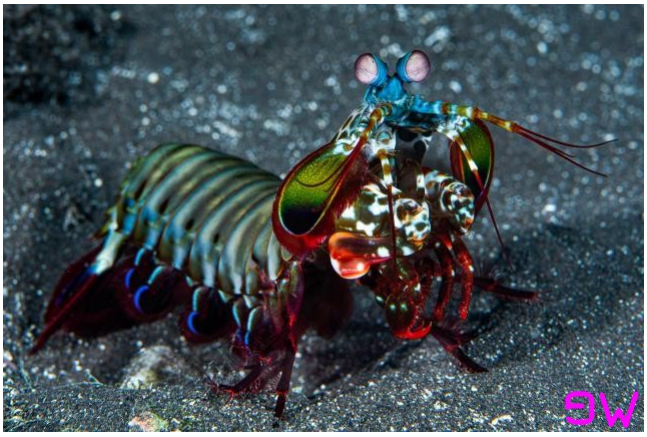
# Laboratorium 5 – sprawozdanie

**Zadanie 1.**

Utworzyłem funkcję *wstaw_inicjaly*, która przylmuje jako parametry obraz, maskę z inicjałami, współrzędne (m,n), oraz wartości nasycenia kolorów r,g,b:

```
def wstaw_inicjaly(obrazek, inicjal, m, n, r, g, b):
    obraz = obrazek.load()
    ini = np.asarray(inicjal)
    w, h  = obrazek.size
    hi, wi = inicjal.size

    for i, j in zakres(hi, wi):
        if 0 == ini[j,i]:
            obraz[w-i,h-j] = (r, g, b)
    obrazek.show()
    return obraz
```

Wynik działania tej funkcji widzimy na poniższym obrazku:



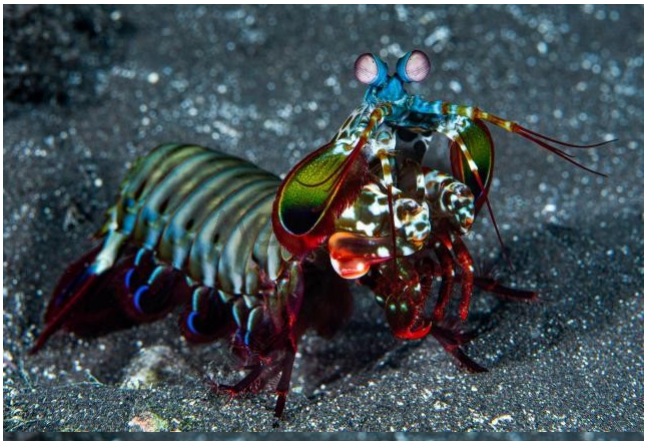
**Zadanie 2.**

Przy użyciu metody load() utworzyłem funkcję filtruj_*inicjaly()*, która na podstawie obrazka, maski z inicjałami , współrzędnych m,n i ustalonego współczynnika stosuje maskę do pikseli odpowiadających inicjałom, by przeprowadzić na nich transformację logarytmiczną.

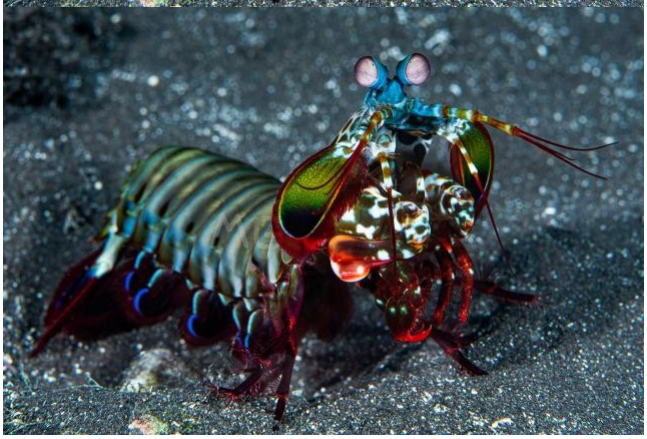
```
def filtruj_inicjaly(obrazek, inicjal, m, n, factor):
    obraz = obrazek.load()
    ini = np.asarray(inicjal)
    w, h  = obrazek.size
    hi, wi = inicjal.size

    for i, j in zakres(wi, hi):
        if 0 == ini[i,j]:
            (r,g,b) = obraz[j+m,i+n]
            r = int(255* np.log(1+(r/factor)))
            g = int(255* np.log(1+(g/factor)))
            b = int(255* np.log(1+(b/factor)))
```

```
            obraz[j+m,i+n] = (r,g,b)
    obrazek.show()
    return obrazek
```

Celem poprawy widoczności maski na moim obrazku, przesunąłem ją w lewo:



Dodatkowo, by poprawić widoczność zmian, zmieniłem współczynnik i wygenerowałem ponownie:



**Zadanie 3.**
Utworzyłem funkcję rysuj_kwadrat_srednia(), o parametrach obraz, współrzędnych (m,n), oraz k, będące szerokością rysowanego kwadratu:

```
def rysuj_kwadrat_srednia(obraz, m,n,k):
    if 0 == k%2:
        print("nieprawidłowa wartość K w rysuj_kwadrat_srednia.
pomijam.")
        return 1
    obrazek = obraz.load()
    zakres = int((k-1)/2)
    print("zakres = ",zakres)
    red = green = blue = 255
    print("green = ",green)

    for i in range (m-zakres, m+zakres):
        for j in range (n-zakres, n+zakres):
            (r,g,b) = obrazek[i, j]
            if red > r:
                red = r
            if green > g:
                green = g
```

```
            if blue > b:
                blue = b

    for i in range (m-zakres, m+zakres):
        for j in range (n-zakres, n+zakres):
            obrazek[i,j] = (red,green,blue)
    print("red = ",red)
    print("green = ",green)
    print("blue = ",blue)
    obraz.show()
    return obraz
```

Następnie wywołałem ją w następujący sposób:

```
obrazek3  = rysuj_kwadrat_srednia(shrimp, 200,200,51)
obrazek31 = rysuj_kwadrat_srednia(obrazek3, 369,69,21)
obrazek32 = rysuj_kwadrat_srednia(obrazek31, 575,75,77)
```

gdzie punkty środkowe to odpowiednio (200,200)-tułów, (369,69)- oko, oraz (575, 75)-tło, a rozmiary kwadratów to 51, 21 oraz 77 pikseli:

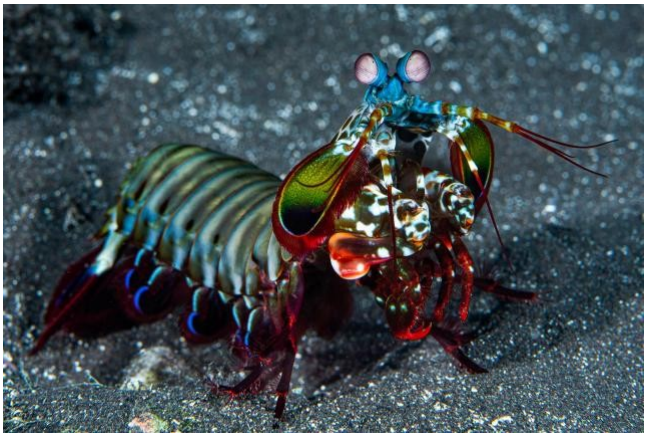
**Zadanie 4.**

Ponieważ o zrobieniu tych zadań przypomniałem sobie w ostatniej możliwej chwili, nie zdążyłem dojść do ładu z zastosowaniem metody .point(), więc napisałem funkcję *kontrast()*, która przyjmuje jako parametr obraz, oraz współczynnik wsp, na podstawie którego obliczana jest zmiana kontrastu:

```
def kontrast(obraz, wsp):
    obrazek = obraz.load()
    mn = (( 255 + wsp)/255)**2
    print("mn = ",mn)
    w,h = obraz.size
    for i,j in zakres(w,h):
        (r,g,b) = obrazek[i,j]
        rr = int(128+(r-128)*mn)
        gg = int(128+(g-128)*mn)
        bb = int(128+(b-128)*mn)
        obrazek[i,j] = (rr,gg,bb)
    obraz.show()
    return obraz
```

Niestety nie udało mi się namierzyć przyczyny, ale pomimo struktury i zapisu identycznego jak w poprzednich raportach, matplotlib umieszczał mi niewłaściwy obraz na wykresie:

„wyrób” matplotliba:
obrazy zapisane oddzielnie odpowiednio dla wartości współczynnika 0, 25 oraz 80:



**Zadanie 5.**
Różnica pomiędzy zastosowaniem numpy i dodaniu do każdej wartości tablicy wartości 100, a zastaosowaniem
obraz.point(lambda i: i+100) wynika z rozbieżności w interpretacji wartości wychodzących powyżej 255: numpy automatycznie „zawija” wartości wykraczające poza zakres uint8.