Współczesne Narzędzia Obliczeniowe
Laboratorium I

Zasady ogólne

Przedmiot składa się z 6 zajęć komputerowych oraz jednego zadania domowego. Na każdych zajęciach można zdobyć od 0 do 5 punktów za zadania standardowe (punkty są typu *integer*), zadania opcjonalne są premiowane dodatkowo oznaczone jako (⋆x pkt). W niektórych zagadnieniach mogą wystąpić punkty ujemne. Co oznacza że można standardowo zdobyć **(35 pkt)**. Prawie każde zadanie (pomijając ostatnie) można donieść najpóźniej tydzień po jego oficjalnym terminie za połowę punktów (zaokrąglając w dół). Próg zaliczenia to: **(22 pkt)** – średnia > 3.0. Na laboratorium **nie wolno** używać gotowych skryptów (dotyczy to zarówno skryptów pobranych z internetu, jak i *pożyczonych* od kolegów). Użycie gotowych skryptów zeruje punktację z danego laboratorium, bez możliwości poprawy. Punktacja za skrypty zbyt podobne do siebie dzielona jest na liczbę osób piszących wspólnie. Laboratorium nie jest poprawiane. Nieobecności należy zgłosić przed zajęciami, zaś zadanie odrobić na równoległych laboratoriach lub donieść po tygodniu za połowę punktów. Materiały na zajęcia można znaleźć pod linkiem: `http://bit.ly/2Sgq1dA`.

1. Cykloidy

Zadanie polega na rysowaniu i animopwaniu odpowiedniej krzywej parametrycznej. Narysowanie krzywej opierającej się na linii prostej **(1 pkt)**, na zadanej przez prowadzącego krzywej **(2 pkt)**. Wczytanie parametrów krzywej za pomocą linii poleceń **(1 pkt)**. Dorobienie animowanej krzywej (animowany promień wodzący) **(1 pkt)**. Brak użycia jakiejkolwiek pętli `for` (⋆1 pkt). Opracowanie wczytywania dowolnej krzywej z wiersza poleceń (⋆2 pkt).

2. Tygrysy

Zadanie polega na implementacji algorytmu znajdowania otoczki wypukłej (*covex hull*). Na początek należy wygenerować losowo 20 punktów (tygrysów) na płaszczyźnie **(1 pkt)**. Następnie zaimplementować algorytm Jarvisa lub Grahama **(2 pkt)**, oraz wpowadzić do niego odpowiednią modyfikację podaną przez prowadzącego. Użycie klasy do implementacji tygrysów (⋆1 pkt). Dodanie aspektu czasu i ruchu tygrysów oraz krokowego rozwijania otoczki (⋆2 pkt).

3. Maze

Zadanie polega na przerobieniu gotowego kodu generującego labirynt, dostępnego na wiki. Adaptacja kodu ma polegać na konwersji ze standardowej kwadratowej wersji pola na zadaną przez prowadzącego np.: trójkątna, hexagonalna, oktagonalna **(3 pkt)**. Dodatkowa implementacja lagorytmu przechodzenia labiryntu **(1 pkt)**.

Maze

```
mport numpy
from numpy.random import randint as rand
import matplotlib.pyplot as pyplot

def maze(width=81, height=51, complexity=.75, density=.75):
    # Only odd shapes
    shape = ((height // 2) * 2 + 1, (width // 2) * 2 + 1)
    # Adjust complexity and density relative to maze size
    complexity = int(complexity * (5 * (shape[0] + shape[1]))) # number of components
    density    = int(density * ((shape[0] // 2) * (shape[1] // 2))) # size of components
    # Build actual maze
    Z = numpy.zeros(shape, dtype=bool)
    # Fill borders
    Z[0, :] = Z[-1, :] = 1
    Z[:, 0] = Z[:, -1] = 1
    # Make aisles
    for i in range(density):
        x, y = rand(0, shape[1] // 2) * 2, rand(0, shape[0] // 2) * 2 # pick a random position
        Z[y, x] = 1
        for j in range(complexity):
            neighbours = []
            if x > 1:             neighbours.append((y, x - 2))
            if x < shape[1] - 2:  neighbours.append((y, x + 2))
            if y > 1:             neighbours.append((y - 2, x))
            if y < shape[0] - 2:  neighbours.append((y + 2, x))
            if len(neighbours):
                y_,x_ = neighbours[rand(0, len(neighbours) - 1)]
                if Z[y_, x_] == 0:
                    Z[y_, x_] = 1
                    Z[y_ + (y - y_) // 2, x_ + (x - x_) // 2] = 1
                    x, y = x_, y_
    return Z

pyplot.figure(figsize=(10, 5))
pyplot.imshow(maze(80, 40), cmap=pyplot.cm.binary, interpolation='nearest')
pyplot.xticks([]), pyplot.yticks([])
pyplot.show()
```

4. Simple image processing

Zadanie polega na zapoznaniu się z biblioteką PIL/Pillow. W tym celu należy wyrysować histogram barw RGB **(2 pkt)** oraz histogram skali szarości (po konwersji obrazu) **(1 pkt)**. Następnie należy przekonwertować obraz do postaci pokazującej kontury (filtr Canny'ego, Sobel'a bądź krzyż Roberts'a) **(2 pkt)**.

5. Advanced image processing

Za pomocą prostych operacji na obrazie należy zlokalizować i wyciąć wskazany przez prowadzącego obiekt. Otoczenie obiektu bounding box'em **(1 pkt)**. Wycięcie bounding box'a **(1 pkt)**, usunięcie tła dookoła obiektu **(3 pkt)**. Zadanie domowe związane z tym zagadnieniem polega na lokalizacji obiektu na różnych obrazach, za każdy obraz **(1 pkt)**. Nie wolno w tym wypadku użyć gotowych metod takich jak np. `TemplateMatching`. Punkty ujemne mogą zostac przyznane za niepotrzebną pętlę `for` np. przy przemiataniu pixel po pixelu **(−1 pkt)**, hardkodowane wartości np. kątów, bezwzględnych ścieżek **(−1 pkt)**, wczytywanie tylko jednego pliku **(−1 pkt)**, pętlę nieskończoną **(−1 pkt)**, tzw. niepotrzebne rzyganie na ekran **(−1 pkt)**. **Jakiekolwiek znamiona plagiatu zerują punktację.**

6. 3d

Zadeanie polega na implementacji algorytmu interpolacji trójwymiarowej. W zadaniu nie wolno używać gotowych metod rozwiązujących zagadnienie. Natomiast można używac metod interpolacji 2d. Obecność na laboratorium **(1 pkt)**, rozwiązanie zadania **(4 pkt)**. Dodatkowo animowanie wyrównywania interpolowanych wartości **(⋆1 pkt)**.